Pyspark Case Study 1

ONLINE BANKING ANALYSIS

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, regexp_replace, trim, to_date, sum as spark_sum, max as spark_max, min as spark_min

spark = SparkSession.builder.appName("OnlineBankingAnalysis").getOrCreate()

loan_df = spark.read.option("header", "true").csv("/content/loan.csv", inferSchema=True)
credit_df = spark.read.option("header", "true").csv("/content/credit card.csv", inferSchema=True)
txn_df = spark.read.option("header", "true").csv("/content/txn.csv", inferSchema=True)

loan_df = loan_df.selectExpr([f"`{c}` as `{c.strip().replace(' ', '_')}`" for c in loan_df.columns])
credit_df = credit_df.selectExpr([f"`{c}` as `{c.strip().replace(' ', '_')}`" for c in credit_df.columns])
txn_df = txn_df.selectExpr([f"`{c}` as `{c.strip().replace(' ', '_')}`" for c in txn_df.columns])

loan_df = loan_df.withColumn("Loan_Amount_CLEAN", regexp_replace(trim(col("Loan_Amount")), ",", "").cast("double"))

txn_df = txn_df.withColumn("WITHDRAWAL_AMT_CLEAN", regexp_replace(trim(col("WITHDRAWAL_AMT")), ",", "").cast("double"))
txn_df = txn_df.withColumn("DEPOSIT_AMT_CLEAN", regexp_replace(trim(col("DEPOSIT_AMT")), ",", "").cast("double"))
txn_df = txn_df.withColumn("BALANCE_CLEAN", regexp_replace(trim(col("BALANCE_AMT")), ",", "").cast("double"))
txn_df = txn_df.withColumn("DATE", to_date("VALUE_DATE", "dd-MM-yyyy"))
```

1. Number of loans in each category

```python
# ---------------- LOAN DATA ----------------

print("1.Number of loans in each category")
loan_df.groupBy("Loan_Category").count().show()
```

```
1.Number of loans in each category
+------------------+-----+
|     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 lack loan

```python
print("2.People who took loan > 1 lakh")
filtered_loan_1 = loan_df.filter(col("Loan_Amount_CLEAN") > 100000)
print("Count:", filtered_loan_1.count())
filtered_loan_1.show(truncate=False )
```

2.People who took loan > 1 lakh
Count: 450

Customer_ID	Age	Gender	Occupation	Marital_Status	Family_Size	Income	Expenditure	Use_Frequency	Loan_Category	Loan_Amount	Overdue	Debt_Record	Returned_Cheque	Dishonour_of_Bill	Loan_Amount_CLEAN
IB14001	30	MALE	BANK MANAGER	SINGLE	4	50000	22199	6	HOUSING	10,00,000	5	42,898	6	9	1000000.0
IB14018	29	MALE	TEACHER	MARRIED	5	45767	12787	3	GOLD LOAN	6,00,000	7	11,000	0	4	600000.0
IB14022	34	MALE	POLICE	SINGLE	4	43521	11999	3	AUTOMOBILE	2,00,000	2	43,898	1	2	200000.0
IB14025	39	FEMALE	TEACHER	MARRIED	6	46619	18675	4	HOUSING	12,09,867	8	29,999	6	8	1209867.0
IB14029	24	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IB14039	45	MALE	ACCOUNT MANAGER	MARRIED	7	45777	18452	4	GOLD LOAN	9,87,611	7	39,999	8	1	987611.0
IB14041	59	FEMALE	ASSISTANT PROFESSOR	MARRIED	4	50999	22999	5	EDUCATIONAL LOAN	5,99,934	3	9,000	9	9	599934.0
IB14042	25	FEMALE	DOCTOR	SINGLE	4	60111	27111	5	TRAVELLING	12,90,929	4	18,000	1	0	1290929.0
IB14045	31	MALE	STORE KEEPER	SINGLE	5	40999	11999	3	BOOK STORES	1,67,654	1	4,500	0	1	167654.0
IB14050	56	MALE	CIVIL ENGINEER	MARRIED	4	NULL	13999	3	HOUSING	10,65,577	6	19,999	4	2	1065577.0
IB14054	58	FEMALE	DOCTOR	MARRIED	5	60000	25000	5	HOUSING	9,00,000	5	21,000	9	0	900000.0
IB14057	25	MALE	AIRPORT OFFICER	SINGLE	4	40000	18888	3	RESTAURANTS	4,00,000	8	11,111	1	7	400000.0
IB14060	36	FEMALE	CLERK	MARRIED	4	35000	15000	3	HOUSING	3,00,000	2	5,600	4	8	300000.0
IB14070	40	MALE	PUBLIC WORKS	MARRIED	4	38000	20000	3	GOLD LOAN	4,00,000	9	19,954	3	2	400000.0
IB14082	60	FEMALE	TEACHER	MARRIED	5	70000	40000	9	GOLD LOAN	2,57,789	4	10,058	4	3	257789.0
IB14085	30	MALE	ELECTRICIAN	MARRIED	4	30000	15000	5	HOUSING	3,54,789	5	32,154	5	5	354789.0
IB14086	51	FEMALE	TECHNICIAN	MARRIED	5	30000	NULL	5	RESTAURANTS	1,25,463	7	52,634	4	10	125463.0
IB14089	25	MALE	PROFESSOR	SINGLE	5	62145	31254	4	BOOK STORES	12,45,789	6	48,596	6	5	1245789.0
IB14092	47	MALE	SYSTEM ENGINEER	MARRIED	4	52364	45612	3	GOLD LOAN	6,54,725	4	67,451	5	4	654725.0
1B14093	21	FEMALE	MANAGER	SINGLE	3	42516	24567	7	AUTOMOBILE	25,69,874	8	89,652	2	3	2569874.0

only showing top 20 rows

3. Number of people with income greater than 60000 rupees

```
print("\n3. People with income > 60000")
filtered_income = loan_df.filter(col("Income") > 60000)
print("Count:", filtered_income.count())
filtered_income.show()
```

```
3. People with income > 60000
Count: 198
```

Customer_ID	Age	Gender	Occupation	Marital_Status	Family_Size	Income	Expenditure	Use_Frequency	Loan_Category	Loan_Amount	Overdue	Debt_Record	Returned_Cheque	Dishonour_of_Bill	Loan_Amount_CLEAN
IB14032	24	MALE	DATA ANALYST	SINGLE	4	60111	28999	6	AUTOMOBILE	35,232	5	33,333	1	2	35232.0
IB14042	25	FEMALE	DOCTOR	SINGLE	4	60111	27111	5	TRAVELLING	12,90,929	4	18,000	1	0	1290929.0
IB14082	60	FEMALE	TEACHER	MARRIED	5	70000	40000	9	GOLD LOAN	2,57,789	4	10,058	4	3	257789.0
IB14089	25	MALE	PROFESSOR	SINGLE	5	62145	31254	4	BOOK STORES	12,45,789	6	48,596	6	5	1245789.0
1B14094	49	MALE	ASSISTANT PROFESSOR	MARRIED	5	65214	42589	5	HOUSING	9,85,412	5	11,254	1	2	985412.0
IB14099	47	FEMALE	DOCTOR	MARRIED	4	72154	45286	4	AUTOMOBILE	7,54,126	2	19,524	5	2	754126.0
IB14104	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	32541	2	AUTOMOBILE	20,45,789	1	16,599	2	3	2045789.0
IB14107	44	FEMALE	ACCOUNT MANAGER	MARRIED	4	800000	15632	8	AUTOMOBILE	23,65,478	5	20,145	3	4	2365478.0
IB14115	41	MALE	BANK MANAGER	MARRIED	6	64125	21246	6	TRAVELLING	6,52,147	5	16,524	3	3	652147.0
IB14121	33	MALE	DOCTOR	MARRIED	6	70000	12541	8	HOUSING	7,45,213	4	19,541	1	3	745213.0
IB14128	46	FEMALE	CLERK	MARRIED	3	750000	25641	5	GOLD LOAN	2,14,569	4	16,324	3	4	214569.0
IB14134	33	MALE	DOCTOR	MARRIED	6	70000	33541	8	BUILDING	7,45,213	4	19,541	1	3	745213.0
IB14146	56	MALE	FIRE DEPARTMENT	MARRIED	6	67890	34567	5	TRAVELLING	6,78,500	5	13,560	3	4	678500.0
IBI4153	58	MALE	SYSTEM ENGINEER	MARRIED	6	76800	NULL	5	TRAVELLING	16,59,080	6	29,000	5	3	1659080.0
IBI4157	35	MALE	BANK MANAGER	MARRIED	4	930000	35680	6	HOUSING	6,79,040	5	34,000	5	5	679040.0
IB14158	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	62541	2	AUTOMOBILE	20,45,789	1	16,599	2	3	2045789.0
IB14163	44	FEMALE	ACCOUNT MANAGER	MARRIED	4	800000	15632	8	COMPUTER SOFTWARES	23,65,478	5	20,145	3	4	2365478.0
IB14171	47	FEMALE	DOCTOR	MARRIED	4	72154	45286	4	AUTOMOBILE	7,54,126	2	19,524	5	2	754126.0
IB14176	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	62541	2	HOUSING	20,45,789	1	16,599	2	3	2045789.0
IB14184	24	MALE	DATA ANALYST	SINGLE	4	60111	28999	6	RESTAURANTS	35,232	5	33,333	1	2	35232.0

only showing top 20 rows

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

```
print("\n4. People with 2+ returned cheques and income < 50000")
filtered_returned_1 = loan_df.filter((col("Returned_Cheque") >= 2) & (col("Income") < 50000))
print("Count:", filtered_returned_1.count())
filtered_returned_1.show()
```

```
4. People with 2+ returned cheques and income < 50000
Count: 137
```

Customer_ID	Age	Gender	Occupation	Marital_Status	Family_Size	Income	Expenditure	Use_Frequency	Loan_Category	Loan_Amount	Overdue	Debt_Record	Returned_Cheque	Dishonour_of_Bill	Loan_Amount_CLEAN
IB14025	39	FEMALE	TEACHER	MARRIED	6	46619	18675	4	HOUSING	12,09,867	8	29,999	6	8	1209867.0
IB14027	51	MALE	SYSTEM MANAGER	MARRIED	3	49999	19111	5	RESTAURANTS	60,676	8	13,000	2	5	60676.0
IB14029	24	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IB14037	54	FEMALE	TEACHER	MARRIED	5	48099	19999	4	RESTAURANTS	30,999	1	12,000	7	5	30999.0
IB14039	45	MALE	ACCOUNT MANAGER	MARRIED	7	45777	18452	4	GOLD LOAN	9,87,611	7	39,999	8	1	987611.0
IB14049	49	MALE	BANK MANAGER	MARRIED	4	45999	14500	4	TRAVELLING	79,999	4	6,700	7	3	79999.0
IB14060	36	FEMALE	CLERK	MARRIED	4	35000	15000	3	HOUSING	3,00,000	2	5,600	4	8	300000.0
IB14070	40	MALE	PUBLIC WORKS	MARRIED	4	38000	20000	3	GOLD LOAN	4,00,000	9	19,954	3	2	400000.0
IB14078	45	FEMALE	FIRE DEPARTMENT	MARRIED	4	40000	18888	4	AUTOMOBILE	70,000	1	0	2	1	70000.0
IB14085	30	MALE	ELECTRICIAN	MARRIED	4	30000	15000	5	HOUSING	3,54,789	5	32,154	5	5	354789.0
IB14086	51	FEMALE	TECHNICIAN	MARRIED	5	30000	NULL	5	RESTAURANTS	1,25,463	7	52,634	4	10	125463.0
1B14093	21	FEMALE	MANAGER	SINGLE	3	42516	24567	7	AUTOMOBILE	25,69,874	8	89,652	2	3	2569874.0
IB14096	33	FEMALE	CLERK	MARRIED	3	35684	15247	3	RESTAURANTS	14,52,637	3	13,547	3	2	1452637.0
IB14106	29	MALE	FIRE DEPARTMENT	MARRIED	5	45213	32457	9	TRAVELLING	15,24,789	7	90,000	2	5	1524789.0
IB14109	56	MALE	DRIVER	MARRIED	5	30000	15426	7	TRAVELLING	9,21,456	6	20,000	4	6	921456.0
IB14113	49	MALE	ASSISTANT MANAGER	MARRIED	7	45612	39542	3	SHOPPING	5,87,412	7	65,412	3	2	587412.0
IB14123	36	MALE	ELECTRICIAN	MARRIED	2	36985	25648	6	AUTOMOBILE	9,85,413	7	20,000	5	3	985413.0
IB14136	36	MALE	ELECTRICIAN	MARRIED	2	36985	25648	6	ELECTRONICS	9,85,413	7	20,000	5	3	985413.0
IB14138	27	FEMALE	SOFTWARE ENGINEER	SINGLE	4	40000	22000	4	GOLD LOAN	4,00,000	4	15,647	5	3	400000.0
IB14143	34	FEMALE	TEACHER	MARRIED	4	45389	NULL	5	HOME APPLIANCES	3,50,050	4	24,000	4	3	350050.0

only showing top 20 rows

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

```
print("\n5. People with 2+ returned cheques and Single")
filtered_returned_2 = loan_df.filter((col("Returned_Cheque") >= 2) & (col("Marital_Status") == "SINGLE"))
print("Count:", filtered_returned_2.count())
filtered_returned_2.show()
```

```
5. People with 2+ returned cheques and Single
Count: 111
```

Customer_ID	Age	Gender	Occupation	Marital_Status	Family_Size	Income	Expenditure	Use_Frequency	Loan_Category	Loan_Amount	Overdue	Debt_Record	Returned_Cheque	Dishonour_of_Bill	Loan_Amount_CLEAN
IB14001	30	MALE	BANK MANAGER	SINGLE	4	50000	22199	6	HOUSING	10,00,000	5	42,898	6	9	1000000.0
IB14012	30	FEMALE	DENTIST	SINGLE	3	58450	27675	5	TRAVELLING	75,000	6	20,876	3	1	75000.0
IB14029	24	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IB14089	25	MALE	PROFESSOR	SINGLE	5	62145	31254	4	BOOK STORES	12,45,789	6	48,596	6	5	1245789.0
1B14093	21	FEMALE	MANAGER	SINGLE	3	42516	24567	7	AUTOMOBILE	25,69,874	8	89,652	2	3	2569874.0
IB14138	27	FEMALE	SOFTWARE ENGINEER	SINGLE	4	40000	22000	4	GOLD LOAN	4,00,000	4	15,647	5	3	400000.0
IBI4155	24	FEMALE	SOFTWARE ENGINEER	SINGLE	4	55680	29000	5	AUTOMOBILE	7,89,000	5	24,000	6	4	789000.0
IB14179	24	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IB14187	30	FEMALE	DENTIST	SINGLE	3	58450	27675	5	TRAVELLING	75,000	6	20,876	3	1	75000.0
IB14220	30	FEMALE	DENTIST	SINGLE	3	58450	27675	5	TRAVELLING	75,000	6	20,876	3	1	75000.0
IB14231	24	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IBI4294	24	FEMALE	SOFTWARE ENGINEER	SINGLE	4	49680	25000	5	GOLD LOAN	7,89,000	5	24,000	6	4	789000.0
IB14309	25	MALE	PROFESSOR	SINGLE	5	62145	31254	4	AUTOMOBILE	12,45,789	6	48,596	6	5	1245789.0
1B14312	21	FEMALE	MANAGER	SINGLE	3	42516	24567	7	EDUCATIONAL LOAN	25,69,874	8	89,652	2	3	2569874.0
IB14317	26	FEMALE	TEACHER	SINGLE	3	45008	17454	4	AUTOMOBILE	3,99,435	9	51,987	4	7	399435.0
IB14326	37	MALE	DOCTOR	SINGLE	2	56856	23678	7	TRAVELLING	588,690	8	87171	4	3	588690.0
IB14361	37	FEMALE	PRODUCT ENGINEER	SINGLE	6	NULL	28140	5	AUTOMOBILE	819,281	8	27586	6	8	819281.0
IB14436	24	FEMALE	SYSTEM MANAGER	SINGLE	4	38401	27512	7	RESTAURANTS	744,697	4	42166	4	2	744697.0
IB14470	29	MALE	NURSE	SINGLE	6	51199	15753	5	COMPUTER SOFTWARES	331,716	3	57156	3	0	331716.0
IB14476	29	MALE	SOFTWARE ENGINEER	SINGLE	5	30454	22210	4	SHOPPING	763,036	6	37762	7	9	763036.0

only showing top 20 rows

6. Number of people with expenditure over 50000 a month

```python
print("\n6. People with monthly expenses > 50000")
filtered_expense = loan_df.filter(col("Expenditure") > 50000)
print("Count:", filtered_expense.count())
filtered_expense.show()
```

```
6. People with monthly expenses > 50000
Count: 6
```

Customer_ID	Age	Gender	Occupation	Marital_Status	Family_Size	Income	Expenditure	Use_Frequency	Loan_Category	Loan_Amount	Overdue	Debt_Record	Returned_Cheque	Dishonour_of_Bill	Loan_Amount_CLEAN
IB14158	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	62541	2	AUTOMOBILE	20,45,789	1	16,599	2	3	2045789.0
IB14176	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	62541	2	HOUSING	20,45,789	1	16,599	2	3	2045789.0
IB14204	54	MALE	AIRPORT OFFICER	MARRIED	6	81000	62541	2	DINNING	20,45,789	1	16,599	2	3	2045789.0
IB14227	54	MALE	AIRPORT OFFICER	MARRIED	6	80000	62541	2	HOUSING	20,45,789	1	16,599	2	3	2045789.0
IB14278	41	MALE	BANK MANAGER	MARRIED	6	64125	51246	6	TRAVELLING	6,52,147	5	16,524	3	3	652147.0
IB15024	26	MALE	DIETICIAN	SINGLE	3	95425	53086	2	HOUSING	488,076	4	61227	5	2	488076.0

7. Credit card users in Spain

```python
# --------------- CREDIT CARD DATA ---------------

print("7. Credit card users in Spain")
spain_credit_users = credit_df.filter(col("Geography") == "Spain")
print("Count:", spain_credit_users.count())
spain_credit_users.show(truncate=False)
```

```
7. Credit card users in Spain
Count: 2477
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	IsActiveMember	EstimatedSalary	Exited
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	1	112542.58	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	0	149756.71	1
12	15737173	Andrews	497	Spain	Male	24	3	0.0	2	0	76390.01	0
15	15600882	Scott	635	Spain	Female	35	7	0.0	2	1	65951.65	0
18	15788218	Henderson	549	Spain	Female	24	9	0.0	2	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0.0	1	0	158684.81	0
22	15597945	Dellucci	636	Spain	Female	32	8	0.0	2	0	138555.46	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0.0	1	0	118913.53	1
31	15589475	Azikiwe	591	Spain	Female	39	3	0.0	3	0	140469.38	1
34	15659428	Maggard	520	Spain	Female	42	6	0.0	2	1	34410.55	0
35	15732963	Clements	722	Spain	Female	29	9	0.0	2	1	142033.07	0
37	15788448	Watson	490	Spain	Male	31	3	145260.23	1	1	114066.77	0
38	15729599	Lorenzo	804	Spain	Male	33	7	76548.6	1	1	98453.45	0
41	15619360	Hsiao	472	Spain	Male	40	4	0.0	1	0	70154.22	0
45	15684171	Bianchi	660	Spain	Female	61	5	155931.11	1	1	158338.39	0
59	15623944	T'ien	511	Spain	Female	66	4	0.0	1	0	1643.11	1
63	15702014	Jeffrey	555	Spain	Male	33	1	56084.69	2	0	178798.13	0
64	15751208	Pirozzi	684	Spain	Male	56	8	78707.16	1	1	99398.36	0
73	15812518	Palermo	657	Spain	Female	37	0	163607.18	1	1	44203.55	0

only showing top 20 rows

8. Maximum withdrawal amount in transactions

```python
# --------------- TRANSACTION DATA ---------------

print("8.Maximum withdrawal amount")
txn_df.select(spark_max("WITHDRAWAL_AMT_CLEAN").alias("Max_Withdrawal")).show()
```

```
8.Maximum withdrawal amount
+--------------+
|Max_Withdrawal|
+--------------+
|  4.594475464E8|
+--------------+
```

9. Minimum withdrawal amount of an account

```python
print("9.Minimum withdrawal amount")
txn_df.select(spark_min("WITHDRAWAL_AMT_CLEAN").alias("Min_Withdrawal")).show()
```

```
9.Minimum withdrawal amount
+--------------+
|Min_Withdrawal|
+--------------+
|          0.01|
+--------------+
```

10. Maximum deposit amount of an account

```
print("10.Maximum deposit amount")
txn_df.select(spark_max("DEPOSIT_AMT_CLEAN").alias("Max_Deposit")).show()
```

```
10.Maximum deposit amount
+-----------+
|Max_Deposit|
+-----------+
|    5.448E8|
+-----------+
```

11. Minimum deposit amount of an account

```
print("11.Minimum deposit amount")
txn_df.select(spark_min("DEPOSIT_AMT_CLEAN").alias("Min_Deposit")).show()
```

```
11.Minimum deposit amount
+-----------+
|Min_Deposit|
+-----------+
|       0.01|
+-----------+
```

12. Total balance amount in every bank account

```
print("12.Total balance in each account")
txn_df.groupBy("Account_No").agg(spark_sum("BALANCE_CLEAN").alias("Total_Balance")).show(truncate = False)
```

```
12.Total balance in each account
+-------------+---------------------+
|Account_No   |Total_Balance        |
+-------------+---------------------+
|409000438611'|-2.4948657706833955E12|
|1196711'     |-1.60476498101275E13  |
|1196428'     |-8.1418498130721E13   |
|409000493210'|-3.2758495213209575E12|
|409000611074'|1.615533622E9         |
|409000425051'|-3.7721184116499877E9 |
|409000405747'|-2.4310804706700016E10|
|409000362497'|-5.2860004792808E13   |
|409000493201'|1.0420831829499985E9  |
|409000438620'|-7.122918679513586E12 |
+-------------+---------------------+
```

13. Customers with withdrawal amount more than 1 lakh

```
print("\n13. Customers with withdrawal > 1 lakh")
high_withdrawal = txn_df.filter(col("WITHDRAWAL_AMT_CLEAN") > 100000).select("Account_No", "WITHDRAWAL_AMT_CLEAN").distinct()
print("Count:", high_withdrawal.count())
high_withdrawal.show(truncate=False)
```

```
13. Customers with withdrawal > 1 lakh
Count: 10058
+-------------+--------------------+
|Account_No   |WITHDRAWAL_AMT_CLEAN|
+-------------+--------------------+
|409000611074'|274600.0            |
|409000493201'|1500000.0           |
|409000493201'|199604.27           |
|409000438620'|186604.0            |
|409000438620'|3.6675558E7         |
|1196711'     |7530283.0           |
|1196428'     |812361.0            |
|1196428'     |6348768.0           |
|1196428'     |3043151.63          |
|409000362497'|576954.0            |
|409000362497'|3423962.0           |
|409000362497'|3.144482503E7       |
|1196428'     |4441827.47          |
|409000611074'|145450.0            |
|409000493201'|119401.28           |
|1196711'     |628945.0            |
|1196428'     |289670.04           |
|409000362497'|3.483281361E7       |
|409000362497'|4.289763641E7       |
|409000362497'|2.678162613E7       |
+-------------+--------------------+
```