**Pyspark Case Study**

**Name: PRIYESHWAR**

**Mail:** [**priyesh2664@gmail.com**](mailto:priyesh2664@gmail.com)

**ONLINE BANKING ANALYSIS**

This is the first project where we worked on apache spark, In this project what we have done is

that we downloaded the datasets from KAGGLE where everyone is aware of, we have

downloaded loan, customers credit card and transactions datasets . After downloading the

datsaets we have cleaned the data . Then after by using new tools and technologies like spark,

HDFS, Hive and many more we have executed new use cases on the datasets, that we have

downloaded from kaggle. As we all know apache spark is a framework that can quickly process

the large datsets.

So now let me explain the dataflow of how we have done is, first primarly we have ingested the

data that is , we retrieved the data and then downloaded the datasets from kaggle and then we

stored this datasets in cloud storage and imported from MYSQL to hive by sqoop this is how we

have ingested the data , second after ingesting the data we have processed the large datasets

in hive and then we have analyzed the data using pyspark in jupyter notebook by implementing

several use cases.

**In loandata.csv file**

**from pyspark import SparkContext**

**from pyspark.sql import SparkSession**

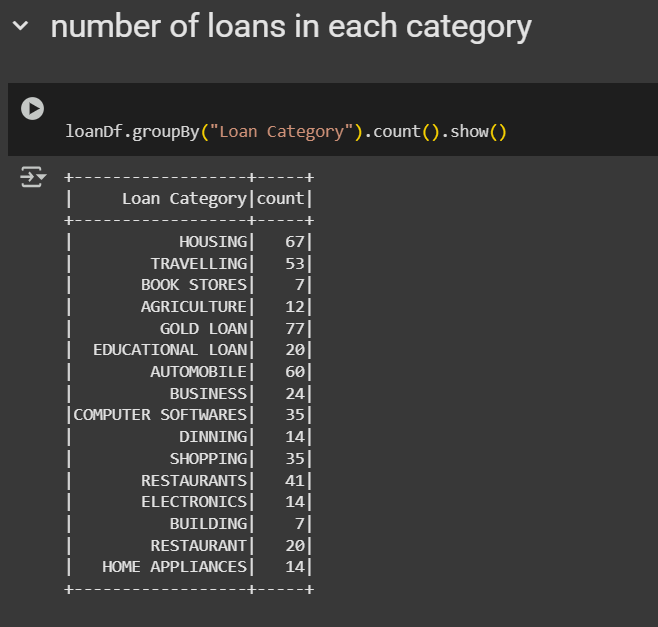
**spark=SparkSession.builder.appName("Case\_Study\_Spark").getOrCreate()**

**loanDf=spark.read.csv("loan.csv",header=True,inferSchema=True)**

***Number Of Loans In Each Category***

Groups loans by category (e.g., personal, home) and counts how many fall into each type.

**loanDf.groupBy("Loan Category").count().show()**

****

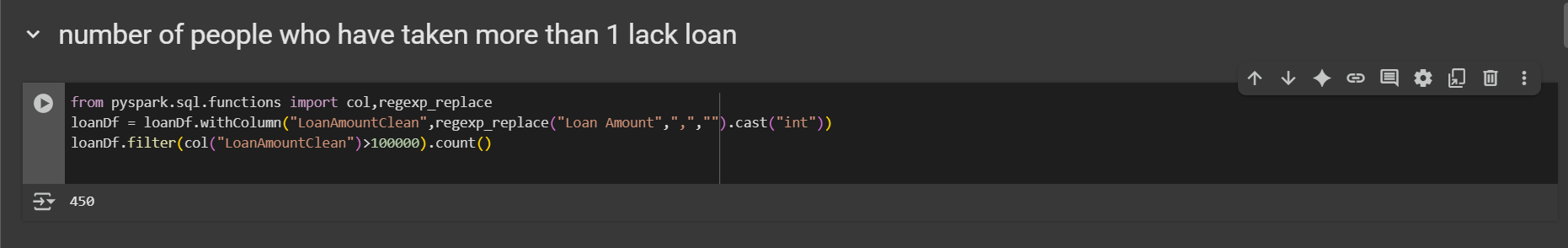
***Number Of People Who Have Taken More Than 1 Lack Loan***

Filters and counts customers who took a loan greater than ₹1 lakh.

**from pyspark.sql.functions import col,regexp\_replace**

**loanDf = loanDf.withColumn("LoanAmountClean",regexp\_replace("Loan Amount",",","").cast("int"))**

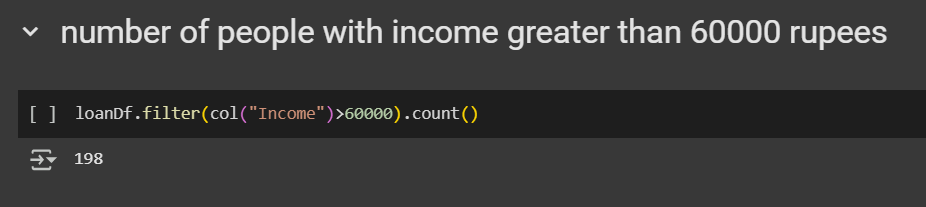
**loanDf.filter(col("LoanAmountClean")>100000).count()**



***Number Of People With Income Greater Than 60000 Rupees***

Counts how many people have a monthly income above ₹60,000.

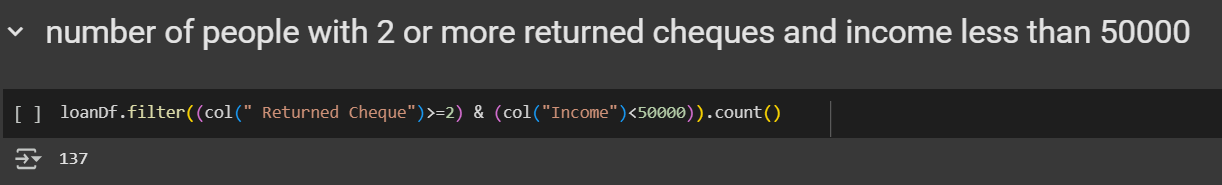
**loanDf.filter(col("Income")>60000).count()**

****

***Number Of People With 2 Or More Returned Cheques And Income Less Than 50000***

Identifies customers with multiple returned cheques and low income—indicating higher financial risk.

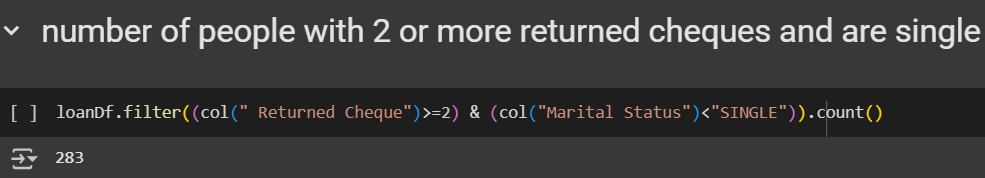
**loanDf.filter((col(" Returned Cheque")>=2) & (col("Income")<50000)).count()**



***Number Of People With 2 Or More Returned Cheques And Are Single***

Filters customers with at least two returned cheques and marked as "Single" in marital status.

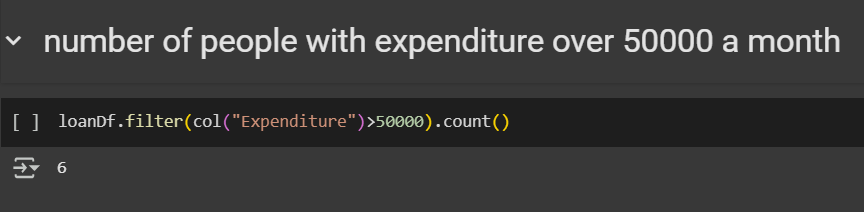
**loanDf.filter((col(" Returned Cheque")>=2) & (col("Marital Status")<"SINGLE")).count()**



***Number Of People With Expenditure Over 50000 A Month***

Counts customers who spend more than ₹50,000 monthly.

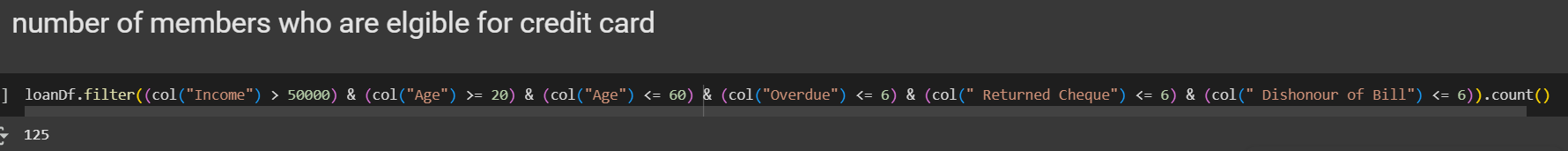
**loanDf.filter(col("Expenditure")>50000).count()**

****

***Number Of Members Who Are Elgible For Credit Card***

Filters individuals eligible for a credit card based on income, age, and financial reliability criteria.

**loanDf.filter((col("Income") > 50000) & (col("Age") >= 20) & (col("Age") <= 60) & (col("Overdue") <= 6) & (col(" Returned Cheque") <= 6) & (col(" Dishonour of Bill") <= 6)).count()**

****

**In credit.csv file**

**creditDf=spark.read.csv("credit card.csv",header=True,inferSchema=True)**

**creditDf.show()**

***Credit Card Users In Spain***

Displays basic details of customers located in Spain.

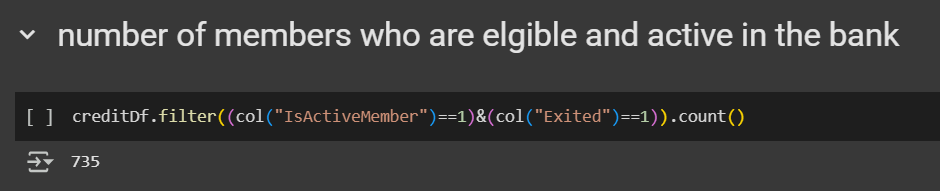
**creditDf.filter(col("Geography")=="Spain").select("CustomerId","Surname","Age","Gender").show()**

****

***Number Of Members Who Are Elgible And Active In The Bank***

Counts users who were active but still chose to leave the bank.

**creditDf.filter((col("IsActiveMember")==1)&(col("Exited")==1)).count()**

****

**In Transactions file**

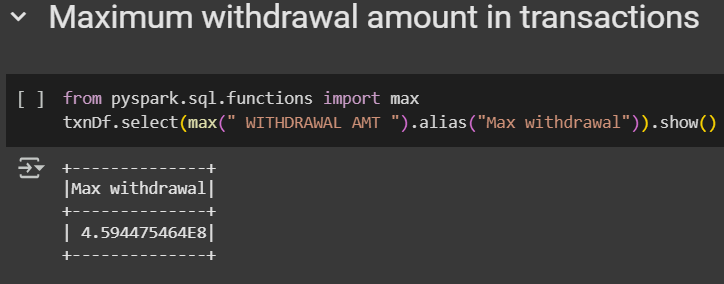
**txnDf=spark.read.csv("txn.csv",header=True,inferSchema=True)**

***Maximum Withdrawal Amount In Transactions***

Finds the largest withdrawal amount recorded in transactions.

**from pyspark.sql.functions import max**

**txnDf.select(max(" WITHDRAWAL AMT ").alias("Max withdrawal")).show()**

****

***MINIMUM WITHDRAWAL AMOUNT OF AN ACCOUNT In Txn.Csv***

Finds the smallest withdrawal value from all transactions.

**from pyspark.sql.functions import min**

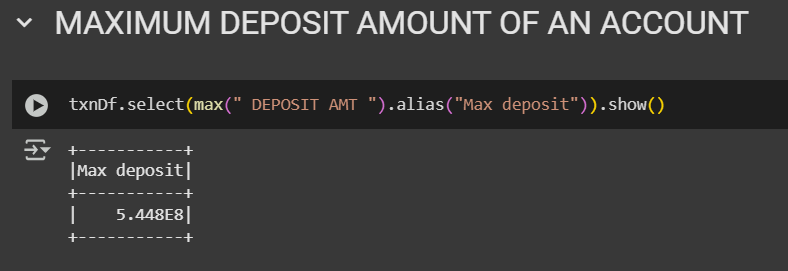
**txnDf.select(min(" WITHDRAWAL AMT ").alias("Min withdrawal")).show()**

****

***MAXIMUM DEPOSIT AMOUNT OF AN ACCOUNT***

Retrieves the highest amounts deposited into accounts.

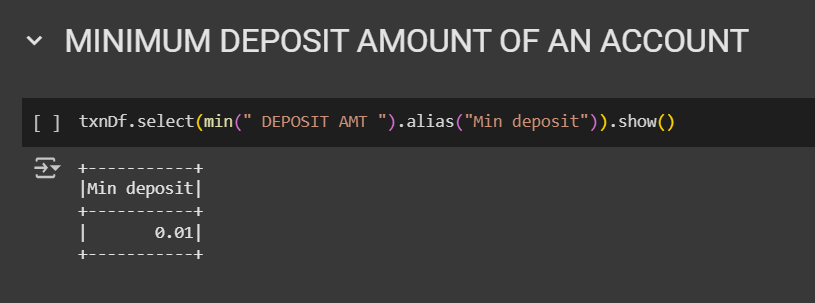
**txnDf.select(max(" DEPOSIT AMT ").alias("Max deposit")).show()**

****

***MINIMUM DEPOSIT AMOUNT OF AN ACCOUNT***

Retrieves the lowest amounts deposited into accounts.

**txnDf.select(min(" DEPOSIT AMT ").alias("Min deposit")).show()**

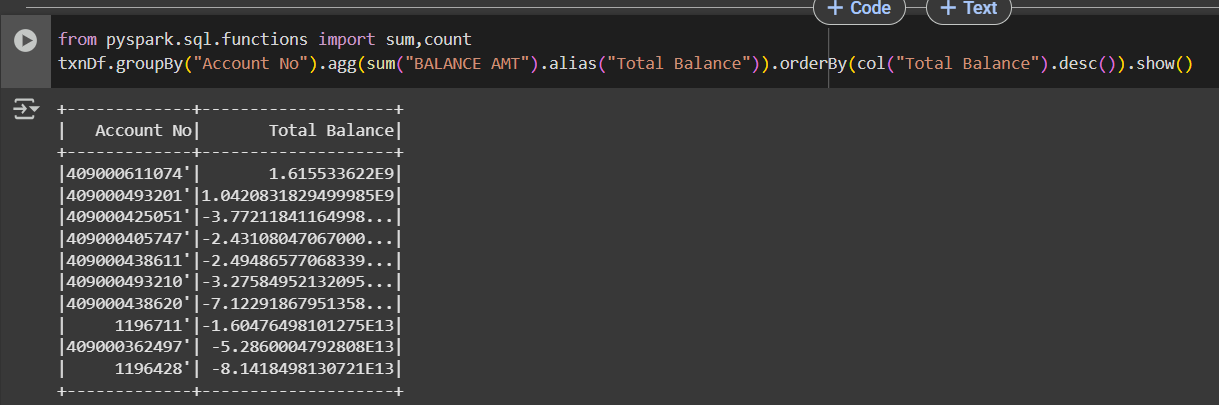
****

***Sum Of Balance In Every Bank Account***

Sums the total balance of each account and sorts them in descending order.

**from pyspark.sql.functions import sum,count**

**txnDf.groupBy("Account No").agg(sum("BALANCE AMT").alias("Total Balance")).orderBy(col("Total Balance").desc()).show()**

****

***Number Of Transaction On Each Date***

Counts how many transactions happened on each date.

**txnDf.groupBy("VALUE DATE").agg(count("TRANSACTION DETAILS").alias("Total Transaction")).orderBy(col("Total Transaction").desc()).show()**

****

***List Of Customers With Withdrawal Amount More Than 1 Lakh***

Lists all transactions where the withdrawal amount exceeded ₹1 lakh.

**txnDf.filter(col(" WITHDRAWAL AMT ")>100000).show()**

