CREDIT CARD FRAUD DETECTION

IMPORT THE REQUIRED LIBRARIES

In [34]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

Open the dataset

In [35]:
```
data=pd.read_csv('creditcard.csv')
```

Check the top 5 rows

In [36]:
```
data.head()
```

Out[36]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

Check the dataset shape

In [5]:
```
data.shape
```

Out[5]:
```
(284807, 31)
```

In [6]:
```
data.describe()
```

Out[6]:

	Time	V1	V2	V3	V4	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e

8 rows × 31 columns

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

Out[7]:

	Time	V1	V2	V3	V4	V5	V6	V7
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006

5 rows × 31 columns

CLASSIFICATION(F=Fraud,NF=Not fraud)

In [8]:
```
F=data.loc[data["Class"]==1]
NF=data.loc[data["Class"]==0]
```

In [9]:
```
F
```

Out[9]:

	Time	V1	V2	V3	V4	V5	V6	V7
541	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387
623	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574
4920	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320
6108	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197
6329	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445
...	...	...	...	...	...	...	...	...
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050

492 rows × 31 columns

In [10]:
```
F.count()
```

Out[10]:
```
Time      492
V1        492
V2        492
V3        492
V4        492
V5        492
V6        492
V7        492
V8        492
V9        492
V10       492
V11       492
V12       492
V13       492
V14       492
V15       492
V16       492
V17       492
V18       492
V19       492
V20       492
V21       492
V22       492
V23       492
V24       492
V25       492
V26       492
V27       492
V28       492
Amount    492
Class     492
dtype: int64
```

In [11]:
```
len(F)
```

Out[11]: 492

In [12]:
```
len(NF)
```

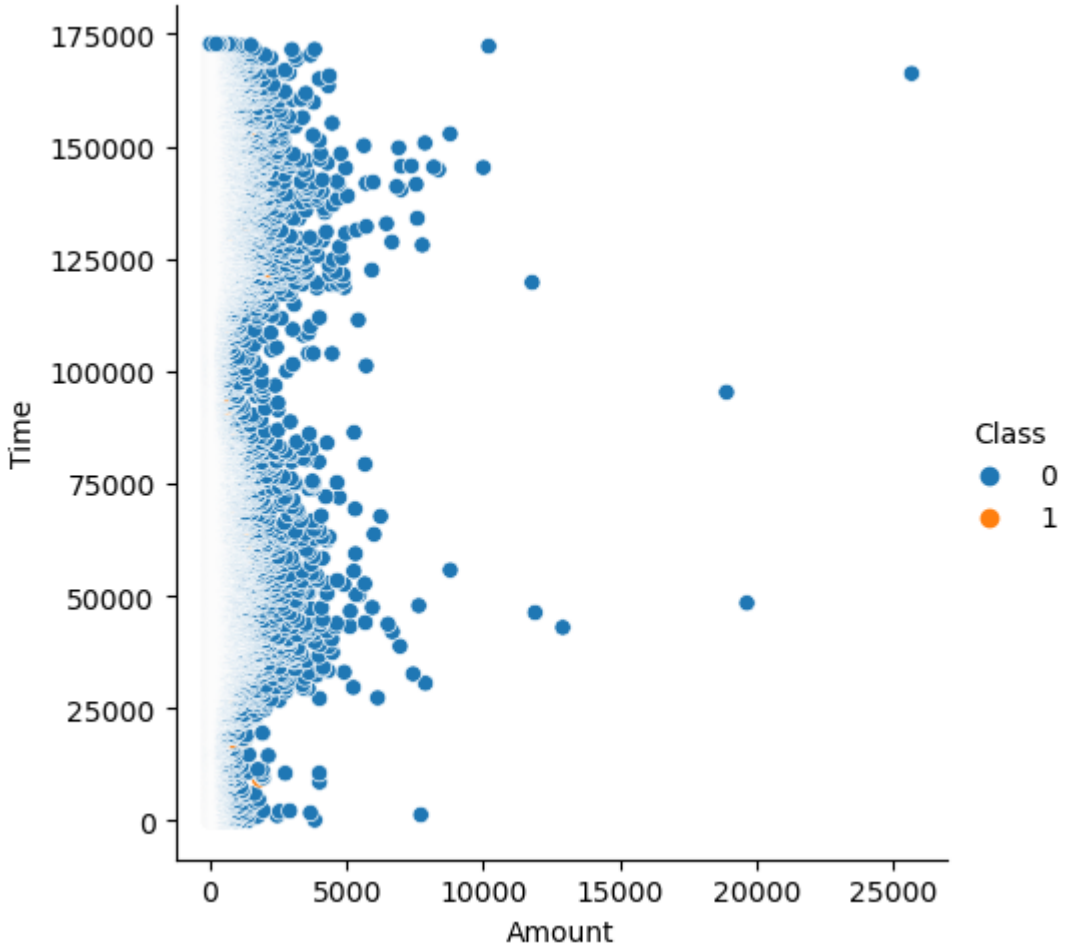
Out[12]: 284315

This step plots the *Seaborn Relational Chart* for the data

In [13]:
```
sns.relplot(x="Amount",y="Time",hue="Class",data=data)
```

```
C:\Users\irfu0\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x2b26a58d790>



Logistic regression Technique

In [14]:
```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

In [15]:
```
x=data.iloc[:,:-1]
y=data["Class"]
```

In [16]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.40)
```

In [17]:
```
clf=linear_model.LogisticRegression(C=1e5)
```

In [63]:
```
clf.fit(x_train,y_train)
```

```
C:\Users\irfu0\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[63]:
```
LogisticRegression
LogisticRegression(C=100000.0)
```

In [50]:
```
y_pred=np.array(clf.predict(x_test))
```

In [51]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [52]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[113683     56]
 [    50    134]]
```

In [53]:
```
accuracy=accuracy_score(y_test,y_pred)
```

In [54]:
```
accuracy
```

Out[54]:
```
0.999069546974711
```

In [55]:
```
from sklearn.metrics import classification_report
```

In [56]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113739
           1       0.71      0.73      0.72       184

    accuracy                           1.00    113923
   macro avg       0.85      0.86      0.86    113923
weighted avg       1.00      1.00      1.00    113923
```

Random forest Technique

In [64]:
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=5,random_state=40)
classifier.fit(x_train,y_train)
```

Out[64]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=5, random_state=40)
```

In [58]:
```
y_pred=classifier.predict(x_test)
```

In [59]:
```
accuracy_score(y_test,y_pred)
```

Out[59]:
```
0.9995786627810012
```

In [60]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113739
           1       0.94      0.79      0.86       184

    accuracy                           1.00    113923
   macro avg       0.97      0.89      0.93    113923
weighted avg       1.00      1.00      1.00    113923
```

In [ ]:

In [ ]: