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

In [2]:
```
dataframe = pd.read_csv('dataset_lab04.csv')
```

In [3]:
```
dataframe.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
```

1. How many rows and columns this dataframe has? Print this information.

```
def lab04_task1_2019_1_60_134():
    print('Number of Rows:',dataframe.shape[0])
    print('Number of Columns:',dataframe.shape[1])
```

In [5]:
```
lab04_task1_2019_1_60_134()
```

```
Number of Rows: 284807
Number of Columns: 31
```

2. Describe (numerical summary) the time and amount column. Print this information.

If we look at the count of Time and ammount we can see that there is 284807 rows in both column and there is no null value because the count of rows is equal to the dataframe total rows number.

The mean of Time is 94813.859575 and the mean of Amount is 88.349619.

Stander devision for the observation of Time is 47488.145955 and amount is 250.120109.

Minumum(min) of the values for Time and Amount observation are 0.000000. Maximum(max) of the values for Time is 172792.000000 and for Amount is 25691.160000.

For Time 25th percentiles is 54201.500000,50th percentiles is 84692.000000, and 75th percentiles is 139320.500000. For Amount 25th percentiles is 5.600000,50th percentiles is 22.000000, and 75th percentiles is 77.165000.

In [6]:
```
def lab04_task2_2019_1_60_134():
    print(dataframe[['Time','Amount']].describe())
```

In [7]:
```
lab04_task2_2019_1_60_134()
```

```
                Time         Amount
count  284807.000000  284807.000000
mean    94813.859575      88.349619
std     47488.145955     250.120109
min         0.000000       0.000000
25%     54201.500000       5.600000
50%     84692.000000      22.000000
75%    139320.500000      77.165000
max    172792.000000   25691.160000
```

3. There are 31 columns in the dataset. Compute some statistical measures like mean, median, standard deviation, variance using Pandas Function for at least two columns. Print this information.

In [12]:
```
def lab04_task3_2019_1_60_134():
    dataframe.columns = ['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']

    Time = dataframe[['Time']]
    print(type(Time))
    print("Mean: " , Time.mean())
    print("Median: " , Time.median())
    print("Standard Deviation: ", Time.std())
    print("Variance: ", Time.var())
    print("\n")

    V1 = dataframe[['V1']]
    print(type(V1))
    print("Mean: " , V1.mean())
    print("Median: " , V1.median())
    print("Standard Deviation: ", V1.std())
    print("Variance: ", V1.var())
    print("\n")

    Amount = dataframe[['Amount']]
    print(type(Amount))
    print("Mean: " , Amount.mean())
    print("Median: " , Amount.median())
    print("Standard Deviation: ", Amount.std())
    print("Variance: ", Amount.var())
    print("\n")

    Class = dataframe[['Class']]
    print(type(Class))
    print("Mean: " , Class.mean())
    print("Median: " , Class.median())
    print("Standard Deviation: ", Class.std())
    print("Variance: ", Class.var())
```

In [13]:
```
lab04_task3_2019_1_60_134()
```

```
<class 'pandas.core.frame.DataFrame'>
Mean:  Time    94813.859575
dtype: float64
Median:  Time    84692.0
dtype: float64
Standard Deviation:  Time    47488.145955
dtype: float64
Variance:  Time    2.255124e+09
dtype: float64


<class 'pandas.core.frame.DataFrame'>
Mean:  V1    3.918649e-15
dtype: float64
Median:  V1    0.018109
dtype: float64
Standard Deviation:  V1    1.958696
dtype: float64
Variance:  V1    3.836489
dtype: float64


<class 'pandas.core.frame.DataFrame'>
Mean:  Amount    88.349619
dtype: float64
Median:  Amount    22.0
dtype: float64
Standard Deviation:  Amount    250.120109
dtype: float64
Variance:  Amount    62560.069046
dtype: float64


<class 'pandas.core.frame.DataFrame'>
Mean:  Class    0.001727
dtype: float64
Median:  Class    0.0
dtype: float64
Standard Deviation:  Class    0.041527
dtype: float64
Variance:  Class    0.001725
dtype: float64
```

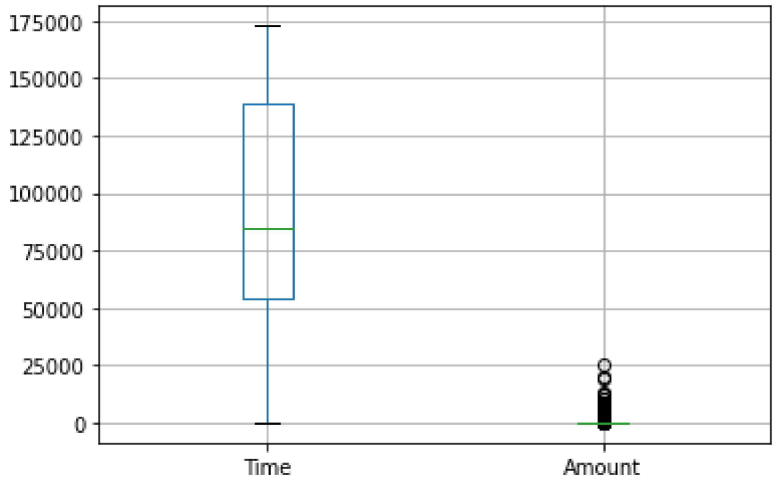
4. Show the Box Plot of Time and Amount column. Also print the value of Q1, Median, Q3, IQR. Are there any outliers? Explain your answer and print it.

For Time :

In [19]:
```
def lab04_task4_T_2019_1_60_134():
    print("Details for Time")
    time= dataframe[['Time']]
    dataframe.boxplot(column = ['Time','Amount'])
    Q1= time.quantile(0.25)
    Q3= time.quantile(0.75)
    print("Lower Quartile(Q1): " , Q1)
    print("Median: " , time.median())
    print("Upper Quartile(Q3): " , Q3)
    IQR = Q3-Q1
    print("IQR: " , IQR)
    print("Lower Bound: ", Q1-(IQR*1.5))
    print("Upper Bound: ", Q3+(IQR*1.5))
    if ((Q1-(IQR*1.5))>time.min()).all():
         print("There is Outliers")
    elif ((Q3+(IQR*1.5))<time.max()).all():
          print("There is Outliers")
    else:
          print("There is no Outliers")
```

In [20]:
```
lab04_task4_T_2019_1_60_134()
```

```
Details for Time
Lower Quartile(Q1):  Time    54201.5
Name: 0.25, dtype: float64
Median:  Time    84692.0
dtype: float64
Upper Quartile(Q3):  Time    139320.5
Name: 0.75, dtype: float64
IQR:  Time    85119.0
dtype: float64
Lower Bound:  Time   -73477.0
dtype: float64
Upper Bound:  Time    266999.0
dtype: float64
There is no Outliers
```

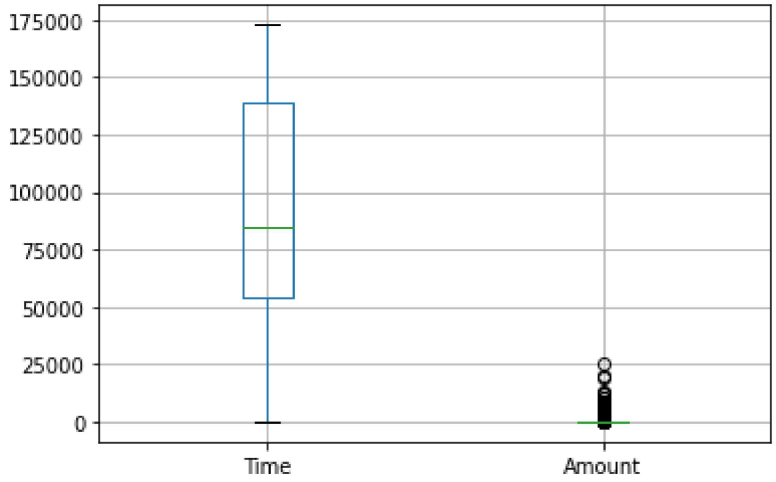


For Amount :

In [21]:
```
def lab04_task4_A_2019_1_60_134():
    print("Details for Amount")
    Amount= dataframe[['Amount']]
    dataframe.boxplot(column = ['Time','Amount'])
    Q1= Amount.quantile(0.25)
    Q3= Amount.quantile(0.75)
    print("Lower Quartile(Q1): " , Q1)
    print("Median: " , Amount.median())
    print("Upper Quartile(Q3): " , Q3)
    IQR = Q3-Q1
    print("IQR: " , IQR)
    print("Lower Bound: ", Q1-(IQR*1.5))
    print("Upper Bound: ", Q3+(IQR*1.5))
    if ((Q1-(IQR*1.5))>Amount.min()).all():
         print("There is Outliers")
    elif ((Q3+(IQR*1.5))<Amount.max()).all():
          print("There is Outliers")
    else:
          print("There is no Outliers")
```

In [22]:
```
lab04_task4_A_2019_1_60_134()
```

```
Details for Amount
Lower Quartile(Q1):  Amount    5.6
Name: 0.25, dtype: float64
Median:  Amount    22.0
dtype: float64
Upper Quartile(Q3):  Amount    77.165
Name: 0.75, dtype: float64
IQR:  Amount    71.565
dtype: float64
Lower Bound:  Amount   -101.7475
dtype: float64
Upper Bound:  Amount    184.5125
dtype: float64
There is Outliers
```



5. Show the Histogram of Time and Amount column. Print the value of the Skewness and Kurtosis using appropriate Pandas functions.

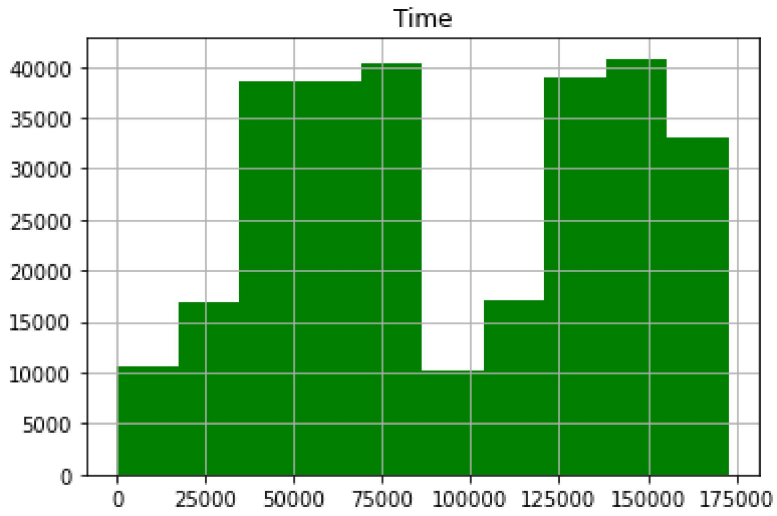
Comment on the type of the data distribution and print it.

For Time :

In [34]:
```
def lab04_task5_T_2019_1_60_134():
    df_hist = dataframe[['Time']]
    df_hist.hist(column = ['Time'], bins = 10, color='Green')
    print("Skewness: " ,df_hist.skew())
    print("Kurtosis: " ,df_hist.kurt())
```

In [35]:
```
lab04_task5_T_2019_1_60_134()
```

```
Skewness:  Time   -0.035568
dtype: float64
Kurtosis:  Time   -1.29353
dtype: float64
```

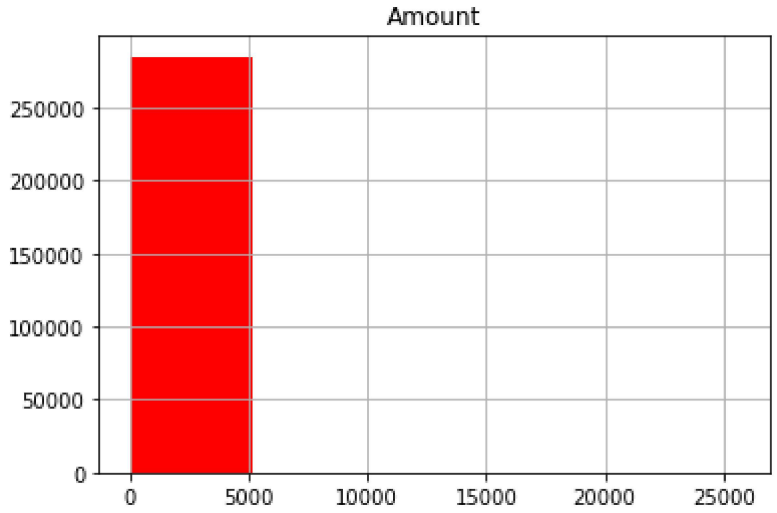


For Amount :

In [36]:
```
def lab04_task5_A_2019_1_60_134():
    df_hist1 = dataframe[['Amount']]
    df_hist1.hist(column = ['Amount'], bins = 5, color='Red')
    print("Skewness: " ,df_hist1.skew())
    print("Kurtosis: " ,df_hist1.kurt())
```

In [37]:
```
lab04_task5_A_2019_1_60_134()
```

```
Skewness:  Amount    16.977724
dtype: float64
Kurtosis:  Amount    845.092646
dtype: float64
```



6. Find the percentage of records with class value = 0 (Non-Fraudulent) and class value = 1 (Fraudulent).Print this information.

In [23]:
```
def lab04_task6_2019_1_60_134():
    class0 = len(dataframe[dataframe['Class']==0])*100
    class1 = len(dataframe[dataframe['Class']==1])*100

    print("Percentage of class value 0 :",class0/len(dataframe['Class']))
    print("Percentage of class value 1 :",class1/len(dataframe['Class']))
```

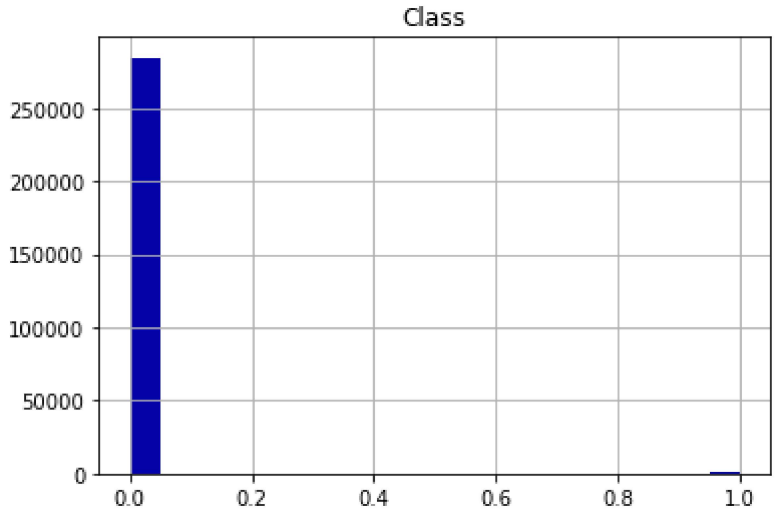
In [38]:
```
lab04_task6_2019_1_60_134()
```

```
Percentage of class value 0 : 99.827251436938
Percentage of class value 1 : 0.1727485630620034
```

7. Show the result you have got in 6 using a Histogram.

In [42]:
```
def lab04_task7_2019_1_60_134():
    df_histo = dataframe[['Class']]
    df_histo.hist(column = ['Class'],color='#0504aa', bins = 'auto')
```

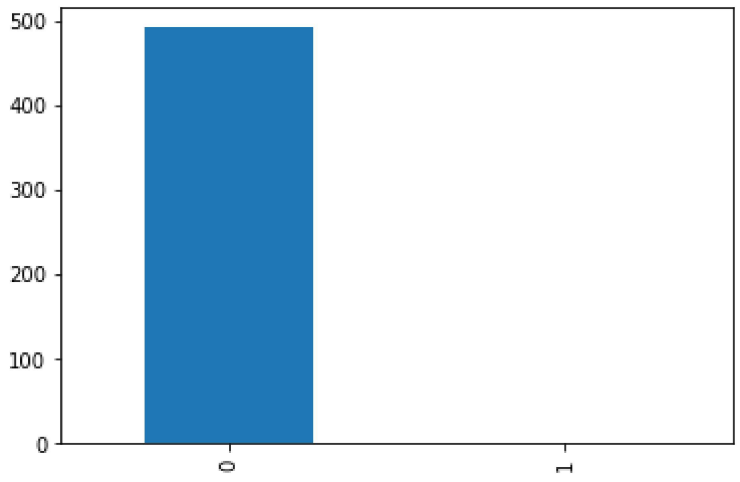
In [43]:
```
lab04_task7_2019_1_60_134()
```



8. Show the result you have got in 6 using a Bar chart. Create the bar chart on the percentage value, not on the total number of occurrences.

In [44]:
```
def lab04_task8_2019_1_60_134():
    Class = dataframe[['Class']]
    zero=(((dataframe['Class'] == 1).sum())/(Class.shape[1]))
    one=(((dataframe['Class'] == 0).sum())/(Class.shape[0]))
    pd.Series([zero,one]).plot(kind="bar")
```

In [45]:
```
lab04_task8_2019_1_60_134()
```

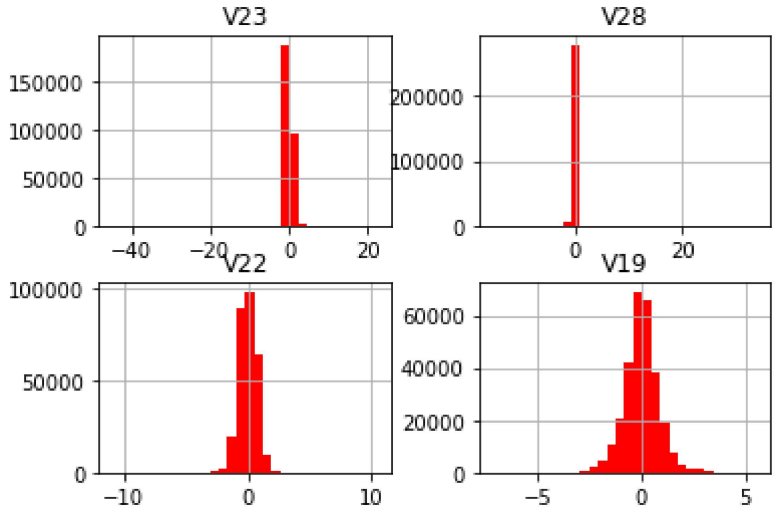


9. Show the Histrogram (data distribution) of a few other columns (your choice) showing both positive and negative skew and also leptokurtic and platykurtic data distribution. So, you should display at least four Histograms.

In [80]:
```
def lab04_task9_2019_1_60_134():
    dataframe.hist(column = ["V23", "V28", "V22", "V19"], bins = 30, color='Red')


    V23_length = dataframe[['V23']]
    print("Skewness: " , V23_length.skew()) #positive or right-skewed
    print("/n/n")

    V28_length = dataframe[['V28']]       #negative or left-skewed
    print("Skewness: " , V28_length.skew())
    print("/n/n")

    V22_length = dataframe[['V22']]
    print("Skewness: " , V22_length.skew()) # Leptokartic
    print("/n/n")

    V19_length = dataframe[['V19']]
    print("Skewness: " , V19_length.skew())  # platykurtic
    print("/n/n")
```

In [81]:
```
lab04_task9_2019_1_60_134()
```

```
Skewness:  V23   -5.87514
dtype: float64
/n/n
Skewness:  V28    11.192091
dtype: float64
/n/n
Skewness:  V22   -0.213258
dtype: float64
/n/n
Skewness:  V19    0.109192
dtype: float64
/n/n
```



10. Find the highest positive correlation among all attributes. While finding the correlation, use appropriate code, not manually. Print this information accordingly.

In [107]:
```
def lab04_task10_2019_1_60_134():
    correlation= dataframe.corr()
    positive_corr= correlation[correlation > 0]
    print(max(positive_corr))
```

In [108]:
```
lab04_task10_2019_1_60_134()
```

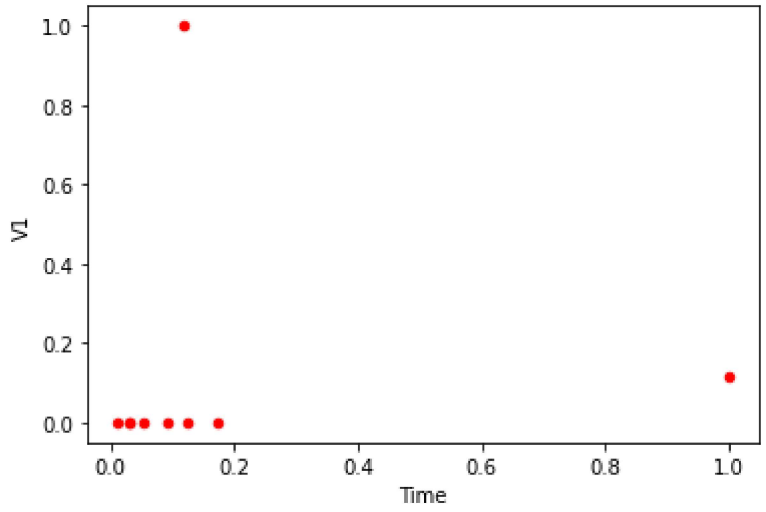
```
V9
```

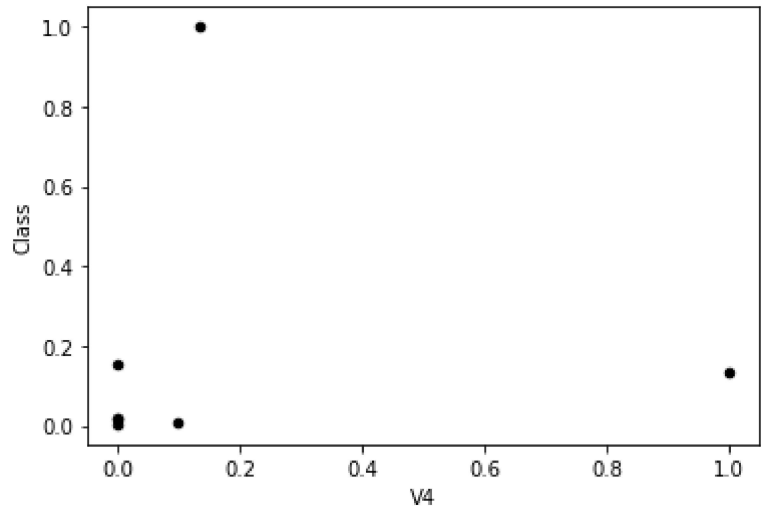
11. Support your findings in Question 10 using a Scatter Plot.

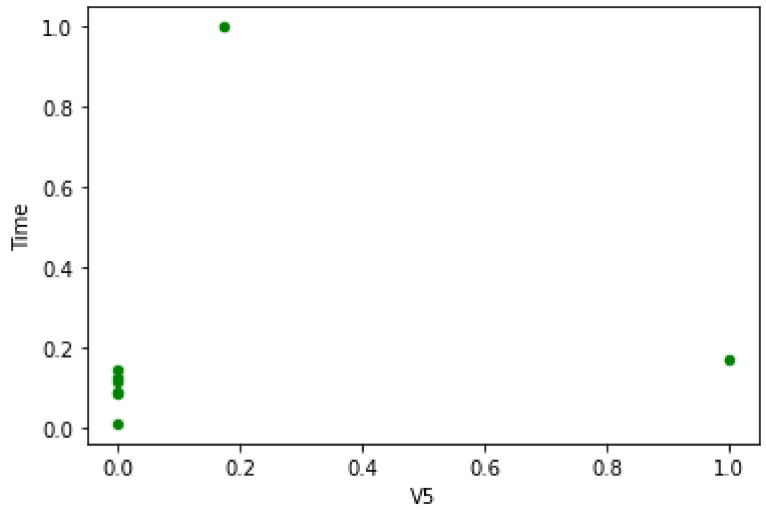
In [109]:
```
def lab04_task11_2019_1_60_134():
    correlation= dataframe.corr()
    positive_corr= correlation[correlation > 0]
    print(max(positive_corr))

    positive_corr.plot.scatter(x= 'Time', y='V1',c='Red')
    positive_corr.plot.scatter(x= 'V4', y='Class', c='Black')
    positive_corr.plot.scatter(x= 'V5', y='Time',c='Green')
    positive_corr.plot.scatter(x= 'Amount', y='Amount',c='Blue')
```

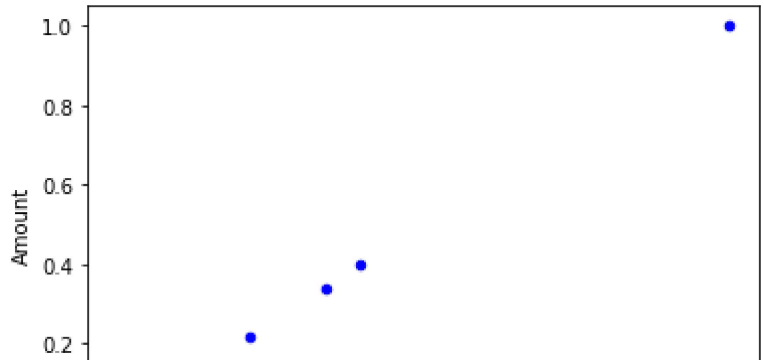
In [110]:
```
lab04_task11_2019_1_60_134()
```

V9









12. Find the highest negative correlation among all attributes. While finding the correlation, use appropriate code, not manually. Print this information accordingly.

In [111]:
```
def lab04_task12_2019_1_60_134():
    correlation= dataframe.corr()
    negative_corr= correlation[correlation < 0]
    print(min(negative_corr))
```

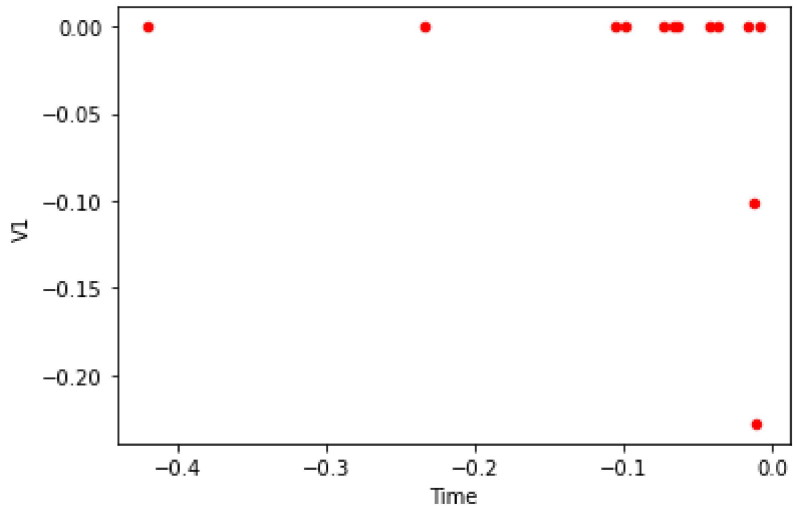
In [112]:
```
lab04_task12_2019_1_60_134()
```

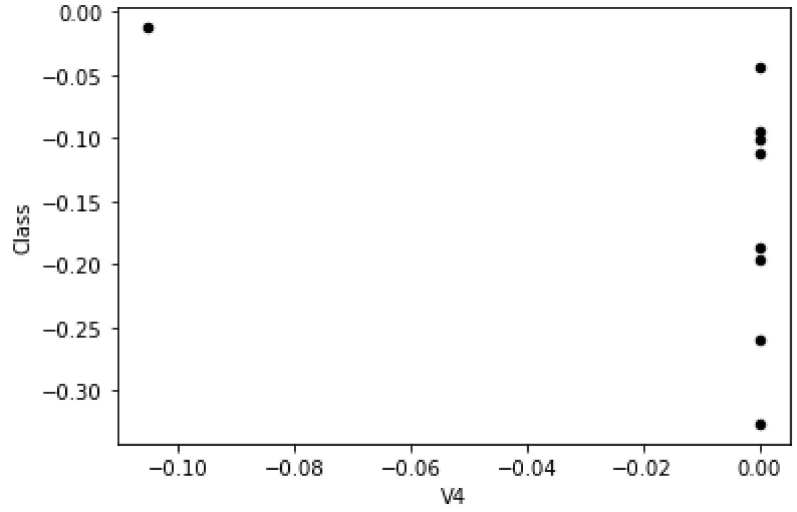
```
Amount
```

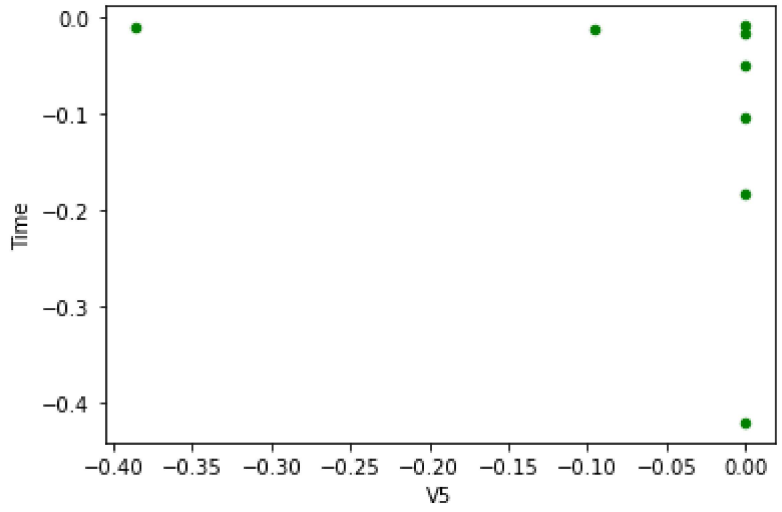
13. Support your findings in Question 12 using a Scatter Plot.

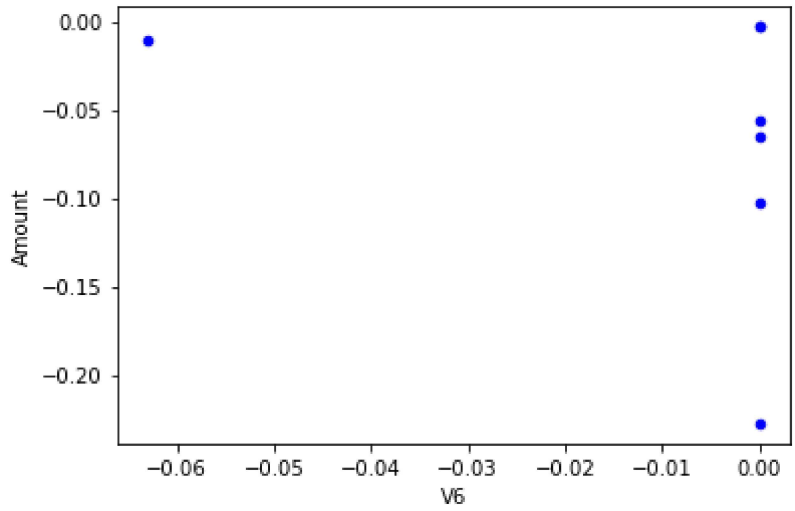
In [114]:
```
def lab04_task13_2019_1_60_134():
    correlation= dataframe.corr()
    negative_corr= correlation[correlation < 0]

    negative_corr.plot.scatter(x= 'Time', y='V1',c='Red')
    negative_corr.plot.scatter(x= 'V4', y='Class', c='Black')
    negative_corr.plot.scatter(x= 'V5', y='Time',c='Green')
    negative_corr.plot.scatter(x= 'V6', y='Amount',c='Blue')
```

In [115]:
```
lab04_task13_2019_1_60_134()
```







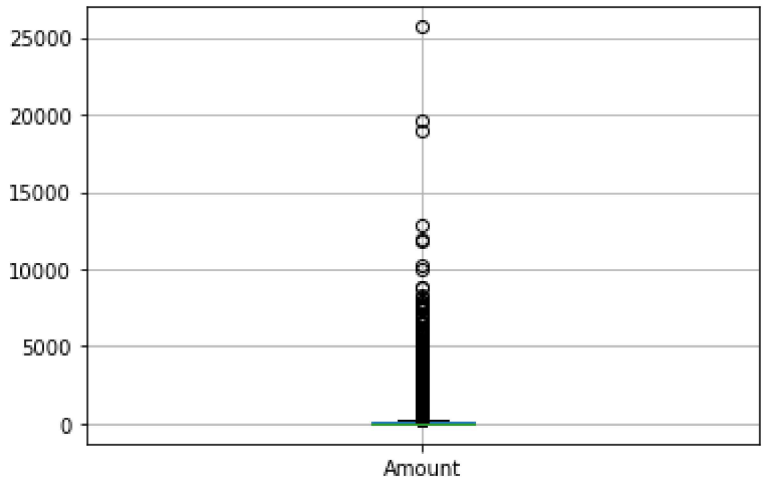


14. Create a Box Plot of the Amount Column.

In [116]:
```
def lab04_task14_2019_1_60_134():
    dataframe.boxplot(column = ['Amount'])
    print("Kurtosis: " , dataframe[['Amount']].kurt())
    print("Skewness: " ,  dataframe[['Amount']].skew())
```

In [117]:
```
lab04_task14_2019_1_60_134()
```

```
Kurtosis:  Amount    845.092646
dtype: float64
Skewness:  Amount    16.977724
dtype: float64
```

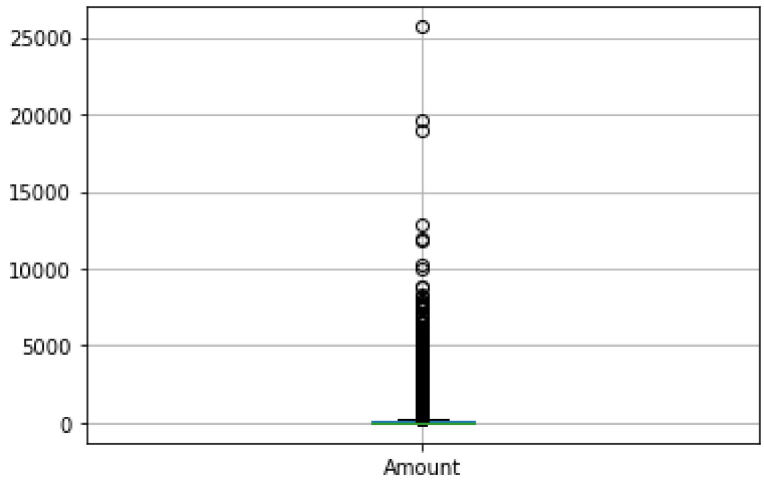


15. Now create two other box plots side by side. The first one will show the Amount column value forwhich the class value = 0 (Non-Fraudulent) and the second one will show the Amount column value for which the class value = 1 (Fraudulent). Do you find any particular pattern by just considering Amount column. Explain your answer and print it accordingly.

In [127]:
```
def lab04_task15_zero_2019_1_60_134():
    AmountForZero = dataframe[(dataframe['Class'] == 0)]
    AmountForZero.boxplot(column = ['Amount'])
    print( AmountForZero['Amount'])

def lab04_task15_one_2019_1_60_134():
    AmountForOne = dataframe[(dataframe['Class'] == 1)]
    AmountForOne.boxplot(column = ['Amount'])
    print( AmountForOne['Amount'])
```

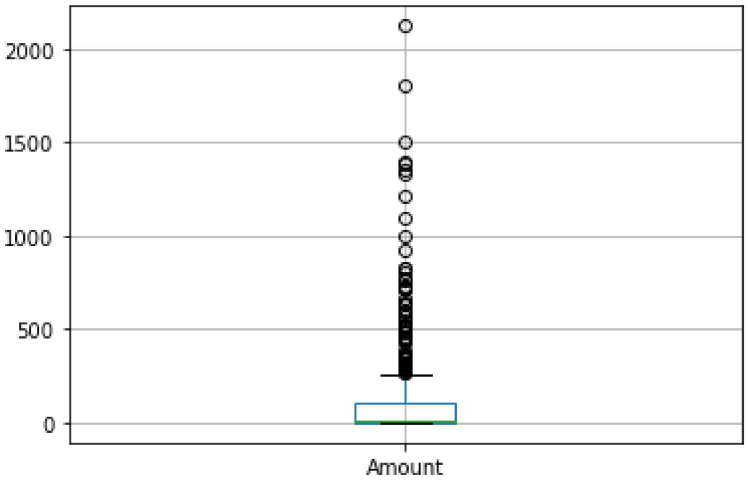
In [128]:
```
lab04_task15_zero_2019_1_60_134()
```

```
0         149.62
1           2.69
2         378.66
3         123.50
4          69.99
           ...
284802      0.77
284803     24.79
284804     67.88
284805     10.00
284806    217.00
Name: Amount, Length: 284315, dtype: float64
```



In [126]:
```
lab04_task15_one_2019_1_60_134()
```

```
541         0.00
623       529.00
4920      239.93
6108       59.00
6329        1.00
            ...
279863    390.00
280143      0.76
280149     77.89
281144    245.00
281674     42.53
Name: Amount, Length: 492, dtype: float64
```



No, I do not find any particular pattern by just considering Amount column. Both Boxplot patterns are same.