Ex.No.7	CURSOR

AIM

To implement cursors in DBMS for efficient row-by-row data retrieval and manipulation.

CREATING A TABLES

SQL> CREATE TABLE emp1 (id NUMBER(8), name VARCHAR2(55), basic NUMBER(8,2));

Table created.

SQL> CREATE TABLE cust1 (id NUMBER(5), name VARCHAR2(50), address VARCHAR2(100));

Table created.

INSERTING VALUES INTO TABLE

SQL> INSERT INTO emp1 VALUES (1, 'praveen', 5000); 1

row created.

SQL> INSERT INTO emp1 VALUES (2, 'ram', 6000);

1 row created.

SQL> INSERT INTO emp1 VALUES (3, 'laxman', 7000);

1 row created.

SQL> INSERT INTO cust1 VALUES (101, 'karthik', 'Erode'); 1

row created.

```
SQL> INSERT INTO cust1 VALUES (102, 'jegan', 'Salem');

1 row created.

SQL> INSERT INTO cust1 VALUES (103, 'kavin', 'Erode');

1 row created.

SQL> COMMIT;

Commit complete.
```

IMPLICIT CURSOR
EXAMPLE-1

```
SQL> DECLARE
       total_rows NUMBER(2);
   BEGIN
       UPDATE emp1 SET basic = basic + 500;

       IF SQL%NOTFOUND THEN
          DBMS_OUTPUT.PUT_LINE('No employees updated.');
       ELSIF SQL%FOUND THEN
          total_rows := SQL%ROWCOUNT;
         DBMS_OUTPUT.PUT_LINE(total_rows || ' employees updated.');
      END IF;
  END;
  /

PL/SQL procedure successfully completed.
```

EXAMPLE -2

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
       total_rows NUMBER(2);
   BEGIN
       UPDATE emp1 SET basic = basic + 500;
```

```
      IF SQL%NOTFOUND THEN
         DBMS_OUTPUT.PUT_LINE('No employees updated.');
      ELSIF SQL%FOUND THEN
         total_rows := SQL%ROWCOUNT;
        DBMS_OUTPUT.PUT_LINE(total_rows || ' employees updated.');
     END IF;
  END;
  /
3 employees updated.

PL/SQL procedure successfully completed.
```

EXAMPLE -3

```
SQL> DECLARE
      total_deleted NUMBER(2);
   BEGIN
      DELETE FROM emp1 WHERE basic < 5500;

      IF SQL%NOTFOUND THEN
         DBMS_OUTPUT.PUT_LINE('No employees deleted.');
      ELSE
         total_deleted := SQL%ROWCOUNT;
        DBMS_OUTPUT.PUT_LINE(total_deleted || ' employees deleted.');
     END IF;
  END;
  /
No employees deleted.

PL/SQL procedure successfully completed.
```

EXAMPLE-4

```
SQL> DECLARE
      total_inserted NUMBER(2);
   BEGIN
      INSERT INTO cust1s VALUES (104, 'kamalesh', 'Perundurai'); INSERT
      INTO cust1s VALUES (105, 'sanjay', 'Erode');
```

```
      total_inserted := SQL%ROWCOUNT;
      DBMS_OUTPUT.PUT_LINE(total_inserted || ' customers inserted.');
      COMMIT;
  END;
  /
1 customers inserted.

PL/SQL procedure successfully completed.
```

EXPLICIT CURSOR

EXAMPLE -1

```
SQL> DECLARE
      c_id    cust1.id%TYPE;
      c_name  cust1.name%TYPE;
      c_addr cust1.address%TYPE;

      CURSOR c_cust1 IS
         SELECT id, name, address FROM cust1s;
   BEGIN
      OPEN c_cust1;
     LOOP
        FETCH c_cust1 INTO c_id, c_name, c_addr;
        EXIT WHEN c_cust1%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(c_id || ' ' || c_name || ' ' || c_addr);
     END LOOP;
     CLOSE c_cust1;
  END;
  /

PL/SQL procedure successfully completed.
```

EXAMPLE -2

```
SQL> DECLARE
      c_id    cust1.id%TYPE;
      c_name  cust1.name%TYPE;
      c_addr cust1.address%TYPE;
```

```
      CURSOR c_cust1 IS
         SELECT id, name, address FROM cust1; BEGIN
      OPEN c_cust1;
     LOOP
        FETCH c_cust1 INTO c_id, c_name, c_addr;
        EXIT WHEN c_cust1%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(c_id || ' ' || c_name || ' ' || c_addr);
     END LOOP;
     CLOSE c_cust1;
  END;
  /
101 Karthik Erode
102 Jegan Salem
103 Kavin Erode

PL/SQL procedure successfully completed.
```

EXAMPLE -3

```
SQL> DECLARE
      e_id    emp1.id%TYPE;
      e_name emp1.name%TYPE;
      e_basic emp1.basic%TYPE;

      CURSOR emp_cursor IS
         SELECT id, name, basic FROM emp1 WHERE basic > 5500;

   BEGIN
     OPEN emp_cursor;
     LOOP
        FETCH emp_cursor INTO e_id, e_name, e_basic;
        EXIT WHEN emp_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('ID: ' || e_id || ', Name: ' || e_name || ',
Basic: ' || e_basic);
     END LOOP;
     CLOSE emp_cursor;
```

```
  END;
  /
ID: 1, Name:ram, Basic: 6500
ID: 2, Name: laxman, Basic: 7500

PL/SQL procedure successfully completed.
```

EXAMPLE -4

```
SQL> DECLARE
       c_id    cust1s.id%TYPE;
       c_name cust1s.name%TYPE; c_addr
       cust1s.address%TYPE;

       CURSOR texas_cust1s IS
          SELECT id, name, address FROM cust1s WHERE address = 'Salem';

    BEGIN
      OPEN texas_cust1s;
      LOOP
         FETCH texas_cust1s INTO c_id, c_name, c_addr;
         EXIT WHEN texas_cust1s%NOTFOUND;

         DBMS_OUTPUT.PUT_LINE('Customer Name: ' || c_name || ', Address: '
|| c_addr);
      END LOOP;
      CLOSE texas_cust1s;
  END;
  /
Customer Name: Jegan, Address: Salem

PL/SQL procedure successfully completed.
```

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
Aim,Algorithm,SQL,PL/SQL	30	
Execution and Result	20	
Viva	10	
Total	60	

RESULT

Achieved controlled and optimized data processing using cursors, enabling complex operations with improved precision.