# Retail Transaction Analysis using PySpark

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

# Create Spark session
spark =
SparkSession.builder.appName("RetailTransactionAnalysis").getOrCreate(
)

# Load dataset
file_path = "synthetic_transactions.csv"  # adjust path if needed
df = spark.read.option("header", True).option("inferSchema",
True).csv(file_path)

# Add derived columns
df = df.withColumn("Revenue", col("Quantity") * col("Price")) \
       .withColumn("Year", year("TransactionDate")) \
       .withColumn("Month", month("TransactionDate")) \
       .withColumn("DayOfWeek", date_format("TransactionDate", "E")) \
       .withColumn("Hour", hour("TransactionDate"))

df.cache()
df.show(5)
```

```
+-------------+----------+---------+-----------+--------+-----+-------
+-------------------+-------------+----+-----+---------+----+
|TransactionID|CustomerID|ProductID|   Category|Quantity|Price|
Revenue|    TransactionDate|PaymentMethod|Year|Month|DayOfWeek|Hour|
+-------------+----------+---------+-----------+--------+-----+-------
+-------------------+-------------+----+-----+---------+----+
|      T000000|      1102|     P003|       Home|       2| 75.0|
150.0|2021-01-16 01:00:00|         Cash|2021|    1|      Sat|   1|
|      T000001|      1435|     P004|     Sports|       2| 60.0|
120.0|2022-07-14 23:00:00|  Net Banking|2022|    7|      Thu|  23|
|      T000002|      1860|     P004|     Sports|       3| 60.0|
180.0|2020-04-30 00:00:00|         Cash|2020|    4|      Thu|   0|
|      T000003|      1270|     P004|     Sports|       1| 60.0|
60.0|2022-06-05 07:00:00|          UPI|2022|    6|      Sun|   7|
|      T000004|      1106|     P001|Electronics|       2|100.0|
200.0|2023-10-20 01:00:00|         Cash|2023|   10|      Fri|   1|
+-------------+----------+---------+-----------+--------+-----+-------
+-------------------+-------------+----+-----+---------+----+
only showing top 5 rows
```

## Question 1: Print the schema and total number of records.

```
df.printSchema()
df.count()
```

```
root
 |-- TransactionID: string (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- ProductID: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- Price: double (nullable = true)
 |-- Revenue: double (nullable = true)
 |-- TransactionDate: timestamp (nullable = true)
 |-- PaymentMethod: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayOfWeek: string (nullable = true)
 |-- Hour: integer (nullable = true)


10000
```

## Question 2: Display 10 sample transactions.

```
df.show(10, truncate=False)
```

```
+-------------+----------+---------+-----------+--------+-----+-------
+------------------+-------------+----+-----+---------+----+
|TransactionID|CustomerID|ProductID|Category   |Quantity|Price|
Revenue|TransactionDate    |PaymentMethod|Year|Month|DayOfWeek|Hour|
+-------------+----------+---------+-----------+--------+-----+-------
+------------------+-------------+----+-----+---------+----+
|T000000      |1102      |P003     |Home       |2       |75.0 |150.0
|2021-01-16 01:00:00|Cash         |2021|1    |Sat      |1   |
|T000001      |1435      |P004     |Sports     |2       |60.0 |120.0
|2022-07-14 23:00:00|Net Banking  |2022|7    |Thu      |23  |
|T000002      |1860      |P004     |Sports     |3       |60.0 |180.0
|2020-04-30 00:00:00|Cash         |2020|4    |Thu      |0   |
|T000003      |1270      |P004     |Sports     |1       |60.0 |60.0
|2022-06-05 07:00:00|UPI          |2022|6    |Sun      |7   |
|T000004      |1106      |P001     |Electronics|2       |100.0|200.0
|2023-10-20 01:00:00|Cash         |2023|10   |Fri      |1   |
|T000005      |1071      |P005     |Books      |3       |30.0 |90.0
|2021-07-08 08:00:00|UPI          |2021|7    |Thu      |8   |
|T000006      |1700      |P003     |Home       |2       |75.0 |150.0
|2023-12-16 15:00:00|Cash         |2023|12   |Sat      |15  |
|T000007      |1020      |P004     |Sports     |3       |60.0 |180.0
|2023-06-13 09:00:00|Credit Card  |2023|6    |Tue      |9   |
|T000008      |1614      |P005     |Books      |4       |30.0 |120.0
|2022-05-05 16:00:00|Debit Card   |2022|5    |Thu      |16  |
```

```
|T000009      |1121      |P004     |Sports     |1       |60.0 |60.0
|2023-07-08 15:00:00|UPI          |2023|7    |Sat      |15  |
+-------------+----------+---------+-----------+--------+-----+-------
+-------------------+-------------+----+-----+---------+----+
only showing top 10 rows
```

## Question 3: What is the range of transaction dates?

```
df.select(min("TransactionDate"), max("TransactionDate")).show()
```

```
+-------------------+-------------------+
|min(TransactionDate)|max(TransactionDate)|
+-------------------+-------------------+
| 2020-01-01 02:00:00| 2023-12-30 20:00:00|
+-------------------+-------------------+
```

## Question 4: Count the number of unique customers.

```
df.select("CustomerID").distinct().count()
```

```
1000
```

## Question 5: List distinct products in the dataset.

```
df.select("Category").distinct().show(truncate=False)
df.select("ProductID").distinct().show(truncate=False)
```

```
+-----------+
|Category   |
+-----------+
|Home       |
|Sports     |
|Electronics|
|Clothing   |
|Books      |
+-----------+

+---------+
|ProductID|
+---------+
|P003     |
|P004     |
|P002     |
|P001     |
|P005     |
+---------+
```

## Question 6: What is the total revenue for each product?

```
df.groupBy("Category").agg(round(sum("Revenue"),
2).alias("TotalRevenue")).orderBy(desc("TotalRevenue")).show()
```

```
+-----------+------------+
|   Category|TotalRevenue|
+-----------+------------+
|Electronics|    510000.0|
|       Home|    359100.0|
|     Sports|    309900.0|
|   Clothing|    242050.0|
|      Books|    152670.0|
+-----------+------------+
```

## Question 7: Which product generated the highest revenue?

```
df.groupBy("Category").agg(sum("Revenue").alias("TotalRevenue")).order
By(desc("TotalRevenue")).limit(1).show()
```

```
+-----------+------------+
|   Category|TotalRevenue|
+-----------+------------+
|Electronics|    510000.0|
+-----------+------------+
```

# Question 8: What is the average quantity sold for each product?

```
df.groupBy("Category")\
.agg(round(avg("Quantity")).alias("AvgQuantity"))\
  .orderBy(desc("AvgQuantity"))\
  .show()
```

```
+-----------+-----------+
|   Category|AvgQuantity|
+-----------+-----------+
|Electronics|        3.0|
|       Home|        2.0|
|     Sports|        2.0|
|   Clothing|        2.0|
|      Books|        2.0|
+-----------+-----------+
```

## Question 9: Count how many transactions were made for each category.

```
df.groupBy("Category").count().orderBy(desc("count")).show()
```

```
+-----------+-----+
|   Category|count|
+-----------+-----+
|     Sports| 2069|
|      Books| 2058|
|Electronics| 1999|
|   Clothing| 1954|
|       Home| 1920|
+-----------+-----+
```

## Question 10: Find the top 3 products by revenue in each category.

```
product_revenue = df.groupBy("Category", "ProductID")\
                    .agg(sum("Revenue").alias("TotalRevenue"))
w = Window.partitionBy("Category").orderBy(desc("TotalRevenue"))

# Add rank and filter top 3 products per category
top_products = product_revenue.withColumn("Rank",
dense_rank().over(w))\
                              .filter(col("Rank") <= 3)
top_products.orderBy("Category", "Rank").show(3, truncate=False)
```

```
+-----------+---------+------------+----+
|Category   |ProductID|TotalRevenue|Rank|
+-----------+---------+------------+----+
|Books      |P005     |152670.0    |1   |
|Clothing   |P002     |242050.0    |1   |
|Electronics|P001     |510000.0    |1   |
+-----------+---------+------------+----+
only showing top 3 rows
```

### Q11: What is the average order value (AOV)?

```
df.select(round(avg("Revenue"), 2).alias("AOV")).show()
```

```
+------+
|   AOV|
+------+
|157.37|
+------+
```

### Q12: Calculate total quantity sold by category.

```
df.groupBy("Category").agg(sum("Quantity").alias("TotalQuantity")).sho
w()
```

```
+-----------+-------------+
|   Category|TotalQuantity|
+-----------+-------------+
|       Home|         4788|
|     Sports|         5165|
|Electronics|         5100|
|   Clothing|         4841|
|      Books|         5089|
+-----------+-------------+
```

### Q13: Identify the month with the highest total revenue.

```
df.groupBy("Month").agg(sum("Revenue").alias("TotalRevenue")) \
  .orderBy(desc("TotalRevenue")).limit(1).show()
```

```
+-----+------------+
|Month|TotalRevenue|
+-----+------------+
|    7|    139610.0|
+-----+------------+
```

### Q14: Find the day of the week with the fewest transactions.

```
df.groupBy("DayOfWeek").count().orderBy("count").limit(1).show()
```

```
+---------+-----+
|DayOfWeek|count|
+---------+-----+
|      Wed| 1387|
+---------+-----+
```

### Q15: Calculate revenue per quarter across all years

```
df.withColumn("Quarter", quarter("TransactionDate")) \
  .groupBy("Year", "Quarter") \
  .agg(sum("Revenue").alias("TotalRevenue")) \
  .orderBy("Year", "Quarter").show()
```

```
+----+-------+------------+
|Year|Quarter|TotalRevenue|
+----+-------+------------+
|2020|      1|     97940.0|
|2020|      2|     97535.0|
```

```
|2020|      3|     98335.0|
|2020|      4|    102935.0|
|2021|      1|    100690.0|
|2021|      2|     91485.0|
|2021|      3|     94700.0|
|2021|      4|    104015.0|
|2022|      1|     94445.0|
|2022|      2|    109140.0|
|2022|      3|     99970.0|
|2022|      4|     98220.0|
|2023|      1|    100760.0|
|2023|      2|     94640.0|
|2023|      3|     91220.0|
|2023|      4|     97690.0|
+----+-------+------------+
```

### Q16: Which customer has the most transactions

```
df.groupBy("CustomerID").count().orderBy(desc("count")).limit(1).show(
)

+----------+-----+
|CustomerID|count|
+----------+-----+
|      1725|   22|
+----------+-----+
```

### Q17: Which customer has spent the most

```
df.groupBy("CustomerID").agg(sum("Revenue").alias("TotalSpend")) \
  .orderBy(desc("TotalSpend")).limit(1).show()

+----------+----------+
|CustomerID|TotalSpend|
+----------+----------+
|      1282|    3840.0|
+----------+----------+
```

### Q18: Top 5 customers by total revenue per year

```
df.groupBy("Year",
"CustomerID").agg(sum("Revenue").alias("TotalRevenue")) \
  .orderBy("Year", desc("TotalRevenue")).show(5)

+----+----------+------------+
|Year|CustomerID|TotalRevenue|
```

```
+----+----------+------------+
|2020|      1737|      1650.0|
|2020|      1880|      1320.0|
|2020|      1330|      1300.0|
|2020|      1423|      1280.0|
|2020|      1112|      1265.0|
+----+----------+------------+
only showing top 5 rows
```

### Q19: How many unique products has each customer purchased

```
df.groupBy("CustomerID").agg(countDistinct("ProductID").alias("UniqueProducts")).show()
```

```
+----------+--------------+
|CustomerID|UniqueProducts|
+----------+--------------+
|      1238|             5|
|      1088|             4|
|      1645|             4|
|      1580|             4|
|      1829|             5|
|      1342|             5|
|      1591|             4|
|      1959|             5|
|      1460|             4|
|      1522|             5|
|      1507|             5|
|      1084|             4|
|      1721|             4|
|      1395|             4|
|      1127|             5|
|      1025|             5|
|      1990|             4|
|      1896|             5|
|      1483|             3|
|      1699|             5|
+----------+--------------+
only showing top 20 rows
```

### Q20: List customers who only used one type of payment method

```
df.groupBy("CustomerID").agg(countDistinct("PaymentMethod").alias("PaymentTypes")) \
  .filter(col("PaymentTypes") == 1).show()
```

```
+----------+------------+
|CustomerID|PaymentTypes|
+----------+------------+
|      1544|           1|
+----------+------------+
```

### Q21: Count transactions per payment method

```
df.groupBy("PaymentMethod").count().orderBy(desc("count")).show()
```

```
+-------------+-----+
|PaymentMethod|count|
+-------------+-----+
|          UPI| 2095|
|   Debit Card| 2081|
|  Credit Card| 1964|
|  Net Banking| 1937|
|         Cash| 1923|
+-------------+-----+
```

### Q22: Most used payment method per product category

```
from pyspark.sql.window import Window
from pyspark.sql.functions import rank

df.groupBy("Category", "PaymentMethod").count() \
  .withColumn("rank",
rank().over(Window.partitionBy("Category").orderBy(desc("count")))) \
  .filter(col("rank") == 1).show()
```

```
+-----------+-------------+-----+----+
|   Category|PaymentMethod|count|rank|
+-----------+-------------+-----+----+
|      Books|          UPI|  450|   1|
|   Clothing|   Debit Card|  420|   1|
|Electronics|          UPI|  424|   1|
|       Home|   Debit Card|  397|   1|
|     Sports|   Debit Card|  462|   1|
+-----------+-------------+-----+----+
```

### Q23: Average transaction revenue per payment method

```
df.groupBy("PaymentMethod").agg(round(avg("Revenue"),
2).alias("AvgRevenue")).show()
```

```
+-------------+----------+
|PaymentMethod|AvgRevenue|
+-------------+----------+
|  Credit Card|    157.96|
|  Net Banking|    154.24|
|         Cash|    158.83|
|   Debit Card|    156.96|
|          UPI|    158.79|
+-------------+----------+
```

### Q24: Customers who used 3 or more different payment methods

```
df.groupBy("CustomerID").agg(countDistinct("PaymentMethod").alias("Methods")) \
  .filter(col("Methods") >= 3).show()
```

```
+----------+-------+
|CustomerID|Methods|
+----------+-------+
|      1238|      4|
|      1645|      5|
|      1342|      4|
|      1591|      4|
|      1959|      4|
|      1829|      5|
|      1580|      5|
|      1088|      5|
|      1896|      4|
|      1460|      4|
|      1507|      5|
|      1127|      5|
|      1990|      5|
|      1025|      3|
|      1395|      4|
|      1721|      4|
|      1522|      4|
|      1084|      5|
|      1303|      4|
|      1322|      5|
+----------+-------+
only showing top 20 rows
```

### Q25: Payment method with the highest average revenue

```
df.groupBy("PaymentMethod").agg(avg("Revenue").alias("AvgRevenue")) \
  .orderBy(desc("AvgRevenue")).limit(1).show()
```

```
+-------------+------------------+
|PaymentMethod|        AvgRevenue|
+-------------+------------------+
|         Cash|158.82735309412377|
+-------------+------------------+
```

### Q26: Revenue by hour of day

```
df.groupBy("Hour").agg(sum("Revenue").alias("TotalRevenue")).orderBy("Hour").show()
```

```
+----+------------+
|Hour|TotalRevenue|
+----+------------+
|   0|     65480.0|
|   1|     65545.0|
|   2|     65570.0|
|   3|     63060.0|
|   4|     68745.0|
|   5|     65640.0|
|   6|     66650.0|
|   7|     70185.0|
|   8|     59375.0|
|   9|     63865.0|
|  10|     64750.0|
|  11|     68145.0|
|  12|     61100.0|
|  13|     61605.0|
|  14|     62195.0|
|  15|     64000.0|
|  16|     68085.0|
|  17|     66510.0|
|  18|     66630.0|
|  19|     63200.0|
+----+------------+
only showing top 20 rows
```

### Q27: Peak transaction hours

```
df.groupBy("Hour").count().orderBy(desc("count")).show(5)
```

```
+----+-----+
|Hour|count|
+----+-----+
|  23|  444|
|  11|  443|
|  16|  440|
```

```
|  17|  434|
|  22|  433|
+----+-----+
only showing top 5 rows
```

### Q28: Total number of transactions per month

```
df.groupBy("Year", "Month").count().orderBy("Year", "Month").show()
```

```
+----+-----+-----+
|Year|Month|count|
+----+-----+-----+
|2020|    1|  208|
|2020|    2|  187|
|2020|    3|  218|
|2020|    4|  191|
|2020|    5|  237|
|2020|    6|  206|
|2020|    7|  217|
|2020|    8|  202|
|2020|    9|  194|
|2020|   10|  230|
|2020|   11|  213|
|2020|   12|  218|
|2021|    1|  220|
|2021|    2|  212|
|2021|    3|  220|
|2021|    4|  217|
|2021|    5|  183|
|2021|    6|  186|
|2021|    7|  215|
|2021|    8|  173|
+----+-----+-----+
only showing top 20 rows
```

### Q29: Most popular shopping days (across years)

```
df.groupBy("DayOfWeek").count().orderBy(desc("count")).show()
```

```
+---------+-----+
|DayOfWeek|count|
+---------+-----+
|      Fri| 1479|
|      Mon| 1451|
|      Thu| 1444|
|      Sat| 1436|
|      Sun| 1408|
```

```
|      Tue| 1395|
|      Wed| 1387|
+---------+-----+
```

### Q30: Weekday-month combo with most transactions > ₹200

```
df.filter(col("Revenue") > 200) \
  .groupBy("DayOfWeek", "Month").count() \
  .orderBy(desc("count")).show(1)
```

```
+---------+-----+-----+
|DayOfWeek|Month|count|
+---------+-----+-----+
|      Thu|    7|   44|
+---------+-----+-----+
only showing top 1 row
```

### Q31: Average transaction amount per category per year

```
df.groupBy("Year", "Category") \
  .agg(round(avg("Revenue"),
2).alias("AvgRevenue")).orderBy("Year").show()
```

```
+----+-----------+----------+
|Year|   Category|AvgRevenue|
+----+-----------+----------+
|2020|     Sports|    149.11|
|2020|Electronics|    254.71|
|2020|   Clothing|    123.51|
|2020|       Home|    181.24|
|2020|      Books|     75.32|
|2021|      Books|     73.18|
|2021|       Home|    187.82|
|2021|     Sports|    152.29|
|2021|   Clothing|    123.39|
|2021|Electronics|    257.87|
|2022|   Clothing|    126.29|
|2022|       Home|    190.33|
|2022|     Sports|     148.6|
|2022|      Books|     74.95|
|2022|Electronics|    248.63|
|2023|   Clothing|    122.17|
|2023|       Home|    188.94|
|2023|Electronics|    259.42|
|2023|     Sports|    149.25|
|2023|      Books|     73.34|
```

```
+----+-----------+----------+
```

### Q32: Monthly revenue totals

```
df.groupBy("Year", "Month").agg(sum("Revenue").alias("TotalRevenue"))
\
  .orderBy("Year", "Month").show()
```

```
+----+-----+------------+
|Year|Month|TotalRevenue|
+----+-----+------------+
|2020|    1|     32625.0|
|2020|    2|     28835.0|
|2020|    3|     36480.0|
|2020|    4|     29065.0|
|2020|    5|     36055.0|
|2020|    6|     32415.0|
|2020|    7|     36665.0|
|2020|    8|     29820.0|
|2020|    9|     31850.0|
|2020|   10|     35680.0|
|2020|   11|     33710.0|
|2020|   12|     33545.0|
|2021|    1|     33505.0|
|2021|    2|     32555.0|
|2021|    3|     34630.0|
|2021|    4|     35655.0|
|2021|    5|     27660.0|
|2021|    6|     28170.0|
|2021|    7|     32665.0|
|2021|    8|     28220.0|
+----+-----+------------+
only showing top 20 rows
```

### Q33: Repeat customers per year

```
yearly_customers = df.select("CustomerID", "Year").distinct()
repeat_customers =
yearly_customers.groupBy("CustomerID").count().filter("count > 1")
repeat_customers.count()
```

```
994
```

### Q34: Customers with increasing yearly spending

```
df.groupBy("CustomerID", "Year") \
  .agg(sum("Revenue").alias("YearlySpend")) \
  .orderBy("CustomerID", "Year").show()
```

```
+----------+----+-----------+
|CustomerID|Year|YearlySpend|
+----------+----+-----------+
|      1000|2020|      465.0|
|      1000|2021|      640.0|
|      1000|2022|      500.0|
|      1000|2023|      670.0|
|      1001|2020|      700.0|
|      1001|2021|      470.0|
|      1001|2022|      225.0|
|      1001|2023|      300.0|
|      1002|2020|       90.0|
|      1002|2021|      600.0|
|      1002|2022|      820.0|
|      1002|2023|      400.0|
|      1003|2020|      600.0|
|      1003|2021|      200.0|
|      1003|2022|      500.0|
|      1004|2020|      775.0|
|      1004|2021|      400.0|
|      1004|2022|      525.0|
|      1004|2023|      255.0|
|      1005|2020|      440.0|
+----------+----+-----------+
only showing top 20 rows
```

### Q35: Highest revenue category per year

```
window = Window.partitionBy("Year").orderBy(desc("TotalRevenue"))
df.groupBy("Year", "Category") \
  .agg(sum("Revenue").alias("TotalRevenue")) \
  .withColumn("rank", rank().over(window)) \
  .filter(col("rank") == 1).show()
```

```
+----+-----------+------------+----+
|Year|   Category|TotalRevenue|rank|
+----+-----------+------------+----+
|2020|Electronics|    127100.0|   1|
|2021|Electronics|    121200.0|   1|
|2022|Electronics|    126800.0|   1|
|2023|Electronics|    134900.0|   1|
+----+-----------+------------+----+
```

### Q36: Add column for unit price

```
df = df.withColumn("UnitPrice", round(col("Revenue") /
col("Quantity"), 2))
df.select("ProductID", "Quantity", "Price", "Revenue",
"UnitPrice").show(5)
```

```
+---------+--------+-----+-------+---------+
|ProductID|Quantity|Price|Revenue|UnitPrice|
+---------+--------+-----+-------+---------+
|     P003|       2| 75.0|  150.0|     75.0|
|     P004|       2| 60.0|  120.0|     60.0|
|     P004|       3| 60.0|  180.0|     60.0|
|     P004|       1| 60.0|   60.0|     60.0|
|     P001|       2|100.0|  200.0|    100.0|
+---------+--------+-----+-------+---------+
only showing top 5 rows
```

### Q37: Extract year, month, and weekday

```
df.select("TransactionDate", "Year", "Month", "DayOfWeek").show(5)
```

```
+-------------------+----+-----+---------+
|    TransactionDate|Year|Month|DayOfWeek|
+-------------------+----+-----+---------+
|2021-01-16 01:00:00|2021|    1|      Sat|
|2022-07-14 23:00:00|2022|    7|      Thu|
|2020-04-30 00:00:00|2020|    4|      Thu|
|2022-06-05 07:00:00|2022|    6|      Sun|
|2023-10-20 01:00:00|2023|   10|      Fri|
+-------------------+----+-----+---------+
only showing top 5 rows
```

### Q38: Flag high-value transactions (Revenue > 300)

```
df.withColumn("HighValue", col("Revenue") > 300) \
  .select("CustomerID", "Revenue", "HighValue").show(5)
```

```
+----------+-------+---------+
|CustomerID|Revenue|HighValue|
+----------+-------+---------+
|      1102|  150.0|    false|
|      1435|  120.0|    false|
|      1860|  180.0|    false|
|      1270|   60.0|    false|
|      1106|  200.0|    false|
+----------+-------+---------+
only showing top 5 rows
```

### Q39: Categorize revenue into buckets

```
df = df.withColumn("RevenueBucket",
    when(col("Revenue") < 100, "Low")
   .when(col("Revenue") <= 200, "Mid")
   .when(col("Revenue") <= 300, "High")
   .otherwise("Premium"))
df.select("Revenue", "RevenueBucket").show(5)
```

```
+-------+-------------+
|Revenue|RevenueBucket|
+-------+-------------+
|  150.0|          Mid|
|  120.0|          Mid|
|  180.0|          Mid|
|   60.0|          Low|
|  200.0|          Mid|
+-------+-------------+
only showing top 5 rows
```

### Q40: Add column to calculate transaction hour.

```
df.select("TransactionDate", "Hour").show(5)
```

```
+-------------------+----+
|    TransactionDate|Hour|
+-------------------+----+
|2021-01-16 01:00:00|   1|
|2022-07-14 23:00:00|  23|
|2020-04-30 00:00:00|   0|
|2022-06-05 07:00:00|   7|
|2023-10-20 01:00:00|   1|
+-------------------+----+
only showing top 5 rows
```

### Q41: Products purchased per category

```
df.groupBy("Category").agg(count("ProductID").alias("ProductPurchaseCo
unt")).show()
```

```
+-----------+--------------------+
|   Category|ProductPurchaseCount|
+-----------+--------------------+
|       Home|                1920|
|     Sports|                2069|
|Electronics|                1999|
|   Clothing|                1954|
|      Books|                2058|
```

```
+-----------+--------------------+
```

### Q42: Total revenue per category per year

```
df.groupBy("Year",
"Category").agg(sum("Revenue").alias("TotalRevenue")).show()
```

```
+----+-----------+------------+
|Year|   Category|TotalRevenue|
+----+-----------+------------+
|2021|      Books|     39300.0|
|2023|   Clothing|     51800.0|
|2022|   Clothing|     61000.0|
|2023|       Home|     83700.0|
|2020|     Sports|     80220.0|
|2020|Electronics|    127100.0|
|2023|Electronics|    134900.0|
|2021|       Home|     89400.0|
|2021|     Sports|     75840.0|
|2020|   Clothing|     64100.0|
|2023|     Sports|     77460.0|
|2021|   Clothing|     65150.0|
|2022|       Home|     95925.0|
|2021|Electronics|    121200.0|
|2020|       Home|     90075.0|
|2020|      Books|     35250.0|
|2022|     Sports|     76380.0|
|2022|      Books|     41670.0|
|2023|      Books|     36450.0|
|2022|Electronics|    126800.0|
+----+-----------+------------+
```

### Q43: Group by customer and calculate AOV

```
df.groupBy("CustomerID").agg(round(avg("Revenue"),
2).alias("AvgOrderValue")).show()
```

```
+----------+-------------+
|CustomerID|AvgOrderValue|
+----------+-------------+
|      1645|       193.33|
|      1829|       161.82|
|      1959|       177.78|
|      1580|        130.0|
|      1238|       175.83|
|      1088|       191.25|
|      1591|        135.0|
```

