In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbn
from sklearn.model_selection import train_test_split

data= pd.read_csv("PS_20174392719_1491204439457_log.csv")
```

In [2]:
```
data
```

Out[2]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873

6362620 rows × 11 columns

In [3]:
```
data.head(10)
```

Out[3]:

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

In [4]:
```
data.columns
```

Out[4]:
```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceO
rig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

In [5]:
```
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
```

In [6]:
```
data['step'].unique()
```

Out[6]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
       170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
       183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
       196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
       235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247,
       248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260,
       261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273,
       274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286,
       287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299,
       300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312,
       313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325,
       326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338,
       339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351,
       352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364,
       365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377,
       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
       391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
       404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416,
       417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
       430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
       443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
       456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468,
       469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
       482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494,
       495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507,
       508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
       521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
       534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
       547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559,
       560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,
       573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585,
       586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598,
       599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611,
       612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624,
       625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637,
       638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650,
       651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663,
       664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676,
       677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,
       690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702,
       703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715,
       716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728,
       729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741,
       742, 743], dtype=int64)
```

In [7]:
```
data.isnull()
```

Out[7]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbala
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
6362615	False	False	False	False	False	False	False	
6362616	False	False	False	False	False	False	False	
6362617	False	False	False	False	False	False	False	
6362618	False	False	False	False	False	False	False	
6362619	False	False	False	False	False	False	False	

6362620 rows × 11 columns

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
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 [9]:
```
data.shape
```

Out[9]:
```
(6362620, 11)
```

In [10]:
```
data['type'].unique()
```

Out[10]:
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

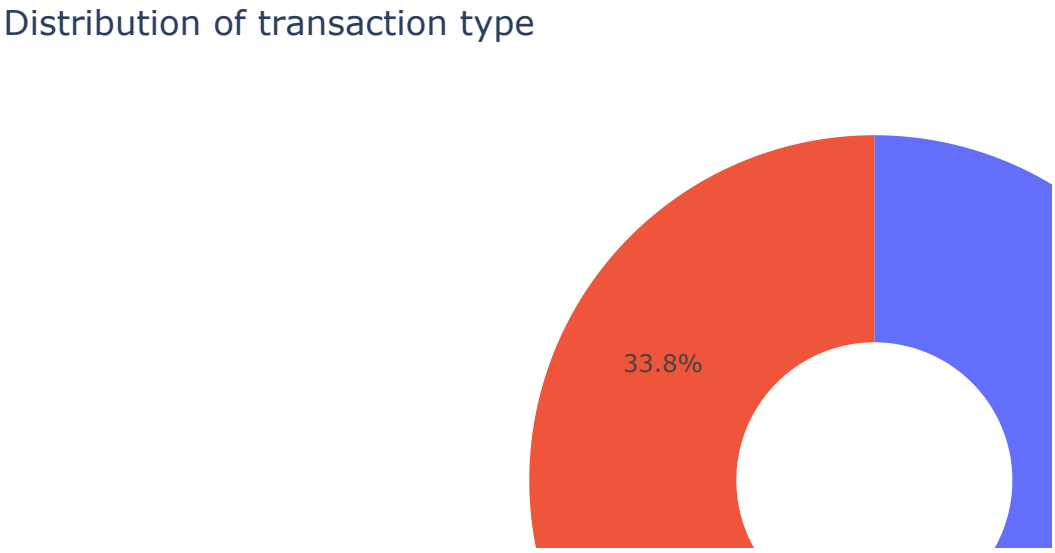
In [11]:
```
type=data['type'].value_counts()
```

In [12]:
```
transaction=type.index
```

In [13]:
```
quantity=type.values
```

In [14]:
```
import plotly.express as px
```

In [15]:
```
px.pie(data, values= quantity, names=transaction, hole=0.4, title="Distribu
```



In [16]:
```
data
```

Out[16]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873

6362620 rows × 11 columns

In [17]:
```
data.replace(to_replace=['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_
```

In [18]:
```
type
```

Out[18]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [19]:
```
data
```

Out[19]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225
2	1	4	181.00	C1305486145	181.00	0.00	C553264065
3	1	1	181.00	C840083671	181.00	0.00	C38997010
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703
...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189

6362620 rows × 11 columns

In [20]:
```
data["isFraud"]=data["isFraud"].map({0:'No Fraud', 1:'Fraud'})
```

In [21]:
```
data
```

Out[21]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225
2	1	4	181.00	C1305486145	181.00	0.00	C553264065
3	1	1	181.00	C840083671	181.00	0.00	C38997010
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703
...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189

6362620 rows × 11 columns

In [22]:
```
x= data[['type','amount','oldbalanceOrg','newbalanceOrig']]
```

In [23]:
```
y= data.iloc[:,-2]
```

In [24]:
```
y
```

Out[24]:
```
0          No Fraud
1          No Fraud
2             Fraud
3             Fraud
4          No Fraud
              ...
6362615       Fraud
6362616       Fraud
6362617       Fraud
6362618       Fraud
6362619       Fraud
Name: isFraud, Length: 6362620, dtype: object
```

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [26]:
```
model= DecisionTreeClassifier()
```

In [27]:
```
xtrain, xtest, ytrain, ytest= train_test_split(x,y, test_size=0.2,random_st
```

In [28]:
```
model.fit(xtrain, ytrain)
```

Out[28]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [29]:
```
model.score(xtest, ytest)
```

Out[29]:
```
0.9997005950378932
```

In [30]:
```
x
```

Out[30]:

	type	amount	oldbalanceOrg	newbalanceOrig
0	2	9839.64	170136.00	160296.36
1	2	1864.28	21249.00	19384.72
2	4	181.00	181.00	0.00
3	1	181.00	181.00	0.00
4	2	11668.14	41554.00	29885.86
...	...	...	...	...
6362615	1	339682.13	339682.13	0.00
6362616	4	6311409.28	6311409.28	0.00
6362617	1	6311409.28	6311409.28	0.00
6362618	4	850002.52	850002.52	0.00
6362619	1	850002.52	850002.52	0.00

6362620 rows × 4 columns

In [31]:
```
model.predict([[2,9800,170136,160296]])
```

```
C:\Users\Prasad\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarni
ng:

X does not have valid feature names, but DecisionTreeClassifier was fitted
with feature names
```

Out[31]:
```
array(['No Fraud'], dtype=object)
```

In [32]:
```
x
```

Out[32]:

	type	amount	oldbalanceOrg	newbalanceOrig
0	2	9839.64	170136.00	160296.36
1	2	1864.28	21249.00	19384.72
2	4	181.00	181.00	0.00
3	1	181.00	181.00	0.00
4	2	11668.14	41554.00	29885.86
...	...	...	...	...
6362615	1	339682.13	339682.13	0.00
6362616	4	6311409.28	6311409.28	0.00
6362617	1	6311409.28	6311409.28	0.00
6362618	4	850002.52	850002.52	0.00
6362619	1	850002.52	850002.52	0.00

6362620 rows × 4 columns

In [ ]: