# University of Colombo School of Computing
## SCS 2209 - DataBase II
## Assignment

---

**Submission Guidelines: Submit a single PDF named `Index_<YourName>.pdf`, including screenshots of database creation, table structures, queries, and results, by the specified deadline. Ensure the content is well-organized and original.**

---

You are asked to design a relational database schema for an **Online Learning Platform** that will support multiple users, courses, and enrollments. The platform has three main entities: Users, Courses, and Enrollments. The Users table will store details about platform users, and each user will be uniquely identified by a user_id. The email field in the Users table should be unique and cannot be NULL. Furthermore, the password field should ensure that all passwords meet the minimum requirement of being at least 8 characters in length. The role field should categorize the user as either a Student, Instructor, or Admin, where these are the only acceptable values. The Courses table should hold information about the available courses, and each course will be uniquely identified by a course_id. A course should have a course_name that cannot be NULL and a course_fee that must be a positive value greater than zero. Additionally, there should be an enrollment_open column indicating whether a course is currently open for enrollment, with the default value set to TRUE. The Enrollments table will track which users are enrolled in which courses. Each enrollment will have a unique enrollment_id, and the combination of user_id and course_id should be unique to prevent multiple enrollments for the same user in the same course. A user can only enroll in a course if the enrollment_open field for that course is set to TRUE. Moreover, the enrollment_date should be automatically set to the current date but should not be allowed to be in the past. You must ensure that all tables are properly related through foreign keys, where user_id in the Enrollments table references the Users table, and course_id in the Enrollments table references the Courses table.

With these requirements in mind, answer the following questions:

1. Write the SQL statements to create the database schema for the Users, Courses, and Enrollments tables. Ensure all constraints are implemented correctly. This includes primary keys, unique constraints, foreign key relationships, check constraints, and default values where necessary.

2. Provide example SQL statements to insert sample data into each of the tables (Users, Courses, Enrollments). Ensure to include valid examples, such as:
    - A student enrolling in an open course.
    - An instructor with an associated course.
    - A user with a valid email and password.

Additionally, include invalid data examples to test the constraints, such as:

- A duplicate email for a user.
- A course with a negative fee.
- A user trying to enroll in a course that is closed for enrollment.
- An attempt to insert an enrollment date that is in the past.

3. Write a SQL query to retrieve a list of all courses that currently have students enrolled. The result should include the course name, course ID, and the number of students enrolled in each course. Make sure your query accounts for users who are enrolled in open courses.

4. Write a SQL query to retrieve all instructors and the courses they are associated with. The result should show the instructor's user_id, name, and the course_name for each course that the instructor teaches.

5. Write a SQL query that finds all courses with open enrollments and returns the total number of enrollments for each course. This query should only include courses that are still open for enrollment (enrollment_open = TRUE).

6. Write a query to list all students who are currently enrolled in a course, including the student's user_id, their name, the course name, and the enrollment_date. Be sure the query is only returning students enrolled in courses that are open for enrollment

7. Create a trigger or check constraint to ensure that the enrollment_date in the Enrollments table is never earlier than today's date. If an attempt is made to insert an enrollment with a past date, it should fail.

8. Modify the Enrollments table schema to include a new column status, which will store the status of the enrollment (e.g., Active, Completed, Dropped). Write the SQL statements to update the Enrollments table and implement any necessary default values or constraints on the new column.

9. Write a stored procedure that allows an instructor to **add a new student** to a course. The procedure should check if the student is already enrolled in the course and, if not, add the student to the Enrollments table. If the student is already enrolled, the procedure should update the enrollment status to Active.

10. Create a trigger that will **automatically update the course fee** in the Courses table when the fee for a course is modified. The trigger should log the change (previous fee, new fee, and timestamp) into an Audit_Log table.

11. Write a stored function that calculates the **average grade** for a student across all courses they have completed. The function should return the average grade for a given user_id.

12. Create a trigger that **prevents deletion of a course** if there are students currently enrolled in that course. The trigger should raise an error message saying, "Cannot delete course with active enrollments."

13. Write a stored procedure that **updates the status** of a student's enrollment to "Completed" for a particular course once the student has completed the course. The procedure should be able to check the student's completion status.

14. Create a stored function that returns the **total number of students** enrolled in a particular course. The function should take the course_id as an argument and return the count of students currently enrolled.

15. Write a trigger to **automatically assign a default status** of "Active" to all new enrollments. This should be executed after a new record is inserted into the Enrollments table.

---

End