**Question 1**

Query 1 - Total income per month(13 months ) and comparing with pervious month and then calculating the amount of change in income and the percent and getting the rank of each month.

SELECT

*-- Select the month and year of the invoice date and format it to be displayed as "MON YYYY"*

to\_char(to\_date( INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ) AS Month ,

*-- Calculate the total income per month by multiplying the quantity and price of each item in the invoice and then summing them up*

sum( QUANTITY \* PRICE) AS Total\_Income ,

*-- Use the lag function to get the total income from the previous month*

lag(sum(QUANTITY \* PRICE)) OVER(ORDER BY TO\_DATE(to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC) AS Prev\_Total\_Income,

*-- Calculate the total change in income from the previous month by subtracting the previous month's total income from the current month's total income*

(sum( QUANTITY \* PRICE) -lag( sum( QUANTITY \* PRICE) ) OVER(ORDER BY TO\_DATE(to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC)) AS Total\_Change,

*-- Calculate the percentage change in income from the previous month by dividing the total change in income by the previous month's total income and multiplying by 100*

round((sum( QUANTITY \* PRICE) -lag( sum( QUANTITY \* PRICE) ) OVER(ORDER BY TO\_DATE(to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC))/lag( sum( QUANTITY \* PRICE) ) OVER(ORDER BY TO\_DATE(to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC),4) \*100 AS percent\_CHANGE ,

*-- Rank the months by total income in descending order*

RANK() OVER(ORDER BY sum( QUANTITY \* PRICE) DESC) AS MONTH\_RANK

FROM

tableretail

GROUP BY

*-- Group the data by month and year of the invoice date*

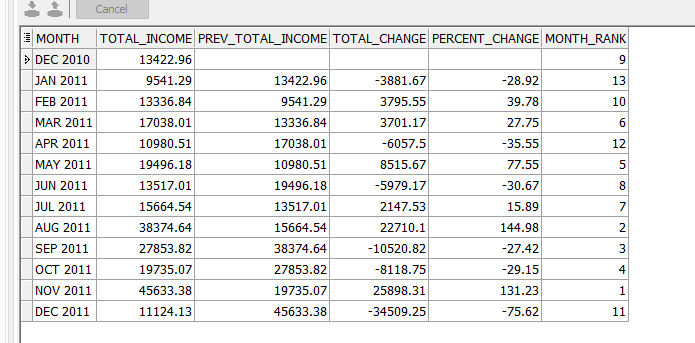
to\_char(to\_date( INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' )

ORDER BY

*-- Order the results by the invoice date in ascending order*

TO\_DATE(to\_char(to\_date( INVOICEDATE,'MM/DD/YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC ;

YYYY HH24:MI'),'MON YYYY' ),'MON YYYY') ASC ;



2- query 2

categorizing the working hours into 5 categories ('Platinum Hour', 'Diamond Hour', 'Gold Hour', 'Silver Hour', 'Bronze Hour')

WITH HOUR\_DETAILS1 AS (

*-- Selecting the invoice, the date of the invoice and calculating the total income for each invoice.*

select invoice, to\_date(INVOICEDATE, 'MM/DD/YYYY HH24:MI') as date1, sum(QUANTITY \* PRICE) as Total\_Income1

from tableretail

*-- Grouping by invoice and date to get total income per invoice and date.*

group by invoice, to\_date(INVOICEDATE, 'MM/DD/YYYY HH24:MI')

*-- Ordering by invoice number.*

ORDER BY invoice

)

, HOUR\_DETAILS2 as (

*-- Selecting the hour in AM/PM format and calculating total income, number of invoices per hour.*

select to\_char(date1, 'HH AM') AS HOUR, sum(Total\_Income1) as Total\_Income, count(1) number\_of\_invoices,

*-- Dividing the data into three equal parts based on the count of invoices and total income per hour respectively.*

NTILE(3) OVER (ORDER BY count(1) desc) AS invoices\_Rank, NTILE(3) OVER (ORDER BY sum(Total\_Income1) desc) AS Income\_Rank

FROM HOUR\_DETAILS1

*-- Grouping by hour in AM/PM format.*

group by to\_char(date1, 'HH AM')

)

*-- Selecting the hour, total income, income rank, number of invoices, invoice rank and hour category.*

SELECT HOUR, Total\_Income, Income\_Rank, number\_of\_invoices, invoices\_Rank,

CASE

*-- Defining the hour category based on income rank and invoice rank.*

WHEN (Income\_Rank, invoices\_Rank) in ((1,1)) THEN 'Platinum Hour'

WHEN (Income\_Rank, invoices\_Rank) in ((1,2),(2,1)) THEN 'Diamond Hour'

WHEN (Income\_Rank, invoices\_Rank) in ((2,2)) THEN 'Gold Hour'

WHEN (Income\_Rank, invoices\_Rank) in ((2,3),(3,2)) THEN 'Silver Hour'

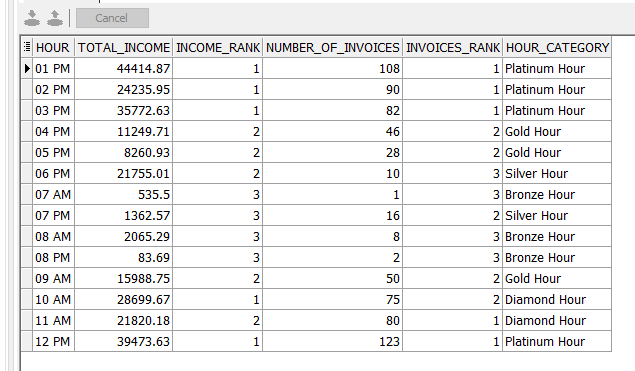
WHEN (Income\_Rank, invoices\_Rank) in ((3,3)) THEN 'Bronze Hour'

END AS Hour\_Category

FROM HOUR\_DETAILS2

*-- Ordering by hour in AM/PM format.*

ORDER BY Hour;



3 – grouping customers into two groups and getting the income and number of custmoers and invoices per group

*-- Create a common table expression to calculate the total sales and number of invoices per customer*

with customer\_total as (

SELECT

customer\_id, *-- select the customer\_id column*

count(1) as number\_of\_invoices, *-- count the number of invoices for each customer*

SUM(QUANTITY\*PRICE) AS Sales, *-- calculate the total sales for each customer by multiplying the quantity and price*

PERCENT\_RANK() OVER (ORDER BY SUM(QUANTITY\*PRICE) DESC) AS SalesRank *-- calculate the percent rank of each customer based on their sales*

FROM TABLERETAIL

GROUP BY customer\_id *-- group the results by customer\_id*

),

*-- Create another common table expression to categorize customers based on their sales rank*

category\_total as (

SELECT

customer\_id, *-- select the customer\_id column*

Sales, *-- select the Sales column*

number\_of\_invoices, *-- select the number\_of\_invoices column*

CASE *-- use a CASE statement to categorize customers based on their sales rank*

WHEN SalesRank <= .2 THEN 'First 20% Group' *-- if the sales rank is less than or equal to 0.2, assign the customer to the 'First 20% Group'*

WHEN SalesRank > .2 THEN 'Last 80% Group' *-- if the sales rank is greater than 0.2, assign the customer to the 'Last 80% Group'*

END AS Customer\_Category

FROM customer\_total *-- use the previously created 'customer\_total' common table expression*

)

*-- Query the results and group by customer category*

SELECT

Customer\_Category, *-- select the Customer\_Category column*

sum(sales) as total\_sales, *-- calculate the total sales for each customer category*

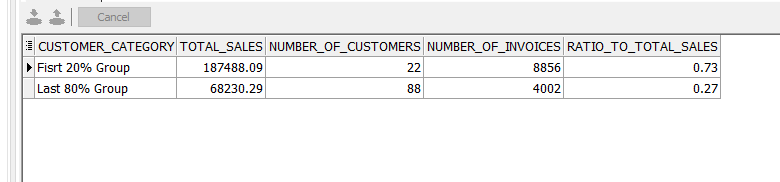
count(customer\_id) as number\_of\_customers, *-- count the number of customers in each category*

sum(number\_of\_invoices) as number\_of\_invoices, *-- calculate the total number of invoices for each customer category*

round(sum(sales)/sum(sum(sales)) over(), 2) as Ratio\_to\_Total\_Sales *-- calculate the ratio of total sales for each customer category*

FROM category\_total

GROUP BY Customer\_Category; *-- group the results by customer category*



4-

query 4

*-- select the customer ID column and calculate metrics for each customer*

SELECT

Customer\_ID,

COUNT(DISTINCT Invoice) AS NUMBER\_OF\_orders, *-- count the number of distinct invoices for each customer*

ROUND(MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) - MIN(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI'))) AS CONSECUTIVE\_PERIOD, *-- calculate the difference between the earliest and latest invoice dates for each customer, rounding to the nearest whole number*

SUM(Quantity \* Price) AS total\_sales, *-- calculate the sum of the product of quantity and price for each customer*

ROUND(CORR(COUNT(DISTINCT Invoice), SUM(Quantity \* Price)) OVER (), 2) AS CORR\_SALES\_INVOICES, *-- calculate the correlation between the number of distinct invoices and the sum of the product of quantity and price for each customer, rounding to two decimal places*

ROUND(CORR(SUM(Quantity \* Price), (MAX(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')) - MIN(TO\_DATE(InvoiceDate, 'MM/DD/YYYY HH24:MI')))) OVER (), 2) AS CORR\_SALES\_days *-- calculate the correlation between the sum of the product of quantity and price and the difference between the earliest and latest invoice dates for each customer, rounding to two decimal places*

FROM

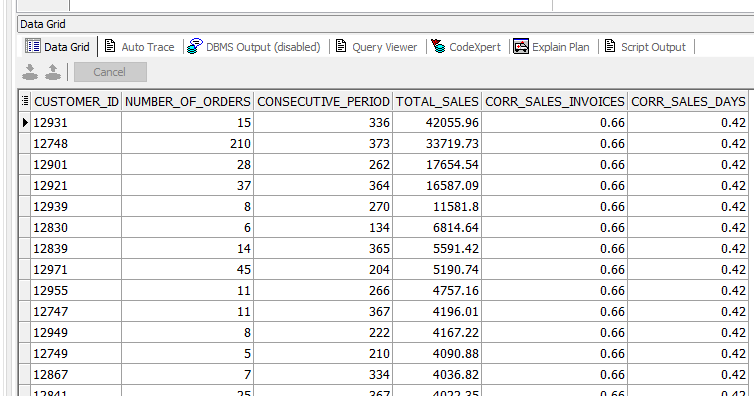
tableRetail *-- select data from the "tableRetail" table*

GROUP BY

Customer\_ID *-- group the data by customer ID*

ORDER BY

total\_sales DESC *-- sort the data by total sales in descending order*



5-

getting the totalsales and nmber of invoices for each stockcode and getting the correlation between them

*-- Select the stock code, count of invoices, total quantity, and total sales for each stock code*

SELECT stockcode, COUNT(1) AS number\_of\_invoices, sum(quantity) as total\_quantity, SUM(QUANTITY\*PRICE) AS total\_Sales,

*-- Calculate the sales rank of each stock code based on their total sales*

RANK() OVER (ORDER BY SUM(QUANTITY\*PRICE) DESC) AS SalesRank,

*-- Calculate the invoice count rank of each stock code based on the number of invoices*

RANK() OVER (ORDER BY COUNT(1) DESC) AS invoice\_count,

*-- Calculate the correlation between the number of invoices and total sales*

ROUND(CORR(COUNT(1), SUM(QUANTITY\*PRICE)) OVER (), 2) AS invoices\_correlation,

*-- Calculate the correlation between the total quantity and total sales*

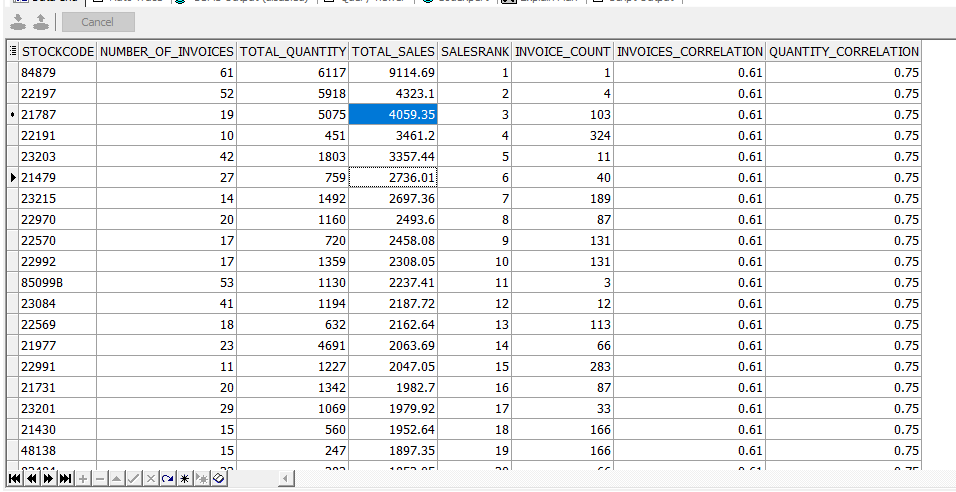
ROUND(CORR(sum(quantity), SUM(QUANTITY\*PRICE)) OVER (), 2) AS quantity\_correlation

*-- Group the results by stock code and order by total sales in descending order*

FROM TABLERETAIL

GROUP BY stockcode

ORDER BY total\_Sales DESC;



**Q2**

*-- Compute the RFM values for each customer*

WITH CUSTOMER\_details AS (

SELECT

CUSTOMER\_ID,

*-- Compute the recency value as the number of days since the last purchase*

ROUND(

MAX(MAX(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'))) OVER() - MAX(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'))

) AS recency,

*-- Compute the frequency value as the number of purchases made by the customer*

COUNT(1) AS frequency,

*-- Compute the monetary value as the total amount spent by the customer*

SUM(quantity \* price) AS Monetary

FROM

tableretail

GROUP BY

CUSTOMER\_ID

),

*-- Compute the RFM scores for each customer*

CUSTOMER\_ID\_SCORE AS (

SELECT

CUSTOMER\_ID,

recency,

frequency,

Monetary,

*-- Compute the recency score based on the recency value*

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

*-- Compute the frequency/monetary score based on the average of frequency and monetary values*

NTILE(5) OVER (ORDER BY (frequency + Monetary) / 2) AS F\_M\_score

FROM

CUSTOMER\_details

)

*-- Categorize the customers based on their RFM scores*

SELECT

CUSTOMER\_ID,

recency,

frequency,

Monetary,

recency\_score,

F\_M\_score,

*-- Categorize the customer based on their recency and frequency/monetary scores*

CASE

WHEN recency\_score >= 4 AND F\_M\_score > 4 THEN 'CHAMPIONS'

WHEN (recency\_score, F\_M\_score) IN ((5, 2), (4, 2), (3, 3), (4, 3)) THEN 'Potential Loyalists'

WHEN (recency\_score, F\_M\_score) IN ((5, 3), (4, 4), (3, 5), (3, 4)) THEN 'Loyal Customers'

WHEN (recency\_score, F\_M\_score) IN ((5, 1)) THEN 'Recent Customers'

WHEN (recency\_score, F\_M\_score) IN ((4, 1), (3, 1)) THEN 'Promising'

WHEN (recency\_score, F\_M\_score) IN ((3, 2), (2, 3), (2, 2)) THEN 'Customers Needing Attention'

WHEN (recency\_score, F\_M\_score) IN ((2, 5), (2, 4), (1, 3)) THEN 'At Risk'

WHEN (recency\_score, F\_M\_score) IN ((1, 5), (1, 4)) THEN 'Cant Lose them'

WHEN (recency\_score, F\_M\_score) IN ((1, 2)) THEN 'Hibernating'

WHEN (recency\_score, F\_M\_score) IN ((1, 1)) THEN 'Lost'

END AS Hour\_Category

FROM

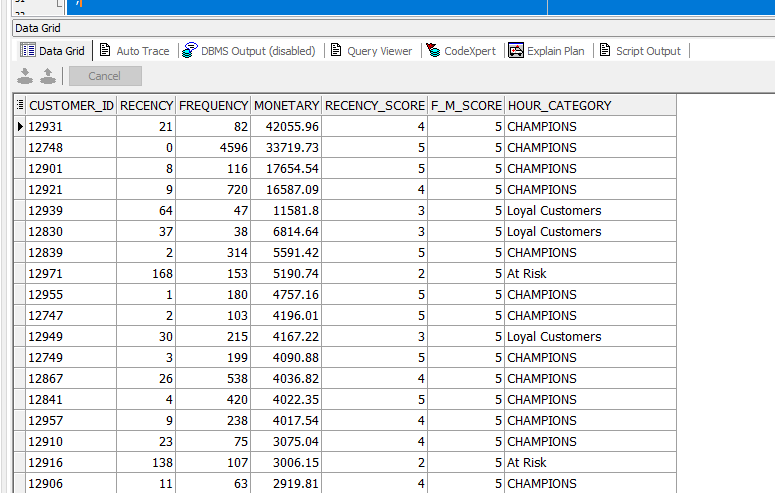
CUSTOMER\_ID\_SCORE

*-- Order the results by the monetary value in descending order*

ORDER BY

Monetary DESC;

;



**Q3**

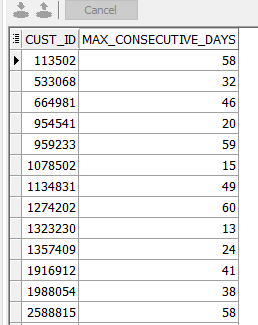
1

select DISTINCT(cust\_id) ,

(MAX( to\_date( INVOICEDATE,'YYYY-MM-DD')) over(PARTITION BY cust\_id ) -MIN( to\_date( INVOICEDATE,'YYYY-MM-DD')) over(PARTITION BY cust\_id ) ) AS MAX\_CONSECUTIVE\_DAYS

FROM Transactions

;



2

WITH CUST\_DETAILS AS(

SELECT

CUST\_ID,

to\_date(INVOICEDATE, 'YYYY-MM-DD') INVOICEDATE, *-- converting the format of the INVOICEDATE col*

SUM(PRICE) OVER ( PARTITION BY CUST\_ID ORDER BY to\_date(INVOICEDATE, 'YYYY-MM-DD') ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW ) AS TOTAL\_TO\_DAY, *-- calculating the tolal money earned for this customer from his his first order till the current order*

to\_date(INVOICEDATE, 'YYYY-MM-DD') - MIN(to\_date(INVOICEDATE, 'YYYY-MM-DD')) OVER (PARTITION BY CUST\_ID) AS days\_between, *-- calculating number of days between the first order for this custmer till the current order*

COUNT(1) OVER (PARTITION BY CUST\_ID ORDER BY to\_date(INVOICEDATE, 'YYYY-MM-DD') ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW ) AS count\_TO\_DAY *-- calculating number of orders for this customer from his his first order till the current order*

FROM TRANSACTIONS

ORDER BY CUST\_ID

)

,cust3 AS (

*-- categorize customers by sales, based on whether their total sales is greater than or equal to 250*

SELECT

CUST\_ID,

INVOICEDATE,

TOTAL\_TO\_DAY,

count\_TO\_DAY,

days\_between,

CASE

WHEN TOTAL\_TO\_DAY >= 250 THEN days\_between *-- if so, then return the days between the first invoice and the one where they reached 250*

WHEN TOTAL\_TO\_DAY < 250 THEN -1 *-- otherwise, return -1*

END AS sales\_Category

FROM CUST\_DETAILS

)

,cust4 AS (

*-- for each customer, calculate the number of invoices and the number of days it took to reach 250 in sales*

*-- only include customers whose total sales were at least 250*

SELECT

cust\_id,

MIN(count\_TO\_DAY) number\_of\_invoices,

MIN(sales\_Category) AS days\_to\_reach\_250

FROM cust3

WHERE sales\_Category >= 0

GROUP BY cust\_id

)

*-- finally, calculate the average number of invoices and the average number of days to reach 250 in sales*

*-- for all customers whose total sales were at least 250*

SELECT

ROUND(AVG(number\_of\_invoices)) AS average\_invoices,

ROUND(AVG(days\_to\_reach\_250)) AS average\_days

FROM cust4;

