Name: Tram Ngo

Online Retail Database

I. Project Statement

1. About the project: This project aims to design and implement a relational database system for an Online Retail Application. This database aims to manage the efficiency of keeping track of the online retails information: Products, Customers, Orders, Payments.
2. Goal: to enable accurate order processing, maintain product availability, and provide secure and reliable transaction handling while ensuring a positive customer experience.

II. Key Feature

- Products: Add, update, delete, and search for products by name, category, price, or availability.
- Customers: Register new customers, manage customer profiles, and track purchase history.
- Orders: Record customer orders, manage shopping carts, track order status (pending, shipped, delivered), and handle returns.
- Payments: Support multiple payment methods, record transactions, and ensure secure processing.
- Shippings: Manage shipping details such as carrier, tracking number, shipping status, and delivery updates.
- Order Items: Record the specific products and quantities included in each order for accurate order fulfillment.

III. Expected Outcome

- A well-structured and normalized database schema to support online retail operations.
- Database query to search the product detail, order with the customer info, manage shipping informations
- Improved efficiency in managing products, customer data, and order transactions.
- Enhanced data integrity, consistency, and security within the system.

IV. Brainstorming about the tables (Entities) and their relationships.

- **Step 1: Creating Entities and Attributes**

1. Products: product_id, product_name, category, price, availability (price >=0, stock quantity >=0)
2. Customers: customer_id, customer_name, email, phone_number, address (email must be unique)
3. Orders: order_id, customer_id (FK), order_date, status
4. Payments: payment_id, order_id(FK), payment_date, method, amount (amount >=0)
5. Shippings: shipping_id, order_id (FK), carrier, tracking_number, status, delivery_date (tracking number and shipping_id must be unique)
6. Order_product: (Join table between Orders and Products): order_id, product_id, quantity, total_price

- **Step 2: Relationship between tables**

1. Products → Order_product: One-to-Many
2. Orders → order_product : One-to-many
3. Customer → Orders : One-To-Many
4. Order → Payments: One-to-One or One-to-Many
5. Orders → Shippings : One-to-Many

V. Creating Tables

1. Products

product_id	INT (PK)
product_name	CHAR (20)
category	VARCHAR(20)
price	DECIMAL (10,2), >=0
availability	INT, >= 0

2. Customers

customer_id	INT (PK)
customer_name	VARCHAR(20)
email	CHAR(20), unique
phone	VARCHAR(20)
address	CHAR(20)

3. **Orders**

order_id	INT (PK)
customer_id	INT (PK)
order_date	DATE
status	VARCHAR(10)

4. **Payments**

payment_id	INT (PK)
order_id	INT (FK)
payment_date	DATE
method	VARCHAR(20)
amount	DECIMAL(10,2)

5. **Shippings**

shipping_id	INT (PK)
order_id	INT (FK)
carrier	VARCHAR(50)
tracking_number	VARCHAR(50)
status	VARCHAR(10)
delivery_date	DATE

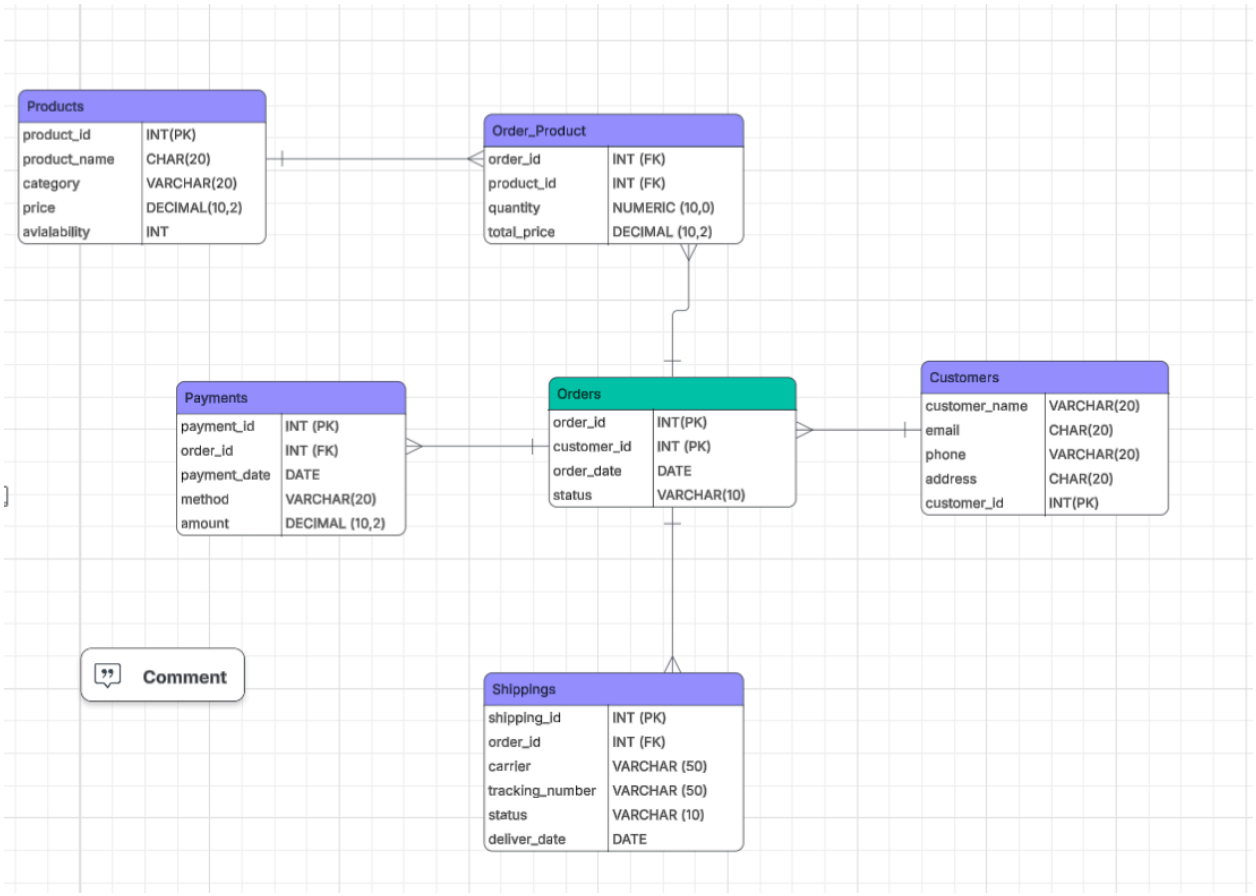
6. **Order_product**

order_id	**INT(FK)**
product_id	**INT(FK)**
quantity	NUMERIC(10,0)
total_price	DECIMAL(10,2)

In this table, the combination of order_id and product_id will be the primary key of this table (composite key)

VI. TABLE SCHEMA & E-R DIAGRAM

This schema will show the tables and their attribute along with the relationship between each table.



VI. TABLE NORMALIZATION

1. **1NF TABLE**

Each column must hold *atomic* (single) values — no repeating groups, lists, or sets → all tables meet the 1NF rule because every attribute has a single value.

2. 2NF TABLE

All the tables met the 2NF rule because every attribute in the table depend on their table's primary keys and are also part of 1NF TABLE

3. 3NF TABLE

All tables met the 3NF rule because every table is in a 2NF table and each of the non_key columns in each table all depend on their key column for every table.

VII. QUERY, REPORTS, AND ROLES

QUERY

- Total Order Cost: calculate the total cost for each order by using formula of (quanity x price)
- Check Stocking: find the low stock products to restock in time.
- Customer History Order: view what order the customer has made
- Payment info: check the amount and payment method of customers.
- Product search: find the desired products.

REPORTS

1. **Invoice Report**
    a. **Content:** Customer name, order details, product names, quantities, unit prices, total cost.
    b. **Purpose:** Keep track for the revenue.
2. **Stock Report**
    a. **Content:** List of products with current `availability`, `category`, and `price`.
    b. **Purpose:** Help the business to manage the stock and their quantity of products.

3. **Monthly Sales Report**

    a. **Content:** Aggregated sales by month, including total revenue and top-selling products.

    b. **Purpose:** Help for visualization the trend or pattern in the sale by analyzing the number or type of product has been sold for every season or month

4. **Customer Statement**

    a. **Content:** Summary of all transactions for a customer, including orders, payments, and outstanding balances.

    b. **Purpose:** Help to track the order history and payment.

5. **Shipping Report**

    a. **Content:** Orders with their shipping status, carrier, tracking number, and delivery dates.

    b. **Purpose:** Helps to keep track of the shipping details for both business and customer.

ROLES

This part indicates who can use the database and what they can do.

- Admin: can access to everything
- Manager: can view the report, run the query but cannot change the database structure
- Employee: can update and change the new record but cannot see the private data or information (customer address, payment, ….)
- Customer: can only see their payment, shipping, order and cannot access everything else.

REFLECTIONS ON PROJECT

Through this project, I learned:

- The importance of proper normalization to main data integrity
- Analyzing the relationships between orders, payments, and products showed how composite keys improve accuracy.
- Arrange roles to improve the security using databases.