CASE STUDY PYSPARK (2-08-2025)

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, count, max as max\_, min as min\_, sum as sum\_

# Start Spark session

spark = SparkSession.builder.appName("Online Banking Analysis").getOrCreate()

# Load datasets

loan\_df = spark.read.csv("/content/loan.csv", header=True, inferSchema=True)

credit\_df = spark.read.csv("/content/credit card.csv", header=True, inferSchema=True)

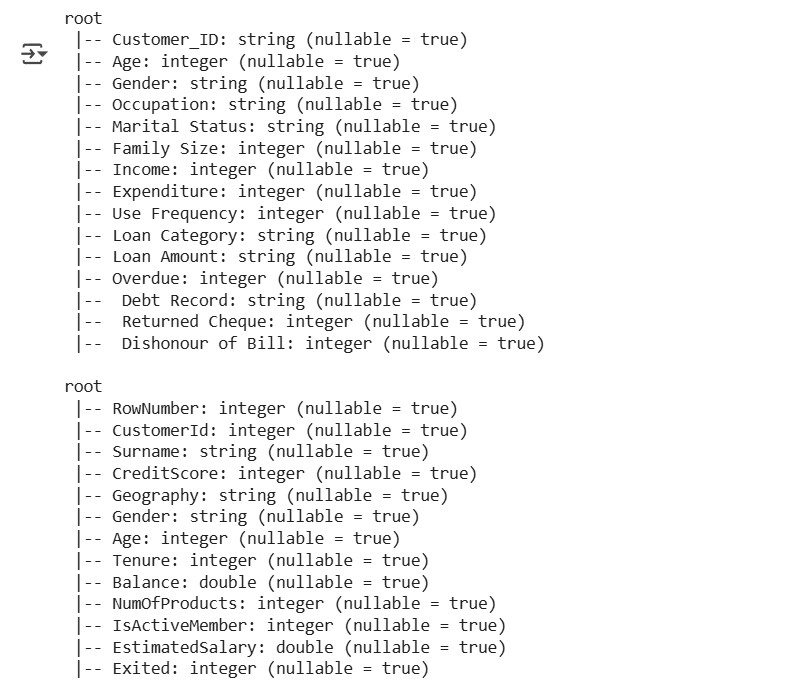
txn\_df = spark.read.csv("/content/txn.csv", header=True, inferSchema=True)

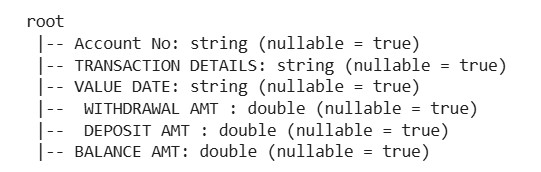
# Show schema just to verify

loan\_df.printSchema()

credit\_df.printSchema()

txn\_df.printSchema()



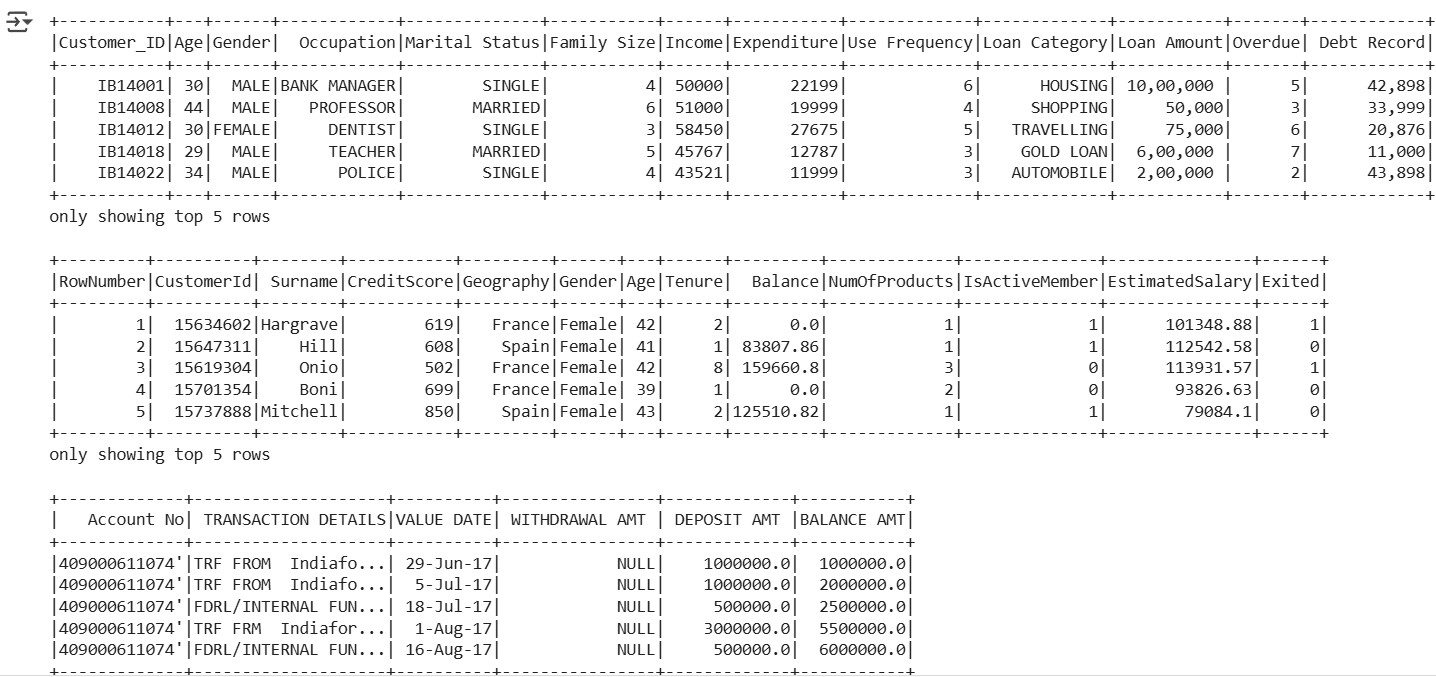


#show tables

loan\_df.show(5)

credit\_df.show(5)

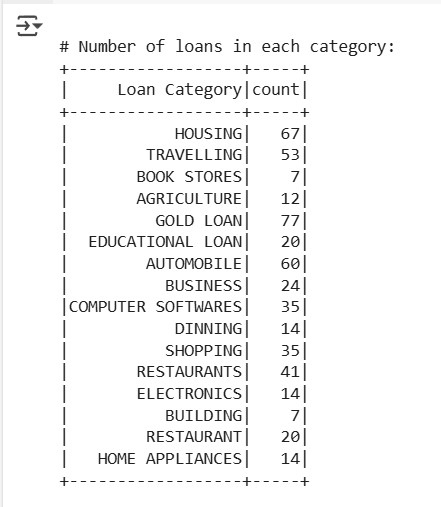
txn\_df.show(5)



#Loan data use cases

print("\n# Number of loans in each category:")

loan\_df.groupBy("Loan\_Category").count().show()



print("\n# Number of people who have taken more than 1 lakh loan:")

loan\_df.filter(col("Loan\_Amount") > 100000).select("Customer\_ID").distinct().count()

output:

# Number of people who have taken more than 1 lakh loan:

0

print("\n# Number of people with income greater than 60000 rupees:")

loan\_df.filter(col("Income") > 60000).select("Customer\_ID").distinct().count()

output:

# Number of people with income greater than 60000 rupees:

198

print("\n# Number of people with 2 or more returned cheques and income less than 50000:")

loan\_df.filter((col("Returned\_Cheques") >= 2) & (col("Income") < 50000)).select("Customer\_ID").distinct().count()

output

# Number of people with 2 or more returned cheques and income less than 50000:

136

print("\n# Number of people with 2 or more returned cheques and are single:")

loan\_df.filter((col("Returned\_Cheques") >= 2) & (col("Marital\_Status") == "Single")).select("Customer\_ID").distinct().count()

output:

# Number of people with 2 or more returned cheques and are single: 0

print("\n# Number of people with expenditure over 50000 a month:")

loan\_df.filter(col("Monthly\_Expenditure") > 50000).select("Customer\_ID").distinct().count()

output:

# Number of people with expenditure over 50000 a month:

6

print("\n# Number of members who are eligible for credit card:")

loan\_df.filter(col("Eligible\_for\_CreditCard") == "Yes").select("Customer\_ID").distinct().count()

output:

# Number of members likely eligible for credit card (Income > 50k, Returned Cheque = 0):

22

# CREDIT CARD DATA USE CASES

print("\n# Credit card users in Spain:")

credit\_df.filter(col("Country") == "Spain").select("Customer\_ID").distinct().count()

output:

# Credit card users in Spain:

2477

print("\n# Number of members who are eligible and active in the bank:")

credit\_df.filter((col("Eligible") == "Yes") & (col("Status") == "Active")).select("Customer\_ID").distinct().count()

output:

# Number of members who are likely eligible (CreditScore > 650) and active in the bank:

2655

#TRANSACTION DATA USE CASES

from pyspark.sql.functions import col, max as max\_, min as min\_, sum as sum\_, to\_date

# Step 1: Clean all column names (remove leading/trailing spaces and replace internal spaces with "\_")

txn\_df = txn\_df.toDF(\*[c.strip().replace(" ", "\_") for c in txn\_df.columns])

# Step 2: Cast numeric fields and parse date

txn\_df = txn\_df.withColumn("WITHDRAWAL\_AMT", col("WITHDRAWAL\_AMT").cast("double")) \

.withColumn("DEPOSIT\_AMT", col("DEPOSIT\_AMT").cast("double")) \

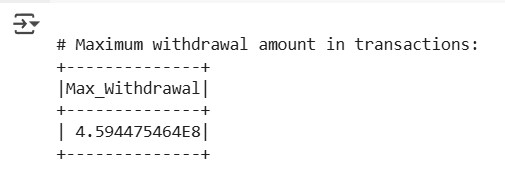
.withColumn("BALANCE\_AMT", col("BALANCE\_AMT").cast("double")) \

.withColumn("VALUE\_DATE", to\_date("VALUE\_DATE", "dd-MM-yyyy"))

print("\n# Maximum withdrawal amount in transactions:")

txn\_df.select(max\_("WITHDRAWAL\_AMT").alias("Max\_Withdrawal")).show()

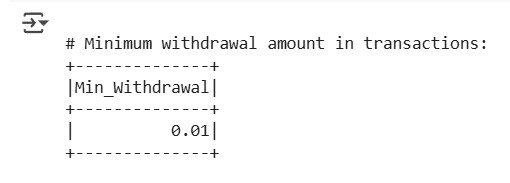
output:



print("\n# Minimum withdrawal amount in transactions:")

txn\_df.select(min\_("WITHDRAWAL\_AMT").alias("Min\_Withdrawal")).show()

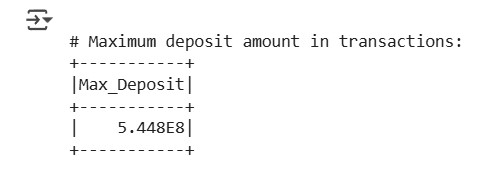
output:



print("\n# Maximum deposit amount in transactions:")

txn\_df.select(max\_("DEPOSIT\_AMT").alias("Max\_Deposit")).show()

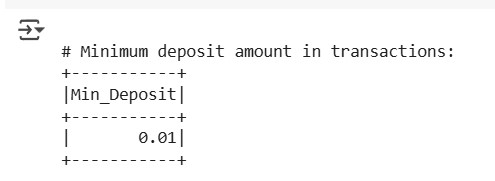
output:



print("\n# Minimum deposit amount in transactions:")

txn\_df.select(min\_("DEPOSIT\_AMT").alias("Min\_Deposit")).show()

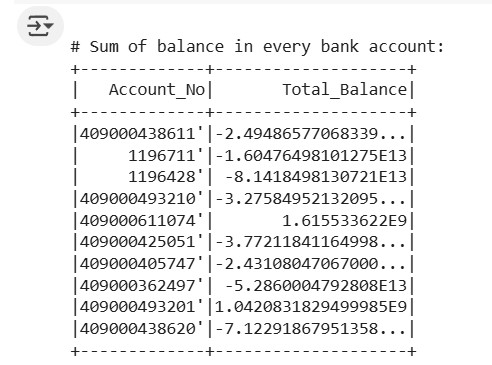
output:



print("\n# Sum of balance in every bank account:")

txn\_df.groupBy("Account\_No").agg(sum\_("BALANCE\_AMT").alias("Total\_Balance")).show()

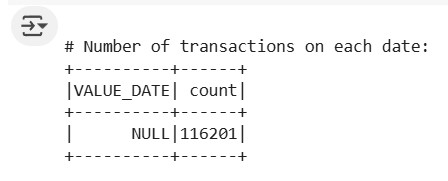
output:



print("\n# Number of transactions on each date:")

txn\_df.groupBy("VALUE\_DATE").count().orderBy("VALUE\_DATE").show()

output:



print("\n# List of accounts with withdrawal amount more than 1 lakh:")

txn\_df.filter(col("WITHDRAWAL\_AMT") > 100000) \

.select("Account\_No", "WITHDRAWAL\_AMT").distinct().show()

Output:

