NAME:AVINASH BHIWAL ROLL NO.:T1851050 PRN NO.:71901284J DIV:A

Assignment No. 9

Aim: Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

Objective:

- To study and implement PL/SQL procedure.
- To study and implement PL/SQL function.

Theory :

A **subprogram** is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the **calling program**.

A subprogram can be created −

- At the schema level
- Inside a package
- Inside a PL/SQL block

At the schema level, subprogram is a **standalone subprogram**. It is created with the CREATE PROCEDURE or the CREATE FUNCTION statement. It is stored in the database and can be deleted with the DROP PROCEDURE or DROP FUNCTION statement.

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parame**ters. PL/SQL provides two kinds of subprograms −**

- **Functions − These subprograms return a single value; mainly used to compute** and return a value.
- **Procedures − These subprograms do not return a value directly; mainly used** to perform an action.

Parts of a PL/SQL Subprogram

Each PL/SQL subprogram has a name, and may also have a parameter list. Like **anonymous PL/SQL blocks, the named blocks will also have the following three parts −**

```
CREATE OR REPLACE PROCEDURE greetings CREATE
[OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [,
...])] {IS | AS}
BEGIN
  < procedure_body >
```

S.No	
1	
	subprogram and cease to exist when the subprogram completes execution.
2	**Executable Part** This is a mandatory part and contains statements that perform the designated action.
3	**Exception-handling** This is again an optional part. It contains the code that handles run-time errors.

Creating a Procedure

A procedure is created with the **CREATE OR REPLACE PROCEDURE** statement. The **simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows −**

Where,

- *procedure-name* specifies the name of the procedure.
- [OR REPLACE] option allows the modification of an existing procedure.
- The optional parameter list contains name, mode and typesof the parameters. **IN** represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- *procedure-body* contains the executable part.
-

The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Example

The following example creates a simple procedure that displays the string 'Hello World!' on the screen when executed.

```
AS
BEGIN
  dbms_output.put_line('Hello World!');
END;
/
```

Deleting a Standalone Procedure

A standalone procedure is deleted with the **DROP PROCEDURE** statement. Syntax for **deleting a procedure is −**

```
DROP PROCEDURE procedure-name;
```

You can drop the **greetings procedure by using the following statement −**

```
DROP PROCEDURE greetings;
```

Parameter Modes in PL/SQL Subprograms

The following table lists out the parameter modes in PL/SQL subprograms −

S.No	Parameter Mode & Description
1	**IN** An IN parameter lets you pass a value to the subprogram. **It is a read-only parameter**. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. **It is the default mode of parameter passing. Parameters are passed by reference**.
2	**OUT** An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. **The actual parameter must be variable and it is passed by value**.

	IN OUT
3	An **IN OUT** parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read. The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. **Actual parameter is passed by value.**

Functions:

A function is same as a procedure except that it returns a value. Therefore, all the discussions of the previous chapter are true for functions too.

Creating a Function

A standalone function is created using the **CREATE FUNCTION** statement. The simplified syntax for the **CREATE OR REPLACE PROCEDURE** st**atement is as follows −**

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
  < function_body >
END
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the
- parameter that will be used to return a value outside of the procedure.
  The function must contain a **return** statement.
- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

Conclusion:-

We have studied and executed PL/SQL stored procedures and functions.

1. CREATE SIMPLE PROCEDURE:

```
mysql> DELIMITER /
mysql> CREATE PROCEDURE DISP() BEGIN SELECT 'HELLO WORLD' AS OUTPUT;
-> END;
-> /
Query OK, 0 rows affected (0.02 sec)

mysql> CALL DISP()/
  +-------------+
  | OUTPUT      |
  +-------------+
  | HELLO WORLD |
  +-------------+
  1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
```

2. CREATE TABLE STUDENT AND INSERT DATA:

```
mysql> CREATE TABLE STUDENT(ROLL_NO INT PRIMARY
KEY, NAME VARCHAR(20), CLASS VARCHAR(20), MARKS
INT); Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO STUDENT VALUES (1,
'AKASH',null,900),
(2,'NIKHIL',null,100),(3,'SHREEJIT',null,600),(4,'ATHIRA',null,800),
(5,'ADITI',null,200),(6,'SAKSHI',null,50);
Query OK, 6 rows affected (0.01 sec)
Records: 6     Duplicates: 0       Warnings: 0

mysql> SELECT * FROM STUDENT;
  +---------+          +-------+-------  +
  | ROLL_NO | NAME      | CLASS | MARKS |
  +---------+          +-------+-------  +
  |          1 | AKASH     | NULL    |    900|
  |          2 | NIKHIL    | NULL    |    100|
  |          3 | SHREEJIT |  NULL    |    600|
  |          4 | ATHIRA    | NULL    |    800|
  |          5 | ADITI     | NULL    |    200|
  |          6 | SAKSHI    | NULL    |     50|
  +---------+          +-------+-------  +
  6 rows in set (0.00 sec)
```

3. CREATE PROCEDURE TO SHOW DATA OF STUDENT:

```
mysql> DELIMITER /
mysql> CREATE PROCEDURE GETDATA(IN ID INT)
-> BEGIN
-> SELECT * FROM STUDENT WHERE ROLL_NO=ID;
-> END;
-> /
```

```
Query OK, 0 rows affected (0.01 sec)
```

4. CREATE PROCEDURE TO UPDATE STATUS OF STUDENT:

```
mysql> CREATE PROCEDURE UPDATECLASS()
-> BEGIN
-> UPDATE STUDENT SET CLASS = 'DISTINCTION' WHERE MARKS<=1500
AND MARKS>=990;
-> UPDATE STUDENT SET CLASS = 'FIRST CLASS' WHERE MARKS<=989
AND MARKS>=900;
-> UPDATE STUDENT SET CLASS = 'SECOND CLASS' WHERE
MARKS<=899 AND MARKS>=825;
-> UPDATE STUDENT SET CLASS = 'POOR' WHERE MARKS<=824;
-> END;
-> /
Query OK, 0 rows affected (0.01 sec)
```

5. CALLING OF PROCEDURES:

```
mysql> CALL UPDATECLASS()/
Query OK, 5 rows affected (0.01 sec)

mysql> SELECT * FROM STUDENT/
  +---------+---------+-------------+-------+
  | ROLL_NO | NAME    | CLASS       | MARKS |
  +---------+---------+-------------+-------+
  |       1 | AKASH   |  FIRST CLASS |   900|
  |       2 | NIKHIL  |   POOR      |   100|
  |       3 | SHREEJIT |   POOR     |   600|
  |       4 | ATHIRA  |   POOR      |   800|
  |       5 | ADITI   |   POOR      |   200|
  |       6 | SAKSHI  |   POOR      |    50|
  +---------+---------+-------------+-------+
6 rows in set (0.00 sec)


mysql> CALL GETDATA(1)/
  +---------+---------+-------------+-------+
  | ROLL_NO | NAME    | CLASS       | MARKS |
  +---------+---------+-------------+-------+
  |       1 | AKASH | FIRST CLASS |     900 |
  +---------+---------+-------------+-------+
  1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

6. CREATE FUNCTION AND CALLING IT:

```
mysql> CREATE FUNCTION SHOW_MARKS(ID INT(10))
-> RETURNS CHAR(100) DETERMINISTIC
-> BEGIN
-> DECLARE V INT(10);
```

```
-> SELECT MARKS INTO V FROM STUDENT WHERE ID=ROLL_NO;
-> RETURN V;
-> END/
Query OK, 0 rows affected, 2 warnings (0.00 sec)

mysql> SELECT SHOW_MARKS(2) AS MARKS FROM STUDENT/
  +-------+
  | MARKS |
  +-------+
  |100    |
  +-------+
1 rows in set (0.00 sec)
```

7. LIST OF PROCEDURES AND FUNCTIONS:

```
mysql> SHOW PROCEDURE STATUS WHERE DB = 'APPLICATIONS'/
+--------------+--------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+----------------------+
| Db           | Name         | Type      | Definer        | Modified            | Created             | Security_type | Comment | character_set_client | collation_connection | Database Collation |
+--------------+--------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+----------------------+
| applications | DISP         | PROCEDURE | root@localhost | 2020-11-11 11:29:02 | 2020-11-11 11:29:02 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci   |
| applications | GETDATA      | PROCEDURE | root@localhost | 2020-11-11 11:38:17 | 2020-11-11 11:38:17 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci   |
| applications | GETSAL       | PROCEDURE | root@localhost | 2020-11-11 11:39:59 | 2020-11-11 11:39:59 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci   |
| applications | UPDATECLASS  | PROCEDURE | root@localhost | 2020-11-11 11:45:51 | 2020-11-11 11:45:51 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci   |
+--------------+--------------+-----------+----------------+---------------------+---------------------+---------------+---------+----------------------+----------------------+----------------------+
4 rows in set (0.00 sec)
```