Transaction Integrity Inspection System

Importing packages

```
import numpy as np
import pandas as pd
```

Loading the dataset

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
data=pd.read_csv('/content/drive/MyDrive/Final year project /Transaction_dataset1.csv')
```

Retrieving the First 5 and last 5 values

```
data.head()
```

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

```
data.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C435674507	484329.37	616886.72	0
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M668364942	0.00	0.00	0
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1355182933	0.00	0.00	0
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1964992463	0.00	0.00	0
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M677577406	0.00	0.00	0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 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  int64  
 10  isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

```
data.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1048575.0
mean	2.696617e+01	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06	1.089097e-03	0.0
std	1.562325e+01	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06	3.298351e-02	0.0
min	1.000000e+00	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0
25%	1.500000e+01	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0
50%	2.000000e+01	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05	0.000000e+00	0.0
75%	3.900000e+01	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06	0.000000e+00	0.0
max	9.500000e+01	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07	1.000000e+00	0.0

```
print(data.isnull().sum())
```

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

```
data.keys()
```

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

```
data.shape
```

```
(1048575, 11)
```

```
print(data.duplicated().sum())
```

```
0
```

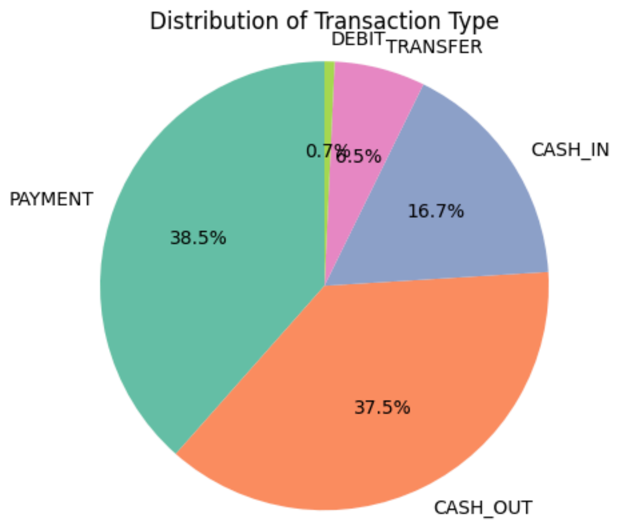
```
data['type'].value_counts()
```

type	count
CASH_OUT	373641
PAYMENT	353873
CASH_IN	227130
TRANSFER	86753
DEBIT	7178

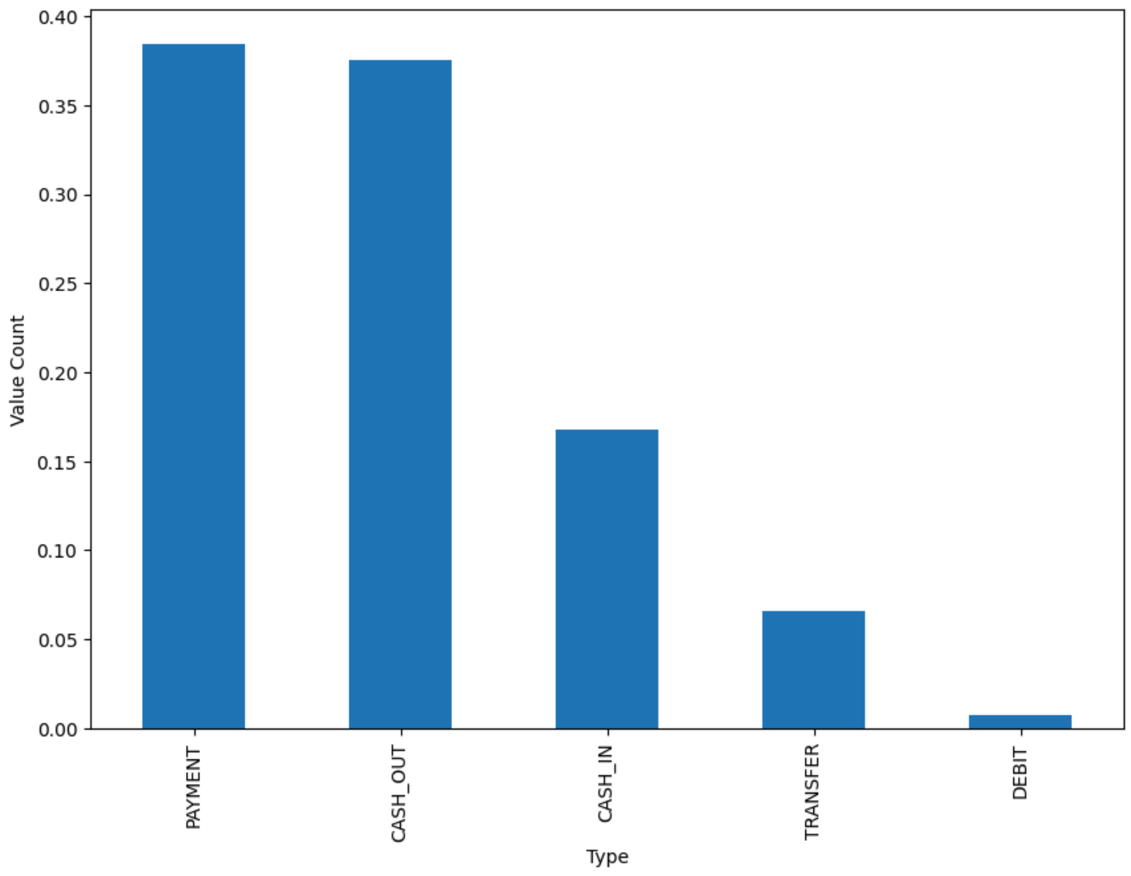
dtype: int64

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm


type = data['type'].value_counts()

# Get the transaction types and their counts
transactions = type.index
quantity = type.values

# Define a list of darker colors
colors = cm.Set2.colors[:len(transactions)]
```

```
# Create the pie chart
fig, ax = plt.subplots()
ax.pie(quantity, labels=transactions, autopct='%1.1f%%', startangle=90, colors=colors)
ax.axis('equal')  # Ensures the pie chart is a perfect circle
ax.set_title('Distribution of Transaction Type')

plt.show()
```

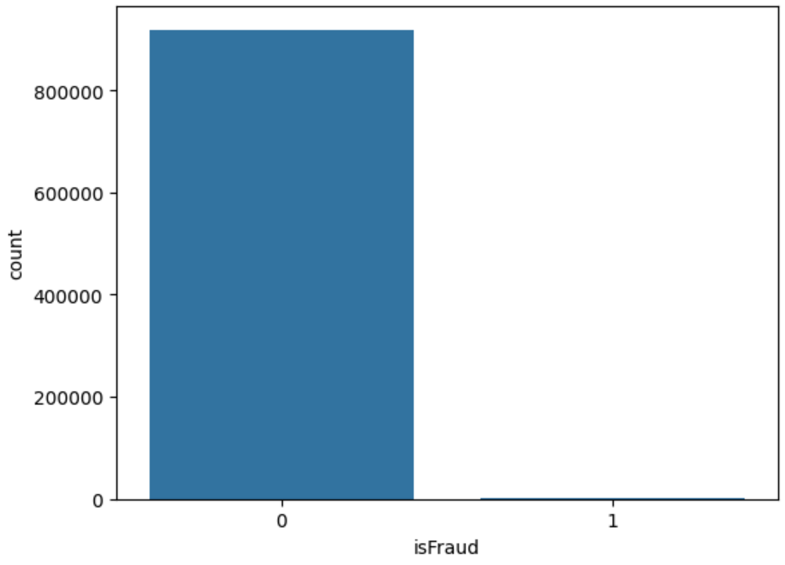


```
fig = plt.figure(figsize=(10, 7))
data['type'].value_counts(normalize=True).plot(kind='bar')
plt.xlabel("Type")
plt.ylabel("Value Count")
plt.show()
```



```
sns.countplot(data=data,x="isFraud")
```

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



Removing unwanted columns

```
data = data.drop('step', axis=1)
```

```
data = data.drop('isFlaggedFraud', axis=1)
```

```
print(data.keys())
```

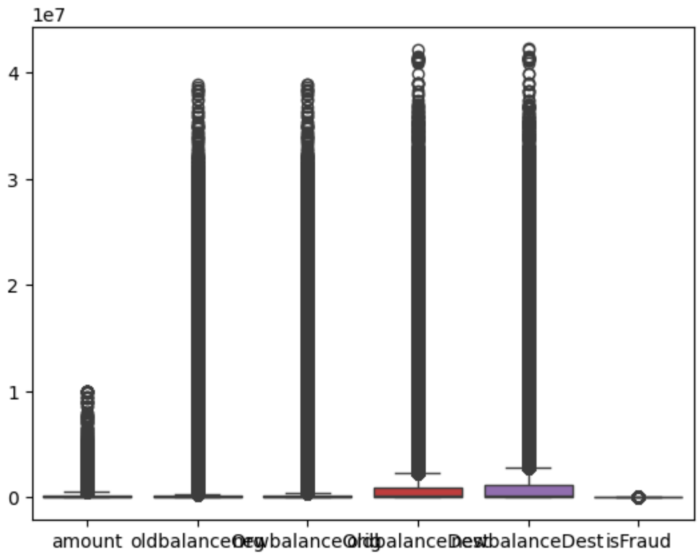
```
Index(['type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud'],
      dtype='object')
```

Removal of outliers

```
import seaborn as sns
```

```
sns.boxplot(data)
```

```
<Axes: >
```



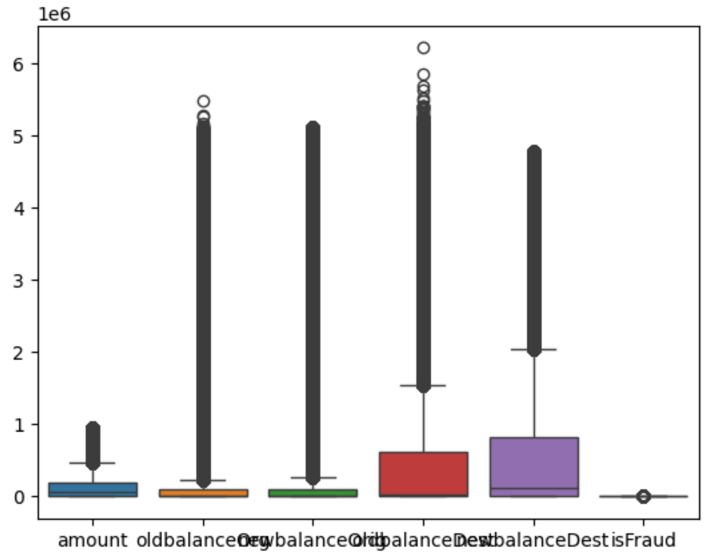
```
num=[var for var in data.columns if data[var].dtype!='O' and var!='isFraud']
num
```

```
['amount',
 'oldbalanceOrg',
 'newbalanceOrig',
 'oldbalanceDest',
 'newbalanceDest']
```

```
from scipy import stats
for x in num:
  bmi_z_score=stats.zscore(data[x])
  data=data[np.abs(bmi_z_score)<=3]
```

```
sns.boxplot(data)
```

```
<Axes: >
```



Data preprocessing

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
data["type"]=le.fit_transform(data["type"])
```

```
data["type"].value_counts()
```

type	count
3	353786
1	344994
0	153946
4	60201
2	6691

dtype: int64

```
# Dividing the dataset into dependent and independent y and x respectively
x=data.drop("isFraud",axis=1)
y=data["isFraud"]
```

```
x.head()
```

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

```
y.head()
```

	isFraud
0	0
1	0
2	1
3	1
4	0

dtype: int64

Splitting the data:

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```