National Economics University

College of Technology

E-commerce Order Manager

11247334 - Pham Le Minh Nhat
11247261 - Nguyen Viet Anh
11247298 - Nguyen Quang Huy

Supervisor: Dr. Hung Tran

A report submitted as the final assignment for the course
Introduction to Database Systems

December 11, 2025

Abstract

This paper presents the design, implementation, and evaluation of a relational database system for E-commerce Order Management. The core contribution of this work is a fully normalised database schema, developed through a systematic process from UNF to 3NF, which ensures data integrity by eliminating redundancy and update anomalies. The conceptual design is formalised via an Entity-Relationship Diagram (ERD) modelling key entities—Customers, Products, Orders, and Order Details—and their relationships. A functional Graphical User Interface (GUI) prototype demonstrates practical data manipulation, while a suite of SQL queries validates the system's ability to support essential business operations, such as order tracking and revenue analysis. Comprehensive testing, including functional, integrity, and relationship tests, confirms the correctness of the implemented relational constraints and workflows. Although the project successfully meets its primary pedagogical objectives, critical analysis identifies notable limitations, including a basic GUI, the absence of advanced database programming constructs (e.g., triggers, stored procedures), and a fundamental lack of security mechanisms. The paper concludes by delineating clear directions for future research, emphasising the imperative to integrate authentication, encapsulate business logic within the database layer, and enhance scalability to transition the prototype into a robust, production-ready system.

Keywords: E-commerce, Database Design, Normalisation, Entity-Relationship Diagram (ERD), SQL, Relational Database Management System (RDBMS), Order Management System, Data Integrity.

Contents

Chapter 1 Introduction

1.1 Project Context

In the contemporary digital marketplace, the efficiency and reliability of an e-commerce platform's backend infrastructure are paramount to its operational success. A foundational component of such a system is the order management database, which orchestrates critical transactions between customers, products, and orders. However, many small to medium-sized enterprises often begin with ad hoc data storage solutions, such as a single, unnormalised spreadsheet or table. This approach, while simple to initiate, rapidly introduces significant operational liabilities. Data redundancy, update anomalies, and inconsistent reporting emerge, leading to inefficiencies, potential revenue leakage, and impaired decision-making. This project addresses a classic yet persistent problem in information systems: the transition from a fragile, flat-file data model to a robust, normalised relational database. The given scenario presents an unnormalised order table with defined functional dependencies, serving as a realistic case study for applying formal database design principles to build a scalable and maintainable e-commerce order manager.

1.2 Objectives

The primary objective of this project is to design, implement, and validate a fully functional database application that resolves the inherent flaws of the initial unnormalised data structure. This goal is decomposed into specific, measurable outcomes:

1. **Data Modelling & Normalisation:** To systematically transform the provided Unnormalised Form (UNF) table into a schema adhering to the Third Normal Form (3NF). This process involves eliminating repeating groups (1NF), removing partial dependencies (2NF), and eliminating transitive dependencies (3NF), thereby ensuring data integrity and minimising redundancy.

2. **Database Implementation:** To translate the normalised logical design into a physical MySQL database schema, complete with proper table definitions, primary keys (PK), foreign keys (FK), and data constraints.

3. **Application Development:** To engineer a complete Python-based application with an intuitive Graphical User Interface (GUI). This application will provide full Create,

Read, Update, and Delete (CRUD) functionality for all core entities, execute essential business intelligence queries, and feature an analytical dashboard with visualisations.

4. **Engineering & Collaboration:** To employ professional software engineering practices, including version control via GitHub with demonstrated collaboration through Issues and Pull Requests, and to ensure code quality and project portability.

5. **Documentation & Communication:** To comprehensively document the process and outcomes through a LaTeX-based technical report, a presentation deck, and a detailed demonstration video, effectively communicating the technical rationale and functional capabilities of the developed system.

1.3 Scope

The scope of this project is deliberately bounded to focus on core database and application development principles. The system will manage the central entities of an e-commerce operation: Customers, Products, Orders, and Order Details. Functionality will encompass complete lifecycle management for these entities, the execution of specified join and aggregation queries, and basic data visualisation. Key deliverables include the normalised SQL schema, the seeded database, the Python application, and all associated documentation.

Explicitly out of scope are advanced features such as user authentication and authorisation, integration with payment gateways or third-party APIs, management of discounts, taxes, or shipping logistics, and deep performance tuning or security hardening beyond basic SQL constraints. This focus allows for a rigorous exploration of fundamental database design and application connectivity within a manageable framework, establishing a robust foundation upon which such advanced features could be subsequently integrated.

Chapter 2 Normalization

2.1 Context

The normalisation process for the sales management system database is undertaken to systematically eliminate data redundancy, prevent update anomalies, ensure consistency, and establish a structurally efficient relational schema. The database encompasses entities such as customers, products, categories, suppliers, employees, orders, and order details. This chapter delineates the stepwise refinement of each entity from its initial Unnormalised Form (UNF) through the First, Second, and Third Normal Forms, ensuring compliance with established relational design principles.

2.2 Unnormalised Form

The unnormalised relation is defined as follows:

UNF(OrderID, CustomerID, CustomerName, ProductID, ProductName, Quantity, Price, OrderDate, OrderStatus)

The functional dependencies identified for this relation are:

- CustomerID → CustomerName
- ProductID → {ProductName, Price}
- OrderID → {CustomerID, OrderDate, OrderStatus}
- {OrderID, ProductID} → Quantity

At this stage, the relation permits repeating groups, as a single order may contain multiple products, each associated with its own quantity. This violates the atomicity requirement of the relational model and obscures the underlying structure of the data.

2.3 First Normal Form

A relation is in First Normal Form (1NF) if all attributes contain atomic values and no repeating groups are present. To achieve 1NF, each combination of *OrderID* and *ProductID* is

represented as a distinct tuple, ensuring that product-level data within an order is stored row by row.

The resulting 1NF relation is expressed as:

$$\text{OrderLine}_{1NF}(\text{OrderID, CustomerID, CustomerName, ProductID, ProductName, Quantity, Price, OrderDate, OrderStatus})$$

The composite primary key of this relation is $\{OrderID, ProductID\}$. Although the relation now satisfies 1NF, it still exhibits significant redundancy, as customer, order, and product information is repeatedly stored for each product in an order.

2.4 Second Normal Form

Second Normal Form (2NF) requires that a relation be in 1NF and that no non-prime attribute is partially dependent on a proper subset of a composite primary key. In the 1NF relation, several attributes depend solely on either *OrderID* or *ProductID*, rather than on the entire composite key.

Specifically, the dependencies

$$\text{OrderID} \rightarrow \{\text{CustomerID, OrderDate, OrderStatus}\}$$

and

$$\text{ProductID} \rightarrow \{\text{ProductName, Price}\}$$

constitute partial dependencies and therefore violate 2NF.

To eliminate these violations, the relation is decomposed into the following relations:

$$\text{Orders(OrderID, CustomerID, OrderDate, OrderStatus)}$$

$$\text{Products(ProductID, ProductName, Price)}$$

$$\text{Quantities(OrderID, ProductID, Quantity)}$$

$$\text{Customers(CustomerID, CustomerName)}$$

Each non-key attribute in these relations is now fully functionally dependent on its respective primary key.

2.5 Third Normal Form

A relation is in Third Normal Form (3NF) if it is in 2NF and contains no transitive dependencies, meaning that non-key attributes do not depend on other non-key attributes. An examination of the decomposed relations shows that all transitive dependencies have been removed.

In the *Customers* relation, *CustomerName* depends solely on the primary key *CustomerID*. Similarly, in the *Products* relation, both *ProductName* and *Price* depend exclusively on *ProductID*. The remaining relations also satisfy the requirements of 3NF.

The final schema in Third Normal Form is summarised in Table 2.1.

Relation	**Primary Key**	**Attributes**
Customers	CustomerID	CustomerName
Products	ProductID	ProductName, Price
Orders	OrderID	CustomerID, OrderDate, OrderStatus
Quantities	{OrderID, ProductID}	Quantity

Table 2.1: Final 3NF schema

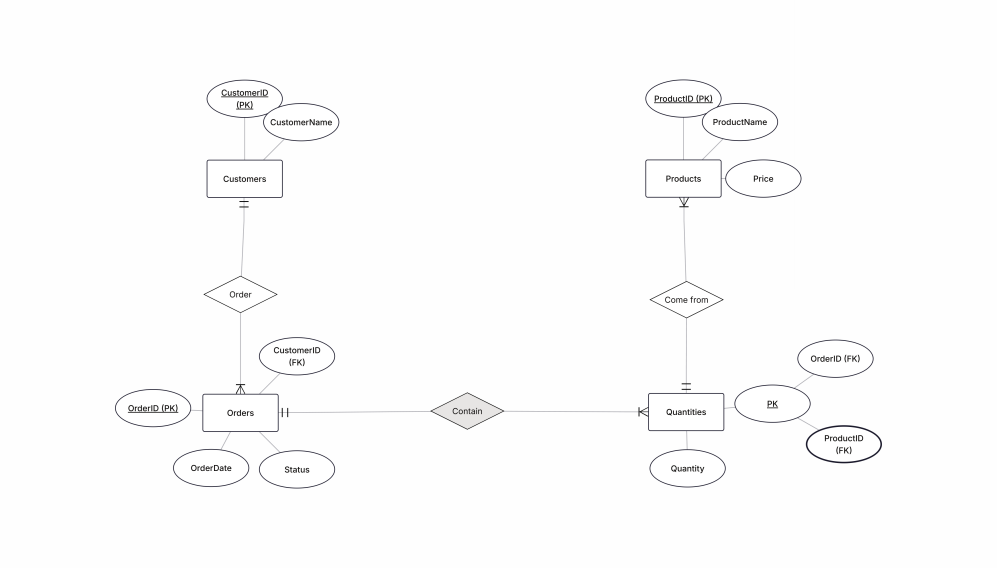
2.6 Conclusion

The normalisation of the order data model from UNF to 3NF exposes the conceptual structure embedded in the original dataset. By systematically applying the rules of normalisation and decomposing the relation according to its functional dependencies, redundancy is minimised and data integrity is enhanced.

The resulting 3NF schema ensures that updates to customer, product, or order information are performed in a single location, thereby preventing inconsistencies. Consequently, the final design provides a robust and semantically clear foundation for the implementation of an order management database system.

Chapter 3 Entity–Relationship Diagram (ERD)

3.1 Context



This chapter presents the Entity–Relationship Diagram (ERD) constructed from the functional dependencies identified within the sales processing system. The schema centres around four core entities: Customers, Products, Orders, and Quantities. Each entity captures a specific aspect of the business process: customers interact with the system by placing orders; products represent the catalogue of items available for purchase; orders record customer-initiated transactions; and quantities serve as the structural bridge linking orders and products. Together, they form a coherent model that captures both the flow of sales activity and the relationships necessary for maintaining referential integrity in the database.

3.2 Description of the ERD

3.2.1 customers

The Customers entity stores identifying information for each individual who interacts with the system. Its structure is defined by *CustomerID*, which serves as the primary key, and the

corresponding *CustomerName*. The functional dependency

$$\text{CustomerID} \rightarrow \text{CustomerName}$$

ensures that each identifier consistently maps to a single customer entry. Since a customer may place multiple orders over time, this entity naturally forms a one-to-many relationship with the Orders entity.

3.2.2 orders

The Orders entity represents all transactions registered in the system. It is defined by the primary key *OrderID* and includes the attributes *OrderDate* and *Status*, which document the time and state of each transaction. The attribute *CustomerID* functions as a foreign key referencing the Customers entity. This structure reflects the functional dependency

$$\text{OrderID} \rightarrow \{\text{CustomerID}, \text{OrderDate}, \text{Status}\},$$

indicating that all recorded order details are fully determined by the order identifier. An order may contain multiple products, requiring a linking structure provided by the Quantities entity.

3.2.3 products

The Products entity represents the inventory of goods available for purchase. It is identified by its primary key *ProductID*, with the associated attributes *ProductName* and *Price*. The functional dependency

$$\text{ProductID} \rightarrow \{\text{ProductName}, \text{Price}\}$$

ensures that product information is uniquely determined by its identifier. Because a product may appear in numerous orders, its relationship with the ordering process is facilitated through its participation in the Quantities table.

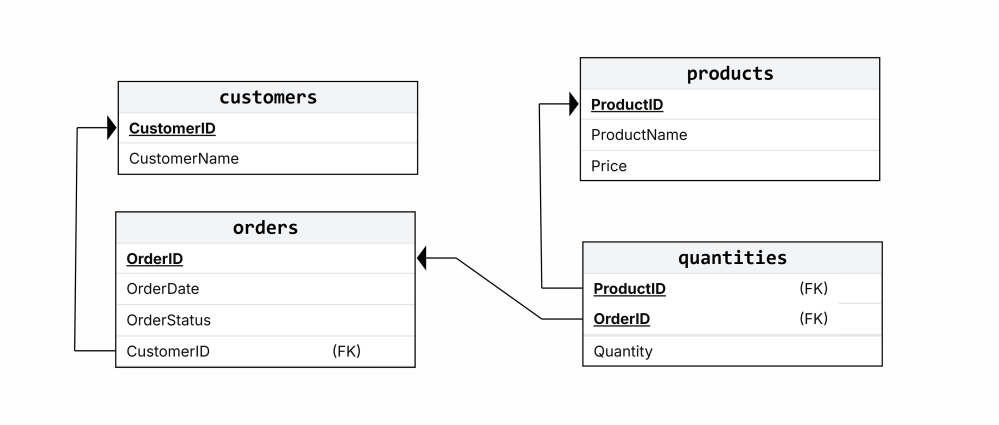
3.2.4 quantities

The Quantities entity functions as an intersection table that resolves the many-to-many relationship between Orders and Products. It is defined by the composite primary key $(\text{OrderID}, \text{ProductID})$, which uniquely identifies each product entry within a given order. The accompanying attribute *Quantity* specifies the number of units of a product that appear in the order. The functional dependency

$$\{\text{OrderID}, \text{ProductID}\} \rightarrow \text{Quantity}$$

ensures that the quantity associated with a specific product in a specific order is uniquely determined by the composite key.

3.3 Relationships



3.3.1 customers–orders

The relationship between Customers and Orders is defined by the fact that a single customer may place multiple orders, while each order corresponds to exactly one customer. This creates a one-to-many relationship, represented by the *CustomerID* foreign key in the Orders entity.

3.3.2 orders–quantities

Each order may include multiple product entries, resulting in several corresponding rows in the Quantities entity. Since each quantity record references exactly one order, the resulting structure forms a one-to-many relationship facilitated by the *OrderID* attribute.

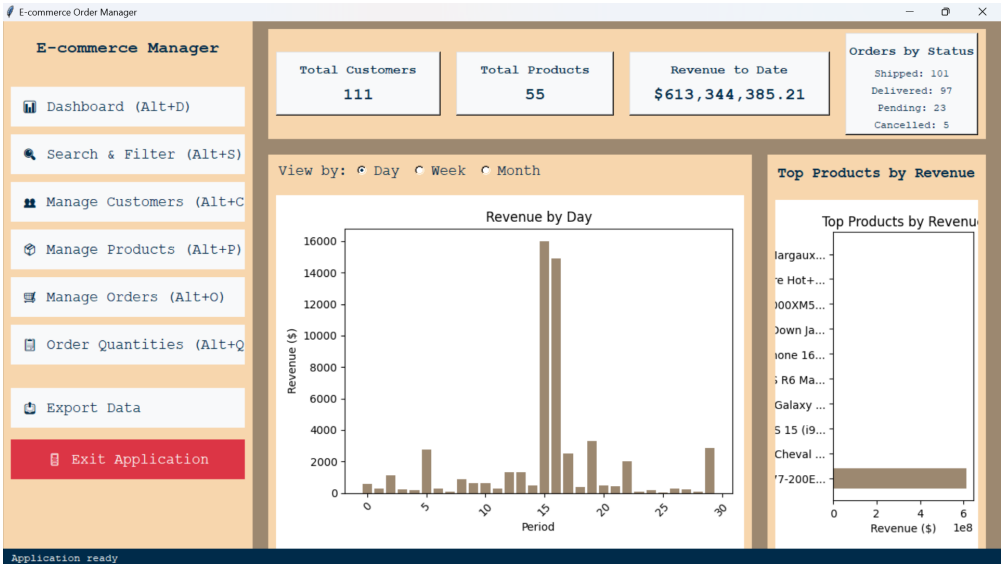
3.3.3 products–quantities

A product may appear in many different orders, and each such appearance is represented as an entry in the Quantities table. Because each quantity record references exactly one product, this relationship is also one-to-many, with *ProductID* ensuring referential integrity.

Chapter 4 Graphical User Interface (GUI)

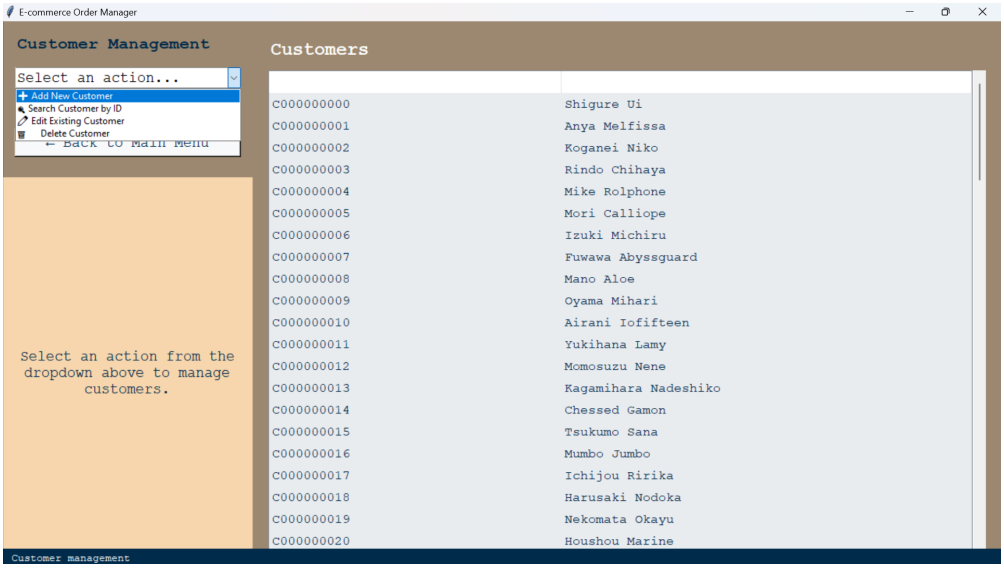
4.1 Context

The GUI demonstrates how users interact with the system to perform operations that correspond to the underlying database structure. Screenshots help verify that the interface supports all necessary workflows.



