EXERCISE-16

PROCEDURES AND FUNCTIONS

PROCEDURES

DEFINITION

A procedure or function is a logically grouped set of SQL and PL/SQL statements that perform a specific task. They are essentially sub-programs. Procedures and functions are made up of,

- Declarative part
- Executable part
- Optional exception handling part

These procedures and functions do not show the errors.

KEYWORDS AND THEIR PURPOSES

REPLACE: It recreates the procedure if it already exists.

PROCEDURE: It is the name of the procedure to be created.

ARGUMENT: It is the name of the argument to the procedure. Paranthesis can be omitted if no arguments are present.

IN: Specifies that a value for the argument must be specified when calling the procedure ie. used to pass values to a sub-program. This is the default parameter.

OUT: Specifies that the procedure passes a value for this argument back to it's calling environment after execution ie. used to return values to a caller of the sub-program.

INOUT: Specifies that a value for the argument must be specified when calling the procedure and that procedure passes a value for this argument back to it's calling environment after execution.

RETURN: It is the datatype of the function's return value because every function must return a value, this clause is required.

PROCEDURES – SYNTAX

```
create or replace procedure <procedure name> (argument {in,out,inout} datatype ) {is,as}
variable declaration;
constant declaration;
begin
PL/SQL subprogram body;
exception
exception PL/SQL block;
end;
```

FUNCTIONS – SYNTAX

```
create or replace function <function name> (argument in datatype,……) return datatype {is,as}
```

```
variable declaration;
constant declaration;
begin
PL/SQL subprogram body;
exception
exception PL/SQL block;
end;
```

CREATING THE TABLE 'ITITEMS' AND DISPLAYING THE CONTENTS

```
SQL> create table ititems(itemid number(3), actualprice number(5), ordid number(4), prodid
number(4));
Table created.

SQL> insert into ititems values(101, 2000, 500, 201);
1 row created.

SQL> insert into ititems values(102, 3000, 1600, 202);
1 row created.

SQL> insert into ititems values(103, 4000, 600, 202);
1 row created.

SQL> select * from ititems;
ITEMID  ACTUALPRICE    ORDID    PRODID
---------  -----------  --------  ---------
     101        2000       500       201
     102        3000      1600       202
     103        4000       600       202
```

PROGRAM FOR GENERAL PROCEDURE – SELECTED RECORD'S PRICE IS INCREMENTED BY 500 , EXECUTING THE PROCEDURE CREATED AND DISPLAYING THE UPDATED TABLE

```
SQL> create procedure itsum(identity number, total number) is price number;
  2  null_price exception;
  3  begin
  4  select actualprice into price from ititems where itemid=identity;
  5  if price is null then
  6  raise null_price;
  7  else
  8  update ititems set actualprice=actualprice+total where itemid=identity;
  9  end if;
 10  exception
 11  when null_price then
 12  dbms_output.put_line('price is null');
 13  end;
 14  /
Procedure created.

SQL> exec itsum(101, 500);
```

NEIL DANIEL A - 240701356 - DBMS RECORD

PL/SQL procedure successfully completed.

```
SQL> select * from ititems;
 ITEMID   ACTUALPRICE      ORDID     PRODID
---------  -----------  ---------  ---------
     101         2500        500        201
     102         3000       1600        202
     103         4000        600        202
```

PROCEDURE FOR 'IN' PARAMETER – CREATION, EXECUTION

```
SQL> set serveroutput on;

SQL> create procedure yyy (a IN number) is price number;
  2  begin
  3  select actualprice into price from ititems where itemid=a;
  4  dbms_output.put_line('Actual price is ' || price);
  5  if price is null then
  6  dbms_output.put_line('price is null');
  7  end if;
  8  end;
  9  /
Procedure created.

SQL> exec yyy(103);
Actual price is 4000
PL/SQL procedure successfully completed.
```

PROCEDURE FOR 'OUT' PARAMETER – CREATION, EXECUTION

```
SQL> set serveroutput on;

SQL> create procedure zzz (a in number, b out number) is identity number;
  2  begin
  3  select ordid into identity from ititems where itemid=a;
  4  if identity<1000 then
  5   b:=100;
  6  end if;
  7  end;
  8  /
Procedure created.

SQL> declare
  2  a number;
  3  b number;
  4  begin
  5  zzz(101,b);
  6  dbms_output.put_line('The value of b is '|| b);
  7  end;
  8  /
The value of b is 100
```

PL/SQL procedure successfully completed.

PROCEDURE FOR 'INOUT' PARAMETER – CREATION, EXECUTION

```
SQL> create procedure itit ( a in out number) is
  2  begin
  3  a:=a+1;
  4  end;
  5  /
  Procedure created.

SQL> declare
  2  a number:=7;
  3  begin
  4  itit(a);
  5  dbms_output.put_line('The updated value is '||a);
  6  end;
  7  /
The updated value is 8
PL/SQL procedure successfully completed.
```

CREATE THE TABLE 'ITTRAIN' TO BE USED FOR FUNCTIONS

```
SQL>create table ittrain ( tno number(10), tfare number(10));
Table created.

SQL>insert into ittrain values (1001, 550);
1 row created.

SQL>insert into ittrain values (1002, 600);
1 row created.

SQL>select * from ittrain;
    TNO      TFARE
---------  ------------
   1001        550
   1002        600
```

PROGRAM FOR FUNCTION AND IT'S EXECUTION

```
SQL> create function aaa (trainnumber number) return number is
  2  trainfunction ittrain.tfare % type;
  3  begin
  4  select tfare into trainfunction from ittrain where tno=trainnumber;
  5  return(trainfunction);
  6  end;
  7  /

Function created.

SQL> set serveroutput on;
```

```
SQL> declare
  2  total number;
  3  begin
  4  total:=aaa (1001);
  5  dbms_output.put_line('Train fare is Rs. '||total);
  6  end;
  7  /
Train fare is Rs.550
PL/SQL procedure successfully completed.
```

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
CREATE OR REPLACE FUNCTION factorial
(n IN NUMBER) RETURN NUMBER IS
  result NUMBER := 1;
BEGIN
  IF n < 0 THEN
    RETURN NULL;
  END IF;
  FOR i IN 1..n LOOP
    result := result * i;
  END LOOP;
  RETURN result;
END;
/
```

```
Function FACTORIAL compiled

Elapsed: 00:00:00.012
```

NEIL DANIEL A - 240701356 - DBMS RECORD

Program 2

Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library

```
CREATE OR REPLACE PROCEDURE get_book_info(
  p_book_id     IN  NUMBER,
  p_title       OUT VARCHAR2,
  p_author      OUT VARCHAR2,
  p_copies      IN OUT NUMBER
) IS
BEGIN
  SELECT title, author, copies INTO p_title, p_author, p_copies
  FROM library_books
  WHERE book_id = p_book_id;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    p_title  := 'NOT FOUND';
    p_author := 'NOT FOUND';
    p_copies := 0;
END;
/
SET SERVEROUTPUT ON;
DECLARE
  v_book_id NUMBER := 1;
  v_title   VARCHAR2(100);
  v_author  VARCHAR2(100);
  v_copies  NUMBER := 0;
BEGIN
  get_book_info(v_book_id, v_title, v_author, v_copies);
  DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
  DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
  DBMS_OUTPUT.PUT_LINE('Copies available: ' || v_copies);
END;
/
```

```
Title: Oracle PL/SQL Programming
Author: Steven Feuerstein
Copies available: 4

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.015
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	