

Analytical SQl

Case Study

Analytical SQL Case Study

First Question:

1- Top 10 Bestselling Products:

```
--top 10 prdoducts
select distinct(stockcode), total_quant from(
select stockcode , sum(quantity) total_quant,
row_number() over ( order by sum(quantity) desc) as top
from tableretail
group by stockcode
)
where top <=10;
```

STOCKCODE	TOTAL_QUANT
84077	7824
84879	6117
22197	5918
21787	5075
21977	4691
21703	2996
17096	2019
15036	1920
23203	1803
21790	1579

- Explanation: This query lists the top 10 products by total quantity sold.
- **Meaning:** It helps the business identify its most popular products and ensure they are adequately stocked and promoted.

2- Top 5% of Customers:

```
= --top 5% of customers

select distinct(customer_id),total_sales,round(rankk,2) as rank from (
select customer_id, sum(quantity *price ) total_sales ,
percent_rank() over(order by (sum(quantity *price )) desc) as rankk
from tableretail
group by customer_id
)
where rankk<=0.05
```

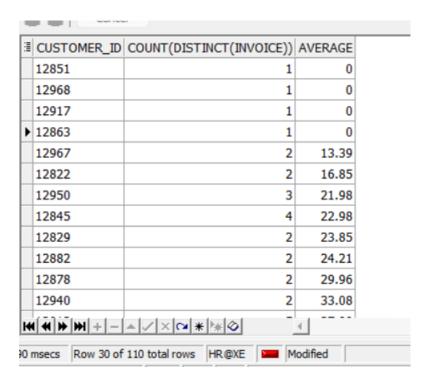
The output:

	_	TOTAL_SALES	KANK
Þ	12931	42055.96	0
	12748	33719.73	0.01
	12901	17654.54	0.02
	12921	16587.09	0.03
	12939	11581.8	0.04
	12830	6814.64	0.05

- Explanation: This query identifies the top 5% of customers based on their total spending.
- **Meaning:** It helps the business focus on its most valuable customers and tailor special offers or loyalty programs to retain their loyalty.

3- Customer Purchase Behavior Analysis:

```
select CUSTOMER_ID , count(distinct(invoice)),
round(avg (last_val - first_val),2) as average
from (
select CUSTOMER_ID , invoice,
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') range between unbounded preceding and unbounded following) as last_val ,
first_value(to_date(invoicedate,'mm/dd/yyyy HH24:MI')) over(partition by CUSTOMER_ID order by to_date(invoicedate,'mm/dd/yyyy HH24:MI')) as first_val
from tableretail)
group by customer_id
order by average;
```



- **Explanation:** This query helps understand how many times each customer made a purchase and calculates the average time between their first and last purchase.
- **Meaning:** By knowing how often customers buy and how long they remain active, the business can tailor marketing and engagement strategies to keep them coming back.

4- Average Purchase Value per Product:

```
select distinct(stockcode),total_quantity,total_sales,round( avg (total_sales/total_quantity) over(partition by stockcode) ,2) as avg from( select stockcode,sum(quantity) over(partition by stockcode) as total_quantity , sum(quantity*price) over (partition by stockcode) as total_sales from tableretail) order by total_sales desc ;
```

_				
∄	STOCKCODE	TOTAL_QUANTITY	TOTAL_SALES	AVG
١	84879	6117	9114.69	1.49
	22197	5918	4323.1	0.73
	21787	5075	4059.35	0.8
	22191	451	3461.2	7.67
	23203	1803	3357.44	1.86
	21479	759	2736.01	3.6
	23215	1492	2697.36	1.81
	22970	1160	2493.6	2.15
	22570	720	2458.08	3.41
	22992	1359	2308.05	1.7
	85099B	1130	2237.41	1.98
	23084	1194	2187.72	1.83
H	+ 4 4	- _ × * * *	Ø (

- **Explanation:** This query finds the average amount customers spend on each product, helping identify high-value items.
- **Meaning:** It helps the business focus on products that bring in more revenue per sale and adjust pricing or promotions accordingly.

5- Customer Lifetime Value:

Cancel						
∄	CUSTOMER_ID	TOTAL_SALES	DIFF	CLV		
١	12747	4196.01	366.96	11.43		
	12748	33719.73	372.98	90.41		
	12749	4090.88	209.77	19.5		
	12820	942.34	323.11	2.92		
	12821	92.72	0			
	12822	948.88	16.85	56.33		
	12823	1759.5	221.81	7.93		
	12824	397.12	0			
	12826	1474.72	362.79	4.06		
	12827	430.15	38.86	11.07		
	12828	1018.71	127.69	7.98		
	12829	293	23.85	12.29		
H	+ H + -	▲	* •			

- **Explanation:** This query calculates the expected revenue from each customer over their entire relationship with the business.
- **Meaning:** It helps the business understand the long-term value of its customers and prioritize efforts to retain the most valuable ones.

6- Customer Count per Month:

```
select count(distinct customer_id) as customer_count,
to_char(to_date(invoicedate,'mm/dd/yyyy HH24:MI'),'mm,yyyy') as month_year
from tableretail
group by to_char(to_date(invoicedate,'mm/dd/yyyy HH24:MI'),'mm,yyyy')
order by customer_count desc
```



- **Explanation:** This query counts the number of unique customers for each month.
- Meaning: It helps the business understand seasonal trends in customer activity and plan marketing campaigns accordingly.

7- Product Sales Quantity per Month:

```
--products and quantity sale for each month
select stockcode , sum(quantity) quant ,to_char(to_date(invoicedate,'mm/dd/yyyy HH24:MI'),'mm,yyyy') as month_year
from tableretail
group by stockcode,to_char(to_date(invoicedate,'mm/dd/yyyy HH24:MI'),'mm,yyyy')
order by quant desc
```

Ī	STOCKCODE	QUANT	MONTH_YEAR	
Þ	84077	5136	10,2011	
	84879	3880	08,2011	
	21977	2700	05,2011	
	22197	2540	08,2011	
	21787	1788	09,2011	
	17096	1728	12,2010	
	84879	1349	11,2011	
	84077	1248	04,2011	
	21787	1200	12,2011	
	22197	1136	11,2011	
	21787	1081	11,2011	
	20974	984	08,2011	
4	() H (())		× (~ * * ◊	

- Explanation: This query shows the quantity of each product sold for each month.
- Meaning: It helps the business analyze product popularity and adjust inventory levels or promotions based on demand trends.

8- Monthly Sales Growth Rate:

MONTH_YEAR	CURRENT_MONTH_SALES	PREVIOUS_MONTH_SALES	GROWTH_RATE_PERCENT
12,2010	13422.96	9541.29	40.68
01,2011	9541.29	13336.84	-28.46
02,2011	13336.84	17038.01	-21.72
03,2011	17038.01	10980.51	55.17
04,2011	10980.51	19496.18	-43.68
05,2011	19496.18	13517.01	44.23
06,2011	13517.01	15664.54	-13.71
07,2011	15664.54	38374.64	-59.18
08,2011	38374.64	27853.82	37.77
09,2011	27853.82	19735.07	41.14
10,2011	19735.07	45633.38	-56.75
11,2011	45633.38	11124.13	310.22

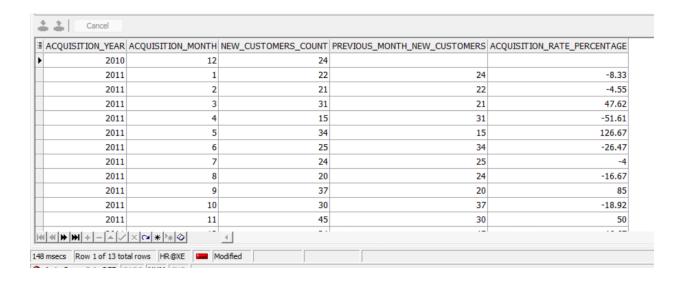
- **Explanation:** This query calculates how much sales have grown or declined from one month to the next.
- **Meaning:** By tracking these changes, the business gains insights into sales trends, enabling adjustments in inventory management and promotional strategies to optimize revenue generation during periods of growth and mitigate declines.

9- Customer Acquisition Rate:

```
with newcustomers as (
select extract(year from min(to_date(invoicedate,'mm/dd/yyyy hh24:mi'))) as acquisition_year,
extract(month from min(to_date(invoicedate,'mm/dd/yyyy hh24:mi'))) as acquisition_month,
count(distinct customer_id) as new_customers_count
from tableretail
group by extract(year from to_date(invoicedate,'mm/dd/yyyy hh24:mi')),
extract(month from to_date(invoicedate,'mm/dd/yyyy hh24:mi'))

select acquisition_month,
new_customers_count,
lag(new_customers_count) over (order by acquisition_year, acquisition_month) as previous_month_new_customers,
round(((inew_customers_count - lag(new_customers_count))
over (order by acquisition_year, acquisition_month)) / lag(new_customers_count)
over (order by acquisition_year, acquisition_month);

ver (order by acquisition_year, acquisition_month);
```



Explanation: This query measures how quickly the business is gaining new customers over a specific period.

Meaning: By monitoring customer acquisition, the business can evaluate the success of its marketing campaigns and refine its strategies to attract a larger customer base. This helps in optimizing resources and investments to drive sustainable growth.

Second Question:

```
--second question
2 • 🖂 with customer_summary as 🕻
              with Customer_summary os t
select distinct customer_id,
count(distinct invoice) over (partition by customer_id) as freq,
round(sum(quantity * price) over (partition by customer_id) / 1000, 2) as monetary,
3
4
5
6
7
8
9
10
11
                 Isast_value(to_date(invoicedate,mm/dd/yyyy hh24:mi'))

over (order by to_date(invoicedate,mm/dd/yyyy hh24:mi')) range between unbounded preceding and unbounded following) as 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') range between unbounded preceding and unbounded following) as first_val
                 select distinct(customer_id), freq, monetary, last_val, first_val, (monetary + freq) / 2 as average, ntile(5) over (order by freq desc) as freq_ntile,
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
                 ntile(5) over (order by monetary desc) as monetary_ntile,
ntile(5) over (order by (last_val - first_val) desc) as r_score
                from customer_summary
                       ct distinct(customer_id), freq, monetary, last_val, first_val, freq_ntile, monetary_ntile, r_score
                 ntile(5) over(order by average) as fm score
               select customer id.
               freq,
monetary,
                round((last val - first val), 2) as recency,
29
30
31
32
               r_score,
9: 5 Row 1 of 110 total rows HR@XE Modified
```

```
select customer_id,freq,monetary,
26
          round((last_val - first_val), 2) as recency,
27
28
          fm_score, r_score,
29
      case
30
      ፅ
             when r_score = 5 and fm_score in (5, 4) then 'champions'
31
32
             when r score = 4 and fm score = 5 then 'champions'
             when r_score = 5 and fm_score = 2 then 'potential loyalists'
33
             when r_score = 4 and fm_score in (2, 3) then 'potential loyalists'
34
35
             when r score = 3 and fm score = 3 then 'potential loyalists'
             when r_score = 5 and fm_score = 3 then 'loyal customers'
36
37
             when r_score = 4 and fm_score = 4 then 'loyal customers'
38
             when r_score = 3 and fm_score in (4, 5) then 'loyal customers'
             when r_score = 5 and fm_score = 1 then 'recent customers'
39
40
             when r_score = 4 and fm_score = 1 then 'promising'
41
             when r score = 3 and fm score = 1 then 'promising'
42
             when r_score = 3 and fm_score = 2 then 'customers needing attention'
43
             when r_score = 2 and fm_score in (2, 3) then 'customers needing attention'
44
             when r score = 1 and fm score = 3 then 'at risk'
45
             when r_score = 2 and fm_score in (4, 5) then 'at risk'
46
             when r_score = 1 and fm_score = 2 then 'hibernating'
47
             when r_score = 1 and fm_score in (4, 5) then 'cant lose them'
             when r_score = 1 and fm_score = 1 then 'lost'
48
49
             else 'undefined'
50
            end
51
52
         from
53
          score_with_fm
54
         order by
55
          customer_id;
CC.
```

The Output:

∄	CUSTOMER_ID	FREQ	MONETARY	RECENCY	FM_SCORE	R_SCORE	(CASEWHENR_SCORE=5ANDFM_SCOREIN(5,4)THEN'CHAMPIONS'W
þ	12747	11	4.2	1.91	5	5	champions
	12748	210	33.72	0	5	5	champions
	12749	5	4.09	3.1	5	5	champions
	12820	4	0.94	2.88	4	5	champions
	12821	1	0.09	213.85	1	1	lost
	12822	2	0.95	70.09	3	3	potential loyalists
	12823	5	1.76	74.2	4	2	at risk
	12824	1	0.4	58.98	2	3	customers needing attention
	12826	7	1.47	2.08	5	5	champions
	12827	3	0.43	5	3	5	loyal customers
	12828	6	1.02	2.15	4	5	champions
	12829	2	0.29	336.05	2	1	hibernating
	12830	6	6.81	37.02	5	3	loyal customers
	12831	1	0.22	261.97	1	1	lost
	12832	2	0.38	31.95	2	3	customers needing attention
	12833	1	0.42	144.94	2	2	customers needing attention
	12834	1	0.31	282.1	1	1	lost
	12836	4	2.61	58.88	4	3	loyal customers
	12837	1	0.13	172.84	1	2	undefined
12838 7 0.68 33 3 notential Invaliete							notantial Invalicte

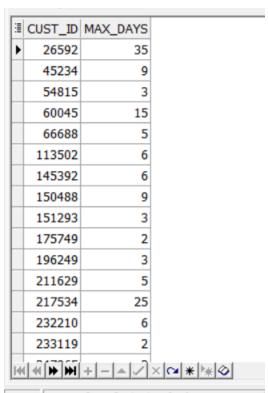
Explanation: This query segments customers based on their transaction frequency, monetary value, and recency of purchase.

Meaning: By analyzing customer behavior, such as how often they buy, how much they spend, and when they last made a purchase, the business can categorize customers into different segments. Each segment represents a different level of engagement and loyalty. This helps tailor marketing strategies and customer service efforts to better meet the needs of each segment, ultimately maximizing customer satisfaction and profitability.

Third Question:

Α-

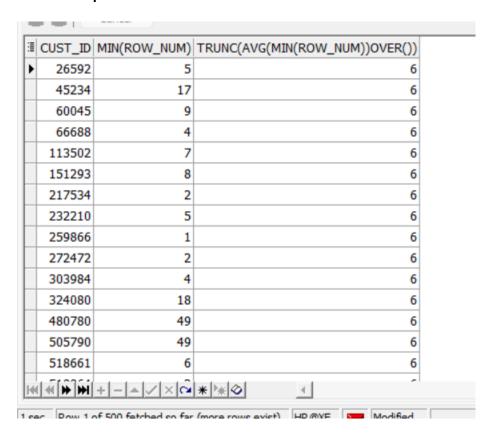
```
--third question a-
     p with purchase_days as (
     select cust_id,
          calendar_dt as purchase_date,
          row_number() over (partition by cust_id order by calendar_dt) as purchase_rank
         from daily_sales
        select cust_id,
         max(count_days) as max_days
     ig from (
0
         select cust_id,
          count(*) as count_days
2
         from purchase_days
3
         group by cust_id, purchase_date - purchase_rank
5
        group by cust_id
6
        order by cust_id;
```



- 1 sec Row 1 of 500 fetched so far (more rows exist)
 - This tells the longest streak of consecutive days a customer made purchases. It's important because it shows how regularly customers buy from the business.
 - It shows how loyal and engaged customers are with our brand.
 - Identifies customers at risk of leaving, allowing us to intervene and retain them.

B-

```
with CustomerTotalSpent as (
          select CUST_ID,
            CALENDAR_DT as InvoiceDate,
            sum(AMT_LE) over (partition by CUST_ID order by CALENDAR_DT rows between unbounded preceding and current row) AS TotalSpent
          from daily_sales
     RankedCustomers as (
          select CUST_ID,InvoiceDate,TotalSpent,
           row_number() over (partition by CUST_ID order by TotalSpent) AS Row_num
            CustomerTotalSpent
2
       select distinct CUST_ID,
         min(Row_num),trunc(avg( min(Row_num)) over())
       from RankedCustomers
       where TotalSpent >= 250
          group by cust_id
          order by cust_id
```



•	It helps identify customers who have spent at least \$250 and calculates the average ranking among them. This information provides insights into the typical spending habits of these customers.