Aim

The aim of this lab is to design, implement, and manage a relational database schema for an academic institution. This involves applying principles of database normalization, manipulating data, and performing complex queries and access control.

Objectives

1. **Schema Design:** Create a normalized database schema in 3NF for managing department and course information.
2. **Data Population:** Insert a substantial amount of sample data into the created tables to simulate a real-world scenario.
3. **Advanced Querying:** Formulate a subquery to retrieve specific data by joining and filtering information from multiple tables.
4. **Access Control:** Use Data Control Language (DCL) commands to manage user permissions on the database tables.

Theory

This lab is built upon several core concepts of relational database management:

- **Database Normalization (3NF):** Normalization is a process of organizing data in a database to reduce redundancy and improve data integrity. Third Normal Form (3NF) requires that a table is in 2NF and that all its columns are non-transitively dependent on the primary key. In this lab, creating separate Departments and Courses tables linked by a foreign key is a key step to achieve 3NF.
- **SQL (Structured Query Language):** The standard language for interacting with relational databases.
    - **DDL (Data Definition Language):** Commands like CREATE TABLE are used to define the database schema.
    - **DML (Data Manipulation Language):** Commands like INSERT INTO are used to add, update, or delete data.
    - **DQL (Data Query Language):** Commands like SELECT are used to retrieve data. This lab uses advanced DQL features like GROUP BY, HAVING, and subqueries.
    - **DCL (Data Control Language):** Commands like GRANT are used to control who has access to the data and what actions they can perform.

Procedure

1. **Part A - Table Creation:** Execute the CREATE TABLE statements for both

Departments and Courses. Ensure the primary and foreign key constraints are correctly defined.

2. **Part B - Data Insertion:** Run the INSERT INTO statements to populate the tables with at least 5 departments and 10 courses, ensuring that the foreign key constraint is satisfied by linking courses to existing departments.
3. **Part C - Subquery Execution:** Write and execute a SELECT statement that uses a subquery to count the number of courses per department and then filters the results to display only the departments with more than two courses.
4. **Part D - Access Control:** Execute the GRANT statement to assign SELECT permissions on the Courses table to the specified user viewer_user.

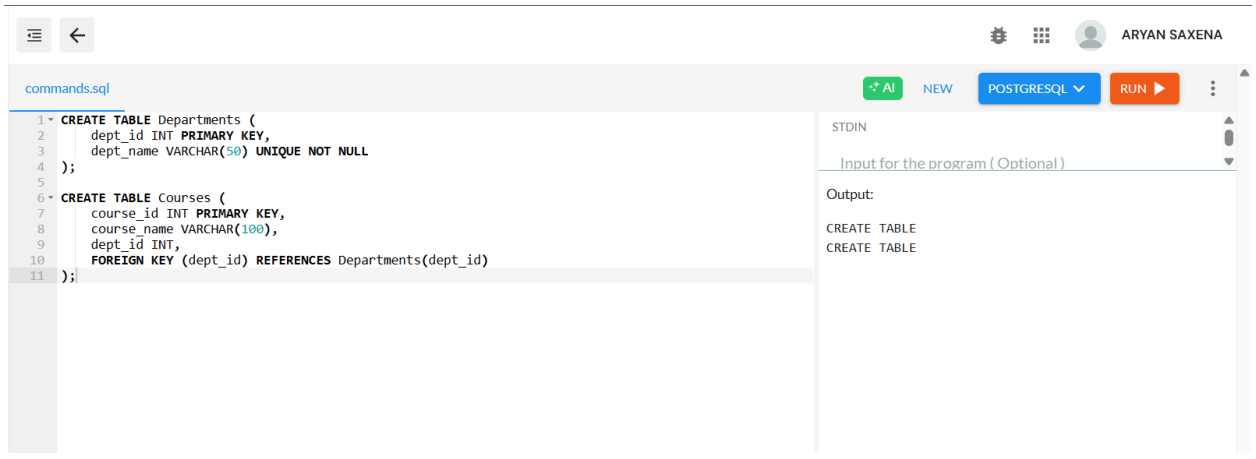
Code

```
-- Part A: Create Department and Course Tables with Normalization (3NF)
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,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);

-- Part B: Insert Sample Data into Department and Course Tables
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),
    (103, 'Algorithms', 1),
    (104, 'Power Systems', 2),
    (105, 'Digital Circuits', 2),
```

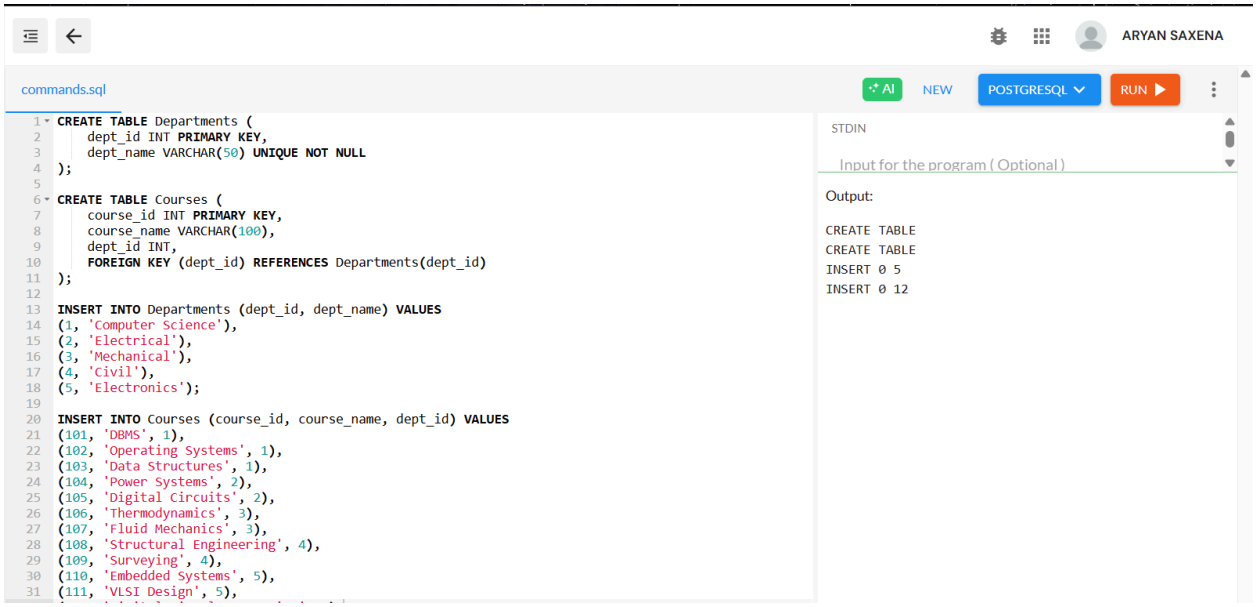
```
    (106, 'Control Systems', 2),
    (107, 'Thermodynamics', 3),
    (108, 'Fluid Mechanics', 3),
    (109, 'Structural Engineering', 4),
    (110, 'Surveying', 4),
    (111, 'Embedded Systems', 5),
    (112, 'VLSI Design', 5);

-- Part C: Retrieve Departments Offering More Than Two Courses Using Subquery
SELECT dept_name
FROM Departments
WHERE dept_id IN (
    SELECT dept_id
    FROM Courses
    GROUP BY dept_id
    HAVING COUNT(course_id) > 2
);

-- Part D: Grant SELECT Access on Courses Table Using DCL
GRANT SELECT ON Courses TO viewer_user;
```

