Chapter 3
SQL*Plus

Objectives
- Use SQL*Plus
- Produce queries that require substitution variables
- Customize the environment
- Produce more readable output
- Create and execute script files

Overview of SQL*Plus
- Log into SQL*Plus
- Describe a table structure
- Edit a SQL command
- Execute a SQL command
- Save SQL statements to files and append SQL statements to files
- Execute saved files
- Load commands from file to buffer to edit

Logging into SQL*Plus
- Look for the SQL Plus icon on the lab desktop
- Your user-name is the first 12 letters of your last name (spaces removed) and then add @caleb to the end as in: smith@caleb
- User-name is case-insensitive
- Your password is the last 6 digits of your student number.

Displaying Table Structure
- Use the SQL*Plus DESCRIBE command to display the structure of a table.

SQL*Plus
- Can edit the buffer
  - (e.g. A[PPEND] *text,* C[HANGE] / *old* / *new,* C[HANGE] / *text* /, CL[EAR] BUFF[ER], DEL)
- Work with files
  - SAVE *filename*
  - GET *filename*
  - START *filename*
  - @ *filename*
  - EDIT *filename*
  - SPOOL *filename*
  - EXIT

  For your labs it is easier to use a text editor such as Notepad to enter your SQL and SQL*Plus commands. Then copy and paste. More about this later!

SQL*Plus Format Commands

- COLUMN [*column option*]
- TTITLE [*text* | OFF | ON]
- BTITLE [*text* | OFF | ON]
- BREAK [ON *report_element*]

The COLUMN Command

- Controls display of a column

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

- CLE[AR]: Clears any column formats
- FOR[MAT] *format*: Changes the display of the column using a format model
- HEA[DING] *text*: Sets the column heading
- JUS[TIFY] {*align*}: Aligns the column heading to be left, center, or right

Using the COLUMN Command

- Create column headings

```
COLUMN ename HEADING 'Employee|Name' FORMAT A15
COLUMN sal JUSTIFY LEFT FORMAT $99,990.00
COLUMN mgr FORMAT 999999999 NULL 'No manager'
```

- Display the current setting for the ENAME column.

```
COLUMN ename
```

- Clear settings for the ENAME column.

```
COLUMN ename CLEAR
```

COLUMN Format Models

Element	*Description*	*Example*	*Result*
A*n*	Sets a display width of *n*	N/A	N/A
9	Single zero-suppression digit	999999	1234
0	Enforces leading zero	099999	01234
$	Floating dollar sign	$9999	$1234
L	Local currency	L9999	L1234
.	Position of decimal point	9999.99	1234.00
,	Thousand separator	9,999	1,234

Column example

```
SQL> COLUMN ename HEADING 'EMPLOYEE|NAME' FORMAT A15
```

Using the TTITLE and BTITLE Commands

- Display headers and footers
  TTI[TLE] [*text*|OFF|ON]

- Set report header
  SQL> TTITLE 'Salary|Report'

- Set report footer
  SQL> BTITLE 'Confidential'

Using the BREAK Command

- Suppresses duplicates and sections rows
  - To suppress duplicates

```
SQL> BREAK ON ename ON job
```

- To section out rows at break values

```
SQL> BREAK ON ename SKIP 4 ON job SKIP2
```

Customizing the SQL*Plus Environment

- Use SET commands to control current session.

```
SET system_variable value
```

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

```
SQL> SET ECHO ON
SQL> SHOW ECHO
echo ON
```

SET Command Variables

- ARRAYSIZE {20 | *n*}
- COLSEP {_ | *text*}
- FEEDBACK {6 | *n* |OFF | ON}
- HEADING {OFF | ON}
- LINESIZE {80 | *n*}
- LONG {80 | *n*}
- PAGESIZE {24 | *n*}
- PAUSE {OFF | ON | *text*}
- TERMOUT {OFF | ON}

Pagesize and Linesize

- To set the size of a "page", use:

```
SET PAGESIZE n
```

- To set the size of a line, use:

```
SET LINESIZE n
```

Substitution Variables

- Use SQL*Plus substitution variables to temporarily store values.
  - Single ampersand (&)
  - Double ampersand (&&)
  - DEFINE and ACCEPT commands
- Pass variable values between SQL statements.
- Dynamically alter headers and footers.

Using the & Substitution Variable

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

```
SQL> SELECT   empno, ename, sal, deptno
  2  FROM     emp
  3  WHERE    empno = &employee_num;

Enter value for employee_num: 7369

    EMPNO ENAME            SAL    DEPTNO
--------- ---------- --------- ---------
     7369 SMITH            800        20
```

Using the SET VERIFY Command

- Toggling the display of the text of a command before and after SQL*Plus replaces substitution variables with values.

```
SQL> SET VERIFY ON
SQL> SELECT   empno, ename, sal, deptno
  2  FROM     emp
  3  WHERE    empno = &employee_num;

Enter value for employee_num: 7369
old   3: WHERE empno = &employee_num
new   3: WHERE empno = 7369
```

Character and Date Values with Substitution Variables

•Use single quotation marks for date and character values.

```
SQL> SELECT ename, deptno, sal*12
  2  FROM   emp
  3  WHERE  job='&job_title';

Enter value for job_title: ANALYST
ENAME          DEPTNO    SAL*12
---------- --------- ---------
     SCOTT         20     36000
      FORD         20     36000
```

Specifying Column Names, Expressions, and Text at Runtime

– Use substitution variables to supplement the following:
  - WHERE condition
  - ORDER BY clause
  - Column expression
  - Table name
  - Entire SELECT statement

```
SQL> SELECT      empno, ename, job, &column_name
  2  FROM        emp
  3  WHERE       &condition
  4  ORDER BY    &order_column;

Enter value for column_name: sal
Enter value for condition: sal>=3000
Enter value for order_column: ename

    EMPNO ENAME      JOB              SAL
--------- ---------- --------- ---------
     7902 FORD       ANALYST         3000
     7839 KING       PRESIDENT       5000
     7788 SCOTT      ANALYST         3000
```

Using the && Substitution Variable

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

```
SQL> SELECT    empno, ename, job, &&column_name
2  FROM        emp
3  ORDER BY    &column_name;

Enter value for column_name: deptno

    EMPNO ENAME      JOB           DEPTNO
--------- ---------- --------- ---------
     7839 KING       PRESIDENT         10
     7782 CLARK      MANAGER           10
     7934 MILLER     CLERK             10
...
14 rows selected.
```

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 in a variable
- If you need to predefine a variable that includes spaces, you must enclose the value within single quotation marks when using the DEFINE command.

The ACCEPT Command

- Creates a customized prompt when accepting user input
- Explicitly defines a NUMBER or DATE datatype variable
- Hides user input for security reasons

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

ACCEPT         dept PROMPT 'Provide the department name: '
SELECT         *
          FROM      dept
WHERE          dname = UPPER('&dept')
```

```
Provide the department name: Sales
   DEPTNO DNAME          LOC
--------- -------------- -------------
       30 SALES          CHICAGO
```

DEFINE and UNDEFINE Commands

- A variable remains defined until you either:
  - Use the UNDEFINE command to clear it
  - Exit SQL*Plus
- You can verify your changes with the DEFINE command.
- To define variables for every session, modify your *login.sql* file so that the variables are created at startup.

Using the DEFINE Command

- Create a variable to hold the department name.

```
SQL> DEFINE deptname = compsci
SQL> DEFINE deptname

DEFINE DEPTNAME        = "compsci" (CHAR)
```

- Use the variable as you would any other variable.

```
SQL> SELECT *
  2  FROM   dept
  3  WHERE  dname = UPPER('&deptname');
  4
```

The UNDEFINE Command

– Deletes one or more user variables that you defined explicitly with the DEFINE command

```
SQL> UNDEFINE deptname
```

Creating a Script File to Run a Report

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

Summary

- Used SQL*Plus
- Produced queries that required substitution variables
- Customized the environment
- Produced more readable output
- Created and executed script files
-

  Oracle documentation link to SQL*Plus:

  https://docs.oracle.com/cd/B10501_01/server.920/a90842/ch13.htm#1011230