**Q1**

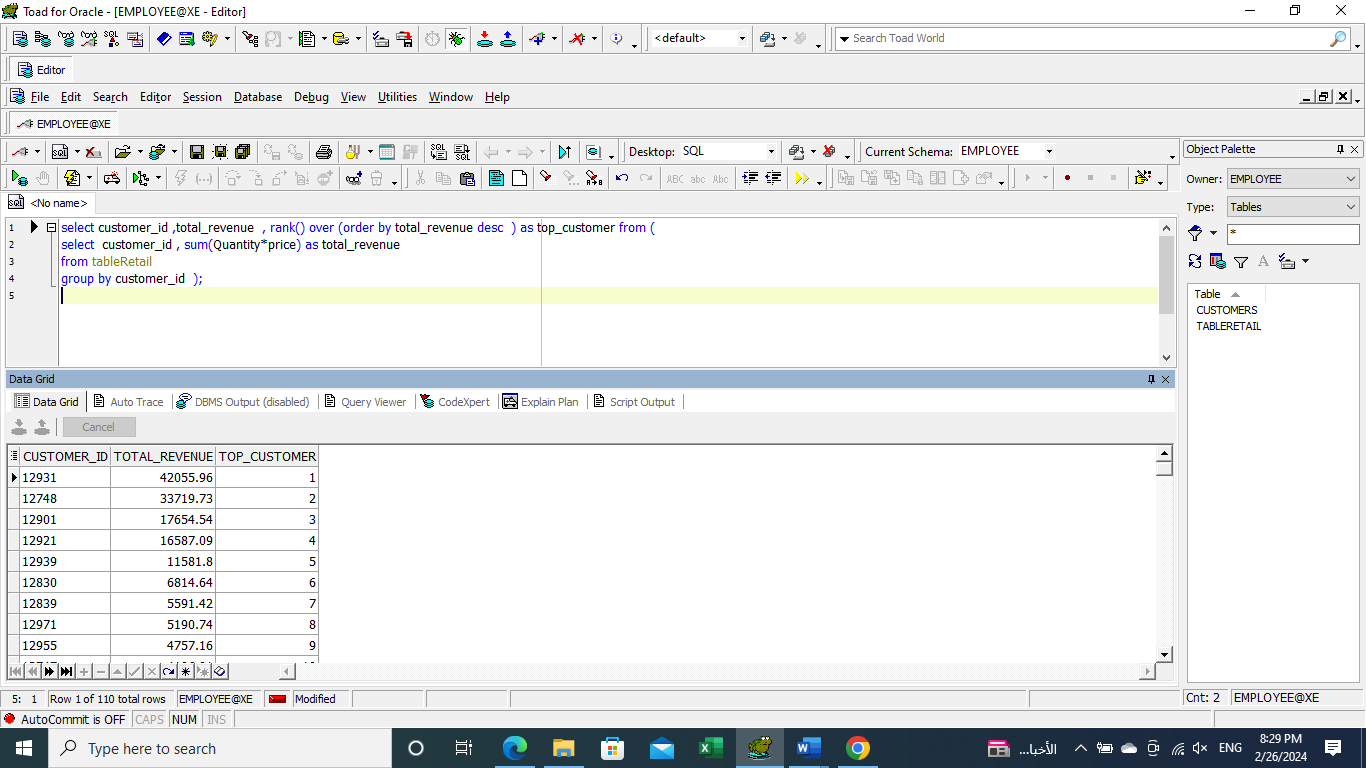
1. **Total Revenue per Customer with Ranking**: This query calculates the total revenue generated by each customer and ranks them based on their total revenue. It helps identify high-value customers.

select customer\_id ,total\_revenue , rank() over (order by total\_revenue desc ) as top\_customer from (

select customer\_id , sum(Quantity\*price) as total\_revenue

from tableRetail

group by customer\_id );



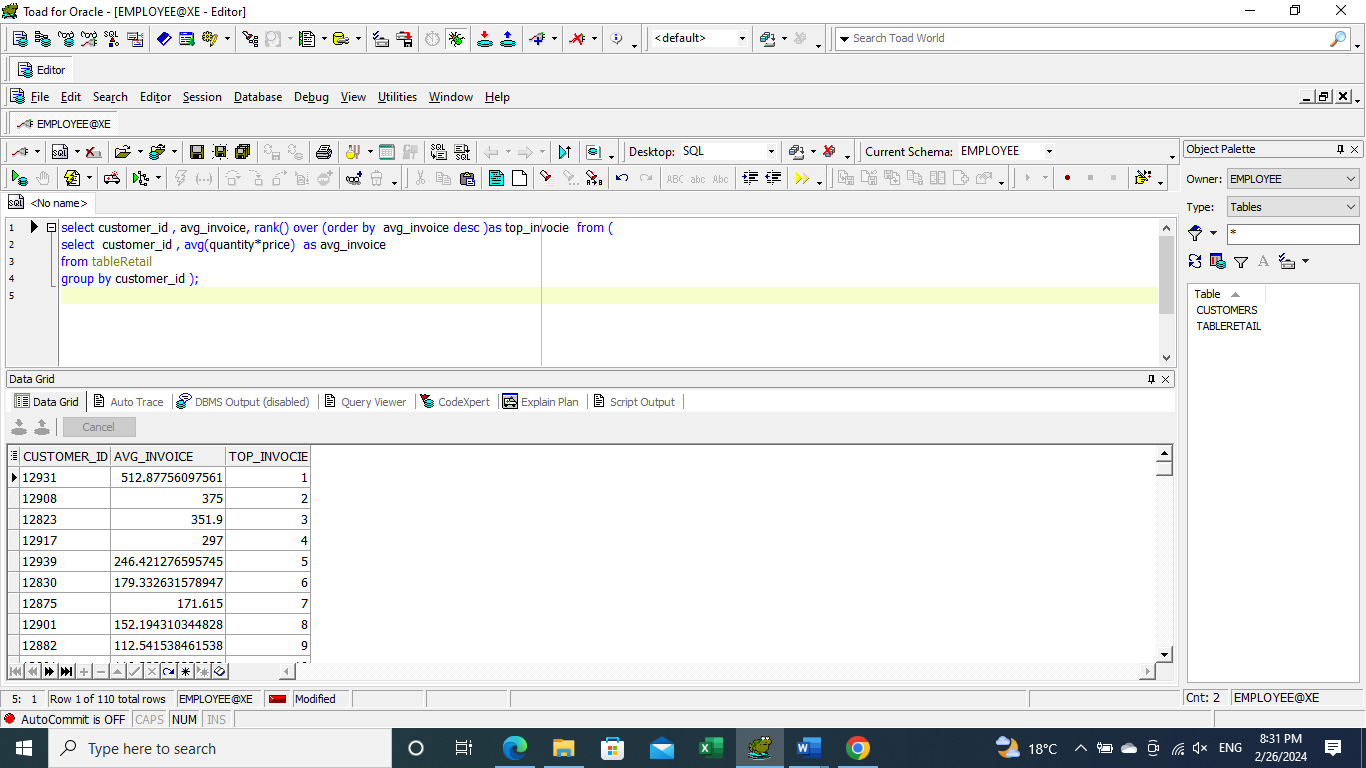
1. **Average Invoice per Customer with Ranking**: This query calculates the average invoice amount for each customer and ranks them based on their average invoice amount. It helps understand spending habits.

select customer\_id , avg\_invoice, rank() over (order by avg\_invoice desc )as top\_invocie from (

select customer\_id , avg(quantity\*price) as avg\_invoice

from tableRetail

group by customer\_id );

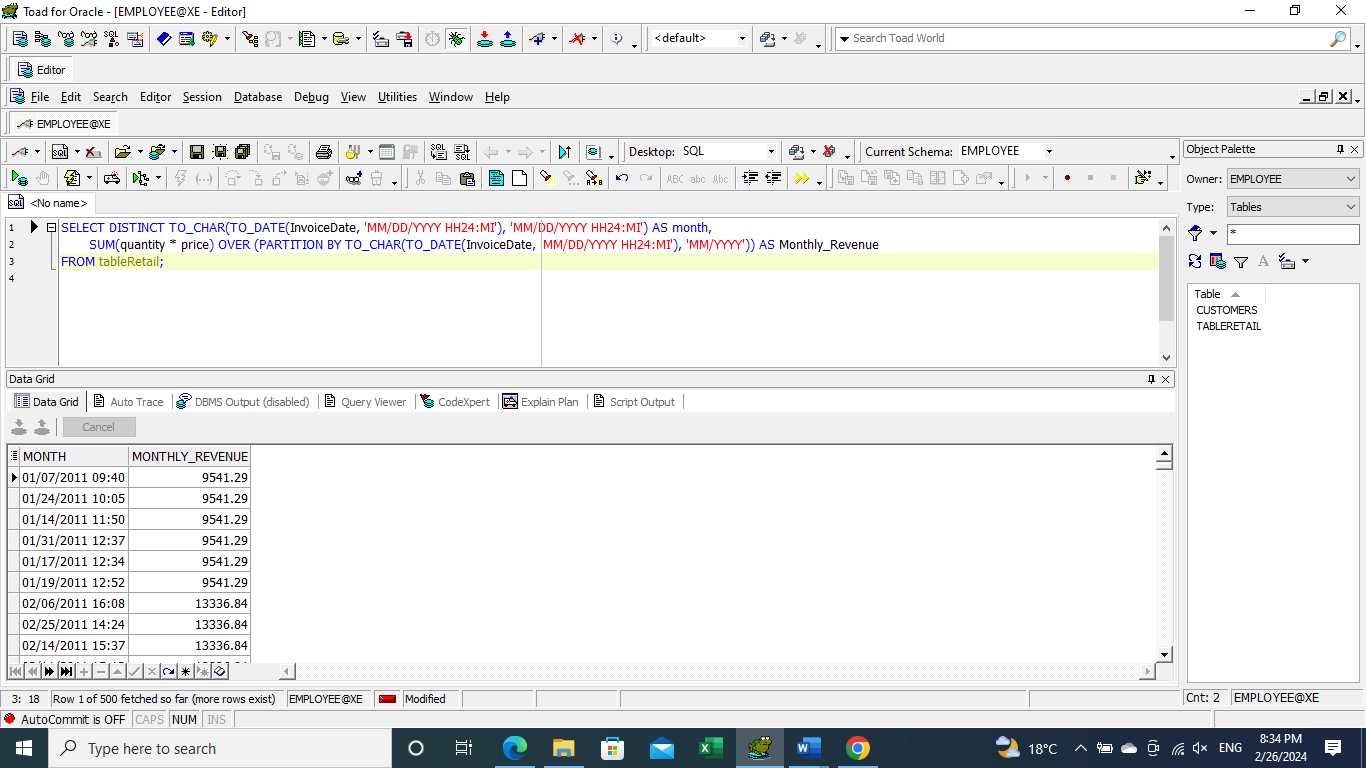


1. **Monthly Revenue**: This query calculates the total revenue for each month, providing insights into revenue trends over time.

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

SUM(quantity \* price) OVER (PARTITION BY TO\_CHAR(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'), 'MM/YYYY')) AS Monthly\_Revenue

FROM tableRetail;

\

1. **Customer Purchase Frequency**: This query calculates the frequency of purchases for each customer and ranks them based on purchase frequency. It helps identify loyal customers.

with customer\_purchase as (

select customer\_id , count(invoice) as purchaces from tableretail

group by customer\_id ),

ranked\_purchase as (

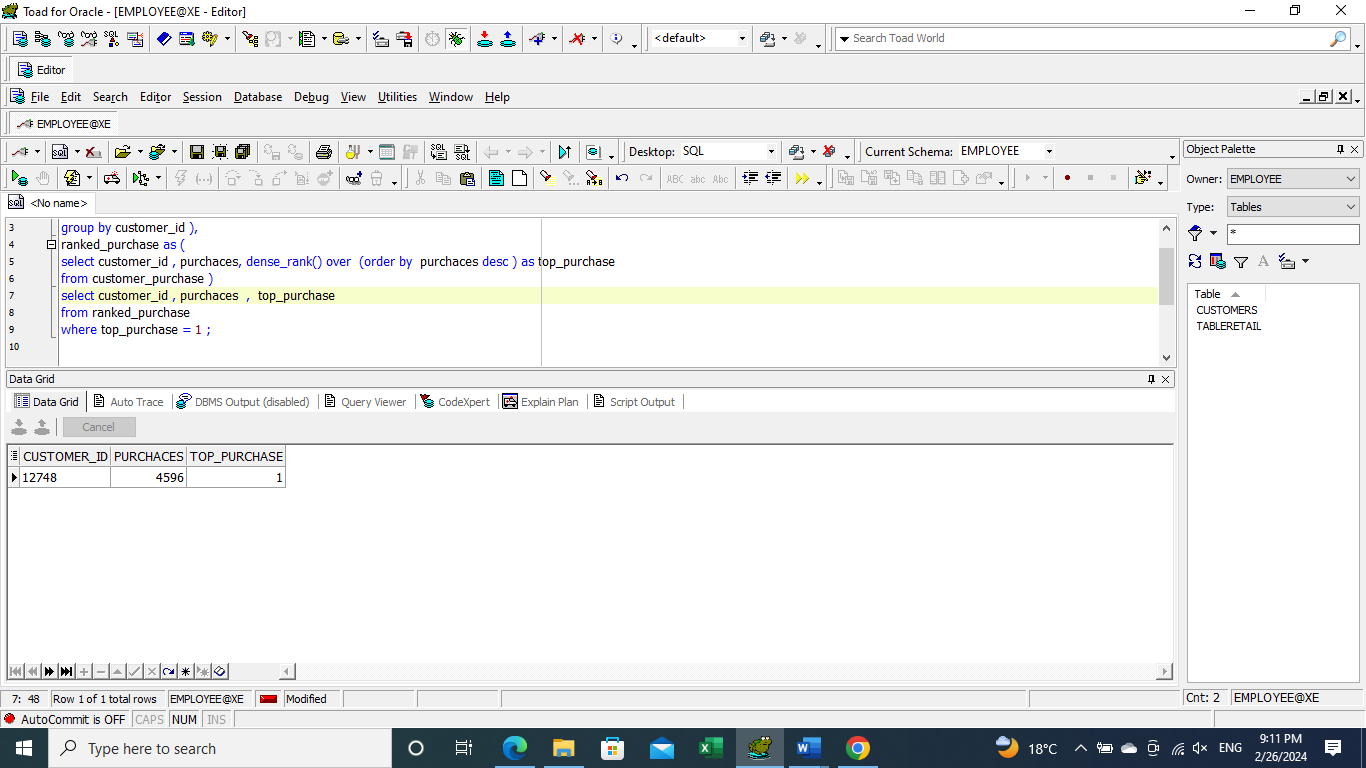
select customer\_id , purchaces, dense\_rank() over (order by purchaces desc ) as top\_purchase

from customer\_purchase )

select customer\_id , purchaces , top\_purchase

from ranked\_purchase

where top\_purchase = 1 ;



1. **Most Ordered Stock Items**: This query calculates the total quantity of each stock item ordered and ranks them based on total quantity. It helps identify popular products.

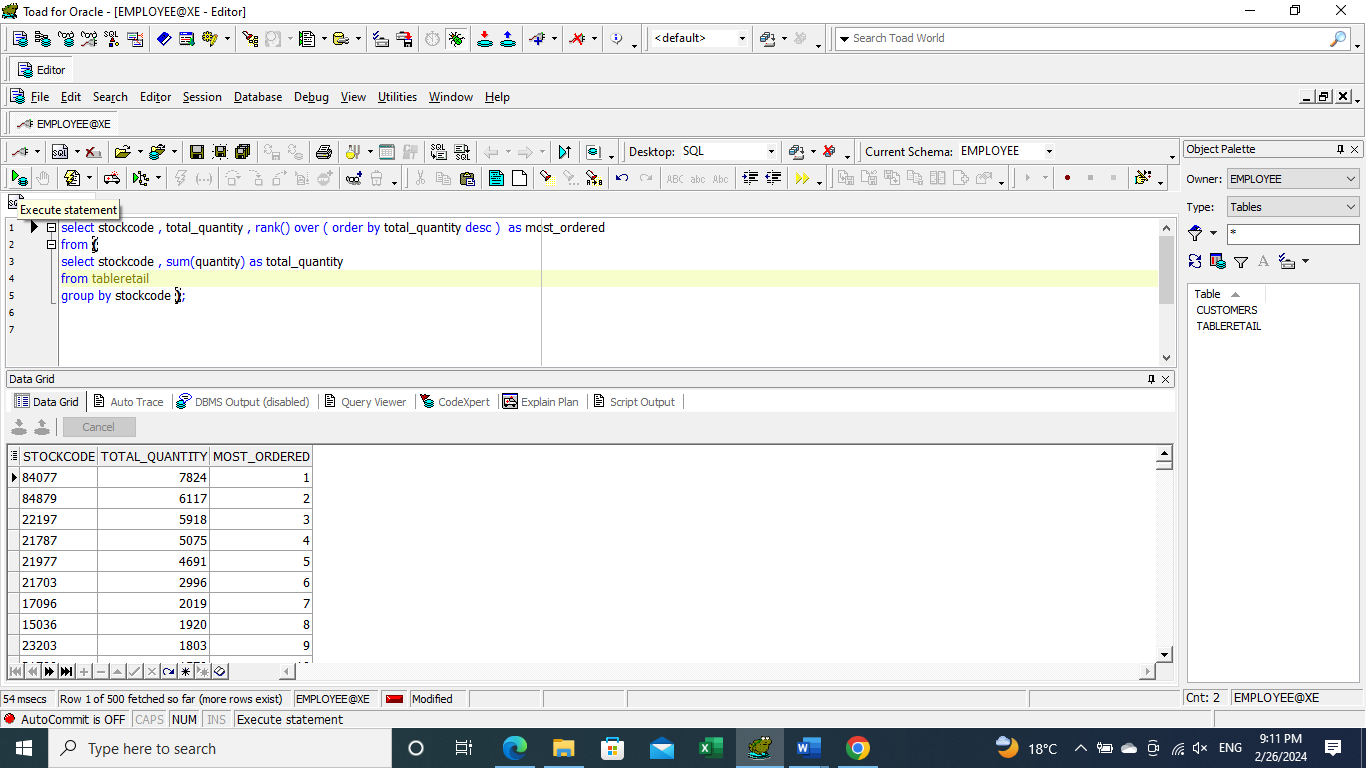
select stockcode , total\_quantity , rank() over ( order by total\_quantity desc ) as most\_ordered

from (

select stockcode , sum(quantity) as total\_quantity

from tableretail

group by stockcode );



**Q2**

This query is used to calculate the RFM (Recency, Frequency, Monetary) scores for each customer based on their purchasing behavior. RFM analysis is a technique used in marketing to segment customers based on their transaction history. Here's a breakdown of the query and its purpose:

1. **RFMData CTE**: This part of the query calculates the recency, frequency, and monetary values for each customer. Recency is calculated as the difference between the most recent invoice date and the maximum invoice date for each customer. Frequency is the count of invoices for each customer, and monetary is the sum of the quantity multiplied by the price for each invoice.
2. **rfm\_score CTE**: This part of the query calculates the RFM scores for each customer by assigning them to quintiles based on their recency, frequency, and monetary values. This helps in ranking customers based on these metrics.
3. **fm\_avg CTE**: This part of the query calculates the average of the frequency and monetary scores to get the FM score, which is used for segmentation.
4. **seg\_cust CTE**: This part of the query assigns a segment to each customer based on their R\_score and FM\_score combination. There are several segments defined, such as "Champions," "Loyal Customers," "Potential Loyalists," etc., each representing a different level of customer engagement and loyalty.
5. **Final SELECT**: This part of the query selects the customer ID, recency, frequency, monetary values, R\_score, FM\_score, and the customer segment for each customer. This information can be used for targeted marketing campaigns, customer retention strategies, and personalized customer engagement based on their segment.

WITH RFMData AS (

SELECT

CUSTOMER\_ID,

(SELECT MAX(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) FROM tableretail) - MAX(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) AS Recency,

COUNT(INVOICE) AS Frequency,

SUM(QUANTITY \* PRICE)AS Monetary

FROM

tableretail

GROUP BY

CUSTOMER\_ID

),

rfm\_score AS (

SELECT

customer\_id,

recency,

Frequency,

Monetary,

NTILE(5) OVER (ORDER BY recency desc ) AS r\_score,

NTILE(5) OVER (ORDER BY frequency)AS f\_score,

NTILE(5) OVER (ORDER BY monetary) AS m\_score

FROM

RFMData

),

fm\_avg AS (

SELECT

customer\_id,

recency,

Frequency,

Monetary,

r\_score,

f\_score,

m\_score,

round( (F\_score + M\_score) / 2) AS FM\_score

FROM

rfm\_score

),

seg\_cust AS (

SELECT

customer\_id,

recency,

Frequency,

Monetary,

r\_score,

f\_score,

m\_score,

FM\_score,

CASE

WHEN R\_score = 5 AND FM\_score = 5 THEN 'Champions'

WHEN R\_score = 5 AND FM\_score = 4 THEN 'Champions'

WHEN R\_score = 4 AND FM\_score = 5 THEN 'Champions'

WHEN R\_score = 5 AND FM\_score = 2 THEN 'Potential Loyalists'

WHEN R\_score = 4 AND FM\_score = 2 THEN 'Potential Loyalists'

WHEN R\_score = 3 AND FM\_score = 3 THEN 'Potential Loyalists'

WHEN R\_score = 4 AND FM\_score = 3 THEN 'Potential Loyalists'

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 = 5 THEN 'Loyal Customers'

WHEN 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 THEN 'Promising'

WHEN R\_score = 3 AND FM\_score = 1 THEN 'Promising'

WHEN R\_score = 3 AND FM\_score = 2 THEN 'Customers Needing Attention'

WHEN R\_score = 2 AND FM\_score = 3 THEN 'Customers Needing Attention'

WHEN R\_score = 2 AND FM\_score = 2 THEN 'Customers Needing Attention'

WHEN R\_score = 2 AND FM\_score = 5 THEN 'At Risk'

WHEN R\_score = 2 AND FM\_score = 4 THEN 'At Risk'

WHEN R\_score = 1 AND FM\_score = 3 THEN 'At Risk'

WHEN R\_score = 1 AND FM\_score = 5 THEN 'Cant Lose Them'

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

fm\_avg

)

SELECT

CUSTOMER\_ID,

Recency,

Frequency,

Monetary,

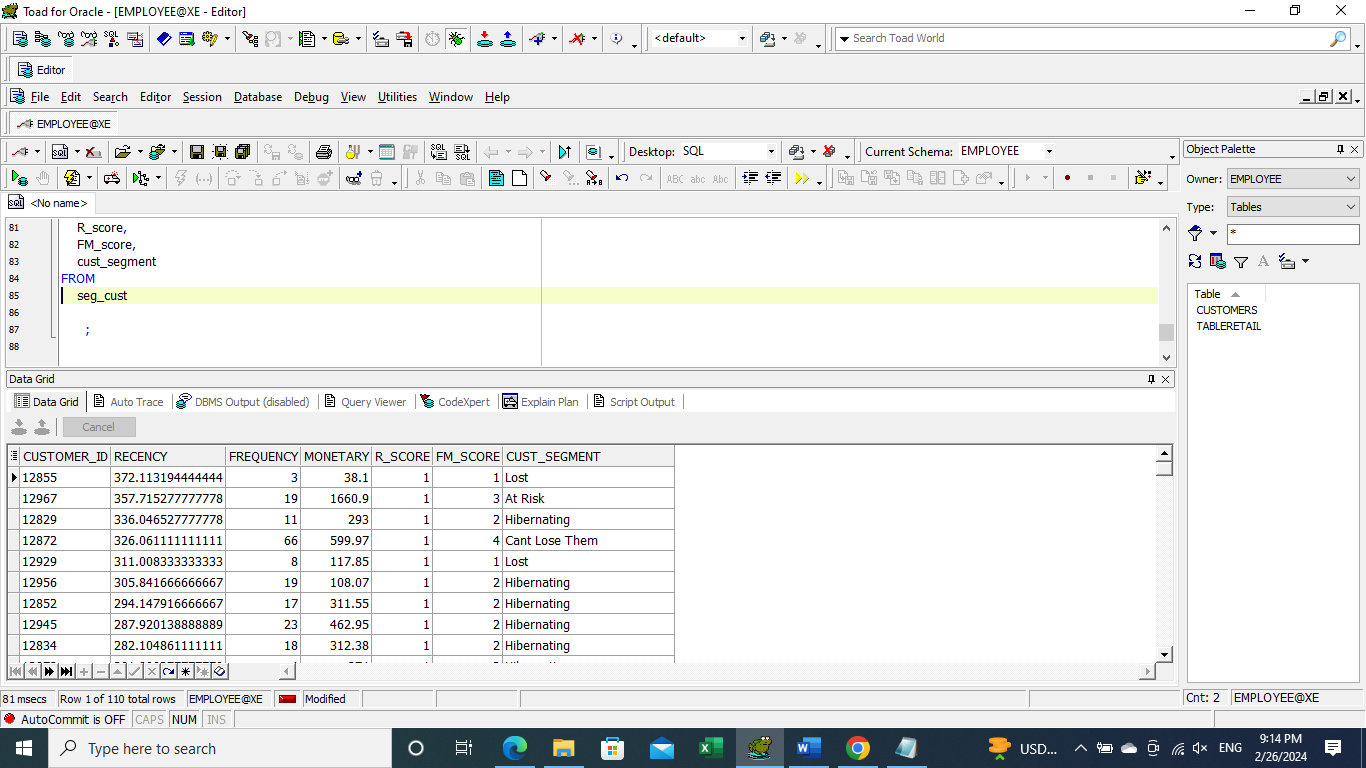
R\_score,

FM\_score,

cust\_segment

FROM

seg\_cust ;



**Q3.A**

This query calculates the maximum number of consecutive days a customer has made purchases. It first assigns a row number to each purchase date for each customer, then calculates the gap in days between each purchase date and its corresponding row number. Finally, it counts the number of consecutive days for each customer and determines the maximum consecutive days for each customer. This information can be useful for understanding customer purchase behavior, identifying regular or frequent purchasers, and tailoring marketing strategies to increase engagement.

with cust\_row as (

select cust\_id , calendar\_dt , row\_number() over (partition by cust\_id order by calendar\_dt ) as rn

from customers ),

cons\_days as (

select cust\_id , calendar\_dt , rn,

calendar\_dt - row\_number() over (partition by cust\_id order by calendar\_dt ) as gap\_days

from cust\_row ) ,

consecutive as (

select cust\_id , gap\_days , count (\*) as consecutive\_days

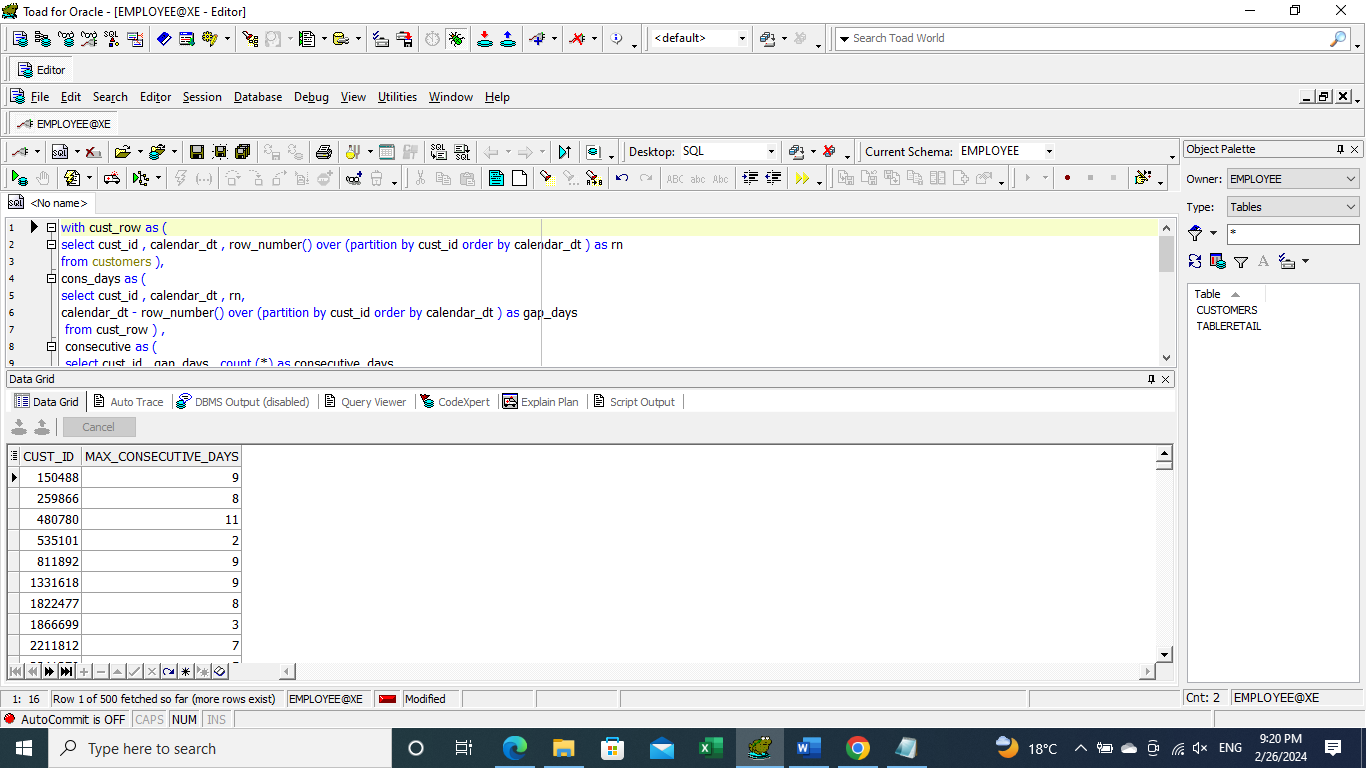
from cons\_days

group by cust\_id , gap\_days)

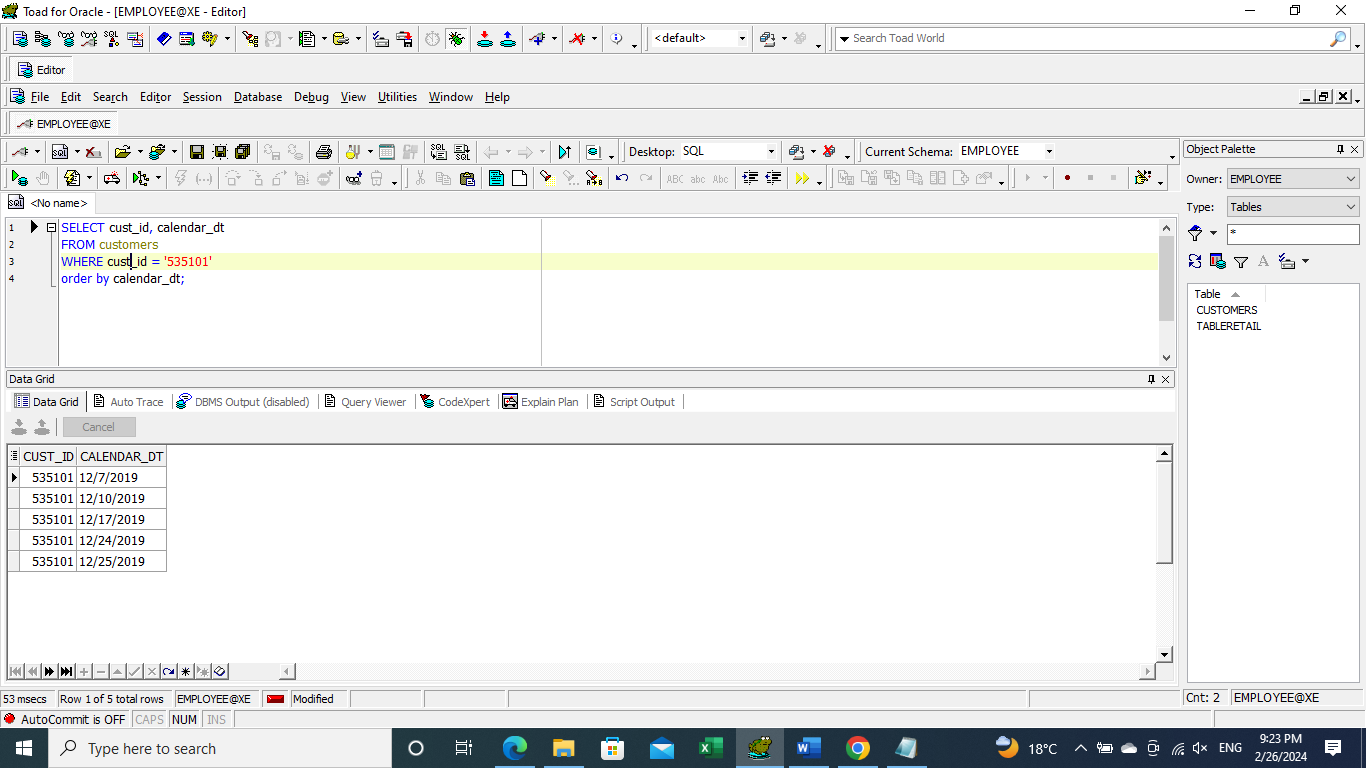
select cust\_id , max(consecutive\_days) as max\_consecutive\_days

from consecutive

group by cust\_id ;



**EXAMPLE:**



**Q3.B**

This query calculates the average number of days it takes for customers to reach a spending threshold of 250 units of currency. It first calculates the total amount spent by each customer on each day and calculates the cumulative total of their spending. Then, it identifies the first day each customer's spending total exceeds or equals 250 units. Finally, it calculates the average number of days it takes across all customers to reach this threshold. This information can be valuable for understanding customer purchasing behavior and can help in designing targeted marketing campaigns or loyalty programs.

WITH cust\_amt AS (

SELECT

cust\_id,

calendar\_dt,

SUM(amt\_le) OVER (PARTITION BY cust\_id ORDER BY calendar\_dt) AS total,

calendar\_dt - FIRST\_VALUE(calendar\_dt) OVER (PARTITION BY cust\_id ORDER BY calendar\_dt)

AS total\_days

FROM

customers

),

amt\_threshold AS (

SELECT

cust\_id,

calendar\_dt,

total , total\_days

FROM

cust\_amt

WHERE

total >= 250

) ,

threshold\_days as (

SELECT

min(total\_days) as total\_days\_threshold , cust\_id

FROM

amt\_threshold

group by cust\_id )

select

round(avg (total\_days\_threshold) ) as threshold

from threshold\_days ;

