SQL Code

```sql

CREATE DATABASE ecommerce;

--Customers table
CREATE TABLE Customers (
    CustomerID INT IDENTITY PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    City VARCHAR(50),
    JoinDate DATE
);


--Products table
CREATE TABLE Products (
    ProductID INT IDENTITY PRIMARY KEY,
    ProductName VARCHAR(100),
    Category VARCHAR(50),
    Price DECIMAL(10,2),
    StockQuantity INT,
    IsActive BIT
);


--Orders table
CREATE TABLE Orders (
    OrderID INT IDENTITY PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    OrderStatus VARCHAR(30),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);


--OrderItems table
CREATE TABLE OrderItems (
    OrderItemID INT IDENTITY PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    ItemPrice DECIMAL(10,2),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);


--Payments table
CREATE TABLE Payments (
    PaymentID INT IDENTITY PRIMARY KEY,
    OrderID INT,
    PaymentDate DATE,
    Amount DECIMAL(10,2),
    PaymentMode VARCHAR(30),
    PaymentStatus VARCHAR(30),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);


INSERT INTO Customers (Name, Email, Phone, City, JoinDate)
VALUES
('Rahul Sharma', 'rahul@gmail.com', '9876543210', 'Bangalore', '2024-01-10'),
('Anita Rao', 'anita@gmail.com', '9876543211', 'Mysore', '2024-01-12'),
('Suresh Kumar', 'suresh@gmail.com', '9876543212', 'Chennai', '2024-02-01'),
('Priya Singh', 'priya@gmail.com', '9876543213', 'Delhi', '2024-02-05'),
('Amit Verma', 'amit@gmail.com', '9876543214', 'Mumbai', '2024-02-10'),
('Neha Patel', 'neha@gmail.com', '9876543215', 'Ahmedabad', '2024-02-15'),
('Rohit Mehta', 'rohit@gmail.com', '9876543216', 'Pune', '2024-03-01'),
('Kavya N', 'kavya@gmail.com', '9876543217', 'Bangalore', '2024-03-05'),
('Manoj Das', 'manoj@gmail.com', '9876543218', 'Kolkata', '2024-03-10'),
('Sneha Iyer', 'sneha@gmail.com', '9876543219', 'Chennai', '2024-03-15');


INSERT INTO Products (ProductName, Category, Price, StockQuantity, IsActive)
VALUES
('Laptop', 'Electronics', 65000, 20, 1),
('Smartphone', 'Electronics', 30000, 30, 1),
('Headphones', 'Electronics', 2500, 50, 1),
('Keyboard', 'Electronics', 1500, 40, 1),
('Office Chair', 'Furniture', 8000, 15, 1),
('Study Table', 'Furniture', 12000, 10, 1),
('Mixer Grinder', 'Home Appliances', 4500, 25, 1),
('Microwave Oven', 'Home Appliances', 15000, 12, 1),
('Running Shoes', 'Fashion', 3500, 60, 1),
('Wrist Watch', 'Fashion', 7000, 35, 1);


INSERT INTO Orders (CustomerID, OrderDate, OrderStatus)
VALUES
(1, '2024-04-01', 'Pending'),
(2, '2024-04-02', 'Delivered'),
(3, '2024-04-03', 'Pending'),
(4, '2024-04-04', 'Cancelled'),
(5, '2024-04-05', 'Delivered'),
(6, '2024-04-06', 'Pending'),
(7, '2024-04-07', 'Delivered'),
(8, '2024-04-08', 'Pending'),
(9, '2024-04-09', 'Delivered'),
(10,'2024-04-10', 'Pending');


INSERT INTO OrderItems (OrderID, ProductID, Quantity, ItemPrice)
VALUES
(1, 1, 1, 65000),
(2, 2, 1, 30000),
(3, 3, 2, 5000),
(4, 4, 1, 1500),
(5, 5, 1, 8000),
(6, 6, 1, 12000),
(7, 7, 2, 9000),
(8, 8, 1, 15000),
(9, 9, 2, 7000),
(10,10,1, 7000);


INSERT INTO Payments (OrderID, PaymentDate, Amount, PaymentMode, PaymentStatus) VALUES
(1, '2024-04-01', 65000, 'UPI', 'Pending'),
(2, '2024-04-02', 30000, 'Credit Card', 'Success'),
(3, '2024-04-03', 5000, 'Debit Card', 'Failed'),
(4, '2024-04-04', 1500, 'UPI', 'Refunded'),
(5, '2024-04-05', 8000, 'Net Banking', 'Success'),
(6, '2024-04-06', 12000, 'UPI', 'Pending'),
(7, '2024-04-07', 9000, 'Credit Card', 'Success'),
(8, '2024-04-08', 15000, 'Debit Card', 'Pending'),
(9, '2024-04-09', 7000, 'UPI', 'Success'),
(10,'2024-04-10', 7000, 'Net Banking', 'Pending');


INSERT INTO Customers (Name, Email, Phone, City, JoinDate)
VALUES
('Vikram Singh', 'vikram@gmail.com', '9876543220', 'Bangalore', '2024-03-20'),
('Ananya Iyer', 'ananya@gmail.com', '9876543221', 'Hyderabad', '2024-03-22');


INSERT INTO Products (ProductName, Category, Price, StockQuantity, IsActive)
VALUES
('Desk Lamp', 'Home Appliances', 2000, 25, 1),
('Bluetooth Speaker', 'Electronics', 3500, 15, 1);


-- New Orders for Rahul (CustomerID=1)
INSERT INTO Orders (CustomerID, OrderDate, OrderStatus)
VALUES
(1, '2024-04-11', 'Delivered'),
(1, '2024-04-12', 'Delivered');

-- Add items in different categories
INSERT INTO OrderItems (OrderID, ProductID, Quantity, ItemPrice)
VALUES
(11, 5, 1, 8000),  -- Furniture
(11, 7, 1, 4500),  -- Home Appliances
(12, 9, 1, 3500);  -- Fashion

-- Payments for new orders
INSERT INTO Payments (OrderID, PaymentDate, Amount, PaymentMode, PaymentStatus)
VALUES
```

```sql
(11, '2024-04-11', 12500, 'UPI', 'Success'),
(12, '2024-04-12', 3500, 'Credit Card', 'Success');


SELECT * FROM Customers;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM OrderItems;
SELECT * FROM Payments;


--1. Retrieve all customers who have not placed any orders yet.
SELECT
   c.CustomerID,
   c.Name,
   o.OrderID
FROM Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL;

--or
SELECT
    CustomerID,
    Name
FROM Customers c
WHERE NOT EXISTS (SELECT 1 FROM Orders o WHERE c.CustomerID = o.CustomerID)



-- 2. Find the top 3 products by total sales amount

SELECT
    TOP 3
    p.ProductID,
    p.ProductName,
    SUM(oi.ItemPrice * oi.Quantity) AS TotalSales
FROM OrderItems oi
JOIN Products p
ON p.ProductID = oi.ProductID
GROUP BY p.ProductName, p.ProductID
ORDER BY TotalSales DESC;

--or
SELECT
    TOP 3
    p.ProductID,
    p.ProductName,
    (
     SELECT SUM(oi.ItemPrice * oi.Quantity)
     FROM OrderItems oi
     WHERE oi.ProductID = p.ProductID
    ) AS Totalales
FROM Products p
ORDER BY Totalales DESC;
```

```sql


--3. Retrieve orders with customer name, total order amount,and payment status

SELECT
    c.CustomerID,
    c.Name,
    SUM(oi.ItemPrice * oi.Quantity) AS OrderTotal,
    p.PaymentStatus
FROM Customers c
JOIN Orders o
    ON c.CustomerID = o.CustomerID
JOIN OrderItems oi
    ON oi.OrderID = o.OrderID
JOIN Payments p
    ON p.OrderID = oi.OrderID
WHERE p.PaymentStatus = 'Success'
GROUP BY
    c.CustomerID,
