In [131]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [132]:

```
#importing stock data
df=pd.read_csv("C:\\Users\\techane\\OneDrive - Nokia\MTNN Documents\\training\\Data Science
```

In [137]:

```
# identifying numerical variables
numeric_col=[i for i in df.columns if df[i].dtype !='object']
```

In [138]:

```
#removing not useful variables
numeric_col.remove('SP500')
```

In [144]:

```
#data frame after removing categorical variables
df_sub=df.loc[:,numeric_col]
numeric_col
```

Out[144]:

```
['NASDAQ.AAL',
 'NASDAQ.AAPL',
 'NASDAQ.ADBE',
 'NASDAQ.ADI',
 'NASDAQ.ADP',
 'NASDAQ.ADSK',
 'NASDAQ.AKAM',
 'NASDAQ.ALXN',
 'NASDAQ.AMAT',
 'NASDAQ.AMD',
 'NASDAQ.AMGN',
 'NASDAQ.AMZN',
 'NASDAQ.ATVI',
 'NASDAQ.AVGO',
 'NASDAQ.BBBY',
 'NASDAQ.BIIB',
 'NASDAQ.CA',
 'NASDAQ.CBOE',
```

In [145]:

```
df_sub.head()
```

Out[145]:

	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP	NASDAQ.ADSK
0	42.3300	143.6800	129.6300	82.040	102.2300	85.2200
1	42.3600	143.7000	130.3200	82.080	102.1400	85.6500
2	42.3100	143.6901	130.2250	82.030	102.2125	85.5100
3	42.3700	143.6400	130.0729	82.000	102.1400	85.4872
4	42.5378	143.6600	129.8800	82.035	102.0600	85.7001

5 rows × 500 columns

In [147]:

```
# Eigen-decomposition: 5-step process
# 1. Normalize columns of $A$ so that each feature has zero mean
mu = np.mean(df_sub,axis=0)
print("what is mean across rows? ",mu)
#print(np.mean(A,axis=0))
```

```
what is mean across rows?  NASDAQ.AAL       47.708346
NASDAQ.AAPL     150.453566
NASDAQ.ADBE     141.317930
NASDAQ.ADI       79.446873
NASDAQ.ADP      103.480398
NASDAQ.ADSK     102.998608
NASDAQ.AKAM      50.894352
NASDAQ.ALXN     122.981163
NASDAQ.AMAT      43.291988
NASDAQ.AMD       12.624442
NASDAQ.AMGN     167.030297
NASDAQ.AMZN     968.747188
NASDAQ.ATVI      57.683091
NASDAQ.AVGO     238.598238
NASDAQ.BBBY      33.413552
NASDAQ.BIIB     272.957640
NASDAQ.CA        32.652696
NASDAQ.CBOE      89.325485
NASDAQ.CELG     126.928020
NASDAQ.CERN      64.227338
NASDAQ.CHRW      69.502897
NASDAQ.CHTR     348.719389
NASDAQ.CINF      73.025271
NASDAQ.CMCSA     39.607469
NASDAQ.CME      121.375413
NASDAQ.COST     165.457076
NASDAQ.CSCO      32.139336
NASDAQ.CSX       51.284218
NASDAQ.CTAS     127.416660
NASDAQ.CTSH      65.928564
                   ...
NYSE.USB         51.863284
NYSE.UTX        119.265065
NYSE.V           95.693963
NYSE.VAR         98.311314
NYSE.VFC         57.134291
NYSE.VLO         65.903890
NYSE.VMC        123.767195
NYSE.VNO         89.828376
NYSE.VTR         66.790525
NYSE.VZ          46.574448
NYSE.WAT        176.242257
NYSE.WEC         62.352736
NYSE.WFC         53.587272
NYSE.WHR        181.971118
NYSE.WM          73.777928
NYSE.WMB         30.102558
NYSE.WMT         77.066819
NYSE.WRK         55.411353
NYSE.WU          19.272765
NYSE.WY          33.248472
NYSE.WYN         97.942211
NYSE.XEC        104.740666
```

```
NYSE.XEL          46.664402
NYSE.XL           43.043984
NYSE.XOM          80.784595
NYSE.XRX          19.300718
NYSE.XYL          54.541988
NYSE.YUM          71.757891
NYSE.ZBH         121.423515
NYSE.ZTS          60.183874
Length: 500, dtype: float64
```

In [148]:

```
#what is the variance of each variable from mean
A = df_sub - mu
print("Does A have zero mean across rows?" ,A)
```

```
Does A have zero mean across rows?        NASDAQ.AAL  NASDAQ.AAPL  NASDAQ.
ADBE  NASDAQ.ADI  NASDAQ.ADP  \
0       -5.378346    -6.773566    -11.68793    2.593127   -1.250398
1       -5.348346    -6.753566    -10.99793    2.633127   -1.340398
2       -5.398346    -6.763466    -11.09293    2.583127   -1.267898
3       -5.338346    -6.813566    -11.24503    2.553127   -1.340398
4       -5.170546    -6.793566    -11.43793    2.588127   -1.420398
5       -5.168446    -6.673566    -11.24793    2.593127   -1.440398
6       -5.238346    -6.589566    -11.13793    2.673127   -1.150398
7       -5.238346    -6.643566    -11.17793    2.743127   -1.110398
8       -5.318346    -6.638566    -11.21793    2.783127   -1.100398
9       -5.378346    -6.653566    -11.10793    2.718127   -1.150398
10      -5.308346    -6.563566    -11.17793    2.763127   -1.280398
11      -5.418346    -6.483566    -10.94793    2.703127   -1.200398
12      -5.418346    -6.533666    -10.85803    2.698127   -1.210398
13      -5.318346    -6.429666    -10.68793    2.713127   -1.170398
14      -5.288646    -6.403566    -10.66793    2.763127   -1.160398
15      -5.278346    -6.389766    -10.62293    2.703127   -1.200398
16      -5.268346    -6.433566    -10.74793    2.693127   -1.180398
```

In [149]:

```
print("Does A have zero mean across rows?")
print(np.mean(A,axis=0))
```

```
Does A have zero mean across rows?
NASDAQ.AAL     -4.473945e-13
NASDAQ.AAPL     1.129299e-13
NASDAQ.ADBE     8.095180e-13
NASDAQ.ADI      4.433276e-13
NASDAQ.ADP      2.407746e-14
NASDAQ.ADSK    -1.321120e-12
NASDAQ.AKAM     2.074824e-13
NASDAQ.ALXN     4.551943e-14
NASDAQ.AMAT    -1.462889e-13
NASDAQ.AMD      6.057607e-14
NASDAQ.AMGN    -1.611048e-13
NASDAQ.AMZN     9.634470e-12
NASDAQ.ATVI    -5.124359e-14
NASDAQ.AVGO     3.746106e-13
NASDAQ.BBBY    -1.052425e-11
NASDAQ.BIIB    -5.414148e-13
NASDAQ.CA      -1.734550e-14
NASDAQ.CBOE    -6.510487e-13
NASDAQ.CELG     5.757507e-13
NASDAQ.CERN     5.875589e-13
NASDAQ.CHRW    -7.791368e-13
NASDAQ.CHTR     2.257744e-13
NASDAQ.CINF     1.700523e-12
NASDAQ.CMCSA    6.903798e-15
NASDAQ.CME      9.930521e-13
NASDAQ.COST    -2.126399e-12
NASDAQ.CSCO     5.938103e-13
NASDAQ.CSX      6.105294e-13
NASDAQ.CTAS     1.619803e-12
NASDAQ.CTSH    -9.862563e-13
                    ...
NYSE.USB       -7.592180e-14
NYSE.UTX        9.541460e-13
NYSE.V          4.532397e-13
NYSE.VAR       -1.312919e-12
NYSE.VFC       -4.255851e-13
NYSE.VLO       -1.886762e-12
NYSE.VMC        2.691302e-14
NYSE.VNO       -6.015421e-13
NYSE.VTR        1.137675e-12
NYSE.VZ        -4.349282e-13
NYSE.WAT       -1.161935e-12
NYSE.WEC       -2.993781e-14
NYSE.WFC       -4.658964e-13
NYSE.WHR        5.163397e-13
NYSE.WM         7.496717e-13
NYSE.WMB        6.750406e-14
NYSE.WMT        4.985222e-13
NYSE.WRK       -7.606542e-13
NYSE.WU        -7.891057e-13
NYSE.WY         6.761374e-13
NYSE.WYN        3.164906e-13
NYSE.XEC        3.970309e-13
NYSE.XEL        1.221453e-12
NYSE.XL         1.766633e-12
```

```
NYSE.XOM       -1.960210e-13
NYSE.XRX       -4.592529e-13
NYSE.XYL        1.842671e-12
NYSE.YUM        2.481407e-14
NYSE.ZBH        1.733799e-13
NYSE.ZTS        2.737155e-14
Length: 500, dtype: float64
```

In [66]:

```
A.head()
```

Out[66]:

	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP	NASDAQ.ADSK
0	-5.378346	-6.773566	-11.68793	2.593127	-1.250398	-17.778608
1	-5.348346	-6.753566	-10.99793	2.633127	-1.340398	-17.348608
2	-5.398346	-6.763466	-11.09293	2.583127	-1.267898	-17.488608
3	-5.338346	-6.813566	-11.24503	2.553127	-1.340398	-17.511408
4	-5.170546	-6.793566	-11.43793	2.588127	-1.420398	-17.298508

5 rows × 500 columns

In [151]:

```
A.plot()
```

Problem 1:

There are various stocks for which we have collected a data set, which all stocks are apparently similar in performance

Answer: NASDAQ.AAL and NASDAQ.ADI , also NASDAQ.AAPL and NASDAQ.ADSK

Problem 2:

How many Unique patterns that exist in the historical stock data set, based on fluctuations in price. Answer:

Problem 3:

Identify which all stocks are moving together and which all stocks are different from each other. Answer:NASDAQ.AAL and NASDAQ.ADI , also NASDAQ.AAPL and NASDAQ.ADSK are moving together

In [75]:

```
# 2. Compute sample covariance matrix $\Sigma = {A^TA}/{(m-1)}$
m,n = A.shape
Sigma = (A.T @ A)/(m-1)
print("---")
print("Sigma:")
print(Sigma)
```

NASDAQ.COST	-13.364094	-15.980539	-45.873875	0.754849	-21.598853
NASDAQ.CSCO	-1.875065	-1.648113	-4.333775	-0.572958	-1.524510
NASDAQ.CSX	5.130944	-0.975188	3.744495	1.306309	-2.952576
NASDAQ.CTAS	5.646981	15.725261	27.496417	0.965458	15.781951
NASDAQ.CTSH	8.622379	17.876235	26.974796	2.260635	7.924173
...	...	...	...	...	...
NYSE.USB	1.298026	1.410011	2.709587	-0.090982	1.731402
NYSE.UTX	7.379969	1.390036	7.991675	1.576815	-2.480911
NYSE.V	5.509695	21.364342	28.859423	1.229431	11.279139
NYSE.VAR	12.475726	10.249454	27.880888	3.925223	1.398270
NYSE.VFC	2.023505	12.429875	17.736781	-0.973378	11.925478
NYSE.VLO	2.120756	1.245862	5.591540	-0.719074	3.942144
NYSE.VMC	6.434789	-11.582195	-6.393777	2.085524	-9.713816
NYSE.VNO	-8.032382	-45.626170	-56.177288	-1.041357	-23.667231
NYSE.VTR	3.235000	-0.772463	6.381245	1.435827	0.214267
NYSE.VZ	-3.378310	2.328594	-1.546604	-0.497096	3.183918
NYSE.WAT	20.173908	21.770504	44.687434	4.739180	2.474672
NYSE.WEC	1.063051	6.399792	8.348193	0.721558	3.007157
NYSE.WFC	0.861587	-4.646576	-2.832439	-0.636691	-0.597032

In [77]:

```
# 3. Perform eigen-decomposition of $\Sigma$ using `np.linalg.eig(Sigma)`
l,X = np.linalg.eig(Sigma)
print("---")
print("Evalues:")
print(l)
print("---")
print("Evectors:")
print(X)
# 4. Compress by ordering $k$ evectors according to largest evalues and compute $A
print("---")
print("Compressed - 500D to 2D:")
Acomp = A @ X[:,:2] # first 2 evectors
print(Acomp.values[:5,:]) # first 5 observations
```

```
---
Evectors:
[[-0.01343332 -0.01712446  0.00801129 ... -0.01299917  0.03335562
   0.0095582 ]
 [-0.02041695  0.01486247 -0.02532257 ...  0.02341258 -0.01032956
  -0.00012904]
 [-0.03841415  0.01133053 -0.01399184 ...  0.03457151  0.00726189
  -0.00532613]
 ...
 [-0.02241145 -0.00144584 -0.01539753 ...  0.00173837  0.0864945
  -0.01899818]
 [-0.00365631 -0.03904339  0.01912915 ...  0.01743407  0.0017988
  -0.00567462]
 [-0.01648086 -0.01612222 -0.01613713 ...  0.02983854 -0.01092281
   0.00252217]]
---
Compressed - 500D to 2D:
[[306.7014591   83.2240155 ]
 [303.07353165  79.13615476]
 [301.07978706  78.29811525]
```

In [67]:

```
#Normalizationm
#from sklearn.preprocessing iport SandardScaler
#norm_df=StandardScaler().fit_transform(df_sub)
from sklearn.preprocessing import StandardScaler
norm_df = StandardScaler().fit_transform(df_sub)
```

In [36]:

```
## Perform PCA and get top 12 PC's
from sklearn.decomposition import PCA
pca=PCA(n_components=12)
pca_df=pca.fit_transform(norm_df)
pca.explained_variance_ratio_.cumsum()
```

Out[36]:

```
array([0.47112747, 0.64316169, 0.76314865, 0.8116034 , 0.84962898,
       0.8776535 , 0.89762461, 0.91066073, 0.9211418 , 0.92864373,
       0.93546419, 0.94123476])
```

In [50]:

```
##Unsuperivsed ML Exmaple
from sklearn.cluster import *
##fit kmeans object to data
kmeans=KMeans(n_clusters=3,n_init=100).fit(pca_df)
#Print location of clusters learned by kmeans object
centroids=kmeans.cluster_centers_
centroids
##save new clusters for chart
y_km=kmeans.fit_predict(pca_df)
y_km=pd.Series(y_km)
pca_df1=pd.DataFrame(pca_df)
```

In [51]:

```
y_km.value_counts()
```

Out[51]:

```
1    16901
0    14199
2    10166
dtype: int64
```

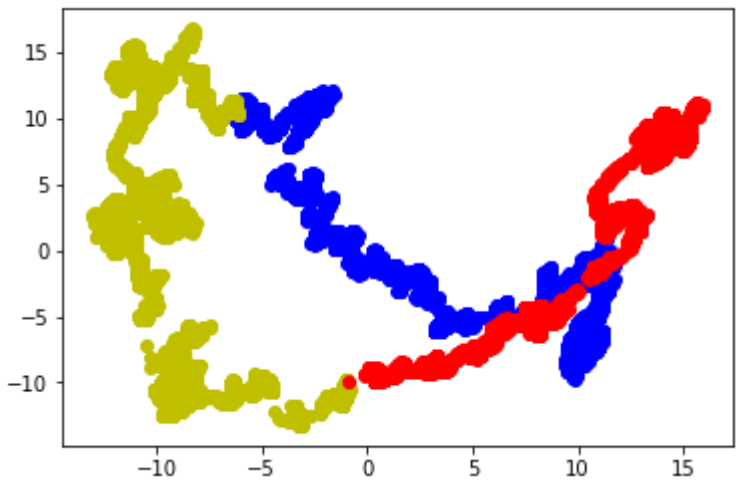
In [52]:

```
#plot the cluster
from scipy.cluster.vq import vq
centroids=kmeans.cluster_centers_
idx,_=vq(pca_df,centroids)

plt.plot(pca_df[idx==0,1],pca_df[idx==0,2],'ob',
        pca_df[idx==1,1],pca_df[idx==1,2],'oy',
        pca_df[idx==2,1],pca_df[idx==2,2],'or',
        pca_df[idx==3,1],pca_df[idx==3,2],'og',)
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x27094953fd0>,
 <matplotlib.lines.Line2D at 0x2709495d128>,
 <matplotlib.lines.Line2D at 0x2709495d4e0>,
 <matplotlib.lines.Line2D at 0x2709495d828>]
```

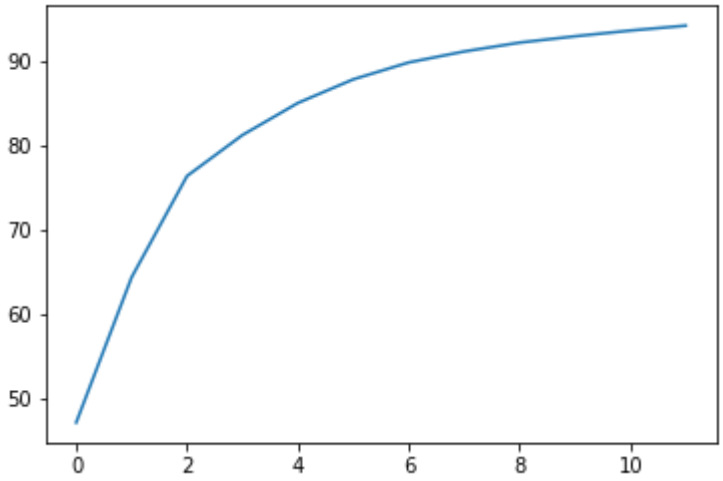


In [53]:

```
##The amount of variance that each PC explains Cumulative variance explains
var=pca.explained_variance_ratio_
var1=np.cumsum(np.round(var,decimals=4)*100)
plt.plot(var1)
```

Out[53]:

```
[<matplotlib.lines.Line2D at 0x270949ae7f0>]
```



In [ ]: