**1. Data cleaning including missing values, outliers and multi-collinearity.

```
import pandas as pd
```

```
df=pd.read_csv("/content/Fraud.csv")
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFr
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	

6362620 rows × 11 columns

```
df.shape
```

```
(6362620, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

```
df.isnull()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFr
0	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	
6362615	False	False	False	False	False	False	False	False	False	False	F
6362616	False	False	False	False	False	False	False	False	False	False	F
6362617	False	False	False	False	False	False	False	False	False	False	F
6362618	False	False	False	False	False	False	False	False	False	False	F
6362619	False	False	False	False	False	False	False	False	False	False	F

6362620 rows × 11 columns

```
x=df[["amount","oldbalanceOrg","newbalanceOrig","oldbalanceDest","newbalanceDest"]].values
x
```

```
array([[9.83964000e+03, 1.70136000e+05, 1.60296360e+05, 0.00000000e+00,
        0.00000000e+00],
       [1.86428000e+03, 2.12490000e+04, 1.93847200e+04, 0.00000000e+00,
        0.00000000e+00],
       [1.81000000e+02, 1.81000000e+02, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00],
       ...,
       [6.31140928e+06, 6.31140928e+06, 0.00000000e+00, 6.84888400e+04,
        6.37989811e+06],
       [8.50002520e+05, 8.50002520e+05, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00],
       [8.50002520e+05, 8.50002520e+05, 0.00000000e+00, 6.51009911e+06,
        7.36010163e+06]])
```

```
import numpy as np
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(x)
x=imputer.transform(x)
x
```

```
array([[9.83964000e+03, 1.70136000e+05, 1.60296360e+05, 0.00000000e+00,
        0.00000000e+00],
       [1.86428000e+03, 2.12490000e+04, 1.93847200e+04, 0.00000000e+00,
        0.00000000e+00],
       [1.81000000e+02, 1.81000000e+02, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00],
       ...,
       [6.31140928e+06, 6.31140928e+06, 0.00000000e+00, 6.84888400e+04,
        6.37989811e+06],
       [8.50002520e+05, 8.50002520e+05, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00],
       [8.50002520e+05, 8.50002520e+05, 0.00000000e+00, 6.51009911e+06,
        7.36010163e+06]])
```

```
y=df[["isFraud"]].values
y
```

```
array([[0],
       [0],
       [1],
       ...,
       [1],
       [1],
       [1]])
```

```
import numpy as np
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(y)
y=imputer.transform(y)
y
```

```
array([[0.],
       [0.],
       [1.],
       ...,
```

```
       [1.],
       [1.],
       [1.]])
```

```
df.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03	2.514687e-06
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02	1.585775e-03
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00	0.000000e+00
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00	0.000000e+00
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00	1.000000e+00

```
df.max()
```

	0
step	743
type	TRANSFER
amount	92445516.64
nameOrig	C999999784
oldbalanceOrg	59585040.37
newbalanceOrig	49585040.37
nameDest	M999999784
oldbalanceDest	356015889.35
newbalanceDest	356179278.92
isFraud	1
isFlaggedFraud	1

dtype: object

```
df.min()
```

	0
step	1
type	CASH_IN
amount	0.0
nameOrig	C1000000639
oldbalanceOrg	0.0
newbalanceOrig	0.0
nameDest	C1000004082
oldbalanceDest	0.0
newbalanceDest	0.0
isFraud	0
isFlaggedFraud	0

dtype: object

```
import numpy as np
import pandas as pd

# Example DataFrame
data = df["amount"]
df = pd.DataFrame(data)

# Z-Score Calculation
from scipy.stats import zscore
```

```
df['Z_Score'] = zscore(df["amount"])

# Removing Outliers
df_without_outliers = df[abs(df['Z_Score']) <= 3
print(df_without_outliers)
```

```
              amount   Z_Score
0            9839.64 -0.281560
1            1864.28 -0.294767
2             181.00 -0.297555
3             181.00 -0.297555
4           11668.14 -0.278532
...              ...       ...
6362613  1258818.82  1.786772
6362614   339682.13  0.264665
6362615   339682.13  0.264665
6362618   850002.52  1.109765
6362619   850002.52  1.109765

[6317675 rows x 2 columns]
```

```
df.max()
```

	0
amount	9.244552e+07
Z_Score	1.527936e+02

dtype: float64

```
pd.isnull(df)
pd.isnull(df).sum()
```

	0
amount	0
Z_Score	0

dtype: int64

FOR REMOVE OUTLIERS FORM THE DATASETS 🐞

```
import pandas as pd

# Load the CSV file
# Replace 'data.csv' with the path to your CSV file
# df = pd.read_csv('data.csv')

# Choose the column you want to check for outliers
# Replace 'column_name' with the name of your column
column = 'amount'

# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df[column].quantile(0.25)
Q3 = df[column].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers
df_without_outliers = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Save the cleaned dataset to a new CSV file
df_without_outliers.to_csv('cleaned_data.csv', index=False)

print("Outliers removed and cleaned data saved to 'cleaned_data.csv'")
```

```
Outliers removed and cleaned data saved to 'cleaned_data.csv'
```