Experiment 8

Student Name: Jobanjot Singh Grewal

UID: 23BIA50005

Branch: BE-AIT-CSE

Section/Group: 23AML_KRG-1

Semester: 5th

Date of Performance: 22 Nov 2025

Subject Name: ADBMS

Subject Code: 23CSP-333

HARD - LEVEL

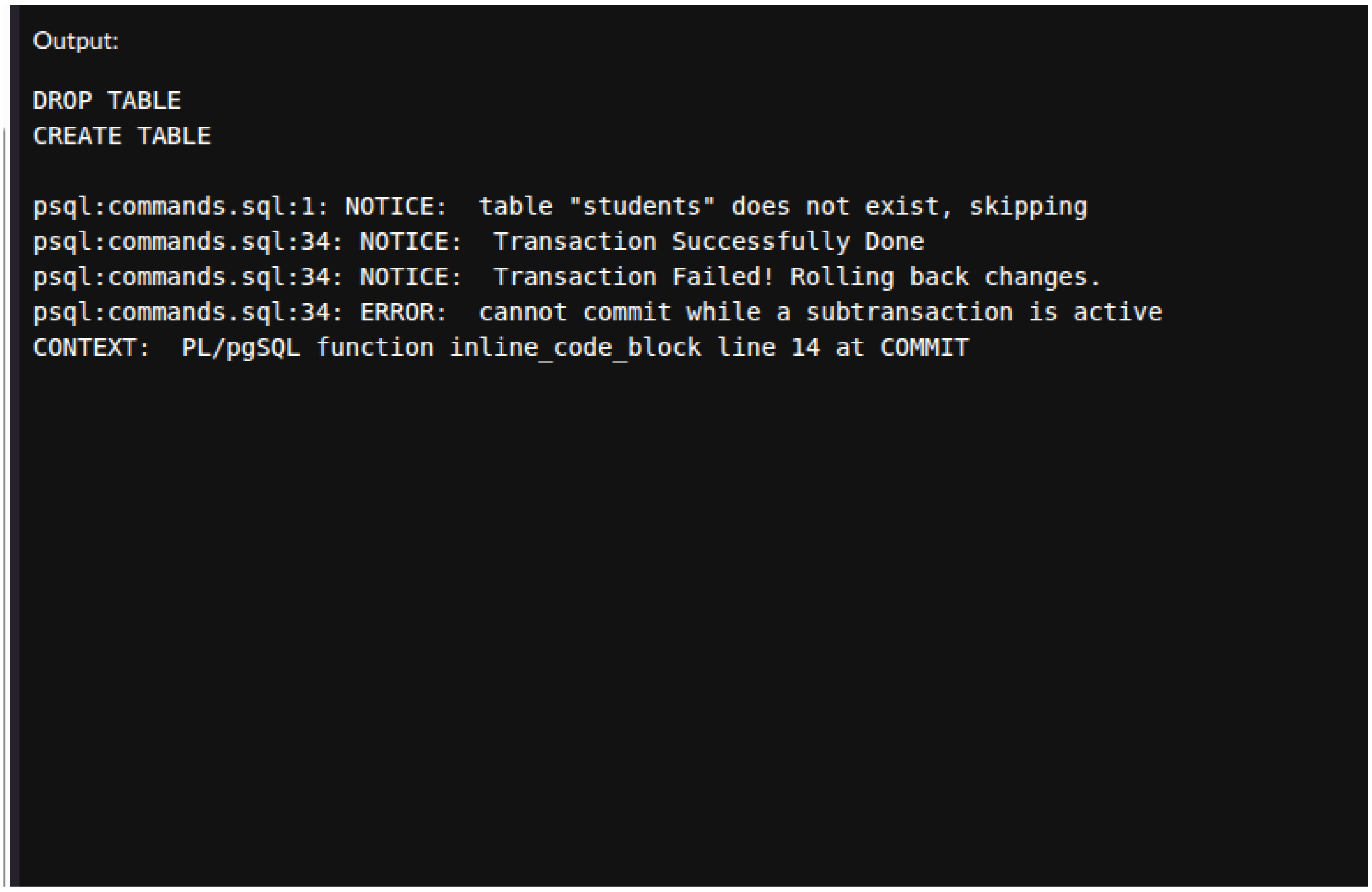
1. **Problem Title:** Transactions
2. **Problem Description:** Design a robust PostgreSQL transaction system for the students table where multiple student records are inserted in a single transaction. If any insert fails due to invalid data, only that insert should be rolled back while preserving the previous successful inserts using savepoints. The system should provide clear messages for both successful and failed insertions, ensuring data integrity and controlled error handling.

SQL Commands:

```
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;
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;
$$;
```

1. **Output:**



```
Output:

DROP TABLE
CREATE TABLE

psql:commands.sql:1: NOTICE:  table "students" does not exist, skipping
psql:commands.sql:34: NOTICE:  Transaction Successfully Done
psql:commands.sql:34: NOTICE:  Transaction Failed! Rolling back changes.
psql:commands.sql:34: ERROR:  cannot commit while a subtransaction is active
CONTEXT:  PL/pgSQL function inline_code_block line 14 at COMMIT
```

Fig1: View OUTPUT

9. Learning Outcomes:

- I learned how to create triggers.
- I learned how to perform types of triggers.