



## Experiment - 8

Name: Lakshay Sharma  
Section: KRG-3B  
Semester : 5<sup>th</sup>  
Subject Name: ADBMS

UID : 23BCS12276  
Branch: BE-CSE  
D.O.P:28/10/2025  
Subject- Code : 23CSP - 333

### **1. Aim:**

Design a robust PostgreSQL transaction system for the students table where multiple student records are inserted in a single transaction.

If any insert fails due to invalid data, only that insert should be rolled back while preserving the previous successful inserts using savepoints.

The system should provide clear messages for both successful and failed insertions, ensuring data integrity and controlled error handling.

## 2. Requirement :

- Design a robust transaction system for the student table that allows inserting multiple student records in a single transaction.
- If any insert fails due to invalid data, only that specific insert should be rolled back, while previously successful inserts are preserved.
- The system should use savepoints to control partial rollbacks.
- Provide clear messages indicating which inserts succeeded and which failed.
- Ensure data integrity and controlled error handling during bulk inserts.

## 3: Objective:

To implement a PostgreSQL transaction system that supports batch insertion of student records with partial rollback capability.

To use savepoints to isolate failures and prevent total transaction failure.

To give real-time feedback for both successful and failed inserts.

To maintain data consistency and integrity while handling invalid or problematic data gracefully.

To provide a structured and robust approach for bulk data operations in the student table.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4: Code:

```
DROP TABLE IF EXISTS students;
```

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

```
-- EXCEPTION HANDLING
```

```
DO $$
```

```
BEGIN TRANSACTION
```

```
-- Start a transaction
```

```
BEGIN
```

```
-- Insert multiple students
```

```
INSERT INTO students(name, age, class) VALUES ('Anisha',16,8);
```

```
INSERT INTO students(name, age, class) VALUES ('Neha',17,8);
```

```
INSERT INTO students(name, age, class) VALUES ('Mayank',19,9);
```

```
-- If all succeed
```

```
RAISE NOTICE ' Transaction Successfully Done';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
-- If any insert fails
```

```
RAISE NOTICE 'Transaction Failed..! Rolling back changes.';
```

```
RAISE; -- this will rollback the entire transaction
```

```
END;
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
END;  
$$;
```

```
SELECT * FROM students;
```

```
--VIOLATED SCENARIO
```

```
DO $$
```

```
BEGIN TRANSACTION
```

```
-- Start a transaction
```

```
BEGIN
```

```
-- Insert multiple students
```

```
INSERT INTO students(name, age, class) VALUES ('Anisha',16,8);
```

```
INSERT INTO students(name, age, class) VALUES ('Mayank',19,9);
```

```
INSERT INTO students(name, age, class) VALUES ('Anisha',17,8); --ERROR
```

```
INSERT INTO students(name, age, class) VALUES ('Mayank',19,9);
```

```
-- If all succeed
```

```
RAISE NOTICE ' Transaction Successfully Done';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
-- If any insert fails
```

```
RAISE NOTICE 'Transaction Failed..! Rolling back changes.';
```

```
RAISE; -- this will rollback the entire transaction
```

```
END;
```

```
END;
```

```
$$;
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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;
$$
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 5: Output:

Output:

```
DROP TABLE  
CREATE TABLE  
DO
```

id	name	age	class
1	Anisha	16	8
2	Neha	17	8
3	Mayank	19	9

(3 rows)

```
psql:commands.sql:1: NOTICE:  table "students" does not exist, skipping  
psql:commands.sql:30: NOTICE:  Transaction Successfully Done  
psql:commands.sql:54: NOTICE:  Transaction Failed..! Rolling back changes.  
psql:commands.sql:54: ERROR:   duplicate key value violates unique constraint "students_name_key"  
DETAIL:  Key (name)=(Anisha) already exists.  
CONTEXT:  SQL statement "INSERT INTO students(name, age, class) VALUES ('Anisha',16,8)"  
PL/pgSQL function inline_code_block line 6 at SQL statement
```