Project Report: Analyzing Customer Behavior Using SQL Queries

1. Introduction:

This report details the analysis of customer behavior using SQL queries on a provided dataset. Through the utilization of SQL, we explore various aspects of customer interactions and purchasing patterns to uncover valuable insights.

2. Dataset Overview:

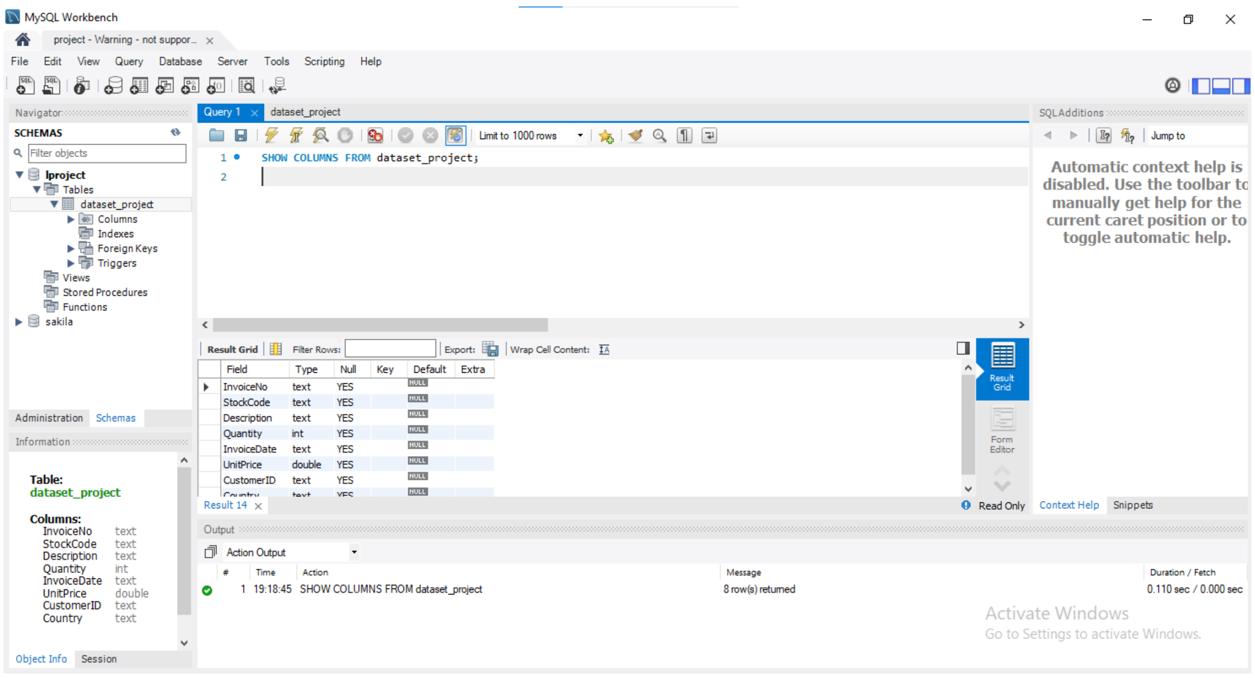
The dataset contains transaction records with fields such as InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country.

3. Beginner Queries:

3.1 Define Meta Data in MySQL Workbench

- **CODE**

```
SHOW COLUMNS FROM dataset_project;
```



```
SHOW COLUMNS FROM dataset_project;
```

Field	Type	Null	Key	Default	Extra
InvoiceNo	text	YES		NULL	
StockCode	text	YES		NULL	
Description	text	YES		NULL	
Quantity	int	YES		NULL	
InvoiceDate	text	YES		NULL	
UnitPrice	double	YES		NULL	
CustomerID	text	YES		NULL	
Country	text	YES		NULL	

Table: dataset_project

Columns:

InvoiceNo	text
StockCode	text
Description	text
Quantity	int
InvoiceDate	text
UnitPrice	double
CustomerID	text
Country	text

```
#   Time      Action                               Message            Duration / Fetch
1   19:18:45  SHOW COLUMNS FROM dataset_project    8 row(s) returned  0.110 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

- **Explanation:**

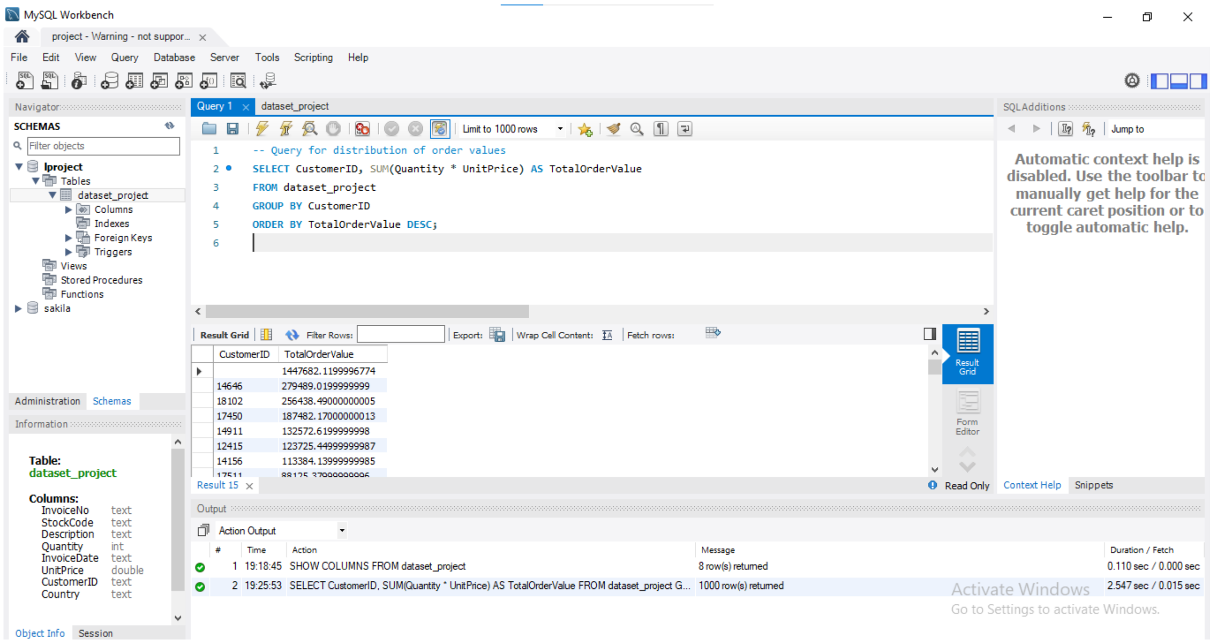
The query above displays the metadata of the dataset, listing the columns, their data types, and any constraints.

3.2 Distribution of Order Values Across Customers

- **Code**:

```
SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue
FROM dataset_project
GROUP BY CustomerID
ORDER BY TotalOrderValue DESC;
```

- **Output Screenshot:**



```
-- Query for distribution of order values
SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue
FROM dataset_project
GROUP BY CustomerID
ORDER BY TotalOrderValue DESC;
```

CustomerID	TotalOrderValue
	1447682.1199996774
14646	279489.0199999999
18102	256438.49000000005
17450	187482.17000000013
14911	132572.6199999998
12415	123725.44999999987
14156	113384.13999999985
17511	88125.37999999996

```
#  Time      Action                                                                        Message               Duration / Fetch
1  19:18:45  SHOW COLUMNS FROM dataset_project                                             8 row(s) returned     0.110 sec / 0.000 sec
2  19:25:53  SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM dataset_project G...  1000 row(s) returned  2.547 sec / 0.015 sec
```

Table: dataset_project

Columns:

InvoiceNo	text
StockCode	text
Description	text
Quantity	int
InvoiceDate	text
UnitPrice	double
CustomerID	text
Country	text

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

- **Explanation**:

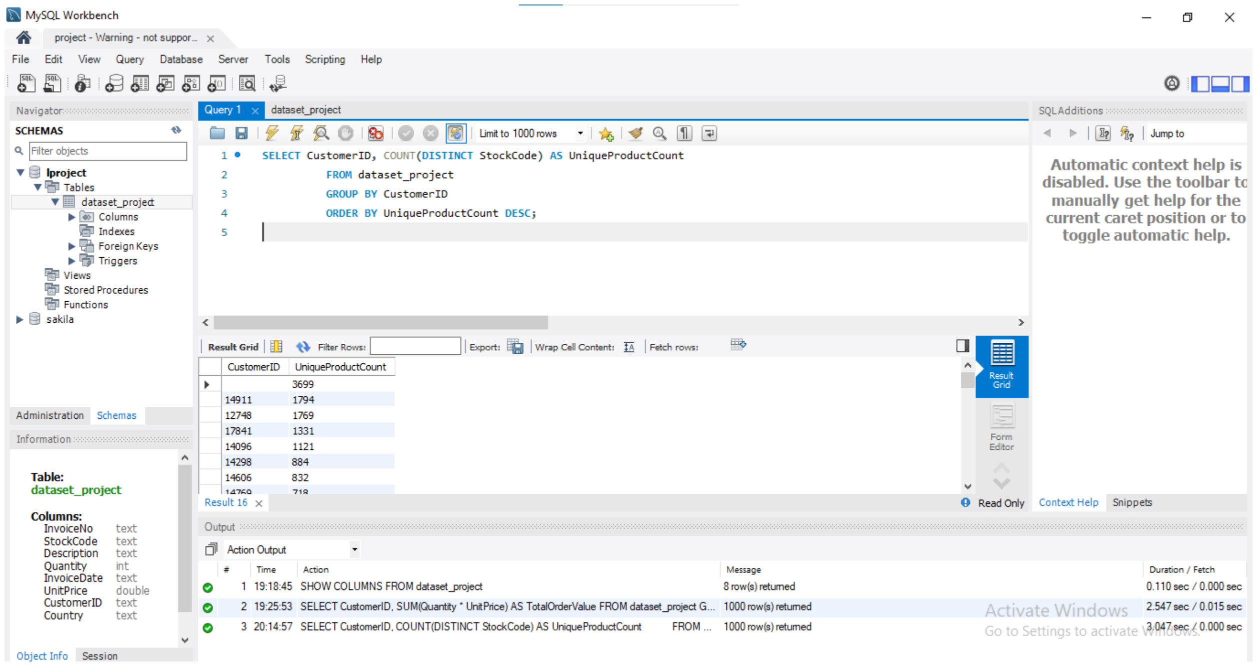
This query calculates the total order value for each customer, providing insight into the distribution of purchasing behavior.

3.3 Unique Products Purchased by Each Customer

- **Code**

```
SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount
FROM dataset_project
GROUP BY CustomerID
ORDER BY UniqueProductCount DESC;
```

- **Output Screenshot:**



```
SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount
        FROM dataset_project
        GROUP BY CustomerID
        ORDER BY UniqueProductCount DESC;
```

CustomerID	UniqueProductCount
	3699
14911	1794
12748	1769
17841	1331
14096	1121
14298	884
14606	832
14769	718

Table: dataset_project

Columns:

InvoiceNo	text
StockCode	text
Description	text
Quantity	int
InvoiceDate	text
UnitPrice	double
CustomerID	text
Country	text

```
#  Time      Action                                                                        Message               Duration / Fetch
1  19:18:45  SHOW COLUMNS FROM dataset_project                                             8 row(s) returned     0.110 sec / 0.000 sec
2  19:25:53  SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM dataset_project G...  1000 row(s) returned  2.547 sec / 0.015 sec
3  20:14:57  SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount    FROM ...  1000 row(s) returned  3.047 sec / 0.000 sec
```

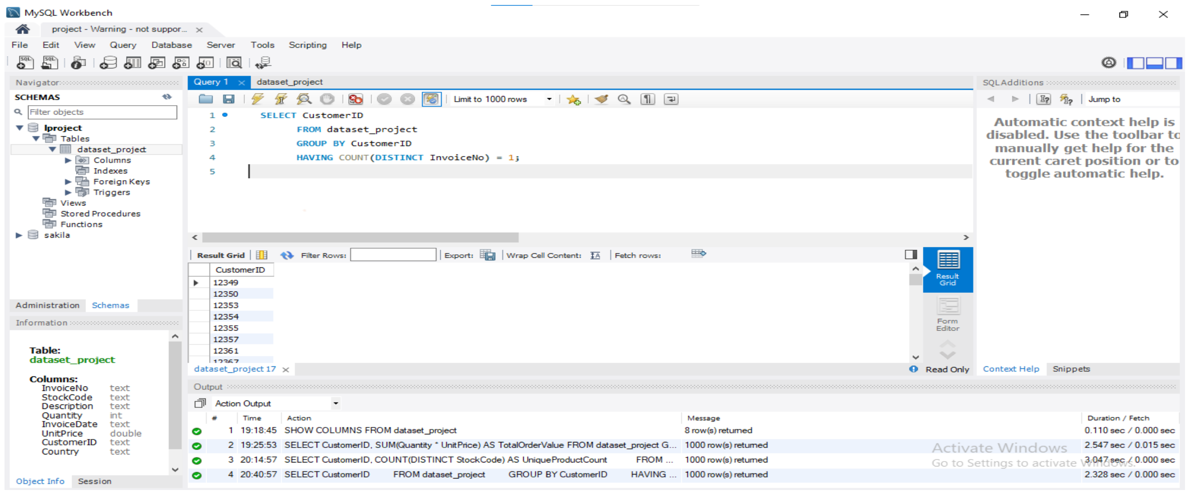
- **Explanation:**
  By counting distinct StockCodes for each customer, this query reveals the number of unique products purchased by each customer.

3.4 Customers with Single Purchase

- Code

SELECT CustomerID
FROM dataset_project
GROUP BY CustomerID
HAVING COUNT(DISTINCT InvoiceNo) = 1;

- **Output Screenshot:**



```
SELECT CustomerID
        FROM dataset_project
        GROUP BY CustomerID
        HAVING COUNT(DISTINCT InvoiceNo) = 1;
```

CustomerID
12349
12350
12353
12354
12355
12357
12361
12367

Table: dataset_project

Columns:

InvoiceNo	text
StockCode	text
Description	text
Quantity	int
InvoiceDate	text
UnitPrice	double
CustomerID	text
Country	text

```
#  Time      Action                                                                        Message               Duration / Fetch
1  19:18:45  SHOW COLUMNS FROM dataset_project                                             8 row(s) returned     0.110 sec / 0.000 sec
2  19:25:53  SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM dataset_project G...  1000 row(s) returned  2.547 sec / 0.015 sec
3  20:14:57  SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount    FROM ...  1000 row(s) returned  3.047 sec / 0.000 sec
4  20:40:57  SELECT CustomerID    FROM dataset_project    GROUP BY CustomerID    HAVING ...  1000 row(s) returned  2.328 sec / 0.000 sec
```

- **Explanation**
  This query identifies customers who have made only a single purchase from the company.

3.5 Most Commonly Purchased Products Together

- **Code**

