University Institute of Engineering

Department of Computer Science & Engineering

EXPERIMENT: 7

NAME : Johnson Kumar **UID** : 23BCS12654

SECTION : KRG_2A **SEMESTER**: 5TH

SUBJECT CODE: 23CSP-339 **SUBJECT** : ADBMS

Problem Statement:

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.

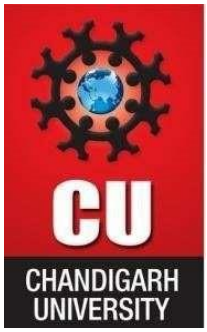
Solution:

```
DROP TABLE IF EXISTS students;

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


DO $$
BEGIN
   -- 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;
END;
$$;
SELECT * FROM students;

--------------------------WRONG DATA TYPE ------------------------
BEGIN;
SAVEPOINT sp1;
INSERT INTO students(name, age, class) VALUES ('Aarav',16,8);
SAVEPOINT sp2;
BEGIN
   INSERT INTO students(name, age, class) VALUES ('Rahul','wrong',9);  -- fails
EXCEPTION WHEN OTHERS THEN
   RAISE NOTICE 'Failed to insert Rahul, rolling back to savepoint sp2';
   ROLLBACK TO SAVEPOINT sp2;
END;

-- Next insert
INSERT INTO students(name, age, class) VALUES ('Sita',17,10);

COMMIT;
```

Output:

id	name	age	class
1	Anisha	16	8
2	Neha	17	8
3	Mayank	19	9
4	Aarav	16	8
5	Sita	17	10



University Institute of Engineering

Department of Computer Science & Engineering

Learning outcomes:

- Learned how to create and reset tables using DROP TABLE IF EXISTS and CREATE TABLE.
- Understood the use of DO $$ blocks for procedural control in PostgreSQL.
- Practiced inserting multiple rows within a transaction to ensure atomic operations.
- Explored exception handling using BEGIN...EXCEPTION...END to manage errors gracefully.
- Used RAISE NOTICE to generate informative messages during execution.