MySql Exercise

DDL (Data Definition Language) Questions

- Create Table: Write a SQL query to create a table named students with the following columns:
    - student_id (INT, Primary Key, Auto Increment)
    - first_name (VARCHAR(50))
    - last_name (VARCHAR(50))
    - birthdate (DATE)
    - email (VARCHAR(100))
    - Alter Table: Write a SQL query to add a column named phone_number (VARCHAR(15)) to the students table.

- Drop Table: Write a SQL query to drop the students table.

- Add Constraint: Write a SQL query to add a unique constraint on the email column of the students table.

- Rename Table: Write a SQL query to rename the students table to university_students.

DML (Data Manipulation Language) Questions 1 :

- **Insert Data**: Write a SQL query to insert a new student into the students table with the following details:
    - first_name: John
    - last_name: Doe
    - birthdate: 1999-05-15
    - email: john.doe@example.com
    - Update Data: Write a SQL query to update the email of the student with student_id 1 to john.newemail@example.com.

- **Delete Data**: Write a SQL query to delete the student record with student_id 1.

- **Select Data**: Write a SQL query to select all columns from the students table where the last_name is 'Doe'.

- **Create Tables**: Create table named courses with the following columns :
    - course_id (INT, Primary Key, Auto Increment)
    - course_name (VARCHAR(100))
    - Give 5 user entries into the table.

- Another table named enrollments with the following columns :
    - student_id (INT, Foreign Key referencing students(student_id))
    - course_id (INT, Foreign Key referencing courses(course_id))
    - Write a SQL query to list all students and the courses they are enrolled in.

Additional Practice Topics

1. **Create Index**: Write a SQL query to create an index on the last_name column of the students table.

2. **Aggregate Functions**: Write a SQL query to count the number of students born after the year 2000.

3. **Group By**: Write a SQL query to group students by their birth year and count the number of students in each group.

Projects In MySql

Basic Projects

1. **Problem Statement - Student Database:**
    a. You are required to develop a Student Database Management System in Python using MySQL.

- **Tasks :**
    - **Create Tables –** Users should be able to define a table with a maximum of 3 fields, specifying field names and their data types (INT or VARCHAR(255)).
    - **Insert Records –** Users should be able to insert values into an existing table. The program should fetch the table structure dynamically.
    - **View Records –** Users should be able to display all records from a selected table.
    - **List All Tables –** The program should retrieve and display all tables in the connected database.
    - **Exit the Program –** The user should be able to exit via a menu-driven system.

- **Requirements:**
    - Use MySQL Connector to establish a connection to a database.
    - Implement error handling for invalid inputs (e.g., non-existing tables, incorrect data types).
    - Ensure user input for table and column names is validated to prevent SQL injection.
    - Provide a menu-based interface for user interaction.

- **Constraints :**
    - The program should support only up to 3 fields per table.
    - Data types allowed: INT and VARCHAR(255).
    - The program should not crash if the user enters incorrect input.

2. **Simple To-Do List Application**

- **Features :**
    - Add tasks with titles and descriptions.
    - View all tasks in a list.
    - Update task status (completed/incomplete).
    - Delete tasks.
- **Tech Stack :**
    - Programming Language: Python
    - Framework: Flask or Django (for web interface)
    - Database: MySQL
    - Libraries: MySQL Connector/Python, Flask-WTF (for forms)

3. **Library Management System:** Develop a system to manage book inventory, including adding, updating, and deleting book records.

- **Objective** : Develop a system to manage book inventory, including adding, updating, and deleting book records.

- **Description** :
    - Database Schema:
        - Table Name: books
        - Columns:
            - book_id (INT, Primary Key, Auto Increment)
            - title (VARCHAR(150))
            - author (VARCHAR(100))
            - published_year (YEAR)
            - genre (VARCHAR(50))
            - quantity (INT)
- **Features :**
    - Add new books with details (title, author, ISBN).
    - Check out books to users.
    - Return books and update their status.
    - View all available books.
- **Tech Stack :**
    - Programming Language: Python
    - Framework: Flask or Django
    - Database: MySQL
    - Libraries: MySQL Connector/Python

- **Tasks** :
    - Create Database and Table: Write SQL commands to create the books table with the specified schema.
    - Insert Data: Add sample book records to the table.
    - Read Data: Retrieve book records from the table.
    - Update Data: Update the details of a book record.
    - Delete Data: Remove a book record from the table.

4. **Simple Blog:** Build a simple blog where users can create, read, update, and delete posts.

- Objective: Build a simple blog where users can create, read, update, and delete posts.

- Description :
    - Database Schema :
        - Table Name : posts
        - Columns :
            - post_id (INT, Primary Key, Auto Increment)
            - title (VARCHAR(200))
            - content (TEXT)
            - author (VARCHAR(100))
            - created_at (TIMESTAMP, Default CURRENT_TIMESTAMP)

- Tasks :
    - Create Database and Table: Write SQL commands to create the posts table with the specified schema.
    - Insert Data: Add sample blog post records to the table.
    - Read Data: Retrieve blog post records from the table.
    - Update Data: Update the details of a blog post.
    - Delete Data: Remove a blog post from the table.

Intermediate Projects

1. **E-commerce Database:** Design a MySQL database for an e-commerce application, including tables for products, customers, orders, and reviews.

Objective: Design a MySQL database for an e-commerce application, including tables for products, customers, orders, and reviews.

- Description :
    - Database Schema:
        - Table Name: products
            - product_id (INT, Primary Key, Auto Increment)
            - name (VARCHAR(100))
            - description (TEXT)
            - price (DECIMAL(10, 2))
            - stock_quantity (INT)
        - Table Name: customers
            - customer_id (INT, Primary Key, Auto Increment)
            - first_name (VARCHAR(50))
            - last_name (VARCHAR(50))
            - email (VARCHAR(100), Unique)
            - address (VARCHAR(200))
        - Table Name: orders
            - order_id (INT, Primary Key, Auto Increment)
            - customer_id (INT, Foreign Key referencing customers(customer_id))
            - order_date (TIMESTAMP, Default CURRENT_TIMESTAMP)
            - total_amount (DECIMAL(10, 2))
        - Table Name: order_items
            - order_item_id (INT, Primary Key, Auto Increment)
            - order_id (INT, Foreign Key referencing orders(order_id))
            - product_id (INT, Foreign Key referencing products(product_id))
            - quantity (INT)
            - price (DECIMAL(10, 2))
        - Table Name: reviews
            - review_id (INT, Primary Key, Auto Increment)
            - product_id (INT, Foreign Key referencing products(product_id))
            - customer_id (INT, Foreign Key referencing customers(customer_id))
            - rating (INT, Check rating between 1 and 5)
            - comment (TEXT)

- Tasks :
    - Create Database and Tables: Write SQL commands to create the necessary tables with the specified schema.

    - Insert Data: Add sample data to each table.
    - Read Data: Retrieve data from the tables, including products, customers, orders, and reviews.
    - Update Data: Update records in the tables (e.g., updating product prices or customer information).
    - Delete Data: Remove records from the tables (e.g., deleting a customer or an order).

- Skills Practiced :
    - Designing relational databases
    - Writing complex SQL queries
    - Creating and managing foreign key relationships

2. **Employee Management System :** Create a system to manage employee records, including attendance, payroll, and performance evaluations.

- Objective: Create a system to manage employee records, including attendance, payroll, and performance evaluations.

- Description :
    - Database Schema :
        - Table Name: employees
            - employee_id (INT, Primary Key, Auto Increment)
            - first_name (VARCHAR(50))
            - last_name (VARCHAR(50))
            - email (VARCHAR(100), Unique)
            - position (VARCHAR(50))
            - salary (DECIMAL(10, 2))
        - Table Name: attendance
            - attendance_id (INT, Primary Key, Auto Increment)
            - employee_id (INT, Foreign Key referencing employees(employee_id))
            - date (DATE)
            - status (ENUM('Present', 'Absent', 'Leave'))
        - Table Name: payroll
            - payroll_id (INT, Primary Key, Auto Increment)
            - employee_id (INT, Foreign Key referencing employees(employee_id))
            - pay_date (DATE)
            - amount (DECIMAL(10, 2))
        - Table Name: performance
            - performance_id (INT, Primary Key, Auto Increment)
            - employee_id (INT, Foreign Key referencing employees(employee_id))
            - review_date (DATE)
            - score (INT, Check score between 1 and 10)
            - comments (TEXT)

- Tasks :
    - Create Database and Tables: Write SQL commands to create the necessary tables with the specified schema.
    - Insert Data: Add sample data to each table.

    - Read Data: Retrieve data from the tables, including employee records, attendance, payroll, and performance evaluations.
    - Update Data: Update records in the tables (e.g., updating employee information or attendance status).
    - Delete Data: Remove records from the tables (e.g., deleting an employee record).

- Skills Practiced :
    - Designing relational databases
    - Writing complex SQL queries
    - Creating and managing foreign key relationships

3. **Event Management :** Develop a database to manage events, including attendee registration, scheduling, and feedback collection.

- Objective : Develop a database to manage events, including attendee registration, scheduling, and feedback collection.

- Description :
    - Database Schema :
        - Table Name : events
            - event_id (INT, Primary Key, Auto Increment)
            - name (VARCHAR(100))
            - date (DATE)
            - location (VARCHAR(100))
            - description (TEXT)
        - Table Name : attendees
            - attendee_id (INT, Primary Key, Auto Increment)
            - first_name (VARCHAR(50))
            - last_name (VARCHAR(50))
            - email (VARCHAR(100), Unique)
        - Table Name : registrations
            - registration_id (INT, Primary Key, Auto Increment)
            - event_id (INT, Foreign Key referencing events(event_id))
            - attendee_id (INT, Foreign Key referencing attendees(attendee_id))
            - registration_date (TIMESTAMP, Default CURRENT_TIMESTAMP)
        - Table Name : feedback
            - feedback_id (INT, Primary Key, Auto Increment)
            - event_id (INT, Foreign Key referencing events(event_id))
            - attendee_id (INT, Foreign Key referencing attendees(attendee_id))
            - rating (INT, Check rating between 1 and 5)
            - comments (TEXT)

- Tasks :
    - Create Database and Tables: Write SQL commands to create the necessary tables with the specified schema.
    - Insert Data: Add sample data to each table.
    - Read Data: Retrieve data from the tables, including events, attendees, registrations, and feedback.
    - Update Data: Update records in the tables (e.g., updating event details or attendee information).

    - Delete Data: Remove records from the tables (e.g., deleting an event or feedback).

- Skills Practiced :
    - Designing relational databases
    - Writing complex SQL queries
    - Creating and managing foreign key relationships

Advanced Projects

1. Healthcare Management System: Design and implement a comprehensive database for managing patient records, appointments, and billing in a healthcare facility.
2. Travel Booking System: Develop a MySQL database for managing flights, hotels, and car rentals, including booking and cancellation functionalities.
3. Social Media Platform: Create a database for a social media platform, including user profiles, posts, comments, and friend connections.