**Name – Aditya Sivam Kashyap**

**UID : 23BCC70036**

**23BCC – 1 (A)**

**Experiment - 2**

**Part A — Create Department and Course Tables with Normalization (up to 3NF)**

**Aim:**  
To design and create normalized database tables (Departments and Courses) up to 3rd Normal Form (3NF) ensuring that each course belongs to exactly one department and department names are unique.

**Theory:**Normalization is the process of organizing data to reduce redundancy and improve data integrity.

* 1NF: Eliminate repeating groups, store atomic values.
* 2NF: Every non-key attribute fully depends on the whole primary key.
* 3NF: Remove transitive dependencies; non-key attributes should depend only on the key.

By separating Departments and Courses, we avoid redundancy (e.g., department name repeated for each course) and maintain referential integrity via foreign keys.

**Query:**

CREATE TABLE Departments (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50) UNIQUE NOT NULL

);

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

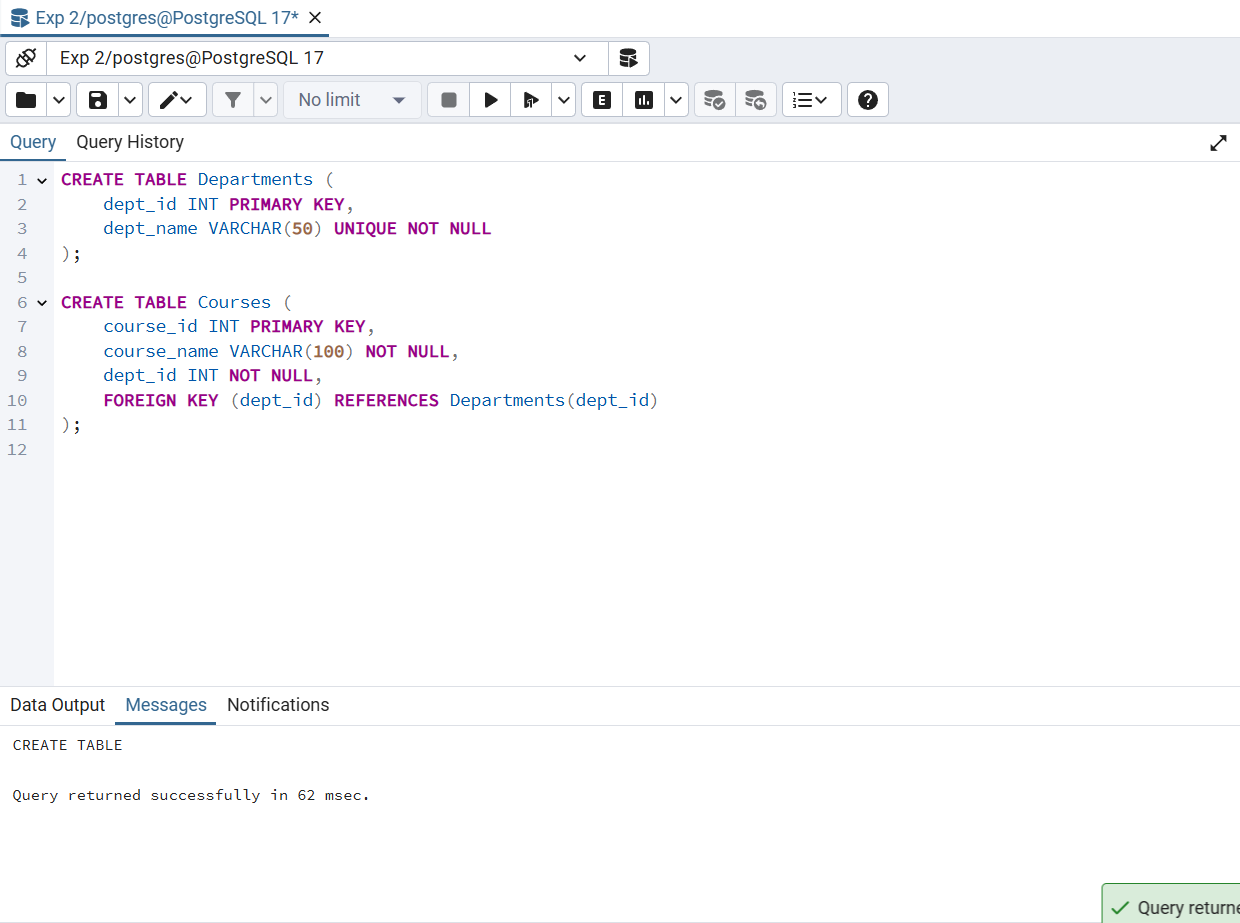
course\_name VARCHAR(100) NOT NULL,

dept\_id INT NOT NULL,

FOREIGN KEY (dept\_id) REFERENCES Departments(dept\_id)

);

**Output:**

****

**Learning Outcome:**Learned how to design normalized tables in 3NF and enforce data integrity using PRIMARY KEY, UNIQUE, and FOREIGN KEY constraints.

**Part B — Insert Sample Data into Department and Course Tables**

**Aim:**  
To insert meaningful sample data into Departments and Courses ensuring referential integrity.

**Theory:**  
In relational databases, INSERT statements are used to populate tables. Foreign keys ensure that inserted child table records refer to valid parent table records.

**Query:**

INSERT INTO Departments (dept\_id, dept\_name) VALUES

(1, 'Computer Science'),

(2, 'Electrical'),

(3, 'Mechanical'),

(4, 'Civil'),

(5, 'Electronics');

INSERT INTO Courses (course\_id, course\_name, dept\_id) VALUES

(101, 'DBMS', 1),

(102, 'Operating Systems', 1),

(111, 'Data Structures', 1),

(103, 'Power Systems', 2),

(104, 'Digital Circuits', 2),

(112, 'Electrical Machines', 2),

(105, 'Thermodynamics', 3),

(106, 'Fluid Mechanics', 3),

(113, 'Machine Design', 3),

(107, 'Structural Engineering', 4),

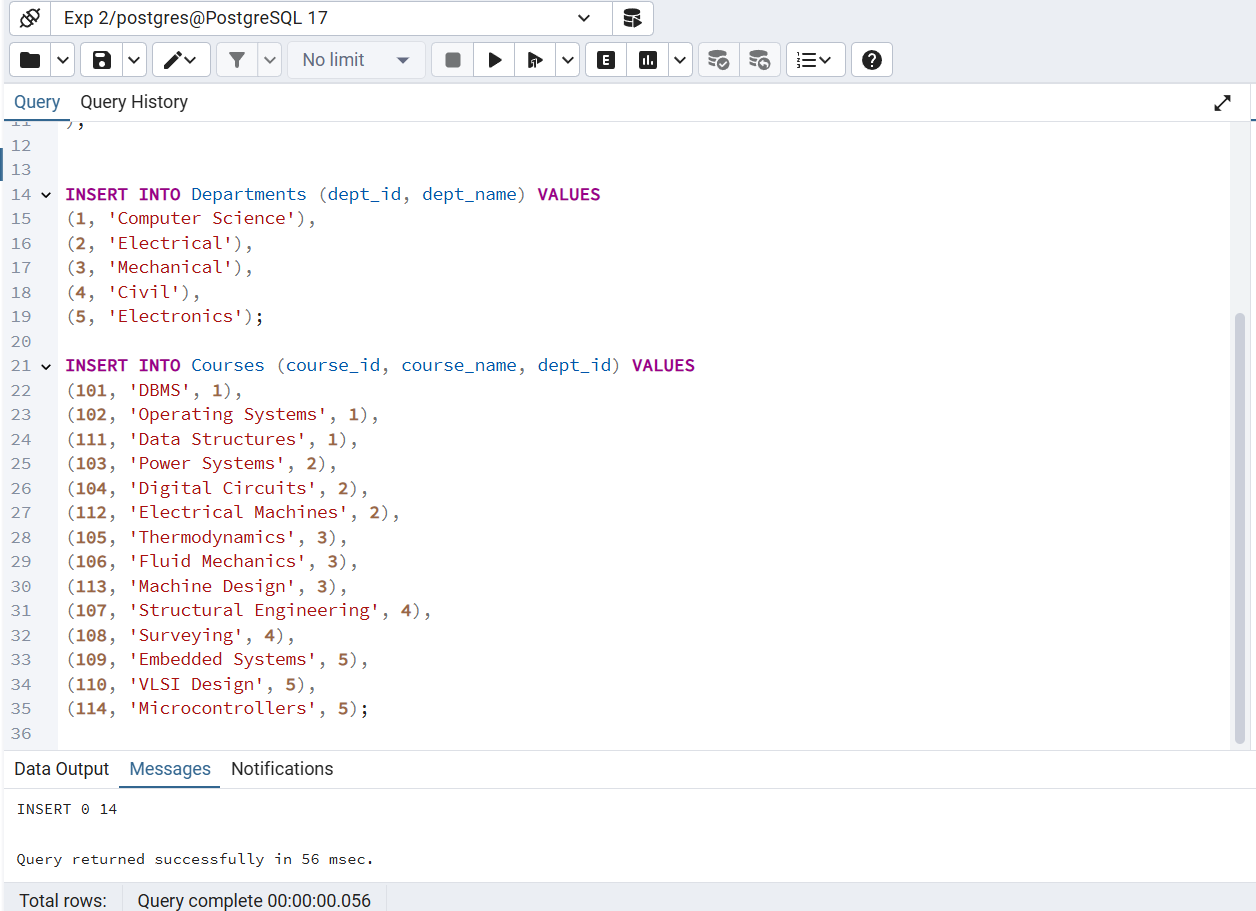
(108, 'Surveying', 4),

(109, 'Embedded Systems', 5),

(110, 'VLSI Design', 5),

(114, 'Microcontrollers', 5);

**Output:**

****

**Learning Outcome:**Understood how to insert data while maintaining referential integrity between related tables.

**Part C — Retrieve Departments Offering More Than Two Courses Using Subquery**

**Aim:**  
To retrieve the names of departments that offer more than two courses using a subquery.

**Theory:**  
A subquery (or inner query) is a query nested inside another query. Here, we use it with GROUP BY and HAVING to filter departments based on course count.

**Query:**

SELECT dept\_name

FROM Departments

WHERE dept\_id IN (

SELECT dept\_id

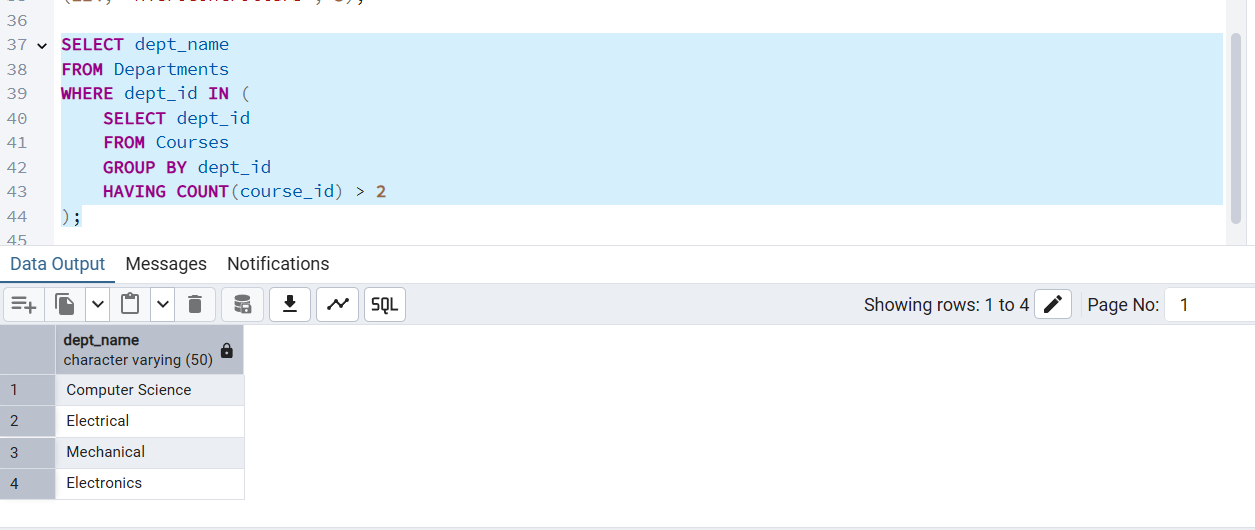
FROM Courses

GROUP BY dept\_id

HAVING COUNT(course\_id) > 2

);

**Output:**

****

**Learning Outcomes:**

Learned to use subqueries with GROUP BY and HAVING to filter aggregated data.

**Part D — Grant SELECT Access on Courses Table Using DCL**

**Aim:**  
To grant read-only access to a user for the Courses table.

**Theory:**  
Data Control Language (DCL) in SQL manages permissions.  
The **GRANT** statement allows users to perform specific operations on database objects.

**Query:**

GRANT SELECT ON Courses TO viewer\_user;

**Output:**



**Learning Outcome:**  
Understood how to use **GRANT** in DCL to control database access at table level.