



## WORKSHEET 8

**Student Name:** Sukhjinder Singh

**UID:** 23BAI70078

**Branch:** CSE(3<sup>rd</sup> Year)

**Section:** 23AIT\_Krg\_G2

**Semester:** 5<sup>th</sup>

**Date of Performance:** 27/10/25

**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:



The screenshot displays a PostgreSQL query editor interface. The top section, titled 'Query', contains a SQL script for a transaction. The script starts with a `SAVEPOINT sp2;` followed by a `BEGIN` block. Inside the block, an `INSERT INTO students(name, age, class) VALUES ('Rahul','wrong',9);` is executed, which fails. This is followed by an `EXCEPTION WHEN OTHERS THEN` block containing `RAISE NOTICE 'Failed to insert Rahul, rolling back to savepoint sp2';` and `ROLLBACK TO SAVEPOINT sp2;`. The `BEGIN` block ends with `END;`. After the `END;`, a comment `-- Next insert` is present, followed by `INSERT INTO students(name, age, class) VALUES ('Sita',17,10);` and `COMMIT; -- commit all successful inserts`. The bottom section of the interface shows the 'Messages' tab, which contains an error message: 'ERROR: syntax error at or near "INSERT"' on line 2. Below this, the SQL state is shown as '42601' and the character as '11'. The 'Data Output' tab shows a 'ROLLBACK' message and a confirmation that the query returned successfully in 35 msec.

```
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
```

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

SQL state: 42601  
Character: 11

ROLLBACK

Query returned successfully in 35 msec.

#### 4. 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.