

Pyspark Case Study

Name: PRIYESHWAR

Mail: 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 datasets 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 datasets.

So now let me explain the dataflow of how we have done is, first primarily 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 loans in each category



```
loanDf.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

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 who have taken more than 1 lack loan

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 income greater than 60000 rupees

[ ] 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 income less than 50000

[ ] 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 2 or more returned cheques and are single

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

↔ 283
```

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 people with expenditure over 50000 a month

[ ] loanDf.filter(col("Expenditure")>50000).count()

↔ 6
```

Number Of Members Who Are Eligible 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()
```

```
number of members who are eligible for credit card

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

↔ 125
```

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()
```

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



CustomerId	Surname	Age	Gender
15647311	Hill	41	Female
15737888	Mitchell	43	Female
15574012	Chu	44	Male
15737173	Andrews	24	Male
15600882	Scott	35	Female
15788218	Henderson	24	Female
15661507	Muldrow	45	Male
15597945	Dellucci	32	Female
15699309	Gerasimov	38	Female
15589475	Azikiwe	39	Female
15659428	Maggard	42	Female
15732963	Clements	29	Female
15788448	Watson	31	Male
15729599	Lorenzo	33	Male
15619360	Hsiao	40	Male
15684171	Bianchi	61	Female
15623944	T'ien	66	Female
15702014	Jeffrey	33	Male
15751208	Pirozzi	56	Male
15812518	Palermo	37	Female

only showing top 20 rows

Number Of Members Who Are Eligible 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()
```

✓ number of members who are eligible and active in the bank

```
[ ] creditDf.filter((col("IsActiveMember")==1)&(col("Exited")==1)).count()
```

735

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()
```

✓ Maximum withdrawal amount in transactions

```
[ ] from pyspark.sql.functions import max
    txnDf.select(max(" WITHDRAWAL AMT ").alias("Max withdrawal")).show()
```

```
┌-----+
|Max withdrawal|
┌-----+
| 4.594475464E8|
┌-----+
```

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()
```

✓ MINIMUM WITHDRAWAL AMOUNT OF AN ACCOUNT in txn.csv

```
from pyspark.sql.functions import min
txnDf.select(min(" WITHDRAWAL AMT ").alias("Min withdrawal")).show()
```

```
↵ +-----+
  |Min withdrawal|
  +-----+
  |          0.01|
  +-----+
```

MAXIMUM DEPOSIT AMOUNT OF AN ACCOUNT

Retrieves the highest amounts deposited into accounts.

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

✓ MAXIMUM DEPOSIT AMOUNT OF AN ACCOUNT

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

```
↵ +-----+
  |Max deposit|
  +-----+
  |    5.448E8|
  +-----+
```

MINIMUM DEPOSIT AMOUNT OF AN ACCOUNT

Retrieves the lowest amounts deposited into accounts.

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

```
✓ MINIMUM DEPOSIT AMOUNT OF AN ACCOUNT
```

```
[ ] txnDf.select(min(" DEPOSIT AMT ").alias("Min deposit")).show()
```

```
⇓
```

Min deposit
0.01

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()
```

```
+ Code + Text
```

```
from pyspark.sql.functions import sum,count
txnDf.groupBy("Account No").agg(sum("BALANCE AMT").alias("Total Balance")).orderBy(col("Total Balance").desc()).show()
```

```
⇓
```

Account No	Total Balance
409000611074'	1.615533622E9
409000493201'	1.0420831829499985E9
409000425051'	-3.77211841164998...
409000405747'	-2.43108047067000...
409000438611'	-2.49486577068339...
409000493210'	-3.27584952132095...
409000438620'	-7.12291867951358...
1196711'	-1.60476498101275E13
409000362497'	-5.2860004792808E13
1196428'	-8.1418498130721E13

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()
```

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

VALUE DATE	Total Transaction
27-Jul-17	567
13-Aug-18	463
8-Nov-17	402
7-Oct-17	382
10-Jul-18	374
12-Dec-17	367
12-Sep-18	365
9-Aug-18	360
19-Sep-17	358
16-Mar-17	353
10-Sep-18	344
14-Jul-17	333
7-Mar-18	319
11-Oct-18	303
22-Aug-17	301
9-Jan-18	299
9-Oct-18	297
20-Apr-18	296
9-Jul-18	292
7-Apr-18	291

only showing top 20 rows

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()
```

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

Account No	TRANSACTION DETAILS	VALUE DATE	WITHDRAWAL AMT	DEPOSIT AMT	BALANCE AMT
409000611074	INDO GIBL Indiafo...	16-Aug-17	133900.0	NULL	8366100.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	195800.0	NULL	8147300.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	143800.0	NULL	7781600.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	331650.0	NULL	7449950.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	129000.0	NULL	7320950.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	230013.0	NULL	7090937.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	367900.0	NULL	6723037.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	108000.0	NULL	6615037.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	141000.0	NULL	6409237.0
409000611074	INDO GIBL Indiafo...	16-Aug-17	206000.0	NULL	5959817.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	242300.0	NULL	5350718.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	113250.0	NULL	5147368.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	206900.0	NULL	4887968.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	276000.0	NULL	4611968.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	171000.0	NULL	4440968.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	189800.0	NULL	4211068.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	271323.0	NULL	3788545.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	200600.0	NULL	3587945.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	176900.0	NULL	3411045.0
409000611074	INDO GIBL Indiafo...	6-Sep-17	150050.0	NULL	3260995.0

only showing top 20 rows