PL/SQL_05

1. Execute the command in the lab_05_01.sql file to create the messages table:

```
-- lab_05_01.sql

DROP TABLE messages;
CREATE TABLE messages (results VARCHAR2(80));
```

Write a PL/SQL block to insert numbers into the messages table.

a) Insert the numbers 1 to 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.

You should see the following output:

```
PL/SQL procedure successfully completed.

RESULTS
--------------------------------------------------------------------------------
1
2
3
4
5
7
9
10

8 rows selected.
```

2. Execute the lab_05_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:

```
--lab_05_02.sql

DROP TABLE emp;
CREATE TABLE emp AS SELECT * FROM employees;
ALTER TABLE emp ADD stars VARCHAR2(50);
```

Create a PL/SQL block that inserts an asterisk in the stars column for every $1,000 of the employee's salary. Save your script as lab_05_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 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 amount. 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.

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 that your PL/SQL block executed successfully.

e) Execute and save your script as lab_05_02_soln.sql. The following should be the output:

```
PL/SQL procedure successfully completed.


EMPLOYEE_ID    SALARY    STARS
-------------------    ----------    ----------------------
            176         8600    *********
```

3. Write a PL/SQL block to accept a year and check whether it is a leap year. For example, if the year entered is 1990, the output should be "1990 is not a leap year."

Hint: The year should be exactly divisible by 4 but not divisible by 100, or it should be divisible by 400.
Save your script as lab_05_03_soln.sql.

Test your solution with the following years:

1900	Not a leap year
2000	Leap year
1996	Leap year
1886	Not a leap year
1992	Leap year

You should see the following output:

```
2000 is a leap year

PL/SQL procedure successfully completed.
```