Importing all the Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix
import warnings
import pickle
```

Reading the CSV

```
df = pd.read_csv('/content/onlinefraud.csv')
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDes
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M19797871
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M20442822
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C5532640
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C389970
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M12307017

```
df.columns
```

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

```
df.drop(['isFlaggedFraud'],axis = 1, inplace = True)
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C5
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M12
...	...	...	...	...	...	...	
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C4
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M6
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M13
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M19
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M6

1048575 rows × 10 columns

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDes
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M19797871
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M20442822
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C5532640
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C389970
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M12307017

```
df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C4
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M6
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M6

```
plt.style.use('ggplot')
warnings.filterwarnings('ignore')
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1
...	...	...	...	...	...	...	
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C4
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M6
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M6

1048575 rows × 10 columns

```
len(df)
```

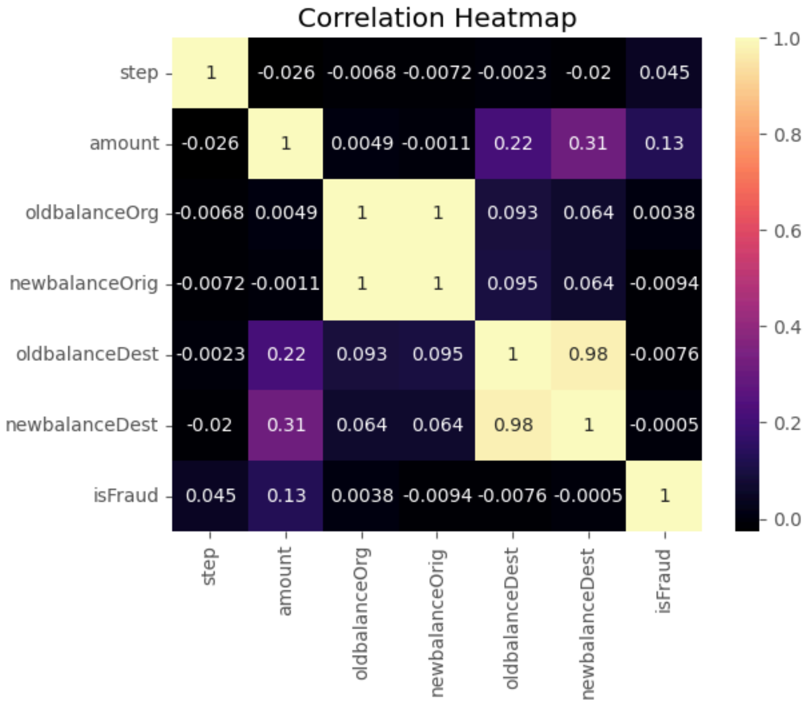
```
1048575
```

```
numeric_df = df.select_dtypes(include='number')

# Compute correlation
correlation = numeric_df.corr()

# Print correlation matrix
print(correlation)
```

```
                    step    amount  oldbalanceOrg  newbalanceOrig  \
step            1.000000 -0.025996      -0.006780       -0.007180   
amount         -0.025996  1.000000       0.004864       -0.001133   
oldbalanceOrg  -0.006780  0.004864       1.000000        0.999047   
newbalanceOrig -0.007180 -0.001133       0.999047        1.000000   
oldbalanceDest -0.002251  0.215558       0.093305        0.095182   
newbalanceDest -0.019503  0.311936       0.064049        0.063725   
isFraud         0.045030  0.128862       0.003829       -0.009438   

                oldbalanceDest  newbalanceDest   isFraud  
step                 -0.002251       -0.019503  0.045030  
amount                0.215558        0.311936  0.128862  
oldbalanceOrg         0.093305        0.064049  0.003829  
newbalanceOrig        0.095182        0.063725 -0.009438  
oldbalanceDest        1.000000        0.978403 -0.007552  
newbalanceDest        0.978403        1.000000 -0.000495  
isFraud              -0.007552       -0.000495  1.000000  
```

Heatmap

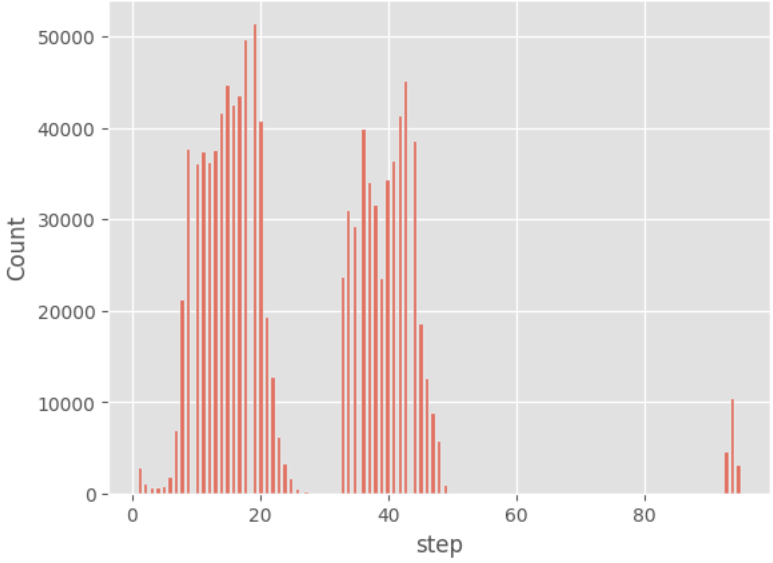
```
sns.heatmap(correlation, annot=True, cmap='magma')
# Add a title
plt.title('Correlation Heatmap')

# Display the heatmap
plt.show()
```



Univariate Analysis

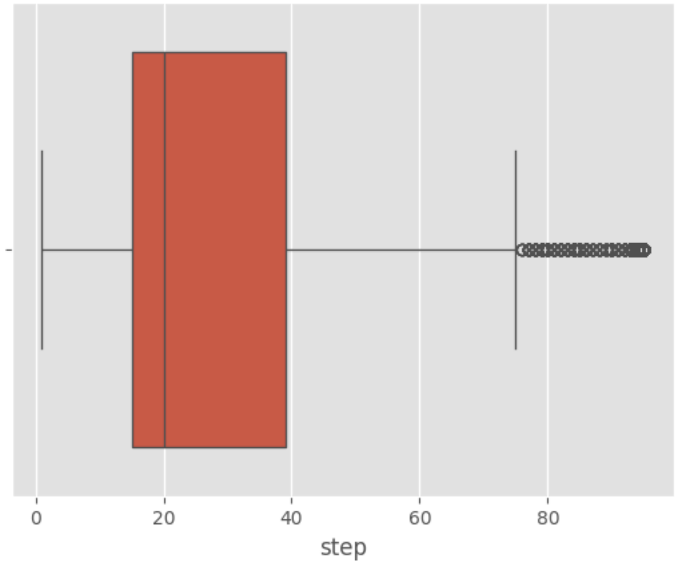
```
sns.histplot(data=df, x='step')
```

```
<Axes: xlabel='step', ylabel='Count'>
```



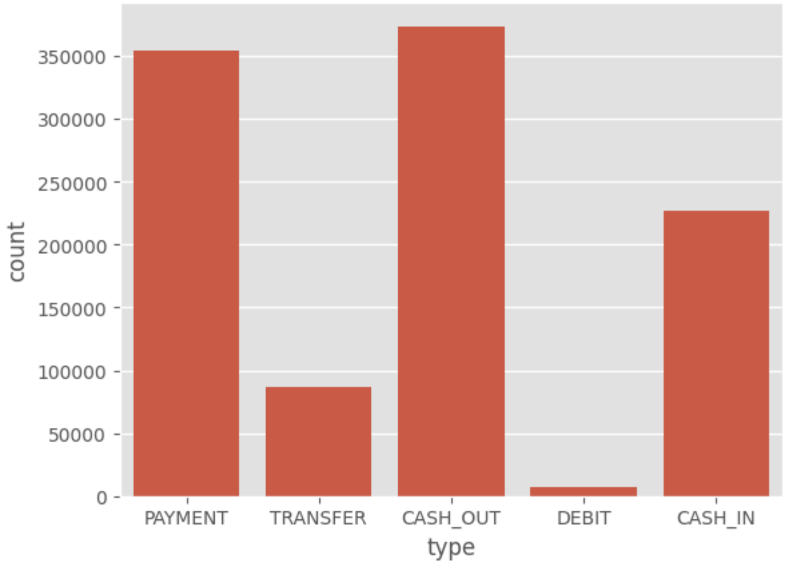
```
sns.boxplot(data=df,x='step')
```

```
<Axes: xlabel='step'>
```



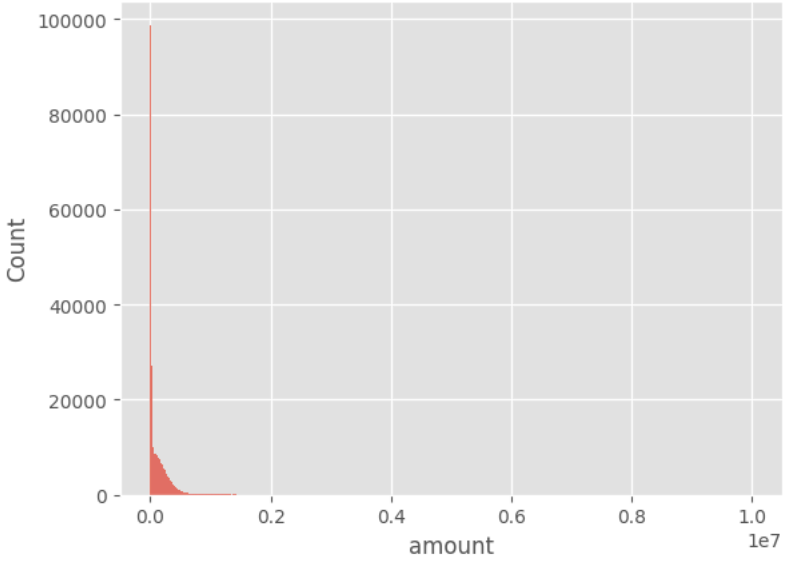
```
sns.countplot(data=df, x='type')
```

```
<Axes: xlabel='type', ylabel='count'>
```



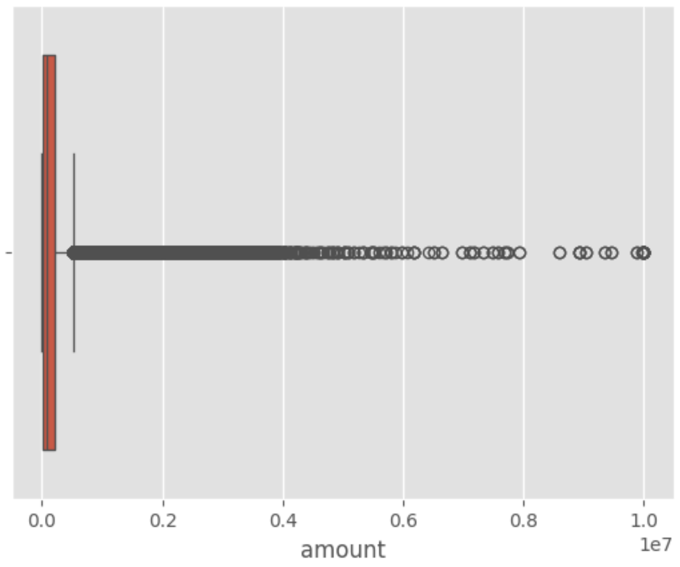
```
sns.histplot(data=df, x='amount')
```

```
<Axes: xlabel='amount', ylabel='Count'>
```



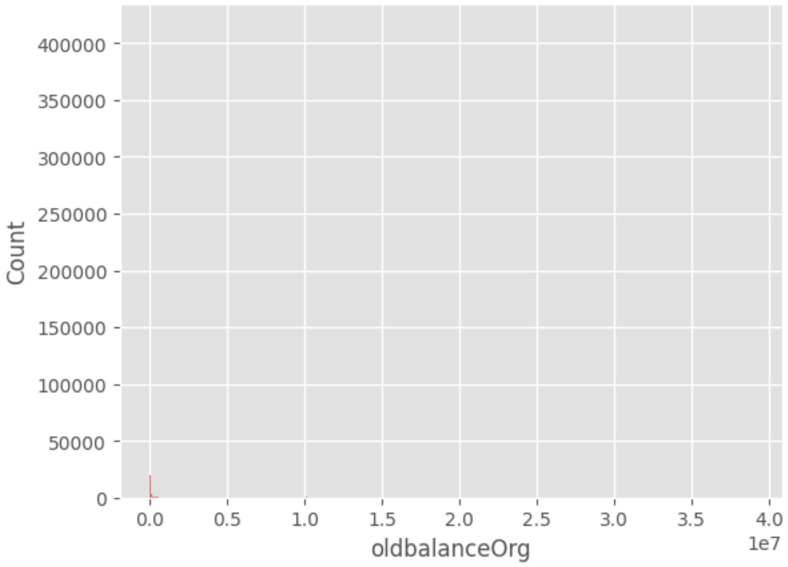
```
sns.boxplot(data=df, x='amount')
```

```
<Axes: xlabel='amount'>
```



```
sns.histplot(data=df, x='oldbalanceOrg')
```

```
<Axes: xlabel='oldbalanceOrg', ylabel='Count'>
```

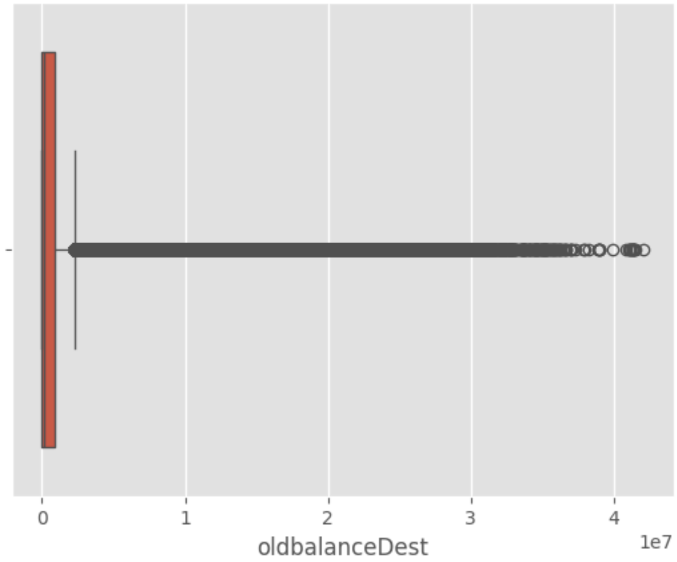


