0.) Import and Clean data

In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [21]:
```
df = pd.read_csv(r"C:\Users\antek\Downloads\Country-data.csv", sep = ",")
```

In [22]:
```
df.head()
```

Out[22]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

In [23]:
```
names = df['country'].copy
X=df.drop(['country'],axis = 1)
```

In [24]:
```
scaler = StandardScaler().fit(X)
X_scaled = scaler.transform(X)
```

1.) Fit a kmeans Model with any Number of Clusters

In [26]:
```
kmeans = KMeans(n_clusters = 5)
kmeans.fit(X_scaled)
```

```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the valu
e of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

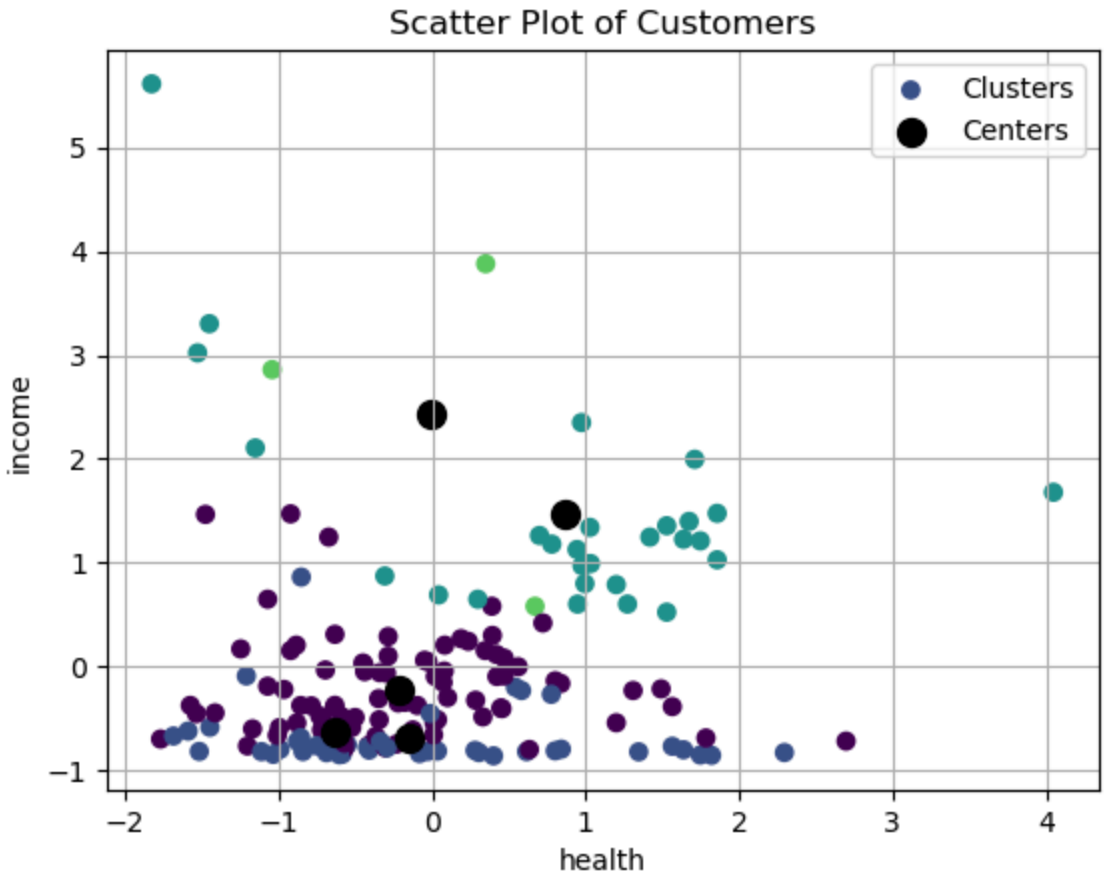
Out[26]:
```
KMeans
KMeans(n_clusters=5)
```

2.) Pick two features to visualize across

In [27]:
```
X.columns
```

Out[27]:
```
Index(['child_mort', 'exports', 'health', 'imports', 'income', 'inflation',
       'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

In [30]:
```
import matplotlib.pyplot as plt

x1_index = 2
x2_index = 4


scatter = plt.scatter(X_scaled[:, x1_index], X_scaled[:, x2_index], c=kmeans.labels_,


centers = plt.scatter(kmeans.cluster_centers_[:, x1_index], kmeans.cluster_centers_[:,

plt.xlabel(X.columns[x1_index])
plt.ylabel(X.columns[x2_index])
plt.title('Scatter Plot of Customers')

# Generate legend
plt.legend()

plt.grid()
plt.show()
```



3.) Check a range of k-clusters and visualize to find the elbow. Test 30 different random starting places for the centroid means

In [32]:
```
wCSSs = []
Ks = range(1,15)
for k in Ks:
    kmeans = KMeans(n_clusters = k,n_init = 30).fit(X_scaled)
    wCSSs.append(kmeans.inertia_)
```

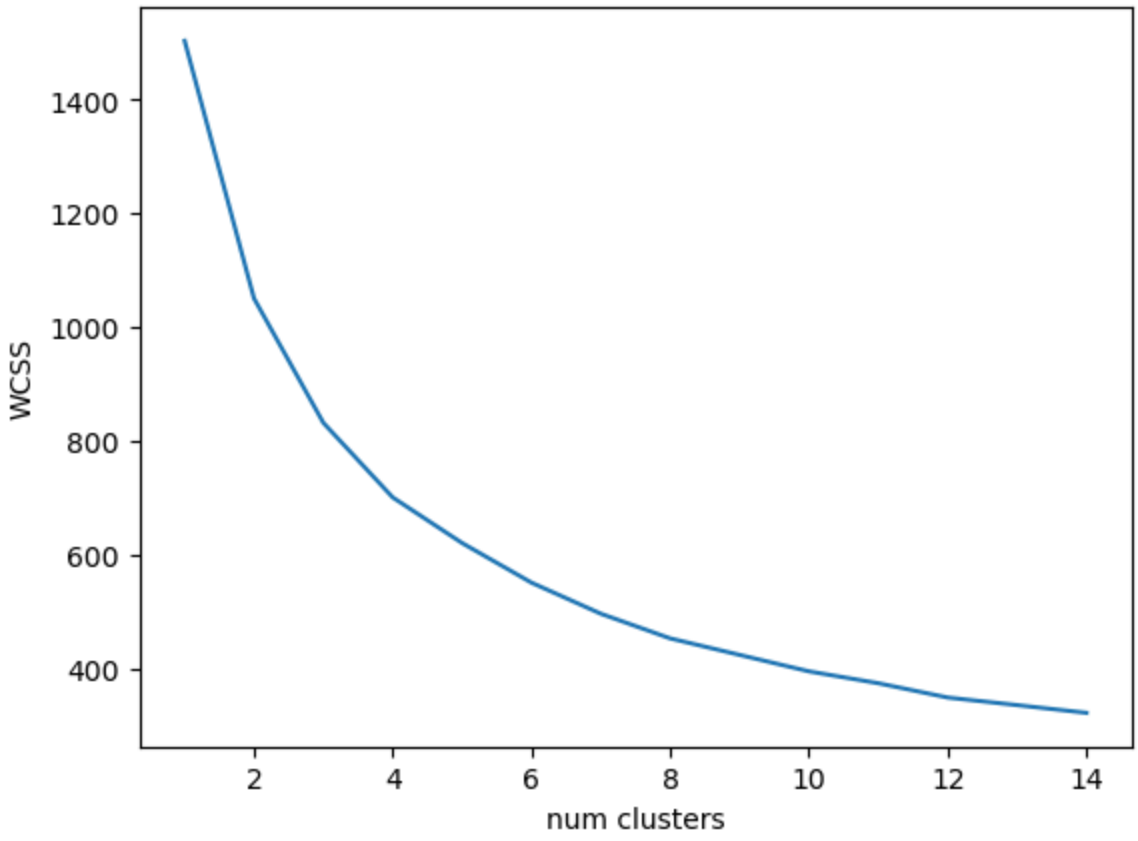
```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

4.) Use the above work and economic critical thinking to choose a number of clusters. Explain why you chose the number of clusters and fit a model accordingly.

In [36]:
```
plt.plot(Ks,wCSSs)
plt.xlabel('num clusters')
plt.ylabel('WCSS')
```

Out[36]:
```
Text(0, 0.5, 'WCSS')
```



In [ ]:
```
#divisions between countries will likely be very difficult to group by in clusters of
```

6.) Do the same for a silhoutte plot

In [37]:
```
from sklearn.metrics import silhouette_score
```

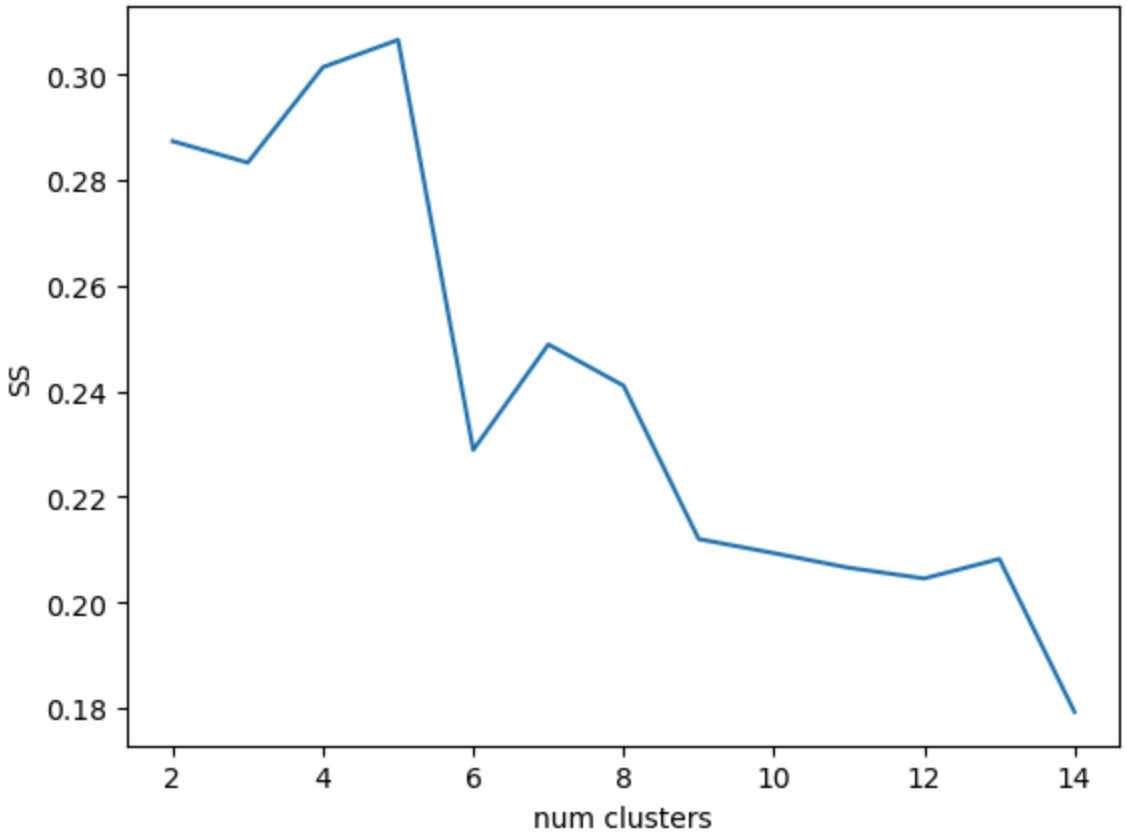
In [40]:
```
SSs = []
Ks = range(2,15)
for k in Ks:
    kmeans = KMeans(n_clusters = k,n_init = 30).fit(X_scaled)
    sil = silhouette_score(X_scaled,kmeans.labels_)
    SSs.append(sil)
```

```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

In [43]:
```
plt.plot(Ks,SSs)
plt.xlabel('num clusters')
plt.ylabel('SS')
```

Out[43]:
```
Text(0, 0.5, 'SS')
```



7.) Create a list of the countries that are in each cluster. Write interesting things you notice.

In [44]:
```
kmeans = KMeans(n_clusters = 2,n_init = 30).fit(X_scaled)
```

```
C:\Users\antek\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

In [46]:
```
preds = pd.DataFrame(kmeans.labels_)
```

In [47]:
```
output = pd.concat((preds,df),axis = 1)
```

In [48]:
```
output
```

Out[48]:

	0	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gd
0	0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	
1	1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4(
2	1	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4.
3	0	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3.
4	1	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12.
...	...	...	...	...	...	...	...	...	...	...	
162	0	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2(
163	1	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13.
164	1	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1.
165	0	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1.
166	0	Zambia	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1.

167 rows × 11 columns

In [50]:
```
print('Cluster 1:' )
list(output.loc[output[0] == 0,'country'] )
```

```
Cluster 1:
```

Out[50]:
```
['Afghanistan',
 'Angola',
 'Bangladesh',
 'Benin',
 'Bolivia',
 'Botswana',
 'Burkina Faso',
 'Burundi',
 'Cambodia',
 'Cameroon',
 'Central African Republic',
 'Chad',
 'Comoros',
 'Congo, Dem. Rep.',
 'Congo, Rep.',
 "Cote d'Ivoire",
 'Egypt',
 'Equatorial Guinea',
 'Eritrea',
 'Gabon',
 'Gambia',
 'Ghana',
 'Guatemala',
 'Guinea',
 'Guinea-Bissau',
 'Guyana',
 'Haiti',
 'India',
 'Indonesia',
 'Iraq',
 'Kenya',
 'Kiribati',
 'Kyrgyz Republic',
 'Lao',
 'Lesotho',
 'Liberia',
 'Madagascar',
 'Malawi',
 'Mali',
 'Mauritania',
 'Micronesia, Fed. Sts.',
 'Mongolia',
 'Mozambique',
 'Myanmar',
 'Namibia',
 'Nepal',
 'Niger',
 'Nigeria',
 'Pakistan',
 'Philippines',
 'Rwanda',
 'Samoa',
 'Senegal',
 'Sierra Leone',
 'Solomon Islands',
 'South Africa',
 'Sudan',
 'Tajikistan',
 'Tanzania',
 'Timor-Leste',
```

```
 'Togo',
 'Tonga',
 'Turkmenistan',
 'Uganda',
 'Uzbekistan',
 'Vanuatu',
 'Yemen',
 'Zambia']
```

In [51]:
```
print('Cluster 2:' )
list(output.loc[output[0] == 1,'country'] )
```

```
Cluster 2:
```

Out[51]:
```
['Albania',
 'Algeria',
 'Antigua and Barbuda',
 'Argentina',
 'Armenia',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas',
 'Bahrain',
 'Barbados',
 'Belarus',
 'Belgium',
 'Belize',
 'Bhutan',
 'Bosnia and Herzegovina',
 'Brazil',
 'Brunei',
 'Bulgaria',
 'Canada',
 'Cape Verde',
 'Chile',
 'China',
 'Colombia',
 'Costa Rica',
 'Croatia',
 'Cyprus',
 'Czech Republic',
 'Denmark',
 'Dominican Republic',
 'Ecuador',
 'El Salvador',
 'Estonia',
 'Fiji',
 'Finland',
 'France',
 'Georgia',
 'Germany',
 'Greece',
 'Grenada',
 'Hungary',
 'Iceland',
 'Iran',
 'Ireland',
 'Israel',
 'Italy',
 'Jamaica',
 'Japan',
 'Jordan',
 'Kazakhstan',
 'Kuwait',
 'Latvia',
 'Lebanon',
 'Libya',
 'Lithuania',
 'Luxembourg',
 'Macedonia, FYR',
 'Malaysia',
 'Maldives',
 'Malta',
```

```
 'Mauritius',
 'Moldova',
 'Montenegro',
 'Morocco',
 'Netherlands',
 'New Zealand',
 'Norway',
 'Oman',
 'Panama',
 'Paraguay',
 'Peru',
 'Poland',
 'Portugal',
 'Qatar',
 'Romania',
 'Russia',
 'Saudi Arabia',
 'Serbia',
 'Seychelles',
 'Singapore',
 'Slovak Republic',
 'Slovenia',
 'South Korea',
 'Spain',
 'Sri Lanka',
 'St. Vincent and the Grenadines',
 'Suriname',
 'Sweden',
 'Switzerland',
 'Thailand',
 'Tunisia',
 'Turkey',
 'Ukraine',
 'United Arab Emirates',
 'United Kingdom',
 'United States',
 'Uruguay',
 'Venezuela',
 'Vietnam']
```

In [3]:

In [ ]:
```
#Cluster 2 possess more developed economics however there are some suprising entries i
#Cluster 1 posses mongolia whiles paraguay is in cluster 2 despite having nearly a 2x
```

8.) Create a table of Descriptive Statistics. Rows being the Cluster number and columns being all the features. Values being the mean of the centroid. Use the nonscaled X values for interprotation

In [58]:
```
output.drop(columns = output[['country']],axis = 1).groupby(0).mean()
```

Out[58]:

0	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	76.280882	30.198515	6.090147	43.642146	4227.397059	11.098750	61.910294	4.413824	1981.23
1	12.161616	48.603030	7.314040	49.121212	26017.171717	5.503545	76.493939	1.941111	20507.97

In [59]:
```
output.drop(columns = output[['country']],axis = 1).groupby(0).std()
```

Out[59]:

0	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	38.076068	18.201742	2.645319	19.323451	4890.581414	13.682630	6.897418	1.285590	2528.50
1	8.523122	30.116032	2.716652	26.928785	20441.749847	6.957187	3.735757	0.486744	20578.72

In [3]:
```

```

9.) Write an observation about the descriptive statistics.

In [3]:
```
#it appears cluster 2 is higher in positive metrics and lower in all negative metrics
```