Ex.No.: 12		**WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS**
Date:	01/10/2024	

AIM:

Create PL/SQL Blocks to perform the Item Transaction Operations using CURSOR, FUNCTION and PROCEDUERE.

ALGORITHM:

STEP-1: Start.

STEP-2: Create two tables Item Master and Item Trans.

itemmaster(itemid , itemname, stockonhand)

itemtrans(itemid ,itemname ,dateofpurchase ,quantity)

STEP-3: Create a PROCEDURE with id, name and quantity as parameters which make a call to the FUNCTION by passing id, name, dop, and quantity as parameters dop is set as sysdate.

STEP-4: Using FUNCTION fetch each record from the table Item Master using CURSOR inside a Loop statement,

If Item Master's ItemId is equal to the entered ID value then exit the loop otherwise fetch the next record.

```
loop
        fetch master into masterrec
        exit when master%notfound
        if masterrec.itemid=id then
  exit;
 end if;
end loop;
```

STEP-5: If Itemmaster's itemid = id then,

Add the Itemmaster's stockonhand with the given quantity and update the ItemMaster table and insert the Item information into the ItemTrans table.

STEP-6: Else, if the inputed item is not present in the ItemMaster table then insert the new Item in both the tables.

STEP-7: Call the Procedure by passing the Item informations which calls the Function.

STEP-8: Exit.

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
```

<u>PROCEDURE FOR 'IN' PARAMETER – CREATION, EXECUTION</u>

```
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.
```

<u>FACTORIAL OF A NUMBER USING FUNCTION</u> — PROGRAM AND EXECUTION

```
SQL>  create function itfact (a number) return number is
  2  fact number:=1;
  3  b number;
  4  begin
  5  b:=a;
  6  while b>0
  7  loop
  8  fact:=fact*b;
  9  b:=b-1;
 10  end loop;
 11  return(fact);
 12  end;
 13  /

Function created.

SQL> set serveroutput on;

SQL> declare
  2  a number:=7;
  3  f number(10);
  4  begin
  5  f:=itfact(a);
  6  dbms_output.put_line('The factorial of the given number is'||f);
  7 end;
  8  /
The factorial of the given number is 5040
PL/SQL procedure successfully completed.
```

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
CREATE OR REPLACE FUNCTION cal_fac(n NUMBER)
   RETURN NUMBER IS Fac NUMBER:=1;
BEGIN
    IF n=0 or n=1 THEN
        return Fac;
    ELSE
        FOR i IN 1..n LOOP
            fac:=fac*i.
        END LOOP;
        RETURN fac;
    END IF;
  END cal fac;
  DECLARE
        num NUMBER:=5;
        result NUMBER;
  BEGIN
        result:=cal_fac(num);
        DBMS_OUTPUT.PUT_LINE(result).
  END;
```

Program 2

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

TO WRITE A PL/SQL BLOCK TO DISPLAY THE EMPLOYEE ID AND EMPLOYEE NAME WHERE DEPARTMENT NUMBER IS 11 USING EXPLICIT CURSORS

```
declare
cursor cenl is select eid,sal from ssempp where dno=11;
ecode ssempp.eid%type;
esal empp.sal%type;
begin
open cenl;
loop
fetch cenl into ecode,esal;
exit when cenl%notfound;
dbms_output.put_line(' Employee code and employee salary are' || ecode ‘and’|| esal);
```

```
CREATE OR REPLACE PROCEDURE book_info (

        p_book_id IN NUMBER,
        p_author OUT VARCHAR2;
        p_avail_cop IN OUT NUMBER; ) IS
BEGIN
    SELECT author ; avail_cop from library_books where book_id = p_bookid;
   IF p_avail_cop > 0 THEN
            p_avail_cop := p.avail_cop-1;
    ELSE
        DBMS_OUTPUT.PUT_LINE ("No available copies');
    END IF;
  END book info;
  DECLARE
       book_id NUMBER := 2;
       author VARCHAR2 (100),
       avail_cop NUMBER := 1;
  BEGIN
        book info (book_id , author, avail_cop).
        DBMS_OUTPUT.PUT_LINE (book_id || author || avail_cop),
  END;
```

```
end loop;
close cenl;
end;

SQL> /
Employee code and employee salary are 1 and 39000
Employee code and employee salary are 5 and 35000
Employee code and employee salary are 6 and 23000

PL/SQL procedure successfully completed.
```

TO WRITE A PL/SQL BLOCK TO UPDATE THE SALARY BY 5000 WHERE THE JOB IS LECTURER , TO CHECK IF UPDATES ARE MADE USING IMPLICIT CURSORS AND TO DISPLAY THE UPDATED TABLE

```
SQL> declare
county number;
begin
update ssempp set sal=sal+10000 where job='lecturer';
county:= sql%rowcount;
if county > 0 then
dbms_output.put_line('The number of rows are '|| county);
end if;
if sql %found then
dbms_output.put_line('Employee record modification successful');
else if sql%notfound then
dbms_output.put_line('Employee record is not found');
end if;
end if;
end;
/
The number of rows are 3

Employee record modification successful

PL/SQL procedure successfully completed.

SQL> select * from ssempp;

     EID  ENAME              JOB                  SAL       DNO
--------- ------------------ ------------------ --------- ---------
       1  nala               lecturer             44000     11
       2  kala               seniorlecturer       20000     12
       5  ajay               lecturer             40000     11
       6  vijay              lecturer             28000     11
       3  nila               professor            60000     12
```