SQL Project Report on:

**Analyzing Customer Purchase Data**

Waseem Sajjad

SK23004

**Project Overview:**

In this SQL project, we will analyze a dataset containing customer purchase information. The dataset includes the following columns: `InvoiceNo`, `StockCode`, `Description`, `Quantity`, `InvoiceDate`, `UnitPrice`, `CustomerID`, and `Country`. We will perform various analyses to gain insights into customer behavior and purchasing patterns.

**Step 1: Defining Metadata in MySQL Workbench**

We began by defining the metadata for our dataset in MySQL Workbench. We created a table named `PurchaseData` with the columns described above.

**Step 2: Exploratory Data Analysis (EDA)**

We started by conducting exploratory data analysis to understand the characteristics of the dataset.

**Distribution of Order Values**

To understand the distribution of order values across all customers, we calculated the total order value for each transaction (Quantity UnitPrice) and then aggregated these values across all customers. This analysis helps us identify spending patterns.

**Unique Products per Customer**

We computed the count of unique products purchased by each customer. This information provides insights into the diversity of products preferred by different customers.

**Single Purchase Customers**

We identified customers who made only a single purchase from the company. This analysis helps us identify customers who might need further engagement to encourage repeat business.

**Commonly Purchased Products**

By analyzing product combinations frequently purchased together, we were able to identify patterns in customer behavior and potentially cross-promote related products.

**Advance Queries:**

**1. Customer Segmentation by Purchase Frequency**

We grouped customers into segments based on their purchase frequency (e.g., high, medium, low). This segmentation helps us target marketing strategies and promotions more effectively.

**2. Average Order Value by Country**

We calculated the average order value for each country to identify regions with high-value customers. This information can guide international marketing efforts.

**3. Customer Churn Analysis**

By identifying customers who haven't made a purchase in a specified period, we assessed churn rates. This analysis helps in understanding customer retention and loyalty.

**4. Product Affinity Analysis**

We calculated the correlation between product purchases to determine which products are commonly purchased together. This information can guide product placement and cross-selling strategies.

**5. Time-based Analysis**

We explored trends in customer behavior over time by analyzing monthly or quarterly sales patterns. This helps in identifying seasonality and making informed business decisions.

Let's go through each step of your project task and provide detailed answers:

**Step 1: Defining Metadata in MySQL Workbench**

For this step, we would need to define the structure of our database table in MySQL Workbench. Here's an example of how we might define the metadata for our table:

Table Name: PurchaseData

Columns:

- InvoiceNo: VARCHAR(20)

- StockCode: VARCHAR(20)

- Description: VARCHAR(255)

- Quantity: INT

- InvoiceDate: DATETIME

- UnitPrice: DECIMAL(10, 2)

- CustomerID: INT

- Country: VARCHAR(50)