```
|      1342|       146.54|
|      1460|        160.0|
|      1896|       155.77|
|      1127|       146.92|
|      1395|       128.57|
|      1483|        116.0|
|      1507|       126.67|
|      1084|       156.25|
|      1025|       116.25|
|      1522|       155.91|
|      1990|       160.63|
|      1721|       146.25|
|      1270|       154.58|
+----------+-------------+
only showing top 20 rows
```

### Q44: Total quantity and revenue per product per year

```
df.groupBy("Year", "ProductID") \
  .agg(sum("Quantity").alias("TotalQty"),
sum("Revenue").alias("TotalRev")).show()
```

```
+----+---------+--------+--------+
|Year|ProductID|TotalQty|TotalRev|
+----+---------+--------+--------+
|2022|     P001|    1268|126800.0|
|2023|     P002|    1036| 51800.0|
|2020|     P001|    1271|127100.0|
|2023|     P005|    1215| 36450.0|
|2021|     P005|    1310| 39300.0|
|2023|     P003|    1116| 83700.0|
|2021|     P002|    1303| 65150.0|
|2021|     P004|    1264| 75840.0|
|2022|     P004|    1273| 76380.0|
|2020|     P004|    1337| 80220.0|
|2023|     P001|    1349|134900.0|
|2020|     P003|    1201| 90075.0|
|2020|     P005|    1175| 35250.0|
|2021|     P003|    1192| 89400.0|
|2020|     P002|    1282| 64100.0|
|2022|     P002|    1220| 61000.0|
|2022|     P003|    1279| 95925.0|
|2021|     P001|    1212|121200.0|
|2023|     P004|    1291| 77460.0|
|2022|     P005|    1389| 41670.0|
+----+---------+--------+--------+
```

### Q45: Top-selling product by quantity per year

```
window = Window.partitionBy("Year").orderBy(desc("Quantity"))
df.withColumn("rank", rank().over(window)) \
  .filter(col("rank") == 1) \
  .select("Year", "ProductID", "Quantity").show()
```

```
+----+---------+--------+
|Year|ProductID|Quantity|
+----+---------+--------+
|2020|     P002|       4|
|2020|     P004|       4|
|2020|     P002|       4|
|2020|     P005|       4|
|2020|     P002|       4|
|2020|     P001|       4|
|2020|     P005|       4|
|2020|     P001|       4|
|2020|     P002|       4|
|2020|     P002|       4|
|2020|     P003|       4|
|2020|     P001|       4|
|2020|     P005|       4|
|2020|     P002|       4|
|2020|     P004|       4|
|2020|     P005|       4|
|2020|     P005|       4|
|2020|     P002|       4|
|2020|     P002|       4|
|2020|     P004|       4|
+----+---------+--------+
only showing top 20 rows
```

### Q46: Save monthly revenue summary as a CSV

```
monthly_revenue = df.groupBy("Year", "Month") \
    .agg(sum("Revenue").alias("TotalRevenue")) \
    .orderBy("Year", "Month")

monthly_revenue.write.mode("overwrite").option("header",
True).csv("output/monthly_revenue")
```

### Q47: Write customer-wise revenue totals to a Parquet file

```
customer_revenue = df.groupBy("CustomerID") \
    .agg(sum("Revenue").alias("TotalRevenue"))

customer_revenue.write.mode("overwrite").parquet("output/customer_reve
nue_totals")
```

### Q48: Export a summary table showing number of transactions per year

```
transactions_per_year = df.groupBy("Year").count()

transactions_per_year.write.mode("overwrite").option("header",
True).csv("output/transactions_per_year")
```

### Q49: Export a table with category vs year as a pivot with total revenue

```
pivot_table = df.groupBy("Category", "Year") \
    .agg(sum("Revenue").alias("TotalRevenue")) \
    .groupBy("Category") \
    .pivot("Year") \
    .sum("TotalRevenue")

pivot_table.write.mode("overwrite").option("header",
True).csv("output/category_year_revenue_pivot")
```

### Q50: Save all high-value transactions (Revenue > ₹250) to a new CSV

```
high_value_txns = df.filter(col("Revenue") > 250)

high_value_txns.write.mode("overwrite").option("header",
True).csv("output/high_value_transactions")
```

## Create the customer_metadata DataFrame

```
from pyspark.sql import Row

# Sample metadata for illustration
metadata_rows = [
    Row(CustomerID=1012, Gender="Male", AgeGroup="25-34"),
    Row(CustomerID=1095, Gender="Female", AgeGroup="35-44"),
    Row(CustomerID=1001, Gender="Male", AgeGroup="18-24")
]

customer_metadata = spark.createDataFrame(metadata_rows)
customer_metadata.show()
```

```
+----------+------+--------+
|CustomerID|Gender|AgeGroup|
+----------+------+--------+
|      1012|  Male|   25-34|
|      1095|Female|   35-44|
|      1001|  Male|   18-24|
+----------+------+--------+
```

### Q51: Total revenue generated by each gender

```
joined_df = df.join(customer_metadata, on="CustomerID", how="inner")

joined_df.groupBy("Gender").agg(sum("Revenue").alias("TotalRevenue")).
show()
```

```
+------+------------+
|Gender|TotalRevenue|
+------+------------+
|  Male|      3615.0|
|Female|      1560.0|
+------+------------+
```

### Q52: Identify customers who made at least one transaction every year (2020–2023)

```
years_active = df.select("CustomerID", "Year").distinct() \
    .groupBy("CustomerID").agg(countDistinct("Year").alias("Years"))

years_active.filter(col("Years") == 4).select("CustomerID").show()
```

```
+----------+
|CustomerID|
+----------+
|      1580|
|      1645|
|      1342|
|      1238|
|      1025|
|      1721|
|      1507|
|      1990|
|      1896|
|      1127|
|      1618|
|      1352|
|      1139|
|      1270|
|      1699|
|      1339|
|      1975|
|      1265|
|      1884|
|      1223|
+----------+
only showing top 20 rows
```

### Q53: Product each customer spent the most on

```
from pyspark.sql.window import Window
spending = df.groupBy("CustomerID", "ProductID") \
    .agg(sum("Revenue").alias("TotalSpend"))

rank_window =
Window.partitionBy("CustomerID").orderBy(desc("TotalSpend"))

spending.withColumn("rank", rank().over(rank_window)) \
    .filter(col("rank") == 1) \
    .select("CustomerID", "ProductID", "TotalSpend").show()
```

```
+----------+---------+----------+
|CustomerID|ProductID|TotalSpend|
+----------+---------+----------+
|      1000|     P001|     700.0|
|      1001|     P001|     800.0|
|      1002|     P001|     800.0|
|      1003|     P001|    1100.0|
|      1004|     P003|     975.0|
|      1005|     P001|     600.0|
|      1006|     P001|     500.0|
|      1007|     P001|     600.0|
|      1008|     P004|     240.0|
|      1009|     P001|    1300.0|
|      1010|     P001|     800.0|
|      1011|     P001|     600.0|
|      1011|     P004|     600.0|
|      1012|     P004|     780.0|
|      1013|     P002|     300.0|
|      1014|     P001|     600.0|
|      1015|     P001|     300.0|
|      1015|     P004|     300.0|
|      1016|     P001|     700.0|
|      1017|     P004|     420.0|
+----------+---------+----------+
only showing top 20 rows
```

### Q54: Product Affinity — Pairs of products bought by same customer

```
df1 = df.select("CustomerID", "ProductID").distinct().alias("a")
df2 = df.select("CustomerID", "ProductID").distinct().alias("b")

product_pairs = df1.join(df2, on="CustomerID") \
    .filter(col("a.ProductID") < col("b.ProductID")) \
    .groupBy("a.ProductID", "b.ProductID") \
    .agg(countDistinct("CustomerID").alias("CustomerCount"))

product_pairs.show()
```

```
+---------+---------+-------------+
|ProductID|ProductID|CustomerCount|
+---------+---------+-------------+
|     P001|     P004|          752|
|     P003|     P005|          747|
|     P001|     P003|          733|
|     P002|     P003|          730|
|     P001|     P002|          741|
|     P002|     P005|          756|
|     P001|     P005|          764|
|     P002|     P004|          761|
|     P004|     P005|          773|
|     P003|     P004|          745|
+---------+---------+-------------+
```

### Q55: Customers who bought in 2020/21 but NOT in 2022/23

```
from pyspark.sql.functions import collect_set

customer_years = df.select("CustomerID", "Year").distinct() \
    .groupBy("CustomerID").agg(collect_set("Year").alias("Years"))

churned = customer_years.filter(
    array_contains(col("Years"), 2020) | array_contains(col("Years"),
2021)
).filter(
    ~array_contains(col("Years"), 2022) &
~array_contains(col("Years"), 2023)
)

churned.select("CustomerID").show()
```

```
+----------+
|CustomerID|
+----------+
|      1109|
|      1218|
|      1271|
|      1331|
|      1447|
|      1544|
|      1940|
+----------+
```

### Q56: Upsell Detection — Low-value in 1st year & high-value later

```
from pyspark.sql.functions import min as min_, max as max_

first_last_txns = df.groupBy("CustomerID", "Year") \
    .agg(min_("Revenue").alias("MinRev"),
max_("Revenue").alias("MaxRev"))

low_then_high = first_last_txns.groupBy("CustomerID") \
    .agg(min_("MinRev").alias("FirstLow"),
max_("MaxRev").alias("LaterHigh")) \
    .filter((col("FirstLow") < 100) & (col("LaterHigh") > 300))

low_then_high.show()
```

```
+----------+--------+---------+
|CustomerID|FirstLow|LaterHigh|
+----------+--------+---------+
|      1829|    30.0|    400.0|
|      1591|    30.0|    400.0|
|      1238|    60.0|    400.0|
|      1025|    50.0|    400.0|
|      1522|    60.0|    400.0|
|      1303|    30.0|    400.0|
|      1270|    60.0|    400.0|
|      1699|    30.0|    400.0|
|      1143|    50.0|    400.0|
|      1223|    30.0|    400.0|
|      1157|    30.0|    400.0|
|      1766|    60.0|    400.0|
|      1133|    60.0|    400.0|
|      1016|    60.0|    400.0|
|      1005|    30.0|    400.0|
|      1160|    60.0|    400.0|
|      1468|    30.0|    400.0|
|      1417|    75.0|    400.0|
|      1525|    60.0|    400.0|
|      1863|    50.0|    400.0|
+----------+--------+---------+
only showing top 20 rows
```

### Q57: Product category switched year to year

```
spending = df.groupBy("CustomerID", "Year", "Category") \
    .agg(sum("Revenue").alias("Total"))

win = Window.partitionBy("CustomerID", "Year").orderBy(desc("Total"))

top_category = spending.withColumn("rank", rank().over(win)) \
    .filter(col("rank") == 1)
```

```
switched = top_category.groupBy("CustomerID") \
    .agg(collect_set("Category").alias("Categories")) \
    .filter(size("Categories") > 1)

switched.show()
```

```
+----------+--------------------+
|CustomerID|          Categories|
+----------+--------------------+
|      1000|[Clothing, Electr...|
|      1001|[Sports, Electron...|
|      1002|[Electronics, Boo...|
|      1004|[Sports, Electron...|
|      1005|[Sports, Electron...|
|      1006|[Sports, Electron...|
|      1007|[Sports, Electron...|
|      1008|[Sports, Electron...|
|      1009|[Sports, Electron...|
|      1010|[Clothing, Electr...|
|      1011|[Sports, Electron...|
|      1012|      [Sports, Home]|
|      1013|[Clothing, Sports...|
|      1014|[Clothing, Electr...|
|      1015|[Sports, Electron...|
|      1016|[Sports, Electron...|
|      1017|[Sports, Electron...|
|      1018|[Clothing, Electr...|
|      1019|[Sports, Books, H...|
|      1020|[Sports, Electron...|
+----------+--------------------+
only showing top 20 rows
```

### Q58: Year-over-year revenue growth per category

```
category_revenue = df.groupBy("Year", "Category") \
    .agg(sum("Revenue").alias("Revenue"))

window = Window.partitionBy("Category").orderBy("Year")

growth = category_revenue.withColumn("PrevRevenue",
lag("Revenue").over(window)) \
    .withColumn("GrowthPct", round(((col("Revenue") -
col("PrevRevenue")) / col("PrevRevenue")) * 100, 2))

growth.filter(col("PrevRevenue").isNotNull()).show()
```

```
+----+-----------+--------+-----------+---------+
|Year|   Category| Revenue|PrevRevenue|GrowthPct|
+----+-----------+--------+-----------+---------+
```

```
|2021|      Books| 39300.0|    35250.0|    11.49|
|2022|      Books| 41670.0|    39300.0|     6.03|
|2023|      Books| 36450.0|    41670.0|   -12.53|
|2021|   Clothing| 65150.0|    64100.0|     1.64|
|2022|   Clothing| 61000.0|    65150.0|    -6.37|
|2023|   Clothing| 51800.0|    61000.0|   -15.08|
|2021|Electronics|121200.0|   127100.0|    -4.64|
|2022|Electronics|126800.0|   121200.0|     4.62|
|2023|Electronics|134900.0|   126800.0|     6.39|
|2021|       Home| 89400.0|    90075.0|    -0.75|
|2022|       Home| 95925.0|    89400.0|      7.3|
|2023|       Home| 83700.0|    95925.0|   -12.74|
|2021|     Sports| 75840.0|    80220.0|    -5.46|
|2022|     Sports| 76380.0|    75840.0|     0.71|
|2023|     Sports| 77460.0|    76380.0|     1.41|
+----+-----------+--------+-----------+---------+
```

### Q59: Customers who purchased from 3+ categories

```
df.groupBy("CustomerID").agg(countDistinct("Category").alias("Category
Count")) \
  .filter(col("CategoryCount") >= 3).show()
```

```
+----------+-------------+
|CustomerID|CategoryCount|
+----------+-------------+
|      1580|            4|
|      1645|            4|
|      1342|            5|
|      1959|            5|
|      1238|            5|
|      1829|            5|
|      1591|            4|
|      1088|            4|
|      1127|            5|
|      1990|            4|
|      1507|            5|
|      1896|            5|
|      1721|            4|
|      1025|            5|
|      1395|            4|
|      1522|            5|
|      1084|            4|
|      1483|            3|
|      1460|            4|
|      1143|            5|
+----------+-------------+
```

```
only showing top 20 rows
```

### Q60: Last transaction category per customer

```
window =
Window.partitionBy("CustomerID").orderBy(desc("TransactionDate"))

df.withColumn("rank", rank().over(window)) \
  .filter(col("rank") == 1) \
  .select("CustomerID", "Category", "TransactionDate").show()
```

```
+----------+-----------+-------------------+
|CustomerID|   Category|    TransactionDate|
+----------+-----------+-------------------+
|      1000|      Books|2023-12-21 18:00:00|
|      1001|      Books|2023-03-15 16:00:00|
|      1002|Electronics|2023-05-14 12:00:00|
|      1003|   Clothing|2022-11-09 00:00:00|
|      1004|       Home|2023-12-18 03:00:00|
|      1005|       Home|2023-04-18 15:00:00|
|      1006|   Clothing|2023-09-20 16:00:00|
|      1007|      Books|2023-09-04 06:00:00|
|      1008|      Books|2023-12-27 17:00:00|
|      1009|     Sports|2023-08-05 13:00:00|
|      1010|Electronics|2023-09-09 00:00:00|
|      1011|      Books|2023-11-14 10:00:00|
|      1012|     Sports|2023-09-27 01:00:00|
|      1013|     Sports|2023-11-05 01:00:00|
|      1014|Electronics|2022-09-25 04:00:00|
|      1015|Electronics|2023-05-14 19:00:00|
|      1016|      Books|2023-11-10 20:00:00|
|      1017|       Home|2023-09-08 22:00:00|
|      1018|       Home|2023-05-24 03:00:00|
|      1019|       Home|2023-06-24 02:00:00|
+----------+-----------+-------------------+
only showing top 20 rows
```