HP NonStop SQL/MX Report Writer Guide

Abstract

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

Product Version

NonStop SQL/MX Release 2.0

Supported Release Version Updates (RVUs)

This publication supports G06.23 and all subsequent G-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
527194-002	August 2004

Document History

Part Number	Product Version	Published
527194-001	NonStop SQL/MX Release 2.0	April 2004
527194-002	NonStop SQL/MX Release 2.0	August 2004

HP NonStop SQL/MX Report Writer Guide

Index | Figures | Tables

What's New in This Manual vii
- Manual Information vii
- New and Changed Information vii

About This Manual ix
- Audience ix
- Further Reading ix
- How to Use This Manual ix
- Related Documentation x
- Notation Conventions xii

1. Introduction to the NonStop SQL/MX Report Writer

Accessing Data From SQL/MP Tables 1-1
Database Organization 1-2
- SQL Database Reports 1-3
- Report Design 1-3

Report Development Steps 1-4
Report Layout and Style 1-5
- Default Report Format 1-6
- Default Layout and Style Options 1-7
- Customized Reports 1-8

Report Writer Components 1-12
- Print Items and Logical Lines 1-13
- Report Formatting Commands 1-14
- Report Formatting Command Clauses 1-15
- Style Options 1-15
- Layout Options 1-16
- Report Functions 1-17
- MXCI Commands 1-17

2. Using MXCI and the Report Writer

Setting the Default Directory 2-1
Starting and Ending an MXCI Session 2-2
The Select-in-Progress Prompt 2-2
Setting Up Your Session Environment 2-2
The Elements of Your Session Environment 2-3
Invoking Report Writer Mode 2-4
Setting and Displaying Options 2-5
Listing Rows of a Report 2-6
Canceling a SELECT Command 2-7
Defining Options for Line Folding 2-7
Defining a Window for Report Output 2-8
Entering Report Formatting Commands 2-10
Specifying Log Files 2-13
Executing a Command Repeatedly 2-13
Repeating Stored Commands 2-14
Defining Reports in Command Files 2-15
Creating a Command File With a Text Editor 2-16
Creating a Command File From a Log File 2-16

3. Selecting Data for a Report

Data Selection 3-1
Developing a Query 3-2
Locating the Data 3-2
Joining Tables 3-4
Using the WHERE Clause 3-4
Selecting Columns of Source Data 3-5
Setting Criteria for Selecting Data 3-6
Comparing a Set of Columns to a Set of Values 3-10
Comparing Character Values 3-11
Specifying More Than One Condition 3-14
Sorting the Data 3-15
Grouping Data for Calculations 3-16
Counting Rows 3-18
Determining Minimum and Maximum Values 3-19
Determining Which Columns to Group 3-20
Selecting Distinct Rows 3-21
Using Expressions to Calculate Report Values 3-22
Using Parameters With SELECT Commands 3-24

Preparing a SELECT Command 3-25
Using Views 3-26
Using Subqueries 3-28
Developing Multistep Queries 3-32
Multilevel Group Aggregates 3-33
Conditional Aggregates 3-34
Row Value as Percent of All Row Values 3-35

4. Customizing a Report

Defining the Layout 4-2
Setting Margins 4-2
Paginating 4-6
Spacing Items and Lines 4-10
Specifying the Items in a Detail Line 4-12
Naming Select List and Detail Line Items 4-14
Organizing Rows Into Break Groups 4-15
Labeling Information 4-17
Specifying Column Headings 4-17
Headings for Column Identifier Print Items 4-17
Multiple-Line Headings 4-19
Specifying Titles 4-20
Page and Report Titles 4-20
Break Titles 4-22
Specifying Footings 4-24
Page and Report Footings 4-24
Break Footings 4-26
Line Numbers 4-27
Text Inserted Into a Line 4-30
Formatting Data Values 4-30
Using the AS Clause to Modify the Default Report Format 4-32
Display Format Specifications 4-32
Numeric Values 4-34
Monetary Values 4-35
Suppressing Leading or Trailing Zeros 4-38
Concatenating Text 4-39
Truncated Values 4-40
Filler Characters 4-40

Formatting Dates and Times 4-42
Date and Time Values 4-42
Julian Timestamp Formats 4-43
SQL/MX Date and Time Formats 4-44
Converting Timestamps 4-45
Formats for Dates Stored as Binary Values 4-46
Conditional Printing of Items or Line Entries 4-46
Redefining Special Characters 4-48
Calculating Totals 4-49
Calculating Subtotals 4-51
Defining a Subtotal Label 4-55
Calculating Subtotals on Conditional Values 4-56
Printing Double-Byte Characters 4-58

A. Migration from SQL/MP Report Writer to SQL/MX Report Writer

Unsupported Commands, Options, and Functions A-1
Changed Commands and Options A-1
Error Handling A-2
Single and Double Quotes A-2

Index

Figures

Figure i. NonStop SQL/MX Library Map xii
Figure 1-1. Sample Content of Tables 1-2
Figure 1-2. Default Report Layout 1-6
Figure 1-3. Commands to Produce A Formatted Report 1-10
Figure 1-4. Formatted Report, page 1 1-11
Figure 1-5. Formatted Report, page 2 1-12
Figure 2-1. Setting and Displaying Options 2-5
Figure 2-2. Listing Rows 2-7
Figure 2-3. Defining a Window Using SET LAYOUT WINDOW 2-8
Figure 2-4. Relation of Window to Output Line 2-10
Figure 2-5. Report Formatting Commands 2-11
Figure 2-6. Repeating Stored Commands 2-15
Figure 3-1. Plan for Report Content 3-3
Figure 3-2. Sample Rows from Joined Tables 3-5
Figure 3-3. Rows Sorted by One Column 3-15
Figure 3-4. Rows Sorted by Two Columns 3-16

Figure 3-5. Expressions in the Select List 3-23
Figure 3-6. Expressions in the Detail Line 3-24
Figure 3-7. An Invoice 3-28
Figure 3-8. A Report With a Subquery 3-32
Figure 4-1. Default Margin Settings in a Displayed Report 4-3
Figure 4-2. Default Margin Settings in a Printer Report 4-4
Figure 4-3. Pagination Features 4-8
Figure 4-4. Result of Using the NEED Clause 4-9
Figure 4-5. Tabbing to a Display or Print Position 4-12
Figure 4-6. Break Groups 4-16
Figure 4-7. Report and Page Titles 4-21
Figure 4-8. A Page Title With a Row Value 4-22
Figure 4-9. Break Titles 4-24
Figure 4-10. Location of Report and Page Footings 4-25
Figure 4-11. A Page Footing 4-26
Figure 4-12. Break Footings 4-27
Figure 4-13. Numbering Lines 4-29
Figure 4-14. Numbering Break Group Lines 4-30
Figure 4-15. Monetary Total and Subtotals 4-36
Figure 4-16. Sample Decoration 4-37
Figure 4-17. A Report With Totals 4-50
Figure 4-18. Formatted Total Values 4-51
Figure 4-19. A Report With Subtotals 4-53
Figure 4-20. A Report With Subtotals (Page 2 of 2) 4-54
Figure 4-21. Customer Orders Summary 4-56
Figure 4-22. Subtotals on Conditional Values 4-58

Tables

Table 1-1. Report Formatting Commands 1-14
Table 1-2. Report Formatting Command Clauses 1-15
Table 1-3. Style Options 1-15
Table 1-4. Layout Options 1-16
Table 1-5. Report Functions 1-17
Table 1-6. MXCI Commands 1-17
Table 3-1. Search Condition Predicates 3-8
Table 3-2. Comparison and LIKE Predicates 3-13
Table 4-1. Default Display Formats 4-31
Table 4-2. Format Characters in AS DATE and AS TIME Clauses 4-44
Table 4-3. DATEFORMAT Display Formats 4-44

What's New in This Manual

Manual Information

Abstract

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

Product Version

NonStop SQL/MX Release 2.0

Supported Release Version Updates (RVUs)

This publication supports G06.23 and all subsequent G-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
527194-002	August 2004

Document History

Part Number	Product Version	Published
527194-001	NonStop SQL/MX Release 2.0	April 2004
527194-002	NonStop SQL/MX Release 2.0	August 2004

New and Changed Information

Formatting of example corrected in Section 4, Customizing a Report.

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/MX database. The manual also provides an overview of the report writer commands and MXCI 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 programming assistance to create reports using advanced features.

Further Reading

Read the *SQL/MX 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/MX Report Writer	Introduces the report writer, describes its components, and compares the default report format with a custom formatted report.
Section 2, Using MXCI and the Report Writer	Gives instructions for running MXCI and using report writer commands. You can define and produce reports in various ways.
Section 3, Selecting Data for a Report	Shows 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, Migration from SQL/MP Report Writer to SQL/MX Report Writer	Lists the SQL/MX report writer features that are different from SQL/MP report writer.

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/MX Reference Manual*, which fully documents the HP implementation of SQL.

Related Documentation

This manual is part of the SQL/MX library of manuals, which includes:

Introductory Guides

SQL/MX Comparison Guide for SQL/MP Users	Describes SQL differences between SQL/MP and SQL/MX.
SQL/MX Quick Start	Describes basic techniques for using SQL in the SQL/MX conversational interface (MXCI). Includes information about installing the sample database.

Reference Manuals

SQL/MX Reference Manual	Describes the syntax of SQL/MX statements, MXCI commands, functions, and other SQL/MX language elements.
SQL/MX Connectivity Service Administrative Command Reference	Describes the SQL/MX administrative command library (MACL) available with the SQL/MX conversational interface (MXCI).
DataLoader/MX Reference Manual	Describes the features and functions of the DataLoader/MX product, a tool to load SQL/MX databases.
SQL/MX Messages Manual	Describes SQL/MX messages.
SQL/MX Glossary	Defines SQL/MX terminology.

Programming Manuals

SQL/MX Programming Manual for C and COBOL	Describes how to embed SQL/MX statements in ANSI C and COBOL programs.
SQL/MX Programming Manual for Java	Describes how to embed SQL/MX statements in Java programs according to the SQLJ standard.

Specialized Guides

SQL/MX Installation and Management Guide	Describes how to plan for, install, create, and manage an SQL/MX database. Explains how to use installation and management commands and utilities.
SQL/MX Query Guide	Describes how to understand query execution plans and write optimal queries for an SQL/MX database.
SQL/MX Data Mining Guide	Describes the SQL/MX data structures and operations to carry out the knowledge-discovery process.
SQL/MX Queuing and Publish/Subscribe Services	Describes how SQL/MX integrates transactional queuing and publish/subscribe services into its database infrastructure.

SQL/MX Report Writer Guide	Describes how to produce formatted reports using data from a NonStop SQL/MX database.
SQL/MX Connectivity Service Manual	Describes how to install and manage the SQL/MX Connectivity Service (MXCS), which enables applications developed for the Microsoft Open Database Connectivity (ODBC) application programming interface (API) and other connectivity APIs to use SQL/MX.
SQL/MX Guide to Stored Procedures in Java	Describes how to use stored procedures that are written in Java within SQL/MX.

Online Help

The SQL/MX Online Help consists of:

Reference Help	Overview and reference entries from the *SQL/MX Reference Manual*.
Messages Help	Individual messages grouped by source from the *SQL/MX Messages Manual.*
Glossary Help	Terms and definitions from the *SQL/MX Glossary*.
NSM/web Help	Context-sensitive help topics that describe how to use the NSM/web management tool.
Visual Query Planner Help	Context-sensitive help topics that describe how to use the Visual Query Planner graphical user interface.

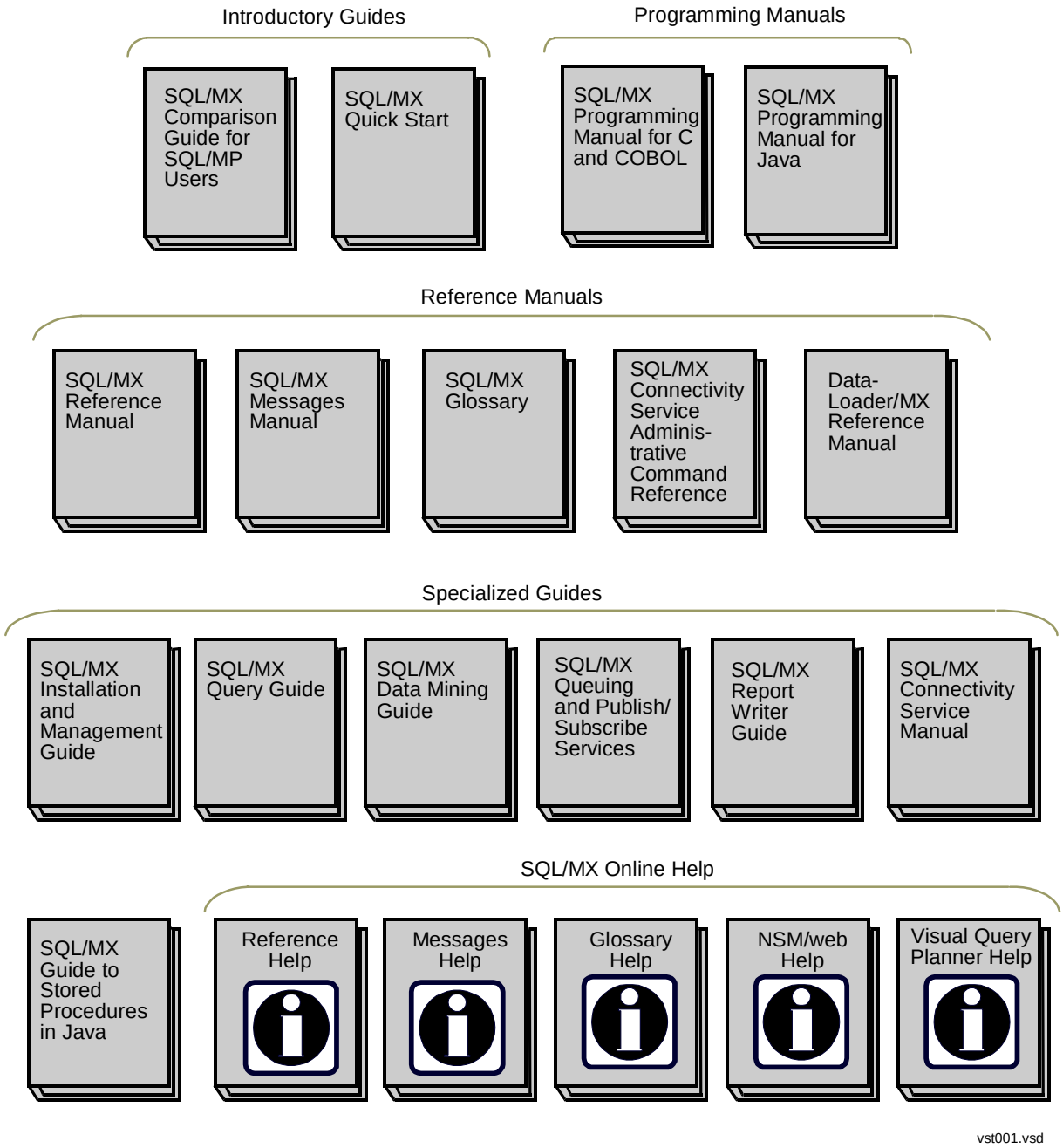
The NSM/web and Visual Query Planner help systems are accessible from their respective applications. You can download the Reference, Messages, and Glossary online help from the $SYSTEM.ZMXHELP subvolume or from the HP NonStop Technical Library (NTL). For more information about downloading online help, see the *SQL/MX Installation and Management Guide*.

The following manuals are part of the SQL/MP library of manuals and are essential references for information about SQL/MP Data Definition Language (DDL) and SQL/MP installation and management:

Related SQL/MP Manuals

SQL/MP Reference Manual	Describes the SQL/MP language elements, expressions, predicates, functions, and statements.
SQL/MP Installation and Management Guide	Describes how to plan, install, create, and manage an SQL/MP database. Describes installation and management commands and SQL/MP catalogs and files.