Output

Part A

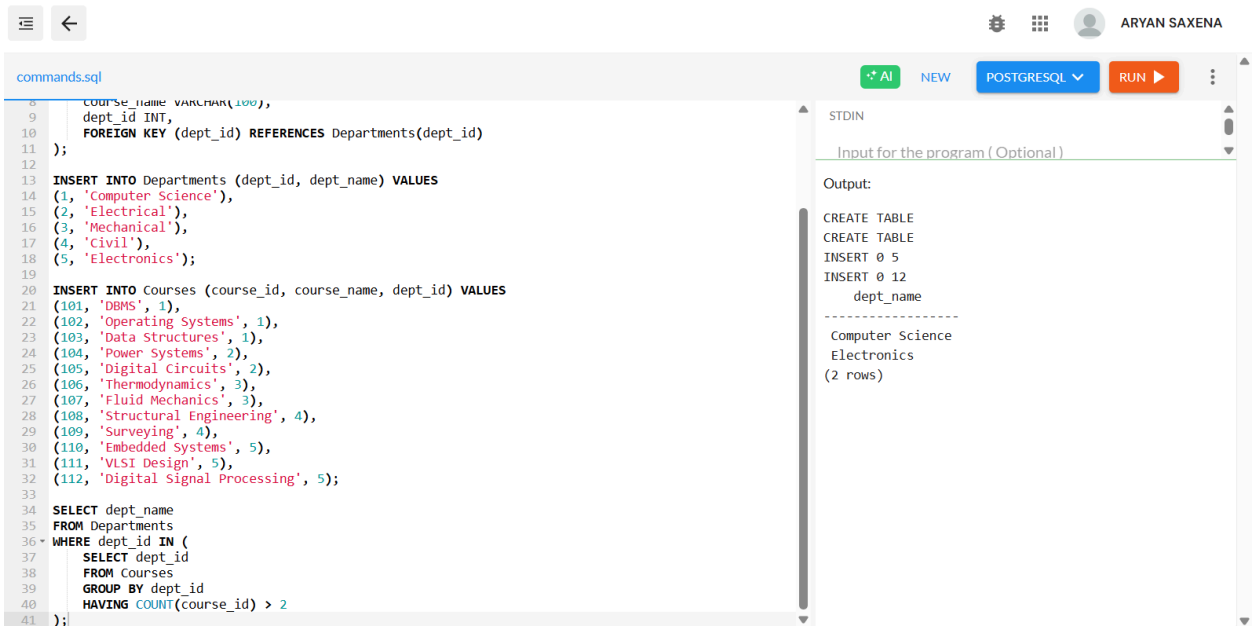


Part B



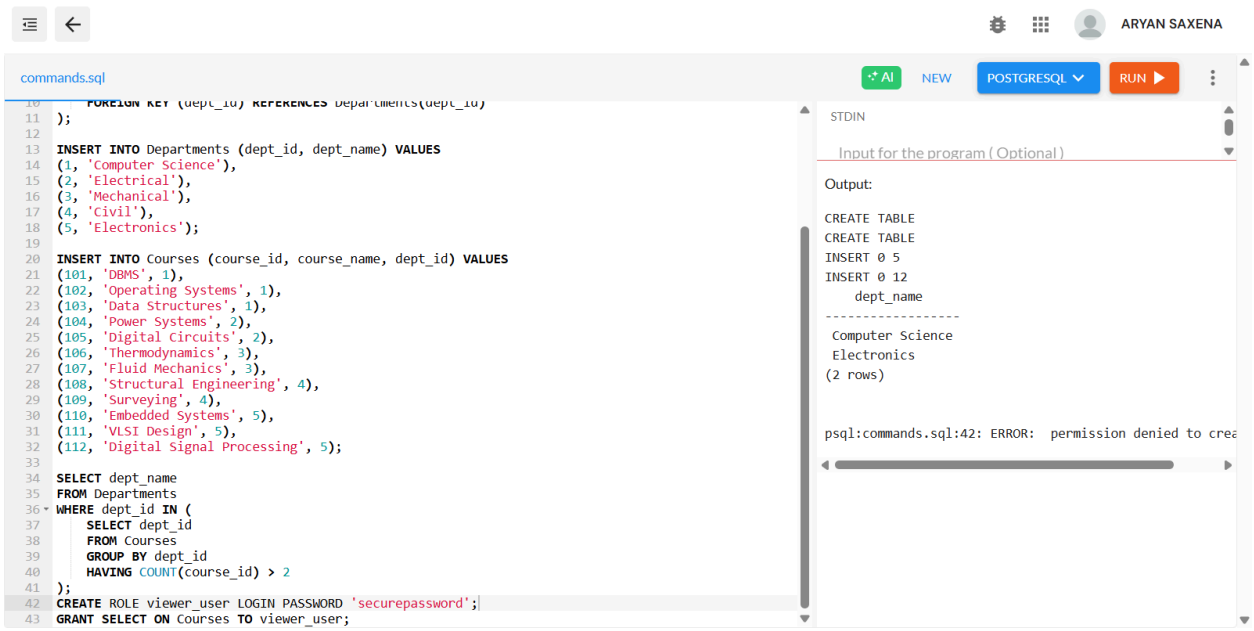

```
CREATE TABLE Departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);

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),
(103, 'Data Structures', 1),
(104, 'Power Systems', 2),
(105, 'Digital Circuits', 2),
(106, 'Thermodynamics', 3),
(107, 'Fluid Mechanics', 3),
(108, 'Structural Engineering', 4),
(109, 'Surveying', 4),
(110, 'Embedded Systems', 5),
(111, 'VLSI Design', 5),
```

Part C




```
    course_name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);

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),
(103, 'Data Structures', 1),
(104, 'Power Systems', 2),
(105, 'Digital Circuits', 2),
(106, 'Thermodynamics', 3),
(107, 'Fluid Mechanics', 3),
(108, 'Structural Engineering', 4),
(109, 'Surveying', 4),
(110, 'Embedded Systems', 5),
(111, 'VLSI Design', 5),
(112, 'Digital Signal Processing', 5);

SELECT dept_name
FROM Departments
WHERE dept_id IN (
    SELECT dept_id
    FROM Courses
    GROUP BY dept_id
    HAVING COUNT(course_id) > 2
);
```

Part D



Learning Outcomes (3)

Upon completion of this lab, students will be able to:

1. **Database Design:** Apply normalization principles to design and create a relational database schema with correct primary and foreign key constraints.
2. **Data Management:** Utilize SQL DML commands to effectively insert and manage data while respecting table relationships.
3. **Data Retrieval and Security:** Write complex queries using subqueries and GROUP BY clauses, and use DCL commands to control user access and permissions.