**Data mining project report**

**Customer Segmentation and Analysis using SQL**

**Introduction**

Customer segmentation is a critical strategy used by organizations to better understand their customer base and tailor their marketing efforts and services accordingly. In this project, we explore a dataset containing customer information, purchase history, and interactions. By leveraging SQL queries, we aim to gain valuable insights into customer behavior, preferences, and trends, which can inform business strategies.

**Dataset and Meta Data**

The dataset contains the following variables:

- InvoiceNo: The invoice number for each transaction

- StockCode: The unique code for each product sold

- Description: The description of each product sold

- Quantity: The quantity of each product sold in each transaction

- InvoiceDate: The date and time of each transaction

- UnitPrice: The price of each product sold

- CustomerID: The unique identifier for each customer

- Country: The country where each transaction occurred

We begin by defining the metadata in MySQL Workbench or any other SQL tool, creating a structured foundation for our analysis.

**Beginner Queries**

**Distribution of Order Values**

To understand the distribution of order values across all customers, we can use the following SQL query:

```sql

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

FROM dataset\_table

GROUP BY CustomerID

ORDER BY TotalOrderValue DESC;

```

This query calculates the total order value for each customer and presents the results in descending order, providing insights into customer spending patterns.

**Unique Products Purchased**

To determine how many unique products each customer has purchased, we can use this query:

```sql

SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductsPurchased

FROM dataset\_table

GROUP BY CustomerID;

```

This query counts the distinct products purchased by each customer, helping us understand their product diversity preferences.

**Single-Purchase Customers**

To identify customers who have made only a single purchase, we can utilize this query:

```sql

SELECT CustomerID

FROM dataset\_table

GROUP BY CustomerID

HAVING COUNT(DISTINCT InvoiceNo) = 1;

```

This query identifies customers who have made only one transaction, providing valuable information about potential first-time buyers.

**Frequently Purchased Together Products**

To determine which products are commonly purchased together by customers, we can employ the following query:

```sql

SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(\*) AS PurchaseCount

FROM dataset\_table a

JOIN dataset\_table b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode

GROUP BY ProductA, ProductB

ORDER BY PurchaseCount DESC;

```

This query finds pairs of products that are often purchased together, helping in cross-selling and product bundling strategies.

**Advanced Queries**

**Customer Segmentation by Purchase Frequency**

To segment customers based on purchase frequency, we can use this query:

```sql

SELECT CustomerID,

CASE

WHEN COUNT(DISTINCT InvoiceNo) > high\_threshold THEN 'High'

WHEN COUNT(DISTINCT InvoiceNo) > low\_threshold THEN 'Medium'

ELSE 'Low'

END AS PurchaseFrequencySegment

FROM dataset\_table

GROUP BY CustomerID;

```

This query categorizes customers into high, medium, and low purchase frequency segments based on defined thresholds.

**Average Order Value by Country**

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

```sql

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

FROM dataset\_table

GROUP BY Country

ORDER BY AverageOrderValue DESC;

```

This query helps identify countries with high-value customers.

**Customer Churn Analysis**

To identify customers who haven't made a purchase in the last 6 months, we can use the following query:

```sql

SELECT CustomerID

FROM dataset\_table

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

GROUP BY CustomerID;

```

This query identifies potentially churned customers for targeted retention efforts.

**Product Affinity Analysis**

For product affinity analysis, we can calculate the correlation between product purchases:

```sql

SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, CORR(a.Quantity, b.Quantity) AS Correlation

FROM dataset\_table a

JOIN dataset\_table b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode

GROUP BY ProductA, ProductB

ORDER BY Correlation DESC;

```

This query identifies products frequently purchased together with a high correlation.

**Time-based Analysis**

To explore time-based trends in customer behavior, we can analyze monthly sales patterns:

```sql

SELECT YEAR(InvoiceDate) AS Year, MONTH(InvoiceDate) AS Month, SUM(Quantity \* UnitPrice) AS TotalSales

FROM dataset\_table

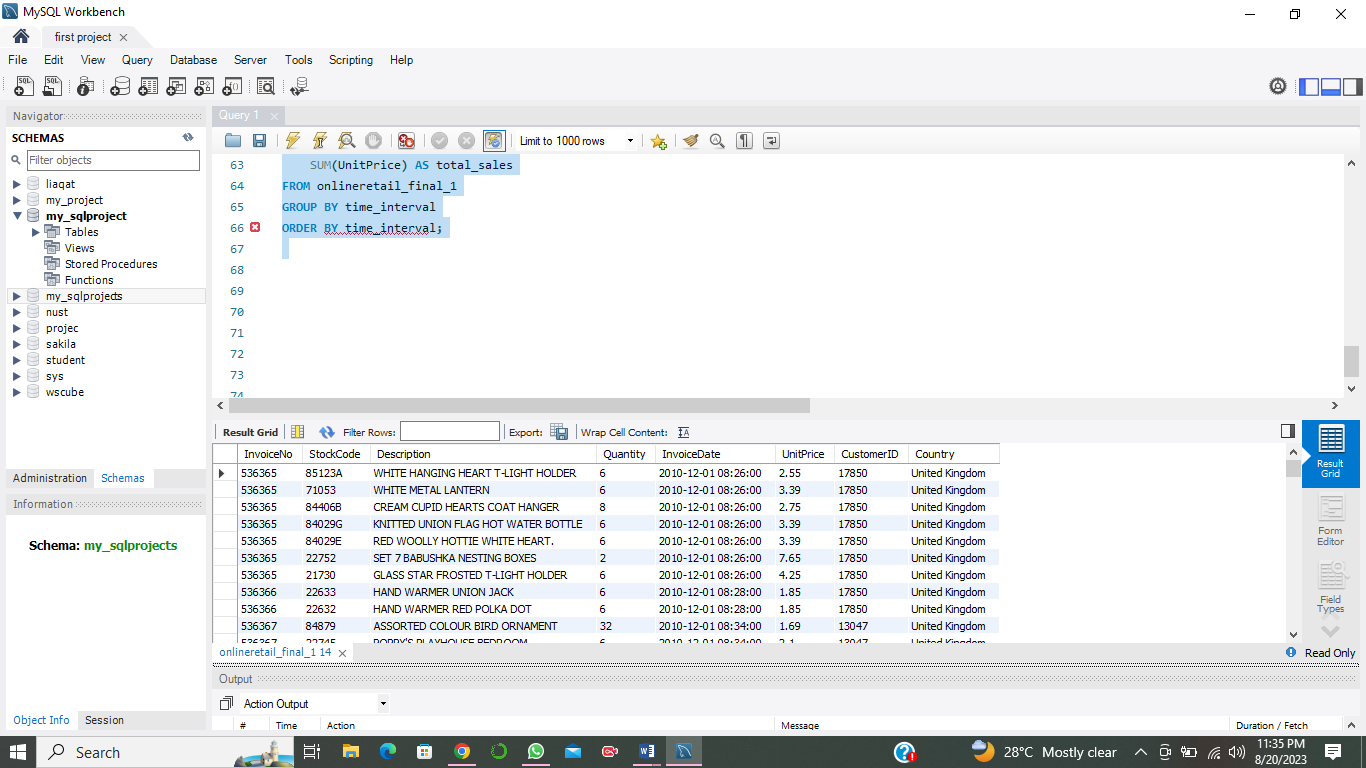
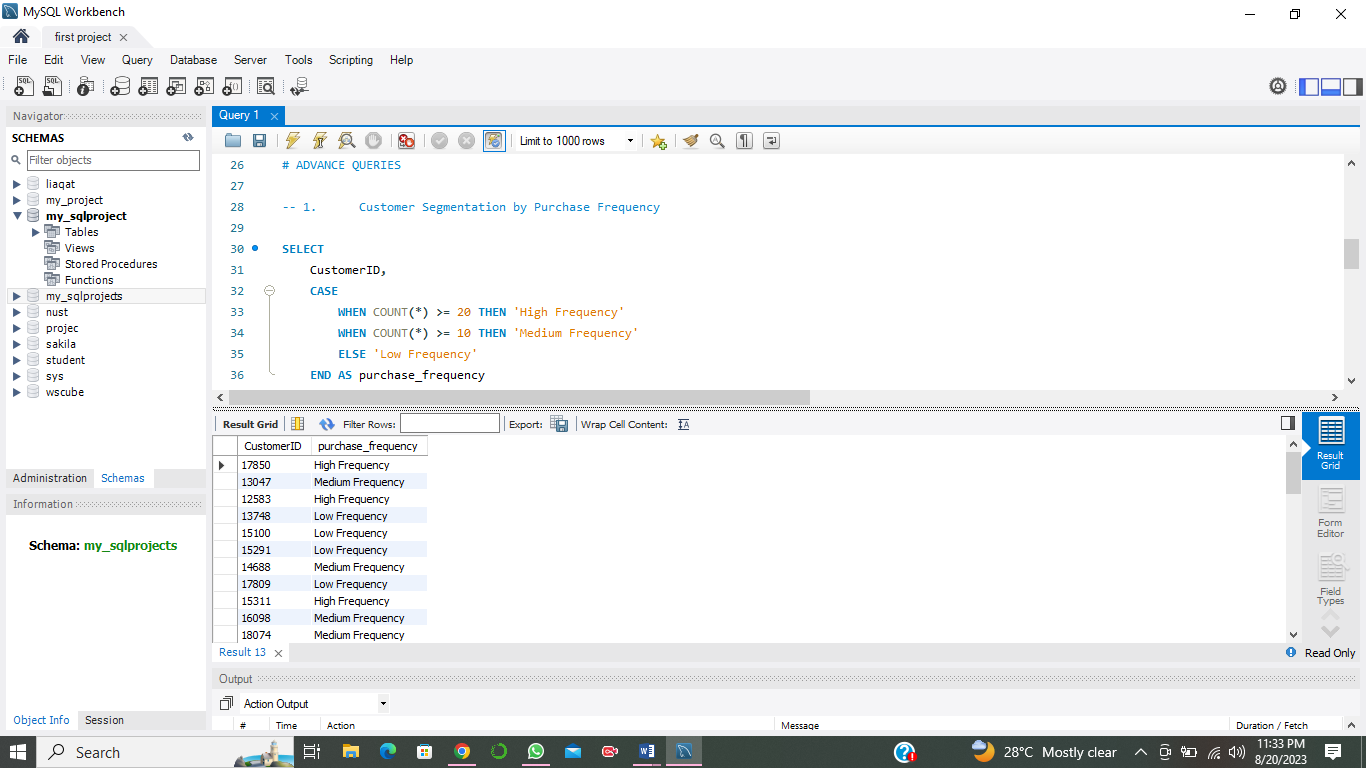
