In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set()
import matplotlib.pyplot as  plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('creditcard.csv')
```

In [3]:
```
df.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.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.

5 rows × 31 columns

In [4]:
```
df.tail()
```

Out[4]:

	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	7.
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.

5 rows × 31 columns

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

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

In [6]:
```
df.columns
```

Out[6]:
```
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 [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
```

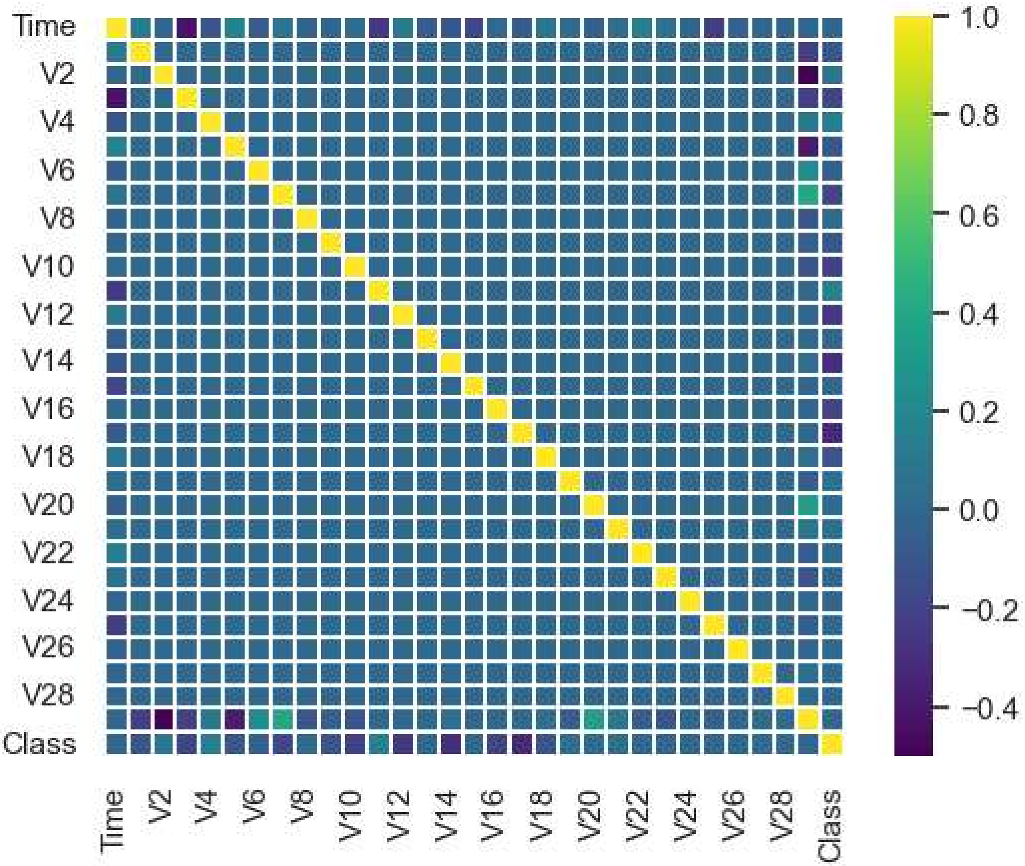
In [8]:
```
df.describe()
```

Out[8]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
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

8 rows × 31 columns

In [9]:
```
sns.heatmap(df.corr(), cmap='viridis', vmax = 1, vmin=-0.5 , square = True , l
plt.show()
```



```
In [10]: df.duplicated()
```

```
Out[10]: 0         False
         1         False
         2         False
         3         False
         4         False
                   ...
         284802    False
         284803    False
         284804    False
         284805    False
         284806    False
         Length: 284807, dtype: bool
```

No duplicate values here

```
In [11]: df.isnull().sum()/len(df)*100
```

```
Out[11]: Time      0.0
         V1        0.0
         V2        0.0
         V3        0.0
         V4        0.0
         V5        0.0
         V6        0.0
         V7        0.0
         V8        0.0
         V9        0.0
         V10       0.0
         V11       0.0
         V12       0.0
         V13       0.0
         V14       0.0
         V15       0.0
         V16       0.0
         V17       0.0
         V18       0.0
         V19       0.0
         V20       0.0
         V21       0.0
         V22       0.0
         V23       0.0
         V24       0.0
         V25       0.0
         V26       0.0
         V27       0.0
         V28       0.0
         Amount    0.0
         Class     0.0
         dtype: float64
```

lets drop Time variable

In [12]:
```
df = df.drop(['Time'], axis=1)
```

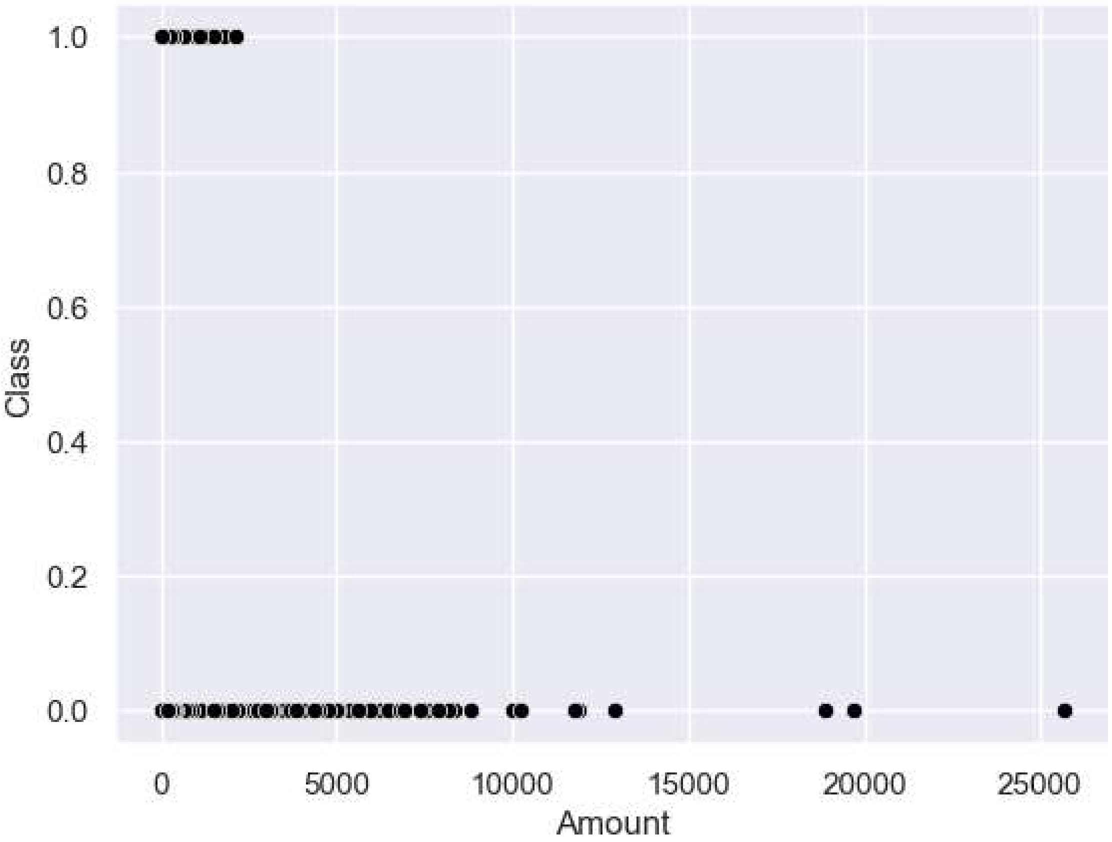
In [13]:
```
df.head()
```

Out[13]:

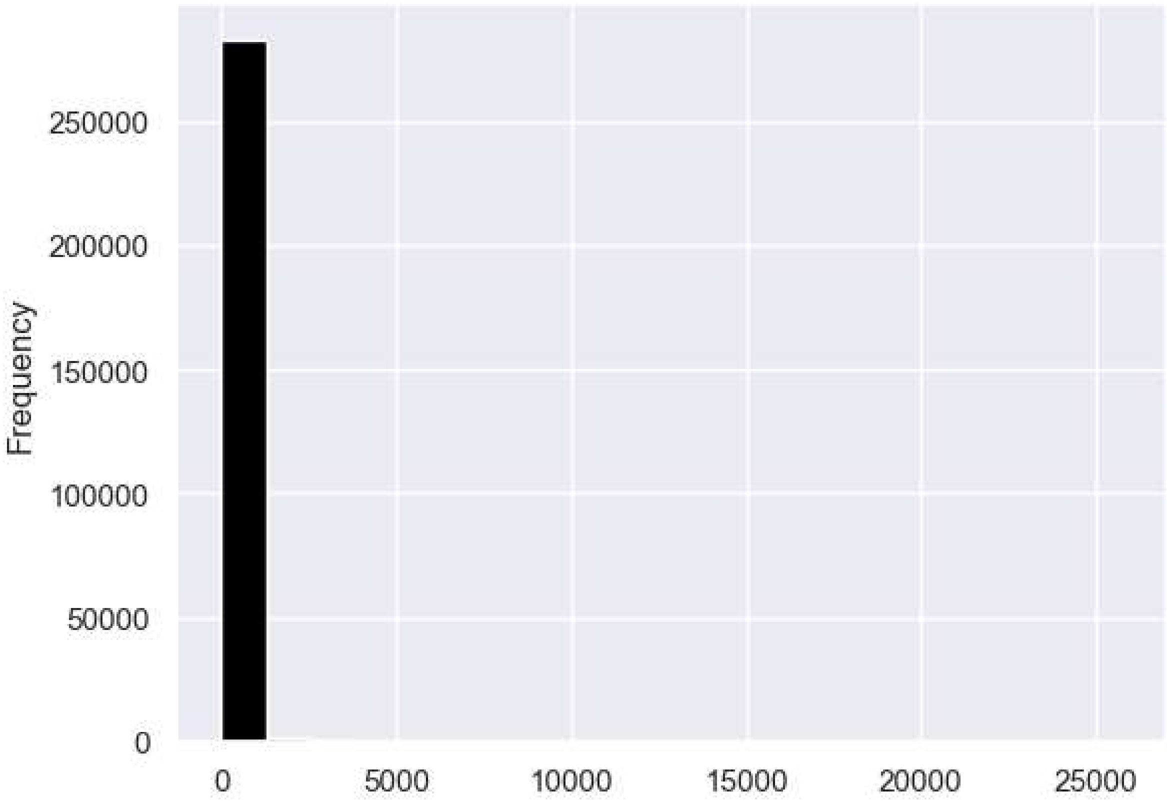
	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

5 rows × 30 columns

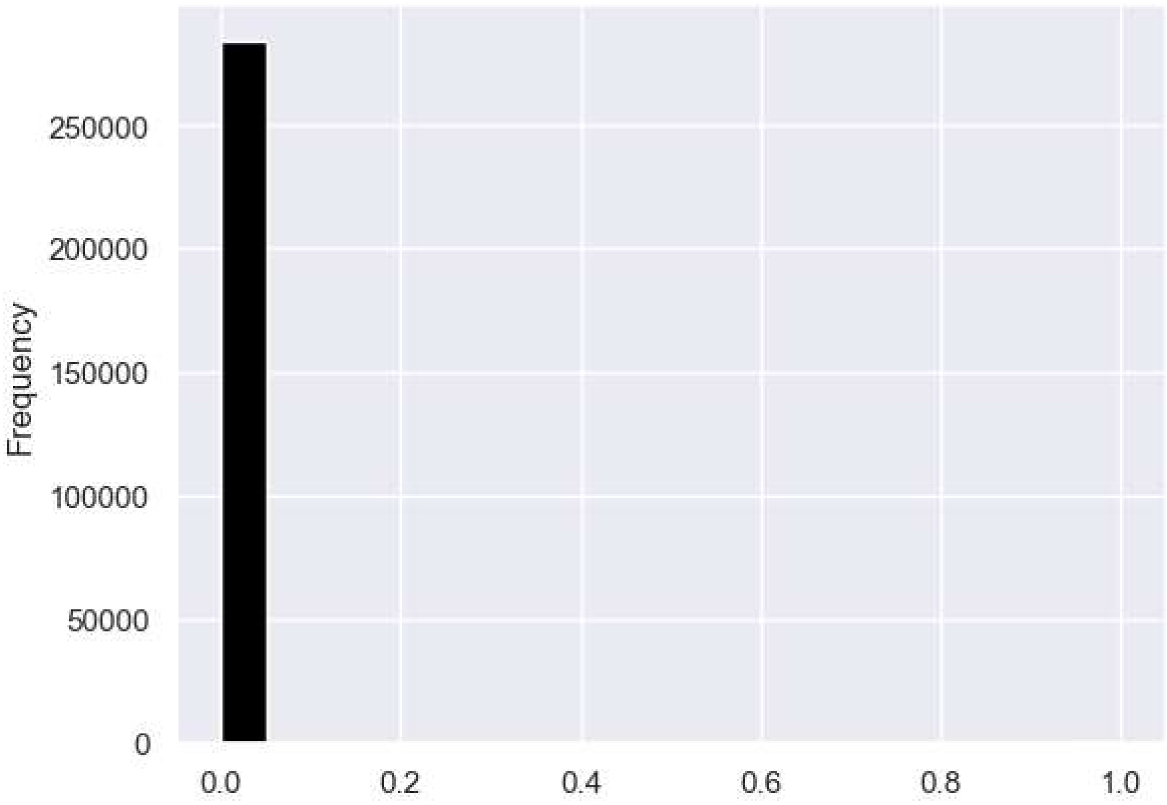
In [14]:
```
sns.scatterplot(x = df['Amount'], y = df['Class'],color = 'black')
plt.show()
```



In [15]:
```
df['Amount'].plot(kind='hist', bins=20,color='black')
plt.show()
```