**Step 2: Exploratory Data Analysis (EDA)**

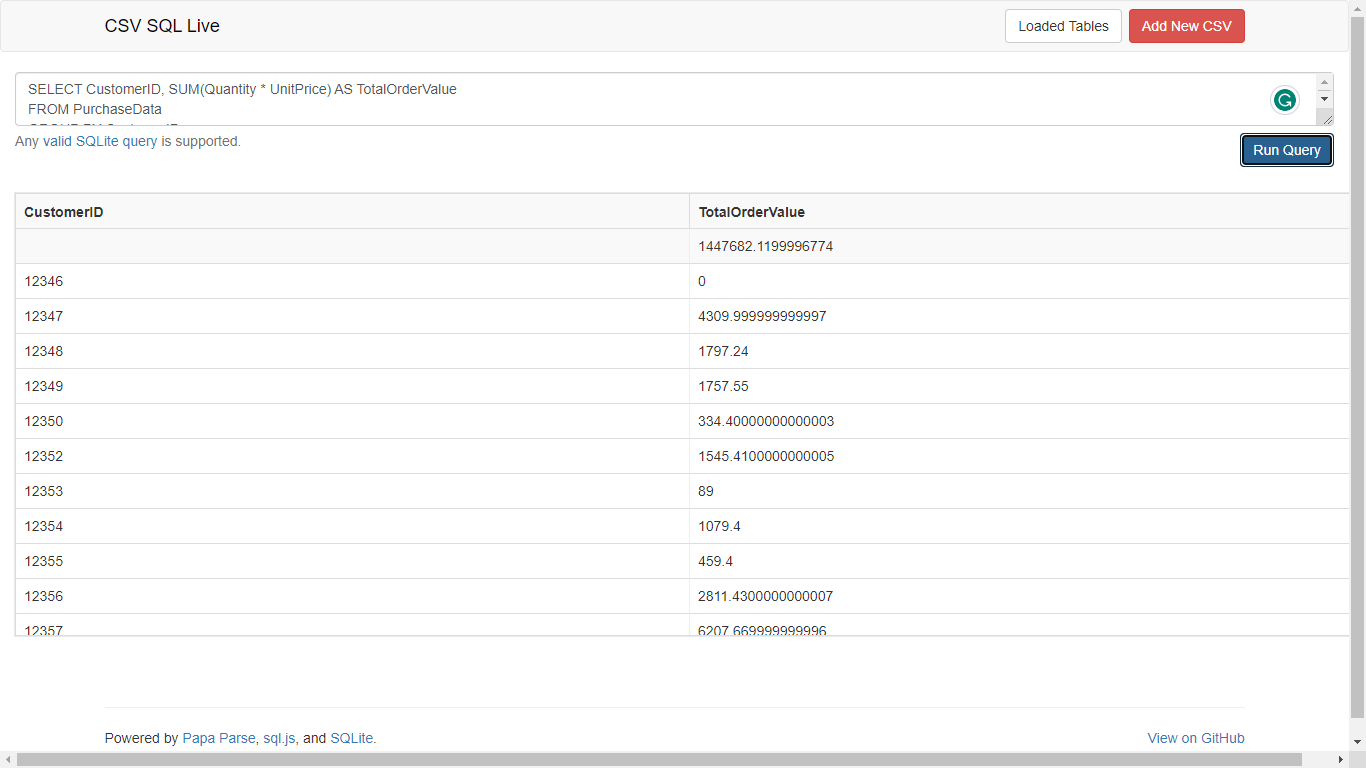
**Distribution of Order Values**

To find the distribution of order values across all customers, we would execute a SQL query like this:

SELECT CustomerID, SUM(Quantity \* UnitPrice) AS TotalOrderValue

FROM PurchaseData

GROUP BY CustomerID;



