EX.NO: 10

TRIGGERS AND CURSORS

AIM

To learn and implement Triggers and Cursors in Oracle SQL for the **Online Food Ordering System**, using the **Custome_Infom** table for customer management.

TABLE STRUCTURE (Custome_Infom)

Column	**Type**	**Description**
CID	NUMBER	Customer ID
NAME	VARCHAR2(30)	Customer Name
EMAIL	VARCHAR2(50)	Customer Email
PHONE	NUMBER	Contact Number
ADDRESS	VARCHAR2(50)	City

TRIGGERS

1. Before Insert Trigger – Validate Email Format

```
CREATE OR REPLACE TRIGGER trg_validate_email
BEFORE INSERT ON Custome_Infom
FOR EACH ROW
BEGIN
   IF NOT REGEXP_LIKE(:NEW.email,
       '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')
   THEN

      RAISE_APPLICATION_ERROR(-20001, 'Invalid Email Format!');

   END IF;

END;

/
```

Test

```
INSERT INTO Custome_Infom
VALUES(101,'Arjun','arjun@gmail.com',9876543210,'Chennai');

1 row created.

INSERT INTO Custome_Infom VALUES(102,'Kavi','kavi@@mail',9876543111,'Erode');

ERROR: ORA-20001: Invalid Email Format!
```

2. After Insert Trigger – Log Newly Registered Customers

```
CREATE OR REPLACE TRIGGER trg_after_insert_customer

AFTER INSERT ON Custome_Infom

FOR EACH ROW

BEGIN

   INSERT INTO Customer_Log(cid, name, log_time)

   VALUES(:NEW.cid, :NEW.name, SYSDATE);

END;

/
```

Test Output

```
INSERT INTO Custome_Infom
VALUES(103,'Priya','priya@gmail.com',9876542222,'Coimbatore');

1 row created.
```

```
SELECT * FROM Customer_Log;
```

```
103    Priya    02-DEC-25
```

3. Before Update Trigger – Prevent Duplicate Phone Numbers

```
CREATE OR REPLACE TRIGGER trg_unique_phone

BEFORE UPDATE ON Custome_Infom

FOR EACH ROW

DECLARE

   v_count NUMBER;

BEGIN

   SELECT COUNT(*) INTO v_count

   FROM Custome_Infom

   WHERE phone = :NEW.phone AND cid != :OLD.cid;

   IF v_count > 0 THEN

      RAISE_APPLICATION_ERROR(-20002, 'Phone Number Already Exists!');

   END IF;

END;

/
```

Test

```
UPDATE Custome_Infom SET phone = 9876543210 WHERE cid = 104;
```

```
ERROR: ORA-20002: Phone Number Already Exists!
```

4. After Delete Trigger – Log Deleted Customers

```
CREATE OR REPLACE TRIGGER trg_delete_customer

AFTER DELETE ON Custome_Infom

FOR EACH ROW
```

```
BEGIN
   INSERT INTO Customer_Log(cid, name, log_time)
   VALUES(:OLD.cid, :OLD.name, SYSDATE);
END;
/
```

Test

```
DELETE FROM Custome_Infom WHERE cid = 101;
1 row deleted.
SELECT * FROM Customer_Log;
101   Arjun   02-DEC-25
```

5. Auto-Generate CID Using Cursor

```
CREATE OR REPLACE TRIGGER trg_auto_cid
BEFORE INSERT ON Custome_Infom
FOR EACH ROW
DECLARE
   CURSOR c1 IS SELECT MAX(cid) FROM Custome_Infom;
   v_max NUMBER;
BEGIN
   OPEN c1;
   FETCH c1 INTO v_max;
   CLOSE c1;
   IF v_max IS NULL THEN
      v_max := 100;
   END IF;
   :NEW.cid := v_max + 1;
END;
/
```

Test

```
INSERT INTO Custome_Infom(name,email,phone,address)
VALUES('Naveen','naveen@gmail.com',9876543333,'Salem');
CID assigned = 105
```

6. After Update Trigger – Track Address Changes

```
CREATE OR REPLACE TRIGGER trg_address_change
AFTER UPDATE OF address ON Custome_Infom
FOR EACH ROW
BEGIN
  INSERT INTO Customer_Log(cid, name, log_time)
   VALUES(:NEW.cid,
      'Address changed from '|| :OLD.address ||' to '|| :NEW.address,
      SYSDATE);
END;
/
```

Test Output

```
UPDATE Custome_Infom SET address='Madurai' WHERE cid=103;
1 row updated.
Customer_Log:
103   Address changed from Coimbatore to Madurai   02-DEC-25
```

CURSORS

1. Simple Cursor – Display Customer Names

```
SET SERVEROUTPUT ON;

DECLARE

   CURSOR c1 IS SELECT name FROM Custome_Infom;

   v_name Custome_Infom.name%TYPE;

BEGIN

   OPEN c1;

   LOOP

      FETCH c1 INTO v_name;

      EXIT WHEN c1%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('Customer: ' || v_name);

   END LOOP;

   CLOSE c1;

END;

/
```

Output

```
Customer: Arjun

Customer: Priya

Customer: Naveen
```

2. Cursor FOR Loop – Display CID and Name

```
BEGIN

   FOR r IN (SELECT cid, name FROM Custome_Infom) LOOP

      DBMS_OUTPUT.PUT_LINE(r.cid || ' - ' || r.name);
   END LOOP;
END;
/
```

3. Cursor With Parameter – Customers From a Specific City

```
DECLARE
   CURSOR c1(p_city VARCHAR2) IS
      SELECT name FROM Custome_Infom WHERE address = p_city;
   v_name VARCHAR2(30);
BEGIN
   OPEN c1('Erode');
   LOOP
      FETCH c1 INTO v_name;
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Customer from Erode: ' || v_name);
   END LOOP;
   CLOSE c1;
END;
/
```

4. Cursor to Count Total Customers

```
DECLARE
   CURSOR c1 IS SELECT cid FROM Custome_Infom;
   v_temp NUMBER;
   total NUMBER := 0;
BEGIN
```

```
   OPEN c1;
   LOOP
      FETCH c1 INTO v_temp;
      EXIT WHEN c1%NOTFOUND;
      total := total + 1;
   END LOOP;
   CLOSE c1;
   DBMS_OUTPUT.PUT_LINE('Total Customers = ' || total);
END;
/
```

5. Cursor With Multiple Columns – Display Name & Email

```
DECLARE
   CURSOR c1 IS SELECT name, email FROM Custome_Infom;
   v_name VARCHAR2(30);
   v_email VARCHAR2(50);
BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO v_name, v_email;
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_name || ' => ' || v_email);
   END LOOP;
   CLOSE c1;
END;
```

```
/
```

6. Cursor to Find Longest Address (Longest City Name)

```
DECLARE
   CURSOR c1 IS SELECT address FROM Custome_Infom;
   v_addr VARCHAR2(50);
   longest VARCHAR2(50) := '';
BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO v_addr;
      EXIT WHEN c1%NOTFOUND;
      IF LENGTH(v_addr) > LENGTH(longest) THEN
         longest := v_addr;
      END IF;
   END LOOP;
   CLOSE c1;
   DBMS_OUTPUT.PUT_LINE('Longest Address = ' || longest);
END;
/
```

OBS	**/10**
COE	**/30**
RECORD	**/10**
VIVA	**/10**
TOTAL	**/60**

RESULT

Thus, all triggers and cursors for the **Custome_Infom** table were successfully implemented and executed in the Online Food Ordering System.