Experiment 7

Student Name:ROHIT KUMAR **UID: 23BCS12640**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 5th **Date of Performance:09/10/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. Aim:

Problem 1:

- Requirements: Design a trigger which:
  Whenever there is an insertion on the STUDENT table, the currently inserted or deleted row should be printed as it is on the output console window.

Problem 2:
Requirements: Design a PostgreSQL trigger that:

- Whenever a new employee is inserted in tbl_employee, a record should be added to tbl_employee_audit like:"Employee name <emp_name> has been added at <current_time>"

- Whenever an employee is deleted from tbl_employee, a record should be added to tbl_employee_audit like:"Employee name <emp_name> has been deleted at <current_time>"

2. Objective:

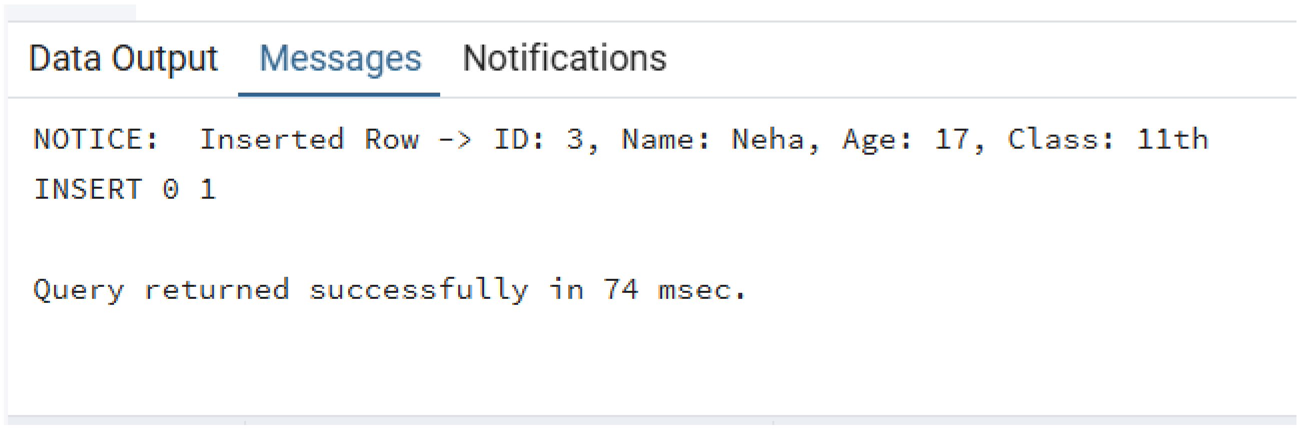
- Design triggers to automatically respond to INSERT and DELETE operations.
- Print inserted or deleted rows to the console output for immediate feedback.
- Log changes in an audit table with descriptive messages.
- Understand the use of NEW and OLD records in trigger functions.
- Gain hands-on experience in PostgreSQL procedural programming.

3. DBMS script and output:

Solution 1:

```
-- Step 1: Create main table
DROP TABLE IF EXISTS student;
CREATE TABLE student (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100),
   age INT,
   class VARCHAR(20)
);
```

```
-- Step 2: Create Trigger Function
CREATE OR REPLACE FUNCTION fn_student_audit()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
    IF TG_OP = 'INSERT' THEN
        RAISE NOTICE 'Inserted Row -> ID: %, Name: %, Age: %, Class: %',
                     NEW.id, NEW.name, NEW.age, NEW.class;
        RETURN NEW;

    ELSIF TG_OP = 'DELETE' THEN
        RAISE NOTICE 'Deleted Row -> ID: %, Name: %, Age: %, Class: %',
                     OLD.id, OLD.name, OLD.age, OLD.class;
        RETURN OLD;
    END IF;

    RETURN NULL;
END;
$$;

-- Step 3: Create Trigger
CREATE TRIGGER trg_student_audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION fn_student_audit();

-- Step 4: Testing
-- Insert new records
INSERT INTO student(name, age, class) VALUES ('Aarav', 16, '10th');
INSERT INTO student(name, age, class) VALUES ('Neha', 17, '11th');

-- Delete a record
DELETE FROM student WHERE name = 'Aarav';

-- Check final data
SELECT * FROM student;
```

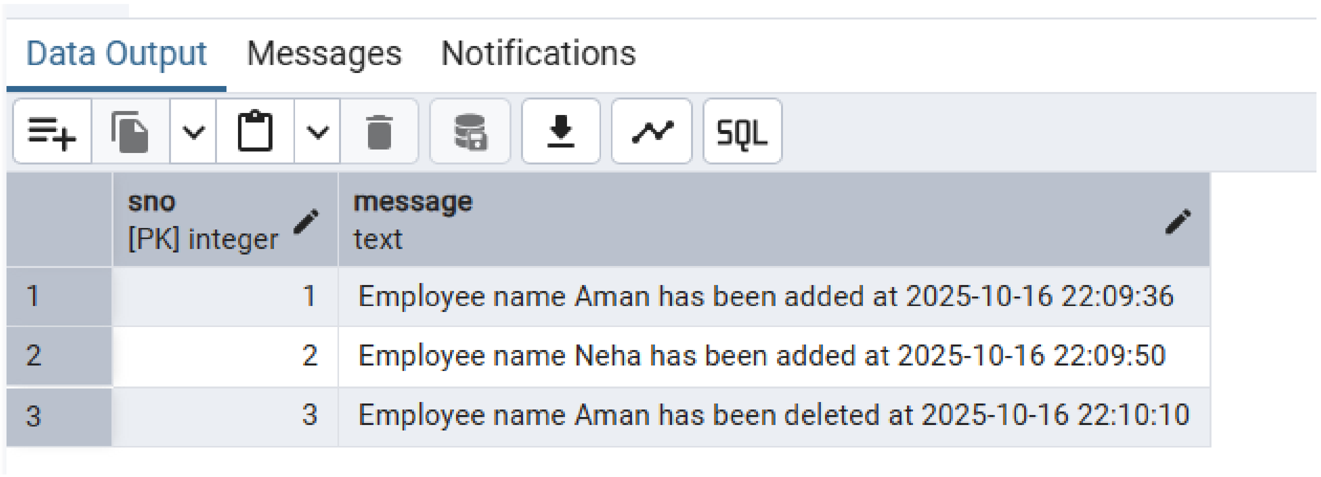


```
NOTICE:  Inserted Row -> ID: 3, Name: Neha, Age: 17, Class: 11th
INSERT 0 1


Query returned successfully in 74 msec.
```

Solution 2:

```
-- Step 1: Create main employee and audit tables
DROP TABLE IF EXISTS tbl_employee_audit;
DROP TABLE IF EXISTS tbl_employee;

CREATE TABLE tbl_employee (
   emp_id SERIAL PRIMARY KEY,
   emp_name VARCHAR(100) NOT NULL,
   emp_salary NUMERIC
);

CREATE TABLE tbl_employee_audit (
   sno SERIAL PRIMARY KEY,
   message TEXT
);

-- Step 2: Create Trigger Function
CREATE OR REPLACE FUNCTION audit_employee_changes()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
   IF TG_OP = 'INSERT' THEN
      INSERT INTO tbl_employee_audit(message)
      VALUES ('Employee name ' || NEW.emp_name ||
            ' has been added at ' || TO_CHAR(NOW(), 'YYYY-MM-DD HH24:MI:SS'));
      RETURN NEW;

   ELSIF TG_OP = 'DELETE' THEN
      INSERT INTO tbl_employee_audit(message)
      VALUES ('Employee name ' || OLD.emp_name ||
            ' has been deleted at ' || TO_CHAR(NOW(), 'YYYY-MM-DD HH24:MI:SS'));
      RETURN OLD;
   END IF;
```

```
    RETURN NULL;
END;
$$;

-- Step 3: Create Trigger
CREATE TRIGGER trg_employee_audit
AFTER INSERT OR DELETE
ON tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();

-- Step 4: Testing the Trigger
-- Insert employees
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Aman', 50000);
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Neha', 60000);

-- Delete an employee
DELETE FROM tbl_employee WHERE emp_name = 'Aman';

-- Step 5: Check Audit Table
SELECT * FROM tbl_employee_audit;

-- Step 6: Check Remaining Employees
SELECT * FROM tbl_employee;
```



	sno [PK] integer	message text
1	1	Employee name Aman has been added at 2025-10-16 22:09:36
2	2	Employee name Neha has been added at 2025-10-16 22:09:50
3	3	Employee name Aman has been deleted at 2025-10-16 22:10:10

4. Learning Outcomes (What I have Learnt):

- Ability to create PL/pgSQL trigger functions in PostgreSQL.
- Understand the difference between row-level and statement-level triggers.
- Learn to use RAISE NOTICE to display runtime information.
- Implement basic database auditing mechanisms.
- Improve skills in automating database tasks and monitoring changes efficiently.