**PYSPARK CASE STUDY**

**Name: Vikas Reddy Gorantla**

**Date: 25-11-2024**

**Loading Loan.csv File:**

# File location and type

file\_location = "/FileStore/tables/loan.csv"

file\_type = "csv"

# CSV options

infer\_schema = "false"

first\_row\_is\_header = "true"

delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.

df = spark.read.format(file\_type) \

  .option("inferSchema", infer\_schema) \

  .option("header", first\_row\_is\_header) \

  .option("sep", delimiter) \

  .load(file\_location)

display(df)

df.createOrReplaceTempView("loan")

**A screenshot of a computer

Description automatically generated**

**1. Number of loans in each category**

loan\_category\_count = df.groupBy("Loan Category").count()

loan\_category\_count.show()

A screenshot of a computer

Description automatically generated

**2. Number of people who have taken more than 1 lakh loan**

people\_over\_1lakh\_loan = df.filter(df["Loan Amount"] > '100000').count()

print("Number of people who have taken more than 1 lakh loan:", people\_over\_1lakh\_loan)

**# 2. Number of people who have taken more than 1 lakh loan -- with where**

people\_over\_1lakh\_loan = df.where(df["Loan Amount"] > '100000').count()

print("Number of people who have taken more than 1 lakh loan:", people\_over\_1lakh\_loan)

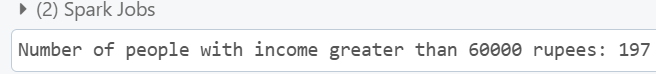
A screenshot of a phone

Description automatically generated

**3. Number of people with income greater than 60000 rupees**

people\_income\_above\_60000 = df.filter(df["Income"] > '60000').count()

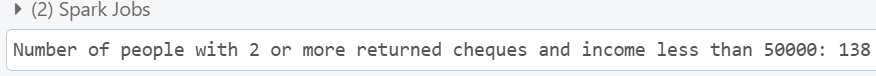
print("Number of people with income greater than 60000 rupees:", people\_income\_above\_60000)



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

people\_with\_2\_or\_more\_cheques\_and\_low\_income = df.filter((df[" Returned Cheque"] >= '2') & (df["Income"] < '50000')).count()

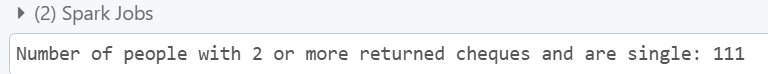
print("Number of people with 2 or more returned cheques and income less than 50000:", people\_with\_2\_or\_more\_cheques\_and\_low\_income)



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

people\_with\_2\_or\_more\_cheques\_and\_single = df.filter((df[" Returned Cheque"] >= '2') & (df["Marital Status"] == "SINGLE")).count()

print("Number of people with 2 or more returned cheques and are single:", people\_with\_2\_or\_more\_cheques\_and\_single)



**6. Number of people with expenditure over 50000 a month**

people\_with\_high\_expenditure = df.filter(df["Expenditure"] > '50000').count()

print("Number of people with expenditure over 50000 a month:", people\_with\_high\_expenditure)



**Loading Credit\_card.csv File:**

# File location and type for 'credit card' table

file\_location = "/FileStore/tables/credit\_card.csv"

file\_type = "csv"

# CSV options

infer\_schema = "false"

first\_row\_is\_header = "true"

delimiter = ","

# Read the CSV file

credit\_card\_df = spark.read.format(file\_type) \

  .option("inferSchema", infer\_schema) \

  .option("header", first\_row\_is\_header) \

  .option("sep", delimiter) \

  .load(file\_location)

# Display the DataFrame to verify

display(credit\_card\_df)

# Create a temporary SQL view for 'credit card'

credit\_card\_df.createOrReplaceTempView("credit\_card")

A screenshot of a computer

Description automatically generated

**1. Credit Card Users in Spain**

credit\_card\_users\_spain = credit\_card\_df.filter(credit\_card\_df["Geography"] == "Spain").count()

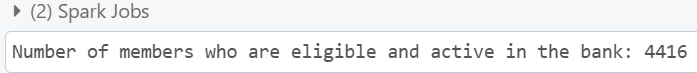
print("Number of credit card users in Spain:", credit\_card\_users\_spain)



**2. Number of members who are eligible and active in the bank**

eligible\_and\_active\_members = credit\_card\_df.filter((credit\_card\_df["IsActiveMember"] == '1') & (credit\_card\_df["Exited"] == '0')).count()

print("Number of members who are eligible and active in the bank:", eligible\_and\_active\_members)



**Loading Txn.csv File:**

# File location and type for 'txn' table

file\_location = "/FileStore/tables/txn.csv"

file\_type = "csv"

# CSV options

infer\_schema = "false"

first\_row\_is\_header = "true"

delimiter = ","

# Read the CSV file

txn\_df = spark.read.format(file\_type) \

  .option("inferSchema", infer\_schema) \

  .option("header", first\_row\_is\_header) \

  .option("sep", delimiter) \

  .load(file\_location)

# Display the DataFrame to verify

display(txn\_df)

# Create a temporary SQL view for 'txn'

txn\_df.createOrReplaceTempView("txn")

A screenshot of a computer

Description automatically generated

**1. Maximum Withdrawal Amount in Transactions**

max\_withdrawal\_amount = txn\_df.agg({" WITHDRAWAL AMT ": "max"}).collect()[0][0]

print("Maximum withdrawal amount in transactions:", max\_withdrawal\_amount)

A screenshot of a computer

Description automatically generated

**2. Minimum Withdrawal Amount of an Account**

min\_withdrawal\_amount = txn\_df.agg({" WITHDRAWAL AMT ": "min"}).collect()[0][0]

print("Minimum withdrawal amount of an account:", min\_withdrawal\_amount)



**3. Maximum Deposit Amount of an Account**

max\_deposit\_amount = txn\_df.agg({" DEPOSIT AMT ": "max"}).collect()[0][0]

print("Maximum deposit amount of an account:", max\_deposit\_amount)



**4. Minimum Deposit Amount of an Account**

min\_deposit\_amount = txn\_df.agg({" DEPOSIT AMT ": "min"}).collect()[0][0]

print("Minimum deposit amount of an account:", min\_deposit\_amount)



**6. Number of transactions on each date**

transactions\_by\_date = txn\_df.groupBy("VALUE DATE").count()

transactions\_by\_date.show()

A screenshot of a computer

Description automatically generated

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

customers\_with\_high\_withdrawal = txn\_df.filter(txn\_df[" WITHDRAWAL AMT "] > 100000).select("Account No").distinct()

customers\_with\_high\_withdrawal.show()

A screenshot of a computer

Description automatically generated