CHAPTER 14

FORMATTING READABLE OUTPUT

LEARNING OBJECTIVES

After completing this chapter, you should be able to do the following:

- Add a column heading with a line break to a report
- Format the appearance of numeric data in a column
- Specify the width of a column
- Substitute a text string for a NULL value in a report
- Add a multiple-line header to a report
- Display a page number in a report
- Add a title and a footer to a report
- Change the setting of an environment variable
- Suppress duplicate report data
- Clear changes made by the COLUMN and BREAK commands
- Perform calculations in a report with the COMPUTE command
- Save report output to a file
- Save report output in HTML format in SQL*Plus

INTRODUCTION

In all the queries you've issued, the results have been displayed as a list. Although you have manipulated the columns displayed in output and used column aliases as column headings in query results, you haven't used SQL*Plus formatting features to create a report. Some formatting features in SQL*Plus include creating report headers and footers, formatting models for column data, suppressing duplicate data for group reports, and performing summary calculations. Although many formatting options are available, this chapter covers only the most commonly used. Understanding these formatting features helps you format output quickly in SQL*Plus; however, most complex reports are created with software programs and report generators, such as Oracle Reports or Crystal Reports.

The basic steps for creating a report in SQL*Plus are entering format option settings and entering a query to retrieve the rows you need. All examples in this chapter are shown with the Oracle 11*g* SQL*Plus client tool. Table 14-1 lists the SQL*Plus formatting options covered in this chapter.

TABLE 14-1 Common Formatting Options in SQL*Plus

Command	Description
START *or* @	Runs a script file
COLUMN	Defines the appearance of column headings and format of the column data
TTITLE	Adds a header to the top of each report page
BTITLE	Adds a footer to the bottom of each report page
BREAK	Suppresses duplicate data for specific columns when displayed in sorted order
COMPUTE	Performs calculations in a report based on the AVG, SUM, COUNT, MIN, and MAX statistical functions
SPOOL	Redirects output to a text file
SET MARKUP HTML ON	Saves SQL*Plus output in HTML format
COLUMN Options	**Description**
HEADING	Adds a column heading to a specified column
FORMAT	Defines the width of columns and applies specific formats to columns containing numeric data
NULL	Indicates text to be substituted for NULL values in a specified column

TABLE 14-1 Common Formatting Options in SQL*Plus (continued)

SQL*Plus Environment Variables	Description
UNDERLINE	Specifies the symbol used to separate a column heading from column contents
LINESIZE	Establishes the maximum number of characters allowed on a single line of output
PAGESIZE	Establishes the maximum number of lines allowed on one page of output

DATABASE PREPARATION

If you have executed the JLDB_Build_8.sql file as instructed in Chapter 8, you can work through the queries shown in this chapter.

THE COLUMN COMMAND

You can use the **COLUMN** command to format both a column heading and the data displayed in the column, depending on the options you use. Figure 14-1 shows the basic syntax of this command.

```
COLUMN [columnname|columnalias] [option]
```

FIGURE 14-1 Syntax of the COLUMN command

Now examine the syntax elements shown in Figure 14-1:

- *columnname*—The specified column is listed immediately after the COLUMN command.
- *columnalias*—You can assign an alias to a column in an SQL statement; however, if you do, you must use it instead of the column name to identify the column when it's referenced in an SQL*Plus formatting option.
- *option*—The options in the COLUMN command determine how the display is affected. Table 4-2 lists the options covered in this section.

TABLE 14-2 Options for the COLUMN Command

Option	Description
FORMAT *formatmodel*	Applies a specified format to column data
HEADING *columnheading*	Specifies the heading for a column
NULL *textmessage*	Identifies the text message to display in place of NULL values

The FORMAT Option

You use the **FORMAT** option to apply a format model to the data displayed in a column. For example, in previous chapters, a book's retail price of $54.50 was displayed as 54.5. You can use the FORMAT option to display the insignificant zero and include the dollar sign so that the retail price is shown as $54.50. Table 14-3 shows some formatting codes used with this option.

TABLE 14-3 Formatting Codes for Columns

Formatting Code	Description	Example	Output
9	Identifies the position of numeric data (suppresses leading zeros)	99999	54
$	Includes a floating dollar sign in output	$9999	$54
,	Indicates where to include a comma when needed (typically for the thousands and millions positions)	9,999	54 *or* 1,212
.	Specifies the number of decimal positions to display	9999.99	54.50
A*n*	Identifies the width of a column in a report	A32	Assigns a width of 32 spaces to a column

Suppose you need to create a report displaying the ISBN, title, cost, and retail price for books with a retail price higher than $50. To make sure each book's cost and retail price are shown with two decimal places, you can create a format model for data in the Cost and Retail columns. Look at the script and output shown in Figure 14-2, and then review these commands.

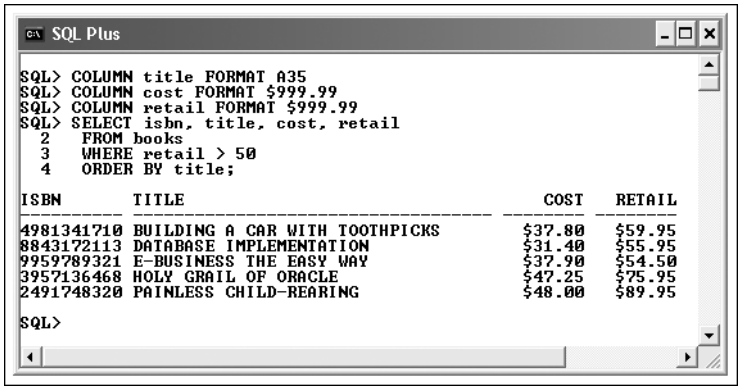

FIGURE 14-2 Script and output to create a formatted report

Examine the elements in Figure 14-2:

- The first line formats the Title column with a width of 35 spaces. Because the BOOKS table defines the column with a width of only 30 spaces, this larger setting creates extra blank space between the report's Title and Cost columns. By contrast, because the ISBN column doesn't have a width format specified, a default of one space is displayed after data in the ISBN column (and before the Title column).
- The second and third lines apply a format model to numeric data in the report's Cost and Retail columns. The format model uses a dollar sign and a series of nines. The dollar sign instructs Oracle 11*g* to include this symbol with each value displayed in the columns. The two nines after the decimal specify including two decimal positions. This option forces insignificant zeros to be included, which are suppressed by default in query results.
- After the Title, Cost, and Retail column formats have been defined, the query for retrieving the rows to display in the report is shown.
- Notice that the column-formatting commands don't end with a semicolon because it isn't required for SQL*Plus commands. However, the SQL statement must still end with a semicolon, as shown.

At this point, you might be thinking "Can't I use the TO_CHAR function in the SQL statement for numeric formatting tasks?" Yes, you could use the TO_CHAR function to add the dollar symbol and force the display of two decimal places. Furthermore, you could use concatenation to add width to displayed columns. So why use the COLUMN FORMAT command? It's somewhat a matter of preference. However, you might need different output formats for certain queries. In this situation, the SQL*Plus COLUMN command can be useful because you could reuse the SQL statement, and the SQL*Plus format options could be used to vary the display, depending on the situation.

NOTE

The COLUMN command doesn't have a specific format model for a DATE datatype column, other than increasing the column width. Therefore, the TO_CHAR function must be used to format date values.

Even though some COLUMN command options can be accomplished easily with SQL statements, this isn't true of all SQL*Plus formatting features. Many can't be done easily—or at all—with SQL statements, as you'll discover while working through this chapter.

If formatting commands will be used repeatedly with the SQL statement to produce a standard report, you can save the script to a file and run it instead of entering the commands each time the report is needed. The script shown in Figure 14-2 can be entered in a file with either of these approaches:

- Enter the commands to be saved, and then type **ed** or **edit** at the SQL> prompt in SQL*Plus to open a text editor, such as Notepad. Select **File, Save As** from the menu, and name the script with an .sql extension.
- Open a text editor outside SQL*Plus, enter the commands, and then save the file with an .sql extension. If you use a word-processing program rather than a text editor, it might embed hidden codes in the file, and the script file doesn't run correctly. If you must use a word-processing program, make sure that when you save the file, you change the file type to a text or an ANSI file (depending on the options available) to prevent embedding hidden codes in the script.

After the script has been created, you can run it in SQL*Plus by entering the SQL*Plus START command, shown in Figure 14-3.

```
START drive:\filename
```

FIGURE 14-3 Command to run a script in SQL*Plus

TIP

If you save the script file with an .sql extension, you don't have to include the extension in the START command. If another file extension is used, such as .txt, you must include it with the filename.

In Figure 14-3, the **START** keyword indicates that a script should be executed. The *drive*:\ specifies the drive and pathname where the file is stored; *filename* represents the name of the script file to run.

NOTE

The "at" sign (@ symbol) can be substituted for the START keyword, as in `@d:\filename`.

Return to the report generated in Figure 14-2. After reviewing the results, you want to make improvements. First, you want to add more space between the ISBN and Title

columns. Second, you want to remove the dollar signs from the Cost and Retail columns because repeating a dollar sign for every currency value in a column can be distracting and add unnecessary clutter. Modify the script as shown in Figure 14-4 to make these changes.

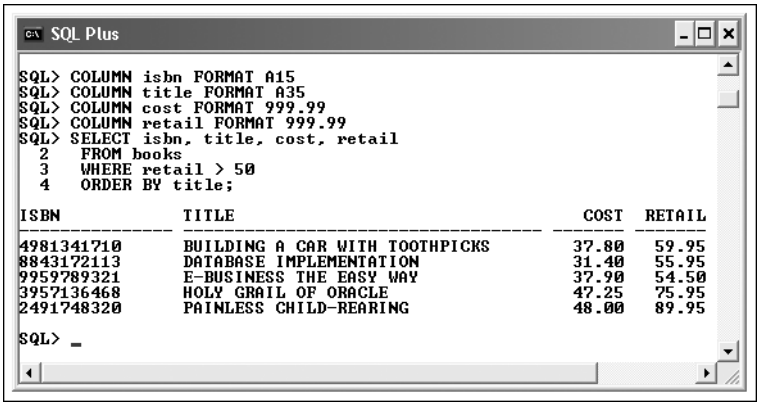

FIGURE 14-4 Revised script

Notice that the space between the ISBN and Title columns has been increased. Because the ISBN column is defined in the BOOKS table as a VARCHAR2 datatype, the data is left-aligned in the column. This alignment results in excess space on the right side of the ISBN column and creates more space between the last digit of a book's ISBN and its corresponding title. On the other hand, columns defined as a NUMBER datatype display their contents right-aligned. Therefore, any excess space that's assigned appears in front of values in the column.

To add space between the Cost and Retail columns, increase the number of nines listed in each column's format model. For example, there are four blank spaces between the last digit in the Cost column and the first digit in the Retail column. To add six more spaces between the columns, simply add six 9s to the format model for the Retail column, as shown in Figure 14-5. This width change affects only text output, as you've seen with character columns.

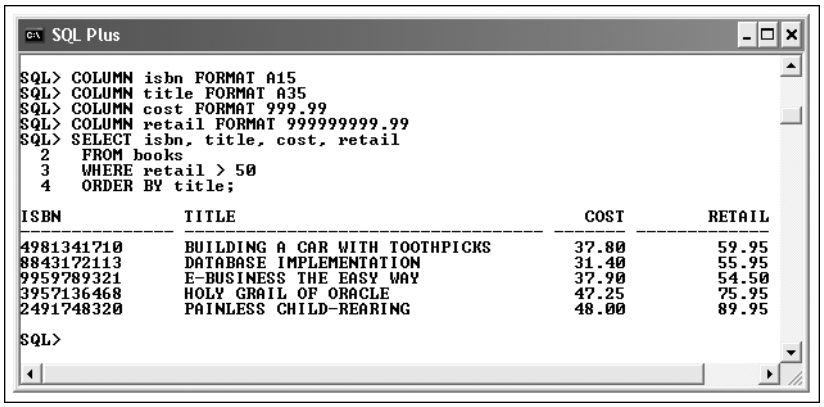

FIGURE 14-5 Modifying the width of numeric column output

Because no book has a retail price of more than $99.99, any excess 9s in the Retail column's format model are represented in the output by a blank space. In essence, the series of 9s establishes the column width. In addition, the number of dashes separating column headings and their contents has increased. The additional dashes under in the Retail column's heading aren't balanced in appearance with the dashes under the Cost column's heading, but you learn a simple solution to this problem in the next section.

The HEADING Option

The **HEADING** option of the COLUMN command is used to specify a column heading. By default, the report's column headings are the same as the BOOKS table's column names. The HEADING option is similar to a column alias: It provides a substitute heading for output. There are, however, two important differences:

- A column name assigned by the HEADING option can't be referenced in a SELECT statement.
- A column name assigned by the HEADING option can contain line breaks. In other words, the column name can be displayed on more than one line.

If you had assigned the column alias "Retail Price" to the Retail column in a SELECT statement, the column would be 12 spaces wide in the output (one space for each letter and one for the space between words). If you use the HEADING option, however, you can specify placing the word "Retail" on one line and "Price" on a second line to give the report a more professional appearance and not waste space. To do this, you use a single vertical bar (|) to indicate where the line break should occur in a column heading, as in `'Retail|Price'`. (Note the use of single quotation marks to create a literal string.)

Modify the previous script to add a HEADING option to the Retail column, as shown in Figure 14-6.

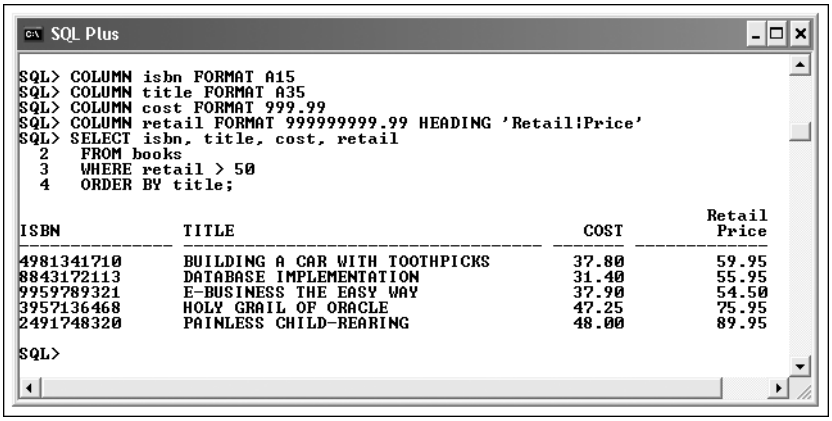

FIGURE 14-6 Heading modified for the Retail column

Of course, now the Title and Cost columns have uppercase column headings, but the Retail column's heading is mixed case, as in the HEADING option. To change the letter case of the Title and Cost column headings, you can include a HEADING option for these columns or use column aliases enclosed with double quotation marks to specify the correct letter case.

However, the problem with the dashes under the column headings still remains. The SQL*Plus environment variable **UNDERLINE** specifies using dashes to separate headings from data in text output. An environment variable, also called a system variable, determines how certain elements are displayed in SQL*Plus. The **SET** command is used to change the value assigned to an environment variable. The UNDERLINE variable setting determines whether a heading separator is displayed and what character is used.

In your report, you want to create the illusion that a column's assigned width is less than its actual setting but still leave adequate spacing between columns. This process has two steps. First, change the UNDERLINE variable so that no dashes (or any other symbol) are displayed. Second, add the number of dashes you want to the HEADING option for each column. Follow these steps, shown in Figure 14-7:

1. At the SQL> prompt, enter **SET UNDERLINE OFF** and press **Enter.** (This command specifies no underlining to separate column headings from column data in the output.)
2. Add a **HEADING** option for each column to include the number of dashes you want. Note that you should include a vertical bar to specify the dashes appearing on a second line. In this example, the number of dashes matches the character width setting for the column output.

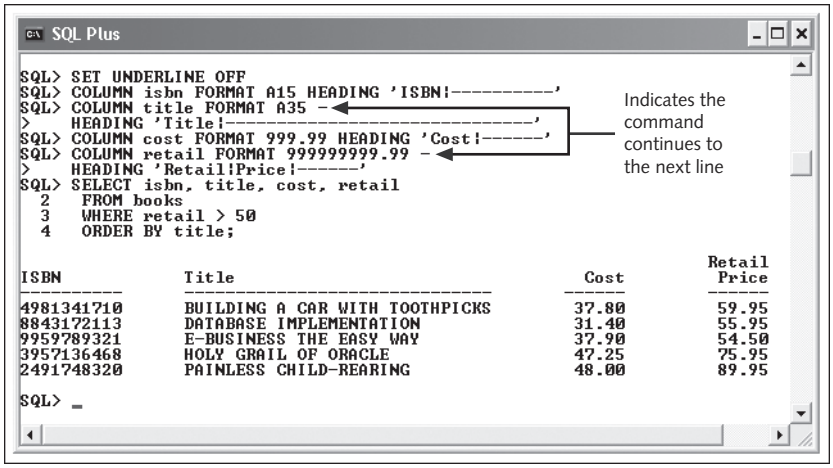

FIGURE 14-7 Controlling the UNDERLINE variable's characters

The COLUMN command for the Title and Retail columns is on two lines. When an SQL*Plus command needs to be entered on more than one line, use a dash at the end of the first line to indicate that the command continues on the next line. In this case, the COLUMN

command for both columns was broken just before the HEADING option. For better readability, the second line of each command is indented, but indenting isn't a requirement.

TIP

The UNDERLINE variable can be set to something other than the default, which is a single dash. For example, `SET UNDERLINE '*'` substitutes an asterisk for a single dash.

The NULL Option

Although this report doesn't contain NULL values, you'll probably work with other reports that do. In some instances, blank spaces are suitable for NULL values. For example, JustLee Books management wants a report showing the lag time for shipping current orders. You might want to substitute the words "Not Shipped" for any NULL values in the Shipdate column. To do this, you can use the **NULL** option of the COLUMN command, as shown in Figure 14-8. The first line sets the UNDERLINE variable on again.

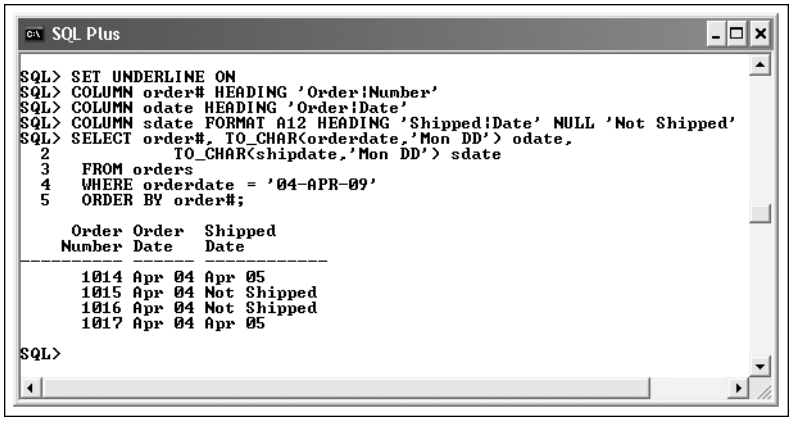

FIGURE 14-8 Substituting text data for a NULL value

NOTE

You can also use the NVL2 function in SQL to substitute a value for a NULL value. However, NVL2 is included in an SQL statement, and the NULL option is used in an SQL*Plus command.

The NULL option in the COLUMN command for the Shipdate column instructs Oracle 11g to display a text message in place of the NULL value in the report's output. Because the text message is a literal string, it's displayed in the same letter case used in the NULL option.

In the SELECT statement, data in the Orderdate and Shipdate columns is retrieved with the TO_CHAR function, which allows specifying a format model. However, when the TO_CHAR function is used, the COLUMN command can't reference the column names Orderdate and Shipdate. Recall that when a function is applied to column data, the output shows the function as the column name. The same thing happens when a function is used in a SELECT statement for a report. Therefore, when a function is used, you must assign a column alias so that the COLUMN command can reference the column with the alias. In this case, `odate` and `sdate` are assigned as the column aliases and used in the HEADING option to provide a column heading for each date column.

HEADERS AND FOOTERS

To create a more polished look for reports, headers and footers are used. Typically, a header serves as the report title, and footers are often used for page numbers or a report author's name, for example. The **TTITLE** command for the header specifies the text or variables (such as the date) to display at the top of the report. The **BTITLE** command defines what's printed at the bottom of the report. Figure 14-9 shows the syntax of the TTITLE and BTITLE commands.

```
TTITLE|BTITLE [option [text|variable]…] [ON|OFF]
```

FIGURE 14-9 Syntax of the TTITLE and BTITLE commands

You can use options to specify the format of header or footer data, where data should be placed, and on which line data should start. Table 14-4 lists some available options for TTITLE and BTITLE.

TABLE 14-4 Options for the TTITLE and BTITLE Commands

Option	Description
CENTER	Centers data between the report's left and right margins
FORMAT	Specifies a format model for data (same elements as for the COLUMN command)
LEFT	Aligns data with the report's left margin
RIGHT	Aligns data with the report's right margin
SKIP *n*	Specifies the number of lines to skip before data display resumes

In Figure 14-9, note that `[text|variable]` means you can apply alignment and FORMAT options to text entered as a literal string or to SQL*Plus variables. Some valid SQL*Plus variables are listed in Table 14-5.

TABLE 14-5 SQL*Plus Variables

Variable	Description
SQL.LNO	Current line number; often included to determine report length or set how many lines can be printed on a page
SQL.PNO	Current page number; helpful for those reading a report
SQL.RELEASE	Current Oracle release number; used for documentation purposes and usually included in case a report doesn't display correctly after an Oracle upgrade
SQL.USER	Current username

Next, you add a header to the previous script to make the report look more professional and indicate its contents. For this example, a good header is "Books in Inventory," centered over the report's contents. In addition, page numbers should be displayed on a separate line at the right side of the report. To add headers and footers, you also need to understand the LINESIZE and PAGESIZE settings, discussed next.

NOTE

If you don't include a format model for the header, the date and page numbers are displayed automatically.

The LINESIZE Environment Variable

How do you determine the center or right or left side of a report? The placement of aligned text is based on the **LINESIZE** environment variable. For example, if LINESIZE has been set to 100, the center of the report is 50 characters from each margin. To ensure correct placement of the header, you need to calculate LINESIZE to find the report's exact width based on column widths. The script in Figure 14-10 shows how to add the LINESIZE and TTITLE definitions.

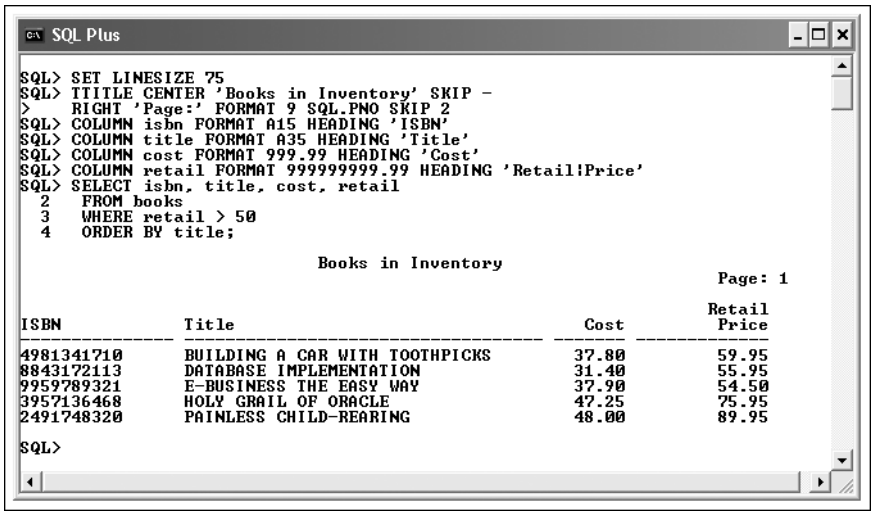

FIGURE 14-10 Adding a report header

Take a closer look at the script elements in Figure 14-10:

- The first line defines a LINESIZE setting of 75 spaces to establish the report's right margin.
- The second line adds a report header. The first part of the TTITLE command specifies centering the words "Books in Inventory" as the header. The command then advances to the report's next line, using the **SKIP** option to begin the next part of the header on a separate line.
- The third line specifies displaying the text "Page:" followed by the value stored in the SQL.PNO variable to display the page number. A format model is added to indicate the page number's expected number of digits. If the format model isn't included, several blank spaces might be displayed between the "Page:" label and the page number.
- At the end of the third line, the command specifies skipping two lines before beginning the column headings.

Now that you've added the report header, it's time to format the footer. Technically, a report's header and footer are displayed on every page. Many organizations prefer having a message such as "End of Report" or "End of Job" displayed at the end of a report instead to make sure no pages are missing. Because the Books in Inventory report is only one page, however, the message "End of Report" is displayed as the footer to indicate that the report has no other pages. In addition, JustLee Books management requires showing the name of the user running a report, so the username is included as part of the footer, as you see in the next section.

The PAGESIZE Environment Variable

Because the footer placement depends on the length of the page, the **PAGESIZE** environment variable needs to be set. Although standard printed reports generally have 60 lines per

page (leaving a small margin at the top and bottom), a standard computer monitor displays about 32 lines, depending on the resolution settings. Because this report is being displayed on a computer screen, the PAGESIZE variable is set to 15 lines, which allows displaying both the header and footer onscreen at the same time.

Figure 14-11 shows how to add the BTITLE command so that the words "End of Report" are centered at the bottom of the page and the words "Run by:" followed by the current user's name are placed at the lower-right side of the page. A format model allowing five spaces is used for the username so that it's displayed flush with the right margin. However, if other users might run this report, extra spaces might be needed to accommodate longer names.

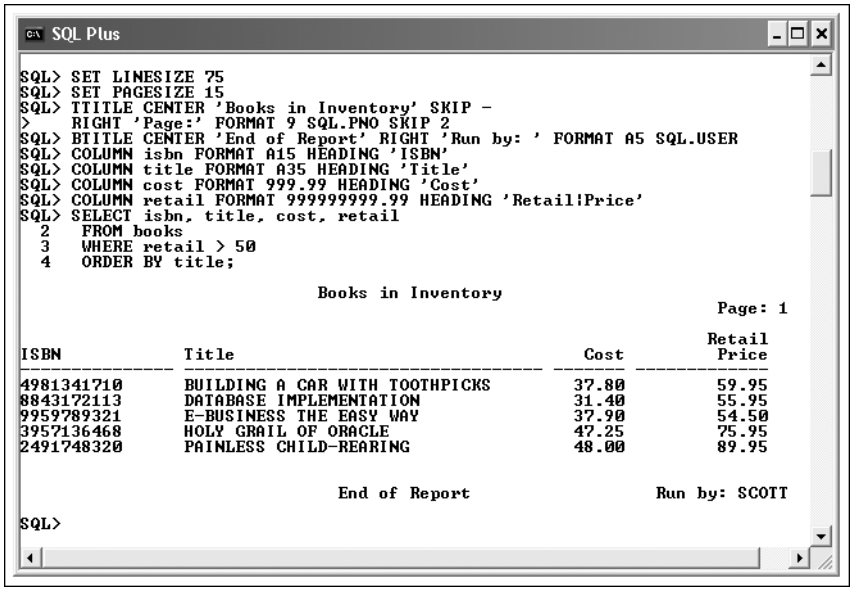

FIGURE 14-11 Adding a footer to the report

Suppose that after the report has been displayed, you can't remember the exact structure of the TTITLE command used to generate the report header. You can use the TTITLE command in SQL*Plus to display the current header settings, as shown in Figure 14-12.

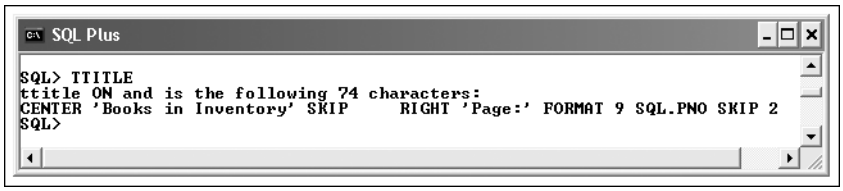

FIGURE 14-12 Checking current header settings

You can also check current settings for the BTITLE and COLUMN commands. With the COLUMN command, make sure you list the name of the column you're requesting (such as `COLUMN isbn`) to display its settings.

THE BREAK COMMAND

The **BREAK** command is used to suppress duplicate data in a report and is especially useful for reports containing groups of data. For example, you're creating a list of all books in inventory, sorted by category, which results in displaying the Computer, Business, and other category column names multiple times. If you use the BREAK command, each category column name appears only once. The BREAK command also includes options for skipping lines after each group and displaying each group on a separate page. The syntax of the BREAK command is shown in Figure 14-13.

```
BREAK ON columnname|columnalias [ON …] [skip n|page]
```

FIGURE 14-13 Syntax of the BREAK command

The effectiveness of the BREAK command is determined by how data is sorted in the SELECT statement. The specified column's contents are displayed once, and the display of all subsequent rows is suppressed until the value in the column changes. For example, if you use the BREAK command with the Category column in a report listing books sorted by category, the first time the Computer category is encountered, the word "COMPUTER" is displayed. The category name is then suppressed until the next category name is encountered. However, if the report data isn't displayed in sorted order, "COMPUTER" might be listed several times in the report. Therefore, you should always include the ORDER BY clause when using the BREAK command.

Duplicate values are suppressed only when they occur in sequence. As soon as the name of the category changes, the new value is displayed. If you want to include subgroupings, you can add another ON clause, followed by the name of the column used for the subgrouping.

Suppose you want to produce a report showing the amount due for orders in the ORDERS table. The orders should be sorted by customer number in the results. Instead of duplicating customer numbers in the report, you can use the BREAK command as shown in Figure 14-14. (The WHERE clause filters customers to avoid creating a long report for this example.)

The BREAK command suppresses duplicate values in the Customer# column. To make the report easier to read, the SKIP keyword could have been used to add a blank line between each group.

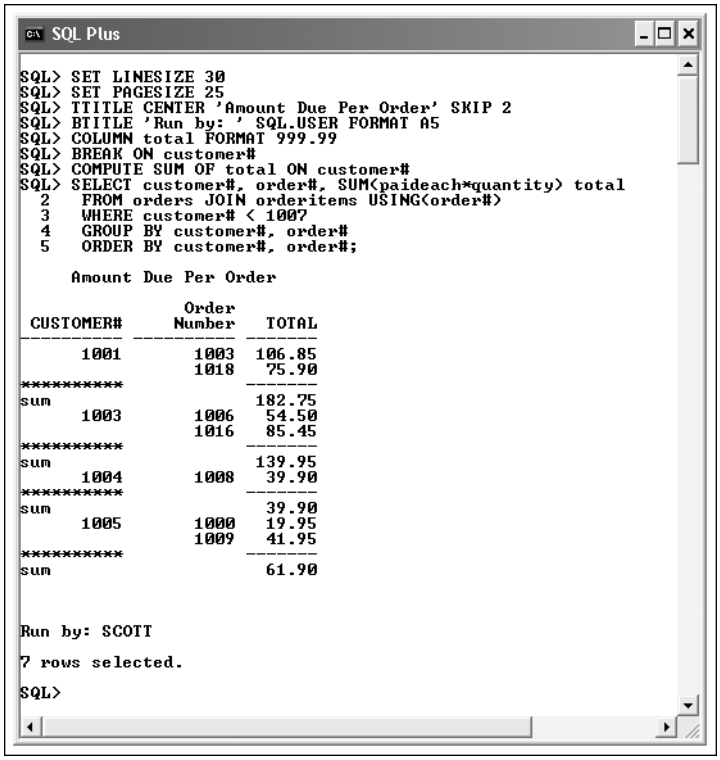

FIGURE 14-14 Script for creating a report listing the amount due for each order

THE CLEAR COMMAND

The **CLEAR** command is used to clear settings for the BREAK and COLUMN commands. Settings for these commands remain even after a report has been finished. For example, if your next query contains a column or alias named "title," the 999.99 format is used because this setting was added in the previous script. Therefore, any time you use the COLUMN or BREAK commands in a script, you should always clear the settings at the end of the script so that they don't affect subsequent reports. Figure 14-15 shows the syntax of the CLEAR command.

```
CLEAR COLUMN|BREAK
```

FIGURE 14-15 Syntax of the CLEAR command

Figure 14-16 shows adding the CLEAR command at the end of the previous script to clear the COLUMN and BREAK settings. Notice that the CLEAR command is issued twice: once for the BREAK command's settings and once for the COLUMN command's settings. After adding the CLEAR command, any settings used for the report are cleared automatically after the report has been displayed.

```
SET LINESIZE 30
SET PAGESIZE 25
TTITLE CENTER 'Amount Due Per Order' SKIP 2
BTITLE 'Run by: ' SQL.USER FORMAT A5
COLUMN total FORMAT 999.99
BREAK ON customer#
SELECT customer#, order#, SUM(paideach*quantity) total
 FROM orders JOIN orderitems USING(order#)
 WHERE customer# < 1007
 GROUP BY customer#, order#
 ORDER BY customer#, order#;
CLEAR BREAK
CLEAR COLUMN
```

FIGURE 14-16 CLEAR command added to the end of a script

THE COMPUTE COMMAND

In addition to determining the total amount due for each order, you can determine how much each customer owes with the **COMPUTE** command. The COMPUTE command can be used with the AVG, SUM, COUNT, MAX, and MIN keywords to determine averages, totals, number of occurrences, highest value, and lowest value. Figure 14-17 shows the syntax of the COMPUTE command.

```
COMPUTE statisticalfunction OF columnname ON groupname
```

FIGURE 14-17 Syntax of the COMPUTE command

For example, you want to determine the total amount each customer owes. To do this, add `COMPUTE SUM OF total ON customer#` to the script, as shown in Figure 14-18. This command instructs Oracle 11g to determine the total (`SUM`) amount due (`OF total`) for each customer (`ON customer#`). The LINESIZE and PAGESIZE variables have been changed to 30 and 25 to accommodate the data added to the report. To reflect the new data, the report header has been changed, too.

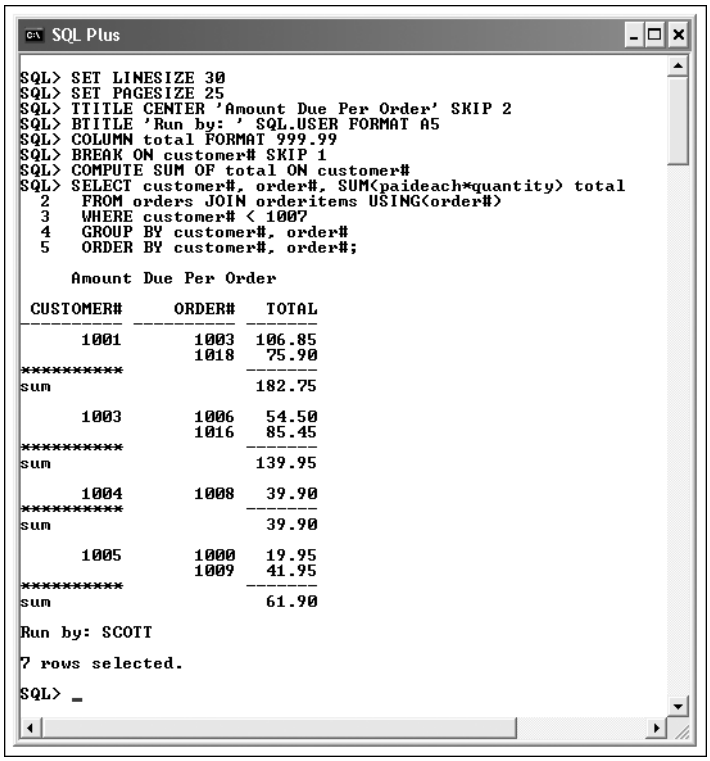

FIGURE 14-18 Adding subtotals to the report

NOTE

The CLEAR COMPUTE command can be added to clear settings for the COMPUTE command.

THE SPOOL COMMAND

Management often wants printed copies of reports you generate, and you can redirect results displayed on the screen to a file that can be opened and printed. In SQL*Plus, you use the **SPOOL** command to save a report's results to a text file. If you add this command to the beginning of the script, the entire script, including the report's format commands, is saved to the file. Figure 14-19 shows the syntax of the SPOOL command.

```
SPOOL drive:/filename
```

FIGURE 14-19 Syntax of the SPOOL command

For example, if you want a report's results *written* to the reportresults.txt file on your C drive for printing, add `SPOOL c:\reportresults.txt` to the beginning of the script, as shown in Figure 14-20. Include the SPOOL OFF command at the end of the script to stop the spooling process.

```
SPOOL c:\reportresults.txt
SET LINESIZE 30
SET PAGESIZE 25
TTITLE CENTER 'Amount Due Per Order' SKIP 2
BTITLE 'Run by: ' SQL.USER FORMAT A5
COLUMN total FORMAT 999.99
BREAK ON customer# SKIP 1
COMPUTE SUM OF total ON customer#
SELECT customer#, order#, SUM(paideach*quantity) total
 FROM orders JOIN orderitems USING(order#)
 WHERE customer# < 1007
 GROUP BY customer#, order#
 ORDER BY customer#, order#;
SPOOL OFF
CLEAR BREAK
CLEAR COLUMN
```

FIGURE 14-20 A script with the SPOOL command

If you want to save just the report output to a file for printing, instead of the script *and* output, add the **SET ECHO OFF** command to the beginning of the script. This command suppresses script commands in the output, so the spooled file contains only the report data.

HTML OUTPUT

You might want report output in HTML form so that you can post it to a Web site. To produce HTML output in SQL*Plus, use the **SET MARKUP HTML ON** command after the SPOOL command. Figure 14-21 shows an example of a script with these commands added.

```
SET ECHO OFF
SPOOL c:\reportresults.html
SET MARKUP HTML ON
SELECT isbn, title, cost, retail
 FROM books
 WHERE retail > 50
 ORDER BY title;
SET MARKUP HTML OFF
SPOOL OFF
SET ECHO ON
```

FIGURE 14-21 A script for producing HTML output

You need to save the script to a file and run it by using the START command so that only HTML report output is saved to the file. The SET ECHO OFF command prevents including formatting and query statements in the output file. If you open the file in a text editor, you see HTML coding. To see the Web page, open the file in a Web browser.

Chapter Summary

- Reports can be created with a variety of formatting commands in SQL*Plus. They are SQL*Plus commands, *not* SQL statements.
- Commands to format reports are usually entered in a script and then executed. The SELECT statement appears after the SQL*Plus formatting commands in the script.
- To run a script file in SQL*Plus, use the START command or the @ symbol before the path and filename.
- The COLUMN command has HEADING, FORMAT, and NULL options for controlling the appearance of columns in a report.
- The HEADING option is used to provide a column heading. A single vertical bar indicates where a line break should occur in the column heading.
- The SQL*Plus UNDERLINE environment variable specifies the symbol for separating column headings from column contents.
- The FORMAT option can be used to create a format model for displaying numeric data or to specify the width of nonnumeric columns.
- The NULL option specifies a text message to display if the column contains a NULL value.
- The TTITLE command identifies the header to display at the top of the report. It can include format models, alignment options, and variables, if needed.
- The BTITLE command identifies the footer to display at the bottom of the report. It follows the same syntax as the TTITLE command.
- Alignment options in the TTITLE and BTITLE commands are affected by the LINESIZE and PAGESIZE variables.
- The BREAK command is used to suppress duplicate data.
- At the end of each script that has a COLUMN or BREAK command, the CLEAR command should be used to clear any settings used in the script.
- The COMPUTE command can be used to calculate totals in a report based on groupings.
- The SPOOL command instructs Oracle 11*g* to redirect output to a file. This command is available only in SQL*Plus.
- The SET MARKUP HTML ON command allows saving SQL*Plus output in HTML format.

Chapter 14 Syntax Summary

The following table summarizes the syntax you have learned in this chapter. You can use the table as a study guide and reference.

Syntax Guide

SQL*Plus Command	Description	Example
START *or* @	Runs a script	**START C:\titlereport.sql** *or* **@C:\titlereport.sql**

SQL*Plus Command	Description	Example
TTITLE[*option*[*text*|*variable*] . . .] [ON|OFF]	Defines a report header	**TTITLE 'Top of|Report'**
BTITLE[*option*[*text*|*variable*] . . .] [ON|OFF]	Defines a report footer	**BTITLE 'Bottom of|Report'**
BREAK ON *columnname*|*columnalias*[ON . . .] [skip n|page]	Suppresses duplicate data in a specified column	**BREAK ON title**
COMPUTE *statisticalfunction* OF columnname ON groupname	Performs calculations in a report, based on the AVG, SUM, COUNT, MIN, and MAX statistical functions	**COMPUTE SUM OF total ON customer#**
SPOOL *filename*	Redirects report output to a text file	**SPOOL c:\reportresults.txt**
CLEAR COLUMN|BREAK	Clears settings for the COLUMN and BREAK commands	**CLEAR COLUMN**
SET MARKUP HTML ON|OFF	Specifies whether SQL*Plus output should be saved in HTML format	**SET MARKUP HTML ON**
COLUMN Command Options		
FORMAT *formatmodel*	Applies a specified format to column data	**COLUMN title FORMAT A30**
HEADING *columnheading*	Indicates a heading for the specified column	**COLUMN title HEADING 'Book|Title'**
NULL *textmessage*	Identifies text to display in place of NULL values	**COLUMN shipdate NULL 'Not Shipped'**

Syntax Guide (continued)

SQL*Plus Command	Description	Example
TTITLE and BTITLE Command Options		
CENTER	Centers data between a report's left and right margins	**TTITLE CENTER 'Report Title'**
FORMAT	Applies a format model to displayed data (uses the same elements as the COLUMN command)	**BTITLE 'Page:' SQL.PNO FORMAT 9**
LEFT	Aligns data with the report's left margin	**TTITLE LEFT 'Internal Use Only'**
RIGHT	Aligns data with the report's right margin	**TTITLE RIGHT 'Internal Use Only'**
SKIP *n*	Specifies the number of lines to skip before the display resumes	**TTITLE CENTER 'ReportTitle' SKIP 2 RIGHT 'Internal Use Only'**
SQL*Plus Variables		
SQL.LNO	Retrieves the current line number	**BTITLE 'End of Page onLine: ' SQL.LNO FORMAT 99**
SQL.PNO	Retrieves the current page number	**BTITLE 'Page: ' SQL.PNO FORMAT 9**
SQL.RELEASE	Retrieves the current Oracle release number	**BTITLE 'Processed on Oracle release number: ' SQL.RELEASE**
SQL.USER	Retrieves the current username	**BTITLE 'Run By: ' SQL.USER FORMAT A8**
SQL*Plus Environment Variables		
UNDERLINE '*x*'|OFF	Specifies the symbol for separating a column heading from column contents	**SET UNDERLINE '='**

SQL*Plus Command	Description	Example
SQL*Plus Environment Variables		
`LINESIZE` *n*	Establishes the maximum number of characters allowed on a single line	**`SET LINESIZE 35`**
`PAGESIZE` *n*	Establishes the maximum number of lines allowed on a page	**`SET PAGESIZE 27`**