1.

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

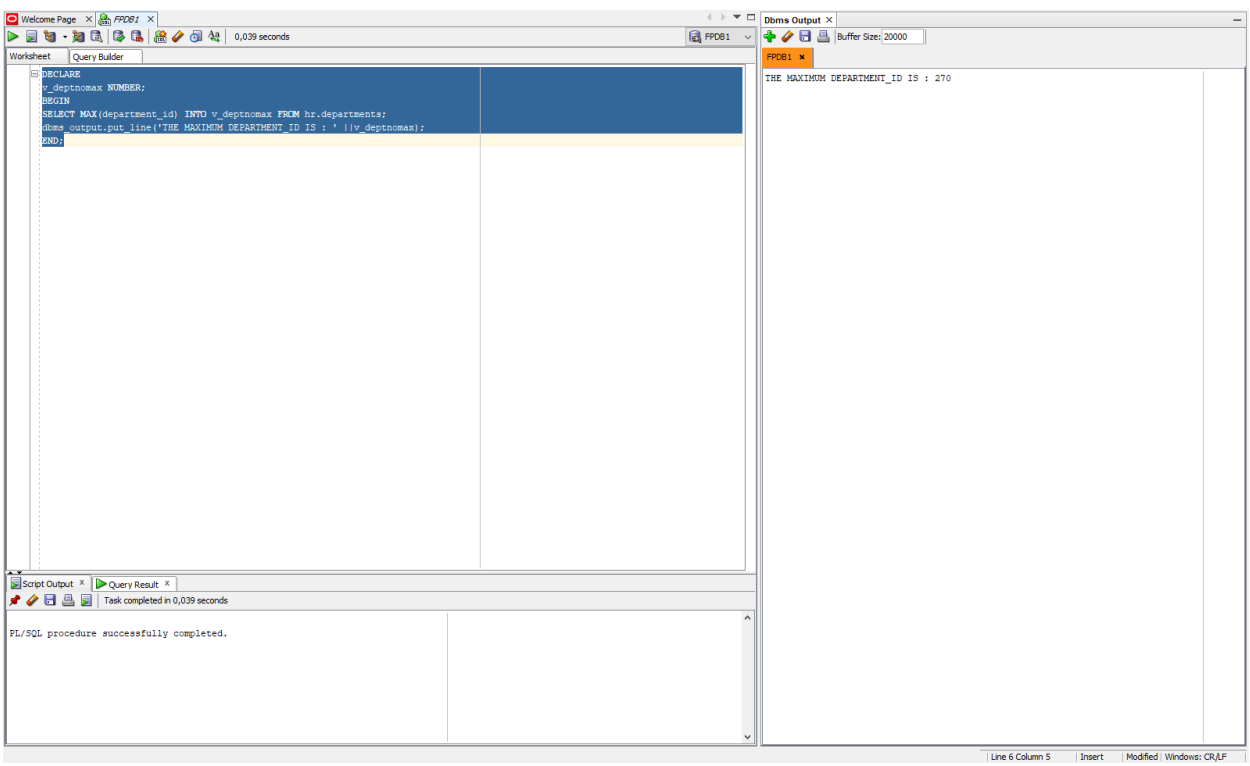
```
DROP table employees2;
DROP table copy_emp;
```

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-



2.

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

a. 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.

b. 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`.

c. 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`.

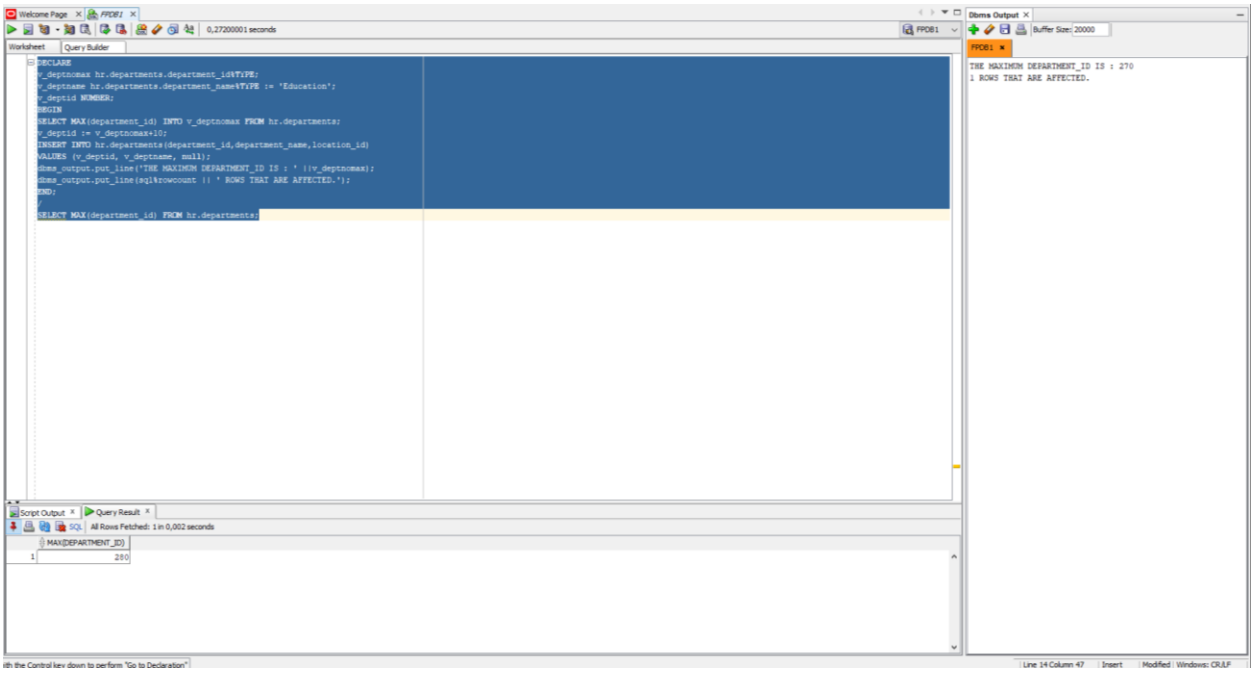
d. Use the SQL attribute `SQL%ROWCOUNT` to display the number of rows that are affected.

e. 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.

f. 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-



3.

Query Result

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`)

a. Start the executable block with the `BEGIN` keyword. Include the `UPDATE` statement t set `location_id` to `3000` for the new department (`v_dept_id` =280).

b. 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.

c. Include a `DELETE` statement to delete the department that you added.

d. Execute and save your script as `lab_05_03_soln.sql`. The sample output is as follows:

Script Output | Query Result

Task completed in 0.425 seconds

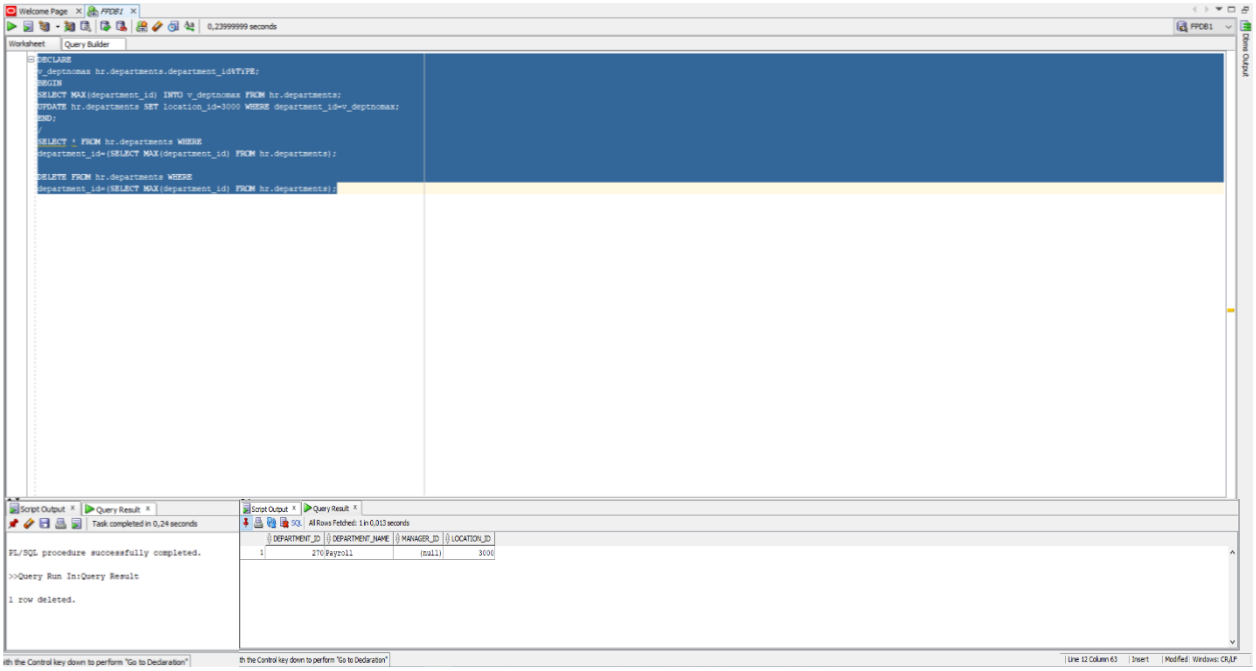
```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result

1 row deleted.
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.172 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	280	Education	(null)	3000

3-

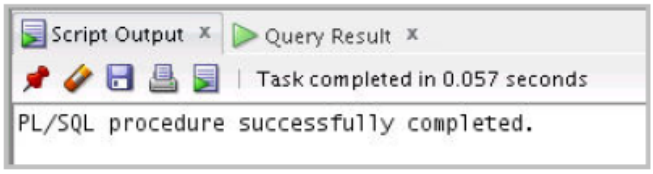

Welcome Page
FPDB1
0,23999999 seconds
Worksheet
Query Builder
DECLARE
v_deptnomax hr.departments.department_id%TYPE;
BEGIN
SELECT MAX(department_id) INTO v_deptnomax FROM hr.departments;
UPDATE hr.departments SET location_id=3000 WHERE department_id=v_deptnomax;
END;
/
SELECT * FROM hr.departments WHERE
department_id=(SELECT MAX(department_id) FROM hr.departments);
DELETE FROM hr.departments WHERE
department_id=(SELECT MAX(department_id) FROM hr.departments);
Script Output
Query Result
Task completed in 0,24 seconds
PL/SQL procedure successfully completed.
>>Query Run In:Query Result
1 row deleted.
All Rows Fetched: 1 in 0,013 seconds
DEPARTMENT_ID
DEPARTMENT_NAME
MANAGER_ID
LOCATION_ID
1
270
Payroll
(null)
3000
Line 12 Column 63
Insert
Modified
Windows: CR/LF

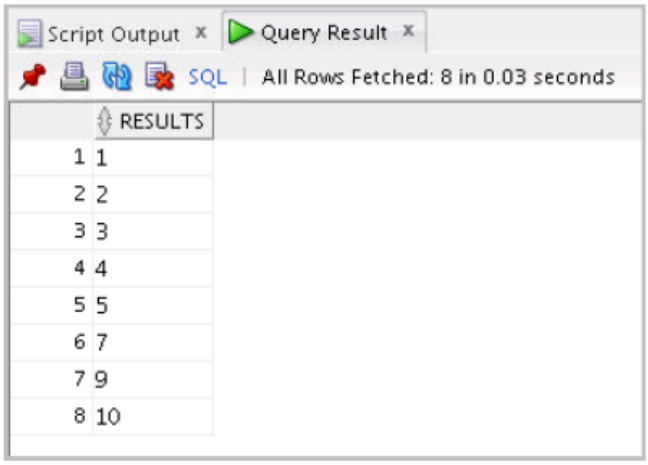
4.

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.

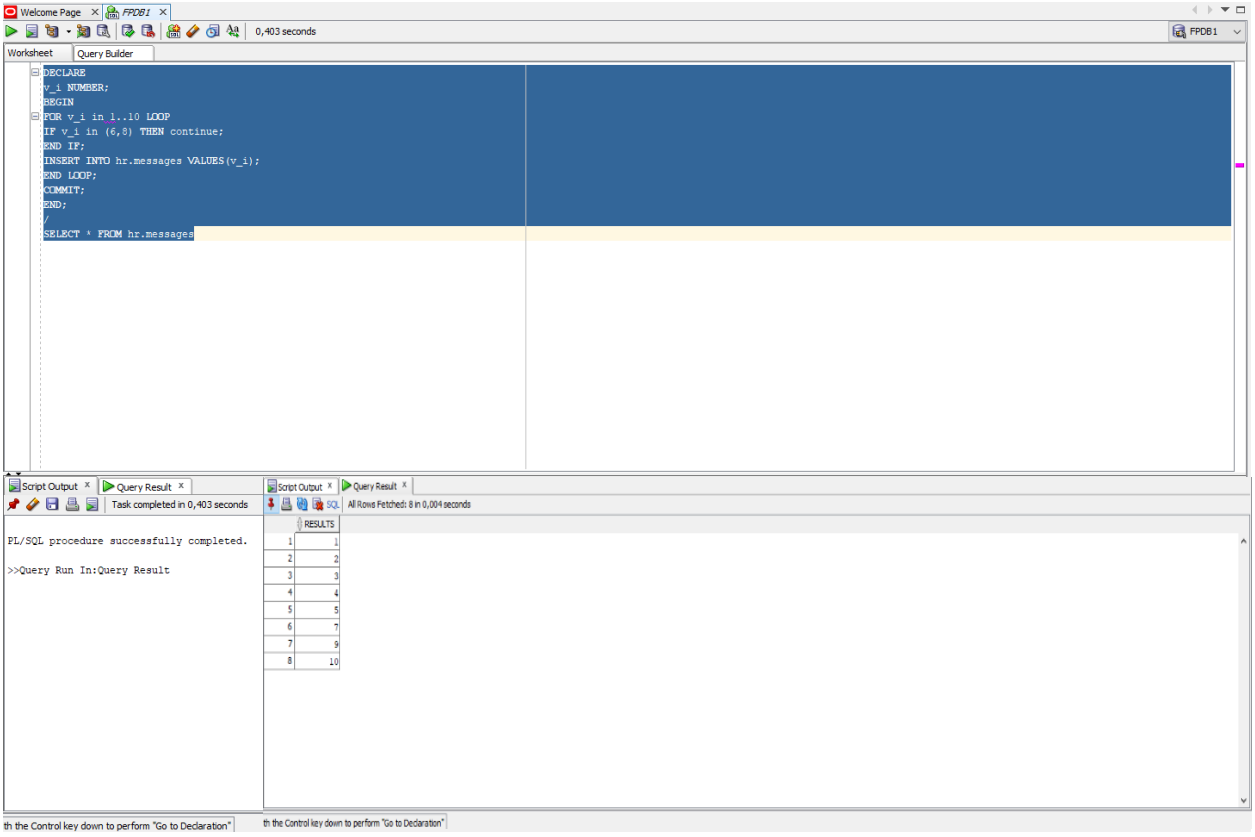
1. Execute the command in the `lab_06_01.sql` file to create the `messages` table. Write a PL/SQL block to insert numbers into the `messages` table.
   a. Insert the numbers 1 through 10, excluding 6 and 8.
   b. Commit before the end of the block.
   c. Execute a `SELECT` statement to verify that your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

Result: You should see the following output:





4-


Welcome Page
FPDB1
0,403 seconds
FPDB1
Worksheet
Query Builder
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
Script Output
Query Result
Task completed in 0,403 seconds
PL/SQL procedure successfully completed.
>>Query Run In:Query Result
Script Output
Query Result
All Rows Fetched: 8 in 0,004 seconds
RESULTS
1 1
2 2
3 3
4 4
5 5
6 7
7 9
8 10
th the Control key down to perform "Go to Declaration"
th the Control key down to perform "Go to Declaration"

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`.

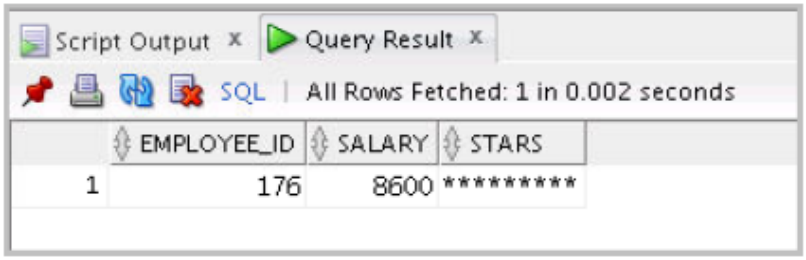
a. In the declarative section of the block, declare a variable `v_empno` of type `emp.employee_id` and initialize it to 176. Declare a variable `v_asterisk` of type `emp.stars` and initialize it to `NULL`. Create a variable `v_sal` of type `emp.salary`.

b. 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).

c. Update the `stars` column for the employee with the string of asterisks. Commit before the end of the block.

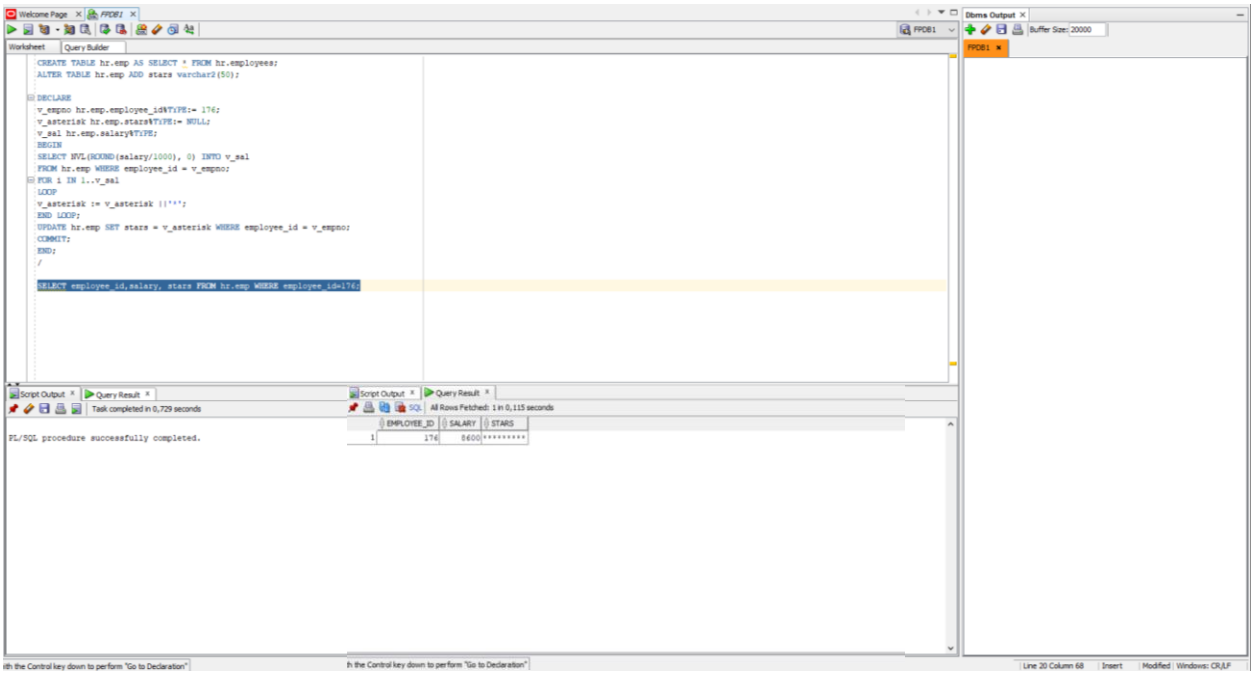
Display the row from the `emp` table to verify whether your PL/SQL block has executed successfully.

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



	EMPLOYEE_ID	SALARY	STARS
1	176	8600	*********

5-


Welcome Page
FPDB1
Worksheet
Query Builder
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(ROUND(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;
/
SELECT employee_id,salary, stars FROM hr.emp WHERE employee_id=176;
Dbms Output
Buffer Size: 20000
Script Output
Query Result
Task completed in 0,729 seconds
PL/SQL procedure successfully completed.
All Rows Fetched: 1 in 0,115 seconds
EMPLOYEE_ID
SALARY
STARS
1
176
8600
****
Line 20 Column 68
Insert
Modified
Windows: CR/LF