Fraud Transactions prediction

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
```

In [2]:

```
df=pd.read_csv("C:\\Users\\dell\\Downloads\\Fraud.csv")
```

In [3]:

```
df.head()#to see Top 5 rows from the given data
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	c
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	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

In [4]:

```
df.tail()#Last 5 rows from the given data
```

Out[4]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C7769
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C18818
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C13651
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C20803
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C8732

Column Descriptions

step - maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).

type - CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.

amount - amount of the transaction in local currency.

nameOrig - customer who started the transaction

oldbalanceOrg - initial balance before the transaction

newbalanceOrig - new balance after the transaction

nameDest - customer who is the recipient of the transaction

oldbalanceDest - initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).

newbalanceDest - new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).

isFraud - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.

isFlaggedFraud - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

In [5]:

```
df.shape #number of rows and columns
```

Out[5]:

```
(6362620, 11)
```

In [6]:

```
df.describe()#for mean,median,mode
```

Out[6]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalar
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561

In [7]:

```
df['step'].max()
```

Out[7]:

```
743
```

The step variable starts from 1 hour to 743 hour (30 days).

median for the newbalance_orig is 0.

In [8]:

```
categorical_attributes = df.select_dtypes(include='object')#To include object dtype colum
```

In [9]:

```
categorical_attributes.describe()
```

Out[9]:

	type	nameOrig	nameDest
count	6362620	6362620	6362620
unique	5	6353307	2722362
top	CASH_OUT	C1902386530	C1286084959
freq	2237500	3	113

The majority type is CASH_OUT with 2237500.

In [10]:

```
df.info()#to check the Dtype of that column
```

```
<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 [11]:

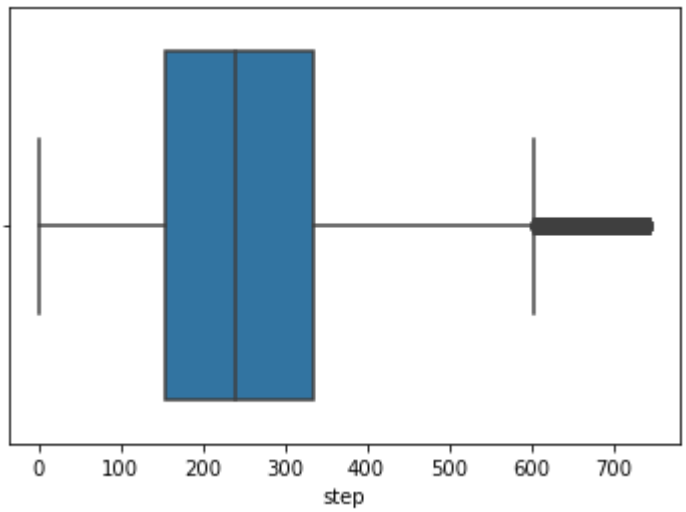
```
df.isnull().sum()#no missing values present in this dataset
```

Out[11]:

```
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 [12]:

```
n=df.select_dtypes(exclude='object')
for i in n.columns:
    sns.boxplot(data=n,x=i)
    plt.show()
```



In [13]:

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

Out[13]:

```
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 [14]:

```
df['type'].unique()#5 types of payments present.
```

Out[14]:

```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [15]:

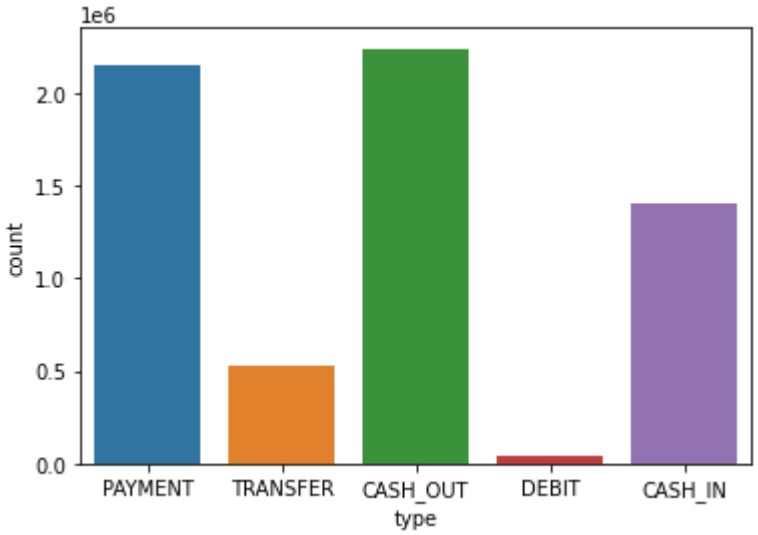
```
df['type'].value_counts()#no.of times that payment occured.
```

Out[15]:

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

In [16]:

```
sns.countplot(df['type'])
plt.show()
```



In [17]:

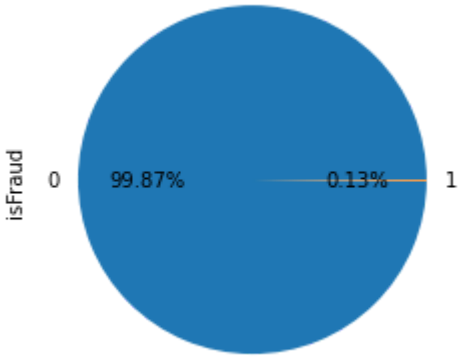
```
df['isFraud'].value_counts()#8213 transactions made by the fraudulent agents.
```

Out[17]:

```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

In [18]:

```
df['isFraud'].value_counts().plot(kind='pie',autopct='%0.2f%%')
plt.show()
```



In [19]:

```
df['isFlaggedFraud'].value_counts()# 16 illegal attempts to transfer more than 2,00,000 i
```

Out[19]:

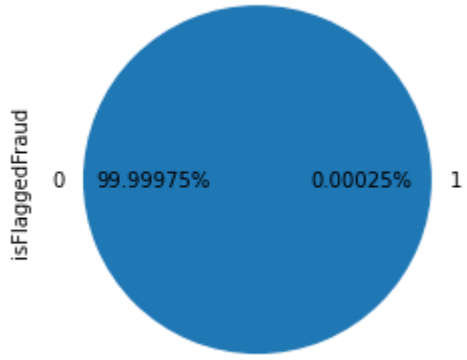
```
0    6362604
1         16
Name: isFlaggedFraud, dtype: int64
```

In [20]:

```
df['isFlaggedFraud'].value_counts().plot(kind='pie',autopct='%0.5f%%')
```

Out[20]:

```
<AxesSubplot:ylabel='isFlaggedFraud'>
```



In [21]:

```
payment=df[(df.type == 'TRANSFER') & (df.isFraud==1)]
payment
```

Out[21]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C5532
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C9727
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C18484
969	1	TRANSFER	1277212.77	C1334405552	1277212.77	0.0	C4316
1115	1	TRANSFER	35063.63	C1364127192	35063.63	0.0	C11364
...	...	...	...	...	...	...	
6362610	742	TRANSFER	63416.99	C778071008	63416.99	0.0	C18125
6362612	743	TRANSFER	1258818.82	C1531301470	1258818.82	0.0	C14709
6362614	743	TRANSFER	339682.13	C2013999242	339682.13	0.0	C18504
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C18818
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C20803

4097 rows × 11 columns

In [22]:

```
payment=df[(df.type == 'CASH_OUT') & (df.isFraud==1)]
payment
```

Out[22]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C389
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C10072
681	1	CASH_OUT	20128.00	C1118430673	20128.00	0.0	C3399
724	1	CASH_OUT	416001.33	C749981943	0.00	0.0	C6673
970	1	CASH_OUT	1277212.77	C467632528	1277212.77	0.0	C7160
...	...	...	...	...	...	...	
6362611	742	CASH_OUT	63416.99	C994950684	63416.99	0.0	C16622
6362613	743	CASH_OUT	1258818.82	C1436118706	1258818.82	0.0	C12407
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C7769
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C13651
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C8732

4116 rows × 11 columns

In [23]:

```
payment=df[(df.type == 'TRANSFER') & (df.isFlaggedFraud==1)]
print(len(payment))
payment
```

16

Out[23]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2736446	212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C639
3247297	250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C1147
3760288	279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C110(
5563713	387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C89'
5996407	425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C1392
5996409	425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C1109
6168499	554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C484
6205439	586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C177(
6266413	617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C661
6281482	646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C1806
6281484	646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C1909
6296014	671	TRANSFER	3441041.46	C917414431	3441041.46	3441041.46	C1082
6351225	702	TRANSFER	3171085.59	C1892216157	3171085.59	3171085.59	C1308
6362460	730	TRANSFER	10000000.00	C2140038573	17316255.05	17316255.05	C1395
6362462	730	TRANSFER	7316255.05	C1869569059	17316255.05	17316255.05	C1861
6362584	741	TRANSFER	5674547.89	C992223106	5674547.89	5674547.89	C1366

In [24]:

```
fraud=df[df.isFraud==1]
fraud
```

Out[24]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C5532
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C389
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C9727
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C10072
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C18484
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C7769
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C18818
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C13651
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C20803
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C8732

8213 rows × 11 columns

In [25]:

```
fraud=df[df.isFlaggedFraud==1]
fraud
```

Out[25]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2736446	212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C639
3247297	250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C1147
3760288	279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C1100
5563713	387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C891
5996407	425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C1392
5996409	425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C1109
6168499	554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C484
6205439	586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C1770
6266413	617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C661
6281482	646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C1806
6281484	646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C1909
6296014	671	TRANSFER	3441041.46	C917414431	3441041.46	3441041.46	C1082
6351225	702	TRANSFER	3171085.59	C1892216157	3171085.59	3171085.59	C1308
6362460	730	TRANSFER	10000000.00	C2140038573	17316255.05	17316255.05	C1395
6362462	730	TRANSFER	7316255.05	C1869569059	17316255.05	17316255.05	C1861
6362584	741	TRANSFER	5674547.89	C992223106	5674547.89	5674547.89	C1366

In [26]:

```
df['step_days'] = df['step'].apply(lambda i: i/24)
df['step_weeks'] = df['step'].apply(lambda i: i/(24*7))

# difference between initial balance before the transaction and new balance after the tra
df['diff_new_old_balance'] =  df['oldbalanceOrg'] -df['newbalanceOrig']

# difference between initial balance recipient before the transaction and new balance rec
df['diff_new_old_destiny'] =  df['oldbalanceDest']- df['newbalanceDest']

# name orig and name dest
df['nameOrig'] = df['nameOrig'].apply(lambda i: i[0])
df['nameDest'] = df['nameDest'].apply(lambda i: i[0])
```

In [27]:

```
df.head()
```

Out[27]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbala
0	1	PAYMENT	9839.64	C	170136.0	160296.36	M	
1	1	PAYMENT	1864.28	C	21249.0	19384.72	M	
2	1	TRANSFER	181.00	C	181.0	0.00	C	
3	1	CASH_OUT	181.00	C	181.0	0.00	C	
4	1	PAYMENT	11668.14	C	41554.0	29885.86	M	

In [28]:

```
df.tail()
```

Out[28]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
6362615	743	CASH_OUT	339682.13	C	339682.13	0.0	C
6362616	743	TRANSFER	6311409.28	C	6311409.28	0.0	C
6362617	743	CASH_OUT	6311409.28	C	6311409.28	0.0	C
6362618	743	TRANSFER	850002.52	C	850002.52	0.0	C
6362619	743	CASH_OUT	850002.52	C	850002.52	0.0	C

In [29]:

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

Out[29]:

```
0.791667     51352
0.750000     49579
7.791667     49083
9.791667     47491
12.791667    46968
             ...
18.000000        4
29.416667        4
28.875000        4
4.666667         2
27.583333        2
Name: step_days, Length: 743, dtype: int64
```

In [30]:

```
print('\n The types of fraudulent transaction are {}'.format(list(df.loc[df.isFraud == 1]

FraudTransfer = df.loc[(df.isFraud == 1) & (df.type == 'TRANSFER')]
FraudCashout = df.loc[(df.isFraud == 1) & (df.type == 'CASH_OUT')]
FraudPayment=df.loc[(df.isFraud==1) & (df.type=='PAYMENT')]

print('\n The number of fraudulent TRANSFERs = {}'.format(len(FraudTransfer)))
print('\n The number of fraudulent CASH_OUTs = {}'.format(len(FraudCashout)))
print('\n The number of fraudulent PAYMENT = {}'.format(len(FraudPayment)))
```

```
 The types of fraudulent transaction are ['TRANSFER', 'CASH_OUT']

 The number of fraudulent TRANSFERs = 4097

 The number of fraudulent CASH_OUTs = 4116

 The number of fraudulent PAYMENT = 0
```

In [31]:

```
setpdays=df.loc[df.isFraud == 1].step_days.drop_duplicates().values
print(setpdays.min())
print(setpdays.max())
```

```
0.041666666666666664
30.958333333333332
```

In [32]:

```
setpdays=df.loc[df.isFlaggedFraud == 1].step_days.drop_duplicates().values
print(setpdays.min())
print(setpdays.max())
```

```
8.833333333333334
30.875
```

In [33]:

```
CountisFlaggedFraud = df.loc[(df.isFlaggedFraud == 1)]

CountisFlaggedFraudWithTransfer = df.loc[(df.isFlaggedFraud == 1) & (df.type == 'TRANSFER

print(len(CountisFlaggedFraud))
print(len(CountisFlaggedFraudWithTransfer))
print(df.shape)
print(CountisFlaggedFraudWithTransfer)

print('\nThe type of transactions in which isFlaggedFraud is set: \
{}'.format(list(df.loc[df.isFlaggedFraud == 1].type.drop_duplicates())))


dfTransfer = df.loc[df.type == 'TRANSFER']
dfFlagged = df.loc[df.isFlaggedFraud == 1]
dfNotFlagged = df.loc[df.isFlaggedFraud == 0]

print('\n The minimum amount transacted when isFlaggedFraud is set ={}'.format(dfFlagged.

print('\n The max amount transacted when isFlaggedFraud is set ={}'.format(dfFlagged.amou

print('\nThe max amount is TRANSFERED when isFlaggedFraud is NOT set ={}'.format(dfNotFla
```

```
16
16
(6362620, 15)
         step      type       amount nameOrig  oldbalanceOrg  newbalanceOr
ig  \
2736446   212  TRANSFER   4953893.08        C     4953893.08      4953893.
08
3247297   250  TRANSFER   1343002.08        C     1343002.08      1343002.
08
3760288   279  TRANSFER    536624.41        C      536624.41       536624.
41
5563713   387  TRANSFER   4892193.09        C     4892193.09      4892193.
09
5996407   425  TRANSFER  10000000.00        C    19585040.37     19585040.
37
5996409   425  TRANSFER   9585040.37        C    19585040.37     19585040.
37
6168499   554  TRANSFER   3576297.10        C     3576297.10      3576297.
10
6205439   586  TRANSFER    353874.22        C      353874.22       353874.
22
6266413   617  TRANSFER   2542664.27        C     2542664.27      2542664.
27
6281482   646  TRANSFER  10000000.00        C    10399045.08     10399045.
08
6281484   646  TRANSFER    399045.08        C    10399045.08     10399045.
08
6296014   671  TRANSFER   3441041.46        C     3441041.46      3441041.
46
6351225   702  TRANSFER   3171085.59        C     3171085.59      3171085.
59
6362460   730  TRANSFER  10000000.00        C    17316255.05     17316255.
05
6362462   730  TRANSFER   7316255.05        C    17316255.05     17316255.
05
6362584   741  TRANSFER   5674547.89        C     5674547.89      5674547.
89

        nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
\
2736446        C             0.0             0.0        1               1
3247297        C             0.0             0.0        1               1
3760288        C             0.0             0.0        1               1
5563713        C             0.0             0.0        1               1
5996407        C             0.0             0.0        1               1
5996409        C             0.0             0.0        1               1
6168499        C             0.0             0.0        1               1
6205439        C             0.0             0.0        1               1
6266413        C             0.0             0.0        1               1
6281482        C             0.0             0.0        1               1
6281484        C             0.0             0.0        1               1
6296014        C             0.0             0.0        1               1
6351225        C             0.0             0.0        1               1
6362460        C             0.0             0.0        1               1
6362462        C             0.0             0.0        1               1
6362584        C             0.0             0.0        1               1

         step_days  step_weeks  diff_new_old_balance  diff_new_old_destiny
2736446   8.833333    1.261905                   0.0                   0.0
3247297  10.416667    1.488095                   0.0                   0.0
3760288  11.625000    1.660714                   0.0                   0.0
```

```
5563713  16.125000    2.303571                0.0                     0.0
5996407  17.708333    2.529762                0.0                     0.0
5996409  17.708333    2.529762                0.0                     0.0
6168499  23.083333    3.297619                0.0                     0.0
6205439  24.416667    3.488095                0.0                     0.0
6266413  25.708333    3.672619                0.0                     0.0
6281482  26.916667    3.845238                0.0                     0.0
6281484  26.916667    3.845238                0.0                     0.0
6296014  27.958333    3.994048                0.0                     0.0
6351225  29.250000    4.178571                0.0                     0.0
6362460  30.416667    4.345238                0.0                     0.0
6362462  30.416667    4.345238                0.0                     0.0
6362584  30.875000    4.410714                0.0                     0.0
```

```
The type of transactions in which isFlaggedFraud is set: ['TRANSFER']
```

```
 The minimum amount transacted when isFlaggedFraud is set =353874.22
```

```
 The max amount transacted when isFlaggedFraud is set =10000000.0
```

```
The max amount is TRANSFERED when isFlaggedFraud is NOT set =92445516.64
```

Fraud transactions occours with in 1 hour with the amount which is less then 200000.

Fraud transactions occours with in 8 days with the amount which is more then 200000.

On average of Fraud transactions occours which is equal to 100000 with in 15 days.

In [34]:

```
FlaggedFound = df.loc[(df.isFraud == 1)]
print(FlaggedFound.describe())
print('\n')
print(FlaggedFound.median())
print('\n')
print(FlaggedFound.max())
print('\n')
```

```
              step        amount  oldbalanceOrg  newbalanceOrig  \
count  8213.000000  8.213000e+03   8.213000e+03    8.213000e+03
mean    368.413856  1.467967e+06   1.649668e+06    1.923926e+05
std     216.388690  2.404253e+06   3.547719e+06    1.965666e+06
min       1.000000  0.000000e+00   0.000000e+00    0.000000e+00
25%     181.000000  1.270913e+05   1.258224e+05    0.000000e+00
50%     367.000000  4.414234e+05   4.389835e+05    0.000000e+00
75%     558.000000  1.517771e+06   1.517771e+06    0.000000e+00
max     743.000000  1.000000e+07   5.958504e+07    4.958504e+07

       oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud    step_day
s  \
count    8.213000e+03    8.213000e+03   8213.0     8213.000000  8213.00000
0
mean     5.442496e+05    1.279708e+06      1.0        0.001948    15.35057
7
std      3.336421e+06    3.908817e+06      0.0        0.044097     9.01619
5
min      0.000000e+00    0.000000e+00      1.0        0.000000     0.04166
7
25%      0.000000e+00    0.000000e+00      1.0        0.000000     7.54166
7
50%      0.000000e+00    4.676420e+03      1.0        0.000000    15.29166
7
75%      1.478287e+05    1.058725e+06      1.0        0.000000    23.25000
0
max      2.362305e+08    2.367265e+08      1.0        1.000000    30.95833
3

        step_weeks  diff_new_old_balance  diff_new_old_destiny
count  8213.000000          8.213000e+03          8.213000e+03
mean      2.192940          1.457275e+06         -7.354580e+05
std       1.288028          2.396099e+06          1.856984e+06
min       0.005952          0.000000e+00         -1.491511e+07
25%       1.077381          1.245826e+05         -4.452574e+05
50%       2.184524          4.363175e+05          0.000000e+00
75%       3.321429          1.503035e+06          0.000000e+00
max       4.422619          1.000000e+07          3.152261e+05


step                     367.000000
amount                441423.440000
oldbalanceOrg         438983.450000
newbalanceOrig             0.000000
oldbalanceDest             0.000000
newbalanceDest          4676.420000
isFraud                    1.000000
isFlaggedFraud             0.000000
step_days                 15.291667
step_weeks                 2.184524
diff_new_old_balance  436317.490000
diff_new_old_destiny       0.000000
dtype: float64


step                           743
type                      TRANSFER
amount                 10000000.0
nameOrig                        C
oldbalanceOrg           59585040.37
newbalanceOrig          49585040.37
```

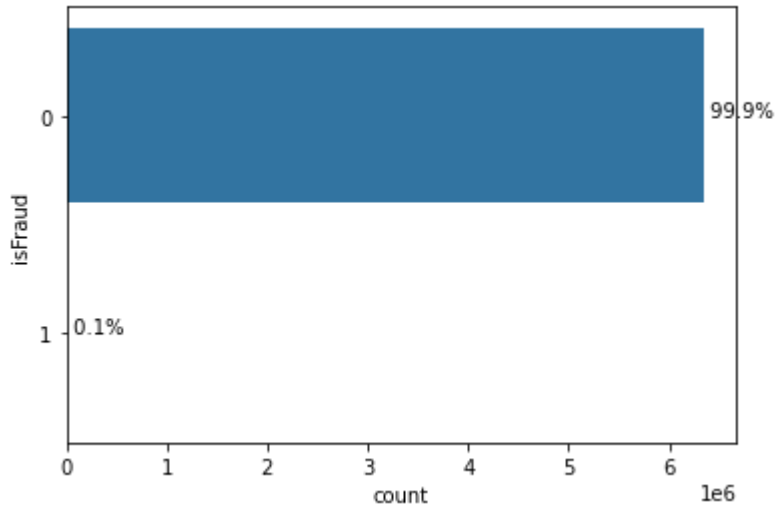
```
nameDest                      C
oldbalanceDest        236230516.82
newbalanceDest        236726494.66
isFraud                          1
isFlaggedFraud                   1
step_days                30.958333
step_weeks                4.422619
diff_new_old_balance     10000000.0
diff_new_old_destiny      315226.07
dtype: object
```

Univariate Analysis

In [35]:

```
ax = sns.countplot(y='isFraud', data=df);

total = df['isFraud'].size
for p in ax.patches:
        percentage = ' {:.1f}%'.format(100 * p.get_width()/total)
        x = p.get_x() + p.get_width() + 0.02
        y = p.get_y() + p.get_height()/2
        ax.annotate(percentage, (x, y))
```



In [36]:

```
df.dtypes
```

Out[36]:

```
step                    int64
type                   object
amount                float64
nameOrig               object
oldbalanceOrg         float64
newbalanceOrig        float64
nameDest               object
oldbalanceDest        float64
newbalanceDest        float64
isFraud                 int64
isFlaggedFraud          int64
step_days             float64
step_weeks            float64
diff_new_old_balance  float64
diff_new_old_destiny  float64
dtype: object
```

Model Implementation

In [37]:

```
from sklearn.preprocessing import LabelEncoder
lr=LabelEncoder()
```

In [38]:

```
df['type']=lr.fit_transform(df['type'])
df['nameOrig']=lr.fit_transform(df['nameOrig'])
df['nameDest'] =lr.fit_transform(df['nameDest'])
```

In [39]:

```
df.dtypes
```

Out[39]:

```
step                    int64
type                    int32
amount                float64
nameOrig                int32
oldbalanceOrg         float64
newbalanceOrig        float64
nameDest                int32
oldbalanceDest        float64
newbalanceDest        float64
isFraud                 int64
isFlaggedFraud          int64
step_days             float64
step_weeks            float64
diff_new_old_balance  float64
diff_new_old_destiny  float64
dtype: object
```

In [40]:

```
FlaggedFound = df.loc[(df.isFraud == 1)]
print(FlaggedFound.describe())
print('\n')
print(FlaggedFound.median())
print('\n')
print(FlaggedFound.max())
print('\n')
```

```
              step         type        amount  nameOrig  oldbalanceOrg  \
count  8213.000000  8213.000000  8.213000e+03    8213.0   8.213000e+03
mean    368.413856     2.496530  1.467967e+06       0.0   1.649668e+06
std     216.388690     1.500087  2.404253e+06       0.0   3.547719e+06
min       1.000000     1.000000  0.000000e+00       0.0   0.000000e+00
25%     181.000000     1.000000  1.270913e+05       0.0   1.258224e+05
50%     367.000000     1.000000  4.414234e+05       0.0   4.389835e+05
75%     558.000000     4.000000  1.517771e+06       0.0   1.517771e+06
max     743.000000     4.000000  1.000000e+07       0.0   5.958504e+07

       newbalanceOrig  nameDest  oldbalanceDest  newbalanceDest  isFraud
\
count    8.213000e+03    8213.0    8.213000e+03    8.213000e+03   8213.0
mean     1.923926e+05       0.0    5.442496e+05    1.279708e+06      1.0
std      1.965666e+06       0.0    3.336421e+06    3.908817e+06      0.0
min      0.000000e+00       0.0    0.000000e+00    0.000000e+00      1.0
25%      0.000000e+00       0.0    0.000000e+00    0.000000e+00      1.0
50%      0.000000e+00       0.0    0.000000e+00    4.676420e+03      1.0
75%      0.000000e+00       0.0    1.478287e+05    1.058725e+06      1.0
max      4.958504e+07       0.0    2.362305e+08    2.367265e+08      1.0

       isFlaggedFraud    step_days   step_weeks  diff_new_old_balance  \
count     8213.000000  8213.000000  8213.000000          8.213000e+03
mean         0.001948    15.350577     2.192940          1.457275e+06
std          0.044097     9.016195     1.288028          2.396099e+06
min          0.000000     0.041667     0.005952          0.000000e+00
25%          0.000000     7.541667     1.077381          1.245826e+05
50%          0.000000    15.291667     2.184524          4.363175e+05
75%          0.000000    23.250000     3.321429          1.503035e+06
max          1.000000    30.958333     4.422619          1.000000e+07

       diff_new_old_destiny
count          8.213000e+03
mean          -7.354580e+05
std            1.856984e+06
min           -1.491511e+07
25%           -4.452574e+05
50%            0.000000e+00
75%            0.000000e+00
max            3.152261e+05


step                      367.000000
type                        1.000000
amount                 441423.440000
nameOrig                    0.000000
oldbalanceOrg          438983.450000
newbalanceOrig              0.000000
nameDest                    0.000000
oldbalanceDest              0.000000
newbalanceDest           4676.420000
isFraud                     1.000000
isFlaggedFraud              0.000000
step_days                  15.291667
step_weeks                  2.184524
diff_new_old_balance   436317.490000
diff_new_old_destiny        0.000000
dtype: float64


step                    7.430000e+02
```

```
type                  4.000000e+00
amount                1.000000e+07
nameOrig              0.000000e+00
oldbalanceOrg         5.958504e+07
newbalanceOrig        4.958504e+07
nameDest              0.000000e+00
oldbalanceDest        2.362305e+08
newbalanceDest        2.367265e+08
isFraud               1.000000e+00
isFlaggedFraud        1.000000e+00
step_days             3.095833e+01
step_weeks            4.422619e+00
diff_new_old_balance  1.000000e+07
diff_new_old_destiny  3.152261e+05
dtype: float64
```

In [41]:

```
fraud=df.values
fraud
```

Out[41]:

```
array([[ 1.00000000e+00,  3.00000000e+00,  9.83964000e+03, ...,
         5.95238095e-03,  9.83964000e+03,  0.00000000e+00],
       [ 1.00000000e+00,  3.00000000e+00,  1.86428000e+03, ...,
         5.95238095e-03,  1.86428000e+03,  0.00000000e+00],
       [ 1.00000000e+00,  4.00000000e+00,  1.81000000e+02, ...,
         5.95238095e-03,  1.81000000e+02,  0.00000000e+00],
       ...,
       [ 7.43000000e+02,  1.00000000e+00,  6.31140928e+06, ...,
         4.42261905e+00,  6.31140928e+06, -6.31140927e+06],
       [ 7.43000000e+02,  4.00000000e+00,  8.50002520e+05, ...,
         4.42261905e+00,  8.50002520e+05,  0.00000000e+00],
       [ 7.43000000e+02,  1.00000000e+00,  8.50002520e+05, ...,
         4.42261905e+00,  8.50002520e+05, -8.50002520e+05]])
```

In [42]:

```
X =fraud[:,0:10] #Predictors
y = fraud[:,10] #Target
print(X)
print(y)
```

```
[[1.00000000e+00 3.00000000e+00 9.83964000e+03 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.00000000e+00 3.00000000e+00 1.86428000e+03 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.00000000e+00 4.00000000e+00 1.81000000e+02 ... 0.00000000e+00
  0.00000000e+00 1.00000000e+00]
 ...
 [7.43000000e+02 1.00000000e+00 6.31140928e+06 ... 6.84888400e+04
  6.37989811e+06 1.00000000e+00]
 [7.43000000e+02 4.00000000e+00 8.50002520e+05 ... 0.00000000e+00
  0.00000000e+00 1.00000000e+00]
 [7.43000000e+02 1.00000000e+00 8.50002520e+05 ... 6.51009911e+06
  7.36010163e+06 1.00000000e+00]]
[0. 0. 0. ... 0. 0. 0.]
```

In [43]:

```
df.head()
```

Out[43]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDes
0	1	3	9839.64	0	170136.0	160296.36	1	0.(
1	1	3	1864.28	0	21249.0	19384.72	1	0.(
2	1	4	181.00	0	181.0	0.00	0	0.(
3	1	1	181.00	0	181.0	0.00	0	21182.(
4	1	3	11668.14	0	41554.0	29885.86	1	0.(

In [44]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state = 0)
```

Logistic Regression

In [45]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [46]:

```
logistic_model = LogisticRegression().fit(X_train,y_train)
ypredicted = logistic_model.predict(X_test)
```

In [47]:

```
ypredicted
```

Out[47]:

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

In [48]:

```
print(confusion_matrix(y_test,ypredicted))
```

```
[[1908779       1]
 [      6       0]]
```

In [49]:

```
print(classification_report(y_test,ypredicted))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   1908780
         1.0       0.00      0.00      0.00         6

    accuracy                           1.00   1908786
   macro avg       0.50      0.50      0.50   1908786
weighted avg       1.00      1.00      1.00   1908786
```

In [50]:

```
print('Accuracy score :',accuracy_score(y_test,ypredicted))
```

```
Accuracy score : 0.9999963327476208
```

KNN Classifier

In [51]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [52]:

```
knn=KNeighborsClassifier()
```

In [53]:

```
knn.fit(X_train,y_train)
```

Out[53]:

```
KNeighborsClassifier()
```

In [54]:

```
pred=knn.predict(X_test)
pred
```

Out[54]:

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

In [55]:

```
print(confusion_matrix(y_test,pred))
```

```
[[1908780       0]
 [      2       4]]
```

In [56]:

```
print(accuracy_score(y_test,pred))
```

```
0.9999989522136059
```

In [57]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   1908780
         1.0       1.00      0.67      0.80         6

    accuracy                           1.00   1908786
   macro avg       1.00      0.83      0.90   1908786
weighted avg       1.00      1.00      1.00   1908786
```

RandomForest

In [58]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

In [59]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state = 0)
```

In [60]:

```
classifier=RandomForestClassifier()
```

In [61]:

```
classifier.fit(X_test,y_test)
```

Out[61]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [62]:

```
y_pred=classifier.predict(X_test)
y_pred
```

Out[62]:

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

In [63]:

```
print(confusion_matrix(y_test,y_pred))
```

```
[[1908780       0]
 [      0       6]]
```

In [64]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   1908780
         1.0       1.00      1.00      1.00         6

    accuracy                           1.00   1908786
   macro avg       1.00      1.00      1.00   1908786
weighted avg       1.00      1.00      1.00   1908786
```

In [65]:

```
print(accuracy_score(y_test,pred))
```

```
0.9999989522136059
```

In [ ]: