In [6]:

```
import sys
import numpy
import pandas
import matplotlib
import scipy
import seaborn

print(sys.version)
print(numpy.__version__)
print(pandas.__version__)
print(matplotlib.__version__)
print(scipy.__version__)
print(seaborn.__version__)
```

```
3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
1.18.1
1.0.1
3.1.3
1.4.1
0.10.0
```

In [7]:

```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [43]:

```
data = pd.read_csv('creditcard.csv')
print(data.columns)
print(data.shape)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V1
0',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V2
0',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
(284807, 31)
```

In [44]:

```
data=data.sample(frac=1.0,random_state=1)
print(data.shape)
print(data.describe())
```

```
(284807, 31)
                Time            V1            V2            V3            
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+
05   
mean    94813.859575  1.170456e-15  2.933501e-16 -1.444466e-15  2.082684e-
15   
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+
00   
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+
00   
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-
01   
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-
02   
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-
01   
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+
01   

                 V5            V6            V7            V8            V
9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+0
5   
mean   1.004808e-15  1.499796e-15 -5.850625e-16  1.314693e-16 -2.359953e-1
5   
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+0
0   
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+0
1   
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-0
1   
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-0
2   
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-0
1   
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+0
1   

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05   
mean   ...  1.284953e-16 -3.518580e-16  2.673736e-16  4.474749e-15   
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01   
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00   
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01   
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02   
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01   
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00   

                V25           V26           V27           V28         Amou
nt  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.0000
00   
mean   5.197903e-16  1.690075e-15 -3.726377e-16 -1.325532e-16      88.3496
19   
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.1201
09   
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.0000
00   
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.6000
00   
```

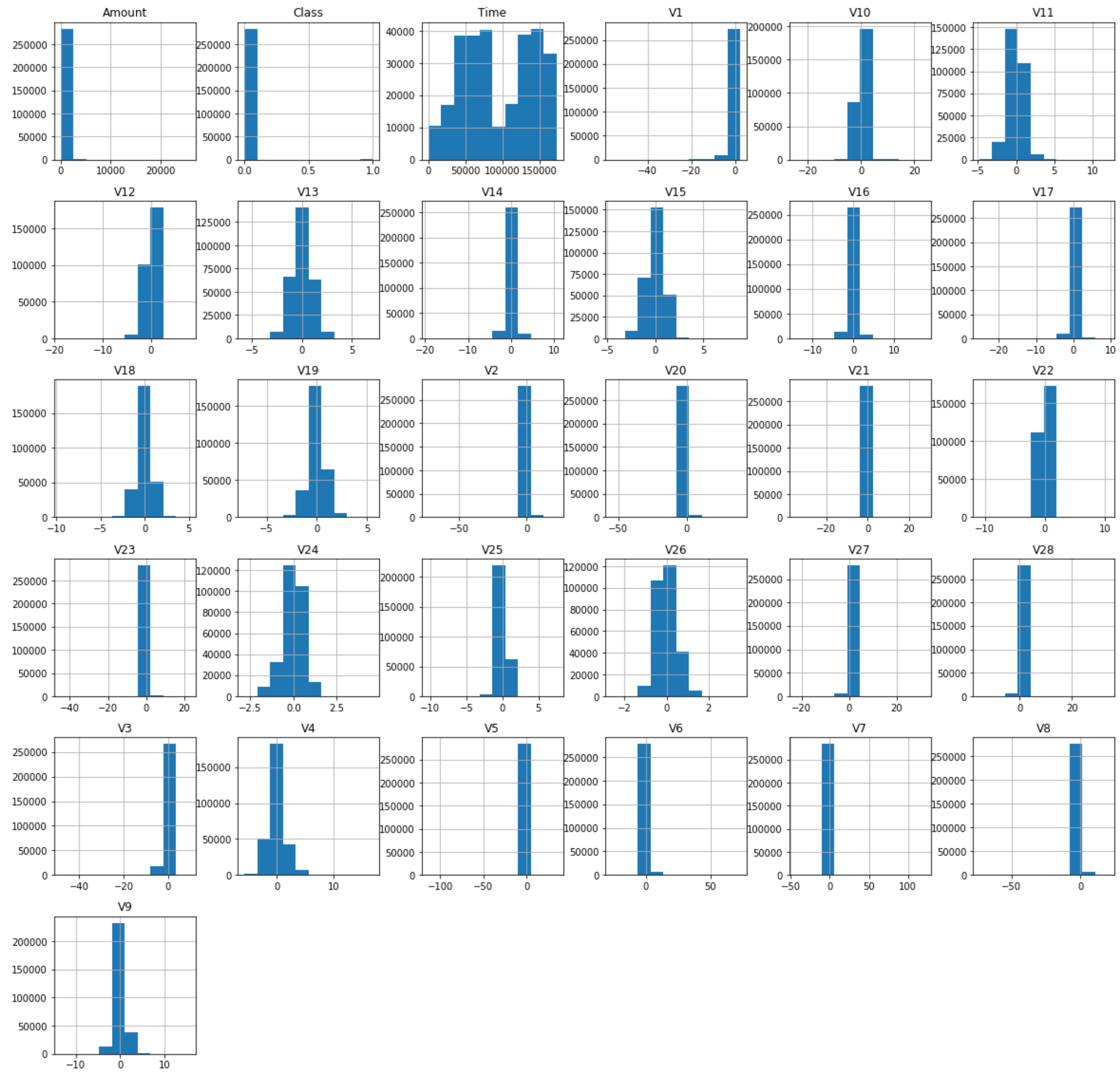
```
50%     1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.0000
00
75%     3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.1650
00
max     7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.1600
00

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

In [45]:

```
data.hist(figsize=(20,20))
plt.show()
```



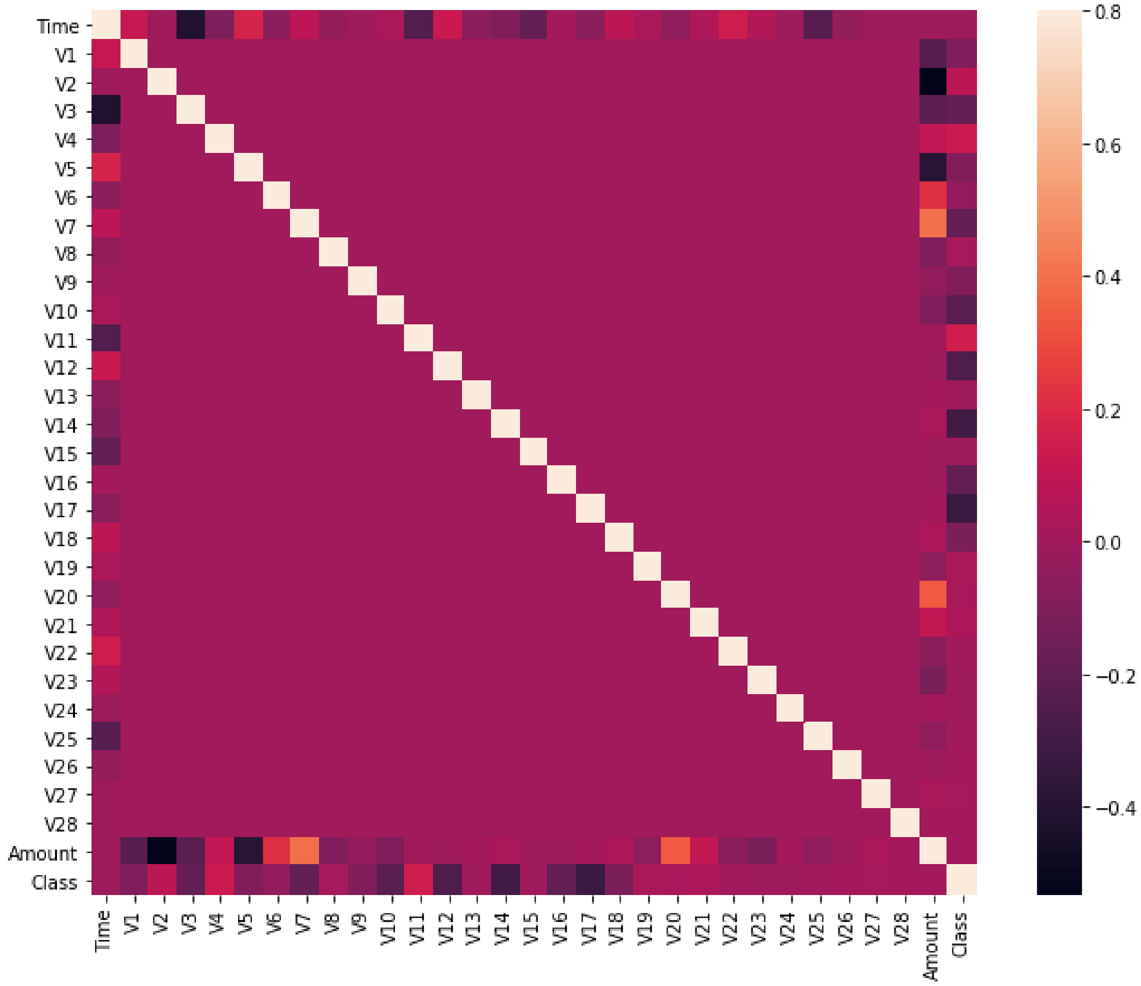
In [47]:

```
# Determine number of fraud cases in dataset
Fraud=data[data['Class']==1]
Valid=data[data['Class']==0]
Valid=Valid.sample(frac=0.1,random_state=1)
outlier_fraction = (len(Fraud))/(len(Valid))
print("Fraud Cases : ",len(Fraud))
print("Valid Cases : ",len(Valid))
print("Outlier fraction : ",outlier_fraction)
```

```
Fraud Cases :  492
Valid Cases :  28432
Outlier fraction :  0.017304445694991556
```

In [48]:

```
# Correlation matrix
corr=data.corr()
fig=plt.figure(figsize=(12,9))
sns.heatmap(corr,vmax=0.8,square=True)
plt.show()
```



In [49]:

```
# Get all the columns from the dataFrame
columns=data.columns.tolist()
# Filter the columns to remove data we do not want
columns=[c for c in columns if c not in ['Class']]
# Store the variable we'll be predicting on
target = "Class"
X = data[columns]
Y = data[target]
# Print shapes
print(X.shape)
print(Y.shape)
```

```
(284807, 30)
(284807,)
```

In [50]:

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import IsolationForest

models=[]
models.append(('LocalOutlierFactor',LocalOutlierFactor(n_neighbors=20,contamination=out
lier_fraction)))
models.append(('IsolationForest',IsolationForest(max_samples=len(X),contamination=outli
er_fraction,random_state=1)))

results = []
names = []
```

In [51]:

```
for name,model in models:
    if (name=='LocalOutlierFactor'):
        y_pred=model.fit_predict(X)
        scores_pred=model.negative_outlier_factor_
    else:
        model.fit(X)
        scores_pred = model.decision_function(X)
        y_pred = model.predict(X)
    # Reshape the prediction values to 0 for valid, 1 for fraud.
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1

    n_errors = (y_pred != Y).sum()

    # Run classification metrics
    print('{}: {}'.format(name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
```

```
LocalOutlierFactor: 5197
0.9817525552391619
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    284315
           1       0.02      0.23      0.04       492

    accuracy                           0.98    284807
   macro avg       0.51      0.61      0.52    284807
weighted avg       1.00      0.98      0.99    284807

IsolationForest: 4617
0.9837890220394864
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    284315
           1       0.08      0.82      0.15       492

    accuracy                           0.98    284807
   macro avg       0.54      0.90      0.57    284807
weighted avg       1.00      0.98      0.99    284807
```