- First, we need to take an overlook for the total revenue and average unit price per year

*--Q1: what is the average price unit and total revenue per year?*

SELECT DISTINCT (Years) , Round(avg(price\*quantity)) AS Avg\_price\_unit,

sum(price\*quantity) AS total\_revenue

FROM

(SELECT INVOICE,

STOCKCODE,

QUANTITY,

INVOICEDATE,

PRICE,

CUSTOMER\_ID,

COUNTRY,

to\_char(to\_date(invoicedate, 'mm/dd/yyyy hh24:mi'), 'yyyy') AS Years

FROM tableretail)

GROUP BY Years

ORDER BY total\_revenue DESC,

avg\_price\_unit DESC;

we found that 2011 has higher average unit price and revenue than 2010.

Table

Description automatically generated

So we assumed that this high revenue is due to high invoice transactions take place in each month in 2011and then we asked ..

*--Q2: how many invoices we make per month?*

SELECT distinct(months\_of\_year) ,

count(invoice) over(PARTITION BY months\_of\_year order by months\_of\_year) AS no\_of\_invoices\_per\_day

FROM

(SELECT INVOICE,

STOCKCODE,

QUANTITY,

INVOICEDATE,

PRICE,

CUSTOMER\_ID,

COUNTRY,

to\_char(to\_date(invoicedate, 'mm/dd/yyyy hh24:mi'), 'MM/YYYY') AS months\_of\_year

FROM tableretail);

And it seems like most frequent invoices took place in the last months of year 201l and it should be considered that transactions that took place in 2010 were only in December and maybe that’s why it has relatively fewer revenue.

Table

Description automatically generated

Now, we’ve become more interested in the stocks that return highest revenue, so we asked what are the high ranked stocks sold?

*-- Q3: top 10 stock revenue per months of year*

select\*

from

(WITH t1 AS

(SELECT to\_char(to\_date(invoicedate, 'mm/dd/yyyy hh24:mi'), 'MM/YYYY') AS months\_of\_year,

stockcode,

sum(quantity\*price) AS revenue

FROM tableretail

GROUP BY invoicedate,

stockcode)

SELECT months\_of\_year,

stockcode,

revenue,

rank () OVER (PARTITION BY months\_of\_year

ORDER BY revenue DESC) AS rank\_

FROM t1

ORDER BY months\_of\_year DESC)

where Rank\_ <= 10;

Table

Description automatically generated

For example here in December 2011 the top stock sold based on its revenue is stockcode: 21787, and in December 2010 top stoke sold is stockcode: 17096.

As known that the revenue is not only dependent on the unit price, but also dependent on the quantities sold per invoice, so if wanted to compare each quantity with the last highest quantity for each invoice we would ask..

*--Q4: what is the last highest quantity for which an order was sold?*

SELECT invoice,

quantity,

LAG(quantity, 1) OVER (PARTITION BY invoice

ORDER BY quantity DESC) last\_highest\_quantity

FROM tableretail;

Table

Description automatically generated

As shown here the last highest quantity for invoice 536415 is “blank” because it’s already the highest value and with each unique invoice the last highest quantity will be black because it has no higher quantity to be compared to.

If we wanted to study the customer behavior in purchasing and see how long it takes them to do another purchase , we would ask ..

*--Q5: How many days after the first purchase of a customer was the next purchase made?*

SELECT invoicedate,

customer\_id,

round(to\_date(invoicedate, 'MM/DD/YYYY hh24:mi')- FIRST\_VALUE(to\_date(invoicedate, 'MM/DD/YYYY hh24:mi')) OVER (PARTITION BY customer\_id

ORDER BY invoicedate)) next\_order\_gap

FROM tableretail

ORDER BY customer\_id,

next\_order\_gap;

Table

Description automatically generated

The negative next order gap indicates that these are purchases happen in in 2010 and the 0 indicates that this is the customer first order for example for customer 12748 his/her second order was after 40 days.

Finally, we should be interested in understanding the company’s first Quarter performance measured by its revenue and the number of transactions committed.

*--Q6: Running total revenue and total transaction for each invoice in the 1st Q per day*

SELECT distinct(invoicedate),

invoice,

SUM (quantity\*price) OVER (PARTITION BY invoice

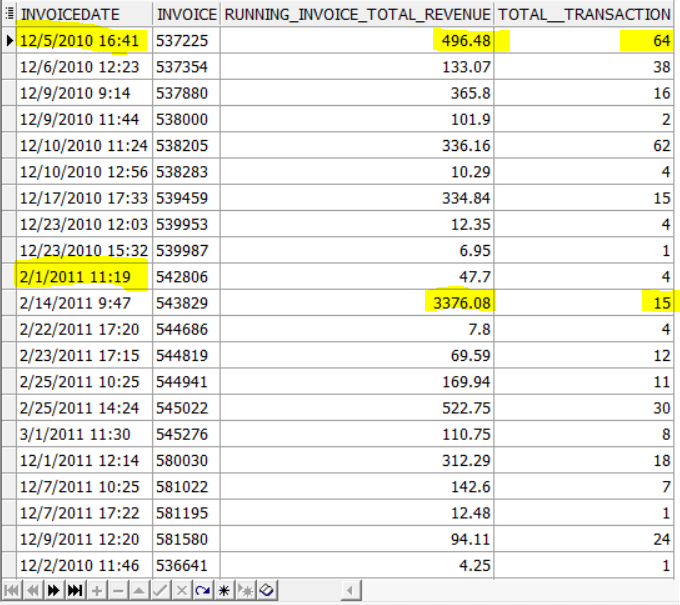
ORDER BY invoicedate) running\_invoice\_total\_revenue,

COUNT (invoice) over(PARTITION BY invoice

ORDER BY invoicedate) AS total\_\_transaction

FROM tableretail

WHERE invoicedate BETWEEN '12/1/2010 15:38' AND '3/1/2011 14:53';



This table tells a lot about the company’s performance and strategy , as you can see in the beginning of their business in 2010 there was many transactions ; however the revenue wasn’t very big and in 2011 the revenue started to increase but the transactions but the transactions didn’t , which means that the transaction wasn’t the factor of their revenue increase it was the unit price of their product instead.

*---------------------------------------------------------Q2 ----------------------------------------------*

*--RFM Calculation*

SELECT customer\_id,

NTILE(5) OVER(ORDER BY Recency) AS Recency,

NTILE(5) OVER(ORDER BY Monetory) AS Monetary

FROM( SELECT customer\_id,

Last\_Date,

ROUND(TO\_DATE('12/9/2011 12:30','MM/DD/YYYY HH24:MI')-Last\_Date) Recency,

Frequency,

Monetory

FROM (SELECT customer\_id,

MAX(TO\_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS Last\_Date,

COUNT(invoice) AS Frequency,

SUM(price\*quantity) AS Monetory

FROM tableretail

GROUP BY customer\_id

ORDER BY customer\_id) );

Table

Description automatically generated with low confidence

*--segmentation*

WITH segmentation AS

(

SELECT customer\_id,

NTILE(5) OVER(ORDER BY Frequency) AS Frequency,

NTILE(5) OVER(ORDER BY Recency) AS Recency,

NTILE(5) OVER(ORDER BY Monetory) AS Monetary,

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

NTILE(5) OVER(ORDER BY Fm\_Score) AS Fm\_Score

FROM( SELECT customer\_id,

Last\_Date,

ROUND(TO\_DATE('12/9/2011 12:30','MM/DD/YYYY HH24:MI')-Last\_Date) Recency,

Frequency,

Monetory,

ROUND(AVG(Frequency) over()) AS r\_Score,

ROUND(AVG(Monetory) over()) AS Fm\_Score

FROM (SELECT customer\_id,

MAX(TO\_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')) AS Last\_Date,

COUNT(invoice) AS Frequency,

SUM(price\*quantity) AS Monetory

FROM tableretail

GROUP BY customer\_id

ORDER BY customer\_id) )

)

SELECT customer\_id,

Frequency,

Recency,

Monetary,

r\_Score,

Fm\_Score,

CASE WHEN Recency = 5 AND Monetary = 5 THEN 'Champions'

WHEN Recency = 4 AND Monetary = 5 THEN 'Champions'

WHEN Recency = 5 AND Monetary = 4 THEN 'Champions'

WHEN Recency = 5 AND Monetary = 2 THEN 'Potential Loyalists'

WHEN Recency = 4 AND Monetary = 2 THEN 'Potential Loyalists'

WHEN Recency = 4 AND Monetary = 3 THEN 'Potential Loyalists'

WHEN Recency = 3 AND Monetary = 3 THEN 'Potential Loyalists'

WHEN Recency = 5 AND Monetary = 3 THEN 'Loyal Customers'

WHEN Recency = 4 AND Monetary = 4 THEN 'Loyal Customers'

WHEN Recency = 3 AND Monetary = 5 THEN 'Loyal Customers'

WHEN Recency = 3 AND Monetary = 4 THEN 'Loyal Customers'

WHEN Recency = 5 AND Monetary = 1 THEN 'Recent Customers'

WHEN Recency = 4 AND Monetary = 1 THEN 'Promising'

WHEN Recency = 3 AND Monetary = 1 THEN 'Promising'

WHEN Recency = 3 AND Monetary = 2 THEN 'Customers Needing Attention'

WHEN Recency = 2 AND Monetary = 3 THEN 'Customers Needing Attention'

WHEN Recency = 2 AND Monetary = 2 THEN 'Customers Needing Attention'

WHEN Recency = 2 AND Monetary = 5 THEN 'At Risk'

WHEN Recency = 2 AND Monetary = 4 THEN 'At Risk'

WHEN Recency = 1 AND Monetary = 3 THEN 'At Risk'

WHEN Recency = 1 AND Monetary = 5 THEN 'Cannot Lose Them'

WHEN Recency = 1 AND Monetary = 4 THEN 'Cannot Lose Them'

WHEN Recency = 1 AND Monetary = 2 THEN 'Hibernating'

WHEN Recency = 1 AND Monetary = 1 THEN 'Lost'

END segmentation

FROM segmentation

ORDER BY Recency DESC, Monetary DESC;

Graphical user interface, application, table, Excel

Description automatically generated