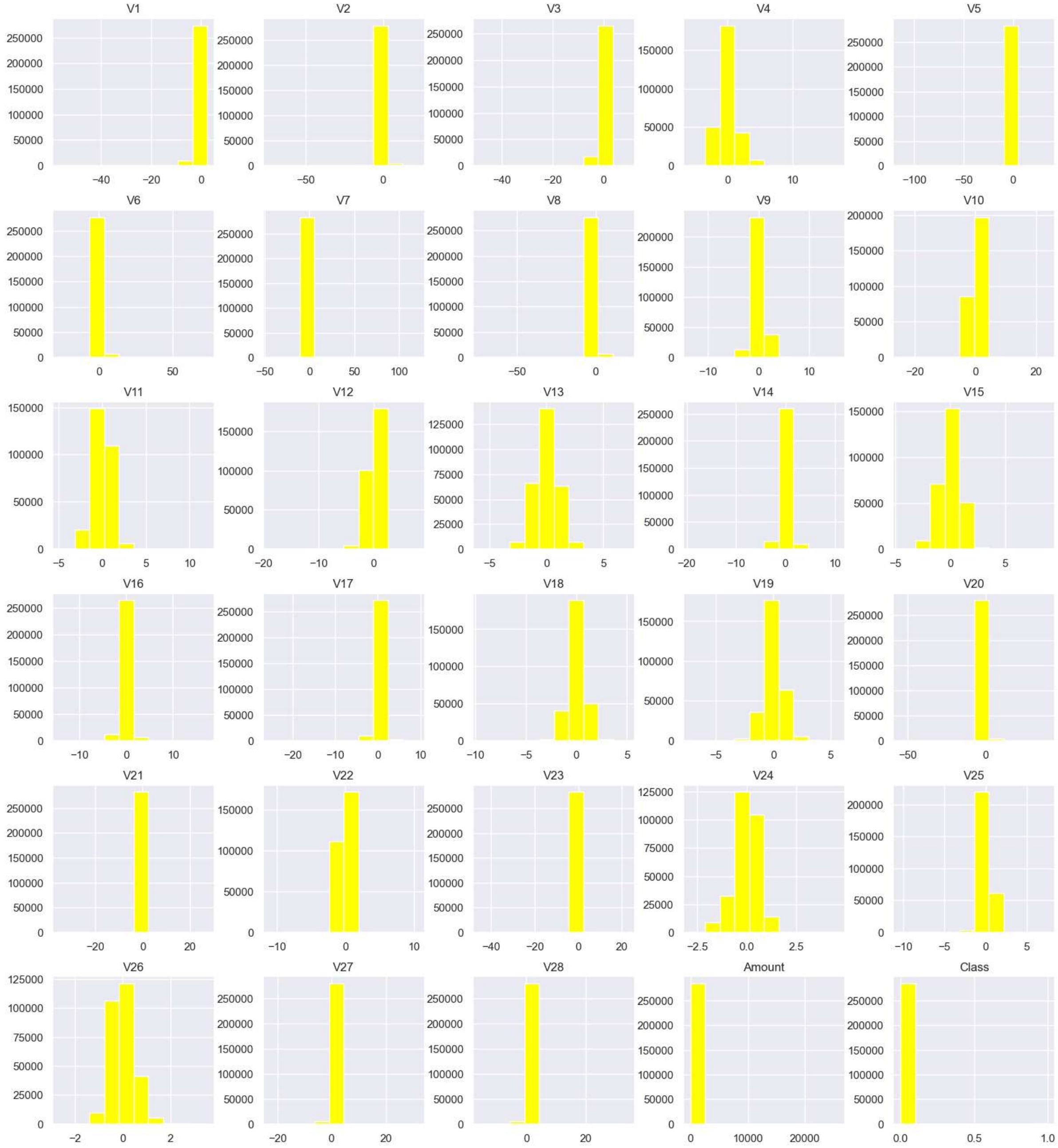


In [16]:
```
df['Class'].plot(kind='hist', bins=20,color='black')
plt.show()
```



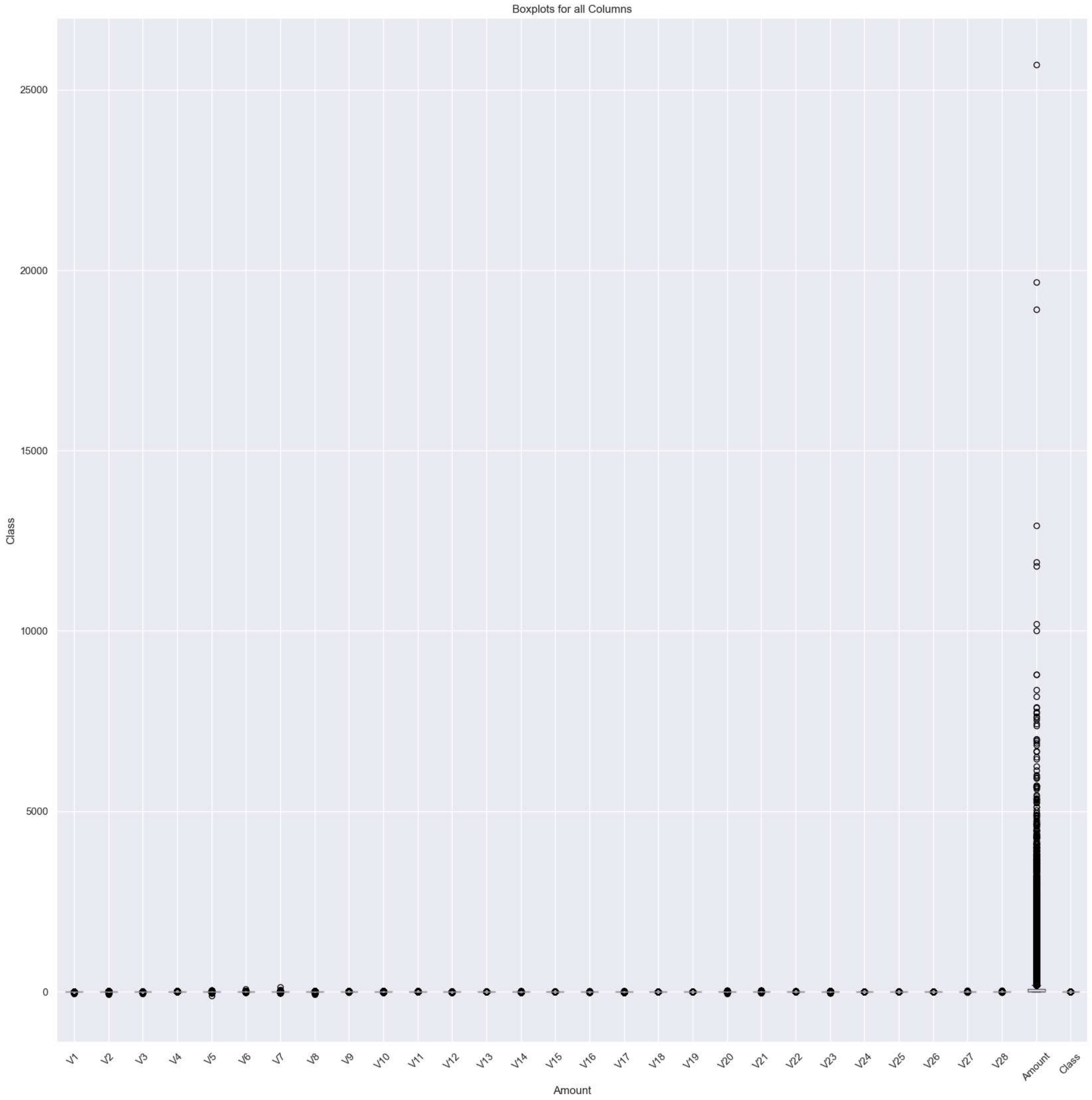
In [17]:
```
df.hist(figsize=(18,20),color='yellow')
```

Out[17]:
```
array([[<Axes: title={'center': 'V1'}>, <Axes: title={'center': 'V2'}>,
        <Axes: title={'center': 'V3'}>, <Axes: title={'center': 'V4'}>,
        <Axes: title={'center': 'V5'}>],
       [<Axes: title={'center': 'V6'}>, <Axes: title={'center': 'V7'}>,
        <Axes: title={'center': 'V8'}>, <Axes: title={'center': 'V9'}>,
        <Axes: title={'center': 'V10'}>],
       [<Axes: title={'center': 'V11'}>, <Axes: title={'center': 'V12'}>,
        <Axes: title={'center': 'V13'}>, <Axes: title={'center': 'V14'}>,
        <Axes: title={'center': 'V15'}>],
       [<Axes: title={'center': 'V16'}>, <Axes: title={'center': 'V17'}>,
        <Axes: title={'center': 'V18'}>, <Axes: title={'center': 'V19'}>,
        <Axes: title={'center': 'V20'}>],
       [<Axes: title={'center': 'V21'}>, <Axes: title={'center': 'V22'}>,
        <Axes: title={'center': 'V23'}>, <Axes: title={'center': 'V24'}>,
        <Axes: title={'center': 'V25'}>],
       [<Axes: title={'center': 'V26'}>, <Axes: title={'center': 'V27'}>,
        <Axes: title={'center': 'V28'}>,
        <Axes: title={'center': 'Amount'}>,
        <Axes: title={'center': 'Class'}>]], dtype=object)
```

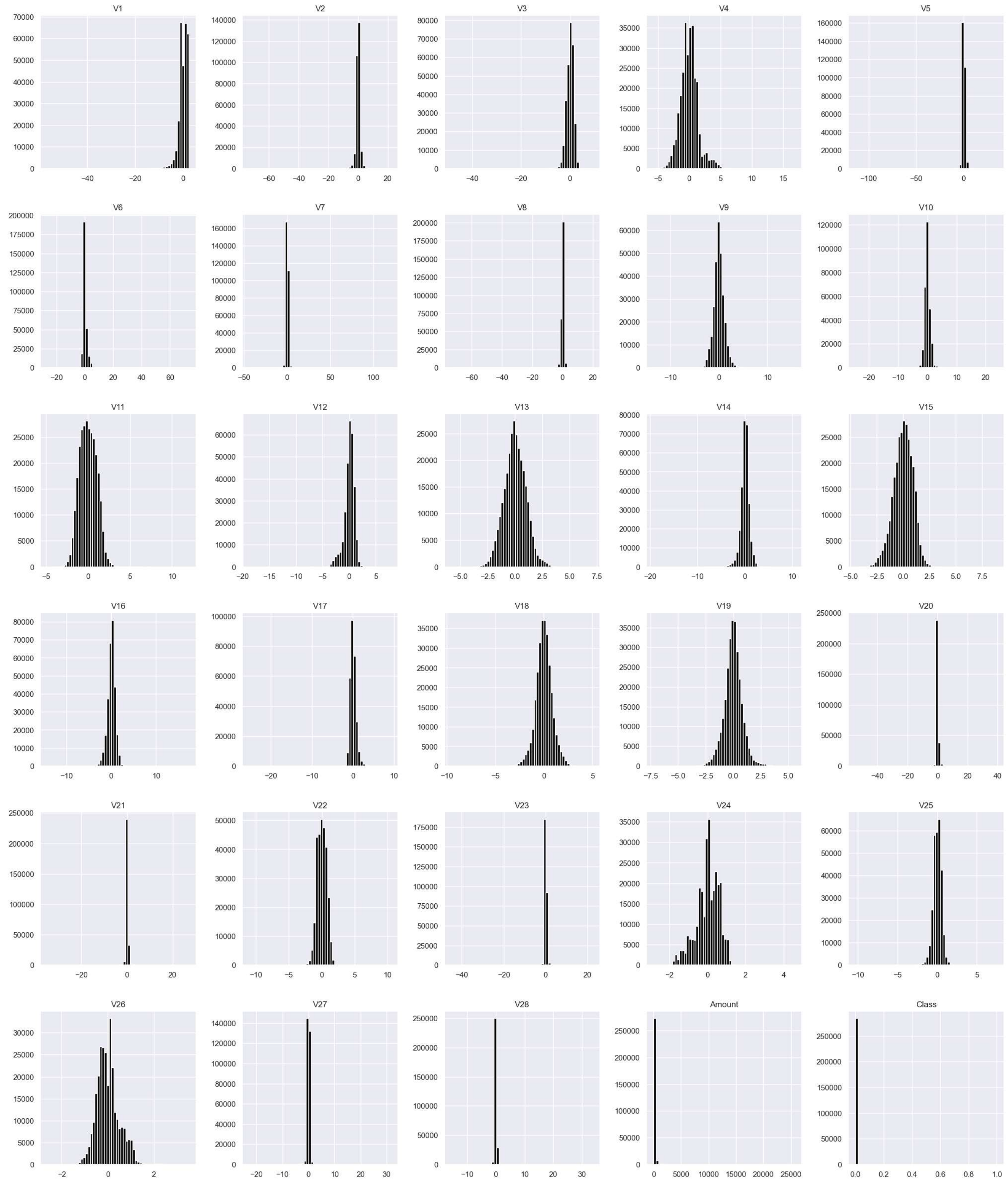

V1
250000 200000 150000 100000 50000 0
−40 −20 0
V2
250000 200000 150000 100000 50000 0
−50 0
V3
250000 200000 150000 100000 50000 0
−40 −20 0
V4
150000 100000 50000 0
0 10
V5
250000 200000 150000 100000 50000 0
−100 −50 0
V6
250000 200000 150000 100000 50000 0
0 50
V7
250000 200000 150000 100000 50000 0
−50 0 50 100
V8
250000 200000 150000 100000 50000 0
−50 0
V9
200000 150000 100000 50000 0
−10 0 10
V10
200000 150000 100000 50000 0
−20 0 20
V11
150000 100000 50000 0
−5 0 5 10
V12
150000 100000 50000 0
−20 −10 0
V13
125000 100000 75000 50000 25000 0
−5 0 5
V14
250000 200000 150000 100000 50000 0
−20 −10 0 10
V15
150000 100000 50000 0
−5 0 5
V16
250000 200000 150000 100000 50000 0
−10 0 10
V17
250000 200000 150000 100000 50000 0
−20 −10 0 10
V18
150000 100000 50000 0
−10 −5 0 5
V19
150000 100000 50000 0
−5 0 5
V20
250000 200000 150000 100000 50000 0
−50 0
V21
250000 200000 150000 100000 50000 0
−20 0 20
V22
150000 100000 50000 0
−10 0 10
V23
250000 200000 150000 100000 50000 0
−40 −20 0 20
V24
125000 100000 75000 50000 25000 0
−2.5 0.0 2.5
V25
200000 150000 100000 50000 0
−10 −5 0 5
V26
125000 100000 75000 50000 25000 0
−2 0 2
V27
250000 200000 150000 100000 50000 0
−20 0 20
V28
250000 200000 150000 100000 50000 0
0 20
Amount
250000 200000 150000 100000 50000 0
0 10000 20000
Class
250000 200000 150000 100000 50000 0
0.0 0.5 1.0

In [18]:

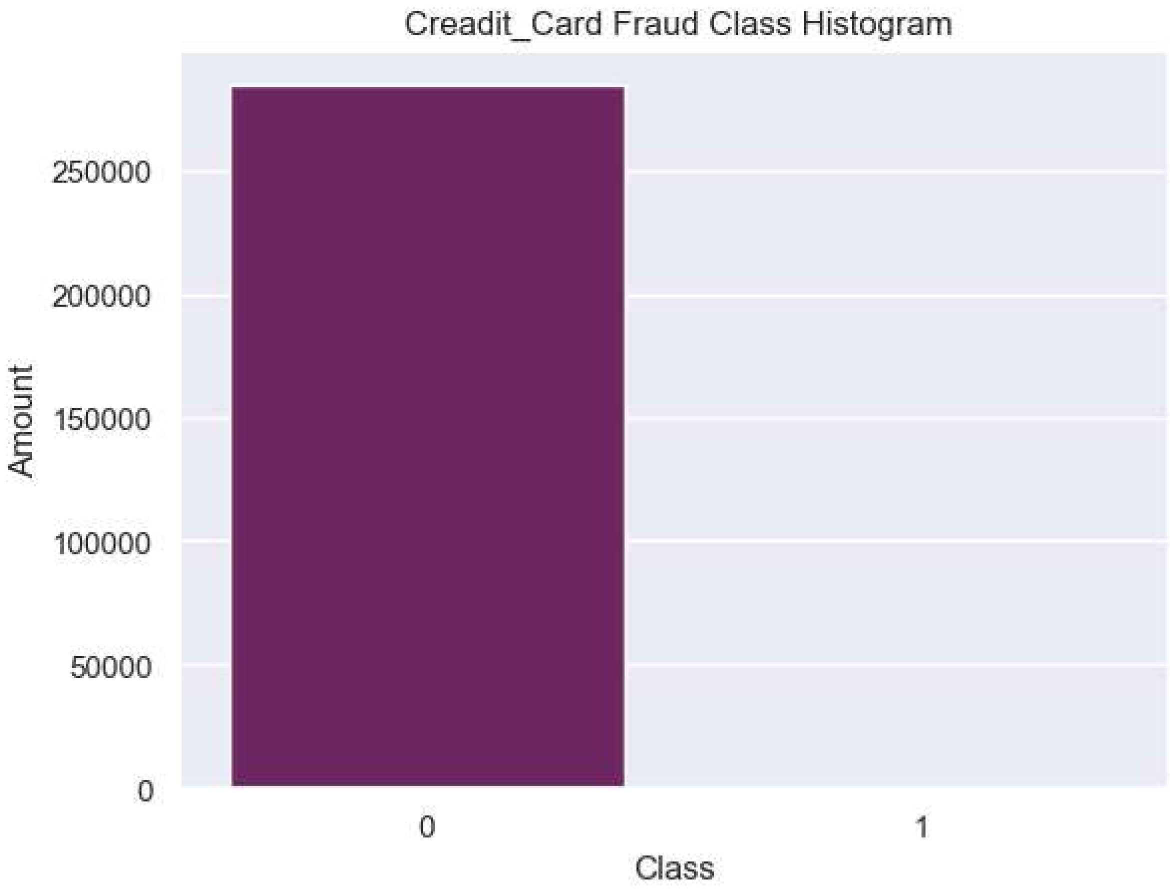
```
num_col = df.select_dtypes(exclude='object').columns.tolist()
plt.figure(figsize = (20,20))
df.boxplot(column = num_col)
plt.title('Boxplots for all Columns')
plt.xticks(rotation = 45)
plt.xlabel('Amount')
plt.ylabel('Class')
plt.grid(True)
plt.show()
```



```
In [19]: df.hist(bins=60,figsize=(25,30),color='black')
         plt.show()
```

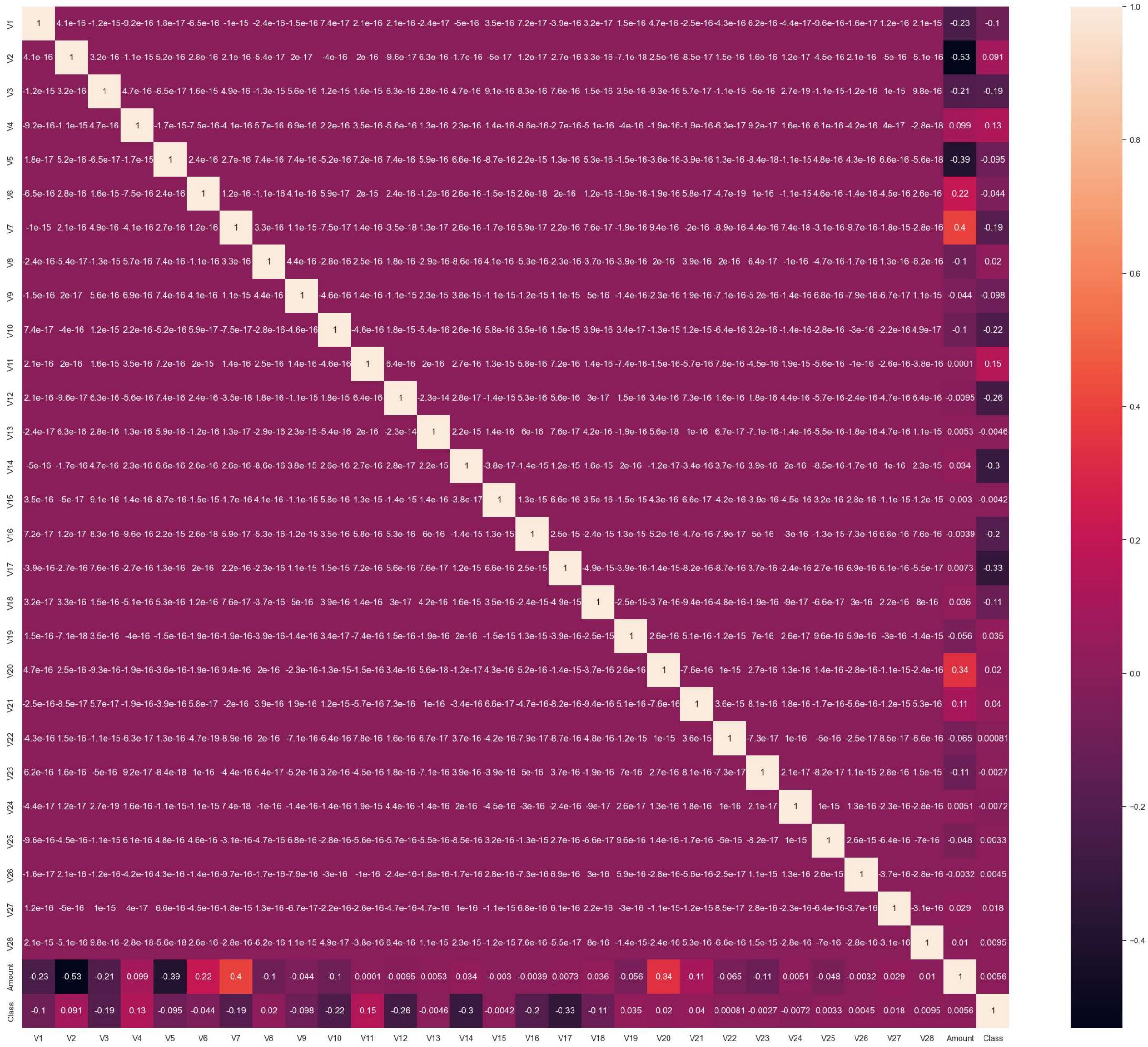


```
In [20]: sns.countplot(x="Class", data=df, palette="inferno")
         plt.title(" Creadit_Card Fraud Class Histogram")
         plt.xlabel("Class")
         plt.ylabel("Amount")
         plt.show()
```



Correlation

In [21]:
```
plt.figure(figsize= (30,25))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Feature Scaling

In [22]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_absolute_percentage_error, mean_squar
from sklearn import metrics
```

In [23]:
```
x= df.drop(['Class'],axis=1)
y= df[['Class']]
```

In [24]:
```
x.head()
```

Out[24]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

5 rows × 29 columns

In [25]:
```
y.head()
```

Out[25]:

	Class
0	0
1	0
2	0
3	0
4	0

In [26]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
pd.DataFrame(sc_x)
```

Out[26]:

	0	1	2	3	4	5	6	7	
0	-0.694242	-0.044075	1.672773	0.973366	-0.245117	0.347068	0.193679	0.082637	0.3
1	0.608496	0.161176	0.109797	0.316523	0.043483	-0.061820	-0.063700	0.071253	-0.2
2	-0.693500	-0.811578	1.169468	0.268231	-0.364572	1.351454	0.639776	0.207373	-1.3
3	-0.493325	-0.112169	1.182516	-0.609727	-0.007469	0.936150	0.192071	0.316018	-1.2
4	-0.591330	0.531541	1.021412	0.284655	-0.295015	0.071999	0.479302	-0.226510	0.7
...	...	...	...	...	...	...	...	...	
284802	-6.065842	6.099286	-6.486245	-1.459641	-3.886611	-1.956690	-3.975628	6.116573	1.7
284803	-0.374121	-0.033356	1.342145	-0.521651	0.629040	0.794446	0.019667	0.246886	0.5
284804	0.980024	-0.182434	-2.143205	-0.393984	1.905833	2.275262	-0.239939	0.593140	0.3
284805	-0.122755	0.321250	0.463320	0.487192	-0.273836	0.468155	-0.554672	0.568631	0.3
284806	-0.272331	-0.114899	0.463866	-0.357570	-0.009089	-0.487602	1.274769	-0.347176	0.4

284807 rows × 29 columns

Split the data into training and test for building the model and for prediction

In [27]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, randor
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(199364, 29) (85443, 29) (199364, 1) (85443, 1)
```

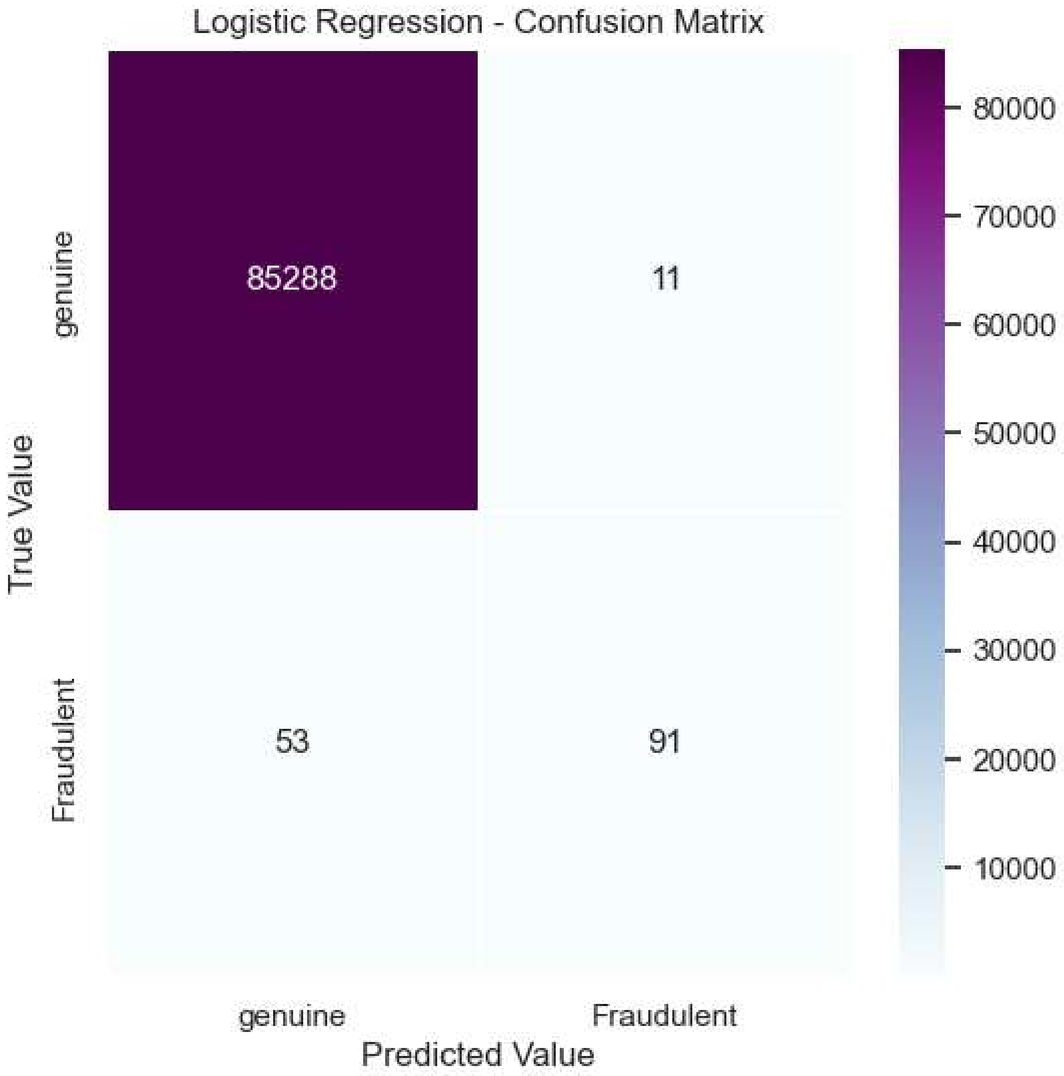
In [28]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
model = LogisticRegression()
model.fit(x_train,y_train)

y_pred = model.predict(x_test)
accuracy = accuracy_score(y_pred,y_test)
```

In [29]:

```
print('Accuracy score : ', accuracy)
print("Model Precision:", round(precision_score(y_test, y_pred),2))
print("Model Recall:", round(recall_score(y_test, y_pred),2))
print("Model F1-Score:", round(f1_score(y_test, y_pred),2))
print("Model ROC:", round(roc_auc_score(y_test, y_pred),2) , '\n')


conf_matrix=confusion_matrix(y_test, y_pred)
labels= ['genuine', 'Fraudulent']
plt.figure(figsize=(6, 6))
conf_matrix=confusion_matrix(y_test, y_pred)
labels= ['genuine', 'Fraudulent']
plt.figure(figsize=(6, 6))

sns.heatmap(pd.DataFrame(conf_matrix), xticklabels= labels, yticklabels= labels
            linewidths= 0.05 ,annot=True, fmt="d" , cmap='BuPu')

print(classification_report(y_test, y_pred, target_names=labels) , '\n')

plt.title("Logistic Regression - Confusion Matrix")
plt.ylabel('True Value')
plt.xlabel('Predicted Value')
plt.show()
```

```
Accuracy score :  0.9992509626300574
Model Precision: 0.89
Model Recall: 0.63
Model F1-Score: 0.74
Model ROC: 0.82

              precision    recall  f1-score   support

     genuine       1.00      1.00      1.00     85299
  Fraudulent       0.89      0.63      0.74       144

    accuracy                           1.00     85443
   macro avg       0.95      0.82      0.87     85443
weighted avg       1.00      1.00      1.00     85443



<Figure size 600x600 with 0 Axes>
```



In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
rfc=RandomForestClassifier()
model1=rfc.fit(x_train,y_train)

y_pred=model1.predict(x_test)
accuracy_score(y_test,y_pred)
```

In [ ]:

```
print('Accuracy score : ', accuracy)
print("Model Precision:", round(precision_score(y_test, y_pred),2))
print("Model Recall:", round(recall_score(y_test, y_pred),2))
print("Model F1-Score:", round(f1_score(y_test, y_pred),2))
print("Model ROC:", round(roc_auc_score(y_test, y_pred),2) , '\n')


conf_matrix=confusion_matrix(y_test, y_pred)
labels= ['genuine', 'Fraudulent']
plt.figure(figsize=(6, 6))
conf_matrix=confusion_matrix(y_test, y_pred)
labels= ['genuine', 'Fraudulent']
plt.figure(figsize=(6, 6))

sns.heatmap(pd.DataFrame(conf_matrix), xticklabels= labels, yticklabels= labels
            linewidths= 0.05 ,annot=True, fmt="d" , cmap='BuPu')

print(classification_report(y_test, y_pred, target_names=labels) , '\n')

plt.title("Random Forest Classifier - Confusion Matrix")
plt.ylabel('True Value')
plt.xlabel('Predicted Value')
plt.show()
```

Conclusion

```
dataset name creaditcard.csv in this dataset we import pandas,numpy,seaborn
and matplotlib librearies.
in these dataset there are 5 rows and 31 columns.all are float values.no
duplicate values.no null values.
with the help of heatmap we check correlation.in this correlation amount and
class is high corrlated.
Class is independent variable.we scale the data with the help of
standardscaler then split data for training and test for building the model
and for prediction.

Apply Logistic Regresion
Accuracy score :  0.9992509626300574
Model Precision: 0.89
Model Recall: 0.63
Model F1-Score: 0.74
Model ROC: 0.82

Random Forest Classifier
Accuracy score :  0.9992509626300574
Model Precision: 0.94
Model Recall: 0.8
Model F1-Score: 0.86
Model ROC: 0.9
```