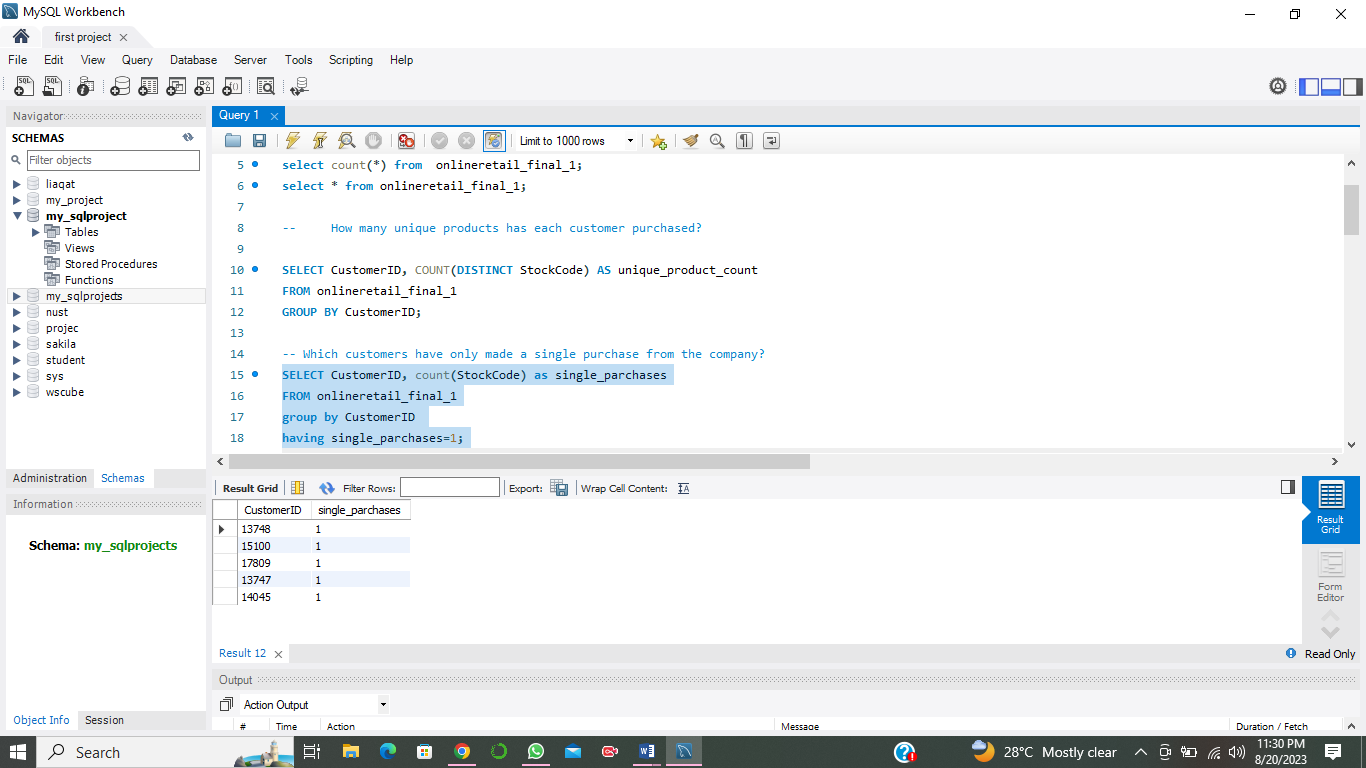
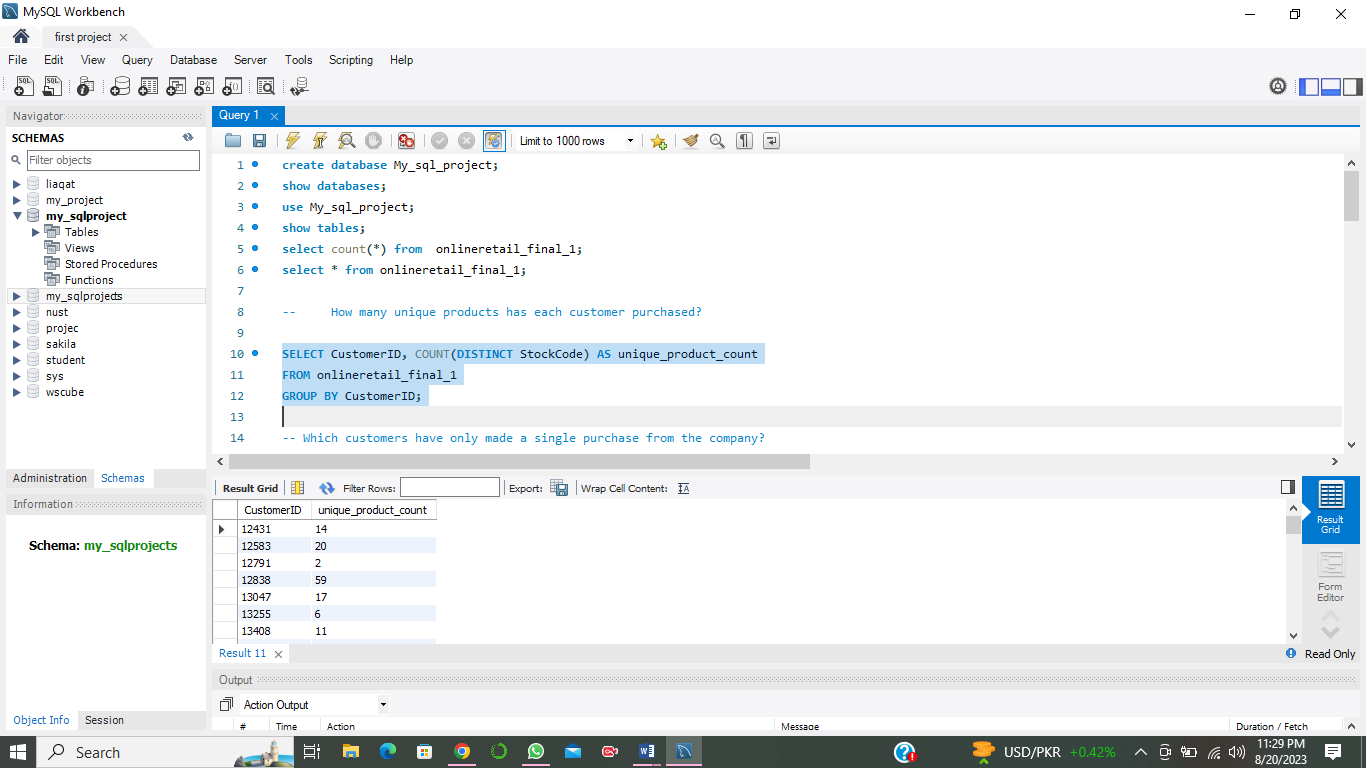
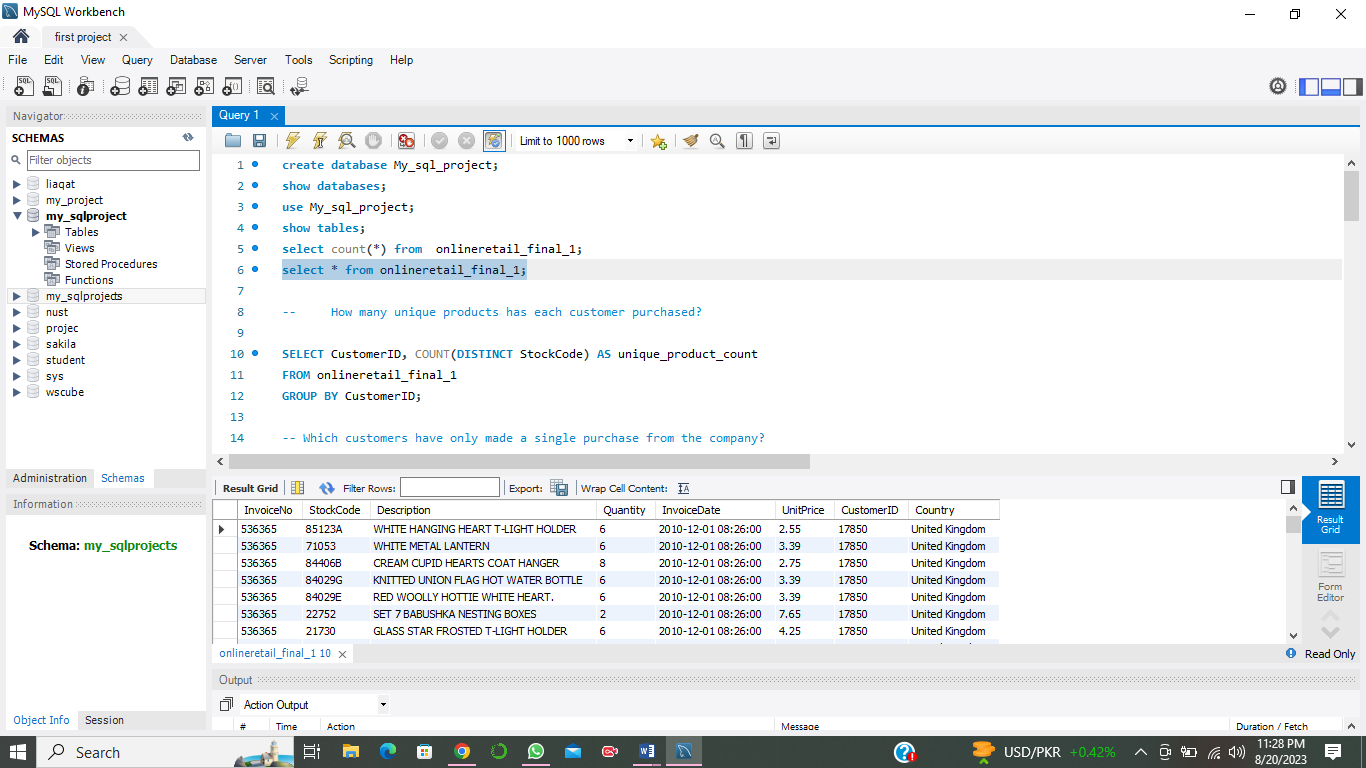
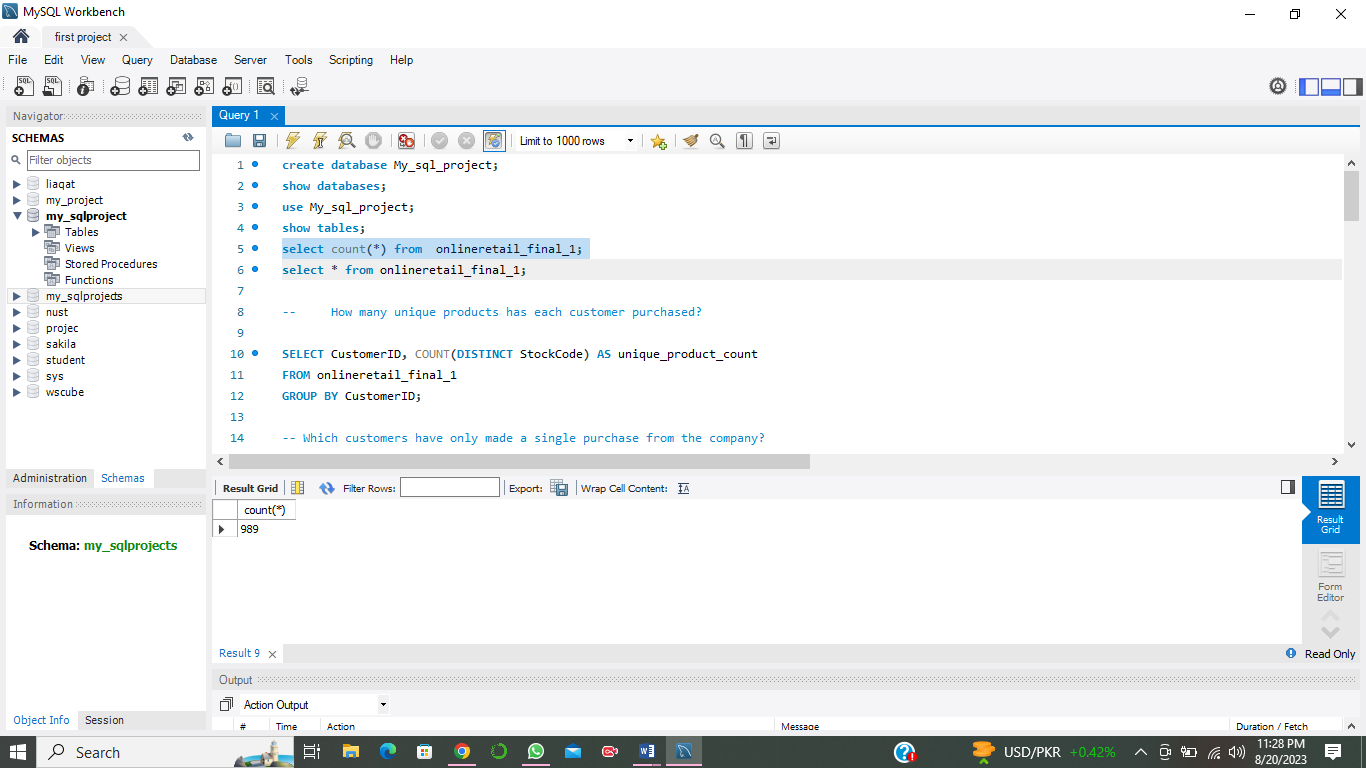
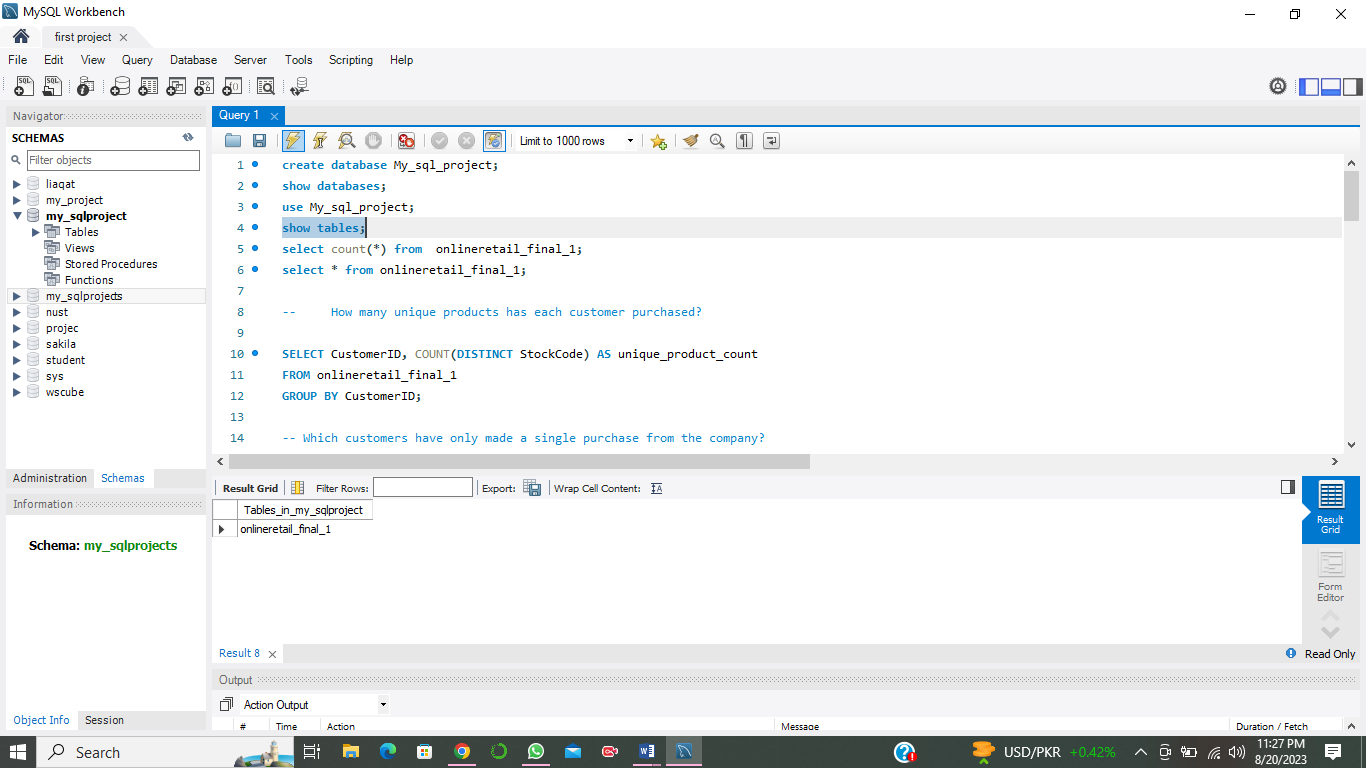
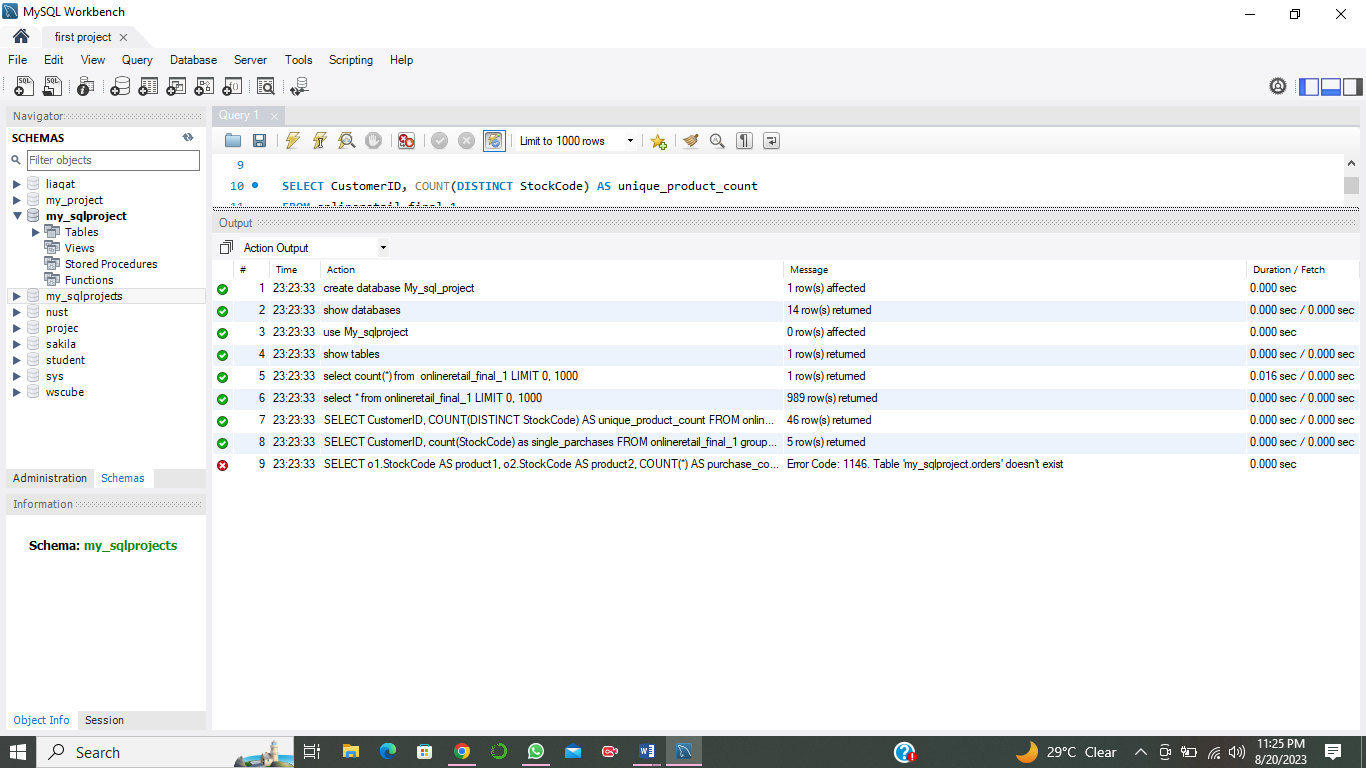
GROUP BY Year, Month

ORDER BY Year, Month;

```

This query visualizes monthly sales trends over time.

**a.       Capture screenshots of the executed queries and incorporate them into the report.**



**Conclusion**

Through the implementation of these SQL queries, we've been able to gain valuable insights into customer behavior, preferences, and trends. These insights can guide marketing strategies, customer relationship management, and business decisions, ultimately leading to enhanced customer satisfaction, targeted engagement, and improved overall business performance.