Assignment No.7

1. Write a update, delete trigger on clientmstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit_trade table. (separate implementation using both row and statement triggers)

```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE TABLE clientmstr (
  2     c_id INT PRIMARY KEY,
  3     client_name VARCHAR(50),
  4     client_address VARCHAR(250)
  5  );
Table created.
SQL> -- Insert sample data into EMP table
SQL> INSERT INTO clientmstr VALUES (1, 'Soham', 'Pen');
1 row created.
SQL> INSERT INTO clientmstr VALUES (2, 'Hari', 'Yavatmal');
1 row created.
SQL> INSERT INTO clientmstr VALUES (3, 'Vedant', 'Pune');
1 row created.
SQL> INSERT INTO clientmstr VALUES (4, 'Mrunalini', 'Nigadi');
1 row created.
SQL> INSERT INTO clientmstr VALUES (5, 'Vaishnavi', 'Mahabaleshwar');
1 row created.
SQL> CREATE TABLE audit_trade (
  2     old_c_id INT,
  3     old_client_name VARCHAR(100),
  4     old_client_address VARCHAR(255),
  5     action VARCHAR(10)
  6  );
```

```
Table created.

SQL> CREATE OR REPLACE TRIGGER update_clientmstr_trigger
  2  BEFORE UPDATE OR DELETE ON clientmstr
  3  FOR EACH ROW
  4  BEGIN
  5     INSERT INTO audit_trade VALUES (:OLD.c_id, :OLD.client_name, :OLD.client_address, 'UPDATE');
  6  END;
  7  /
Trigger created.

SQL> UPDATE clientmstr set client_address='KP' where c_id=3;

1 row updated.

SQL> select *from audit_trade;

  OLD_C_ID
----------
OLD_CLIENT_NAME
--------------------------------------------------------------------------------
OLD_CLIENT_ADDRESS
--------------------------------------------------------------------------------
ACTION
----------
         3
Vedant
Pune
UPDATE

SQL> delete clientmstr where c_id=1;

1 row deleted.

SQL> select *from audit_trade;

  OLD_C_ID
----------
```

OLD_CLIENT_NAME

OLD_CLIENT_ADDRESS

ACTION

3

Vedant

Pune

UPDATE

3. Write a Database trigger for following requirements:

Employee salary of last three month is stored in the emp_sal table.

emp_sal(emp_no, sal1,sal2,sal3)

before inserting salary into emp_sal table, if salary of employee in any of the last three month is greater than Rs. 50,000/- then entry of average salary along with emp_no needs to be inserted into new table emp_new(emp_no, avg_sal).

INPUT CODE AND OUTPUT:

```
SQL> CREATE TABLE emp_sal (
  2     emp_no INT,
  3     sal1 NUMBER(10, 2),
  4     sal2 NUMBER(10, 2),
  5     sal3 NUMBER(10, 2)
  6  );
```

Table created.

```
SQL> CREATE TABLE emp_new (
  2     emp_no INT,
  3     avg_sal NUMBER(10, 2)
SQL> CREATE OR REPLACE TRIGGER insert_emp_sal_trigger
  2  BEFORE INSERT ON emp_sal
```

```
 3  FOR EACH ROW
 4  DECLARE
 5     avg_salary NUMBER(10, 2);
 6  BEGIN
 7     avg_salary := ( :NEW.sal1 + :NEW.sal2 + :NEW.sal3 ) / 3;
 8     IF :NEW.sal1 > 50000 OR :NEW.sal2 > 50000 OR :NEW.sal3 > 50000 THEN
 9        -- Insert into emp_new table
10        INSERT INTO emp_new (emp_no, avg_sal) VALUES (:NEW.emp_no, avg_salary);
11     END IF;
12  END;
13  /
Trigger created.
SQL> INSERT INTO emp_sal  VALUES (1, 45000, 48000, 49000);
1 row created.
SQL> INSERT INTO emp_sal  VALUES (2, 55000, 56000, 57000);
1 row created.
SQL> INSERT INTO emp_sal  VALUES (3, 50000, 52000, 53000);
1 row created.
SQL> INSERT INTO emp_sal  VALUES (4, 49000, 48000, 51000);
1 row created.
SQL> SELECT * FROM emp_sal;
   EMP_NO      SAL1      SAL2      SAL3
---------- ---------- ---------- ----------
       1     45000     48000     49000
       2     55000     56000     57000
       3     50000     52000     53000
       4     49000     48000     51000

SQL> SELECT * FROM emp_new;
```

```
   EMP_NO    AVG_SAL
---------- ----------
         2      56000
         3   51666.67
         4   49333.33
```

2. Write a before trigger for Insert, update event considering following requirement:

Emp(e_no, e_name, salary)

I) Trigger action should be initiated when salary is tried to be inserted is less than Rs. 50,000/-

II) Trigger action should be initiated when salary is tried to be updated for value less than Rs. 50,000/-

Action should be rejection of update or Insert operation by displaying appropriate error message.

Also the new values expected to be inserted will be stored in new table Tracking(e_no, salary).

INPUT CODE AND OUTPUT:

```
SET SERVEROUTPUT ON;
CREATE TABLE Emp (
   e_no INT PRIMARY KEY,
   e_name VARCHAR(50),
   salary DECIMAL(10,2)
);
-- Create Tracking table
CREATE TABLE Tracking (
   e_no INT,
   salary DECIMAL(10,2)
);
SQL> CREATE OR REPLACE TRIGGER check_salary
  2  BEFORE INSERT OR UPDATE ON Emp
  3  FOR EACH ROW
  4  BEGIN
```

```
 5     IF :NEW.salary < 50000 THEN
 6         INSERT INTO Tracking (e_no, salary) VALUES (:NEW.e_no, :NEW.salary);
 7         dbms_output.put_line('Salary cannot be less than Rs. 50,000');
 8     END IF;
 9  END;
10  /
Trigger created.
SQL> INSERT INTO Emp VALUES(2, 'Soham', 52000);
SQL> INSERT INTO Emp VALUES(4, 'Hari', 55000);
SQL> BEGIN
 2     INSERT INTO Emp VALUES(5, 'Vedant', 40000);
 3  EXCEPTION
 4     WHEN OTHERS THEN
 5         DBMS_OUTPUT.PUT_LINE('Error!!!!Try again later.');
 6  END;
 7  /
PL/SQL procedure successfully completed.
SQL> SELECT * FROM Emp;
    E_NO E_NAME                                           SALARY
---------- -------------------------------------------------- ----------
         1 Mrunalini                                       60000
         2 Soham                                           52000
         3 Hari                                            55000
         4 Sejal                                           55000
         5 Vedant                                          40000
SQL> SELECT * FROM Tracking;
    E_NO     SALARY
---------- ----------
         5      40000
```

```
SQL> CREATE TABLE emp_sal (
  2     emp_no INT,
  3     sal1 NUMBER(10, 2),
  4     sal2 NUMBER(10, 2),
  5     sal3 NUMBER(10, 2)
  6  );
Table created.
SQL> CREATE TABLE emp_new (
  2     emp_no INT,
  3     avg_sal NUMBER(10, 2)
  4  );
Table created.
SQL> CREATE OR REPLACE TRIGGER insert_emp_sal_trigger
  2  BEFORE INSERT ON emp_sal
  3  FOR EACH ROW
  4  DECLARE
  5     avg_salary NUMBER(10, 2);
  6  BEGIN
  7     avg_salary := ( :NEW.sal1 + :NEW.sal2 + :NEW.sal3 ) / 3;
  8     IF :NEW.sal1 > 50000 OR :NEW.sal2 > 50000 OR :NEW.sal3 > 50000 THEN
  9        -- Insert into emp_new table
 10         INSERT INTO emp_new (emp_no, avg_sal) VALUES (:NEW.emp_no, avg_salary);
 11     END IF;
 12  END;
 13  /
Trigger created.
SQL> INSERT INTO emp_sal  VALUES (1, 45000, 48000, 49000);
1 row created.
SQL> INSERT INTO emp_sal  VALUES (2, 55000, 56000, 57000);
```

1 row created.

SQL> INSERT INTO emp_sal VALUES (3, 50000, 52000, 53000);

1 row created.

SQL> INSERT INTO emp_sal VALUES (4, 49000, 48000, 51000);

1 row created.

SQL> SELECT * FROM emp_sal;

EMP_NO	SAL1	SAL2	SAL3
1	45000	48000	49000
2	55000	56000	57000
3	50000	52000	53000
4	49000	48000	51000

SQL> SELECT * FROM emp_new;

EMP_NO	AVG_SAL
2	56000
3	51666.67
4	49333.33

2. Write a before trigger for Insert, update event considering following requirement:

Emp(e_no, e_name, salary)

I) Trigger action should be initiated when salary is tried to be inserted is less than Rs. 50,000/-

II) Trigger action should be initiated when salary is tried to be updated for value less than Rs.

50,000/-

Action should be rejection of update or Insert operation by displaying appropriate error message.

Also the new values expected to be inserted will be stored in new table Tracking(e_no, salary).

INPUT CODE AND OUTPUT:

```
SET SERVEROUTPUT ON;
CREATE TABLE Emp (
   e_no INT PRIMARY KEY,
```

```
    e_name VARCHAR(50),
    salary DECIMAL(10,2)
);
-- Create Tracking table
CREATE TABLE Tracking (
    e_no INT,
    salary DECIMAL(10,2)
);
SQL> CREATE OR REPLACE TRIGGER check_salary
  2  BEFORE INSERT OR UPDATE ON Emp
  3  FOR EACH ROW
  4  BEGIN
  5      IF :NEW.salary < 50000 THEN
  6          INSERT INTO Tracking (e_no, salary) VALUES (:NEW.e_no, :NEW.salary);
  7          dbms_output.put_line('Salary cannot be less than Rs. 50,000');
  8      END IF;
  9  END;
 10  /
Trigger created.
SQL> INSERT INTO Emp VALUES(1, 'Soham', 60000);
1 row created.
SQL> INSERT INTO Emp VALUES(2, 'Vedant', 52000);
1 row created.
SQL> INSERT INTO Emp VALUES(3, 'Hari', 55000);
1 row created.
SQL> INSERT INTO Emp VALUES(4, 'hari', 55000);
1 row created.
SQL> BEGIN
  2      INSERT INTO Emp VALUES(5, 'Sakshi', 40000);
```

```
  3  EXCEPTION
  4     WHEN OTHERS THEN
  5        DBMS_OUTPUT.PUT_LINE('Error!!!!Try again later.');
  6  END;
  7  /
```

PL/SQL procedure successfully completed.

```
SQL> SELECT * FROM Emp;

     E_NO E_NAME                                           SALARY
---------- -------------------------------------------------- ----------
         1 Soham                                          60000
         2 Vedant                                                 52000
         3 Hari                                           55000
         4 hari                                           55000
         5 Sakshi                                         40000
SQL> SELECT * FROM Tracking;

     E_NO     SALARY
---------- ----------
         5      40000
```