Analytical SQL Project

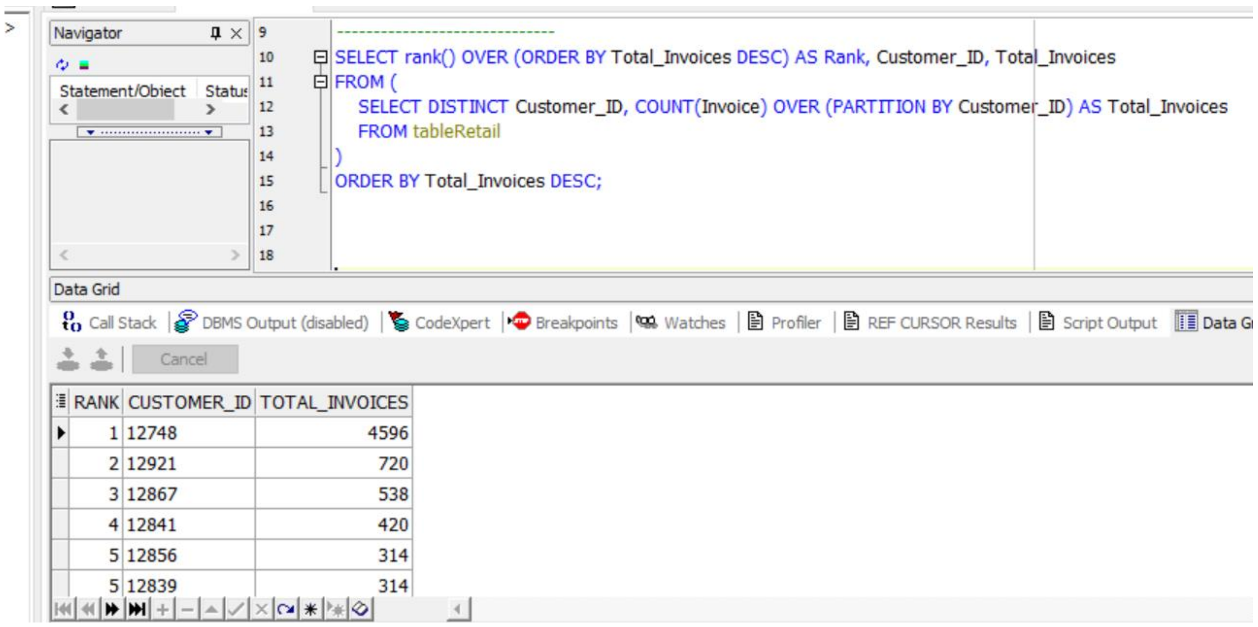
Q1- Using OnlineRetail dataset

These questions were asked to obtain some insights about the data.

1- Who are the top five customers with the highest number of purchases made?

Answering this question can help us:

- identify the best customers based on their purchasing behavior.
- And can inform decision-making around creating targeted offers or promotions to incentivize these customers to make additional purchases.
- SELECT rank() OVER (ORDER BY Total_Invoices DESC) AS Rank, Customer_ID, Total_Invoices
- FROM (
- SELECT DISTINCT Customer_ID, COUNT(Invoice) OVER (PARTITION BY Customer_ID) AS Total_Invoices
- FROM tableRetail
-)
- ORDER BY Total_Invoices DESC;



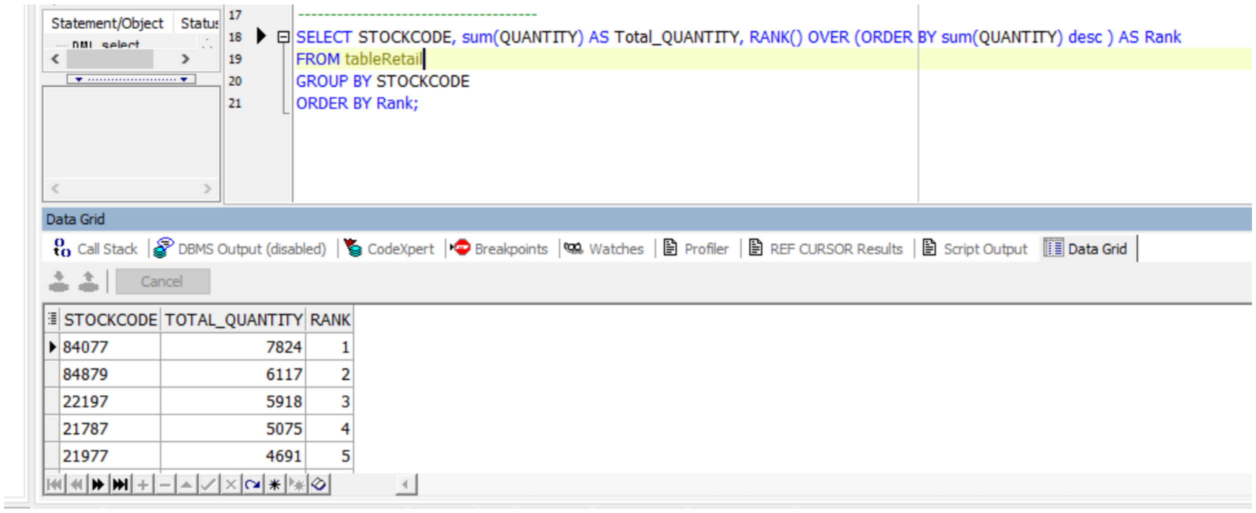
```
-------------------------------
SELECT rank() OVER (ORDER BY Total_Invoices DESC) AS Rank, Customer_ID, Total_Invoices
FROM (
    SELECT DISTINCT Customer_ID, COUNT(Invoice) OVER (PARTITION BY Customer_ID) AS Total_Invoices
    FROM tableRetail
)
ORDER BY Total_Invoices DESC;
```

RANK	CUSTOMER_ID	TOTAL_INVOICES
1	12748	4596
2	12921	720
3	12867	538
4	12841	420
5	12856	314
5	12839	314

2-What are the top five STOCK codes sold in terms of quantity?

By answering this question:

- We can identify the top-selling products, which can help us to increase their visibility in the market .and allocate more space for them in-store.
- We can also use this information to strategically place these products alongside lower-selling products to increase the chances of cross-selling and upselling.
- We can make more informed decisions around inventory management and purchasing to ensure that we always have enough stock of these high-selling products to meet customer demand.
- SELECT STOCKCODE, sum(QUANTITY) AS Total_QUANTITY, RANK() OVER (ORDER BY sum(QUANTITY) desc) AS Rank
- FROM tableRetail
- GROUP BY STOCKCODE
- ORDER BY Rank;



```
------------------------------------
SELECT STOCKCODE, sum(QUANTITY) AS Total_QUANTITY, RANK() OVER (ORDER BY sum(QUANTITY) desc ) AS Rank
FROM tableRetail
GROUP BY STOCKCODE
ORDER BY Rank;
```

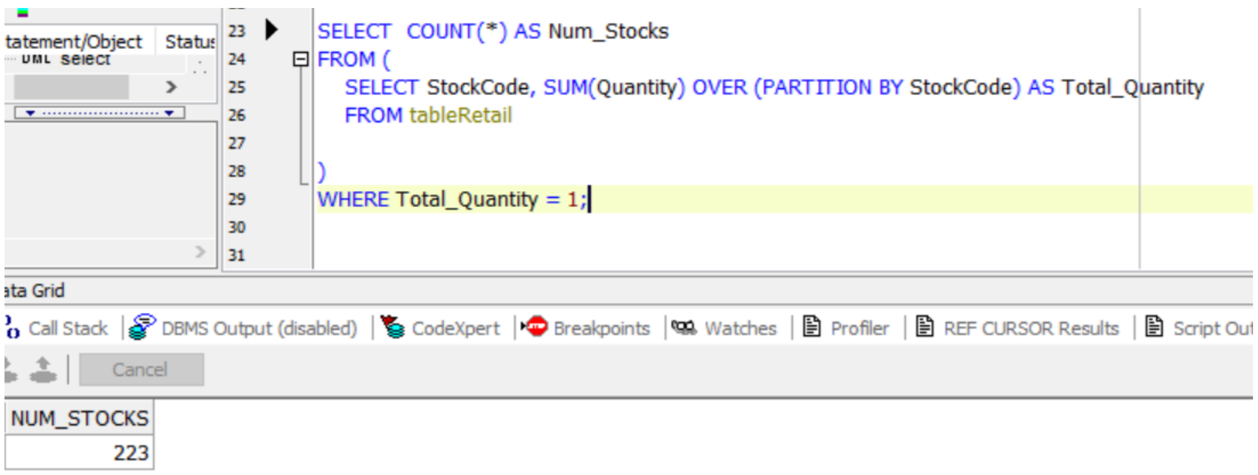
STOCKCODE	TOTAL_QUANTITY	RANK
84077	7824	1
84879	6117	2
22197	5918	3
21787	5075	4
21977	4691	5

3-How many stocks were purchased only once?

By answering this question:

- We can identify the least selling products.
- the business can identify which products need additional support to increase sales
- Improving the overall performance of the products and planning new marketing initiatives to increase demand for these products.

```
SELECT  COUNT(*) AS Num_Stocks
FROM (
    SELECT StockCode, SUM(Quantity) OVER (PARTITION BY StockCode) AS Total_Quantity
    FROM tableRetail

)
WHERE Total_Quantity = 1;
```



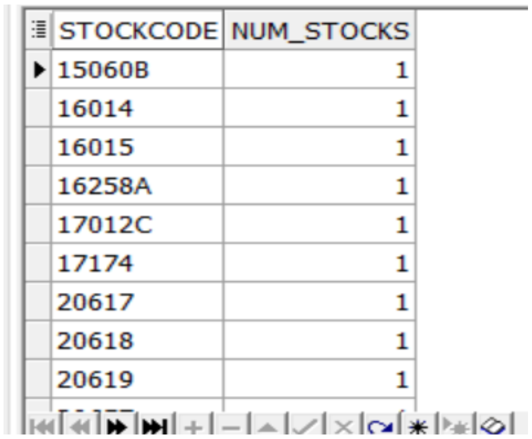
```
SELECT  COUNT(*) AS Num_Stocks
FROM (
    SELECT StockCode, SUM(Quantity) OVER (PARTITION BY StockCode) AS Total_Quantity
    FROM tableRetail

)
WHERE Total_Quantity = 1;
```

```
NUM_STOCKS
       223
```

-

4-Sample of these Stocks

- SELECT StockCode, COUNT(*) AS Num_Stocks
- FROM (
-     SELECT StockCode, SUM(Quantity) OVER (PARTITION BY StockCode) AS Total_Quantity
-     FROM tableRetail
-)
- WHERE Total_Quantity = 1
- GROUP BY StockCode;

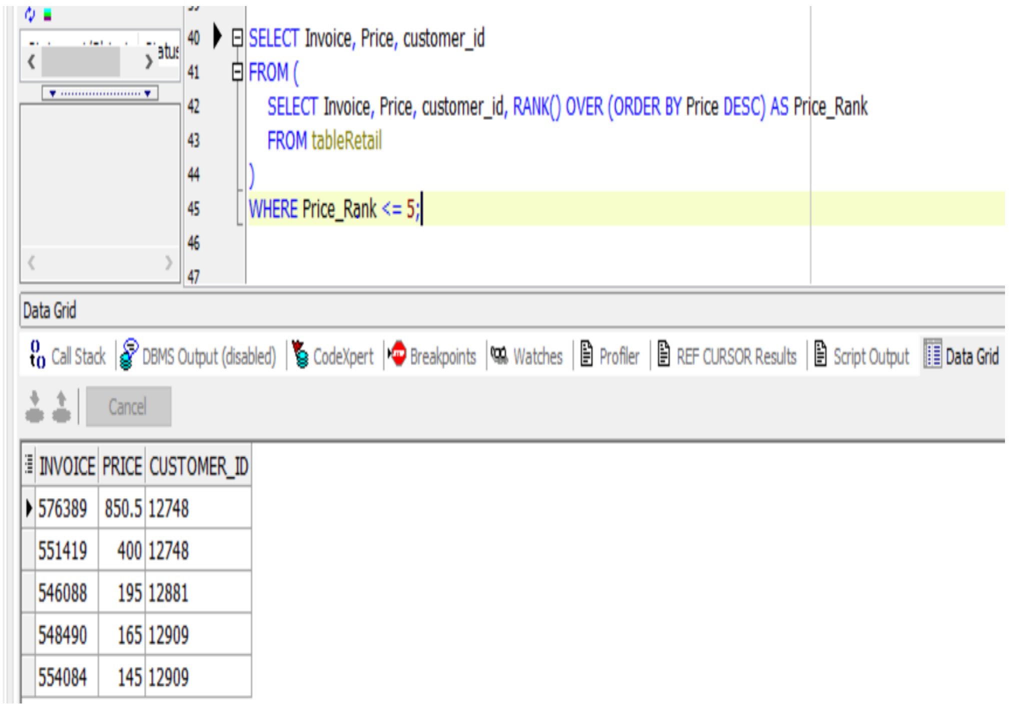


STOCKCODE	NUM_STOCKS
15060B	1
16014	1
16015	1
16258A	1
17012C	1
17174	1
20617	1
20618	1
20619	1

5- What is the top 5 highest-priced invoices in the "tableRetail" table?

This information can then be used to inform pricing strategies, inventory management, and marketing efforts to target high-value customers and increase revenue.

```
 SELECT StockCode, COUNT(*) AS Num_Stocks
FROM (
   SELECT StockCode, SUM(Quantity) OVER (PARTITION BY StockCode) AS Total_Quantity
   FROM tableRetail
)
WHERE Total_Quantity = 1
         GROUP BY StockCode;
```



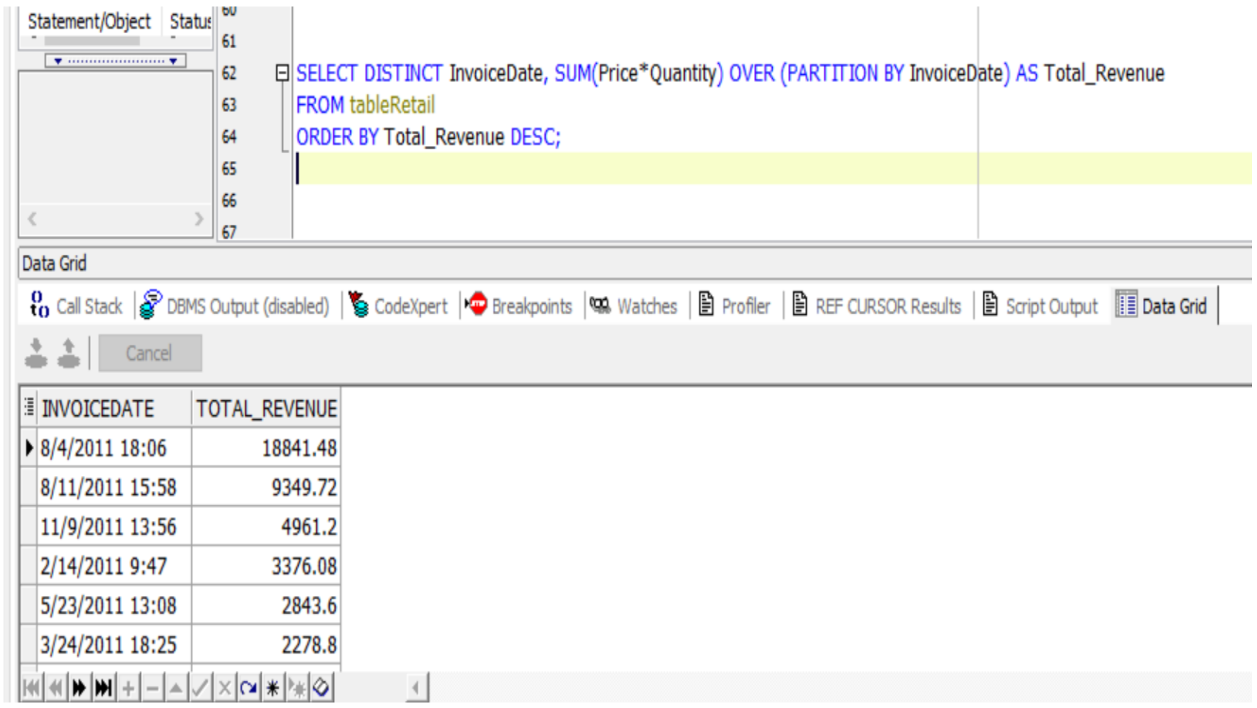
```
SELECT Invoice, Price, customer_id
FROM (
   SELECT Invoice, Price, customer_id, RANK() OVER (ORDER BY Price DESC) AS Price_Rank
   FROM tableRetail
)
WHERE Price_Rank <= 5;
```

INVOICE	PRICE	CUSTOMER_ID
576389	850.5	12748
551419	400	12748
546088	195	12881
548490	165	12909
554084	145	12909

6-Which invoicedates have the highest revenue?

By answering this question :

- We can provide valuable insights into the overall sales performance of the business
- And help identify trends or seasonal fluctuations in revenue.
- This information can then be used to inform business decisions around inventory management, marketing campaigns, and other strategies to drive revenue growth.
- SELECT distinct(INVOICEDATE),sum(Price*QUANTITY) OVER (partition by INVOICEDATE) As Total_Revenue
- FROM tableRetail
- order by Total_Revenue desc;



```
SELECT DISTINCT InvoiceDate, SUM(Price*Quantity) OVER (PARTITION BY InvoiceDate) AS Total_Revenue
FROM tableRetail
ORDER BY Total_Revenue DESC;
```

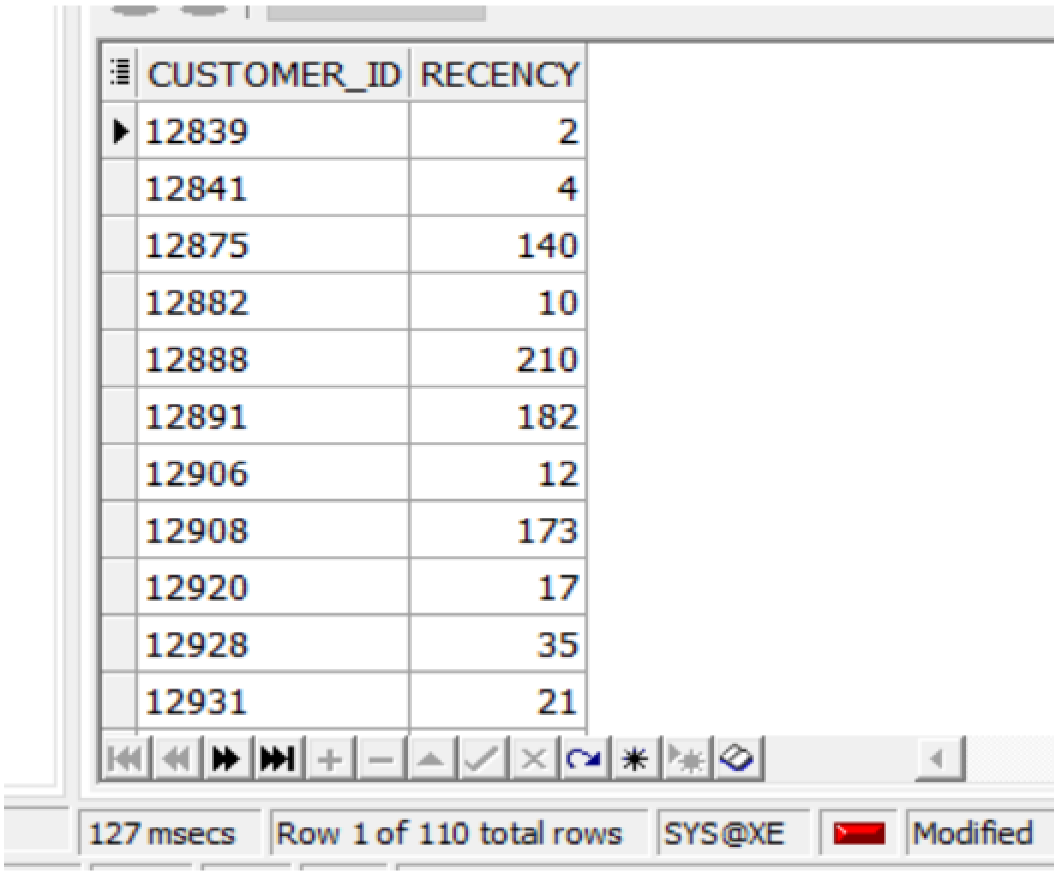
INVOICEDATE	TOTAL_REVENUE
8/4/2011 18:06	18841.48
8/11/2011 15:58	9349.72
11/9/2011 13:56	4961.2
2/14/2011 9:47	3376.08
5/23/2011 13:08	2843.6
3/24/2011 18:25	2278.8

Answering of Q2:

- STEP 1 : Calculate the recency

-- This query calculates the recency of each customer's last transaction in number of days using reference date based on the maximum date in the entire dataset.

```
SELECT distinct(customer_id),round(MONTHS_BETWEEN((SELECT MAX(TO_DATE(InvoiceDate,
'MM/DD/YYYY HH24:MI:SS')) FROM tableRetail),MAX(TO_DATE(InvoiceDate, 'MM/DD/YYYY
HH24:MI:SS'))over(partition by customer_id))*30) AS Recency
        FROM tableRetail;
```



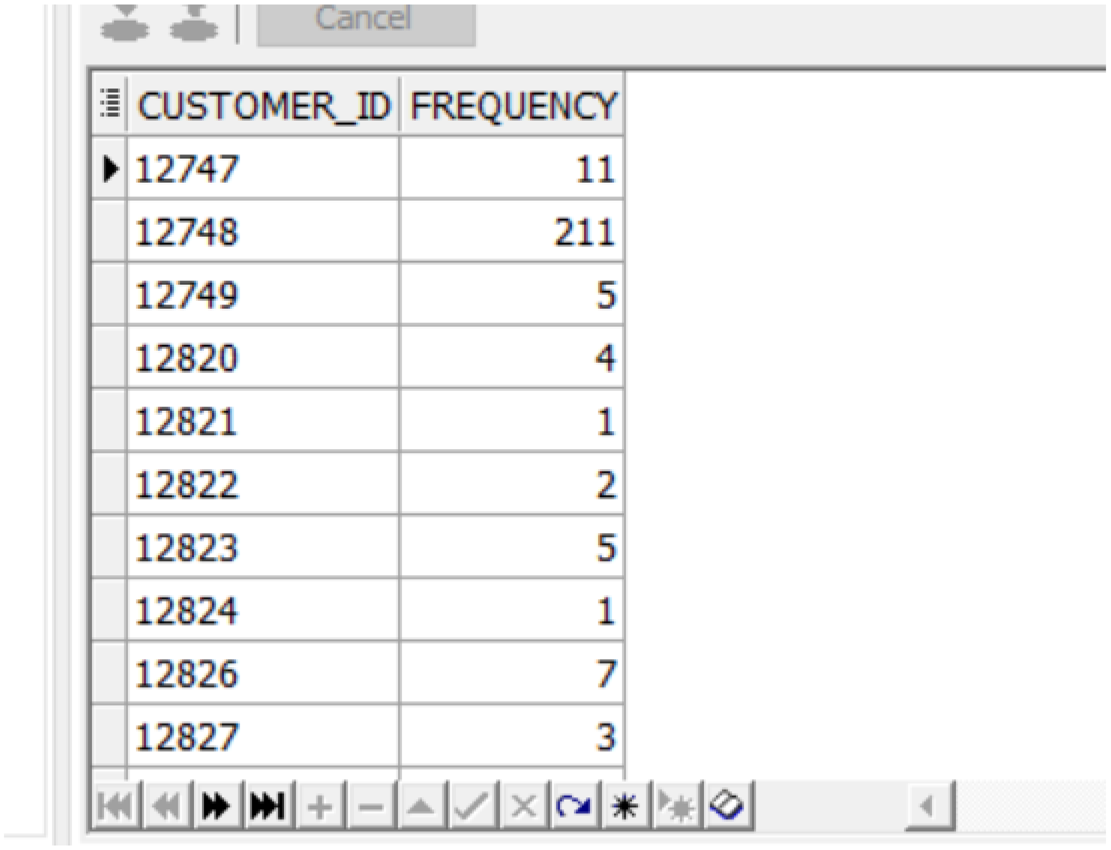
CUSTOMER_ID	RECENCY
12839	2
12841	4
12875	140
12882	10
12888	210
12891	182
12906	12
12908	173
12920	17
12928	35
12931	21

