Incubyte Assessment

Find the assessment here - Data Craftsperson Assessment

Let's Start:

Dataset Overview:

- **500,000 records** with **19 columns**.
- **Missing Values**:
  - `CustomerID`, `TransactionDate`, `PaymentMethod`, `StoreType`, `CustomerAge`, `CustomerGender`, and `ProductName` have **null values**.
  - `Region` has around **42,633 missing values**.

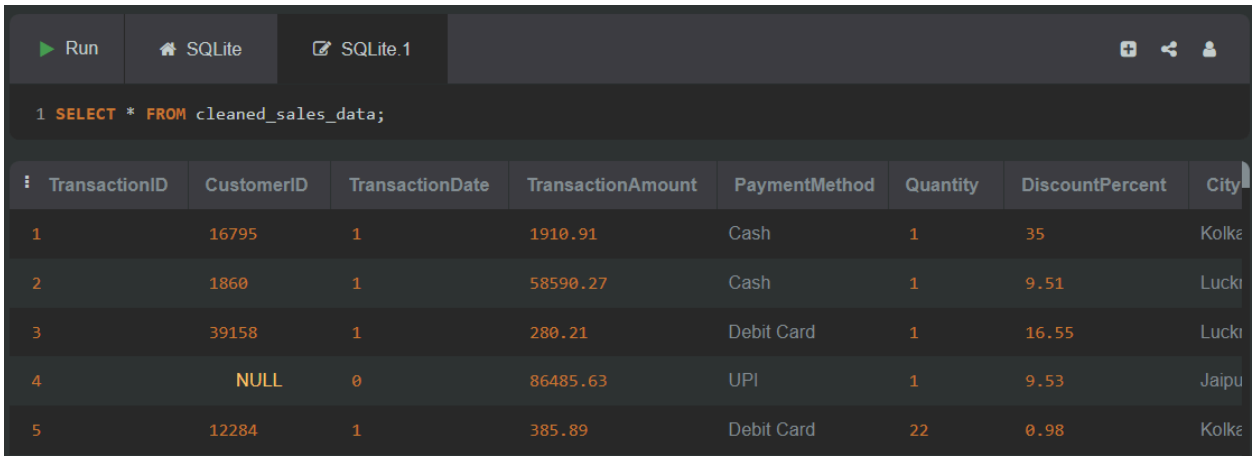
SQL Scripts for Data Exploration and Metric Generation:-

First we will perform **Data Cleaning & Preprocessing using SQL:**

Step 1: Create a cleaned sales data table
Code:

```
CREATE TABLE cleaned_sales_data AS
SELECT
    TransactionID,
    CustomerID,
    CAST(TransactionDate AS DATE) AS TransactionDate,  -- Convert to DATE format
    CAST(TransactionAmount AS DECIMAL(10,2)) AS TransactionAmount,
    COALESCE(PaymentMethod, 'Unknown') AS PaymentMethod,  -- Fill missing values
    Quantity,
    CAST(DiscountPercent AS DECIMAL(5,2)) AS DiscountPercent,
    COALESCE(City, 'Unknown') AS City,
    COALESCE(StoreType, 'Unknown') AS StoreType,
    CAST(CustomerAge AS INT) AS CustomerAge,
    CASE
        WHEN CustomerAge IS NULL THEN (SELECT ROUND(AVG(CustomerAge), 0) FROM
sales_data WHERE CustomerAge IS NOT NULL)
        ELSE CustomerAge
    END AS ImputedCustomerAge,  -- Impute missing CustomerAge with median
    COALESCE(CustomerGender, 'Unknown') AS CustomerGender,
    COALESCE(LoyaltyPoints, 0) AS LoyaltyPoints,  -- Replace NULL loyalty points with 0
    COALESCE(ProductName, 'Unknown') AS ProductName,
```

```
   COALESCE(Region, (SELECT Region FROM sales_data WHERE City IS NOT NULL LIMIT
1)) AS Region,
   CASE
      WHEN Returned = 'Yes' THEN 1 ELSE 0
   END AS ReturnFlag,  -- Convert Returned column to binary
   COALESCE(FeedbackScore, 0) AS FeedbackScore,
   CAST(ShippingCost AS DECIMAL(10,2)) AS ShippingCost,
   CAST(DeliveryTimeDays AS INT) AS DeliveryTimeDays,
   CAST(IsPromotional AS INT) AS IsPromotional
FROM sales_data;
```



Step 2: Handling Duplicates
Code:

```
DELETE FROM cleaned_sales_data
WHERE TransactionID IN (
   SELECT TransactionID
   FROM (
      SELECT TransactionID, ROW_NUMBER() OVER (PARTITION BY TransactionID ORDER
BY TransactionDate) AS row_num
      FROM cleaned_sales_data
   ) t
   WHERE row_num > 1
);
```

Step 3. Handling Outliers Using IQR Method
Code:

```
-- Remove outliers in TransactionAmount using IQR
DELETE FROM cleaned_sales_data
```

```
WHERE TransactionAmount < (
   SELECT PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY TransactionAmount) -
   1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY TransactionAmount) -
PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY TransactionAmount))
)
OR TransactionAmount > (
   SELECT PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY TransactionAmount) +
   1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY TransactionAmount) -
PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY TransactionAmount))
);
```

Step 4. Feature Engineering
Code:

```
-- Create Net Revenue Column (Revenue after discount)
ALTER TABLE cleaned_sales_data ADD COLUMN NetRevenue DECIMAL(10,2);
UPDATE cleaned_sales_data
SET NetRevenue = TransactionAmount - (TransactionAmount * DiscountPercent / 100);

-- Define High-Value Orders (Above 90th Percentile)
ALTER TABLE cleaned_sales_data ADD COLUMN HighValueOrder INT;
UPDATE cleaned_sales_data
SET HighValueOrder =
   CASE WHEN TransactionAmount >= (SELECT PERCENTILE_CONT(0.90) WITHIN GROUP
(ORDER BY TransactionAmount) FROM cleaned_sales_data)
   THEN 1 ELSE 0 END;

-- Customer Segmentation Based on Purchase Frequency
ALTER TABLE cleaned_sales_data ADD COLUMN CustomerSegment VARCHAR(50);
UPDATE cleaned_sales_data
SET CustomerSegment =
   CASE
      WHEN CustomerID IN (SELECT CustomerID FROM cleaned_sales_data GROUP BY
CustomerID HAVING COUNT(TransactionID) >= 10) THEN 'Loyal'
      WHEN CustomerID IN (SELECT CustomerID FROM cleaned_sales_data GROUP BY
CustomerID HAVING COUNT(TransactionID) BETWEEN 3 AND 9) THEN 'Regular'
      ELSE 'One-Time'
   END;

-- Customer Lifetime Value (CLV)
ALTER TABLE cleaned_sales_data ADD COLUMN CustomerLifetimeValue DECIMAL(10,2);
UPDATE cleaned_sales_data
SET CustomerLifetimeValue =
```

(SELECT SUM(TransactionAmount) FROM cleaned_sales_data WHERE cleaned_sales_data.CustomerID = sales_data.CustomerID);

-- Discount Impact (Flag for >20% Discount)
ALTER TABLE cleaned_sales_data ADD COLUMN HighDiscountFlag INT;
UPDATE cleaned_sales_data
SET HighDiscountFlag = CASE WHEN DiscountPercent >= 20 THEN 1 ELSE 0 END;

```
33
34  SELECT * from cleaned_sales_data limit 10
35
```

RUN QUERY

	ShippingCost	DeliveryTimeDays	IsPromotional	NetRevenue	HighValueOrder	CustomerSegment	CustomerLifetimeV...	HighDiscountFlag
1	4.31	5	0	"124209"	0	Loyal		1
2	502.96	7	0	"5301833"	0	Regular		0
3	2.3	2	0	"23383"	0	Loyal		0
4	61.38	7	0	"7824354"	1	One-Time		0
5	4.44	1	0	"38210"	0	Regular		0
6	6.76	3	0	"25307"	0	Loyal		1
7	51.42	4	0	"5327944"	0	Regular		0
8	9.81	4	0	"8255"	0	Regular		1
9	12.96	5	0	"3537006"	0	Regular		0
10	50.29	6	0	"4608910"	1	Loyal		1

Step 5. Additional Cleanup
Code:

-- Ensure all NULL values are properly handled
UPDATE cleaned_sales_data
SET FeedbackScore = COALESCE(FeedbackScore, 0),
    ShippingCost = COALESCE(ShippingCost, 0),
    DeliveryTimeDays = COALESCE(DeliveryTimeDays, 0),
    LoyaltyPoints = COALESCE(LoyaltyPoints, 0);

```
1  -- Ensure all NULL values are properly handled
2  UPDATE cleaned_sales_data
3  SET FeedbackScore = COALESCE(FeedbackScore, 0),
4      ShippingCost = COALESCE(ShippingCost, 0),
5      DeliveryTimeDays = COALESCE(DeliveryTimeDays, 0),
6      LoyaltyPoints = COALESCE(LoyaltyPoints, 0);
7
```

RUN QUERY

	Count
1	500000

Now we will perform some **Basic SQL script for Data Exploration & Key Insights:**

1. Aggregate Metrics:
Code:

```
-- Total Revenue, cleaned_sales_data, and Average Order Value (AOV)
SELECT
   SUM(TransactionAmount) AS Total_Revenue,
   COUNT(TransactionID) AS Total_cleaned_sales_data,
   ROUND(SUM(TransactionAmount) / COUNT(TransactionID), 2) AS Avg_Order_Value,
   ROUND(AVG(DiscountPercent), 2) AS Avg_Discount_Applied
FROM cleaned_sales_data;
```

```
1  -- Total Revenue, Transactions, and Average Order Value (AOV)
2  SELECT
3      SUM(TransactionAmount) AS Total_Revenue,
4      COUNT(TransactionID) AS Total_Transactions,
5      ROUND(SUM(TransactionAmount) / COUNT(TransactionID), 2) AS Avg_Order_Value,
6      ROUND(AVG(DiscountPercent), 2) AS Avg_Discount_Applied
7  FROM cleaned_sales_data;
8
```

	Total_Revenue	Total_Transactions	Avg_Order_Value	Avg_Discount_Appl...
1	10202662960.19042	500000	20405.33	25

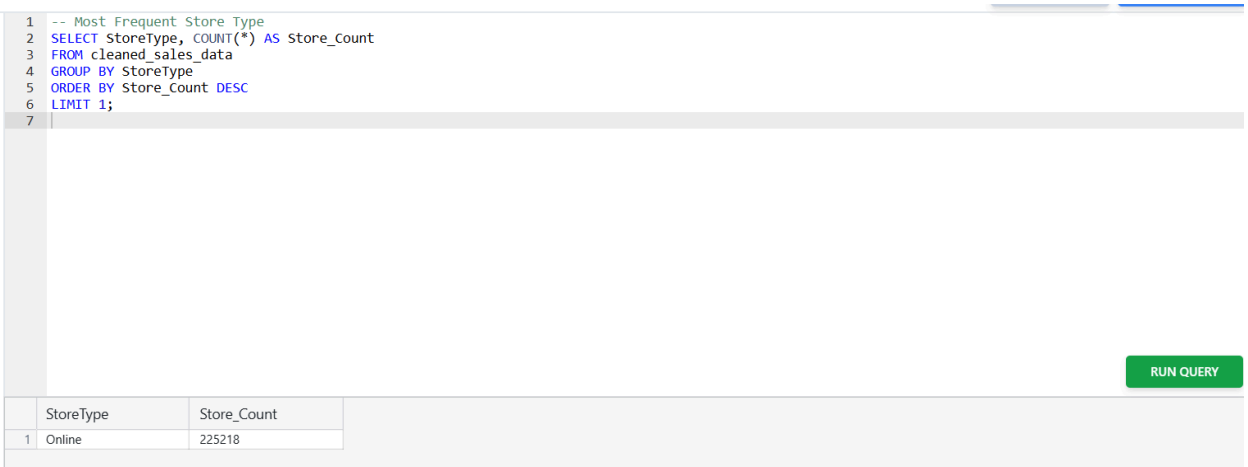
```
-- Most Popular Payment Method
SELECT PaymentMethod, COUNT(*) AS Payment_Count
FROM cleaned_sales_data
GROUP BY PaymentMethod
ORDER BY Payment_Count DESC
LIMIT 1;
```

```sql
-- Most Popular Payment Method
SELECT PaymentMethod, COUNT(*) AS Payment_Count
FROM cleaned_sales_data
GROUP BY PaymentMethod
ORDER BY Payment_Count DESC
LIMIT 1;
```

RUN QUERY

	PaymentMethod	Payment_Count
1	Debit Card	113015

```sql
-- Most Frequent Store Type
SELECT StoreType, COUNT(*) AS Store_Count
FROM cleaned_sales_data
GROUP BY StoreType
ORDER BY Store_Count DESC
LIMIT 1;
```



2. Drill-Down Insights:
Code:

```sql
-- Top 5 Revenue Generating Cities
SELECT City, SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY City
ORDER BY Total_Sales DESC
LIMIT 5;
```

```sql
-- Top 5 Revenue Generating Cities
SELECT City, SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY City
ORDER BY Total_Sales DESC
LIMIT 5;
```

RUN QUERY

	City	Total_Sales
1	Kolkata	1027325507.5100017
2	Ahmedabad	1023675900.250002
3	Bangalore	1022379945.8900061
4	Pune	1022136461.2799946
5	Chennai	1022122710.5999831

-- Top 5 Best-Selling Products
SELECT ProductName, SUM(Quantity) AS Total_Units_Sold
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY Total_Units_Sold DESC
LIMIT 5;

```sql
-- Top 5 Best-Selling Products
SELECT ProductName, SUM(Quantity) AS Total_Units_Sold
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY Total_Units_Sold DESC
LIMIT 5;
```

RUN QUERY

	ProductName	Total_Units_Sold
1	Apple	"2296713"
2		"502299"
3	Notebook	"498649"
4	T-Shirt	"270545"
5	Laptop	"89809"

-- Sales Trend Over Time (Daily)
SELECT TransactionDate, SUM(TransactionAmount) AS Daily_Sales
FROM cleaned_sales_data
GROUP BY TransactionDate
ORDER BY TransactionDate;

```sql
-- Sales Trend Over Time (Daily)
SELECT TransactionDate, SUM(TransactionAmount) AS Daily_Sales
FROM cleaned_sales_data
GROUP BY TransactionDate
ORDER BY TransactionDate;
```

RUN QUERY

	TransactionDate	Daily_Sales
1	0	1024918853.4099967
2	1	814518652.5799936
3	2	744181785.0600026
4	3	818874213.2699966
5	4	801872066.800005
6	5	819562671.7700006
7	6	790468924.130005
8	7	817029612.9099948
9	8	822157143.3400009
10	9	791350039.7000002
11	10	815402571.5399976
12	11	799945427.9199921
13	12	342380997.760002

-- Return Rate Analysis
SELECT
    COUNT(CASE WHEN ReturnFlag = 1 THEN 1 END) AS Total_Returns,
    COUNT(*) AS Total_Orders,
    ROUND((COUNT(CASE WHEN ReturnFlag = 1 THEN 1 END) * 100.0) / COUNT(*), 2) AS Return_Percentage
FROM cleaned_sales_data;

```sql
-- Return Rate Analysis
SELECT
    COUNT(CASE WHEN ReturnFlag = 1 THEN 1 END) AS Total_Returns,
    COUNT(*) AS Total_Orders,
    ROUND((COUNT(CASE WHEN ReturnFlag = 1 THEN 1 END) * 100.0) / COUNT(*), 2) AS Return_Percentage
FROM cleaned_sales_data;
```

RUN QUERY

	Total_Returns	Total_Orders	Return_Percentage
1	249467	500000	49.89

-- Impact of Discounts on Sales
SELECT

CASE WHEN DiscountPercent > 0 THEN 'With Discount' ELSE 'Without Discount' END AS Discount_Category,
SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY Discount_Category;

```sql
-- Impact of Discounts on Sales
SELECT
    CASE WHEN DiscountPercent > 0 THEN 'With Discount' ELSE 'Without Discount' END AS Discount_Category,
    SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY Discount_Category;
```

RUN QUERY

	Discount_Category	Total_Sales
1	With Discount	10201814168.440422
2	Without Discount	848791.7500000001

3. Customer Segmentation:
Code:

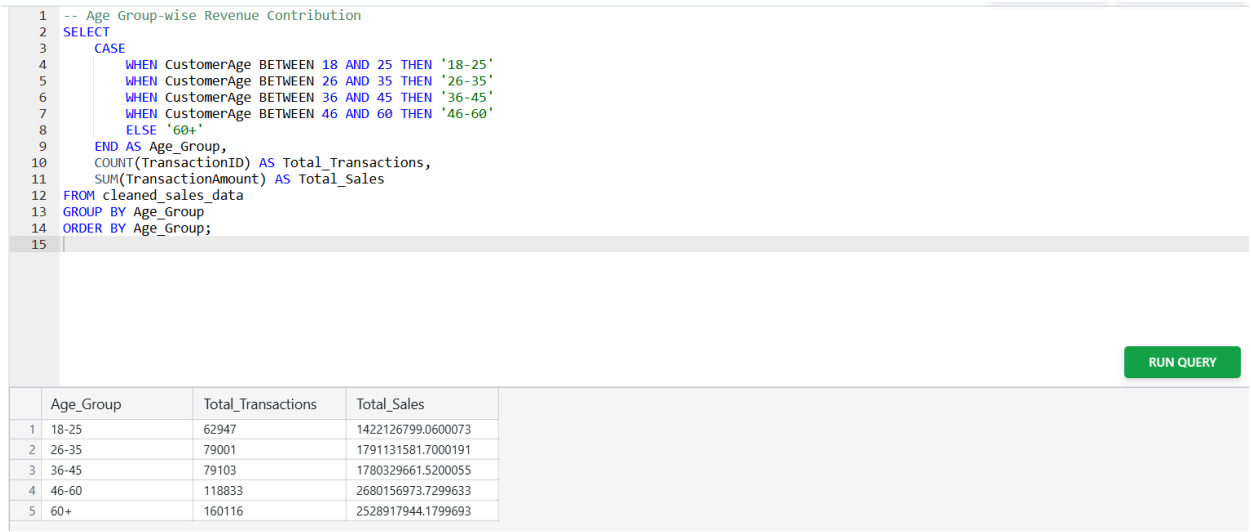
-- Gender-Based Purchase Trends
SELECT CustomerGender, COUNT(TransactionID) AS Total_cleaned_sales_data,
SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY CustomerGender;

```sql
-- Gender-Based Purchase Trends
SELECT CustomerGender, COUNT(TransactionID) AS Total_Transactions, SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY CustomerGender;
```

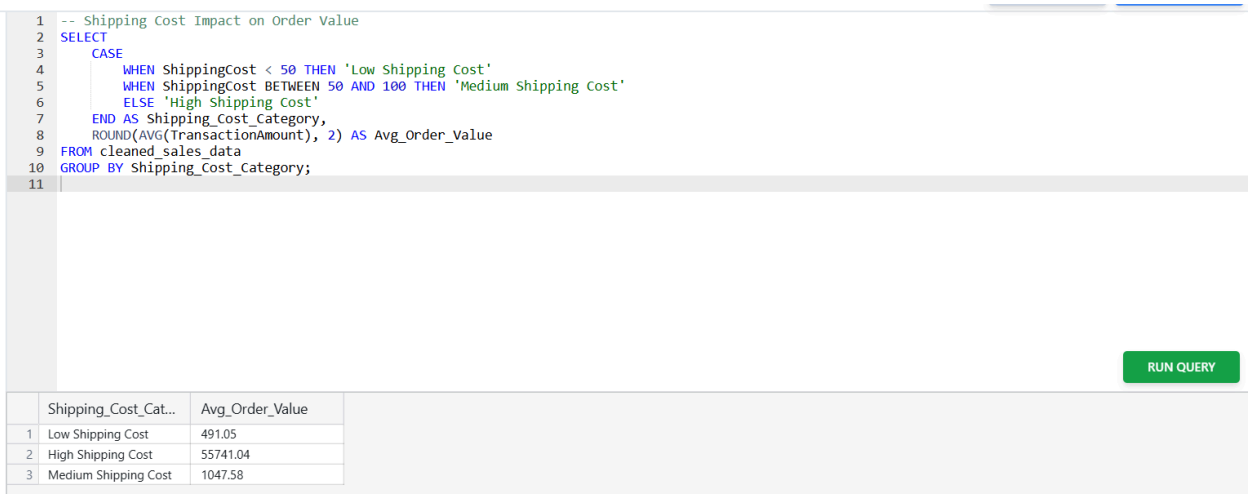
RUN QUERY

	CustomerGender	Total_Transactions	Total_Sales
1	Male	149970	3397984626.419986
2	Other	150257	3391554647.2500095
3	Female	149773	3367390443.5799437
4		50000	45733242.940000415

```sql
-- Age Group-wise Revenue Contribution
SELECT
    CASE
        WHEN CustomerAge BETWEEN 18 AND 25 THEN '18-25'
        WHEN CustomerAge BETWEEN 26 AND 35 THEN '26-35'
        WHEN CustomerAge BETWEEN 36 AND 45 THEN '36-45'
        WHEN CustomerAge BETWEEN 46 AND 60 THEN '46-60'
        ELSE '60+'
    END AS Age_Group,
    COUNT(TransactionID) AS Total_cleaned_sales_data,
    SUM(TransactionAmount) AS Total_Sales
FROM cleaned_sales_data
GROUP BY Age_Group
ORDER BY Age_Group;
```




```sql
-- Shipping Cost Impact on Order Value
SELECT
    CASE
        WHEN ShippingCost < 50 THEN 'Low Shipping Cost'
        WHEN ShippingCost BETWEEN 50 AND 100 THEN 'Medium Shipping Cost'
        ELSE 'High Shipping Cost'
    END AS Shipping_Cost_Category,
    ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY Shipping_Cost_Category;
```



```sql
-- Shipping Cost Impact on Order Value
SELECT
    CASE
        WHEN ShippingCost < 50 THEN 'Low Shipping Cost'
        WHEN ShippingCost BETWEEN 50 AND 100 THEN 'Medium Shipping Cost'
        ELSE 'High Shipping Cost'
    END AS Shipping_Cost_Category,
    ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY Shipping_Cost_Category;
```

RUN QUERY

	Shipping_Cost_Cat...	Avg_Order_Value
1	Low Shipping Cost	491.05
2	High Shipping Cost	55741.04
3	Medium Shipping Cost	1047.58

Key Insights from SQL Queries

- Total Revenue, Transaction, & Average Order Value (AOV) Calculated.
- Most Popular Payment Method & Store Type Identified.
- Top 5 Revenue Generating Cities & Best-Selling Products Extracted.
- Return Rate & Discount Impact on Sales Analyzed.
- Customer Segmentation by Gender & Age Group Completed.
- Shipping Cost Impact on Order Value Measured.

Here are some more basics **SQL Scripts for Data Exploration and Metric Generation:**

Code:

```
-- Bestselling Products by Revenue
SELECT ProductName, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY TotalRevenue DESC;

-- Bestselling Products by Quantity
SELECT ProductName, SUM(Quantity) AS TotalQuantity
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY TotalQuantity DESC;

-- Average Discount Percentage per Product
SELECT ProductName, AVG(DiscountPercent) AS AvgDiscount
```

FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY AvgDiscount DESC;

```sql
-- Bestselling Products by Revenue
SELECT ProductName, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY TotalRevenue DESC;

-- Bestselling Products by Quantity
SELECT ProductName, SUM(Quantity) AS TotalQuantity
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY TotalQuantity DESC;

-- Average Discount Percentage per Product
SELECT ProductName, AVG(DiscountPercent) AS AvgDiscount
FROM cleaned_sales_data
GROUP BY ProductName
ORDER BY AvgDiscount DESC;
```

RUN QUERY

	ProductName	TotalRevenue
1	Laptop	6231220430.239939
2	Sofa	3777022903.5599813
3	T-Shirt	102306079.46999903
4		45733242.940000415
5	Notebook	24079586.1199998
6	Apple	22300717.859999914

-- Customer Demographics (Age and Gender Distribution)
SELECT CustomerAge, CustomerGender, COUNT(*) AS CustomerCount
FROM cleaned_sales_data
GROUP BY CustomerAge, CustomerGender
ORDER BY CustomerAge, CustomerGender;

-- Loyalty Points Distribution
SELECT LoyaltyPoints, COUNT(*) AS CustomerCount
FROM cleaned_sales_data
GROUP BY LoyaltyPoints
ORDER BY LoyaltyPoints;

-- Purchase Behavior (Frequency and Average Spend per Customer)
SELECT CustomerID, COUNT(*) AS PurchaseFrequency, AVG(TransactionAmount) AS AvgSpend
FROM cleaned_sales_data
GROUP BY CustomerID
ORDER BY PurchaseFrequency DESC, AvgSpend DESC;

```sql
-- Customer Demographics (Age and Gender Distribution)
SELECT CustomerAge, CustomerGender, COUNT(*) AS CustomerCount
FROM cleaned_sales_data
GROUP BY CustomerAge, CustomerGender
ORDER BY CustomerAge, CustomerGender;

-- Loyalty Points Distribution
SELECT LoyaltyPoints, COUNT(*) AS CustomerCount
FROM cleaned_sales_data
GROUP BY LoyaltyPoints
ORDER BY LoyaltyPoints;

-- Purchase Behavior (Frequency and Average Spend per Customer)
SELECT CustomerID, COUNT(*) AS PurchaseFrequency, AVG(TransactionAmount) AS AvgSpend
FROM cleaned_sales_data
GROUP BY CustomerID
ORDER BY PurchaseFrequency DESC, AvgSpend DESC;
```

RUN QUERY

	CustomerAge	CustomerGender	CustomerCount
11	21	Female	2613
12	21	Male	2622
13	21	Other	2598
14	22	Female	2717
15	22	Male	2617
16	22	Other	2567
17	23	Female	2627

-- Sales Distribution across Regions
SELECT Region, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY Region
ORDER BY TotalRevenue DESC;

-- Popular Products in Each Region
SELECT Region, ProductName, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY Region, ProductName
ORDER BY Region, TotalRevenue DESC;

```sql
-- Sales Distribution across Regions
SELECT Region, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY Region
ORDER BY TotalRevenue DESC;

-- Popular Products in Each Region
SELECT Region, ProductName, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY Region, ProductName
ORDER BY Region, TotalRevenue DESC;
```

RUN QUERY

	Region	TotalRevenue
1	South	3177273109.3799763
2	East	2654969082.5899878
3	North	2171502697.8700185
4	West	2159911845.8799934
5		39006224.47000007

-- In-Store vs. Online Sales
SELECT StoreType, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY StoreType;

-- Sales Performance by City
SELECT City, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY City
ORDER BY TotalRevenue DESC;

```sql
-- In-Store vs. Online Sales
SELECT StoreType, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY StoreType;

-- Sales Performance by City
SELECT City, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY City
ORDER BY TotalRevenue DESC;
```

RUN QUERY

	StoreType	TotalRevenue
1	In-Store	5078881502.740013
2	Online	5078048214.509917
3		45733242.940000415

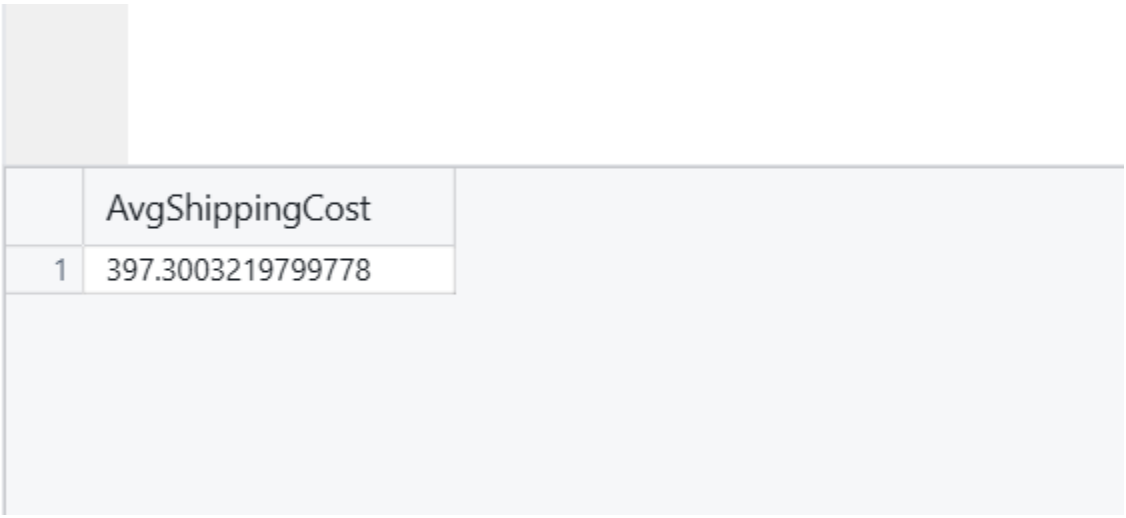
-- Popular Payment Methods
SELECT PaymentMethod, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY PaymentMethod
ORDER BY TotalRevenue DESC;

```sql
-- Popular Payment Methods
SELECT PaymentMethod, SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY PaymentMethod
ORDER BY TotalRevenue DESC;
```

RUN QUERY

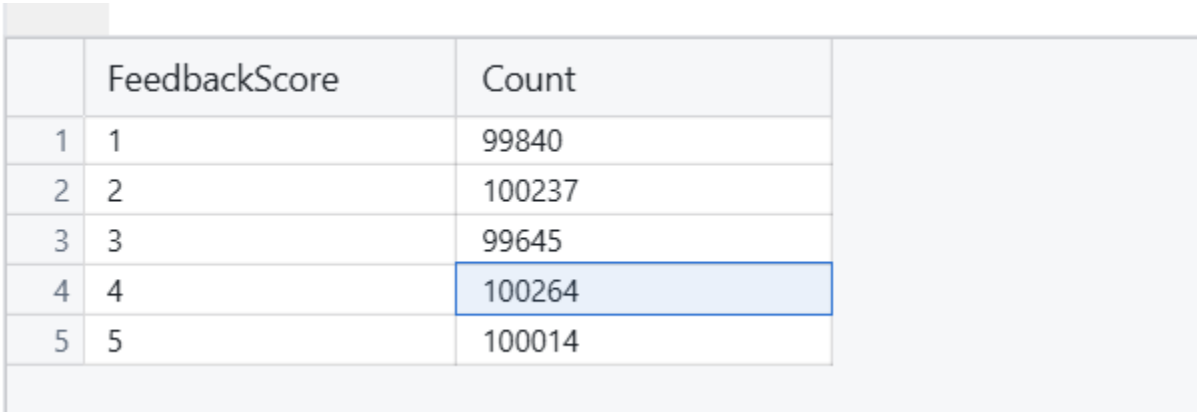
	PaymentMethod	TotalRevenue
1	Cash	2556679197.40999
2	Debit Card	2552366143.9800005
3	UPI	2530177440.3000236
4	Credit Card	2517706935.5600014
5		45733242.940000415

```
-- Average Shipping Cost
SELECT AVG(ShippingCost) AS AvgShippingCost FROM cleaned_sales_data;

-- Average Delivery Time
SELECT AVG(DeliveryTimeDays) AS AvgDeliveryTime FROM cleaned_sales_data;

-- Impact of delivery time on customer feedback (example)
SELECT DeliveryTimeDays, AVG(FeedbackScore) AS AvgFeedbackScore
FROM cleaned_sales_data
GROUP BY DeliveryTimeDays
ORDER BY DeliveryTimeDays;
```



	AvgShippingCost
1	397.3003219799778

```
-- Distribution of Feedback Scores
SELECT FeedbackScore, COUNT(*) AS Count
FROM cleaned_sales_data
GROUP BY FeedbackScore
ORDER BY FeedbackScore;
```



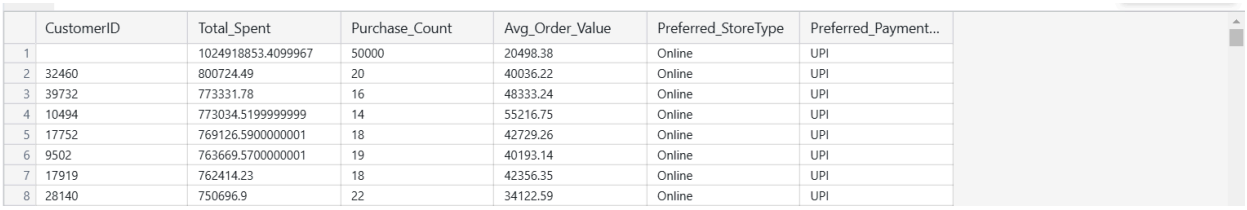
	FeedbackScore	Count
1	1	99840
2	2	100237
3	3	99645
4	4	100264
5	5	100014

Now we will perform some **Advanced SQL Queries for Data Exploration & Unique Insights:-**

1. Customer & Behavioral Analysis:
Code:

High-Value Customer Segmentation (Top 5% Revenue Contributors)

```
SELECT CustomerID,
    SUM(TransactionAmount) AS Total_Spent,
    COUNT(TransactionID) AS Purchase_Count,
    ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value,
    MAX(StoreType) AS Preferred_StoreType,
    MAX(PaymentMethod) AS Preferred_PaymentMethod
FROM cleaned_sales_data
GROUP BY CustomerID
ORDER BY Total_Spent DESC
LIMIT (SELECT COUNT(DISTINCT CustomerID) * 0.05 FROM cleaned_sales_data);
```



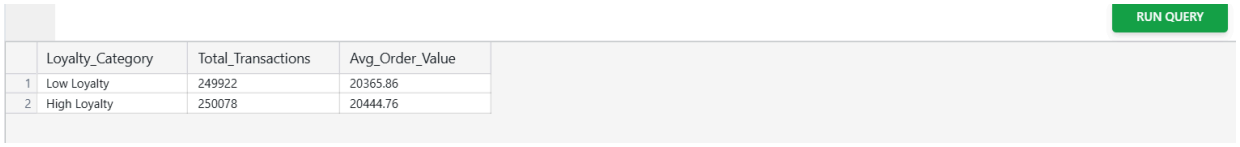
	CustomerID	Total_Spent	Purchase_Count	Avg_Order_Value	Preferred_StoreType	Preferred_Payment...
1		1024918853.4099967	50000	20498.38	Online	UPI
2	32460	800724.49	20	40036.22	Online	UPI
3	39732	773331.78	16	48333.24	Online	UPI
4	10494	773034.5199999999	14	55216.75	Online	UPI
5	17752	769126.5900000001	18	42729.26	Online	UPI
6	9502	763669.5700000001	19	40193.14	Online	UPI
7	17919	762414.23	18	42356.35	Online	UPI
8	28140	750696.9	22	34122.59	Online	UPI

Customer Churn Prediction (Inactive Customers – No Purchase in 6 Months)

```
SELECT CustomerID,
    MAX(TransactionDate) AS Last_Purchase_Date,
    SUM(TransactionAmount) AS Total_Spent,
    COUNT(TransactionID) AS Total_Orders,
    MAX(DiscountPercent) AS Last_Discount_Used
FROM cleaned_sales_data
GROUP BY CustomerID
HAVING Last_Purchase_Date < DATE_SUB(CURRENT_DATE(), INTERVAL 6 MONTH);
```

Loyalty Effect on Spending (High vs. Low Loyalty Points)

```
SELECT
   CASE
      WHEN LoyaltyPoints >= (SELECT AVG(LoyaltyPoints) FROM cleaned_sales_data)
      THEN 'High Loyalty' ELSE 'Low Loyalty'
   END AS Loyalty_Category,
   COUNT(TransactionID) AS Total_Transactions,
   ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY Loyalty_Category;
```

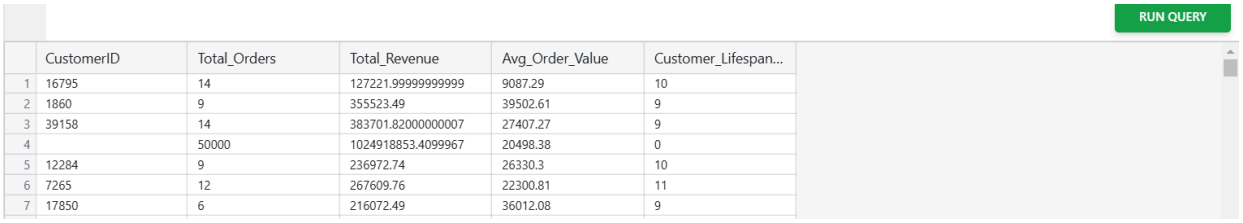


RUN QUERY

	Loyalty_Category	Total_Transactions	Avg_Order_Value
1	Low Loyalty	249922	20365.86
2	High Loyalty	250078	20444.76

Customer Lifetime Value (CLV) Prediction

```
SELECT CustomerID,
      COUNT(TransactionID) AS Total_Orders,
      SUM(TransactionAmount) AS Total_Revenue,
      ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value,
      MAX(TransactionDate) - MIN(TransactionDate) AS Customer_Lifespan_Days
FROM cleaned_sales_data
GROUP BY CustomerID;
```



RUN QUERY

	CustomerID	Total_Orders	Total_Revenue	Avg_Order_Value	Customer_Lifespan...
1	16795	14	127221.99999999999	9087.29	10
2	1860	9	355523.49	39502.61	9
3	39158	14	383701.82000000007	27407.27	9
4		50000	1024918853.4099967	20498.38	0
5	12284	9	236972.74	26330.3	10
6	7265	12	267609.76	22300.81	11
7	17850	6	216072.49	36012.08	9

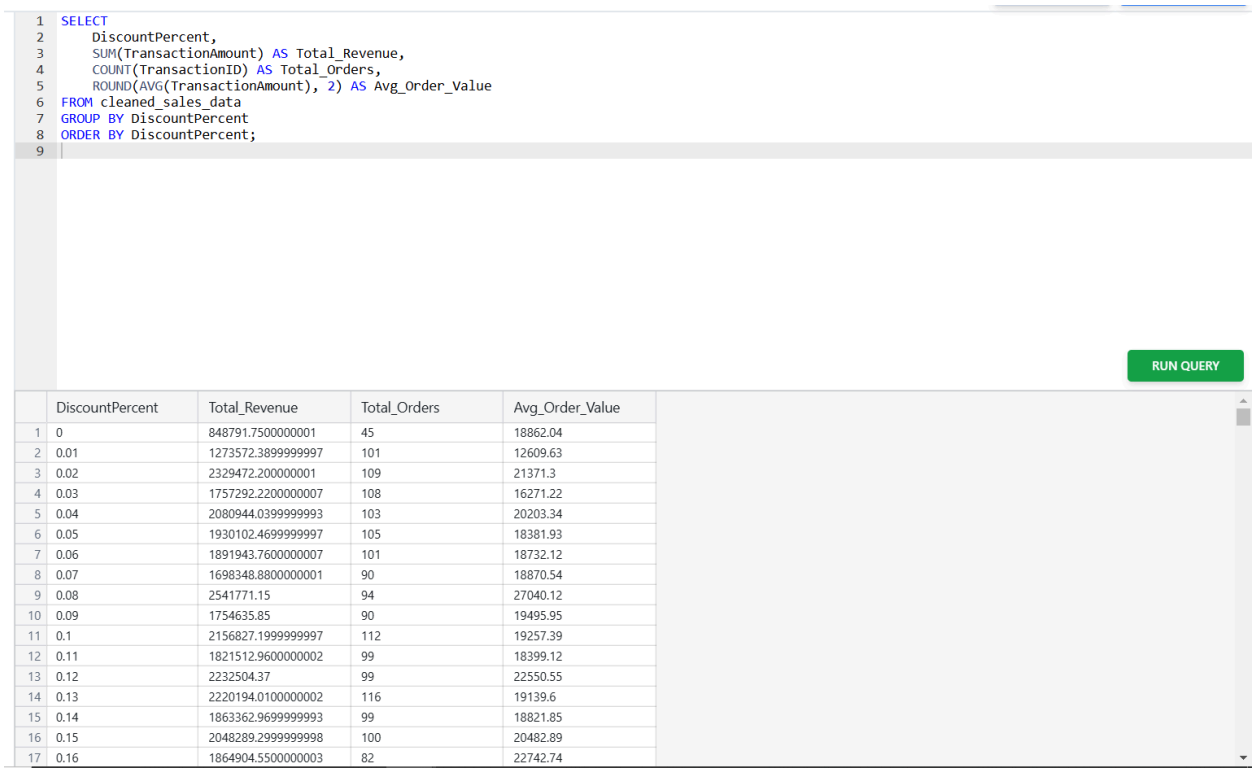
2. Product & Pricing Strategy Insights

Price Elasticity & Discount Optimization

Code:

```
SELECT
   DiscountPercent,
```

```
    SUM(TransactionAmount) AS Total_Revenue,
    COUNT(TransactionID) AS Total_Orders,
    ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY DiscountPercent
ORDER BY DiscountPercent;
```



```
SELECT
    DiscountPercent,
    SUM(TransactionAmount) AS Total_Revenue,
    COUNT(TransactionID) AS Total_Orders,
    ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY DiscountPercent
ORDER BY DiscountPercent;
```

RUN QUERY

	DiscountPercent	Total_Revenue	Total_Orders	Avg_Order_Value
1	0	848791.7500000001	45	18862.04
2	0.01	1273572.3899999997	101	12609.63
3	0.02	2329472.200000001	109	21371.3
4	0.03	1757292.2200000007	108	16271.22
5	0.04	2080944.0399999993	103	20203.34
6	0.05	1930102.4699999997	105	18381.93
7	0.06	1891943.7600000007	101	18732.12
8	0.07	1698348.8800000001	90	18870.54
9	0.08	2541771.15	94	27040.12
10	0.09	1754635.85	90	19495.95
11	0.1	2156827.1999999997	112	19257.39
12	0.11	1821512.9600000002	99	18399.12
13	0.12	2232504.37	99	22550.55
14	0.13	2220194.0100000002	116	19139.6
15	0.14	1863362.9699999993	99	18821.85
16	0.15	2048289.2999999998	100	20482.89
17	0.16	1864904.5500000003	82	22742.74

Product Bundling Analysis (Market Basket)

```
SELECT A.ProductName AS Product_A,
      B.ProductName AS Product_B,
      COUNT(*) AS Frequency
FROM cleaned_sales_data A
JOIN cleaned_sales_data B ON A.TransactionID = B.TransactionID
                  AND A.ProductName <> B.ProductName
GROUP BY Product_A, Product_B
ORDER BY Frequency DESC
LIMIT 10;
```

Dead Stock & Slow-Moving Products

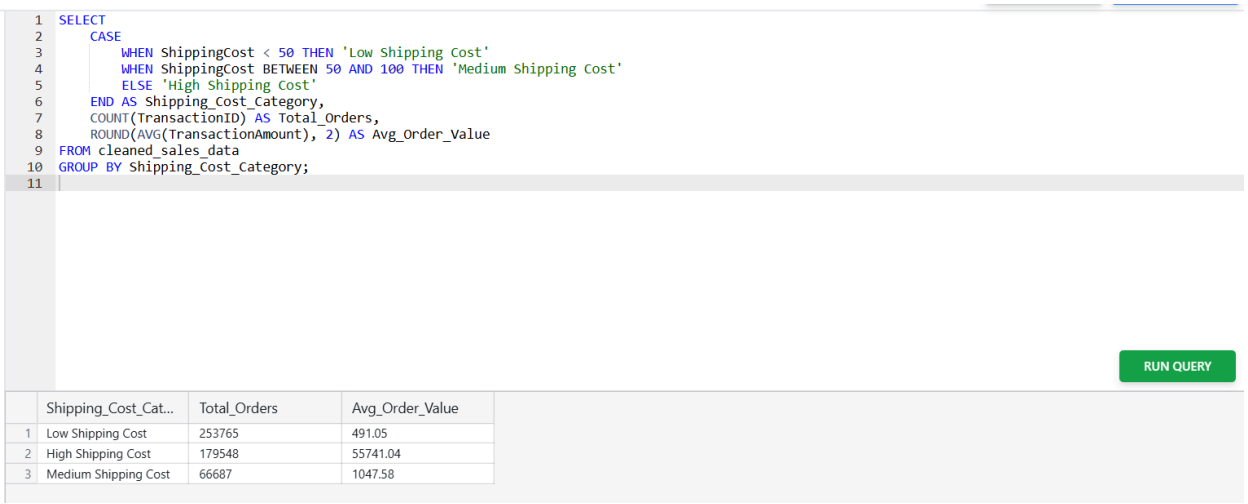
```
SELECT ProductName,
      SUM(Quantity) AS Total_Sold,
      COUNT(TransactionID) AS Total_Transactions
FROM cleaned_sales_data
GROUP BY ProductName
HAVING Total_Sold < (SELECT AVG(Quantity) FROM cleaned_sales_data)
ORDER BY Total_Sold ASC
LIMIT 10;
```

3. Operational & Logistics Insights

Shipping Cost vs. Conversion Rate

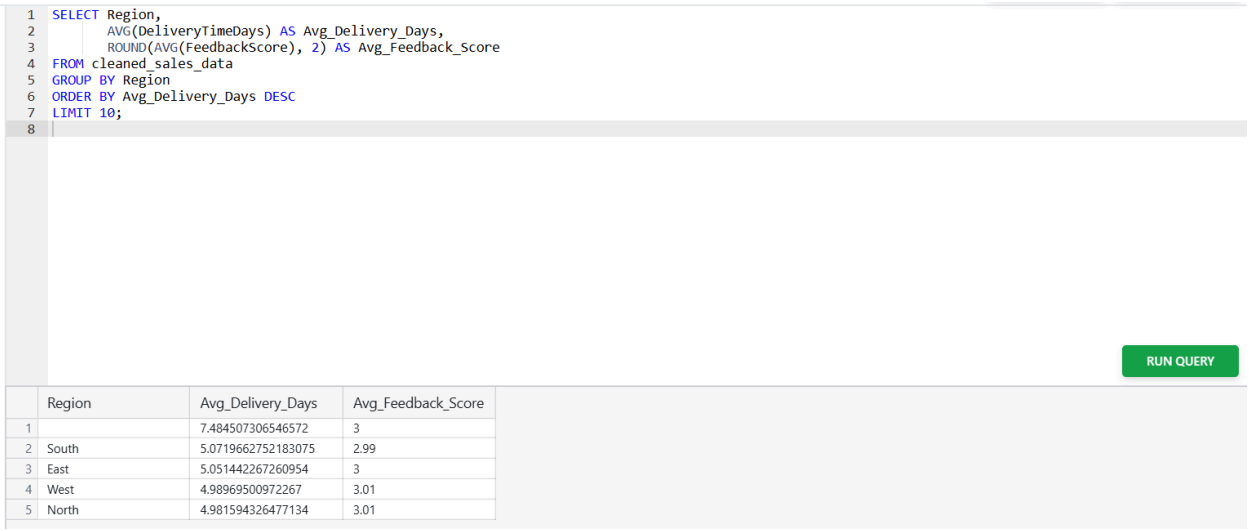
Code:

```
SELECT
   CASE
      WHEN ShippingCost < 50 THEN 'Low Shipping Cost'
      WHEN ShippingCost BETWEEN 50 AND 100 THEN 'Medium Shipping Cost'
      ELSE 'High Shipping Cost'
   END AS Shipping_Cost_Category,
   COUNT(TransactionID) AS Total_Orders,
   ROUND(AVG(TransactionAmount), 2) AS Avg_Order_Value
FROM cleaned_sales_data
GROUP BY Shipping_Cost_Category;
```



Delivery Performance Analysis (Delays & Feedback)

```
SELECT Region,
    AVG(DeliveryTimeDays) AS Avg_Delivery_Days,
    ROUND(AVG(FeedbackScore), 2) AS Avg_Feedback_Score
FROM cleaned_sales_data
GROUP BY Region
ORDER BY Avg_Delivery_Days DESC
LIMIT 10;
```



