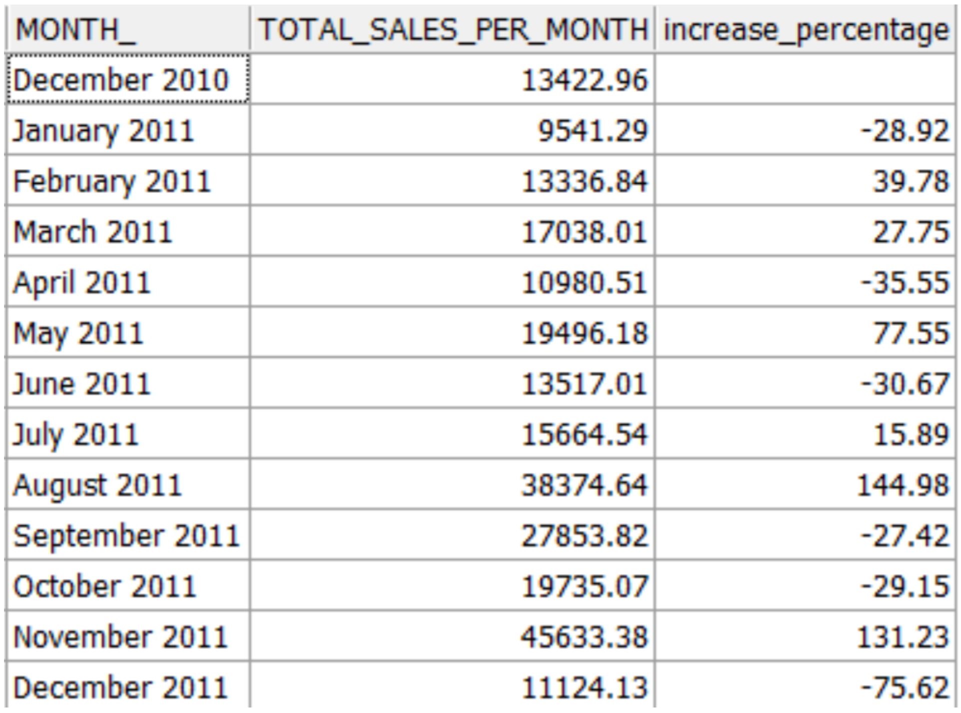
Analytical SQL Project

Background:
Customers has purchasing transaction that we shall be monitoring to get intuition behind each customer behavior to target the customers in the most efficient and proactive way, to increase sales/revenue , improve customer retention and decrease churn.

First Analyzing Sales Performance and Customer Behavior

a) This query calculates total per month, computes the percentage increase in sales compared to the previous month to compare sales trends over time .

```
with month_sales as (
select distinct
    to_char(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'), 'fmMonth YYYY') as month_,
    sum(price * quantity) over (partition by to_char(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'), 'fmMonth YYYY')) as
Total_sales_per_month
from
    tableRetail
    ),
lag as (
    select  month_,Total_sales_per_month,  lag(Total_sales_per_month, 1) over(order by to_date(month_, 'Mon-yyyy'))
prev_sales
    from month_sales
),
percent_ as (
    select month_, Total_sales_per_month , round(100*(Total_sales_per_month - prev_sales)/prev_sales,2)
"increase_percentage"
    from lag
    order by to_date(month_, 'Mon-yyyy')
)
select * from percent_;
```

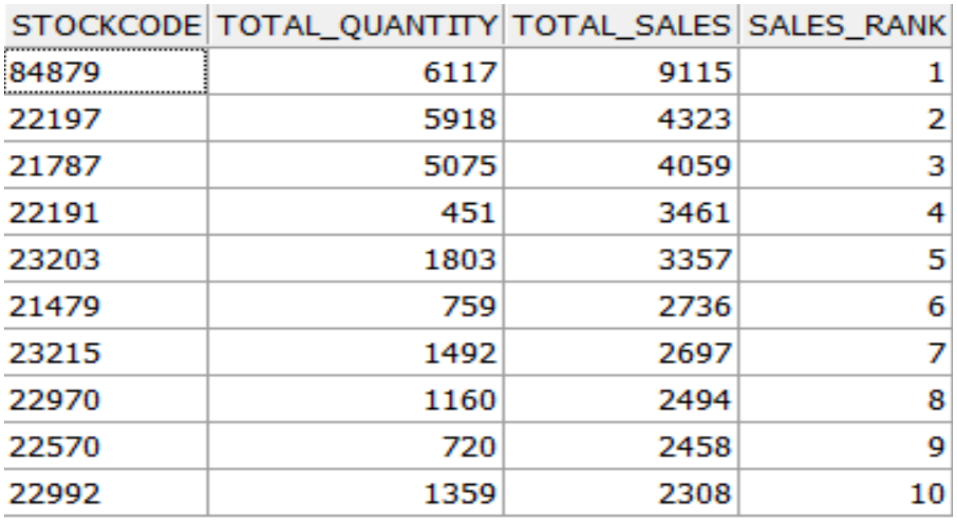


MONTH_	TOTAL_SALES_PER_MONTH	increase_percentage
December 2010	13422.96	
January 2011	9541.29	-28.92
February 2011	13336.84	39.78
March 2011	17038.01	27.75
April 2011	10980.51	-35.55
May 2011	19496.18	77.55
June 2011	13517.01	-30.67
July 2011	15664.54	15.89
August 2011	38374.64	144.98
September 2011	27853.82	-27.42
October 2011	19735.07	-29.15
November 2011	45633.38	131.23
December 2011	11124.13	-75.62

As we see in August-2011 and November-2011 have highest total sales which reflect increase percentage compared to previous month

b) This query retrieves information about the top 10 best-selling products. It calculates the total quantity sold, total sales amount, and assigns a sales rank based on the total sales amount.

```
select stockcode,  Total_quantity , Total_sales , Sales_rank
from (
select stockcode , sum(quantity) as Total_quantity ,
                   round(sum(price * quantity)) as Total_sales
                   , dense_rank() over (order by sum(price * quantity) desc) as Sales_rank
from tableRetail
group by stockcode)
where Sales_rank <= 10
order by Sales_rank ;
```

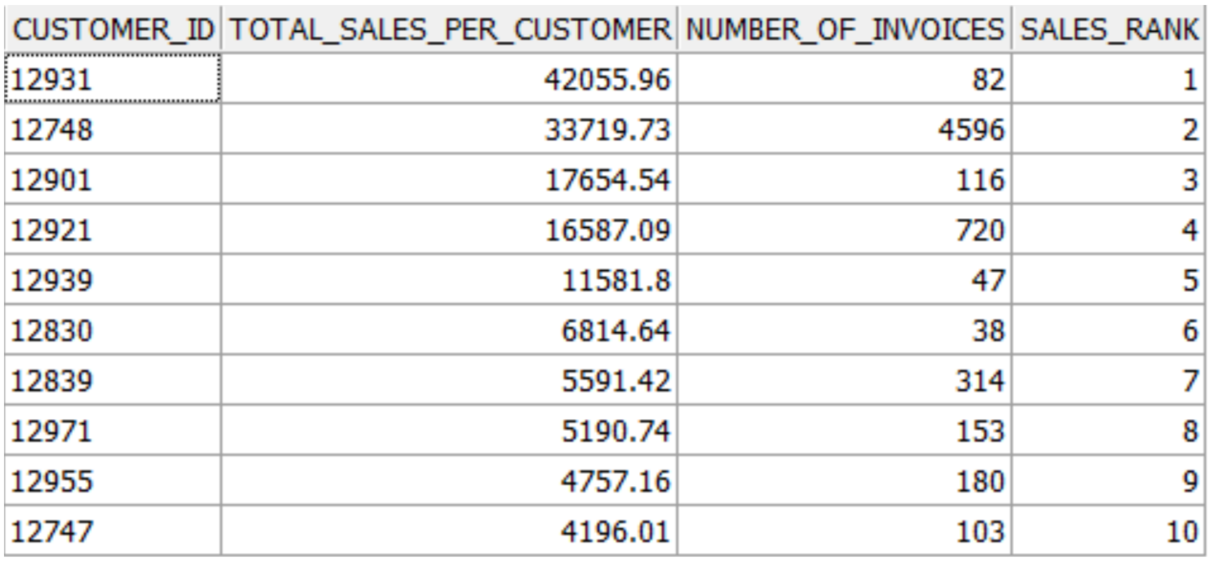


STOCKCODE	TOTAL_QUANTITY	TOTAL_SALES	SALES_RANK
84879	6117	9115	1
22197	5918	4323	2
21787	5075	4059	3
22191	451	3461	4
23203	1803	3357	5
21479	759	2736	6
23215	1492	2697	7
22970	1160	2494	8
22570	720	2458	9
22992	1359	2308	10

As we see products 84879 , 22197 , 21787 have the highest profitability and also have the highest quantity purchased

c) This query calculate the top 10 customers with the highest total sales and presents , the number of invoices for each customer

```
select customer_id ,total_sales_per_customer , Number_of_invoices ,Sales_rank
from (
select  customer_id , sum(quantity * price ) as  total_sales_per_customer ,
                              count(invoice)  as Number_of_invoices ,
                              dense_rank() over (order by sum(price * quantity) desc) as Sales_rank
from tableRetail
group by customer_id
)
where sales_rank <= 10
order by Sales_rank ;
```

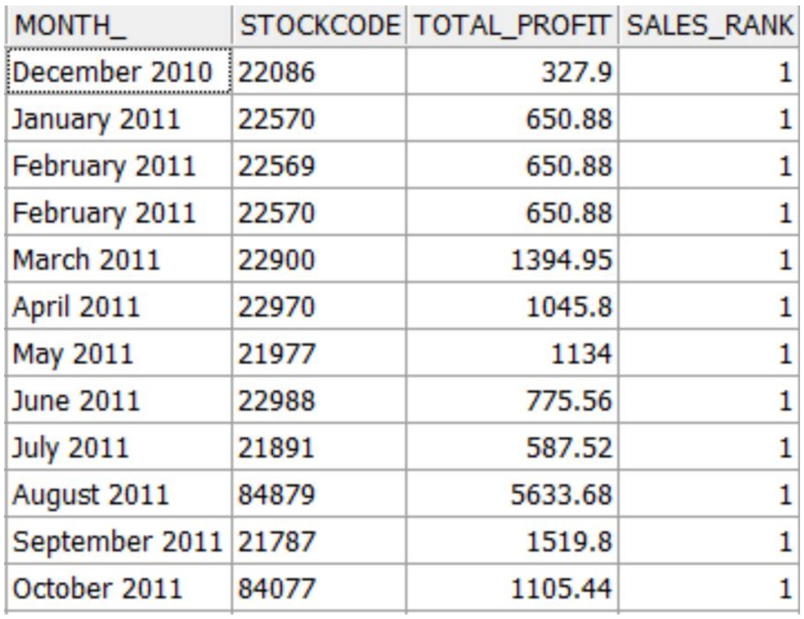


CUSTOMER_ID	TOTAL_SALES_PER_CUSTOMER	NUMBER_OF_INVOICES	SALES_RANK
12931	42055.96	82	1
12748	33719.73	4596	2
12901	17654.54	116	3
12921	16587.09	720	4
12939	11581.8	47	5
12830	6814.64	38	6
12839	5591.42	314	7
12971	5190.74	153	8
12955	4757.16	180	9
12747	4196.01	103	10

The customers with id's 12931 , 12748, 12901 have the highest Sales which mean there are valuable customers but Customers with id's 12748 , 12921 have the highest number of invoices

d) This query provides insights into the most profitable product for each month and their respective total profits

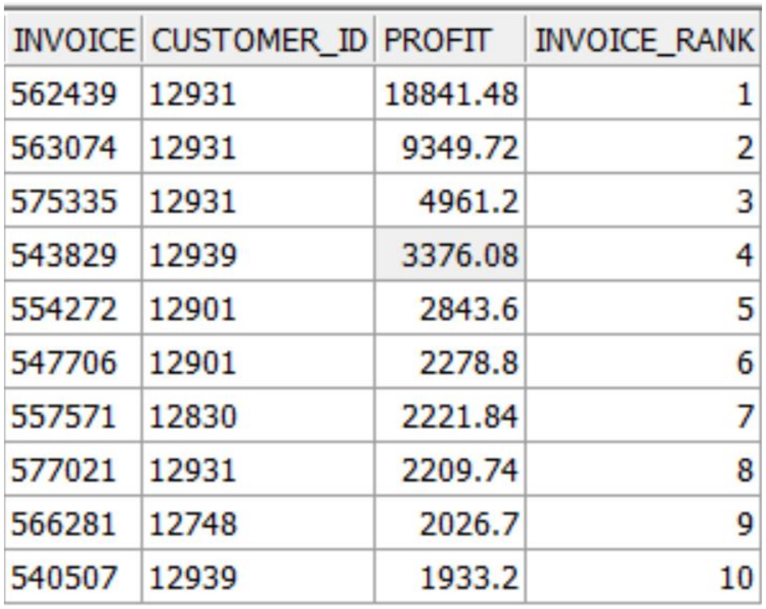
```
with product_month as (
select stockcode , (price*quantity) as profit , to_char(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'), 'fmMonth YYYY') as
month_
from tableRetail
),
most_purchased as (
   select month_ ,  stockcode , dense_rank() over(partition by month_ order by sum(profit) desc) as Sales_rank
                                   ,sum(profit) as total_profit
   from  product_month
   group by month_ , stockcode
   ),
top_rank as (
   select month_ , stockcode , total_profit , Sales_rank
   from most_purchased
   where Sales_rank =1
   )
select * from top_rank
order by to_date('01 ' || month_, 'DD Month YYYY');
```



MONTH_	STOCKCODE	TOTAL_PROFIT	SALES_RANK
December 2010	22086	327.9	1
January 2011	22570	650.88	1
February 2011	22569	650.88	1
February 2011	22570	650.88	1
March 2011	22900	1394.95	1
April 2011	22970	1045.8	1
May 2011	21977	1134	1
June 2011	22988	775.56	1
July 2011	21891	587.52	1
August 2011	84879	5633.68	1
September 2011	21787	1519.8	1
October 2011	84077	1105.44	1

e) This query helps identify the top-performing invoices based on their total profit and presents information about cust_id of it's owner

```
select invoice , customer_id , profit , invoice_rank
from (
select invoice , customer_id , sum(price * quantity) as profit ,
                                        dense_rank() over ( order by sum(price*quantity) desc ) as invoice_rank
from tableRetail
group by invoice , customer_id
)
where   invoice_rank <= 10;
```

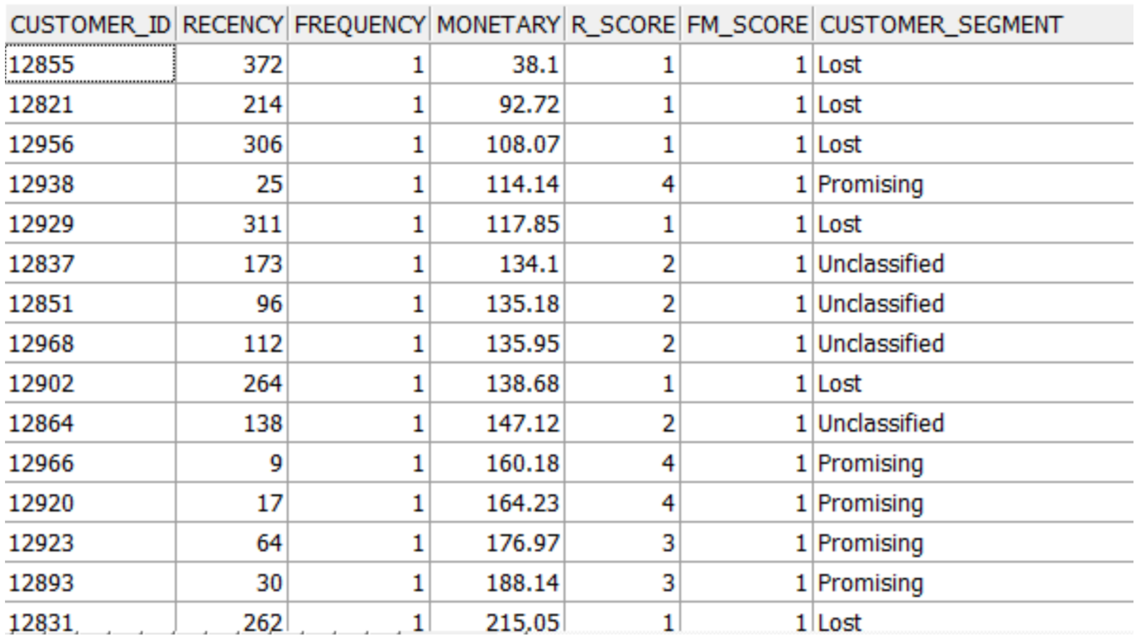


INVOICE	CUSTOMER_ID	PROFIT	INVOICE_RANK
562439	12931	18841.48	1
563074	12931	9349.72	2
575335	12931	4961.2	3
543829	12939	3376.08	4
554272	12901	2843.6	5
547706	12901	2278.8	6
557571	12830	2221.84	7
577021	12931	2209.74	8
566281	12748	2026.7	9
540507	12939	1933.2	10

As we can see the customer with id 12931 have 4 of the top 10 profitable invoice And customer 12901 have 2 invoices which mean they are valuable customers

After exploring the data let's segment customers based on recency, frequency, and monetary (RFM) scores, categorizing them into distinct segments such as "Champions," "Loyal Customers," "Promising," and "Lost," based on their purchasing behavior.

```
with table_1 as (
select  distinct customer_id ,round(maxdate - max(to_date(invoicedate, 'mm/dd/yyyy hh24:mi'))  over(partition by
customer_id)) as  recency
                                    ,count(distinct invoice) over(partition by customer_id) as  frequency
                                    ,round(sum(price * quantity) over(partition by customer_id),2) as monetary
from tableretail, (select max(to_date(invoicedate, 'mm/dd/yyyy hh24:mi')) maxdate from tableretail)
),
r_fm_scores as (
select customer_id ,recency , frequency , monetary , ntile(5) over(order by recency desc) as r_score
                                                               ,ntile(5) over(order by ((frequency + monetary) / 2) ) as fm_score
from table_1
),
cust_segment as (
select customer_id ,recency , frequency , monetary , r_score , fm_score ,
 CASE
        WHEN  (r_score= 5 AND  fm_score= 5) OR
              (r_score= 5 AND  fm_score= 4) OR
              (r_score= 4 AND  fm_score= 5) THEN 'Champions'

        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 = 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 = 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 'Can not lose them'

        WHEN (r_score = 1 AND fm_score = 2) THEN 'Hibernating'

        WHEN (r_score = 1 AND fm_score = 1) THEN 'Lost'

        ELSE 'Unclassified'
       END AS customer_segment
from r_fm_scores)
select * from cust_segment;
```



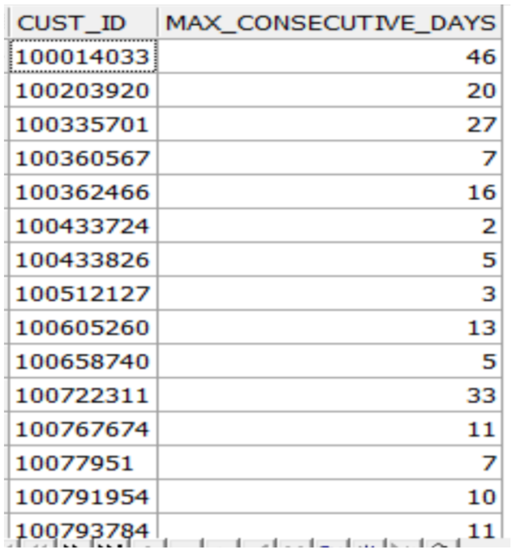
CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	CUSTOMER_SEGMENT
12855	372	1	38.1	1	1	Lost
12821	214	1	92.72	1	1	Lost
12956	306	1	108.07	1	1	Lost
12938	25	1	114.14	4	1	Promising
12929	311	1	117.85	1	1	Lost
12837	173	1	134.1	2	1	Unclassified
12851	96	1	135.18	2	1	Unclassified
12968	112	1	135.95	2	1	Unclassified
12902	264	1	138.68	1	1	Lost
12864	138	1	147.12	2	1	Unclassified
12966	9	1	160.18	4	1	Promising
12920	17	1	164.23	4	1	Promising
12923	64	1	176.97	3	1	Promising
12893	30	1	188.14	3	1	Promising
12831	262	1	215.05	1	1	Lost

Analyzing Customer Purchasing Behavior

a) Maximum consecutive purchase days per customer:

Calculates the maximum number of consecutive days on which each customer made purchases, indication customer engagement and loyalty

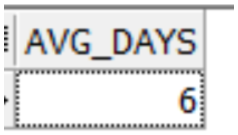
```
with difference as (
select cust_id, calendar_dt
              , lead(calendar_dt) over (partition by cust_id order by calendar_dt) as next_days
              , calendar_dt - row_number() over (partition by cust_id order by calendar_dt) as difference
from customer
),
calculate_days as (
select cust_id , calendar_dt , next_days , difference ,  count(*) over (partition by cust_id, difference order by
calendar_dt) as days_no
from difference
)
select cust_id , max(days_no) as Max_consecutive_days
from calculate_days
group by cust_id
```



CUST_ID	MAX_CONSECUTIVE_DAYS
100014033	46
100203920	20
100335701	27
100360567	7
100362466	16
100433724	2
100433826	5
100512127	3
100605260	13
100658740	5
100722311	33
100767674	11
10077951	7
100791954	10
100793784	11

b) Average Days or Transactions did the customer take to reach 250 le total sales

```
with customer_days as (
select cust_id , calendar_dt , sum(amt_le) over(partition by cust_id order by calendar_dt rows between unbounded
preceding and current row) as sum_amt
                                        ,count(*) over(partition by cust_id order by calendar_dt ) as total_days
from customer
where amt_le != 0
),
days_till_250 as (
select cust_id  , (min(total_days)) days_to_250
from customer_days
where sum_amt >= 250
group by cust_id
)
select round(avg(days_to_250)) from days_till_250;
```



AVG_DAYS
6