**EX NO.: 06 CURSOR PROCEDURE FUNCTIONS**

**AIM:**

To write a SQL program to work with cursor, procedure and functions.

**PROCEDURE:**

**Step 1**: Open Run SQL on Command line and connect to SQL

**Step 2:** Then work with database using SQL queries.

**PL/SQL PROCEDURE:**

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

* **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.
* **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

**Syntax for creating procedure:**

CREATE [OR REPLACE] PROCEDURE procedure\_name

[ (parameter [,parameter]) ]

IS

[declaration\_section]

BEGIN

executable\_section

[EXCEPTION

exception\_section]

END [procedure\_name];

**TABLE QUERY:**

create table employee(emp\_id number(5)primary key, emp\_name varchar2(20), city varchar2(20), salary number(7), age number(5));

insert into employee values (1, 'Raju', 'Pdy', 800000, 20);

insert into employee values (2, 'Niteesh', 'Pdy', 790000, 21);

insert into employee values (3, 'Punith', 'AP', 750000, 20);

insert into employee values (4, 'Sidharth', 'MP', 650000, 21);

insert into employee values (5, 'Mantu', 'Delhi', 900000, 22);

**PROGRAM CODE:**

DECLARE

PROCEDURE pro

AS

BEGIN

dbms\_output.put\_line('It is working perfectly!');

END;

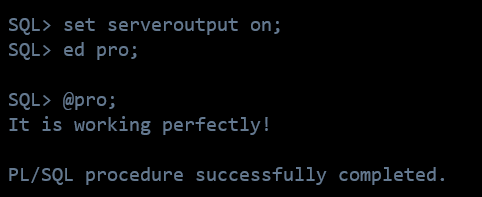
BEGIN

pro();

END;

/

**OUTPUT:**



**PL/SQL – CURSORS:**

A cursor is used to referred to a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

* Implicit Cursors
* Explicit Cursors

**IMPLICIT CURSOR:**

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.

**1 %FOUND**

Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.

**2 %NOTFOUND**

The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

**3 %ISOPEN**

Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.

**4 %ROWCOUNT**

Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

**EXPLICIT CURSOR:**

Explicit cursors are programmer-defined cursors for gaining more control over the context area.

**The syntax for creating an explicit cursor is −**

CURSOR cursor\_name IS select\_statement;

**Working with an explicit cursor includes the following steps −**

* Declaring the cursor for initializing the memory
* Opening the cursor for allocating the memory
* Fetching the cursor for retrieving the data
* Closing the cursor to release the allocated memory

**PROGRAM CODE:**

DECLARE

e\_id employee.emp\_id%type;

e\_name employee.emp\_name%type;

e\_city employee.city%type;

cursor e\_employee is

select emp\_id, emp\_name, city from employee;

begin

open e\_employee;

loop

fetch e\_employee into e\_id, e\_name, e\_city;

exit when e\_employee%notfound;

dbms\_output.put\_line(e\_id || ' ' || e\_name || ' ' || e\_city);

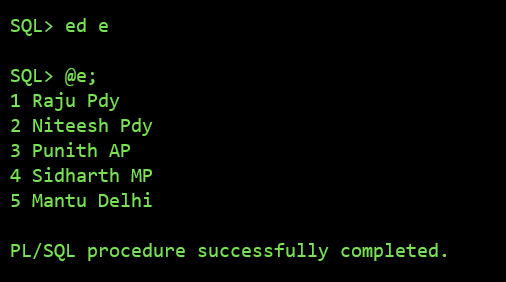
end loop;

close e\_employee;

end;

/

**OUTPUT:**



**PL/SQL FUNCTION:**

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value.

**Syntax to create a function:**

CREATE [OR REPLACE] FUNCTION function\_name [parameters]

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

RETURN return\_datatype

{IS | AS}

BEGIN

< function\_body >

END [function\_name];

**PROGRAM CODE:**

DECLARE

n number;

t number;

FUNCTION func

RETURN number IS

total number(2) := 0;

BEGIN

SELECT count(\*) into total

FROM employee;

RETURN total;

END;

BEGIN

n:=2;

t:=func();

dbms\_output.put\_line(t);

END;

/

**OUTPUT:**

