1. **Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them.**

**Answer :**

**✅ Task Overview**

1. **Design the database** using DDL statements.
2. **Apply Normalization** to make the schema efficient and reduce redundancy.
3. **Ensure data integrity** through constraints and relationships.

**📐 1. Normalization Applied**

We'll apply up to **3rd Normal Form (3NF)**:

|  |  |  |
| --- | --- | --- |
| **Step** | **Normalization Rule** | **Result** |
| **1NF** | Atomic values (no repeating groups) | Each table follows this. |
| **2NF** | Remove partial dependencies | No partial dependencies; primary keys used well. |
| **3NF** | Remove transitive dependencies | All non-key attributes depend only on the key. |

✔️ Our design is already normalized into **3NF**.

**✅ Full MySQL SQL Code (Standalone Execution)**

**-- Clean slate: Drop the database if it already exists**

**DROP DATABASE IF EXISTS CoursePlatform;**

**-- Create a fresh database**

**CREATE DATABASE CoursePlatform;**

**USE CoursePlatform;**

**-- Create Students table**

**CREATE TABLE Students (**

**student\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100) NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**age INT CHECK (age BETWEEN 15 AND 100)**

**);**

**-- Create Courses table**

**CREATE TABLE Courses (**

**course\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**course\_name VARCHAR(100) NOT NULL,**

**description TEXT**

**);**

**-- Create Enrollments table**

**CREATE TABLE Enrollments (**

**enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**student\_id INT NOT NULL,**

**course\_id INT NOT NULL,**

**enrollment\_date DATE NOT NULL,**

**FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE,**

**FOREIGN KEY (course\_id) REFERENCES Courses(course\_id) ON DELETE CASCADE**

**);**

**-- Insert Indian Students**

**INSERT INTO Students (name, email, age) VALUES**

**('Aarav Sharma', 'aarav.sharma@example.com', 20),**

**('Ananya Singh', 'ananya.singh@example.com', 22),**

**('Rohan Patel', 'rohan.patel@example.com', 21),**

**('Priya Verma', 'priya.verma@example.com', 23);**

**-- Insert Sample Courses**

**INSERT INTO Courses (course\_name, description) VALUES**

**('Database Management Systems', 'Learn relational databases, SQL, and normalization.'),**

**('Web Development', 'HTML, CSS, JavaScript, and backend development.'),**

**('Data Structures', 'Stacks, queues, linked lists, trees, and graphs.');**

**-- Insert Enrollments**

**INSERT INTO Enrollments (student\_id, course\_id, enrollment\_date) VALUES**

**(1, 1, '2024-09-01'), -- Aarav in DBMS**

**(1, 2, '2024-09-03'), -- Aarav in Web Dev**

**(2, 1, '2024-09-02'), -- Ananya in DBMS**

**(3, 3, '2024-09-05'), -- Rohan in DS**

**(4, 2, '2024-09-06'); -- Priya in Web Dev**

**-- Show final output: who is enrolled in what**

**SELECT \* FROM Students;**

**SELECT \* FROM Courses;**

**SELECT**

**e.enrollment\_id,**

**s.name AS student\_name,**

**c.course\_name,**

**e.enrollment\_date**

**FROM Enrollments e**

**JOIN Students s ON e.student\_id = s.student\_id**

**JOIN Courses c ON e.course\_id = c.course\_id;**