Install & Set Up Spark

```
# Install Spark and dependencies (only once per session)
!wget -q https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
!tar xf spark-3.5.0-bin-hadoop3.tgz
!pip install -q findspark

# Set environment variables
import os
os.environ["SPARK_HOME"] = "/content/spark-3.5.0-bin-hadoop3"
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"

# Initialize findspark
import findspark
findspark.init()
```

Start Spark Session

```
from pyspark.sql import SparkSession

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

```

Upload Your CSV Files

```
from google.colab import files
uploaded = files.upload()

```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving txn.csv to txn.csv
Saving credit card.csv to credit card.csv
```

Load DataFrames

```
loan_df = spark.read.option("header", True).option("inferSchema", True).csv("loan.csv")
credit_df = spark.read.option("header", True).option("inferSchema", True).csv("credit.csv")
txn_df = spark.read.option("header", True).option("inferSchema", True).csv("txn.csv")

```

Perform Analytics

Loan Dataset Use Cases:

Loan Dataset Analysis

This section focuses on analyzing the `loan_df` dataframe. The analysis includes:

- Displaying the first few rows and schema.
- Counting the number of loans in each category.
- Identifying customers based on loan amount, income, and returned cheques.

```
from pyspark.sql.functions import col, count, max, min, sum
loan_df.show(n=20, truncate=True, vertical=False)
```

```
+-----------+---+------+-------------------+--------------+-----------+------+-----------+-------------+------------------+--------
|Customer_ID|Age|Gender|         Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|     Loan Category|Loan Amou
+-----------+---+------+-------------------+--------------+-----------+------+-----------+-------------+------------------+--------
|    IB14001| 30|  MALE|       BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|           HOUSING| 10,00,00
|    IB14008| 44|  MALE|          PROFESSOR|       MARRIED|          6| 51000|      19999|            4|          SHOPPING|     50,6
|    IB14012| 30|FEMALE|            DENTIST|        SINGLE|          3| 58450|      27675|            5|        TRAVELLING|     75,6
|    IB14018| 29|  MALE|            TEACHER|       MARRIED|          5| 45767|      12787|            3|         GOLD LOAN|  6,00,00
|    IB14022| 34|  MALE|             POLICE|        SINGLE|          4| 43521|      11999|            3|        AUTOMOBILE|  2,00,00
|    IB14024| 55|FEMALE|              NURSE|       MARRIED|          6| 34999|      19888|            4|        AUTOMOBILE|     47,7
|    IB14025| 39|FEMALE|            TEACHER|       MARRIED|          6| 46619|      18675|            4|           HOUSING| 12,09,86
|    IB14027| 51|  MALE|     SYSTEM MANAGER|       MARRIED|          3| 49999|      19111|            5|       RESTAURANTS|     60,6
|    IB14029| 24|FEMALE|            TEACHER|        SINGLE|          3| 45008|      17454|            4|        AUTOMOBILE|  3,99,43
```

```
|    IB14031| 37|FEMALE|  SOFTWARE ENGINEER|       MARRIED|          5| 55999|      23999|            5|        AUTOMOBILE|     60,9
|    IB14032| 24|  MALE|       DATA ANALYST|        SINGLE|          4| 60111|      28999|            6|        AUTOMOBILE|     35,2
|    IB14034| 32|  MALE|   PRODUCT ENGINEER|       MARRIED|          6|  NULL|      29000|            7|COMPUTER SOFTWARES|     80,6
|    IB14037| 54|FEMALE|            TEACHER|       MARRIED|          5| 48099|      19999|            4|       RESTAURANTS|     30,9
|    IB14039| 45|  MALE|    ACCOUNT MANAGER|       MARRIED|          7| 45777|      18452|            4|         GOLD LOAN|  9,87,61
|    IB14041| 59|FEMALE|ASSISTANT PROFESSOR|       MARRIED|          4| 50999|      22999|            5|  EDUCATIONAL LOAN|  5,99,93
|    IB14042| 25|FEMALE|             DOCTOR|        SINGLE|          4| 60111|      27111|            5|        TRAVELLING| 12,90,92
|    IB14045| 31|  MALE|       STORE KEEPER|        SINGLE|          5| 40999|      11999|            3|       BOOK STORES|  1,67,65
|    IB14049| 49|  MALE|       BANK MANAGER|       MARRIED|          4| 45999|      14500|            4|        TRAVELLING|     79,9
|    IB14050| 56|  MALE|     CIVIL ENGINEER|       MARRIED|          4|  NULL|      13999|            3|           HOUSING| 10,65,57
|    IB14054| 58|FEMALE|             DOCTOR|       MARRIED|          5| 60000|      25000|            5|           HOUSING|  9,00,06
+-----------+---+------+-------------------+--------------+-----------+------+-----------+-------------+------------------+--------
only showing top 20 rows
```

```
loan_df.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)
```

```
loan_df.dtypes
```

```
[('Customer_ID', 'string'),
 ('Age', 'int'),
 ('Gender', 'string'),
 ('Occupation', 'string'),
 ('Marital Status', 'string'),
 ('Family Size', 'int'),
 ('Income', 'int'),
 ('Expenditure', 'int'),
 ('Use Frequency', 'int'),
 ('Loan Category', 'string'),
 ('Loan Amount', 'string'),
 ('Overdue', 'int'),
 (' Debt Record', 'string'),
 (' Returned Cheque', 'int'),
 (' Dishonour of Bill', 'int')]
```

```
loan_df = loan_df.withColumn("Loan Amount", col("Loan Amount").cast("int"))
```

```
# Number of loans in each category
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|
+------------------+-----+
```

```
# People who took more than ₹1L loan
loan_df.filter(col("Loan Amount") > 100000).count()
```

```
0
```

```
# People with income > ₹60,000
loan_df.filter(col("Income") > 60000).count()
```

```
198
```

```
# ≥2 returned cheques and income < ₹50,000
loan_df.filter((col(" Returned Cheque") >= 2) & (col("Income") < 50000)).count()
```

```
137
```

```
# ≥2 returned cheques and Single
loan_df.filter((col(" Returned Cheque") >= 2) & (col("Marital Status") == "Single")).count()
```

```
0
```

```
# Expenditure over ₹50,000
loan_df.filter(col("Expenditure") > 50000).count()
```

Credit Dataset Use Cases:

Credit Card Dataset Analysis

This section focuses on analyzing the `credit_df` dataframe. The analysis includes:

- Displaying the first few rows and schema.
- Identifying customers based on their geography and credit score/active member status.

```
credit_df.show(n=20, truncate=True, vertical=False)
```

```
+---------+----------+---------+-----------+---------+------+---+------+---------+-------------+--------------+---------------+----
|RowNumber|CustomerId|  Surname|CreditScore|Geography|Gender|Age|Tenure|  Balance|NumOfProducts|IsActiveMember|EstimatedSalary|Exite
+---------+----------+---------+-----------+---------+------+---+------+---------+-------------+--------------+---------------+----
|        1|  15634602| Hargrave|        619|   France|Female| 42|     2|      0.0|            1|             1|      101348.88|
|        2|  15647311|     Hill|        608|    Spain|Female| 41|     1| 83807.86|            1|             1|      112542.58|
|        3|  15619304|     Onio|        502|   France|Female| 42|     8| 159660.8|            3|             0|      113931.57|
|        4|  15701354|     Boni|        699|   France|Female| 39|     1|      0.0|            2|             0|       93826.63|
|        5|  15737888| Mitchell|        850|    Spain|Female| 43|     2|125510.82|            1|             1|        79084.1|
|        6|  15574012|      Chu|        645|    Spain|  Male| 44|     8|113755.78|            2|             0|      149756.71|
|        7|  15592531| Bartlett|        822|   France|  Male| 50|     7|      0.0|            2|             1|        10062.8|
|        8|  15656148|   Obinna|        376|  Germany|Female| 29|     4|115046.74|            4|             0|      119346.88|
|        9|  15792365|       He|        501|   France|  Male| 44|     4|142051.07|            2|             1|        74940.5|
|       10|  15592389|       H?|        684|   France|  Male| 27|     2|134603.88|            1|             1|       71725.73|
|       11|  15767821|   Bearce|        528|   France|  Male| 31|     6|102016.72|            2|             0|       80181.12|
|       12|  15737173|  Andrews|        497|    Spain|  Male| 24|     3|      0.0|            2|             0|       76390.01|
|       13|  15632264|      Kay|        476|   France|Female| 34|    10|      0.0|            2|             0|       26260.98|
|       14|  15691483|     Chin|        549|   France|Female| 25|     5|      0.0|            2|             0|      190857.79|
|       15|  15600882|    Scott|        635|    Spain|Female| 35|     7|      0.0|            2|             1|       65951.65|
|       16|  15643966|  Goforth|        616|  Germany|  Male| 45|     3|143129.41|            2|             1|       64327.26|
|       17|  15737452|    Romeo|        653|  Germany|  Male| 58|     1|132602.88|            1|             0|        5097.67|
|       18|  15788218|Henderson|        549|    Spain|Female| 24|     9|      0.0|            2|             1|       14406.41|
|       19|  15661507|  Muldrow|        587|    Spain|  Male| 45|     6|      0.0|            1|             0|      158684.81|
|       20|  15568982|      Hao|        726|   France|Female| 24|     6|      0.0|            2|             1|       54724.03|
+---------+----------+---------+-----------+---------+------+---+------+---------+-------------+--------------+---------------+----
only showing top 20 rows
```

```
credit_df.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)
```

```
credit_df.dtypes
```

```
[('RowNumber', 'int'),
 ('CustomerId', 'int'),
```

```
 ('Surname', 'string'),
 ('CreditScore', 'int'),
 ('Geography', 'string'),
 ('Gender', 'string'),
 ('Age', 'int'),
 ('Tenure', 'int'),
 ('Balance', 'double'),
 ('NumOfProducts', 'int'),
 ('IsActiveMember', 'int'),
 ('EstimatedSalary', 'double'),
 ('Exited', 'int')]
```

```
# Users from Spain
credit_df.filter(col("Geography") == "Spain").count()

```

```
2477
```

```
# Eligible & Active (e.g., CreditScore > 650 and IsActiveMember = 1)
credit_df.filter((col("CreditScore") > 650) & (col("IsActiveMember") == 1)).count()
```

```
2655
```

Transaction Dataset Use Cases:

Transaction Dataset Analysis

This section focuses on analyzing the `txn_df` dataframe. The analysis includes:

- Displaying the first few rows and schema.
- Calculating maximum withdrawal and deposit amounts.
- Calculating the total balance per account.
- Counting the number of transactions per date.
- Identifying customers who withdrew more than a certain amount.

```
txn_df.show(n=20, truncate=True, vertical=False)
```

```
+-------------+--------------------+----------+----------------+-------------+-----------+
|   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|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|            NULL|     500000.0|  6500000.0|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|            NULL|     500000.0|  7000000.0|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|            NULL|     500000.0|  7500000.0|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|            NULL|     500000.0|  8000000.0|
|409000611074'|FDRL/INTERNAL FUN...| 16-Aug-17|            NULL|     500000.0|  8500000.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|        133900.0|         NULL|  8366100.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|         18000.0|         NULL|  8348100.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|          5000.0|         NULL|  8343100.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|        195800.0|         NULL|  8147300.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|         81600.0|         NULL|  8065700.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|         41800.0|         NULL|  8023900.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|         98500.0|         NULL|  7925400.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|        143800.0|         NULL|  7781600.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|        331650.0|         NULL|  7449950.0|
|409000611074'|INDO GIBL Indiafo...| 16-Aug-17|        129000.0|         NULL|  7320950.0|
+-------------+--------------------+----------+----------------+-------------+-----------+
only showing top 20 rows
```

```
txn_df.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)
```

```
txn_df.dtypes
```

```
[('Account No', 'string'),
 ('TRANSACTION DETAILS', 'string'),
 ('VALUE DATE', 'string'),
```

```
 (' WITHDRAWAL AMT ', 'double'),
 (' DEPOSIT AMT ', 'double'),
 ('BALANCE AMT', 'double')]
```

```
# Max withdrawal amount
txn_df.select(max(" WITHDRAWAL AMT ")).show()
```

```
+---------------------+
|max( WITHDRAWAL AMT )|
+---------------------+
|        4.594475464E8|
+---------------------+
```

```
# Min withdrawal per account
txn_df.groupBy("Account No").agg(min(" WITHDRAWAL AMT ").alias("Min Withdrawal")).show()

```

```
+-------------+--------------+
|   Account No|Min Withdrawal|
+-------------+--------------+
|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|
+-------------+--------------+
```

```
# Max deposit per account
txn_df.groupBy("Account No").agg(max(" DEPOSIT AMT ").alias("Max Deposit")).show()

```

```
+-------------+-------------+
|   Account No|  Max Deposit|
+-------------+-------------+
|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'|    1000000.0|
|409000438620'|      5.448E8|
+-------------+-------------+
```

```
# Total balance per account
txn_df.groupBy("Account No").agg(sum("BALANCE AMT").alias("Total Balance")).show()

```

```
+-------------+--------------------+
|   Account No|       Total Balance|
+-------------+--------------------+
|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'|1.0420831829499985E9|
|409000438620'|-7.12291867951358...|
+-------------+--------------------+
```

```
# Number of transactions per date
txn_df.groupBy("VALUE DATE").agg(count("*").alias("Transaction Count")).show()

```

```
+----------+-----------------+
|VALUE DATE|Transaction Count|
+----------+-----------------+
| 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
```

```
# Customers who withdrew > ₹1L
txn_df.filter(col(" WITHDRAWAL AMT ") > 100000).select("Account No").distinct().show()
```

```
+-------------+
|   Account No|
+-------------+
|409000438611'|
|     1196711'|
|     1196428'|
|409000493210'|
|409000611074'|
|409000425051'|
|409000405747'|
|409000362497'|
|409000493201'|
|409000438620'|
+-------------+
```

This notebook analyzes online banking data using Apache Spark. It covers three datasets: loan, credit card, and transaction data.

The analysis includes:

- Loading and inspecting the dataframes.
- Performing basic queries and aggregations on each dataset.
- Identifying potential insights from the data.