```sql
--EVERMART_ONLINE_TRANSACTION_PORTFOLIO_PROJECT

--#To return all the values in the table

SELECT *
FROM Customers

SELECT *
FROM Products

SELECT *
FROM Transactions

--#Total Revenue Generated

SELECT SUM(TotalValue) AS Total_Revenue
FROM Transactions


--#Monthly Revenue Trend

SELECT MONTH(TransactionDate) AS Month, SUM(TotalValue) AS Monthly_Revenue
FROM Transactions
GROUP BY MONTH(TransactionDate)
ORDER BY Month


--#Top 5 Best-Selling Products

SELECT TOP 5 p.ProductName, SUM(t.Quantity) AS Total_Sold
FROM Transactions t
JOIN Products p ON t.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY Total_Sold DESC


--#Top 5 Customers by Total Spend

SELECT TOP 5 c.CustomerName, SUM(t.TotalValue) AS Total_Spent
FROM Transactions t
JOIN Customers c ON t.CustomerID = c.CustomerID
GROUP BY c.CustomerName
ORDER BY Total_Spent DESC


--#Extracting year from date

Select SignupDate, LEFT(SignupDate,4) AS Year
FROM Customers
```

```sql
--#Average Order Value

SELECT AVG(TotalValue) AS Average_Order_Value
FROM Transactions


--#Revenue by Product Category

SELECT p.Category, SUM(t.TotalValue) AS Category_Revenue
FROM Transactions t
JOIN Products p ON t.ProductID = p.ProductID
GROUP BY p.Category
ORDER BY Category_Revenue DESC


--#Number of Transactions Per Customer

SELECT CustomerID, COUNT(TransactionID) AS Transaction_Count
FROM Transactions
GROUP BY CustomerID
ORDER BY Transaction_Count DESC


--#Most Popular Product Category

SELECT TOP 1 p.Category, COUNT(t.TransactionID) AS Sales_Count
FROM Transactions t
JOIN Products p ON t.ProductID = p.ProductID
GROUP BY p.Category
ORDER BY Sales_Count DESC


--#Categorize Customers Based on Total Purchases

SELECT CustomerID,
       SUM(TotalValue) AS Total_Spent,
       CASE
           WHEN SUM(TotalValue) > 1500 THEN 'VIP'
           WHEN SUM(TotalValue) BETWEEN 500 AND 1500 THEN 'Regular'
           ELSE 'New'
       END AS Customer_Category
FROM Transactions
GROUP BY CustomerID


--#Revenue Contribution by Region
```

```sql
SELECT c.Region, SUM(t.TotalValue) AS Total_Revenue
FROM Transactions t
JOIN Customers c ON t.CustomerID = c.CustomerID
GROUP BY c.Region
ORDER BY Total_Revenue DESC


--#Count of Returning Customers

SELECT CustomerID, COUNT(DISTINCT TransactionDate) AS Purchase_Days
FROM Transactions
GROUP BY CustomerID
HAVING COUNT(DISTINCT TransactionDate) > 1


--#Customers Who Have Not Made Any Purchase

SELECT c.CustomerID, c.CustomerName
FROM Customers c
LEFT JOIN Transactions t ON c.CustomerID = t.CustomerID
WHERE t.TransactionID IS NULL


--#Product Price Discrepancies in Transactions

SELECT t.ProductID, p.ProductName, p.Price AS Expected_Price, t.Price AS
  Actual_Price
FROM Transactions t
JOIN Products p ON t.ProductID = p.ProductID
WHERE p.Price <> t.Price;


--#First and Most Recent Purchase Date per Customer

SELECT CustomerID, MIN(TransactionDate) AS First_Purchase, MAX(TransactionDate) AS
   Last_Purchase
FROM Transactions
GROUP BY CustomerID


--#Highest Revenue Day

SELECT TOP 1 TransactionDate, SUM(TotalValue) AS Daily_Revenue
FROM Transactions
GROUP BY TransactionDate
ORDER BY Daily_Revenue DESC


--#Add and Update a New Column for Discount Eligibility
```

```sql
ALTER TABLE Transactions ADD Discount_Eligible VARCHAR(10)

UPDATE Transactions
SET Discount_Eligible =
    CASE
        WHEN TotalValue >= 1000 THEN 'Yes'
        ELSE 'No'
    END
```