Case Study - Retail Analytics - II

Challenge4:

Problem statement Send feedback

Write a SQL query to clean the DATE column in the dataset.

Steps:

- *Create a separate table and change the data type of the date column as it is in TEXT format and name it as you wish to.*
- *Remove the original table from the database.*
- *Change the name of the new table and replace it with the original name of the table.*

Hint:

- *Use the "Sales_transaction" tables.*
- *The resulting table will display a separate column named TransactionDate_updated.*

Output format:

TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price	TransactionDate_updated
Transaction1	Customer 1	Product 1	NUM	YYYY-MM-DD	DECINUM	YYYY-MM-DD
Transaction2	Customer 2	Product 2	NUM	YYYY-MM-DD	DECINUM	YYYY-MM-DD
Transaction3	Customer 3	Product 3	NUM	YYYY-MM-DD	DECINUM	YYYY-MM-DD
Transaction4	Customer 4	Product 4	NUM	YYYY-MM-DD	DECINUM	YYYY-MM-DD

Note: *The NUM in the output format denotes a numerical values, DECINUM denotes a decimal values, Transaction 1 denotes to transaction number 1, Customer 1 also means the customer with unique ID 1, TransactionDate is in text format thus, cannot be considered as date and the TransactionDate_updated is in Date format (YYYY-MM-DD)*

```
CREATE TABLE sales_updates AS
SELECT
    * ,
    STR_TO_DATE(TransactionDate , "%d/%m/%y") AS TransactionDate_updated
FROM sales;

SELECT * FROM sales_updates;

DROP TABLE sales;

ALTER TABLE sales_updates RENAME TO sales;
```

TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price	TransactionDate_updated
1	103	120	3	01/01/23	30.43	2023-01-01
2	436	126	1	01/01/23	15.19	2023-01-01
3	861	55	3	01/01/23	67.76	2023-01-01
4	271	27	2	01/01/23	65.77	2023-01-01
5	107	118	1	01/01/23	14.55	2023-01-01
6	72	53	1	01/01/23	26.27	2023-01-01
7	701	39	2	01/01/23	95.92	2023-01-01
8	21	65	4	01/01/23	17.19	2023-01-01
9	615	145	4	01/01/23	66	2023-01-01
10	122	158	2	01/01/23	22.27	2023-01-01
11	467	181	2	01/01/23	69	2023-01-01
12	215	13	3	01/01/23	18.78	2023-01-01
13	331	21	1	01/01/23	14.29	2023-01-01
14	459	147	3	01/01/23	53.98	2023-01-01
15	88	53	2	01/01/23	26.27	2023-01-01
16	373	19	3	01/01/23	96.33	2023-01-01

ProductID	QuantityPurchased	TransactionDate	Price	TransactionDate_updated
120	3	2023-01-01	30.43	2023-01-01
126	1	2023-01-01	15.19	2023-01-01
55	3	2023-01-01	67.76	2023-01-01
27	2	2023-01-01	65.77	2023-01-01
118	1	2023-01-01	14.55	2023-01-01
53	1	2023-01-01	26.27	2023-01-01
39	2	2023-01-01	95.92	2023-01-01
65	4	2023-01-01	17.19	2023-01-01
145	4	2023-01-01	66	2023-01-01
158	2	2023-01-01	22.27	2023-01-01

Console

```
-- Platform
CREATE TABLE sales_updates AS
SELECT
    * ,
    CAST(TransactionDate AS DATE) AS TransactionDate_updated
FROM sales;
```

Challenge5:

Total Sales Summary

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write a SQL query to summarize the total sales and quantities sold per product by the company.

(Here, the data has been already cleaned in the previous steps and from here we will be understanding the different types of data analysis from the given dataset.)

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table will display the total quantity purchased by the customers and the total sales done by the company to evaluate the product performance.*
- *Return the result table in descending order corresponding to Total Sales Column.*

Output format:

ProductID	TotalUnitsSold	TotalSales
Product 1	NUM	DECINUM
Product 2	NUM	DECINUM
Product 3	NUM	DECINUM
Product 4	NUM	DECINUM
Product 5	NUM	DECINUM

Note: *The NUM in the output format denotes a numerical value, Product 1 denotes any type of product corresponding to the ID from the dataset and DECINUM means a decimal value.*

```
SELECT
    ProductID,
    SUM(QuantityPurchased) AS TotalUnitsSold,
    SUM(QuantityPurchased * Price) AS TotalSales
FROM sales
GROUP BY ProductID
ORDER BY TotalSales DESC;
```

ProductID	TotalUnitsSold	TotalSales
17	100	9450
87	92	7817.24
179	86	7388.26
96	72	7132.32
54	86	7052.86
187	82	6915.88
156	76	6827.84
57	78	6622.2
200	69	6479.79
127	68	6415.8
28	69	6386.64
106	63	6262.83
104	72	6230.16
195	87	6229.2
103	66	6191.46
85	62	6188.22

Challenge6:

Write a SQL query to count the number of transactions per customer to understand purchase frequency.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table will be counting the number of transactions corresponding to each customerID.*
- *Return the result table ordered by NumberOfTransactions in descending order.*

Output format:

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

```
SELECT
    CustomerID,
    COUNT(*) AS NumberOfTransactions
FROM sales
GROUP BY CustomerID
ORDER BY NumberOfTransactions DESC;
```

CustomerID	NumberOfTransactions
664	14
958	12
99	12
113	12
929	12
936	12
670	12
39	12
277	11
476	11
776	11
727	11
648	11
613	11
268	11
881	11

Challenge7:

Problem statement Send feedback

Write a SQL query to evaluate the performance of the product categories based on the total sales which help us understand the product categories which needs to be promoted in the marketing campaigns.

Hint:

- *Use the "Sales_transaction" and "product_inventory" table.*
- *The resulting table must display product categories, the aggregated count of units sold for each category, and the total sales value per category.*
- *Return the result table ordering by TotalSales in descending order.*

Output format:

Category	TotalUnitsSold	TotalSales
Category 1	NUM	DECINUM
Category 2	NUM	DECINUM
Category 3	NUM	DECINUM
Category 4	NUM	DECINUM

```
-- Challenge 7:
SELECT * FROM sales;
SELECT * FROM product;

SELECT
    p.Category,
    SUM(s.QuantityPurchased) AS TotalUnitsSold,
    SUM(s.QuantityPurchased * s.Price) AS TotalSales
FROM sales s
JOIN product p
ON s.ProductID = p.ProductID
GROUP BY Category
ORDER BY TotalSales DESC;
```

Category	TotalUnitsSold	TotalSales
Home & Kitchen	3477	217755.94000000026
Electronics	3037	177548.48000000007
Clothing	2810	162874.21000000005
Beauty & Health	3001	143824.98999999947

Challenge8:

Write a SQL query to find the top 10 products with the highest total sales revenue from the sales transactions. This will help the company to identify the High sales products which needs to be focused to increase the revenue of the company.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table should be limited to 10 productIDs whose TotalRevenue (Product of Price and QuantityPurchased) is the highest.*
- *Return the result table ordering by TotalRevenue in descending order.*

Output format:

```
+-----------+--------------------+
| ProductID |     TotalRevenue   |
+-----------+--------------------+
| Product 1 |         DECINUM    |
| Product 2 |         DECINUM    |
| Product 3 |         DECINUM    |
| Product 4 |         DECINUM    |
+-----------+--------------------+
```

```sql
SELECT
    ProductID,
    SUM(QuantityPurchased * Price) AS TotalRevenue
FROM sales
GROUP BY ProductID
ORDER BY TotalRevenue DESC
LIMIT 10;
```

ProductID	TotalRevenue
17	9450
87	7817.239999999998
179	7388.259999999998
96	7132.3200000000015
54	7052.8600000000015
187	6915.880000000003
156	6827.840000000002
57	6622.199999999999
200	6479.790000000001
127	6415.799999999999

Challenge9:

bottom 10 Values

Write a SQL query to find the ten products with the least amount of units sold from the sales transactions provided that at least one unit was sold for those products.

having > 0

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table should be limited to 10 productIDs whose TotalUnitsSold (sum of QuantityPurchased) is the least. (The limit value can be adjusted accordingly)*
- *Return the result table ordering by TotalUnitsSold in ascending order.*

Output format:

ProductID	TotalUnitsSold
Product 1	NUM
Product 2	NUM
Product 3	NUM
Product 4	NUM
Product 5	NUM

```
-- Challenge 9:
SELECT * FROM sales;
SELECT
    ProductID,
    SUM(QuantityPurchased) AS TotalUnitsSold
FROM sales
GROUP BY ProductID
HAVING TotalUnitsSold > 0
ORDER BY TotalUnitsSold
LIMIT 10;
```

ProductID	TotalUnitsSold
142	27
33	31
174	33
159	35
60	35
41	35
91	35
198	36
124	39
163	39

Challenge10:

Write a SQL query to identify the sales trend to understand the revenue pattern of the company.

Hint:

- *Use the "sales_transaction" table.*
- *The resulting table must have DATETRANS in date format, count the number of transaction on that particular date, total units sold and the total sales took place.*
- *Return the result table ordered by datetrans in descending order.*

Output format:

DATETRANS	Transaction_count	TotalUnitsSold	TotalSales
YYYY-MM-DD	NUM	NUM	DECINUM
YYYY-MM-DD	NUM	NUM	DECINUM
YYYY-MM-DD	NUM	NUM	DECINUM
YYYY-MM-DD	NUM	NUM	DECINUM
YYYY-MM-DD	NUM	NUM	DECINUM

```
-- Challenge 10:
SELECT * FROM sales;

SELECT
    TransactionDate_updated AS DATETRANS,
    COUNT(*) AS Transaction_count,
    SUM(QuantityPurchased) AS TotalUnitsSold,
    SUM(QuantityPurchased * Price) AS TotalSales
FROM sales
GROUP BY DATETRANS
ORDER BY DATETRANS DESC;
```

DATETRANS	Transaction_count	TotalUnitsSold	TotalSales
2023-07-28	8	18	1158.8600000000001
2023-07-27	24	58	3065.8099999999995
2023-07-26	24	58	3168.0400000000004
2023-07-25	24	54	2734.26
2023-07-24	24	63	3691.079999999999
2023-07-23	24	57	3578.5800000000004
2023-07-22	24	62	3350.8
2023-07-21	24	61	3443.72
2023-07-20	24	60	3216.57
2023-07-19	24	52	2068.5000000000005
2023-07-18	24	57	3251.0699999999997
2023-07-17	24	56	3051.5099999999998
2023-07-16	24	66	3145.46
2023-07-15	24	64	4231.93