In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

In [2]:
```
master_stocks=pd.read_csv(r"D:\PG-DAI\MachineLearning\Assessment\2 Stocks\Stocks.csv",index_col=0, parse_dates=True)
```

In [3]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalized = normalizer.fit_transform(master_stocks)
normalized = pd.DataFrame(normalized)
```

In [4]:
```
dataset = normalized
```

In [ ]:
```

```

In [5]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [6]:
```
pca=PCA()
```

In [7]:
```
pca.fit(dataset)
pca.explained_variance_ratio_
```

Out[7]:
```
array([8.04404689e-02, 5.19094992e-02, 4.36165478e-02, 3.45087572e-02,
       3.27412627e-02, 3.14581729e-02, 3.03444349e-02, 2.58773055e-02,
       2.50243147e-02, 2.44208855e-02, 2.38435957e-02, 2.21524499e-02,
       2.18475042e-02, 2.12363637e-02, 2.01636425e-02, 1.99710713e-02,
       1.94534505e-02, 1.84696307e-02, 1.80950586e-02, 1.75619213e-02,
```

```
       1.69351391e-02, 1.62717571e-02, 1.61155228e-02, 1.57215124e-02,
       1.53369877e-02, 1.51309538e-02, 1.49817207e-02, 1.41739316e-02,
       1.37499692e-02, 1.34650706e-02, 1.30974534e-02, 1.29956985e-02,
       1.21881813e-02, 1.20508896e-02, 1.19313317e-02, 1.16792927e-02,
       1.13531186e-02, 1.11888215e-02, 1.07344111e-02, 1.04579645e-02,
       1.03078845e-02, 9.96366611e-03, 9.70631542e-03, 9.52800441e-03,
       9.14422560e-03, 8.98942165e-03, 8.55160526e-03, 8.38159223e-03,
       8.04082170e-03, 7.88500173e-03, 7.81652350e-03, 7.63905575e-03,
       7.16085098e-03, 6.92838481e-03, 6.62759143e-03, 6.06418144e-03,
       5.59293281e-03, 5.27082236e-03, 3.70505274e-03, 1.58640853e-32])
```

In [8]:

```
pca.explained_variance_ratio_.cumsum()
```

Out[8]:

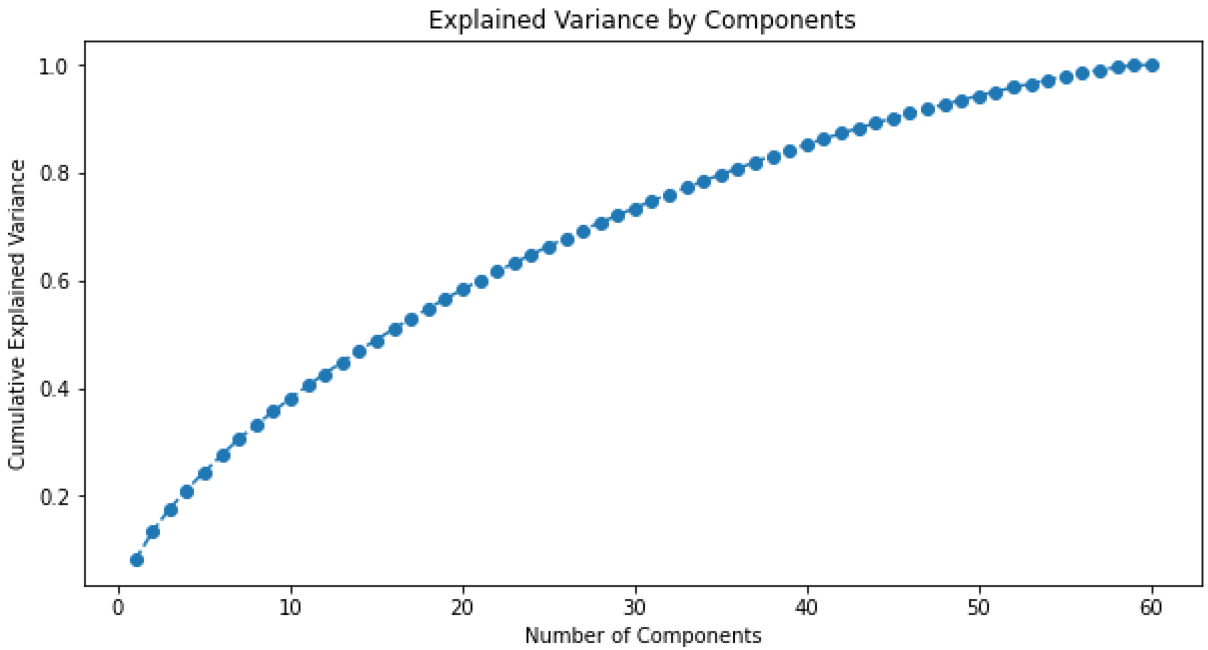
```
array([0.08044047, 0.13234997, 0.17596652, 0.21047527, 0.24321654,
       0.27467471, 0.30501914, 0.33089645, 0.35592076, 0.38034165,
       0.40418525, 0.42633769, 0.4481852 , 0.46942156, 0.48958521,
       0.50955628, 0.52900973, 0.54747936, 0.56557442, 0.58313634,
       0.60007148, 0.61634323, 0.63245876, 0.64818027, 0.66351726,
       0.67864821, 0.69362993, 0.70780386, 0.72155383, 0.7350189 ,
       0.74811636, 0.76111205, 0.77330024, 0.78535113, 0.79728246,
       0.80896175, 0.82031487, 0.83150369, 0.8422381 , 0.85269607,
       0.86300395, 0.87296762, 0.88267393, 0.89220194, 0.90134616,
       0.91033558, 0.91888719, 0.92726878, 0.9353096 , 0.9431946 ,
       0.95101113, 0.95865018, 0.96581103, 0.97273942, 0.97936701,
       0.98543119, 0.99102412, 0.99629495, 1.        , 1.        ])
```

In [9]:

```
# Plot the cumulative variance explained by total number of components.

# On this graph we choose the subset of components we want to keep.
# Generally, we want to keep around 80 % - 90% of the explained variance.
plt.figure(figsize=(10,5))

plt.plot (range (1,61), pca.explained_variance_ratio_.cumsum (), marker = 'o', linestyle = '--')

plt.title('Explained Variance by Components')

plt.xlabel('Number of Components')

plt.ylabel('Cumulative Explained Variance')
```

Out[9]:

```
Text(0, 0.5, 'Cumulative Explained Variance')
```



In [ ]:

In [10]:

```
pca=PCA(n_components= 2)
pca.fit(dataset)
pca.explained_variance_ratio_
```

Out[10]:

```
array([0.08044047, 0.05190949])
```

In [11]:

```
df= pca.transform(dataset)
print(df)
df1=np.transpose(df)
PCA1=df1[0]
PCA2=df1[1]
```

```
[[-0.11824823  0.35296398]
 [-0.15926247  0.10761514]
 [-0.03864502  0.31577576]
 [-0.06890844  0.00815713]
```

```
[-0.00444909  0.09115912]
[-0.27560099 -0.11617978]
[ 0.15529006 -0.24118928]
[-0.14047121 -0.11239899]
[-0.23158323 -0.08188801]
[ 0.53217835  0.08635414]
[-0.12271266 -0.25029236]
[-0.17209558  0.1728141 ]
[-0.04493132 -0.29422712]
[-0.14968265 -0.10769057]
[-0.12245937  0.2063754 ]
[-0.21691251  0.04979786]
[-0.09310356 -0.13321219]
[-0.05757547  0.28627918]
[-0.24495527 -0.09369863]
[ 0.10244377 -0.20305513]
[ 0.13174751  0.13051999]
[-0.16707865 -0.09697708]
[-0.19321261  0.14271219]
[-0.02041894  0.09578808]
[-0.12453192  0.19216232]
[ 0.37624537 -0.06208548]
[-0.239637   -0.18078083]
[ 0.56199062  0.05582227]
[ 0.40018357 -0.0160266 ]
[ 0.09934897  0.15531213]
[ 0.03526919  0.17170239]
[ 0.20450812  0.11343026]
[-0.01463738 -0.06069971]
[-0.11085189  0.22296014]
[-0.11096881 -0.07306946]
[-0.24405328  0.08940732]
[ 0.01824889  0.14806255]
[ 0.1125811  -0.2589201 ]
[ 0.49764734  0.03335719]
[ 0.2688715  -0.1466831 ]
[ 0.49744765 -0.03368875]
[ 0.39385375 -0.00298386]
[-0.05967575 -0.32728195]
[-0.08255778 -0.09535435]
[-0.21642115 -0.14802573]
[-0.19778501 -0.06806181]
[ 0.07075827 -0.25225739]
[-0.11725829  0.1952064 ]
```

```
 [-0.10046737 -0.0504449 ]
 [-0.08686366 -0.31506294]
 [-0.11306335  0.23403964]
 [-0.20275424  0.2149784 ]
 [ 0.08762533 -0.25321625]
 [-0.141865   -0.0318676 ]
 [ 0.22018436  0.15308596]
 [-0.2008282  -0.12099524]
 [ 0.41715123  0.07438395]
 [ 0.01779595 -0.27485557]
 [-0.16860599  0.07565833]
 [-0.02623754  0.32728944]]
```

In [12]:

```
from sklearn.cluster import KMeans
```

In [13]:

```
sse = []
kmeans = range(1,10)
for k in kmeans:
    km = KMeans(n_clusters=k)
    km.fit(df)
    sse.append(km.inertia_)

print(sse)
```

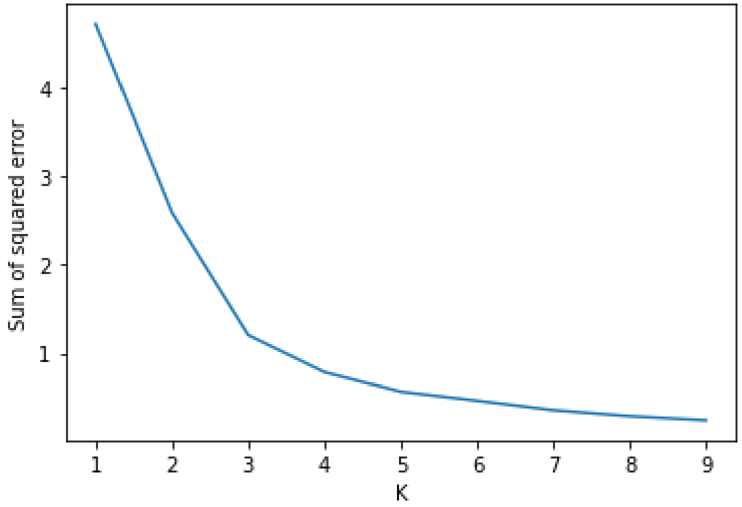
```
[4.720262296516009, 2.586956572598725, 1.2048595969390796, 0.7881450092230488, 0.5667681929767235, 0.46558764941028347, 0.35915571
78886689, 0.29236410558105763, 0.2456107907650285]
```

In [14]:

```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(kmeans,sse)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x2892886de50>]
```



In [15]:
```
km = KMeans(n_clusters=10)
y_predicted = km.fit_predict(df)
y_predicted
```

Out[15]:
```
array([7, 3, 7, 8, 6, 0, 2, 8, 0, 9, 5, 3, 5, 8, 7, 3, 8, 7, 0, 2, 1, 0,
       3, 6, 7, 4, 0, 9, 4, 1, 6, 1, 8, 7, 8, 3, 6, 2, 9, 4, 9, 4, 5, 8,
       0, 0, 2, 7, 8, 5, 7, 3, 2, 8, 1, 0, 9, 2, 3, 7])
```

In [16]:
```
km.cluster_centers_
```

Out[16]:
```
array([[-0.22173619, -0.11332589],
       [ 0.16394724,  0.13808709],
       [ 0.09108241, -0.24724895],
       [-0.19384238,  0.12185476],
       [ 0.35978855, -0.05694476],
       [-0.07854585, -0.29671609],
       [ 0.00716251,  0.12667804],
       [-0.09209679,  0.25922803],
       [-0.1002958 , -0.0729534 ],
       [ 0.50128304,  0.04324576]])
```

In [17]:
```
df_plot= pd.DataFrame()
df_plot['pca1']=np.transpose(PCA1)
df_plot['pca2']=np.transpose(PCA2)
```

```
df_plot['cluster']=y_predicted
df_plot
```

Out[17]:

	pca1	pca2	cluster
0	-0.118248	0.352964	7
1	-0.159262	0.107615	3
2	-0.038645	0.315776	7
3	-0.068908	0.008157	8
4	-0.004449	0.091159	6
5	-0.275601	-0.116180	0
6	0.155290	-0.241189	2
7	-0.140471	-0.112399	8
8	-0.231583	-0.081888	0
9	0.532178	0.086354	9
10	-0.122713	-0.250292	5
11	-0.172096	0.172814	3
12	-0.044931	-0.294227	5
13	-0.149683	-0.107691	8
14	-0.122459	0.206375	7
15	-0.216913	0.049798	3
16	-0.093104	-0.133212	8
17	-0.057575	0.286279	7
18	-0.244955	-0.093699	0
19	0.102444	-0.203055	2
20	0.131748	0.130520	1
21	-0.167079	-0.096977	0

	pca1	pca2	cluster
22	-0.193213	0.142712	3
23	-0.020419	0.095788	6
24	-0.124532	0.192162	7
25	0.376245	-0.062085	4
26	-0.239637	-0.180781	0
27	0.561991	0.055822	9
28	0.400184	-0.016027	4
29	0.099349	0.155312	1
30	0.035269	0.171702	6
31	0.204508	0.113430	1
32	-0.014637	-0.060700	8
33	-0.110852	0.222960	7
34	-0.110969	-0.073069	8
35	-0.244053	0.089407	3
36	0.018249	0.148063	6
37	0.112581	-0.258920	2
38	0.497647	0.033357	9
39	0.268872	-0.146683	4
40	0.497448	-0.033689	9
41	0.393854	-0.002984	4
42	-0.059676	-0.327282	5
43	-0.082558	-0.095354	8
44	-0.216421	-0.148026	0
45	-0.197785	-0.068062	0

	pca1	pca2	cluster
46	0.070758	-0.252257	2
47	-0.117258	0.195206	7
48	-0.100467	-0.050445	8
49	-0.086864	-0.315063	5
50	-0.113063	0.234040	7
51	-0.202754	0.214978	3
52	0.087625	-0.253216	2
53	-0.141865	-0.031868	8
54	0.220184	0.153086	1
55	-0.200828	-0.120995	0
56	0.417151	0.074384	9
57	0.017796	-0.274856	2
58	-0.168606	0.075658	3
59	-0.026238	0.327289	7

In [18]:

```
df_plot1 = df_plot[df_plot.cluster==0]
df_plot2 = df_plot[df_plot.cluster==1]
df_plot3 = df_plot[df_plot.cluster==2]
df_plot4 = df_plot[df_plot.cluster==3]
df_plot5 = df_plot[df_plot.cluster==4]
df_plot6 = df_plot[df_plot.cluster==5]
df_plot7 = df_plot[df_plot.cluster==6]
df_plot8 = df_plot[df_plot.cluster==7]
df_plot9 = df_plot[df_plot.cluster==8]
df_plot10 = df_plot[df_plot.cluster==9]
```

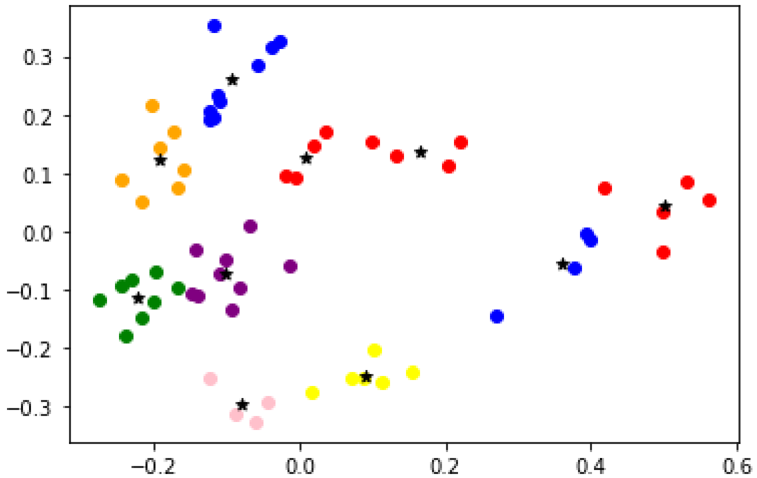
In [20]:

```
plt.scatter(df_plot1['pca1'],df_plot1['pca2'],color='green',label='cluster 1')
plt.scatter(df_plot2['pca1'],df_plot2['pca2'],color='red',label='cluster 2')
plt.scatter(df_plot3['pca1'],df_plot3['pca2'],color='yellow',label='cluster 3')
```

```python
plt.scatter(df_plot4['pca1'],df_plot4['pca2'],color='orange',label='cluster 4')
plt.scatter(df_plot5['pca1'],df_plot5['pca2'],color='blue',label='cluster 5')
plt.scatter(df_plot6['pca1'],df_plot6['pca2'],color='pink',label='cluster 6')
plt.scatter(df_plot7['pca1'],df_plot7['pca2'],color='red',label='cluster 7')
plt.scatter(df_plot8['pca1'],df_plot8['pca2'],color='blue',label='cluster 8')
plt.scatter(df_plot9['pca1'],df_plot9['pca2'],color='purple',label='cluster 9')
plt.scatter(df_plot10['pca1'],df_plot10['pca2'],color='red',label='cluster 10')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='black',marker='*',label='centroid')
# plt.legend()
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x289289ddd60>
```



In [22]:

```
!set PATH=/Library/TeX/texbin:$PATH
```

In [ ]: