EX.NO:12 03.05.2025	**PROCEDURES AND FUNCTIONS**

AIM:

To develop and execute PL/SQL procedures and functions to perform specific tasks, understand parameter passing, and demonstrate modular programming in Oracle PL/SQL.

CREATING TABLE

```
SQL> create table rectangle(length number(5),breadth number(5),area number(10,2));

Table created.

SQL> insert into rectangle values(5, 4, 20);

 1  row created.

SQL> insert into rectangle values(6, 3, 18);

 1  row created.

SQL>

SQL> insert into rectangle values(7, 2, 14);

1 row created.
```

A SIMPLE PL/SQL PROCEDURE

```
SQL> declare
  2    length number := 8;
  3    breadth number := 5;
  4    area number(10,2);
  5    begin
  6    area := length * breadth;
  7    insert into rectangle values (length, breadth, area);
  8    dbms_output.put_line('rectangle inserted: area = ' || area);
  9    end;
 10  /

PL/SQL procedure successfully completed.

SQL> select * from rectangle;
```

LENGTH	BREADTH	AREA
5	4	20
6	3	18
7	2	14
8	5	40

PL/SQL PROCEDURE WITH SIMPLE LOOP

```
SQL> declare
  2  length number := 1;
  3  breadth number := 4;
  4  area number(10,2);
  5  begin
  6  for i in 1..5 loop
  7  area := length * breadth;
  8  insert into rectangle values(length, breadth, area);
  9  length := length + 1;
  10 end loop;
  11 dbms_output.put_line('5 rectangles inserted successfully.');
  12 end;
  13 /
```

PL/SQL procedure successfully completed.

```
SQL> select * from rectangle;
```

LENGTH	BREADTH	AREA
5	4	20
6	3	18
7	2	14
8	5	40

1	4	4
2	4	8
3	4	12
4	4	16
5	4	20
1	4	4
2	4	8

PL/SQL PROCEDURE WITH FOR LOOP

```
SQL> declare
length number := 1;
breadth number := 4;
area number(10,2);
begin
for i in 1..5 loop
area := length * breadth;
insert into rectangle values(length, breadth, area);
length := length+1;
end loop;
dbms_output.put_line('5 rectangles inserted successfully.');
end;
/
```

PL/SQL procedure successfully completed.

SQL> select * from rectangle;

LENGTH	BREADTH	AREA
5	4	20
6	3	18

7	2	14
8	5	40
1	4	4
2	4	8
3	4	12
4	4	16
5	4	20

9 rows selected.

PL/SQL PROCEDURE WITH WHILE LOOP

```
SQL> declare
  2     length number := 1;
  3     breadth number := 5;
  4     area number(10,2);
  5     begin
  6     while length <= 5 loop
  7     area := length * breadth;
  9     insert into rectangle
 10     values (length, breadth, area);
 12     length := length + 1;
 13     end loop;
 15     dbms_output.put_line('5 rectangles inserted using while loop.');
 16     end;
 17    /
```

PL/SQL procedure successfully completed.

```
SQL> select * from rectangle;
```

LENGTH	BREADTH	AREA
3	4	12

4	4	16
5	4	20
1	5	5
2	5	10
3	5	15
4	5	20
5	5	25

8 rows selected.

PL/SQL PROCEDURE WITH EXCEPTION

```
SQL> declare
  2     length number := 10;
  3     breadth number := 0;
  4     area number(10,2);
  5     begin
  6     area := length / breadth;
  7     dbms_output.put_line('area: ' || area);
  8      exception
  9      when zero_divide then
  10    dbms_output.put_line('error: division by zero is not allowed.');
  11    when others then
  12    dbms_output.put_line('an unexpected error occurred.');
  13    end;
  14    /
SQL> select * from rectangle;

LENGTH          BREADTH          AREA
------------    --------------   ------------
         3                  4             12
```

4	4	16
5	4	20
1	5	5
2	5	10
3	5	15
4	5	20
5	5	25

8 rows selected.

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
Aim,algorithm,SQL,PL/SQL	30	
Execution and Result	20	
Viva	10	
Total	60	

RESULT:

The PL/SQL procedure and function were executed successfully.They performedoperations like inserting data and calculating results using parameters.This experiment demonstrated modular coding and reusability in PL/SQL.