```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


data = pd.read_csv('/content/PS_20174392719_1491204439457_log.csv')
print(data.shape)
data.head()
```

```
(6362620, 11)
```

| | step | type | amount | nameOrig | oldbalanceOrg | newbalanceOrig | nameDest |
|---|---|---|---|---|---|---|---|
| **0** | 1 | PAYMENT | 9839.64 | C1231006815 | 170136.0 | 160296.36 | M1979787155 |
| **1** | 1 | PAYMENT | 1864.28 | C1666544295 | 21249.0 | 19384.72 | M2044282225 |
| **2** | 1 | TRANSFER | 181.00 | C1305486145 | 181.0 | 0.00 | C553264065 |
| **3** | 1 | CASH_OUT | 181.00 | C840083671 | 181.0 | 0.00 | C38997010 |

```
# Checking for null values in the dataset.
data.isnull().sum()
```

| | 0 |
|---|---|
| **step** | 0 |
| **type** | 0 |
| **amount** | 0 |
| **nameOrig** | 0 |
| **oldbalanceOrg** | 0 |
| **newbalanceOrig** | 0 |
| **nameDest** | 0 |
| **oldbalanceDest** | 0 |
| **newbalanceDest** | 0 |
| **isFraud** | 0 |
| **isFlaggedFraud** | 0 |

**dtype:** int64

```
# Checking the whole dataset and well known about type of features and their datatypes.
data.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
```

```
# Removing the unnecessary features for model building and training.
data.drop(columns=['nameOrig', 'nameDest'], axis=1, inplace=True)
```

Double-click (or enter) to edit

```
data.head()
```

| | step | type | amount | oldbalanceOrg | newbalanceOrig | oldbalanceDest | newbalanceD |
|---|---|---|---|---|---|---|---|
| **0** | 1 | PAYMENT | 9839.64 | 170136.0 | 160296.36 | 0.0 | |
| **1** | 1 | PAYMENT | 1864.28 | 21249.0 | 19384.72 | 0.0 | |
| **2** | 1 | TRANSFER | 181.00 | 181.0 | 0.00 | 0.0 | |
| **3** | 1 | CASH_OUT | 181.00 | 181.0 | 0.00 | 21182.0 | |

```
# Counting the number of discrete vaulues in the feature 'type'.
data.type.value_counts()
```

| type | count |
|---|---|
| **CASH_OUT** | 2237500 |
| **PAYMENT** | 2151495 |
| **CASH_IN** | 1399284 |
| **TRANSFER** | 532909 |
| **DEBIT** | 41432 |

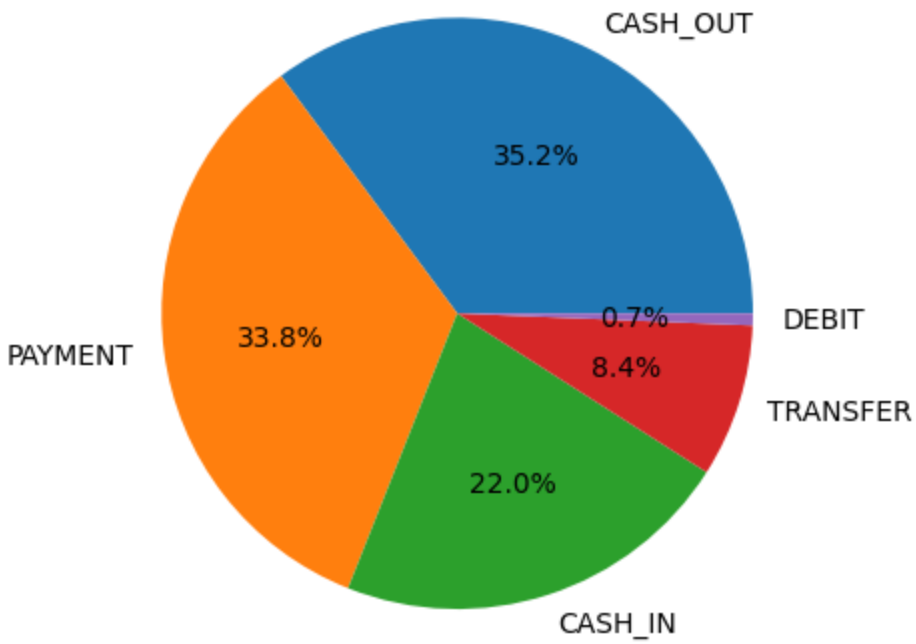
**dtype:** int64

```
target = 'isFraud'


# Segregate the counting to plot and visualize the methods of transactions.
type = data['type'].value_counts()
transactions = type.index
quantity = type.values
quantity
```

```
array([2237500, 2151495, 1399284,  532909,   41432])
```

```
# By plotting the pie chart, visualize the whole dataset.
plt.pie(quantity, labels=transactions, autopct="%1.1f%%")
plt.title('Distribution Pie Chart of Transactions Type')
plt.show()
```




```
data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2, "CASH_IN": 3, "TRANSFER": 4, '
```

```
data.info()
```