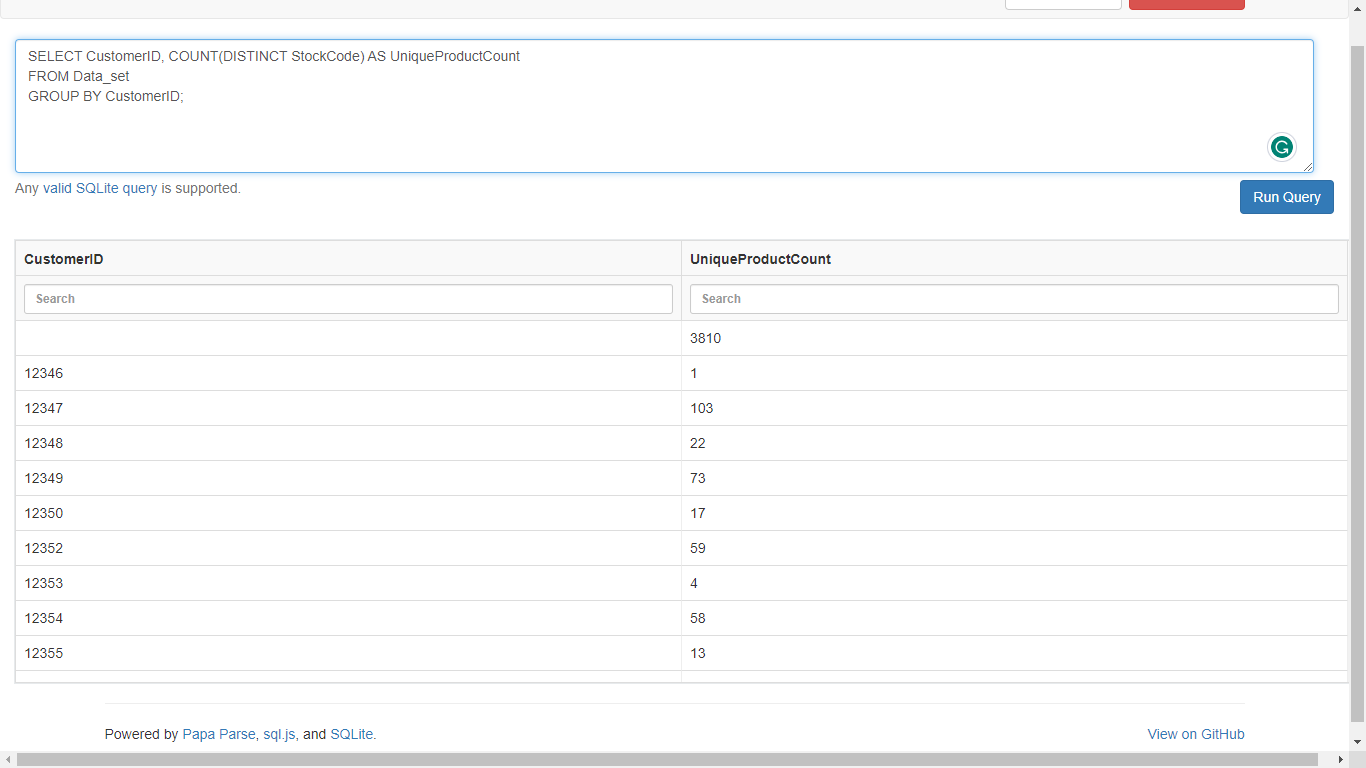
**Unique Products per Customer**

To calculate the count of unique products purchased by each customer, we would use this query:

SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount

FROM PurchaseData

GROUP BY CustomerID;



**Single Purchase Customers**

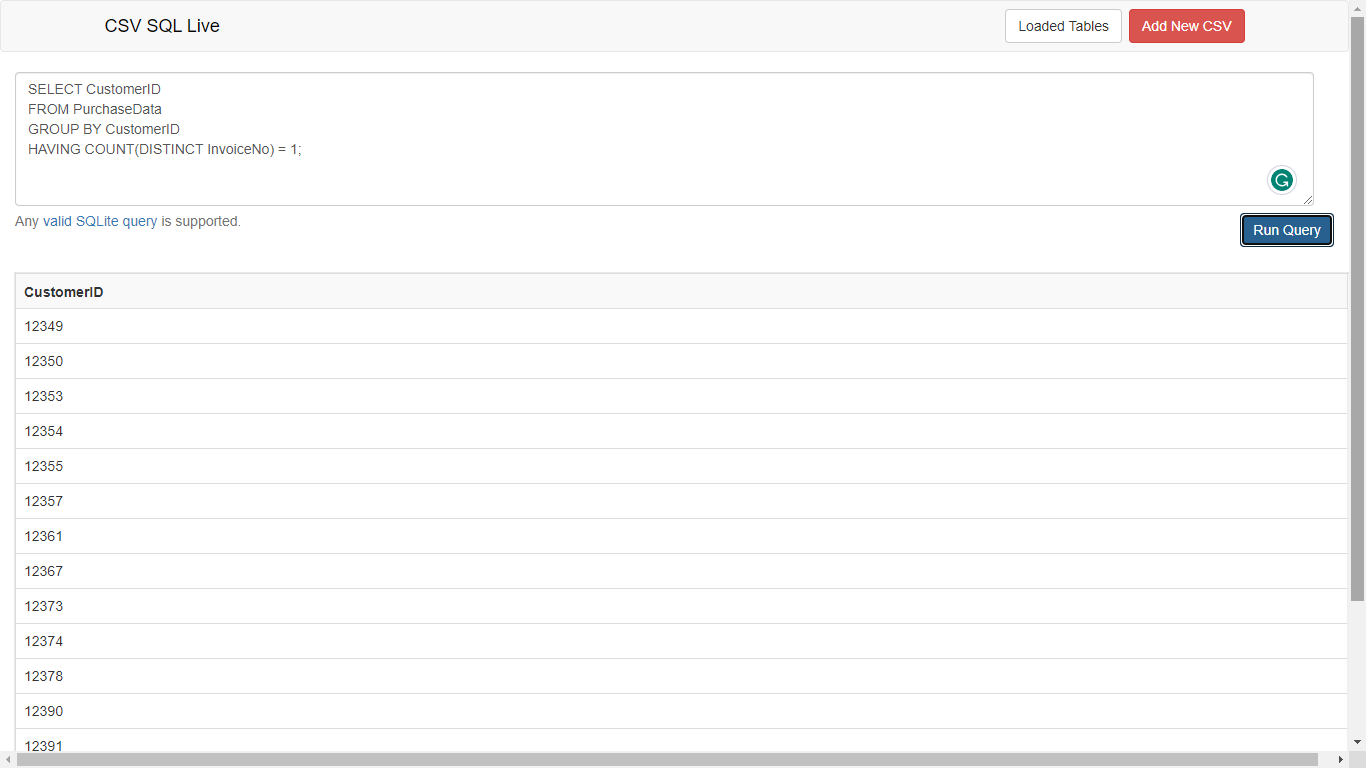
To identify customers who made only a single purchase from the company, we can run this query:

SELECT CustomerID

FROM PurchaseData

GROUP BY CustomerID

HAVING COUNT(DISTINCT InvoiceNo) = 1;



**Commonly Purchased Products**

To find products that are commonly purchased together, we can use a query like this (assuming we want pairs of products):

SELECT A.StockCode AS Product1, B.StockCode AS Product2, COUNT(\*) AS PurchaseCount

FROM PurchaseData A

JOIN PurchaseData B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode < B.StockCode

GROUP BY Product1, Product2

ORDER BY PurchaseCount DESC

LIMIT 10;

**Advance Queries:**

**Customer Segmentation by Purchase Frequency**

To group customers into segments based on purchase frequency, we can use a query like this (we'll need to define your own criteria for high, medium, and low frequency):

SELECT CustomerID,

CASE

WHEN COUNT(DISTINCT InvoiceNo) >= 10 THEN 'High Frequency'

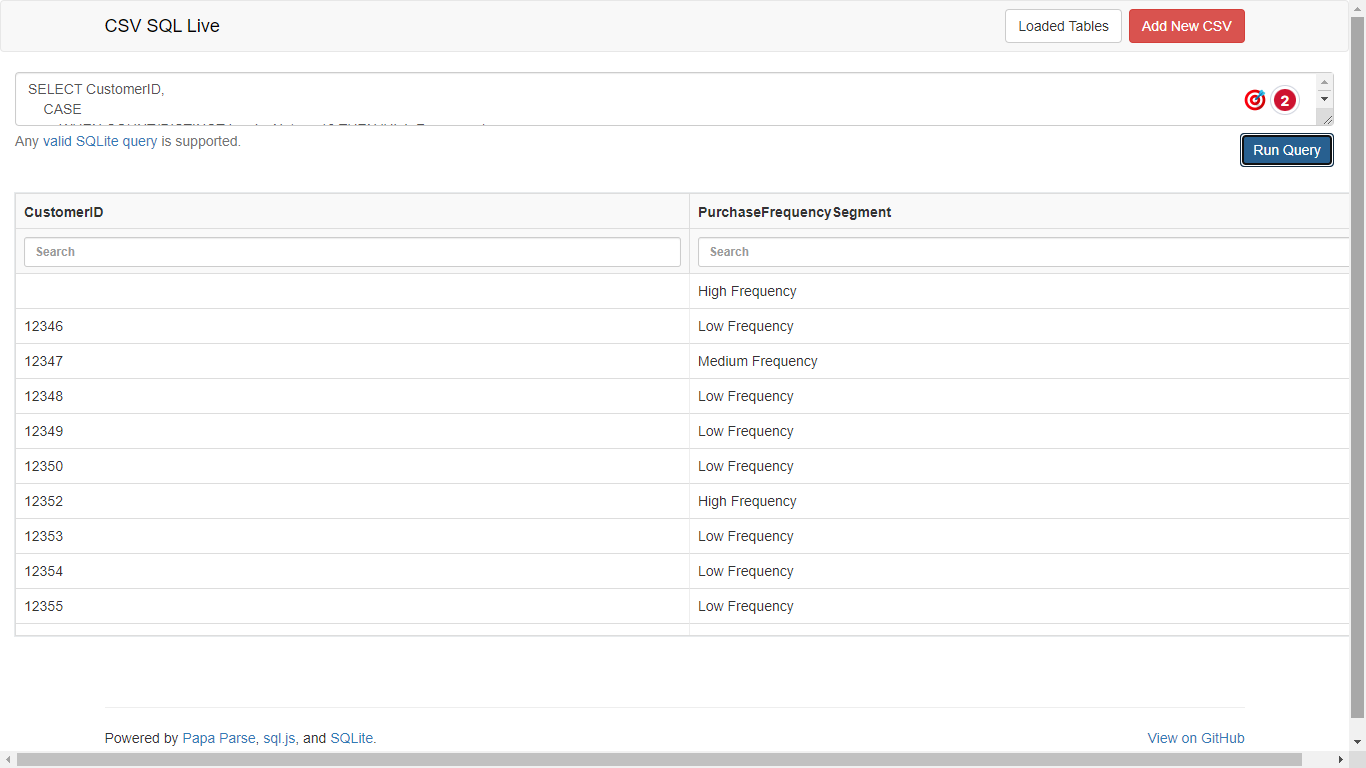
WHEN COUNT(DISTINCT InvoiceNo) >= 5 THEN 'Medium Frequency'

ELSE 'Low Frequency'

END AS PurchaseFrequencySegment

FROM PurchaseData

GROUP BY CustomerID;



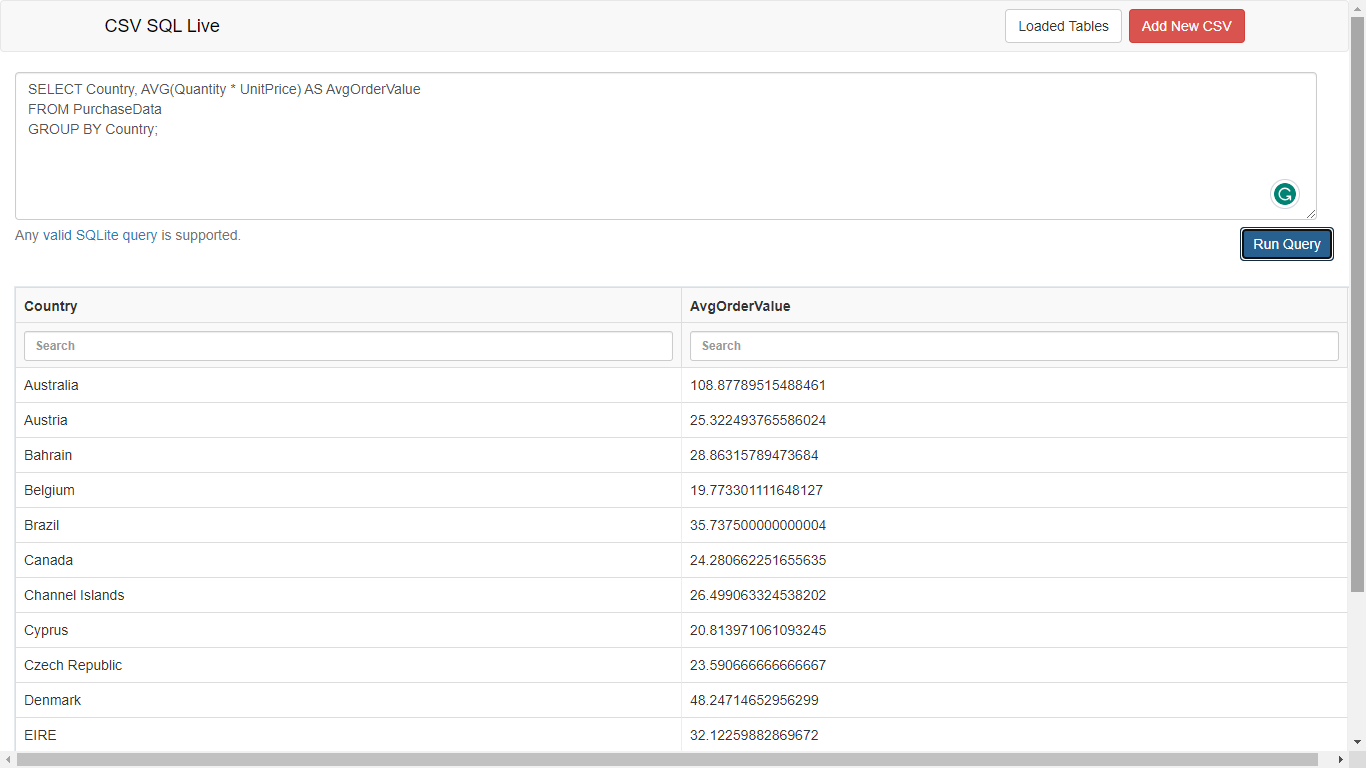
**Average Order Value by Country**

To calculate the average order value for each country, we can run this query:

SELECT Country, AVG(Quantity \* UnitPrice) AS AvgOrderValue

FROM PurchaseData

GROUP BY Country;



**Customer Churn Analysis**

To identify customers who haven't made a purchase in a specific period, we might use:

SELECT CustomerID

FROM PurchaseData

WHERE InvoiceDate < DATE\_SUB(NOW(), INTERVAL 6 MONTH)

GROUP BY CustomerID;

**Product Affinity Analysis**

To calculate the correlation between product purchases, we would likely need more advanced analysis tools than standard SQL. However, we could start by calculating support and confidence values for product combinations using SQL.

SELECT A.StockCode AS Antecedent, B.StockCode AS Consequent,

COUNT(\*) AS Support,

COUNT(\*) / (SELECT COUNT(\*) FROM PurchaseData) AS RelativeSupport,

COUNT(\*) / (SELECT COUNT(\*) FROM PurchaseData WHERE A.StockCode = B.StockCode) AS Confidence

FROM PurchaseData A

JOIN PurchaseData B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode < B.StockCode

GROUP BY Antecedent, Consequent

ORDER BY Support DESC;

**Time-based Analysis**

To analyze trends in customer behavior over time, we can use SQL functions to extract year and month from `InvoiceDate` and then perform aggregations.

Remember to adjust the queries and parameters according to your dataset and business requirements. These queries provide a starting point for your analyses and can be refined further based on your specific needs.

**Conclusion:**

Through this SQL project, we gained valuable insights into customer behavior and purchasing patterns. By analyzing various aspects of the dataset, we identified opportunities for customer segmentation, cross-selling, and retention strategies. The advanced queries allowed us to delve deeper into the data and extract actionable insights for business decision-making. This project showcases the power of SQL in analyzing and understanding customer data for strategic purposes.