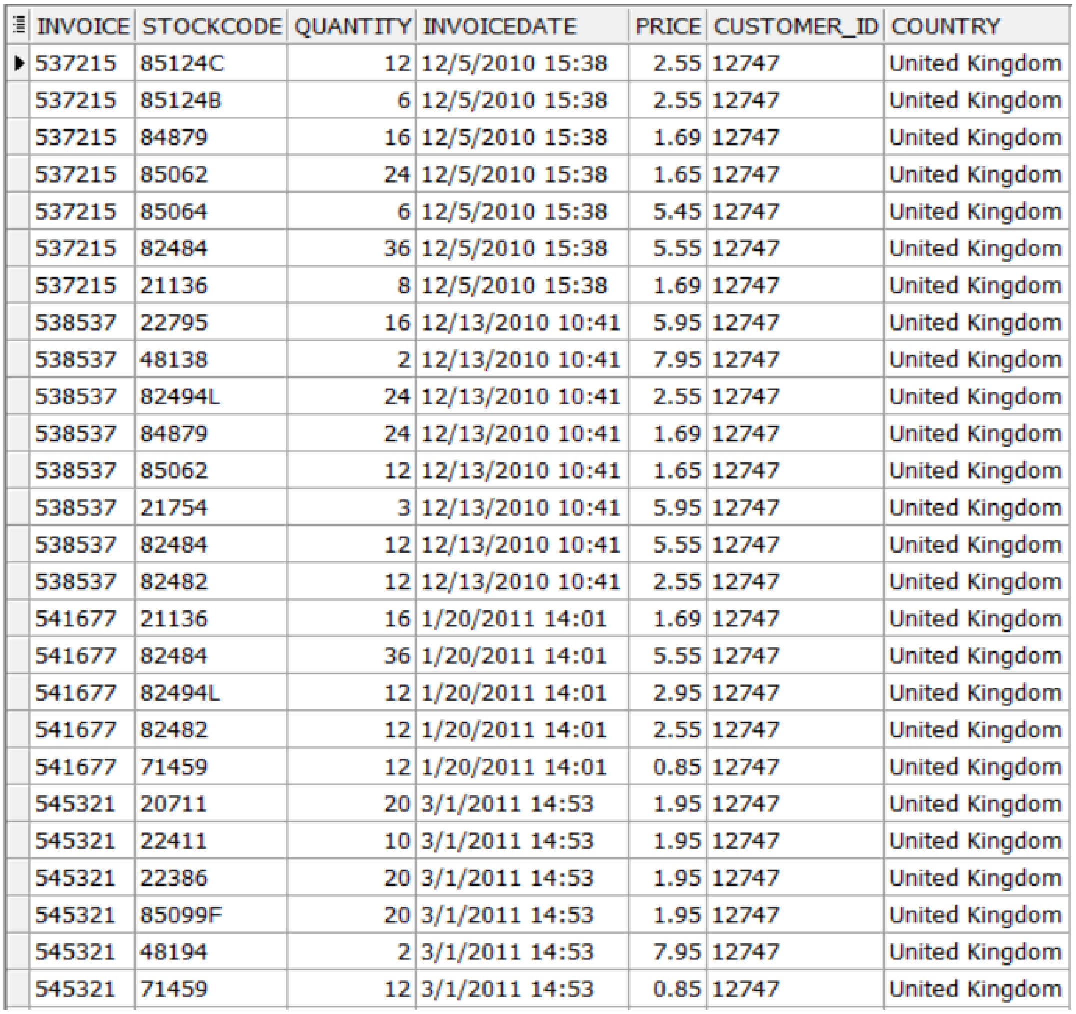
# ANALYTICAL SQL CASE STUDY

ADHAM AYMAN ELSAYED

ITI

## Description of Data:

- **STOCKCODE:** This column likely represents a unique identifier or code for a particular stock item or product sold.
- **QUANTITY:** This column indicates the quantity of the stock item or product sold in a particular transaction.
- **PRICE:** This column denotes the price per unit of the stock item or product sold. It represents the amount of money charged for each unit of the item.
- **INVOICEDATE:** This column contains the date when the transaction was invoiced or recorded.
- **INVOICE:** This column likely contains a unique identifier or code for each invoice generated for a transaction.
- **CUSTOMER_ID:** This column likely represents a unique identifier or code assigned to each customer who made a purchase.
- **COUNTRY:** This column indicates the country associated with the customer or the location where the transaction took place.



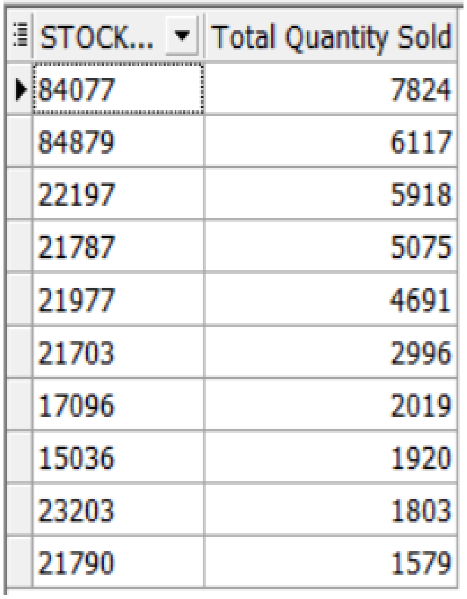
| INVOICE | STOCKCODE | QUANTITY | INVOICEDATE | PRICE | CUSTOMER_ID | COUNTRY |
|---|---|---|---|---|---|---|
| 537215 | 85124C | 12 | 12/5/2010 15:38 | 2.55 | 12747 | United Kingdom |
| 537215 | 85124B | 6 | 12/5/2010 15:38 | 2.55 | 12747 | United Kingdom |
| 537215 | 84879 | 16 | 12/5/2010 15:38 | 1.69 | 12747 | United Kingdom |
| 537215 | 85062 | 24 | 12/5/2010 15:38 | 1.65 | 12747 | United Kingdom |
| 537215 | 85064 | 6 | 12/5/2010 15:38 | 5.45 | 12747 | United Kingdom |
| 537215 | 82484 | 36 | 12/5/2010 15:38 | 5.55 | 12747 | United Kingdom |
| 537215 | 21136 | 8 | 12/5/2010 15:38 | 1.69 | 12747 | United Kingdom |
| 538537 | 22795 | 16 | 12/13/2010 10:41 | 5.95 | 12747 | United Kingdom |
| 538537 | 48138 | 2 | 12/13/2010 10:41 | 7.95 | 12747 | United Kingdom |
| 538537 | 82494L | 24 | 12/13/2010 10:41 | 2.55 | 12747 | United Kingdom |
| 538537 | 84879 | 24 | 12/13/2010 10:41 | 1.69 | 12747 | United Kingdom |
| 538537 | 85062 | 12 | 12/13/2010 10:41 | 1.65 | 12747 | United Kingdom |
| 538537 | 21754 | 3 | 12/13/2010 10:41 | 5.95 | 12747 | United Kingdom |
| 538537 | 82484 | 12 | 12/13/2010 10:41 | 5.55 | 12747 | United Kingdom |
| 538537 | 82482 | 12 | 12/13/2010 10:41 | 2.55 | 12747 | United Kingdom |
| 541677 | 21136 | 16 | 1/20/2011 14:01 | 1.69 | 12747 | United Kingdom |
| 541677 | 82484 | 36 | 1/20/2011 14:01 | 5.55 | 12747 | United Kingdom |
| 541677 | 82494L | 12 | 1/20/2011 14:01 | 2.95 | 12747 | United Kingdom |
| 541677 | 82482 | 12 | 1/20/2011 14:01 | 2.55 | 12747 | United Kingdom |
| 541677 | 71459 | 12 | 1/20/2011 14:01 | 0.85 | 12747 | United Kingdom |
| 545321 | 20711 | 20 | 3/1/2011 14:53 | 1.95 | 12747 | United Kingdom |
| 545321 | 22411 | 10 | 3/1/2011 14:53 | 1.95 | 12747 | United Kingdom |
| 545321 | 22386 | 20 | 3/1/2011 14:53 | 1.95 | 12747 | United Kingdom |
| 545321 | 85099F | 20 | 3/1/2011 14:53 | 1.95 | 12747 | United Kingdom |
| 545321 | 48194 | 2 | 3/1/2011 14:53 | 7.95 | 12747 | United Kingdom |
| 545321 | 71459 | 12 | 3/1/2011 14:53 | 0.85 | 12747 | United Kingdom |