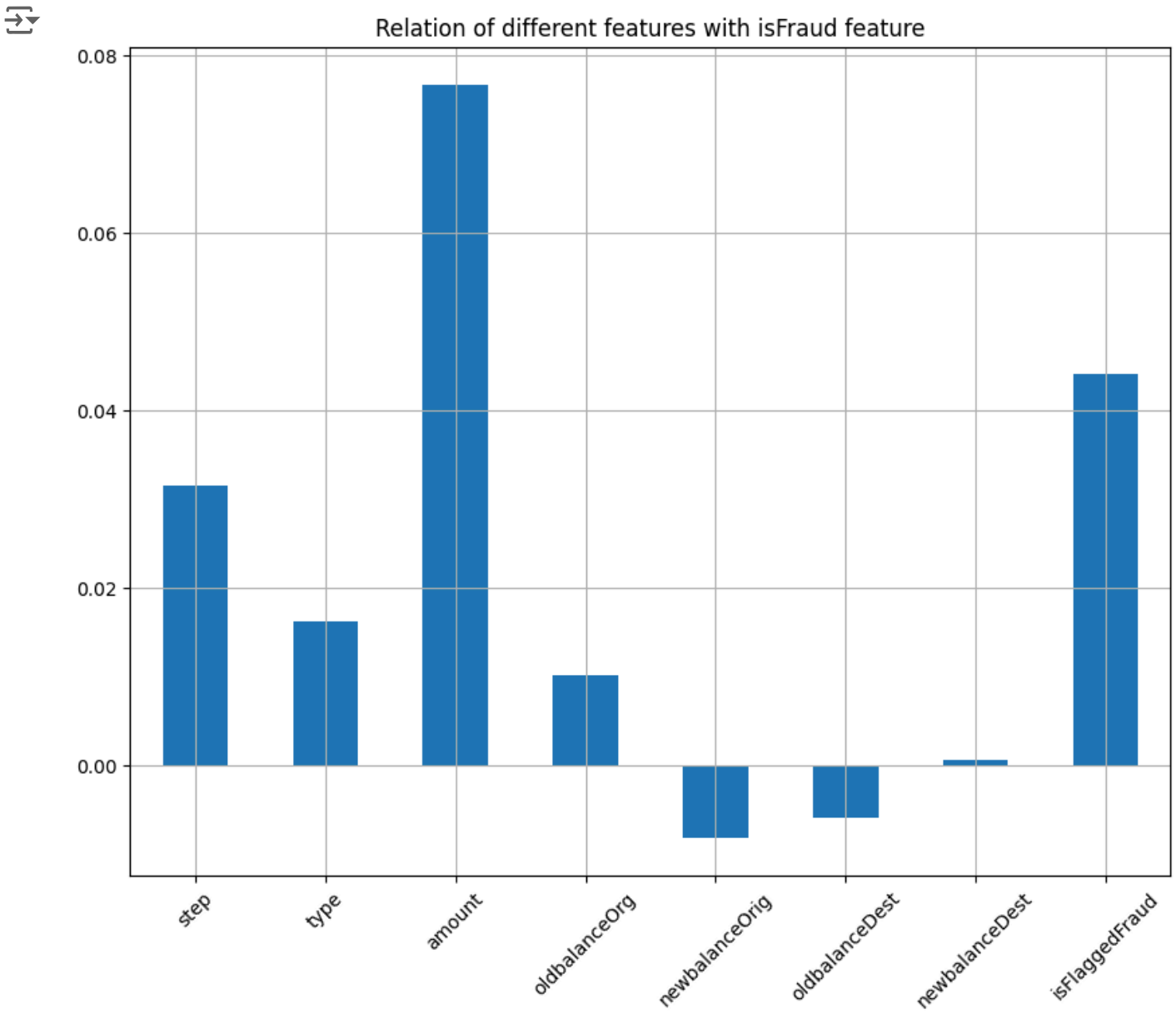
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 9 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            int64  
 2   amount          float64
 3   oldbalanceOrg   float64
 4   newbalanceOrig  float64
 5   oldbalanceDest  float64
 6   newbalanceDest  float64
 7   isFraud         int64  
 8   isFlaggedFraud  int64  
dtypes: float64(5), int64(4)
memory usage: 436.9 MB
```

```
# Checking for the relation of different features with the target variable.
correlation = data.corr()
correlation[target].sort_values(ascending=False)
```

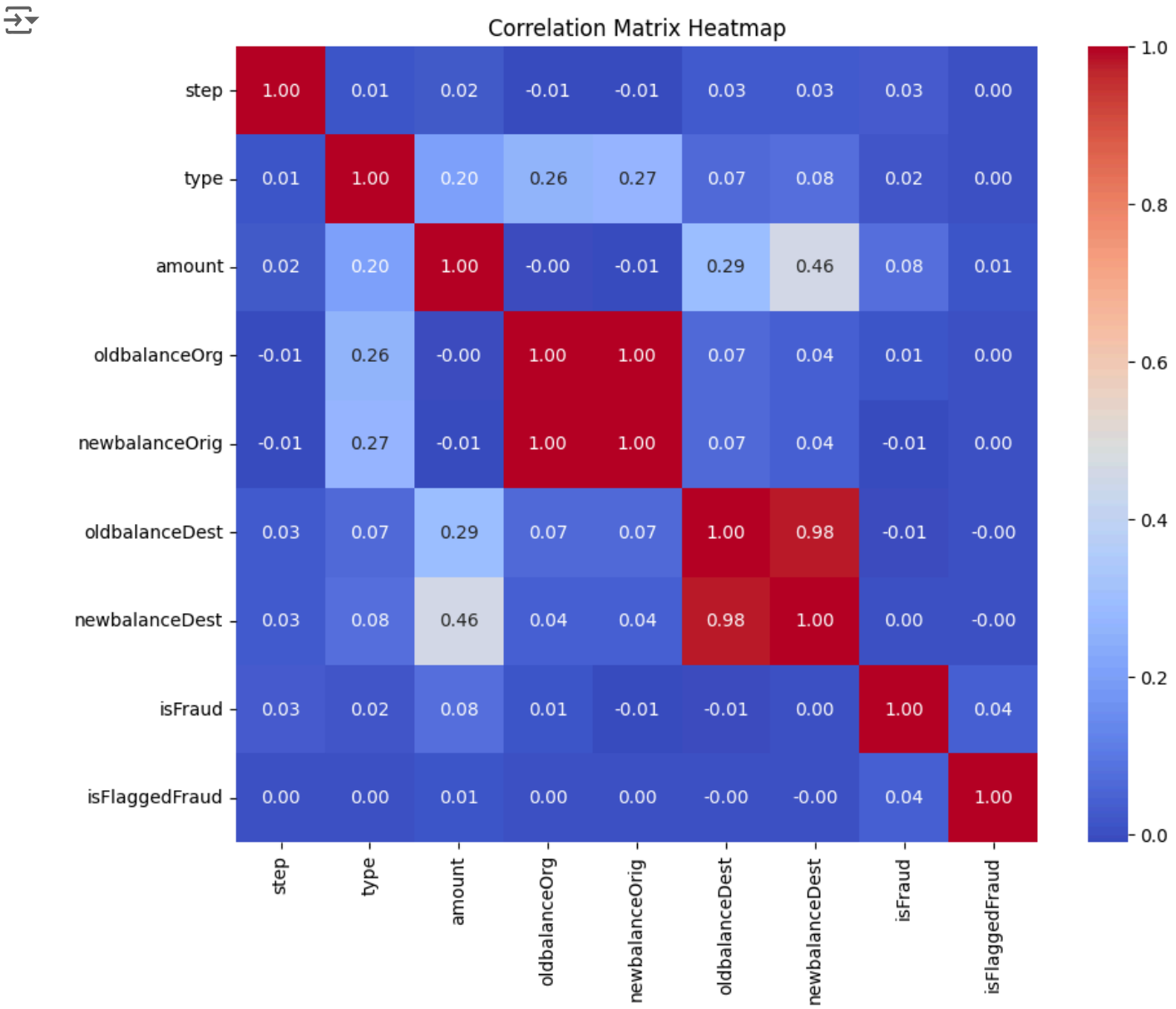
| | isFraud |
|---|---|
| **isFraud** | 1.000000 |
| **amount** | 0.076688 |
| **isFlaggedFraud** | 0.044109 |
| **step** | 0.031578 |
| **type** | 0.016171 |
| **oldbalanceOrg** | 0.010154 |
| **newbalanceDest** | 0.000535 |
| **oldbalanceDest** | -0.005885 |
| **newbalanceOrig** | -0.008148 |

**dtype:** float64

```
data2 = data.drop(columns=[target], axis=1)
data2.corrwith(data[target]).plot.bar(figsize=(10, 8), title=f'Relation of different feature
plt.show()
```




```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



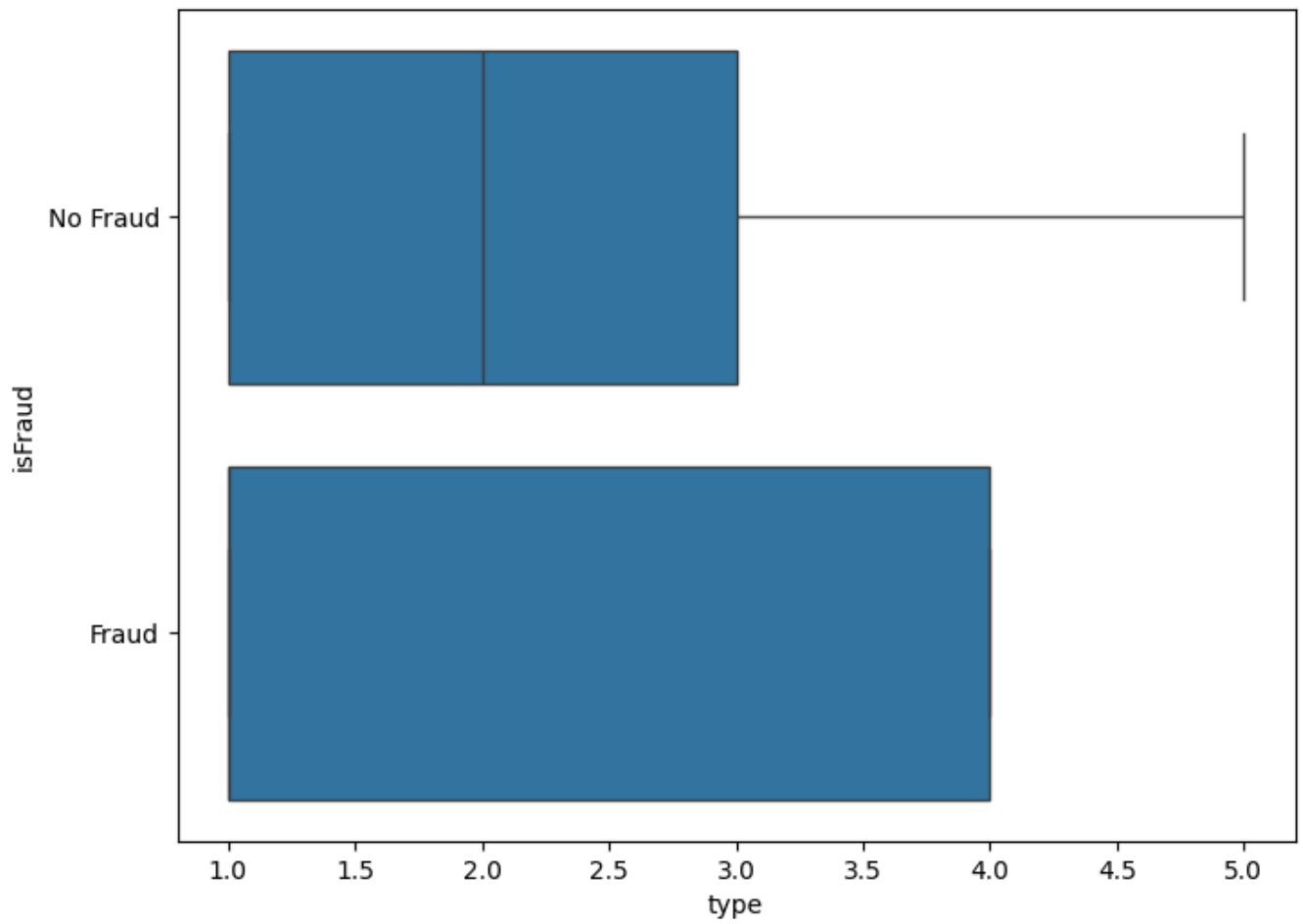

```
data[target] = data[target].map({0: "No Fraud", 1: "Fraud"})
```

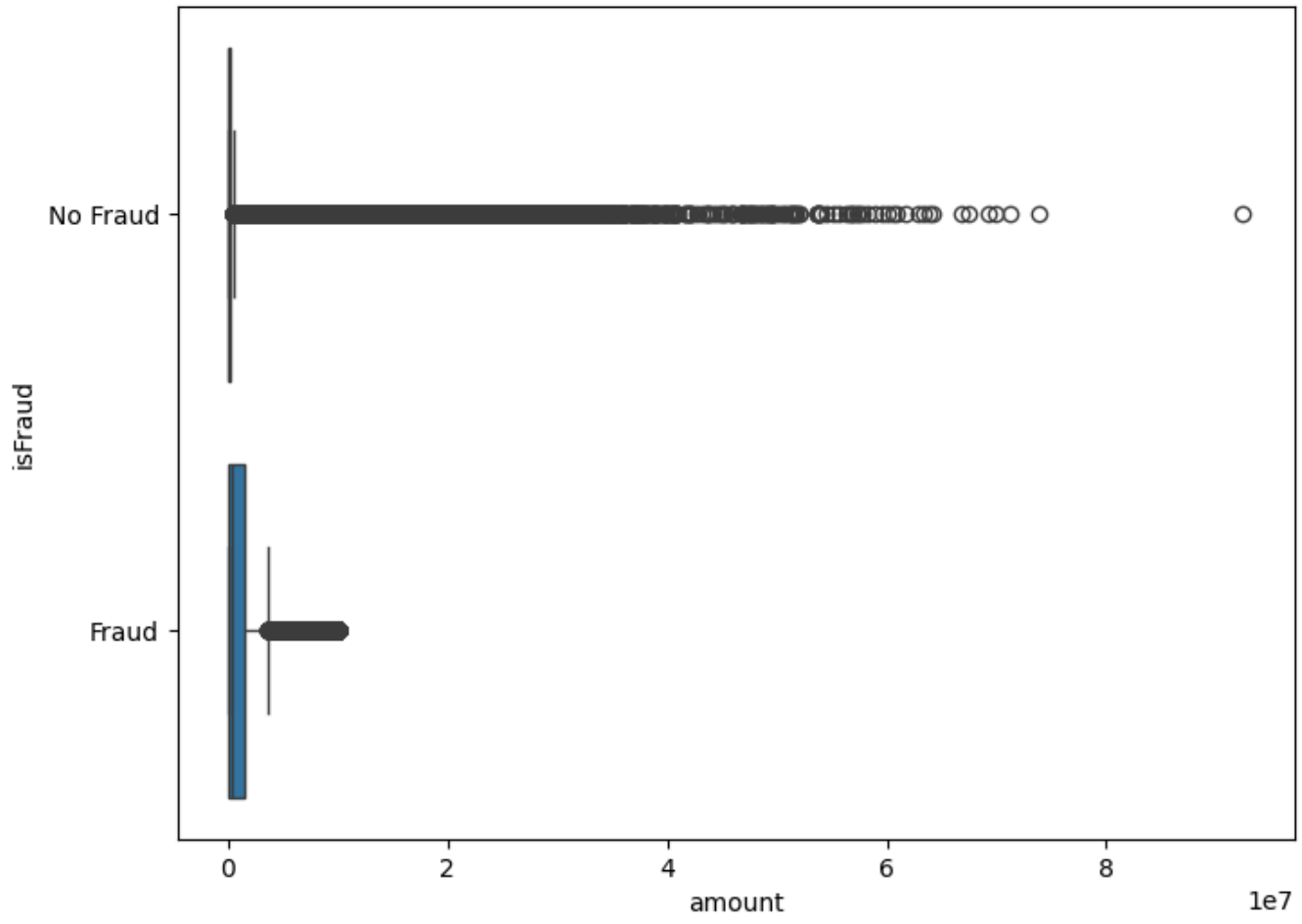
```
data.head()
```

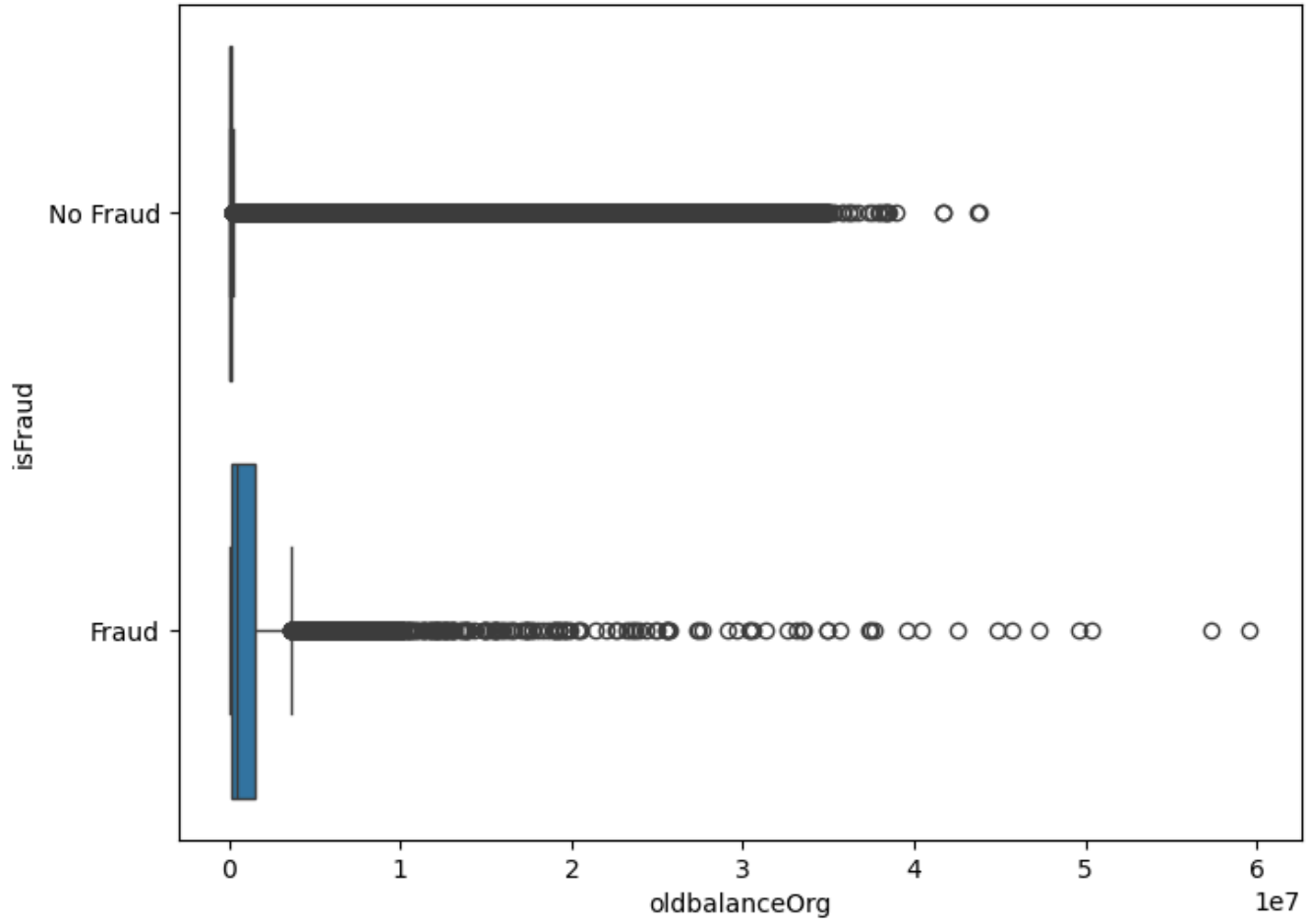
| | step | type | amount | oldbalanceOrg | newbalanceOrig | oldbalanceDest | newbalanceDest | i |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 9839.64 | 170136.0 | 160296.36 | 0.0 | 0.0 | |
| **1** | 1 | 2 | 1864.28 | 21249.0 | 19384.72 | 0.0 | 0.0 | |
| **2** | 1 | 4 | 181.00 | 181.0 | 0.00 | 0.0 | 0.0 | |

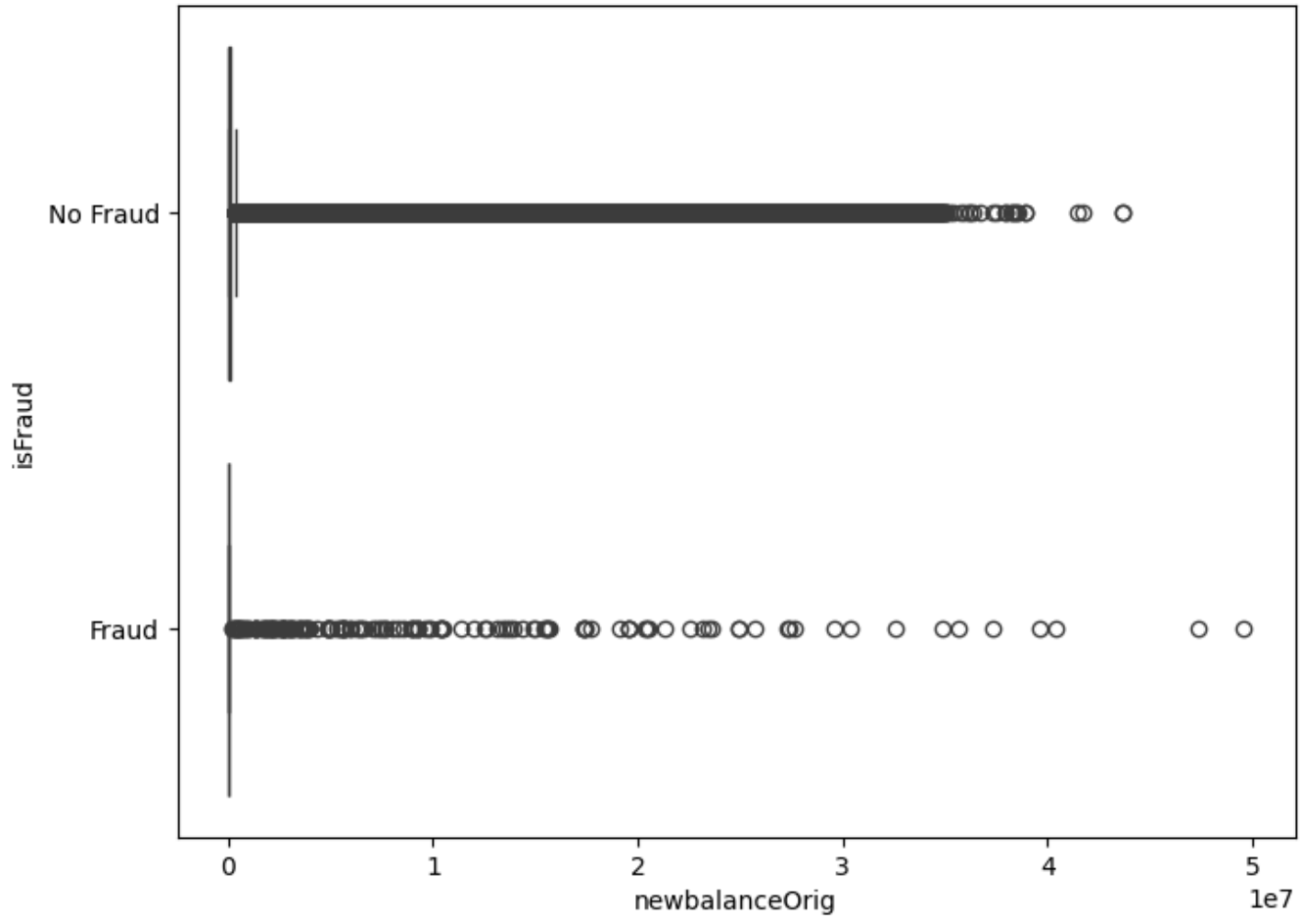
```
def plot(feature):
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=feature, y=target, data=data)
    plt.show()

features = ['type', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalar
for feature in features:
    plot(feature)
```


No Fraud
Fraud
isFraud
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
type


No Fraud
Fraud
isFraud
0
2
4
6
8
amount
1e7


No Fraud
Fraud
isFraud
0
1
2
3
4
5
6
oldbalanceOrg
1e7


No Fraud
Fraud
isFraud
0
1
2
3
4
5
newbalanceOrig
1e7