HP NonStop SQL/MP Report Writer Guide

Abstract

This manual explains how to use the HP NonStop™ SQL/MP report writer commands, clauses, and functions, and the SQLCI options that relate to reports.The manual tells how to produce formatted reports using data from a NonStop SQL/MP database.

Product Version

NonStop SQL/MP D30

Supported Release Version Updates (RVUs)

This publication supports D40.00 and all subsequent D-series RVUs, and G01.00 and all subsequent G-series RVUs, until otherwise indicated in a new edition.

Part Number	Published
527213-001	September 2003

Document History

Part Number	Product Version	Published
107934	NonStop SQL/MP D30	December 1994
527213-001	NonStop SQL/MP D30	September 2003

HP NonStop SQL/MP Report Writer Guide

Index   Figures   Tables

Figures

Figures (continued)

Figures (continued)

Tables

What's New in This Manual

Manual Information

Abstract

This manual explains how to use the HP NonStop™ SQL/MP report writer commands, clauses, and functions, and the SQLCI options that relate to reports.The manual tells how to produce formatted reports using data from a NonStop SQL/MP database.

Product Version

NonStop SQL/MP D30

Supported Release Version Updates (RVUs)

This publication supports D40.00 and all subsequent D-series RVUs, and G01.00 and all subsequent G-series RVUs, until otherwise indicated in a new edition.

Part Number	Published
527213-001	September 2003

Document History

Part Number	Product Version	Published
107934	NonStop SQL/MP D30	December 1994
527213-001	NonStop SQL/MP D30	September 2003

New and Changed Information

- This publication has been updated to reflect new page formats and product names.
- Product names in graphic representations are consistent with the current product interface.
- The technical content of this publication has not been updated and reflects the state of the product at the G06.21 release version update (RVU).

About This Manual

This manual provides task-oriented instructions on how to design and produce reports using the report writer and information from a NonStop SQL/MP database. The manual also provides an overview of the report writer commands and SQLCI commands that are directly related to formatting reports.

Audience

Readers should have some experience in using command interpreters and be familiar with the HP NonStop Kernel operating system. If you are not a programmer, you might need to request a programmer's assistance in creating reports that require the use of the more advanced features.

As a reader of this manual, you should have read the *Introduction to NonStop SQL* so that you are familiar with the terminology. You can also read the *NonStop SQL Quick Start* for a quick introduction to expressing ad hoc queries and producing simple reports.

How to Use This Manual

This manual is organized to make instructions for specific report writing tasks easy to find:

Section 1, Introduction to the NonStop SQL/MP Report Writer — introduces the report writer, describes its components, and compares the default report format with a custom formatted report

Section 2, Using SQLCI and the Report Writer — gives instructions for running SQLCI and using report writer commands. This section describes various ways to define and produce reports

Section 3, Selecting Data for a Report — shows you how to develop queries and specify calculations to process the source data for a report

Section 4, Customizing a Report — gives instructions for defining the report layout and style. You can give detailed specifications for output lines, line items, column headings, titles, and footings

Section A, Comparison of the Report Writer and the Enform Language — lists the SQLCI and report writer features that correspond to features of the Enform language

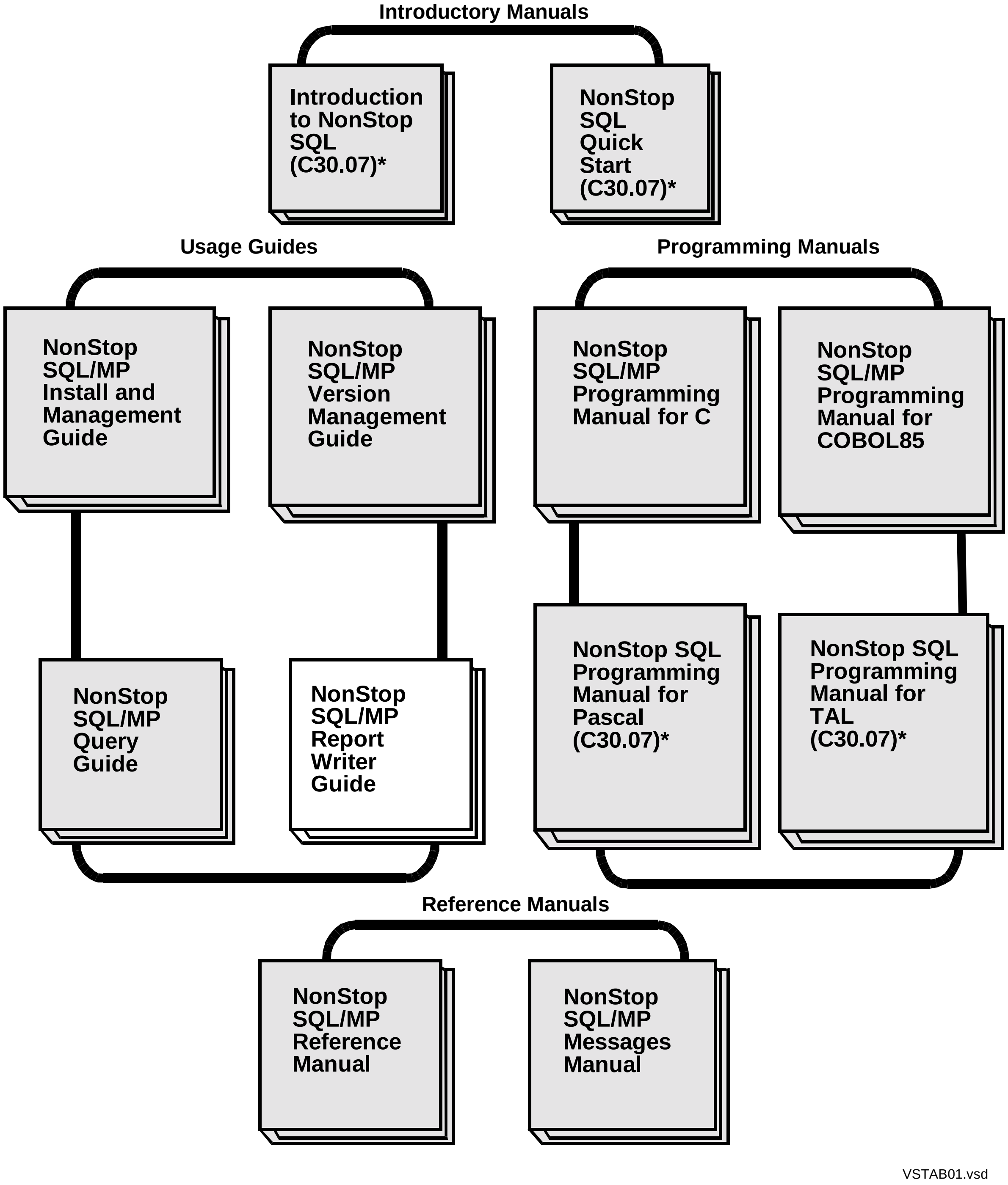
The version of SQL implemented by HP and referred to in this manual conforms with the ANSI SQL standard; minor exceptions are noted in the *SQL/MP Reference Manual*, which fully documents the HP implementation of SQL.

Related Manuals

This manual is part of the NonStop SQL library of manuals (as shown in Figure i). The library includes the following manuals:

- *Introduction to NonStop SQL* provides an overview of the SQL/MP relational database management system.
- *SQL Quick Start* describes how to run SQLCI, how to execute simple queries on a database, how to modify data, and how to produce a formatted report.
- *SQL/MP Installation and Management Manual* explains how to plan, install, create, and manage a NonStop SQL database; describes the syntax of installation and management commands; and describes NonStop SQL catalogs and file structures.
- *SQL/MP Query Guide* describes how to retrieve and modify data in a SQL/MP database and how to analyze and improve query performance.
- *SQL/MP Version Management Guide* describes the rules governing version management for the NonStop SQL software, catalogs, objects, messages, programs, and data structures.
- *SQL/MP Programming Manual* for C and COBOL85 and the *SQL Programming Manual* for Pascal and TAL describe the programmatic interface for the particular language.
- *SQL/MP Reference Manual* describes the SQL language elements, expressions, functions, and statements. (See this manual for the complete description and syntax of the report writer commands and the SELECT statement.)
- *SQL/MP Messages Manual* describes NonStop SQL/MP messages for the conversational interface, the application programming interface, and utilities.

Figure i. **NonStop SQL/MP Library Map**


Introductory Manuals
Introduction to NonStop SQL (C30.07)*
NonStop SQL Quick Start (C30.07)*
Usage Guides
Programming Manuals
NonStop SQL/MP Install and Management Guide
NonStop SQL/MP Version Management Guide
NonStop SQL/MP Programming Manual for C
NonStop SQL/MP Programming Manual for COBOL85
NonStop SQL/MP Query Guide
NonStop SQL/MP Report Writer Guide
NonStop SQL Programming Manual for Pascal (C30.07)*
NonStop SQL Programming Manual for TAL (C30.07)*
Reference Manuals
NonStop SQL/MP Reference Manual
NonStop SQL/MP Messages Manual
VSTAB01.vsd

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

`file-name`

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *`Italic computer type`* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

`pathname`

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num   ]
   [ -num  ]
   [ text  ]

K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]…
```

```
[ - ] {0|1|2|3|4|5|6|7|8|9}…
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char…"
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;
```

```
LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]…
```

!i and !o. In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                   !i
                         , error        ) ;             !o
```

!i,o. In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                     !i,o
```

!i:i. In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length          !i:i
                            , filename2:length ) ;      !i:i
```

!o:i. In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                       !i
                        , [ filename:maxlen ] ) ;       !o:i
```

1

Introduction to the NonStop SQL/MP Report Writer

NonStop SQL/MP is a relational database management system that uses the industry standard SQL language to define and manipulate data. The NonStop SQL/MP conversational interface (SQLCI) includes a report writer that enables you to format data retrieved with a SELECT command and to display or print the formatted data in a report. The report writer consists of formatting commands and a set of options that you can set to control the layout and style of reports.

When you are preparing a report, you can also use general SQLCI commands to control your session environment.

Note. The SELECT command referred to in this manual is a SELECT statement you enter by using SQLCI.

Data Organization

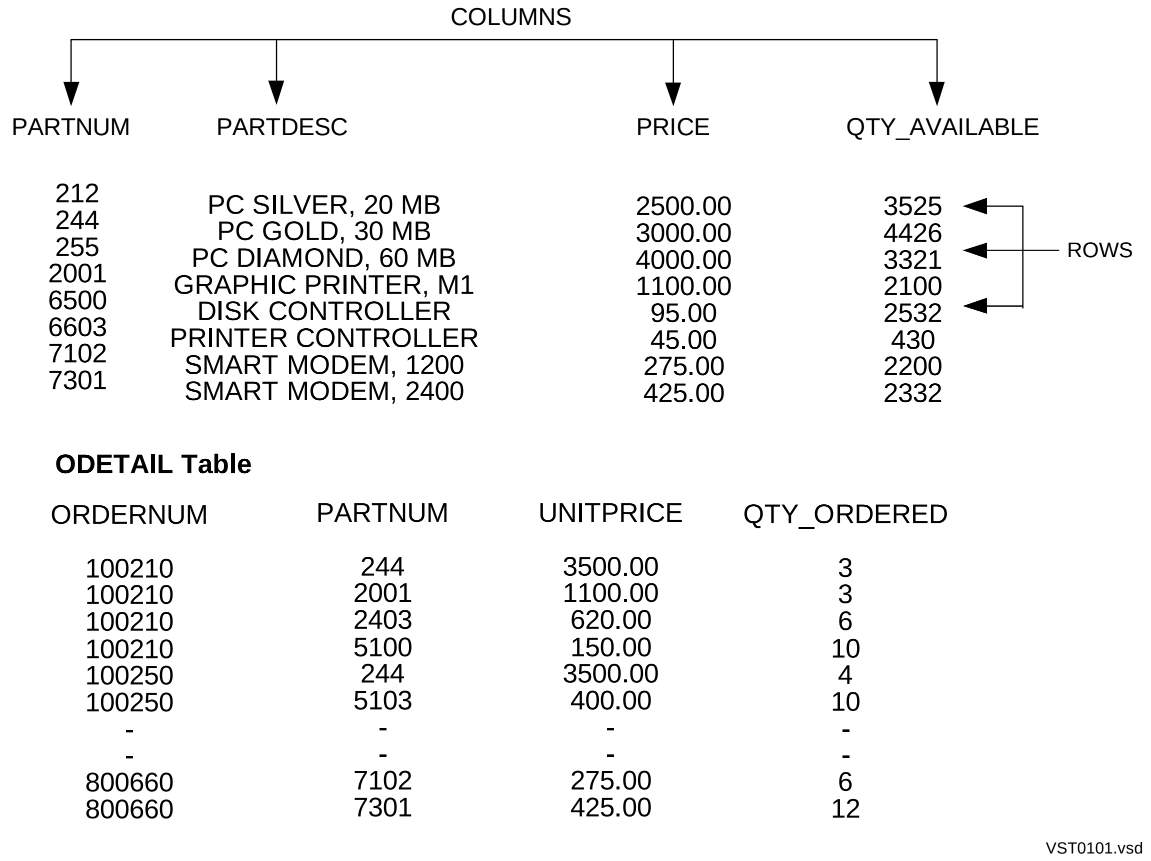
In a NonStop SQL/MP database, data is organized and maintained in tables. A table consists of columns and rows:

- Each row contains pieces of related data, such as an employee's identification number, name, and address.
- Each column contains data of the same type, such as all employee numbers.

SQLCI displays rows horizontally and columns vertically. Columns have names; rows do not.

Figure 1-1 contains sample rows from the PARTS and ODETAIL tables of an inventory database. The PARTS table describes each part in the inventory. The ODETAIL table contains order details (line entries).

Figure 1-1. Sample Content of Tables


COLUMNS
ROWS
VST0101.vsd

PARTNUM	PARTDESC	PRICE	QTY_AVAILABLE
212	PC SILVER, 20 MB	2500.00	3525
244	PC GOLD, 30 MB	3000.00	4426
255	PC DIAMOND, 60 MB	4000.00	3321
2001	GRAPHIC PRINTER, M1	1100.00	2100
6500	DISK CONTROLLER	95.00	2532
6603	PRINTER CONTROLLER	45.00	430
7102	SMART MODEM, 1200	275.00	2200
7301	SMART MODEM, 2400	425.00	2332

ODETAIL Table

ORDERNUM	PARTNUM	UNITPRICE	QTY_ORDERED
100210	244	3500.00	3
100210	2001	1100.00	3
100210	2403	620.00	6
100210	5100	150.00	10
100250	244	3500.00	4
100250	5103	400.00	10
-	-	-	-
-	-	-	-
800660	7102	275.00	6
800660	7301	425.00	12

SQL Database Reports

SQLCI generates a default report definition for each SELECT command you enter during your SQLCI session. You can override characteristics of the default report by using the layout options, style options, and report formatting commands.

Layout and style options are not related to a specific SELECT command. These options are global; they control general properties of reports. Layout options, such as page length and margin settings, control the appearance of a report on the screen or printed page. Style options, such as the decimal character, underline character, or format of dates, control the appearance of items in a report. Your current layout and style options affect the output of any SELECT command you enter during your SQLCI session.

Report formatting commands specify a report template, including such elements as detail lines, titles, and footings. These commands are related to a specific SELECT command through database column references.

The report writer uses the information in the current report definition to format your report. You can display or print data retrieved by the SELECT command in the default report format that exists for each SELECT command by not using any of the style options, layout options, or report formatting commands. However, if you want to design your own report, you can revise the default format by entering report formatting commands or by using the SET LAYOUT or SET STYLE commands to change the default values of layout or style options.

Using the report writer features, the SELECT command, and the LIST command, you can develop a report in stages and view each change as you make it.

When you are satisfied with the format and content of your report, you can either enter the LIST ALL command to produce the report, or enter the SAVE command to save the final version of the options, formatting commands, and SELECT command in a file. To print the report that you have defined and saved, use the OBEY command as described in Defining Reports in Command Files on page 2-15.

Report Development Steps

The steps for using the SQLCI report writer to develop a report are described in detail in the remainder of the manual. The following summary provides an overview of the steps:

1. Selecting data
   To produce a report, you must execute a SELECT command. In the SELECT command, you specify the data to be retrieved, the tables that contain the data, and the criteria for selecting rows and joining rows from different tables and views. See the *SQL/MP Query Guide* and the *SQL/MP Reference Manual* for descriptions of the SELECT command and its components. Before you specify the SELECT command, you should plan the content and arrangement of information in the report. To determine the SELECT requirements, you need to know only the basic items of data that appear in each line. You can design the complete report with titles, headings, and so forth later. You should include any report elements that require data from the database so you know what to specify in the SELECT command.

   Note. Depending upon the criteria you include in a SELECT command, your query might affect system performance or require a long time to complete. Before you issue a SELECT command, you should consider preparing the SELECT command and generating an explanation of the resources required to retrieve the data you need. See the *SQL/MP Reference Manual* for information about the PREPARE command and EXPLAIN utility.

2. Formatting the report
   The simplest way to produce a report is to display or print the output of the SELECT command using the report writer default format. To produce more elaborate reports, use report formatting commands to specify the content of detail lines, titles, footings, and column headings. Report formatting commands can also calculate subtotals of values in columns at specified break points. A break point occurs when the value in a column changes; for example, when the department number changes in a report with rows ordered by department number. In addition to subtotals at break points, you can calculate the total value of a column over all rows in the report. To change the appearance of the report, use the layout options.

You can specify layout options, such as margins, the number of lines per page, line spacing, and whether default headings are to print the above columns of data.

3. Customizing the style
   Use the style options to customize subtotal labels, the style of date and time formats, and special characters representing the underline and the decimal point.

4. Displaying and printing the report
   To display or print all or part of the report, use the LIST command. Reports are written to the OUT_REPORT file. If you want to see the report on your terminal, you can use the default OUT_REPORT file (typically, your terminal). To print a report, specify the name of a printer or a spooler as the OUT_REPORT file.

5. Saving the report definition
   If you develop your report interactively, you can use the SAVE command to save in an EDIT file the commands you used to define the report. Then each time you want to print the report, use an OBEY command to execute the saved commands. You can use a text editor to create the command file directly or to modify a saved file.

Report Layout and Style

You specify the layout of a report by using the SET LAYOUT command and report formatting commands. You specify the style of a report by using the SET STYLE command.

Each layout and style option has a default value. The options apply to any report you print while the option is set. A layout or style option stays in effect until you set it to some other value, reset it to its default value, or end your SQLCI session.

For example, the default value for the PAGE_COUNT layout option is ALL; this option prints the entire report. You can set a page count limit for the reports you are producing. If you set a limit and do not want that limit applied to the next report you print, you must reset the page count by entering either of the following commands at the SQLCI prompt (>>):

```
>> RESET LAYOUT PAGE_COUNT;
>> SET LAYOUT PAGE_COUNT ALL;
```

To reset a style option, use the RESET STYLE command. (The prompts >>, +>, and S> are described under Starting and Ending an SQLCI Session on page 2-1.

Report formatting commands relate to a specific SELECT command by referencing database columns. For example, you can specify the format of a line that prints the total of all values in a column, such as QTY_ORDERED. After you list all the retrieved rows or cancel the SELECT command, the report formatting commands are canceled.

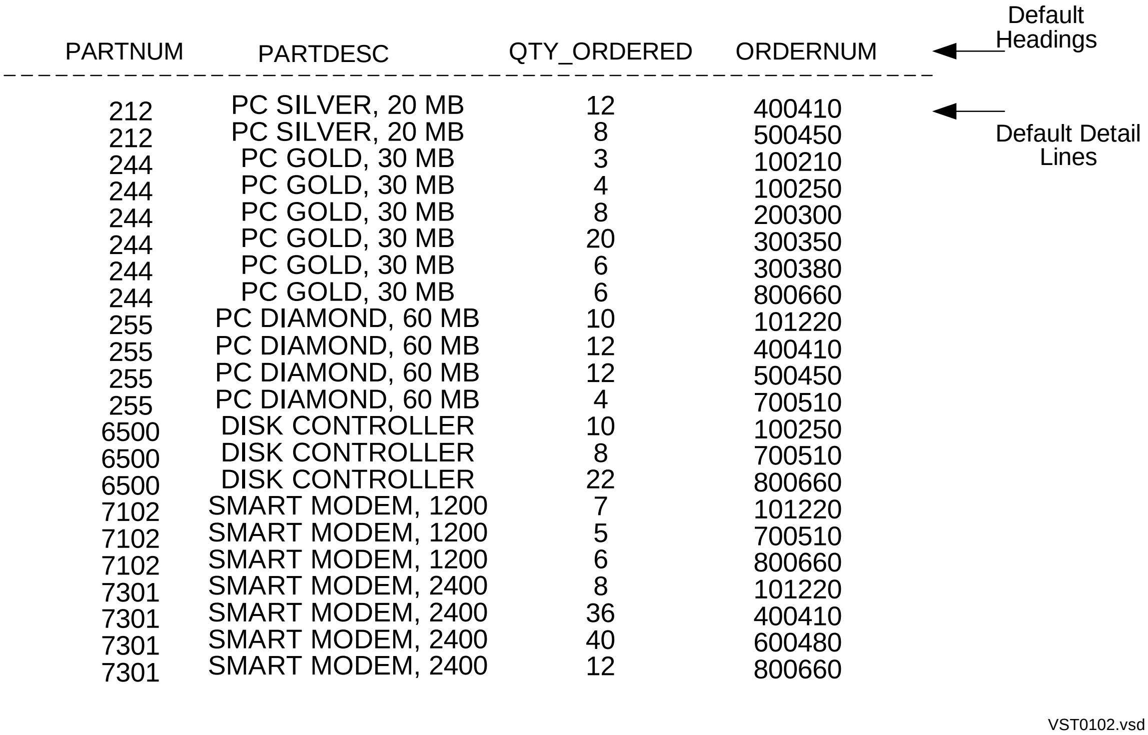
If you have saved the SELECT command and the report formatting commands in a file, you can reexecute them. You can also use the HISTORY, FC, or exclamation point (!) command to reexecute report commands executed earlier in the current session.

Default Report

You can display or print a report without specifying any report formatting commands, layout options, or style options. The report is produced in the default format and is called the default report. Figure 1-2 shows an example of a default report produced with these commands:

```
>> VOLUME SALES;
>> SET LIST_COUNT 0;
>> SELECT P.PARTNUM, PARTDESC, QTY_ORDERED, ORDERNUM
+> FROM PARTS P, ODETAIL O
+> WHERE P.PARTNUM = O.PARTNUM
+> AND ( P.PARTNUM < 300
+> OR P.PARTNUM >= 6500 )
+> ORDER BY P.PARTNUM, ORDERNUM ;
S> LIST ALL;
```

Figure 1-2. Example of Default Report Layout



```
  PARTNUM      PARTDESC            QTY_ORDERED   ORDERNUM
-------------------------------------------------------------
    212     PC SILVER, 20 MB           12         400410
    212     PC SILVER, 20 MB            8         500450
    244      PC GOLD, 30 MB             3         100210
    244      PC GOLD, 30 MB             4         100250
    244      PC GOLD, 30 MB             8         200300
    244      PC GOLD, 30 MB            20         300350
    244      PC GOLD, 30 MB             6         300380
    244      PC GOLD, 30 MB             6         800660
    255     PC DIAMOND, 60 MB          10         101220
    255     PC DIAMOND, 60 MB          12         400410
    255     PC DIAMOND, 60 MB          12         500450
    255     PC DIAMOND, 60 MB           4         700510
   6500     DISK CONTROLLER            10         100250
   6500     DISK CONTROLLER             8         700510
   6500     DISK CONTROLLER            22         800660
   7102    SMART MODEM, 1200            7         101220
   7102    SMART MODEM, 1200            5         700510
   7102    SMART MODEM, 1200            6         800660
   7301    SMART MODEM, 2400            8         101220
   7301    SMART MODEM, 2400           36         400410
   7301    SMART MODEM, 2400           40         600480
   7301    SMART MODEM, 2400           12         800660
```

The report contains a detail line for each row retrieved by the SELECT command. A row must meet the criteria specified in the WHERE clause; the part number must be less than 300 or greater than or equal to 6500.

The ORDER BY clause specifies that rows are arranged in ascending order by part number and, for each part number, in ascending order by order number.

The WHERE clause also specifies that rows of the PARTS table are joined with rows of the ODETAIL table when the rows have the same part number.

These are the components of the default report:

- Detail lines
    - Each detail line contains all elements of the select list in the order you specify them in the SELECT command. (In Figure 1-2, the select list is the columns shown on the same line as SELECT.)
    - Values are displayed in the default display formats. The report writer determines the default display format by the data type defined for a column in the table definition.
    - The width of a report column is equal to the heading or item width, depending on which is wider. The report columns are separated by two spaces. Numeric items are right justified and character items are left justified.
    - Lines are single spaced. If the print line does not fit within the left and right margins, the line is folded. The place at which the line is broken depends on the setting of the LOGICAL _FOLDING option.
    - If the print line does not fit on one line because it exceeds the width of the output device, it is either wrapped to the next line or truncated, depending on the setting of the WRAP option.
- Headings
    - Each element that is a table or view column has a column heading consisting of the heading or column name as defined in the catalog. An element that is an expression or numeric parameter has the default heading (EXPR). The headings are underlined and separated from the detail lines by a blank line.

The layout and style options affect the default report. For example, by default the left margin is 0; the output line begins at the left edge of your screen or printed page. The right margin is set to the width of the output device. The page length is the current setting of the PAGE_LENGTH layout option.

By default, the top and bottom margins of a report are each one blank line.

For more information on summaries of the layout and style option, which include the default setting of each option, see Report Writer Components on page 1-10. For detailed information about the layout and style options, see the *SQL/MP Reference Manual*.

Customized Report

By entering report formatting commands, you can define these additional report elements:

- Page title: A page title appears at the top of each page.
- Report title: The main title of the report appears at the top of the first page (below the page title, if there is one).

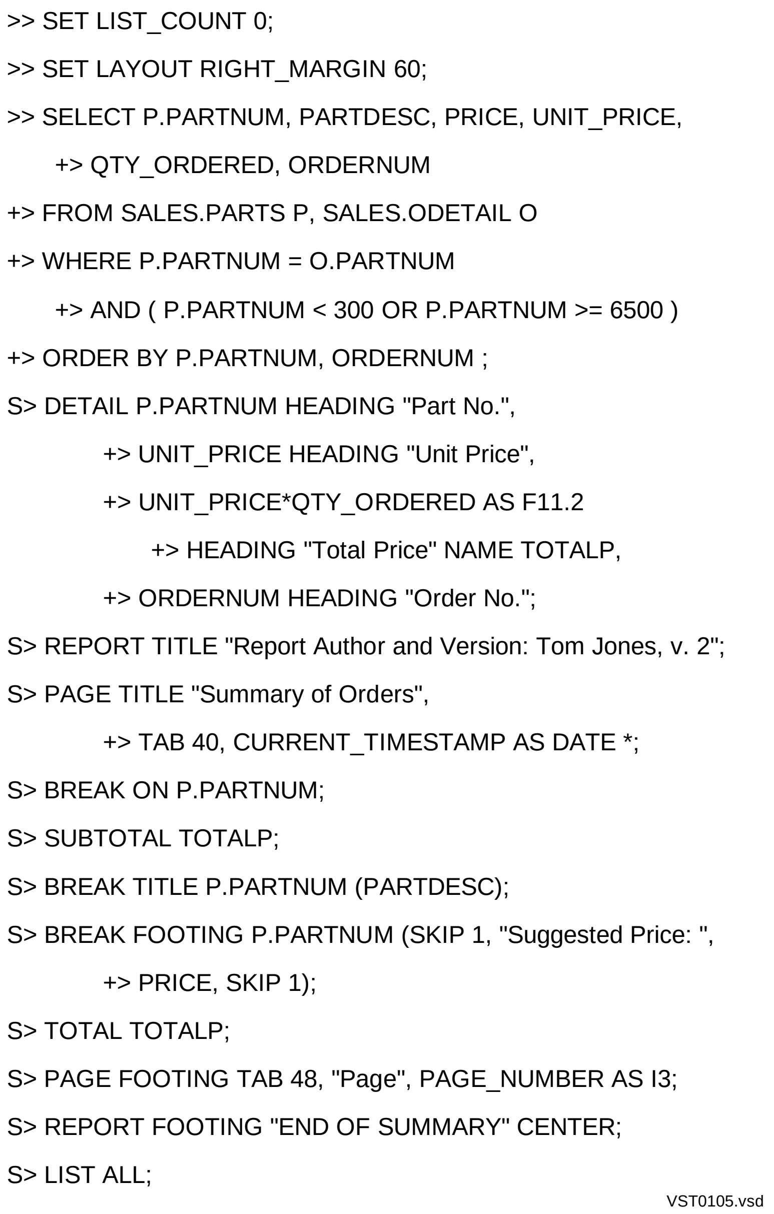
- Detail lines: A detail line is printed for each row of the result table described by the select list in the SELECT command. You can do the following:
  - Include items from the select list, expressions, or literals
  - Connect two or more print items, omitting intervening blanks
  - Specify conditions for printing items
  - Define headings and names for items
  - Use positional clauses like SKIP and PAGE to control the position of the next detail item
- Break groups: A break group consists of all detail lines with the same value in a specific column (the break column).
- Break title: A break title precedes the first line of a break group.
- Break footing: A break footing follows the last line of a break group.
- Subtotals for numeric columns can be calculated when the value of a specific break column changes or each time any break column value changes. A subtotal appears after the last line of the break group you specify.
- Totals: Any numeric column in the detail line can have a total value calculated and printed following the last detail line in the report.
- Report footing: A report footing appears at the end of the report (above the page footing, if there is one).
- Page footing: A page footing appears at the bottom of each page.

For more information on the list of the report formatting commands and brief descriptions of the purpose of each one, see Report Writer Components on page 1-10.

A Sample Formatted Report

Figure 1-3 shows a formatted report produced by using the following commands:

Figure 1-3. A Sample Formatted Report



```
>> SET LIST_COUNT 0;
>> SET LAYOUT RIGHT_MARGIN 60;
>> SELECT P.PARTNUM, PARTDESC, PRICE, UNIT_PRICE,
     +> QTY_ORDERED, ORDERNUM
+> FROM SALES.PARTS P, SALES.ODETAIL O
+> WHERE P.PARTNUM = O.PARTNUM
     +> AND ( P.PARTNUM < 300 OR P.PARTNUM >= 6500 )
+> ORDER BY P.PARTNUM, ORDERNUM ;
S> DETAIL P.PARTNUM HEADING "Part No.",
          +> UNIT_PRICE HEADING "Unit Price",
          +> UNIT_PRICE*QTY_ORDERED AS F11.2
               +> HEADING "Total Price" NAME TOTALP,
          +> ORDERNUM HEADING "Order No.";
S> REPORT TITLE "Report Author and Version: Tom Jones, v. 2";
S> PAGE TITLE "Summary of Orders",
          +> TAB 40, CURRENT_TIMESTAMP AS DATE *;
S> BREAK ON P.PARTNUM;
S> SUBTOTAL TOTALP;
S> BREAK TITLE P.PARTNUM (PARTDESC);
S> BREAK FOOTING P.PARTNUM (SKIP 1, "Suggested Price: ",
          +> PRICE, SKIP 1);
S> TOTAL TOTALP;
S> PAGE FOOTING TAB 48, "Page", PAGE_NUMBER AS I3;
S> REPORT FOOTING "END OF SUMMARY" CENTER;
S> LIST ALL;
```

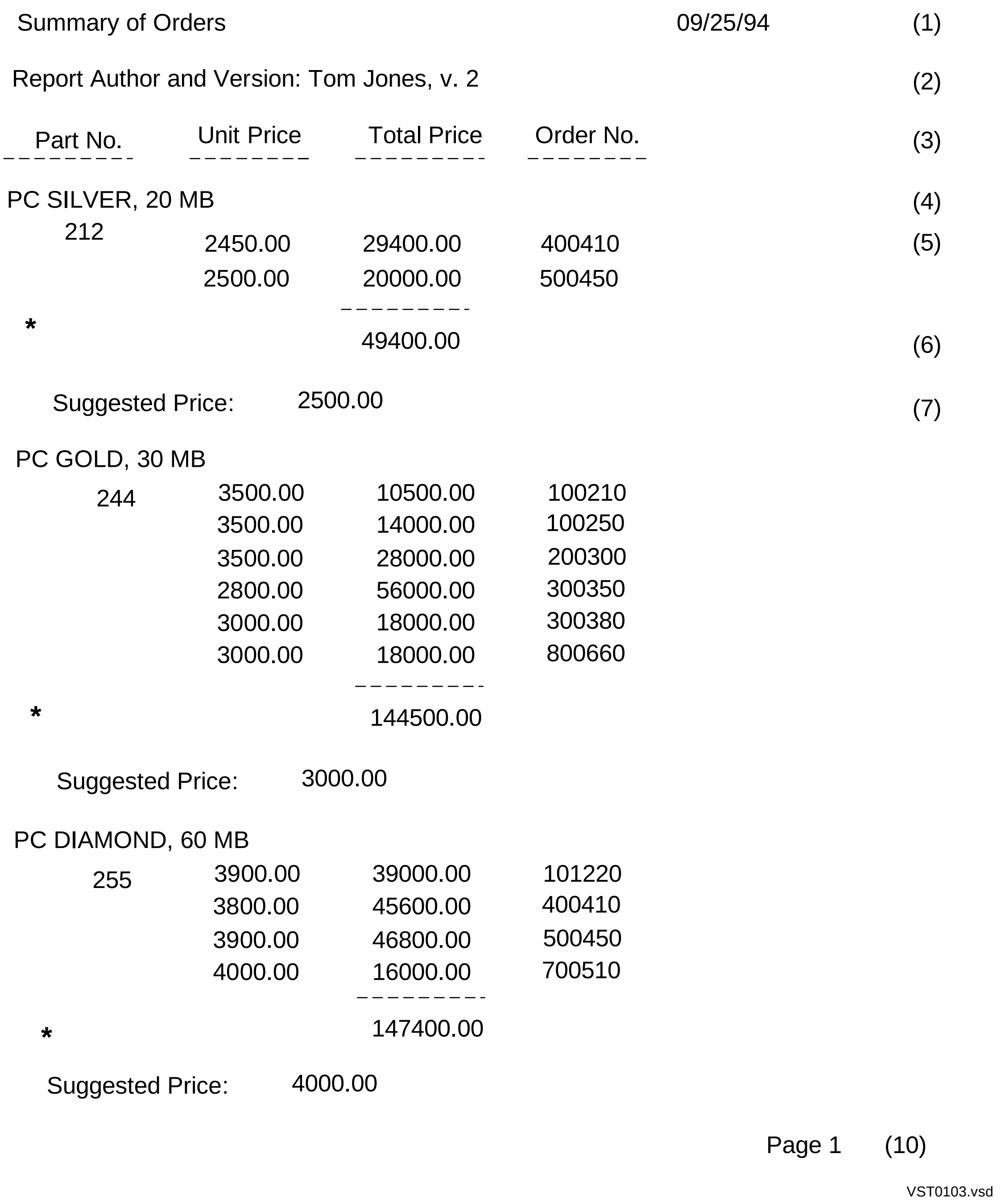
VST0105.vsd

The report contains the following components:

1. Page title
2. Report title
3. Headings defined in the DETAIL command
4. Break title
5. Detail line
6. Subtotal line with a subtotal label, which is an asterisk (*)

7. Break footing
8. Total lines
9. Report footing
10. Page footing
11. Report Layout and Style

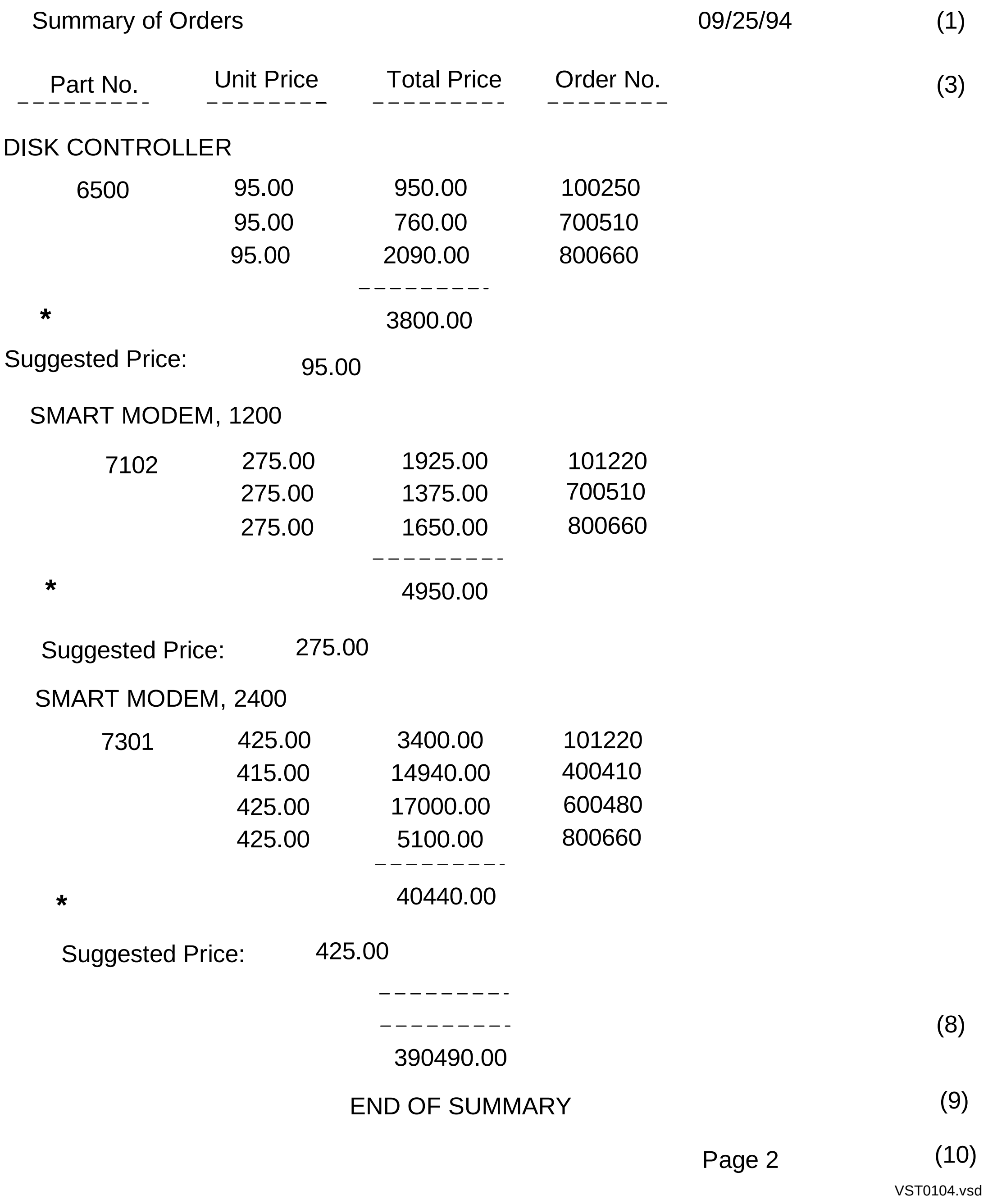
Figure 1-4. Example of a Formatted Report



```
 Summary of Orders                                        09/25/94        (1)

Report Author and Version: Tom Jones, v. 2                                (2)

  Part No.      Unit Price     Total Price    Order No.                   (3)
----------    ---------      ----------    ---------

PC SILVER, 20 MB                                                          (4)
     212
               2450.00        29400.00       400410                       (5)
               2500.00        20000.00       500450
                             ----------
 *
                              49400.00                                    (6)

   Suggested Price:     2500.00                                           (7)

 PC GOLD, 30 MB
        244      3500.00        10500.00       100210
                 3500.00        14000.00       100250
                 3500.00        28000.00       200300
                 2800.00        56000.00       300350
                 3000.00        18000.00       300380
                 3000.00        18000.00       800660
                               ----------
 *
                                144500.00

   Suggested Price:     3000.00

 PC DIAMOND, 60 MB
        255      3900.00        39000.00       101220
                 3800.00        45600.00       400410
                 3900.00        46800.00       500450
                 4000.00        16000.00       700510
                               ----------
  *                             147400.00

   Suggested Price:     4000.00

                                                          Page 1    (10)
```

VST0103.vsd

Figure 1-5. Example of a Formatted Report



```
        Summary of Orders                                       09/25/94          (1)

     Part No.      Unit Price     Total Price    Order No.                        (3)
   ---------       --------       ---------      --------

  DISK CONTROLLER
          6500        95.00          950.00        100250
                      95.00          760.00        700510
                      95.00         2090.00        800660
                                   ---------
    *                               3800.00
  Suggested Price:            95.00

     SMART MODEM, 1200
             7102     275.00        1925.00        101220
                      275.00        1375.00        700510
                      275.00        1650.00        800660
                                   ---------
    *                               4950.00

      Suggested Price:        275.00
      SMART MODEM, 2400
             7301     425.00        3400.00        101220
                      415.00       14940.00        400410
                      425.00       17000.00        600480
                      425.00        5100.00        800660
                                   ---------
     *                             40440.00

        Suggested Price:       425.00
                                   ---------
                                   ---------                                      (8)
                                  390490.00
                              END OF SUMMARY                                      (9)

                                                        Page 2                   (10)
                                                                       VST0104.vsd
```

Report Writer Components

In addition to SQLCI commands that are useful when generating reports, the report writer has the following components:

- Print items and logical lines
- Report formatting commands

- Clauses
- Style options
- Layout options
- Report functions

This subsection provides a summary of the report writer components. For a detailed description of these components, refer to the *SQL/MP Reference Manual.*

Print Items and Logical Lines

Print items identify the items to print in the titles, footings, and detail lines of your report and are optionally accompanied by formatting instructions. A list of print items defines a logical line.

A print item is generally a column identifier, a literal, an arithmetic expression, or any of these clauses: CONCAT, IF/THEN/ELSE, NEED, PAGE, SPACE, SKIP, and TAB. (In some clauses, restrictions apply to print items. For specific information about restrictions, see the descriptions of individual clauses and commands in the *SQL/MP Reference Manual*.)

A column identifier identifies one of the columns in the result of a SELECT command (a column in the select list) or a named column in the detail line. The column identifier can be a column name, a column number, an alias name, or a detail alias name. A column name is the name of a column specified in the select list. You must qualify an unqualified name if it is the same as any other unqualified name in the select list. For example, if the select list includes EMPLOYEE.DEPTNUM and DEPT.DEPTNUM, you must qualify these names when specifying a column identifier. If only EMPLOYEE.DEPTNUM is in the select list, you can omit the qualifier.

A column number specifies the column in a numeric position of the select list. The first item in the select list is COL 1. You can use this form to refer to literals and expressions.

An alias name is defined in a NAME command. You can use an alias name to refer to a name you assign to a literal or expression.

A detail alias name is defined in the NAME clause of a DETAIL command. You can use a detail alias name as a column identifier in any command except DETAIL.

A print list is a list of print items. You can also specify print lists within another print item in a CONCAT and IF/THEN/ELSE clause.

The print list you specify in a BREAK TITLE, BREAK FOOTING, DETAIL, PAGE TITLE, PAGE FOOTING, REPORT TITLE, or REPORT FOOTING report formatting command defines a logical line. Depending on margin settings, device widths, and use of the SKIP clause, a logical line might be displayed or printed on multiple physical lines.

A logical line can fill at most 512 single-byte print positions. To compute the number of positions, add together the field widths of all print items and the number of spaces between items. For a print item formatted with the AS clause descriptor Cn.w, use w as the field width. For example, if a VARCHAR column named JOBRESP is formatted with AS C0.30, use 30 as the field width for that item.

Report Formatting Commands

Report formatting commands specify the content of detail lines, titles, footings, and column headings. Report formatting commands can also calculate subtotals of values in columns at specified break points, as well as calculate the total value of a column over all rows in the report.

The report formatting commands are summarized in Table 1-1.

Table 1-1. Report Formatting Commands

Command	What the Command Defines
BREAK FOOTING	Text to print at the end of a break group
BREAK ON	The grouping of detail lines by column (print item) value
BREAK TITLE	Text to print at the start of a break group
DETAIL	Detail line content and format
NAME	Name for a column in the select list (to be used in report definition only)
PAGE FOOTING	Text to print at the bottom of each page
PAGE TITLE	Text to print at the top of each page
REPORT FOOTING	Text to print at the end of the report
REPORT TITLE	Text to print at the start of the report
SUBTOTAL	Which items to subtotal and when to calculate the subtotal
TOTAL	Which items to total

Report Formatting Command Clauses

Clauses in report formatting commands specify display formats for print items. The report formatting command clauses are summarized in Table 1-2

Table 1-2. Report Formatting Command Clauses (page 1 of 2)

Clause	What the Clause Specifies
AS	Display format for a print item
AS DATE/TIME	Display format for a date, time, or date and time value
CONCAT	Print one or more print items without intervening spaces
IF/THEN/ELSE	Condition for printing items
NEED	Number of lines required on a page to print the following print items
PAGE	Advance to the next page and optionally start a new page-number sequence

Table 1-2. Report Formatting Command Clauses (page 2 of 2)

Clause	What the Clause Specifies
SKIP	Advance a specific number of lines before printing the next print item
SPACE	Number of blanks to print before the next print item
TAB	Print position of the next print item

Note. For detailed descriptions of the NEED, PAGE, SKIP, SPACE, and TAB clauses, see the DETAIL command in the *SQL/MP Reference Manual*.

Style Options

Style options control the appearance of print items in a report. The style options are summarized in Table 1-3.

Table 1-3. Style Options

Style Option	Default	What the Option Defines
DATE_FORMAT	M2/D2/Y2	Default format for date values
DECIMAL_POINT	Period (.)	Single-byte symbol for decimal point
HEADINGS	ON	Whether headings are generated
NEWLINE_CHAR	Slash (/)	Single-byte character to indicate a new line in a heading
NULL_DISPLAY	Question mark (?)	Single-byte character to represent null values
OVERFLOW_CHAR	Asterisk (*)	Single-byte filler character printed when a value exceeds a field size
ROWCOUNT	ON	Whether the row-count line is generated
SUBTOTAL_LABEL	*	Label to print in a break column when a subtotal is printed
TIME_FORMAT	HP2:M2:S2	Default format for time values
UNDERLINE_CHAR	Hyphen (-)	Single-byte character to be the underline
VARCHAR_WIDTH	80	Maximum number of characters in a variable-length print item

Layout Options

Layout options control the appearance of a report on the screen or printed page. The layout options are summarized in Table 1-4.

Table 1-4. Layout Options

Layout Option	Default	What the Option Defines
CENTER_REPORT	OFF	Centering of the entire report
LEFT_MARGIN	0	Left margin of the report
LINE_SPACING	1	Number of lines to advance before the next line
LOGICAL_FOLDING	ON	Whether an item of the default detail line moves to the next line when the item does not fit within the margins
PAGE_COUNT	ALL	Maximum number of pages
PAGE_LENGTH	ALL (terminal) 60 (other devices)	Number of lines per page
RIGHT_MARGIN	Output device width	Right margin of the report
SPACE	2	Number of single-byte spaces between columns
WINDOW	TAB 1	Print item or print position to display at the left edge of the output device

Report Function

Report functions provide timestamps (both current and for a specified date and time), the current line number, and the current page number. The report functions are summarized in Table 1-5.

Table 1-5. Report Functions

Function	Value Returned
COMPUTE_TIMESTAMP	Timestamp for a specified date and time
CURRENT_TIMESTAMP	Timestamp for the current date and time
LINE_NUMBER	Current line number within a break group, page, or report
PAGE_NUMBER	Current page number

SQLCI Commands

Table 1-6 summarizes the SQLCI commands that are useful when generating reports.

Table 1-6. SQLCI Commands

Command	Description
CANCEL	Cancels the current SELECT command.
EXECUTE	Executes a compiled command.
LIST	Displays rows retrieved by the SELECT command.
LOG	Starts or ends the logging of session activity to a file.
OUT	Specifies or closes the output file.
OUT_REPORT	Directs the output of a SELECT command to a specific report file or closes the current report file.
PREPARE	Compiles a command. Useful for compiling a SELECT command before producing a report.
RESET LAYOUT	Resets layout options to default settings.
RESET REPORT	Deletes commands from the current report definition or deletes columns or alias names from stored report formatting commands in the current report definition.
RESET SESSION	Resets session options to default settings.
RESET STYLE	Resets style options to default settings.
SELECT	Retrieves data from tables and views.
SET LAYOUT	Sets layout options to new values.
SET SESSION	Sets session options to new values. Only the LIST_COUNT session option is described in this manual.
SET STYLE	Sets style options to new values.
SHOW LAYOUT	Displays the values of layout options.
SHOW REPORT	Displays report formatting commands and the most recent SELECT command.
SHOW SESSION	Displays the values of session options.
SHOW STYLE	Displays the values of style options.

Note. The LIST_COUNT session option specifies the number of rows the LIST and SELECT commands display.

2

Using SQLCI and the Report Writer

You can control the environment of a report writing session by using general SQLCI commands. To develop a report, you can do the following:

- Start and exit an SQLCI session.
- Set up attributes of your session environment such as default options, the default volume, and output files.
- Use windows to view vertical segments of a report.
- Enter report formatting commands.
- Execute commands repeatedly.
- Define reports in command files.

For detailed syntax and descriptions of the SQLCI commands introduced in this section, see the *SQL/MP Reference Manual*.

Starting and Ending an SQLCI Session

To start an SQLCI session, enter the following at the TACL prompt:

```
TACL 5> SQLCI
SQL Conversational Interface ...
>>
```

When waiting for a command, SQLCI displays the standard SQLCI prompt (>>). If you have not completed a command by entering a semicolon (;), SQLCI displays the command continuation prompt (+>) until a semicolon is entered.

When a SELECT command is in progress, SQLCI displays the select-in-progress prompt (S>), indicating that SQLCI is waiting for report commands. To get this prompt, you must set the LIST_COUNT session option before you enter a SELECT command. For more information, see Listing Rows of a Report on page 2-5.

To end an SQLCI session, enter the following at the SQLCI prompt:

```
>> EXIT
```

Setting Up Your Session Environment

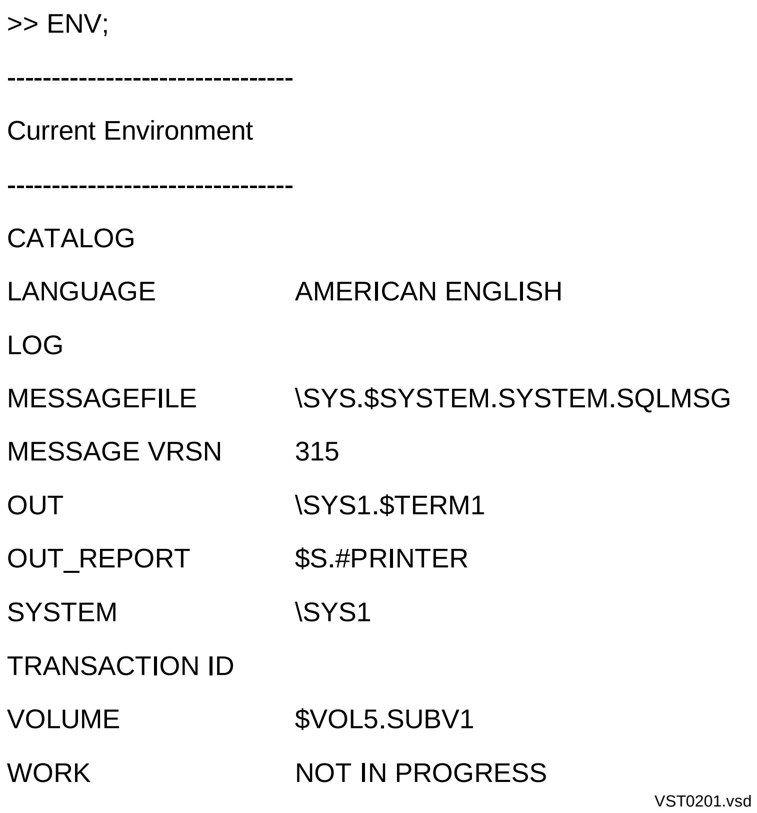
Any report you create is affected by the current environment of your SQLCI session, such as the current default system, volume, and subvolume. When you specify the name of a table or view, you can indicate the volume and subvolume that contain the object by specifying the Guardian name of the object: for example:

```
 \SYS1.$VOL1.SALES.ORDERS.
```

If an object is on the current default volume and subvolume, you can omit everything but the table or view name. The system expands the name by using the current default system, volume, and subvolume.

You can use the ENV command to display the current default settings, as shown in Figure 2-1.

Figure 2-1. Setting Up Your Session Environment



```
>> ENV;

--------------------------------

Current Environment

--------------------------------

CATALOG

LANGUAGE            AMERICAN ENGLISH

LOG

MESSAGEFILE         \SYS.$SYSTEM.SYSTEM.SQLMSG

MESSAGE VRSN        315

OUT                 \SYS1.$TERM1

OUT_REPORT          $S.#PRINTER

SYSTEM              \SYS1

TRANSACTION ID

VOLUME              $VOL5.SUBV1

WORK                NOT IN PROGRESS
```

VST0201.vsd

If you have not set the default system, the default is the system on which you started your session. The system name appears in the environment report only if you specify the name in a SYSTEM command or in an ALTER DEFINE command that changes the =_DEFAULTS DEFINE. For more information about DEFINEs, see the *SQL /MP Reference Manual*.

Other elements of your session environment are:

CATALOG The current default catalog; this catalog does not affect your report writing activity.

LANGUAGE The language of the text in the message file.

LOG The file to which SQLCI logs your session activity. SQLCI writes the commands you enter and the output produced by the commands to the log file. For example, a SELECT command and the rows selected by the command are written to the log file. You specify the log file in a LOG command. You can request that only commands be logged.

MESSAGEFILE The name of the current SQL message file.

MESSAGE VRSN The version of the NonStop SQL/MP software in use.

OUT The file to which SQLCI writes output from commands, prompts, and messages. By default, the OUT file is usually your terminal. You can enter the OUT command to specify another OUT file.

OUT_REPORT The file to which SQLCI writes the output from SELECT commands. If you have not entered an OUT_REPORT command to specify an OUT_REPORT file, the output from SELECT commands is written to the OUT file.

TRANSACTION ID The transaction identifier of the current TMF transaction, if one is in progress.

WORK The current state of TMF transactions. IN PROGRESS indicates a TMF transaction has been initiated. NOT IN PROGRESS indicates that no transaction has been initiated since any previous transaction was committed or aborted.

Setting the Default Volume and Subvolume

To set the default volume and subvolume, enter the VOLUME command:

```
>> VOLUME $VOL1.PERSNL;
```

The system you are using to run SQLCI is the default system unless you enter a SYSTEM or VOLUME command to specify some other default system.

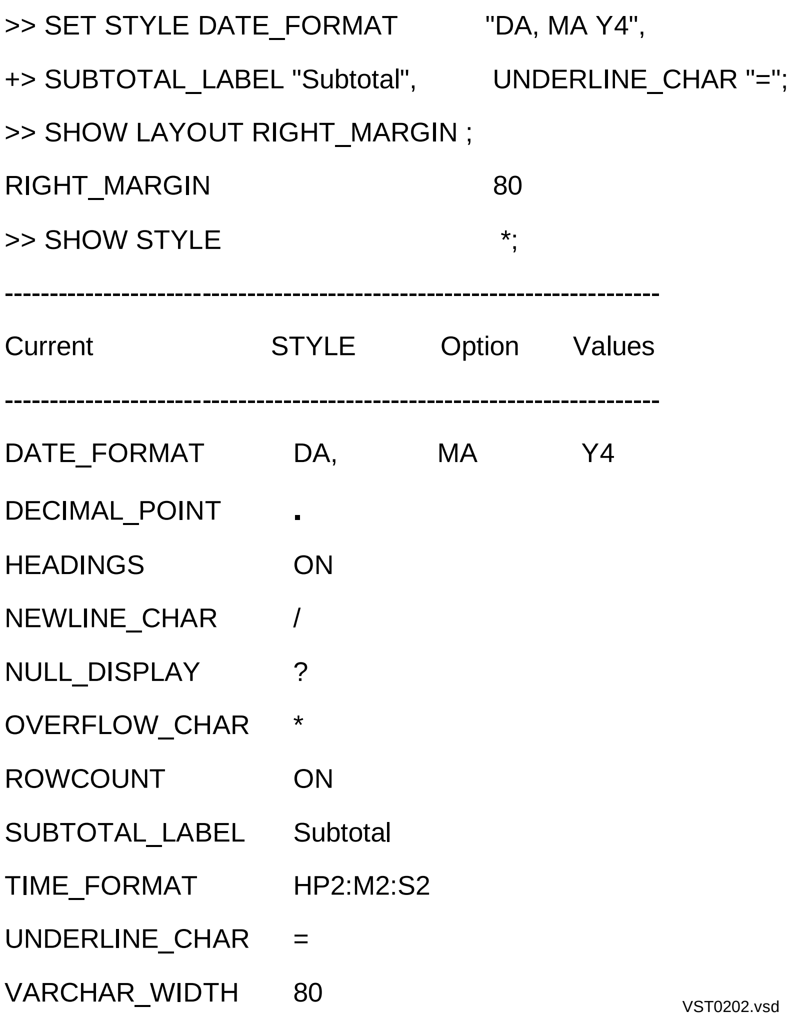
Setting and Displaying Options

There are three basic commands for working with options: SET, RESET, and SHOW. Layout options, such as line spacing and margin settings, control the appearance of a report on the screen or printed page.

Style options, such as the decimal character and underline character, control the appearance of items in a report.

Session options primarily affect the way SQLCI communicates with you: for example, how many lines of SELECT output appear before a pause, how SQLCI responds to the BREAK key, whether output lines that exceed the output device width are wrapped or truncated, and whether warning messages are displayed. Figure 2-2 illustrates the setting and displaying options.

Figure 2-2. Setting and Displaying Options



```
>> SET STYLE DATE_FORMAT          "DA, MA Y4",
+> SUBTOTAL_LABEL "Subtotal",      UNDERLINE_CHAR "=";
>> SHOW LAYOUT RIGHT_MARGIN ;
RIGHT_MARGIN                       80
>> SHOW STYLE                       *;
---------------------------------------------------------------------
Current              STYLE       Option     Values
---------------------------------------------------------------------
DATE_FORMAT          DA,         MA         Y4
DECIMAL_POINT        .
HEADINGS             ON
NEWLINE_CHAR         /
NULL_DISPLAY         ?
OVERFLOW_CHAR        *
ROWCOUNT             ON
SUBTOTAL_LABEL       Subtotal
TIME_FORMAT          HP2:M2:S2
UNDERLINE_CHAR       =
VARCHAR_WIDTH        80
```

VST0202.vsd

You can use the RESET, RESET LAYOUT, and RESET STYLE commands to reset specific options or all options to their default values. For example, the following commands reset the line spacing to 1 (single spacing) and reset all style options to their default values:

```
>> RESET LAYOUT LINE_SPACING;
>> RESET STYLE * ;
```

You can enter SET, RESET, and SHOW commands at the standard SQLCI prompt (>>) or the select-in-progress prompt (S>).

In the tasks described in this manual, you can see more illustrations of how options affect your reports. Except for LIST_COUNT, all options are set to their default values unless the example specifically sets the option. All examples have LIST_COUNT set to

0. For more information on the summary of the options, see Report Writer Components on page 1-10.

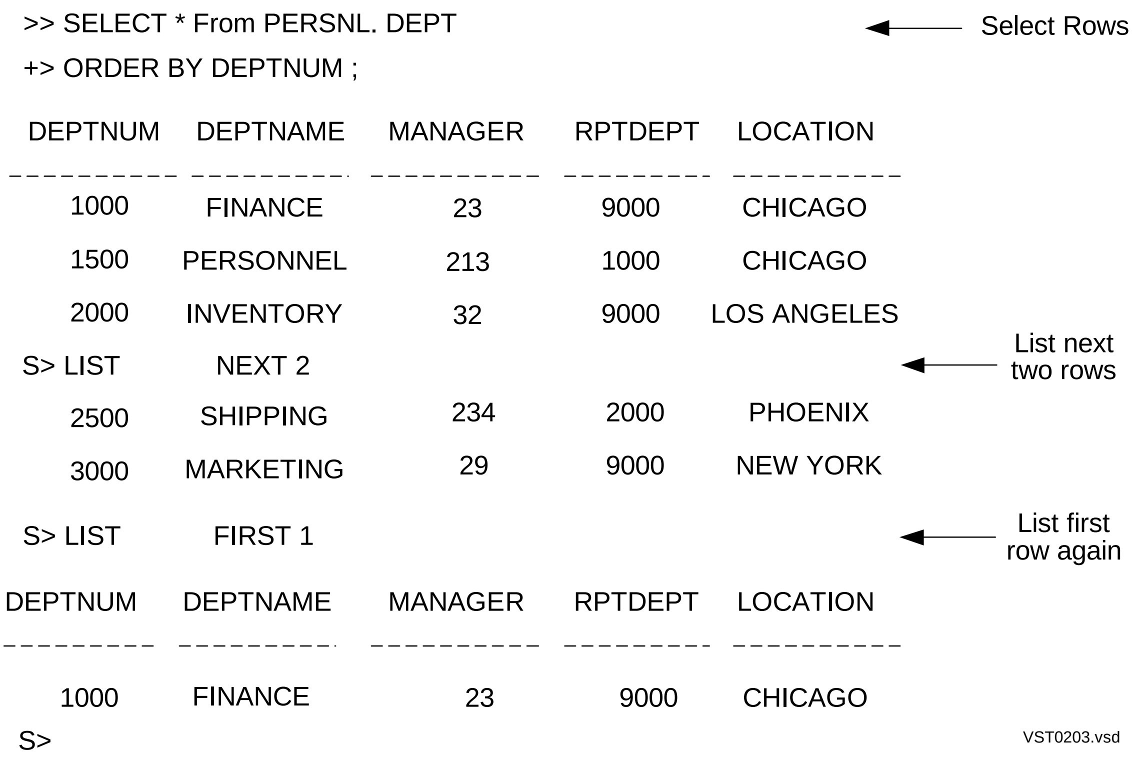
Listing Rows of a Report

You retrieve rows of data by using the SELECT command. You can specify the number of rows to be displayed or printed by setting the LIST_COUNT session option and by using the LIST command. By default, LIST_COUNT is set to ALL when you begin your SQLCI session. Unless you want all retrieved rows displayed, you should set the LIST_COUNT value before you enter a SELECT command. For example, the following command specifies listing 3 rows before a pause:

```
>> SET LIST_COUNT 3;
```

After the first three rows appear, SQLCI displays the select-in-progress prompt (S>). As shown in Figure 2-3, you can use the LIST command to display rows following the last displayed row or to return to the first retrieved row before displaying more rows.

Figure 2-3. Example of Listing Rows



```
>> SELECT * From PERSNL. DEPT
+> ORDER BY DEPTNUM ;

 DEPTNUM    DEPTNAME     MANAGER    RPTDEPT   LOCATION
---------- ----------  ---------- --------- ----------
    1000     FINANCE        23        9000    CHICAGO
    1500    PERSONNEL      213        1000    CHICAGO
    2000    INVENTORY       32        9000   LOS ANGELES
S> LIST       NEXT 2
    2500     SHIPPING      234        2000    PHOENIX
    3000    MARKETING       29        9000    NEW YORK

S> LIST       FIRST 1
DEPTNUM    DEPTNAME      MANAGER    RPTDEPT   LOCATION
---------- ----------  ---------- --------- ----------
   1000     FINANCE         23        9000    CHICAGO
S>
```

You can abbreviate LIST (L), FIRST (F), NEXT (N), and ALL (A). Pressing RETURN at the S> prompt is the same as entering LIST NEXT n (specifying n as the current LIST_COUNT value).

The number you specify for the value of n relates to rows. For example, if the information from a row appears on two output lines and you specify LIST NEXT 3, six output lines are listed in addition to any headings, titles, footings, and other output that is not counted.

LIST ALL lists all selected rows and then returns you to the standard prompt and cancels the SELECT command.

Canceling a SELECT Command

To stop listing rows and cancel a SELECT command, enter the CANCEL command at the S> prompt:

```
S> CANCEL;
```

If you accidentally cancel the SELECT command, you can execute it again. For more information, see Executing a Command Repeatedly on page 2-13.

Defining Options for Line Wrapping

The WRAP session option relates to the output device width. If the right margin is beyond the right most position of the device, WRAP determines whether a line is wrapped or truncated.

Note. If you are using a double-byte character set, such as Tandem Kanji or Tandem KSC5601, see Printing Double-Byte Characters on page 4-58 for guidelines to follow when using the WRAP option.

Defining Options for Line Folding

The report writer also folds lines. Folding relates to the area defined within the margins. If the information in a print list does not fit within the margins, the information is always folded to the next line.

For the default detail line only, you can set the LOGICAL_FOLDING layout option to control how information is folded. When LOGICAL_FOLDING is ON, a single item is never split and continued on the next line. The default setting for

LOGICAL_FOLDING is ON. If LOGICAL_FOLDING is OFF, then an item might be split. If the information fits within the margins, LOGICAL_FOLDING has no effect regardless of the setting of the WRAP option and whether or not the output line fits on the device.

Note. If you are using a double-byte character set, such as Tandem Kanji or Tanderm KSC5601, see Printing Double-Byte Characters on page 4-58 for guidelines to follow when using the LOGICAL_FOLDING option.

Defining a Window for Report Output

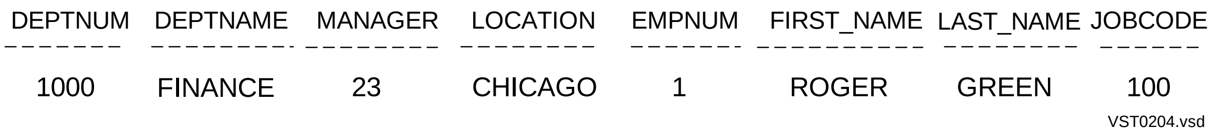
If your report is wider than the maximum width of the device on which you are displaying or printing it, you can use the SET LAYOUT WINDOW command to specify which vertical portion of the report you want to see. For example, suppose a report is designed for a printer that prints 120 single-byte characters per line. The margins are set and the SELECT command select list is specified as follows:

```
>> SET LAYOUT LEFT_MARGIN 8;
>> SET LAYOUT RIGHT_MARGIN 120;
>> SELECT D.DEPTNUM, DEPTNAME, MANAGER, LOCATION, EMPNUM,
+> FIRST_NAME, LAST_NAME, JOBCODE
.
.
+> ... ;
```

The report is produced in the default format.

If you want to display the report on your terminal, the text that does not fit within the device width is wrapped to the next line by default. For example, the headings and first detail line appear as follows:

Figure 2-4. Defining a Window for Report Output

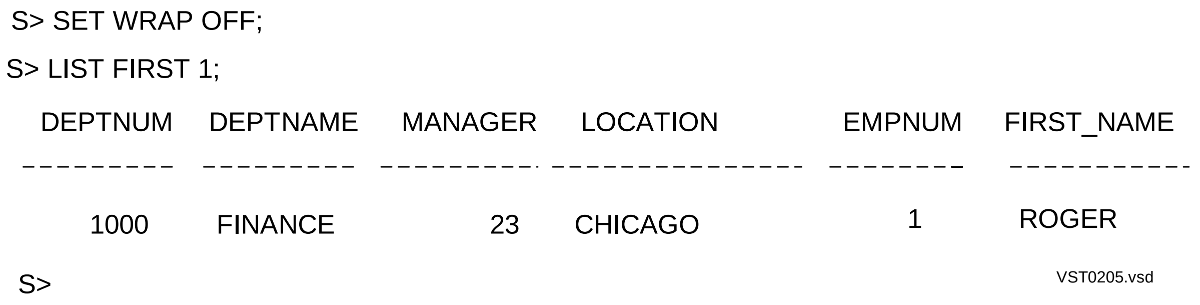


```
 DEPTNUM  DEPTNAME  MANAGER  LOCATION  EMPNUM  FIRST_NAME LAST_NAME JOBCODE
-------  --------- -------- --------  ------- ---------- --------- ------

  1000    FINANCE     23     CHICAGO     1       ROGER      GREEN     100
```

VST0204.vsd

As you can see, the report is difficult to read in this arrangement. If you want to see the report as it would appear on the printer without the wrapped lines, set the WRAP session option to OFF and use the SET LAYOUT WINDOW command to view different parts of the report:

Figure 2-5. Defining a Window for Report Output

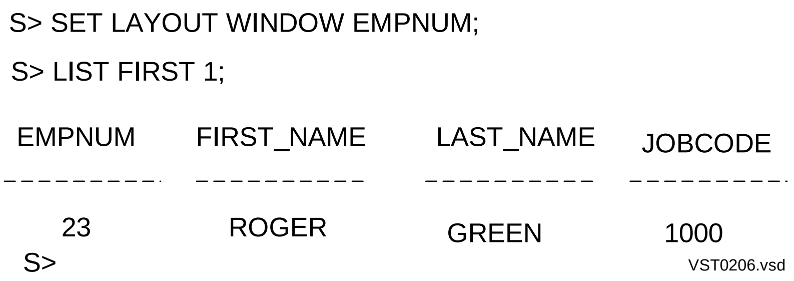


```
S> SET WRAP OFF;
S> LIST FIRST 1;
   DEPTNUM    DEPTNAME    MANAGER    LOCATION          EMPNUM    FIRST_NAME
 ---------  ---------  ---------- ---------------  --------   -----------

     1000      FINANCE          23    CHICAGO              1       ROGER

S>
```

VST0205.vsd

To make the EMPNUM column appear at the left edge of the output, enter WINDOW EMPNUM as shown in Figure 2-6:

Figure 2-6. Defining a Window for Report Output



```
S> SET LAYOUT WINDOW EMPNUM;
S> LIST FIRST 1;

EMPNUM      FIRST_NAME     LAST_NAME    JOBCODE
----------  -----------    -----------  ----------

     23        ROGER         GREEN         1000
S>
```

VST0206.vsd

In the preceding example, you can specify SET LAYOUT WINDOW TAB 60 to produce the same result. You can use the TAB form of SET LAYOUT WINDOW at either the standard prompt or the select-in-progress prompt. You must specify the print position relative to the full output line, including the margin. The first print position is always the same as position LEFT_MARGIN 0. For example, if you set LEFT_MARGIN to 5, SET LAYOUT WINDOW TAB 5 moves the character at the left margin to the left edge of the window.

At the select-in-progress prompt only, you can define a window by the column name or column number of the left most select item you want to display. For example, to display the fourth column of the select list, enter the following:

```
S> SET LAYOUT WINDOW COL 4;
```

Figure 2-7 illustrates the relation of the left edge of a window to the output line of a report. At each numbered step, the headings and first row are displayed.

At (1), the following layout options are in effect:

- The left margin is 5; five blank positions of the output line precede the detail line content.
- The right margin is 120; the output line ends at column 120. If the information in the detail line does not fit on one line, the information continues on the next line at the left margin.
- The device width for the terminal is assumed to be 64.
- WRAP is set OFF.
- The default window begins at position TAB 1 (the first print position), which is in the margin.
- The first detail line begins with the department number field; the field is 7 characters wide, and the number is right justified.

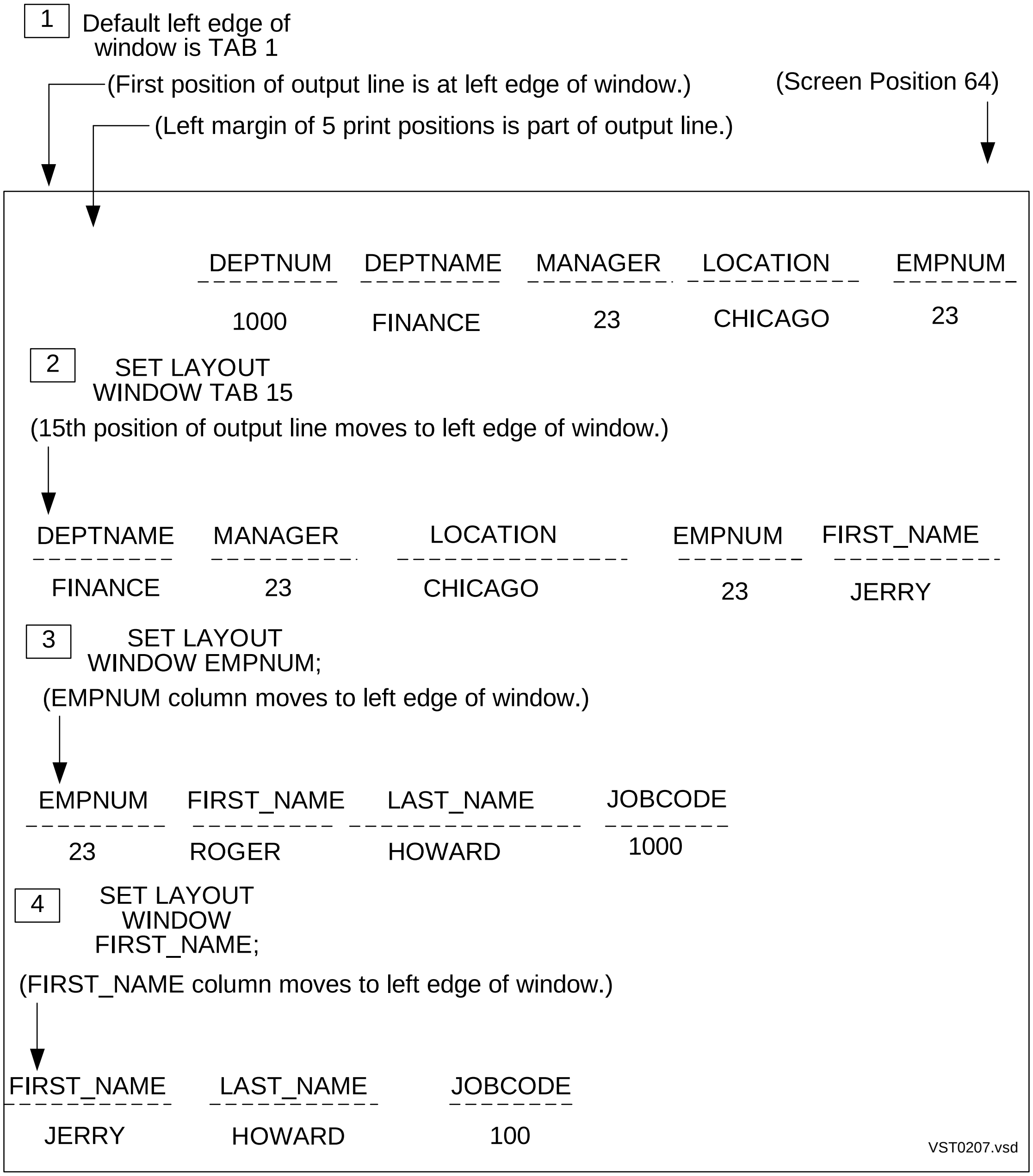
At (2), the window is specified to begin at TAB 15.

At (3), the window is specified to begin at column EMPNUM.

At (4), the window is specified to begin at column FIRST_NAME.

For an overview of how to set margins, see Setting Margins on page 4-1.

Figure 2-7. Relation of Window to Output Line

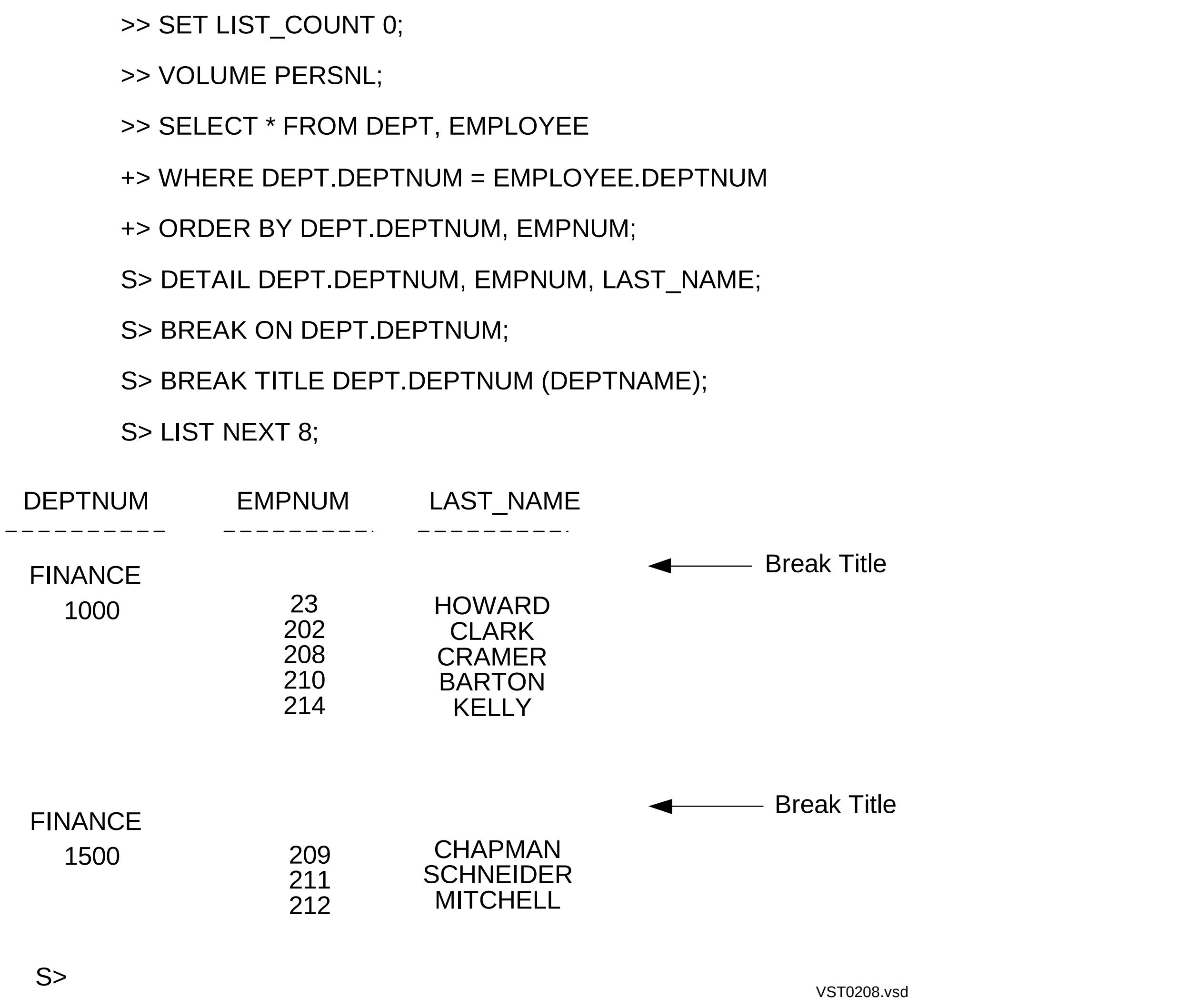

1
Default left edge of window is TAB 1
(First position of output line is at left edge of window.)
(Left margin of 5 print positions is part of output line.)
(Screen Position 64)
DEPTNUM DEPTNAME MANAGER LOCATION EMPNUM
1000 FINANCE 23 CHICAGO 23
2
SET LAYOUT WINDOW TAB 15
(15th position of output line moves to left edge of window.)
DEPTNAME MANAGER LOCATION EMPNUM FIRST_NAME
FINANCE 23 CHICAGO 23 JERRY
3
SET LAYOUT WINDOW EMPNUM;
(EMPNUM column moves to left edge of window.)
EMPNUM FIRST_NAME LAST_NAME JOBCODE
23 ROGER HOWARD 1000
4
SET LAYOUT WINDOW FIRST_NAME;
(FIRST_NAME column moves to left edge of window.)
FIRST_NAME LAST_NAME JOBCODE
JERRY HOWARD 100
VST0207.vsd

Entering Report Formatting Commands

The report formatting commands must be associated with a specific SELECT command. You can enter these commands only at the select-in-progress prompt (S>). This prompt appears after you enter the SELECT command, but only if the number of rows to be listed is greater than the current value of the LIST_COUNT session option. If you want to enter report formatting commands, you should set a low LIST_COUNT value: for example, 0 or 1.

In Figure 2-8, DETAIL, BREAK ON, and BREAK TITLE are report formatting commands that refer to the selected data.

Figure 2-8. Example of Report Formatting Commands



```
>> SET LIST_COUNT 0;
>> VOLUME PERSNL;
>> SELECT * FROM DEPT, EMPLOYEE
+> WHERE DEPT.DEPTNUM = EMPLOYEE.DEPTNUM
+> ORDER BY DEPT.DEPTNUM, EMPNUM;
S> DETAIL DEPT.DEPTNUM, EMPNUM, LAST_NAME;
S> BREAK ON DEPT.DEPTNUM;
S> BREAK TITLE DEPT.DEPTNUM (DEPTNAME);
S> LIST NEXT 8;

 DEPTNUM       EMPNUM      LAST_NAME
----------   ----------   ----------

 FINANCE
   1000          23         HOWARD
                202         CLARK
                208         CRAMER
                210         BARTON
                214         KELLY


 FINANCE
   1500         209         CHAPMAN
                211         SCHNEIDER
                212         MITCHELL

 S>
```

There can be only one version of each of the following commands in effect at one time:

- BREAK ON
- DETAIL
- PAGE FOOTING
- PAGE TITLE
- REPORT FOOTING
- REPORT TITLE
- TOTAL

If you enter a second version of any of these commands, that version replaces the previous one. A report formatting command is canceled if you do any of the following:

- Cancel the current SELECT command
- Enter LIST ALL
- Reset the command using RESET REPORT

You can have more than one BREAK FOOTING, BREAK TITLE, or SUBTOTAL command in effect at a time, but only one of each type of command can refer to a specific break column defined in the BREAK ON command.

For example, suppose you enter the following BREAK ON command:

```
S> BREAK ON DEPTNUM, JOBCODE;
```

You can request a subtotal of the first column for every department number as follows:

```
S> SUBTOTAL COL 1 OVER DEPTNUM;
```

Column 1 will be subtotaled whenever the value of DEPTNUM changes.

To request a subtotal of the second column whenever the job code changes, enter the following:

```
S> SUBTOTAL COL 2 OVER JOBCODE;
```

If you enter another SUBTOTAL OVER command for the same break item, the previous SUBTOTAL OVER command for that item is replaced.

For example, the following SUBTOTAL command replaces the previous SUBTOTAL OVER JOBCODE command:

```
S> SUBTOTAL COL 3 OVER JOBCODE;
```

If you enter a SUBTOTAL command without an OVER clause, the items specified will be subtotaled whenever the value of any break item changes. This command provides a short way of specifying SUBTOTAL OVER for all break items.

You can have only one version of a SUBTOTAL command without an OVER clause current at one time. In the following example, the SUBTOTAL COL 1 command specifies that column 1 will be subtotaled whenever the value of any break item changes. However, if you enter the SUBTOTAL COL 2 command, only column 2 will be subtotaled:

```
S> BREAK ON DEPTNUM, JOBCODE;
S> SUBTOTAL COL 1;
       .
       .
S> SUBTOTAL COL 2;
```

Suppose you enter the following commands:

```
S> BREAK ON DEPTNUM;
S> BREAK TITLE DEPTNUM ("Department Number:", DEPTNUM);
```

If you decide you want to change the break title, reenter the command:

```
S> BREAK TITLE DEPTNUM ("Dept. No. ", DEPTNUM);
```

The second BREAK TITLE command replaces the first.

If you enter BREAK ON, SUBTOTAL, or TOTAL to the report format after listing some rows, the report writer returns you to the beginning of the SELECT output, as if you had entered the LIST FIRST command.

Specifying Output Files

There are three types of files to which you can direct different types of output when using SQLCI:

- OUT files
- OUT_REPORT files
- Log files

OUT Files

By default, all SQLCI output is directed to your terminal during an SQLCI session. However, you can direct SQLCI output to a specific file by using the OUT command. Enter the following command to direct SQLCI output to a spooler collector for a printer:

```
>> OUT $S.#PRINTZ;
```

All information that SQLCI produces is written to the OUT file, including output from commands such as SHOW and DISPLAY STATISTICS, data from a SELECT command (unless you specify an OUT_REPORT file), and diagnostic messages.

OUT_REPORT Files

You can direct output from a SELECT command, both default reports and customized reports, to a specific file by using the OUT_REPORT command. Enter the following command to direct reports to a disk file named DRAFT:

```
>> OUT_REPORT DRAFT;
```

The following command directs reports to a spooler collector for a printer:

```
>> OUT_REPORT $S.#PRINTXX;
```

Log Files

If you specify a log file for your session by entering the LOG command, SQLCI logs the commands you enter and the information that the commands produce to a log file. The content of the log file is the same as the information you see on your screen when the OUT file is your terminal and you have not specified an OUT_REPORT file. Optionally, you can specify that you want only the commands written to the log file.

If you want to print a report, and you want the report definition to be printed preceding the report, you can use the log file.

Suppose the file REPORT55 contains all the commands needed to print a report. The first command in the file is:

```
LOG $S.#PRINTER;
```

Enter the following command to print the definition as well as the report:

```
>> OBEY REPORT55;
```

As the commands in the command file execute, they are logged to the log file. The report is written to the log file and to the OUT_REPORT file.

To turn logging off, enter:

```
>> LOG;
```

Executing a Command Repeatedly

There are three ways to execute commands repeatedly:

- Use the FC, !, and HISTORY commands.
- Use the PREPARE and EXECUTE commands. (For an example, see Preparing a SELECT Command on page 3-24.)
- Use the OBEY command.

If you want to execute a set of commands repeatedly, you should put the commands in an command file. You can either use a text editor to create the file or enter the commands interactively and then use the SAVE command to write them to a command file. For more information on examples of creating command files, see Defining Reports in Command Files on page 2-15.

Repeating Stored Commands

You can use the FC and ! commands to repeat a command you have entered during your SQLCI session. The exclamation point command repeats a command but does not allow you to modify it. The FC command allows you to edit the command before reexecuting it.

To execute a previous command that begins with the text SELECT P, enter the following:

```
>> ! SELECT P
```

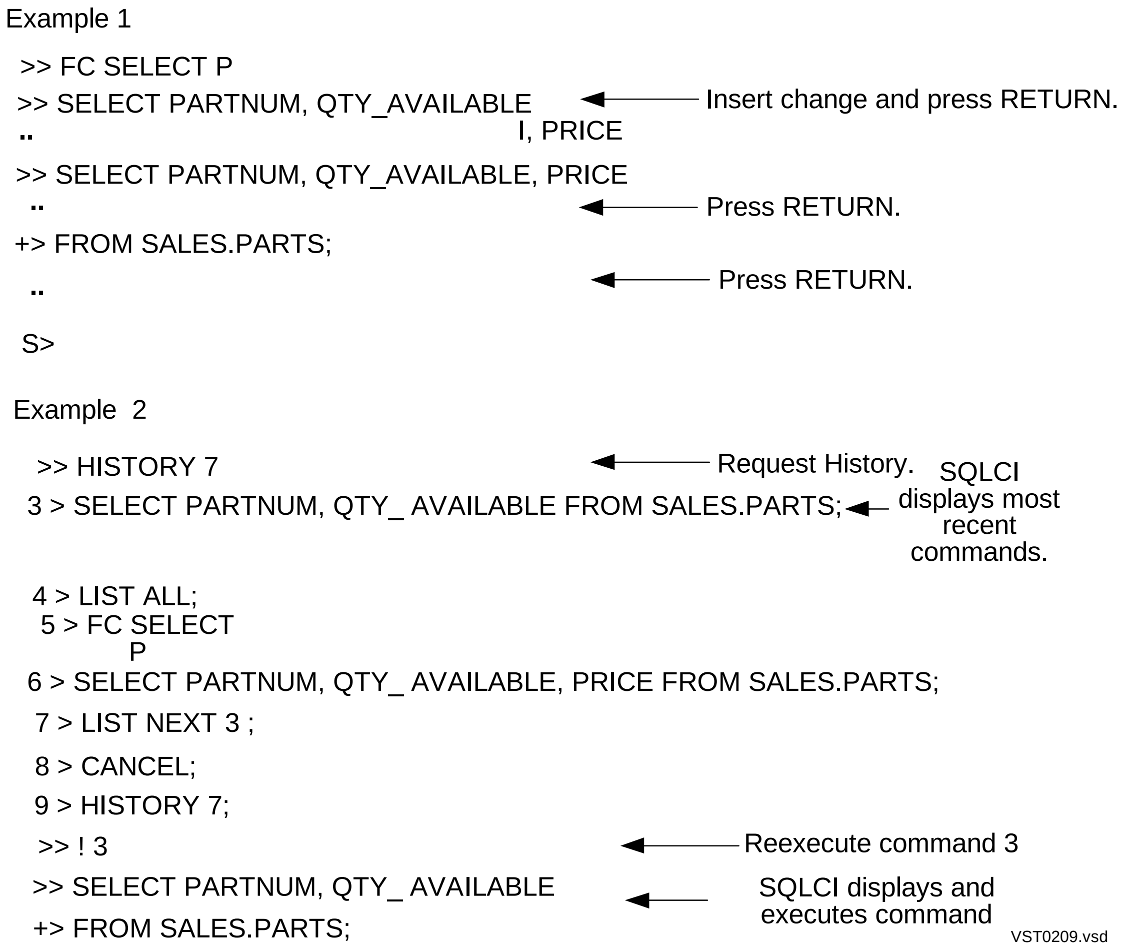
SQLCI displays and executes the command:

```
>> SELECT PARTNUM, QTY_AVAILABLE
+> FROM SALES.PARTS;
```

Figure 2-9 shows two examples of repeating commands. The first example shows how to revise a command and reexecute it by using the FC command. Enter sufficient text to distinguish the command from other previously entered commands.

The second example in Figure 2-9 shows how to display the last seven commands and reexecute command number three by using the HISTORY command. You can use the HISTORY command to display up to 25 of the most recently executed commands.

Figure 2-9. Example of Repeating Stored Commands



Example 1

```
>> FC SELECT P
>> SELECT PARTNUM, QTY_AVAILABLE
..                                  I, PRICE
>> SELECT PARTNUM, QTY_AVAILABLE, PRICE
 ..
+> FROM SALES.PARTS;
 ..
S>
```

Insert change and press RETURN.

Press RETURN.

Press RETURN.

Example 2

```
 >> HISTORY 7
3 > SELECT PARTNUM, QTY_ AVAILABLE FROM SALES.PARTS;
4 > LIST ALL;
 5 > FC SELECT
         P
6 > SELECT PARTNUM, QTY_ AVAILABLE, PRICE FROM SALES.PARTS;
 7 > LIST NEXT 3 ;
 8 > CANCEL;
 9 > HISTORY 7;
 >> ! 3
>> SELECT PARTNUM, QTY_ AVAILABLE
+> FROM SALES.PARTS;
```

Request History.

SQLCI displays most recent commands.

Reexecute command 3

SQLCI displays and executes command

VST0209.vsd

The semicolon terminator is optional for FC and ! commands. You can enter these commands at the S> prompt also. If you are listing rows retrieved by a SELECT command and decide to change the command, use a CANCEL command to return to the SQLCI prompt. Then enter the FC or ! command.

Defining Reports in Command Files

There are three ways to define a report in a command file:

- You can prepare the report in steps. This method lets you view the result of the report formatting commands you enter as you work and then save the report commands in a file when you are satisfied with the report format.
- You can create the report commands in a command file using a text editor.
- You can log only the commands to a log file and then use the log file as a command file.

Saving Report Commands in an Command file

To create a report and test the results as you work, follow these steps:

1. If you want to enter report formatting commands, you must prevent the entire SELECT command output from displaying immediately. To do this, set the LIST_COUNT option to a number less than the number of rows that will be retrieved. If you do not know how many rows will be selected, set LIST_COUNT to 0:

```
>> SET LIST_COUNT 0;
>>
```

2. Enter the SELECT command. If you specify the SELECT command correctly, the S> prompt appears. If you have made any errors, an error message appears. Use the FC command to correct any errors in the command.

```
>> SELECT O.ORDERNUM, ORDER_DATE, CUSTNUM,
+> PARTNUM, UNIT_PRICE, QTY_ORDERED
+> FROM SALES.ORDERS O, SALES.ODETAIL OD
+> WHERE O.ORDERNUM = OD.ORDERNUM
+> ORDER BY CUSTNUM, O.ORDERNUM, ORDER_DATE, PARTNUM;
S>
```

3. Enter one or more report formatting commands. For example, you might want to define the detail line. In a DETAIL command you can specify a subset of the select list, rearrange the select list items, and specify expressions based on select list items. You can add other commands such as BREAK ON.

