3/5/2024

Analytical SQL

case study

Hassan Hosny Hassan

# Presenting an intricately crafted retail dataset dashboard that not only showcases visually stunning insights but also delves into the nuanced layers of information, providing a comprehensive and profound understanding of the intricate dynamics at play within the retail landscape.

A close-up of a data dashboard

Description automatically generated

# **Analytical SQL Case Study**

Background: Customers has purchasing transaction that we shall be monitoring to get intuition behind each customer behavior to target the customers in the most efficient and proactive way, to increase sales/revenue , improve customer retention and decrease churn. You will be given a dataset, and you will be required to answer using SQL Analytical functions you have learnt in the course.

Q1- Using OnlineRetail dataset

• write at least 5 analytical SQL queries that tells a story about the data

• write small description about the business meaning behind each query

1. Each customer with his total number of invoices

select customer\_id, count(distinct invoice) as invoice\_counts

from tableretail

group by customer\_id

order by invoice\_counts desc;

A screenshot of a computer

Description automatically generated

1. Showing the Popular Products (Top 20)

-- Popular Products (Top 20)

select \*

from

(

select stockcode, sum(quantity) product\_counts

from tableretail

group by stockcode

order by product\_counts desc

)

where rownum <= 20;

A screenshot of a number

Description automatically generated

1. Showing each invoice\_id with its amount of money in which date and the customer who made it to know who made the highest invoices

select customer\_id, invoice, sum(quantity\*price) sales, to\_date(invoicedate, 'MM-DD-YYYY HH24:MI') as "Date"

from tableretail

group by customer\_id, invoice,invoicedate

order by sales desc;

A screenshot of a data

Description automatically generated

1. Showing each product and sum of quantity that have been sold and total sales for each product.

select stockcode, quantity, sum(sales) as product\_sell\_price

from

(

select stockcode, sum(quantity) quantity, sum(quantity\*price) sales

from tableretail

group by stockcode

order by sales desc

)

group by stockcode, quantity

order by product\_sell\_price desc;

A screenshot of a data

Description automatically generated

1. Customer Segmentation Analysis to show how much each customer pays for the invoices and get the rank for each customer based on total sales for each customer.

* Note: it doesn’t matter how many invoices customers make, but it matters how much they pay for each invoice.

-- Customer Segmentation Analysis

with cust as(

select

customer\_id,

count(distinct invoice) invoice\_count,

sum(quantity\*price) total\_spending

from tableretail

group by customer\_id

order by total\_spending desc)

select customer\_id, invoice\_count, total\_spending,

rank() over(order by total\_spending desc) as ranking\_by\_sales

from cust;

A screenshot of a table

Description automatically generated

1. Showing each customer with corresponding sales for each invoice over its date and also cumulative total sales for each customer until a specific invoice.

select customer\_id, invoicedate, cust\_sales,

sum(cust\_sales) over(partition by customer\_id order by invoicedate) as running\_total

from

(

select

customer\_id,

to\_date(invoicedate, 'MM-DD-YYYY HH24:MI') as invoicedate,

sum(quantity\*price) cust\_sales

from tableretail

group by customer\_id, to\_date(invoicedate, 'MM-DD-YYYY HH24:MI')

order by customer\_id, invoicedate

);

A screenshot of a data

Description automatically generated

1. Showing each stock\_code with corresponding quantity over a specific date and also cumulative total quantities for each stock\_code until a specific invoice\_date.

select stockcode, invoicedate, count\_products\_sold,

sum(count\_products\_sold) over(partition by stockcode order by invoicedate) running\_total

from

(

select stockcode, to\_date(invoicedate, 'MM-DD-YYYY HH24:MI') as invoicedate, sum(quantity) as count\_products\_sold

from tableretail

group by stockcode, to\_date(invoicedate, 'MM-DD-YYYY HH24:MI')

order by stockcode, invoicedate

);

A screenshot of a data

Description automatically generated

1. Showing each day with its sales and ordering them from the heights, in order to know in which days we achieve a lot of sales.

select distinct to\_date(invoicedate, 'MM-DD-YYYY HH24:MI') as invoicedate,

sum(quantity\*price) over(partition by to\_date(invoicedate, 'MM-DD-YYYY HH24:MI')) sales\_per\_day

from tableretail

order by sales\_per\_day desc;

A screenshot of a calendar

Description automatically generated

1. Showing each invoice with the corresponding stock\_code have been bought on it and the deviation of the price from the average price for each stock code.

select invoice, stockcode, price, avg\_price,

-- calculate the deviation of the price from the average price for each stockcode.

round(price - avg\_price, 2) as price\_deviation,

dense\_rank() over(order by price) as ranking

from(

select invoice, stockcode, price,

round(avg(price) over(partition by stockcode),2) as avg\_price

from tableretail);

A screenshot of a computer

Description automatically generated

Q2- After exploring the data now you are required to implement a Monetary model for customers behavior for product purchasing and segment each customer based on the below 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 (**Hint**: choose a reference date, which is the most recent purchase in the dataset )
* **Frequency** => how many times the customer has bought from our store
* **Monetary** => how much each customer has paid for our products

As there are many groups for each of the R, F, and M features, there are also many potential permutations, this number is too much to manage in terms of marketing strategies.

For this, we would decrease the permutations **by getting the average scores of the frequency and monetary** (as both of them are indicative to purchase volume anyway)

# Expected outcome

A screenshot of a data sheet

Description automatically generated

Label each customer based on the below values

A group of people's group of people

Description automatically generated with medium confidence

The dataset used in this context is identical to the one in the first question. I conducted analyses on customer recency, frequency, and monetary metrics to classify them into distinct segments. This segmentation helps determine the most appropriate approach for interacting with each customer based on their respective segment.

WITH customer\_sales as (

select

customer\_id,

sum(price \* quantity) as sum\_of\_sales

from tableRetail

group by customer\_id),

RFM as (

select distinct customer\_id,

round(max(to\_date(invoicedate, 'MM-DD-YYYY HH24:MI')) over() - max(to\_date(invoicedate, 'MM-DD-YYYY HH24:MI')) over(partition by customer\_id)) as recency,

count(invoice) over(partition by customer\_id) as frequency,

round(sum((quantity\*price)/1000) over(partition by customer\_id), 2) as monetary

from tableretail),

RFM\_scores as (

select customer\_id, recency, frequency, monetary,

ntile(5) over (order by recency) as r\_score,

ntile(5) over (order by frequency+monetary) as fm\_score

from RFM)

select customer\_id, recency, frequency, monetary, r\_score, fm\_score,

CASE

WHEN R\_SCORE=5 AND FM\_SCORE=5 THEN 'Champion'

WHEN R\_SCORE=5 AND FM\_SCORE=4 THEN 'Champion'

WHEN R\_SCORE=4 AND FM\_SCORE=5 THEN 'Champion'

WHEN R\_SCORE=5 AND FM\_SCORE=2 THEN 'Potential Loyalist'

WHEN R\_SCORE=4 AND FM\_SCORE=2 THEN 'Potential Loyalist'

WHEN R\_SCORE=3 AND FM\_SCORE=3 THEN 'Potential Loyalist'

WHEN R\_SCORE=4 AND FM\_SCORE=3 THEN 'Potential Loyalist'

WHEN R\_SCORE=5 AND FM\_SCORE=3 THEN 'Loyal Customer'

WHEN R\_SCORE=4 AND FM\_SCORE=4 THEN 'Loyal Customer'

WHEN R\_SCORE=3 AND FM\_SCORE=5 THEN 'Loyal Customer'

WHEN R\_SCORE=3 AND FM\_SCORE=4 THEN 'Loyal Customer'

WHEN R\_SCORE=5 AND FM\_SCORE=1 THEN 'Recent Customer'

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 'Cant Lose Them'

WHEN R\_SCORE=1 AND FM\_SCORE=4 THEN 'Cant Lose Them'

WHEN R\_SCORE=1 AND FM\_SCORE=2 THEN 'Hibernating'

WHEN R\_SCORE=1 AND FM\_SCORE=1 THEN 'Lost'

ELSE 'Other'

END AS "CUSTOMER GROUP" from RFM\_scores ;

A screenshot of a computer

Description automatically generated

Q.3

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

# Expected Output

A black and white rectangular object with numbers

Description automatically generated

with CTE as(

select cust\_id, calendar\_dt,

calendar\_dt - row\_number() over (partition by cust\_id order by calendar\_dt) as date\_check

from cust\_transactions

)

select cust\_id, max(counts) as max\_consecutive\_days

from

(

select cust\_id, count(date\_check) counts, min(calendar\_dt) min\_date, max(calendar\_dt) max\_date

from cte

group by cust\_id, date\_check

)

group by cust\_id;

A screenshot of a computer

Description automatically generated

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

with CTE as(

select cust\_id, calendar\_dt, amt\_le,

sum(amt\_le) over(partition by cust\_id order by calendar\_dt) as cum\_total\_spend,

row\_number() over(partition by cust\_id order by calendar\_dt) as days\_count

from cust\_transactions),

CTE2 as(

select

cust\_id,

cum\_total\_spend,

days\_count

from CTE

where cum\_total\_spend >= 250),

average as(

select

cust\_id,

min(days\_count) as counts

from CTE2

group by cust\_id

order by counts desc)

select

round(avg(counts)) as avg\_days\_reach\_250LE

from average;

A screenshot of a computer

Description automatically generated