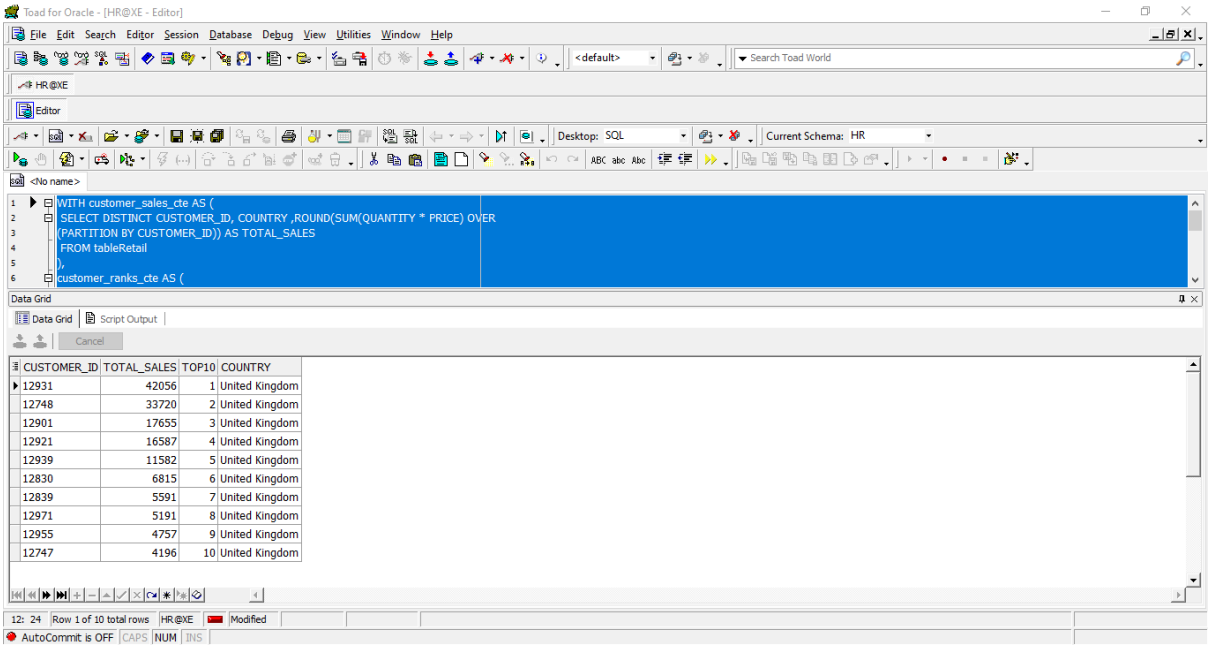
Q1

1-

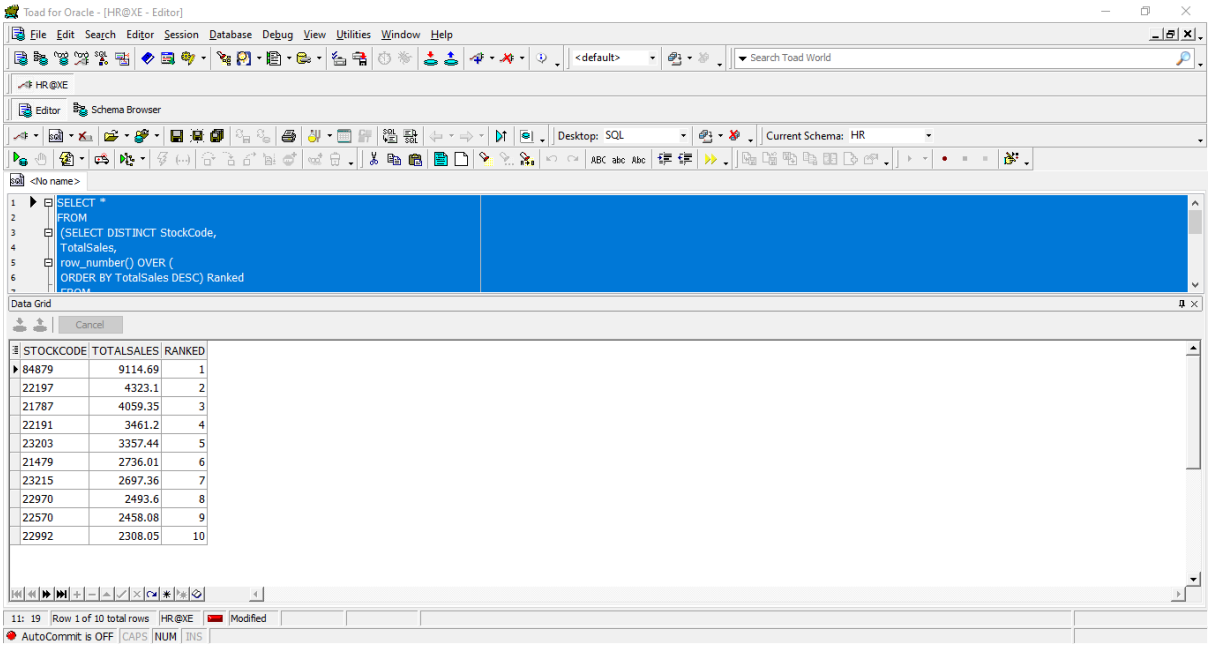
```
WITH customer_sales_cte AS (
 SELECT DISTINCT CUSTOMER_ID, COUNTRY ,ROUND(SUM(QUANTITY * PRICE) OVER
(PARTITION BY CUSTOMER_ID)) AS TOTAL_SALES
 FROM tableRetail
),
customer_ranks_cte AS (
 SELECT CUSTOMER_ID,COUNTRY ,TOTAL_SALES, RANK() OVER (ORDER BY TOTAL_SALES DESC)
AS TOP10
 FROM customer_sales_cte
)
SELECT CUSTOMER_ID, TOTAL_SALES , TOP10 ,COUNTRY
FROM customer_ranks_cte
WHERE TOP10 <= 10;
```



Here I want to determine the top 10 customers' sales to give them some vouchers.

2-

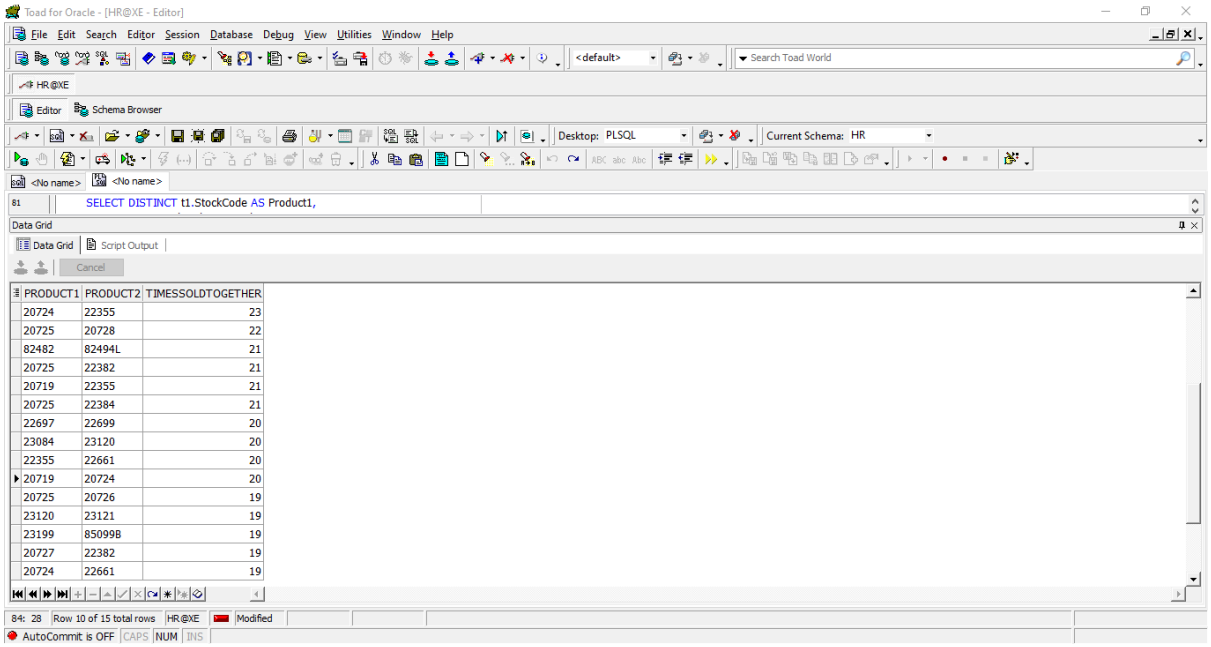
```
SELECT *
FROM
 (SELECT DISTINCT StockCode,
 TotalSales,
 row_number() OVER (
 ORDER BY TotalSales DESC) Ranked
 FROM
 (SELECT DISTINCT StockCode,
 sum(Quantity * price) OVER (PARTITION BY StockCode) AS TotalSales
 FROM tableRetail))
WHERE Ranked <= 10
ORDER BY Ranked;
```



Here I want to determine the top 10 products' sales as they're high in demand.

3-

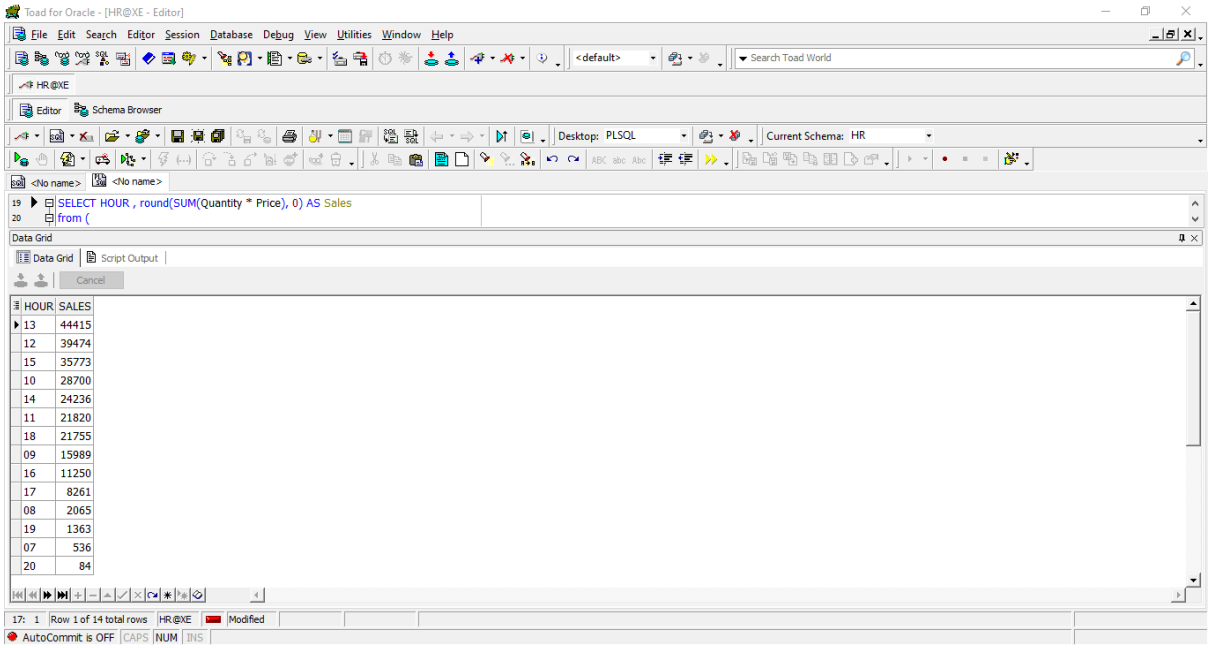
```
SELECT Product1, Product2, TimesSoldTogether
FROM (
   SELECT Product1, Product2, TimesSoldTogether,
         DENSE_RANK() OVER (ORDER BY TimesSoldTogether DESC) AS rn
   FROM (
      SELECT DISTINCT t1.StockCode AS Product1,
                  t2.StockCode AS Product2,
                  COUNT(*) OVER (PARTITION BY t1.StockCode, t2.StockCode) AS
TimesSoldTogether
      FROM tableRetail t1
      JOIN tableRetail t2 ON t1.Invoice = t2.Invoice
                     AND t1.StockCode < t2.StockCode
   )
)
WHERE rn <= 5;
```



Here I want to determine the top 5 products' sales as they're sold together, which could help me taking business decisions.

4-

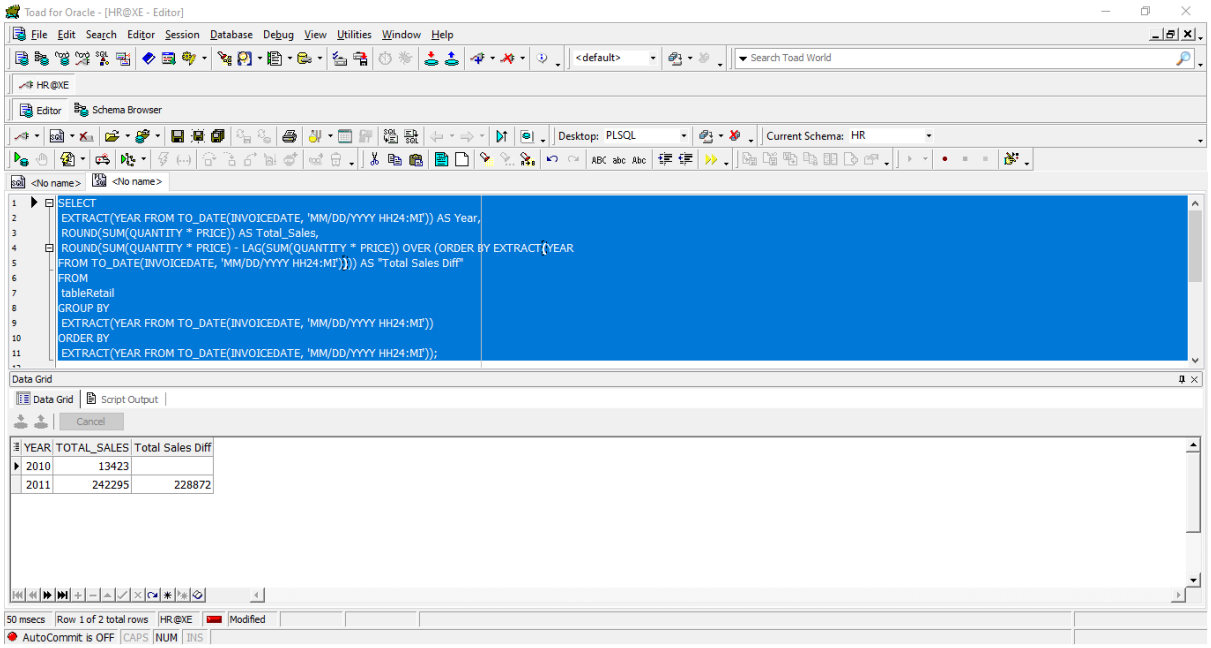
```
SELECT HOUR , round(SUM(Quantity * Price), 0) AS Sales
from (
 SELECT TO_CHAR(TO_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'), 'HH24') AS HOUR ,
 Quantity , Price
FROM tableRetail )
GROUP BY HOUR
ORDER BY sales DESC ;
```



Here I want to know the total sales per hour to determine the lowest and highest sales time which benefits me in taking business decisions.

5-

```
SELECT
 EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) AS Year,
 ROUND(SUM(QUANTITY * PRICE)) AS Total_Sales,
 ROUND(SUM(QUANTITY * PRICE) - LAG(SUM(QUANTITY * PRICE)) OVER (ORDER BY
EXTRACT(YEAR
FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')))) AS "Total Sales Diff"
FROM
 tableRetail
GROUP BY
 EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'))
ORDER BY
 EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'));
```



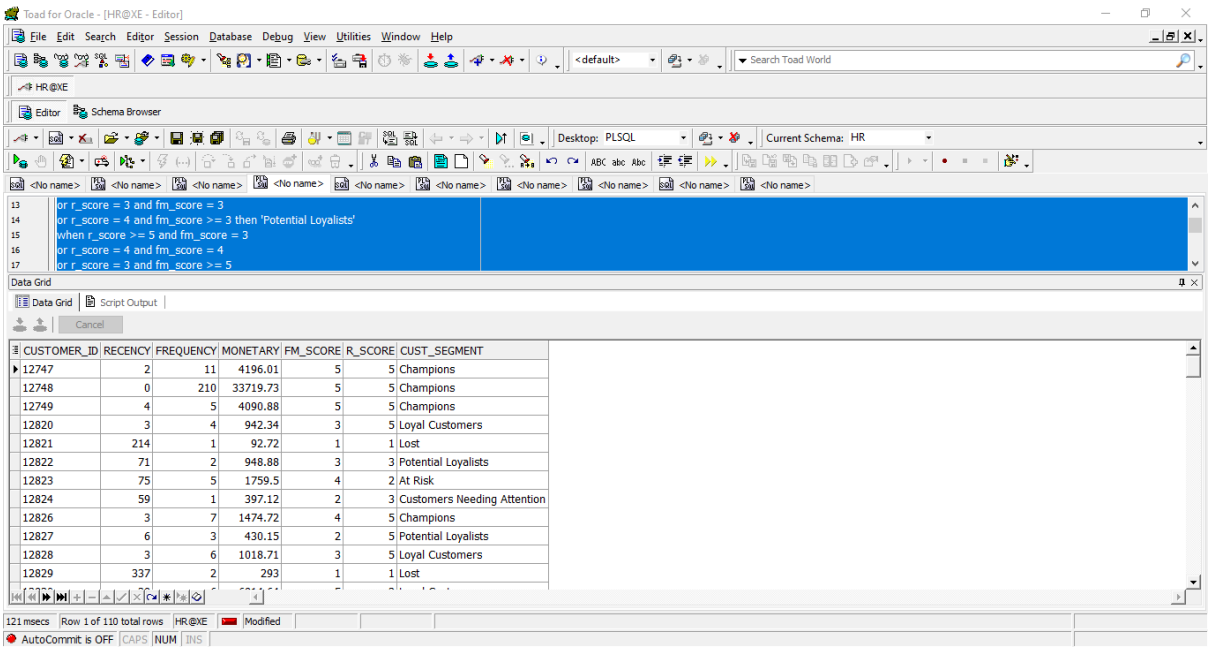
Here I want to know the total sales per Year and the sales difference between every year and the recent year to know if we're growing or the sales have decreased.

Q2

```
select CUSTOMER_ID,
 recency,
 frequency,
 monetary,
 fm_score ,
 r_score
 , Case
when r_score >= 5 and fm_score >= 5
or r_score >= 5 and fm_score =4
or r_score = 4 and fm_score >= 5 then 'Champions'
when r_score >= 5 and fm_score = 2
or r_score = 4 and fm_score = 2
or r_score = 3 and fm_score = 3
or r_score = 4 and fm_score >= 3 then 'Potential Loyalists'
when r_score >= 5 and fm_score = 3
or r_score = 4 and fm_score = 4
or r_score = 3 and fm_score >= 5
or r_score = 3 and fm_score >= 4 then 'Loyal Customers'
when r_score >= 5 and fm_score = 1 then 'Recent Customers'
when r_score = 4 and fm_score = 1
or r_score = 3 and fm_score = 1 then 'Promising'
when r_score = 3 and fm_score = 2
or r_score = 2 and fm_score = 3
or r_score = 2 and fm_score = 2 then 'Customers Needing Attention'
when r_score = 2 and fm_score >= 5
or r_score = 2 and fm_score = 4
or r_score = 1 and fm_score = 3 then 'At Risk'
when r_score = 1 and fm_score >= 5
or r_score = 1 and fm_score = 4 then 'Cant Lose Them'
when r_score = 1 and fm_score = 2
or r_score = 2 and fm_score = 1 then 'Hibernating'
when r_score = 1 and fm_score <= 1 then 'Lost'
End cust_segment
from
(
SELECT CUSTOMER_ID , recency , frequency,
 monetary,
 NTILE(5) OVER (ORDER BY recency desc) AS r_score ,
 NTILE(5) OVER (ORDER BY (frequency + monetary)/2 ) AS fm_score
from (
SELECT DISTINCT
 CUSTOMER_ID,
 FIRST_VALUE(DAYS_BETWEEN_INVOICES IGNORE NULLS) OVER (PARTITION BY CUSTOMER_ID
ORDER
BY DAYS_BETWEEN_INVOICES ASC) AS recency,
 frequency,
 monetary

FROM
 (
 SELECT DISTINCT
 CUSTOMER_ID,
```

```
 CEIL(FIRST_VALUE(TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) OVER (ORDER BY
TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI') DESC) - TO_DATE(INVOICEDATE, 'MM/DD/YYYY
HH24:MI')) AS DAYS_BETWEEN_INVOICES,
 SUM(price *quantity) OVER (PARTITION BY CUSTOMER_ID) AS monetary,
 COUNT(DISTINCT INVOICE) OVER (PARTITION BY CUSTOMER_ID ) AS frequency
 FROM
 tableRetail
 ORDER BY
 CUSTOMER_ID )
)
ORDER BY
 CUSTOMER_ID );
```



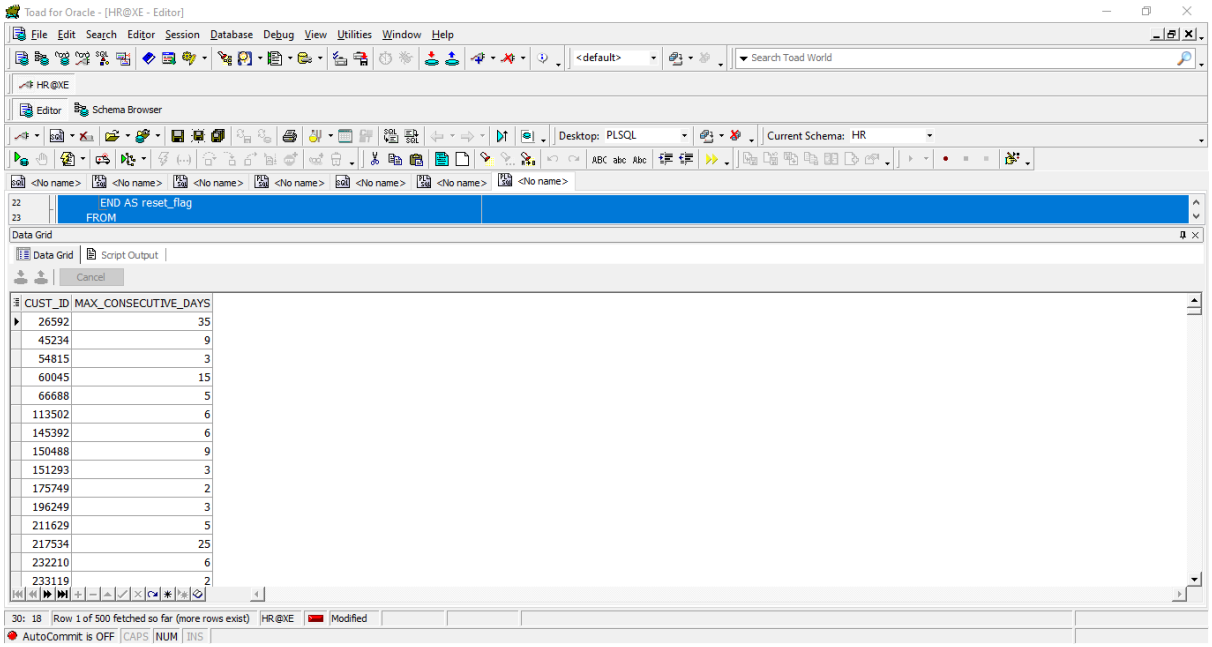
CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	FM_SCORE	R_SCORE	CUST_SEGMENT
12747	2	11	4196.01	5	5	Champions
12748	0	210	33719.73	5	5	Champions
12749	4	5	4090.88	5	5	Champions
12820	3	4	942.34	3	5	Loyal Customers
12821	214	1	92.72	1	1	Lost
12822	71	2	948.88	3	3	Potential Loyalists
12823	75	5	1759.5	4	2	At Risk
12824	59	1	397.12	2	3	Customers Needing Attention
12826	3	7	1474.72	4	5	Champions
12827	6	3	430.15	2	5	Potential Loyalists
12828	3	6	1018.71	3	5	Loyal Customers
12829	337	2	293	1	1	Lost

Q3

1-

```
SELECT CUST_ID, MAX(cons_days) as max_consecutive_days
FROM (
 SELECT CUST_ID, COUNT(*) AS cons_days
 FROM (
SELECT
    CUST_ID,
    order_date,
    ROW_NUMBER() OVER (PARTITION BY CUST_ID, grp ORDER BY order_date) AS rn
     , grp
FROM (
    SELECT
        CUST_ID,
        CALENDAR_DT AS order_date,
        SUM(reset_flag) OVER (PARTITION BY CUST_ID ORDER BY CALENDAR_DT) AS grp
    FROM (
        SELECT
```

```
        CUST_ID,
        CALENDAR_DT,
        CASE
            WHEN CALENDAR_DT - LAG(CALENDAR_DT) OVER (PARTITION BY CUST_ID ORDER BY
CALENDAR_DT) > 1 THEN 1
            ELSE 0
        END AS reset_flag
    FROM
        CUSTOMERS
  )
)
 )
 GROUP BY CUST_ID , grp
)
GROUP BY CUST_ID
order by CUST_ID;
```



2-

```
WITH daily_spending AS (
 SELECT
 CUST_ID,
 CALENDAR_DT,
 SUM(AMT_LE) OVER (PARTITION BY CUST_ID ORDER BY CALENDAR_DT) AS total_spending
 FROM
 CUSTOMERS
),
threshold_unreached AS (
 SELECT
 CUST_ID,
 CALENDAR_DT,
 total_spending
 FROM
 daily_spending
```

```
 WHERE
 total_spending < 250
),
threshold_reached AS (
 SELECT
 CUST_ID,
 CALENDAR_DT,
 total_spending
 FROM
 daily_spending
 WHERE
 total_spending >= 250
),
avg_days as (SELECT
 CUST_ID,
 COUNT( CALENDAR_DT) +1 AS days_to_reach_threshold
FROM
 threshold_unreached
where CUST_ID in (select CUST_ID from threshold_reached )
GROUP BY
 CUST_ID
order by CUST_ID )
SELECT round (avg(days_to_reach_threshold),2) as average_days from avg_days ;
```