DATABASE MANAGEMENT SYSTEM - CSA0593

ASSIGNMENT 5

K.SAI YASWANTH

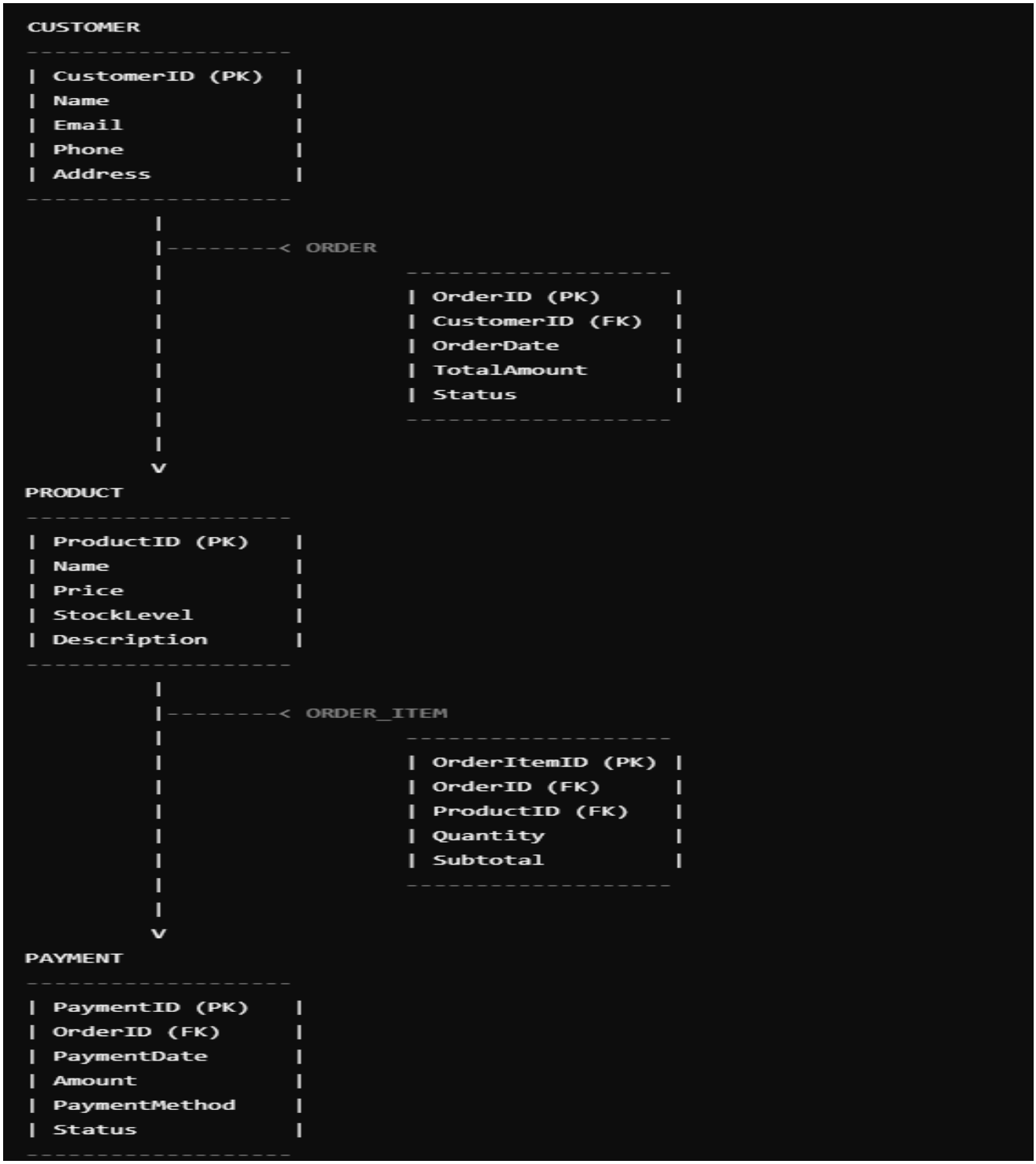
192373061

QUESTION:

- Design a database schema and write SQL code for managing customers, products, orders, and payments.
  Model tables for customers, products, orders, and payments.
- Write stored procedures for placing orders and processing refunds.
- Implement triggers to update stock levels and order statuses.
- Write SQL queries to analyze order trends and product popularity.