4.1.1 Customer Interface

Displays customer information and supports adding, editing, managing records, and search customer by ID.

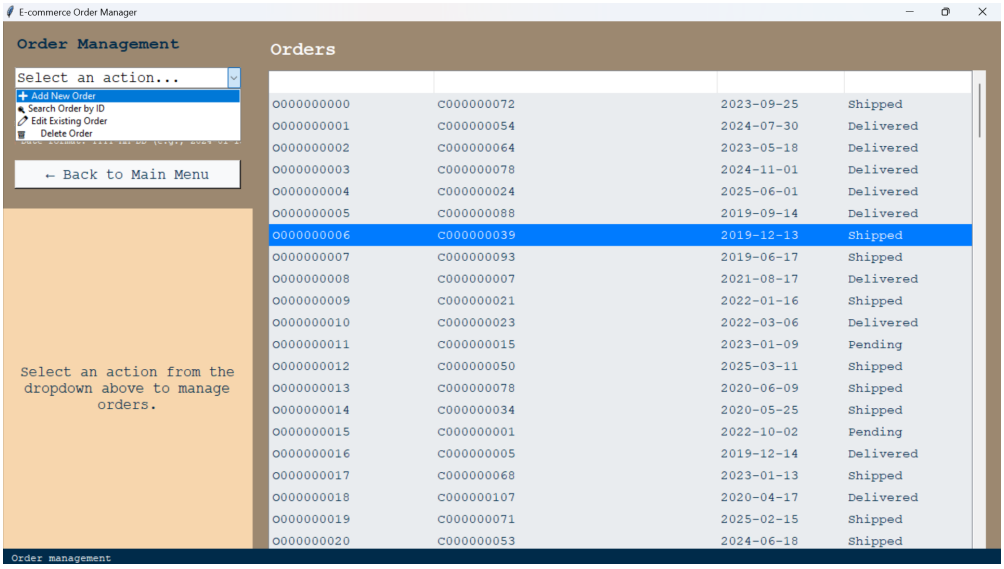


4.1.2 Product Interface

Shows product details, and also supports adding, editing, deleting, and search product by ID like the customer interface.

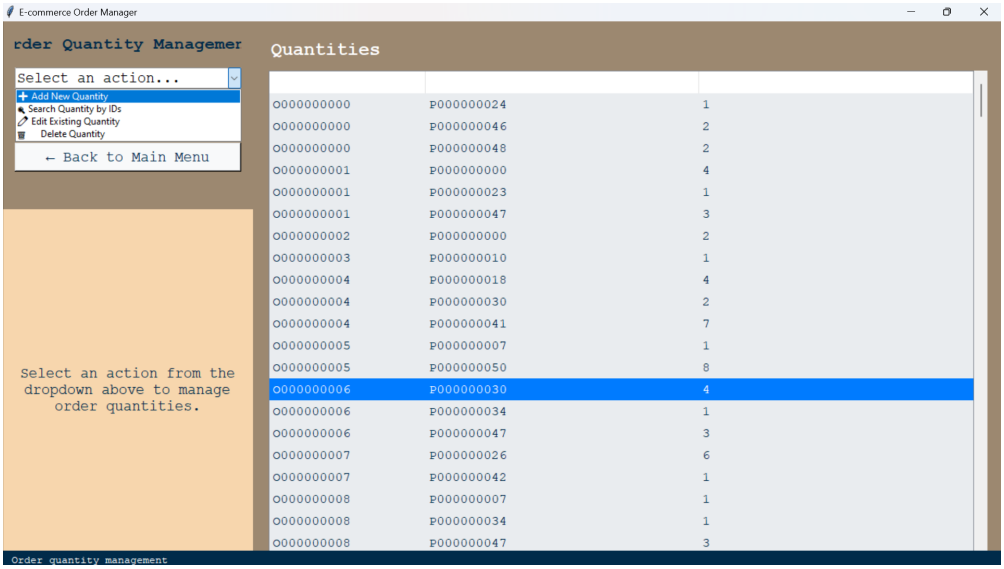
4.1.3 Order Interface

Allows users to create new orders, search by ID, and editing.



