Case Study - Retail Analytics - II

Challenge- 2

Problem statement Send feedback

Write a query to identify the discrepancies in the price of the same product in "sales_transaction" and "product_inventory" tables. Also, update those discrepancies to match the price in both the tables.

Hint:

- *Use the "sales_transaction" and the "product_inventory" tables.*
- *There will be two resulting tables in the output. First, the table where the discrepancies will be identified and in the second table we can check if the discrepancies were updated or not.*

Output format:

```
+---------------+------------------+----------------+
| TransactionID | TransactionPrice | InventoryPrice |
+---------------+------------------+----------------+
| Transaction 1 |        NUM       |       DECINUM  |
| Transaction 2 |        NUM       |       DECINUM  |
| Transaction 3 |        NUM       |       DECINUM  |
| Transaction 4 |        NUM       |       DECINUM  |
+---------------+------------------+----------------+

+---------------+------------+-----------+--------------------+-----------------+-------+
| TransactionID | CustomerID | ProductID | QuantityPurchased  | TransactionDate | Price |
+---------------+------------+-----------+--------------------+-----------------+-------+
| Transaction 1| Customer 1 | Product 1 |        NUM         |       TEXT      |DECINUM|
| Transaction 2| Customer 2 | Product 2 |        NUM         |       TEXT      |DECINUM|
| Transaction 3| Customer 3 | Product 3 |        NUM         |       TEXT      |DECINUM|
| Transaction 4| Customer 4 | Product 4 |        NUM         |       TEXT      |DECINUM|
| Transaction 5| Customer 5 | Product 5 |        NUM         |       TEXT      |DECINUM|
+---------------+------------+-----------+--------------------+-----------------+-------+
```

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 and TransactionDate is in text format thus, cannot be considered as date.*

```
60 •  SELECT * FROM Products;
```

Result Grid | Filter Rows: | Export:

ProductID	ProductName	Category	StockLevel	Price
1	Product_1	Clothing	22	46.11
2	Product_2	Home & Kitchen	140	81.6
3	Product_3	Home & Kitchen	473	78.72
4	Product_4	Clothing	386	22.06
5	Product_5	Beauty & Health	284	17.97
6	Product_6	Home & Kitchen	449	91.73
7	Product_7	Home & Kitchen	319	58.2
8	Product_8	Home & Kitchen	155	87.2
9	Product_9	Clothing	470	15.23
10	Product_10	Electronics	419	57.39
11	Product_11	Electronics	112	58.55
12	Product_12	Electronics	389	87.46
13	Product_13	Electronics	138	18.78
14	Product_14	Electronics	421	31.64
15	Product_15	Home & Kitchen	373	46.59
16	Product_16	Home & Kitchen	417	87.93

```
61 •  SELECT * FROM Sales;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

```sql
SELECT
    p.ProductID,
    TransactionID,
    s.Price AS TransactionPrice,
    p.Price AS InventoryPrice
FROM sales s
JOIN products p
ON p.ProductID = s.ProductID
WHERE s.Price <> p.Price;
```

ProductID	TransactionID	TransactionPrice	InventoryPrice
51	88	9312	93.12
51	236	9312	93.12
51	591	9312	93.12
51	1377	9312	93.12
51	1910	9312	93.12
51	2608	9312	93.12
51	2939	9312	93.12
51	3377	9312	93.12
51	3635	9312	93.12
51	3839	9312	93.12
51	3918	9312	93.12
51	3959	9312	93.12
51	3962	9312	93.12
51	4148	9312	93.12
51	4158	9312	93.12
51	4221	9312	93.12
51	4408	9312	93.12
51	4532	9312	93.12

What if, we have to handle the multiple productID for fixing the incorrect price.

```sql
UPDATE Sales
SET Price = 93.12 --
WHERE ProductID = 51;
```

Dynamic Value
IN (51,27,33,55,44)

```sql
-- Generalized Solution .......
SET SQL_SAFE_UPDATES = 0;
UPDATE sales s
SET Price = (
    SELECT p.price FROM Products p
    WHERE s.ProductID = p.ProductID
)
WHERE ProductID IN (
    SELECT ProductID FROM Products p
    WHERE p.Price <> s.Price
);

SELECT * FROM Sales WHERE ProductID = 51;
-- WHERE ProductID IN (x1,x2,x3,x4......)
```

	ProductID	TransactionID	TransactionPrice	InventoryPrice

TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price
88	562	51	2	04/01/23	93.12
236	231	51	2	10/01/23	93.12
591	820	51	2	25/01/23	93.12
1377	172	51	4	27/02/23	93.12
1910	482	51	3	21/03/23	93.12
2608	950	51	1	19/04/23	93.12
2939	944	51	2	03/05/23	93.12
3377	422	51	3	21/05/23	93.12
3635	534	51	4	01/06/23	93.12

Challenge- 3

</> Fixing Null Values

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

Problem statement Send feedback

Write a SQL query to identify the null values in the dataset and replace those by "Unknown".

Hint:

- *Use the customer_profiles table.*
- *Identify the columns which contains null values and count the number of cells containing null values. Update those values with "unknown" and showcase the changes that the query has created.*

Output format:

```
+----------+
| count(*) |
+----------+
|   NUM    |
+----------+
```

```
+-------------+------+--------+----------+----------+
| CustomerID  | Age  | Gender | Location | JoinDate |
+-------------+------+--------+----------+----------+
| Customer 1  |  NUM | Other  | East     | DD/MM/YY |
| Customer 2  |  NUM | Male   | North    | DD/MM/YY |
| Customer 3  |  NUM | Other  | North    | DD/MM/YY |
| Customer 4  |  NUM | Other  | Unknown  | DD/MM/YY |
| Customer 5  |  NUM | Male   | North    | DD/MM/YY |
+-------------+------+--------+----------+----------+
```

Note: *NUM in the output format denotes a numerical value and joinDate is in Date format (DD/MM/YY)*

IS NULL
LIKE = " ";

```
-- Challenge 3 - Fixing Null Values :

SELECT * FROM Customers;
SELECT * FROM Customers WHERE Location LIKE "";
SELECT COUNT(*) FROM Customers WHERE Location IS NULL;
SELECT COUNT(*) FROM Customers WHERE Location LIKE "";

-- Update the Location Column with Unknown
UPDATE Customers
SET Location = "Unknown"
WHERE Location LIKE "";
-- 13 row(s) affected Rows matched: 13  Changed: 13  Warnings: 0

UPDATE Customers
SET Location = "Unknown"
WHERE Location IS NULL;
```

CustomerID	Age	Gender	Location	JoinDate
1	63	Other	East	01/01/20
2	63	Male	North	02/01/20
3	34	Other	North	03/01/20
4	19	Other	Unknown	04/01/20
5	57	Male	North	05/01/20
6	22	Other	South	06/01/20
7	56	Other	East	07/01/20
8	65	Female	East	08/01/20
9	33	Male	West	09/01/20
10	34	Male	East	10/01/20
11	44	Other	North	11/01/20

Challenge- 4

</> Cleaning Date

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

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)*

Field	Type	Null	Key	Default	Extra
TransactionID	int	YES		NULL	
CustomerID	int	YES		NULL	
ProductID	int	YES		NULL	
QuantityPurcha...	int	YES		NULL	
TransactionDate	text	YES		NULL	
Price	double	YES		NULL	

```
122 •  SELECT * FROM Sales_Updates;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

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

```
-- Challenge 4 : Cleaning Date
SELECT * FROM Sales;
DESC Sales;

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;
```

sales_transactions

As_per_Platform

```
CREATE TABLE sales_updates AS
SELECT
    TransactionID,
    CustomerID,
    ProductID,
    QuantityPurchased,
    TransactionDate,
    Price,
    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;
```

Challenge- 5

</> 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.*

```sql
-- Challenge 5 : Total Sales Summary

SELECT * FROM Sales;

SELECT
    ProductID,
    SUM(QuantityPurchased) AS TotalUnitsSold,
    ROUND(SUM(QuantityPurchased * Price),2) 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

Challenge- 6

Customer Purchase Frequency

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

Problem statement Send feedback

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          |
+----------------+--------------------------+
```

Note: *The NUM in the output format denotes a numerical value and Customer 1 denote the customer ID in the sales_transaction table.*

```sql
-- Challenge 6 : Customer Purchase Frequency ....

SELECT * FROM Sales;

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
659	11
704	11
75	11

Challenge- 7

Product Categories Performance

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

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

Note: *The NUM in the output format denotes a numerical value, Category 1 denotes any type of category from the dataset and DECINUM means a decimal value.*

```sql
-- Challenge 7 : Product Categories Performance

SELECT * FROM Sales;
SELECT * FROM Products;

SELECT
    p.Category,
    SUM(s.QuantityPurchased) AS TotalUnitsSold,
    ROUND(SUM(s.QuantityPurchased * s.Price),2) AS TotalSales
FROM Products p
JOIN Sales s
ON p.ProductID = s.ProductID
GROUP BY p.Category
ORDER BY TotalSales DESC;
```

Category	TotalUnitsSold	TotalSales
Home & Kitchen	3477	217755.94
Electronics	3037	177548.48
Clothing	2810	162874.21
Beauty & Health	3001	143824.99

Challenge- 8

High Sales Products

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

Problem statement Send feedback

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     |
+-----------+--------------------+
```

Note: *The DECINUM in the output format denotes a decimal number.*

```sql
-- Challenge 8 : High Sales Products

SELECT * FROM Sales;

SELECT
    ProductID,
    ROUND(SUM(QuantityPurchased * Price),2) AS TotalRevenue
FROM Sales
GROUP BY ProductID
ORDER BY TotalRevenue DESC
LIMIT 10;
```

ProductID	TotalRevenue
17	9450
87	7817.24
179	7388.26
96	7132.32
54	7052.86
187	6915.88
156	6827.84
57	6622.2
200	6479.79
127	6415.8

Challenge- 9

Low Sales Products

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

Problem statement Send feedback

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.

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

Note: *NUM in the output format denotes a numerical value.*

```sql
-- Challenge 9 : Low Sales Products

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

Challenge- 10

Sales Trend

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

Problem statement Send feedback

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

Note: *The NUM in the output format denotes number and DECINUM denotes decimal number*

```sql
-- Challenge 10 : Sales Trend
SELECT * FROM Sales;

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

DATETRANS	Transaction_count	TotalUnitsSold	TotalSales
2023-07-28	8	18	1159
2023-07-27	24	58	3066
2023-07-26	24	58	3168
2023-07-25	24	54	2734
2023-07-24	24	63	3691
2023-07-23	24	57	3579
2023-07-22	24	62	3351
2023-07-21	24	61	3444
2023-07-20	24	60	3217
2023-07-19	24	52	2069
2023-07-18	24	57	3251
2023-07-17	24	56	3052
2023-07-16	24	66	3145
2023-07-15	24	64	4232
2023-07-14	24	52	2939
2023-07-13	24	61	3290
2023-07-12	24	63	3668
2023-07-11	24	58	3246

Challenge- 11

Growth Rate of Sales

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

Problem statement Send feedback

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:

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

Note: *NUM in the output format denotes a numerical value.*

```
WITH MonthlySales AS (
    SELECT
        EXTRACT(MONTH FROM TransactionDate_updated) AS month,
        ROUND(SUM(QuantityPurchased * Price) , 2) AS total_sales
    FROM Sales
    GROUP BY 1
)
SELECT * FROM MonthlySales;
```

month	total_sales
1	104289.18
2	96690.99
3	103271.49
4	101561.09
5	102998.84
6	102210.28
7	90981.75

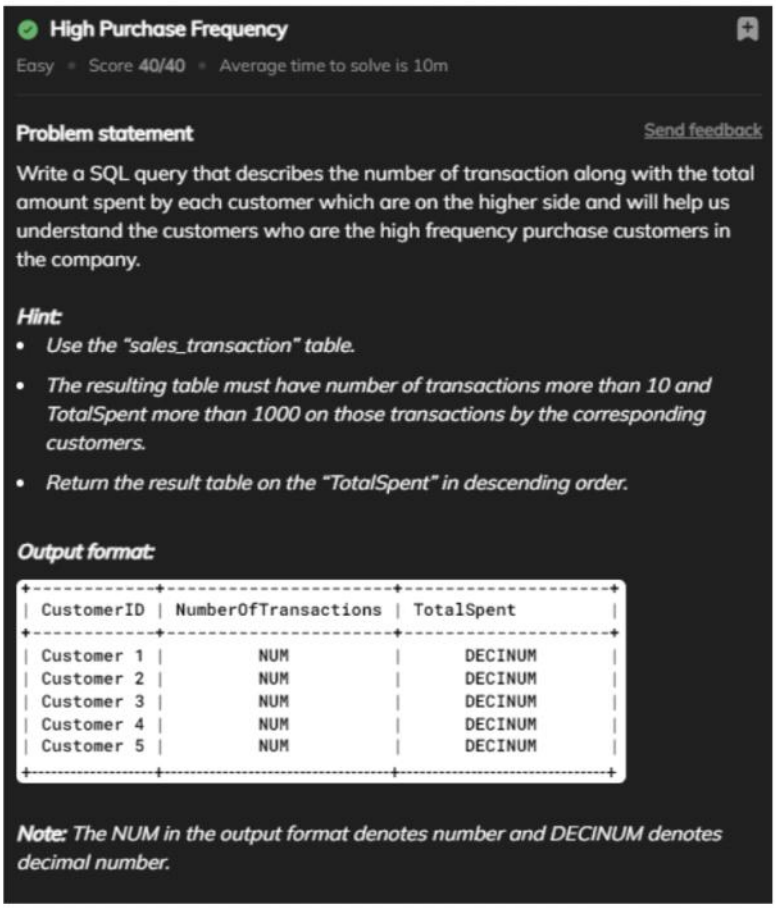
```
-- Challenge 11 : Growth Rate of Sales

SELECT * FROM Sales;

WITH MonthlySales AS (
    SELECT
        EXTRACT(MONTH FROM TransactionDate_updated) AS month,
        ROUND(SUM(QuantityPurchased * Price) , 2) AS total_sales
    FROM Sales
    GROUP BY 1
)
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 MonthlySales
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

Challenge- 12



High Purchase Frequency

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

Problem statement Send feedback

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

Note: *The NUM in the output format denotes number and DECINUM denotes decimal number.*

```sql
-- Challenge 12 : High Purchase Frequency
SELECT * FROM Sales;

SELECT
    CustomerID,
    COUNT(TransactionID) AS NumberOfTransactions,
    ROUND(SUM(QuantityPurchased * Price) , 2) As TotalSpent
FROM Sales
GROUP BY CustomerID
HAVING NumberOfTransactions > 10 AND TotalSpent > 1000
ORDER BY TotalSpent DESC;
```

CustomerID	NumberOfTransactions	TotalSpent
936	12	2834.47
664	14	2519.04
670	12	2432.15
39	12	2221.29
958	12	2104.71
75	11	1862.73
476	11	1821.44
929	12	1798.42
881	11	1713.23
704	11	1628.34
648	11	1573
776	11	1551.01
99	12	1547.36

648	11	1573
776	11	1551.01
99	12	1547.36
113	12	1525.46
613	11	1451.27
727	11	1415.65
676	11	1196.97
277	11	1163.38