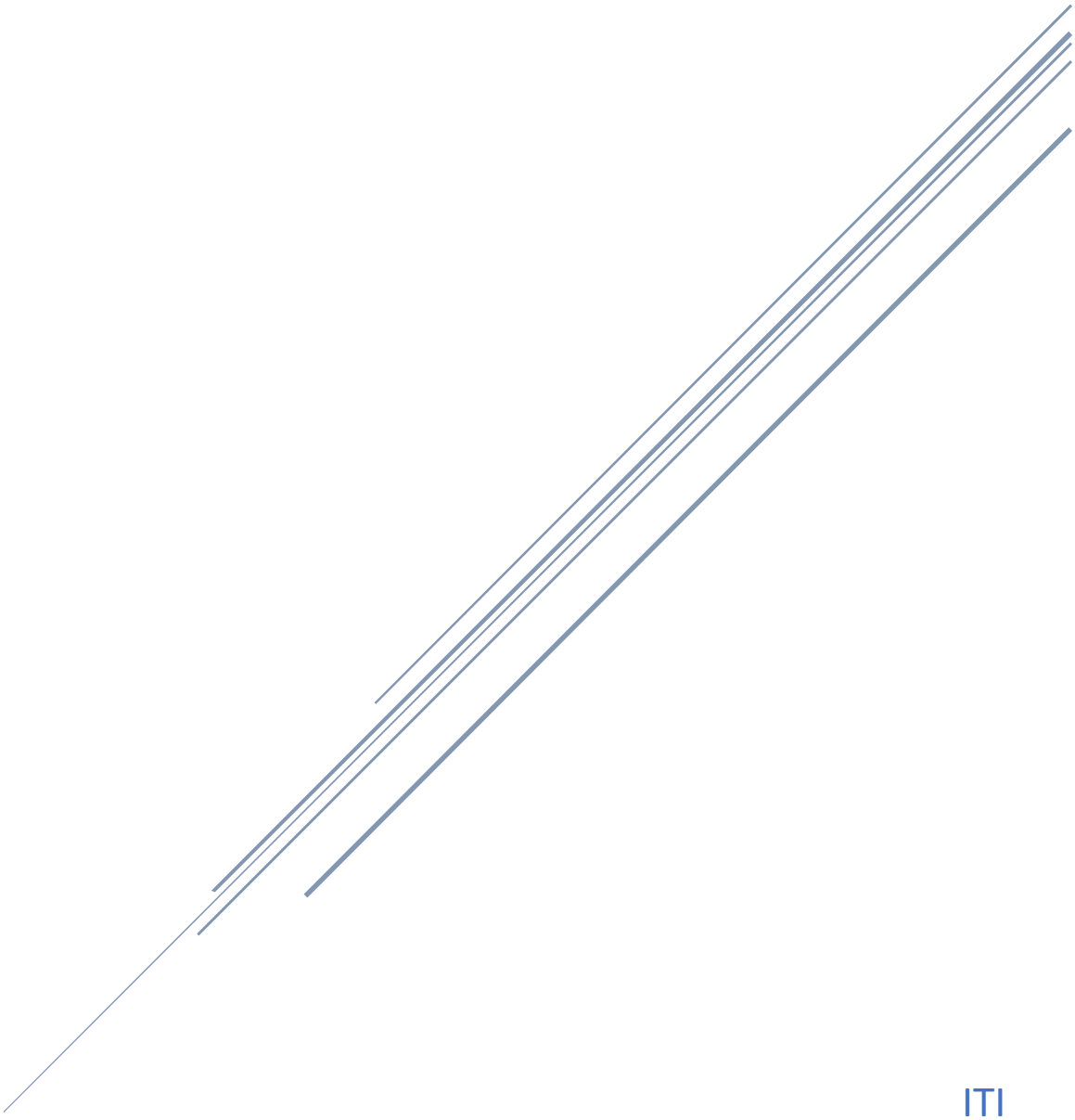
ANALYTICAL SQL

CASE STUDY



ITI

DATA INTEGRATION & VISULIZATION

To evaluate the data, specific questions must be asked so that the data may be revealed and the business can get insight.

Q1- Using Online Retail dataset

1- WHAT IS THE TOTAL SALES OF EACH MONTH?

- By comparing total sales across different months, businesses can identify which months are the most and the least successful in terms of revenue generation. As Jan is the least selling month, the business can reveal the reason behind that decrease and try to work on that issue by introduce new products/services to stimulate demand during those periods. And as Nov. is the most selling month it can reveal the impact of season on sales.

```
WITH cte_year_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'), 'month') 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'), 'month')
)
   SELECT year,
     (FIRST_VALUE(month) OVER (PARTITION BY sum(sales) ORDER BY month)) AS selling_month,
    sum(sales) total_sales
  from cte_year_sales
  group by year, month order by sum(sales);
```

YEAR	SELLING_MONTH	TOTAL_SALES
2011	january	9541.29
2011	april	10980.51
2011	december	11124.13
2011	february	13336.84
2010	december	13453.56
2011	june	13517.01
2011	july	15664.54
2011	march	17038.01
2011	may	19496.18
2011	october	19735.07
2011	september	27853.82
2011	august	38374.64
2011	november	45633.38

2- what is the least sales by product in each month and at Jan?

- The least sales by product in each month

```
WITH cte_year_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'), 'month') AS month,
      SUM(quantity * price) AS sales,
      stockcode
   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'), 'month'),
         stockcode
),
cte_least_sales_month AS (
   SELECT year, month, sum(sales) total_sales, stockcode,
        ROW_NUMBER() OVER (PARTITION BY month ORDER BY sum(sales)) AS rn
   FROM cte_year_sales
   group by year, month, stockcode
),
cte_least_selling_items as(
   SELECT year, month, stockcode, total_sales,
        ROW_NUMBER() OVER (PARTITION BY rn ORDER BY total_sales) AS product_rn
   FROM cte_least_sales_month )
   select* from cte_least_selling_items where product_rn = 1;
```

YEAR	MONTH	STOCKCODE	TOTAL_SALES	PRODUCT_RN
2011	september	21403	0.12	1
2010	december	10120	0.21	1
2011	december	22754	0.21	1
2011	december	16237	0.21	1
2011	december	21306	0.29	1
2011	december	22481	0.39	1
2011	december	85071D	0.39	1
2011	december	85071A	0.39	1
2011	december	22267	0.39	1
2011	december	22190	0.39	1
2011	december	22100	0.39	1
2011	december	22398	0.39	1
2011	december	85071B	0.39	1
2011	december	85071C	0.39	1
2011	december	21786	0.42	1
2011	december	21918	0.42	1
2011	december	22753	0.42	1
2011	december	84692	0.42	1
2010	december	79149B	0.42	1
2011	december	84199	0.42	1
2011	december	22419	0.42	1

- The least sales of product at Jan

```
 WITH cte_stockcode AS (
SELECT
    stockcode AS product_id,
    SUM(quantity * price) AS sales,
    TO_CHAR(TO_DATE(INVOICEDATE , 'MM/DD/YYYY HH24:MI'), 'dd') AS month
FROM tableretail
GROUP BY stockcode, TO_CHAR(TO_DATE(INVOICEDATE , 'MM/DD/YYYY HH24:MI'), 'dd')
)
SELECT
    product_id,
    month,
    SUM(sales) AS total_sales,
    DENSE_RANK() OVER (ORDER BY SUM(sales)) AS dr
FROM
    cte_stockcode
WHERE
    month = '01'
GROUP BY
    product_id,
    month
ORDER BY
    SUM(sales);
```

PRODUCT_ID	MONTH	TOTAL_SALES	DR
21084	01	0.19	1
21088	01	0.38	2
85071C	01	0.39	3
85071B	01	0.39	3
85071D	01	0.39	3
85071A	01	0.39	3
37413	01	0.39	3
22533	01	0.42	4
84692	01	0.42	4
22532	01	0.42	4
84199	01	0.42	4
22535	01	0.42	4
22998	01	0.42	4
22999	01	0.42	4
85032B	01	0.65	5
85032D	01	0.65	5
22335	01	0.65	5
22439	01	0.65	5
21386	01	0.76	6
47593B	01	0.78	7
22398	01	0.78	7
37327	01	0.78	7

- This question reveals the least sales by product so the business can stop producing them or trying to increase the sales of these products.

3- what is the most purchased products in Nov.?

- As Nov. has the highest sales so revealing the most purchased items would increase the sales as the business can prepare more quantity and can offer more discounts.

```
SELECT
    stockcode AS product_id,
    SUM(quantity * price) AS total_sales,
    dense_rank() over(ORDER BY SUM(quantity) DESC) AS rank
FROM tableretail
WHERE EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) = 11
GROUP BY stockcode;
```

PRODUCT_ID	TOTAL_SALES	RANK
84879	1991.81	1
22197	835.6	2
21787	918.85	3
23203	1664.9	4
21703	246.72	5
84378	987.8	6
23215	1303.2	7
21135	155.76	8
23084	1028.44	9
20974	213.15	10
23201	670.12	11
22071	176.5	12
22070	168	13
21790	195.94	14
21479	1214.25	15
16008	34.56	16
23310	118.44	17
84077	69.6	18
22564	230.21	19
22610	41.04	20

4- what is the most selling product not only in Nov.?
– this question reveals the most purchased product so that the business can prepare further quantity of these product.

```
SELECT
  stockcode AS product_id,
  SUM(quantity * price) AS total_sales,
  dense_rank() over(order by sum(quantity) desc) as Rank
FROM tableretail
GROUP BY stockcode order by rank;
```

PRODUCT_ID	TOTAL_SALES	RANK
84077	1788.96	1
84879	9114.69	2
22197	4323.1	3
21787	4059.35	4
21977	2063.69	5
21703	826.32	6
17096	343.23	7
15036	1329.36	8
23203	3357.44	9
21790	1011.67	10
22988	1730.53	11
23215	2697.36	12
20974	824.7	13
22992	2308.05	14
21731	1982.7	15
22693	1200.72	16
40016	335.28	17
22991	2047.05	18
23084	2187.72	19

5- what is the monthly customers growth rate?

- By tracking the number of customers each month and comparing it to the previous month, the business can assess its customer retention and growth rates over time.

```
  WITH customers_monthly AS (
   SELECT
      EXTRACT(YEAR FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) AS year,
      EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) AS month,
     count(customer_id) AS customers
   FROM tableRetail
   GROUP BY EXTRACT(YEAR FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')),
EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'))
),
customers_previous_month AS (
   SELECT
      year,
      month,
      customers,
      LAG(customers) OVER (ORDER BY year, month) AS previous_customer
   FROM customers_monthly
)
SELECT
   year,
   month,
   ROUND(customers) AS customers,
   ROUND(previous_customer) AS previous_customer,
   CASE
      WHEN previous_customer IS NULL THEN 0
      ELSE ROUND(CAST((customers - previous_customer) / previous_customer * 100 AS numeric), 2)
   END AS customers_growth_rate
FROM customers_previous_month
ORDER BY year, month;
```

YEAR	MONTH	CUSTOMERS	PREVIOUS_CUSTOMER	CUSTOMERS_GROWTH_RATE
2010	12	1139		0
2011	1	461	1139	-60
2011	2	556	461	21
2011	3	727	556	31
2011	4	479	727	-34
2011	5	983	479	105
2011	6	745	983	-24
2011	7	743	745	0
2011	8	595	743	-20
2011	9	1411	595	137
2011	10	1003	1411	-29
2011	11	3200	1003	219
2011	12	817	3200	-74

6- What is the average time between purchases for customers?

- Understanding the purchasing frequency enables businesses to tailor marketing campaigns more effectively. They can time promotions, discounts, and advertisements to coincide with when customers are most likely to make a purchase, thereby increasing conversion rates and ROI on marketing efforts.

```
WITH cte_invoice_date AS (
    SELECT
        Customer_ID,
        TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI') as invoice_date
    FROM tableretail
),
PurchaseGaps AS (
    SELECT
        customer_id,
        invoice_date,
        LAG(invoice_date) OVER (PARTITION BY Customer_ID ORDER BY invoice_date) AS
PreviousPurchaseDate,
        (invoice_date - LAG(invoice_date) OVER (PARTITION BY Customer_ID ORDER BY invoice_date))
AS TimeBetweenPurchases
    FROM cte_invoice_date
),
CustomerProductSales AS (
    SELECT
        Customer_ID,
        stockcode as product_id,
        COUNT(*) AS ProductPurchaseCount
    FROM tableretail
    GROUP BY Customer_ID, stockcode
),
RankedProducts AS (
    SELECT
        Customer_ID,
        Product_ID,
        ProductPurchaseCount,
        RANK() OVER (PARTITION BY Customer_ID ORDER BY ProductPurchaseCount DESC) AS
ProductRank
    FROM CustomerProductSales
)
SELECT
    pg.Customer_ID,
    ROUND(AVG(pg.TimeBetweenPurchases)) AS AverageTimeBetweenPurchases,
    rp.Product_ID AS MostSellingProduct,
    rp.ProductPurchaseCount
FROM PurchaseGaps pg
JOIN RankedProducts rp ON pg.Customer_ID = rp.Customer_ID AND rp.ProductRank = 1
WHERE pg.PreviousPurchaseDate IS NOT NULL
GROUP BY pg.Customer_ID, rp.Product_ID, rp.ProductPurchaseCount;
```

CUSTOMER_ID	AVERAGETIMEBETWEENPURCHASES	MOSTSELLINGPRODUCT	PRODUCTPURCHASECOUNT
12820	6	84946	2
12824	0	21506	1
12824	0	23101	1
12824	0	23320	1
12824	0	85048	1
12827	2	22139	3
12830	4	21703	4
12832	3	46000R	1
12833	0	35004B	1
12833	0	21135	1
12833	0	22171	1
12834	0	22249	1
12836	1	21733	3
12836	1	21755	3
12837	0	22847	1
12837	0	23046	1
12837	0	22844	1
12842	7	22991	1
12842	7	22919	1
12853	2	22171	3
12857	2	20914	2

7- Do the sales always rise near the holiday season for all years?

- The question reveals whether there is a relation between increasing sales and the holiday season or not.

```
WITH holiday_sales AS (
    SELECT
        EXTRACT(YEAR FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) AS sales_year,
        EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) AS sales_month,
        SUM(quantity * price) AS total_sales
    FROM
        tableretail
    WHERE
        EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) IN (12, 11)
    GROUP BY
        EXTRACT(YEAR FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')),
        EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'))
),
average_monthly_sales AS (
    SELECT
        EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) AS month,
        AVG(quantity * price) AS avg_sales
    FROM
        tableretail
```

```
    GROUP BY
        EXTRACT(MONTH FROM TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'))
)
SELECT
    hs.sales_year,
    hs.sales_month,
    hs.total_sales,
    ams.avg_sales,
    CASE
        WHEN hs.total_sales > ams.avg_sales THEN 'Above Average'
        WHEN hs.total_sales < ams.avg_sales THEN 'Below Average'
        ELSE 'Equal to Average'
    END AS sales_comparison
FROM
    holiday_sales hs
JOIN
    average_monthly_sales ams ON hs.sales_month = ams.month
ORDER BY
    hs.sales_year, hs.sales_month;
```

SALES_YEAR	SALES_MONTH	TOTAL_SALES	AVG_SALES	SALES_COMPARISON
2010	12	13453.56	12.5652811860941	Above Average
2011	11	45633.38	14.26043125	Above Average
2011	12	11124.13	12.5652811860941	Above Average

- This pattern indicates that customers tend to spend more during these festive periods, likely due to factors such as gift-giving, promotions, and seasonal discounts, so this allows businesses to better forecast revenue and plan inventory, staffing, and marketing strategies accordingly.

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

```
WITH cte_customers AS (
   SELECT
   customer_id,
round(
 (SELECT MAX(TO_DATE(INVOICEDATE , 'MM/DD/YYYY HH24:MI')) FROM tableretail ) -
   MAX(TO_DATE(INVOICEDATE , 'MM/DD/YYYY HH24:MI'))) AS RECENCY,
   COUNT(DISTINCT INVOICEdate) AS FREQUENCY,
   SUM(quantity * price) AS Monetary
FROM tableretail
GROUP BY customer_id
),
cte_r_rm AS (
   SELECT
      customer_id,
      RECENCY,
      FREQUENCY,
      Monetary,
      NTILE(5) OVER(ORDER BY Recency DESC) AS R_Score,
    round ((NTILE(5) OVER(ORDER BY AVG(frequency) DESC) + NTILE(5) OVER(ORDER BY
AVG(Monetary) DESC))/2) AS F_M_Score
   FROM cte_customers
   group by customer_id, RECENCY, FREQUENCY, Monetary
)
SELECT
   customer_id,
   RECENCY,
   FREQUENCY,
   Monetary,
   R_Score,
   F_M_Score,
   CASE
         WHEN R_Score = 5 AND F_M_Score IN (5, 4) THEN 'Champions'
         WHEN R_Score = 4 AND F_M_Score = 5 THEN 'Champions'
         WHEN R_Score = 5 AND F_M_Score = 2 THEN 'Potential Loyalists'
         WHEN R_Score = 4 AND F_M_Score in (2 , 3) THEN 'Potential Loyalists'
         WHEN R_Score = 3 AND F_M_Score = 3 THEN 'Potential Loyalists'
```

```
        WHEN R_Score = 5 AND F_M_Score = 3 THEN 'Loyal Customers'
        WHEN R_Score = 4 AND F_M_Score = 4 THEN 'Loyal Customers'
        WHEN R_Score = 3 AND F_M_Score in (4 , 5) THEN 'Loyal Customers'
        WHEN R_Score = 5 AND F_M_Score = 1 THEN 'Recent Customers'
        WHEN R_Score = 4 AND F_M_Score = 1 THEN 'Promising'
        WHEN R_Score = 3 AND F_M_Score = 1 THEN 'Promising'
        WHEN R_Score = 3 AND F_M_Score = 2 THEN 'Customers Needing Attention'
        WHEN R_Score = 2 AND F_M_Score IN (2, 3) THEN 'Customers Needing Attention'
         WHEN R_Score = 1 AND F_M_Score = 3 THEN 'At Risk'
        WHEN R_Score = 2 AND F_M_Score IN (4, 5) THEN 'At Risk'
        WHEN R_Score = 1 AND F_M_Score = 2 THEN 'Hibernating'
        WHEN R_Score = 1 AND F_M_Score IN (4, 5) THEN 'Cant Lose Them'
        WHEN R_Score = 1 AND F_M_Score = 1 THEN 'Lost'
        ELSE 'Undefined'
    END AS Customer_Segment
FROM cte_r_rm
order by customer_id;
```

CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	F_M_SCORE	CUSTOMER_SEGMENT
12747	2	11	4226.61	5	1	Recent Customers
12748	0	211	33719.73	5	1	Recent Customers
12749	3	5	4090.88	5	2	Potential Loyalists
12820	3	4	942.34	5	3	Loyal Customers
12821	214	1	92.72	1	5	Cant Lose Them
12822	70	2	948.88	3	4	Loyal Customers
12823	74	5	1759.5	2	2	Customers Needing Attention
12824	59	1	397.12	3	5	Loyal Customers
12826	2	7	1474.72	5	2	Potential Loyalists
12827	5	3	430.15	5	4	Champions
12828	2	6	1018.71	5	3	Loyal Customers
12829	336	2	293	1	5	Cant Lose Them
12830	37	6	6814.64	3	1	Promising
12831	262	1	215.05	1	5	Cant Lose Them
12832	32	2	383.03	3	4	Loyal Customers
12833	145	1	417.38	2	4	At Risk
12834	282	1	312.38	1	5	Cant Lose Them
12836	59	5	2612.86	3	2	Customers Needing Attention
12837	173	1	134.1	2	5	At Risk
12838	33	2	683.13	3	3	Potential Loyalists
12839	2	14	5591.42	5	1	Recent Customers
12840	143	4	2726.77	2	2	Customers Needing Attention

Q3- You are given the below dataset, which is the daily purchasing transactions for customers.

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

```
WITH ranked_transactions AS (
    SELECT
        cust_id,
        Calendar_Dt,
        ROW_NUMBER() OVER (PARTITION BY cust_id ORDER BY Calendar_Dt) AS rn
    FROM
        customertransactions
),
transaction_diffs AS (
    SELECT
        cust_id,
        Calendar_Dt,
        Calendar_Dt - rn AS date_diff
    FROM
        ranked_transactions
)
SELECT
    cust_id,
    MAX(consecutive_days) AS max_consecutive_days
FROM (
    SELECT
        cust_id,
        COUNT(date_diff) AS consecutive_days
    FROM
        transaction_diffs
    GROUP BY
        cust_id, date_diff
)
GROUP BY
    cust_id
ORDER BY cust_id;
```

CUST_ID	MAX_CONSECUTIVE_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
247965	2
259866	8
272472	36
303984	5
324080	8
339749	12
369391	44

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

```
WITH customer_transactions_total AS (
    SELECT
        cust_id,
        calendar_dt,
        SUM(amt_le) OVER (PARTITION BY cust_id ORDER BY calendar_dt) AS total_spent
    FROM
        customertransactions
),
low_spending_customers AS (
    SELECT
        cust_id,
        calendar_dt
    FROM
        customer_transactions_total
    WHERE
        total_spent < 250
),
high_spending_customers AS (
    SELECT
        cust_id,
        calendar_dt,
        total_spent
    FROM
        customer_transactions_total
    WHERE
        total_spent >= 250
),
low_spending_customer_days AS (
    SELECT
        cust_id,
        COUNT(calendar_dt) AS days
    FROM
        low_spending_customers
    GROUP BY
        cust_id
)
SELECT
   ROUND(AVG(days)) AS average_days
FROM
    low_spending_customer_days
WHERE
    cust_id IN (SELECT cust_id FROM high_spending_customers);
```

AVERAGE_DAYS
6