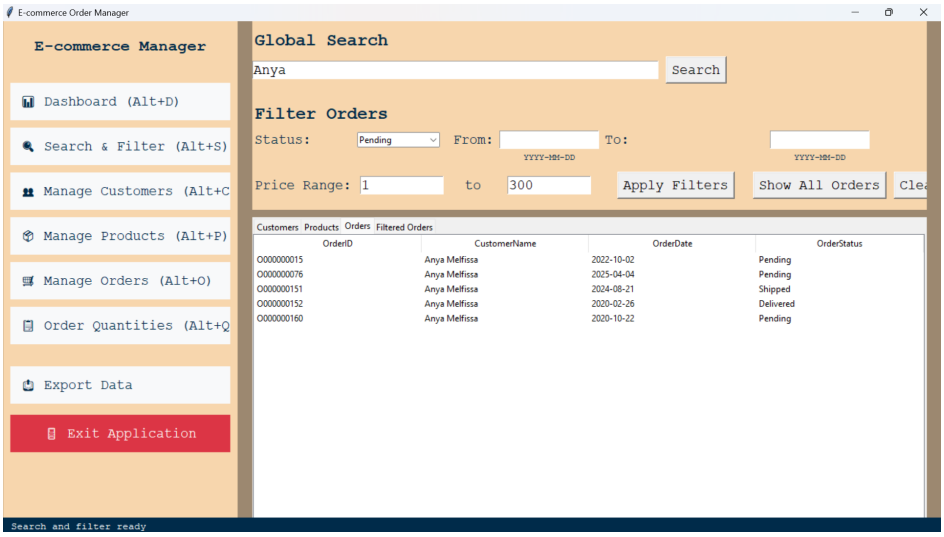
4.1.4 Order Quantity Interface

Shows the quantity of the products, also supports like products interface.

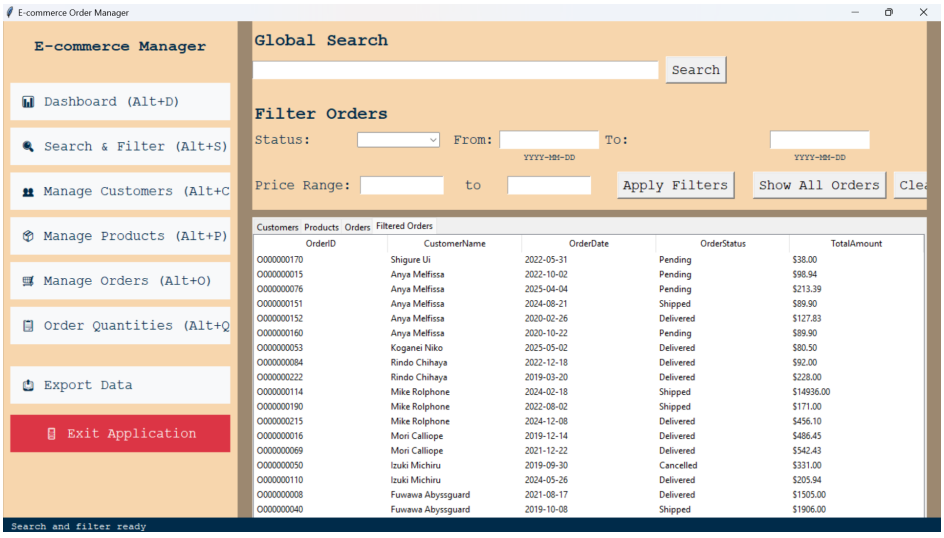


Chapter 5 SQL Queries

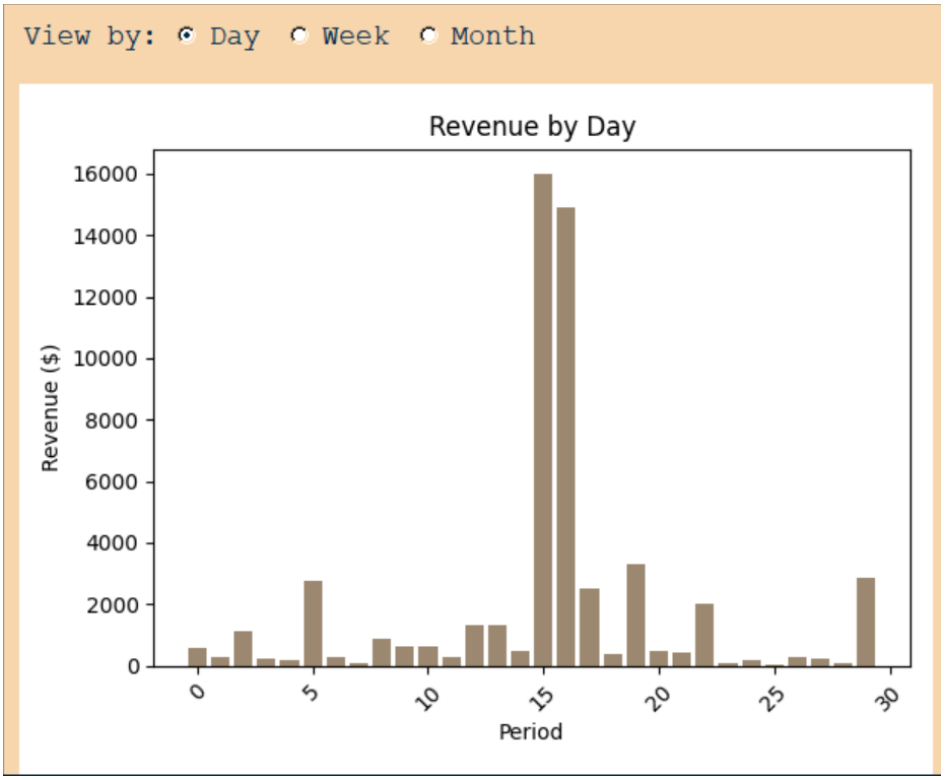
5.1 INNER JOIN



5.2 Multi-table JOIN



5.3 Order summary (revenue)



Chapter 6 Results Interpretation

6.1 Context

This chapter provides an interpretation of the results obtained from executing queries and analysing the data stored within the database. The purpose of this analysis is to determine whether the implemented database design not only ensures structural correctness but also supports efficient and meaningful information retrieval. By examining the resulting outputs, it becomes possible to evaluate how effectively the schema captures real-world interactions and whether it provides reliable insights for practical operational use.

6.2 Customer-Related Insights

The analysis of customer-related data reveals patterns in purchasing behaviour and activity levels among individual users of the system. Query results consistently highlight the number of orders placed by each customer, the frequency with which transactions are carried out, and the extent to which customers participate in higher-value purchases. Together, these findings demonstrate that the relationship between Customers and Orders operates as intended. The one-to-many structure linking these entities allows the system to trace customer activity accurately and to generate customer-specific summaries that reflect their engagement with the platform.

6.3 Product-Related Insights

The evaluation of product data offers further insight into how the system manages inventory information and purchase tendencies. The results frequently illustrate which products are most often selected by customers, revealing purchasing preferences and trends. They also confirm that price information is applied consistently across orders and that stock movement is reflected transparently through recurring product entries. These observations indicate that the link between the Products entity and the Quantities table is functioning correctly. The database reliably associates products with their occurrences in orders, enabling coherent tracking of item popularity and pricing behaviour.

6.4 Order and Revenue Insights

A broader examination of order-related results uncovers meaningful patterns regarding overall transaction volume and revenue generation. The data reveals not only the total number of orders recorded in the system but also the average size of these orders and the distribution of their monetary values. Through these results, it becomes evident that the relationship between Orders and Quantities is accurately implemented. The system successfully aggregates transaction details and reflects real customer activity, ensuring that calculated indicators such as revenue totals, order size, and purchase distribution closely align with the stored operational data.

6.5 Workflow Observations

In contexts where additional system users or workflow components are involved, the query results also offer insight into how different roles interact with the database. These findings may reflect the efficiency with which orders are processed, the consistency of data entry practices, or the presence of missing or irregular data points. Such observations, though dependent on the specific system configuration, help assess whether the database design adequately supports real-world operational workflows and highlights potential areas for optimisation or correction.

Chapter 7 Testing

7.1 Context

Testing is an essential phase in the development lifecycle of the database system, undertaken to validate that the implemented schema, constraints, relationships, and operational workflows function as intended. This chapter outlines the comprehensive testing procedures applied to ensure the reliability, integrity, and performance of the e-commerce order management system.

7.2 Functional Testing

Functional testing was conducted to verify that all core operations of the system produce accurate and consistent outcomes. This encompassed the insertion of new records across entities such as customers, products, and orders, as well as the updating and deletion of existing entries. Each operation was examined to confirm that data modifications were correctly reflected across related tables and that system behaviour remained consistent with defined business rules. Additionally, validation rules—including mandatory fields, numeric positivity for price and quantity, and permissible status values—were rigorously enforced to ensure data quality at the point of entry.

