



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 8

Student Name: Jyoti Kumari
Branch: BE-CSE
Semester: 5
Subject name: ADBMS

UID: 23BCS10877
Section/Group: KRG-3B
Date of performance: 29-10-2025
Subject code: 23CSP-333

HARD LEVEL PROBLEM

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

HINT: You have to use savepoints.

CODE:

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

```
CREATE TABLE students (
```

```
    id SERIAL PRIMARY KEY,
```

```
    name VARCHAR(50) UNIQUE,
```

```
    age INT,
```

```
    class INT
```

```
);
```

```
-- ===== SUCCESSFUL SCENARIO =====
```

```
DO $$
```

```
BEGIN
```

```
    -- Start main transaction block
```

```
    RAISE NOTICE '--- Starting Transaction (Successful Scenario) ---';
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

BEGIN

INSERT INTO students(name, age, class) VALUES ('Tanmay', 18, 10);

RAISE NOTICE 'Inserted: Tanmay';

EXCEPTION WHEN OTHERS THEN

RAISE NOTICE 'Failed to insert Tanmay';

END;

BEGIN

INSERT INTO students(name, age, class) VALUES ('Aniket', 19, 8);

RAISE NOTICE 'Inserted: Aniket';

EXCEPTION WHEN OTHERS THEN

RAISE NOTICE 'Failed to insert Aniket';

END;

BEGIN

INSERT INTO students(name, age, class) VALUES ('Jyoti', 20, 9);

RAISE NOTICE 'Inserted: Jyoti';

EXCEPTION WHEN OTHERS THEN

RAISE NOTICE 'Failed to insert Jyoti';

END;

RAISE NOTICE 'Transaction Completed Successfully!';

END;

\$\$;

SELECT * FROM students;

-- ===== VIOLATED SCENARIO =====

DO \$\$

BEGIN

RAISE NOTICE '--- Starting Transaction (Violated Scenario) ---';



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

BEGIN

```
INSERT INTO students(name, age, class) VALUES ('Tanmay', 16, 8); -- duplicate (error)
```

```
RAISE NOTICE 'Inserted: Tanmay';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
RAISE NOTICE 'Failed to insert Tanmay (duplicate entry)';
```

```
END;
```

BEGIN

```
INSERT INTO students(name, age, class) VALUES ('Aniket', 19, 9); -- duplicate (error)
```

```
RAISE NOTICE 'Inserted: Aniket';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
RAISE NOTICE 'Failed to insert Aniket (duplicate entry)';
```

```
END;
```

BEGIN

```
INSERT INTO students(name, age, class) VALUES ('Jyoti', 17, 8); -- duplicate (error)
```

```
RAISE NOTICE 'Inserted: Jyoti';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
RAISE NOTICE 'Failed to insert Jyoti (duplicate entry)';
```

```
END;
```

BEGIN

```
INSERT INTO students(name, age, class) VALUES ('Keshav', 19, 9); -- success
```

```
RAISE NOTICE 'Inserted: Keshav';
```

```
EXCEPTION WHEN OTHERS THEN
```

```
RAISE NOTICE 'Failed to insert Keshav';
```

```
END;
```

```
RAISE NOTICE 'Transaction Completed (Partial Success).';
```

```
END;
```

```
$$;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

SELECT * FROM students;

```
psql:commands.sql:16: NOTICE: table "students" does not exist, skipping
psql:commands.sql:55: NOTICE: --- Starting Transaction (Successful Scenario) ---
psql:commands.sql:55: NOTICE: Inserted: Tanmay
psql:commands.sql:55: NOTICE: Inserted: Aniket
psql:commands.sql:55: NOTICE: Inserted: Jyoti
psql:commands.sql:55: NOTICE: Transaction Completed Successfully!
psql:commands.sql:96: NOTICE: --- Starting Transaction (Violated Scenario) ---
psql:commands.sql:96: NOTICE: Failed to insert Tanmay (duplicate entry)
psql:commands.sql:96: NOTICE: Failed to insert Aniket (duplicate entry)
psql:commands.sql:96: NOTICE: Failed to insert Jyoti (duplicate entry)
psql:commands.sql:96: NOTICE: Inserted: Keshav
psql:commands.sql:96: NOTICE: Transaction Completed (Partial Success).
```