

Experiment-6

Student Name: Gauri Prabhakar

Branch: 18AITAIML-2

Semester: 7

Subject Name: Advanced Database Management Lab

UID: 18BCS6201

Section/Group: B

Date of Performance: 13th October, 2021

Subject Code: CSP - 434

1. Aim/Overview of the practical:

To Implement PL/SQL programming using Cursors.

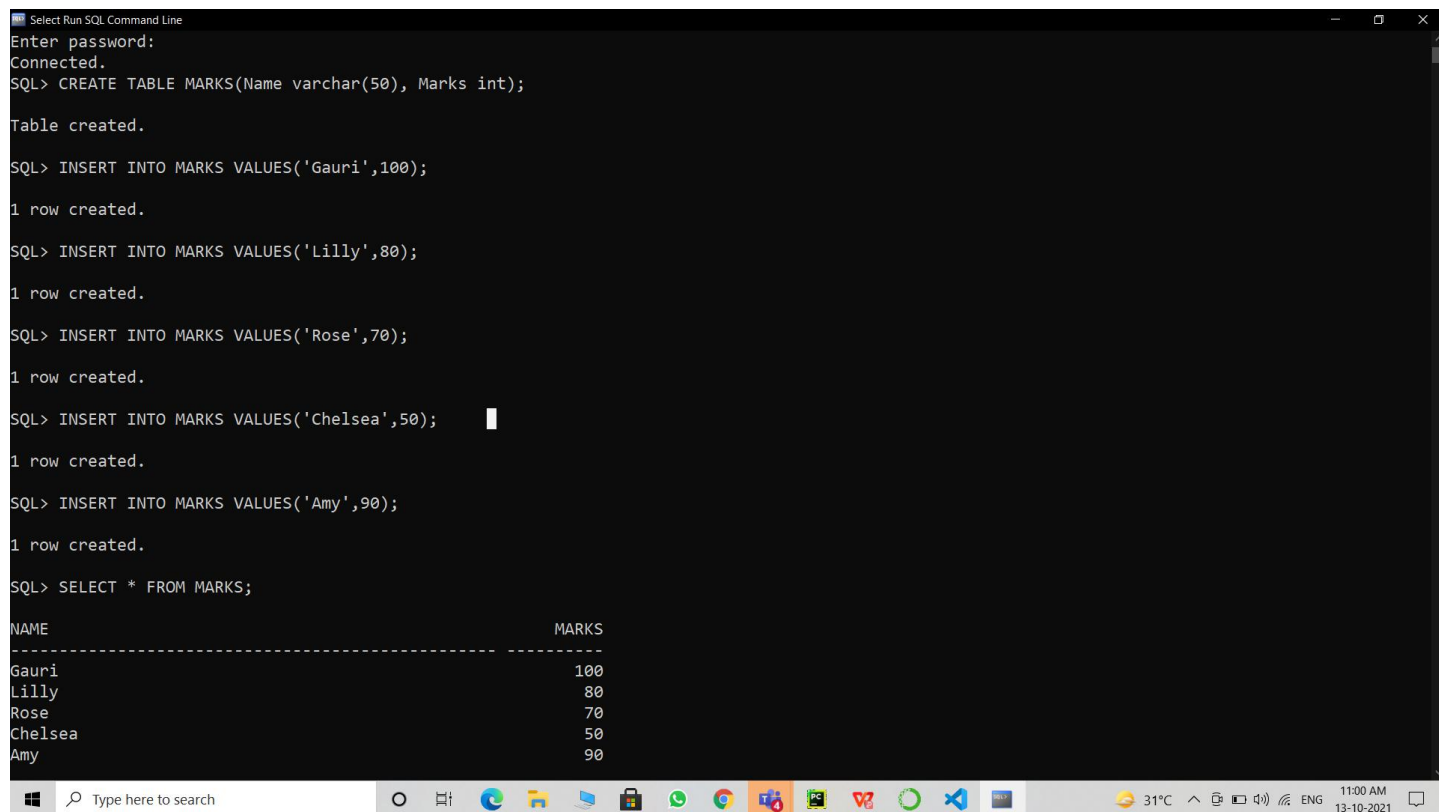
2. Task to be done:

To Implement PL/SQL programming using Cursors.

3. Steps to be followed:

Creating a table MARKS and then returning it:

```
1. CREATE TABLE MARKS(Name varchar(50), Marks int);  
   INSERT INTO MARKS VALUES('Gauri',100);  
   INSERT INTO MARKS VALUES('Lilly',80);  
   INSERT INTO MARKS VALUES('Rose',70);  
   INSERT INTO MARKS VALUES('Chelsea',50);  
   INSERT INTO MARKS VALUES('Amy',90);  
   SELECT * FROM MARKS;
```



The screenshot shows a Windows command prompt window titled "Select Run SQL Command Line". The user has entered a password and is connected to a database. The following SQL commands are executed:

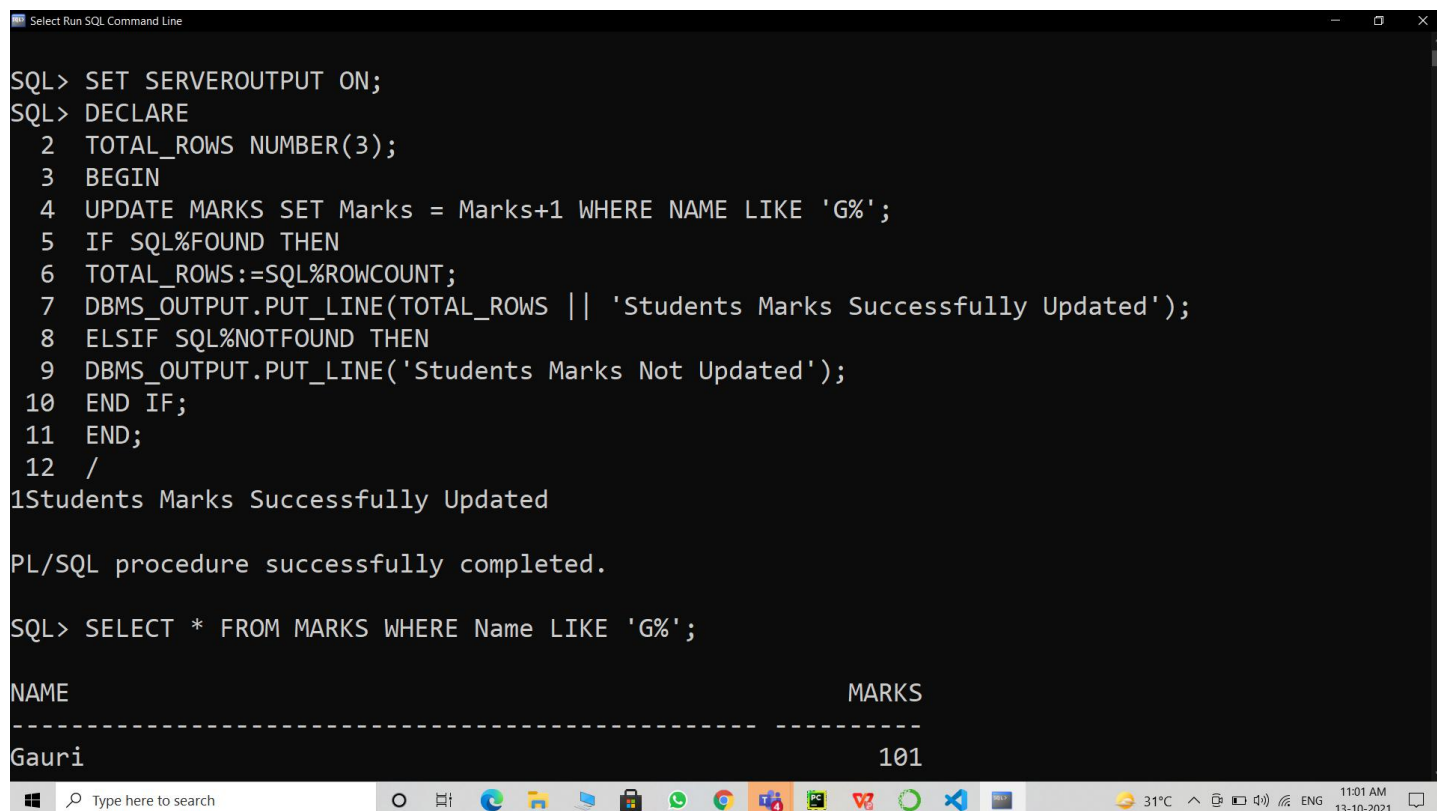
```
SQL> CREATE TABLE MARKS(Name varchar(50), Marks int);  
Table created.  
SQL> INSERT INTO MARKS VALUES('Gauri',100);  
1 row created.  
SQL> INSERT INTO MARKS VALUES('Lilly',80);  
1 row created.  
SQL> INSERT INTO MARKS VALUES('Rose',70);  
1 row created.  
SQL> INSERT INTO MARKS VALUES('Chelsea',50);  
1 row created.  
SQL> INSERT INTO MARKS VALUES('Amy',90);  
1 row created.  
SQL> SELECT * FROM MARKS;
```

The output of the SELECT statement is displayed as follows:

NAME	MARKS
Gauri	100
Lilly	80
Rose	70
Chelsea	50
Amy	90

Implementing IMPLICIT CURSORS and Updating the Marks by 1 for specific tuples which qualify a set condition and then returning the updated table:

```
2. SET SERVEROUTPUT ON;
DECLARE
TOTAL_ROWS NUMBER(3);
BEGIN
UPDATE MARKS SET Marks = Marks+1 WHERE NAME LIKE 'G%';
IF SQL%FOUND THEN
TOTAL_ROWS:=SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE(TOTAL_ROWS || 'Students Marks Successfully Updated');
ELSIF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Students Marks Not Updated');
END IF;
END;
/
SELECT * FROM MARKS WHERE Name LIKE 'G%';
```



The screenshot shows a SQL Command Line window with the following content:

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  TOTAL_ROWS NUMBER(3);
  3  BEGIN
  4  UPDATE MARKS SET Marks = Marks+1 WHERE NAME LIKE 'G%';
  5  IF SQL%FOUND THEN
  6  TOTAL_ROWS:=SQL%ROWCOUNT;
  7  DBMS_OUTPUT.PUT_LINE(TOTAL_ROWS || 'Students Marks Successfully Updated');
  8  ELSIF SQL%NOTFOUND THEN
  9  DBMS_OUTPUT.PUT_LINE('Students Marks Not Updated');
 10  END IF;
 11  END;
 12  /
1Students Marks Successfully Updated

PL/SQL procedure successfully completed.

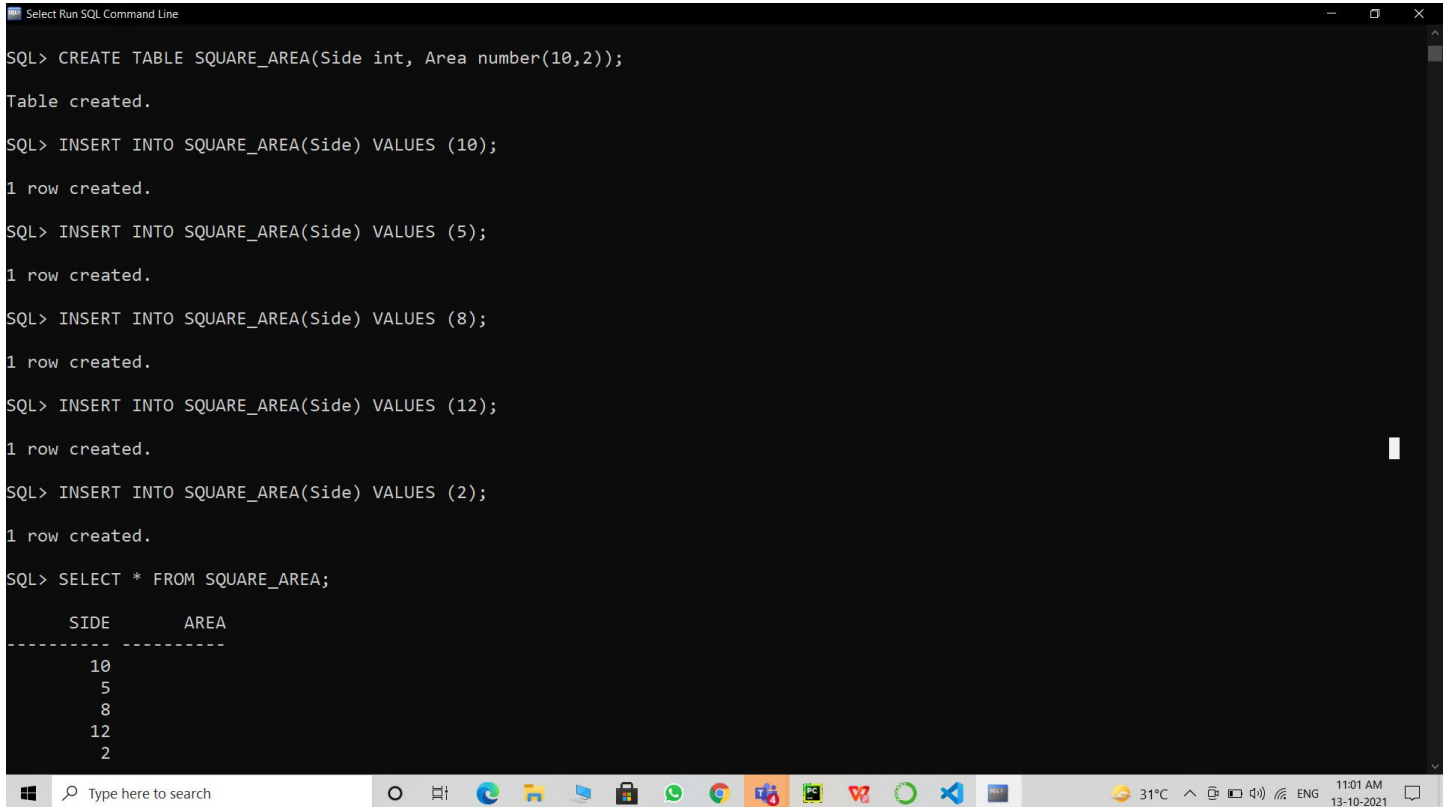
SQL> SELECT * FROM MARKS WHERE Name LIKE 'G%';
```

NAME	MARKS
Gauri	101

The bottom of the screenshot shows a Windows taskbar with various icons and a system tray displaying the temperature (31°C), time (11:01 AM), and date (13-10-2021).

Creating a table 'SQUARE_AREA' and returning it:

```
3. CREATE TABLE SQUARE_AREA(Side int, Area number(10,2));  
INSERT INTO SQUARE_AREA(Side) VALUES (10);  
INSERT INTO SQUARE_AREA(Side) VALUES (5);  
INSERT INTO SQUARE_AREA(Side) VALUES (8);  
INSERT INTO SQUARE_AREA(Side) VALUES (12);  
INSERT INTO SQUARE_AREA(Side) VALUES (2);  
SELECT * FROM SQUARE_AREA;
```



The screenshot shows a SQL Command Line window with the following commands and output:

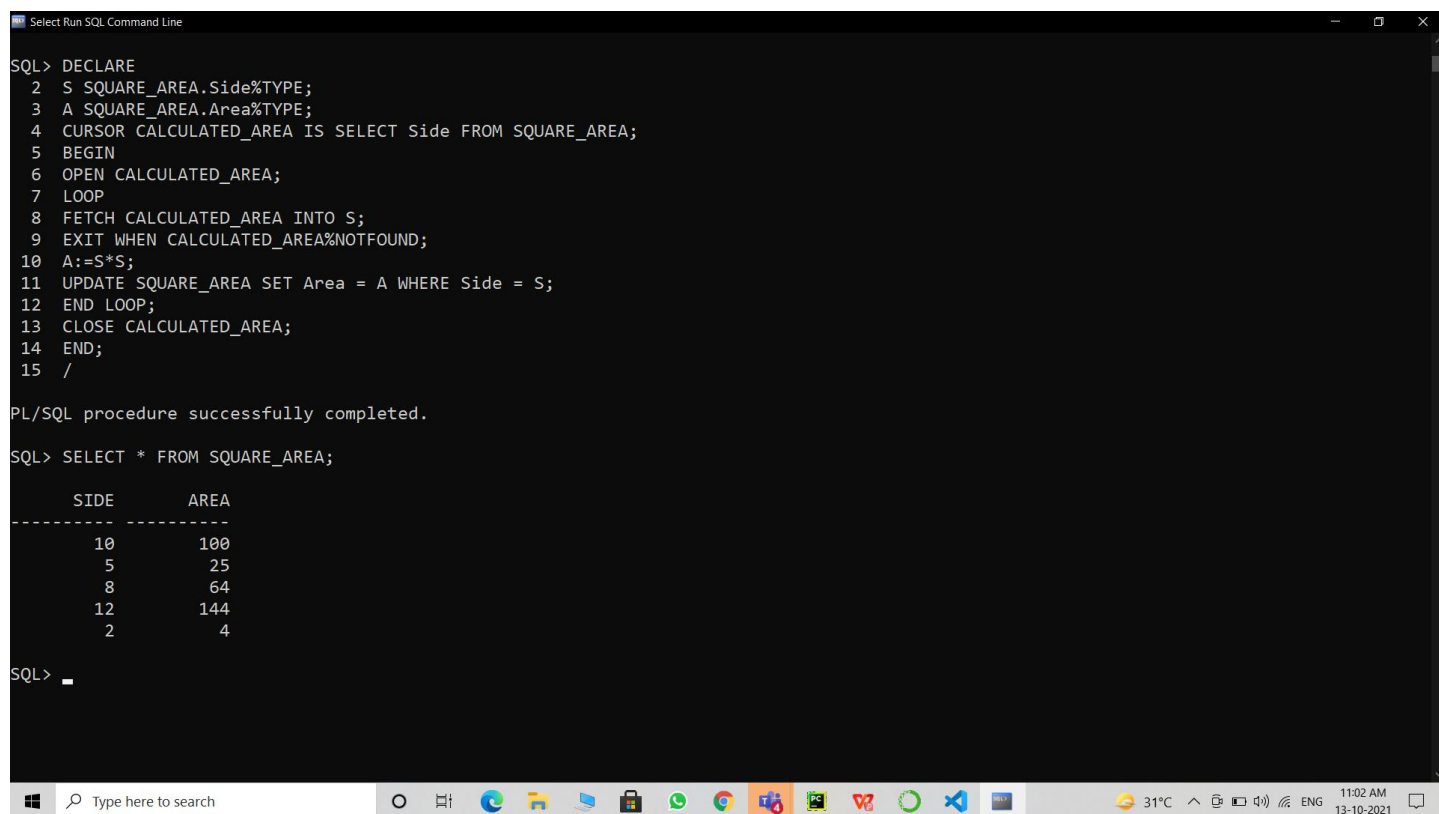
```
SQL> CREATE TABLE SQUARE_AREA(Side int, Area number(10,2));  
Table created.  
SQL> INSERT INTO SQUARE_AREA(Side) VALUES (10);  
1 row created.  
SQL> INSERT INTO SQUARE_AREA(Side) VALUES (5);  
1 row created.  
SQL> INSERT INTO SQUARE_AREA(Side) VALUES (8);  
1 row created.  
SQL> INSERT INTO SQUARE_AREA(Side) VALUES (12);  
1 row created.  
SQL> INSERT INTO SQUARE_AREA(Side) VALUES (2);  
1 row created.  
SQL> SELECT * FROM SQUARE_AREA;
```

SIDE	AREA
10	
5	
8	
12	
2	

Implementing EXPLICIT CURSOR, calculating the AREA and updating it into the 'SQUARE_AREA' table:

4. DECLARE

```
S SQUARE_AREA.Side%TYPE;  
A SQUARE_AREA.Area%TYPE;  
CURSOR CALCULATED_AREA IS SELECT Side FROM SQUARE_AREA;  
BEGIN  
OPEN CALCULATED_AREA;  
LOOP  
FETCH CALCULATED_AREA INTO S;  
EXIT WHEN CALCULATED_AREA%NOTFOUND;  
A:=S*S;  
UPDATE SQUARE_AREA SET Area = A WHERE Side = S;  
END LOOP;  
CLOSE CALCULATED_AREA;  
END;  
/  
SELECT * FROM SQUARE_AREA;
```



The screenshot shows a SQL Command Line window with the following content:

```
SQL> DECLARE  
2 S SQUARE_AREA.Side%TYPE;  
3 A SQUARE_AREA.Area%TYPE;  
4 CURSOR CALCULATED_AREA IS SELECT Side FROM SQUARE_AREA;  
5 BEGIN  
6 OPEN CALCULATED_AREA;  
7 LOOP  
8 FETCH CALCULATED_AREA INTO S;  
9 EXIT WHEN CALCULATED_AREA%NOTFOUND;  
10 A:=S*S;  
11 UPDATE SQUARE_AREA SET Area = A WHERE Side = S;  
12 END LOOP;  
13 CLOSE CALCULATED_AREA;  
14 END;  
15 /  
  
PL/SQL procedure successfully completed.  
  
SQL> SELECT * FROM SQUARE_AREA;
```

SIDE	AREA
10	100
5	25
8	64
12	144
2	4

SQL> _

4. Result/Output/Writing Summary:

- Successfully implemented CURSORS.
- Successfully implemented IMPLICIT CURSORS.
- Successfully implemented EXPLICIT CURSORS.
- Successfully understood the functioning and importance of the above mentioned.

5. Learning outcomes (What I have learnt):

- How to implement CURSORS on SQL Command Line.
- How to implement IMPLICIT CURSORS on SQL Command Line.
- How to implement EXPLICIT CURSORS on SQL Command Line.
- How to implement CURSORS on a table.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			