

CASE STUDY - ONLINE BANKING ANALYSIS

Submitted By-
Subrat Shukla, DE-1

Loading Datasets:

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import col, count, sum, max, min
```


```
✓ [2] 1 spark = SparkSession.builder.appName("Online Banking Analysis").getOrCreate()
```

```
✓ [4] 1 # Load the CSV files
13s 2 loan_df = spark.read.csv("loan.csv", header=True, inferSchema=True)
3 credit_df = spark.read.csv("credit card.csv", header=True, inferSchema=True)
4 txn_df = spark.read.csv("txn.csv", header=True, inferSchema=True)
5
```

• In Loan Data CSV file-

1. Number of loans in each category

```
✓ 2s 1 loan_df.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

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

```
✓ [6] 1 loan_df.filter(col("Loan Amount") > 100000).count()
```

0s
⇒ 0

3. Number of people with income greater than 60000 rupees

```
✓ [7] 1 loan_df.filter(col("Income") > 60000).count()
```

2

⇒ 198

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

```
✓ [9] 1 cheques_and_income_count = loan_df.filter((col("Returned Cheque") >= 2) & (col("Income") < 50000)).count()  
2 print(f"Number of people with 2 or more returned cheques and income less than 50,000: {cheques_and_income_count}")  
3
```

⇒ Number of people with 2 or more returned cheques and income less than 50,000: 137

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

```
✓ [10] 1 cheques_and_single_count = loan_df.filter((col("Returned Cheque") >= 2) & (col("Marital Status") == "Single")).count()  
2 print(f"Number of people with 2 or more returned cheques and are single: {cheques_and_single_count}")  
3
```

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

6. Number of people with expenditure over 50000 a month

```
✓ [11] 1 high_expenditure_count = loan_df.filter(col("Expenditure") > 50000).count()  
2 print(f"Number of people with expenditure over 50,000 a month: {high_expenditure_count}")  
3
```

⇒ Number of people with expenditure over 50,000 a month: 6

7. Number of members who are eligible for credit card

```
✓ [12] 1 credit_card_eligible = loan_df.filter((col("Income") > 50000) & (col("Returned Cheque") == 0)).count()  
2 print(f"Number of members eligible for a credit card: {credit_card_eligible}")  
3
```

⇒ Number of members eligible for a credit card: 22

- In Credit CSV file-

1. Credit card users in Spain

```
✓ [21] 1 spain_users_count = credit_df.filter(col("Geography") == "Spain").count()  
0s      2 print(f"Number of credit card users in Spain: {spain_users_count}")  
      3
```

➡ Number of credit card users in Spain: 2477

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

```
✓ [22] 1 eligible_and_active_count = credit_df.filter(  
0s      2     (col("CreditScore") > 600) & (col("IsActiveMember") == 1)  
      3 ).count()  
      4  
      5 print(f"Number of members who are eligible and active in the bank: {eligible_and_active_count}")
```

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

- In Transactions file-

1. Maximum withdrawal amount in transactions.csv

```
✓ [13] 1 max_withdrawal = txn_df.agg(max(" WITHDRAWAL AMT ").alias("MaxWithdrawal")).collect()[0][0]  
1s      2 print(f"Maximum withdrawal amount: {max_withdrawal}")
```

➡ Maximum withdrawal amount: 459447546.4

2. Minimum withdrawal amount in transactions.csv

```
✓ [14] 1 min_withdrawal_per_account = txn_df.groupBy("Account No").agg(min(" WITHDRAWAL AMT ").alias("MinWithdrawal"))  
1s      2 min_withdrawal_per_account.show()  
      3
```

➡

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

3. Maximum deposit amount of an account

```
✓ [15] 1 max_deposit_per_account = txn_df.groupBy("Account No").agg(max(" DEPOSIT AMT ").alias("MaxDeposit"))  
1s      2 max_deposit_per_account.show()  
      3
```

➡

Account No	MaxDeposit
409000438611'	1.7025E8
1196711'	5.0E8
1196428'	2.119594422E8
409000493210'	1.5E7
409000611074'	3000000.0
409000425051'	1.5E7
409000405747'	2.021E8
409000362497'	2.0E8
409000493201'	100000.0
409000438620'	5.448E8

4. Minimum deposit amount of an account

```
[16] 1 min_deposit_per_account = txn_df.groupby("Account No").agg(min("DEPOSIT AMT").alias("MinDeposit"))
      2 min_deposit_per_account.show()
```

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

5. Sum of balance in every bank account

```
[17] 1 total_balance_per_account = txn_df.groupby("Account No").agg(sum("BALANCE AMT").alias("TotalBalance"))
      2 total_balance_per_account.show()
```

```
+-----+-----+
| Account No|TotalBalance|
+-----+-----+
|409000438611'| -2.49486577068339...|
|      1196711'| -1.60476498101275E13|
|      1196428'| -8.1418498130721E13|
|409000493210'| -3.27584952132095...|
|409000611074'|  1.615533622E9|
|409000425051'| -3.77211841164998...|
|409000405747'| -2.43108047067000...|
|409000362497'| -5.2860004792808E13|
|409000493201'|  1.042083182949985E9|
|409000438620'| -7.12291867951358...|
+-----+-----+
```

6. Number of transaction on each date

```
[18] 1 transactions_per_date = txn_df.groupby("VALUE DATE").agg(count("*").alias("TransactionCount"))
      2 transactions_per_date.show()
      3
```

```
+-----+-----+
| 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|
| 21-Jan-16|              76|
|  4-May-18|              92|
|  7-Sep-17|              94|
+-----+-----+
```

only showing top 20 rows

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

```
✓ [19] 1 high_withdrawals = txn_df.filter(col(" WITHDRAWAL AMT ") > 100000).select("Account No", " WITHDRAWAL AMT ")  
0s 2 high_withdrawals.show()  
3
```

```
⇒ +-----+  
| 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 |  
|409000611074'| 176900.0 |  
|409000611074'| 150050.0 |  
+-----+  
only showing top 20 rows
```

--Thank You!