**Case Study: E-Commerce Application Using Python and MySQL**

### ****1. Project Title****

**E-Commerce Application using Python and MySQL**

### ****2. Project Purpose and Background****

In the digital era, shopping habits have shifted to online platforms. However, small businesses and students often struggle to access or develop custom online ordering systems that allow users to browse products, add them to cart, and place orders.  
This project simulates a basic E-Commerce application using Python for backend logic and MySQL for data persistence.

### ****3. Problem Statement****

Traditional manual methods of order-taking result in time consumption, errors, and lack of proper tracking. This project solves that by providing a structured, database-driven system for handling customer orders.

### ****4. Project Objectives (SMART Goals)****

* **S**pecific: To design and develop a working mini e-commerce backend system.
* **M**easurable: Success measured by ability to add/view/order products from CLI.
* **A**chievable: Using Python and MySQL only.
* **R**ealistic: Focused on backend structure and logic.
* **T**ime-bound: Completed within project submission schedule.

### ****5. Methodology Chosen****

**Agile-inspired Iterative Development** was used:

* Weekly tasks with module-wise development (Customer → Product → Cart → Order).
* Tested after each implementation.
* Modular and flexible to enhancement.

### ****6. Requirements Gathering & Analysis****

#### **Functional Requirements**

* Register/Login as a customer.
* View products and check availability.
* Add products to cart.
* Place an order.
* Track order with details.

#### **Non-functional Requirements**

* Data integrity and referential control using SQL constraints.
* Real-time data interaction with backend.
* Menu-driven CLI experience.

### ****7. Schema Design****

The relational schema is designed based on **3NF** (Third Normal Form) and follows referential integrity.

#### **1. Customers Table**

| **Column** | **Type** | **Constraint** |
| --- | --- | --- |
| customer\_id | INT | Primary Key |
| name | VARCHAR | Not Null |
| email | VARCHAR | Unique |
| password | VARCHAR | Not Null |

#### **2.** products **Table**

| **Column** | **Type** | **Constraint** |
| --- | --- | --- |
| product\_id | INT | Primary Key |
| name | VARCHAR | Not Null |
| price | DECIMAL | Not Null |
| description | TEXT | Optional |
| stockQuantity | INT | Not Null |

#### **3.** cart **Table**

| **Column** | **Type** | **Constraint** |
| --- | --- | --- |
| cart\_id | INT | Primary Key |
| customer\_id | INT | Foreign Key → customers |
| product\_id | INT | Foreign Key → products |
| quantity | INT | Not Null |

#### **4.** orders **Table**

| **Column** | **Type** | **Constraint** |
| --- | --- | --- |
| order\_id | INT | Primary Key |
| customer\_id | INT | Foreign Key → customers |
| order\_date | DATE | Not Null |
| total\_price | DECIMAL | Computed |
| shipping\_address | VARCHAR | Not Null |

#### **5.** order**\_items Table**

| **Column** | **Type** | **Constraint** |
| --- | --- | --- |
| order\_item\_id | INT | Primary Key |
| order\_id | INT | Foreign Key → orders |
| product\_id | INT | Foreign Key → products |
| quantity | INT | Not Null |

### ****8. SQL Operations and Commands Used****

#### **DDL (Data Definition Language)**

* CREATE DATABASE, CREATE TABLE
* Primary key, foreign key constraints
* AUTO\_INCREMENT for IDs

**DML (Data Manipulation Language)**

* INSERT INTO, UPDATE, DELETE, SELECT
* JOIN queries to fetch combined data from multiple tables
* GROUP BY, ORDER BY to organize results

#### **SQL Constraints Used**

* PRIMARY KEY, FOREIGN KEY
* NOT NULL, UNIQUE
* AUTO\_INCREMENT (for ID generation)
* Referential constraints

### ****9. Technical Terminologies****

| **Term** | **Meaning** |
| --- | --- |
| **CRUD** | Basic database operations (Create, Read, Update, Delete) |
| **DAO Pattern** | Design structure separating business logic from database access |
| **JOIN** | Combines rows from two or more tables |
| **Normalization** | Reducing data redundancy |
| **Referential Integrity** | Ensures that foreign keys point to valid primary keys |
| **Foreign Key Constraint** | Prevents orphan records by enforcing link to another table |
| **Modular Design** | Separates code into logical reusable parts |
| **Unit Testing** | Validates each module with test cases |

### ****10. Challenges Faced and Solutions****

| **Challenge** | **Solution** | |
| --- | --- | --- |
| Handling foreign key relationships | | Added constraints manually and used ON DELETE CASCADE logic |
| Joining multiple tables for order tracking | | Used INNER JOIN queries |
| Validating stock before placing order | | Implemented quantity checks in Python before INSERT |

### ****11. Future Enhancements****

* Add web interface using Flask or Django
* Add payment gateway integration (dummy for now)
* Enable PDF invoice generation
* Role-based access: Admin vs Customer

### ****12. Conclusion****

This E-Commerce application demonstrates how a backend-driven product ordering system can be designed using real-world relational principles. The project includes proper schema design, validation logic, user operations, and order tracking. It combines **Python programming, database operations, exception handling**, and **project design principles**—making it a well-rounded software engineering learning experience.