

COMP 7115 Data Base Management Systems

E-Commerce Platform

Professor: Purna Kumar Thota

Submitted by Team Group-8

- 1.Aravind Reddy Kasireddy**
- 2.Sai Manish Kurukunda**
- 3.Sandeep Lakineni**
- 4.Aparna Makkena**
- 5.Gopi Krishna Raparla**

Contents

Project Overview.....	3
Challenges faced:.....	4
Design and Implementation:.....	6
Database Design:.....	6
ER DIAGRAM:.....	6
Normalization.....	7
Normal Form (1NF) and 2nd Normal Form (2NF):.....	7
3rd Normal Form (3NF):.....	7
Schema Creation.....	8
Backend Functionalities.....	17
Performance Optimization.....	18
1. Query Optimization.....	18
2. Indexing.....	18
3. Caching.....	18
Security and Privacy.....	18
1. Data Encryption.....	18
2. Authentication and Authorization.....	18
3. Input Validation and Sanitization.....	19
Analytics and Reporting.....	19
Conclusion.....	20
References.....	21

Project Overview

- This project aims to design and implementation of a MySQL relational database for an e-commerce platform called "MyShoppe." The platform aims to provide a seamless online shopping experience for customers and a reliable platform for vendors to list and sell their products. The database management system plays a crucial role in ensuring efficient data storage, retrieval, and management of various entities involved in the e-commerce ecosystem.
- This project aims to create a full-featured e-commerce platform, similar to Amazon. The app will facilitate seamless trading and information exchange for all stakeholders involved with the products (Li, Zhao, Xu, & Pu, 2020).
- To achieve this ambitious goal, we understand the critical role of a well-maintained database system. We will meticulously define both functional and non-functional requirements to ensure the database meets all necessary criteria.
- Our e-commerce solution will be developed by thoroughly researching academic literature, industry standards, and best practices for secure, efficient, and user-friendly e-commerce platforms.

Planning and Database Design

- We began by discussing the project overview and core concepts like database building and e-commerce platform development.
- We researched e-commerce applications and databases through research papers to gather information.
- Focusing on the database and platform, we identified key entities and their relationships. We also discussed appropriate primary and foreign keys for each entity.
- Once the entities were defined, we created and tested the database tables in MySQL to ensure they functioned correctly.

Database Population and Testing

- We built queries to insert data into all the tables and searched for suitable datasets online. We populated the tables with some of this data.
- We constructed the complete database with table creation and data insertion queries, ensuring no errors were present.
- We then ran various queries to verify the database responded correctly, completing the database building and testing phase.

E-commerce Platform Development

- To learn about building e-commerce applications, we explored online resources including YouTube videos and IEEE papers.
- We identified the application's core functionalities like login, signup, and a homepage, and began building and testing them.
- We used an existing application template and customized it to meet our requirements by adding the necessary features.
- Finally, we integrated various products into the application and established a payment gateway.

Challenges faced:

Below are the challenges that we faced during the building of the platform and database:

Database Design Challenges:

- **Defining Table Relationships:** Initially, we encountered difficulty determining the relationships (cardinalities) between tables. We consulted various resources to understand cardinalities and successfully mapped out the connections between tables.
- **Data Normalization:** Minimizing data redundancy while maintaining data integrity (normalization) proved to be a hurdle. We addressed this by optimizing the database structure to reduce duplication and ensure data consistency.
- **Query Performance with Large Datasets:** We anticipated potential slowdowns with a large data volume. To address this, we focused on data optimization to achieve faster query response times.
- **Syntax Errors During Testing:** During database table testing, we encountered syntax errors. We promptly identified and corrected these errors to ensure the database functioned smoothly.

E-commerce Application Development Challenges:

- **Building Core functionalities:** Developing core features like login, signup, and the homepage presented initial challenges. By reviewing various resources, we were able to overcome these hurdles and successfully build these functionalities.
- **Product Integration and Pricing:** Adding products and assigning prices initially posed difficulties. However, through continued exploration of reference materials, we were able to successfully integrate products and establish a pricing structure.

