Name : Saksham Rathore UID : 23BCS10814

Part A

Title: Create Department and Course Tables with Normalization (up to 3NF)

Description:

You are designing an academic schema to manage departments and the courses they offer. Normalize the design into 3NF using two tables: `Departments` and `Courses`. Ensure each course belongs to exactly one department, and department names are not duplicated.

Input Format:

- Table **Departments** with columns:
  - `dept_id` (INT, Primary Key)
  - `dept_name` (VARCHAR(50))
- Table **Courses** with columns:
  - `course_id` (INT, Primary Key)
  - `course_name` (VARCHAR(100))
  - `dept_id` (INT, Foreign Key referencing Departments)

Output Format:

No output — just successful creation of the normalized tables.

Solution

```
-- 1: Create the Departments table

CREATE TABLE Departments (

   dept_id INT PRIMARY KEY,

   dept_name VARCHAR(50) UNIQUE NOT NULL

);



--  2: Create the Courses table

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)

);
```

Part B

Title: Insert Sample Data into Department and Course Tables

Description:

After defining the schema, your task is to populate the `Departments` and `Courses` tables with at least 5 departments and 10 courses. Ensure that multiple courses are associated with each department.

Input Format:

Pre-existing **Departments** and **Courses** table structures from Problem 2A.

Output Format:

No output — just successful insertion of sample data.

Solution

```
--Insert Data into Departments Table
INSERT INTO Departments (dept_id, dept_name) VALUES
(1, 'Computer Science'),
(2, 'Mathematics'),
(3, 'Physics'),
(4, 'English Literature'),
(5, 'Economics');


--Insert Data into Courses Table
INSERT INTO Courses (course_id, course_name, dept_id) VALUES
(101, 'Data Structures', 1),
(102, 'Operating Systems', 1),
(103, 'Linear Algebra', 2),
(104, 'Calculus I', 2),
(105, 'Quantum Mechanics', 3),
(106, 'Classical Mechanics', 3),
(107, 'British Literature', 4),
(108, 'Creative Writing', 4),
(109, 'Microeconomics', 5),
(110, 'Macroeconomics', 5);
```

Part C

Title: Retrieve Departments Offering More Than Two Courses Using Subquery

Description:

Given the `Departments` and `Courses` tables, write a subquery to find the names of departments that offer more than **two courses**.

Input Format:

- Table **Departments** with columns:
  - `dept_id` (INT, Primary Key)
  - `dept_name` (VARCHAR(50))
- Table **Courses** with columns:
  - `course_id` (INT, Primary Key)
  - `course_name` (VARCHAR(100))
  - `dept_id` (INT, Foreign Key referencing Departments)

Output Format:

A list of department names (`dept_name`) that offer more than two courses.

Solution

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

Output

dept_name
Computer Science
Electronics
Mechanical
Civil

Part D

Title: Grant SELECT Access on Courses Table Using DCL

Description:

You are required to allow a user named `viewer_user` to only read the data from the `Courses` table. Use a DCL command to grant this access.

Input Format:

- Table **Courses** already exists.
- User **viewer_user** exists.

Output Format:

No output — just successful execution of a DCL command.

Solution

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