DEPARTMENTOF
COMPUTERSCIENCE&ENGINEERING
Discover. Learn. Empower.

# WORKSHEET 7

**Student Name: Harsh** **UID: 23BAI70474**

**Branch: CSE(3rd Year)** **Section: 23AIT_Krg_G2**

**Semester: 5th** **Date of Performance: 27/10/25**

**Subject Name: ADBMS** **Subject Code: 23CSP-333**

1. **AIM:**

**i)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.

**Objective:**

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.

**ii) 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.

**Objective:**

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.

2. **Tools Used :** PostGres

## Solutions:

Q1)

--CREATING A TABLE

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


--TRIGGER FUNCTION

CREATE OR REPLACE FUNCTION fn_student_audit()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$ BE
GIN
   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;
$$;


--CREATING A TRIGGER
CREATE TRIGGER trg_student_audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION fn_student_audit();

Q2)
CREATE TABLE tbl_employee
   ( emp_id SERIAL PRIMARY
   KEY,                  emp_name
   VARCHAR(100), designation
   VARCHAR(50),
```

```
    salary NUMERIC(10,2)
);
CREATE TABLE tbl_employee_audit (
    audit_id SERIAL PRIMARY KEY,
    message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION audit_employee_changes()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$ BE
GIN
    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 TRIGGER trg_employee_audit
AFTER INSERT OR DELETE
ON tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();


INSERT INTO tbl_employee (emp_name, designation, salary)
VALUES ('Supriya Dutta', 'Software Engineer', 55000);

SELECT * FROM tbl_employee_audit;

DELETE FROM tbl_employee WHERE emp_name = 'Supriya Dutta';

SELECT * FROM tbl_employee_audit;
```

## 3. Output:

Query | Query History | Scratch Pad

```
    RETURN NULL;
END;
$$;


--CREATING A TRIGGER
CREATE TRIGGER trg_student_audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION fn_student_audit();

INSERT INTO student (name, age, class)
VALUES ('Supriya Dutta', 21, 'CS101');
```

Data Output | Messages | Notifications

```
NOTICE:  Inserted Row -> ID: 1, Name: Supriya Dutta, Age: 21, Class: CS101
INSERT 0 1

Query returned successfully in 42 msec.
```

Query | Query History | Scratch Pad

```
ON tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();


INSERT INTO tbl_employee (emp_name, designation, salary)
VALUES ('Supriya Dutta', 'Software Engineer', 55000);

SELECT * FROM tbl_employee_audit;

DELETE FROM tbl_employee WHERE emp_name = 'Supriya Dutta';

SELECT * FROM tbl_employee_audit;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No: 1 of 1

| | audit_id [PK] integer | message text | created_at timestamp without time zone |
|---|---|---|---|
| 1 | 1 | Employee name Supriya Dutta has been added at 2025-10-21 21:02:59.425952+05:30 | 2025-10-21 21:02:59.425952 |

Query | Query History | Scratch Pad

```
ON tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();


INSERT INTO tbl_employee (emp_name, designation, salary)
VALUES ('Supriya Dutta', 'Software Engineer', 55000);

SELECT * FROM tbl_employee_audit;

DELETE FROM tbl_employee WHERE emp_name = 'Supriya Dutta';

SELECT * FROM tbl_employee_audit;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2 | Page No: 1 of 1

| | audit_id [PK] integer | message text | created_at timestamp without time zone |
|---|---|---|---|
| 1 | 1 | Employee name Supriya Dutta has been added at 2025-10-21 21:02:59.425952+05:30 | 2025-10-21 21:02:59.425952 |
| 2 | 2 | Employee name Supriya Dutta has been deleted at 2025-10-21 21:03:19.998826+05:30 | 2025-10-21 21:03:19.998826 |

## 4. 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.