

(https://databricks.com)

CASE STUDY - ONLINE BANKING ANALYSIS - Sivaprakash V

Import libraries & Initiate session

```
# initialize the session
from pyspark import SparkContext
from pyspark.sql import SparkSession

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.appName('Case study program').getOrCreate()
```

Upload dataset

```
5

data_credit =spark.read.csv("/FileStore/tables/creditCard.csv",inferSchema=True,header=True)

data_txn =spark.read.csv("/FileStore/tables/txn.csv",inferSchema=True,header=True)

data_loan =spark.read.csv("/FileStore/tables/bankloan.csv",inferSchema=True,header=True)
```

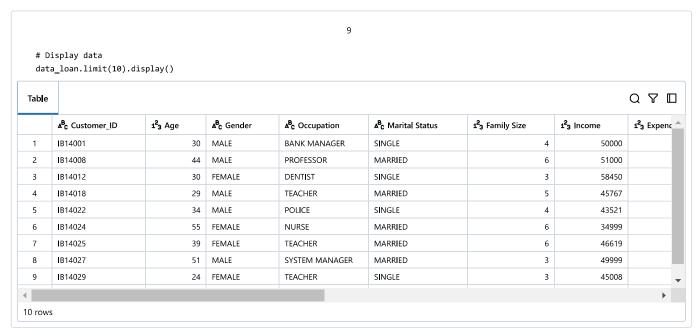
Exploring data

Loan Data

```
# Print Schema
data_loan.printSchema()

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)
```

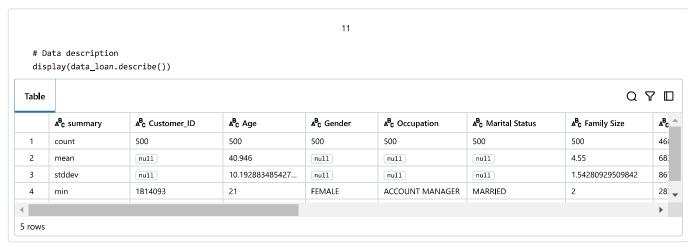


10

11/25/24, 8:47 PM

```
# Number of rows
num_rows1 = data_loan.count()
# Number of columns
num_columns1 = len(data_loan.columns)
print(f"Number of rows: {num_rows1}")
print(f"Number of columns: {num_columns1}")

Number of rows: 500
Number of columns: 15
```



```
Customer_ID
                        0
                        0
             Age
           Gender
                        0
                        0
       Occupation
   Marital Status
                        0
      Family Size
                        0
          Income
                        32
      Expenditure
                        19
    Use Frequency
                        0
                        0|
    Loan Category
      Loan Amount
                        0
                        0
         Overdue
                        0|
      Debt Record
                        0
  Returned Cheque
                        0
| Dishonour of Bill|
```

```
# Drop rows where any column has a NULL value
clean_loan = data_loan.dropna(how="any")

num_rows01 = data_loan.count()
num_rows02 = clean_loan.count()
print('With NULL')
print(f"Number of rows: {num_rows01}")
print("After Cleaning")
print(f"Number of rows: {num_rows02}")

With NULL
Number of rows: 500
After Cleaning
Number of rows: 449
```

Transaction data

```
# Print Schema
data_txn.printSchema()

root
|-- Account No: string (nullable = true)

Sivaprakash V
```

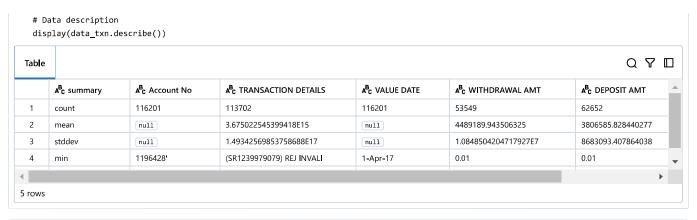
```
|-- TRANSACTION DETAILS: string (nullable = true)
|-- VALUE DATE: string (nullable = true)
|-- WITHDRAWAL AMT : double (nullable = true)
|-- DEPOSIT AMT : double (nullable = true)
|-- BALANCE AMT: double (nullable = true)
```

16 # Display data data_txn.limit(10).display() QTD Table ABC TRANSACTION DETAILS ABC VALUE DATE 1.2 WITHDRAWAL AMT 1.2 DEPOSIT AMT 1.2 BALANCE AM A^Bc Account No 409000611074 TRF FROM Indiaforensic SERVICES 29-Jun-17 1000000 null 409000611074 2 TRF FROM Indiaforensic SERVICES 5-Jul-17 1000000 null 409000611074 FDRL/INTERNAL FUND TRANSFE 18-Jul-17 null 500000 409000611074 TRF FRM Indiaforensic SERVICES 3000000 1-Aug-17 null FDRL/INTERNAL FUND TRANSFE 5 409000611074 16-Aug-17 500000 null 409000611074 16-Aug-17 500000 FDRL/INTERNAL FUND TRANSFE null 7 409000611074 FDRL/INTERNAL FUND TRANSFE 16-Aug-17 null 500000 409000611074 FDRL/INTERNAL FUND TRANSFE 16-Aug-17 500000 null 409000611074 16-Aug-17 500000 FDRL/INTERNAL FUND TRANSFE null 10 rows

Number of rows
num_rows2 = data_txn.count()
Number of columns
num_columns2 = len(data_txn.columns)
print(f"Number of rows: {num_rows2}")
print(f"Number of columns: {num_columns2}")

Number of rows: 116201
Number of columns: 6

18



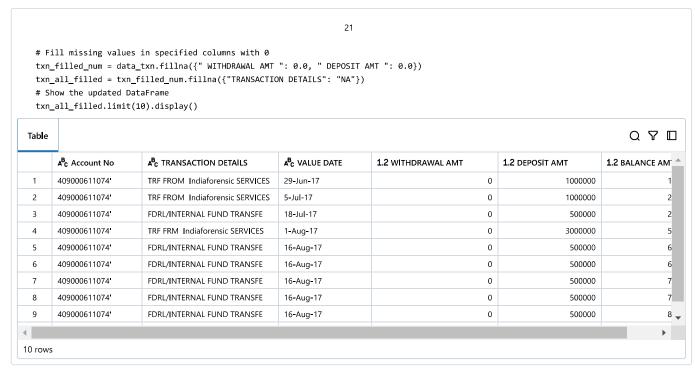
```
19
  # Calculate null counts
  null_counts1 = data_txn.select(
      [sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in data_txn.columns]
  )
  null_counts_dict1 = null_counts1.collect()[0].asDict()
  transposed_null_counts1 = spark.createDataFrame(
      [(key, value) for key, value in null_counts_dict1.items()],
      schema=["Column", "Null Count"]
  # Show the transposed DataFrame
  transposed_null_counts1.show()
+----+
            Column Null Count
        Account No
TRANSACTION DETAILS
                        2499
        VALUE DATE
                           0
   WITHDRAWAL AMT
                       62652
      DEPOSIT AMT
                       53549
       BALANCE AMT
```

20

```
# Drop rows where any column has a NULL value
clean_txn = data_txn.dropna(how="any")

num_rows11 = data_txn.count()
num_rows12 = clean_txn.count()
print('With NULL')
print(f"Number of rows: {num_rows11}")
print("After Cleaning")
print(f"Number of rows: {num_rows12}")

With NULL
Number of rows: 116201
After Cleaning
Number of rows: 0
```

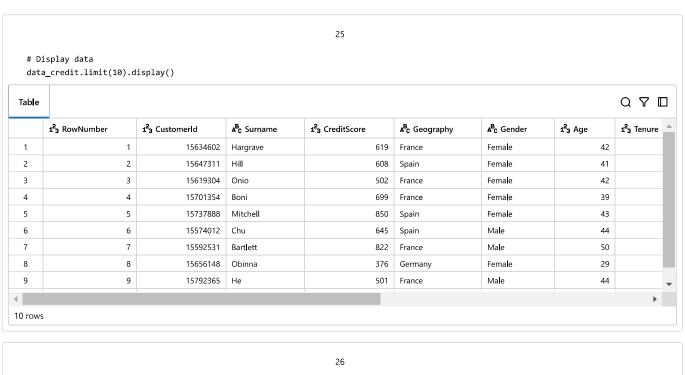


22

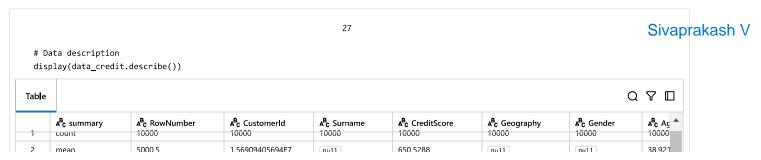
```
# Calculate null counts
  null_counts4 = txn_all_filled.select(
      [sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in txn_all_filled.columns]
  null_counts_dict4 = null_counts4.collect()[0].asDict()
  transposed_null_counts4 = spark.createDataFrame(
      [(key, value) for key, value in null_counts_dict4.items()],
      schema=["Column", "Null Count"]
  # Show the transposed DataFrame
  transposed null counts4.show()
+----+
            Column Null Count
        Account No
TRANSACTION DETAILS
                           0
        VALUE DATE
                           01
   WITHDRAWAL AMT
                           0
      DEPOSIT AMT
                           0
       BALANCE AMT
                           0
```

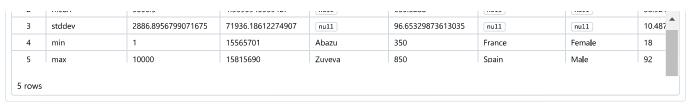
Credit data

```
24
  # Print Schema
  data_credit.printSchema()
root
 |-- RowNumber: integer (nullable = true)
 |-- CustomerId: integer (nullable = true)
 |-- Surname: string (nullable = true)
 |-- CreditScore: integer (nullable = true)
 |-- Geography: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Tenure: integer (nullable = true)
 |-- Balance: double (nullable = true)
 |-- NumOfProducts: integer (nullable = true)
 |-- IsActiveMember: integer (nullable = true)
 |-- EstimatedSalary: double (nullable = true)
 |-- Exited: integer (nullable = true)
```







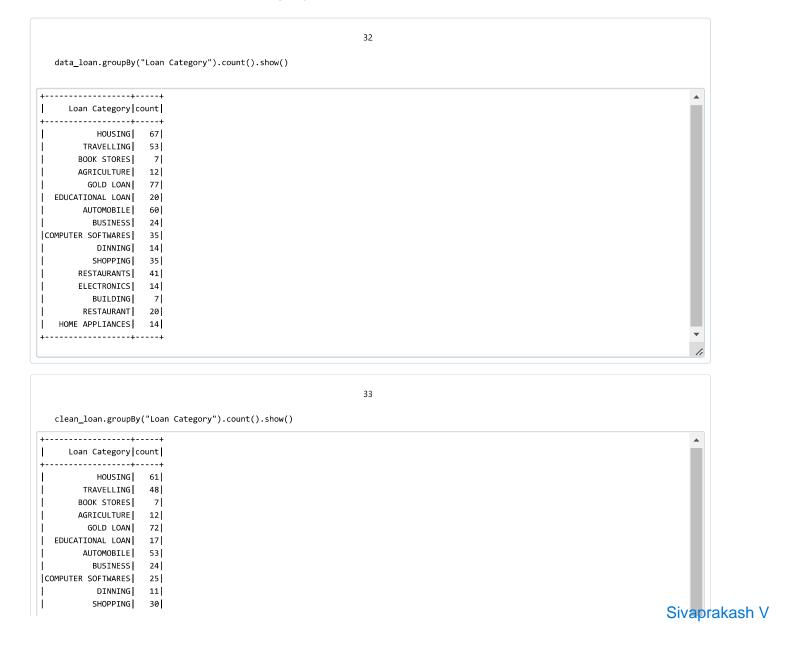


```
28
  # Calculate null counts
  null_counts2 = data_credit.select(
      [sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in data_credit.columns]
  null_counts_dict2 = null_counts2.collect()[0].asDict()
  transposed_null_counts2 = spark.createDataFrame(
      [(key, value) for key, value in null_counts_dict2.items()],
      schema=["Column", "Null Count"]
  # Show the transposed DataFrame
  transposed_null_counts2.show()
+-----+
        Column | Null Count |
      RowNumber
    CustomerId
                       0
       Surname
                       0
   CreditScore
                       0
     Geography
                       0
                       0
                       0
        Tenure
       Balance
                       0
 NumOfProducts
| IsActiveMember|
                       0
|EstimatedSalary|
        Exited
```

Question & Solution

In loandata.csv file

1. number of loans in each category



2. number of people who have taken more than 1 lack loan

```
35
   from pyspark.sql.functions import col, regexp_replace
   # since here , present in loan amount column we are replacing the comma
   # then cast it as integer
   # Remove commas and cast the Loan Amount column to integer
   loan_with_null_cast = data_loan.withColumn(
       "Loan Amount",
       regexp_replace(col("Loan Amount"), ",", "").cast("int")
   loan with null cast.printSchema()
   loan_02 = loan_with_null_cast.filter(col("Loan Amount") > 100000)
   num_rows111 = loan_02.count()
   print(f"Number of people taken more then 1 lakh in raw data: {num rows111}")
 |-- 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: integer (nullable = true)
 |-- Overdue: integer (nullable = true)
 |-- Debt Record: string (nullable = true)
 |-- Returned Cheque: integer (nullable = true)
 |-- Dishonour of Bill: integer (nullable = true)
Number of rows: 450
                                                                                                                                    Sivaprakash V
```

36

```
from pyspark.sql.functions import col, regexp_replace
   # since here , present in loan amount column we are replacing the comma
   # then cast it as integer
   # Remove commas and cast the Loan Amount column to integer
   loan_clean_cast = clean_loan.withColumn(
       "Loan Amount",
       regexp_replace(col("Loan Amount"), ",", "").cast("int")
   loan_clean_cast.printSchema()
   loan 04 = loan clean cast.filter(col("Loan Amount") > 100000)
   #display(loan 02)
   num_rows112 = loan_04.count()
   print(f"Number of people taken more then 1 lakh in clean data: {num rows112}")
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: integer (nullable = true)
 |-- Overdue: integer (nullable = true)
 |-- Debt Record: string (nullable = true)
 |-- Returned Cheque: integer (nullable = true)
 |-- Dishonour of Bill: integer (nullable = true)
Number of rows: 409
```

3. number of people with income greater than 60000 rupees

```
loan_05 = data_loan.filter(col("Income") > 60000)
num_rows114 = loan_05.count()
print(f"Number of people with income greater than 60000 rupees on raw data: {num_rows114}")

Number of people with income greater than 60000 rupees: 198
```

```
loan_06 = clean_loan.filter(col("Income") > 60000)
num_rows115 = loan_06.count()
print(f"Number of people with income greater than 60000 rupees on clean data: {num_rows115}")

Number of people with income greater than 60000 rupees on clean data: 192
```

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

```
loan_07 = data_loan.filter((col(" Returned Cheque") >=2)&(col("Income") <50000))
num_rows007 = loan_07.count()
print(f"No of people with returned cheq>=2 & salary<50000data (raw): {num_rows007}")

No of people with returned cheq>=2 & salary<50000data (raw): 137
```

```
loan_08 = clean_loan.filter((col(" Returned Cheque") >=2)&(col("Income") <50000))
num_rows008 = loan_08.count()
print(f"No of people with returned cheq>=2 & salary<50000data (clean): {num_rows008}")

No of people with returned cheq>=2 & salary<50000data (clean): 132
```

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

```
loan_09 = data_loan.filter((col(" Returned Cheque") >= 2) & (col("Marital Status") == "Single"))
num_rows009 = loan_09.count()
print(f"No of people with returned cheq>=2 & single: {num_rows009}")

No of people with returned cheq>=2 & single: 0
```

6. number of people with expenditure over 50000 a month

```
loan_10 = data_loan.filter(col("Expenditure") >50000)
num_rows010 = loan_10.count()
print(f"No of people with expenditure>50000 (raw): {num_rows010}")

No of people with expenditure>50000 (raw): 6
```

```
loan_11 = clean_loan.filter(col("Expenditure") >50000)
num_rows011 = loan_11.count()
print(f"No of people with expenditure>50000(clean): {num_rows011}")

No of people with expenditure>50000(clean): 6
```

7. number of members who are elgible for credit card

```
eligible_customers1 = data_loan.filter(
    (col("Income") > 20000) &
    (col(" Returned Cheque") == 0) & # No returned cheques
    (col(" Dishonour of Bill") == 0)
)

# Count the number of eligible members
eligible_count1 = eligible_customers1.count()
print(f"No of people eligible for loan (raw): {eligible_count1}")
```

```
eligible_customers2 = clean_loan.filter(
    (col("Income") > 20000) &
    (col(" Returned Cheque") == 0) &
    (col(" Dishonour of Bill") == 0)
)

# Count the number of eligible members
eligible_count2 = eligible_customers2.count()
print(f"No of people eligible for loan (clean): {eligible_count2}")

No of people eligible for loan (clean): 2
```

In credit.csv file

1. credit card users in Spain

```
credit_01 = data_credit.filter(col("Geography") =='Spain')
count01 = credit_01.count()
print(f"No of Credit card users in Spain: {count01}")
No of Credit card users in Spain: 2477
```

2. number of members who are elgible and active in the bank

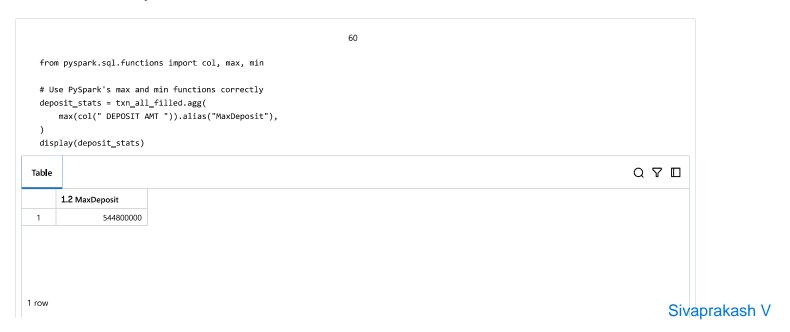
```
55
   #Works based on certain assumptions
   eligible_active_customers = data_credit.filter(
       (col("CreditScore") >= 600) &
                                           # Credit score threshold
       (col("Balance") > 0) &
                                          # Non-zero balance
       (col("EstimatedSalary") >= 20000) & # Minimum salary threshold
       (col("Exited") == 0) &
                                         # Customer has not exited
       (col("IsActiveMember") == 1)
                                          # Customer is active
   # Count the number of eligible and active customers
   eligible_active_count = eligible_active_customers.count()
   print(f"Number of eligible and active customers: {eligible_active_count}")
Number of eligible and active customers: 1803
```

In Transactions file

1. Maximum withdrawal amount in transactions Minimum withdrawal amount of an account



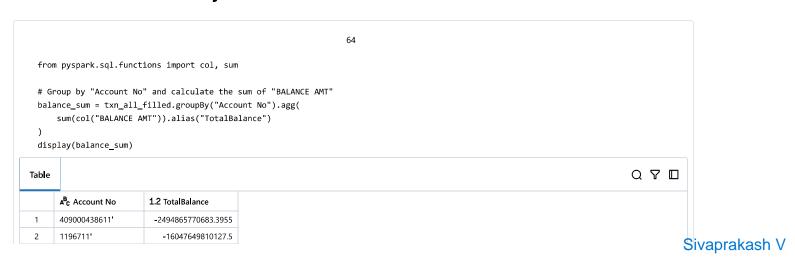
2. maximum deposit amount of an account



3. minimum deposit amount of an account

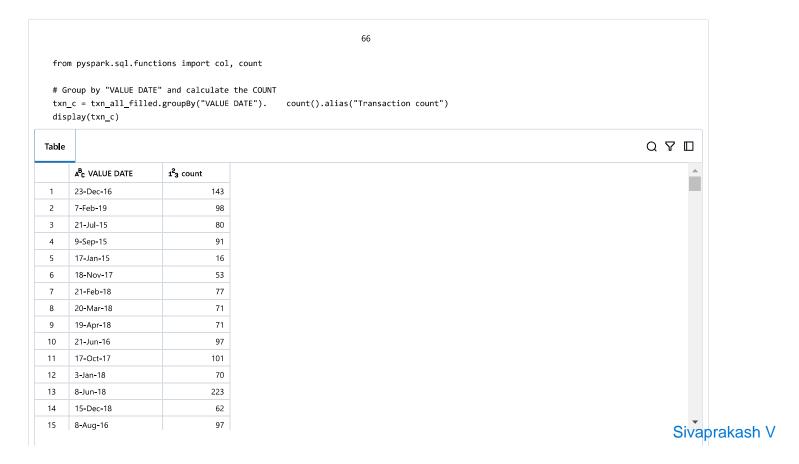


4. sum of balance in every bank account



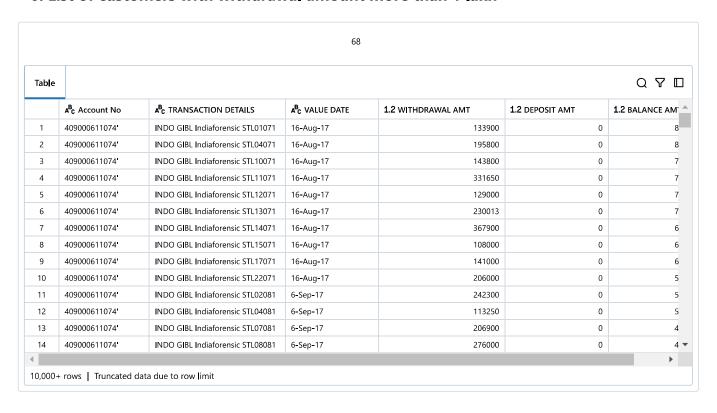
3	1196428'	- 81418498130721
4	409000493210'	-3275849521320.9575
5	409000611074'	1615533622
6	409000425051	-3772118411.6499877
7	409000405747'	-24310804706.700016
8	409000493201'	1042083182.9499985
9	409000438620'	-7122918679513.586
10	409000362497'	-52860004792808
10 row	'S	

5. Number of transaction on each date



1,294 rows

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



Thank You