Experiment No. : 10

Title: Write Stored Procedure and Function in PL/SQL.

Objectives:

1. To learn stored procedures in PL/SQL.
2. To learn functions in PL/SQL.

Key Concepts: PL/SQL, function, stored procedure

Theory:

Procedure is a subprogram which consists of a set of SQL statements. It is not very different from function. A procedure or function is a logically grouped set of SQL statements that perform a specific task. A stored procedure or function is a named PL/SQL code block that have been compiled and stored in one of the Oracle engine's system tables.

To make a procedure or function dynamic either of them can be passed parameters before execution. A procedure or function can then change the way it works depending upon the parameters passed prior to its execution.

Procedures and functions are made up of a declarative part, an executable part and an optional exception-handling part. A declaration part consists of declarations of variables. An executable part consist of the logic in form of SQL statements and exception handling part handles any error during run-time.

Difference between procedures and functions:

A function must return a value back to the caller. A function can return only one value. By defining multiple out parameters in a procedure, multiple values can be passed to the caller. The out variable being global by nature.

Syntax to create stored procedure:

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

```
   < procedure_body >
END procedure_name;
```

Example: create stored procedure:

Create a procedure that takes the name as input and prints the welcome message as output.

```
CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN VARCHAR2)
IS
BEGIN
dbms_output.put_line ('Welcome '|| p_name);
END;
/
```

To Execute use EXEC command to call procedure

```
EXEC welcome_msg ('DKTE');
```

Syntax to create function:

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

IN : This parameter is used for giving input to the subprograms. It is a read-only variable inside the subprograms.

OUT : This parameter is used for getting output from the subprograms. It is a read-write variable inside the subprograms.

IN OUT : This parameter is used for both giving input and for getting output from the subprograms. It is a read-write variable inside the subprograms.

Example: create function:

```
CREATE OR REPLACE FUNCTION welcome_msg_func ( p_name IN VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
RETURN ('Welcome '|| p_name);
END;
/
DECLARE
lv_msg VARCHAR2(250);
BEGIN
lv_msg := welcome_msg_func ('Guru99');
dbms_output.put_line(lv_msg);
END;

SELECT welcome_msg_func('Guru99') FROM DUAL;
```

Syntax to drop stored procedure and functions:

```
Drop procedure procedure_name;
Drop function function_name;
```

Algorithm:

1. Start
2. Create table as per required field for stored procedure.
3. Insert data into above created table.
4. Write stored procedure to retrieve the balance as OUT parameter when user input account number as IN parameter.
5. Write PL/SQL block to Input account number from user and Call above created stored procedure and Display retrieved balance.
6. Execute the above PL/SQL block
7. Stop.

Examples:

1. Write a procedure which accept the account number of a customer and retrieve the balance.
2. Write a procedure which accept the deptno and print minimum salary of employee working in that department.
3. Write a function which accept a deptno and check whether it is present in dept Table or not. If it is present print number of employees working in that department.