A

Project On

“Online Transaction Fraud Detection”

Submitted By:

Project Team 11

Gaurav Singh (PL)

B V Sailesh

Bhupesh Ghadigaonkar

Divyani Bairam

Shubham Bele

In partial fulfilment of

the requirements for the award of the degree of

Post Graduate Diploma

In

Big Data Analytics

(PG-DBDA)

Project Guide: Project Coordinator:

Swapnil Adhav **Vineeta Singh**

Abstract

We have done data analysis, data visualisation and prediction of data for this project. The dataset we used is of Online Transactions Fraud Detection which we gathered from the Kaggle.

The whole project is based on Pandas used in Juypter Notebook on Local Host and Pyspark is used as Google colaboratory and Google Drive as cloud platform and Power BI. For performing big data analytics we used Google Colaboratory Pyspark service. We used PySpark by creating an Environment in google colabratory. For data visualisation we used Power BI desktop. We used various visuals like bar charts, donut charts, cards, map, slicers, etc. and gathered some insights about the data.

For machine learning we used a python notebook in Jupyter Lab. We did EDA on the data and after that we did prediction using some machine learning algorithms available in python libraries.
This whole process and the insights about the data are explained in detail in this report.

Contents

Chapter I: INTRODUCTION

1.1 Introduction of Project

Online payment fraud detection is a common problem that businesses face when processing payments online. The goal is to detect fraudulent transactions before they are processed and approved, in order to prevent financial losses and protect the business and its customers.

In this project we downloaded the ‘Online Transactions Fraud Detection’ data set from kaggle. We used pandas for machine learning and PySpark for getting insights for the same purpose. By analysing the data we gathered some insights about it and we have described the same in this project report. For visualising the same analysed data we used Microsoft Power BI Desktop and created interactive dashboards. We also did prediction using the same dataset. For that we used various machine learning algorithms available in python libraries. The predicted data and information are described in the report.

1.2 Introduction to dataset

The available fields are as follows:

- **step**: represents a unit of time where 1 step equals 1 hour
- **type**: type of online transaction
- **amount**: the amount of the transaction
- **nameOrig**: customer starting the transaction
- **oldbalanceOrg**: balance before the transaction
- **newbalanceOrig**: balance after the transaction
- **nameDest**: recipient of the transaction
- **oldbalanceDest**: initial balance of recipient before the transaction
- **newbalanceDest**: the new balance of recipient after the transaction
- **isFraud**: fraud transaction
- **isFlaggedFraud**: Flagged fraud transaction

1.3 Problem Statement

Online payment fraud detection is a common problem that businesses face when processing payments online. Fraudulent transactions can occur when someone uses deceptive or illegal practices to obtain money or assets from individuals or businesses through online channels such as websites, mobile apps, or other online platforms. The goal is to detect fraudulent transactions before they are processed and approved, in order to prevent financial losses and protect the business and its customers.

1.4 Objectives

• Analysing the data according to different type of transactions.
• Analysing the data for different sources and destinations where the fraud had actually happened and giving the insights through some pre-defined questions.

• Visualizing the data for better understanding

• Creating interactive dashboards

• Predicting the future data

Chapter II: BIG DATA ANALYTICS

2.1 Data Collection

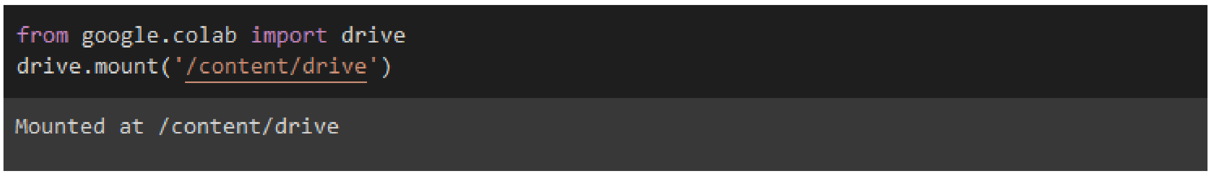
2.1.1 Dataset Source

We have chosen kaggle as a source for the above dataset

2.2 Creating Environment and required software using Google Colab:

To store the data we are mounting the data in google drive for use.

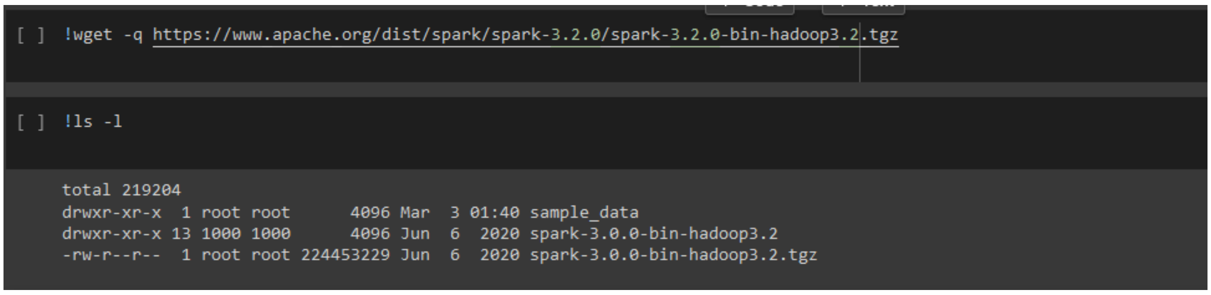
We uploaded the data file in google drive in Project folder



```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

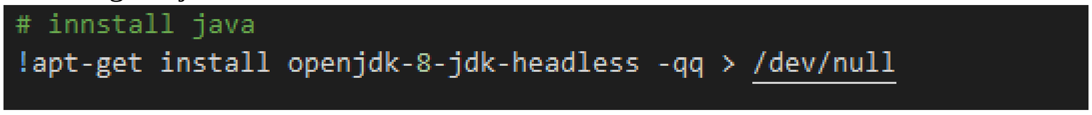
Now creating the environment in google colab for pyspark

1. We downloaded the Hadoop pyspark in the server for futher use and checking the files.



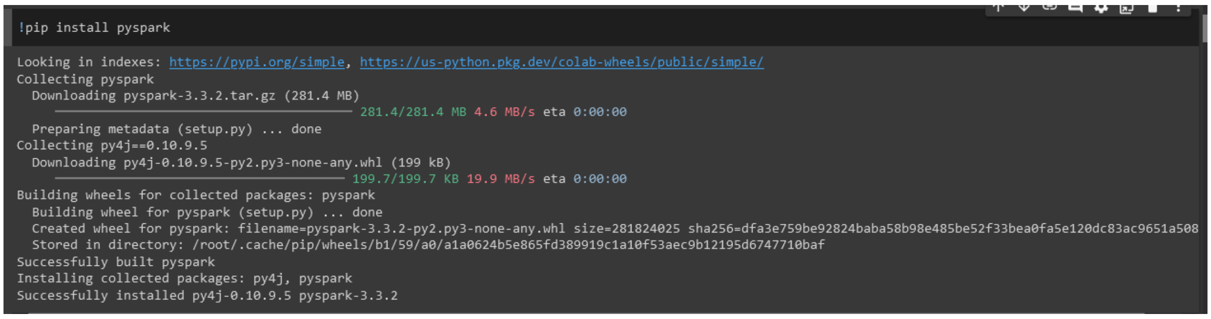
```
[ ]  !wget -q https://www.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz

[ ]  !ls -l

     total 219204
     drwxr-xr-x  1 root root      4096 Mar  3 01:40 sample_data
     drwxr-xr-x 13 1000 1000      4096 Jun  6  2020 spark-3.0.0-bin-hadoop3.2
     -rw-r--r--  1 root root 224453229 Jun  6  2020 spark-3.0.0-bin-hadoop3.2.tgz
```

2. Installing the java in the server



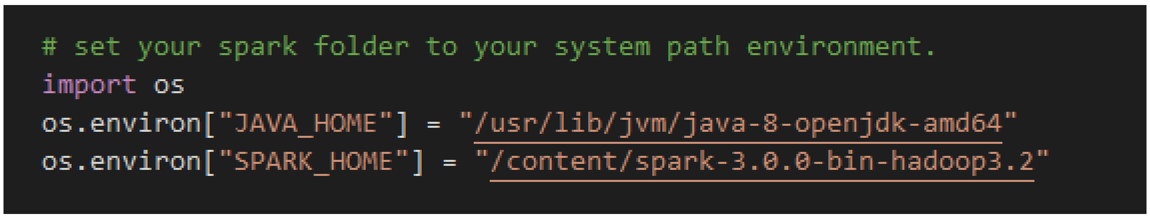
```
# innstall java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

3. Installing the pyspark



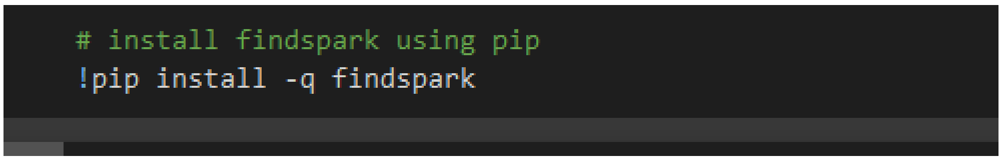
```
!pip install pyspark

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.3.2.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 4.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 19.9 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.2-py2.py3-none-any.whl size=281824025 sha256=dfa3e759be92824baba58b98e485be52f33bea0fa5e120dc83ac9651a508
  Stored in directory: /root/.cache/pip/wheels/b1/59/a0/a1a0624b5e865fd389919c1a10f53aec9b12195d6747710baf
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.2
```

4. Setting the environment for java and pyspark



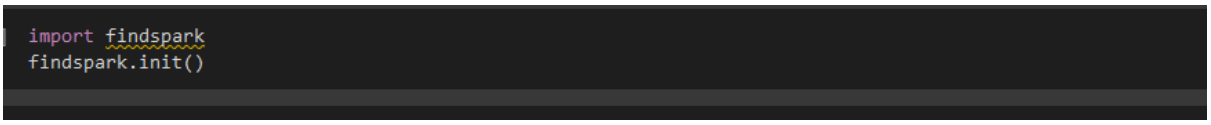
```
# set your spark folder to your system path environment.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"
```

5. Installing the findspark



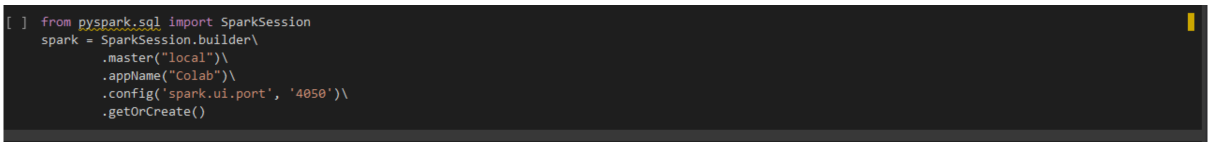
```
# install findspark using pip
!pip install -q findspark
```

6. Importing the findspark



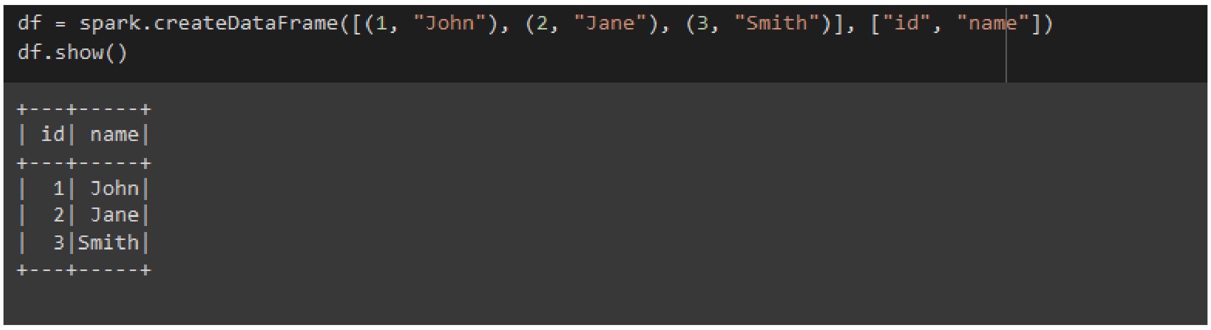
```
import findspark
findspark.init()
```

7. Setting up session for spark sql



```
[ ]  from pyspark.sql import SparkSession
     spark = SparkSession.builder\
             .master("local")\
             .appName("Colab")\
             .config('spark.ui.port', '4050')\
             .getOrCreate()
```

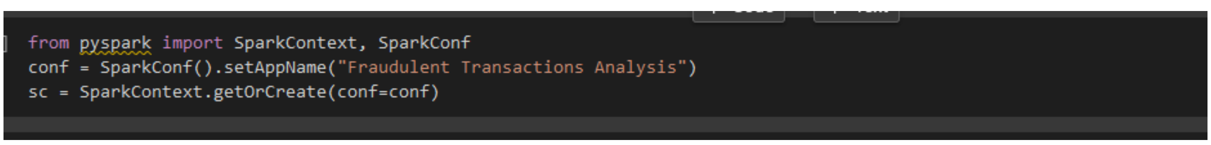
8. Checking the spark sql by using a dummy



```
df = spark.createDataFrame([(1, "John"), (2, "Jane"), (3, "Smith")], ["id", "name"])
df.show()
```

```
+---+-----+
| id| name|
+---+-----+
|  1| John|
|  2| Jane|
|  3|Smith|
+---+-----+
```

9. Setting session for spark rdd



```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("Fraudulent Transactions Analysis")
sc = SparkContext.getOrCreate(conf=conf)
```

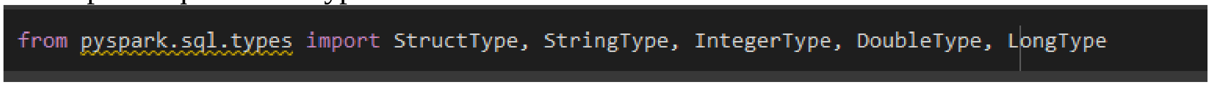
2.3 Analysis using Spark SQL

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as a distributed SQL query engine. It enables unmodified Hadoop Hive queries to run up to 100x faster on existing deployments and data.

Spark SQL brings native support for SQL to Spark and streamlines the process of querying data stored both in RDDs (Spark's distributed datasets) and in external sources. Spark SQL conveniently blurs the lines between RDDs and relational tables.

Below are the steps to create a table in PySpark

1. Launch PySpark
2. Import required datatypes



```
from pyspark.sql.types import StructType, StringType, IntegerType, DoubleType, LongType
```

3. Creating a schema with required column names and their datatypes.

```
schema2 = StructType().add("step",IntegerType(),True).add("type",StringType(),True).add("amount",DoubleType(),True).add("origin",StringType(),True).add("sender_old_balance",DoubleType(),True).add("sender_new_balance",DoubleType(),True).add("destination",StringType(),True).add("receiver_old_balance",DoubleType(),True).add("receiver_new_balance",DoubleType(),True).add("isfraud",IntegerType(),True).add("isFlaggedFraud",IntegerType(),True)
```

[Here True means if the column allows null values, true for nullable, and false for not nullable.]

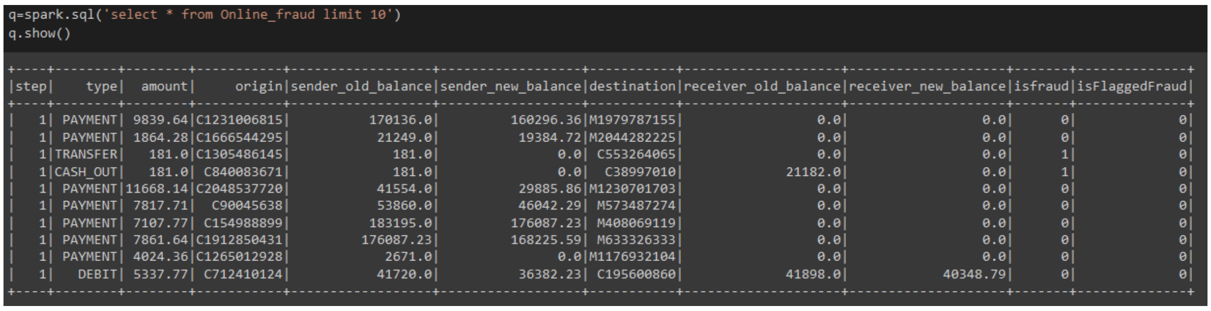
4. Reading the data from .csv file –

```
df_with_schema2 = spark.read.format("csv").option("header", "True").schema(schema2).load("/content/drive/MyDrive/project/PS_20174392719_1491204439457_log.csv")
```

5. For performing SQL queries on a schema we have to create a temporary table. This temporary table is automatically dropped after the session ends. To register a 'temptable' –

```
df_with_schema2.registerTempTable("Online_fraud")
```

6. Checking through temporary table:



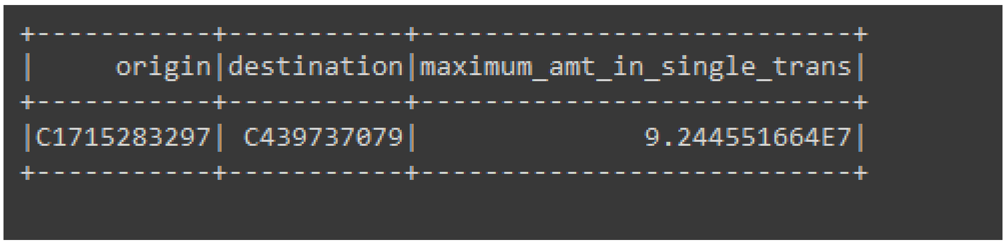
```
q=spark.sql('select * from Online_fraud limit 10')
q.show()
```

```
+----+--------+--------+-----------+------------------+------------------+-----------+--------------------+--------------------+-------+--------------+
|step|    type|  amount|     origin|sender_old_balance|sender_new_balance|destination|receiver_old_balance|receiver_new_balance|isfraud|isFlaggedFraud|
+----+--------+--------+-----------+------------------+------------------+-----------+--------------------+--------------------+-------+--------------+
|   1| PAYMENT| 9839.64|C1231006815|          170136.0|         160296.36|M1979787155|                 0.0|                 0.0|      0|             0|
|   1| PAYMENT| 1864.28|C1666544295|           21249.0|          19384.72|M2044282225|                 0.0|                 0.0|      0|             0|
|   1|TRANSFER|   181.0|C1305486145|             181.0|               0.0| C553264065|                 0.0|                 0.0|      1|             0|
|   1|CASH_OUT|   181.0| C840083671|             181.0|               0.0|  C38997010|             21182.0|                 0.0|      1|             0|
|   1| PAYMENT|11668.14|C2048537720|           41554.0|          29885.86|M1230701703|                 0.0|                 0.0|      0|             0|
|   1| PAYMENT| 7817.71|  C90045638|           53860.0|          46042.29| M573487274|                 0.0|                 0.0|      0|             0|
|   1| PAYMENT| 7107.77| C154988899|          183195.0|         176087.23| M408069119|                 0.0|                 0.0|      0|             0|
|   1| PAYMENT| 7861.64|C1912850431|         176087.23|         168225.59| M633326333|                 0.0|                 0.0|      0|             0|
|   1| PAYMENT| 4024.36|C1265012928|            2671.0|               0.0|M1176932104|                 0.0|                 0.0|      0|             0|
|   1|   DEBIT| 5337.77| C712410124|           41720.0|          36382.23| C195600860|             41898.0|            40348.79|      0|             0|
+----+--------+--------+-----------+------------------+------------------+-----------+--------------------+--------------------+-------+--------------+
```

After registering temp table we can perform the SQL queries on the table. We had to answer some business questions listed below. We have answered them using Spark SQL.

1. What is the maximum amount that was transferred in a single transaction from and to which account?
   q1=spark.sql('select origin,destination,amount as maximum_amt_in_single_trans from Online_fraud order by amount desc limit 1')
   q1.show()

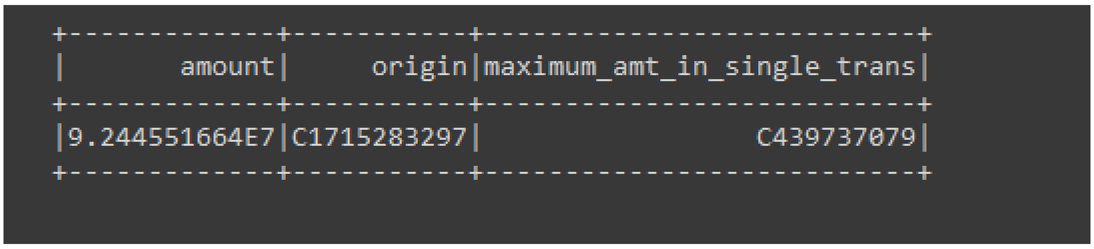
Output:



```
+-----------+-----------+--------------------------+
|     origin|destination|maximum_amt_in_single_trans|
+-----------+-----------+--------------------------+
|C1715283297| C439737079|               9.244551664E7|
+-----------+-----------+--------------------------+
```

2. What is the maximum amount that was transferred in a single transaction and how many such transactions were flagged as fraudulent?
   q2=spark.sql('select amount,origin,destination as maximum_amt_in_single_trans from Online_fraud order by amount desc limit 1')
   q2.show()

Output:

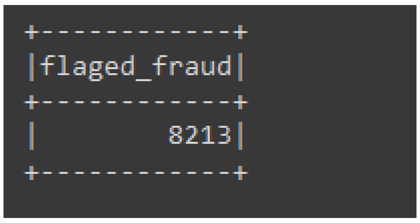


```
+-------------+-----------+--------------------------+
|       amount|     origin|maximum_amt_in_single_trans|
+-------------+-----------+--------------------------+
|9.244551664E7|C1715283297|                C439737079|
+-------------+-----------+--------------------------+
```

3. How many customers initiated transactions that were flagged as fraudulent?
   q3=spark.sql('select count(origin) as flaged_fraud from online_fraud where isfraud=1')

   q3.show()

Output:

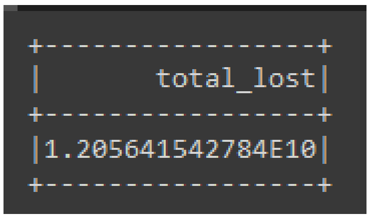


```
+------------+
|flaged_fraud|
+------------+
|        8213|
+------------+
```

4. What is the total amount of money lost due to fraudulent activity?
   q4=spark.sql('SELECT SUM(amount) as total_lost FROM online_fraud WHERE isFraud = 1')

   q4.show()

Output:

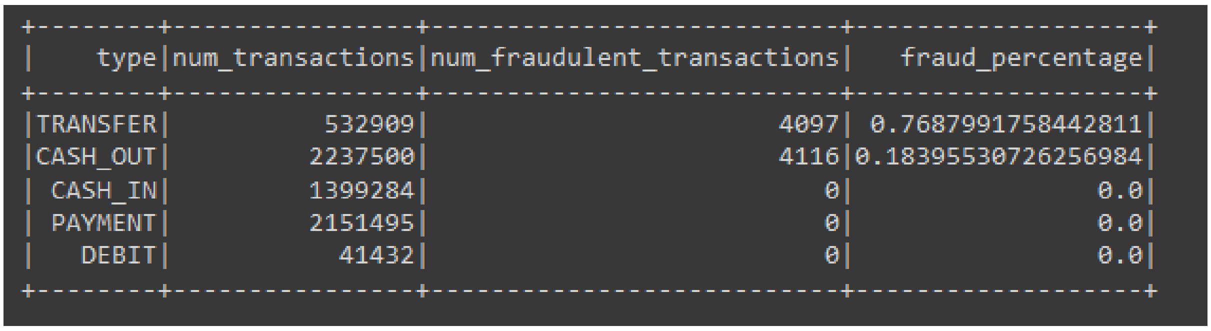


```
+-----------------+
|       total_lost|
+-----------------+
|1.205641542784E10|
+-----------------+
```

5. Which type of transaction has the highest percentage of fraudulent activity?

q5=spark.sql('SELECT type, COUNT(*) as num_transactions, COUNT(CASE WHEN isFraud = 1 THEN 1 END) as num_fraudulent_transactions, (COUNT(CASE WHEN isFraud = 1 THEN 1 END) / COUNT(*))*100 as fraud_percentage FROM online_fraud GROUP BY type ORDER BY fraud_percentage DESC ;')
q5.show()

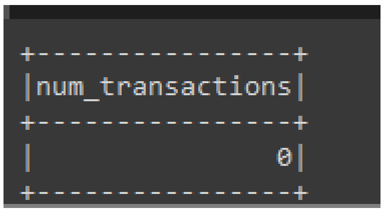
Output:



```
+--------+----------------+---------------------------+-------------------+
|    type|num_transactions|num_fraudulent_transactions|   fraud_percentage|
+--------+----------------+---------------------------+-------------------+
|TRANSFER|          532909|                       4097| 0.7687991758442811|
|CASH_OUT|         2237500|                       4116|0.18395530726256984|
| CASH_IN|         1399284|                          0|                0.0|
| PAYMENT|         2151495|                          0|                0.0|
|   DEBIT|           41432|                          0|                0.0|
+--------+----------------+---------------------------+-------------------+
```

6. How many transactions involved customers with negative balances before the transaction?
q6=spark.sql('SELECT COUNT(*) as num_transactions FROM online_fraud WHERE sender_old_balance < 0 OR receiver_old_balance < 0')
q6.show()

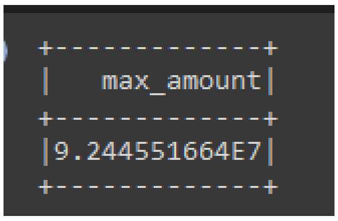
Output:



```
+----------------+
|num_transactions|
+----------------+
|               0|
+----------------+
```

7. What is the maximum amount of money transferred in a single transaction that was not flagged as fraudulent?
q7=spark.sql("SELECT MAX(amount) AS max_amount FROM online_fraud WHERE isFraud = 0")
q7.show()

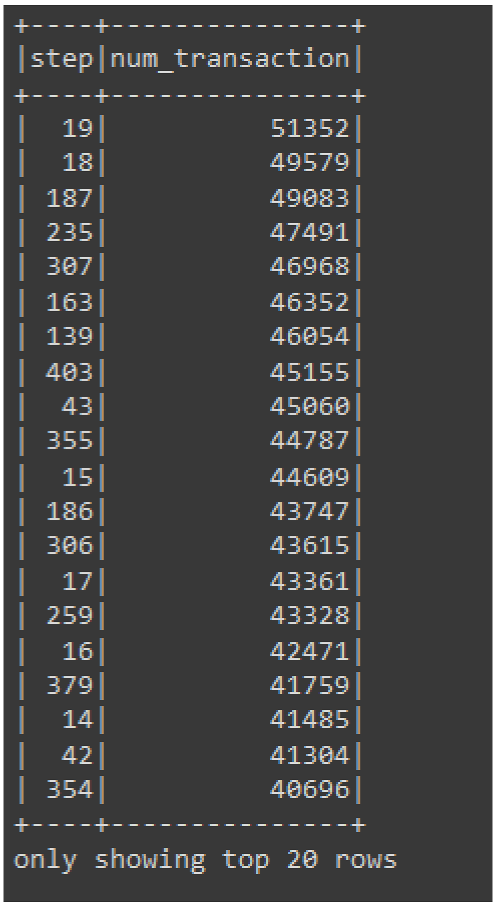
Output:



```
+-------------+
|   max_amount|
+-------------+
|9.244551664E7|
+-------------+
```

8. How many transactions were processed in each step?
q8=spark.sql("SELECT step, COUNT(*) AS num_transaction FROM online_fraud GROUP BY step order by count(*) desc")
q8.show(20)

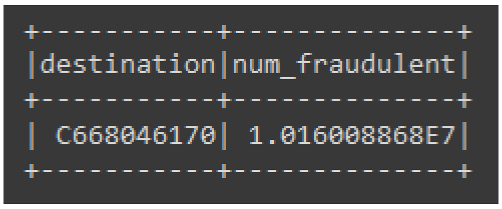
Output:



```
+----+---------------+
|step|num_transaction|
+----+---------------+
|  19|          51352|
|  18|          49579|
| 187|          49083|
| 235|          47491|
| 307|          46968|
| 163|          46352|
| 139|          46054|
| 403|          45155|
|  43|          45060|
| 355|          44787|
|  15|          44609|
| 186|          43747|
| 306|          43615|
|  17|          43361|
| 259|          43328|
|  16|          42471|
| 379|          41759|
|  14|          41485|
|  42|          41304|
| 354|          40696|
+----+---------------+
only showing top 20 rows
```

9. Which name dest is credited with most amount in case of fraud?
   q11=spark.sql("SELECT destination, sum(amount) AS num_fraudulent FRO M online_fraud WHERE isFraud = 1 GROUP BY destination ORDER BY nu m_fraudulent DESC LIMIT 1")
   q11.show()

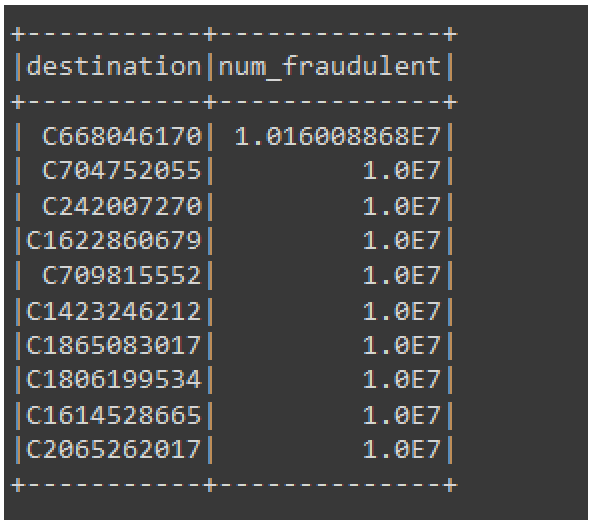
Output:



```
+-----------+--------------+
|destination|num_fraudulent|
+-----------+--------------+
| C668046170| 1.016008868E7|
+-----------+--------------+
```

10. Top 10 destaccount which has maximum amount in case of fraud ?
    q12=spark.sql("SELECT destination, sum(amount) AS num_fraudulent FRO M online_fraud WHERE isFraud = 1 GROUP BY destination ORDER BY nu m_fraudulent DESC LIMIT 10")
    q12.show()

Output:



```
+-----------+--------------+
|destination|num_fraudulent|
+-----------+--------------+
| C668046170| 1.016008868E7|
| C704752055|         1.0E7|
| C242007270|         1.0E7|
|C1622860679|         1.0E7|
| C709815552|         1.0E7|
|C1423246212|         1.0E7|
|C1865083017|         1.0E7|
|C1806199534|         1.0E7|
|C1614528665|         1.0E7|
|C2065262017|         1.0E7|
+-----------+--------------+
```

2.4 Analysis using PySpark

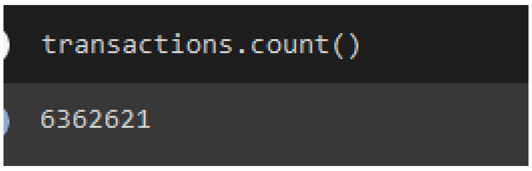
PySpark is a Python API for Apache Spark to process datasets in a distributed cluster. It is written in Python to run a Python application using Apache Spark Capabilities.
PySpark is very well used in Data Science and Machine Learning community as there are many widely used data science libraries written in Python.
PySpark has modules such as Resilient Distributed Dataset (RDD), Dataframe, Pyspark streaming, MLib etc.

For performing PySpark operations first we created RDD and loading the data.

```
dataset = sc.textFile("/content/drive/MyDrive/project/PS_20174392719_1491204439457_log.csv")
```

Checking the above by running the

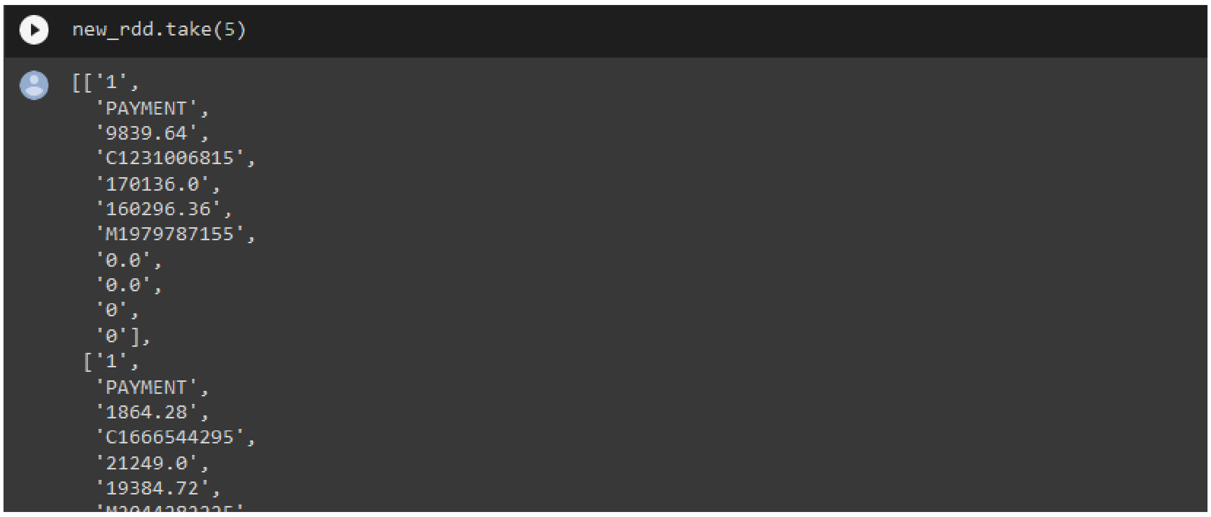
```
transactions = dataset.map(lambda line: line.split(","))
```



```
transactions.count()

6362621
```

Then we removed header, for that we performed following operations:

```
header = transactions.first()
new_rdd = transactions.filter(lambda x: x != header)
```



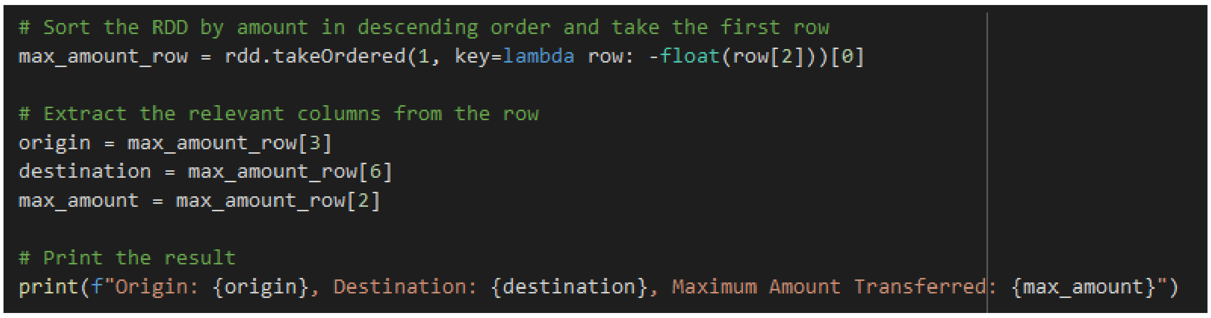
```
new_rdd.take(5)
```

```
[['1',
  'PAYMENT',
  '9839.64',
  'C1231006815',
  '170136.0',
  '160296.36',
  'M1979787155',
  '0.0',
  '0.0',
  '0',
  '0'],
 ['1',
  'PAYMENT',
  '1864.28',
  'C1666544295',
  '21249.0',
  '19384.72',
```

```
header = dataset.first()
rdd = dataset.filter(lambda row: row != header)
rdd = rdd.map(lambda row: row.split(","))
rdd = rdd.map(lambda row: (int(row[0]), row[1], float(row[2]), row[3], float(row[4]), float(ro
w[5]), row[6], float(row[7]), float(row[8]), int(row[9]), int(row[10])))
```

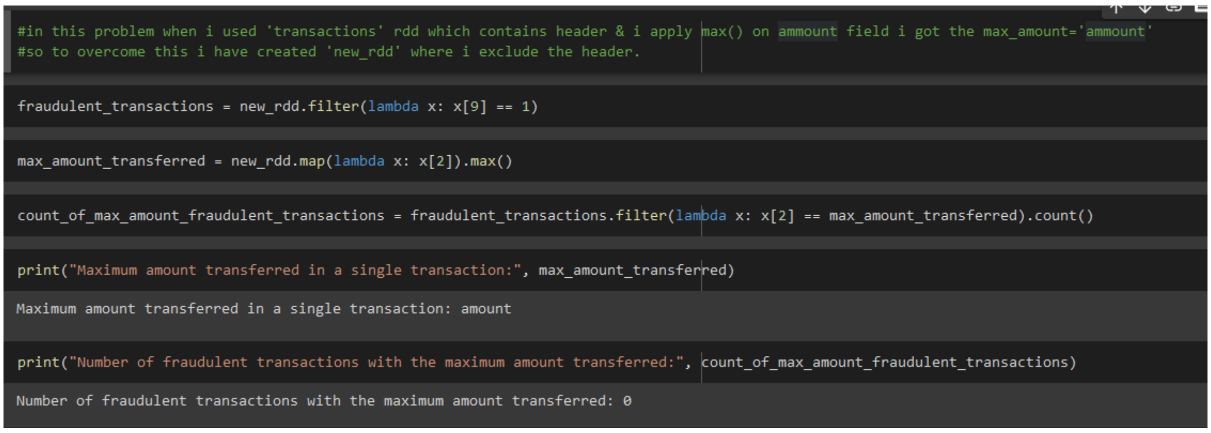
Following business problems we solved using pyspark.

1. What is the maximum amount that was transferred in a single transaction from and to which account?



```
# Sort the RDD by amount in descending order and take the first row
max_amount_row = rdd.takeOrdered(1, key=lambda row: -float(row[2]))[0]

# Extract the relevant columns from the row
origin = max_amount_row[3]
destination = max_amount_row[6]
max_amount = max_amount_row[2]

# Print the result
print(f"Origin: {origin}, Destination: {destination}, Maximum Amount Transferred: {max_amount}")
```

2. What is the maximum amount that was transferred in a single transaction and how many such transactions were flagged as fraudulent?



```
#in this problem when i used 'transactions' rdd which contains header & i apply max() on ammount field i got the max_amount='ammount'
#so to overcome this i have created 'new_rdd' where i exclude the header.
```

```
fraudulent_transactions = new_rdd.filter(lambda x: x[9] == 1)
```

```
max_amount_transferred = new_rdd.map(lambda x: x[2]).max()
```

```
count_of_max_amount_fraudulent_transactions = fraudulent_transactions.filter(lambda x: x[2] == max_amount_transferred).count()
```

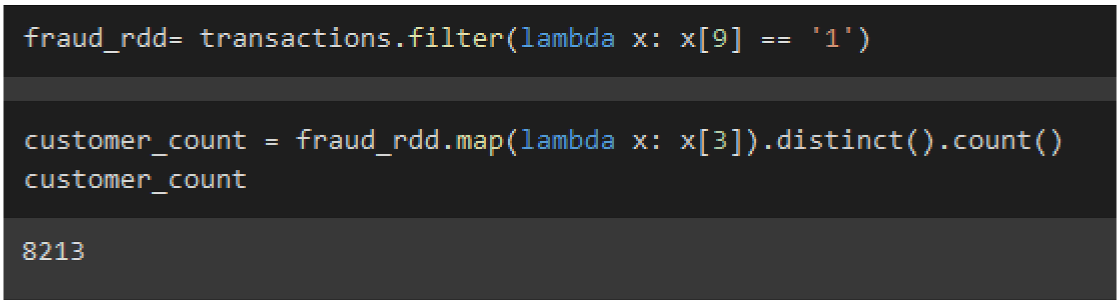
```
print("Maximum amount transferred in a single transaction:", max_amount_transferred)
```

```
Maximum amount transferred in a single transaction: amount
```

```
print("Number of fraudulent transactions with the maximum amount transferred:", count_of_max_amount_fraudulent_transactions)
```

```
Number of fraudulent transactions with the maximum amount transferred: 0
```

3. How many customers initiated transactions that were flagged as fraudulent?

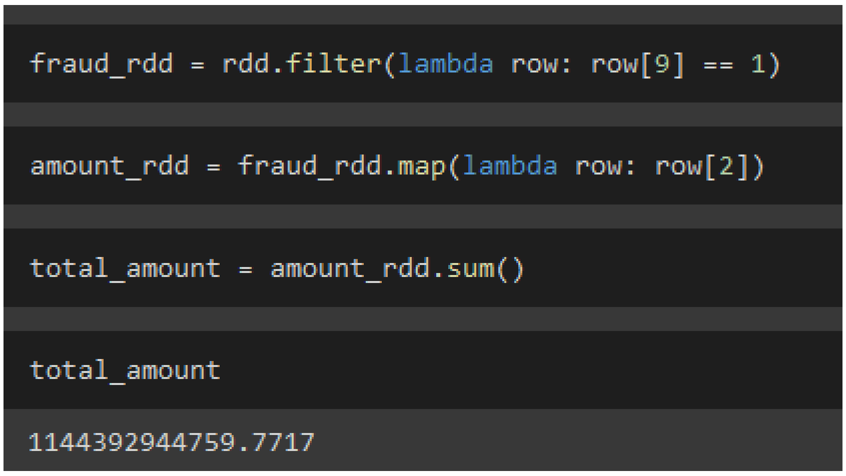


```
fraud_rdd= transactions.filter(lambda x: x[9] == '1')
```

```
customer_count = fraud_rdd.map(lambda x: x[3]).distinct().count()
customer_count
```

```
8213
```

4. What is the total amount of money lost due to fraudulent activity?



```
fraud_rdd = rdd.filter(lambda row: row[9] == 1)
```

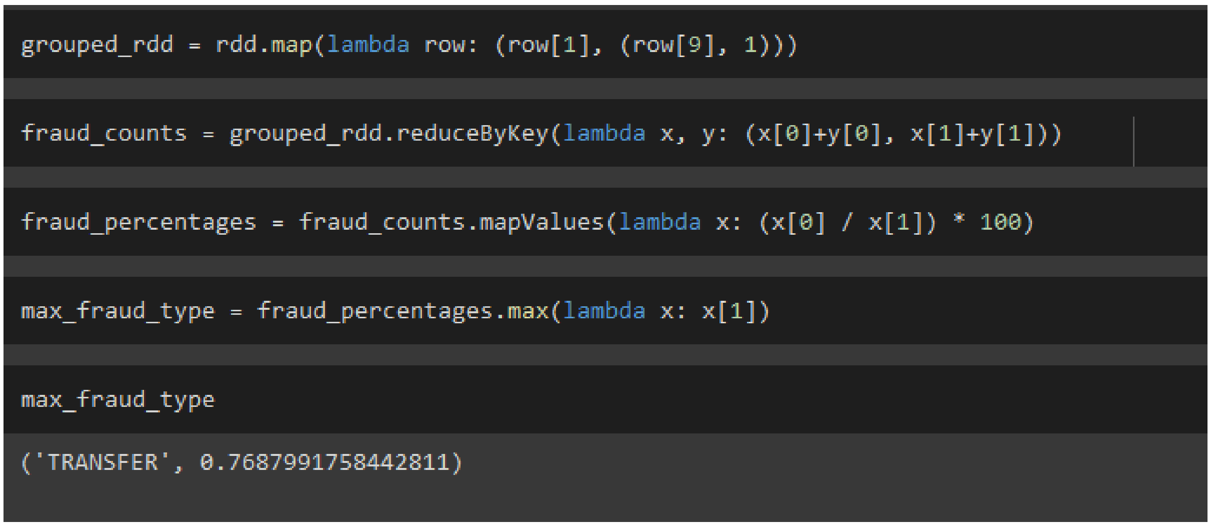
```
amount_rdd = fraud_rdd.map(lambda row: row[2])
```

```
total_amount = amount_rdd.sum()
```

```
total_amount
```

```
1144392944759.7717
```

5. Which type of transaction has the highest percentage of fraudulent activity?



```
grouped_rdd = rdd.map(lambda row: (row[1], (row[9], 1)))
```

```
fraud_counts = grouped_rdd.reduceByKey(lambda x, y: (x[0]+y[0], x[1]+y[1]))
```

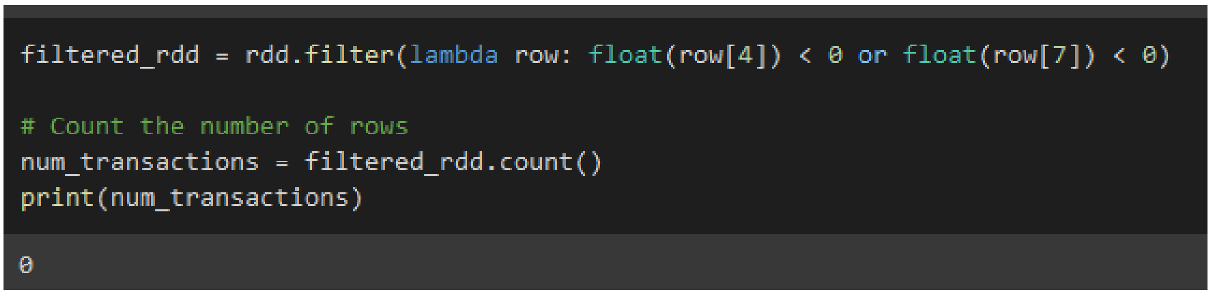
```
fraud_percentages = fraud_counts.mapValues(lambda x: (x[0] / x[1]) * 100)
```

```
max_fraud_type = fraud_percentages.max(lambda x: x[1])
```

```
max_fraud_type
```

```
('TRANSFER', 0.7687991758442811)
```

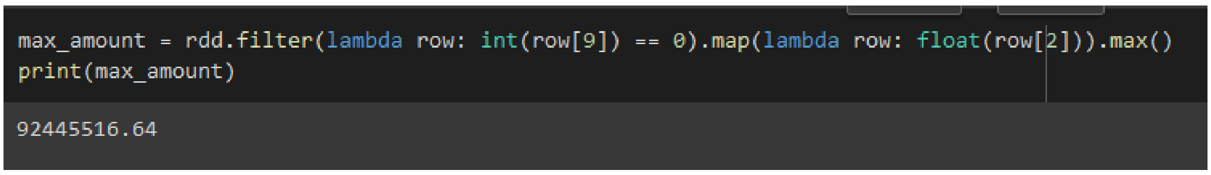
6. How many transactions involved customers with negative balances before the transaction?



```
filtered_rdd = rdd.filter(lambda row: float(row[4]) < 0 or float(row[7]) < 0)

# Count the number of rows
num_transactions = filtered_rdd.count()
print(num_transactions)
```

```
0
```

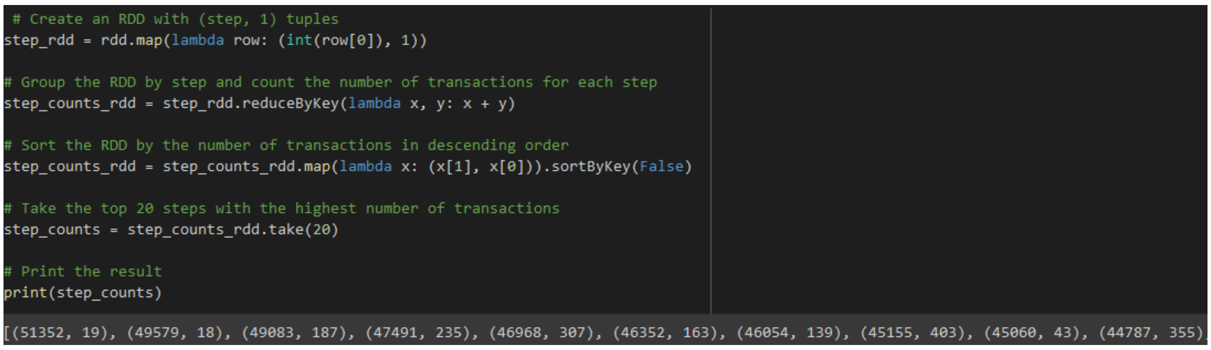
7. What is the maximum amount of money transferred in a single transaction that was not flagged as fraudulent?



```
max_amount = rdd.filter(lambda row: int(row[9]) == 0).map(lambda row: float(row[2])).max()
print(max_amount)
```

```
92445516.64
```

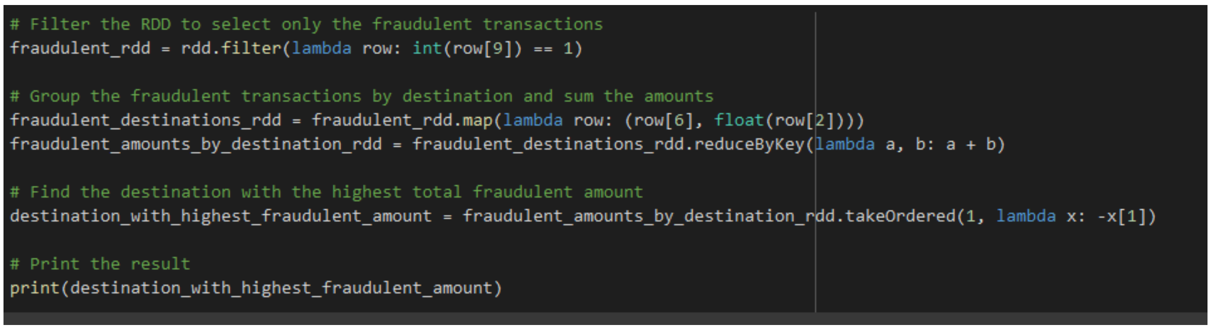
8. How many transactions were processed in each step?



```
 # Create an RDD with (step, 1) tuples
step_rdd = rdd.map(lambda row: (int(row[0]), 1))

# Group the RDD by step and count the number of transactions for each step
step_counts_rdd = step_rdd.reduceByKey(lambda x, y: x + y)

# Sort the RDD by the number of transactions in descending order
step_counts_rdd = step_counts_rdd.map(lambda x: (x[1], x[0])).sortByKey(False)

# Take the top 20 steps with the highest number of transactions
step_counts = step_counts_rdd.take(20)

# Print the result
print(step_counts)
```

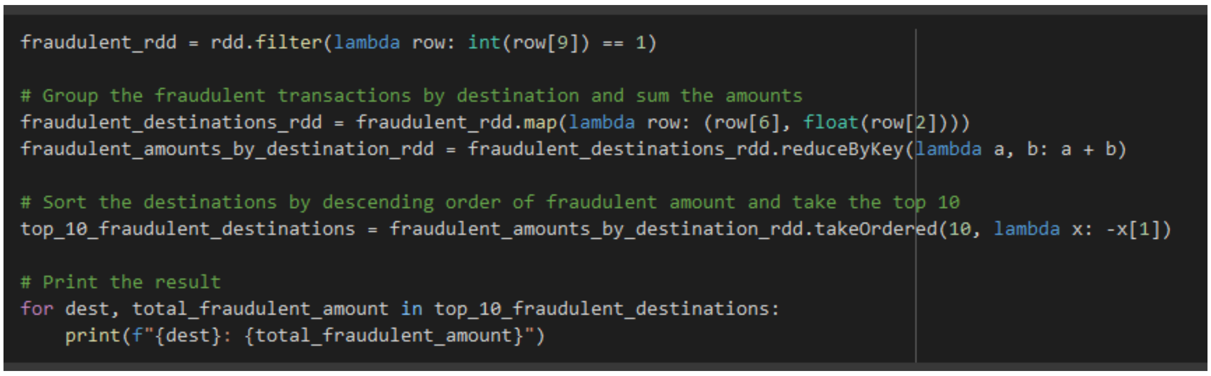
```
[(51352, 19), (49579, 18), (49083, 187), (47491, 235), (46968, 307), (46352, 163), (46054, 139), (45155, 403), (45060, 43), (44787, 355)
```

9. Which name dest is credited with most amount in case of fraud?



```
# Filter the RDD to select only the fraudulent transactions
fraudulent_rdd = rdd.filter(lambda row: int(row[9]) == 1)

# Group the fraudulent transactions by destination and sum the amounts
fraudulent_destinations_rdd = fraudulent_rdd.map(lambda row: (row[6], float(row[2])))
fraudulent_amounts_by_destination_rdd = fraudulent_destinations_rdd.reduceByKey(lambda a, b: a + b)

# Find the destination with the highest total fraudulent amount
destination_with_highest_fraudulent_amount = fraudulent_amounts_by_destination_rdd.takeOrdered(1, lambda x: -x[1])

# Print the result
print(destination_with_highest_fraudulent_amount)
```

10. Top 10 destaccount which has maximum amount in case of fraud ?

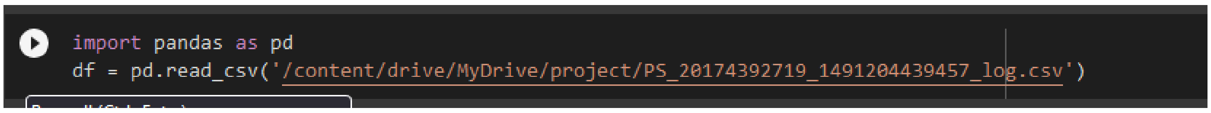


```
fraudulent_rdd = rdd.filter(lambda row: int(row[9]) == 1)

# Group the fraudulent transactions by destination and sum the amounts
fraudulent_destinations_rdd = fraudulent_rdd.map(lambda row: (row[6], float(row[2])))
fraudulent_amounts_by_destination_rdd = fraudulent_destinations_rdd.reduceByKey(lambda a, b: a + b)

# Sort the destinations by descending order of fraudulent amount and take the top 10
top_10_fraudulent_destinations = fraudulent_amounts_by_destination_rdd.takeOrdered(10, lambda x: -x[1])

# Print the result
for dest, total_fraudulent_amount in top_10_fraudulent_destinations:
    print(f"{dest}: {total_fraudulent_amount}")
```

2.5 Analysis using Pandas

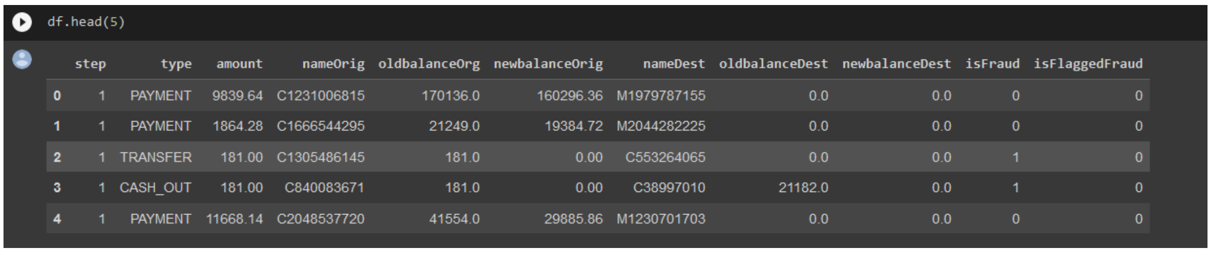
Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Importing the pandas



```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/project/PS_20174392719_1491204439457_log.csv')
```

Checking for data

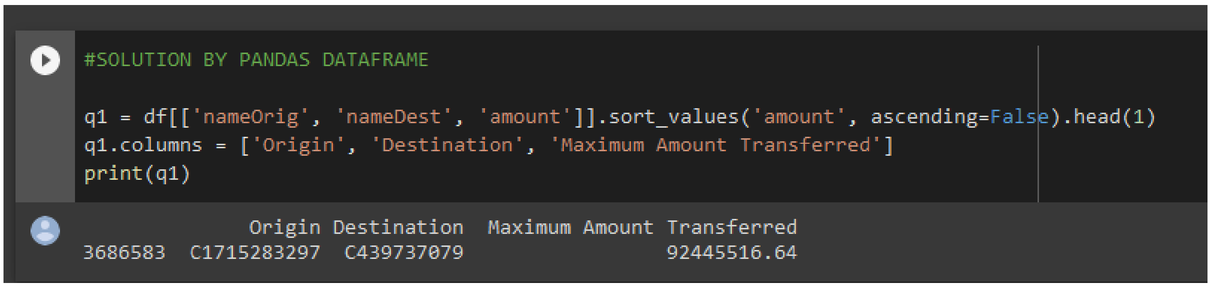


```
df.head(5)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Below are the following business question solved using pandas:

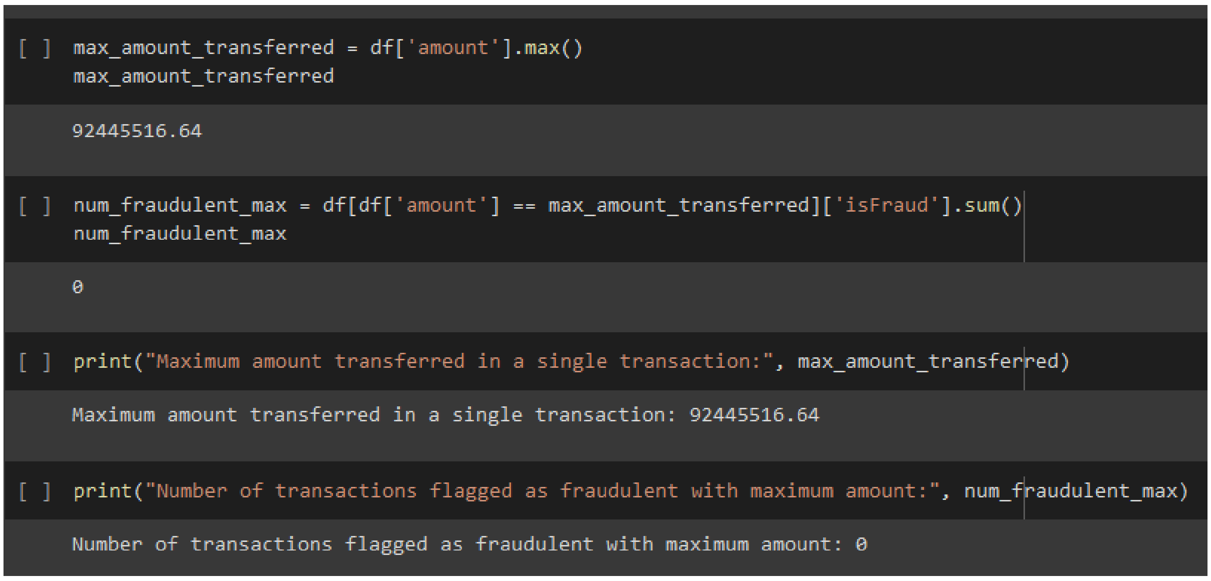
1. What is the maximum amount that was transferred in a single transaction from and to which account?



```
#SOLUTION BY PANDAS DATAFRAME

q1 = df[['nameOrig', 'nameDest', 'amount']].sort_values('amount', ascending=False).head(1)
q1.columns = ['Origin', 'Destination', 'Maximum Amount Transferred']
print(q1)
```

```
                Origin Destination  Maximum Amount Transferred
3686583   C1715283297  C439737079                 92445516.64
```

2. What is the maximum amount that was transferred in a single transaction and how many such transactions were flagged as fraudulent?



```
[ ] max_amount_transferred = df['amount'].max()
    max_amount_transferred
```

```
92445516.64
```

```
[ ] num_fraudulent_max = df[df['amount'] == max_amount_transferred]['isFraud'].sum()
    num_fraudulent_max
```

```
0
```

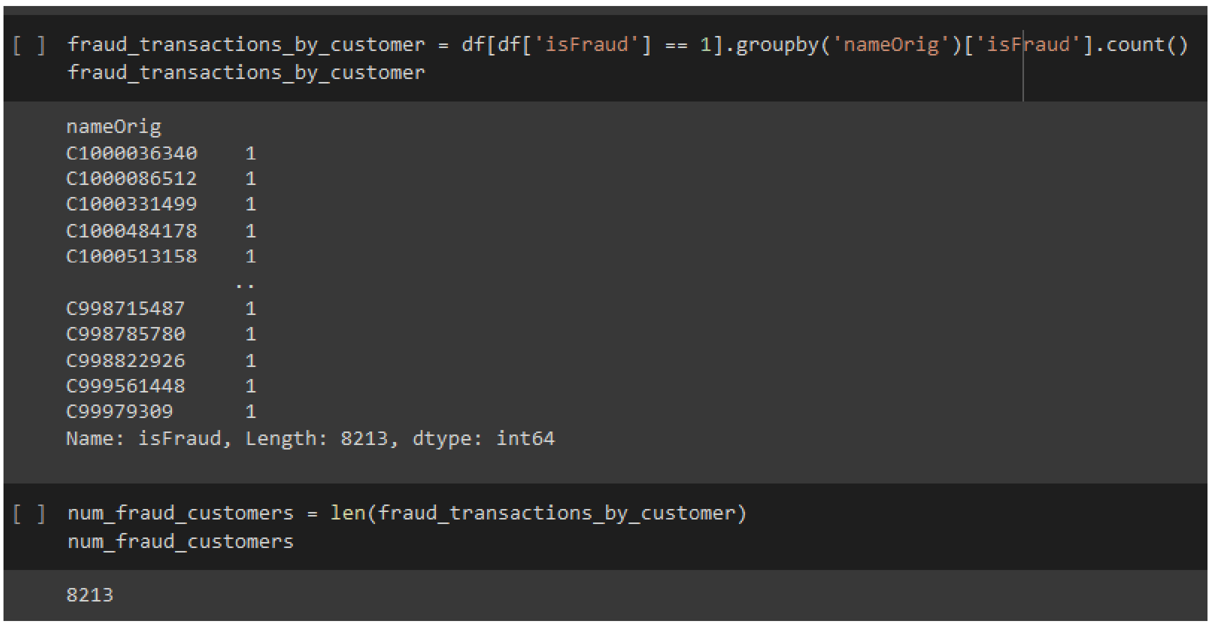
```
[ ] print("Maximum amount transferred in a single transaction:", max_amount_transferred)
```

```
Maximum amount transferred in a single transaction: 92445516.64
```

```
[ ] print("Number of transactions flagged as fraudulent with maximum amount:", num_fraudulent_max)
```

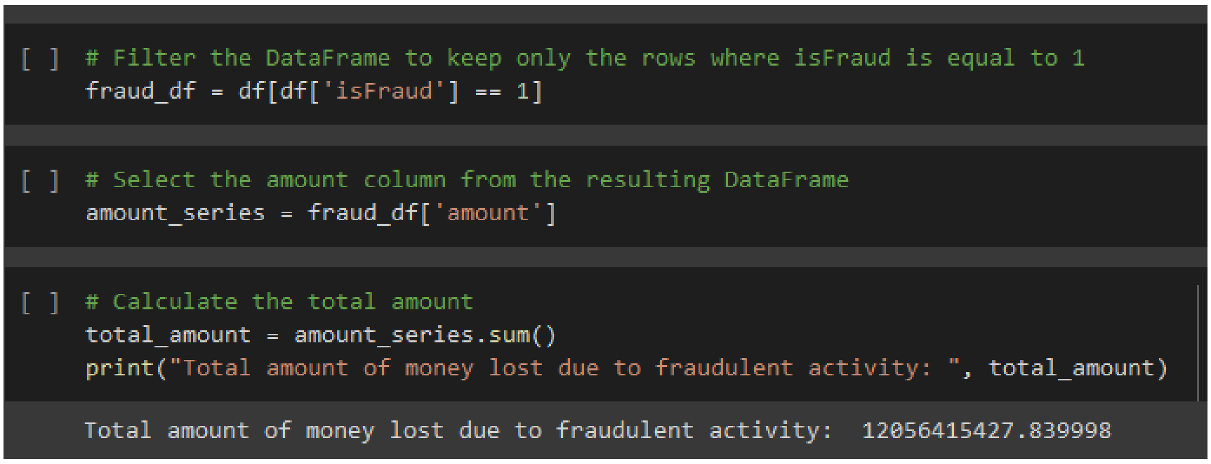
```
Number of transactions flagged as fraudulent with maximum amount: 0
```

3. How many customers initiated transactions that were flagged as fraudulent?



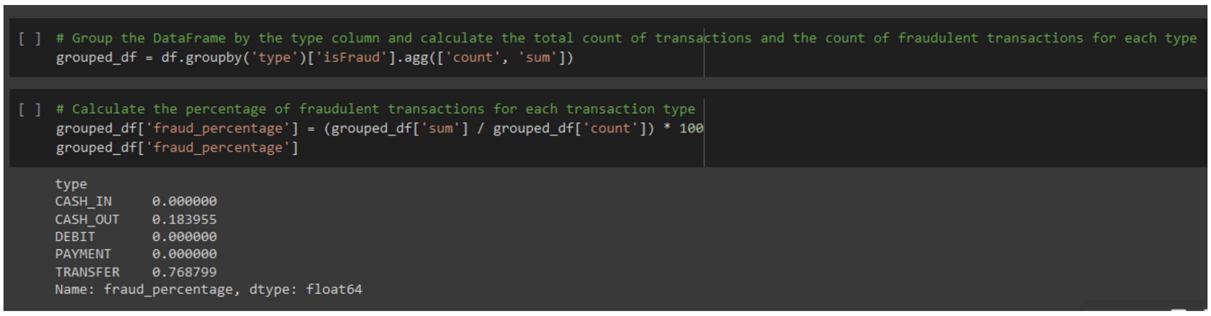
```
[ ] fraud_transactions_by_customer = df[df['isFraud'] == 1].groupby('nameOrig')['isFraud'].count()
    fraud_transactions_by_customer

    nameOrig
    C1000036340    1
    C1000086512    1
    C1000331499    1
    C1000484178    1
    C1000513158    1
                  ..
    C998715487     1
    C998785780     1
    C998822926     1
    C999561448     1
    C99979309      1
    Name: isFraud, Length: 8213, dtype: int64

[ ] num_fraud_customers = len(fraud_transactions_by_customer)
    num_fraud_customers

    8213
```

4. What is the total amount of money lost due to fraudulent activity?



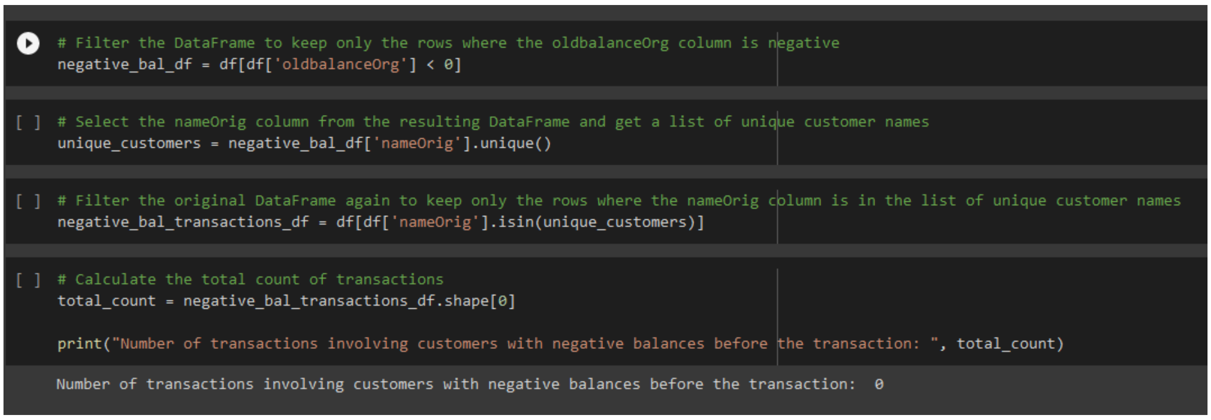
```
[ ] # Filter the DataFrame to keep only the rows where isFraud is equal to 1
    fraud_df = df[df['isFraud'] == 1]

[ ] # Select the amount column from the resulting DataFrame
    amount_series = fraud_df['amount']

[ ] # Calculate the total amount
    total_amount = amount_series.sum()
    print("Total amount of money lost due to fraudulent activity: ", total_amount)

    Total amount of money lost due to fraudulent activity:  12056415427.839998
```

5. Which type of transaction has the highest percentage of fraudulent activity?



```
[ ] # Group the DataFrame by the type column and calculate the total count of transactions and the count of fraudulent transactions for each type
    grouped_df = df.groupby('type')['isFraud'].agg(['count', 'sum'])

[ ] # Calculate the percentage of fraudulent transactions for each transaction type
    grouped_df['fraud_percentage'] = (grouped_df['sum'] / grouped_df['count']) * 100
    grouped_df['fraud_percentage']

    type
    CASH_IN     0.000000
    CASH_OUT    0.183955
    DEBIT       0.000000
    PAYMENT     0.000000
    TRANSFER    0.768799
    Name: fraud_percentage, dtype: float64
```

6. How many transactions involved customers with negative balances before the transaction?



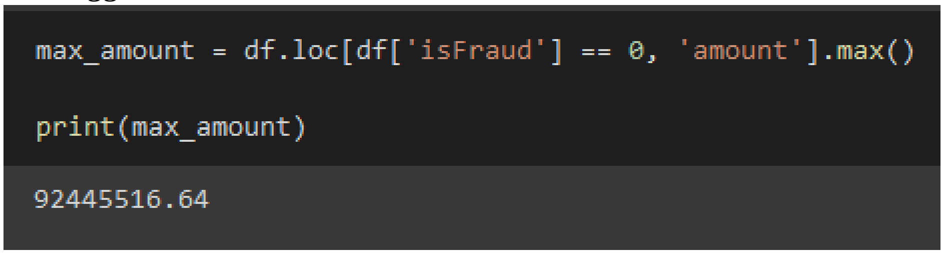
```
# Filter the DataFrame to keep only the rows where the oldbalanceOrg column is negative
negative_bal_df = df[df['oldbalanceOrg'] < 0]
```

```
[ ] # Select the nameOrig column from the resulting DataFrame and get a list of unique customer names
    unique_customers = negative_bal_df['nameOrig'].unique()
```

```
[ ] # Filter the original DataFrame again to keep only the rows where the nameOrig column is in the list of unique customer names
    negative_bal_transactions_df = df[df['nameOrig'].isin(unique_customers)]
```

```
[ ] # Calculate the total count of transactions
    total_count = negative_bal_transactions_df.shape[0]

    print("Number of transactions involving customers with negative balances before the transaction: ", total_count)

    Number of transactions involving customers with negative balances before the transaction:  0
```

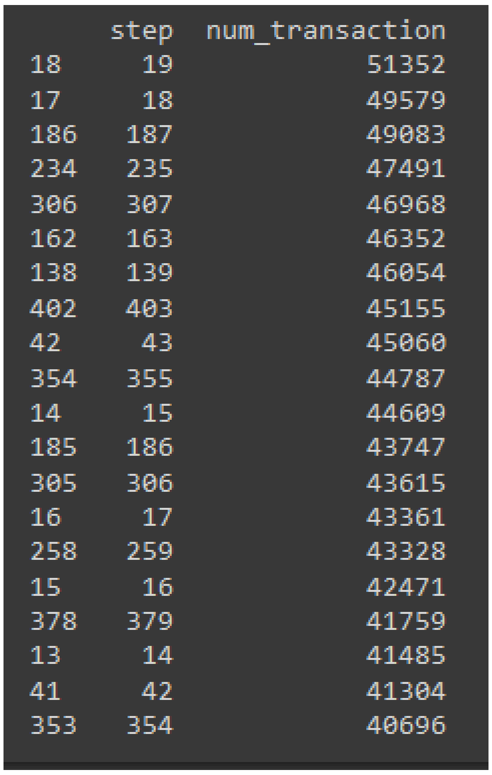
7. What is the maximum amount of money transferred in a single transaction that was not flagged as fraudulent?



```
max_amount = df.loc[df['isFraud'] == 0, 'amount'].max()

print(max_amount)

92445516.64
```

8. How many transactions were processed in each step?

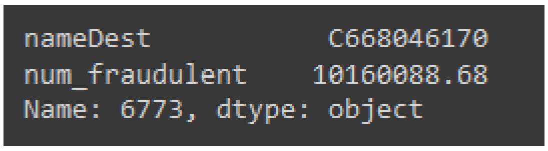
```
step_counts = df.groupby('step')['step'].count().reset_index(name='num_transaction').sort_values(by=['num_transaction'], ascending=False)

print(step_counts.head(20))
```



```
     step  num_transaction
18     19            51352
17     18            49579
186   187            49083
234   235            47491
306   307            46968
162   163            46352
138   139            46054
402   403            45155
42     43            45060
354   355            44787
14     15            44609
185   186            43747
305   306            43615
16     17            43361
258   259            43328
15     16            42471
378   379            41759
13     14            41485
41     42            41304
353   354            40696
```

9. Which name dest is credited with most amount in case of fraud?

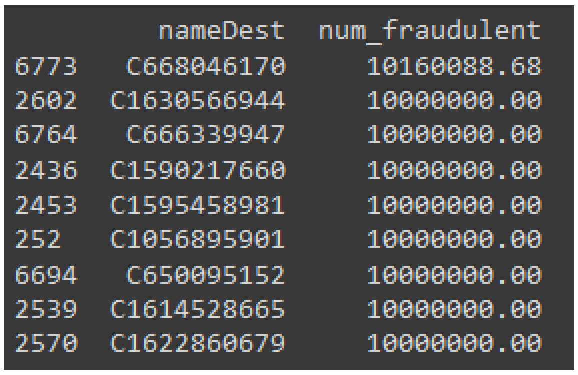
```
fraudulent_destinations = df.loc[df['isFraud'] == 1].groupby('nameDest')['amount'].sum().
reset_index(name='num_fraudulent').sort_values(by=['num_fraudulent'], ascending=False
)

max_fraudulent_destination = fraudulent_destinations.iloc[0]

print(max_fraudulent_destination)
```



```
nameDest            C668046170
num_fraudulent     10160088.68
Name: 6773, dtype: object
```

10. Top 10 destaccount which has maximum amount in case of fraud ?

```
Top_fraudulent_destinations = df.loc[df['isFraud'] == 1].groupby('nameDest')[
'amount'].sum().reset_index(name='num_fraudulent').sort_values(by=['num_fr
audulent'], ascending=False)
Top_10_fraudulent_destinations =Top_fraudulent_destinations.iloc[0:9]
print(Top_10_fraudulent_destinations)
```



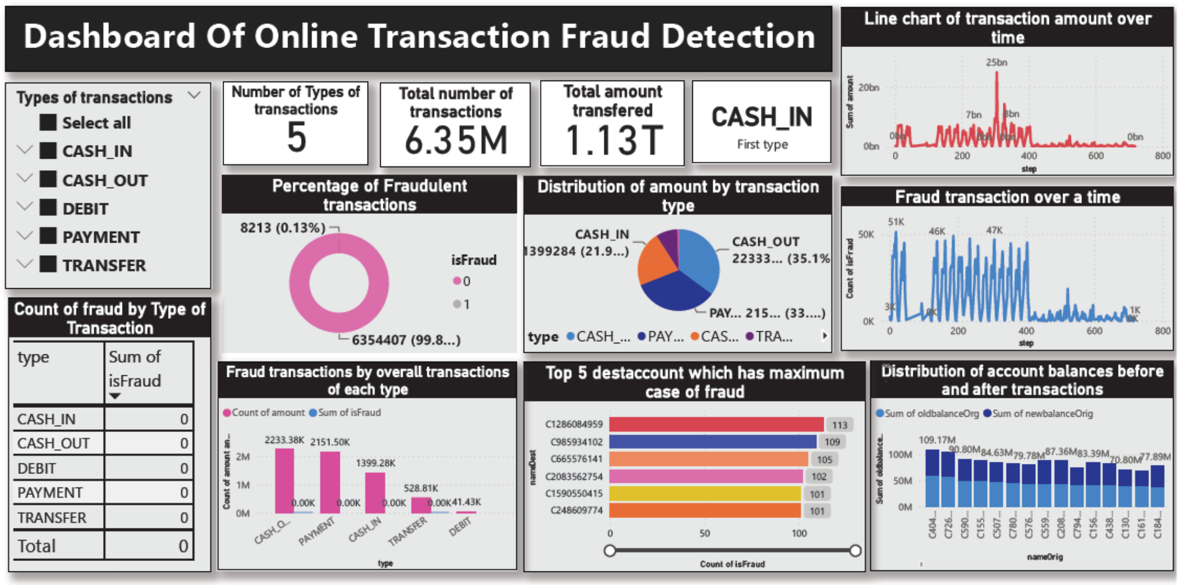
```
        nameDest  num_fraudulent
6773   C668046170     10160088.68
2602  C1630566944     10000000.00
6764   C666339947     10000000.00
2436  C1590217660     10000000.00
2453  C1595458981     10000000.00
252   C1056895901     10000000.00
6694   C650095152     10000000.00
2539  C1614528665     10000000.00
2570  C1622860679     10000000.00
```

Chapter III: DATA VISUALISATION

3.1 Introduction to Power BI

Power BI is a Data Visualization and Business Intelligence tool that converts data from different data sources to interactive dashboards and BI reports. Power BI suite provides multiple software, connector, and services - Power BI desktop, Power BI service based on SaaS, and mobile Power BI apps available for different platforms. These set of services are used by business users to consume data and build BI reports.
Why is visualization important because with the technology revolution data went from expensive, difficult to find and to abundant, cheap data to store, understand and analyse the data. Microsoft Power BI is a data visualization tool that allows you to quickly connect your data, prepare it, and model it as you like. It gives a professional environment allowing you to acquire analytics and reporting capabilities. You can publish these reports, therefore allowing all the users to avail the latest information. It gives you the power to transform all your data into live interactive visuals, create customized real time business view dashboards, thus extracting business intelligence for enhanced decision making

Here we have done the analysis of "Online Transaction Fault Detection" dataset using Power BI application, there are several key insights that we have drawn from the provided dataset:



type	Sum of isFraud
CASH_IN	0
CASH_OUT	0
DEBIT	0
PAYMENT	0
TRANSFER	0
Total	0

1. DONUT CHART

   DESCRIPTION: The donut chart represents the overall fraud rate. The chart is divided into two sections: fraud transactions and non-fraud transactions. The two sections are represented in different colours. The chart provides a visual representation of the proportion of fraudulent transactions in the dataset.

   INSIGHTS & CONCLUSION: By looking at the chart, we can determine that the fraud rate for online transactions in the dataset is relatively low, with only a small percentage(0.13%) of transactions being classified as fraud. This suggests that the majority of online transactions are legitimate.

2. PIE CHART

   DESCRIPTION: The pie chart shows the distribution of transactions based on their transaction type. The chart is divided into segments based on the different transaction types, such as CASH_IN, CASH_OUT, DEBIT, PAYMENT, and TRANSFER. Each segment is labeled with its corresponding transaction type, and the size of the segment corresponds to the proportion of transactions with that type.

   INSIGHTS & CONCLUSION: We can see that the most common transaction types are CASH_OUT(35.17%) and PAYMENT(33.81%), followed by CASH_IN(21.99) and TRANSFER(8.38). Additionally, the chart shows that DEBIT transactions are the least common transaction type in the dataset.

3. Table Chart

   DESCRIPTION: The table chart shows the count of fraud transactions based on their transaction type. The chart is organized in rows by transaction type and in columns by the count of fraudulent transactions for each type. The table provides a detailed breakdown of the number of fraudulent transactions by transaction type.

   INSIGHTS & CONCLUSION: We can see that the highest that is 4116 number of fraudulent transactions occur in CASH_OUT transactions, followed by TRANSFER transactions with count of 4097 fraud transactions. Overall total 8213 transactions were caught as a fraudulent transactions. CASH_IN transactions have the lowest number of fraudulent transactions. The chart provides valuable insights into the areas where fraudulent activity is most prevalent, allowing the project team to take targeted measures to prevent future fraudulent transactions. By monitoring the trends in the fraudulent transaction count, the team can evaluate the effectiveness of their fraud detection system and make necessary improvements.

4. Line Chart:

   DESCRIPTION: The line chart shows the relationship between the step and the isFraud column and the step and the amount column in the dataset. The x-axis of the chart represents the step number which represents time, and the y-axis represents amount column in 1st line chart and isFraud column in 2nd line chart.

   INSIGHTS & CONCLUSION: The line chart provides insights into the relationship between the step and the isFraud column and the step and the amount column. In the first chart (step vs amount) we can see that the amount of transactions is generally higher in the earlier steps and then levels off in the later steps, suggesting that the volume of transactions may be more concentrated during certain periods, In the second chart (step vs isFraud), we can see that there are spikes in the number of fraud transactions in certain steps, indicating that there may be specific periods of time where fraudulent activity is more prevalent.

5. HORIZONTAL BAR CHART

   DESCRIPTION: The horizontal bar chart shows the top 5 recipient accounts (nameDest) that have the highest number of fraudulent transactions. The y-axis of the chart represents the recipient account (nameDest), and the x-axis represents the number of fraudulent transactions (isFraud). The chart displays horizontal bars that represent each recipient account, sorted in descending order by the number of fraudulent transactions.

INSIGHTS & CONCLUSION: The horizontal bar chart provides insights into which recipient accounts are most frequently targeted by fraudsters. By identifying the top 5 recipient accounts that have the highest number of fraudulent transactions, the chart can help the project team to focus its fraud prevention efforts on these accounts. Additionally, the chart can help the team to identify any patterns or trends in the data that may be contributing to the high number of fraudulent transactions.

6. STACKED BAR CHART

DESCRIPTION: The stacked bar chart shows the distribution of account balance before and after transactions, with the name of the account owner (nameOrigin) on the x-axis and the balance amounts on the y-axis. The chart is divided into two sections, with the lower section representing the old balance before the transaction, and the upper section representing the new balance after the transaction.

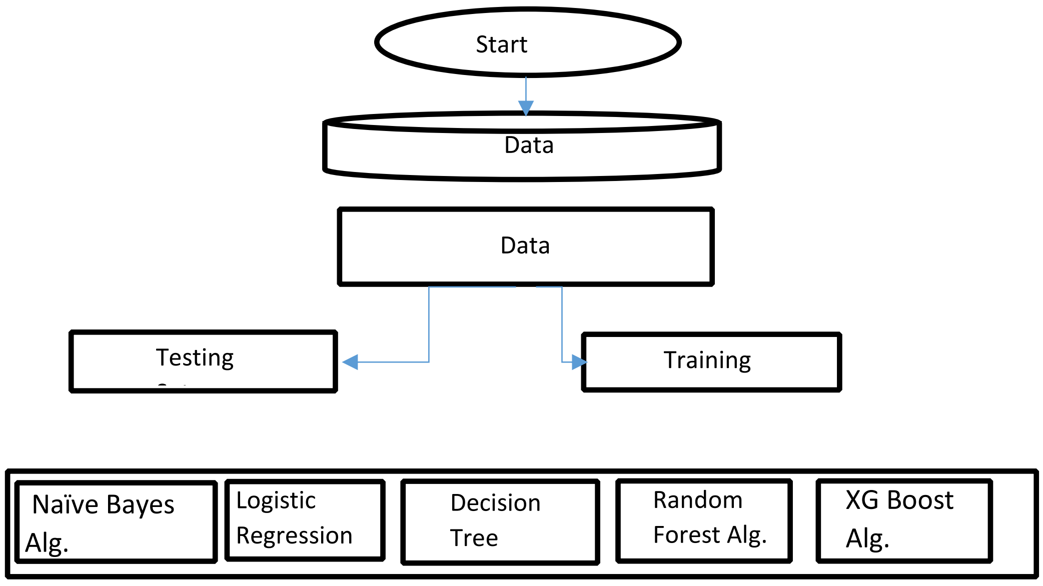
INSIGHTS & CONCLUSION: By comparing the old and new balance sections, we can see the net change in the balance due to the transaction. Overall, the chart can be used to identify trends in the account balance distributions, such as whether there are certain balance ranges that are more commonly associated with transactions or if there are certain account owners who consistently have high or low balances.

Chapter IV: MACHINE LEARNING

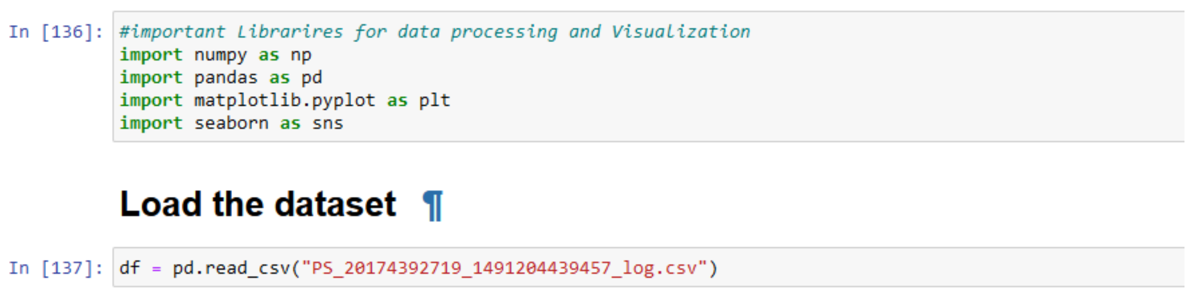
4.1 Introduction to Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.
There Are Different Types of machine Learning algorithm based on dataset we choose the correct model
Models Used in Project:
1) Logistic Regression Algorithm
2) Decision Tree Algorithm
3) Random Forest Algorithm
4)Naïve Bayes Algorithm
5)XGboost Algorithm

Flow Diagram:



Here we are importing the data and importing the libraries in the juypter Notebook



```
In [136]: #important Librarires for data processing and Visualization
          import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
```

Load the dataset ¶

```
In [137]: df = pd.read_csv("PS_20174392719_1491204439457_log.csv")
```

4.2 EDA and Data Pre-processing:

4.2.1 Introduction to EDA

What is EDA?
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.
Need of EDA Before Machine Learning?
Helps you prepare your dataset for analysis. Allows a machine learning model to predict our dataset better. Gives you more accurate results. It also helps us to choose a better machine learning model.
For EDA we use Python Libraries:-
1. Pandas
2. NumPy
3. Matplotlib
4. Seaborn

Before EDA we need to check following parameter:
1. NULL values in Dataset
2. Duplicate Values
3. Multicollinearity

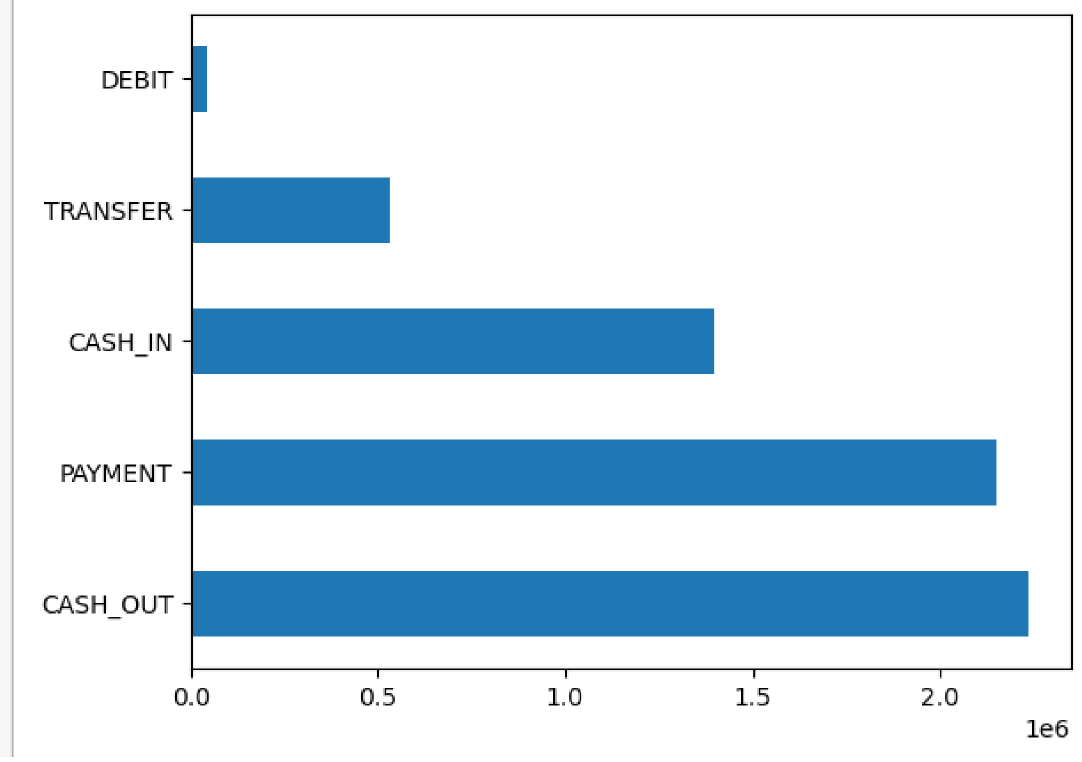
4.2.2 - Operation Perform on EDA
We Drop the Columns which is not Much important as per analysis
- Transaction unique identifier : Not co-related with the price column
- PPD Category Type : It is similar to duration column.
- Record Status : It contains monthly file only Indicates additions, changes and deletions to the records so not necessary.

EDA operations perform for Analysis the dataset:-

Univariate Analysis

Univariate analysis is used to analyse the data of single variable. Here we will analyse using different plots.

1. **We can see that the type of transactions done and the count of transactions**

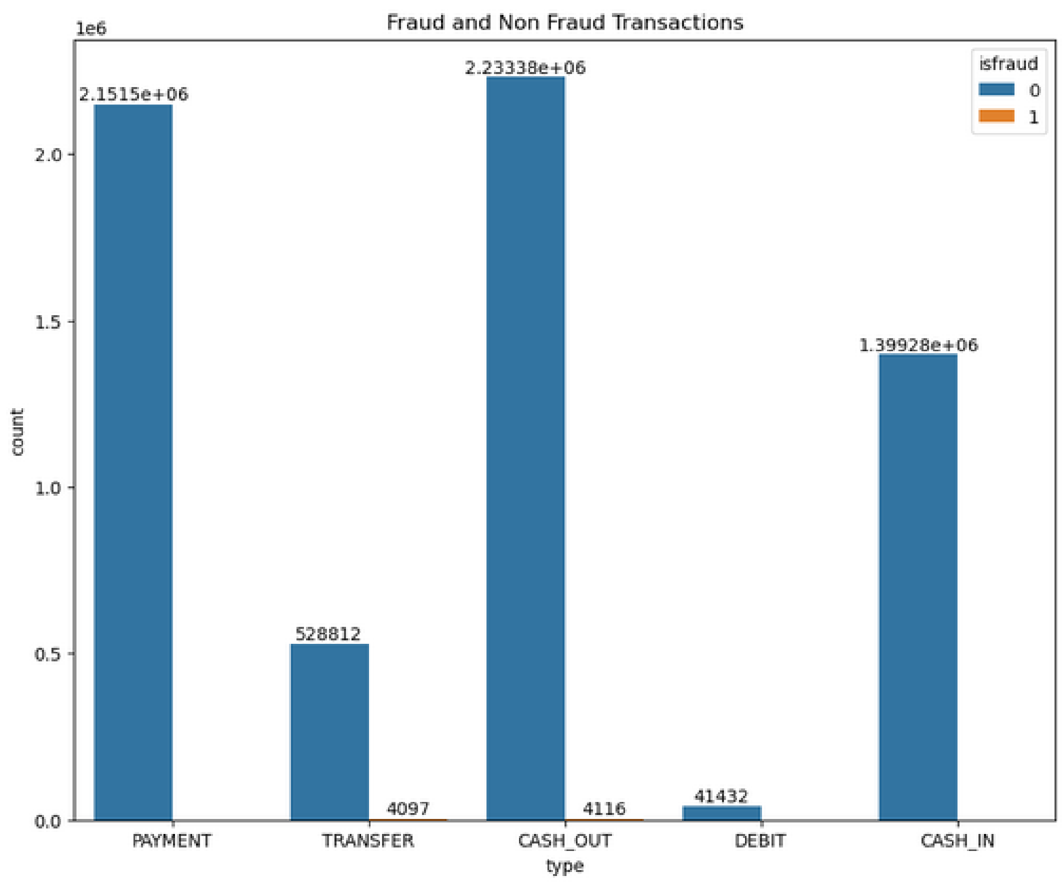


As we have already seen there are 5 types of payment happened.

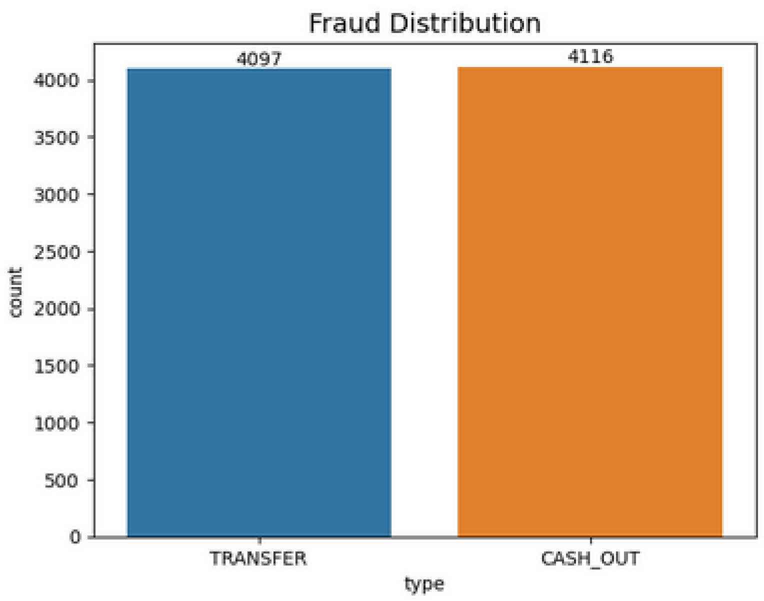
More money has been transacted mostly through cash out followed by payment type.

CASH_OUT	**2237500**
PAYMENT	**2151495**
CASH_IN	**1399284**
TRANSFER	**532909**
DEBIT	**41432**

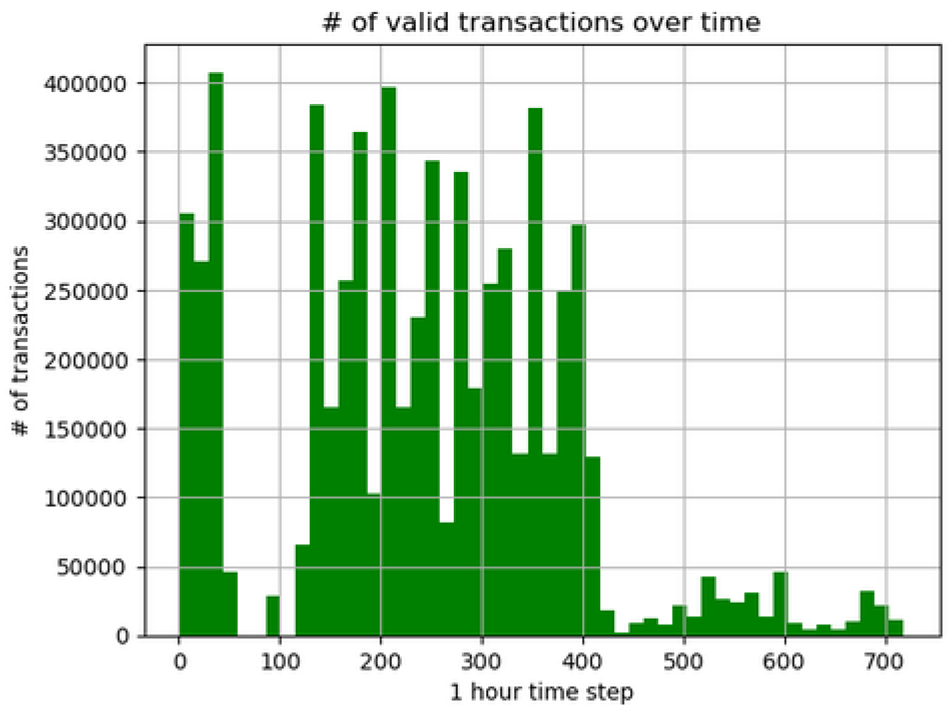
2. **Segregation of the type of transactions on the basis of fraud done in any particular type of transaction:**



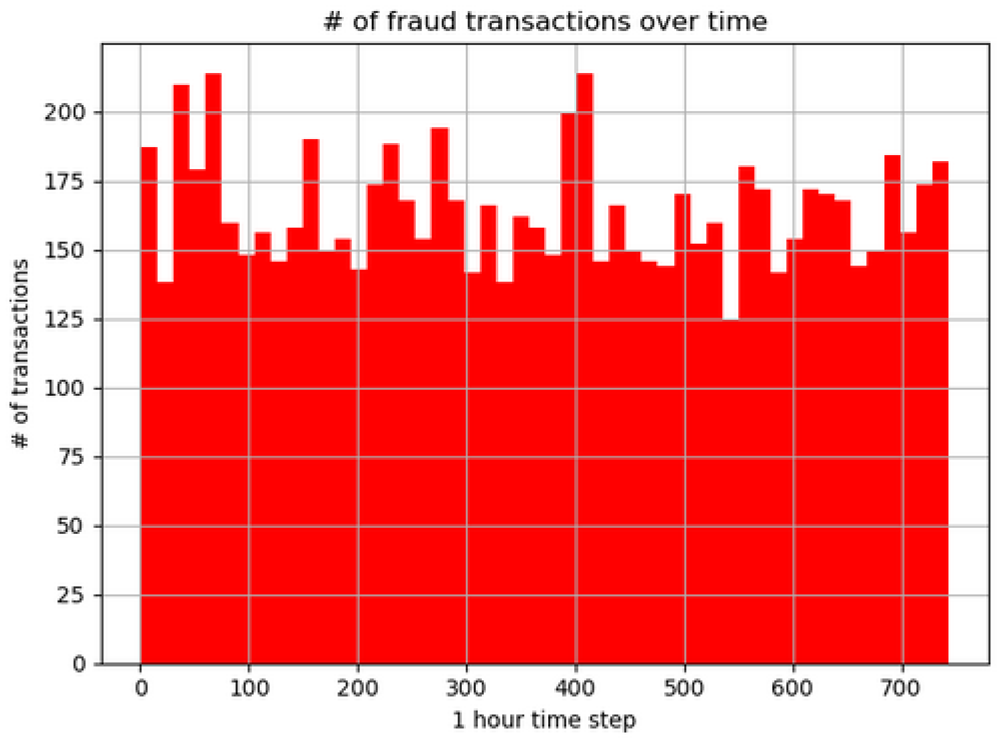
Now we can say that two type has fault transaction which is flagged so now we can see the distribution of the type of transaction:



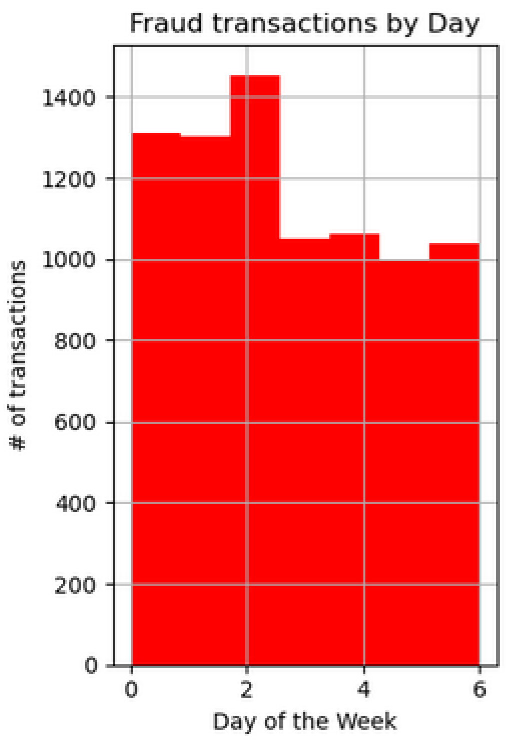
Since we have Step as time component we can say that

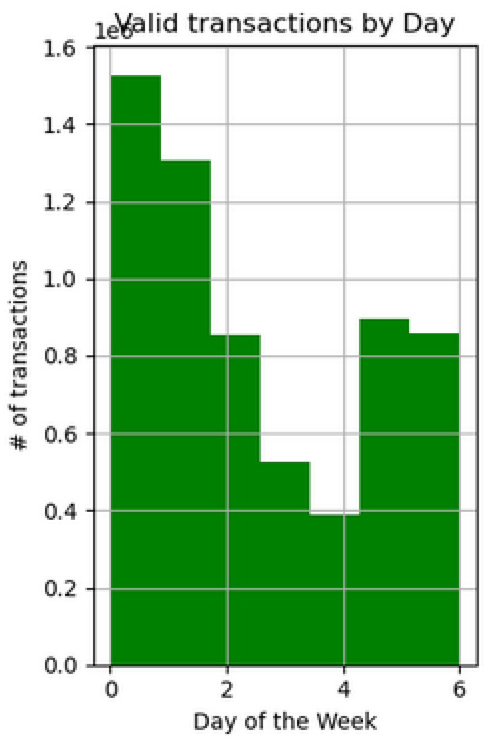


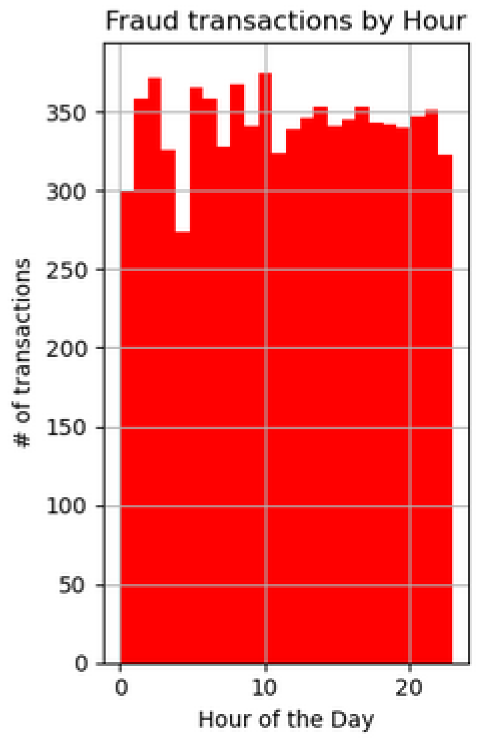
# of valid transactions over time
# of transactions
1 hour time step
0
50000
100000
150000
200000
250000
300000
350000
400000
0
100
200
300
400
500
600
700

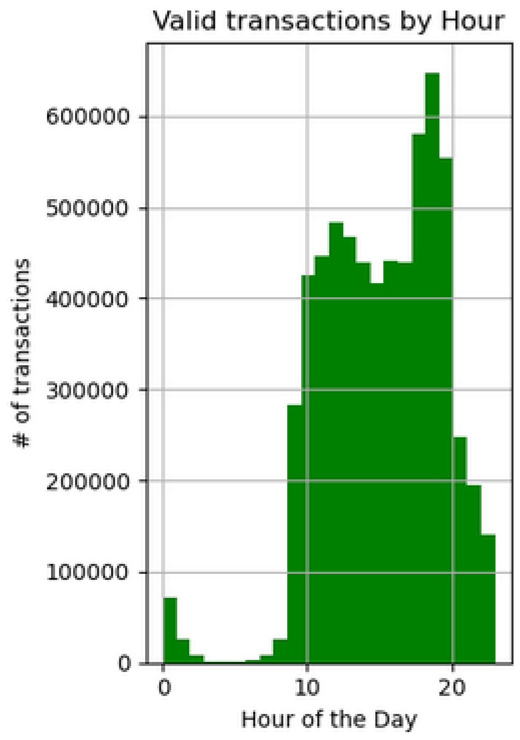


# of fraud transactions over time
# of transactions
1 hour time step
0
25
50
75
100
125
150
175
200
0
100
200
300
400
500
600
700



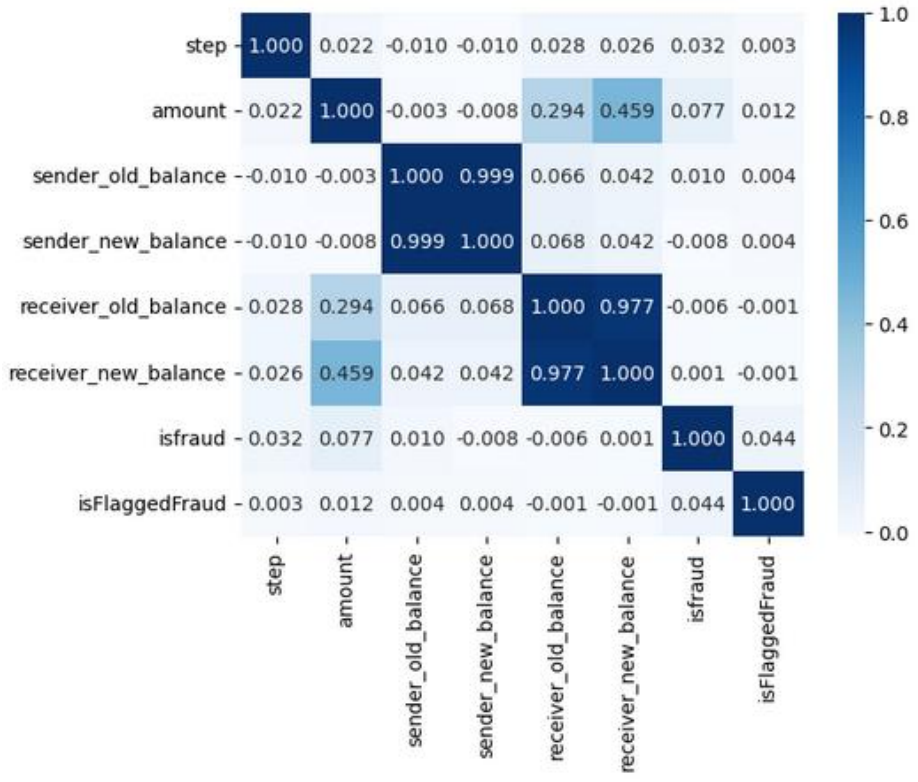






Bivariate Analysis

Bivariate analysis is used to analyse the data of Two or more variable. Here we will analyse using Heat Map.



4.3 Preparing for the Machine Learning Model

4.3.1 Splitting the data into test train using Sklearn

Python-Library to For the Machine learning is sklearn. Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python.
Split the data set into two pieces — a training set and a testing set. This consists of random sampling without replacement about 80 percent of the rows (you can vary this) and putting them into your training set. The remaining 20 percent is put into your test set. (“X_train,” “X_test,” “y_train,” “y_test”) for a particular train test split.

4.4.2 Rescale variables via standardization

For this we need to import StandardScaler from sklearn.preprocessing.
Python sklearn library offers us with StandardScaler() function to **standardize the data values into a standard format.**StandardScaler removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way.

4.4.3 Logistic Regression Model

 Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
 Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or

False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

Mathematically we represent a logistic regression equation:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Where

y=Dependent Variable

x,x_1,………. and x_n=Independent variable

b_0,………..and bn= coefficient of the dependent variables.

So in our case we are using the logistic regression on y which is IsFraud column

And x is all other variables are termed as independent variables used in model.

4.4.3.1 Testing and Prediction of model

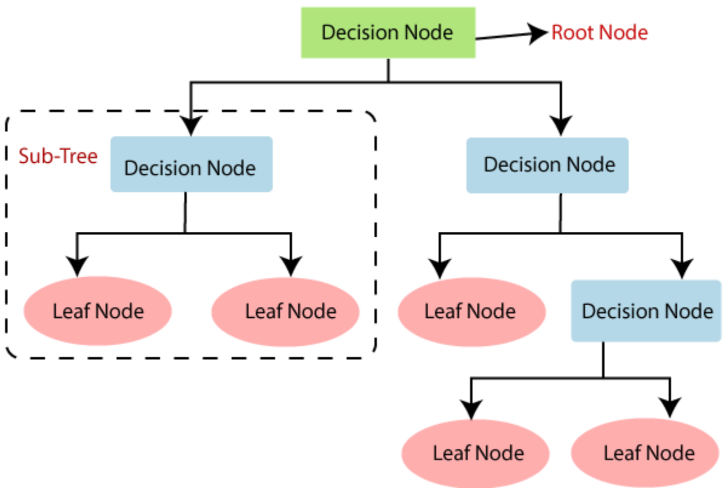
Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.
The Accuracy of the model is Near By 92% Which is high enough but checked our data set with other another algorithm to check which has more or less accuracy compared to this model.

4.4.4 Applying Decision Tree Model

• Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

• In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

• In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and based on the comparison, follows the branch and jumps to the next node.



4.4.4.1 Testing and Prediction of model

Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.
The Accuracy of the model is Near By 99% Which is high enough but checked our data set with other another algorithm to check which has more or less accuracy compared to this model.

4.4.5 Naïve Bayes Algorithm

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.
- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the sprobability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

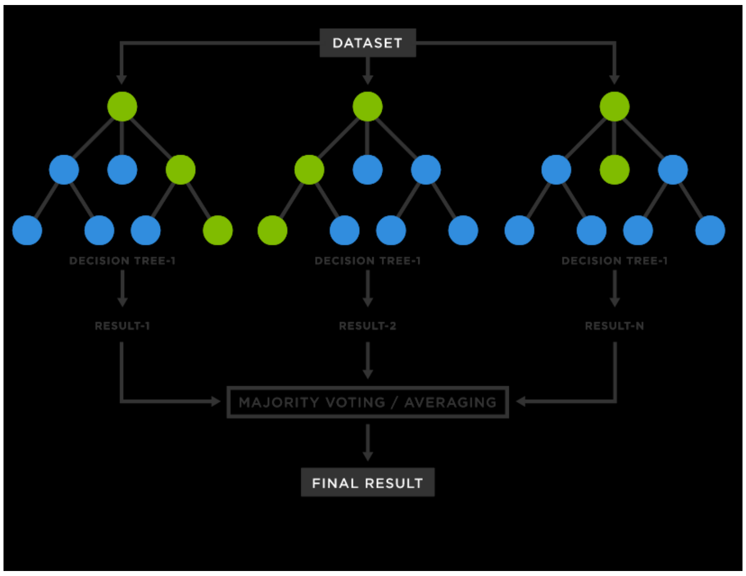
4.4.5.1 Testing and Prediction of model

Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.
The Accuracy of the model is Near By 92% which is high enough but checked our data set with other another algorithm to check which has more or less accuracy compared to this model.

4.4.6 Random forest Algorithm

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample.



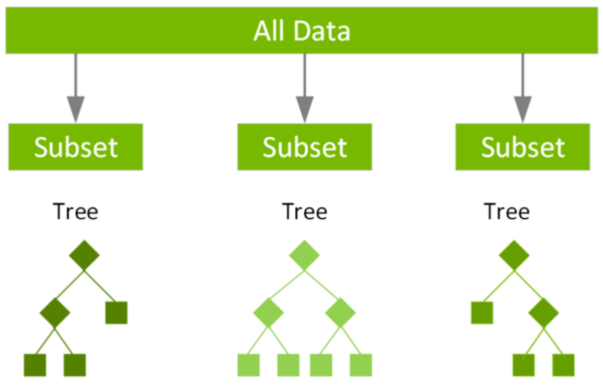
Similar steps are to be followed as above for data preparation.

4.4.6.1 Testing and Prediction of model

Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.
The Accuracy of the model is Near By 99% which is high enough but checked our data set with other another algorithm to check which has more or less accuracy compared to this model.

4.4.7 XG boost

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. A Gradient Boosting Decision Trees (GBDT) is a decision tree ensemble learning algorithm similar to random forest, for classification and regression. Ensemble learning algorithms combine multiple machine learning algorithms to obtain a better model.



The term "gradient boosting" comes from the idea of "boosting" or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model

4.4.7.1 Testing and Prediction of model

Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.
The Accuracy of the model is Near By 99% which is high enough but checked our data set with other another algorithm to check which has more or less accuracy compared to this model.

4.5 Conclusion from machine learning:

Accuracy and Prediction:
After applying Random Forest Algorithm we got the nearby 99% accuracy. Which is high enough.
Because applying model on 6.3 million is big task we have make sure that approximation of each model is almost nearly eqaul. And every model was trained on the same.

Chapter V: CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

From this project we did hands on with Pyspark. We did data analysis using the PySpark and pandas. In Pyspark we performed some business question using Spark SQL and by creating RDD.
For visualisation we used Power BI Desktop and gathered many insights from the data easily with the help of dashboards we created.
After that we performed EDA using Jupyter notebook . Then we used 6.3 Million records for the prediction purpose. For prediction we used
1) Logistic Regression Algorithm
2) Decision Tree Algorithm
3) Random Forest Algorithm
4) Naïve Bayes Algorithm
5) XGboost Algorithm

5.2 Future Scope

With the advancement of technology, the methods used by fraudsters are also becoming more sophisticated, making it necessary to implement new strategies to prevent online fraud transactions. One can implement AI and ML can be used to analyse patterns and detect anomalies in transactions, making it easier to identify potential fraudsters and prevent fraudulent transactions.
Educating people on cybersecurity risks and how to avoid them can help prevent online fraud transactions. Collaboration and partnerships between businesses, financial institutions, and government agencies can help prevent online fraud transactions by sharing information and resources.