In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv("fraud.csv")
```

In [3]:
```
df.head(15)
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	n
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	
10	1	DEBIT	9644.94	C1900366749	4465.00	0.00	C997608398	10845.0	
11	1	PAYMENT	3099.97	C249177573	20771.00	17671.03	M2096539129	0.0	
12	1	PAYMENT	2560.74	C1648232591	5070.00	2509.26	M972865270	0.0	
13	1	PAYMENT	11633.76	C1716932897	10127.00	0.00	M801569151	0.0	
14	1	PAYMENT	4098.78	C1026483832	503264.00	499165.22	M1635378213	0.0	

In [4]:
```
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
```

In [5]:
```
df.describe()
```

Out[5]:

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

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [7]:
```
df['isFlaggedFraud'].value_counts()
```

Out[7]:
```
0    6362604
1         16
Name: isFlaggedFraud, dtype: int64
```

In [8]:
```
df['isFraud'].value_counts()
```

Out[8]:
```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

In [9]:
```
df['step'].value_counts()
```

Out[9]:
```
19     51352
18     49579
187    49083
235    47491
307    46968
       ...  
432        4
706        4
693        4
112        2
662        2
Name: step, Length: 743, dtype: int64
```

In [10]:
```
df['type'].value_counts()
```

Out[10]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [11]:
```
df['oldbalanceDest'].value_counts()
```

Out[11]:
```
0.00           2704388
10000000.00        615
20000000.00        219
30000000.00         86
40000000.00         31
               ...   
174945.83            1
2851171.35           1
740725.47            1
76834.40             1
732602.41            1
Name: oldbalanceDest, Length: 3614697, dtype: int64
```

In [12]:
```
df['newbalanceDest'].value_counts()
```

Out[12]:
```
0.00           2439433
10000000.00         53
9714818.91          32
19169204.93         29
16532032.16         25
               ...   
350033.41            1
302234.71            1
219824.01            1
808511.54            1
970408.13            1
Name: newbalanceDest, Length: 3555499, dtype: int64
```

In [13]:
```
df.drop('isFlaggedFraud', inplace=True, axis=1)
```

In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [15]:
```
df.drop('step', inplace=True, axis=1)
```

In [16]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

In [17]:
```
df['type'] = label_encoder.fit_transform(df['type'])
df['nameOrig'] = label_encoder.fit_transform(df['nameOrig'])
df['nameDest'] = label_encoder.fit_transform(df['nameDest'])
```

In [18]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, fmt='.2f')
```

Out[18]:
```
<AxesSubplot:>
```

In [19]:
```
df.drop(['nameOrig','nameDest'], inplace=True, axis=1)
```

In [20]:
```
from sklearn.preprocessing import MinMaxScaler
scalar = MinMaxScaler()
```

In [21]:
```
X = df.drop('isFraud',axis=1)
y = df['isFraud']
```

In [22]:
```
X = scalar.fit_transform(X)
```

In [23]:
```
from sklearn.model_selection import train_test_split as tt
X_train,X_test,y_train,y_test = tt(X,y,test_size=0.2,random_state=0)
```

In [24]:
```
from sklearn.linear_model import LogisticRegression as LR
lr = LR()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
```

In [25]:
```
y_pred
```

Out[25]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [26]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[26]:
```
0.9988078810301416
```

In [27]:
```
from sklearn.tree import DecisionTreeClassifier as DTC
model = DTC()
```

In [28]:
```
model.fit(X_train, y_train)
y_model = model.predict(X_test)
```

In [29]:
```
y_model
```

Out[29]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [30]:
```
accuracy_score(y_test, y_model)
```

Out[30]:
```
0.9997139543144177
```

In [ ]:
```

```