```
S> DETAIL O.ORDERNUM,
+> PARTNUM,
+> UNIT_PRICE,
+> QTY_ORDERED,
+> UNIT_PRICE*QTY_ORDERED;
S> BREAK ON O.ORDERNUM ;
S> LIST FIRST 5;

ORDERNUM    PARTNUM    UNIT_PRICE     QTY_ORDERED     (EXPR)

---------- ------- ------------ ----------- -----------------
---

200320        5504     165.00            5              825.00

              6201     195.00           16             3120.00

              6301     245.00            6             1470.00

              6400     540.00            7             3780.00

200490        3210     715.00            1              715.00

S>
```

You might want to change a layout or style option. For example, you can increase the number of spaces between columns to 5 as follows:

```
S> SET LAYOUT SPACE 5;
S> LIST FIRST 1;
ORDERNUM PARTNUM UNIT_PRICE

-------- ------- ----------
```

Continue adding and revising formatting commands and listing lines until you are satisfied with your report definition.

To return to the standard SQLCI prompt, enter either LIST ALL or CANCEL. Remember that LIST ALL and CANCEL delete all formatting commands.

4. Use the following SAVE commands to save in a file the final version of the SELECT command, each report formatting command, and each option. In this example, the file is named ORDFILE:

```
>> SAVE SESSION * TO ORDFILE CLEAR;
>> SAVE LAYOUT * TO ORDFILE;
>> SAVE STYLE * TO ORDFILE;
>> SAVE REPORT * TO ORDFILE;
```

SAVE SESSION * saves all session options, including LIST_COUNT. CLEAR clears the previous contents of ORDFILE. SAVE REPORT * saves the SELECT command and all report formatting commands. The other SAVE commands save the layout and style options.

You must save the SET LIST_COUNT command before the SELECT command; otherwise, the report formatting commands will not execute. If you have not set any other session options that you want to save, you can save only LIST_COUNT by entering:

```
SAVE LIST_COUNT TO ORDFILE CLEAR;
```

You can include more than one report definition in a file by defining sections. A section begins with the following header line:

```
?SECTION section-name
```

Each section can contain a different report definition or portion of a report definition that you want to execute separately. For example, to execute a section named SECT1 in the file named REPORTS, enter:

```
>> OBEY REPORTS (SECT1);
```

Remember these points when saving a report definition:

- If you are using the default settings for the layout and style options, you do not have to save these options. You can either include a RESET LAYOUT * and RESET STYLE * command in the command file, or you can enter these commands before you execute the command file. If you know the default settings are in effect, you can simply execute the command file.

- You can save other commands such as the VOLUME command. For example, the following command saves the command that begins with the letters VOLUME $I from the history buffer:

```
SAVE COMMAND VOLUME $I TO ORDFILE;
```

You can use the HISTORY command to display commands you might want to save. The order in which you save the commands can be significant; for example, you should save the VOLUME command before you save a SELECT command that depends on the correct default volume being set.

5. You can produce the report again by entering the OBEY command at the standard prompt:

```
>> OBEY ORDFILE ;
```

The select-in-progress prompt appears after you execute the file. You can use the LIST command to list all or part of your report:

```
S> LIST ALL;
```

To print your report, use the OUT_REPORT command before you enter the OBEY command to specify a printer as the output device for reports as shown in this example:

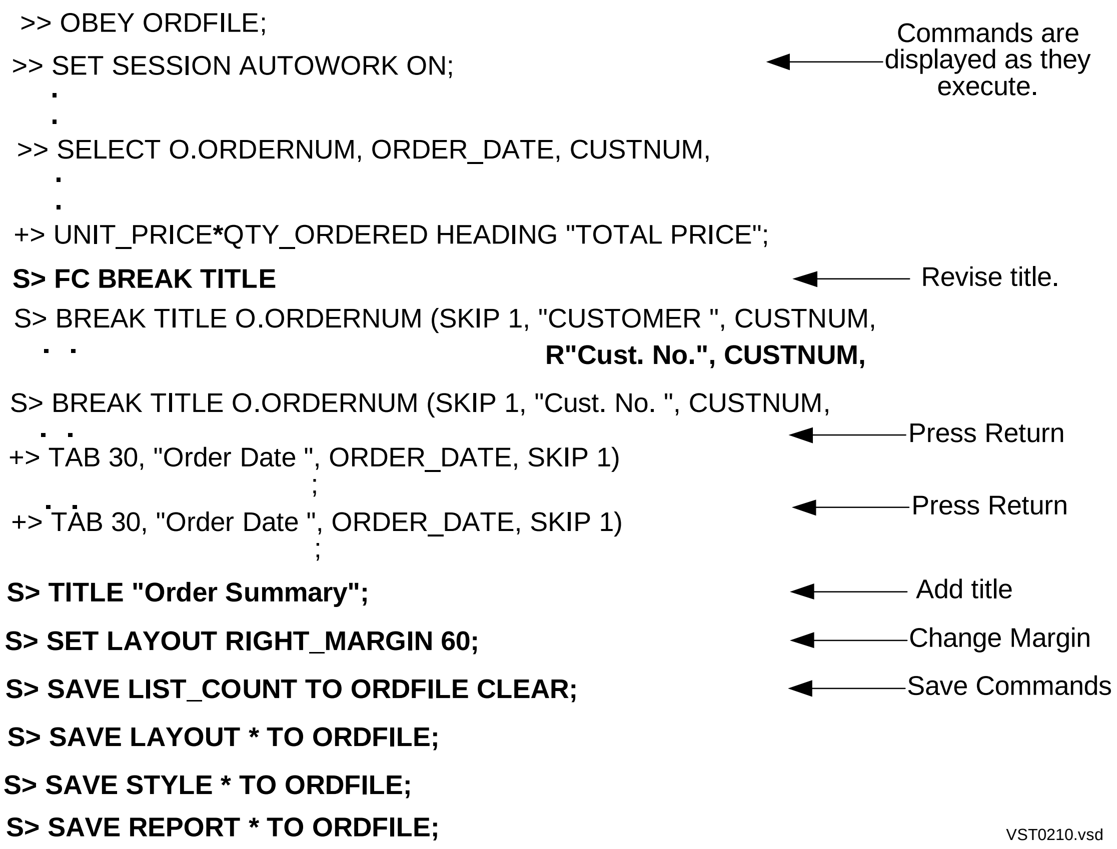
```
>> OUT_REPORT $S1.#PRINT;
>> OBEY ORDFILE;
```

When you are defining a report interactively, remember:

- You can change part of the report by entering a single command again without entering all of the commands.
- If you enter a BREAK, SUBTOTAL, or TOTAL command to the report format after listing some rows, the report writer returns you to the beginning of the SELECT output, as if you had entered the LIST FIRST command.

If the command file does not include a LIST command, you can make changes to the report definition at the select-in-progress prompt (S>) as shown in Figure 2-10. Thus, you can revise commands if necessary and save the report definition again by using the CLEAR option of the SAVE command to erase the previous report definition.

Figure 2-10. Modifying OBEY Commands



```
>> OBEY ORDFILE;
>> SET SESSION AUTOWORK ON;
   .
   .
>> SELECT O.ORDERNUM, ORDER_DATE, CUSTNUM,
   .
   .
+> UNIT_PRICE*QTY_ORDERED HEADING "TOTAL PRICE";
S> FC BREAK TITLE
S> BREAK TITLE O.ORDERNUM (SKIP 1, "CUSTOMER ", CUSTNUM,
  . .                                     R"Cust. No.", CUSTNUM,
S> BREAK TITLE O.ORDERNUM (SKIP 1, "Cust. No. ", CUSTNUM,
  . .
+> TAB 30, "Order Date ", ORDER_DATE, SKIP 1)
                          ;
  . .
+> TAB 30, "Order Date ", ORDER_DATE, SKIP 1)
                          ;
S> TITLE "Order Summary";
S> SET LAYOUT RIGHT_MARGIN 60;
S> SAVE LIST_COUNT TO ORDFILE CLEAR;
S> SAVE LAYOUT * TO ORDFILE;
S> SAVE STYLE * TO ORDFILE;
S> SAVE REPORT * TO ORDFILE;
```

Creating an Command File With a Text Editor

You can use a text editor to create a command file that contains all of the commands needed to define a report. The SELECT command must precede the report formatting commands. To prevent any SELECT output from printing or displaying before the report formatting commands execute, enter the following command before the SELECT command:

```
>> SET LIST_COUNT 0;
```

You can use the editor not only to create a report definition, but also to modify a file you create with the SAVE command. For example, if you decide you are going to run a report repeatedly, you might want to add more commands to the file.

For more information about providing data selection criteria when you execute a command file, see Using Parameters With SELECT Commands on page 3-23.

Creating an Command File From a Log File

You can use a log file as a command file. When you start logging, specify that you want only commands written to the log file. For example, the following command specifies only commands are to be logged to a file named REPDEFN:

```
>> LOG REPDEFN COMMANDS CLEAR;
```

Next, enter commands in the same order as shown in Saving Report Commands in an Command file on page 2-16. Set the LIST_COUNT option, enter a SELECT command, and then enter the report formatting commands. (With this method, you do not need to save the commands by using the SAVE command.)

To stop logging commands, enter the following:

```
>> LOG ;
```

You can now use the log file as a command file. With this method, every command you enter is written to the log file, including FC and LIST. You might want to use a text editor to examine the file and make any necessary changes before using the file.

Using the Report Writer to Produce Text From Data

You can use the report writer to produce text from data, thereby creating a command file or an input file for other applications.

The following example illustrates how you can retrieve statistics from one SQL catalog table and produce several edit files. These edit files are then used as command files to transfer the statistics to a different catalog for analysis.

In this example, RPXSTATS is a set of SQLCI reports that can obtain the statistics, INVOKE directives, and FILEINFO commands for either a single table or for all tables and the associated indexes (if any).

RUNXREPS is the command file that evokes RPXSTATS.

Before you run RUNXREPS, you must assign a value to the parameter named ?TBOL to specify the data that you want to retrieve, as follows:

```
SET PARAM ?TBOL value
```

- To retrieve data for a single partition of a single table, *value* is a fully qualified file name as follows:

  ```
  SET PARAM ?TBOL "%$DATA.SUBVOL.ATABLE%";
  ```

- To retrieve data for all partitions of a single table, *value* is a TABLENAME without qualification as follows:

  ```
  SET PARAM ?TBOL "%ATABLE%";
  ```

- To retrieve data for all tables in the current catalog, *value* is the wild-card character % as follows:

  ```
  SET PARAM ?TBOL "%";
  ```

RPXSTATS produces the following three files:

- SQLUPDS, for the statistics
- FILEINF, an intermediate command file that creates the SQLFILES file
- SQLFILES, a file for the INVOKE directives and FILEINFO commands

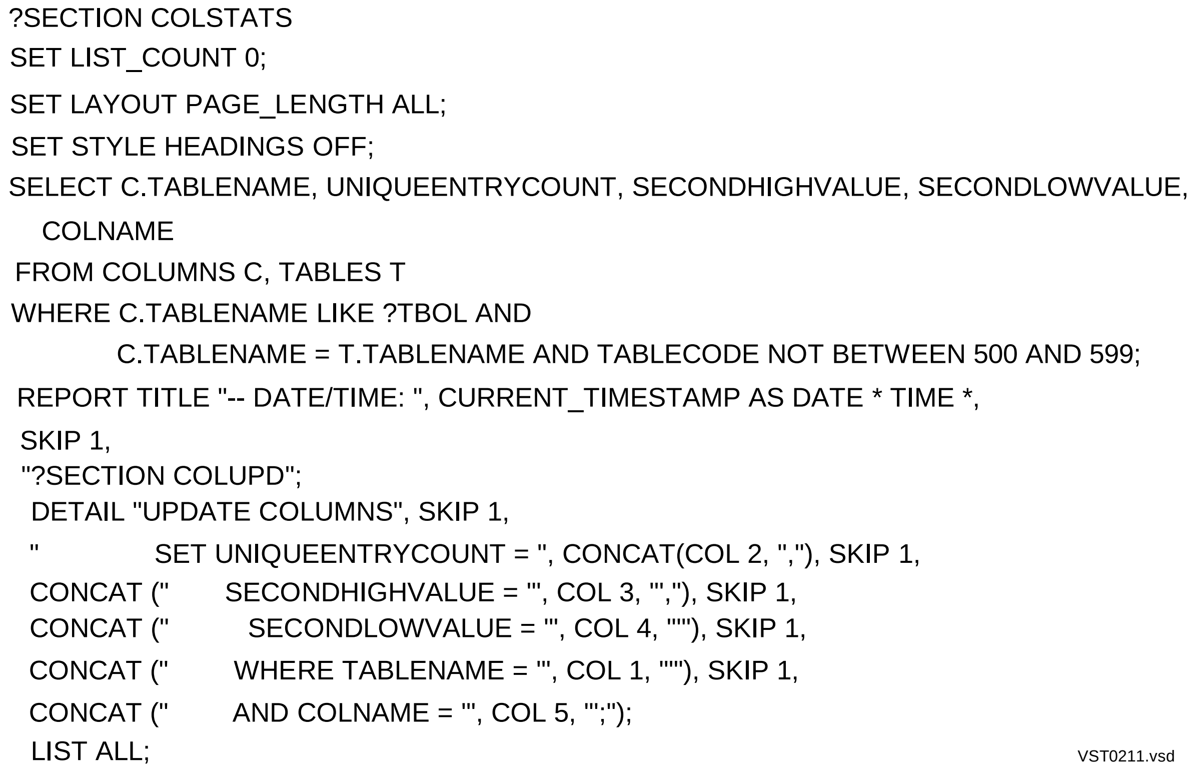
The RUNXREPS command file purges any pre-existing data from these files using the CLEAR option. Therefore, you need to rename SQLFILES and SQLUPDS to save them if you want to evoke the RUNXREPS command file more than once.

The following example illustrates the RUNXREPS command file. Note that the file named FILEINF created by the OUT_REPORT command is then immediately invoked by the OBEY FILEINF command.

```
OUT_REPORT SQLUPDS CLEAR;
OBEY RPXSTATS(COLSTATS, BASETAB, INDEXNFILES);
OUT_REPORT FILEINF CLEAR;
OBEY RPXSTATS(INVOCATION,INFOFILE);
OUT_REPORT ;
OUT SQLFILES CLEAR;
OBEY FILEINF (INVOCATION,FILEDATA);
OUT ;
```

Figure 2-11, Figure 2-12, and Figure 2-13 illustrate the RPXSTATS command file.

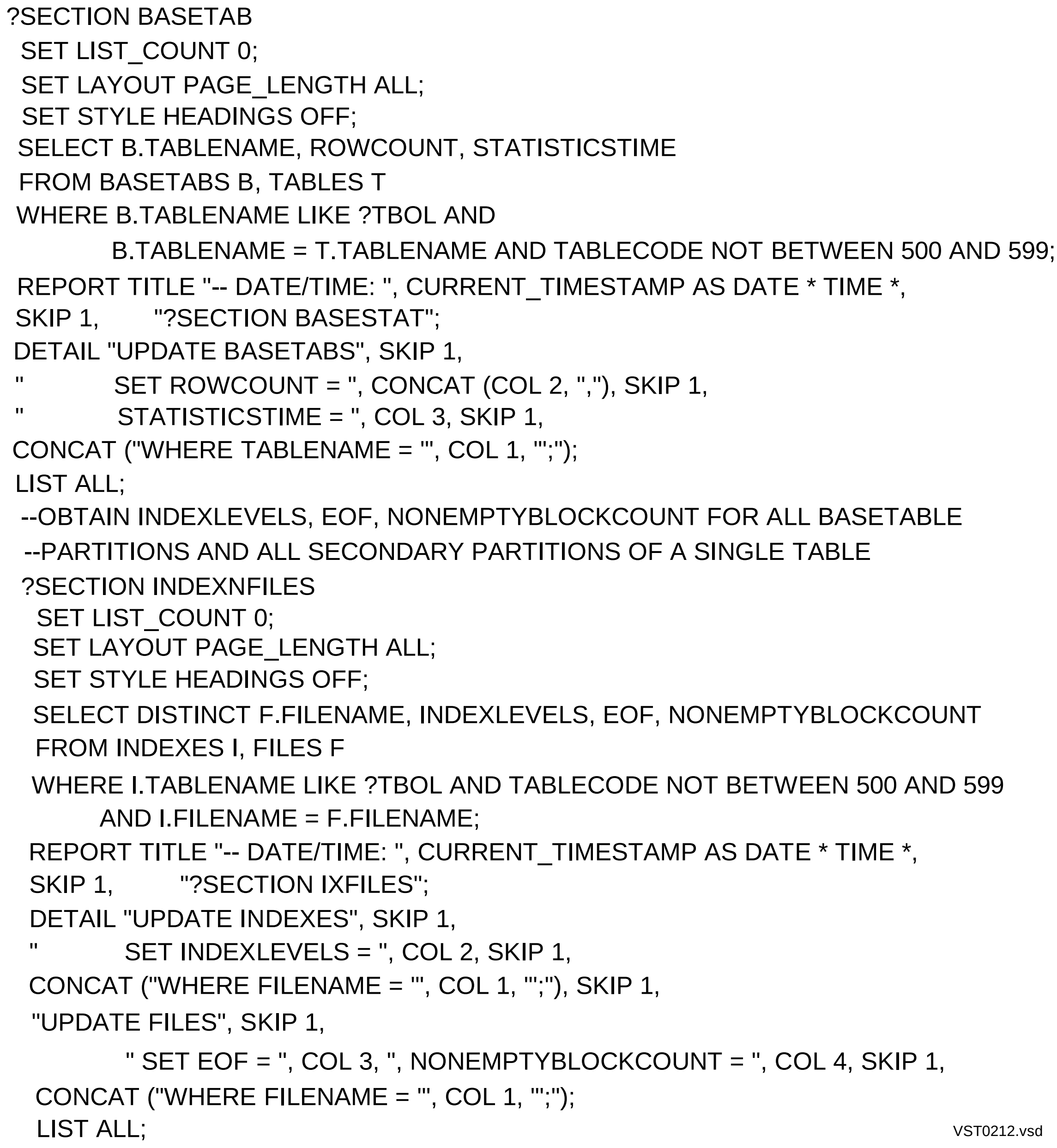
Figure 2-11. Example of Using the Report Writer to Produce Text From Data



```
?SECTION COLSTATS
SET LIST_COUNT 0;
SET LAYOUT PAGE_LENGTH ALL;
SET STYLE HEADINGS OFF;
SELECT C.TABLENAME, UNIQUEENTRYCOUNT, SECONDHIGHVALUE, SECONDLOWVALUE,
   COLNAME
 FROM COLUMNS C, TABLES T
 WHERE C.TABLENAME LIKE ?TBOL AND
          C.TABLENAME = T.TABLENAME AND TABLECODE NOT BETWEEN 500 AND 599;
 REPORT TITLE "-- DATE/TIME: ", CURRENT_TIMESTAMP AS DATE * TIME *,
 SKIP 1,
 "?SECTION COLUPD";
  DETAIL "UPDATE COLUMNS", SKIP 1,
  "           SET UNIQUEENTRYCOUNT = ", CONCAT(COL 2, ","), SKIP 1,
  CONCAT ("      SECONDHIGHVALUE = '", COL 3, "',"), SKIP 1,
  CONCAT ("        SECONDLOWVALUE = '", COL 4, "'"), SKIP 1,
  CONCAT ("       WHERE TABLENAME = '", COL 1, "'"), SKIP 1,
  CONCAT ("       AND COLNAME = '", COL 5, "';");
  LIST ALL;
```

VST0211.vsd

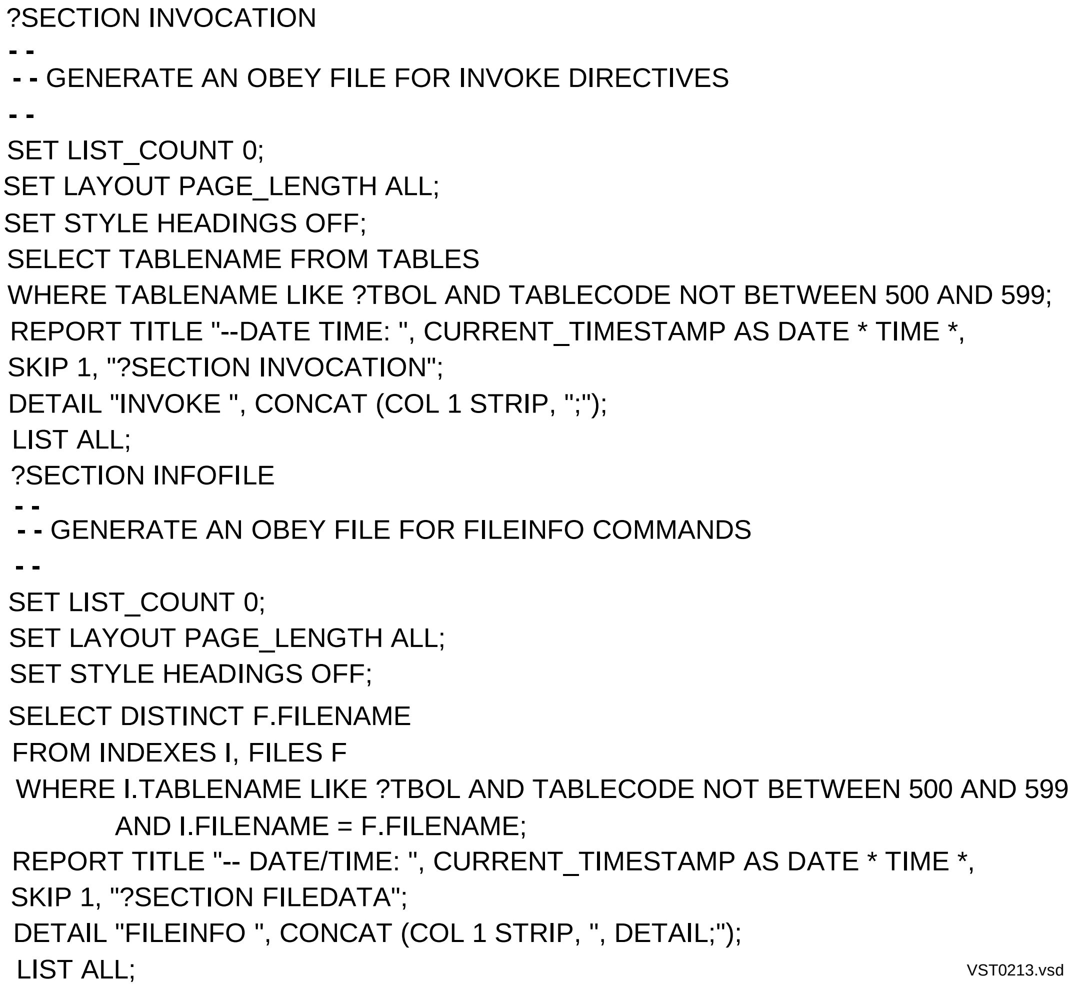
Figure 2-12. Example of Using the Report Writer to Produce Text From Data



```
?SECTION BASETAB
 SET LIST_COUNT 0;
 SET LAYOUT PAGE_LENGTH ALL;
 SET STYLE HEADINGS OFF;
 SELECT B.TABLENAME, ROWCOUNT, STATISTICSTIME
 FROM BASETABS B, TABLES T
 WHERE B.TABLENAME LIKE ?TBOL AND
            B.TABLENAME = T.TABLENAME AND TABLECODE NOT BETWEEN 500 AND 599;
 REPORT TITLE "-- DATE/TIME: ", CURRENT_TIMESTAMP AS DATE * TIME *,
 SKIP 1,       "?SECTION BASESTAT";
 DETAIL "UPDATE BASETABS", SKIP 1,
 "            SET ROWCOUNT = ", CONCAT (COL 2, ","), SKIP 1,
 "            STATISTICSTIME = ", COL 3, SKIP 1,
 CONCAT ("WHERE TABLENAME = '", COL 1, "';");
 LIST ALL;
 --OBTAIN INDEXLEVELS, EOF, NONEMPTYBLOCKCOUNT FOR ALL BASETABLE
 --PARTITIONS AND ALL SECONDARY PARTITIONS OF A SINGLE TABLE
 ?SECTION INDEXNFILES
  SET LIST_COUNT 0;
  SET LAYOUT PAGE_LENGTH ALL;
  SET STYLE HEADINGS OFF;
  SELECT DISTINCT F.FILENAME, INDEXLEVELS, EOF, NONEMPTYBLOCKCOUNT
  FROM INDEXES I, FILES F
  WHERE I.TABLENAME LIKE ?TBOL AND TABLECODE NOT BETWEEN 500 AND 599
          AND I.FILENAME = F.FILENAME;
  REPORT TITLE "-- DATE/TIME: ", CURRENT_TIMESTAMP AS DATE * TIME *,
  SKIP 1,        "?SECTION IXFILES";
  DETAIL "UPDATE INDEXES", SKIP 1,
  "           SET INDEXLEVELS = ", COL 2, SKIP 1,
  CONCAT ("WHERE FILENAME = '", COL 1, "';"), SKIP 1,
  "UPDATE FILES", SKIP 1,
             " SET EOF = ", COL 3, ", NONEMPTYBLOCKCOUNT = ", COL 4, SKIP 1,
  CONCAT ("WHERE FILENAME = '", COL 1, "';");
  LIST ALL;
```

VST0212.vsd

Figure 2-13. Example of Using the Report Writer to Produce Text From Data



```
?SECTION INVOCATION
--
-- GENERATE AN OBEY FILE FOR INVOKE DIRECTIVES
--
SET LIST_COUNT 0;
SET LAYOUT PAGE_LENGTH ALL;
SET STYLE HEADINGS OFF;
SELECT TABLENAME FROM TABLES
WHERE TABLENAME LIKE ?TBOL AND TABLECODE NOT BETWEEN 500 AND 599;
REPORT TITLE "--DATE TIME: ", CURRENT_TIMESTAMP AS DATE * TIME *,
SKIP 1, "?SECTION INVOCATION";
DETAIL "INVOKE ", CONCAT (COL 1 STRIP, ";");
LIST ALL;
?SECTION INFOFILE
--
-- GENERATE AN OBEY FILE FOR FILEINFO COMMANDS
--
SET LIST_COUNT 0;
SET LAYOUT PAGE_LENGTH ALL;
SET STYLE HEADINGS OFF;
SELECT DISTINCT F.FILENAME
FROM INDEXES I, FILES F
WHERE I.TABLENAME LIKE ?TBOL AND TABLECODE NOT BETWEEN 500 AND 599
          AND I.FILENAME = F.FILENAME;
REPORT TITLE "-- DATE/TIME: ", CURRENT_TIMESTAMP AS DATE * TIME *,
SKIP 1, "?SECTION FILEDATA";
DETAIL "FILEINFO ", CONCAT (COL 1 STRIP, ", DETAIL;");
LIST ALL;
```

3 Selecting Data for a Report

Before you can define a report, you must select the data by entering a SELECT command. The report formatting commands you specify refer to columns (or print items) specified in the select list of the SELECT command. This section provides examples of the following techniques for selecting data:

- Developing a query
  - Determining which tables contain the data
  - Selecting the column values you need
  - Setting criteria for selecting rows
  - Sorting the data
- Computing values based on column values from groups of rows: averages, sums, minimums, maximums, and counts
- Selecting rows with a distinct value in a column
- Using expressions to calculate report item values
- Using parameters to provide different selection criteria each time you produce a report
- Preparing SELECT commands to be executed more than once
- Creating views to simplify the specification of queries, save time, and make data access and reporting easier for nontechnical users of the database

The SELECT command selects the data that will be formatted by the report formatting commands or will be displayed or printed in the default report format. For a complete description of the SELECT command and all language elements of SQL, see the *SQL/MP Reference Manual*. For a complete description of developing queries, see the *SQL/MP Query Guide*.

Developing a Query

Before you develop a query to select data, you should plan the content of the report. Consider the following questions:

- What columns of information do you need and how should the columns be arranged?
- In what order should the rows appear?
- What subtotals and totals are to be calculated?
- What information do you want to include in the titles and footings?

At this point, you do not have to consider the details of formatting the report. You can enhance the format after you have generated the basic content.

After you know what information you want, you can compose a SELECT command to retrieve the data. The SELECT command must retrieve all the data you need, including information in columns, titles, headings, and footings. In the following pages, one approach to composing a SELECT command is presented.

Figure 3-1 illustrates the general content of a report on suppliers and the parts they supply. The italicized words *date* and *time*, and the numbers below the column headings are notes about what data is needed to produce the report.

Locating the Data

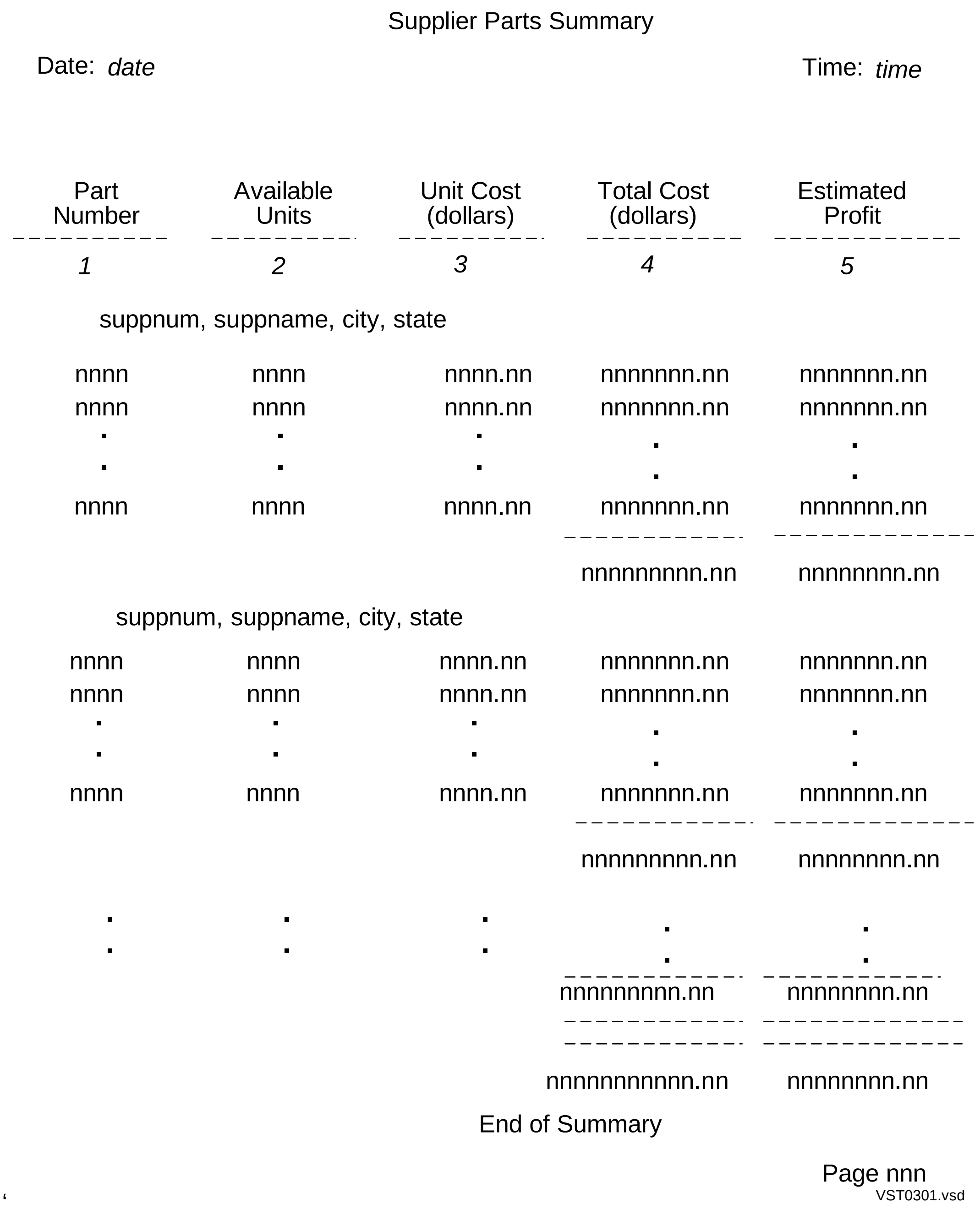
First, you must determine which database tables contain the data. The example in Figure 3-1 uses data from the SUPPLIER, PARTSUPP, and PARTS tables. The report contains the following data in the numbered output line items:

```
1 PARTS.PARTNUM
2 PARTS.QTY_AVAILABLE
3 PARTSUPP.PARTCOST
4 PARTSUPP.PARTCOST * PARTS.QTY_AVAILABLE
5 PARTS.QTY_AVAILABLE * (PARTS.PRICE - PARTSUPP.PARTCOST)
```

Each column name in this list is qualified by the name of the table that contains the column. You can use the INVOKE command to determine the names of columns in a table.

The report title includes the date and the time when the report is produced. These values are to be provided through the CURRENT_TIMESTAMP function.

Figure 3-1. Plan for Report Content



```
                             Supplier Parts Summary

  Date:  date                                                Time:  time


    Part        Available       Unit Cost      Total Cost      Estimated
   Number         Units         (dollars)      (dollars)        Profit
 ----------    ----------      ----------     ----------     ------------
     1             2               3              4               5

       suppnum, suppname, city, state

    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
     .            .               .              .               .
     .            .               .              .               .
    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
                                           ------------    -------------
                                           nnnnnnnnn.nn     nnnnnnnn.nn
         suppnum, suppname, city, state
    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
     .            .               .              .               .
     .            .               .              .               .
    nnnn         nnnn          nnnn.nn      nnnnnnn.nn      nnnnnnn.nn
                                            ------------   -------------
                                           nnnnnnnnn.nn     nnnnnnnn.nn

     .            .               .              .               .
     .            .               .              .               .
                                           ------------   ------------
                                          nnnnnnnnn.nn    nnnnnnnn.nn
                                           ------------   -------------
                                           ------------   -------------

                                         nnnnnnnnnnn.nn   nnnnnnnn.nn
                              End of Summary

                                                      Page nnn
                                                         VST0301.vsd
```

The detail lines are to be ordered by supplier with a break on SUPPLIER.SUPPNUM, SUPPLIER.SUPPNAME, SUPPLIER.CITY, and SUPPLIER.STATE. These values are printed in the break title. No SELECT data is needed for the page footing or report footing.

The FROM clause of the SELECT command specifies the names of all tables from which the SELECT command retrieves data. For example, the following command retrieves values from the PARTS table. An asterisk in the select list retrieves a value for each column of the table. The table you specify must be on the current default subvolume, or you must qualify the table name in the FROM clause, as shown:

```
>> SELECT * FROM SALES.PARTS;
```

To select the data needed for the report in Figure 3-1, you must join rows of the three tables. First, consider the joining of rows from the PARTS and PARTSUPP tables that have the same PARTNUM value:

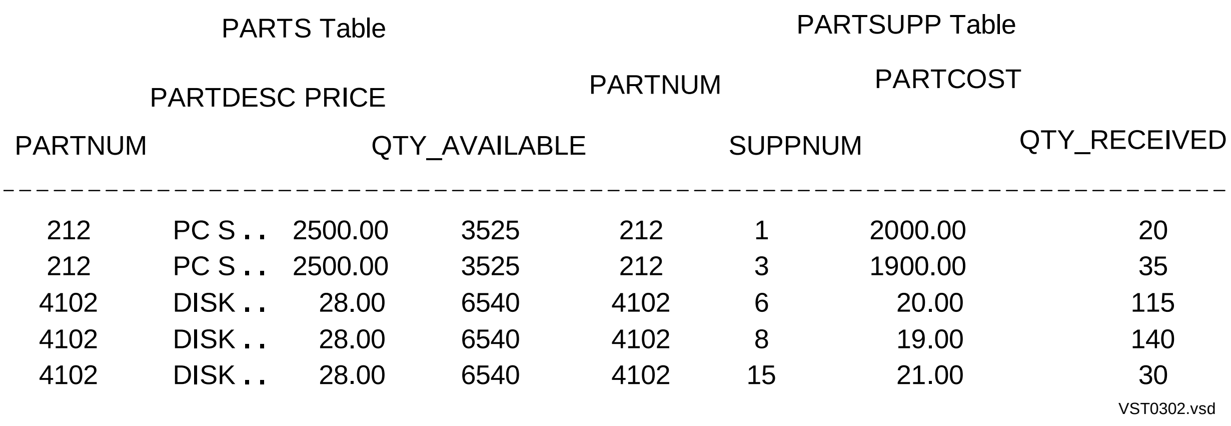
```
>> VOLUME INVENT;
>> SELECT *
+> FROM SALES.PARTS, PARTSUPP
+> WHERE PARTS.PARTNUM = PARTSUPP.PARTNUM;
```

The VOLUME command makes INVENT the current default subvolume. If you omit the VOLUME command, the PARTSUPP table name should be qualified as INVENT.PARTSUPP.

The WHERE clause specifies how the rows are to be joined; in this example, rows with the same part number in each table are joined. The expression in the WHERE clause is called a comparison predicate. If you omit the WHERE clause, each row of the PARTS table is joined with each row of the PARTSUPP table. Because this approach results in an inefficient query; it is recommended that you include a WHERE clause with a predicate to indicate how to join the tables in the FROM clause.

Figure 3-2 illustrates the result of joining these tables. Suppose PARTS contains two rows and PARTSUPP contains five rows.

Figure 3-2. Sample Rows from Joined Tables



PARTS Table				PARTSUPP Table			
PARTNUM	PARTDESC	PRICE	QTY_AVAILABLE	PARTNUM	SUPPNUM	PARTCOST	QTY_RECEIVED
212	PC S . .	2500.00	3525	212	1	2000.00	20
212	PC S . .	2500.00	3525	212	3	1900.00	35
4102	DISK . .	28.00	6540	4102	6	20.00	115
4102	DISK . .	28.00	6540	4102	8	19.00	140
4102	DISK . .	28.00	6540	4102	15	21.00	30

VST0302.vsd

Because the part number column has the same name in the PARTS table and the PARTSUPP table, you must qualify the column name with the table name. You can use the table name, which is the implicit correlation name: for example, PARTS.PARTNUM. Alternatively, you can define an explicit correlation name in the

FROM clause to use as an abbreviation for qualifying column names. For example, the following command defines P and PS as explicit correlation names:

```
>> SELECT *
+> FROM SALES.PARTS P, PARTSUPP PS
+> WHERE P.PARTNUM = PS.PARTNUM;
```

Correlation names are required in some types of subqueries. These subqueries are discussed in Using Subqueries on page 3-28.

To join the SUPPLIER table to the other two tables, you can join the rows of the result table shown in Figure 3-2 to rows of the SUPPLIER table that have the same SUPPNUM value. The way to express this relation is as follows:

```
>> SELECT *
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+> AND PS.SUPPNUM = S.SUPPNUM;
```

The SUPPLIER table is also on the INVENT subvolume, so the name does not need to be qualified.

Using DEFINEs

For queries that you intend to execute many times, you should use DEFINE names in the FROM clause. For example, you can use the DEFINE name =PARTS in the FROM clause. Assume the following command is in the command file named FINDPART:

```
SELECT PARTNUM, PARTDESC, QTY_AVAILABLE FROM =PARTS;
```

Before you execute the query, you specify the actual file name that corresponds to =PARTS by using an ADD DEFINE command such as the following:

```
>> ADD DEFINE =PARTS, FILE $WAREHS1.SALES.PARTS;
>> OBEY FINDPART;
```

The query searches the PARTS table on subvolume $WAREHS1.SALES.

DEFINEs allow you to use more readable names in SELECT commands, and they provide location independence. For more information about using DEFINEs, see the *SQL/MP Reference Manual*.

Selecting Columns of Source Data

You can specify the content of the detail lines of a report in the SELECT command select list. If you want to retrieve values for use in the report but do not want the values to appear in the detail lines, you specify both a select list and a DETAIL command.

The select list must specify each column of data used in the report. For example, a column value you print in a title or footing does not have to appear in the detail line, but it must be retrieved in the select list.

You select specific columns by specifying the column names. The select list for the supplier parts summary consists of the column names in boldface type that appear in the following example:

```
>> SELECT S.SUPPNUM,
+>        SUPPNAME,
+>        CITY,
+>        STATE,
+>        P.PARTNUM,
+>        QTY_AVAILABLE,
+>        PARTCOST,
+>        PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+>    AND PS.SUPPNUM = S.SUPPNUM;
```

These are the only columns you need from the result of joining the three tables. You must qualify SUPPNUM and PARTNUM because they appear twice in the initially joined result table.

If you intend to use all the columns from a table, you can specify an asterisk (*) or an asterisk qualified by the table name instead of specifying all the column names as shown:

```
>> SELECT * FROM PARTS;
```

or

```
>> SELECT EMPLOYEE.*, JOBDESC
+> FROM EMPLOYEE, JOB
+> WHERE EMPLOYEE.JOBCODE = JOB.JOBCODE;
```

If you specify a DETAIL command, it determines the order of columns in report lines. The order of the select list is important only when you do not specify a DETAIL command.

By using the asterisk, you limit the options available to the query optimizer. You should specify exactly the columns you need to provide more choices to the optimizer for selecting a query plan.

Setting Criteria for Selecting Data

In addition to specifying how tables are joined, the WHERE clause specifies conditions for selecting particular rows of data from the result table. The result table is the logical table specified by the FROM clause, select list, and the WHERE clause itself, in the case of joined tables.

For example, the following command selects only rows describing parts with at least 50 units available:

```
>> SELECT *
+> FROM SALES.PARTS
+> WHERE QTY_AVAILABLE >= 50;
```

In the next example, the WHERE clause specifies conditions for selecting rows and specifies the method for joining three tables. Only information about local suppliers is selected; that is, suppliers in areas with postal codes between 95400 and 95500.

```
>> VOLUME INVENT;
>> SELECT S.SUPPNUM, SUPPNAME, P.PARTNUM,
+> QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+> AND PS.SUPPNUM = S.SUPPNUM
+> AND POSTCODE BETWEEN "95400" AND "95500";
```

Columns you refer to in the WHERE clause do not have to appear in the select list, but they must be columns from one of the tables in the FROM clause.

The search condition you specify in a WHERE clause consists of predicates connected by the Boolean operators NOT, AND, and OR.

Comparison predicates and quantified predicates can include the following comparison operators:

=	Equal
<>	Not equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Table 3-1 summarizes the types of predicates you can use to express search conditions.

Table 3-1. Search Condition Predicates (page 1 of 2)

Predicate	Purpose
Comparison	Compares values of two expressions, two sets of expressions, or the value of an expression and a single value resulting from a subquery; for example: PARTNUM = 244 (Part number must equal 244.)
Quantified	Compares the value of an expression to all or any of the values of a single column result from a subquery; for example: PRICE < ALL (SELECT UNIT_PRICE FROM ODETAIL WHERE PARTNUM = 5505) (Price must be less than the unit price in all orders for part number 5505.)
BETWEEN	Determines if a value is in the range of two other values, or if a set of values is in the range of two other sets of values; for example: PARTNUM BETWEEN 100 AND 500 (Part number must be greater than or equal to 100 and less than or equal to 500.)

Table 3-1. Search Condition Predicates (page 2 of 2)

Predicate	Purpose
IN	Determines if a value is equal to any of the values in a list or in a collection of values; for example: PARTNUM IN (100, 120, 150) (Part number must be 100, 120, or 150.)
LIKE	Searches for strings to match a pattern that can contain wild-card characters percent (%) and underscore (_) ; for example: PARTDESC LIKE "DISK_T%" (Part description contains DISK followed by exactly one character, followed by T, followed by zero or more characters.)

EXISTS	Determines whether any rows satisfy the conditions of a subquery; for example:
	SELECT * FROM ORDERS O
	WHERE EXISTS (SELECT *
	FROM ODETAIL OD
	WHERE OD.ORDERNUM = O.ORDERNUM
	AND PARTNUM = 244);
	(Orders must include part number 244.)

NULL	Determines whether a column contains a null value; for example: UNIT_PRICE IS NULL (Unit price must be null.)

The following examples illustrate the effect of each predicate when included in the following SELECT command:

```
>> VOLUME INVENT;
>> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+> AND PS.SUPPNUM = S.SUPPNUM
+> ( Substitute predicate example from following text.)
```

- The following comparison predicate specifies only suppliers identified by numbers less than or equal to 400:

```
+> AND S.SUPPNUM <= 400;
```

- Quantified predicates are useful for selecting rows based on their relation to all or any rows selected by a separate search condition. For example, you can select PARTSUPP table rows that contain a part cost value greater than the cost of every part in the table supplied by supplier number 6 as shown:

```
+> AND PARTCOST > ALL ( SELECT PARTCOST FROM PARTSUPP
+> WHERE SUPPNUM = 6 ) ;
```

The subquery selects the part cost from each row describing parts supplied by supplier number 6. The greatest part cost is $1100.00. The main query selects rows with a part cost value greater than all values selected by the subquery; that is, rows with a part cost greater than $1100.00.

The next quantified predicate selects rows with a part cost equal to any of the parts supplied by supplier number 1:

```
+> AND PARTCOST = ANY ( SELECT PARTCOST FROM PARTSUPP
+> WHERE SUPPNUM = 1 ) ;
```

For more examples of quantified predicates as well as examples of EXISTS predicates, see the *SQL/MP Reference Manual*.

- The following BETWEEN predicate specifies only suppliers of parts numbered from 4000 through 6103:

  ```
  +> AND P.PARTNUM BETWEEN 4000 AND 6103;
  ```

- The IN predicate can locate a specific list of values, or all values except those in a specific list. This predicate allows you to express multiple conditions efficiently.

  ```
  +> AND P.PARTNUM IN (1, 3, 15);
  +> AND P.PARTNUM NOT IN (4, 7, 21, 45);
  ```

You can also use NOT with the LIKE, EXISTS, and NULL predicates.

You can use a subquery to select a list of values. This technique allows you to compare information from another table without joining tables in the main query. The subquery selects the rows you want to compare.

In the next example, the subquery (shown in boldface type) selects from the PARTLOC table part numbers for parts with greater than 500 units on hand in a single location. The main query selects the part description and price for these parts from the PARTS table.

```
>> SELECT PARTNUM, PARTDESC, PRICE
+> FROM SALES.PARTS
+> WHERE PARTNUM IN (SELECT PARTNUM
+>                  FROM INVENT.PARTLOC
+>                   WHERE QTY_ON_HAND > 500);
S> LIST NEXT 2;

PARTNUM     PARTDESC        PRICE

------- ----------------- -----------

2001    GRAPHIC PRINTER,M1  1100.00
2403    DAISY PRINTER,T2     650.00
S>
```

Note that a subquery select list can contain only one element. The element can be an expression or column name.

To confirm the result of the previous query, you can select the PARTLOC information for the two parts. The QTY_ON_HAND value is greater than 500 in at least one location.

```
>> SELECT * FROM INVENT.PARTLOC
+> WHERE PARTNUM IN ( 2001, 2403 );
S> LIST ALL;

LOC_CODE PARTNUM QTY_ON_HAND

-------- ------- -----------

A10        2001   800
A88        2403   735
G88        2403   32
P10         2001     0

--- 4 row(s) selected.
```

Comparing a Set of Columns to a Set of Values

You can use comparison and BETWEEN predicates to select rows based on values from more than one column in the row when the data types of the expressions are compatible. For example, you might want to select a range of rows from a list of names by examining both the first and last names. In the EMPLOYEE table, these names are stored in separate columns.

The following query selects employee numbers and job codes for all employees whose names are between CLARK, LARRY and FOLEY, MARK:

```
>> SELECT EMPNUM, JOBCODE
+> FROM PERSNL.EMPLOYEE
+> WHERE LAST_NAME, FIRST_NAME BETWEEN "CLARK", "LARRY"
+> AND "FOLEY", "MARK";
```

The names CLARK, JUNE and FOLEY, MAVA would not be selected.

The following query selects a customer located at 2300 BROWN BLVD in FRESNO, CALIFORNIA:

```
>> SELECT CUSTNUM, CUSTNAME
+> FROM SALES.CUSTOMER
+> WHERE STREET = "2300 BROWN BLVD" AND
+> CITY = "FRESNO" AND
+> STATE = "CALIFORNIA";
```

You can simplify this query as follows:

```
>> SELECT CUSTNUM, CUSTNAME
+> FROM SALES.CUSTOMER
+> WHERE STREET, CITY, STATE =
+> "2300 BROWN BLVD", "FRESNO", "CALIFORNIA";
```

Note. To compare character data that uses collations, see “COMPARISON” in the *SQL/MP Reference Manual*

Comparing Character Values

If a column contains text (character data), you must enclose the comparison value in single or double quotation marks. You must enter the characters you want to match exactly as the characters are stored in the column. For example, to find a value that consists of uppercase letters, enter uppercase letters. You must also include the same number of spaces within the comparison string as there are spaces stored within the column value.

The following predicate will not select "PC Diamond, 60 MB" or "PC DIAMOND, 60MB".

```
+> AND PARTDESC = "PC DIAMOND, 60 MB";
```

You can use the UPSHIFT function to select without regard to whether a character value is in lowercase or uppercase. For example, the following predicate selects "PC Diamond, 60 MB" and "PC DIAMOND, 60 MB" but not "PC DIAMOND, 60MB" (because of the missing space):

```
+> AND UPSHIFT (PARTDESC) = "PC DIAMOND, 60 MB";
```

Note. TUPSHIFT is a function that upshifts single-byte characters. You cannot use the UPSHIFT function if the column contains a double-byte character set, such as Tandem Kanji or Tandem KSC5601.

The LIKE predicate lets you look for similar values by specifying only a few characters and using the following wild-card characters:

- % Percent sign indicates zero or more characters of any type are acceptable.
- _ Underscore indicates any single character is acceptable.

Character data can be stored in columns of data type CHAR, PIC X, VARCHAR, NATIONAL CHAR, and NCHAR.

PIC X columns, and CHAR, NATIONAL CHAR, and NCHAR columns defined without the VARYING clause, contain fixed-length values; every row of the table contains a value of the same length in the column. When you insert a value, the system fills the value with blanks if necessary.

VARCHAR columns and CHAR, NATIONAL CHAR, AND NCHAR columns defined with the VARYING clause contain variable-length values; values vary in length from row to row. When you insert a value, only the characters you enter are stored.

The following rules describe how the system compares character values in comparison and LIKE predicates. (BETWEEN and IN predicates follow the same rules as comparison predicates.)

- Trailing blanks are significant for fixed-length columns. For example, the value "DISK" inserted in a CHAR(6) column is "DISK".
- Only the data inserted is significant for variable-length columns. For example, the value "DISK" inserted in a VARCHAR(6) column is "DISK".
- When processing comparison predicates, the system pads the comparison value or the column value (whichever is smaller) to make the values the same length.
- When processing LIKE predicates, the system does not pad the column value with blanks.
- Unless you specify a % wild-card character in the comparison value, the condition is met only if the column value and the comparison value are the same length.

Table 3-2 provides some examples to illustrate these rules.

Table 3-2. Comparison and LIKE Predicates

Column Definition	Value Inserted	Stored Value	Predicate	Result
Fixed-length:	“5MB”	“5MB^^^”	LIKE “%MB”	Fails, last 2 characters not MB
`CHAR (6)`				
`PIC X (6)`				
`NATIONAL CHAR (3)`				
`NCHAR (6)`				
			LIKE “%MB%”	Succeeds
			= “5MB”	Succeeds
	“120^MB”	“120^MB”	LIKE “%MB”	Succeeds, last 2 characters are MB
Variable-length	“5MB”	“5MB”	LIKE “%MB”	Succeeds
`CHAR VARYING (6)`				
`VARCHAR (6)`			= “5MB”	Succeeds
`NATIONAL CHAR VARYING (3)`			LIKE “5MB”	Succeeds
			LIKE “5MB%”	Succeeds
`NCHAR VARYING (3)`			LIKE “_MB”	Succeeds
			LIKE “_MB%”	Succeeds
	“120^MB”	“120^MB”	LIKE “%MB”	Succeeds

^ indicates a space (blank character)

Note. Specifying a pattern beginning with percent (%) in a LIKE predicate can result in a scan of a complete table. You should not use this type of pattern when executing a query online unless other predicates in the query provide access paths through indexes or primary keys. If you need to use this type of pattern, execute the query in batch mode when the system has fewer demands on its resources.

When you compare columns of fixed-length character data types, the LIKE predicate might be more restrictive than a comparison predicate because the trailing blanks are significant. You should use comparison predicates to search for exact values.

If you cannot locate a value in a column of a variable-length character data type, it might be because trailing blanks were included when the value was inserted in the table. For example, a value of "5MB" will not be located by LIKE "%MB". Try including a % at the end of the comparison value.

Specifying More Than One Condition

You can use the Boolean operators NOT, AND, and OR to select data that satisfies more than one condition as shown:

```
>> SELECT PARTNUM, PARTDESC
+> FROM SALES.PARTS
+> WHERE QTY_AVAILABLE < 2500
+> AND PARTNUM BETWEEN 2000 AND 3000
+> OR PARTNUM > 6000 ;
```

The following is the order of evaluation from first to last:

Expressions within parentheses
NOT
AND
OR

The interpretation of these conditions is this: If the quantity available is less than 2500, part numbers from 2000 through 3000 are selected. Part numbers greater than 6000 are selected regardless of the quantity available.

The effect of parentheses is illustrated by the following revision of these conditions:

```
>> SELECT PARTNUM, PARTDESC
+> FROM SALES.PARTS
+> WHERE QTY_AVAILABLE < 2500
+> AND ( PARTNUM BETWEEN 2000 AND 3000
+> OR PARTNUM > 6000 ) ;
```

The interpretation of these conditions is this: If the quantity available is less than 2500, part numbers from 2000 through 3000 or greater than 6000 are selected.

If you want NOT to apply to more than one predicate, you must enclose the predicates in parentheses as shown:

```
WHERE NOT (PARTNUM BETWEEN 2000 AND 3000
OR PARTNUM > 6000)
```

Rows with part numbers between 2000 and 3000 or part numbers greater than 6000 are not selected. All other rows are selected.

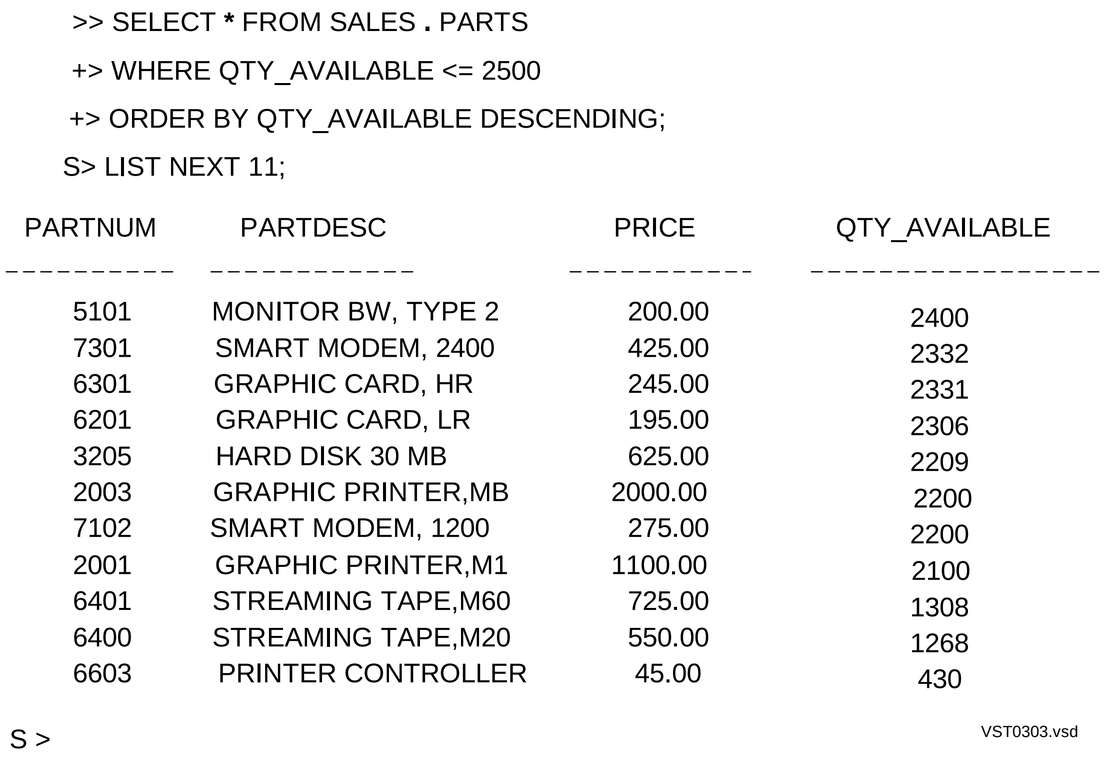
Consider these points:

- The AND that appears in a BETWEEN predicate does not connect two predicates or search conditions.
- You can specify NOT BETWEEN, NOT IN, and NOT LIKE, but you cannot specify NOT =; you must use <> to indicate not equal.

Sorting the Data

The ORDER BY clause of the SELECT command determines the order in which rows appear in the report. For example, the rows selected by the query in Figure 3-3 are displayed in descending order by quantity available.

Figure 3-3. Example of Rows Sorted by One Column



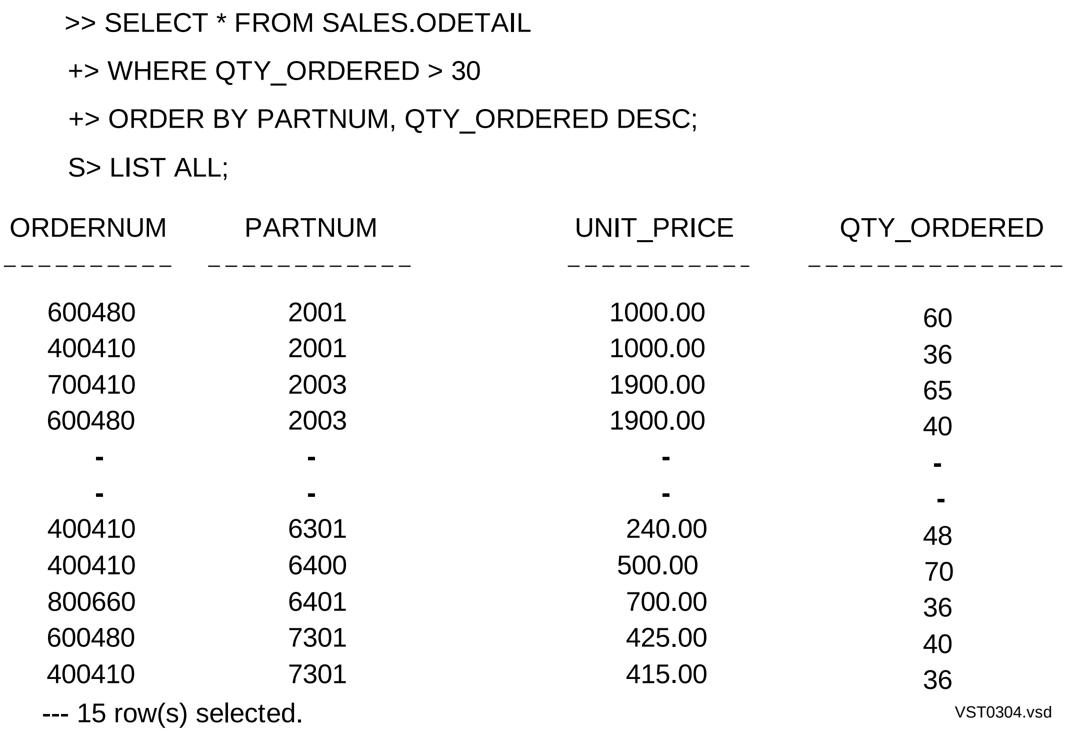
```
      >> SELECT * FROM SALES . PARTS
      +> WHERE QTY_AVAILABLE <= 2500
      +> ORDER BY QTY_AVAILABLE DESCENDING;
     S> LIST NEXT 11;

 PARTNUM      PARTDESC                PRICE       QTY_AVAILABLE
----------  ------------           -----------   -----------------

     5101    MONITOR BW, TYPE 2       200.00          2400
     7301    SMART MODEM, 2400        425.00          2332
     6301    GRAPHIC CARD, HR         245.00          2331
     6201    GRAPHIC CARD, LR         195.00          2306
     3205    HARD DISK 30 MB          625.00          2209
     2003    GRAPHIC PRINTER,MB      2000.00          2200
     7102    SMART MODEM, 1200        275.00          2200
     2001    GRAPHIC PRINTER,M1      1100.00          2100
     6401    STREAMING TAPE,M60       725.00          1308
     6400    STREAMING TAPE,M20       550.00          1268
     6603    PRINTER CONTROLLER        45.00           430

 S >
```

VST0303.vsd

You can specify ascending or descending order, and you can specify more than one column as the basis for sorting. For example, the query in Figure 3-4 selects entries in which the quantity of parts ordered is greater than 30. The entries are arranged in ascending order by part number and descending order by quantity ordered.

Note. If you specify more than one column as a basis for sorting, all of the specified columns should contain data from the same character set.

Figure 3-4. Example of Rows Sorted by Two Columns



```
>> SELECT * FROM SALES.ODETAIL
+> WHERE QTY_ORDERED > 30
+> ORDER BY PARTNUM, QTY_ORDERED DESC;
S> LIST ALL;

ORDERNUM      PARTNUM              UNIT_PRICE      QTY_ORDERED
----------   ------------          -----------    ---------------

  600480        2001                 1000.00           60
  400410        2001                 1000.00           36
  700410        2003                 1900.00           65
  600480        2003                 1900.00           40
     -            -                     -               -
     -            -                     -               -
  400410        6301                  240.00           48
  400410        6400                 500.00            70
  800660        6401                  700.00           36
  600480        7301                  425.00           40
  400410        7301                  415.00           36
--- 15 row(s) selected.
```

VST0304.vsd

The ORDER BY clause is required when you want to specify break points in a report. For more information, see Calculating Subtotals on page 4-51.

Grouping Data for Calculations

You can use aggregate functions to combine information from groups of rows and to calculate values such as averages. Each group of rows results in one detail line of the report.

Computing a Sum or Average

A typical application for grouping rows is to compute a sum or average. For example, the following query groups rows of the ODETAIL table by PARTNUM and computes the sum of the quantity ordered of each part:

```
>> SELECT PARTNUM, SUM (QTY_ORDERED)
+> FROM SALES.ODETAIL
+> GROUP BY PARTNUM
+> ORDER BY PARTNUM;
S> LIST NEXT 2;
PARTNUM (EXPR)
------- -------------------
212 20
244 47
```

The GROUP BY clause determines the rows to which the function is applied. Each group of rows with the same part number is processed to determine a sum.

You can use the following query to confirm which rows were grouped for parts 212 and 244 in the previous query:

```
>> SELECT PARTNUM, QTY_ORDERED
+> FROM SALES.ODETAIL
+> ORDER BY PARTNUM;
S> LIST N 8;
PARTNUM QTY_ORDERED
------- -----------
212 12
212 8
244 3
244 4
244 8
244 20
244 6
244 6
```

Consider the following points when using a GROUP BY clause or aggregate functions:

- The select list can include only columns specified in the GROUP BY clause (grouping columns), or the result of a function applied to a column. For example, if you try to include UNIT_PRICE in the select list of the first of the preceding two queries, an error message appears.

- If you omit the GROUP BY clause from a SELECT command that includes a SUM function, the sum of all retrieved rows is calculated; the group consists of the entire result table. In this case, you must also omit PARTNUM from the select list because it is no longer a grouping column.

- You do not have to specify an ORDER BY clause when you are grouping rows. You need the ORDER BY clause only if you want the rows that result from the grouping to be arranged in a specific order. In the first of the preceding two queries, you could specify ORDER BY PARTNUM to arrange the rows by part number, or you could specify ORDER BY 2 to arrange the rows by the total quantity ordered. Because SUM (QTY_ORDERED) does not have a name, you specify 2 to indicate the second column in the select list.

In the second query, you use the ORDER BY clause to display all rows with the same part number together. You cannot include a GROUP BY clause in this case because you want to display each row of the group to see the values of QTY_ORDERED.

- In summary, use the ORDER BY clause to arrange rows in sequence; use the GROUP BY clause to combine values and create one row from a group of rows.
- If you want to include the part description in the first query, you can join the ODETAIL and PARTS tables. Because rows in the result table for a specific part number all contain the same part description, you can include the part description as a grouping column without changing the result. Add the column name PARTDESC to the select list and to the GROUP BY clause. The same rows will be grouped. You must qualify the PARTNUM column or the reference will be ambiguous.

```
>> SELECT O.PARTNUM, PARTDESC, SUM (QTY_ORDERED)
+> FROM SALES.ODETAIL O, SALES.PARTS P
+> WHERE O.PARTNUM = P.PARTNUM
+> GROUP BY O.PARTNUM, PARTDESC;
S> LIST NEXT 2;
PARTNUM PARTDESC (EXPR)
------- ------------------ -------------------
212 PC SILVER, 20 MB 20
244 PC GOLD, 30 MB 47
```

You can also include PRICE and QTY_AVAILABLE in the select list and GROUP BY clause without changing the formation of the groups. However, if you include a column from the ODETAIL table, the number of groups will increase because ORDERNUM, UNIT_PRICE, and QTY_ORDERED have different values for the same part number.

The same type of query computes an average. The next query calculates both the average price and total quantity ordered of each part. The default heading for an expression is (EXPR).

```
>> SELECT PARTNUM, AVG (UNIT_PRICE), SUM(QTY_ORDERED)
+> FROM SALES.ODETAIL
+> ORDER BY PARTNUM
+> GROUP BY PARTNUM;
S> LIST N 3;
PARTNUM (EXPR) (EXPR)
------- -------------------- --------------------
212 2475.00 20
244 3216.00 47
255 3900.00 38
S>
```

If you omit PARTNUM from the select list, and you omit the GROUP BY clause, the query computes an average unit price and sums the quantity ordered using all rows selected. The result is one row of output.

Counting Rows

You can count all rows or distinct rows in a group. Suppose you want to know the names of all customers who have at least two current orders placed.

One way to do this is to join the CUSTOMER and ORDERS tables, group the rows by customer number and customer name, and display the count of all current orders. By examining the report, you can locate groups that have at least two orders: for example, BROWN MEDICAL CO.

```
>> SELECT C.CUSTNUM, CUSTNAME, COUNT (DISTINCT ORDERNUM)
+> FROM SALES.CUSTOMER C, SALES.ORDERS O
+> WHERE C.CUSTNUM = O.CUSTNUM
+> GROUP BY C.CUSTNUM, CUSTNAME;
S> LIST N 3;
CUSTNUM CUSTNAME (EXPR)
------- ------------------ -------------------
21 CENTRAL UNIVERSITY 1
123 BROWN MEDICAL CO 2
143 STEVENS SUPPLY 1
```

You can also select and display only the grouped rows that have more than two orders by moving the COUNT function to the WHERE clause as shown:

```
>> SELECT C.CUSTNUM, CUSTNAME
+> FROM SALES.CUSTOMER C, SALES.ORDERS O
+> WHERE C.CUSTNUM = O.CUSTNUM
+> AND COUNT (DISTINCT ORDERNUM) >= 2
+> GROUP BY C.CUSTNUM, CUSTNAME;
S> LIST ALL;
CUSTNUM CUSTNAME
------- ------------------
123 BROWN MEDICAL CO
--- 1 row(s) selected.
```

The next query produces the same result as the previous one by using a HAVING clause:

```
>> SELECT C.CUSTNUM, CUSTNAME
+> FROM CUSTOMER C, ORDERS O
+> WHERE C.CUSTNUM = O.CUSTNUM
+> GROUP BY C.CUSTNUM, CUSTNAME
+> HAVING COUNT (DISTINCT ORDERNUM) >= 2;
```

A HAVING clause is similar to a WHERE clause, but the HAVING clause is applied to the results of the GROUP BY clause. A column you specify in a HAVING clause must be a grouping column or the argument of a function.

Determining Minimum and Maximum Values

You can use the MIN and MAX functions to determine a minimum and maximum value in a column or of an expression. You can apply the function to a group of rows or to the entire result of a SELECT command. For example, the following query determines the minimum and maximum unit price charged for each part in the current set of orders:

```
>> VOLUME SALES;
>> SELECT P.PARTNUM, MIN(UNIT_PRICE), MAX(UNIT_PRICE)
+> FROM PARTS P, ODETAIL O
+> WHERE P.PARTNUM = O.PARTNUM
+> GROUP BY P.PARTNUM;
S> LIST NEXT 3;
PARTNUM (EXPR) (EXPR)
------- ----------- -----------
212 2450.00 2500.00
244 2800.00 3500.00
255 3800.00 4000.00
```

Determining Which Columns to Group

By specifying different sets of columns in the GROUP BY clause, you change the results of the functions you apply to the group. Consider the following examples.

- This example counts employees in departments numbered less than 2000. Department 1000 has five employees.

```
>> SELECT DEPTNUM, COUNT(*)
+> FROM PERSNL.EMPLOYEE
+> WHERE DEPTNUM < 2000
+> GROUP BY DEPTNUM;
DEPTNUM (EXPR)
------- -------------------
1000 5
1500 4
--- 2 row(s) selected.
```

- This example counts employees with the same job code in each department. In department 1000, there are three employees with job code 500.

```
>> SELECT DEPTNUM, JOBCODE, COUNT(*)
+> FROM PERSNL.EMPLOYEE
+> WHERE DEPTNUM < 2000
+> GROUP BY DEPTNUM, JOBCODE;
DEPTNUM JOBCODE (EXPR)
------- ------- -------------------
1000 100 1
1000 500 3
1000 900 1
1500 100 1
1500 600 2
1500 900 1

--- 6 row(s) selected.
```

- This example groups employees by job code and counts the employees with the same job code in the same department. The result of this query is the same as the previous one, but the rows are organized differently.

```
>> SELECT JOBCODE, DEPTNUM, COUNT(*)
+> FROM PERSNL.EMPLOYEE
+> WHERE DEPTNUM < 2000
+> GROUP BY JOBCODE, DEPTNUM;
JOBCODE DEPTNUM (EXPR)
------- ------- -------------------
100 1000 1
100 1500 1
500 1000 3
600 1500 2
900 1000 1
900 1500 1
```

The group to which the function is applied is determined by all the grouping columns you specify. The last column you specify determines how precisely the groups are divided.

Note that you do not have to specify an ORDER BY clause unless you want the groups arranged in a particular order, or you want to specify break groups in the report. For more information, see Organizing Rows Into Break Groups on page 4-14.

Selecting Distinct Rows

The rows you retrieve with a SELECT command can contain duplicate values. If you want only one entry in your report for each distinct value, you can specify the DISTINCT keyword preceding the select list.

The following query reports which parts are currently on order. The ODETAIL table contains two rows for part number 212 and six rows for part number 244. The query selects only one row for each part.

```
>> SELECT DISTINCT PARTNUM
+> FROM SALES.ODETAIL;
PARTNUM
-------
212
244
255
2001
.
.
```

You can expand this query to print the part descriptions also:

```
>> SELECT DISTINCT P.PARTNUM, PARTDESC
+> FROM SALES.PARTS P, SALES.ODETAIL O
+> WHERE P.PARTNUM = O.PARTNUM;
PARTNUM PARTDESC
------- ------------------
212 PC SILVER, 20 MB
244 PC GOLD, 30 MB
. .
. .
```

The following examples illustrate different ways of selecting distinct values.

The PARTSUPP table contains rows that record a part number, the supplier number, the part cost, and the quantity received. If the supplier changes the cost of a part, more than one row of the PARTSUPP might describe the same part number and supplier. If you want to count the number of distinct suppliers of each part, you can use the following query:

```
>> SELECT PARTNUM, COUNT ( DISTINCT SUPPNUM )
+> FROM INVENT.PARTSUPP
+> GROUP BY PARTNUM;
S> LIST NEXT 2;
PARTNUM (EXPR)
------- ----------------------
212 2
244 2
S>
```

The next query counts the number of orders taken by each sales representative who has more than one customer with current orders:

```
>> SELECT SALESREP, COUNT (DISTINCT ORDERNUM)
+> FROM SALES.ORDERS OX
+> GROUP BY SALESREP
+> HAVING EXISTS (SELECT SALESREP
+> FROM SALES.ORDERS
+> WHERE OX.SALESREP = SALESREP
+> AND COUNT (DISTINCT CUSTNUM) > 1
+> GROUP BY SALESREP);
SALESREP (EXPR)
-------- -------------------
220 3
226 3
```

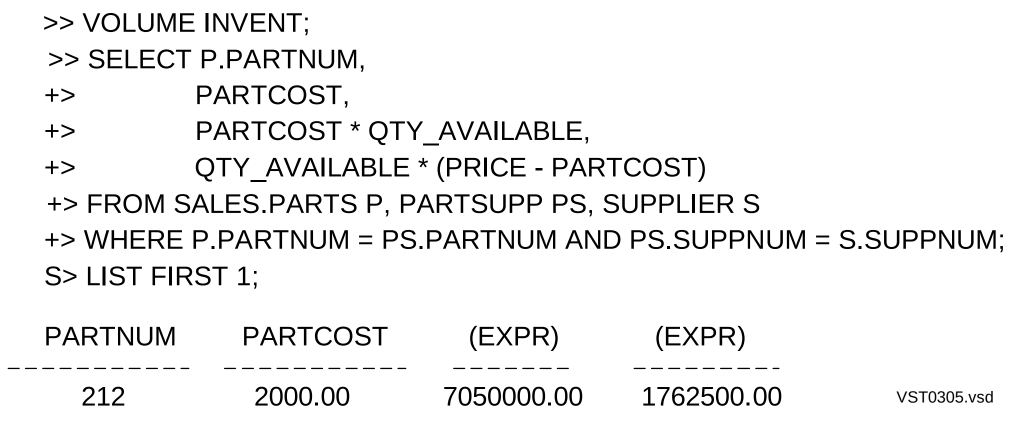
If you specify DISTINCT preceding the select list, you cannot specify it in any aggregate function in the select list or in any predicate of the WHERE or HAVING clause.

Using Expressions to Calculate Report Values

You can calculate values for items in a report output line. If you are printing the report in the default report format, you specify the calculation in the select list.

The query in Figure 3-5 includes two expressions in the select list. The first expression calculates the total part cost for the available quantity of each part. The second expression calculates the profit (price minus cost) for the available quantity of each part.

Figure 3-5. Expressions in the Select List



```
>> VOLUME INVENT;
 >> SELECT P.PARTNUM,
+>            PARTCOST,
+>            PARTCOST * QTY_AVAILABLE,
+>            QTY_AVAILABLE * (PRICE - PARTCOST)
 +> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM AND PS.SUPPNUM = S.SUPPNUM;
S> LIST FIRST 1;

   PARTNUM      PARTCOST        (EXPR)        (EXPR)
-----------   -----------     -------      ---------
     212         2000.00     7050000.00    1762500.00
```

An expression in the select list can refer to column names, literals, parameter names, and aggregate functions. These functions are described in Grouping Data for Calculations on page 3-15.

You can use the following arithmetic operators:

+ Addition
- Subtraction
* Multiplication
/ Division
** Exponentiation (ab)

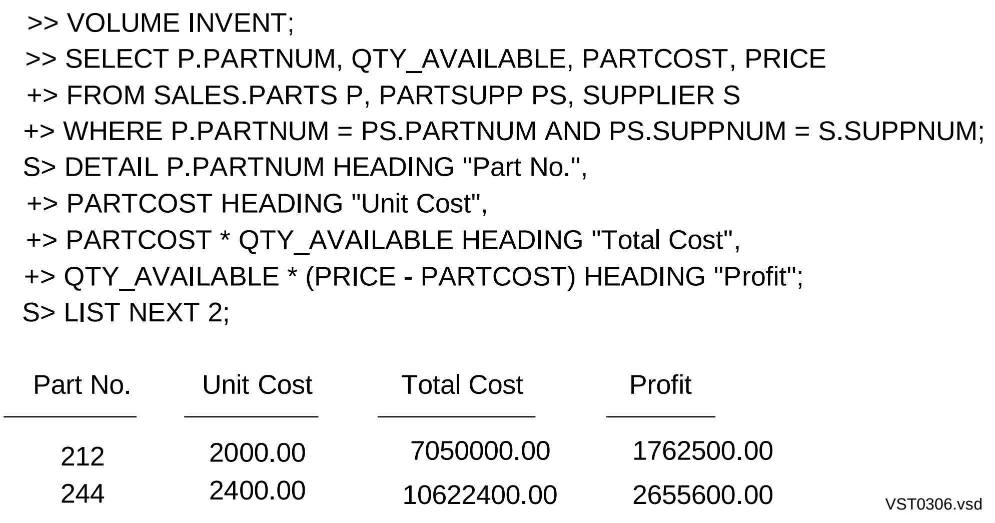
The order of evaluation of an expression from first to last is:

Expressions within parentheses
Unary operators, plus (+) and minus (-
Exponentiation
Multiplication and division
Addition and subtraction

For a complete description of expressions and syntax rules, see the *SQL/MP Reference Manual*.

If you are formatting a report, you can calculate column values in the select list or in a DETAIL command. Figure 3-6 shows another way to define the previous report by including expressions in the DETAIL command instead of the select list.

Figure 3-6. Expressions in the Detail Line



```
>> VOLUME INVENT;
>> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM AND PS.SUPPNUM = S.SUPPNUM;
S> DETAIL P.PARTNUM HEADING "Part No.",
+> PARTCOST HEADING "Unit Cost",
+> PARTCOST * QTY_AVAILABLE HEADING "Total Cost",
+> QTY_AVAILABLE * (PRICE - PARTCOST) HEADING "Profit";
S> LIST NEXT 2;

 Part No.    Unit Cost     Total Cost      Profit
----------  ----------   ------------   --------

   212        2000.00     7050000.00    1762500.00
   244        2400.00    10622400.00    2655600.00
```

VST0306.vsd

You should consider the following restrictions that apply to expressions depending on where you specify them:

You can include aggregate functions in a select list expression but not in a DETAIL command expression.

You can include report functions in a DETAIL command expression but not in a select list expression. The report functions are LINE_NUMBER, COMPUTE_TIMESTAMP, CURRENT_TIMESTAMP, and PAGE_NUMBER. However, you can use the NonStop SQL/MP date-time functions both in the select list and in DETAIL command expressions.

Using Parameters With SELECT Commands

You can use named parameters in a SELECT command to specify that a value will be provided when the command is executed. For the complete syntax description of named parameters, see the *SQL/MP Reference Manual*. To follow this example, you should know that the form of a named parameter is:

```
?[ simple-name]
```

For example, ?CUSTNO is a parameter that passes a customer number to SQLCI:

```
SELECT * FROM SALES.CUSTOMER WHERE CUSTNUM = ?CUSTNO ;
```

Named parameters can be useful in a report definition that you specify in an EDIT file and execute by using the OBEY command. To set the value, an operator enters a command such as the following before executing the command file:

```
>> SET PARAM ?CUSTNO 324 ;
```

You can specify the value for a parameter as a numeric or string literal, or you can use the CURRENT_TIMESTAMP or COMPUTE_TIMESTAMP functions to calculate the value. You cannot use the CURRENT function or any other date-time functions to specify a parameter value.

You cannot refer to parameters in a report formatting command. For another method of passing values to commands, see Using TACL to Pass Parameters on page 4-44.

Suppose a SELECT command in an command file named SALESUM consists of the following commands:

```
SET LIST_COUNT 0;
VOLUME SALES;
SELECT P.PARTNUM,
       SUM (QTY_ORDERED),
       SUM (UNIT_PRICE * QTY_ORDERED)
    FROM PARTS P, ORDERS R, ODETAIL OD
    WHERE R.ORDERNUM = OD.ORDERNUM AND OD.PARTNUM = P.PARTNUM
       AND ORDER_DATE BETWEEN 870101 AND ?TODAY
       AND P.PARTNUM BETWEEN ?FIRSTPART AND ?LASTPART
    GROUP BY P.PARTNUM
    ORDER BY P.PARTNUM;
LIST ALL;
```

To execute this query and print the default report, enter these commands:

```
>> SET PARAM ?FIRSTPART 2001,
+> ?LASTPART 5110,
+> ?TODAY 870811;
>> OUT_REPORT $S.#PRINTER;
>> OBEY SALESSUM;
```

The printed report appears as follows:

```
PARTNUM (EXPR) (EXPR)
------- -------------------- --------------------
2001 125 126900.00
2002 46 108000.00
2003 40 126900.00
. . .
. . .
. . .
5110 12 6300.00
```

You can display the current values of named parameters by using the SHOW PARAM command.

In a SELECT command, you can also use unnamed parameters if you plan to prepare the command before executing it. For descriptions of the SET PARAM and SHOW PARAM commands, see the *SQL/MP Reference Manual*.

Preparing a SELECT Command

You can use the PREPARE command to compile a SELECT command and then use the EXECUTE command to execute the query. This preparation is useful when you intend to produce several reports with the same SELECT command but based on data selected by different criteria. Preparation also allows you to use the EXPLAIN command to determine the system resources required to execute the query. For descriptions of the PREPARE, EXECUTE, and EXPLAIN commands, see the *SQL/MP Reference Manual*.

For example, suppose you want to print information about the orders from a customer assigned to a specific sales representative. This report is produced each month for sales representatives and all of their customers. You can create an command file that contains a command to prepare the SELECT command.

This SELPREP file contains the following PREPARE command:

```
PREPARE SELCUSTINFO FROM
"    SELECT * FROM SALES.CUSTOMER C, "
&"                 SALES.ORDERS R,   "
&"                 SALES.ODETAIL OD, "
&"                 SALES.PARTS P     "
&"   WHERE C.CUSTNUM = R.CUSTNUM     "
&"     AND R.ORDERNUM = OD.ORDERNUM  "
&"     AND OD.PARTNUM = P.PARTNUM    "
&"     AND C.CUSTNUM = ? "
&"     AND SALESREP = ?";
```

The SELECT command contains two unnamed parameters to be given values through an EXECUTE command.

You can create another command file to produce the reports; this file is called PRINTREP.

```
SET LIST_COUNT 0;

OUT_REPORT $S.#PRINTER

EXECUTE SELCUSTINFO USING 21, 223;

LIST ALL;

EXECUTE SELCUSTINFO USING 123, 226;

LIST ALL;

.

.

EXECUTE SELCUSTINFO USING 7777, 220;

LIST ALL;
```

To prepare the SELECT command enter the following command:

```
>> OBEY SELPREP;
( PREPARE command is displayed here.)
.
.
--- SQL command prepared.
```

To execute the prepared SELECT command for each report and print the report, enter the following command:

```
>> OBEY PRINTREP;
```

Using Views

A view is a logical table derived by projecting a subset of the columns or selecting a subset of the rows from one or more tables or from another view. You specify the definition of the view by using a SELECT command.

By defining a view, you can provide a way in which other users can retrieve values without specifying complicated queries. For example, the following commands define a view named CUSTORD to be used to print invoices and reports that summarize order information. The view columns are specified in parentheses following the view name. The rest of the command specifies the selection of columns and rows for the view.

```
>> VOLUME SALES;
>> CREATE VIEW CUSTORD
+>       ( CUSTNUM, CUSTNAME,
+>         STREET, CITY, STATE, POSTCODE,
+>         CREDIT,
+>         ORDERNUM, ORDER_DATE, DELIV_DATE,
+>         SALESREP,
+>         PARTNUM, UNIT_PRICE, QTY_ORDERED )
+> AS SELECT
+>         C.CUSTNUM, CUSTNAME,
+>         STREET, CITY, STATE, POSTCODE,
+>         CREDIT,
+>         O.ORDERNUM, ORDER_DATE, DELIV_DATE,
+>         SALESREP,
+>         PARTNUM, UNIT_PRICE, QTY_ORDERED
+>      FROM SALES.CUSTOMER C,
+>           SALES.ORDERS O,
+>           SALES.ODETAIL OD
+>      WHERE C.CUSTNUM = O.CUSTNUM
+>            AND O.ORDERNUM = OD.ORDERNUM ;
```

With the CUSTORD view defined, you can select data for an invoice by entering a simple SELECT command:

```
>> SELECT * FROM SALES.CUSTORD ;
```

To further simplify your task, you can specify in an EDIT file the set of report formatting commands to print the invoice.

The following commands define one version of an invoice. These commands are stored in a file named INVOICE. For explanations and examples of the report formatting commands used in this report, see Section 4, Customizing a Report.

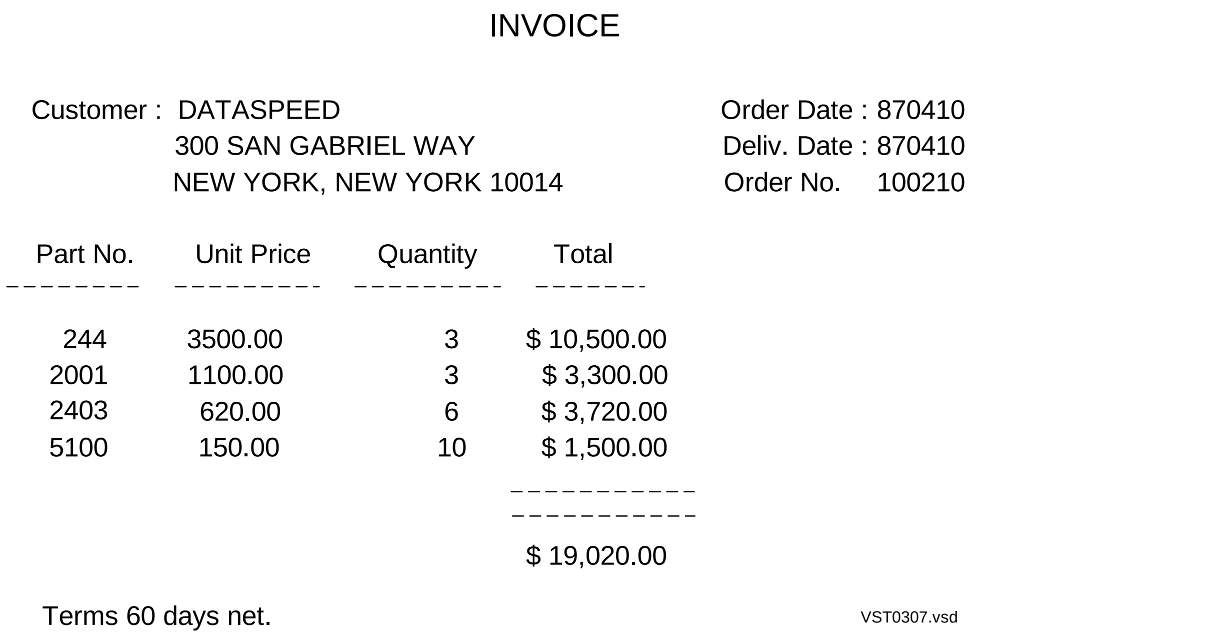
```
SET LAYOUT RIGHT_MARGIN 65, PAGE_LENGTH 24;
PAGE TITLE "INVOICE" CENTER;
REPORT TITLE "Customer: ", CUSTNAME,
       TAB 40, "Order Date: ", ORDER_DATE AS I6,
       SKIP 1,
       TAB 11, STREET,
       TAB 40, "Deliv. Date: ", DELIV_DATE AS I6,
       SKIP 1,
       TAB 11,
       CONCAT (CITY STRIP, ", ", STATE STRIP, SPACE 1,
POSTCODE),
       TAB 40, "Order No.", ORDERNUM;
DETAIL PARTNUM AS I6
           HEADING "Part No.",
       UNIT_PRICE AS F8.2
           HEADING "Unit Price",
       QTY_ORDERED AS I8
          HEADING "Quantity",
       UNIT_PRICE * QTY_ORDERED AS M<$ZZZ,ZZ9.99>
          HEADING "Total" NAME TOTALPRICE ;
TOTAL TOTALPRICE;
REPORT FOOTING "Terms 60 days net.";
```

To print an invoice, enter the following commands:

```
>> SET LIST_COUNT 0;
>> SELECT * FROM RWVIEWS.CUSTORD
+> WHERE CUSTNUM = 1234 AND ORDERNUM = 100210;
S> OBEY INVOICE;
```

Figure 3-7 shows a sample invoice (line numbers do not appear on the printed report).

Figure 3-7. Example of an Invoice



```
                              INVOICE

  Customer :  DATASPEED                          Order Date : 870410
              300 SAN GABRIEL WAY                Deliv. Date : 870410
              NEW YORK, NEW YORK 10014           Order No.    100210

   Part No.      Unit Price     Quantity      Total
  ---------     ----------     ----------   -------

     244          3500.00          3       $ 10,500.00
    2001          1100.00          3        $ 3,300.00
    2403           620.00          6        $ 3,720.00
    5100           150.00         10        $ 1,500.00
                                           -----------
                                           -----------
                                           $ 19,020.00

   Terms 60 days net.
```

VST0307.vsd

Using the view saves time if you want to define several reports based on the same data. For example, you might have a summary report that uses the same data as the preceding invoice. You can select information about orders with delivery dates prior to June 1, 2000 by entering the following command:

```
>> SELECT * FROM RWVIEWS.CUSTORD
+> WHERE DELIV_DATE < 000601 ;
```

You can then print a report using the retrieved data.

For more information about views, see the *SQL/MP Reference Manual*.

Using Subqueries

A subquery is a special form of the SELECT command. A subquery selects only for purposes of comparison. You specify subqueries in predicates of a search condition in the WHERE clause or the HAVING clause of a SELECT command.

A correlated subquery is evaluated for each row selected by the main query. A subquery that does not contain a correlated reference is evaluated once. The result is used for evaluating the WHERE clause against each row selected by the main query.

The following commands illustrate a query that does not contain a correlated reference:

```
>> SELECT SUPPNUM, PARTNUM, PARTCOST
+> FROM INVENT.PARTSUPP
+> WHERE PARTNUM = 2003
+> AND PARTCOST > (SELECT MIN(PARTCOST)
+> FROM INVENT.PARTSUPP
+> WHERE PARTNUM = 2003);
S> LIST ALL;
SUPPNUM PARTNUM PARTCOST
------- ------- --------
2 2003 1400.00
10 2003 1450.00
--- 2 row(s) selected.

>>
```

This query finds all suppliers who charge more than the minimum price for part number 2003. The subquery is evaluated once to determine the minimum cost for the part. Each row selected by the main query is compared to the result of the subquery.

You can replace the numeric literal 2003 with the parameter ?PART to create a general query for gathering this information for any part.

The following commands illustrate a correlated query which finds suppliers whose price exceeds the average price for a part:

```
>> SELECT SUPPNUM, PARTNUM, PARTCOST
+> FROM INVENT.PARTSUPP XP
+> WHERE PARTCOST > (SELECT AVG(PARTCOST)
+> FROM INVENT.PARTSUPP P
+> WHERE XP.PARTNUM = P.PARTNUM)
+> ORDER BY SUPPNUM;
S> LIST ALL;
SUPPNUM PARTNUM PARTCOST
------- ------- --------
1 212 2000.00
1 244 2400.00
1 255 3300.00
1 2405 500.00
2 2001 750.00
2 2003 1400.00
. . .
. . .
15 4102 21.00

--- 18 row(s) selected.

>>
```

The subquery is evaluated for each row selected by the main query. The FROM clause of the main query defines the correlation name XP for the PARTSUPP table. The subquery defines the correlation name P for the PARTSUPP table. The WHERE clause defines the correlation. The subquery averages rows from the PARTSUPP table with a PARTNUM value equal to the PARTNUM value of the current row from the outer query.

You do not have to define a correlation name for both the main query and subquery in order to perform this operation. You could use the implicit correlation name in the outer query.

The following specification uses an implicit correlation name to achieve the same result as the preceding query:

```
>> SELECT SUPPNUM, PARTNUM, PARTCOST
+> FROM INVENT.PARTSUPP
+> WHERE PARTCOST > (SELECT AVG(PARTCOST)
+> FROM INVENT.PARTSUPP P
+> WHERE PARTSUPP.PARTNUM = P.PARTNUM)
+> ORDER BY SUPPNUM;
```

Defining explicit correlation names provides clearer documentation of what the query does.

You can use quantified predicates for selecting rows in relation to all or any of the rows selected by a different search condition. For example, you can select PARTSUPP table rows that contain a part cost greater than the suggested price for the part in the PARTS table:

```
>> SELECT *
+> FROM PARTSUPP PS
+> WHERE PARTCOST > ANY (SELECT PRICE
+> FROM SALES.PARTS
+> WHERE PARTS.PARTNUM = PS.PARTNUM);
S> LIST NEXT 5;
PARTNUM SUPPNUM PARTCOST QTY_RECEIVED
------- ------- ------------ ------------
212 6 3000.00 2

***WARNING from SQLCI[10098] There are no more selected rows.

S>
```

The quantified predicate is evaluated for each row selected by the outer query. The subquery selects the PARTCOST from the PARTS table for the part with the same number as the current row of the outer query. Only one part qualifies for selection.

In the next example, the subquery sums the quantity ordered of each part retrieved in the outer query. The main query selects parts for which the quantity on order is greater than or equal to a specified percent of the parts available. The query allows you to enter the percent as a parameter each time you execute the query. The query is in an command file named ORDERPCT.

```
VOLUME SALES;
SET LIST_COUNT 0;
SELECT X.PARTNUM,
       PARTDESC,
       ORDERNUM,
       QTY_AVAILABLE,
       QTY_ORDERED
  FROM PARTS, ODETAIL X
  WHERE PARTS.PARTNUM = X.PARTNUM
      AND (?PERCENT/100) * QTY_AVAILABLE
          < ( SELECT SUM(QTY_ORDERED)
              FROM ODETAIL
              WHERE X.PARTNUM = ODETAIL.PARTNUM)
  ORDER BY X.PARTNUM;
DETAIL PARTNUM AS I4 HEADING "PART",
       PARTDESC,
       QTY_AVAILABLE AS I6 HEADING "AVAILABLE",
       ORDERNUM AS I6 HEADING "ORDER NO.",
       QTY_ORDERED AS I5 HEADING "ORDERED";
BREAK ON PARTNUM, PARTDESC, QTY_AVAILABLE;
SUBTOTAL QTY_ORDERED OVER QTY_AVAILABLE;
```

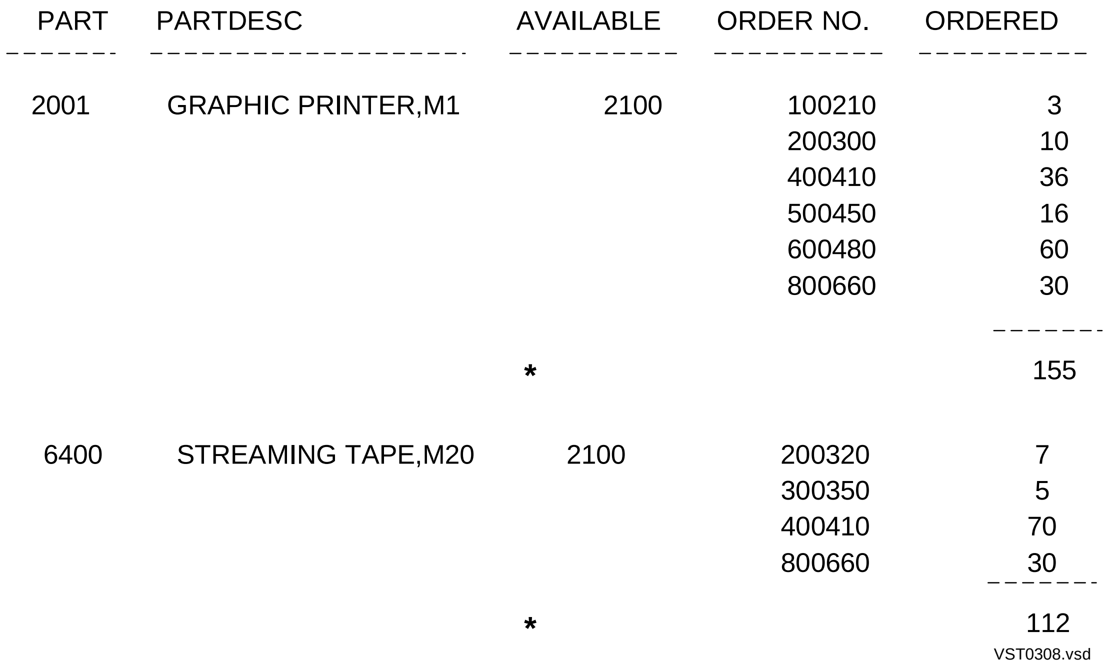
To execute this query using a value of 5 percent, enter:

```
>> SET PARAM ?PERCENT 5;
>> OBEY ORDERPCT;
>> LIST ALL;
```

The report in Figure 3-8 shows that part numbers 2001 and 6400 satisfy the conditions of the query.

If you do not specify OVER QTY_AVAILABLE in the SUBTOTAL command, the subtotals will be calculated when any break column value changes. In this example, the values in all three break columns change at the same time. Thus, printing the subtotals for each break column is redundant.

Figure 3-8. Example of Report With a Subquery



```
  PART    PARTDESC                    AVAILABLE    ORDER NO.     ORDERED
 -------  --------------------------  -----------  -----------  -----------

  2001     GRAPHIC PRINTER,M1               2100        100210            3
                                                        200300           10
                                                        400410           36
                                                        500450           16
                                                        600480           60
                                                        800660           30
                                                                     -------
                                     *                                  155

   6400     STREAMING TAPE,M20          2100           200320            7
                                                       300350            5
                                                       400410           70
                                                       800660           30
                                                                     -------
                                     *                                  112
```

VST0308.vsd

Consider the following points when using subqueries:

- A subquery must be enclosed in parentheses.
- The select list can be a single expression, an asterisk (*), or a correlation name followed by an asterisk. You cannot specify an asterisk in any predicates except EXISTS unless the FROM clause of the subquery refers to a single table or view consisting of a single column.
- A subquery in a comparison predicate must result in a single value; either one column of one row must satisfy the search condition or the select list element must be an aggregate function.
  A subquery in an EXISTS, IN, or quantified predicate can have more than one row that satisfies the search condition.
- You can nest subqueries up to 16 levels. Subqueries within the same WHERE clause are at the same level. A subquery within the WHERE clause of another subquery is at a different level.

Developing Multistep Queries

The following techniques use temporary tables to select data based on more than one query. To create a table, you use the CREATE TABLE statement. You must have access to a catalog to define a table, or you must have the authority to create your own catalog. For information about the requirements for creating tables, see the CREATE TABLE statement in the *SQL/MP Reference Manual*.

Multilevel Group Aggregates

Grouping Data for Calculations on page 3-15 describes ways to apply aggregate functions to groups of rows. If you want to apply aggregate functions to multiple levels of groups, you must specify more than one query and use temporary tables.

For example, suppose you want to report the average salary for each department and within each department for each job classification. Follow these steps:

1. Create a temporary table to contain the department number and average salary for each department. Use the INVOKE command to determine the data type for the DEPTNUM column of the DEPT table. Then use CREATE TABLE to create the temporary table:

```
>> CREATE TABLE DEPTAVG (
+> DEPTNUM NUMERIC (4) UNSIGNED NO DEFAULT,
+> AVGSAL NUMERIC (6) UNSIGNED NO DEFAULT )
+> CATALOG TEMPTABS ;
```

2. Insert the department number and average salary for each department in the DEPTAVG table:

```
>> INSERT INTO DEPTAVG
+> (SELECT DEPTNUM, AVG(SALARY)
+> FROM PERSNL.EMPLOYEE
+> GROUP BY DEPTNUM);
```

3. Create another temporary table to contain the job code and average salary for each type of job within a department:

```
>> CREATE TABLE JOBAVG (
+> DEPTNUM NUMERIC (4) UNSIGNED NO DEFAULT,
+> JOBCODE NUMERIC (4) UNSIGNED NO DEFAULT,
+> AVGSAL NUMERIC (6) UNSIGNED NO DEFAULT )
+> CATALOG TEMPTABS ;
```

4. Insert the department number, job code, and average salary for each job in each department in the JOBAVG table:

```
>> INSERT INTO JOBAVG
+> (SELECT DEPTNUM, JOBCODE, AVG(SALARY)
+> FROM PERSNL.EMPLOYEE
+> GROUP BY DEPTNUM, JOBCODE);
```

5. Join the temporary tables and select the report information:

```
>> SET LIST_COUNT 0;
>> SELECT D.DEPTNUM, JOBCODE, D.AVGSAL, J.AVGSAL
+> FROM DEPTAVG D, JOBAVG J
+> WHERE D.DEPTNUM = J.DEPTNUM
+> ORDER BY D.DEPTNUM;
S> NAME COL 3 DEPT_AVGSAL;
S> NAME COL 4 JOB_AVGSAL;
S> DETAIL DEPTNUM, DEPT_AVGSAL, JOBCODE, JOB_AVGSAL;
S> BREAK ON DEPTNUM, DEPT_AVGSAL;
S> LIST N 8;
```

```
DEPTNUM DEPT_AVGSAL JOBCODE JOB_AVGSAL
------- ----------- ------- ----------
1000 52000 100 137000
500 34666
900 19000
1500 41250 100 90000
600 29000
900 17000
2000 50000 100 13800
200 24000
S>
```

The BREAK ON command suppresses printing of the same department number and salary average in multiple lines. You can drop the tables or purge the data and reuse the tables in future reports.

Conditional Aggregates

To produce a report that contains aggregate function values calculated for groups selected by different WHERE clause criteria, use a multiple step query. For example, assume that a report counts the number of employees whose salaries are in these ranges:

Range 1 < 20000
Range 2 < 50000
Range 3 < 200000

The report contains one detail line for each department.

Follow these steps to create the report:

1. Create a temporary table for the report values. Define the RANGE n columns with the system default so you do not have to insert a value:

   ```
   >> CREATE TABLE DEPTTEMP (
   +> DEPTNUM NUMERIC (4) UNSIGNED NO DEFAULT,
   +> RANGE1 INTEGER UNSIGNED DEFAULT SYSTEM,
   +> RANGE2 INTEGER UNSIGNED DEFAULT SYSTEM,
   +> RANGE3 INTEGER UNSIGNED DEFAULT SYSTEM )
   +> CATALOG TEMPTABS ;
   ```

2. Insert the count for RANGE1 in the DEPTTEMP table:

   ```
   >> INSERT INTO DEPTTEMP (DEPTNUM, RANGE1)
   +> (SELECT DEPTNUM, COUNT(*) FROM PERSNL.EMPLOYEE
   +> WHERE SALARY < 20000
   +> GROUP BY DEPTNUM );
   ```

3. Enter two more INSERT commands to insert the counts for RANGE2 and RANGE3. For each command, substitute RANGE2 or RANGE3 in the select list and change the value in the WHERE clause to 50000 first and then to 200000.

4. Select the report information from the DEPTTEMP table:

```
>> SET LIST_COUNT 0;
>> SELECT DEPTNUM, SUM(RANGE1), SUM(RANGE2), SUM(RANGE3)
+> FROM DEPTTEMP
+> GROUP BY DEPTNUM;
S> DETAIL DEPTNUM,
+> COL 2 AS I12 HEADING "SAL. < 20000",
+> COL 3 AS I12 HEADING "SAL. < 50000",
+> COL 4 AS I12 HEADING "SAL. < 200000";
S> LIST ALL;

DEPTNUM SAL. < 20000 SAL. < 50000 SAL. < 200000
------- ------------ ------------ -------------

1000 1 3 5
1500 1 3 4
2000 0 4 5
. . . .
. . . .
4000 1 9 15
9000 0 1 2
--- 11 row(s) selected.
>>
```

By default, the range columns that receive no values from an INSERT command are set to the system default value of zero; thus, they do not affect the final sums computed for each column.

You can use this technique for other queries when you want to group rows and compute aggregates based on different conditions.

Row Value as Percent of All Row Values

You can also use multiple step queries to compute what percent the current row value is of all row values. For example, the following report displays the percent that an individual's salary is of all salaries in a department. The report is produced with the following steps:

1. Create a temporary table to contain the average salary for each department:

```
>> CREATE TABLE TEMPTABS.AVGTEMP (
+> DEPTNUM NUMERIC (4) UNSIGNED NOT NULL,
+> AVGSAL NUMERIC (6) UNSIGNED NOT NULL)
+> CATALOG TEMPTABS ;
```

2. Insert the department number and average salary into the temporary table:

```
>> INSERT INTO TEMPTABS.AVGTEMP
+> (SELECT DEPTNUM, AVG(SALARY) FROM PERSNL.EMPLOYEE
+> GROUP BY DEPTNUM);
```

3. Select the information for the report. Include an expression in the select list to compute the percent of the department average:

```
>> SELECT E.DEPTNUM, EMPNUM, LAST_NAME, SALARY,
+> SALARY/AVGSAL*100.00
+> FROM PERSNL.EMPLOYEE E, TEMPTABS.AVGTEMP A
+> WHERE E.DEPTNUM = A.DEPTNUM;
S> DETAIL DEPTNUM, EMPNUM, LAST_NAME,
+> SALARY AS F10.2,
+> COL 5 AS F10.2 HEADING "PCT OF AVG";
S> LIST ALL;

DEPTNUM EMPNUM LAST_NAME SALARY PCT OF AVG
------- ------ --------------- ---------- ----------

1000 23 HOWARD 137000.10 263.46
1000 202 CLARK 25000.75 48.08
1000 208 CRAMER 19000.00 36.54
. . . . .
. . . . .
. . . . .
9000 1 GREEN 175500.00 165.00
9000 337 CLARK 37000.00 34.82

--- 57 row(s) selected.

>>
```

You can drop the temporary table or purge the data and keep the table for use in producing future reports.

4 Customizing a Report

You can use the report formatting commands and the layout and style options to enhance a report. The examples in this section show you how to produce the special effects you might want in a report. The following tasks are described:

- Defining the layout by setting margins, paginating, and spacing items and lines
- Organizing rows into break groups to emphasize sets and subsets of values and to compute subtotals
- Labeling information with column headings, titles, footings, line numbers, and descriptive text
- Formatting data such as numeric values, monetary values, names, dates and times; suppressing leading and trailing zeros; truncating values; replacing blank print positions with filler characters
- Specifying the items to be used in detail lines
- Printing items or line entries conditionally
- Redefining special characters such as the underline and decimal point
- Calculating totals and subtotals
- Printing double-byte characters

Defining the Layout

You control the layout of a report by specifying margins, pagination, and spacing.

Setting Margins

The current setting of the layout options LEFT_MARGIN and RIGHT_MARGIN determines the margins of a report.

If you want a margin to precede the leftmost printed item in your report, you must set the left margin to the number of blanks desired. For example, when the default left margin setting (0) is in effect, the first character of the output line is printed in print position 1.

Note. YA print position is defined in this manual as the space in the output line occupied by one single-byte character. A double-byte character occupies two print positions. When calculating print positions for an output line, special consideration must be given to cases where the output line can contain both single and double-byte characters. For more information, see Printing Double-Byte Characters on page 4-58.

If you want a margin of 8 print positions (a blank report field of 8 single-byte or 4 double-byte characters), enter SET LAYOUT LEFT_MARGIN 8. The detail line output begins in print position 9 with 8 blanks preceding each printed or displayed line.

When you are working at a terminal, the default OUT_REPORT file is the terminal. The default right margin is 80 for most terminals. To display the current right margin, enter the following:

```
>> SHOW LAYOUT RIGHT_MARGIN;
RIGHT_MARGIN     80
```

If you are designing a report to be printed on a wider page, you can set the right margin as needed. For example, if you want the last print position to be 100, enter the following command:

```
>> SET LAYOUT RIGHT_MARGIN 100;
```

A report with a left margin of 8 and a right margin of 80 can have 72 single-byte characters per displayed or printed line. The first character is in position 9 and the last character is in position 80.

The margins you set stay in effect until you end your SQLCI session or you reset the margins.

Output lines (detail lines, titles, footings, or any other output) that extend beyond the right margin are folded to the next line. If you are using the default detail line, you can use the LOGICAL_FOLDING layout option to specify that you want the detail lines broken between print items rather than within a print item. If you are specifying a detail line, insert a SKIP clause before a print item to force the item to be printed on the next line.

Figure 4-1 and Figure 4-2 illustrate the default margin settings for displayed and printed output.

Figure 4-1. Default Margin Settings - Displayed Report

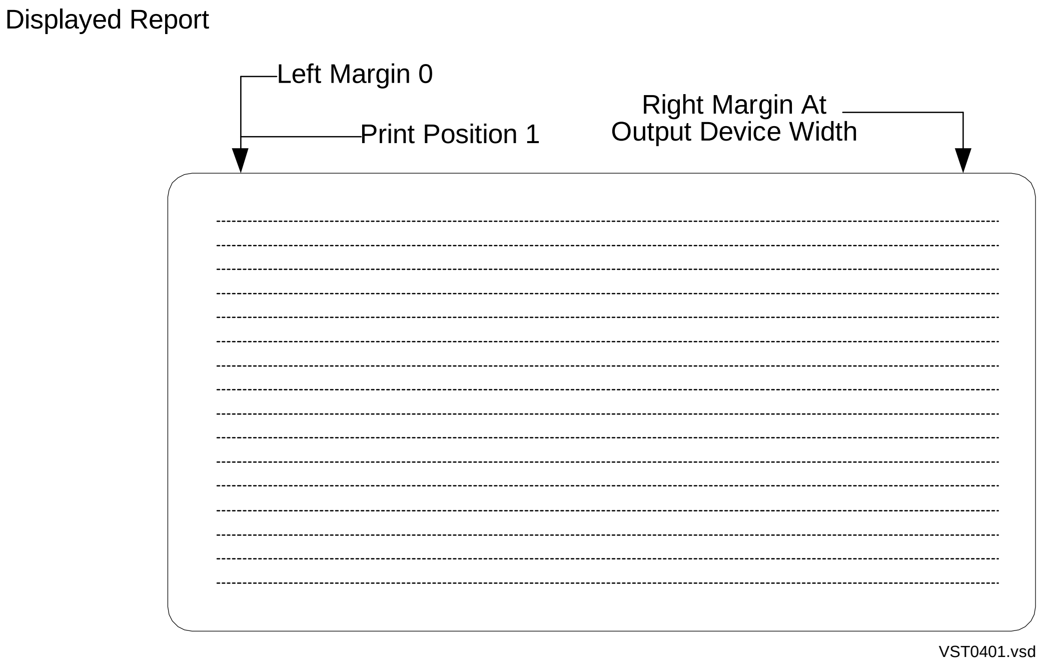
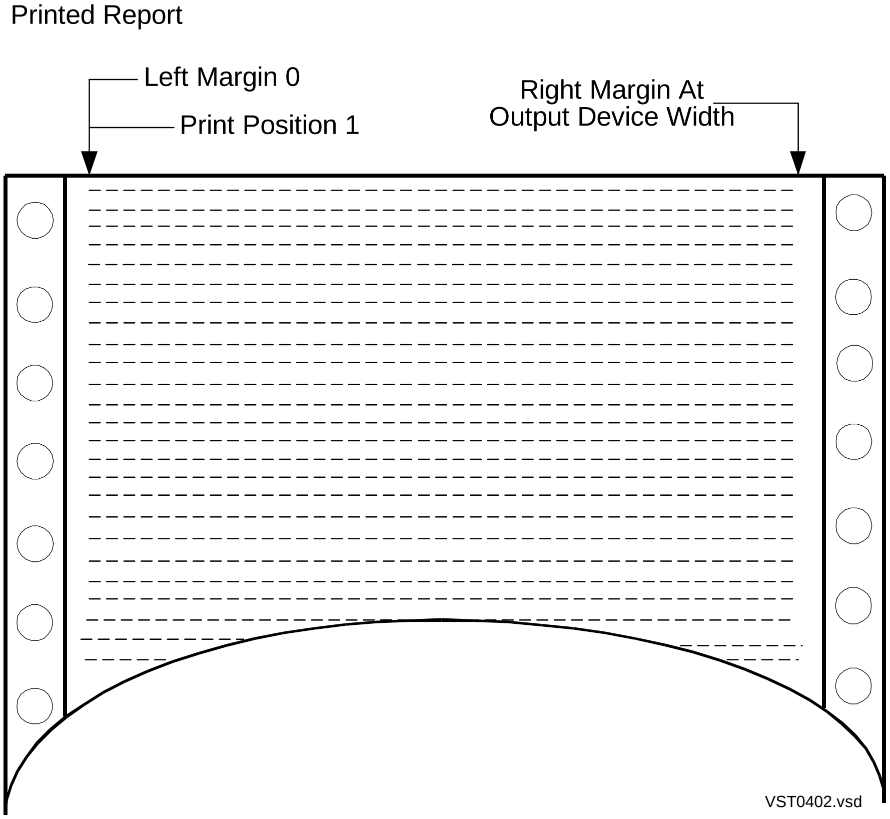


Figure 4-2. Default Margin Settings - Printer Report



The default top and bottom margins are one line each. You can increase the apparent size of these margins with a PAGE TITLE or PAGE FOOTING command. If you have no title, specify PAGE TITLE "" for a top margin of 3 lines or specify `PAGE TITLE SKIP` *`desired-margin-size`* `- 3` for more than 3 lines.

For example, the following command produces a margin size of 4 blank lines preceding the headings (SKIP 4 - 3 = SKIP 1):

```
S> PAGE TITLE SKIP 1;

   ( 4 blank lines appear here. )

CUSTNUM CUSTNAME
------- --------
```

The first blank line is the default top margin. The second blank line is the specified skip. The third blank line is the default skip for the title line. The fourth blank line is the default skip following the title line.

If you specify text in the title line, specify `PAGE TITLE SKIP` *`desired-margin-size - 1`*, *`" text".`* For example, the following command produces a desired margin size of 4 lines (SKIP 4 - 1 = SKIP 3):

```
S> PAGE TITLE SKIP 3, "Summary of Employees"; (
 4 blank lines appear here. )

Summary of Employees

CUSTNUM CUSTNAME

------- --------
```

The first blank line is the default top margin. The next three blank lines are the specified skips. The blank line following the title is the default skip that always follows the title line.

You use the same technique for bottom margins, except SKIP follows the footing text if specified. In the next example, the page footing is positioned above a bottom margin of 3 lines (2 lines specified by SKIP 2 and 1 line, which is the default bottom margin):

```
S> PAGE FOOTING "Page ", PAGE_NUMBER AS I2 , SKIP 2;
.
.
Page 1



( New page starts here. )
```

For more information about these commands, see Titles on page 4-18 and Footings on page 4-22.

Paginating

The following report writer features control where page breaks occur in a report:

PAGE_LENGTH

This layout option specifies the number of lines from the top to the bottom of a page. For example, the following command sets a page length of 55 lines:

```
>> SET LAYOUT PAGE_LENGTH 55;
```

The default page length for a printed report is 60 lines. On a terminal, the default page length is ALL; the entire report is a single page unless you specify the PAGE clause.

When designing your report, remember that the default top and bottom margins (1 line each) and the lines of the page footing fit within the page length. The report writer uses the remaining space for detail lines, titles, footings, subtotals, and totals.

PAGE Clause

In detail lines, titles, and footings, you can specify a page break by including the print item PAGE *number*. The variable *number* specifies the number of the next page and starts a new page numbering sequence. If you omit number, the current sequence continues. For example, the following command specifies a page break after each break footing is printed:

```
S> BREAK FOOTING JOBCODE ( "Job Title: ",
+> JOBDESC, PAGE);
```

In this example, one numbering sequence is used from the beginning to the end of the report.

NEED clause

In detail lines, titles, and footings, you can specify the number of subsequent lines that must fit on the current page. If the lines do not fit, a page break occurs. For example, the following command specifies that the break title must be printed on the next page unless the next 4 lines of output fit on the current page:

```
S> BREAK TITLE JOBCODE ( NEED 4, "Job Code:
",
+> JOBCODE);
```

You can specify the maximum number of pages to be printed in a report by setting the PAGE_COUNT layout option. For example, the following command sets the limit to 100 pages:

```
>> SET LAYOUT PAGE_COUNT 100;
```

The default PAGE_COUNT value is ALL; the entire report is printed or displayed.

If you want to number the pages in a report, you can use the PAGE_NUMBER function to retrieve the number of the current page. The first page of the report is numbered 1. The report writer maintains the page count based on the page breaks that occur. These page breaks include page breaks generated by a PAGE clause or a NEED clause.

The report in Figure 4-3 illustrates features related to page breaks and page numbering. The horizontal lines indicate where the page breaks occur.

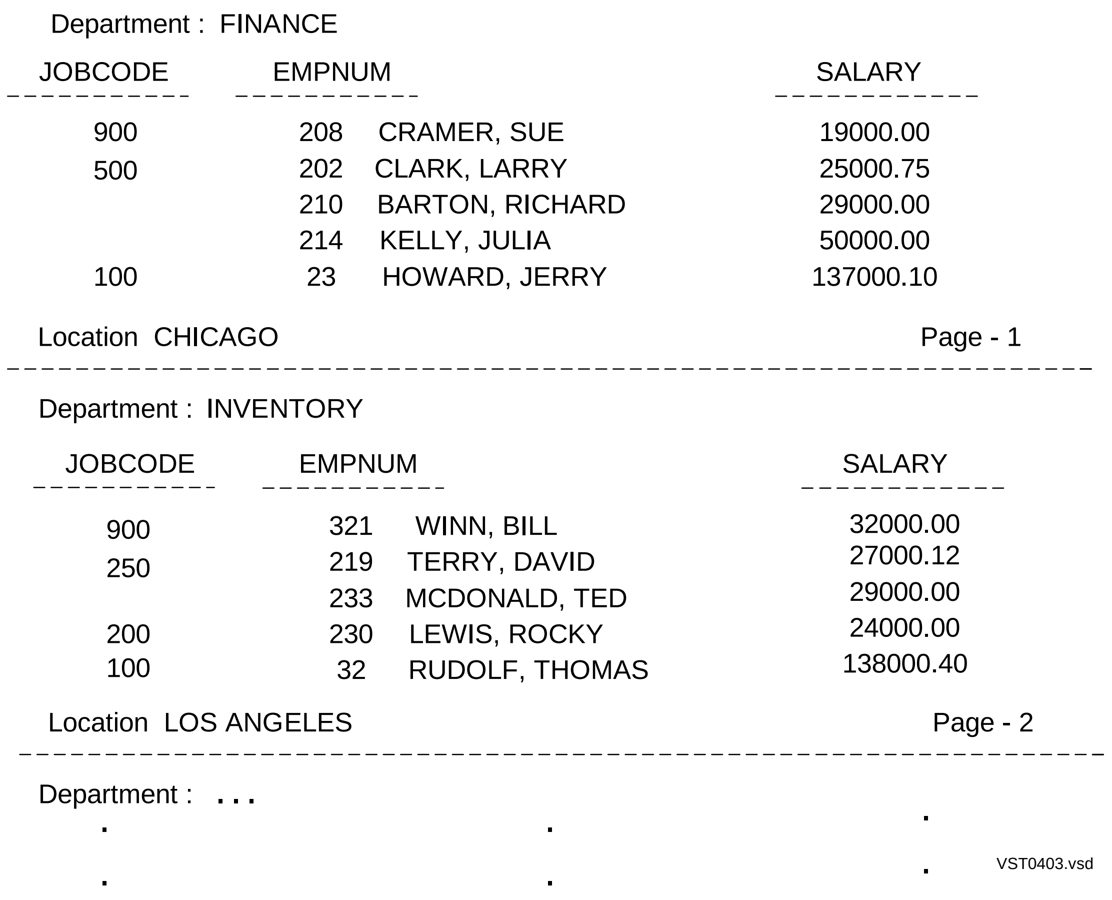
The following command selects the data for the report:

```
>> SELECT *
+> FROM PERSNL.EMPLOYEE E, PERSNL.DEPT
+> WHERE E.DEPTNUM = DEPT.DEPTNUM
+> ORDER BY E.DEPTNUM, JOBCODE DESC ;
```

The following commands define the report:

```
S> DETAIL JOBCODE,
+> EMPNUM,
+> CONCAT (LAST_NAME STRIP, ", ", FIRST_NAME)
+> AS A25 NOHEAD,
+> SALARY;
S> BREAK ON E.DEPTNUM, JOBCODE;
S> BREAK FOOTING E.DEPTNUM (PAGE) ;
S> PAGE TITLE "Department: ", DEPTNAME ;
S> PAGE FOOTING "Location ", LOCATION, TAB 50,
+> "Page - ", PAGE_NUMBER AS I2;
S> SET LAYOUT PAGE_LENGTH 15;
```

Figure 4-3. Example of Pagination Features



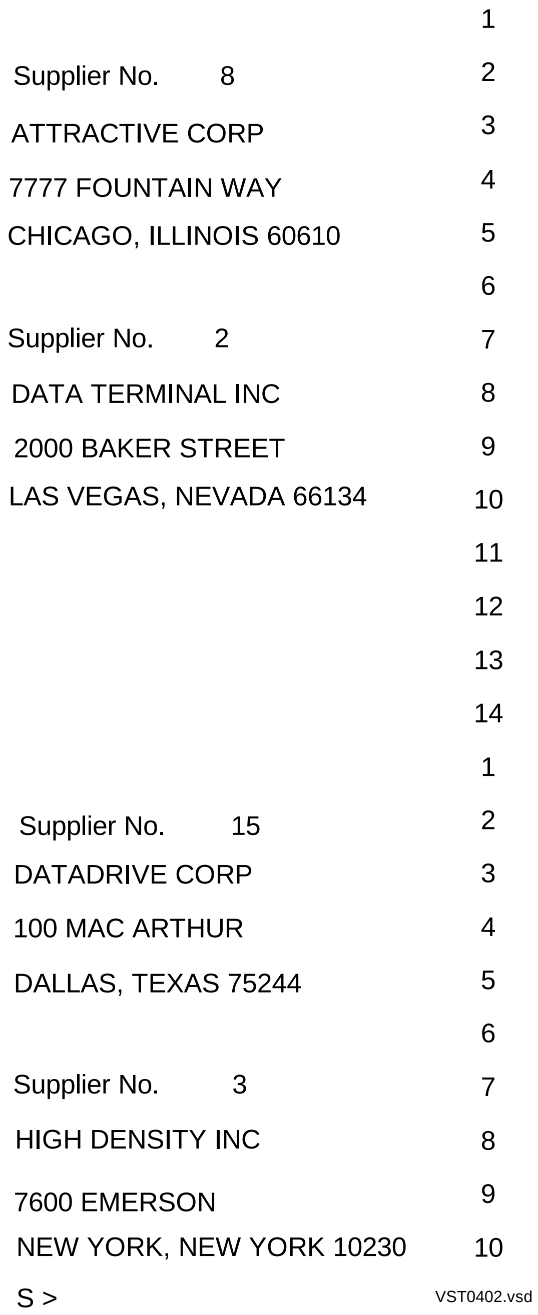
```
   Department :  FINANCE

  JOBCODE       EMPNUM                              SALARY
 -----------   -----------                        ------------

     900          208   CRAMER, SUE                 19000.00
     500          202   CLARK, LARRY                25000.75
                  210   BARTON, RICHARD             29000.00
                  214   KELLY, JULIA                50000.00
     100           23   HOWARD, JERRY              137000.10

  Location  CHICAGO                                       Page - 1
 -----------------------------------------------------------------
  Department :  INVENTORY

   JOBCODE       EMPNUM                              SALARY
  -----------   -----------                        ------------

     900          321   WINN, BILL                  32000.00
     250          219   TERRY, DAVID                27000.12
                  233   MCDONALD, TED               29000.00
     200          230   LEWIS, ROCKY                24000.00
     100           32   RUDOLF, THOMAS             138000.40

  Location  LOS ANGELES                                   Page - 2
 -----------------------------------------------------------------
  Department :  ...
       .                        .                   .
       .                        .                   .
```

The NEED clause is useful when you want to keep a set of lines together, such as a complete address. In the next example, a page break will not occur in the middle of information about a single supplier.

```
>> SELECT * FROM INVENT.SUPPLIER
+> ORDER BY SUPPNAME;
S> DETAIL NEED 4, "Supplier No.", SUPPNUM NOHEAD, SKIP 1,
+>        SUPPNAME NOHEAD, SKIP 1,
+>        STREET NOHEAD, SKIP 1,
+>        CONCAT (CITY STRIP, ", ", STATE STRIP,
+>                SPACE 1, POSTCODE) NOHEAD,
+>                SKIP 1 ;
S> SET LAYOUT PAGE_LENGTH 14;
S> LIST FIRST 4;
```

Figure 4-4 shows the first 4 rows of output. The column of numbers on the right does not appear in the report; the numbers indicate the 14 lines on each page. The third detail line appears on the second page because only 3 more lines are available on the first page.

Figure 4-4. Result of Using the NEED Clause



```
                                         1
 Supplier No.      8                     2
 ATTRACTIVE CORP                         3
 7777 FOUNTAIN WAY                       4
 CHICAGO, ILLINOIS 60610                 5
                                         6
 Supplier No.      2                     7
 DATA TERMINAL INC                       8
 2000 BAKER STREET                       9
 LAS VEGAS, NEVADA 66134                10
                                        11
                                        12
                                        13
                                        14
                                         1
 Supplier No.      15                    2
 DATADRIVE CORP                          3
 100 MAC ARTHUR                          4
 DALLAS, TEXAS 75244                     5
                                         6
 Supplier No.      3                     7
 HIGH DENSITY INC                        8
 7600 EMERSON                            9
 NEW YORK, NEW YORK 10230               10
 S >
```

VST0402.vsd

Spacing Items and Lines

The following report writer features control spacing of print items and lines:

LINE_SPACING

This layout option indicates to the report writer how many lines to advance between report lines. The default setting for LINE_SPACING when you begin an SQLCI session is 1, which results in a single-spaced report. For double spacing, enter the following:

```
>> SET LAYOUT LINE_SPACING 2;
```

SKIP clause

You can specify a SKIP clause as a print item in a detail line, title, or footing. SKIP advances the number of lines you specify before displaying or printing the next item. The LINE_SPACING option defines the increment for the SKIP clause.

For example, suppose LINE_SPACING is 1. The following detail line specifies that the supplier number and name appear on one line; the street on the next line; and the city, state, and postal code on the following line:

```
S> DETAIL SUPPNUM, SUPPNAME, SKIP 1, STREET,
+> SKIP 1, CITY, STATE, POSTCODE;
```

If you set LINE_SPACING to 2 before printing the report, a blank line appears between each output line. SKIP 1 advances 2 lines (SKIP *number* * LINE_SPACING *number*).

SPACE option

The SPACE layout option specifies the default number of single-byte spaces between print items of a detail line as follows:

```
>> SET LAYOUT SPACE 3;
```

The number you specify stays in effect until you end your SQLCI session or change the number. The default setting when you begin an SQLCI session is 2 spaces. The report writer does not insert spaces preceding or following a string literal unless you specify a heading for the string.

SPACE clause

You can use the SPACE clause as a print item in a detail line, title, or footing to insert a specified number of spaces before displaying or printing the next item. This clause temporarily overrides the SPACE option; the clause determines the amount of space between the print items that precede and follow the clause. For example, output from the following detail line has 2 spaces between all items except the EMPNUM and LAST_NAME items:

```
S> DETAIL EMPNUM, SPACE 5, LAST_NAME,
+> FIRST_NAME, JOBCODE;
```

The SPACE clause is also useful in titles and footings. For example, the following command generates a title with 6 spaces between DEPTNUM and the LOCATION label:

```
S> PAGE TITLE "Department No. ",
+> DEPTNUM AS I4,
+> SPACE 6, "Location: ", LOCATION;
```

Note If you are defining a report that contains double-byte characters, see Printing Double-Byte Characters on page 4-58 for special considerations regarding using the SPACE clause.

TAB clause

Another way to position print items is by using the TAB clause as a print item. You can tab to a specific print position in a detail line, title, or footing. The position you specify is not relative to the current left margin; it is an absolute position relative to the available print positions on the output device. However, you must specify a tab that is within the margins.

For example, suppose the left margin is 8 and the right margin is 80. The following command prints the Location label beginning at print position 35. The label begins in the 27th print position to the right of the left margin.

```
S> PAGE TITLE "Dept. No. ", DEPTNUM AS I4,
+> TAB 35, "Location: ", LOCATION;
```

Figure 4-5 illustrates the result of the preceding title command as it appears on a terminal and a printed page.

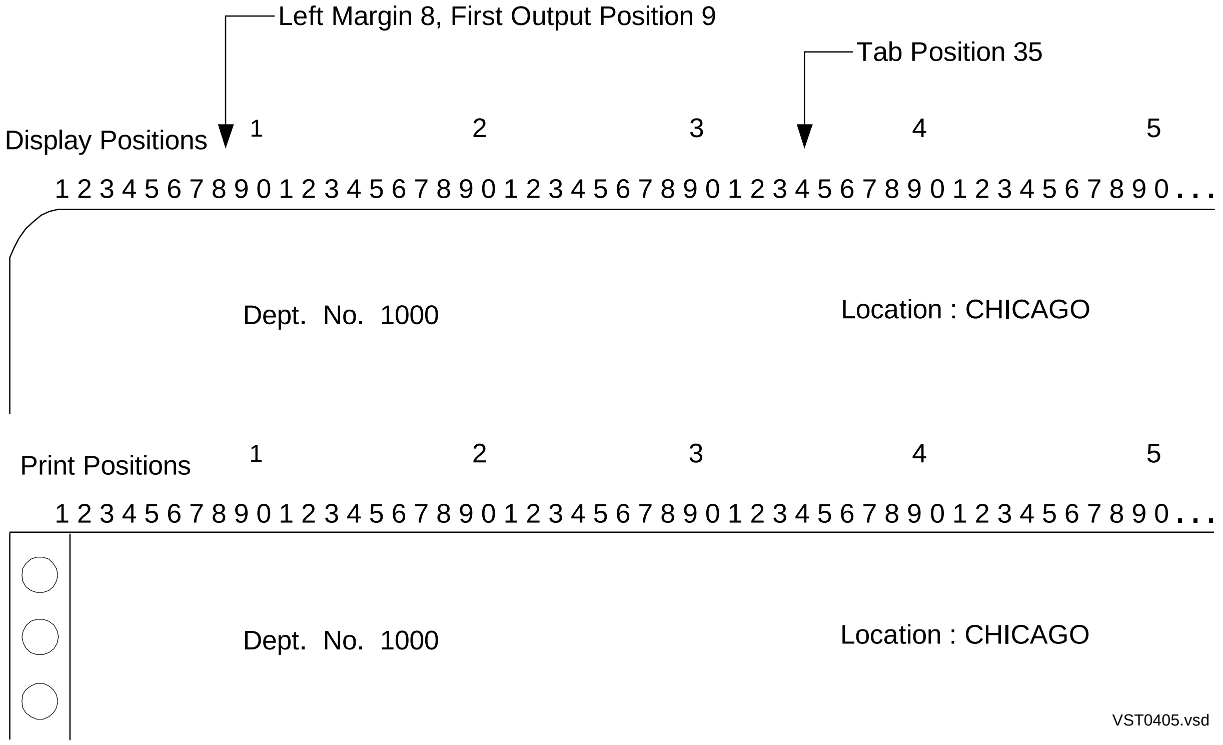
If you change the left margin of a report, you must also change any tab specifications to ensure that the alignment of information does not change relative to other information in the report.

Note If you are defining a report that contains double-byte characters, see“Printing Double-Byte Characters on page 4-58 for special considerations regarding using the TAB clause.

AS clause

You can also control the spacing of items through the display formats you specify in an AS clause. For more information, see Formatting Data Values on page 4-28.

Figure 4-5. Tabbing to a Display or Print Position



Specifying the Items in a Detail Line

In the default report format, the select list determines which print items appear in each detail line of the report. This default detail line can be described in terms of the DETAIL command as follows:

```
DETAIL COL 1, COL 2, COL 3,... ;
```

Each column number corresponds to the ordinal position of an item in the select list. For example, the following select list has a default detail line of 5 items:

```
>> SELECT EMPNUM, LAST_NAME, FIRST_NAME, JOBCODE, SALARY
+> FROM PERSNL.EMPLOYEE
+> ORDER BY SALARY DESCENDING;
S> LIST NEXT 3;
EMPNUM LAST_NAME FIRST_NAME JOBCODE SALARY
------ --------------- -------------- ------- -----------

1 GREEN ROGER 100 175500.00
32 RUDLOFF THOMAS 100 138000.40
23 HOWARD JERRY 100 137000.10
S>
```

You can refer to columns LAST_NAME and FIRST_NAME as COL 2 and COL 3.

```
S> DETAIL col 2, col 3;
```

When you use a DETAIL command, consider these points:

- You can specify only one DETAIL command at a time. You can edit the command by using FC, replace the command by reentering it, or delete the command by entering RESET REPORT DETAIL.
- In report formatting commands, you can refer to the items of the select list by column name or by column number

  Note. When you refer to a select list column by number in a SELECT command clause such as GROUP BY and ORDER BY, you specify only the number: for example, GROUP BY1. When you refer to a select list column by number in a report formatting command such as BREAK ON, you specify COL *number*, for example, BREAK ON Col 2.

- You can omit a correlation name when specifying a column in a report command if the reference is unambiguous. For example, if D.DEPTNUM and E.DEPTNUM are items in the select list, you must include the correlation name in report commands. If DEPTNUM appears only once in the list as E.DEPTNUM, you can omit the correlation name in report commands.
- You can define a report and specify elements such as page titles, footings, subtotals, and totals without specifying a DETAIL command. The default DETAIL line is in effect at the select-in-progress prompt.

Use the DETAIL command when you want to do either of the following:

- Omit select list columns from the output line. For example, you might need to select a column for the purpose of ordering the rows, but you do not want the column to appear in the report.

  ```
  >> SET LIST_COUNT 0;
  >> SELECT EMPNUM, LAST_NAME, FIRST_NAME, JOBCODE, SALARY
  +> FROM PERSNL.EMPLOYEE
  +> ORDER BY SALARY DESCENDING;
  S> DETAIL EMPNUM, LAST_NAME, FIRST_NAME, JOBCODE;
  S> LIST FIRST 2;

  EMPNUM LAST_NAME FIRST_NAME JOBCODE
  ------ ---------------- --------------- -------

  1 GREEN ROGER 100
  32 RUDLOFF THOMAS 100
  S>
  ```

- Specify headings, spacing between items, multiple line entries, concatenation of items, conditional printing of information, and other special handling of the information.

If you decide that you want to enhance the format of the information selected by the previous command, you can enter a DETAIL command. For example, to override the default headings, enter the following:

```
S> DETAIL EMPNUM HEADING "Employee No.",
+> LAST_NAME NOHEAD,
+> FIRST_NAME NOHEAD,
+> JOBCODE HEADING "Job Code";
S> LIST FIRST 2;
Employee No. Job Code
------------ --------

1 GREEN ROGER 100
32 RUDLOFF THOMAS 100
S>
```

For more information about defining headings, see Column Headings on page 4-16.

Naming Select List and Detail Line Items

You can assign an alias name to any item in the select list by using the NAME command. You can then refer to that item by its alias in a DETAIL command or in any other report formatting command. Alias names are useful for referring to expressions.

In the following example, the alias name TOTAL_COST is assigned to COL 3, and the alias name PROFIT is assigned to COL 4. The alias names are used in the DETAIL command:

```
>> SET LIST_COUNT 0;
>> SELECT P.PARTNUM, PARTCOST,
+> PARTCOST * QTY_AVAILABLE,
+> QTY_AVAILABLE * (PRICE - PARTCOST)
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+> AND PS.SUPPNUM = S.SUPPNUM;
S> NAME COL 3 TOTAL_COST;
S> NAME COL 4 PROFIT;
S> DETAIL PARTNUM HEADING "Part No.",
+>        PARTCOST HEADING "Unit Cost",
+>        TOTAL_COST HEADING "Total Cost",
+>         PROFIT HEADING "Profit";
```

You can refer to a select list column by an alias name in any part of your report definition.

If you need to refer to a detail item that does not have a column name or number, you can assign the item a detail alias name. For example, the NAME clause in the following DETAIL command defines the detail alias name PROFIT:

```
>> SET LIST_COUNT 0;
>> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM AND PS.SUPPNUM = S.SUPPNUM;
S> DETAIL PARTNUM HEADING "Part No.",
+>    QTY_AVAILABLE HEADING "Units",
+>    PARTCOST HEADING "Unit Cost",
+>    PARTCOST * QTY_AVAILABLE HEADING "Total Cost",
+>    QTY_AVAILABLE * (PRICE - PARTCOST)
+>      HEADING "Profit"
+>       NAME PROFIT;
```

The following TOTAL command refers to the PROFIT column:

```
S> TOTAL PROFIT;
```

You cannot refer to a detail alias name within the DETAIL command itself; you can only define the name there and use it in succeeding commands.

Organizing Rows Into Break Groups

In addition to grouping rows for aggregate function calculations, you can group rows into break groups. The important differences between these two types of groups are as follows:

- A group formed by a GROUP BY clause results in one row; the row is a combination of all rows in the group. Typically, each item in the select list either has the same value in all of the rows of the group or is an aggregate function applied to the values in the rows of the group. See Grouping Data for Calculations on page 3-15.
- A break group is formed by specifying an ORDER BY clause and BREAK ON command. In a break group, rows are not combined. Each row appears in the report. The rows are grouped together, and the break column value appears only once unless a page break occurs in the middle of the break group.

You can print a title before the break group rows and print a footing following the rows. You can calculate subtotals following a break group.

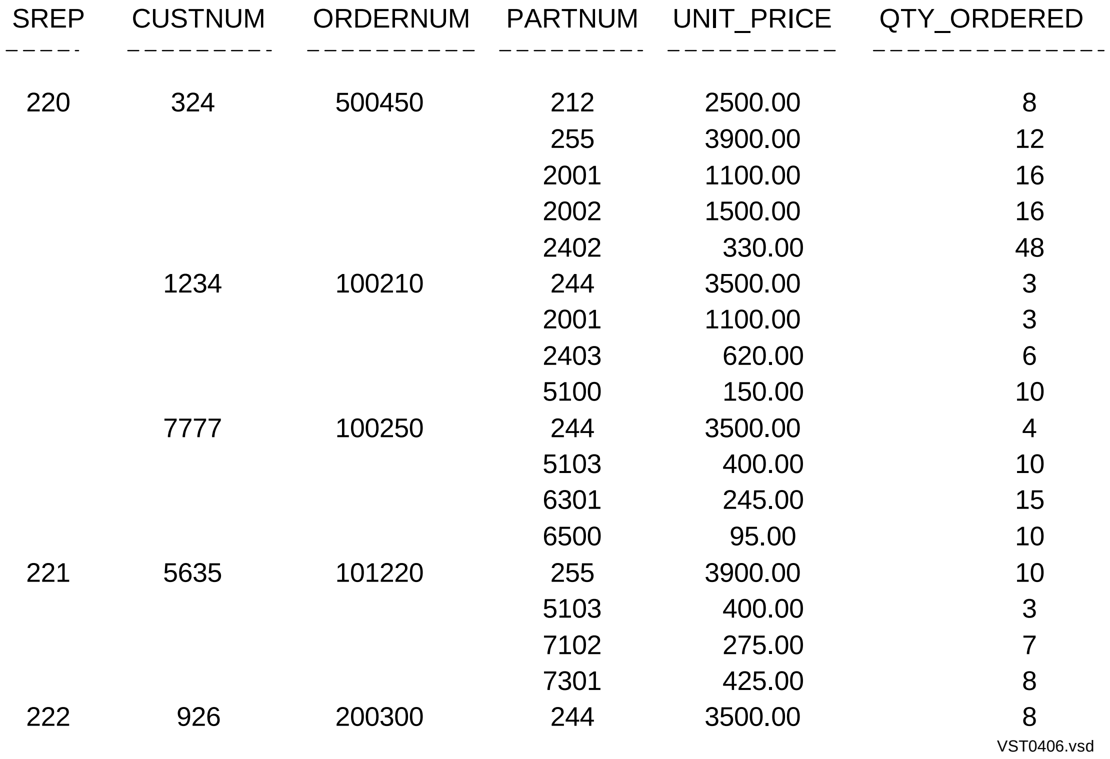
You can divide a break group into more break groups. The following example illustrates a report defined with three levels of break groups. The commands that define the report are:

```
>> SET LIST_COUNT 0;
>> VOLUME SALES;
>> SELECT SALESREP, C.CUSTNUM, R.ORDERNUM, PARTNUM,
+>      UNIT_PRICE, QTY_ORDERED
+> FROM CUSTOMER C, ORDERS R, ODETAIL OD
+> WHERE C.CUSTNUM = R.CUSTNUM
+>      AND R.ORDERNUM = OD.ORDERNUM
+> ORDER BY SALESREP, C.CUSTNUM, R.ORDERNUM, PARTNUM;
S> DETAIL SALESREP AS I4 HEADING "SREP",
+>        CUSTNUM,
+>        ORDERNUM AS I8,
+>        PARTNUM,
+>        UNIT_PRICE,
+>        QTY_ORDERED;
S> BREAK ON SALESREP, C.CUSTNUM, R.ORDERNUM;
S> LIST N 18;
```

Sales representatives form the major break group. The customers of each sales representative form the next level break group. The orders from each customer form the last break group. A title and footing can be printed at each level.

Figure 4-6 shows the resulting report.

Figure 4-6. Example of Break Groups



```
SREP   CUSTNUM    ORDERNUM   PARTNUM   UNIT_PRICE   QTY_ORDERED
----   --------   ---------  --------  ----------   -------------

 220      324      500450       212     2500.00          8
                                255     3900.00         12
                               2001     1100.00         16
                               2002     1500.00         16
                               2402      330.00         48
         1234      100210       244     3500.00          3
                               2001     1100.00          3
                               2403      620.00          6
                               5100      150.00         10
         7777      100250       244     3500.00          4
                               5103      400.00         10
                               6301      245.00         15
                               6500       95.00         10
 221     5635      101220       255     3900.00         10
                               5103      400.00          3
                               7102      275.00          7
                               7301      425.00          8
 222      926      200300       244     3500.00          8
```

VST0406.vsd

When you use the BREAK ON command, consider these points:

- You can specify only one BREAK ON command at a time. You can edit the command by using FC, replace the command by reentering it, delete the command by entering RESET REPORT BREAK ON, or modify the BREAK ON command by using RESET REPORT BREAK (`column-list`).
- The BREAK ON command must define the break groups at all levels. Specify the groups and subgroups from most inclusive to least inclusive.
- You should include an ORDER BY clause in the associated SELECT command to sort the break columns.
- You can specify a break column that is not in the detail line. The break occurs, but the item is not printed. For example, you might want to print the break column value in the break title but not in the detail line.
- If the break column is an item in the detail line, you can use the SUPPRESS option to ensure that the column value appears only in the first row of the group. If a page break occurs in the middle of the group, the value appears again in the first row of the new page. If you specify the NOSUPPRESS option, the column value appears in each detail line of the group.

Labeling Information

A report can contain various types of labels: column headings, titles, footings, line numbers, and text inserted between print items in an output line.

Column Headings

- You can choose to have default column headings or customized column headings in a report.
- If you use the default report format, default headings appear. The heading for a print item that is a column of a table or view is the heading that was specified when the table or view was created or altered. If no heading was specified, then the default heading is the column name. The default heading for a print item that is an expression is (EXPR). String literals and print items defined with an IF/THEN/ELSE clause or a CONCAT clause do not have default headings.
- You can suppress all headings in the report by setting the HEADINGS style option OFF. After you set HEADINGS OFF, all headings are suppressed in any report you produce until you set HEADINGS ON or you end your SQLCI session.
- You can specify your own headings in a DETAIL command. You can also specify that a print item has no heading.

Headings for Column Identifier Print Items

The report writer uses heading information in the following order of precedence to produce headings for report columns:

1. The HEADINGS option of the SET STYLE command enables or suppresses headings for the report. You can enable headings for the report by specifying HEADINGS ON or suppress headings by specifying HEADINGS OFF. The default is HEADINGS ON.
2. HEADING or NOHEAD clauses of the DETAIL command control headings for print items. You can specify a heading with the HEADING option for a print item in the detail list. You can also specify NOHEAD to suppress a heading for a print item.
3. Defined alias or detail names for the report provide headings of affected print items.
4. A heading that exists as part of the column definition of a table or view provides a heading for a print item specified by the column identifier. This heading is specified when the database administrator creates or alters a view or table and stores it in an SQL catalog. To display such a heading easily, select the column of the view or table and display the results in the default report format. If the column heading is different from the column name displayed by an INVOKE command entered in item 5, the column has a stored heading.

5. The column name is the default heading for a print item that is a column of a table or view. The column name is defined when the database administrator creates the table or view or adds the column to the table or view. To display column names, use the INVOKE command to display a table or view definition.

6. The heading EXPR is the default heading for expressions, functions, and numeric literals specified in the DETAIL command.

7. No heading is the default for string literals, IF/THEN/ELSE items, and CONCAT items specified in the DETAIL command.

The following example shows how you can specify a heading for a print item in a DETAIL command. The DETAIL command defines a heading for DEPTNUM, specifies no heading for DEPTNAME, and accepts the default heading for LOCATION.

```
>> SELECT * FROM PERSNL.DEPT
+> ORDER BY DEPTNUM;
S> DETAIL DEPTNUM HEADING "DEPARTMENT",
+>        DEPTNAME NOHEAD,
+>        LOCATION;
S> LIST NEXT 2;


DEPARTMENT LOCATION
---------- -----------

1000 FINANCE CHICAGO
1500 PERSONNEL CHICAGO
S>
```

The following example specifies a heading for the concatenated names of employees:

```
>> SELECT * FROM PERSNL.EMPLOYEE E, PERSNL.JOB
+> WHERE E.DEPTNUM = 4000
+>    AND E.JOBCODE = JOB.JOBCODE;
S> DETAIL EMPNUM HEADING "Employee No.",
+>        CONCAT (FIRST_NAME STRIP, SPACE 1, LAST_NAME)
+>           AS A25 HEADING "Name" CENTER,
+>        JOBDESC HEADING "Job Title" CENTER ;
S> LIST N 3;


Employee No. Name Job Title
------------ ------------------------- ------------------

65 RACHEL MCKAY MANAGER
87 ERIC BROWN SYSTEM ANALYST
104 DAVID STRAND SYSTEM ANALYST
S>
```

The previous detail line illustrates these important points:

- If the width of a heading is greater than the width of the print item value, the heading determines the field size. A numeric value is right-justified; a character value is left-justified. A NULL character value is left-justified.

- The width of the display field for a print item that is a table or view column is determined by the default display format for the data type of the column. You can override the default width by specifying an AS clause: for example, CONCAT (FIRST_NAME, SPACE 1, LAST_NAME) AS A25.

You cannot use an AS clause with columns of the INTERVAL data type. For more information about displaying totals and subtotals on columns of the INTERVAL data type, see Calculating Totals on page 4-49.

For more information about the AS clause and default display formats, see Formatting Data Values on page 4-28.

- A centered heading might not appear to be centered if the left-justified values are followed by blanks.

A report can include multiple-line headings. You indicate where the line breaks occur by using the current new-line character. The default NEWLINE_CHAR value is a slash (/).

The following commands define a supplier parts summary that contains multiple line headings. The command that selects the data follows:

```
>> SELECT *
+> FROM SALES.PARTS P, INVENT.PARTSUPP PS, INVENT.SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM AND PS.SUPPNUM = S.SUPPNUM
+> ORDER BY S.SUPPNUM, P.PARTNUM;
```

The following commands define the detail line and display the first row of output:

```
S> DETAIL P.PARTNUM HEADING "Part/Number" CENTER,
+>        QTY_AVAILABLE HEADING "Available/Units" CENTER,
+>        PARTCOST HEADING "Unit Cost/(dollars)" CENTER,
+>        PARTCOST * QTY_AVAILABLE
+>        HEADING "Total Cost/(dollars)" CENTER;
S> LIST N 1;

Part Available Unit Cost Total Cost
Number Units (dollars) (dollars)
------ ----------- ------------ --------------------

212 3525 2000.00 7050000.00
S>
```

You can specify a different new-line character as follows:

```
>> SET STYLE NEWLINE_CHAR "!";
```

The first line of the preceding DETAIL line must then be changed as follows:

```
S> DETAIL P.PARTNUM HEADING "Part!Number" CENTER,
```

Titles

A report can contain three types of titles:

- A page title appears at the top of each page of a report.
- A report title follows the page title on the first page of the report.
- A break title precedes the break group for which it is defined.

Only one REPORT TITLE and one PAGE TITLE command are in effect at a time. There can be one BREAK TITLE command for each item specified in a BREAK ON command. You can edit a command by using FC, replace a command by reentering it, or delete a command by using RESET REPORT.

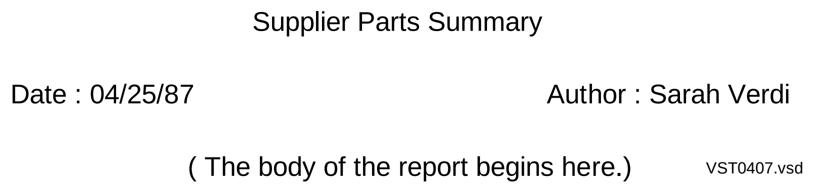
Page and Report Titles

The following commands produce a page and report title for the supplier parts summary:

```
S> PAGE TITLE "Supplier Parts Summary" CENTER;
S> REPORT TITLE "Date: ", CURRENT_TIMESTAMP AS DATE *,
+> TAB 40, "Author: Sarah Verdi";
```

The right margin is 65. The page title is centered. Titles on the first page appear as shown in Figure 4-7.

Figure 4-7. Example of Report and Page Titles



```
                    Supplier Parts Summary

Date : 04/25/87                          Author : Sarah Verdi

               ( The body of the report begins here.)
```

VST0407.vsd

A blank line follows the page title and the report title. The report title appears only on the first page of the report.

You can include columns from the select list, named print items from the DETAIL print list, string literals, and arithmetic expressions in a title. If you want to include in a title an unnamed item from the detail line (for example, an expression), you must define a detail alias name for the item.

The value used in the title is taken from the first output row on the page (for page titles) and the first output row of the report (for report titles). For example, the following commands specify that the page title include a customer number:

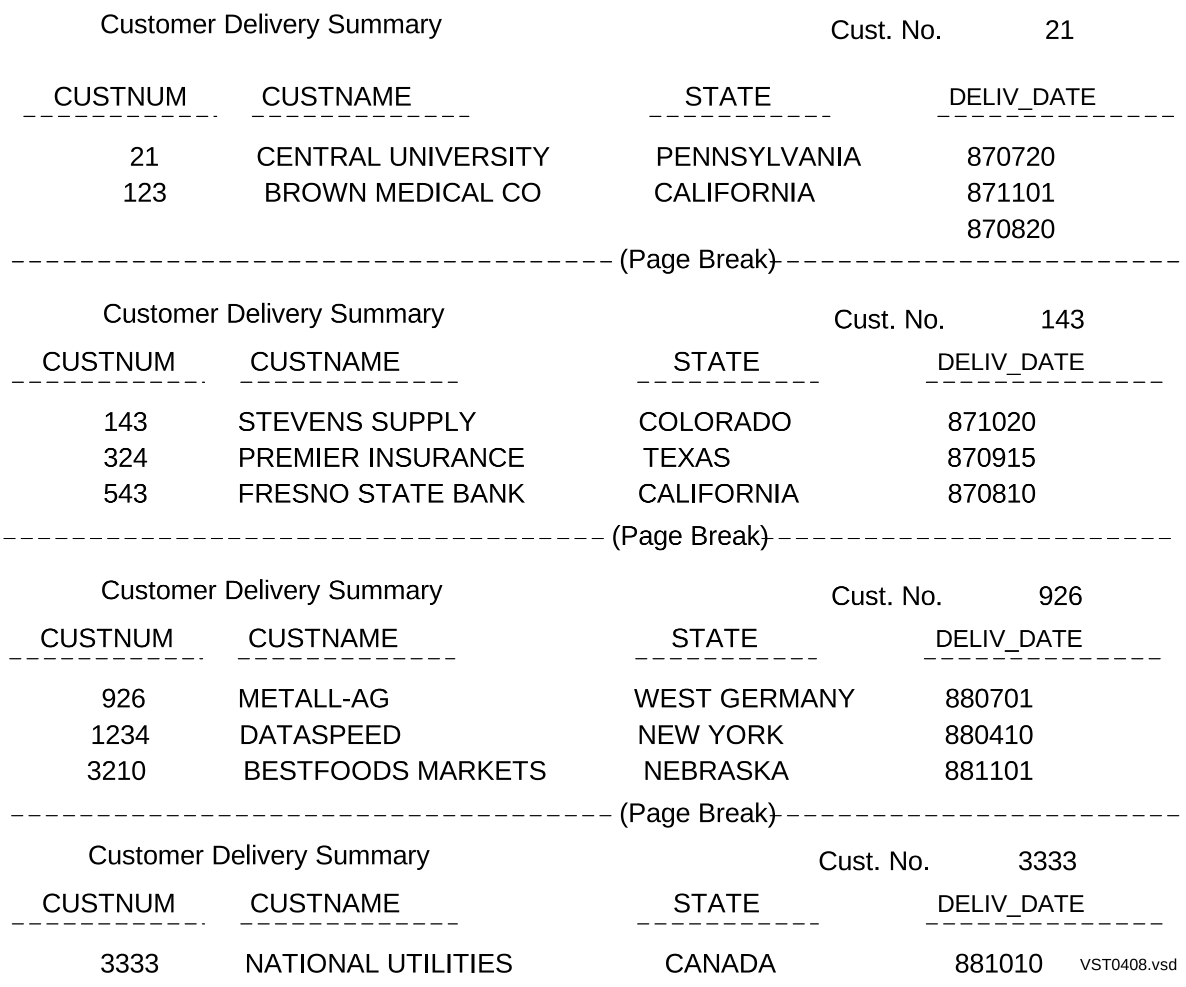
```
>> SELECT *
+> FROM SALES.CUSTOMER C, SALES.ORDERS O
+> WHERE C.CUSTNUM = O.CUSTNUM
+> ORDER BY C.CUSTNUM, ORDERNUM;
S> DETAIL C.CUSTNUM,
+>        CUSTNAME,
+>        STATE,
+>        DELIV_DATE;
S> BREAK ON C.CUSTNUM, CUSTNAME, CITY, STATE;
```

```
S> PAGE TITLE "Customer Delivery Summary ",
+>        TAB 45,
+>        "Cust. No.",
+>        C.CUSTNUM AS I4 ;
S> SET LAYOUT PAGE_LENGTH 10;
S> LIST N 10;
```

Figure 4-8 shows the first nine rows of report output, which appears on four pages. The customer number in each page title is taken from the first row of output on the page. The page length, which is ten lines, accommodates these items:

- The default top margin (a blank line)
- A single-line title followed by a blank line
- Two lines of headings followed by a blank line
- Three detail lines
- The default bottom margin (a blank line)

Figure 4-8. Example of a Page Title With a Row Value



```
      Customer Delivery Summary                    Cust. No.      21

  CUSTNUM      CUSTNAME                      STATE          DELIV_DATE
  -----------  -------------                 ----------     -------------

       21      CENTRAL UNIVERSITY            PENNSYLVANIA     870720
      123       BROWN MEDICAL CO             CALIFORNIA       871101
                                                              870820
------------------------------------ (Page Break) ------------------------
      Customer Delivery Summary                    Cust. No.      143
  CUSTNUM      CUSTNAME                      STATE          DELIV_DATE
  -----------  -------------                 ----------     -------------

      143      STEVENS SUPPLY                COLORADO         871020
      324      PREMIER INSURANCE              TEXAS           870915
      543      FRESNO STATE BANK             CALIFORNIA       870810
------------------------------------ (Page Break) ------------------------
      Customer Delivery Summary                    Cust. No.      926
  CUSTNUM      CUSTNAME                      STATE          DELIV_DATE
  -----------  -------------                 ----------     -------------

      926      METALL-AG                     WEST GERMANY     880701
     1234      DATASPEED                     NEW YORK         880410
     3210       BESTFOODS MARKETS             NEBRASKA        881101
------------------------------------ (Page Break) ------------------------
      Customer Delivery Summary                    Cust. No.      3333
  CUSTNUM      CUSTNAME                      STATE          DELIV_DATE
  -----------  -------------                 ----------     -------------

     3333      NATIONAL UTILITIES              CANADA          881010
```

VST0408.vsd

Break Titles

You can specify a break title for each break group. You can have more than one BREAK TITLE command in effect at a time, but each command must relate to a different break column.

In a break title, you can include columns from the select list, named print items from the DETAIL print list, string literals, and arithmetic expressions. If a column identifier is used as a print item in a BREAK TITLE command, the value for that item is taken from the first detail line of the break group.

You must enclose the print list of a BREAK TITLE command in parentheses.

You can delete the break title associated with each specified break column by using the RESET REPORT BREAK TITLE (`column-list`) command.

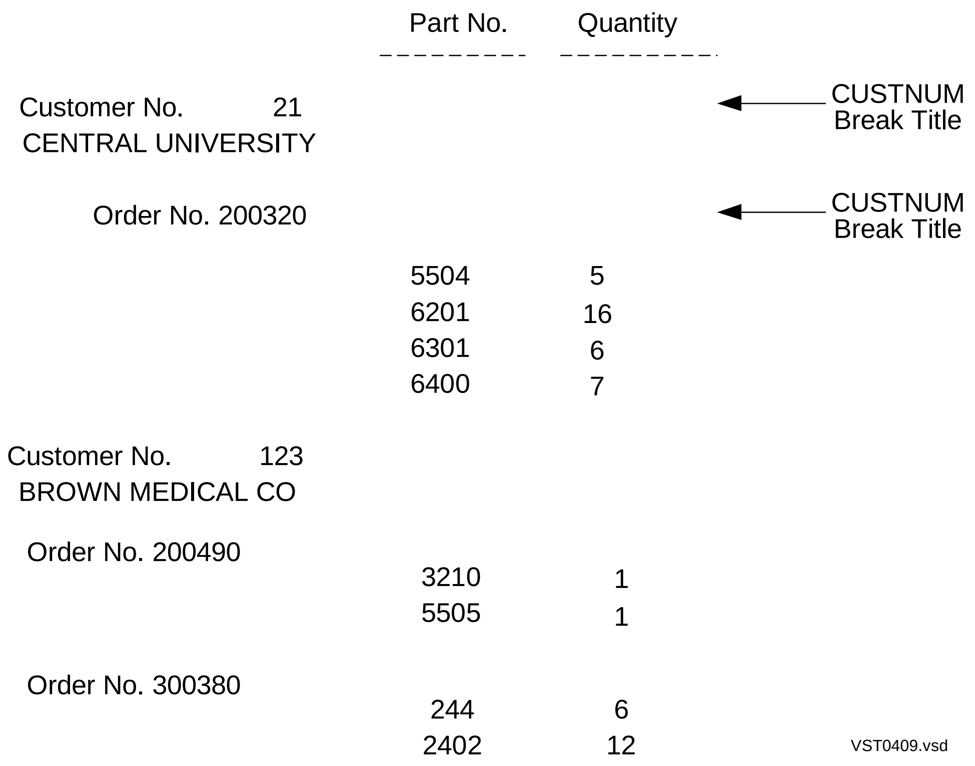
If you want to include an unnamed item from the detail line in a break title, you must define a detail alias name for the item.

In the following example, a report generates break titles for two break groups—one for each customer and one for each order. The CUSTNUM value in the first title is taken from the first row of each customer break group. The ORDERNUM value is taken from the first row of each order break group.

```
>> VOLUME SALES;
>> SELECT *
+> FROM CUSTOMER C, ORDERS R, ODETAIL OD
+> WHERE C.CUSTNUM = R.CUSTNUM
+>     AND R.ORDERNUM = OD.ORDERNUM
+> ORDER BY C.CUSTNUM, R.ORDERNUM;
S> DETAIL TAB 24,
+>        PARTNUM HEADING "Part No.",
+>        QTY_ORDERED HEADING "Quantity";
S> BREAK ON C.CUSTNUM, CUSTNAME, R.ORDERNUM;
S> BREAK TITLE C.CUSTNUM
+>        (SKIP 1,
+>        "Customer No.",
+>        C.CUSTNUM,
+>        SKIP 1,
+>        CUSTNAME );
S> BREAK TITLE R.ORDERNUM
+>        (TAB 5,
+>        "Order No.",
+>        R.ORDERNUM AS I8);
S> LIST N 8;
```

Figure 4-9 shows the first eight rows of the report.

Figure 4-9. Example of Break Titles



```
                              Part No.      Quantity
                             ---------     ----------

 Customer No.       21
  CENTRAL UNIVERSITY

       Order No. 200320

                              5504           5
                              6201          16
                              6301           6
                              6400           7

Customer No.       123
 BROWN MEDICAL CO

  Order No. 200490
                               3210          1
                               5505          1

  Order No. 300380
                               244           6
                               2402         12
```

Footings

A report can contain three types of footings:

- A page footing appears at the bottom of each page of a report.
- A report footing precedes the page footing on the last page of the report.
- A break footing follows the break group for which it is defined.

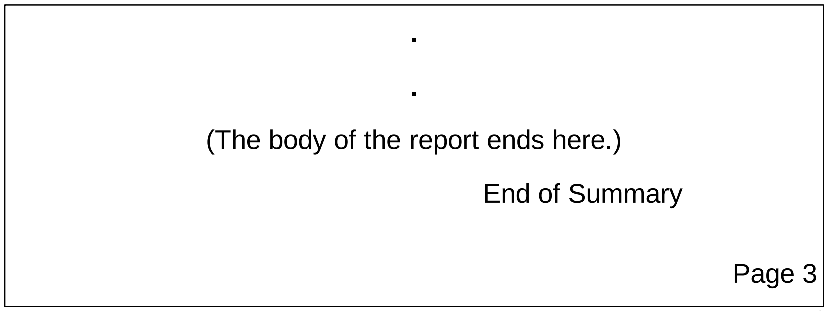
Page and Report Footings

The following commands produce a page and report footing:

```
   S> REPORT FOOTING "End of Summary" CENTER;
   S> PAGE FOOTING TAB 50, "Page ", PAGE_NUMBER AS I2 ;
```

The right margin is 65. The report footing is centered. Footings on the last page appear as shown in Figure 4-10.

Figure 4-10. Location of Report and Page Footings



.

.

(The body of the report ends here.)

End of Summary

Page 3

VST0410.vsd

A blank line separates the body of the report from the page footing. On the last page, the report footing is separated from the body of the report by a blank line. The other pages of the report do not contain the report footing.

You can include columns from the select list, named print items from the DETAIL print list, string literals, and arithmetic expressions in a footing. If you want to include in a footing the value of an unnamed item from the detail line, you must define a detail alias name for the item.

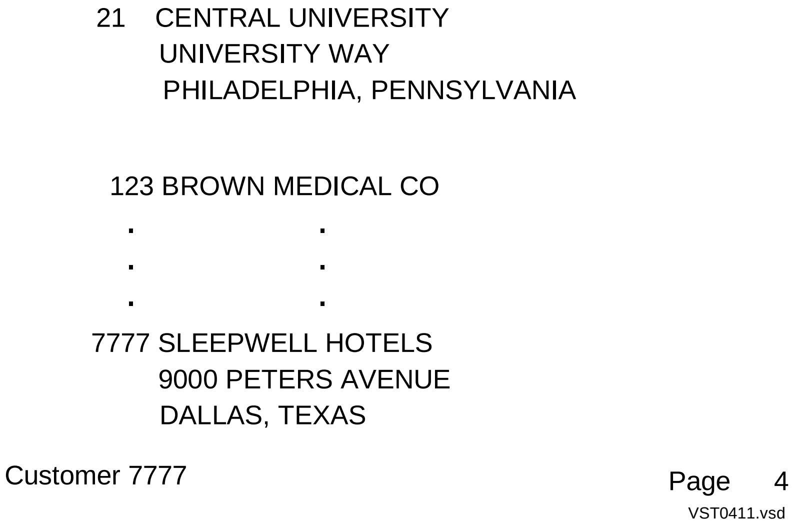
There is only one REPORT FOOTING and one PAGE FOOTING command in effect at a time. You can edit a command by using FC, replace a command by reentering it, or delete a command by using RESET REPORT.

If a column identifier is used as a print item in a footing, the value for that item is taken from the last detail line on the page (for page titles) and the last detail line of the report (for report titles). For example, the following commands specify that the page footing includes the customer number:

```
>> SET STYLE HEADINGS OFF;
>> SELECT *
+> FROM SALES.CUSTOMER
+> ORDER BY CUSTNUM;
S> DETAIL CUSTNUM, CUSTNAME, SKIP 1,
+>        TAB 10, STREET, SKIP 1,
+>        TAB 10, CONCAT (CITY STRIP, ", ", STATE),
+>        SKIP 1;
S> PAGE FOOTING "Customer ", CUSTNUM, TAB 50,
+>        "Page ", PAGE_NUMBER AS I2 ;
S> LIST ALL;
```

Figure 4-11 shows the report. The customer number is taken from the last output row on the page.

Figure 4-11. Example of a Page Footing



```
    21  CENTRAL UNIVERSITY
         UNIVERSITY WAY
          PHILADELPHIA, PENNSYLVANIA


     123 BROWN MEDICAL CO
       .            .
       .            .
       .            .
    7777 SLEEPWELL HOTELS
         9000 PETERS AVENUE
         DALLAS, TEXAS

 Customer 7777                                   Page    4
```

VST0411.vsd

Break Footings

You can specify a break footing for each break group. You can have more than one BREAK FOOTING command in effect at a time, but each command must relate to a different break column.

You can include columns from the select list, named print items from the DETAIL print list, string literals, and arithmetic expressions in a break footing. If a column identifier is used as a print item in a BREAK FOOTING command, the value for that item is taken from the last detail line of the break group.

You must enclose the print list of a BREAK FOOTING command in parentheses.

You can delete the break title associated with each specified break column by using the RESET REPORT BREAK FOOTING (`column-list`) command.

If you want to include an unnamed item from the detail line in a break footing, you must define a detail alias name for the item.

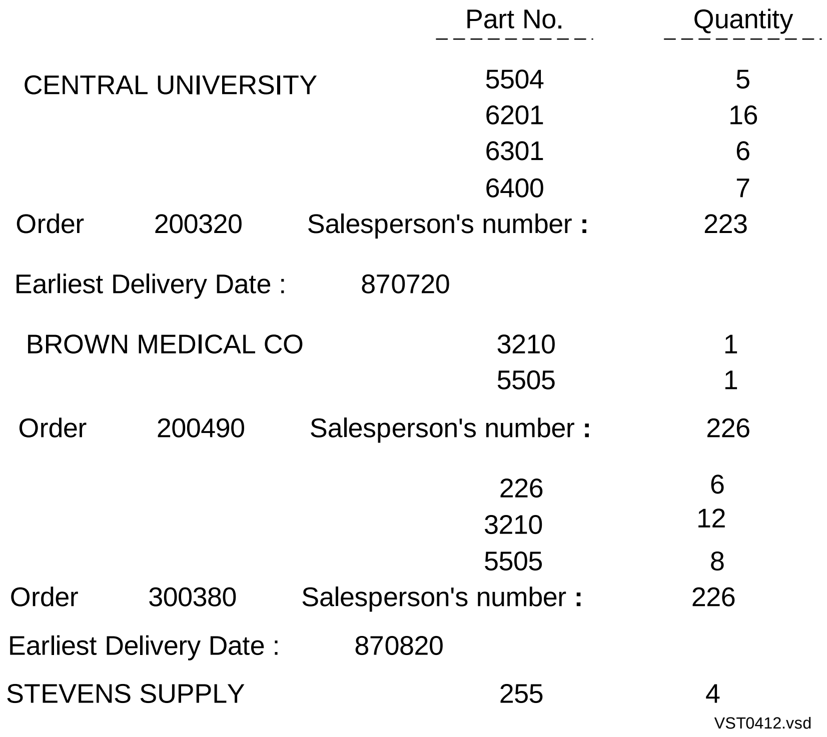
In the following example, a report generates break footings for two break groups—one for each customer and one for each order:

```
>> VOLUME SALES;
>> SELECT *
+> FROM CUSTOMER C, ORDERS R, ODETAIL OD
+> WHERE C.CUSTNUM = R.CUSTNUM
+>     AND R.ORDERNUM = OD.ORDERNUM
+> ORDER BY C.CUSTNUM, R.ORDERNUM, DELIV_DATE DESC;
S> DETAIL CUSTNAME NOHEAD, SPACE 5,
+>        PARTNUM HEADING "Part No.",
+>        QTY_ORDERED HEADING "Quantity";
S> BREAK ON C.CUSTNUM, CUSTNAME, R.ORDERNUM;
S> BREAK FOOTING C.CUSTNUM
```

```
+>         ("Earliest Delivery Date: ",
+>         DELIV_DATE AS I6,
+>         SKIP 1) ;
S> BREAK FOOTING R.ORDERNUM
+>         ("Order", R.ORDERNUM,
+>         SPACE 3,
+>         "Salesperson's number: ",
+>         SALESREP,
+>         SKIP 1) ;
S> LIST NEXT 10;
```

Figure 4-12 shows the report. The DELIV_DATE value in the first footing is taken from the last row of each customer break group. The ORDERNUM value is taken from the last row of each order break group.

Figure 4-12. Example of Break Footings



```
                                      Part No.       Quantity
                                     ----------     ----------

  CENTRAL UNIVERSITY                   5504              5
                                       6201             16
                                       6301              6
                                       6400              7
 Order      200320    Salesperson's number :          223

 Earliest Delivery Date :       870720

  BROWN MEDICAL CO                      3210            1
                                        5505            1

 Order      200490    Salesperson's number :          226

                                        226            6
                                       3210           12
                                       5505            8
 Order      300380    Salesperson's number :         226
 Earliest Delivery Date :       870820
 STEVENS SUPPLY                         255            4
```

VST0412.vsd

Line Numbers

The report writer provides a LINE_NUMBER function for numbering the lines of a report. You can use one sequence for the whole report or restart the numbering at a new page or at a new break group.

The line number is incremented at each detail line. If the detail line is printed as multiple output lines, the number is incremented only once for each set of output lines.

The default display format for the LINE_NUMBER value is I11. To number all detail lines in one sequence, include the print item:

```
LINE_NUMBER OVER REPORT
```

or

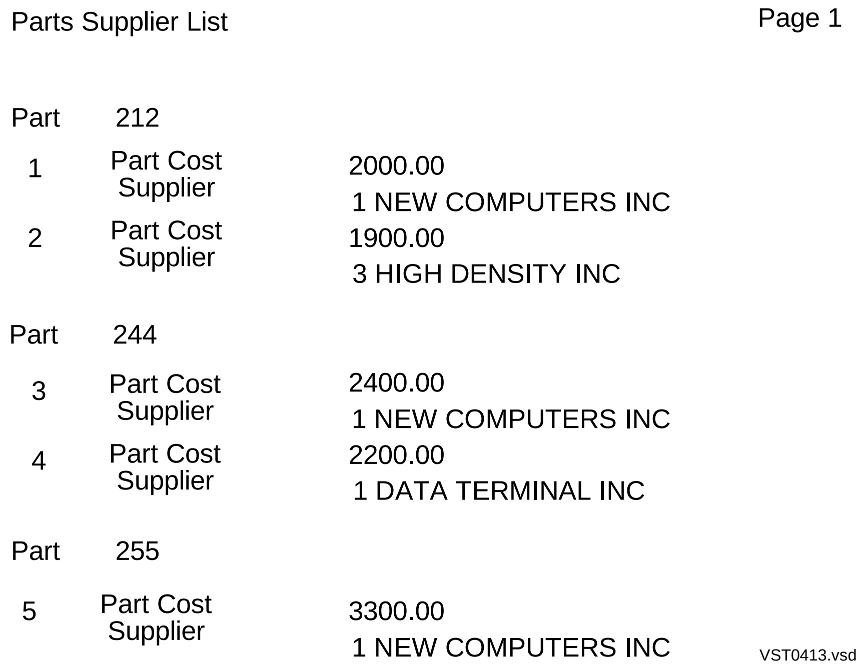
```
LINE_NUMBER
```

Suppose you want to number the entries in a listing of parts and the suppliers who supply them. The numbering sequence is to be restarted at each new page. The following commands produce the list:

```
>> SELECT * FROM INVENT.SUPPLIER, INVENT.PARTSUPP
+> WHERE SUPPLIER.SUPPNUM = PARTSUPP.SUPPNUM
+> ORDER BY PARTNUM, SUPPLIER.SUPPNUM ;
S> SET STYLE HEADINGS OFF;
S> DETAIL LINE_NUMBER OVER PAGE AS I4,
+>        TAB 7,
+>        "Part Cost",
+>        PARTCOST ,
+>        SKIP 1,
+>        TAB 7,
+>        "Supplier",
+>        SUPPLIER.SUPPNUM,
+>        SUPPNAME;
S> BREAK ON PARTNUM;
S> BREAK TITLE PARTNUM ("Part ", PARTNUM);
S> TITLE "Parts Supplier List", TAB 48,
+>       "Page ", PAGE_NUMBER AS I3;
S> LIST NEXT 5;
```

Figure 4-13 shows the report.

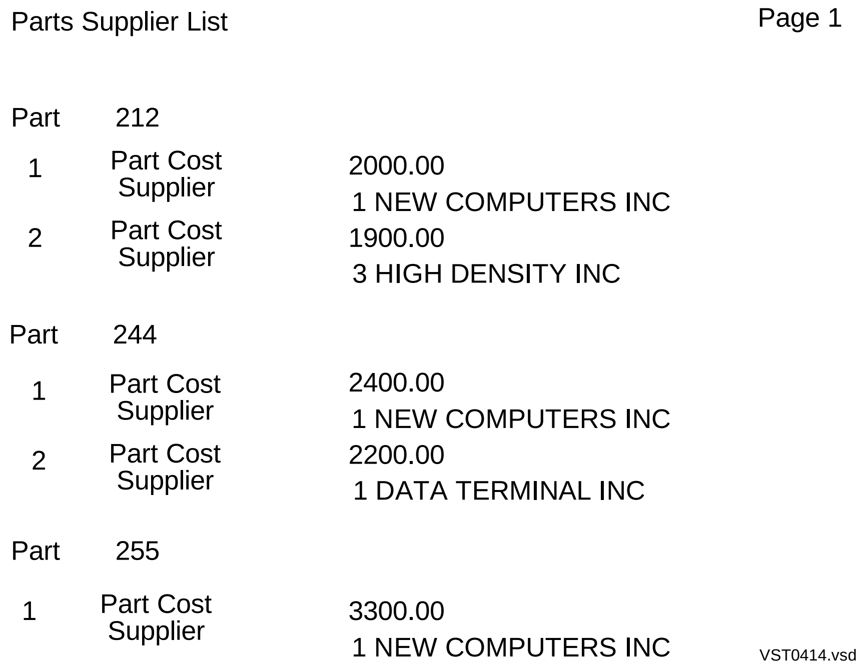
Figure 4-13. Example of Numbering Lines



```
Parts Supplier List                                      Page 1


Part      212
  1        Part Cost         2000.00
           Supplier           1 NEW COMPUTERS INC
  2        Part Cost         1900.00
           Supplier           3 HIGH DENSITY INC

Part      244
  3        Part Cost         2400.00
           Supplier           1 NEW COMPUTERS INC
  4        Part Cost         2200.00
           Supplier           1 DATA TERMINAL INC

Part      255
  5       Part Cost          3300.00
          Supplier            1 NEW COMPUTERS INC         VST0413.vsd
```

If you change the previous DETAIL command to specify LINE_NUMBER OVER PARTNUM AS I4, the line number is reset at the beginning of each break group as shown in Figure 4-14.

Figure 4-14. Example of Numbering Break Group Lines



```
Parts Supplier List                                      Page 1


Part      212
  1        Part Cost         2000.00
           Supplier           1 NEW COMPUTERS INC
  2        Part Cost         1900.00
           Supplier           3 HIGH DENSITY INC

Part      244
  1        Part Cost         2400.00
           Supplier           1 NEW COMPUTERS INC
  2        Part Cost         2200.00
           Supplier           1 DATA TERMINAL INC

Part      255
  1       Part Cost          3300.00
          Supplier            1 NEW COMPUTERS INC         VST0414.vsd
```

Text Inserted into a Line

You can insert text into a detail line, title, or footing by specifying a string literal as a print item. Figure 4-13 illustrates this technique. You can use the technique to create the following effects:

- Insert commas between concatenated items.
- Insert descriptions of an item that precedes or follows the string literal. For example, the DETAIL, BREAK TITLE, and TITLE commands in the preceding report definition include descriptive string literals.

You can also insert text in a line with AS clause decorations. For more information, see Filler Characters on page 4-38.

Formatting Data Values

You can use the AS clause to specify a display format for a print item that is a column identifier, literal, arithmetic expression, or CONCAT clause. If you do not specify an AS clause, or if you use the default report definition to display your data, the report writer uses the default display formats shown in Table 4-1.

The data type of a table column is defined in the catalog that contains the description of the table.

Table 4-1. Default Display Formats (page 1 of 2)

SQL Data Type	Default Display Format
Fixed-length Characters:	An (or A255 if n > 255)
CHAR(n)	
PIC X(n) DISPLAY	
NATIONAL CHAR(n)	
NCHAR(n)	
Variable-length Characters:	
VARCHAR(n)	
NATIONAL CHARACTER VARYING (n)	
NCHAR VARYING(n)	
NUMERIC(1,s) — NUMERIC (4,s)	I6 (or F7.s if s > 0)
NUMERIC(1,s) UNSIGNED — NUMERIC(4,s) UNSIGNED	I5 (or F6.s if s > 0)
NUMERIC(5,s) — NUMERIC(9,s)	I11 (or F12.s if s > 0)
NUMERIC(5,s) UNSIGNED — NUMERIC(9,s) UNSIGNED	I10 (or F11.s if s > 0)
NUMERIC(10,s) — NUMERIC(18,s)	I20 (or F21.s if s > 0)

Table 4-1. Default Display Formats (page 2 of 2)

SQL Data Type	Default Display Format
DECIMAL (n,s)	In+1 (or Fn+2.s if s > 0)
DECIMAL (n,s) UNSIGNED	In (or Fn+1.s if s > 0)
PIC S9(i) [V9(s)]	Ii+1 (or Fi+2.s if s > 0)
PIC 9(i) [V9(s)]	Ii (or Fi+1.s if s > 0)
PIC SV9(s)	Fs+2.s
PIC V9(s)	Fs+1.s
FLOAT (where precision is 1 through 22)	E14.7
FLOAT (where precision is 23 through 54)	E24.17
REAL	E14.7
DOUBLE PRECISION	E24.17
n	Length
vw	Current VARCHAR_WIDTH value
s	Scale of the data type
i	Integer
A, I, F	For more information see Display Format Specifications on page 4-30.
E	Edit descriptor, explained in the *Guardian Programmer's Guide*
DATE	yyyy-mm-dd
DATETIME	yyyy-mm-dd:hh:mm:ss.msssss
TIME	hh:mm:ss
TIMESTAMP	yyyy-mm-dd:hh:mm:ss.msssss
INTERVAL	yyyy-mm or dd:hh:mm:ss.msssss
yyyy	Years
-mm	Months
dd	Days
hh	Hours
:mm	Minutes
ss	Seconds
msssss	Microseconds

You might decide that the default display produces a format that does not suit your needs. You can add an AS clause to the print item in the DETAIL command to override the default format for any data type except DATE, DATETIME, INTERVAL, TIME, and TIMESTAMP. You can use the AS clause to modify the field width, adjust the scale of a number, insert special symbols called decorations, and change the justification of a character value from left to right.

Display Format Specifications

A display format specification can include several elements: display descriptors, scalesign descriptors, decorations, and modifiers.

- The essential part of a display format is the *`display-descriptor`*. A display descriptor controls the width of a field and, for numeric values, the scale of the value.

  The display descriptors for character values follow:

  `A[ w]` specifies alphanumeric display of width `w`; for example, A20 displays characters in a field that is 20 single-byte print positions wide.

  `C n[. w]` specifies alphanumeric display for long character values such as VARCHAR values. This format allows you to display a value on `n` lines in a field of width `w`; for example, C30.10 displays 30 single-byte characters, 10 characters per line.

  If you do not include `w`, the C format is the same as the A format.

  Values are left justified in A and C formats.

  Note. If you are defining a report that contains double-bye characters, see Printing Double-Byte Characters on page 4-58 for special considerations regarding using display descriptors.

  The display descriptors for numeric values follow:

  `F w. d[. m]` specifies fixed-point display of width `w`. You can specify the number of significant digits to the right of the decimal point with `d`. You can specify the number of digits to the left of the decimal point with `m`. For example, F10.2.6 specifies that all values are displayed with 2 digits following the decimal point and 6 digits preceding it. The field width is 10.

  `I w[. m]` specifies integer display of width `w`. You can specify the number of required digits with `m`, which might result in leading zeros. For example, I6.4 specifies that all values are displayed as integers with a field width of 6 and 4 required digits.

  M *mask* specifies a template, enclosed in angle brackets, apostrophes, or quotes, for displaying data. You can insert characters and specify whether digits are to be included in the value. Masks are useful for monetary values, dates, and suppression of leading and trailing zeros. For example, M<9,999> inserts a comma in the thousandths place.

  Values are right justified in F and I descriptor and date-time formats.

- For numeric values, you can include a *`scale-sign-descriptor`* to specify a scale factor.

- You can specify `decorations`; these are character strings inserted at a specified location in the value depending upon whether the value is negative, positive, zero, or too large for the field. You can specify multiple conditions. For more information on examples of using decorations, see Monetary Values on page 4-32.

- You can specify `modifiers` to control justification, insert filler characters, and specify an overflow character for the print item. A `modifier` applies to a single print item and overrides report defaults. For example, you can specify a special overflow character for one print item and use the default OVERFLOW_CHAR value for other print items.

Numeric Values

For numeric values, you can use the I and F display descriptors. In the following example, QTY_RECEIVED appears in the detail line. The data type defined for QTY_RECEIVED is NUMERIC (7); the value is always an integer. By default, the display format is I11. The DETAIL command specifies a narrower field:

```
>> SELECT * FROM INVENT.PARTSUPP;
S> DETAIL PARTNUM,
+> QTY_RECEIVED AS I6 HEADING "QTY" CENTER,
+> PARTCOST * QTY_RECEIVED HEADING "TOTAL COST" CENTER;
S> LIST NEXT 2;

PARTNUM   QTY          TOTAL COST
------- ------ --------------------

    212     20             40000.00
    212     35             66500.00

S>
```

The following DETAIL command includes an expression in COL 4 that results in a fixed-point number. The AS clause specifies a field 10 single-byte characters wide with 4 digits appearing to the right of the decimal point. This example uses a temporary table created under Row Value as Percent of All Row Values on page 3-35.

```
>> SELECT E.DEPTNUM, EMPNUM, SALARY, SALARY/AVGSAL*100.00
+> FROM PERSNL.EMPLOYEE E, TEMPTABS.AVGTEMP A
+> WHERE E.DEPTNUM = A.DEPTNUM;
S> DETAIL DEPTNUM, EMPNUM,
+>        SALARY AS F10.2,
+>        COL 4 AS F10.4 HEADING "PCT OF AVG";
S> LIST N 6;

DEPTNUM EMPNUM     SALARY  PCT OF AVG
------- ------ ----------  ----------

   9000      1  175500.00    165.1765
   1000     23  137000.00    263.4617
   3000     29  136000.00    240.1766
   2000     32  138000.40    276.0008
   3200     39   75000.00    211.6970
   3100     43   90000.00    225.8809
S>
```

In a DETAIL command print item, the AS clause must precede the HEADING and NAME options.

In the next example, the scale-sign descriptor specifies a scale factor for a very large number. The total cost value is scaled to thousands of dollars. The descriptor -3P specifies a scale factor of 10** -3 (or 0.001).

You cannot specify a scale-sign descriptor for an I display descriptor.

```
 >> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST
+> FROM SALES.PARTS P, INVENT.PARTSUPP PS
+> WHERE P.PARTNUM = PS.PARTNUM
+> ORDER BY P.PARTNUM;
S> DETAIL PARTNUM HEADING "Part No.",
+>        PARTCOST AS F10.2 HEADING "Part Cost/(dollars)",
+>        PARTCOST * QTY_AVAILABLE AS "-3P F10.2"
+>            HEADING "Total Cost/(thousands/of dollars)",
+>        PARTCOST * QTY_AVAILABLE AS F15.2
+>            HEADING "Total Cost/Without/Scale Factor";
S> LIST NEXT 4;

Total Cost Total Cost
Part Cost (Thousands Without
Part No. (Dollars)  of Dollars) Scale Factor
-------- --------- ----------- ---------------

212        2000.00   7050.00    7050000.00
212        1900.00   6697.50    6697500.00
244        2400.00   10622.40  10622400.00
244        2200.00   9737.20    9737200.00

S>
```

Monetary Values

If you want to include a symbol such as the dollar sign in a monetary value, you can use a mask or decoration. A mask or decoration inserts the symbol at the same print position on the output line regardless of the value of the item.

This example applies a mask to insert a dollar sign:

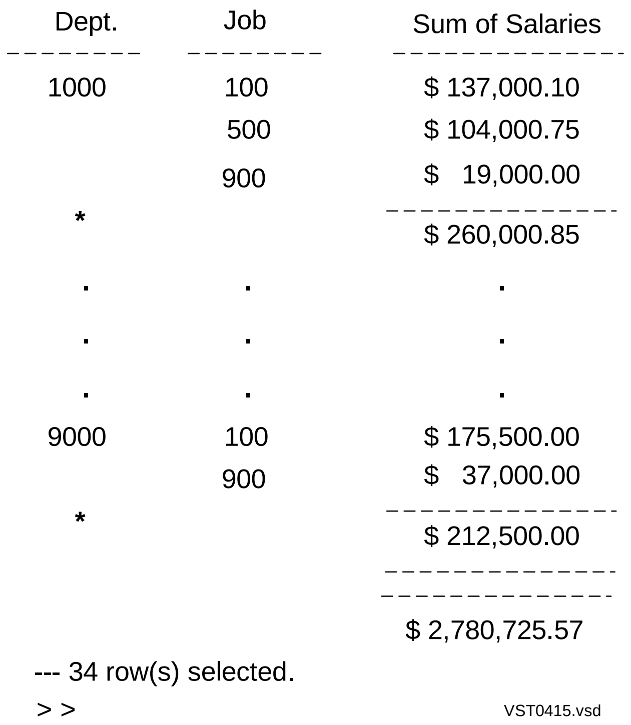
```
>> SELECT JOBCODE, AVG(SALARY)
+> FROM PERSNL.EMPLOYEE
+> GROUP BY JOBCODE;
S> DETAIL JOBCODE AS I4 HEADING "Job",
+>        COL 2 AS M<$ZZZ,ZZZ.99> HEADING "Avg. Sal.";
S> LIST NEXT 3;

Job    Avg. Sal.
---- -----------

100  $105,954.59
200  $ 24,000.00
250  $ 28,000.06
S>
```

If you calculate any totals or subtotals, the mask must accommodate the resulting values. If the previous query is modified to calculate the sum of salaries for each job code within each department and to produce subtotals and totals, the mask must be modified to accommodate larger numbers in the detail line, and large numbers in the subtotal and total lines.

```
>> SELECT DEPTNUM, JOBCODE, SUM(SALARY)
+> FROM PERSNL.EMPLOYEE
+> ORDER BY DEPTNUM
+> GROUP BY DEPTNUM, JOBCODE;
S> DETAIL DEPTNUM AS I6 HEADING "Dept.",
+>        JOBCODE AS I4 HEADING "Job",
+>        COL 3 AS M<$ZZZ,ZZZ,ZZZ.99>
+>              HEADING "Sum of Salaries";
S> BREAK ON DEPTNUM ;
S> SUBTOTAL COL 3 OVER DEPTNUM ;
S> TOTAL COL 3;
S> LIST ALL ;
```

Figure 4-15 shows the report.

Figure 4-15. Example of Monetary Total and Subtotals



```
   Dept.       Job        Sum of Salaries
 ---------  ---------    ---------------
   1000        100        $ 137,000.10
               500        $ 104,000.75
               900        $   19,000.00
                         ---------------
    *                     $ 260,000.85
     .           .             .
     .           .             .
     .           .             .
   9000        100        $ 175,500.00
               900        $   37,000.00
                         ---------------
    *                     $ 212,500.00
                         ---------------
                         ---------------
                        $ 2,780,725.57
 --- 34 row(s) selected.
 > >
```

If you want the dollar sign adjacent to the value, you can modify the original query by adding a PF modifier and by revising the mask descriptor:

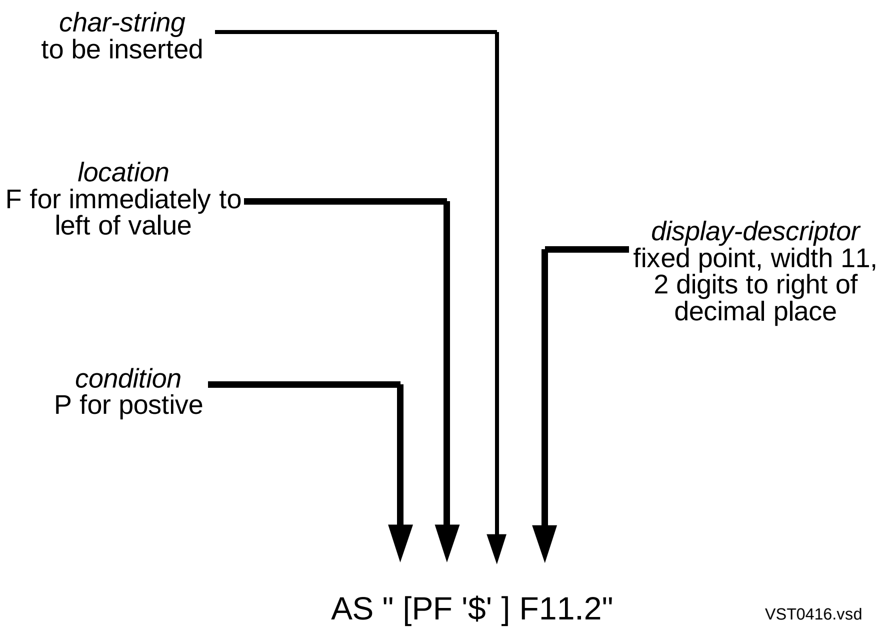
```
>> SELECT JOBCODE, AVG(SALARY)
+> FROM PERSNL.EMPLOYEE
+> GROUP BY JOBCODE;
S> DETAIL JOBCODE AS I4 HEADING "Job",
+> COL 2 AS "[PF'$'] M<Z,ZZZ,ZZZ.99>"
+> HEADING "Avg. Sal.";
S> LIST NEXT 5;


Job    Avg. Sal.

---- -----------

100  $105,954.59
200  $24,000.00
250  $28,000.06
300  $31,123.05
400  $77,400.00
S>
```

Figure 4-16 interprets the display format specified for COL 2 in the previous query. The form of `decoration` is:

```
condition location char-string
```

The entire display format must be enclosed in quotation marks. The decoration must be enclosed in square brackets.

Figure 4-16. Example of a Decoration



The previous decoration specification assumes that all values will be positive. If you expect negative or zero values, you must expand the decoration as shown:

```
AS "[MA1'CR',MPF'$',ZA1'                ' ] F13.2"
```

In this clause, three decorations apply to the value:

- In the first decoration, *`condition`* M specifies that this decoration applies if the value is negative; *`location`* A1 specifies print position 1 for printing *`charstring`* CR.
- In the second decoration, *`condition`* MP specifies that this decoration applies if the value is negative or positive; *`location`* and *`char-string`* are the same as those shown in Figure 4-16.
- In the third decoration, *`condition`* Z specifies that this decoration applies if the value is zero; *`location`* A1 specifies print position 1 for printing *`char-string`*, which is all blanks.

For character print items, you can specify only *`condition`* P.

The following list shows the result of applying the preceding AS clause to different values:

Value	Result
-100	CR $ 100.00
4500	$ 4500
0	

You can use the BZ modifier instead of the third decoration to specify that you want the field left blank if the value is zero:

```
AS "[MA1'CR',MPF'$' BZ ] F13.2"
```

You can insert the European decimal character in monetary values by changing the default decimal character. For more information, see Redefining Special Characters on page 4-48.

Suppressed Leading or Trailing Zeros

Leading zeros are suppressed automatically when you specify the *`F w. d[. m]`* display descriptor and omit *`m`*. For example, F10.2 suppresses leading zeros, but F10.2.5 prints leading zeros if the number of significant digits to the left of the decimal point is less than 5.

Descriptor	Result
F10.2	355.67
F10.2.5	00.355.67

The `I w[. m]` display descriptor also suppresses leading zeros if you omit *m*. For example, I8 does not print leading zeros but I8.6 does if the integer consists of fewer than 6 digits.

Descriptor	Result
18	355
18.6	000355

In a mask, you can use the uppercase character Z to suppress leading and trailing zeros.

Mask	Value	Result
M<ZZZ,ZZ9.99>	2.453	2.45
	9023.00	9023.00
M<9,999>	5432	5,432
	300	0,300
	0	0,000
M<ZZZZ>	300	300
M<$ZZZ9.99>	5432	$5432.00
	300.153	$ 300.15
	0	$ 0.00

You can use the BZ modifier to suppress all zeros when the value is zero: for example, AS "[BZ] M<ZZZZ>". If you include a modifier, you must enclose the modifier and display descriptor in quotation marks.

For information about replacing suppressed leading zeros with a character, see Filler Characters on page 4-38.

You can also suppress trailing zeros by truncating the value.

Text Concatenation

The CONCAT clause is useful for formatting names of persons or places. You can concatenate parts of names stored in separate columns of a table and insert characters between the concatenated values.

In the EMPLOYEE table, FIRST_NAME and LAST_NAME are stored in separate columns. You can define a print item of a detail line to concatenate the names in two ways:

```
CONCAT (LAST_NAME STRIP, ", ", FIRST_NAME)
```

or

```
CONCAT (FIRST_NAME STRIP, SPACE 1, LAST_NAME)
```

The first clause produces BENEDETTI, JULIO, and the second clause produces JULIO BENEDETTI. You can use a similar technique to combine city and state or to combine addresses in one field.

If you specify STRIP, the report writer strips trailing blanks from the value before concatenating it. The default width of the concatenated item is the sum of the original column widths before the blanks are stripped plus the width of any inserted strings.

If you want to specify the width of the field, include an AS clause in the print item:

```
CONCAT (FIRST_NAME STRIP, SPACE 1, LAST_NAME) AS A25
```

Single Items on Multiple Lines

You can print a variable-length character value that contains more characters than fit on a single line by using the C display descriptor of the AS clause. For example, suppose the JOB table contains a third column named JOBRESP of data type VARCHAR, which can contain 200 single-byte characters.

In the following example, suppose the left margin is 5 and the right margin is 50. The DETAIL line is defined as follows:

```
S> DETAIL JOBDESC HEADING “Job Title“,
+> JOBCODE HEADING “Job Code“, SKIP 2,
+> JOBRESP NOHEAD AS "[F]C0.45";
```

The JOBRESP characters start on a new line with up to 45 characters appearing on a line. The F modifier preceding the C display descriptor specifies splitting the value at a blank if possible. The headings and first detail line of the report are as follows:

```
Job Title            Job Code
------------------- -----------

ENGINEER               AA03
```

Designs and develops integrated circuits and other electronic products, assists in planning future projects, writes specifications, and oversees product testing.

For descriptions of the C display descriptor and F modifier, see Display Format Specifications on page 4-30.

Truncated Values

The VARCHAR_WIDTH style option specifies the maximum number of characters that can appear in a print item with a value of a variable-length character data type. The excess characters are truncated. The default is 80 single-byte characters. You can set a value up to 255 single-byte (or 177 double-byte) characters; for example:

```
>> SET STYLE VARCHAR_WIDTH 120;
```

If you set the width to a value greater than the number of characters on a line, you should format the variable-length print items as described in, Single Items on Multiple Lines on page 4-37.

Values of fixed-length character data types are truncated if they do not fit in the display format you specify.

Note. If you are defining a report that contains double-byte characters, see Printing Double-Byte Characters on page 4-58 for special considerations regarding truncating values.

By default, numeric values are not truncated. The report writer fills a field of a numeric print item that is too large for its display format with overflow characters.

The overflow character is determined by the setting of the OVERFLOW_CHAR style option.

You can truncate the decimal part of a fixed-point number by specifying the number of significant digits to the right of the decimal point.

For example, F10.2 truncates any digits beyond the first two following the decimal point.

Filler Characters

The report writer uses the following types of filler characters:

- If a numeric value is too large for the specified display format, the report writer fills the field with the default overflow character, which is an asterisk (*). You can change the overflow character by setting another single character; for example, a pound sign:

  ```
  >> SET STYLE OVERFLOW_CHAR "#";
  ```

  You can also override the default overflow character by including the OC modifier in the display format for a particular print item. For example, the following AS clause specifies that the item is to be filled with plus signs when overflow occurs:

  ```
  AS "[OC'+'] F8.2"
  ```

- The FL modifier of display formats specifies a character to be used to fill a field in the following circumstances:
  - A value in an A descriptor field does not fill the field. For example, "[FL'*'] A12" produces FINANCE*****.
  - Leading zeros are to be replaced. For example, "[FL'*']I6" produces ***355.

  - ○ Embedded text in a mask descriptor is not printed because the digits that surround the text are not printed. For example, "[FL'*']M<$ZZZZ9.99>" produces $**355.67.

- Decorations in a display format specify characters to be inserted in a print item value depending on whether the value is positive, negative, zero, or too large for the field.

  For a general description of decorations, see Monetary Values on page 4-32.

The following DETAIL command specifies that leading zeros in the price field are to be suppressed and replaced by asterisks. Trailing blanks in the part description field are to be filled with periods.

```
 >> SELECT * FROM SALES.PARTS;
S> DETAIL PARTNUM,
+>        PARTDESC AS "[FL '.'] A20",
+>        PRICE AS "[FL '*'] F8.2",
+>        QTY_AVAILABLE AS I5 HEADING "QTY";
S> LIST NEXT 4;


PARTNUM   PARTDESC             PRICE   QTY
------- -------------------- -------- -----

2402     DAISY PRINTER,T1 .. **350.00  4425
2403     DAISY PRINTER,T2 .. **650.00  3312
3103     LASER PRINTER, X1 ..*4200.00  3300
3201     HARD DISK 20 MB ..  **525.00  4436

S>
```

The PARTDESC column is defined with data type CHAR, so the trailing character is appended to any trailing blanks in the value. If PARTDESC were a VARCHAR column, the trailing characters would begin immediately after the last stored character.

Note that modifiers must be enclosed in square brackets, and a display format with a modifier must be enclosed in double quotation marks.

In the next example, the price field is filled with asterisks between the dollar sign and first digit; they replace the embedded comma and leading zeros of the value. The result of applying this format to 500.00 is $***500.00.

```
PRICE AS "[FL'*'] M<$ZZ,ZZ9.99>",
```

Formatting Dates and Times

To format a date and time in your report, you must consider the data type of the date and time value to be formatted. The data type is determined by the way the value is generated or stored.

Date and Time Values

A report can contain dates and times selected from columns of the following data types: DATETIME, DATE, TIME, or TIMESTAMP. You can also use date-time literals or expressions to generate dates and times for a report. For a complete description of date-time data types and literals, see the *SQL/MP Reference Manual*.

You can compute a date and time by using the COMPUTE_TIMESTAMP, CURRENT_TIMESTAMP, or CURRENT function.

- The COMPUTE_TIMESTAMP function produces a Julian timestamp (sometimes called a Guardian timestamp) for the date and, optionally, the time you specify:

  ```
  COMPUTE_TIMESTAMP (6/18/2000 05:20:30:000:000)
  ```

  or

  ```
  COMPUTE_TIMESTAMP (6/18/2000)
  ```

  The data type of a Julian timestamp is LARGEINT.

- The CURRENT_TIMESTAMP function produces a Julian timestamp for the current date and time at the time the line containing CURRENT_TIMESTAMP is printed. For example, if you use CURRENT_TIMESTAMP in the report title and the report footing, the time portion of the values might differ. The data type of the result is LARGEINT.

  COMPUTE_TIMESTAMP and CURRENT_TIMESTAMP are report writer functions and can be used only in report command print lists and in the SET PARAM and EXECUTE commands. For an example of using the SET PARAM command, see Using Parameters With SELECT Commands on page 3-23.

- The CURRENT function produces a timestamp for the current date and time. The result is a value of data type TIMESTAMP. Each time you execute a SELECT command, a TIMESTAMP value is generated and saved until you execute another SELECT command. Regardless of where you specify CURRENT, the date and time returned is based on the TIMESTAMP value generated when the previous SELECT command was executed.

  You can use the CURRENT function (and all other NonStop SQL/MP date-time functions) anywhere that an SQL expression is allowed. Functions can be used with the SELECT command, the value list of an INSERT or the UPDATE command, and a report command print list. You cannot use the CURRENT function in a SET PARAM or EXECUTE command. For descriptions of the SQL date-time functions, see the *SQL/MP Reference Manual*.

You might also store a date as a numeric value but not as a timestamp. See Formats for Dates Stored as Binary Values on page 4-44.

Julian Timestamp Formats

By using the AS DATE/TIME clause, you can specify the date format and time format of a Julian timestamp.

The following command specifies a page title that includes the local date and time:

```
S> PAGE TITLE "Supplier Parts Summary", TAB 30,
+>     CURRENT_TIMESTAMP AS DATE * TIME *;
S> LIST N 1;

Supplier Parts Summary 02/24/87 09:43:04 PM
                 .               .
                .                 .
```

By default, the date and time are printed in these formats: M2/D2/Y2 and HP2:M2:S2. You can change the default date and time formats for your SQLCI session by setting the DATE_FORMAT and TIME_FORMAT style options as indicated:

```
>> SET STYLE DATE_FORMAT "MA3 D2, Y4",
+>         TIME_FORMAT "HB 'hours' MB2 'minutes'";
```

The new default date format produces Feb 24, 2002, and the new default time format produces 21 hours 44 minutes.

To print the time before the date, you can specify:

```
AS TIME * DATE *
```

If you want only the date or only the time printed, you can omit the option you do not want.

To override the default date format, specify a date format in the AS DATE/TIME clause of a report command:

```
S> PAGE TITLE "Supplier Parts Summary", TAB 30,
+>     CURRENT_TIMESTAMP AS DATE "Y4 MA3 D2";
S> LIST N 1;


Supplier Parts Summary 2000 FEB 24
```

To override the default time format, specify a time format in the AS DATE/TIME clause. For example, the following AS TIME clause specifies a format in local civil time:

```
S> REPORT FOOTING "Report completed at ",
+>     CURRENT_TIMESTAMP AS TIME "HP2:M2" IN LCT;
S> LIST ALL;
. .
. .
Report completed at 02:14 AM
```

Table 4-2 on page 4-42 summarizes the characters you use to specify date and time formats in an AS DATE/TIME clause, a DATE_FORMAT style option, or a TIME_FORMAT style option.

Table 4-2. Format Characters in AS DATE and AS TIME Clauses

Date		Time	
M	Month	H	Hour
D	Day	M	Minute
Y	Year	S	Second
A	All characters of day or month	C	Hundreth of second
A*n*	*n* characters only	T	Thousandths of second
B	Suppress leading zeros	P	Hour as modulo 12 followed by AM or PM
O	All digits of day	B	Suppress leading zeros
O*n*	*n* digits only	*n*	Number of digits
n	Number of characters or digits		

You can format a column value with the AS DATE/TIME clause only if the value is stored as a Julian timestamp of data type NUMERIC(18) or LARGEINT.

You can display the current default date and time formats by entering the following:

```
>> SHOW STYLE DATE_FORMAT, TIME_FORMAT;
```

Formats for NonStop SQL/MP

If your report includes a date or time generated or stored as one of the NonStop SQL/MP date-time data types, you must do one of the following:

- Accept the default format for DATE, TIME, DATETIME, or TIMESTAMP data types as described in Table 4-1 on page 4-28.
- Use the DATEFORMAT function to specify that you want the USA or EUROPEAN version of the format for that data type. The formats produced by the DATEFORMAT function for data types DATETIME and TIMESTAMP are shown in Table 4-3. The data type of the value returned by DATEFORMAT is CHAR.

Table 4-3. DATEFORMAT Display Formats

Format Name	Format
DEFAULT	yyy-mm-dd:hh:mm:ss.msssss
USA	mm/dd/yyyy hh:mm:ss.msssss AM|PM
EUROPEAN	dd.mm.yyyy hh.mm.ss.msssss

- Convert the value to a Julian timestamp by using the JULIANTIMESTAMP function.

  If you are formatting only a part of the date-time value, the applicable parts of the format are used. For example, if you specify YEAR TO DAY in the DATEFORMAT function, the EUROPEAN format will be *dd.mm.yyyy*.

The following print items produce identical formats:

```
CURRENT

DATEFORMAT (CURRENT, DEFAULT)
```

A current date and time generated by either of these functions would appear as follows:

```
2000-04-15:18:22:05.61003
```

If you specify the following print item:

```
DATEFORMAT (CURRENT, EUROPEAN)
```

the date and time would appear as follows:

```
15.04.2000 18.22.05.61003
```

You can request a different number of significant digits in the seconds value. For a complete description of the CURRENT function, see the *SQL/MP Reference Manual*.

If you want to insert characters or spacing within a European date and time, you can include print items such as the following in a report formatting command:

```
DATEFORMAT (CURRENT YEAR TO DAY, EUROPEAN), "*****",
DATEFORMAT (CURRENT HOUR TO MINUTE)
```

The result would appear as follows:

```
15.04.2000 ***** 18.22
```

Note. For values of data type INTERVAL, you must use the default format shown in Table 4-1 on page 4-28.

Converting Timestamps

You can use the JULIANTIMESTAMP function to convert a value with a NonStop SQL/MP date-time data type to a Julian timestamp of data type LARGEINT. You can use the CONVERTTIMESTAMP function to convert a LARGEINT timestamp value to a value of data type TIMESTAMP. The availability of these conversion functions increases your formatting options.

For example, you cannot format a TIMESTAMP value using the AS DATE or AS TIME clause or the A descriptor of the AS clause, unless you use the JULIANTIMESTAMP function to convert the value to a Julian timestamp. You might want to use this conversion to print the name of the month instead of a number. For example, you can specify the following print item to print a date, such as January 5, 2000:

```
JULIANTIMESTAMP (CURRENT) AS DATE "MA DB2, Y4"
```

Note that you do not need to include YEAR TO DAY with CURRENT because the AS DATE clause selects only the month, day, and year from the timestamp.

You can use the JULIANTIMESTAMP function with a TIMESTAMP column named NEXT_MEETING:

```
JULIANTIMESTAMP (NEXT_MEETING) AS DATE "MA DB2, Y4"
```

To get the day of the week printed as a word such as Sunday, you can use the following print item:

```
JULIANTIMESTAMP (CURRENT) AS DATE "DA"
```

For a TIMESTAMP column named XX, you can use the following print item:

```
JULIANTIMESTAMP (XX) AS DATE "DA"
```

If your database contains Julian timestamps in a column, but you want to print them in European format, you can use the CONVERTTIMESTAMP function. For example, assume DATE_TIME is a Julian timestamp:

```
DATEFORMAT (CONVERTTIMESTAMP (DATE_TIME), EUROPEAN)
```

Formats for Dates Stored as Binary Values

If the date is stored as a binary value but not as a timestamp, you can use a mask to enhance the display. For example, the DELIV_DATE column is defined as data type NUMERIC(6) and contains values such as 870618 representing YYMMDD. If you want to print the date with the units separated by slashes, you can use the following mask in the print item:

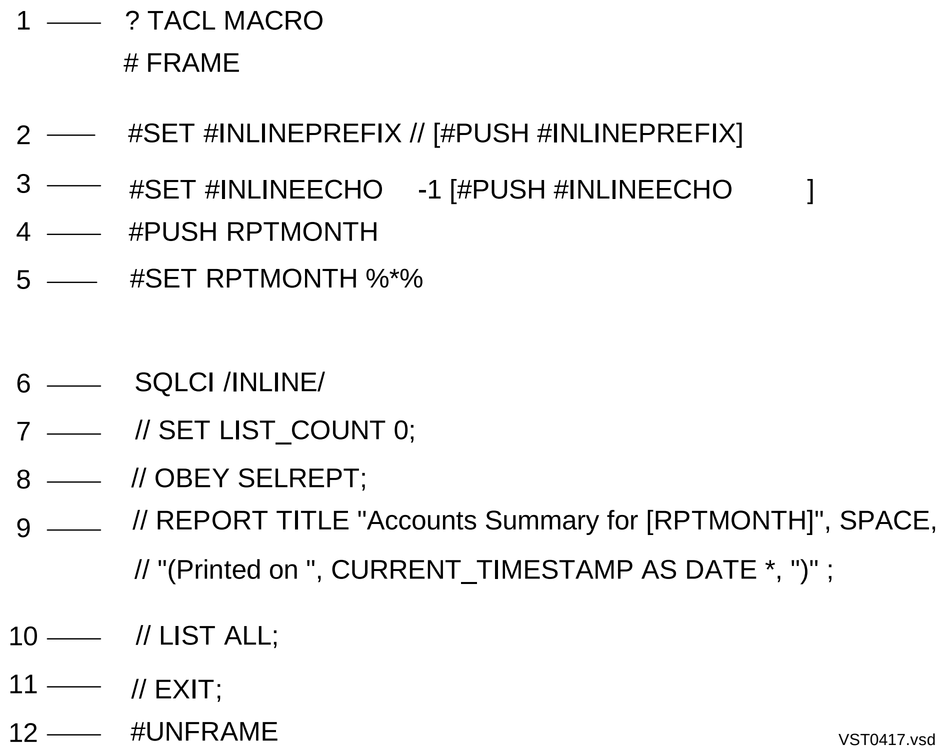
```
DELIV_DATE AS M<03/03/03>
```

The printed result is 03/07/18.

Using TACL to Pass Parameters

If you want to pass a parameter value to a report formatting command, you can write a TACL macro and pass the parameter to the macro when you run it. The parameter is inserted into the report formatting command before it is sent to SQLCI. Figure 4-17 on page 4-45 illustrates this technique.

Figure 4-17. Example of TACL Macro for Passing Parameter



```
1  ——  ? TACL MACRO
       # FRAME

2  ——  #SET #INLINEPREFIX // [#PUSH #INLINEPREFIX]
3  ——  #SET #INLINEECHO    -1 [#PUSH #INLINEECHO        ]
4  ——  #PUSH RPTMONTH
5  ——  #SET RPTMONTH %*%


6  ——  SQLCI /INLINE/
7  ——  // SET LIST_COUNT 0;
8  ——  // OBEY SELREPT;
9  ——  // REPORT TITLE "Accounts Summary for [RPTMONTH]", SPACE,
       // "(Printed on ", CURRENT_TIMESTAMP AS DATE *, ")" ;

10 ——  // LIST ALL;
11 ——  // EXIT;
12 ——  #UNFRAME
```

In Figure 4-17, an command file named SELREPT contains a SELECT command and report definition used to produce a monthly accounts summary. Because the report is not always printed during the month to which it applies, the month is passed as a parameter value instead of computed using the CURRENT_TIMESTAMP function. The printing date is computed with the CURRENT_TIMESTAMP function.

The TACL macro shown in Figure 4-16 is defined in a file named TMREPT01. A parameter named RPTMONTH is used to pass the month and year to which the report applies to the REPORT TITLE command specified in the macro.

To run the report, you enter the following command at the TACL prompt:

```
MYTACL > RUN TMREPT01 June, 2001
```

The parameter value “June, 2001” is inserted in the REPORT TITLE command. (The TACL process must have been started with a process name if you want to use the INLINE facility.)

The report title appears as follows:

```
Accounts Summary for June, 2001 (Printed on 07/03/03)
```

The lines of the TACL macro perform the following operations:

1. Declares the beginning of a TACL macro and specifies a frame for keeping track of pushed variables.

2. Defines the inline prefix as two slashes (//). TACL sends prefixed lines to the inline process (in this case, SQLCI) and waits until the inline process issues another prompt.
3. Sets the inline echo variable to enable echoing.
4. Defines the variable RPTMONTH.
5. Sets RPTMONTH to the parameter value entered with the RUN command (in this case, June, 2001).
6. Runs SQLCI, specifying the INLINE run option.
7. Sends a SET LIST_COUNT command to the SQLCI process.
8. Sends an OBEY command to the SQLCI process. The SELECT command and report formatting commands in the command file execute. The data is selected and the report formatted, but no lines are printed yet.
9. Substitutes the value of RPTMONTH in the REPORT TITLE command and sends the command to the SQLCI process.
10. Sends a LIST command to the SQLCI process. The report is printed.
11. Sends an EXIT command to SQLCI.
12. Pops all variables pushed since the last #FRAME function call.

You can use TACL macros to perform complex database operations by modifying SQLCI and report formatting commands at the time you run the macro.

To learn how to write TACL macros and get complete information about the TACL INLINE facility, see the *TACL Reference Manual*.

Conditional Printing of Items or Line Entries

By using an IF/THEN/ELSE clause, you can specify conditions for printing items or lists of items, and you can specify alternate items to be printed if the conditions are not met. An IF/THEN/ELSE clause can be included in a detail line, title, or footing and within a CONCAT clause. You can also include an IF/THEN/ELSE clause in another IF/THEN/ELSE clause.

The report defined by the following commands conditionally prints text depending on the employee's salary:

```
>> SELECT EMPNUM, FIRST_NAME, LAST_NAME, DEPTNAME,
+> SALARY
+> FROM PERSNL.EMPLOYEE E, PERSNL.DEPT D
+> WHERE E.DEPTNUM = D.DEPTNUM
+> ORDER BY LAST_NAME;
S> DETAIL CONCAT (FIRST_NAME STRIP, SPACE 1, LAST_NAME)
+>           AS A20 HEADING "NAME",
+>        DEPTNAME,
+>        IF SALARY < 30000 THEN ("Evaluate")
+>           ELSE (IF SALARY < 60000 THEN ("Wait 1 month")
+>                 ELSE ("Postpone"))
+>           HEADING "REVIEW PLAN";
S> LIST N 10;


NAME DEPTNAME REVIEW PLAN
-------------------- ------------ ------------
HERB ALBERT ENGLND SALES Wait 1 month
RICHARD BARTON FINANCE Evaluate
MARLENE BONNY RESEARCH Evaluate
ERIC BROWN RESEARCH Postpone
SUSAN CHAPMAN PERSONNEL Evaluate
JOHN CHOU ASIA SALES Evaluate
DINAH CLARK CORPORATE Wait 1 month
LARRY CLARK FINANCE Evaluate
MANFRED CONRAD RESEARCH Wait 1 month
STEVE COOK RESEARCH Postpone
S>
```

You should consider the following points when specifying an IF/THEN/ELSE clause:

- The print list must be enclosed in parentheses. In the preceding example, the ELSE print list consists of another IF/THEN/ELSE clause.
- The report writer determines the space required for the result of the clause by using the longest of the alternate print lists. The shorter print list result is padded with blanks.
- If you omit the ELSE clause, nothing is printed when the condition is not met.
- You specify the condition by using the NonStop SQL/MP search condition syntax and rules with the following exceptions:
    - You cannot include subqueries.
    - Arithmetic expressions cannot include aggregate functions or report functions.

You can specify predicates as the following example illustrates:

```
>> SELECT * FROM PERSNL.EMPLOYEE;
S> DETAIL LAST_NAME AS A12,
+>        FIRST_NAME AS A12,
+>        IF DEPTNUM BETWEEN 3000 AND 4000
+>           THEN ("Sales")
+>        ELSE (IF DEPTNUM IN (2000, 2500)
+>           THEN ("Shipping")
+> ELSE ("Research") ) HEADING "DIVISION";
S> LIST N 5 ;


LAST_NAME FIRST_NAME DIVISION
------------ ------------ --------


GREEN ROGER Research
HOWARD JERRY Research
```

```
RAYMOND JANE Sales
RUDLOFF THOMAS Shipping
SAFFERT KLAUS Sales
S>
```

You can use decorations to specify conditions for printing information within a print item. For a description of decorations, see Monetary Values on page 4-32.

For another type of conditional printing, see Filler Characters on page 4-38.

Note. Because the value in a column specified by using an IF/THEN/ELSE clause is always in character format, you cannot use the SUBTOTAL or TOTAL command to calculate totals on the column. For another method, for totaling columns containing conditional values, see Calculating Subtotals on Conditional Values on page 4-56.

Redefining Special Characters

The report writer has special characters that have default values. You can change these values with the SET STYLE command as follows:

UNDERLINE_CHAR The default single-byte character used for underlining. Underline characters are printed below headings, subtotals, and totals. The default is a hyphen (-). You can specify any printable single-byte character.

DECIMAL_POINT The default single-byte decimal character. The decimal point you set is used in numeric print items. You can specify a period(.) or a comma (,). The default decimal character is a period.

NEWLINE_CHAR The default character used to indicate where a new line begins in a multiple-line heading. The default character is a slash (/). You can specify any single-byte character except circumflex (^) and hyphen (-).

NULL_DISPLAY The default character used to indicate a null value. The default character is a question mark (?). You can specify any single-byte character.

OVERFLOW_CHAR The default character used to fill a field when the value is too large for the display format. The default character is an asterisk (*). You can specify any single-byte character.

To change a style option, you enter a SET STYLE command and specify one or more options as shown:

```
>> SET STYLE UNDERLINE_CHAR "=", OVERFLOW_CHAR "+";
```

You can display the current default characters by entering the following:

```
>> SHOW STYLE *;
```

You can reset an option to its default with the RESET STYLE command. For example, the next command resets the default underline character to the underscore (_):

```
>> RESET STYLE UNDERLINE_CHAR;
```

The next command sets all style options to their default values:

```
>> RESET STYLE *;
```

Calculating Totals

You can calculate totals of print-item values. The values must be numeric. A total appears at the end of the report. For example, suppose you want to total the quantity ordered of all parts in the following report:

```
>> SET LIST_COUNT 0;
>> SELECT PARTNUM, SUM (QTY_ORDERED)
+> FROM SALES.ODETAIL
+> GROUP BY PARTNUM;
S> TOTAL COL 2;
S> LIST ALL;


PARTNUM (EXPR)
------- -------------------



212 20
244 47
255 38
. .
. .
. .
7102 18
7301 96
-------------------
-------------------
1406


--- 29 row(s) selected.
```

In a TOTAL command, you can refer to an item of the select list by column name, alias name, or column number. You can reference a print item of the detail line by column name or detail alias name.

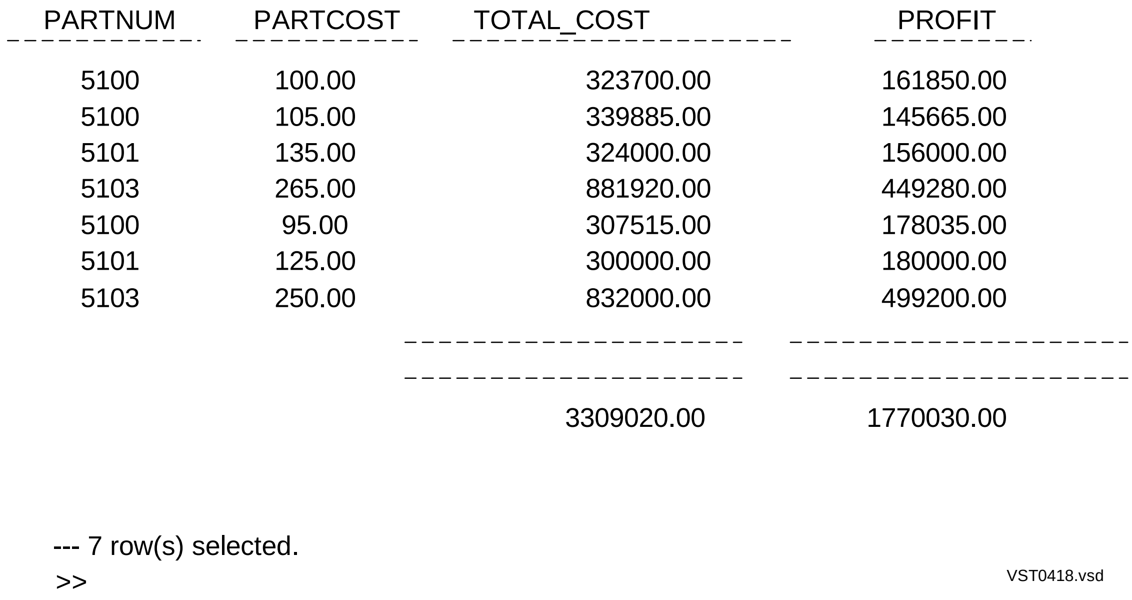
Only one TOTAL command is in effect at a time. If you want to compute totals for more than one item, you must specify all the items in one command. You can edit the TOTAL command by using FC, replace the command by reentering it, delete the command by using RESET REPORT TOTAL, or modify the TOTAL command by using RESET REPORT TOTAL (*column-list*).

In the next example, the columns that contain expressions are assigned detail alias names. These columns are totaled at the end of the report.

```
>> VOLUME INVENT;
>> SET LIST_COUNT 0;
>> SELECT S.SUPPNUM, SUPPNAME, CITY, STATE, P.PARTNUM,
+> QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+> AND PS.SUPPNUM = S.SUPPNUM
+> AND P.PARTNUM IN (5100, 5101, 5103)
+> ORDER BY S.SUPPNUM, P.PARTNUM;
S> DETAIL PARTNUM,
+> PARTCOST AS F8.2,
+> PARTCOST * QTY_AVAILABLE NAME TOTAL_COST,
+> QTY_AVAILABLE * (PRICE - PARTCOST) NAME PROFIT;
+> TOTAL TOTAL_COST, PROFIT;
S> LIST ALL;
```

Figure 4-18 shows the resulting report.

Figure 4-18. Example of a Report With Totals



```
  PARTNUM     PARTCOST    TOTAL_COST                  PROFIT
-----------  ----------  --------------------      ---------

     5100      100.00             323700.00        161850.00
     5100      105.00             339885.00        145665.00
     5101      135.00             324000.00        156000.00
     5103      265.00             881920.00        449280.00
     5100       95.00             307515.00        178035.00
     5101      125.00             300000.00        180000.00
     5103      250.00             832000.00        499200.00
                       --------------------   --------------------
                       --------------------   --------------------

                                 3309020.00       1770030.00


   --- 7 row(s) selected.
   >>
```

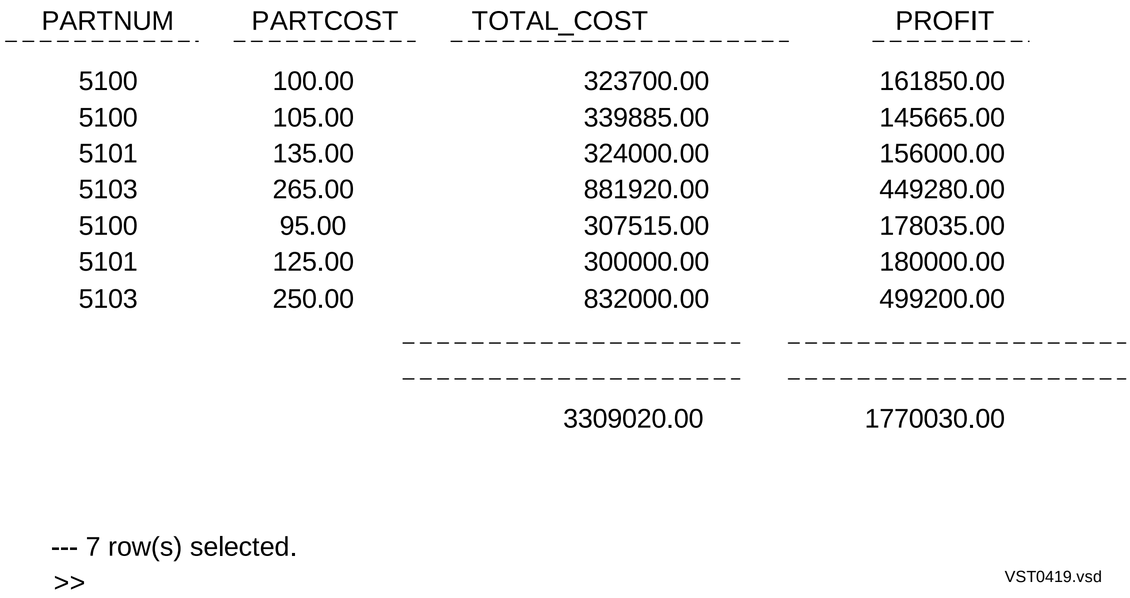
VST0418.vsd

Totals are printed in a default format. You can override the size of the field and the format of the value by using an AS clause for the print item in the DETAIL command. For example, suppose you want to specify smaller fields for the last two columns of the report:

```
S> DETAIL PARTNUM,
+>        PARTCOST,
+>        PARTCOST * QTY_AVAILABLE AS F14.2
+>            NAME TOTAL_COST,
+>        QTY_AVAILABLE * (PRICE - PARTCOST) AS F12.2
+>            NAME PROFIT;
+> TOTAL TOTAL_COST, PROFIT;
S> LIST ALL;
```

Figure 4-19 shows the resulting report.

Figure 4-19. Example of Formatted Total Values



```
   PARTNUM        PARTCOST       TOTAL_COST                      PROFIT
 ------------   -----------   ---------------------          ----------

     5100         100.00                 323700.00            161850.00
     5100         105.00                 339885.00            145665.00
     5101         135.00                 324000.00            156000.00
     5103         265.00                 881920.00            449280.00
     5100          95.00                 307515.00            178035.00
     5101         125.00                 300000.00            180000.00
     5103         250.00                 832000.00            499200.00
                            ---------------------   ---------------------
                            ---------------------   ---------------------

                                        3309020.00           1770030.00


   --- 7 row(s) selected.
   >>
```

VST0419.vsd

The display format you specify must be large enough to contain the result of the TOTAL command.

You cannot use an AS clause to format a value of the INTERVAL data type. These values must be displayed or printed in the default format. If you calculate a total (or subtotal) of INTERVAL values, the resulting value might not fit in the field for the default format. If you plan to compute totals for a column of this data type, the column definition specified when the table is created should include a precision large enough to contain the total values. For example, even though the column will contain only 2-digit values, the precision might be 6 digits (or more) to accommodate total values.

Calculating Subtotals

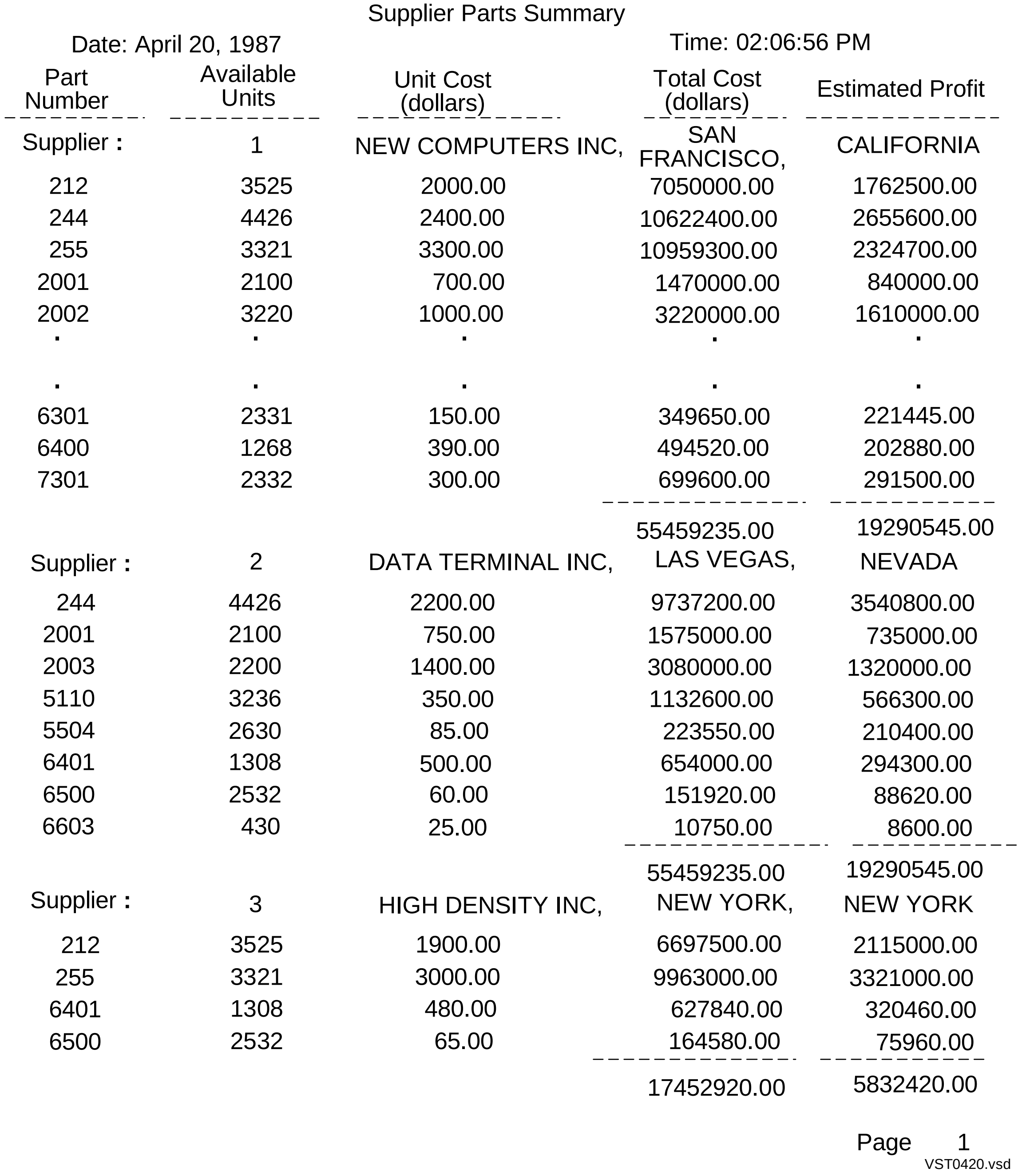
You can calculate subtotals for one or more columns each time a break point occurs. Use the ORDER BY clause of the SELECT command and the BREAK ON command to define the break groups. For more information about these commands, see Organizing Rows Into Break Groups on page 4-14.

In the supplier parts summary report in Figure 4-19, subtotals are computed for the total cost and estimated profit. The report is defined with the following commands:

```
>> VOLUME INVENT;
>> SET LIST_COUNT 0;
>> SET LAYOUT LEFT_MARGIN 8, RIGHT_MARGIN 72,
+>            PAGE_LENGTH 56;
>> SELECT S.SUPPNUM, SUPPNAME, CITY, STATE, P.PARTNUM,
+>         QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SALES.PARTS P, PARTSUPP PS, SUPPLIER S
+> WHERE P.PARTNUM = PS.PARTNUM
+>        AND PS.SUPPNUM = S.SUPPNUM
+> ORDER BY S.SUPPNUM, P.PARTNUM;
S> DETAIL PARTNUM AS I6 HEADING "Part/Number" CENTER,
+>    QTY_AVAILABLE AS I9 HEADING "Available/Units" CENTER,
+>    PARTCOST AS F11.2 HEADING "Unit Cost/(dollars)" CENTER,
+>        PARTCOST * QTY_AVAILABLE AS F14.2
+>            HEADING "Total Cost/(dollars)" CENTER
+>            NAME TOTAL_COST,
+>        QTY_AVAILABLE * (PRICE - PARTCOST) AS F11.2
+>            HEADING "Estimated/Profit" CENTER
+>            NAME PROFIT;
S> BREAK ON S.SUPPNUM;
S> BREAK TITLE S.SUPPNUM
+>            ("Supplier:", S.SUPPNUM,
+>            CONCAT (SUPPNAME STRIP, ", ",CITY STRIP, ", ",
+>                    STATE),
+>            SKIP 1) ;
S> SUBTOTAL TOTAL_COST, PROFIT OVER S.SUPPNUM;
+> TOTAL TOTAL_COST, PROFIT;
S> PAGE TITLE "Supplier Parts Summary" CENTER;
S> REPORT TITLE "Date: ",
S>            CURRENT_TIMESTAMP AS DATE “MA DB2, Y4”,
+>            TAB 48,
+>            "Time: ", CURRENT_TIMESTAMP AS TIME *;
S> REPORT FOOTING "End of Summary" CENTER;
S> PAGE FOOTING TAB 58, "Page ", PAGE_NUMBER AS I3;
S> OUT_REPORT $S.#PRINTER;
S> LIST ALL;
```

Figure 4-20 shows the resulting two-page report.

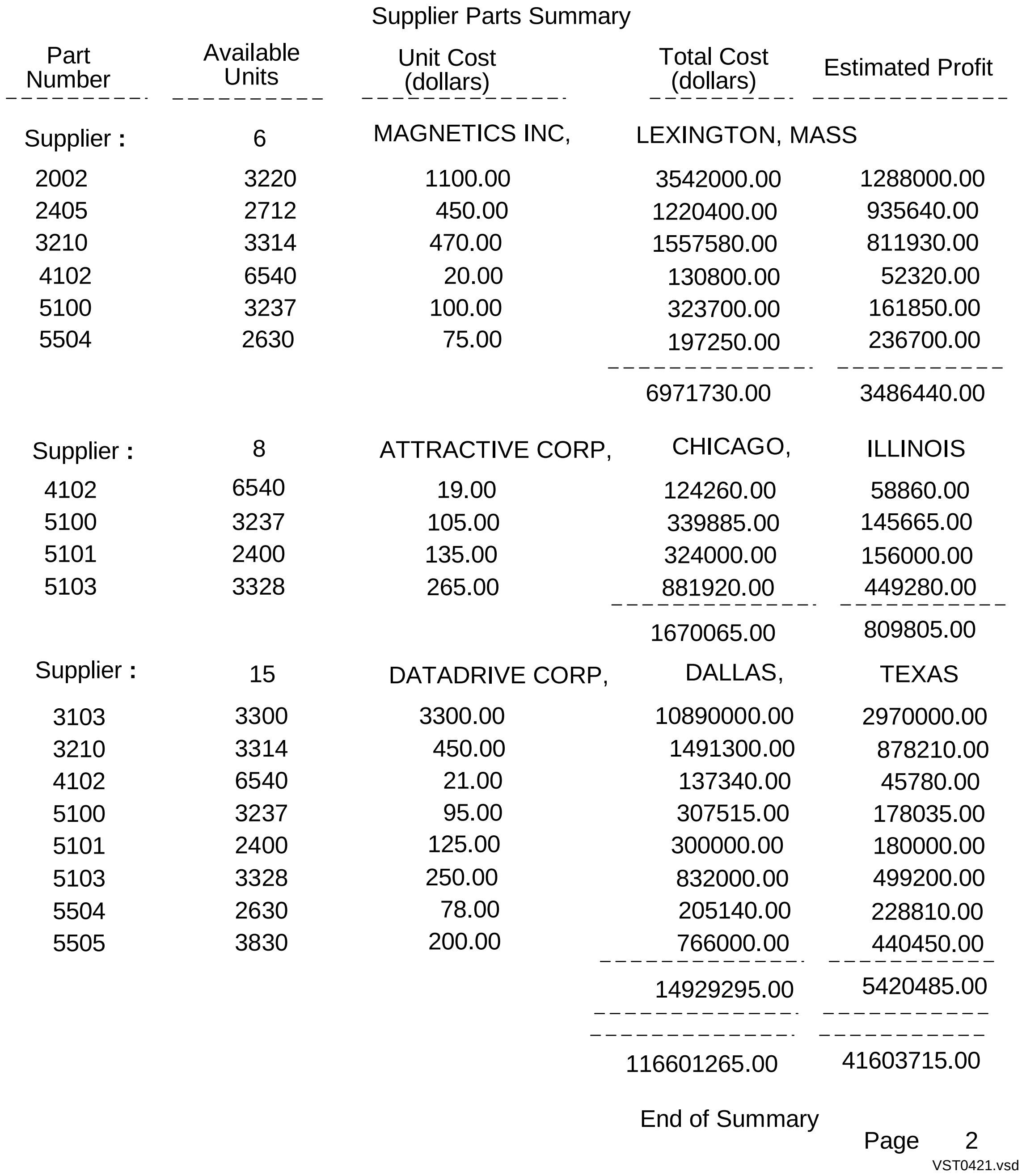
Figure 4-20. Example of Report With Subtotals (Page 1 of 2)



```
                              Supplier Parts Summary
      Date: April 20, 1987                          Time: 02:06:56 PM
   Part        Available        Unit Cost           Total Cost
   Number        Units          (dollars)           (dollars)        Estimated Profit
 ----------   ----------     ---------------       ----------      --------------
  Supplier :       1        NEW COMPUTERS INC,         SAN
                                                   FRANCISCO,        CALIFORNIA
   212           3525          2000.00            7050000.00        1762500.00
   244           4426          2400.00           10622400.00        2655600.00
   255           3321          3300.00           10959300.00        2324700.00
   2001          2100           700.00            1470000.00         840000.00
   2002          3220          1000.00            3220000.00        1610000.00
    .              .              .                   .                 .

    .              .              .                   .                 .
   6301          2331           150.00             349650.00         221445.00
   6400          1268           390.00             494520.00         202880.00
   7301          2332           300.00             699600.00         291500.00
                                               ---------------    ------------
                                                 55459235.00       19290545.00
   Supplier :      2        DATA TERMINAL INC,    LAS VEGAS,          NEVADA
    244          4426          2200.00            9737200.00        3540800.00
   2001          2100           750.00            1575000.00         735000.00
   2003          2200          1400.00            3080000.00        1320000.00
   5110          3236           350.00            1132600.00         566300.00
   5504          2630            85.00             223550.00         210400.00
   6401          1308           500.00             654000.00         294300.00
   6500          2532            60.00             151920.00          88620.00
   6603           430            25.00              10750.00           8600.00
                                                ---------------    -----------
                                                  55459235.00      19290545.00
   Supplier :      3        HIGH DENSITY INC,      NEW YORK,          NEW YORK
    212          3525          1900.00            6697500.00        2115000.00
    255          3321          3000.00            9963000.00        3321000.00
   6401          1308           480.00             627840.00         320460.00
   6500          2532            65.00             164580.00          75960.00
                                              ---------------    ------------
                                                  17452920.00       5832420.00

                                                                  Page      1
```

VST0420.vsd

Figure 4-21. Example of Report With Subtotals (Page 2 of 2)



```
                                 Supplier Parts Summary
   Part          Available       Unit Cost            Total Cost
  Number           Units         (dollars)            (dollars)      Estimated Profit
 ----------     ----------    --------------         ----------     -------------

 Supplier :          6         MAGNETICS INC,       LEXINGTON, MASS
  2002            3220            1100.00              3542000.00      1288000.00
  2405            2712             450.00              1220400.00       935640.00
  3210            3314             470.00              1557580.00       811930.00
  4102            6540              20.00               130800.00        52320.00
  5100            3237             100.00               323700.00       161850.00
  5504            2630              75.00               197250.00       236700.00
                                                    --------------   -----------
                                                       6971730.00      3486440.00

 Supplier :          8        ATTRACTIVE CORP,        CHICAGO,         ILLINOIS
  4102            6540              19.00               124260.00        58860.00
  5100            3237             105.00               339885.00       145665.00
  5101            2400             135.00               324000.00       156000.00
  5103            3328             265.00               881920.00       449280.00
                                                    --------------   -----------
                                                       1670065.00       809805.00
 Supplier :         15        DATADRIVE CORP,         DALLAS,           TEXAS
  3103            3300            3300.00             10890000.00      2970000.00
  3210            3314             450.00              1491300.00       878210.00
  4102            6540              21.00               137340.00        45780.00
  5100            3237              95.00               307515.00       178035.00
  5101            2400             125.00               300000.00       180000.00
  5103            3328             250.00               832000.00       499200.00
  5504            2630              78.00               205140.00       228810.00
  5505            3830             200.00               766000.00       440450.00
                                                    --------------   -----------
                                                      14929295.00      5420485.00
                                                    --------------   -----------
                                                    --------------   -----------
                                                     116601265.00     41603715.00

                                                     End of Summary
                                                                      Page      2
```

VST0421.vsd

Consider these points when using the SUBTOTAL command:

- You can specify the column identifier values in any order.
- You must define the break groups with a BREAK ON command.
- You must specify in one SUBTOTAL command all columns you want to subtotal over the same break group. You can specify additional SUBTOTAL commands for other break groups.

- The OVER clause indicates when a subtotal is to be calculated. If you omit the OVER clause, the subtotals appear whenever any data value in any currently defined break column changes. Only one SUBTOTAL command without an OVER clause is in effect at one time.
- You can delete all subtotal commands by entering:

  ```
  S> RESET REPORT SUBTOTAL;
  ```

  RESET REPORT BREAK also resets subtotals by deleting the specified break columns. You can change a SUBTOTAL command with the FC command, replace the command by reentering it with the same OVER specification, or modify the SUBTOTAL command by using RESET REPORT SUBTOTAL (*column-list*).
- Entering a SUBTOTAL command returns you to the first row of selected output.
- You should make the display format for the column large enough to accommodate the subtotal values.

Defining a Subtotal Label

A subtotal label appears in the break column when the subtotal is calculated. In the report in Figure 4-20, no subtotal labels appear because the break column SUPPNUM does not appear in the detail line. When the value of SUPPNUM changes, the subtotal is calculated but no label appears.

The report in Figure 4-22 summarizes current orders. Subtotals are calculated over more than one break group.

The default subtotal label is an asterisk (*). You can set a different default label with the SET STYLE command. For example, the following command sets the label used in Figure 4-22:

```
>> SET STYLE SUBTOTAL_LABEL "* * *";
```

You can specify from 0 through 255 single-byte characters (or 0 through 177 doublebyte characters) in the label. To turn off subtotal labeling, specify zero characters as follows:

```
>> SET STYLE SUBTOTAL_LABEL "";
```

The label you specify applies to all subtotals. If the label does not fit in the column, the string is truncated.

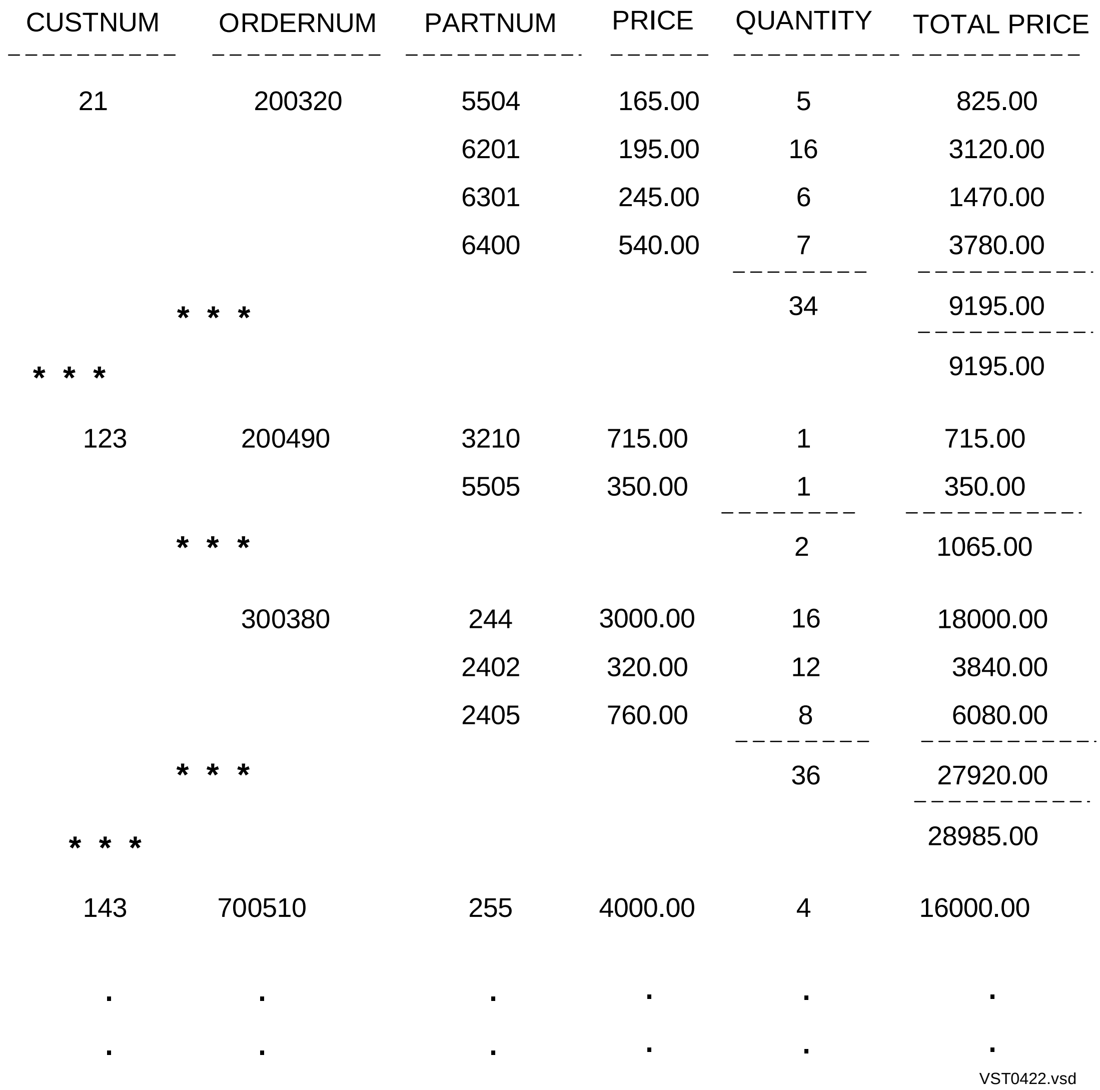
Note. If you are defining a report that contains double-byte characters, see Printing Double-Byte Characters on page 4-58 for special considerations regarding defining subtotal labels.

The following commands define the report in Figure 4-20:

```
>> VOLUME SALES;
>> SELECT * FROM ORDERS O, ODETAIL OD
+> WHERE O.ORDERNUM = OD.ORDERNUM
+> ORDER BY CUSTNUM, OD.ORDERNUM;
S> DETAIL CUSTNUM,
```

```
+>        OD.ORDERNUM,
+>        PARTNUM,
+>        UNIT_PRICE AS F8.2 HEADING "PRICE",
+>        QTY_ORDERED AS I8 HEADING "QUANTITY",
+>        UNIT_PRICE * QTY_ORDERED AS F14.2
+>            HEADING "TOTAL PRICE" NAME TOTALP ;
S> BREAK ON CUSTNUM, OD.ORDERNUM;
S> SUBTOTAL QTY_ORDERED, TOTALP OVER OD.ORDERNUM;
S> SUBTOTAL TOTALP OVER CUSTNUM;
S> LIST ALL;
```

Figure 4-22. Customer Orders Summary



```
  CUSTNUM     ORDERNUM    PARTNUM     PRICE   QUANTITY  TOTAL PRICE
----------  ----------  ----------  ------  ---------- ----------

    21        200320      5504      165.00      5        825.00
                          6201      195.00     16       3120.00
                          6301      245.00      6       1470.00
                          6400      540.00      7       3780.00
                                              --------  ----------
             * * *                             34       9195.00
                                                        ----------
  * * *                                                 9195.00

    123       200490      3210      715.00      1        715.00
                          5505      350.00      1        350.00
                                              --------  ----------
             * * *                              2       1065.00

              300380       244     3000.00     16      18000.00
                          2402      320.00     12       3840.00
                          2405      760.00      8       6080.00
                                              --------  ----------
             * * *                             36      27920.00
                                                        ----------
   * * *                                               28985.00

    143       700510       255     4000.00      4      16000.00

     .          .           .         .         .          .
     .          .           .         .         .          .
```

Calculating Subtotals on Conditional Values

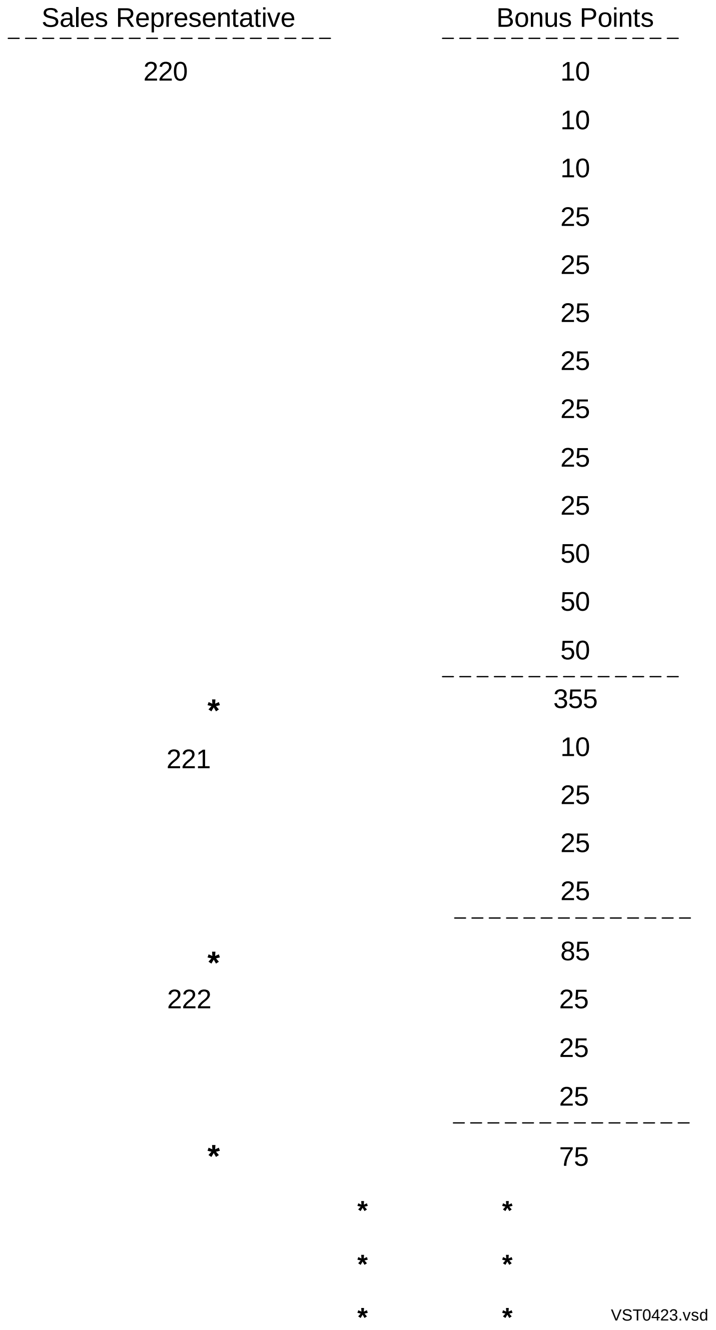
The value in a column specified by using an IF/THEN/ELSE clause is in character format and cannot be subtotaled or totaled by using the SUBTOTAL or TOTAL command. You can use the UNION operator to specify a select list column that contains a conditional value and can be subtotaled.

In the following example, a union of three SELECT statements (shown in boldface type) retrieves the data for a report. Column 2 of the detail line contains the number of bonus points the sales representative has earned based on the quantity of parts ordered by customers: 10 points for up to 5 units of a specific part, 25 points for 6 to 15 units, and 50 points for more than 15 units. A subtotal of all bonus points for each sales representative is computed.

```
>> VOLUME SALES;
>> SET LIST_COUNT 0;
>> SELECT SALESREP, 10
+> FROM ORDERS ORD, ODETAIL OD
+> WHERE ORD.ORDERNUM = OD.ORDERNUM
+>   AND QTY_ORDERED <= 5
+> UNION ALL
>> SELECT SALESREP, 25
+> FROM ORDERS ORD, ODETAIL OD
+> WHERE ORD.ORDERNUM = OD.ORDERNUM
+>   AND QTY_ORDERED BETWEEN 6 AND 15
+> UNION ALL
>> SELECT SALESREP, 50
+> FROM ORDERS ORD, ODETAIL OD
+> WHERE ORD.ORDERNUM = OD.ORDERNUM
+>   AND QTY_ORDERED > 15
+> ORDER BY SALESREP;
S> DETAIL SALESREP HEADING ”Sales Representative”,
+> COL 2 HEADING ”Bonus Points” ;
S> BREAK ON SALESREP;
S> SUBTOTAL COL 2 OVER SALESREP ;
S> LIST ALL ;
```

Figure 4-23 shows the first few lines of the resulting report.

Figure 4-23. Example of Subtotals on Conditional Values



```
 Sales Representative        Bonus Points
 --------------------        --------------
        220                       10
                                  10
                                  10
                                  25
                                  25
                                  25
                                  25
                                  25
                                  25
                                  25
                                  50
                                  50
                                  50
                             --------------
           *                     355
          221                     10
                                  25
                                  25
                                  25
                              --------------
           *                      85
          222                     25
                                  25
                                  25
                              --------------
           *                      75
                    *        *
                    *        *
                    *        *
```

VST0423.vsd

Printing Double-Byte Characters

If you are using a double-byte character set, such as Tandem Kanji or Tandem KSC5601, there are certain conditions in which formatting specifications can split double-byte characters when output is displayed. When double-byte characters are split, they appear to be incorrect, as do subsequent characters preceding the next blank character in the output line.

This subsection describes how to avoid splitting double-byte characters during the following operations:

- Wrapping or folding text to a new line
- Truncating text using display descriptors
- Overlapping text when tabbing backwards
- Truncating subtotal labels

Note. In the examples in this subsection, the values c1, c2, c3, and so forth represent double-byte characters, and the @ and # characters represent bytes that appear incorrect.

Wrapping or Folding Text With Double-Byte Characters

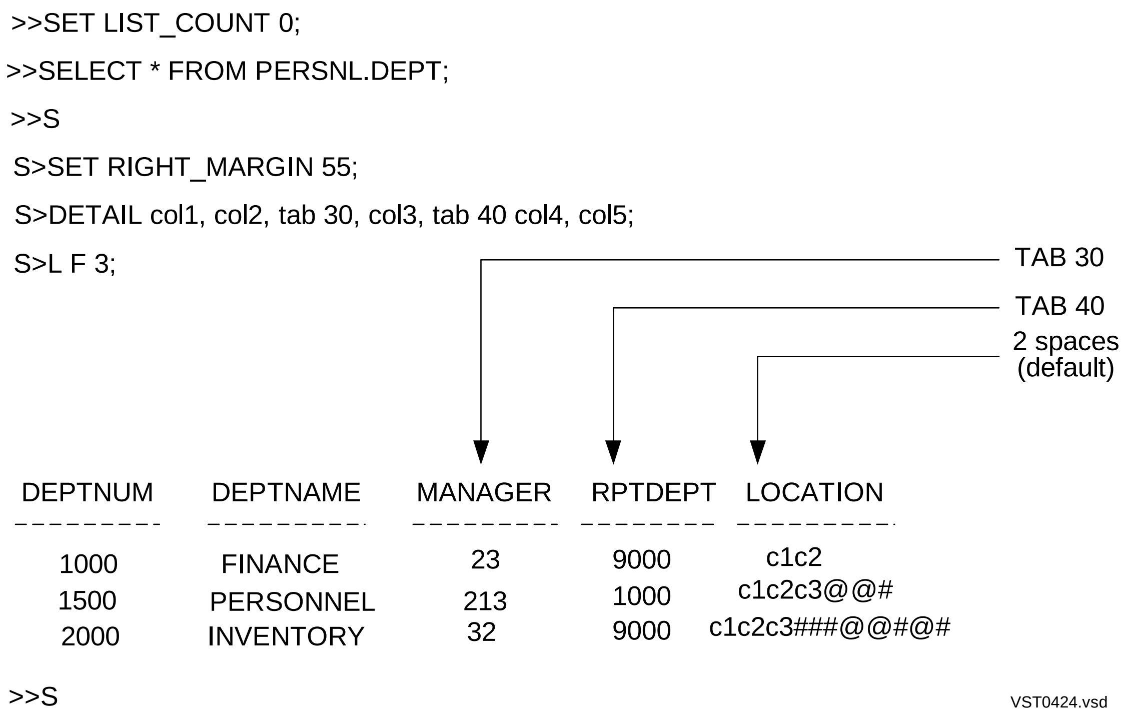
Folding relates to the length defined within the right and left margins (as specified by the LEFT_MARGIN and RIGHT_MARGIN layout options). If the length of a logical report line does not fit within the margins, the logical report line is folded to the next line.

Wrapping can occur when the right margin is beyond the right most position of the output device. Each report line that extends beyond the right most position of the output device is wrapped if the WRAP option of the SET SESSION command is set to ON; otherwise the report line is truncated at the right most position of the output device.

If you are using the default detail line, with the LOGICAL_FOLDING layout option set to ON, double-byte characters in a display column are never split when text wraps or folds. If you generate your own detail report line, you need to make sure that folding or wrapping does not split double-byte characters. If splitting occurs, you can adjust the detail line with the SPACE or TAB clause.

Figure 4-24 demonstrates a report with incorrect output resulting from text folding in the middle of a double-byte character. In this example, the LOCATION column is defined as CHAR(9), using the Kanji character set. The right margin is set to print position 55, and the LOCATION column starts in print position 49. This specification splits the fourth double-byte character, when present, causing it and subsequent characters in the display column to appear incorrect.

Figure 4-24. Splitting Double-Byte Characters With Text Wrapping

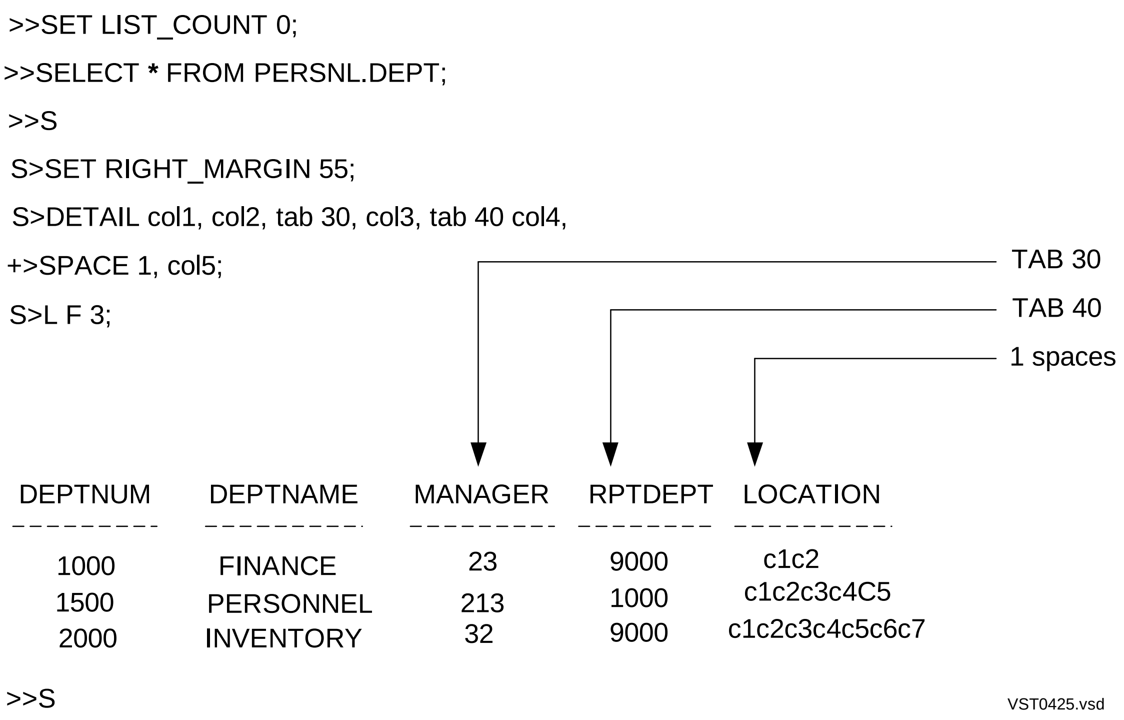


```
>>SET LIST_COUNT 0;
>>SELECT * FROM PERSNL.DEPT;
>>S
S>SET RIGHT_MARGIN 55;
S>DETAIL col1, col2, tab 30, col3, tab 40 col4, col5;
S>L F 3;

 DEPTNUM     DEPTNAME    MANAGER  RPTDEPT  LOCATION
 ---------   ----------  -------- -------- ----------

   1000       FINANCE       23      9000      c1c2
   1500      PERSONNEL     213      1000    c1c2c3@@#
   2000      INVENTORY      32      9000  c1c2c3###@@#@#

>>S
```

Using the SPACE Clause to Adjust Display Columns

One way to adjust a detail line so that double-byte characters are not split with text wrapping or folding is to use the SPACE clause, as shown in Figure 4-25.

Figure 4-25. Using the SPACE Clause to Adjust Display Columns



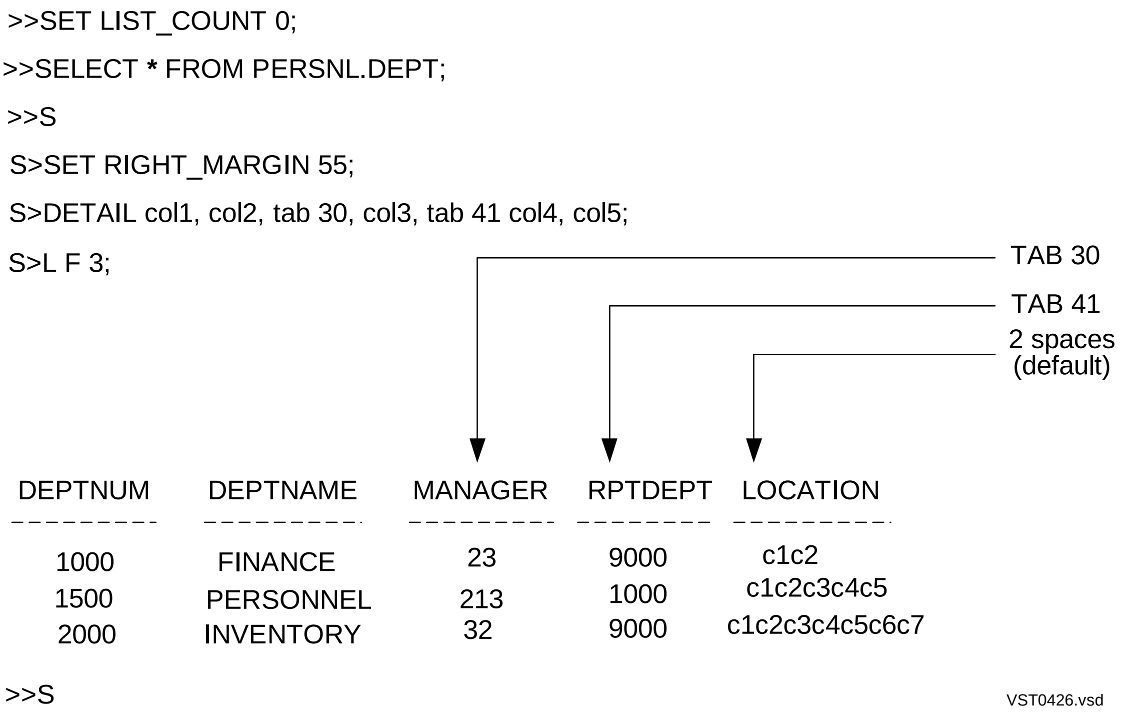
```
>>SET LIST_COUNT 0;
>>SELECT * FROM PERSNL.DEPT;
>>S
S>SET RIGHT_MARGIN 55;
S>DETAIL col1, col2, tab 30, col3, tab 40 col4,
+>SPACE 1, col5;
S>L F 3;

 DEPTNUM     DEPTNAME     MANAGER   RPTDEPT   LOCATION
---------   ----------   --------- --------- ----------
  1000        FINANCE       23        9000       c1c2
  1500       PERSONNEL      213       1000     c1c2c3c4C5
  2000       INVENTORY      32        9000    c1c2c3c4c5c6c7

>>S
```

Using the TAB Clause to Adjust Display Columns

Another way to adjust a report so that double-byte characters are not split with text wrapping or folding is to use the TAB clause.

Figure 4-26 demonstrates changing the TAB clause specification from 40 to 41 to adjust the detail line so that double-byte characters are not split if text wrapping occurs.

Figure 4-26. Using the TAB Clause to Adjust Display Columns



```
>>SET LIST_COUNT 0;
>>SELECT * FROM PERSNL.DEPT;
>>S
S>SET RIGHT_MARGIN 55;
S>DETAIL col1, col2, tab 30, col3, tab 41 col4, col5;
S>L F 3;

DEPTNUM      DEPTNAME     MANAGER   RPTDEPT  LOCATION
---------    ---------    ---------  --------  ---------
   1000        FINANCE       23        9000     c1c2
   1500       PERSONNEL      213       1000    c1c2c3c4c5
   2000       INVENTORY      32        9000  c1c2c3c4c5c6c7

>>S
```

Note. The same wrapping, or the same folding, can occur for print lines generated for the other report writer commands, such as REPORT TITLE, REPORT FOOTING, PAGE TITLE, AND PAGE FOOTING. Use the SPACE or TAB clauses as required to adjust spacing if double-byte characters are being split.

Using Display Descriptors with Double-Byte Characters

Display descriptors, specified by the AS clause, define the display format for print items.

A[*w*] specifies an alphanumeric field with a display width of *w* print positions. If *w* is omitted, the width of the print item is used. If the print item is smaller than *w*, the item is left-justified with blank fill. If *w* is too small for the value, the value is truncated.

C*n*.[*w*] specifies the format for long character fields such as VARCHAR fields. The *n* specifies the number of print positions to print or display. If *n* is 0, the entire field is printed. The *w* specifies the number of print positions per line. If you do not include *w*, the C format is the same as the A format.

There are two scenarios in which display descriptors can split double-byte characters:

- The width of the display descriptor is odd-numbered
- The width of the display descriptor is too small

Using An Odd-Numbered Width Specification

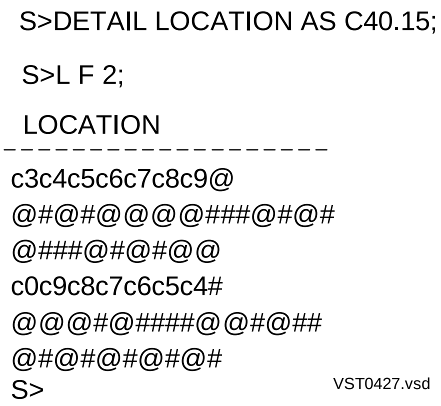
When a double-byte character column is specified with either display descriptor A or display descriptor C, and the display-width specification is an odd number for either *n* or *w*, double-byte character splitting can occur. To avoid this condition, always specify an even-numbered display descriptor for a column that contains double-byte characters.

For example, suppose that a column named LOCATION is defined as CHAR (20) with the character set Kanji and that it contains the following data:

```
c1c2c3c4c5c6c7c8c9c0
c0c9c8c7c6c5c4c3c2c1c0c9c8c7c6
c1c3c5c7c9c0c8c6c4c2c0c1c3
c2c4c6c8c0c9c7c5c3c1c0c3
```

In Figure 4-27, the display descriptor, C40.15, is an odd number, which splits the eighth double-byte character. To adjust for this condition, the display descriptor can be specified as C40.14 or C40.16.

Figure 4-27. Splitting Double-Byte Characters With an Odd-Numbered Display Descriptor



```
 S>DETAIL LOCATION AS C40.15;

 S>L F 2;

 LOCATION
--------------------
c3c4c5c6c7c8c9@
@#@#@@@@###@#@#
@###@#@#@@
c0c9c8c7c6c5c4#
@@@#@####@@#@##
@#@#@#@#@#
S>
```

VST0427.vsd

Using Incorrectly-Sized Display Descriptors

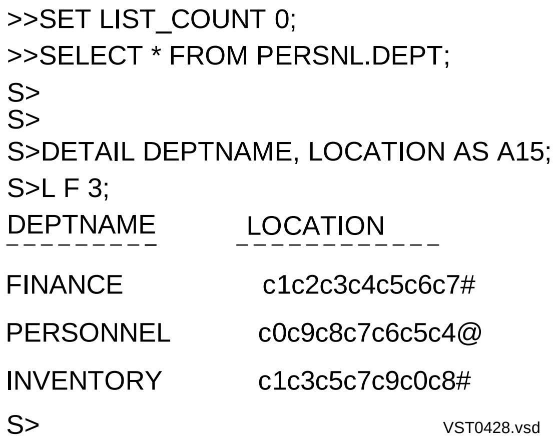
Data truncation occurs when the width of a display descriptor is too small for the value in the field. If the field contains a double-byte character, truncation can split the character. To avoid this condition, check that display descriptors specify areas large enough for the field value.

For example, suppose a column named LOCATION is defined as CHAR (16) with the character set Kanji and that it contains the following data:

```
c1c2c3c4c5c6c7c8c9c0
c0c9c8c7c6c5c4c3c2c1c0c9c8c7c6
c1c3c5c7c9c0c8c6c4c2c0c1c3
c2c4c6c8c0c9c7c5c3c1c0c3
```

In Figure 4-28, the display descriptor, A15, defines a width that is too small, which truncates the display column in the middle of the eighth double-byte character. To adjust for this condition, the display descriptor can be specified as A16.

Figure 4-28. Splitting Double-Byte Characters With Incorrectly-Sized Display Descriptors



```
>>SET LIST_COUNT 0;
>>SELECT * FROM PERSNL.DEPT;
S>
S>
S>DETAIL DEPTNAME, LOCATION AS A15;
S>L F 3;
DEPTNAME         LOCATION
---------       ------------

FINANCE           c1c2c3c4c5c6c7#

PERSONNEL        c0c9c8c7c6c5c4@

INVENTORY        c1c3c5c7c9c0c8#

S>
```

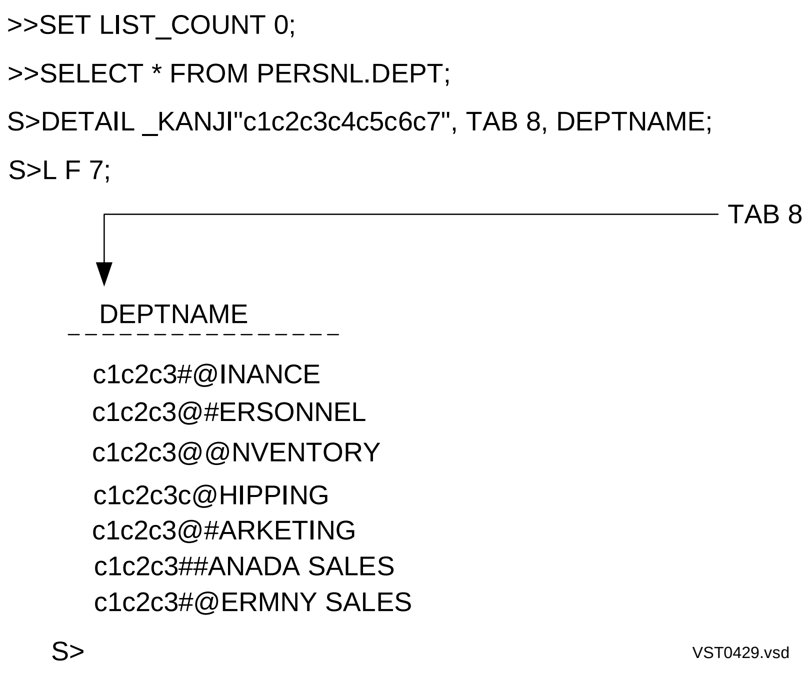
VST0428.vsd

Tabbing Backwards With Double-Byte Characters

When you tab backwards on a detail line using the TAB clause, you can cause data to overlap previous data. If the data consists of double-byte characters, character splitting can occur when overlapped with other data.

In the example shown in Figure 4-29, the TAB 8 clause in the DETAIL command causes the DEPTNAME to start printing in print position 8. This printing position splits the fourth double-byte character (c4), causing it to appear incorrect. To adjust for this condition, the TAB value can be set to either 7 or 9.

Figure 4-29. Splitting Double-Byte Characters With the TAB Clause



```
>>SET LIST_COUNT 0;
>>SELECT * FROM PERSNL.DEPT;
S>DETAIL _KANJI"c1c2c3c4c5c6c7", TAB 8, DEPTNAME;
S>L F 7;

      DEPTNAME
    ________________
      c1c2c3#@INANCE
      c1c2c3@#ERSONNEL
      c1c2c3@@NVENTORY
      c1c2c3c@HIPPING
      c1c2c3@#ARKETING
      c1c2c3##ANADA SALES
      c1c2c3#@ERMNY SALES

   S>
```

Displaying Subtotal Labels With Double-Byte Characters

To identify the break group to which a subtotal corresponds, the report writer prints a subtotal label under the break column. The SUBTOTAL_LABEL option of the SET STYLE command specifies a label for the report writer to print in the break column on the same line as a subtotal. Only one subtotal label can be defined at a time, and the report writer uses that label for all break groups.

When printing subtotal labels with double-byte characters, there are two conditions in which double-byte characters can be split:

- Different break and subtotal column
- Same break and subtotal column

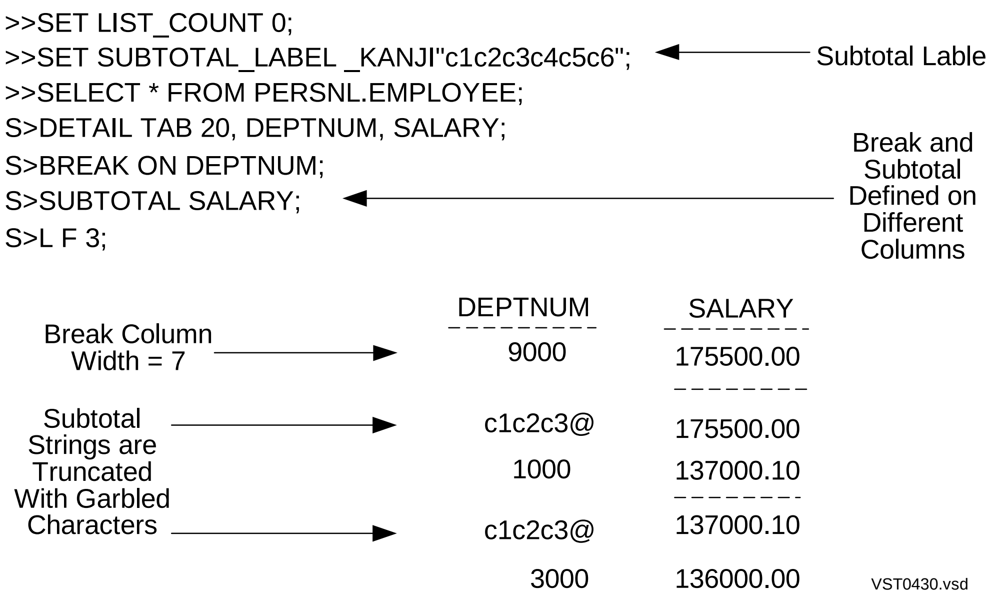
To correct reports that are splitting characters in subtotal labels, you can use either an even-numbered “I” display descriptor, or a correctly-sized “I” display descriptor.

Using Different Break and Subtotal Columns

In cases where the subtotal column is not the same as the break column, and the subtotal label does not fit within the width of the break column, the subtotal label is truncated. The truncation can split a double-byte character.

Figure 4-30 shows an example of what can happen when the subtotal and break columns are different, and data truncation occurs on a field containing double-byte characters.

Figure 4-30. Splitting Characters With Different Break and Subtotal Columns



```
>>SET LIST_COUNT 0;
>>SET SUBTOTAL_LABEL _KANJI"c1c2c3c4c5c6";
>>SELECT * FROM PERSNL.EMPLOYEE;
S>DETAIL TAB 20, DEPTNUM, SALARY;
S>BREAK ON DEPTNUM;
S>SUBTOTAL SALARY;
S>L F 3;

                      DEPTNUM      SALARY
                     ---------   ---------
                        9000     175500.00
                                 --------
                      c1c2c3@    175500.00
                        1000     137000.10
                                 --------
                      c1c2c3@    137000.10
                         3000    136000.00
```

Subtotal Lable
Break and Subtotal Defined on Different Columns
Break Column Width = 7
Subtotal Strings are Truncated With Garbled Characters
VST0430.vsd

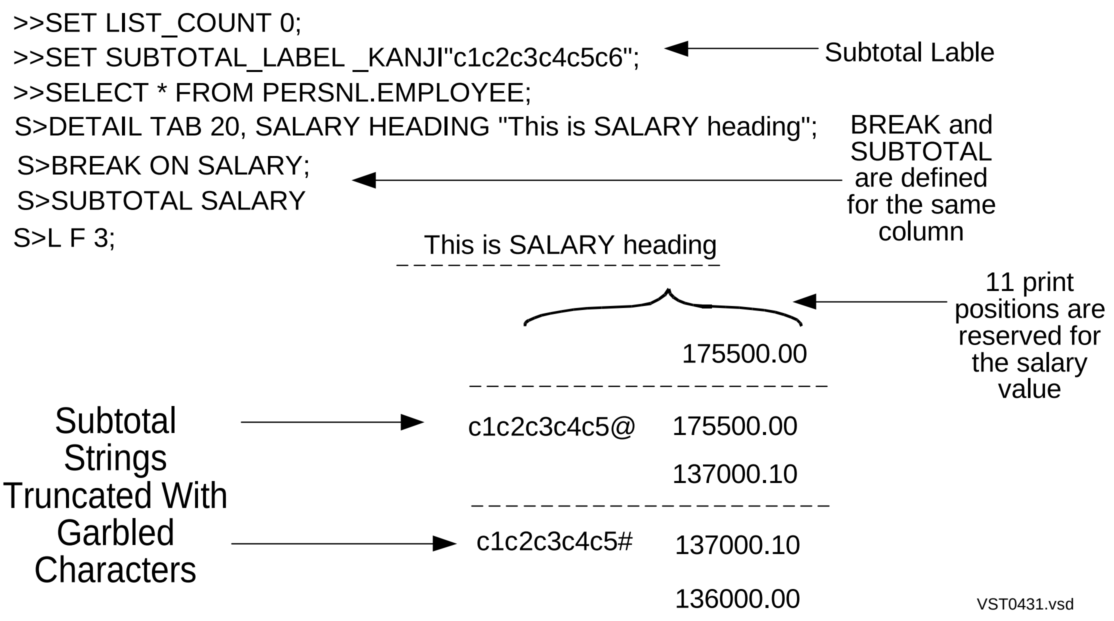
Using the Same Break and Subtotal Columns

In cases where the subtotal column is the same as the break column, the subtotal label and the subtotal value both print under that one column. If the column width is large enough to accommodate both the label and the value, both are printed; otherwise, the label is truncated and can be entirely overwritten by the subtotal value. When the subtotal label is not entirely overwritten, but truncated, character splitting can occur. Figure 4-31 shows an example of this condition.

Suppose for this example that the SALARY column requires that 11 print positions be reserved for displaying values, as defined by the following:

```
salary numeric (8,2) unsigned
```

Figure 4-31. Splitting Characters With Same Break and Subtotal Columns



```
>>SET LIST_COUNT 0;
>>SET SUBTOTAL_LABEL _KANJI"c1c2c3c4c5c6";
>>SELECT * FROM PERSNL.EMPLOYEE;
S>DETAIL TAB 20, SALARY HEADING "This is SALARY heading";
S>BREAK ON SALARY;
S>SUBTOTAL SALARY
S>L F 3;
                                   This is SALARY heading
                                   --------------------

                                         175500.00
                              ---------------------
                              c1c2c3c4c5@ 175500.00
                                          137000.10
                              ---------------------
                              c1c2c3c4c5# 137000.10
                                          136000.00
```

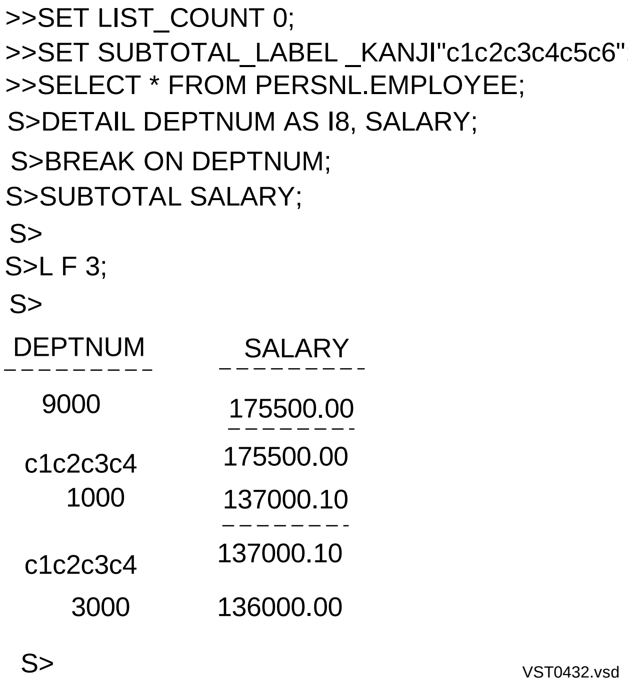
Subtotal Lable
BREAK and SUBTOTAL are defined for the same column
11 print positions are reserved for the salary value
Subtotal Strings Truncated With Garbled Characters
VST0431.vsd

Using Even-Numbered Display Descriptors for Subtotal Column Labels

Figure 4-32 demonstrates how to use an even-numbered “I” display descriptor to format a report. In this example, the break column is different from the subtotal column, but you can use the same techniques as in the previous scenario (where the break column is the same as the subtotal column).

The addition of the I8 display descriptor adjusts the DEPTNUM column so that it occupies eight print positions (an even number) instead of seven.

Figure 4-32. Using an Even-Numbered Display Descriptor for Subtotal Labels



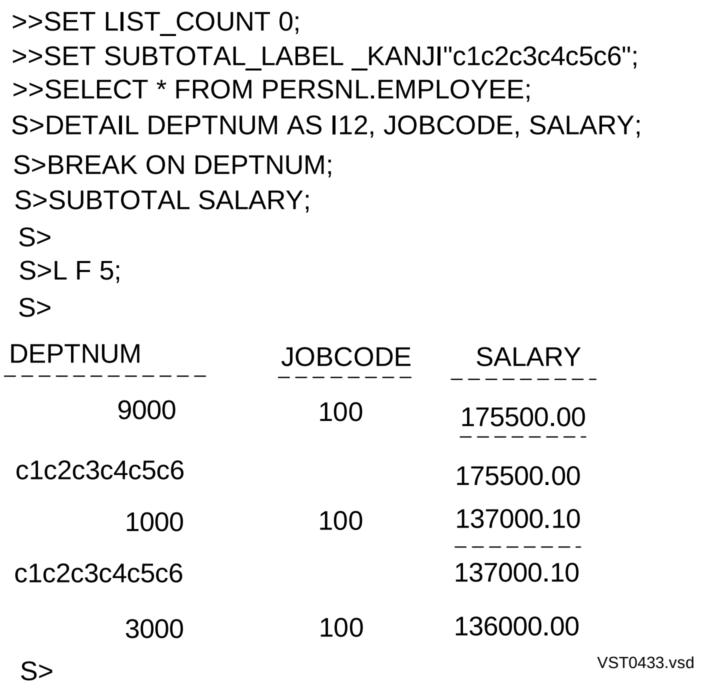
```
>>SET LIST_COUNT 0;
>>SET SUBTOTAL_LABEL _KANJI"c1c2c3c4c5c6";
>>SELECT * FROM PERSNL.EMPLOYEE;
S>DETAIL DEPTNUM AS I8, SALARY;
S>BREAK ON DEPTNUM;
S>SUBTOTAL SALARY;
S>
S>L F 3;
S>
 DEPTNUM        SALARY
---------      ---------
   9000        175500.00
               --------
 c1c2c3c4      175500.00
     1000      137000.10
               --------
 c1c2c3c4      137000.10
     3000      136000.00

S>
```

Using Correctly-Sized “I” Display Descriptors With Subtotal Column Labels

Figure 4-33 demonstrates how to use the “I” display descriptor with a number that is larger than the column in order to accommodate the entire subtotal string.

The addition of the I12 display descriptor adjusts the DEPTNUM column so that it occupies twelve print positions (an even number) instead of seven.

Figure 4-33. Using a Correctly-Sized Display Descriptor for Subtotal Column Lables



```
>>SET LIST_COUNT 0;
>>SET SUBTOTAL_LABEL _KANJI"c1c2c3c4c5c6";
>>SELECT * FROM PERSNL.EMPLOYEE;
S>DETAIL DEPTNUM AS I12, JOBCODE, SALARY;
S>BREAK ON DEPTNUM;
S>SUBTOTAL SALARY;
S>
S>L F 5;
S>
DEPTNUM                JOBCODE    SALARY
------------           --------  ---------
        9000              100     175500.00
                                  --------
c1c2c3c4c5c6                      175500.00
        1000              100     137000.10
                                  --------
c1c2c3c4c5c6                      137000.10
        3000              100     136000.00
S>
```

VST0433.vsd

A

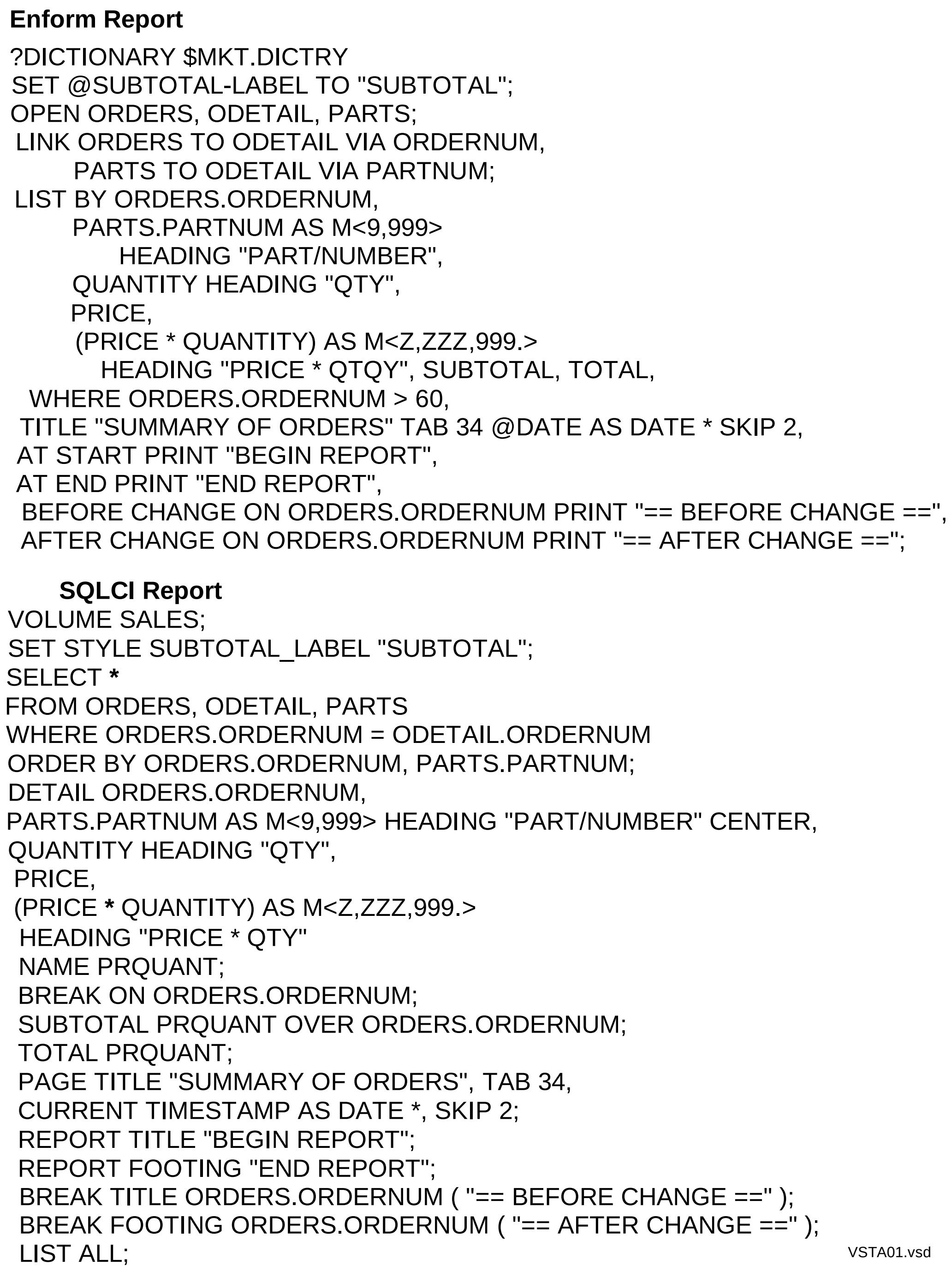
Comparison of the Report Writer and the Enform Language

The NonStop SQL/MP report writer is used to produce reports based on data from a NonStop SQL/MP database. Enform is a language that produces reports based on data from a database that uses Enscribe files. This appendix compares features of the two report writing tools.

Figure A-1 shows a sample Enform program followed by the SQLCI commands that produce the same report. There are many similarities between the two methods for producing reports.

Table A-1 lists Enform statements with the SQLCI and report writer features that you use to perform similar operations. Table A-2 lists Enform clauses and the corresponding SQLCI or report writer features. Table A-3 lists Enform option variables and the corresponding SQLCI or report writer features. Table A-4 lists Enform system variables. Table A-5 lists Enform commands.

Figure A-1. Enform Report and SQLCI Report



Enform Report

```
?DICTIONARY $MKT.DICTRY
SET @SUBTOTAL-LABEL TO "SUBTOTAL";
OPEN ORDERS, ODETAIL, PARTS;
 LINK ORDERS TO ODETAIL VIA ORDERNUM,
        PARTS TO ODETAIL VIA PARTNUM;
 LIST BY ORDERS.ORDERNUM,
        PARTS.PARTNUM AS M<9,999>
             HEADING "PART/NUMBER",
        QUANTITY HEADING "QTY",
        PRICE,
        (PRICE * QUANTITY) AS M<Z,ZZZ,999.>
            HEADING "PRICE * QTQY", SUBTOTAL, TOTAL,
   WHERE ORDERS.ORDERNUM > 60,
  TITLE "SUMMARY OF ORDERS" TAB 34 @DATE AS DATE * SKIP 2,
  AT START PRINT "BEGIN REPORT",
  AT END PRINT "END REPORT",
  BEFORE CHANGE ON ORDERS.ORDERNUM PRINT "== BEFORE CHANGE ==",
  AFTER CHANGE ON ORDERS.ORDERNUM PRINT "== AFTER CHANGE ==";
```

SQLCI Report

```
VOLUME SALES;
SET STYLE SUBTOTAL_LABEL "SUBTOTAL";
SELECT *
FROM ORDERS, ODETAIL, PARTS
WHERE ORDERS.ORDERNUM = ODETAIL.ORDERNUM
ORDER BY ORDERS.ORDERNUM, PARTS.PARTNUM;
DETAIL ORDERS.ORDERNUM,
PARTS.PARTNUM AS M<9,999> HEADING "PART/NUMBER" CENTER,
QUANTITY HEADING "QTY",
 PRICE,
 (PRICE * QUANTITY) AS M<Z,ZZZ,999.>
  HEADING "PRICE * QTY"
  NAME PRQUANT;
  BREAK ON ORDERS.ORDERNUM;
  SUBTOTAL PRQUANT OVER ORDERS.ORDERNUM;
  TOTAL PRQUANT;
  PAGE TITLE "SUMMARY OF ORDERS", TAB 34,
  CURRENT TIMESTAMP AS DATE *, SKIP 2;
  REPORT TITLE "BEGIN REPORT";
  REPORT FOOTING "END REPORT";
  BREAK TITLE ORDERS.ORDERNUM ( "== BEFORE CHANGE ==" );
  BREAK FOOTING ORDERS.ORDERNUM ( "== AFTER CHANGE ==" );
  LIST ALL;
```

VSTA01.vsd

Table A-1. Enform Statements

Enform Statements	SQLCI With Report Writer
LIST	SELECT, DETAIL, and LIST commands
FIND	SELECT and LIST commands (1)
CLOSE	RESET PARAM command (1)
	CANCEL command
DECLARE	No user variables, user aggregates, (2) or user tables; can use aggregates in SELECT command for grouped rows only
DELINK	(2)
DICTIONARY	VOLUME command (1)
EXIT	EXIT command
LINK	SELECT...FROM table1, table2,...WHERE
LINK OPTIONAL	SELECT...FROM table1 LEFT JOIN table2 ON...
OPEN	(2)
PARAM	SET PARAM command (set in SQLCI, command interpreter params passed automatically)
SET	SET LAYOUT | STYLE | SESSION | PARAM (no user variables or user tables)
	—parameters allowed in SELECT command only
AT END	REPORT FOOTING and TOTAL commands
	(only current report)
AT START	REPORT TITLE command (current report)
FOOTING	PAGE FOOTING command (current report)
SUBFOOTING	(2)
SUBTITLE	(2)
TITLE	PAGE TITLE command (current report)

1. No comparable feature
2. Provides part of the functionality

Table A-2. Enform Clauses (page 1 of 2)

Enform Clauses	SQLCI With Report Writer
AFTER CHANGE	BREAK TITLE command
AS, AS DATE, AS TIME	AS and AS DATE/TIME clauses (use in print-item)
ASCD and DESC	SELECT .. ORDER BY..
AT END PRINT	REPORT FOOTING command (no default)
AT START PRINT	REPORT TITLE command (no default)

Table A-2. Enform Clauses (page 2 of 2)

Enform Clauses	**SQLCI With Report Writer**
BEFORE CHANGE	BREAK FOOTING command
BY and BY DESC	SELECT..GROUP BY..ORDER BY.. and BREAK ON command
CENTER	Only for column headings, titles, (1) footings, and the whole report body
CUM and CUM OVER	(1)
FOOTING	PAGE FOOTING command
FORM	PAGE and NEED clauses of print-item
HEADING	HEADING clause in DETAIL line
INTERNAL JULIAN-DATE	(1)
NOHEAD	NOHEAD clause in DETAIL line
NOHEAD ALL	HEADINGS style option OFF
NOPRINT	IF/THEN/ELSE or AS clause or item omitted from DETAIL command but included in ORDER BY clause
NOPRINT ALL	DETAIL
PCT	‘Multistep queries
PCT OVER	Multistep queries
SKIP	SKIP clause of *print-item*
SPACE	SPACE clause of *print-item*
SUBFOOTING	(1)
SUBTITLE	(1)
SUBTOTAL	SUBTOTAL command (based on BREAK command)
SUPPRESS	IF/THEN/ELSE clause of DETAIL command
TAB	TAB clause of *print-item*
TIMESTAMP-DATE	AS DATE/TIME clause
TIMESTAMP-TIME	
TITLE	PAGE TITLE command (no default)
TOTAL	TOTAL command
WHERE	SELECT command WHERE clause
(1) No comparable feature	

Table A-3. Enform Option Variables

Enform Option Variable	SQLCI With Report Writer
@BLANK-WHEN-ZERO	AS clause with BZ modifier (2)
@BREAK-KEY	SET SESSION BREAK_KEY command
@CENTER-PAGE	CENTER_REPORT layout option
@COPIES	OUT_REPORT..SPOOL3 option
@COST-TOLERANCE	(1)
@DATE-FORMAT	DATE_FORMAT style option
@DECIMAL	DECIMAL_POINT style option
@DISPLAY-COUNT	LIST_COUNT session option
@HEADING	HEADINGS style option
@LINES	PAGE_LENGTH layout option
@MARGIN	LEFT_MARGIN layout option
@NEWLINE	NEWLINE_CHAR style option
@NONPRINT-REPLACE	(1)
@OVERFLOW	OVERFLOW_CHAR and TRUNCATE style options
@PAGES	PAGE_COUNT layout option
@PRIMARY-EXTENT-SIZE	(1)
@READS	(1)
@SECONDARY-EXTENT-SIZE	(1)
@SPACE	SPACE layout option
@STATS	DISPLAY STATISTICS and STATISTICS option (2)
@SUBTOTAL-LABEL	SUBTOTAL_LABEL style option
@SUMMARY-ONLY	Only by SELECT ...GROUP BY with aggregate functions in the select list
@TARGET-RECORDS	(1)
@TIME-FORMAT	TIME_FORMAT style option
@UNDERLINE	UNDERLINE_CHAR style option
@VSPACE	LINE_SPACING layout option
@WARN	WARNINGS session option
@WIDTH	RIGHT_MARGIN style option

(1) No comparable feature
(2) Provides part of the functionality

Table A-4. Enform System Variables

Enform System Variable	SQLCI With Report Writer
@DATE	CURRENT_TIMESTAMP or CURRENT function
@TIME	CURRENT_TIMESTAMP or CURRENT function
@LINENO	LINE_NUMBER function
@PAGENO	PAGE_NUMBER function

Table A-5. Enform Commands

Enform Command	SQLCI Command
?ASSIGN	ADD DEFINE and ALTER DEFINE
?ATTACH	(1)
?COMPILE	PREPARE (2)
?DICTIONARY	VOLUME (2)
?EDIT	EDIT and TEDIT
?EXECUTE	EXECUTE (2)
?HELP	HELP
?OUT	OUT_REPORT
?RUN	OBEY
?SECTION	?SECTION
?SHOW	ENV and SHOW
?SOURCE	(1)

(1) No comparable feature

(2) Provides part of the functionality

SQLCI and the report writer provide some features not available in Enform:

- NAME command Assigns a name to a select list item for use in other commands.
- CONCAT clause Concatenates print items, optionally stripping trailing blanks.
- NEED clause Prints lines conditionally depending on space remaining on current page.
- VARCHAR_WIDTH option Specifies the maximum number of variable-width characters to be printed or displayed in a field.
- LOGICAL_FOLDING option Specifies whether items (in the default DETAIL line) that do not fit within the margins are split or moved as a whole unit to the next line.
- WINDOW command Defines a window to view a vertical segment of a report.
- WRAP option Determines whether output lines that do not fit within the device width are wrapped or truncated.

Note that some aggregate functions are available within the SELECT command select list for grouped values. There are no user or target aggregates. Although SQLCI and the report writer do not directly provide the capability of multilevel group aggregates or conditional aggregates, you can perform these operations using multistep queries. For more information on examples of performing these operations in SQLCI, see Developing Multistep Queries on page 3-32.

In SQLCI, user-defined parameters are not allowed in report formatting commands.

Index