Worksheet 7

1. Aim:

1. Design a PostgreSQL trigger that performs the following task:
   a. Whenever a new record is inserted into the student table, the inserted row should be displayed on the output console.
   b. Similarly, when a record is deleted from the student table, the deleted row should also be displayed on the console.

2. Create PostgreSQL triggers to maintain an audit log for employee actions.

   a. Whenever a new employee is inserted into tbl_employee, a record should be inserted into tbl_employee_audit with the message: "Employee name <emp_name> has been added at <current_time>"
   b. Whenever an employee is deleted from tbi_employee, a record should be inserted into tbl_employee_audit with the message: "Employee name <emp_name> has been deleted at <current_time>"

2. Objective:

- Maintain a complete and reliable record of all employee insertions and deletions for accountability and auditing purposes.
- Automatically insert descriptive audit messages into tbl_employee_audit whenever changes occur in tbl_employee, without requiring manual input.
- Guarantee that every change in the employee table is consistently tracked in real-time, reducing the risk of unrecorded modifications.
- Store timestamps and employee names in the audit log to create a chronological history of employee activity for future reference and compliance checks.
- Increase visibility into employee-related database actions, supporting internal monitoring, troubleshooting, and security reviews.

3. Code:

1.

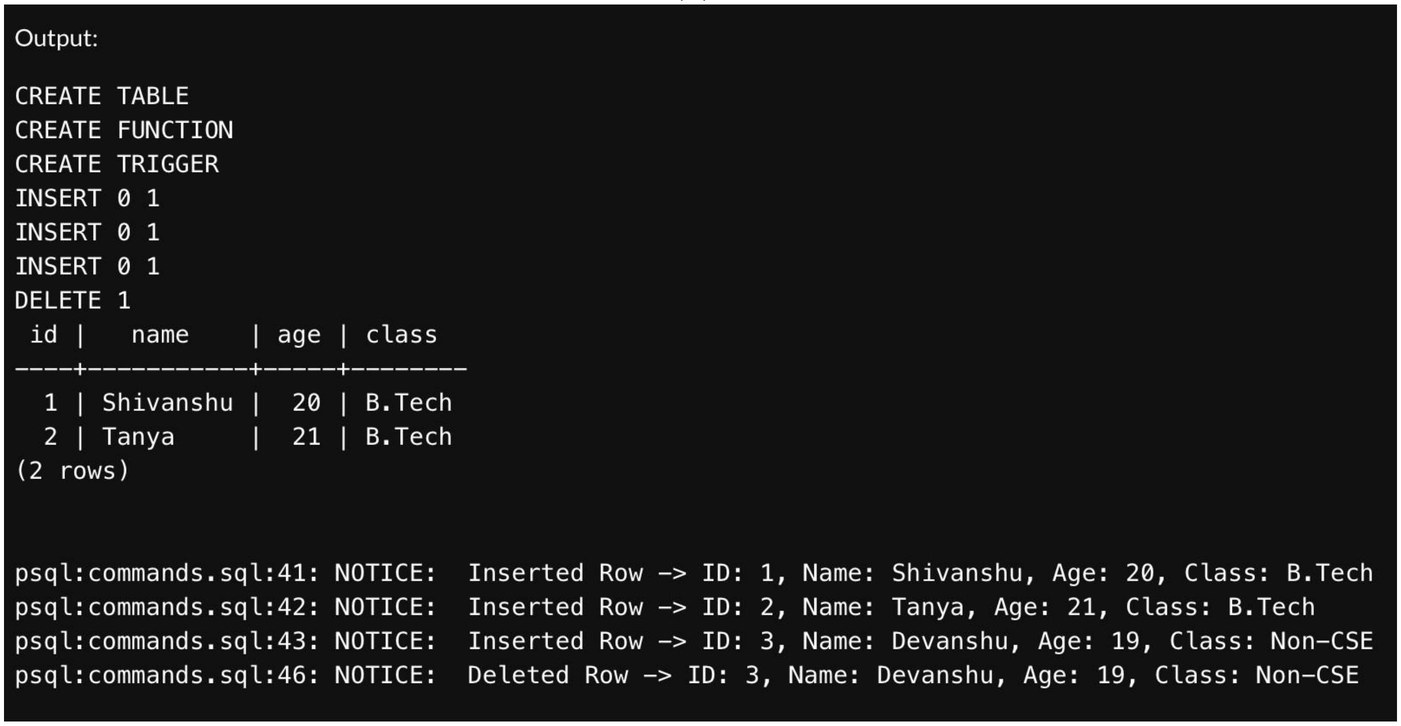
```
-- Create the student table
CREATE TABLE student (
id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    age INT,     class
VARCHAR(50)
);

-- Create the 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;
$$;

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

-- Test the trigger

-- Insert records
INSERT INTO student(name, age, class) VALUES ('Shivanshu', 20, 'B.Tech');
INSERT INTO student(name, age, class) VALUES ('Tanya', 21, 'B.Tech');
INSERT INTO student(name, age, class) VALUES ('Devanshu', 19, 'Non-CSE');

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

SELECT * FROM student;
```

2.

```
-- Create employee and audit tables 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
);

-- 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 employees
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Shivanshu', 90000);
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Tanya', 95000);
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Karan', 100000);

-- Delete one employee
DELETE FROM tbl_employee WHERE emp_name = 'Karan';

SELECT * FROM tbl_employee;
SELECT * FROM tbl_employee_audit;
```

4. Output:

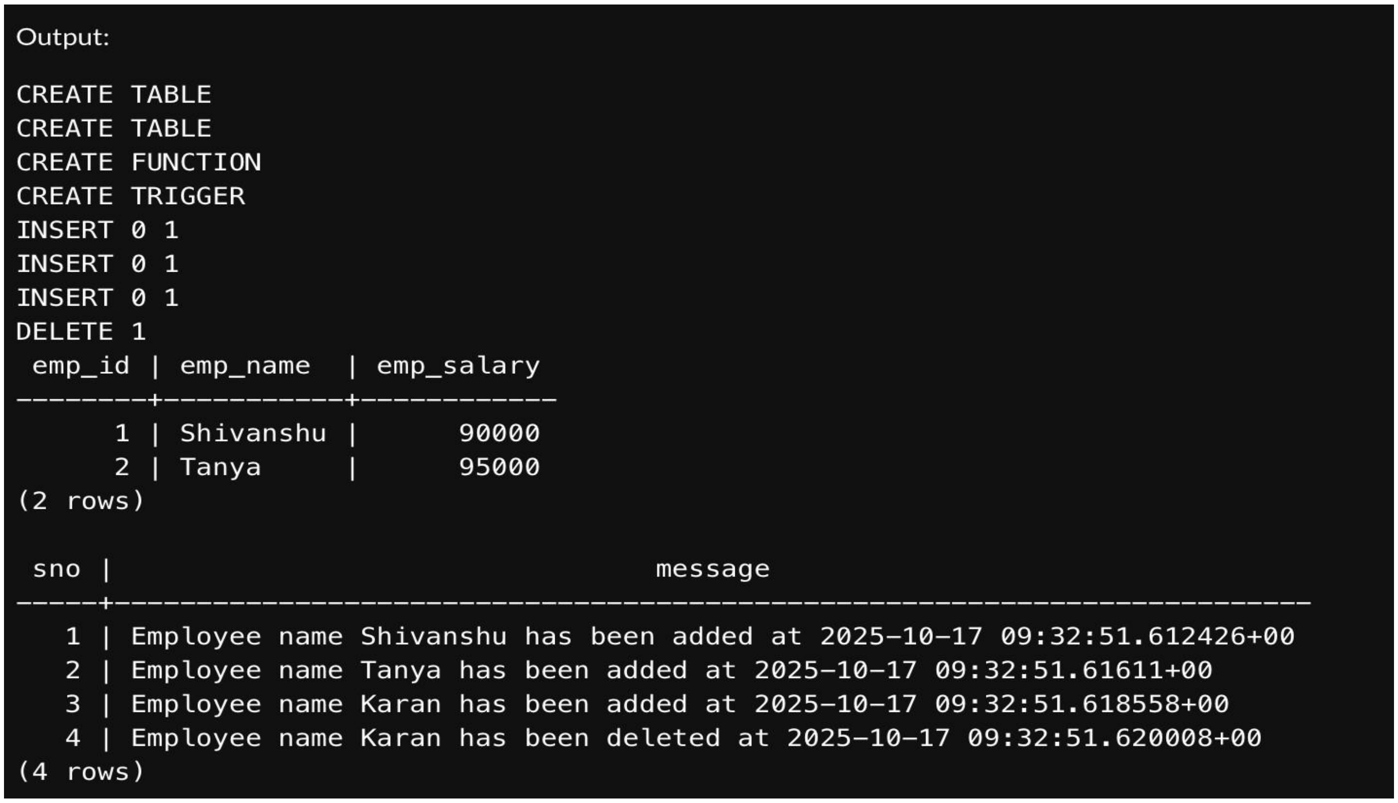
(1)



Output:

```
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
INSERT 0 1
INSERT 0 1
DELETE 1
 id |    name    | age | class
----+------------+-----+--------
  1 | Shivanshu |  20 | B.Tech
  2 | Tanya      |  21 | B.Tech
(2 rows)


psql:commands.sql:41: NOTICE:  Inserted Row -> ID: 1, Name: Shivanshu, Age: 20, Class: B.Tech
psql:commands.sql:42: NOTICE:  Inserted Row -> ID: 2, Name: Tanya, Age: 21, Class: B.Tech
psql:commands.sql:43: NOTICE:  Inserted Row -> ID: 3, Name: Devanshu, Age: 19, Class: Non-CSE
psql:commands.sql:46: NOTICE:  Deleted Row -> ID: 3, Name: Devanshu, Age: 19, Class: Non-CSE
```

(2)



Output:

```
CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
INSERT 0 1
INSERT 0 1
DELETE 1
 emp_id | emp_name  | emp_salary
--------+-----------+------------
      1 | Shivanshu |      90000
      2 | Tanya     |      95000
(2 rows)

 sno |                               message
-----+-----------------------------------------------------------------------
   1 | Employee name Shivanshu has been added at 2025-10-17 09:32:51.612426+00
   2 | Employee name Tanya has been added at 2025-10-17 09:32:51.61611+00
   3 | Employee name Karan has been added at 2025-10-17 09:32:51.618558+00
   4 | Employee name Karan has been deleted at 2025-10-17 09:32:51.620008+00
(4 rows)
```

5. Learning Outcomes:

- Understanding Trigger Mechanisms
- Practical Use of Trigger Functions
- Implementing Auditing and Logging
- Event-driven Automation in Databases