



# Producing Readable Output with *i*SQL\*Plus

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Schedule:	Timing	Topic
	35 minutes	Lecture
	35 minutes	Practice
	70 minutes	Total

# Objectives

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

- **Produce queries that require a substitution variable**
- **Customize the *iSQL\*Plus* environment**
- **Produce more readable output**
- **Create and execute script files**

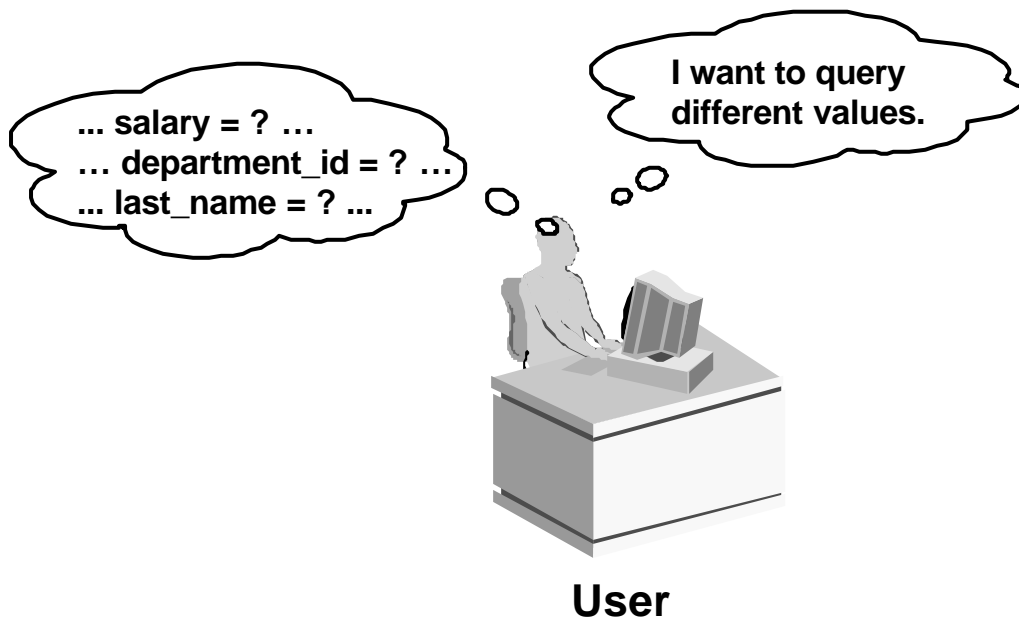
ORACLE

## Lesson Aim

In this lesson, you will learn how to include *iSQL\*Plus* commands to produce more readable SQL output.

You can create a command file containing a `WHERE` clause to restrict the rows displayed. To change the condition each time the command file is run, you use substitution variables. Substitution variables can replace values in the `WHERE` clause, a text string, and even a column or a table name.

# Substitution Variables



## Substitution Variables

The examples so far have been hard-coded. In a finished application, the user would trigger the report, and the report would run without further prompting. The range of data would be predetermined by the fixed `WHERE` clause in the *iSQL\*Plus* script file.

Using *iSQL\*Plus*, you can create reports that prompt the user to supply their own values to restrict the range of data returned by using substitution variables. You can embed *substitution variables* in a command file or in a single SQL statement. A variable can be thought of as a container in which the values are temporarily stored. When the statement is run, the value is substituted.

## Instructor Note

Demo: `7_varno.sql`, `7_varyes.sql`

Purpose: To illustrate returning all rows and using a case-insensitive query with substitution variables.

With *iSQL\*Plus* 9.0.1.0.1, there is a bug when using `&substitution` and wildcards (%) for character values.

This bug has been reported. The `7_varyes.sql` will produce an error in *iSQL\*Plus*, but the concept is important for students continuing classes using other products (such as Forms, Reports). You may want to demonstrate `7_varyes.sql` in the *SQL\*Plus* environment as an option.

# Substitution Variables

Use *iSQL\*Plus* substitution variables to:

- **Temporarily store values**
  - Single ampersand (&)
  - Double ampersand (&&)
  - **DEFINE** command
- **Pass variable values between SQL statements**
- **Dynamically alter headers and footers**

ORACLE

7-4

Copyright © Oracle Corporation, 2001. All rights reserved.

## Substitution Variables

In *iSQL\*Plus*, you can use single ampersand (&) substitution variables to temporarily store values. You can predefine variables in *iSQL\*Plus* by using the **DEFINE** command. **DEFINE** creates and assigns a value to a variable.

### Examples of Restricted Ranges of Data

- Reporting figures only for the current quarter or specified date range
- Reporting on data relevant only to the user requesting the report
- Displaying personnel only within a given department

## Other Interactive Effects

Interactive effects are not restricted to direct user interaction with the **WHERE** clause. The same principles can be used to achieve other goals. For example:

- Dynamically altering headers and footers
- Obtaining input values from a file rather than from a person
- Passing values from one SQL statement to another

*iSQL\*Plus* does not support validation checks (except for data type) on user input.

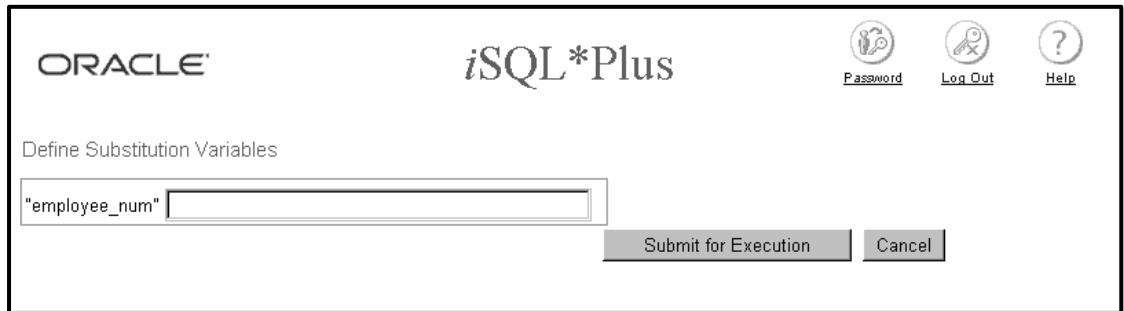
## Instructor Note

A substitution variable can be used anywhere in SQL and *iSQL\*Plus* commands, except as the first word entered at the command prompt.

# Using the & Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value.

```
SELECT  employee_id, last_name, salary, department_id
FROM    employees
WHERE   employee_id = &employee_num ;
```



The screenshot shows the iSQL\*Plus web interface. At the top, there's a header with the Oracle logo, the text 'iSQL\*Plus', and three circular icons labeled 'Password', 'Log Out', and 'Help'. Below the header, the text 'Define Substitution Variables' is displayed. Underneath, there's a text input field containing the string '"employee\_num"'. To the right of this field are two buttons: 'Submit for Execution' and 'Cancel'.

ORACLE

## Single-Ampersand Substitution Variable

When running a report, users often want to restrict the data returned dynamically. *iSQL\*Plus* provides this flexibility by means of user variables. Use an ampersand (&) to identify each variable in your SQL statement. You do not need to define the value of each variable.

Notation	Description
<i>&amp;user_variable</i>	Indicates a variable in a SQL statement; if the variable does not exist, <i>iSQL*Plus</i> prompts the user for a value ( <i>iSQL*Plus</i> discards a new variable once it is used.)

The example on the slide creates an *iSQL\*Plus* substitution variable for an employee number. When the statement is executed, *iSQL\*Plus* prompts the user for an employee number and then displays the employee number, last name, salary, and department number for that employee. With the single ampersand, the user is prompted every time the command is executed, if the variable does not exist.

# Using the & Substitution Variable

The screenshot shows the Oracle iSQL\*Plus interface. At the top, the Oracle logo and 'iSQL\*Plus' text are visible. On the right, there are links for 'Password', 'Log Out', and 'Help'. The main area is titled 'Define Substitution Variables'. It shows a prompt for the variable '&employee\_num' with the value '101' entered. A circled '1' points to the input field. Below the input field are two buttons: 'Submit for Execution' and 'Cancel'. A circled '2' points to the 'Submit for Execution' button. Below the input area, the SQL statement is shown: 'old 3: WHERE employee\_id = &employee\_num' and 'new 3: WHERE employee\_id = 101'. The results of the query are displayed in a table with four columns: EMPLOYEE\_ID, LAST\_NAME, SALARY, and DEPARTMENT\_ID. The first row shows the data for employee\_id 101: Kochhar, 17000, and 90.

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

ORACLE

## Single-Ampersand Substitution Variable

When iSQL\*Plus detects that the SQL statement contains an &, you are prompted to enter a value for the substitution variable named in the SQL statement. Once you enter a value and click the Submit for Execution button, the results are displayed in the output area of your iSQL\*Plus session.

# Character and Date Values with Substitution Variables

Use single quotation marks for date and character values.

```
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title' ;
```

Define Substitution Variables

"job\_title" IT\_PROG

Submit for Execution

Cancel

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

ORACLE

## Specifying Character and Date Values with Substitution Variables

In a WHERE clause, date and character values must be enclosed within single quotation marks. The same rule applies to the substitution variables.

Enclose the variable in single quotation marks within the SQL statement itself.

The slide shows a query to retrieve the employee names, department numbers, and annual salaries of all employees based on the job title value of the *iSQL\*Plus* substitution variable.

**Note:** You can also use functions such as UPPER and LOWER with the ampersand. Use UPPER( '&job\_title' ) so that the user does not have to enter the job title in uppercase.

# Specifying Column Names, Expressions, and Text

Use substitution variables to supplement the following:

- WHERE conditions
- ORDER BY clauses
- Column expressions
- Table names
- Entire SELECT statements

ORACLE

7-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## Specifying Column Names, Expressions, and Text

Not only can you use the substitution variables in the WHERE clause of a SQL statement, but these variables can also be used to substitute for column names, expressions, or text.

### Example

Display the employee number and any other column and any condition of employees.

```
SELECT  employee_id, &column_name
FROM    employees
WHERE   &condition;
```

"column\_name"

"condition"

EMPLOYEE_ID	JOB_ID
200	AD_ASST

I the preceding statement.

**Note:** A substitution variable can be used anywhere in the SELECT statement, except as the first word entered at the command prompt.



# Specifying Column Names, Expressions, and Text

```
SELECT      employee_id, last_name, job_id,
            &column_name
FROM        employees
WHERE       &condition
ORDER BY    &order_column ;
```

Define Substitution Variables

"column\_name" salary  
"condition" salary > 15000  
"order\_column" last\_name

Submit for Execution

Cancel

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
102	De Haan	AD_VP	17000
100	King	AD PRES	24000
101	Kochhar	AD_VP	17000

ORACLE

7-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## Specifying Column Names, Expressions, and Text (continued)

The slide example displays the employee number, name, job title, and any other column specified by the user at run time, from the EMPLOYEES table. You can also specify the condition for retrieval of rows and the column name by which the resultant data has to be ordered.

### Instructor Note

Demo: 7\_expr.sql

Purpose: To illustrate changing column names and conditions by using substitution variables

# Defining Substitution Variables

- You can predefine variables using the *iSQL\*Plus* **DEFINE** command.  
**DEFINE** *variable* = *value* creates a user variable with the CHAR data type.
- If you need to predefine a variable that includes spaces, you must enclose the value within single quotation marks when using the **DEFINE** command.
- A defined variable is available for the session

ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## Defining Substitution Variables

You can predefine user variables before executing a **SELECT** statement. *iSQL\*Plus* provides the **DEFINE** command for defining and setting substitution variables:

Command	Description
<b>DEFINE</b> <i>variable</i> = <i>value</i>	Creates a user variable with the CHAR data and assigns a value to it
<b>DEFINE</b> <i>variable</i>	Displays the variable, its value, and its data type
<b>DEFINE</b>	Displays all user variables with their values and data types

## Instructor Note

Mention that *iSQL\*Plus* commands can continue onto multiple lines and that they require the continuation character, the hyphen.

# DEFINE and UNDEFINE Commands

- A variable remains defined until you either:
  - Use the UNDEFINE command to clear it
  - Exit *iSQL\*Plus*
- You can verify your changes with the DEFINE command.

```
DEFINE job_title = IT_PROG
DEFINE job_title
DEFINE JOB_TITLE          = "IT_PROG" (CHAR)
```

```
UNDEFINE job_title
DEFINE job_title
SP2-0135: symbol job_title is UNDEFINED
```

ORACLE

## The DEFINE and UNDEFINE Commands

Variables are defined until you either:

- Issue the UNDEFINE command on a variable
- Exit *iSQL\*Plus*

When you undefine variables, you can verify your changes with the DEFINE command. When you exit *iSQL\*Plus*, variables defined during that session are lost.

# Using the DEFINE Command with & Substitution Variable

- Create the substitution variable using the **DEFINE** command.

```
DEFINE employee_num = 200
```

- Use a variable prefixed with an ampersand (&) to substitute the value in the SQL statement.

```
SELECT employee_id, last_name, salary, department_id  
FROM   employees  
WHERE  employee_id = &employee_num ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
200	Whalen	4400	10

ORACLE

## Using the DEFINE Command

The example on the slide creates an *iSQL\*Plus* substitution variable for an employee number by using the **DEFINE** command, and at run time displays the employee number, name, salary, and department number for that employee.

Because the variable is created using the *iSQL\*Plus* **DEFINE** command, the user is not prompted to enter a value for the employee number. Instead, the defined variable value is automatically substituted in the **SELECT** statement.

The **EMPLOYEE\_NUM** substitution variable is present in the session until the user undefines it or exits the *iSQL\*Plus* session.

# Using the && Substitution Variable

Use the double-ampersand (&&) if you want to reuse the variable value without prompting the user each time.

```
SELECT    employee_id, last_name, job_id, &&column_name
FROM      employees
ORDER BY  &&column_name;
```

Define Substitution Variables

"column\_name" department\_id

Submit for Execution

Cancel

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

...

20 rows selected.

ORACLE

## Double-Ampersand Substitution Variable

You can use the double-ampersand (&&) substitution variable if you want to reuse the variable value without prompting the user each time. The user will see the prompt for the value only once. In the example on the slide, the user is asked to give the value for variable *column\_name* only once. The value supplied by the user (*department\_id*) is used both for display and ordering of data.

iSQL\*Plus stores the value supplied by using the DEFINE command; it will use it again whenever you reference the variable name. Once a user variable is in place, you need to use the UNDEFINE command to delete it.

# Using the VERIFY Command

Use the **VERIFY** command to toggle the display of the substitution variable, before and after *iSQL\*Plus* replaces substitution variables with values.

```
SET VERIFY ON
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num;
```

"employee\_num" 200

```
old   3: WHERE  employee_id = &employee_num
new   3: WHERE  employee_id = 200
```

ORACLE

## The VERIFY Command

To confirm the changes in the SQL statement, use the *iSQL\*Plus* **VERIFY** command. Setting **SET VERIFY ON** forces *iSQL\*Plus* to display the text of a command before and after it replaces substitution variables with values.

The example on the slide displays the old as well as the new value of the **EMPLOYEE\_ID** column.

# Customizing the *iSQL\*Plus* Environment

- Use **SET** commands to control current session.

```
SET system_variable value
```

- Verify what you have set by using the **SHOW** command.

```
SET ECHO ON
```

```
SHOW ECHO  
echo ON
```

ORACLE

## Customizing the *iSQL\*Plus* Environment

You can control the environment in which *iSQL\*Plus* is currently operating by using the **SET** commands.

### Syntax

```
SET system_variable value
```

In the syntax:

*system\_variable* is a variable that controls one aspect of the session environment

*value* is a value for the system variable

You can verify what you have set by using the **SHOW** command. The **SHOW** command on the slide checks whether **ECHO** had been set on or off.

To see all **SET** variable values, use the **SHOW ALL** command.

For more information, see *iSQL\*Plus User's Guide and Reference*, "Command Reference."

# SET Command Variables

- **ARRAYSIZE** {20 | *n*}
- **FEEDBACK** {6 | *n* | OFF | ON}
- **HEADING** {OFF | ON}
- **LONG** {80 | *n*} | ON | *text*}

```
SET HEADING OFF
```

```
SHOW HEADING  
HEADING OFF
```

ORACLE

## SET Command Variables

SET Variable and Values	Description
ARRAY[SIZE] { 20   <i>n</i> }	Sets the database data fetch size
FEED[BACK] { <u>6</u>   <i>n</i>   OFF   ON }	Displays the number of records returned by a query when the query selects at least <i>n</i> records
HEA[DING] { OFF   ON }	Determines whether column headings are displayed in reports
LONG { <u>80</u>   <i>n</i> }	Sets the maximum width for displaying LONG values

**Note:** The value *n* represents a numeric value. The underlined values indicate default values. If you enter no value with the variable, *iSQL\*Plus* assumes the default value.



# iSQL\*Plus Format Commands

- **COLUMN** [*column option*]
- **TTITLE** [*text* | OFF | ON]
- **BTITLE** [*text* | OFF | ON]
- **BREAK** [ON *report\_element*]

ORACLE

7-17

Copyright © Oracle Corporation, 2001. All rights reserved.

## Obtaining More Readable Reports

You can control the report features by using the following commands:

Command	Description
COL[UMN] [ <i>column option</i> ]	Controls column formats
TTI[TLE] [ <i>text</i>   OFF   ON]	Specifies a header to appear at the top of each page of the report
BTI[TLE] [ <i>text</i>   OFF   ON]	Specifies a footer to appear at the bottom of each page of the report
BRE[AK] [ON <i>report_element</i> ]	Suppresses duplicate values and divides rows of data into sections by using line breaks

### Guidelines

- All format commands remain in effect until the end of the iSQL\*Plus session or until the format setting is overwritten or cleared.
- Remember to reset your iSQL\*Plus settings to the default values after every report.
- There is no command for setting an iSQL\*Plus variable to its default value; you must know the specific value or log out and log in again.
- If you give an alias to your column, you must reference the alias name, not the column name.

# The COLUMN Command

## Controls display of a column:

```
COL[UMN] [{column|alias} [option]]
```

- **CLE[AR]: Clears any column formats**
- **HEA[DING] *text*: Sets the column heading**
- **FOR[MAT] *format*: Changes the display of the column using a format model**
- **NOPRINT | PRINT**
- **NULL**

ORACLE

## COLUMN Command Options

Option	Description
CLE[AR]	Clears any column formats
HEA[DING] <i>text</i>	Sets the column heading (a vertical line ( ) forces a line feed in the heading if you do not use justification.)
FOR[MAT] <i>format</i>	Changes the display of the column data
NOPRI[NT]	Hides the column
NUL[L] <i>text</i>	Specifies text to be displayed for null values
PRI[NT]	Shows the column

# Using the COLUMN Command

- Create column headings.

```
COLUMN last_name HEADING 'Employee|Name'  
COLUMN salary JUSTIFY LEFT FORMAT $99,990.00  
COLUMN manager FORMAT 999999999 NULL 'No manager'
```

- Display the current setting for the LAST\_NAME column.

```
COLUMN last_name
```

- Clear settings for the LAST\_NAME column.

```
COLUMN last_name CLEAR
```

ORACLE

## Displaying or Clearing Settings

To show or clear the current COLUMN command settings, use the following commands:

Command	Description
COL[UMN] <i>column</i>	Displays the current settings for the specified column
COL[UMN]	Displays the current settings for all columns
COL[UMN] <i>column</i> CLE[AR]	Clears the settings for the specified column
CLE[AR] COL[UMN]	Clears the settings for all columns

# COLUMN Format Models

Element	Description	Example	Result
9	Single zero-suppression digit	999999	1234
0	Enforces leading zero	099999	001234
\$	Floating dollar sign	\$9999	\$1234
L	Local currency	L9999	L1234
.	Position of decimal point	9999.99	1234.00
,	Thousand separator	9,999	1,234

ORACLE

## COLUMN Format Models

The slide displays sample COLUMN format models.

The Oracle server displays a string of pound signs (#) in place of a whole number whose digits exceed the number of digits provided in the format model. It also displays pound signs in place of a value whose format model is alphanumeric but whose actual value is numeric.

# Using the BREAK Command

**Use the BREAK command to suppress duplicates.**

```
BREAK ON job_id
```

ORACLE

## The BREAK Command

Use the BREAK command to divide rows into sections and suppress duplicate values. To ensure that the BREAK command works effectively, use the ORDER BY clause to order the columns that you are breaking on.

### Syntax

```
BREAK on column[|alias|row]
```

In the syntax:

column[ alias row]	suppresses the display of duplicate values for a given column
--------------------	---

Clear all BREAK settings by using the CLEAR command:

```
CLEAR BREAK
```

# Using the TTITLE and BTITLE Commands

- Display headers and footers.

```
TTI[TLE] [text|OFF|ON]
```

- Set the report header.

```
TTITLE 'Salary|Report'
```

- Set the report footer.

```
BTITLE 'Confidential'
```

ORACLE

## The TTITLE and BTITLE Commands

Use the TTITLE command to format page headers and the BTITLE command for footers. Footers appear at the bottom of the page.

The syntax for BTITLE and TTITLE is identical. Only the syntax for TTITLE is shown. You can use the vertical bar (|) to split the text of the title across several lines.

### Syntax

```
TTI[TLE] | BTI[TLE] [text|OFF|ON]
```

In the syntax:

*text* represents the title text (enter single quotes if the text is more than one word).

OFF | ON toggles the title either off or on. It is not visible when turned off.

The TTITLE example on the slide sets the report header to display Salary centered on one line and Report centered below it. The BTITLE example sets the report footer to display Confidential. TTITLE automatically puts the date and a page number on the report.

**Note:** The slide gives an abridged syntax for TTITLE and BTITLE. Various options for TTITLE and BTITLE are covered in another SQL course.

### **Instructor Note**

SQL\*Plus 3.3 introduced the commands REPHEADER and REPFOOTER. REPHEADER places and formats a specified report header at the top of each report or lists the current REPHEADER definition. REPFOOTER places and formats a specified report footer at the bottom of each report or lists the current REPFOOTER definition.

# Creating a Script File to Run a Report

1. **Create and test the SQL `SELECT` statement.**
2. **Save the `SELECT` statement into a script file.**
3. **Load the script file into an editor.**
4. **Add formatting commands before the `SELECT` statement.**
5. **Verify that the termination character follows the `SELECT` statement.**

ORACLE

## Creating a Script File to Run a Report

You can either enter each of the *iSQL\*Plus* commands at the SQL prompt or put all the commands, including the `SELECT` statement, in a command (or script) file. A typical script consists of at least one `SELECT` statement and several *iSQL\*Plus* commands.

### How to Create a Script File

1. Create the SQL `SELECT` statement at the SQL prompt. Ensure that the data required for the report is accurate before you save the statement to a file and apply formatting commands. Ensure that the relevant `ORDER BY` clause is included if you intend to use breaks.
2. Save the `SELECT` statement to a script file.
3. Edit the script file to enter the *iSQL\*Plus* commands.
4. Add the required formatting commands before the `SELECT` statement. Be certain not to place *iSQL\*Plus* commands within the `SELECT` statement.
5. Verify that the `SELECT` statement is followed by a run character, either a semicolon (`;`) or a slash (`/`).



# Creating a Script File to Run a Report

6. Clear formatting commands after the **SELECT** statement.
7. Save the script file.
8. Load the script file into the *iSQL\*Plus* text window, and click the Execute button.

ORACLE

## How to Create a Script File (continued)

6. Add the format-clearing *iSQL\*Plus* commands after the run character. Alternatively, you can store all the format-clearing commands in a reset file.
7. Save the script file with your changes.
8. Load the script file into the *iSQL\*Plus* text window, and click the Execute button.

## Guidelines

- You can include blank lines between *iSQL\*Plus* commands in a script.
- If you have a lengthy *iSQL\*Plus* or *SQL\*Plus* command, you can continue it on the next line by ending the current line with a hyphen (-).
- You can abbreviate *iSQL\*Plus* commands.
- Include reset commands at the end of the file to restore the original *iSQL\*Plus* environment.

**Note:** REM represents a remark or comment in *iSQL\*Plus*.

# Sample Report

Fri Sep 28

Employee  
Report

page 1

Job Category	Employee	Salary
AC_ACCOUNT	Gietz	\$8,300.00
AC_MGR	Higgins	\$12,000.00
AD_ASST	Whalen	\$4,400.00
IT_PROG	Ernst	\$6,000.00
	Hunold	\$9,000.00
	Lorentz	\$4,200.00
MK_MAN	Hartstein	\$13,000.00
MK_REP	Fay	\$6,000.00
SA_MAN	Zlotkey	\$10,500.00
SA_REP	Abel	\$11,000.00
	Grant	\$7,000.00
	Taylor	\$8,600.00

Confidential

...

ORACLE

7-26

Copyright © Oracle Corporation, 2001. All rights reserved.

## Example

Create a script file to create a report that displays the job ID, last name, and salary for every employee whose salary is less than \$15,000. Add a centered, two-line header that reads “Employee Report” and a centered footer that reads “Confidential.” Rename the job title column to read “Job Category” split over two lines. Rename the employee name column to read “Employee.” Rename the salary column to read “Salary” and format it as \$2,500.00.

```
SET FEEDBACK OFF
TTITLE 'Employee|Report'
BTITLE 'Confidential'
BREAK ON job_id
COLUMN job_id HEADING 'Job|Category'
COLUMN last_name HEADING 'Employee'
COLUMN salary HEADING 'Salary' FORMAT $99,999.99
REM ** Insert SELECT statement
SELECT job_id, last_name, salary
FROM employees
WHERE salary < 15000
ORDER BY job_id, last_name
/
REM clear all formatting commands ...
SET FEEDBACK ON
COLUMN job_id CLEAR
COLUMN last_name CLEAR
COLUMN salary CLEAR
CLEAR BREAK
...
```

# Summary

**In this lesson, you should have learned how to:**

- **Use *iSQL\**Plus substitution variables to store values temporarily**
- **Use `SET` commands to control the current *iSQL\**Plus environment**
- **Use the `COLUMN` command to control the display of a column**
- **Use the `BREAK` command to suppress duplicates and divide rows into sections**
- **Use the `TTITLE` and `BTITLE` commands to display headers and footers**

ORACLE

## Summary

In this lesson, you should have learned about substitution variables and how useful they are for running reports. They give you the flexibility to replace values in a `WHERE` clause, column names, and expressions. You can customize reports by writing script files with:

- Single ampersand substitution variables
- Double ampersand substitution variables
- The `DEFINE` command
- The `UNDEFINE` command
- Substitution variables in the command line

You can create a more readable report by using the following commands:

- `COLUMN`
- `TTITLE`
- `BTITLE`
- `BREAK`

# Practice 7 Overview

**This practice covers the following topics:**

- **Creating a query to display values using substitution variables**
- **Starting a command file containing variables**

ORACLE®

## Practice 7 Overview

This practice gives you the opportunity to create files that can be run interactively by using substitution variables to create run-time selection criteria.

## Practice 7

Determine whether the following two statements are true or false:

1. The following statement is valid:

```
DEFINE & p_val = 100
```

True/False

2. The DEFINE command is a SQL command.

True/False

3. Write a script to display the employee last name, job, and hire date for all employees who started between a given range. Concatenate the name and job together, separated by a space and comma, and label the column Employees. In a separate SQL script file, use the DEFINE command to provide the two ranges. Use the format MM/DD/YYYY. Save the script files as lab7\_3a.sql and lab7\_3b.sql.

```
DEFINE low_date = 01/01/1998
DEFINE high_date = 01/01/1999
```

- 4.

in a given location. The search condition should allow for case-insensitive searches of the department location. Save the script file as lab7\_4.sql.

EMPLOYEES	HIRE_DATE
Matos, ST_CLERK	15-MAR-98
Vargas, ST_CLERK	09-JUL-98
Taylor, SA_REP	24-MAR-98

EMPLOYEE NAME	JOB_ID	DEPARTMENT NAME
Whalen	AD_ASST	Administration
King	AD_PRES	Executive
Kochhar	AD_VP	Executive
De Haan	AD_VP	Executive
Higgins	AC_MGR	Accounting
Gietz	AC_ACCOUNT	Accounting

6 rows selected.

**Practice 7 (continued)**

5. Modify the code in `lab7_4.sql` to create a report containing the department name, employee last name, hire date, salary, and annual salary for each employee in a given location. Label the columns `DEPARTMENT NAME`, `EMPLOYEE NAME`, `START DATE`, `SALARY`, and `ANNUAL SALARY`, placing the labels on multiple lines. Resave the script as `lab7_5.sql`, and execute the commands in the script.

DEPARTMENT NAME	EMPLOYEE NAME	START DATE	SALARY	ANNUAL SALARY
Accounting	Higgins	07-JUN-94	\$12,000.00	\$144,000.00
	Gietz	07-JUN-94	\$8,300.00	\$99,600.00
Administration	Whalen	17-SEP-87	\$4,400.00	\$52,800.00
Executive	King	17-JUN-87	\$24,000.00	\$288,000.00
	Kochhar	21-SEP-89	\$17,000.00	\$204,000.00
	De Haan	13-JAN-93	\$17,000.00	\$204,000.00