Business Goals

The primary business goals of this ecommerce platform are:

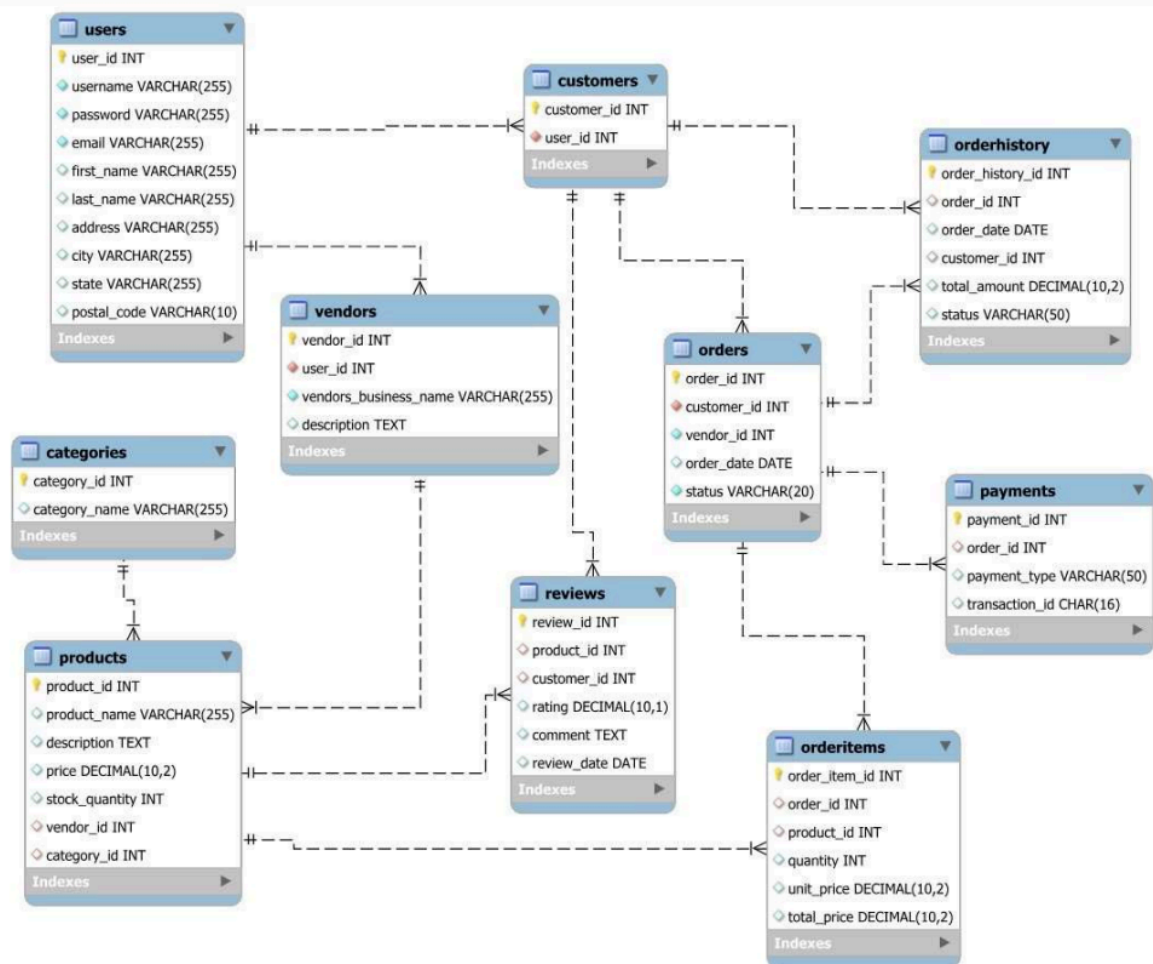
- Provide a user-friendly platform for customers to browse and purchase products from a wide range of vendors.
- Enable vendors to easily list, manage, and monitor their product inventory and sales.
- Ensure secure user authentication and authorization for both customers and vendors.
- Implement a robust order management system to track and process customer orders efficiently.
- Maintain a comprehensive product catalogue with detailed information and categorization.
- Analyze customer behaviour, sales trends, and vendor performance to drive data-driven business decisions.

Design and Implementation:

Database Design:

ER DIAGRAM:

The database design for MyShoppe is represented by the following Entity-Relationship (ER) diagram, which depicts the relationships between various entities and their attributes. The diagram employs crow's foot notation to indicate minimum and maximum cardinalities.



Normalization

Normal Form (1NF) and 2nd Normal Form (2NF):

To achieve 2NF, we must first ensure that the database is in 1st Normal Form (1NF), as stated above, which means eliminating repeating groups and ensuring that each column contains atomic (indivisible) values. After that, we need to meet an additional requirement for 2NF:

A table is in 2NF if it is in 1NF and if all non-key attributes are fully functionally dependent on the entire primary key.

