PYSPARK CASE STUDY

Title: Online Banking Analysis

Data and Files Collected:

loan.csv

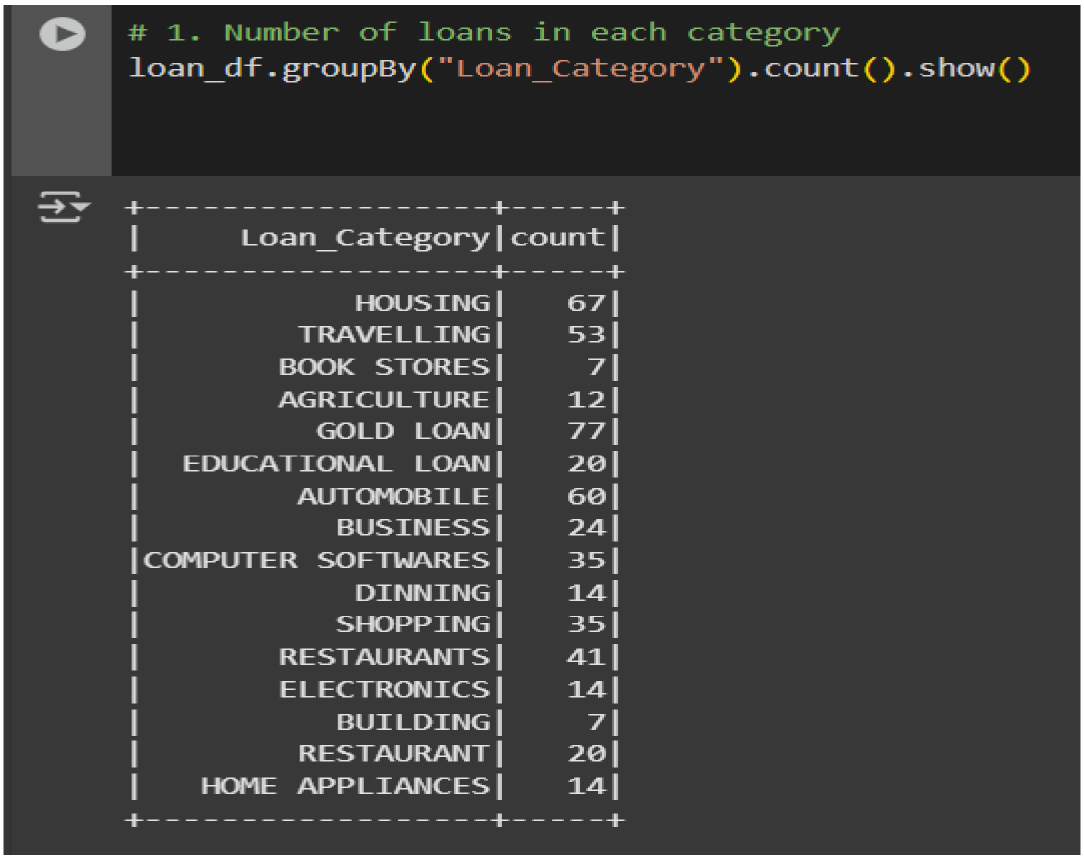
credit card.csv

txn.csv

In loandata.csv file

1.number of loans in each category

loan_df.groupBy("Loan_Category").count().show()

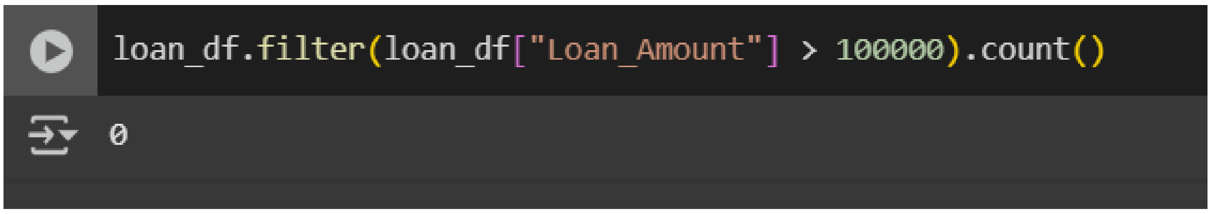


```
# 1. Number of loans in each category
loan_df.groupBy("Loan_Category").count().show()
```

```
+------------------+-----+
|     Loan_Category|count|
+------------------+-----+
|           HOUSING|   67|
|        TRAVELLING|   53|
|       BOOK STORES|    7|
|       AGRICULTURE|   12|
|         GOLD LOAN|   77|
|  EDUCATIONAL LOAN|   20|
|        AUTOMOBILE|   60|
|          BUSINESS|   24|
|COMPUTER SOFTWARES|   35|
|           DINNING|   14|
|          SHOPPING|   35|
|       RESTAURANTS|   41|
|       ELECTRONICS|   14|
|          BUILDING|    7|
|        RESTAURANT|   20|
|   HOME APPLIANCES|   14|
+------------------+-----+
```

2.number of people who have taken more than 1 lack loan

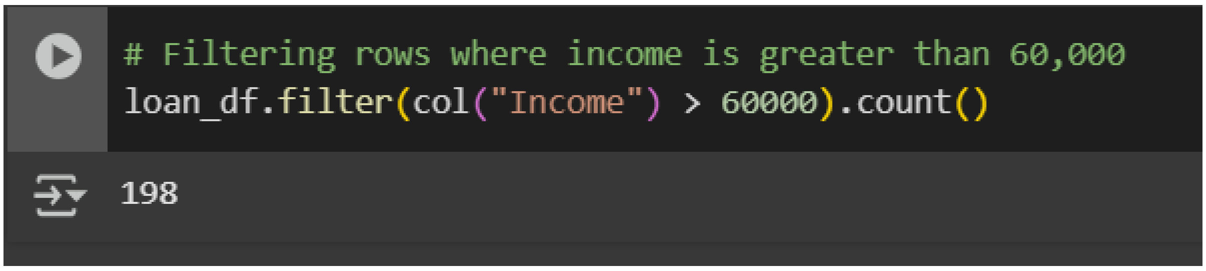
loan_df.filter(loan_df["Loan_Amount"] > 100000).count()



```
loan_df.filter(loan_df["Loan_Amount"] > 100000).count()
```
```
0
```

3.number of people with income greater than 60000 rupees

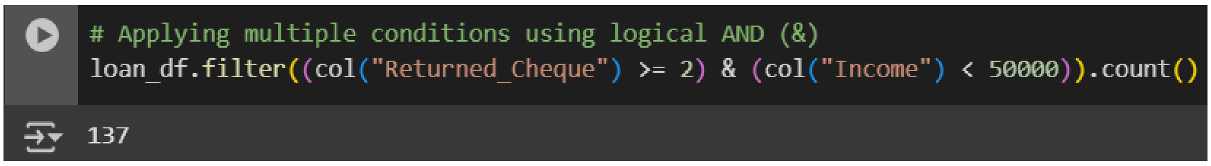
loan_df.filter(col("Income") > 60000).count()



```
# Filtering rows where income is greater than 60,000
loan_df.filter(col("Income") > 60000).count()
```
```
198
```

4.number of people with 2 or more returned cheques and income less than 50000

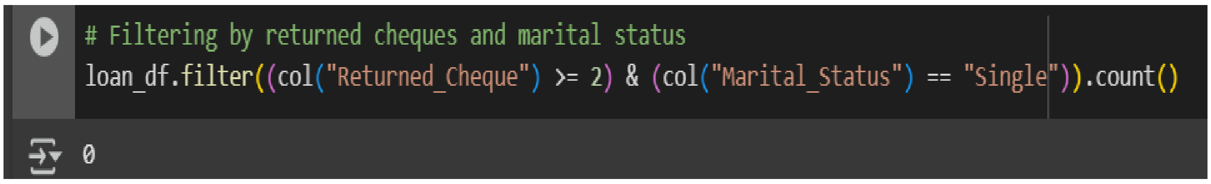
loan_df.filter((col("Returned_Cheque") >= 2) & (col("Income") < 50000)).count()



```
# Applying multiple conditions using logical AND (&)
loan_df.filter((col("Returned_Cheque") >= 2) & (col("Income") < 50000)).count()
```
```
137
```

5.number of people with 2 or more returned cheques and are single

loan_df.filter((col("Returned_Cheque") >= 2) & (col("Marital_Status") == "Single")).count()

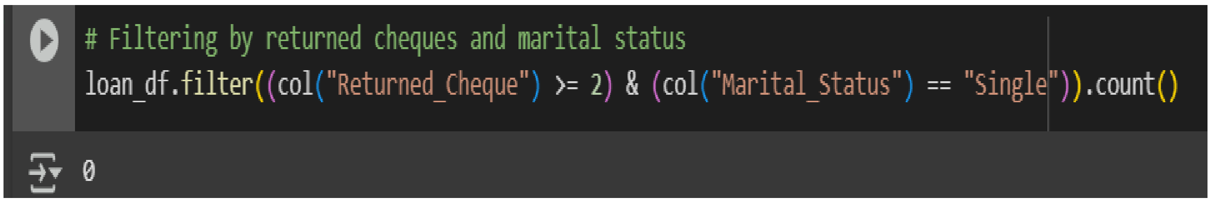


```
# Filtering by returned cheques and marital status
loan_df.filter((col("Returned_Cheque") >= 2) & (col("Marital_Status") == "Single")).count()
```

```
0
```

6.number of people with expenditure over 50000 a month

loan_df.filter(col("Expenditure") > 50000).count()

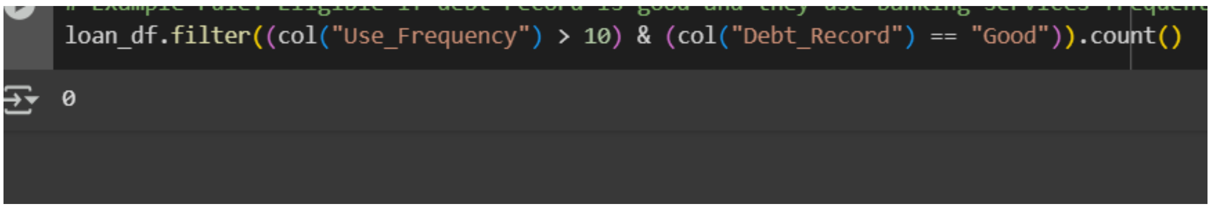


```
# Filtering by returned cheques and marital status
loan_df.filter((col("Returned_Cheque") >= 2) & (col("Marital_Status") == "Single")).count()
```

```
0
```

7.number of members who are elgible for credit card

loan_df.filter((col("Use_Frequency") > 10) &
(col("Debt_Record") == "Good")).count()



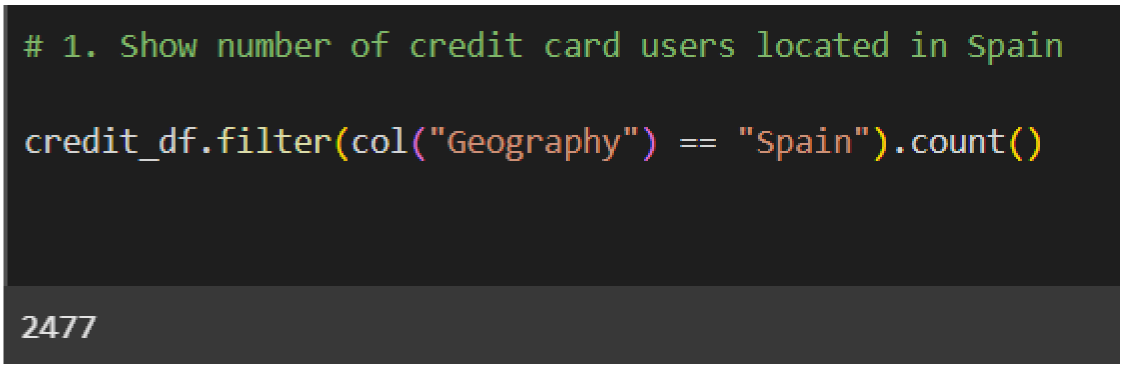
```
# Example rule: Eligible if debt record is good and they use banking services frequent
loan_df.filter((col("Use_Frequency") > 10) & (col("Debt_Record") == "Good")).count()
```

```
0
```

In credit.csv file

1.credit card users in Spain

credit_df.filter(col("Geography") == "Spain").count()



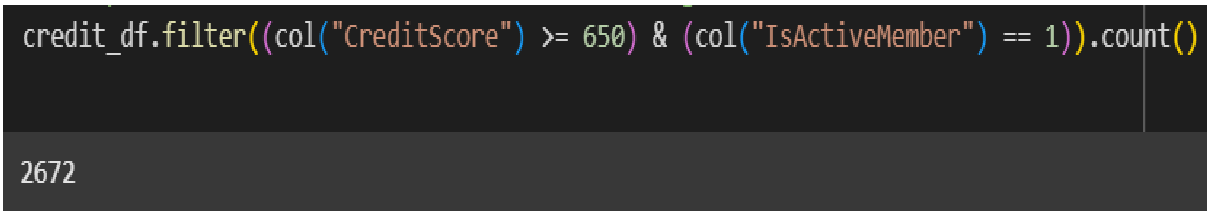
```
# 1. Show number of credit card users located in Spain

credit_df.filter(col("Geography") == "Spain").count()
```

```
2477
```

2.number of members who are elgible and active in the bank

credit_df.filter((col("CreditScore") >= 650) & (col("IsActiveMember") == 1)).count()



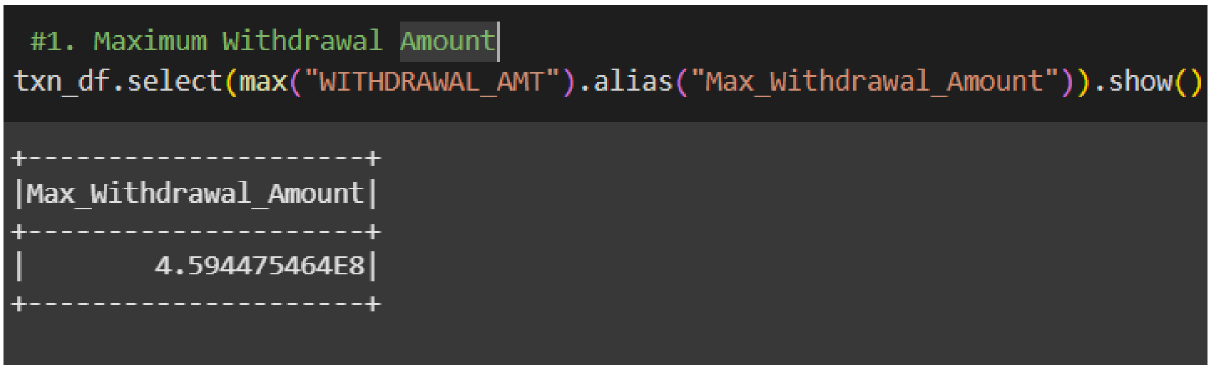
```
credit_df.filter((col("CreditScore") >= 650) & (col("IsActiveMember") == 1)).count()
```

```
2672
```

In Transactions file

1.Maximum withdrawal amount in transactions

txn_df.select(max("WITHDRAWAL_AMT").alias("Max_Withdrawal_Amount")).show()

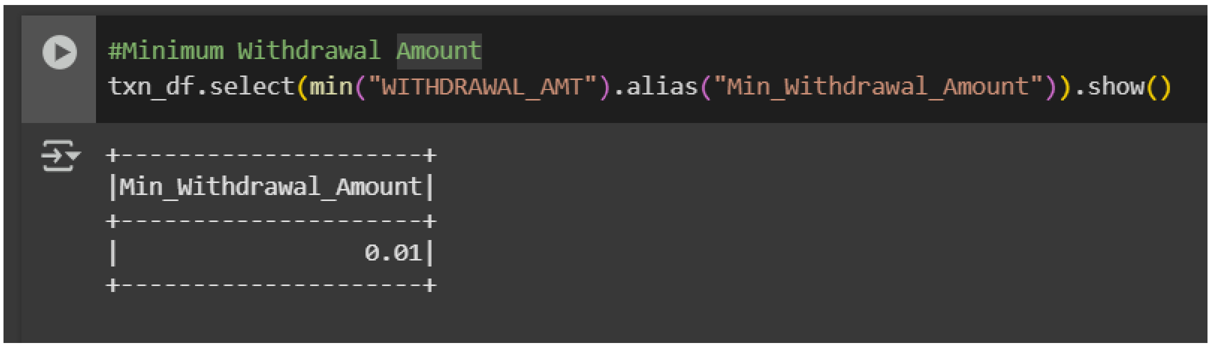


```
#1. Maximum Withdrawal Amount
txn_df.select(max("WITHDRAWAL_AMT").alias("Max_Withdrawal_Amount")).show()
```

```
+---------------------+
|Max_Withdrawal_Amount|
+---------------------+
|        4.594475464E8|
+---------------------+
```

2.minimum withdrawal amount of an account in txn.csv

txn_df.select(min("WITHDRAWAL_AMT").alias("Min_Withdrawal_Amount")).show()

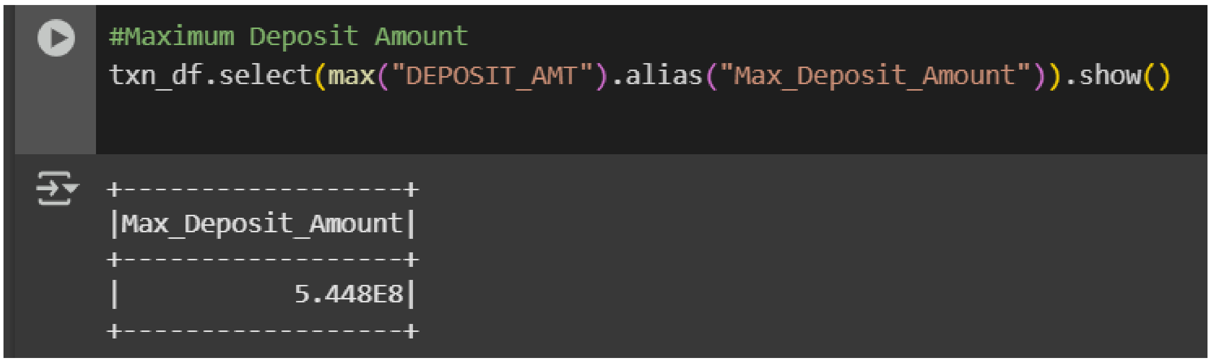


```
#Minimum Withdrawal Amount
txn_df.select(min("WITHDRAWAL_AMT").alias("Min_Withdrawal_Amount")).show()
```

```
+---------------------+
|Min_Withdrawal_Amount|
+---------------------+
|                 0.01|
+---------------------+
```

3.maximum deposit amount of an account

txn_df.select(max("DEPOSIT_AMT").alias("Max_Deposit_Amount")).show()

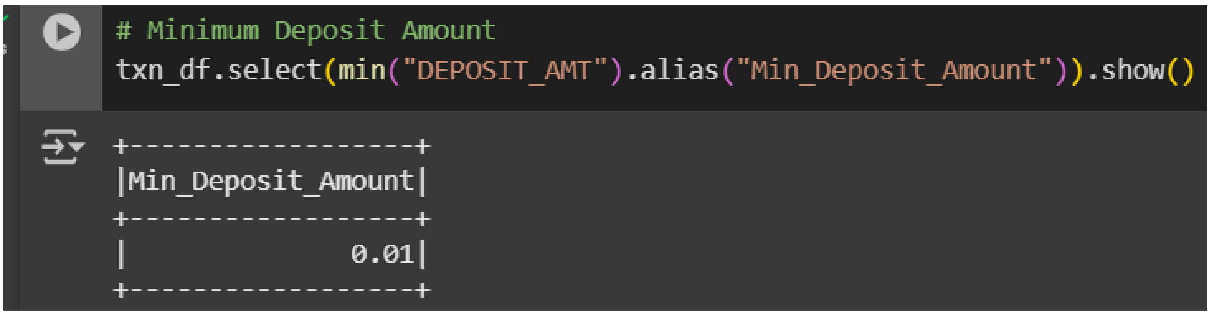


```
#Maximum Deposit Amount
txn_df.select(max("DEPOSIT_AMT").alias("Max_Deposit_Amount")).show()
```

```
+------------------+
|Max_Deposit_Amount|
+------------------+
|           5.448E8|
+------------------+
```

4.minimum deposit amount of an account

txn_df.select(min("DEPOSIT_AMT").alias("Min_Deposit_Amount")).show()



```
# Minimum Deposit Amount
txn_df.select(min("DEPOSIT_AMT").alias("Min_Deposit_Amount")).show()
```

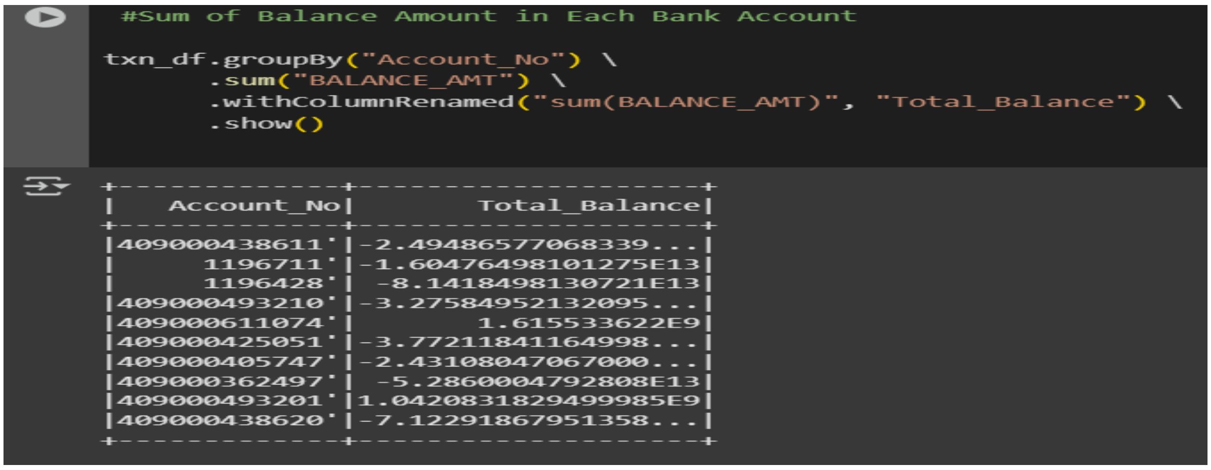
```
+------------------+
|Min_Deposit_Amount|
+------------------+
|              0.01|
+------------------+
```

5.sum of balance in every bank account

txn_df.groupBy("Account_No") \

.sum("BALANCE_AMT") \

.withColumnRenamed("sum(BALANCE_AMT)", "Total_Balance") \.show()



```
#Sum of Balance Amount in Each Bank Account

txn_df.groupBy("Account_No") \
       .sum("BALANCE_AMT") \
       .withColumnRenamed("sum(BALANCE_AMT)", "Total_Balance") \
       .show()
```

```
+---------------+--------------------+
|     Account_No|       Total_Balance|
+---------------+--------------------+
|409000438611'|-2.49486577068339...|
|       1196711'|-1.60476498101275E13|
|       1196428'|  -8.1418498130721E13|
|409000493210'|-3.27584952132095...|
|409000611074'|         1.615533622E9|
|409000425051'|-3.77211841164998...|
|409000405747'|-2.43108047067000...|
|409000362497'|  -5.2860004792808E13|
|409000493201'|1.0420831829499985E9|
|409000438620'|-7.12291867951358...|
+---------------+--------------------+
```

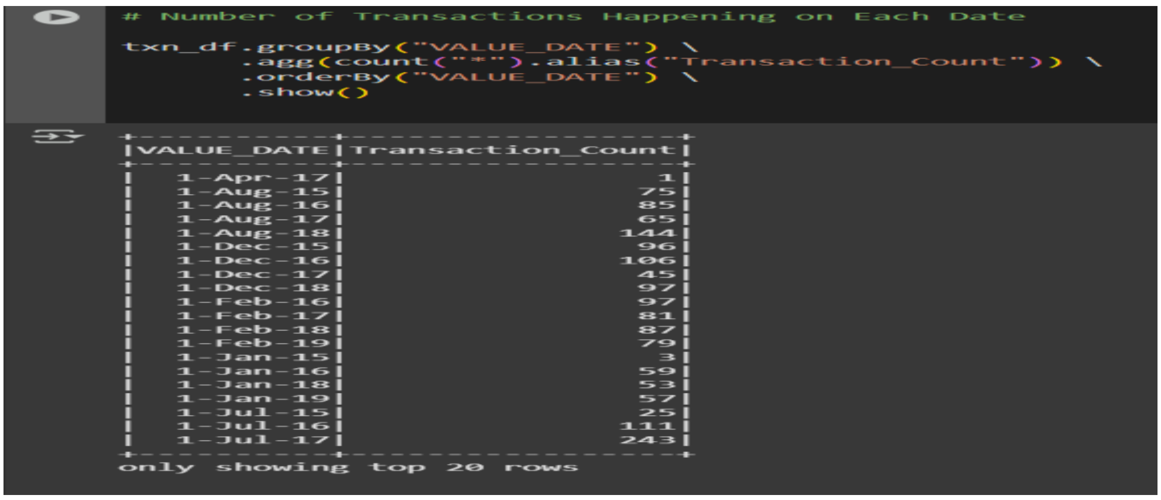
6.Number of transaction on each date

txn_df.groupBy("VALUE_DATE") \

.agg(count("*").alias("Transaction_Count")) \

.orderBy("VALUE_DATE") \

.show()



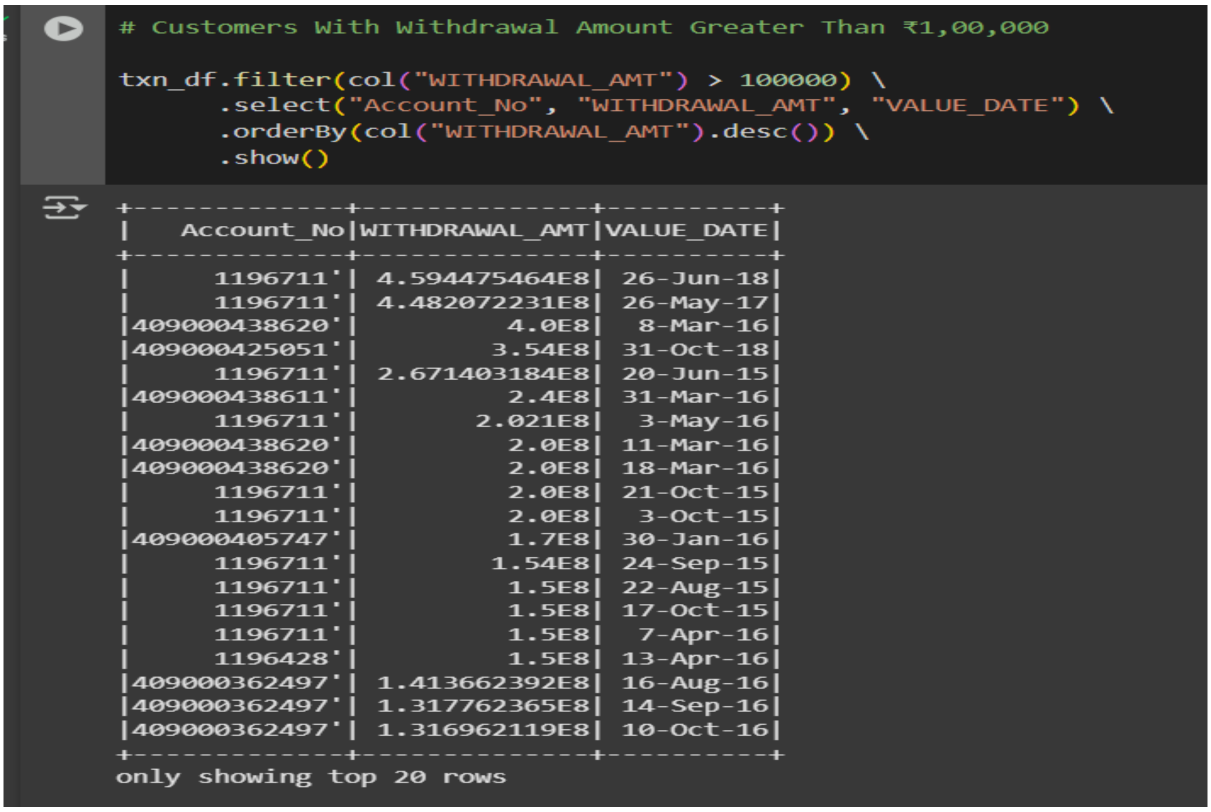
```
# Number of Transactions Happening on Each Date

txn_df.groupBy("VALUE_DATE") \
        .agg(count("*").alias("Transaction_Count")) \
        .orderBy("VALUE_DATE") \
        .show()
```

```
+----------+-----------------+
|VALUE_DATE|Transaction_Count|
+----------+-----------------+
|  1-Apr-17|                1|
|  1-Aug-15|               75|
|  1-Aug-16|               85|
|  1-Aug-17|               65|
|  1-Aug-18|              144|
|  1-Dec-15|               96|
|  1-Dec-16|              106|
|  1-Dec-17|               45|
|  1-Dec-18|               97|
|  1-Feb-16|               97|
|  1-Feb-17|               81|
|  1-Feb-18|               87|
|  1-Feb-19|               79|
|  1-Jan-15|                3|
|  1-Jan-16|               59|
|  1-Jan-18|               53|
|  1-Jan-19|               57|
|  1-Jul-15|               25|
|  1-Jul-16|              111|
|  1-Jul-17|              243|
+----------+-----------------+
only showing top 20 rows
```

7.List of customers with withdrawal amount more than 1 lakh

txn_df.filter(col("WITHDRAWAL_AMT") > 100000) \
    .select("Account_No", "WITHDRAWAL_AMT", "VALUE_DATE") \
    .orderBy(col("WITHDRAWAL_AMT").desc()) \
    .show()



```
# Customers With Withdrawal Amount Greater Than ₹1,00,000

txn_df.filter(col("WITHDRAWAL_AMT") > 100000) \
        .select("Account_No", "WITHDRAWAL_AMT", "VALUE_DATE") \
        .orderBy(col("WITHDRAWAL_AMT").desc()) \
        .show()
```

```
+-------------+--------------+----------+
|   Account_No|WITHDRAWAL_AMT|VALUE_DATE|
+-------------+--------------+----------+
|     1196711'| 4.594475464E8| 26-Jun-18|
|     1196711'| 4.482072231E8| 26-May-17|
|409000438620'|         4.0E8|  8-Mar-16|
|409000425051'|        3.54E8| 31-Oct-18|
|     1196711'| 2.671403184E8| 20-Jun-15|
|409000438611'|         2.4E8| 31-Mar-16|
|     1196711'|       2.021E8|  3-May-16|
|409000438620'|         2.0E8| 11-Mar-16|
|409000438620'|         2.0E8| 18-Mar-16|
|     1196711'|         2.0E8| 21-Oct-15|
|     1196711'|         2.0E8|  3-Oct-15|
|409000405747'|         1.7E8| 30-Jan-16|
|     1196711'|        1.54E8| 24-Sep-15|
|     1196711'|         1.5E8| 22-Aug-15|
|     1196711'|         1.5E8| 17-Oct-15|
|     1196711'|         1.5E8|  7-Apr-16|
|     1196428'|         1.5E8| 13-Apr-16|
|409000362497'| 1.413662392E8| 16-Aug-16|
|409000362497'| 1.317762365E8| 14-Sep-16|
|409000362497'| 1.316962119E8| 10-Oct-16|
+-------------+--------------+----------+
only showing top 20 rows
```