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

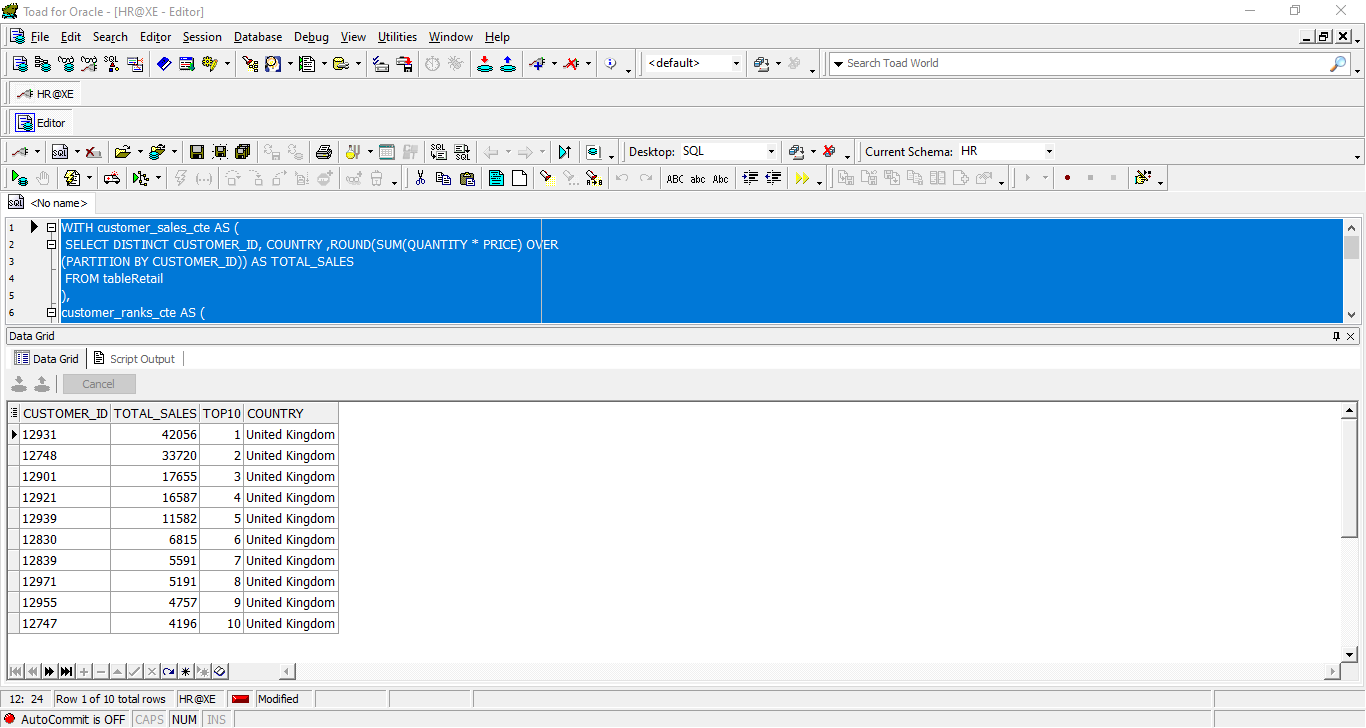
FROM customer\_sales\_cte

)

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

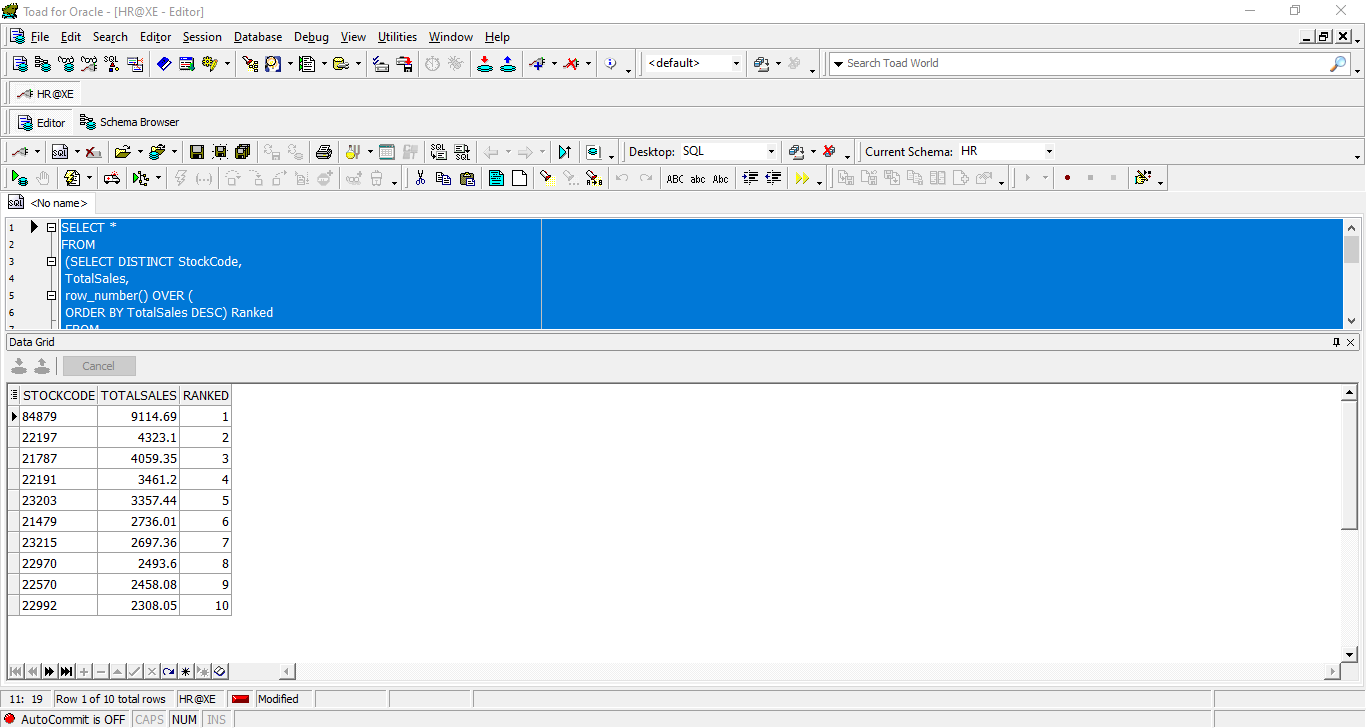
(SELECT DISTINCT StockCode,

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

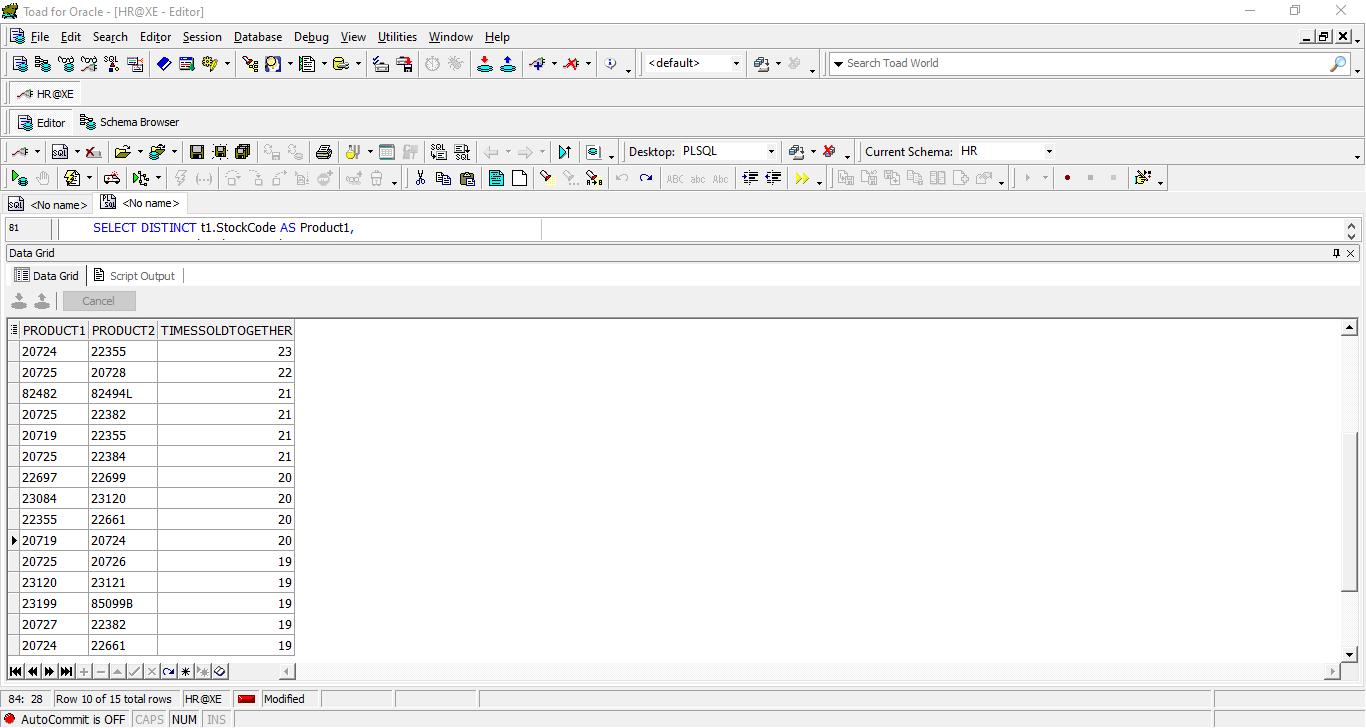
JOIN tableRetail t2 ON t1.Invoice = t2.Invoice

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 (

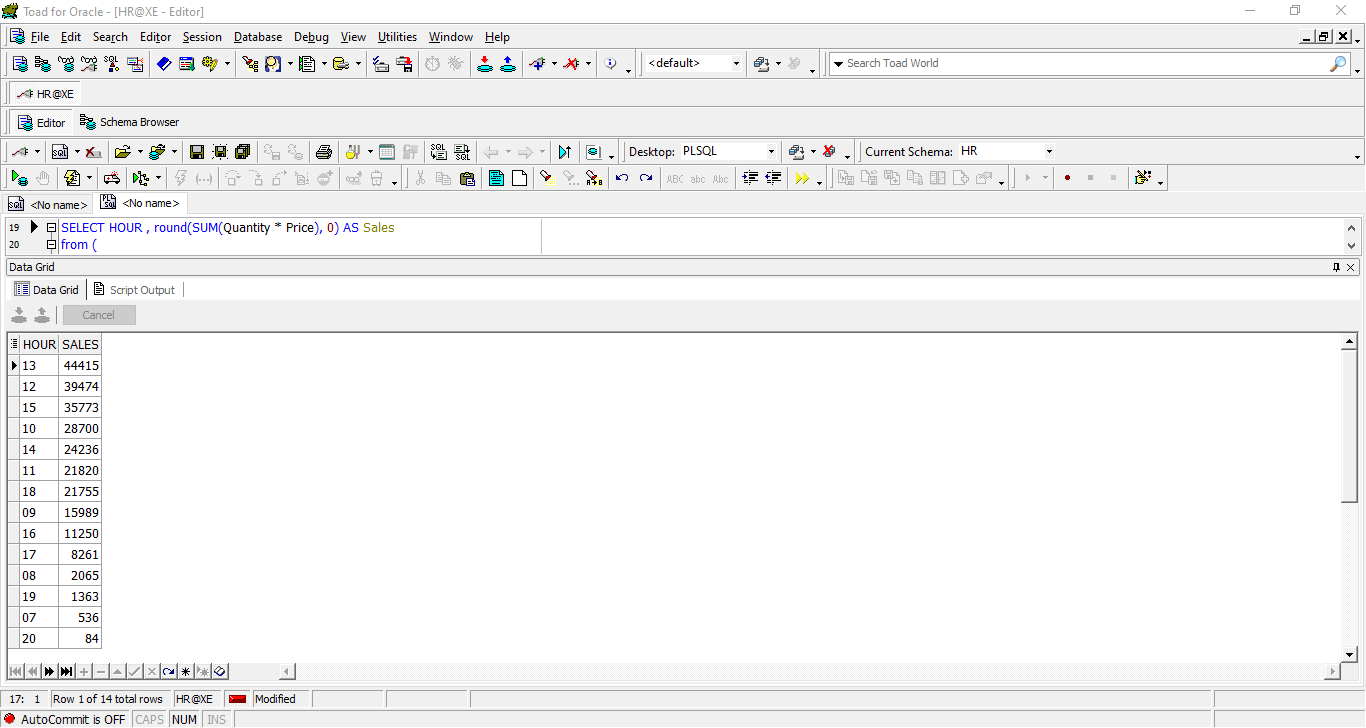
SELECT TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'), 'HH24') AS HOUR ,

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

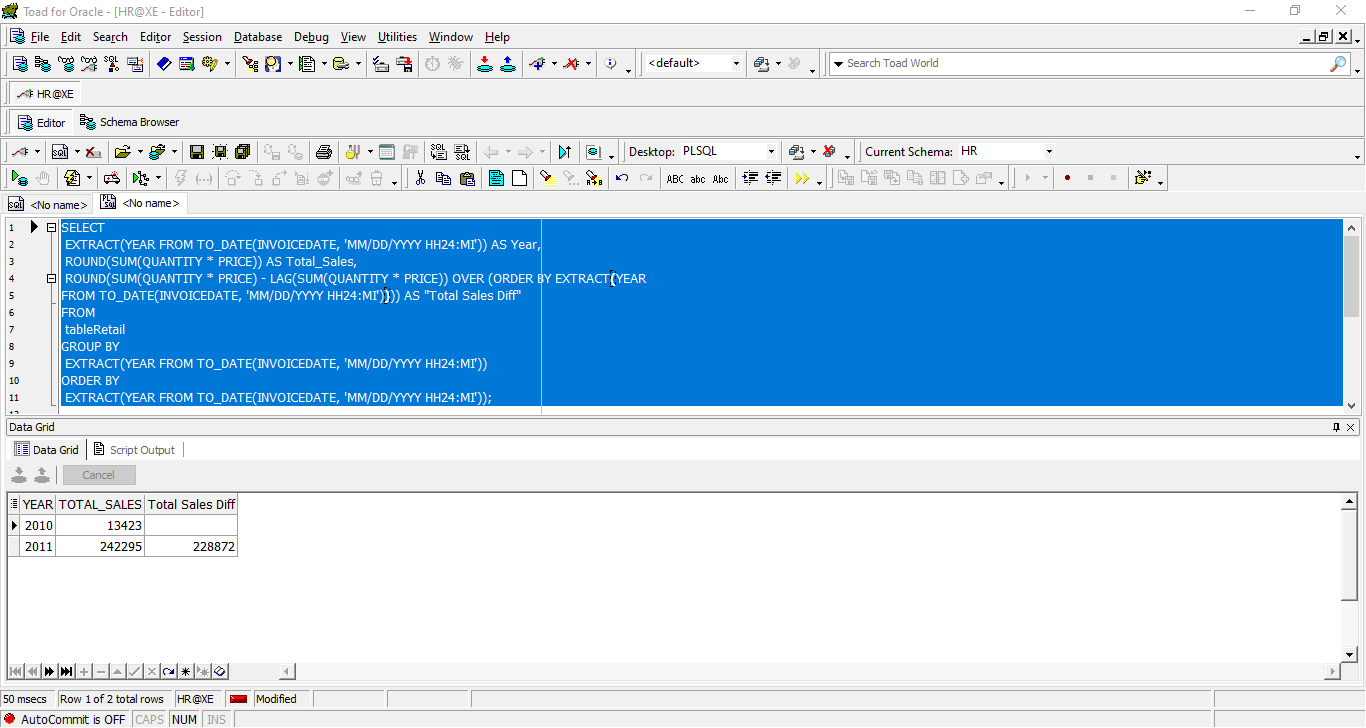
tableRetail

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

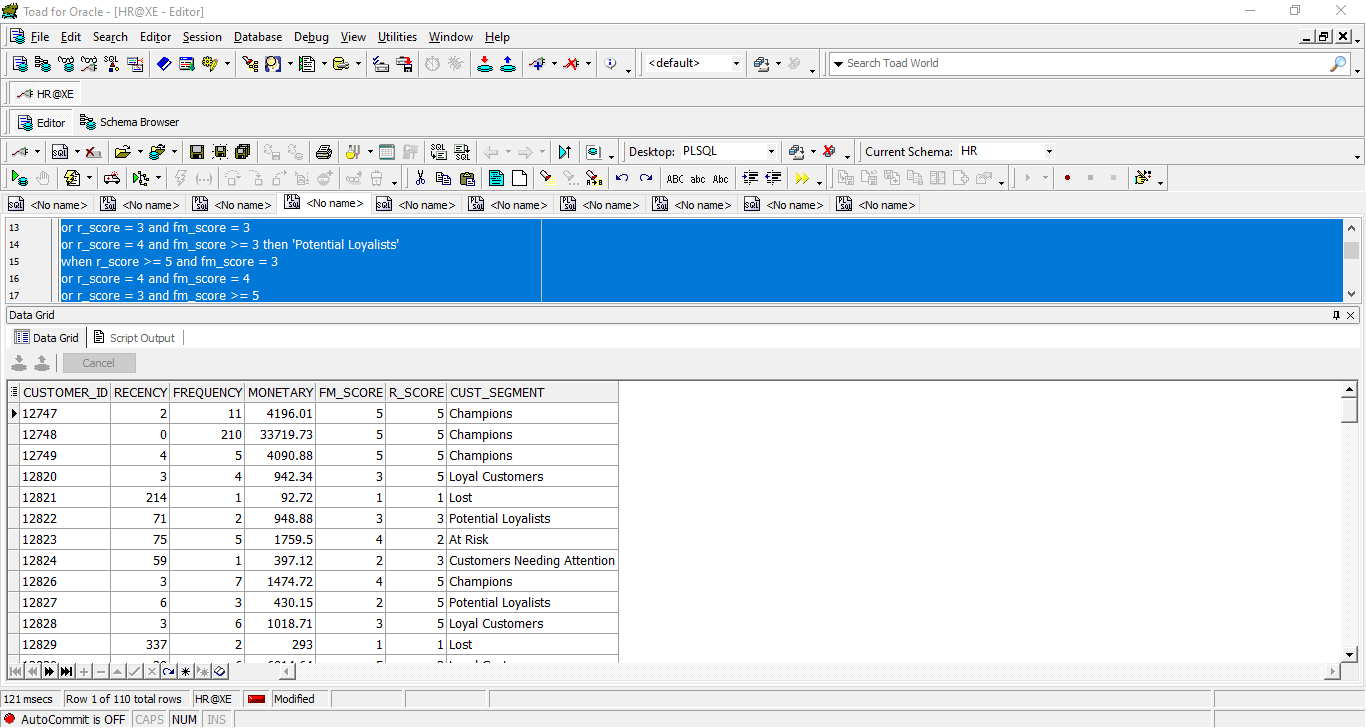
ORDER BY

CUSTOMER\_ID )

)

ORDER BY

CUSTOMER\_ID );



-------------------------------------------------------------------------------------------------

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

)

)

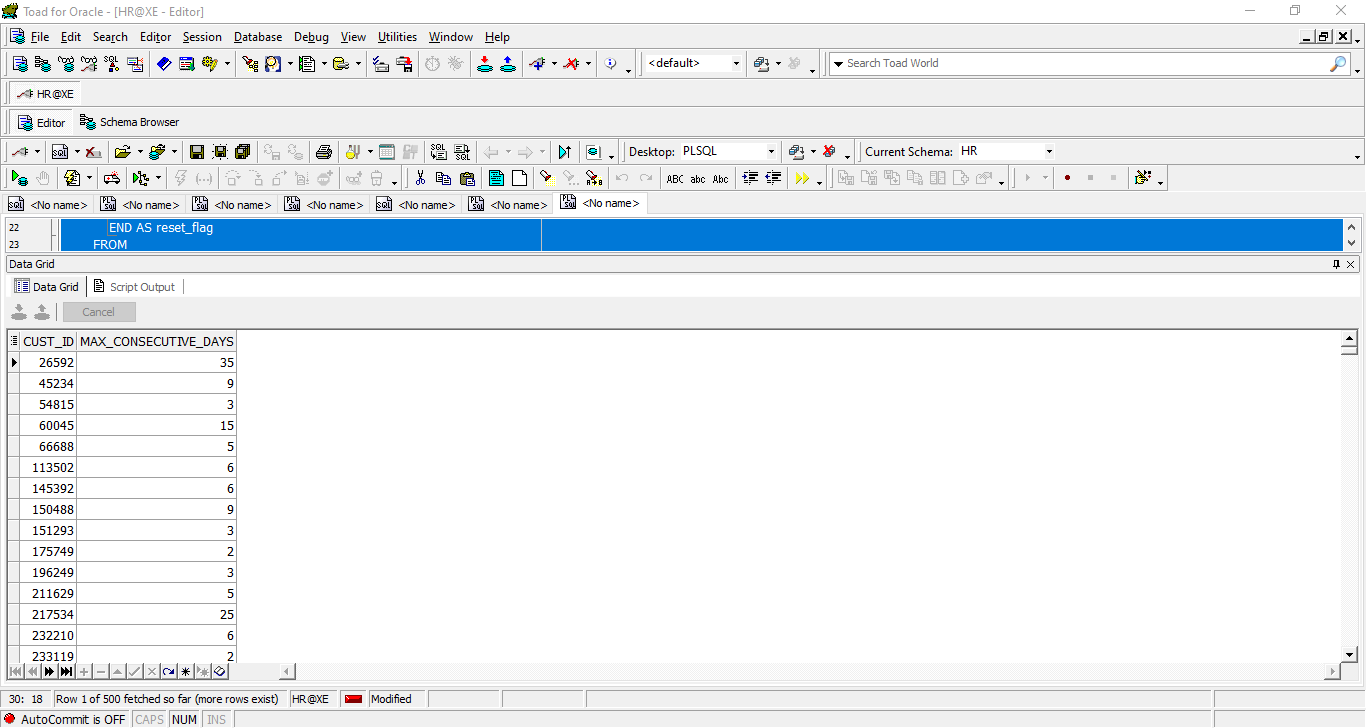
)

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 ;

A screenshot of a computer

Description automatically generated