In our e-commerce database, let's take the "Orders" and "OrderItems" tables as an example:

- The "Orders" table likely has a primary key (e.g., OrderID), which uniquely identifies each order.

- The "OrderItems" table may have an OrderItemID as the primary key, and it likely has a reference to the OrderID from the "Orders" table as a foreign key.

To ensure that these tables are in 2NF, all non-key attributes in the "OrderItems" table should be fully functionally dependent on the entire primary key, which consists of OrderItemID and OrderID. This means that each column in the "OrderItems" table should be relevant to both the OrderItemID and OrderID. This design is suitable for tracking items within orders.

3rd Normal Form (3NF):

To achieve 3NF, we must first ensure that our database is in 2NF. After achieving 2NF, we need to meet an additional requirement for 3NF:

A table is in 3NF if it is in 2NF and if all attributes are functionally dependent only on the primary key. In the context of our e-commerce database, we can consider the "Users" table:

- The "Users" table may have a primary key, such as UserID, and other attributes like Username, Email, and Password.

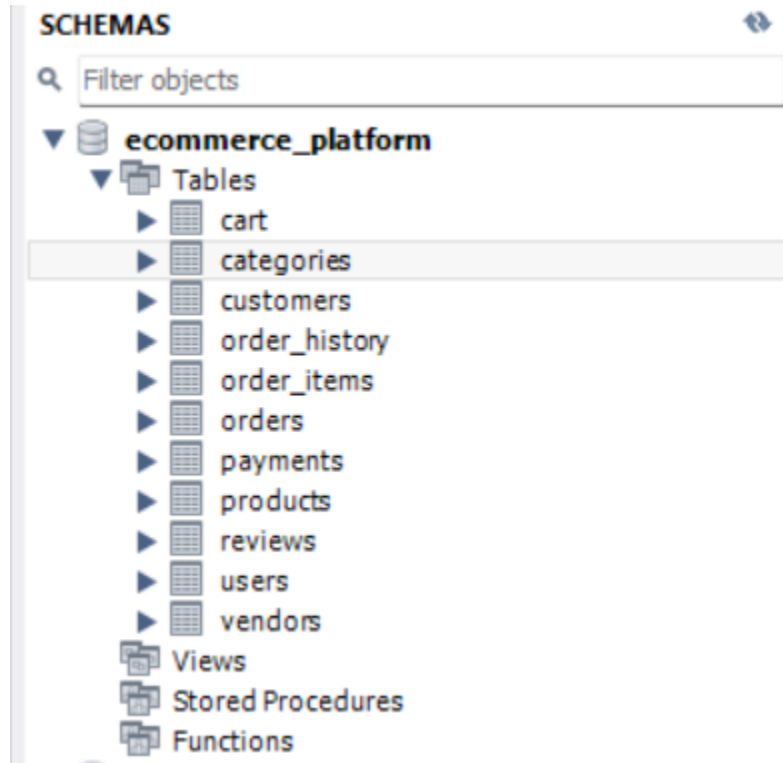
To make the "Users" table 3NF, we should ensure that all non-key attributes (in this case, Username, Email, and Password) are directly related to the primary key (UserID) and not dependent on each other. For example, Username and Email should not be functionally dependent on one another.

In practice, the "Users" table is likely already in 3NF since these attributes are usually unrelated to each other. The same normalization principles apply to other tables.

The database tables have been designed following the principles of normalization to reduce data redundancy and improve data integrity. The tables are normalized up to the Third Normal Form (3NF) to eliminate partial and transitive dependencies.

Schema Creation

CREATE DATABASE ecommerce_platform



Tables Creation and Sample data insertion

-- Create the categories table

```
CREATE TABLE categories (  
  category_id INT AUTO_INCREMENT PRIMARY KEY,  
  category_name VARCHAR(100) NOT NULL,  
  category_description TEXT,  
  parent_category_id INT DEFAULT NULL,  
  FOREIGN KEY (parent_category_id) REFERENCES categories(category_id)  
);
```

-- Insert sample data into categories table

```
INSERT INTO categories (category_name, category_description, parent_category_id) VALUES  
(  
'Electronics', 'Electronic devices and accessories', NULL),  
(  
'Computers', 'Desktop computers, laptops, and accessories', 1),  
(  
'Smartphones', 'Smartphones and mobile accessories', 1),  
(  
'Clothing', 'Apparel and fashion items', NULL),
```



```
(
    ('Men's Clothing', 'Clothing for men', 4),
    ('Women's Clothing', 'Clothing for women', 4),
    ('Home & Kitchen', 'Furniture, appliances, and home decor', NULL),
    ('Beauty', 'Makeup, skincare, and personal care products', NULL),
    ('Sports & Outdoors', 'Sports equipment and outdoor gear', NULL),
    ('Books', 'Books and digital publications', NULL);

```

Query 1 categories x

Limit to 1000 rows

```
1 • SELECT * FROM ecommerce_platform.categories;
```

category_id	category_name	category_description	parent_category_id
1	Electronics	Electronic devices and accessories	NULL
2	Computers	Desktop computers, laptops, and accessories	1
3	Smartphones	Smartphones and mobile accessories	1
4	Clothing	Apparel and fashion items	NULL
5	Men's Clothing	Clothing for men	4
6	Women's Clothing	Clothing for women	4
7	Home & Kitchen	Furniture, appliances, and home decor	NULL
8	Beauty	Makeup, skincare, and personal care products	NULL
9	Sports & Outdoors	Sports equipment and outdoor gear	NULL
10	Books	Books and digital publications	NULL
NULL	NULL	NULL	NULL

```
-- Create the customers table
```

```
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone_number VARCHAR(20),
    billing_address TEXT,
    shipping_address TEXT
);
```

```
-- Insert sample data into customers table
```

```
INSERT INTO customers (first_name, last_name, email, phone_number, billing_address,
shipping_address) VALUES
('John', 'Doe', 'john.doe@example.com', '+1234567890', '123 Main St, City, Country', '123 Main
St, City, Country'),
('Jane', 'Smith', 'jane.smith@example.com', '+9876543210', '456 Oak Ave, City, Country', '456
Oak Ave, City, Country'),
```

```
(
    'Michael', 'Johnson', 'michael.johnson@example.com', '+5678901234', '789 Maple Rd, City, Country', '789 Maple Rd, City, Country'),
    ('Joanas', 'Doe', 'joanas.doe@example.com', '+1234567890', '123 Main St, City, Country', '123 Main St, City, Country'),
    ('Janet', 'Smith', 'janet.smith@example.com', '+9876543210', '456 Oak Ave, City, Country', '456 Oak Ave, City, Country'),
    ('Mickey', 'Johnson', 'mickey.johnson@example.com', '+5678901234', '789 Maple Rd, City, Country', '789 Maple Rd, City, Country');

```

```
1 • SELECT * FROM ecommerce_platform.customers;
```

Result Grid							
Filter Rows:							
Edit: Export/Import: Wrap Cell Content: I A							
	customer_id	first_name	last_name	email	phone_number	billing_address	shipping_address
▶	6	John	Doe	john.doe@example.com	+1234567890	123 Main St, City, Country	123 Main St, City, Country
	7	Jane	Smith	jane.smith@example.com	+9876543210	456 Oak Ave, City, Country	456 Oak Ave, City, Country
	8	Michael	Johnson	michael.johnson@example.com	+5678901234	789 Maple Rd, City, Country	789 Maple Rd, City, Country
	9	Joanas	Doe	joanas.doe@example.com	+1234567890	123 Main St, City, Country	123 Main St, City, Country
	10	Janet	Smith	janet.smith@example.com	+9876543210	456 Oak Ave, City, Country	456 Oak Ave, City, Country
	11	Mickey	Johnson	mickey.johnson@example.com	+5678901234	789 Maple Rd, City, Country	789 Maple Rd, City, Country
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- Create the vendors table
```

```
CREATE TABLE vendors (
    vendor_id INT AUTO_INCREMENT PRIMARY KEY,
    company_name VARCHAR(100) NOT NULL,
    contact_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone_number VARCHAR(20),
    website VARCHAR(100),
    address TEXT
);
```

```
-- Insert sample data into vendors table
```

```
INSERT INTO vendors (company_name, contact_name, email, phone_number, website, address) VALUES
('ABC Electronics', 'John Smith', 'sales@abcelectronics.com', '+1234567890', 'www.abcelectronics.com', '123 Main St, City, Country'),
('Fashion Trends', 'Jane Doe', 'contact@fashiontrends.com', '+9876543210', 'www.fashiontrends.com', '456 Oak Ave, City, Country');
```

1 • `SELECT * FROM ecommerce_platform.vendors;`

