**2. Automate the creation of a data model using Python**

* **Level 1:**  
  Write a Python script using SQLAlchemy ORM to define a complete database schema for an e-commerce platform, including tables for users, products, orders, and reviews. Automate migrations with Alembic.
* **Level 2:**  
  Create a Python program that uses SQLAlchemy to define and create tables for a student registration system (students, courses, registrations). Include basic constraints and relationships.
* **Level 3:**

Write a simple Python script that creates two tables (students and courses) in SQLite using raw SQL commands executed from Python.

**Process:**

**LEVEL:-1**

**Create a full database schema for an e-commerce platform (users, products, orders, reviews) using Python and raw SQL (with MySQL).**

**Theoretical Step-by-Step Guide:**

1. **Understand the Business Model**
   * Users place orders
   * Orders contain products
   * Users can review products
   * Orders and products are connected via order\_items
2. **Design Your Entity Tables**
   * **Users**: user\_id, name, email
   * **Products**: product\_id, name, price
   * **Orders**: order\_id, user\_id, order\_date
   * **Order\_Items**: id, order\_id, product\_id, quantity, price
   * **Reviews**: review\_id, user\_id, product\_id, rating, comment
3. **Define Relationships**
   * One user can place many orders → FK in orders
   * One order can contain many products → M:M via order\_items
   * A user can write reviews → FK in reviews
   * One product can have many reviews
4. **Choose the Database**
   * Use **MySQL** for a production-level model
   * Connect using **mysql-connector-python** in Python
5. **Plan Data Types & Constraints**
   * Use INT AUTO\_INCREMENT PRIMARY KEY for IDs
   * Use VARCHAR for strings
   * Use DECIMAL for prices
   * Add **FOREIGN KEYS** to enforce referential integrity
6. **Set Up MySQL Connection in Python**
   * Use mysql.connector.connect() with proper credentials
7. **Write Raw SQL CREATE TABLE Queries**
   * Write individual queries for each table with proper constraints
   * Run them using a cursor object in Python
8. **Commit and Close**
   * Commit the changes to save the schema
   * Close the connection after success
9. **(Optional)** Automate migrations
   * You can write logic to compare existing tables and make schema changes (manually or via .sql scripts)

**LEVEL:2**

**Create a student registration system with relationships using Python and SQLite.**

**Theoretical Step-by-Step Guide:**

1. **Understand the Objective**  
   Create a system with these entities:
   * students: who register
   * courses: that students register for
   * registrations: to record who enrolled in what
2. **Choose the Database**  
   Use **SQLite**, suitable for small systems and easy testing.
3. **Design the Schema**
   * **Students Table**: student\_id, name, email
   * **Courses Table**: course\_id, course\_name, department
   * **Registrations Table**: registration\_id, student\_id, course\_id, registration\_date
4. **Define Relationships**
   * **One-to-Many**: One student can register for many courses
   * **Many-to-Many**: Managed using a separate registrations table
   * Use **FOREIGN KEY** to enforce relationships
5. **Plan Data Types & Constraints**
   * Use AUTOINCREMENT for IDs
   * Use TEXT for names/emails
   * Apply **UNIQUE** constraint on email
   * Use FOREIGN KEY references in the registrations table
6. **Write SQL Statements in Python**  
   Use Python's sqlite3 to define and run your schema queries.
7. **Test the Model (Optional)**  
   Insert a few sample students, courses, and registrations to test it works.
8. **Close the Connection**

**LEVEL:-3**

**Create two tables (students and courses) in SQLite using raw SQL and Python.**

**Theoretical Step-by-Step Guide:**

1. **Understand the Objective**  
   You are building two simple tables:
   * students: stores student details
   * courses: stores course details
2. **Choose the Database**  
   Use **SQLite**, which is embedded in Python and does not require installation.
3. **Design the Tables**
   * **Students Table**: fields → student\_id, name, age
   * **Courses Table**: fields → course\_id, course\_name, credits
4. **Plan the Data Types**
   * Use INTEGER for IDs and age
   * Use TEXT or VARCHAR for names
   * Use PRIMARY KEY to uniquely identify each row
5. **Connect to the Database**  
   Use Python’s sqlite3 library to create or connect to a .db file.
6. **Write SQL CREATE TABLE Queries**  
   Use raw SQL queries inside Python to define table structures.
7. **Execute the Queries**  
   Use a cursor to execute the queries.
8. **Save the Changes**  
   Call commit() to save the schema in the database.
9. **Close the Connection**  
   Always close the connection using conn.close().

**Expected Output**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Output** | **Key Tables** | **Relationships** | **Constraint Checks** |
| Level:-3 | students, courses created in SQLite | 2 | None | Basic PK |
| Level:-2 | students, courses, registrations in SQLite | 3 | 1:M via FK | PK + FK |
| Level:-1 | Full e-commerce schema in MySQL | 5 | Complex M:M, 1:M | PK + FK + Constraints |