ANALYTICAL SQL CASE STUDY

Prepared by:

Ahmed Hatem

Information
Technology
Institute

Analytical SQL Case Study

First Question:

Objective:

Customers have purchasing transactions 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.

Understanding our data:

Our data is a retail transaction from our customers, and it contains:

- **Invoice Number:** Unique identifier for each transaction.
- **Stock code:** Identifier for products in inventory.
- **Quantity:** Number of items bought or sold.
- **Invoice Date:** Date when invoice was issued.
- **Price:** Monetary value of a product.
- **Customer ID:** Unique identifier for customers.
- **Country:** Geographic location of customer or transaction.

Our data has **12858 rows**, and here is a small sample of the data.

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

I am asked to:

1. Apply at least 5 analytical SQL queries that tell a story about the data.
2. Write a small **description** about the business meaning behind each query.

First Query:

What are the most selling products?

Description:

This SQL query helps us understand product popularity by calculating total sales quantities for each stock code. The use of dense_rank() helps identify best-selling products. This data-driven insight can guide marketing strategies and inventory management, improving sales and customer satisfaction.

```
with sum_quantity as(
select stockcode, sum(quantity) as total_quantity
from tableretail
group by stockcode
order by total_quantity desc
)
select stockcode, total_quantity,
dense_rank() over(order by total_quantity desc) as most_selling
from sum_quantity;
```

STOCKCODE	TOTAL_QUANTITY	MOST_SELLING
84077	7824	1
84879	6117	2
22197	5918	3
21787	5075	4
21977	4691	5
21703	2996	6
17096	2019	7
15036	1920	8
23203	1803	9
21790	1579	10

Second Query:

What are the monthly sales compared to the average sales?

Description:

This SQL query helps businesses to compare their monthly sales data with the average, identifying which months are performing above or below the average. This analysis is crucial for strategic planning, enabling businesses to pinpoint trends, seasonal fluctuations, and areas needing attention or optimization. The query empowers businesses to make data-driven decisions, boosting sales performance and overall profitability.

```
WITH monthly_sales AS (
    SELECT
        EXTRACT(MONTH FROM TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS Month,
        EXTRACT(YEAR FROM TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS Year,
        SUM(price * quantity) AS "Sales/Month",
        AVG(SUM(price * quantity)) OVER() AS Avg_Sales
    FROM tableretail
    GROUP BY EXTRACT(MONTH FROM TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')), EXTRACT(YEAR
FROM TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI'))
    ORDER BY Year, Month
)

SELECT
    Month,
    Year,
    "Sales/Month",
    ROUND(Avg_Sales, 2) AS Avg_Sales,
    ROUND("Sales/Month" - Avg_Sales, 2) AS Difference,
    CASE
        WHEN "Sales/Month" < Avg_Sales THEN 'Below Average'
        ELSE 'Above Average'
    END AS Status
FROM monthly_sales;
```

MONTH	YEAR	Sales/Month	AVG_SALES	DIFFERENCE	STATUS
12	2010	13422.96	19670.64	-6247.68	Below Average
1	2011	9541.29	19670.64	-10129.35	Below Average
2	2011	13336.84	19670.64	-6333.8	Below Average
3	2011	17038.01	19670.64	-2632.63	Below Average
4	2011	10980.51	19670.64	-8690.13	Below Average
5	2011	19496.18	19670.64	-174.46	Below Average
6	2011	13517.01	19670.64	-6153.63	Below Average
7	2011	15664.54	19670.64	-4006.1	Below Average
8	2011	38374.64	19670.64	18704	Above Average
9	2011	27853.82	19670.64	8183.18	Above Average
10	2011	19735.07	19670.64	64.43	Above Average
11	2011	45633.38	19670.64	25962.74	Above Average
12	2011	11124.13	19670.64	-8546.51	Below Average

Third Query

What are the top customers who paid more than 1000?

Description:

This SQL query addresses this need by analyzing customer spending patterns. It calculates the total amount paid by each customer, ranks them based on this total spend, and identifies those who have spent over $1000. By focusing on these high-value customers, businesses can tailor marketing strategies, offer targeted promotions, and build customer loyalty, ultimately maximizing revenue and profitability.

```
WITH top_customers AS (
SELECT customer_id, SUM(price * quantity) AS "Amount Paid"
FROM tableretail
GROUP BY customer_id
)
SELECT customer_id, "Amount Paid", DENSE_RANK() OVER(ORDER BY "Amount Paid" DESC) AS "Top Customers"
FROM top_customers
where "Amount Paid" >= 1000;
```

CUSTOMER_ID	Amount Paid	Top Customers
12931	42055.96	1
12748	33719.73	2
12901	17654.54	3
12921	16587.09	4
12939	11581.8	5
12830	6814.64	6
12839	5591.42	7
12971	5190.74	8
12955	4757.16	9
12747	4196.01	10
12949	4167.22	11
12749	4090.88	12
12867	4036.82	13

Fourth Query

What are the products that make the most revenue?

Description:

The business needs this query to gain a clear understanding of their top-selling products. This knowledge can help optimize inventory management, adjust pricing strategies, target marketing efforts, optimize supply chains, guide product development, and make more accurate revenue forecasts. Ultimately, this information is crucial for increasing sales and profitability.

```
WITH top_products AS (
SELECT stockcode, SUM(price * quantity) AS "Revenue Made"
FROM tableretail
GROUP BY stockcode
)
SELECT stockcode, "Revenue Made", DENSE_RANK() OVER(ORDER BY "Revenue Made" DESC) AS "Top
Products"
FROM top_products;
```

STOCKCODE	Revenue Made	Top Products
84879	9114.69	1
22197	4323.1	2
21787	4059.35	3
22191	3461.2	4
23203	3357.44	5
21479	2736.01	6
23215	2697.36	7
22970	2493.6	8
22570	2458.08	9
22992	2308.05	10

Fifth Query:

What is the average time between purchases for each customer?

Description:

The query provides a means of gaining deep insights into customer behavior, which is vital for crafting effective business strategies. By analyzing the frequency of customer visits and the intervals between them, businesses can identify their most loyal customers, providing an opportunity to reward and retain them. Furthermore, this data offers a window into customer engagement levels, enabling businesses to fine-tune their marketing strategies for maximum effectiveness. The query's ability to detect patterns in visit intervals also aids in identifying customers at risk of churn, allowing for proactive retention strategies. Additionally, it sheds light on peak visit times, which can inform staffing and inventory management decisions, leading to increased operational efficiency.

```
WITH cst_visits AS (
  SELECT customer_id,
    TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI') AS "Invoice Date",
    LAG(TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) OVER(PARTITION BY customer_id ORDER BY
TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS "Previous Visit",
    TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')
      - LAG(TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) OVER(PARTITION BY customer_id ORDER BY
TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS "Difference between visits"
  FROM tableretail
  GROUP BY customer_id,  TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')
  ORDER BY customer_id, TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')
)
SELECT customer_id, "Invoice Date", "Previous Visit",
  CASE
    WHEN "Difference between visits" IS NOT NULL
    THEN ROUND("Difference between visits", 1) || ' Days'
    ELSE NULL
  END AS "Difference between visits",
  ROUND(AVG("Difference between visits")  OVER(PARTITION BY customer_id), 1) || ' Days' AS "Average
Difference"
FROM cst_visits;
```

CUSTOMER_ID	Invoice Date	Previous Visit	Difference between visits	Average Difference
12826	12/9/2010 3:21:00 PM			60.5 Days
12826	1/19/2011 12:52:00 PM	12/9/2010 3:21:00 PM	40.9 Days	60.5 Days
12826	1/27/2011 12:06:00 PM	1/19/2011 12:52:00 PM	8 Days	60.5 Days
12826	6/14/2011 9:41:00 AM	1/27/2011 12:06:00 PM	137.9 Days	60.5 Days
12826	9/29/2011 10:55:00 AM	6/14/2011 9:41:00 AM	107.1 Days	60.5 Days
12826	11/10/2011 3:37:00 PM	9/29/2011 10:55:00 AM	42.2 Days	60.5 Days
12826	12/7/2011 10:25:00 AM	11/10/2011 3:37:00 PM	26.8 Days	60.5 Days
12827	10/26/2011 3:44:00 PM			19.4 Days
12827	11/7/2011 11:29:00 AM	10/26/2011 3:44:00 PM	11.8 Days	19.4 Days
12827	12/4/2011 12:17:00 PM	11/7/2011 11:29:00 AM	27 Days	19.4 Days
12828	8/1/2011 4:16:00 PM			25.5 Days
12828	8/19/2011 9:23:00 AM	8/1/2011 4:16:00 PM	17.7 Days	25.5 Days
12828	9/1/2011 5:14:00 PM	8/19/2011 9:23:00 AM	13.3 Days	25.5 Days
12828	10/5/2011 10:53:00 AM	9/1/2011 5:14:00 PM	33.7 Days	25.5 Days
12828	12/2/2011 11:12:00 AM	10/5/2011 10:53:00 AM	58 Days	25.5 Days
12828	12/7/2011 8:45:00 AM	12/2/2011 11:12:00 AM	4.9 Days	25.5 Days
12829	12/14/2010 2:54:00 PM			23.8 Days
12829	1/7/2011 11:13:00 AM	12/14/2010 2:54:00 PM	23.8 Days	23.8 Days
12830	6/21/2011 10:53:00 AM			26.8 Days
12830	7/6/2011 10:52:00 AM	6/21/2011 10:53:00 AM	15 Days	26.8 Days
12830	7/28/2011 5:17:00 PM	7/6/2011 10:52:00 AM	22.3 Days	26.8 Days
12830	7/28/2011 5:18:00 PM	7/28/2011 5:17:00 PM	0 Days	26.8 Days
12830	9/9/2011 3:02:00 PM	7/28/2011 5:18:00 PM	42.9 Days	26.8 Days

Sixth Query:

What is the distribution of the total amount spent per customer?

Description:

The query aims to identify top customers based on their spending habits and group them into quartiles for further analysis. This segmentation is crucial for tailoring marketing strategies, offering promotions, managing inventory, and providing a personalized level of service. The data provided by quartiles enables businesses to understand customer spending patterns, forecast future growth, and make informed decisions to drive profitability and enhance customer satisfaction. Ultimately, the quartile analysis serves as a valuable tool for businesses seeking to optimize their operations and increase revenue by targeting high-value customers effectively.

```
WITH top_customers AS (
SELECT customer_id, SUM(price * quantity) AS "Amount Paid"
FROM tableretail
GROUP BY customer_id
)
SELECT customer_id, "Amount Paid",
NTILE(4) OVER(ORDER BY "Amount Paid" DESC) AS Quartile
FROM top_customers;
```

CUSTOMER_ID	Amount Paid	QUARTILE
12840	2726.77	1
12836	2612.86	1
12913	2483.63	1
12856	2179.93	1
12909	2178.67	1
12935	2160.7	1
12948	2064.95	1
12928	2062.7	1
12853	1957.1	1
12963	1856.63	1
12950	1843	2
12823	1759.5	2
12843	1702.26	2
12912	1662.3	2
12967	1660.9	2

Information
Technology
Institute

Seventh Query:

How do customers' purchases compare to the average?

Description:

The SQL query helps businesses analyze the spending patterns of top customers, comparing their total spend to the average. This analysis aids in targeted marketing, customer retention, and revenue growth. The classification of customers as 'Above Average' or 'Below Average' spenders enables resource optimization and customer segmentation, leading to tailored marketing strategies. Overall, the query is instrumental in enhancing sales, customer satisfaction, and revenue.

```
WITH top_customers AS (
    SELECT customer_id,
        SUM(price * quantity) AS "Amount Paid"
    FROM tableretail
    GROUP BY customer_id
)

SELECT customer_id,
    "Amount Paid",
    ROUND(AVG("Amount Paid") OVER(), 2) AS "Average Purchase",
    "Amount Paid" - ROUND(AVG("Amount Paid") OVER(), 2) AS "Difference",
    CASE
        WHEN "Amount Paid" > ROUND(AVG("Amount Paid") OVER(), 2) THEN 'Above Average'
        ELSE 'Below Average'
    END AS Status
FROM top_customers;
```

CUSTOMER_ID	Amount Paid	Average Purchase	Difference	STATUS
12828	1018.71	2324.71	-1306	Below Average
12829	293	2324.71	-2031.71	Below Average
12833	417.38	2324.71	-1907.33	Below Average
12841	4022.35	2324.71	1697.64	Above Average
12844	325.96	2324.71	-1998.75	Below Average
12856	2179.93	2324.71	-144.78	Below Average
12883	703.47	2324.71	-1621.24	Below Average
12884	309.05	2324.71	-2015.66	Below Average
12916	3006.15	2324.71	681.44	Above Average
12921	16587.09	2324.71	14262.38	Above Average
12933	607.53	2324.71	-1717.18	Below Average
12826	1474.72	2324.71	-849.99	Below Average
12830	6814.64	2324.71	4489.93	Above Average
12840	2726.77	2324.71	402.06	Above Average
12849	1050.89	2324.71	-1273.82	Below Average
12864	147.12	2324.71	-2177.59	Below Average
12871	380.64	2324.71	-1944.07	Below Average

Second Question:

We are using the same dataset from the last question which is the retail transaction.

Objective:

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)

Label each customer based on the below values:

Group name	Recency score	AVG(Frequency & Monetary) score
Champions	5	5
	5	4
	4	5
Potential Loyalists	5	2
	4	2
	3	3
	4	3
Loyal Customers	5	3
	4	4
	3	5
	3	4
Recent Customers	5	1
Promising	4	1
	3	1
Customers Needing Attention	3	2
	2	3
	2	2
At Risk	2	5
	2	4
	1	3
Cant Lose Them	1	5
	1	4
Hibernating	1	2
Lost	1	1

Description:

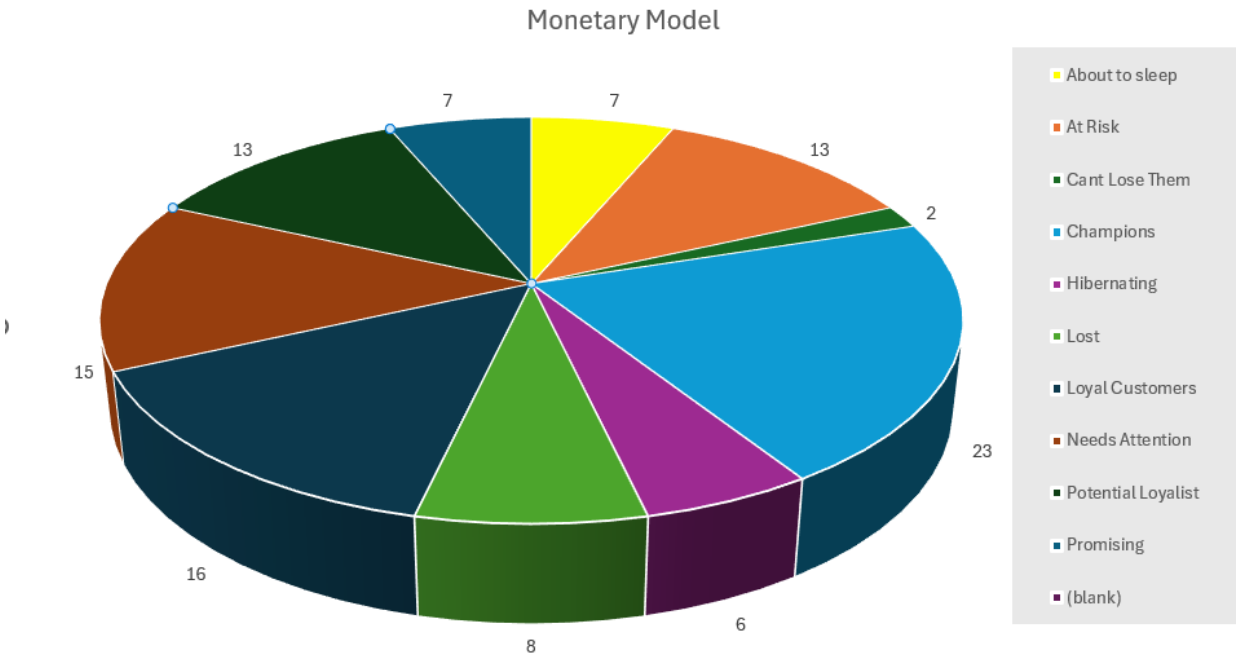
The business need for the query is to implement a Monetary model for customer behavior in product purchasing and segment each customer into predefined groups based on their recency, frequency, and monetary values. The goal is to identify customer segments that can guide marketing strategies, retention efforts, and personalized customer experiences. The segments are as follows:

1. **Champions**: Customers who are both frequent and high spenders. They are the most valuable customers and should be treated with special care.
2. **Loyal Customers**: Customers who are not as frequent as Champions but spend a significant amount of money. They are also valuable and should be encouraged to continue purchasing.
3. **Potential Loyalists**: Customers who are frequent but not as high spenders. They have the potential to become Loyal Customers and should be nurtured.
4. **Recent Customers**: Customers who have made recent purchases. They may need some attention to keep them engaged.
5. **Promising**: Customers who are either frequent or spend a significant amount but not both. They have potential but may need different strategies to convert them into higher-value customers.
6. **Customers Needing Attention**: Customers who are neither frequent nor high spenders. They may need special offers or attention to keep them engaged.
7. **At Risk**: Customers who were once Champions or Loyal Customers but have decreased their frequency or monetary value. They need special attention to prevent them from leaving.
8. **Can't Lose Them**: Customers who are either infrequent or low spenders but have been with the business for a long time. They may be loyal but need attention to increase their value.
9. **Hibernating**: Customers who were once Champions or Loyal Customers but haven't made purchases recently. They need to be re-engaged to return to their previous status.
10. **Lost**: Customers who were once Champions or Loyal Customers but haven't made purchases for a long time. They may be lost forever, but efforts can still be made to re-engage them.

This query calculates the recency, frequency, and monetary values for each customer, segments them into the predefined groups, and provides insights for targeted marketing and retention efforts. By understanding the behavior of different customer segments, businesses can optimize their strategies and maximize customer lifetime value.

```
WITH retail_transactions AS (
    SELECT customer_id, Quantity, Price, Invoice,
        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
    FROM tableRetail
),
rfm AS (
    SELECT customer_id, Recency,
        COUNT(DISTINCT invoice) AS Frequency,
        SUM(price * quantity) AS Monetary
    FROM retail_transactions
    GROUP BY customer_id, recency
),
f_score AS (
    SELECT customer_id, recency, frequency,
        NTILE(5) OVER (ORDER BY frequency ) AS F_score,
        monetary,
        ROUND(PERCENT_RANK() OVER (ORDER BY monetary ), 2) AS monetary_percent_rank
    FROM rfm
),
fm_score AS (
    SELECT customer_id, recency, frequency, monetary, monetary_percent_rank,
        NTILE(5) OVER (ORDER BY recency DESC) AS R_score,
        NTILE(5) OVER (ORDER BY (F_score + monetary_percent_rank) / 2) AS FM_score
    FROM f_score
)
SELECT customer_id, recency, frequency, monetary, R_score, monetary_percent_rank FM_score,
    ROUND(monetary / SUM(monetary) OVER (), 2) AS monetary_percentage,
    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 Loyalist'
        WHEN R_score = 4 AND FM_score IN (2 , 3) THEN 'Potential Loyalist'
        WHEN R_score = 3 AND FM_score = 3 THEN 'Potential Loyalist'
        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 (5, 4) THEN 'Loyal Customers'
        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 = 2 AND FM_score IN (3, 2) THEN 'Needs Attention'
        WHEN R_score = 3 AND FM_score = 2 THEN 'Needs Attention'
        WHEN R_score = 2 AND FM_score IN (5, 4) THEN 'At Risk'
        WHEN R_score = 1 AND FM_score = 3 THEN 'At Risk'
        WHEN R_score = 1 AND FM_score IN (5, 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 'About to sleep '
    END AS Customer_segmentation
FROM fm_score
ORDER BY customer_id DESC;
```

CUSTOM...	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	MONETARY_PERCENTAGE	CUSTOMER_SEGMENTATION
12971	168	45	5190.74	2	0.94	0.02	At Risk
12970	7	4	452.24	5	0.36	0	Loyal Customers
12968	112	1	135.95	2	0.06	0	About to sleep
12967	358	2	1660.9	1	0.71	0.01	At Risk
12966	9	1	160.18	4	0.09	0	Potential Loyalist
12965	89	1	771.91	2	0.49	0	Needs Attention
12963	8	8	1856.63	5	0.75	0.01	Champions
12962	7	2	266.39	5	0.15	0	Potential Loyalist
12957	9	8	4017.54	4	0.87	0.02	Champions
12956	306	1	108.07	1	0.02	0	Hibernating
12955	1	11	4757.16	5	0.93	0.02	Champions
12953	9	1	329.85	4	0.21	0	Promising
12952	5	4	1387.79	5	0.64	0.01	Loyal Customers
12951	8	6	1064.07	4	0.58	0	Loyal Customers
12950	2	3	1843	5	0.74	0.01	Loyal Customers
12949	30	8	4167.22	3	0.91	0.02	Loyal Customers
12948	16	7	2064.95	4	0.78	0.01	Champions
12947	143	6	1603.99	2	0.69	0.01	At Risk
12945	288	1	462.95	1	0.37	0	Lost
12944	35	2	604.51	3	0.42	0	Needs Attention
12942	131	3	683.9	2	0.45	0	Needs Attention
12940	54	2	913.54	3	0.51	0	Needs Attention



Third Question:

Understanding our data:

Our data is daily purchasing transactions for customers.

Cust_Id	Date	Amount
145272	11/5/2019	1.59
145272	11/6/2019	2.98
145272	11/7/2019	2.19
145272	11/8/2019	8.74
1026223	11/3/2019	2
1026223	11/7/2019	33
1026223	11/8/2019	25.5
1767267	11/1/2019	132.69
1767267	11/2/2019	18.64
1767267	11/3/2019	0.4
1767267	11/4/2019	126.33
1767267	11/6/2019	1.92
1767267	11/7/2019	10.07

Our dataset contains **574396 rows.**

Objective:

You are required to answer two questions:

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

Description:

The analysis of the maximum number of consecutive days a customer makes purchases is invaluable for businesses seeking to optimize their operations and enhance customer satisfaction. This query enables businesses to identify loyal customers, understand their purchasing patterns, and tailor marketing strategies to suit their needs. By identifying peak purchase periods and customer engagement levels, businesses can optimize inventory management, offer targeted promotions, and identify opportunities for cross-selling and upselling. This information can ultimately lead to improved customer satisfaction, increased revenue, and long-term customer loyalty.

```
WITH difference AS (
   SELECT Cust_Id, calendar_dt,
     calendar_dt - ROW_NUMBER() OVER (PARTITION BY Cust_Id ORDER BY calendar_dt) AS "Group"
   FROM transactions
), count_days as(
SELECT cust_id,
   COUNT("Group") OVER (PARTITION BY "Group", cust_id) AS count_group
FROM difference
   )
SELECT  cust_id, MAX (count_group) as "Max Consecutive Day"
FROM count_days
GROUP BY cust_id
ORDER BY cust_id;
```

CUST_ID	Max Consecutive Day
26592	35
45234	9
54815	3
60045	15
66688	5
113502	6
145392	6
150488	9
151293	3
175749	2
196249	3

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

Description:

The business need for the query that analyzes the average number of days/transactions it takes a customer to reach a spending threshold of 250 L.E is to understand the purchasing behavior of customers and optimize marketing strategies and inventory management. By identifying the average number of days/transactions required for customers to reach the spending threshold, businesses can adjust their promotions, loyalty programs, and pricing strategies to encourage customers to reach the threshold more quickly. This information can also help businesses anticipate demand and manage inventory more effectively, ensuring that they have the right products in stock to meet customer needs. Ultimately, understanding the average number of days/transactions it takes for customers to reach the spending threshold can help businesses improve customer satisfaction and increase revenue.

```
WITH customers_sales AS (
   SELECT cust_id , calendar_dt ,
   COUNT(calendar_dt) OVER (PARTITION BY cust_id ORDER BY TO_DATE(calendar_dt, 'YYYY-MM-DD')) AS count_days
, Amt_LE ,
   SUM(Amt_LE) OVER (PARTITION BY cust_id ORDER BY TO_DATE(calendar_dt, 'YYYY-MM-DD')) AS total_amount
   FROM transactions
   ) ,

high_amounts AS (
   SELECT cust_id , count_days ,total_amount
   FROM customers_sales
   WHERE total_amount >=250
   )

SELECT ROUND(AVG (count_days), 2) AS avg_days
FROM high_amounts
WHERE (cust_id, total_amount) IN (SELECT cust_id , MIN(total_amount)
                                  FROM high_amounts
                                    GROUP BY cust_id
                                    ) ;
```

AVG_DAYS
6.31