Practice

Using SQL*Plus

Practice Target

In this practice, you will get familiar with using SQL*Plus utility.

Practice Assumption

This practice assumes that `winsrv` is up and running.

Note: All the demonstration steps in this practice apply in Linux environments as well.

Exploring SQL*Plus Basic Capabilities

SQL*Plus is probably the most common utility used by DBAs. That is why it is important for a DBA to be familiar with using it. In the following steps, you will explore the basic SQL*Plus capabilities.

First, you will learn about the different ways of invoking SQ*Plus and logging in to the database using the operating system authentication.

1. In `winsrv`, open a command prompt Window as **oracle**.

2. Issue the following command to invoke SQL*Plus without connecting to any database.

```
sqlplus /nolog
```

3. In SQL*Plus commad prompt, issue the following command to connect to the local database as sysdba.

Observe that we managed to connect to the database as `SYS` without providing the password. This method of connecting to the database is called Operating System Authentication method and you will learn about it later in the course. For now, you just need to understand that the connection attempt succeeds because the logged on OS user (`oracle`) is a member of the `ORA_DBA` group in Windows and `dba` group in Linux.

In SQL*Plus, `CONN` is an abbreviation of the `CONNECT`. You can use either of them. Nearly all the SQL*Plus commands have abbreviation forms.

If there is more than one database running in the system, the command connects to the database instance pointed by the variable `ORACLE_SID`

```
conn / as sysdba
```

4. Exit from SQL*Plus.

`QUIT` is also an acceptable command to exit from SQL*Plus.

```
exit
```

5. Issue the following command to invoke SQL*Plus and login to the database as `sysdba` in the same time.

```
sqlplus / as sysdba
```

6. In SQL*Plus command prompt, issue the following command to display the current user.

When you login as `sysdba` to the database, the current user is always considered as `SYS`. This user is the *superuser* for Oracle database. It has the full privileges on running almost any command on the database.

```
show user
```

7. Exit from SQL*Plus.

```
exit
```

8. Issue the following command to login to the database as `SYSTEM`. We provided the system password when you created the database. The code assumed the password is ABcd##1234

This command invokes SQL*Plus and login to the database as `SYSTEM` user.

The username is not case-sensitive. The password is.

```
sqlplus system/ABcd##1234

SQL> show user
```

9. Issue the following command to login as `HR` to the database.

```
conn hr/ABcd##1234

SQL> show user
```

You learnt how to invoke SQL*Plus and login to the local database. Later in the course, you will learn about logging on to Oracle databases via the Listener.

Let's now learn some common SQL*Plus commands.

Note: SQL*Plus commands are not the same as the SQL statements. SQL statements are standard statements that are sent to the database to retrieve some data, manipulate some data, or control a database feature or functionality. SQL*Plus commands are SQL*Plus-specific and they do not affect the data in the database.

10. Issue the following command to display the structure of `EMPLOYEES` table.

Observe the following about the commands.

- `DESCRIBE` is not a SQL statement. It is a SQL*Plus command.
- The short form of the `DESCRIBE` is `DESC`.
- The command is case-insensitive

```
DESCRIBE EMPLOYEES

desc employees
```

11. Invoke the following query to retrieve a sample data from `EMPLOYEES` table.

The latest SQL statement (not SQL*Plus command) you issue in SQL*Plus is saved in the SQL*Plus buffer. Observe that the statement ends with a semicolon.

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM EMPLOYEES ORDER BY 1;
```

12. Issue the following commands to list the contents of the SQL*Plus buffer.

Observe the following about the commands.

- `LIST` is used to retrieve the SQL statement saved in the SQL*Plus buffer
- The short form of the `LIST` is `L`.

```
list
l
```

13. Issue the following command to edit the statement in the buffer with the Notepad.

Observe that the semicolon is replaced by the forward slash. This is the expected behavior.

We can control which editor to open by setting the variable `EDITOR`. This is true in Windows and Linux operating system. In `srv1`, we can set the `EDITOR` to `vi`.

The abbreviation form of `EDIT` command is `ED`

```
edit
```

14. Modify the code to the following in the Notepad. Then close Notepad and save the changes.

The statement is changed in the buffer.

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM EMPLOYEES
WHERE EMPLOYEE_ID=100
ORDER BY 1
```

15. Verify the changes were applied on the SQL*Plus buffer.

```
l
```

16. Run the statement in the buffer.

To run the statement in the buffer, you can simply type / (forward slash) and press [Enter]. Alternatively, you can use `run` command, or simply type `r`.

```
/
```

17. Issue the following commands to control the width of `FIRST_NAME` and `SALARY` columns.

`COLUMN` is a SQL*Plus command that is used to control the format of the displayed column values. In the code below, we set the maximum characters of the `FIRST_NAME` to 10 characters and add a thousand separator in the `SALARY` column.

```
column FIRST_NAME format a10
column SALARY format 999,999
-- or issue the following:
col FIRST_NAME for a10
col SALARY for 999,999
```

18. Run the statement in the buffer again.

Observe the impact of the `FORMAT` command.

```
/
```

19. Issue the following commands.

Because the size of the `FORMAT` value (3 digits) is less than the number of the digits in the column value, the column value is displayed as `#` symbols.

```
column SALARY format 999
/
-- return the format of salary back
column SALARY format 999,999
```

You learnt some command editing and formatting commands. Let's learn on some SQL*Plus file commands. You will begin with saving and running SQL script files.

20. Save the contents of the buffer contents into a file named as list-emps.

Observe the command automatically added the sql extension.

`SAV` is the shortcut for the `SAVE` command.

```
save list-emps
```

21. List the contents of the current folder without exiting from SQL*Plus.

Use the SQL*Plus command `HOST` to submit a command to the operating system.

Note: If we were in Linux, we would issue the command `host ls` instead

```
host dir *
```

22. List the file we saved in SQL*Plus.

```
host dir list-emps*
```

23. Issue the following command to run the script

The files to be run from within SQL*Plus command prompt are called SQL script files. Observe that we do not need to include the extension in the file name.

The '@' is equivalent to the `START` command.

SQL*Plus reads the SQL script files and executes all the statements and commands in it sequentially. If one of the statements fails, it goes to the next statement and execute it.

```
@list-emps
start list-emps
```

We saw how to save the SQL statement in the buffer into external files. But sometimes we need to save the output of a SQL statement in external text files. To achieve this task, we use the SQL*Plus command `SPOOL`. In the following steps, you will learn about using this command.

24. Enter the following statement in the SQL*Plus command prompt. Press [Enter] twice after entering the code.

Observe the statement does not execute because it does not end with the semicolon symbol ';'. However, it is saved in the buffer.

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM EMPLOYEES
ORDER BY 1
```

25. Issue the following command.

This command instructs the SQL*Plus to save its output to the given file.

```
spool query-output.log
```

26. Run the statement in the buffer.

```
/
```

27. Issue the following command

The SQ*Plus output is not saved into the external file until this command is issued.

```
spool off
```

28. Display the contents of the query-output.log

To get rid of the headers, you can issue the command SET HEADING OFF

```
host notepad query-output.log
```

29. Exit from SQL*Plus.

```
exit
```

30. Open the `list-emps.sql` script in Notepad and add the `QUIT` command to the end of the file.

```
notepad list-emps.sql
```

```
SELECT EMPLOYEE_ID,  FIRST_NAME, LAST_NAME, SALARY FROM EMPLOYEES
WHERE EMPLOYEE_ID=100
ORDER BY 1
/

quit
```

31. Invoke SQL*Plus, connect to the database as `HR`, and run the `list-emps` script in one command.

SQL*Plus starts, logs on to the database as `HR`, executes the script, and quits.

```
sqlplus hr/ABcd##1234 @list-emps
```

32. Invoke SQL*Plus and connect to the database as `SYS`

```
sqlplus / as sysdba
```

33. Run the following `SET` command to change the prompt characters for the SQL*Plus.

We can set the prompt to a static string.

```
set sqlprompt "_user'@'_connect_identifier'> '"
```

34. Exit from SQL*Plus.

```
exit
```

35. Shutdown the vm.

Summary

- In this lecture, you learnt how to invoke SQL*Plus and login to the local running database instance.
- You learnt the following SQL*Plus commands:

Command	Abbreviation Form	Description
CONNECT	**CONN**	Connect to a local or remote database
DESCRIBE	**DESC**	Display the structure of a table or view
LIST	**L**	Display the contents of the SQL*Plus buffer
EDIT		Open the editor with the current buffer contents for editing
RUN	**R** **/**	Run the statement in the SQL*Plus buffer
SAVE		Save the contents of the SQL*Plus buffer in external file (called SQL script file)
HOST		Execute an OS command
START	**@**	Read the contents of external SQL script file and execute them
SPOOL <file name> **...** **SPOOL OFF**		Save the output the statements and commands executed in SQL*Plus between the SPOOL and SPOOL OFF commands into external file
set sqlprompt		Set the SQL*Plus prompt characters