7/10/25

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);
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

```
Set SERVEROUTPUT ON;
Create or replace function factorial (n number)
Return Number
IS
  fact Number := 1;
Begin
  For i in 1..n Loop
     fact := fact * i;
End Loop;
Return fact;
End;
/
Declare
  num Number := 5;
  result Number;
Begin
  result := factorial (num);
  DBMS_OUTPUT.PUT_LINE ('Factorial of ' || num || ' is ' || result
  END;
/
```

Program 2

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

```
Create table library(
    book_id Number Primary Key;
    book_title varchar2(100);
    author_name varchar2(100);
    Price       Number
);
Insert into library values (1,'Database Systems', 'Ramez Elmasri', 550);
Insert into library values (2,'Operating Systems', 'Galvin', 600);
Insert into library values (3,'Computer Networks', 'Andrew')
Create or replace Procedure
get_book_info(
    P_book_id In Number,
    P_book_title out varchar2,
    P_author_title out Varchar2,
    P_Price In Out Number
)
IS
Begin
Select book_title, author_name
P_price
Into P_book_title, P_author_name
P_price
From library
where book_id = P_book_id;
Exception
when No_data_Found Then
    P_book_title = 'Book NOT Found';
    P_author_name := Null;
    P_Price := 0;
```

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

PL/SQL

Control Structures

In addition to SQL commands,PL/SQL can also process data usin flow of statements.the flow of control statements are classified into the following categories.

- Conditional control -Branching
- Iterative control - looping
- Sequential control

BRANCHING in PL/SQL:

Sequence of statements can be executed on satisfying certain condition .

If statements are being used and different forms of if are:

1.Simple IF

2.ELSIF

3.ELSE IF

***SIMPLE IF*:**

Syntax:

```
IF condition THEN
   statement1;
   statement2;
END IF;
```

***IF-THEN-ELSE STATEMENT*:**

Syntax:

```
IF condition THEN
   statement1;
ELSE
   statement2;
END IF;
```

ELSIF STATEMENTS:

Syntax:

```
IF condition1 THEN
  statement1;
```

```
ELSIF condition2 THEN
  statement2;
ELSIF condition3 THEN
  statement3;
ELSE
  statementn;
END IF;
```

NESTED IF :

Syntax:

```
IF condition THEN
   statement1;
ELSE
  IF condition THEN
    statement2;
  ELSE
    statement3;
  END IF;
END IF;
ELSE
  statement3;
END IF;
```

SELECTION IN PL/SQL(Sequential Controls)

SIMPLE CASE

Syntax:

```
CASE SELECTOR
   WHEN Expr1 THEN statement1;
   WHEN Expr2 THEN statement2;
:
```

ELSE

Statement n;

END CASE;

SEARCHED CASE:

CASE

WHEN searchcondition1 THEN statement1;

WHEN searchcondition2 THEN statement2;

:

:

ELSE

statementn;

END CASE;

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct.

It is broadly classified into:

- Simple Loop
- For Loop
- While Loop

SIMPLE LOOP

Syntax:

LOOP

statement1;

EXIT [WHEN Condition];

END LOOP;

WHILE LOOP

Syntax:

WHILE condition LOOP

statement1;

statement2;

END LOOP;

FOR LOOP

Syntax:

```
FOR counter IN [REVERSE]
   LowerBound..UpperBound
LOOP
statement1;
statement2;
END LOOP;
```

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
Set serveroutput ON;
Declare
   V_Salary Number;
   V_incentive Number;
Begin
   Select Salary into V_Salary from employees where
   employee_id = 110;
   If V_Salary >= 50000 then
      V_incentive = V_Salary * 0.10;
   Else if
      V_Salary >=
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
Declare
    "my Var" Number = 100;
BEGIN
    DBMS_Output.Put_Line (my var);
    DBMS_Output.Put_Line ("my var");
    End;
/
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
Declare
    v_emp_id employees.employee_id Type := 122;
Begin
    update employees
    Set Salary = Salary + (Salary * 0.10)
    where employee_Id = v_emp_id;
DBMS Output_Line ("Salary updated Successfully
        for employee ID :" v_emp_id)
Exception : when NO_Data_Found then
    DBMS_Output.Putline ("Employee not found");
when others then
    DBMS_Output_Line ("Error");
END;
```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```
Create or replace procedure check_nulls is
    a Number := 10;
    b Number := NULL;
Begin
   If a is Not NULL and b is Not NULL then
DBMS_output_Put_Line ('Both values are not NULL');
   Else
   DBMS_Output.Put_Line ("At least one value is NUL
   END IF;
END;
/
BEGIN
   check_NULLS;
END;
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
Declare
    V_name_varchar = 20 := 'Rajesh';
Begin
    if v_name Like 'Ray' then
    DBMS_Output.Put_Line ('Name Starts with RA');
    END IF;
If v_name Like 'Rajesh' then
DBMS_Output.Put_Line ("Second Character");
END IF;

If 'A#B' Like 'A#_Y" Escape '#' then
    DBMS_Output.Put_Line ("Escape character
                    used correctly");
    End if;

END;
/
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
Declare
    a Number := 50;
    b Number := 30;
    num_Small Number;
    num_large Number;
Begin
    If a<b then
        num_Small := a;
        num_large := b;
    Else
        num_Small := b;
        num_large := a;
    END IF;
DBMS_output.Put_Line ('Small Number:' || num_small);
DBMS.Output.Put_Line ('Large Number:' || num_large);
END;
/
```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
Begin
    update employees
Set Salary = Salary + (Salary * 0.05)
Where employee_id = 110 and Sales >= target;
If SQL Rowcount > 0 then
    DBMS_output.Put_Line ('Incentive added");
Else
DBMS_ output.Put_Line ('Record not updated');
END IF;
END;
/
```

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
Serveroutput ON;
Create or replace Procedure
Calc_incentive (
    P_emp_id in Number,
    P_Sales in Number.
) IS
    V_incentive Number;
Begin
  If P_Sales >=10000 Then
      V_incentive := 5000;
  Elsif P_sales >= 50000 Then
      V_incentive := 2000;

  Else
     V_incentive := 0;
  End if;
  Update employee
  Set incentive_v_incentive
  where employee_id = P_emp_id;
  If SQL%. Rowcount >0 then
DBMS_output.put_Line (' Record updated Incentive = ' || V_incentive)
  Else
    DBMS_output. PutLine (' NO Record found for Employee ID')
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
Set Serveroutput ON;
Declare
   V_emp_count Number;
   V_vaccancies Number;
   V_remaining Number;
Begin
   Select Count(*) Into V_emp_Coard
   From employee
   where department_id =50;
   V_remaining := V_vaccancies -V_emp_count;
   DBMS_Output.Put_Line('Number of employees in Dept 50')
   If V_remaining >0 then
DBMS_Output.Put_Line ("Vacancies available: "|| V_remaining);
   Else
      DBMS_Output.Put_Line ('No vacancies in Dept 50');
   END IF;
END;
/
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
Set ServerOutput ON;
Declare
    v_dept_id    Number := &dept_id;
    v_emp_count  Number;
    v_total_Post Number := 45;
    v_Vacancies  Number;

BEGIN
    Select Count(*) into v_emp_count
    From employee
    where department_id = v_dept_id;
    v_Vacancies := v_total_posts - v_emp_count;
DBMS_Output.Put_Line('Department ID:' || v_dept_id);
DBMS_Output.Put_Line('Number of Employees:' || v_emp_count);
END;
/
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
Set Serveroutput ON;
Declare
    V_emp_id
employee.employee_id %TYPE;
    V_name
employee.employee_name %TYPE;
    V_job
employee.job_id %.TYPE;
    V_hire_date
employee.hire_data%TYPE;
    V_Salary
employee.Salary %.TYPE;
Cursor emp_cur IS
    Select employee_id, first_name, job_id, hire_date,
  Salary
    From employee;
Begin
    DBMS.output.Put_Line ('Emp_ID Name, Job_ID, Hire_date
END;
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
Set Serveroutput ON;
Declare
    Cursor emp_cur IS
        Select e.employee_id,
               e.first_name,
               d.department_name.
        From employee e
        Join department d
        ON e.department_id = d.department_id;
BEGIN
    DBMS_Output.Put_Line ("Emp_ID  Name  Dept");
    DBMS_Output.Put_Line ('_ _ _ _ _ _ _ _ _ _');
For emp_rec in emp_cur Loop
    DBMS_Output.Put_Line(emp_rec.employee_id|| ' ' ||
    RPAD (emp_rec.first_name, 12)|| ' ' ||
    emp_rec.department_name);
    END LOOP;
END;
/.
```

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
Set Serveroutput ON;
Declare
    Cursor job_cur IS
      Select Job_id, job_title, min_Salary
      From Jobs;
BEGIN
    DBMS_Output.Put_Line ('Job_ID, Job_title
DBMS_Output.Put_Line ('----------')          Min_Salary')

  For Job_rec In Job_cur Loop
    DBMS_Output.Put_Line (RPAD (Job_rec, Job_id 12)
    || ' ' || RPAD (Job_rec, Job_title, 25) || ' ' ||
    Job_rec, min_Salary));
    END LOOP;

  END;
/
```

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
Set Serveroutput ON;
Declare
    Cursor emp_cur IS
        Select e.employee_id,
               e.first_name,
               jh.start_date,
        From employee e
        Join job_history ON
        ON e.employee_id = jh.employee_id;
BEGIN
    DBMS_Output.Put_Line ('Emp_id Name Start_Date');
    DBMS_Output.Put_Line ('----------');
    For emp_rec IN emp_cur Loop
        DBMS_Output.Put_Line (emp_rec.employee_id || ' ' ||
        RPAD (emp_rec.first_name, 12) || ' ' ||
        TOCHAR (emp_rec.start_date, 'DD-MON-YYYY'));
        END LOOP;
END;
```

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```
Begin
    For rec IN (
        SELECT e.employee_id, e.first_name, jh.end_date
        From employees e Join Job_history jh
        ON e.employees_id = jh.employee_id
    ) Loop
        DBMS_output.Put_Line 'ID': ||rec.employee_id||
'Name : ||rec.first_name||
" End Date :|| rec.end date);
End Loop;
);
```

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