In [ ]:
```
7321 - NANDHA COLLEGE OF TECHNOLOGY

ADS_Phase1-CreditCard_Fraud_Detection

*******************************************
PROJECT BY                                *
                                          *
DHARUN KUMAR T  #Teamlead                 *
BALAMURUGAN                               *
KAVIYARASU                                *
ARUN KUMAR                                *
*******************************************
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sys
import scipy
```

In [3]:
```
data = pd.read_csv(r'C:\Users\dharu\Downloads\Project-CreditCardFraudDetection\Dataset\creditcard.csv')
```

In [4]:
```
print(data.columns)
```

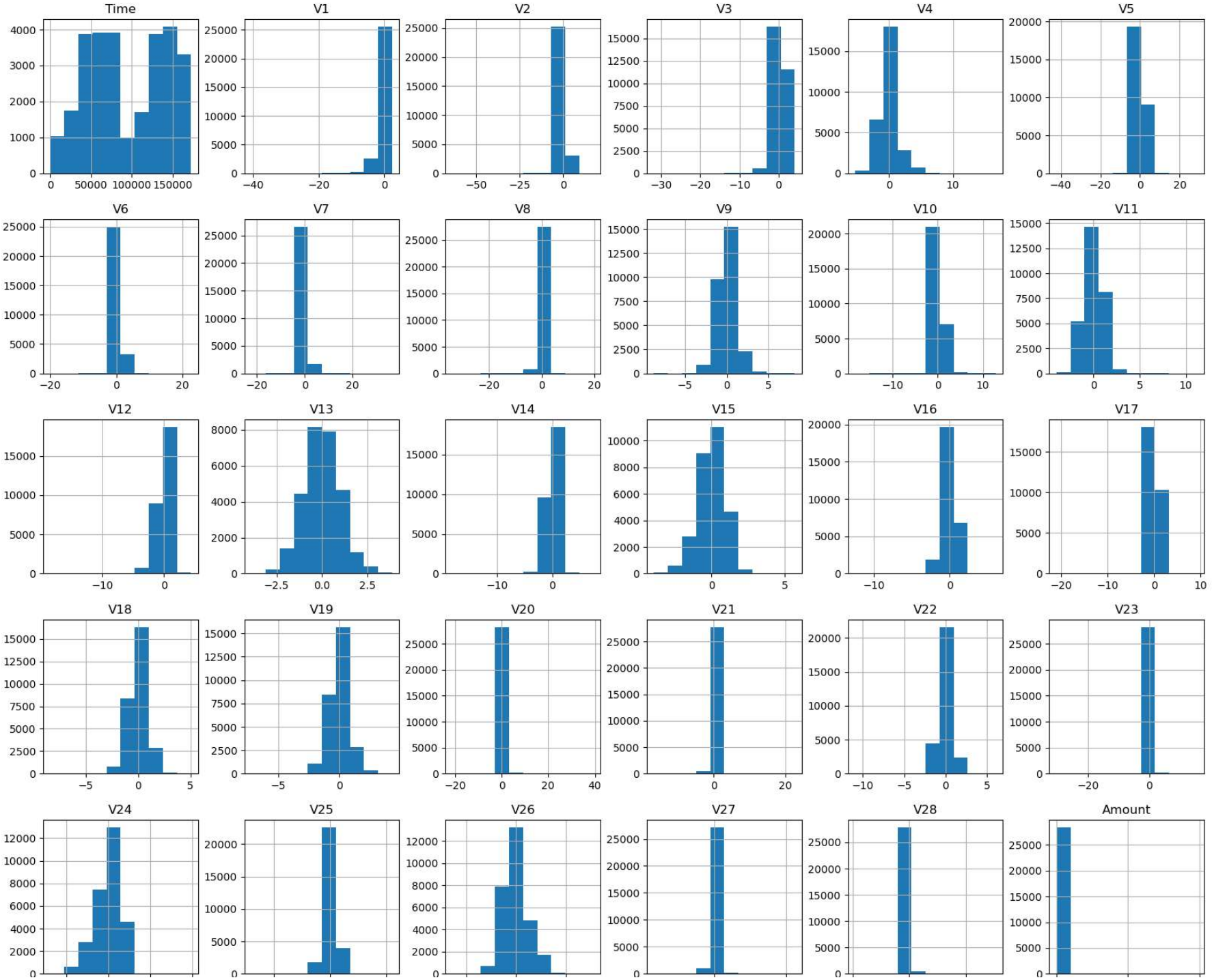
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

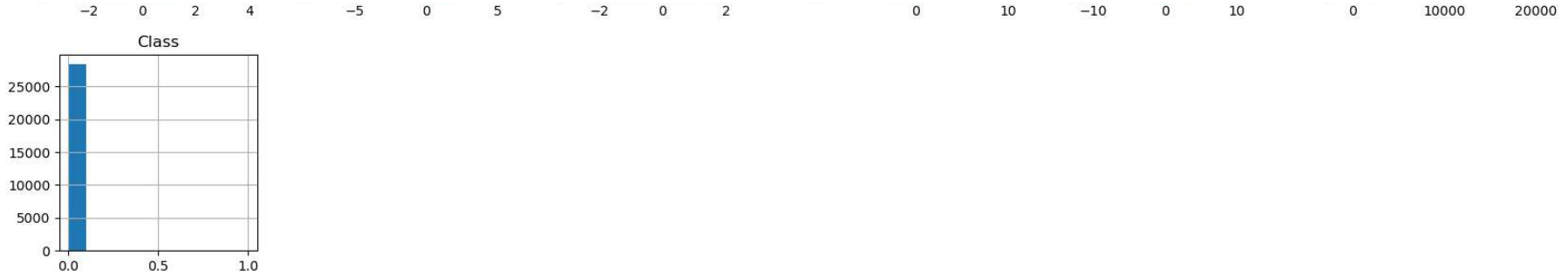
In [5]:
```
data = data.sample(frac=0.1, random_state = 1)
print(data.shape)
print(data.describe())
```

```
(28481, 31)
                Time            V1            V2            V3            V4  \
count   28481.000000  28481.000000  28481.000000  28481.000000  28481.000000
mean    94705.035216     -0.001143     -0.018290      0.000795      0.000350
std     47584.727034      1.994661      1.709050      1.522313      1.420003
min         0.000000    -40.470142    -63.344698    -31.813586     -5.266509
25%     53924.000000     -0.908809     -0.610322     -0.892884     -0.847370
50%     84551.000000      0.031139      0.051775      0.178943     -0.017692
75%    139392.000000      1.320048      0.792685      1.035197      0.737312
max    172784.000000      2.411499     17.418649      4.069865     16.715537

                 V5            V6            V7            V8            V9  \
count  28481.000000  28481.000000  28481.000000  28481.000000  28481.000000
mean      -0.015666      0.003634     -0.008523     -0.003040      0.014536
std        1.395552      1.334985      1.237249      1.204102      1.098006
min      -42.147898    -19.996349    -22.291962    -33.785407     -8.739670
25%       -0.703986     -0.765807     -0.562033     -0.208445     -0.632488
50%       -0.068037     -0.269071      0.028378      0.024696     -0.037100
75%        0.603574      0.398839      0.559428      0.326057      0.621093
max       28.762671     22.529298     36.677268     19.587773      8.141560

       ...           V21           V22           V23           V24  \
count  ...  28481.000000  28481.000000  28481.000000  28481.000000
mean   ...      0.004740      0.006719     -0.000494     -0.002626
std    ...      0.744743      0.728209      0.645945      0.603968
min    ...    -16.640785    -10.933144    -30.269720     -2.752263
25%    ...     -0.224842     -0.535877     -0.163047     -0.360582
50%    ...     -0.029075      0.014337     -0.012678      0.038383
75%    ...      0.189068      0.533936      0.148065      0.434851
max    ...     22.588989      6.090514     15.626067      3.944520

                V25           V26           V27           V28        Amount  \
count  28481.000000  28481.000000  28481.000000  28481.000000  28481.000000
mean      -0.000917      0.004762     -0.001689     -0.004154     89.957884
std        0.520679      0.488171      0.418304      0.321646    270.894630
min       -7.025783     -2.534330     -8.260909     -9.617915      0.000000
25%       -0.319611     -0.328476     -0.071712     -0.053379      5.980000
50%        0.015231     -0.049750      0.000914      0.010753     22.350000
75%        0.351466      0.253580      0.090329      0.076267     78.930000
max        5.541598      3.118588     11.135740     15.373170  19656.530000
```

```
              Class
count  28481.000000
mean       0.001720
std        0.041443
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000

[8 rows x 31 columns]
```

In [6]:
```
data.hist(figsize = (20, 20))
plt.show()
```


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



In [7]:
```
Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]
```

In [8]:
```
outlier_fraction = len(Fraud)/float(len(Valid))
print(outlier_fraction)
```

```
0.0017234102419808666
```

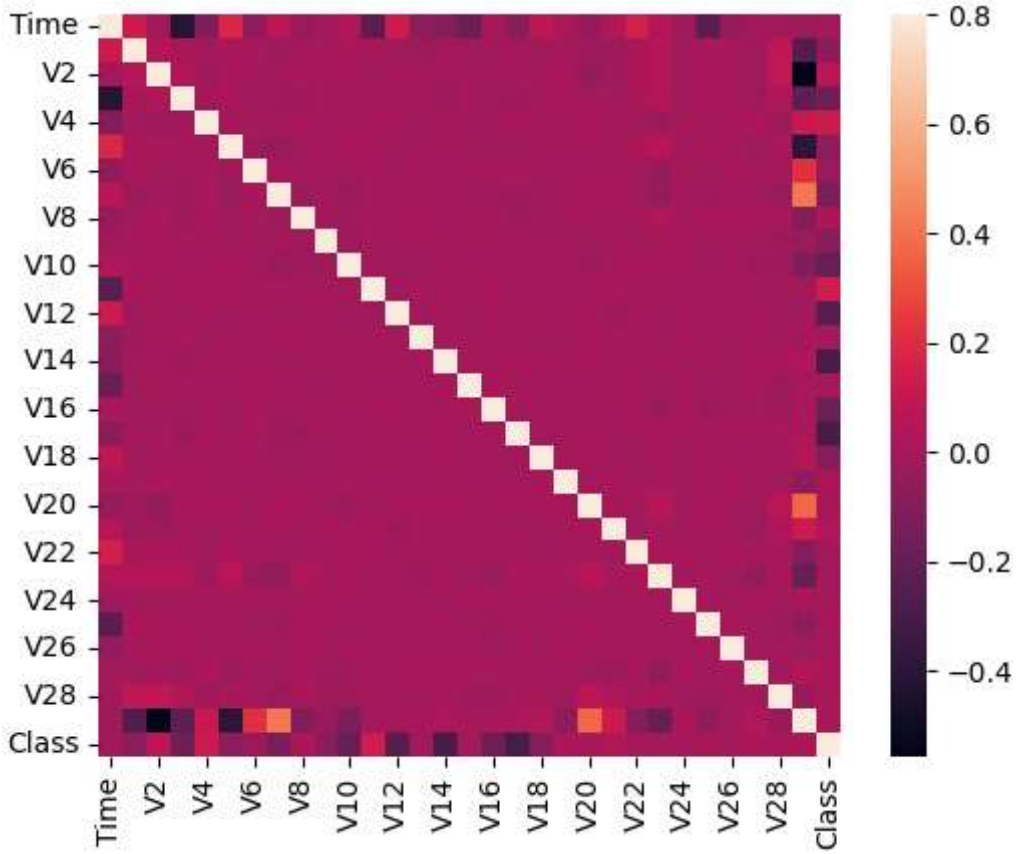
In [9]:
```
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```

```
Fraud Cases: 49
Valid Transactions: 28432
```

In [10]:
```
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
```

```
<Figure size 1200x900 with 0 Axes>
```

In [11]:
```
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



In [12]:
```
columns = data.columns.tolist()
```

In [13]:
```
columns = [c for c in columns if c not in ["Class"]]
```

In [14]:
```
target = "Class"
```

In [16]:
```
X = data[columns]
Y = data[target]
```

In [17]:
```
print(X.shape)
print(Y.shape)
```

```
(28481, 30)
(28481,)
```

In [18]:
```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
```

In [19]:
```
state = 1
```

In [20]:
```
classifiers = {
    "Isolation Forest": IsolationForest(max_samples=len(X),
                                        contamination=outlier_fraction,
                                        random_state=state),
    "Local Outlier Factor": LocalOutlierFactor(
        n_neighbors=20,
        contamination=outlier_fraction)}

plt.figure(figsize=(9, 7))
n_outliers = len(Fraud)
```

```
<Figure size 900x700 with 0 Axes>
```