Ex.No.11	TRIGGERS

AIM

To implement and demonstrate the use of database triggers to perform and control INSERT, UPDATE, and DELETE function.

CREATE TABLE

```
SQL> CREATE TABLE students (
  2     student_id NUMBER PRIMARY KEY,
  3     name VARCHAR2(50),
  4     department VARCHAR2(30)
  5  );
```

Table created.

INSERT VALUES TO TABLE

```
SQL> INSERT INTO students (student_id, name, department) VALUES (1, 'Jeevashre',
'Computer Science');
```

1 row created.

```
SQL> INSERT INTO students (student_id, name, department) VALUES (2, 'Nandhini',
'Information Technology');
```

1 row created.

```
SQL> INSERT INTO students (student_id, name, department  VALUES (3, ‘Pooja',
'Electronics');
```

1 row created.

```
SQL> CREATE TABLE audit_students (
  2     student_id NUMBER,
  3     action_time DATE,
  4     action_type VARCHAR2(10)
  5  );
```

Table created.

```
SQL> CREATE OR REPLACE TRIGGER trg_audit_students
  2  AFTER INSERT ON students
  3  FOR EACH ROW
  4  BEGIN
  5     INSERT INTO audit_students(student_id, action_time, action_type)
  6     VALUES(:NEW.student_id, SYSDATE, 'INSERT');
  7  END;
  8  /
Trigger created.

SQL> INSERT INTO students (student_id, name, department)
  2  VALUES (4, 'David', 'Mechanical');
1 row created.

SQL> SELECT * FROM audit_students;
STUDENT_ID ACTION_TI ACTION_TYP
---------- --------- ----------
         4 03-MAY-25 INSERT

SQL> CREATE OR REPLACE TRIGGER trg_update_students
  2  AFTER UPDATE ON students
  3  FOR EACH ROW
  4  BEGIN
  5     INSERT INTO audit_students(student_id, action_time, action_type)
  6     VALUES(:NEW.student_id, SYSDATE, 'UPDATE');
  7  END;
  8  /
Trigger created.
```

```
SQL> UPDATE students
  2  SET department = 'AI & DS'
  3  WHERE student_id = 2;
Enter value for ds: 2
old   2: SET department = 'AI & DS'
new   2: SET department = 'AI 2'
1 row updated.

SQL> SELECT * FROM audit_students;
STUDENT_ID ACTION_TI ACTION_TYP
---------- --------- ----------
         4 03-MAY-25 INSERT
         2 03-MAY-25 UPDATE

SQL> CREATE OR REPLACE TRIGGER trg_delete_students
  2  AFTER DELETE ON students
  3  FOR EACH ROW
  4  BEGIN
  5      INSERT INTO audit_students(student_id, action_time, action_type)
  6      VALUES(:OLD.student_id, SYSDATE, 'DELETE');
  7  END;
  8  /
Trigger created.

SQL> DELETE FROM students
  2  WHERE student_id = 3;
1 row deleted.

SQL> SELECT * FROM audit_students;
```

```
STUDENT_ID ACTION_TI ACTION_TYP
---------- --------- ----------
         4 03-MAY-25 INSERT
         2 03-MAY-25 UPDATE
         3 03-MAY-25 DELETE
```

EXAMPLE 1

INSERT, UPDATE, DELETE ON STUDENTS TABLE

```
SQL> CREATE OR REPLACE TRIGGER trg_students_all_actions
  2  AFTER INSERT OR UPDATE OR DELETE ON students
  3  FOR EACH ROW
  4  BEGIN
  5     IF INSERTING THEN
  6        INSERT INTO audit_students(student_id, action_time, action_type)
  7        VALUES(:NEW.student_id, SYSDATE, 'INSERT');
  8     ELSIF UPDATING THEN
  9        INSERT INTO audit_students(student_id, action_time, action_type)
 10        VALUES(:NEW.student_id, SYSDATE, 'UPDATE');
 11     ELSIF DELETING THEN
 12        INSERT INTO audit_students(student_id, action_time, action_type)
 13        VALUES(:OLD.student_id, SYSDATE, 'DELETE');
 14     END IF;
 15  END;
 16  /
Trigger created.

SQL> INSERT INTO students (student_id, name, department)
  2  VALUES (5, 'Eva', 'Data Science');
1 row created.
```

```
SQL> UPDATE students
  2  SET department = 'Cybersecurity'
  3  WHERE student_id = 1;
1 row updated.

SQL> DELETE FROM students
  2  WHERE student_id = 2;
1 row deleted.

SQL> SELECT * FROM audit_students;
STUDENT_ID ACTION_TI ACTION_TYP
---------- --------- ----------
         4 06-MAY-25 INSERT
         2 06-MAY-25 UPDATE
         3 06-MAY-25 DELETE
         5 06-MAY-25 INSERT
         5 06-MAY-25 INSERT
         1 06-MAY-25 UPDATE
         1 06-MAY-25 UPDATE
         2 06-MAY-25 DELETE
         2 06-MAY-25 DELETE
9 rows selected.
```

EXAMPLE 2

PREVENT NULL VALUE FOR DEPARTMENT

```
SQL> CREATE OR REPLACE TRIGGER trg_prevent_null_dept
  2  BEFORE UPDATE ON students
  3  FOR EACH ROW
  4  BEGIN
  5     IF :NEW.department IS NULL THEN
```

```
  6         RAISE_APPLICATION_ERROR(-20002, 'Department cannot be set to NULL.');
  7     END IF;
  8  END;
  9  /
Trigger created.


SQL> UPDATE students
  2  SET department = NULL
  3  WHERE student_id = 1;
UPDATE students
       *
ERROR at line 1:
ORA-20002: Department cannot be set to NULL.
ORA-06512: at "SYSTEM.TRG_PREVENT_NULL_DEPT", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRG_PREVENT_NULL_DEPT'
```

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

RESULT

Thus, the experiment successfully showcased how database triggers can be used for enforcing business rules and maintaining audit trails automatically.