Purchase Transactions Report

I am pleased to share the latest report on my data analysis efforts. The report covers a comprehensive analysis of the data gathered over the past months. It highlights some key insights that I believe can help us make more informed decisions moving forward.

The report is organized into several sections, including an overview of the data, queries that answer our questions about data, and a detailed analysis of trends and patterns we have observed behind each query.

Overview:

The company in the United Kingdom records purchasing transactions of customers, I am doing some analysis so we can make insights that will help us to target the customers in the most efficient and proactive way, improve customer retention, decrease churn, and increase our sales.

Queries And Insights:

First Query:

*-- ranking total sales from every customer*

SELECT customer\_id, SUM(quantity \* price) total\_sales, RANK() OVER(ORDER BY SUM(quantity \* price) DESC) rank\_sales

FROM tableretail

GROUP BY customer\_id;

Here, I was trying to figure out how much the purchases cost every customer and rank customers based on the total amount of purchases they made (total sales) So I can know my top customers.

The customer with id = 12931 is the most valuable (entering the highest total sales to the company with about 42055 total sales from that customer).

Second Query:

*-- extracting month from the invoice date*

SELECT TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm') month,

*-- calculating monthly sales and running total sales according to month for our top customer*

SUM(quantity \* price) monthly\_sales,

SUM(SUM(quantity \* price))

OVER (ORDER BY TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm'))

running\_total

FROM tableretail

*-- filtering data based on our top customer*

WHERE customer\_id = 12931

GROUP BY TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm')

ORDER BY month;

In the previous query, I knew that the customer with id = 12931 is the most valuable customer so I investigated more information about his monthly sales and running total of his sales by month. This will help me to track my top customer.

Third Query:

*-- calculating monthly sales and monthly sales growth*

SELECT TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm') month, SUM(quantity \* price) monthly\_sales,

*-- calculating pervious month sales using lag function*

LAG(SUM(quantity \* price)) OVER (ORDER BY TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm')) previous\_month\_sales,

*-- monthly sales growth = total sales of current month - total sales of previous month*

SUM(quantity \* price) - LAG(SUM(quantity \* price)) OVER (ORDER BY TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm')) monthly\_sales\_growth

FROM tableretail

GROUP BY TO\_CHAR(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi'), 'mm')

ORDER BY month;

Now, I calculate monthly sales and their growth per month. The month that the company made the highest sales is November and the month that the company made the lowest sales is January.

In months like February, March, May, July, August, and November there was growth in the sales but in the other months, there was no growth but loss in our sales!

Fourth Query:

WITH tenure\_vs\_sales AS (

SELECT DISTINCT customer\_id, Sum(quantity \* price) OVER(PARTITION BY customer\_id) total\_sales,

*-- first purchase every customer made*

FIRST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(PARTITION BY customer\_id ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) first\_purchase,

*-- last purchase every customer made*

LAST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(PARTITION BY customer\_id ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) last\_purchase,

*-- the tenure that every customer makes his purchases on*

ROUND(LAST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(PARTITION BY customer\_id ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) -

FIRST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(PARTITION BY customer\_id ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)) tenure

FROM tableretail

)

*-- correlation between the tenure that every customer makes his purchases on and his total sales*

SELECT tenure\_vs\_sales.\*, CORR(total\_sales, tenure) OVER() corr\_tenure\_sales

FROM tenure\_vs\_sales;

Here, I see if there is a correlation between the tenure that our customers made their purchases and the total sales.

Seeing if the total sales increased when the tenure was long or not.

The correlation is about 0.42 and this value is not big so we can’t say that there is a correlation between the tenure and total sales.

Fifth Query:

*-- finding out the top 10 sold products*

WITH most\_sold\_products AS (

SELECT stockcode, SUM(quantity) number\_of\_orders

FROM tableretail

GROUP BY stockcode

ORDER BY number\_of\_orders DESC

)

SELECT \*

FROM most\_sold\_products

WHERE ROWNUM <= 10;

Here, I find out the top 10 sold products (the products that customers purchased in high amount of quantity).

The product with code 84077 is the most sold product.

Now, After exploring the data, I will implement a Monetary model for customer behavior for product purchasing and segment each customer based on these groups:

(Champions - Loyal Customers - Potential Loyalists - Recent Customers - Promising - Customers Needing Attention - At Risk - Cant Lose Them -Hibernating - Lost).

The customers will be grouped based on 3 main values:

* Recency: how recent the last transaction is (I will choose the most recent purchase in the dataset as the reference date).
* Frequency: how many times the customer has bought from our store.
* Monetary: how much each customer has paid for our products.

And based on the customer score of recency, frequency, and monetary, the customer will be in one of the previous groups.

This query below is the application of what I have said:

*-- calculating recency: diff between the last purchase overall and the last purchase by the customer*

*-- frequency: count of orders the customer has bought from our store*

*-- monetary: total purchases that every customer paid for products*

WITH recency\_frequency\_monetary AS (

SELECT DISTINCT customer\_id, ROUND(FIRST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') DESC)

- FIRST\_VALUE(TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) OVER(PARTITION BY customer\_id ORDER BY TO\_DATE(invoicedate, 'mm/dd/yyyy hh24:mi') DESC)) recency,

COUNT(\*) OVER(PARTITION BY customer\_id) frequency,

SUM(quantity \* price) OVER (PARTITION BY customer\_id) monetary

FROM tableretail

),

*-- categorizing recency, frequency, and monetary into 5 categories(scores)*

scores AS (

SELECT recency\_frequency\_monetary.\*, NTILE(5) OVER(ORDER BY recency DESC) r\_score,

NTILE(5) OVER(ORDER BY(frequency + monetary) / 2) fm\_score

FROM recency\_frequency\_monetary

)

*-- labeling our customers according to their scores*

SELECT scores.\*, CASE WHEN (r\_score = 5 AND fm\_score in (5,4)) OR (r\_score = 4 AND fm\_score = 5) then 'Champions'

WHEN (r\_score in (4,5) AND fm\_score = 2) OR (r\_score in (4,3) AND fm\_score = 3) then 'Potential Loyalists'

WHEN (r\_score = 5 AND fm\_score = 3) OR (r\_score = 4 AND fm\_score = 4) OR (r\_score = 3 AND fm\_score in (5,4)) then 'Loyal Customers'

WHEN (r\_score = 5 AND fm\_score = 1) then 'Recent Customers'

WHEN (r\_score in (4,3) AND fm\_score = 1) then 'Promising'

WHEN (r\_score = 3 AND fm\_score = 2) OR (r\_score = 2 AND fm\_score in (2,3)) then 'Customers Needing Attention'

WHEN (r\_score = 1 AND fm\_score = 3) OR (r\_score = 2 AND fm\_score in (4,5)) then 'At Risk'

WHEN (r\_score = 1 AND fm\_score in (4,5)) then 'Cant Lose Them'

WHEN (r\_score = 1 AND fm\_score = 2) then 'Hibernating'

WHEN (r\_score = 1 AND fm\_score = 1) then 'Lost'

END AS cust\_segment

FROM scores;

Now, with the new dataset, I will find the daily purchasing transactions for customers with this query:

*-- assigning a unique row number to each transaction for each customer*

WITH unique\_transaction AS (

SELECT cust\_id, calendar\_dt,

ROW\_NUMBER() OVER (PARTITION BY cust\_id ORDER BY calendar\_dt) row\_num\_transaction

FROM dailytransactions

),

*-- finding the consecutive days that the customer made purchases, assigning another row number but this time by customer id and by the diff between the unique row number and the calendar date*

cons\_days AS (

SELECT cust\_id, ROW\_NUMBER() OVER (PARTITION BY cust\_id, TO\_DATE(calendar\_dt, 'mm/dd/yyyy') - row\_num\_transaction ORDER BY calendar\_dt) consecutive\_days

FROM unique\_transaction

)

*-- finding the max number of consecutive days a customer made purchases*

SELECT cust\_id, MAX(consecutive\_days) max\_consecutive\_days

FROM cons\_days

GROUP BY cust\_id;

Here, I started by assigning a row number to each transaction of each customer so I can use it to calculate consecutive days.

Then I used another row number but this time to calculate the number of consecutive days for each transaction partitioned by cust\_id and the difference between the calendar\_dt and the unique row number, this difference is used to group consecutive transactions by their date sequence.

In the final, I grouped the result by cust\_id so I can get the maximum number of consecutive days for each customer.

To find the number of days/transactions it takes a customer to reach a spent threshold of 250 L.E on average, I used this query:

*-- calculating the average number of transactions it takes a customer to reach a spent threshold of 250 L.E*

*-- I assumed that a customer could reach the 250 L.E threshold only once and that the threshold could be reached*

*-- either by a single transaction or multiple transactions*

threshold of 250 L.E

WITH num\_transactions AS (

SELECT cust\_id, MIN(TO\_DATE(calendar\_dt, 'mm/dd/yyyy')) AS first\_date, COUNT(\*) AS num\_days

FROM dailytransactions

WHERE amt\_LE >= 250

GROUP BY cust\_id

)

SELECT AVG(num\_days) AS avg\_days\_to\_reach\_threshold

FROM num\_transactions ;

In this query, first I grouped the transactions by customer and finding the earliest date on which each customer made a purchase of 250 L.E or more (which I called their "first purchase").

It then counts the number of days between the first purchase and subsequent purchases that meet the 250 L.E threshold.

The outer query then calculates the average number of days/transactions it takes for a customer to reach the 250 L.E.

Note: I assumed that a customer could reach the 250 L.E threshold only once and that the threshold could be reached either by a single transaction or multiple transactions.

In conclusion, I would like to thank you for the opportunity to present this report. I appreciate your attention and interest in the topic, and I look forward to any feedback or questions you may have.