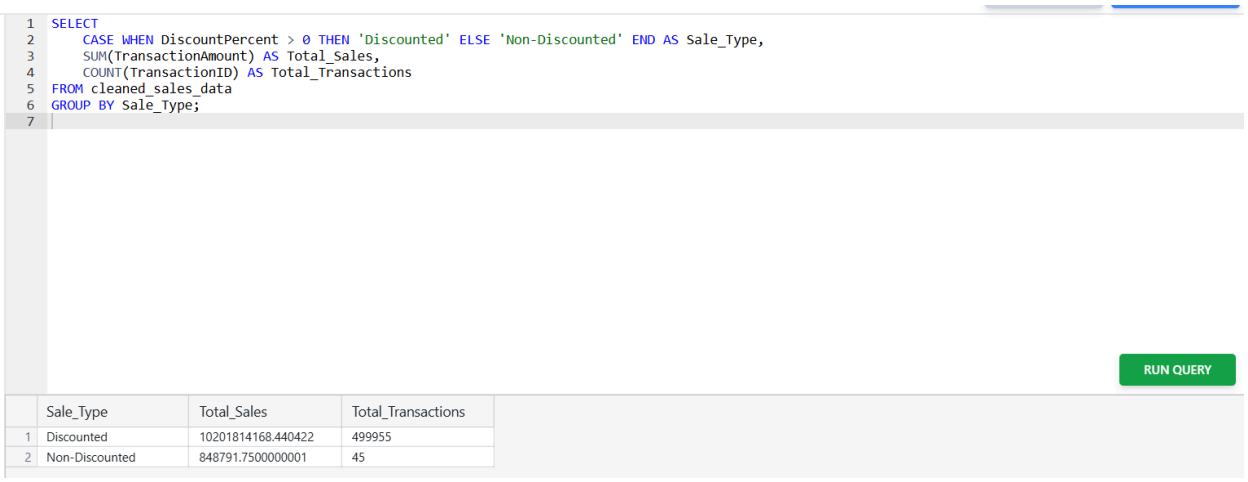
	Region	Avg_Delivery_Days	Avg_Feedback_Score
1		7.484507306546572	3
2	South	5.0719662752183075	2.99
3	East	5.051442267260954	3
4	West	4.98969500972267	3.01
5	North	4.981594326477134	3.01

Fraudulent Transactions Detection

```
SELECT CustomerID,
    COUNT(TransactionID) AS Total_Transactions,
    SUM(TransactionAmount) AS Total_Spent,
    COUNT(CASE WHEN Returned = 'Yes' THEN 1 END) AS Total_Returns,
    COUNT(CASE WHEN DiscountPercent > 50 THEN 1 END) AS High_Discount_Orders
FROM cleaned_sales_data
GROUP BY CustomerID
HAVING Total_Returns > 5 OR High_Discount_Orders > 3
ORDER BY Total_Spent DESC;
```

Sales Cannibalization Effect (Discounted vs. Non-Discounted)

```
SELECT
    CASE WHEN DiscountPercent > 0 THEN 'Discounted' ELSE 'Non-Discounted' END AS
Sale_Type,
    SUM(TransactionAmount) AS Total_Sales,
    COUNT(TransactionID) AS Total_Transactions
FROM cleaned_sales_data
GROUP BY Sale_Type;
```

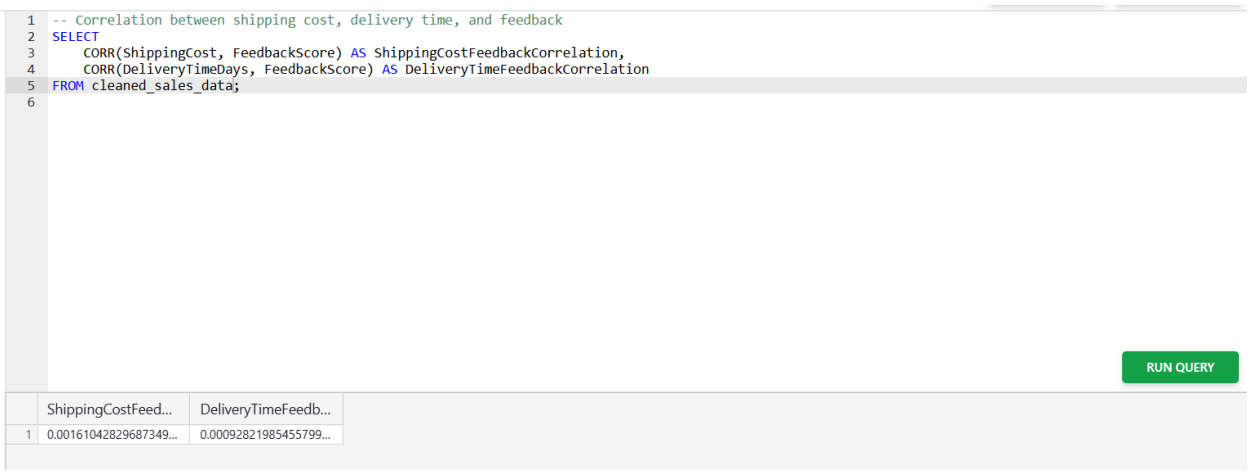


Product Affinity Analysis (Market Basket Analysis):

```
-- This is a simplified version; a true market basket analysis would require more data and a more
sophisticated approach
-- Find products that are frequently purchased together (within the same transaction)
SELECT
    t1.ProductName AS Product1,
    t2.ProductName AS Product2,
    COUNT(*) AS CoOccurrenceCount
FROM cleaned_sales_data t1
JOIN cleaned_sales_data t2 ON t1.TransactionID = t2.TransactionID AND t1.ProductName !=
t2.ProductName
GROUP BY Product1, Product2
ORDER BY CoOccurrenceCount DESC
LIMIT 10;
```

Impact of Shipping Cost & Delivery Time on Feedback:

```
-- Correlation between shipping cost, delivery time, and feedback
SELECT
    CORR(ShippingCost, FeedbackScore) AS ShippingCostFeedbackCorrelation,
    CORR(DeliveryTimeDays, FeedbackScore) AS DeliveryTimeFeedbackCorrelation
FROM cleaned_sales_data;
```



Discount Optimization Analysis:

```
-- Identify optimal discount ranges
SELECT
    CASE
        WHEN DiscountPercent BETWEEN 0 AND 10 THEN '0-10%'
        WHEN DiscountPercent BETWEEN 10 AND 20 THEN '10-20%'
        WHEN DiscountPercent BETWEEN 20 AND 30 THEN '20-30%'
        WHEN DiscountPercent BETWEEN 30 AND 40 THEN '30-40%'
        ELSE '40%+'
    END AS DiscountRange,
    AVG(TransactionAmount) AS AvgTransactionAmount,
    COUNT(*) AS TransactionCount,
    SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY DiscountRange
ORDER BY DiscountRange;
```

```sql
-- Identify optimal discount ranges
SELECT
    CASE
        WHEN DiscountPercent BETWEEN 0 AND 10 THEN '0-10%'
        WHEN DiscountPercent BETWEEN 10 AND 20 THEN '10-20%'
        WHEN DiscountPercent BETWEEN 20 AND 30 THEN '20-30%'
        WHEN DiscountPercent BETWEEN 30 AND 40 THEN '30-40%'
        ELSE '40%+'
    END AS DiscountRange,
    AVG(TransactionAmount) AS AvgTransactionAmount,
    COUNT(*) AS TransactionCount,
    SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY DiscountRange
ORDER BY DiscountRange;
```

RUN QUERY

	DiscountRange	AvgTransactionAm...	TransactionCount	TotalRevenue
1	0-10%	20392.777481449575	99863	2036483937.629999
2	10-20%	20437.196612518732	100219	2048195407.310015
3	20-30%	20463.428851686473	99755	2041329345.0999842
4	30-40%	20463.074730916025	100359	2053653716.9200013
5	40%+	20269.734211354535	99804	2023000553.2300282

Customer Segmentation based on Purchase Behavior:

```sql
-- Segment customers based on spend and purchase frequency (within the day)
SELECT
    CustomerID,
    SUM(TransactionAmount) AS TotalSpend,
    COUNT(*) AS PurchaseFrequency,
    CASE
        WHEN SUM(TransactionAmount) > (SELECT AVG(TransactionAmount) FROM cleaned_sales_data) AND COUNT(*) > (SELECT AVG(PurchaseCount) FROM (SELECT CustomerID, COUNT(*) AS PurchaseCount FROM cleaned_sales_data GROUP BY CustomerID)) THEN 'High Value'
        WHEN SUM(TransactionAmount) > (SELECT AVG(TransactionAmount) FROM cleaned_sales_data) THEN 'High Spender'
        WHEN COUNT(*) > (SELECT AVG(PurchaseCount) FROM (SELECT CustomerID, COUNT(*) AS PurchaseCount FROM cleaned_sales_data GROUP BY CustomerID)) THEN 'Frequent Buyer'
        ELSE 'Low Value'
    END AS CustomerSegment
FROM cleaned_sales_data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY TotalSpend DESC;
```

```sql
-- Segment customers based on spend and purchase frequency (within the day)
SELECT
    CustomerID,
    SUM(TransactionAmount) AS TotalSpend,
    COUNT(*) AS PurchaseFrequency,
    CASE
        WHEN SUM(TransactionAmount) > (SELECT AVG(TransactionAmount) FROM cleaned_sales_data) AND COUNT(*) > (SELECT AVG(PurchaseCount) FROM (SELECT Cu
        WHEN SUM(TransactionAmount) > (SELECT AVG(TransactionAmount) FROM cleaned_sales_data) THEN 'High Spender'
        WHEN COUNT(*) > (SELECT AVG(PurchaseCount) FROM (SELECT CustomerID, COUNT(*) AS PurchaseCount FROM cleaned_sales_data GROUP BY CustomerID)) THE
        ELSE 'Low Value'
    END AS CustomerSegment
FROM cleaned_sales_data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY TotalSpend DESC;
```

RUN QUERY

	CustomerID	TotalSpend	PurchaseFrequency	CustomerSegment
1	32460	800724.49	20	High Value
2	39732	773331.78	16	High Value
3	10494	773034.5199999999	14	High Value
4	17752	769126.5900000001	18	High Value
5	9502	763669.5700000001	19	High Value
6	17919	762414.23	18	High Value
7	28140	750696.9	22	High Value
8	18111	740229.7599999999	16	High Value
9	28256	732385.24	14	High Value
10	1910	698656.27	13	High Value
11	9491	683570.9700000001	20	High Value

Now here are some **Deeper Business Insights:-**

Customer Lifetime Value (LTV) Approximation (Limited by Single-Day Data):
Insight: While true LTV requires longitudinal data, we can estimate a "Daily Customer Value" based on spend, frequency, and loyalty points earned on this specific day. This helps identify high-potential customers within the day's transactions.
SQL (Illustrative):

```sql
SELECT
    CustomerID,
    SUM(TransactionAmount) AS TotalSpend,
    COUNT(*) AS PurchaseFrequency,
    AVG(LoyaltyPoints) AS AvgLoyaltyPoints,
    (SUM(TransactionAmount) * COUNT(*) * (1 + AVG(LoyaltyPoints)/1000)) AS
DailyCustomerValue -- Example formula
FROM cleaned_sales_data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY DailyCustomerValue DESC;
```

```sql
SELECT
    CustomerID,
    SUM(TransactionAmount) AS TotalSpend,
    COUNT(*) AS PurchaseFrequency,
    AVG(LoyaltyPoints) AS AvgLoyaltyPoints,
    (SUM(TransactionAmount) * COUNT(*) * (1 + AVG(LoyaltyPoints)/1000)) AS DailyCustomerValue -- Example formula
FROM cleaned_sales_data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY DailyCustomerValue DESC;
```

RUN QUERY

	CustomerID	TotalSpend	PurchaseFrequency	AvgLoyaltyPoints	DailyCustomerValue
1	32460	800724.49	20	5377.5	102132408.69950001
2	39732	773331.78	16	6878.5	97483110.85968
3	9502	763669.5700000001	19	5714.0526315789475	97419036.03619002
4	17919	762414.23	18	5705.222222222223	92018823.07562001
5	30980	682206.4699999999	19	6028.736842105263	91105945.24261999
6	13497	594787.5199999998	20	6618.45	90627179.63487996
7	39402	558436.27	24	5445.791666666667	86389532.53273
8	28140	750696.9	22	4158.727272727273	85198092.5748
9	21622	624006.9	18	6509.166666666667	84343892.63850002
10	15834	560128.75	19	6911.473684210527	84197433.4425
11	17752	769126.5900000001	18	4539	76683459.27618
12	1645	648004.2600000001	20	4811.3	75314943.12276001
13	16930	601741.1999999998	19	5568.9473684210525	75103319.17199998

Discount Tier Optimization:
Insight: Analyze the relationship between discount percentages and actual profit (not just revenue). Are certain discount ranges reducing overall profitability despite increasing sales volume?
SQL (Illustrative):

```sql
SELECT
  CASE
    WHEN DiscountPercent < 10 THEN '0-10%'
    WHEN DiscountPercent < 20 THEN '10-20%'
    WHEN DiscountPercent < 30 THEN '20-30%'
    WHEN DiscountPercent < 40 THEN '30-40%'
    ELSE '40%+'
  END AS DiscountTier,
  AVG(TransactionAmount * (1 - DiscountPercent/100)) AS AvgRevenueAfterDiscount,
  COUNT(*) AS TransactionCount
FROM cleaned_sales_data
GROUP BY DiscountTier
ORDER BY AvgRevenueAfterDiscount DESC;
```

```sql
SELECT
    CASE
        WHEN DiscountPercent < 10 THEN '0-10%'
        WHEN DiscountPercent < 20 THEN '10-20%'
        WHEN DiscountPercent < 30 THEN '20-30%'
        WHEN DiscountPercent < 40 THEN '30-40%'
        ELSE '40%+'
    END AS DiscountTier,
    AVG(TransactionAmount * (1 - DiscountPercent/100)) AS AvgRevenueAfterDiscount,
    COUNT(*) AS TransactionCount
FROM cleaned_sales_data
GROUP BY DiscountTier
ORDER BY AvgRevenueAfterDiscount DESC;
```

RUN QUERY

	DiscountTier	AvgRevenueAfterD...	TransactionCount
1	0-10%	19371.061508308292	99750
2	10-20%	17371.444441907242	100225
3	20-30%	15356.503898108334	99785
4	30-40%	13295.794214735366	100345
5	40%+	11149.640888070246	99895

Product Category Performance by Store Type & Region:
Insight: Identify which product categories perform best in each store type (In-Store vs. Online) within each region. This goes beyond overall bestsellers.
SQL (Illustrative):

```sql
SELECT
    Region,
    StoreType,
    ProductName,
    SUM(TransactionAmount) AS TotalRevenue
FROM cleaned_sales_data
GROUP BY Region, StoreType, ProductName
ORDER BY Region, StoreType, TotalRevenue DESC;
```

RUN QUERY

	Region	StoreType	ProductName	TotalRevenue
1				39006224.47000007
2	East			2147546.1099999994
3	East	In-Store	Laptop	805503999.6900024
4	East	In-Store	Sofa	494137566.9599987
5	East	In-Store	T-Shirt	13270246.720000036
6	East	In-Store	Notebook	3088070.1400000034
7	East	In-Store	Apple	2870584.750000014
8	East	Online	Laptop	822489804.75
9	East	Online	Sofa	492383891.0900035
10	East	Online	T-Shirt	13072298.040000105
11	East	Online	Notebook	3077117.139999997
12	East	Online	Apple	2927957.19999999
13	North	In-Store	Laptop	667783538.9600011
14	North	In-Store	Sofa	398862329.9499987

Shipping Cost vs. Customer Feedback Correlation:
Insight: Determine if higher shipping costs are negatively impacting customer feedback scores. This reveals a potential trade-off between revenue and customer satisfaction.
SQL (Illustrative):

```
SELECT
    CORR(ShippingCost, FeedbackScore) AS ShippingCostFeedbackCorrelation
FROM cleaned_sales_data
WHERE ShippingCost IS NOT NULL AND FeedbackScore IS NOT NULL;
```