Q1:

1- What are the top-selling products in terms of quantity sold?

```
WITH quantity_sum AS (
SELECT  stockcode,  SUM(quantity) OVER(PARTITION BY stockcode  ) AS sum_quantity
FROM tableRetail)
SELECT DISTINCT  stockcode, sum_quantity
FROM quantity_sum
WHERE sum_quantity = (SELECT MAX(sum_quantity) FROM quantity_sum);
```

➔ The business insights :By identifying the top-selling products based on quantity sold, the store owner gets insights into consumer preferences and demand patterns. This information useful for management decisions, highlights popular product categories, and a good guide to ensure stock availability for high-demand items.

2- What are the top-selling products in terms of quantity sold for each month?

```
WITH quantity_sum AS (
   SELECT
      TO_CHAR(TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI'),'MM') AS mon,
      stockcode,
      SUM(quantity) OVER(PARTITION BY TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI'),
stockcode) AS sum_quantity
   FROM
      tableRetail
)
SELECT   mon,   stockcode,  sum_quantity AS max_sum_quantity_per_month
FROM  quantity_sum
WHERE   (mon, sum_quantity) IN (
 SELECT  mon, MAX(sum_quantity) AS max_sum_quantity_per_month
 FROM  quantity_sum
GROUP BY  mon
   )
ORDER BY mon;
```

➔ The business insights :Through this, we can identify the trends in each month by tracking the best-selling products each month . the owner can identify seasonal fluctuations, capitalize on emerging trends, and tailor marketing campaigns to promote relevant products, maximizing sales opportunities throughout the year.

3- What is the average transaction size for each hour of the day?

```
 SELECT DISTINCT
      TO_CHAR(TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI'),'HH24') AS Hours,
     ROUND(AVG(quantity*price) OVER(PARTITION BY   TO_CHAR(TO_DATE(invoicedate
,'MM/DD/YYYY HH24:MI'),'HH24')),2)||'$' AS avg_sales
   FROM
      tableRetail
   ORDER BY Hours;
```

➔ The business insights :Understanding transaction sizes across different hours of the day provides valuable insights into consumer behavior and purchasing patterns throughout the day. By identifying peak hours of transaction activity and average transaction sizes, the store manager can optimize staffing levels, schedule promotions during high-traffic periods, and tailor product offerings to meet customer demand

4- What are the total sales per month ?

```
SELECT DISTINCT
       TO_CHAR(TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI'),'MM') AS mon,
       SUM(quantity*price) OVER(PARTITION BY  TO_CHAR(TO_DATE(invoicedate ,'MM/DD/YYYY
HH24:MI'),'MM'))||'$' AS TOTAL_seles_per_month
   FROM
      tableRetail
      ORDER BY MON;
```

➔ The business insights :Monitoring total sales per month helps businesses evaluate the effectiveness of their sales efforts and business strategies. Comparing sales performance across different months allows businesses to assess the impact of marketing initiatives, product launches, pricing changes, and other factors on overall revenue generation.

5- What is the average price per unit for each product?

```
SELECT DISTINCT stockcode , ROUND(AVG(price) OVER(PARTITION BY stockcode),2)||'$' AS
avg_price_for_product
     FROM
       tableRetail;
```

➔ The business insights :Comparing the average price per unit of products with those of competitors can provide valuable competitive insights. Businesses can assess whether their prices are competitive within the market and adjust pricing strategies to maintain a competitive edge.

6- Who are the top 10 customers?

```
WITH customer_sales AS (
   SELECT customer_id, SUM(price * quantity) ||'$' AS total_sales
   FROM  tableRetail
   GROUP BY customer_id
)
SELECT
   customer_id, total_sales
FROM (
   SELECT customer_id,  total_sales, DENSE_RANK() OVER (ORDER BY total_sales DESC) AS
sales_rank
   FROM customer_sales
)
WHERE sales_rank <= 10;
```

➔ The business insights :customers typically have a higher level of loyalty and satisfaction with the products or services offered by the business. By understanding their needs, preferences, and purchasing behavior, businesses can create and implement strategies to keep these customers engaged and satisfied.

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

As there are many groups for each of the R, F, and M features, there are also many potential permutations, this number is too much to manage in terms of marketing strategies. For this, we would decrease the permutations by getting the average scores of the frequency and monetary (as both of them are indicative to purchase volume anyway

```
WITH cus_data AS(
SELECT DISTINCT  customer_id , round ( ( SELECT MAX( TO_DATE (invoicedate,'MM/DD/YYYY
HH24:MI') ) FROM tableretail ) - FIRST_VALUE(TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI') IGNORE
NULLS)
OVER(PARTITION BY customer_id ORDER BY TO_DATE(invoicedate ,'MM/DD/YYYY HH24:MI') DESC )
)AS recency ,
COUNT(DISTINCT invoicedate ) OVER(PARTITION BY  customer_id ) AS frequency,
SUM(price * quantity) OVER(PARTITION BY  customer_id ) AS monetary
FROM tableRetail
),
cus_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 cus_data
   )
SELECT customer_id, recency,  frequency,  monetary, r_score,ROUND( (f_score+m_score)/2) AS
fm_score , CASE
                                                    WHEN r_score =5 and
ROUND( (f_score+m_score)/2)  =5 THEN 'Champions'
                                                    WHEN r_score =4 and
ROUND( (f_score+m_score)/2)  =5 THEN 'Champions'
                                                    WHEN r_score =5 and
ROUND( (f_score+m_score)/2)  =4 THEN 'Champions'
                                                    WHEN r_score =5 and
ROUND( (f_score+m_score)/2)  =2 THEN 'Potential Loyalists'
                                                    WHEN r_score =4 and
ROUND( (f_score+m_score)/2)  =2 THEN 'Potential Loyalists'
                                                    WHEN r_score =4 and
ROUND( (f_score+m_score)/2)  =3 THEN 'Potential Loyalists'
```

```
ROUND( (f_score+m_score)/2)  =3 THEN 'Potential Loyalists'
ROUND( (f_score+m_score)/2)  =3 THEN 'Loyal Customers'
ROUND( (f_score+m_score)/2)  =4 THEN 'Loyal Customers'
ROUND( (f_score+m_score)/2)  =5 THEN 'Loyal Customers'
ROUND( (f_score+m_score)/2)  =4 THEN 'Loyal Customers'
ROUND( (f_score+m_score)/2)  =1 THEN 'Recent Customers'
ROUND( (f_score+m_score)/2)  =1 THEN 'Promising'
ROUND( (f_score+m_score)/2)  =1 THEN 'Promising'
ROUND( (f_score+m_score)/2)  =2 THEN 'Customers Needing Attention'
ROUND( (f_score+m_score)/2)  =3 THEN 'Customers Needing Attention'
ROUND( (f_score+m_score)/2)  =2 THEN 'Customers Needing Attention'
ROUND( (f_score+m_score)/2)  =5 THEN 'At Risk'
ROUND( (f_score+m_score)/2)  =4 THEN 'At Risk'
ROUND( (f_score+m_score)/2)  =3 THEN 'At Risk'
ROUND( (f_score+m_score)/2)  =5 THEN 'Cant Lose Them'
ROUND( (f_score+m_score)/2)  =4 THEN 'Cant Lose Them'
ROUND( (f_score+m_score)/2)  =2 THEN 'Hibernating'
ROUND( (f_score+m_score)/2)  =1 THEN 'Hibernating'
ROUND( (f_score+m_score)/2)  =1 THEN 'Lost'

FROM cus_score
;
```

```
WHEN r_score =3 and
WHEN r_score =5 and
WHEN r_score =4 and
WHEN r_score =3 and
WHEN r_score =3 and
WHEN r_score =5 and
WHEN r_score =4 and
WHEN r_score =3 and
WHEN r_score =3 and
WHEN r_score =2 and
WHEN r_score =2 and
WHEN r_score =2 and
WHEN r_score =2 and
WHEN r_score =1 and
WHEN r_score =1 and
WHEN r_score =1 and
WHEN r_score =1 and
WHEN r_score =2 and
WHEN r_score =1 and
END AS group_name
```

Q3:

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

```
CREATE TABLE transactions ( Cust_Id number ,
                            Calendar_Dt date ,
                            Amt_LE  float );
                            ----------------------

WITH gap_tab AS (
  SELECT
```

```
        Cust_Id,
        Calendar_Dt,
        LAG(Calendar_Dt, 1) OVER(PARTITION BY Cust_Id ORDER BY Calendar_Dt) AS pre_date,
        Calendar_Dt - LAG(Calendar_Dt, 1) OVER(PARTITION BY Cust_Id ORDER BY Calendar_Dt) AS gap
    FROM
        transactions
),
group_tab AS(
SELECT
    Cust_Id, Calendar_Dt, pre_date, gap,
     SUM(CASE WHEN gap = 1 or gap is null THEN 0 ELSE 1 END) OVER(PARTITION BY Cust_Id ORDER
BY Calendar_Dt)
        AS grp
FROM
    gap_tab),
 count_tab AS(
SELECT  Cust_Id, Calendar_Dt, pre_date, gap, grp ,COUNT(*) OVER(PARTITION BY Cust_Id , grp) AS
cont
FROM group_tab
)
SELECT DISTINCT Cust_Id, MAX(cont) OVER(PARTITION BY Cust_Id) AS max_consecutive_days
FROM count_tab
ORDER BY Cust_Id
    ;
```

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

```
    WITH spend_tab AS(
    SELECT Cust_Id ,Calendar_Dt ,Amt_LE ,SUM(Amt_LE) OVER(PARTITION BY Cust_Id ORDER BY
    Calendar_Dt ) AS TOTAL_SPEND
    FROM transactions
    ),
    sh_spend AS(
    SELECT Cust_Id,Calendar_Dt,Amt_LE,max(TOTAL_SPEND) over (partition by Cust_Id) as max_spend
     FROM spend_tab
     WHERE TOTAL_SPEND - Amt_LE < 250
    ),
    spend_days AS(
    SELECT  distinct  Cust_Id , COUNT(Calendar_Dt) OVER(PARTITION BY Cust_Id) AS TOTAL_DAYS
    FROM sh_spend
    where max_spend >=250
    ORDER BY Cust_Id
    )
    SELECT ROUND(AVG(TOTAL_DAYS)) ||' Days'AS AVG_DAYS FROM spend_days ;
```