```
-- Create the users table
CREATE TABLE users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  password VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  user_type ENUM('customer', 'vendor') NOT NULL,
  customer_id INT DEFAULT NULL,
  vendor_id INT DEFAULT NULL,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY (vendor_id) REFERENCES vendors(vendor_id)
);
```

[illegible]

-- Create the products table

```
CREATE TABLE products (  
  product_id INT AUTO_INCREMENT PRIMARY KEY,  
  product_name VARCHAR(200) NOT NULL,  
  product_description TEXT,  
  price DECIMAL(10,2) NOT NULL,  
  discounted_price DECIMAL(10,2),  
  stock_quantity INT NOT NULL,  
  vendor_id INT NOT NULL,  
  category_id INT NOT NULL,  
  product_image VARCHAR(200),  
  FOREIGN KEY (vendor_id) REFERENCES vendors(vendor_id),  
  FOREIGN KEY (category_id) REFERENCES categories(category_id)  
);
```

-- Insert sample data into products table

```
INSERT INTO products (product_name, product_description, price, discounted_price,  
stock_quantity, vendor_id, category_id, product_image) VALUES  
(  
'Apple iPhone 13 Pro', 'Apple iPhone 13 Pro, 256GB, Sierra Blue', 999.99, 949.99, 100, 1, 3,  
'iphone13pro.jpg'),  
(  
'Samsung Galaxy S22', 'Samsung Galaxy S22, 128GB, Phantom Black', 799.99, NULL, 200, 1, 3,  
'galaxys22.jpg'),  
(  
'Men"s Casual Shirt', 'Men"s Cotton Casual Shirt, Size L, Blue', 29.99, 24.99, 50, 2, 5,  
'mens_shirt.jpg');
```

Result Grid									
Filter Rows:									
Edit: Export/Import: Wrap Cell Content:									
	product_id	product_name	product_description	price	discounted_price	stock_quantity	vendor_id	category_id	product_image
▶	1	Apple iPhone 13 Pro	Apple iPhone 13 Pro, 256GB, Sierra Blue	999.99	949.99	100	1	3	iphone13pro.jpg
	2	Samsung Galaxy S22	Samsung Galaxy S22, 128GB, Phantom Black	799.99	NULL	200	1	3	galaxys22.jpg
	3	Men"s Casual Shirt	Men"s Cotton Casual Shirt, Size L, Blue	29.99	24.99	50	2	5	mens_shirt.jpg
★	NULL	NULL	NULL	NULL	NULL	50	NULL	NULL	NULL

-- Create the orders table

```
CREATE TABLE orders (  
  order_id INT AUTO_INCREMENT PRIMARY KEY,  
  customer_id INT NOT NULL,  
  order_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  total_amount DECIMAL(10,2) NOT NULL,  
  payment_status ENUM('pending', 'paid', 'failed') NOT NULL DEFAULT 'pending',  
  shipping_address TEXT NOT NULL,  
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)  
);
```

	order_id	customer_id	order_date	total_amount	payment_status	shipping_address
*	NULL	NULL	NULL	NULL	NULL	NULL

-- Create the cart table

```
CREATE TABLE cart (
  cart_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT NOT NULL,
  product_id INT NOT NULL,
  quantity INT NOT NULL,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

	cart_id	customer_id	product_id	quantity
*	NULL	NULL	NULL	NULL

-- Create the order_items table

```
CREATE TABLE order_items (
  order_item_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  product_id INT NOT NULL,
  quantity INT NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  FOREIGN KEY (order_id) REFERENCES orders(order_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Result Grid Filter Rows: <input type="text"/> Edit:					
	order_item_id	order_id	product_id	quantity	price
▶	1	1	1	1	19.09
*	NULL	NULL	NULL	NULL	NULL

-- Create the payments table

```
CREATE TABLE payments (
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  payment_method VARCHAR(50) NOT NULL,
  payment_amount DECIMAL(10,2) NOT NULL,
  payment_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (order_id) REFERENCES orders(order_id)
);
```

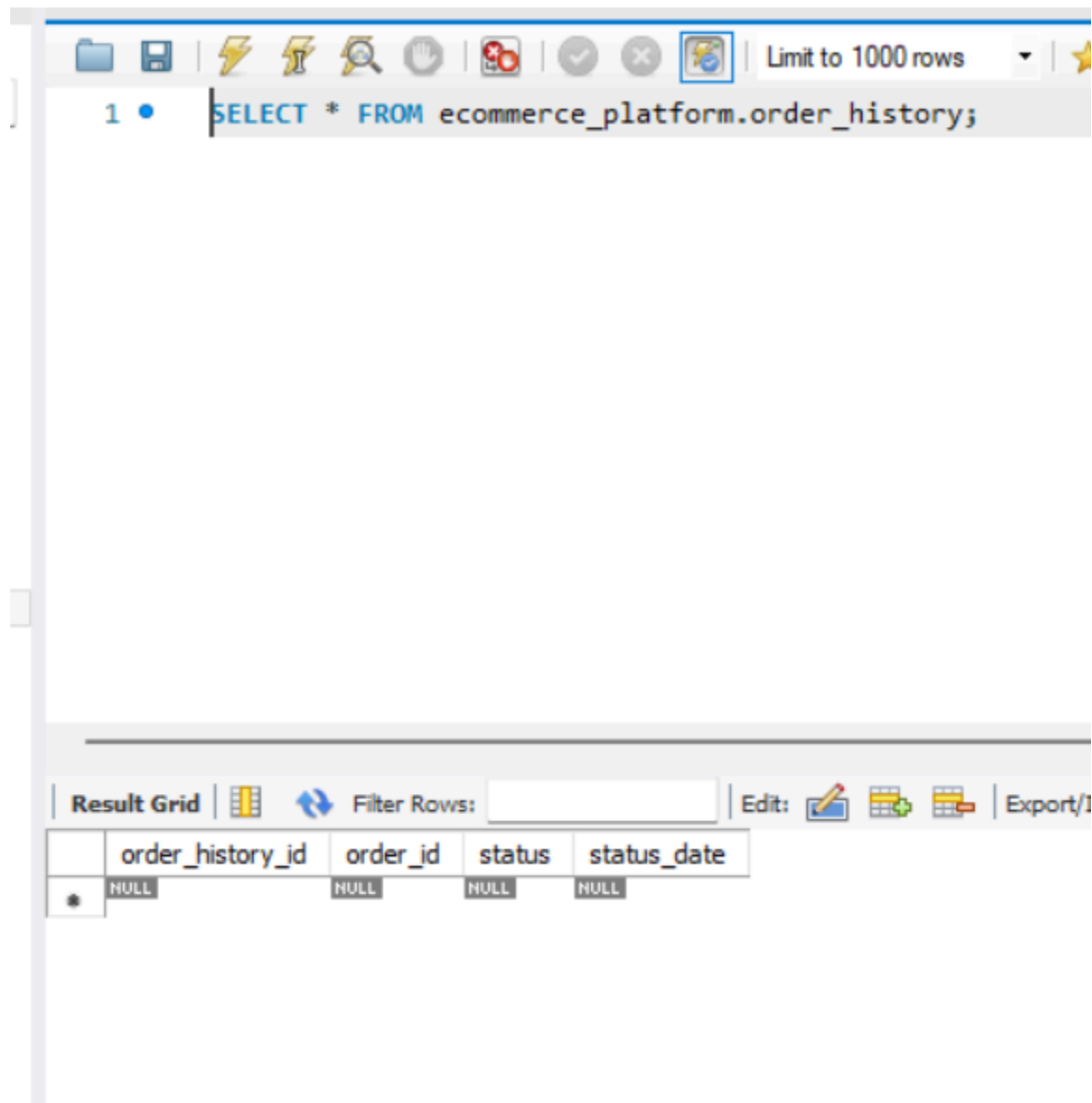
Limit to 1000 rows									
1 • SELECT * FROM ecommerce_platform.payments;									

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import: Wrap Cell Content:					
	payment_id	order_id	payment_method	payment_amount	payment_date
*	NULL	NULL	NULL	NULL	NULL

-- Create the order_history table

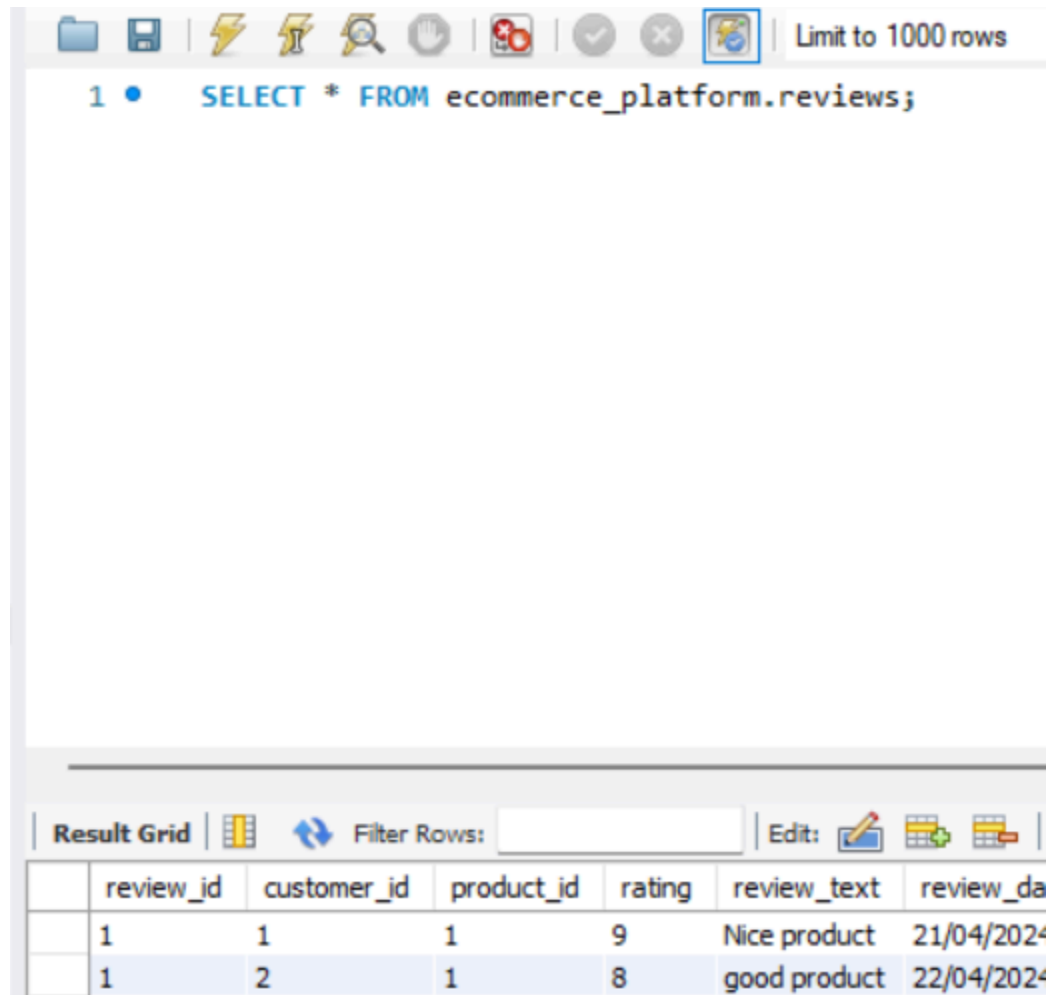
```
CREATE TABLE order_history (
```

```
order_history_id INT AUTO_INCREMENT PRIMARY KEY,  
order_id INT NOT NULL,  
status VARCHAR(50) NOT NULL,  
status_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
FOREIGN KEY (order_id) REFERENCES orders(order_id)  
);
```



```
-- Create the reviews table  
CREATE TABLE reviews (  
  review_id INT AUTO_INCREMENT PRIMARY KEY,  
  customer_id INT NOT NULL,  
  product_id INT NOT NULL,
```

rating INT NOT NULL,
review_text TEXT,
review_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
FOREIGN KEY (product_id) REFERENCES products(product_id)
);



The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons for file operations, execution, and navigation. Below the toolbar, a SQL query is entered in a text area: `1 • SELECT * FROM ecommerce_platform.reviews;`. The query is highlighted in blue. Below the query area, there is a section for the results, labeled "Result Grid". It includes a "Filter Rows:" input field and an "Edit:" button. The results are displayed in a table with the following columns: review_id, customer_id, product_id, rating, review_text, and review_da. The table contains two rows of data.

<u>review_id</u>	<u>customer_id</u>	<u>product_id</u>	rating	review_text	review_da
1	1	1	9	Nice product	21/04/2024
1	2	1	8	good product	22/04/2024

Backend Functionalities

The backend of the ShopNow e-commerce platform will be responsible for handling various functionalities, including user authentication, product management, order processing, and more. Here's an overview of the key backend functionalities:

- User Authentication and Registration
 - Implement secure user registration and login mechanisms using PHP sessions or authentication libraries like Laravel's Auth.
 - Store user information, including hashed passwords, securely in the MySQL database.
- Product Management
 - Develop a system for third-party sellers to register their companies and list their products on the platform.
 - Implement functionality for sellers to add, update, and remove products, manage categories, and upload product information and images.
- Product Listings
 - Implement search and filtering capabilities for customers to browse and search for products based on categories, keywords, and other filters.
 - Retrieve and display detailed product information, including descriptions, prices, and seller information, on individual product pages.
- Shopping Cart and Checkout
 - Develop a shopping cart system that allows customers to add products to their cart, update quantities, and view cart totals.
 - Implement a secure checkout process that integrates with a payment gateway (e.g., PayPal, Stripe) for processing customer payments.
- Order Management
 - Create order records in the database upon successful checkout, storing order details, product information, and customer information.
 - Enable customers to view their order history and order details, including order status and shipping information.
 - Provide functionality for sellers to manage order statuses, update shipping information, and track orders through their lifecycle.
- User Profiles
 - Develop user profile pages for customers and sellers, displaying order history, contact details, and basic information.
 - Allow sellers to manage their product listings and inventory from their profiles.

Performance Optimization

To ensure optimal performance and efficient data retrieval, various optimization techniques will be implemented in the ShopNow e-commerce platform.

1. Query Optimization

Optimize SQL queries by rewriting complex queries, avoiding unnecessary joins or subqueries, and utilizing appropriate aggregation functions.

Implement query caching mechanisms to improve response times for frequently executed queries.

2. Indexing

Create indexes on frequently queried columns to enhance data retrieval performance.

Utilize composite indexes for queries involving multiple columns.

3. Caching

Implement caching mechanisms, such as Redis or Memcached, to cache frequently accessed data and reduce database load.

Cache static content, such as product images and descriptions, to improve response times.

Security and Privacy

Security and privacy are of utmost importance in an e-commerce platform, as sensitive information such as user credentials and payment details are involved. The ShopNow platform will implement the following measures to ensure data security and user privacy:

1. Data Encryption

Implement encryption mechanisms, such as bcrypt or Argon2, to securely store user passwords in the database.

Encrypt sensitive data, such as payment information, using industry-standard encryption algorithms like AES.

2. Authentication and Authorization

Implement secure authentication mechanisms, such as JSON Web Tokens (JWT) or session management, to ensure only authorized users can access specific resources.
Implement role-based access control (RBAC) to restrict access to sensitive functionalities based on user roles (e.g., customer, seller, admin).

3. Input Validation and Sanitization

Implement input validation and sanitization techniques to prevent common web vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
Utilize libraries or frameworks that provide built-in protection against such vulnerabilities, such as Laravel's built-in security features.

Analytics and Reporting

To gain valuable insights into customer behavior, sales trends, and seller performance, the ShopNow platform will integrate analytics and reporting capabilities. This will involve:

Integrating analytics tools like Google Analytics to track user behavior, website traffic, and conversion rates.

Developing dashboards and reports for sellers to monitor their product listings, sales performance, and inventory levels.

Generating sales reports and analytics for administrators to analyze platform-wide performance and identify areas for improvement.

Conclusion

The e-commerce platform aims to provide a seamless and secure online shopping experience for customers while enabling third-party companies to list and manage their products efficiently. The project's first phase focuses on creating a robust MySQL relational database to store and manage various entities involved in the e-commerce ecosystem.

The comprehensive database design, incorporating normalization principles and performance optimization techniques, ensures data integrity, efficiency, and scalability. The backend functionalities, including user authentication, product management, order processing, and user profiles, are designed to meet the platform's functional requirements.

The integration of analytics and reporting capabilities will provide valuable insights into customer behavior, sales trends, and seller performance, enabling data-driven business decisions and continuous improvement of the platform. Overall, the e-commerce platform aims to foster a thriving e-commerce ecosystem by providing a reliable and scalable platform for third-party sellers while delivering an exceptional shopping experience for customers.

References

[1] Elmasri, R., & Navathe, S. B. (2011). Fundamentals of database systems (6th ed.). Addison-Wesley.

[2] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). Database system concepts (6th ed.). McGraw-Hill.

[3] Li, X., Zhao, X., Xu, W. (Ato), & Pu, W. (2020). Measuring ease of use of mobile applications in e-commerce retailing from the perspective of consumer online shopping behaviour patterns. Journal of Retailing and Consumer Services, 55, 102093.
<https://doi.org/10.1016/j.jretconser.2020.102093>

[4] Fowler, M. (2003). Patterns of enterprise application architecture. Addison-Wesley Professional.

[5] Github references

1. github.com/VaishakVk/inventory
2. github.com/OGDee93/buypy14
3. github.com/mjbhobe/ChocolafStyle