```
SELECT A.StockCode AS Product1, B.StockCode AS Product2, COUNT(*) AS Frequency
FROM dataset_project A
JOIN dataset_project B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode < B.StockCode
GROUP BY Product1, Product2
ORDER BY Frequency DESC
Limit 10;
```

- **Explanation**

This query identifies products that are frequently purchased together by customers, revealing potential product affinities.

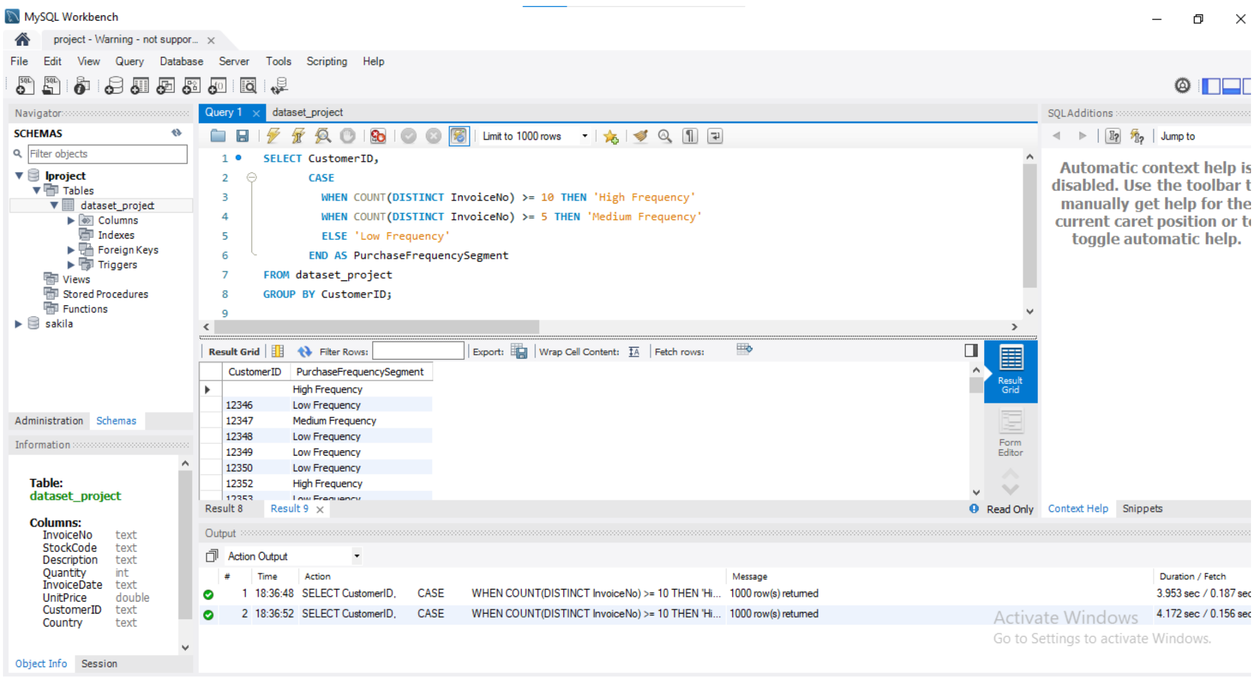
4. Advanced Queries:

4.1 Customer Segmentation by Purchase Frequency**

- **Code**

```
SELECT CustomerID,
    CASE
      WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'High Frequency'
      WHEN COUNT(DISTINCT InvoiceNo) >= 5 THEN 'Medium Frequency'
      ELSE 'Low Frequency'
    END AS PurchaseFrequencySegment
FROM dataset_project
GROUP BY CustomerID;
```

- **Output Screenshot:**



```
SELECT CustomerID,
       CASE
         WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'High Frequency'
         WHEN COUNT(DISTINCT InvoiceNo) >= 5 THEN 'Medium Frequency'
         ELSE 'Low Frequency'
       END AS PurchaseFrequencySegment
FROM dataset_project
GROUP BY CustomerID;
```

CustomerID	PurchaseFrequencySegment
	High Frequency
12346	Low Frequency
12347	Medium Frequency
12348	Low Frequency
12349	Low Frequency
12350	Low Frequency
12352	High Frequency

