Data-Engineering PySpark Case-Study

J Jatin

ONLINE BANKING ANALYSIS

This is the first project where we worked on Apache spark, in this project what we have done is download the datasets from KAGGLE where everyone is aware of, we have downloaded loans, customers credit card and transactions datasets . After downloading the datasets, we have cleaned up the data . Then after 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, the 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.

A) In loandata.csv file

1: Load the Loan.csv file

```python
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, lit

# Initialize Spark session
spark = SparkSession.builder.appName("LoanDataAnalysis").getOrCreate()

# Load the CSV file
file_path = "/FileStore/tables/Loan/loan.csv"  # Replace with your actual file path
loan_df = spark.read.option("header", "true").csv(file_path, inferSchema=True)

# Display the schema and a sample of the data
loan_df.printSchema()
loan_df.show(5)
```

```
root
 |-- Customer_ID: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Marital Status: string (nullable = true)
 |-- Family Size: integer (nullable = true)
 |-- Income: integer (nullable = true)
 |-- Expenditure: integer (nullable = true)
 |-- Use Frequency: integer (nullable = true)
 |-- Loan Category: string (nullable = true)
 |-- Loan Amount: string (nullable = true)
 |-- Overdue: integer (nullable = true)
 |--  Debt Record: string (nullable = true)
 |--  Returned Cheque: integer (nullable = true)
 |--  Dishonour of Bill: integer (nullable = true)
```

```
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-------
----+-------+------------+----------------+------------------+
|Customer_ID|Age|Gender|  Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|Loan Category|Loan Am
ount|Overdue| Debt Record| Returned Cheque| Dishonour of Bill|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-------
----+-------+------------+----------------+------------------+
|    IB14001| 30|  MALE|BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|      HOUSING| 10,00,
000 |      5|      42,898|               6|                 9|
|    IB14008| 44|  MALE|   PROFESSOR|       MARRIED|          6| 51000|      19999|            4|     SHOPPING|     5
0,000|      3|      33,999|               1|                 5|
|    IB14012| 30|FEMALE|     DENTIST|        SINGLE|          3| 58450|      27675|            5|   TRAVELLING|     7
5,000|      6|      20,876|               3|                 1|
|    IB14018| 29|  MALE|     TEACHER|       MARRIED|          5| 45767|      12787|            3|    GOLD LOAN|  6,00,
000 |      7|      11,000|               0|                 4|
|    IB14022| 34|  MALE|      POLICE|        SINGLE|          4| 43521|      11999|            3|   AUTOMOBILE|  2,00,
000 |      2|      43,898|               1|                 2|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-------
----+-------+------------+----------------+------------------+
only showing top 5 rows
```

2: Number of loans in each category

07:28 PM (3s) 2

```
#Number of loans in each category
loan_category_count = loan_df.groupBy("Loan Category").count()
loan_category_count.show()
```

(2) Spark Jobs

loan_category_count: pyspark.sql.dataframe.DataFrame = [Loan Category: string, count: long]

```
+------------------+-----+
|     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|
+------------------+-----+
```

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

07:29 PM (1s) 3

```
#Number of people who have taken more than 1 lakh loan
high_loan_count = loan_df.filter(col("Loan Amount") > 100000).count()
print(f"Number of people who have taken more than 1 lakh loan: {high_loan_count}")
```

(2) Spark Jobs

```
Number of people who have taken more than 1 lakh loan: 0
```

1

4: Number of people with income greater than 60,000 rupees

```
07:30 PM (1s)    4

#Number of people with income greater than 60,000 rupees
high_income_count = loan_df.filter(col("Income") > 60000).count()
print(f"Number of people with income greater than 60,000 rupees: {high_income_count}")

(2) Spark Jobs
Number of people with income greater than 60,000 rupees: 198
```

5: Number of people with 2 or more returned cheques and income less than 50,000

```
07:32 PM (1s)    5

#Number of people with 2 or more returned cheques and income less than 50,000
cheques_and_income_count = loan_df.filter((col(" Returned Cheque") >= 2) & (col("Income") < 50000)).count()
print(f"Number of people with 2 or more returned cheques and income less than 50,000: {cheques_and_income_count}")

(2) Spark Jobs
Number of people with 2 or more returned cheques and income less than 50,000: 137
```

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

```
07:34 PM (1s)    6    Python

#Number of people with 2 or more returned cheques and are single
cheques_and_single_count = loan_df.filter((col(" Returned Cheque") >= 2) & (col("Marital Status") == "Single")).count
()
print(f"Number of people with 2 or more returned cheques and are single: {cheques_and_single_count}")

(2) Spark Jobs
Number of people with 2 or more returned cheques and are single: 0
```

7: Number of people with expenditure over 50,000 a month

```
07:49 PM (1s)    7

#Number of people with expenditure over 50,000 a month
high_expenditure_count = loan_df.filter(col("Expenditure") > 50000).count()
print(f"Number of people with expenditure over 50,000 a month: {high_expenditure_count}")

(2) Spark Jobs
Number of people with expenditure over 50,000 a month: 6
```

8: Number of members eligible for a credit card

Assuming the eligibility criteria for a credit card:

1. Age ≥ 18.
2. Income > ₹75,000.
3. Overdue < 2.

Just now (<1s) 8

```
#Number of members eligible for a credit card
credit_card_eligible = loan_df.filter(
    (col("Age") >= 21) &
    (col("Income") > 75000) &
    (col("Overdue") < 2)
).agg(count("*").alias("EligibleCreditCardCount"))

credit_card_eligible.show()
```

(2) Spark Jobs

credit_card_eligible: pyspark.sql.dataframe.DataFrame = [EligibleCreditCardCount: long]

```
+-----------------------+
|EligibleCreditCardCount|
+-----------------------+
|                     17|
+-----------------------+
```

B) In credit.csv file

1: Load the CSV file

```
07:56 PM (2s)                                   1

# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count

# Initialize Spark session
spark = SparkSession.builder.appName("CreditCardAnalysis").getOrCreate()

# Load the CSV file
file_path = "/FileStore/tables/credit_card.csv"  # Replace with your file path
credit_card_df = spark.read.option("header", "true").csv(file_path, inferSchema=True)

# Display the schema and a sample of the data
credit_card_df.printSchema()
credit_card_df.show(5)
```

```
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+--------------+--------
-------+------+
|RowNumber|CustomerId| Surname|CreditScore|Geography|Gender|Age|Tenure|  Balance|NumOfProducts|IsActiveMember|Estimate
dSalary|Exited|
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+--------------+--------
-------+------+
|        1|  15634602|Hargrave|        619|   France|Female| 42|     2|      0.0|            1|             1|      10
1348.88|     1|
|        2|  15647311|    Hill|        608|    Spain|Female| 41|     1| 83807.86|            1|             1|      11
2542.58|     0|
|        3|  15619304|    Onio|        502|   France|Female| 42|     8| 159660.8|            3|             0|      11
3931.57|     1|
|        4|  15701354|    Boni|        699|   France|Female| 39|     1|      0.0|            2|             0|       9
3826.63|     0|
|        5|  15737888|Mitchell|        850|    Spain|Female| 43|     2|125510.82|            1|             1|
79084.1|     0|
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+--------------+--------
-------+------+
only showing top 5 rows
```

2: Number of credit card users in Spain

```
07:58 PM (1s)                                   2

# Number of credit card users in Spain
spain_users_count = credit_card_df.filter(col("Geography") == "Spain").count()
print(f"Number of credit card users in Spain: {spain_users_count}")

(2) Spark Jobs

Number of credit card users in Spain: 2477
```

3: Number of members eligible for a credit card and active in the bank

08:02 PM (1s) 3

```python
# Number of members eligible for a credit card and active in the bank
eligible_and_active_count = credit_card_df.filter(
    (col("CreditScore") > 600) & (col("IsActiveMember") == 1)
).count()

print(f"Number of members who are eligible and active in the bank: {eligible_and_active_count}")
```

(2) Spark Jobs

```
Number of members who are eligible and active in the bank: 3639
```

+ Code + Text

C) In Transactions file

1: Load the Transactions CSV File

08:07 PM (2s) 1 Python

```python
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, max, min, sum, count

# Initialize Spark session
spark = SparkSession.builder.appName("TransactionsAnalysis").getOrCreate()

# Load the CSV file
file_path = "/FileStore/tables/txn.csv"  # Replace with your file path
txn_df = spark.read.option("header", "true").csv(file_path, inferSchema=True)

# Display the schema and a sample of the data
txn_df.printSchema()
txn_df.show(5)
```

(3) Spark Jobs

txn_df: pyspark.sql.dataframe.DataFrame = [Account No: string, TRANSACTION DETAILS: string ... 4 more fields]

```
root
 |-- Account No: string (nullable = true)
 |-- TRANSACTION DETAILS: string (nullable = true)
 |-- VALUE DATE: string (nullable = true)
 |--  WITHDRAWAL AMT : double (nullable = true)
 |--  DEPOSIT AMT : double (nullable = true)
 |-- BALANCE AMT: double (nullable = true)
```

```
+-------------+--------------------+----------+---------------+-------------+-----------+
|   Account No| TRANSACTION DETAILS|VALUE DATE| WITHDRAWAL AMT | DEPOSIT AMT |BALANCE AMT|
+-------------+--------------------+----------+---------------+-------------+-----------+
|409000611074'|TRF FROM  Indiafo...| 29-Jun-17|           null|    1000000.0|  1000000.0|
|409000611074'|TRF FROM  Indiafo...|  5-Jul-17|           null|    1000000.0|  2000000.0|
|409000611074'|FDRL/INTERNAL FUN...| 18-Jul-17|           null|     500000.0|  2500000.0|
|409000611074'|TRF FRM  Indiafor...|  1-Aug-17|           null|    3000000.0|  5500000.0|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|           null|     500000.0|  6000000.0|
+-------------+--------------------+----------+---------------+-------------+-----------+
only showing top 5 rows
```

2: Maximum Withdrawal Amount in Transactions

08:12 PM (1s) 2

```python
#Maximum Withdrawal Amount in Transactions
max_withdrawal = txn_df.agg(max(" WITHDRAWAL AMT ").alias("MaxWithdrawal")).collect()[0][0]
print(f"Maximum withdrawal amount: {max_withdrawal}")
```

(2) Spark Jobs

```
Maximum withdrawal amount: 459447546.4
```

3: Minimum Withdrawal Amount of an Account

08:13 PM (2s) 3 Python

```python
#Minimum Withdrawal Amount of an Account
min_withdrawal_per_account = txn_df.groupBy("Account No").agg(min(" WITHDRAWAL AMT ").alias("MinWithdrawal"))
min_withdrawal_per_account.show()
```

(2) Spark Jobs

min_withdrawal_per_account: pyspark.sql.dataframe.DataFrame = [Account No: string, MinWithdrawal: double]

```
+-------------+-------------+
|   Account No|MinWithdrawal|
+-------------+-------------+
|409000438611'|          0.2|
|     1196711'|         0.25|
|     1196428'|         0.25|
|409000493210'|         0.01|
|409000611074'|        120.0|
|409000425051'|         1.25|
|409000405747'|         21.0|
|409000493201'|          2.1|
|409000438620'|         0.34|
|409000362497'|         0.97|
+-------------+-------------+
```

4: Maximum Deposit Amount of an Account

```
#Maximum Deposit Amount of an Account
max_deposit_per_account = txn_df.groupBy("Account No").agg(max(" DEPOSIT AMT ").alias("MaxDeposit"))
max_deposit_per_account.show()
```

(2) Spark Jobs

max_deposit_per_account: pyspark.sql.dataframe.DataFrame = [Account No: string, MaxDeposit: double]

```
+-------------+------------+
|   Account No|  MaxDeposit|
+-------------+------------+
|409000438611'|   1.7025E8|
|     1196711'|      5.0E8|
|     1196428'|2.119594422E8|
|409000493210'|      1.5E7|
|409000611074'|  3000000.0|
|409000425051'|      1.5E7|
|409000405747'|    2.021E8|
|409000493201'|  1000000.0|
|409000438620'|    5.448E8|
|409000362497'|      2.0E8|
+-------------+------------+
```

5: Minimum Deposit Amount of an Account

▶ ✓ 08:14 PM (1s) 5

```
#Minimum Deposit Amount of an Account
min_deposit_per_account = txn_df.groupBy("Account No").agg(min(" DEPOSIT AMT ").alias("MinDeposit"))
min_deposit_per_account.show()
```

(2) Spark Jobs

min_deposit_per_account: pyspark.sql.dataframe.DataFrame = [Account No: string, MinDeposit: double]

```
+-------------+----------+
|   Account No|MinDeposit|
+-------------+----------+
|409000438611'|      0.03|
|     1196711'|      1.01|
|     1196428'|       1.0|
|409000493210'|      0.01|
|409000611074'|    1320.0|
|409000425051'|       1.0|
|409000405747'|     500.0|
|409000493201'|       0.9|
|409000438620'|      0.07|
|409000362497'|      0.03|
+-------------+----------+
```

6: Sum of Balance in Every Bank Account

```
08:15 PM (1s)                                6

#Sum of Balance in Every Bank Account
total_balance_per_account = txn_df.groupBy("Account No").agg(sum("BALANCE AMT").alias("TotalBalance"))
total_balance_per_account.show()
```

(2) Spark Jobs

total_balance_per_account: pyspark.sql.dataframe.DataFrame = [Account No: string, TotalBalance: double]

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

7: Number of Transactions on Each Date

```
Just now (1s)                                7                    Python

#Number of Transactions on Each Date
transactions_per_date = txn_df.groupBy("VALUE DATE").agg(count("*").alias("TransactionCount"))
transactions_per_date.show()
```

(2) Spark Jobs

transactions_per_date: pyspark.sql.dataframe.DataFrame = [VALUE DATE: string, TransactionCount: long]

```
+----------+----------------+
|VALUE DATE|TransactionCount|
+----------+----------------+
| 23-Dec-16|             143|
|  7-Feb-19|              98|
| 21-Jul-15|              80|
|  9-Sep-15|              91|
| 17-Jan-15|              16|
| 18-Nov-17|              53|
| 21-Feb-18|              77|
| 20-Mar-18|              71|
| 19-Apr-18|              71|
| 21-Jun-16|              97|
| 17-Oct-17|             101|
|  3-Jan-18|              70|
|  8-Jun-18|             223|
| 15-Dec-18|              62|
|  8-Aug-16|              97|
| 17-Dec-16|              74|
|  3-Sep-15|              83|
```

8: List of Customers with Withdrawal Amount More Than 1 Lakh

```
#List of Customers with Withdrawal Amount More Than 1 Lakh
high_withdrawals = txn_df.filter(col(" WITHDRAWAL AMT ") > 100000).select("Account No", " WITHDRAWAL AMT ")
high_withdrawals.show()
```

▸ (1) Spark Jobs

▸ high_withdrawals: pyspark.sql.dataframe.DataFrame = [Account No: string, WITHDRAWAL AMT : double]

```
+-------------+---------------+
|   Account No| WITHDRAWAL AMT |
+-------------+---------------+
|409000611074'|       133900.0|
|409000611074'|       195800.0|
|409000611074'|       143800.0|
|409000611074'|       331650.0|
|409000611074'|       129000.0|
|409000611074'|       230013.0|
|409000611074'|       367900.0|
|409000611074'|       108000.0|
|409000611074'|       141000.0|
|409000611074'|       206000.0|
|409000611074'|       242300.0|
|409000611074'|       113250.0|
|409000611074'|       206900.0|
|409000611074'|       276000.0|
|409000611074'|       171000.0|
|409000611074'|       189800.0|
|409000611074'|       271323.0|
|409000611074'|       200600.0|
```