In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [3]:

```
df =pd.read_csv("D:\Data files\creditcard.csv")
```

In [5]:

```
df.head(10)
```

Out[5]:

	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	0.066
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	-0.339
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	-0.689
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	-1.175
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	0.141
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	-0.371
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	-0.780
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	-0.649
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	1.011
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	-0.385

10 rows × 31 columns

In [6]:

```
df.tail()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V2
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	1.01448
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	0.01246
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	-0.03750
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	-0.16329
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	0.37677

5 rows × 31 columns

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
```

```
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [8]:

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

Out[8]:

```
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 [9]:

```
#now gettinng some info regarding our target column"Class"

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

Out[9]:

```
0    284315
1       492
Name: Class, dtype: int64
```

In [10]:

```
Normal = df[df.Class== 0]
fraud = df[df.Class== 1]
```

In [11]:

```
print(fraud)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
541        406.0 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623        472.0 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.064823
4920      4462.0 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075788
6108      6986.0 -4.397974  1.358367 -2.592844  2.679787 -1.128131 -1.706536
6329      7519.0  1.234235  3.019740 -4.304597  4.732795  3.624201 -1.357746
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V21       V22       V23  \
541    -2.537387  1.391657 -2.770089  ...  0.517232 -0.035049 -0.465211
623     0.325574 -0.067794 -0.270953  ...  0.661696  0.435477  1.375966
4920    0.562320 -0.399147 -0.238253  ... -0.294166 -0.932391  0.172726
6108   -3.496197 -0.248778 -0.247768  ...  0.573574  0.176968 -0.436207
```

```
6329    1.713445 -0.496358 -1.282858  ... -0.379068 -0.704181 -0.656805
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  0.778584 -0.319189  0.639419
280143 -1.413170  0.248525 -1.127396  ...  0.370612  0.028234 -0.145640
280149 -2.234739  1.210158 -0.652250  ...  0.751826  0.834108  0.190944
281144 -2.208002  1.058733 -1.632333  ...  0.583276 -0.269209 -0.456108
281674  0.223050 -0.068384  0.577829  ... -0.164350 -0.295135 -0.072173

             V24       V25       V26       V27       V28  Amount  Class
541     0.320198  0.044519  0.177840  0.261145 -0.143276    0.00      1
623    -0.293803  0.279798 -0.145362 -0.252773  0.035764  529.00      1
4920   -0.087330 -0.156114 -0.542628  0.039566 -0.153029  239.93      1
6108   -0.053502  0.252405 -0.657488 -0.827136  0.849573   59.00      1
6329   -1.632653  1.488901  0.566797 -0.010016  0.146793    1.00      1
...          ...       ...       ...       ...       ...     ...    ...
279863 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00      1
280143 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76      1
280149  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89      1
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[492 rows x 31 columns]
```

In [13]:

```
fraud.Amount.describe()
```

Out[13]:

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [14]:

```
Normal.Amount.describe()
```

Out[14]:

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
```

```
25%            5.650000
50%           22.000000
75%           77.050000
max        25691.160000
Name: Amount, dtype: float64
```

In [15]:

```
df.corr()
```

Out[15]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V
Time	1.000000	1.173963e-01	-1.059333e-02	-4.196182e-01	-1.052602e-01	1.730721e-01	-6.301647e-02	8.471437e-02	-3.694943e-02	-8.660434e 0
V1	0.117396	1.000000e+00	4.135835e-16	-1.227819e-15	-9.215150e-16	1.812612e-17	-6.506567e-16	-1.005191e-15	-2.433822e-16	-1.513678e 1
V2	-0.010593	4.135835e-16	1.000000e+00	3.243764e-16	-1.121065e-15	5.157519e-16	2.787346e-16	2.055934e-16	-5.377041e-17	1.978488e-1
V3	-0.419618	-1.227819e-15	3.243764e-16	1.000000e+00	4.711293e-16	-6.539009e-17	1.627627e-15	4.895305e-16	-1.268779e-15	5.568367e-1
V4	-0.105260	-9.215150e-16	-1.121065e-15	4.711293e-16	1.000000e+00	-1.719944e-15	-7.491959e-16	-4.104503e-16	5.697192e-16	6.923247e-1
V5	0.173072	1.812612e-17	5.157519e-16	-6.539009e-17	-1.719944e-15	1.000000e+00	2.408382e-16	2.715541e-16	7.437229e-16	7.391702e-1
V6	-0.063016	-6.506567e-16	2.787346e-16	1.627627e-15	-7.491959e-16	2.408382e-16	1.000000e+00	1.191668e-16	-1.104219e-16	4.131207e-1
V7	0.084714	-1.005191e-15	2.055934e-16	4.895305e-16	-4.104503e-16	2.715541e-16	1.191668e-16	1.000000e+00	3.344412e-16	1.122501e-1
V8	-0.036949	-2.433822e-16	-5.377041e-17	-1.268779e-15	5.697192e-16	7.437229e-16	-1.104219e-16	3.344412e-16	1.000000e+00	4.356078e-1
V9	-0.008660	-1.513678e-16	1.978488e-17	5.568367e-16	6.923247e-16	7.391702e-16	4.131207e-16	1.122501e-15	4.356078e-16	1.000000e+0
V10	0.030617	7.388135e-17	-3.991394e-16	1.156587e-15	2.232685e-16	-5.202306e-16	5.932243e-17	-7.492834e-17	-2.801370e-16	-4.642274e 1
V11	-0.247689	2.125498e-16	1.975426e-16	1.576830e-15	3.459380e-16	7.203963e-16	1.980503e-15	1.425248e-16	2.487043e-16	1.354680e-1

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V
V12	0.124348	2.053457e-16	-9.568710e-17	6.310231e-16	-5.625518e-16	7.412552e-16	2.375468e-16	-3.536655e-18	1.839891e-16	-1.079314e 1
V13	-0.065902	-2.425603e-17	6.295388e-16	2.807652e-16	1.303306e-16	5.886991e-16	-1.211182e-16	1.266462e-17	-2.921856e-16	2.251072e-1
V14	-0.098757	-5.020280e-16	-1.730566e-16	4.739859e-16	2.282280e-16	6.565143e-16	2.621312e-16	2.607772e-16	-8.599156e-16	3.784757e-1
V15	-0.183453	3.547782e-16	-4.995814e-17	9.068793e-16	1.377649e-16	-8.720275e-16	-1.531188e-15	-1.690540e-16	4.127777e-16	-1.051167e-1
V16	0.011903	7.212815e-17	1.177316e-17	8.299445e-16	-9.614528e-16	2.246261e-15	2.623672e-18	5.869302e-17	-5.254741e-16	-1.214086e 1
V17	-0.073297	-3.879840e-16	-2.685296e-16	7.614712e-16	-2.699612e-16	1.281914e-16	2.015618e-16	2.177192e-16	-2.269549e-16	1.113695e-1
V18	0.090438	3.230206e-17	3.284605e-16	1.509897e-16	-5.103644e-16	5.308590e-16	1.223814e-16	7.604126e-17	-3.667974e-16	4.993240e-1
V19	0.028975	1.502024e-16	-7.118719e-18	3.463522e-16	-3.980557e-16	-1.450421e-16	-1.865597e-16	-1.881008e-16	-3.875186e-16	-1.376135e 1
V20	-0.050866	4.654551e-16	2.506675e-16	-9.316409e-16	-1.857247e-16	-3.554057e-16	-1.858755e-16	9.379684e-16	2.033737e-16	-2.343720e 1
V21	0.044736	-2.457409e-16	-8.480447e-17	5.706192e-17	-1.949553e-16	-3.920976e-16	5.833316e-17	-2.027779e-16	3.892798e-16	1.936953e-1
V22	0.144059	-4.290944e-16	1.526333e-16	-1.133902e-15	-6.276051e-17	1.253751e-16	-4.705235e-19	-8.898922e-16	2.026927e-16	-7.071869e 1
V23	0.051142	6.168652e-16	1.634231e-16	-4.983035e-16	9.164206e-17	-8.428683e-18	1.046712e-16	-4.387401e-16	6.377260e-17	-5.214137e 1
V24	-0.016182	-4.425156e-17	1.247925e-17	2.686834e-19	1.584638e-16	-1.149255e-15	-1.071589e-15	7.434913e-18	-1.047097e-16	-1.430343e 1
V25	-0.233083	-9.605737e-16	-4.478846e-16	-1.104734e-15	6.070716e-16	4.808532e-16	4.562861e-16	-3.094082e-16	-4.653279e-16	6.757763e-1
V26	-0.041407	-1.581290e-17	2.057310e-16	-1.238062e-16	-4.247268e-16	4.319541e-16	-1.357067e-16	-9.657637e-16	-1.727276e-16	-7.888853e 1
V27	-0.005135	1.198124e-16	-4.966953e-16	1.045747e-15	3.977061e-17	6.590482e-16	-4.452461e-16	-1.782106e-15	1.299943e-16	-6.709655e 1

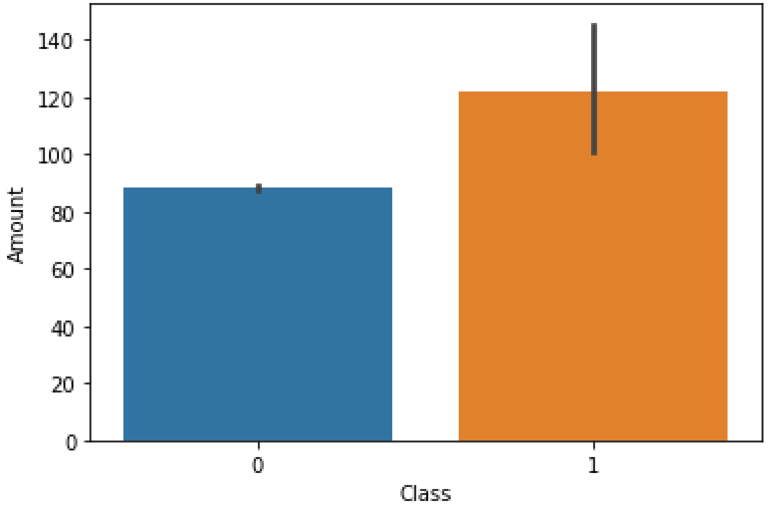
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V
V28	-0.009413	2.083082e-15	-5.093836e-16	9.775546e-16	-2.761403e-18	-5.613951e-18	2.594754e-16	-2.776530e-16	-6.200930e-16	1.110541e-1
Amount	-0.010596	-2.277087e-01	-5.314089e-01	-2.108805e-01	9.873167e-02	-3.863563e-01	2.159812e-01	3.973113e-01	-1.030791e-01	-4.424560e 0
Class	-0.012323	-1.013473e-01	9.128865e-02	-1.929608e-01	1.334475e-01	-9.497430e-02	-4.364316e-02	-1.872566e-01	1.987512e-02	-9.773269e 0

31 rows × 31 columns

In [16]:

```
sns.barplot(x = df['Class'], y = df['Amount'])
```

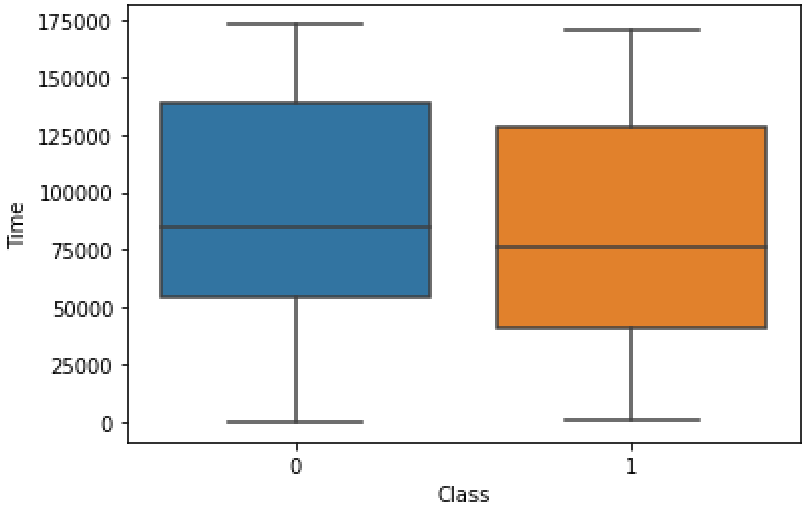
Out[16]: <AxesSubplot:xlabel='Class', ylabel='Amount'>



In [17]:

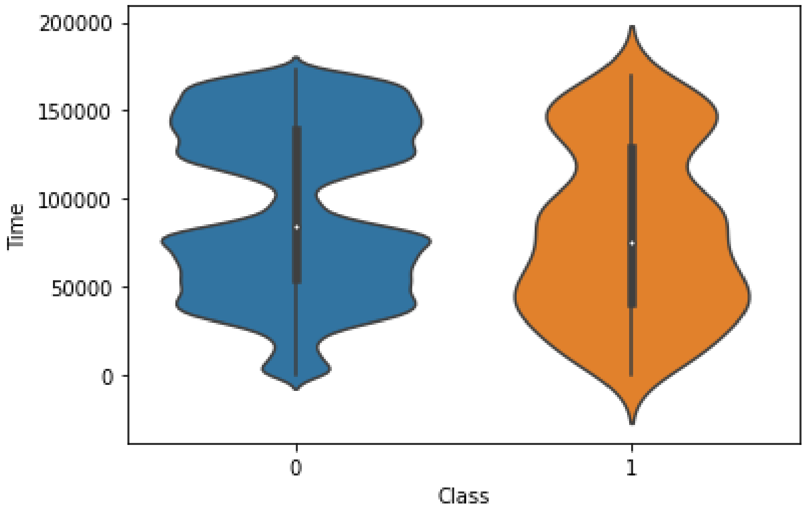
```
sns.boxplot(y = df['Time'], x = df['Class'])
```

Out[17]: <AxesSubplot:xlabel='Class', ylabel='Time'>



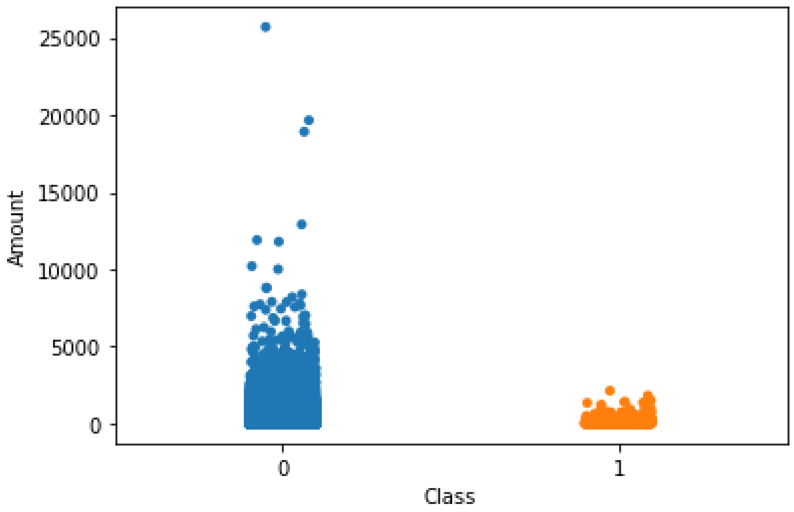
In [18]:
```
sns.violinplot(y = df['Time'], x = df['Class'])
```

Out[18]:
```
<AxesSubplot:xlabel='Class', ylabel='Time'>
```



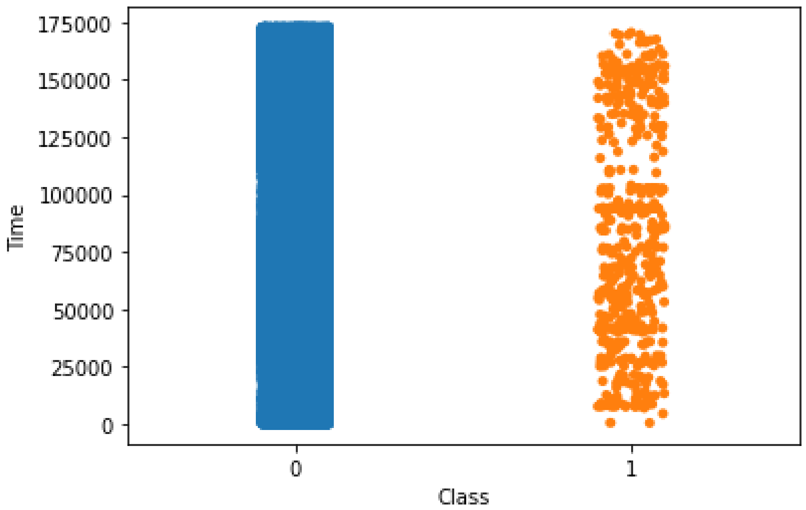
In [19]:
```
sns.stripplot(y = df['Amount'], x = df['Class'])
```

Out[19]: <AxesSubplot:xlabel='Class', ylabel='Amount'>



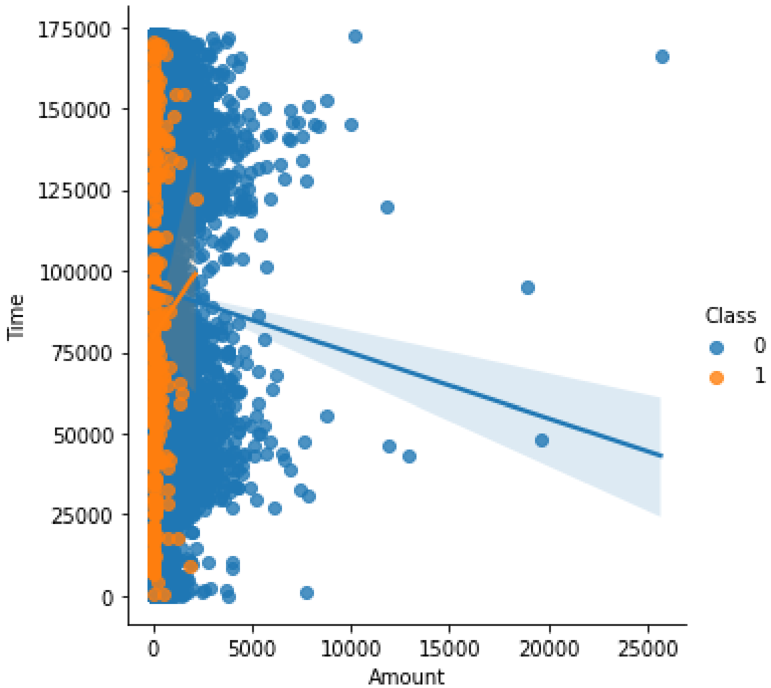
In [20]:
```
sns.stripplot(y = df['Time'], x = df['Class'])
```

Out[20]: <AxesSubplot:xlabel='Class', ylabel='Time'>



In [21]:
```
sns.lmplot(x='Amount', y='Time', data=df, hue="Class")
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x231486b03d0>



In [23]:
```
df.groupby('Class').mean()
```

Out[23]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	
Class														
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.001235	-0.0(
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.0'

2 rows × 30 columns

data set is quite unbalanced there is huge difference btw normal and fraudulent transaction

In [22]:
```
Normal_sample = Normal.sample(n=492)
```

In [24]:
```
new_DS = pd.concat([Normal_sample,fraud],axis = 0)
```

In [25]:
```
new_DS.head(10)
```

Out[25]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V
114141	73352.0	-0.643988	0.027076	0.833219	0.056823	-0.736694	1.367252	-0.315082	0.791502	-1.388007	...	0.013705	0.331415	0.1524
237943	149461.0	-1.096401	0.632418	1.670101	-0.025553	-0.171099	0.185873	-0.303729	0.646746	-0.265961	...	0.306181	0.823680	-0.4953
9471	14029.0	-1.392495	0.648354	2.773889	3.196665	-0.600547	1.236008	0.305347	0.093096	1.202625	...	-0.353657	-0.058054	0.1779
242283	151423.0	0.232189	0.941870	-0.611982	-0.562880	1.258595	-0.919127	1.143471	-0.438695	0.188747	...	-0.286990	-0.464250	0.2123
27621	34629.0	0.773271	-1.114798	0.740682	0.255502	-1.277445	0.055913	-0.566152	0.189316	0.800965	...	0.224493	0.213760	-0.2644
104528	69111.0	-0.514623	0.977262	1.126972	-0.114227	0.412296	-0.365229	0.919477	-0.059294	-0.990392	...	0.185562	0.523681	-0.0973
105914	69746.0	-1.865757	1.733006	0.720265	-0.222788	-0.983962	-0.606863	0.571125	0.471771	-0.031584	...	-0.305593	-0.553122	0.1602
166242	117949.0	-0.475986	-4.123578	-3.881512	1.375609	-0.640504	-1.308704	2.823923	-1.046474	-0.336423	...	1.088579	0.272964	-1.3633
80871	58692.0	-0.846328	0.124692	2.393975	1.506541	0.009698	1.405863	0.821615	-0.284832	-0.012751	...	0.045530	0.900258	0.0035
124096	77174.0	1.383051	-0.808056	0.648904	-0.925350	-1.143584	0.038841	-1.205615	0.187688	-0.456206	...	0.369750	0.938388	-0.2085

10 rows × 31 columns

In [26]:
```
new_DS.tail(10)
```

Out[26]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V
274382	165981.0	-5.766879	-8.402154	0.056543	6.950983	9.880564	-5.773192	-5.748879	0.721743	-1.076274	...	0.880395	-0.130436	2.2414
274475	166028.0	-0.956390	2.361594	-3.171195	1.970759	0.474761	-1.902598	-0.055178	0.277831	-1.745854	...	0.473211	0.719400	0.1224
275992	166831.0	-2.027135	-1.131890	-1.135194	1.086963	-0.010547	0.423797	3.790880	-1.155595	-0.063434	...	-0.315105	0.575520	0.4908
276071	166883.0	2.091900	-0.757459	-1.192258	-0.755458	-0.620324	-0.322077	-1.082511	0.117200	-0.140927	...	0.288253	0.831939	0.1420
276864	167338.0	-1.374424	2.793185	-4.346572	2.400731	-1.688433	0.111136	-0.922038	-2.149930	-2.027474	...	-0.870779	0.504849	0.1379
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	0.6394
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	-0.1456
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	0.1909
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	-0.4561
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	-0.0721

10 rows × 31 columns

In [28]:

```
new_DS['Class'].value_counts()
```

Out[28]:

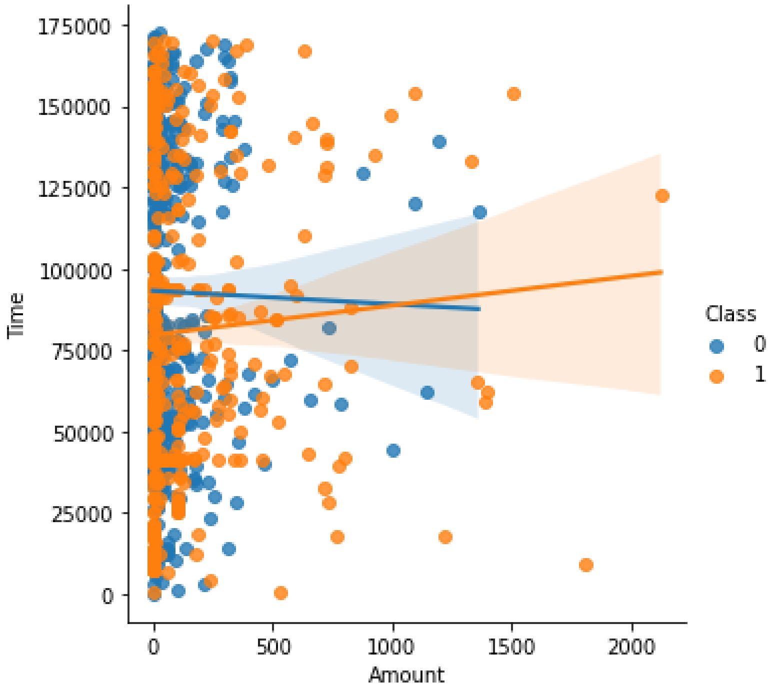
```
0    492
1    492
Name: Class, dtype: int64
```

In [29]:

```
sns.lmplot(x='Amount', y='Time', data=new_DS, hue="Class")
```

Out[29]:

```
<seaborn.axisgrid.FacetGrid at 0x231494d37c0>
```



In [30]:

```
# spliting data set for training and testing
# step#1 spliting the label(Y) nd Factors(x)

X= new_DS.drop(columns ='Class', axis=1)
y= new_DS['Class']
```

In [32]:

```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
114141   73352.0 -0.643988  0.027076  0.833219  0.056823 -0.736694  1.367252
237943  149461.0 -1.096401  0.632418  1.670101 -0.025553 -0.171099  0.185873
9471     14029.0 -1.392495  0.648354  2.773889  3.196665 -0.600547  1.236008
242283  151423.0  0.232189  0.941870 -0.611982 -0.562880  1.258595 -0.919127
27621    34629.0  0.773271 -1.114798  0.740682  0.255502 -1.277445  0.055913
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
```

```
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V20       V21       V22  \
114141 -0.315082  0.791502 -1.388007  ... -0.262706  0.013705  0.331415
237943 -0.303729  0.646746 -0.265961  ...  0.102339  0.306181  0.823680
9471    0.305347  0.093096  1.202625  ...  0.156233 -0.353657 -0.058054
242283  1.143471 -0.438695  0.188747  ...  0.064516 -0.286990 -0.464250
27621  -0.566152  0.189316  0.800965  ...  0.348621  0.224493  0.213760
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135

             V23       V24       V25       V26       V27       V28  Amount
114141  0.152402 -1.192954 -0.445267 -0.169486  0.096869  0.110978  135.00
237943 -0.495328 -0.277173  0.603836  0.089087 -0.019539 -0.046429    9.99
9471    0.177994 -0.005416  0.032830  0.236688  0.270658 -0.006230  136.90
242283  0.212312  0.583572 -1.276640 -0.073954  0.097633  0.013812    1.29
27621  -0.264411  0.091172  0.110221  1.090849 -0.088614  0.038372  230.00
...          ...       ...       ...       ...       ...       ...     ...
279863  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00
280143 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76
280149  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
281674 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

In [33]:

```
print(y)
```

```
114141    0
237943    0
9471      0
242283    0
27621     0
         ..
279863    1
```

```
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

In [34]:
```
#step 2 divide X nd Y further into training and testing

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.25, stratify= y, random_state= 2)
```

In [35]:
```
print(X_train.shape)
```

```
(738, 30)
```

In [36]:
```
print(X_test.shape)
```

```
(246, 30)
```

In [37]:
```
print(y_train.shape)
```

```
(738,)
```

Applying model

In [38]:
```
ml_model = LogisticRegression()
```

In [39]:
```
ml_model.fit(X_train, y_train)
```

Out[39]:
```
LogisticRegression()
```

Evaluation of accuracy

```
In [40]: #accuracy score of training data
Train_prediction=ml_model.predict(X_train)
Acc_train= accuracy_score(Train_prediction, y_train)
```

```
In [41]: #check accuracy score
print('Accuracy score of training data is', Acc_train)
```

```
Accuracy score of training data is 0.9200542005420054
```

```
In [42]: #accuracy score of training data
Test_prediction=ml_model.predict(X_test)
Acc_test= accuracy_score(Test_prediction, y_test)
```

```
In [43]: print('Accuracy score of test data is', Acc_test)
```

```
Accuracy score of test data is 0.9065040650406504
```

Confusion matrix

```
In [44]: print (confusion_matrix(Test_prediction, y_test))
```

```
[[122  22]
 [  1 101]]
```

```
In [45]: print (confusion_matrix(Train_prediction, y_train))
```

```
[[352  42]
 [ 17 327]]
```

```
In [46]: print (classification_report(Test_prediction, y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.85      0.91       144
           1       0.82      0.99      0.90       102
```

```
    accuracy                           0.91       246
   macro avg       0.91      0.92      0.91       246
weighted avg       0.92      0.91      0.91       246
```

In [ ]:

```

```