CREDIT CARD FRAUD DETECTION

In [1]:
```
#importing libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
#Load the data

cf = pd.read_csv("C:/Users/yuvak/OneDrive/Pictures/creditcard fraud detection/credit
```

In [3]:
```
cf.head()
```

Out[3]:

	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	0.363
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817

5 rows × 31 columns

In [4]:
```
len(cf)
```

Out[4]:
```
284807
```

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

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

In [6]:
```
cf.index
```

Out[6]:
```
RangeIndex(start=0, stop=284807, step=1)
```

In [7]:
```
cf.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]:
```
cf.dtypes
```

Out[8]:
```
Time      float64
V1        float64
V2        float64
V3        float64
V4        float64
V5        float64
V6        float64
V7        float64
V8        float64
V9        float64
V10       float64
V11       float64
V12       float64
V13       float64
V14       float64
V15       float64
V16       float64
V17       float64
V18       float64
V19       float64
V20       float64
V21       float64
V22       float64
V23       float64
V24       float64
V25       float64
V26       float64
V27       float64
V28       float64
Amount    float64
Class       int64
dtype: object
```

In [9]:
```
cf.describe()
```

Out[9]:

	Time	V1	V2	V3	V4	V5	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2

	Time	V1	V2	V3	V4	V5	
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7

8 rows × 31 columns

In [10]:
```
fraud = cf.loc[cf['Class'] == 1]
normal = cf.loc[cf['Class'] == 0]
```

In [11]:
```
fraud.count()
```

Out[11]:
```
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
```

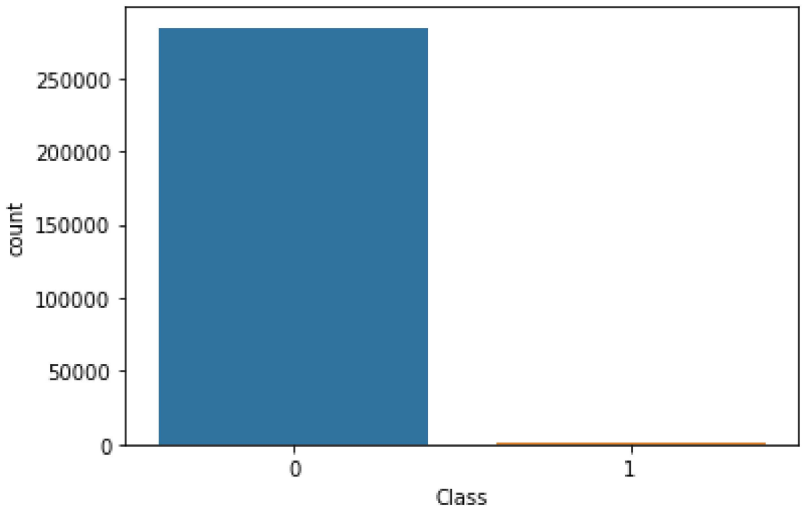
In [12]:
```
normal.count()
```

Out[12]:
```
Time      284315
V1        284315
V2        284315
V3        284315
V4        284315
V5        284315
V6        284315
V7        284315
V8        284315
V9        284315
V10       284315
V11       284315
V12       284315
```

```
V13       284315
V14       284315
V15       284315
V16       284315
V17       284315
V18       284315
V19       284315
V20       284315
V21       284315
V22       284315
V23       284315
V24       284315
V25       284315
V26       284315
V27       284315
V28       284315
Amount    284315
Class     284315
dtype: int64
```

In [13]:
```
sns.countplot(x = 'Class', data = cf)
```

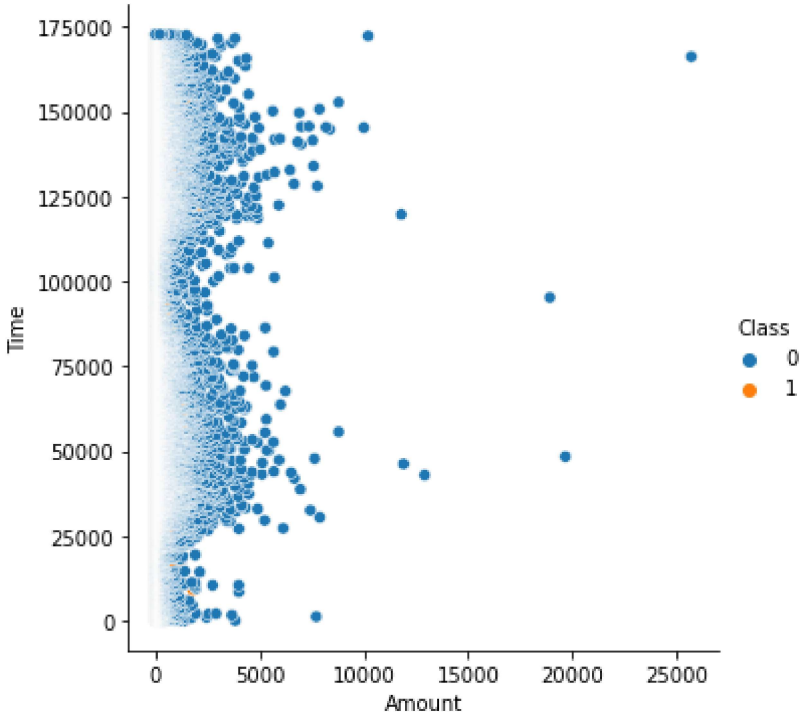
Out[13]: <AxesSubplot:xlabel='Class', ylabel='count'>



In [14]:
```
#relationalplots

sns.relplot(x = 'Amount', y = 'Time', hue = 'Class', data = cf)
```

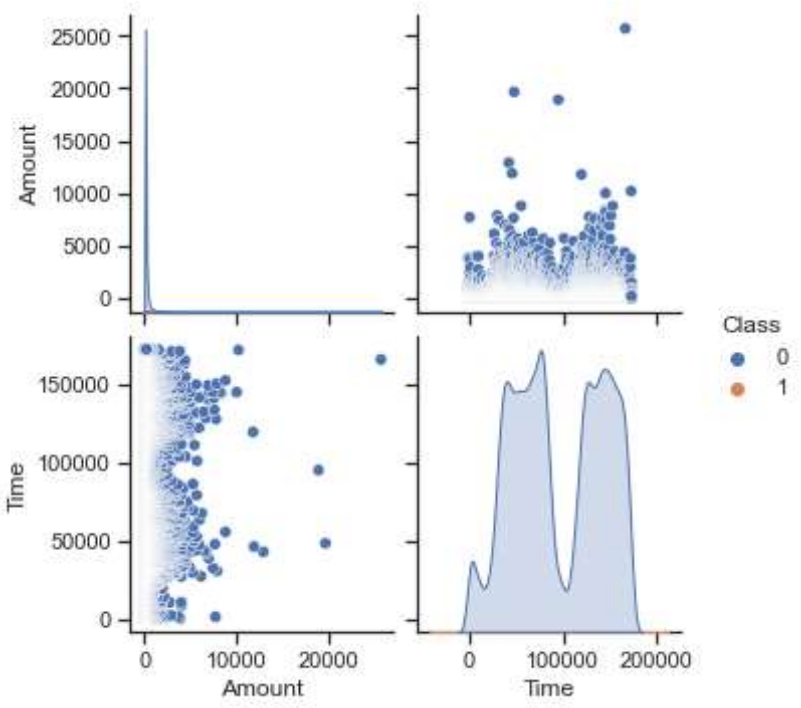
Out[14]: <seaborn.axisgrid.FacetGrid at 0x1d017993220>



In [15]:
```
#pairplot

sns.set(style="ticks")
sns.pairplot(cf, hue="Class", vars=["Amount", "Time"])
```

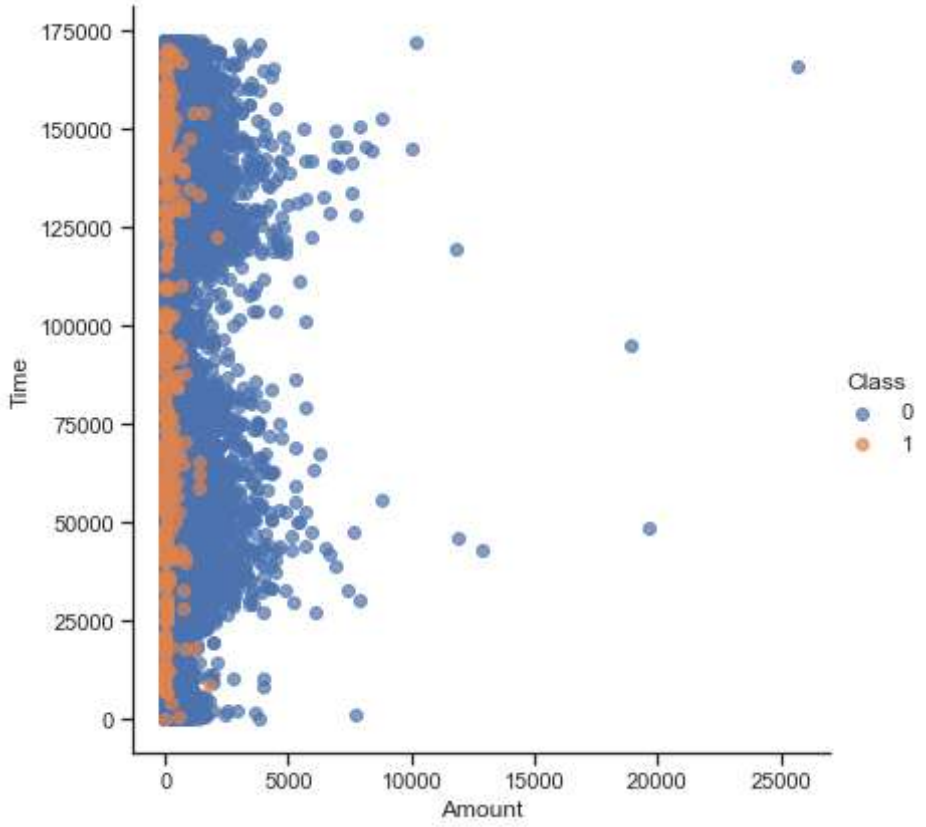
Out[15]: <seaborn.axisgrid.PairGrid at 0x1d0173368b0>



In [16]:
```
#FacetGrid plot

g = sns.FacetGrid(cf, hue="Class", height=6)
g.map(plt.scatter, "Amount", "Time", alpha=0.7)
g.add_legend()
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x1d017a06760>



In [17]:
```
cf.isna()
```

Out[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
284803	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
284804	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
284805	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
284806	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False

284807 rows × 31 columns

In [18]:
```
cf.isna().sum()
```

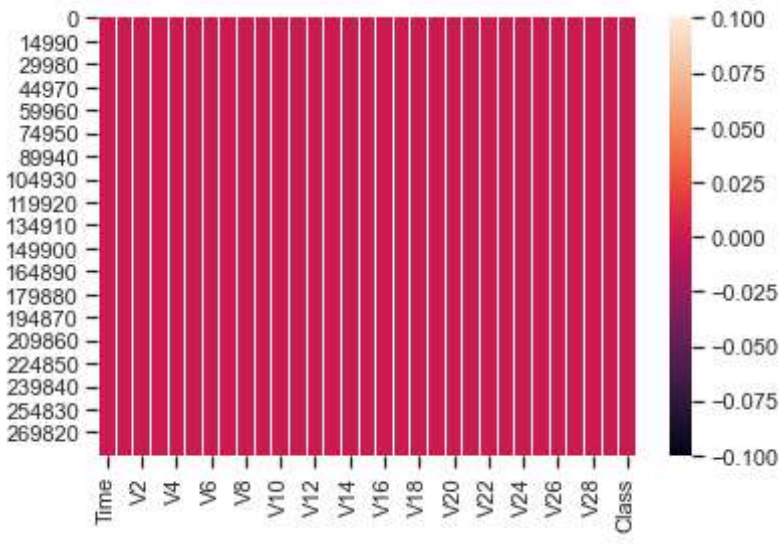
Out[18]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
```

```
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
```

No null values are presented

In [19]:
```
sns.heatmap(cf.isna())  #correlation if any nulls are occurs!
```

Out[19]: <AxesSubplot:>



In [20]:
```
#test_train_split

from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

In [21]:
```
#separate features for the model developing

x = cf.iloc[:,:-1]
y = cf['Class']
```

In [22]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.35)
```

In [23]:
```
#Logistic regression model

clf = linear_model.LogisticRegression(C = 1e5)
```

In [24]:
```
#fit the classification model

clf.fit(x_train, y_train)
```

```
C:\Users\yuvak\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: Co
nvergenceWarning: 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 in:
    https://scikit-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-regression
  n_iter_i = _check_optimize_result(
```

Out[24]:
```
LogisticRegression

LogisticRegression(C=100000.0)
```

In [25]:
```
#consistency and ease of comparison through numpy array

y_pred = np.array(clf.predict(x_test))
y = np.array(y_test)
```

In [26]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [27]:
```
#confusion_matrix

print(confusion_matrix(y_test, y_pred))
```

```
[[99459    44]
 [   57   123]]
```

In [28]:
```
#accuarcy_score

print(accuracy_score(y_test, y_pred))
```

```
0.9989867881183351
```

In [29]:
```
#classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99503
           1       0.74      0.68      0.71       180

    accuracy                           1.00     99683
   macro avg       0.87      0.84      0.85     99683
weighted avg       1.00      1.00      1.00     99683
```

In [ ]: