SQL Project

Retail Analytics Case Study

SACHIN |Data Analyst

Overview

In the rapidly evolving retail sector, businesses continually seek innovative strategies to stay ahead of the competition, improve customer satisfaction, and optimize operational efficiency. Leveraging data analytics has become a cornerstone for achieving these objectives. This case study focuses on a retail company that has encountered challenges in understanding its sales performance, customer engagement, and inventory management. Through a comprehensive data analysis approach, the company aims to identify high or low sales products, effectively segment its customer base, and analyze customer behavior to enhance marketing strategies, inventory decisions, and overall customer experience.

Business Problems

The retail company has observed stagnant growth and declining customer engagement metrics over the past quarters. Initial assessments indicate potential issues in product performance variability, ineffective customer segmentation, and lack of insights into customer purchasing behavior. The company seeks to leverage its sales transaction data, customer profiles, and product inventory information to address the following key business problems.

Product Performance Variability:

Identifying which products are performing well in terms of sales and which are not. This insight is crucial for inventory management and marketing focus.

Customer Segmentation:

The company lacks a clear understanding of its customer base segmentation. Effective segmentation is essential for targeted marketing and enhancing customer satisfaction.

Customer Behaviour Analysis:

Understanding patterns in customer behavior, including repeat purchases and loyalty indicators, is critical for tailoring customer engagement strategies and improving retention rates.

OBJECTIVES

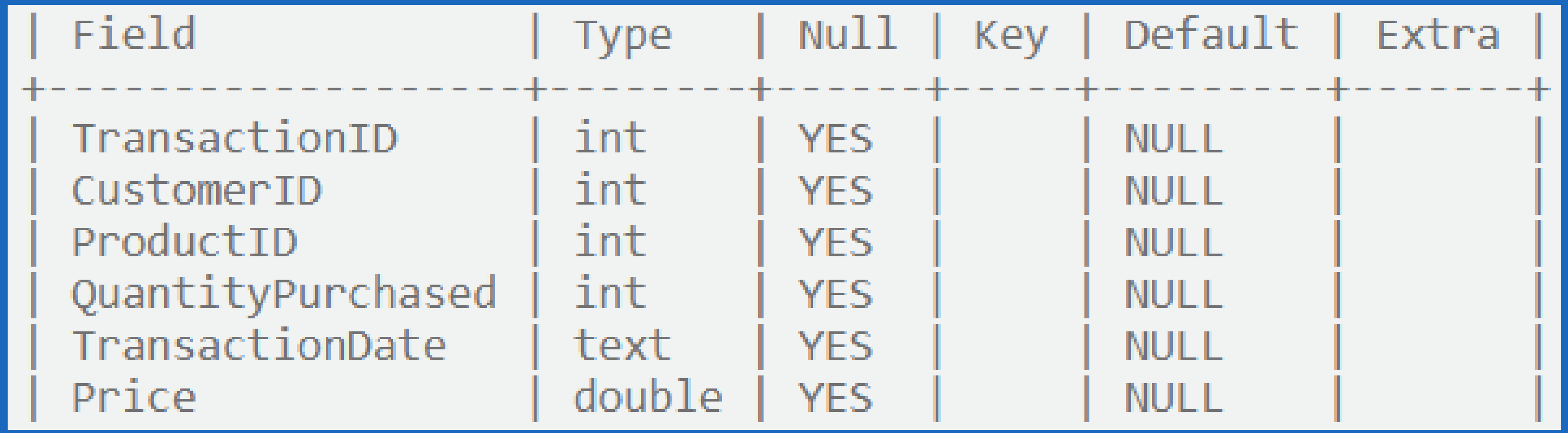
- To utilize SQL queries for data cleaning and exploratory data analysis to ensure data quality and gain initial insights.
- To identify high and low sales products to optimize inventory and tailor marketing efforts.
- To segment customers based on their purchasing behavior for targeted marketing campaigns. Create Customer segments -

Total Quantity of Products Purchased	Customer Segment
0	No Orders
1-10	Low
10-30	Mid
>30	High Value

- To analyze customer behavior for insights on repeat purchases and loyalty. informing customer retention strategies.

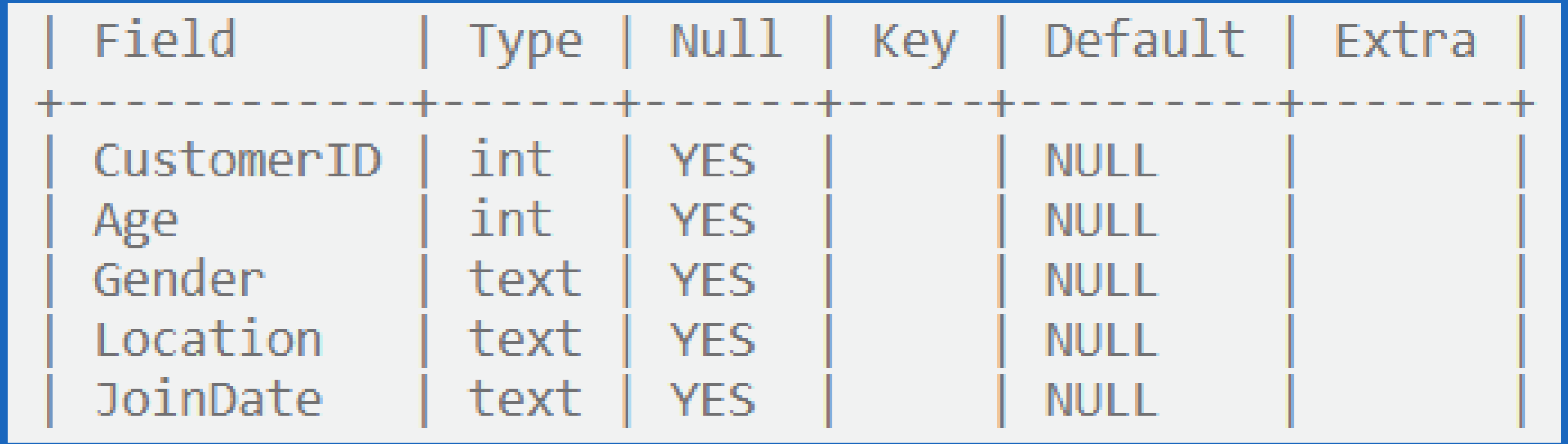
TABLES USED

sales_transaction



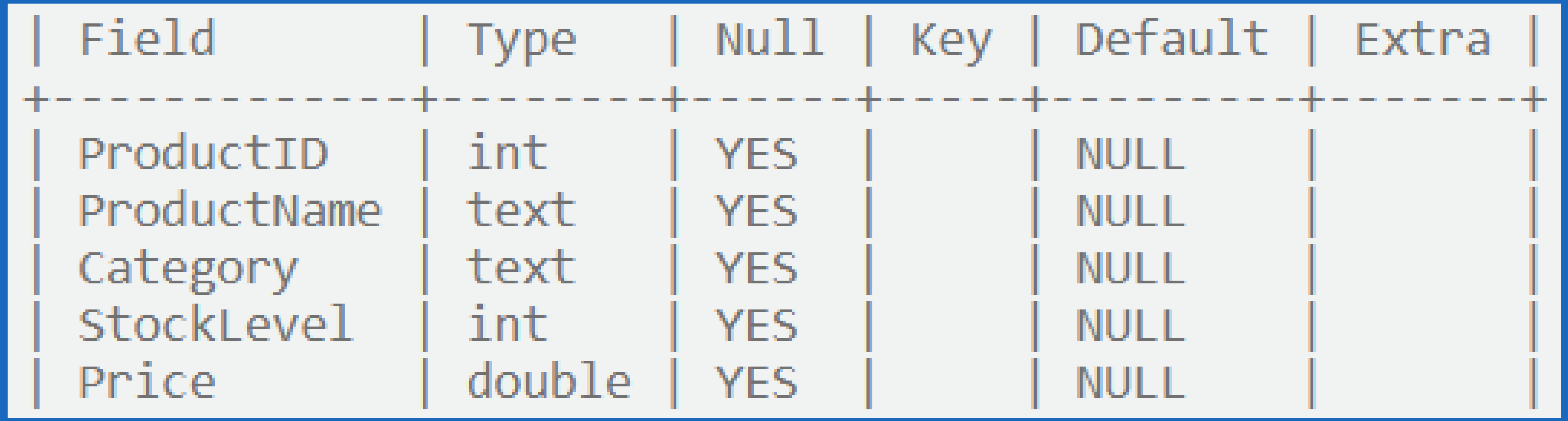
```
| Field             | Type   | Null | Key | Default | Extra |
+-------------------+--------+------+-----+---------+-------+
| TransactionID     | int    | YES  |     | NULL    |       |
| CustomerID        | int    | YES  |     | NULL    |       |
| ProductID         | int    | YES  |     | NULL    |       |
| QuantityPurchased | int    | YES  |     | NULL    |       |
| TransactionDate   | text   | YES  |     | NULL    |       |
| Price             | double | YES  |     | NULL    |       |
```

customer_profiles



```
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| CustomerID | int  | YES  |     | NULL    |       |
| Age        | int  | YES  |     | NULL    |       |
| Gender     | text | YES  |     | NULL    |       |
| Location   | text | YES  |     | NULL    |       |
| JoinDate   | text | YES  |     | NULL    |       |
```

Product_Inventory



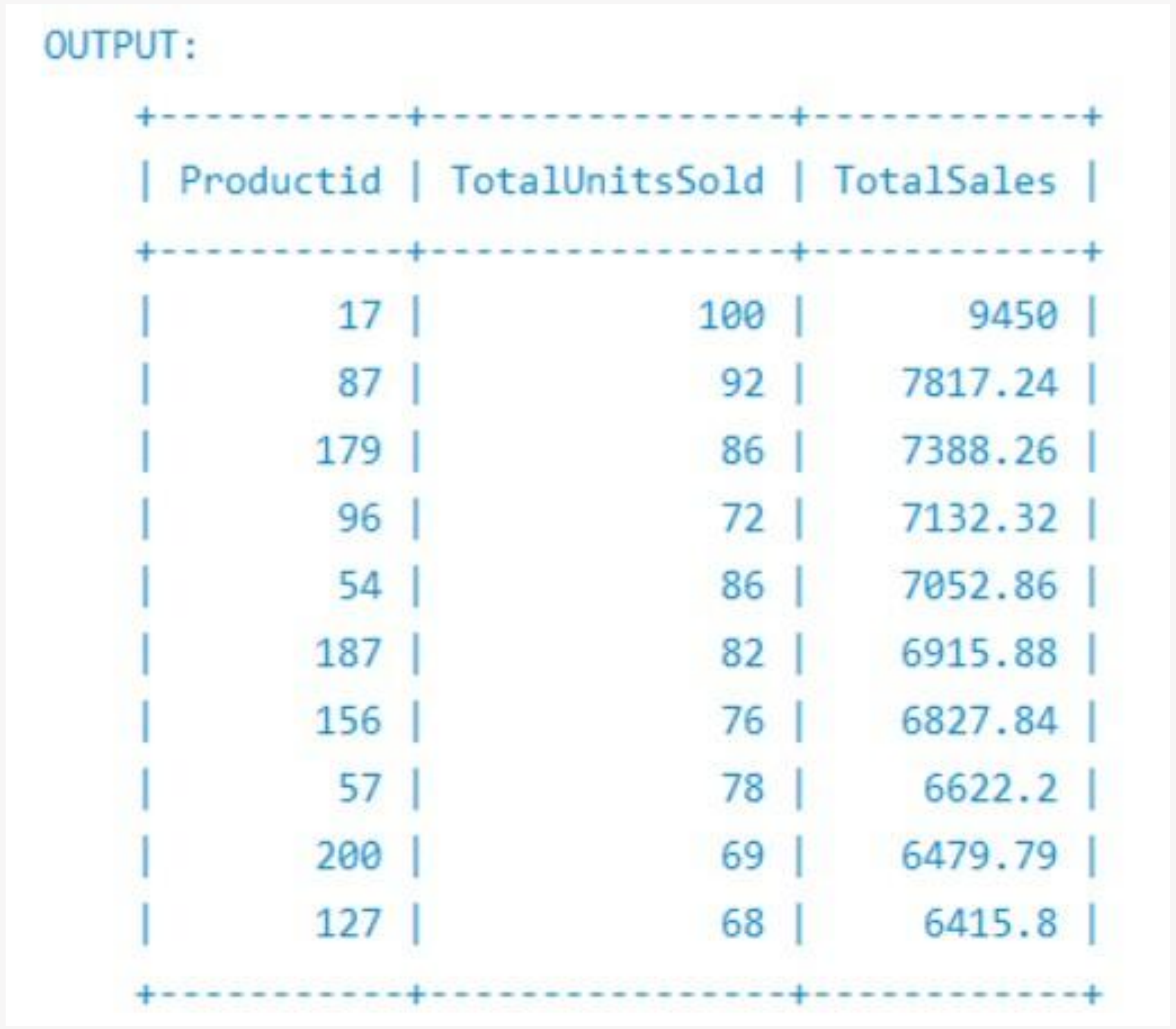
```
| Field       | Type   | Null | Key | Default | Extra |
+-------------+--------+------+-----+---------+-------+
| ProductID   | int    | YES  |     | NULL    |       |
| ProductName | text   | YES  |     | NULL    |       |
| Category    | text   | YES  |     | NULL    |       |
| StockLevel  | int    | YES  |     | NULL    |       |
| Price       | double | YES  |     | NULL    |       |
```

Total Sales Summary

Problem : Summarize the total sales and quantities sold per product by the company.

Query:

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



```
OUTPUT:

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

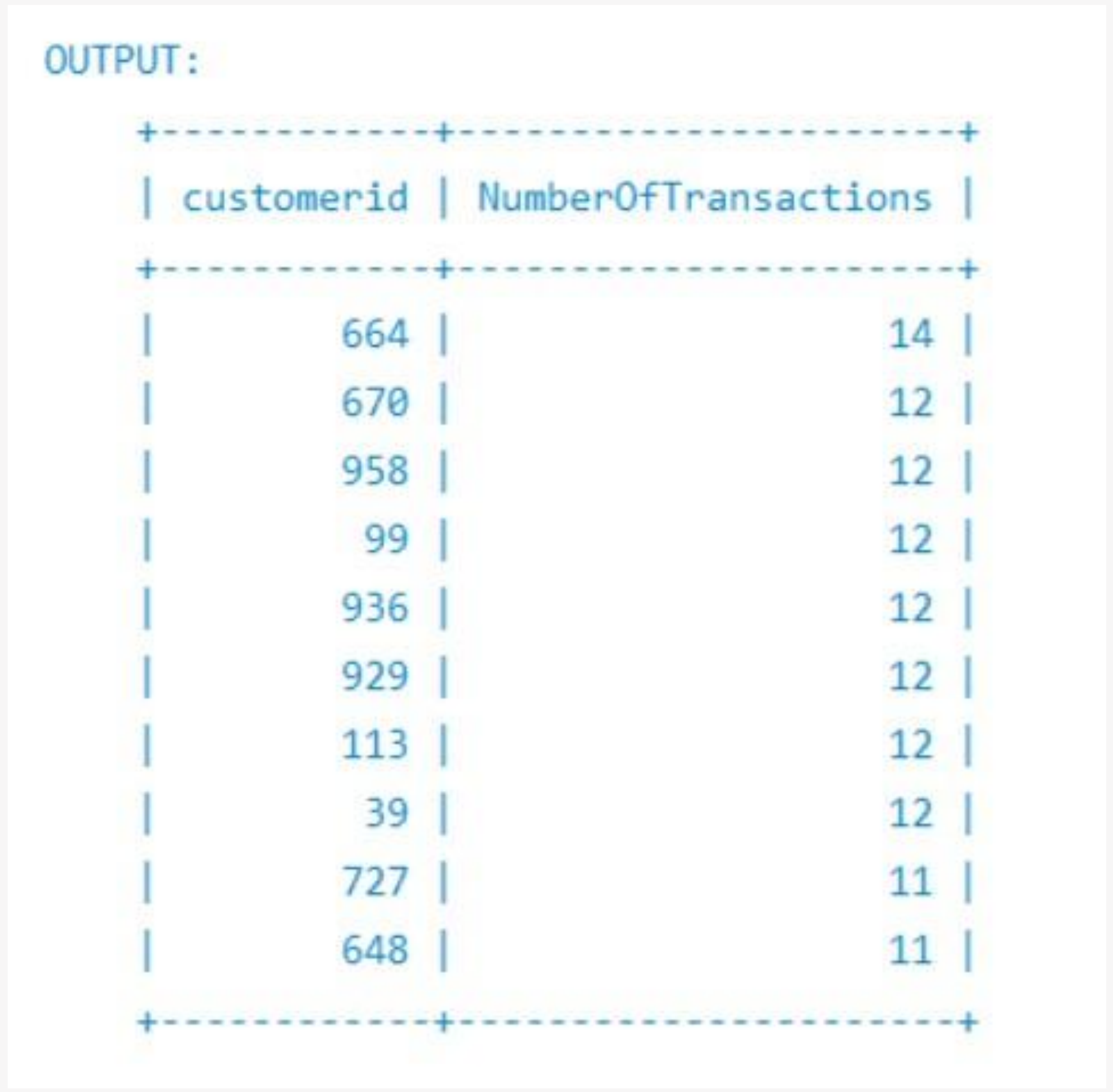
Insight: Products with the highest revenue are Product IDs 17, 87, and 179.

Customer Purchase Frequency

Problem : Count the number of transactions per customer to understand purchase frequency.

Query:

```
SELECT
   CustomerID,
   COUNT(TransactionID) AS NumberOfTransactions
FROM Sales_transaction
GROUP BY CustomerID
ORDER BY NumberOfTransactions DESC
LIMIT 10;
```



```
OUTPUT:

+------------+----------------------+
| customerid | NumberOfTransactions |
+------------+----------------------+
|        664 |                   14 |
|        670 |                   12 |
|        958 |                   12 |
|         99 |                   12 |
|        936 |                   12 |
|        929 |                   12 |
|        113 |                   12 |
|         39 |                   12 |
|        727 |                   11 |
|        648 |                   11 |
+------------+----------------------+
```

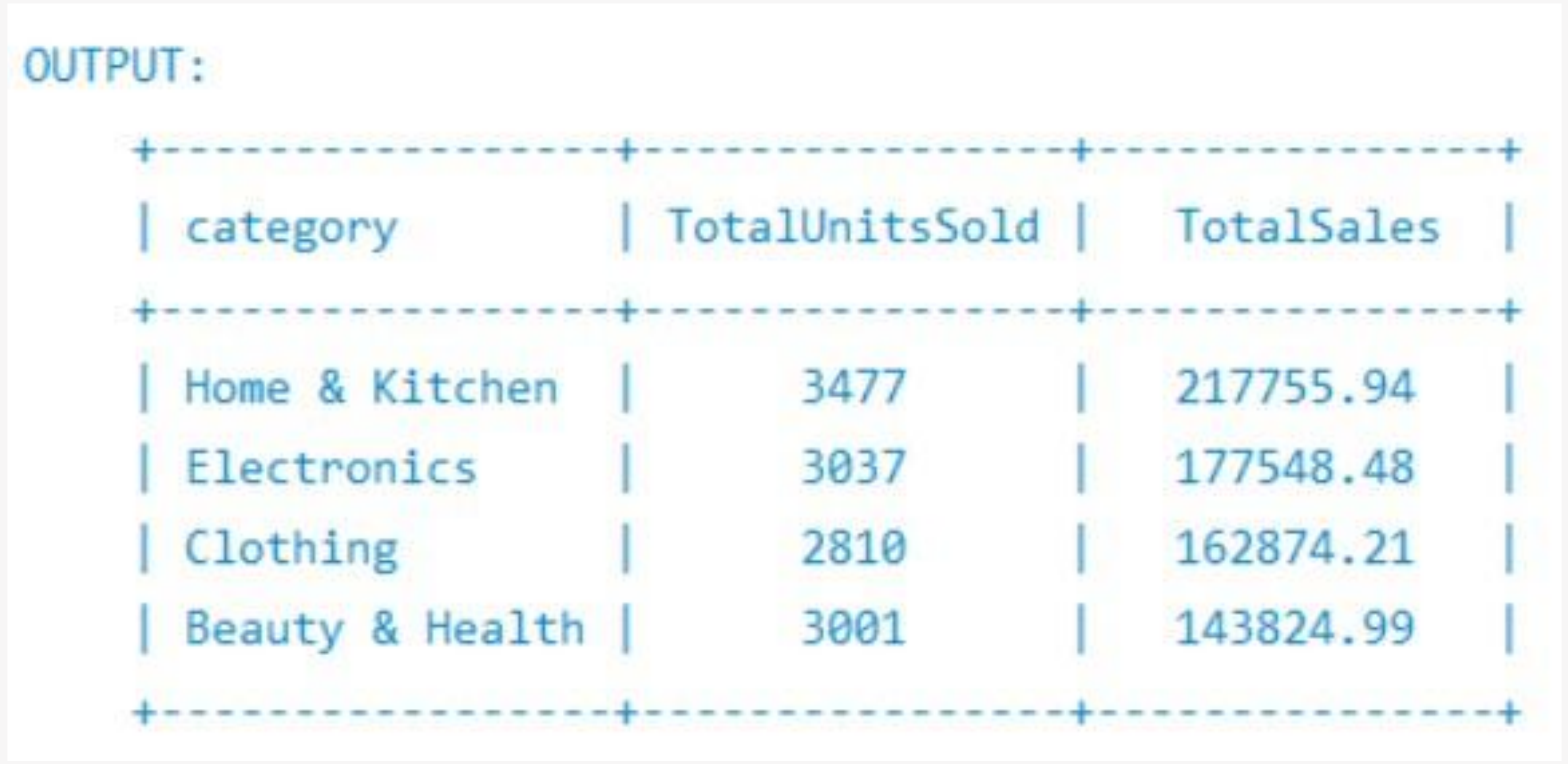
Insight: Customers such as 664, 670, and 958 made 12-14 purchases.

Product Categories Performance

Problem : Evaluate product categories based on total sales to guide marketing strategies.

Query:

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



```
OUTPUT:

+-----------------+---------------+--------------+
| category        | TotalUnitsSold |   TotalSales  |
+-----------------+---------------+--------------+
| Home & Kitchen  |     3477      |  217755.94   |
| Electronics     |     3037      |  177548.48   |
| Clothing        |     2810      |  162874.21   |
| Beauty & Health |     3001      |  143824.99   |
+-----------------+---------------+--------------+
```

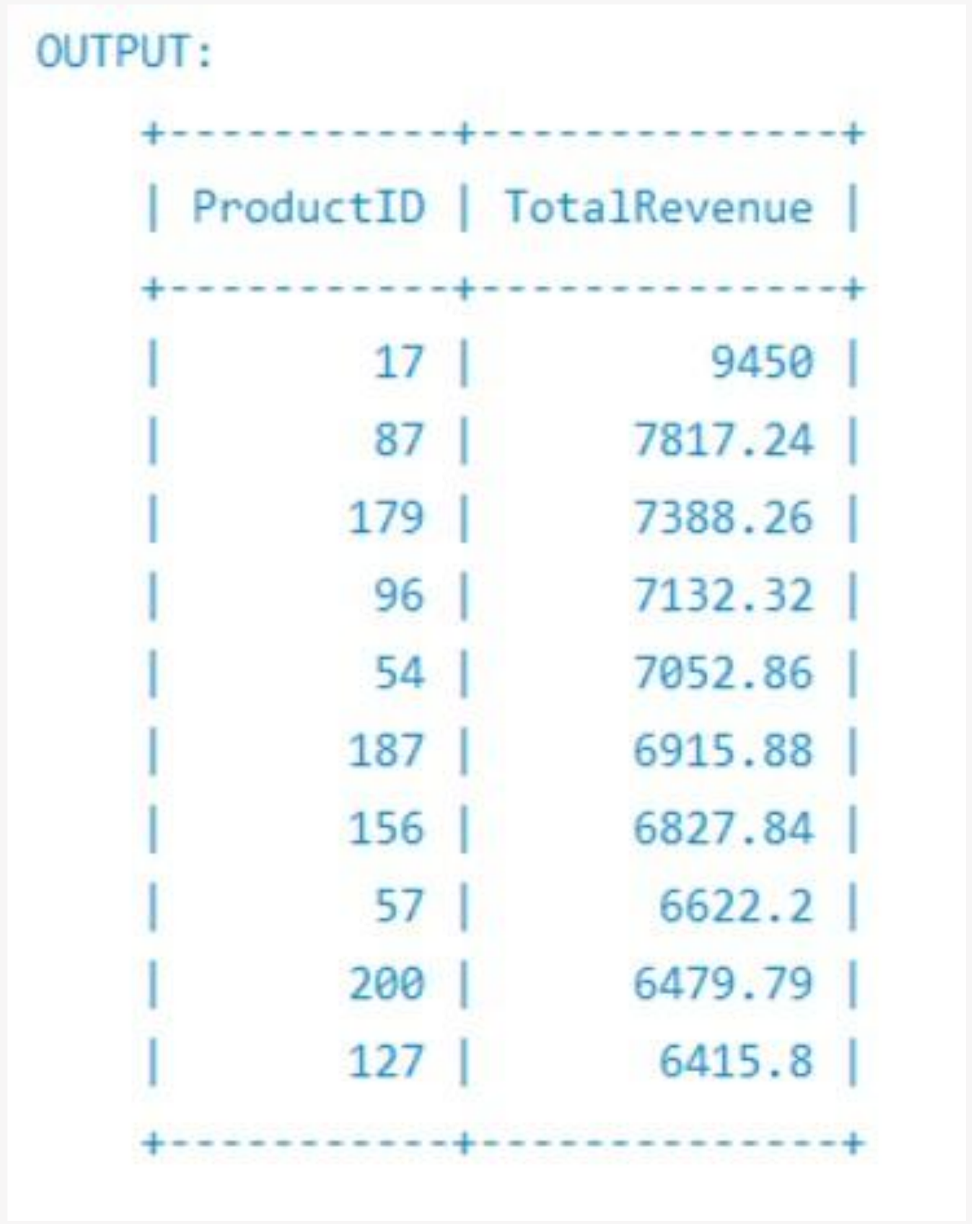
Insight: 'Home & Kitchen' is top-performing category.

High Sales Products

Problem : Identify the top 10 products with the highest total sales revenue.

Query:

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



```
OUTPUT:

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

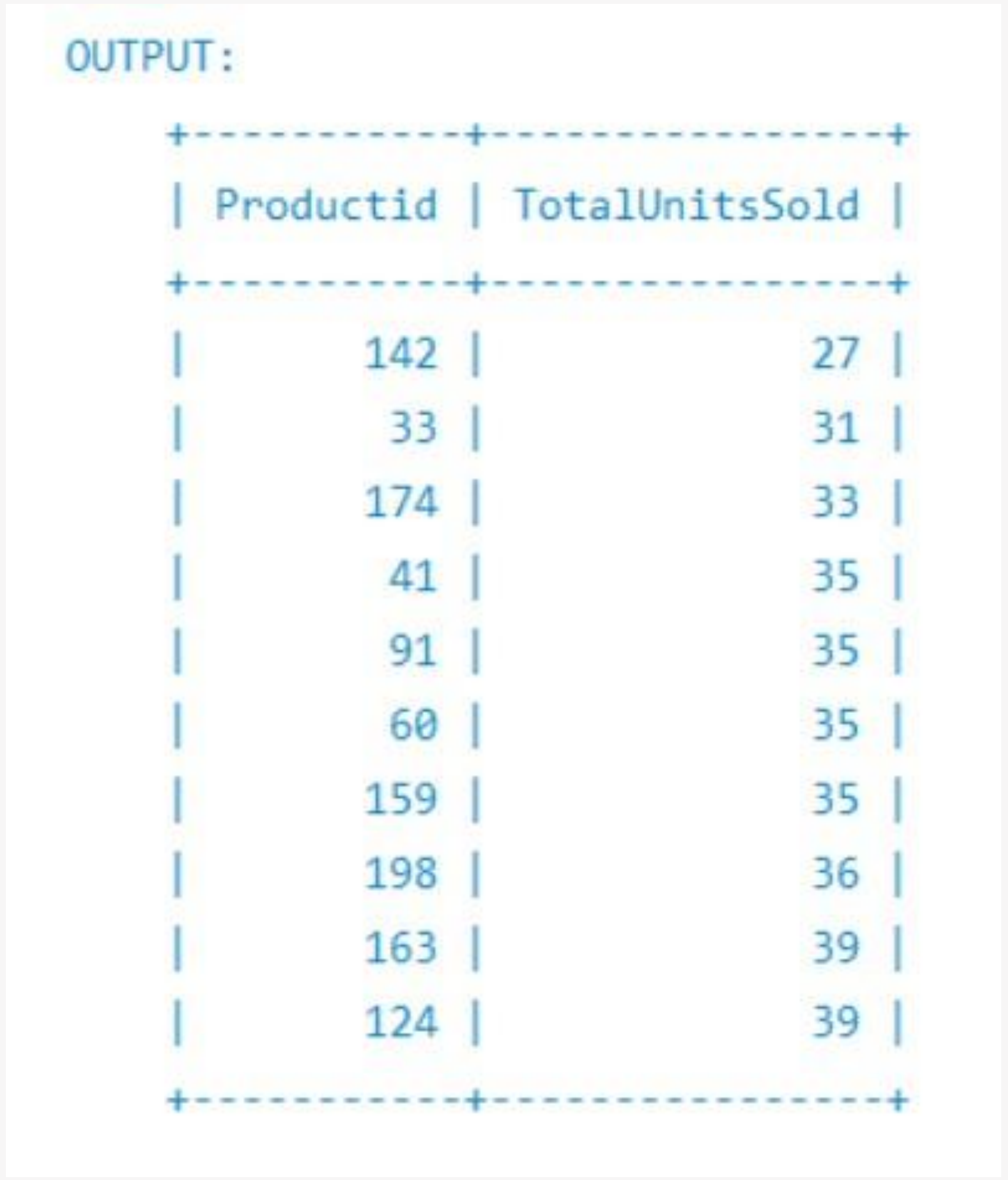
Insight: Top revenue contributors mirror those in Q1..

Low Sales Products

Problem : Find the ten products with the least amount of units sold (at least 1 unit sold).

Query:

```
SELECT
    ProductID,
    SUM(QuantityPurchased) AS TotalUnitsSold
FROM Sales_transaction
GROUP BY ProductID
HAVING SUM(QuantityPurchased) > 0
ORDER BY TotalUnitsSold ASC
LIMIT 10;
```



```
OUTPUT:

+-----------+----------------+
| Productid | TotalUnitsSold |
+-----------+----------------+
|       142 |             27 |
|        33 |             31 |
|       174 |             33 |
|        41 |             35 |
|        91 |             35 |
|        60 |             35 |
|       159 |             35 |
|       198 |             36 |
|       163 |             39 |
|       124 |             39 |
+-----------+----------------+
```

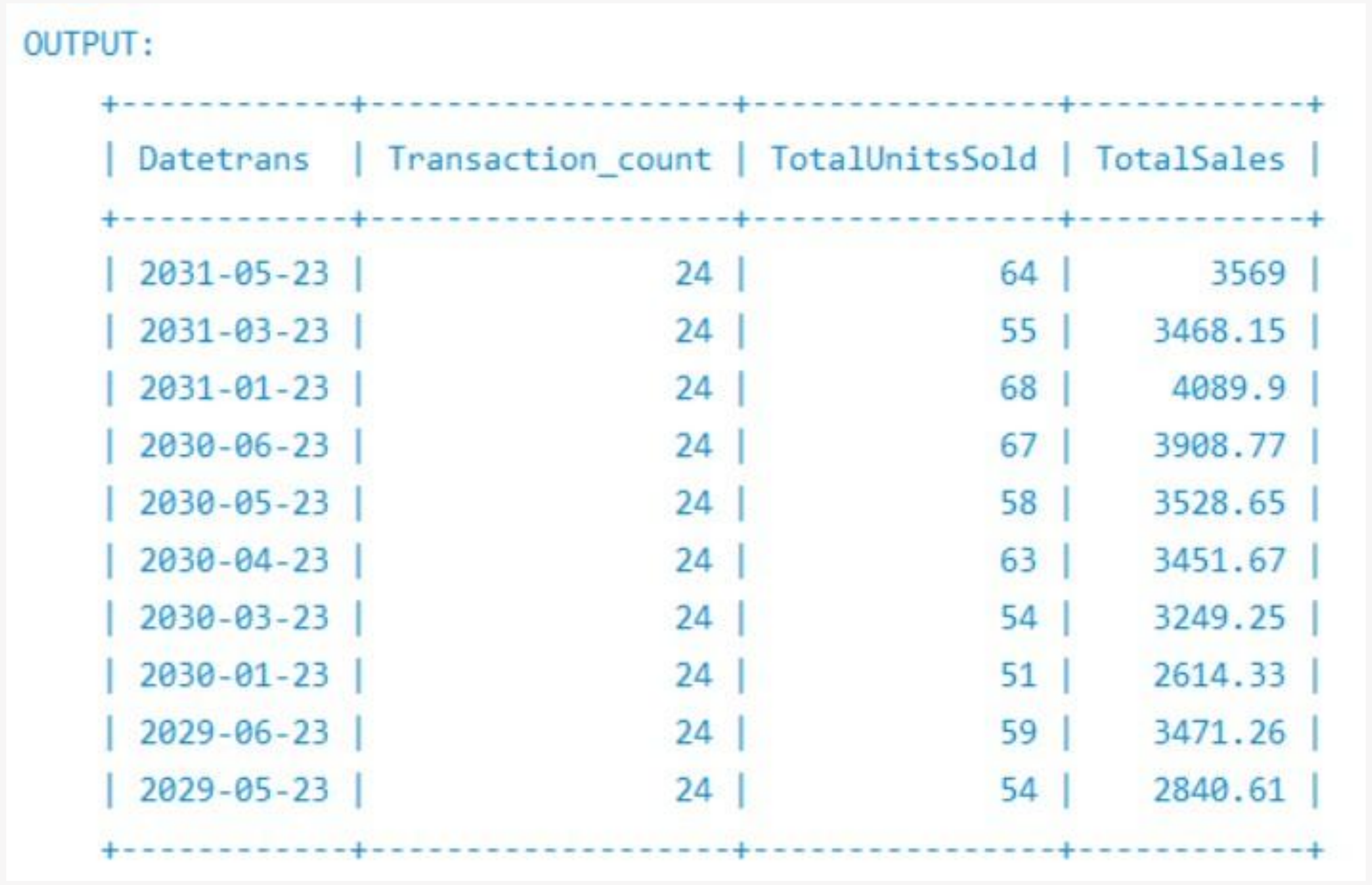
Insight: Products like 142, 33, and 174 had the lowest sales.

Sales Trend

Problem : Identify the sales trend based on daily data.

Query:

```
SELECT
   TransactionDate AS DateTrans,
   COUNT(*) AS Transaction_count,
   SUM(QuantityPurchased) AS TotalUnitsSold,
   ROUND(SUM(Price * QuantityPurchased), 2)
AS TotalSales
FROM Sales_transaction
GROUP BY DateTrans
ORDER BY DateTrans DESC
LIMIT 10;
```



```
OUTPUT:

    +------------+-------------------+----------------+------------+
    | Datetrans  | Transaction_count | TotalUnitsSold | TotalSales |
    +------------+-------------------+----------------+------------+
    | 2031-05-23 |                24 |             64 |       3569 |
    | 2031-03-23 |                24 |             55 |    3468.15 |
    | 2031-01-23 |                24 |             68 |     4089.9 |
    | 2030-06-23 |                24 |             67 |    3908.77 |
    | 2030-05-23 |                24 |             58 |    3528.65 |
    | 2030-04-23 |                24 |             63 |    3451.67 |
    | 2030-03-23 |                24 |             54 |    3249.25 |
    | 2030-01-23 |                24 |             51 |    2614.33 |
    | 2029-06-23 |                24 |             59 |    3471.26 |
    | 2029-05-23 |                24 |             54 |    2840.61 |
    +------------+-------------------+----------------+------------+
```

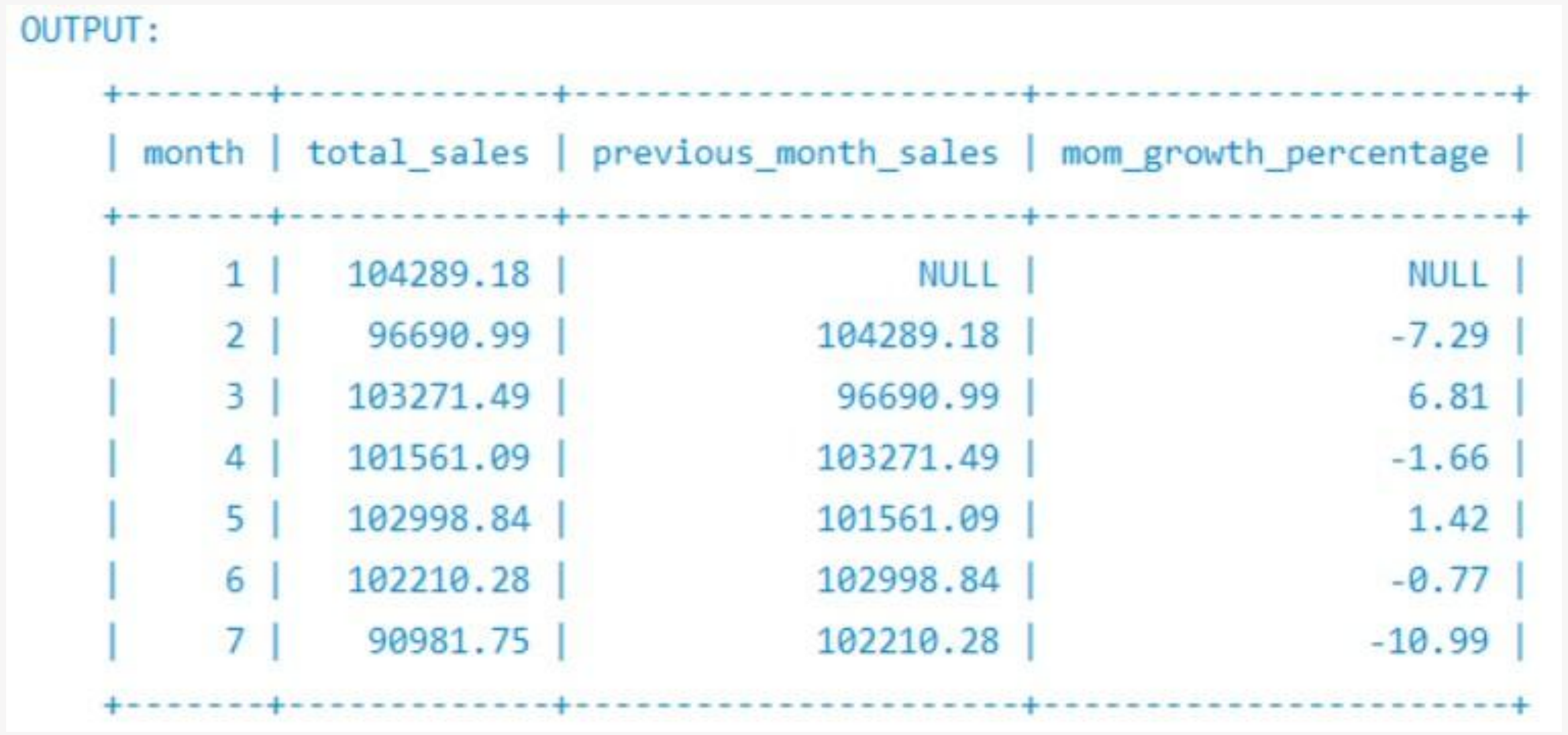
Insight: Revenue is stable across high-transaction days..

Growth Rate of Sales

Problem :Calculate month-on-month growth rate of sales.

Query:

```
WITH Monthly_sales AS
 (
   SELECT
      EXTRACT(MONTH FROM TransactionDate) AS month,
      ROUND(SUM(QuantityPurchased * Price), 2) AS total_sales
   FROM Sales_transaction
   GROUP BY EXTRACT(MONTH FROM TransactionDate)
 )
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 Monthly_sales
ORDER BY month;
```



```
OUTPUT:

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

Insight: Monthly growth fluctuates significantly..

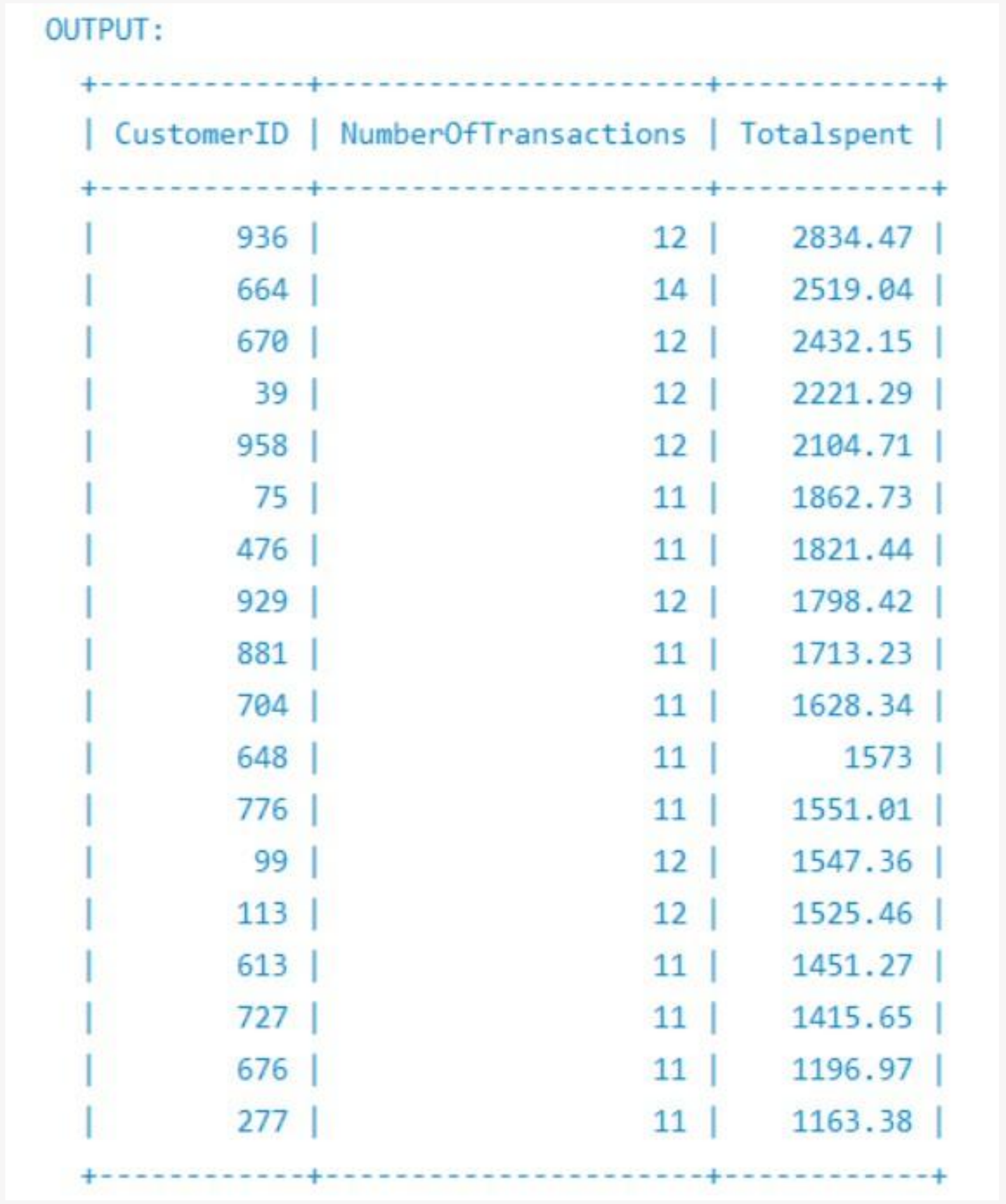
High Purchase Frequency

Problem :Describes the number of transaction along with the total amount spent by each customer. Having number of transactions more than 10 and TotalSpent more than 1000.

Query:

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

Insight: 17+ loyal customers spent over ₹1000 each.



```
OUTPUT:
+------------+----------------------+------------+
| 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 |
|        113 |                   12 |    1525.46 |
|        613 |                   11 |    1451.27 |
|        727 |                   11 |    1415.65 |
|        676 |                   11 |    1196.97 |
|        277 |                   11 |    1163.38 |
+------------+----------------------+------------+
```

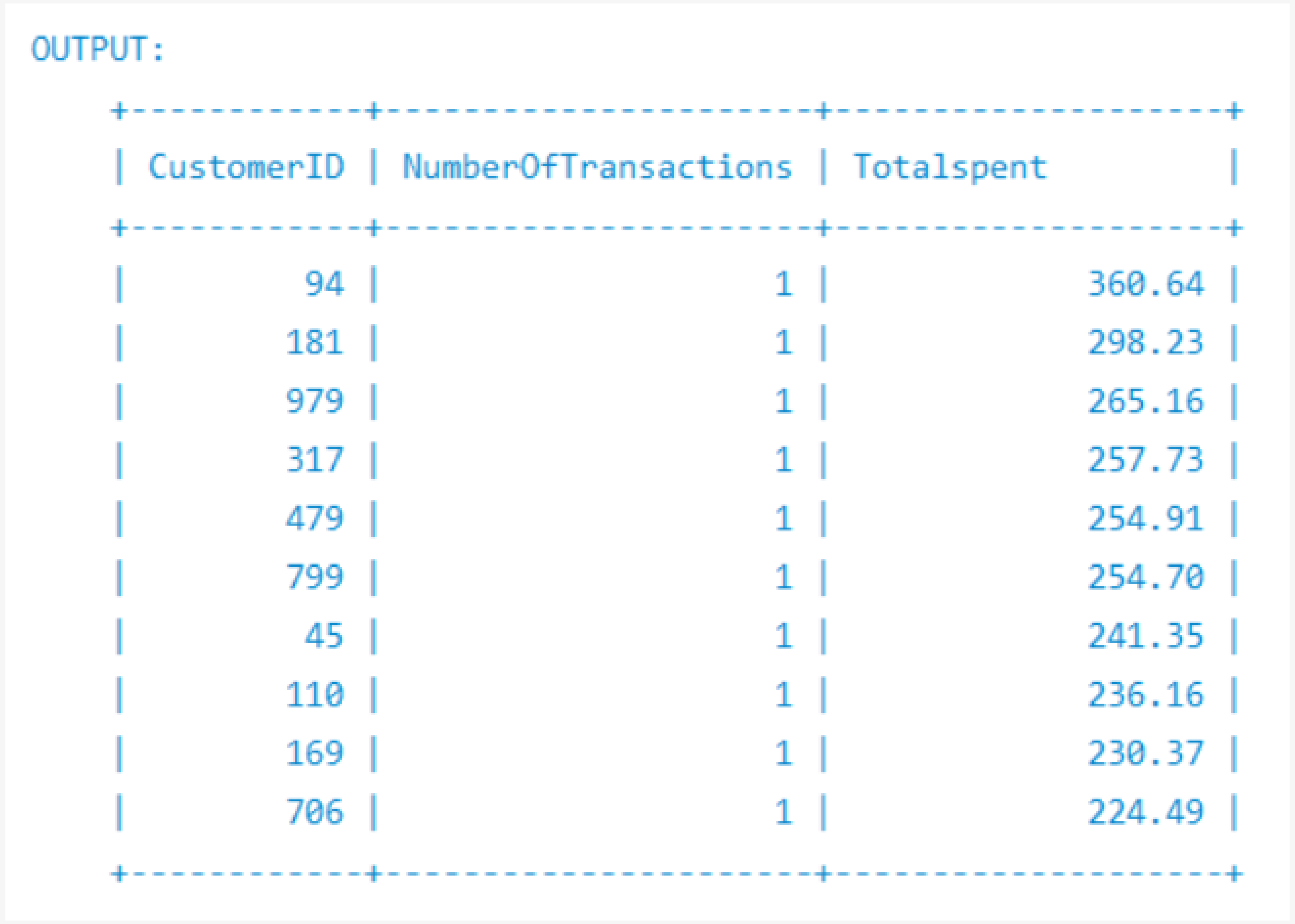
Occasional Customers

Problem: Identify the occasional customers who exhibit low purchase frequency within the company.

Query:

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

Insight: Many are one-time buyers.



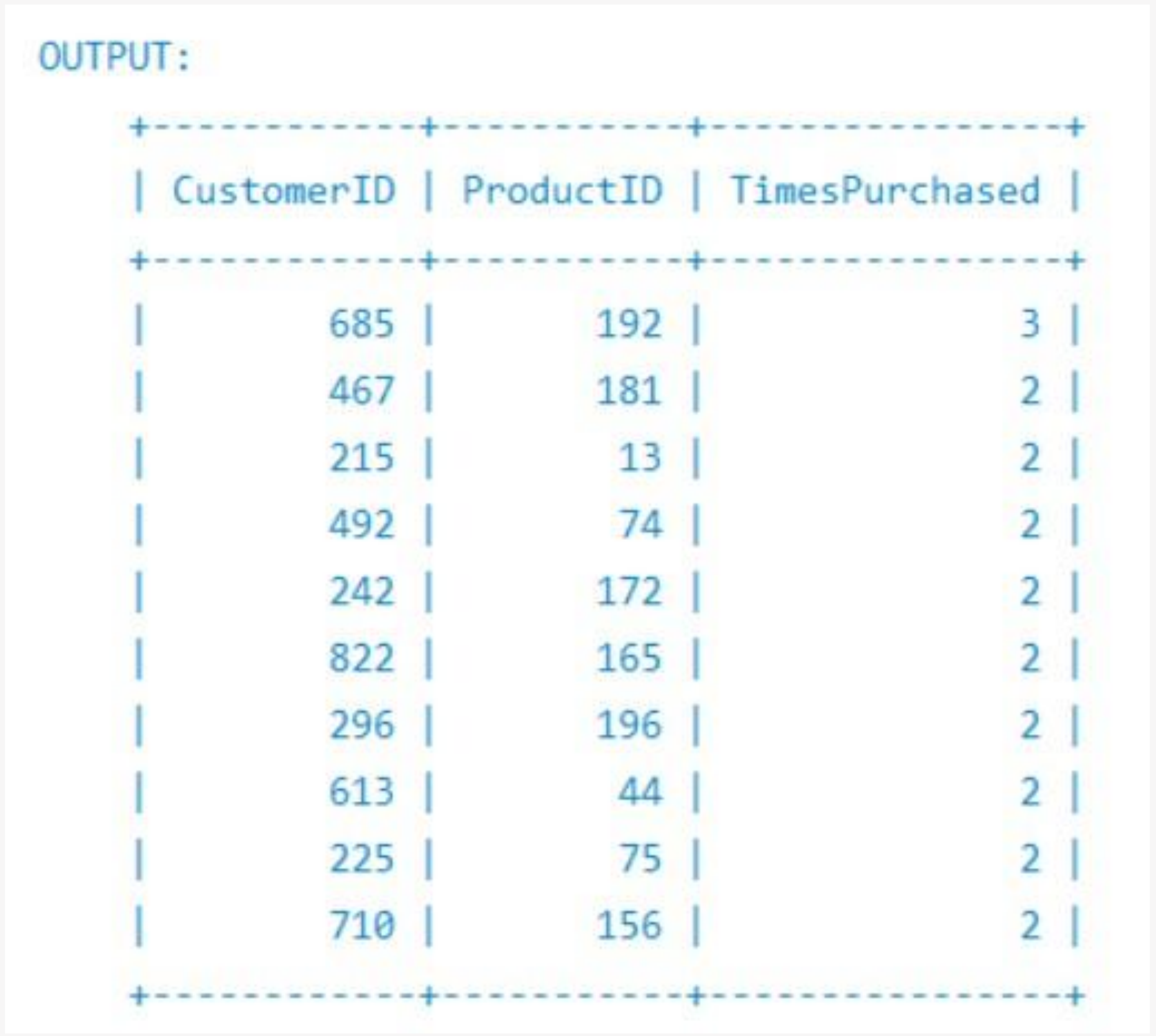
```
OUTPUT:

+------------+----------------------+--------------------+
| 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 |             254.70 |
|         45 |                    1 |             241.35 |
|        110 |                    1 |             236.16 |
|        169 |                    1 |             230.37 |
|        706 |                    1 |             224.49 |
+------------+----------------------+--------------------+
```

Repeat Purchases

Problem: Describes the total number of purchases made by each customer against each productID to understand the repeat customers in the company.

Query:

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



```
OUTPUT:

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

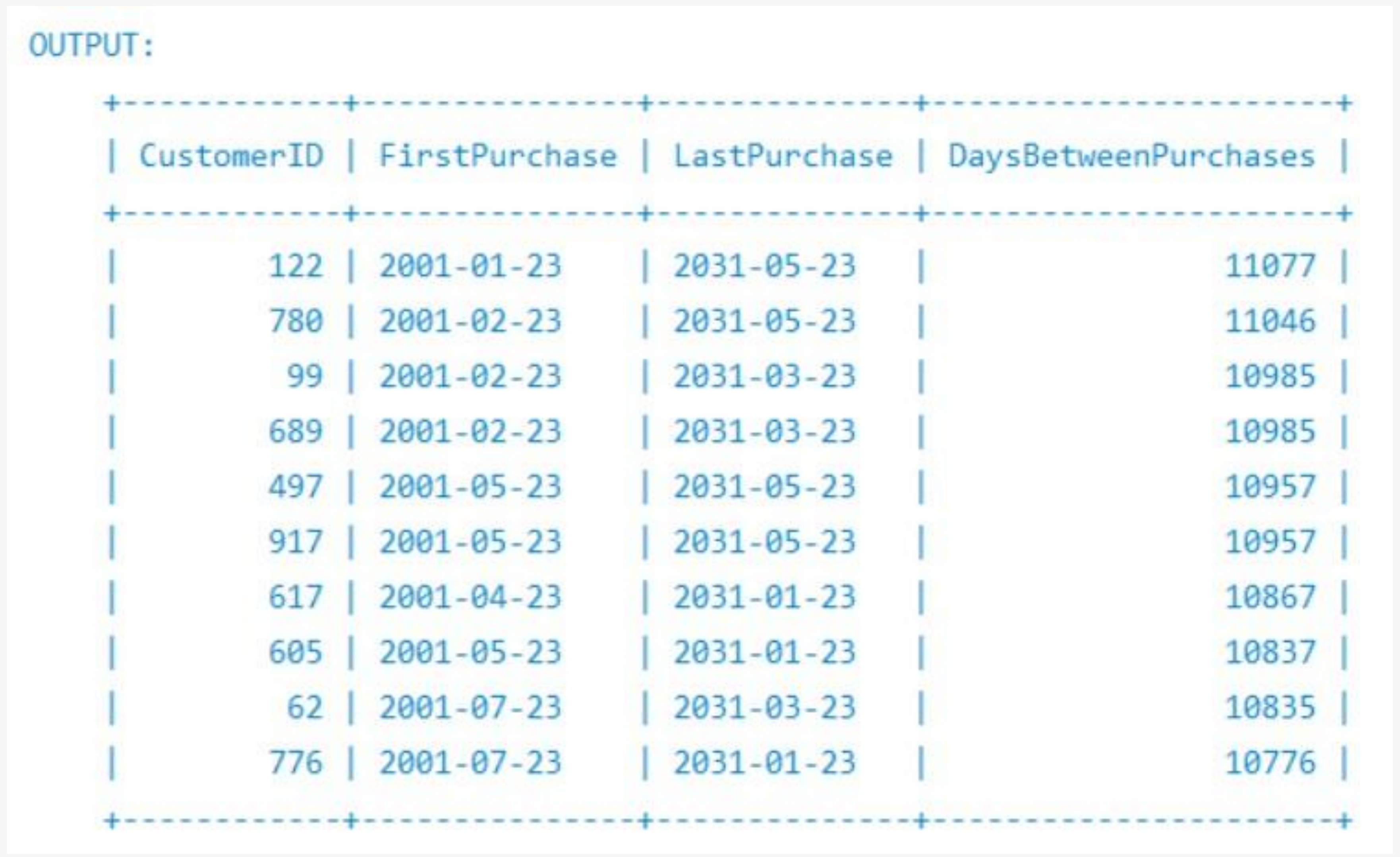
Insight: Repeat buys show brand/product loyalty.

Loyalty Indicators

Problem: Calculate duration between first and last purchase in that particular company to understand the loyalty of the customer.

Query:

```
SELECT
   CustomerID,
   MIN(TransactionDate) AS FirstPurchase,
   MAX(TransactionDate) AS LastPurchase,
   DATEDIFF(MAX(TransactionDate),
MIN(TransactionDate)) AS
DaysBetweenPurchases
FROM Sales_transaction
GROUP BY CustomerID
HAVING DaysBetweenPurchases > 0
ORDER BY DaysBetweenPurchases DESC;
```



```
OUTPUT:
    +------------+---------------+--------------+----------------------+
    | CustomerID | FirstPurchase | LastPurchase | DaysBetweenPurchases |
    +------------+---------------+--------------+----------------------+
    |        122 | 2001-01-23    | 2031-05-23   |                11077 |
    |        780 | 2001-02-23    | 2031-05-23   |                11046 |
    |         99 | 2001-02-23    | 2031-03-23   |                10985 |
    |        689 | 2001-02-23    | 2031-03-23   |                10985 |
    |        497 | 2001-05-23    | 2031-05-23   |                10957 |
    |        917 | 2001-05-23    | 2031-05-23   |                10957 |
    |        617 | 2001-04-23    | 2031-01-23   |                10867 |
    |        605 | 2001-05-23    | 2031-01-23   |                10837 |
    |         62 | 2001-07-23    | 2031-03-23   |                10835 |
    |        776 | 2001-07-23    | 2031-01-23   |                10776 |
    +------------+---------------+--------------+----------------------+
```

Insight: Some customers have over 30 years of purchase history.
These are highly loyal customers.

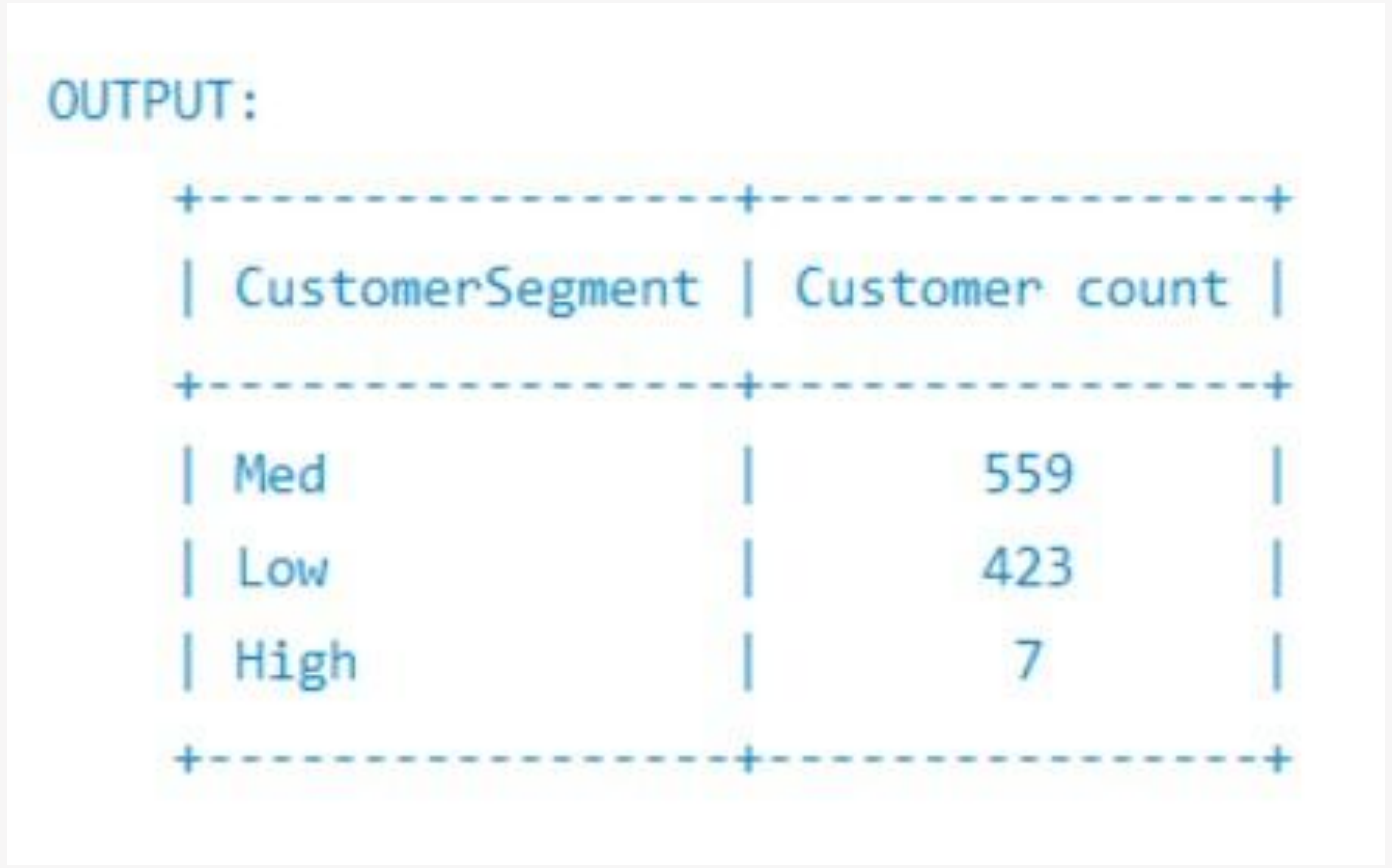
Customer Segmentation

Problem: Segment customers based on quantity purchased.
Customer segments on the following criteria-

Total quantity of the purchased products	vs Customer_segment
1-10	LOW
11-30	MID
>30	HIGH

Query:

```
SELECT
   CASE
      WHEN TotalQuantity BETWEEN 1 AND 10 THEN 'Low'
      WHEN TotalQuantity BETWEEN 11 AND 30 THEN 'Med'
      WHEN TotalQuantity > 30 THEN 'High'
   END AS CustomerSegment,
   COUNT(*)
FROM (
   SELECT
      s.CustomerID,
      SUM(s.QuantityPurchased) AS TotalQuantity
   FROM Sales_transaction s
   JOIN Customer_profiles c ON s.CustomerID = c.CustomerID
   GROUP BY s.CustomerID
) AS totalcustomers
GROUP BY CustomerSegment;
```



```
OUTPUT:

+-----------------+-----------------+
| CustomerSegment | Customer count  |
+-----------------+-----------------+
| Med             |      559        |
| Low             |      423        |
| High            |        7        |
+-----------------+-----------------+
```

Insight: Majority are in 'Med' segment, followed by 'Low'.