Advantage™ VISION:Inquiry®

Reference Guide

6.5

Computer Associates®

IQHUG065.PDF/D07-101-010

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, the user may print a reasonable number of copies of this documentation for its own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software of the user will have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2003 Computer Associates International, Inc. (CA).

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introducing VISION:Inquiry

What Does VISION:Inquiry Look Like? 1-2
How Do You Use VISION:Inquiry? 1-2
Inquiries 1-4
    Commands 1-4
The Directory 1-5
    Database Definitions 1-6
    Stored Inquiries and Functions 1-6
    Vocabulary 1-6
    User Profiles 1-6

Chapter 2: Databases and Files

Database Concepts 2-1
    Field and Segment Names 2-2
    Multiple Occurrences of Segments 2-4
Structured Databases 2-5
    Root and Dependent Segments and Levels 2-5
    Database Legs 2-6
    Database Access 2-6
Test Databases 2-7
    PLANT Database 2-7
    SKILL Database 2-10
VSAM Files 2-11
    VSAM Test Files 2-12
    VSPLANT File 2-12
VSSKILL File 2-12
    VSAM Hierarchical Test Files 2-14
    VSHPLANT Test File 2-15
    VSHSKILL Test File 2-16
DB2 Concepts 2-16
    Structured Query Language (SQL) 2-17

DB2 Test Tables Not Using the Native SQL Syntax Facility (Names as supplied or Assigned) 2-18
DB2 Test Views Not Using the Native SQL Syntax Facility (Names as Supplied or Assigned) 2-19
DB2 Test Tables Using the Native SQL Syntax Facility (Names for Embedded SQL SELECT) 2-20
DB2 Test Views Using the Native SQL Syntax Facility (Names for Embedded SQL SELECT) 2-21

Chapter 3: Structure and Concept of VISION:Inquiry

Statement Structure 3-1
Separators 3-3
Continuing Statements 3-3
Improving Readability 3-3
Terminating an Inquiry 3-4
Names 3-4
Constants 3-5
Character Constants 3-5
Numeric Constants 3-6
Titles 3-6
Operators 3-7

Chapter 4: Using VISION:Inquiry

The VISION:Inquiry Screen Under IMS 4-2
Entering an Inquiry Under IMS 4-3
Entering a Second Inquiry Under IMS 4-5
The VISION:Inquiry Screen Under CICS 4-6
Basic Mapping Support (BMS) Paging Options 4-8
Entering a Second Inquiry Under CICS 4-9
Signing Off 4-9
Error Handling 4-10
Correcting Errors 4-11
Operating Modes 4-11
Online Inquiry Processing 4-11
Conversational Mode with Checkpoint 4-13
CONTINUE Command 4-13
DEFER Command 4-14
ID Number 4-15
CONTINUE DEFERRED INQUIRY Command 4-15
DELETE DEFERRED INQUIRY Command 4-16
Conversational Mode with UDO 4-16
SHOW Command 4-16
Batch Message Inquiry Processing (IMS only) 4-17
Batch Inquiry Processing 4-17
TSO Inquiry Processing (IMS only) 4-18

Chapter 5: Simple Reports

DISPLAY Command 5-1
Displaying Two Segments 5-3
Displaying Multiple Segments 5-4
Accessing Two legs of a Database 5-6
Vertical Display - Row Mode Formatting 5-8
Horizontal and Vertical Display with Native SQL Syntax 5-9
Displaying Title Lines 5-10
DISPLAY Command Summary 5-11
Conditional Selection Phrase 5-12
The Positional Test 5-14
The Existence Test 5-17
Compound Conditional Selection 5-18
Relational and Logical Operators 5-19
Using the LIKE Operator 5-20
Evaluating Logical Expressions 5-21
Using the AND Operator 5-22
Using the OR Operator 5-22
Using Parentheses 5-23
Selecting Two Legs of a Database 5-26
Conditional Selection Phrase Summary 5-27
SORT Command 5-28
Displaying Fields in Ascending or Descending Sequence 5-28
Using SORT with DISPLAY 5-28
Ascending or Descending Order 5-29
Using the SORT Command 5-29
Comparing Sorted and Unsorted Output 5-31
Using Multiple SORT Commands 5-32
Reversing the Order of the Fields to be Sorted 5-33
Sorting Blank Fields 5-34
Sorting in Descending Order 5-35
Using SORT ASC and SORT DSC 5-36
Using Multiple SORT DSC Commands 5-37
Use of SORT with Subtotals and SUM 5-38
SORT Command Summary 5-38
The LIMIT Command 5-39
Highest Level Segment Illustration 5-40
Leftmost Segment Illustration 5-41
LIMIT Command Applied to Leftmost Segment 5-42
LIMIT Command Applied to Highest Level Segment 5-43
LIMIT Command Summary 5-43
OUTPUT Command 5-44
OUTPUT Command Under IMS 5-44
Using an OUTPUT Command 5-44
Continuing an Inquiry Sent to Another Terminal 5-45

Routing to Another Device 5-45
OUTPUT Command Under CICS 5-46
Using an OUTPUT Command 5-47
Error Message Screen 5-47
Continuing an Inquiry Sent to Another Terminal 5-48
Routing to Another Device 5-48
OUTPUT Command Summary 5-49

Chapter 6: Summarizing Data

TOTAL Command 6-1
TOTAL with Conditional Selection 6-2
Using TOTAL with Numeric Fields Only 6-3
Using TOTAL with Other Commands 6-4
TOTAL Command Summary 6-4
COUNT Command 6-5
Using COUNT with Other Commands 6-6
COUNT Command Summary 6-6
AVERAGE Command 6-7
Using AVERAGE with Other Commands 6-8
AVERAGE Command Summary 6-9
Combining TOTAL, COUNT, and AVERAGE 6-9
Using TOTAL with Two Fields 6-11
TOTAL, COUNT, and AVERAGE Command Summary 6-12
Subtotaling - Using a Group Field or Control Break 6-12
Subtotaling with a TOTAL Command 6-13
Subtotaling a COUNT in a DISPLAY Command 6-14
Subtotaling on AVERAGE and Using Parentheses 6-15
Combining Subtotal Commands 6-16
Using Subtotals with Grand Summaries 6-17
Using SORT with Subtotals 6-18
Subtotal Command Summary 6-18
Subtotal Output 6-18
Grand Summaries with Subtotals 6-18

Chapter 7: Assignment Statement and Arithmetic Processing

Assignment Statement 7-1
Assignment Statement Summary 7-3
Arithmetic Calculations 7-3
Arithmetic Expression 7-4
Arithmetic Operators 7-4
Hierarchy of Operation 7-4
Number of Decimal Places 7-4

Arithmetic Processing with the Assignment Statement 7-5
Arithmetic Calculation in an Assignment Statement 7-5
Displaying Results of an Arithmetic Calculation 7-6
Complex Arithmetic Calculation and a Temporary Field in a Conditional Phrase 7-7
Arithmetic Calculation in a Conditional Phrase 7-8
Using the Results of a Calculation in Another Calculation 7-9
Arithmetic Calculation, Conditional Selection, and Display of Fields 7-10
Arithmetic Processing Summary 7-11
Operators 7-11
Hierarchy of Operations 7-11
SUM Command 7-12
SUM Command (Example 1) 7-13
SUM Command (Example 2) 7-14
SUM and TOTAL in One Inquiry 7-15
Grand Totals using SUM 7-16
Arithmetic Processing with SUM 7-16
Creating an Expression 7-18
Using SUM on Fields from Different Segments 7-18
Use of Parentheses 7-20
Showing Only Group Level Totals 7-22
Using SUM with Temporary Fields 7-23
SUM Command Summary 7-24

Chapter 8: Accessing Multiple Databases and Files

FIND Command 8-1
Using FIND and DISPLAY to Access Two Databases 8-3
Data Selection 8-7
Conditional Selection in a FIND 8-8
Expanding the Conditional Selection in a DISPLAY Statement 8-8
Expanded Conditional Selection in FIND and DISPLAY Statements 8-9
Sequencing Data Output 8-11
SORT in Related DISPLAY Segment 8-12
Using FIND with an Arithmetic Calculation 8-13
Testing the Results of an Arithmetic Calculation in a FIND 8-14
Using a FIND Arithmetic Calculation in a DISPLAY Arithmetic Calculation 8-15
Testing a Result Field in a DISPLAY Statement 8-16
Calculation from a FIND Statement Used in a Calculation in Conditional Selection Phrase 8-17
Accessing More Than Two Databases 8-18
Accessing the Same Database More Than One Time 8-20
Any-All Conditional Selection 8-20
Selection on Sibling Segments 8-21
Using FIND with a Manufacturing Database 8-21
Accessing VSAM Files and an IMS (DL/I) Database 8-23

Accessing a VSAM FIle, an IMS (DL/I) Database, and another VSAM FIle 8-24
Accessing VSAM Files and DB2 Tables 8-25
Accessing DB2 Tables and an IMS (DL/I) Database 8-27
Using FIND with the LIMIT Command 8-28
FIND Command Summary 8-29

Chapter 9: Using Directory Commands - Storing Inquiries and Functions

DEFINE Command 9-2
  Storing Inquiries with the DEFINE Command 9-2
    Using DEFINE DIRECTORY INQUIRY to store Inquiries without SQL 9-2
    Using DEFINE DIRECTORY INQUIRY to store Inquiries with SQL 9-2
    DEFINE Command with a Comment 9-4
    Defining Inquiries with Substitutable Values 9-5
    Replacing a Stored Inquiry 9-7
  Storing Functions 9-8
    Using the DEFINE DIRECTORY FUNCTION to Store a Function 9-9
    Storing a Function with No Column Heading 9-10
  Storing Inquiries that Call Stored Functions 9-11
  Storing an Inquiry with Two Stored Functions and a Substitutable Value 9-12
  Calling an Undefined Function 9-13
  Storing Inquiries with the FIND Command 9-14
  DEFINE Command Summary - Inquiries Without SQL 9-14
  DEFINE Command Summary - Inquiries With SQL 9-16
DISPLAY DIRECTORY Command 9-18
  DISPLAY DIRECTORY INQUIRY (PDI) Command - Inquiries Without SQL 9-18
  DISPLAY DIRECTORY INQUIRY (PDI) Command - Inquiries with SQL 9-19
  Using DISPLAY DIRECTORY FUNCTION (PDF) to List a Stored Function 9-20
  Using DISPLAY DIRECTORY to List All Inquiries or Functions 9-20
    Displaying All Inquiries Stored for a Database 9-21
    Displaying All Functions Stored for a Database 9-22
  Using DISPLAY DIRECTORY to List All Inquiries - with SQL 9-22
  DISPLAY DIRECTORY Command Summary - Inquiries Without SQL 9-23
  DISPLAY DIRECTORY Command Summary - Inquiries With SQL 9-23
  Display Stored Inquiries for All Databases 9-24
  Display Stored Inquiry Names and Their Associated Comments 9-25
  Display Comments Command Summary – Inquiries without SQL 9-25
  Display Comments Command Summary – Inquiries with SQL 9-26
DELETE DIRECTORY Command 9-26
  DELETE DIRECTORY INQUIRY Command - Inquiries with SQL 9-27
  Using DELETE DIRECTORY FUNCTION to Delete Stored Functions 9-27
  DELETE Command Summary - Inquiries Without SQL 9-29
  DELETE Command Summary - Inquiries with SQL 9-29

Chapter 10: Using Stored Inquiries and Functions

Executing a Stored Inquiry ... 10-1
Executing a Stored Inquiry with SQL ... 10-2
  Using Substitutable Values in Stored Inquiries ... 10-3
    Using Different Substitutable Values in Subsequent Runs ... 10-5
  Using COUNT and TOTAL in a Stored Inquiry ... 10-6
  Using Stored Inquiries with LIMIT and OUTPUT ... 10-7
  Using Stored Inquiries Summary - Inquiries Without SQL ... 10-8
  Using Stored Inquiries Summary - Inquiries with SQL ... 10-8
Text Editor Facility ... 10-9
  EDITSQ Command ... 10-9
    Using EDITSQ with Inquiries Containing SQL ... 10-9
  Displaying Inquiries for Editing ... 10-10
  Text Editor Screen Layout ... 10-11
  Text Editor PF/PA Keys and Their Functions ... 10-13
    Returning to the Text Editor Screen After an Error ... 10-19
  Text Editor Primary Commands ... 10-20
  Primary Commands Equivalent to PF/PA Key Functions ... 10-21
  Text Editor Line Commands ... 10-22
    Copying a Number of Lines ... 10-23
    Inserting Blank Lines ... 10-24
    Combining the D and I Line Commands ... 10-24
    Entering an Incorrect Line Command ... 10-25
Using Stored Functions ... 10-25
  Using More Than One Stored Function ... 10-27
  Using a Stored Inquiry that Calls a Stored Function ... 10-28
  Using a Stored Inquiry with Two Stored Functions and a Substitutable Value ... 10-29
  Substituting a Value in a Stored Inquiry that Calls Two Functions ... 10-30
  Using Stored Functions Summary ... 10-30

Chapter 11: User Defined Output

Overview of UDO Functions ... 11-1
Using the FORMAT Command Modifier ... 11-2
What is the Syntax of a UDO Inquiry? ... 11-3
Is UDO Compatible with Basic VISION:Inquiry? ... 11-4
How is the Screen Displayed? ... 11-4
What Does the UDO Screen Look Like Under IMS? ... 11-5
Titling Your Report ... 11-6
  Centering the Title ... 11-7
  Including an Apostrophe in a Title ... 11-8

Horizontal Movement Commands - COL and SP 11-9
Using SP without an Integer 11-10
Omitting a Horizontal Movement Command in the Middle of an Inquiry 11-11
Overlaping Fields 11-12
No Space Between the Command and the Integer 11-13
Vertical Movement Commands 11-14
Using the SKIP Command to Add Blank Lines 11-14
Using LINE and SKIP 11-15
Using LINE as a Page Break 11-16
Page Breaks Affect Printing of Literals 11-17
Using DATE, DATEF, TIME, and PAGE 11-18
Customizing Output Format 11-19
Conditional Selection 11-20
Using Conditional Selection in a Formatted Inquiry 11-20
Entering the Same Inquiry Differently 11-21
Using SP Instead of COL 11-22
REPEAT Command 11-22
Using the REPEAT Command 11-23
Using Two REPEAT Commands 11-24
Using REPEAT Commands with Conditional Selection 11-25
Using the REPEAT Command on Dependent Segments 11-26
Partial Fielding (PF) Command 11-27
Using the NOSP Command 11-28
Grand Summary Commands 11-29
Using Grand Summaries in a UDO Inquiry 11-29
Subtotals 11-30
Using Subtotals and Grand Summaries in a UDO Inquiry 11-31
EDIT Command 11-32
Using the Edit Specification N with the EDIT Command 11-33
Using the Edit Specification F with the EDIT Command 11-33
Using the Edit Specification Z with the EDIT Command 11-34
Using EDIT Only with Numeric Field Names 11-34
Arithmetic Processing 11-35
Using an Arithmetic Calculation in a Formatted Inquiry 11-35
Using a Complex Calculation and a Temporary Field with UDO 11-36
FIND Command 11-37
Using a FIND Command in a Formatted Inquiry 11-39
Storing Inquiries 11-40
Defining a Formatted Stored Inquiry 11-40
Using a Formatted Stored Inquiry 11-41
Substitutable Values 11-42
Defining a Formatted Stored Inquiry with Substitutable Values 11-42
Using Substitutable Values in a Formatted Stored Inquiry 11-42
Substituting Different Values in the Same Formatted Stored Inquiry 11-43

Stored Functions 11-44
Storing a Function for a Formatted Inquiry 11-44
Using a Stored Function in a Formatted Inquiry 11-45
Using a Formatted Stored Inquiry with a Stored Function and Substitutable Values 11-46
Executing a Formatted Stored Inquiry with a Stored Function and Substitutable Values 11-47
UDO Summary 11-47

Appendix A: Commands, Noise Words, and Name Combinations

Commands A-1
Notes on the Commands A-4
Noise Words A-4
Name Combinations A-5
Notes on Name Combinations A-5
DISPLAY DIRECTORY MAP (PDM) Command with IMS (DL/I) A-6
DISPLAY DIRECTORY MAP (PDM) Command with VSAM Non-Hierarchical File A-7
DISPLAY DIRECTORY MAP (PDM) Command with VSAM Hierarchical File A-8
DISPLAY DIRECTORY MAP (PDM) Command with DB2 Table A-9
DISPLAY DIRECTORY VOCABULARY (PDV) Command A-10
DISPLAY DIRECTORY DATA (PDD) Command A-11
PDD Command for a DB2 Table A-12
DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE) Command A-14
DISPLAY DIRECTORY LTERM (PDL Command) in IMS A-16
DISPLAY DIRECTORY LTERM (PDL Command) in CICS A-17
DISPLAY DIRECTORY DATA DESCRIPT (PDDDS) Command A-18
DISPLAY DIRECTORY MAP WHOLE (PDMW) Command A-20

Appendix B: Quick Reference

Notation Rules B-1
Terms B-2
Continuous Versus Conversational Mode B-2
UDO Versus Non-UDO B-2
PAGE and TIME Limit Checkpoint B-2
Stored Function B-2
Stored Inquiry B-2
System Database B-3
System Names B-3
Temporary Fields B-3
Text Editor B-3
Native SQL Support B-3

Commands/Statements Summary B-4
AVERAGE, COUNT, TOTAL B-4
COLUMN B-4
CONTINUE B-4
CONTINUE DEFERRED INQUIRY (CDI) B-4
DEFER B-4
DEFINE DIRECTORY FUNCTION (DDF) B-4
DEFINE DIRECTORY INQUIRY (DDI) B-4
DELETE DEFERRED INQUIRY B-4
DELETE DIRECTORY FUNCTION B-4
DELETE DIRECTORY INQUIRY B-4
DISPLAY B-5
DISPLAY DIRECTORY DATA (PDD) B-5
DISPLAY DIRECTORY DATA DESCRIPTION (PDDDS) B-5
DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE) B-5
DISPLAY DIRECTORY FUNCTION (PDF) B-5
DISPLAY DIRECTORY INQUIRY (PDI) B-5
DISPLAY DIRECTORY INQUIRY COMMENT (PDIC) B-5
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE (PDICW) B-5
DISPLAY DIRECTORY INQUIRY WHOLE (PDIW) B-5
DISPLAY DIRECTORY LTERM (PDL) B-5
DISPLAY DIRECTORY TERM (PDT) B-6
DISPLAY DIRECTORY MAP (PDM) B-6
DISPLAY DIRECTORY MAP WHOLE (PDMW) B-6
DISPLAY DIRECTORY VOCABULARY (PDV) B-6
DISPLAY FORMAT B-6
EDIT B-6
EDITSQ B-6
EXTRACT B-6
FIND B-6
IF B-7
LIMIT B-7
LINE B-7
NOSPACE B-7
OUTPUT B-7
PCDELETE B-7
PCEXTRACT B-7
PCLOAD B-7
PF B-8
REPEAT B-8
SHOW B-8
SKIP B-8
SORT B-8
SPACE B-8
SUM B-8

Commands/Statements Syntax B-9
  Arithmetic Expression B-9
  AVERAGE, COUNT, TOTAL....... B-10
  DISPLAY (Non-UDO)....... B-11
  DISPLAY FORMAT (UDO)....... B-12
  EXTRACT B-13
  FIND....... B-14
  IF B-14
    Relational condition B-14
    Existence Condition....... B-15
    Positional Condition....... B-15
    Compound Condition....... B-15
  LIMIT B-15
  OUTPUT B-16
  PCEXTRACT....... B-16
  SORT B-16
  SUM....... B-17
Stored Inquiries/Functions....... B-17
  DEFINE DIRECTORY INQUIRY (DDI) B-17
  DEFINE DIRECTORY FUNCTION (DDF)....... B-18
  DISPLAY DIRECTORY INQUIRY (PDI)....... B-18
  DISPLAY DIRECTORY INQUIRY COMMENT (PDIC) B-19
  DISPLAY DIRECTORY FUNCTION (PDF)....... B-20
  EDITSQ B-20
  DELETE DIRECTORY INQUIRY and DELETE DIRECTORY FUNCTION....... B-20
  Running Stored Inquiries....... B-21
  Using Stored Functions B-21
Terminal Commands B-21
  Starting VISION:Inquiry B-21
  CONTINUE....... B-22
  DEFER....... B-22
  CONTINUE DEFERRED INQUIRY (CDI) B-22
  DELETE DEFERRED INQUIRY....... B-22
  SHOW B-23
Directory Commands B-23
  DISPLAY DIRECTORY MAP (PDM)....... B-23
  DISPLAY DIRECTORY MAP WHOLE (PDMW) B-23
  DISPLAY DIRECTORY VOCABULARY (PDV) B-24
  DISPLAY DIRECTORY DATA (PDD & PDD WHOLE) B-24
  DISPLAY DIRECTORY DATA DESCRIPTION (PDDDS) B-25
  DISPLAY DIRECTORY TERMINAL (PDT)....... B-25

Native SQL Commands/Statement Syntax B-25
SELECT Statement B-25
DISPLAY B-25
EXTRACT B-26
DEFINE DIRECTORY INQUIRY B-26
DISPLAY DIRECTORY INQUIRY B-26
DISPLAY DIRECTORY INQUIRY COMMENT B-26
DELETE DIRECTORY INQUIRY B-26
EDITSQ B-26
Running Native SQL Syntax Inquiries B-26
DISPLAY DIRECTORY DATA B-27
Noise/Reserved Words B-27
Noise Words B-27
RESERVED WORDS B-27
Command/Keyword Summary B-28
Verbs B-28
Stored Inquiry and Function Commands B-29
TERMINAL COMMANDS B-29
DIRECTORY COMMANDS B-30
OPERATORS B-31
UDO COMMANDS and KEYWORDS B-32
Conditional Selection B-32
Text Editor Commands B-33
Primary Commands B-33
Line Commands B-34

Index

Chapter 1

Introducing VISION:Inquiry

Advantage™ VISION:Inquiry® (hereafter referred to as VISION:Inquiry) provides a simple way for users with little or no data processing background to produce a variety of reports using information stored in IMS™ (DL/I) databases, VSAM files, and DB2® tables. (DB2 is an option and may not be available at your installation.)

- You can enter free-form English-like statements to direct VISION:Inquiry to retrieve the data you want. Let VISION:Inquiry select the formatting details, or format the output yourself using User Defined Output (UDO).

  Note: User Defined Output (UDO) is not supported with the native SQL syntax facility. Any reference to UDO should be ignored for this facility.

- With the DB2 option, VISION:Inquiry can access DB2 tables. You can use VISION:Inquiry's syntax, or use native SQL syntax to embed SQL SELECT statements into VISION:Inquiry statements. In this guide, any reference to DB2 applies to both methods, unless otherwise noted.
- You can direct VISION:Inquiry to execute your inquiry and return the output to your terminal, another terminal, or a printer.
- You can create and modify inquiries and mathematical functions, and save them on your VISION:Inquiry directory for later use.
- You can access the information in up to 16 databases, files, and DB2 tables simultaneously and transparently in one inquiry.
- Using VISION:Inquiry with VSAM files and DB2 tables is identical to its use with IMS (DL/I) databases containing a single segment type, except where noted otherwise. The output from the examples in this guide was created from a structured database. If you use VSAM files or DB2 tables (or both), your output will be different (no suppression of repeating fields).
- The term 'database', in most cases, can be interpreted as a VSAM file or a DB2 table.
- VSAM files are supported in all environments except IMS online (MPP). The use of VISION:Inquiry with VSAM hierarchical files is identical to its use with IMS (DL/I) structured databases, except where noted otherwise.

What Does VISION:Inquiry Look Like?

A VISION:Inquiry statement is called an inquiry. An inquiry supplies to the system the information you want to see and how you want it processed. A typical inquiry statement could be:

```
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE ED.SCHOOL IF EMP.SEX = 'F';
```

This inquiry statement directs VISION:Inquiry to display the names of all women employees, plant IDs, educational degrees, and the schools from which they graduated.

How Do You Use VISION:Inquiry?

From your terminal, you enter an inquiry (like the one above). VISION:Inquiry executes the inquiry and responds with your output (as shown in Figure 1-1).

```
PAGE:       TRANCODE: II      INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE ED.SCHOOL IF EMP.SEX = 'F';


 PLANT.ID  EMP.NAME                   ED.DEGREE  ED.SCHOOL
 10100     PHYLLIS LOCKMEYER          BA         WISCONSIN
           MARY ANN THOMAS            HS
 20150     WILMA FORD                 HS
           SUSAN WARE                 BS         MICHIGAN
 30200     JANE LOWELL                HS
           PATRICIA BLAKELY           BS         ARIZONA
           SHARON DALEY               HS
 40300     JOAN EVANS                 BS         U OF CONN
                                      MS         MIT
                                      PD         MIT
 50300     MADELYN BATES              BA         OBERLIN
           VICKY WARD                 HS
 60200     MARCIE MORINO              BA         TUFTS
           KAREN REDFERN              HS
 70500     AGNES COVINGTON            HS
           MARTHA WALLINGHAM          BS         OHIO STATE

 *IXX9121  END OF INQUIRY         (85,7 USER DB CALL,ROOTS)
```

Figure 1-1   Inquiry Output

Or, you can specify in your inquiry how you want your output formatted. VISION:Inquiry executes the inquiry and responds with a formatted output that may include title, column headings, labels, date, time, and page number.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 18 'TOY COMPANY ROSTER'
SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;
                 TOY COMPANY ROSTER


       PLANT                   EMPLOYEE                 EMPLOYEE
   NUMBER                       NAME                     NUMBER

       10100            WILLIAM AMES                     10103
                        PHYLLIS LOCKMEYER                10104
                        MARY ANN THOMAS                  10105
       20150            WILMA FORD                       21116
                        CHARLES SALTER                   21124
                        PETER ZATKIN                     21137
```

Figure 1-2 User Defined Output

Note: Chapter 11, "User Defined Output" describes how you can format your inquiries.

You can also enter the name of a stored inquiry. VISION:Inquiry finds the inquiry, executes it, and generates your output.

```
PAGE:      TRANCODE: II     INQUIRY:
WOMEN.DEGREES  PLANT;

 PLANT.ID  EMP.NAME                  ED.DEGREE  ED.SCHOOL
 10100     PHYLLIS LOCKMEYER         BA         WISCONSIN
           MARY ANN THOMAS           HS
 20150     WILMA FORD                HS
           SUSAN WARE                BS         MICHIGAN
 30200     JANE LOWELL               HS
           PATRICIA BLAKELY          BS         ARIZONA
           SHARON DALEY              HS
 40300     JOAN EVANS                BS         U OF CONN
                                     MS         MIT
                                     PD         MIT
 50300     MADELYN BATES             BA         OBERLIN
           VICKY WARD                HS
 60200     MARCIE MORINO             BA         TUFTS
           KAREN REDFERN             HS
 70500     AGNES COVINGTON           HS
```

Figure 1-3 Stored Inquiry

The above inquiry has been stored under the name WOMEN.DEGREES. Chapter 4, "Using VISION:Inquiry" explains how to enter an inquiry from your terminal.

Inquiries

The words in inquiry statements (when the native SQL syntax facility is not used) represent commands, database names, field names, and operators.

- The commands tell VISION:Inquiry what actions to perform.
- The database and field names indicate what database information should be used.
- The other words clarify or modify the operations being performed.

These terms are explained in detail in Chapter 3, "Structure and Concept of VISION:Inquiry".

For the native SQL syntax facility, the inquiry statement consists of commands, SQL SELECT statement, and delimiters. The format of inquiries with the native SQL syntax facility is explained in detail in Chapter 5, "Simple Reports".

Commands

The commands available with VISION:Inquiry are described and illustrated in detail in Chapter 5, "Simple Reports" through Chapter 11, "User Defined Output" of this guide.

- If the VISION:Journey® for Windows option is installed in your system, see the *VISION:Journey for Windows User's Guide* for information and examples.
- Any reference to Advantage™ Intraccess™, hereafter referred to as Intraccess, in this guide is applicable only if the Intraccess option is installed in your system. For more information and examples, see the Intraccess documentation.

Following are the VISION:Inquiry commands with a brief description of their function:

AVERAGE	Averages the contents of a specified numeric field.
COUNT	Counts the number of occurrences of a specified field.
DEFINE	Stores inquiries and mathematical functions on the directory.
DISPLAY	Displays data from specified fields.
EDITSQ	Displays the stored inquiries in edit mode. The inquiries can then be edited, restored, and executed.

EXTRACT	Extracts data from a database/file and writes the data to a sequential data set. For more information and examples of the EXTRACT command, see the *VISION:Inquiry Technical Reference Guide*.
FIND	Allows two or more databases/files to be accessed in one inquiry. See Chapter 8, "Accessing Multiple Databases and Files" for FIND command examples of databases, VSAM files, VSAM hierarchical files, and DB2 tables.
FORMAT	Use to specify how you want the output arranged on the page or screen.
LIMIT	Limits the number of output lines displayed.
OUTPUT	Sends the output from the inquiry to another terminal or a printer.
SORT	Sequences output data into a specified order.
SUM	Defines mathematical operations to be performed on specified fields. SUM is not supported with VSAM non-hierarchical files and DB2 tables.
TOTAL	Totals the contents of a specified field.

DEFINE, DISPLAY, EDITSQ, EXTRACT, LIMIT, and OUTPUT are the only VISION:Inquiry commands supported in the native SQL syntax facility.

The other VISION:Inquiry commands must be implemented through the SQL SELECT statement clauses. For descriptions and exact syntax of the SQL SELECT statement, see to the appropriate IBM® manual.

The Directory

The directory is a library, stored in the system database, that contains many kinds of information related to VISION:Inquiry. It is set up and maintained by the data processing personnel or your system administrator. The directory information that you need to be familiar with is described below.

Database Definitions

Database definitions describe the IMS (DL/I) databases, VSAM files, and DB2 tables that can be accessed by VISION:Inquiry at your installation.

In this guide, the information from two IMS (DL/I) databases (PLANT and SKILL), four VSAM files (VSPLANT, VSSKILL, VSHPLANT, and VSHSKILL), and thirteen DB2 tables and views is used to generate examples. These databases/files are described in Chapter 2, "Databases and Files".

DB2 is an option. If DB2 has been installed on your system, the DB2 tables and views will be available through VISION:Inquiry.

No database definition is required for native SQL syntax inquiries, which get the DB2 table information directly from the DB2 catalog.

Stored Inquiries and Functions

VISION:Inquiry has provisions for saving user-defined inquiries and mathematical functions for repeated use. The stored inquiries can also be edited after saving. These are described in Chapter 9, "Using Directory Commands - Storing Inquiries and Functions" and Chapter 10, "Using Stored Inquiries and Functions".

Vocabulary

Commands, noise words, and operators are called the system vocabulary. The particular word used to indicate a function may be customized by an installation.

For example, one installation may use DISPLAY to indicate that data should be listed. Others may prefer REPORT or LIST for this function.

This guide uses the words supplied with the VISION:Inquiry System. Your system administrator can tell you what vocabulary is used at your particular installation.

User Profiles

Not all users of VISION:Inquiry need to access all areas of a database or use all of the commands available. VISION:Inquiry has provisions that tailor the range of commands and database fields that may be accessed to a user's specific requirements. This range of accessible commands and fields is called a User Profile and is defined by your system administrator.

Chapter

2

Databases and Files

A **database** or **file** is a collection of information relating to a certain subject or class of information, such as payroll, inventory, shipments, or sales.

VISION:Inquiry can access the information in a database, restructure it to meet the needs of the inquiry statement, and then display it on the terminal screen or list it on the printer.

The data in the database, however, always remains the same. It is not affected by how you manipulate data in your inquiries, as VISION:Inquiry accesses the data in READ ONLY mode.

The database is created by data processing personnel and is usually stored on disk.

Database Concepts

The elements which make up a database are defined below and illustrated in Figure 2-1.

A **database** is a collection of related records that forms a unit.

A **record** is a set of related data. For example, the information relating to the code, description, quantity on hand, and selling price of a toy, the plant where it is produced, and the people who work at that plant could constitute a record.

A **segment** of a record is a collection of related pieces of information. For example, information relating to employee number, employee name, and employee sex might compose a segment. There may be more than one segment in a record, each relating to different information.

Each one of the items of information within a segment is a **field**. A field holds one individual piece of data or information, such as a name or a number.

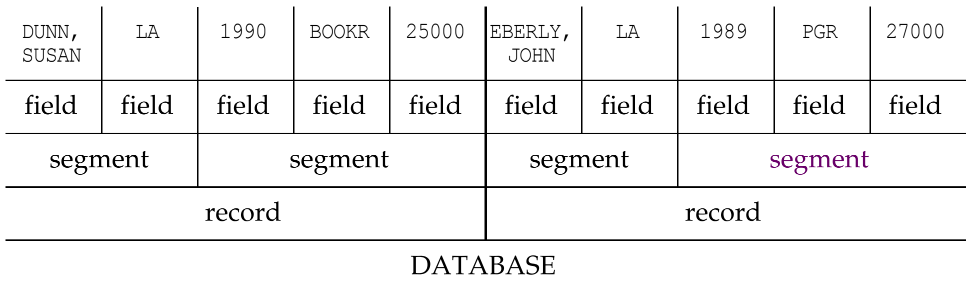


Figure 2-1 Elements of a Database

Field and Segment Names

Names can be assigned to both fields and segments to represent the kind of data they hold. For example, the database shown in Figure 2-1 could be illustrated as shown in Figure 2-2.

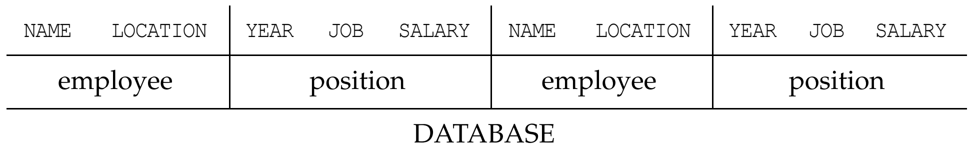
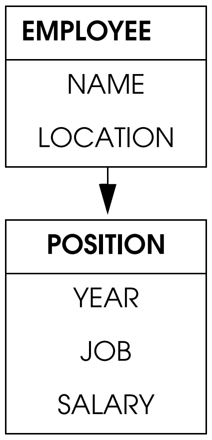


Figure 2-2 Field and Segment Names

Each record in the database has the same layout, the same segments, and the same fields. Thus, all records in a database could be represented by the diagram in Figure 2-3.



The name of the segment is at the top of each box; the field names are listed below each segment name.

Figure 2-3 Relationship of Fields and Segments in a Record

Multiple Occurrences of Segments

Suppose Susan Dunn has held jobs in several departments in the company. She has one occurrence of each of the fields in the POSITION segment for each job. Figure 2-4 shows this arrangement.

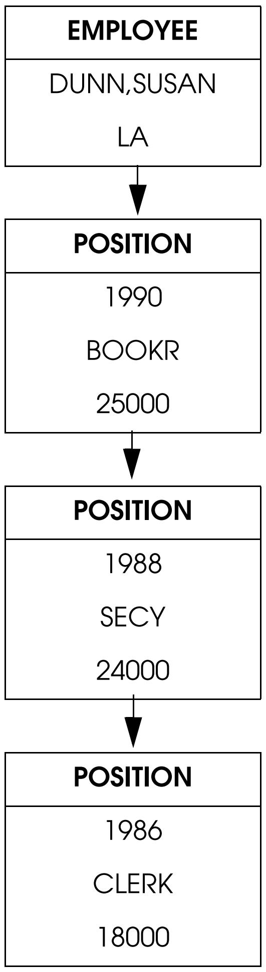


Figure 2-4 Multiple Occurrences of the Position Segment

Other employees with different job histories have a different number of occurrences of the POSITION segment. For example, John Eberly may have only two occurrences, and new hires in the firm have only one, representing their current position.

Structured Databases

A database containing records with multiple occurrences of segments and fields is called a **structured** or **hierarchical database**. It is organized in a dependency-related format; each lower segment is dependent upon a higher segment.

For example, all three occurrences of POSITION for Susan Dunn in Figure 2-4 are dependent upon the same root segment, EMPLOYEE. This creates a structure that resembles an inverted tree; there is a primary element (the root) and various dependent elements (the branches).

All of these elements are called segments. The root segment ordinarily contains the name of a large generalized subject area or class of information. The lower level segments are dependent segments containing specific subtopics and details that relate to the root segment. Each segment is made up of related fields.

Root and Dependent Segments and Levels

The **root segment** is the identifying segment for an entire record. Each record in a database contains only one root segment. Root segments are always at level 1. All other segments are dependent (or subordinate) segments. They begin at level 2.

Figure 2-5 shows the arrangement of a database with a root segment and dependent segments at various levels. The segments have been labeled **A** to **F**.

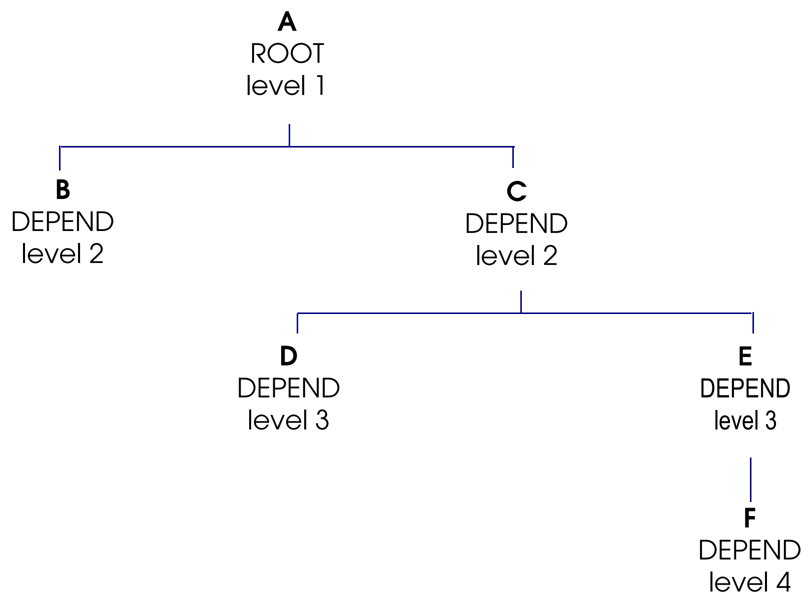


Figure 2-5 Dependent Segment Levels

The connecting lines and level numbers show relationships between segments. Level 4 is dependent upon level 3, level 3 is dependent upon level 2, and level 2 is dependent upon level 1, the root segment.

The number of dependent segments and their arrangements varies with each database, but each record within the database has the same arrangement.

Database Legs

The pathway formed by the root segment and its dependent segments is called a **database leg**. For example, in Figure 2-5, the root segment A and the dependent segment B form one database leg. A second leg is formed by the segments A, C, and D. Segments A, C, E, and F form a third leg. Another entire set of legs is formed by using segment C as the root segment. In this instance, C and D form one database leg and C, E, and F form a second leg.

Database Access

Database segments can be accessed in a variety of ways by an inquiry. The following table describes the variations:

Vertically	All or some of the fields in one leg can be used by an inquiry.
Horizontally	Information in all occurrences of one segment (for all records) can be used.
Combination	Many combinations of these two patterns can be used.

An inquiry may address all, or only some, of the occurrences of segments that are available.

Test Databases

The examples in this guide are built around two test databases: PLANT and SKILL. They describe a toy company with seven plants and 26 employees.

- The PLANT database contains information relating to the products and employees of the company.
- The SKILL database describes each employee's job classification and job code.

The databases are organized as hierarchical, structured files. Each record in the database has a root segment, and one or more dependent (or subordinate) segments, forming a three-dimensional, tree-like structure.

Figure 2-6 lists some information on the plants in the databases.

Plant Number	Plant Name	Number of Employees	Number of Toys
10100	DALLAS SALES	3	0
20150	REMOTE CONTROL PRODUCTS	4	6
30200	CORPORATE HDQTRS DALLAS	6	0
40300	DELUXE PRODUCTS	2	4
50300	MECHANICAL PRODUCTS	3	6
60200	BASIC TOYS	4	9
70500	DISTRIBUTION	4	20

Figure 2-6 Plants in the Test Databases

PLANT Database

Figure 2-7 diagrams a record in the PLANT database. The segment name is shown at the top of each segment level. The field names in that segment are shown below the segment name. The level number of each segment is indicated. There is one of these structures for each plant in the database.

In the PLANT database, each record describes a plant within the company. Because there are seven plants in the toy company, there are seven root segments. Each has its own tree-like structure. The root segment is named PLANT and contains the plant number and name, phone number, and code representing the region of the country where it is located.

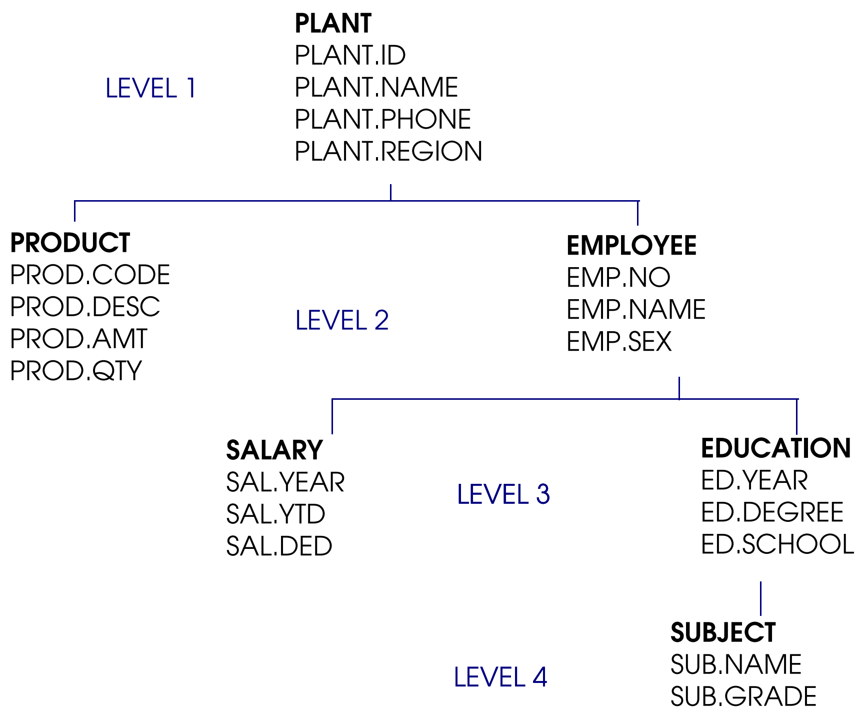


Figure 2-7 A Record in the PLANT Database

There are two segments at level 2, each dependent on the root segment.

- The first, PRODUCT, contains the names and codes of each toy manufactured, its selling price, and inventory on hand. There is one occurrence of this segment for each toy manufactured or distributed by a plant. Some toys are found at several plants, and there is one occurrence of the PRODUCT segment for this toy at each plant. Only five plants produce toys; the other two are concerned with sales and administration. Therefore, only five records have PRODUCT segments. The other two records have the potential for this segment, but it is not used. The PRODUCT and root segments form the left leg of the database.
- The second dependent segment at level 2, EMPLOYEE, contains the employee's name, identification number, and sex. There is one occurrence of this segment for each employee in a plant. The first plant, 10100, has three occurrences of this segment; the next plant, 20150, has four occurrences, and

so on. Altogether, since there are 26 employees represented in the database disbursed among the seven plants, there is a total of 26 occurrences of this segment in the database.

The two segments at level 3, SALARY and EDUCATION, are subordinate to the EMPLOYEE segment. There can be multiple occurrences of these segments for each employee.

- The SALARY segment contains information regarding salary amount, salary deductions, and the year represented. There is one occurrence of this segment for each year the person was employed at the toy company.
- The EDUCATION segment contains information about the dates of graduation, the degree(s) earned, and the school(s) attended. There is one occurrence of this segment for each degree the employee has earned. For college graduates, high school records are ignored; for those whose highest degree earned was high school, there is one occurrence of this segment, indicating year of graduation and “HS” for degree.

The SUBJECT segment, at level 4, is dependent upon the EDUCATION segment. There is one occurrence of this segment for each college degree an employee has earned. Each contains information about the major area of study for that degree and the grade earned.

The PLANT, EMPLOYEE, and SALARY segments form one leg, and the PLANT, EMPLOYEE, EDUCATION, and SUBJECT segments form a second leg.

SKILL Database

The SKILL database is a one-legged hierarchical structure that describes the job classifications in the toy company. Figure 2-8 diagrams a record in the SKILL database.

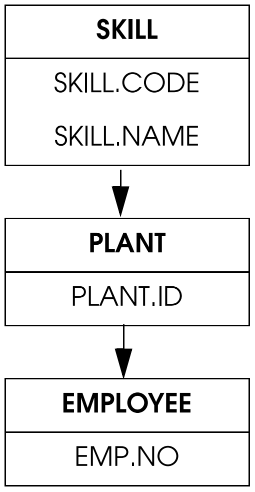


Figure 2-8 A Record in the SKILL Database

There are 13 records in the database, each representing one job classification. The root segment, SKILL, contains the name of one job skill and its code number.

The PLANT segment, at level 2, is dependent upon the SKILL segment. There is one occurrence of PLANT for each plant in the company that employs a person in this job classification.

The EMPLOYEE segment, at level 3, is dependent upon the PLANT segment. It contains the identification number of each employee in the plant holding that job classification. Two fields, PLANT.ID and EMP.NO, are present in both databases and allow the databases to be related to each other. The field names and data types for the PLANT and SKILL databases are listed in Figure 2-9.

Plant Database

Field Name	Type	Field Name	Type
PLANT.ID	character	EMP.NO	numeric
PLANT.NAME	character	EMP.NAME	character
PLANT.PHONE	character	EMP.SEX	character
PLANT.REGION	character	SAL.YEAR	numeric
PROD.CODE	character	SAL.YTD	numeric

Plant Database

Field Name	Type	Field Name	Type
PROD.DESC	character	SAL.DED	numeric
PROD.AMT	numeric	ED.YEAR	numeric
PROD.QTY	numeric	ED.DEGREE	character
		ED.SCHOOL	character
		SUB.NAME	character
		SUB.GRADE	character

Skill Database

Field Name	Type
SKILL.CODE	numeric
SKILL.NAME	character
PLANT.ID	numeric
EMP.NO	character

Figure 2-9 Field Names for PLANT and SKILL Databases

You need to know the field names in your database when you write inquiries using VISION:Inquiry. Your system administrator can give you a list of the names and indicate whether they hold numeric or character data.

VSAM Files

VISION:Inquiry supports the following VSAM file types in all environments except IMS online (MPP).

- Key Sequenced Data Set (KSDS): This is a keyed type data set. As such, data can be accessed directly by using the key field.
- Entry Sequenced Data Set (ESDS): Processed sequentially.
- Relative Record Data Set (RRDS): Processed sequentially.

There are two types of VSAM files supported in VISION:Inquiry: single segment and multiple segment. A multiple segment structure is used to define a VSAM file that uses the COBOL occurs clause. For the purpose of this guide, the single **segment structure** is called **VSAM non-hierarchical** and the **multiple segment** structure is called **VSAM hierarchical**.

VISION:Inquiry also supports VSAM files with fixed or variable occurrences of data items. In this guide, these types of files are referenced as hierarchical files and will be discussed later.

VSAM Test Files

The VSAM test files are VSPLANT, VSSKILL, VSHPLANT, and VSHSKILL. They describe a company with seven plants and 26 employees. The VSPLANT and VSHPLANT files contain information relating to the employees of the company. The VSSKILL and VSHSKILL files describe each employee's job classification and job code.

- The VSPLANT file is organized as a VSAM KSDS file with the employee number as the key field.
- The VSSKILL file is organized as a VSAM RRDS file.
- The VSHPLANT file is organized as a VSAM KSDS file with the PLANT.ID as the key field.
- The VSHSKILL file is organized as a VSAM KSDS file with the job classification as the key field.

VSPLANT File

Figure 2-10 diagrams a record and the field names in the VSPLANT file. The VSPLANT file describes the employees of the plants within the company. The company has 26 employees in seven plants. It is a VSAM KSDS file, and the key field is VSEMP.NO.

```
VSPLANT.ID VSEMP.NO VSEMP.NAME VSEMP.SEX VSED.DEGREE VSSAL.YTD VSSAL.DED
```

Figure 2-10 A Record in the VSPLANT File

VSSKILL File

Figure 2-11 diagrams a record and the field names in the VSSKILL file. The VSSKILL file describes the job classifications in the company. It is a VSAM RRDS file and is processed sequentially.

```
VSPLANT.ID  VSEMP.NO  VSSKILL.CODE  VSSKILL.NAME
```

Figure 2-11 A Record in the VSSKILL File

The field names and data types for the test VSAM files are listed in Figure 2-12.

VSPLANT File		**VSSKILL File**	
Field Name	**Type**	**Field Name**	**Type**
VSPLANT.ID	character	VSPLANT.ID	numeric
VSEMP.NO	numeric	VSEMP.NO	character
VSEMP.NAME	character	VSSKILL.CODE	numeric
VSEMP.SEX	character	VSSKILL.NAME	character
VSED.DEGREE	character		
VSSAL.YTD	numeric		
VSSAL.DED	numeric		

Figure 2-12 Field Names for the Test VSAM Files

VSAM Hierarchical Test Files

You can use the VSAM hierarchical file structure to define your VSAM files with fixed or variable occurrences of data items to VISION:Inquiry. You can access this data online in CICS®, in TSO, or in batch through VISION:Inquiry, execute inquiries, and create reports. Figure 2-13 is an example of a VSAM hierarchical record layout in COBOL. This record has both fixed and variable occurrences.

```
01      EMPLOYEE-RECORD.
        05 EMP-NO               PIC 9(5).
        05 EMP-NAME             PIC X(30).
        05 EMP-NUM-DEGREES      PIC 9.
           .
           .
           .
        05 EMP-DEGREE-DATA      OCCURS 1 TO 5 TIMES
           DEPENDING ON EMP-NUM-DEGREES.
           10 YEAR                 PIC 99.
           10 SCHOOL               PIC X(20).
           10 DEGREE               PIC X(10).
        05 SAL-HISTORY             OCCURS 10 TIMES.
           10 SAL-YEAR             PIC 99.
           10 SAL-AMT              PIC S9(8)V99.
```

Figure 2-13 VSAM Hierarchical Record Layout (In COBOL)

VISION:Inquiry maps the VSAM hierarchical record layout in Figure 2-13 into the hierarchical structure in Figure 2-14. **Fixed occurring segments** always occur the same number of times for each record of the file. **Variable occurring segments** occur a different number of times based upon a field in the parent segment.

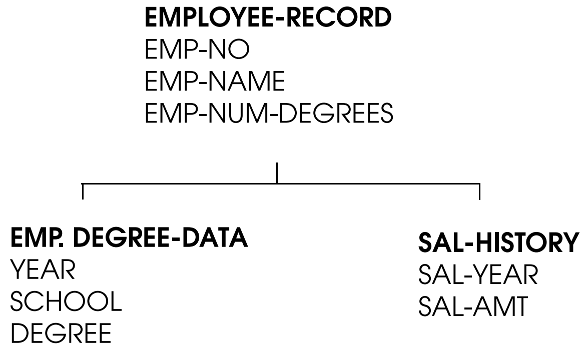


Figure 2-14 VSAM Hierarchical Record Layout (In VISION:Inquiry)

In Figure 2-14, the DEGREE segment is variable occurring based upon the field EMP-NUM-DEGREES. EMP-NUM-DEGREES must be in the DEGREE segment's parent (EMPLOYEE segment).

- Variable occurring segments are allowed for KSDS and ESDS files.
- Fixed occurring segments are allowed for KSDS, ESDS, and RRDS files.

The following sections describe two sample VSAM hierarchical test files supplied with the product.

VSHPLANT Test File

Figure 2-15 diagrams a record and the field names in the VSHPLANT file. Each record describes a plant within the company. Because the company has seven plants, there are seven records in the VSAM file. Each record contains multiple occurrences of product and employee data. The file is a KSDS file with VSHPLANT.ID as the KEY field.

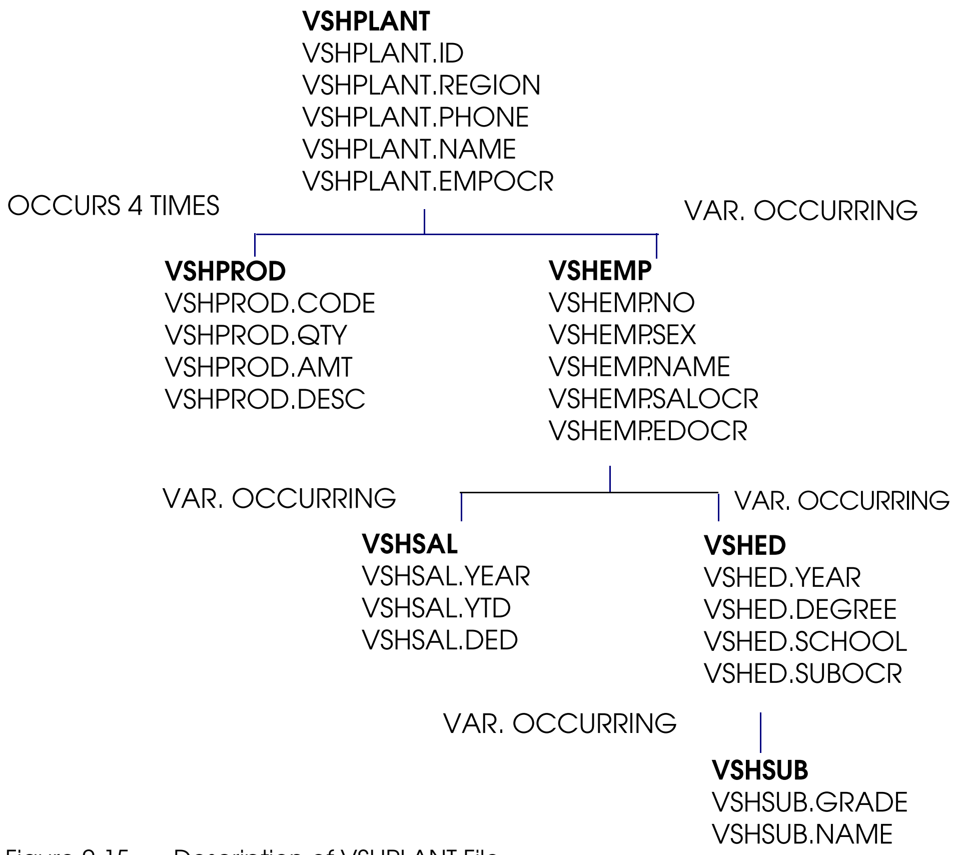


Figure 2-15 Description of VSHPLANT File

VSHSKILL Test File

Figure 2-16 diagrams a record and the field names in the VSHSKILL file. Each record describes a job classification within the company. There are 13 records in the VSAM file, one for each job classification. Each record contains multiple occurrences of plant and employee data. The file is a KSDS file with VSHSKILL.CODE as the KEY field.

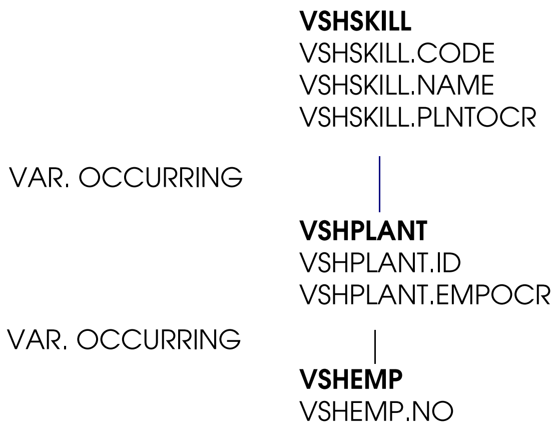


Figure 2-16 A Record in VSHSKILL File

DB2 Concepts

DATABASE 2 (DB2) is a relational database management system that manages data for large system users. All DB2 data exists in one or more tables. A table consists of a specific number of columns and some number of unordered rows.

The following table compares conventional terms with terms used with DB2.

Conventional Term	Relational Term
Database: A collection of data sets	Relational Database: A collection of tables
Data Set: A collection of records	Table: A collection of rows
Records: A collection of fields	Row: A collection of values
Field	Column

Figure 2-17 Comparison of Conventional Terms to Relational (DB2) Terms

The data in a table is arranged in *columns* and *rows*.

- Columns contain the same kind of data, have names (which appear at the top when columns are displayed on a terminal screen), and appear vertically when displayed on a terminal screen.
- Rows usually contain different kinds of data about a single entity, do not have names, and appear horizontally when displayed on a terminal screen.

A table can consist of several rows with several columns. However, a table can also consist of one of the following:

- One row with several columns

DEPTNO	DEPTNAME	MGRNO
BO1	PLANNING	000020

- Several rows with one column

DEPTNO
A00
B01

- One row with one column

DEPTNO
A00

- No rows with one or more columns. This is sometimes called an “empty table.”

Structured Query Language (SQL)

SQL is the basic language used to communicate with DB2. Despite its simpler structure, it provides a richer database manipulation language than the basic language (DL/I) used to access IMS. However, the VISION:Inquiry language contains the same kinds of advanced database query capabilities and can be used effectively with either database system. Users will find no difference between DB2 tables and IMS databases with only one segment type.

Figure 2-18 and Figure 2-19 diagram the test DB2 tables and views associated with column names used in this guide to run the examples when the native SQL facility is **not** used.

DB2 Test Tables Not Using the Native SQL Syntax Facility (Names as supplied or Assigned)

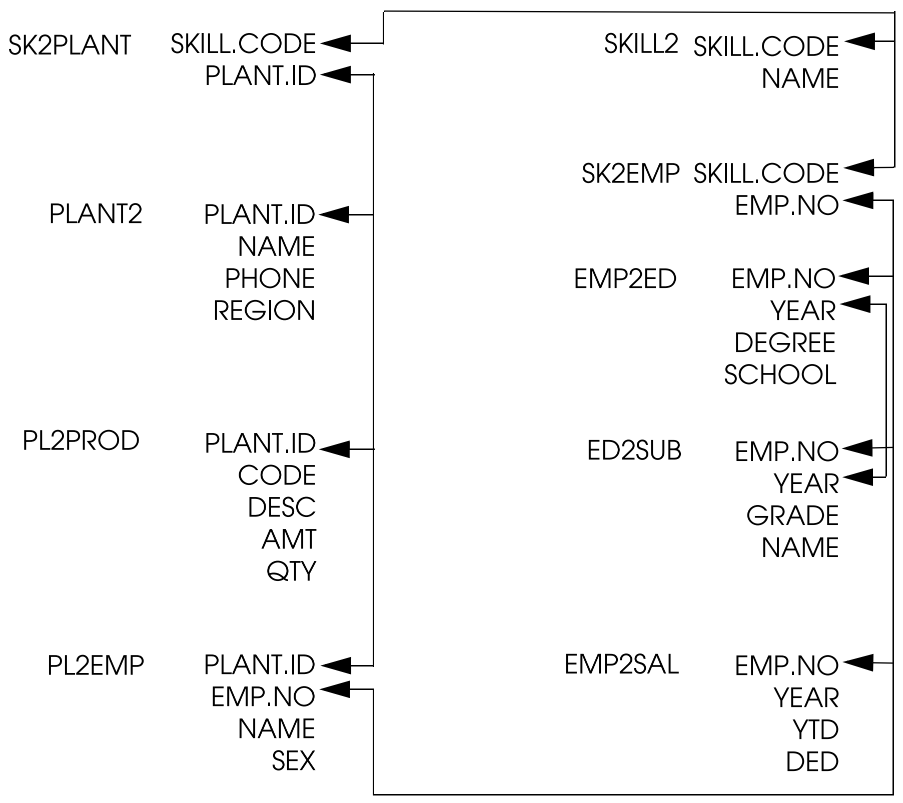


Figure 2-18 DB2 Test Tables and Views (1 of 4)

The arrows indicate the columns on which the tables can be joined. These columns can be used as matching fields when accessing multiple fields in VISION:Inquiry native mode, or AQF (Automatic Query Facility).

DB2 Test Views Not Using the Native SQL Syntax Facility (Names as Supplied or Assigned)

PRODUCTS	**SALARIES**	**EDUCATION**	**SKILLS**
PLANT_ID	PLANT_ID	PLANT_ID	SKILL_CODE
PLANT_NAME	PLANT_NAME	PLANT_NAME	SKILL_NAME
PLANT_PHONE	PLANT_PHONE	PLANT_PHONE	PLANT_ID
PLANT_REGION	PLANT_REGION	PLANT_REGION	EMP_NO
PROD_CODE	EMP_NO	EMP_NO	
PROD_DESC	EMP_NAME	EMP_NAME	
PROD_AMT	EMP_SEX	EMP_SEX	
PROD_QTY	SAL_YEAR	ED_YEAR	
	SAL_YTD	ED_DEGREE	
	SAL_DED	ED_SCHOOL	
		SUB_NAME	
		SUB_GRADE	

Figure 2-19 DB2 Test Tables and Views (2 of 4)

You can also use the native SQL syntax facility and combine the VISION:Inquiry commands with the SQL SELECT statement to get the output report back from a DB2 table. Figure 2-18 through Figure 2-21 show the names of the test DB2 tables and their columns as supplied or assigned by the system administrator.

The DB2 table and column names used in the inquiry with the native SQL syntax facility must be the actual names used at the time of table definition, which may be different from those shown in Figure 2-18 and Figure 2-19. Figure 2-20 and Figure 2-21 show the names that should be used in the embedded SQL SELECT statement of the inquiry using the native SQL syntax facility.

DB2 Test Tables Using the Native SQL Syntax Facility (Names for Embedded SQL SELECT)

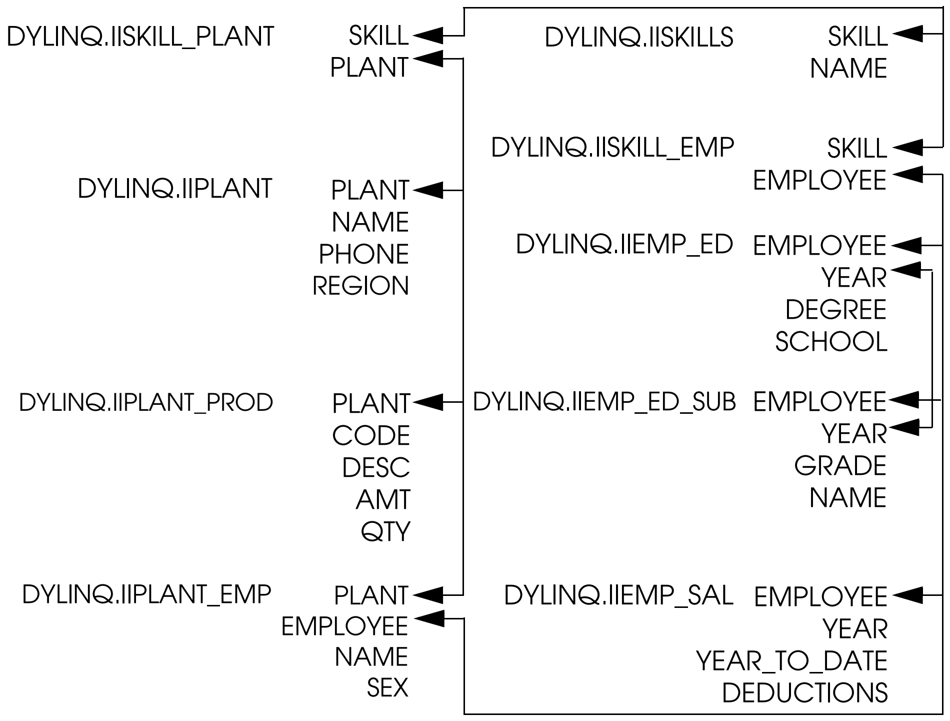


Figure 2-20 DB2 Test Tables and Views (3 of 4)

DB2 Test Views Using the Native SQL Syntax Facility (Names for Embedded SQL SELECT)

DYLINQ. IIPRODUCTS	**DYLINQ. IISALARIES**	**DYLINQ. IIEDUCATION**	**DYLINQ. IISKILLS**
PLANT_ID	PLANT_ID	PLANT_ID	SKILL_CODE
PLANT_NAME	PLANT_NAME	PLANT_NAME	SKILL_NAME
PLANT_PHONE	PLANT_PHONE	PLANT_PHONE	PLANT_ID
PLANT_REGION	PLANT_REGION	PLANT_REGION	EMP_NO
PROD_CODE	EMP_NO	EMP_NO	
PROD_DESC	EMP_NAME	EMP_NAME	
PROD_AMT	EMP_SEX	EMP_SEX	
PROD_QTY	SAL_YEAR	ED_YEAR	
	SAL_YTD	ED_DEGREE	
	SAL_DED	ED_SCHOOL	
		SUB_NAME	
		SUB_GRADE	

Figure 2-21 DB2 Test Tables and Views (4 of 4)

Note that accessing DB2 tables is an optional feature. Check with your system administrator to verify that the DB2 option is available at your installation, and that the names used in Figure 2-20 and Figure 2-21 are the correct names.

Chapter

3

Structure and Concept of VISION:Inquiry

VISION:Inquiry inquiries represent operations to be performed by the system. Inquiries are composed of one or more statements. Statements give VISION:Inquiry information on what databases to use, what data to select from the databases, and how to summarize the data selected.

The native SQL syntax facility will be explained later in this guide. It does not apply to the information in this chapter.

Statement Structure

Statements are composed of commands, database names, and field names. They can also include titles, constants (both character and numeric), operators, and noise words. The example below is a typical default inquiry statement. It tells VISION:Inquiry to list the female employees and their salaries for each plant. (See Chapter 11, "User Defined Output" for an explanation of statement syntax using UDO).

`DISPLAY`	`FROM`	`PLANT`	`THE`	`'FEMALE EMPLOYEES INFORMATION'`
command to list	noise word	database to be accessed	noise word	title

`PLANT.ID EMP.NAME SAL.YTD`	`IF EMP.SEX = 'F'`	`;`
fields from the database to be listed	conditional selection phrase	termination symbol

The **command** usually appears first in the inquiry statement. It tells VISION:Inquiry the specific processing to be performed. The commands are explained in Chapter 5, "Simple Reports" through Chapter 11, "User Defined Output". A complete list is shown in Appendix A, "Commands, Noise Words, and Name Combinations".

Noise words are optional words used in a statement for clarity. A complete list of noise words is shown in Appendix A, "Commands, Noise Words, and Name Combinations".

The **database name**, which is required in every inquiry, follows the initial command in the inquiry statement. It tells VISION:Inquiry which database to access. The databases used in this guide are named PLANT and SKILL. Your system administrator can tell you the names of the databases you can use at your installation.

The **titles** represent one or two lines of character constants that appear at the top of each page of the report. The titles must follow the command or data base name as the first two items to be displayed, and can be up to 70 characters.

The **field names** represent fields in the database being accessed. They tell VISION:Inquiry which data in the database to use.

Operators are arithmetic, relational, and logical symbols and are explained in Chapter 5, "Simple Reports", and Chapter 7, "Assignment Statement and Arithmetic Processing".

The sample inquiry statement includes a **conditional selection phrase** that allows you to select only specific data for processing. In this case, because EMP.SEX = 'F' was specified, the inquiry statement selects only data on women from the database. Conditional selection is explained in Chapter 5, "Simple Reports". The 'F' in the conditional selection phrase is a character constant. The '=' is a relational operator. Character and numeric constants and operators are discussed later in this chapter.

Each inquiry is terminated with a semicolon (;).

Multiple Field Names Following a Command

When several field names or values follow a command, they are all assumed to belong to that command until another command is entered or the end of the statement is reached.

Example:

```
DISPLAY  PLANT  PLANT.ID  EMP.NAME  SAL.YTD  SORT  SAL.YTD ;
```

PLANT.ID, EMP.NAME, and SAL.YTD are fields belonging to the DISPLAY command. In this example, the second SAL.YTD is assumed to belong to the SORT command.

It is also assumed that the database you are using for both commands is PLANT.

Separators

The database names, commands, field names, values, and titles that make up an inquiry statement must be separated from each other by at least one blank space. Spaces cannot be embedded within commands, words, names, or values. Spaces may be included in character constants, and such usage is explained later in this chapter.

Parentheses and operators also serve as separators; spaces are optional if these separators are present.

Commas may not be used as separators.

Valid Example:

```
DISPLAY PLANT PLANT.ID EMP.NAME IF SAL.YTD + 300 GT 25000;
```

Invalid Example:

```
DI SPLAY PLANT PLANT,ID, EMP.NAME IF SAL.YTD + 300 GT25000;
```

Names, commands, constants, and operators cannot be split onto two lines except under the following circumstances.

Continuing Statements

When any inquiry statement is too long to fit on one line on the terminal screen, it automatically wraps around onto subsequent lines, sometimes splitting names, commands, constants, or operators.

Example:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME EMP.SEX ED.DEGREE ED.
YEAR ED.SCHOOL SUB.NAME SUB.GRADE;
```

As long as this occurs during a wraparound, it is not an error condition.

Improving Readability

To improve readability, you can also enter each phrase of an inquiry statement on a separate line on the screen. If a line does not completely fill the screen, words may not be broken.

Valid Example:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME EMP.SEX
ED.DEGREE ED.YEAR ED.SCHOOL
SUB.NAME SUB.GRADE;
```

Invalid Example:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME EMP.
SEX ED.DEGREE ED.YEAR ED.SCHOOL
SUB.NAME SUB.GRADE;
```

For more information on entering inquiries on the terminal, see Chapter 4, "Using VISION:Inquiry".

Terminating an Inquiry

Terminate each inquiry statement with a semicolon (;). Systems operating in batch mode require two semicolons. Check with your system administrator to learn how your inquiries must be terminated. You can use two semicolons after your new inquiry. VISION:Inquiry ignores whatever follows two semicolons.

Names

A **name** identifies a database, database field, temporary field, or a stored inquiry or function.

- The database and field names are assigned in the database definition.
- Temporary field names, stored inquiries, and function names are assigned by the user.

In VISION:Inquiry, the symbols A-Z, $, @, and # are recognized as alphabetic characters, and 0-9 are recognized as numeric characters. "_" and "." are recognized as special characters. An alphanumeric character is either an alphabetic or a numeric character.

Database names may have up to 8 characters. The first must be an alphabetic character followed by alphanumeric characters.

Field, stored inquiry, and **stored function names** may have up to 32 characters. The first must be an alphabetic character followed by alphanumeric characters, special characters, or any combination of the two.

Temporary field names may have up to 32 characters. They always start with the % symbol followed by alphanumeric characters, special characters, or any combination of the two.

Valid Examples	Invalid Examples
sal .year	79
Z123	,T93
A	4salary

The invalid examples are incorrect because names must begin with an alphabetic character.

Field names may not duplicate system-defined names, such as commands, built-in function names, or operators.

Constants

Constants can be character or numeric. The value of a constant does not change during the execution of VISION:Inquiry.

Character Constants

A **character constant** represents explicit character data.

- It consists of up to 256 characters enclosed in single quotation marks.
- Any character that can be entered may be included within the character constant.
- Any blank space that is part of the character constant must be enclosed within the quotation marks. When using a display terminal keyboard, you indicate a space by pressing the space bar, not by moving the cursor.
- If the length of the character constant is greater than the length of the data field, the character constant is truncated on the right.
- If the length of the character constant is less than the length of the data field, the character constant is extended on the right with blanks.
- A character value is left-justified in the field when it is output.

Embedded single quotation marks are represented by two single quotation marks.

Examples:

’management’	’ leading space’	’X139’
’m.i.t.’	’trailing spaces ’	
’file clerk’	’women’’s rights’	

Numeric Constants

A **numeric constant** represents explicit numeric data.

- It consists of one to 14 digits to the left of the decimal point, and up to four digits to the right of the decimal point.
- A numeric constant must begin with a digit; it may not begin with a decimal point.
- The digits must follow each other without intervening spaces or characters.
- Numeric values are right-justified in the field in which they are output.

Valid Examples	Invalid Examples	Why Example is Invalid
789651	$18,000	begins with a $, has an embedded comma
1.69	1 5000	has an embedded space
0.001	1E56	has an embedded character
3	.97	does not begin with a digit

Numeric constants involved in addition and subtraction operations may have only two digits to the right of the decimal point. Numbers with more digits are truncated.

Examples:

0.21

831.88

Constants involved in multiplication and division operations may have up to four digits to the right of the decimal point. Numbers with more digits are truncated.

Examples:

0.2161

831.886

Titles

Titles consist of up to 70 characters enclosed in single quotation marks. The other rules for titles are the same as for character constants (see Character Constants on page 3-5).

Operators

VISION:Inquiry recognizes the following operators.

Relational Operators		
LT	<	Less than
LE	<=	Less than or equal to
EQ	=	Equal to
GE	>=	Greater than or equal to
GT	>	Greater than
NE	¬=	Not equal to
LIKE		Partial matching for character fields

Arithmetic Operators	
+	Addition
-	Subtraction
*	Multiplication
/	Division

Logical Operators		
AND	&	Logical product (all must be true)
OR	|	Logical sum (at least one must be true)

Chapter

4

Using VISION:Inquiry

Most installations access VISION:Inquiry using a terminal operating in IMS (Information Management System) or in CICS (Customer Information Control System). You initiate VISION:Inquiry, enter your inquiry, and receive the results back at your terminal. There are some variations to this arrangement, which are discussed later in this chapter.

Terminals vary from one system to another, but they all have keys that perform similar functions. Your system administrator can give you instructions on how to use your particular terminal. You need to know which keys do the following actions:

- Enter (submit) lines
- Page forward or backward
- Move the cursor
- Insert characters
- Delete characters
- Clear the screen.

In this chapter, we use "Enter" as the name of the submit key. Your keyboard submit key may be Return or something else.

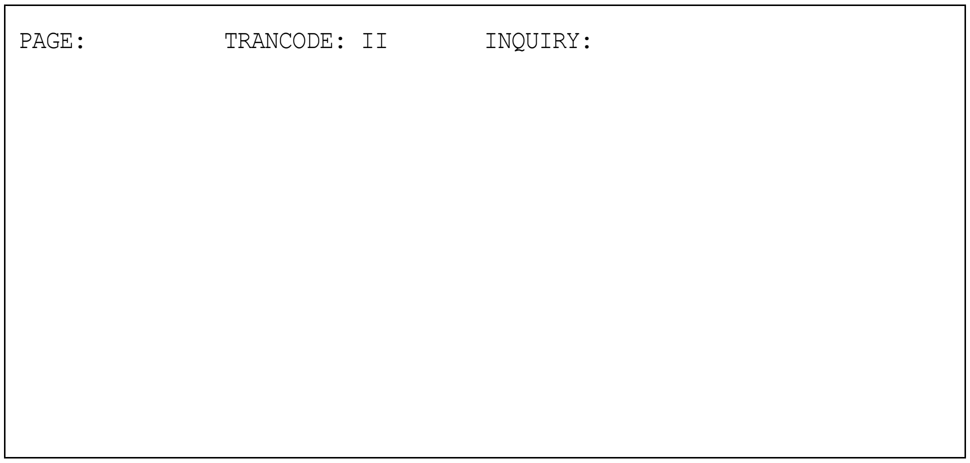
Most terminals also have a set of keys called program function (PF) keys. In the standard VISION:Inquiry installation for IMS users, three of these keys (PF1, PF2, PF3) are programmed to replace three VISION:Inquiry commands. These are explained later in this chapter. Check with your system administrator for the functions performed by the program function keys on your terminal.

The VISION:Inquiry Screen Under IMS

You enter the following command to tell IMS that you want to use VISION:Inquiry.

```
/FORMAT INQIMS
```

IMS responds with the screen shown in Figure 4-1.



```
PAGE:          TRANCODE: II       INQUIRY:
```

Figure 4-1 Screen Under IMS

VISION:Inquiry also uses "/FORMAT INQUDO." See Chapter 11, "User Defined Output" for a complete explanation.

Some installations may use slightly different wording, but the command performs the same function.

PAGE The PAGE field, at the left of the screen, is an option used to view a specific group of lines on the screen. Its operation is defined at each installation and can be explained by your system administrator. A page is the number of lines of data that fit on one screen.

TRANCODE The TRANCODE field is where you enter your transaction code. The transaction code is a string of one to eight characters that tells VISION:Inquiry what User Profile to use for your inquiry. It is assigned by your system administrator at your installation. Some installations use only one User Profile; in these cases, you do not have to enter any trancode at all.

INQUIRY The INQUIRY field is where you enter your inquiry.

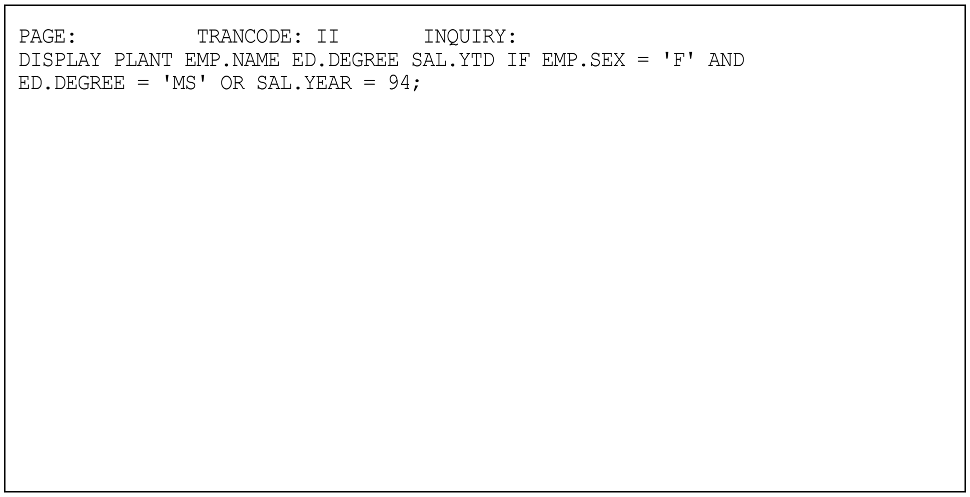
Entering an Inquiry Under IMS

Now you are ready to enter your inquiry. The cursor is usually positioned two spaces after TRANCODE, waiting for your entry. (The cursor is usually positioned after INQUIRY on systems not requiring a trancode.) If the cursor is positioned anywhere else, move it two spaces to the right of TRANCODE.

You type the trancode and move the cursor to the right of INQUIRY or to the beginning of the next line. You are now ready to enter your inquiry. You may continue to type characters in a continuous stream so that they fill up the line and wrap around to the next line, or you may begin each phrase on a separate line.

At the end of the command line, type the closing semicolon, and press Enter. This action submits your inquiry for execution.

Figure 4-2 shows the screen with the trancode and inquiry keyed in. II is the trancode used in the examples. For clarity, the balance of inquiries in this guide start on the first line following INQUIRY.



```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND
ED.DEGREE = 'MS' OR SAL.YEAR = 94;
```

Figure 4-2	Trancode and Inquiry Keyed In and Submitted

The size of the screen image area can vary because installations may define their own dimensions. The entire screen area is used in the example above, 80 characters across a line and 23 lines down a page.

In the examples in this guide, the VISION:Inquiry screen is divided into two sections: the top portion, reserved for the inquiry; and the bottom portion, used for the output.

VISION:Inquiry executes the inquiry and displays the first page of the output on the lower portion of the screen, as shown in Figure 4-3.

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND ED.DEGREE
= 'MS' OR SAL.YEAR = 94;


EMP.NAME                   ED.DEGREE       SAL.YTD
WILLIAM AMES               BA            52,000.00
PHYLLIS LOCKMEYER          BA            48,000.00
WILMA FORD                 HS            15,600.00
CHARLES SALTER             BA            30,000.00
PETER ZATKIN               BA            50,000.00
                           MA
SUSAN WARE                 BS            32,000.00
JOHN HENRY CRANE           HS            19,600.00
FREDERICH GRAY             BA            48,000.00
                           MA
MITCHELL J HOOPS           BS            92,000.00
                           MA
JANE LOWELL                HS            22,000.00
DONALD M KING              BS            66,000.00
                           MS
```

Figure 4-3 First Page of Output

Input Lines and Output Lines

Depending on your system, the number of lines allocated to the inquiry and output portions may vary.

- The number of lines is installation-defined and systems have more or less lines for their inquiry portion.
- Any inquiry requiring more than the inquiry portion area is not executed.
- In some systems, inquiry and output lines do not share a screen and do not appear at the same time. In this arrangement, the inquiry appears by itself on only one screen and the output appears on subsequent screens, overlaying the inquiry.

After You Submit a Query

After you have submitted your inquiry for execution, VISION:Inquiry checks it for errors. If any are found, it responds with a descriptive message and waits for you to correct the errors and resubmit the inquiry. (Error handling is discussed in more detail later in this chapter.) Then VISION:Inquiry performs the steps in your inquiry and displays the output generated. An “END OF INQUIRY” message appears at the bottom of the screen on the last page of your output. The message also shows the total number of user database calls and root calls. Figure 4-3 shows the first page of the output generated by the inquiry. You know there are more pages because you do not have the “END OF INQUIRY” message.

Press PA1 or the page forward key and the next output page appears, overlaying the bottom 16 lines; the inquiry remains on the screen. Figure 4-4 shows the second page of output from your inquiry.

If the PF1 key is pressed after the "END OF INQUIRY" message appears on the screen, IMS responds with an "END OF OUTPUT" message on the screen and the remainder of the screen is cleared.

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND ED.DEGREE
= 'MS' OR SAL.YEAR = 94;

EMP.NAME                   ED.DEGREE       SAL.YTD
DONALD M KING              PD
JOAN EVANS                 BS            64,000.00
                           MS
                           PD
JONATHAN OAKS              BS            36,000.00
DAVID YORK                 BA            37,000.00
RUSSELL M SIMMONS          BA            48,000.00
                           MA
KAREN REDFERN              HS            22,000.00
RONALD T JACKSON           BA            46,000.00
STEPHEN MCGEE              HS            19,000.00
AGNES COVINGTON            HS            20,000.00
*
*
IXX9121  END OF INQUIRY.                (218,7 USER DB CALLS,ROOTS)
```

Figure 4-4 Second Page of Output

The asterisks at the left side of the screen and the informational message "IXX9121 END OF INQUIRY" at the bottom of the page tell you that this is the last page of the output. The message in Figure 4-4 also shows that 218 user database calls, including seven root calls, were issued for processing the query. Other informational messages can be displayed after an inquiry is executed, depending on the steps it performs.

Entering a Second Inquiry Under IMS

If you want to enter another inquiry under the same trancode, type the new inquiry (overlaying the old), and press Enter. The new output also overlays the old.

If the new inquiry is shorter than the old one, you must type a semicolon, erase the rest of the old inquiry, and then submit the new inquiry.

If you use *two* semicolons after your new inquiry, you do not need to erase the rest of the old inquiry. VISION:Inquiry ignores whatever follows two semicolons.

If you wish to reenter the existing inquiry, you must retype at least one letter of the old inquiry, and then press Enter. The inquiry is executed again, and the output is regenerated.

If a different trancode is required for a new inquiry, type in the new trancode (overlaying the old) and enter your inquiry.

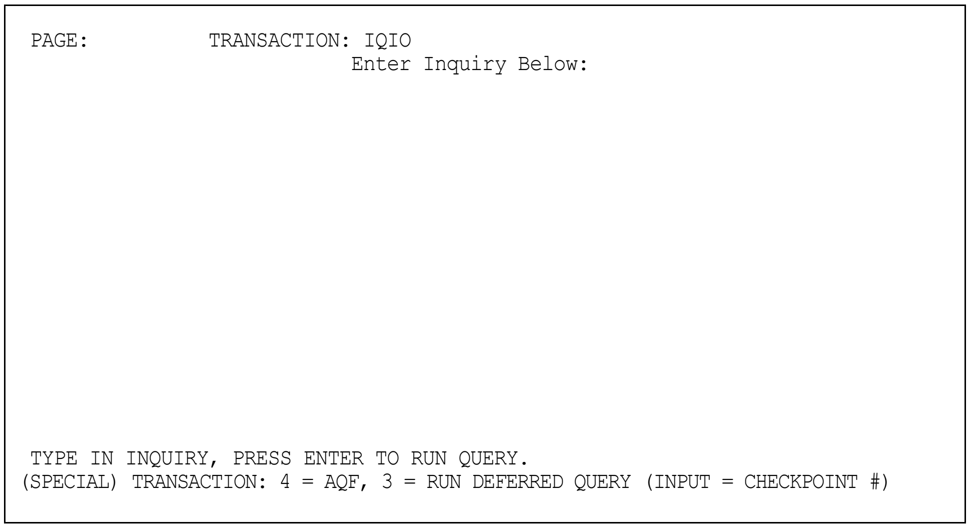
You can clear the screen between inquiries; however, you must then also bring up the VISION:Inquiry screen again. To do this, press the Clear key, reenter the /FORMAT INQIMS command and trancode, and enter your new inquiry.

The VISION:Inquiry Screen Under CICS

Enter the following transaction ID to tell CICS that you wish to use VISION:Inquiry.

```
IQIO
```

CICS responds with the screen shown in Figure 4-5.



```
 PAGE:          TRANSACTION: IQIO
                               Enter Inquiry Below:

















 TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
(SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 4-5 Screen Under CICS

PAGE The PAGE field, at the left of the screen, is an option used to view a specific group of lines on the screen.

- To view a specific page, type P/n (where n = page number) or type L for last, and press Enter.
- To send the output to a specific terminal, type C/terminal ID in the PAGE field, and press Enter.

TRANSACTION Use the TRANSACTION field to enter your transaction ID. The transaction code is a string of one to four characters that tells VISION:Inquiry what User Profile to use for your inquiry.

Notice the messages at the bottom of the screen:

```
TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
(SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED
QUERY (INPUT = CHECKPOINT #)
```

CICS users may use the TRANSACTION field to request access to the Automatic Query Facility (AQF) or to run a deferred inquiry.

Deferring or continuing an inquiry is discussed later in this chapter.

Enter Inquiry Below This field indicates that your inquiry comes next.

Note: Delete any entries in the PAGE field before making an entry in the TRANSACTION field.

In the examples in this guide, the VISION:Inquiry screen is divided into two sections: the top portion is reserved for the inquiry (starting from the third line) and the bottom portion for the output. VISION:Inquiry executes the inquiry and displays the output on the bottom portion of the screen.

Input Lines and Output Lines

Depending on your system, the number of lines allocated to the inquiry and output portions may vary.

- Systems may allow more or fewer lines for the inquiry portion.
- Any inquiry requiring more than the inquiry portion area is not executed. In some systems, the output lines may overlay some parts of the inquiry.
- In other systems, inquiry and output lines do not share a screen and do not appear at the same time. In this arrangement, the inquiry appears by itself on one screen and the output appears on subsequent screens, overlaying the inquiry.

The output is displayed a page at a time. Figure 4-6 shows the next to last page of output from your query.

```
PAGE: P/N      TRANSACTION: IQIO
                             Enter Inquiry Below:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND ED.DEGREE =
'MS'  OR SAL.YEAR = 94;


EMP.NAME                  ED.DEGREE       SAL.YTD
DONALD M KING             PD
JOAN EVANS                BS            64,000.00
                          MS
                          PD
JONATHAN OAKS             BS            36,000.00
DAVID YORK                BA            37,000.00
RUSSELL M SIMMONS         BA            48,000.00
                          MA
KAREN REDFERN             HS            22,000.00
RONALD T JACKSON          BA            46,000.00
STEPHEN MCGEE             HS            19,000.00
AGNES COVINGTON           HS            20,000.00
*
*
IXX9121  END OF INQUIRY.                (218,7 USER DB CALLS,ROOTS)
  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
 (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 4-6   Next to Last Page of Output

The "IXX9121 END OF INQUIRY" message appears at the bottom of the screen on the next to last page of your output, as shown in Figure 4-6. It also shows that 218 user database calls, including seven root calls, were used for processing the query. The last page contains only the generated query.

Basic Mapping Support (BMS) Paging Options

Basic Mapping Support is unique to CICS. Each input and output screen is called a map. Each map contains areas for input and informational messages. Output maps can be accessed by using the PAGE field. Enter standard BMS paging options in the PAGE field.

Some of the BMS paging options are indicated at the top of each output screen. (See Figure 4-6.)

- Type P/N in the PAGE field to access the next page of output.
- Type P/2 to access the second page. P/3 accesses the third page, and so on.
- P/L displays the last page. (This page contains only your query.) P/ is the IBM default; it may be different at your installation.

If the page forward key is pressed after the last page of output, or a page number is entered that is greater than the number of output pages of the inquiry, a CICS error such as the following is displayed:

```
DFH4108 xxxx requested page value not valid
```

After the message, the remainder of the screen is cleared. You can clear the screen and either type IQIO to start again or type a valid P/ command, clear the rest of the message, and press Enter to get back to the output of the inquiry.

Entering a Second Inquiry Under CICS

Under CICS, you may enter a second inquiry on any page of output. You may use two semicolons after your new inquiry. VISION:Inquiry ignores whatever follows two semicolons. The last page (original inquiry page) has 20 lines of input. On the other pages, there are only four lines available for input because the output lines overlay the input from the fifth line of input.

If you wish to reenter the existing inquiry, delete any entry in the PAGE field and press Enter. The inquiry is executed again, and the output is regenerated.

If a different transaction code is required for a new inquiry, delete any entry in the PAGE field, type the new code letters in the TRANSACTION field (overlaying the old transaction code) and enter your new inquiry.

You can clear the screen between inquiries; however, you must then request the VISION:Inquiry screen again. To do this, press the Clear key, retype the transaction code, IQIO, and enter your new inquiry.

Signing Off

No formal signoff procedure is necessary with VISION:Inquiry. In most installations, you simply leave the terminal. The next user may continue to use VISION:Inquiry, or clear the screen and bring up another system.

If you wish, you may clear the screen with the Clear key before you leave; the next person can then bring up whatever system is needed.

Error Handling

VISION:Inquiry checks an inquiry for several types of errors when it is submitted for execution. For example:

- The inquiry must follow the rules related to VISION:Inquiry and IMS or CICS.
- All words in the inquiry (commands, field names, and so on) must be spelled correctly.
- All databases and fields that are referenced must actually exist and be within the range specified in your User Profile.

If errors are present, a descriptive message displays. Messages from VISION:Inquiry, like the one below, display just below the inquiry:

```
IXX0109       DATABASE NAME NOT FOUND IN INQUIRY
```

The error code IXX0109 identifies this error as processing error #109. The text following the code tells you that the name of the database was omitted from the inquiry. For such errors, type only the corrections, and resubmit the inquiry for execution by pressing the Enter key. In the case above, you would type the correct database name in the inquiry statement, and press Enter.

You can find VISION:Inquiry processing error and informational messages in *Advantage VISION:Inquiry Messages Guide*. The terminology in the error messages is slightly different from that used in this guide.

- **Verb** in the messages is equivalent to command in this guide.
- **Object** refers to database names and field names.
- **Object variable field** is another term for temporary field.
- **Data name** is the same as field name or data field name.

Error messages can also be tailored to the installation. Ask your system administrator to explain any messages you do not understand.

If a message from IMS or CICS, such as the one below, displays at the bottom of the screen, enter the correct information, and resubmit your inquiry.

```
DFS064 NO SUCH TRANSACTION CODE
```

The message may remain on the screen until a new message replaces it, or until the screen is cleared. If you want it to clear, check with your system administrator. If you clear the screen to remove the message, you must re-invoke the VISION:Inquiry screen with the /FORMAT statement (or transaction code for CICS) and reenter your inquiry.

Correcting Errors

When you make an error in an inquiry, VISION:Inquiry cannot always determine what you meant to enter. The message it responds with may not pinpoint exactly where the problem is. This is especially true when the inquiry is complex, and involves sets of parentheses.

Following are a few hints to use if you receive a message that you have difficulty interpreting.

- Remove phrases from the inquiry one at a time, until the error message disappears. Correct the error and add the phrases back, one at a time, until the inquiry is complete.
- Pare down the inquiry to its most basic phrase (that is, just fields to be displayed) and start adding phrases until the error message appears. Correct the error and continue adding phrases until the inquiry is complete.
- If you are working with sets of parentheses, remove them one at a time until you can pinpoint where the error is. Correct the error and start adding the parentheses back.

Operating Modes

There are variations in the way that VISION:Inquiry receives and processes inquiries and returns output:

- Online Inquiry Processing
- Batch Message Inquiry Processing
- Batch Inquiry Processing
- TSO Inquiry Processing

These variations are discussed in this section.

Online Inquiry Processing

Online Inquiry Processing, the most commonly used method, has been discussed up to this point. Inquiries are entered online, executed, and the output is returned to the same terminal (or another terminal, or printer, as requested by the inquiry), while you wait.

The output can be viewed one of two ways:

Continuous Mode	In this mode, the output generated by an online inquiry returns to the terminal without interruption. You simply page forward, viewing each page, until you receive the "END OF INQUIRY" message. The examples in the rest of this Reference Guide are executed in continuous mode.
Conversational Mode	In this mode, the amount of output returned to the terminal is limited by either the number of calls made to the database or the number of pages that are output.

If you are limited by the number of database calls, the following VISION:Inquiry message displays when the maximum number of calls has been made:

```
IXX9123 INQUIRY TIME LIMIT EXCEEDED
```

The following VISION:Inquiry message displays when the maximum number of pages has been displayed:

```
IXX9122 INQUIRY PAGE END
```

After each checkpoint (if the checkpoint facility is included), you can elect to continue with the next group of output lines, or to defer viewing the output at a later time. If the checkpoint facility is not included, the inquiry is stopped and cannot be continued or deferred.

Note: The output from inquiries containing the SORT command always returns in continuous mode.

Your system administrator can tell you whether you are working in continuous or conversational mode of operation, and if the checkpoint facility is included.

Conversational Mode with Checkpoint

The following group of examples illustrate the conversational mode of operation, with the checkpoint facility included. In this instance, the output is limited by the number of database calls.

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD ED.DEGREE SUB.NAME
IF SAL.YTD GE 40000 COUNT (EMP.NAME ED.DEGREE SAL.YTD) TOTAL SAL.YTD
AVERAGE  SAL.YTD TOTAL (EMP.NAME SAL.YTD);

PLANT.ID  EMP.NAME                     SAL.YTD   ED.DEGREE  SUB.NAME
10100     WILLIAM AMES               52,000.00
                                     64,000.00   BA         BUSINESS
** EMP.NAME = WILLIAM AMES
COUNTS     ED.DEGREE    SAL.YTD
                   1          2
TOTALS           SAL.YTD
              116,000.00
          PHYLLIS LOCKMEYER           48,000.00
                                      59,000.00   BA        MARKETING
*
*
*
*
*
 IXX9123  INQUIRY TIME LIMIT EXCEEDED.   (30,2 USER DB CALLS,ROOTS)
```

Figure 4-7 Conversational Mode of Operation

The first part of the output displays in this example. You have reached the maximum number of database calls partway through the output and have received the “INQUIRY TIME LIMIT EXCEEDED” message. VISION:Inquiry stops and waits for you to continue or defer processing the inquiry.

CONTINUE Command

If you want to see more of the output now and you are using IMS, press the PF1 key, or type

```
CONTINUE;
```

and press Enter.

If you are under CICS, enter 1 in the TRANSACTION field to continue an inquiry. The last page of the output containing the inquiry is replaced by the # command, which is a synonym for the CONTINUE command.

In Figure 4-8, “CONTINUE;” has been entered next to INQUIRY on the top line. (If you enter the CONTINUE command so that it overlays the inquiry, erase the rest of the old inquiry or enter two semicolons.) VISION:Inquiry displays the next group of output lines and again runs into a time limit.

```
PLANT.ID  EMP.NAME                      SAL.YTD   ED.DEGREE  SUB.NAME
** EMP.NAME = PHYLLIS LOCKMEYER
COUNTS     ED.DEGREE    SAL.YTD
                   1          2
TOTALS           SAL.YTD
              107,000.00
20150     PETER ZATKIN                44,000.00
                                      50,000.00
                                      56,000.00   BA         ACCOUNTING
                                                  MA         FINANCE
** EMP.NAME = PETER ZATKIN
COUNTS     ED.DEGREE    SAL.YTD
                   2          3
TOTALS           SAL.YTD
              150,000.00
          SUSAN WARE                  41,000.00   BS         ENGR
 IXX9123  INQUIRY TIME LIMIT EXCEEDED.   (60,3 USER DB CALLS,ROOTS)
```

Figure 4-8 CONTINUE Command

You can enter CONTINUE again to view the next portion, or defer processing the next part of the inquiry. VISION:Inquiry saves the rest of the output until you are ready to continue viewing the lines, or until you delete the rest of the inquiry.

DEFER Command

To defer viewing the output while using IMS, press the PF2 key, or type

```
DEFER;
```

and press Enter.

To defer an inquiry under CICS, delete any entries in the PAGE field and enter 2 in the TRANSACTION field. The last page of the output containing the inquiry is replaced by the DI command, which is a synonym for the DEFER command.

In Figure 4-9, the DEFER command has been entered next to INQUIRY.

```
PAGE                 TRANCODE: II         INQUIRY:  DEFER;

IXX0140 INQUIRY IS DEFERRED AS ID '0002'
```

Figure 4-9 DEFER Command

In this example, the DEFER command is entered after receiving the “INQUIRY TIME LIMIT EXCEEDED” message. VISION:Inquiry responds with:

```
IXX0140 INQUIRY IS DEFERRED AS ID '0002'
```

ID Number

The output from this inquiry has been stored under the ID number “2”. Remember this number; you will need it later. ID number 0001 is reserved for the inquiry being executed. The first inquiry deferred is labeled number 2, the next one number 3, and so on.

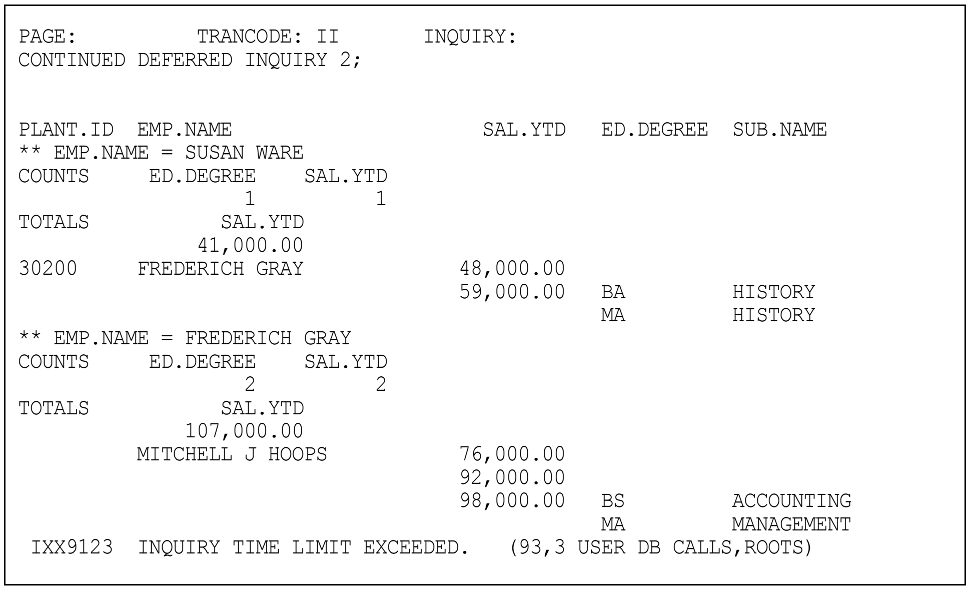
CONTINUE DEFERRED INQUIRY Command

When you are ready to view the rest of the output (and if you are using IMS) type the deferred inquiry number, and press the PF3 key, or enter

```
CONTINUE DEFERRED INQUIRY 2;
```

To continue a deferred inquiry under CICS, delete any entries in the PAGE field, type 3 in the TRANSACTION field, type the deferred inquiry number in the input area, and press Enter to process your request.

VISION:Inquiry responds with the next page of the output, as shown in Figure 4-10, and deletes the stored deferred inquiry.



```
PAGE:          TRANCODE: II       INQUIRY:
CONTINUED DEFERRED INQUIRY 2;


PLANT.ID  EMP.NAME                     SAL.YTD   ED.DEGREE  SUB.NAME
** EMP.NAME = SUSAN WARE
COUNTS     ED.DEGREE    SAL.YTD
                   1          1
TOTALS           SAL.YTD
               41,000.00
30200     FREDERICH GRAY             48,000.00
                                     59,000.00   BA         HISTORY
                                                 MA         HISTORY
** EMP.NAME = FREDERICH GRAY
COUNTS     ED.DEGREE    SAL.YTD
                   2          2
TOTALS           SAL.YTD
              107,000.00
          MITCHELL J HOOPS           76,000.00
                                     92,000.00
                                     98,000.00   BS         ACCOUNTING
                                                 MA         MANAGEMENT
 IXX9123  INQUIRY TIME LIMIT EXCEEDED.   (93,3 USER DB CALLS,ROOTS)
```

Figure 4-10 CONTINUE DEFERRED INQUIRY Command

The system begins where you left off when you deferred the inquiry, and displays the next page of the output. You again receive the “INQUIRY TIME LIMIT EXCEEDED” message. To continue the inquiry to completion, enter the CONTINUE command as explained earlier in this section.

DELETE DEFERRED INQUIRY Command

If you decide you do not need to see any more output from a deferred inquiry, type

```
DELETE DEFERRED INQUIRY 2;
```

and press Enter. (The “2” indicates which inquiry to delete.)

VISION:Inquiry displays the following message when it has taken the appropriate action:

```
IXX0122 DEFERRED INQUIRY ID '2' HAS BEEN DELETED
```

Conversational Mode with UDO

When you are operating in conversational mode under User Defined Output (UDO) and have reached the limit of your database calls, you will receive the “INQUIRY TIME LIMIT EXCEEDED” message. You get this message because no data is output until a whole page has been formatted. See Chapter 11, “User Defined Output” for an explanation of User Defined Output.

SHOW Command

Using UDO, when a database timeout is encountered, you can request a partial report by entering the SHOW command.

```
SHOW;
```

After the SHOW command, the display looks like Figure 4-11:

```
PAGE:          TRANCODE: II       INQUIRY:
SHOW;


             ** TOTAL/AVERAGE SALARIES FOR WOMEN **
                  ** HOLDING COLLEGE DEGREES **
       NAME                DEGREE            SALARIES
PHYLLIS LOCKMEYER            BA              48,000.00
                                             59,000.00
SUSAN WARE                   BS              32,000.00
                                             41,000.00
PATRICIA BLAKELY             BS              30,000.00
NUMBER OF EMPLOYEES:                                 3
TOTAL SALARIES:                             210,000.00
AVERAGE SALARY:                              42,000.00
 IXX0880 END OF SHOW OUTPUT
```

Figure 4-11 Entering the SHOW Command

The SHOW command is an expedient way of looking at summaries and subtotals to see what has been accumulated up to the timeout.

The "END OF SHOW" message tells you that you have seen all the output accessed by that SHOW command. At end-of-page, the SHOW command displays the next available record.

If you wish to see more of the output and you are running IMS, press the PF1 key, or enter the CONTINUE command. Under CICS, delete any entries in the PAGE field and enter 1 in the TRANSACTION field to continue the inquiry.

Figure 4-12 shows the results of continuing a SHOW command.

```
PAGE:          TRANCODE: II       INQUIRY: CONTINUE;


              ** TOTAL/AVERAGE SALARIES FOR WOMEN **
                   ** HOLDING COLLEGE DEGREES **
        NAME                DEGREE             SALARIES
PHYLLIS LOCKMEYER            BA               48,000.00
                                              59,000.00
SUSAN WARE                   BS               32,000.00
                                              41,000.00
PATRICIA BLAKELY             BS               30,000.00
JOAN EVANS                   BS               64,000.00
                             MS               73,200.00
                             PD
MADELYN BATES                BA               32,000.00
MARCIE MORINO                BA               31,000.00
MARTHA WALLINGHAM            BS               33,000.00
NUMBER OF EMPLOYEES:                                  7
 IXX9122  INQUIRY PAGE END.               (80,5 USER DB CALLS,ROOTS)
```

Figure 4-12 Continuing the SHOW Command Output

Output is added to the bottom of the screen; the top of the screen remains the same. When you reach the end of the inquiry page, a new screen with new data appears.

Batch Message Inquiry Processing (IMS only)

With this method, the inquiry is input in the same manner as in IMS online processing from a terminal. Processing and output functions are not performed immediately. A group of inquiries is held in a queue and executed at a later time. Output is in continuous mode.

Batch Inquiry Processing

With this method, input is 80-byte records and output is to the system output device (that is, line printer). Inquiries in the batch mode must be terminated by two semicolons.

TSO Inquiry Processing (IMS only)

With this method, VISION:Inquiry operates under the Time Sharing Option (TSO). If your system works under TSO, your system administrator can explain how to sign on and how the terminal operates. Inquiries under TSO must be terminated by two semicolons (IMS version only).

Chapter 5 Simple Reports

This chapter demonstrates inquiries that produce simple reports.

DISPLAY Command

The DISPLAY command tells VISION:Inquiry to list the contents of one or more fields from the database on either the screen or the printed report. The fields displayed are usually listed in columnar format, left to right.

This is a DISPLAY command:

`DISPLAY`	`PLANT`	`EMP.NAME    EMP.SEX`	`;`
command	database name	fields	termination symbol

This command tells VISION:Inquiry to display the contents of two fields, EMP.NAME and EMP.SEX, from the database named PLANT. The database name, PLANT, identifies the database containing these two fields. The database name is required in every inquiry.

Note that all the examples in this chapter (and in Chapter 6, "Summarizing Data" through Chapter 11, "User Defined Output") are shown in IMS format. The examples use IMS (DL/I) databases, PLANT and SKILL. CICS format is slightly different (see Chapter 4, "Using VISION:Inquiry"). VSAM test files, VSPLANT and VSSKILL, can replace the IMS (DL/I) test databases. Use the appropriate field names, as described in Chapter 2, "Databases and Files".

There are two ways to access test DB2 tables, if your VISION:Inquiry system has a DB2 option:

- Using native VISION:Inquiry syntax, the IMS(DL/I) examples in this guide can be replaced with the test DB2 tables using appropriate field names, as described in Chapter 2, "Databases and Files".
- Using the native SQL syntax facility, which allows the embedded SQL SELECT statement with the test DB2 table columns.

Example:

`DISPLAY`	`EXECSQL`	`SELECT * FROM DYLINQ.IIPLANT`	`ENDEXEC`	`;`
command	beginning delimiter	embedded SQL SELECT statement	ending delimiter	termination symbol

The inquiry format and more detailed information regarding this facility are described in this chapter.

Following are several examples of inquiries and output results.

Example: "Identify all employees by name and sex."

```
PAGE:        TRANCODE: II    INQUIRY:
DISPLAY PLANT EMP.NAME EMP.SEX;



EMP.NAME                    EMP.SEX
WILLIAM AMES                M
PHYLLIS LOCKMEYER           F
MARY ANN THOMAS             F
WILMA FORD                  F
CHARLES SALTER              M
PETER ZATKIN                M
SUSAN WARE                  F
JOHN HENRY CRANE            M
FREDERICH GRAY              M
MITCHELL J HOOPS            M
JANE LOWELL                 F
PATRICIA BLAKELY            F
SHARON DALEY                F
DONALD M KING               M
JOAN EVANS                  F
JONATHAN OAKS               M
```

Figure 5-1 DISPLAY Command

Notice that the fields are output horizontally across the page. Each field name is listed at the top of each column. If you refer to the diagram of the database, you will notice that EMP.NAME and EMP.SEX come from the same segment. There is one occurrence of sex for each occurrence of name.

Displaying Two Segments

In the following example, note that the higher-level segment prints only once when you access segments at different levels. In Figure 5-2, two different segments from two different levels within the database have been accessed. These are the root segment, PLANT (level 1), and one of the dependent segments, EMPLOYEE (level 2).

`DISPLAY PLANT`	`PLANT.ID`	`EMP.NAME   EMP.SEX;`
	level 1	level 2

Example: "List the names and sexes of the employees at each of the plants."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME EMP.SEX;



PLANT.ID  EMP.NAME                     EMP.SEX
10100     WILLIAM AMES                 M
          PHYLLIS LOCKMEYER            F
          MARY ANN THOMAS              F
20150     WILMA FORD                   F
          CHARLES SALTER               M
          PETER ZATKIN                 M
          SUSAN WARE                   F
30200     JOHN HENRY CRANE             M
          FREDERICH GRAY               M
          MITCHELL J HOOPS             M
          JANE LOWELL                  F
          PATRICIA BLAKELY             F
          SHARON DALEY                 F
40300     DONALD M KING                M
          JOAN EVANS                   F
50300     JONATHAN OAKS                M
```

Figure 5-2   Displaying Two Segments of the Database

Not only does the name of each plant appear only once, the data is also grouped by plant. PLANT.ID is called the group field or control field. There are no group fields for VSAM non-hierarchical files and DB2 tables because a VSAM non-hierarchical file and a DB2 table are considered as a single segment database.

If a subfield is used as a group field, the entire field is used instead of the subfield. However, you can assign the subfield to a temporary field (discussed in Chapter 7, "Assignment Statement and Arithmetic Processing") and use the temporary field as a group field. See your system administrator for more information.

Displaying Multiple Segments

If you access multiple segments, VISION:Inquiry creates groups within groups. Figure 5-3 illustrates this with the following statement.

```
DISPLAY PLANT  |  PLANT.ID  |  EMP.NAME EMP.SEX  |  ED.DEGREE;
               |            |                    |
               |  level 1   |      level 2       |   level 3
```

Level 3 is grouped in level 2, and level 2 is grouped in level 1. The printing of the field names of level 1 is automatically suppressed as is the repetitive information from level 2.

Example: "Identify each employee by name and sex. List the employee's educational background and plant location."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME EMP.SEX ED.DEGREE;

PLANT.ID  EMP.NAME                   EMP.SEX  ED.DEGREE
10100     WILLIAM AMES               M        BA
          PHYLLIS LOCKMEYER          F        BA
          MARY ANN THOMAS            F        HS
20150     WILMA FORD                 F        HS
          CHARLES SALTER             M        BA
          PETER ZATKIN               M        BA
                                              MA
          SUSAN WARE                 F        BS
30200     JOHN HENRY CRANE           M        HS
          FREDERICH GRAY             M        BA
                                              MA
          MITCHELL J HOOPS           M        BS
                                              MA
          JANE LOWELL                F        HS
          PATRICIA BLAKELY           F        BS
          SHAR0N DALEY               F        HS
40300     DONALD ME KING             M        BS
```

Figure 5-3 Displaying Multiple Segments

ED.DEGREE is a subgroup of EMP.NAME. If more than one degree is present, the related name is printed only once. The group field or control field suppresses extraneous printing and displays the subgroups this way for readability.

VSAM non-hierarchical files and DB2 tables are treated as having one level only. There can be no group or control fields for VSAM non-hierarchical files or DB2 tables.

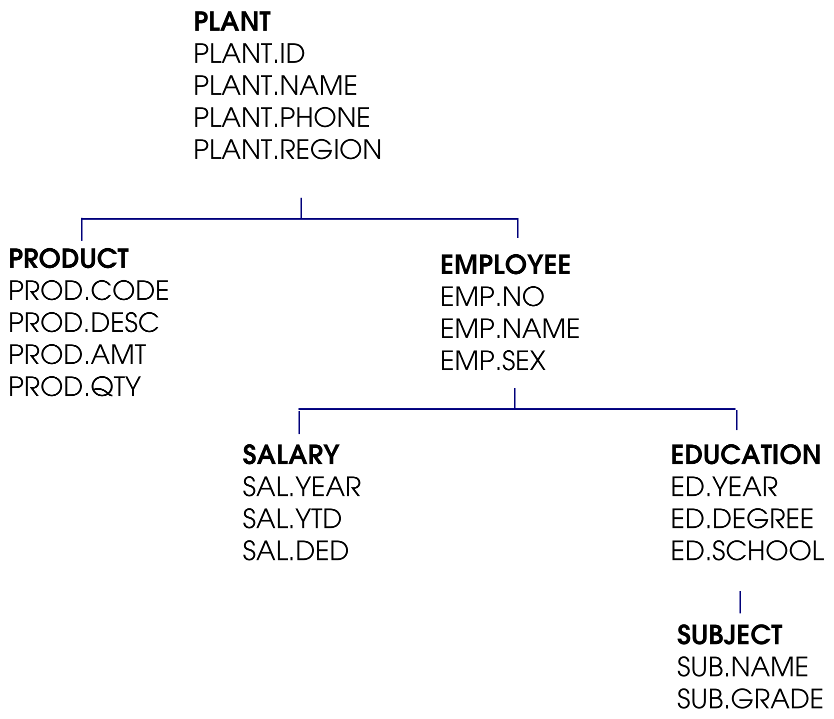


Figure 5-4 The PLANT Database

Figure 5-4 is a diagram of the PLANT database. You can see that the examples in Figure 5-1 through Figure 5-3 accessed only one leg of the database. However, if the data you need happens to be in two legs of the database, the output generated is slightly different.

Accessing Two legs of a Database

The following inquiry statement contains fields from two legs of the database.

`DISPLAY PLANT`	`PLANT.ID`	`PROD.CODE`	`EMP.NAME;`
	root segment	left leg	right leg

This is what that retrieval looks like diagrammatically:

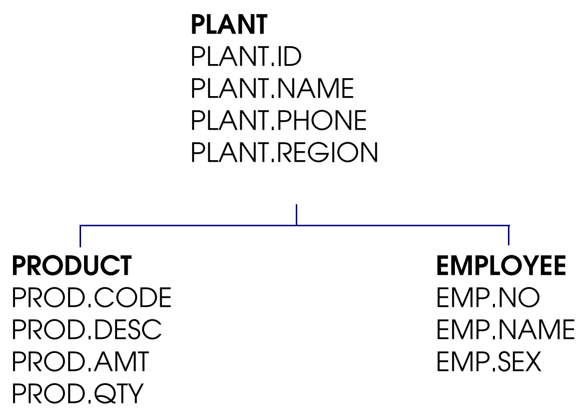


Figure 5-5 Accessing Two Legs of the Database

Figure 5-6 displays the output from this example. The fields in the left leg (PLANT.ID and PROD.CODE) are displayed first in the order in which they are listed in the inquiry statement. The field in the right leg (EMP.NAME) is displayed next. If there were more than one field listed from the right leg, the fields would be displayed in the order in which they were listed in the inquiry statement.

Example: "Identify the employees working at each plant and the products produced."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID PROD.CODE EMP.NAME ;


PLANT.ID  PROD.CODE  EMP.NAME
10100                WILLIAM AMES
                     PHYLLIS LOCKMEYER
                     MARY ANN THOMAS
20150     PO
          RO
          SC
          SJ
          SR
          TR         WILMA FORD
                     CHARLES SALTER
                     PETER ZATKIN
                     SUSAN WARE
30200                JOHN HENRY CRANE
                     FREDERICH GRAY
                     MITCHELL J HOOPS
                     JANE LOWELL
                     PATRICIA BLAKELY
                     SHARON DALEY
40300     AS
```

Figure 5-6 Output of Inquiry Accessing Two Legs of the Database

In this example, the first PLANT.ID (a sales office) has no associated products to display, so only the EMP.NAMEs from that plant appear. However, Plant 20150 is the Remote Control Division. The toys produced there are listed by PROD.CODE. Then the EMP.NAMEs associated with each plant are displayed, with the first EMP.NAME output on the same line as the last PROD.CODE.

Also, note the horizontal display. The field names are output across the page in the following order: PLANT.ID, PROD.CODE, and EMP.NAME. The information for each field is displayed under the field name.

Vertical Display - Row Mode Formatting

When the DISPLAY statement is lengthy and there is not enough room across the page for all the field names to be displayed, they are displayed one after another down the page, producing a vertical report. This vertical report is also called row mode report formatting. Figure 5-7 illustrates a vertical display.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME EMP.SEX ED.DEGREE ED.YEAR
ED.SCHOOL SUB.NAME SUB.GRADE;


PLANT.ID     = 10100
  EMP.NO     = 10103
  EMP.NAME   = WILLIAM AMES
  EMP.SEX    = M
  ED.DEGREE  = BA
  ED.YEAR    = 86
  ED.SCHOOL  = TULANE
  SUB.NAME   = BUSINESS
  SUB.GRADE  = B
  EMP.NO     = 10104
  EMP.NAME   = PHYLLIS LOCKMEYER
  EMP.SEX    = F
  ED.DEGREE  = BA
  ED.YEAR    = 87
  ED.SCHOOL  = WISCONSIN
  SUB.NAME   = MARKETING
  SUB.GRADE  = B
  EMP.NO     = 10105
  EMP.NAME   = MARY ANN THOMAS
  EMP.SEX    = F
```

Figure 5-7 Example of a Vertical Display

For IMS (DL/I) databases and VSAM hierarchical files, the field names are displayed in the order of the hierarchical structure of the database, that is, top to bottom, left to right.

For VSAM non-hierarchical files and DB2 tables, as in horizontal reports, the field names are displayed in the same order in which they appeared in the inquiry statement.

In a vertical display, the group break characteristics disappear and, when SORT is used, field names are suppressed if the field contains all blanks.

Horizontal and Vertical Display with Native SQL Syntax

Figure 5-8 and Figure 5-9 show the horizontal and vertical display examples of inquiries using the native SQL syntax facility of the DB2 option.

```
PAGE:      TRANCODE: II       INQUIRY:
DISPLAY EXECSQL SELECT * FROM DYLINQ.IIEMP_SAL ENDEXEC;;

EMPLOYEE  YEAR  YEAR_TO_DATE   DEDUCTIONS
10103     94       52,000.00     7,400.00
10103     95       64,000.00     9,000.00
10104     94       48,000.00     6,400.00
10104     95       59,000.00     8,200.00
10105     95       15,600.00     1,370.00
21116     94       15,600.00     1,260.00
21116     95       18,800.00     1,980.00
21124     93       24,000.00     2,020.00
21124     94       30,000.00     3,140.00
21124     95       39,000.00     6,000.00
21137     93       44,000.00     6,800.00
21137     94       50,000.00     7,800.00
21137     95       56,000.00     9,580.00
21164     94       32,000.00     5,200.00
21164     95       41,000.00     7,000.00
30201     93       13,400.00     1,380.00
30201     94       19,600.00     1,960.00
30201     95       22,000.00     2,600.00
30202     94       48,000.00     9,580.00
```

Figure 5-8 Example of a Horizontal Display Using the Native SQL Syntax Facility

```
PAGE:      TRANCODE: II       INQUIRY:
D EXECSQL SELECT * FROM DYLINQ.IISALARIES ENDEXEC;;

PLANT_ID        = 10100
  PLANT_NAME    = DALLAS SALES
  PLANT_PHONE   = 4613130
  PLANT_REGION  = SW
  EMP_NO        = 10103
  EMP_NAME      = WILLIAM AMES
  EMP_SEX       = M
  SAL_YEAR      = 94
  SAL_YTD       = 52,000.00
  SAL_DED       = 7,400.00
PLANT_ID        = 10100
  PLANT_NAME    = DALLAS SALES
  PLANT_PHONE   = 4613130
  PLANT_REGION  = SW
  EMP_NO        = 10103
  EMP_NAME      = WILLIAM AMES
  EMP_SEX       = M
  SAL_YEAR      = 95
  SAL_YTD       = 64,000.00
  SAL_DED       = 9,000.00
```

Figure 5-9 Example of a Vertical Display Using the Native SQL Syntax Facility

Displaying Title Lines

You can have one or two title lines at the top of each page of your report. The title line is a character constant enclosed between single quotation marks and can be up to 70 characters. A title line of more than 70 characters will be truncated on the right. The title lines must be the first one or two items in the inquiry statement following the DISPLAY command or the data base name. The title lines are only displayed for the horizontal display (column mode formatting) and ignored for the vertical display (row mode formatting). Figure 5-10 shows the output of an inquiry with a title line.

```
PAGE:      TRANCODE: II       INQUIRY:
DISPLAY PLANT `       REPORT OF EMPLOYEES WITH YEARLY SALARIES'
PLANT.ID EMP.NO EMP.NAME SAL.YTD;;

      REPORT OF EMPLOYEES WITH YEARLY SALARIES
PLANT.ID  EMP.NO   EMP.NAME                        SAL.YTD
10100      10103   WILLIAM AMES                  52,000.00
                                                 64,000.00
           10104   PHYLLIS LOCKMEYER             48,000.00
                                                 59,000.00
           10105   MARY ANN THOMAS               15,600.00
20150      21116   WILMA FORD                    15,600.00
                                                 18,800.00
           21124   CHARLES SALTER                24,000.00
                                                 30,000.00
                                                 39,000.00
           21137   PETER ZATKIN                  44,000.00
```

Figure 5-10 Example of a Report With a Title Line

Title line(s) can also be used with native SQL syntax facility as shown in Figure 5-11.

```
PAGE:      TRANCODE: II       INQUIRY:
DISPLAY `       REPORT OF EMPLOYEES WITH YEARLY SALARIES'
`            AND SALARY DEDUCTIONS'
EXECSQL SELECT * FROM DYLINQ.IIEMP_SAL ENDEXEC;;

       REPORT OF EMPLOYEES WITH YEARLY SALARIES
            AND SALARY DEDUCTIONS
EMPLOYEE  YEAR  YEAR_TO_DATE  DEDUCTIONS
10103     94       52,000.00    7,400.00
10103     95       64,000.00    9,000.00
10104     94       48,000.00    6,400.00
10104     95       59,000.00    8,200.00
10105     95       15,600.00    1,370.00
21116     94       15,600.00    1,260.00
21116     95       18,800.00    1,980.00
21124     93       24,000.00    2,020.00
21124     94       30,000.00    3,140.00
21124     95       39,000.00    6,000.00
21137     93       44,000.00    6,800.00
```

Figure 5-11 Example of a Native SQL Syntax Facility Report With Title Lines

DISPLAY Command Summary

For inquiries that do not use the native SQL syntax facility:

DISPLAY or D	The DISPLAY command displays information from the database.
database name	This is the name of the database containing the information to be listed. (If the database is the first one in the directory, its name may be omitted here.)
title line	One or two character constants (up to 70 characters) to be displayed at the top of each page of output.
display list	The list of information to be displayed can include the contents of one or more fields from the database, temporary fields, or the results of arithmetic calculations. (Temporary fields and arithmetic processing are discussed in Chapter 7, "Assignment Statement and Arithmetic Processing".)

The DISPLAY command allows both horizontal and vertical reports.

For inquiries which use the native SQL syntax facility of the DB2 option:

DISPLAY or **D**	The DISPLAY command displays information from the DB2 table.
title line	One or two character constants (up to 70 characters) to be displayed at the top of each page of output.
EXECSQL	The starting delimiter of the SQL SELECT statement.
SELECT statement	Any valid SQL SELECT statement. Refer to the appropriate IBM manual for description and syntax of SQL SELECT statement.
ENDEXEC	The end delimiter of the SQL SELECT statement.

Notes:

- Separate the commands, delimiters, and SELECT statement that make up an inquiry statement from each other by at least one blank space.
- When an inquiry statement is too long to fit on one line on the terminal screen, it automatically wraps around onto subsequent lines, sometimes splitting commands, delimiters, or elements of the SELECT statement.
- Terminate each inquiry statement with a semicolon (;). Systems operating in batch mode require two semicolons. You may use two semicolons after your new inquiry. VISION:Inquiry ignores whatever follows two semicolons.
- The DISPLAY command allows both horizontal and vertical reports.
- Title lines are ignored for vertical reports.
- The output column headings in horizontal display and the field names in vertical display appear as column names in the SQL Descriptor Area (SQLDA). When an expression or column function is used, there is no column name in SQLDA. In this case, the character "." appears as the column name in horizontal display or field heading in vertical display. See the appropriate IBM manual for the description and content of the SQL Descriptor Area (SQLDA).
- When DISPLAY is requested for a column with null values, the output will be "????".

Conditional Selection Phrase

This section does not apply to the native SQL syntax facility. See the appropriate IBM manual for a description and syntax of the conditional selection clause (WHERE) in the SQL SELECT statement.

Conditional selection allows you to select only specific data for processing. By adding conditional selection phrases to the inquiry statement, only the information that meets the conditions in the phrase is displayed.

Conditional selection indicates that only certain data is to be processed by an inquiry statement.

Look at one such statement:

DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD | $\underbrace{\texttt{IF SAL.YEAR = 94;}}_{\text{conditional phrase}}$

The conditional selection begins at the word IF and includes a list of one or more conditions. The list of conditions is called a logical expression. This is what the logical expression in the example looks like:

$$\underbrace{\texttt{IF SAL.YEAR = 94;}}_{\substack{\text{logical expression} \\ \text{(with one condition)}}}$$

The condition in the phrase (if the SAL.YEAR = 94) is tested against each occurrence of the specified field (in this case, SAL.YEAR). The data is processed only if the field contains the specified value.

This is what the statement is asking for:

do this . . .	. . . if this is true
`DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD`	`IF SAL.YEAR = 94;`

Example: "List the employees who worked during 1994. Include the employee's salary and plant location."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD IF SAL.YEAR = 94;

PLANT.ID  EMP.NAME                       SAL.YTD
10100     WILLIAM AMES                 52,000.00
          PHYLLIS LOCKMEYER            48,000.00
20150     WILMA FORD                   15,600.00
          CHARLES SALTER               30,000.00
          PETER ZATKIN                 50,000.00
          SUSAN WARE                   32,000.00
30200     JOHN HENRY CRANE             19,600.00
          FREDERICH GRAY               48,000.00
          MITCHELL J HOOPS             92,000.00
          JANE LOWELL                  22,000.00
40300     DONALD M KING                66,000.00
          JOAN EVANS                   64,000.00
50300     JONATHAN OAKS                36,000.00
60200     DAVID YORK                   37,000.00
          RUSSELL M SIMMONS            48,000.00
```

Figure 5-12 DISPLAY Command with a Conditional Selection Phrase

Only those employees who worked during 1994 have been displayed, along with the salaries they received in 1994, and their plant locations. Without the conditional selection phrase, all occurrences of PLANT.ID, EMP.NAME, and SAL.YTD would have been reported.

The Positional Test

The positional test does not apply to VSAM non-hierarchical files and DB2 tables.

The FIRST and LAST positional test allows you to obtain the first or last occurrence of a segment. This test, which is another form of relational expression, may only be used with lower level segments.

The format of the positional test is:

```
IF FIELD.NAME FIRST
```

or

```
IF FIELD.NAME LAST
```

How is this incorporated into an inquiry statement? Suppose you wanted to list the year that each employee began working at the toy company. To obtain that list, use the FIRST positional test.

```
DISPLAY PLANT  PLANT.ID  EMP.NAME  SAL.YTD SAL.YEAR  IF SAL.YEAR FIRST;
```

FIRST positional test

In the following example, Figure 5-13 displays the output.

Example: “List the year each employee began working. Include the employee's plant location and the salary received when first employed.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD SAL.YEAR
IF SAL.YEAR FIRST;

PLANT.ID  EMP.NAME                        SAL.YTD   SAL.YEAR
10100     WILLIAM AMES                  52,000.00         94
          PHYLLIS LOCKMEYER             48,000.00         94
          MARY ANN THOMAS               15,600.00         95
20150     WILMA FORD                    15,600.00         94
          CHARLES SALTER                24,000.00         93
          PETER ZATKIN                  44,000.00         93
          SUSAN WARE                    32,000.00         94
30200     JOHN HENRY CRANE              13,400.00         93
          FREDERICH GRAY                48,000.00         94
          MITCHELL J HOOPS              76,000.00         93
          JANE LOWELL                   22,000.00         94
          PATRICIA BLAKELY              30,000.00         95
          SHARON DALEY                  17,000.00         95
40300     DONALD M KING                 58,000.00         93
          JOAN EVANS                    64,000.00         94
50300     JONATHAN OAKS                 36,000.00         94
          MADELYN BATES                 32,000.00         95
          VICKY WARD                    20,000.00         95
60200     DAVID YORK                    31,000.00         93
```

Figure 5-13 Using the FIRST Positional Test

Figure 5-13 shows the first year each employee began working for the toy company, and the salary earned.

Now suppose that you are interested in knowing the highest educational degree each of those employees attained. You would use the LAST positional test to obtain this information.

```
DISPLAY PLANT PLANT.ID EMP.NAME  ED.DEGREE    IF ED.DEGREE LAST;
```

LAST positional test

In the following example, Figure 5-14 displays the output.

Example: "List the last educational degree each of the employees has attained. Also list the employee's plant location."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE IF ED.DEGREE LAST;


PLANT.ID  EMP.NAME                    ED.DEGREE
10100     WILLIAM AMES                BA
          PHYLLIS LOCKMEYER           BA
          MARY ANN THOMAS             HS
20150     WILMA FORD                  HS
          CHARLES SALTER              BA
          PETER ZATKIN                MA
          SUSAN WARE                  BS
30200     JOHN HENRY CRANE            HS
          FREDERICH GRAY              MA
          MITCHELL J HOOPS            MA
          JANE LOWELL                 HS
          PATRICIA BLAKELY            BS
          SHARON DALEY                HS
40300     DONALD M KING               PD
          JOAN EVANS                  PD
50300     JONATHAN OAKS               BS
          MADELYN BATES               BA
          VICKY WARD                  HS
60200     DAVID YORK                  BA
```

Figure 5-14   Using the LAST Positional Test

Figure 5-14 lists the last educational degree each employee achieved.

Every database is organized differently. See your system administrator to find out how yours is organized so that you can use the FIRST and LAST positional tests.

The Existence Test

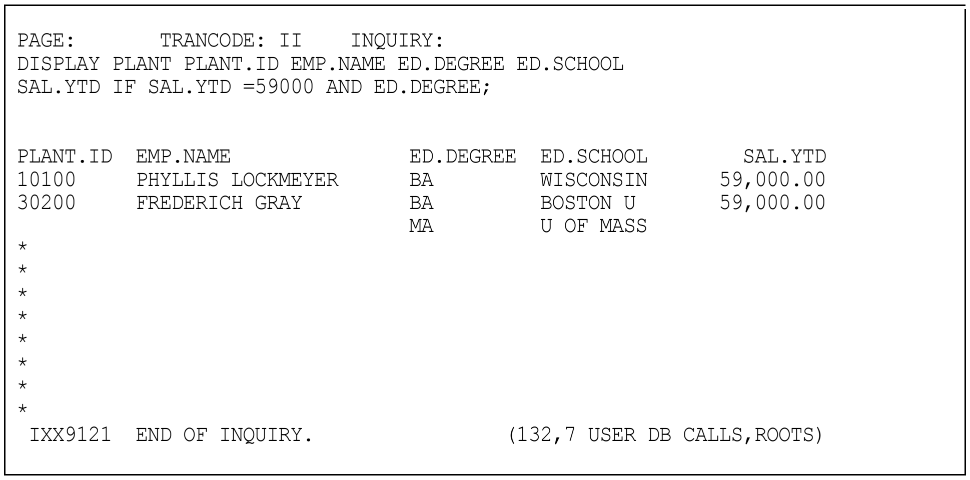
Note: The existence test does not apply to DB2 tables nor to VSAM hierarchical or non-hierarchical files.

Another form of logical expression is the existence test. It is used to determine whether or not there are segments present that contain a field. To test for the existence of a field, simply add a field name with no associated relational operator to your inquiry. An existence test looks like this:

```
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE ED.SCHOOL SAL.YTD
IF SAL.YTD = 59000 AND ED.DEGREE;
```

where “ED.DEGREE” is the existence test.

Example: “List the employees who have educational backgrounds and who earn $59,000. Include the employee's name, plant, degree, school, and salary.”



```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE ED.SCHOOL
SAL.YTD IF SAL.YTD =59000 AND ED.DEGREE;


PLANT.ID  EMP.NAME               ED.DEGREE  ED.SCHOOL        SAL.YTD
10100     PHYLLIS LOCKMEYER      BA         WISCONSIN      59,000.00
30200     FREDERICH GRAY         BA         BOSTON U       59,000.00
                                 MA         U OF MASS
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (132,7 USER DB CALLS,ROOTS)
```

Figure 5-15 Using the Existence Test

Because the ED.DEGREE field exists in a segment of the PLANT database, it was displayed by the inquiry.

Compound Conditional Selection

The logical expression in the conditional selection phrase may test for more than one condition. Two or more conditions are connected using the AND operator. All conditions must be met before the data is displayed.

```
DISPLAY PLANT PLANT.ID EMP.NAME SUB.NAME  IF EMP.SEX = 'F' AND ED.DEGREE = 'BS';
```

conditional phrase that
tests for two conditions

Example: “List all women holding a BS degree. Also list their plant location and their major in college.”

```
PAGE:       TRANCODE: II    INQUIRY:
 DISPLAY PLANT PLANT.ID EMP.NAME SUB.NAME IF EMP.SEX = 'F' AND
 ED.DEGREE = 'BS';


 PLANT.ID  EMP.NAME                   SUB.NAME
 20150     SUSAN WARE                 ENGR
 30200     PATRICIA BLAKELY           BUSINESS
 40300     JOAN EVANS                 PHYSICS
 70500     MARTHA WALLINGHAM          ENGR
 *
 *
 *
 *
 *
 *
 IXX9121  END OF INQUIRY.               (81,7 USER DB CALLS,ROOTS)
```

Figure 5-16 Using a Compound Selection Clause Within the Conditional Selection Phrase

Only those employees who meet both conditions are listed in Figure 5-16.

Note that in this statement the relational expressions contain the character constants F and BS. Character constants are alphanumeric data and must be enclosed within single quotation marks in the inquiry statement.

Relational and Logical Operators

Each conditional test in the logical expression is called a relational expression.

A relational expression is composed of two values connected by a relational operator. The values can be data fields, character or numeric constants, or arithmetic expressions. (Arithmetic expressions are defined in Chapter 7, "Assignment Statement and Arithmetic Processing".)

A relationship test expresses a condition or association between two terms. If the relationship specified by the test is true, the expression is true. If the relationship is false, the expression is false.

`... IF EMP.SEX = 'F'`	`AND ED.DEGREE = 'BS';`
relational expression	relational expression

logical expression

The relational operators used in VISION:Inquiry are.

Symbol	**Meaning**
LT or <	Less than
LE or <=	Less than or equal to
EQ, EQUAL, or =	Equal to
GE or >=	Greater than or equal to
GT or >	Greater than
NE or ¬=	Not equal to
LIKE	Partial matching for character fields

Relational expressions in a logical expression are connected by the logical operators AND and OR.

- Conditions connected by AND must all be true for the logical expression to be true. The expressions in Figure 5-16 are connected by AND; both must be true.
- At least one of the conditions connected by OR must be true for the logical expression to be true.

Using the LIKE Operator

The LIKE operator can be used for partial matching of the character fields.

The % and _ characters have a special meaning when used in the search string with the LIKE operator for partial matching. The % character indicates any string of zero or more characters. The _ character indicates any single character. There might be a case when the "%" and "_" characters are part of the search string. For this case, there is an escape character that designates a real "%" or "_" character immediately following the escape character. The escape character can be any character and is preceded by the ESCAPE keyword in the inquiry. If more than one escape character is specified in a query, the last one takes effect and the others are ignored. If the escape character is specified in a query with no LIKE operator in the IF criteria, the escape character is ignored and the process continues. If neither of the % and _ characters is used in the IF criteria with the LIKE operator, the LIKE operator is treated as the "=" operator and the process continues.

When the escape character is also part of the search string, it must be keyed in as two escape characters, such as ++. Some examples of the character constants used with the LINK operator are:

IF PLANT.NAME LIKE '%PRODUCTS%'	Plant names containing the word PRODUCTS
IF PLANT.NAME LIKE 'BASIC%'	Plant names starting with the word BASIC
IF SKILL.NAME LIKE '%CLERK'	Skill names ending with the word CLERK
IF SKILL.NAME LIKE '%FILE_CLERK%'	Skill names containing the word FILE and CLERK separated with one character
IF SKILL.NAME LIKE '%FILE+_CLERK%' ESCAPE '+'	Skill names containing the word FILE_CLERK
IF SKILL.NAME LIKE 'C_D,,FILE%,_CLERK' ESCAPE ','	Skill names containing the character 'C' as the first character and the word 'D,FILE' starting from the third character and end with the word '_CLERK'.

Evaluating Logical Expressions

Logical expressions are evaluated according to a **hierarchy of operations**, which follows the rules of Boolean logic. This hierarchy depends upon the priority of the operators, as follows:

- Expressions are evaluated from left to right:
  Expressions enclosed in parentheses are evaluated first.
  If parentheses are nested, the inner set is evaluated first.
- Arithmetic operators are evaluated in the following order:
  Multiplication (*) and division (/) are performed first.
  Addition (+) and subtraction (-) are performed next.
- Relational operators are evaluated:
  EQ, NE, LT, GT, LE, and GE
- Logical operators are evaluated:
  AND
  OR

The following expressions are evaluated following the above rules:

Expression:	A and B or C	A and B	or C
Performed in this order:		1st	2nd

If you add parentheses to the expression, the parenthetical segment is evaluated first:

Expression:	A and (B or C)	A and	(B or C)
Performed in this order:		2nd	1st

If sets of parentheses are nested, the innermost set is evaluated first.

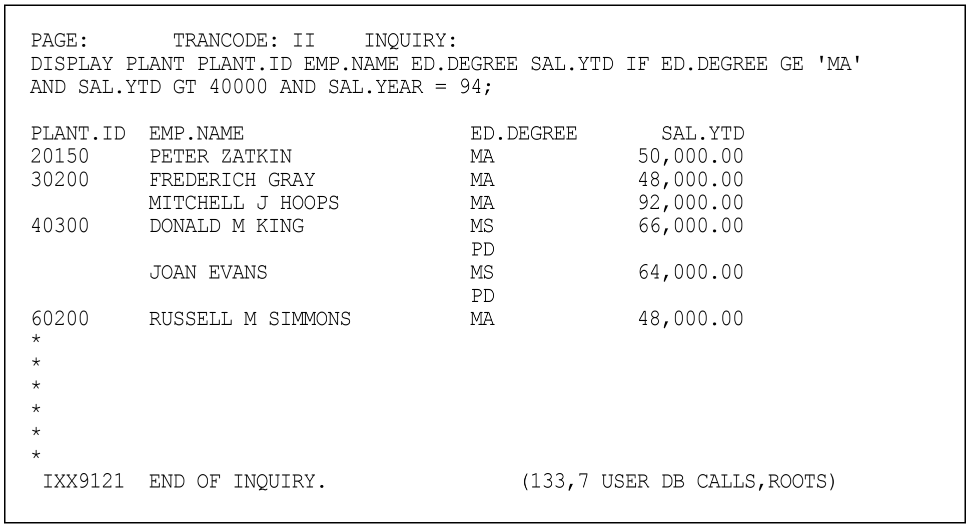
Using the AND Operator

Consider the following statement that uses the AND logical operator:

```
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SAL.YTD IF
ED.DEGREE GE 'MA' AND SAL.YTD GT 20000 AND SAL.YEAR = 94;
```

Figure 5-17 displays only those people who fulfill all three conditions of the conditional selection phrase. The relational expressions connected by AND must all be true: educational degree must be equal to or greater than MA, earnings must be more than $40,000, and the salary year must be 1994.

Example: “List the names, educational degrees, salaries, and plant identification for all employees who worked in 1994 if their degree is equal to or greater than Master of Arts and their yearly salary is greater than $40,000.”



```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SAL.YTD IF ED.DEGREE GE 'MA'
AND SAL.YTD GT 40000 AND SAL.YEAR = 94;

PLANT.ID  EMP.NAME                   ED.DEGREE       SAL.YTD
20150     PETER ZATKIN               MA            50,000.00
30200     FREDERICH GRAY             MA            48,000.00
          MITCHELL J HOOPS           MA            92,000.00
40300     DONALD M KING              MS            66,000.00
                                     PD
          JOAN EVANS                 MS            64,000.00
                                     PD
60200     RUSSELL M SIMMONS          MA            48,000.00
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (133,7 USER DB CALLS,ROOTS)
```

Figure 5-17 Using the AND Logical Operator

Using the OR Operator

If you need to locate people who meet at least one of several conditions, use OR as the logical operator to link the two relational expressions together. This is how it works:

```
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SAL.YTD
IF ED.DEGREE = 'HS' OR SAL.YTD LE 42000;
```

This is what the statement is requesting:

do this:	`DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SAL.YTD`
if this is true:	`IF ED.DEGREE = 'HS'`
or this is true:	`OR SAL.YTD LE 42000;`

Only one of the two relational expressions has to be true. People are selected from the database only if they are high school graduates or if their earnings are less than or equal to the stated amount (or both).

Example: "List the employees who have a high school education or who earn less than or equal to $42,000 a year. Also list their plant identification, educational degrees, and salaries."

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SAL.YTD IF
ED.DEGREE EQ 'HS' OR SAL.YTD LE 42000;


PLANT.ID  EMP.NAME                    ED.DEGREE       SAL.YTD
10100     MARY ANN THOMAS             HS            15,600.00
20150     WILMA FORD                                15,600.00
                                      HS            18,800.00
          CHARLES SALTER                            24,000.00
                                                    30,000.00
                                      BA            39,000.00
          SUSAN WARE                                32,000.00
                                      BS            41,000.00
30200     JOHN HENRY CRANE                          13,400.00
                                                    19,600.00
                                      HS            22,000.00
          JANE LOWELL                               22,000.00
                                      HS            26,000.00
          PATRICIA BLAKELY            BS            30,000.00
          SHARON DALEY                HS            17,000.00
```

Figure 5-18 Using the OR Logical Operator

In the ED.DEGREE column, notice some BA and BS degrees listed in addition to high school diplomas. The people with college educations have been displayed because they fulfill the second half of the conditional selection phrase: they earn $42,000 a year or less.

Using Parentheses

To evaluate salaries of all the women from the database whose educational degrees equaled MS or who worked during 1994, make use of parentheses, as in the following:

```
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND
(ED.DEGREE = 'MS' OR SAL.YEAR = 94);
```

This is what the statement is requesting:

do this: `DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD`

if this is true: `IF EMP.SEX = 'F'` *(2nd)*

and either of these is true: `(ED.DEGREE='MS' OR SAL.YEAR=94);` *(1st)*

The expression in parentheses in Figure 5-19 is evaluated first. Because the connector within the parentheses is OR, only one of the conditions has to be true. Therefore, those employees holding a Master of Science degree or those who had worked during 1994 are selected.

AND connects the two halves of the logical expression; both halves of the conditional expression have to be true for the statement to be true. This means that only women who hold an MS degree or who worked during 1994 are selected.

Example: “Identify all the women whose educational degrees equal MS or who worked during 1994. Also list their salaries.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX EQ 'F' AND
(ED.DEGREE = 'MS' OR SAL.YEAR = 94);;

EMP.NAME                   ED.DEGREE       SAL.YTD
PHYLLIS LOCKMEYER          BA            48,000.00
WILMA FORD                 HS            15,600.00
SUSAN WARE                 BS            32,000.00
JANE LOWELL                HS            22,000.00
JOAN EVANS                 BS            64,000.00
                           MS
                           PD
KAREN REDFERN              HS            22,000.00
AGNES COVINGTON            HS            20,000.00
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (120,7 USER DB CALLS,ROOTS)
```

Figure 5-19 Using Parentheses Within the Conditional Selection Phrase

Parentheses are important in an inquiry statement because the conditions contained within the parentheses are evaluated first. This becomes obvious when the previous statement is entered without using parentheses.

```
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F'
AND ED.DEGREE = 'MS' OR SAL.YEAR = 94;
```

This is what the statement is requesting:

do this: `DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD`

if both of these are true: `IF EMP.SEX='F' AND ED.DEGREE='MS'` *(1st)*

or this is true: `OR SAL.YEAR=94;` *(2nd)*

The statement without the parentheses is requesting that women holding Master of Science degrees or people working for the company during 1994 be selected. Figure 5-20 shows that report.

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD IF EMP.SEX = 'F' AND
ED.DEGREE = 'MS' OR SAL.YEAR = 94;

EMP.NAME                   ED.DEGREE       SAL.YTD
WILLIAM AMES               BA            52,000.00
PHYLLIS LOCKMEYER          BA            48,000.00
WILMA FORD                 HS            15,600.00
CHARLES SALTER             BA            30,000.00
PETER ZATKIN               BA            50,000.00
                           MA
SUSAN WARE                 BS            32,000.00
JOHN HENRY CRANE           HS            19,600.00
FREDERICH GRAY             BA            48,000.00
                           MA
MITCHELL J HOOPS           BS            92,000.00
                           MA
JANE LOWELL                HS            22,000.00
DONALD M KING              BS            66,000.00
                           MS
```

Figure 5-20 Omitting the Parentheses from Within the Conditional Selection Phrase

When the statement was run without parentheses, the logical expressions within the conditional selection phrase were evaluated differently from those in the previous example.

The previous examples accessed only one leg of the database, the one containing the EMPLOYEE, SALARY, EDUCATION, and SUBJECT segments.

Selecting Two Legs of a Database

Now look at an example of two legs of the database being accessed with the following statement.

```
DISPLAY PLANT PROD.CODE PROD.DESC EMP.NAME IF ED.DEGREE = 'MA';
```

This inquiry statement accesses the PRODUCT and EMPLOYEE segments of the database. The conditional selection phrase, however, limits the selection criterion to the second leg by asking that only those employees with Master of Arts degrees be chosen.

Example: “For those employees holding MA degrees, list the employee name, product code, and product description of the toys produced.”

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PROD.CODE PROD.DESC EMP.NAME IF
ED.DEGREE = 'MA';


PROD.CODE  PROD.DESC                  EMP.NAME
PO         POST OFFICE
RO         ROBOTS
SC         SLOT CARS
SJ         SKI JUMP
SR         SKATING RINK
TR         TOBOGGAN RUN               PETER ZATKIN
                                      FREDERICH GRAY
                                      MITCHELL J HOOPS
DA         DOLL ACCESSORIES
DC         DOLL CLOTHES
DF         DOLL HOUSE FURNITURE
DH         DOLL HOUSE
DO         DOLL
KS         KALEIDOSCOPE
PB         PADDLEBAT
PG         PUNCHING BAG
TB         TEDDY BEAR                 RUSSEL M SIMMONS
*
IXX9121  END OF INQUIRY            (84,7 USER DB CALLS, ROOTS)
```

Figure 5-21 Selecting on Two Legs of a Database

In Figure 5-21, the data was limited to one leg of the database by the conditional selection phrase (IF ED.DEGREE = ‘MA’). Therefore, only those employees with Master of Arts degrees were accessed from the database. The product codes and descriptions that those employees produced were accessed because PROD.CODE and PROD.DESC were listed in the inquiry statement.

Conditional Selection Phrase Summary

IF or **WITH**	Only specific data is to be accessed.
logical expression	The conditions the data must meet before it is accessed.

A character constant in a conditional phrase must be enclosed in single quotation marks.

Expressions are evaluated according to a hierarchy of operations:

- From left to right.
- Expressions in parentheses are evaluated first.
- Arithmetic operators are evaluated: * and / first, then + and -.
- Relational operators are evaluated next:

  EQ, NE, LT, GT, LE, GE, LIKE
- Logical operators are evaluated next:

  AND, OR

Conditions connected by AND must all be true before the data is processed.

At least one of the conditions connected by OR must be true for the data to be selected.

A relational expression is part of a conditional selection phrase and is composed of two values connected by a relational operator. The relational operators are:

Symbol	Meaning
LT or <	Less than
LE or <=	Less than or equal to
EQ, EQUAL, or =	Equal to
GE or >=	Greater than or equal to
GT or >	Greater than
NE or ¬ =	Not equal to
LIKE	Partial matching for character fields

Relational expressions composed of fields from different segments are not allowed.

As a result of evaluating all of the selection clauses, the phrase is either true or false. If the final condition is true, the record is processed; if it is false, the record is bypassed.

When comparing two fields from the segment, the right field type is converted to the left field type before the comparison is made. Computer Associates recommends that comparing two unlike field types be kept to a minimum.

SORT Command

Note: This section does not apply to the native SQL syntax facility. See the appropriate IBM manual for a description and syntax of the sort clause (ORDER BY) in the SQL SELECT statement.

Displaying Fields in Ascending or Descending Sequence

To display data in a sequence different from the one in which it is stored in the database, use the SORT command. The SORT command tells VISION:Inquiry which fields you want sorted and how you want them sequenced.

You use the SORT command to sequence fields based on one or more group fields or control fields.

- The first field that you specify is called the **primary sort field.**
- The next field specified is called the **secondary sort field.**

The SORT command sequences the data first in the order of the primary field, and then in the order of the secondary field.

The fields to be sorted follow the SORT command from left to right in the order you want them sequenced. Each field may be sorted in ascending or descending order.

Using SORT with DISPLAY

SORT must be used with the DISPLAY command.

SORT can be used with any field name in the database, except a subfield. If a subfield is used with SORT, the entire field is used instead of the subfield. However, you can assign the subfield to a temporary field (discussed in Chapter 7, "Assignment Statement and Arithmetic Processing") and use the temporary field as a group field. See your system administrator for more information.

Ascending or Descending Order

SORT can be formatted in either ascending (ASC) or descending (DSC) order.

- Ascending order means that numeric data goes from a lower to a higher value (0-9) as it performs the sequencing; alpha characters start with A and go to Z.
- Descending order means that numeric data goes from a higher value to a lower one as it performs the sequencing; alpha characters begin with Z and go to A. Numeric characters are higher than alpha characters in sequence.

VISION:Inquiry automatically assumes ASC unless DSC is specified in the inquiry.

See Chapter 6, "Summarizing Data" for information on SORT with subtotals.

Using the SORT Command

To sort data, insert the SORT command immediately in front of the field name or names you want sequenced. This is how to write a SORT command in an inquiry statement:

`DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD`	`SORT`	`SAL.YTD`
`IF SAL.YEAR = 94;`	command	field to be sorted in ASC order

Because neither ASC nor DSC is specified, SORT defaults to ASC and sorts in ascending sequence.

Example: “List all employees who worked during 1994. Show their plant identification and their salaries. Arrange the listing in ascending order by salary.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD SORT SAL.YTD IF SAL.YEAR = 94;

PLANT.ID  EMP.NAME                        SAL.YTD
20150     WILMA FORD                    15,600.00
70500     STEPHEN MCGEE                 19,000.00
30200     JOHN HENRY CRANE              19,600.00
70500     AGNES COVINGTON               20,000.00
30200     JANE LOWELL                   22,000.00
60200     KAREN REDFERN                 22,000.00
20150     CHARLES SALTER                30,000.00
          SUSAN WARE                    32,000.00
50300     JONATHAN OAKS                 36,000.00
60200     DAVID YORK                    37,000.00
70500     RONALD T JACKSON              46,000.00
10100     PHYLLIS LOCKMEYER             48,000.00
30200     FREDERICH GRAY                48,000.00
60200     RUSSELL M SIMMONS             48,000.00
20150     PETER ZATKIN                  50,000.00
10100     WILLIAM AMES                  52,000.00
40300     JOAN EVANS                    64,000.00
          DONALD M KING                 66,000.00
30200     MITCHELL J HOOPS              92,000.00
```

Figure 5-22   Sorting on SAL.YTD

In Figure 5-22, SAL.YTD is the field being sorted. Notice that SAL.YTD is sorted in ascending order with the salary of $15,600 being displayed first, followed by $19,000, and then by $19,600.

Comparing Sorted and Unsorted Output

Now compare the output you just looked at to the output of the same statement without the SORT command (Figure 5-23).

Example: "List all employees who worked during 1994. Show their plant locations and their salaries."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD IF SAL.YEAR = 94;



PLANT.ID  EMP.NAME                       SAL.YTD
10100     WILLIAM AMES                 52,000.00
          PHYLLIS LOCKMEYER            48,000.00
20150     WILMA FORD                   15,600.00
          CHARLES SALTER               30,000.00
          PETER ZATKIN                 50,000.00
          SUSAN WARE                   32,000.00
30200     JOHN HENRY CRANE             19,600.00
          FREDERICH GRAY               48,000.00
          MITCHELL J HOOPS             92,000.00
          JANE LOWELL                  22,000.00
40300     DONALD M KING                66,000.00
          JOAN EVANS                   64,000.00
50300     JONATHAN OAKS                36,000.00
60200     DAVID YORK                   37,000.00
          RUSSELL M SIMMONS            48,000.00
          KAREN REDFERN                22,000.00
70500     RONALD T JACKSON             46,000.00
          STEPHEN MCGEE                19,000.00
          AGNES COVINGTON              20,000.00
```

Figure 5-23 Output Without SORT Invoked

As Figure 5-23 illustrates, no sort action was applied to any field within the inquiry statement. Therefore, all you have displayed is a listing of those employees who worked for the company during 1994 and the salaries they received. The PLANT.ID field is listed in ascending order because that is the way it is stored in the database.

When you compare the output of Figure 5-22 to that of Figure 5-23, you see that the SORT command allows you to display the data in a sequence different from the way it is stored in the database.

Using Multiple SORT Commands

The SORT command can be used for any field or combination of fields in the database.

When two or more SORT commands are written into a single inquiry statement, the primary field is sorted first, the secondary field is sorted within the already sorted primary field, and so on. Multiple SORT commands sort groups within groups until all the groups have been sequenced.

```
DISPLAY PLANT PLANT.ID PLANT.REGION EMP.NAME SAL.YTD

  SORT PLANT.REGION      SORT SAL.YTD      IF SAL.YEAR = 94;
```

| primary sort | secondary sort |

Example: "List the employees by salary year to date, plant location, and plant region if the employee worked in 1994. Sort the data by plant region and by salary within each region."

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID PLANT.REGION EMP.NAME SAL.YTD
SORT PLANT.REGION SORT SAL.YTD IF SAL.YEAR =94;

PLANT.ID  PLANT.REGION  EMP.NAME                      SAL.YTD
20150     MW            WILMA FORD                  15,600.00
                        CHARLES SALTER              30,000.00
                        SUSAN WARE                  32,000.00
                        PETER ZATKIN                50,000.00
40300     NC            JOAN EVANS                  64,000.00
                        DONALD M KING               66,000.00
60200     NE            KAREN REDFERN               22,000.00
                        DAVID YORK                  37,000.00
                        RUSSELL M SIMMONS           48,000.00
50300     NW            JONATHAN OAKS               36,000.00
70500     SE            STEPHEN MCGEE               19,000.00
                        AGNES COVINGTON             20,000.00
                        RONALD T JACKSON            46,000.00
30200     SW            JOHN HENRY CRANE            19,600.00
                        JANE LOWELL                 22,000.00
10100     SW            PHYLLIS LOCKMEYER           48,000.00
30200     SW            FREDERICH GRAY              48,000.00
10100     SW            WILLIAM AMES                52,000.00
30200     SW            MITCHELL J HOOPS            92,000.00
```

Figure 5-24   Sorting on Multiple Fields

As Figure 5-24 illustrates, the data is first sorted by region. Within each region, it is further sorted by salary. The first sort field (PLANT.REGION) listed in the inquiry statement becomes the primary sort and the second sort field (SAL.YTD) becomes the secondary sort.

The hierarchy of sort action is determined by the position of the sort field in the inquiry statement.

Reversing the Order of the Fields to be Sorted

In the next example, the order of the fields to be sorted has been reversed.

```
DISPLAY PLANT PLANT.ID PLANT.REGION EMP.NAME SAL.YTD
SORT SAL.YTD SORT PLANT.REGION IF SAL.YEAR = 94;
```

Example: "List each employee by salary year to date, plant location, and plant region if the employee worked during 1994. Sort the data by salary and by region within each salary."

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID PLANT.REGION EMP.NAME SAL.YTD SORT SAL.YTD
SORT PLANT.REGION IF SAL.YEAR = 94;;

PLANT.ID  PLANT.REGION  EMP.NAME                        SAL.YTD
20150     MW            WILMA FORD                    15,600.00
70500     SE            STEPHEN MCGEE                 19,000.00
30200     SW            JOHN HENRY CRANE              19,600.00
70500     SE            AGNES COVINGTON               20,000.00
60200     NE            KAREN REDFERN                 22,000.00
30200     SW            JANE LOWELL                   22,000.00
20150     MW            CHARLES SALTER                30,000.00
                        SUSAN WARE                    32,000.00
50300     NW            JONATHAN OAKS                 36,000.00
60200     NE            DAVID YORK                    37,000.00
70500     SE            RONALD T JACKSON              46,000.00
60200     NE            RUSSELL M SIMMONS             48,000.00
10100     SW            PHYLLIS LOCKMEYER             48,000.00
30200     SW            FREDERICH GRAY                48,000.00
20150     MW            PETER ZATKIN                  50,000.00
10100     SW            WILLIAM AMES                  52,000.00
40300     NC            JOAN EVANS                    64,000.00
                        DONALD M KING                 66,000.00
30200     SW            MITCHELL J HOOPS              92,000.00
```

Figure 5-25 Reversing the Order of SORT Commands

Figure 5-25 shows that the first SORT command orders the data by salary. The only actual sorting of PLANT.REGION occurs at the fifth and sixth detail lines and at the twelfth through the fourteenth detail lines. This happens because the salaries are all the same ($22,000 and $48,000, respectively) and are sorted by PLANT.REGION.

The non-printing of PLANT.ID at the eighth detail line occurred because the two employees (Charles Salter and Susan Ware) both work at plant 20150. When sorting is specified, the data is not grouped by levels, as it is without sorting.

Printing of a field value is suppressed if the same value occurs for that field on the preceding line. Suppression occurs on the fields in the order that you specify them in the DISPLAY command, from left to right. Suppression checking is stopped for a line as soon as a field value is different from the one in the preceding line, even if there are more fields to the right of that one that are the same.

The above inquiry statement, with its two SORT commands, emphasizes the fact that you must be familiar with your database in order to be able to use the SORT command effectively.

Sorting Blank Fields

In sorting, blank fields are less than alpha characters and alpha characters are less than digits. This means that if you asked VISION:Inquiry to sort M, (blank), and 5 in ascending order, the following would be the order of display.

```
(blank), M, and 5
```

The next inquiry illustrates part of this point.

Example: "List the name of each employee in Plant 20150 with their educational degree and the subjects studied. Arrange the list in ascending alphabetical order by those subjects."

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME ED.DEGREE SUB.NAME SORT SUB.NAME IF
PLANT.ID = 20150;


PLANT.ID  EMP.NAME                   ED.DEGREE  SUB.NAME
20150     WILMA FORD                 HS
          PETER ZATKIN               BA         ACCOUNTING
          CHARLES SALTER             BA         EDUCATION
          SUSAN WARE                 BS         ENGR
          PETER ZATKIN               MA         FINANCE
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (24,1 USER DB CALLS,ROOTS)
```

Figure 5-26 Sorting Blanks

The SORT command in Figure 5-26 was applied to the field called SUB.NAME. Because Wilma Ford has a high school degree, there is no data in that field; it is blank and is listed first in the output. Accounting, education, engr, and finance were then sorted. Because he has two degrees, Peter Zatkin is listed twice, once for accounting and again for finance.

Sorting in Descending Order

As mentioned previously, you can specify either ascending or descending order with the SORT command.

In this section, SORT DSC is discussed. Although SORT ascending is usually written simply as SORT, in the next few examples SORT ASC is used in the inquiry statements in order to distinguish it from SORT DSC.

```
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD SORT DSC SAL.YTD IF SAL.YEAR = 93;
```

sort this field in descending order

Example: "List each employee, his or her plant identification, and salary if the employee worked during 1993. Arrange the list in descending order by salary."

Note that SAL.YTD is sorted from the highest value ($76,000) to the lowest ($13,400).

```
PAGE:       TRANCODE: II    INQUIRY:  DISPLAY PLANT PLANT.ID EMP.NAME
SAL.YTD SORT DSC SAL.YTD IF SAL.YEAR = 93;;

PLANT.ID  EMP.NAME                        SAL.YTD
30200     MITCHELL J HOOPS              76,000.00
40300     DONALD M KING                 58,000.00
20150     PETER ZATKIN                  44,000.00
70500     RONALD T JACKSON              39,000.00
60200     DAVID YORK                    31,000.00
20150     CHARLES SALTER                24,000.00
30200     JOHN HENRY CRANE              13,400.00
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (74,7 USER DB CALLS,ROOTS)
```

Figure 5-27 SORT DSC Command

Using SORT ASC and SORT DSC

Both SORT ASC and SORT DSC can be used in the same inquiry statement.

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME SAL.YTD IF
    SAL.YTD GE 30000 | SORT DSC PLANT.ID  | SORT ASC SAL.YTD; |
                     |                    |                   |
                     |   sort this field  |  sort this field  |
                     |   in DSC order     |  in ASC order     |
```

Example: "List all the employees whose salaries are equal to or greater than $30,000 per year. Arrange the list in descending order by plant identification and ascending order by salary."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME SAL.YTD IF SAL.YTD
GE 30000 SORT DSC PLANT.ID SORT ASC SAL.YTD;


PLANT.ID  EMP.NO   EMP.NAME                        SAL.YTD
70500      70511   RONALD T JACKSON              32,000.00
           70529   MARTHA WALLINGHAM             33,000.00
           70511   RONALD T JACKSON              39,000.00
                                                 46,000.00
                                                 52,000.00
60200      60205   DAVID YORK                    31,000.00
           60251   MARCIE MORINO                 31,000.00
           60205   DAVID YORK                    37,000.00
                                                 43,000.00
           60209   RUSSELL M SIMMONS             48,000.00
                                                 60,000.00
50300      50322   MADELYN BATES                 32,000.00
           50304   JONATHAN OAKS                 36,000.00
                                                 46,000.00
40300      40304   DONALD M KING                 58,000.00
           40306   JOAN EVANS                    64,000.00
           40304   DONALD M KING                 66,000.00
           40306   JOAN EVANS                    73,200.00
           40304   DONALD M KING                 75,000.00
```

Figure 5-28   Using SORT DSC and SORT ASC in the Same Inquiry

PLANT.ID is the primary sort field. Therefore, because SORT DSC was specified, the plants are listed in descending order, beginning with 70500, followed by 60200, and so forth.

SAL.YTD is the secondary sort field. The salaries are sorted in ascending order, from lowest to highest, within each plant.

Using Multiple SORT DSC Commands

You may use as many SORT DSC commands as you need within a single inquiry statement. The following statement uses two SORT DSC commands.

Example: "List all the employees whose salaries are $30,000 or greater. Include plant location, employee name, employee number, and salary on the list. Arrange the list in descending order by plant location and salary."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME SAL.YTD IF SAL.YTD
GE 30000 SORT DSC PLANT.ID SORT DSC SAL.YTD;


PLANT.ID  EMP.NO   EMP.NAME                         SAL.YTD
70500      70511   RONALD T JACKSON               52,000.00
                                                  46,000.00
                                                  39,000.00
           70529   MARTHA WALLINGHAM              33,000.00
           70511   RONALD T JACKSON               32,000.00
60200      60209   RUSSELL M SIMMONS              60,000.00
                                                  48,000.00
           60205   DAVID YORK                     43,000.00
                                                  37,000.00
                                                  31,000.00
           60251   MARCIE MORINO                  31,000.00
50300      50304   JONATHAN OAKS                  46,000.00
                                                  36,000.00
           50322   MADELYN BATES                  32,000.00
40300      40304   DONALD M KING                  75,000.00
           40306   JOAN EVANS                     73,200.00
           40304   DONALD M KING                  66,000.00
           40306   JOAN EVANS                     64,000.00
           40304   DONALD M KING                  58,000.00
```

Figure 5-29 Using Multiple SORT DSC Commands

In Figure 5-29, both PLANT.ID and SAL.YTD have been sorted in descending order.

As in the previous example, the primary SORT action was applied to the PLANT.ID field. Because SORT DSC was requested, the plants are again listed in descending order, beginning with 70500.

In this example, however, the SORT DSC action was also specified for the secondary sort, which was applied to the SAL.YTD field. The salaries were grouped again according to the plants and then sorted in descending order, beginning with the highest and descending to the lowest.

There may be some restrictions placed on using the SORT command at your installation. Contact your system administrator for more information.

Use of SORT with Subtotals and SUM

The SUM command and the subtotal commands (explained in the next two chapters) are computed at different times in the data selection process. The SUM command accumulates as the data is input, and its control breaks are determined by the actual hierarchical structure. When VISION:Inquiry computes subtotals, the control breaks are determined by the data itself.

The SORT command, as it reorders the data, also reconstructs the hierarchy with UDO (User Defined Output). This new hierarchy is what determines the control breaks and data suppression of the output. Because of this, the SUM can generate unexpected output when the data is reordered through use of the SORT.

As long as the **summary control fields** are used as the SORT control fields, the summaries are correct. However, the use of one field as the SORT control field and another as the subtotal control field can produce unexpected results.

SORT Command Summary

SORT	Use SORT to sequence data in an order different from the one in which it is stored in the database. It must always be used with the DISPLAY command.
ASC, DSC	You can specify SORT with either ascending (ASC) or descending (DSC) order. VISION:Inquiry defaults to SORT ASC unless you specify DSC. ASC is optional.
field name	Use SORT with any field name in the database except a subfield.

Any number of SORT commands can be incorporated into the inquiry statement. A SORT command can also be specified with multiple fields. In this case, the first field after the SORT command will be the primary sort field, the second field will be the secondary sort field, and so on.

The LIMIT Command

The LIMIT command provides you with the ability to obtain a subset or a sampling of the data from an inquiry statement.

Using the LIMIT Command During Testing

If you are unsure of the results of a lengthy inquiry and want to avoid consuming expensive processing time, limit the scope of the inquiry while testing. Add the LIMIT command to an inquiry to restrict the amount of output and see whether the report is suitable. If the report is suitable, you can delete the LIMIT command from the inquiry and run the inquiry again. On the other hand, if the sample report does not meet your needs, you can rework your inquiry, run it again using the LIMIT command, and look at the second sample report.

Know the Structure of Your Database

There is one qualification to using the LIMIT command: you must know the structure of your database in order to correctly read the report you receive. Only by knowing your database are you able to judge whether the LIMIT command output meets your needs. Figure 5-30 illustrates the output of the following statement containing the LIMIT command.

```
DISPLAY PLANT EMP.NAME ED.DEGREE IF ED.DEGREE = 'BA'   LIMIT 3;
```

LIMIT command

Example: “Produce a sample listing of three employees whose degree equals BA. List those employees by name and degree.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE IF ED.DEGREE = 'BA' LIMIT 3;

EMP.NAME                   ED.DEGREE
WILLIAM AMES               BA
PHYLLIS LOCKMEYER          BA
CHARLES SALTER             BA
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (23,2 USER DB CALLS,ROOTS)
```

Figure 5-30 Using the LIMIT Command

The LIMIT command cannot be used alone in an inquiry statement. You can use LIMIT with any command except SORT. In this example, it was incorporated into a DISPLAY command.

Highest Level Segment Illustration

The LIMIT command is applied to the highest level segment referenced in the inquiry. In Figure 5-30, EMP.NAME is the highest segment referenced. (See the illustration in Figure 5-31.)

VSAM non-hierarchical files and DB2 tables are treated as a single segment database; thus, the LIMIT command limits the number of lines of data displayed in the output.

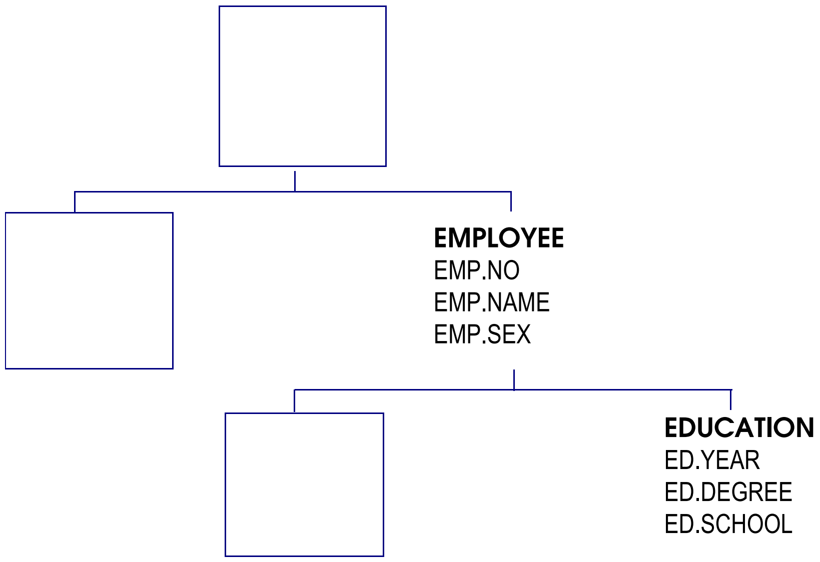


Figure 5-31 Highest Level Segment Illustration

The LIMIT action is applied to EMP.NAME, and the first three names are displayed.

Leftmost Segment Illustration

If the LIMIT command is used in an inquiry that references more than one segment at the same hierarchical level, LIMIT is applied to the leftmost segment of the database. For example:

```
DISPLAY PLANT PROD.CODE EMP.NAME ED.DEGREE LIMIT 6;
```

The database looks like this:

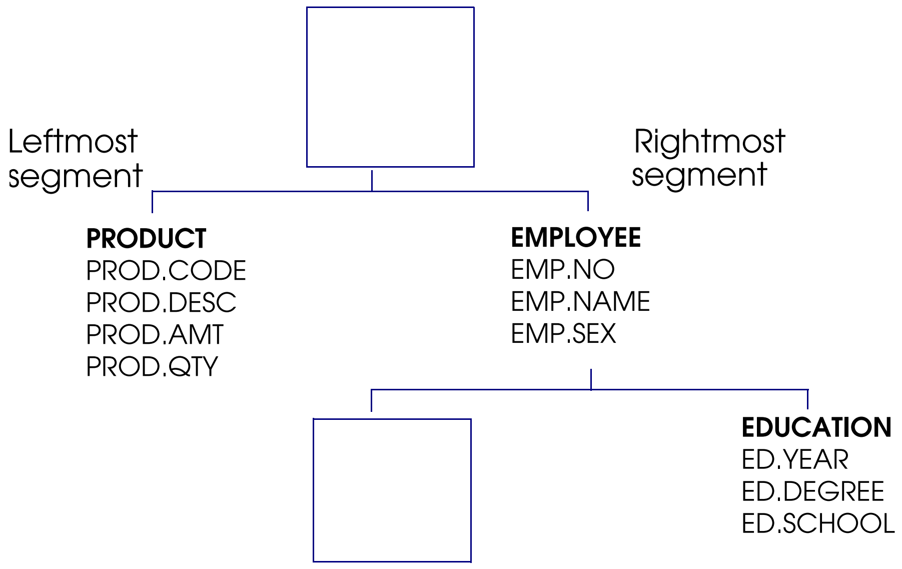


Figure 5-32 Leftmost Segment Illustration

LIMIT Command Applied to Leftmost Segment

PROD.CODE and EMP.NAME are fields in two different segments at the same level of the database. The LIMIT action is applied to PROD.CODE because it is in the leftmost segment accessed.

Example: “Produce a complete listing of the product codes of the toys produced, and the employees and their educational backgrounds. However, to be sure that the inquiry statement is accurate, take a sampling of six product codes before running the complete job.”

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PROD.CODE EMP.NAME ED.DEGREE LIMIT 6;



PROD.CODE  EMP.NAME                   ED.DEGREE
           WILLIAM AMES               BA
           PHYLLIS LOCKMEYER          BA
           MARY ANN THOMAS            HS
PO
RO
SC
SJ
SR
TR         WILMA FORD                 HS
           CHARLES SALTER             BA
           PETER ZATKIN               BA
                                      MA
           SUSAN WARE                 BS
           JOHN HENRY CRANE           HS
           FREDERICH GRAY             BA
                                      MA
           MITCHELL J HOOPS           BS
                                      MA
           JANE LOWELL                HS
```

Figure 5-33 LIMIT Command Applied to the Leftmost Segment

Notice that the LIMIT command was applied to the PROD.CODE only, limiting that output to six codes, PO, RO, SC, SJ, SR, and TR. This was the sample requested in the inquiry statement. The EMP.NAMEs and ED.DEGREEs are displayed because they are treated like lower level segments and are unaffected by the LIMIT command.

LIMIT Command Applied to Highest Level Segment

In the following example (Figure 5-34), LIMIT is applied to the highest level segment (PLANT.ID) to illustrate that the lower level segments are unaffected by the LIMIT action.

```
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD IF SAL.YEAR = 95
LIMIT 3;
```

Example: “List those employees, their plant identification and salaries if they worked during 1995. Run a sampling of three first.”

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD IF SAL.YEAR = 95 LIMIT 3;

PLANT.ID  EMP.NAME                        SAL.YTD
10100     WILLIAM AMES                  64,000.00
          PHYLLIS LOCKMEYER             59,000.00
          MARY ANN THOMAS               15,600.00
20150     WILMA FORD                    18,800.00
          CHARLES SALTER                39,000.00
          PETER ZATKIN                  56,000.00
          SUSAN WARE                    41,000.00
30200     JOHN HENRY CRANE              22,000.00
          FREDERICH GRAY                59,000.00
          MITCHELL J HOOPS              98,000.00
          JANE LOWELL                   26,000.00
          PATRICIA BLAKELY              30,000.00
          SHARON DALEY                  17,000.00

 IXX9121  END OF INQUIRY.                (74,4 USER DB CALLS,ROOTS)
```

Figure 5-34 LIMIT Applied to Highest Level Segment

The LIMIT command was applied to the PLANT.ID field because it is the highest level segment in the inquiry statement. Therefore, three PLANT.IDs were output. In addition, all of the employees who worked in these plants during 1995 were displayed with their salaries because the LIMIT action did not affect the lower level segments.

LIMIT Command Summary

LIMIT	Restricts the number of items displayed from the database.
integer	The number of items displayed depends upon the integer inserted immediately after the command. The integer can be from 1 to the total number of occurrences of the field to be displayed.

LIMIT allows you to get a subset or sampling of data.

Do not use LIMIT as the only command in the inquiry statement; it is always used in conjunction with another command, except SORT.

LIMIT is applied only to the highest referenced segment in the inquiry statement.

If two segments at the same level are displayed, LIMIT is applied to the leftmost segment.

You can use LIMIT with the native SQL syntax facility.

OUTPUT Command

The OUTPUT command permits the output from an inquiry to be sent to another terminal that has been previously defined to VISION:Inquiry.

OUTPUT Command Under IMS

OUTPUT destinations include a system printer (when running in the Batch Message Processing mode), a hard copy slave printer, or another logical terminal (when running in the online mode).

The LTERMs that describe the printers and the sending and receiving terminals must all be in the same directory.

It is this great flexibility in routing inquiries to various destinations that makes the OUTPUT command so important.

The inquiry that is to be sent to another terminal is checked first for syntax by VISION:Inquiry. If any syntax errors are present, the inquiry is returned to the input terminal; once the inquiry statement has been corrected, it is reentered. When no errors are detected in the syntax, the inquiry is switched to the output terminal that has been previously specified. Upon receiving the output, the input terminal receives a message that the output was sent.

Using an OUTPUT Command

This is how to write an OUTPUT command.

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME     OUTPUT    'USER-DEFINED TERMINAL';
```

command | routing destination

Figure 5-35 illustrates the message the input terminal receives informing it that the output was routed to another destination.

```
PAGE:       TRANCODE: II     INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME OUTPUT 'CONV505';


IXX0325  INQUIRY WILL BE SENT TO LTERM/TERM 'CONV505'.
         CICS USERS SHOULD PRESS THE ENTER KEY FOR MORE MESSAGES.
```

Figure 5-35 Message Sent to Input Terminal Regarding Routing of Output

Each terminal has an assigned name. You reference the appropriate output device by specifying its name in the OUTPUT command. In the command, CONV505 was referenced. Therefore, the output from the inquiry was routed to the terminal that previously had been defined as CONV505. Your system will have different names assigned to its terminals. Check with your system administrator to find out what your names are.

When using the OUTPUT command, treat the name of your user-defined terminal as a character constant and enclose it within single quotation marks.

Continuing an Inquiry Sent to Another Terminal

To continue an inquiry that was sent to another terminal, enter CONTINUE OUTPUT 'lterm name' or OUTPUT 'lterm name' and use the PF1 key.

Routing to Another Device

Optionally, you can add an MFS 'modname' to the OUTPUT command. This is necessary when the output device requires a special VISION:Inquiry format. Your system administrator can tell you what MFS modname to use and when to use this parameter. An example of an inquiry using this parameter is as follows:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME OUTPUT 'CONV505' 'INQSPC';
```

The inquiry that is to be routed to a different destination is first checked for syntax by VISION:Inquiry.

If any errors are present, the inquiry is returned to the input terminal. In the following inquiry statement, EMP.NAME has been misspelled.

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NMAE OUTPUT 'CONV505';
```

This is what you see on your terminal screen if you make this syntax error.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NMAE OUTPUT 'CONV505';



IXX0106 DATA NAME 'EMP.NMAE' NOT FOUND IN DIRECTORY.
         THE NAME SHOWN ABOVE IS INTERPRETED TO BE A FIELD
         OR DATABASE NAME, OR A STORED INQUIRY OR FUNCTION
         NAME. HOWEVER, IT CANNOT BE LOCATED IN THE
         DICTIONARY ASSOCIATED WITH YOUR TERMINAL.
         PLEASE CHECK THE SPELLING.
*
*
*
*
*
*
*
*
*
*
IXX0199 INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

Figure 5-36 Syntax Error Returned to Input Terminal

If this syntax error displays at your terminal, correct the spelling of EMP.NAME and reenter the inquiry. Once the inquiry is switched to the previously specified output terminal, you receive the message that the output has been sent to that terminal.

OUTPUT Command Under CICS

The OUTPUT command permits the output from an inquiry to be sent to another terminal that has been previously defined to VISION:Inquiry. These destinations include any other terminal (including printers) defined to CICS that BMS (Basic Mapping Support) understands.

The TERMs that describe the printers and the sending and receiving terminals must all be in the same directory. BMS must also include support for those terminals.

It is this great flexibility in routing inquiries to various destinations that makes the OUTPUT command so important.

The inquiry that is to be sent to another terminal is checked first for syntax by VISION:Inquiry. If any syntax errors are present, the inquiry is returned to the input terminal; once the inquiry statement has been corrected, it is reentered.

When no errors are detected in the syntax, the inquiry is switched to the output terminal that had been specified previously. Upon receiving the output, the input terminal receives a message that the output was sent.

Using an OUTPUT Command

This is how to write an OUTPUT command:

```
DISPLAY SALARIES PLANT_ID EMP_NO EMP_NAME     OUTPUT    'USER-DEFINED TERMINAL';
```

command | routing destination

Figure 5-37 on page 5-47 illustrates the message the input terminal receives informing it that the output will be routed to another destination.

Error Message Screen

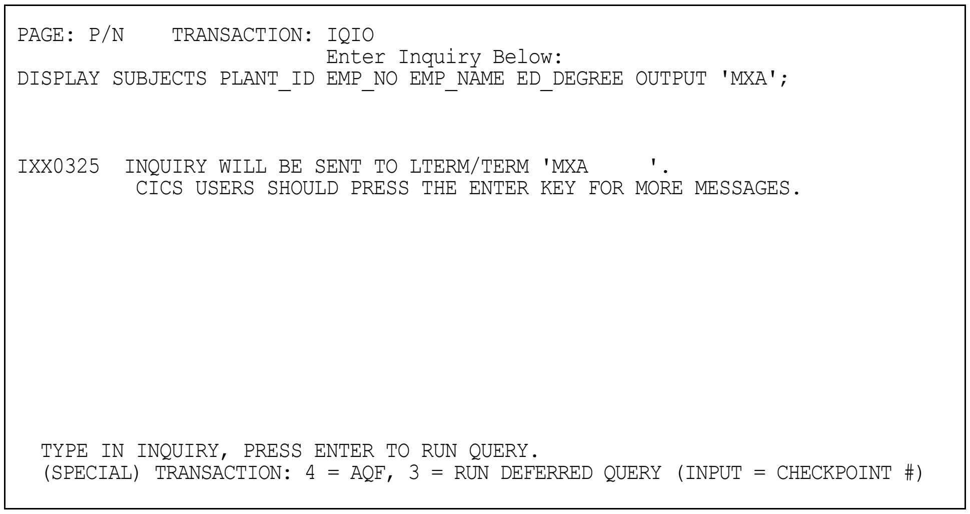
Note: If you do not enter the T/C command and try to process another inquiry, the error message screen may return from the previous output command instead.

To see any error message(s) or the final message:

- Type T/C (or equivalent command) to terminate the current message in the PAGE field and press Enter;

  or

- Press Enter. This automatically issues the T/C command for any messages left.



```
PAGE: P/N    TRANSACTION: IQIO
                          Enter Inquiry Below:
DISPLAY SUBJECTS PLANT_ID EMP_NO EMP_NAME ED_DEGREE OUTPUT 'MXA';



IXX0325  INQUIRY WILL BE SENT TO LTERM/TERM 'MXA     '.
          CICS USERS SHOULD PRESS THE ENTER KEY FOR MORE MESSAGES.











  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
  (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 5-37   Message Sent to Input Terminal Regarding Routing of Output

Each terminal has an assigned name. You reference the appropriate output device by specifying its name in the OUTPUT command.

MXA was referenced in the OUTPUT command. Therefore, the output from the inquiry was routed to the terminal that previously had been defined as MXA. Your system will have different names assigned to its terminals. Check with your system administrator to find out what your names are.

When using the OUTPUT command, treat the name of your user-defined terminal as a character constant and enclose it within single quotation marks.

Continuing an Inquiry Sent to Another Terminal

To continue an inquiry that was sent to another terminal, enter CONTINUE OUTPUT 'term name'; otherwise, the output is returned to your terminal. You may send CONTINUE OUTPUT to any other terminal defined to VISION:Inquiry and CICS.

Routing to Another Device

Optionally, you can add a BMS mapset to the OUTPUT command. This is necessary when the output device requires a special VISION:Inquiry format. Your system administrator can tell you what BMS mapset to use and when to use this parameter.

An example of an inquiry using this parameter is as follows:

```
DISPLAY SALARIES PLANT_ID EMP_NO EMP_NAME OUTPUT 'MXA' 'INQSPC';
```

The output from an inquiry that is to be routed to a different destination is first checked for syntax by VISION:Inquiry. If any errors are present, the inquiry is returned to the input terminal. In the following inquiry statement, EMP_NAME has been misspelled:

```
DISPLAY SALARIES PLANT_ID EMP_NO EMP_NMAE OUTPUT 'MXA';
```

This is what you see on your terminal if you make this syntax error:

```
 PAGE: P/N      TRANSACTION: IQIO
                            Enter Inquiry Below:
 DISPLAY SALARIES PLANT_ID EMP_NO EMP_NMAE OUTPUT 'MXA';



 IXX0106  DATA NAME 'EMP_NMAE' NOT FOUND IN DIRECTORY.
          THE NAME SHOWN ABOVE IS INTERPRETED TO BE A FIELD
          OR DATABASE NAME, OR A STORED INQUIRY OR FUNCTION
          NAME. HOWEVER, IT CANNOT BE LOCATED IN THE
          DICTIONARY ASSOCIATED WITH YOUR TERMINAL.
          PLEASE CHECK THE SPELLING.
*
*
*
*
*
*
*
*
 IXX0199  INQUIRY TERMINATED DUE TO ABOVE ERRORS.
 TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
  (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 5-38   Syntax Error Returned to Input Terminal

If this syntax error is returned to you at your terminal, correct the spelling of EMP_NAME and reenter the inquiry. After it executes, you receive a message that the output has been sent to the previously specified terminal.

OUTPUT Command Summary

OUTPUT	The OUTPUT command routes the report to another destination.
device name	Device name indicates the device to which the output should be sent. The name is limited to eight characters and must be enclosed within quotation marks.
MFS modname or **BMS mapset name**	The MFS modname for IMS or BMS mapset name for CICS is the name of the terminal format to be used for this output device. This parameter is optional and must be enclosed within quotation marks when specified.

When running in the online region, you can use the OUTPUT Command to direct the display to another terminal or hard copy slave printer.

When running in the BMP mode, you can use the OUTPUT command to direct the results of an inquiry to a SYSOUT printer.

Names of output devices are assigned by the system administrator. They should be treated as character constants and enclosed within single quotation marks.

You can also use the OUTPUT command with the native SQL syntax facility.

Chapter

6 Summarizing Data

In this chapter, TOTAL, COUNT, and AVERAGE grand summaries are discussed. Subtotaling is discussed with each of these commands.

This chapter does not apply to the native SQL syntax facility. See the appropriate IBM manual for a description and syntax of the summarizing data (GROUP BY) and column functions (such as SUM, AVG, COUNT) in the SQL SELECT statement.

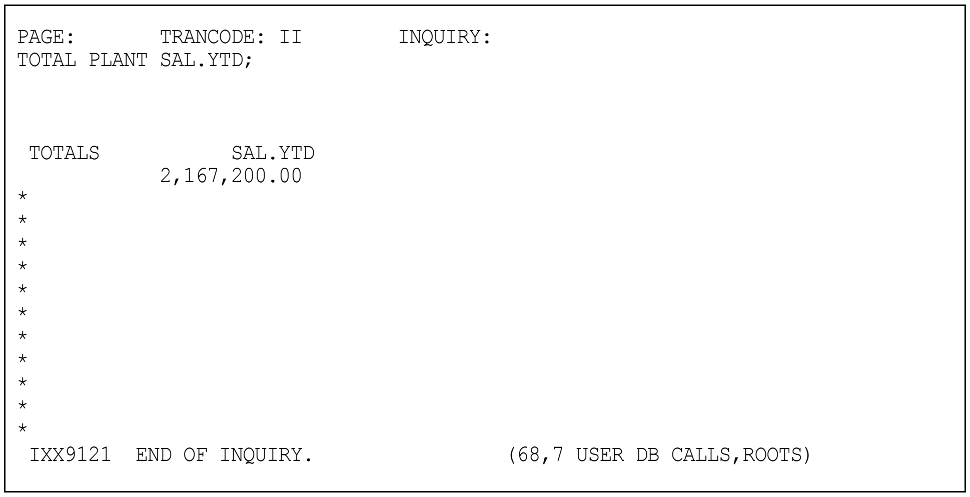
TOTAL Command

The TOTAL command totals the contents of a specified numeric field.

```
TOTAL PLANT SAL.YTD;
```

The TOTAL command accumulates grand totals of the field SAL.YTD from the PLANT database. It tells VISION:Inquiry to add all the values of SAL.YTD together and display the grand total.

Example: "Total the amount of salaries that have been paid."



```
PAGE:       TRANCODE: II        INQUIRY:
TOTAL PLANT SAL.YTD;



 TOTALS           SAL.YTD
            2,167,200.00
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (68,7 USER DB CALLS,ROOTS)
```

Figure 6-1 TOTAL Command

Figure 6-1 shows the grand total of salaries for everyone who is working for the organization. If an employee has been working for four years, those four years of salaries have been included in the total figure on the report.

TOTAL with Conditional Selection

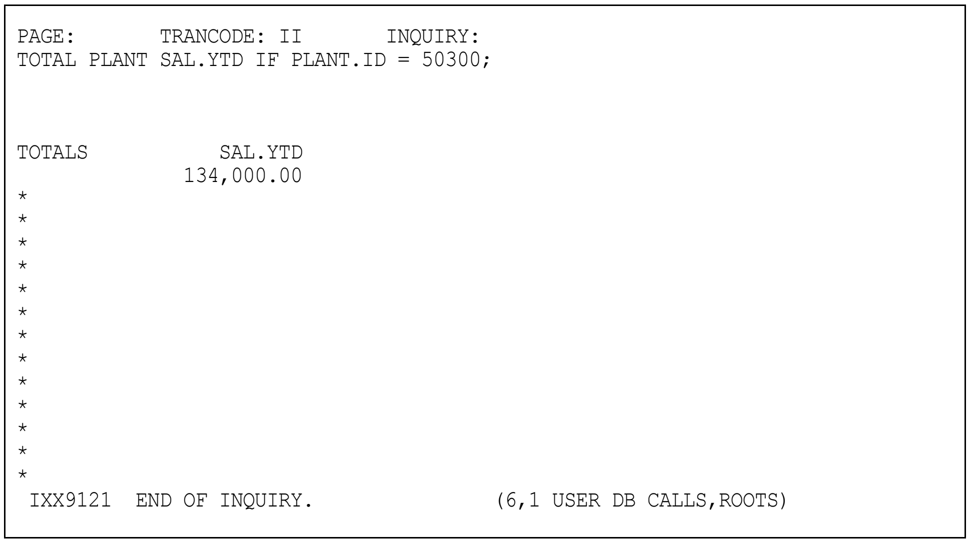
The values that are included in TOTAL can be limited by a conditional selection phrase.

For example, the statement

```
TOTAL PLANT SAL.YTD IF PLANT.ID = 50300;
```

requests that only the salaries for one plant be totaled.

Example: “Total the amount of salaries paid in Plant 50300.”



```
PAGE:       TRANCODE: II       INQUIRY:
TOTAL PLANT SAL.YTD IF PLANT.ID = 50300;



TOTALS           SAL.YTD
              134,000.00
*
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.               (6,1 USER DB CALLS,ROOTS)
```

Figure 6-2 Using Conditional Selection with TOTAL

The conditional phrase, IF PLANT.ID = 50300, limits the salary amount being totaled to a specific plant. If you compare the total salaries in Plant 50300 to those of the whole company, as displayed in Figure 6-1, you begin to understand the relationship of Plant 50300 to the organization as a whole.

Using TOTAL with Numeric Fields Only

Use the TOTAL command with numeric fields only. The following command attempts to TOTAL a non-numeric field:

```
TOTAL PLANT EMP.NAME
```

The field EMP.NAME is alpha rather than numeric. Figure 6-3 shows the error message that you will receive if you enter this command.

Example: “What happens when you try to total a non-numeric field?”

```
PAGE:       TRANCODE: II        INQUIRY:
TOTAL PLANT EMP.NAME;



IXX0410 THE FIELD 'EMP.NAME' CONTAINS INVALID DATA.
        TOTAL OR AVERAGE NOT POSSIBLE
*
*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (2,1 USER DB CALLS,ROOTS)
```

Figure 6-3 Using TOTAL with a Non-Numeric Field

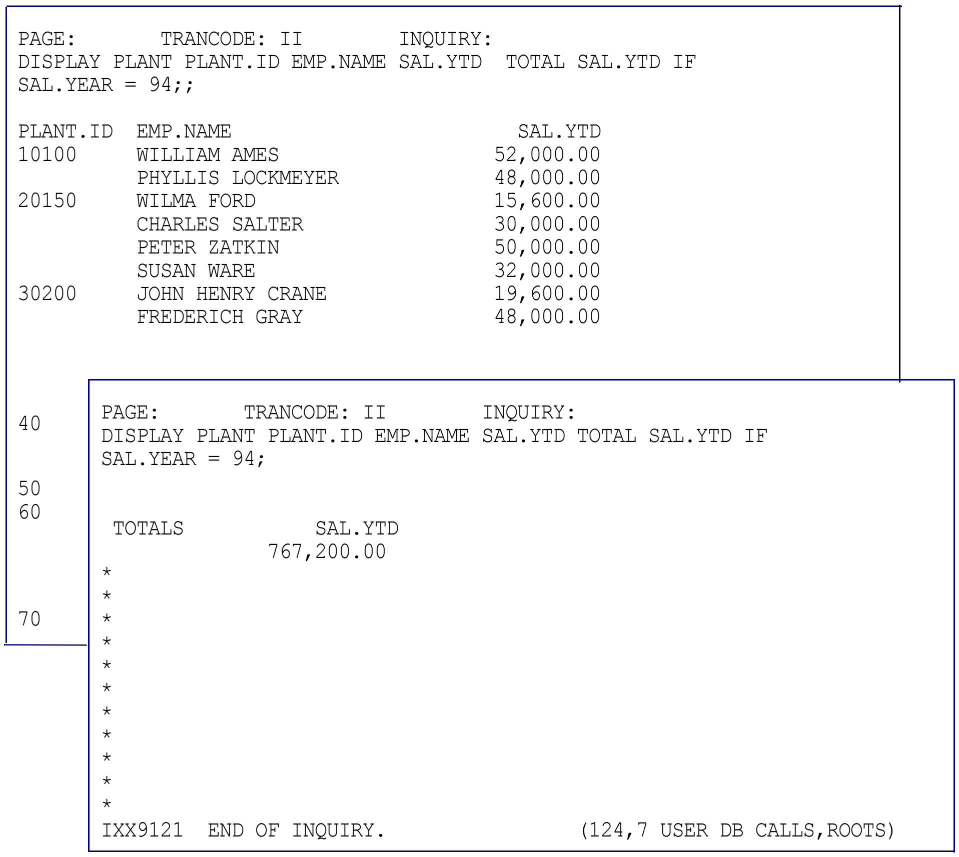
Using TOTAL with Other Commands

You can use TOTAL alone or with other commands, such as DISPLAY.

```
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD TOTAL SAL.YTD IF SAL.YEAR = 94;
```

The report is displayed on two pages. The different values of PLANT.ID, EMP.NAME, and SAL.YTD appear on the first page of the report; the TOTAL appears on the second page. Figure 6-4 shows the output.

Example: “Create a listing of employees, their salaries for 1994, and their plant identifications. Also, total all the salaries that were paid in 1994.”



```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD  TOTAL SAL.YTD IF
SAL.YEAR = 94;;

PLANT.ID  EMP.NAME                        SAL.YTD
10100     WILLIAM AMES                  52,000.00
          PHYLLIS LOCKMEYER             48,000.00
20150     WILMA FORD                    15,600.00
          CHARLES SALTER                30,000.00
          PETER ZATKIN                  50,000.00
          SUSAN WARE                    32,000.00
30200     JOHN HENRY CRANE              19,600.00
          FREDERICH GRAY                48,000.00


40

50
60



70
```

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD TOTAL SAL.YTD IF
SAL.YEAR = 94;


 TOTALS           SAL.YTD
              767,200.00
*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (124,7 USER DB CALLS,ROOTS)
```

Figure 6-4   Using TOTAL with DISPLAY

TOTAL Command Summary

TOTAL	TOTAL accumulates grand totals.
name	The name of the field to be totaled. Use only numeric fields with the TOTAL command.

COUNT Command

The COUNT command calculates a grand count of occurrences of the specified field. You can use any type (alpha, alphanumeric, or numeric) field with the COUNT command. The following statement generates a grand count of female employees only.

```
COUNT PLANT EMP.NAME IF EMP.SEX = 'F';
```

Example: "What is the total number of female employees?"

```
PAGE:       TRANCODE: II       INQUIRY:
COUNT PLANT EMP.NAME IF EMP.SEX = 'F';

 COUNTS      EMP.NAME
                  14

*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (41,7 USER DB CALLS,ROOTS)
```

Figure 6-5 COUNT Command

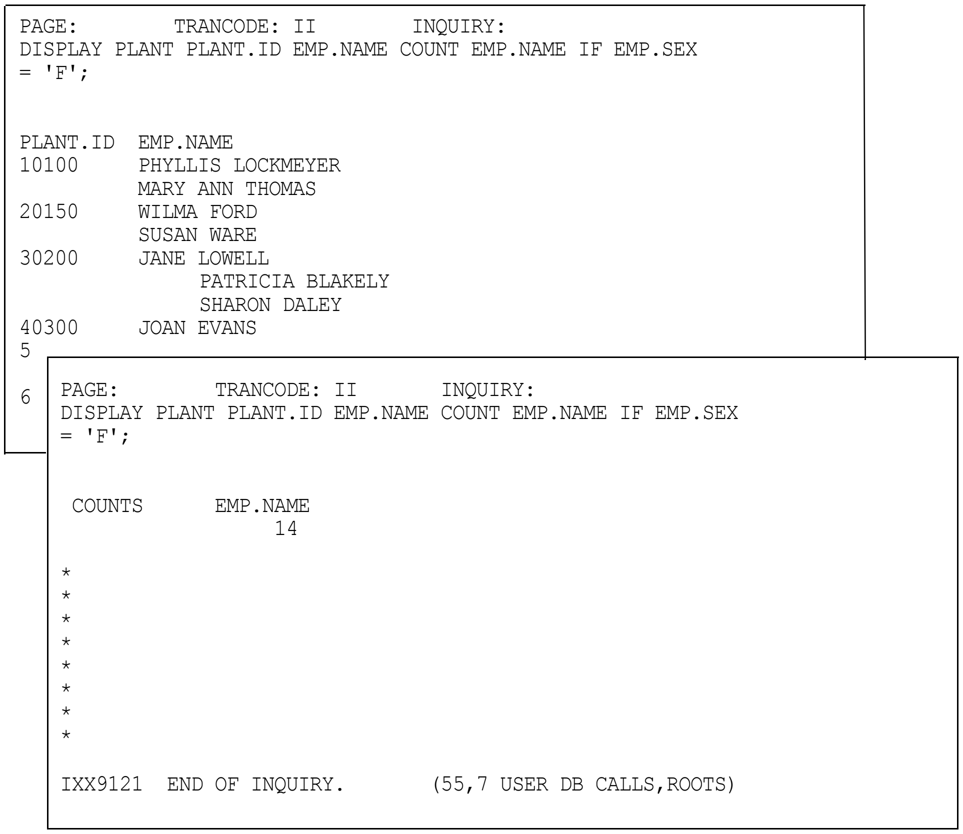
COUNT accumulates a grand count. When you use COUNT alone, your report contains only the grand counted summary.

Using COUNT with Other Commands

As with TOTAL, you can use COUNT with other commands.

```
DISPLAY PLANT PLANT.ID EMP.NAME COUNT EMP.NAME IF EMP.SEX = 'F';
```

Example: "Produce a listing and a count of the women employees."



```
PAGE:        TRANCODE: II        INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME COUNT EMP.NAME IF EMP.SEX
= 'F';


PLANT.ID  EMP.NAME
10100     PHYLLIS LOCKMEYER
          MARY ANN THOMAS
20150     WILMA FORD
          SUSAN WARE
30200     JANE LOWELL
               PATRICIA BLAKELY
               SHARON DALEY
40300     JOAN EVANS
5

6
```

```
PAGE:        TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME COUNT EMP.NAME IF EMP.SEX
= 'F';


 COUNTS      EMP.NAME
                  14

*
*
*
*
*
*
*
*

IXX9121  END OF INQUIRY.       (55,7 USER DB CALLS,ROOTS)
```

Figure 6-6   Using COUNT with DISPLAY

When the COUNT command is combined with the DISPLAY command, the results are obvious. The first part of the report lists the data specified by the DISPLAY command. In this case, it is a list of female employees at each location where they work (see Figure 6-6). The following page of the report contains the grand count of the number of women working throughout the plants.

COUNT Command Summary

COUNT	COUNT calculates a grand count of the occurrences of a specified field.
name	The field to be counted. You can use any field name (alpha as well as numeric) with the COUNT command.

You can use COUNT alone or with the DISPLAY command.

AVERAGE Command

The AVERAGE command calculates a grand average of the contents of the specified field. This is done by adding the contents of all occurrences of the field to be averaged, and dividing by the number of occurrences.

You can use AVERAGE with any numeric field.

```
AVERAGE PLANT SAL.YTD IF EMP.SEX = 'M' AND ED.DEGREE NE 'HS';
```

The AVERAGE command generates a grand average of SAL.YTD for only the male employees who have college degrees.

Example: “What is the average salary of the male employees who have college degrees?”

```
PAGE:       TRANCODE: II        INQUIRY:
AVERAGE PLANT SAL.YTD IF EMP.SEX = 'M' AND ED.DEGREE NE 'HS';



 AVERAGES         SAL.YTD
               51,888.88

*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (92,7 USER DB CALLS,ROOTS)
```

Figure 6-7   AVERAGE Command

The conditional phrase, IF EMP.SEX = ’M’ AND ED.DEGREE NE ‘HS’, was used to limit the salaries being included in the average to only those men who hold college degrees. From the database, those degrees include BA, BS, MA, MS, and PD.

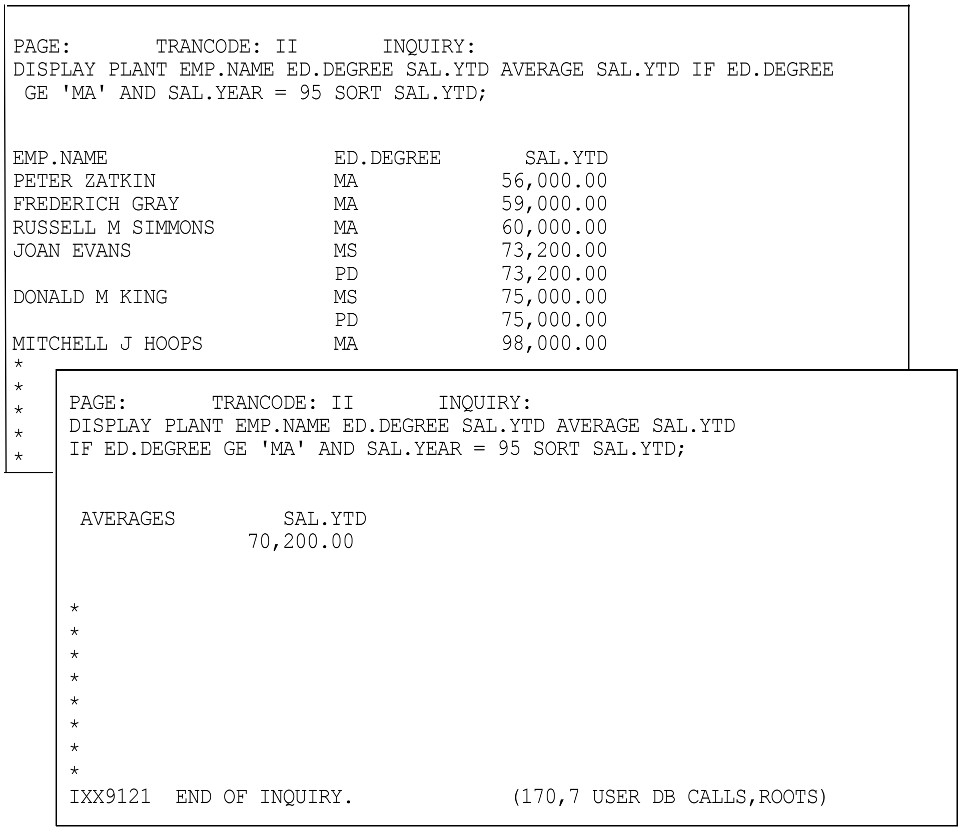
Using AVERAGE with Other Commands

You can use AVERAGE, like TOTAL and COUNT, alone or with other commands.

The next example shows TOTAL used with DISPLAY and SORT.

```
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD AVERAGE SAL.YTD
IF ED.DEGREE GE 'MA' AND SAL.YEAR = 95 SORT SAL.YTD;
```

Example: “List the employees, their educational degrees, and their salaries if they hold a graduate degree; average and sort by 1995 salaries.”



```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD AVERAGE SAL.YTD IF ED.DEGREE
 GE 'MA' AND SAL.YEAR = 95 SORT SAL.YTD;


EMP.NAME                   ED.DEGREE       SAL.YTD
PETER ZATKIN               MA            56,000.00
FREDERICH GRAY             MA            59,000.00
RUSSELL M SIMMONS          MA            60,000.00
JOAN EVANS                 MS            73,200.00
                           PD            73,200.00
DONALD M KING              MS            75,000.00
                           PD            75,000.00
MITCHELL J HOOPS           MA            98,000.00
*
*
*
*
*
```

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.DEGREE SAL.YTD AVERAGE SAL.YTD
IF ED.DEGREE GE 'MA' AND SAL.YEAR = 95 SORT SAL.YTD;


 AVERAGES         SAL.YTD
               70,200.00



*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.            (170,7 USER DB CALLS,ROOTS)
```

Figure 6-8   Using AVERAGE with DISPLAY

In Figure 6-8, the conditional selection phrase, IF ED.DEGREE GE ’MA’ AND SAL.YEAR = 95, was used; therefore, only a few of the employees from the database were selected for the average.

The average of the salaries is shown on a second page of the report.

AVERAGE Command Summary

AVERAGE	AVERAGE calculates a grand average of the contents of the specified field.
name	The field name to be averaged. Use only numeric fields with the AVERAGE command.

You can use AVERAGE alone or with the DISPLAY command.

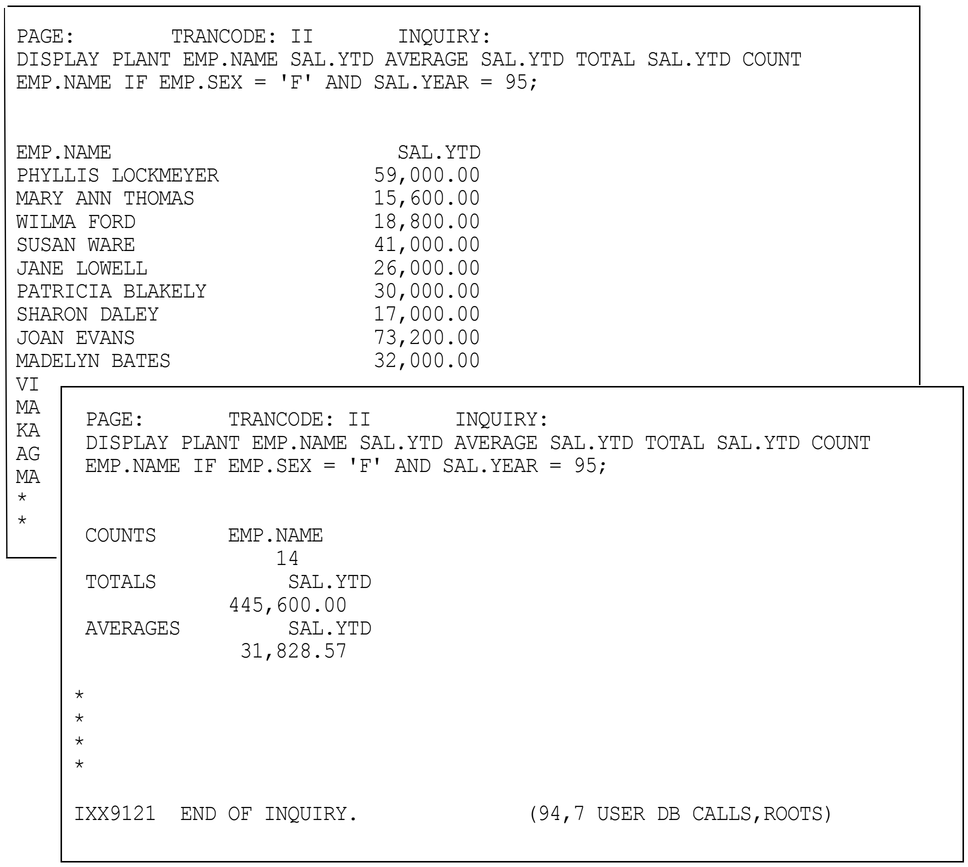
Combining TOTAL, COUNT, and AVERAGE

You can use multiple grand summary commands within a single inquiry statement.

Assume that a report is required listing the names of all the female employees and the salary each earned in 1995. Assume also that a count of all those employees selected is required, along with a grand total and a grand average of their salaries. This report can be produced with the following inquiry.

```
DISPLAY PLANT EMP.NAME SAL.YTD AVERAGE SAL.YTD TOTAL SAL.YTD COUNT
EMP.NAME IF EMP.SEX = 'F' AND SAL.YEAR = 95;
```

Example: “Produce a listing of the women who worked in 1995 and their salaries. Average and total those salaries and then count the women.”



```
PAGE:        TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME SAL.YTD AVERAGE SAL.YTD TOTAL SAL.YTD COUNT
EMP.NAME IF EMP.SEX = 'F' AND SAL.YEAR = 95;


EMP.NAME                        SAL.YTD
PHYLLIS LOCKMEYER             59,000.00
MARY ANN THOMAS               15,600.00
WILMA FORD                    18,800.00
SUSAN WARE                    41,000.00
JANE LOWELL                   26,000.00
PATRICIA BLAKELY              30,000.00
SHARON DALEY                  17,000.00
JOAN EVANS                    73,200.00
MADELYN BATES                 32,000.00
VI
MA
KA
AG
MA
*
*
```

```
 PAGE:       TRANCODE: II       INQUIRY:
 DISPLAY PLANT EMP.NAME SAL.YTD AVERAGE SAL.YTD TOTAL SAL.YTD COUNT
 EMP.NAME IF EMP.SEX = 'F' AND SAL.YEAR = 95;


 COUNTS      EMP.NAME
                 14
 TOTALS           SAL.YTD
             445,600.00
 AVERAGES         SAL.YTD
              31,828.57

*
*
*
*

IXX9121  END OF INQUIRY.              (94,7 USER DB CALLS,ROOTS)
```

Figure 6-9 Using DISPLAY, AVERAGE, TOTAL, and COUNT in a Single Inquiry

The inquiry contains four commands — DISPLAY, AVERAGE, TOTAL, and COUNT. A single conditional selection phrase is used to retrieve those employees who are women and who worked during 1995. This selected data is then processed by each of the specified commands.

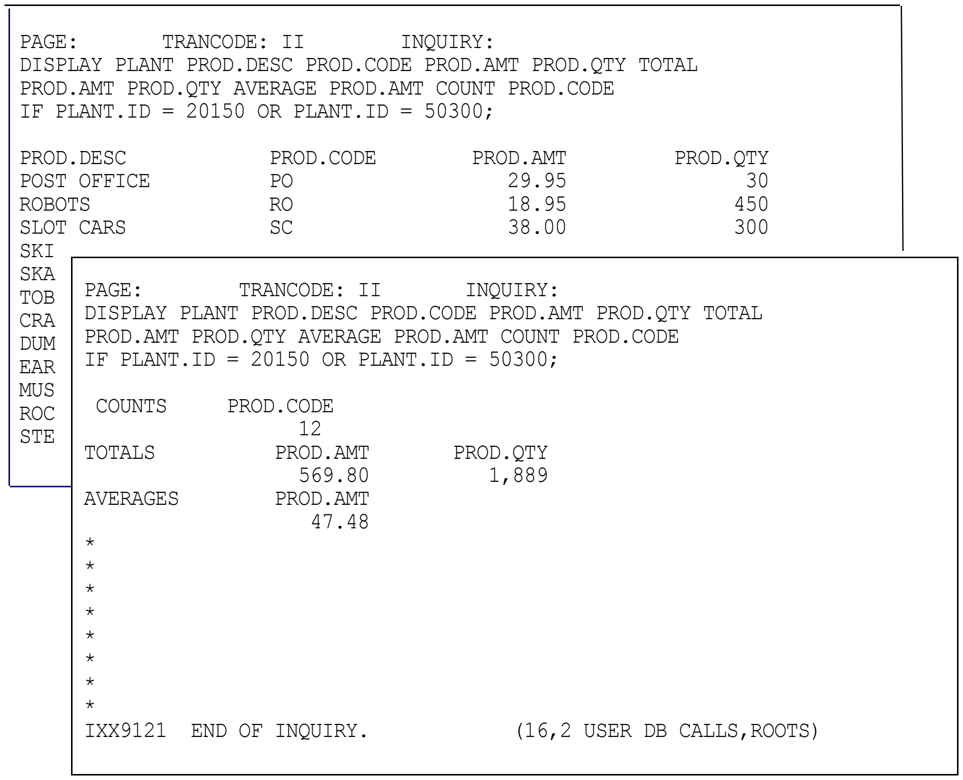
Notice that the order of output is different from the order of commands specified in the inquiry statement. Regardless of the order in which the commands are specified, the order of output is DISPLAY, COUNT, TOTAL, AVERAGE.

Using TOTAL with Two Fields

Notice what happens when a single TOTAL command is used and it acts upon two fields.

```
DISPLAY PLANT PROD.DESC PROD.CODE PROD.AMT PROD.QTY TOTAL
PROD.AMT PROD.QTY AVERAGE PROD.AMT COUNT PROD.CODE
IF PLANT.ID = 20150 OR PLANT.ID = 50300;
```

Example: “For Plants 20150 and 50300, create a report showing product description, product code, product amount, and product quantity. Determine the totals of product amount and product quantity, display the average of product amount, and count the number of products listed.



```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT PROD.DESC PROD.CODE PROD.AMT PROD.QTY TOTAL
PROD.AMT PROD.QTY AVERAGE PROD.AMT COUNT PROD.CODE
IF PLANT.ID = 20150 OR PLANT.ID = 50300;

PROD.DESC            PROD.CODE         PROD.AMT          PROD.QTY
POST OFFICE          PO                   29.95                30
ROBOTS               RO                   18.95               450
SLOT CARS            SC                   38.00               300
SKI
SKA
TOB
CRA
DUM
EAR
MUS
ROC
STE
```

```
PAGE:        TRANCODE: II       INQUIRY:
DISPLAY PLANT PROD.DESC PROD.CODE PROD.AMT PROD.QTY TOTAL
PROD.AMT PROD.QTY AVERAGE PROD.AMT COUNT PROD.CODE
IF PLANT.ID = 20150 OR PLANT.ID = 50300;

 COUNTS     PROD.CODE
                  12
TOTALS          PROD.AMT       PROD.QTY
                  569.80          1,889
AVERAGES        PROD.AMT
                   47.48
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.            (16,2 USER DB CALLS,ROOTS)
```

Figure 6-10 Using a Single TOTAL Command for Two Fields

Notice that the TOTAL command acted upon two separate fields — PROD.AMT and PROD.QTY, giving a grand total for each field.

TOTAL, COUNT, and AVERAGE Command Summary

You can use TOTAL, COUNT, and AVERAGE commands in one inquiry. Each may act upon more than one field.

You can also use more than one COUNT, TOTAL, or AVERAGE command in an inquiry.

Regardless of the placement of the grand summary commands in the inquiry statement, their order of output is COUNT, TOTAL, AVERAGE.

When used with DISPLAY, grand summaries are displayed on a separate page.

When requested for a field in a segment with no occurrences for that record, the output will be as follows.

Command	Output
Count	0
Total	0
Average	?????

Subtotaling - Using a Group Field or Control Break

A **subtotal** is a total of only part of the data. To arrive at a subtotal, the data is broken into groups, such as by plant or sex; then each of the groups is totaled separately. The field that controls the group is called a **group field**, or **control break**. When the group field changes, the subtotal is printed.

The subtotal allows for summarizing one or more fields by groups rather than generating a grand total.

To subtotal data, enclose all of the field names to be acted upon within parentheses. The first of these fields becomes the group field. When the value of this field changes, the subtotal is printed. The remaining field names identify what is to be subtotaled.

In front of the parentheses, add one of the following commands — TOTAL, COUNT, or AVERAGE. The following is an example of a subtotal command:

`PLANT`	`TOTAL`	`(PLANT.ID`	`SAL.YTD);`
database name	command	group field name	field to be totaled

Notice that the control field and the field to be subtotaled are on the same leg of the database. The results are unpredictable if more than one leg of the database is accessed, as with the command AVERAGE (PROD.AMT ED.DEGREE).

Subtotaling with a TOTAL Command

Example: “What is the total amount of salaries paid at each plant?”

```
PAGE:       TRANCODE: II       INQUIRY:
PLANT TOTAL (PLANT.ID SAL.YTD);



 ** PLANT.ID = 10100
TOTALS           SAL.YTD
              238,600.00
** PLANT.ID = 20150
TOTALS           SAL.YTD
              350,400.00
** PLANT.ID = 30200
TOTALS           SAL.YTD
              523,000.00
** PLANT.ID = 40300
TOTALS           SAL.YTD
              336,200.00
** PLANT.ID = 50300
TOTALS           SAL.YTD
              134,000.00
** PLANT.ID = 60200
TOTALS           SAL.YTD
              298,000.00
** PLANT.ID = 70500
```

Figure 6-11 Subtotaling a TOTAL

Note the two asterisks in front of each PLANT.ID. These asterisks identify the output as a subtotal. Also, note that the group field name (PLANT.ID) is printed, identifying the figure beneath it as belonging to one specific plant.

Subtotaling a COUNT in a DISPLAY Command

Any of the subtotal commands can be combined with DISPLAY. This is how to write such an inquiry:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME ED.DEGREE COUNT (EMP.NAME
ED.DEGREE) IF ED.DEGREE NE 'HS' AND PLANT.ID  = 30200;
```

Example: “How many college degrees do each of the employees of Plant 30200 have?”

```
 PAGE:       TRANCODE: II       INQUIRY:
 DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME ED.DEGREE COUNT
 (EMP.NAME ED.DEGREE) IF ED.DEGREE NE 'HS' AND PLANT.ID
 = 30200;

 PLANT.ID  EMP.NO   EMP.NAME                   ED.DEGREE
 30200      30202   FREDERICH GRAY             BA
                                               MA
 ** EMP.NAME = FREDERICH GRAY
 COUNTS     ED.DEGREE
                    2
            30205   MITCHELL J HOOPS           BS
                                               MA
 ** EMP.NAME = MITCHELL J HOOPS
 COUNTS     ED.DEGREE
                    2
            30211   PATRICIA BLAKELY           BS
 ** EMP.NAME = PATRICIA BLAKELY
 COUNTS     ED.DEGREE
                    1
 *
 *
 *
IXX9121  END OF INQUIRY.                (26,1 USER DB CALLS,ROOTS)
```

Figure 6-12 Subtotaling a COUNT

Notice that EMP.NAME, which is the first field in the parentheses, is the group field name and is set off by two asterisks. The count of ED.DEGREE appears under each group field name.

Subtotaling on AVERAGE and Using Parentheses

You can include several fields within the parentheses to be subtotaled. The following is an example:

```
DISPLAY PLANT EMP.NAME ED.SCHOOL AVERAGE (PLANT.ID SAL.YTD
SAL.DED) IF ED.DEGREE = 'MA';
```

Example: “At each plant, what is the average salary and salary deduction for the employees who hold ’MA’ degrees? From what schools did they graduate?”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME ED.SCHOOL AVERAGE (PLANT.ID SAL.YTD
SAL.DED) IF ED.DEGREE = 'MA';
EMP.NAME                   ED.SCHOOL
PETER ZATKIN               TEXAS
** PLANT.ID = 20150
AVERAGES         SAL.YTD        SAL.DED
               50,000.00       8,060.00
FREDERICH GRAY             U OF MASS
MITCHELL J HOOPS           USC
** PLANT.ID = 30200
AVERAGES         SAL.YTD        SAL.DED
               74,600.00      15,155.60
RUSSELL M SIMMONS          WHARTON
** PLANT.ID = 60200
AVERAGES         SAL.YTD        SAL.DED
               54,000.00       8,960.00
*
*
*
 IXX9121  END OF INQUIRY.                (95,7 USER DB CALLS,ROOTS)
```

Figure 6-13 Subtotaling an AVERAGE

In Figure 6-13, PLANT.ID is used as the group field and an AVERAGE is calculated for both SAL.YTD and SAL.DED.

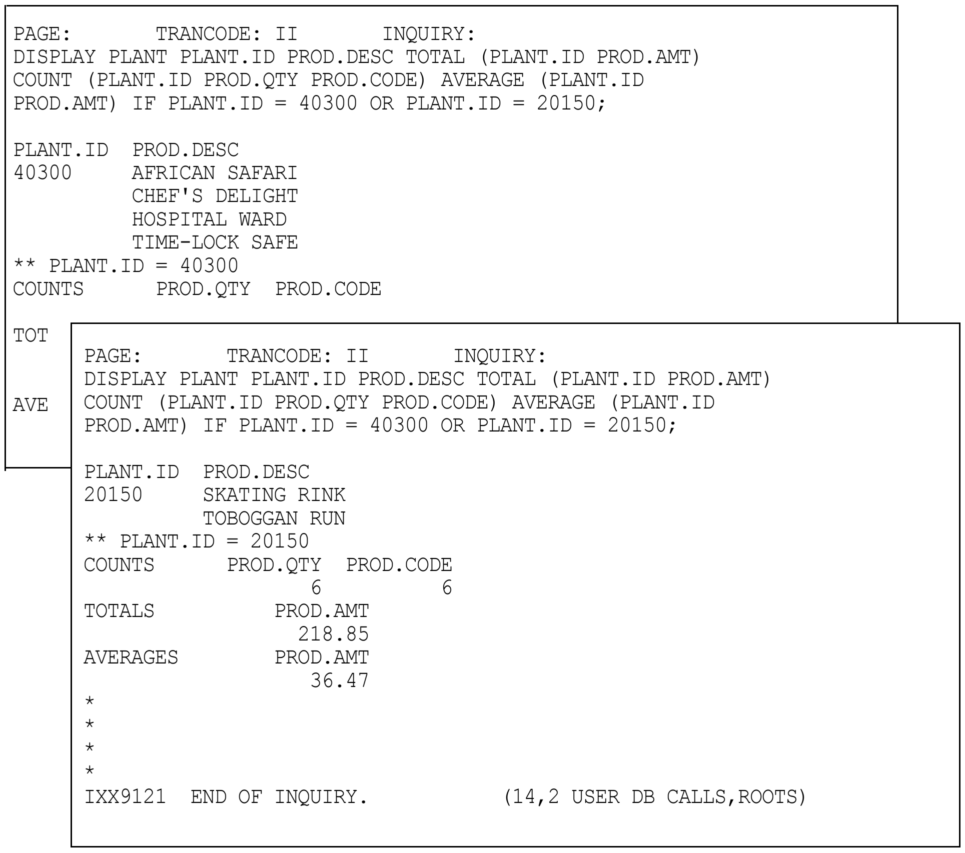
Note that in Plant 30200, both Frederich Gray and Mitchell J Hoops fulfill the requirement as stated in the conditional selection phrase of the inquiry statement. Therefore, their salaries are added together and then averaged; the same thing happens with their salary deductions. The averaged subtotals are output under the two asterisks of the group field name, PLANT.ID.

Combining Subtotal Commands

You can use all three subtotal commands in the same inquiry statement. Consider the following inquiry:

```
DISPLAY PLANT PLANT.ID PROD.DESC TOTAL (PLANT.ID PROD.AMT)
COUNT (PLANT.ID PROD.QTY PROD.CODE) AVERAGE (PLANT.ID
PROD.AMT) IF PLANT.ID = 40300 OR PLANT.ID = 20150;
```

Example: “Give statistical summary information on products in Plants 40300 or 20150.”



```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID PROD.DESC TOTAL (PLANT.ID PROD.AMT)
COUNT (PLANT.ID PROD.QTY PROD.CODE) AVERAGE (PLANT.ID
PROD.AMT) IF PLANT.ID = 40300 OR PLANT.ID = 20150;

PLANT.ID  PROD.DESC
40300     AFRICAN SAFARI
          CHEF'S DELIGHT
          HOSPITAL WARD
          TIME-LOCK SAFE
** PLANT.ID = 40300
COUNTS      PROD.QTY  PROD.CODE

TOT


AVE
```

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID PROD.DESC TOTAL (PLANT.ID PROD.AMT)
COUNT (PLANT.ID PROD.QTY PROD.CODE) AVERAGE (PLANT.ID
PROD.AMT) IF PLANT.ID = 40300 OR PLANT.ID = 20150;

PLANT.ID  PROD.DESC
20150     SKATING RINK
          TOBOGGAN RUN
** PLANT.ID = 20150
COUNTS      PROD.QTY  PROD.CODE
                   6          6
TOTALS          PROD.AMT
                  218.85
AVERAGES        PROD.AMT
                   36.47
*
*
*
*
IXX9121  END OF INQUIRY.           (14,2 USER DB CALLS,ROOTS)
```

Figure 6-14 Taking a Subtotal of TOTAL, COUNT, and AVERAGE in One Inquiry Statement

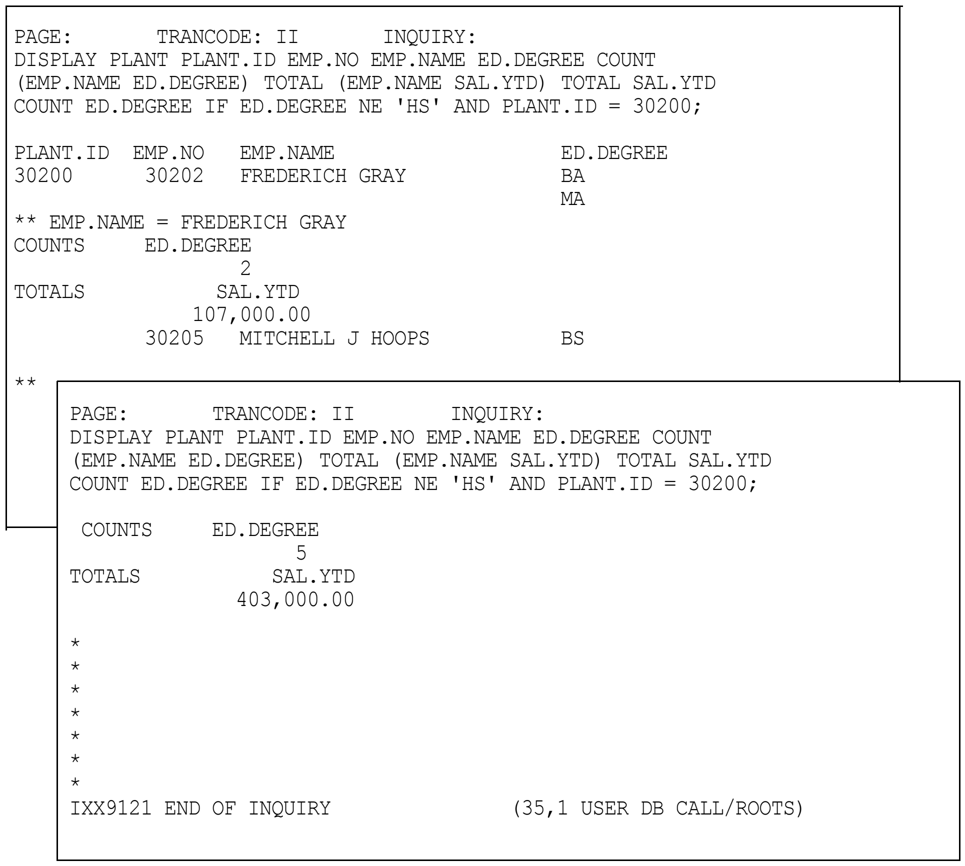
Figure 6-14 first displays the PLANT.ID and PROD.DESC because that is how the inquiry statement read. Then it lists the group field name, which, in this case, is PLANT.ID, identified by two asterisks. Finally, the subtotals are shown. However, the order of the output is different from the order specified in the inquiry statement. In fact, the subtotals are displayed in the same order as that of the grand summaries — COUNT, TOTAL, AVERAGE.

Using Subtotals with Grand Summaries

You can use subtotals with the grand summaries in a single inquiry statement.

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME ED.DEGREE COUNT
(EMP.NAME ED.DEGREE) TOTAL (EMP.NAME SAL.YTD) TOTAL SAL.YTD
COUNT ED.DEGREE IF ED.DEGREE NE 'HS' AND PLANT.ID = 30200;
```

Example: “Statistical information is needed about salaries and degrees for employees with college degrees in Plant 30200.”



```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME ED.DEGREE COUNT
(EMP.NAME ED.DEGREE) TOTAL (EMP.NAME SAL.YTD) TOTAL SAL.YTD
COUNT ED.DEGREE IF ED.DEGREE NE 'HS' AND PLANT.ID = 30200;

PLANT.ID  EMP.NO   EMP.NAME                    ED.DEGREE
30200      30202   FREDERICH GRAY              BA
                                               MA
** EMP.NAME = FREDERICH GRAY
COUNTS     ED.DEGREE
                   2
TOTALS           SAL.YTD
               107,000.00
           30205   MITCHELL J HOOPS            BS

**
```

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME ED.DEGREE COUNT
(EMP.NAME ED.DEGREE) TOTAL (EMP.NAME SAL.YTD) TOTAL SAL.YTD
COUNT ED.DEGREE IF ED.DEGREE NE 'HS' AND PLANT.ID = 30200;

 COUNTS     ED.DEGREE
                   5
TOTALS           SAL.YTD
              403,000.00

*
*
*
*
*
*
*
IXX9121 END OF INQUIRY               (35,1 USER DB CALL/ROOTS)
```

Figure 6-15 Using Subtotals with Grand Summaries

In Figure 6-15, two subtotals and two grand summaries were used in one inquiry statement. The subtotals were COUNT and TOTAL. The grand summaries were TOTAL and COUNT.

The subtotals are displayed exactly as before. Note that the grand summaries are output to a separate page. Also, note that the grand summary commands appear in the first half of the inquiry statement and that the subtotal commands appear in the second half of the statement. This order was exactly reversed in the previous example, as shown in Figure 6-15. Therefore, you can see that you may write your inquiry statements in the order that makes sense to you.

Using SORT with Subtotals

When using the SORT command with subtotals, the SORT field should be the group field. This is the most logical use of the SORT command; if not used in this manner, undesirable results will occur.

Subtotal Command Summary

```
command (group field field(s) to be summarized)
```

To get a subtotal, enclose all of the fields to be subtotaled within the set of parentheses.

In front of the parentheses, add one of the following — TOTAL, COUNT, or AVERAGE. The first of the fields within the parentheses becomes the group field name, which identifies the group being subtotaled on the report.

Subtotal Output

The output for subtotal, subcount, and subaverage is indicated by two asterisks (**) that precede the group field name.

Regardless of how they are specified in the inquiry statements, subtotals are displayed in the following order — COUNT, TOTAL, AVERAGE.

Grand Summaries with Subtotals

When grand summaries are used with subtotals in an inquiry statement, they may appear in any order within the statement. All grand TOTALs, COUNTs, and AVERAGEs are displayed on a separate page, even when subtotals are used.

If grand summaries and subtotals are required, both must be requested in the inquiry statement.

Chapter

7 Assignment Statement and Arithmetic Processing

This chapter does not apply to the native SQL syntax facility. See the appropriate IBM manual for a description and syntax of the expressions and calculations in the SQL SELECT statement.

Assignment Statement

The assignment statement sets up a temporary field and assigns it a value. The value being assigned may be the result of an arithmetic calculation on the contents of another field or another field of the database. The temporary field exists only in the inquiry in which the assignment statement appears and cannot be accessed by other inquiries.

`%`	`SALARY`	`=`	`SAL.YTD`
indicates a temporary field	name of temporary field	assignment operator	value being assigned

The name of the temporary field is selected by the user and can have up to 32 characters. The temporary name (without the % sign) prints out as the column heading. The value to the right of the equal sign is assigned to the temporary field and is stored under that name.

In the example shown above, the contents of the database field named SAL.YTD are being assigned to a new temporary field named SALARY. At any time during the execution of the inquiry of which it is a part, SALARY contains the current value of SAL.YTD.

Temporary fields take on the characteristics of the field name being assigned to them. Because SAL.YTD is a numeric field, SALARY is numeric also. Because they are both numeric, they appear right-justified in the column.

Do not use any of the names that have already been defined in your directory (such as a field name or a database name) as the name of a temporary field.

Example: "Prepare a report illustrating the four types of assignment values."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT SAL.YTD %SALARY = SAL.YTD %RATE = 300
%ITEM = '1995 BONUS' %BONUS = SAL.YTD + 300
IF SAL.YEAR = 95;

    SAL.YTD         SALARY    RATE  ITEM                      BONUS
   64,000.00      64,000.00   300   1995 BONUS             64,300.00
   59,000.00      59,000.00   300   1995 BONUS             59,300.00
   15,600.00      15,600.00   300   1995 BONUS             15,900.00
   18,800.00      18,800.00   300   1995 BONUS             19,100.00
   39,000.00      39,000.00   300   1995 BONUS             39,300.00
   56,000.00      56,000.00   300   1995 BONUS             56,300.00
   41,000.00      41,000.00   300   1995 BONUS             41,300.00
   22,000.00      22,000.00   300   1995 BONUS             22,300.00
   59,000.00      59,000.00   300   1995 BONUS             59,300.00
   98,000.00      98,000.00   300   1995 BONUS             98,300.00
   26,000.00      26,000.00   300   1995 BONUS             26,300.00
   30,000.00      30,000.00   300   1995 BONUS             30,300.00
   17,000.00      17,000.00   300   1995 BONUS             17,300.00
   75,000.00      75,000.00   300   1995 BONUS             75,300.00
   73,200.00      73,200.00   300   1995 BONUS             73,500.00
   46,000.00      46,000.00   300   1995 BONUS             46,300.00
   32,000.00      32,000.00   300   1995 BONUS             32,300.00
   20,000.00      20,000.00   300   1995 BONUS             20,300.00
```

Figure 7-1 Assignment Statement

Figure 7-1 shows an inquiry containing four assignment statements, each assigning a different kind of value to a temporary field. (Each line in the output manipulates the SAL.YTD amount for one employee.)

```
%RATE = 300
```

This statement assigns a numeric constant to a temporary field named RATE. Because it is a constant, the value of RATE is the same each time it is displayed. Constants appear left-justified in the field.

```
%ITEM = '1995 BONUS'
```

This statement assigns a character constant to a temporary field named ITEM. Because ITEM is a constant, its value is the same each time it is displayed, and it appears left-justified in the field. Note that a character constant must always be enclosed within quotation marks.

```
%BONUS = SAL.YTD + 300
```

This statement assigns the results of an arithmetic calculation to a temporary field named BONUS. A numeric constant is being added to the contents of a database field. Because SAL.YTD is numeric, the result is numeric and appears right-justified in the column. This type of usage is explained in more detail later in this chapter.

Assignment Statement Summary

%name	% indicates a temporary field. User-selected field name can have up to 32 characters. If more than one temporary field is assigned in an inquiry, each name must be unique. Do not use any names that have already been defined in your directory as the name of a temporary field.
=	Assignment operator assigns value on the right to the name on the left.
value	Value on the right of the assignment operator can be the result of an arithmetic calculation, the contents of a database field, or a numeric or alphanumeric constant. ■ A numeric constant may have up to 14 digits and two decimal places. ■ A decimal value must have a digit to the left of the decimal point, such as '0.10.' ■ An alphanumeric constant may have up to 256 characters.

Arithmetic Calculations

Arithmetic calculations can be performed in the following ways:

- Arithmetic calculations can be performed with the summary commands (TOTAL, AVERAGE, and COUNT) which were explained in Chapter 6, "Summarizing Data". These generate automatic summaries of the data displayed in a report. The steps involved in calculating these summaries are not under the control of the user.
- You can also design your own arithmetic calculations for execution on data in the database. These types of calculations can be performed in one of two ways: with the assignment or the SUM statements (which are discussed in this chapter) or with stored functions (which are discussed in Chapter 9, "Using Directory Commands - Storing Inquiries and Functions").

Arithmetic Expression

An **arithmetic expression** is one that contains arithmetic operators and the values that are to be manipulated. The values may be constants (which do not change), or data field names (which represent values that may change). At least one field from the database must be included in an expression.

Arithmetic Operators

The arithmetic operators used in VISION:Inquiry are:

Symbol	Meaning
*	Multiplication
/	Division
+	Addition
-	Subtraction

The "+" and "-" symbols are also used as leading signs for numeric constants.

Hierarchy of Operation

The operations within an arithmetic expression are executed according to a specific hierarchy, basically from left to right. However, multiplication and division are performed before addition and subtraction.

Any operation enclosed in parentheses is calculated first. If sets of parentheses are nested, the innermost set is cleared first. Parentheses can also be used at any time for clarification.

Number of Decimal Places

The constants used in an arithmetic calculation may have two or four decimal places depending on the arithmetic operations being performed.

- Addition and subtraction are limited to two decimal places, while multiplication and division may have up to four decimal places.

  Note: If you attempt to divide by zero (0), VISION:Inquiry divides by 1 and does not display a message.

- The output field has two decimal places in addition and subtraction, and four decimal places in multiplication and division.

The temporary field receiving the results of an arithmetic calculation is a numeric field and its contents are right-justified in the column.

Arithmetic Processing with the Assignment Statement

The results of an arithmetic calculation can be assigned to a temporary field by an assignment statement and displayed, or can be used as the basis for selecting only certain records for processing. The terms in the arithmetic calculation are specified by the user.

`%BONUS`	`=`	`SAL.YTD * 0.175`
temporary field		arithmetic expression

In this example, the contents of SAL.YTD are multiplied by 0.175, and the result is assigned to the temporary field named BONUS.

Arithmetic Calculation in an Assignment Statement

Figure 7-2 shows the same assignment statement used in an inquiry. The value of BONUS is calculated and displayed for each record in the database that meets the specified condition (SAL.YEAR = 95).

Example: "Calculate a bonus of 17.5% for each employee for 1995."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME SAL.YTD SAL.DED
 %BONUS = SAL.YTD * 0.175  IF SAL.YEAR = 95;

EMP.NAME                       SAL.YTD     SAL.DED             BONUS
WILLIAM AMES                 64,000.00    9,000.00       11,200.0000
PHYLLIS LOCKMEYER            59,000.00    8,200.00       10,325.0000
MARY ANN THOMAS              15,600.00    1,370.00        2,730.0000
WILMA FORD                   18,800.00    1,980.00        3,290.0000
CHARLES SALTER               39,000.00    6,000.00        6,825.0000
PETER ZATKIN                 56,000.00    9,580.00        9,800.0000
SUSAN WARE                   41,000.00    7,000.00        7,175.0000
JOHN HENRY CRANE             22,000.00    2,600.00        3,850.0000
FREDERICH GRAY               59,000.00   13,600.00       10,325.0000
MITCHELL J HOOPS             98,000.00   19,998.00       17,150.0000
JANE LOWELL                  26,000.00    5,800.00        4,550.0000
PATRICIA BLAKELY             30,000.00    4,120.00        5,250.0000
SHARON DALEY                 17,000.00    1,560.00        2,975.0000
DONALD M KING                75,000.00   17,400.00       13,125.0000
JOAN EVANS                   73,200.00   15,600.00       12,810.0000
JONATHAN OAKS                46,000.00   10,040.00        8,050.0000
MADELYN BATES                32,000.00    5,780.00        5,600.0000
VICKY WARD                   20,000.00    2,400.00        3,500.0000
DAVID YORK                   43,000.00    6,800.00        7,525.0000
```

Figure 7-2   Arithmetic Calculation in an Assignment Statement

Since multiplication is performed, BONUS has four decimal places and is right-justified.

Displaying Results of an Arithmetic Calculation

Example: “Calculate the same bonus as in Figure 7-2, but display the results without a column heading.”

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT EMP.NAME SAL.YTD SAL.DED
SAL.YTD * 0.175  IF SAL.YEAR = 95;

EMP.NAME                       SAL.YTD     SAL.DED                 .
WILLIAM AMES                 64,000.00    9,000.00       11,200.0000
PHYLLIS LOCKMEYER            59,000.00    8,200.00       10,325.0000
MARY ANN THOMAS              15,600.00    1,370.00        2,730.0000
WILMA FORD                   18,800.00    1,980.00        3,290.0000
CHARLES SALTER               39,000.00    6,000.00        6,825.0000
PETER ZATKIN                 56,000.00    9,580.00        9,800.0000
SUSAN WARE                   41,000.00    7,000.00        7,175.0000
JOHN HENRY CRANE             22,000.00    2,600.00        3,850.0000
FREDERICH GRAY               59,000.00   13,600.00       10,325.0000
MITCHELL J HOOPS             98,000.00   19,998.00       17,150.0000
JANE LOWELL                  26,000.00    5,800.00        4,550.0000
PATRICIA BLAKELY             30,000.00    4,120.00        5,250.0000
SHARON DALEY                 17,000.00    1,560.00        2,975.0000
DONALD M KING                75,000.00   17,400.00       13,125.0000
JOAN EVANS                   73,200.00   15,600.00       12,810.0000
JONATHAN OAKS                46,000.00   10,040.00        8,050.0000
MADELYN BATES                32,000.00    5,780.00        5,600.0000
VICKY WARD                   20,000.00    2,400.00        3,500.0000
DAVID YORK                   43,000.00    6,800.00        7,525.0000
```

Figure 7-3 Display Results of Arithmetic Calculation

The inquiry in Figure 7-3 is similar to the one in Figure 7-2, except that the results of the arithmetic calculation are not assigned to a temporary field before being displayed. From the output you can see that the values are the same, but a decimal point replaces the column heading.

Complex Arithmetic Calculation and a Temporary Field in a Conditional Phrase

Example: "If net salaries were raised by 20%, which employees would still be receiving less than or equal to $30,000 for the year 1995?"

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME SAL.YTD SAL.DED
 %NEW.SAL = (SAL.YTD - SAL.DED) * 1.20
IF %NEW.SAL LE 30000 AND SAL.YEAR = 95;

EMP.NAME                        SAL.YTD     SAL.DED          NEW.SAL
MARY ANN THOMAS               15,600.00    1,370.00       17,076.0000
WILMA FORD                    18,800.00    1,980.00       20,184.0000
JOHN HENRY CRANE              22,000.00    2,600.00       23,280.0000
JANE LOWELL                   26,000.00    5,800.00       24,240.0000
SHARON DALEY                  17,000.00    1,560.00       18,528.0000
VICKY WARD                    20,000.00    2,400.00       21,120.0000
KAREN REDFERN                 26,000.00    3,960.00       26,448.0000
STEPHEN MCGEE                 23,000.00    2,560.00       24,528.0000
AGNES COVINGTON               23,000.00    2,960.00       24,048.0000
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (85,7 USER DB CALLS,ROOTS)
```

Figure 7-4 More Complex Arithmetic Calculation and a Temporary Field in a Conditional Phrase

Figure 7-4 illustrates a more complex arithmetic calculation and a conditional phrase that tests the value of a temporary field.

$$\texttt{\%NEW.SAL =} \quad \underbrace{\underbrace{\texttt{(SAL.YTD - SAL.DED)}}_{\text{1st}} \quad \texttt{* 1.20}}_{\text{2nd}}$$

This expression calculates a 20% increase of the salary field. According to the hierarchy of operations, the first step is to clear the parentheses and calculate the net salary. Then this value is multiplied by 1.20 to determine the amount of the new salary.

$$\texttt{\%NEW.SAL =} \quad \underbrace{\texttt{SAL.YTD} \quad \texttt{-} \quad \underbrace{\texttt{SAL.DED} \quad \texttt{* 1.20}}_{\text{1st}}}_{\text{2nd}}$$

Had the parentheses not been included, the order of operations would be different and a different answer would have resulted. In this case, SAL.DED would be multiplied by 1.20 first and then subtracted from SAL.YTD. Mary Ann Thomas would have a new salary of $13,956 rather than $17,076.

Arithmetic Calculation in a Conditional Phrase

```
IF %NEW.SAL LE 30000 AND SAL.YEAR = 95
```

This conditional phrase indicates that only those records that meet two conditions should be displayed — the value of NEW.SAL must be less than or equal to $30,000 and the salary year must be 1995.

Example: “A 10% bonus is being considered. Which employees would receive more than $4,400 for the year 1995?”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME SAL.YTD SAL.DED
 %NEG.BAL = SAL.DED - SAL.YTD
IF SAL.YTD * 0.10 GT 4400 AND SAL.YEAR = 95;

EMP.NAME                      SAL.YTD     SAL.DED            NEG.BAL
WILLIAM AMES                64,000.00    9,000.00          55,000.00-
PHYLLIS LOCKMEYER           59,000.00    8,200.00          50,800.00-
PETER ZATKIN                56,000.00    9,580.00          46,420.00-
FREDERICH GRAY              59,000.00   13,600.00          45,400.00-
MITCHELL J HOOPS            98,000.00   19,998.00          78,002.00-
DONALD M KING               75,000.00   17,400.00          57,600.00-
JOAN EVANS                  73,200.00   15,600.00          57,600.00-
JONATHAN OAKS               46,000.00   10,040.00          35,960.00-
RUSSELL M SIMMONS           60,000.00   10,140.00          49,860.00-
RONALD T JACKSON            52,000.00    6,800.00          45,200.00-
*
*
*
*
 IXX9121  END OF INQUIRY.                (98,7 USER DB CALLS,ROOTS)
```

Figure 7-5   Arithmetic Calculation in a Conditional Phrase

Figure 7-5 illustrates an arithmetic calculation in selective processing and shows how a negative value is displayed.

```
IF SAL.YTD * 0.10 GT 4400 AND SAL.YEAR = 95
```

The product of SAL.YTD * 0.10 is calculated and tested against 4400. The line is displayed only if it is larger than 4400 and the year is 1995. You do not see the results of the calculation.

If you needed to view the results, you would perform the calculation in an assignment and place the name of the temporary field in the conditional phrase (see Figure 7-4).

Using the Results of a Calculation in Another Calculation

```
%NEG.BAL = SAL.DED - SAL.YTD
```

This calculation yields a negative result which appears in the field entitled NEG.BAL. The negative sign (-) appears to the right of the number.

Example: “Calculate and display the yearly and monthly amounts paid out in bonuses for each employee for the year 1995. The bonuses are calculated at 14.55% of the employee's SAL.YTD.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT EMP.NAME %BONUS = SAL.YTD * 0.1455
 %MONTH.BONUS = %BONUS / 12 IF
SAL.YEAR = 95;

EMP.NAME                                 BONUS           MONTH.BONUS
WILLIAM AMES                         9,312.0000             776.0000
PHYLLIS LOCKMEYER                    8,584.5000             715.3750
MARY ANN THOMAS                      2,269.8000             189.1500
WILMA FORD                           2,735.4000             227.9500
CHARLES SALTER                       5,674.5000             472.8750
PETER ZATKIN                         8,148.0000             679.0000
SUSAN WARE                           5,965.5000             497.1250
JOHN HENRY CRANE                     3,201.0000             266.7500
FREDERICH GRAY                       8,584.5000             715.3750
MITCHELL J HOOPS                    14,259.0000           1,188.2500
JANE LOWELL                          3,783.0000             315.2500
PATRICIA BLAKELY                     4,365.0000             363.7500
SHARON DALEY                         2,473.5000             206.1250
DONALD M KING                       10,912.5000             909.3750
JOAN EVANS                          10,650.6000             887.5500
JONATHAN OAKS                        6,693.0000             557.7500
MADELYN BATES                        4,656.0000             388.0000
VICKY WARD                           2,910.0000             242.5000
DAVID YORK                           6,256.5000             521.3750
```

Figure 7-6 Using a Temporary Field (Containing the Results of an Arithmetic Calculation) as a Term in Another Arithmetic Calculation

Figure 7-6 illustrates a calculation that uses a temporary field as one of its terms. First,

```
%BONUS = SAL.YTD * 0.1455
```

calculates the yearly bonus for each employee. Then,

```
%MONTH.BONUS = %BONUS / 12
```

accesses the current value of BONUS and divides it by 12 to calculate the monthly bonus.

The first calculation in the series (%BONUS) must access at least one numeric field from the database. This assigns numeric characteristics to the field.

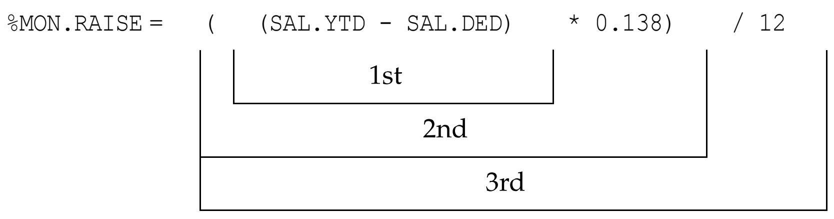
Arithmetic Calculation, Conditional Selection, and Display of Fields

Example: “A 13.8% raise in net salary is being considered. Management needs to see the present SAL.YTD and planned monthly raise for each employee in Plant 30200 for the year 1995.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD
 %MON.RAISE = ((SAL.YTD - SAL.DED) * 0.138) / 12
IF PLANT.ID = 30200 AND SAL.YEAR = 95;

PLANT.ID  EMP.NAME                      SAL.YTD           MON.RAISE
30200     JOHN HENRY CRANE            22,000.00            223.1000
          FREDERICH GRAY              59,000.00            522.1000
          MITCHELL J HOOPS            98,000.00            897.0230
          JANE LOWELL                 26,000.00            232.3000
          PATRICIA BLAKELY            30,000.00            297.6200
          SHARON DALEY                17,000.00            177.5600
*
*
*
*
*
*
IXX9121  END OF INQUIRY.              (32,1 USER DB CALLS,ROOTS)
```

Figure 7-7 Complex Arithmetic Calculation, Conditional Selection, and Display of Fields of the Database

The arithmetic calculation uses nested parentheses to control the order in which the steps that calculate the net raise are executed.


```
%MON.RAISE =   (   (SAL.YTD - SAL.DED)    * 0.138)    / 12
```

Following the hierarchy of operations, the expression in the inner parentheses is evaluated first — SAL.DED is subtracted from SAL.YTD to calculate the present net salary. The expression now looks like this:

```
%MON.RAISE = (NET.SALARY * 0.138) / 12
```

Again, the parentheses are cleared first and the net salary is multiplied by 0.138 to determine the amount of the yearly raise. The expression now looks like this:

```
%MON.RAISE = YEARLY.RAISE / 12
```

The last calculation is performed — the yearly raise is divided by 12 to calculate the monthly raise (MON.RAISE). The conditional phrase in the inquiry limits the output to the employees in plant 30200.

Arithmetic Processing Summary

Operators

Following are the standard arithmetic operators.

Symbol	Meaning
*	Multiplication
/	Division
+	Addition
-	Subtraction

Hierarchy of Operations

- Expressions are evaluated from left to right.
- Expressions in parentheses are evaluated first; if parentheses are nested, the inner set is evaluated first.
- Arithmetic operators are evaluated in the following order:
  - Multiplication and division
  - Addition and subtraction

Arithmetic expressions are composed of arithmetic operators, fields from the database, temporary fields, and numeric constants.

Constants used in arithmetic expressions are limited to two decimal places if used with addition or subtraction operators; they may have four decimal places if used with multiplication or division operators.

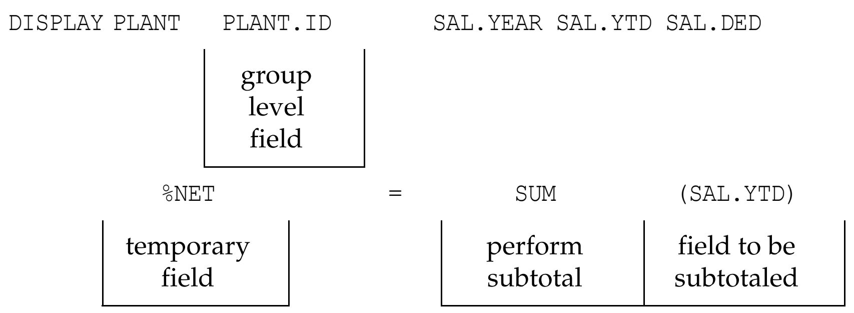
The results of an arithmetic calculation can be displayed, used in a conditional phrase, or assigned to a temporary field.

A temporary field being used in an arithmetic calculation must be a numeric field.

SUM Command

Note: The SUM command does not apply to VSAM non-hierarchical files or DB2 tables.

The SUM command performs subtotals and arithmetic calculations on subgroups of data from the database.



```
DISPLAY PLANT   PLANT.ID       SAL.YEAR SAL.YTD SAL.DED
           %NET             =        SUM        (SAL.YTD)
```

This statement indicates that the values of SAL.YTD should be totaled for each plant and displayed in a column labeled NET. The field to be summarized is next to the SUM command and is enclosed in parentheses. The group field is the next higher segment referenced in the DISPLAY list.

VISION:Inquiry scans the list and determines the database segment from which each field came. If one is from a higher level segment, it becomes the group field.

SUM Command (Example 1)

In this example, SAL.YTD, the field to be subtotaled, is from the SALARY segment of the database. PLANT.ID in the DISPLAY list is the only field in the list from a higher level segment (PLANT); it becomes the group field.

The output from this inquiry is shown in Figure 7-8.

Example: “Total SAL.YTD for each plant.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YEAR SAL.YTD SAL.DED
 %NET = SUM(SAL.YTD);


PLANT.ID  SAL.YEAR        SAL.YTD     SAL.DED                     NET
10100           94      52,000.00    7,400.00
                95      64,000.00    9,000.00
                94      48,000.00    6,400.00
                95      59,000.00    8,200.00
                95      15,600.00    1,370.00              238,600.00
20150           94      15,600.00    1,260.00
                95      18,800.00    1,980.00
                93      24,000.00    2,020.00
                94      30,000.00    3,140.00
                95      39,000.00    6,000.00
                93      44,000.00    6,800.00
                94      50,000.00    7,800.00
                95      56,000.00    9,580.00
                94      32,000.00    5,200.00
                95      41,000.00    7,000.00              350,400.00
30200           93      13,400.00    1,380.00
                94      19,600.00    1,960.00
                95      22,000.00    2,600.00
                94      48,000.00    9,580.00
```

Figure 7-8 SUM Command (Example 1)

Note that, in this figure, the function of the command is similar to subtotaling with the TOTAL command, except for the output. The values of NET are displayed in a column under the name of the temporary field.

SUM Command (Example 2)

Example: "Total SAL.YTD for each employee."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO SAL.YEAR SAL.YTD SAL.DED %NET =
SUM (SAL.YTD);

PLANT.ID  EMP.NO   SAL.YEAR        SAL.YTD     SAL.DED            NET
10100      10103         94      52,000.00    7,400.00
                         95      64,000.00    9,000.00     116,000.00
           10104         94      48,000.00    6,400.00
                         95      59,000.00    8,200.00     107,000.00
           10105         95      15,600.00    1,370.00      15,600.00
20150      21116         94      15,600.00    1,260.00
                         95      18,800.00    1,980.00      34,400.00
           21124         93      24,000.00    2,020.00
                         94      30,000.00    3,140.00
                         95      39,000.00    6,000.00      93,000.00
           21137         93      44,000.00    6,800.00
                         94      50,000.00    7,800.00
                         95      56,000.00    9,580.00     150,000.00
           21164         94      32,000.00    5,200.00
                         95      41,000.00    7,000.00      73,000.00
30200      30201         93      13,400.00    1,380.00
                         94      19,600.00    1,960.00
                         95      22,000.00    2,600.00      55,000.00
           30202         94      48,000.00    9,580.00
```

Figure 7-9 SUM Command (Example 2)

This example is the same as Figure 7-8, except that EMP.NO has been added to the display list. EMP.NO, from the EMPLOYEE segment, is now the next higher level segment referenced in the inquiry and becomes the group field.

SUM and TOTAL in One Inquiry

Example: "Display SAL.YTD showing subtotals for both plants and employees."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NO SAL.YEAR SAL.YTD SAL.DED %NET = SUM
(SAL.YTD) TOTAL (PLANT.ID SAL.YTD);;

PLANT.ID  EMP.NO   SAL.YEAR        SAL.YTD     SAL.DED            NET
10100      10103         94      52,000.00    7,400.00
                         95      64,000.00    9,000.00     116,000.00
           10104         94      48,000.00    6,400.00
                         95      59,000.00    8,200.00     107,000.00
           10105         95      15,600.00    1,370.00      15,600.00
** PLANT.ID = 10100
TOTALS            SAL.YTD
              238,600.00
20150      21116         94      15,600.00    1,260.00
                         95      18,800.00    1,980.00      34,400.00
           21124         93      24,000.00    2,020.00
                         94      30,000.00    3,140.00
                         95      39,000.00    6,000.00      93,000.00
           21137         93      44,000.00    6,800.00
                         94      50,000.00    7,800.00
                         95      56,000.00    9,580.00     150,000.00
           21164         94      32,000.00    5,200.00
                         95      41,000.00    7,000.00      73,000.00
** PLANT.ID = 20150
```

Figure 7-10 SUM and TOTAL Commands in One Inquiry

In the figure above, a SUM command is used to total SAL.YTD for each employee. (EMP.NO references the next higher level segment, EMPLOYEE.) A TOTAL command is used to generate the subtotals for each plant.

Grand Totals using SUM

When using both commands in one inquiry, the output is more pleasing if TOTAL generates the higher level totals.

Example: "Only the SAL.YEAR, SAL.YTD, and SAL.DED fields need to be seen."

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT SAL.YEAR SAL.YTD SAL.DED %NET = SUM(SAL.YTD);;

SAL.YEAR        SAL.YTD     SAL.DED                   NET
      94      37,000.00    4,800.00
      95      43,000.00    6,800.00
      94      48,000.00    7,780.00
      95      60,000.00   10,140.00
      95      31,000.00    3,560.00
      94      22,000.00    2,400.00
      95      26,000.00    3,960.00
      92      32,000.00    2,008.00
      93      39,000.00    3,400.00
      94      46,000.00    5,600.00
      95      52,000.00    6,800.00
      94      19,000.00    1,700.00
      95      23,000.00    2,560.00
      94      20,000.00    1,960.00
      95      23,000.00    2,960.00
      95      33,000.00    3,980.00          2,167,200.00
*
*
 IXX9121  END OF INQUIRY.                (68,7 USER DB CALLS,ROOTS)
```

Figure 7-11 Grand Totals Using the SUM Command

In Figure 7-11, no higher level segments are referenced. All the fields in the DISPLAY list are from the same segment as the field being subtotaled. This arrangement generates a grand total of all the values of SAL.YTD, printed out under the column heading NET. The result is similar to the TOTAL SAL.YTD statement, except for the appearance of the output.

Arithmetic Processing with SUM

You can also use the SUM command without a temporary field name.

```
SUM(SAL.YTD)
```

This statement lists the values of SAL.YTD in a column with a period in place of a column heading.

More than one field may be summed in an inquiry, but each must have its own SUM command. Both are allocated at the same group level.

```
%SALARY = SUM(SAL.YTD) %DEDUCT = SUM(SAL.DED)
```

Example: “Display the 1995 net salary for each employee in each plant.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YTD SAL.DED %EMP.NET = SAL.YTD - SAL.DED
%NET = SUM(SAL.YTD - SAL.DED) IF SAL.YEAR = 95;

PLANT.ID       SAL.YTD    SAL.DED               EMP.NET           NET
10100       64,000.00   9,000.00              55,000.00
            59,000.00   8,200.00              50,800.00
            15,600.00   1,370.00              14,230.00     120,030.00
20150       18,800.00   1,980.00              16,820.00
            39,000.00   6,000.00              33,000.00
            56,000.00   9,580.00              46,420.00
            41,000.00   7,000.00              34,000.00     130,240.00
30200       22,000.00   2,600.00              19,400.00
            59,000.00  13,600.00              45,400.00
            98,000.00  19,998.00              78,002.00
            26,000.00   5,800.00              20,200.00
            30,000.00   4,120.00              25,880.00
            17,000.00   1,560.00              15,440.00     204,322.00
40300       75,000.00  17,400.00              57,600.00
            73,200.00  15,600.00              57,600.00     115,200.00
50300       46,000.00  10,040.00              35,960.00
            32,000.00   5,780.00              26,220.00
            20,000.00   2,400.00              17,600.00      79,780.00
60200       43,000.00   6,800.00              36,200.00
```

Figure 7-12   Arithmetic Processing with SUM Command

There are several ways to perform the arithmetic calculations needed for this report.

Method 1: Figure 7-12 uses an assignment statement to perform the calculation for each employee and the SUM command to calculate the net amount for each plant.

The following statement calculates and displays the net salary for each employee:

```
%EMP.NET = SAL.YTD - SAL.DED
```

The following statement shows the SUM command being used to calculate the net salary for each plant:

```
%NET = SUM(SAL.YTD - SAL.DED)
```

The arithmetic terms that are controlled by a SUM command are enclosed in parentheses. The group level of the SUM command (PLANT in this example) is the next higher level segment referenced in the DISPLAY list.

To execute the SUM command, VISION:Inquiry first calculates the net salary for the first employee and saves this total. It then calculates the net salary for the second employee and adds it to the first total. Then the calculation is repeated for the third employee and added to the accumulating total.

At this point, all the employees in the first plant have been processed and the net total is displayed. This process is then repeated for the second plant. The arithmetic expression is calculated once for each employee; but only the final, cumulative total is displayed.

Method 2: The following statement is an alternative way of composing the SUM statement to produce the same results.

```
%NET = SUM(SAL.YTD) - SUM(SAL.DED)
```

With this method, all the values of SAL.YTD for each plant are added to form one total and all the values of SAL.DED are added to form a second total. When all the employees in a plant have been processed, the second total is subtracted from the first total to calculate the net amount for each plant.

Be careful with this arrangement, however. Due to the laws of arithmetic, the results would be different if the connecting arithmetic operator specified multiplication or division, instead of subtraction.

Creating an Expression

When you create the expression, select the method that reflects what you want to accomplish. If you want the expression to be executed for each record, use one SUM command. If you want one or more columns to be processed separately and their final totals manipulated, use separate SUM commands.

Using SUM on Fields from Different Segments

You can use the SUM command in arithmetic operations on fields from different segments, provided they lie along a single hierarchical path.

For example, SUM can be used to add up all occurrences of a field in a dependent segment, and then add that sum to a field in the parent. However, in coding the arithmetic statement, all dependent segment fields must be to the left of the parent field. For example, if the parent field is on the left of the SUM statement, the calculation is computed only once per root segment. When the dependent segment field is to the left of the parent field, the calculation is computed once for each parent field.

Note: Using two or more SUM statements on different segments on different paths or trying to compute a SUM on an arithmetic expression involving fields from different paths produces undesirable results.

Example: “Display the amount of commissions paid out by each plant in 1995. Commissions are figured as 10% of salary, less 9% of deductions.”

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YTD SAL.DED   %COMMISS = SUM(SAL.YTD *
0.10 - SAL.DED * 0.09) IF SAL.YEAR = 95;

PLANT.ID       SAL.YTD     SAL.DED                COMISS
10100        64,000.00    9,000.00
             59,000.00    8,200.00
             15,600.00    1,370.00           12,188.7000
20150        18,800.00    1,980.00
             39,000.00    6,000.00
             56,000.00    9,580.00
             41,000.00    7,000.00           13,269.6000
30200        22,000.00    2,600.00
             59,000.00   13,600.00
             98,000.00   19,998.00
             26,000.00    5,800.00
             30,000.00    4,120.00
             17,000.00    1,560.00           20,908.9800
40300        75,000.00   17,400.00
             73,200.00   15,600.00           11,850.0000
50300        46,000.00   10,040.00
             32,000.00    5,780.00
             20,000.00    2,400.00            8,160.2000
60200        43,000.00    6,800.00
```

Figure 7-13 SUM Command with an Arithmetic Expression

Figure 7-13 illustrates a SUM command containing an arithmetic expression with several terms. All the terms enclosed within the parentheses are executed for each employee in a plant, but the final amount is displayed only once per plant.

Use of Parentheses

The placement of parentheses in the SUM statement is important. It changes the way the arithmetic expression is executed, and sometimes changes the final value of the expression.

Example: Placement of Parentheses in SUM Statement with no Differences in Results

The next three statements are equivalent and all produce the same output.

```
%COMMISS = SUM(SAL.YTD * 0.10 - SAL.DED * 0.09)
```

This is the same statement as in Figure 7-13. The expression is executed once for each employee.

```
%COMMISS = SUM((SAL.YTD * 0.10) - (SAL.DED * 0.09))
```

This is the same statement as above, except that two sets of inner parentheses have been added for clarity.

```
%COMMISS = SUM(SAL.YTD * 0.10) - SUM(SAL.DED * 0.09)
```

In this statement, each column is added separately, and then the totals are subtracted. Be careful with this arrangement, however. Due to the laws of arithmetic, the results are different if the connecting operator specifies multiplication or division instead of subtraction.

Example: Placement of Parentheses with a Difference in Results

Now, look at another example explaining placement of parentheses. Suppose COMMISN was calculated as 10% of SAL.YTD, less 100. If you use the statement:

```
%COMMISN = SUM(SAL.YTD * 0.10 - 100)
```

100 is subtracted each time the expression is executed (once for each employee).

On the other hand, in the statement:

```
%COMM = SUM(SAL.YTD * 0.10) - 100
```

the expression "- 100" is outside the parentheses, and, therefore, not under the control of the SUM command. 100 is subtracted only once, as the last step in the calculation.

Figure 7-14 shows the difference in output between these two statements.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YTD %COMMISN = SUM(SAL.YTD * 0.10 - 100)
%COMM = SUM(SAL.YTD * 0.10) - 100 IF SAL.YEAR = 95;;

PLANT.ID        SAL.YTD                 COMMISN                COMM
10100         64,000.00
              59,000.00
              15,600.00             13,560.0000         13,760.0000
20150         18,800.00
              39,000.00
              56,000.00
              41,000.00             15,080.0000         15,380.0000
30200         22,000.00
              59,000.00
              98,000.00
              26,000.00
              30,000.00
              17,000.00             24,600.0000         25,100.0000
40300         75,000.00
              73,200.00             14,620.0000         14,720.0000
50300         46,000.00
              32,000.00
              20,000.00              9,500.0000          9,700.0000
60200         43,000.00
```

Figure 7-14   Placement of Parentheses with the SUM Command (Example 1)

Figure 7-15 shows a similar parenthetical arrangement using the data from Figure 7-14.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YTD %TEMP = SUM(SAL.YTD * 0.10) %COMMISS =
SUM(SAL.YTD * 0.10) - (SAL.DED * 0.09) IF SAL.YEAR = 95;;

PLANT.ID        SAL.YTD                    TEMP             COMMISS
10100         64,000.00
              59,000.00
              15,600.00             13,860.0000         13,736.7000
20150         18,800.00
              39,000.00
              56,000.00
              41,000.00             15,480.0000         14,850.0000
30200         22,000.00
              59,000.00
              98,000.00
              26,000.00
              30,000.00
              17,000.00             25,200.0000         25,059.6000
40300         75,000.00
              73,200.00             14,820.0000         13,416.0000
50300         46,000.00
              32,000.00
              20,000.00              9,800.0000          9,584.0000
60200         43,000.00
```

Figure 7-15   Placement of Parentheses with the SUM Command (Example 2)

When “- (SAL.DED * 0.09)” is placed outside the parentheses of the SUM command, it is no longer under its control. Only the last occurrence of this expression (1370 * 0.09 = 123.3) is subtracted from the accumulated total of SAL.YTD * 0.10. In the output, TEMP is the accumulated total, and COMMISS is the accumulated total less 123.3.

Since the parentheses around SAL.DED * 0.09 are only for clarity, the statement:

```
%COMMISS = SUM(SAL.YTD * 0.10) - SAL.DED * 0.09
```

is equivalent and produces the same results.

Showing Only Group Level Totals

If you need to see only the PLANT totals, you can rewrite the DISPLAY statement and produce the output in Figure 7-16.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID %TEMP = SUM(SAL.YTD * 0.10) %COMMISS =
SUM(SAL.YTD * 0.10) - (SAL.DED * 0.09) IF SAL.YEAR = 95;;

PLANT.ID                 TEMP               COMMISS
10100             13,860.0000           13,736.7000
20150             15,480.0000           14,850.0000
30200             25,200.0000           25,059.6000
40300             14,820.0000           13,416.0000
50300              9,800.0000            9,584.0000
60200             16,000.0000           15,643.6000
70500             13,100.0000           12,741.8000
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (126,7 USER DB CALLS,ROOTS)
```

Figure 7-16   Showing Only Group Level Totals

Using SUM with Temporary Fields

You can use a temporary field that holds the results of an arithmetic calculation in an arithmetic expression in a SUM statement. In this case, a grand total is generated and the subtotal level indicated in the DISPLAY list is ignored. Compare Figure 7-16 with Figure 7-17.

```
PAGE:       TRANCODE: II    INQUIRY:
DISPLAY PLANT PLANT.ID SAL.YTD %TEMP = SAL.DED * 0.09
%COMMISS = SUM(SAL.YTD * 0.10 - %TEMP) IF SAL.YEAR = 95;;

PLANT.ID       SAL.YTD                 TEMP              COMMISS
50300        46,000.00             903.6000
             32,000.00             520.2000
             20,000.00             216.0000
60200        43,000.00             612.0000
             60,000.00             912.6000
             31,000.00             320.4000
             26,000.00             356.4000
70500        52,000.00             612.0000
             23,000.00             230.4000
             23,000.00             266.4000
             33,000.00             358.2000          91,809.0800
*
*
*
*
 IXX9121  END OF INQUIRY.                (126,7 USER DB CALLS,ROOTS)
```

Figure 7-17	SUM Command with Temporary Fields

Do not use a temporary field assigned by a SUM statement in another SUM statement. If it is, the column heading appears, but zeroes are listed as its value.

Do not use temporary fields assigned by SUM statements in conditional phrases.

SUM Command Summary

%temporary field name =	Receives the results of a calculation. Its use is optional. If it is not present, a period is displayed instead of a column heading.
SUM	The SUM command performs subtotals and arithmetic calculations on subgroups. **Note:** SUM is not supported for VSAM non-hierarchical files or DB2 tables.
(arithmetic expression)	The expression enclosed in parentheses to its right should be subtotaled. The subtotal group is indicated by a field name in the display list that references a segment level higher than the ones referenced within the parentheses. ■ If no higher level segment is referenced, a grand total is generated. ■ If only higher level segments are referenced in the display list, only the subtotals will be displayed. An arithmetic expression follows the rules for arithmetic processing. It must be enclosed in parentheses. At least one term from the database must be referenced either in this expression or by the expression assigned to a temporary field name.

If a SUM statement uses the results of another arithmetic expression through a temporary field name, the subtotal level is ignored and grand totals are generated.

A SUM statement may not use the results of another SUM statement.

Chapter

8

Accessing Multiple Databases and Files

This chapter does not apply to the native SQL syntax facility. See the appropriate IBM manual for a description and syntax of the joining data from more than one DB2 table in the SQL SELECT statement.

FIND Command

You can use the FIND command to access fields from more than one database in the same inquiry and to access the same database more than one time. It is used in conjunction with a DISPLAY statement.

Two separate statements are written when FIND is used. The first statement incorporates the FIND command and gives VISION:Inquiry information regarding the first database. The second is a DISPLAY statement that gives information on the second database, indicates what to do with the information from both databases, and tells VISION:Inquiry how to relate the databases together.

Not all databases can be linked together. There must be some relationship and matching fields between them. For example, payrolls are often prepared by using two or more databases. Usually, there is a master database containing general information relating to each employee in the company (such as name, address, number of exemptions, kind and amount of deductions, and so on). A second database might contain information from each plant or department telling the number of regular hours worked by each employee, number of overtime hours worked, and information on commissions and bonuses, and so on.

There has to be a way to identify which information from the two databases belongs together. This is usually done by having matching fields in both databases; that is, the employees in the master database might be identified by ID number, and the hours worked in the other database would also be identified by ID number. When VISION:Inquiry relates the two databases together, it matches ID numbers and merges related information from the two databases. The ID number is called the match field. In an inventory database, the part number might be used as the match field; in an employee database, the social security number might be used as the match field.

The PLANT and SKILL databases were designed with matching fields so that they can be used together. The PLANT database is organized by plant. It contains two sets of information — one relating to products produced at each plant, and the other relating to employees at each plant and their salary and educational histories.

The SKILL database is organized by job classification (called skills). It tells which plants have employees in each job classification, and gives the ID number of the employees holding that classification.

Each employee appears once in the SKILL database under the appropriate job classification. The PLANT.ID appears many times under all classifications in which its employees fit.

The PLANT and SKILL databases have two fields in common — PLANT.ID and EMP.NO. VISION:Inquiry can match on either of these fields to merge the related information. In these databases, the names of the matching fields are the same. This is not required (the names may be different); it depends on how they are assigned in the databases. The contents, however, must match in order to link the databases.

Using FIND and DISPLAY to Access Two Databases

The following example is an inquiry containing a FIND command. It shows how to transfer values from one database to another, and how to test for matching fields.

Example: "Produce a listing of the job classifications and job codes in the toy company. Show the plants that have employees in each classification and their geographical region (PLANT.REGION)."

```
PAGE:       TRANCODE: II       INQUIRY:
FIND SKILL %CODE = SKILL.CODE %SKILL = SKILL.NAME
%PLANT = PLANT.ID;
DISPLAY PLANT %CODE %SKILL PLANT.ID PLANT.REGION
IF PLANT.ID = %PLANT;;
CODE   SKILL                 PLANT.ID  PLANT.REGION
   8   FILE CLERK            30200     SW
  15   SECRETARY             10100     SW
                             20150     MW
                             30200     SW
                             50300     NW
                             60200     NE
                             70500     SE
  25   ACCOUNTANT            20150     MW
  28   ENGINEER              20150     MW
                             70500     SE
  40   EXPEDITER             30200     SW
                             70500     SE
  44   COORDINATOR           50300     NW
                             60200     NE
  45   BOOKKEEPER            30200     SW
  46   SALESPERSON           10100     SW
  50   ADMINISTRATOR         20150     MW
                             30200     SW
                             50300     NW
```

Figure 8-1 FIND Statement

Below is the inquiry followed by a detailed explanation of how it works.

```
FIND SKILL %CODE=SKILL.CODE %SKILL=SKILL.NAME %PLANT=PLANT.ID;
DISPLAY PLANT %CODE %SKILL PLANT.ID PLANT.REGION IF PLANT.ID =
%PLANT;;
```

`FIND`	Links two databases.
`SKILL`	Specifies the name of the first database.
`%CODE = SKILL.CODE`	Assigns a value of SKILL.CODE to a temporary field and transfers the value to a DISPLAY statement.
`%SKILL = SKILL.NAME`	Assigns a related value of SKILL.NAME to a temporary field for transfer to the DISPLAY statement.

`%PLANT = PLANT.ID`	Assigns a value of PLANT.ID to a temporary field for transfer to the DISPLAY statement.
`;`	Terminates the FIND statement.
`DISPLAY`	Specifies the command to list.
`PLANT`	Specifies the name of the second database.
`%CODE` `%SKILL`	Displays these fields from the first database.
`PLANT.ID` `PLANT.REGION`	Displays these fields from the PLANT database.
`IF PLANT.ID = %PLANT`	Tests a field from the PLANT database against a field from the first database for matching values.
`;;`	Terminates the inquiry.

This example tells VISION:Inquiry to display fields from both the SKILL and PLANT databases, and to align the databases by matching the PLANT.ID fields.

FIND Statement

The FIND statement tells VISION:Inquiry which information is needed from the first database. In this example, the SKILL database is accessed first.

- %CODE = SKILL.CODE assigns one value of SKILL.CODE to %CODE. (The first code listed in the database is 8, and is assigned to %CODE the first time through the inquiry.)
- %SKILL = SKILL.NAME assigns the related value of SKILL.NAME to %SKILL. (File clerk is the first related value in this instance.)
- %PLANT = PLANT.ID assigns a value of PLANT.ID to %PLANT. VISION:Inquiry goes to the first plant listed under 'FILE CLERK' (30200, in this example) and assigns it to %PLANT.

At this point, the first execution of the FIND statement has been completed and execution moves to the DISPLAY statement.

DISPLAY Statement

The DISPLAY statement indicates which fields are needed from the second database, which field to match on, and how to manipulate and display the items listed. This example displays two fields from the PLANT database (PLANT.ID and PLANT.REGION) and two fields from the SKILL database (%SKILL and %CODE).

The conditional phrase, IF PLANT.ID = %PLANT, performs the matching operation. The first time through the inquiry, %PLANT is equal to 30200, and VISION:Inquiry searches for the matching value in PLANT. Because PLANT.ID is a key field from the root segment database, VISION:Inquiry can go directly to the root segment that has a key value of 30200. It accesses the related value of PLANT.REGION and displays the fields indicated in the DISPLAY statement. At this point, the first execution of the inquiry is completed, and control returns to the FIND statement for the next execution.

There is only one ‘FILE CLERK’ in the toy company; therefore, the second time through the inquiry, %CODE is assigned a value of 15 and %SKILL becomes ’SECRETARY’. The first PLANT.ID listed under ’SECRETARY’ is 10100, which is assigned to %PLANT. VISION:Inquiry accesses the matching PLANT.ID in the PLANT database and displays the appropriate line of data.

Control once again returns to the FIND statement. More than one plant is listed under ’SECRETARY’, so the next PLANT.ID, 20150, is assigned to %PLANT for transfer to the DISPLAY statement. The values of %CODE and %SKILL remain the same. This process continues until all the PLANT.IDs for each SKILL.CODE have been transferred and displayed.

The display statement can also contain one or two title lines as explained in Chapter 5, “Simple Reports”.

Matching Value Not Found

If a matching value for %PLANT is not found in the PLANT database, no line is displayed, and control returns to the FIND statement for the next execution of the inquiry.

Output

The output in Figure 8-1 is sequenced in the order in which the data is stored in the database accessed by the FIND statement. The fields appear in the order in which they are listed in the DISPLAY statement.

Terminating with One Semicolon and Two Semicolons

Note that the FIND statement is terminated with one semicolon and that two semicolons are used to terminate the inquiry.

- The double semicolons are required on systems operating in batch mode.
- A single semicolon would be sufficient to terminate the inquiry for other systems.

Matching

In the conditional selection phrase that performs the matching operation, the field from the database accessed by the DISPLAY statement must be on the left side of the relational test, and the field transferred from the FIND statement must be on the right side.

Figure 8-1 also illustrates the ability of VISION:Inquiry to match on numeric data regardless of whether it is stored in alphanumeric or numeric fields. PLANT.ID is defined as alphanumeric in the PLANT database and as numeric in the SKILL database.

- Matching on key fields of the *root segment* allows VISION:Inquiry to go directly to the appropriate root segment in the second database.
- Matching on key fields of *lower segments* only, or on non-key fields, requires VISION:Inquiry to search sequentially through all segments to find the one with the matching value. This requires more computer time.

Figure 8-2 matches on a lower level key field.

Example: "Produce a list of employees by job classification. Show their plant identification."

```
PAGE:      TRANCODE: II        INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO;
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL
IF EMP.NO = %EMP.NUM;;

PLANT.ID  EMP.NO   EMP.NAME                    SKILL
30200      30215   SHARON DALEY                FILE CLERK
10100      10105   MARY ANN THOMAS             SECRETARY
20150      21116   WILMA FORD
30200      30207   JANE LOWELL
50300      50323   VICKY WARD
60200      60258   KAREN REDFERN
70500      70522   AGNES COVINGTON
20150      21124   CHARLES SALTER              ACCOUNTANT
20150      21164   SUSAN WARE                  ENGINEER
70500      70529   MARTHA WALLINGHAM
30200      30201   JOHN HENRY CRANE            EXPEDITER
70500      70519   STEPHEN MCGEE
50300      50322   MADELYN BATES               COORDINATOR
60200      60251   MARCIE MORINO
30200      30211   PATRICIA BLAKELY            BOOKKEEPER
```

Figure 8-2   FIND Statement with Match on a Lower Level Key Field

VISION:Inquiry assigns the first value of SKILL.NAME to %SKILL (file clerk) and the first related value of EMP.NO to %EMP.NUM (30215). Control drops to the DISPLAY statement. VISION:Inquiry sequentially searches through the root segments and the related lower level segments until it finds the matching value of EMP.NO (30215). At that point, it displays all the related information from both databases.

Then it continues searching through the rest of the segments in the database to see if there are any other EMP.NOs with a value of 30215. (In this database, there is only one.) When all segments have been searched, control returns to the FIND statement to process the next value of SKILL.NAME (secretary) and searches for and displays all secretarial employees.

Had the inquiry matched on both PLANT.ID (a root key field) and EMP.NO (a lower level segment key field), VISION:Inquiry could directly access both PLANT.ID and EMP.NO in the second database.

```
FIND SKILL %SKILL=SKILL.NAME %EMP.NUM=EMP.NO %PLANT=PLANT.ID;
DISPLAY PLANT PLANT.ID EMP.NAME %SKILL IF EMP.NO = %EMP.NUM
AND PLANT.ID = %PLANT;
```

Again, VISION:Inquiry assigns the first value of SKILL.NAME to %SKILL. Then it assigns the first related value of PLANT.ID to %PLANT and the first related value of EMP.NO to %EMP.NUM.

Control passes to the DISPLAY statement. The conditional phrase now matches on two fields. Because PLANT.ID is a root key field, VISION:Inquiry goes directly to the appropriate root segment. EMP.NO is also a segment key field, so VISION:Inquiry can now go directly to the correct EMPLOYEE segment. No sequential searching is involved.

The displayed output from both inquiries is the same.

Data Selection

You can add a conditional selection phrase to the FIND statement to limit the data transferred to the DISPLAY statement.

```
FIND SKILL %SKILL=SKILL.NAME %EMP.NUM=EMP.NO

  IF SKILL.NAME='SECRETARY';
```

conditional phrase limiting information transferred to the DISPLAY statement

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL IF EMP.NO = %EMP.NUM ;
```

VISION:Inquiry assigns the value of 'SECRETARY' to SKILL.NAME and %SKILL and finds the first employee listed under that job classification. At this point, EMP.NO is set to 10105 and transferred to the DISPLAY statement where it is matched and the related information displayed.

Then, the next EMP.NO for 'SECRETARY' is found, transferred to the DISPLAY statement, and listed. This continues until all secretaries have been accessed and displayed. Figure 8-3 illustrates this example.

Example: "List the employees who hold the job classification of secretary."

```
PAGE:      TRANCODE: II       INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'SECRETARY';
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL
IF EMP.NO = %EMP.NUM;;
PLANT.ID  EMP.NO   EMP.NAME                   SKILL
10100      10105   MARY ANN THOMAS            SECRETARY
20150      21116   WILMA FORD
30200      30207   JANE LOWELL
50300      50323   VICKY WARD
60200      60258   KAREN REDFERN
70500      70522   AGNES COVINGTON
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (129,55 USER DB CALLS,ROOTS)
```

Figure 8-3   FIND Statement Containing a Conditional Phrase

The conditional selection phrase in the FIND statement can also contain the LIKE operator for partial matching on character fields as explained in Chapter 5, "Simple Reports".

Conditional Selection in a FIND

Note that a conditional selection phrase in the FIND statement can only test the fields from the database being accessed by that statement. It cannot test a temporary field or a field from another database. The phrase

```
IF %SKILL = 'SECRETARY'
```

is invalid, because %SKILL is a temporary field. The inquiry would not be executed.

The conditions in the conditional selection phrase in the FIND statement must be satisfied before control passes to the DISPLAY statement.

In Figure 8-3, SKILL is a group field and appears only once on the listing.

Expanding the Conditional Selection in a DISPLAY Statement

The conditional selection phrase in the DISPLAY statement can be expanded to further limit the data selected for output.

```
IF EMP.NO = %EMP.NUM AND SAL.YEAR = 95 AND SAL.YTD LT 24000;
```

Two conditions have been added to the conditional selection phrase in the DISPLAY statement, which further limits output to secretaries earning less than $24,000 in 1995. The test for matching fields is still included in the phrase. The fields being tested in the conditional selection phrase are from the database being accessed by the DISPLAY statement. Figure 8-4 shows the output.

Example: "List only secretaries who earned less than $24,000 in 1995."

```
PAGE:       TRANCODE: II       INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'SECRETARY';
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL SAL.YTD
IF EMP.NO = %EMP.NUM AND SAL.YEAR = 95 AND SAL.YTD LT 24000;;
PLANT.ID  EMP.NO EMP.NAME                  SKILL              SAL.YTD
10100     10105  MARY ANN THOMAS           SECRETARY       15,600.00
20150     21116  WILMA FORD                                18,800.00
50300     50323  VICKY WARD                                20,000.00
70500     70522  AGNES COVINGTON                           23,000.00
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (141,55 USER DB CALLS,ROOTS)
```

Figure 8-4 Expanding Conditional Selection with a Phrase in the DISPLAY Statement

Expanded Conditional Selection in FIND and DISPLAY Statements

The full range of logical expressions can be used in the conditional selection phrase of the DISPLAY and FIND statements.

```
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO IF SKILL.NAME =
'SECRETARY' AND (PLANT.ID = 10100 OR PLANT.ID GT 60000) ;
```

This conditional phrase further limits output to secretaries from only some of the plants. The conditional selection phrase could have been written to test specifically for each of the two plants; however, in this case, it is more efficient to use the GT operator. The output is shown in Figure 8-5.

Example: “Further limit the output from Figure 8-4 to show only secretaries from plants 10100, 60200, and 70500 who meet the other criteria.”

```
PAGE:       TRANCODE: II       INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'SECRETARY' AND (PLANT.ID = 10100 OR PLANT.ID GT 60000);
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL SAL.YTD
IF EMP.NO = %EMP.NUM AND SAL.YEAR = 95 AND SAL.YTD LT 24000;;
PLANT.ID EMP.NO  EMP.NAME                SKILL                 SAL.YTD
10100     10105  MARY ANN THOMAS         SECRETARY           15,600.00
70500     70522  AGNES COVINGTON                             23,000.00
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (89,34 USER DB CALLS,ROOTS)
```

Figure 8-5   Expanded Conditional Selection Phrase in the FIND Statement

The secretaries from plants 10100 and 70500 are the only ones who meet the present criteria; this data is displayed on the report.

Because PLANT.ID occurs in both databases accessed by the FIND and DISPLAY statements, the conditional tests for the plants could have been put in the DISPLAY statement, rather than in the FIND statement. However, this would be less efficient and more time consuming. The earlier that data selection takes place, the less amount of data that must be transferred by the FIND statement.

Sequencing Data Output

Data displayed by an inquiry containing the FIND command is in order according to the way it is stored in the database accessed by the FIND statement. Figure 8-6 shows the output when the FIND statement accesses the SKILL database.

Example: "Display the list of employees in the order in which the employees are stored in the SKILL database. Show the SKILL.CODE, SKILL.NAME, and SAL.YTD for 1995."

```
PAGE:      TRANCODE: II    INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO %CODE = SKILL.CODE;
DISPLAY PLANT PLANT.ID EMP.NAME %CODE %SKILL SAL.YTD IF EMP.NO =
%EMP.NUM  AND SAL.YEAR = 95;
PLANT.ID  EMP.NAME                    CODE   SKILL              SAL.YTD
30200     SHARON DALEY                   8   FILE CLERK      17,000.00
10100     MARY ANN THOMAS               15   SECRETARY       15,600.00
20150     WILMA FORD                                         18,800.00
30200     JANE LOWELL                                        26,000.00
50300     VICKY WARD                                         20,000.00
60200     KAREN REDFERN                                      26,000.00
70500     AGNES COVINGTON                                    23,000.00
20150     CHARLES SALTER                25   ACCOUNTANT      39,000.00
20150     SUSAN WARE                    28   ENGINEER        41,000.00
70500     MARTHA WALLINGHAM                                  33,000.00
30200     JOHN HENRY CRANE              40   EXPEDITER       22,000.00
70500     STEPHEN MCGEE                                      23,000.00
50300     MADELYN BATES                 44   COORDINATOR     32,000.00
60200     MARCIE MORINO                                      31,000.00
30200     PATRICIA BLAKELY              45   BOOKKEEPER      30,000.00
10100     WILLIAM AMES                  46   SALESPERSON     64,000.00
10100     PHYLLIS LOCKMEYER                                  59,000.00
```

Figure 8-6 Data Sequenced by Order in which it is Stored in the First Database Accessed

Note that the data is ordered and grouped by SKILL.CODE.

SORT in Related DISPLAY Segment

You can use the SORT command in the related DISPLAY segment to change the order in which the displayed data is sequenced. This is illustrated in Figure 8-7.

Example: "Display the same data shown in Figure 8-6, except show it in order by PLANT.ID."

```
PAGE:      TRANCODE: II    INQUIRY:
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO %CODE = SKILL.CODE;
DISPLAY PLANT PLANT.ID EMP.NAME %CODE %SKILL SAL.YTD SORT PLANT.ID
IF EMP.NO = %EMP.NUM AND SAL.YEAR = 95;;
PLANT.ID  EMP.NAME                CODE   SKILL                SAL.YTD
10100     MARY ANN THOMAS           15   SECRETARY          15,600.00
          WILLIAM AMES              46   SALESPERSON        64,000.00
          PHYLLIS LOCKMEYER         46   SALESPERSON        59,000.00
20150     WILMA FORD                15   SECRETARY          18,800.00
          CHARLES SALTER            25   ACCOUNTANT         39,000.00
          SUSAN WARE                28   ENGINEER           41,000.00
          PETER ZATKIN              50   ADMINISTRATOR      56,000.00
30200     SHARON DALEY               8   FILE CLERK         17,000.00
          JANE LOWELL               15   SECRETARY          26,000.00
          JOHN HENRY CRANE          40   EXPEDITER          22,000.00
          PATRICIA BLAKELY          45   BOOKKEEPER         30,000.00
          FREDERICH GRAY            50   ADMINISTRATOR      59,000.00
          MITCHELL J HOOPS          60   VICE PRESIDENT     98,000.00
40300     DONALD M KING             70   DESIGNER           75,000.00
          JOAN EVANS                70   DESIGNER           73,200.00
50300     VICKY WARD                15   SECRETARY          20,000.00
          MADELYN BATES             44   COORDINATOR        32,000.00
          JONATHAN OAKS             50   ADMINISTRATOR      46,000.00
60200     KAREN REDFERN             15   SECRETARY          26,000.00
```

Figure 8-7 SORT in Related DISPLAY Segment

SORT PLANT.ID has been added to the DISPLAY statement to reorder the data by PLANT.ID, rather than by SKILL.CODE. Note that the data is grouped by PLANT.ID. Fields from both databases accessed by the FIND and DISPLAY statements can be sorted.

Using FIND with an Arithmetic Calculation

You can also perform arithmetic calculations in the FIND statement.

```
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO
%SALARY = SAL.YTD - SAL.DED IF SAL.YEAR = 95;
```

In this example, an arithmetic expression has been placed in the FIND statement to calculate the net value of salary for each employee:

```
%SALARY = SAL.YTD - SAL.DED
```

The value of %SALARY is transferred to the DISPLAY statement for output.

Example: “Display a list of employees showing their EMP.NOs, EMP.NAMEs, net salaries for 1995, and SKILL.NAMEs.”

```
PAGE:       TRANCODE: II       INQUIRY:
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY =
SAL.YTD - SAL.DED IF SAL.YEAR = 95;
DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME %SALARY IF
EMP.NO = %EMP.NUM;;

EMP.NO  EMPLOYEE                   SKILL.NAME                SALARY
10103   WILLIAM AMES               SALESPERSON            55,000.00
10104   PHYLLIS LOCKMEYER          SALESPERSON            50,800.00
10105   MARY ANN THOMAS            SECRETARY              14,230.00
21116   WILMA FORD                 SECRETARY              16,820.00
21124   CHARLES SALTER             ACCOUNTANT             33,000.00
21137   PETER ZATKIN               ADMINISTRATOR          46,420.00
21164   SUSAN WARE                 ENGINEER               34,000.00
30201   JOHN HENRY CRANE           EXPEDITER              19,400.00
30202   FREDERICH GRAY             ADMINISTRATOR          45,400.00
30205   MITCHELL J HOOPS           VICE PRESIDENT         78,002.00
30207   JANE LOWELL                SECRETARY              20,200.00
30211   PATRICIA BLAKELY           BOOKKEEPER             25,880.00
30215   SHARON DALEY               FILE CLERK             15,440.00
40304   DONALD M KING              DESIGNER               57,600.00
```

Figure 8-8 FIND Statement with an Arithmetic Calculation

Note that in this example, the FIND statement accesses the PLANT database and the DISPLAY statement accesses the SKILL database.

The expression that calculates the net salary is placed in the FIND statement because the fields involved are part of the PLANT database. The results of the calculations are transferred to the DISPLAY statement and listed under the name SALARY.

Testing the Results of an Arithmetic Calculation in a FIND

The results of arithmetic processing can be used in a conditional selection phrase to limit the data that is transferred to the DISPLAY statement.

```
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY = SAL.YTD
- SAL.DED IF SAL.YEAR = 95 AND SAL.YTD - SAL.DED GT 30000;
```

The calculation that determines the net salary is shown twice in the FIND statement.

- The first time, the results are assigned to a temporary field (%SALARY) for transfer to the DISPLAY statement.
- The second time, the calculation is performed within a conditional phrase in a relational expression, to compare the net salary to 30000.

Only information relating to 1995 net salaries that are greater than 30000 is transferred to the DISPLAY statement. The temporary field %SALARY cannot be tested directly in the conditional selection phrase in the FIND statement. Figure 8-9 shows the output.

Example: "Modify the listing in Figure 8-8 to show only those employees whose 1995 net salaries are greater than $30,000."

```
PAGE:       TRANCODE: II       INQUIRY:
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY =
SAL.YTD - SAL.DED IF SAL.YEAR = 95 AND SAL.YTD - SAL.DED GT 30000;
DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME %SALARY IF
EMP.NO = %EMP.NUM;;

EMP.NO  EMPLOYEE                   SKILL.NAME                  SALARY
10103   WILLIAM AMES               SALESPERSON              55,000.00
10104   PHYLLIS LOCKMEYER          SALESPERSON              50,800.00
21124   CHARLES SALTER             ACCOUNTANT               33,000.00
21137   PETER ZATKIN               ADMINISTRATOR            46,420.00
21164   SUSAN WARE                 ENGINEER                 34,000.00
30202   FREDERICH GRAY             ADMINISTRATOR            45,400.00
30205   MITCHELL J HOOPS           VICE PRESIDENT           78,002.00
40304   DONALD M KING              DESIGNER                 57,600.00
40306   JOAN EVANS                 DESIGNER                 57,600.00
50304   JONATHAN OAKS              ADMINISTRATOR            35,960.00
60205   DAVID YORK                 PROMOTIONS WRITER        36,200.00
60209   RUSSELL M SIMMONS          SENIOR ADMINISTRATOR     49,860.00
70511   RONALD T JACKSON           ADMINISTRATOR            45,200.00
*
 IXX9121  END OF INQUIRY.              (865,176 USER DB CALLS,ROOTS)
```

Figure 8-9 Testing the Results of an Arithmetic Calculation in a FIND Statement

Only the information relating to employees who meet our criteria is displayed.

The calculation that assigns the value of net salary to %SALARY is necessary to display the net salary on the output. If net salary did not need to be displayed, the calculation could have been eliminated. The calculation is needed in the conditional selection phrase to limit the data that is transferred to the DISPLAY statement.

Using a FIND Arithmetic Calculation in a DISPLAY Arithmetic Calculation

The results of an arithmetic calculation performed in a FIND statement can be used in an arithmetic calculation in the related DISPLAY statement.

```
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY =
SAL.YTD - SAL.DED IF SAL.YTD - SAL.DED GT 30000 AND SAL.YEAR = 95;
DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME %RAISE = %SALARY
* 0.17 IF EMP.NO = %EMP.NUM;
```

Each value of %SALARY transferred from the FIND statement is multiplied by 0.17 to calculate the value of %RAISE. The output is shown in Figure 8-10.

Example: "Calculate and display a raise of 17% for employees listed in Figure 8-9."

```
PAGE:       TRANCODE: II       INQUIRY:
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY =
SAL.YTD - SAL.DED IF SAL.YTD - SAL.DED GT 30000 AND SAL.YEAR = 95;
DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME %RAISE = %SALARY
* 0.17 IF EMP.NO = %EMP.NUM;;
EMP.NO  EMPLOYEE                   SKILL.NAME                  RAISE
10103   WILLIAM AMES               SALESPERSON            9,350.0000
10104   PHYLLIS LOCKMEYER          SALESPERSON            8,636.0000
21124   CHARLES SALTER             ACCOUNTANT             5,610.0000
21137   PETER ZATKIN               ADMINISTRATOR          7,891.4000
21164   SUSAN WARE                 ENGINEER               5,780.0000
30202   FREDERICH GRAY             ADMINISTRATOR          7,718.0000
30205   MITCHELL J HOOPS           VICE PRESIDENT        13,260.3400
40304   DONALD M KING              DESIGNER               9,792.0000
40306   JOAN EVANS                 DESIGNER               9,792.0000
50304   JONATHAN OAKS              ADMINISTRATOR          6,113.2000
60205   DAVID YORK                 PROMOTIONS WRITER      6,154.0000
60209   RUSSELL M SIMMONS          SENIOR ADMINISTRATOR   8,476.2000
70511   RONALD T JACKSON           ADMINISTRATOR          7,684.0000
*
IXX9121  END OF INQUIRY.                (865,176 USER DB CALLS,ROOTS)
```

Figure 8-10 Using the Result Field of an Arithmetic Calculation from the FIND Statement in a Calculation in the DISPLAY Statement

The calculation for net salary is still used twice in the FIND statement — once to limit the data selected and once to assign a value to %SALARY which is used in the calculation in the DISPLAY statement.

Testing a Result Field in a DISPLAY Statement

The result field of a calculation in a DISPLAY statement that includes a field from a FIND statement can be tested in a conditional phrase under certain circumstances; the calculation must include a field from the database accessed by the DISPLAY statement.

```
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY = SAL.YTD - SAL.DED
IF SAL.YTD - SAL.DED GT 30000 AND SAL.YEAR = 95 ; DISPLAY SKILL EMP.NO %EMPLOYEE
SKILL.NAME
 %AMOUNT = %SALARY + SKILL.CODE IF %AMOUNT GT 50000 AND EMP.NO = %EMP.NUM;
```

result field	field from FIND statement	field from SKILL database	result field in conditional phrase

In this inquiry, a result field (%AMOUNT) calculated in the DISPLAY statement is tested in a conditional phrase. %SALARY is a result field that has been transferred from the FIND statement. SKILL.CODE is a field from the database accessed by the DISPLAY statement (SKILL). The conditional phrase compares the result field of this calculation (%AMOUNT) to 50000. Only values greater than 50000 are displayed. The output is shown in Figure 8-11.

Example: "Calculate and display a value named %AMOUNT by adding net salary to SKILL.CODE. Display only those employees whose AMOUNT is greater than 50000."

```
 PAGE:       TRANCODE: II       INQUIRY:
 FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY =
 SAL.YTD - SAL.DED IF SAL.YTD - SAL.DED GT 30000 AND SAL.YEAR = 95;
 DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME %AMOUNT = %SALARY
 + SKILL.CODE IF %AMOUNT GT 50000 AND EMP.NO = %EMP.NUM;;
 EMP.NO  EMPLOYEE                   SKILL.NAME                AMOUNT
 10103   WILLIAM AMES               SALESPERSON            55,046.00
 10104   PHYLLIS LOCKMEYER          SALESPERSON            50,846.00
 30205   MITCHELL J HOOPS           VICE PRESIDENT         78,062.00
 40304   DONALD M KING              DESIGNER               57,670.00
 40306   JOAN EVANS                 DESIGNER               57,670.00
 *
 *
 *
 *
 *
 *
 *
 *
 *
IXX9121  END OF INQUIRY.                (506,176 USER DB CALLS,ROOTS)
```

Figure 8-11   Testing a Result Field in a Conditional Phrase in a DISPLAY Statement

The results of the calculation in the DISPLAY statement appear under the column heading AMOUNT.

The conditional test IF SAL.YTD - SAL.DED GT 30000 in the FIND statement could be deleted from the inquiry without changing the results. It is, however, more efficient to limit the amount of data transferred to the DISPLAY statement.

Calculation from a FIND Statement Used in a Calculation in Conditional Selection Phrase

A calculation containing fields from both databases can also be tested directly in a conditional selection phrase in the DISPLAY statement.

```
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY = SAL.YTD - SAL.DED
  IF SAL.YTD - SAL.DED GT 30000  AND SAL.YEAR = 95;
  DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME  IF
    %SALARY  +  SKILL.CODE  GT 50000 AND EMP.NO = %EMP.NUM;
```

field from FIND statement	field from SKILL database

The conditional phrase in the DISPLAY statement compares the sum of %SALARY (from the FIND statement) and SKILL.CODE (from the SKILL database) to 50000. The output is shown in Figure 8-12.

Example: “List only those employees whose net salary added to SKILL.CODE is greater than 50000.”

```
PAGE:      TRANCODE: II      INQUIRY:
FIND PLANT %EMPLOYEE = EMP.NAME %EMP.NUM = EMP.NO %SALARY = SAL.YTD - SAL.DED
  IF SAL.YTD - SAL.DED GT 30000 AND SAL.YEAR = 95 ;
  DISPLAY SKILL EMP.NO %EMPLOYEE SKILL.NAME IF
  %SALARY + SKILL.CODE GT 50000 AND EMP.NO = %EMP.NUM;;
EMP.NO  EMPLOYEE                   SKILL.NAME
10103   WILLIAM AMES               SALESPERSON
10104   PHYLLIS LOCKMEYER          SALESPERSON
30205   MITCHELL J HOOPS           VICE PRESIDENT
40304   DONALD M KING              DESIGNER
40306   JOAN EVANS                 DESIGNER
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.              (502,176 USER DB CALLS,ROOTS)
```

Figure 8-12 Arithmetic Calculation with Field from a FIND Statement Used in Calculation in Conditional Selection Phrase

Because the sum of %SALARY and SKILL.CODE is not assigned to a temporary field, it is not displayed in the listing.

Note that the calculation in the conditional selection phrase contains a field from the database accessed by the DISPLAY statement. VISION:Inquiry does not perform the conditional test unless it is present.

Accessing More Than Two Databases

You can use multiple FIND statements in an inquiry to access up to 16 databases. All databases may match on the same field, or each database may match on a field that relates only to the next database accessed.

```
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO;
FIND PAYROLL %HOURS = (REG.HOURS + OVER.HOURS) * RATE %ID = EMP.ID
  IF EMP.ID = %EMP.NUM;
  DISPLAY PLANT PLANT.ID EMP.NAME %SKILL %HOURS IF EMP.NO =%ID;
```

This inquiry accesses three databases: SKILL, PAYROLL, and PLANT. (PAYROLL is a fictitious database shown for illustration only.) In this inquiry, all three databases match on employee identification number.

- The first FIND statement assigns a value of SKILL.NAME to %SKILL and the first related value of EMP.NO to %EMP.NUM.
- The second FIND statement calculates %HOURS by adding REG.HOURS to OVER.HOURS and multiplying the sum by RATE. REG.HOURS, OVER.HOURS, and RATE are fields from the PAYROLL database. The related value of EMP.ID is assigned to %ID for transfer to the DISPLAY statement.
- The second FIND statement must also match against a field from the first FIND statement in order to locate the related information; the conditional phrase, IF EMP.ID = %EMP.NUM, performs this function.
  - VISION:Inquiry searches all the EMP.ID numbers in PAYROLL until it finds the one that matches %EMP.NUM (from the first FIND statement). Then it assigns EMP.ID to %ID, calculates %HOURS, and transfers those values to the DISPLAY statement.
  - If VISION:Inquiry cannot find a value of EMP.ID to match %EMP.NUM, it does not transfer any values to the DISPLAY statement. It returns to the first FIND statement and assigns the next value of EMP.NO to %EMP.NUM. It drops down again to the second FIND statement and looks for a match. This continues until all values of EMP.NO in the first FIND statement have been processed.

Assume that the first time through the inquiry you find the matching value of EMP.ID for %EMP.NUM and that the second FIND statement is executed for this employee. The related values of %HOURS and %ID are transferred to the DISPLAY statement.

VISION:Inquiry executes IF EMP.NO = %ID and finds the matching value of EMP.NO. Then it displays the related values of PLANT.ID and EMP.NAME, along with %SKILL and %HOURS (transferred from the FIND statements). VISION:Inquiry then returns to the first FIND statement to repeat the process on the next value of EMP.NO.

If there is no matching value for %ID in the PLANT database, the DISPLAY statement is not executed and no line is output. Control returns to the first FIND statement to find the next value of EMP.NO, and the process is repeated.

In this example, the databases matched on the same field. The next example illustrates matching on different fields.

```
FIND SKILL     %PLANT = PLANT.ID;
FIND PAYROLL   %EMP = EMP.ID IF PLANT.NO = %PLANT;
DISPLAY PLANT  PLANT.ID %EMP IF EMP.NO = %EMP;
```

The first and second databases match on plant identification number, and the second and third databases match on employee identification number.

- The first FIND statement performs whatever steps are indicated and assigns a value of PLANT.ID to %PLANT for transfer to the second FIND statement.
- The second FIND statement locates a matching value of PLANT.NO (if it is present), performs whatever steps are indicated on the related data, and assigns the appropriate value of EMP.ID to %EMP for transfer to the DISPLAY statement.
- The DISPLAY statement finds the matching value of EMP.NO (if present) for %EMP and performs the indicated steps. This completes the first pass through the inquiry, and information on one employee is displayed. If there is only one employee in a plant, control returns to the first FIND statement and repeats the process for the next plant.

Assume, however, that in this case there are many employees working at each plant. After you have completed the first pass through the inquiry and displayed the information for the first employee, control returns to the second FIND statement, assigns the next value of EMP.ID to %EMP, and repeats the procedures in that statement for the second employee. Then control drops to the DISPLAY statement, matches EMP.NO to the new value of %EMP, and displays the related information for the second employee.

Again, control returns to the second FIND statement to process the third employee. This procedure is repeated until all employees for the first plant have been processed. Only then will control return to the first FIND statement to repeat the entire process for the next plant.

Accessing the Same Database More Than One Time

FIND and DISPLAY statements in an inquiry do not have to access different databases; they can access the same database. The same process of matching on a field and assigning values to temporary fields for transfer must still be done.

Any-All Conditional Selection

This is a useful technique in certain situations involving “any-all” conditional selection; if ANY occurrence of a field meets a specific condition, show ALL the occurrences of that field.

For example, if the toy company wanted to see all the degrees held by those employees who have Master of Science degrees, they would enter the following inquiry:

```
FIND PLANT %EMPLOYEE = EMP.NO IF ED.DEGREE = 'MS';
DISPLAY PLANT EMP.NO EMP.NAME ED.DEGREE IF EMP.NO =
%EMPLOYEE;
```

The output is shown in Figure 8-13.

Example: “List all the degrees held by employees with MS degrees.”

```
PAGE:       TRANCODE: II       INQUIRY:
FIND PLANT %EMPLOYEE = EMP.NO IF ED.DEGREE = 'MS';
DISPLAY PLANT EMP.NO EMP.NAME ED.DEGREE IF EMP.NO = %EMPLOYEE;;


EMP.NO   EMP.NAME                   ED.DEGREE
 40304   DONALD M KING              BS
                                    MS
                                    PD
 40306   JOAN EVANS                 BS
                                    MS
                                    PD
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (94,21 USER DB CALLS,ROOTS)
```

Figure 8-13 Accessing the Same Database Twice in Any-All Conditional Selection

Two employees have Master of Science degrees. Both also hold Bachelor of Science and Doctoral degrees.

Selection on Sibling Segments

Note: For IMS (DL/I) and VSAM hierarchical databases only. Not applicable to VSAM non-hierarchical files or DB2 tables.

VISION:Inquiry performs data selection in hierarchical sequence. As a result, not all data for two segments on the same level is displayed if they are both used in the conditional selection.

For example, if it is required to print all occurrences of the SAL and ED segments if one of the SAL.YEARs is '94' or one of the degrees is 'MS', the following does not work:

```
DISPLAY PLANT EMP.NAME SAL.YEAR ED.DEGREE IF SAL.YEAR =
'94' OR ED.DEGREE = 'MS';
```

Structuring the inquiry as follows gives the desired results.

```
FIND PLANT %EMP.NO = EMP.NO IF SAL.YEAR = '94' OR ED.DEGREE = 'MS';
DISPLAY PLANT EMP.NAME SAL.YEAR ED.DEGREE IF EMP.NO =  %EMP.NO;
```

Note that the field EMP.NO must have a unique value in each employee segment.

Using FIND with a Manufacturing Database

Accessing the same database twice is also useful when dealing with databases in which related elements are stored in different locations; this is a common situation in manufacturing environments.

For example, assume that the toy company has a database named ASSEMBLY, which lists the components of each of their toys. The structure looks like this.

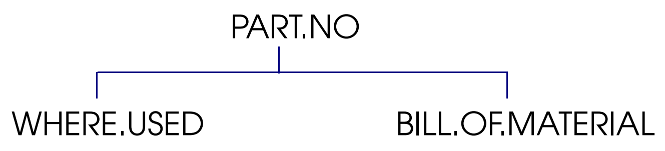


Figure 8-14 ASSEMBLY Database

- The PART.NO segment indicates the part number of a salable product, a major assembly, a subassembly, or type of raw material, and contains information on that component.
- The WHERE.USED segment contains information about the toy (or higher level assembly) in which the PART.NO is used.
- The BILL.OF.MATERIAL segment holds information about each component used in a toy or assembly, and indicates the part number of the component and the number required for making one of the assembly.

There is an occurrence in the PART.NO segment for each part number in the BILL.OF.MATERIAL segment, which provides inventory information on that component.

Suppose the toy company wanted to check availability of all parts required for the wheel assembly of the dumpster truck. It would enter the following inquiry:

```
FIND ASSEMBLY %USED = WHERE.USED %PART = PART.NO %BILL =
BILL.OF.MATERIAL IF PART.NO = 'WHEEL' AND WHERE.USED = 'DUMPSTER';
DISPLAY ASSEMBLY %USED %PART %BILL QTY.ON.HAND QTY.RQD IF PART.NO =
%BILL;
```

VISION:Inquiry first locates the PART.NO segment for 'WHEEL' and the related WHERE.USED segment for 'DUMPSTER', and then assigns the first related value of BILL.OF.MATERIAL to %BILL (assume 302). Then VISION:Inquiry locates the PART.NO segment for 302 and displays the related inventory information. This process continues until all components have been listed.

Accessing VSAM Files and an IMS (DL/I) Database

You can use the FIND command to access a combination of up to 16 VSAM files and databases. The following are examples that use both a VSAM file and an IMS (DL/I) database. The description of the VSAM test file, used in these examples, can be found in Chapter 2, "Databases and Files" of this Reference Guide.

Figure 8-15 uses the FIND and DISPLAY commands as well as a conditional phrase to access the IMS (DL/I) SKILL database and the VSAM VSPLANT file. The following examples use an IMS (DL/I) database and a VSAM file under CICS.

VISION:Inquiry searches for the first occurrence of 'SECRETARY' and assigns that value to the temporary field %SKILL. The related EMP.NO is assigned to %EMP.NUM; control then goes to the DISPLAY command. Figure 8-15 displays the output.

Example: "List the employees that have a secretarial position. Use the IMS (DL/I) SKILL database and VSAM VSPLANT file."

```
 PAGE:     TRANSACTION: IQIO
                        Enter Inquiry Below:
 FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO IF SKILL.NAME =
'SECRETARY' ; DISPLAY VSPLANT VSPLANT.ID VSEMP.NO VSEMP.NAME %SKILL IF
 VSEMP.NO = %EMP.NUM;;

 VSPLANT.ID  VSEMP.NO   VSEMP.NAME                 SKILL
 10100         10,105   MARY ANN THOMAS            SECRETARY
 20150         21,116   WILMA FORD
 30200         30,207   JANE LOWELL
 50300         50,323   VICKY WARD
 60200         60,258   KAREN REDFERN
 70500         70,522   AGNES COVINGTON
 *
 *
 *
 *
 *
 *
 IXX9121  END OF INQUIRY.                (27,19 USER DB CALLS,ROOTS)
  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
 (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 8-15 FIND Statement that Accesses an IMS (DL/I) Database and a VSAM File

In Figure 8-15, the SKILL.NAME field (secretary) is from a structured database, SKILL. Multiple occurrences of segments from a database are suppressed; the value from the SKILL.NAME field, 'SECRETARY', is printed only once.

Accessing a VSAM FIle, an IMS (DL/I) Database, and another VSAM FIle

The next example illustrates how VISION:Inquiry can access fields from three databases and/or files.

The first FIND statement uses the VSAM file, VSSKILL. The second FIND statement uses the IMS (DL/I) PLANT database, and then a DISPLAY command is used to access the VSAM file, VSPLANT.

Example: "List the employees, their jobs, and net salaries by plant."

```
PAGE:       TRANSACTION: IQIO
                         Enter Inquiry Below:
FIND VSSKILL %SKILL = VSSKILL.NAME %EMP.NUM = VSEMP.NO; FIND PLANT
%SALARY = (SAL.YTD - SAL.DED) %ID = EMP.NO IF EMP.NO = %EMP.NUM;
DISPLAY VSPLANT VSPLANT.ID VSEMP.NAME %SKILL %SALARY IF VSEMP.NO = %ID;

VSPLANT.ID    VSEMP.NAME           SKILL                      SALARY
10100        WILLIAM AMES         SALESPERSON              55,000.00
10100        WILLIAM AMES                                  44,600.00
10100        PHYLLIS LOCKMEYER    SALESPERSON              50,800.00
10100        PHYLLIS LOCKMEYER                             41,600.00
10100        MARY ANN THOMAS      SECRETARY                14,230.00
20150        WILMA FORD           SECRETARY                16,820.00
20150        WILMA FORD                                    14,340.00
20150        CHARLES SALTER       ACCOUNTANT               33,000.00
20150        CHARLES SALTER                                26,860.00
20150        CHARLES SALTER                                21,980.00
20150        PETER ZATKIN         ADMINISTRATOR            46,420.00
20150        PETER ZATKIN                                  42,200.00
20150        PETER ZATKIN                                  37,200.00
20150        SUSAN WARE           ENGINEER                 34,000.00
20150        SUSAN WARE                                    26,800.00
   TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
 (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 8-16 FIND Statement that Accesses an IMS (DL/I) Database and Two VSAM Files

Notice that the output is the plant ID (VSPLANT.ID) and employee name (VSEMP.NAME) from the VSAM VSPLANT file, the employee's job position (VSSKILL.NAME) from the VSAM VSSKILL file, and the employee's net salary (%SALARY) from the IMS (DL/I) PLANT database.

Accessing VSAM Files and DB2 Tables

The following are examples that use both a VSAM file and a DB2 table. Figure 8-17 uses the FIND and DISPLAY commands, as well as a conditional phrase, to access the VSAM VSPLANT file and the DB2 SKILLS table.

VISION:Inquiry searches for the first occurrence of 'SECRETARY' and assigns that value to the temporary field %SKILL. The related EMP_NO is assigned to %EMP.NUM; control then passes to the DISPLAY command. Figure 8-17 displays the output.

Example: "List the employees that have a secretarial position. Use the VSAM VSPLANT file and the DB2 SKILLS table."

```
PAGE:       TRANSACTION: IQIO
                         Enter Inquiry Below:
FIND SKILLS %SKILL = SKILL_NAME %EMP.NUM = EMP_NO IF
SKILL_NAME = 'SECRETARY' ;
DISPLAY VSPLANT VSPLANT.ID VSEMP.NO VSEMP.NAME %SKILL IF VSEMP.NO
 = %EMP.NUM;;
VSPLANT.ID  VSEMP.NO   VSEMP.NAME                 SKILL
10100         10,105   MARY ANN THOMAS            SECRETARY
30200         30,207   JANE LOWELL                SECRETARY
50300         50,323   VICKY WARD                 SECRETARY
70500         70,522   AGNES COVINGTON            SECRETARY
20150         21,116   WILMA FORD                 SECRETARY
60200         60,258   KAREN REDFERN              SECRETARY
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (12,12 USER DB CALLS,ROOTS)
  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
 (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 8-17 FIND Statement that Accesses a VSAM Non-Hierarchical File and a DB2 Table

The next example illustrates how VISION:Inquiry can access fields from three files. The first FIND statement uses the VSAM file, VSSKILL. The second FIND statement uses the DB2 SALARIES table; then a DISPLAY command is used to access the VSAM file, VSPLANT.

Example: “List the employees, their jobs and net salaries by plant.”

```
PAGE:       TRANSACTION: IQIO
                         Enter Inquiry Below:
FIND VSSKILL %SKILL = VSSKILL.NAME %EMP.NUM = VSEMP.NO;
FIND SALARIES %SALARY = (SAL_YTD - SAL_DED) %ID = EMP_NO IF EMP_NO
= %EMP.NUM;
DISPLAY VSPLANT VSPLANT.ID VSEMP.NAME %SKILL %SALARY IF VSEMP.NO = %ID;;
VSPLANT.ID    VSEMP.NAME            SKILL                      SALARY
10100         WILLIAM AMES          SALESPERSON             55,000.00
10100         WILLIAM AMES                                  44,600.00
10100         PHYLLIS LOCKMEYER     SALESPERSON             50,800.00
10100         PHYLLIS LOCKMEYER                             41,600.00
10100         MARY ANN THOMAS       SECRETARY               14,230.00
20150         WILMA FORD            SECRETARY               16,820.00
20150         WILMA FORD                                    14,340.00
20150         CHARLES SALTER        ACCOUNTANT              33,000.00
20150         CHARLES SALTER                                26,860.00
20150         CHARLES SALTER                                21,980.00
20150         PETER ZATKIN          ADMINISTRATOR           46,420.00
20150         PETER ZATKIN                                  42,200.00
20150         PETER ZATKIN                                  37,200.00
20150         SUSAN WARE            ENGINEER                34,000.00
20150         SUSAN WARE                                    26,800.00
  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
 (SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 8-18 FIND Statement that Accesses a DB2 Table and Two VSAM Non-Hierarchical Files

Notice that the output is the plant ID (VSPLANT.ID) and employee name (VSEMP.NAME) from the VSAM VSSKILL file, the employee’s job position (VSSKILL.NAME) from the VSAM VSSKILL file, and the employee’s net salary (%SALARY) from the DB2 SALARIES table.

Accessing DB2 Tables and an IMS (DL/I) Database

VISION:Inquiry allows you to access a combination of databases. The concept is the same as previously described in this chapter. The following example uses a DB2 table, SALARIES, and an IMS (DL/I) database, PLANT.

This example creates a report of all male employees who have earned at least a Bachelor of Arts degree and who earned $50,000 or more. Notice that a conditional selection statement occurs in both the FIND and DISPLAY statements. Figure 8-19 displays the report.

Example: “List the male employees who have at least a BA degree and who earn $50,000 or more.”

```
PAGE:       TRANCODE: II   INQUIRY :
FIND PLANT %PLANT = PLANT.ID %DEGREE = ED.DEGREE IF ED.DEGREE GE
'BA' AND EMP.SEX = 'M';  DISPLAY SALARIES %PLANT EMP_NAME
%DEGREE SAL_YTD IF SAL_YTD GE 50000 AND EMP_SEX = 'M' AND
PLANT_ID = %PLANT;;

PLANT  EMP_NAME                   DEGREE       SAL_YTD
10100  WILLIAM AMES               BA        52,000.00
       WILLIAM AMES                         64,000.00
20150  PETER ZATKIN               BA        50,000.00
       PETER ZATKIN                         56,000.00
       PETER ZATKIN               BA        50,000.00
       PETER ZATKIN                         56,000.00
       PETER ZATKIN               MA        50,000.00
       PETER ZATKIN                         56,000.00
30200  FREDERICH GRAY             HS        59,000.00
       MITCHELL J HOOPS                     76,000.00
       MITCHELL J HOOPS                     92,000.00
       MITCHELL J HOOPS                     98,000.00
       FREDERICH GRAY             BA        59,000.00
       MITCHELL J HOOPS                     76,000.00
       MITCHELL J HOOPS                     92,000.00
```

Figure 8-19 FIND Statement that Accesses a DB2 Table and an IMS (DL/I) Database

Using FIND with the LIMIT Command

The following figure shows how more than one LIMIT command may be used with a FIND and a DISPLAY command.

Example: “List the employees for each plant, but limit the output of employees and plants.”

```
PAGE:       TRANCODE: II       INQUIRY:
FIND PLANT %PLANT.ID = PLANT.ID LIMIT 3;
DISPLAY PLANT %PLANT.ID EMP.NAME IF PLANT.ID = %PLANT.ID LIMIT 2;;

PLANT.ID  EMP.NAME
10100     WILLIAM AMES
          PHYLLIS LOCKMEYER
20150     WILMA FORD
          CHARLES SALTER
30200     JOHN HENRY CRANE
          FREDERICH GRAY
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (19,7 USER DB CALLS,ROOTS)
```

Figure 8-20 LIMIT Command Used with a FIND and a DISPLAY Command

Figure 8-20 illustrates the multiple usage of the LIMIT command. Applied to the FIND statement, the LIMIT command produces the first three occurrences of the PLANT.ID that match the PLANT.ID of the DISPLAY statement. Applied to the DISPLAY statement, the LIMIT command produces the first two occurrences of EMP.NAME.

If you displayed the PLANT.ID instead of %PLANT.ID, you would get all the employees from the first three plants. The LIMIT command is applied to the PLANT.ID field because it is a higher level segment than EMP.NAME; the LIMIT command on %PLANT.ID is ignored.

FIND Command Summary

FIND (I or INTER)	Command indicates that one or more databases are accessed by the inquiry.
database name	Name of database to be accessed by this FIND statement.
%temporary field = database field	Contents of database field on the right are assigned to the temporary field on the left. There may be a list of such assignments in a FIND statement. The temporary fields and their contents are transferred to the related DISPLAY statement.
%temporary field = database field to be matched	Required assignment in a FIND statement, used to match related information in two databases. This assigns current value of database field to temporary field for transfer to next FIND or DISPLAY statement. It is matched against the contents of a database field in the next database accessed.
%temporary field = arithmetic expression	At least one field referenced in the arithmetic expression must be from the database accessed by the FIND statement. The result field cannot be tested in a conditional selection phrase in the FIND statement.
LIMIT	Command limits the number of items produced from the FIND database.
IF logical expression	Data is transferred to the next FIND or the DISPLAY statement only if the conditions in the logical expression are met. The logical expression may contain arithmetic, relational, and logical terms. The phrase may test fields from the database accessed in that statement. The arithmetic expression must contain at least one field from the database accessed by the FIND statement.
;	The semicolon terminates the FIND statement.
DISPLAY	Command to display fields both from the database accessed by this statement and transferred from the FIND statements.
database name	The name of database to be accessed by this statement.
title line	One or two character constants (up to 70 characters) to be displayed at the top of each page of output.

%temporary field(s)	Field(s) transferred from FIND statements for display or sorting. May be used as part of an arithmetic calculation.
field name(s)	Field(s) from the database accessed in this statement for display, sorting, or summary.
arithmetic calculation	May include temporary fields, database fields, and constants. If result field is to be tested in a conditional phrase, the calculation must contain at least one field from this database.
SORT	Command sorts fields from FIND or DISPLAY statements.
TOTAL, COUNT AVERAGE	Summary commands can be used to summarize database fields. They cannot be used to summarize temporary fields transferred from FIND statements.
LIMIT	Command limits the number of items displayed from the DISPLAY database.
IF database field = %temporary field	Relational test matches fields from previous and current databases. Temporary field holds value of match field transferred from previous FIND statement. Database field holds current value of match field from database accessed by this statement. The statement in which this conditional phrase appears is not executed unless the two fields match. The name of the database field must appear on the left side of the equation, and the temporary field on the right side. Phrase may test for more than one match field. IF must be present in the DISPLAY statement and in all FIND statements except the first.
IF logical expression	Logical expression may contain logical, relational, and arithmetic terms. If result field is to be tested, the calculation must contain at least one field from the database accessed by this statement. The relational test to match fields must be included as part of the logical expression.
;	The semicolon terminates the DISPLAY statement. Two semicolons are required in systems operating in batch mode.

The conditional selection phrase in a FIND statement must be satisfied before control passes to the next FIND or DISPLAY statement.

The FIND and DISPLAY statements in an inquiry may both access the same database. Up to 16 databases may be accessed in an inquiry with up to 15 FIND statements and one DISPLAY statement.

Databases may be matched on more than one field. The databases may match on the same or different fields, but each database must be matched to the next one accessed.

If more than one FIND statement is used, all FIND statements except the first one must contain an assignment statement to transfer a value of the match field to the next statement. The first FIND statement needs only the assignment statement. The DISPLAY statement needs only the relational test.

Chapter 9

Using Directory Commands - Storing Inquiries and Functions

The directory was described in Chapter 1, "Introducing VISION:Inquiry" as a library that holds many kinds of information relating to VISION:Inquiry. One of its purposes is to hold stored inquiries and functions.

- **Stored inquiries** are those that you or your system administrator write and save in the directory for future use. You can open and execute a stored inquiry whenever you need that output or report.
- **Stored functions** are mathematical calculations that you or your system administrator write and save in the directory for future use. They are later integrated into inquiries to perform the calculations needed to produce a particular output or report. No functions can be stored for native SQL syntax inquiries.

The process of saving inquiries and functions in the directory has several advantages:

- The same inquiry can be used many times by the same or different users.
- The same function can be called by many different inquiries. This saves considerable programming and computer time, ensures accuracy of VISION:Inquiry statement(s), and provides standardization of output and reports.
- The stored inquiries, except SQL syntax stored queries, can also be processed by requests from the PC-based Intraccess. The output result is then downloaded to the PC. The content of the stored query determines whether data or reports are downloaded. If the stored query contains a UDO query or summary commands such as TOTAL, a report is produced and sent to the PC; otherwise, data only is sent to the PC. To find out about the characteristics of data and report, see the Intraccess documentation.

  Intraccess does not support SQL syntax stored queries in this release.

In some instances, you may only use stored inquiries and may not prepare your own statements. With stored inquiries, you access information from databases and produce reports without having to learn how to write VISION:Inquiry statements.

This chapter explains how to store inquiries and functions in the directory, how to look at the stored items, and how to delete items that are no longer needed. The next chapter explains how to use stored inquiries and functions.

DEFINE Command

The DEFINE command stores the described inquiry or arithmetic function in the directory under an assigned name.

Storing Inquiries with the DEFINE Command

You use the DEFINE command to store inquiries with and without SQL syntax.

Using DEFINE DIRECTORY INQUIRY to store Inquiries without SQL

Example: Storing an inquiry without native SQL syntax of DB2 option.

`DEFINE`	`DIRECTORY`	`INQUIRY`	`PLANT`	`'ROSTER`	`[,`	`comment]'`	`AS`
command to store inquiry	store item here	type of item to store	related to this database	name to be assigned to stored inquiry	comma separates inquiry name and comment	comment up to 60 bytes	noise word

```
DISPLAY PLANT 'EMPLOYEE INFORMATION FOR PLANTS WITH LOCATIONS ENDING WITH 00'
PLANT .ID EMP.NO EMP.NAME IF PLANT.ID LIKE '%00';
```

inquiry to be stored under the name ROSTER

Using DEFINE DIRECTORY INQUIRY to store Inquiries with SQL

Example: Storing an inquiry with native SQL syntax of DB2 option.

`DEFINE`	`DIRECTORY`	`INQUIRY`	`'ROSTER`	`[,`	`comment]'`
command to store inquiry	store item here	type of item to store	name to be assigned to stored inquiry	comma separates inquiry name and comment	comment up to 60 bytes

```
DISPLAY EXECSQL SQL SELECT statement ENDEXEC;
```

native SQL syntax inquiry to be stored under the name ROSTER

The DEFINE statement names the inquiry ROSTER and stores it in the directory under that name. Later, when you want to use the inquiry, you have only to reference it by name.

Note that in the first example, storing an inquiry without native SQL syntax, the name of the database appears twice—once in the DEFINE portion of the statement to tell VISION:Inquiry to which database it is related, and again in the inquiry itself to tell VISION:Inquiry which database has the data needed by the inquiry.

Example: “Prepare a stored inquiry that lists the plants, the names of their employees, and the related employee numbers. For plants with locations ending with 00. Also, display a title line at the top of each page.”

```
PAGE:      TRANCODE: II        INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'ROSTER' AS
DISPLAY PLANT ‘EMPLOYEE INFORMATION FOR PLANTS WITH LOCATIONS ENDING WITH 00’
PLANT.ID EMP.NO EMP.NAME IF PLANT.ID LIKE ‘%00’;


IXX0254 DEFINE OF 'ROSTER' IS ADDED.
```

Figure 9-1 DEFINE Command

This example shows the message returned by VISION:Inquiry when the inquiry has been successfully stored. The message tells you that the inquiry named ROSTER has been added to the list of stored inquiries indicated in the DEFINE command.

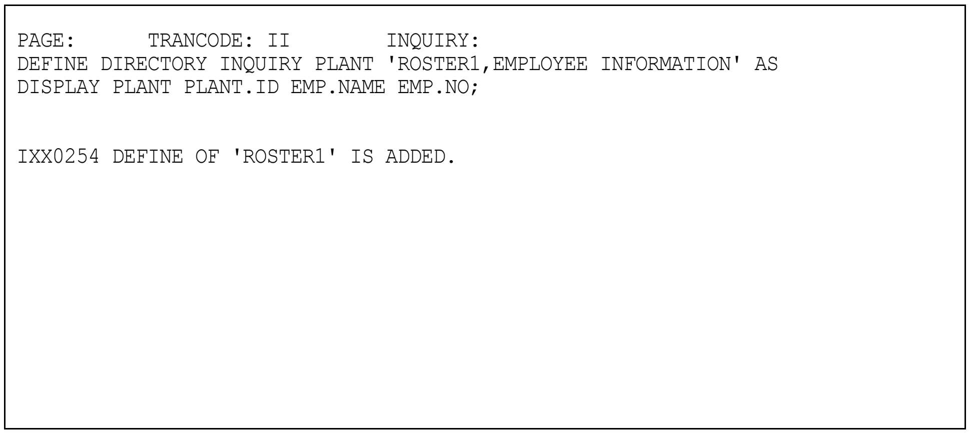
DEFINE Command with a Comment

Figure 9-2 shows that you can add a comment of up to 60 bytes to your stored inquiry.

- Your comment will be displayed when the PDIC command is used. The PDIC command is discussed later in this chapter, beginning with Display Stored Inquiry Names and Their Associated Comments on page 9-25.
- Your comment will also be displayed on the Text Editor screen when the EDITSQ command is used (see Chapter 10, “Using Stored Inquiries and Functions”).
- Your comment also appears when you use the Intraccess option (see the Intraccess documentation for more information).

The comment has no effect on the processing of the inquiry. Semicolons and single quotation marks are not allowed in the comment text.

Example: “Prepare a stored inquiry that lists the plants, the names of their employees, and the related employee numbers. Document the stored inquiry with a comment.”



```
PAGE:      TRANCODE: II        INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'ROSTER1,EMPLOYEE INFORMATION' AS
DISPLAY PLANT PLANT.ID EMP.NAME EMP.NO;


IXX0254 DEFINE OF 'ROSTER1' IS ADDED.
```

Figure 9-2   DEFINE Command with a Comment

The example is identical to the previous example except that a comment has been added. VISION:Inquiry saves the inquiry under the name ROSTER with EMPLOYEE INFORMATION as the comment.

The comment will be displayed when using the PDICW command or on the comment line of the Text Editor screen when the EDITSQ command is used to edit the stored inquiry ROSTER1.

Defining Inquiries with Substitutable Values

Note: This section is not applicable to the native SQL syntax facility of DB2 option.

A very important feature of stored inquiries is that they can be designed so that the values in the conditional selection phrase can be changed each time the inquiry is run.

This means that each time the inquiry is executed, it can access a different part of the database and output different information. The same inquiry can be used to report information on plant 10100 one time and plant 50300 the next time, or salary information for 1995 one time and 1994 the next. In each case, the inquiry is written and stored only once.

The conditional values needed for a particular run are indicated when the inquiry is executed. The execution and substitution process is illustrated in the next chapter. Writing inquiries with substitutable values is explained next.

Example: "Relate salaries and degrees and vary the conditions each time the inquiry is run."

```
PAGE:       TRANCODE: II      INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'DEGREE.SALARY' AS
DISPLAY PLANT EMP.NO EMP.NAME ED.DEGREE SAL.YTD
IF SAL.YEAR = 95 AND ED.DEGREE = %1 AND SAL.YTD LE %2;

IXX0255 DEFINE OF 'DEGREE.SALARY' IS ADDED.
```

Figure 9-3 Defining a Stored Inquiry with Substitutable Values

The inquiry DEGREE.SALARY, which is being stored by the DEFINE statement above, has three relational expressions. The first limits the output to 1995. The next two permit test values to be substituted at the time the inquiry is executed.

Take a closer look at the relational expressions with substitutable values:

```
IF SAL.YEAR = 95 AND      ED.DEGREE = %1      AND      SAL.YTD LE %2
                      |                    |        |                 |
                      |    substitute      |        |   substitute    |
                      |    value #1        |        |   value #2      |
```

The second relational expression 'ED.DEGREE = %1' tells VISION:Inquiry that the data selected for output is limited by ED.DEGREE, but that the particular value of ED.DEGREE is indicated later. The last relational expression, 'SAL.YTD LE %2', indicates that the output is also limited by SAL.YTD, but that the test value of SAL.YTD is given later.

The symbols for the substitutable values, %1 and %2, are assigned by their order of appearance in the statement. If another relational expression testing EMP.SEX is added to the conditional phrase above, its symbol is %3.

DEGREE.SALARY is stored in the directory. You can use DEGREE.SALARY by referring to its name and executing it by listing the particular values of ED.DEGREE and SAL.YTD needed for that run. This process is illustrated in the next chapter.

Although all VISION:Inquiry commands can be included in a stored inquiry, LIMIT and OUTPUT are ignored in processing. Chapter 10, "Using Stored Inquiries and Functions" shows how to use LIMIT and OUTPUT when working with stored inquiries.

Figure 9-4 shows a more complex stored inquiry containing several commands.

This example illustrates storing an inquiry that has several commands and three substitutable values in its conditional phrase.

```
PAGE:       TRANCODE: II       INQUIRY:
 DEFINE DIRECTORY INQUIR1Y PLANT 'SALARY.REPORT' AS
 DISPLAY PLANT PLANT.ID EMP.NAME SAL.YTD
 COUNT EMP.NAME TOTAL SAL.YTD IF SAL.YEAR = 95
 AND ED.DEGREE = %1 AND (PLANT.ID = %2 OR PLANT.ID = %3);

 IXX0255 DEFINE OF 'SALARY.REPORT' IS ADDED
```

Figure 9-4 Stored Inquiry with Substitutable Values and Several VISION:Inquiry Commands

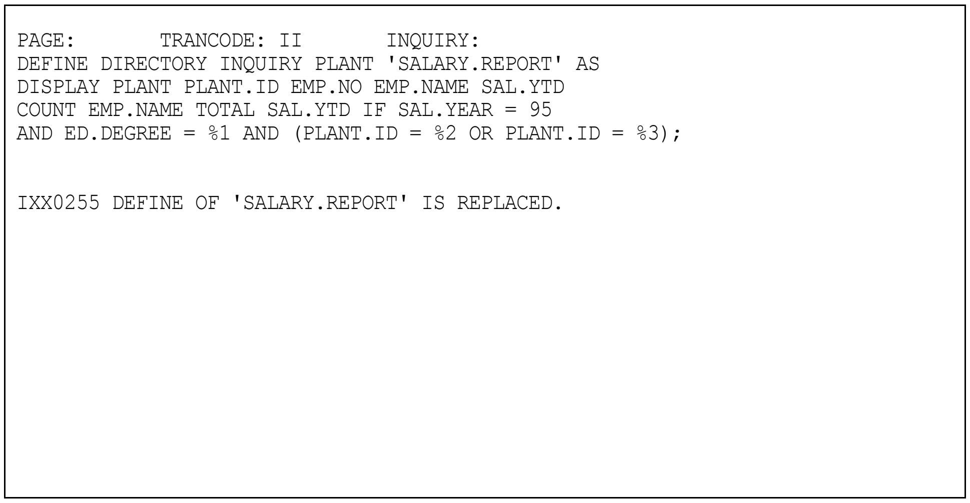
When you are ready to execute SALARY.REPORT, refer to it by name and give the three values to be substituted. The inquiry is then executed, and the indicated summaries are calculated. This process is illustrated in the next chapter.

Replacing a Stored Inquiry

Suppose that after you define a stored inquiry, you need to change part of it.

For example, you decide to add EMP.NO to the inquiry above. In that case, you correct the DEFINE statement and execute it again. VISION:Inquiry replaces the old version of SALARY.REPORT with the new one and returns the message shown below.

You can also change a stored inquiry using the Text Editor (see Chapter 10, "Using Stored Inquiries and Functions" for additional information about the Text Editor).



```
PAGE:       TRANCODE: II       INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'SALARY.REPORT' AS
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME SAL.YTD
COUNT EMP.NAME TOTAL SAL.YTD IF SAL.YEAR = 95
AND ED.DEGREE = %1 AND (PLANT.ID = %2 OR PLANT.ID = %3);


IXX0255 DEFINE OF 'SALARY.REPORT' IS REPLACED.
```

Figure 9-5 Replacing a Stored Inquiry

This points out the necessity of taking care when assigning names to inquiries. If you accidentally assign an old name to a new inquiry, the old inquiry is replaced.

Storing Functions

Note: This section is not applicable to the native SQL syntax facility of DB2 option.

You use the DEFINE command to name and store an arithmetic expression in the directory. This expression can then be called and used by one or more inquiries.

These arithmetic expressions are called functions. Storing functions is particularly useful when the function is complex and involves many operations, or when the same function is used by many inquiries. Stored functions can be called by stored inquiries or by inquiries as they are keyed in at the terminal.

`DEFINE`	`DIRECTORY`	`FUNCTION`	`PLANT`	`'MON.SAL'`	`AS`
command to store	store item here	item type to store	related to this database	name to be assigned to function	noise word

```
      %MONTH.SAL = (SAL.YTD - SAL.DED) / 12 ;
```

function to be named MON.SAL
and stored in the directory

Note: The function is not a complete inquiry statement and cannot be executed by itself; it must be called by an inquiry. This process is illustrated in the next chapter.

Using the DEFINE DIRECTORY FUNCTION to Store a Function

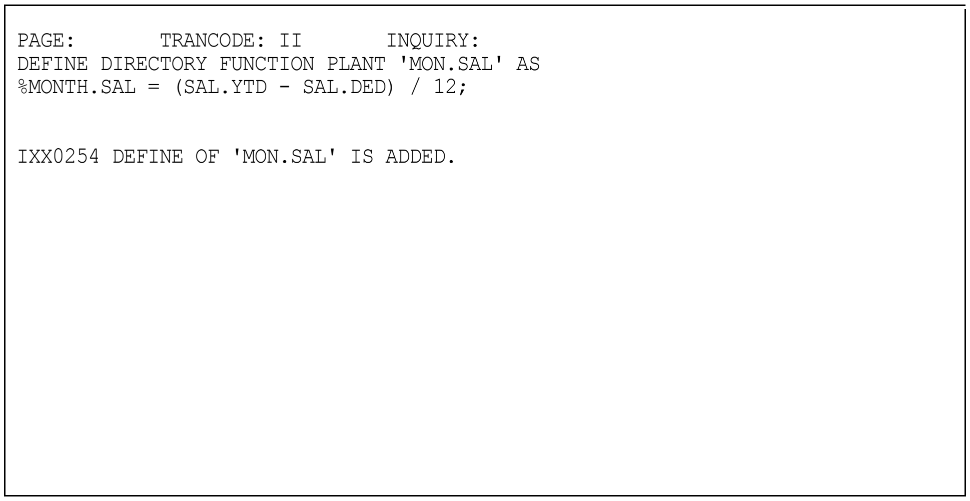
The DEFINE command tells VISION:Inquiry that the arithmetic expression shown on the second line is to be named MON.SAL and stored in the list of functions in the directory.

- The name of the function in the DEFINE statement must be enclosed in single quotation marks. It tells VISION:Inquiry what name to assign to the function as it is stored.
- The name cannot be a field name, command, or stored inquiry name; an error message will be issued if you try to do so.

A function is an arithmetic expression that follows the arithmetic processing rules explained in Chapter 7, "Assignment Statement and Arithmetic Processing". %MONTH.SAL is the name of the temporary field that holds the result of the arithmetic calculation. When the value of the temporary field is displayed by an inquiry statement, MONTH.SAL appears as the column heading.

The names of the function (MON.SAL) and the temporary field (MONTH.SAL) may be the same if you wish. It may be easier to relate them to each other if they match.

Example: "Write and store a function that calculates the monthly salary."



```
PAGE:       TRANCODE: II       INQUIRY:
DEFINE DIRECTORY FUNCTION PLANT 'MON.SAL' AS
%MONTH.SAL = (SAL.YTD - SAL.DED) / 12;


IXX0254 DEFINE OF 'MON.SAL' IS ADDED.
```

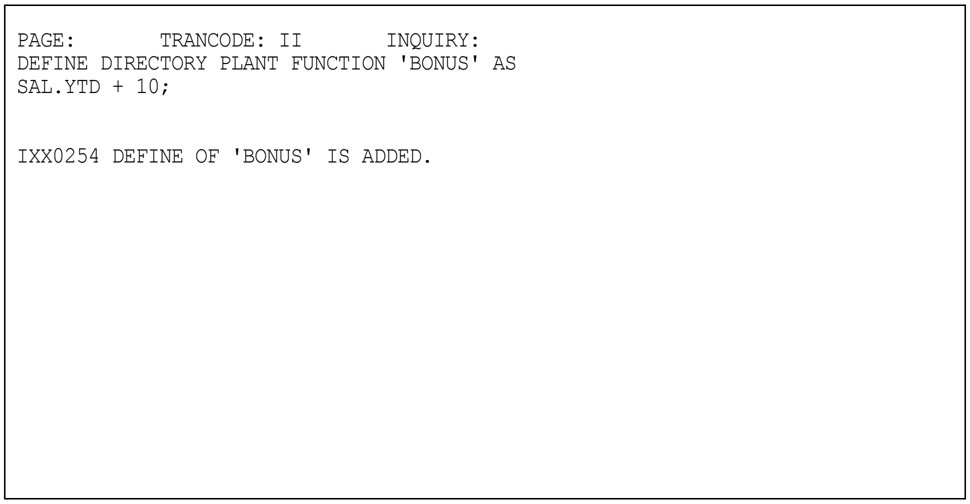
Figure 9-6   Storing a Function

This example shows the message that VISION:Inquiry returns when the function is successfully stored.

Storing a Function with No Column Heading

Figure 9-7 illustrates a DEFINE statement that stores a function named BONUS. In this function, the result of the arithmetic calculation is not assigned to a temporary field. When the results of the function are displayed by an inquiry statement, a period appears as the column heading.

Example: "Calculate a bonus of salary plus $10."



```
PAGE:       TRANCODE: II       INQUIRY:
DEFINE DIRECTORY PLANT FUNCTION 'BONUS' AS
SAL.YTD + 10;


IXX0254 DEFINE OF 'BONUS' IS ADDED.
```

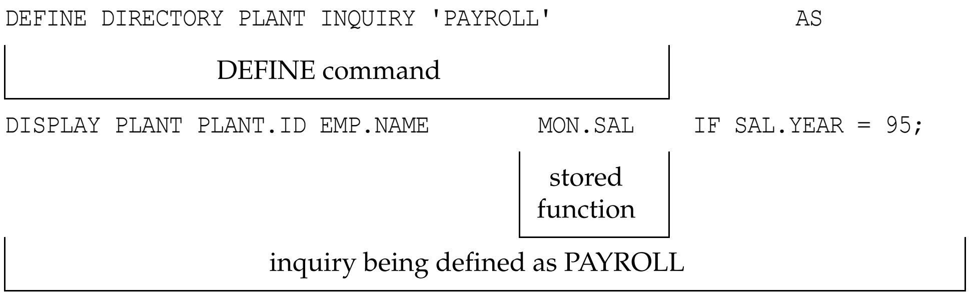
Figure 9-7 Storing a Function with No Column Heading

The function named BONUS has been added to the functions stored in the directory. Stored functions in which arithmetic results are not assigned to temporary fields may not begin with parentheses.

Storing Inquiries that Call Stored Functions

Note: This section is not applicable to the native SQL syntax facility of DB2 option.

This DEFINE command adds PAYROLL to the inquiries related to the PLANT database.

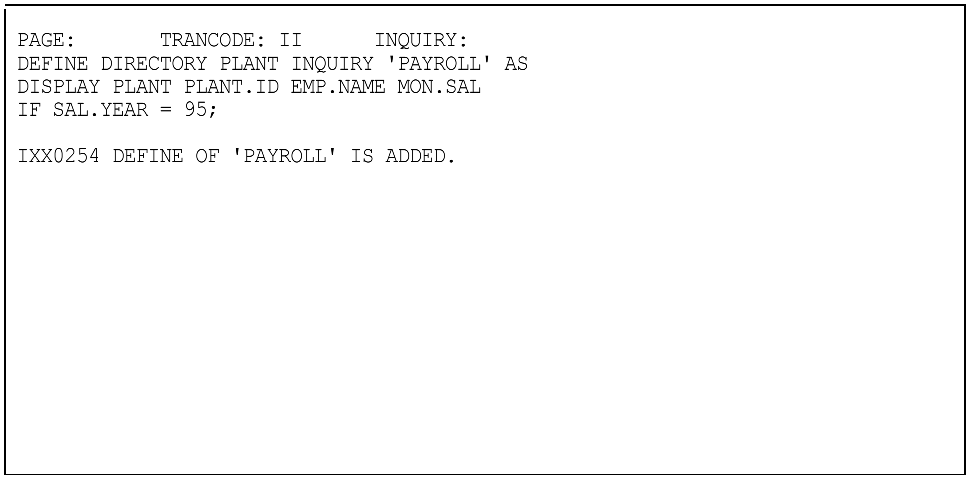


```
DEFINE DIRECTORY PLANT INQUIRY 'PAYROLL'                AS
DISPLAY PLANT PLANT.ID EMP.NAME        MON.SAL    IF SAL.YEAR = 95;
```

PAYROLL calls the stored function named MON.SAL. This is done by listing the name of the function as one of the items to be displayed on the report. VISION:Inquiry automatically substitutes the terms in the function%MON.SAL = (SAL.YTD-SAL.DED)/12 for the name MON.SAL when the inquiry is stored.

When the inquiry is executed, the steps in the calculation are carried out and the results assigned to the temporary field named MONTH.SAL. This process is illustrated in the next chapter.

Example: "Produce a report that shows monthly salary earned by each employee in 1995. There is a stored function that performs this calculation."



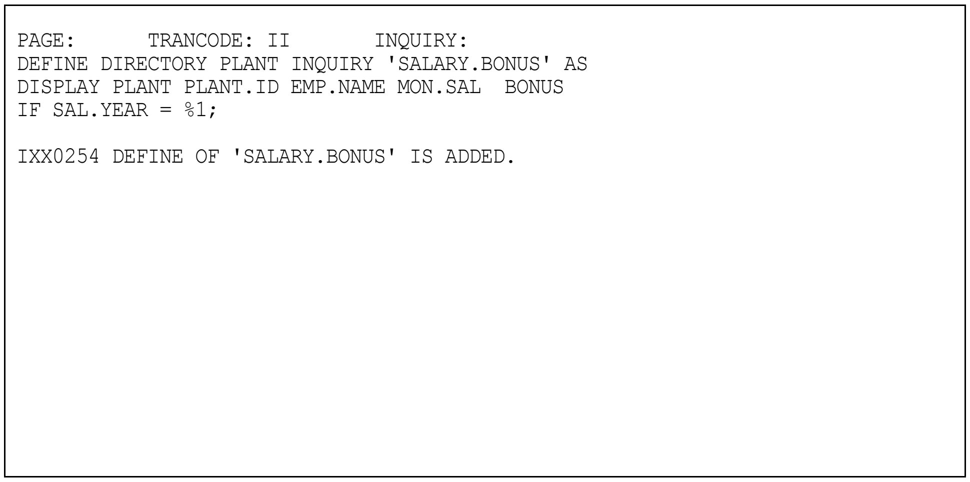
```
PAGE:       TRANCODE: II      INQUIRY:
DEFINE DIRECTORY PLANT INQUIRY 'PAYROLL' AS
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL
IF SAL.YEAR = 95;

IXX0254 DEFINE OF 'PAYROLL' IS ADDED.
```

Figure 9-8 Storing an Inquiry that Calls a Stored Function

This DEFINE command adds PAYROLL to the stored inquiries related to the PLANT database. Note that the name of this stored function (MON.SAL) is not enclosed in quotation marks.

Storing an Inquiry with Two Stored Functions and a Substitutable Value

Example: “Produce a report showing the monthly salary and bonus for each employee. Create the inquiry so that it can run for different years. There are stored functions that perform both calculations.”



```
PAGE:      TRANCODE: II       INQUIRY:
DEFINE DIRECTORY PLANT INQUIRY 'SALARY.BONUS' AS
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL  BONUS
IF SAL.YEAR = %1;

IXX0254 DEFINE OF 'SALARY.BONUS' IS ADDED.
```

Figure 9-9 Stored Inquiry that Calls Two Stored Functions and Has a Substitutable Value

This DEFINE command adds SALARY.BONUS to the inquiries stored for the PLANT database. Included are the names of two stored functions in this inquiry, and the test value of the conditional phrase is a substitutable value.

VISION:Inquiry substitutes the terms of both functions for their names as it stores the inquiry. The substitutable value is indicated each time the inquiry is executed.

If you wish to change the terms in a function called by a stored inquiry, you must first redefine the function and then redefine the stored inquiry.

This example illustrates the versatility of the stored inquiry. An inquiry that performs a variety of operations and utilizes several stored functions can be stored and then used additional times; each time, it can be executed against different groups of data.

Calling an Undefined Function

A function must be defined before it can be called by an inquiry. The following example shows the message returned by VISION:Inquiry if you call a function that has not been defined, or if you misspell the name of the function.

```
 PAGE:       TRANCODE: II       INQUIRY:
 DEFINE DIRECTORY PLANT INQUIRY 'REPORT' AS
 DISPLAY PLANT PLANT.ID EMP.NAME SAL.MONTH
 IF SAL.YEAR = %1;

 IXX0106 DATA NAME 'SAL.MONTH' NOT FOUND IN DIRECTORY.
          THE NAME SHOWN ABOVE IS INTERPRETED TO BE A FIELD
          OR DATABASE NAME, OR A STORED INQUIRY OR FUNCTION
          NAME. HOWEVER, IT CANNOT BE LOCATED IN THE DICTIONARY
          ASSOCIATED WITH YOUR TERMINAL.
          PLEASE CHECK THE SPELLING.
*
*
*
*
*
*
*
 IXX0199 INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

Figure 9-10 Calling an Undefined Function

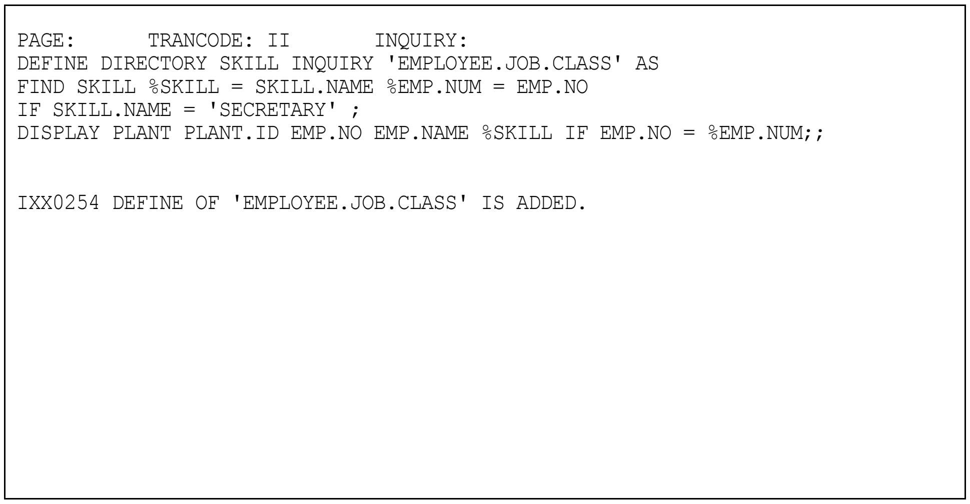
In the example above, the name of the function has been misspelled, and VISION:Inquiry tells you that it cannot find SAL.MONTH in the directory. The inquiry named REPORT is not stored.

Storing Inquiries with the FIND Command

Note: This section is not applicable to the native SQL syntax facility of DB2 option.

Inquiries accessing more than one database with the FIND command can also be stored on the directory. This is illustrated in Figure 9-11.

Example: “Store the inquiry that displays all employees who are secretaries.”



```
PAGE:      TRANCODE: II       INQUIRY:
DEFINE DIRECTORY SKILL INQUIRY 'EMPLOYEE.JOB.CLASS' AS
FIND SKILL %SKILL = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'SECRETARY' ;
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME %SKILL IF EMP.NO = %EMP.NUM;;


IXX0254 DEFINE OF 'EMPLOYEE.JOB.CLASS' IS ADDED.
```

Figure 9-11   Storing an Inquiry Containing a FIND Command

Note that the database listed in the first FIND command is the one that is referenced in the DEFINE command.

DEFINE Command Summary - Inquiries Without SQL

This section summarizes DEFINE when used without the native SQL syntax facility of DB2 option.

DEFINE DIRECTORY INQUIRY or DDI	Stores an inquiry in the directory. The DEFINE DIRECTORY INQUIRY or DDI command must be placed at the beginning of the inquiry when storing the inquiry.

or

DEFINE DIRECTORY FUNCTION or DDF	Stores a function in the directory.
database name	Relates the stored item to this database.
'name, comment'	Name indicates the name being assigned to the inquiry or function. It should be enclosed in quotation marks; it may not be the name of a field, command, or other stored inquiry function. The name can be up to 32 characters in length, and the first character must be alphabetic. Comment indicates an optional comment that is stored along with the inquiry and can be up to 60 bytes. The comment cannot be used for stored functions.

The inquiry or function to be stored should be listed next.

The conditional values in the inquiry being stored do not have to be explicitly indicated. They can be replaced with substitutable values in the stored inquiry; the actual values are indicated during execution. The substitutable values (%1,%2,%3, etc.) are assigned from left to right, and are replaced during execution from left to right.

The DEFINE statement should reference the database listed in the first FIND statement when storing an inquiry that accesses more than one database.

All VISION:Inquiry commands can be included in a stored inquiry. The LIMIT and OUTPUT commands will be ignored during execution if they are included in a stored inquiry, but will be displayed when editing the stored inquiry using the EDITSQ command. See Chapter 10, "Using Stored Inquiries and Functions" for more on this, and for how to add the LIMIT and OUTPUT commands when executing a stored inquiry.

The inquiry to be stored can also contain one or two title lines.

Stored functions can be integrated into new inquiries and stored inquiries.

Stored functions in which arithmetic results are not assigned to temporary fields may not begin with parentheses.

To change the terms in a stored function that is part of a stored inquiry, you need to first redefine the function and then redefine the calling inquiry.

DEFINE Command Summary - Inquiries With SQL

This section summarizes DEFINE when using the native SQL syntax facility of DB2 option.

DEFINE DIRECTORY INQUIRY	Stores an inquiry in the directory. The DEFINE DIRECTORY INQUIRY or DDI command must be placed at the beginning of the inquiry when storing the inquiry.
or	
DDI	
'name, comment'	Name indicates the name being assigned to the inquiry. It should be enclosed in quotation marks. The name can be up to 32 characters in length, and the first must be alphabetical. Comment indicates an optional comment that is stored along with the inquiry and can be up to 60 bytes.

The native SQL syntax inquiry to be stored should be listed next.

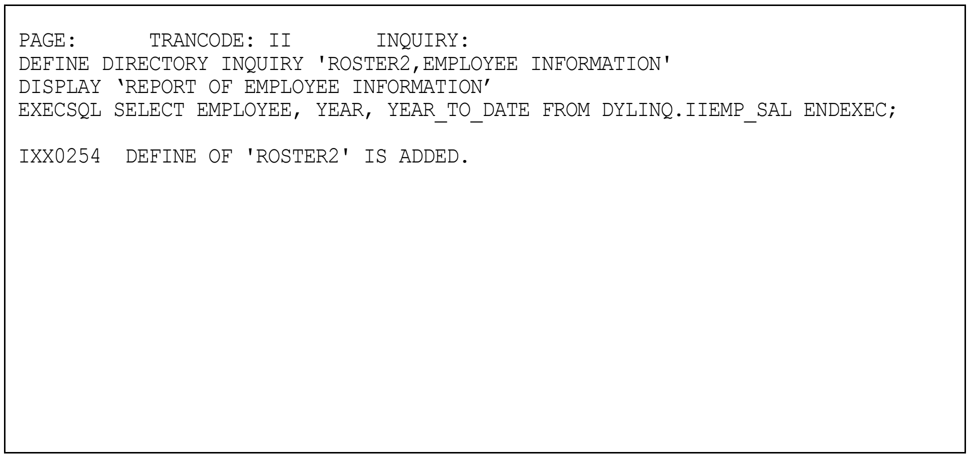
The EXECSQL keyword in the inquiry relates the stored inquiry to the native SQL syntax facility. A unique name must be assigned to each native SQL syntax inquiry at the time of definition (storing) the inquiry. Otherwise, the stored inquiry will replace the existing inquiry in the directory with the same name.

The LIMIT and OUTPUT commands will be ignored during execution if they are included in a stored inquiry, but will be displayed when editing the stored inquiry using the EDITSQ command.

The inquiry to be stored can also contain one or two title lines.

See Chapter 10, "Using Stored Inquiries and Functions" for more on using EDITSQ, and for how to add the LIMIT and OUTPUT commands when executing a stored inquiry.

Example: "Prepare a native SQL syntax stored inquiry that lists the employee information from the employee DB2 table. The output will also display a title line at the top of each page."



```
PAGE:      TRANCODE: II       INQUIRY:
DEFINE DIRECTORY INQUIRY 'ROSTER2,EMPLOYEE INFORMATION'
DISPLAY ‘REPORT OF EMPLOYEE INFORMATION’
EXECSQL SELECT EMPLOYEE, YEAR, YEAR_TO_DATE FROM DYLINQ.IIEMP_SAL ENDEXEC;

IXX0254  DEFINE OF 'ROSTER2' IS ADDED.
```

Figure 9-12 A Native SQL Syntax Stored Inquiry Using the DEFINE DIRECTORY Command

DISPLAY DIRECTORY Command

Use the DISPLAY DIRECTORY command to tell VISION:Inquiry to list one or more items stored in the directory. Individual inquiries or functions may be displayed, or all the stored items relating to a database may be shown.

`DISPLAY DIRECTORY`	`INQUIRY`	`PLANT`	`'DEGREE.SALARY'`
display items from the directory	item type to display	items related to this database	name of item to be displayed

This statement tells VISION:Inquiry to list the inquiry named DEGREE.SALARY that has been stored in the directory. Note that the name of the inquiry is enclosed in quotation marks.

DISPLAY DIRECTORY INQUIRY (PDI) Command - Inquiries Without SQL

Example: "List the stored inquiry named DEGREE.SALARY."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY DIRECTORY INQUIRY PLANT 'DEGREE.SALARY';

DEGREE.SALARY                    MXA      01-14-2002 14:29
DISPLAY PLANT    EMP.NO EMP.NAME ED.DEGREE SAL.YTD IF SAL.YEAR =   95
AND  ED.DEGREE =     %001 AND SAL.YTD <=     %002 ;
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure 9-13 Displaying a Stored Inquiry from the Directory

DISPLAY DIRECTORY INQUIRY (PDI) Command - Inquiries with SQL

You can use the DISPLAY DIRECTORY INQUIRY or PDI command to display native SQL syntax stored inquiries.

DISPLAY DIRECTORY	INQUIRY	EXECSQL	ROSTER2	ENDEXEC;
display items from the directory	item type to display	native SQL syntax delimiter	name of item to be displayed	native SQL syntax delimiter

This statement tells VISION:Inquiry to list the native SQL syntax inquiry named ROSTER2, which has been stored in the directory.

Example: "List the native SQL syntax stored inquiry named ROSTER2."

```
PAGE:      TRANCODE: II       INQUIRY:
DISPLAY DIRECTORY INQUIRY EXECSQL ROSTER2 ENDEXEC;

 ROSTER2               VTPSPG89 03-21-2001 15:10
DISPLAY 'REPORT OF EMPLOYEE INFORMATION'
EXECSQL SELECT EMPLOYEE, YEAR, YEAR_TO_DATE FROM
DYLINQ.IIEMP_SAL ENDEXEC;
*
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.               (0,0 USER DB CALLS,ROOTS)
```

Figure 9-14 Displaying a Native SQL Syntax Stored Inquiry from the Directory

Figure 9-13 and Figure 9-14 show the output from the DISPLAY DIRECTORY command just described. VISION:Inquiry lists the name of the requested inquiry and indicates the name of the last terminal, date and time that modified the inquiry, or, in case of no modification, the date and time the inquiry was stored and the name of the originating terminal.

In the inquiry without native SQL syntax, the query is listed in the format in which it is stored internally. The spaces between the words and lines and the format of the numbers appear differently from the way they looked when the inquiry was entered.

In the inquiry with native SQL syntax, the query is listed in the same format in which it is stored.

The comment, if it exists, is not displayed when using the DISPLAY DIRECTORY command.

Using DISPLAY DIRECTORY FUNCTION (PDF) to List a Stored Function

Figure 9-15 illustrates using the DISPLAY DIRECTORY FUNCTION or PDF command to show a stored function.

Example: “Display the stored function named MON.SAL.”

```
PAGE:       TRANCODE: II      INQUIRY:
DISPLAY DIRECTORY FUNCTION PLANT 'MON.SAL';



MON.SAL                          MXA      11-15-2001 14:17
%MONTH.SAL = (SAL.YTD - SAL.DED) / '12.0000 '
*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure 9-15 Using the DISPLAY DIRECTORY Command to List a Stored Function

The first line of the output contains the terminal name, date, and time the function was last modified; or, in case of no modification, the name of the originating terminal and the date and time when the function was stored. The last line shows the terms of the function as they are stored in the directory.

Using DISPLAY DIRECTORY to List All Inquiries or Functions

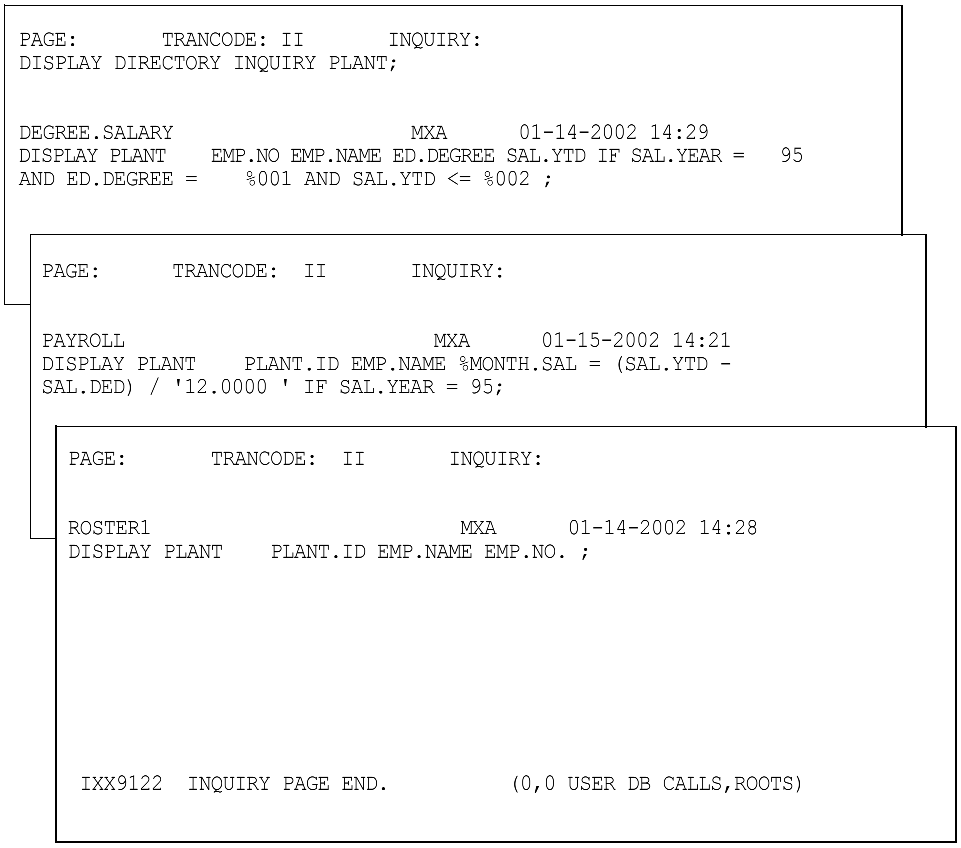
You can also use the DISPLAY DIRECTORY command to display all the inquiries or functions stored in the directory for a database.

`DISPLAY DIRECTORY`	`INQUIRY`	`PLANT`
display items from the directory	stored item type to display	related to this database

This statement tells VISION:Inquiry to display all the inquiries stored in the directory for the PLANT database.

Displaying All Inquiries Stored for a Database

Example: "List all the stored inquiries that relate to the PLANT database."



```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY DIRECTORY INQUIRY PLANT;


DEGREE.SALARY                    MXA      01-14-2002 14:29
DISPLAY PLANT    EMP.NO EMP.NAME ED.DEGREE SAL.YTD IF SAL.YEAR =   95
AND ED.DEGREE =    %001 AND SAL.YTD <= %002 ;
```

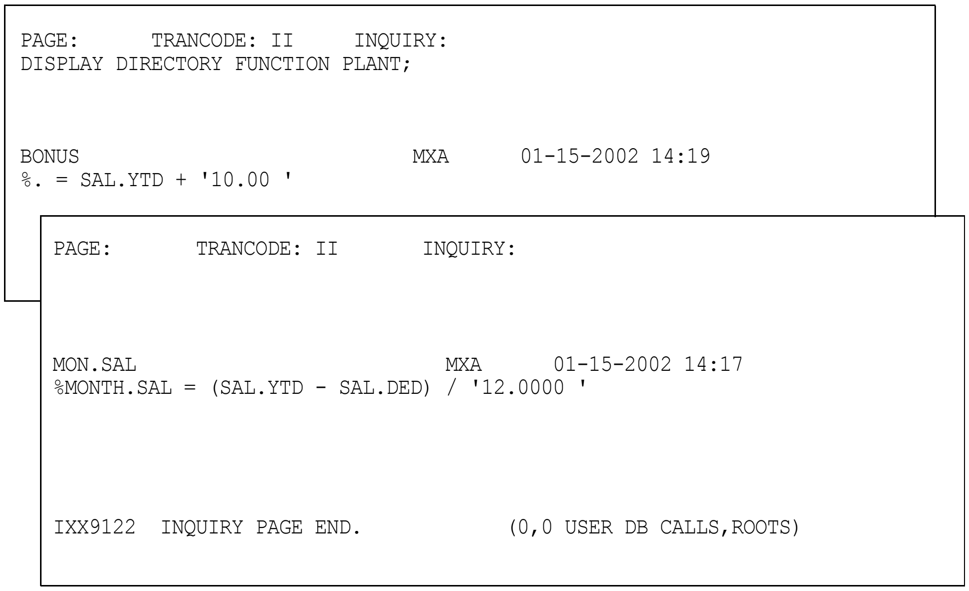
```
PAGE:      TRANCODE:  II       INQUIRY:


PAYROLL                          MXA      01-15-2002 14:21
DISPLAY PLANT    PLANT.ID EMP.NAME %MONTH.SAL = (SAL.YTD -
SAL.DED) / '12.0000 ' IF SAL.YEAR = 95;
```

```
PAGE:       TRANCODE:  II       INQUIRY:


ROSTER1                          MXA      01-14-2002 14:28
DISPLAY PLANT    PLANT.ID EMP.NAME EMP.NO. ;






 IXX9122  INQUIRY PAGE END.          (0,0 USER DB CALLS,ROOTS)
```

Figure 9-16   Displaying All Inquiries Stored for a Database

Displaying All Functions Stored for a Database

Example: “List the stored functions that relate to the PLANT database.”



```
PAGE:      TRANCODE: II     INQUIRY:
DISPLAY DIRECTORY FUNCTION PLANT;


BONUS                                MXA      01-15-2002 14:19
%. = SAL.YTD + '10.00 '
```

```
PAGE:       TRANCODE: II       INQUIRY:



MON.SAL                              MXA      01-15-2002 14:17
%MONTH.SAL = (SAL.YTD - SAL.DED) / '12.0000 '




IXX9122  INQUIRY PAGE END.            (0,0 USER DB CALLS,ROOTS)
```

Figure 9-17 Displaying All Functions Stored for a Database

The functions in Figure 9-17 are in alphabetical order, with each appearing on a separate page.

A DISPLAY DIRECTORY list command shows either inquiries or functions. Separate commands must be used to request each listing.

Using DISPLAY DIRECTORY to List All Inquiries - with SQL

The DISPLAY DIRECTORY command can also be used to display all the native SQL syntax stored inquiries.

`DISPLAY DIRECTORY`	`INQUIRY`	`EXECSQL`	`ENDEXEC;`
display items from the directory	item type to display	native SQL syntax delimiter	native SQL syntax delimiter

This statement tells VISION:Inquiry to display all the inquiries stored in the directory for the native SQL syntax facility. Each stored inquiry will be displayed on a separate page.

DISPLAY DIRECTORY Command Summary - Inquiries Without SQL

This section summarizes DISPLAY DIRECTORY for inquiries without the native SQL syntax facility of DB2 option.

DISPLAY DIRECTORY INQUIRY or PDI	Displays one or more inquiries that are stored in the directory.
or	
DISPLAY DIRECTORY FUNCTION or PDF	Displays one or more functions that are stored in the directory.
database name	Relates the stored items to this database.
'item name'	Displays this inquiry or function. If 'item name' is not present in the statement, all the inquiries or functions related to that database are displayed. The name must be enclosed in quotation marks.

DISPLAY DIRECTORY Command Summary - Inquiries With SQL

This section summarizes DISPLAY DIRECTORY for inquiries using the native SQL syntax facility of DB2 option:

DISPLAY DIRECTORY INQUIRY or PDI	Displays one or more inquiries that are stored in the directory.
EXECSQL	Starting delimiter of the native SQL statement.
item name	Displays this native SQL syntax inquiry. If item name is not present in the statement, all the stored native SQL syntax inquiries are displayed.
ENDEXEC	End delimiter of the native SQL statement.

Display Stored Inquiries for All Databases

Note: This section is not applicable to the native SQL syntax facility of DB2 option.

Commands discussed earlier in the chapter must be used to display the stored inquiries with the native SQL syntax facility.

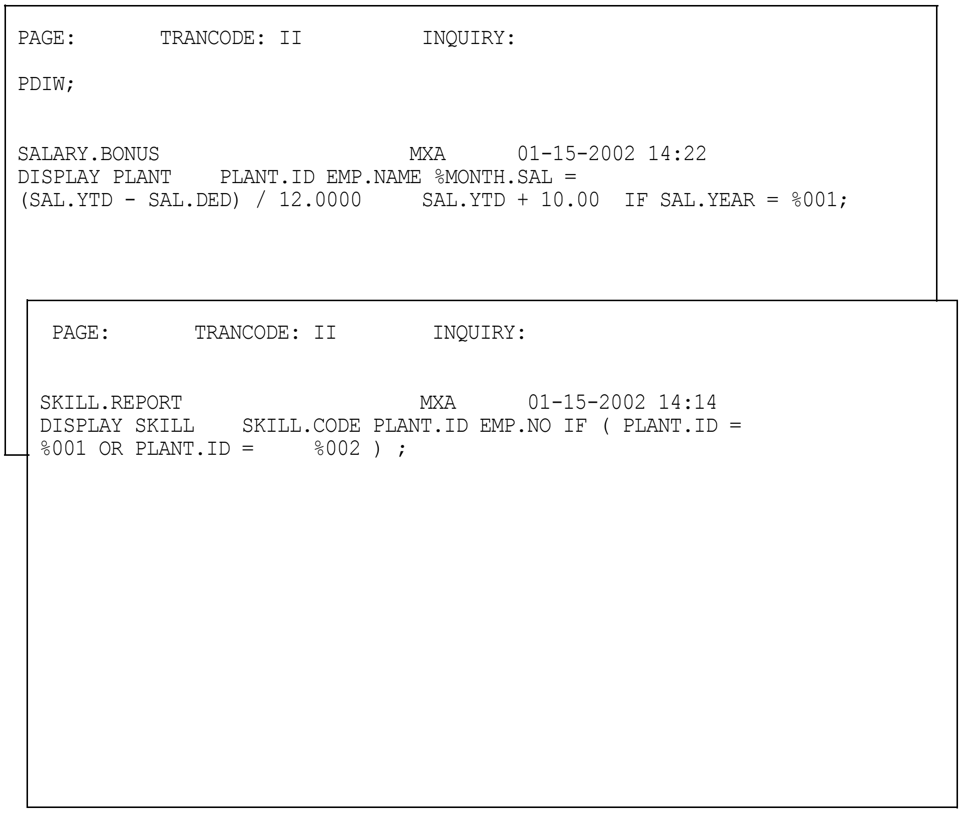
The following command allows you to display all stored inquiries for all databases.

```
DISPLAY DIRECTORY INQUIRY WHOLE;
```

or

```
PDIW;
```

Below are some of the inquiries we have saved in our directory.



```
PAGE:       TRANCODE: II          INQUIRY:

PDIW;


SALARY.BONUS                        MXA      01-15-2002 14:22
DISPLAY PLANT    PLANT.ID EMP.NAME %MONTH.SAL =
(SAL.YTD - SAL.DED) / 12.0000     SAL.YTD + 10.00  IF SAL.YEAR = %001;
```

```
 PAGE:       TRANCODE: II        INQUIRY:


SKILL.REPORT                        MXA      01-15-2002 14:14
DISPLAY SKILL    SKILL.CODE PLANT.ID EMP.NO IF ( PLANT.ID =
%001 OR PLANT.ID =     %002 ) ;
```

Figure 9-18 Displaying All Inquiries Stored in a Directory

Use the paging commands to view all of the inquiries stored in your directory.

Display Stored Inquiry Names and Their Associated Comments

Use the DISPLAY DIRECTORY INQUIRY COMMENT or PDIC command to display the query names and their associated comments stored for a specific database or for all databases. You can also use this command to display a comment associated with a stored query. This command can also be used for native SQL syntax. Figure 9-19 shows the output of the PDIC command to display the comments for the inquiries stored in the directory for the PLANT database.

```
PAGE:      TRANCODE: II       INQUIRY:
PDIC PLANT;

QUERY_NAME              DB_NAME TERM_NME DATE   TIME
 ROSTER1                        PLANT   MXA      02-22-02 08:27
    EMPLOYEE INFORMATION
 DEGREE.SALARY                   PLANT   L012     02-12-02 10:07
```

Figure 9-19 Display the Comments Associated With Queries Stored for the PLANT database

The output for each stored query consists of two lines. The first line shows the query name and database name of the stored query as well as the terminal name that stored the query and date and time the query is stored or last updated. The second line shows the comment stored with the query. A blank line displays if there is no comment associated with the stored query.

Display Comments Command Summary – Inquiries without SQL

This section summarizes the command used to display the comments associated with stored inquiries, for inquiries without the native SQL syntax facility of DB2 option.

DISPLAY DIRECTORY INQUIRY COMMENT **or** **PDIC**	Display one or more inquiry names and comments associated with them that are stored in the directory for a data base.
database name	Relates the stored inquiries to this database.
'stored inquiry name'	Displays this inquiry and its comment. If the name is not present in the statement, all the inquiries and their comments related to that database are displayed. In case of no comment, a blank comment line is displayed. The name must be enclosed in quotation marks.

You can also display the comments associated with stored inquiries for all the databases.

DISPLAY DIRECTORY INQUIRY COMMENTWHOLE **or** **PDICW**	Display the inquiry names, and comments associated with them, that are stored in the directory for all data bases. In case of no comment, a blank comment line is displayed.

Display Comments Command Summary – Inquiries with SQL

This section summarizes the command used to display the comments associated with stored inquiries for inquiries using the native SQL syntax facility of DB2 option.

DISPLAY DIRECTORY INQUIRY COMMENT **or** **PDIC**	Display one or more inquiry names and comments associated with them that are stored in the directory for a data base.
EXECSQL	Starting delimiter of the native SQL statement.
’stored inquiry name’	Displays this native SQL syntax stored inquiry and its comment. If the name is not present in the statement, all the stored native SQL inquiries and their comments are displayed. In case of no comment, a blank comment line is displayed.
ENDEXEC	Ending delimiter of the native SQL statement.

DELETE DIRECTORY Command

The DELETE command deletes a stored inquiry or function from the directory.

`DELETE DIRECTORY`	`INQUIRY`	`PLANT`	`'SALARY.BONUS';`
delete item stored in the directory	item to be deleted	item is related to this database	name of item to be deleted

This statement tells VISION:Inquiry to remove the inquiry named SALARY.BONUS from the stored inquiries belonging to the PLANT database. The output is shown in Figure 9-20.

Example: “The stored inquiry named SALARY.BONUS is no longer needed.”

```
PAGE:       TRANCODE: II        INQUIRY:
DELETE DIRECTORY INQUIRY PLANT 'SALARY.BONUS';


IXX0121 STORED INQUIRY/FUNCTION NAME 'SALARY.BONUS' HAS BEEN DELETED.
```

Figure 9-20 Deleting Inquiries from the Directory

The output from Figure 9-20 shows the message that is displayed when an item has been deleted from the directory.

DELETE DIRECTORY INQUIRY Command - Inquiries with SQL

The DELETE DIRECTORY command can also delete a native SQL syntax stored inquiry from the directory.

`DELETE DIRECTORY`	`INQUIRY`	`EXECSQL`	`ROSTER2`	`ENDEXEC;`
delete item stored in the directory	item type to be deleted	native SQL syntax delimiter	name of item to be deleted	native SQL syntax delimiter

This statement tells VISION:Inquiry to delete the native SQL syntax stored inquiry named ROSTER2 from the directory.

Using DELETE DIRECTORY FUNCTION to Delete Stored Functions

Stored functions can also be deleted.

`DELETE DIRECTORY`	`FUNCTION`	`PLANT`	`'MON.SAL';`
	item type to be deleted		name of item to be deleted

This statement tells VISION:Inquiry to delete the stored function named MON.SAL from the directory.

Deleting a stored function has no effect on stored inquiries that already contain the stored function, because the terms of the function have already been integrated into the inquiry. PAYROLL and SALARY.BONUS remain the same and can be executed as before.

Example: "The function named MON.SAL is no longer needed."

```
PAGE:       TRANCODE:  II        INQUIRY:
DELETE DIRECTORY FUNCTION PLANT 'MON.SAL' ;


IXX0121 STORED INQUIRY/FUNCTION NAME 'MON.SAL' HAS BEEN DELETED.
```

Figure 9-21 Deleting Stored Functions from the Directory

The output in Figure 9-21 shows the message VISION:Inquiry displays when it has deleted a function from the directory. Stored items must be deleted individually. The DELETE command cannot be used to delete all the items as a group.

DELETE Command Summary - Inquiries Without SQL

This section summarizes DELETE for inquiries without the native SQL syntax facility of DB2 option.

DELETE DIRECTORY	Tells VISION:Inquiry to delete an item from the directory.
INQUIRY or **FUNCTION**	Item is an inquiry or function.
database name	The item belongs to this database.
'item name'	The name of the item to be deleted. It must be enclosed in quotation marks.

DELETE Command Summary - Inquiries with SQL

This section summarizes DELETE for inquiries using the native SQL syntax facility of DB2 option.

DELETE DIRECTORY INQUIRY	Deletes a stored inquiry from the directory.
EXECSQL	The starting delimiter of the native SQL statement.
item name	The name of the native SQL syntax inquiry to be deleted.
ENDEXEC	The end delimiter of the native SQL statement.

Chapter 10

Using Stored Inquiries and Functions

This chapter describes how to use stored inquiries.

You and your system administrator write and store inquiries in the directory. The stored inquiries can be used later for repeated use.

Inquiries are stored in directories and are related to the particular database that they access. The names of the stored inquiries are assigned at the time they are stored. You can use the DISPLAY DIRECTORY commands in Chapter 9, "Using Directory Commands - Storing Inquiries and Functions" to see the names of inquiries that have been stored in a database directory. Your system administrator can provide you with a list of names of stored inquiries that you may execute.

Assume that an inquiry named ROSTER is stored in the directory, and contains the following statement:

```
DISPLAY PLANT 'EMPLOYEE INFORMATION FOR PLANTS WITH LOCATIONS ENDING WITH 00'
PLANT.ID EMP.NO EMP.NAME IF PLANT.ID LIKE '%00';
```

Executing a Stored Inquiry

To execute this stored inquiry, enter the following:

```
ROSTER             PLANT;
```

name of stored inquiry	related to this database

VISION:Inquiry retrieves the inquiry from the directory and executes it. The output from this stored inquiry is the same as if you had entered the statement at the terminal.

Figure 10-1 shows the display of ROSTER PLANT.

Example: "Use a stored inquiry to list all the employees by number and name within each plant for plant locations ending with 00. Display a title line at the top of each page of the report."

```
PAGE:      TRANCODE: II       INQUIRY:
ROSTER PLANT;

EMPLOYEE INFORMATION FOR PLANTS WITH LOCATIONS ENDING WITH 00
PLANT.ID  EMP.NO   EMP.NAME
10100      10103   WILLIAM AMES
           10104   PHYLLIS LOCKMEYER
           10105   MARY ANN THOMAS
30200      30201   JOHN HENRY CRANE
           30202   FREDERICH GRAY
           30205   MITCHELL J HOOPS
           30207   JANE LOWELL
           30211   PATRICIA BLAKELY
           30215   SHARON DALEY
40300      40304   DONALD M KING
           40306   JOAN EVANS
50300      50304   JONATHAN OAKS
           50322   MADELYN BATES
           50323   VICKY WARD
60200      60205   DAVID YORK
           60209   RUSSELL M SIMMONS
           60251   MARCIE MORINO
           60258   KAREN REDFERN
```

Figure 10-1 Using a Stored Inquiry Named 'ROSTER'

Because VISION:Inquiry defaults to PLANT, the first database in the directory, the previous inquiry statement could have been written simply as:

```
ROSTER;
```

It is a good idea to include the database name in the statement, especially when you are using different databases and are storing many inquiries with the same names in different directories.

Executing a Stored Inquiry with SQL

To execute stored inquiries that use the native SQL syntax facility, surround the stored inquiry name by the SQL delimiters, EXECSQL and ENDEXEC, as follows:

EXECSQL	ROSTER2	ENDEXEC;
native SQL syntax delimiter	name of stored inquiry	native SQL syntax delimiter

VISION:Inquiry retrieves the inquiry from the directory and executes it. There is no database name associated with the native SQL syntax stored inquiries. Every inquiry has a unique name and must be enclosed in native SQL syntax delimiters for execution.

Using Substitutable Values in Stored Inquiries

Note: This section is not applicable to the native SQL syntax facility of the DB2 option.

In some stored inquiries, the test values in the conditional selection phrases have not been explicitly stated. They have been replaced with substitutable values in the form of %1, %2, %3, and so on.

When you call a stored inquiry with substitutable values, you tell VISION:Inquiry what specific values you want to test for (such as 1995, 1994, and so on). VISION:Inquiry makes the substitution of the particular data you have indicated. This allows you to use the same inquiry against many sets of data.

This is what substitutable values look like in a stored inquiry.

```
DISPLAY PLANT EMP.NO EMP.NAME ED.DEGREE SAL.YTD
IF SAL.YEAR = 95 AND ED.DEGREE = %1 AND SAL.YTD LE %2;
```

When this stored inquiry is executed, the values are substituted on a one-to-one basis from left to right. %1 is replaced first, followed by %2.

The previous inquiry was stored under the user-selected name of DEGREE.SALARY. When you want to use it, you enter the name of the stored inquiry, the name of the database, and the substitutable values.

`DEGREE.SALARY`	`PLANT`	`'HS' 42000;`
name of the stored inquiry	database name	specific values being tested for

When this stored inquiry is executed, it displays the employee number and name, educational degree, and year-to-date salary from the PLANT database if ED.DEGREE = ‘HS’, if SAL.YTD is equal to or less than $42,000, and if the person worked during 1995.

Note that a character constant within a calling statement must be enclosed within single quotation marks.

Example: “There is a stored inquiry that lists those employees who graduated from high school, if they are earning $42,000 or less and worked during 1995. The inquiry also lists them by employee number and salary year-to-date. Use the stored inquiry to report this information.”

```
 PAGE:      TRANCODE: II       INQUIRY:
 DEGREE.SALARY PLANT 'HS' 42000;



 EMP.NO   EMP.NAME                  ED.DEGREE       SAL.YTD
  10105   MARY ANN THOMAS           HS            15,600.00
  21116   WILMA FORD                HS            18,800.00
  30201   JOHN HENRY CRANE          HS            22,000.00
  30207   JANE LOWELL               HS            26,000.00
  30215   SHARON DALEY              HS            17,000.00
  50323   VICKY WARD                HS            20,000.00
  60258   KAREN REDFERN             HS            26,000.00
  70519   STEPHEN MCGEE             HS            23,000.00
  70522   AGNES COVINGTON           HS            23,000.00
*
*
*
*
*
 IXX9121  END OF INQUIRY.               (148,7 USER DB CALLS,ROOTS)
```

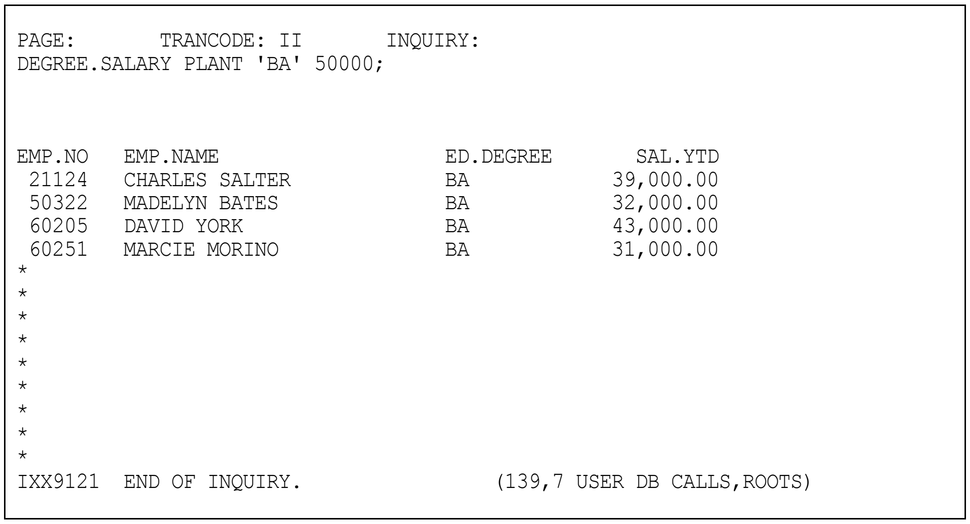
Figure 10-2 Using Substitutable Values in Stored Inquiries

Using Different Substitutable Values in Subsequent Runs

You can use different substitutable values in your stored inquiry simply by entering the new values at your terminal. To see a report of the employees holding a Bachelor of Arts degree who are earning $50,000 or less, you substitute these values into the calling statement.

```
DEGREE.SALARY PLANT 'BA' 50000;
```

Example: "Report those employees who have BA degrees, earn $50,000 or less, and worked during 1995. There is a stored inquiry that does this."



```
PAGE:       TRANCODE: II       INQUIRY:
DEGREE.SALARY PLANT 'BA' 50000;



EMP.NO   EMP.NAME                   ED.DEGREE       SAL.YTD
 21124   CHARLES SALTER             BA            39,000.00
 50322   MADELYN BATES              BA            32,000.00
 60205   DAVID YORK                 BA            43,000.00
 60251   MARCIE MORINO              BA            31,000.00
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                 (139,7 USER DB CALLS,ROOTS)
```

Figure 10-3 Using a Different Set of Substitutable Values

The report formats in Figure 10-2 and Figure 10-3 are the same. The different values substituted in the conditional selection phrase of the inquiry statement selected different employees from the database.

Using different sets of substitutable values in your stored inquiries gives you both flexibility and ease of operation when you are executing your inquiry statements.

Using COUNT and TOTAL in a Stored Inquiry

Look at another example that shows you the ease with which you may execute these inquiries. SALARY.REPORT has been defined as:

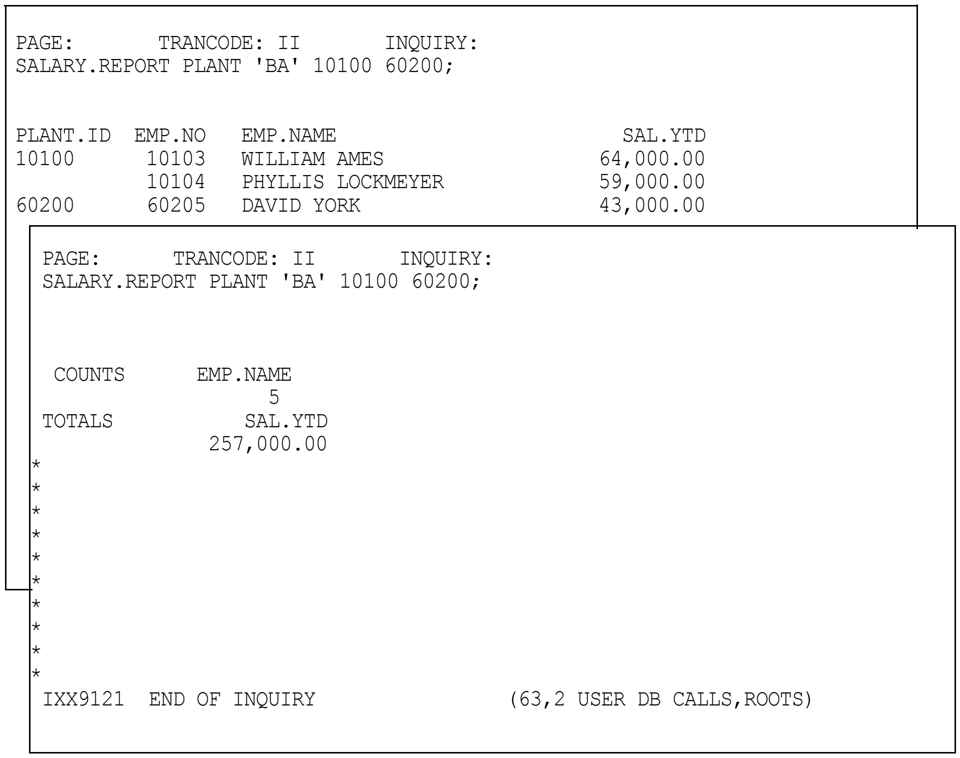
```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME SAL.YTD COUNT EMP.NAME
TOTAL SAL.YTD IF SAL.YEAR = 95 AND ED.DEGREE = %1 AND
(PLANT.ID = %2 OR PLANT.ID = %3);
```

This is what you enter at your terminal to execute the inquiry statement.

```
SALARY.REPORT PLANT 'BA' 10100 60200;
```

Figure 10-4 shows that the output appears on two pages, with the grand summaries displayed on the second page of the report.

Example: "List those employees who worked during 1995, if they have a BA degree and if they worked in either plant 10100 or 60200. Also, list the employee's plant location, employee number, and salary. Count the number of employees and list the total of their salaries. Use the stored inquiry to do this."



```
PAGE:       TRANCODE: II       INQUIRY:
SALARY.REPORT PLANT 'BA' 10100 60200;


PLANT.ID  EMP.NO   EMP.NAME                          SAL.YTD
10100      10103   WILLIAM AMES                    64,000.00
           10104   PHYLLIS LOCKMEYER               59,000.00
60200      60205   DAVID YORK                      43,000.00
```

```
PAGE:      TRANCODE: II       INQUIRY:
SALARY.REPORT PLANT 'BA' 10100 60200;



 COUNTS       EMP.NAME
                    5
TOTALS            SAL.YTD
              257,000.00
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY                (63,2 USER DB CALLS,ROOTS)
```

Figure 10-4 Using COUNT and TOTAL in a Stored Inquiry

Using Stored Inquiries with LIMIT and OUTPUT

If the LIMIT command is included in an inquiry definition, it is ignored during execution. However, the LIMIT command is displayed when editing the stored inquiry using EDITSQ command (discussed later in this chapter).

In the same way, OUTPUT is ignored during execution if it is included in a stored inquiry. OUTPUT must be added to the stored inquiry when it is executed.

You can add the LIMIT command to a stored inquiry when it is executed. For example, to execute the DEGREE.SALARY stored inquiry and limit the output to 4, you enter the following:

```
DEGREE.SALARY PLANT 'HS' 42000 LIMIT 4;
```

Example: "Before listing the employees by number within plant location, take a sampling of 4."

```
PAGE:       TRANCODE: II       INQUIRY:
DEGREE.SALARY PLANT 'HS' 42000 LIMIT 4;


EMP.NO   EMP.NAME                   ED.DEGREE       SAL.YTD
 10105   MARY ANN THOMAS            HS            15,600.00
 21116   WILMA FORD                 HS            18,800.00
 30201   JOHN HENRY CRANE           HS            22,000.00
 30207   JANE LOWELL                HS            26,000.00
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                 (79,3 USER DB CALLS,ROOTS)
```

Figure 10-5 Using LIMIT with the Stored Inquiry

In order to limit the output to four EMP.NOs (the highest segment referenced in the command), the LIMIT command and the integer 4 were added to the inquiry that was used in Figure 10-2. When you compare the outputs of Figure 10-2 and Figure 10-5, you see that the LIMIT command affected the output of the latter inquiry, limiting it to the first four EMP.NOs accessed from the database.

Using Stored Inquiries Summary - Inquiries Without SQL

This section summarizes stored inquiries without native SQL syntax facility.

stored inquiry name	Name under which inquiry is stored in the directory.
database name	Inquiry belongs to this database.
substitutable value(s)	Substitute this value for a condition in the selection phrase.

A statement may contain several substitutable values. They are substituted on a one-to-one basis from left to right: %1 is replaced first, then %2, and so on.

Character constants in substitutable values must be enclosed in single quotation marks.

Using Stored Inquiries Summary - Inquiries with SQL

This section summarizes stored inquiries with native SQL syntax facility.

EXECSQL	Starting delimiter of the native SQL statement.
stored inquiry name	Name under which the native SQL syntax inquiry is stored in the directory.
ENDEXEC	End delimiter of the native SQL statement.

Note:

When used in an inquiry with or without SQL syntax, the LIMIT and OUTPUT commands included in the stored inquiry are displayed when editing the stored inquiry using EDITSQ command (discussed later in this chapter), but are ignored when executing the stored inquiry. However, both may be added to a stored inquiry when it is executed.

Text Editor Facility

The Text Editor facility gives you full editing capability of the inquiries stored in the directory. Note that in the examples in this section, it is assumed that you have already stored an inquiry with the name EMPINFO for the PLANT database, with the following content:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME;;
```

EDITSQ Command

The EDITSQ command displays a stored inquiry on the Text Editor screen for editing.

`EDITSQ`	`PLANT`	`'EMPINFO';;`
command to edit	item is related to this database	name of item to be edited

This EDITSQ command tells VISION:Inquiry to display the existing inquiry named EMPINFO from the stored inquiries belonging to the PLANT database.

Using EDITSQ with Inquiries Containing SQL

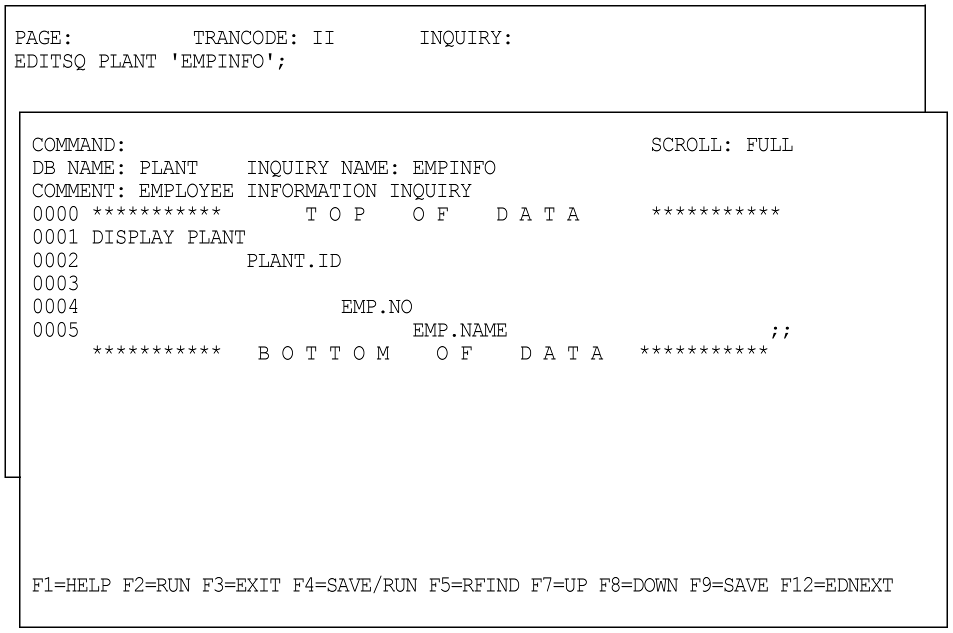
The EDITSQ command can also be used to display a native SQL syntax stored inquiry on the Text Editor screen.

`EDITSQ`	`EXECSQL`	`ROSTER2`	`ENDEXEC;`
command to edit	native SQL syntax delimiter	name of item to be edited	native SQL syntax delimiter

EDITSQ, along with the delimiters, tells VISION:Inquiry to display the existing native SQL syntax inquiry named ROSTER2, from the stored inquiries in the directory.

Displaying Inquiries for Editing

Figure 10-6 shows the EDITSQ command and VISION:Inquiry Text Editor screen as output to the command with database name PLANT, the stored inquiry name EMPINFO, and a 60-byte comment.



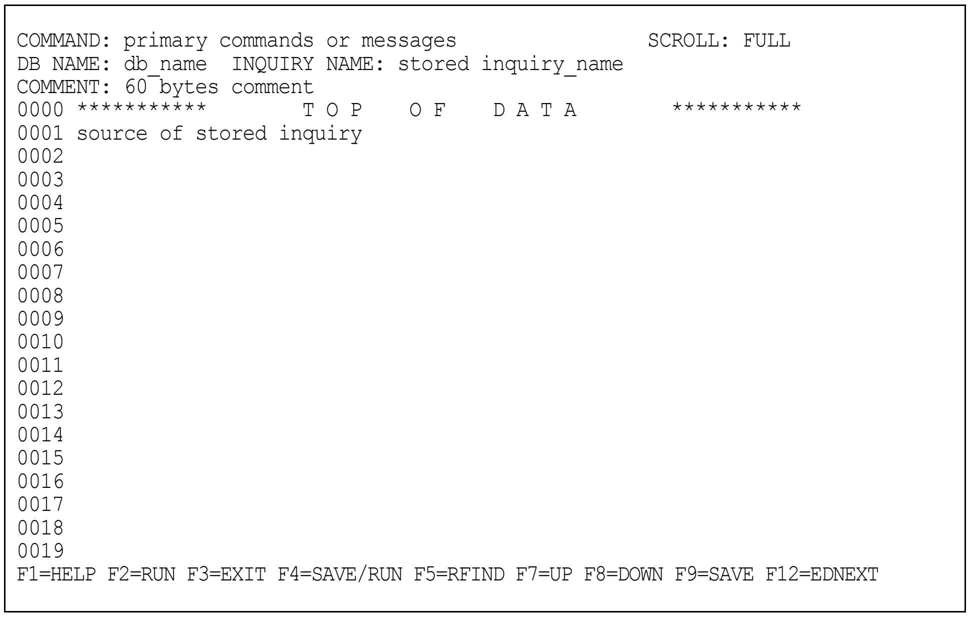
```
PAGE:          TRANCODE: II       INQUIRY:
EDITSQ PLANT 'EMPINFO';
```

```
COMMAND:                                                      SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********      T O P    O F    D A T A      ***********
0001 DISPLAY PLANT
0002             PLANT.ID
0003
0004                    EMP.NO
0005                          EMP.NAME                      ;;
     ***********   B O T T O M    O F    D A T A   ***********








F1=HELP F2=RUN F3=EXIT F4=SAVE/RUN F5=RFIND F7=UP F8=DOWN F9=SAVE F12=EDNEXT
```

Figure 10-6 Display Inquiries for Editing

Text Editor Screen Layout

The Text Editor screen is a full editing screen containing fields that you can change, and two areas, command and line, for entering the commands. Figure 10-7 shows the Text Editor screen layout.



```
COMMAND: primary commands or messages                SCROLL: FULL
DB NAME: db_name  INQUIRY NAME: stored inquiry_name
COMMENT: 60 bytes comment
0000 ***********        T O P    O F    D A T A       ***********
0001 source of stored inquiry
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
F1=HELP F2=RUN F3=EXIT F4=SAVE/RUN F5=RFIND F7=UP F8=DOWN F9=SAVE F12=EDNEXT
```

Figure 10-7 Text Editor Screen Layout

The description of the fields and areas in Figure 10-7 are as follows:

Field/Area	Description
COMMAND	This is the primary command line. The commands which affect a group of primary commands can start in any position of the COMMAND field. The primary command area is also used for displaying the messages issued by the Text Editor.
SCROLL	Allows you to scroll the lines up or down.
DB NAME	The database name to which the stored inquiry belongs. It can be changed when saving the inquiry or asking for another inquiry to edit. For native SQL syntax stored inquiries, the DB NAME field shows the keyword EXECSQL, which should not be changed when using Text Editor functions/commands.

INQUIRY NAME	The stored inquiry name. It can be changed when saving the inquiry or asking for another inquiry to edit. The Text Editor screen INQUIRY NAME field is 32 characters in length. When you edit a stored inquiry with a name of more than 32 characters, only the first 32 characters are used. If you save the inquiry, it is saved by the first 32 characters of the name, or a new name that you enter.
COMMENT	The 60-byte comment area. It can be changed when saving the inquiry. The characters semicolon and single quotation mark are not allowed in the comment text.
LINE COMMANDS	Line or sequence field commands can be entered by typing over the 4-digit sequence number of one or more lines on the Text Editor screen. Line commands affect a single line or a block of lines. Error messages appear if the line commands are incorrect or incomplete.
SOURCE OF STORED INQUIRY	The source of the stored inquiry is displayed from the fifth line of the first page (screen) of the Text Editor, and is continued on the subsequent pages (screens) starting from the fourth line. Each line is 72 bytes long, and a 4-digit sequence number is assigned to each line. You can change or edit the source, and then execute or save the inquiry using the PF keys. The Text Editor can process up to 304 lines of stored inquiries.
PF KEY FUNCTIONS	The last line of the screen shows what function is assigned to each PF key.

Text Editor PF/PA Keys and Their Functions

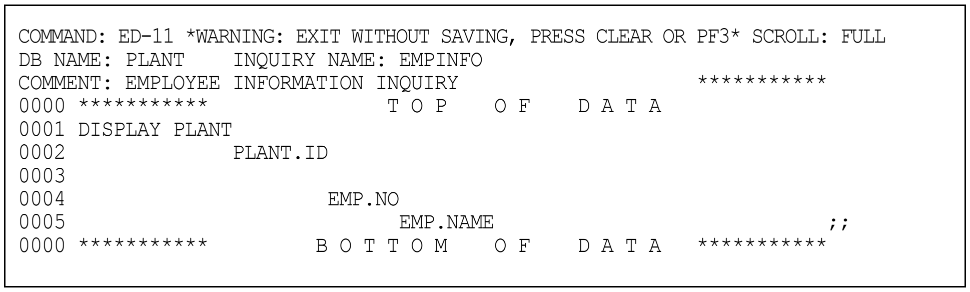
The PF/PA keys are:

PF1 (HELP) Shows a description of the available commands and the PF key functions.

PF2 (RUN) Passes the inquiry on the Text Editor screen to VISION:Inquiry for execution and returns the output back on the native VISION:Inquiry screen. This key does not save the stored inquiry back to the directory.

PF3 or Clear (EXIT) Pressing the PF3 key the second time clears the screen and exits from Text Editor and the VISION:Inquiry transaction.

- For the CICS version, the Clear key acts exactly the same as the PF3 key.
- For the IMS version, the Clear key clears the screen, exits the Text Editor, and does not send a message.

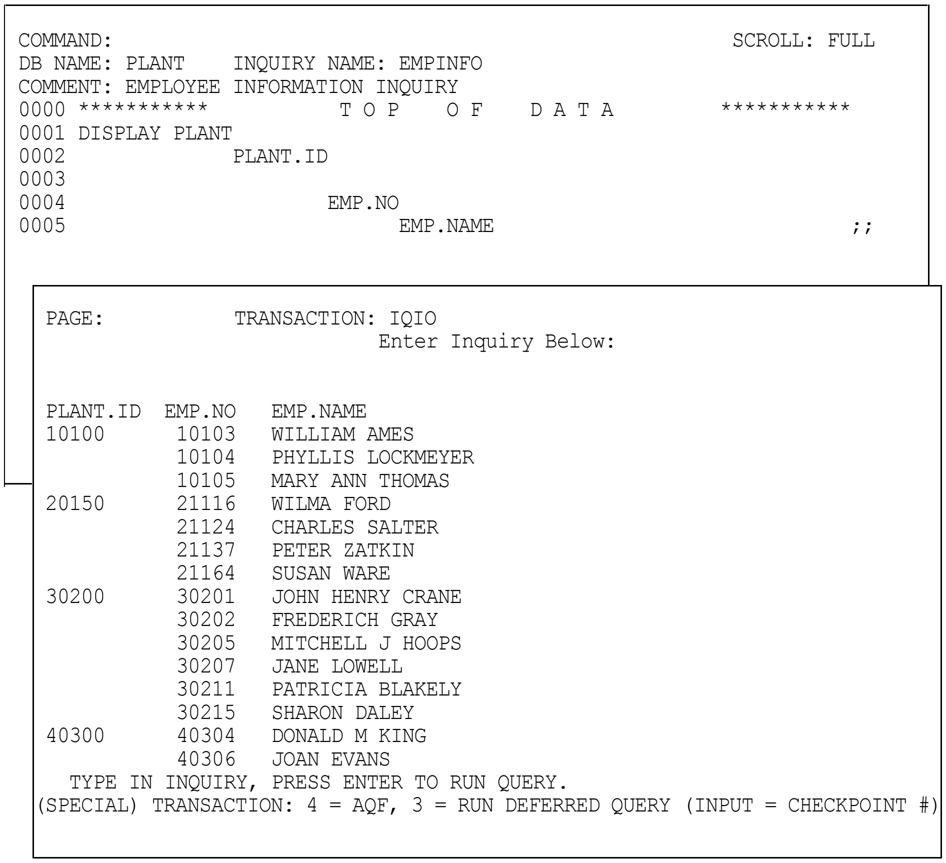


```
COMMAND: ED-11 *WARNING: EXIT WITHOUT SAVING, PRESS CLEAR OR PF3* SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY                    ***********
0000 ***********              T O P    O F    D A T A
0001 DISPLAY PLANT
0002             PLANT.ID
0003
0004                     EMP.NO
0005                           EMP.NAME                            ;;
0000 ***********        B O T T O M    O F    D A T A    ***********
```

Figure 10-8 The Text Editor Screen When PF3 or the Clear Key is Pressed

PF4 (SAVE/RUN) This key has two functions:

- It saves the inquiry with the name in the 'INQUIRY NAME' field for the database in the 'DB NAME' field. If the inquiry name already exists in the directory, the Text Editor first issues a warning message on the command line.
- Pressing PF4 for the second time replaces the existing inquiry. The second function is the same as the PF2 key, which executes the inquiry and returns the output to the native VISION:Inquiry screen.



```
COMMAND:                                                     SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********           T O P    O F    D A T A         ***********
0001 DISPLAY PLANT
0002                PLANT.ID
0003
0004                        EMP.NO
0005                              EMP.NAME                             ;;
```

```
PAGE:           TRANSACTION: IQIO
                            Enter Inquiry Below:


PLANT.ID  EMP.NO   EMP.NAME
10100      10103   WILLIAM AMES
           10104   PHYLLIS LOCKMEYER
           10105   MARY ANN THOMAS
20150      21116   WILMA FORD
           21124   CHARLES SALTER
           21137   PETER ZATKIN
           21164   SUSAN WARE
30200      30201   JOHN HENRY CRANE
           30202   FREDERICH GRAY
           30205   MITCHELL J HOOPS
           30207   JANE LOWELL
           30211   PATRICIA BLAKELY
           30215   SHARON DALEY
40300      40304   DONALD M KING
           40306   JOAN EVANS
  TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
(SPECIAL) TRANSACTION: 4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 10-9 The Text Editor Screen When PF4 is Pressed

PF5 (RFIND)	Repeats the last FIND command entered and finds the next occurrence of a string by moving the cursor to the first position of the string. The search begins after the current cursor position. The error message "STRING NOT FOUND" is issued when there is no string found from after the cursor position to the bottom of data. Pressing PF5 after this message is issued will start the process from the top of data, provided that you have not moved the cursor or changed any line. This key should always be used after the primary command FIND (which is discussed later). The FIND command can be used to find the first occurrence of a string.
PF6 (RCHANGE)	Repeats the last CHANGE command entered and changes the next occurrence of a string starting from the cursor position. Data is shifted to the right if necessary, and the cursor moves to the first position after the changed string. The error message "STRING NOT FOUND" is issued when there is no string found to be changed from the cursor position to the bottom of data. Pressing PF6 after this message is issued will start the process from the top of data, provided that you have not moved the cursor or changed any line. This key should always be used after the primary command CHANGE (which is discussed later). The CHANGE command can be used to change the first occurrence of a string.
PF7, PF8 (UP, DOWN)	Scrolls the screen up or down on the Text Editor screen. The number of lines it scrolls depends on the content of the 'SCROLL' field:

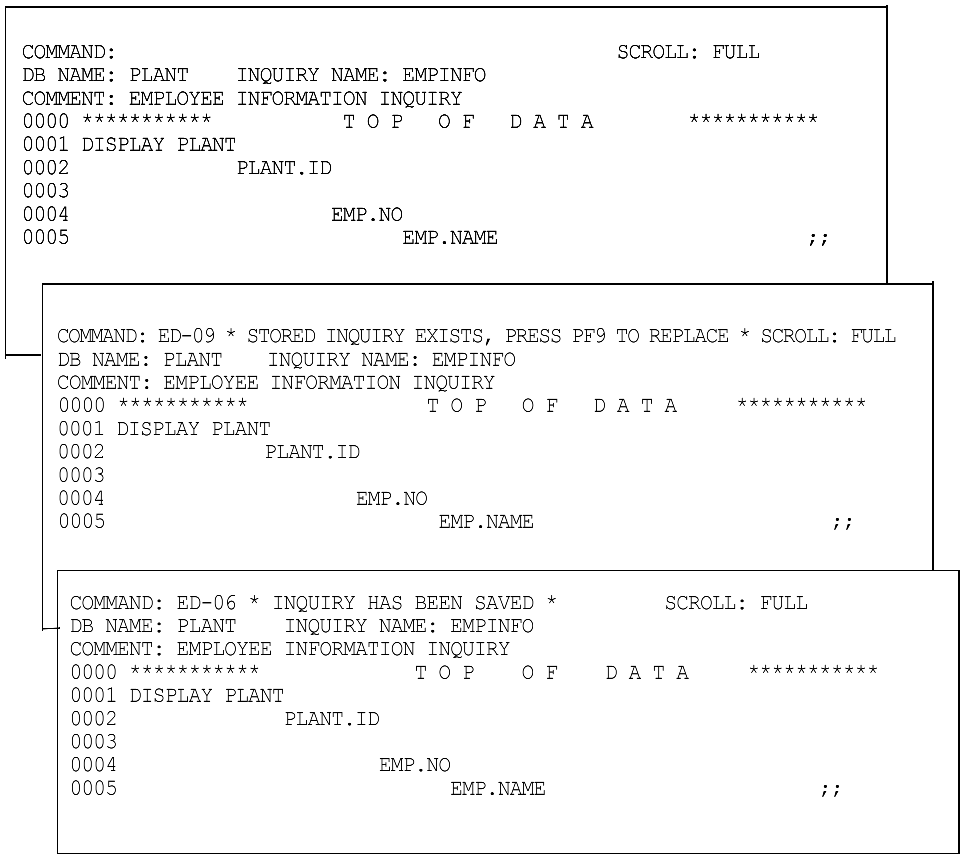
Scroll field	**Number of lines scrolled**
FULL	20
HALF	10
CSR	Scrolls until the line that contains the cursor becomes the first data line on the Text Editor screen.

The number of lines to be scrolled can also be entered directly in the COMMAND field.

PF9 (SAVE) Saves the inquiry (in the directory) with the name in the 'INQUIRY NAME' field for the database in the 'DB NAME' field.

It issues a message after the save is completed and stays in the Text Editor screen for further editing. If the inquiry name already exists in the directory, the Text Editor first issues a warning message on the command line.

Pressing PF9 for the second time replaces the existing inquiry in the directory and returns back to the Text Editor screen. Figure 10-10 shows an example of using the PF9 key.



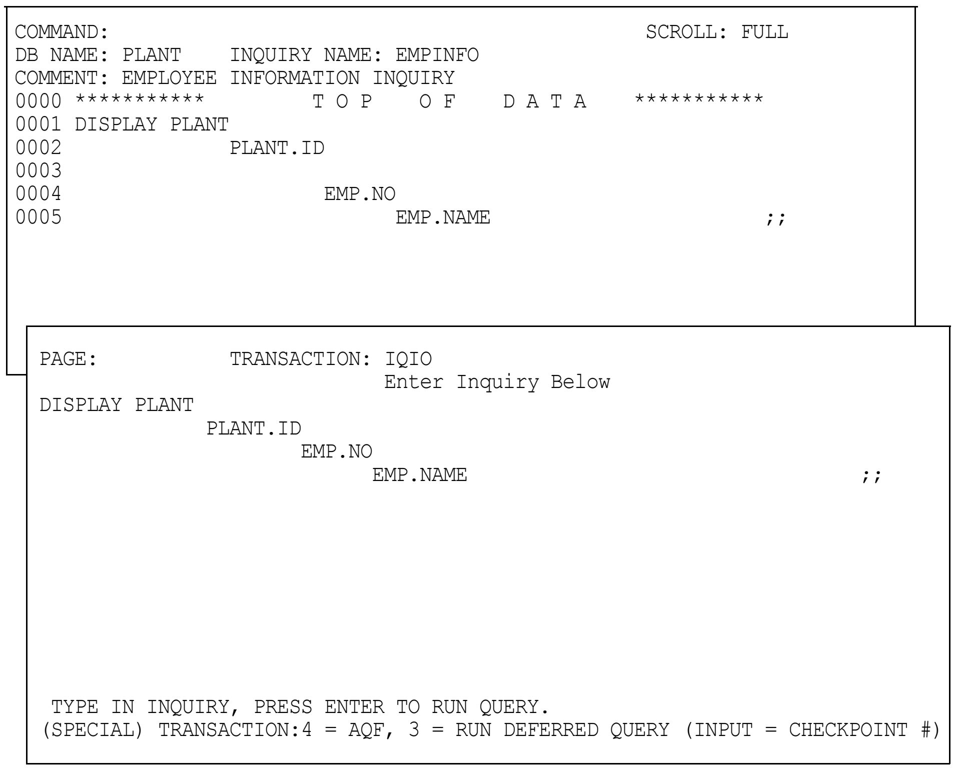
```
COMMAND:                                                   SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********           T O P   O F   D A T A       ***********
0001 DISPLAY PLANT
0002              PLANT.ID
0003
0004                      EMP.NO
0005                            EMP.NAME                          ;;
```

```
COMMAND: ED-09 * STORED INQUIRY EXISTS, PRESS PF9 TO REPLACE * SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********               T O P   O F   D A T A     ***********
0001 DISPLAY PLANT
0002             PLANT.ID
0003
0004                     EMP.NO
0005                            EMP.NAME                        ;;
```

```
COMMAND: ED-06 * INQUIRY HAS BEEN SAVED *         SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********             T O P   O F   D A T A     ***********
0001 DISPLAY PLANT
0002             PLANT.ID
0003
0004                     EMP.NO
0005                           EMP.NAME                       ;;
```

Figure 10-10 The Text Editor Screen When PF9 is Pressed

PF10 (PASS) Passes the inquiry to native VISION:Inquiry for execution. You should then press Enter for processing. Figure 10-11 shows an example of using the PF10 key.



```
COMMAND:                                                    SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********         T O P    O F    D A T A    ***********
0001 DISPLAY PLANT
0002              PLANT.ID
0003
0004                      EMP.NO
0005                            EMP.NAME                       ;;
```

```
PAGE:           TRANSACTION: IQIO
                             Enter Inquiry Below
DISPLAY PLANT
              PLANT.ID
                      EMP.NO
                            EMP.NAME                             ;;

 TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
(SPECIAL) TRANSACTION:4 = AQF, 3 = RUN DEFERRED QUERY (INPUT = CHECKPOINT #)
```

Figure 10-11 The Text Editor Screen when PF10 is Pressed

PF11 (SAVE/PASS) Functions as the combination of the PF9 and PF10 keys. It saves the inquiry first and then passes it to native VISION:Inquiry for execution.

PF12 (EDNEXT) Displays on the Text Editor screen the next stored inquiry whose name is entered in the 'INQUIRY NAME' field for the database in the 'DB NAME' field.

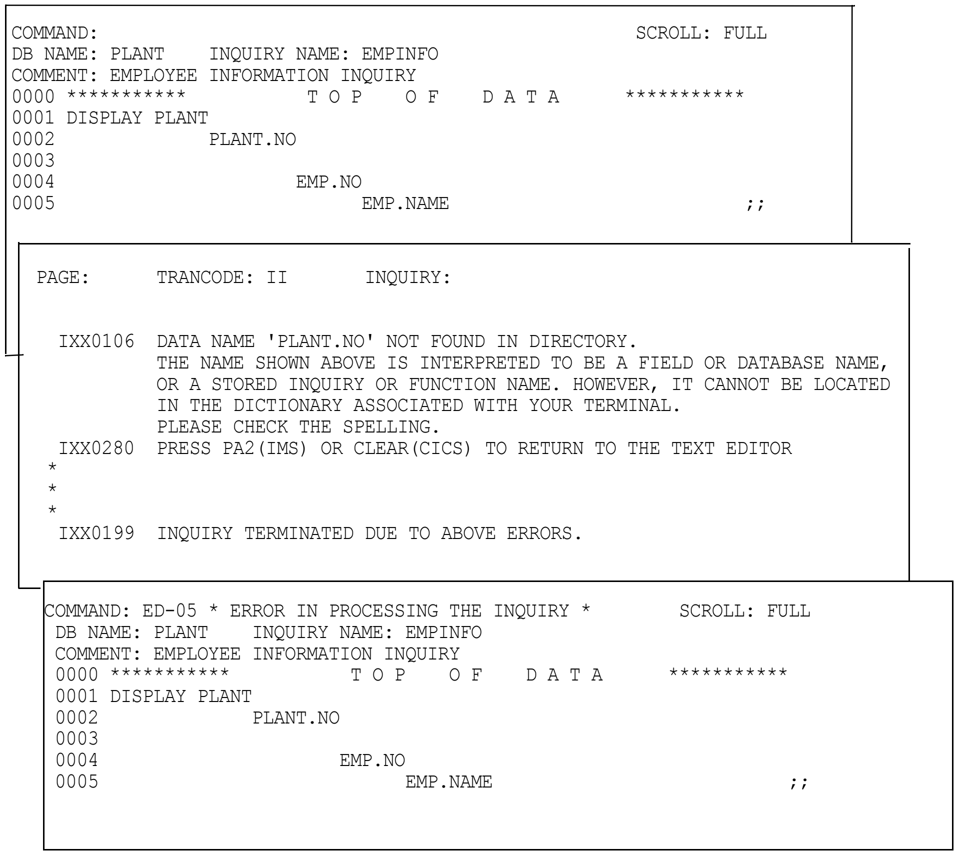
For example, in Figure 10-12, after replacing the stored inquiry 'EMPINFO', you can enter the name of another stored inquiry in the 'INQUIRY NAME' field and the name of its database in the 'DB NAME' field and press the PF12 key to edit and process it without leaving the Text Editor screen.

PA1, **PA2**, or **PA3** (**REFRESH**)	Reverses the last editing function and refreshes the screen from the last screen it sent you. For the IMS version, see the note below.

Notes:

- An error message displays if any error is detected during the processing of the inquiry. This allows you to correct the error on the Text Editor screen and rerun the inquiry. See Figure 10-12.
- PF10 and PF11 can process inquiries up to 20 lines long. If inquiries are longer than 20 lines, you should use the PF2 and PF4 keys instead.
- Using the PF4 and PF11 keys to replace a stored inquiry will issue a warning message on the command line. Pressing the PF key for a second time replaces the inquiry. Using other PF keys between the first and second press of the save key is allowed, except in the case of pressing the PF12 key or changing the 'DB NAME' or 'INQUIRY NAME' fields. In these instances, the process is reset and the warning message is issued again.
- If your terminal has 24 PF keys, you may use PF13 as PF1, PF14 as PF2, PF15 as PF3, and so on.
- Primary commands are also available with equivalent functions for most of the PF and PA keys described above. These commands are described in Primary Commands Equivalent to PF/PA Key Functions on page 10-21.
- The PA1, PA2, or PA3 key is available for the CICS version only.
  For the IMS version, use the REFRESH primary command.
- You can also edit and process inquiries stored in a connected directory using the Text Editor facility with the following considerations:
  - The primary directory is searched first for the specified stored inquiry. Then connected directories, if any, are searched.
  - The Text Editor Save command will always save the inquiry in the primary directory. Trying to save an edited stored inquiry from a connected directory will not replace it but saves it as a new inquiry in the primary directory. This is a security feature of VISION:Inquiry.
  - See your system administrator, or see the *Advantage VISION:Inquiry for IMS and TSO Technical Reference Guide*, for more detailed information about primary and connected directories.
- The LIMIT and OUTPUT commands stored in the inquiry are displayed on the Text Editor screen and executed when Text Editor commands such as PF2 (Run) and PF4 (Save/Run) are processed.

Returning to the Text Editor Screen After an Error



```
COMMAND:                                                  SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********          T O P    O F    D A T A      ***********
0001 DISPLAY PLANT
0002             PLANT.NO
0003
0004                     EMP.NO
0005                           EMP.NAME                           ;;
```

```
PAGE:      TRANCODE: II       INQUIRY:


  IXX0106  DATA NAME 'PLANT.NO' NOT FOUND IN DIRECTORY.
           THE NAME SHOWN ABOVE IS INTERPRETED TO BE A FIELD OR DATABASE NAME,
           OR A STORED INQUIRY OR FUNCTION NAME. HOWEVER, IT CANNOT BE LOCATED
           IN THE DICTIONARY ASSOCIATED WITH YOUR TERMINAL.
           PLEASE CHECK THE SPELLING.
  IXX0280  PRESS PA2(IMS) OR CLEAR(CICS) TO RETURN TO THE TEXT EDITOR
 *
 *
 *
  IXX0199  INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

```
COMMAND: ED-05 * ERROR IN PROCESSING THE INQUIRY *        SCROLL: FULL
 DB NAME: PLANT    INQUIRY NAME: EMPINFO
 COMMENT: EMPLOYEE INFORMATION INQUIRY
 0000 ***********          T O P    O F    D A T A      ***********
 0001 DISPLAY PLANT
 0002             PLANT.NO
 0003
 0004                     EMP.NO
 0005                           EMP.NAME                           ;;
```

Figure 10-12 Running a Stored Inquiry from the Text Editor Screen with an Invalid Field Name

To return to the Text Editor screen after an error is detected, press the PA2 key (for IMS) or the Clear key (for CICS).

Text Editor Primary Commands

The primary commands can start in any position of the COMMAND field. You can find a one-line description of the commands by pressing the PF1 (HELP) key. The following gives the description of each command in more detail:

U # or D #	Scrolls up or down "#" lines (a space is required between the command and "#").
#	Used with PF7/PF8 to scroll up/down "#" lines.
PL or Bottom	Displays the last page (last 20 lines) of the stored inquiry on the Text Editor screen.
PF or PB	Uses the scroll amount (Full=20 lines, Half=10 lines) to do forward or backward positioning.
HF or HB	Half-page (10 lines) forward or backward positioning.
Find xx	Finds and moves the cursor to the first position of the string "xx". Search starts after the current cursor position. "xx" can be any string enclosed between single or double quotation marks or spaces, depending on the contents of the string. You can use the PF5 key to find the next occurrences of the string.
Change xx yy	Changes the next occurrence of "xx" to "yy," and the cursor moves to the first position after the changed string. You can use the PF6 key to change the next occurrence of the string from the cursor position. See *Change xx yy ALL* next.
Change xx yy ALL	If the keyword ALL is used, the command changes all the occurrences of "xx" to "yy," and the cursor moves to the first position after the first changed string. "xx" and "yy" can be strings enclosed between single or double quotation marks or spaces, depending on the contents of the string. If "yy" is longer than "xx", the data is shifted to the right; if it does not fit on the line, an error message is issued and the cursor is positioned at the start of the string that caused the error.

Notes:

- If a primary command is entered incorrectly, the Text Editor facility will highlight the command and wait for you to correct it.
- In the FIND and CHANGE commands, the following conditions apply to the string.

String contains	String should be enclosed between
comma, space, or *	single or double quotation marks
single quotation mark	double quotation marks
double quotation marks	single quotation mark
any other character	space, single or double quotation marks

- The FIND and CHANGE commands can be abbreviated. As shown above, at least those parts of the command that are in uppercase should be entered (for example, the FIND command can be entered as F, FI, FIN, or FIND).
- The function of the FIND and CHANGE commands start from the cursor position, except when the ALL parameter is used with the CHANGE command. In the latter case, the whole inquiry is affected.

Primary Commands Equivalent to PF/PA Key Functions

The following shows the primary commands that you can use as an alternative to the PF and PA keys.

Note that part of the commands are in uppercase. To specify a command, you must enter the uppercase letters of the command. The lowercase letters are optional.

Primary command	PF/PA key equivalent
REFResh	PA1, PA2, or PA3
HElp	PF1
RUn	PF2
EXit or =X	PF3
SAVE/Run	PF4
Save	PF9
PAss	PF10
SAVE/Pass	PF11
EDit/next	PF12

The PA1, PA2, or PA3 key is available for the CICS version only. For the IMS version, use the REFRESH primary command.

Text Editor Line Commands

The line or sequence field commands usually apply to a single line or a group of lines on the Text Editor screen. These commands can be entered by typing over the 4-digit sequence number on each line of the Text Editor screen. You can find a one-line description of the commands by pressing the PF1 (HELP) key.

The following gives the description of each command in more detail:

I	Inserts one blank line after this line.
I#	Inserts "#" blank lines after this line.
D	Deletes this line.
D#	Deletes "#" lines starting from this line.
DD	Defines the start and end of a group of lines to be deleted.
C	Copies this line after or before the indicated target line.
C#	Copies "#" lines (starting from this line) after or before the indicated target line.
CC	Defines the start and end of a group of lines to be copied after or before the indicated target line. See Figure 10-13 and Figure 10-14.
M	Moves this line after or before the indicated target line.
M#	Move "#" lines (starting from this line) after or before the indicated target line.
MM	Defines the start and end of a group of lines to be moved after or before the indicated target line.
R#	Repeats "#" duplicate copies of the line containing R immediately after this line. If "#" is not specified, one copy is repeated.
RR	Defines the start and end of a group of lines to be repeated.
A, B	Defines the target line, "A" (after) this line, "B" (before) this line, for the copy and move line commands, C, C#, CC, M, M#, and MM.

Copying a Number of Lines

```
0000 ***********        T O P    O F    D A T A      ***********
0001  DISPLAY PLANT
CC02              PLANT.ID
0003                      EMP.NO
0004                            EMP.NAME
CC05                                    SAL.YTD                 ;;
0006
A007
```

Figure 10-13 Copies Lines 2 Through 5 After Line 7

```
0000 ***********         T O P    O F    D A T A     ***********
0001  DISPLAY PLANT
0002              PLANT.ID
0003                      EMP.NO
0004                            EMP.NAME
0005                                    SAL.YTD                 ;;
0006
0007
0008              PLANT.ID
0009                      EMP.NO
0010                            EMP.NAME
0011                                    SAL.YTD                 ;;
```

Figure 10-14 Result of the Execution of the Command in Figure 10-13

Notes:

- The line commands, I, I#, or A, may be entered on the line with the sequence number 0, which inserts line(s) or copies/moves line(s) at the beginning of the Text Editor screen. See Figure 10-15 and Figure 10-16. The line command B may be entered on the last line indicating "bottom of data" to copy/move line(s) at the bottom of data.
- Multiple I, I#, D, D#, and DD commands may be entered on one screen in one operation of the text editor. See Figure 10-17 and Figure 10-18.
- When inserting lines, if no data is entered on an inserted line, it is deleted the next time you press Enter. The exceptions are when an error message is issued, and when the inserted lines are part of a move/copy operation.
- Only one pair of CC, MM, RR or one C, C#, M, M#, R, or R# command can be specified in one operation of the text editor.
- The sequence numbers of the lines are resequenced automatically after the execution of the line commands.
- If an incorrect or incomplete line command is entered, the Text Editor will issue an error message and wait for you to correct the error (see Figure 10-19).

Inserting Blank Lines

```
I300 ***********                 T O P    O F    D A T A          ***********
0001 DISPLAY PLANT
0002              PLANT.ID
0003
0004                      EMP.NO
0005                            EMP.NAME
```

Figure 10-15 Inserts Three Blank Lines Before Line 1

```
0000 ***********                    T O P    O F    D A T A         ***********
....
....
....
0004 DISPLAY PLANT
0005              PLANT.ID
0006
0007                      EMP.NO
0008                            EMP.NAME                         ;;
```

Figure 10-16 Result of the Execution of the Command in Figure 10-15

Combining the D and I Line Commands

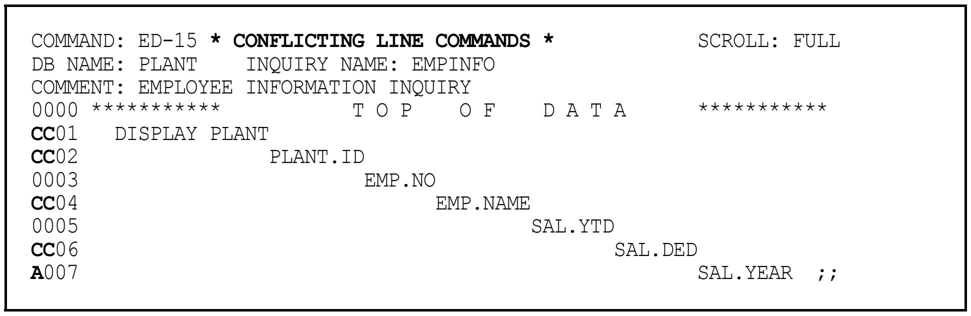
```
0000 ***********                T O P    O F    D A T A     ***********
I001  DISPLAY PLANT
0002               PLANT.ID
D003                       EMP.NO
0004                             EMP.NAME
D005                                     SAL.YTD
I2                                              SAL.DED
0007                                                   SAL.YEAR   ;;
```

Figure 10-17 Combines D and I Line Commands

```
0000 ***********                T O P    O F    D A T A     ***********
0001  DISPLAY PLANT
....
0003               PLANT.ID
0004                       EMP.NAME
0005                               SAL.DED
....
....
0008                                      SAL.YEAR                ;;
```

Figure 10-18 Result of the Execution of the Commands in Figure 10-17

Entering an Incorrect Line Command



```
COMMAND: ED-15 * CONFLICTING LINE COMMANDS *           SCROLL: FULL
DB NAME: PLANT    INQUIRY NAME: EMPINFO
COMMENT: EMPLOYEE INFORMATION INQUIRY
0000 ***********          T O P    O F    D A T A      ***********
CC01   DISPLAY PLANT
CC02                PLANT.ID
0003                        EMP.NO
CC04                              EMP.NAME
0005                                      SAL.YTD
CC06                                             SAL.DED
A007                                                    SAL.YEAR  ;;
```

Figure 10-19 Result of Entering an Incorrect Line Command

Using Stored Functions

Note: Stored functions are not supported with the native SQL syntax facility of DB2 option.

You have stored the following calculation as a function in the directory with the name MON.SAL.

```
%MONTH.SAL = (SAL.YTD - SAL.DED) / 12
```

Call the stored function by using its assigned name in an inquiry statement.

```
DISPLAY PLANT PLANT.ID EMP.NAME        MON.SAL     IF SAL.YEAR = 95;
```

name of function

VISION:Inquiry automatically replaces the name of the function MON.SAL with the arithmetic calculation and performs the indicated arithmetic operations when it executes the DISPLAY statement.

Example: "List the employees and their monthly salaries by plant identification if they worked during 1995. Use the stored function in the inquiry statement."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL IF SAL.YEAR = 95;



PLANT.ID  EMP.NAME                             MONTH.SAL
10100     WILLIAM AMES                        4,583.3333
          PHYLLIS LOCKMEYER                   4,233.3333
          MARY ANN THOMAS                     1,185.8333
20150     WILMA FORD                          1,401.6666
          CHARLES SALTER                      2,750.0000
          PETER ZATKIN                        3,868.3333
          SUSAN WARE                          2,833.3333
30200     JOHN HENRY CRANE                    1,616.6666
          FREDERICH GRAY                      3,783.3333
          MITCHELL J HOOPS                    6,500.1666
          JANE LOWELL                         1,683.3333
          PATRICIA BLAKELY                    2,156.6666
          SHARON DALEY                        1,286.6666
40300     DONALD M KING                       4,800.0000
          JOAN EVANS                          4,800.0000
50300     JONATHAN OAKS                       2,996.6666
          MADELYN BATES                       2,185.0000
          VICKY WARD                          1,466.6666
```

Figure 10-20 Calling a Stored Function to an Inquiry

As Figure 10-20 illustrates, the results of the calculations are listed under the column heading MONTH.SAL.

Using More Than One Stored Function

The following inquiry statement uses the two stored functions, MON.SAL and BONUS:

```
DISPLAY PLANT PLANT.ID EMP.NAME    MON.SAL  BONUS   IF SAL.YEAR = 95;
                                       |                  |
                                       |      stored      |
                                       |     functions    |
```

You already know what has been defined as MON.SAL. The following has been stored under the name BONUS.

```
SAL.YTD + 10
```

Example: "List the employees, their monthly salaries, and bonuses by plant identification if they worked during 1995. Use two stored functions that perform these calculations."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL BONUS IF SAL.YEAR = 95;


PLANT.ID EMP.NAME                      MONTH.SAL                      .
10100    WILLIAM AMES                 4,583.3333              64,010.00
         PHYLLIS LOCKMEYER            4,233.3333              59,010.00
         MARY ANN THOMAS              1,185.8333              15,610.00
20150    WILMA FORD                   1,401.6666              18,810.00
         CHARLES SALTER               2,750.0000              39,010.00
         PETER ZATKIN                 3,868.3333              56,010.00
         SUSAN WARE                   2,833.3333              41,010.00
30200    JOHN HENRY CRANE             1,616.6666              22,010.00
         FREDERICH GRAY               3,783.3333              59,010.00
         MITCHELL J HOOPS             6,500.1666              98,010.00
         JANE LOWELL                  1,683.3333              26,010.00
         PATRICIA BLAKELY             2,156.6666              30,010.00
         SHARON DALEY                 1,286.6666              17,010.00
40300    DONALD M KING                4,800.0000              75,010.00
         JOAN EVANS                   4,800.0000              73,210.00
50300    JONATHAN OAKS                2,996.6666              46,010.00
         MADELYN BATES                2,185.0000              32,010.00
         VICKY WARD                   1,466.6666              20,010.00
```

Figure 10-21 Using Two Stored Functions to an Inquiry

In Figure 10-21, there is no column heading for BONUS because the results of the arithmetic calculation were not assigned to a temporary field when the function was defined. Therefore, the column heading appears as a period.

Using a Stored Inquiry that Calls a Stored Function

Stored inquiries also use stored functions. VISION:Inquiry copies the steps in the functions into the inquiry as it is stored.

For example, PAYROLL has been defined as:

```
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL IF SAL.YEAR = 95;
```

To execute the inquiry, enter:

```
PAYROLL PLANT;
```

The steps in the function are performed when the stored inquiry is executed.

Example: “List the employees and their monthly salaries by plant identification.”

```
PAGE:      TRANCODE: II       INQUIRY:
PAYROLL PLANT;


PLANT.ID  EMP.NAME                            MONTH.SAL
10100     WILLIAM AMES                       4,583.3333
          PHYLLIS LOCKMEYER                  4,233.3333
          MARY ANN THOMAS                    1,185.8333
20150     WILMA FORD                         1,401.6666
          CHARLES SALTER                     2,750.0000
          PETER ZATKIN                       3,868.3333
          SUSAN WARE                         2,833.3333
30200     JOHN HENRY CRANE                   1,616.6666
          FREDERICH GRAY                     3,783.3333
          MITCHELL J HOOPS                   6,500.1666
          JANE LOWELL                        1,683.3333
          PATRICIA BLAKELY                   2,156.6666
          SHARON DALEY                       1,286.6666
40300     DONALD M KING                      4,800.0000
          JOAN EVANS                         4,800.0000
50300     JONATHAN OAKS                      2,996.6666
          MADELYN BATES                      2,185.0000
          VICKY WARD                         1,466.6666
```

Figure 10-22 Using a Stored Inquiry that Calls a Function

Figure 10-22 shows that by using a simple entry at the terminal, an entire inquiry (including stored functions) can be called and executed.

Using a Stored Inquiry with Two Stored Functions and a Substitutable Value

The stored inquiry, SALARY.BONUS, is defined as:

```
DISPLAY PLANT PLANT.ID EMP.NAME MON.SAL BONUS IF SAL.YEAR = %1;
```

SALARY.BONUS contains two stored functions, MON.SAL and BONUS, and a substitutable value. The exact value of SAL.YEAR is indicated when the inquiry is executed.

SALARY.BONUS	PLANT	95 ;
stored inquiry that calls in two stored functions	stored in directory for this database	substitutable value

Example: "List the employees, their monthly salaries, and bonuses if they worked during 1995."

```
PAGE:      TRANCODE: II   INQUIRY: SALARY.BONUS PLANT 95;;


PLANT.ID EMP.NAME                    MONTH.SAL                      .
10100    WILLIAM AMES               4,583.3333             64,010.00
         PHYLLIS LOCKMEYER          4,233.3333             59,010.00
         MARY ANN THOMAS            1,185.8333             15,610.00
20150    WILMA FORD                 1,401.6666             18,810.00
         CHARLES SALTER             2,750.0000             39,010.00
         PETER ZATKIN               3,868.3333             56,010.00
         SUSAN WARE                 2,833.3333             41,010.00
30200    JOHN HENRY CRANE           1,616.6666             22,010.00
         FREDERICH GRAY             3,783.3333             59,010.00
         MITCHELL J HOOPS           6,500.1666             98,010.00
         JANE LOWELL                1,683.3333             26,010.00
         PATRICIA BLAKELY           2,156.6666             30,010.00
         SHARON DALEY               1,286.6666             17,010.00
40300    DONALD M KING              4,800.0000             75,010.00
         JOAN EVANS                 4,800.0000             73,210.00
50300    JONATHAN OAKS              2,996.6666             46,010.00
         MADELYN BATES              2,185.0000             32,010.00
         VICKY WARD                 1,466.6666             20,010.00
```

Figure 10-23 Using a Stored Inquiry with Two Stored Functions and a Substitutable Value

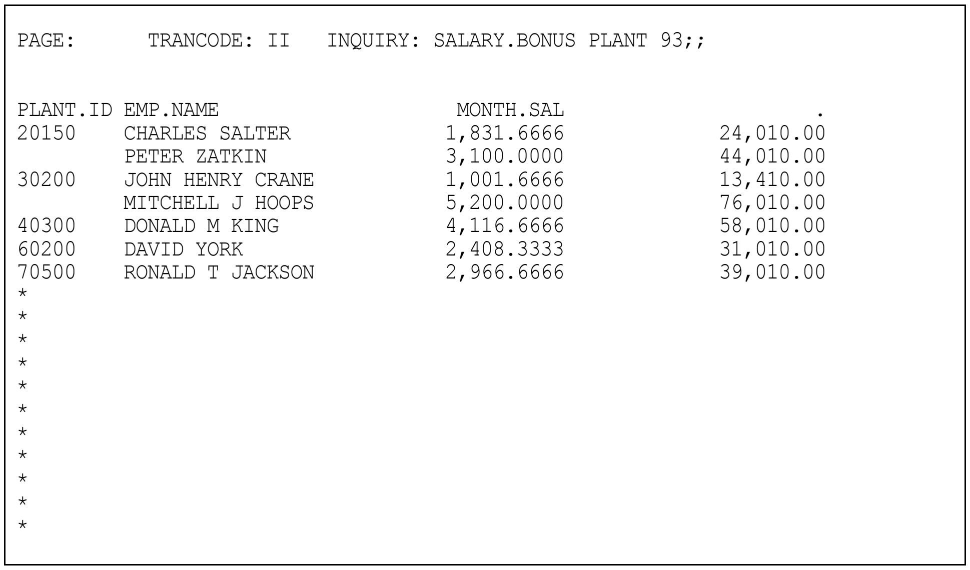
VISION:Inquiry substitutes '95' for %1, displays the requested fields, executes the steps in the two functions, and displays the results (see Figure 10-23).

Substituting a Value in a Stored Inquiry that Calls Two Functions

If you want to see information on people who worked during 1993, enter the appropriate substitutable value when you execute the stored inquiry.

```
SALARY.BONUS PLANT 93
```

Example: "List the employees, their monthly salaries, and bonuses if they worked during 1993."



```
PAGE:      TRANCODE: II   INQUIRY: SALARY.BONUS PLANT 93;;


PLANT.ID EMP.NAME                    MONTH.SAL                     .
20150    CHARLES SALTER             1,831.6666             24,010.00
         PETER ZATKIN               3,100.0000             44,010.00
30200    JOHN HENRY CRANE           1,001.6666             13,410.00
         MITCHELL J HOOPS           5,200.0000             76,010.00
40300    DONALD M KING              4,116.6666             58,010.00
60200    DAVID YORK                 2,408.3333             31,010.00
70500    RONALD T JACKSON           2,966.6666             39,010.00
*
*
*
*
*
*
*
*
*
*
*
```

Figure 10-24 Substituting a Value in a Stored Inquiry that Calls Two Functions

VISION:Inquiry substitutes '93' for %1, executes the steps in the two functions, and displays a different report. Compare Figure 10-23 with Figure 10-24.

Using Stored Functions Summary

A stored function is executed in one of two ways — either by adding the assigned name of the function to an inquiry statement, or by executing a stored inquiry that calls in the function.

When the name of a stored function is added to an inquiry statement, the results are treated like a field. On the output, the temporary field name of the function becomes the column heading.

Chapter

11 User Defined Output

You use the User Defined Output (UDO) to format output:

- Control the placement of data in the output of your inquiries.
- Define titles, column headings, and labels, and allows you to set their position on the output.
- Print the system TIME, DATE, and PAGE fields.
- Edit numeric fields with specifications for suppression of zeroes and placement of a decimal point and commas.
- Display part of a field.
- Limit the occurrences of output using the UDO REPEAT command.

This chapter presents an overview of UDO and describes how to write a UDO inquiry using the FORMAT command modifier. Examples are provided.

The UDO feature of VISION:Inquiry does not support native SQL syntax inquiries. However, VISION:Inquiry automatically formats the output of native SQL syntax inquiries.

Overview of UDO Functions

This chapter discusses titling a report, the horizontal and vertical movement commands, the REPEAT command, the partial fielding command, grand summaries, the EDIT command, arithmetic processing, accessing multiple databases, and stored inquiries and functions.

The section UDO Summary on page 11-47 reviews all of the UDO commands and defines the parameters of each command.

Using the FORMAT Command Modifier

You use a command modifier to tell VISION:Inquiry to arrange your output according to your specifications, and UDO produces the formatted report. The command modifier is FORMAT. It is used in conjunction with the DISPLAY command.

DISPLAY	PLANT	FORMAT
command	database name	command modifier

Note: Use the LINE and SKIP specifications together carefully. Incorrect use can produce undesirable results.

The FORMAT command modifier is followed by a series of other commands that tell VISION:Inquiry where to display the output on the report. These commands are:

COL (or **COLUMN**)	Displays the data in the specified column.
EDIT	Defines the specifications for editing a numeric field.
LINE	Controls the movement of data down the page by specific line number.
NOSP (or **NOSPACE**)	Controls the default spacing between output fields.
PF	Displays part of a character or user field.
REP (or **REPEAT)**	Limits the occurrences of data output to the screen.
SKIP	Controls the movement of data down the page by skipping lines.
SP (or **SPACE**)	Positions the data in the space +1 specified by the integer used with the command.

All of these commands are discussed and used in examples in this chapter.

What is the Syntax of a UDO Inquiry?

The syntax of the UDO inquiry is a slightly modified version of the basic inquiry to accommodate the UDO commands.

Formatted inquiries are composed of one or more statements that provide VISION:Inquiry with information about what database to use, what data to select from the database, how to summarize, edit, or limit that data, and how to display the data on the output screen or page. UDO inquiries consist of commands, database names, and field names. They also may include character or numeric constants, relational and logical operations, and noise words.

The following example is a typical UDO inquiry statement. It tells VISION:Inquiry to display the names and plant locations of the women who worked for the toy company during 1994. It also specifies how that information is to be output on the screen or page.

```
DISPLAY PLANT FORMAT
LINE 1 SP 16 'TOY COMPANY ROSTER'
SKIP 'PLANT' COL 30 'WOMEN'
SKIP 2 PLANT.ID COL 25 EMP.NAME
IF EMP.SEX = 'F' AND SAL.YEAR = 94;
```

Below is an explanation of how the inquiry works.

`DISPLAY`	The command.
`PLANT`	The database to be accessed.
`FORMAT`	The primary command to invoke UDO.
`LINE 1`	Begin the output on line 1 of the output section of the display screen.
`SP 16`	Count 16 positions from the left margin and begin displaying the output in the next position.
`’TOY COMPANY ROSTER’`	These three words are a character constant and are enclosed in single quotation marks; this becomes the title of your report. Any text may be used as a character constant. Character constants are defined and discussed in Chapter 3, “Structure and Concept of VISION:Inquiry”.
`SKIP`	Output on the next line.
`’PLANT’`	Another character constant. This becomes one of your column headings. Because you have not requested otherwise, this is output at the left margin of the screen.
`COL 30`	Begin display of the next data item at screen position 30.

`'WOMEN'`	Another character constant; it becomes a column heading.
`SKIP 2`	Leave one line before beginning the output on the next line.
`PLANT.ID`	A field from the PLANT database that you want displayed.
`COL 25`	Begin display of the next data item or constant at screen position 25.
`EMP.NAME`	The next field that you want from the PLANT database.
`IF EMP.SEX = 'F' AND SAL.YEAR = 94`	Conditional selection phrases that select only specific data for processing.
`;`	Termination symbol.

Is UDO Compatible with Basic VISION:Inquiry?

Because UDO uses the following VISION:Inquiry commands in its inquiry statements, it is compatible with basic, or non-UDO, VISION:Inquiry.

AVERAGE	LIMIT
COUNT	OUTPUT
DEFINE	SORT
DISPLAY	TOTAL
FIND	

The LIMIT and OUTPUT commands have not been used in examples in this chapter. They are used in a UDO inquiry exactly as they are used in a non-UDO inquiry. See Chapter 5, "Simple Reports", if you have questions about these two commands.

How is the Screen Displayed?

To use basic, or non-UDO, VISION:Inquiry under IMS, you enter the Message Format Service (MFS) module:

```
/FORMAT INQIMS
```

To use UDO under IMS, you enter the MFS module:

```
/FORMAT INQUDO
```

IMS responds in the usual way by displaying the VISION:Inquiry screen. The special UDO MFS module provides you more space for your input lines.

There is no special module or command for accessing UDO under CICS. However, the VISION:Inquiry default map has 20 input lines and can be used for either UDO or non-UDO inquiries. Note that the output will limit the display of input lines to four lines. The last page of output (which contains the original inquiry) accepts 20 lines of input.

What Does the UDO Screen Look Like Under IMS?

As with basic VISION:Inquiry, the UDO screen is divided into two sections: the top portion is reserved for the inquiry and the bottom portion for the output. Because of the formatting specifications, UDO inquiries are usually longer than those used in a basic inquiry. More lines should be allocated to a UDO inquiry as shown in this guide.

See your system administrator to learn more about your terminal characteristics in VISION:Inquiry, how your system operates under UDO, and how it utilizes different screens.

Titling Your Report

To create a title, enclose any text within single quotation marks. The text is treated by VISION:Inquiry as a character constant. In Figure 11-1, the title of your report is 'TOY COMPANY ROSTER'. Although none of these words exist in the database, you may use them, or any others, if you treat them as character constants.

Example: "Create a titled listing of the toy company roster. Include employee name, identification number, and plant location in the listing."

```
PAGE:       TRANCODE: II          INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


  TOY COMPANY ROSTER


   PLANT           EMPLOYEE              EMPLOYEE
  NUMBER             NAME                 NUMBER

  10100          WILLIAM AMES             10103
                 PHYLLIS LOCKMEYER        10104
                 MARY ANN THOMAS          10105
  20150          WILMA FORD               21116
                 CHARLES SALTER           21124
                 PETER ZATKIN             21137
                 SUSAN WARE               21164
  30200          JOHN HENRY CRANE         30201
                 FREDERICH GRAY           30202
                 MITCHELL J HOOPS         30205
```

Figure 11-1 A UDO Report with a Title

In Figure 11-1, a title was added at the left margin of the screen.

The title could have been placed in the center of the display by using one of the horizontal movement commands. The horizontal movement commands and all other UDO commands are discussed on the following pages.

Centering the Title

Figure 11-2 illustrates that centering titles makes reports look more professional.

With UDO, you decide where you want your output placed, specify what you want in your inquiry, and let VISION:Inquiry do the rest.

```
PAGE:         TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


                  TOY COMPANY ROSTER


   PLANT           EMPLOYEE              EMPLOYEE
  NUMBER             NAME                 NUMBER

  10100          WILLIAM AMES             10103
                 PHYLLIS LOCKMEYER        10104
                 MARY ANN THOMAS          10105
  20150          WILMA FORD               21116
                 CHARLES SALTER           21124
                 PETER ZATKIN             21137
                 SUSAN WARE               21164
  30200          JOHN HENRY CRANE         30201
                 FREDERICH GRAY           30202
                 MITCHELL J HOOPS         30205
```

Figure 11-2 A UDO Report with a Centered Title

Including an Apostrophe in a Title

If you wish to have the title of your report read 'TOY COMPANY'S ROSTER', you would represent the apostrophe with two single quotation marks. After adding the "S", enclose the whole text of the character constant within single quotation marks.

```
PAGE:       TRANCODE: II          INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 18 'TOY COMPANY''S ROSTER'
SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


                 TOY COMPANY'S ROSTER


   PLANT           EMPLOYEE               EMPLOYEE
  NUMBER             NAME                  NUMBER

  10100          WILLIAM AMES              10103
                 PHYLLIS LOCKMEYER         10104
                 MARY ANN THOMAS           10105
  20150          WILMA FORD                21116
                 CHARLES SALTER            21124
                 PETER ZATKIN              21137
                 SUSAN WARE                21164
  30200          JOHN HENRY CRANE          30201
                 FREDERICH GRAY            30202
                 MITCHELL J HOOPS          30205
```

Figure 11-3 Changing the Title to 'Toy Company's Roster'

The two single quotation marks are entered on the second line of the inquiry. They tell VISION:Inquiry that a printable quotation mark has been entered, rather than a delimiting one. When VISION:Inquiry displays the output, one of the single quotation marks is dropped so that the title appears on the report as 'TOY COMPANY'S ROSTER'.

Horizontal Movement Commands - COL and SP

In the inquiries for the first three examples, titles and column headings are used. The output is positioned to appear at specific points on the display. To achieve this positioning, the horizontal movement commands COL and SP are used.

COL is always followed by an integer, as COL 42 in Figure 11-4. This directs VISION:Inquiry to begin the output 42 positions from the left margin of the screen.

```
PAGE:       TRANCODE: II           INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


                  TOY COMPANY ROSTER


  PLANT           EMPLOYEE              EMPLOYEE
 NUMBER             NAME                 NUMBER

 10100          WILLIAM AMES             10103
                PHYLLIS LOCKMEYER        10104
                MARY ANN THOMAS          10105
 20150          WILMA FORD               21116
                CHARLES SALTER           21124
                PETER ZATKIN             21137
                SUSAN WARE               21164
 30200          JOHN HENRY CRANE         30201
                FREDERICH GRAY           30202
                MITCHELL J HOOPS         30205
```

Figure 11-4 Using the Horizontal Movement Commands

In Figure 11-4, SP 18 positions the title 18 spaces from the left margin of the screen and begin the output in the next position. Thus, the command SP 18 actually begins the display in the 19th position (or n + 1).

The commands SP 3 and SP 11, on the third line of the inquiry, count 3 and 11 spaces, respectively, from the end of the last entry and begin the output in the next position.

In this display, column headings were used above each field of output. Each column heading was treated as a character constant. Any text may be used as a column heading as long as it is treated as a constant and enclosed in single quotation marks.

Using an integer with the SP command is optional.

- If the integer is omitted from the SP command in the middle of an inquiry, VISION:Inquiry defaults to one space between the end of one field or column heading and the beginning of another.
- The only exception to this is when the integer is omitted from the SP command at the beginning of a line. In this event, VISION:Inquiry displays the data at the left margin of the screen. LINE LINE causes the output to skip two lines (as shown in Figure 11-5).

Using SP without an Integer

```
PAGE:       TRANCODE: II         INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 18 'TOY COMPANY ROSTER'
SKIP 2 SP   'PLANT' SP   'EMPLOYEE' COL 42 'EMPLOYEE'
LINE SP   'NUMBER' SP   'NAME' COL 43 'NUMBER'
LINE LINE SP   PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


                 TOY COMPANY ROSTER


PLANT EMPLOYEE                          EMPLOYEE
NUMBER NAME                              NUMBER

10100           WILLIAM AMES             10103
                PHYLLIS LOCKMEYER        10104
                MARY ANN THOMAS          10105
20150           WILMA FORD               21116
                CHARLES SALTER           21124
                PETER ZATKIN             21137
                SUSAN WARE               21164
30200           JOHN HENRY CRANE         30201
                FREDERICH GRAY           30202
                MITCHELL J HOOPS         30205
```

Figure 11-5 Using SP Without an Integer

By deleting the integer from the SP command in Figure 11-5, the column headings EMPLOYEE and NAME each defaulted to one space between the column headings PLANT and NUMBER.

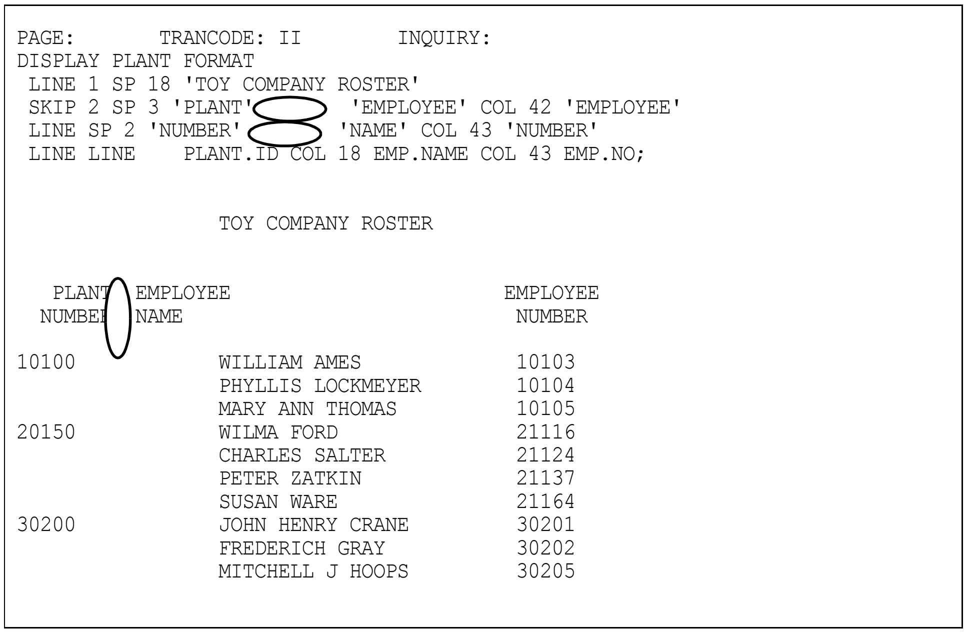
The integer was also deleted from the SP command on the last line of the inquiry. The PLANT.ID field was output at the left margin of the screen.

Omitting a Horizontal Movement Command in the Middle of an Inquiry

If you omit a horizontal movement command in the middle of an inquiry, VISION:Inquiry defaults to two spaces between the end of one field or character constant and the beginning of another.

Small horizontal ovals have been added to Figure 11-6 to illustrate the positions where horizontal movement commands have been omitted from the inquiry.

A small vertical oval shows where VISION:Inquiry has inserted double spaces between column headings as a result.



```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT'         'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER'        'NAME' COL 43 'NUMBER'
 LINE LINE    PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


                 TOY COMPANY ROSTER


   PLANT  EMPLOYEE                       EMPLOYEE
  NUMBER  NAME                            NUMBER

10100            WILLIAM AMES             10103
                 PHYLLIS LOCKMEYER        10104
                 MARY ANN THOMAS          10105
20150            WILMA FORD               21116
                 CHARLES SALTER           21124
                 PETER ZATKIN             21137
                 SUSAN WARE               21164
30200            JOHN HENRY CRANE         30201
                 FREDERICH GRAY           30202
                 MITCHELL J HOOPS         30205
```

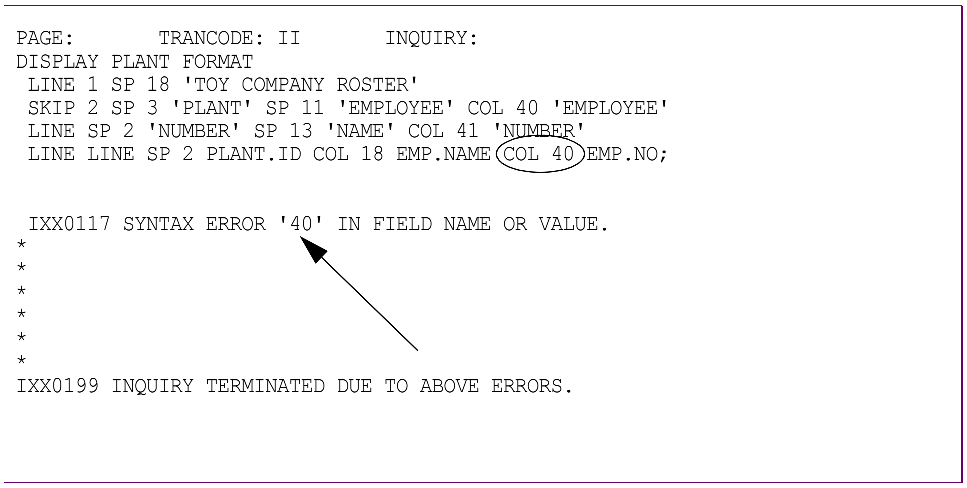
Figure 11-6 A Default of Two Positions When the Horizontal Movement Commands are Omitted from the Inquiry

In Figure 11-6, the SP command was omitted in front of EMPLOYEE on the second line and in front of NAME on the third line. The display shows these column headings output two spaces beyond the PLANT and NUMBER column headings. This is the normal default spacing between fields or character constants when horizontal movement commands are not specified in an inquiry.

The SP command was also omitted in front of the PLANT.ID field on the last line of the inquiry. When you compare Figure 11-5 to Figure 11-6, you will see that the PLANT.ID field begins in the same place in both displays.

Overlaping Fields

When positioning your fields across the screen, it is important that you know the length of each line to avoid overlapping. Figure 11-7 shows what happens when the fields in your inquiry overlap.



```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 40 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 41 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 40 EMP.NO;


 IXX0117 SYNTAX ERROR '40' IN FIELD NAME OR VALUE.
*
*
*
*
*
*
IXX0199 INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

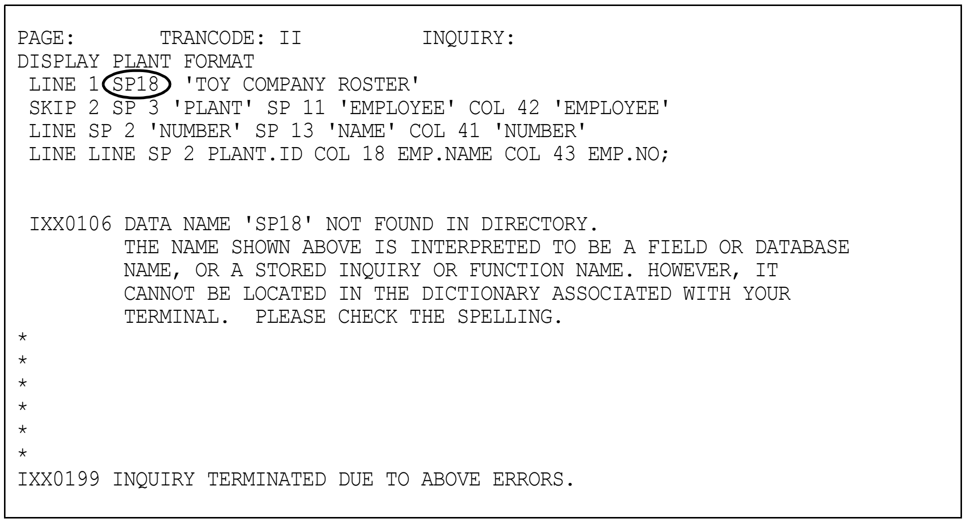
Figure 11-7   Error Message When a Field Overlaps Another Field

The field, EMP.NAME, has been defined to be 25 positions long. By positioning EMP.NAME in COL 18, the next field or character constant cannot be positioned until COL 43. Note that the inquiry format specification indicates that EMP.NO is to begin in COL 40, shown in the small oval added to Figure 11-7; hence, an overlap will occur. The erroneous value of ’40’ is returned in the error message, indicated by the arrow.

To avoid such errors, check with your system administrator for a listing of field lengths before you use UDO.

No Space Between the Command and the Integer

In Figure 11-8, there is no space between the SP command and the integer 18, so the system treats SP18 as one word. Figure 11-8 shows the error message you receive if you fail to leave at least one blank space between the command and the integer.



```
PAGE:       TRANCODE: II          INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP18  'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 41 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;


 IXX0106 DATA NAME 'SP18' NOT FOUND IN DIRECTORY.
         THE NAME SHOWN ABOVE IS INTERPRETED TO BE A FIELD OR DATABASE
         NAME, OR A STORED INQUIRY OR FUNCTION NAME. HOWEVER, IT
         CANNOT BE LOCATED IN THE DICTIONARY ASSOCIATED WITH YOUR
         TERMINAL.  PLEASE CHECK THE SPELLING.
*
*
*
*
*
*
IXX0199 INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

Figure 11-8 Error Message When Space is Omitted Between Command and Integer

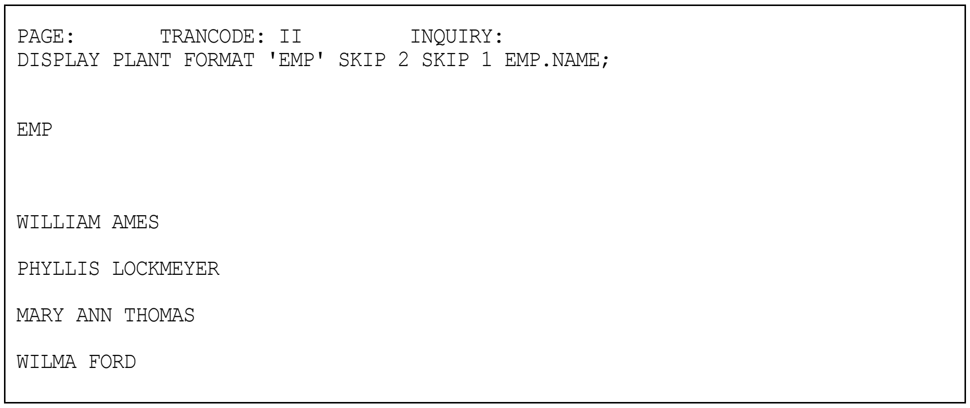
Vertical Movement Commands

Note: When using the LINE and SKIP specifications together, undesirable results can occur if used incorrectly.

The vertical movement of the page is controlled by the commands LINE and SKIP.

- When SKIP is used with an integer, specify the number of lines (minus one) to omit before printing the next line of output.
- When LINE is used with an integer, specify the exact screen line number where the data is to be output. Remember that line numbers correspond to the output section of the display screen.
- When the integer is omitted from LINE, VISION:Inquiry outputs the data on the following line of the display.

Using the SKIP Command to Add Blank Lines



```
PAGE:       TRANCODE: II         INQUIRY:
DISPLAY PLANT FORMAT 'EMP' SKIP 2 SKIP 1 EMP.NAME;


EMP



WILLIAM AMES

PHYLLIS LOCKMEYER

MARY ANN THOMAS

WILMA FORD
```

Figure 11-9 Additional Vertical Spacing (Blank Lines) After the Title

The inquiry “DISPLAY PLANT FORMAT ’EMP’ SKIP 2 SKIP 1 EMP.NAME” prints the first employee’s name with skip of 3 (2 + 1) and the others with skip of 1. This is used when additional space is desired between the first employee name and the title.

Using LINE and SKIP

The LINE or SKIP command directly preceding a group of repeated items applies to each occurrence of the highest-level item in the group. A default of next line is assumed for all lower-level items in the group of repeated items.

In Figure 11-10, the commands LINE and SKIP moved the output down the page. The command LINE LINE leaves one extra blank line, but only for the first record of PLANT.ID.

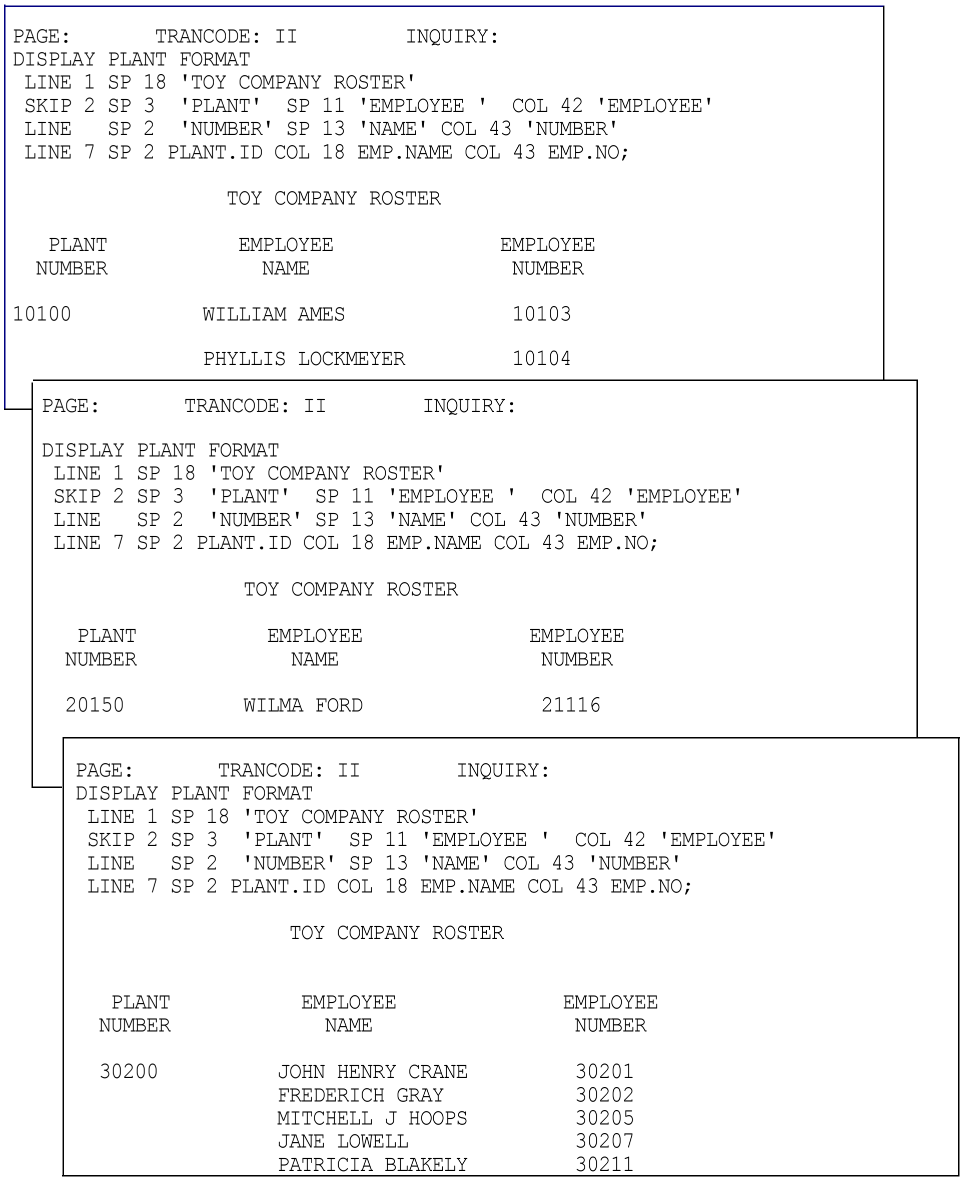
```
PAGE:          TRANCODE: II          INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3 'PLANT' SP 11 'EMPLOYEE' COL 42 'EMPLOYEE'
 LINE SP 2 'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;

                  TOY COMPANY ROSTER

   PLANT           EMPLOYEE              EMPLOYEE
  NUMBER             NAME                 NUMBER

  10100          WILLIAM AMES             10103
                 PHYLLIS LOCKMEYER        10104
                 MARY ANN THOMAS          10105
  20150          WILMA FORD               21116
                 CHARLES SALTER           21124
                 PETER ZATKIN             21137
                 SUSAN WARE               21164
  30200          JOHN HENRY CRANE         30201
                 FREDERICH GRAY           30202
                 MITCHELL J HOOPS         30205
```

Figure 11-10 Using LINE and SKIP for Vertical Movement

Using LINE as a Page Break

To have the employees from each plant output to a separate page, specify the line number where you want the data output in the inquiry.

Example: “List the data from the toy company roster printed on a separate page for each plant.”



```
PAGE:       TRANCODE: II         INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3  'PLANT'  SP 11 'EMPLOYEE '  COL 42 'EMPLOYEE'
 LINE   SP 2  'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE 7 SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;

                  TOY COMPANY ROSTER

   PLANT            EMPLOYEE               EMPLOYEE
  NUMBER              NAME                  NUMBER

10100            WILLIAM AMES               10103

                 PHYLLIS LOCKMEYER          10104
```

```
PAGE:       TRANCODE: II        INQUIRY:

DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3  'PLANT'  SP 11 'EMPLOYEE '  COL 42 'EMPLOYEE'
 LINE   SP 2  'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE 7 SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;

                 TOY COMPANY ROSTER

   PLANT            EMPLOYEE               EMPLOYEE
  NUMBER              NAME                  NUMBER

  20150          WILMA FORD                 21116
```

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
 LINE 1 SP 18 'TOY COMPANY ROSTER'
 SKIP 2 SP 3  'PLANT'  SP 11 'EMPLOYEE '  COL 42 'EMPLOYEE'
 LINE   SP 2  'NUMBER' SP 13 'NAME' COL 43 'NUMBER'
 LINE 7 SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;

                 TOY COMPANY ROSTER


   PLANT            EMPLOYEE               EMPLOYEE
  NUMBER              NAME                  NUMBER

  30200          JOHN HENRY CRANE           30201
                 FREDERICH GRAY             30202
                 MITCHELL J HOOPS           30205
                 JANE LOWELL                30207
                 PATRICIA BLAKELY           30211
```

Figure 11-11   Using LINE as a Page Break

By explicitly specifying LINE 7, VISION:Inquiry outputs each plant location on line 7 of a separate page. Another method to output each plant location on a separate page is to use the PLANT.ID field as part of the report title area. For example:

```
LINE 1 SP 5 'TOY COMPANY ROSTER FOR PLANT NUMBER' PLANT.ID
```

Such explicit specification uses the command as a page break.

When fields from different levels are specified on the same line as an explicit line number, the page break is controlled on the highest level. It is recommended that you specify the explicit line number for the highest level segment in your inquiry statement as follows.

```
LINE 7 SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO
```

You receive an error message if your highest level segment does not have an explicit line number but your lower level segment does.

Likewise, it is recommended that you use SKIP to position your summaries any number of lines below your data items.

Page Breaks Affect Printing of Literals

Forcing page breaks through the use of explicit line numbers also affects the printing of literals. For instance, the following inquiry causes the literal GRAND TOTAL to be printed on each page even though the grand total value is only on the last page.

```
DISPLAY PLANT FORMAT
LINE 5 'EMPLOYEE:' EMP.NO EMP.NAME
SKIP SAL.YEAR SAL.YTD
SKIP SKIP 'GRAND TOTAL' TOTAL SAL.YTD;
```

Using DATE, DATEF, TIME, and PAGE

DATE, DATEF, TIME, and PAGE may be added to your output. These are treated as fields; therefore, they are not enclosed within single quotation marks as character constants are.

- DATE supplies the current date in the form MM/DD/YY.
- DATEF supplies the current date in the form MM/DD/YYYY.
- TIME supplies the current time in the form HH:MM:SS.
- PAGE supplies the page numbers for the inquiry.

DATE, DATEF, TIME, and PAGE are available for display only; they cannot be used in a conditional selection phrase.

Figure 11-12 shows these fields added to the toy company roster inquiry.

Example: "Produce a listing of the toy company roster, including the time and date it was compiled. Also, add page numbers."

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 DATE COL 18 'TOY COMPANY ROSTER' SP 4 TIME COL 57 PAGE
SKIP 2 SP 3  'PLANT' SP 11 'EMPLOYEE'  COL 42 'EMPLOYEE'
LINE SP 2  'NUMBER' SP 13 'NAME'  COL 43 'NUMBER'
LINE LINE SP 2 PLANT.ID COL 18 EMP.NAME COL 43 EMP.NO;

01/15/2002         TOY COMPANY ROSTER    15:37:53        1

   PLANT           EMPLOYEE              EMPLOYEE
  NUMBER             NAME                 NUMBER

  10100          WILLIAM AMES             10103
                 PHYLLIS LOCKMEYER        10104
                 MARY ANN THOMAS          10105
  20150          WILMA FORD               21116
                 CHARLES SALTER           21124
                 PETER ZATKIN             21137
                 SUSAN WARE               21164
  30200          JOHN HENRY CRANE         30201
                 FREDERICH GRAY           30202
                 MITCHELL J HOOPS         30205
```

Figure 11-12 Using DATE, TIME, and PAGE in a Formatted Inquiry

The DATE, TIME, and PAGE were added to the first line of the inquiry. If you replace the DATE field with the DATEF field, the date format will display as: 01/15/2002.

The page number prints consecutively on each page of the report.

Customizing Output Format

The UDO function not only allows you to format your output to meet your needs; it also allows you to enhance your output to improve readability. Figure 11-13 illustrates how to do this.

Example: “Produce a listing of the employees in plant location 10100. Include their names, salaries and educational background in the listing. Because this is a top priority report, make it stand out in some way.”

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 '**' COL 66 '**'
LINE SP 5 '***'  SP 12 '**TOP PRIORITY REPORT**' SP 5 '***'
LINE SP 15 '++ LISTING OF EMPLOYEES IN PLANT 10100 ++'
LINE COL 10 'NAME OF' COL 40 'SALARIES' COL 55 'EDUCATION'
LINE COL 9 'EMPLOYEE' COL 41 'EARNED' COL 56 'ACHIEVED'
LINE COL 9 '********' COL 40 '********' COL 55 '********'
SKIP COL 2 EMP.NAME COL 36 SAL.YTD COL 58 ED.DEGREE IF PLANT.ID = 10100;


**                                                               **
     ***            **TOP PRIORITY REPORT**     ***
               ++ LISTING OF EMPLOYEES IN PLANT 10100 ++
         NAME OF                     SALARIES       EDUCATION
        EMPLOYEE                      EARNED         ACHIEVED
        ********                     ********       ********
WILLIAM AMES                   52,000.00          BA
                               64,000.00
PHYLLIS LOCKMEYER              48,000.00          BA
                               59,000.00
MARY ANN THOMAS                15,600.00          HS
```

Figure 11-13 Adding Embellishments to Your Output

The output stands out because the first line of its titles identifies it as ’TOP PRIORITY REPORT’. You may add symbols, characters, or numbers to your output by treating them as character constants and enclosing them within single quotation marks.

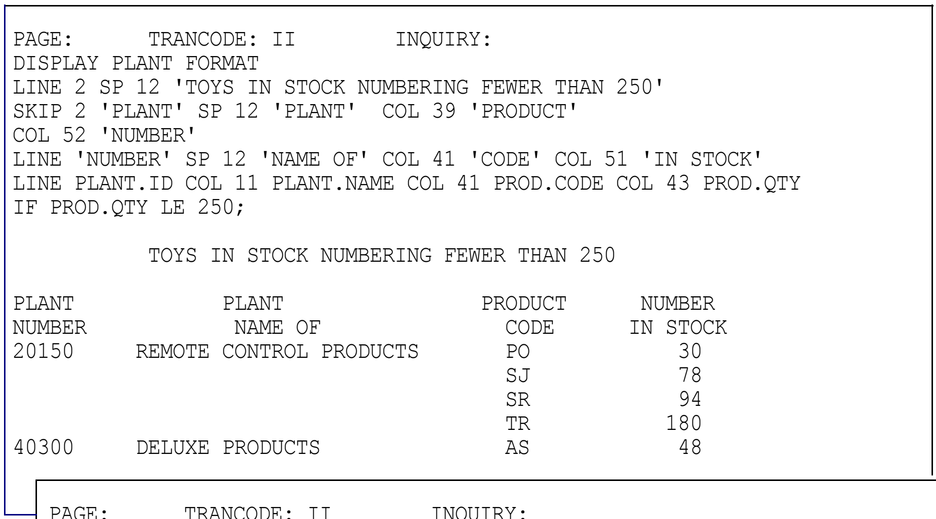
Conditional Selection

The last example used conditional selection to select specific data for processing. Output was limited to only those employees working in plant location 10100. As you can see, the conditional selection phrase is positioned in the last part of a UDO inquiry just as it is in a non-UDO inquiry.

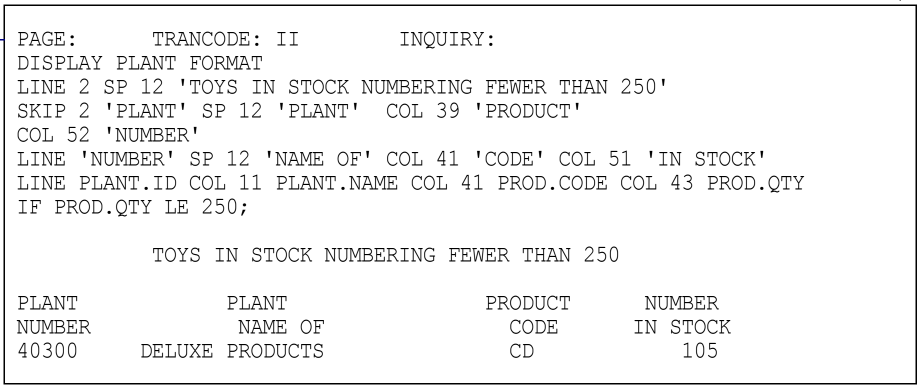
Using Conditional Selection in a Formatted Inquiry

The next example uses conditional selection to show you how to enter the same inquiry in two different ways.

Example: “How many toys are in stock that number 250 or less? Show the plant location and name where each toy is located. List the toy by product code. Make the output as easy to read as possible.”



```
PAGE:      TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE 2 SP 12 'TOYS IN STOCK NUMBERING FEWER THAN 250'
SKIP 2 'PLANT' SP 12 'PLANT'  COL 39 'PRODUCT'
COL 52 'NUMBER'
LINE 'NUMBER' SP 12 'NAME OF' COL 41 'CODE' COL 51 'IN STOCK'
LINE PLANT.ID COL 11 PLANT.NAME COL 41 PROD.CODE COL 43 PROD.QTY
IF PROD.QTY LE 250;

           TOYS IN STOCK NUMBERING FEWER THAN 250

PLANT            PLANT                 PRODUCT      NUMBER
NUMBER            NAME OF                CODE      IN STOCK
20150     REMOTE CONTROL PRODUCTS        PO           30
                                         SJ           78
                                         SR           94
                                         TR          180
40300     DELUXE PRODUCTS                AS           48
```



```
PAGE:      TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE 2 SP 12 'TOYS IN STOCK NUMBERING FEWER THAN 250'
SKIP 2 'PLANT' SP 12 'PLANT'  COL 39 'PRODUCT'
COL 52 'NUMBER'
LINE 'NUMBER' SP 12 'NAME OF' COL 41 'CODE' COL 51 'IN STOCK'
LINE PLANT.ID COL 11 PLANT.NAME COL 41 PROD.CODE COL 43 PROD.QTY
IF PROD.QTY LE 250;

           TOYS IN STOCK NUMBERING FEWER THAN 250

PLANT            PLANT                 PRODUCT      NUMBER
NUMBER            NAME OF                CODE      IN STOCK
40300     DELUXE PRODUCTS                CD          105
```

Figure 11-14   Using Conditional Selection in a Formatted Inquiry

In Figure 11-14, the title of the report appears on line 2 of the output. Although this created space between the inquiry and the output, making the output easier to read, it also shortened the output page because the title appears on line 13 rather than line 12, as is normal.

When you look at the inquiry in Figure 11-14, you see that COL 52 'NUMBER' was entered one line below the rest of the column headings with which it belongs. Because VISION:Inquiry ignores spaces at input time, it continued printing the character constant 'NUMBER' on the same line as the rest of the column headings. This is exactly what was wanted.

VISION:Inquiry with UDO always repeats the higher level segment fields specified in the DISPLAY statement at the top of a new page for clarity, as long as more than one segment or subtotal is used. This can be seen on the second page of Figure 11-14. The plant number "40300" and plant name "Deluxe Products" have been repeated because they are in higher level segments.

Entering the Same Inquiry Differently

```
 PAGE:       TRANCODE: II        INQUIRY:
 DISPLAY PLANT FORMAT  LINE 2 SP 12 'TOYS IN STOCK NUMBERING
 FEWER THAN 250' SKIP 2 'PLANT' SP 12 'PLANT'  COL 39 'PRODUCT' COL 52
'NUMBER' LINE 'NUMBER' SP 12 'NAME OF' COL 41 'CODE' COL 51 'IN STOCK'
 LINE PLANT.ID COL 11 PLANT.NAME COL 41 PROD.CODE COL 43 PROD.QTY IF
 PROD.QTY LE 250;

             TOYS IN STOCK NUMBERING FEWER THAN 250


 PLANT            PLANT                 PRODUCT      NUMBER
 NUMBER            NAME OF                CODE      IN STOCK
 20150     REMOTE CONTROL PRODUCTS        PO           30
                                          SJ           78
                                          SR           94
                                          TR          180
 40300     DELUXE PRODUCTS                AS           48
                                          CD          105
```

Figure 11-15 Entering the Same Inquiry from Figure 11-14 Differently

When you look at the inquiry in Figure 11-15, you can see that it is the same inquiry as the previous one. However, the inquiry was entered differently.

The inquiry in Figure 11-15 has been entered as a continuous stream of input. You may enter inquiries this way because VISION:Inquiry has been programmed to respond to the LINE and SKIP commands. When it sees either command, it begins printing on a new line and treats what follows as a unit belonging together, until it reads another command.

Using SP Instead of COL

Be careful not to confuse COL with SP when you enter your inquiries. Had SP 52 ’NUMBER’ been entered instead of COL 52 ’NUMBER’, VISION:Inquiry would have tried to count 52 positions from the last entry. Because the last entry began in COL 39, the output would have run off the screen.

Figure 11-16 shows the error message that would have resulted.

```
PAGE:       TRANCODE: II      INQUIRY:
DISPLAY PLANT FORMAT
LINE 2 SP 12 'TOYS IN STOCK NUMBERING FEWER THAN 250'
SKIP 2 'PLANT' SP 12 'PLANT'  COL 39 'PRODUCT'
SP 52 'NUMBER'
LINE 'NUMBER' SP 12 'NAME OF' COL 41 'CODE' COL 51 'IN STOCK'
LINE PLANT.ID COL 11 PLANT.NAME COL 41 PROD.CODE COL 43 PROD.QTY
IF PROD.QTY LE 250;


IXX0820 FORMAT LINE SIZE EXCEEDS TERMINAL LINE SIZE.
*
*
*
*
*
*
IXX0199 INQUIRY TERMINATED DUE TO ABOVE ERRORS.
```

Figure 11-16 Error Message When SP is Used Instead of COL and the Line Size Exceeds the Defined Width of the Terminal

REPEAT Command

The REPEAT or REP command specifies the number of segment occurrences displayed from the database. To limit the number of calls to the database, use the LIMIT command.

The REPEAT or REP command does not stop the database scan. It limits output to the screen.

The REP command is always used with an integer and parentheses. The integer, or repetition number, indicates the maximum number of times that the data specified from the leftmost highest level segment on the database within the parentheses is to be repeated. The parentheses define the range of the control of the REP command.

```
REP                   3              (PLANT.ID       EMP.NAME   SAL.YTD)
```

command	maximum number of times data is repeated	highest-level field in parentheses	lower-level fields accessed

Because VSAM files and DB2 tables are treated as a single segment database, the integer following the REPEAT command indicates the number of lines displayed at the output.

Using the REPEAT Command

Example: "Show a listing of employees and their salaries. Limit the listing to two plants."

```
PAGE:      TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE REP 2 (PLANT.ID EMP.NAME SAL.YTD);



10100  WILLIAM AMES                  52,000.00
                                     64,000.00
       PHYLLIS LOCKMEYER             48,000.00
                                     59,000.00
       MARY ANN THOMAS               15,600.00
20150  WILMA FORD                    15,600.00
                                     18,800.00
       CHARLES SALTER                24,000.00
                                     30,000.00
                                     39,000.00
       PETER ZATKIN                  44,000.00
```

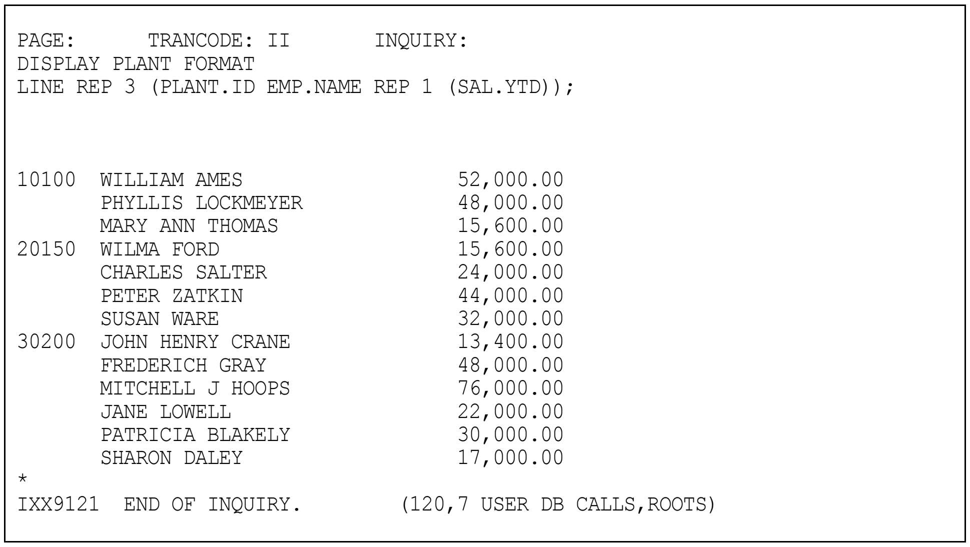
Figure 11-17   Using the REPEAT Command

In Figure 11-17, the limit action was applied to the highest level segment referenced from the database. In the inquiry, this segment was PLANT.ID. Therefore, two plant locations were displayed with the employees of each and their salaries.

Using Two REPEAT Commands

Limiting control may be placed on all levels of the hierarchical structure by using nested REP commands.

Example: “Show a listing of employees and the first salaries they received. Limit the listing to three plant locations.”



```
PAGE:      TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE REP 3 (PLANT.ID EMP.NAME REP 1 (SAL.YTD));



10100  WILLIAM AMES                  52,000.00
       PHYLLIS LOCKMEYER             48,000.00
       MARY ANN THOMAS               15,600.00
20150  WILMA FORD                    15,600.00
       CHARLES SALTER                24,000.00
       PETER ZATKIN                  44,000.00
       SUSAN WARE                    32,000.00
30200  JOHN HENRY CRANE              13,400.00
       FREDERICH GRAY                48,000.00
       MITCHELL J HOOPS              76,000.00
       JANE LOWELL                   22,000.00
       PATRICIA BLAKELY              30,000.00
       SHARON DALEY                  17,000.00
*
IXX9121  END OF INQUIRY.        (120,7 USER DB CALLS,ROOTS)
```

Figure 11-18 Using Two REPEAT Commands

The first REP command applies to the PLANT.ID and EMP.NAME fields and outputs three plant locations and all associated employees at each. The second REP command within the nested parentheses limits the salary field to one per employee. The first occurrence of this field is displayed.

Using REPEAT Commands with Conditional Selection

If you wanted to see the last occurrence of this field displayed, you would write the inquiry as seen in Figure 11-19.

Example: “Show a listing of employees and the last salaries they received. Limit the listing to the same three plants as in the previous report.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE REP 3 (PLANT.ID EMP.NAME REP 1 (SAL.YTD))
IF SAL.YTD LAST;


10100  WILLIAM AMES                 64,000.00
       PHYLLIS LOCKMEYER            59,000.00
       MARY ANN THOMAS              15,600.00
20150  WILMA FORD                   18,800.00
       CHARLES SALTER               39,000.00
       PETER ZATKIN                 56,000.00
       SUSAN WARE                   41,000.00
30200  JOHN HENRY CRANE             22,000.00
       FREDERICH GRAY               59,000.00
       MITCHELL J HOOPS             98,000.00
       JANE LOWELL                  26,000.00
       PATRICIA BLAKELY             30,000.00
       SHARON DALEY                 17,000.00
*
IXX9121  END OF INQUIRY.           (119,7 USER DB CALLS,ROOTS)
```

Figure 11-19 Using Two REPEAT Commands and Conditional Selection

In the last three examples, the REP command was applied to the root segment of the database.

Using the REPEAT Command on Dependent Segments

In the next example, the REP command is applied to a dependent segment only.

Example: “Show a listing of employees, their salaries, and educational backgrounds. Limit the output to six employees and two degrees.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE REP 6 (EMP.NAME SAL.YTD REP 2 (ED.DEGREE));



WILLIAM AMES                  52,000.00   BA
                              64,000.00
PHYLLIS LOCKMEYER             48,000.00   BA
                              59,000.00
MARY ANN THOMAS               15,600.00   HS
WILMA FORD                    15,600.00   HS
                              18,800.00
CHARLES SALTER                24,000.00   BA
                              30,000.00
                              39,000.00
PETER ZATKIN                  44,000.00   BA
                              50,000.00   MA
                              56,000.00
*
IXX9121  END OF INQUIRY.         (180,7 USER DB CALLS,ROOTS)
```

Figure 11-20   Using the REPEAT Command on Dependent Segments

The first REP command was applied to the EMP.NAME field. The first six employees were displayed from the database along with the salaries they earned. The second REP command limited the output to two degrees. Only Peter Zatkin met this requirement.

Partial Fielding (PF) Command

You use the partial fielding (PF) command to process any part of a character or user field.

A user field (defined as TYPE= U or 1 - 9 in the MAPGEN) provides a special capability within VISION:Inquiry to process data fields that are not formatted in a standard data format. See your system administrator for further information about user fields.

When you use partial fielding, you must specify the start location within the specified data field and the length of the partial field. This command may not be used as part of a conditional selection phrase.

This is how to write a partial fielding command.

PF	(PLANT.PHONE	1	3)
command	data field to be partial fielded	start location	length of partial field

Example: “Create a listing of the plant phone numbers. Include plant names and regions in the display. To make the numbers easy to read, separate the first three digits from the last four. Arrange the listing alphabetically by plant name.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 18 '- PLANT TELEPHONE NUMBERS -'
SKIP 2 SP 6 'PLANT' COL 35 'PLANT' COL 46 'PHONE'
LINE  SP 7 'NAME' COL 34 'REGION' COL 45 'NUMBER'
SKIP SKIP PLANT.NAME COL 36 PLANT.REGION
COL 43 PF(PLANT.PHONE 1 3) SP 0 '-' SP 0 PF(PLANT.PHONE 4 4)
SORT PLANT.NAME;


                 - PLANT TELEPHONE NUMBERS -


      PLANT                        PLANT      PHONE
       NAME                       REGION     NUMBER

BASIC TOYS                          NE     347-6000
CORPORATE HDQTRS DALLAS             SW     781-4200
DALLAS SALES                        SW     461-3130
DELUXE PRODUCTS                     NC     729-4138
DISTRIBUTION                        SC     550-7121
MECHANICAL PRODUCTS                 NW     849-7000
```

Figure 11-21 Using the PF and SORT Commands

In Figure 11-21, partial fielding was used to hyphenate the PLANT.PHONE field.

- The phone number started in position 1 of the PLANT.PHONE field and ran for a length of three characters before a hyphen was inserted.
- No spaces were left before or after the hyphen, which was treated as a character constant and enclosed within single quotation marks.
- Finally, VISION:Inquiry was told to begin partial fielding again in location 4 of the PLANT.PHONE field and to continue for a length of four.

The PLANT.NAME field was sorted in ascending order. SORT is used in UDO exactly as it is used in basic VISION:Inquiry.

Using the NOSP Command

Another way to write this inquiry is with the NOSP command, which suppresses default spacing of all fields that follow the command in the inquiry.

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 18 '- PLANT TELEPHONE NUMBERS -'
SKIP 2 SP 6 'PLANT' COL 35 'PLANT' COL 46 'PHONE'
LINE  SP 7 'NAME' COL 34 'REGION' COL 45 'NUMBER'
SKIP SKIP PLANT.NAME COL 36 PLANT.REGION NOSP
COL 43 PF(PLANT.PHONE 1 3) '-'  PF(PLANT.PHONE 4 4)
SORT PLANT.NAME;


                 - PLANT TELEPHONE NUMBERS -


      PLANT                        PLANT      PHONE
       NAME                       REGION     NUMBER

BASIC TOYS                          NE     347-6000
CORPORATE HDQTRS DALLAS             SW     781-4200
DALLAS SALES                        SW     461-3130
DELUXE PRODUCTS                     NC     729-4138
DISTRIBUTION                        SC     550-7121
MECHANICAL PRODUCTS                 NW     849-7000
```

Figure 11-22 Using the NOSP Command to Save Coding Time

By using the NOSP command, coding time was saved because the SP 0s did not have to be added to the inquiry.

The NOSP command is useful in the output of raw data.

Grand Summary Commands

With UDO, the grand summary commands, COUNT, TOTAL, and AVERAGE, are used exactly as they are in basic VISION:Inquiry.

By using UDO, you are able to control the location of the grand summary commands on your screen. However, because you do not know how many data items are displayed by your inquiry, you cannot place your summaries on a specific line.

Using Grand Summaries in a UDO Inquiry

The following inquiry uses the grand summary commands, TOTAL and AVERAGE.

Example: "List the toys produced in plant 40300. Include toy name, retail price, and quantity on hand in the listing. Also total the number of toys on hand and average the retail price of those toys."

```
PAGE:          TRANCODE: II      INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 13 '** TOYS IN DELUXE PRODUCTS DIVISION **'
LINE   SP 5 'TOY' COL 40 'RETAIL' COL 55 'NUMBER'
LINE   SP 5 'NAME' COL 40 'PRICE' COL 55 'ON HAND'
LINE   SP 5 '****' COL 40 '*****' COL 55 '*******'
LINE PROD.DESC COL 31 PROD.AMT COL 46 PROD.QTY
LINE 'TOTAL QUANTITY ON HAND:' COL 41 TOTAL PROD.QTY
LINE 'AVERAGE PRICE OF TOYS:' AVERAGE PROD.AMT
IF PLANT.ID = 40300;


             ** TOYS IN DELUXE PRODUCTS DIVISION **
     TOY                                 RETAIL         NUMBER
     NAME                                PRICE          ON HAND
     ****                                *****          *******
AFRICAN SAFARI                           98.00             48
CHEF'S DELIGHT                           79.00            105
HOSPITAL WARD                            99.00             76
TIME-LOCK SAFE                           65.00             34
TOTAL QUANTITY ON HAND:                                   263
AVERAGE PRICE OF TOYS:                   85.25
*
IXX9121  END OF INQUIRY.                  (6,1 USER DB CALLS,ROOTS)
```

Figure 11-23 Using the Grand Summaries, TOTAL and AVERAGE, in a UDO Inquiry

Notice that the grand summaries are displayed on the same page as the rest of the output from the inquiry. This is one of the differences between UDO and basic VISION:Inquiry. In basic VISION:Inquiry, grand summaries are output on a separate page.

The result of the TOTAL and AVERAGE commands is 21 positions or bytes with decimal places or 20 bytes without decimal places (15 significant digits plus four commas, plus the sign, with or without one decimal point).

The result of the COUNT command is eight positions. VISION:Inquiry uses these 21, 20, and 8 bytes internally for computation. You see these positions reflected in your output. Moreover, when you are formatting your inquiry, you must allow for them. Do not assign any of these positions to another field, or an error message results.

Spaces were not specified between the summary labels and the output fields. VISION:Inquiry defaulted two spaces after both grand summary labels. Because the TOTAL field in Figure 11-23 does not contain decimal places, it is allotted 20 bytes. It displayed its output in the seventeenth through the nineteenth positions.

																2	6	3	

17th, 18th, 19th, 20th

Because the AVERAGE field in Figure 11-23 contains decimal places, the system allows it 21 bytes. It displayed its output in the sixteenth through the twentieth positions. Both TOTAL and AVERAGE use the last byte for the sign.

															8	5	.	2	5	

16th, 17th, 18th, 19th, 20th, 21st

Subtotals

Subtotals in UDO are specified in your inquiry exactly as they are in basic VISION:Inquiry. To get a subtotal, enclose the control field plus the fields to be subtotaled within parentheses and add the appropriate command in front of the left parenthesis.

VISION:Inquiry associates the summary field with the related control break field. Every time a new control break occurs, the summary is printed on the same line with the related control break field. This is undesirable when detail lines (from the lower level segment) are printed along with the summary. Also, a literal specified to be displayed in front of the summary is displayed on every line.

To improve the output format of the summary and literal in this case, you can use the SKIP or LINE command to force the printing of the literal and summary on a different line.

Using Subtotals and Grand Summaries in a UDO Inquiry

The following is an inquiry example that first subtotals both employee names and salaries, and then calculates a grand summary average of some of the salaries.

Example: “Show a listing of the female staffing level. Include plant locations, number of women at each, and the count of the number of salaries paid to those women at each location. Also, list the average salary paid to the women.”

```
PAGE:          TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 12 '** FEMALE STAFFING LEVEL **'
SKIP COL 5 'PLANT' COL 22 'NUMBER OF' COL 36 'TOTAL'
LINE COL 4 'NUMBER' COL 22 'FEMALES'  COL 34 'SALARIES'
LINE COL 5 PLANT.ID COL 19 COUNT (PLANT.ID EMP.NAME)
COUNT (PLANT.ID SAL.YTD)
LINE 'AVERAGE SALARY:' AVERAGE SAL.YTD IF EMP.SEX = 'F';


            ** FEMALE STAFFING LEVEL **
    PLANT            NUMBER OF     TOTAL
   NUMBER            FEMALES     SALARIES
    10100                2         3
    20150                2         4
    30200                3         4
    40300                1         2
    50300                2         2
    60200                2         3
    70500                2         3
AVERAGE SALARY:             31,866.66
*
IXX9121 END OF INQUIRY.              (90,7 USER DB CALLS,ROOTS)
```

Figure 11-24 Using Subtotals and Grand Summaries in a UDO Inquiry

The system does not display subtotals in UDO with double asterisks as it does in basic VISION:Inquiry. Instead, you may label your subtotal output or display it under column headings.

Subtotals use the same internal format to calculate subtotals that the grand summary commands use: 20 or 21 positions for TOTAL and AVERAGE, eight positions for COUNT.

The inquiry code specified that the first subtotal, COUNT (PLANT.ID EMP.NAME), began in COL 19. The system moved to column 19 and then spaced across the page for eight additional positions to display its count of employee names. Because there was not any spacing specified for the second subtotal, COUNT (PLANT.ID SAL.YTD), the system defaulted two spaces, moved across the screen eight positions, and displayed its output.

You cannot assign specific line numbers for subtotal output because you do not know how many data elements are displayed by your inquiry.

EDIT Command

The EDIT command overrides the VISION:Inquiry automatic editing function.

You can specify three types of editing on numeric fields only with this command. These are:

N	No editing.
F	Full editing. This includes decimal point placement, comma insertion, and zero suppression.
Z	Zero suppression and decimal point placement only.

To write an EDIT command, enclose in parentheses the field to be edited and the edit specification, which must be treated as a character constant and enclosed within single quotation marks.

For example:

`EDIT`	`(SAL.YTD`	`'F')`
command	numeric field	edit specification

The three types of editing specifications are shown in Figure 11-25, Figure 11-26, and Figure 11-27 to illustrate how each performs editing.

Using the Edit Specification N with the EDIT Command

Example: “List employees in plant location 10100. Include salaries and educational degrees in the listing. Count the number of employees.”

```
PAGE:       TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1  'LISTING OF EMPLOYEES BY PLANT'
SKIP 'PLANT.ID:' PLANT.ID
SKIP COL 11 EMP.NAME COL 40 EDIT (SAL.YTD 'N') COL 55 ED.DEGREE
SKIP 'NUMBER OF EMPLOYEES:' COL 36 COUNT EMP.NAME
IF PLANT.ID = 10100;


LISTING OF EMPLOYEES BY PLANT
PLANT.ID:  10100
          WILLIAM AMES                 005200000      BA
                                       006400000
          PHYLLIS LOCKMEYER            004800000      BA
                                       005900000
          MARY ANN THOMAS              001560000      HS
NUMBER OF EMPLOYEES:                      3
*
*
*
IXX9121  END OF INQUIRY.                (19,1 USER DB CALLS,ROOTS)
```

Figure 11-25 Using the Edit Specification N with the EDIT Command

Using the Edit Specification F with the EDIT Command

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1  'LISTING OF EMPLOYEES BY PLANT'
SKIP 'PLANT.ID:' PLANT.ID
SKIP COL 11 EMP.NAME COL 40 EDIT (SAL.YTD 'F') COL 55 ED.DEGREE
SKIP 'NUMBER OF EMPLOYEES:' COL 36 COUNT EMP.NAME
IF PLANT.ID = 10100;


LISTING OF EMPLOYEES BY PLANT
PLANT.ID:  10100
          WILLIAM AMES                    52,000.00   BA
                                          64,000.00
          PHYLLIS LOCKMEYER               48,000.00   BA
                                          59,000.00
          MARY ANN THOMAS                 15,600.00   HS
NUMBER OF EMPLOYEES:                      3
*
*
*
IXX9121  END OF INQUIRY.                (19,1 USER DB CALLS,ROOTS)
```

Figure 11-26 Using the Edit Specification F with the EDIT Command

Using the Edit Specification Z with the EDIT Command

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1  'LISTING OF EMPLOYEES BY PLANT'
SKIP 'PLANT.ID:' PLANT.ID
SKIP COL 11 EMP.NAME COL 40 EDIT (SAL.YTD 'Z') COL 55 ED.DEGREE
SKIP 'NUMBER OF EMPLOYEES:' COL 36 COUNT EMP.NAME
IF PLANT.ID = 10100;


LISTING OF EMPLOYEES BY PLANT
PLANT.ID:  10100
          WILLIAM AMES                   52000.00     BA
                                         64000.00
          PHYLLIS LOCKMEYER              48000.00     BA
                                         59000.00
          MARY ANN THOMAS                15600.00     HS
NUMBER OF EMPLOYEES:                      3
*
*
*
IXX9121  END OF INQUIRY.                (19,1 USER DB CALLS,ROOTS)
```

Figure 11-27 Using the Edit Specification Z with the EDIT Command

Note that 'N' editing gives you unedited data, 'F' editing gives you fully edited data with leading zeroes suppressed, commas, and a decimal point inserted, and 'Z' editing gives you leading zeroes suppressed and a decimal point inserted but no commas. The positioning of summaries varies according to the editing specifications and the resulting field length.

Using EDIT Only with Numeric Field Names

The EDIT command can only be used with numeric database field names. You cannot override the editing of grand summaries or subtotals. When you use the EDIT command with a temporary field, you must compute the temporary field in a FIND statement first. Check with your system administrator to learn how this is done.

Arithmetic Processing

You use arithmetic processing with UDO exactly the same as you use it in basic VISION:Inquiry. With UDO, however, you may position your output on your screen or report page and add titles and column headings, rather than use the temporary field names for this purpose.

Using an Arithmetic Calculation in a Formatted Inquiry

In Figure 11-28, a simple arithmetic calculation and an equally simple conditional selection phrase are used in a formatted inquiry.

Example: "Calculate a bonus of 3.5% for all employees employed by the toy company in 1993. What will that bonus mean in both yearly and monthly terms of each salary? Sort that information in ascending order by salary."

```
PAGE:       TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE SP 15 '** YEARLY AND MONTHLY BONUSES FOR **'
 LINE SP 29 'EMPLOYEES'
 SKIP SP 2 'EMPLOYEE' COL 38 'YEARLY' COL 62 'MONTHLY'
 SKIP SP 4 'NAME' COL 39 'BONUS' COL 62 'BONUS'
 SKIP EMP.NAME COL 27 %BONUS = SAL.YTD * 0.0350
 COL 50 %MONTH.BONUS = %BONUS / 12 IF SAL.YEAR = 93
SORT SAL.YTD;


               ** YEARLY AND MONTHLY BONUSES FOR **
                             EMPLOYEES
  EMPLOYEE                           YEARLY                  MONTHLY
    NAME                              BONUS                  BONUS
JOHN HENRY CRANE                     469.0000                 39.0833
CHARLES SALTER                       840.0000                 70.0000
DAVID YORK                         1,085.0000                 90.4166
RONALD T JACKSON                   1,365.0000                113.7500
PETER ZATKIN                       1,540.0000                128.3333
DONALD M KING                      2,030.0000                169.1666
MITCHELL J HOOPS                   2,660.0000                221.6666
```

Figure 11-28 Using an Arithmetic Calculation in a Formatted Inquiry

The arithmetic calculation in Figure 11-28 calculated a 3.5 percent bonus for all those employed by the toy company in 1993. After multiplying the SAL.YTD field by 0.0350 to arrive at the yearly bonus, the result was divided by 12 to calculate the monthly increase. The output was sorted in ascending order by salary.

VISION:Inquiry allows 21 positions for all temporary numeric fields.

Because the output had two temporary fields, YEARLY BONUS and MONTHLY BONUS, care had to be taken in formatting the inquiry. Notice that the YEARLY BONUS column heading began in columns 38 and 39, but the output begins in column 27 because of the 21 positions reserved for temporary fields. The same is true of the column heading MONTHLY BONUS and its output.

Because formatting temporary fields is tricky, you may have to work with your output before you arrive at a screen or a page that is acceptable to you.

Using a Complex Calculation and a Temporary Field with UDO

Here is a complex arithmetic calculation and a conditional selection phrase that tests the value of the temporary field that is titled PLANNED RAISE.

Example: “If net salaries were raised by 11%, which male employees would still be earning less than $26,000?”

```
PAGE:          TRANCODE: II        INQUIRY:
DISPLAY PLANT FORMAT
LINE SP 10 'LISTING OF MALE EMPLOYEES EARNING LESS THAN $26,000'
LINE SP 24 'AFTER CURRENT RAISE'
LINE LINE 'EMPLOYEE' COL 32 'GROSS' COL 45 'SALARY'
COL 65 'PLANNED'
LINE SP 2 'NAME' COL 32 'SALARY' COL 44 'DEDUCTIONS'
COL 66 'RAISE'
SKIP SKIP EMP.NAME COL 28 SAL.YTD COL 43 SAL.DED COL 55 %NEW.SAL =
(SAL.YTD - SAL.DED) * 1.11 IF %NEW.SAL LT 26000 AND EMP.SEX = 'M';

          LISTING OF MALE EMPLOYEES EARNING LESS THAN $26,000
                        AFTER CURRENT RAISE

EMPLOYEE                   GROSS        SALARY              PLANNED
  NAME                     SALARY      DEDUCTIONS            RAISE

CHARLES SALTER            24,000.00    2,020.00           24,397.8000
JOHN HENRY CRANE          13,400.00    1,380.00           13,342.2000
                          19,600.00    1,960.00           19,580.4000
                          22,000.00    2,600.00           21,534.0000
STEPHEN MCGEE             19,000.00    1,700.00           19,203.0000
                          23,000.00    2,560.00           22,688.4000
*
*
IXX9121  END OF INQUIRY.                (91,7 USER DB CALLS,ROOTS)
```

Figure 11-29 Using a Complex Arithmetic Calculation and a Temporary Field that has a User Defined Column Heading in a Conditional Selection Phrase in a Formatted Inquiry

The more fields that you have in your inquiry, the more critical the spacing across your screen becomes, and the more careful you have to be in the placement of your temporary fields that require those 20 or 21 bytes.

The title of the report and the name of the column headings are the only elements that distinguish this UDO output from that of a non-UDO inquiry. The actual inquiry calculated an 11 percent increase of the salary field. VISION:Inquiry calculated net salary (fields in the parentheses) and multiplied this figure by 1.11 to determine the amount of the salary after the planned raise.

The conditional selection phrase limited the records selected to only those male employees who would be earning less than $26,000 a year.

FIND Command

In both non-UDO and UDO VISION:Inquiry, you use the FIND command to access fields from several databases in the same inquiry. You can also use the FIND command to access the same database more than one time.

Two separate statements are written when the FIND command is used:

- The first statement uses the FIND command and gives VISION:Inquiry information regarding the first database.
- The second statement uses the DISPLAY command and gives information on the second database, indicates what to do with the data accessed from both databases, and tells VISION:Inquiry how to relate the databases together.

See Chapter 8, "Accessing Multiple Databases and Files" for an explanation of the FIND command, and for examples of accessing a combination of databases, DB2 tables, and VSAM files.

Following is a formatted inquiry using the FIND command.

```
FIND SKILL %SKILL.NAME = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'ADMINISTRATOR';
DISPLAY PLANT FORMAT
LINE 1 SP 23' JOB CLASSIFICATION LIST'
SKIP SP 20 'CLASSIFICATION =' %SKILL.NAME
SKIP 2 COL 17 'PLANT' COL 34 'EMPLOYEE' COL 55 'NAME'
SKIP SKIP COL 17 PLANT.ID COL 35 EMP.NO COL 52 EMP.NAME
IF EMP.NO = %EMP.NUM;
```

Below is an explanation of how the inquiry works.

`FIND`	`SKILL`	`%SKILL.NAME = SKILL.NAME`
links two databases together	name of the first database	assigns a value of SKILL.NAME to a temporary field for transfer to a DISPLAY statement

`%EMP.NUM = EMP.NO`	`IF SKILL.NAME = 'ADMINISTRATOR'`	`;`
assigns a value of EMP.NO to temporary field for transfer to DISPLAY statement	conditional selection phrase limits records selected to only those whose SKILL.NAME is equal to character constant 'ADMINISTRATOR'	terminates FIND command

`DISPLAY`	`PLANT`	`FORMAT`
command to list	second database	command to format output

`LINE 1`	`SP 23`	`'JOB CLASSIFICATION LIST'`
controls vertical movement	controls horizontal positioning	character constant used to title this report

`SKIP`	`SP 20`	`'CLASSIFICATION='`	`%SKILL.NAME`
controls vertical movement	controls horizontal positioning	character constant used to title	displays this field from the first database

`SKIP 2`	`COL 17`	`'PLANT' COL 34 'EMPLOYEE' COL 55 'NAME'`
controls vertical movement	begins in position 17	character constants (in single quotation marks) used as column headings

`SKIP SKIP`	`COL 17`	`PLANT.ID COL 35 EMP.NO COL 52 EMP.NAME`
controls vertical movement	positions horizontal movement	fields from PLANT database

`IF EMP.NO = %EMP.NUM`	`;`
test fields from PLANT database against fields from SKILL database for matching value	terminates DISPLAY command

As with basic VISION:Inquiry, two separate statements are written when FIND is used in an inquiry. The first statement uses the FIND command and relays information to VISION:Inquiry concerning the SKILL database.

The second statement uses the DISPLAY command and informs VISION:Inquiry about the PLANT database, tells the system what to do with the information from both databases, and tells VISION:Inquiry to relate the two databases by aligning them to match on the EMP.NO field. Finally, the second statement tells VISION:Inquiry how to format the output for display.

Using a FIND Command in a Formatted Inquiry

Example: “List the employees who are classified as administrators. Include plant location, employee identification, and name. Use ’ADMINISTRATOR’ in the report title.”

```
PAGE:          TRANCODE:  II         INQUIRY:
FIND SKILL %SKILL.NAME = SKILL.NAME %EMP.NUM = EMP.NO
IF SKILL.NAME = 'ADMINISTRATOR';
DISPLAY PLANT FORMAT
LINE 1 SP 23 'JOB CLASSIFICATION LIST'
SKIP SP 20 'CLASSIFICATION = ' %SKILL.NAME
SKIP 2 COL 17 'PLANT' COL 34 'EMPLOYEE' COL 55 'NAME'
SKIP SKIP COL 17 PLANT.ID COL 35 EMP.NO COL 52 EMP.NAME
IF EMP.NO = %EMP.NUM;

                       JOB CLASSIFICATION LIST
                    CLASSIFICATION =   ADMINISTRATOR


                PLANT            EMPLOYEE             NAME

                20150             21137            PETER ZATKIN
                30200             30202            FREDERICH GRAY
                50300             50304            JONATHAN OAKS
                70500             70511            RONALD T JACKSON
*
*
IXX9121  END OF INQUIRY.                (91,41 USER DB CALLS,ROOTS)
```

Figure 11-30 Using a FIND Command in a Formatted Inquiry

Storing Inquiries

Using UDO, you can format stored inquiries that you or your system administrator write and store in the directory for your future use. You can call and execute these inquiries when you need the output.

Defining a Formatted Stored Inquiry

If you wanted to format and store an inquiry for displaying the toy company listing, you would do it this way:

Example: "Create a stored inquiry that lists the plants and the related employee identification numbers and names. Also include title, date, page numbers, and column headings in this stored inquiry."

```
PAGE:          TRANCODE: II        INQUIRY:
DEFINE DIRECTORY PLANT INQUIRY 'EMPLOYEE.LISTING' AS
DISPLAY PLANT FORMAT
LINE COL 4 'TOY COMPANY LISTING: 2001' DATE PAGE
SKIP SKIP SP 2 'PLANT' SP 5 'EMPLOYEE' SP 7 'EMPLOYEE'
LINE      SP 1 'NUMBER' SP 6 'NUMBER' SP 10 'NAME'
LINE      SP 2 PLANT.ID  SP 5 EMP.NO  SP 7 EMP.NAME;


IXX0254 DEFINE OF 'EMPLOYEE.LISTING' IS ADDED.
```

Figure 11-31   Defining a Formatted Stored Inquiry

Formatting and storing an inquiry in the directory is the same in UDO as it is in basic VISION:Inquiry. See Chapter 9, "Using Directory Commands - Storing Inquiries and Functions" for more details about storing inquiries.

By using the FORMAT command, you can specify exactly how the data will look when the stored inquiry executes. The report is have a title, the current date, and page numbers. Also, identify each of the fields with a column heading.

Using a Formatted Stored Inquiry

Figure 11-32 shows the report when the stored formatted inquiry EMPLOYEE.LISTING was executed.

Example: “List all the employees by plant location, identification number, and name. Include a title, date of output, page numbers, and column headings by using the formatted stored inquiry called’EMPLOYEE.LISTING’.L

```
PAGE:       TRANCODE: II       INQUIRY:
EMPLOYEE.LISTING PLANT;




   TOY COMPANY LISTING: 2001  11/16/01       1

  PLANT     EMPLOYEE       EMPLOYEE
 NUMBER      NUMBER          NAME
  10100     10103        WILLIAM AMES
            10104        PHYLLIS LOCKMEYER
            10105        MARY ANN THOMAS
  20150     21116        WILMA FORD
            21124        CHARLES SALTER
            21137        PETER ZATKIN
            21164        SUSAN WARE
```

Figure 11-32   Using a Formatted Stored Inquiry Named ‘EMPLOYEE.LISTING’

Substitutable Values

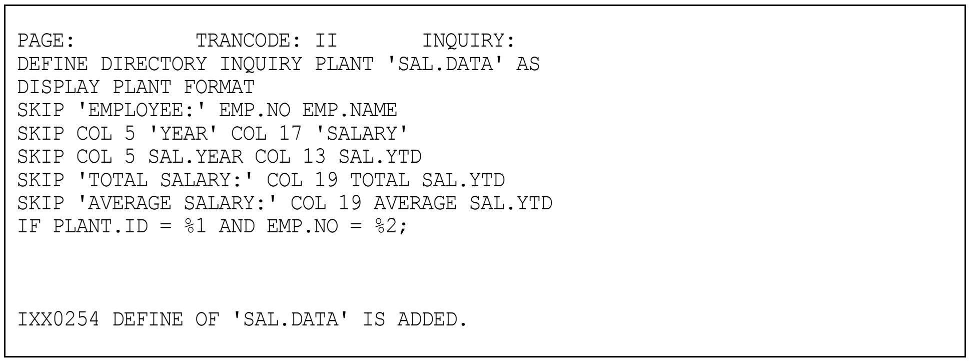
Using UDO, you can use formatted stored inquiries with substitutable values. These formatted stored inquiries operate the same as they do in basic VISION:Inquiry. When you call the formatted stored inquiry, you also provide any substitutable values. VISION:Inquiry makes the substitution of the data and processes the inquiry.

See Chapter 9, "Using Directory Commands - Storing Inquiries and Functions" for more details about storing inquiries with substitutable values.

Defining a Formatted Stored Inquiry with Substitutable Values

Figure 11-33 illustrates how to define a formatted stored inquiry with substitutable values to VISION:Inquiry.

Example: "Information is needed relating to salary years, salaries associated with those years, and total and average salaries. Also vary the plant location and employee identification numbers each time the inquiry is run."



```
PAGE:          TRANCODE: II       INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'SAL.DATA' AS
DISPLAY PLANT FORMAT
SKIP 'EMPLOYEE:' EMP.NO EMP.NAME
SKIP COL 5 'YEAR' COL 17 'SALARY'
SKIP COL 5 SAL.YEAR COL 13 SAL.YTD
SKIP 'TOTAL SALARY:' COL 19 TOTAL SAL.YTD
SKIP 'AVERAGE SALARY:' COL 19 AVERAGE SAL.YTD
IF PLANT.ID = %1 AND EMP.NO = %2;



IXX0254 DEFINE OF 'SAL.DATA' IS ADDED.
```

Figure 11-33 Defining a Formatted Stored Inquiry with Substitutable Values

SAL.DATA was formatted to display a listing of a single employee's salary data. The listing includes titles, column headings, employee identification and name, years worked and corresponding salaries earned, and, finally, summaries of total and average salary.

Using Substitutable Values in a Formatted Stored Inquiry

To use SAL.DATA, you enter the name of the stored inquiry, the name of the database, and the substitutable values:

```
SAL.DATA  PLANT  40300  40304;
```

Example: “Use a stored inquiry that lists salary years, related salaries earned during those years, and total and average salaries paid out. Use a formatted stored inquiry to report this information for Donald M. King.”

```
PAGE:       TRANCODE: II       INQUIRY:
SAL.DATA PLANT 40300 40304




EMPLOYEE  40304   DONALD M KING
    YEAR        SALARY
     93        58,000.00
     94        66,000.00
     95        75,000.00
TOTAL SALARY:               199,000.00
AVERAGE SALARY:              66,333.33
*
*
*
IXX9121  END OF INQUIRY.                (7,1 USER DB CALLS,ROOTS)
```

Figure 11-34 Using Substitutable Values in a Formatted Stored Inquiry

Substituting Different Values in the Same Formatted Stored Inquiry

You can use different substitutable values for your formatted stored inquiries simply by entering the new values. For example, if you were interested in seeing a report of employee Frederick Gray, you would substitute the values of his PLANT.ID and EMP.NO into the calling statement. Figure 11-35 shows the display.

Example: “Use the formatted stored inquiry to report salary years, salary earned during those related years, total and average salaries for Frederick Gray.”

```
PAGE:       TRANCODE: II        INQUIRY:
SAL.DATA PLANT 30200 30202;




EMPLOYEE  30202   FREDERICH GRAY
    YEAR        SALARY
     94        48,000.00
     95        59,000.00
TOTAL SALARY:               107,000.00
AVERAGE SALARY:              53,500.00
*
*
*
*
IXX9121  END OF INQUIRY.                (6,1 USER DB CALLS,ROOTS)
```

Figure 11-35 Substituting Different Values in the Same Formatted Stored Inquiry

You can also edit the stored UDO inquiries using the Text Editor facility the same way you edit basic VISION:Inquiry stored inquiries. See Chapter 10, "Using Stored Inquiries and Functions" for additional information about using the Text Editor.

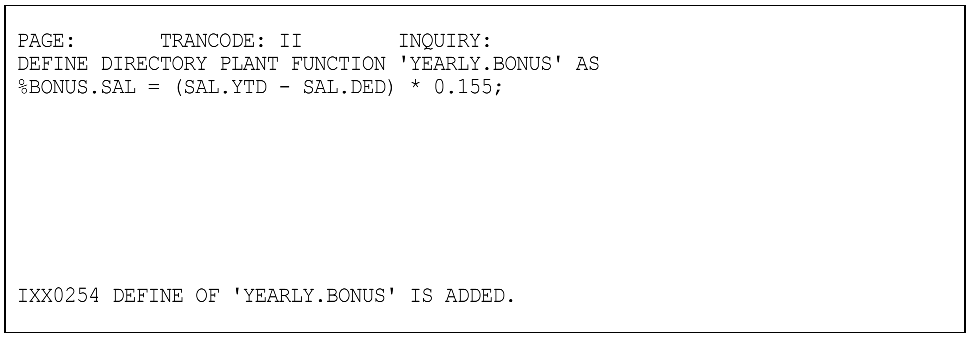
Stored Functions

Using UDO, you or your system administrator can create mathematical calculations to store as functions in the directory for future use. Later, these stored functions can be integrated into a formatted inquiry to perform the calculations needed to produce a particular report.

Storing a Function for a Formatted Inquiry

You define your function to the directory in UDO the same way as you define it in basic VISION:Inquiry (see Figure 11-36).

Example: "Write and store a function that calculates a 15.5% net bonus."



```
PAGE:       TRANCODE: II        INQUIRY:
DEFINE DIRECTORY PLANT FUNCTION 'YEARLY.BONUS' AS
%BONUS.SAL = (SAL.YTD - SAL.DED) * 0.155;









IXX0254 DEFINE OF 'YEARLY.BONUS' IS ADDED.
```

Figure 11-36   Storing a Function

Once your stored function has been defined to the directory, you can use its assigned name in a formatted inquiry statement. VISION:Inquiry automatically replaces the name of the function with the steps in the arithmetic calculations, performs the indicated arithmetic operations, and then formats your output when it executes the DISPLAY statement.

Using a Stored Function in a Formatted Inquiry

Figure 11-37 shows how the stored function YEARLY.BONUS was used in a formatted inquiry.

Example: “How much would each woman receive if she was awarded a 15.5% bonus? Include in the listing employee name, current salary, salary deductions, and bonus. Use the stored function in the formatted inquiry statement.”

```
PAGE:          TRANCODE: II       INQUIRY:
DISPLAY PLANT FORMAT
LINE 1 SP 15 '*** 1995 BONUSES FOR WOMEN EMPLOYEES ***'
SKIP 2 COL 4 'EMPLOYEE' COL 31 'CURRENT' COL 46 'SALARY'
COL 65 'YEARLY' LINE COL 6 'NAME' COL 32 'SALARY'
COL 45 'DEDUCTIONS' COL 66 'BONUS'
SKIP SKIP EMP.NAME COL 28 SAL.YTD COL 44 SAL.DED COL 55 YEARLY.BONUS
IF SAL.YEAR = 95 AND EMP.SEX = 'F';


               *** 1995 BONUSES FOR WOMEN EMPLOYEES ***


   EMPLOYEE                CURRENT        SALARY             YEARLY
     NAME                   SALARY       DEDUCTIONS           BONUS

PHYLLIS LOCKMEYER          59,000.00     8,200.00          7,874.0000
MARY ANN THOMAS            15,600.00     1,370.00          2,205.6500
WILMA FORD                 18,800.00     1,980.00          2,607.1000
SUSAN WARE                 41,000.00     7,000.00          5,270.0000
JANE LOWELL                26,000.00     5,800.00          3,131.0000
PATRICIA BLAKELY           30,000.00     4,120.00          4,011.4000
```

Figure 11-37 Using a Stored Function in a Formatted Inquiry

As Figure 11-37 illustrates, the results of the arithmetic calculation are listed under the column heading YEARLY BONUS. The output has been formatted exactly as specified with title and column headings.

Using a Formatted Stored Inquiry with a Stored Function and Substitutable Values

Example: "Create a report with title and column headings that shows the plant locations and yearly increase given to employees. It is important that this inquiry can be used for different years and various educational degrees. There is a stored function called YEARLY.BONUS that performs this calculation."

```
PAGE:          TRANCODE: II         INQUIRY:
DEFINE DIRECTORY INQUIRY PLANT 'BONUS.PLUS' AS
DISPLAY PLANT FORMAT
LINE SP 16 'COST OF LIVING INCREASE LISTING'
SKIP 2 SP 3 'PLANT' SP 12 'EMPLOYEE' COL 37 'YEARLY'
LINE   SP 2  'NUMBER' SP 14 'NAME' COL 56 'INCREASE'
SKIP 2 PLANT.ID COL 20 EMP.NAME COL 46 YEARLY.BONUS
IF SAL.YEAR = %1 AND ED.DEGREE = %2;



IXX0254 DEFINE OF 'BONUS.PLUS' IS REPLACED.
```

Figure 11-38 Using a Formatted Stored Inquiry with a Stored Function and Two Substitutable Values

The DEFINE statement adds BONUS.PLUS to the inquiries stored for the PLANT database. Included in this inquiry is the name of one stored function, YEARLY.BONUS. VISION:Inquiry substitutes the terms of the function for its name as it stores the inquiry; it also stores the formatting specifications.

The inquiry contains two substitutable values, SAL.YEAR and ED.DEGREE, that form the conditional (IF) selection. The substitutable values are indicated each time the inquiry is executed.

Notice in Figure 11-38 that the message "DEFINE OF'BONUS.PLUS' IS REPLACED." was received. The usual message, "DEFINE OF'BONUS.PLUS' IS ADDED." was not received because you had to change the formatting specifications several times to achieve the formatted output you wanted. "DEFINE OF'BONUS.PLUS' IS REPLACED." is the message you receive when you make corrections or modifications to an inquiry or function that has already been added to the directory. To execute this stored inquiry that calls a function, you would enter this statement at your terminal:

```
BONUS.PLUS  PLANT  95  'PD';
```

The steps in the function are performed when the stored inquiry is executed.

Executing a Formatted Stored Inquiry with a Stored Function and Substitutable Values

Example: “List the employees, their plant location, and their yearly increase given them by the 15.5% bonus if they worked during 1995 and they hold a Ph.D. degree.”

```
PAGE:       TRANCODE: II      INQUIRY:
BONUS.PLUS PLANT 95 'PD';



                COST OF LIVING INCREASE LISTING


   PLANT            EMPLOYEE                              YEARLY
  NUMBER              NAME                               INCREASE


40300              DONALD M KING                        8,928.0000
                   JOAN EVANS                           8,928.0000
*
IXX9121  END OF INQUIRY.                (141,7 USER DB CALLS,ROOTS)
```

Figure 11-39 Executing a Formatted Stored Inquiry with a Stored Function and Two Substitutable Values

By using a simple entry, an entire inquiry, including stored function and formatting, may be called and executed.

UDO Summary

FORMAT Within an inquiry statement, this command indicates that the UDO function constructs the output display. FORMAT is the primary UDO command; it must be used with DISPLAY and the database name, as: DISPLAY PLANT FORMAT.

This command is followed by other commands that are listed below.

FORMAT	LINE n	Character constant
	SKIP n	Name
	COLUMN n	System name
	COL n	Repeat n (any of the above)
	SPACE n	EDIT (name ’N’,’F’, or ’Z’)

SP n	PF (name start length)
NOSPACE	Page break
NOSP	

LINE n Controls the movement of the data down the page.

- When LINE is used with an integer, the number represents the exact line number where the data is to be output.
- When the integer is omitted from LINE, VISION:Inquiry outputs the data on the following line of the display.

The LINE command directly preceding a group of repeated items applies to each occurrence of the highest level item in the group.

A default of next line is assumed for all lower level items in the group of repeated items. 'n' can be a value from 1 to the logical page length.

The **logical page length** is the number of lines per page (TERMLTH parameter) multiplied by the number of pages (PAGE parameter). TERMLTH and PAGE parameters are specified at installation time.

SKIP n Controls the movement of the data down the page.

When SKIP is used with an integer, the number represents the number of lines to be skipped before the output is printed.

The SKIP command directly preceding a group of repeated items applies to each occurrence of the highest level item in the group.

A default of next line is assumed for all lower level items in the group of repeated items. 'n' can be a value from 0 to the page length minus 1.

The logical page length is the number of lines per page (TERMLTH parameter) multiplied by the number of pages (PAGE parameter). TERMLTH and PAGE parameters are specified at installation time.

COLUMN n
COL n Positions the data across the page.

COLUMN (or COL) is always used with an integer. Output begins in the position specified by the integer. 'n' can be a value from 1 to the page width.

SPACE n **SP n**	Positions the data across the page. Using the integer with the SPACE (or SP) command is optional. ■ When the integer is omitted, VISION:Inquiry defaults to one position between the end of one field and the beginning of the next field. ■ When the integer is specified, 'n' represents the number of blank columns between output. 'n' can be a value from 0 to the page width minus 1. When the SPACE and the COLUMN commands are omitted in the inquiry, VISION:Inquiry defaults to two positions except at the beginning of a line.	
NOSPACE **NOSP**	Suppresses the normal two-position default spacing of all fields or character constants that follow the command in the inquiry. You can use this command to output raw data, which is output exactly as it appears on the database, one field following another with no spaces between them. You can also use the NOSP command to save coding time.	
Character constant	Any text enclosed within single quotation marks.	
Name	One of the following:	
	Field name	Any database field name.
	Temporary field name	Any internal temporary field name currently existing in VISION:Inquiry.
System name	DATE	Supplies the current date in the form MM/DD/YY.
	DATEF	Supplies the current date in the form MM/DD/YYYY.
	TIME	Supplies the current time in the form HH:MM:SS.
	PAGE	Supplies the page numbers for the inquiry. Do not use DATE, DATEF, TIME, and PAGE in conditional selection phrases.

REPEAT n **REP n**	Specifies the number of segment occurrences displayed from the database. It does not stop the database scan. It limits output to the screen. The integer indicates the maximum number of times that the data from the leftmost highest level segment of the database is to be repeated. Always use this command with an integer and parentheses. ■ 'n' is a value from 0 to the number of occurrences in the segment. ■ The parentheses define the range of the control of the REP command — REP 3 (PLANT.ID PROD.CODE).
EDIT (name 'N', 'F', or 'Z')	Indicates that the field following the EDIT command is to be edited as specified in the edit specifications for output. Apply this command to numeric fields only.

name	Either a field name or a temporary field name as specified above.

Edit is an optional function performed on output data items. The editing specifications are:

N	No editing.
F	Full editing. This includes decimal point placement, comma insertion, and zero suppression.
Z	Zero suppression and decimal point placement only.

Treat the editing specifications as character constants. They follow the field to be edited. Both must be within parentheses; if the parentheses are omitted, VISION:Inquiry defaults to the OUTEDIT parameter value in the MAPGEN specified by the system administrator at installation time.

PF (name start length)	Use to display part of a field. The field that is to be processed, the start location, and the length of the partial field must be enclosed within parentheses. Do not use this command in a conditional selection phrase.

name	Either a field name or a temporary field name.
start	An integer indicating the start location of the partial field within the specified data field.
length	An integer indicating the length of the partial field.

page break	By explicitly specifying the line number on which the output is to be displayed in the inquiry, VISION:Inquiry prints all occurrences of that field on the specified line number of all subsequent pages. You must specify the explicit line number to the highest level segment in the inquiry.
/FORMAT INQUDO	IMS command that displays the UDO MFS module. (IMS users only.)

Appendix

A Commands, Noise Words, and Name Combinations

This appendix summarizes commands, noise words, and name combinations.

Commands

The VISION:Inquiry commands described in this guide are listed below, along with a brief description of their function and a reference to the chapter in which they are explained.

Some VISION:Inquiry commands do not have to be completely spelled out in an inquiry; you can use abbreviations (synonyms) instead. The standard abbreviations are shown on the following pages.

Each installation specifies the actual names of the commands and their abbreviations at installation time. Check with your system administrator for the particular names used in VISION:Inquiry.

The native SQL syntax facility can be used by certain VISION:Inquiry commands as specified on the following pages. The other applicable VISION:Inquiry commands must be implemented using the capabilities of the SQL SELECT statement.

Command	Function	Synonym	Chapter
DISPLAY	Displays indicated fields	D PRINT	Chapter 5
IF	Performs conditional selection	WITH	Chapter 5
FIRST	Tests first occurrence of a field		Chapter 5
LAST	Tests last occurrence of a field		Chapter 5
ESCAPE	Used with the LIKE operator		Chapter 5
SORT	Resequences data as it is output		Chapter 5
ASC (default)	Sorts into ascending order		Chapter 5
DSC	Sorts into descending order		Chapter 5

Command	Function	Synonym	Chapter
LIMIT	Limits the number of lines output		Chapter 5
OUTPUT	Sends output display to another output device		Chapter 5
TOTAL	Calculates and displays grand or group totals		Chapter 6
COUNT	Calculates and displays grand or group counts		Chapter 6
AVERAGE	Calculates and displays grand or group averages		Chapter 6
SUM	Performs subtotals and arithmetic calculations on subgroups of data		Chapter 7
FIND	Use to access to one or more databases	INTER I	Chapter 8
DEFINE DIRECTORY INQUIRY	Stores an inquiry in the directory	DDI	Chapter 9
DEFINE DIRECTORY FUNCTION	Stores a function in the directory	DDF	Chapter 9
DISPLAY DIRECTORY INQUIRY	Displays one or more stored inquiries	PDI	Chapter 9
DISPLAY DIRECTORY INQUIRY WHOLE	Displays all the stored inquiries	PDIW	Chapter 9
DISPLAY DIRECTORY INQUIRY COMMENT	Displays one or more stored inquiry names and their associated comments.	PDIC	Chapter 9
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE	Displays all the stored inquiry names and their associated comments.	PDICW	Chapter 9

Command	Function	Synonym	Chapter
DISPLAY DIRECTORY FUNCTION	Displays one or more stored functions	PDF	Chapter 9
DELETE DIRECTORY INQUIRY	Deletes a stored inquiry from the directory		Chapter 9
DELETE DIRECTORY FUNCTION	Deletes a stored function from the directory		Chapter 9
name of stored query or function	Use stored inquiries or functions		Chapter 10
EDITSQ	Use to edit stored inquiries		Chapter 10
FORMAT	Primary UDO command that produces formatted output		Chapter 11
COLUMN n	Positions data across the page	COL n	Chapter 11
SPACE n	Positions data across the page	SP n	Chapter 11
LINE n	Positions data down the page		Chapter 11
SKIP n	Positions data down the page		Chapter 11
NOSPACE	Turns off the spacing between output fields	NOSP	Chapter 11
EDIT	Defines the specifications for editing a numeric field		Chapter 11
REPEAT n	Specifies the number of segment occurrences displayed from the database	REP n	Chapter 11
PF	Use to display part of a field		Chapter 11
SHOW	Use to see the next record in your buffer when you have reached the limit of your database calls when operating in conversational mode using the FORMAT command		Chapter 4
CONTINUE	Indicates that processing of an inquiry should continue	PF1	Chapter 4

Command	Function	Synonym	Chapter
DEFER	Indicates that processing of an inquiry should be deferred	PF2 DI	Chapter 4
CONTINUE DEFERRED INQUIRY	Indicates that processing of a deferred inquiry should now be continued	PF3 CDI	Chapter 4
DELETE DEFERRED INQUIRY	Deletes a deferred inquiry from the system		Chapter 4

Notes on the Commands

- FIRST, LAST, and SUM do not apply to VSAM non-hierarchical files and DB2 tables.
- PF1, PF2, and PF3 are program function keys on terminal keyboards for systems operating under IMS. For systems operating under CICS, type in TRANCODE area: 1 for CONTINUE, 2 for DEFER, and 3 for CONTINUE DEFERRED INQUIRY.
- Can also be used with the native SQL syntax inquiries:

DISPLAY	DELETE DEFERRED INQUIRY
OUTPUT	LIMIT
DISPLAY DIRECTORY INQUIRY	DEFINE DIRECTORY INQUIRY
DISPLAY DIRECTORY INQUIRY COMMENT	DELETE DIRECTORY INQUIRY
EDITSQ	CONTINUE
DEFER	CONTINUE DEFERRED INQUIRY

Noise Words

Note: Do not use noise words in inquiries using the native SQL syntax facility. They terminate the process and issue an error message.

You can add the following words to an inquiry for clarity; however, they are not required. Note that the use of noise words increases translation time.

A	FROM	IS	THAN
AN	HAD	NO	THE

ARE	HAS	NONE	TO
AS	HAVE	OF	WAS
BY	IN	ON	WERE

Name Combinations

You can use the following name combinations or their abbreviations with VISION:Inquiry.

DISPLAY DIRECTORY MAP	PDM
DISPLAY DIRECTORY MAP WHOLE	PDMW
DISPLAY DIRECTORY INQUIRY	PDI
DISPLAY DIRECTORY INQUIRY WHOLE	PDIW
DISPLAY DIRECTORY INQUIRY COMMENT	PDIC
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE	PDICW
DISPLAY DIRECTORY VOCABULARY	PDV
DISPLAY DIRECTORY DATA	PDD
DISPLAY DIRECTORY DATA WHOLE	PDD WHOLE
DISPLAY DIRECTORY FUNCTION	PDF
DISPLAY DIRECTORY LTERM	PDL
DISPLAY DIRECTORY DATA DESCRIPT	PDDDS

Notes on Name Combinations

- As a security feature, the DISPLAY DIRECTORY commands shown above cannot have their output routed to another terminal.
- DISPLAY DIRECTORY MAP (PDM) displays only the first page of the map.
- You can use the commands DISPLAY DIRECTORY INQUIRY (PDI) and DISPLAY DIRECTORY DATA (PDD) with native SQL syntax inquiries.

You can find examples for DISPLAY DIRECTORY INQUIRY (PDI and PDIW), DISPLAY DIRECTORY INQUIRY COMMENT (PDIC and PDICW), and DISPLAY DIRECTORY FUNCTION (PDF) in Chapter 9, "Using Directory Commands - Storing Inquiries and Functions". Examples for the balance of the DISPLAY DIRECTORY commands follow.

DISPLAY DIRECTORY MAP (PDM) Command with IMS (DL/I)

The following is an example of using the DISPLAY DIRECTORY MAP (PDM) command with the IMS (DL/I) database, PLANT.

```
PAGE:       TRANCODE: II INQUIRY :
PDM PLANT;

PLANT    IIDBDDM
SEGMENT  PARENT   LV  S/L  P/L    KEY       L TYPE
PLANT             01 0040 00000 PLANTKEY  005 KEYED
PROD     PLANT    02 0035 00005 PRODKEY   002 KEYED
EMP      PLANT    02 0031 00005 EMPKEY    005 KEYED
SAL      EMP      03 0011 00010 SALKEY    002 KEYED
ED       EMP      03 0014 00010 EDKEY     002 KEYED
SUB      ED       04 0012 00012 SUBKEY    010 KEYED
*
*
*
*
IXX9121 END OF INQUIRY.       (0,0 USER DB CALLS,ROOTS)
```

Figure A-1 DISPLAY DIRECTORY MAP Command

Following are explanations for the headings in Figure A-1.

PLANT	The VISION:Inquiry database name, followed by the DBD name.
SEGMENT	The name of each segment within the database.
PARENT	The name of the parent for that segment.
LV	The segment level.
S/L	The length of the segment.
P/L	The path length. It is the combined length of each key field from the root segment along the path to this segment. It does not include the length of the key field from this segment.
KEY	The name of the key field for this segment.
L	The length of the key field.
TYPE	Whether or not the segment has a key field.

DISPLAY DIRECTORY MAP (PDM) Command with VSAM Non-Hierarchical File

The following is an example of the DISPLAY DIRECTORY MAP (PDM) command with the VSAM non-hierarchical file, VSPLANT.

```
PAGE:           TRANSACTION: II INQUIRY:

PDM VSPLANT;;


VSPLANT VSPLDS
TYPE  KEY         R/L   K/L
KSDS  KEYED       00080 005
*
*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure A-2 DISPLAY DIRECTORY MAP for VSAM Non-Hierarchical File

Following are explanations for the headings in Figure A-2.

VSPLANT	The VISION:Inquiry file name, followed by the VSAM file ddname.
TYPE	The type of VSAM file (that is, KSDS, ESDS, or RRDS).
KEY	Whether or not the VSAM file is keyed.
R/L	The length of the record.
K/L	The length of the key field.

DISPLAY DIRECTORY MAP (PDM) Command with VSAM Hierarchical File

The following is an example of using the DISPLAY DIRECTORY MAP (PDM) command with the VSAM hierarchical file, VSHPLANT.

```
PAGE:          TRANSACTION: II INQUIRY :

PDM VSHPLANT;;

VSHPLANT VSHPLDS
SEGMENT  PARENT   LV  S/L  P/L    KEY       L TYPE
VSHPLANT          01 0041 00000 VSHPLDS   005 KEYED
VSHPROD  VSHPLANT 02 0035 00005           000 NON-KEYED
VSHEMP   VSHPLANT 02 0035 00005           000 NON-KEYED
VSHSAL   VSHEMP   03 0011 00005           000 NON-KEYED
VSHED    VSHEMP   03 0016 00005           000 NON-KEYED
VSHSUB   VSHED    04 0012 00005           000 NON-KEYED
*
*
*
*
*
  IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure A-3 DISPLAY DIRECTORY MAP for VSAM Hierarchical File

Following are explanations for the headings in Figure A-3.

VSHPLANT	The VISION:Inquiry file name, followed by the VSAM hierarchical file ddname.
SEGMENT	The name of each segment defined in the VSAM file.
PARENT	The name of the parent for that segment.
LV	The segment level.
S/L	The length of the segment.
P/L	The path length. It is the combined length of each key field from the root segment along the path to this segment. It does not include the length of the key field from this segment.
KEY	The name of the key field for this segment.
L	The length of the key field.
TYPE	Whether or not the segment has a key field.

DISPLAY DIRECTORY MAP (PDM) Command with DB2 Table

The following is an example of the DISPLAY DIRECTORY MAP (PDM) command with the DB2 table, SALARIES.

```
 PAGE:      TRANCODE: II       INQUIRY:
 PDM SALARIES;;

 SALARIES DYLINQ
 SEGMENT  PARENT   LV  S/L  P/L    KEY       L TYPE
 DB2SEG            01 0091 00000          000 NON-KEYED
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure A-4 DISPLAY DIRECTORY MAP for DB2 Table

Following are explanations for the headings in Figure A-4.

SALARIES	The VISION:Inquiry database name, followed by the AUTHID name.
SEGMENT	The segment is always DB2SEG for DB2 databases.
PARENT	This is not applicable to DB2 tables.
LV	This is not applicable to DB2 tables and is always 01.
S/L	The length of the table.
P/L	This is not applicable to DB2 tables.
KEY	This is not applicable to DB2 tables.
L	This is not applicable to DB2 tables. It is always 0.
TYPE	This is not applicable to DB2 tables. It is always non-keyed.

DISPLAY DIRECTORY VOCABULARY (PDV) Command

Figure A-5 shows a DISPLAY DIRECTORY VOCABULARY (PDV) example.

- To see all the names in the vocabulary, enter the PDV command, as shown in Figure A-5.
- To see a specific name, enter the PDV command, followed by the name surrounded by single quotation marks, for example: PDV ’name’.

Note: The DISPLAY DIRECTORY VOCABULARY (PDV) command can only be applied to the DB2 tables defined in the VISION:Inquiry directory by the system administrator. The DB2 tables accessed by the native SQL syntax facility are not required to be defined in the directory.

```
PAGE:        TRANCODE: II      INQUIRY:
PDV ;

NAME                                   CODE
<                                      5004
<=                                     5010
(                                      6600
+                                      5800
|                                      6100
&                                      6000
*                                      5A00
)                                      6700
;                                      6C00
¬=                                     5018
-                                      5900
/                                      5B00
>                                      5008
>=                                     5014
#                                      1B00
=                                      5000
```

Figure A-5 DISPLAY DIRECTORY VOCABULARY Command

Following are explanations for the column headings, NAME and CODE, used in the example:

NAME	The character or characters that are included as part of the vocabulary.
CODE	The VISION:Inquiry internal code for each item of the vocabulary.

DISPLAY DIRECTORY DATA (PDD) Command

The following is an example of the DISPLAY DIRECTORY DATA (PDD) command for the databases/files defined in the VISION:Inquiry directory by the system administrator.

- To see all the data elements from the specified file/database in alphabetical order, enter the PDD command followed by the file/database name.
- To display the data elements starting from 'ABC', specify a specific data element, for example: PDD PLANT 'ABC'1;;

```
PAGE:       TRANCODE: II  INQUIRY:
PDD PLANT;


NAME                             TYPE SEGMENT  T  O    L   S  S/O S/L E/L  1
DC01                             IQY
ED.DEGREE                        DDNK ED       C 0002 002 000 000 000 002
ED.SCHOOL                        DDNK ED       C 0004 010 000 000 000 010
ED.YEAR                          DDK  ED       N 0000 002 000 000 000 004
EMP.NAME                         DDNK EMP      C 0006 025 000 000 000 025
EMP.NO                           DDK  EMP      N 0000 005 000 000 000 006
EMP.SEX                          DDNK EMP      C 0005 001 000 000 000 001
PLANT                            DB
PLANT.ID                         RK   PLANT    C 0000 005 000 000 000 005
PLANT.NAME                       DDNK PLANT    C 0014 025 000 000 000 025
PLANT.PHONE                      DDNK PLANT    C 0007 007 000 000 000 007
PLANT.REGION                     DDNK PLANT    C 0005 002 000 000 000 002
PROD.AMT                         DDNK PROD     B 0006 004 002 000 000 015
PROD.CODE                        DDK  PROD     C 0000 002 000 000 000 002
```

Figure A-6 DISPLAY DIRECTORY DATA Command

Following are explanations for the headings in Figure A-6.

NAME	Is followed by all the data items in the directory	
TYPE	A description of the field	
	IQY	a stored inquiry
	FNC	a stored function
	DDNK	data definition, non-keyed
	DDK	data definition, keyed
	RK	root key
	DB, VSAM	database/VSAM file name (not applicable to DB2)
	SYNK	synonym, non-keyed
	SYK	synonym, keyed

	SRCH	data definition, search field
	INDX	data definition secondary index/alternative key field

SEGMENT	The name of the segment in which the field is contained, for IMS and VSAM hierarchical data sets. For DB2 tables, always "DB2SEG." For VSAM non-hierarchical files: " ,KS" for KSDS type data sets " ,ES" for ESDS type data sets " ,RR" for RRDS type data sets
T	Field type

C	character
N	numeric
B	binary
P	packed
Y	substring of a packed field

O	The starting position of the field relative to the beginning of the segment
L	The length of the field within the segment
S	The number of decimal positions
S/O	Used for substrings of packed fields. It is the starting position of a digit of the substring
S/L	Used for substrings. It is the length of the substring
E/L	The default edited length of a field when displayed at the terminal

PDD Command for a DB2 Table

You can use the DISPLAY DIRECTORY DATA (PDD) command to get the characteristics of data elements of a DB2 table directly from the DB2 catalog. This is useful for the DB2 tables that are accessed through the native SQL facility and are not defined in the VISION:Inquiry directory.

Enter the PDD command followed by the DB2 table name enclosed in native SQL syntax delimiters EXECSQL and ENDEXEC.

The output will be the characteristics of the columns for the specified DB2 table in the order defined to the table. You can also get the output in alphabetical order of the column names by adding the ORDER BY command to the DB2 table name in the inquiry.

Figure A-7 and Figure A-8 show the output of the PDD command for a DB2 table, in the order defined and in alphabetical order, respectively.

```
PAGE:      TRANCODE: II       INQUIRY:
PDD EXECSQL DYLINQ.IIEMP_SAL ENDEXEC;

         SQLNAME                SQLTYPE    SQLLEN  PRCSN  SCALE  NULL
------------------------------  --------   ------  -----  -----  ----
EMPLOYEE                        CHARACTR   000005                  N
YEAR                            CHARACTR   000002                  N
YEAR_TO_DATE                    DECIMAL    000005    09     02     Y
DEDUCTIONS                      DECIMAL    000004    07     02     Y
*
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.             (0,0 USER DB CALLS,ROOTS)
```

Figure A-7 PDD Command Using DB2 Catalog

```
PAGE:      TRANCODE: II       INQUIRY:
PDD EXECSQL DYLINQ.IIEMP_SAL ORDER BY ENDEXEC;

         SQLNAME                SQLTYPE    SQLLEN  PRCSN  SCALE  NULL
------------------------------  --------   ------  -----  -----  ----
DEDUCTIONS                      DECIMAL    000004    07     02     Y
EMPLOYEE                        CHARACTR   000005                  N
YEAR                            CHARACTR   000002                  N
YEAR_TO_DATE                    DECIMAL    000005    09     02     Y
*
*
*
*
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.             (0,0 USER DB CALLS,ROOTS)
```

Figure A-8 PDD Command Using DB2 Catalog (Sorted Output)

Following are explanations for the headings in Figure A-7 and Figure A-8.

SQLNAME	Column name
SQLTYPE	Data type

CHARACTER	Fixed-length Character
DATE	Date
TIME	Time
TIMESTMP	Timestamp
FLOAT	Single or Double Precision Floating-point
LRGINT	Large Integer
SMLINT	Small Integer
VARCHAR	Varying-length Character (maximum length 254)
LONGCHAR	Varying-length Character (maximum length 254)
VARGRAPH	Varying-length Graphic (maximum length127)
LNGGRAPH	Varying-length Graphic (maximum length127)
GRAPHICS	Fixed-length Graphic
DECIMAL	Decimal Number

SQLLEN	Length of the column in the DB2 table
PRCSN	Total number of digits for a DECIMAL data type, otherwise blank
SCALE	Number of digits to the right of the decimal point for DECIMAL data type, otherwise blank

DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE) Command

You can also combine the PDD command with the WHOLE keyword to display the 60-byte description for each field in the file/database.

Enter the PDD WHOLE command followed by the file/database name to see all the data elements, as well as the field descriptions, for all the fields from the specified file/database in alphabetical order.

- Two lines of output will be created for each field: the field characteristics on the first line, and the field description on the second line.
- The output for the other elements in the file/database, such as stored inquiries, consists of one line.

Notes:

- The PDD WHOLE command can only be applied to the DB2 tables defined in the VISION:Inquiry directory by the system administrator.
- The DB2 tables accessed by the native SQL syntax facility are not required to be defined in the directory.

The following is an example of the DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE) command.

```
PAGE:       TRANCODE:  II     INQUIRY:
PDD WHOLE PLANT;;

NAME                                   TYPE SEGMENT  T  O    L   S  S/O S/L E/L
ED.DEGREE                              DDNK ED      C 0002 002 000 000 000 002
  DEGREE ATTAINED
ED.SCHOOL                              DDNK ED      C 0004 010 000 000 000 010
  SCHOOL ATTENDED
ED.YEAR                                DDK  ED      N 0000 002 000 000 000 004
  YEAR OF GRADUATION
EMP.NAME                               DDNK EMP     C 0006 025 000 000 000 025
  NAME OF EMPLOYEE
EMP.NO                                 DDK  EMP     N 0000 005 000 000 000 006
  IDENTIFICATION CODE FOR EMPLOYEE
EMP.SEX                                DDNK EMP     C 0005 001 000 000 000 001
  EMPLOYEE'S SEX AS OF DATE OF HIRE
PLANT                       DB
PLANT.ID                               RK   PLANT   C 0000 005 000 000 000 005
  IDENTIFICATION CODE FOR PLANT
PLANT.NAME                             DDNK PLANT   C 0014 025 000 000 000 025
  NAME OF PLANT
PLANT.PHONE                            DDNK PLANT   C 0007 007 000 000 000 007
  MAIN SWITCHBOARD PHONE NUMBER
```

Figure A-9   DISPLAY DIRECTORY DATA WHOLE Command

DISPLAY DIRECTORY LTERM (PDL Command) in IMS

Figure A-10 is an example of using the DISPLAY DIRECTORY LTERM (PDL) command in an IMS system.

- To see all the terminals defined to the system, enter the PDL command, as shown in Figure A-10.
- To see the characteristics of a specific terminal, enter the PDL command, followed by the name of the terminal surrounded by single quotation marks, for example: PDL 'name'.

```
PAGE:       TRANCODE: II    INQUIRY:
PDL;

NAME     MFS_NAME TERM_L TERM_W TIME PAGE CKPT MODE I/S MFSLN PC_MFS PCPAGE
AXT**    ID4QRY   042    080    0120 0200 YES  110  15  131            0001
BXT**    ID4INQ   039    079    0100 0001 YES  111  20  119            0001
CXT**    ID4QRY   042    080    0120 0200 YES  110  15  131            0001
DYL***** ID4INQ   039    079    0120 0005 YES  111  39  119            0001
NMSPG***          020    079    0250 0005 YES  111  20  119   IDFTSP7  0002
DYL11111 ID4INQ   039    079    0100 0001 YES  111  20  119            0001
DYL11112 ID4INQ   039    079    0100 0001 YES  111  20  119            0001
DYL11115 ID4UDO   029    030    0200 0002 YES  101  10  131            0001
DYL11116 ID4QRY   042    080    0120 0200 YES  110  15  131            0001
DYL11117 ID4UDO   029    030    0200 0002 YES  101  10  131            0001
DYL11120 ID4UDO   029    040    0200 0002 YES  101  10  131            0001
DYL11121 ID4UDO   030    030    0200 0002 YES  101  10  131            0001
DYL11122 ID4UDO   029    030    0200 0002 YES  111  10  131            0001
DYL11123 ID4UDO   029    030    0150 0002 YES  101  10  131            0001
DYL11124 ID4UDO   029    030    0200 0003 YES  101  10  131            0001
```

Figure A-10 DISPLAY DIRECTORY LTERM for IMS

Following are explanations for the headings in Figure A-10. For more information, see the *Advantage VISION:Inquiry for IMS and TSO Technical Reference Guide*.

NAME	The logical terminal name.
MFS_NAME	The name of a predefined MFS description.
TERM_L	The number of lines of output data per logical page.
TERM_W	The number of characters of a line of output for the terminal.
TIME	Number of database calls/VSAM reads before a checkpoint is taken.
PAGE	Number of logical pages to be processed before a checkpoint is taken.
CKPT	Shows if a checkpoint is allowed for this terminal.

MODE	The mode type of the terminal.
I/S	The number of lines inserted into the message queue with one INSERT call.
MFSLN	The length of the MFLD statements of the MFSGEN.
PCPAGE	The page limit checkpoint; multiplied by TERM_L, gives the number of records extracted before a page limit checkpoint occurs (VISION:Journey and Intraccess use only).

DISPLAY DIRECTORY LTERM (PDL Command) in CICS

Figure A-11 shows an example of the DISPLAY DIRECTORY LTERM (PDL) command in a CICS system.

- To see all the terminals defined to the system, enter the PDL command, as shown in Figure A-11.
- To see characteristics of a specific terminal, enter the PDL command followed by the name of the terminal, surrounded by single quotation marks, for example: PDL 'name'.

```
PAGE: P/N    TRANSACTION: IQIO
                           Enter Inquiry Below:


PDL;


NAME     BMS_NAME TERM_L TERM_W TIME PAGE CKPT MODE FTG       PC_BMS   PCPAGE
L***     INQMAP1  020    079    0250 0005 YES  101  NO                 0001
S***              020    079    0000 0000 YES  001  NO                 0001
********          012    079    2500 0005 YES  101  NO        IDFTSP7  0002
BATCH             055    132    0000 0000 YES  000  NO                 0001
DY1               020    079    0030 0001 YES  101  NO                 0001
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
 TYPE IN INQUIRY, PRESS ENTER TO RUN QUERY.
```

Figure A-11   DISPLAY DIRECTORY LTERM for CICS

Note: The parameters TERM_L and TERM_W are used only for UDO and the OUTPUT command. For non-UDO, the parameters are calculated dynamically.

Following are brief descriptions of the entries in Figure A-11. For more information, see the *Advantage VISION:Inquiry for IMS and TSO Technical Reference Guide.*

NAME	The terminal name.
BMS_NAME	The name of a predefined BMS mapset.
TERM_L	The number of lines of output data per logical page.
TERM_W	The number of characters of a line of output for the terminal.
TIME	The number of DL/I calls/VSAM reads before a checkpoint is taken.
PAGE	The number of logical pages to be processed before a checkpoint is taken.
CKPT	Shows if a checkpoint is allowed for this terminal.
MODE	The mode type of the terminal.
FTG	Whether or not the footing map is used when the output pages are built.
PCPAGE	The page limit checkpoint; multiplied by TERM_L, gives the number of records extracted before a page limit checkpoint occurs. (VISION:Journey and Intraccess use only.)

DISPLAY DIRECTORY DATA DESCRIPT (PDDDS) Command

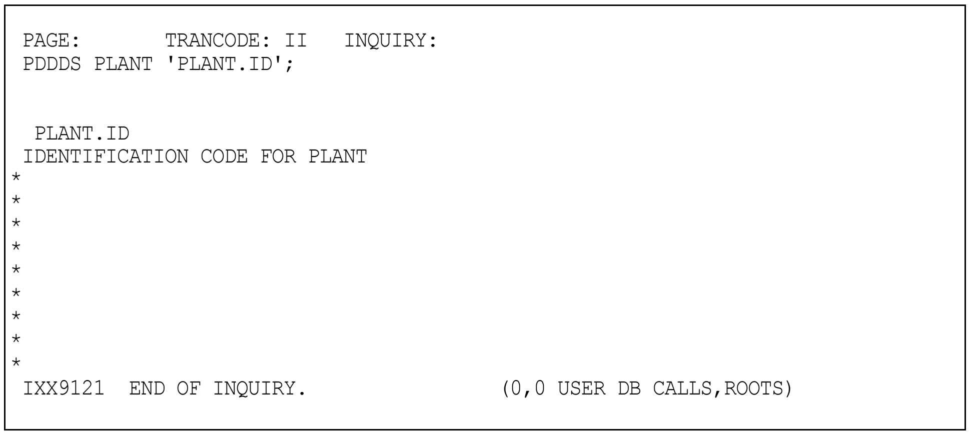
Figure A-12 shows an example of the DISPLAY DIRECTORY DATA DESCRIPT (PDDDS) command. It displays the 60-byte field description for the field PLANT.ID from the PLANT database.

- To display a field description, enter the PDDDS command, followed by the file/database name. Enclose the field name, PLANT.ID, in single quotation marks, for example: PDDDS PLANT 'PLANT.ID'.

  Field descriptions should be included at the time the system is generated by the system administrator in the DESCRIPT parameter of the FIELD statement.

- To display only the database description, enter the PDDDS command followed by only the database name.

  The database description should be included at the time the system is generated by the system administrator in the DESCRIPT parameter of the MAPGEN statement.



```
 PAGE:       TRANCODE: II   INQUIRY:
 PDDDS PLANT 'PLANT.ID';


  PLANT.ID
 IDENTIFICATION CODE FOR PLANT
*
*
*
*
*
*
*
*
*
 IXX9121  END OF INQUIRY.                (0,0 USER DB CALLS,ROOTS)
```

Figure A-12 DISPLAY DIRECTORY DATA DESCRIPT Command

Notes:

- The PDDDS command can only be applied to the tables defined in the VISION:Inquiry directory by the system administrator.
- The DB2 tables accessed by the native SQL syntax facility are not required to be defined in the directory.

DISPLAY DIRECTORY MAP WHOLE (PDMW) Command

Figure A-13 is an example of the DISPLAY DIRECTORY MAP WHOLE (PDMW) command. It lists all the databases/files and their types available to the terminal/user through the primary directory and the connected directories (if any exist).

Notes:

- The PDMW command can only list the tables defined in the VISION:Inquiry directory by the system administrator.
- The DB2 tables accessed by the native SQL syntax facility are not required to be defined in the directory.

```
 PAGE:          TRANCODE: II       INQUIRY: PDMW;;


APPL    DIRECTORY     DATE      TIME     MAPNAME  DBDNAME  DB_NAME  DB_TYPE 1
- - - -  - - - -   - - - - - - - - - - - - - - -  - - - -  - - - -  - - - -
II       IIDMDIR   10/07/1995 13.36.44.7 IIDMMAP  IIDBDDM  PLANT    IMS
II       IIDMDIR   10/07/1995 13.36.44.7 IIDSMAP  IIDBDDS  SKILL    IMS
*
*
*
*
*
*
*
*
*
*
*
IXX9121  END OF INQUIRY.                        (0,0 USER DB CALLS, ROOTS)
```

Figure A-13 DISPLAY DIRECTORY MAP WHOLE Command

Following are brief descriptions of the entries in Figure A-13; for more information, see the *Advantage VISION:Inquiry for IMS and TSO Technical Reference Guide*.

APPL	The VISION:Inquiry transaction code (application name).
DIRECTORY	The directory name (primary or connected) of the terminal/user.
DATE	Last date that the directory was (re)defined.
TIME	Last time that the directory was (re)defined.
MAPNAME	The internal name of the database/file.
DBDNAME	The DBD name for the IMS database, the authorization ID of the DB2 table/view, or the ddname of the VSAM file.
DB_NAME	The name used in the inquiry to refer to the database/file.
DB_TYPE	The type of database/file (for example, IMS, DB2, VSAMKSDS).

Appendix

B Quick Reference

You can use this appendix as a quick reference for VISION:Inquiry commands. It contains the syntax for commands and keywords for IMS (DL/I) and DB2 databases, and for VSAM files.

VISION:Inquiry uses free-form statements. Most commands may be used with other commands. The section Commands/Statements Syntax on page B-9 illustrates, at a minimum, what is required to use for that particular command or statement.

Notation Rules

This summary uses the following standard notation in defining VISION:Inquiry statements:

- The ‘or’ symbol (the character |) separates alternatives. Only one alternative may be selected.
- An underscore (_) indicates a default option.
- Braces ({ }) are used to group related items that are alternatives. One item must be chosen.
- Brackets ([ ]) are used to indicate optional items. Everything within the brackets is optional and may be omitted.
- Upper-case letters and special characters are specified exactly as shown.
- Lower-case letters require a specific value to be substituted in their place.
- Parentheses must be entered exactly as shown.
- Keywords and operands between a set of **!** and **!**... indicate that repetition of what is between the **!** and **!**... is allowed.

Terms

VISION:Inquiry uses the following terms:

Continuous Versus Conversational Mode

There are two modes of operation in VISION:Inquiry:

- Continuous mode: the output generated by an inquiry is returned to the terminal without interruption.
- Conversational mode: the amount of output returned to the terminal is limited by the number of calls made to the database/file (TIME limit) or the number of pages that are output (PAGE limit).

UDO Versus Non-UDO

There are two types of inquiries in VISION:Inquiry:

- UDO (User Defined Output): the placement of the output of the inquiry is controlled by the user. UDO inquiries are specified with DISPLAY FORMAT commands.
- Non-UDO: the placement of the output of the inquiry is done by the product automatically. Non-UDO inquiries are specified by simple DISPLAY commands (without FORMAT).

PAGE and TIME Limit Checkpoint

The system administrator can set up limits in the processing of inquiries for each user in the conversational mode of operation. The processing of the inquiry stops and an internal checkpoint is taken whenever any limit is reached. The user can then continue or defer processing using the checkpoint taken by the product. The limits are:

- PAGE limit: limits the number of output pages.
- TIME limit: limits the number of calls to the database/file.

Stored Function

Arithmetic expressions can be stored in the system database for further use. A stored function can be called by name and integrated into inquiries to perform the calculations whenever needed.

Stored Inquiry

Inquiries can be stored in the system database for further use. A stored inquiry can be called by name and executed later whenever the output is needed.

System Database

The nucleus of the system; maintains all the elements and items required to interpret and process the inquiries.

System Names

System names are the reserved words used in UDO inquiries and are available for display only. These names are:

- DATE: Supplies the current date in the form MM/DD/YY.
- DATEF: Supplies the current date in the form MM/DD/YYYY.
- PAGE: Supplies the page numbers for the inquiry output.
- TIME: Supplies the current time in the form HH:MM:SS.

Temporary Fields

Fields defined in an inquiry temporarily. Temporary fields exist only during the processing of the inquiry. They cannot be accessed by other inquiries. The value of arithmetic expressions or fields can be assigned to a temporary field. One of the main uses of temporary fields is to pass values from one statement to another when accessing multiple databases/files.

Text Editor

The VISION:Inquiry facility used to edit stored inquiries online.

Native SQL Support

The native SQL syntax facility allows you to use embedded SQL SELECT statements in your inquiries. Many programmers and database administrators already know SQL and prefer to use it rather than VISION:Inquiry syntax. This facility also allows you to use DB2 features not supported under VISION:Inquiry syntax.

Commands/Statements Summary

AVERAGE, COUNT, TOTAL

Each command displays a grand or subsummary. These commands may also be used with the DISPLAY command.

COLUMN

Positions the data across the page in UDO inquiries.

CONTINUE

Continues the processing of an inquiry in conversational mode when the page or time limit checkpoint has occurred.

CONTINUE DEFERRED INQUIRY (CDI)

Continues the processing of a deferred inquiry in conversational mode.

DEFER

When the page or time limit checkpoint has occurred, defers until a later time, the processing of an inquiry in conversational mode.

DEFINE DIRECTORY FUNCTION (DDF)

Stores an arithmetic expression (function) in the system database as the function name. The stored function can then be called by name in an inquiry for processing.

DEFINE DIRECTORY INQUIRY (DDI)

Stores an inquiry in the system database as the inquiry name. The stored inquiry can then be called by name for processing.

DELETE DEFERRED INQUIRY

Deletes a deferred inquiry when no further processing or viewing of output is needed from the system.

DELETE DIRECTORY FUNCTION

Deletes a stored function from the system database.

DELETE DIRECTORY INQUIRY

Deletes a stored inquiry from the system database.

DISPLAY

Displays data from specified fields. In non-UDO syntax (DISPLAY without FORMAT keyword), VISION:Inquiry arranges the output automatically.

DISPLAY DIRECTORY DATA (PDD)

Displays the characteristics of the data items of the specified map (database).

DISPLAY DIRECTORY DATA DESCRIPTION (PDDDS)

Displays the description of the specified map (database) or the field.

DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE)

Displays the characteristics and description of the data items of the specified map (database).

DISPLAY DIRECTORY FUNCTION (PDF)

Displays the content of the stored function specified in the command. Another option is displaying all the stored functions for a specified database.

DISPLAY DIRECTORY INQUIRY (PDI)

Displays the content of a stored inquiry for a user database specified in the command. Another option is displaying all the stored inquiries for a specific user database.

DISPLAY DIRECTORY INQUIRY COMMENT (PDIC)

Displays the information about a stored inquiry such as name, related database, date and time created or last modified, and comment. Another option is displaying the information of all the stored inquiries for a specific user database.

DISPLAY DIRECTORY INQUIRY COMMENT WHOLE (PDICW)

Displays the information such as name, related database, date and time created or last modified, and comment about the stored inquiries for all the user databases in the system database.

DISPLAY DIRECTORY INQUIRY WHOLE (PDIW)

Displays the stored inquires for all user databases in the system database.

DISPLAY DIRECTORY LTERM (PDL)

Displays the characteristics of the terminals defined in the system database.

DISPLAY DIRECTORY TERM (PDT)

Displays the characteristics of the terminals defined in the system database.

DISPLAY DIRECTORY MAP (PDM)

Displays the characteristics of the specified map (database).

DISPLAY DIRECTORY MAP WHOLE (PDMW)

Displays the characteristics of all the directories or maps (databases) in the system database or all the maps of a specific directory depending on the options specified in the command.

DISPLAY DIRECTORY VOCABULARY (PDV)

Displays the names that are included in the vocabulary.

DISPLAY FORMAT

The FORMAT keyword, in conjunction with DISPLAY command, identifies a UDO inquiry. It displays data from the specified fields in the specified arrangement.

EDIT

Formats a numeric field according to the specified option in UDO inquiries.

EDITSQ

Displays the specified stored inquiry content in Text Editor mode for online editing.

EXTRACT

Extracts data from a database/file and writes it to a sequential data set.

FIND

Allows two or more databases/files to be accessed in one inquiry.

IF

Designates the start of a compare statement. Can be used for:

- Relational test, such as equal to or greater than.
- Positional test which allows the selection of the first or last occurrence of a segment in IMS databases.
- Existence test which determines the presence of a segment in IMS databases.
- Compound test which is a combination of the above conditions.

LIMIT

Limits the number of output lines displayed. Use the LIMIT command with the FIND, DISPLAY, EXTRACT, or PCEXTRACT command. (PCEXTRACT is generated by VISION:Journey.)

LINE

Controls the movement of data down the page in UDO inquiries.

NOSPACE

Suppresses the default two-position spacing in UDO inquiries.

OUTPUT

Sends the output of an inquiry to another display terminal, printer, or a PC file. Use the OUTPUT command with the DISPLAY command.

PCDELETE

Deletes the data before downloading to the PC from the download data set. This command applies to an online environment and is generated by VISION:Journey for Windows.

PCEXTRACT

Extracts and downloads data from the host server to a PC file. This command applies to an online environment and is generated by VISION:Journey for Windows.

PCLOAD

Downloads the partial data from the host server to a PC file when a page or time limit checkpoint has occurred. This command applies to an online environment and is generated by VISION:Journey for Windows.

PF

Allows part of a character field to be displayed in UDO inquiries.

REPEAT

Specifies the number of segment occurrences displayed from the database in UDO inquiries.

SHOW

Allows you to see what is in the buffer after a time limit checkpoint has occurred in conversational mode, for a UDO inquiry.

SKIP

Controls the movement of data down the page in UDO inquiries.

SORT

Sequences output data into a specified order. Use the SORT command with the DISPLAY or PCEXTRACT command. (PCEXTRACT is generated by VISION:Journey.)

SPACE

Positions data across the page in UDO inquiries.

SUM

Creates a subtotal of the specified field or arithmetic expression. Use the SUM command with the DISPLAY command.

Commands/Statements Syntax

Arithmetic Expression

(temporary field name =)

```
        (!(!...)
        operand-1!{+|-|*|/}(!(!...)
        operand-2 (!)!...)!... (!)!...))
```

- The **temporary field name** can be up to 32 characters and must start with a % sign.
- **Operand-1** and **operand-2** may each be a numeric field name or numeric constant. There must be at least one reference to a field name.
- The **order of execution** of the operators is from left to right, with multiplication and division performed before addition and subtraction. Expressions in parentheses are executed first.

Examples:

```
SAL.YTD - SAL.DED
```

```
%EXPR1 = ((SAL.YTD + 200) * 10) /100
```

AVERAGE, COUNT, TOTAL

{AVERAGE | COUNT | TOTAL}

(database name) !{operand-1 |
(group field name operand-1)}!...(IF statement); (;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- **Operand-1** can be a field name (its type must be numeric for TOTAL and AVERAGE commands), temporary field name, or arithmetic expression.
- If **group field name** syntax is used, the group field will be the control break field; when the value of the control break field changes, the subtotal is displayed. The group field and **operand-1** must be in the same leg of the database if an IMS database is used. (A leg is the pathway formed by the root segment and its dependent segments.)
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples

```
AVERAGE PLANT SAL.YTD * 1.1;

COUNT PLANT (PLANT.ID EMP.NAME) IF PLANT.ID > 30000;

TOTAL PLANT SAL.YTD (PLANT.ID SAL.YTD) SAL.DED * 1.1;
```

DISPLAY (Non-UDO)

{DISPLAY | D | PRINT} (database name)

(title-1)
(title-2)
!operand-1!...
(!summary statement!...)
(!SORT statement!...}
(LIMIT statement)
(OUTPUT statement)
(IF statement);(;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- Title-1 and Title-2 can be up to 70 character constants that appear at the top of each page of the report.
- **Operand-1** can be a field name, temporary field name, stored function, arithmetic expression, or SUM statement.
- **Summary** commands are AVERAGE, COUNT, and TOTAL.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples:

```
D PLANT.ID %NET = SAL.YTD - SAL.DED SORT PLANT.ID IF %NET > 30000;

D PLANT PLANT.ID EMP.NO SAL.YTD TOTAL SAL.DED
        AVERAGE (PLANT.ID  SAL.YTD) OUTPUT 'TERM' 'INQMAP1';

D PLANT ‘SALARY INFORMATION’ %NAME=EMP.NAME SAL.YTD - SAL.DED IF
        (SAL.YTD > 30000 OR PLANT.REGION = 'NE') AND ED.DEGREE > 'BA'
        LIMIT 5;
```

DISPLAY FORMAT (UDO)

```
{DISPLAY | D | PRINT} FORMAT (database name)

            (!LINE!... | LINE (n1)) ({COLUMN | COL} n2)
            (!SKIP!... | SKIP (n3))  (SPACE | SP (n4)) (NOSPACE | NOSP)
               !{operand-1 |
            {REPEAT | REP} n5(!operand-2!...) |
                PF (operand-3 start length) |
                EDIT(operand-4 {'N' | 'F' | 'Z'})}!...
            (!SORT statement!...)
            (LIMIT statement)
            (OUTPUT statement)
            (IF statement);(;)
```

- If the **database name** is not entered, the default is the first database listed in the directory.
- **n1** is an integer from 1 to the page length. The default is next line.
- **n2** is an integer from 1 to the page width. If COLUMN n2 command is omitted, the default is 2 spaces between the output, except at the beginning of a line, where the output starts in column 1.
- **n3** is an integer from 0 (next line) to the page length minus 1. The default is 0.
- **n4** is an integer from 0 to the page width minus 1. The default is 1.
- **Operand-1** can be a field name, temporary field name, stored function, arithmetic expression, SUM statement, summary statement (AVERAGE, COUNT, and TOTAL), system name, or character constant within quotation marks.
- **n5** is an integer from 0 to the number of occurrences in the segment. If 0 is entered, all occurrences are displayed. Nested REPEAT commands are allowed.
- **Operand-2** can be a field name, temporary field name, or arithmetic expression.
- **Operand-3** can be a non-numeric field name or temporary field name.
- **Operand-4** can be a numeric field name or temporary field name.
- **EDIT** command options are **'N'** (no editing), **'F'** (full editing of decimal point, comma placement, leading zero suppression), and **'Z'** (leading zero suppression and decimal point placement).
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples:

```
DISPLAY FORMAT PLANT LINE 1 COL 30 'THIS IS TITLE LINE' COL 60 DATE
        LINE PLANT.ID SP 10 EMP.NO EMP.NAME SORT PLANT.ID
        IF PLANT.ID > 20150;

DISPLAY FORMAT PLANT LINE 1 PAGE SPACE 20 'TITLE' COL 60 TIME
        SKIP SKIP PLANT.ID COL 20 PF(PLANT.PHONE 1 3) NOSP '-' NOSP
        PF(PLANT.PHONE 4 4);

DISPLAY FORMAT PLANT PLANT.ID REPEAT 3 (EMP.NO) EDIT(SAL.YTD 'F')
        IF SAL.YTD > 29000 AND ED.DEGREE >= 'BA' OUTPUT 'PRT1';
```

EXTRACT

{EXTRACT | E} (database name)

```
          !operand-1!...
             (LIMIT statement)
             (IF statement);(;)
```

- If the **database name** is not entered, the default is the first database listed in the directory.
- **Operand-1** may be a field name, temporary field name, stored function, or arithmetic expression.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Example:

```
E PLANT.ID EMP.NAME SAL.YTD IF SAL.YTD > 29000;
```

FIND

!{FIND | I | INTER}(database name)

```
!{temporary field name= {field name |
    arithmetic expression}}!... (IF statement);!...
  {DISPLAY statement | DISPLAY FORMAT statement |
   EXTRACT statement | PCEXTRACT statement};(;)
```

- **FIND** command may be used up to 15 times within one inquiry.
- If the **database name** is not entered, the default is the first database listed in the directory.
- The **temporary field name(s)** in the FIND statement(s) may be used in the subsequent statement(s) as the match field(s) in the IF statement and/or in the last statement as a display/extracted field.
- The **PCEXTRACT** command applies to VISION:Journey.
- The **second semicolon** at the end of the statement is required in Batch and TSO.

Examples:

```
FIND SKILL %EMP=EMP.NO %SKILL = SKILL.CODE IF PLANT.ID > 20150;

DISPLAY PLANT PLANT.ID EMP.NO %SKILL IF EMP.NO = %EMP
        AND SAL.YTD > 29000;
```

IF

Conditional selection includes relational conditions, existence conditions, positional conditions, and compound conditions.

Relational condition

```
{IF | WITH}  operand-1  {EQUAL | EQ | =}   operand-2   ESCAPE ‘character’

                        {NE   | ¬=   }
                        {GT   | >    }
                        {GE   | >=   }
                        {LT   | <    }
                        {LE   | <=   }
                        {LIKE}
```

- **Operand-1** and **operand-2** can each be a field name, arithmetic expression, temporary field name, numeric constant, or literal constant between quotation marks except for the LIKE operator that **operand-1** and **operand-2** must be a character field and literal constant between quotation marks, respectively. There must be at least one reference to a field name. Fields within relational conditions must be from the same segment. The literal constant used as **operand-2** with the LIKE operator should contain the special characters, % or _. If neither of the special characters used in the literal constant, the LIKE operator will be treated as EQUAL operator. ESCAPE keyword is used with the LIKE operator and is followed by a single character between quotation marks. Escape keyword will be ignored if no LIKE operator is used in the IF selection.

Existence Condition

{IF | WITH} field name

Positional Condition

{IF | WITH} field name {FIRST | LAST}

- **Existence** and **positional** conditions are not supported for VSAM non-hierarchical files and DB2 databases.

Compound Condition

{IF | WITH} (!(!...) condition !{{AND | &} | {OR | |}}

(!(!...) condition (!)!...)!... (!)!...)

- **OR** operator is not allowed in the compound condition containing a positional condition.
- **Conditions in parentheses** are evaluated first. The order of evaluation of operators is **AND** and then **OR**.

Examples:

```
IF EMP.SEX = 'F'
   IF SAL.YTD LAST
   IF PROD.CODE
   IF PLANT.REGION = 'NE' & (SAL.YTD * 1.1 > 32500 | ED.DEGREE)
```

LIMIT

LIMIT n

- **LIMIT** can not be used with the SORT command and is applied to the referenced field that is in the highest level segment. If two fields are referenced that exist on the same level but in different segments, the LIMIT is applied to the leftmost segment field.
- **n** is an integer value from 1 to the total number of occurrences of the field to be displayed.

OUTPUT

OUTPUT 'device name' ('IMS MFS name' | 'CICS BMS mapset name')

- **'device name'** may be another terminal, system printer, or a hard copy slave printer. It is limited to 8 characters for IMS and 4 characters for CICS. The device name can also be a dummy name. In this case, the output is sent to a PC file and you must have VISION:Journey for Windows installed with VISION:Inquiry.

Examples:

```
OUTPUT 'TERM'
```

```
OUTPUT 'PRT1' 'INQMAP1'
```

PCEXTRACT

{PCEXTRACT | PCE} 'pc filename' (database name)
!operand-1!...
(!SORT statement!...)
(LIMIT statement)
(IF statement);

- The **PCEXTRACT** command only applies to VISION:Journey.
- **'pc filename'** must be a standard DOS PC file name and may include a path.
- If the **database name** is not entered, the default is the first database listed in the directory.
- **Operand-1** may be a field name, temporary field name, stored function, or arithmetic expression.

Example:

```
PCE 'C:\JOURNEY\NAME1.TXT' PLANT PLANT.ID EMP.NO
        %NET = SAL.YTD - SAL.DED IF %NET > 25000;
```

SORT

SORT !(<u>ASC</u> | DSC) operand-1!...

- **SORT** may not be used with the LIMIT command.
- **ASC** - sort in ascending order.
- **DSC** - sort in descending order.
- **Operand-1** may be a field name, arithmetic expression, or a temporary field name.

Examples:

```
SORT PLANT.ID

SORT PLANT.ID DSC EMP.NO

SORT DSC PLANT.ID EMP.NO
```

In third example, **DSC** applies to both fields.

SUM

(temporary field name) SUM(operand-1)

- The **temporary field name** can be up to 32 characters and must start with % sign.
- A **SUM** statement cannot call the results of another SUM statement. The SUM command is not supported for VSAM non-hierarchical files and DB2 databases.
- **Operand-1** can be a numeric field name, arithmetic expression, or a temporary field name.

Examples:

```
SUM(SAL.YTD - SAL.DED)

%TEMP1 = SUM(SAL.YTD)
```

Stored Inquiries/Functions

You can define, display, edit, delete, and run stored inquiries and functions.

DEFINE DIRECTORY INQUIRY (DDI)

{DEFINE {DIRECTORY | DIRECT} INQUIRY | DDI} (database name)

'inquiry name (,comment)' (as) inquiry ; (;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- The first character of **'inquiry name'** must be alphabetic.
- **[as] inquiry** is stored in the directory with **'inquiry name'** assigned to it. In IF statements, variable values may be represented in the inquiry by using %1 through %n; actual values may be substituted when the inquiry is executed.

The actual values are assigned from left to right to the variables. LIMIT and OUTPUT commands cannot be stored in an inquiry. Stored functions may be part of a stored inquiry.

- The **second semicolon** at the end of the statement is required in Batch and TSO.

Examples:

```
DEFINE DIRECTORY INQUIRY PLANT 'ROSTER,PLANT AND EMPLOYEE REPORT' AS
        DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME;

DDI PLANT 'PART.ROSTER' DISPLAY PLANT PLANT.ID EMP.NAME
        IF PLANT.ID = %1;
```

DEFINE DIRECTORY FUNCTION (DDF)

{DEFINE {DIRECTORY | DIRECT} FUNCTION | DDF} (database name)

'function name' (as) function ; (;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- The first character of **'function name'** must be alphabetic.
- **[as] function** is an arithmetic expression (function) which is stored in the directory with '**function name'** assigned to it.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples:

```
DEFINE DIRECTORY FUNCTION PLANT 'BONUS' AS SAL.YTD * 1.1;

DDF PLANT 'MON.SAL' %MONTH.SAL = (SAL.YTD - SAL.DED) / 12;
```

DISPLAY DIRECTORY INQUIRY (PDI)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY | PDI}

(database name) ('inquiry name') ; (;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- The statement displays the stored inquiry specified in **inquiry name**; if the inquiry name is not entered, the default is all the inquiries for the database.

Examples:

```
DISPLAY DIRECTORY INQUIRY PLANT 'ROSTER';
```

```
DISPLAY DIRECTORY INQUIRY PLANT ;
```

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY WHOLE | PDIW}; (;)

- The statement displays all the inquiries stored for the transaction.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Example:

```
DISPLAY DIRECTORY INQUIRY WHOLE;
```

DISPLAY DIRECTORY INQUIRY COMMENT (PDIC)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY COMMENT | PDIC}
(database name) ('inquiry name');(;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- The statement displays the stored inquiry information and comment for the specified in **inquiry name**; if the inquiry name is not entered, the default is all the inquiries for the database.

Examples:

```
DISPLAY DIRECTORY INQUIRY COMMENT PLANT 'ROSTER';
```

```
PDIC PLANT;
```

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY COMMENT WHOLE | PDICW}; (;)

- The statement displays all the inquiry information and comments stored for the transaction.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Example:

```
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE;
```

DISPLAY DIRECTORY FUNCTION (PDF)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} FUNCTION | PDF}

(database name) ('function name') ; (;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- **'function name'** specifies the stored function to be displayed; if 'function name' is not entered, the default is all the stored functions for the database.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples:

```
DISPLAY DIRECTORY FUNCTION PLANT 'BONUS';
```

```
DISPLAY DIRECTORY FUNCTION PLANT;
```

EDITSQ

EDITSQ (database name) 'inquiry name' ;

- If the **database name** is not entered, the default is the first database listed in the directory.
- This command is not available in batch, TSO and BMP environments.

Example:

```
EDITSQ PLANT 'ROSTER';
```

DELETE DIRECTORY INQUIRY and DELETE DIRECTORY FUNCTION

DELETE {DIRECTORY | DIRECT} INQUIRY (database name) 'inquiry name' ; (;)
DELETE {DIRECTORY | DIRECT} FUNCTION (database name) 'function name' ; (;)

- The statements delete the stored inquiry or function specified in **'inquiry name'** or **'function name'**.
- If the **database name** is not entered, the default is the first database listed in the directory.
- The **second semicolon** at the end of the statements is required in batch and TSO.

Examples:

```
DELETE DIRECTORY INQUIRY PLANT 'ROSTER';
```

```
DELETE DIRECTORY FUNCTION 'BONUS';
```

Running Stored Inquiries

inquiry name (database name) (!constant!...); (;)

- **name** is the name of the inquiry stored in the directory.
- If the **database name** is not entered, the default is the first database listed in the directory.
- **Character constants** must be enclosed in single quotation marks. The constant is used only as a substitution value.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples:

```
ROSTER PLANT;
```

```
DEGREE.SALARY PLANT 'BA' 25000;
```

Using Stored Functions

Stored functions may be used in DISPLAY, DISPLAY FORMAT, EXTRACT, and PCEXTRACT statements. See each statement for the syntax.

Example:

```
DISPLAY PLANT PLANT.ID EMP.NAME BONUS IF EMP.SEX = 'M';
```

- **BONUS** is the name of a stored function.

Terminal Commands

Starting VISION:Inquiry

```
{/FORMAT | /FOR} MFS module name
```

Starts the IMS version of VISION:Inquiry. The name of the command is installation dependent.

or

```
transaction id
```

Starts the CICS version of VISION:Inquiry. The name of the command is installation dependent.

Examples:

```
/FORMAT INQIMS
```

```
IQIO
```

CONTINUE

{CONTINUE | PF1 key | #}

(OUTPUT 'device name'
('IMS MFS name' | 'CICS BMS mapset name')) ; (;)

Entered after “time limit exceeded” or “page end” message is displayed to indicate that the processing of the current inquiry should continue.

DEFER

{DEFER | PF2 key | DI} ; (;)

Entered after “time limit exceeded” or “page end” message to defer processing of the current inquiry until a later time. An ID number is assigned to the inquiry; make a note of it for later use.

- The **PF key** is not supported for CICS.
- The **second semicolon** at the end of the statement is required in batch and TSO.

CONTINUE DEFERRED INQUIRY (CDI)

{{CONTINUE | #} DEFERRED INQUIRY | PF3 key | CDI} id number ; (;)

The specified deferred inquiry should now be continued.

- The **PF key** is not supported for CICS.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Examples

```
CONTINUE DEFERRED INQUIRY 2;
```

```
CDI 3;
```

DELETE DEFERRED INQUIRY

DELETE DEFERRED INQUIRY id number ; (;)

Deletes the deferred inquiry specified by **id number** from the system.

- The **second semicolon** at the end of the statement is required in batch and TSO.

Example:

```
DELETE DEFERRED INQUIRY 2;
```

SHOW

SHOW ; (;)

- Displays the accumulated screen output from a UDO inquiry not otherwise displayed because a full screen of data was not yet available and a checkpoint timeout occurred causing the message INQUIRY TIME LIMIT EXCEEDED to appear.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Directory Commands

DISPLAY DIRECTORY MAP (PDM)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} MAP | PDM} (database name) ;(;)

- If the **database name** is not entered, the default is the first database listed in the directory.
- The **second semicolon** at the end of the statement is required in batch and TSO.

DISPLAY DIRECTORY MAP WHOLE (PDMW)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} MAP WHOLE | PDMW}

(' ' | 'directory name') ; (;)

- The **second semicolon** at the end of the statement is required in batch and TSO.

The output is:

- The list of all the directories and databases available to the terminal/user through the primary directory and the connected directories (if any exist) if no operand is specified
- The list of all the directories for the transaction if literal constant ' ' is specified as operand
- The list of all the databases associated with the specified directory for the transaction (and the connected directories, if any exist) if the directory name is specified as operand.

Examples:

```
DISPLAY DIRECTORY MAP WHOLE;
```

```
PDMW ' ';
```

```
PDMW 'IIDMDIR';
```

DISPLAY DIRECTORY VOCABULARY (PDV)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT}

{VOCABULARY | VOCAB} |PDV} ; (;)

- The **second semicolon** at the end of the statement is required in batch and TSO.

DISPLAY DIRECTORY DATA (PDD & PDD WHOLE)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} DATA (WHOLE) | PDD (

WHOLE)}

(database name) ('field name') ; (;)

- The statement displays the characteristics of the data items starting from the specified **'field name'**. If a field name is omitted, the characteristics of all the data items for the specified database will be displayed.
- If the **WHOLE** keyword is used with the PDD command, it also displays the field description along with the characteristic of the field.
- If the **database name** is not entered, the default is the first database listed in the directory.
- The **second semicolon** at the end of the statement is required in batch and TSO.

DISPLAY DIRECTORY DATA DESCRIPTION (PDDDS)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} DATA DESCRIPT | PDDDS}

(database name) (field name) ; (;)

- The statement displays the description of the field specified in **'field name'**. If a field name is not specified, the description of the database is displayed.
- If the **database name** is not entered, the default is the first database listed in the directory.
- The **second semicolon** at the end of the statement is required in batch and TSO.

DISPLAY DIRECTORY TERMINAL (PDT)

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT}

{LTERM | TERM} | {PDL} | {PDT}} ('terminal name') ; (;)

- The statement displays the terminal characteristics of the terminal specified in **'terminal name'**. If the **'terminal name'** is omitted, the characteristics of all the terminals will be displayed.
- The **second semicolon** at the end of the statement is required in batch and TSO.

Native SQL Commands/Statement Syntax

SELECT Statement

Any valid DB2 SELECT statement.

DISPLAY

DISPLAY EXECSQL SELECT statement ENDEXEC

(LIMIT command) (OUTPUT command) ; (;)

Displays the result of the SELECT statement in non-UDO format.

EXTRACT

EXTRACT EXECSQL SELECT statement ENDEXEC
(LIMIT command) ; (;)

Writes the result of the SELECT statement into the EXTRACT data set.

DEFINE DIRECTORY INQUIRY

{DEFINE {DIRECTORY | DIRECT} INQUIRY | DDI} 'inquiry name
(,comment)' (as) native SQL syntax inquiry ; (;)

DISPLAY DIRECTORY INQUIRY

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY | PDI} EXECSQL
(inquiry name) ENDEXEC ; (;)

DISPLAY DIRECTORY INQUIRY COMMENT

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} INQUIRY COMMENT | PDIC}
EXECSQL (inquiry name) ENDEXEC ; (;)

DELETE DIRECTORY INQUIRY

DELETE {DIRECTORY | DIRECT} INQUIRY EXECSQL inquiry name ENDEXEC ; (;)

EDITSQ

EDITSQ EXECSQL inquiry name ENDEXEC ;

Running Native SQL Syntax Inquiries

EXECSQL inquiry name ENDEXEC (LIMIT command) (OUTPUT command) ; (;)
Executes the SQL syntax inquiry stored in the directory as inquiry name.

DISPLAY DIRECTORY DATA

{{DISPLAY | D | PRINT} {DIRECTORY | DIRECT} DATA | PDD}

EXECSQL DB2 table name (ORDER BY) ENDEXEC; (;)

Displays the characteristics of the columns of the specified DB2 table in the order defined or sorted by the column name if the "ORDER BY" statement is specified.

- The **second semicolon** at the end of the statements above is required in batch and TSO.

Noise/Reserved Words

Noise Words

A	FROM	IS	THAN
AN	HAD	NO	THE
ARE	HAS	NONE	TO
AS	HAVE	OF	WAS
BY	IN	ON	WERE

RESERVED WORDS

DATE	PAGE
DATEF	TIME
EXCLUDE	EXECSQL
INCLUDE	ENDEXEC
ESCAPE	

Command/Keyword Summary

Verbs

Command/Keyword	Synonym	Symbol
AVERAGE		
COUNT		
DISPLAY	D, PRINT	
EXTRACT	E	
FIND	I, INTER	
LIMIT		
OUTPUT		
PCDELETE	PCD	
PCEXTRACT	PCE	
PCLOAD	PCL	
SORT ASC DSC		
SUM		
TOTAL		

- The **PCDELETE**, **PCEXTRACT,** and **PCLOAD** commands apply to VISION:Journey.
- **SUM** is not supported for VSAM files and DB2 databases.

Stored Inquiry and Function Commands

Command/Keyword	Synonym
DEFINE DIRECTORY INQUIRY	DDI
DEFINE DIRECTORY FUNCTION	DDF
DELETE DIRECTORY INQUIRY	
DELETE DIRECTORY FUNCTION	
DISPLAY DIRECTORY INQUIRY	PDI
DISPLAY DIRECTORY INQUIRY WHOLE	PDIW
DISPLAY DIRECTORY INQUIRY COMMENT	PDIC
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE	PDICW
DISPLAY DIRECTORY FUNCTION	PDF
EDITSQ	

TERMINAL COMMANDS

Command/Keyword	Synonym	Symbol
CONTINUE		#
CONTINUE DEFERRED INQUIRY	CDI	
DEFER	DI	
DELETE DEFERRED INQUIRY		
SHOW		

DIRECTORY COMMANDS

Command/Keyword	Synonym
DISPLAY DIRECTORY DATA	PDD
DISPLAY DIRECTORY DATA WHOLE	PDD WHOLE
DISPLAY DIRECTORY DATA DESCRIPTION	PDDDS
DISPLAY DIRECTORY MAP	PDM
DISPLAY DIRECTORY MAP WHOLE	PDMW
DISPLAY DIRECTORY TERM	PDT
DISPLAY DIRECTORY LTERM	PDL
DISPLAY DIRECTORY VOCABULARY	PDV

OPERATORS

Use only the symbol for **Division**, **Addition**, **Semicolon**, **Subtraction**, **Multiplication**, **Left Parenthesis**, and **Right Parenthesis**.

Operator / Abbreviation	Synonym	Symbol
Equal	EQ	=
Less Than / LT	LT	<
Greater Than / GT	GT	>
Lesser or Equal To / LE	LE	<=
Greater or Equal To / GE	GE	>=
Not Equal To / NE	NE	¬=
LIKE	LIKE	
Division		/
Addition		+
Semicolon		;
Subtraction		-
Multiplication		*
Left Parenthesis		(
Right Parenthesis		)

UDO COMMANDS and KEYWORDS

Command / Keyword	Synonym
COLUMN	COL
EDIT	
FORMAT	
LINE	
NOSPACE	NOSP
PF	
REPEAT	REP
SHOW	
SKIP	
SPACE	SP

Conditional Selection

Command/Keyword	Synonym	Symbol
AND		&
FIRST		
IF	WITH	
LAST		
OR		|
ESCAPE		

FIRST and **LAST** are not supported for VSAM files and DB2 databases.

ESCAPE keyword is only supported with the LIKE operator.

Text Editor Commands

The Text Editor facility of VISION:Inquiry is controlled with two types of commands: primary commands, and line commands. Each type is listed below with its equivalent or action keys.

Upper-case letters represent the minimum that is required for each command. Some commands have equivalent PF/PA keys, shown in the syntax along with the command.

Primary Commands

Command	Equivalent key(s)
REFResh	PA1, PA2, or PA3 (CICS version only)
HElp	PF1
RUn	PF2
EXit or =X	PF3
SAVE/Run	PF4
Save	PF9
PAss	PF10
SAVE/Pass	PF11
EDit/next	PF12
U #	# with PF7
D #	# with PF8
PF	
PB	
PL	
HF	
HB	
Find string	
Change old string new string [ALL]	

Strings in **FIND** and **CHANGE** commands must be enclosed by spaces, or single or double quotation marks. After the FIND or CHANGE command, PF5 and PF6 may be used for repeat find (**RFIND**) or change (**RCHANGE**) respectively.

Line Commands

Command	Action / Keys
Insert commands	Inserts one or more blank data entry lines. I I#
Delete commands	Deletes one or more lines. D D# DD
Copy commands	Copies one or more lines to the location specified by target line identifier. C C# CC
Move commands	Moves one or more lines to the location specified by target line identifier. M M# MM
Repeat commands	Repeats one or more lines. R R# RR
Target line identifiers	Identifies the target line (after or before) for COPY and MOVE commands. A B

Index

Symbols

/FORMAT, 4-6
/FORMAT INQIMS, 4-2
/FORMAT INQUDO, 11-5, 11-51

A

Advantage Intraccess, 1-4
arithmetic calculations, 7-3
  example, 7-7, 7-8
  with constants, 7-4
arithmetic expressions, B-9
arithmetic operators, 3-7, 7-4
arithmetic processing, 11-35
ASC, A-1, B-28
AVERAGE, 1-4, 6-7, 6-12, A-2, B-4, B-10, B-28
  description, 6-7
  with DISPLAY, 6-8
  with UDO, 11-29

B

Basic Mapping Support (BMS), 4-8
batch, 3-4, 4-17
  inquiry processing, 4-17
  message inquiry processing, 4-17
  terminate with, 4-17
BMS mapset name, 5-49
BMS paging, 4-8

C

calling statement, 10-5
CDI (CONTINUE DEFERRED INQUIRY), A-4
changing, 9-7
  stored inquiries, 9-7
character constants, 10-3
  surround with single quotation marks, 10-3
checkpoint, 4-12, B-2
Clear key, 10-13
  for CICS, 10-13
  for IMS, 10-13
collection of tables, 2-16
COLUMN (COL), 11-9, 11-47, 11-48, A-3, B-4
  with UDO, 11-2
column heading, 10-30
  decimal point heading, 7-6
  example, 9-10
  stored inquiries, 10-27
  using temporary fields, 7-1
  with functions, 9-9
columns, 2-16
commands, 1-4, 3-1
  AVERAGE, 1-4, 6-7
  CONTINUE, 4-13
  COUNT, 1-4, 6-5

DEFER, 4-14
DELETE, 9-28
DISPLAY, 1-4, 5-1
EDITSQ, 1-4
EXTRACT, 1-5
FIND, 1-5
FORMAT, 1-5
LIMIT, 1-5, 5-39
OUTPUT, 1-5, 5-44, 5-49
SORT, 1-5
SUM, 1-5
TOTAL, 1-5, 6-1

comments, 9-4

conditional selection, 3-2, 5-12, B-15
any-all, 8-20
compound, 5-18, B-15
description, 5-12
example, 5-13, 5-26
existence, B-15
IF, 5-12
positional, B-15
relational, B-15
UDO, 11-20
with AND, 5-22
with DISPLAY, 8-17
with FIND, 8-7
with result fields, 8-29

constant, 3-1
character, 3-3, 3-5, 5-27
numeric, 3-6

continuation of inquiries, 3-3

CONTINUE, 4-13, 4-15, A-3, B-4, B-22, B-29

CONTINUE DEFERRED INQUIRY (CDI), 4-16, A-4, B-4, B-22, B-29

continuous mode, 4-12, 4-17, B-2

control break, 5-38, 11-30
with subtotals, 6-12

control field, 5-3

controlling page breaks
UDO, 11-17

conversational mode, 4-12, 4-13, B-2

COUNT, 1-4, 6-5, 6-12, A-2, B-4, B-10, B-28
description, 6-5
with DISPLAY, 6-6
with UDO, 11-29

D

database, 1-1, 2-1
access, 2-6, 8-1
definition, 1-6, 2-1
elements, 2-1
hierarchical, 2-5
legs, 2-6
multiple access, 8-20, 8-24, 8-26
PLANT, 5-5
terms, 2-16

DATE, 11-1, 11-18, 11-49, B-3

DATEF, B-3

DB2, 1-1
access, 1-1
concepts, 2-16
description, 2-16
examples, 8-25
option, 1-6
relational databases, 2-16
SQL, 2-17
tables, 1-1, 2-17, 8-27, A-4
tables and views, 1-6

DDF (DEFINE DIRECTORY FUNCTION), 9-15, A-2

DDI (DEFINE DIRECTORY INQUIRY), 9-15, A-2

decimal places, 3-6
addition/subtraction, 7-4
multiplication/division, 7-4
numeric constants, 3-6
with AVERAGE, 11-30
with COUNT, 11-30
with TOTAL, 11-30

DEFER (DI), 4-14, 4-16, A-4, B-4, B-22, B-29

DEFINE, 9-9, 11-46
  description, 9-2

DEFINE DIRECTORY FUNCTION (DDF), 9-15, A-2, B-4, B-18, B-29

DEFINE DIRECTORY INQUIRY (DDI), 9-2, 9-15, A-2, B-4, B-17, B-26, B-29
  inquiries with SQL, 9-2
  inquiries without SQL, 9-2
  with a comment, 9-4

DELETE, 9-28
  description, 9-26

delete
  stored functions, 9-27
  stored inquiries, 9-27

DELETE DEFERRED INQUIRY, A-4, B-4, B-22, B-29

DELETE DIRECTORY FUNCTION, 9-27, 9-29, A-3, B-4, B-20, B-29

DELETE DIRECTORY INQUIRY, 9-27, 9-29, A-3, B-4, B-20, B-26, B-29

dependent segments, 2-5
  arithmetic calculation, 7-18

DFH4108 (CICS system error), 4-9

DFS064 error, 4-10

directory, 1-1, 10-2
  description, 1-5
  purposes, 9-1
  using stored inquiries, 10-1

DISPLAY, 1-4, 5-1, A-1, B-5, B-11, B-25, B-28
  with UDO, 11-2

DISPLAY DIRECTORY, 9-18, A-5, A-6, A-7, A-8, A-9, A-11, A-16
  description, 9-18

DISPLAY DIRECTORY DATA (PDD), A-5, A-11, B-5, B-24, B-27, B-30

DISPLAY DIRECTORY DATA DESCRIPT (PDDDS), A-5, A-18

DISPLAY DIRECTORY DATA DESCRIPTION (PDDDS), B-5, B-25, B-30

DISPLAY DIRECTORY DATA WHOLE (PDD WHOLE), A-5, A-14, B-5, B-30

DISPLAY DIRECTORY FUNCTION (PDF), 9-23, A-3, A-5, B-5, B-20, B-29

DISPLAY DIRECTORY INQUIRY (PDI), 9-23, A-5, B-5, B-18, B-19, B-29

DISPLAY DIRECTORY INQUIRY COMMENT (PDIC), A-2, A-5, B-29

DISPLAY DIRECTORY INQUIRY COMMENT WHOLE (PDICW), A-2, A-5, B-5, B-29

DISPLAY DIRECTORY INQUIRY COMMENT(PDIC), B-5

DISPLAY DIRECTORY INQUIRY WHOLE (PDIW), A-2, A-5, B-5, B-29

DISPLAY DIRECTORY LTERM (PDL), A-5, A-16, A-18, B-5, B-30
  iwith CICS, A-17
  with IMS (DL/I), A-16

DISPLAY DIRECTORY MAP (PDM), A-5, A-9, B-6, B-23, B-30
  with DB2 tables, A-9
  with IMS (DL/I), A-6
  with VSAM hierarchical files, A-8

DISPLAY DIRECTORY MAP (PDM))
  with VSAM non-hierarchical files, A-7

DISPLAY DIRECTORY MAP WHOLE (PDMW), A-5, A-20, B-6, B-23, B-30

DISPLAY DIRECTORY TERM (PDT), B-6, B-25, B-30

DISPLAY DIRECTORY VOCABULARY (PDV), A-5, A-10, B-6, B-24, B-30

DISPLAY FORMAT, B-6, B-12

display modes, 5-7
  horizontal, 5-7
  vertical, 5-8

documentation, 1-4
  IBM manual, 5-11, 5-12, 5-28
  Intraccess, 9-1, 9-4
  Intraccess help, 1-4
  VISION:Inquiry for IMS and TSO Technical Reference Guide, 10-18, A-16, A-18, A-21
  VISION:Inquiry Messages Guide, 4-10

VISION:Inquiry Reference Guide, 4-12, 8-23
VISION:Journey for Windows User's Guide, 1-4

DSC, A-1, B-28

E

EDIT, 11-2, 11-32, 11-50, A-3, B-6
syntax, 11-32

EDit/next, 10-21

EDITSQ, 1-4, 9-4, 10-9, A-3, B-6, B-20, B-26, B-29
description, 10-9
Text Editor screen, 10-10

end of Inquiry, 4-8

ENTER INQUIRY, 4-7

error handling, 4-10
corrections, 4-11

ESCAPE, A-1

ESDS (Entry Sequenced Data Set), 2-11

EXit or =X, 10-21

EXTRACT, 1-5, B-6, B-13, B-26, B-28

F

fields, 2-2, 2-16
character, 11-27
control fields, 5-3
definition, 2-2
display part of a field, 11-2
group field, 5-3
partial processing, 11-27
subfields, 5-4
user, 11-27

FIND, 1-5, 11-34, 11-37, 11-38, A-2, B-6, B-14, B-28
DB2 example, 8-25, 8-27
description, 8-1
example, 9-14
synonyms I or INTER, 8-29
VSAM example, 8-23
with DEFINE, 9-15

FIRST, A-1, A-4

FORMAT, 1-5, 11-47, A-3
COL or COLUMN, 11-2
EDIT, 11-2
LINE, 11-2
NOSP or NOSPACE, 11-2
PF, 11-2
REP or REPEAT, 11-2
SKIP, 11-2
SP or SPACE, 11-2

format output, 1-3

G

grand summary commands, 11-29
AVERAGE, 11-29
COUNT, 11-29
TOTAL, 11-29

grand totals, 7-16
SUM, 7-24

group field, 5-3

H

HElp, 10-21

hierarchical, 1-1, 2-12
databases, 8-21
files, 1-1, 1-5, 5-8
sibling segments, 8-21

hierarchy of operations, 5-19, 5-27, 7-11

horizontal report, 5-7

I

IF, A-1, B-7, B-14

II (sample trancode), 4-3

IMS (DL/I), 1-1, 5-1
test databases, 1-6

inquiries, 1-2
  definition, 1-4
  edit, 1-4
  stored, 1-4, 1-6
  terminating, 8-5
INQUIRY (field), 4-2
inquiry names, 10-3
  user-selected name, 10-3
Intraccess, 1-4, 9-1, 9-4, A-17, A-18
IQIO (CICS transaction code), 4-6, 4-9
IXX0109 error, 4-10
IXX0122 delete deferred inquiry, 4-16
IXX0140 deferred ID, 4-15
IXX0880 end of Show output, 4-16
IXX9121 end of inquiry, 4-5, 4-8
IXX9122 maximum # pages, 4-12
IXX9123 maximum # calls, 4-12

J

join, 2-19

K

KSDS (Key Sequenced Data Set), 2-11

L

LAST, A-1, A-4
laws of arithmetic, 7-18
LIMIT, 1-5, 5-40, 8-29, A-2, B-7, B-15, B-28
  description, 5-39
  example, 5-42, 5-43
  structured database, 5-41
  with FIND, 8-28
  with stored inquiries, 10-7, 10-8
limit output, 5-39, 11-22
LINE, 11-2, 11-14, 11-47, 11-48, A-3, B-7
logical expressions, 5-13, 5-21
  evaluating, 5-21
  with existence test, 5-17
  with FIND, 8-9
  with relational operators, 5-19
logical operators, 3-7, 5-19
logical product, 3-7
logical sum, 3-7
LTERMs, 5-44

M

match fields, 2-19, 8-1
matching fields
  with FIND, 8-1
messages, 4-10
  output, 4-8
  terminology, 4-10
MFS modname, 5-49
modes, 4-12
  continuous mode, 4-12
  conversational mode, 4-12

N

name combinations, A-5
name rules, 3-4
names
  data name, 4-10
  device names, 5-49
  field names, 4-10
  functions, 9-15
  inquiries, 9-15
  temporary field names, 7-3
negative values displayed, 7-8
noise words, 1-4, 3-2, A-4, B-27
non-hierarchical, 2-12

NOSPACE (NOSP), 11-2, 11-47, 11-49, B-7

notation, B-1

O

online processing, 4-11

operators, 3-2, 3-7
- arithmetic, 3-7
- logical, 3-7, 5-19
- relational, 3-7, 5-19

OUTPUT, 1-5, 5-48, A-2, B-7, B-16, B-28
- description, 5-44
- example, 5-45
- under CICS, 5-46
- under IMS, 5-44
- with stored inquiries, 10-8

P

PA keys, 10-13

PA1, 4-5, 10-21

PA2, 10-21

PA3, 10-21

PAGE, 4-2, 4-6, 11-1, 11-18, 11-49, B-2, B-3
- P/, 4-8
- P/L, 4-8
- P/N, 4-8

Page break, 11-51

parentheses, 5-21, 5-23, 5-24, 6-15, 7-4, 7-20, 9-15
- nested, 7-10
- with stored functions, 9-10
- with subtotals, 6-12, 6-15

PAss, 10-21

PC_BMS, A-18

PCDELETE, B-7, B-28

PCEXTRACT, B-7, B-16, B-28

PCLOAD, B-7, B-28

PCPAGE, A-17, A-18
- for Intraccess, A-17, A-18
- for VISION:Journey, A-17, A-18

PDD (DISPLAY DIRECTORY DATA), A-5, A-11

PDD WHOLE (DISPLAY DIRECTORY DATA WHOLE), A-5, A-14

PDDDS (DISPLAY DIRECTORY DATA DESCRIPT), A-5, A-18

PDF (DISPLAY DIRECTORY FUNCTION), 9-23, A-3, A-5

PDI (DISPLAY DIRECTORY INQUIRY), 9-23, A-5

PDIC (DISPLAY DIRECTORY INQUIRY COMMENT), A-2

PDICW (DISPLAY DIRECTORY INQUIRY COMMENT WHOLE), A-2

PDIW (DISPLAY DIRECTORY INQUIRY WHOLE), A-2, A-5

PDL (DISPLAY DIRECTORY LTERM), A-5, A-16, A-17

PDM (DISPLAY DIRECTORY MAP), A-5, A-7, A-8, A-9

PDMW (DISPLAY DIRECTORY MAP WHOLE), A-20

PDMW (DISPLAY DIRECTORY MAP WHOLE), A-5

PDV (DISPLAY DIRECTORY VOCABULARY), A-5, A-10

PF, A-3, B-8

PF (partial fielding), 11-2, 11-51
- character field, 11-27
- description, 11-27
- process part of a field, 11-27
- user field, 11-27
- with UDO, 11-27

PF keys, 4-1, 10-13

PF1, 10-21, A-3

PF10, 10-21

PF11, 10-21

PF12, 10-21

PF2, 10-21, A-4

PF3, 10-21, A-4

PF4, 10-21

PF9, 10-21

PLANT database, 1-6, 2-7
  description, 2-7
  examples, 5-1

positional test, 5-14
  FIRST, 5-14
  LAST, 5-15

R

records, 2-1, 2-16

REFRESH, 10-22

REFResh, 10-21

relational condition, B-14

relational expressions, 5-19, 5-27
  AND, 5-22
  OR, 5-22

relational operators, 3-7, 5-19, 5-27

REPEAT (REP), 11-2, 11-22, 11-50, A-3, B-8

REPEAT or REP
  description, 11-22

reserved words, B-27

routing output, 5-44

rows, 2-16

RRDS (Relative Record Data Set), 2-11

RUn, 10-21

S

Save, 10-21

SAVE/Pass, 10-21

SAVE/Run, 10-21

segments, 2-1
  dependent, 2-5
  multiple occurrence, 5-4, 8-23
  root, 2-5

SHOW, 4-16, A-3, B-8, B-23, B-29
  continuing the SHOW command output, 4-17
  end of show, 4-17

single quotation marks, 10-3
  use around character constants, 10-3

SKILL database, 1-6, 2-7
  description, 2-10
  Examples, 5-1

SKIP, 11-2, 11-14, 11-47, 11-48, A-3, B-8

SORT, 1-5, 4-12, 5-28, A-1, B-8, B-16, B-28
  ascending, 5-28
  default, 5-29
  descending, 5-28, 5-35
  example, 8-12
  multiple fields, 5-32
  using blanks, 5-34
  with LIMIT, 5-40
  with subtotals, 6-18
  with UDO, 5-38

SPACE (SP), 11-2, 11-47, 11-49, A-3, B-8

SQL, 2-17
  Description, 2-17
  stored functions not supported with, 10-25
  VISION:Inquiry commands use with SQL, A-4
  VISION:Inquiry noise words not supported, A-4

SQL commands, B-25
  SELECT, B-25

SQL Descriptor Area (SQLDA), 5-12

SQLDA (SQL Descriptor Area), 5-12

starting, B-21
  in CICS, B-21
  in IMS, B-21

store inquiries
  replacing, 9-7

stored functions, 9-1, 9-8, 9-15, 10-25, 10-30, 11-44, B-2
  not supported with the native SQL syntax facility, 10-25

stored inquiries, 1-6, 9-1, 9-4, 9-5, 10-1, 10-7, 10-8, 11-40, B-2, B-21

calling stored function, 9-8
changing, 9-7
Example, 1-4
LIMIT ignored during processing, 9-6
OUTPUT ignored during processing, 9-6
running, B-21
summary with SQL, 10-8
summary without SQL, 10-8
using stored functions, 10-28, 10-29
with substitutable values, 9-5

storing, 9-9
a function, 9-9
a function with no column heading, 9-10

Structured Query Language (SQL), 2-17

subfield, 5-38

submitting a second inquiry, 4-5

submitting inquiries, 4-3

substitutable values, 9-5, 9-6, 9-12, 9-15, 10-3, 10-5, 10-8, 10-29, 11-42

subtotals, 6-12, 6-18, 11-30
description, 6-12
example, 6-13
syntax, 6-18
using group field or control break, 6-12
with AVERAGE, 6-15
with SORT, 6-18

SUM, 1-5, 5-38, A-2, A-4, B-8, B-17, B-28
arithmetic calculation, 7-3
description, 7-12
different segments, 7-18
parentheses, 7-20

suppression, 1-1
blank fields, 5-8
repeated fields, 1-1, 5-33

syntax errors, 5-44, 5-46
example, 5-49

system names, B-3

system vocabulary, 1-6

T

T/C Terminate current message, 5-47

tables, 2-16
columns, 2-17
rows, 2-17

temporary fields, 7-3, 7-23
% 1st character in name, 3-4
%names, 7-3
description, 7-1
example, 7-2
numeric, 11-36
object variable field, 4-10
with FIND, 8-8
with stored functions, 9-15

terminating inquiries, 3-2, 3-4

terminology
data name, 4-10
object, 4-10
object variable field, 4-10
verb, 4-10

TERMs, 5-46

terms, B-2

test databases, 1-6, 2-7
IMS (DL/I), 1-6
VSAM, 1-6

Text Editor, 9-7, 10-9
PA1, PA2, or PA3 (REFRESH), 10-18
PF1 (HELP), 10-13
PF10 (PASS), 10-17
PF11 (SAVE/PASS), 10-17
PF12 (EDNEXT), 10-17
PF2 (RUN), 10-13
PF3 or CLEAR (EXIT), 10-13
PF4 (SAVE/RUN), 10-13
PF5 (RFIND), 10-15
PF6 (RCHANGE), 10-15
PF7 (UP), 10-15
PF8 (DOWN), 10-15
PF9 (SAVE), 10-16

Text Editor line commands, 10-22
  A,B, 10-22
  C, 10-22
  C#, 10-22
  CC, 10-22
  D, 10-22
  D#, 10-22
  DD, 10-22
  I, 10-22
  I#, 10-22
  M, 10-22
  M#, 10-22
  MM, 10-22
  R#, 10-22
  RR, 10-22

Text Editor primary commands
  change xx yy ALL, 10-20
  find xx, 10-20
  PF or PB, 10-20
  PL or Bottom, 10-20
  U# or D#, 10-20

Text Editor screen, 10-10
  layout, 10-11
  PF/PA keys, 10-13
  primary commands, 10-20

TIME, 11-1, 11-18, 11-49, B-2, B-3

title, 11-6
  example, 11-7, 11-8

TOTAL, 1-5, 6-1, 6-12, A-2, B-4, B-10, B-28
  description, 6-1
  example, 6-2
  with DISPLAY, 6-4
  with UDO, 11-29

TRANCODE, 4-2

TRANSACTION, 4-7

TSO, 4-18
  inquiry processing, 4-18
  terminating inquiries, 4-18

U

UDO (User Defined Output), B-2

use with stored inquities, 9-4

User Defined Output (UDO), 1-1
  Example, 1-3
  SHOW, 4-16

user defined output (UDO), 11-1
  /FORMAT INQUDO, 11-51
  arithmetic processing, 11-35
  commands, 11-2
  conditional selection, 11-20
  DATE, 11-18
  description, 11-1
  EDIT command, 11-32
  FIND command, 11-39
  grand summary commands, 11-29
  horizontal movement commands, 11-9
  LINE command, 11-15
  PAGE, 11-18
  REPEAT command, 11-23
  SKIP command, 11-14
  SP command, 11-9
  stored functions, 11-44
  stored inquiries, 11-40
  substitutable values, 11-42, 11-46
  subtotals, 11-30
  syntax, 11-3
  TIME, 11-18
  vertical movement commands, 11-14

user profiles, 1-6

user-selected name, 10-3

using different substitutable vales, 10-5

V

verbs, B-28

vertical report, 5-8

VISION:Inquiry

commands, A-1
formatted output, 1-3
syntax, 3-1
typical inquiry, 1-2
under CICS, 4-1, 4-6
under IMS, 4-1, 4-2

VISION:Journey, A-17, A-18, B-7, B-28
PCDELETE, B-28
PCEXTRACT, B-28
PCLOAD, B-28

VSAM files, 1-1, 2-11, 8-23
description, 2-11, 2-12
ESDS, 2-11
example, 8-23, 8-24, 8-25
hierarchical files, 1-1
KSDS, 2-11
multiple-access, 8-24
not in IMS online (MPP) environment, 2-11
RSDS, 2-11
test files, 1-6

VSAM hierarchical files, 2-12
DISPLAY DIRECTORY MAP (PDM), A-8

VSAM non-hierarchical files, 2-12, A-4, A-7
DISPLAY DIRECTORY MAP (PDM), A-7

VSHPLANT, 1-6, 2-12
description, 2-15

VSHSKILL, 1-6, 2-12, 2-16
description, 2-16

VSPLANT, 1-6, 2-12, 5-1
description, 2-12

VSSKILL, 1-6, 2-12, 5-1
description, 2-12

W

wraparound, 3-3