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

In [2]:
```
data=pd.read_csv('credit card.csv')
```

In [3]:
```
data
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	i
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	

6362620 rows × 11 columns

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
step            0
type            0
amount          0
nameOrig        0
oldbalanceOrg   0
newbalanceOrig  0
nameDest        0
oldbalanceDest  0
newbalanceDest  0
isFraud         0
isFlaggedFraud  0
dtype: int64
```

So this dataset does not have any null values. Before moving forward, now, let's have a look at the type of transaction mentioned in the dataset:

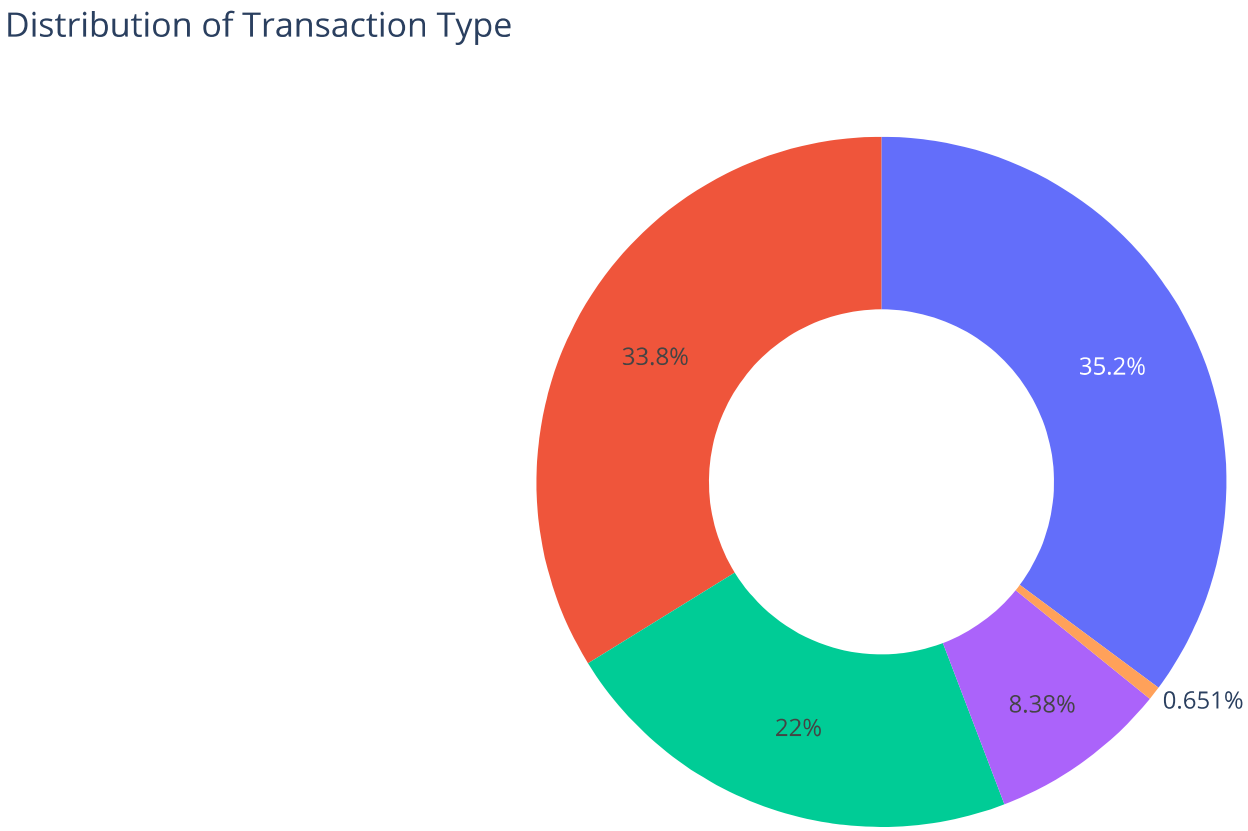
In [5]:
```
data['type']=np.array(data['type'])
```

In [6]:
```
data.type.value_counts()
```

Out[6]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [7]:
```
type = data["type"].value_counts()
transactions = type.index
quantity = type.values
```

In [8]:
```
import plotly.express as px
figure = px.pie(data,
             values=quantity,
             names=transactions,hole = 0.5,
             title="Distribution of Transaction Type")
figure.show()
```



Now let's have a look at the correlation between the features of the data with the isFraud column:

In [9]:
```
correlation = data.corr()
```

In [10]:
```
correlation["isFraud"].sort_values(ascending=False)
```

Out[10]:
```
isFraud           1.000000
amount            0.076688
isFlaggedFraud    0.044109
step              0.031578
oldbalanceOrg     0.010154
newbalanceDest    0.000535
oldbalanceDest   -0.005885
newbalanceOrig   -0.008148
Name: isFraud, dtype: float64
```

Now let's transform the categorical features into numerical. Here I will also transform the values of the isFraud column into No Fraud and Fraud labels to have a better understanding of the output:

In [11]:
```
data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2,
                                 "CASH_IN": 3, "TRANSFER": 4,
                                 "DEBIT": 5})
```

In [12]:
```
data["isFraud"] = data["isFraud"].map({0: "No Fraud", 1: "Fraud"})
```

In [13]:
```
data
```

Out[13]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	No Fraud
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	No Fraud
2	1	4	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	Fraud
3	1	1	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	Fraud
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	No Fraud
...	...	...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	Fraud
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	Fraud
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	Fraud
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	Fraud
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	Fraud

6362620 rows × 11 columns

In [14]:
```
data['isFraud'].value_counts()
```

Out[14]:
```
No Fraud    6354407
Fraud          8213
Name: isFraud, dtype: int64
```

Now let's train a classification model to classify fraud and non-fraud transactions. Before training the model, I will split the data into training and test sets:

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
x = np.array(data[["type", "amount", "oldbalanceOrg", "newbalanceOrig"]])
```

In [17]:
```
y = np.array(data[["isFraud"]])
```

Now let's train the online payments fraud detection model:

In [18]:
```
# training a machine learning model
from sklearn.tree import DecisionTreeClassifier
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.10, random_state=42)
model = DecisionTreeClassifier()
```

In [19]:
```
model.fit(xtrain, ytrain)
```

Out[19]:
```
DecisionTreeClassifier()
```

In [20]:
```
print(model.score(xtest, ytest))
```

```
0.9997343861491021
```

Now let’s classify whether a transaction is a fraud or not by feeding about a transaction into the model:

In [21]:
```
# prediction
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
features = np.array([[4, 9000.60, 9000.60, 0.0]])
print(model.predict(features))
```

```
['Fraud']
```

In [22]:
```
# prediction
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
features = np.array([[2, 20000.60, 29000.60, 9000.0]])
print(model.predict(features))
```

```
['No Fraud']
```