**Assignment: Course Management System - SQL Schema and Queries**

**Instructions:**

* Submit your SQL scripts and queries via a GitHub repository.
* Ensure that your SQL script initializes the database, handles errors, and enforces constraints.
* Write optimized queries for the given scenarios.

**Problem Statement**

Design a **Course Management System (CMS)** for an educational platform. The system should store information about **courses, students, instructors, enrollments, payments, and assessments**.

**Database Schema**

**Table: Courses**

* **CourseID** (Primary Key, int) - Unique identifier for each course.
* **CourseName** (string) - Name of the course.
* **Category** (string) - Category of the course (e.g., "Technology", "Business").
* **Duration** (int) - Duration in hours.
* **InstructorID** (Foreign Key, int) - References the Instructor teaching the course.

**Table: Instructors**

* **InstructorID** (Primary Key, int) - Unique identifier for each instructor.
* **FullName** (string) - Name of the instructor.
* **Email** (string) - Email address of the instructor.
* **Expertise** (string) - Subject expertise of the instructor.

**Table: Students**

* **StudentID** (Primary Key, int) - Unique identifier for each student.
* **FullName** (string) - Name of the student.
* **Email** (string) - Email address of the student.
* **PhoneNumber** (string) - Contact number.

**Table: Enrollments**

* **EnrollmentID** (Primary Key, int) - Unique identifier for each enrollment.
* **StudentID** (Foreign Key, int) - References StudentID from Students table.
* **CourseID** (Foreign Key, int) - References CourseID from Courses table.
* **EnrollmentDate** (datetime) - Date and time of enrollment.

**Table: Payments**

* **PaymentID** (Primary Key, int) - Unique identifier for each payment.
* **StudentID** (Foreign Key, int) - References StudentID from Students table.
* **AmountPaid** (decimal) - Amount paid for the course.
* **PaymentDate** (datetime) - Date and time of the payment.

**Table: Assessments**

* **AssessmentID** (Primary Key, int) - Unique identifier for each assessment.
* **CourseID** (Foreign Key, int) - References CourseID from Courses table.
* **AssessmentType** (string) - Type of assessment (e.g., "Quiz," "Assignment").
* **TotalMarks** (int) - Maximum marks for the assessment.

**Tasks**

**Database Initialization**

1. **Create the SQL schema** for the Course Management System with the above tables.
2. **Define constraints:**
   * Primary keys, foreign keys, and unique constraints.
   * Ensure proper data integrity (e.g., students cannot enroll in the same course multiple times).
   * Prevent duplication of records where necessary.

**SQL Query Challenges**

1. **Retrieve Available Courses:**
   * Write an SQL query to list all courses, including their **Course Name, Category, Duration, and Instructor Name**.
2. **Retrieve Students Enrolled in a Specific Course:**
   * Write a query to fetch the **Student Name, Email, and Enrollment Date** for students enrolled in a course (use a parameter for CourseID).
3. **Update Instructor Information (Stored Procedure):**
   * Create a stored procedure to **update an instructor's Full Name and Email** based on InstructorID.
4. **Calculate Total Payments per Student:**
   * Write an SQL query to retrieve the **Student Name and Total Amount Paid**.
   * Ensure that students with no payments are still included.
5. **Retrieve Students Without Enrollments:**
   * Fetch a list of students who have **not enrolled in any course**.
6. **Retrieve Monthly Revenue:**
   * Write an SQL query to **calculate total payments received per month and year**.
7. **Find Students Enrolled in More Than 3 Courses:**
   * Retrieve student details for those who have enrolled in **more than 3 courses**.
8. **Retrieve Instructor-wise Course Count:**

* List **Instructors** along with the **number of courses** they are teaching.

1. **Find Students Without Payments:**

* Write a query to retrieve students who have enrolled in at least one course but **have not made any payment**.

1. **Retrieve Courses with No Enrollments:**

* Fetch a list of **courses that have never been enrolled in**.

1. **Find the Most Popular Course:**

* Write an SQL query to **determine the course with the highest number of enrollments**.

1. **Retrieve Students and Their Total Marks in a Course:**

* Write a query that retrieves each student’s **name, course name, and their total assessment marks**.

1. **List Courses with Assessments but No Enrollments:**

* Find courses that have **assessments but no student enrollments**.

1. **Retrieve Payment Status per Student:**

* Display each student's **name, number of enrolled courses, and total amount paid**.

1. **Find Course Pairs with the Same Instructor:**

* List pairs of courses that are **taught by the same instructor**.

1. **List All Possible Student-Course Combinations:**

* Retrieve a **Cartesian product** of all students and courses (potential enrollments).

1. **Determine the Instructor with the Highest Number of Students:**

* Find the **Instructor Name and the number of students enrolled in their courses**.

1. **Trigger to Prevent Double Enrollment:**

* Create a **trigger** to prevent a student from enrolling in the same course more than once.