ANSWER:

CONCEPTUAL E.R.DIAGRAM:



LOGICAL E.R.DIAGRAM:

PHYSICAL E.R.DIAGRAM:

MYSQL STATEMENTS:

```
mysql
CREATE DATABASE EcommerceDB;

USE EcommerceDB;

CREATE TABLE Customers (
  CustomerID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Phone VARCHAR(20)
);

CREATE TABLE Products (
  ProductID INT AUTO_INCREMENT PRIMARY KEY,
  ProductName VARCHAR(100),
  ProductDescription VARCHAR(255),
  Price DECIMAL(10, 2),
  StockLevel INT
);

CREATE TABLE Orders (
  OrderID INT AUTO_INCREMENT PRIMARY KEY,
  CustomerID INT,
```

```
  OrderDate DATE,
  OrderStatus VARCHAR(20),
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE OrderItems (
  OrderItemID INT AUTO_INCREMENT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT,
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);

CREATE TABLE Payments (
  PaymentID INT AUTO_INCREMENT PRIMARY KEY,
  OrderID INT,
  PaymentDate DATE,
  PaymentAmount DECIMAL(10, 2),
  PaymentMethod VARCHAR(50),
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Stored Procedures:

```
mysql
DELIMITER //

CREATE PROCEDURE sp_PlaceOrder(
  IN customerID INT,
  IN orderDate DATE
)
BEGIN
  INSERT INTO Orders (CustomerID, OrderDate, OrderStatus)
  VALUES (customerID, orderDate, 'Pending');

  DECLARE newOrderID INT;
  SET newOrderID = LAST_INSERT_ID();

  INSERT INTO OrderItems (OrderID, ProductID, Quantity)
  VALUES (newOrderID, (SELECT ProductID FROM Products LIMIT 1), 1);
END //

CREATE PROCEDURE sp_ProcessRefund(
  IN orderID INT,
  IN paymentID INT,
  IN refundAmount DECIMAL(10, 2)
)
BEGIN
  UPDATE Payments
```

```
  SET PaymentAmount = PaymentAmount - refundAmount
  WHERE PaymentID = paymentID;

  UPDATE Orders
  SET OrderStatus = 'Refunded'
  WHERE OrderID = orderID;
END //

DELIMITER;
```

Triggers:

```
mysql
DELIMITER //

CREATE TRIGGER tr_UpdateStockLevel
AFTER INSERT ON OrderItems
FOR EACH ROW
BEGIN
  UPDATE Products
  SET StockLevel = StockLevel - NEW.Quantity
  WHERE ProductID = NEW.ProductID;
END //
```

```
CREATE TRIGGER tr_UpdateOrderStatus
AFTER UPDATE ON Payments
FOR EACH ROW
BEGIN
  UPDATE Orders
  SET OrderStatus = 'Paid'
  WHERE OrderID = NEW.OrderID;
END //

DELIMITER;
```

SQL Queries for Analysis:

```
mysql
-- Order Trends
SELECT
  MONTH(OrderDate) AS Month,
  COUNT(*) AS TotalOrders,
  SUM(PaymentAmount) AS TotalRevenue
FROM
  Orders
  JOIN Payments ON Orders.OrderID = Payments.OrderID
GROUP BY
  MONTH(OrderDate);
```

```
-- Product Popularity
SELECT
  ProductName,
  COUNT(*) AS TotalOrders,
  SUM(Quantity) AS TotalQuantity
FROM
  Products
  JOIN OrderItems ON Products.ProductID = OrderItems.ProductID
GROUP BY
  ProductName;

-- Customer Order History
SELECT
  CustomerID,
  FirstName,
  LastName,
  COUNT(*) AS TotalOrders
FROM
  Customers
  JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY
  CustomerID, FirstName, LastName;
```

Conclusion:

This database design provides a comprehensive foundation for managing customers, products, orders, and payments. The stored procedures simplify order placement and refund processing, while the triggers ensure data consistency and accuracy. The SQL queries enable analysis of order trends, product popularity, and customer order history.