Case Study - Retail Analytics - III

Challenge11:

Write a SQL query to understand the month on month growth rate of sales of the company which will help understand the growth trend of the company.

Hint:

- *Use the "sales_transaction" table.*
- *The resulting table must extract the month from the transactiondate and then the Month on month growth percentange should be calculated. (Total sales present month - total sales previous month/ total sales previous month * 100)*
- *Round all numerical answers to 2 decimal places*
- *Return the result table ordering by month.*

Output format:

month	total_sales	previous_month_sales	mom_growth_percentage
Month 1	DECINUM	NULL	NULL
Month 2	DECINUM	DECINUM	DECINUM
Month 3	DECINUM	DECINUM	DECINUM
Month 4	DECINUM	DECINUM	DECINUM
Month 5	DECINUM	DECINUM	DECINUM

```
WITH MonthSales AS (
    SELECT
        EXTRACT(MONTH FROM TransactionDate_updated) AS month,
        ROUND(SUM(QuantityPurchased * Price),2) AS total_sales
    FROM sales
    GROUP BY EXTRACT(MONTH FROM TransactionDate_updated)
)
SELECT
    month,
    total_sales,
    LAG(total_sales) OVER (ORDER BY month) AS previous_month_sales,
    ROUND(((total_sales - LAG(total_sales) OVER (ORDER BY month)) /
    LAG(total_sales) OVER (ORDER BY month)) * 100,2) AS mom_growth_percentage
FROM MonthSales
ORDER BY month;
```

month	total_sales	previous_month_sales	mom_growth_percentage
1	104289.18	NULL	NULL
2	96690.99	104289.18	-7.29
3	103271.49	96690.99	6.81
4	101561.09	103271.49	-1.66
5	102998.84	101561.09	1.42
6	102210.28	102998.84	-0.77
7	90981.75	102210.28	-10.99

Challenge12:

Write a SQL query that describes the number of transaction along with the total amount spent by each customer which are on the higher side and will help us understand the customers who are the high frequency purchase customers in the company.

Hint:

- *Use the "sales_transaction" table.*
- *The resulting table must have number of transactions more than 10 and TotalSpent more than 1000 on those transactions by the corresponding customers.*
- *Return the result table on the "TotalSpent" in descending order.*

Output format:

CustomerID	NumberOfTransactions	TotalSpent
Customer 1	NUM	DECINUM
Customer 2	NUM	DECINUM
Customer 3	NUM	DECINUM
Customer 4	NUM	DECINUM
Customer 5	NUM	DECINUM

```
-- Challenge12:
SELECT * FROM sales;

SELECT
    CustomerID,
    COUNT(TransactionID) AS  NumberOfTransactions,
    SUM(QuantityPurchased * Price) AS TotalSpent
FROM sales
GROUP BY CustomerID
HAVING NumberOfTransactions > 10 AND TotalSpent > 1000
ORDER BY TotalSpent DESC;
```

CustomerID	NumberOfTransactions	TotalSpent
936	12	2834.4700000000003
664	14	2519.04
670	12	2432.15
39	12	2221.29
958	12	2104.71
75	11	1862.7299999999998
476	11	1821.4399999999998
929	12	1798.42
881	11	1713.2300000000002
704	11	1628.34
648	11	1572.9999999999998
776	11	1551.0100000000002
99	12	1547.3599999999997
113	12	1525.46
613	11	1451.2700000000002
727	11	1415.6499999999999
676	11	1196.9699999999998
277	11	1163.3799999999999

Challenge13:

Write a SQL query that describes the number of transaction along with the total amount spent by each customer, which will help us understand the customers who are occasional customers or have low purchase frequency in the company.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table must have number of transactions less than or equal to 2 and corresponding total amount spent on those transactions by related customers.*
- *Return the result table of "NumberOfTransactions" in ascending order and "TotalSpent" in descending order.*

Output format:

```
+------------+----------------------+--------------------+
| CustomerID | NumberOfTransactions | TotalSpent         |
+------------+----------------------+--------------------+
| Customer 1 |          NUM         |       DECINUM      |
| Customer 2 |          NUM         |       DECINUM      |
| Customer 3 |          NUM         |       DECINUM      |
| Customer 4 |          NUM         |       DECINUM      |
| Customer 5 |          NUM         |       DECINUM      |
```

```sql
SELECT
    CustomerID,
    COUNT(*) AS  NumberOfTransactions,
    SUM(QuantityPurchased * Price) AS TotalSpent
FROM sales
GROUP BY CustomerID
HAVING NumberOfTransactions <= 2
ORDER BY NumberOfTransactions ASC, TotalSpent DESC;
```

CustomerID	NumberOfTransactions	TotalSpent
94	1	360.64
181	1	298.23
979	1	265.16
317	1	257.73
479	1	254.91
799	1	254.70000000000002
45	1	241.35000000000002
110	1	236.16
169	1	230.37
706	1	224.49
965	1	215.72
212	1	203.96999999999997
333	1	189
603	1	171.56
890	1	171.4

Challenge14:

Write a SQL query that describes the total number of purchases made by each customer against each productID to understand the repeat customers in the company.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table must have "CustomerID", "ProductID" and the number of times that particular customer have purchases the product.*
- *The number of times the customer has purchased should be more than once.*
- *Return the result table in descending order corresponding to the TimesPurchased column.*

Output format:

```
+------------+--------------+-----------------------+
| CustomerID | ProductID    | TimesPurchased        |
+------------+--------------+-----------------------+
| Customer 1 | Product 1    |          NUM          |
| Customer 2 | Product 2    |          NUM          |
| Customer 3 | Product 3    |          NUM          |
| Customer 4 | Product 4    |          NUM          |
| Customer 5 | Product 5    |          NUM          |
```

```
SELECT
    CustomerID,
    ProductID,
    COUNT(*) AS TimesPurchased
FROM sales
GROUP BY CustomerID, ProductID
HAVING TimesPurchased > 1
ORDER BY TimesPurchased DESC;
```

CustomerID	ProductID	TimesPurchased
685	192	3
467	181	2
215	13	2
492	74	2
242	172	2
822	165	2
296	196	2
613	44	2
225	75	2
710	156	2
2	65	2
852	66	2
49	179	2
626	189	2
986	116	2
39	57	2
444	170	2
69	59	2
254	190	2
336	16	2
664	75	2
99	10	2

Challenge15:

Problem statement Send feedback

Write a SQL query that describes the duration between the first and the last purchase of the customer in that particular company to understand the loyalty of the customer.

Hints:

- *Use the "Sales_transaction" table.*
- *The DATE column will be majorly in use in the question and the TransactionDate column in Sales_transaction is in text format. Thus, the format of the TransactionDate column should be changed.*
- *The resulting table must have the first date of purchase, the last date of purchase and the difference between the first and the last date of purchase.*
- *The difference between the first and the last date of purchase should be more than 0.*
- *Return the table in descending order corresponding to DaysBetweenPurchases.*

Output Format:

CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchases
Customer 1	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 2	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 3	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 4	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 5	YYYY-MM-DD	YYYY-MM-DD	NUM

Note: *The YYYY/MM/DD in the "FirstPurchase" and "LastPurchase" should be in this date format and the NUM in "DaysBetweenPurchases" is in integer format.*

```
-- Challenge 15 : Loyal Indicator

SELECT * FROM sales;

WITH transanctionDate AS (
    SELECT
        CustomerID,
    STR_TO_DATE(TransactionDate, '%d/%m/%y') AS TransactionDate
    FROM sales
)
SELECT
    CustomerID,
    MIN(TransactionDate) AS FirstPurchase,
    MAX(TransactionDate) AS LastPurchase,
    DATEDIFF(MAX(TransactionDate),MIN(TransactionDate)) AS DaysBetweenPurchases
FROM transanctionDate
GROUP BY CustomerID
HAVING DaysBetweenPurchases > 0
ORDER BY DaysBetweenPurchases DESC;
```

CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchases
215	2023-01-01	2023-07-28	208
414	2023-01-02	2023-07-26	205
664	2023-01-01	2023-07-24	204
701	2023-01-01	2023-07-23	203
277	2023-01-02	2023-07-24	203
22	2023-01-02	2023-07-24	203
976	2023-01-02	2023-07-24	203
647	2023-01-03	2023-07-25	203
162	2023-01-05	2023-07-27	203
806	2023-01-02	2023-07-23	202
511	2023-01-02	2023-07-23	202
703	2023-01-05	2023-07-26	202
188	2023-01-06	2023-07-27	202
380	2023-01-06	2023-07-27	202
566	2023-01-04	2023-07-24	201
748	2023-01-02	2023-07-21	200
687	2023-01-03	2023-07-22	200
943	2023-01-04	2023-07-23	200
65	2023-01-06	2023-07-25	200
648	2023-01-06	2023-07-25	200
476	2023-01-02	2023-07-20	199
958	2023-01-03	2023-07-21	199