15. Demonstrate Procedures, Functions, and Packages in PLSQL.

subprogram

- A **subprogram** is a program unit/module that performs a particular task.
- A subprogram can be invoked by another program which is called the **calling program.**
- A subprogram can be created –
  - At the schema level
  - Inside a package
  - Inside a PL/SQL block

subprogram

- 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**.**

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.
- **Declarative Part**
- **Executable Part**
- **Exception-handling**

Procedure

- **Procedure** in PL/SQL is a sub-program mainly used to perform an action**.**
- It is also called stored procedure.
- It is used to perform DML opeartions.

Creating a Procedure

Syntax:

```
CREATE  [OR REPLACE] PROCEDURE  <procedure_name>
[ parameter_name   IN | OUT | IN OUT type,…]
IS | AS
BEGIN
< procedure_body >
END procedure_name;
```

Creating Procedure – Example1

```
CREATE OR REPLACE PROCEDURE hello
IS
BEGIN
dbms_output.put_line('Hello World!');
END;
/
```

Execute a procedure

- A procedure can be executed in two ways.

1. Using the **EXECUTE keyword** IN SQL PROM**PT.**

- **SYNTAX**:
- EXECUTE procedurename;
- Example : **SQL>** EXECUTE HELLO;

  Hello World

  PL/SQL procedure successfully completed.

Execute a procedure

2. The procedure can also be called from another PL/SQL block –

```
BEGIN
hello;
END;
/
```

```
Hello World
PL/SQL procedure successfully completed.
```

Types of Parameters - Procedure

- Parameters are used to send or receive runtime values.
- **IN** - used to send values to a procedure
- **OUT** - used to get values from procedure.
- **IN OUT-** used to send values to and get values from a procedure.

IN- Parameters Example

```
CREATE OR REPLACE PROCEDURE  addition
 ( x in NUMBER, y in NUMBER)
IS
Z number;
BEGIN
z := x + y;
dbms_output.put_line(‘ result is  ' || z);
END;
/
```

Output:

SQL> exec addition(1,2);

Result is 3.

IN OUT PARAMETER- PROCEDURE

computes the square of a passed value.

```
CREATE PROCEDURE squarenum(x IN OUT number)
IS
BEGIN
x := x * x;
END;
/
```

```
declare   a NUMBER:= 10;
BEGIN
Dbms_output.put_line( ‘ before calling value a is’ || a);
squarenum(a);
dbms_output.put_line(' Square of  10 :  ' || a);
END;
```

Output: square of 10 is 100

Create a procedure to find no of employees

```
create or replace procedure p1  is
total number;
begin
select Count(*) into total from emp;
dbms_output.put_line('no of employees ' || total);
end;
/
```

```
begin
P1;
end;
```

PL/sql procedure to insert a row into emp

```
create or replace procedure  insert_emp
(id IN NUMBER, name IN VARCHAR2(20))
is
begin
insert into emp values(id, name);
 dbms_output.put_line(‘ inserted row successfully’);
end;
/
```

```
Exec insert_emp(1,’a’);
```

Output: inserted successfully.

Drop procedure

- **DROP PROCEDURE** <name>;

- **Example**: drop procedure p1;
- **Output**: procedure dropped.

function

- A function is same as a procedure except that it returns a value

CREATE [OR REPLACE] **FUNCTION** function_name
[parameter]
RETURN return_datatype
IS | **AS**
[declaration_section]
BEGIN
executable_section
[EXCEPTION - exception_section]
END ;

Create a FUNCTION to find no of employees

```
CREATE OR REPLACE FUNCTION totalemp
RETURN number
IS
   total number ;
BEGIN
   SELECT count(*) into total FROM emp;
   RETURN total;
END;
```

```
DECLARE x NUMBER;
BEGIN
X  := TOTALEMP;
dbms_output.put_line('no of employees ' || X);
END;
/
```

PL/sql function to insert a row into emp

```
create or replace function  insert_emp
(id IN NUMBER, name IN VARCHAR2(20))
Return number
is
begin
insert into emp values(id, name);
 dbms_output.put_line(‘ inserted row successfully’);
Return 0;
end;
/
```

_____

```
Declare a number;
Begin
a :=  insert_emp(2,’b’);
End;
/
```

Output: inserted successfully.