Experiment 7

Student Name: Tanisha Kumari **UID:** 23BCS12542
Branch: CSE **Section/Group:** KRG_2B
Semester: 5th **Date of Performance:** 09/10/25
Subject Name: ADBMS **Subject Code:** 23CSP-333

1. Aim:

Part A – Triggers: Student Data Change Monitoring (Medium)
EduSmart Institute wants to monitor all insertions and deletions in the student database. Whenever a new student record is inserted or deleted from the student table, the details of that record should be displayed on the PostgreSQL console window.

Part B – Triggers: Employee Activity Logging (Hard)
TechSphere Solutions wants to maintain an automatic audit trail for all employee additions and deletions in the company database.
Whenever a new employee is added or removed from the tbl_employee table, an entry should be recorded in the tbl_employee_audit table for tracking purposes.

2. Objective:

Medium-Level Problem:
Design a PostgreSQL trigger that:
1. Prints the complete details of the inserted or deleted student record using RAISE NOTICE.
2. Activates automatically after every INSERT or DELETE operation on the student table.

Hard-Level Problem:
Design a PostgreSQL trigger that:
1. Inserts a message in tbl_employee_audit whenever a new employee is added or deleted.
2. The message should include the employee's name and the current timestamp.
3. Activates automatically after every INSERT or DELETE operation on tbl_employee.

3. ADBMS script and output:

MEDIUM-LEVEL PROBLEM

```
-- Create the table
CREATE TABLE student (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100),
   age INT,
   class VARCHAR(50)
);

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;
$$;

-- Create the trigger
CREATE TRIGGER trg_student_audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION fn_student_audit();
```

OUTPUT:-

Output:

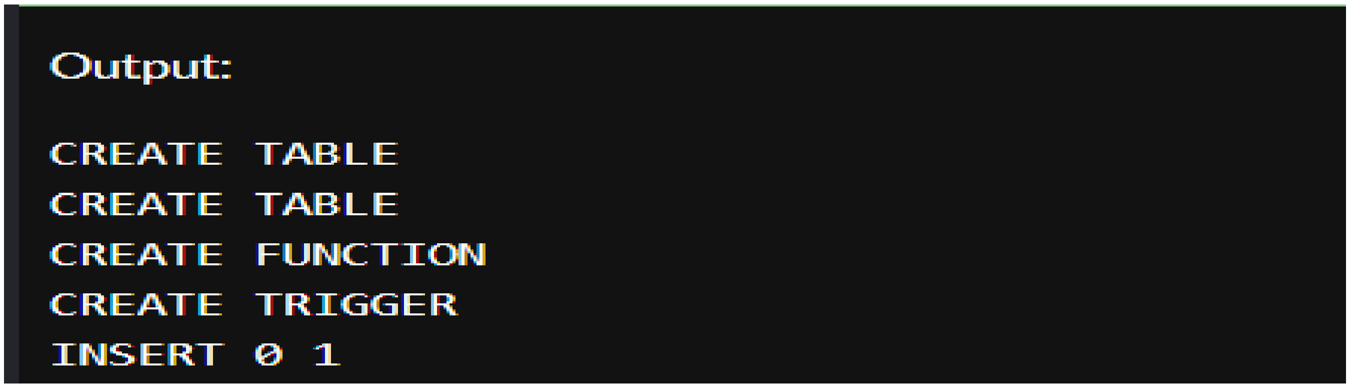
```
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
```

HARD LEVEL PROBLEM:

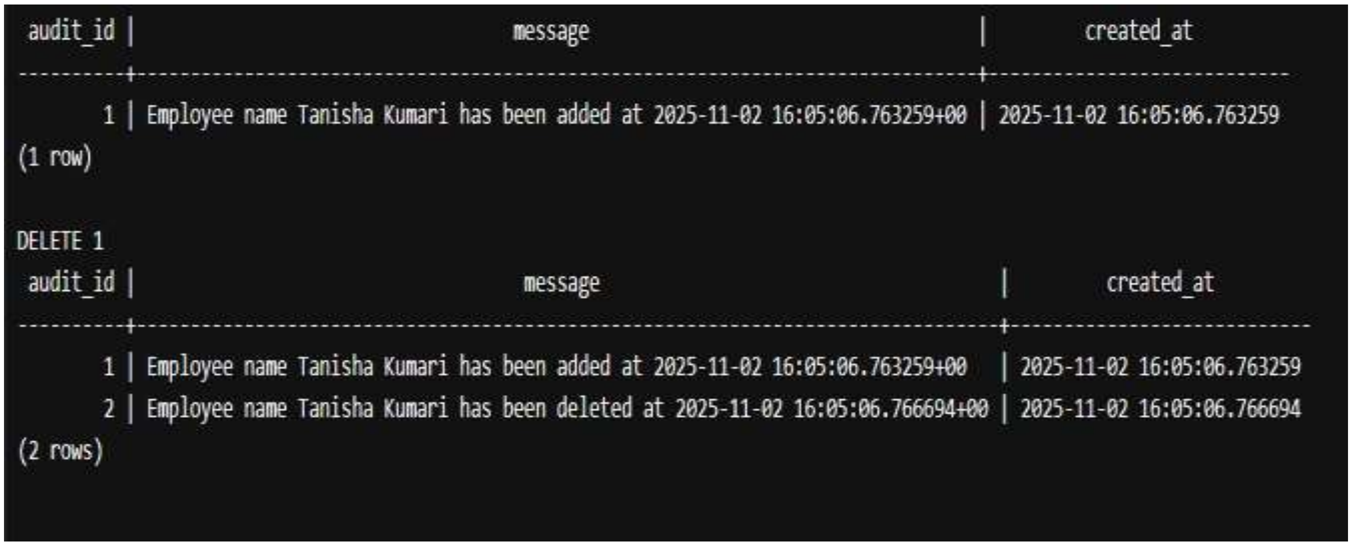
```
-- Create the employee table
CREATE TABLE tbl_employee (
   emp_id SERIAL PRIMARY KEY,
   emp_name VARCHAR(100),
   designation VARCHAR(50),
   salary NUMERIC(10,2)
);

-- Create the audit table
CREATE TABLE tbl_employee_audit (
   audit_id SERIAL PRIMARY KEY,
   message TEXT,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- Create the 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 ' || NOW());
        RETURN NEW;

    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO tbl_employee_audit(message)
        VALUES ('Employee name ' || OLD.emp_name || ' has been deleted at ' || NOW());
        RETURN OLD;
    END IF;

    RETURN NULL;
END;
$$;
-- Create the trigger
CREATE TRIGGER trg_employee_audit
AFTER INSERT OR DELETE
ON tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();

-- Test the trigger
INSERT INTO tbl_employee (emp_name, designation, salary)
VALUES ('Tanisha Kumari', 'Software Engineer', 55000);
-- Check audit logs
SELECT * FROM tbl_employee_audit;
-- Delete the record
DELETE FROM tbl_employee WHERE emp_name = 'Tanisha Kumari';
-- Check audit logs again
SELECT * FROM tbl_employee_audit;
```

OUTPUTS:



```
Output:

CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
```



```
 audit_id |                              message                                |         created_at
----------+---------------------------------------------------------------------+----------------------------
        1 | Employee name Tanisha Kumari has been added at 2025-11-02 16:05:06.763259+00 | 2025-11-02 16:05:06.763259
(1 row)

DELETE 1
 audit_id |                               message                                 |         created_at
----------+-----------------------------------------------------------------------+----------------------------
        1 | Employee name Tanisha Kumari has been added at 2025-11-02 16:05:06.763259+00   | 2025-11-02 16:05:06.763259
        2 | Employee name Tanisha Kumari has been deleted at 2025-11-02 16:05:06.766694+00 | 2025-11-02 16:05:06.766694
(2 rows)
```

Learning Outcomes:

1. Understand the concept and purpose of database triggers in PostgreSQL.
2. Learn how to automate data tracking using AFTER INSERT and AFTER DELETE triggers.
3. Gain hands-on experience with trigger functions written in PL/pgSQL.
4. Develop the ability to implement audit logging for real-time database monitoring.
5. Enhance skills in maintaining data integrity and traceability in relational databases