```
df['nameDest'].value_counts()
```

```
nameDest
C985934102     98
C1286084959    96
C1590550415    89
C248609774     88
C665576141     87
               ..
M382871047      1
M322765556      1
M1118794441     1
M1127250627     1
M677577406      1
Name: count, Length: 449635, dtype: int64
```

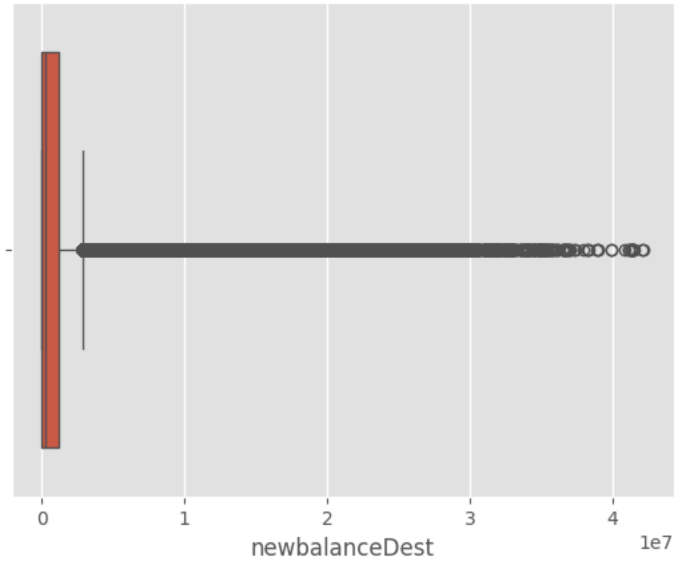
```
sns.boxplot(data=df, x='oldbalanceDest')
```

```
<Axes: xlabel='oldbalanceDest'>
```



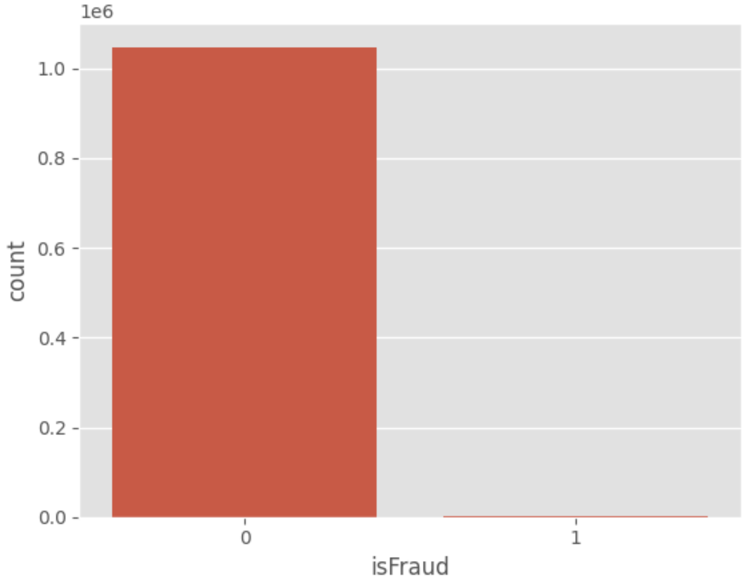
```
sns.boxplot(data=df, x='newbalanceDest')
```

```
<Axes: xlabel='newbalanceDest'>
```



```
sns.countplot(data=df, x='isFraud')
```

```
<Axes: xlabel='isFraud', ylabel='count'>
```



```
df['isFraud'].value_counts()
```

```
isFraud
0    1047433
1       1142
Name: count, dtype: int64
```

```
df.loc[df['isFraud']==0, 'isFraud'] = 'is not Fraud'
df.loc[df['isFraud']==1, 'isFraud'] = 'is Fraud'
```

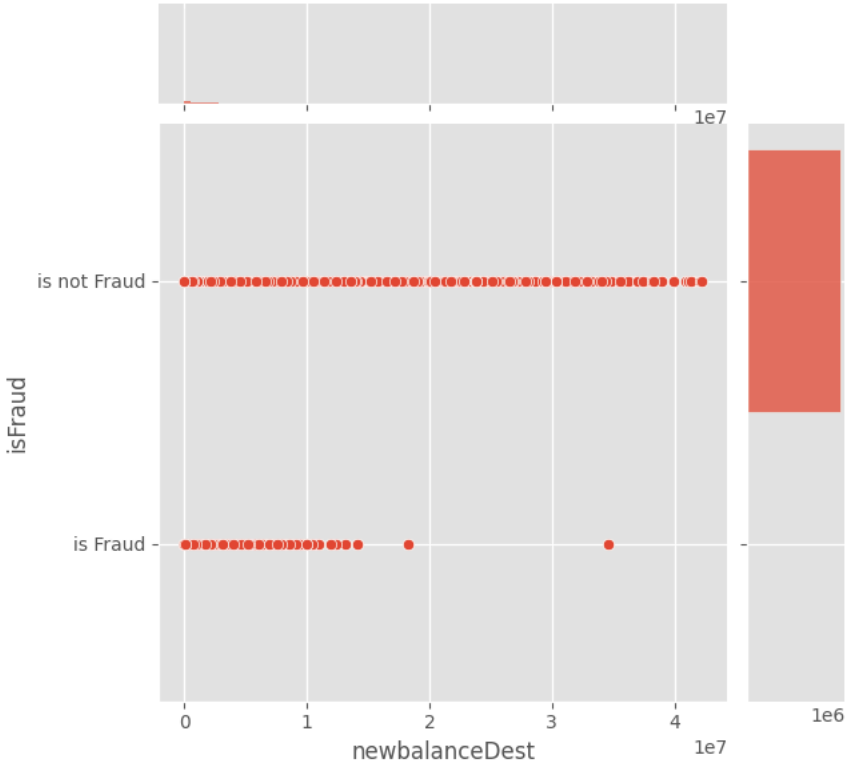
```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72
2	1	TRANSFER	181.00	C1305486145	181.00	0.00
3	1	CASH_OUT	181.00	C840083671	181.00	0.00
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86
...	...	...	...	...	...	...
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64

Bi Variate Analysis

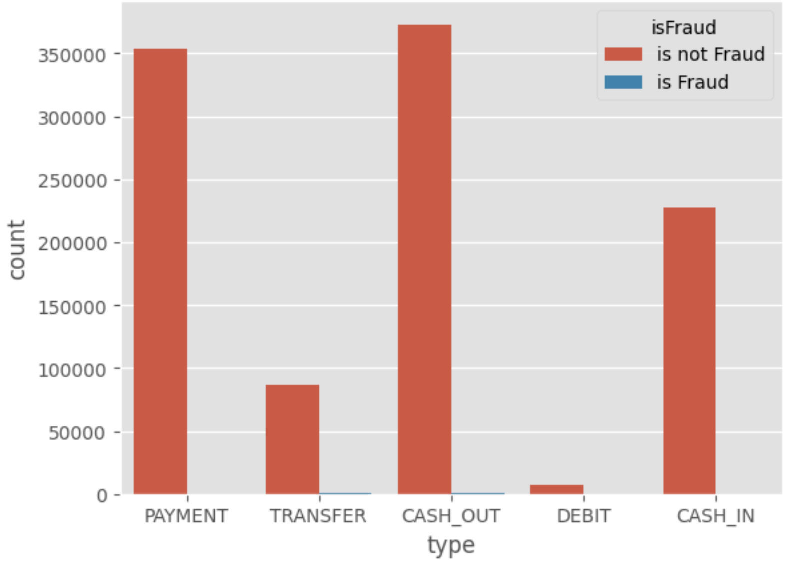
```
sns.jointplot(data=df,x='newbalanceDest',y='isFraud')
```

```
<seaborn.axisgrid.JointGrid at 0x7ec48f50b310>
```



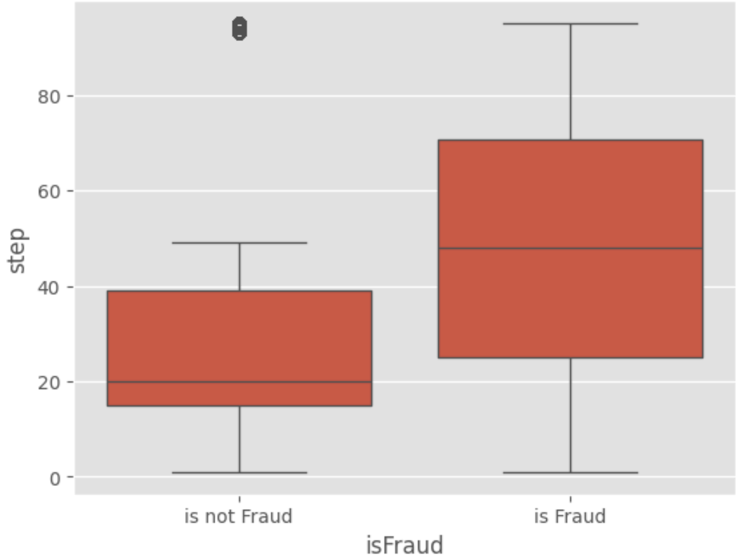
```
sns.countplot(data=df,x='type',hue='isFraud')
```

```
<Axes: xlabel='type', ylabel='count'>
```



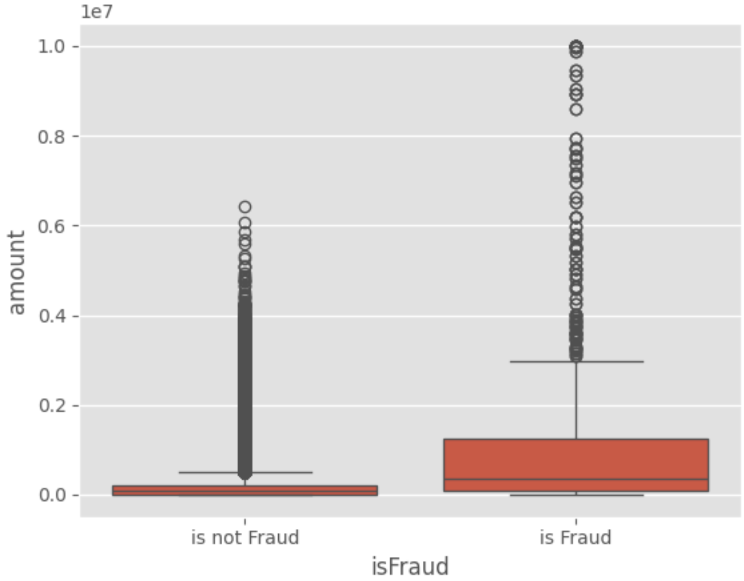
```
sns.boxplot(data=df, x='isFraud', y='step')
```

```
<Axes: xlabel='isFraud', ylabel='step'>
```



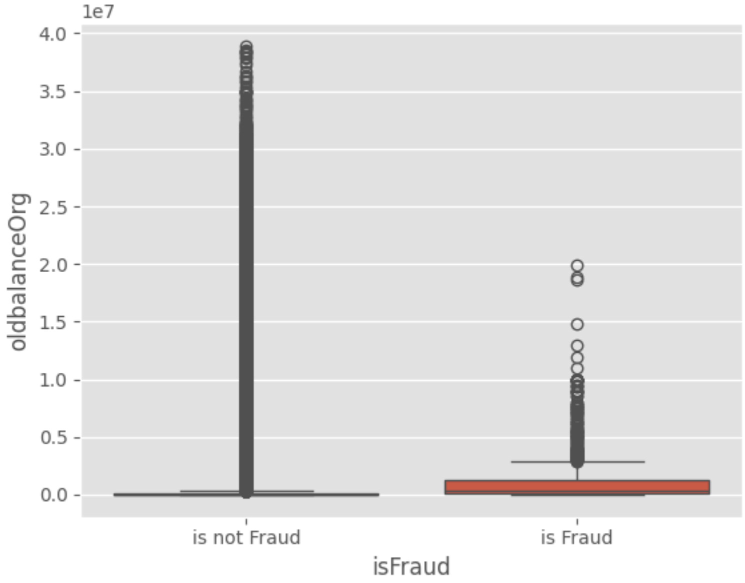
```
sns.boxplot(data=df, x='isFraud', y='amount')
```

```
<Axes: xlabel='isFraud', ylabel='amount'>
```



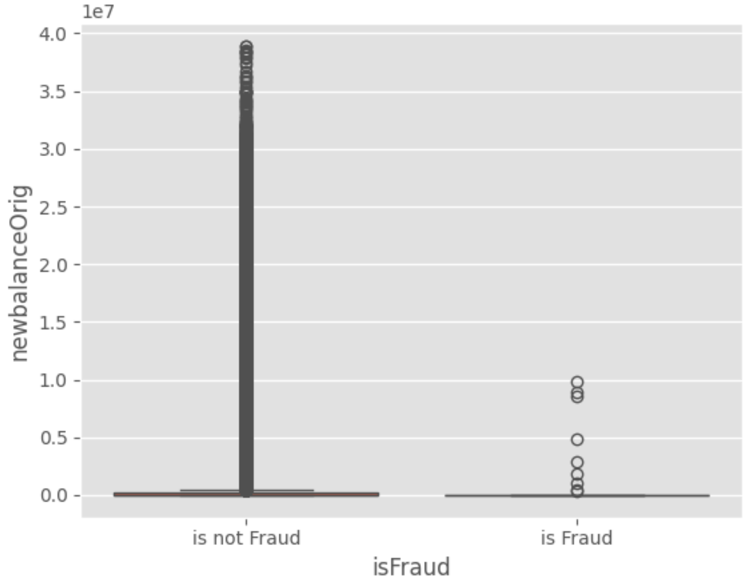
```
sns.boxplot(data=df, x='isFraud', y='oldbalanceOrg')
```

```
<Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>
```



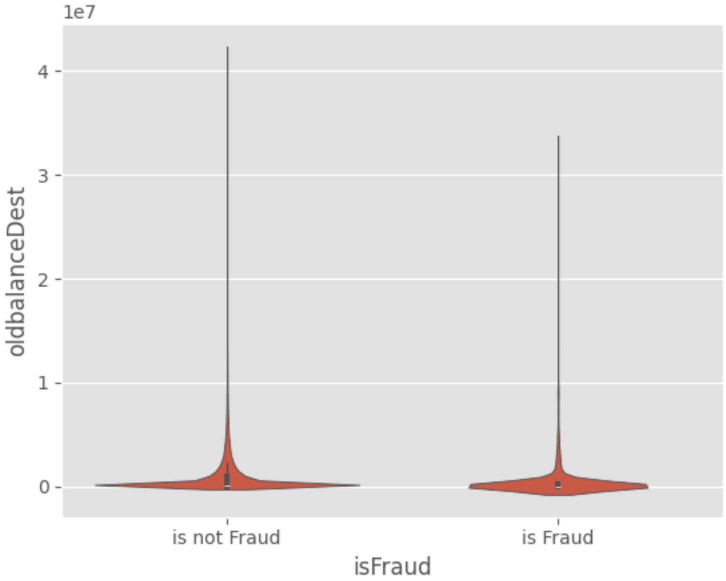
```
sns.boxplot(data=df, x='isFraud', y='newbalanceOrig')
```

```
<Axes: xlabel='isFraud', ylabel='newbalanceOrig'>
```



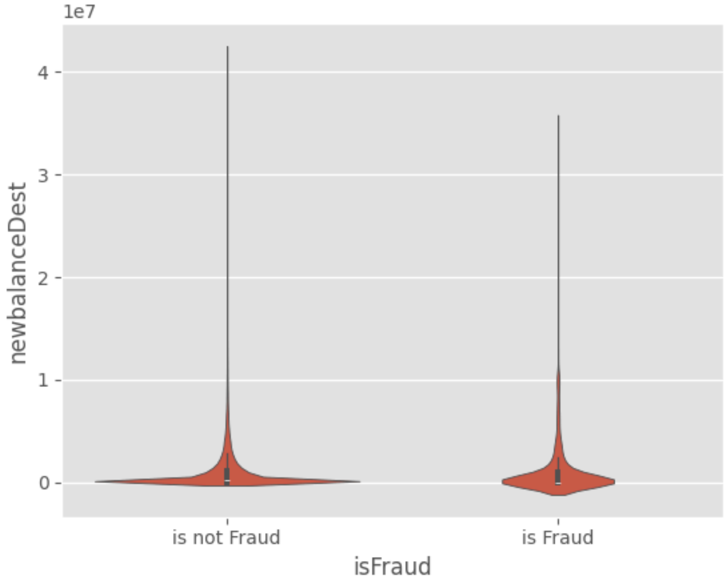
```
sns.violinplot(data=df,x='isFraud',y='oldbalanceDest')
```

```
<Axes: xlabel='isFraud', ylabel='oldbalanceDest'>
```



```
sns.violinplot(data=df,x='isFraud',y='newbalanceDest')
```

```
<Axes: xlabel='isFraud', ylabel='newbalanceDest'>
```



Descriptive Analysis

```
df.describe(include='all')
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanc
count	1.048575e+06	1048575	1.048575e+06	1048575	1.048575e+06	1.04857
unique	NaN	5	NaN	1048317	NaN	
top	NaN	CASH_OUT	NaN	C1214450722	NaN	
freq	NaN	373641	NaN	2	NaN	
mean	2.696617e+01	NaN	1.586670e+05	NaN	8.740095e+05	8.93808
std	1.562325e+01	NaN	2.649409e+05	NaN	2.971751e+06	3.00827
min	1.000000e+00	NaN	1.000000e-01	NaN	0.000000e+00	0.00000
25%	1.500000e+01	NaN	1.214907e+04	NaN	0.000000e+00	0.00000
50%	2.000000e+01	NaN	7.634333e+04	NaN	1.600200e+04	0.00000
75%	3.900000e+01	NaN	2.137619e+05	NaN	1.366420e+05	1.74600
max	9.500000e+01	NaN	1.000000e+07	NaN	3.890000e+07	3.89000

DATA PRE-PROCESSING

Checking for Null values

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

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

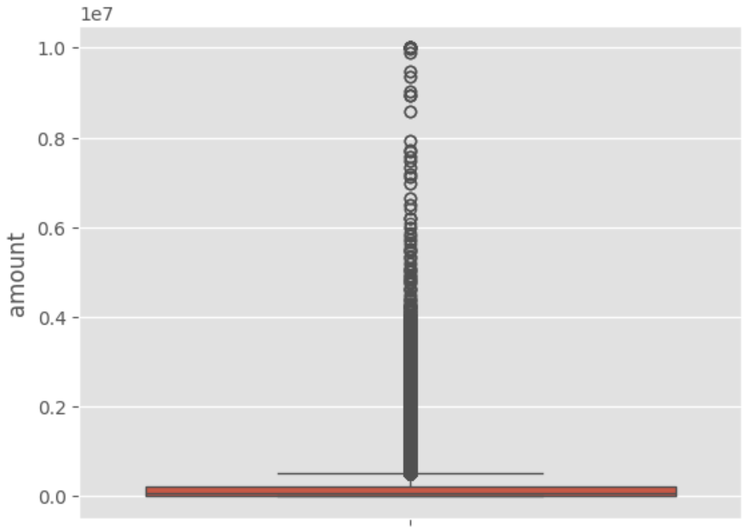
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object 
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object 
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  object 
dtypes: float64(5), int64(1), object(4)
memory usage: 80.0+ MB
```

Handling Outliers

```
sns.boxplot(df['amount'])
```

```
<Axes: ylabel='amount'>
```



Remove the Outliers

```
from scipy import stats
print(stats.mode(df['amount']))
print(np.mean(df['amount']))
```

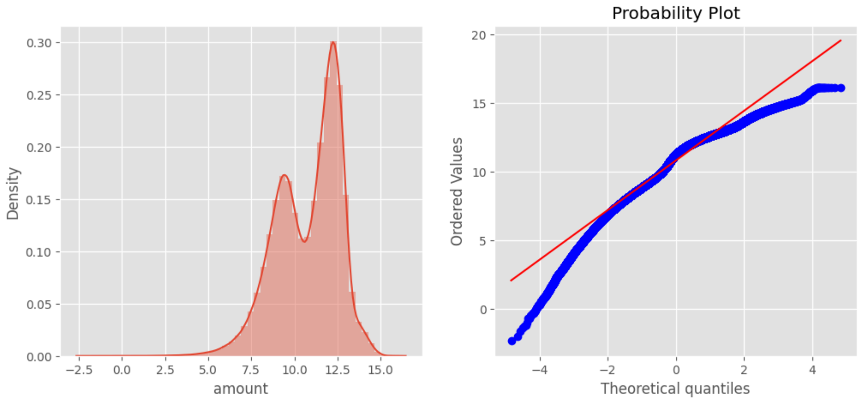
```
ModeResult(mode=10000000.0, count=14)
158666.9755271392
```

```
q1 = np.quantile(df['amount'],0.25)
q3 = np.quantile(df['amount'],0.75)

IQR = q3-q1

upper_bound = q3+(1.5*IQR)
lower_bound = q1-(1.5*IQR)

print('q1 :',q1)
print('q3 :',q3)
print('IQR :',IQR)
print('Upper Bound :', upper_bound)
print('Lower Bound :', lower_bound)
print('Skewed data :',len(df[df['amount']>upper_bound]))
print('Skewed data :',len(df[df['amount']<lower_bound]))
```

```
q1 : 12149.065
q3 : 213761.89
IQR : 201612.825
Upper Bound : 516181.12750000006
Lower Bound : -290270.17250000004
Skewed data : 53088
Skewed data : 0
```

```
def transformationPlot(feature):
  plt.figure(figsize=(12,5))
  plt.subplot(1,2,1)
  sns.distplot(feature)
  plt.subplot(1,2,2)
  stats.probplot(feature,plot=plt)
```

```
transformationPlot(np.log(df['amount']))
```



```
df['amount']=np.log(df['amount'])
```

Object Data LabelEncoding

```
from sklearn.preprocessing import LabelEncoder

la = LabelEncoder()
df['type'] = la.fit_transform(df['type'])
```

```
df['type'].value_counts()
```

```
type
1    373641
3    353873
0    227130
```

```
4     86753
2      7178
Name: count, dtype: int64
```

```
x = df.drop('isFraud',axis = 1)
y = df['isFraud']
```

```
x
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDe
0	1	3	9.194174	C1231006815	170136.00	160296.36	M1979787
1	1	3	7.530630	C1666544295	21249.00	19384.72	M2044282
2	1	4	5.198497	C1305486145	181.00	0.00	C553264
3	1	1	5.198497	C840083671	181.00	0.00	C38997
4	1	3	9.364617	C2048537720	41554.00	29885.86	M1230701
...	...	...	...	...	...	...	
1048570	95	1	11.794771	C1179511630	479803.00	347245.65	C435674
1048571	95	3	9.202042	C1956161225	90545.00	80627.64	M668364
1048572	95	3	9.556766	C2037964975	20545.00	6404.95	M135518
1048573	95	3	9.212343	C1633237354	90605.00	80584.95	M196499
1048574	95	3	9.345748	C1264356443	80584.95	69134.92	M677577

1048575 rows × 9 columns