



WORKSHEET 8

Student Name: Jayaprakash

UID: 23BAI70240

Branch: B.E CSE-H (AIML)

Section: 23-AIT_KRG _G2

Semester: 5th

Date of Performance: 27-10-2025

Subject Name: ADBMS

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. Tools Used: Postgres

Solutions:

Q1)

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

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

```
DO  
$$      BE  
GIN  
BEGIN  
    INSERT INTO students(name, age, class) VALUES ('Supriya',16,8);  
    INSERT INTO students(name, age, class) VALUES ('Rakshit',17,8);
```

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

RAISE NOTICE 'Transaction Successfully Done';

EXCEPTION
WHEN OTHERS THEN
    RAISE NOTICE 'Transaction Failed..! Rolling back changes.';
    RAISE;
END;
END;
$$;
```

```
SELECT * FROM students;
```

WRONG DATA TYPE SCENARIO

```
BEGIN; -- start transaction

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; -- commit all successful inserts
```

3. Output:

Kargil Practicals/postgres@PostgreSQL 17

Query History

```
23     INSERT INTO students(name, age, class) VALUES ('Varun',19,9);
24
25     RAISE NOTICE 'Transaction Successfully Done';
26
27     EXCEPTION
28         WHEN OTHERS THEN
29             RAISE NOTICE 'Transaction Failed..! Rolling back changes.';
30             RAISE;
31     END;
32
33 $$;
34
35 SELECT * FROM students;
```

Data Output Messages Notifications

NOTICE: Transaction Successfully Done
DO

Query returned successfully in 39 msec.

Kargil Practicals/postgres@PostgreSQL 17

Query History

```
28     WHEN OTHERS THEN
29         RAISE NOTICE 'Transaction Failed..! Rolling back changes.';
30         RAISE;
31     END;
32
33 $$;
34
35 SELECT * FROM students;
```

-----WRONG DATA TYPE SCENARIO-----

```
38
39 BEGIN;
40
41 SAVEPOINT spl;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	id [PK] integer	name character varying (50)	age integer	class integer
1	1	Supriya	16	8
2	2	Rakshit	17	8
3	3	Varun	19	9

The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for 'Query' (selected), 'Query History', 'Scratch Pad', and a close button. Below the tabs is a code editor window containing the following SQL script:

```

42
43  SAVEPOINT sp2;
44  BEGIN
45      INSERT INTO students(name, age, class) VALUES ('Rahul','wrong',9); -- fails
46  EXCEPTION WHEN OTHERS THEN
47      RAISE NOTICE 'Failed to insert Rahul, rolling back to savepoint sp2';
48      ROLLBACK TO SAVEPOINT sp2;
49  END;
50
51  -- Next insert
52  INSERT INTO students(name, age, class) VALUES ('Sita',17,10);
53
54  COMMIT; -- commit all successful inserts
55

```

Below the code editor is a 'Data Output' tab (selected), which displays the following error message:

ERROR: syntax error at or near "INSERT"
LINE 2: INSERT INTO students(name, age, class) VALUES ('Rahul',...
A

SQL state: 42601
Character: 11

Below the error message is another 'Data Output' tab, which shows the command 'ROLLBACK'.

Query returned successfully in 35 msec.

3. Learning Outcomes:

- ❑ Understand the concept of PostgreSQL transactions and how to start, commit, and rollback.
- ❑ Learn how to use **SAVEPOINT** to handle partial rollbacks within a transaction.
- ❑ Practice controlled error handling for individual insert failures without affecting other successful operations.
- ❑ Gain experience in maintaining **data integrity** while performing multiple inserts.
- ❑ Learn to generate informative **NOTICES** to monitor transaction progress and errors.