

PRODUCING READABLE OUTPUT WITH SQL*PLUS
-- CHAPTER 8 --

- Interactive Reports: Using SQL*Plus, you can create reports that prompt the user to supply their own values to restrict the range of data to be returned. To create interactive reports, 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.
- In SQL*Plus, you can use & (the ampersand) to temporarily store values. And you can use the DEFINE and ACCEPT commands to predefine variables. ACCEPT reads a line of user input and stores it in a variable.
- The following example shows the user of substitution variables...

```
SQL> select empno, ename, sal, deptno
2  from emp
3  where empno = &employee_num
4  /
Enter value for employee_num: 7369
old 3: where empno = &employee_num
new 3: where empno = 7369
```

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	1500	20

- Basically, we created a query in which we made a comparison to a variable, which SQL*Plus then inquired about its value, and once it was provided, it was able to continue with the query. In the following example, we accomplish the same query using all variable names...

```
SQL> select &EmployeeNumber, &EmployeeName, &Salary, &DepartmentNumber
2  from &TableName
3  where &EmployeeNumber = &EmployeeNumber1
4  /
Enter value for employeenum: empno
Enter value for employeename: ename
Enter value for salary: sal
Enter value for departmentnumber: deptno
old 1: select &EmployeeNumber, &EmployeeName, &Salary, &DepartmentNumber
new 1: select empno, ename, sal, deptno
Enter value for tablename: emp
old 2: from &TableName
new 2: from emp
Enter value for employeenum: empno
Enter value for employeenum1: 7369
old 3: where &EmployeeNumber = &EmployeeNumber1
new 3: where empno = 7369
```

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	1500	20

- Note that SQL*Plus will inquire about the value of each one of the variable names before it will be able to return any meaningful results.
- Using the SET VERIFY command: Toggling the display of the text before and after SQL*Plus replaces substitution variables with values. To change the display, simply issue the following command: set verify ON/OFF. When ON, it will show the old and new values of any variables used in the query.
- Character and Date Values with Substitution Variables.

```
SQL> select ename, deptno, sal*12
2  from emp
3  where job='&job_title'
4  /
Enter value for job_title: ANALYST
```

ENAME	DEPTNO	SAL*12
SMITH	20	18000
SCOTT	20	24000
ADAMS	20	18000
H	3	18000
"HELLO"	50	24000

- Note that in the above example, we need to either include single quotes around the variable name or if we want to omit them, we have to include them when SQL*Plus prompts for the character value, so instead of entering ANALIYST, we would enter 'ANALYST'.
- Note that in order for this query to find any records...ANALYST must be all in caps...in order to avoid this, you may enter [analyst] so as long as you use the UPPER() function as shown below...

```
SQL> select ename, deptno, sal*12
2  from emp
3  where job=upper('&job_title')
4  /
Enter value for job_title: analyst
```

ENAME	DEPTNO	SAL*12
SMITH	20	18000
SCOTT	20	24000
ADAMS	20	18000
H	3	18000
"HELLO"	50	24000

- LOWER() is also available for this purpose....
- Note that substitution variables can pretty much take the place of anything including WHERE, FROM, ORDER BY, Column expressions, table-Names, basically, the entire SELECT statement...To show this, take a look at the following example:

```

SQL> select &first, &second
       2  &fromClause &tableName
       3  &whereClause sal &theOperator 3000;
Enter value for first: ename
Enter value for second: sal
Enter value for fromclause: from
Enter value for tablename: emp
Enter value for whereclause: where
Enter value for theoperator: >

```

```

ENAME          SAL
-----
MOLITOR        5000
SHANK          5500
ALBERT         4000

```

- Now, let's take a look at the && Substitution Variable: Use the double-ampersand (&&) if you want to reuse the variable value without prompting the user each time. The user will get a single prompt regardless of how many times the variable name appears in the SELECT statement. Interesting to note is that once defined, the variable name retains its value across select statement sessions.
- Defining User Variables: You can predefine variables using one of two SQL*Plus commands:
 - DEFINE: Create a CHAR datatype user variable
 - ACCEPT: Read user input and store it into a variable;
- If you need to predefine a variable that includes empty spaces, make sure you use quotation marks when using the DEFINE statement.
 - DEFINE variable = value
 - Creates a CHAR datatype user variable and assigns a value to it.
 - DEFINE variable
 - Displays the variable, its value, and its datatype
 - DEFINE
 - Displays ALL user variables with value and datatype
 - ACCEPT
 - Reads a line of user input and stores it in a variable
- Note that when defining variables, you do not need to include the & symbol. In fact, if you do, it does not work! Also, the string of characters you define as, does not need quotes, unless it needs to have empty spaces embedded in between.
- The ACCEPT command:
 - Creates a customized prompt when accessing user input
 - Explicitly defines a NUMBER or DATE datatype variable
 - Hides user input for security reasons
- **SYNTAX:**

```
ACCEPT variable [datatype] [FORMAT format] [PROMPT text] [HIDE]
```

- IN the syntax, variable is the name of the variable that stores the value (if it does not exist, SQL*Plus will create one). Datatype is NUMBER, CHAR, or DATE (CHAR has a maximum length of 240 bytes).

DATE checks against a format model, and the datatype is CHAR). FOR[MAT] format specifies the format model – for example, A10 or 9.999. PROMPT text displays the text before the user can enter the value. HIDE suppresses what the user enters – for example, a password.

- Here's an example on how to use the ACCEPT command...

```
SQL> accept dept prompt 'Provide the department name: '  
Provide the department name: Sales  
SQL> select *  
2 from dept  
3 where dname = upper('&dept');  
old 3: where dname = upper('&dept')  
new 3: where dname = upper('Sales')
```

DEPTNO	DNAME	LOC
30	SALES	CHICAGO

- Now look at Customizing the SQL*Plus Environment:
 - Use the SET commands to control the current session.
 - SET system_variable value
 - Use the SHOW command to verify what you SET...

```
SQL> set echo on  
SQL> show echo  
echo ON
```

- To see all set variables, use the SHOW ALL command.

```
SQL> SHOW ALL  
appinfo is ON and set to "SQL*Plus"  
arraysize 15  
autocommit OFF  
autoprint OFF  
autotrace OFF  
shiftinout INVISIBLE  
blockterminator "." (hex 2e)  
btitle OFF and is the 1st few characters of the next SELECT statement  
cmdsep OFF  
colsep " "  
compatibility version NATIVE  
concat "." (hex 2e)  
copycommit 0  
COPYTYPECHECK is ON  
define "&" (hex 26)  
echo ON  
editfile "afiedt.buf"  
embedded OFF  
escape OFF  
FEEDBACK ON for 6 or more rows  
flagger OFF  
flush ON  
heading ON  
headsep "|" (hex 7c)  
linesize 100  
lno 5  
loboffset 1  
long 80
```

```

longchunksize 80
newpage 1
null ""
numformat ""
numwidth 9
pagesize 24
PAUSE is OFF
pno 1
recsep WRAP
recsepchar " " (hex 20)
release 801050000
repfooter OFF and is NULL
repheader OFF and is NULL
serveroutput OFF
showmode OFF
spool OFF
sqlcase MIXED
sqlcode 904
sqlcontinue "> "
sqlnumber ON
sqlprefix "#" (hex 23)
sqlprompt "SQL> "
sqlterminator ";" (hex 3b)
suffix "sql"
tab ON
termout ON
time OFF
timing OFF
trimout ON
trimsPOOL OFF
ttitle OFF and is the 1st few characters of the next SELECT statement
underline "-" (hex 2d)
USER is "SCOTT"
verify ON
wrap : lines will be wrapped

```

- **Set Command Variables:**

- `ARRAYSIZE { 20 | N }`
 - Sets database to data fetch size
- `COLSEP { _ | text }`
 - Sets text to be printed between columns (default is single space)
- `FEEDBACK { 6 | n | OFF | ON }`
 - Displays the number of records returned by a query when the query selects at least n records.
- `HEADING { OFF | ON }`
 - Determines whether column headings are displayed in reports or not.
- `LINESIZE { 80 | n }`
 - Sets the number of characters to n for reports
- `LONG { 80 | n }`
 - Sets the maximum length for displaying LONG values.
- `PAGESIZE { 24 | n }`
 - Determines the number of lines per page of output.
- `PAUSE { OFF | ON | text }`
 - Allows you to control scrolling of your terminal (You must press [RETURN] after seeing each pause).
- `TERMOUT { OFF | ON }`
 - Determines whether output is displayed on screen.

- **IMPORTANT: SAVING CUSTOMIZATIONS** in the **login.sql** File:

- The login.sql file contains standard SET and other SQL*Plus commands that are implemented at login
- You can modify the login.sql to contain additional SET commands.
- The login.sql file is located at **C:\orant\DBS\login.sql** in this computer. If this is not the location in your system, just have the operating system look for it for you...do a search!
- SQL*Plus Format Commands and their corresponding Options...
 - **COLUMN** [column option]
 - Controls column formats.
 - CLEAR: Clears any column formats
 - FOR[MAT] format: Changes the display of the column data
 - HEA[DING]: text: Sets the column heading (a vertical line (|) will force a line feed in the heading if you do not use justification).
 - JUS[TIFY]{ALIGN}: Aligns the column heading to be left, center, or right.
 - NOPRI[NT]: Hides the column.
 - NUL[L]: Specifies text to be displayed for null values.
 - PRI[NT]: Shows the column.
 - TRU[NCATED]: Truncates the string at the end of the first line of display.
 - WRA[PPED]: Wraps the end of the string to the next line.
 - **TTITLE** [text|OFF|ON]
 - Specifies a header to appear at the top of each page.
 - **BTITLE**[text|OFF|ON]
 - Specifies a footer to appear at the bottom of each page of the report
 - **BREAK**[ON report_element]
 - Suppress duplicate values and sections rows of data with line feeds.
- Note that all formatting settings remain in effect until the end of the setting.
- Using the TTITLE and BTITLE commands: Display headers and footers...
 - TTI[TLE] [text|OFF|ON]
 - TTITLE 'Salary|Report' -- Headers
 - BTITLE 'Confidential' -- Footers
- The following example demonstrates the use of setting pagesize, tttitle, btitle, and break.

```
SQL> set pagesize 25
SQL> tttitle 'Salary Report|September, 2002'
SQL> btitle 'Salary Report'
SQL> break on job skip 1
SQL> r
 1 select empno, ename, job, mgr, sal, deptno
 2 from emp
 3 where sal between &Low and &High
 4* order by job
Enter value for low: 1000
Enter value for high: 2000
old 3: where sal between &Low and &High
new 3: where sal between 1000 and 2000

Fri Sep 06                               page 1

                Salary Report
                September, 2002

EMPNO ENAME      JOB              MGR      SAL      DEPTNO
-----
7788 SCOTT       ANALYST         7566     2000     20
```

```

8001 "HELLO"                2000      50
8060 FREY                   2000      50
9200 KWIATKI                2000      50
9100 DUTT                    2000      50
8090 GORDON                 2000      50
9600 BUSH                    2000      50
1010 HOMER                  2000      50
9230 GREEN                  2000      50
9712 TESTTIME              2000      50
9310 NKLGNFGLKB            2000      50
9326 HOLLYO                 2000      50
9300 REDCORNER             2000      50
9120 RED                    2000      50
1000                       2000      50

7566 JONES      MANAGER      7839      2000      20

```

Salary Report

Fri Sep 06

page 2

**Salary Report
September, 2002**

```

EMPNO  ENAME      JOB              MGR      SAL      DEPTNO
-----  -
7698  BLAKE      MANAGER          7839     2000     30
7782  CLARK      MANAGER          7839     2000     10

7839  KING      PRESIDENT                3000     10

321   BOBBY     SALESMAN                3000     50

```

Salary Report

20 rows selected.

- Notice that each page is 25 line long and that there are line breaks (of 1 in this case) between records that have a job-transition (i.e., from MANAGER to PRESIDENT).
- Creating a Script File to Run a Report
 - Create the SQL SELECT statement
 - Save the SELECT statement to a script file
 - Load the script file into the editor
 - Add formatting commands before the SELECT statements
 - Verify that the termination character follows the SELECT statement.
- How to create a Script File
 1. Create the SQL SELECT statements at the SQL prompt. Ensure that the data required for the report is accurate before you save the statements to a file and apply formatting commands. Endure 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 SQL*Plus commands
 4. Add the required formatting commands before the SELECT statement. Be certain not to place SQL*Plus commands within the SELECT statement.
 5. Verify that the SELECT statement is followed by a run character, either a semicolon(;) or a slash (/).

6. Clear formatting commands after the SELECT statement.
7. Save the script file.
8. Enter "START filename" to run the script.

sampleReport.sql

```

SET PAGESIZE 37
SET LINESIZE60
SET FEEDBACK OFF
TTITLE 'Employee Report'
BTITLE 'Confidential'
BREAK ON job
COLUMN job HEADING 'Job|Category' FORMAT A15
COLUMN ename HEADING 'Employee' FORMAT A15
COLUMN sal HEADING 'Salary', FORMAT $99.999.99
REM ** Insert SELECT statement
SELECT job, ename, sal
FROM emp
WHERE sal < 3000
ORDER BY job, ename
/
SET FEEDBACK ON
REM clear all formatting commands..
/

```

* The following is a sample report generated from the script above...

Fri Sep 06

page 1

Employee Report

Job Category	Employee	SAL

ANALYST	"HELLO"	2000
	ADAMS	1500
	BUSH	2000
	DUTT	2000
	FREY	2000
	GORDON	2000
	GREEN	2000
	H	1500
...		