EXPERIMENT-02

AIM:

(i) Create Department and Course Tables with normalization up to 3NF using DDL commands
(ii) Insert Sample Records into Department and Course Tables
(iii) Retrieve Department Names Offering More Than Two Courses Using Subquery
(iv) Grant SELECT Access on Courses Table Using DCL

OBJECTIVE:

The objective of this experiment is to practice designing normalized relational schemas using 3NF, implementing referential integrity using primary and foreign keys, and working with DDL, DML, and DCL commands.
This experiment enables students to:

- Normalize data for academic course management
- Insert meaningful sample records
- Use subqueries for filtering grouped data
- Learn basic access control in SQL
  By performing this experiment locally, students will gain hands-on experience in managing relational databases for institutional data and applying subqueries and access rights effectively.

PROCEDURE:

- Launch your local SQL platform or any preferred database environment.
- Use CREATE TABLE to define the Departments table with:
- dept_id (Primary Key)
- dept_name (VARCHAR)
- Define the Courses table using CREATE TABLE with:
- course_id (Primary Key)
- course_name (VARCHAR)
- dept_id (Foreign Key referencing Departments.dept_id)
- Insert sample records into Departments using INSERT INTO with at least 5 department entries.
- Insert sample records into Courses using INSERT INTO with at least 10 course entries, ensuring each course is linked to a valid department.
- Write a SELECT query with a subquery to retrieve department names that offer more than two courses.
- Use GRANT SELECT ON Courses TO viewer_user; to allow read-only access on the Courses table for a specific user.
- Validate all results by checking relational consistency, correctness of subquery output, and successful execution of DCL command.

PROBLEM STATEMENT:

Problem Statement 1: Design a normalized academic schema for departments and courses by creating two relational tables: Departments and Courses. Each course must belong to exactly one department (one-to-many relationship). Apply normalization up to 3NF and use proper primary and foreign key constraints to maintain data integrity.

Query 1:

```
create table Departments(dept_id int primary key, dept_name varchar(50));

create table Courses(course_id int primary key, course_name varchar(100), dept_id int, foreign key(dept_id) references Departments(dept_id));
```

OUTPUT 1:

	dept_id [PK] integer	dept_name character varying (50)
1	1	Computer Science
2	2	Electrical
3	3	Mechanical
4	4	Civil
5	5	Electronics

	course_id [PK] integer	course_name character varying (100)	dept_id integer
1	101	DBMS	1
2	102	Operating Systems	1
3	103	Power Systems	2
4	104	Digital Circuits	2
5	105	Thermodynamics	3
6	106	Fluid Mechanics	3
7	107	Structural Engineering	4
8	108	Surveying	4
9	109	Embedded Systems	5
10	110	VLSI Design	5

Problem Statement 2: Insert at least 5 department records and 10 course records into the previously created tables. Ensure referential integrity by linking courses to the correct department through dept_id.

Query 2:

```
insert into Departments values(1, 'Computer Science'),(2, 'Electrical'),(3, 'Mechanical'),(4, 'Civil'),(5, 'Electronics');

insert into Courses values(101, 'DBMS', 1),(102, 'OPerating Systems', 1),(103, 'Power Systems', 2),(104, 'Digital Circuits', 2),

(105, 'Thermodynamics', 3),(106, 'Fluid Mechamics', 3),(107, 'Structural Engineering', 4),(108, 'Surveying', 4),(109, 'Embedded Systems', 5),(110, 'VLSI Design', 5);

select * from departments;

select * from courses;
```

OUTPUT 2:

	dept_id [PK] integer	dept_name character varying (50)
1	1	Computer Science
2	2	Electrical
3	3	Mechanical
4	4	Civil
5	5	Electronics

	course_id [PK] integer	course_name character varying (100)	dept_id integer
1	101	DBMS	1
2	102	Operating Systems	1
3	103	Power Systems	2
4	104	Digital Circuits	2
5	105	Thermodynamics	3
6	106	Fluid Mechanics	3
7	107	Structural Engineering	4
8	108	Surveying	4
9	109	Embedded Systems	5
10	110	VLSI Design	5

Problem Statement 3: Retrieve names of departments that offer more than two courses using a subquery.

Query 3:

```
select dept_name from Departments where dept_id in (select dept_id from Courses group by dept_id having count(*) >= 2);
```

OUTPUT 3:

	dept_name character varying (50)
1	Computer Science
2	Electrical
3	Mechanical
4	Civil
5	Electronics

Problem Statement 4: Grant read-only access to the Courses table to a user named viewer_user using DCL command.

Query 4:

```
GRANT SELECT ON Courses TO viewer_user;
```

Output 4:

```
GRANT

Query returned successfully in 51 msec.
```