

1.

Note: If you have executed the code examples for this lesson, make sure that you execute the following code before starting this practice:

```
DROP table employees2;  
DROP table copy_emp;
```

In this practice, you use PL/SQL code to interact with the Oracle Server.

1. Create a PL/SQL block that selects the maximum department ID in the `departments` table and stores it in the `v_max_deptno` variable. Display the maximum department ID.
 - a. Declare a variable `v_max_deptno` of type `NUMBER` in the declarative section.
 - b. Start the executable section with the `BEGIN` keyword and include a `SELECT` statement to retrieve the maximum `department_id` from the `departments` table.
 - c. Display `v_max_deptno` and end the executable block.
 - d. Execute and save your script as `lab_05_01_soln.sql`. The sample output is as follows:

```
PL/SQL procedure successfully completed.
```

```
The maximum department_id is : 270
```

1-

The screenshot displays the Oracle SQL Developer environment. The main window is titled 'Worksheet' and contains a PL/SQL script. The script declares a variable `v_deptnomax` of type `NUMBER`, starts with `BEGIN`, and includes a `SELECT` statement to find the maximum `department_id` from the `hr.departments` table, storing it in `v_deptnomax`. It then uses `dbms_output.put_line` to display the result and ends with `END;`. The status bar at the top indicates the execution time is 0.039 seconds.

Below the script editor, there are two panes. The 'Script Output' pane on the left shows the message 'PL/SQL procedure successfully completed.' The 'Query Result' pane on the right shows the output of the `SELECT` statement: 'THE MAXIMUM DEPARTMENT_ID IS : 270'.

The bottom status bar shows 'Line 6 Column 5 | Insert | Modified | Windows: CR/AF'.

2.

Modify the PL/SQL block that you created in step 1 to insert a new department into the `departments` table.

- Load the `lab_05_01_soln.sql` script. Declare two variables:
`v_dept_name` of type `departments.department_name` and
`v_dept_id` of type `NUMBER`.
Assign 'Education' to `v_dept_name` in the declarative section.
- You have already retrieved the current maximum department number from the `departments` table. Add 10 to it and assign the result to `v_dept_id`.
- Include an `INSERT` statement to insert data into the `department_name`, `department_id`, and `location_id` columns of the `departments` table.
Use the values in `v_dept_name` and `v_dept_id` for `department_name` and `department_id`, respectively, and use `NULL` for `location_id`.
- Use the SQL attribute `SQL%ROWCOUNT` to display the number of rows that are affected.
- Execute a `SELECT` statement to check whether the new department is inserted. You can terminate the PL/SQL block with `/` and include the `SELECT` statement in your script.
- Execute and save your script as `lab_05_02_soln.sql`. The sample output is as follows:

```
PL/SQL procedure successfully completed.

The maximum department_id is : 270
SQL%ROWCOUNT gives 1
```

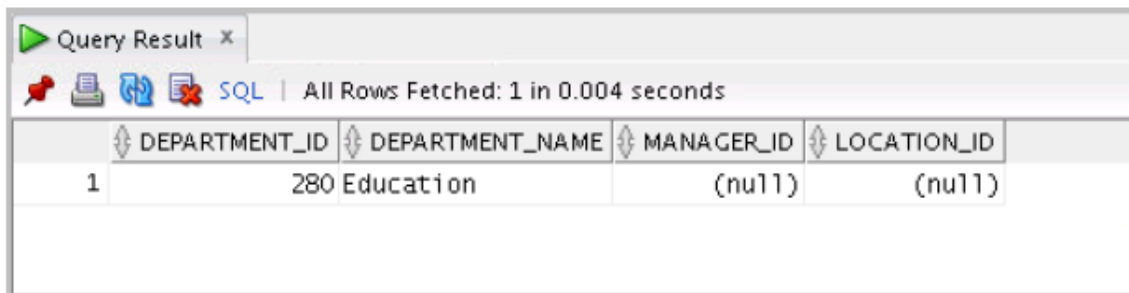
2-

The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL script in the 'Script Editor' tab. The script declares two variables, `v_deptname` and `v_deptid`, and performs an `INSERT` operation into the `departments` table. It also uses `SQL%ROWCOUNT` to display the number of rows affected. The script is terminated with a forward slash (`/`).

The 'Script Output' window at the bottom shows the execution results. It displays the message 'PL/SQL procedure successfully completed.' followed by the output of the `SELECT` statement: 'THE MAXIMUM DEPARTMENT_ID IS : 270' and '1 ROWS THAT ARE AFFECTED.'

The 'Query Result' window at the bottom shows the results of the `SELECT` statement. It displays a single row with the value '270' in the column 'MAX(DEPARTMENT_ID)'.

3.



Query Result x

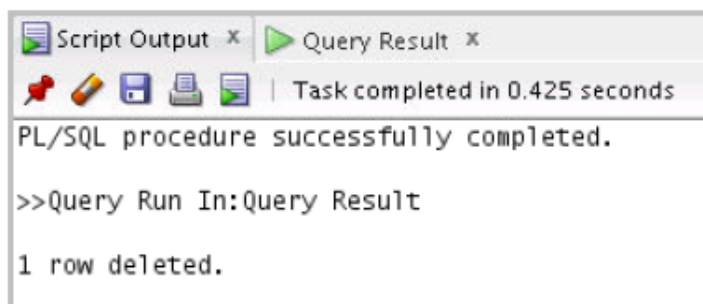
SQL | All Rows Fetched: 1 in 0.004 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	280	Education	(null)	(null)

In step 2, you set `location_id` to `NULL`. Create a PL/SQL block that updates `location_id` to 3000 for the new department.

Note: If you successfully completed step 2, continue with step 3a. If not, first execute the solution script `/soln/sol_05.sql`. (Task 2 in `sol_05.sql`)

- Start the executable block with the `BEGIN` keyword. Include the `UPDATE` statement to set `location_id` to 3000 for the new department (`v_dept_id = 280`).
- End the executable block with the `END` keyword. Terminate the PL/SQL block with `;/` and include a `SELECT` statement to display the department that you updated.
- Include a `DELETE` statement to delete the department that you added.
- Execute and save your script as `lab_05_03_soln.sql`. The sample output is as follows:



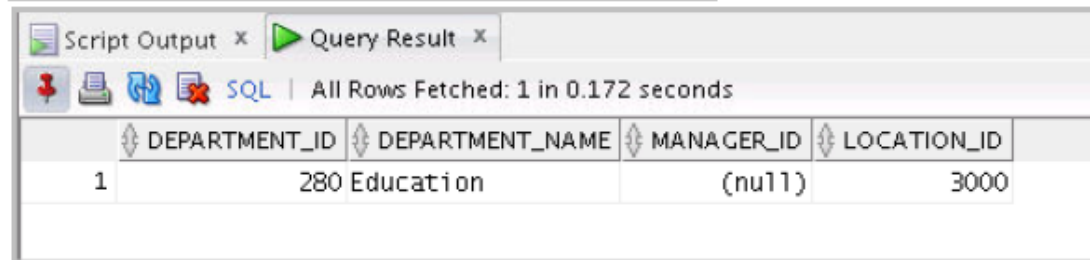
Script Output x Query Result x

Task completed in 0.425 seconds

PL/SQL procedure successfully completed.

>>Query Run In:Query Result

1 row deleted.



Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.172 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	280	Education	(null)	3000

3-

The screenshot displays the Oracle SQL Developer environment. The main window is titled 'Worksheet' and contains a PL/SQL script. The script declares a variable `v_deptname` and performs several operations on the `hr.departments` table: inserting a new row, updating the `location_id` of the department with the maximum `department_id`, and deleting the department with the maximum `department_id`.

```
DECLARE
v_deptname hr.departments.department_id%TYPE;
BEGIN
SELECT MAX(department_id) INTO v_deptname FROM hr.departments;
UPDATE hr.departments SET location_id=3000 WHERE department_id=v_deptname;
END;

/

SELECT * FROM hr.departments WHERE
department_id=(SELECT MAX(department_id) FROM hr.departments);

DELETE FROM hr.departments WHERE
department_id=(SELECT MAX(department_id) FROM hr.departments);
```

Below the script, the 'Script Output' pane shows the execution status: 'PL/SQL procedure successfully completed.' and '1 row deleted.'.

The 'Query Result' pane shows the result of a query executed after the script. The query is `SELECT * FROM hr.departments`, and the result is a single row with the following values:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	270 Payroll	(null)	3000

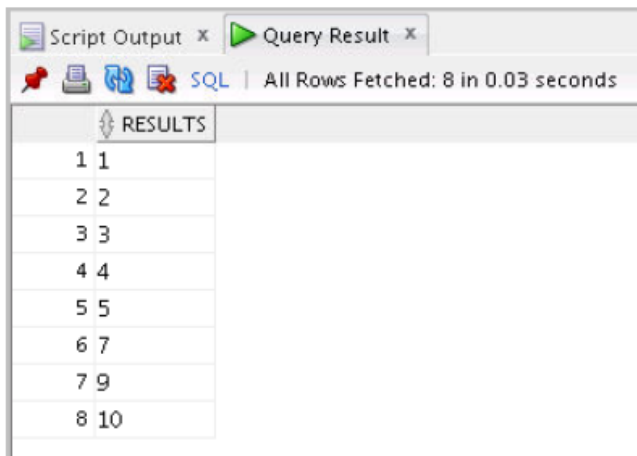
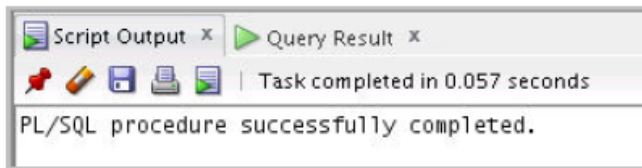
The status bar at the bottom indicates 'Line 12 Column 63', 'Insert', and 'Modified: Windows: OUP'.

4.

In this practice, you create PL/SQL blocks that incorporate loops and conditional control structures. This practice tests your understanding of various `IF` statements and `LOOP` constructs.

1. Execute the command in the `lab_06_01.sql` file to create the `messages` table. Write a PL/SQL block to insert numbers into the `messages` table.
 - a. Insert the numbers 1 through 10, excluding 6 and 8.
 - b. Commit before the end of the block.
 - c. Execute a `SELECT` statement to verify that your PL/SQL block worked.

Result: You should see the following output:



The screenshot shows the 'Query Result' window in SQL Developer. The status bar indicates 'All Rows Fetched: 8 in 0.03 seconds'. The main area displays a table with 8 rows of data.

	RESULTS
1	1
2	2
3	3
4	4
5	5
6	7
7	9
8	10

4-

The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL script in the 'Query Builder' tab. The script is as follows:

```
DECLARE
v_i NUMBER;
BEGIN
FOR v_i in 1..10 LOOP
IF v_i in (6,8) THEN continue;
END IF;
INSERT INTO hr.messages VALUES(v_i);
END LOOP;
COMMIT;
END;
/
SELECT * FROM hr.messages
```

The status bar at the top indicates '0,403 seconds'. Below the script editor, the 'Script Output' and 'Query Result' tabs are visible. The 'Script Output' tab shows the message 'PL/SQL procedure successfully completed.' and '>>Query Run In:Query Result'. The 'Query Result' tab shows a table with 10 rows of data:

1	2
1	1
2	2
3	3
4	4
5	5
6	7
7	9
8	10

The status bar at the bottom indicates 'Task completed in 0,403 seconds' and 'All Rows Fetched: 8 in 0,004 seconds'.

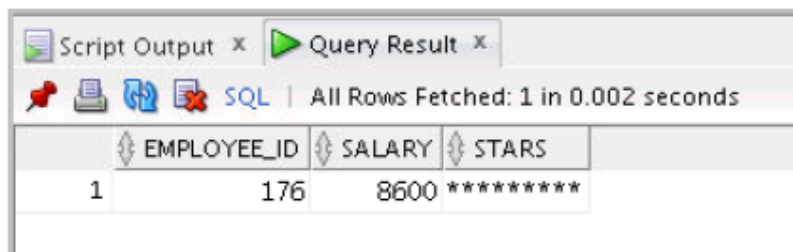
5.

Execute the `lab_06_02.sql` script. This script creates an `emp` table that is a replica of the `employees` table. It alters the `emp` table to add a new column, `stars`, of `VARCHAR2` data type and size 50. Create a PL/SQL block that inserts an asterisk in the `stars` column for every \$1000 of an employee's salary. Save your script as `lab_06_02_soln.sql`.

- In the declarative section of the block, declare a variable `v_empno` of type `emp.employee_id` and initialize it to 176. Declare a variable `v_asterisk` of type `emp.stars` and initialize it to `NULL`. Create a variable `v_sal` of type `emp.salary`.
- In the executable section, write logic to append an asterisk (*) to the string for every \$1,000 of the salary. For example, if the employee earns \$8,000, the string of asterisks should contain eight asterisks. If the employee earns \$12,500, the string of asterisks should contain 13 asterisks (rounded to the nearest whole number).
- Update the `stars` column for the employee with the string of asterisks. Commit before the end of the block.

Display the row from the `emp` table to verify whether your PL/SQL block has executed successfully.

Execute and save your script as `lab_06_02_soln.sql`. The output is as follows:



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with three columns: 'EMPLOYEE_ID', 'SALARY', and 'STARS'. The table contains one row with the following values: '1', '176', and '8600 *****'. The status bar at the top indicates 'All Rows Fetched: 1 in 0.002 seconds'.

EMPLOYEE_ID	SALARY	STARS
1	176	8600 *****

5-

The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL script in the 'Query Builder' tab. The script includes a table creation, variable declarations, a loop to calculate a salary percentage, and an update statement. The 'Script Output' window at the bottom left shows the message 'PL/SQL procedure successfully completed.' The 'Query Result' window at the bottom right shows a table with three columns: EMPLOYEE_ID, SALARY, and STARS, containing one row of data.

```
CREATE TABLE hr.emp AS SELECT * FROM hr.employees;
ALTER TABLE hr.emp ADD stars varchar2(50);

DECLARE
  v_empno hr.emp.employee_id%TYPE := 176;
  v_asterisk hr.emp.stars%TYPE := NULL;
  v_sal hr.emp.salary%TYPE;
BEGIN
  SELECT NVL(RXND(salary/1000), 0) INTO v_sal
  FROM hr.emp WHERE employee_id = v_empno;
  FOR i IN 1..v_sal
  LOOP
    v_asterisk := v_asterisk || '*';
  END LOOP;
  UPDATE hr.emp SET stars = v_asterisk WHERE employee_id = v_empno;
  COMMIT;
END;
```

PL/SQL procedure successfully completed.

EMPLOYEE_ID	SALARY	STARS
176	8600	*****