In [16]:
```
# Data Analysis & Wrangling

import pandas as pd
import numpy as np
import random as rnd

# Visualization

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [17]:
```
# Read The Dataset

CreditCard = pd.read_csv("E:/Data Sets/creditcard.csv")
print(CreditCard)
```

```
            Time         V1         V2        V3        V4        V5  \
0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018
2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
...          ...        ...        ...       ...       ...       ...
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

              V6        V7        V8        V9  ...       V21       V22  \
0       0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838
1      -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672
2       1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679
3       1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274
4       0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278
...          ...       ...       ...       ...  ...       ...       ...
284802 -2.606837 -4.918215  7.305334  1.914428  ...  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  ...  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  ...  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  ...  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
...          ...       ...       ...       ...       ...       ...     ...
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
```

```
1          0
2          0
3          0
4          0
...      ...
284802     0
284803     0
284804     0
284805     0
284806     0

[284807 rows x 31 columns]
```

In [18]:
```
# Viewing The Data First 10 Rows From The CreditCard Dataset

CreditCard.head(10)
```

Out[18]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.06
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.33
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.68
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.17
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.14
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.37
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.78
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.64
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.01
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.38

10 rows × 31 columns

In [13]:
```
# Viewing The Data Last 20 Rows From The CreditCard Dataset

CreditCard.tail(20)
```

Out[13]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
284787	172769.0	-1.029719	-1.110670	-0.636179	-0.840816	2.424360	-2.956733	0.283610	-0.332656	-0.247488	...	0.353722	0.488487	0.29
284788	172770.0	2.007418	-0.280235	-0.208113	0.335261	-0.715798	-0.751373	-0.458972	-0.140140	0.959971	...	-0.208260	-0.430347	0.41
284789	172770.0	-0.446951	1.302212	-0.168583	0.981577	0.578957	-0.605641	1.253430	-1.042610	-0.417116	...	0.851800	0.305268	-0.14
284790	172771.0	-0.515513	0.971950	-1.014580	-0.677037	0.912430	-0.316187	0.396137	0.532364	-0.224606	...	-0.280302	-0.849919	0.30
284791	172774.0	-0.863506	0.874701	0.420358	-0.530365	0.356561	-1.046238	0.757051	0.230473	-0.506856	...	-0.108846	-0.480820	-0.07
284792	172774.0	-0.724123	1.485216	-1.132218	-0.607190	0.709499	-0.482638	0.548393	0.343003	-0.226323	...	0.414621	1.307511	-0.05
284793	172775.0	1.971002	-0.699067	-1.697541	-0.617643	1.718797	3.911336	-1.259306	1.056209	1.315006	...	0.188758	0.694418	0.16
284794	172777.0	-1.266580	-0.400461	0.956221	-0.723919	1.531993	-1.788600	0.314741	0.004704	0.013857	...	-0.157831	-0.883365	0.08
284795	172778.0	-12.516732	10.187818	-8.476671	-2.510473	-4.586669	-1.394465	-3.632516	5.498583	4.893089	...	-0.944759	-1.565026	0.89
284796	172780.0	1.884849	-0.143540	-0.999943	1.506772	-0.035300	-0.613638	0.190241	-0.249058	0.666458	...	0.144008	0.634646	-0.04
284797	172782.0	-0.241923	0.712247	0.399806	-0.463406	0.244531	-1.343668	0.929369	-0.206210	0.106234	...	-0.228876	-0.514376	0.27
284798	172782.0	0.219529	0.881246	-0.635891	0.960928	-0.152971	-1.014307	0.427126	0.121340	-0.285670	...	0.099936	0.337120	0.25
284799	172783.0	-1.775135	-0.004235	1.189786	0.331096	1.196063	5.519980	-1.518185	2.080825	1.159498	...	0.103302	0.654850	-0.34
284800	172784.0	2.039560	-0.175233	-1.196825	0.234580	-0.008713	-0.726571	0.017050	-0.118228	0.435402	...	-0.268048	-0.717211	0.29
284801	172785.0	0.120316	0.931005	-0.546012	-0.745097	1.130314	-0.235973	0.812722	0.115093	-0.204064	...	-0.314205	-0.808520	0.05
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.01
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.01
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.03
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.16
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.37

20 rows × 31 columns

In [19]:
```
CreditCard.columns
```

Out[19]:
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

In [21]:
```
# I Wanted To See The Description Of The Dataset While Rounding Up The Floated Fraction To 2 Decimals.
# I Also Transpose The Result To Be Able To See The Entire Columns

round(CreditCard.describe(), 2).T

#CreditCard.round(2)
```

Out[21]:

	count	mean	std	min	25%	50%	75%	max
Time	284807.0	94813.86	47488.15	0.00	54201.50	84692.00	139320.50	172792.00
V1	284807.0	0.00	1.96	-56.41	-0.92	0.02	1.32	2.45
V2	284807.0	-0.00	1.65	-72.72	-0.60	0.07	0.80	22.06
V3	284807.0	-0.00	1.52	-48.33	-0.89	0.18	1.03	9.38
V4	284807.0	0.00	1.42	-5.68	-0.85	-0.02	0.74	16.88
V5	284807.0	0.00	1.38	-113.74	-0.69	-0.05	0.61	34.80
V6	284807.0	0.00	1.33	-26.16	-0.77	-0.27	0.40	73.30
V7	284807.0	-0.00	1.24	-43.56	-0.55	0.04	0.57	120.59
V8	284807.0	0.00	1.19	-73.22	-0.21	0.02	0.33	20.01
V9	284807.0	-0.00	1.10	-13.43	-0.64	-0.05	0.60	15.59
V10	284807.0	0.00	1.09	-24.59	-0.54	-0.09	0.45	23.75
V11	284807.0	0.00	1.02	-4.80	-0.76	-0.03	0.74	12.02
V12	284807.0	0.00	1.00	-18.68	-0.41	0.14	0.62	7.85
V13	284807.0	0.00	1.00	-5.79	-0.65	-0.01	0.66	7.13
V14	284807.0	-0.00	0.96	-19.21	-0.43	0.05	0.49	10.53
V15	284807.0	-0.00	0.92	-4.50	-0.58	0.05	0.65	8.88
V16	284807.0	-0.00	0.88	-14.13	-0.47	0.07	0.52	17.32
V17	284807.0	-0.00	0.85	-25.16	-0.48	-0.07	0.40	9.25
V18	284807.0	0.00	0.84	-9.50	-0.50	-0.00	0.50	5.04
V19	284807.0	0.00	0.81	-7.21	-0.46	0.00	0.46	5.59
V20	284807.0	0.00	0.77	-54.50	-0.21	-0.06	0.13	39.42
V21	284807.0	-0.00	0.73	-34.83	-0.23	-0.03	0.19	27.20
V22	284807.0	-0.00	0.73	-10.93	-0.54	0.01	0.53	10.50
V23	284807.0	-0.00	0.62	-44.81	-0.16	-0.01	0.15	22.53
V24	284807.0	0.00	0.61	-2.84	-0.35	0.04	0.44	4.58

	count	mean	std	min	25%	50%	75%	max
V25	284807.0	-0.00	0.52	-10.30	-0.32	0.02	0.35	7.52
V26	284807.0	-0.00	0.48	-2.60	-0.33	-0.05	0.24	3.52
V27	284807.0	0.00	0.40	-22.57	-0.07	0.00	0.09	31.61
V28	284807.0	-0.00	0.33	-15.43	-0.05	0.01	0.08	33.85
Amount	284807.0	88.35	250.12	0.00	5.60	22.00	77.16	25691.16
Class	284807.0	0.00	0.04	0.00	0.00	0.00	0.00	1.00

In [22]:
```
# Checking For Any NaN or Null Values In The Columns Of The Dataset

CreditCard.isna().sum()
```

Out[22]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [23]:
```python
# Checking The Number Of Duplicated Values In The Database

CreditCard.duplicated().sum()
```

Out[23]:
```
1081
```

In [24]:
```python
# Dropping Or Deleting The Duplicated Data Values

CreditCard.drop_duplicates()
```

Out[24]:

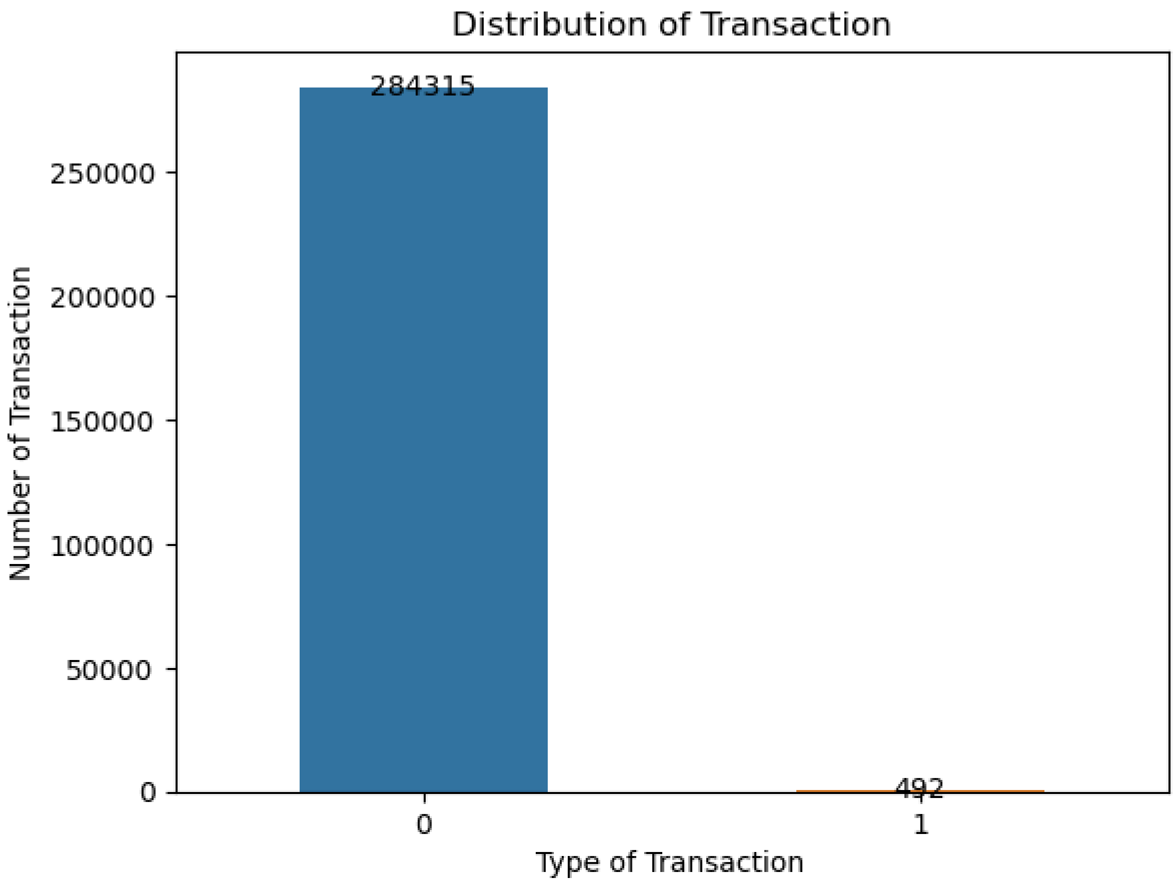
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.11
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.10
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.90
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.19
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.01
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.01
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.03
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.16
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.37

283726 rows × 31 columns

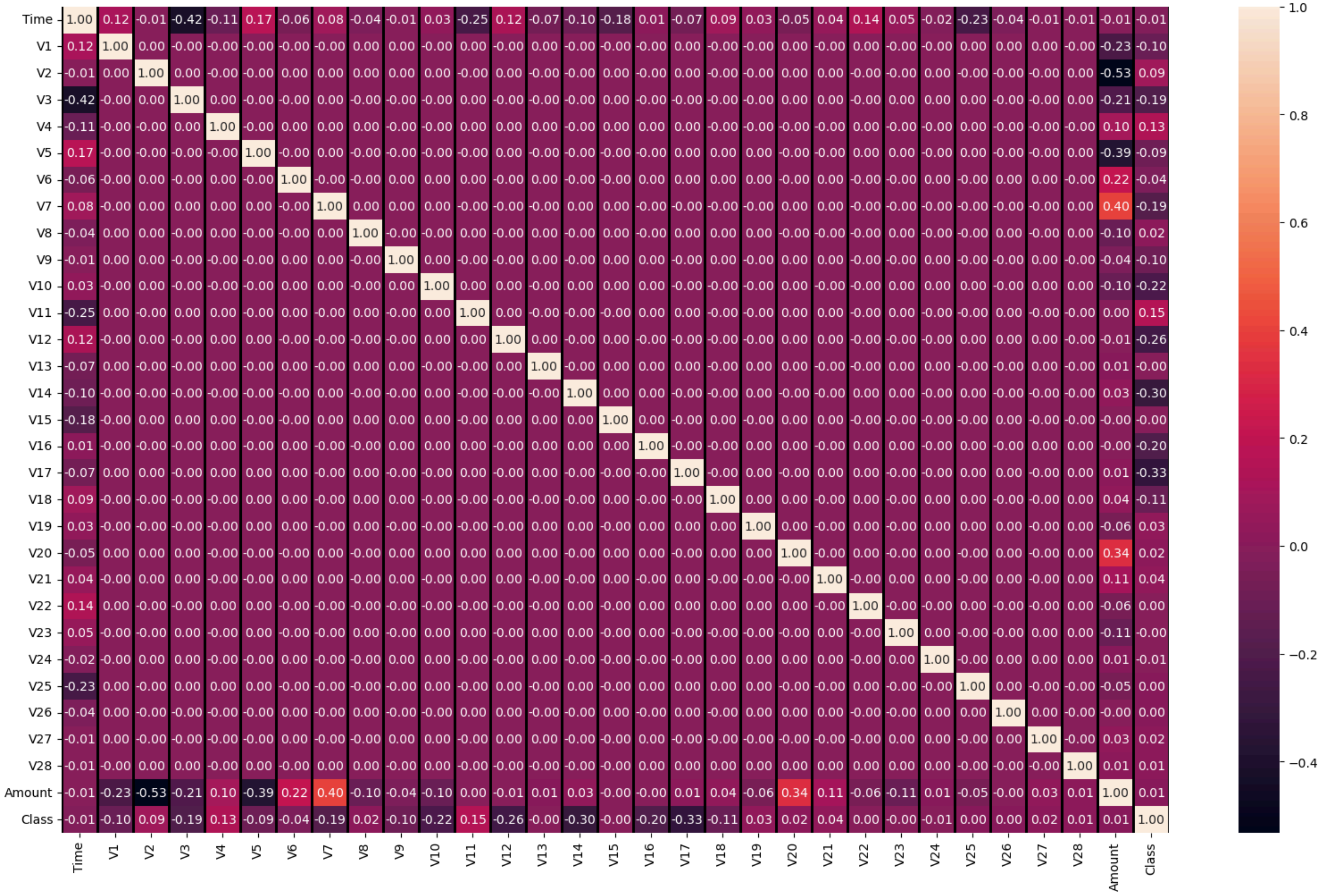
In [25]:
```
# Checking The Class Of Identified Legit & Fraudulent Transactions

CreditCard["Class"].value_counts()
```

Out[25]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [32]:
```
import seaborn as sb
import matplotlib.pyplot as plt

ax = sb.countplot(data = CreditCard, x ="Class", width = 0.5)
ax.set_title("Distribution of Transaction")
plt.xlabel("Type of Transaction")
plt.ylabel("Number of Transaction")

# Add bar labels
for p in ax.patches:
    ax.annotate(f"{p.get_height():.0f}", (p.get_x() + p.get_width() / 2, p.get_height()), ha="center", va="center")

plt.show()
```



In [33]:
```
plt.figure(figsize = (20, 12))
ax = sb.heatmap(CreditCard.corr(), annot = True, fmt = '.2f')

for i in range(CreditCard.shape[1] + 1):
    ax.axvline(i, color ='black', lw = 2)
    ax.axvline(i, color ='black', lw = 2)

#Plt.tight_Layout()
plt.show()
```


Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [35]:
```
# Create A Function That Seperates The Class Of Transactions Between Fraud & Legit Transactions

def split_data_by_class(CreditCard):

    legit = CreditCard[CreditCard["Class"] == 0]
    fraud = CreditCard[CreditCard["Class"] == 1]
    return legit, fraud

# Example Usage

legit_df, fraud_df = split_data_by_class(CreditCard)
```

In [36]:
```
legit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 284315 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284315 non-null  float64
 1   V1      284315 non-null  float64
 2   V2      284315 non-null  float64
 3   V3      284315 non-null  float64
 4   V4      284315 non-null  float64
 5   V5      284315 non-null  float64
 6   V6      284315 non-null  float64
 7   V7      284315 non-null  float64
 8   V8      284315 non-null  float64
 9   V9      284315 non-null  float64
 10  V10     284315 non-null  float64
 11  V11     284315 non-null  float64
 12  V12     284315 non-null  float64
 13  V13     284315 non-null  float64
 14  V14     284315 non-null  float64
 15  V15     284315 non-null  float64
 16  V16     284315 non-null  float64
 17  V17     284315 non-null  float64
 18  V18     284315 non-null  float64
 19  V19     284315 non-null  float64
 20  V20     284315 non-null  float64
 21  V21     284315 non-null  float64
 22  V22     284315 non-null  float64
 23  V23     284315 non-null  float64
 24  V24     284315 non-null  float64
 25  V25     284315 non-null  float64
 26  V26     284315 non-null  float64
 27  V27     284315 non-null  float64
 28  V28     284315 non-null  float64
 29  Amount  284315 non-null  float64
 30  Class   284315 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 69.4 MB
```

In [37]:
```
fraud_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 492 entries, 541 to 281674
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    492 non-null    float64
 1   V1      492 non-null    float64
 2   V2      492 non-null    float64
 3   V3      492 non-null    float64
 4   V4      492 non-null    float64
 5   V5      492 non-null    float64
 6   V6      492 non-null    float64
 7   V7      492 non-null    float64
 8   V8      492 non-null    float64
 9   V9      492 non-null    float64
 10  V10     492 non-null    float64
 11  V11     492 non-null    float64
 12  V12     492 non-null    float64
 13  V13     492 non-null    float64
 14  V14     492 non-null    float64
 15  V15     492 non-null    float64
 16  V16     492 non-null    float64
 17  V17     492 non-null    float64
 18  V18     492 non-null    float64
 19  V19     492 non-null    float64
 20  V20     492 non-null    float64
 21  V21     492 non-null    float64
 22  V22     492 non-null    float64
 23  V23     492 non-null    float64
 24  V24     492 non-null    float64
 25  V25     492 non-null    float64
 26  V26     492 non-null    float64
 27  V27     492 non-null    float64
 28  V28     492 non-null    float64
 29  Amount  492 non-null    float64
 30  Class   492 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 123.0 KB
```

In [38]:
```
legit_df.describe().T
```

Out[38]:

	count	mean	std	min	25%	50%	75%	max
Time	284315.0	94838.202258	47484.015786	0.000000	54230.000000	84711.000000	139333.000000	172792.000000
V1	284315.0	0.008258	1.929814	-56.407510	-0.917544	0.020023	1.316218	2.454930
V2	284315.0	-0.006271	1.636146	-72.715728	-0.599473	0.064070	0.800446	18.902453
V3	284315.0	0.012171	1.459429	-48.325589	-0.884541	0.182158	1.028372	9.382558
V4	284315.0	-0.007860	1.399333	-5.683171	-0.850077	-0.022405	0.737624	16.875344
V5	284315.0	0.005453	1.356952	-113.743307	-0.689399	-0.053457	0.612181	34.801666
V6	284315.0	0.002419	1.329913	-26.160506	-0.766847	-0.273123	0.399619	73.301626
V7	284315.0	0.009637	1.178812	-31.764946	-0.551442	0.041138	0.571019	120.589494
V8	284315.0	-0.000987	1.161283	-73.216718	-0.208633	0.022041	0.326200	18.709255
V9	284315.0	0.004467	1.089372	-6.290730	-0.640412	-0.049964	0.598230	15.594995
V10	284315.0	0.009824	1.044204	-14.741096	-0.532880	-0.091872	0.455135	23.745136
V11	284315.0	-0.006576	1.003112	-4.797473	-0.763447	-0.034923	0.736362	10.002190
V12	284315.0	0.010832	0.945939	-15.144988	-0.402102	0.141679	0.619207	7.848392
V13	284315.0	0.000189	0.995067	-5.791881	-0.648067	-0.013547	0.662492	7.126883
V14	284315.0	0.012064	0.897007	-18.392091	-0.422453	0.051947	0.494104	10.526766
V15	284315.0	0.000161	0.915060	-4.391307	-0.582812	0.048294	0.648842	8.877742
V16	284315.0	0.007164	0.844772	-10.115560	-0.465543	0.067377	0.523738	17.315112
V17	284315.0	0.011535	0.749457	-17.098444	-0.482644	-0.064833	0.399922	9.253526
V18	284315.0	0.003887	0.824919	-5.366660	-0.497414	-0.002787	0.501103	5.041069
V19	284315.0	-0.001178	0.811733	-7.213527	-0.456366	0.003117	0.457499	5.591971
V20	284315.0	-0.000644	0.769404	-54.497720	-0.211764	-0.062646	0.132401	39.420904
V21	284315.0	-0.001235	0.716743	-34.830382	-0.228509	-0.029821	0.185626	22.614889
V22	284315.0	-0.000024	0.723668	-10.933144	-0.542403	0.006736	0.528407	10.503090
V23	284315.0	0.000070	0.621541	-44.807735	-0.161702	-0.011147	0.147522	22.528412
V24	284315.0	0.000182	0.605776	-2.836627	-0.354425	0.041082	0.439869	4.584549

	count	mean	std	min	25%	50%	75%	max
V25	284315.0	-0.000072	0.520673	-10.295397	-0.317145	0.016417	0.350594	7.519589
V26	284315.0	-0.000089	0.482241	-2.604551	-0.327074	-0.052227	0.240671	3.517346
V27	284315.0	-0.000295	0.399847	-22.565679	-0.070852	0.001230	0.090573	31.612198
V28	284315.0	-0.000131	0.329570	-15.430084	-0.052950	0.011199	0.077962	33.847808
Amount	284315.0	88.291022	250.105092	0.000000	5.650000	22.000000	77.050000	25691.160000
Class	284315.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

In [39]:
```
fraud_df.describe().T
```

Out[39]:

	count	mean	std	min	25%	50%	75%	max
Time	492.0	80746.806911	47835.365138	406.000000	41241.500000	75568.500000	128483.000000	170348.000000
V1	492.0	-4.771948	6.783687	-30.552380	-6.036063	-2.342497	-0.419200	2.132386
V2	492.0	3.623778	4.291216	-8.402154	1.188226	2.717869	4.971257	22.057729
V3	492.0	-7.033281	7.110937	-31.103685	-8.643489	-5.075257	-2.276185	2.250210
V4	492.0	4.542029	2.873318	-1.313275	2.373050	4.177147	6.348729	12.114672
V5	492.0	-3.151225	5.372468	-22.105532	-4.792835	-1.522962	0.214562	11.095089
V6	492.0	-1.397737	1.858124	-6.406267	-2.501511	-1.424616	-0.413216	6.474115
V7	492.0	-5.568731	7.206773	-43.557242	-7.965295	-3.034402	-0.945954	5.802537
V8	492.0	0.570636	6.797831	-41.044261	-0.195336	0.621508	1.764879	20.007208
V9	492.0	-2.581123	2.500896	-13.434066	-3.872383	-2.208768	-0.787850	3.353525
V10	492.0	-5.676883	4.897341	-24.588262	-7.756698	-4.578825	-2.614184	4.031435
V11	492.0	3.800173	2.678605	-1.702228	1.973397	3.586218	5.307078	12.018913
V12	492.0	-6.259393	4.654458	-18.683715	-8.688177	-5.502530	-2.974088	1.375941
V13	492.0	-0.109334	1.104518	-3.127795	-0.979117	-0.065566	0.672964	2.815440
V14	492.0	-6.971723	4.278940	-19.214325	-9.692723	-6.729720	-4.282821	3.442422
V15	492.0	-0.092929	1.049915	-4.498945	-0.643539	-0.057227	0.609189	2.471358
V16	492.0	-4.139946	3.865035	-14.129855	-6.562915	-3.549795	-1.226043	3.139656
V17	492.0	-6.665836	6.970618	-25.162799	-11.945057	-5.302949	-1.341940	6.739384
V18	492.0	-2.246308	2.899366	-9.498746	-4.664576	-1.664346	0.091772	3.790316
V19	492.0	0.680659	1.539853	-3.681904	-0.299423	0.646807	1.649318	5.228342
V20	492.0	0.372319	1.346635	-4.128186	-0.171760	0.284693	0.822445	11.059004
V21	492.0	0.713588	3.869304	-22.797604	0.041787	0.592146	1.244611	27.202839
V22	492.0	0.014049	1.494602	-8.887017	-0.533764	0.048434	0.617474	8.361985
V23	492.0	-0.040308	1.579642	-19.254328	-0.342175	-0.073135	0.308378	5.466230
V24	492.0	-0.105130	0.515577	-2.028024	-0.436809	-0.060795	0.285328	1.091435

	count	mean	std	min	25%	50%	75%	max
V25	492.0	0.041449	0.797205	-4.781606	-0.314348	0.088371	0.456515	2.208209
V26	492.0	0.051648	0.471679	-1.152671	-0.259416	0.004321	0.396733	2.745261
V27	492.0	0.170575	1.376766	-7.263482	-0.020025	0.394926	0.826029	3.052358
V28	492.0	0.075667	0.547291	-1.869290	-0.108868	0.146344	0.381152	1.779364
Amount	492.0	122.211321	256.683288	0.000000	1.000000	9.250000	105.890000	2125.870000
Class	492.0	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [40]:
```
# Sampling The Legit Transaction To A Match a 492 Rows

new_legit_df = legit_df.sample(n = 494)
```

In [41]:
```
print(new_legit_df)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
265733  162024.0  1.974207 -0.387430 -0.451689  0.134795 -0.515102 -0.229197
61753    49972.0 -0.758607  0.449404  1.065224 -1.031592  1.919861  4.114126
80912    58712.0 -1.271449  0.119183  2.663678 -0.376898 -1.099947  0.683283
71661    54388.0  1.127480 -0.023485  0.704488  0.767368 -0.421560  0.089989
123351   76892.0  0.726659 -1.082094  0.922459  0.348667 -0.974547  0.963774
...          ...       ...       ...       ...       ...       ...       ...
189145  128276.0  0.438663 -3.829177 -1.439196  0.026570 -2.114516 -0.477849
247375  153555.0  2.003203 -0.905000 -0.621291 -1.596843 -0.715603 -0.317958
277346  167601.0  2.043279  0.107894 -1.707238  0.438046  0.380913 -0.895351
189809  128570.0 -0.086611 -0.499212  1.049712 -1.573742 -0.259965 -0.932601
4618      3980.0 -0.831683  1.600308  1.395368  0.453966  0.081371 -0.847971

              V7        V8        V9  ...       V21       V22       V23  \
265733 -0.633304  0.147492  1.236572  ... -0.098344 -0.216356  0.310272
61753  -0.474848  1.218376  0.161696  ... -0.135367 -0.305711 -0.271313
80912   0.612386  0.301021  0.269563  ...  0.334883  0.889258  0.135969
71661  -0.251870  0.122076  0.216395  ...  0.012993  0.327089 -0.055351
123351 -0.701790  0.469820  0.928445  ... -0.022200 -0.209063 -0.023710
...          ...       ...       ...  ...       ...       ...       ...
189145  0.228858 -0.278248  0.250504  ...  0.314252 -1.077135 -0.329995
247375 -0.735025 -0.108139  2.292791  ...  0.058700  0.409572  0.052608
277346  0.203800 -0.240379  0.495955  ... -0.353709 -0.949931  0.330885
189809 -0.380530 -0.021664  1.547456  ...  0.337319  1.140715  0.307814
4618    0.717814 -0.370201  1.434830  ... -0.473444 -0.767179  0.022050

             V24       V25       V26       V27       V28  Amount  Class
265733 -0.551677 -0.391564 -0.910489  0.051991 -0.043761    1.00      0
61753   1.033032  0.392411  0.382671  0.298780  0.130677   11.50      0
80912   0.142221  0.354853  0.569772  0.238929  0.146259  209.00      0
71661   0.307961  0.483801  0.435412  0.002703  0.000866    5.47      0
123351 -0.222476 -0.122534  0.935237 -0.039644  0.028365  188.45      0
...          ...       ...       ...       ...       ...     ...    ...
189145 -0.039156 -0.864277 -0.659462 -0.163278  0.107499  894.42      0
247375 -0.930707 -0.189862 -0.254123  0.054057 -0.030343   58.97      0
277346  0.567781 -0.269384  0.168143 -0.065790 -0.027555   16.99      0
189809  0.050595 -1.612195 -0.356134  0.356488  0.317420   29.02      0
4618    0.263935 -0.484123  0.065505  0.241210  0.109369    2.15      0

[494 rows x 31 columns]
```

In [42]:
```
# Combine The Fraud & The Datasets

combine_df = pd.concat([new_legit_df, fraud_df], axis = 0)
```

In [43]:
```
# View The Combined Fraud & Legit Datasets

combine_df
```

Out[43]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V
265733	162024.0	1.974207	-0.387430	-0.451689	0.134795	-0.515102	-0.229197	-0.633304	0.147492	1.236572	...	-0.098344	-0.216356	0.310
61753	49972.0	-0.758607	0.449404	1.065224	-1.031592	1.919861	4.114126	-0.474848	1.218376	0.161696	...	-0.135367	-0.305711	-0.271
80912	58712.0	-1.271449	0.119183	2.663678	-0.376898	-1.099947	0.683283	0.612386	0.301021	0.269563	...	0.334883	0.889258	0.135
71661	54388.0	1.127480	-0.023485	0.704488	0.767368	-0.421560	0.089989	-0.251870	0.122076	0.216395	...	0.012993	0.327089	-0.055
123351	76892.0	0.726659	-1.082094	0.922459	0.348667	-0.974547	0.963774	-0.701790	0.469820	0.928445	...	-0.022200	-0.209063	-0.023
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072

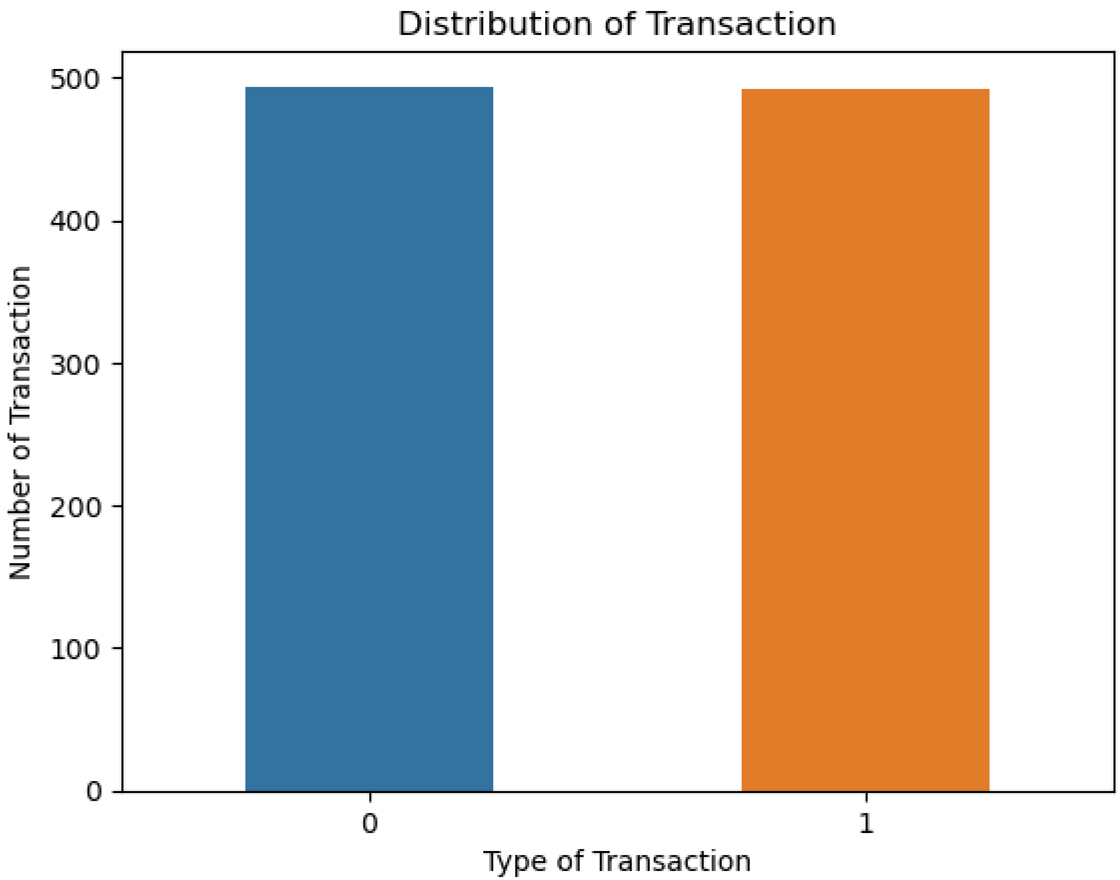
986 rows × 31 columns

In [44]:
```
# Viewing The New Combine Data

combine_df["Class"].value_counts()
```

Out[44]:
```
Class
0    494
1    492
Name: count, dtype: int64
```

In [45]:
```
# Visualizing The New Combine Data

import seaborn as sb
import matplotlib.pyplot as plt

ax = sb.countplot(data = combine_df, x ="Class", width = 0.5)
ax.set_title("Distribution of Transaction")
plt.xlabel("Type of Transaction")
plt.ylabel("Number of Transaction")

# Add bar labels
for x in ax.containers:
    ax.bar_label(i)

plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[45], line 13
     11 # Add bar labels
     12 for x in ax.containers:
---> 13     ax.bar_label(i)
     15 plt.show()

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:2714, in Axes.bar_label(self, container, labels, fmt, la
bel_type, padding, **kwargs)
   2710     return 1 if x >= 0 else -1
   2712 _api.check_in_list(['edge', 'center'], label_type=label_type)
-> 2714 bars = container.patches
   2715 errorbar = container.errorbar
   2716 datavalues = container.datavalues

AttributeError: 'int' object has no attribute 'patches'
```



In [46]:
```
x = combine_df.drop(columns = "Class", axis = 1)
y = combine_df['Class']
```

In [47]:
```
x
```

Out[47]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V
265733	162024.0	1.974207	-0.387430	-0.451689	0.134795	-0.515102	-0.229197	-0.633304	0.147492	1.236572	...	-0.290591	-0.098344	-0.216
61753	49972.0	-0.758607	0.449404	1.065224	-1.031592	1.919861	4.114126	-0.474848	1.218376	0.161696	...	0.282982	-0.135367	-0.305
80912	58712.0	-1.271449	0.119183	2.663678	-0.376898	-1.099947	0.683283	0.612386	0.301021	0.269563	...	0.544463	0.334883	0.889
71661	54388.0	1.127480	-0.023485	0.704488	0.767368	-0.421560	0.089989	-0.251870	0.122076	0.216395	...	-0.101155	0.012993	0.327
123351	76892.0	0.726659	-1.082094	0.922459	0.348667	-0.974547	0.963774	-0.701790	0.469820	0.928445	...	0.196459	-0.022200	-0.209
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	1.252967	0.778584	-0.319
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.226138	0.370612	0.028
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.247968	0.751826	0.834
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.306271	0.583276	-0.269
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.017652	-0.164350	-0.295

986 rows × 30 columns

In [48]:
```
y
```

Out[48]:
```
265733    0
61753     0
80912     0
71661     0
123351    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 986, dtype: int64
```

In [50]:
```
# Importing The Model Building Libraries

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [51]:
```
# Splitting & Training The Datasets

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 5)
```

In [52]:
```
print(x_train)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
269075  163516.0  2.271188 -0.481887 -1.666204 -1.095342  0.043503 -0.938331
159254  112381.0  2.025281 -0.131606 -1.421224  0.079675  0.203822 -0.560311
208606  137193.0  2.002944 -0.116406 -0.972243  0.453323 -0.161306 -0.416279
278609  168320.0 -0.428184  1.548500 -0.295260  0.613493  0.520980  0.730037
42009    40919.0 -2.740483  3.658095 -4.110636  5.340242 -2.666775 -0.092782
...          ...       ...       ...       ...       ...       ...       ...
175681  122442.0  2.002072 -0.477963 -0.293072  0.481426 -0.915512 -0.716347
142461   84730.0 -1.178840 -2.091563  1.321520 -0.548771 -1.500015 -0.570644
88307    62080.0 -1.599457  2.607720 -2.987193  3.064156 -2.497914 -0.541103
284623  172617.0  2.038001 -0.089042 -2.260703  0.095598  0.697565 -1.111463
191267  129186.0  0.290155  0.049243 -0.740524  2.865463  1.395294 -0.535163

              V7        V8        V9  ...       V20       V21       V22  \
269075 -0.089118 -0.428072 -0.882930  ...  0.075132  0.037109  0.092434
159254  0.005475 -0.088744  0.583915  ... -0.264670  0.318212  1.003712
208606 -0.477768  0.044646  1.047847  ... -0.172185 -0.340028 -0.913069
278609 -0.257539 -2.362050 -0.473266  ... -0.099233  2.085477 -1.044427
42009  -4.388699 -0.280133 -2.821895  ...  0.185325  2.417495 -0.097712
...          ...       ...       ...  ...       ...       ...       ...
175681 -0.676667 -0.034583  1.325683  ... -0.261841  0.234757  0.843774
142461  0.231241 -0.211763 -2.836147  ...  1.060681 -0.046149 -0.654341
88307  -2.277786  1.268166 -1.997331  ...  0.225333  0.662933  0.184087
284623  0.734825 -0.482797  0.017646  ... -0.055889  0.084980  0.173524
191267  0.142543 -0.222770 -1.463691  ...  0.247580  0.337349  1.018191

             V23       V24       V25       V26       V27       V28  Amount
269075  0.200103  0.506221  0.039206 -0.335699 -0.044277 -0.055951    9.99
159254  0.022232  0.710926  0.278382 -0.467728 -0.001156 -0.058464    1.00
208606  0.452291  0.467572 -0.616831  0.098869 -0.021057 -0.011901    6.50
278609  0.353363 -1.039268 -1.032648  0.597424  0.571382  0.293567    0.89
42009   0.382155 -0.154757 -0.403956  0.277895  0.830062  0.218690  112.33
...          ...       ...       ...       ...       ...       ...     ...
175681  0.133598  0.086810 -0.250919  0.643172 -0.030009 -0.051378    5.99
142461  1.153665  0.301179 -0.391794 -0.286919  0.018805  0.188031  390.00
88307  -0.089452 -0.506000 -0.062259 -0.052714  0.322854  0.135268  180.00
284623 -0.014877  0.543928  0.249404  0.705467 -0.141402 -0.067495   66.30
191267  0.303550  0.833886 -1.222306  2.745261 -0.220402  0.168233    7.18

[739 rows x 30 columns]
```

In [53]:
```
print(x_test)
```

```
            Time         V1         V2         V3         V4         V5         V6  \
243547  151972.0  -6.618211   3.835943  -6.316453   1.844111  -2.476892  -1.886718
67262    52439.0 -15.128164  -4.759922  -4.388698   2.968675   1.412581   1.017313
43773    41646.0  -3.240187   2.978122  -4.162314   3.869124  -3.645256  -0.126271
119714   75556.0  -0.734303   0.435519  -0.530866  -0.471120   0.643214   0.713832
125491   77691.0   1.453135  -0.897291   0.165177  -1.606075  -0.915341  -0.190063
...          ...        ...        ...        ...        ...        ...        ...
4888      4416.0   0.946443   0.600387   0.889782   0.929058  -0.088773  -0.750707
154697  102625.0  -4.221221   2.871121  -5.888716   6.890952  -3.404894  -1.154394
212516  138894.0  -1.298443   1.948100  -4.509947   1.305805  -0.019486  -0.509238
213092  139107.0  -4.666500  -3.952320   0.206094   5.153525   5.229469   0.939040
174607  121987.0  -3.083586  -5.493558  -1.380488  -0.720477   4.448321  -4.769857

              V7         V8         V9  ...       V20        V21       V22  \
243547 -3.817495   0.613470  -1.482121  ... -0.953827   1.636622  0.038727
67262  -9.760064 -14.018265  -0.041771  ... -7.348950 -10.738634  4.198538
43773  -4.744730  -0.065331  -2.168366  ... -0.224043   2.601441  0.231910
119714 -1.234572  -2.551412  -2.057724  ...  0.864536  -1.004877  1.150354
125491 -0.801854  -0.030195  -2.396045  ... -0.293676  -0.577895 -1.348865
...          ...        ...        ...  ...       ...        ...       ...
4888    0.351693  -0.401452   1.263447  ... -0.061922  -0.274584 -0.250811
154697 -7.739928   2.851363  -2.507569  ... -0.227882   1.620591  1.567947
212516 -2.643398   1.283545  -2.515356  ...  0.250415   1.178032  1.360989
213092 -0.635033  -0.704506  -0.234786  ... -2.286137  -0.664263  1.821422
174607 -0.980379  -0.423880  -1.171931  ...  1.577776   1.279182  2.018716

             V23        V24        V25        V26        V27        V28  Amount
243547  0.278218   0.786670   0.063895   0.154707  -2.042403   1.405141   57.73
67262   7.441508  -1.163202   1.156147   0.022968   3.416390  -2.723858    1.18
43773  -0.036490   0.042640  -0.438330  -0.125821   0.421300   0.003146  172.32
119714 -0.152555  -1.386745   0.004716   0.219146  -0.058257   0.158048   29.95
125491  0.255947  -0.373492   0.045295  -0.611944   0.032547   0.010199   20.00
...          ...        ...        ...        ...        ...        ...     ...
4888    0.474237   0.549103  -1.324774  -0.103604  -0.221122  -0.350267    0.89
154697 -0.578007  -0.059045  -1.829169  -0.072429   0.136734  -0.599848    7.59
212516 -0.272013  -0.325948   0.290703   0.841295   0.643094   0.201156    0.01
213092  0.113563  -0.759673  -0.502304   0.630639  -0.513880   0.729526   22.47
174607  1.376520   0.567477  -0.920750  -0.234729  -0.156569   0.196564  249.29

[247 rows x 30 columns]
```

In [54]:
```
print(y_train)
```

```
269075    0
159254    0
208606    0
278609    0
42009     1
         ..
175681    0
142461    0
88307     1
284623    0
191267    1
Name: Class, Length: 739, dtype: int64
```

In [55]:
```
print(y_test)
```

```
243547    1
67262     0
43773     1
119714    1
125491    0
         ..
4888      0
154697    1
212516    1
213092    1
174607    0
Name: Class, Length: 247, dtype: int64
```

In [56]:
```
print(x_train.shape, x_test.shape)
```

```
(739, 30) (247, 30)
```

In [57]:
```
model = LogisticRegression()
```

In [58]:
```
model.fit(x_train, y_train)
```

Out[58]:
```
▾ LogisticRegression
LogisticRegression()
```

In [60]:
```
model.fit(x_test, y_test)
```

Out[60]:
```
▾ LogisticRegression
LogisticRegression()
```

In [64]:
```
x_train_predict = model.predict(x_train)
training_data_accuracy = accuracy_score(x_train_predict, y_train)
print("The model's training data accuracy is: {:.2f}%".format(training_data_accuracy * 100))
```

```
The model's training data accuracy is: 91.34%
```

In [67]:
```
x_test_predict = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_predict, y_test)
print("The model's test data accuracy is: {:.2f}%".format(test_data_accuracy * 100))
```

```
The model's test data accuracy is: 91.09%
```

In [ ]:
```

```