127 msecs | Row 1 of 110 total rows | SYS@XE | Modified

- Step 2: Calculate the Frequency Column.

  The number of times the customer has brought from the store.

- SELECT
-    distinct(customer_id),
-   COUNT(distinct (INVOICEDATE)) over(partition by customer_id) as frequency
- FROM tableRetail
- Order by   customer_id;

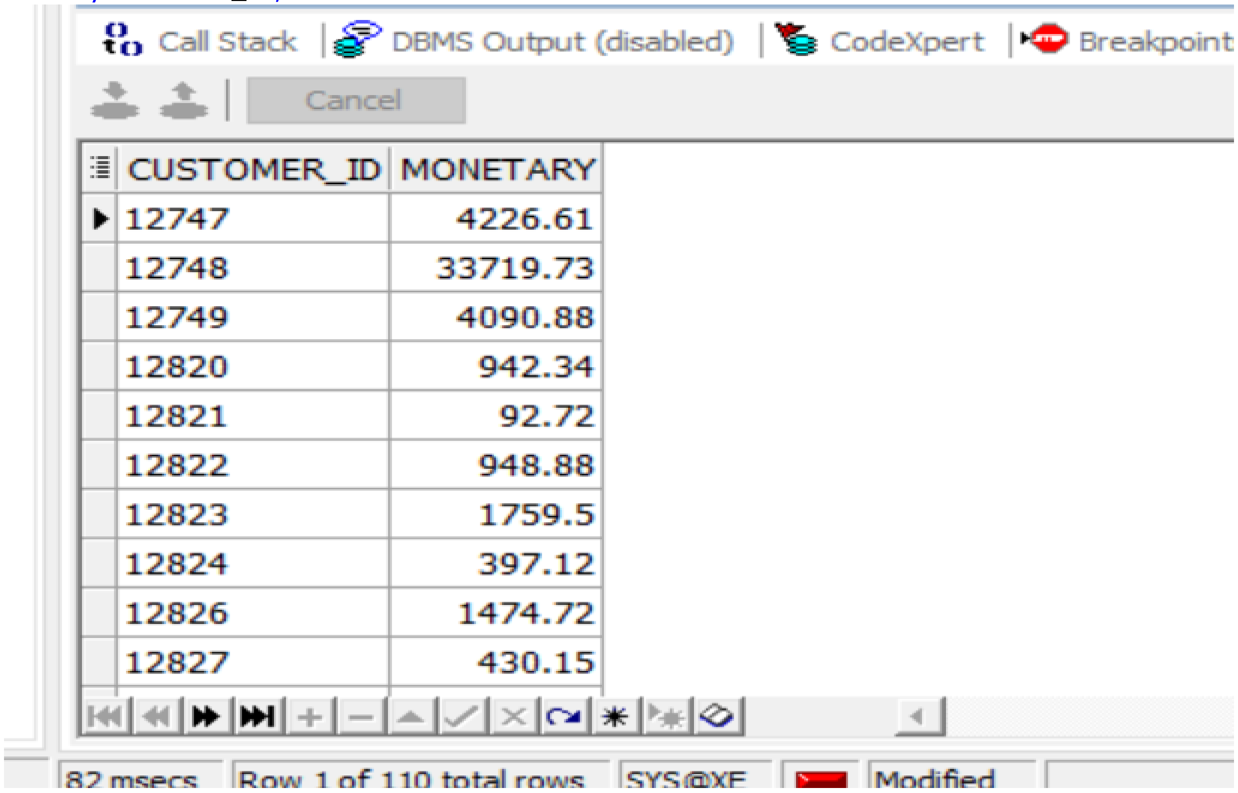


CUSTOMER_ID	FREQUENCY
12747	11
12748	211
12749	5
12820	4
12821	1
12822	2
12823	5
12824	1
12826	7
12827	3

- Step 3: Calculate the Monetary Column.

  how much each customer has paid for our products.

```
--- Monetary Column:
SELECT distinct(Customer_ID), SUM(Price * Quantity)over(partition by customer_id) AS Monetary
FROM tableRetail
    order by  Customer_ID;
```

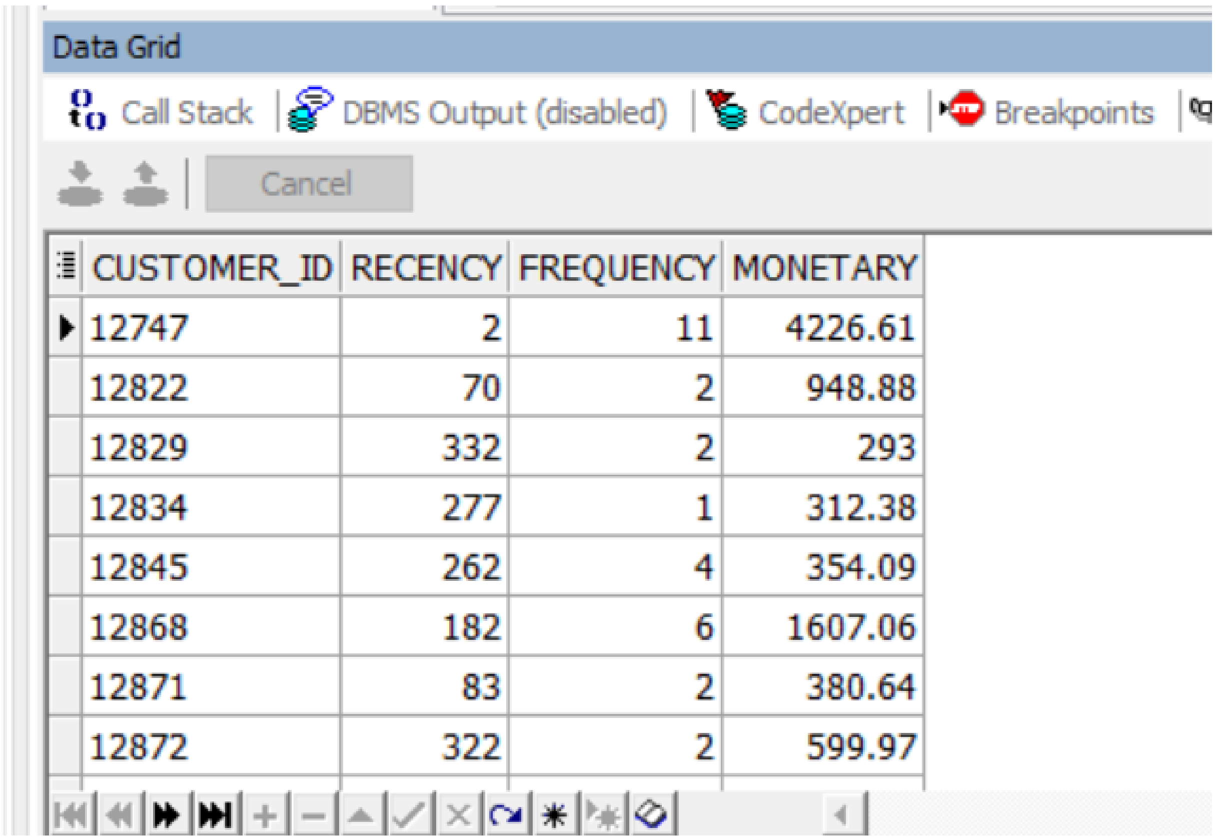


CUSTOMER_ID	MONETARY
12747	4226.61
12748	33719.73
12749	4090.88
12820	942.34
12821	92.72
12822	948.88
12823	1759.5
12824	397.12
12826	1474.72
12827	430.15

```
82 msecs | Row 1 of 110 total rows | SYS@XE | Modified
```

Step 4 : calculate the Recency,frequency,Monetary

```
WITH rfm_customers AS (
SELECT distinct(customer_id),round(MONTHS_BETWEEN((SELECT MAX(TO_DATE(InvoiceDate,
'MM/DD/YYYY HH24:MI:SS')) FROM tableRetail),MAX(TO_DATE(InvoiceDate, 'MM/DD/YYYY
HH24:MI:SS'))over(partition by customer_id))*30) AS Recency,
COUNT(distinct (INVOICEDATE))  over(partition by customer_id) as frequency,
 SUM(Price * Quantity)over(partition by customer_id) AS Monetary
FROM tableRetail

)

    SELECT * FROM rfm_customers
```

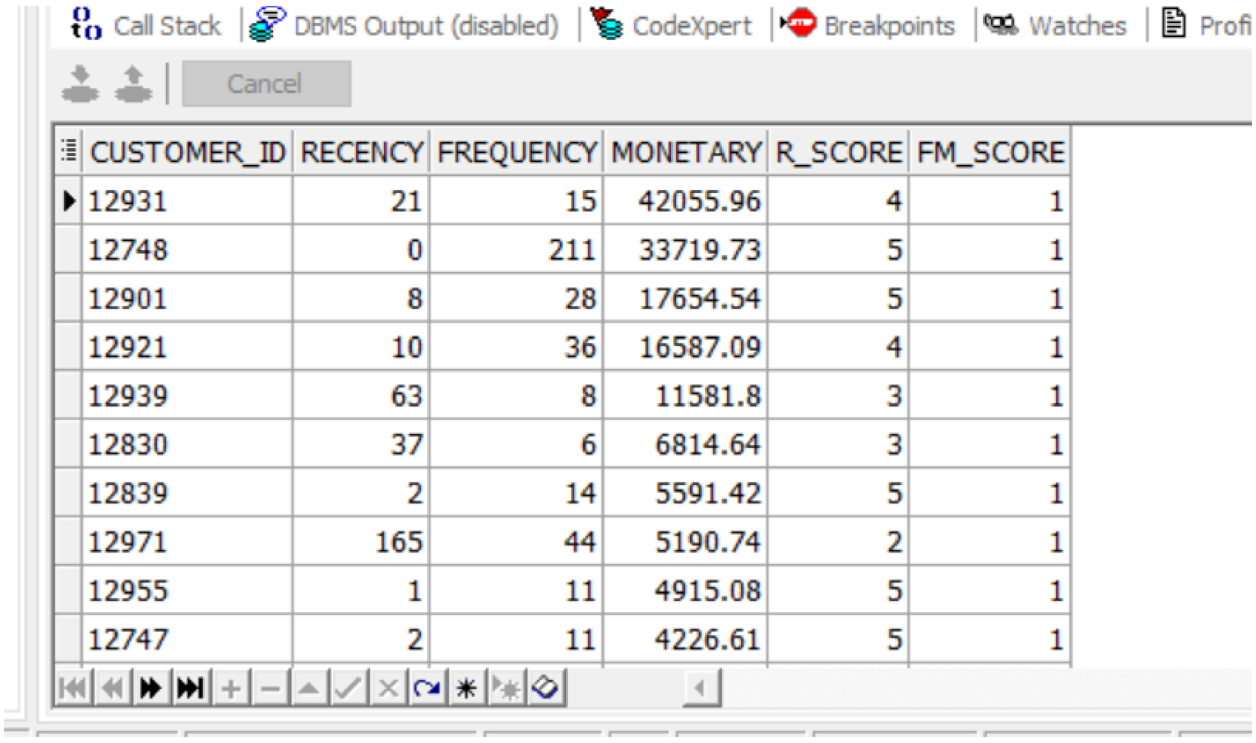


CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY
12747	2	11	4226.61
12822	70	2	948.88
12829	332	2	293
12834	277	1	312.38
12845	262	4	354.09
12868	182	6	1607.06
12871	83	2	380.64
12872	322	2	599.97

Step 5: Calculate the R_Score and fm_score

Categorizing my customers into 5 categories based on their average score of FREQUENCY,MONETARY together (FM_SCORE) and RECENCY score on its own (R_SCORE) USING NTILE FUNCTION .

```
--Calculate the R_Score and fm_score
rfm_scores AS (
  SELECT customer_id, Recency, frequency, Monetary,
        NTILE(5) OVER(ORDER BY Recency DESC) AS R_Score,
        (NTILE(5) OVER(ORDER BY AVG(frequency) DESC) + NTILE(5) OVER(ORDER BY AVG(Monetary)
DESC))/2 AS FM_Score

  FROM rfm_customers
  group by  Recency, frequency, Monetary, customer_id
)
select * from rfm_scores;
```



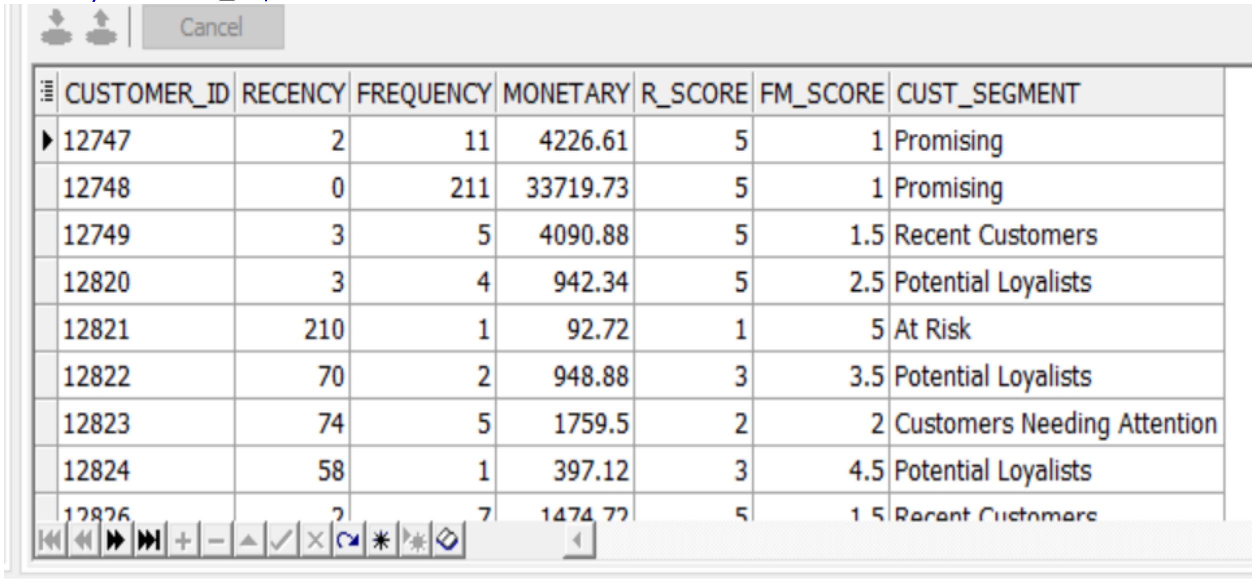
CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE
12931	21	15	42055.96	4	1
12748	0	211	33719.73	5	1
12901	8	28	17654.54	5	1
12921	10	36	16587.09	4	1
12939	63	8	11581.8	3	1
12830	37	6	6814.64	3	1
12839	2	14	5591.42	5	1
12971	165	44	5190.74	2	1
12955	1	11	4915.08	5	1
12747	2	11	4226.61	5	1

- Step 6: Segmenting :

  customers into Champions - Loyal Customers - Potential Loyalists – Recent Customers – Promising -Customers Needing Attention - At Risk - Cant Lose Them – Hibernating – Lost. Using Case-When according to their R_score and fm_score.

```
----Final Step if calculate the Recency,frequency,Monetary
WITH rfm_customers AS (
SELECT distinct(customer_id),round(MONTHS_BETWEEN((SELECT MAX(TO_DATE(InvoiceDate,
'MM/DD/YYYY HH24:MI:SS')) FROM tableRetail),MAX(TO_DATE(InvoiceDate, 'MM/DD/YYYY
HH24:MI:SS'))over(partition by customer_id))*30) AS Recency,
COUNT(distinct (INVOICEDATE))  over(partition by customer_id) as frequency,
 SUM(Price * Quantity)over(partition by customer_id) AS Monetary
FROM tableRetail

),
--Calculate the R_Score and fm_score
rfm_scores AS (
  SELECT customer_id, Recency, frequency, Monetary,
         NTILE(5) OVER(ORDER BY Recency DESC) AS R_Score,
         (NTILE(5) OVER(ORDER BY AVG(frequency) DESC) + NTILE(5) OVER(ORDER BY AVG(Monetary)
DESC))/2 AS FM_Score

  FROM rfm_customers
  group by  Recency, frequency, Monetary, customer_id
)
```

```
select  customer_id, Recency, frequency, Monetary,R_Score,FM_Score,

    CASE
     WHEN R_Score >=4 AND FM_Score >=4 THEN 'Champions'
    WHEN R_Score >=3 AND FM_Score >=2 THEN 'Potential Loyalists'
    WHEN R_Score >=3 AND FM_Score >=3 THEN 'Loyal Customers'
    WHEN R_Score =5 AND FM_Score >1 THEN 'Recent Customers'
    WHEN R_Score >=3 AND FM_Score >=1 THEN 'Promising'
    WHEN R_Score >=2 AND FM_Score >=2 THEN 'Customers Needing Attention'
    WHEN R_Score >=1 AND FM_Score >=3 THEN 'At Risk'
    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'

    END AS Cust_segment

FROM rfm_scores
order by customer_id ;
```



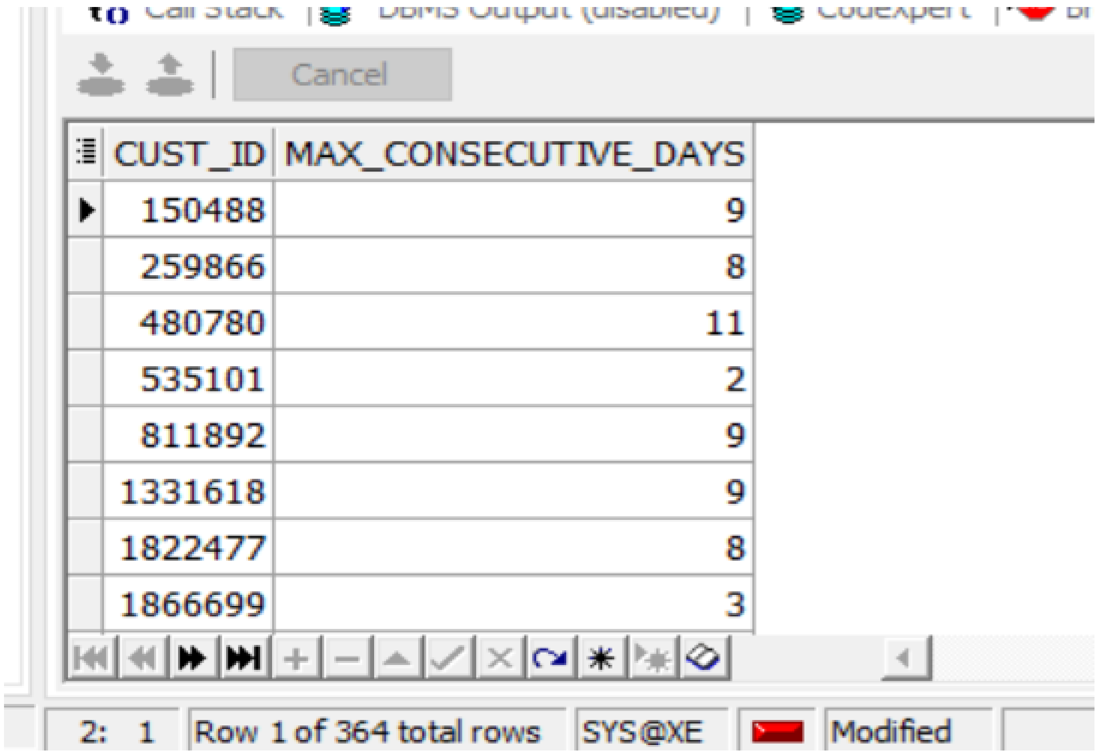
CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	CUST_SEGMENT
12747	2	11	4226.61	5	1	Promising
12748	0	211	33719.73	5	1	Promising
12749	3	5	4090.88	5	1.5	Recent Customers
12820	3	4	942.34	5	2.5	Potential Loyalists
12821	210	1	92.72	1	5	At Risk
12822	70	2	948.88	3	3.5	Potential Loyalists
12823	74	5	1759.5	2	2	Customers Needing Attention
12824	58	1	397.12	3	4.5	Potential Loyalists
12826	2	7	1474.72	5	1.5	Recent Customers

Answering of Q2:

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

This SQL query calculates the maximum number of consecutive days on which each customer made a purchase.

```
--selects the customer ID and the maximum number of consecutive days on which each customer made a
purchase.
SELECT cust_id, MAX(consecutive_days) AS max_consecutive_days
--selects the customer ID and a count of the number of consecutive days on which the customer made a
purchase.
FROM (
  SELECT cust_id, COUNT(*) AS consecutive_days
  --calculate a difference between the purchase date for each transaction and a running count of the
transactions
  FROM (
    SELECT cust_id, p_date, p_date - ROW_NUMBER() OVER (PARTITION BY cust_id ORDER BY p_date)
AS date_diff
    FROM daily_purchasing
  )
  GROUP BY cust_id, date_diff
)
GROUP BY cust_id;
```



Call Stack | DBMS Output (disabled) | CodeXpert | Bl

Cancel

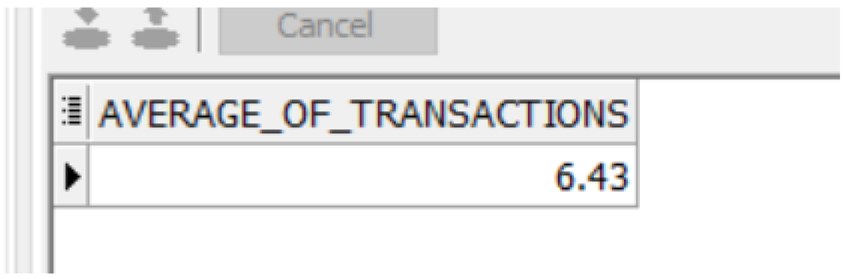
CUST_ID	MAX_CONSECUTIVE_DAYS
150488	9
259866	8
480780	11
535101	2
811892	9
1331618	9
1822477	8
1866699	3

```
2:   1   Row 1 of 364 total rows   SYS@XE   Modified
```

B) On average, How many days/transactions does it take a customer to reach a spent

threshold of 250 L.E?

```
--  select the average number of transactions it takes for a customer to reach a spent threshold of 250
L.E and rounds the result to 2 decimal places.
Select round(avg(days_of_250),2) average_of_transactions
from
----calculate the min number of transactions it takes for each customer to reach a spent threshold of 250
L.E.
( select cust_id,min(count_of_trans) days_of_250
from(
----calculates the running total of the amount values for each customer
--- and use the dense rank() to rank the  sum of the total spend
select table1.*, dense_rank() over(partition by cust_id order by sum_total) as count_of_trans
from(
select  daily_purchasing.*,sum(AMOUNT) over(partition by cust_id order by p_date rows between
unbounded preceding and current row) sum_total
from daily_purchasing
)table1
) table2
where sum_total>=250
group by cust_id
)table3;
-----------
```



AVERAGE_OF_TRANSACTIONS
6.43