7.3 Integrity Testing

Integrity testing focused on the enforcement of structural and referential constraints within the database. This included verifying that primary key uniqueness is maintained, foreign key relationships are properly upheld, and that constraints such as *NOT NULL* and *UNIQUE* are respected. Specific scenarios were tested, such as attempting to create an order without a valid customer reference or inserting duplicate key values, to confirm that the database correctly rejects invalid operations. These tests ensure the preservation of referential integrity and data consistency throughout the system.

7.4 Relationship Testing

The relational design of the database necessitated thorough verification of entity relationships, including one-to-many and many-to-many associations. Testing confirmed that a single customer can place multiple orders and that the same product can appear in numerous distinct

orders. Each relationship was evaluated to ensure that joins and foreign key references perform correctly and that data retrieval across related tables yields logically consistent results, thereby affirming the stability of the relational model.

7.5 Edge Case Testing

To assess the robustness of the system under atypical or boundary conditions, edge case testing was performed. Scenarios included processing orders with extremely high or zero quantities, managing customers with no recorded purchase history, and handling products that appear across a large number of orders. Additionally, tests were conducted to evaluate system behavior when encountering missing or incomplete data, such as null values in required fields. These examinations ensure that the database and application remain stable and predictable under non-standard operating conditions.

7.6 Performance Testing

Performance testing was conducted to evaluate the efficiency and responsiveness of the database under various operational loads. This included the execution of complex queries involving multi-table joins, retrieval of large datasets such as extensive order histories, and simulation of concurrent user interactions. Response times were monitored to ensure that the system meets acceptable performance thresholds and that indexing and schema design adequately support scalable usage. These tests provide assurance that the system can handle realistic workloads without significant degradation in performance.

Chapter 8 Conclusion

This project has successfully accomplished its primary objective: the design and implementation of a normalised, fully functional relational database system for e-commerce order management. Customers, Products, Orders, and Order Details are the primary operational entities. The schema, which has been thoroughly normalised to the Third Normal Form (3NF), offers a structurally robust basis that minimises redundancy and guarantees data integrity. The accompanying Entity-Relationship Diagram (ERD) accurately models the business logic, while the developed Graphical User Interface (GUI) and comprehensive SQL query set demonstrate the system's capacity to execute essential operational workflows. Validation through systematic testing confirms the correctness of relational constraints and the reliability of basic CRUD (Create, Read, Update, Delete) operations. As an academic prototype, the system fulfils its pedagogical purpose, illustrating key principles of database theory and application.

8.1 Limitations of Existing Research

Notwithstanding the successful demonstration of core functionalities, the present work is intentionally bounded by the scope of a term project, resulting in several significant limitations that preclude its direct deployment in a production environment. The most prominent constraints pertain to three domains: interface sophistication, database programmability, and system security. The implemented GUI, while functional, remains rudimentary; it lacks advanced user experience features such as dynamic filtering, input validation, and a cohesive design framework, which limits its practical usability. More critically, the database schema operates primarily as a passive data store. It does not encapsulate business logic within the database layer through mechanisms such as triggers for automated integrity enforcement or stored procedures for complex transaction processing, thereby placing the entire operational burden on the application layer. The most substantial limitation, however, is the complete absence of a security framework. The system implements no authentication or authorisation protocols, creating a fundamental vulnerability that renders it unsuitable for any context involving sensitive or proprietary data. These limitations collectively define the gap between a proven conceptual model and a robust, deployable application system.

Furthermore, the state of knowledge required exceeds the scope of a term paper. Given these constraints, the authors are unable to fully grasp the current situation, and their limited understanding prevents further meaningful analysis. Accordingly, the remaining discussion is left to the reader.

8.2 Directions for Future Research

The identified limitations provide a clear and structured agenda for subsequent research and development. Future work should prioritise the transformation of this prototype into a secure and scalable application. The immediate research direction must address the critical security vacuum through the design and integration of a robust authentication and role-based access control (RBAC) system. Concurrently, the database architecture should be enhanced by migrating key business rules into the database layer. This involves developing stored procedures for routine and complex operations and implementing triggers to maintain derived data consistency (e.g., real-time inventory updates), thereby improving performance and data integrity.

Further research should also explore scalability under large-volume transaction loads, investigating indexing strategies, query optimisation, and potential database partitioning techniques. Extending the system's business logic to encompass a broader e-commerce feature set—such as integrated payment processing, shipment tracking, and customer relationship management (CRM) modules—represents another significant avenue. Finally, a complete redesign of the front end into a responsive, modern web application would be essential for user adoption. In summary, this project establishes a validated foundational model; the proposed directions chart a path for evolving it from an academic exercise into a comprehensive, secure, and industrially relevant e-commerce management solution.