Queries

Q1)

*--q1*

*--- customer who acheive most sales*

*-----first i will sum price for each customer to get sales for each customer*

*---second i will rank customer per each sales the highst will ranking 1 and the last will be 110*

*----here we find that the highst customer achieve sales is 12748*

select customer\_id,sales\_,rank() over(order by sales\_ desc) rank\_

from

(select customer\_id,sales\_

from(

select customer\_id ,invoicedate,price,sum(price) over(partition by customer\_id order by invoicedate rows between unbounded preceding and unbounded following ) as sales\_

from tableRetail

)

group by customer\_id,sales\_

order by sales\_ desc);

*----------------------------------------------------------------------------*

*--- second query*

*-- which month achieve highst sales the highst month on sales is november and the less month is jaun*

select month, sales\_per\_month

from(

select m.\* ,sum(price) over(partition by month) as sales\_per\_month

from

(select t.\*,to\_char(to\_date(invoicedate,'mm/dd/yyyy hh24:mi'),'mm')as month

from tableRetail t) m

order by sales\_per\_month desc)

group by month , sales\_per\_month

order by sales\_per\_month desc;

*----------------------------------------------------------------------------------------------------------------------------*

*----3query*

select \*

from tableRetail

*-- the data contain sales in two years 2010 and 2011 iwill show which quarter in 2years achieve highst sales*

*-- the highst quarter achieve sales is the 4 quarter this mean almostly this because the offer at the end of the year and the black friday sales*

select distinct quarter,sum(price) over(partition by quarter) as quarter\_sales

from(

select t.\*,to\_char(to\_date(invoicedate,'mm/dd/yyyy hh24:mi'),'q')as quarter

from tableRetail t)

order by quarter\_sales desc;

*-------------------------------------------------------------------------------------------------------------------*

*----query 4*

*-- we have 2335 stock saled in 2010 and 2011*

*-- we want see which stock is return highst sales*

select count(distinct stockcode)

from tableRetail;

*--- which stock return highst sales*

*-- the highst stock is the stock with code : M so we can increase this stock to achieve more sales*

select distinct stockcode,stock\_sales

from(

select stockcode,quantity,price ,sum(price) over(partition by stockcode) stock\_sales

from tableRetail)

order by stock\_sales desc;

*-----------------------------------------------------------------------------------------------------------*

*---q5which year acheive highst sales to see the sales are increasing or not*

*----2011 is highst than 2010*

*----in 2011 achieve sales 29489.72 more than 2010*

select year ,year\_sales,lag(year\_sales) over(order by year\_sales ) as lag ,(year\_sales-nvl(lag(year\_sales) over(order by year\_sales ),0)) as diff

from(

select distinct year ,sum(price) over(partition by year) as year\_sales

from(

select t.\*,to\_char(to\_date(invoicedate,'mm/dd/yyyy hh24:mi'),'yyyy')as year

from tableRetail t)

order by year\_sales desc);

*-------------------------------------------------------------------------------------------------------*

*----q6 average quantity per each stock to see which stock is more quantity*

select distinct stockcode ,avg(quantity) over(partition by stockcode) q\_avg

from tableRetail

order by q\_avg desc;

Q2)

*--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*

select distinct customer\_id,recency,frequency,mentory,r\_score,fm\_score,

case

when r\_score =4 and fm\_score=5 or r\_score=5 and fm\_score=5 or r\_score=5 and fm\_score=4 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) then 'Hibernating'

else 'Lost'

end cust\_seement

from(

select distinct customer\_id,recency,frequency,mentory,r\_score,ntile(5) over(order by avg\_fm\_score) as fm\_score

from(

select distinct customer\_id,recency,frequency,mentory,ntile(5) over(order by recency desc) as r\_score,(frequency+mentory)/2 as avg\_fm\_score

from(

select distinct customer\_id,last\_cust,max\_date,recency,count(invoice) over(partition by customer\_id) as frequency,sum(price) over(partition by customer\_id) as mentory

from (

select distinct customer\_id,price,invoice,last\_cust,max\_date,round(max\_date-last\_cust) as recency

from(

select distinct customer\_id,price ,invoice ,max(to\_date(invoicedate,'mm/dd/yyyy hh24:mi'))over() max\_date,

last\_value( to\_date(invoicedate , 'MM/DD/YYYY hh24:mi'))over(partition by customer\_id order by to\_date(invoicedate , 'MM/DD/YYYY hh24:mi') rows between unbounded preceding and unbounded following ) last\_cust

from tableretail) ))))

order by customer\_id;

Q3-A)

*-Q3*

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

*---My algorithm*

*---- first iwill select the prev of each transaction day to get different between them*

*---- second i will get diffeet between the current day (transaction day ) and the previous day in the order and i will ranking its partition by cust\_id order by calendar\_dt desc*

*---- third i will filterf it where diff between current day and prev not = null or not=1 and i will get diff between each rank and the prev rank this will return all number of connceive for each customer*

*----fourth iwill get max conn days for each customer*

select cust\_id,max(con\_days) max\_con\_days

from(

select gap.\*,lag(rank\_) over(partition by cust\_id order by calendar\_dt desc) as lag,rank\_-nvl(lag(rank\_) over(partition by cust\_id order by calendar\_dt desc),0) as con\_days

from(

select l.\*,calendar\_dt-perv\_day as diff, rank ()

over (partition by cust\_id order by calendar\_dt desc) as rank\_

from

(

select cust\_id,calendar\_dt,lead (calendar\_dt)

over(partition by cust\_id order by calendar\_dt desc ) as perv\_day

from customer\_purchacing) l) gap

where diff!=1 or diff is null

)

group by cust\_id

order by max\_con\_days desc

;

Q3-B)

*---q3 b*

*----------------------------------------------------*

*---*

*--1 first t want to sum amt\_le partirion by cusromer*

*--2 i will ranking each row(transaction) partition by customer oreder by date to make me able to access the nunmber of the transaction that customer acheive 250*

*--3 new column calendar\_dt - first day that the customer make transaction (dufference between first day and the current day(wich the day the customer achieve 250)*

*--4 iwill extract notransaction and day for each customer distinct and get averge*

select avg(trans),avg(days)

from(

select distinct cust\_id,first\_value(num\_of\_trans) over(partition by cust\_id order by calendar\_dt) trans,first\_value(no\_days) over(partition by cust\_id order by calendar\_dt) days

from(

select customer\_purchacing.\* , sum(amt\_le) over(partition by cust\_id order by calendar\_dt) as sum\_sales,rank() over(partition by cust\_id order by calendar\_dt) as num\_of\_trans,

calendar\_dt -first\_value(calendar\_dt) over(partition by cust\_id order by calendar\_dt) as no\_days

from customer\_purchacing)

where sum\_sales>=250

order by cust\_id);