Technical Report: Online Sales Database Project

1. Introduction

This technical report details the design, creation, and usage of an SQL database for an Online Sales System. The project was undertaken to fulfill requirements including database design, normalization.

2. Analysis of Client's Needs, Users, and Entities.

2.1. Client's Needs

The primary needs for an online sales platform include:

- **Product Management:** The ability to add, update, categorize, and remove products. This includes managing product details like name, description, price, images, and stock levels.
- **Customer Management:** Secure registration and login for customers, profile management (including addresses), and order history tracking.
- **Order Processing:** A system for customers to place orders, and for administrators to manage these orders through various stages (e.g., pending, processing, shipped, delivered, cancelled).
- **Inventory Control:** Tracking product stock levels to prevent overselling and to manage reordering.
- **Shopping Cart Functionality:** Allowing customers to add multiple items to a cart before proceeding to checkout.
- **Secure Transactions:** Ensuring that payment information and customer data are handled securely (though the database itself stores payment status/method, not sensitive payment details like full credit card numbers).
- **Reporting and Analytics:** The ability for administrators to generate reports on sales, popular products, customer activity, etc. (supported by the query capabilities).
- **User Roles and Permissions:** Differentiating access levels for various types of users (e.g., customers, administrators with different responsibilities).

2.2. Identified Users

Two primary categories of users were identified for this system:

1. **Customers:** These are the end-users who interact with the front-end of the online store. Their activities include:
    - Browsing and searching for products.
    - Viewing product details.
    - Registering for an account and managing their profile.
    - Adding products to a shopping cart.
    - Placing orders and making payments (interfacing with a payment gateway, which is outside the DB scope but influences DB design for order status).
    - Viewing their order history.
    - Managing shipping and billing addresses.
2. **Administrators/Staff:** These users manage the backend operations of the online store. Their roles and responsibilities can vary, leading to sub-categories of administrators (e.g., SuperAdmin, Product Manager, Order Manager). Their activities include:
    - Adding new products and updating existing product information (descriptions, prices, images).
    - Managing product categories.
    - Monitoring and updating inventory levels.
    - Processing customer orders (confirming payment, initiating shipping).
    - Managing customer accounts (e.g., handling queries, password resets).
    - Viewing sales reports and other platform analytics.
    - Managing website content, promotions, and potentially user roles.

2.3. Identified Entities and Key Attributes

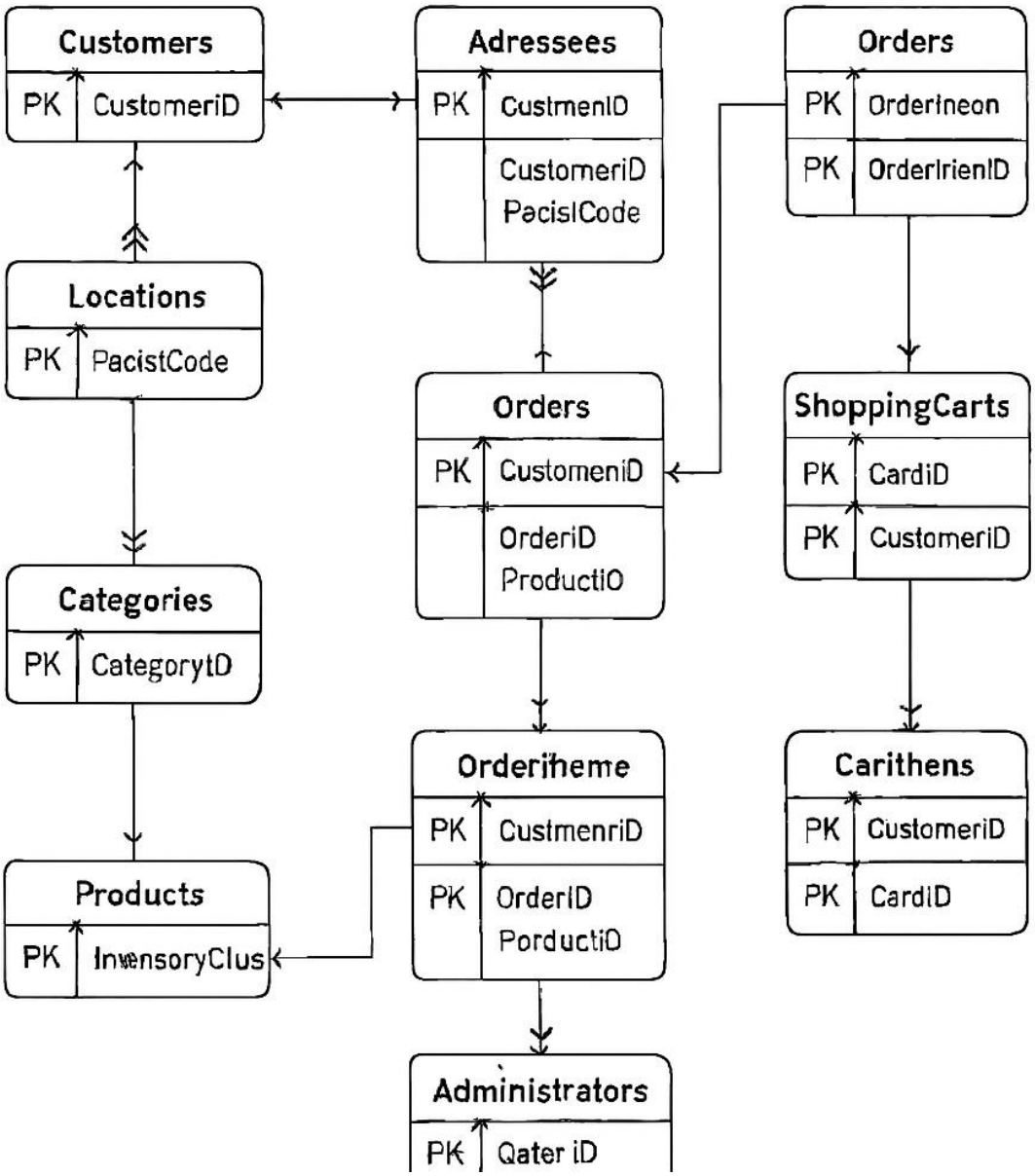
Based on the needs analysis and user identification, the following core entities were established for the online sales database. Each entity is listed with its primary purpose and key attributes that were considered during the initial design phase. These were further refined during ER modeling and normalization.

- **Customers:** Stores information about registered users.

    - Attributes: **CustomerID** (PK), **FirstName**, **LastName**, **Email**, **PasswordHash**, **PhoneNumber**, **RegistrationDate**, **LastLoginDate**.
- **Addresses:** Manages multiple shipping and billing addresses for customers.
    - Attributes: **AddressID** (PK), **CustomerID** (FK), **AddressType**, **StreetAddress**, **City**, **State**, **PostalCode**, **Country**.
- **Products:** Contains details about the items sold on the platform.
    - Attributes: **ProductID** (PK), **ProductName**, **Description**, **CategoryID** (FK), **UnitPrice**, **ImageURL**, **DateAdded**, **LastUpdated**.
- **Categories:** Organizes products into logical groups.
    - Attributes: **CategoryID** (PK), **CategoryName**, **Description**.
- **Inventory/Stock (InventoryStock):** Tracks the quantity of each product available.
    - Attributes: **StockID** (PK), **ProductID** (FK), **QuantityInStock**, **ReorderLevel**, **LastStockUpdate**.
- **Orders:** Records customer purchases.
    - Attributes: **OrderID** (PK), **CustomerID** (FK), **OrderDate**, **OrderStatus**, **ShippingAddressID** (FK), **BillingAddressID** (FK), **TotalAmount**, **PaymentMethod**, **PaymentStatus**.
- **OrderItems:** A junction table detailing which products and in what quantities are included in each order.
    - Attributes: **OrderItemID** (PK), **OrderID** (FK), **ProductID** (FK), **Quantity**, **UnitPriceAtPurchase**, **Subtotal**.
- **ShoppingCarts:** Temporarily stores items a customer intends to buy.
    - Attributes: **CartID** (PK), **CustomerID** (FK), **DateCreated**, **LastUpdated**.
- **CartItems:** A junction table for items in a shopping cart.
    - Attributes: **CartItemID** (PK), **CartID** (FK), **ProductID** (FK), **Quantity**, **DateAdded**.
- **Administrators/Staff (Administrators):** Manages backend users and their roles.
    - Attributes: **AdminID** (PK), **Username**, **PasswordHash**, **Email**, **Role**, **LastLoginDate**.

- **Locations (Identified during Normalization):** Stores unique combinations of postal codes, cities, states, and countries to reduce redundancy in addresses.
    - Attributes: **PostalCode** (PK), **City**, **State**, **Country**.

3. ER Database Design and Mapping Process



3.1. ER Diagram Description

The ER diagram includes the entities identified in the analysis phase, along with their primary attributes and the relationships connecting them. Cardinality constraints (One-to-One, One-to-Many, Many-to-Many) are defined for each relationship to accurately model the business rules of an online sales system.

The key relationships depicted are:

- **Customers and Addresses:** A one-to-many relationship. One customer can have multiple addresses (shipping, billing), but each address record belongs to a single customer.
- **Customers and Orders:** A one-to-many relationship. One customer can place multiple orders, while each order is associated with exactly one customer.
- **Customers and ShoppingCarts:** A one-to-one relationship. Each customer has one active shopping cart.
- **Orders and Products:** A many-to-many relationship, resolved through the **OrderItems** junction table. An order can contain multiple products, and a product can be part of multiple orders.
- **Products and Categories:** A many-to-one relationship. Many products can belong to one category, but each product is assigned to a single category.
- **Products and InventoryStock:** A one-to-one relationship. Each product has a corresponding record in the inventory stock.
- **ShoppingCarts and Products:** A many-to-many relationship, resolved through the **CartItems** junction table. A shopping cart can contain multiple products, and a product can be in multiple shopping carts (across different customers).
- **Addresses and Locations:** A many-to-one relationship (after normalization). Many address records can share the same postal code, which points to a unique entry in the **Locations** table.
- **Orders and Addresses:** Each order references two addresses (one for shipping, one for billing) from the **Addresses** table.

3.2. Relational Schema Mapping

The detailed relational schema, including table names, column names, data types, primary keys (PK), foreign keys (FK), and other constraints (e.g., NOT NULL, UNIQUE), is as follows:

1. **Customers Table:**

- **CustomerID** INT, PK, AUTO_INCREMENT
- **FirstName** VARCHAR(100), NOT NULL
- **LastName** VARCHAR(100), NOT NULL
- **Email** VARCHAR(255), NOT NULL, UNIQUE
- **PasswordHash** VARCHAR(255), NOT NULL
- **PhoneNumber** VARCHAR(20)
- **RegistrationDate** TIMESTAMP, DEFAULT CURRENT_TIMESTAMP
- **LastLoginDate** TIMESTAMP

2. **Locations Table (Post-Normalization):**
    - **PostalCode** VARCHAR(20), PK
    - **City** VARCHAR(100), NOT NULL
    - **State** VARCHAR(100)
    - **Country** VARCHAR(100), NOT NULL
3. **Addresses Table (Post-Normalization):**
    - **AddressID** INT, PK, AUTO_INCREMENT
    - **CustomerID** INT, NOT NULL, FK REFERENCES **Customers**(**CustomerID**)
    - **AddressType** VARCHAR(50), NOT NULL
    - **StreetAddress** VARCHAR(255), NOT NULL
    - **PostalCode** VARCHAR(20), NOT NULL, FK REFERENCES **Locations**(**PostalCode**)
4. **Categories Table:**
    - **CategoryID** INT, PK, AUTO_INCREMENT
    - **CategoryName** VARCHAR(100), NOT NULL, UNIQUE
    - **Description** TEXT
5. **Products Table:**
    - **ProductID** INT, PK, AUTO_INCREMENT

- **ProductName** VARCHAR(255), NOT NULL
- **Description** TEXT
- **CategoryID** INT, FK REFERENCES **Categories**(**CategoryID**)
- **UnitPrice** DECIMAL(10, 2), NOT NULL
- **ImageURL** VARCHAR(2048)
- **DateAdded** TIMESTAMP, DEFAULT CURRENT_TIMESTAMP
- **LastUpdated** TIMESTAMP

6. **InventoryStock Table:**
    - **StockID** INT, PK, AUTO_INCREMENT
    - **ProductID** INT, NOT NULL, UNIQUE, FK REFERENCES **Products**(**ProductID**)
    - **QuantityInStock** INT, NOT NULL, DEFAULT 0
    - **ReorderLevel** INT, DEFAULT 10
    - **LastStockUpdate** TIMESTAMP
7. **Orders Table:**
    - **OrderID** INT, PK, AUTO_INCREMENT
    - **CustomerID** INT, NOT NULL, FK REFERENCES **Customers**(**CustomerID**)
    - **OrderDate** TIMESTAMP, DEFAULT CURRENT_TIMESTAMP
    - **OrderStatus** VARCHAR(50), NOT NULL
    - **ShippingAddressID** INT, NOT NULL, FK REFERENCES **Addresses**(**AddressID**)
    - **BillingAddressID** INT, NOT NULL, FK REFERENCES **Addresses**(**AddressID**)
    - **TotalAmount** DECIMAL(12, 2), NOT NULL (Denormalized for performance)
    - **PaymentMethod** VARCHAR(50)
    - **PaymentStatus** VARCHAR(50), NOT NULL
8. **OrderItems Table:**
    - **OrderItemID** INT, PK, AUTO_INCREMENT

- **OrderID** INT, NOT NULL, FK REFERENCES **Orders**(**OrderID**)
- **ProductID** INT, NOT NULL, FK REFERENCES **Products**(**ProductID**)
- **Quantity** INT, NOT NULL
- **UnitPriceAtPurchase** DECIMAL(10, 2), NOT NULL
- **Subtotal** DECIMAL(12, 2), NOT NULL (Denormalized for performance)
- UNIQUE (**OrderID**, **ProductID**)

9. **ShoppingCarts Table:**
    - **CartID** INT, PK, AUTO_INCREMENT
    - **CustomerID** INT, NOT NULL, UNIQUE, FK REFERENCES **Customers**(**CustomerID**)
    - **DateCreated** TIMESTAMP, DEFAULT CURRENT_TIMESTAMP
    - **LastUpdated** TIMESTAMP
10. **CartItems Table:**
    - **CartItemID** INT, PK, AUTO_INCREMENT
    - **CartID** INT, NOT NULL, FK REFERENCES **ShoppingCarts**(**CartID**)
    - **ProductID** INT, NOT NULL, FK REFERENCES **Products**(**ProductID**)
    - **Quantity** INT, NOT NULL
    - **DateAdded** TIMESTAMP, DEFAULT CURRENT_TIMESTAMP
    - UNIQUE (**CartID**, **ProductID**)
11. **Administrators Table:**
    - **AdminID** INT, PK, AUTO_INCREMENT
    - **Username** VARCHAR(100), NOT NULL, UNIQUE
    - **PasswordHash** VARCHAR(255), NOT NULL
    - **Email** VARCHAR(255), NOT NULL, UNIQUE
    - **Role** VARCHAR(50), NOT NULL
    - **LastLoginDate** TIMESTAMP

4. Normalization Process

4.1. First Normal Form (1NF)

- **Rule:** All attributes must be atomic, and each table must have a primary key. No repeating groups.
- **Application:** The initial schema was designed with atomic attributes (e.g., **FirstName** is a single unit, not further divided for this scope) and primary keys for all tables. Repeating groups were avoided by design (e.g., multiple addresses for a customer are handled in a separate **Addresses** table linked by **CustomerID**).

4.2. Second Normal Form (2NF)

- **Rule:** Must be in 1NF, and all non-key attributes must be fully functionally dependent on the entire primary key. This is mainly relevant for tables with composite primary keys.
- **Application:**
    - Tables with single-attribute primary keys automatically satisfy 2NF if they are in 1NF.
    - For tables with composite keys like **OrderItems** (PK: **OrderID**, **ProductID**) and **CartItems** (PK: **CartID**, **ProductID**), all non-key attributes (**Quantity**, **UnitPriceAtPurchase**, **Subtotal** in **OrderItems**; **Quantity**, **DateAdded** in **CartItems**) were found to be dependent on the *entire* composite key, not just a part of it.

4.3. Third Normal Form (3NF)

- **Rule:** Must be in 2NF, and there should be no transitive dependencies (where a non-key attribute depends on another non-key attribute).
- **Application:**
    - A transitive dependency was identified in the initial **Addresses** table: **AddressID** -> **PostalCode** -> (**City**, **State**, **Country**). To resolve this, the **Addresses** table was decomposed:
        - A new **Locations** table was created: **Locations** (**PostalCode** PK, **City**, **State**, **Country**).

    - The **Addresses** table was modified: **Addresses** (**AddressID** PK, **CustomerID** FK, **AddressType**, **StreetAddress**, **PostalCode** FK referencing **Locations**).

- Derived attributes like **TotalAmount** in **Orders** (sum of **OrderItems.Subtotal**) and **Subtotal** in **OrderItems** (**Quantity * UnitPriceAtPurchase**) were noted. While strictly, these could be calculated on demand, they were retained for performance and convenience, acknowledging this as a deliberate denormalization. This decision is common in practical database design for OLTP systems.