

# Project Report: Exploring Customer Insights Through 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.

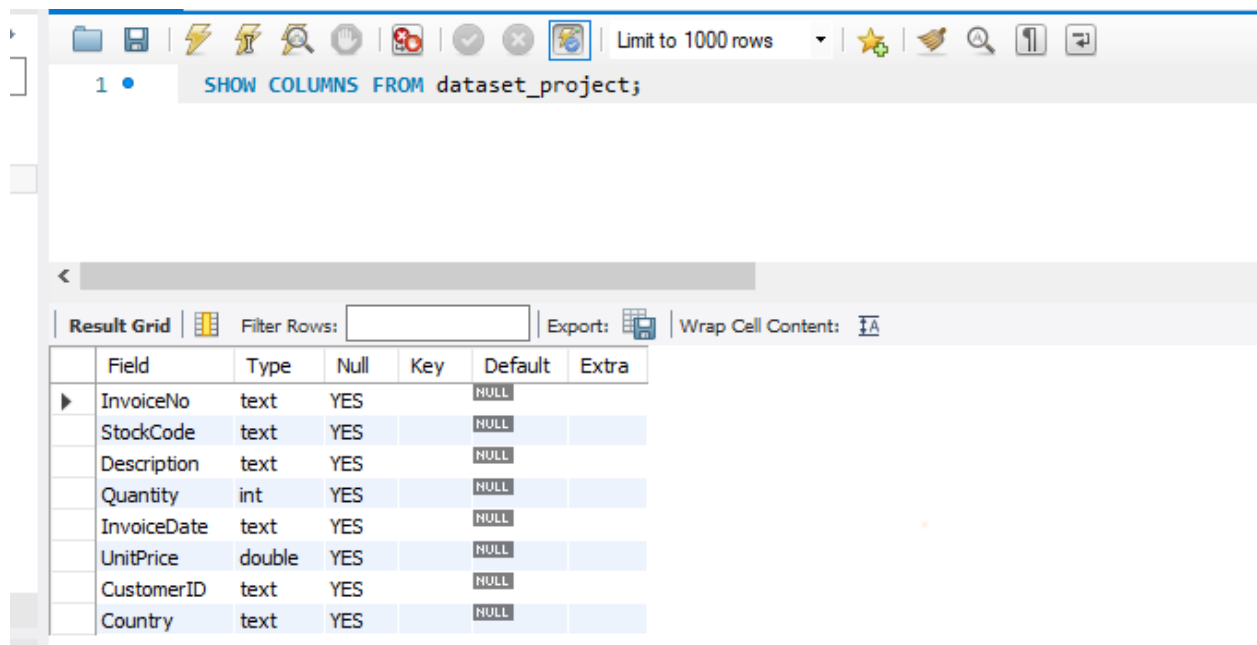
## 2. Dataset Overview:

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;
- **Output Screenshot:**



The screenshot displays the MySQL Workbench interface. The SQL editor at the top contains the query: `SHOW COLUMNS FROM dataset_project;`. Below the editor, the 'Result Grid' tab is active, showing the output of the query. The result is a table with 8 columns: Field, Type, Null, Key, Default, Extra, and an additional column. The data rows list the fields: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country, along with their respective data types, nullability, and default values.

	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		

- **Explanation:**

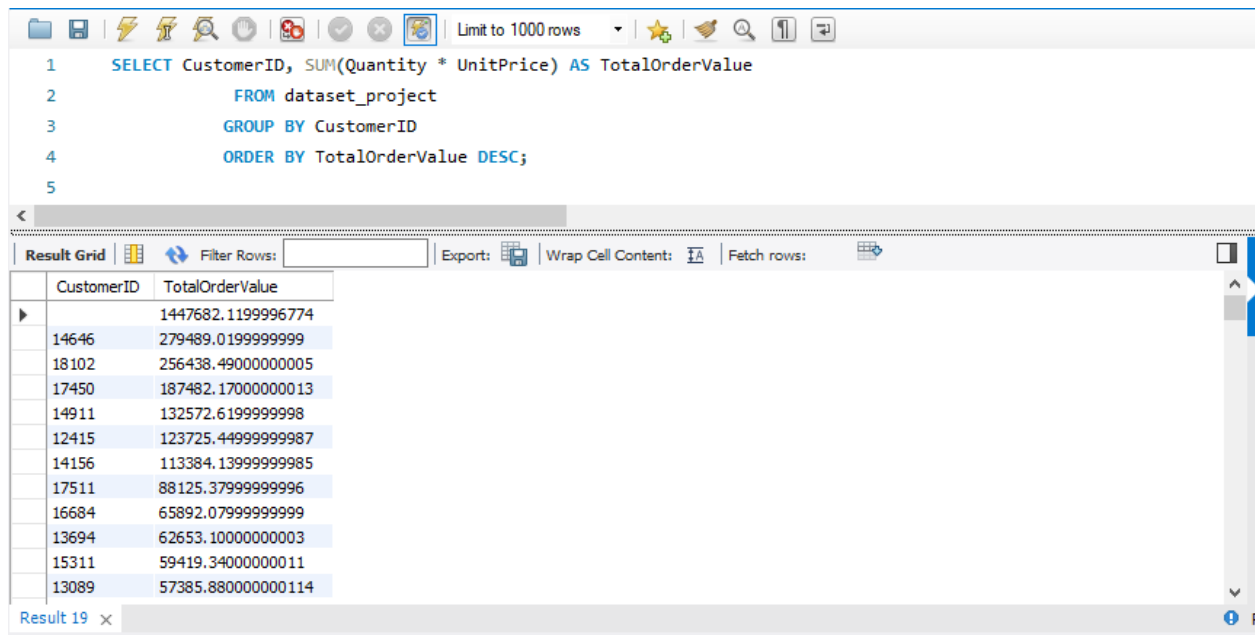
The query above displays the metadata of the dataset, listing the columns, their data types, and any constraints.

### 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:**



The screenshot shows a SQL query editor with the following query:

```
1 SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue
2 FROM dataset_project
3 GROUP BY CustomerID
4 ORDER BY TotalOrderValue DESC;
5
```

Below the query, the results are displayed in a table with two columns: CustomerID and TotalOrderValue. The table is sorted in descending order of TotalOrderValue.

CustomerID	TotalOrderValue
14646	279489.01999999999
18102	256438.49000000005
17450	187482.17000000013
14911	132572.61999999998
12415	123725.44999999987
14156	113384.13999999985
17511	88125.37999999996
16684	65892.07999999999
13694	62653.10000000003
15311	59419.34000000011
13089	57385.880000000114

- **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;
```

The screenshot shows a SQL query editor with the following query:

```

1  SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount
2  FROM dataset_project
3  GROUP BY CustomerID
4  ORDER BY UniqueProductCount DESC;
5

```

Below the query editor is the 'Result Grid' showing the results of the query. The grid has two columns: 'CustomerID' and 'UniqueProductCount'. The results are sorted in descending order of 'UniqueProductCount'.

CustomerID	UniqueProductCount
14911	1794
12748	1769
17841	1331
14096	1121
14298	884
14606	832
14769	718
14156	716
14646	703
13089	636
14456	580

- **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:**

```

2      FROM dataset_project
3      GROUP BY CustomerID
4      HAVING COUNT(DISTINCT InvoiceNo) = 1;
5
6

```

CustomerID
12349
12350
12353
12354
12355
12357
12361
12367
12373
12374
12378
12390

dataset\_project 22 x

- **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.

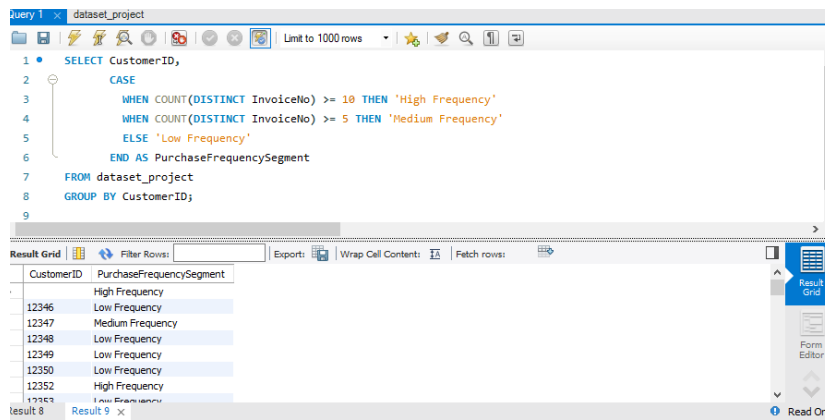
## 4. Advanced Queries:

### 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:**



The screenshot shows a SQL query editor with the following query:

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

The results are displayed in a table with the following data:

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

- **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:**

The screenshot shows a SQL query editor with the following query:

```

1 • SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue
2 FROM dataset_project
3 GROUP BY Country
4 ORDER BY AvgOrderValue DESC;
5

```

Below the query is a result grid with two columns: Country and AvgOrderValue. The data is sorted in descending order of AvgOrderValue.

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

- **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;

```

The screenshot shows a SQL query editor with the following query:

```

1 • SELECT CustomerID
2 FROM dataset_project
3 WHERE InvoiceDate <= DATE_SUB(NOW(), INTERVAL 6 MONTH)
4 GROUP BY CustomerID;
5

```

Below the query is a result grid with one column: CustomerID. The data lists customer IDs who haven't made a purchase in the last 6 months.

CustomerID
17850
13047
12583
13748
15100
15291
14688
17860

- **Explanation**

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

### 4.4 Time-based Analysis

- **Code**

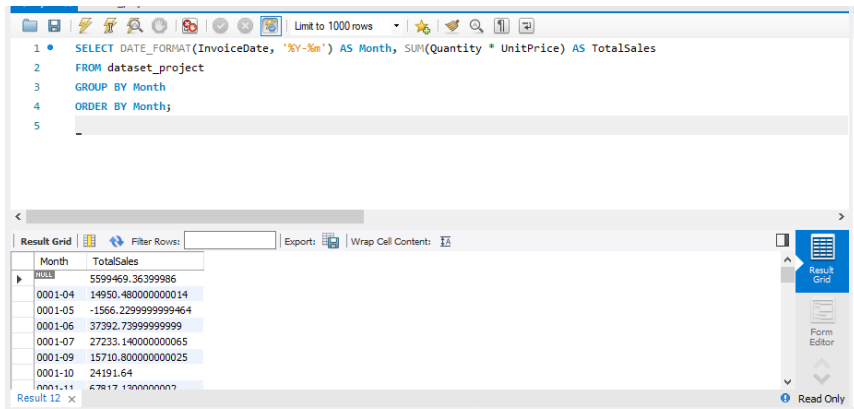
```

SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month, SUM(Quantity) AS TotalQuantity
FROM dataset_project

```

GROUP BY Month  
ORDER BY Month;

● **Output Screenshot**



● **Explanation**

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

5. Conclusion:

In conclusion, the execution of both beginner and advanced queries has provided valuable insights into customer behavior and interactions within the dataset. The beginner queries revealed essential aspects such as order value distribution, unique product preferences, and customer engagement. These insights serve as the groundwork for understanding customer preferences.

Moving to the advanced queries, we gained a deeper understanding of customer segmentation, geographical preferences, churn behavior, product affinities, and time-based trends. These insights enable data-driven decision-making and strategic planning to enhance customer engagement, optimize resources, and boost business growth.

Together, the combination of both beginner and advanced queries has provided a comprehensive perspective on customer behavior, enabling us to tailor strategies and initiatives that resonate with our customers and drive positive outcomes for the business.