In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
```

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

In [3]:
```python
df.head(10)
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlag
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0	
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0	
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0	
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0	
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0	
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0	

Check the shape of Dataset

In [4]:
```python
df.shape
```

Out[4]:
```
(6362620, 11)
```

In [5]:
```python
print("Row in dataset:",df.shape[0])
print("Columns in dataset:",df.shape[1])
```

```
Row in dataset: 6362620
Columns in dataset: 11
```

In [6]:
```python
df.columns
```

Out[6]:
```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

All information regarding the dataset

In [7]:
```python
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
```

The is Fraud is a class Variable we convert intothe Object Type

In [8]:
```python
df.columns
```

Out[8]:
```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

In [9]:
```
df['isFraud']=df["isFraud"].astype("object")
```

In [10]:
```
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         object
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(2), object(4)
memory usage: 534.0+ MB
```

For better understanding we rename the columnsAnd also the rearrange

In [11]:
```
df.rename(columns={'step':'Step','type':'Type','amount':'Amount','nameOrig':'Name_Orig','oldbalanceOrg':'Old_ba
```

In [12]:
```
df.head(5)
```

Out[12]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	nameDest	Old_balance_Dest	New_balance_Dest	Is_Fr
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	

In [13]:
```
df.rename(columns={'nameDest':'Name_dest','isFlaggedFraud':'Is_Flagged_Fraud'},inplace=True)
```

In [14]:
```
df.head()
```

Out[14]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_Fr
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	

Summary Statistics

In [15]:
```
#summary Statistcs for numerical variabel
df.describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
Step	6362620.0	2.433972e+02	1.423320e+02	1.0	156.00	239.000	3.350000e+02	7.430000e+02
Amount	6362620.0	1.798619e+05	6.038582e+05	0.0	13389.57	74871.940	2.087215e+05	9.244552e+07
Old_balance_Org	6362620.0	8.338831e+05	2.888243e+06	0.0	0.00	14208.000	1.073152e+05	5.958504e+07
New_balance_Orig	6362620.0	8.551137e+05	2.924049e+06	0.0	0.00	0.000	1.442584e+05	4.958504e+07
Old_balance_Dest	6362620.0	1.100702e+06	3.399180e+06	0.0	0.00	132705.665	9.430367e+05	3.560159e+08
New_balance_Dest	6362620.0	1.224996e+06	3.674129e+06	0.0	0.00	214661.440	1.111909e+06	3.561793e+08
Is_Flagged_Fraud	6362620.0	2.514687e-06	1.585775e-03	0.0	0.00	0.000	0.000000e+00	1.000000e+00

In [16]:
```python
df.describe(include='all').T
```

Out[16]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	
Step	6362620.0	NaN	NaN	NaN	243.397246	142.331971	1.0	156.0	239.0	335.0	
Type	6362620	5	CASH_OUT	2237500	NaN	NaN	NaN	NaN	NaN	NaN	
Amount	6362620.0	NaN	NaN	NaN	179861.903549	603858.231463	0.0	13389.57	74871.94	208721.4775	
Name_Orig	6362620	6353307	C1902386530	3	NaN	NaN	NaN	NaN	NaN	NaN	
Old_balance_Org	6362620.0	NaN	NaN	NaN	833883.104074	2888242.673007	0.0	0.0	14208.0	107315.175	
New_balance_Orig	6362620.0	NaN	NaN	NaN	855113.668579	2924048.502971	0.0	0.0	0.0	144258.41	
Name_dest	6362620	2722362	C1286084959	113	NaN	NaN	NaN	NaN	NaN	NaN	
Old_balance_Dest	6362620.0	NaN	NaN	NaN	1100701.66652	3399180.112969	0.0	0.0	132705.665	943036.7075	3
New_balance_Dest	6362620.0	NaN	NaN	NaN	1224996.398202	3674128.942094	0.0	0.0	214661.44	1111909.25	3
Is_Fraud	6362620.0	2.0	0.0	6354407.0	NaN	NaN	NaN	NaN	NaN	NaN	
Is_Flagged_Fraud	6362620.0	NaN	NaN	NaN	0.000003	0.001586	0.0	0.0	0.0	0.0	

Missing Values Check

In [17]:
```python
df.isnull().sum()
```

Out[17]:
```
Step                0
Type                0
Amount              0
Name_Orig           0
Old_balance_Org     0
New_balance_Orig    0
Name_dest           0
Old_balance_Dest    0
New_balance_Dest    0
Is_Fraud            0
Is_Flagged_Fraud    0
dtype: int64
```

In [18]:
```python
sum(df.isnull().sum())
```

Out[18]:
```
0
```

In [19]:
```python
df[df.duplicated()].count()
```

Out[19]:
```
Step                0
Type                0
Amount              0
Name_Orig           0
Old_balance_Org     0
New_balance_Orig    0
Name_dest           0
Old_balance_Dest    0
New_balance_Dest    0
Is_Fraud            0
Is_Flagged_Fraud    0
dtype: int64
```

In [20]:
```python
df['Is_Fraud']=df['Is_Fraud'].astype('int')
```

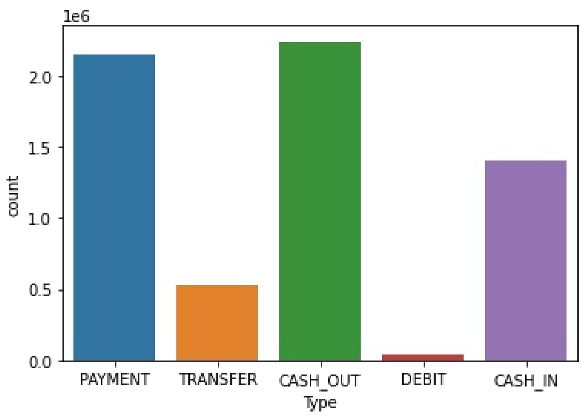
In [21]:
```python
df.info()
```

