

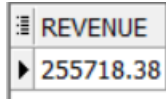
Analytical SQL Case Study

Part1:

Here in the exploration phase, I wanted to explore the data and I focused on sales and went with drill down approach

1-

```
select sum(Quantity * Price) as revenue  
from tableRetail;
```

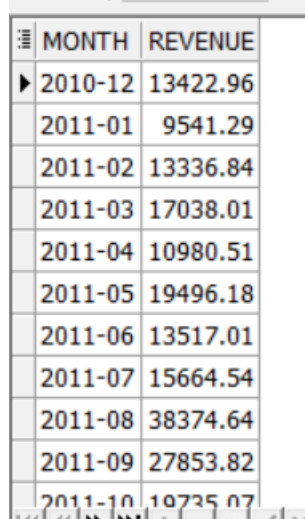


REVENUE
255718.38

In the first query, I went with an overall view of the sum of sales

2-

```
with retail_data as (  
select quantity, price, to_char(to_date(invoicedate, 'mm/dd/yyyy hh24:mi'), 'yyyy-mm') as month  
from tableretail)  
select month, sum(quantity * price) as revenue  
from retail_data  
group by month  
order by month asc;
```



MONTH	REVENUE
2010-12	13422.96
2011-01	9541.29
2011-02	13336.84
2011-03	17038.01
2011-04	10980.51
2011-05	19496.18
2011-06	13517.01
2011-07	15664.54
2011-08	38374.64
2011-09	27853.82
2011-10	19735.07

The second step is going to less grain and I went with the total sales for each month, I used CTE here even if it's not better performance it at least offers better readability

3-

```
with retail_data as (  
select stockcode, sum(quantity) as totalquantity, sum(price * quantity) as  
totalrevenue, row_number() over (order by sum(price * quantity) desc) as revenuerank  
from tableretail
```

```

group by stockcode
order by totalrevenue desc)
select stockcode, totalquantity, totalrevenue, revenuerank
from retail_data
where revenuerank <= 5

```

STOCKCODE	TOTALQUANTITY	TOTALREVENUE	REVENUERANK
84879	6117	9114.69	1
22197	5918	4323.1	2
21787	5075	4059.35	3
22191	451	3461.2	4
23203	1803	3357.44	5

Here I went more into detail and focused on products, I chose to select the top 5 products by the total sales and the total quantity for each product

4-

```

with retail_data as (
select stockcode, count(*) as purchases, row_number() over (order by count(*) desc) as
revenuerank
from tableretail
group by stockcode
order by purchases desc)
select stockcode, purchases, revenuerank
from retail_data
where revenuerank <= 5;

```

STOCKCODE	PURCHASES	REVENUERANK
84879	61	1
22086	56	2
850998	53	3
22197	52	4
85123A	49	5

Here the query is at the same level of detail as the previous one, it also looks at the product level but here come the top 5 products by quantity we can use this and the previous one to compare if more quantity = more sales or no

5-

```

with retail_data as (

```

```

select customer_id, sum(quantity) as totalquantity, sum(price * quantity) as
totalrevenue, row_number() over (order by sum(price * quantity) desc) as revenuerank
from tableretail
group by customer_id
order by totalrevenue desc)
select customer_id, totalquantity, totalrevenue, revenuerank
from retail_data
where revenuerank <= 5

```

Here we look at the angle of customers we have the top 5 customers by revenue and the total quantity of products they bought this will be in line with the following analysis when segmenting customers

	CUSTOMER_ID	TOTALQUANTITY	TOTALREVENUE	REVENUERANK
▶	12931	28004	42055.96	1
	12748	25748	33719.73	2
	12901	23075	17654.54	3
	12921	9526	16587.09	4
	12939	4876	11581.8	5

6-

here is an extra step i took because of questions I had on my mind

```

SELECT CORR(num_visits, avg_money_spent)
FROM (
  SELECT
    Customer_ID,
    COUNT(DISTINCT INVOICE) AS num_visits,
    AVG(Price * Quantity) AS avg_money_spent
  FROM tableretail
  GROUP BY Customer_ID
)

```

	CORR_VISIT_REV
▶	-0.00579528506046825

Here I wanted to look at the correlation between the number of visits and the money spent, I couldn't look at the number of visits and the sum of revenue as it has no meaning it obvious that it will increase also so I looked at the average money spent to see if coming again will increase the amount spent and according to that the result was (-.005) it decrease but with the small effect but I think they should look at this aspect to increase revenue

```

SELECT
  CORR(Quantity, Price * Quantity) AS correlation
FROM
  Tableretail;

```

CORRELATION
▶ 0.701055428719604

Here I wanted to look at the correlation between quantity and revenue it will obviously will be positive but I wanted to find it's strength will it be a strong increase or not and it did it was (0.7)

Part2:

```
with retail_data as (
SELECT Customer_ID,
       ROUND(((SELECT MAX(TO_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')) FROM tableretail) -
MAX(TO_DATE(invoicedate, 'mm/dd/yyyy hh24:mi')))) AS Recency,
       COUNT(DISTINCT INVOICE) as Frequency,
       ROUND(SUM(Price * Quantity),2) as Monetary
FROM
  tableretail
GROUP BY
  Customer_ID)
SELECT Customer_ID,
       Recency,
       Frequency,
       Monetary,
       NTILE(5) OVER (ORDER BY Recency desc)as r_score,
       NTILE(5) OVER(ORDER BY ROUND((Monetary/Frequency),2))as fm_score,
       CASE
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 5 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 5 THEN 'Champions'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 5 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'Champions'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 4 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 5 THEN 'Champions'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 5 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 2 THEN 'Potential Loyalists'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 4 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 2 THEN 'Potential Loyalists'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 3 THEN 'Potential Loyalists'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 4 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 3 THEN 'Potential Loyalists'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 5 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 3 THEN 'Loyal Customers'
         WHEN NTILE(5) OVER (ORDER BY Recency desc) = 4 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'Loyal Customers'
```

```

        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 5 THEN 'Loyal Customers'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'Loyal Customers'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'Recent Customers'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 4 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 1 THEN 'Promising'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 1 THEN 'Promising'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 3 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 2 THEN 'Customers Needing Attention'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 2 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 3 THEN 'Customers Needing Attention'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 2 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 2 THEN 'Customers Needing Attention'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 2 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 5 THEN 'At Risk'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 2 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'At Risk'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 1 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 3 THEN 'At Risk'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 1 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 5 THEN 'Cant Lose Them'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 1 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 4 THEN 'Cant Lose Them'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 1 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 2 THEN 'Hibernating'
        WHEN NTILE(5) OVER (ORDER BY Recency desc) = 1 AND NTILE(5) OVER(ORDER
BY ROUND((Monetary/Frequency),2)) = 1 THEN 'Lost'
        END AS cust_segment
FROM
retail_data

```

CUSTOMER_ID	RECECY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	CUST_SEGMENT
12875	143	2	343.23	2	2	Customers Needing Attention
12883	24	4	703.47	4	2	Potential Loyalists
12923	64	1	176.97	3	2	Customers Needing Attention
12888	214	2	354.12	1	2	Hibernating
12951	8	6	1064.07	4	2	Potential Loyalists
12915	148	2	363.65	2	2	Customers Needing Attention
12895	42	2	372.8	3	2	Customers Needing Attention
12893	30	1	188.14	3	2	Customers Needing Attention
12871	84	2	380.64	2	2	Customers Needing Attention
12879	44	3	573.22	3	2	Customers Needing Attention
12832	32	2	383.03	3	2	Customers Needing Attention
12733	26	2	627.52	4	2	Potential Loyalists

Here I wanted to segment customers based on recency and monetary and frequency as they can be considered as indication for customers activities .

First was recency I took the maximum date as a reference as the most recent date and I subtracted from it the max date for each customer.

Second, is the frequency here I used count the distinct invoices.

Third is the monetary it's the sum of price*quantity

Then the r_score is segmenting the customers into 5 groups descending as the most recent customer gets into group 5.

The last one is fm_score is the monetary value divided by the frequency here the output is the average spending per each visit which I found is a good indication for spending and it combines both of them.

Then after that, I divided it into 5 groups also and segmented them to the required criteria.

Part 4 :

The following are store data set but from another source we wanted to know What is the maximum number of consecutive days a customer made purchases and On average, How many days/transactions does it take a customer to reach a spent threshold of 250 L.E?

A-

at first i thought of using lag by getting the transaction date and subtracting the lag from it if it's = 1 then count this column where it's equal to 1, but the problem that faced me was that if a customer skipped 1 day and then continued they will be added not be separated.

I found a way on the internet that used a brilliant method for this case

```
WITH daily_totals AS (
  SELECT
    cust_id,
```

```

transaction_date,
COUNT(*) OVER (PARTITION BY cust_id, transaction_date - ROWNUM) AS consec_days
FROM
customer_transactions
)
SELECT
cust_id,
MAX(consec_days) AS max_consec_days
FROM
daily_totals
GROUP BY
Cust_id;

```

CUST_ID	MAX_CONSEC_DAYS
145272	4
1026223	2
1767267	4

Here it took the transaction date and made it - row number so it will get all date that is following each other to the same day they will be counted together and then partitioned by the customer id and the date we made above it will avoid the problem we faced above as if it skipped a day then continued the day that we will partition by will be a different day

B-

```

SELECT AVG(diff_days)
FROM (
SELECT cust_id,
MIN(CASE WHEN cumulative_sum >= 250 THEN CALENDAR_DT END) -
MIN(CALENDAR_DT) AS diff_days
FROM (
SELECT cust_id, CALENDAR_DT, AMT_LE,
SUM(AMT_LE) OVER (PARTITION BY cust_id ORDER BY CALENDAR_DT) AS
cumulative_sum
FROM sales_data
where AMT_LE !=0)
GROUP BY cust_id)

```

AVG_DAYS
10.8072427393331

Here we wanted to figure out the average days the customers take to reach 250 L.E. Here I thought of it from the perspective of all customers, not each customer as the number will have no meaning.

Here I first calculated the sum of the amount spent partitioned by customer to get the cumulative money spent then in the outer query I got the first day that the customer reached the 250 thresholds by using min with case when instead of writing another sub-query and then subtracted it from the minimum date for this customer to get the time that took him to reach the amount,

In the outermost query, I got the average of this period

**** Note:- I used where amt_LE !=0 in the innermost query because I found a lot of transactions that have 0 as the amount paid I don't know how is this recorded without money paid either it's an error or he used a coupon or voucher and didn't pay anything. It made difference in the output without the condition it was 11.3 after the condition it's now 10.8****

I hope that the illustration and formatting were done well.Thanks