Figure i. NonStop SQL/MX Library Map



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
```

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1 Introduction to the NonStop SQL/MX Report Writer

Accessing Data From SQL/MP Tables	1-1
Database Organization	1-2
Report Development Steps	1-4
Default Report Format	1-6
Customized Reports	1-8
Report Formatting Commands	1-14
Report Formatting Command Clauses	1-15
Style Options	1-15
Layout Options	1-16
Report Functions	1-17
MXCI Commands	1-17

NonStop SQL/MX is a relational database management system that uses the industry standard SQL language to define and manipulate data. The SQL/MX conversational interface (MXCI) 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 MXCI commands to control your session environment.

Note. The SELECT command referred to in this manual is a SELECT statement you enter by using MXCI in report writer mode.

Examples and database references in this guide refer to the SQL/MX sample database. See the *SQL/MX Reference Manual* for details.

Accessing Data From SQL/MP Tables

SQL/MX Report Writer supports SQL/MP tables. You can access data from SQL/MP tables by creating an alias using the CREATE SQLMP ALIAS statement. Also, SQL/MP tables, views, indexes, and catalogs can be accessed by using SQL/MX DML statements. For details, see the *SQL/MX Reference Manual*.

For information on the SQL/MP language, see the *SQL/MP Reference Manual* .

Database Organization

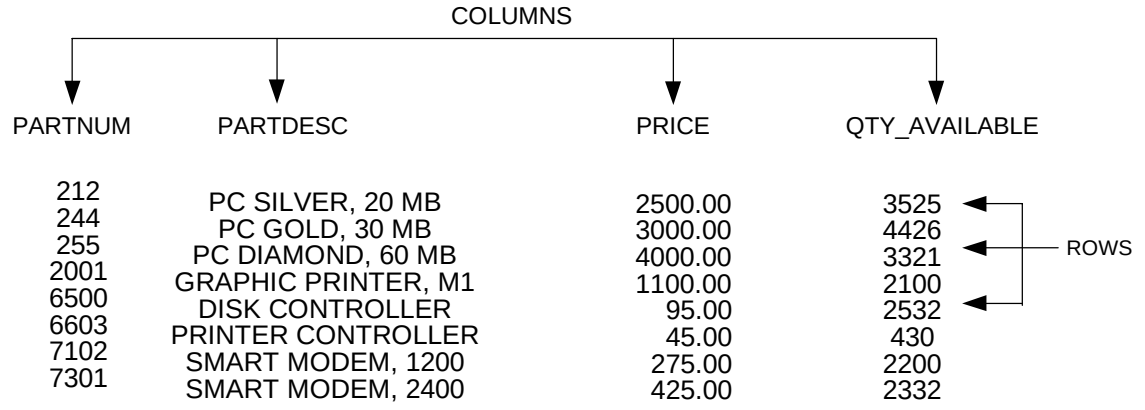
In an SQL/MX database, data is organized and maintained in tables. A table consists of vertical columns and horizontal rows:

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

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



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

VST0101.vsd

SQL Database Reports

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

Layout and style options are global and not related to a specific SELECT command. Your current layout and style options affect the output of any SELECT command you enter during your report writer session.

Layout Options

Layout options control general properties of reports, such as page length, margin settings, and the appearance of the entire report on the screen or printed page.

Style Options

Style options, such as the decimal character, underline character, or format of dates, control the appearance of individual items in a report.

Report Formatting Commands

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.

Report Design

To display or print data retrieved by the SELECT command, use the default report definition or create a report definition of your own.

To use the default report format, accept the default style options, layout options, and report formatting commands. See Default Report Format on page 1-6 for a report created with the default report definition.

To design your own report, change the current report definition by entering report formatting commands or by using the SET LAYOUT or SET STYLE commands to change the default values.

Note. The current report definition determines the format of your report. A layout or style option stays in effect until you set it to some other value, reset it to its default value, or exit report writer mode.

Use the report writer features, the SELECT command, and the LIST command to 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 enter the LIST ALL command to produce the report. To print the report, you must log your session and print the file from HP NonStop Kernel Open System Services (OSS).

Report Development Steps

The steps for using the MXCI report writer to develop a report are described in detail in the remainder of this guide. This summary provides an overview:

1. Setting report writer mode
   Before developing a report, you must set report writer mode .

   Note. All the examples in this guide assume that you have set report writer mode before executing any report writer commands.

2. Selecting data
   To produce a report, you must execute a SELECT command. In the SELECT command:

   - Specify the data to be retrieved.
   - Specify the tables that contain the data.
   - Specify the criteria for selecting rows and joining rows from different tables and views.

   See the *SQL/MX Query Guide* and the *SQL/MX 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 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/MX Reference Manual* for information about the PREPARE command and EXPLAIN utility.

3. Formatting a 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. See Default Report Format on page 1-6 for a report created with the default report format.

   To produce more elaborate reports, use report formatting commands to:

   - Specify the content of detail lines, titles, footings, and column headings.
   - 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.
   - Calculate the total value of a column over all rows in the report.

To change the appearance of a report, use the layout options to determine:

- Margins
- The number of lines per page
- Line spacing
- Whether default headings are to print above columns of data

4. Customizing the style
   Use the style options to customize:

   - Subtotal labels
   - The style of date and time formats
   - Special characters representing the underline and decimal point

5. Displaying and printing a report
   To display or print all or part of the report, use the LIST command.

   The LIST command displays the report on your terminal (default standard output).

   To print a report, you must log your session to a file and then print the file from OSS. For more information, see Specifying Log Files **on page 2-13**.

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. Options apply to any report you print while the option is set. For example, the default value for the PAGE_COUNT layout option is ALL, which 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 these commands at the MXCI 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 MXCI Session on page 2-2.)

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.

You can use the HISTORY, FC, or exclamation point (!) command to reexecute report commands executed earlier in the current session.

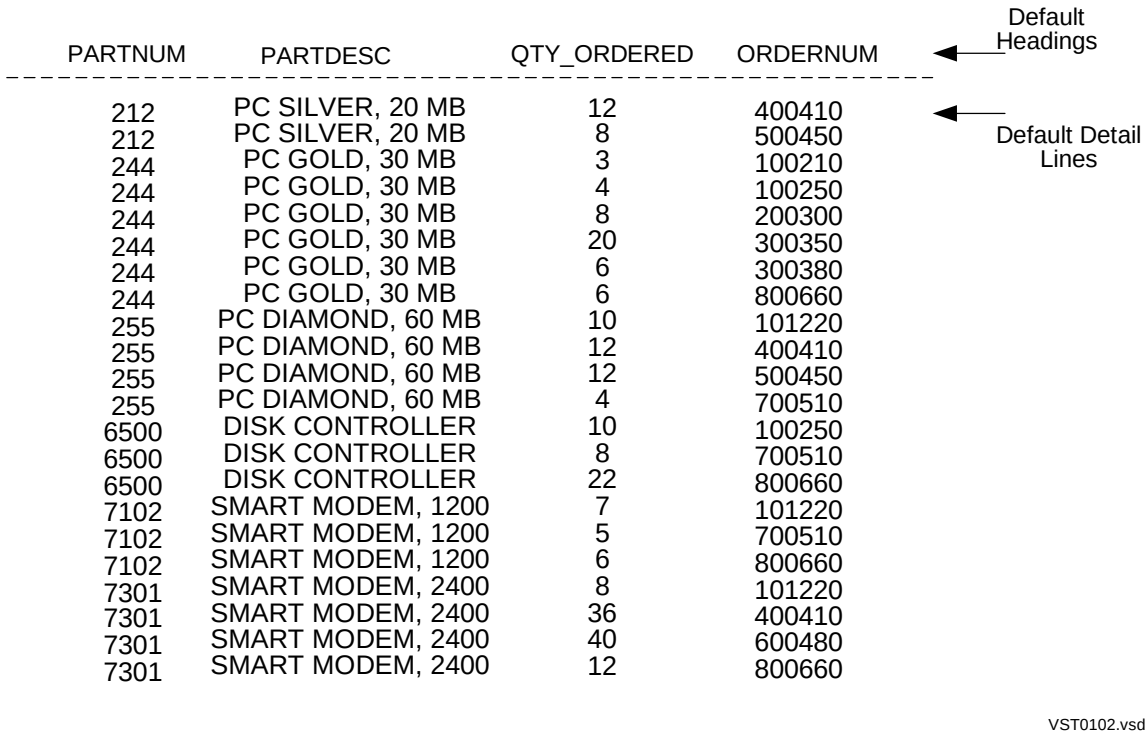
Default Report Format

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*. The tables referred to are in the same catalog and schema, so your default catalog and schema should have been set at the beginning of this session. See Setting Up Your Session Environment on page 2-2 for details on setting the default catalog and schema.

Figure 1-2 shows a default report produced with these commands:

```
>> MODE REPORT;
>> 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. Default Report Layout



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.

The components of the default report are:

- Detail lines
  - Each detail line contains all elements of the select list in the order you specify in the SELECT command. (In Figure 1-2, the columns of data correspond to the data items as they were listed in the SELECT command.)
  - Values are displayed in the default display format, which is determined by the data type defined for the column in the table definition.
  - The width of a report column is equal to the heading or item width, depending on which is wider. 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 according to the current setting of the LOGICAL _FOLDING option.
- Headings
  - Each element that is a table or view column has a column heading consisting of a heading or column name as defined in the catalog. An element that is an expression or numeric parameter has the default heading (EXPR). Headings are underlined and separated from the detail lines by a blank line.

Default Layout and Style Options

For a complete summary for all layout options, see Layout Options on page 1-16.

Report Element	Default Setting	Result	Option to Set
Left margin	0 (zero)	The output line begins at the left edge of the screen or printed page.	LEFT_MARGIN
Right Margin	Width of the output device	The output line ends at the right edge of the screen or printed page.	RIGHT_MARGIN
Page length	60	Displays/prints the number of lines to which PAGE_LENGTH is set.	PAGE_LENGTH
Top margin	1 (one)	Displays/prints one blank line at top of report.	PAGE TITLE
Bottom margin	1 (one)	Displays/prints one blank line at bottom of report.	PAGE FOOTING

For a summary of layout and style options, which includes the default setting of each option, see Report Writer Components on page 1-12. For detailed information about layout and style options, see the *SQL/MX Reference Manual*.

Customized Reports

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

- Page title: A title appears at the top of each page.
- Report title: The main title of the report appears at the top of only 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:
  - 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 report formatting commands, see Report Writer Components on page 1-12.

A Sample Formatted Report

Figure 1-3 shows commands to produce a formatted report. Figure 1-4 and Figure 1-5 show the two pages of the resulting report.

The report contains:

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 (RIGHT_MARGIN)

Figure 1-3. Commands to Produce A Formatted Report

```
>>MODE REPORT;
>> SET LAYOUT RIGHT_MARGIN 60;
>> SELECT P.PARTNUM, PARTDESC, PRICE, UNIT_PRICE,
     +> 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> 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

Figure 1-4. Formatted Report, page 1

Summary of Orders 09/25/03 (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. Formatted Report, page 2

```
Summary of Orders                                        09/25/03        (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 MXCI commands that are useful when generating reports, the report writer has these 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, see the *SQL/MX 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 one of these:

- A column identifier
- A literal
- An arithmetic expression
- 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/MX 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:

Column Identifier	Description
Column name	The name of a column described in the select list. ● Must be qualified if it is the same as another unqualified name in the select list (for example, EMPLOYEE.DEPTNUM and DEPT.DEPTNUM). ● If only one name is in the select list (for example, EMPLOYEE.DEPTNUM), you can omit the qualifier.

Column number	Specifies the column in a numeric position of the select list The first item in the select list is COL 1. Use this form to refer to literals and expressions.
Alias name	A name defined in a NAME command Use an alias name to refer to a name you assign to a literal or expression.
Detail alias name	A name defined in the NAME clause of a DETAIL command 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.

AL 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. Use *n* as the field width for a print item formatted with the AS clause descriptor An.w. For example, if a VARCHAR column named JOBRESP is formatted with AS A0.30, 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. They 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 (page 1 of 2)

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

Table 1-1. Report Formatting Commands (page 2 of 2)

Command	What the Command Defines
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

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
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/MX 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 (page 1 of 2)

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

Table 1-3. Style Options (page 2 of 2)

Style Option	Default	What the Option Defines
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 Functions

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

MXCI Commands

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

Table 1-6. MXCI 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.
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 STYLE	Resets style options to default settings.
SELECT	Retrieves data from tables and views.
SET LAYOUT	Sets layout options to new values.
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 STYLE	Displays the values of style options.

2 Using MXCI and the Report Writer

Setting the Default Directory	2-1
Starting and Ending an MXCI Session	2-2
Setting Up Your Session Environment	2-2
The Elements of Your Session Environment	2-3
Invoking Report Writer Mode	2-4
Setting and Displaying Options	2-5
Entering Report Formatting Commands	2-10
Executing a Command Repeatedly	2-13
Defining Reports in Command Files	2-15

You can control the environment of a report writing session by using general MXCI commands. To develop a report:

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

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

Setting the Default Directory

To set the default directory, enter the `cd` command at the OSS shell prompt:

```
> cd /usr/dirname;
```

Starting and Ending an MXCI Session

To start an MXCI session, at the OSS shell prompt, enter:

```
/home/sql:mxci
Hewlett-Packard NonStop(TM) SQL/MX Conversational Interface 2.0
(c) Copyright 2003 Hewlett-Packard Development Company, LP.
>>
```

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

To end an MXCI session, at the MXCI prompt, enter:

```
>> exit;
End of MXCI Session
```

The Select-in-Progress Prompt

After you enter the SELECT command in REPORT mode, MXCI displays the select-in-progress prompt (S>), indicating that MXCI is waiting for report writer commands.

Setting Up Your Session Environment

Any report you create is affected by the current environment of your MXCI session, such as the current default catalog and schema.

For example, to set a logical name for the default catalog:

```
SET CATALOG SAMDBCAT;
```

To set a logical name for the default schema, use the command:

```
SET SCHEMA SAMDBCAT.SALES;
```

If an object is on the default schema, you can refer to it by the table name only.

```
>>SELECT * FROM CUSTOMER;
```

If you are creating a report that uses data from tables in different schemas, you must refer to the table by a fully qualified name, including the catalog, schema, and table name:

```
>>SELECT * FROM SAMDBCAT.SALES.CUSTOMER;
```

If you have a default schema set, you can refer to data from tables in that schema by table name only. However, you must use the fully qualified name for data from tables in other schemas. You might find it easier in this case not to set a default schema.

The catalog and schema you have set are in effect until the end of your session or until you execute another SET CATALOG or SET SCHEMA command.

Note. All examples in this guide assume that you have already set the correct names for the catalog and schema, as appropriate, at the beginning of your session.

The Elements of Your Session Environment

CURRENT DIRECTORY	Path name of the current server directory. It can be changed by using the cd command.
HOME DIRECTORY	The default directory.
LIST_COUNT	Current setting of this session option.
LOG FILE	File to which MXCI logs your session activity. MXCI 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	Name of the current SQL/MX message file.
MESSAGEFILE LANG	Language of the text in the message file.
MESSAGEFILE VRSN	The version of SQL/MX software in use.
SQL CATALOG	The current default catalog name.
SQL SCHEMA	The current default schema name.
TRANSACTION ID	The transaction identifier of the current TMF transaction, if one is in progress.
TRANSACTION STATE	Transaction status: ● *in progress* — a TMF transaction has been initiated ● *not in progress* — no transaction has been initiated since any previous transaction was committed or aborted.
WARNINGS	Current setting for warnings: ● *On* ● *Off*

Use the ENV command to display the current default settings, as shown:

```
>> env;
---------------------------------
Current Environment
---------------------------------
CURRENT DIRECTORY  /usr/yourdir/bin
HOME DIRECTORY     /usr/yourdir
LIST_COUNT         4294967295
LOG FILE
MESSAGEFILE        /usr/yourdir/bin/mxcierrors.cat
TERMINAL CHARSET   ISO88591
MESSAGEFILE LANG   US English
MESSAGEFILE VRSN   {2003-09-19 09:43 NSK:FIGARO/SUPER.SUPER}
SQL CATALOG        CAT
SQL SCHEMA         SCH
TRANSACTION ID
TRANSACTION STATE  not in progress
WARNINGS           on
>>
```

Invoking Report Writer Mode

To invoke report writer commands from the MXCI command line, set report writer mode:

```
>> MODE REPORT;
```

To exit report writer mode, enter:

```
>> MODE SQL;
```

To find out which mode is currently set, enter:

```
>> MODE DISPLAY;
```

Note. If you are not in report writer mode, report writer commands entered into the MXCI command line are treated as syntax errors.

Setting and Displaying Options

There are three basic commands for working with options: SET, RESET, and SHOW. There are two types of options you can use to customize your reports:

- 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.

Figure 2-1 illustrates setting and displaying options.

Figure 2-1. 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, to reset the line spacing to 1 (single spacing) and reset all style options to their default values, enter:

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

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

Scenarios throughout this guide illustrate how options affect your reports. For a summary of options, see Report Writer Components on page 1-12.

Listing Rows of a Report

To retrieve rows of data, use the SELECT command.

As shown in Figure 2-2, Listing Rows, on page 2-7, 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.

LIST Command Options

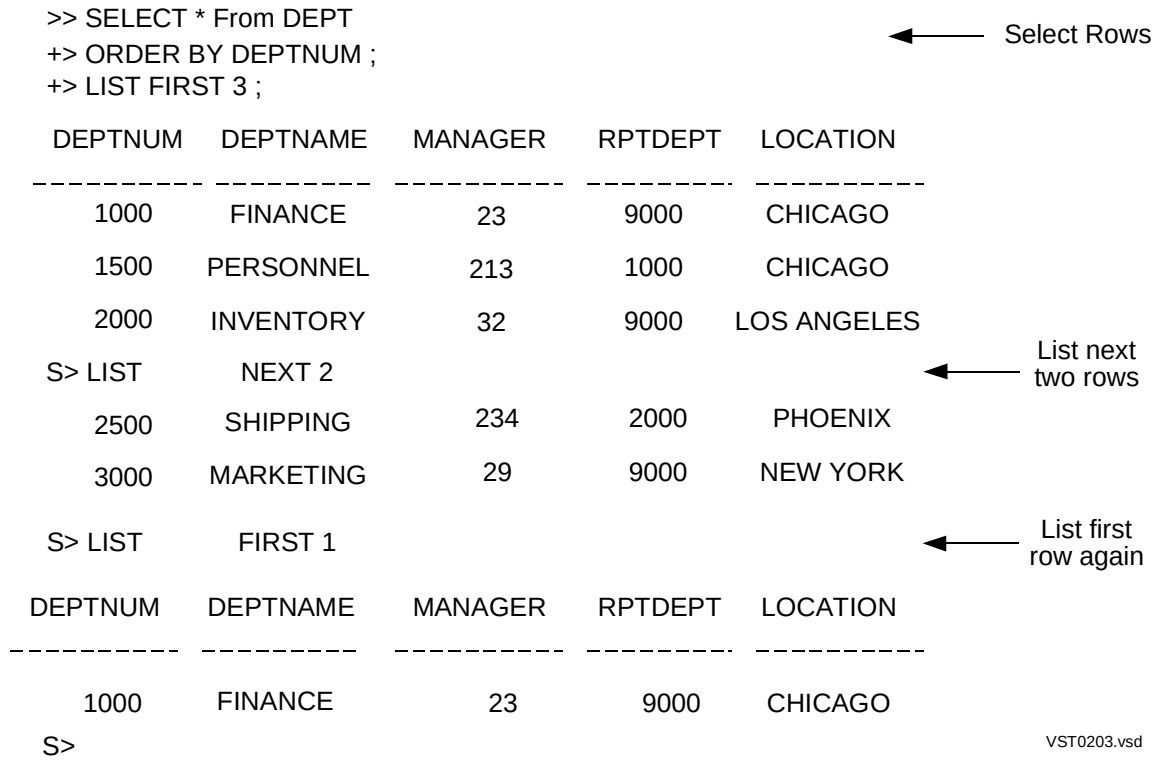
The three options for the LIST command are:

Option	Action
FIRST *n**	Lists the first *n* rows retrieved
NEXT *n**	Lists the next *n* rows from the last displayed row
ALL	Lists all selected rows and then returns you to the standard prompt and cancels the SELECT command

**n* is the current setting for LIST_COUNT.

Note. The number you specify for the value of *n* specifies rows of retrieved data. Therefore, 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.

Figure 2-2. Listing Rows



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 Folding

Line folding occurs when the information in a print list does not fit within the defined margins. In this case, the information is folded to (continued on) the next line.

The LOGICAL_FOLDING layout option controls how information is folded for the default detail line only. Settings for the LOGICAL_FOLDING option are:

ON A single item is never folded to the next line. This is the default setting.

OFF An item is folded to the next line if it does not fit within the margins.

If the information fits within the margins, LOGICAL_FOLDING has no effect.

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, the device determines at what point an output item will wrap to the next line.

Use the SET LAYOUT WINDOW command to specify which vertical portion of the report you want to see. For example, suppose that 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:

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

Use the SET LAYOUT WINDOW command to view different parts of the report. For example, to make the EMPNUM column appear at the left edge of the output, enter WINDOW EMPNUM, as shown in Figure 2-3.

Figure 2-3. Defining a Window Using SET LAYOUT WINDOW

```
S> SET LAYOUT WINDOW EMPNUM;

S> LIST FIRST 1;

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

    23          ROGER           GREEN          1000
S>
```

VST0206.vsd

You could produce the same result as Figure 2-3 by specifying SET LAYOUT WINDOW TAB 60.

To use the TAB form of SET LAYOUT WINDOW:

- Place your cursor at either the standard or the select-in-progress prompt.
- Specify the print position relative to the full output line, including the margin.

  The default first print position is always the same as position LEFT_MARGIN 0. Therefore, 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.

Note that you cannot use the TAB form of SET LAYOUT WINDOW within an IF-THEN-ELSE clause. In that case, the TAB setting is ignored, although an error is not generated.

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:

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

Figure 2-4 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 Step 1, these 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, it continues on the next line at the left margin.
- The default device width for the terminal is 64.
- 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, which is 7 characters wide. The number is right justified.

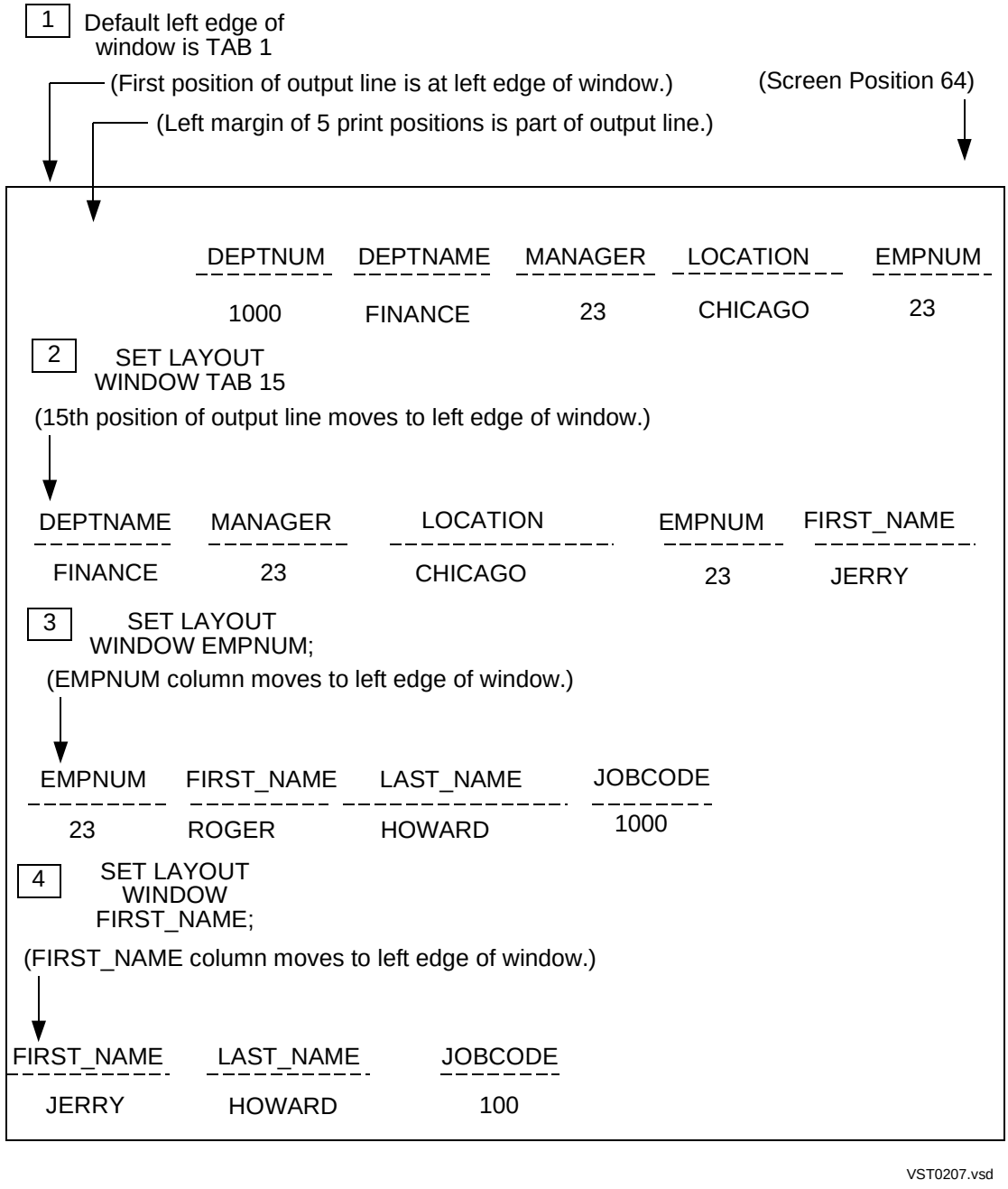
At Step 2, the window is specified to begin at TAB 15.

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

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

For an overview of how to set margins, see Setting Margins on page 4-2.

Figure 2-4. Relation of Window to Output Line



Entering Report Formatting Commands

Report formatting commands must be associated with a specific SELECT command. You can enter these commands only at the select-in-progress prompt (S>).

In Figure 2-5, DETAIL, BREAK ON, and BREAK TITLE are report formatting commands that refer to the selected data.

Figure 2-5. Report Formatting Commands

```
>> 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                                  <-- Break Title
1000          23        HOWARD
              202       CLARK
              208       CRAMER
              210       BARTON
              214       KELLY

FINANCE                                  <-- Break Title
1500          209       CHAPMAN
              211       SCHNEIDER
              212       MITCHELL

S>
```

VST0208.vsd

You can enter only one version of each of these commands 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 that you enter this 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:

```
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, this 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 next example, the SUBTOTAL COL 1 command specifies that column 1 will be subtotaled whenever the value of a 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 these 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 SELECT output, as if you had entered the LIST FIRST command.

Specifying Log Files

If you specify a log file for your session by entering the LOG command, MXCI 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. Optionally, you can specify that you want only the commands written to the log file.

To print a report with the report definition preceding it, you can use the log file.

Suppose the file REPORT55 contains all the commands needed to prepare a report. To include the definition as well as the report, enter:

```
>> 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.

To turn logging off, enter:

```
>> LOG;
```

Print the file REPORT55 from OSS using the *lp* command. For information about the *lp* command, see the *Open System Services User's Guide*.

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 3-25.)
- Use the OBEY command.

To execute a set of commands repeatedly, use a text editor to put the commands in 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 MXCI 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:

```
>> ! SELECT P
```

MXCI displays and executes the command:

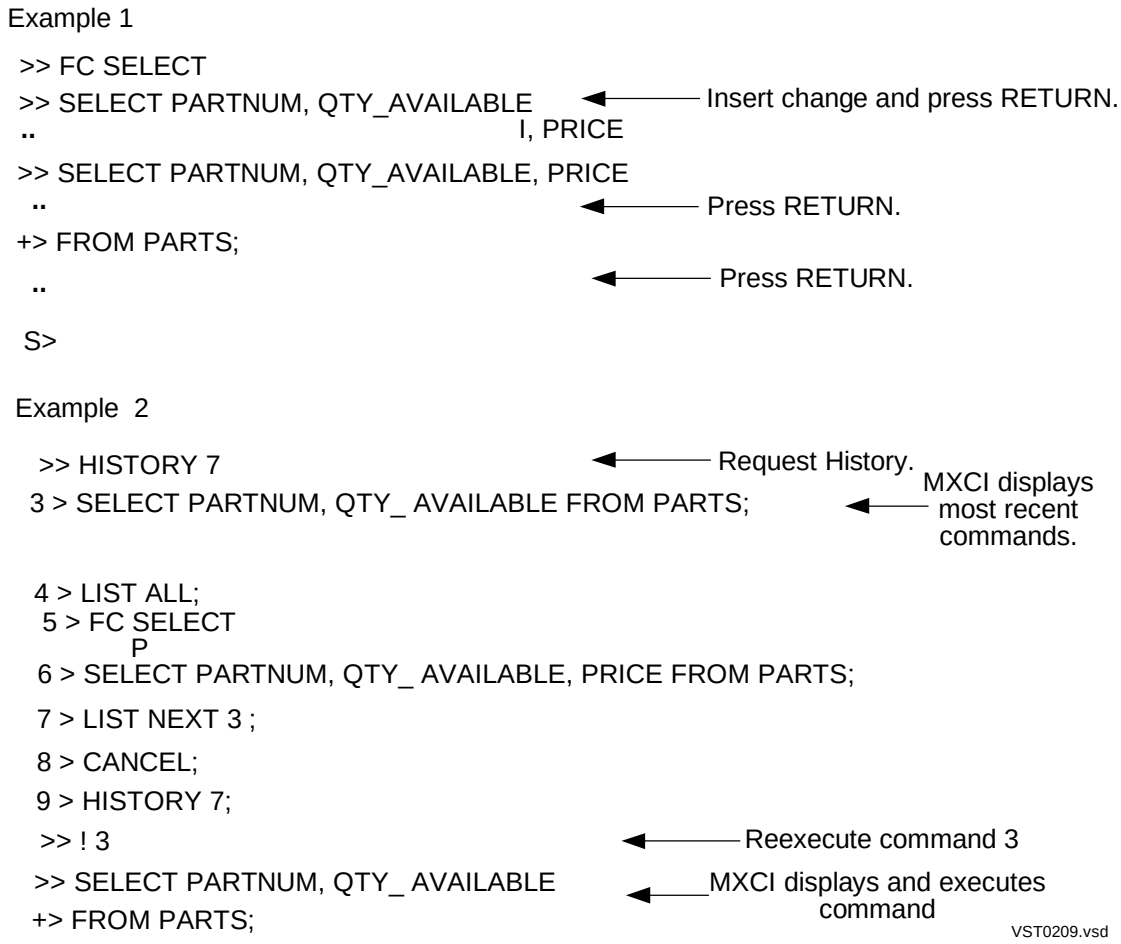
```
>> SELECT PARTNUM, QTY_AVAILABLE
+> FROM PARTS;
```

Figure 2-6 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 2-6 shows how to display the last 7 commands and reexecute command number 3 by using the HISTORY command. You can use the HISTORY command to display up to 25 of the most recently executed commands.

You can enter these commands at the S> prompt also.

Figure 2-6. Repeating Stored Commands



Note. You must use the semicolon terminator for FC and ! commands.

If you are listing rows retrieved by a SELECT command and decide to change the command, use a CANCEL command to return to the MXCI prompt. Then, enter the FC or ! command.

Defining Reports in Command Files

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

- 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.

Creating a Command File With a Text Editor

You can use a text editor to create a command file that contains all the commands needed to define a report. Note these considerations:

- The first command must set mode to REPORT.
- The SELECT command must precede the report formatting commands.

For more information about providing data selection criteria when you execute a command file, see Using Parameters With SELECT Commands on page 3-24.

Creating a Command File From a Log File

1. When you start logging, specify that you want only commands (not output) written to the log file.

   To specify that only commands are to be logged to a file named REPDEFN, enter:

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

2. Set REPORT mode.
3. Enter a SELECT command.
4. Enter report formatting commands.

To stop logging commands, enter:

```
>> 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 necessary changes before using the file.

3 Selecting Data for a Report

Data Selection	3-1
Developing a Query	3-2
Joining Tables	3-4
Comparing a Set of Columns to a Set of Values	3-10
Comparing Character Values	3-11
Grouping Data for Calculations	3-16
Using Expressions to Calculate Report Values	3-22
Preparing a SELECT Command	3-25
Using Views	3-26
Using Subqueries	3-28
Developing Multistep Queries	3-32

Before you can define a report, you must retrieve 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.

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/MX Reference Manual*. For a complete description of developing queries, see the *SQL/MX Query Guide*.

Data Selection

This section discusses 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

Developing a Query

Before you develop a query to select data, consider these 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, 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.

Locating the Data

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.

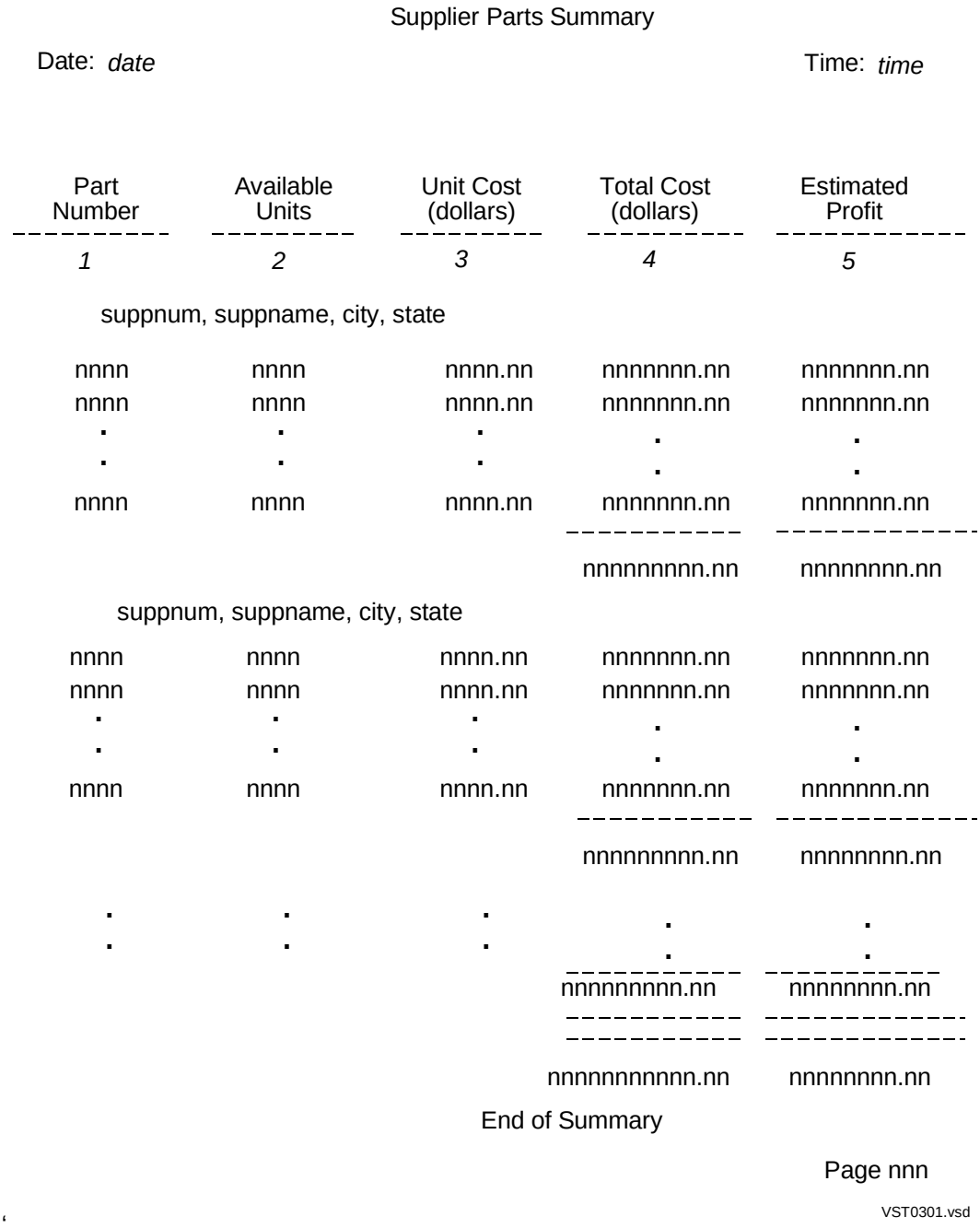
First, you must determine which database tables contain the data. Figure 3-1 uses data from the SUPPLIER, PARTSUPP, and PARTS tables. The report contains this data in the numbered output line items:

```
1 SAMDBCAT.SALES.PARTS.PARTNUM
2 SAMDBCAT.SALES.PARTS.QTY_AVAILABLE
3 SAMDBCAT.INVENT.PARTSUPP.PARTCOST
4 SAMDBCAT.INVENT.PARTSUPP.PARTCOST * SAMDBCAT.SALES.PARTS.QTY_AVAILABLE
5 SAMDBCAT.SALES.PARTS.QTY_AVAILABLE * (SAMDBCAT.SALES.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



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, to retrieve values from the PARTS table, enter:

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

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 catalog and schema.

Joining Tables

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

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

Note that the tables are in different schemas, so the table names must be qualified with the catalog and schema name.

Using the WHERE Clause

The WHERE clause specifies how the rows are to be joined. In Figure 3-2, 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, you should 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 that 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, to define P and PS as explicit correlation names, enter:

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

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

To join the SUPPLIER table to the other two tables, join the rows of the result table shown in Figure 3-2 to rows of the SUPPLIER table that have the same SUPPNUM value:

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

If the SUPPLIER table is also on the default schema, you need not qualify the name. Otherwise, qualify the name with the correct catalog and schema, as shown.

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 column names. The select list for the supplier parts summary consists of the column names in boldface type:

```
>> SELECT S.SUPPNUM,
+>        SUPPNAME,
+>        CITY,
+>        STATE,
+>        P.PARTNUM,
+>        QTY_AVAILABLE,
+>        PARTCOST,
+>        PRICE
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
+>      SAMDBCAT.INVENT.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:

```
>> 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.

To select only rows describing parts with at least 50 units available, enter:

```
>> SELECT *
+> FROM 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.

```
>> SELECT S.SUPPNUM, SUPPNAME, P.PARTNUM,
+> QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
+>      SAMDBCAT.INVENT.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. (See Specifying More Than One Condition on page 3-14.)

Comparison predicates and quantified predicates can include these 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

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.)
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.)

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

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

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

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

- Use quantified predicates to select 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:

  ```
  +> 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, or 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/MX Reference Manual*.

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

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

- Use the IN predicate to locate a specific list of values or all values except those in a specific list. Use this predicate 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, which enables 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 SAMDBCAT.SALES.PARTS
+> WHERE PARTNUM IN (SELECT PARTNUM
+>                   FROM SAMDBCAT.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 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.

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

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

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

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

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

To simplify this query, enter:

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

Note. To compare character data that uses collations, see the COMPARISON entry in the *SQL/MX Reference Manual*

Comparing Character Values

If a column contains text (character data), you must enclose the comparison value in single quotation marks.The characters you enter must match exactly as the characters are stored in the column.

For example, to find a value that consists of uppercase letters, enter uppercase letters. To select without regard to whether a character value is in lowercase or uppercase, use the UPSHIFT function.

Note. The UPSHIFT function upshifts single-byte characters, such as those in ISO88591 and UCS2 character sets. You cannot use the UPSHIFT function if the column contains a double-byte character set.

You must also include the same number of spaces within the comparison string as there are spaces stored within the column value.

This predicate does not select ’PC Diamond 60 MB’ or ’PC DIAMOND, 60MB’:

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

This 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';
```

Use the LIKE predicate to search for similar values by specifying only a few characters and using these 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, report writer 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 report writer compares character values in comparison and LIKE predicates based on these rules. (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 comparison value or the column value (whichever is smaller) are padded to make the values the same length.
- The column value for LIKE predicates is not padded 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 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
			LIKE ‘%MB’	Succeeds, last 2 characters are MB
	‘120^MB’	‘120^MB’		
Variable-length	‘5MB’	‘5MB’	LIKE ‘%MB’	Succeeds
`CHAR VARYING (6)`				
`VARCHAR (6)`			= ‘5MB’	Succeeds
`NATIONAL CHAR VARYING (3)`			LIKE ‘5MB’	Succeeds
`NCHAR VARYING (3)`			LIKE “5MB%”	Succeeds
			LIKE ‘_MB’	Succeeds
			LIKE ‘_MB%’	Succeeds
			LIKE ‘%MB’	
	‘120^MB’	‘120^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.

Because trailing blanks are significant for the LIKE predicate, it might be more restrictive than a comparison predicate when comparing columns of fixed-length character data types.

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 ’ is not located by LIKE ’%MB’. Use comparison predicates to search for exact values, and try including a percent sign (%) 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:

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

The order of evaluation is:

1. Expressions within parentheses
2. NOT
3. AND
4. OR

For example, 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 this revision of these conditions:

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

For example, 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, enclose the predicates in parentheses:

```
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.
- You cannot use conditional expressions that require subqueries (EXISTS, ALL, SOME). Parentheses are not allowed in conditional statements. Use nested IF-THEN-ELSE statements for these types of queries.

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. 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 the specified columns should contain data from the same character set.

Figure 3-4. 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 next 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 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 this query to confirm which rows were grouped for parts 212 and 244 in the previous query:

```
>> SELECT PARTNUM, QTY_ORDERED
+> FROM 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
```

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.

When using a GROUP BY clause or aggregate function, consider these points:

- 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.
- 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.
- 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 ODETAIL O, 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 increases 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 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 that 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 CUSTOMER C, 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:

```
>> MODE REPORT;
>> SELECT C.CUSTNUM, CUSTNAME
+> FROM CUSTOMER C, 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.
```

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

```
>> MODE REPORT;
>> 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, this query determines the minimum and maximum unit price charged for each part in the current set of orders:

```
>> 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 functions you apply to the group. For example:

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

```
>> SELECT DEPTNUM, COUNT(*)
+> FROM 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 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 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. 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 4-15.

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 4-15.

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.

This 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 ODETAIL;
PARTNUM
-------
212
244
255
2001
.
.
```

You can expand this query to also print part descriptions:

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

These 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. To count the number of distinct suppliers of each part, use this query:

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

- This 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 ORDERS OX
+> GROUP BY SALESREP
+> HAVING EXISTS (SELECT SALESREP
+> FROM 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

```
 >> SELECT P.PARTNUM,
 +>            PARTCOST,
 +>            PARTCOST * QTY_AVAILABLE,
 +>            QTY_AVAILABLE * (PRICE - PARTCOST)
+>               FROM SAMDBCAT.SALES.PARTS P,
+>               SAMDBCAT.INVENT.PARTSUPP PS,
+>                SAMDBCAT.INVENT.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
```

VST0305.vsd

An expression in the select list can refer to column names, literals, parameter names, and aggregate functions. These functions are described under Grouping Data for Calculations on page 3-16.

You can use these arithmetic operators:

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

The order of evaluation of an expression is:

1. Expressions within parentheses
2. Unary operators, plus (+) and minus (-)
3. Exponentiation
4. Multiplication and division
5. Addition and subtraction

For a complete description of expressions and syntax rules, see the *SQL/MX 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 report in Figure 3-5 by including expressions in the DETAIL command instead of the select list.

Figure 3-6. Expressions in the Detail Line

```
>> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST, PRICE
        +> FROM SAMDBCAT.SALES.PARTS P,
        +> SAMDBCAT.INVENT.PARTSUPP PS,
        +> SAMDBCAT.INVENT.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

Consider these 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
  - PAGE_NUMBER
- You can use SQL/MX date-time functions in the select list only.

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/MX Reference Manual*. The form of a named parameter is:

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

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

```
SELECT * FROM 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 before executing the command file. For example:

```
>> 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.

Suppose that a SELECT command in a command file named SALESUM consists of these commands:

```
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;
```

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/MX 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. Use this approach to produce several reports with the same SELECT command but based on data selected by different criteria.

With preparation you also can use the EXPLAIN command to determine system resources required to execute the query. For descriptions of the PREPARE, EXECUTE, and EXPLAIN commands, see the *SQL/MX Reference Manual*.

For example, suppose that 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 their customers. You can create a command file that contains a command to prepare the SELECT command.

This SELPREP file contains this PREPARE command:

```
PREPARE SELCUSTINFO FROM
'    SELECT * FROM CUSTOMER C, '
'                  ORDERS R,   '
'                  ODETAIL OD, '
'                  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.

```
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:

```
>> OBEY SELPREP;

( PREPARE command is displayed here.)

.

.

   --- SQL command prepared.
```

To print a report, use the LOG command to log your session and then print the log file through OSS. For more information, see the *Open System Services User's Guide.*

Using Views

A view is a logical table derived by projecting a subset of columns or selecting a subset of 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.

```
>> 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 CUSTOMER C,
+>           ORDERS O,
+>           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:

```
>> SELECT * FROM CUSTORD ;
```

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

These commands define one version of an invoice and are stored in a file named INVOICE. For explanations and examples of 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:

```
>> 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. An Invoice

INVOICE

Customer : DATASPEED
300 SAN GABRIEL WAY
NEW YORK, NEW YORK 10014

Order Date : 870410
Deliv. Date : 870410
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 a 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 invoice in Figure 3-7. You can select information about orders with delivery dates prior to June 1, 2000 by entering:

```
>> 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/MX Reference Manual*.

Using Subqueries

A subquery is a special form of the SELECT command that selects only for purposes of comparison. You specify subqueries in the predicates of a search condition in the WHERE clause or the HAVING clause of a SELECT command. Subqueries cannot be used inside conditional expressions.

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.

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

```
>> SELECT SUPPNUM, PARTNUM, PARTCOST
+> FROM IPARTSUPP
+> WHERE PARTNUM = 2003
+> AND PARTCOST > (SELECT MIN(PARTCOST)
+> FROM 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.

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

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

```
>> SELECT SUPPNUM, PARTNUM, PARTCOST
+> FROM PARTSUPP XP
+> WHERE PARTCOST > (SELECT AVG(PARTCOST)
+> FROM 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 to perform this operation. You could use the implicit correlation name in the outer query.

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

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

***WARNING from MXCI[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. You can enter the percent as a parameter each time you execute the query, which is in a command file named ORDERPCT.

```
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 are calculated when any break column value changes. In this example, the values in all three break columns change at the same time. Therefore, printing the subtotals for each break column is redundant.

Figure 3-8. A 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

When using subqueries, consider the following points :

- 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 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, but subqueries within the WHERE clause of another subquery are at a different level.

Developing Multistep Queries

The techniques discussed next use interim 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/MX Reference Manual*.

Multilevel Group Aggregates

Grouping Data for Calculations on page 3-16 describes ways to apply aggregate functions to groups of rows. To apply aggregate functions to multiple levels of groups, you must specify more than one query and use temporary tables.

For example, to report the average salary for each department and within each department for each job classification:

1. Create an interim 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 )
+> ;
```

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

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

3. Create another interim 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 )
+> ;
```

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 EMPLOYEE
+> GROUP BY DEPTNUM, JOBCODE);
```

5. Join the interim 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, suppose 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.

To create the report:

1. Create an interim table for the report values. Define the RANGE n columns with the system default so that 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 )
+> ;
```

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

```
>> INSERT INTO DEPTTEMP (DEPTNUM, RANGE1)
+> (SELECT DEPTNUM, COUNT(*) FROM 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:

```
>> 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, so 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 next report displays the percent that an individual's salary is of all salaries in a department. To produce the report:

1. Create an interim table to contain the average salary for each department:

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

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

```
>> INSERT INTO AVGTEMP
+> (SELECT DEPTNUM, AVG(SALARY) FROM 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 EMPLOYEE E, 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 or keep the interim table.

4 Customizing a Report

Setting Margins	4-2
Paginating	4-6
Spacing Items and Lines	4-10
Specifying the Items in a Detail Line	4-12
Naming Select List and Detail Line Items	4-14
Organizing Rows Into Break Groups	4-15
Specifying Column Headings	4-17
Specifying Titles	4-20
Specifying Footings	4-24
Formatting Data Values	4-30
Conditional Printing of Items or Line Entries	4-46
Redefining Special Characters	4-48
Calculating Totals	4-49
Printing Double-Byte Characters	4-58

You can use report formatting commands and layout and style options to enhance a report. Examples in this section show you how to produce the special effects you might want in a report. These 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; and 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 settings of the layout options LEFT_MARGIN and RIGHT_MARGIN determine the margins of a report.

If you want a margin to precede the leftmost printed item in your report, set the left margin to the number of blanks needed. 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. A print position is defined in this guide 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 standard output is the terminal. The default right margin is 80 for most terminals. To display the current right margin, enter:

```
>> 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:

```
>> 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 MXCI 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. See Defining Options for Line Folding on page 2-7 for details.

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 in a Displayed Report

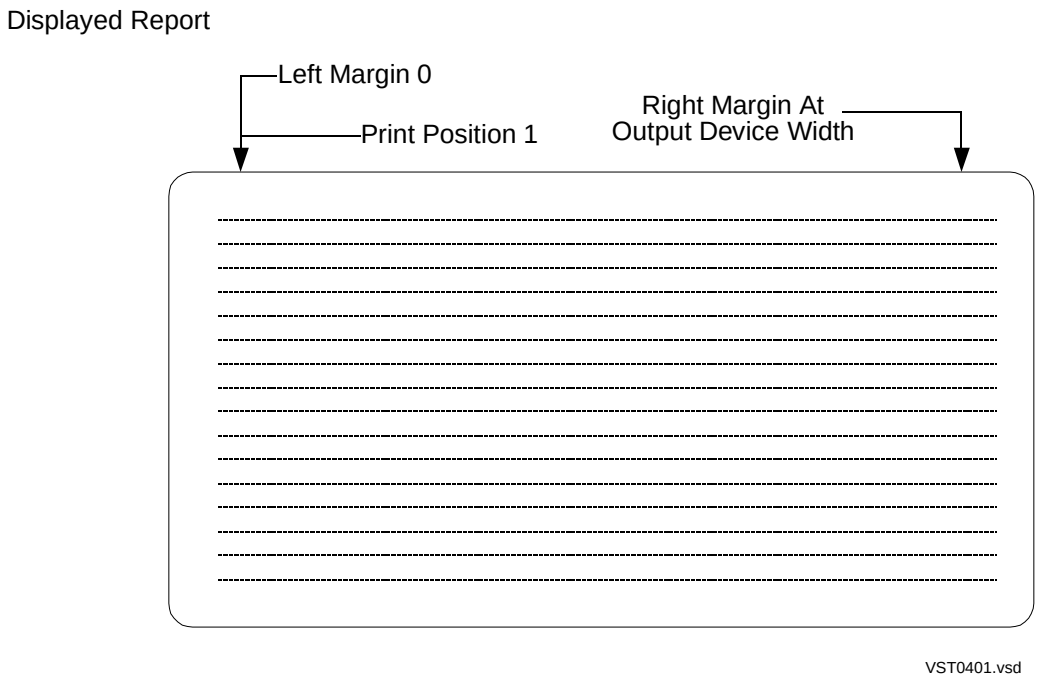
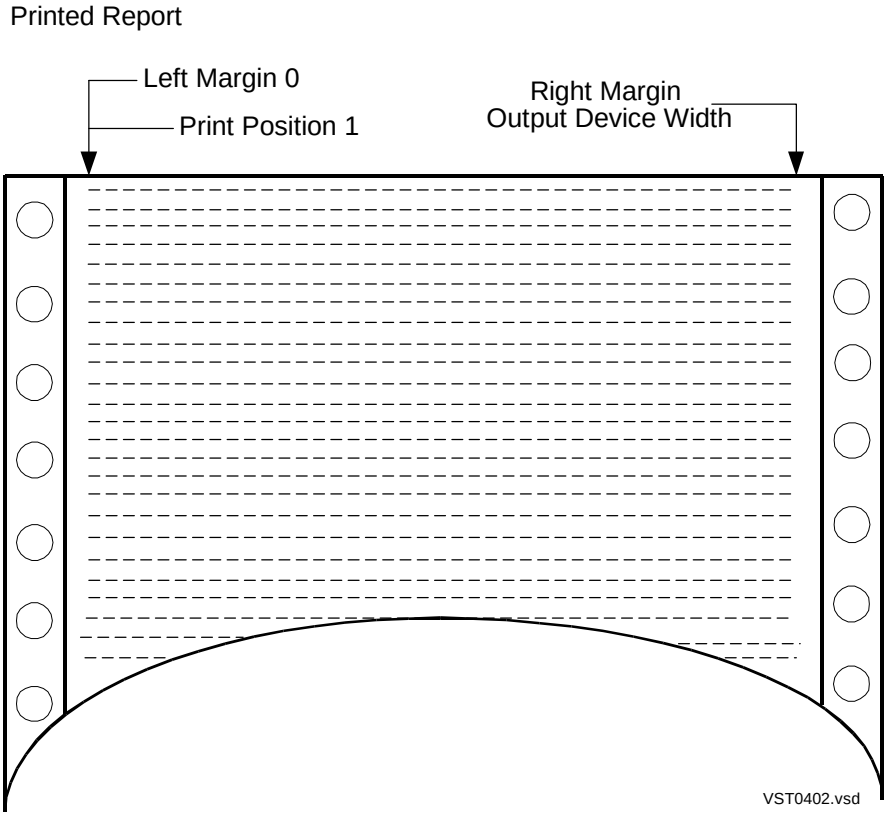


Figure 4-2. Default Margin Settings in a Printer Report



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, to produce a margin size of 4 blank lines preceding the headings (SKIP 4 - 3 = SKIP 1), enter:

```
S> PAGE TITLE SKIP 1;

   ( 4 blank lines appear here. )

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

The four blank lines are created as follows:

- 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, to produce a desired margin size of 4 lines (SKIP 4 - 1 = SKIP 3), enter:

```
S> PAGE TITLE SKIP 3, 'Summary of Employees'; (

 4 blank lines appear here. )

Summary of Employees

CUSTNUM CUSTNAME

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

In this case, the four blank lines are created as follows:

- 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.

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 Specifying Titles on page 4-20 and Specifying Footings on page 4-24.

Paginating

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

PAGE_LENGTH — Specifies the number of lines from the top to the bottom of a page. For example, to set a page length of 55 lines, enter:

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

Default page length for a printed report = 60 lines.

Default page length on a terminal 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 — Specifies a page break in detail lines, titles, and footings.

Use the clause:

PAGE *number*

number specifies the number on the next page and starts a new page numbering sequence.

If you omit number, the current sequence continues.

For example, to specify a page break after each break footing is printed, enter:

```
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 — Specifies the number of subsequent lines that must fit on the current page in detail lines, titles, and footings.

Use the clause:

NEED *number*

number specifies the number of lines required.

If the lines do not fit, a page break occurs. For example, to specify that the break title must be printed on the next page unless the next 4 lines of output fit on the current page, enter:

```
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, to set the limit to 100 pages, enter:

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

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

To number the pages in a report, 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, including those 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 page breaks occur.

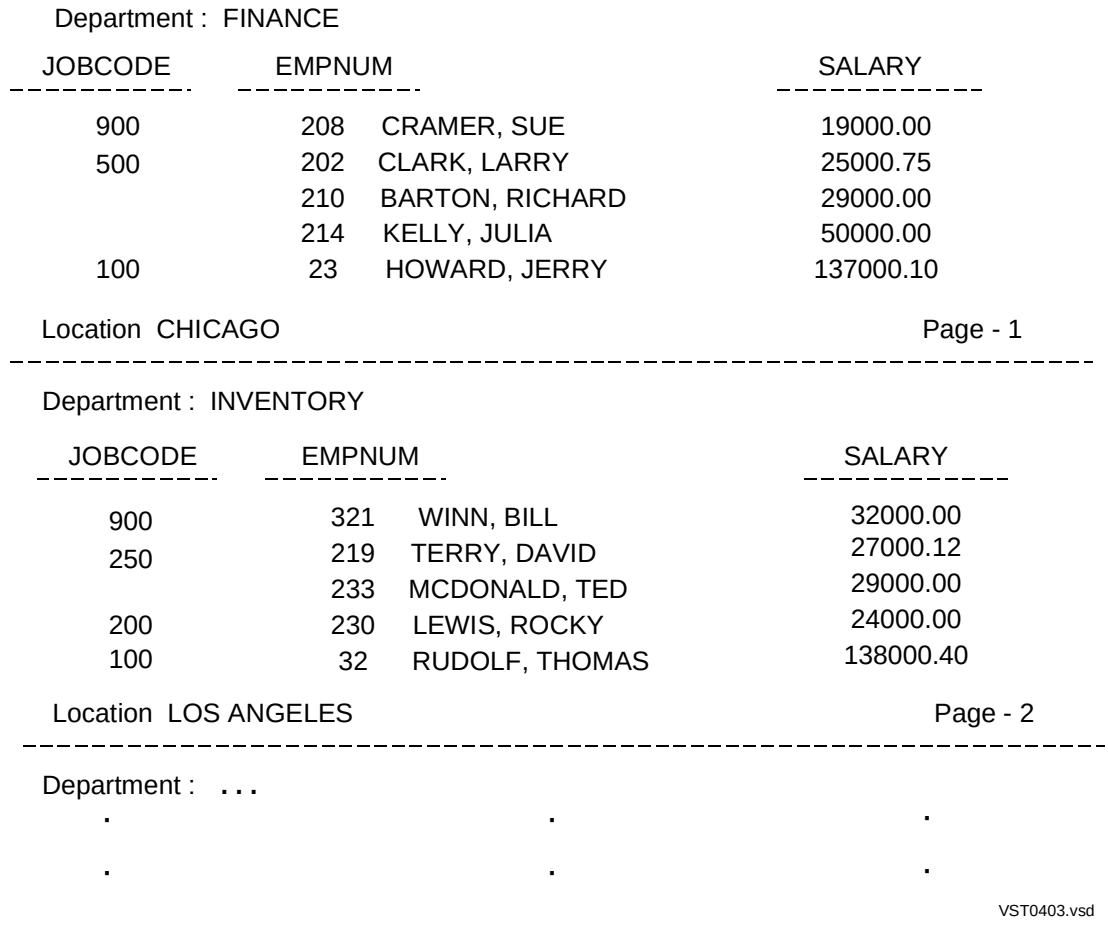
Tho select data for the report, enter:

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

These 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. Pagination Features



Use the NEED clause is 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 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 indicates the 14 lines on each page. These numbers do not appear in the report.

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 >
```

VST0404.vsd

Spacing Items and Lines

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

LINE_SPACING Sets the number of lines to advance between report lines.

Default setting = 1 (a single-spaced report)

For double spacing, enter:

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

SKIP clause Specifies the number of additional times to invoke LINE_SPACING to create extra space before displaying or printing the next item in a detail line, title, or footing.

Use the clause:

SKIP *number*

number is a multiplier to be used with the current value set for LINE_SPACING. The default value is 1.

For example, if LINE_SPACING is set to 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, STREET,
+> SKIP 1, CITY, STATE, POSTCODE;
```

If you set LINE_SPACING to 2 before printing the report, SKIP causes a blank line to appear between each output line. SKIP 3, however, advances 6 lines, so 5 blank lines appear between each output line.

SPACE option Specifies the default number of single-byte spaces between print items of a detail line:

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

Default setting = 2 (at the beginning of an MXCI session)

The number you specify stays in effect until you end an MXCI session or change the number.

The report writer does not insert spaces preceding or following a string literal unless you specify a heading for the string.

SPACE clause

Specifies a number of spaces to insert before displaying or printing the next item in a detail line, title, or footing.

NOTE: This clause temporarily overrides the SPACE option; the clause determines the amount of space between the print items that precede and follow the clause.

- Detail line example:

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

  Output from the detail line has 2 spaces between all items except the EMPNUM and LAST_NAME items.

- Title/Footing example:

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

  This command generates a title with 6 spaces between DEPTNUM and the LOCATION label.

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

Specifies a print position in a detail line, title, or footing.

The position specified:

- Is not relative to the current left margin. It is an absolute position relative to the available print positions on the output device.
- Must be within the margins.

For example, suppose that the left margin is 8 and the right margin is 80. To print 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 this title command as it appears on a terminal and a printed page.

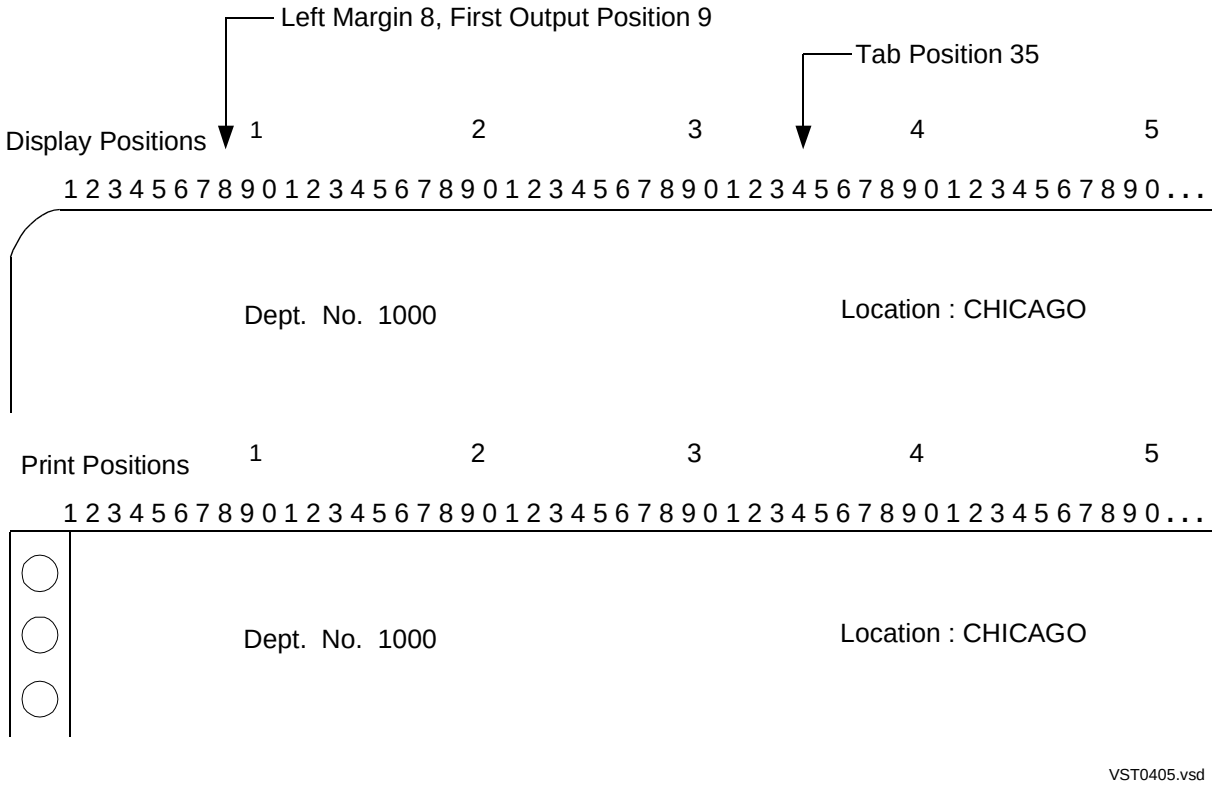
If you change the left margin of a report, you must also change tab specifications to ensure that the alignment of information does not change relative to other information in the report.

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 4-30.

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:

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

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

```
>> SELECT EMPNUM, LAST_NAME, FIRST_NAME, JOBCODE, SALARY
+> FROM 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 col2, col3;
```

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.

  ```
  >> SELECT EMPNUM, LAST_NAME, FIRST_NAME, JOBCODE, SALARY
  +> FROM 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 information.

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:

```
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 Specifying Column Headings on page 4-17.

Naming Select List and Detail Line Items

To assign an alias name to items in the select list, use 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 this 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:

```
>> SELECT P.PARTNUM, PARTCOST,
+> PARTCOST * QTY_AVAILABLE,
+> QTY_AVAILABLE * (PRICE - PARTCOST)
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
+>      SAMDBCAT.INVENT.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 this DETAIL command defines the detail alias name PROFIT:

```
>> SELECT P.PARTNUM, QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
+>      SAMDBCAT.INVENT.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;
```

To refer to the PROFIT column, enter:

```
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:

- For a group formed by a GROUP BY clause:
  - The result is 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 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 3-16.
- For a break group:
  - The 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. This example illustrates a report defined with three levels of break groups. The commands that define the report are:

```
>> 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;
```

The break groups for this report are:

- 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. 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:
  - SUPPRESS ensures 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.
  - NOSUPPRESS means that 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.

Specifying Column Headings

- You can choose default column headings or customized column headings in a report.
- For the default report format, the headings are:
  - For a print item that is a column of a table or view: the heading that was specified when the table or view was created or altered.
  - If no heading was specified, 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 the reports you produce until you set HEADINGS ON, or you end your MXCI session.
- You can use a DETAIL command to specify your own headings or 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.
   - HEADINGS ON (the default) enables headings for the report.
   - HEADINGS OFF suppresses headings for the report.

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.

This example shows how to 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 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>
```

To specify a heading for the concatenated names of employees, enter:

```
>> SELECT * FROM EMPLOYEE E, 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:

  ```
  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 4-30.
- A centered heading might not appear to be centered if the left-justified values are followed by blanks.

Multiple-Line Headings

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 (/).

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

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

To define the detail line and display the first row of output, enter:

```
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>
```

To specify a different new-line character, enter:

```
>> SET STYLE NEWLINE_CHAR '!';
```

The first line of the DETAIL line must then be changed:

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

Specifying Titles

A report can contain three types of titles:

- A page title appears at the top of each page of a report. Only one PAGE TITLE can be in effect at a time.
- A report title follows the page title on the first page of the report. Only one REPORT TITLE can be in effect at a time.
- A break title precedes the break group for which it is defined. 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.

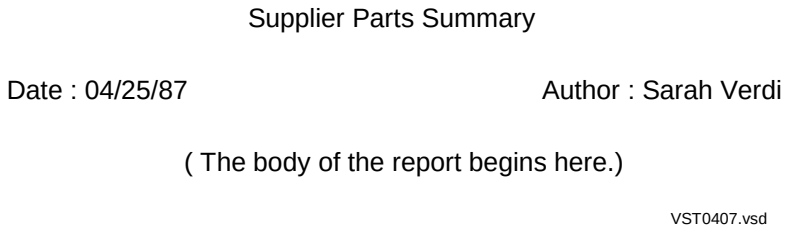
Page and Report Titles

To produce a page and report title for the supplier parts summary, enter:

```
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. Report and Page Titles



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

A title can include:

- Columns from the select list
- Named print items from the DETAIL print list
- String literals
- Arithmetic expressions

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.

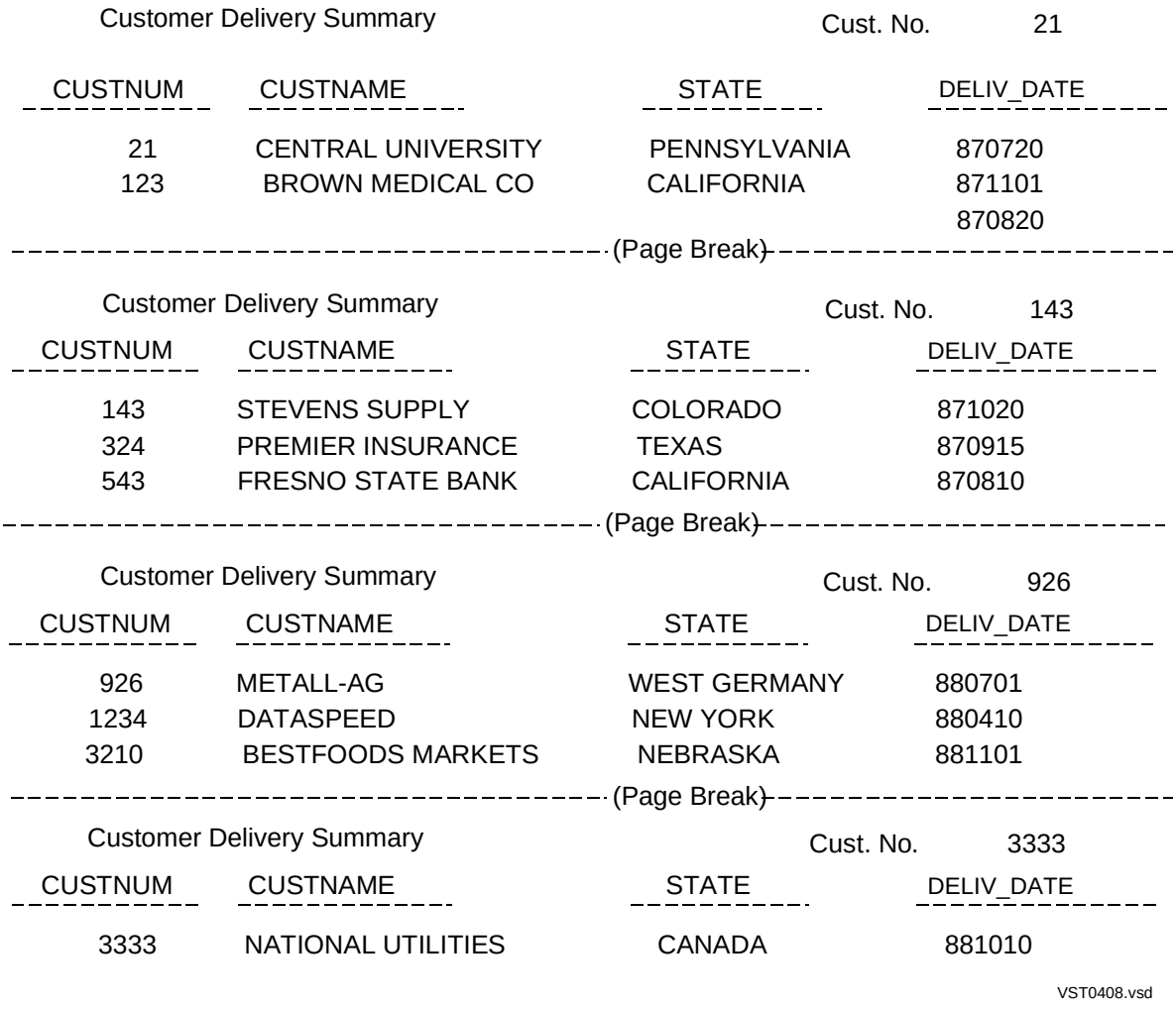
The value used in the title is taken from the first output row on the page (for page titles) and the first output row of the report (for report titles). For example, to specify that the page title include a customer number, enter:

```
>> SELECT *
+> FROM CUSTOMER C,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 NEXT 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. A Page Title With a Row Value



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
- Arithmetic expressions

If you use a column identifier 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.

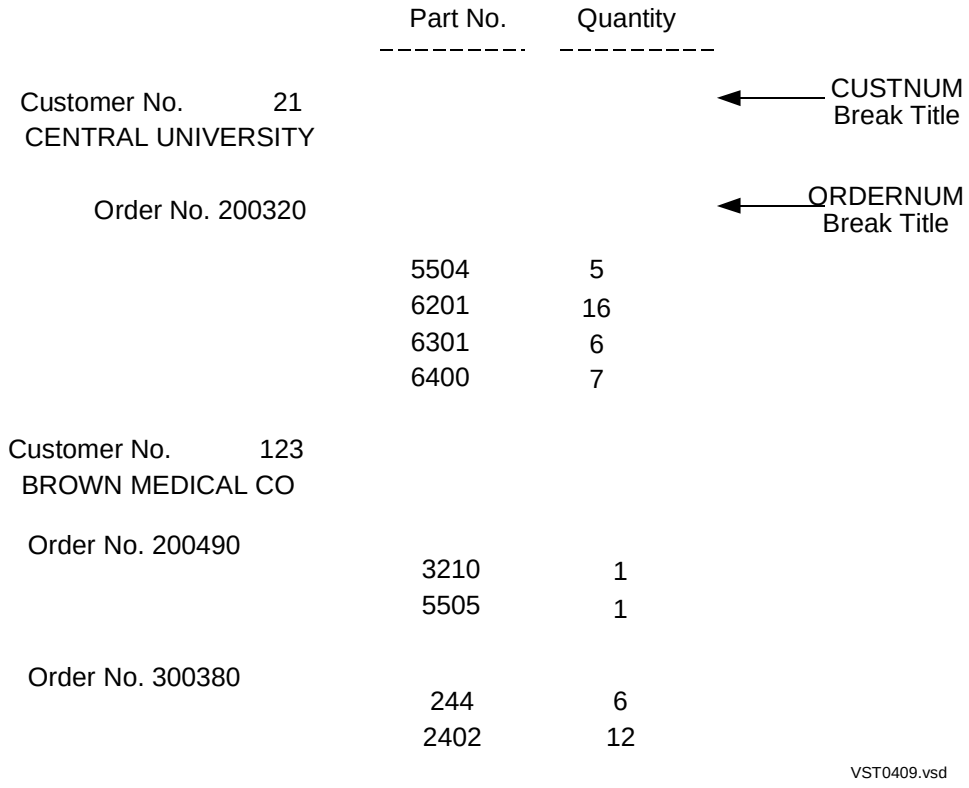
To include an unnamed item from the detail line in a break title, you must define a detail alias name for the item.

In this 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.

```
>> 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 NEXT 8;
```

Figure 4-9 shows the first eight rows of the report.

Figure 4-9. Break Titles



Specifying 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.

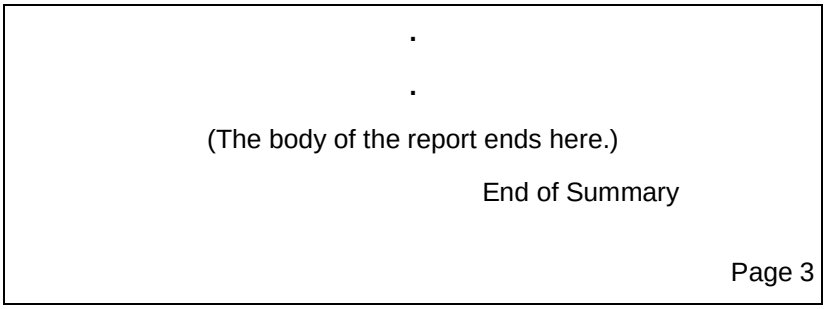
Page and Report Footings

To produce a page and report footing, enter:

```
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



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.

A footing can include:

- Columns from the select list
- Named print items from the DETAIL print list
- String literals
- Arithmetic expressions

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.

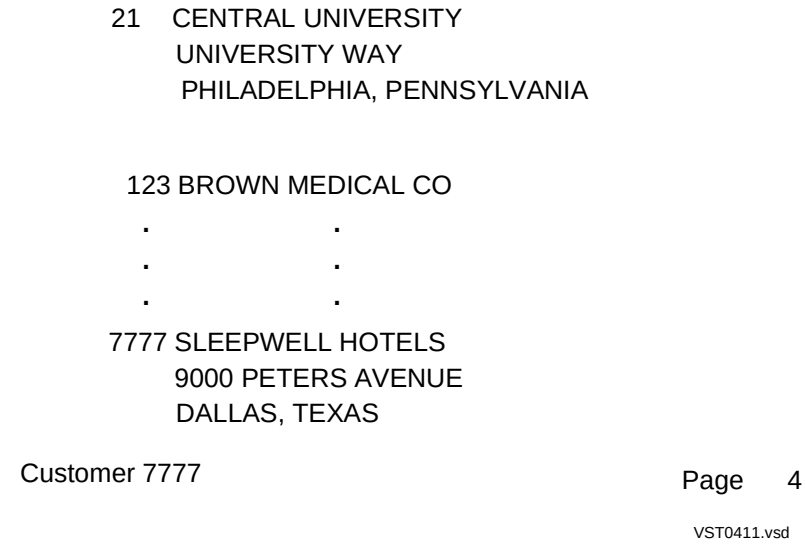
Only one REPORT FOOTING and one PAGE FOOTING command are in effect at a time. You can edit a command by using FC, replace a command by reentering it, or delete a command by using RESET REPORT.

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, to specify that the page footing includes the customer number, enter:

```
>> SET STYLE HEADINGS OFF;
>> SELECT *
+> FROM 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. A Page Footing



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.

A break footing can include:

- Columns from the select list
- Named print items from the DETAIL print list
- String literals
- Arithmetic expressions

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.

To include an unnamed item from the detail line in a break footing, you must define a detail alias name for the item.

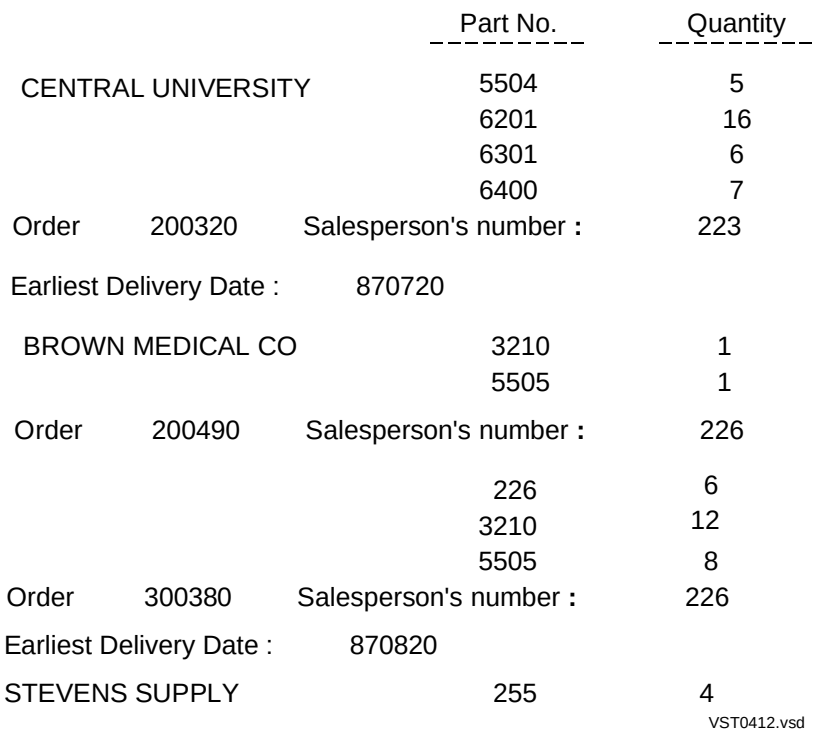
In this example, a report generates break footings for two break groups—one for each customer and one for each order:

```
>> 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. Break Footings



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:

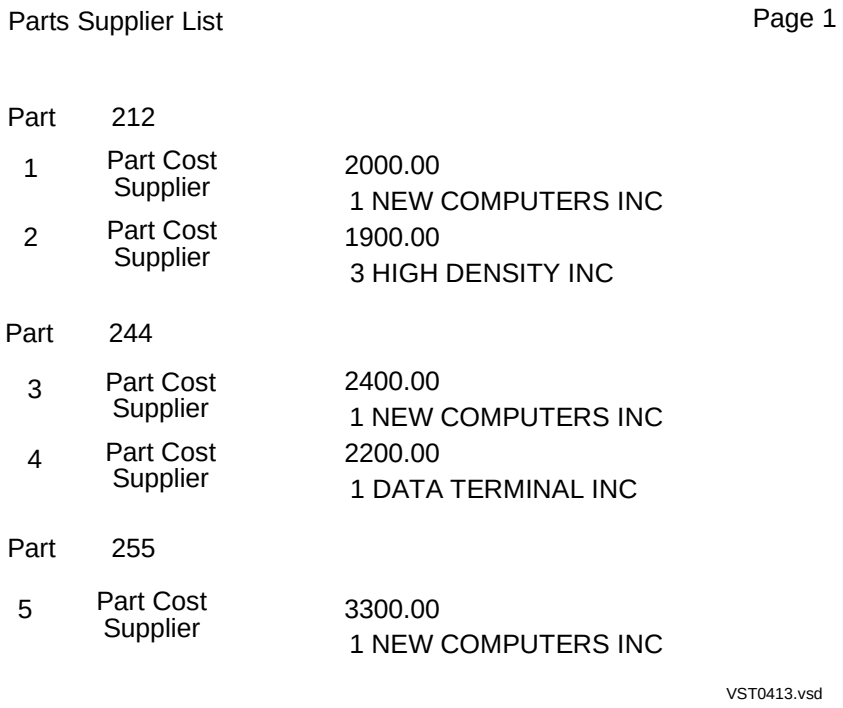
```
LINE_NUMBER OVER REPORT

       or

LINE_NUMBER
```

Suppose that 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. To produce the list, enter:

```
>> SELECT * FROM SUPPLIER, 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;
```

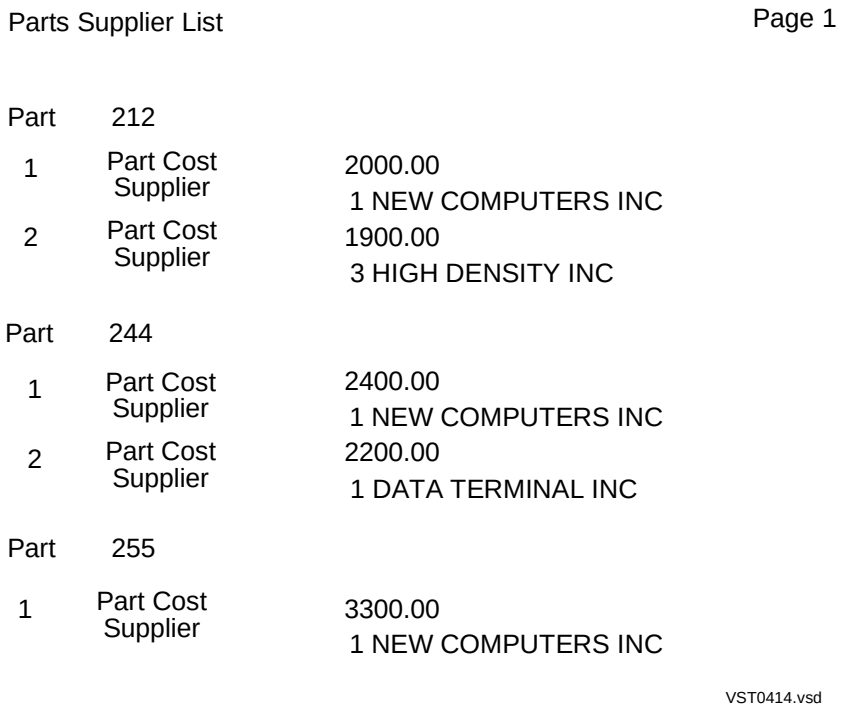
Figure 4-13 shows the report.

Figure 4-13. Numbering Lines



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. Numbering Break Group Lines



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 these 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 Using Decorations in Display Formats on page 4-36 and Filler Characters on page 4-40.

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 listed in Table 4-1.

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 —	I5 (or F6.s if s > 0)
NUMERIC(4,s) UNSIGNED	
NUMERIC(5,s) — NUMERIC(9,s)	I11 (or F12.s if s > 0)
NUMERIC(5,s) UNSIGNED —	I10 (or F11.s if s > 0)
NUMERIC(9,s) UNSIGNED	
NUMERIC(10,s) — NUMERIC(18,s)	I20 (or F21.s if s > 0)
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

Table 4-1. Default Display Formats (page 2 of 2)

SQL Data Type	Default Display Format
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 4-32.
E	Edit descriptor, explained in the *Guardian Programmer's Guide*
DATE	yyyy-mm-dd
DATETIME	yyyy-mm-dd:hh:mm:ss.mssssss
TIME	hh:mm:ss
TIMESTAMP	yyyy-mm-dd:hh:mm:ss.mssssss
INTERVAL	yyyy-mm or dd:hh:mm:ss.mssssss
yyyy	Years
-mm	Months
dd	Days
hh	Hours
:mm	Minutes
ss	Seconds
mssssss	Microseconds

Using the AS Clause to Modify the Default Report Format

If the default display produces a format that does not suit your needs, 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.

You can use the AS clause to override the default format for any data type, with these exceptions:

- DATE
- DATETIME
- INTERVAL
- TIME
- TIMESTAMP

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 are:

  - A[w] specifies alphanumeric display of width w. For example, A20 displays characters in a field that is 20 single-byte print positions wide. Values are left justified in this format.

  Note. If you are defining a report that contains double-bye characters, see Printing Double-Byte Characters on page 4-58 for special considerations for using display descriptors.

  The display descriptors for numeric values are:

  - 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.

- You can include a *scale-sign-descriptor* to specify a scale factor for numeric values.
- You can specify decorations. These character strings are 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 4-35.
- 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

Use the I and F display descriptors for numeric values. In this 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 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>
```

This 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 an interim 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 EMPLOYEE E, 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 NEXT 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 this 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 SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.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 to insert the symbol at the same print position on the output line, regardless of the value of the item.

Using Masks in Display Formats

To apply a mask to insert a dollar sign, enter:

```
>> SELECT JOBCODE, AVG(SALARY)
+> FROM 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 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. 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.
> >

VST0415.vsd

Using Decorations in Display Formats

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 EMPLOYEE
```

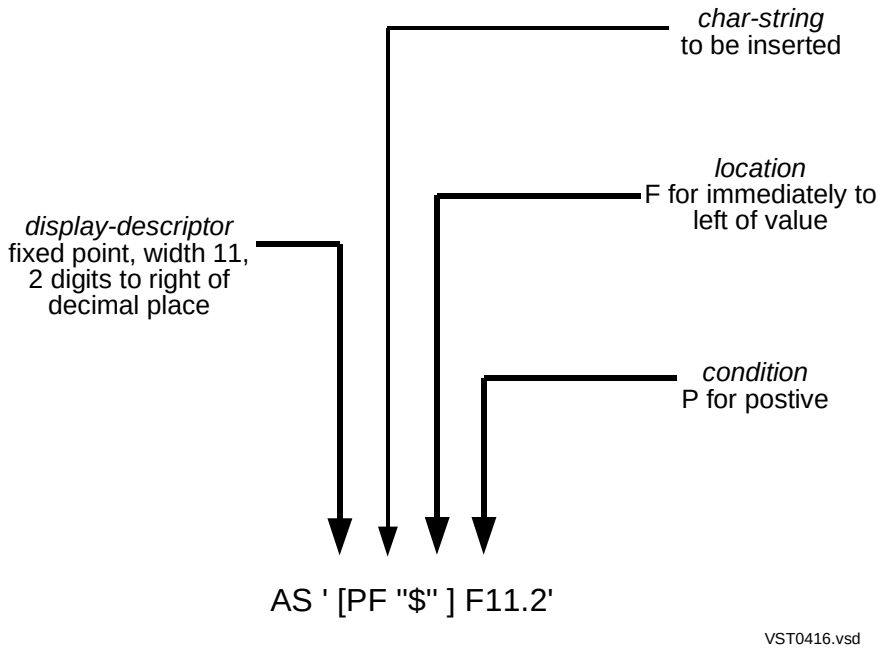
```
+> GROUP BY JOBCODE;
S> DETAIL JOBCODE AS I4 HEADING 'Job',
+> COL 2 AS '[PF''$''] M<Z,ZZZ,ZZZ.99>'
+> HEADING 'Avg. Sal.';
S> LIST NEXT 5;

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. Sample Decoration



The previous decoration specification requires that all values be positive. If you expect negative or zero values, you must expand the decoration:

```
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.

This 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.

Suppressing 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

Use the uppercase letter *Z* to suppress leading and trailing zeros in a mask:

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 single quotation marks.

For information about replacing suppressed leading zeros with a character, see Filler Characters on page 4-40.

You can also suppress trailing zeros by truncating the value.

Concatenating Text

Use the CONCAT clause 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.

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
```

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;
```

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. If a numeric print item is too large for its display format, it is filled 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 these 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 in two ways:

  - Set another single character; for example, a pound sign:

    ```
    >> SET STYLE OVERFLOW_CHAR '#';
    ```

  - Include 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 when:

- A value in an A display 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 4-35.

This 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 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 single 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 DATETIME, DATE, TIME, or TIMESTAMP data types. 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/MX 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. Therefore, if you use CURRENT_TIMESTAMP in the report title and the report footing, the times printed 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 3-24.

- 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 SQL/MX 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 UPDATE command, and a report command print list. You cannot use the CURRENT function in a SET PARAM or EXECUTE command. For descriptions of SQL date-time functions, see the *SQL/MX 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 4-46.

Julian Timestamp Formats

By using the AS DATE/TIME clause, you can specify the date format and time format of a Julian timestamp.

To specify a page title that includes the local date and time, enter:

```
S> PAGE TITLE 'Supplier Parts Summary', TAB 30,
+>     CURRENT_TIMESTAMP AS DATE * TIME *;
S> LIST NEXT 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 MXCI session by setting the DATE_FORMAT and TIME_FORMAT style options as:

```
>> 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, enter:

```
AS TIME * DATE *
```

If you want only the date or only the time printed, you can omit the unwanted option.

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 NEXT 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 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	Hundredth 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.

To display the current default date and time formats, enter:

```
>> SHOW STYLE DATE_FORMAT, TIME_FORMAT;
```

SQL/MX Date and Time Formats

If your report includes a date or time generated or stored as an SQL/MX date-time data type, 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 4-31.
- 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 listed 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.mssssss
USA	mm/dd/yyyy hh:mm:ss.mssssss AM|PM
EUROPEAN	dd.mm.yyyy hh.mm.ss.mssssss

- 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*.

These print items produce identical formats:

```
CURRENT
```

```
DATEFORMAT (CURRENT, DEFAULT)
```

A current date and time generated by either of these functions would appear as:

```
2000-04-15:18:22:05.61003
```

If you specify the print item:

```
DATEFORMAT (CURRENT, EUROPEAN)
```

the date and time would appear as:

```
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/MX Reference Manual*.

To insert characters or spacing within a European date and time, you can include print items in a report formatting command:

```
DATEFORMAT (CURRENT YEAR TO DAY, EUROPEAN), '*****',
DATEFORMAT (CURRENT HOUR TO MINUTE)
```

The result would appear as:

```
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 4-31.

Converting Timestamps

You can use the JULIANTIMESTAMP function to convert a value with an SQL/MX 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 display 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, use:

```
JULIANTIMESTAMP (CURRENT) AS DATE 'DA'
```

For a TIMESTAMP column named XX, use:

```
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, suppose that 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. To print the date with the units separated by slashes, use this mask in the print item:

```
DELIV_DATE AS M<03/03/03>
```

The printed result is 03/07/18.

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 in this example conditionally prints text depending on the employee's salary:

```
>> SELECT EMPNUM, FIRST_NAME, LAST_NAME, DEPTNAME,
+> SALARY
+> FROM EMPLOYEE E, 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 NEXT 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>
```

When specifying an IF/THEN/ELSE clause, consider these points :

- 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 SQL/MX search condition syntax and rules with these exceptions:
  - You cannot include subqueries.
  - Arithmetic expressions cannot include aggregate functions or report functions.

To specify predicates:

```
>> SELECT * FROM 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 NEXT 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 4-35.

For another type of conditional printing, see Filler Characters on page 4-40.

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, enter a SET STYLE command and specify one or more options:

```
>> SET STYLE UNDERLINE_CHAR '=', OVERFLOW_CHAR '+';
```

To display current default characters, enter:

```
>> SHOW STYLE *;
```

To reset an option to its default, use the RESET STYLE command. For example, to reset the default underline character to the underscore (_), enter:

```
>> RESET STYLE UNDERLINE_CHAR;
```

To set all style options to their default values, enter:

```
>> RESET STYLE *;
```

Calculating Totals

You can calculate totals of numeric print-item values. A total appears at the end of the report. For example, in the following report, you want the total of all parts ordered:

```
>> SELECT PARTNUM, SUM (QTY_ORDERED)
+> FROM 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*).

In this example, the columns that contain expressions are assigned detail alias names. These columns are totaled at the end of the report.

```
>> SELECT S.SUPPNUM, SUPPNAME, CITY, STATE, P.PARTNUM,
+> QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
+>      SAMDBCAT.INVENT.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 4-17 shows the resulting report.

Figure 4-17. 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, 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 4-18 shows the resulting report.

Figure 4-18. 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 two-digit values, the precision might be six 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 4-15.

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 these commands:

```
>> SET LAYOUT LEFT_MARGIN 8, RIGHT_MARGIN 72,
+>            PAGE_LENGTH 56;
>> SELECT S.SUPPNUM, SUPPNAME, CITY, STATE, P.PARTNUM,
+>         QTY_AVAILABLE, PARTCOST, PRICE
+> FROM SAMDBCAT.SALES.PARTS P,
+>      SAMDBCAT.INVENT.PARTSUPP PS,
```

```
+>      SAMDBCAT.INVENT.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> LIST ALL;
```

Figure 4-19 and Figure 4-20 show the resulting two-page report.

Figure 4-19. A Report With Subtotals

Supplier Parts Summary

Date: April 20, 1987 Time: 02:06:56 PM

Part Number	Available Units	Unit Cost (dollars)	Total Cost (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 4-20. A Report With Subtotals (Page 2 of 2)

Supplier Parts Summary

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

When using the SUBTOTAL command, consider these points:

- 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.
- To delete all subtotal commands, enter:

  ```
  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 4-21 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, to set the label used in Figure 4-21:

```
>> SET STYLE SUBTOTAL_LABEL '* * *';
```

You can specify from 0 through 255 single-byte characters (or 0 through 177 double-byte characters) in the label. To turn off subtotal labeling, specify zero characters:

```
>> 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.

To define the report in Figure 4-21:

```
>> 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 4-21. 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
.	.	.	.	.	.
.	.	.	.	.	.

VST0422.vsd

Calculating Subtotals on Conditional Values

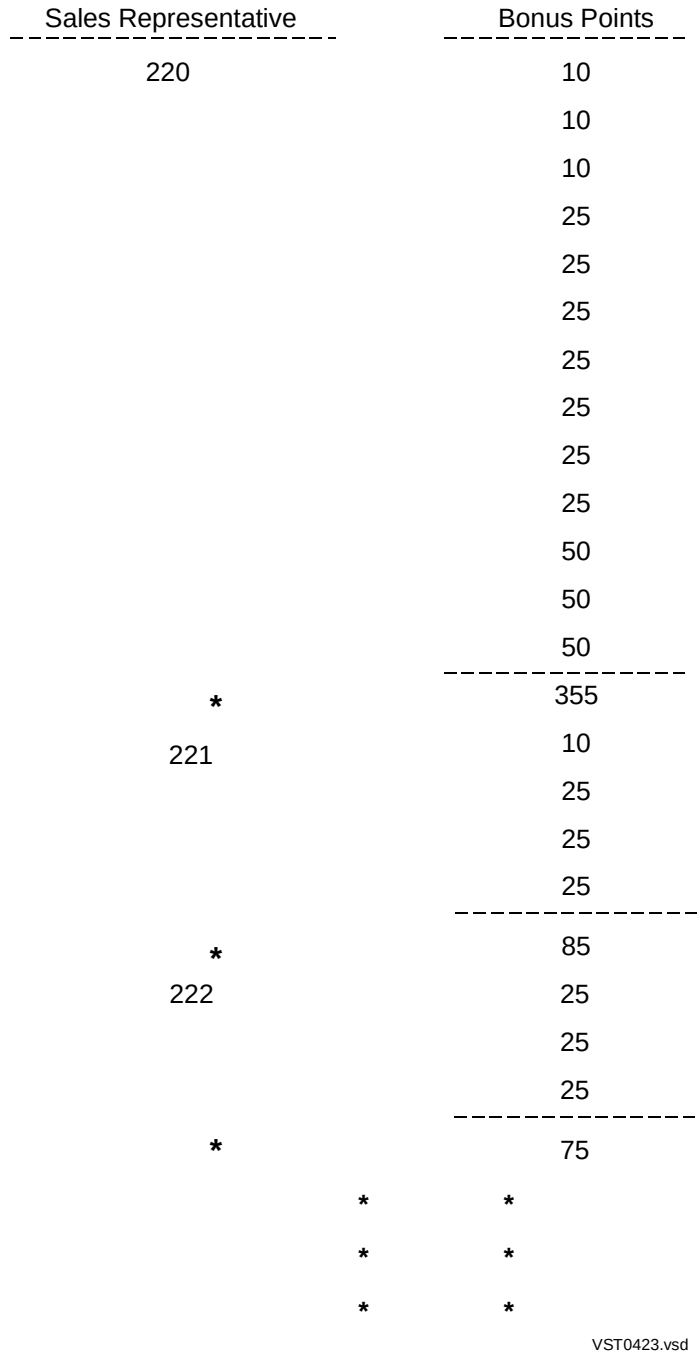
The value in a column specified by using an IF/THEN/ELSE clause is in character format and cannot be subtotaled or totaled by using the SUBTOTAL or TOTAL command. You can use the UNION operator to specify a select list column that contains a conditional value and can be subtotaled.

In this example, a union of three SELECT statements (shown in boldface) 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.

```
>> 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 4-22 shows the first few lines of the resulting report.

Figure 4-22. Subtotals on Conditional Values



Printing Double-Byte Characters

SQL/MX currently supports printing of a single byte characters only. A double-byte character printed in a report is represented by a question mark (?).

A

Migration from SQL/MP Report Writer to SQL/MX Report Writer

This appendix discusses unsupported and changed commands and options, error handling, and the use of single and double quotation marks in SQL/MX report writer.

Unsupported Commands, Options, and Functions

- ADD DEFINE
- OUT_REPORT
- RESET REPORT SELECT
- RESET SESSION
- SET SESSION
- WRAP
- CN[.m] display descriptor for multiline text
- [LCT/GMT] display descriptor for DATE/TIME
- More than one parameter in the same BETWEEN clause
- Conditional expressions in parentheses (You must use nested IF/THEN/ELSE clauses)
- Ampersand (&) to break up a long string literal
- Printing reports (You must print reports as LOG files from OSS, using the *lp* command. For more information, see the *Open System Services User's Guide*.)

Changed Commands and Options

- An AS clause specified on CHAR/VARCHAR datatype in a DETAIL command causes filler characters to be substituted for trailing spaces in the column value.
- LIST, FIRST, NEXT, and ALL cannot be abbreviated. Enter the word.
- LIST NEXT *n* must be explicitly typed. Pressing ENTER at the S> prompt does not perform this command.
- Setting LIST_COUNT does not implicitly enable report writer. You must explicitly set MODE REPORT.
- RESET REPORT commands allow only one option.

- RESET REPORT TOTAL (column-name) allows only one column.
- Subqueries are not allowed within SQL/MX report writer commands.
- For logical folding, SQL/MX report writer does not maintain spaces between columns. The next folded column begins at the first print position of the next line, regardless of the SPACE option setting.
- Display of exponents of an integer data type results in approximate floating point format:

SQL/MP Report Writer	SQL/MX Report Writer
4.000000	4.00000000000000000E+000

- Numeric expressions are evaluated to these data types:

When …	Resulting data type is …	Display length is …
All operands are integers	SQLDT_64BIT_S	21
One or more operands are FLOAT	SQLDT_IEEE_DOUBLE	23

Error Handling

- SQL/MX report writer parses a command until it encounters an error. At that time, an error message is issued, and parsing stops.
- SET LEFT/RIGHT MARGIN ALL issues a syntax error.
- If nonfatal errors are encountered in processing a report (a LIST command has been issued), SQL/MX report writer displays the report first, followed by the errors encountered. Currently, the only such error is error 15521.

Single and Double Quotes

- You must enclose string literals in single quotes.
- You must enclose AS clause modifiers in single quotes.
- You must enclose a special character decoration or filler character (such as $, +, or *) in a quoted string (such as an AS clause modifier) in single quotes, plus an additional set of single quotes.
- You must enclose option values in SET commands in single quotes.
- In SQL/MX report writer, you must use double quotes to:
  - Indicate a case-sensitive column name
  - Indicate a column name that is the same as an SQL/MX keyword

Index

A

A display descriptor 4-33
ADD DEFINE command A-1
Aggregate functions
  applied conditionally 3-33
  groups of rows 3-16
  using 3-17
Alias name 4-14
Alphanumeric character comparison 3-11
Ampersand A-1
AND operator 3-14
Arithmetic expression, evaluation of 3-23
Arithmetic operators 3-23
AS clause
  formatting Julian timestamp 4-43
  in DETAIL command 4-34
  Julian timestamp data type requirement 4-44
  overview 4-32
Averages, computing 3-1
AVG function 3-18

B

BETWEEN
  comparing a set of columns 3-10
  one parameter allowed A-1
  overview 3-8
Blanks
  trailing 3-12
blanks
  padding with 4-47
  trailing 4-41
  trailing, with CHAR data type 4-41
blanks, padding with 4-47
Blanks, trailing 3-13
  using to compare characters 3-13
Boolean operators 3-14
Break
  column subtotaling 1-8
  footing 1-8, 4-24
  group 1-8, 4-15
  title 1-8, 4-22
BREAK FOOTING command 4-26
BREAK ON command 4-15, 4-16
BZ modifier 4-38, 4-39

C

CANCEL command 2-7
Catalog, creating 3-32
CENTER_REPORT 1-16
CHAR data type comparison 3-12
Characters
  comparing character data types 3-12
  comparing character values 3-11
  comparing lowercase 3-11
  comparing uppercase 3-11
  display descriptors for 4-33
  double-byte, printing 4-58
  justification 1-7, 4-32
  maximum in a line 1-16
  redefining special characters 4-48
  VARCHAR data type comparison 3-12
  variable-length 3-13
Clauses
  report formatting commands 1-14
  report formatting, summary of 1-15
CN[.m] display descriptor A-1
Collations, using 3-11
Columns
  data types 3-10
  description of 1-13
  determining grouping of 3-20
  headings 4-17
  identifiers 1-13

Columns (continued)
omitting from output line 4-13
qualified names for 3-2
selecting data from 3-5
Command file
defining report in 2-15
Command file, defining report in 2-15
Commands
canceling formatting 2-12
continuation prompt 2-2
entering 2-10
executing again 2-14
executing previous 2-14
repeating 2-14
report formatting 1-14
stored 2-14
Comparison predicate
comparing character values 3-11
comparing uppercase characters 3-11
example of 3-7
operators 3-7
rules 3-12
Compiling SELECT command 3-24
COMPUTE_TIMESTAMP function 4-42
Computing averages 3-1
CONCAT clause 1-15, 4-30, 4-39
Conditional
aggregates 3-34
expressions, not allowed in parentheses A-1
subtotals, calculating 4-51
Conditions
for display format 4-37
for printing item 4-46
multiple, in a query 3-34
order of evaluation 3-14
selection 3-6
specifying more than one 3-14
Converting timestamps 4-45
CONVERTTIMESTAMP function 4-45
Correlated subqueries 3-29
Correlation name 3-5, 3-30
explicit, example 3-30
explicit, using 3-31
implicit 3-5
implicit, example 3-5
COUNT function 3-19
CREATE CATALOG command 3-32
CREATE VIEW command 3-27
CURRENT function 4-42
CURRENT_TIMESTAMP 4-42

D

Data
formatting 4-30
grouping for calculations 3-16
locating, in database 3-2
organization 1-2
sorting 3-15
source 3-5
Data types
See also individual data types
character, comparing 3-12
formats 4-30
numeric expressions, of A-2
Database
organization 1-2
SQL/MX 1-2
Date
converting 4-45
formatting
European 4-45
overview 4-45
with AS DATE/TIME 4-44
local 4-43
DATEFORMAT function 4-44
Date-time data types 4-42, 4-44
Date-time functions 4-42
DATE_FORMAT style option 4-43
DECIMAL_POINT style option 4-48

Decorations
display of 4-33
form of 4-37
Defaults
catalog, logical name 2-2
data type formats 4-30
date and time formats 4-43
decimal character for numeric print item 4-48
directory 2-1
heading 4-13
margin, bottom 4-4
margin, top 4-4
overflow character 4-33
report format 1-3, 1-6
report format example 1-6
schema,logical name 2-2
spacing 4-10
subtotal label 4-51
Descriptor
See Display descriptors
Detail alias names 4-49
DETAIL command 4-13
Detail line
description of 1-7
expressions in 3-23
items in 4-12
naming items 4-14
Digit display 4-34
Directory
display current setting 2-4
display default setting 2-4
setting default 2-1
Display descriptors
A 4-33
CN[.m] A-1
F 4-33, 4-34
for character values 4-33
I 4-33, 4-35
LCT/GMT display descriptor A-1
Display descriptors (continued)
M (mask) 4-33
overview 4-33
scale-sign descriptor 4-34
template for mask 4-33
using 4-32
Display formats
specifications for 4-32
table of 4-31
Display modifiers 4-32
Distinct rows
in count 3-18
selecting 3-21
Dollar sign insertion 4-36
Double spacing 4-10
Double-byte characters, printing considerations for 4-58

E

Editing
commands 2-12
report 2-16
ELSE clause 4-46
ENV command 2-4
Environment, session 2-2
Errors, in report writer A-2
European decimal character 4-38
Evaluation of expression, arithmetic 3-23
EXECUTE command 3-25
EXISTS predicate 3-9
EXPLAIN command 3-25
Exponentiation 3-23
Exponents, display A-2
Expressions
calculating report values with 3-22
detail line 3-24
evaluation 3-14, 3-23

F

F display descriptor
example 4-34
using 4-34
zero suppression 4-38
FC command 4-55
Field filler characters 4-1, 4-40
Field size 4-19
File
See Command file
Fitting lines on a page 4-6, 4-21
Fixed-point display 4-33
FL modifier 4-40
Folding a line 2-7
Footings
break 1-8, 4-24
page 1-8
Format of data types 4-30
Formatting clauses, summary of 1-14
Formatting commands
canceling 2-12
summary of 1-14
FROM clause 3-4
Functions, summary of report 1-17

G

GROUP BY clause
computing sums or averages with 3-18
select list relation 3-17
Grouping columns
choosing 3-21
select list relation 3-17
Guardian timestamp
See Julian timestamp

H

Headings
multiple-line 4-19
suppressing 4-17

I

I display descriptor
overview 4-33
using 4-35
zero suppression with 4-38
IF/THEN/ELSE clause
subtotaling, substitute method 4-56
using 4-46
IN predicate 3-9
Integer display 4-33
INTERVAL data type, totaling 4-51
Invoice example 3-28
INVOKE command 3-2
Items,naming with an alias 4-14

J

Julian timestamp 4-42
JULIANTIMESTAMP function 4-44
Justification of values 4-32

L

Layout options
CENTER_REPORT 1-16
definition 1-3
LOGICAL_FOLDING 2-7, A-2
PAGE_COUNT 1-5
PAGE_LENGTH 4-6
summary of 1-16
using 4-2
LIKE predicate
comparison operators 3-7
using 3-12
Lines
folding 2-7
heading 4-19
inserting text in 4-30
keeping together on page 4-6
logical 1-13
maximum length 1-16

Lines (continued)
numbering, in a report 4-27
spacing of 4-10
LINE_NUMBER function 4-27
LINE_SPACING, setting 4-10
LIST command
abbreviation not allowed A-1
ALL parameter 4-6
LIST NEXT must be typed A-1
options 2-6
using 2-6
LIST_COUNT A-1
Locating data in database 3-2
Log file
creating 2-13
creating command file from 2-14
displaying current 2-4
printing reports with 2-13
Logical line
description 1-14
size 1-14
Logical tables
definition 3-6
views 3-26
LOGICAL_FOLDING layout option
spaces between columns A-2
using 2-7
Lowercase comparison 3-11

M

M (mask) display descriptor 4-33
Margins
layout option settings for 1-7, 4-2
setting 1-7
wider than device 2-8
Masks
conditional printing of text 4-46
date format 4-46
zero suppression 4-33
MAX function 3-19
MIN function 3-19
MODE command
MODE DISPLAY 2-4
MODE REPORT 2-4, A-1
MODE SQL 2-4
Modifiers, display 4-32
Monetary values 4-1, 4-38
Multistep queries 3-35
MXCI
ending 2-2
standard prompt 2-2, 2-6
starting 2-2
MXCI commands, summary of 1-17

N

NAME clause 4-14
NAME command 4-14
Named parameters 3-25
NATIONAL CHAR data type comparison 3-12
NEED clause
description 4-6
example 4-8
NEWLINE_CHAR style option 4-19, 4-48
NOHEAD clause 4-18
NOT operator 3-14
NULL predicate 3-9
NULL_DISPLAY style option 4-48
Numeric values
display descriptors for 4-33
formatting 4-34
justification 4-32

O

OBEY command file
See Command file
OC modifier 4-40
Offset 4-2

Operators
arithmetic 3-23
Boolean 3-7
comparison 3-7
Options
layout, definition 2-5
setting and displaying 2-5
style, definition 2-5
OR operator 3-7
Order
arithmetic evaluation 3-23
condition evaluation 3-14
select list 3-17
ORDER BY clause
break groups, relation to 4-15
description of 3-17
example 1-6
row grouping 3-17
Outer query 3-30
OUT_REPORT A-1
Overflow character, single item 4-33
OVERFLOW_CHAR style option 4-40

P

Page
breaks 4-6
fitting lines on 4-6, 4-21
footing 1-8, 4-6
length 4-6
length, setting 1-7
numbering 4-6
title 4-20
PAGE clause 4-6
PAGE FOOTING command 4-25
PAGE TITLE command 4-20
PAGE_COUNT 1-5
PAGE_LENGTH layout option 4-6
PAGE_NUMBER function 4-7
Pagination 4-1, 4-6
Parameters
named 3-24
unnamed 3-26
user-defined in SELECT command 3-24
Parentheses in expressions 3-14
Percent of all row values 3-35
PIC X data type comparison 3-12
Predicates 3-7
PREPARE command 3-25
Print items
conditional printing 4-13
description of 1-13
field size 4-19
filler characters 4-1
formatting 4-30, 4-45
print lists 1-14
spaces between 4-10
Printing reports
using a LOG file A-1
with formatting commands 2-13
Prompt
command continuation prompt 2-2
MXCI standard prompt 2-2, 2-6
select-in-progress prompt 2-2
Proper names, formatting 4-14, 4-39

Q

Qualified column name 3-2
Quantified predicates
comparison operators and 3-7
example of 3-30
Queries
See also SELECT command
developing 3-1
multistep 3-32
Quotation marks, rules for using A-2

R

Replacing commands 2-12
REPORT FOOTING command 4-25
Report formatting
  clauses, summary of 1-15
  commands, summary of 1-14
Report functions
  summary of 1-17
  using 3-24
Reports
  customized 1-8
  date and time, current 3-2
  defining 2-15
  development steps 1-4
  footing 1-8, 4-24
  formatted 1-8
  formatted, example 1-8
  functions summary 1-17
  layout and style 1-5
  printing as LOG file A-1
  saving 2-16
  subquery used with 3-30
  title 1-8
RESET command 2-5
RESET LAYOUT command 2-6
RESET REPORT A-1
RESET REPORT SELECT A-1
RESET REPORT TOTAL A-2
RESET SESSION A-1
RESET STYLE command 2-6
Rows
  averaging 3-16
  break groups 4-1, 4-15
  counting 3-18
  description of 1-2
  functions applied to groups of 3-33
  grouping for calculations 3-16
  listing 2-6
  maximum value 3-19
Rows (continued)
  minimum value 3-19
  returning to the first 2-6
  selecting distinct 3-21
  sorting 3-1
  summing 3-16
  value as percent of all rows 3-35

S

Scale factor for display value, specifying 4-34
Scale-sign descriptors 4-33
Search conditions
  Boolean operators for 3-7
  IF/THEN/ELSE clause 4-47
  multiple 3-9
SELECT command
  canceling 2-6, 2-7
  compiling 3-24
  first row, returning to 2-13
  parameters in select list 3-24
  preparing 3-25
Select list
  asterisk (*) 3-32
  DISTINCT keyword 3-21
  expressions in 3-22
  GROUP BY clause relation 3-17
  naming items in 4-14
  omitting from output line 4-13
  order 3-6
  subquery 3-9
Selection criteria 3-1
Select-in-progress prompt 2-2
Session environment 2-1
SET LAYOUT command 1-3
SET PARAM command 3-25
SET SESSION A-1
SET STYLE command 1-3
Setting margins 4-2

SHOW LAYOUT
definition 1-17
example 4-2
Single spacing 2-6, 4-10
SKIP clause 4-10
Sorting data 3-1
Source data 3-5
SPACE clause 4-11
SPACE layout option 4-10
Spacing
double 4-10
items 4-1
line 4-1
of items and lines 4-11
single 4-10
Special characters
redefining 4-48
! 1-5
$, inserting in value 4-35
Stored commands 2-14
Strings
output line, insert in 4-2
spacing 4-10
Style options
definition 1-3
summary of 1-15
Subqueries
considerations 3-32
IF/THEN/ELSE 4-46
IN predicate 3-9
in report writer commands A-2
nested 3-32
select list 3-10
using 3-28
SUBTOTAL command 4-54
Subtotals
calculating 4-51, 4-56
conditional values, on 4-56
description of 1-8
example of 4-53
Subtotals (continued)
label for 4-55
SUM function 3-17, 3-18
Suppressing
headings 4-17
zero value 4-33, 4-38, 4-39

T

TAB clause
using 4-11
Tables
data in 3-2
description of 1-2
joining 3-4, 3-9
names 3-4
qualified name 2-2
SQL/MP 1-1
Template for display mask 4-33
Text
creating command file from 2-15
editor, creating a report with 2-15
Time
formatting overview 4-42
local 4-43
Timestamps, converting 4-45
to print name of month 4-45
TIME_FORMAT style option 4-43
TMF transaction status 2-3
TOTAL command, using 4-49
Totals
calculating 4-49
conditional values, on 4-56
description of 1-8
example 4-49
Trailing
blanks 3-12
zeros 4-38

U

UNDERLINE_CHAR style option 4-48
UNION operator 4-56
Unnamed parameters 3-25
UPSHIFT function 3-11
USA date-time format 4-44
User-defined parameters
in SELECT command 3-25

V

Values, justification of 4-32
VARCHAR data type, comparison of 3-12
VARCHAR_WIDTH style option 4-40
Views
creating 3-26
using 3-2

W

WHERE clause 1-7, 3-4
Wild-card characters 3-8
% 3-8
_ (underscore) 3-8
WINDOW layout option, using 2-8
Window, defining for a report 2-8
WRAP command A-1

Z

Zero suppression 4-33, 4-38, 4-39
Zeroes
leading 4-1
replacing with blanks 4-38
suppressing 4-33, 4-38, 4-39
trailing 4-1

Special Characters

! command 1-5
executing 2-14
% wild-card character 3-8, 3-12
& (ampersand) A-1
* in select list 3-6
*, in select list 3-32
_ (underscore) 3-8