```
<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   Name_Orig         object 
 4   Old_balance_Org   float64
 5   New_balance_Orig  float64
 6   Name_dest         object 
 7   Old_balance_Dest  float64
 8   New_balance_Dest  float64
 9   Is_Fraud          int32  
 10  Is_Flagged_Fraud  int64  
dtypes: float64(5), int32(1), int64(2), object(3)
memory usage: 509.7+ MB
```

feature Extraction¶

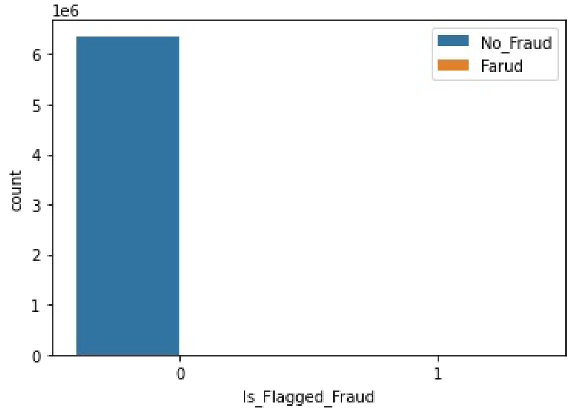
In [22]:
```
100*df[df['Is_Fraud']==1].New_balance_Orig.value_counts()/len(df[df['Is_Fraud']==1].New_balance_Orig)
```
Out[22]:
```
0.00           98.051869
17316255.05     0.036527
10399045.08     0.036527
19585040.37     0.036527
4953893.08      0.024352
                  ...
34892193.09     0.012176
1975271.77      0.012176
11975271.77     0.012176
1653144.10      0.012176
29585040.37     0.012176
Name: New_balance_Orig, Length: 145, dtype: float64
```
In [23]:
```
100*df[df['Is_Fraud']==1].New_balance_Dest.value_counts()/len(df[df['Is_Fraud']==1].New_balance_Dest)
```
Out[23]:
```
0.00           49.811275
10000000.00     0.645318
1064995.85      0.024352
127905.82       0.024352
1165187.89      0.024352
                  ...
3098931.52      0.012176
143526.32       0.012176
1532241.85      0.012176
495991.64       0.012176
7360101.63      0.012176
Name: New_balance_Dest, Length: 4067, dtype: float64
```
Data visualization¶

In [24]:
```
df.columns
```
Out[24]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```
In [25]:
```
df['Is_Flagged_Fraud'].value_counts()
```
Out[25]:
```
0    6362604
1         16
Name: Is_Flagged_Fraud, dtype: int64
```
In [26]:
```
100*df['Is_Fraud'].value_counts()/len(df["Is_Fraud"])
```
Out[26]:
```
0    99.870918
1     0.129082
Name: Is_Fraud, dtype: float64
```
what is most frequent transaction in the givin dataset

In [27]:
```
sns.countplot(x='Type',data=df)
```
Out[27]:
```
<AxesSubplot:xlabel='Type', ylabel='count'>
```


In [28]:
```
df.columns
```
Out[28]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```
In [29]:
```
sns.countplot(x='Is_Flagged_Fraud',hue='Is_Fraud',data=df)
plt.legend(labels=['No_Fraud','Farud'])
```

Out[29]:
```
<matplotlib.legend.Legend at 0x1de979facd0>
```



In [30]:
```python
df.head()
```

Out[30]:

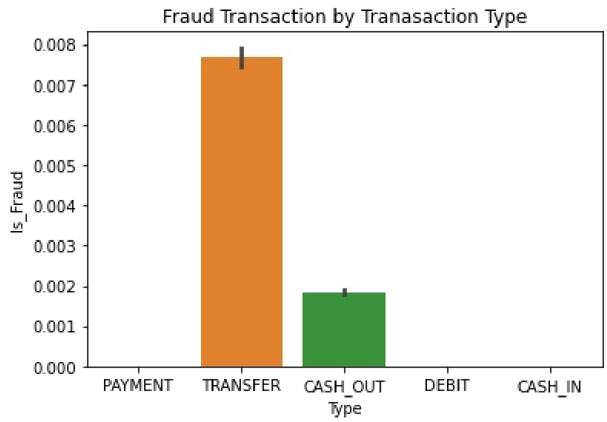
	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_Fr
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	

In [31]:
```python
100*df[df["Is_Fraud"]==1].Type.value_counts()/len(df[df["Is_Fraud"]==1].Type)
```

Out[31]:
```
CASH_OUT    50.11567
TRANSFER    49.88433
Name: Type, dtype: float64
```

In [32]:
```python
sns.barplot(x="Type",y="Is_Fraud",data=df)
plt.title('Fraud Transaction by Tranasaction Type')
```

Out[32]:
```
Text(0.5, 1.0, 'Fraud Transaction by Tranasaction Type')
```



1]Only the Transfer And cash_out tarnsaction can be fraudulent. so ther is a no Fraud obtain by using the Other transaction type like Payment, Debit and Cash_in method . 2]So we are use only two transaction type for further data visulization we can remove thos method

In [33]:
```python
# Retraining only Cash_out and Transfer type
df=df.loc[df['Type'].isin(["CASH_OUT",'TRANSFER']),:]
```

In [34]:
```python
df.reset_index(drop=True,inplace=True)
```

In [35]:
```python
df.head()
```

Out[35]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_F
0	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	
1	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	
2	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	
3	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	
4	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	

In [36]:
```python
df.shape
```

Out[36]: (2770409, 11)

In [37]:
```
Counter(df["Amount"]==0)
```

Out[37]:
```
Counter({False: 2770393, True: 16})
```

In [38]:
```
Counter(df["Amount"]<0)
```

Out[38]:
```
Counter({False: 2770409})
```

Asuume that if there is a transaction amount is zero so the transaction is Fraudulent so remove the 0 amount from transaction

In [39]:
```
#remove 0 value amount
df=df.loc[df['Amount']>0,:]
```

In [40]:
```
df.head()
```

Out[40]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_F
0	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	
1	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	
2	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	
3	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	
4	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	

In [41]:
```
Counter(df['New_balance_Orig'])
```

Out[41]:
```
Counter({0.0: 2496640,
         16503.2: 1,
         162075.98: 1,
         25203.05: 1,
         17725.67: 1,
         783697.68: 1,
         610710.98: 1,
         30359.46: 1,
         167365.73: 1,
         137034.36: 1,
         214285.52: 1,
         1571970.16: 1,
         315983.22: 1,
         2955.68: 1,
         5861124.98: 1,
         9807.8: 1,
         98984.23: 1,
         2369160.89: 1,
         3522.52: 1,
         247294.52: 1,
         1891.79: 1,
         2895.06: 1,
         88926.2: 1,
         121552.02: 1,
         8933.73: 1,
         233940.02: 1,
         97055.04: 1,
         5292599.2: 1,
         5265590.36: 1,
         4934045.11: 1,
         4891515.83: 1,
         4639386.1: 1,
         4165916.16: 1,
         1848507.28: 1,
         984117.79: 1,
         708022.87: 1,
         575667.54: 2,
         6458842.26: 1,
         7239.33: 1,
         669906.01: 1,
         149735.97: 1,
         96641.81: 1,
         52920.71: 1,
         1574226.59: 1,
         808818.01: 1,
         151457.73: 1,
         814466.17: 1,
         3326.07: 1,
         1323331.91: 1,
         30040.03: 1,
         140757.42: 1,
         33316.54: 1,
         77710.3: 1,
         557537.26: 1,
         1593177.6: 1,
```

1517262.16: 1,
90540.7: 1,
15117.71: 1,
696313.01: 1,
595572.48: 1,
581406.1: 1,
446862.5: 1,
398655.68: 1,
115711.62: 1,
96441.48: 1,
90948.3: 1,
209548.6: 1,
2956.89: 1,
