Project

Using SQLite or Mysql…etc

1. Hospital Management System

Tables:

- **Patients:** patient_id, name, age, gender, contact_info, admission_date
- **Doctors:** doctor_id, name, specialization, contact_info, department
- **Appointments:** appointment_id, patient_id, doctor_id, appointment_date, status

Use Cases:

- Retrieve all patients admitted under a specific doctor.
- Identify appointments within a date range.
- Find patients who have missed appointments.

---

2. Company Employee Management

Tables:

- **Employees:** employee_id, name, designation, department, salary
- **Departments:** department_id, name, location
- **Projects:** project_id, name, deadline, department_id

Use Cases:

- List employees working on projects in a specific department.
- Update project deadlines and track overdue projects.
- Identify employees earning above a certain threshold.

---

3. Online Retail Store

Tables:

- **Products:** product_id, name, category, price, stock
- **Customers:** customer_id, name, email, phone_number

- **Orders:** order_id, customer_id, product_id, order_date, quantity

Use Cases:

- Generate sales reports for specific products or categories.
- Identify customers who made repeat purchases.
- Analyze stock levels to determine reordering requirements.

---

4. University Management System

Tables:

- **Students:** student_id, name, age, major, enrollment_date
- **Courses:** course_id, name, credits, instructor
- **Enrollments:** enrollment_id, student_id, course_id, grade

Use Cases:

- Identify students enrolled in a specific course.
- Calculate the average grade for a course.
- Update course details and track student progress.

Implementation Plan:

1. **Create Schema and ERD:**
    - Design an Entity-Relationship Diagram (ERD) showing the relationships among the tables.
    - Implement the schema creation in SQL (CREATE TABLE commands).
2. **Create Tables and Relationships:**
    - Define tables with primary keys, foreign keys, and constraints.
    - Insert rows in tables.
3. **Update Tables:**
    - Use UPDATE to modify records (e.g., update a patient's status or order quantity).
4. **Select with Multiple Conditions:**

- Use SELECT queries with WHERE, logical operators (AND, OR), and filtering.
- Update Values in rows .

5. **Joins:**
    - Perform inner joins to combine related data (e.g., patients with doctors).
    - Use left or right joins for advanced queries.
6. **Multiple Joins:**
    - Query data from three or more tables (e.g., orders linked with customers and products).
7. **Subqueries:**
    - Use subqueries for advanced filtering or calculations (e.g., find customers who ordered most).

Python and Pandas:

1. **Read Data:**
    - Use pandas.read_sql() to load data from the database.
2. **Manipulate Data:**
    - Add columns: df['new_column'] = df['existing_column'] * 2
    - Delete columns: df.drop(columns=['column_name'], inplace=True)
3. **Indexing:**
    - Access rows using loc and iloc for specific slicing.
4. **Export Data:**
    - Export modified data using df.to_csv() or df.to_excel().
5. **Upload All Work In Github**