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 employee (
  2     id     NUMBER(5),
  3     name   VARCHAR2(50),
  4     basic  NUMBER(8,2)
  5  );
```

Table created.

```
SQL> CREATE TABLE customers (
  2     id      NUMBER(5),
  3     name    VARCHAR2(50),
  4     address VARCHAR2(100)
  5  );
```

Table created.

INSERTING VALUES INTO TABLE

SQL> INSERT INTO employee (ename, eid, dob, basic, da, total) VALUES ('Kavin', 1, TO_DATE('6-Oct-1990', 'DD-Mon-YYYY'), 25000, NULL, NULL);

1 row created.

SQL>

SQL> INSERT INTO employee (ename, eid, dob, basic, da, total) VALUES ('Arun', 2, TO_DATE('8-Sep-1990', 'DD-Mon-YYYY'), 17000, NULL, NULL);

1 row created.

SQL>

SQL> INSERT INTO employee (ename, eid, dob, basic, da, total) VALUES ('Sankar', 3, TO_DATE('1-Jan-1989', 'DD-Mon-YYYY'), 12000, NULL, NULL);

1 row created.

SQL>

SQL> INSERT INTO employee (ename, eid, dob, basic, da, total) VALUES ('Radha', 4, TO_DATE('10-Apr-1982', 'DD-Mon-YYYY'), 35000, NULL, NULL);

1 row created.

SQL> INSERT INTO customers (id, name, address) VALUES (1, 'John Doe', 'New York');

1 row created.

SQL> INSERT INTO customers (id, name, address) VALUES (2, 'Jane Smith', 'California');

1 row created.

SQL> INSERT INTO customers (id, name, address) VALUES (3, 'Mike Johnson', 'Texas');

1 row created.

SQL> INSERT INTO customers (id, name, address) VALUES (4, 'Emily Davis', 'Florida');

1 row created.

SQL> INSERT INTO customers (id, name, address) VALUES (5, 'Robert Brown', 'Nevada');

1 row created.

IMPLICIT CURSOR

EXAMPLE-1

```
SQL> DECLARE
  2     total_rows NUMBER(2);
  3  BEGIN
  4     UPDATE employee SET basic = basic + 500;
  5
  6     IF SQL%NOTFOUND THEN
  7        DBMS_OUTPUT.PUT_LINE('No employees updated.');
  8     ELSIF SQL%FOUND THEN
  9        total_rows := SQL%ROWCOUNT;
 10         DBMS_OUTPUT.PUT_LINE(total_rows || ' employees updated.');
 11     END IF;
 12  END;
 13  /
```

PL/SQL procedure successfully completed.

EXAMPLE -2

```
SQL> DECLARE
  2    v_rows_updated NUMBER(2);
  3  BEGIN
  4    UPDATE employee
  5    SET basic = basic + 1000;
```

```
 6
 7    IF SQL%FOUND THEN
 8       v_rows_updated := SQL%ROWCOUNT;
 9       DBMS_OUTPUT.PUT_LINE(v_rows_updated || ' employees salary updated.');
10    ELSE
11       DBMS_OUTPUT.PUT_LINE('No employees found.');
12    END IF;
13  END;
14  /
```

PL/SQL procedure successfully completed.

EXAMPLE-3

```
SQL> DECLARE
 2  BEGIN
 3    INSERT INTO employee (ename, eid, dob, basic, da, total)
 4    VALUES ('Priya', 6, TO_DATE('5-May-1995','DD-Mon-YYYY'), 28000, NULL,
NULL);
 5
 6    IF SQL%FOUND THEN
 7       DBMS_OUTPUT.PUT_LINE('New employee inserted.');
 8    ELSE
 9       DBMS_OUTPUT.PUT_LINE('Insert failed.');
10    END IF;
11  END;
12  /
```

PL/SQL procedure successfully completed.

EXPLICIT CURSOR

EXAMPLE -1

```
SQL> DECLARE
```

```
 2     c_id    customers.id%TYPE;
 3     c_name  customers.name%TYPE;
 4     c_addr  customers.address%TYPE;
 5
 6     CURSOR c_customers IS
 7        SELECT id, name, address FROM customers;
 8  BEGIN
 9     OPEN c_customers;
10      LOOP
11        FETCH c_customers INTO c_id, c_name, c_addr;
12        EXIT WHEN c_customers%NOTFOUND;
13
14        DBMS_OUTPUT.PUT_LINE(c_id || ' ' || c_name || ' ' || c_addr);
15     END LOOP;
16     CLOSE c_customers;
17  END;
18  /
```

PL/SQL procedure successfully completed.

EXAMPLE -2

```
SQL> DECLARE
 2    c_id customers.id%TYPE;
 3    c_name customers.name%TYPE;
 4    c_addr customers.address%TYPE;
 5
 6    CURSOR c_customers IS
 7      SELECT id, name, address FROM customers;
 8  BEGIN
 9   OPEN c_customers;
10    LOOP
11      FETCH c_customers INTO c_id, c_name, c_addr;
12      EXIT WHEN c_customers%NOTFOUND;
13      DBMS_OUTPUT.PUT_LINE('ID: ' || c_id || ', Name: ' || c_name || ', Address: ' ||
c_addr);
14    END LOOP;
15    CLOSE c_customers;
```

```
 16  END;
 17  /
```

PL/SQL procedure successfully completed.

EXAMPLE -3

```
SQL> DECLARE
  2    v_id customers.id%TYPE;
  3    v_address customers.address%TYPE;
  4
  5    CURSOR c_customers IS
  6       SELECT id, address FROM customers;
  7  BEGIN
  8    OPEN c_customers;
  9    LOOP
 10       FETCH c_customers INTO v_id, v_address;
 11       EXIT WHEN c_customers%NOTFOUND;
 12
 13       UPDATE customers
 14       SET address = v_address || ', USA'
 15       WHERE id = v_id;
 16    END LOOP;
 17    CLOSE c_customers;
 18
 19    DBMS_OUTPUT.PUT_LINE('Address updated for all customers.');
 20  END;
 21  /
```

PL/SQL procedure successfully completed.

EXAMPLE -4

```
SQL> DECLARE
  2    v_id customers.id%TYPE;
  3
  4    CURSOR c_customers IS
  5       SELECT id FROM customers WHERE id > 3;
  6  BEGIN
  7    OPEN c_customers;
  8    LOOP
  9       FETCH c_customers INTO v_id;
 10       EXIT WHEN c_customers%NOTFOUND;
 11
 12       DELETE FROM customers
 13       WHERE id = v_id;
 14    END LOOP;
 15    CLOSE c_customers;
 16
 17    DBMS_OUTPUT.PUT_LINE('Customers with ID > 3 deleted.');
 18  END;
 19  /

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.