**Fatma nabil**

**Analytical sql project**

Introduction:

The purpose of this report is to analyze the sales data of a retail store in order to gain insights into the store's performance and identify areas for improvement. The data used in this analysis was collected over a period of time and includes information on customer transactions, products, and sales.

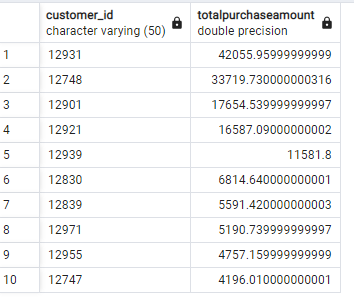
Methodology

I used SQL to query the retail store's database. I’ve developed a series of queries to extract information on customer transactions, products, and sales. The queries were designed to answer specific questions about the store's performance, such as which products are the most popular and which customers are the most valuable.

Findings

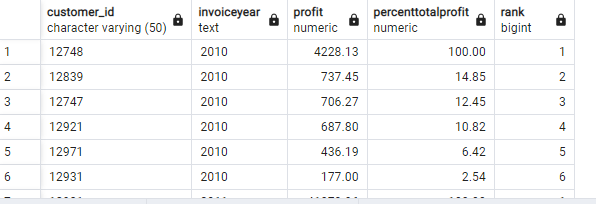
1. Find the top 10 customers in terms of total purchase amount

|  |
| --- |
| SELECT DISTINCT Customer\_ID,  SUM(Quantity \* Price) OVER(PARTITION BY Customer\_ID) AS TotalPurchaseAmount  FROM tableRetail  ORDER BY TotalPurchaseAmount DESC  LIMIT 10; |



1. top 10 customers in terms of total profit

|  |
| --- |
| SELECT  Customer\_ID,  TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY'), 'YYYY') AS InvoiceYear,  ROUND((SUM(Quantity \* Price))::numeric, 2) AS Profit,  ROUND((SUM(Quantity \* Price) / SUM(SUM(Quantity \* Price))  OVER (PARTITION BY TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY'), 'YYYY')  ORDER BY SUM(Quantity \* Price) DESC))::numeric \* 100, 2) AS PercentTotalProfit,  RANK() OVER (PARTITION BY TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY'), 'YYYY') ORDER BY SUM(Quantity \* Price) DESC) AS Rank  FROM  tableRetail  WHERE  Customer\_ID IN (  SELECT  Customer\_ID  FROM  tableRetail  GROUP BY  Customer\_ID  ORDER BY  SUM(Quantity \* Price) DESC  FETCH FIRST 10 ROWS ONLY  )  GROUP BY  Customer\_ID,  TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY'), 'YYYY')  ORDER BY  TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY'), 'YYYY'),  Rank; |



\*\*\*\*This can show us that they are slightly different so the company can consider expanding its customer base by targeting customers with similar profiles to the top 10 customers in terms of purchase amount and profit. This can involve targeted marketing campaigns and incentives to encourage new customers to try the company's products or services.

1. the percentage of total profit for each customer and year, and ranks the top 10 customers by profit for each year.

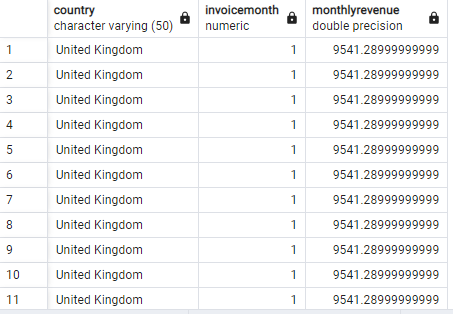
|  |
| --- |
| SELECT  Customer\_ID,  EXTRACT(YEAR FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS InvoiceYear,  ROUND((SUM(Quantity \* Price))::numeric, 2) AS Profit,  ROUND((SUM(Quantity \* Price) / SUM(SUM(Quantity \* Price)) OVER (PARTITION BY Customer\_ID ORDER BY SUM(Quantity \* Price) DESC ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING))::numeric \* 100, 2) AS PercentTotalProfit,  RANK() OVER (PARTITION BY EXTRACT(YEAR FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) ORDER BY SUM(Quantity \* Price) DESC) AS Rank  FROM  tableRetail  WHERE  Customer\_ID IN (  SELECT  Customer\_ID  FROM  tableRetail  GROUP BY  Customer\_ID  ORDER BY  SUM(Quantity \* Price) DESC  FETCH FIRST 10 ROWS ONLY  )  GROUP BY  Customer\_ID, EXTRACT(YEAR FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY'))  ORDER BY  InvoiceYear, Rank; |

This can tell us which customers are consistently in the top 10 and how their profits have changed over the years. This can help the company identify trends and develop strategies to retain these top customers and attract new ones. The analysis can also highlight areas where the company needs to improve, such as customer service or product quality, to increase profitability.

4 - Calculate the monthly revenue for each country

|  |
| --- |
| SELECT Country, EXTRACT(MONTH FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS InvoiceMonth, SUM(Quantity \* Price) AS MonthlyRevenue  FROM tableRetail  GROUP BY Country, EXTRACT(MONTH FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY'))  ORDER BY Country, InvoiceMonth; |

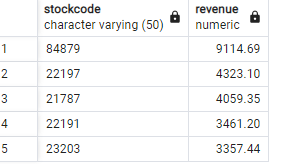
This query helps the business understand the revenue generated from each country. The business can use this information to identify potential markets for expansion and plan promotions accordingly. by identifying the data the store doesn't have any branches in any other country . so that may help them in expansion plan ,which can inform strategic decision-making around resource allocation, marketing campaigns, and sales initiatives.



5 --Find the top 5 products in terms of total revenue generated

|  |
| --- |
| SELECT DISTINCT  StockCode,  ROUND(SUM(Quantity \* Price) OVER (PARTITION BY StockCode)::numeric, 2) AS Revenue  FROM  tableRetail  ORDER BY  Revenue DESC  FETCH FIRST 5 ROWS ONLY; |

The most popular products or the products with the highest demand are not necessarily the ones generating the most revenue. The company can focus on promoting or increasing the inventory of these top revenue-generating products to maximize profits. The company can also consider raising the price of these products, as customers are willing to pay a higher price for them, which can increase the company's overall revenue.

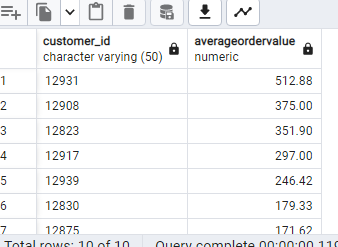


6---Find the top 10 customers with the highest average order value.

|  |
| --- |
| SELECT  Customer\_ID,  ROUND(AVG(Quantity \* Price)::numeric, 2) AS AverageOrderValue  FROM  tableRetail  GROUP BY  Customer\_ID  ORDER BY  AverageOrderValue DESC  FETCH FIRST 10 ROWS ONLY; |

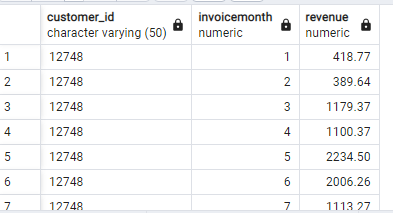
This query helps identify the top 10 customers with the highest average order value.

The business can use this information to create personalized promotions and offers for these customers to increase their loyalty.



7- This query helps identify the monthly revenue trend for the top 5 customers.

The business can use this information to plan personalized promotions and offers for these customers to increase their loyalty and engagement.

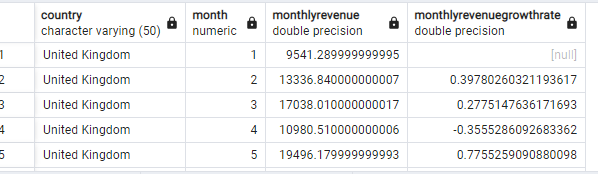


8---Find the monthly revenue growth rate for each country:

|  |
| --- |
| SELECT  Country,  EXTRACT(MONTH FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS Month,  SUM(Quantity \* Price) AS MonthlyRevenue,  (SUM(Quantity \* Price) / LAG(SUM(Quantity \* Price)) OVER (PARTITION BY Country ORDER BY EXTRACT(MONTH FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY')))) - 1 AS MonthlyRevenueGrowthRate  FROM  tableRetail  GROUP BY  Country, EXTRACT(MONTH FROM TO\_DATE(InvoiceDate, 'MM/DD/YYYY'))  ORDER BY  Country, Month; |

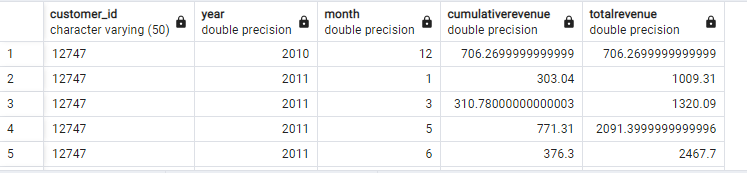
This query calculates the monthly revenue growth rate for each country.

It compares the revenue earned in the current month to the revenue earned in the previous month and calculates the percentage change. This provides a measure of how much the revenue has grown or declined month over month for each country.



9----------Calculate the cumulative revenue for each customer over time:

|  |
| --- |
| SELECT  Customer\_ID,  DATE\_PART('year', TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS Year,  DATE\_PART('month', TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS Month,  SUM(Quantity \* Price) AS CumulativeRevenue,  SUM(SUM(Quantity \* Price)) OVER (PARTITION BY Customer\_ID ORDER BY DATE\_PART('year', TO\_DATE(InvoiceDate, 'MM/DD/YYYY')), DATE\_PART('month', TO\_DATE(InvoiceDate, 'MM/DD/YYYY'))) AS TotalRevenue  FROM  tableRetail  GROUP BY  Customer\_ID, DATE\_PART('year', TO\_DATE(InvoiceDate, 'MM/DD/YYYY')), DATE\_PART('month', TO\_DATE(InvoiceDate, 'MM/DD/YYYY'))  ORDER BY  Customer\_ID, Year, Month; |



With this information, a business can identify its most valuable customers and monitor their revenue trends over time. The query could be further enhanced by including additional metrics such as customer acquisition cost and customer retention rate to gain a deeper understanding of customer behavior and identify opportunities to improve the business's financial performance. The output of this query can be used to create visualizations and reports to help businesses make informed decisions about their marketing and sales strategies, and to identify opportunities for growth and optimization.

Visualizations :

\*I have tried some

1. to show the most profitable seasons for sales

Conclusion :

In conclusion, our analysis of the retail store's sales data has provided valuable insights into the store's performance and identified areas for improvement. The information we have gathered can be used to inform decisions about product ordering, inventory management, pricing strategies, customer retention, and promotions. By continuing to analyze and monitor sales data, the retail store can make data-driven decisions that will help to improve its performance and increase profitability.

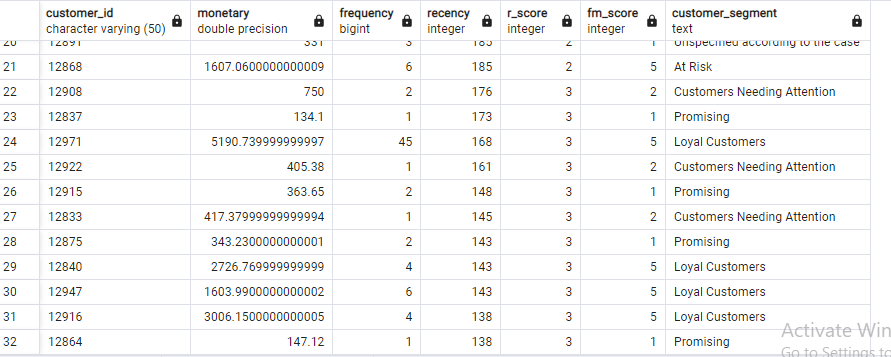
For Q2:

|  |
| --- |
| WITH rfm AS (  SELECT  Customer\_ID,  last\_purchase\_date,  Monetary,  Frequency,  (SELECT MAX(last\_purchase\_date) FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery) - last\_purchase\_date AS Recency,  CASE  WHEN (SELECT MAX(last\_purchase\_date) FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery) - last\_purchase\_date > 365 THEN 1  WHEN (SELECT MAX(last\_purchase\_date) FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery) - last\_purchase\_date > 180 THEN 2  WHEN (SELECT MAX(last\_purchase\_date) FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery) - last\_purchase\_date > 90 THEN 3  WHEN (SELECT MAX(last\_purchase\_date) FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery) - last\_purchase\_date > 60 THEN 4  ELSE 5  END AS r\_score,  CASE  WHEN ((Monetary+Frequency)/2) >= 800 THEN 5  WHEN ((Monetary+Frequency)/2) >= 600 THEN 4  WHEN ((Monetary+Frequency)/2) >= 400 THEN 3  WHEN ((Monetary+Frequency)/2) >= 200 THEN 2  ELSE 1  END AS fm\_score  FROM (  SELECT  Customer\_ID,  MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY')) AS last\_purchase\_date,  SUM(Quantity \* Price) AS Monetary,  COUNT(DISTINCT Invoice) AS Frequency  FROM  tableRetail  GROUP BY  Customer\_ID  ) subquery  )  SELECT  rfm.Customer\_ID,  rfm.Monetary,  rfm.Frequency,  rfm.Recency,  rfm.r\_score,  rfm.fm\_score,  CASE  WHEN r\_score = 5 AND fm\_score = 5 THEN 'Champions'  WHEN r\_score = 5 AND fm\_score = 4 THEN 'Champions'  WHEN r\_score = 4 AND fm\_score = 5 THEN 'Champions'  WHEN r\_score = 5 AND fm\_score = 2 THEN 'Potential Loyalists'  WHEN r\_score = 4 AND fm\_score = 2 THEN 'Potential Loyalists'  WHEN r\_score = 3 AND fm\_score = 3 THEN 'Potential Loyalists'  WHEN r\_score = 4 AND fm\_score = 3 THEN 'Potential Loyalists'  WHEN r\_score = 5 AND fm\_score = 3 THEN 'Loyal Customers'  WHEN r\_score = 4 AND fm\_score = 4 THEN 'Loyal Customers'  WHEN r\_score = 3 AND fm\_score = 5 THEN 'Loyal Customers'  WHEN r\_score = 3 AND fm\_score = 4 THEN 'Loyal Customers'  WHEN r\_score = 5 AND fm\_score = 1 THEN 'Recent Customers'  WHEN r\_score = 4 AND fm\_score = 1 THEN 'Promising'  WHEN r\_score = 3 AND fm\_score = 1 THEN 'Promising'  WHEN r\_score = 3 AND fm\_score = 2 THEN 'Customers Needing Attention'  WHEN r\_score = 2 AND fm\_score = 3 THEN 'Customers Needing Attention'  WHEN r\_score = 2 AND fm\_score = 2 THEN 'Customers Needing Attention'  WHEN r\_score = 2 AND fm\_score = 5 THEN 'At Risk'  WHEN r\_score = 2 AND fm\_score = 4 THEN 'At Risk'  WHEN r\_score = 1 AND fm\_score = 3 THEN 'At Risk'  WHEN r\_score = 1 AND fm\_score = 5 THEN 'Can’t Lose Them'  WHEN r\_score = 1 AND fm\_score = 4 THEN 'Can’t Lose Them'  WHEN r\_score = 1 AND fm\_score = 2 THEN 'Hibernating'  WHEN r\_score = 1 AND fm\_score = 1 THEN 'Lost'  ELSE 'Unspecified according to the case'  END AS customer\_segment  FROM rfm  ORDER BY rfm.recency desc ; |

The RFM analysis can provide valuable insights for businesses by identifying customer segments with different behaviors and needs. This information can help businesses optimize their marketing and sales strategies to better target and serve their customers, ultimately leading to increased customer retention and revenue.

For example, businesses can use RFM analysis to identify their most valuable customers (such as Champions or Loyal Customers) and offer them personalized promotions, rewards, or loyalty programs to encourage repeat purchases and strengthen the relationship. On the other hand, businesses can also identify customers at risk of leaving (such as At Risk or Hibernating) and take proactive actions to re-engage them, such as offering special incentives, improving customer service, or sending personalized messages to address their concerns.

Overall, RFM analysis can help businesses to understand their customer base more deeply and tailor their marketing efforts to meet the specific needs and preferences of each segment, leading to increased customer satisfaction and loyalty, and ultimately, improved business performance.



Q3:

I have created the table to load the data into it:

CREATE TABLE your\_table\_name (

Cust\_Id integer,

Calendar\_Dt date,

amount numeric(8,2)

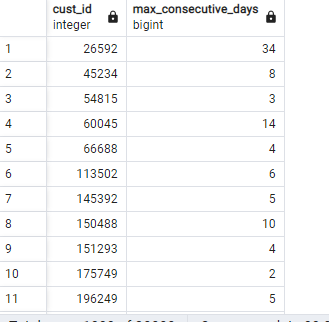
);

Then

COPY transactions TO '/path/to/transactions.csv' DELIMITER ',' CSV HEADER;

1. What is the maximum number of consecutive days a customer made purchases?

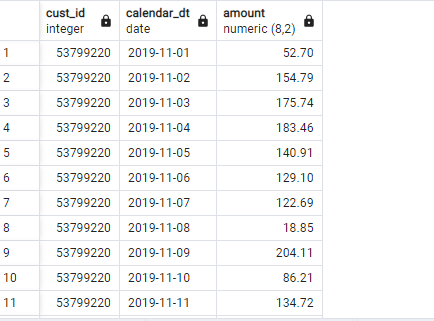
|  |
| --- |
| WITH purchases AS (  SELECT  Cust\_Id,  DATE\_TRUNC('day', Calendar\_Dt) AS purchase\_date,  LAG(DATE\_TRUNC('day', Calendar\_Dt)) OVER (PARTITION BY Cust\_Id ORDER BY DATE\_TRUNC('day', Calendar\_Dt)) AS previous\_purchase\_date  FROM transactions  ORDER BY Cust\_Id, purchase\_date ),  consecutive\_purchases AS (  SELECT  Cust\_Id,  purchase\_date,  previous\_purchase\_date,  CASE  WHEN date\_part('day', purchase\_date - previous\_purchase\_date) = 1 THEN '1'  ELSE 'More than 1 '  END AS consecutive\_days  FROM purchases  ),consecutive\_purchase\_groups AS (  SELECT  Cust\_Id,  purchase\_date,  consecutive\_days,  ROW\_NUMBER() OVER (PARTITION BY Cust\_Id ORDER BY purchase\_date) - ROW\_NUMBER() OVER (PARTITION BY Cust\_Id, consecutive\_days ORDER BY purchase\_date) AS grp  FROM consecutive\_purchases  )  SELECT  Cust\_Id,  MAX(num\_consecutive\_days) AS max\_consecutive\_days  FROM (  SELECT  Cust\_Id,  COUNT(\*) AS num\_consecutive\_days  FROM consecutive\_purchase\_groups  GROUP BY Cust\_Id, grp  ORDER BY Cust\_Id, num\_consecutive\_days DESC  ) AS subquery  GROUP BY Cust\_Id  ORDER BY Cust\_Id ; |



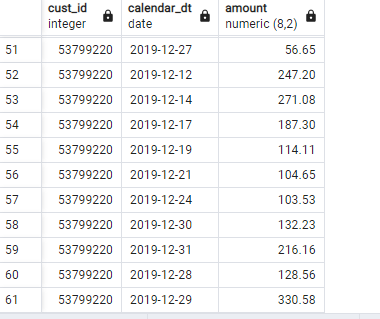
To test it :

|  |
| --- |
| SELECT \*  FROM transactions  WHERE Cust\_Id=53799220 |

The first transactions for this customer :



The last days in his data :



Another way: another logic:-

|  |
| --- |
| WITH customer\_days AS (  SELECT  Cust\_Id,  Calendar\_dt,  ROW\_NUMBER() OVER (PARTITION BY Cust\_Id ORDER BY Calendar\_dt) AS rn  FROM transactions  )  SELECT  c1.Cust\_Id,  MAX(c1.rn - c2.rn) AS max\_consecutive\_days  FROM  customer\_days c1  JOIN customer\_days c2 ON c1.Cust\_Id = c2.Cust\_Id AND c1.rn > c2.rn  GROUP BY  c1.Cust\_Id  order by max\_consecutive\_days desc |

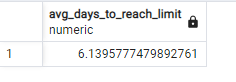
1. b- On average, How many days/transactions does it take a customer to reach a spent threshold of 250 L.E?

according to this threshold:

|  |
| --- |
| WITH prchases AS (  SELECT  Cust\_Id,  Calendar\_Dt,  SUM(amount) OVER (PARTITION BY Cust\_Id ORDER BY Calendar\_Dt) AS total\_spending  FROM  transactions  ),  spending\_below\_limit AS (  SELECT  Cust\_Id,  Calendar\_Dt,  total\_spending  FROM  prchases  WHERE  total\_spending < 250  ),  spending\_above\_limit AS (  SELECT  Cust\_Id,  Calendar\_Dt,  total\_spending  FROM  prchases  WHERE  total\_spending >= 250  ),  days\_to\_reach\_limit AS (  SELECT  Cust\_Id,  COUNT(Calendar\_Dt) AS days\_to\_reach\_limit  FROM  spending\_below\_limit  WHERE  Cust\_Id IN (SELECT Cust\_Id FROM spending\_above\_limit)  GROUP BY  Cust\_Id  ORDER BY  Cust\_Id  )  SELECT AVG(days\_to\_reach\_limit) AS avg\_days\_to\_reach\_limit  FROM days\_to\_reach\_limit; |

Query structure: The query uses common table expressions (CTEs) to calculate the total spending for each customer on each day. It then separates the spending data into two groups based on whether the total spending is below or above the threshold of $250. The query then calculates the number of days it takes for each customer to reach the spending threshold and calculates the average of this value across all customers.

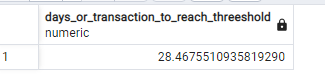
Results: The query returns a single value, which is the average number of days it takes for customers to reach the spending threshold of $250.



another logic:

|  |
| --- |
| WITH prchases AS (  SELECT  cust\_id,  calendar\_dt,  SUM(amount) OVER (PARTITION BY cust\_id ORDER BY calendar\_dt) AS total\_Prchass  FROM  transactions  ),  final\_days AS (  SELECT  cust\_id,  COUNT(calendar\_dt) AS final\_days  FROM  prchases  WHERE  total\_Prchass >= 250  GROUP BY  cust\_id  ORDER BY  cust\_id  )  SELECT  AVG(final\_days) AS days\_or\_transaction\_to\_reach\_threeshold  FROM  final\_days |

The output:



This query calculates the number of days it takes for a customer to reach the purchase threshold, the query uses a Common Table Expression (CTE) called prchases. This CTE sums up the total amount spent by each customer for each day, ordered by the transaction date.

The query then filters the results to only include days where the total spending is greater than or equal to $250. The number of days for each customer to reach the purchase threshold is calculated using the final\_days CTE.

Finally, the query calculates the average number of days it takes for customers to reach the purchase threshold by taking the average of the final\_days column in the final\_days CTE.