# Question One:

## 1 - Top Selling Products

This query aims to retrieve the top 10 best-selling products based on the total quantity sold. The query first calculates the total quantity sold for each product, ranks them based on the quantity sold, and then selects the top 10 products based on their ranking. This provides a list of the top 10 best-selling products along with their total quantity sold.

```
select stockcode , "Total Quantity Sold"
from (
        select stockcode , "Total Quantity Sold" , rank() over(order by "Total Quantity Sold" desc)
as top
        from (
                select distinct stockcode , sum(quantity) over(partition by stockcode) "Total Quantity
Sold"
                from tableretail))
     where top <= 10 ;
```



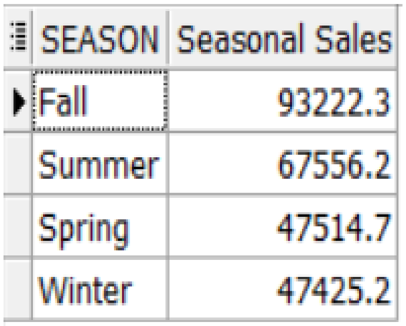
| STOCK... | Total Quantity Sold |
|---|---|
| 84077 | 7824 |
| 84879 | 6117 |
| 22197 | 5918 |
| 21787 | 5075 |
| 21977 | 4691 |
| 21703 | 2996 |
| 17096 | 2019 |
| 15036 | 1920 |
| 23203 | 1803 |
| 21790 | 1579 |

## 2 - Seasonal Sales Analysis

This query calculates the average seasonal sales for each season and orders the results based on the average seasonal sales in descending order. this query provides insights into the average sales performance for each season, allowing for analysis and comparison of seasonal trends in sales data.

```
select season , round(avg("Seasonal Sales"),1) as "Seasonal Sales"
from (
        select season , sum(quantity * price) over(partition by season) as "Seasonal Sales"
        from(
                select quantity , price ,
                case
                    when to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'mm') in (12,1,2)
then 'Winter'
                    when to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'mm') in (3,4,5) then
'Spring'
                    when to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'mm') in (6,7,8) then
'Summer'
                    else 'Fall'
                    end as season

                from tableretail))
group by season
order by "Seasonal Sales" desc;
```



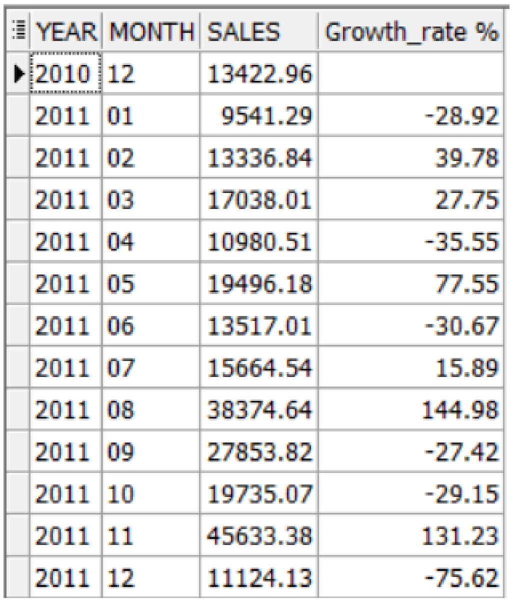
| SEASON | Seasonal Sales |
|---|---|
| Fall | 93222.3 |
| Summer | 67556.2 |
| Spring | 47514.7 |
| Winter | 47425.2 |

### 3 - Monthly Growth Rate

This query calculates the growth rate percentage of sales on a monthly basis.

- The final result provides the year, month, sales, and growth rate percentage for each month compared to the previous month's sales.
- This allows for the analysis of monthly sales trends and the identification of periods of growth or decline.

```
with monthly_sales as (
select
        to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'yyyy') as Year ,
        to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'mm') as Month,
        sum(quantity * price) as Sales
        from tableretail
 group by  to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'yyyy') ,
to_char(to_date(invoicedate ,'mm/dd/yyyy HH24:MI'), 'mm')
                    ) ,
per_month_sales as (
select year , month , sales ,
        lag(sales) over (order by year , month) as previous_sales
from monthly_sales
)
select year , month , sales , round((sales - previous_sales) / previous_sales * 100 , 2) as
"Growth_rate %"
from per_month_sales ;
```



| YEAR | MONTH | SALES | Growth_rate % |
|---|---|---|---|
| 2010 | 12 | 13422.96 | |
| 2011 | 01 | 9541.29 | -28.92 |
| 2011 | 02 | 13336.84 | 39.78 |
| 2011 | 03 | 17038.01 | 27.75 |
| 2011 | 04 | 10980.51 | -35.55 |
| 2011 | 05 | 19496.18 | 77.55 |
| 2011 | 06 | 13517.01 | -30.67 |
| 2011 | 07 | 15664.54 | 15.89 |
| 2011 | 08 | 38374.64 | 144.98 |
| 2011 | 09 | 27853.82 | -27.42 |
| 2011 | 10 | 19735.07 | -29.15 |
| 2011 | 11 | 45633.38 | 131.23 |
| 2011 | 12 | 11124.13 | -75.62 |

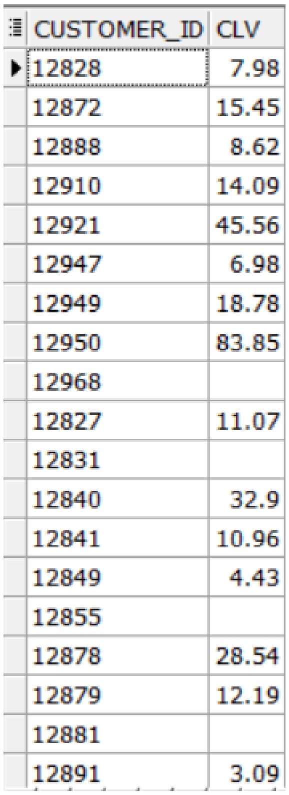
## 4 - Customer Lifetime Value

This query calculates the Customer Lifetime Value (CLV) for each distinct customer based on their total amount spent divided by the time span between their first and last purchase dates. The final result provides the CLV for each distinct customer.

CLV represents the predicted revenue a customer will generate throughout their relationship with the business, based on their historical purchasing behavior.

This metric helps businesses understand the long-term value of their customers and can inform marketing and customer retention strategies.

```
        select distinct customer_id , round(total_amount / nullif((last_date - first_date),0) ,  2)
as CLV
from(
      select customer_id , sum(price*quantity) over(partition by customer_id) as total_amount ,
         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')
range between unbounded preceding and unbounded following) as last_date ,
         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'))
as first_date
      from tableretail) ;
```

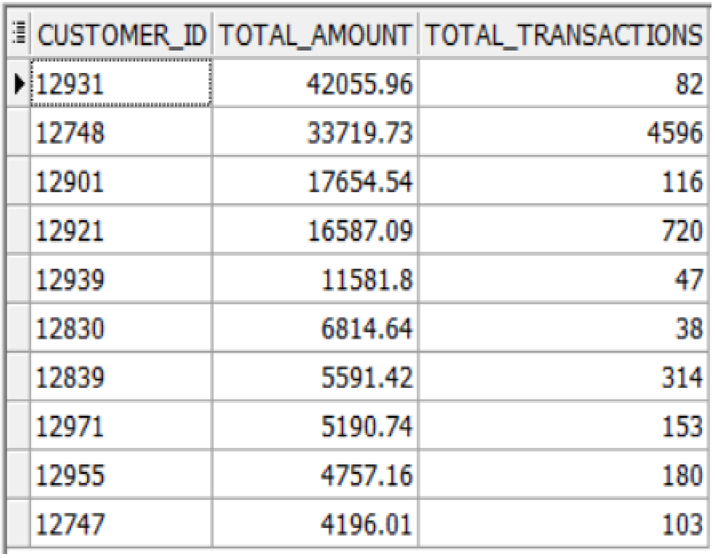


| CUSTOMER_ID | CLV |
|---|---|
| 12828 | 7.98 |
| 12872 | 15.45 |
| 12888 | 8.62 |
| 12910 | 14.09 |
| 12921 | 45.56 |
| 12947 | 6.98 |
| 12949 | 18.78 |
| 12950 | 83.85 |
| 12968 | |
| 12827 | 11.07 |
| 12831 | |
| 12840 | 32.9 |
| 12841 | 10.96 |
| 12849 | 4.43 |
| 12855 | |
| 12878 | 28.54 |
| 12879 | 12.19 |
| 12881 | |
| 12891 | 3.09 |

## 5 - Top 10 customers

This query aims to retrieve information about the top 10 customers based on their total amount spent and the total number of transactions they've made.

```
with top_cust as(
        select distinct customer_id , sum(quantity * price) over(partition by customer_id ) as
total_amount ,
        count(invoice) over(partition by customer_id) as total_transactions
        from tableretail )
select customer_id , total_amount  ,  total_transactions
from  (
        select customer_id , total_amount , total_transactions , row_number() over(order by
total_amount desc) as rank
        from top_cust)
where rank <= 10 ;
```



| CUSTOMER_ID | TOTAL_AMOUNT | TOTAL_TRANSACTIONS |
|---|---|---|
| 12931 | 42055.96 | 82 |
| 12748 | 33719.73 | 4596 |
| 12901 | 17654.54 | 116 |
| 12921 | 16587.09 | 720 |
| 12939 | 11581.8 | 47 |
| 12830 | 6814.64 | 38 |
| 12839 | 5591.42 | 314 |
| 12971 | 5190.74 | 153 |
| 12955 | 4757.16 | 180 |
| 12747 | 4196.01 | 103 |

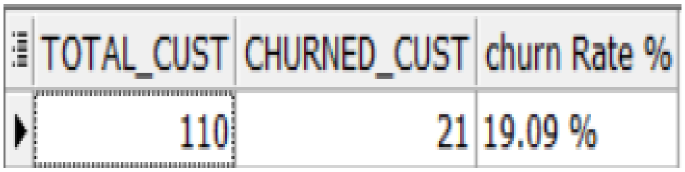
## 6 – Churn Rate

The query you provided calculates the churn rate percentage. Customers with a duration of at least 6 months are considered churned.

- Total_Cust: Total count of distinct customers in the tableretail table.
- Churned_Cust: Count of churned customers based on the provided duration criteria.
- Churn Rate %: Churn rate percentage calculated by dividing the count of churned customers by the total count of customers, multiplied by 100 and rounded to two decimal places.

```
with cust_dur as (
select count(customer_ID) as Churned_Cust
from (
        select customer_id , trunc(months / 30 ) as duration
        from (
                select distinct customer_id ,
                (last_value (to_date(invoicedate,'mm/dd/yyyy HH24:MI'))
                over (order by to_date(invoicedate,'mm/dd/yyyy HH24:MI') range between
unbounded preceding and unbounded following)
                -
                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')
range between unbounded preceding and unbounded following)) as Months
                from tableretail)
        where months / 30 >=6 )
),
total_cust as (
select count(distinct customer_id) as Total_Cust
from tableretail
)

select Total_Cust, Churned_Cust ,round((Churned_Cust / Total_Cust )*100, 2 ) || ' %'  "churn
Rate %"
from cust_dur , total_cust ;
```



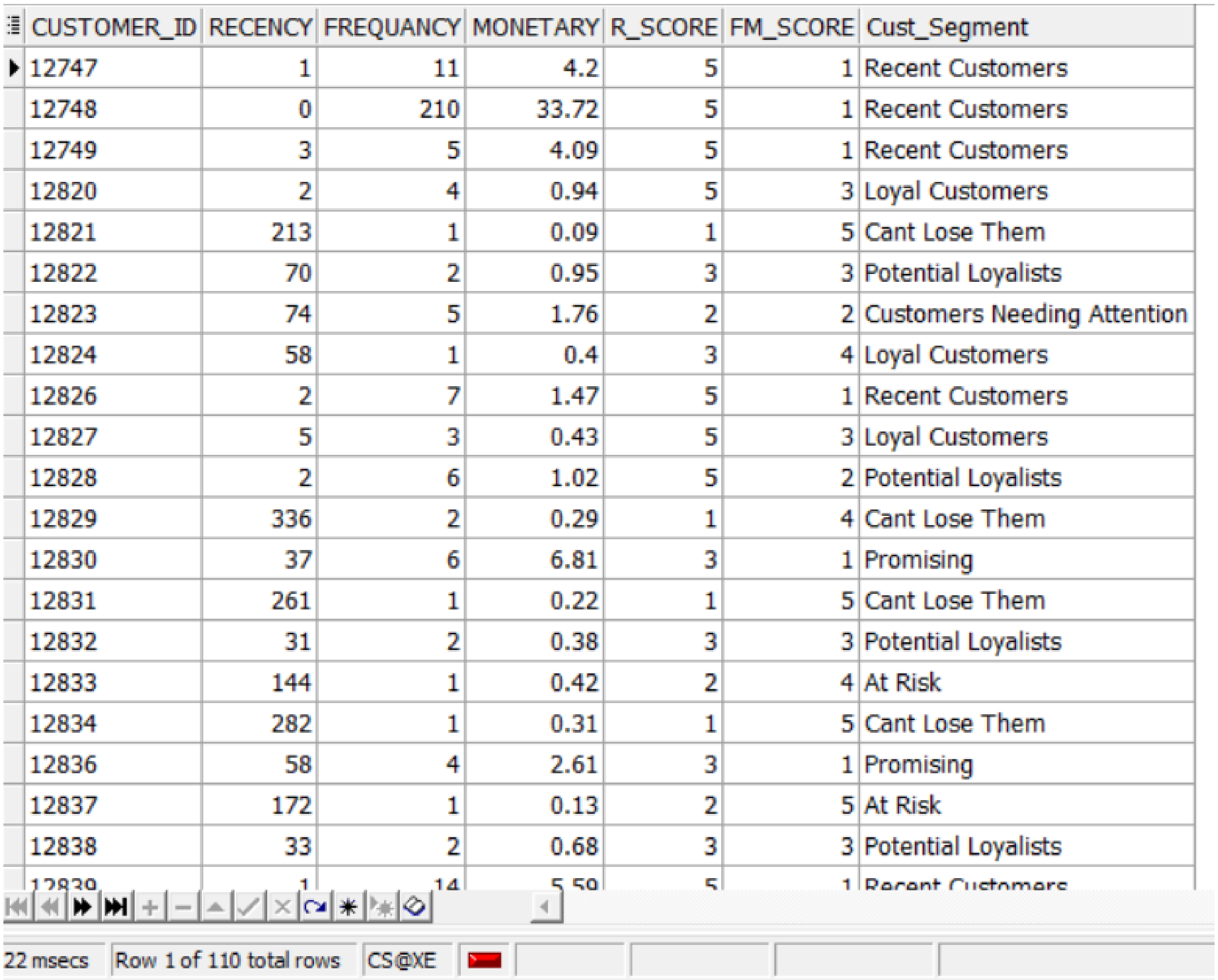
| TOTAL_CUST | CHURNED_CUST | churn Rate % |
|---|---|---|
| 110 | 21 | 19.09 % |

## Question Two:

This query segments customers based on the RFM (Recency, Frequency, Monetary) analysis and assigns them to specific customer segments. This query helps businesses understand their customer base better by segmenting customers into different groups based on their purchasing behavior, allowing for targeted marketing strategies and personalized approaches to customer management.

```
with rfm as(
select customer_id , recency , frequancy , monetary , R_score , trunc((F_score + M_score)/2)
as FM_Score
from
    (
    select customer_id , recency , frequancy , monetary , ntile (5) over (order by recency desc)
as R_score ,
    ntile(5) over(order by  Frequancy desc) as F_score , ntile(5) over(order by  monetary desc)
as M_score
    from (
        select distinct customer_id ,
            trunc(last_value (to_date(invoicedate,'mm/dd/yyyy HH24:MI'))
            over (order by to_date(invoicedate,'mm/dd/yyyy HH24:MI') range between unbounded
preceding and unbounded following)
             -
            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')
range between unbounded preceding and unbounded following)) as Recency ,
            count(distinct invoice) over(partition by customer_id  ) as Frequancy ,
            round((sum(quantity * price) over(partition by customer_id)) / 1000 , 2) as Monetary
            from tableretail ))
 order by customer_id  )
select customer_id , recency , frequancy , monetary , R_score ,FM_Score ,
case
            when R_Score = 5 and FM_Score in (5, 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 in (2 , 3) then 'Potential Loyalists'
            when R_Score = 3 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 in (4 , 5) 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 in (2, 3) then 'Customers Needing Attention'
            when R_Score = 1 and FM_Score = 3 then 'At Risk'
            when R_Score = 2 and FM_Score in (4, 5) then 'At Risk'
            when R_Score = 1 and FM_Score = 2 then 'Hibernating'
```

```
          when R_Score = 1 and FM_Score in (4, 5) then 'Cant Lose Them'
          when R_Score = 1 and FM_Score = 1 then 'Lost'
          else 'Undefined'
          end as "Cust_Segment"
from rfm;
```



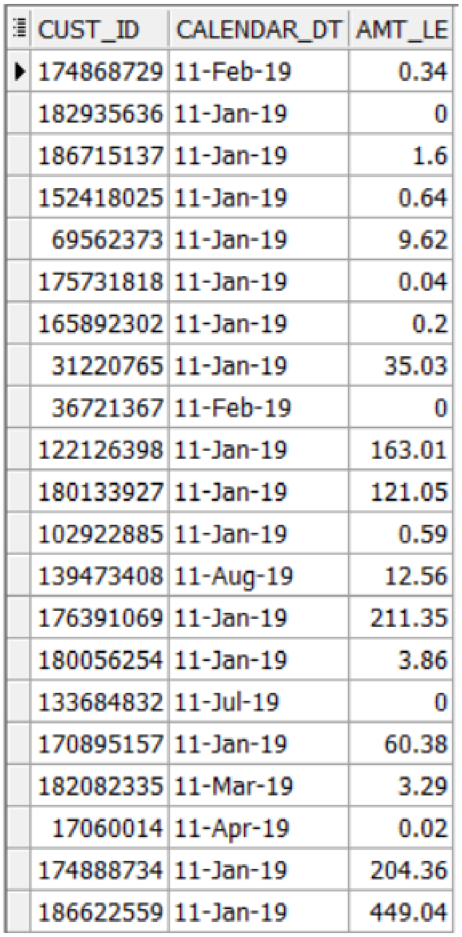
| CUSTOMER_ID | RECENCY | FREQUANCY | MONETARY | R_SCORE | FM_SCORE | Cust_Segment |
|---|---|---|---|---|---|---|
| 12747 | 1 | 11 | 4.2 | 5 | 1 | Recent Customers |
| 12748 | 0 | 210 | 33.72 | 5 | 1 | Recent Customers |
| 12749 | 3 | 5 | 4.09 | 5 | 1 | Recent Customers |
| 12820 | 2 | 4 | 0.94 | 5 | 3 | Loyal Customers |
| 12821 | 213 | 1 | 0.09 | 1 | 5 | Cant Lose Them |
| 12822 | 70 | 2 | 0.95 | 3 | 3 | Potential Loyalists |
| 12823 | 74 | 5 | 1.76 | 2 | 2 | Customers Needing Attention |
| 12824 | 58 | 1 | 0.4 | 3 | 4 | Loyal Customers |
| 12826 | 2 | 7 | 1.47 | 5 | 1 | Recent Customers |
| 12827 | 5 | 3 | 0.43 | 5 | 3 | Loyal Customers |
| 12828 | 2 | 6 | 1.02 | 5 | 2 | Potential Loyalists |
| 12829 | 336 | 2 | 0.29 | 1 | 4 | Cant Lose Them |
| 12830 | 37 | 6 | 6.81 | 3 | 1 | Promising |
| 12831 | 261 | 1 | 0.22 | 1 | 5 | Cant Lose Them |
| 12832 | 31 | 2 | 0.38 | 3 | 3 | Potential Loyalists |
| 12833 | 144 | 1 | 0.42 | 2 | 4 | At Risk |
| 12834 | 282 | 1 | 0.31 | 1 | 5 | Cant Lose Them |
| 12836 | 58 | 4 | 2.61 | 3 | 1 | Promising |
| 12837 | 172 | 1 | 0.13 | 2 | 5 | At Risk |
| 12838 | 33 | 2 | 0.68 | 3 | 3 | Potential Loyalists |
| 12839 | 1 | 14 | 5.59 | 5 | 1 | Recent Customers |

22 msecs | Row 1 of 110 total rows | CS@XE

## Question Three:

## Description of Data:

- **CALENDAR_DT:** This column represents a date or timestamp associated with a particular event or transaction.
- **CUST_ID:** This column represents a unique identifier or code assigned to each customer.
- **AMT_LE**: it seems to represent a monetary value or amount associated with each event or transaction. It could be the amount spent by a customer or the amount of a loan, depending on the context of the dataset.



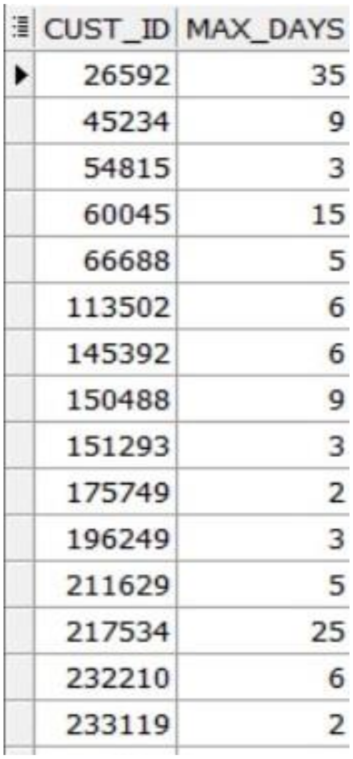
| CUST_ID | CALENDAR_DT | AMT_LE |
|---|---|---|
| 174868729 | 11-Feb-19 | 0.34 |
| 182935636 | 11-Jan-19 | 0 |
| 186715137 | 11-Jan-19 | 1.6 |
| 152418025 | 11-Jan-19 | 0.64 |
| 69562373 | 11-Jan-19 | 9.62 |
| 175731818 | 11-Jan-19 | 0.04 |
| 165892302 | 11-Jan-19 | 0.2 |
| 31220765 | 11-Jan-19 | 35.03 |
| 36721367 | 11-Feb-19 | 0 |
| 122126398 | 11-Jan-19 | 163.01 |
| 180133927 | 11-Jan-19 | 121.05 |
| 102922885 | 11-Jan-19 | 0.59 |
| 139473408 | 11-Aug-19 | 12.56 |
| 176391069 | 11-Jan-19 | 211.35 |
| 180056254 | 11-Jan-19 | 3.86 |
| 133684832 | 11-Jul-19 | 0 |
| 170895157 | 11-Jan-19 | 60.38 |
| 182082335 | 11-Mar-19 | 3.29 |
| 17060014 | 11-Apr-19 | 0.02 |
| 174888734 | 11-Jan-19 | 204.36 |
| 186622559 | 11-Jan-19 | 449.04 |

### A- Maximum number of consecutive days a customer made purchases.

The query appears to be aimed at calculating the maximum number of consecutive days that each customer has made transactions. The query calculates, for each customer, the maximum number of consecutive days on which transactions occurred. This information can be valuable for analyzing customer behavior patterns, such as how frequently and consistently customers make transactions over time.

```
with diff_date as(
select cust_id,  Calendar_Dt, Calendar_Dt  - "rank" AS date_diff
from (
        select cust_id , Calendar_Dt , row_number() over(partition by cust_id  order by
calendar_dt) "rank"
        from customertransaction)
)

select cust_id , max(consecutive_days) as max_days
from(
        select cust_id , count(date_diff) as consecutive_days
        from diff_date
        group by cust_id , date_diff
        )
group by cust_id
   order by cust_id ;
```

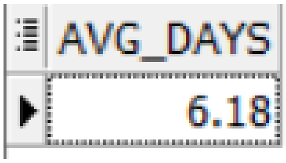


| CUST_ID | MAX_DAYS |
|---|---|
| 26592 | 35 |
| 45234 | 9 |
| 54815 | 3 |
| 60045 | 15 |
| 66688 | 5 |
| 113502 | 6 |
| 145392 | 6 |
| 150488 | 9 |
| 151293 | 3 |
| 175749 | 2 |
| 196249 | 3 |
| 211629 | 5 |
| 217534 | 25 |
| 232210 | 6 |
| 233119 | 2 |

### B - Average days does it take a customer to reach a spent threshold of 250 L.E

The query seems to calculate the average number of days it takes for customers to accumulate a total amount of at least 250 LE across their transactions. This information can be useful for understanding customer spending patterns and determining thresholds for targeted marketing or loyalty programs.

```
with rank_amt as(
select cust_id ,total_amt , row_number() over(partition by cust_id order by total_amt) as rank
from (
        select cust_id , sum(amt_le) over(partition by cust_id order by calendar_dt) as total_amt
        from customertransaction
        )
)
select round(avg(count_cust_days),2) as avg_days
from( select cust_id ,
         min(case when total_amt >= 250 then rank end ) as count_cust_days
        from  rank_amt
        group by cust_id ) ;
```



| AVG_DAYS |
|---|
| 6.18 |