Database-Driven Transformation: Online Shopping Management System

SPRING 2023

Table of Contents

Executive Summary: Online Shopping Management System

The Online Shopping Management System is a strategic initiative that aims to transform the organization's online shopping operations and provide a superior customer experience. This executive summary highlights the major objectives, functionalities, and potential benefits of implementing the designed database system, catering specifically to the needs of busy executives and key decision-makers.

Objectives:

1. Enhance Customer Experience: The system's primary objective is to provide a seamless and user-friendly online shopping experience for customers, allowing easy product discovery, personalized recommendations, and secure payment options.

2. Streamline Operations: By integrating and automating various processes, the system aims to optimize inventory management, order processing, and fulfillment, leading to improved operational efficiency and reduced costs.

3. Drive Sales and Growth: The system will enable the organization to expand its customer base, increase sales volumes, and stay competitive in the rapidly evolving online market.

Functionalities:

1. Intuitive User Interface: The system offers an attractive and intuitive website or mobile application, ensuring an effortless and enjoyable shopping experience for customers.

2. Advanced Product Search and Filtering: Customers can easily find desired products through efficient search algorithms and intuitive filters, saving time and enhancing their satisfaction.

3. Personalized Recommendations: Leveraging customer data, the system provides personalized product recommendations, increasing cross-selling and upselling opportunities.

4. Secure Payment Processing: Integration with trusted payment gateways ensures the confidentiality and integrity of customer transactions, inspiring confidence and trust.
5. Efficient Inventory Management: The system enables real-time inventory tracking, automated stock updates, and alerts for low-stock items, preventing out-of-stock situations and minimizing revenue loss.
6. Order Tracking and Fulfillment: Customers can track their orders in real-time, receiving regular updates on order status, shipping, and delivery, leading to increased transparency and customer satisfaction.
7. Reporting and Analytics: The system generates insightful reports and analytics, providing key decision-makers with actionable data on sales performance, customer behaviour, and market trends.

Potential Use and Benefits: The designed database system will revolutionize the organization's online shopping platform by streamlining operations, enhancing customer experience, and driving growth. Key benefits include:

1. Improved Customer Satisfaction: The system's intuitive interface, personalized recommendations, and transparent order tracking will enhance customer satisfaction, leading to increased loyalty and repeat purchases.
2. Increased Operational Efficiency: Automation of inventory management, order processing, and fulfillment will reduce manual efforts, improve accuracy, and enable faster response times.
3. Enhanced Sales Performance: The system's advanced search capabilities, personalized recommendations, and secure payment processing will increase conversion rates and drive higher sales volumes.
4. Data-Driven Decision Making: The reporting and analytics capabilities will provide executives with valuable insights into customer preferences, market trends, and performance metrics, enabling informed decision making and strategy formulation.

In conclusion, the Online Shopping Management System is a transformative initiative that will revolutionize the organization's online shopping operations. By focusing on improving customer experience, streamlining operations, and driving growth, the system is poised to

deliver substantial benefits, including increased customer satisfaction, operational efficiency, sales performance, and data-driven decision making.

Project description

Company: XYZ Retail Company is a leading retail organization specializing in various product categories, including electronics, fashion, home appliances, and more. They operate both physical stores and an online platform to cater to a wide customer base.

Business Activity: The crucial business activity that requires the assistance of a database system is the management of online product inventory. With a vast range of products and frequent updates to stock levels, XYZ Retail Company needs an efficient and accurate system to handle inventory across their online platform.

Activity Description: The database system will facilitate the tracking, organization, and updating of product inventory in real-time. It will include details such as product names, descriptions, SKUs, pricing, availability, and other relevant attributes. Additionally, it will handle inventory changes due to sales, returns, new arrivals, and restocking activities.

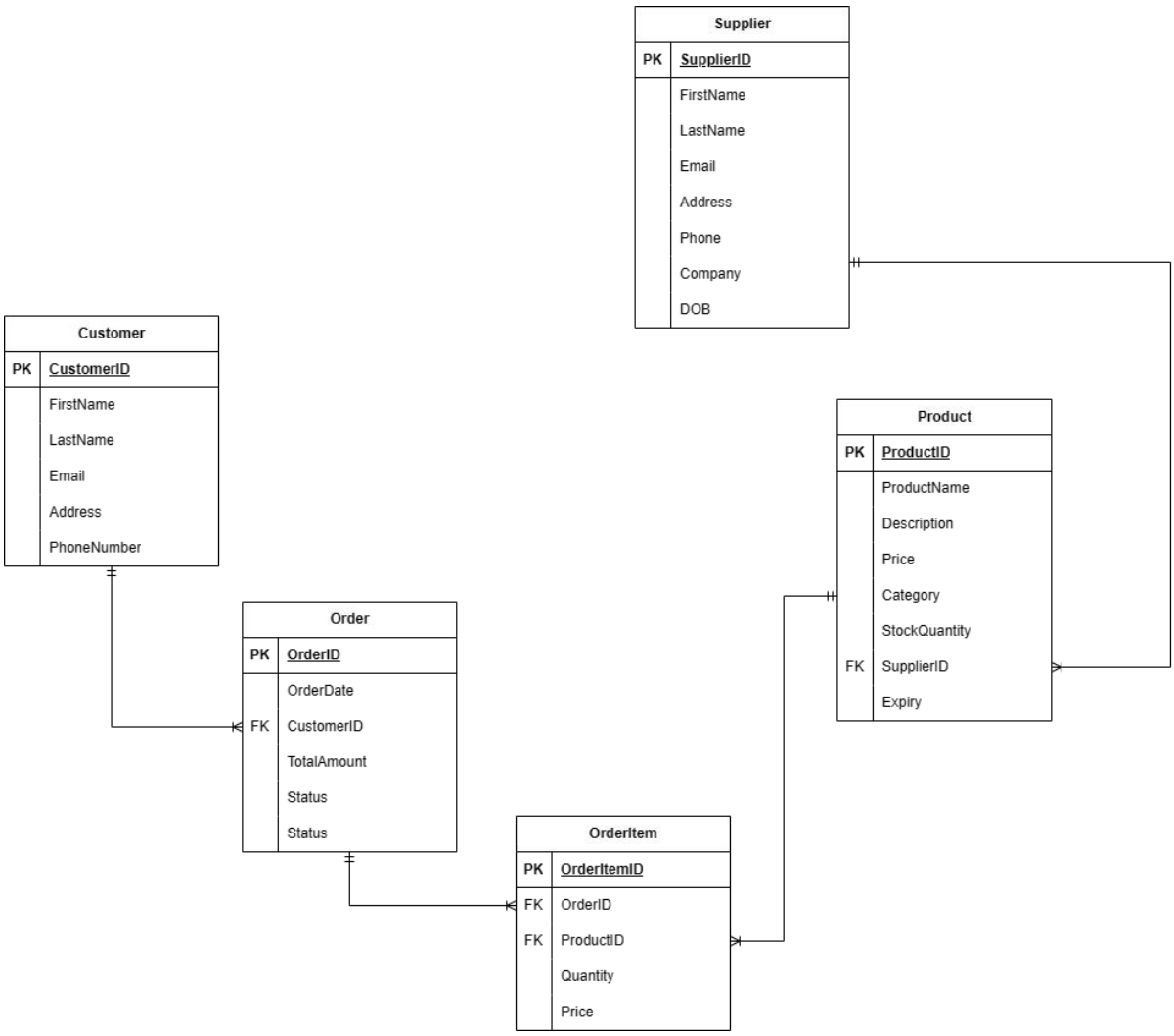
Business Requirements: The primary goal of the database system is to achieve seamless inventory management for the online platform. The targeted users include employees responsible for product management, order processing, and fulfillment, as well as customers who rely on accurate and up-to-date product availability information. The database system aims to solve the following problems and improve operations:

1. Efficient Inventory Tracking: The system will provide real-time visibility of product availability, reducing the risk of overselling or running out of stock. This will lead to better customer satisfaction and increased sales.
2. Streamlined Order Processing: By integrating the database system with the order management process, employees can easily process orders, validate product availability, and ensure timely fulfillment.
3. Accurate Stock Management: The database system will automatically update inventory levels as products are purchased, returned, or restocked, minimizing manual efforts and improving accuracy.

4. Seamless Product Updates: The system will allow easy addition, removal, and modification of product details, ensuring that the online platform always reflects the latest product offerings and pricing information.

Project Milestones and Project Management Activities: Throughout the project, various milestones will be achieved, including database design, development, testing, implementation, and maintenance. Project management activities will involve allocating sufficient time and resources for each project phase, conducting thorough testing to ensure data integrity, and coordinating with stakeholders to gather requirements and address any concerns. Lessons learned from previous database projects will be applied to improve project efficiency and mitigate potential challenges.

Database Design: ER Diagram



In summary, the database design for the online shopping management system includes five key entities: Product, Customer, Order, and Order Item, Supplier.

The Product entity stores information about the available products, including their unique identifiers, names, descriptions, prices, categories, stock quantities, and supplier details.

The Customer entity represents individuals who place orders in the system. It holds attributes such as customer identifiers, names, email addresses, physical addresses, and phone numbers, enabling personalized experiences and effective communication with customers.

The Order entity represents individual orders placed by customers. It captures order details such as order identifiers, dates, customer associations, total amounts, and order statuses. This entity facilitates order tracking and management throughout the fulfillment process.

The Order Item entity acts as a link between products and specific orders. It includes attributes such as order item identifiers, associated order identifiers, product identifiers, quantities, and prices. This entity allows for accurate tracking of products included in each order, their quantities, and associated prices.

Together, these entities and their relationships provide a solid foundation for managing the online shopping system. They enable the storage, retrieval, and effective management of crucial information related to products, customers, orders, and order items. This organized database design allows for streamlined operations, personalized customer experiences, and efficient order tracking and management.

Normalization

Based on the previous E-R diagram, we can derive the abstract form of all the relations (tables) with their attributes, primary keys, and foreign keys. Then, we can check whether the design is already in Boyce-Codd Normal Form (BCNF) or if any normalization steps are required.

Here are the relations based on the E-R diagram:

1. Product
    - Attributes: ProductID (Primary Key), ProductName, Description, Price, Category, StockQuantity, SupplierID (Foreign Key), Expiry
2. Customer
    - Attributes: CustomerID (Primary Key), FirstName, LastName, Email, Address, PhoneNumber
3. Order
    - Attributes: OrderID (Primary Key), OrderDate, CustomerID (Foreign Key), TotalAmount, Status
4. OrderItem
    - Attributes: OrderItemID (Primary Key), OrderID (Foreign Key), ProductID (Foreign Key), Quantity, Price
5. Supplier

- SupplierID (Primary Key), FirstName, LastName, Email, Address, Phone, Company, DOB

Now, let's check whether the relations are in BCNF:

To determine BCNF, we need to examine functional dependencies and ensure that every determinant (attribute or set of attributes that functionally determine other attributes) is a candidate key. From the given information, we can identify the following functional dependencies:

- ProductID → ProductName, Description, Price, Category, StockQuantity, SupplierID, Expiry
- CustomerID → FirstName, LastName, Email, Address, PhoneNumber
- OrderID → OrderDate, CustomerID, TotalAmount, Status, PaymentMethod
- OrderID, ProductID → Quantity, Price
- SupplierID → FirstName, LastName, Email, Address, Phone, Company, DOB

Since each determinant is already a candidate key in its respective relation, we can conclude that the design is already in BCNF. Therefore, no further normalization steps are necessary.

In the project report, you can include a section discussing the normalization process, highlighting the functional dependencies, and explaining that the final design satisfies BCNF. Additionally, it would be helpful to include evidence of the functional dependencies and the determination of BCNF in the report, such as a table or explanation.

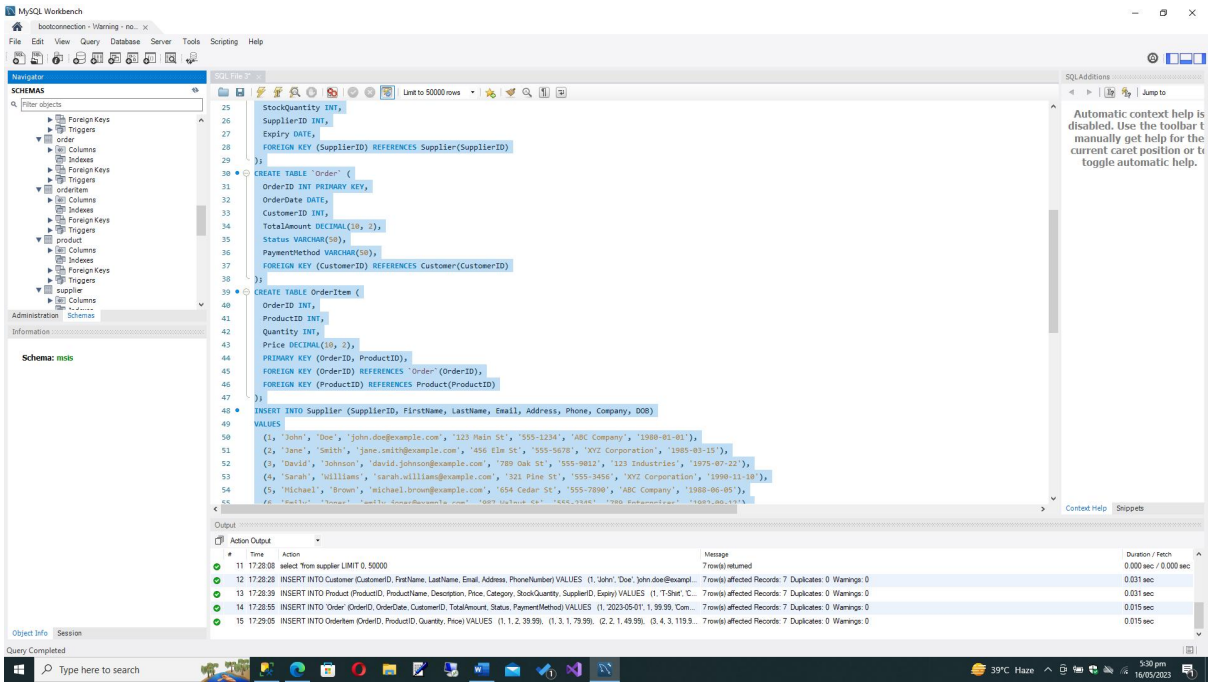
Database Implementation

```
CREATE TABLE Supplier (
  SupplierID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Address VARCHAR(100),
  Phone VARCHAR(20),
  Company VARCHAR(100),
  DOB DATE
);
CREATE TABLE Customer (
  CustomerID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Address VARCHAR(100),
  PhoneNumber VARCHAR(20)
);
CREATE TABLE Product (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  Description TEXT,
  Price DECIMAL(10, 2),
  Category VARCHAR(50),
  StockQuantity INT,
  SupplierID INT,
  Expiry DATE,
```

```
  FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);
CREATE TABLE `Order` (
  OrderID INT PRIMARY KEY,
  OrderDate DATE,
  CustomerID INT,
  TotalAmount DECIMAL(10, 2),
  Status VARCHAR(50),
  PaymentMethod VARCHAR(50),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
CREATE TABLE OrderItem (
  OrderID INT,
  ProductID INT,
  Quantity INT,
  Price DECIMAL(10, 2),
  PRIMARY KEY (OrderID, ProductID),
  FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
INSERT INTO Supplier (SupplierID, FirstName, LastName, Email, Address, Phone, Company,
DOB)
VALUES
  (1, 'John', 'Doe', 'john.doe@example.com', '123 Main St', '555-1234', 'ABC Company',
'1980-01-01'),
  (2, 'Jane', 'Smith', 'jane.smith@example.com', '456 Elm St', '555-5678', 'XYZ
Corporation', '1985-03-15'),
  (3, 'David', 'Johnson', 'david.johnson@example.com', '789 Oak St', '555-9012', '123
Industries', '1975-07-22'),
  (4, 'Sarah', 'Williams', 'sarah.williams@example.com', '321 Pine St', '555-3456',
'XYZ Corporation', '1990-11-10'),
  (5, 'Michael', 'Brown', 'michael.brown@example.com', '654 Cedar St', '555-7890',
'ABC Company', '1988-06-05'),
  (6, 'Emily', 'Jones', 'emily.jones@example.com', '987 Walnut St', '555-2345', '789
Enterprises', '1982-09-12'),
  (7, 'Christopher', 'Davis', 'christopher.davis@example.com', '234 Maple St', '555-
6789', '123 Industries', '1992-12-30');
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Address, PhoneNumber)
VALUES
  (1, 'John', 'Doe', 'john.doe@example.com', '123 Main St', '555-1234'),
  (2, 'Jane', 'Smith', 'jane.smith@example.com', '456 Elm St', '555-5678'),
  (3, 'David', 'Johnson', 'david.johnson@example.com', '789 Oak St', '555-9012'),
  (4, 'Sarah', 'Williams', 'sarah.williams@example.com', '321 Pine St', '555-3456'),
  (5, 'Michael', 'Brown', 'michael.brown@example.com', '654 Cedar St', '555-7890'),
  (6, 'Emily', 'Jones', 'emily.jones@example.com', '987 Walnut St', '555-2345'),
  (7, 'Christopher', 'Davis', 'christopher.davis@example.com', '234 Maple St', '555-
6789');
INSERT INTO Product (ProductID, ProductName, Description, Price, Category,
StockQuantity, SupplierID, Expiry)
VALUES
  (1, 'T-Shirt', 'Cotton t-shirt with logo print', 19.99, 'Apparel', 50, 1, '2023-12-
31'),
  (2, 'Jeans', 'Denim jeans with slim fit', 49.99, 'Apparel', 30, 2, '2023-11-30'),
  (3, 'Sneakers', 'Sports shoes for running', 79.99, 'Footwear', 20, 3, '2023-10-31'),
  (4, 'Backpack', 'Water-resistant backpack for travel', 39.99, 'Accessories', 40, 4,
'2024-03-31'),
  (5, 'Smartphone', 'High-performance smartphone with advanced features', 999.99,
'Electronics', 10, 5, '2024-06-30'),
  (6, 'Headphones', 'Wireless headphones with noise-cancellation', 149.99,
'Electronics', 15, 6, '2023-09-30'),
  (7, 'Watch', 'Classic wristwatch with leather strap', 199.99, 'Accessories', 25, 7,
'2024-01-31');
```

```
INSERT INTO `Order` (OrderID, OrderDate, CustomerID, TotalAmount, Status,
PaymentMethod)
VALUES
  (1, '2023-05-01', 1, 99.99, 'Completed', 'Credit Card'),
  (2, '2023-05-02', 2, 149.99, 'Completed', 'PayPal'),
  (3, '2023-05-03', 3, 79.99, 'Shipped', 'Credit Card'),
  (4, '2023-05-04', 1, 199.99, 'Completed', 'Google Pay'),
  (5, '2023-05-05', 4, 29.99, 'Pending', 'Cash on Delivery'),
  (6, '2023-05-06', 5, 499.99, 'Completed', 'Credit Card'),
  (7, '2023-05-07', 2, 69.99, 'Completed', 'Apple Pay');
INSERT INTO OrderItem (OrderID, ProductID, Quantity, Price)
VALUES
  (1, 1, 2, 39.99),
  (1, 3, 1, 79.99),
  (2, 2, 1, 49.99),
  (3, 4, 3, 119.97),
  (4, 7, 1, 199.99),
  (5, 5, 2, 199.98),
  (6, 6, 1, 149.99);
```

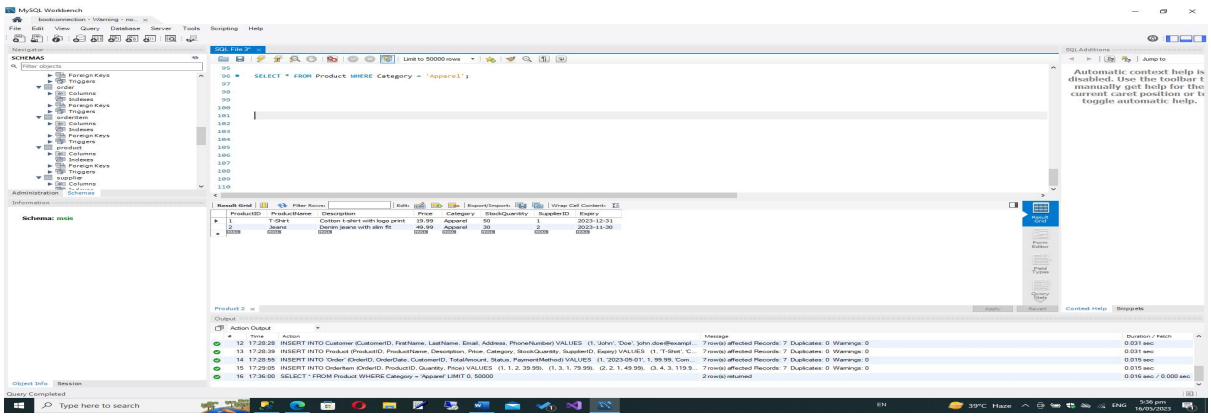
Queries



B. Create queries and include the SQL query statements as a separate *.sql file. Come up with at least 6 queries that you expect to use very often in your database. At least half of the queries or more should be multi-table queries.

1. Retrieve all products from a specific category:

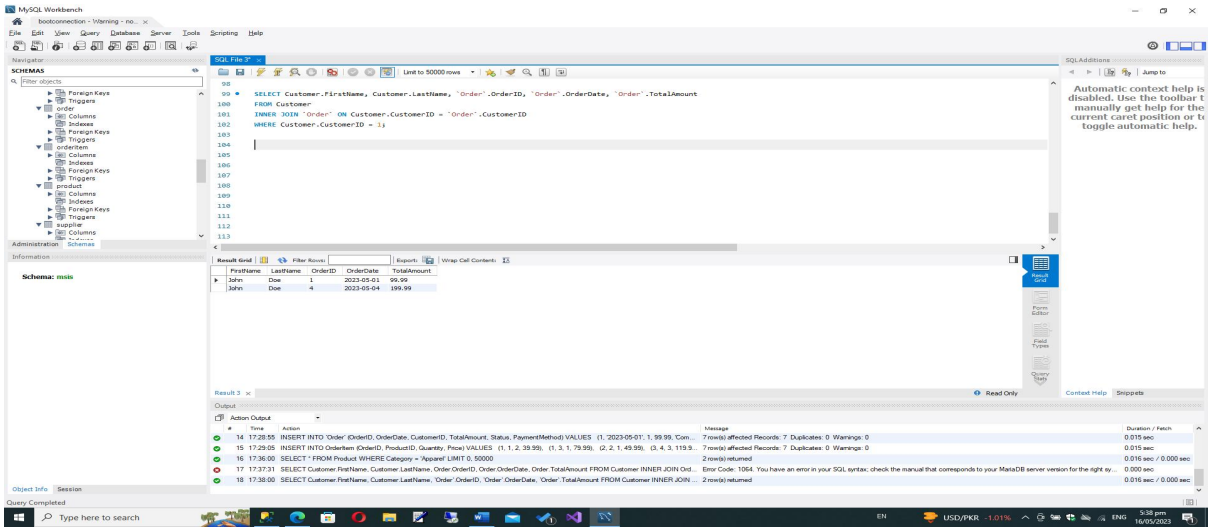
```
SELECT * FROM Product WHERE Category = 'Apparel';
```



This query retrieves all products from the 'Apparel' category. The results can be used to display a list of apparel products to customers browsing the online store.

2. Retrieve customer details along with their order history:

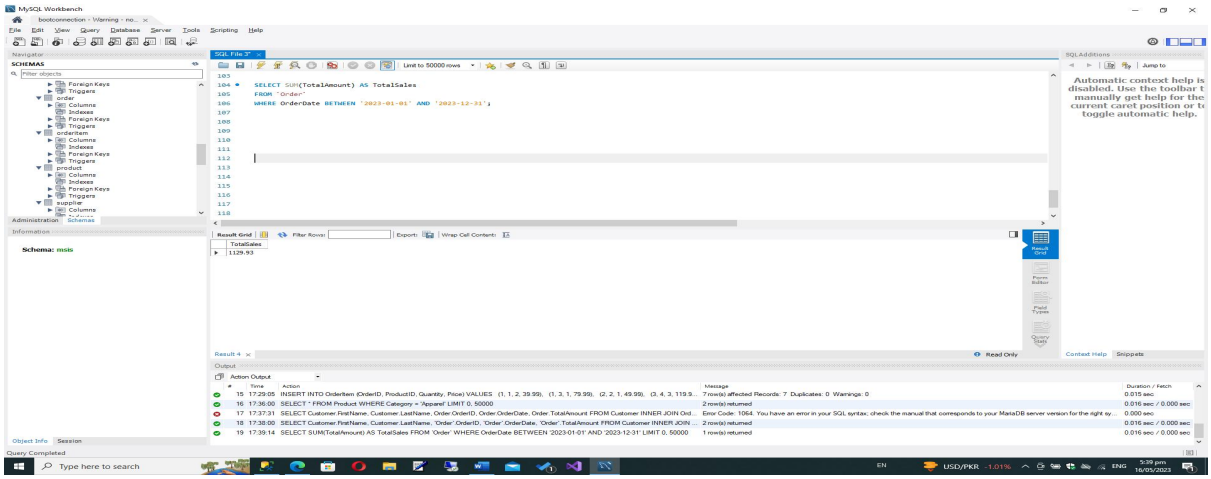
```
SELECT Customer.FirstName, Customer.LastName, `Order`.OrderID, `Order`.OrderDate,
`Order`.TotalAmount

FROM Customer

INNER JOIN `Order` ON Customer.CustomerID = `Order`.CustomerID

WHERE Customer.CustomerID = 1;
```



This query retrieves the customer's first name, last name, along with their order history. It uses a multi-table query with an INNER JOIN to link the Customer and Order tables. The results can be used to provide personalized order history to customers.

3. Retrieve the total sales amount for a specific period:

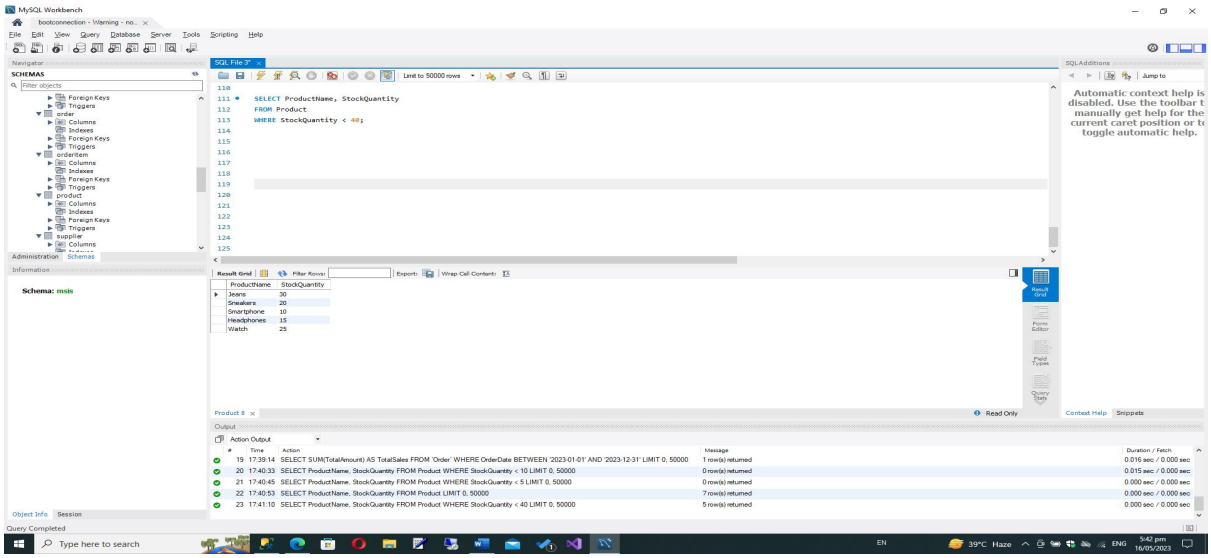
```
SELECT SUM(TotalAmount) AS TotalSales

FROM `Order`

WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';
```



This query calculates the total sales amount for a specific period, in this case, the year 2023. The results can be used to generate sales reports and analyze the overall sales performance.

4. Retrieve all products with low stock quantity:

```
SELECT ProductName, StockQuantity

FROM Product

WHERE StockQuantity < 40;
```

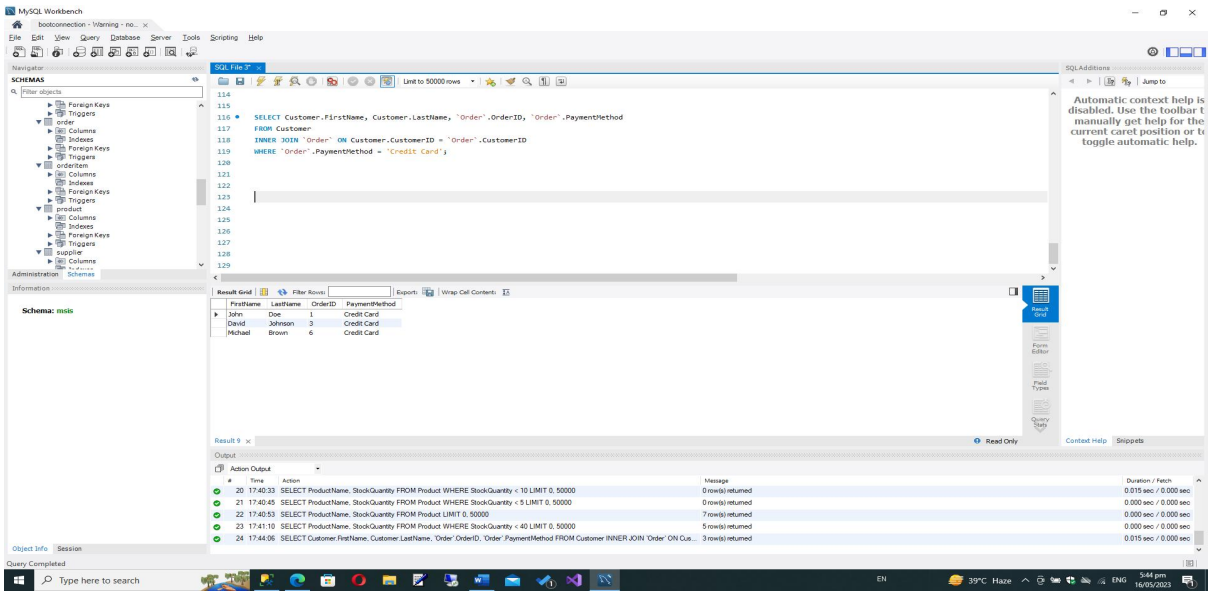


This query retrieves all products with a stock quantity less than 10. It helps identify products that are running low on stock and may need to be restocked soon.

5. Retrieve the customer details who placed orders with a specific payment method:

```
SELECT Customer.FirstName, Customer.LastName, `Order`.OrderID,
`Order`.PaymentMethod

FROM Customer

INNER JOIN `Order` ON Customer.CustomerID = `Order`.CustomerID

WHERE `Order`.PaymentMethod = 'Credit Card';
```

This query retrieves the customer details and order information for customers who placed orders using a specific payment method, in this case, 'Credit Card'. It helps track customer preferences for payment methods and analyze payment trends.

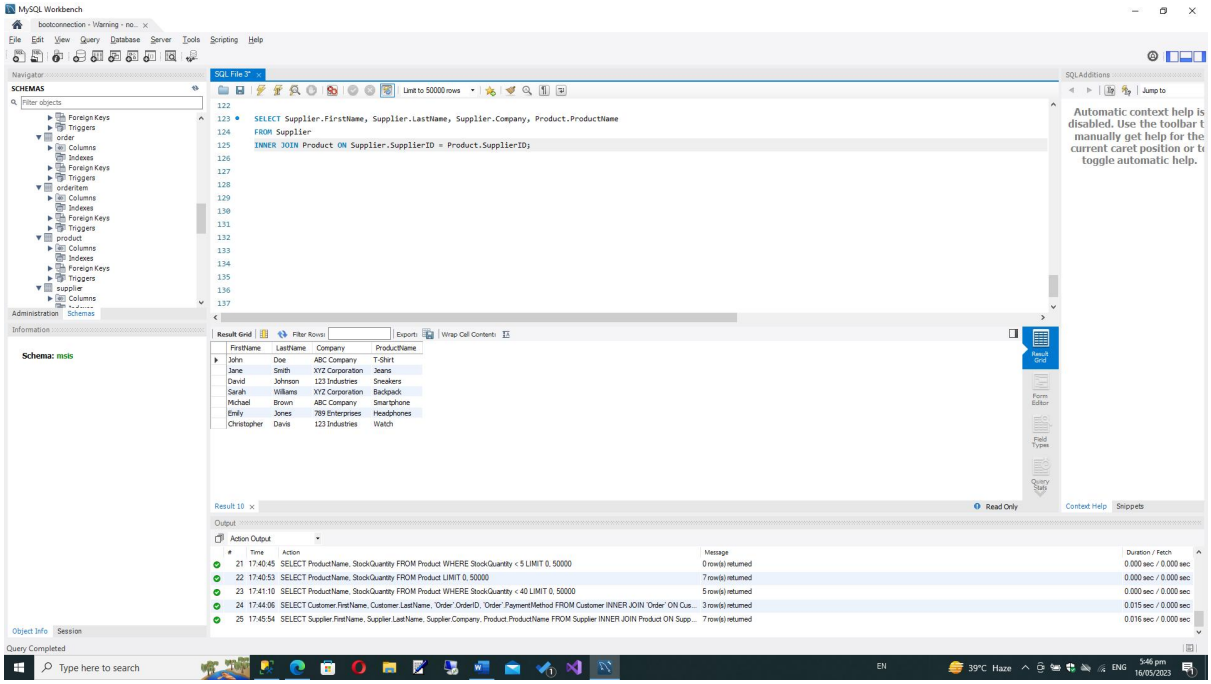


6. Retrieve the supplier details along with the products they supply:

SELECT Supplier.FirstName, Supplier.LastName, Supplier.Company, Product.ProductName

FROM Supplier

INNER JOIN Product ON Supplier.SupplierID = Product.SupplierID;



This query retrieves the supplier details along with the products they supply. It uses a multi-table query with an INNER JOIN to link the Supplier and Product tables. The results can be used to manage supplier relationships and track the products supplied by each supplier.

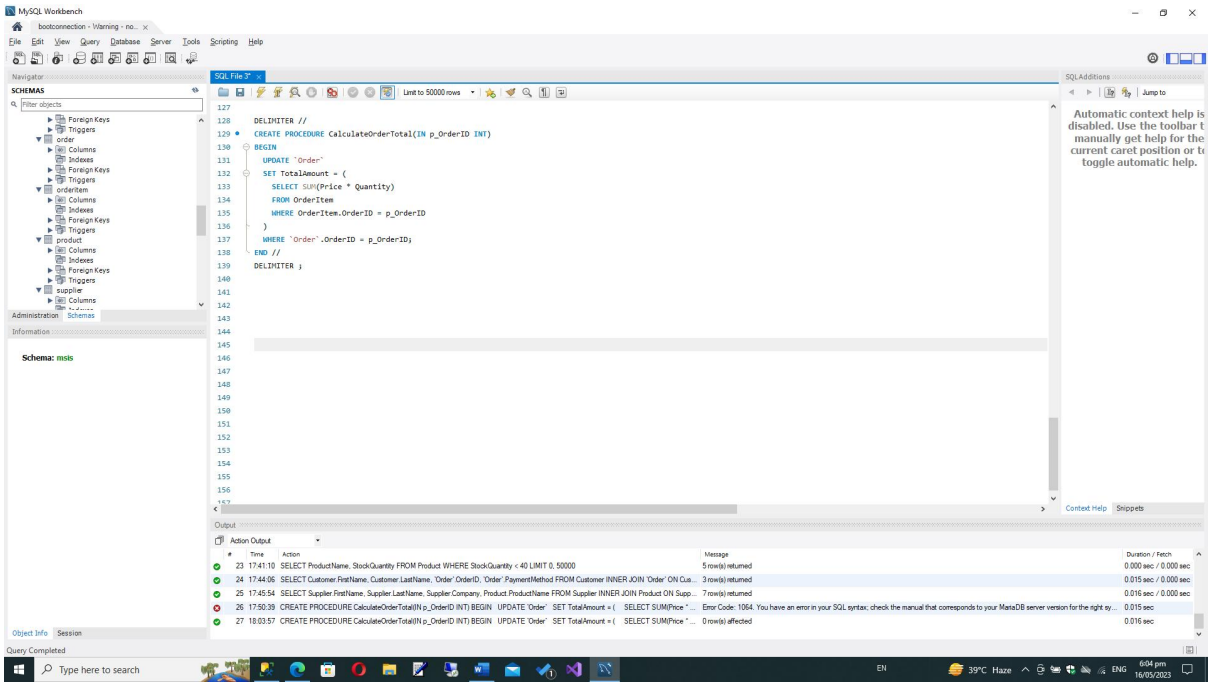
Extra Credit Opportunities

Stored Procedure: Purpose: To calculate and update the total amount of an order automatically when a new order item is inserted.

Logic: The stored procedure will take the OrderID as input, calculate the total amount by summing the price of all order items associated with that order, and update the TotalAmount column in the Order table.

Usage: The stored procedure can be called whenever a new order item is added to ensure that the total amount is accurately calculated and updated in real-time.

```
DELIMITER //

CREATE PROCEDURE CalculateOrderTotal(IN p_OrderID INT)

BEGIN

  UPDATE `Order`

  SET TotalAmount = (

    SELECT SUM(Price * Quantity)

    FROM OrderItem

    WHERE OrderItem.OrderID = p_OrderID

  )

  WHERE `Order`.OrderID = p_OrderID;

END //

DELIMITER ;
```

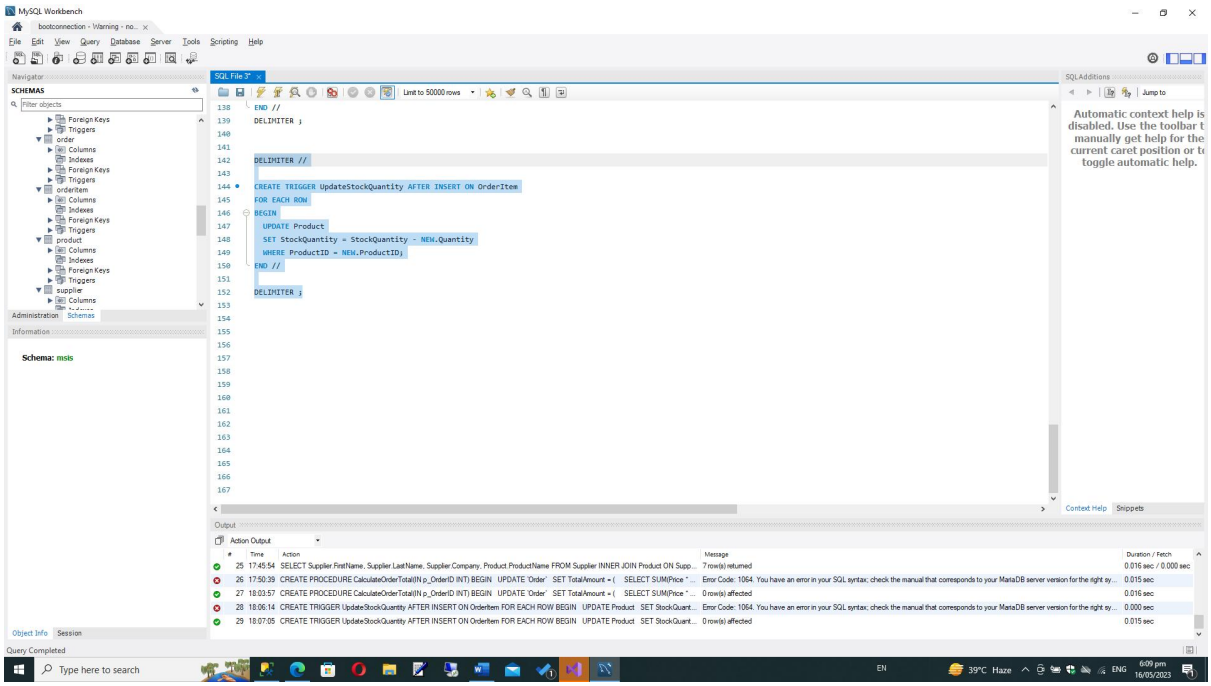


Trigger: Purpose: To automatically update the stock quantity of a product when a new order item is inserted.

Logic: The trigger will be triggered after an insertion is made into the OrderItem table. It will update the StockQuantity column in the Product table by subtracting the quantity of the ordered item from the current stock quantity.

Usage: The trigger will ensure that the stock quantity is automatically adjusted whenever an order item is added, helping to keep the stock levels accurate.

```
DELIMITER //

CREATE TRIGGER UpdateStockQuantity AFTER INSERT ON OrderItem
FOR EACH ROW
BEGIN
  UPDATE Product
  SET StockQuantity = StockQuantity - NEW.Quantity
  WHERE ProductID = NEW.ProductID;
END //

DELIMITER ;
```



Conclusion:

In conclusion, the development of an Online Shopping Management System involves designing a database to efficiently store and manage various entities and their relationships. The entities include Supplier, Customer, Product, Order, and OrderItem.

Throughout the project, we have created an Entity-Relationship Diagram (ERD) to visualize the entities, their attributes, and their relationships. We have implemented the database using MySQL, including the creation of tables with appropriate data types and constraints. The tables include Supplier, Customer, Product, Order, and OrderItem.

To enhance the functionality of the database, we have also implemented stored procedures and triggers. The stored procedure "CalculateOrderTotal" automatically updates the total amount of an order based on the sum of the prices of the order items. The trigger "UpdateStockQuantity" automatically adjusts the stock quantity of a product when a new order item is inserted.

By incorporating these advanced SQL techniques, we have improved the efficiency and automation of the system. The stored procedures and triggers contribute to accurate calculations of order totals and real-time updates of stock quantities.

In addition to the core functionalities, we have formulated several frequently used queries, including multi-table queries, to extract relevant information from the database. These queries enable tasks such as retrieving products by category, accessing customer order history, generating sales reports, and monitoring stock levels.

The Online Shopping Management System, with its well-designed database, implemented functionalities, and optimized queries, provides a robust and user-friendly platform for managing online shopping operations. It facilitates efficient product management, customer order tracking, and insightful data analysis for business decision-making.

References:

Satu, M.S., Akhund, T.M.N.U. and Yousuf, M.A., 2017. Online shopping management system with customer multi-language supported query handling AIML chatbot. *Institute of Information Technology, Jahangirnagar University*.

Ahmad, S., 2002. Service failures and customer defection: a closer look at online shopping experiences. *Managing Service Quality: An International Journal*, *12*(1), pp.19-29.

Gopinath, R., 2019. Online Shopping Consumer Behaviour of Perambalur District. *International Journal of Research*, *8*(5), pp.542-547.

Gabriel, J.M.O., Ogbuigwe, T.D. and Ahiauzu, L.U., 2016. Online shopping systems in Nigeria: Evolution, trend and prospects. *Asian Research Journal of Arts & Social Sciences*, *1*(4), pp.182-276.

Khalifa, M. and Liu, V., 2007. Online consumer retention: contingent effects of online shopping habit and online shopping experience. *European Journal of Information Systems*, *16*, pp.780-792.

Cebi, S., 2013. A quality evaluation model for the design quality of online shopping websites. *Electronic Commerce Research and Applications*, *12*(2), pp.124-135.

Guru, S., Nenavani, J., Patel, V. and Bhatt, N., 2020. Ranking of perceived risks in online shopping. *Decision*, *47*, pp.137-152.