```
#  Time      Action                                                                     Message                Duration / Fetch
1  18:36:48  SELECT CustomerID,   CASE   WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'Hi...  1000 row(s) returned   3.953 sec / 0.187 sec
2  18:36:52  SELECT CustomerID,   CASE   WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'Hi...  1000 row(s) returned   4.172 sec / 0.156 sec
```

- **Explanation**

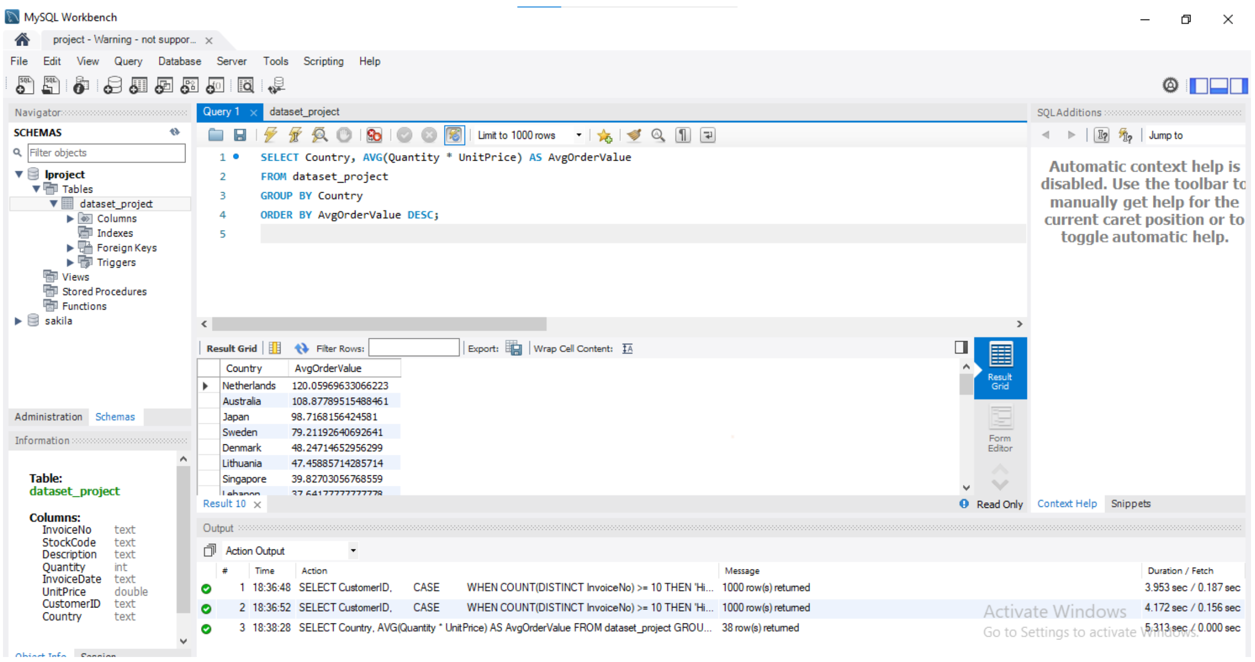
  This query categorizes customers into segments based on their purchase frequency, allowing for targeted marketing and engagement strategies.

4.2 Average Order Value by Country

- **Code**

```
SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue
FROM dataset_project
GROUP BY Country
ORDER BY AvgOrderValue DESC;
```

- **Output Screenshot:**



```
SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue
FROM dataset_project
GROUP BY Country
ORDER BY AvgOrderValue DESC;
```

Country	AvgOrderValue
Netherlands	120.05969633066223
Australia	108.87789515488461
Japan	98.7168156424581
Sweden	79.21192640692641
Denmark	48.24714652956299
Lithuania	47.45885714285714
Singapore	39.82703056768559
Lebanon	37.64177777777778

```
#  Time      Action                                                                    Message              Duration / Fetch
1  18:36:48  SELECT CustomerID,   CASE   WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'Hi...  1000 row(s) returned  3.953 sec / 0.187 sec
2  18:36:52  SELECT CustomerID,   CASE   WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'Hi...  1000 row(s) returned  4.172 sec / 0.156 sec
3  18:38:28  SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM dataset_project GROU...  38 row(s) returned  5.313 sec / 0.000 sec
```

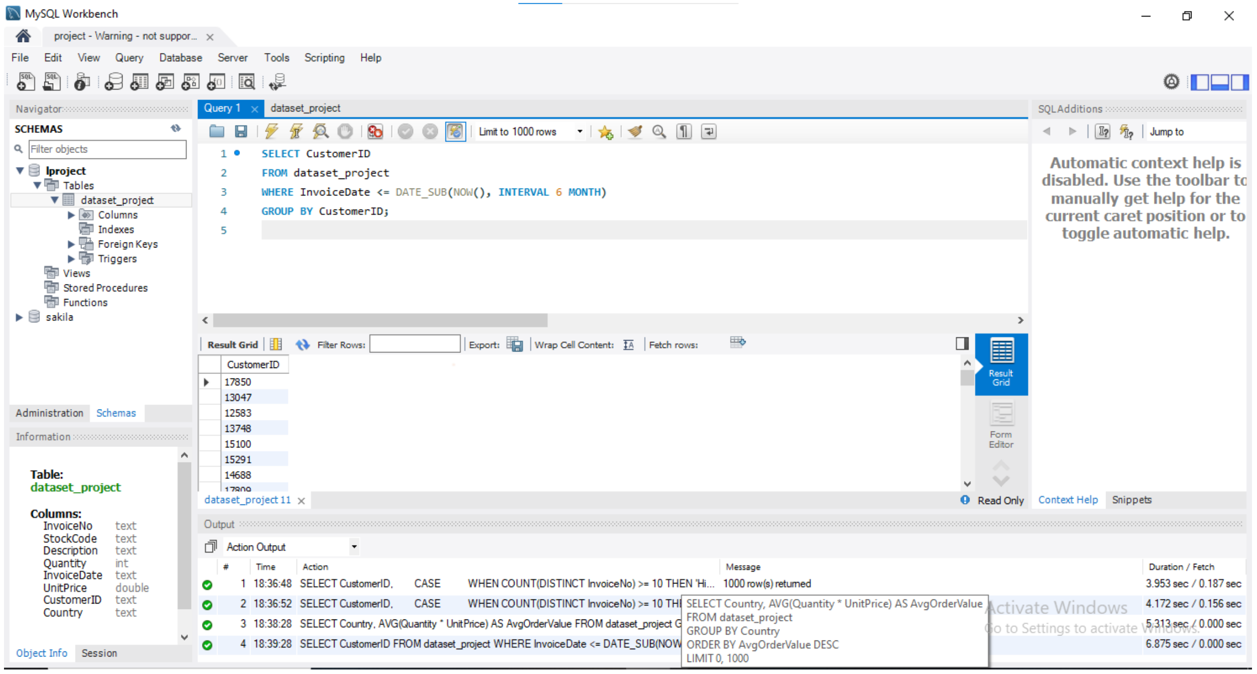
- **Explanation**

This query calculates the average order value for each country, helping to identify regions with the most valuable customers.

4.3 Customer Churn Analysis

- **Code**

```
SELECT CustomerID
FROM dataset_project
WHERE InvoiceDate < DATE_SUB(NOW(), INTERVAL 6 MONTH)
GROUP BY CustomerID;
```



```
SELECT CustomerID
FROM dataset_project
WHERE InvoiceDate <= DATE_SUB(NOW(), INTERVAL 6 MONTH)
GROUP BY CustomerID;
```

```
CustomerID
17850
13047
12583
13748
15100
15291
14688
```

Table: dataset_project

Columns:

InvoiceNo	text
StockCode	text
Description	text
Quantity	int
InvoiceDate	text
UnitPrice	double
CustomerID	text
Country	text

```
#  Time      Action                                                                        Message                Duration / Fetch
1  18:36:48  SELECT CustomerID,    CASE    WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'Hi...  1000 row(s) returned   3.953 sec / 0.187 sec
2  18:36:52  SELECT CustomerID,    CASE    WHEN COUNT(DISTINCT InvoiceNo) >= 10 THE                                4.172 sec / 0.156 sec
3  18:38:28  SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM dataset_project G                     5.313 sec / 0.000 sec
4  18:39:28  SELECT CustomerID FROM dataset_project WHERE InvoiceDate <= DATE_SUB(NOW                             6.875 sec / 0.000 sec
```

```
SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue
FROM dataset_project
GROUP BY Country
ORDER BY AvgOrderValue DESC
LIMIT 0, 1000
```

- **Explanation**

  By identifying customers who haven't made a purchase in the last 6 months, this query helps assess customer churn.

4.4 Product Affinity Analysis

- **Code**

```
SELECT a.StockCode AS Product1, b.StockCode AS Product2, COUNT(*) AS Frequency
FROM dataset_project a
JOIN dataset_project b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode
GROUP BY Product1, Product2
ORDER BY Frequency DESC;
```

- **Explanation**

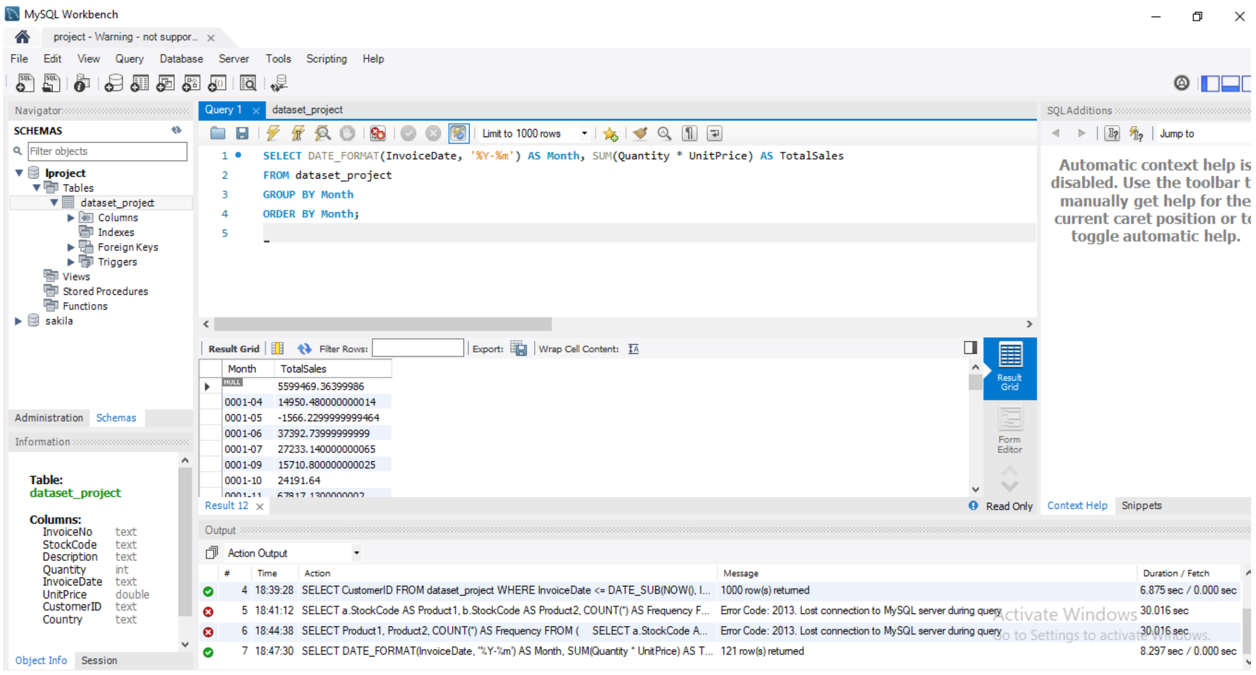
  This query determines which products are often purchased together by customers, aiding in cross-selling strategies.

4.5 Time-based Analysis

- **Code**

```
SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month, SUM(Quantity) AS TotalQuantity
FROM dataset_project
GROUP BY Month
ORDER BY Month;
```

- **Output Screenshot**



```
SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month, SUM(Quantity * UnitPrice) AS TotalSales
FROM dataset_project
GROUP BY Month
ORDER BY Month;
```

Month	TotalSales
NULL	5599469.36399986
0001-04	14950.480000000014
0001-05	-1566.2299999999464
0001-06	37392.73999999999
0001-07	27233.140000000065
0001-09	15710.800000000025
0001-10	24191.64
0001-11	67817.1300000002

```
#  Time      Action                                                                        Message                                                       Duration / Fetch
4  18:39:28  SELECT CustomerID FROM dataset_project WHERE InvoiceDate <= DATE_SUB(NOW(), I...  1000 row(s) returned                                          6.875 sec / 0.000 sec
5  18:41:12  SELECT a.StockCode AS Product1, b.StockCode AS Product2, COUNT(*) AS Frequency F...  Error Code: 2013. Lost connection to MySQL server during query  30.016 sec
6  18:44:38  SELECT Product1, Product2, COUNT(*) AS Frequency FROM (    SELECT a.StockCode A...  Error Code: 2013. Lost connection to MySQL server during query  30.016 sec
7  18:47:30  SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month, SUM(Quantity * UnitPrice) AS T...  121 row(s) returned                                           8.297 sec / 0.000 sec
```

- **Explanation**

    This query explores trends in customer behavior over time, revealing monthly sales patterns.

5. Conclusion:

Incorporating both beginner and advanced queries, our analysis has revealed multifaceted insights into customer behavior within the dataset. The basic queries provided initial glimpses into order values, product preferences, and customer engagement. Building upon this foundation, the advanced queries delved deeper, uncovering customer segmentation, geographic trends, churn risk, product affinities, and temporal patterns.

These insights equip us with the knowledge to tailor strategies for diverse customer segments, optimize marketing efforts, and enhance customer satisfaction. The synergy between basic and advanced queries has transformed raw data into actionable intelligence, driving data-driven decision-making and fostering growth for our business.