

Practices for Lesson 6: Writing Control Structures

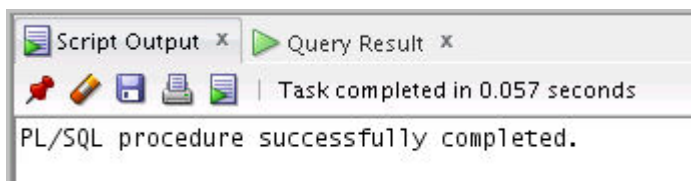
Chapter 6

Practice 6: Writing Control Structures

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.

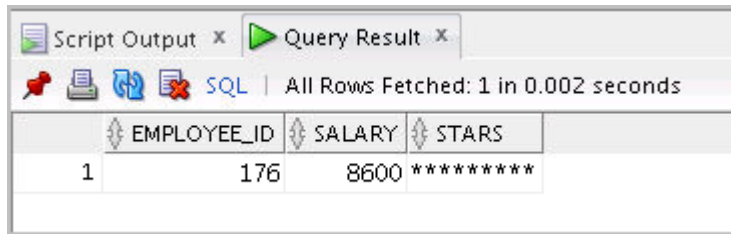
Result: You should see the following output:

A screenshot of the SQL Developer Query Result window. The window has two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, showing a table with 8 rows of data. The table has a header row labeled 'RESULTS' and a body with 8 rows of data. The data is as follows:

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

2. 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.

- d. Display the row from the emp table to verify whether your PL/SQL block has executed successfully.
- e. Execute and save your script as lab_06_02_soln.sql. The output is as follows:



The screenshot shows a SQL Developer window titled 'Query Result'. It displays a single row of data from a table. The columns are EMPLOYEE_ID, SALARY, and STARS. The values are 1, 176, and 8600 ***** respectively. The status bar indicates 'All Rows Fetched: 1 in 0.002 seconds'.

EMPLOYEE_ID	SALARY	STARS
1	176	8600 *****

Solution 6: Writing Control Structures

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.

```
BEGIN
FOR i in 1..10 LOOP
  IF i = 6 or i = 8 THEN
    null;
  ELSE
    INSERT INTO messages(results)
    VALUES (i);
  END IF;
END LOOP;
COMMIT;
END;
/
```

- c. Execute a `SELECT` statement to verify that your PL/SQL block worked.

```
SELECT * FROM messages;
```

Result: You should see the following output:

The screenshot shows two windows from SQL Developer. The top window, titled 'Script Output', displays the message 'PL/SQL procedure successfully completed.' after running a script. The bottom window, titled 'Query Result', shows the results of a 'SELECT * FROM messages;' query. It indicates 'All Rows Fetched: 8 in 0.03 seconds' and 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

2. 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 the 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`.

```
DECLARE
    v_empno          emp.employee_id%TYPE := 176;
    v_asterisk       emp.stars%TYPE := NULL;
    v_sal            emp.salary%TYPE;
```

- 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.

```
BEGIN
    SELECT NVL(ROUND(salary/1000), 0) INTO v_sal
    FROM emp WHERE employee_id = v_empno;

    FOR i IN 1..v_sal
        LOOP
            v_asterisk := v_asterisk || '*';
        END LOOP;
```

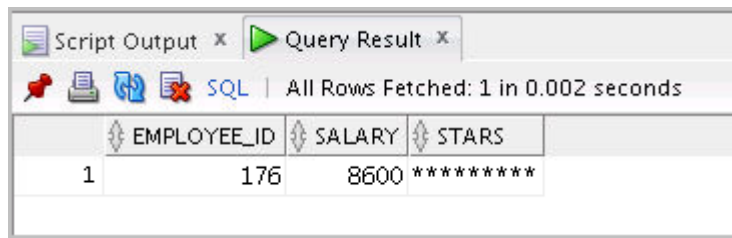
- c. Update the `stars` column for the employee with the string of asterisks. Commit before the end of the block.

```
    UPDATE emp SET stars = v_asterisk
    WHERE employee_id = v_empno;
    COMMIT;
END;
/
```

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

```
SELECT employee_id, salary, stars
FROM emp WHERE employee_id = 176;
```

- e. 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 of data. The status bar indicates 'All Rows Fetched: 1 in 0.002 seconds'.

EMPLOYEE_ID	SALARY	STARS
1	176	8600 *****