In [25]:
```
# IMPORTING LIBRARIES :
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# READING DATASET :
data=pd.read_csv('creditcard.csv.zip')
data.head()
```

Out[4]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.0986 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.0851 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.2476 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.3774 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.2705 |

5 rows × 31 columns

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

Out[6]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | |
|---|---|---|---|---|---|---|---|---|
| **284802** | 172786.0 | -11.881118 | 10.071785 | -9.834783 | -2.066656 | -5.364473 | -2.606837 | -4.9182 |
| **284803** | 172787.0 | -0.732789 | -0.055080 | 2.035030 | -0.738589 | 0.868229 | 1.058415 | 0.0243 |
| **284804** | 172788.0 | 1.919565 | -0.301254 | -3.249640 | -0.557828 | 2.630515 | 3.031260 | -0.2968 |
| **284805** | 172788.0 | -0.240440 | 0.530483 | 0.702510 | 0.689799 | -0.377961 | 0.623708 | -0.6861 |
| **284806** | 172792.0 | -0.533413 | -0.189733 | 0.703337 | -0.506271 | -0.012546 | -0.649617 | 1.5770 |

5 rows × 31 columns

In [7]:
```
# NULL VALUES :
data.isnull().sum()
```

Out[7]:
```
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 [ ]:
```
# Thus there are no null values in the dataset
```

In [8]:
```
data.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 [9]:
```
# DESCRIPTIVE STATISTICS
data.describe().T.head()
```

Out[9]:

| | count | mean | std | min | 25% | 50% | |
|---|---|---|---|---|---|---|---|
| **Time** | 284807.0 | 9.481386e+04 | 47488.145955 | 0.000000 | 54201.500000 | 84692.000000 | 13932 |
| **V1** | 284807.0 | 1.168375e-15 | 1.958696 | -56.407510 | -0.920373 | 0.018109 | |
| **V2** | 284807.0 | 3.416908e-16 | 1.651309 | -72.715728 | -0.598550 | 0.065486 | |
| **V3** | 284807.0 | -1.379537e-15 | 1.516255 | -48.325589 | -0.890365 | 0.179846 | |
| **V4** | 284807.0 | 2.074095e-15 | 1.415869 | -5.683171 | -0.848640 | -0.019847 | |

In [10]:
```
data.shape
```

Out[10]:
```
(284807, 31)
```

In [ ]:
```
# Thus there are 284807 rows and 31 columns
```

In [11]:
```
data.columns
```

Out[11]:
```
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 [13]:
```
#FRAUD CASES AND GENUINE CASES
fraud_cases=len(data[data['Class']==1])
```

In [14]:
```
print('Number of fraud Cases :',fraud_cases)
```

```
Number of fraud Cases : 492
```

In [15]:
```
non_fraud_cases=len(data[data['Class']==0])
```

In [16]:
```
print('Number of  Non fraud Cases :',non_fraud_cases)
```

```
Number of  Non fraud Cases : 284315
```

In [17]:
```
fraud=data[data['Class']==1]
```

In [18]:
```
genuine=data[data['Class']==0]
```

In [19]:
```
#Statistical Measure of the data
fraud.Amount.describe()
```

Out[19]:
```
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
```

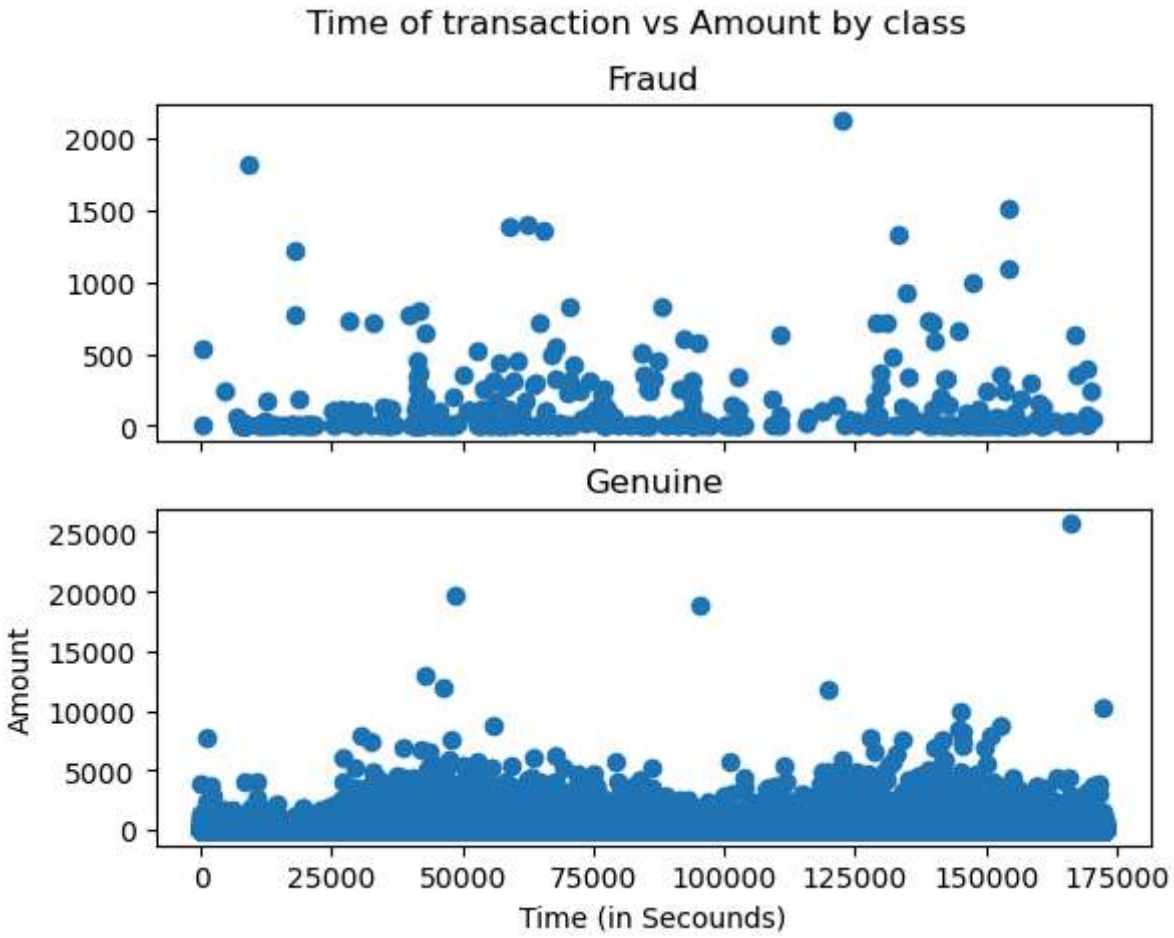
In [20]:
```
genuine.Amount.describe()
```

Out[20]:
```
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 [22]:
```
# EDA
data.hist(figsize=(20,20),color='lime')
plt.show()
```

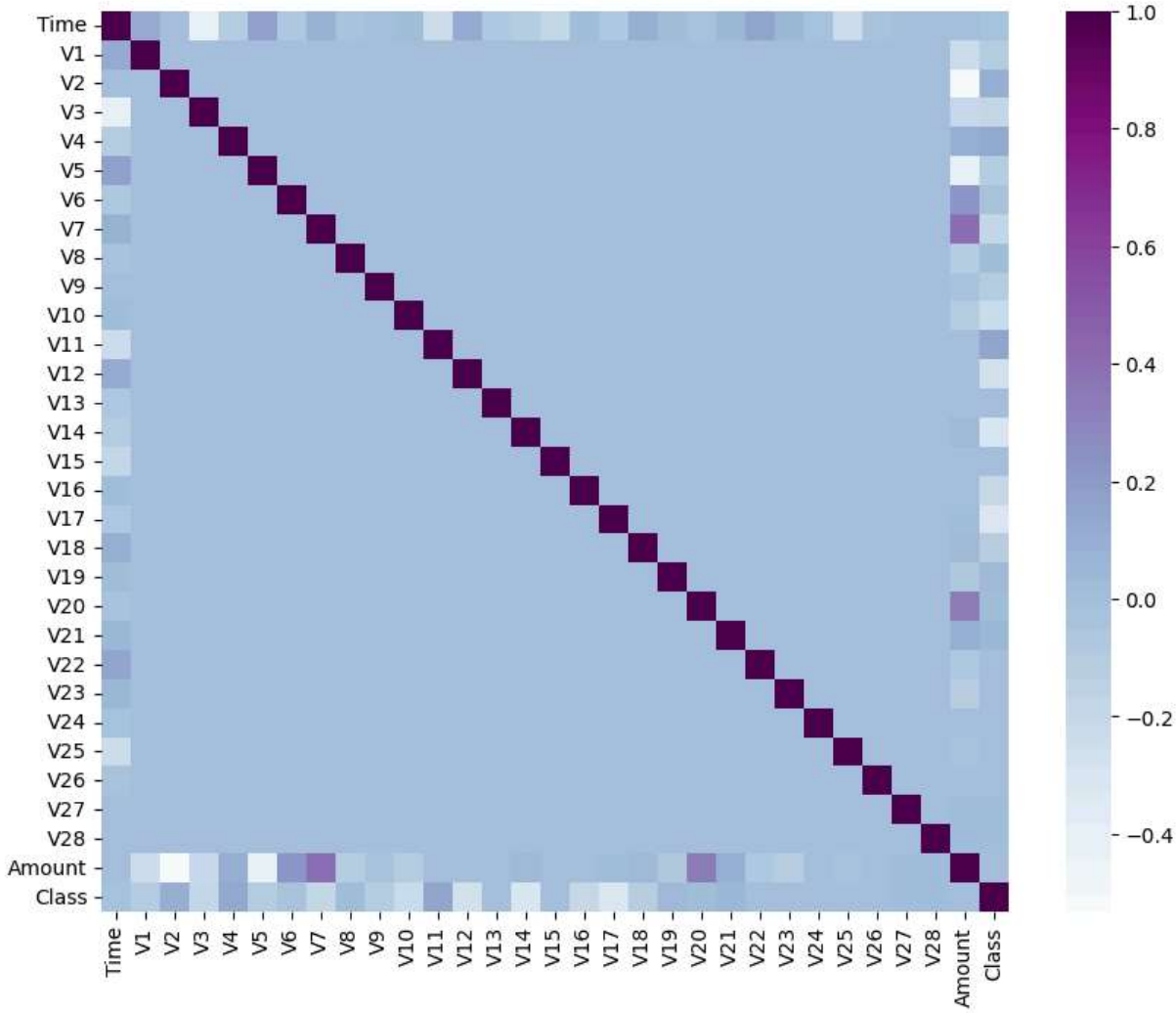

In [28]:
```
f,(ax1,ax2)=plt.subplots(2,1,sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(fraud.Time,fraud.Amount)
ax1.set_title('Fraud')
ax2.scatter(genuine.Time,genuine.Amount)
ax2.set_title('Genuine')
```

```
plt.xlabel('Time (in Secounds)')
plt.ylabel('Amount')
plt.show()
```




In [29]:
```
# Correlation
plt.figure(figsize=(10,8))
corr=data.corr()
sns.heatmap(corr,cmap='BuPu')
```

Out[29]: <Axes: >




In [30]:
```
# LET US BUILD OUR MODELS
from sklearn.model_selection import train_test_split
```

In [31]:
```
X=data.drop(['Class'],axis=1)
```

In [55]:
```
y=data['Class']
```

In [34]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=123)
```

In [35]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [36]:
```
rfc=RandomForestClassifier()
```

In [38]:
```
model=rfc.fit(X_train,y_train)
```

In [39]:
```
prediction=model.predict(X_test)
```

In [42]:
```
from sklearn.metrics import accuracy_score
```

In [43]:
```
accuracy_score(y_test,prediction)
```

Out[43]:
```
0.9995552590615966
```

In [ ]:
```
# MODEL 2 :
```

In [49]:
```
from sklearn .linear_model import LogisticRegression
```

In [44]:
```
X1=data.drop(['Class'],axis=1)
```

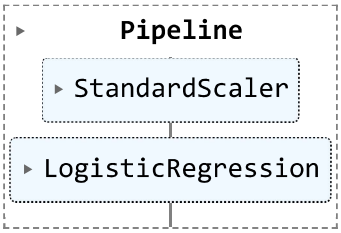
In [45]:
```
y1=data['Class']
```

In [56]:
```
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.30,random_stat
```

In [72]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

In [76]:
```
model2 = make_pipeline(StandardScaler(), LogisticRegression())
model2.fit(X_train, y_train)
```

Out[76]:




In [77]:
```
prediction2=model2.predict(X1_test)
```

In [78]:
```
accuracy_score(y1_test,prediction2)
```

Out[78]:
```
0.9990285921608558
```

In [ ]:
```
# MODEL 3 :
```

In [59]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [60]:
```
X2=data.drop(['Class'],axis=1)
```

In [61]:
```
y2=data['Class']
```

In [62]:
```
dt=DecisionTreeRegressor()
```

In [63]:
```
X2_train,X2_test,y2_train,y2_test=train_test_split(X2,y2,test_size=0.3,random_state
```

In [64]:
```
model3=dt.fit(X2_train,y2_train)
```

In [65]:
```
prediction3=model3.predict(X2_test)
```

In [66]:
```
accuracy_score(y2_test,prediction3)
```

Out[66]: 0.9991456292499094

In [ ]:
```
# Overall Models Performed with a very high accuracy .
```