EXERCISE 7: TRIGGERS & CURSORS

Name : Jagtap Mahesh
Reg No. 24MCS1017

`

Answer all questions.

1. Create a trigger named display_salary_changes. The trigger should fire whenever there is a delete or insert or update on the customers table. The difference in salary should be computed and displayed.

Assume that the table customers contains the fields id, name, age, address, salary.

```
CREATE TABLE customers_24mcs1017 (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(20),
  age NUMBER,
  address VARCHAR2(20),
  salary NUMBER
);

INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (101, 'Mahesh', 30, 'Delhi', 50600);

INSERT INTO customers_24mcs1017(id, name, age, address, salary)
VALUES (102, 'Rohit', 28, 'Pune', 60500);

INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (103, 'Akshay', 35, 'Mumbai', 70500);



CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers_24mcs1017
FOR EACH ROW
DECLARE
  sal_diff NUMBER;
BEGIN
  IF INSERTING THEN
    dbms_output.put_line('New salary: ' || :NEW.salary);
  ELSIF UPDATING THEN
    sal_diff := :NEW.salary - :OLD.salary;
    dbms_output.put_line('Old salary: ' || :OLD.salary);
    dbms_output.put_line('New salary: ' || :NEW.salary);
    dbms_output.put_line('Salary difference: ' || sal_diff);
  ELSIF DELETING THEN
    dbms_output.put_line('Old salary: ' || :OLD.salary);
  END IF;
END;
```

```
/
```

```
SQL> CREATE OR REPLACE TRIGGER display_salary_changes
  2  BEFORE DELETE OR INSERT OR UPDATE ON customers_24mcs1017
  3  FOR EACH ROW
  4  DECLARE
  5      sal_diff NUMBER;
  6  BEGIN
  7      IF INSERTING THEN
  8          dbms_output.put_line('New salary: ' || :NEW.salary);
  9      ELSIF UPDATING THEN
 10          sal_diff := :NEW.salary - :OLD.salary;
 11          dbms_output.put_line('Old salary: ' || :OLD.salary);
 12          dbms_output.put_line('New salary: ' || :NEW.salary);
 13          dbms_output.put_line('Salary difference: ' || sal_diff);
 14      ELSIF DELETING THEN
 15          dbms_output.put_line('Old salary: ' || :OLD.salary);
 16      END IF;
 17  END;
 18  /

Trigger created.
```

```
SQL> SET SERVEROUTPUT ON;
SQL> INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
  2  VALUES (104, 'Rajesh', 30, 'Hydrabad', 29000);
New salary: 29000

1 row created.

SQL> UPDATE customers_24mcs1017
  2  SET salary = 70000
  3  WHERE id = 101;
Old salary: 50600
New salary: 70000
Salary difference: 19400

1 row updated.

SQL> DELETE FROM customers_24mcs1017
  2  WHERE id = 103;
Old salary: 70500

1 row deleted.
```

2. Create a trigger named display_semester_changes. The trigger should fire whenever a student

semester value is changed in the student table. Assume that the student table contains the fields regno, name, age, dept, semester. Display the old and new values in the command line.

```
CREATE TABLE student_24mcs1017 (
   regno     NUMBER PRIMARY KEY,
   name      VARCHAR2(20),
   age       NUMBER,
   dept      VARCHAR2(20),
   semester   NUMBER
);

INSERT INTO student_24mcs1017 (regno, name, age, dept, semester)
VALUES (1, 'Mahesh', 20, 'Computer Science', 3);

INSERT INTO student_24mcs1017 (regno, name, age, dept, semester)
VALUES (2, 'Amit', 21, 'Mathematics', 4);

CREATE OR REPLACE TRIGGER display_semester_changes
AFTER UPDATE OF semester
ON student_24mcs1017
FOR EACH ROW
BEGIN
   IF :OLD.semester <> :NEW.semester THEN
      DBMS_OUTPUT.PUT_LINE('Semester changed for Student ' || :NEW.name || ' (RegNo:
' || :NEW.regno || ')');
      DBMS_OUTPUT.PUT_LINE('Old Semester: ' || :OLD.semester || ', New Semester: ' ||
:NEW.semester);
   END IF;
END;
/


UPDATE student_24mcs1017
SET semester = 5
WHERE regno = 1;
```

```
SQL> CREATE OR REPLACE TRIGGER display_semester_changes
  2  AFTER UPDATE OF semester
  3  ON student_24mcs1017
  4  FOR EACH ROW
  5  BEGIN
  6      IF :OLD.semester <> :NEW.semester THEN
  7          DBMS_OUTPUT.PUT_LINE('Semester changed for Student ' || :NEW.name || ' (RegNo: ' || :NEW.regno || ')');
  8          DBMS_OUTPUT.PUT_LINE('Old Semester: ' || :OLD.semester || ', New Semester: ' || :NEW.semester);
  9      END IF;
 10  END;
 11  /

Trigger created.

SQL> UPDATE student_24mcs1017
  2  SET semester = 5
  3  WHERE regno = 1;
Semester changed for Student Mahesh (RegNo: 1)
Old Semester: 3, New Semester: 5

1 row updated.
```

3. Demonstrate an example for implicit cursor – ROWCOUNT.

```
DECLARE
  v_rows_affected NUMBER;
BEGIN
INSERT INTO customers_24mcs1017(id, name, age, address, salary)
VALUES (105, 'Virat', 32, 'Pune', 30500);

  v_rows_affected := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows inserted: ' || v_rows_affected);

  UPDATE customers_24mcs1017 SET salary = salary * 1.1 ;
  v_rows_affected := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' || v_rows_affected);

  -- Delete statement
  DELETE FROM customers_24mcs1017 WHERE address='Pune' ;
  v_rows_affected := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows deleted: ' || v_rows_affected);

  FOR rec IN (SELECT * FROM customers_24mcs1017) LOOP
    DBMS_OUTPUT.PUT_LINE('Emp ID: ' || rec.id || ', Emp Name: ' || rec.name || ', Salary:
' || rec.salary);
  END LOOP;
END;
/
```

```
SQL> DECLARE
  2      v_rows_affected NUMBER;
  3  BEGIN
  4  INSERT INTO customers_24mcs1017(id, name, age, address, salary)
  5  VALUES (105, 'Virat', 32, 'Pune', 30500);
  6
  7      v_rows_affected := SQL%ROWCOUNT;
  8      DBMS_OUTPUT.PUT_LINE('Number of rows inserted: ' || v_rows_affected);
  9
 10      UPDATE customers_24mcs1017 SET salary = salary * 1.1 ;
 11      v_rows_affected := SQL%ROWCOUNT;
 12      DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' || v_rows_affected);
 13
 14      -- Delete statement
 15      DELETE FROM customers_24mcs1017 WHERE address='Pune' ;
 16      v_rows_affected := SQL%ROWCOUNT;
 17      DBMS_OUTPUT.PUT_LINE('Number of rows deleted: ' || v_rows_affected);
 18
 19      FOR rec IN (SELECT * FROM customers_24mcs1017) LOOP
 20          DBMS_OUTPUT.PUT_LINE('Emp ID: ' || rec.id || ', Emp Name: ' || rec.name || ', Salary: ' || rec.salary);
 21      END LOOP;
 22  END;
 23  /
Number of rows inserted: 1
Number of rows updated: 4
Number of rows deleted: 2
Emp ID: 101, Emp Name: Mahesh, Salary: 55660
Emp ID: 103, Emp Name: Akshay, Salary: 77550

PL/SQL procedure successfully completed.
```

4. Create an explicit cursor named c_customers and fetch the id, name and address of all customers in the customer table using the cursor.

```
DECLARE
  CURSOR c_customers IS
    SELECT id, name, address
    FROM customers_24mcs1017;

  v_emp_id customers_24mcs1017.id%TYPE;
  v_emp_name customers_24mcs1017.name%TYPE;
  v_address customers_24mcs1017.address%TYPE;
BEGIN
  OPEN c_customers;
  LOOP
    FETCH c_customers INTO v_emp_id, v_emp_name, v_address;
    EXIT WHEN c_customers%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ', Address: '
|| v_address);
  END LOOP;
  CLOSE c_customers;
END;
/
```

```
SQL> DECLARE
  2      CURSOR c_customers IS
  3          SELECT id, name, address
  4          FROM customers_24mcs1017;
  5
  6      v_emp_id customers_24mcs1017.id%TYPE;
  7      v_emp_name customers_24mcs1017.name%TYPE;
  8      v_address customers_24mcs1017.address%TYPE;
  9  BEGIN
 10      OPEN c_customers;
 11      LOOP
 12          FETCH c_customers INTO v_emp_id, v_emp_name, v_address;
 13          EXIT WHEN c_customers%NOTFOUND;
 14          DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ', Address: ' || v_address);
 15      END LOOP;
 16      CLOSE c_customers;
 17  END;
 18  /
ID: 101, Name: Mahesh, Address: Delhi
ID: 103, Name: Akshay, Address: Mumbai

PL/SQL procedure successfully completed.
```

5. Create an explicit cursor named c_customers and fetch the details of all customers in the customer table whose age is greater than 50 using the cursor.

```
INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (106, 'Aditya', 56, 'mumbai', 82200);

INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (107, 'Vikas', 52, 'Chennai', 73200);

DECLARE
  CURSOR c_customers IS
    SELECT id, name, age, address
    FROM customers_24mcs1017
    WHERE age > 50;

  v_emp_id customers_24mcs1017.id%TYPE;
  v_emp_name customers_24mcs1017.name%TYPE;
  v_age customers_24mcs1017.age%TYPE;
  v_address customers_24mcs1017.address%TYPE;
BEGIN
  OPEN c_customers;
  LOOP
    FETCH c_customers INTO v_emp_id, v_emp_name, v_age, v_address;
    EXIT WHEN c_customers%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ', Age: ' ||
v_age || ', Address: ' || v_address);
  END LOOP;

  CLOSE c_customers;
END;
/
```

```
SQL> DECLARE
  2      CURSOR c_customers IS
  3          SELECT id, name, age, address
  4          FROM customers_24mcs1017
  5          WHERE age > 50;
  6
  7      v_emp_id customers_24mcs1017.id%TYPE;
  8      v_emp_name customers_24mcs1017.name%TYPE;
  9      v_age customers_24mcs1017.age%TYPE;
 10      v_address customers_24mcs1017.address%TYPE;
 11  BEGIN
 12      OPEN c_customers;
 13      LOOP
 14          FETCH c_customers INTO v_emp_id, v_emp_name, v_age, v_address;
 15          EXIT WHEN c_customers%NOTFOUND;
 16
 17          DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ', Age: ' || v_age || ', Address: ' || v_address);
 18      END LOOP;
 19
 20      CLOSE c_customers;
 21  END;
 22  /
ID: 106, Name: Aditya, Age: 56, Address: mumbai
ID: 107, Name: Vikas, Age: 52, Address: Chennai

PL/SQL procedure successfully completed.
```

6. Create an explicit cursor named c_customers and fetch the details of all customers who are minors.

```
INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (108, 'Raju', 16, 'Banglore', 12000);

INSERT INTO customers_24mcs1017 (id, name, age, address, salary)
VALUES (109, 'Abhi', 15, 'Thane', 9300);

DECLARE
   CURSOR c_customers IS
      SELECT id, name, age, address
      FROM customers_24mcs1017
      WHERE age < 18;

   -- Variables to hold fetched data
   v_emp_id customers_24mcs1017.id%TYPE;
   v_emp_name customers_24mcs1017.name%TYPE;
   v_age customers_24mcs1017.age%TYPE;
   v_address customers_24mcs1017.address%TYPE;
BEGIN
   OPEN c_customers;

   LOOP
      FETCH c_customers INTO v_emp_id, v_emp_name, v_age, v_address;
      EXIT WHEN c_customers%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ',
Age: ' || v_age || ', Address: ' || v_address);
   END LOOP;
   CLOSE c_customers;
END;
/
```

```
SQL> DECLARE
  2      CURSOR c_customers IS
  3          SELECT id, name, age, address
  4          FROM customers_24mcs1017
  5          WHERE age < 18;
  6
  7      -- Variables to hold fetched data
  8      v_emp_id customers_24mcs1017.id%TYPE;
  9      v_emp_name customers_24mcs1017.name%TYPE;
 10      v_age customers_24mcs1017.age%TYPE;
 11      v_address customers_24mcs1017.address%TYPE;
 12  BEGIN
 13      OPEN c_customers;
 14
 15      LOOP
 16          FETCH c_customers INTO v_emp_id, v_emp_name, v_age, v_address;
 17          EXIT WHEN c_customers%NOTFOUND;
 18
 19          DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' || v_emp_name || ', Age: ' || v_age || ', Address: ' || v_address);
 20      END LOOP;
 21      CLOSE c_customers;
 22  END;
 23  /
ID: 108, Name: Raju, Age: 16, Address: Banglore
ID: 109, Name: Abhi, Age: 15, Address: Thane

PL/SQL procedure successfully completed.
```