databrickscase_study

(https://databricks.com)

CASE STUDY - ONLINE BANKING ANALYSIS - Sivaprakash V

Import libraries & Initiate session

3

```
# 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

8

```
# Print Schema
data_loan.printSchema()
```

```
root
 |-- Customer_ID: string (nullable = true)
```

Sivaprakash V

```
|-- 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)
```

9

```
# Display data
data_loan.limit(10).display()
```

Table

	Customer_ID	Age	Gender	Occupation	Marital Status	Family Size	Income	Expend
1	IB14001	30	MALE	BANK MANAGER	SINGLE	4	50000	
2	IB14008	44	MALE	PROFESSOR	MARRIED	6	51000	
3	IB14012	30	FEMALE	DENTIST	SINGLE	3	58450	
4	IB14018	29	MALE	TEACHER	MARRIED	5	45767	
5	IB14022	34	MALE	POLICE	SINGLE	4	43521	
6	IB14024	55	FEMALE	NURSE	MARRIED	6	34999	
7	IB14025	39	FEMALE	TEACHER	MARRIED	6	46619	
8	IB14027	51	MALE	SYSTEM MANAGER	MARRIED	3	49999	
9	IB14029	24	FEMALE	TEACHER	SINGLE	3	45008	

10 rows

10

Sivaprakash V

```
# 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
```

11

```
# Data description
display(data_loan.describe())
```

Table

	summary	Customer_ID	Age	Gender	Occupation	Marital Status	Family Size	
1	count	500	500	500	500	500	500	46
2	mean	null	40.946	null	null	null	4.55	68
3	stddev	null	10.192883485427...	null	null	null	1.54280929509842	86
4	min	1B14093	21	FEMALE	ACCOUNT MANAGER	MARRIED	2	28

5 rows

12

```
from pyspark.sql.functions import col, sum, when, lit

# Calculate null counts
null_counts = data_loan.select(
    [sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in data_loan.columns]
)
null_counts_dict = null_counts.collect()[0].asDict()
transposed_null_counts = spark.createDataFrame(
    [(key, value) for key, value in null_counts_dict.items()],
    schema=["Column", "Null Count"]
)
# Show the transposed DataFrame
transposed_null_counts.show()
```

```
+-----------------+----------+
|           Column|Null Count|
+-----------------+----------+
```

Sivaprakash V

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

13

```
# 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

15

```
# Print Schema
data_txn.printSchema()
```

```
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)
```

16

```
# Display data
data_txn.limit(10).display()
```

Table

	Account No	TRANSACTION DETAILS	VALUE DATE	WITHDRAWAL AMT	DEPOSIT AMT	BALANCE AMT
1	409000611074'	TRF FROM Indiaforensic SERVICES	29-Jun-17	null	1000000	1
2	409000611074'	TRF FROM Indiaforensic SERVICES	5-Jul-17	null	1000000	2
3	409000611074'	FDRL/INTERNAL FUND TRANSFE	18-Jul-17	null	500000	2
4	409000611074'	TRF FRM Indiaforensic SERVICES	1-Aug-17	null	3000000	5
5	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	null	500000	6
6	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	null	500000	6
7	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	null	500000	7
8	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	null	500000	7
9	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	null	500000	8

10 rows

17

```
# 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

Sivaprakash V

```
# Data description
display(data_txn.describe())
```

Table

	summary	Account No	TRANSACTION DETAILS	VALUE DATE	WITHDRAWAL AMT	DEPOSIT AMT
1	count	116201	113702	116201	53549	62652
2	mean	null	3.675022545399418E15	null	4489189.943506325	3806585.828440277
3	stddev	null	1.49342569853758688E17	null	1.0848504204717927E7	8683093.407864038
4	min	1196428'	(SR1239979079) REJ INVALI	1-Apr-17	0.01	0.01

5 rows

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|         0|
|TRANSACTION DETAILS|      2499|
|        VALUE DATE|         0|
|    WITHDRAWAL AMT |     62652|
|       DEPOSIT AMT |     53549|
|       BALANCE AMT|         0|
+------------------+----------+
```

20

Sivaprakash V

```
# 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
```

21

```
# Fill missing values in specified columns with 0
txn_filled_num = data_txn.fillna({" WITHDRAWAL AMT ": 0.0, " DEPOSIT AMT ": 0.0})
txn_all_filled = txn_filled_num.fillna({"TRANSACTION DETAILS": "NA"})
# Show the updated DataFrame
txn_all_filled.limit(10).display()
```

Table

	Account No	TRANSACTION DETAILS	VALUE DATE	1.2 WITHDRAWAL AMT	1.2 DEPOSIT AMT	1.2 BALANCE AM
1	409000611074'	TRF FROM Indiaforensic SERVICES	29-Jun-17	0	1000000	1
2	409000611074'	TRF FROM Indiaforensic SERVICES	5-Jul-17	0	1000000	2
3	409000611074'	FDRL/INTERNAL FUND TRANSFE	18-Jul-17	0	500000	2
4	409000611074'	TRF FRM Indiaforensic SERVICES	1-Aug-17	0	3000000	5
5	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	0	500000	6
6	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	0	500000	6
7	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	0	500000	7
8	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	0	500000	7
9	409000611074'	FDRL/INTERNAL FUND TRANSFE	16-Aug-17	0	500000	8

10 rows

22

Sivaprakash V

```
# 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|         0|
|TRANSACTION DETAILS|         0|
|         VALUE DATE|         0|
|    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)
```

Sivaprakash V

25

```
# Display data
data_credit.limit(10).display()
```

Table

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
1	1	15634602	Hargrave	619	France	Female	42	
2	2	15647311	Hill	608	Spain	Female	41	
3	3	15619304	Onio	502	France	Female	42	
4	4	15701354	Boni	699	France	Female	39	
5	5	15737888	Mitchell	850	Spain	Female	43	
6	6	15574012	Chu	645	Spain	Male	44	
7	7	15592531	Bartlett	822	France	Male	50	
8	8	15656148	Obinna	376	Germany	Female	29	
9	9	15792365	He	501	France	Male	44	

10 rows

26

```
# Number of rows
num_rows3 = data_credit.count()
# Number of columns
num_columns3 = len(data_credit.columns)
print(f"Number of rows: {num_rows3}")
print(f"Number of columns: {num_columns3}")
```

```
Number of rows: 10000
Number of columns: 13
```

27

Sivaprakash V

```
# Data description
display(data_credit.describe())
```

Table

	summary	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Ag
1	count	10000	10000	10000	10000	10000	10000	10000
2	mean	5000.5	1.56909405694E7	null	650.5288	null	null	38.921

3	stddev	2886.8956799071675	71936.18612274907	null	96.65329873613035	null	null	10.487
4	min	1	15565701	Abazu	350	France	Female	18
5	max	10000	15815690	Zuyeva	850	Spain	Male	92

5 rows

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|         0|
|     CustomerId|         0|
|        Surname|         0|
|    CreditScore|         0|
|      Geography|         0|
|         Gender|         0|
|            Age|         0|
|         Tenure|         0|
|        Balance|         0|
|  NumOfProducts|         0|
| IsActiveMember|         0|
|EstimatedSalary|         0|
|         Exited|         0|
+---------------+----------+
```

Question & Solution

In loandata.csv file

Sivaprakash V

1. number of loans in each category

32

```
data_loan.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|
+------------------+-----+
```

33

```
clean_loan.groupBy("Loan Category").count().show()
```

```
+------------------+-----+
|     Loan Category|count|
+------------------+-----+
|           HOUSING|   61|
|        TRAVELLING|   48|
|       BOOK STORES|    7|
|       AGRICULTURE|   12|
|         GOLD LOAN|   72|
|  EDUCATIONAL LOAN|   17|
|        AUTOMOBILE|   53|
|          BUSINESS|   24|
|COMPUTER SOFTWARES|   25|
|           DINNING|   11|
|          SHOPPING|   30|
```

Sivaprakash V

```
|       RESTAURANTS|   37|
|       ELECTRONICS|   13|
|          BUILDING|    6|
|        RESTAURANT|   20|
|   HOME APPLIANCES|   13|
+------------------+-----+
```

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}")
```

```
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: 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

41

```
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
```

42

```
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

44

```
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

46

```
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
```

```
47
```

```
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

```
49
```

```
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}")
```

```
3
```

```
50
```

```
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
```

Sivaprakash V

In credit.csv file

1. credit card users in Spain

```
                                                                                53

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

```
                                                                                58
```

Sivaprakash V

```
from pyspark.sql.functions import col, max, min

# Use PySpark's max and min functions correctly
withdrawal_stats = txn_all_filled.agg(
    max(col(" WITHDRAWAL AMT ")).alias("MaxWithdrawal"),
    min(col(" WITHDRAWAL AMT ")).alias("MinWithdrawal")
)
display(withdrawal_stats)
```

Table

	1.2 MaxWithdrawal	1.2 MinWithdrawal
1	459447546.4	0

1 row

2. maximum deposit amount of an account

60

```
from pyspark.sql.functions import col, max, min

# Use PySpark's max and min functions correctly
deposit_stats = txn_all_filled.agg(
    max(col(" DEPOSIT AMT ")).alias("MaxDeposit"),
)
display(deposit_stats)
```

Table

	1.2 MaxDeposit
1	544800000

1 row

Sivaprakash V

3. minimum deposit amount of an account

62

```
from pyspark.sql.functions import col, max, min

# Use PySpark's max and min functions correctly
deposit_stats1 = txn_all_filled.agg(
    min(col(" DEPOSIT AMT ")).alias("MinDeposit"),
)
display(deposit_stats1)
```

Table

	1.2 MinDeposit
1	0

1 row

4. sum of balance in every bank account

64

```
from pyspark.sql.functions import col, sum

# Group by "Account No" and calculate the sum of "BALANCE AMT"
balance_sum = txn_all_filled.groupBy("Account No").agg(
    sum(col("BALANCE AMT")).alias("TotalBalance")
)
display(balance_sum)
```

Table

	Account No	1.2 TotalBalance
1	409000438611'	-2494865770683.3955
2	1196711'	-16047649810127.5

Sivaprakash V

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 rows

5. Number of transaction on each date

66

```
from pyspark.sql.functions import col, count

# Group by "VALUE DATE" and calculate the COUNT
txn_c = txn_all_filled.groupBy("VALUE DATE").    count().alias("Transaction count")
display(txn_c)
```

Table

	VALUE DATE	count
1	23-Dec-16	143
2	7-Feb-19	98
3	21-Jul-15	80
4	9-Sep-15	91
5	17-Jan-15	16
6	18-Nov-17	53
7	21-Feb-18	77
8	20-Mar-18	71
9	19-Apr-18	71
10	21-Jun-16	97
11	17-Oct-17	101
12	3-Jan-18	70
13	8-Jun-18	223
14	15-Dec-18	62
15	8-Aug-16	97

Sivaprakash V

1,294 rows

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

68

Table

	Account No	TRANSACTION DETAILS	VALUE DATE	WITHDRAWAL AMT	DEPOSIT AMT	BALANCE AM
1	409000611074'	INDO GIBL Indiaforensic STL01071	16-Aug-17	133900	0	8
2	409000611074'	INDO GIBL Indiaforensic STL04071	16-Aug-17	195800	0	8
3	409000611074'	INDO GIBL Indiaforensic STL10071	16-Aug-17	143800	0	7
4	409000611074'	INDO GIBL Indiaforensic STL11071	16-Aug-17	331650	0	7
5	409000611074'	INDO GIBL Indiaforensic STL12071	16-Aug-17	129000	0	7
6	409000611074'	INDO GIBL Indiaforensic STL13071	16-Aug-17	230013	0	7
7	409000611074'	INDO GIBL Indiaforensic STL14071	16-Aug-17	367900	0	6
8	409000611074'	INDO GIBL Indiaforensic STL15071	16-Aug-17	108000	0	6
9	409000611074'	INDO GIBL Indiaforensic STL17071	16-Aug-17	141000	0	6
10	409000611074'	INDO GIBL Indiaforensic STL22071	16-Aug-17	206000	0	5
11	409000611074'	INDO GIBL Indiaforensic STL02081	6-Sep-17	242300	0	5
12	409000611074'	INDO GIBL Indiaforensic STL04081	6-Sep-17	113250	0	5
13	409000611074'	INDO GIBL Indiaforensic STL07081	6-Sep-17	206900	0	4
14	409000611074'	INDO GIBL Indiaforensic STL08081	6-Sep-17	276000	0	4

10,000+ rows | Truncated data due to row limit

Sivaprakash V

Thank You

-Sivaprakash V