    c.Name,
    p.PaymentStatus;


    --or
SELECT
    c.CustomerID,
    c.Name,
    p.PaymentStatus,
    (
     SELECT SUM(oi.ItemPrice * oi.Quantity)
     FROM OrderItems oi
     WHERE oi.OrderID = o.OrderID
     ) AS ItemTotal
FROM Customers c
JOIN Orders o
ON o.CustomerID = c.CustomerID
JOIN Payments p
ON p.OrderID = o.OrderID
WHERE p.PaymentStatus = 'Success';


--4. Calculate total revenue per month in 2024

SELECT
    YEAR(p.PaymentDate) AS Year,
    MONTH(p.PaymentDate) AS Month,
    SUM(p.Amount) AS TotalRevenue
FROM Payments p
JOIN Orders o
    ON o.OrderID = p.OrderID
WHERE YEAR(p.PaymentDate) = 2024
AND p.PaymentStatus = 'Success'
GROUP BY
     YEAR(p.PaymentDate),
     MONTH(p.PaymentDate);


--5. Find active products that have never been ordered

SELECT
    p.ProductID,
    p.ProductName,
    oi.OrderID,
    p.IsActive
FROM Products p
LEFT JOIN OrderItems oi
ON oi.ProductID = p.ProductID
WHERE oi.OrderID IS NULL AND p.IsActive = 1;

--or
SELECT
    p.ProductID,
    p.Productname,
    p.IsActive
FROM Products p
WHERE p.IsActive = 1
AND NOT EXISTS ( Select 1
                 FROM OrderItems oi
                 WHERE oi.ProductID = p.ProductID
                );


--6. Identify customer(s) who spent the most            ---TOP 1 WITH TIES
SELECT
    TOP 1 WITH TIES
    c.CustomerID,
    c.Name,
    SUM(p.Amount) As TotalSpent
FROM Customers c
JOIN Orders o
    ON o.CustomerID = c.CustomerID
JOIN Payments p
    ON p.OrderID = o.OrderID
GROUP BY
    c.CustomerID,
    c.Name
ORDER BY TotalSpent DESC;

--or Using CTE
WITH HighestSpentCustomers AS (
    SELECT
        c.CustomerID,
        c.Name,
        SUM(p.Amount) AS TotalSpent,
        RANK() OVER(ORDER BY SUM(p.Amount) DESC) AS Rankk
     FROM Customers c
     JOIN Orders o
     ON o.CustomerID = c.CustomerID
     JOIN Payments p
```

```sql
         ON p.OrderID = o.OrderID
         GROUP BY c.CustomerID, c.Name
)
SELECT
    CustomerID,
    Name,
    TotalSpent,
    Rankk
FROM HighestSpentCustomers
WHERE Rankk = 1;


--* 7. Retrieve orders having at least one item priced above 10,000

--SELECT
--    o.OrderID,
--    o.CustomerID,
--    SUM(oi.ItemPrice * oi.Quantity) AS ItemPrice
--FROM Orders o
--JOIN OrderItems oi
--ON oi.OrderID = o.OrderID
--GROUP BY
--    o.CustomerID,
--    o.OrderID
--HAVING SUM(oi.ItemPrice * oi.Quantity) > 10000

--or
SELECT
    o.OrderID,
    o.CustomerID,
    oi.ItemPrice
FROM Orders o
JOIN OrderItems oi
ON oi.OrderID = o.OrderID
WHERE oi.ItemPrice > 10000;

--or
SELECT
    o.OrderID,
    o.CustomerID
FROM Orders o
WHERE EXISTS (
                SELECT 1,
                oi.ItemPrice
                FROM OrderItems oi
                WHERE oi.OrderID = o.OrderID
                AND oi.ItemPrice > 10000
            );



--8. Show first order date and total orders per customer

SELECT
    c.CustomerID,
    MIN(o.OrderDate) AS FirstOrderDate, --3
    COUNT(o.OrderID) AS NoOfOrders
FROM Orders o
JOIN Customers c                        --1
ON c.CustomerID = o.CustomerID
GROUP BY                                --2
    c.CustomerID
ORDER BY FirstOrderDate;                --4


--9. Find average quantity ordered per product category
 SELECT
    AVG(oi.Quantity) as AvgCount,
    p.Category
FROM Products AS p
JOIN OrderItems AS oi
ON p.ProductID = oi.ProductID
GROUP BY p.Category;

 select * from OrderItems


--10. List customers with number of pending payments
SELECT
    c.CustomerID,
    p.PaymentStatus,
    COUNT(p.PaymentID) As NoOfPendingOrders
FROM Payments p
JOIN Orders oi
ON oi.OrderID = p.OrderID
JOIN Customers c
ON c.CustomerID = oi.CustomerID
GROUP BY
    c.CustomerID,
    p.PaymentStatus
HAVING p.PaymentStatus = 'Pending'

select * from Payments;

insert into Payments values(1, '2024-05-22',300.00, 'CASH', 'Pending');

-- 11. Show cumulative sales per day
SELECT
    p1.PaymentDate,
    (
        SELECT SUM(p2.Amount)
        FROM Payments p2
        WHERE p2.PaymentDate <= p1.PaymentDate
          AND p2.PaymentStatus = 'Success'
    ) AS CumulativeSales
FROM Payments p1
WHERE p1.PaymentStatus = 'Success'
GROUP BY p1.PaymentDate
ORDER BY p1.PaymentDate;

```

```sql

--12. Retrieve orders with total amount greater than 20,000
SELECT
    oi.OrderID,
    SUM(oi.ItemPrice * oi.Quantity) AS TotalOrderAmout
FROM OrderItems oi
GROUP BY oi.OrderID
HAVING SUM(oi.ItemPrice * oi.Quantity) > 20000;


--13. Find products with stock < 20 and total quantity sold > 5
SELECT
    p.ProductID,
    p.ProductName,
    oi.Quantity,
    SUM(oi.Quantity) AS TotalQuantitySold
FROM Products p
JOIN OrderItems oi
    ON oi.ProductID = p.ProductID
WHERE p.StockQuantity < 20
GROUP BY p.ProductID,
         p.ProductName,
            oi.Quantity
HAVING SUM(oi.Quantity) > 5;

SELECT * FROM Orders;


--14. Retrieve last 2 orders per customer
WITH LastTwoOrders AS (
    SELECT
        c.CustomerID,
        c.Name,
        o.OrderDate,
        COUNT(o.OrderID) As NoOfOrders,
        DENSE_RANK() OVER(
                            PARTITION BY c.CustomerID
                            ORDER BY o.OrderDate DESC
                          ) AS RecentOrders
    FROM Orders o
    JOIN Customers c
    ON c.CustomerID = o.CustomerID
    GROUP BY
        c.CustomerID,
        c.Name,
        o.OrderDate
)
SELECT lo.*
FROM LastTwoOrders lo
WHERE lo.RecentOrders <=2;


--15. Find customers who ordered more than one product in the same order
 SELECT
    c.CustomerID,
    c.Name,
    o.OrderID,
    COUNT(oi.ProductID) As NoOfProducts
FROM Orders o
JOIN OrderItems oi
ON oi.OrderID = o.OrderID
JOIN Customers c
ON c.CustomerID = o.CustomerID
GROUP BY
     c.CustomerID,
    c.Name,
    o.OrderID
HAVING COUNT(oi.ProductID) > 1;

select * from Products;

--16. Retrieve orders with late payments (3-day rule)


--17. Calculate total revenue lost due to failed or refunded payments



--18. Find customers who ordered from more than 2 categories



--19. Increase price of active electronics products by 10%


--20. Delete cancelled orders with no successful payments
```