26136.14: 1,
120039.86: 1,
71856.76: 1,
206.12: 1,
141949.63: 1,
187096.37: 1,
381598.24: 1,
91191.64: 1,
19340.02: 1,
37068.76: 1,
131161.66: 1,
10779.79: 1,
11530.13: 1,
71287.2: 1,
14210.92: 1,
13629.57: 1,
5187.4: 1,
224060.15: 1,
80305.45: 1,
178616.24: 1,
120074.73: 1,
9719.72: 1,
21944.26: 1,
10119.47: 1,
20462.56: 1,
215080.02: 1,
9076.2: 1,
535641.97: 1,
501176.6: 1,
8109.5: 1,
105279.81: 1,
4408.53: 1,
957.65: 1,
58642.78: 1,
6292.72: 1,
38478.16: 1,
152729.43: 1,
2930418.44: 1,
26982.84: 1,
646875.4: 1,
24896.56: 1,
57250.21: 1,
16020.25: 1,
26488.54: 2,
50893.41: 1,
16567.87: 1,
98714.61: 1,
38799.34: 1,
90414.04: 1,
628791.48: 1,
21551.37: 1,
51110.31: 1,
80064.48: 1,
10883.6: 1,
295881.79: 1,
181733.94: 1,
256358.52: 1,
646196.83: 1,
78234.23: 1,
61578.57: 1,
4132.9: 1,
29908.64: 1,
9990.06: 1,
223361.33: 1,
96646.3: 1,
50071.11: 1,
60227.53: 1,
41304.43: 1,
89435.23: 1,
7776.39: 1,
295.65: 1,
95624.52: 1,
34196.38: 1,
785.74: 1,
399979.13: 1,
540.77: 1,

182490.89: 1,
28041.27: 1,
263368.87: 1,
97591.82: 1,
434626.99: 1,
24163.59: 1,
237443.71: 1,
96037.24: 1,
20247.55: 1,
13557.11: 1,
18276.03: 1,
82038.12: 1,
27626.25: 1,
75678.6: 1,
587.49: 1,
22159.27: 1,
17298.53: 1,
28973.52: 1,
4065.74: 1,
41196.46: 1,
103670.95: 1,
39276.47: 1,
4169.05: 1,
93907.01: 1,
29147.61: 1,
62486.16: 1,
45564.87: 1,
253513.04: 1,
541387.8: 1,
62105.69: 1,
346665.78: 1,
8907.97: 1,
47958.66: 1,
72277.19: 1,
17029.23: 1,
176718.81: 1,
720190.64: 1,
1563279.2: 1,
377558.57: 1,
34801.25: 1,
271309.17: 1,
4435.59: 1,
131574.31: 1,
22149.8: 1,
760639.94: 1,
842508.92: 1,
76899.35: 1,
405978.14: 1,
59824.59: 1,
54119.14: 1,
305.66: 1,
33850.04: 1,
281972.44: 1,
9689.31: 1,
38072.1: 1,
271493.37: 1,
6341.36: 1,
127891.14: 1,
279293.79: 1,
17481.83: 1,
424659.77: 1,
507810.53: 1,
35962.21: 1,
122290.08: 1,
1170985.59: 1,
347493.94: 1,
179508.98: 1,
16962.43: 1,
1418889.24: 1,
986412.38: 1,
1878.54: 1,
361344.83: 1,
122010.83: 1,
8883.06: 1,
59993.51: 1,
11318.68: 1,
167177.58: 1,
120428.33: 1,
1037040.13: 1,
6423.28: 1,
329687.96: 1,
148647.34: 1,
64228.45: 1,
104010.58: 1,
201194.1: 1,
244039.63: 1,
214606.43: 1,
763336.9: 1,
729533.72: 1,

32507.28: 1,
32564.05: 1,
461811.41: 1,
220182.09: 1,
18498.0: 1,
121888.7: 1,
153571.84: 1,
82986.13: 1,
71250.61: 1,
140168.02: 1,
297238.73: 1,
4099.81: 2,
8471.28: 1,
231080.84: 1,
36317.32: 1,
37827.13: 1,
8645.66: 1,
551952.83: 1,
77130.4: 1,
74855.48: 1,
69935.63: 1,
7590.52: 1,
1065053.21: 1,
909418.64: 1,
294081.02: 1,
253442.46: 1,
329088.84: 1,
813862.2: 1,
203973.13: 1,
998438.95: 1,
715414.31: 1,
14510.72: 1,
362219.88: 1,
628405.74: 1,
114559.37: 1,
100939.88: 1,
65591.04: 1,
445057.21: 1,
118662.54: 1,
140758.06: 1,
30699.68: 1,
103334.31: 1,
767112.01: 1,
1835890.27: 1,
1507508.16: 1,
1281922.04: 1,
1033426.33: 1,
707111.1: 1,
252910.65: 1,
62111.14: 1,
141866.28: 1,
82181.16: 1,
381400.76: 1,
2268.8: 1,
28969.4: 1,
844080.69: 1,
529175.59: 1,
53056.54: 1,
18889.01: 1,
14922.0: 1,
2235.96: 1,
15721.11: 1,
412517.18: 1,
120661.17: 1,
32583.34: 1,
15975.53: 1,
182113.96: 1,
170465.33: 1,
99972.49: 1,
132596.49: 1,
1523582.38: 1,
348916.63: 1,
316162.94: 1,
257069.12: 1,
46171.9: 1,
1179604.66: 1,
816713.91: 1,
711497.93: 1,
444635.45: 1,
226381.78: 1,
160128.05: 1,
11033.14: 1,
226638.12: 1,
10597.64: 1,
16522.69: 1,
149352.33: 1,
66002.55: 1,
7616.12: 1,
333440.8: 1,

82481.9: 1,
791885.97: 1,
228525.37: 1,
696178.08: 1,
12949.86: 1,
343151.23: 1,
598567.97: 1,
371278.54: 1,
53797.04: 1,
61242.51: 1,
303260.78: 1,
131642.17: 1,
61477.06: 1,
77299.67: 1,
38294.83: 1,
6113.17: 1,
37053.94: 1,
64519.25: 1,
321128.37: 1,
12452.77: 1,
294751.76: 1,
221363.79: 1,
126304.44: 1,
166995.99: 1,
157368.72: 1,
572413.6: 1,
88189.84: 1,
15040.97: 1,
26967.67: 2,
107416.01: 1,
460432.63: 1,
4624.21: 1,
175711.12: 1,
152350.11: 1,
234.04: 1,
15311.51: 1,
54773.81: 1,
32365.98: 1,
50204.82: 1,
13300.89: 1,
120049.44: 1,
29563.74: 1,
40260.79: 1,
718216.4: 1,
207742.79: 1,
6253.0: 1,
25726.89: 1,
55586.05: 1,
32753.99: 1,
82108.42: 1,
173863.3: 1,
69321.52: 1,
59863.35: 1,
631078.29: 1,
122034.94: 1,
899516.87: 1,
723387.06: 1,
143251.02: 1,
216327.12: 1,
29477.05: 1,
21799.62: 1,
84439.59: 1,
278727.56: 1,
26812.62: 1,
50947.36: 1,
311247.21: 1,
684904.15: 1,
142453.91: 1,
29383.41: 1,
46474.84: 1,
497347.42: 1,
253368.14: 1,
958036.5: 1,
6712.04: 1,
24254.4: 1,
55442.03: 1,
6469.52: 1,
88942.78: 1,
58608.85: 1,
102211.14: 1,
39974.29: 1,
62868.54: 1,
242719.04: 1,
67512.15: 1,
27006.75: 1,
32130.35: 1,
78548.44: 1,
38675.05: 1,
525965.63: 1,

143624.95: 1,
400936.89: 1,
209511.9: 1,
163125.76: 1,
179369.63: 1,
69447.6: 1,
204352.02: 1,
1166971.78: 1,
12727.44: 1,
29076.48: 1,
6950.06: 1,
8999.75: 1,
18626.64: 1,
376593.46: 1,
184913.41: 1,
649330.84: 1,
1561487.59: 1,
27625.83: 1,
572801.56: 1,
38416.28: 1,
19398.71: 1,
1426353.89: 1,
42806.82: 1,
38626.66: 1,
76279.4: 1,
5707.48: 1,
6176.8: 1,
431360.97: 1,
742110.2: 1,
45384.44: 1,
176310.21: 1,
125432.8: 1,
169794.82: 1,
922559.11: 1,
250285.43: 1,
226307.93: 1,
133956.73: 1,
11464.04: 1,
155536.38: 1,
353808.89: 1,
524288.37: 1,
297961.26: 1,
232156.07: 1,
654291.97: 1,
554885.05: 1,
56555.5: 1,
26640.69: 1,
390539.72: 1,
51930.59: 1,
194561.46: 1,
291834.84: 1,
34515.26: 1,
2588602.45: 1,
240117.67: 1,
52125.77: 1,
41884.88: 1,
1999481.09: 1,
1838358.98: 1,
1711619.6: 1,
1609792.69: 1,
1258036.35: 1,
2135962.15: 1,
2097981.84: 1,
360228.58: 1,
151947.25: 1,
204919.54: 1,
31639.91: 2,
36495.06: 1,
84945.39: 1,
122226.58: 1,
531923.44: 1,
98372.63: 1,
21743.09: 1,
30953.27: 1,
1274868.82: 1,
1090214.75: 1,
546067.07: 1,
491377.87: 1,
139703.48: 1,
36799.34: 1,
660757.97: 1,
343455.83: 1,
63911.41: 1,
1674.07: 1,
124220.23: 1,
200307.92: 1,
132162.26: 1,
71486.36: 1,
508209.44: 1,

404045.66: 1,
148046.18: 1,
258773.27: 1,
8997.03: 1,
136547.07: 1,
78825.35: 1,
46534.2: 1,
362255.94: 1,
87478.86: 1,
826065.48: 1,
290379.23: 1,
470369.35: 1,
728414.36: 1,
33051.23: 1,
174396.51: 1,
4651.85: 1,
11558.52: 1,
59489.03: 1,
21090.72: 1,
2236504.82: 1,
1276796.48: 1,
559471.73: 1,
308773.19: 1,
58790.6: 1,
1335228.79: 1,
456464.85: 1,
399808.23: 1,
206252.01: 1,
32479.83: 1,
72565.51: 1,
82747.45: 1,
1068434.79: 1,
558967.43: 1,
1929190.19: 1,
250872.05: 1,
554870.01: 1,
1301698.04: 1,
943458.1: 1,
70902.66: 1,
167560.3: 1,
34005.47: 1,
13351.11: 1,
528529.23: 1,
11033.64: 1,
85059.97: 1,
226869.64: 1,
64837.48: 1,
98090.88: 1,
62697.87: 1,
32455.97: 1,
9136.27: 1,
688804.35: 1,
434645.34: 1,
423106.51: 1,
35338.84: 1,
98558.1: 1,
82102.82: 1,
16309.26: 1,
359777.46: 1,
15021.17: 1,
106283.79: 1,
204360.31: 1,
99544.99: 1,
18715.52: 1,
127736.49: 1,
13585.04: 1,
1732193.03: 1,
1728907.45: 1,
1195256.81: 1,
1151647.9: 1,
887597.79: 1,
637185.74: 1,
389702.2: 1,
13589.59: 1,
240290.22: 1,
436235.72: 1,
97069.73: 1,
85152.97: 1,
32225.03: 1,
5031174.59: 1,
81038.6: 1,
24178.9: 1,
192885.34: 1,
190185.16: 1,
35094.83: 1,
17178.69: 1,
547998.97: 1,
153646.24: 1,
113476.09: 1,

30270.09: 1,
229085.69: 1,
73559.65: 1,
496485.18: 1,
1161103.63: 1,
318059.48: 1,
473967.51: 1,
434529.26: 1,
4200.85: 1,
288902.08: 1,
11790.79: 1,
17178.64: 1,
455907.36: 1,
211338.7: 1,
82577.67: 1,
96444.93: 1,
432683.36: 1,
128084.09: 1,
77306.77: 1,
1023860.79: 1,
223612.86: 1,
175143.96: 1,
839394.15: 1,
35033.13: 1,
95938.39: 1,
7826289.26: 1,
2143887.8: 1,
2004194.0: 1,
1770523.08: 1,
52810.96: 1,
621105.26: 1,
193511.14: 1,
78212.96: 1,
999447.8: 1,
452703.67: 1,
2397228.93: 1,
1581.61: 1,
205174.64: 1,
358742.56: 1,
22352.49: 1,
3096997.24: 1,
3068764.71: 1,
6785.72: 1,
2749.25: 1,
448541.93: 1,
98039.3: 1,
498128.38: 1,
47675.51: 1,
188073.98: 1,
18561.02: 1,
253549.38: 1,
361883.49: 1,
29645.73: 1,
14060.39: 1,
12107.78: 1,
182379.67: 1,
105327.41: 1,
138750.38: 1,
96909.58: 1,
108562.68: 1,
20094.3: 1,
147031.9: 1,
197805.44: 1,
41280.48: 1,
32827.87: 1,
711247.64: 1,
98248.77: 1,
14391.23: 1,
100355.9: 1,
690793.82: 1,
6991.58: 1,
14089.72: 2,
75243.91: 1,
23661.05: 2,
2618.0: 1,
3313.62: 1,
31673.89: 1,
11181.35: 1,
146120.75: 1,
3713204.65: 1,
73412.59: 1,
1549746.57: 1,
347074.49: 1,
1266088.88: 1,
8376.8: 1,
25973.27: 1,
394455.36: 1,
14542.28: 1,
221251.38: 1,

118831.88: 1,
11150736.44: 1,
17070.58: 1,
42816.53: 1,
59159.77: 1,
453317.14: 1,
24251.48: 1,
249300.69: 1,
41482.56: 1,
1334404.89: 1,
1276550.33: 1,
1185706.95: 1,
2482459.99: 1,
883352.28: 1,
143792.36: 1,
39514.82: 1,
30450.26: 1,
1335352.65: 1,
910251.45: 1,
407437.54: 1,
90190.4: 1,
486645.09: 1,
158811.27: 1,
28341.12: 2,
43877.6: 1,
89569.67: 1,
344507.33: 1,
23077.92: 1,
435486.29: 1,
229567.42: 1,
155777.95: 1,
2143898.13: 1,
2060643.32: 1,
20434.9: 1,
278675.26: 1,
1257854.82: 1,
105827.12: 1,
36599.65: 1,
2783183.21: 1,
2619956.07: 1,
2353609.96: 1,
147901.53: 1,
169830.4: 1,
442869.68: 1,
12927.79: 1,
450645.35: 1,
32074.9: 1,
87404.94: 1,
2765.16: 1,
138756.18: 1,
754836.66: 1,
21342.99: 1,
126118.17: 1,
1386428.21: 1,
214709.81: 1,
135489.41: 1,
40791.37: 1,
2373962.1: 1,
744203.53: 1,
52140.52: 1,
59471.29: 1,
45434.17: 1,
2204491.42: 1,
1147235.3: 1,
1015672.93: 1,
990338.21: 1,
652837.27: 1,
392308.03: 1,
343443.26: 1,
1154317.42: 1,
135331.98: 1,
60393.55: 1,
23417.44: 1,
144167.44: 1,
134955.37: 1,
601781.91: 1,
459971.12: 1,
73703.05: 1,
214023.59: 1,
30611.97: 1,
235242.49: 1,
1761993.9: 1,
1477544.9: 1,
1435408.11: 1,
969613.02: 1,
262835.22: 1,
1207165.99: 1,
7454.94: 1,
53632.9: 1,

850699.86: 1,
668811.71: 1,
668998.86: 1,
9400.91: 1,
3803124.27: 1,
1882771.18: 1,
9116.71: 1,
58705.81: 1,
93652.8: 1,
97990.73: 1,
4750.57: 1,
488300.86: 1,
193075.86: 1,
13482.32: 1,
51209.9: 1,
769.66: 1,
1928215.67: 1,
1860067.73: 1,
1766734.82: 1,
1526804.79: 1,
1403125.39: 1,
1227093.68: 1,
1214054.07: 1,
952728.57: 1,
809015.1: 1,
495514.43: 1,
376367.94: 1,
208905.45: 1,
179408.63: 1,
146419.66: 1,
114714.4: 1,
890293.35: 1,
405814.33: 1,
331061.49: 1,
76728.27: 1,
102511.66: 1,
18350.02: 1,
66358.47: 1,
45349.06: 1,
698388.0: 1,
297794.28: 1,
181071.43: 1,
56481.06: 1,
160692.15: 1,
22287.19: 1,
14261.93: 1,
871289.83: 1,
821183.86: 1,
570332.79: 1,
428117.11: 1,
292398.84: 1,
205907.71: 1,
29870.9: 1,
1211471.17: 1,
47819.69: 1,
86482.85: 1,
6918.57: 1,
1398.45: 1,
155195.61: 1,
140760.62: 1,
74606.37: 1,
114630.25: 1,
12272.14: 1,
1836186.73: 1,
152502.56: 1,
281373.23: 1,
298801.24: 1,
133763.49: 1,
10910.63: 1,
12124.69: 1,
764484.8: 1,
1165923.35: 1,
801428.57: 1,
535054.62: 1,
996509.73: 1,
678410.37: 1,
482465.71: 1,
251753.14: 1,
13796061.54: 1,
13673792.01: 1,
491.7: 1,
11787.23: 1,
69394.7: 1,
6338.8: 1,
153590.44: 1,
102855.52: 1,
320776.06: 1,
167038.38: 1,
1461059.36: 1,

360392.29: 1,
3345644.07: 1,
18685.58: 1,
1224252.54: 1,
750796.46: 1,
110182.15: 1,
89630.83: 1,
69979.06: 1,
16741.1: 1,
96549.18: 1,
583609.27: 1,
2427654.95: 1,
2394939.52: 1,
1298690.3: 1,
16978.93: 1,
192826.79: 1,
436654.5: 1,
68168.87: 1,
2892745.54: 1,
2528788.63: 1,
21630.92: 1,
597025.56: 1,
574090.22: 1,
38523.24: 1,
1045315.28: 1,
1069004.15: 1,
9304.41: 1,
93878.45: 1,
461830.62: 1,
60777.71: 1,
45729.37: 1,
8954.2: 1,
145928.83: 1,
77407.69: 1,
9895.71: 1,
220743.64: 1,
79777.53: 1,
395564.39: 1,
12864.44: 1,
347263.76: 1,
177475.02: 1,
635355.44: 1,
9594.65: 1,
807542.59: 1,
14579.53: 1,
1093323.8: 1,
151977.89: 1,
85234.9: 1,
60462.35: 1,
312080.68: 1,
88575.14: 1,
49331.48: 1,
853587.78: 1,
178269.04: 1,
679955.66: 1,
1147.24: 1,
57790.73: 1,
18641.96: 1,
85951.21: 1,
2161358.35: 1,
331220.64: 1,
3298.25: 1,
148564.66: 1,
131020.77: 1,
449250.7: 1,
167237.01: 1,
19173.1: 1,
24139.68: 1,
2330025.08: 1,
21869.74: 1,
9594.47: 1,
38240.9: 1,
376471.99: 1,
561663.06: 1,
1862636.18: 1,
1537008.05: 1,
1387163.2: 1,
1367179.83: 1,
933650.27: 1,
385529.13: 1,
380361.77: 1,
252826.08: 1,
74408.86: 1,
858968.72: 1,
12147.43: 1,
172956.95: 1,
129956.74: 1,
129385.32: 1,
99908.25: 1,

```
            129363.32: 1,
            1668305.23: 1,
            1609185.35: 1,
            1601670.97: 1,
            1332071.67: 1,
            1172298.77: 1,
            831178.91: 1,
            740061.59: 1,
            221344.45: 1,
            482610.94: 1,
            51157.58: 1,
            2086207.02: 1,
            409.47: 1,
            770872.35: 1,
            102070.77: 1,
            42869.59: 1,
            60008.26: 1,
            23501.89: 1,
            81415.93: 1,
            551358.17: 1,
            2786271.9: 1,
            40946.26: 1,
            44596.57: 1,
            452034.32: 1,
            170309.52: 1,
            994391.7: 1,
            4401631.66: 1,
            4173293.65: 1,
            71717.16: 1,
            41358.44: 1,
            9268.88: 1,
            750353.32: 1,
            5909.44: 1,
            80606.79: 1,
            177166.04: 1,
            28265.23: 1,
            98931.71: 1,
            76120.37: 1,
            50310.89: 1,
            24633.08: 1,
            25057.11: 1,
            20241.42: 1,
            3548553.54: 1,
            125987.15: 1,
            8410.73: 1,
            4463342.32: 1,
            137988.93: 1,
            133554.02: 1,
            125617.79: 1,
            33415.41: 1,
            36259.88: 1,
            8864.63: 1,
            21864.67: 1,
            290793.95: 1,
            248032.8: 1,
            ...})
```

In [42]:
```
Counter(df['Amount']==0)
```

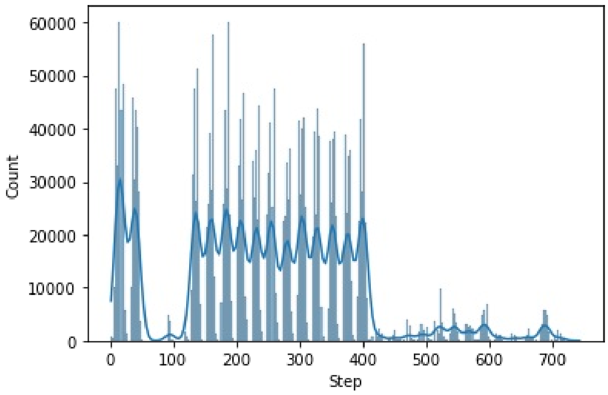
Out[42]:
```
Counter({False: 2770393})
```

In [43]:
```
df.columns
```

Out[43]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```

In [44]:
```
sns.histplot(data=df,x='Step',kde=True)
```

Out[44]:
```
<AxesSubplot:xlabel='Step', ylabel='Count'>
```



In [45]:
```python
df.columns
```

Out[45]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```

In [46]:
```python
df.drop('Is_Flagged_Fraud',axis=1)
```

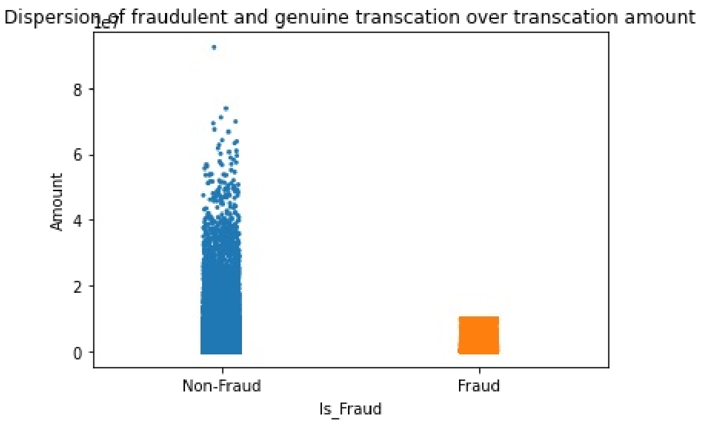
Out[46]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_De
0	1	TRANSFER	181.00	C1305486145	181.00	0.0	C553264065	0.00	0.0
1	1	CASH_OUT	181.00	C840083671	181.00	0.0	C38997010	21182.00	0.0
2	1	CASH_OUT	229133.94	C905080434	15325.00	0.0	C476402209	5083.00	51513.4
3	1	TRANSFER	215310.30	C1670993182	705.00	0.0	C1100439041	22425.00	0.0
4	1	TRANSFER	311685.89	C1984094095	10835.00	0.0	C932583850	6267.00	2719172.8
...	...	...	...	...	...	...	...	...	...
2770404	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.1
2770405	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.0
2770406	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.1
2770407	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.0
2770408	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.6

2770393 rows × 10 columns

For the long data we are using the Stripplot for the distribution

In [47]:
```python
sns.stripplot(x='Is_Fraud',y='Amount',data=df,size=3,jitter=0.07,dodge=True)
plt.title("Dispersion of fraudulent and genuine transcation over transcation amount")
plt.xticks([0,1],['Non-Fraud','Fraud'])
plt.show()
```



BALANCES

Originator's balance and recipient's balance

In [48]:
```python
df.columns
```

Out[48]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```

In [49]:
```python
#Zero balance check
100*df[df["Is_Fraud"]==0].Old_balance_Org.value_counts()/len(df[df["Is_Fraud"]==0].Old_balance_Org)
```

Out[49]:
```
0.00          47.373213
154.00         0.015712
124.00         0.015459
109.00         0.015386
186.00         0.015386
                 ...
148855.26      0.000036
220611.00      0.000036
38401.79       0.000036
599002.00      0.000036
168046.00      0.000036
Name: Old_balance_Org, Length: 431693, dtype: float64
```

In [50]:
```
#Zero balance check
100*df[df["Is_Fraud"]==0].New_balance_Dest.value_counts()/len(df[df["Is_Fraud"]==0].New_balance_Dest)
```

Out[50]:
```
0.00           0.452828
16532032.16    0.000796
19169204.93    0.000760
4743010.67     0.000652
16408480.12    0.000579
                 ...
16395701.39    0.000036
344337.62      0.000036
3557827.05     0.000036
44049.67       0.000036
82096.45       0.000036
Name: New_balance_Dest, Length: 2559669, dtype: float64
```

In [51]:
```
#incorrect balance check
100*df[df["Is_Fraud"]==0].New_balance_Orig.value_counts()/len(df[df["Is_Fraud"]==0].New_balance_Orig)
```

Out[51]:
```
0.00          90.095091
2305.53        0.000109
14403.77       0.000109
26284.34       0.000109
174.94         0.000109
                 ...
82659.59       0.000036
5535.97        0.000036
377214.15      0.000036
16004.54       0.000036
6141.97        0.000036
Name: New_balance_Orig, Length: 271833, dtype: float64
```

In [52]:
```
100*df[df["Is_Fraud"]==0].Old_balance_Dest.value_counts()/len(df[df["Is_Fraud"]==0].Old_balance_Dest)
```

Out[52]:
```
0.00           13.900860
10000000.00     0.021794
20000000.00     0.007928
30000000.00     0.003113
40000000.00     0.001122
                  ...
628398.40       0.000036
9173129.25      0.000036
792166.89       0.000036
4888662.39      0.000036
24893.67        0.000036
Name: Old_balance_Dest, Length: 2358040, dtype: float64
```

In [53]:
```
#comparision the Fraud and Non Fraud Transaction where originator's initial balance Zero
#
100*df[df['Is_Fraud']==1].Old_balance_Org.value_counts()/len(df[df['Is_Fraud']==1].Old_balance_Org)
```

Out[53]:
```
10000000.00    1.732341
0.00           0.304990
1165187.89     0.048798
429257.45      0.048798
181.00         0.024399
                 ...
19110884.44    0.012200
29110884.44    0.012200
4892193.09     0.012200
14892193.09    0.012200
12740879.15    0.012200
Name: Old_balance_Org, Length: 4094, dtype: float64
```

In [54]:
```
print('% of fraudulent transcation where initial balance of originator is 0: 0.30%')
```

```
% of fraudulent transcation where initial balance of originator is 0: 0.30%
```

In [55]:
```
#Non_Fraud
100*df[df['Is_Fraud']==0].Old_balance_Org.value_counts()/len(df[df['Is_Fraud']==0].Old_balance_Org)
```

Out[55]:
```
0.00          47.373213
154.00         0.015712
124.00         0.015459
109.00         0.015386
186.00         0.015386
                 ...
148855.26      0.000036
220611.00      0.000036
38401.79       0.000036
599002.00      0.000036
168046.00      0.000036
Name: Old_balance_Org, Length: 431693, dtype: float64
```

In [56]:
```
print('% of non-fraud transcation where initial balance of originator is 0: 47.37%')
```

```
% of non-fraud transcation where initial balance of originator is 0: 47.37%
```

In [57]:
```
df.columns
```

Out[57]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud'],
      dtype='object')
```

In [58]:
```
#inaccuracies in originator and recipient balance
df["OrigBalance_inacc"]=(df[ 'Old_balance_Org']-df['Amount'])-df['New_balance_Orig']
```

In [59]:
```
df.head()
```

Out[59]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_F
0	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	
1	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	
2	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	
3	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	
4	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	

In [60]:
```
df["DestBalance_inacc"]=(df[ 'Old_balance_Dest']+df['Amount'])-df['New_balance_Dest']
```

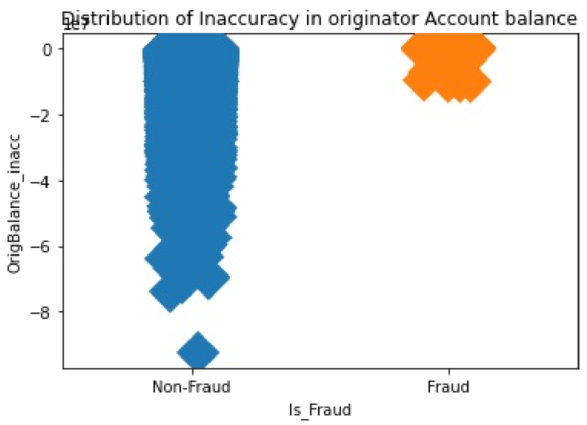
In [61]:
```
 df.head()
```

Out[61]:

	Step	Type	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_F
0	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	
1	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	
2	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	
3	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	
4	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	

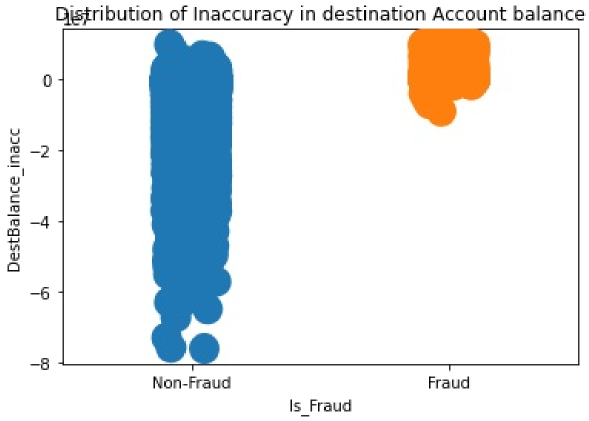
In [62]:
```
#Distribution of Inaccuracy in originator Account balance
sns.stripplot(x='Is_Fraud',y='OrigBalance_inacc',data=df, size=20, marker="D",
                       edgecolor="gray")
plt.xticks([0,1],['Non-Fraud','Fraud'])
plt.title('Distribution of Inaccuracy in originator Account balance')
```

Out[62]:
```
Text(0.5, 1.0, 'Distribution of Inaccuracy in originator Account balance')
```



In [63]:
```
#Distribution of Inaccuracy in destination Account balance
sns.stripplot(x='Is_Fraud',y='DestBalance_inacc',data=df, size=20,
                       edgecolor="gray")
plt.xticks([0,1],['Non-Fraud','Fraud'])
plt.title('Distribution of Inaccuracy in destination Account balance')
```

Out[63]: Text(0.5, 1.0, 'Distribution of Inaccuracy in destination Account balance')

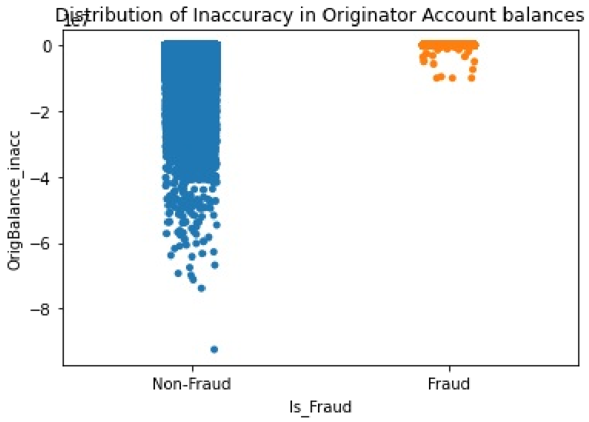


In [64]:
```python
df.columns
```

Out[64]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud', 'OrigBalance_inacc',
       'DestBalance_inacc'],
      dtype='object')
```

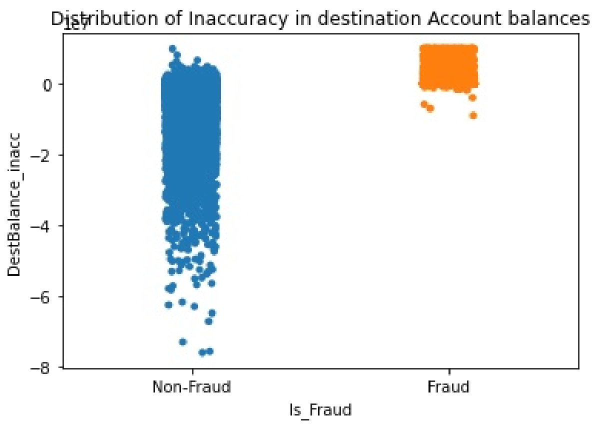
In [65]:
```python
sns.stripplot(x='Is_Fraud',y='OrigBalance_inacc',data=df)
plt.xticks([0,1],['Non-Fraud','Fraud'])
plt.title('Distribution of Inaccuracy in Originator Account balances')
```

Out[65]: Text(0.5, 1.0, 'Distribution of Inaccuracy in Originator Account balances')



In [66]:
```python
sns.stripplot(x='Is_Fraud',y='DestBalance_inacc',data=df)
plt.xticks([0,1],['Non-Fraud','Fraud'])
plt.title('Distribution of Inaccuracy in destination Account balances')
```

Out[66]: Text(0.5, 1.0, 'Distribution of Inaccuracy in destination Account balances')



Predictive Modelling for Fraud Detection

In [67]:
```python
df.columns
```

Out[67]:
```
Index(['Step', 'Type', 'Amount', 'Name_Orig', 'Old_balance_Org',
       'New_balance_Orig', 'Name_dest', 'Old_balance_Dest', 'New_balance_Dest',
       'Is_Fraud', 'Is_Flagged_Fraud', 'OrigBalance_inacc',
       'DestBalance_inacc'],
      dtype='object')
```

The name(or_Id) columns is not needed for the classification so remove them

In [68]:
```
df.drop(['Name_Orig','Name_dest'],axis=1)
```

Out[68]:

	Step	Type	Amount	Old_balance_Org	New_balance_Orig	Old_balance_Dest	New_balance_Dest	Is_Fraud	Is_Flagged_Frau
0	1	TRANSFER	181.00	181.00	0.0	0.00	0.00	1	
1	1	CASH_OUT	181.00	181.00	0.0	21182.00	0.00	1	
2	1	CASH_OUT	229133.94	15325.00	0.0	5083.00	51513.44	0	
3	1	TRANSFER	215310.30	705.00	0.0	22425.00	0.00	0	
4	1	TRANSFER	311685.89	10835.00	0.0	6267.00	2719172.89	0	
...	...	...	...	...	...	...	...	...	
2770404	743	CASH_OUT	339682.13	339682.13	0.0	0.00	339682.13	1	
2770405	743	TRANSFER	6311409.28	6311409.28	0.0	0.00	0.00	1	
2770406	743	CASH_OUT	6311409.28	6311409.28	0.0	68488.84	6379898.11	1	
2770407	743	TRANSFER	850002.52	850002.52	0.0	0.00	0.00	1	
2770408	743	CASH_OUT	850002.52	850002.52	0.0	6510099.11	7360101.63	1	

2770393 rows × 11 columns

we have one categorical variable in the dataset-the Type this feature needs to be encoded as binary variable

Encoding

In [69]:
```
df=pd.get_dummies(df,columns=['Type'],prefix=['Type'])
```

In [70]:
```
df.head()
```

Out[70]:

	Step	Amount	Name_Orig	Old_balance_Org	New_balance_Orig	Name_dest	Old_balance_Dest	New_balance_Dest	Is_Fraud	Is_Flag
0	1	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	1	
1	1	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	1	
2	1	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	0	
3	1	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	0	
4	1	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	0	

Split The Data

In [71]:
```
X=df.drop(['Is_Fraud','Name_dest','Name_Orig'],axis=1)
y=df['Is_Fraud']
```

In [72]:
```
X
```

Out[72]:

	Step	Amount	Old_balance_Org	New_balance_Orig	Old_balance_Dest	New_balance_Dest	Is_Flagged_Fraud	OrigBalance_inacc
0	1	181.00	181.00	0.0	0.00	0.00	0	0.00
1	1	181.00	181.00	0.0	21182.00	0.00	0	0.00
2	1	229133.94	15325.00	0.0	5083.00	51513.44	0	-213808.94
3	1	215310.30	705.00	0.0	22425.00	0.00	0	-214605.30
4	1	311685.89	10835.00	0.0	6267.00	2719172.89	0	-300850.89
...	...	...	...	...	...	...	...	...
2770404	743	339682.13	339682.13	0.0	0.00	339682.13	0	0.00
2770405	743	6311409.28	6311409.28	0.0	0.00	0.00	0	0.00
2770406	743	6311409.28	6311409.28	0.0	68488.84	6379898.11	0	0.00
2770407	743	850002.52	850002.52	0.0	0.00	0.00	0	0.00
2770408	743	850002.52	850002.52	0.0	6510099.11	7360101.63	0	0.00

2770393 rows × 11 columns

In [73]:
```
y
```

Out[73]:
```
0          1
1          1
2          0
3          0
4          0
          ..
2770404    1
2770405    1
2770406    1
2770407    1
2770408    1
Name: Is_Fraud, Length: 2770393, dtype: int32
```

Standardizing the data

In [74]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [75]:
```
X=pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
```

In [76]:
```
X
```

Out[76]:

	Step	Amount	Old_balance_Org	New_balance_Orig	Old_balance_Dest	New_balance_Dest	Is_Flagged_Fraud	OrigBalance_ina
0	-1.701817	-0.357468	-0.188848	-0.106389	-0.403155	-0.438260	-0.002403	0.32672
1	-1.701817	-0.357468	-0.188848	-0.106389	-0.398142	-0.438260	-0.002403	0.32672
2	-1.701817	-0.099577	-0.128591	-0.106389	-0.401952	-0.427246	-0.002403	0.08245
3	-1.701817	-0.115148	-0.186763	-0.106389	-0.397848	-0.438260	-0.002403	0.08154
4	-1.701817	-0.006592	-0.146457	-0.106389	-0.401672	0.143133	-0.002403	-0.01698
...	...	...	...	...	...	...	...	
2770388	3.537664	0.024943	1.161993	-0.106389	-0.403155	-0.365632	-0.002403	0.32672
2770389	3.537664	6.751439	24.922894	-0.106389	-0.403155	-0.438260	-0.002403	0.32672
2770390	3.537664	6.751439	24.922894	-0.106389	-0.386947	0.925841	-0.002403	0.32672
2770391	3.537664	0.599763	3.192506	-0.106389	-0.403155	-0.438260	-0.002403	0.32672
2770392	3.537664	0.599763	3.192506	-0.106389	1.137493	1.135420	-0.002403	0.32672

2770393 rows × 11 columns

In [77]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [78]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

In [79]:
```
models=[ LogisticRegression(),SVC(),DecisionTreeClassifier(),MLPClassifier(),RandomForestClassifier()]
```

In [80]:
```
svc=SVC()
```

In [81]:
```
svc.fit(X_train,y_train)
```
Out[81]:
```
SVC()
```
In [82]:
```
y_pred=svc.predict(X_test)
```
In [87]:
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import confusion_matrix,classification_report
```
In [88]:
```
print("accuracy score of the SVC model is:",accuracy_score(y_test,y_pred))
```
```
accuracy score of the SVC model is: 0.9987558926650608
```
In [89]:
```
print("Classification report :",classification_report(y_test,y_pred))
```
```
Classification report :               precision    recall  f1-score   support

           0       1.00      1.00      1.00    828636
           1       0.99      0.59      0.74      2482

    accuracy                           1.00    831118
   macro avg       0.99      0.80      0.87    831118
weighted avg       1.00      1.00      1.00    831118
```
In [90]:
```
lr=LogisticRegression()
```
In [91]:
```
lr.fit(X_train,y_train)
```
Out[91]:
```
LogisticRegression()
```
In [92]:
```
y_pred=lr.predict(X_test)
```
In [93]:
```
print("accuracy score of the LogisticRegression model is:",accuracy_score(y_test,y_pred))
```
```
accuracy score of the LogisticRegression model is: 0.9983552275368841
```
In [102…
```
print("Classification report is:",classification_report(y_test,y_pred))
```
```
Classification report is:               precision    recall  f1-score   support

           0       1.00      1.00      1.00    828636
           1       0.91      0.50      0.64      2482

    accuracy                           1.00    831118
   macro avg       0.96      0.75      0.82    831118
weighted avg       1.00      1.00      1.00    831118
```
In [96]:
```
d=DecisionTreeClassifier()
```
In [97]:
```
d.fit(X_train,y_train)
```
Out[97]:
```
DecisionTreeClassifier()
```
In [98]:
```
y_pred=lr.predict(X_test)
```
In [99]:
```
print("accuracy score of the DecisionTreeClassifier model is:",accuracy_score(y_test,y_pred))
```
```
accuracy score of the DecisionTreeClassifier model is: 0.9983552275368841
```
In [101…
```
print("Classification report :",classification_report(y_test,y_pred))
```
```
Classification report :               precision    recall  f1-score   support

           0       1.00      1.00      1.00    828636
           1       0.91      0.50      0.64      2482

    accuracy                           1.00    831118
   macro avg       0.96      0.75      0.82    831118
weighted avg       1.00      1.00      1.00    831118
```

5]What are the key factors that predict fraudulent customer?

Ans:The Original balance inaccuracy and the Destination balance accuracy is the key factor that predict fraudulent customer.

6] Do these factors make sense? If yes, How? If not, How not?

Ans:1]Yes, The inaccuracy is the difference between what the balance should be accounting for the transaction amount and what it is recorded as balance.

2] Inaccuracy in destination balance is likely to be a negative in case of geninue but positive in case of fraud transaction

7]What kind of prevention should be adopted while company update its infrastructure?

Ans:1]Must have a integrating emerging technologies in their systems. 2]Banks can adopt is to screen public records of the applicants, thereby ensuring their credit worthiness. 3]Analysis of financial patterns of the entity or individual. 4]Banks should leverage on the advancements in AI & ML technology, to preemptively analyze patterns and learn from historical cases.

8]Assuming these actions have been implemented, how would you determine if they work?

Ans:1]Machine learning (ML) is the science of creating and applying algorithms that are capable of learning from the past. Machine learning finds a perfect use case in fraud detection. Machine learning algorithms learn to tell fraudulent operations from legitimate ones without raising the suspicions of those executing the transactions. Machine learning can fight financial fraud by using big data better and faster than humans ever will be able to. 2]Once documents have been verified to be authentic and original, we move on to the next step in fraud prevention, due diligence. Due diligence covers a lot of aspects that need to be considered before extending loans to individuals or business entities. 3]Tax filings, be it ITR or GST filings, are a good indicator of the business health and validity of an entity. Lack of GST or ITR data is cause for concern for any lending institution, as it can be an indicator of fraudulent intention or activities.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js