# Q1: Exploring the data

The data provided shows a transactions for each product (Stock Code) in each Invoice made.

Every entry records the invoice number, product purchased, customer ID, date of purchase, price and quantity of that product as well as the country (the data provided is for United Kingdom only).

* Get the number of records in data set.

*-- count number of orders*

select count(INVOICE) from retail;

* Count distinct the number of orders to and compare it to the last result to see if there is any repetition.

*-- count distinct number of orders*

select count(distinct INVOICE) from retail ;

* The result shows that order ID is not unique for every record.
* The following query get the number of occurrence for the combination of customer id, order id and product id.

*-- count the occurence of orders and ,products andcustomer id*

select count(\*)

from (select INVOICE, STOCKCODE,CUSTOMER\_ID from retail group by INVOICE, STOCKCODE,CUSTOMER\_ID) ;

* The query shows the total number of the records in data set. Meaning that the combination of these three columns makes a unique record.
* The following query shows the first and last date of purchase and the time span of the data provided.

*-- using CTE to query last and first of invoice evermade*

with invoicedate as ( select

first\_value(INVOICEDATE)

over(order by to\_date(INVOICEDATE,'MM/DD/YYYY HH24:mi')) first\_purchase

,first\_value(INVOICEDATE)

over(order by to\_date(INVOICEDATE,'MM/DD/YYYY HH24:mi') desc ) last\_purchase

from retail)

*-- query firist and last date and calculate the time sapn of the data provided*

select first\_purchase, last\_purchase,

to\_date( to\_char(to\_date(last\_purchase,'MM/DD/YYYY HH24:mi'),'MM/DD/YYYY') ,'MM/DD/YYYY')

-to\_date( to\_char(to\_date(first\_purchase,'MM/DD/YYYY HH24:mi'),'MM/DD/YYYY') ,'MM/DD/YYYY')as span

from invoicedate

group by first\_purchase, last\_purchase;

* The following query count how many different products in the data

with product as(

select count(1) as product from retail group by STOCKCODE)

select count(product) as "count of products"

from product ;

* Finding the most ordered product

*-- ranking the most ordered product*

select

STOCKCODE,sum(QUANTITY) quantity , rank()over(order by sum(QUANTITY) desc) as rank

from retail

group by STOCKCODE

* Ranking months based on number of orders

*-- ranking month based on number of roders*

select to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:mi'),'MM/YYYY') order\_date

, count(INVOICE) "number\_of\_orders"

, rank() over(order by count(INVOICE) desc) as "Top\_demand\_month"

from retail

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

* Ranking month based on total revenue for each month

*-- ranking month based of revenue each month*

SELECT

TO\_CHAR(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'), 'MM/YYYY') AS order\_date,

SUM(quantity \* price) AS total\_revenue,

RANK() OVER (ORDER BY SUM(quantity \* price) DESC) AS top\_revenue\_month

FROM retail

GROUP BY TO\_CHAR(TO\_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'), 'MM/YYYY');

* Ranking customers based on number of orders

*-- ranking top customers based on number of orders*

select

CUSTOMER\_ID

, count(INVOICE) number\_of\_orders

,rank() over(order by count(INVOICE) desc ) top\_customer

from retail

group by CUSTOMER\_ID

# Q2:

*-- summarizing the data*

with detail\_retail as ( *-- creating CTE with the requred columns & add (sales) column*

select

CUSTOMER\_ID ,INVOICE

,to\_char(to\_date(INVOICEDATE,'MM/DD/YYYY HH24:mi'),'MM/DD/YYYY') as INVOICEDATE *-- get date column in required format*

, (QUANTITY\*PRICE) as sales *-- get sales column*

from retail )

,sumr\_retail as( *-- CTE holding summarized data (remove any duplicate rows) i.e one row for every customer and invoice*

select

CUSTOMER\_ID ,INVOICE

,INVOICEDATE , sum(sales) as sales *-- calculate sales column*

from detail\_retail

group by CUSTOMER\_ID ,INVOICE ,INVOICEDATE) *-- grouping rows to remove duplicates*

,sumr\_retail2 as ( *--CTE to get most-recent-date , and last purchase date for each customer*

select

CUSTOMER\_ID, INVOICE , INVOICEDATE , sales

, first\_value(INVOICEDATE) *-- most recent purchase date*

over(order by to\_date(INVOICEDATE,'MM/DD/YYYY') desc ) last\_purchase

, first\_value(INVOICEDATE) *-- most recent purchase date for each customer*

over(partition by CUSTOMER\_ID order by to\_date(INVOICEDATE,'MM/DD/YYYY') desc ) cust\_last\_purchase

from sumr\_retail

order by CUSTOMER\_ID)

, retail\_final as ( *-- calculate recency , monetary and frequency*

select distinct

CUSTOMER\_ID

*--how far is the last purchase*

, to\_date(last\_purchase ,'MM/DD/YYYY') - to\_date(cust\_last\_purchase ,'MM/DD/YYYY') as recency

*-- number of orders for each customer*

, count(INVOICE) over(partition by CUSTOMER\_ID order by CUSTOMER\_ID ) as frequency

,sum(sales) over(partition by CUSTOMER\_ID) as monetary *-- total sales for each customer*

from sumr\_retail2)

,final\_data as ( *-- allocating score based on RFM values*

select

CUSTOMER\_ID,recency,frequency,monetary

,ntile(5) over(order by recency desc) as r\_score

,ntile(5) over(order by ((frequency+monetary)/2)) fm\_score

from retail\_final )

select

CUSTOMER\_ID,recency,frequency,monetary

,r\_score,fm\_score

,case *-- customer segment column based on r\_score and fm\_score*

when r\_score in (5,4) and fm\_score in (5,4) then 'Champions'

when r\_score in (5,4,3) and fm\_score in (2,3) then 'Potential Loyalists'

when r\_score in (5,4,3) and fm\_score in (4,5,3) then ' Loyal Customers'

when r\_score in (5) and fm\_score in (1) then 'Recent Customers'

when r\_score in (4,3) and fm\_score in (1) then 'promising'

when r\_score in (2,3) and fm\_score in (2,3) then 'customers needing attention'

when r\_score in (2,1) and fm\_score in (5,3,4,1) then 'at risk'

when r\_score in (1) and fm\_score in (5,4) then 'can''t lose them'

when r\_score in (1) and fm\_score in (2) then 'hibernating'

when r\_score in (1) and fm\_score in (1) then 'lost'

end as cust\_segment

from final\_data;

# Q3:

# [A]

*-- CTE that calculates the row number for each customer's calendar dates*

WITH consecutive\_dates AS (

SELECT cust\_id, CALENDAR\_DT, ROW\_NUMBER() OVER (PARTITION BY cust\_id ORDER BY CALENDAR\_DT) AS rn

FROM customer

),

consecutive\_days (cust\_id, CALENDAR\_DT, rn, days) AS (

*-- Select the first row for each customer, setting the number of consecutive days to 1*

SELECT cust\_id, CALENDAR\_DT, rn, 1 AS days

FROM consecutive\_dates

WHERE rn = 1

UNION ALL

*-- For subsequent rows, compare the current row's calendar date to the previous one to determine if the dates are consecutive*

SELECT cd.cust\_id, cd.CALENDAR\_DT, cd.rn,

CASE

WHEN cd.CALENDAR\_DT = cd2.CALENDAR\_DT + 1 THEN cd2.days + 1 *-- If the dates are consecutive, set the number of consecutive days to the previous row's number of consecutive days plus 1*

ELSE 1 *-- else, reset the number of consecutive days to 1, indicating the start of a new range of consecutive days*

END

FROM consecutive\_dates cd

INNER JOIN consecutive\_days cd2 ON cd.cust\_id = cd2.cust\_id AND cd.rn = cd2.rn + 1 *-- Join the current row to the previous row based on the customer ID and the row number*

)

*-- Select the customer ID and the maximum number of consecutive days for each customer*

SELECT cust\_id, MAX(days) AS max\_consecutive\_days

FROM consecutive\_days

GROUP BY cust\_id;

# [B]

with detail as ( *-- using CTE and calculate running total and number of days/transaction req.for later calculation*

select CUST\_ID

,count(CALENDAR\_DT) over(partition by CUST\_ID order by CALENDAR\_DT rows between unbounded preceding and current row ) as days

,sum(AMT\_LE) over(partition by CUST\_ID order by CALENDAR\_DT rows between unbounded preceding and current row) as running\_amt

from customer)

*-- getting average of days that reach the threshold of 250*

select avg(days) as average\_days

from detail

where running\_amt >= 250;