
(http://www.pieriandata.com)

CIA Country Analysis and Clustering

Source: All these data sets are made up of data from the US government.
https://www.cia.gov/library/publications/the-world-factbook/docs/faqs.html
(https://www.cia.gov/library/publications/the-world-factbook/docs/faqs.html)

Goal:

Gain insights into similarity between countries and regions of the world by experimenting with different cluster amounts. What do these clusters represent? *Note: There is no 100% right answer, make sure to watch the video for thoughts.*

Imports and Data

TASK: Run the following cells to import libraries and read in data.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\22 - K-Means Clustering\\32407456-CIA-Country-Facts.csv")
```

Exploratory Data Analysis

TASK: Explore the rows and columns of the data as well as the data types of the columns.

In [3]:

```
# CODE HERE
df.head()
```

Out[3]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	G d
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	4
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	6
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	8
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	19

In [704]:

```

```

Out[704]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	G d
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	4
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	6
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	8
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	19

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Country                             227 non-null    object 
 1   Region                              227 non-null    object 
 2   Population                          227 non-null    int64  
 3   Area (sq. mi.)                      227 non-null    int64  
 4   Pop. Density (per sq. mi.)          227 non-null    float64
 5   Coastline (coast/area ratio)        227 non-null    float64
 6   Net migration                       224 non-null    float64
 7   Infant mortality (per 1000 births)  224 non-null    float64
 8   GDP ($ per capita)                  226 non-null    float64
 9   Literacy (%)                        209 non-null    float64
 10  Phones (per 1000)                   223 non-null    float64
 11  Arable (%)                          225 non-null    float64
 12  Crops (%)                           225 non-null    float64
 13  Other (%)                           225 non-null    float64
 14  Climate                             205 non-null    float64
 15  Birthrate                           224 non-null    float64
 16  Deathrate                           223 non-null    float64
 17  Agriculture                         212 non-null    float64
 18  Industry                            211 non-null    float64
 19  Service                             212 non-null    float64
dtypes: float64(16), int64(2), object(2)
memory usage: 35.6+ KB
```

In [705]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Country                             227 non-null    object 
 1   Region                              227 non-null    object 
 2   Population                          227 non-null    int64  
 3   Area (sq. mi.)                      227 non-null    int64  
 4   Pop. Density (per sq. mi.)          227 non-null    float64
 5   Coastline (coast/area ratio)        227 non-null    float64
 6   Net migration                       224 non-null    float64
 7   Infant mortality (per 1000 births)  224 non-null    float64
 8   GDP ($ per capita)                  226 non-null    float64
 9   Literacy (%)                        209 non-null    float64
 10  Phones (per 1000)                   223 non-null    float64
 11  Arable (%)                          225 non-null    float64
 12  Crops (%)                           225 non-null    float64
 13  Other (%)                           225 non-null    float64
 14  Climate                             205 non-null    float64
 15  Birthrate                           224 non-null    float64
 16  Deathrate                           223 non-null    float64
 17  Agriculture                         212 non-null    float64
 18  Industry                            211 non-null    float64
 19  Service                             212 non-null    float64
dtypes: float64(16), int64(2), object(2)
memory usage: 35.6+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDF
count	2.270000e+02	2.270000e+02	227.000000	227.000000	224.000000	224.000000	226.(
mean	2.874028e+07	5.982270e+05	379.047137	21.165330	0.038125	35.506964	9689.8
std	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.889269	35.389899	10049.
min	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.990000	2.290000	500.(
25%	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.927500	8.150000	1900.(
50%	4.786994e+06	8.660000e+04	78.800000	0.730000	0.000000	21.000000	5550.(
75%	1.749777e+07	4.418110e+05	190.150000	10.345000	0.997500	55.705000	15700.(
max	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.060000	191.190000	55100.(

In [706]:

Out[706]:

	count	mean	std	min	25%	50%	
Population	227.0	2.874028e+07	1.178913e+08	7026.000	437624.00000	4786994.000	1.74977
Area (sq. mi.)	227.0	5.982270e+05	1.790282e+06	2.000	4647.50000	86600.000	4.41811
Pop. Density (per sq. mi.)	227.0	3.790471e+02	1.660186e+03	0.000	29.15000	78.800	1.90150
Coastline (coast/area ratio)	227.0	2.116533e+01	7.228686e+01	0.000	0.10000	0.730	1.03450
Net migration	224.0	3.812500e-02	4.889269e+00	-20.990	-0.92750	0.000	9.9750(
Infant mortality (per 1000 births)	224.0	3.550696e+01	3.538990e+01	2.290	8.15000	21.000	5.57050
GDP ($ per capita)	226.0	9.689823e+03	1.004914e+04	500.000	1900.00000	5550.000	1.57000
Literacy (%)	209.0	8.283828e+01	1.972217e+01	17.600	70.60000	92.500	9.80000
Phones (per 1000)	223.0	2.360614e+02	2.279918e+02	0.200	37.80000	176.200	3.89650
Arable (%)	225.0	1.379711e+01	1.304040e+01	0.000	3.22000	10.420	2.00000
Crops (%)	225.0	4.564222e+00	8.361470e+00	0.000	0.19000	1.030	4.44000
Other (%)	225.0	8.163831e+01	1.614083e+01	33.330	71.65000	85.700	9.54400
Climate	205.0	2.139024e+00	6.993968e-01	1.000	2.00000	2.000	3.00000
Birthrate	224.0	2.211473e+01	1.117672e+01	7.290	12.67250	18.790	2.98200
Deathrate	223.0	9.241345e+00	4.990026e+00	2.290	5.91000	7.840	1.06050
Agriculture	212.0	1.508443e-01	1.467980e-01	0.000	0.03775	0.099	2.2100(
Industry	211.0	2.827109e-01	1.382722e-01	0.020	0.19300	0.272	3.4100(
Service	212.0	5.652830e-01	1.658410e-01	0.062	0.42925	0.571	6.7850(

Exploratory Data Analysis

Let's create some visualizations. Please feel free to expand on these with your own analysis and charts!

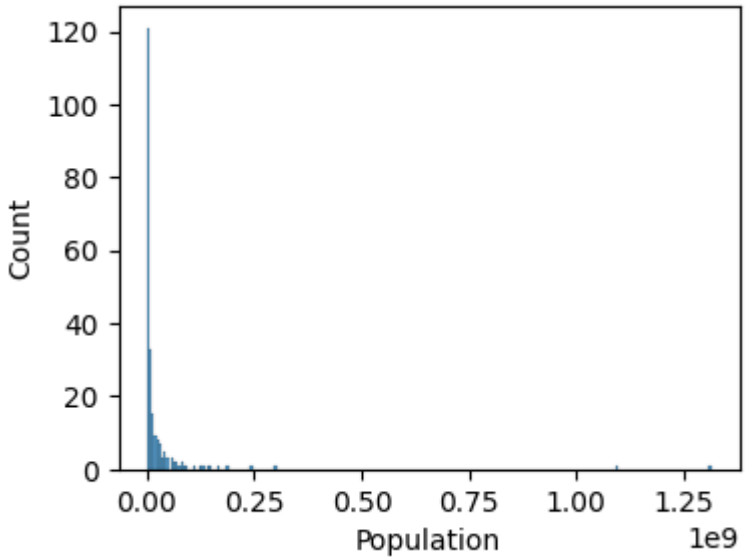
TASK: Create a histogram of the Population column.

In [12]:

```
# CODE HERE
plt.figure(figsize=(4,3))
sns.histplot(data=df,x='Population')
```

Out[12]:

```
<AxesSubplot: xlabel='Population', ylabel='Count'>
```



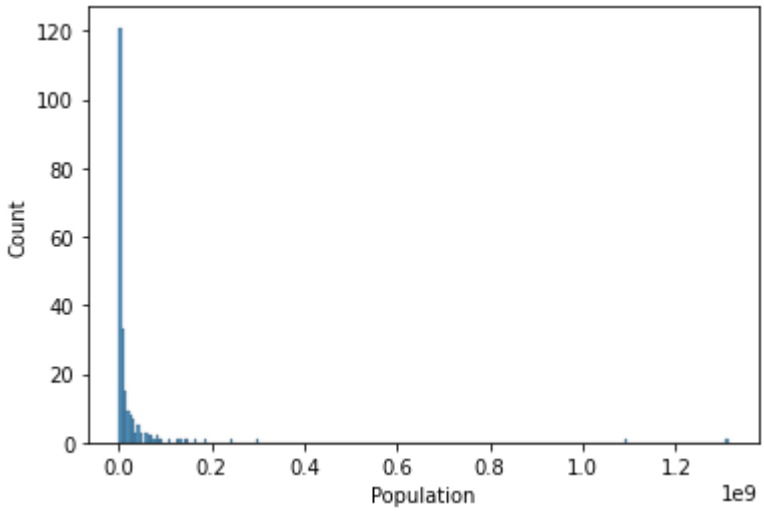
In [708]:

```

```

Out[708]:

```
<AxesSubplot:xlabel='Population', ylabel='Count'>
```



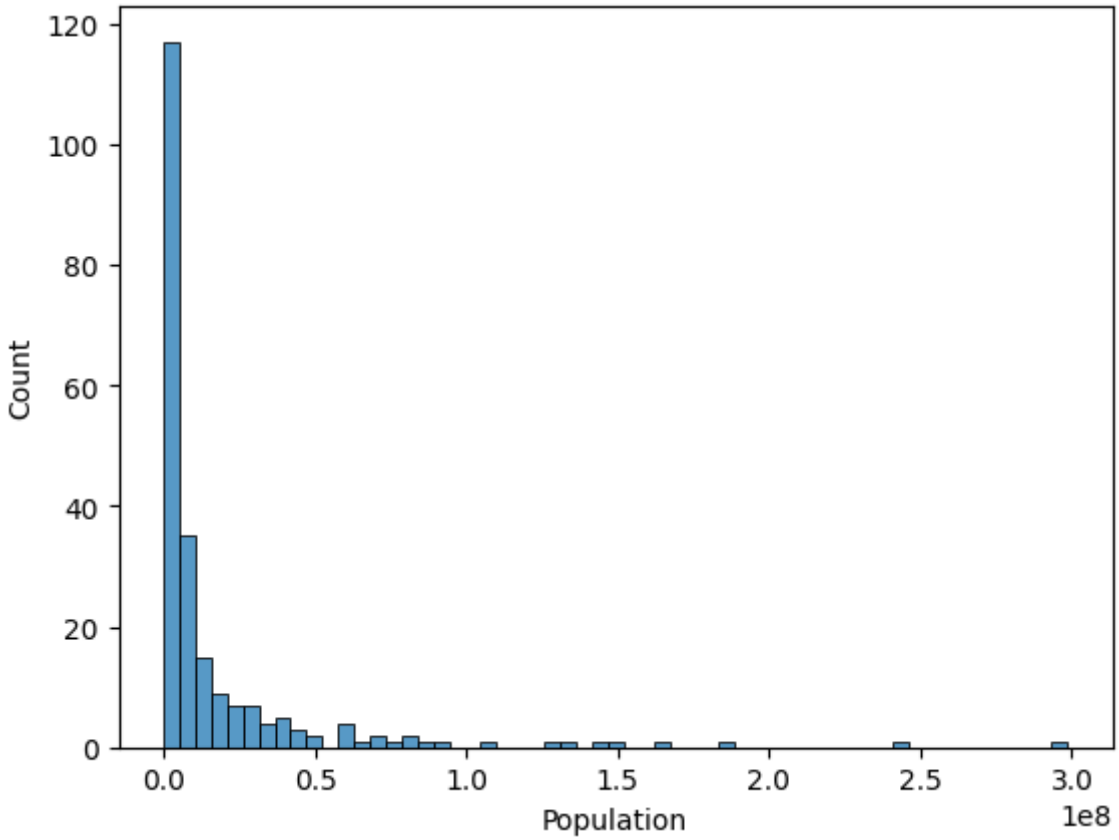
TASK: You should notice the histogram is skewed due to a few large countries, reset the X axis to only show countries with less than 0.5 billion people

In [15]:

```
#CODE HERE
sns.histplot(data=df[df['Population']<500000000],x='Population')
```

Out[15]:

<AxesSubplot: xlabel='Population', ylabel='Count'>



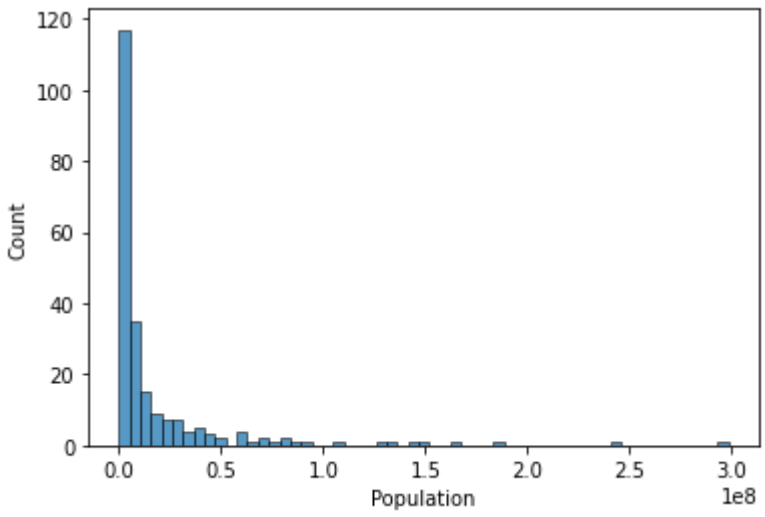
In [710]:

```

```

Out[710]:

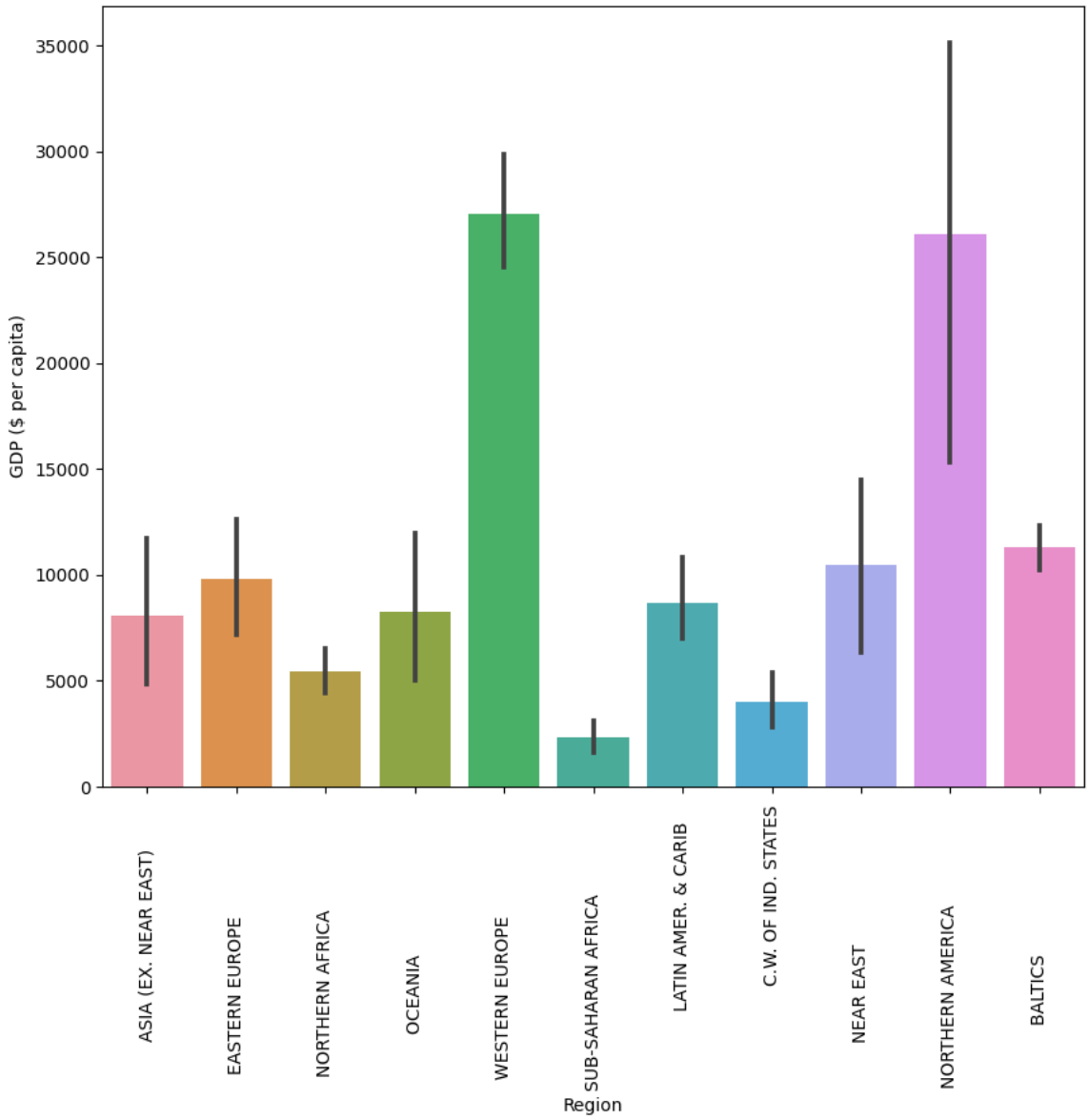
<AxesSubplot:xlabel='Population', ylabel='Count'>



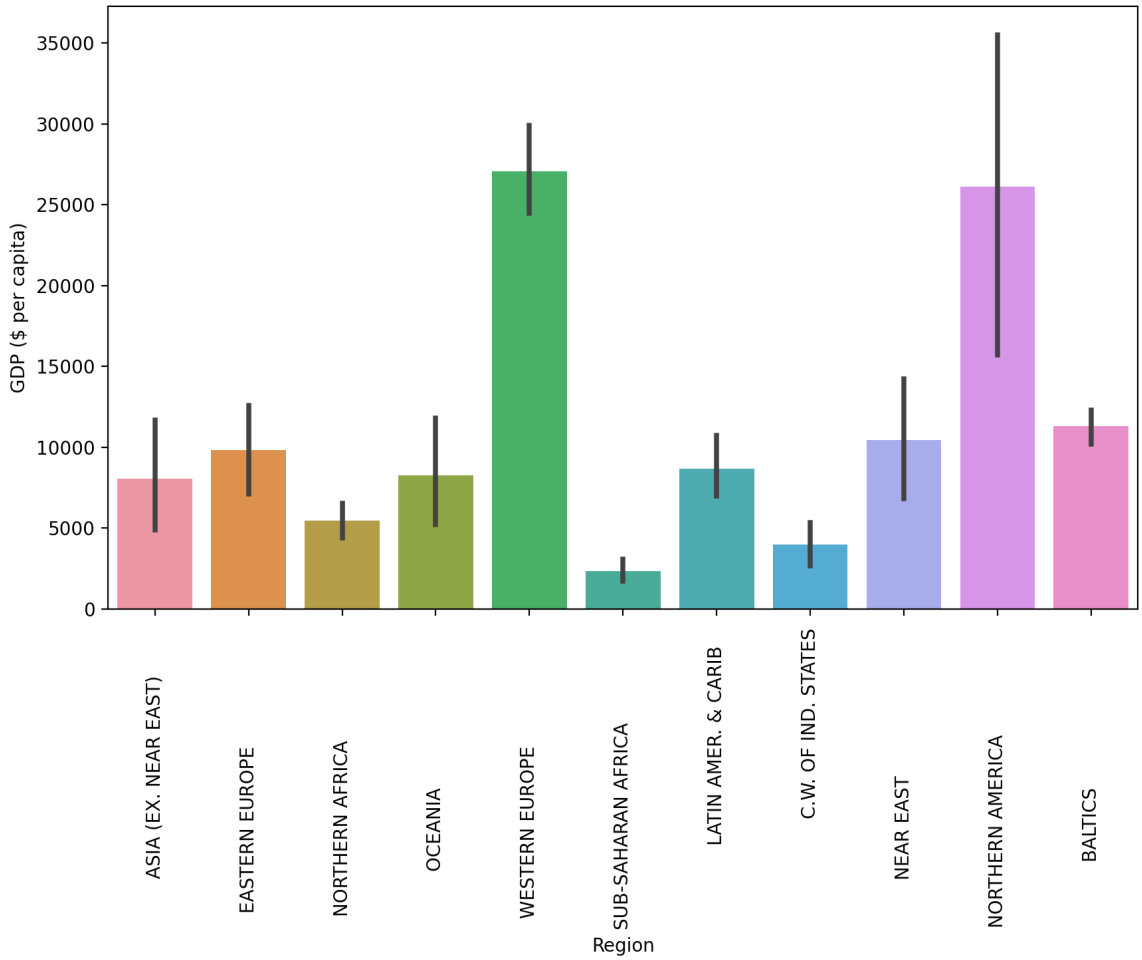
TASK: Now let's explore GDP and Regions. Create a bar chart showing the mean GDP per Capita per region (recall the black bar represents std).

In [27]:

```
# CODE HERE
plt.figure(figsize=(10,8))
sns.barplot(data=df,y='GDP ($ per capita)',x='Region')
plt.xticks(rotation=90);
```



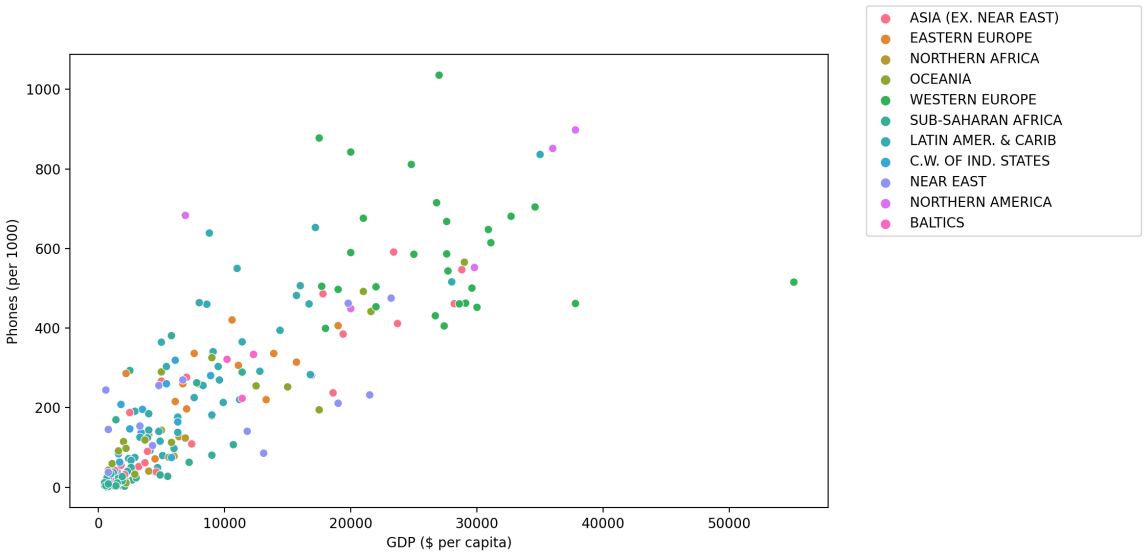
In [712]:



TASK: Create a scatterplot showing the relationship between Phones per 1000 people and the GDP per Capita. Color these points by Region.

In [53]:

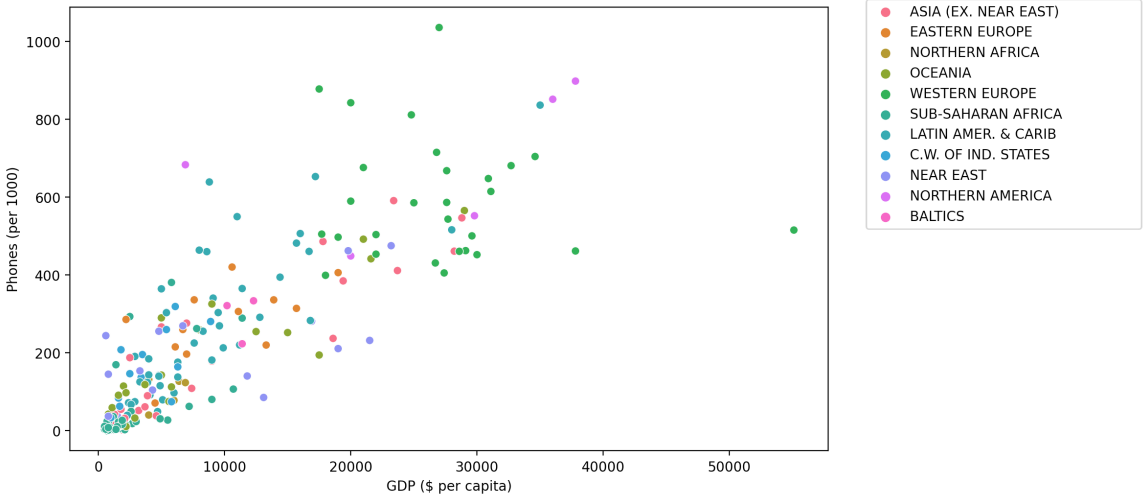
```
#CODE HERE
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df,x='GDP ($ per capita)',y='Phones (per 1000)',hue='Region')
plt.legend(loc=(1.05,.6));
```



In [714]:

Out[714]:

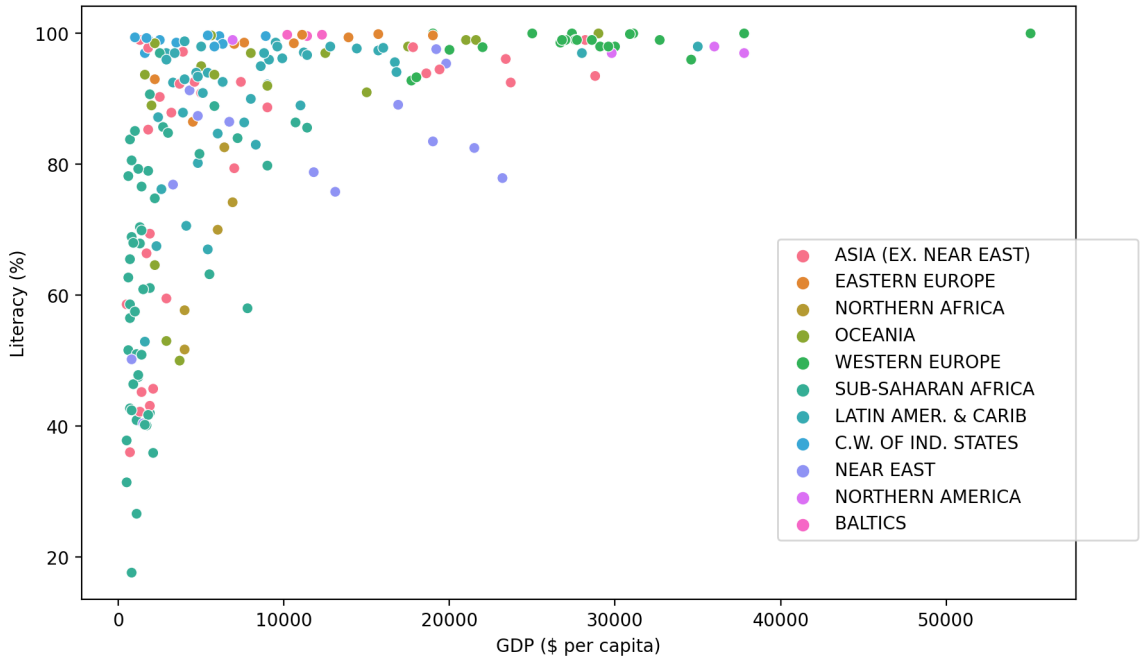
```
<matplotlib.legend.Legend at 0x194e5896160>
```



TASK: Create a scatterplot showing the relationship between GDP per Capita and Literacy (color the points by Region). What conclusions do you draw from this plot?

In [52]:

```
#CODE HERE
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df,x='GDP ($ per capita)',y='Literacy (%)',hue='Region')
plt.legend(loc=(0.7,0.1));
```



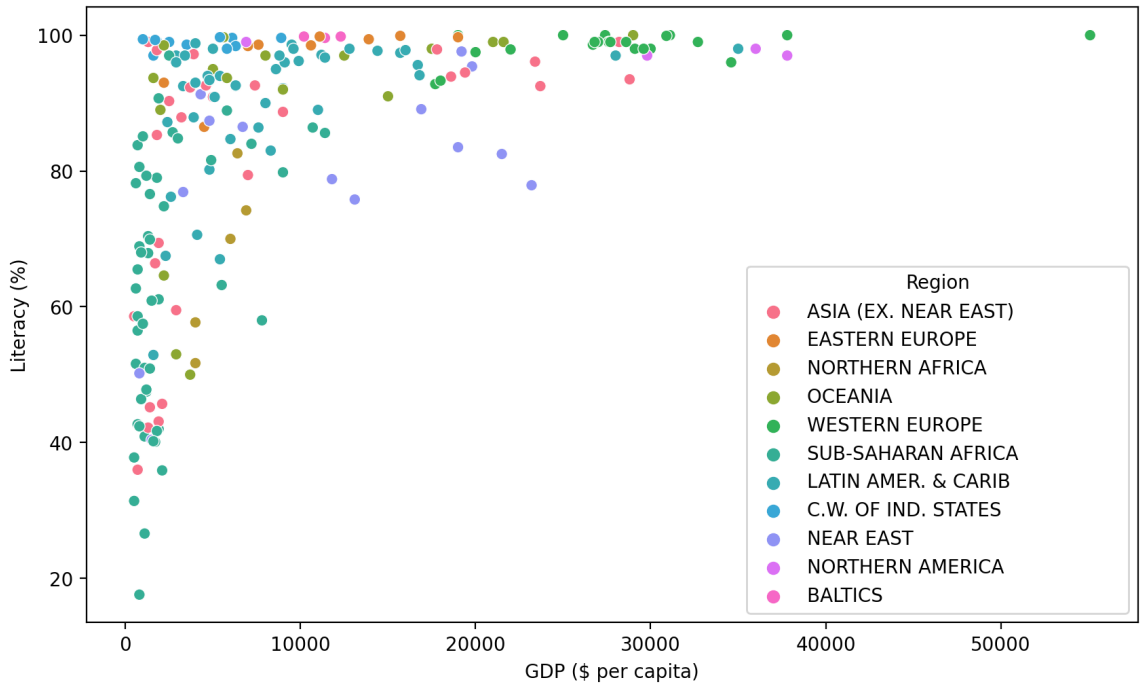
In [716]:

```

```

Out[716]:

```
<AxesSubplot:xlabel='GDP ($ per capita)', ylabel='Literacy (%)'>
```



TASK: Create a Heatmap of the Correlation between columns in the DataFrame.

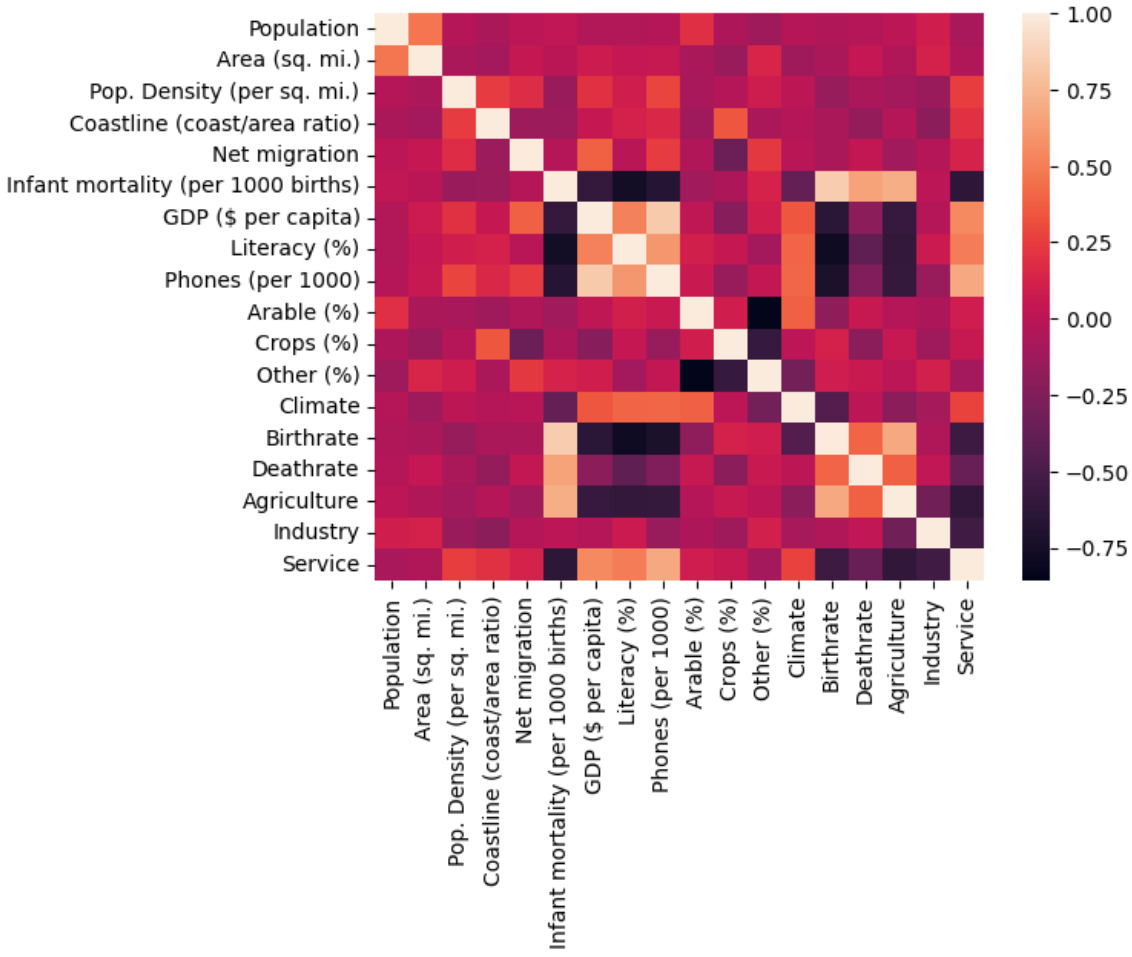
In [55]:

```
#CODE HERE
sns.heatmap(df.corr())
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_10852\2629760279.py:2: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

Out[55]:

```
<AxesSubplot: >
```

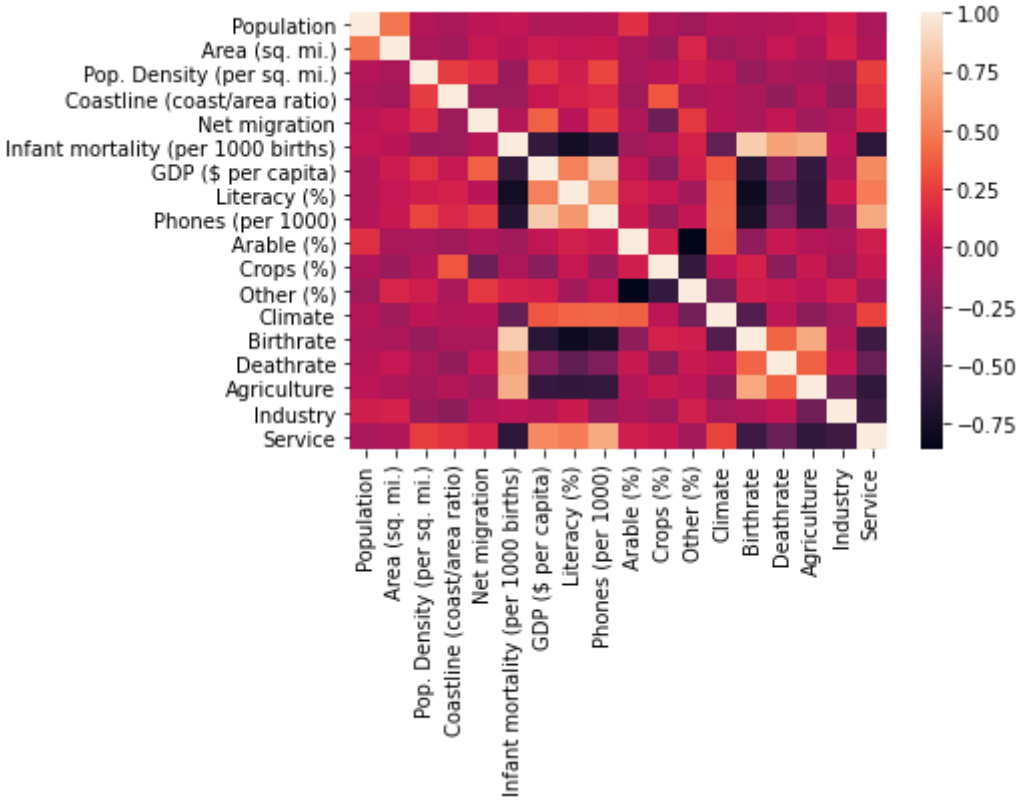


In [718]:

```
```

Out[718]:

```
<AxesSubplot:>
```



TASK: Seaborn can auto perform hierarchal clustering through the clustermap() function. Create a clustermap of the correlations between each column with this function.

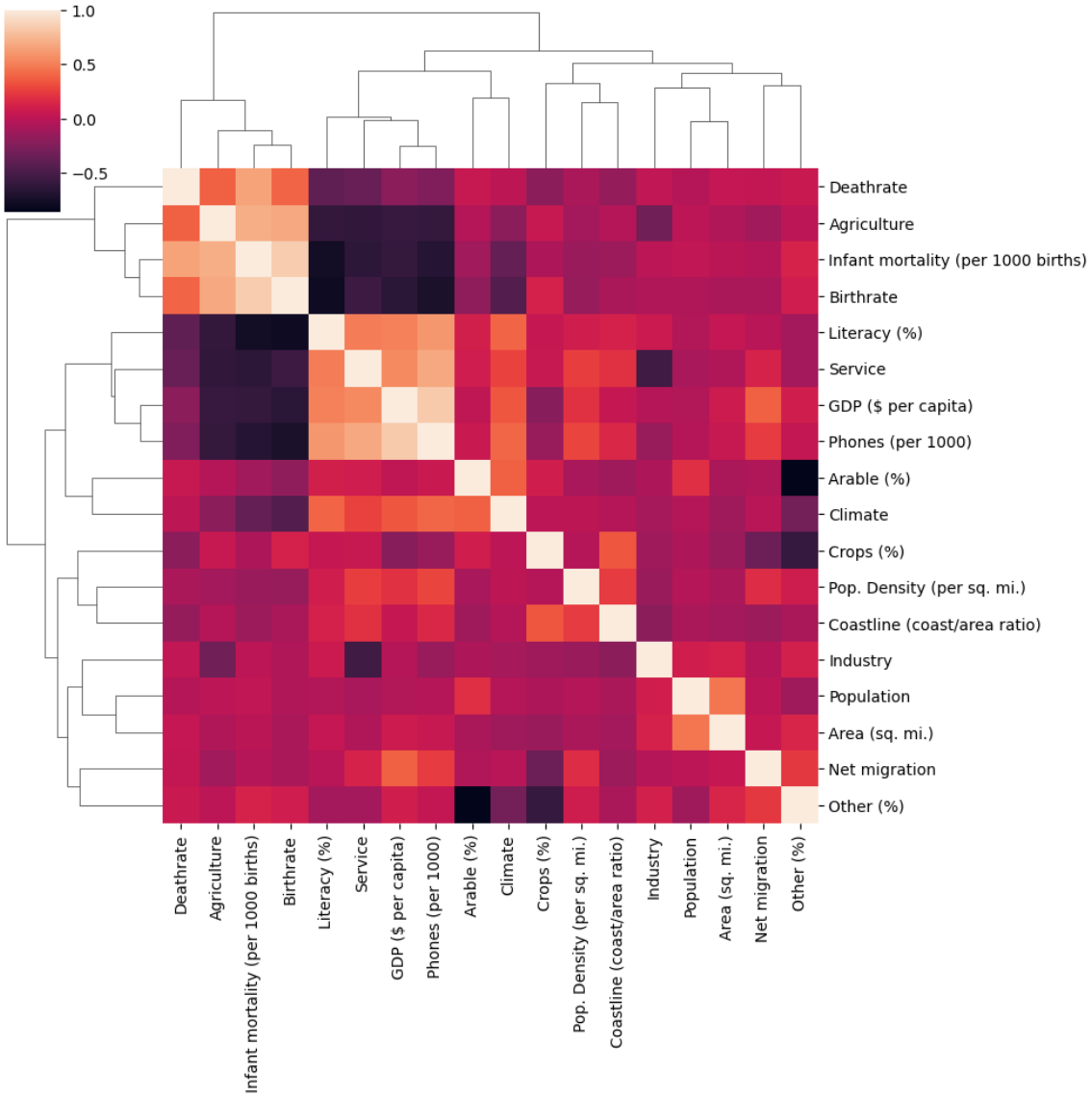
In [56]:

```
# CODE HERE
sns.clustermap(df.corr())
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_10852\3737290923.py:2: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  sns.clustermap(df.corr())
```

Out[56]:

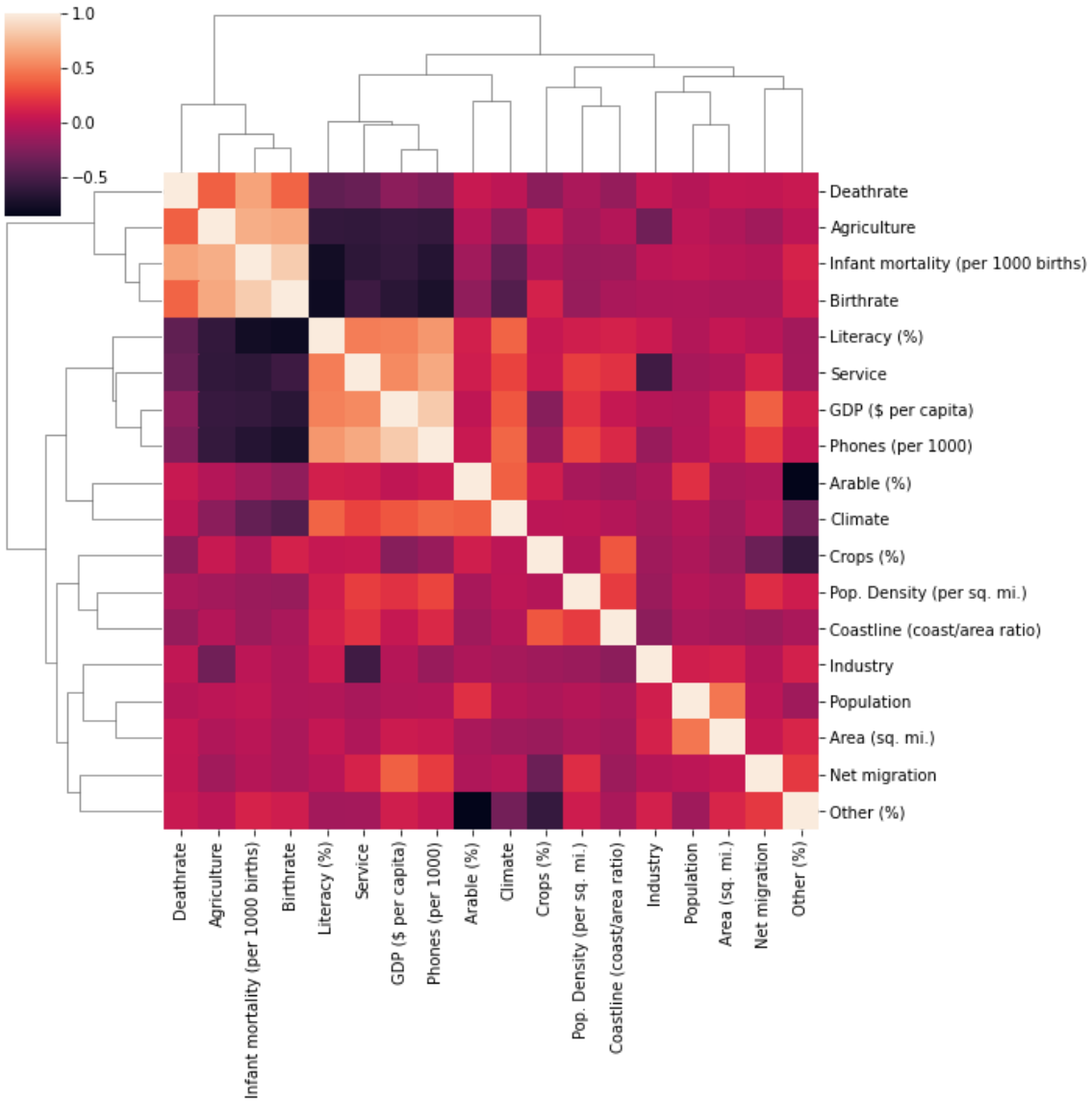
```
<seaborn.matrix.ClusterGrid at 0x1d0de070940>
```



In [720]:

Out[720]:

```
<seaborn.matrix.ClusterGrid at 0x194e3dc53d0>
```



Data Preparation and Model Discovery

Let's now prepare our data for Kmeans Clustering!

Missing Data

TASK: Report the number of missing elements per column.

In [58]:

```
#CODE HERE
df.isnull().sum()
```

Out[58]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         3
Infant mortality (per 1000 births)    3
GDP ($ per capita)                    1
Literacy (%)                         18
Phones (per 1000)                     4
Arable (%)                            2
Crops (%)                             2
Other (%)                             2
Climate                              22
Birthrate                             3
Deathrate                             4
Agriculture                          15
Industry                             16
Service                              15
dtype: int64
```

In [722]:

```

```

Out[722]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         3
Infant mortality (per 1000 births)    3
GDP ($ per capita)                    1
Literacy (%)                         18
Phones (per 1000)                     4
Arable (%)                            2
Crops (%)                             2
Other (%)                             2
Climate                              22
Birthrate                             3
Deathrate                             4
Agriculture                          15
Industry                             16
Service                              15
dtype: int64
```

TASK: What countries have NaN for Agriculture? What is the main aspect of these countries?

In [60]:

```
df[df['Agriculture'].isnull()]['Country']
```

Out[60]:

```
3            American Samoa
4                   Andorra
78                Gibraltar
80                Greenland
83                     Guam
134                Mayotte
140              Montserrat
144                   Nauru
153      N. Mariana Islands
171             Saint Helena
174    St Pierre & Miquelon
177              San Marino
208       Turks & Caicos Is
221       Wallis and Futuna
223          Western Sahara
Name: Country, dtype: object
```

In [723]:

```

```

Out[723]:

```
3            American Samoa
4                   Andorra
78                Gibraltar
80                Greenland
83                     Guam
134                Mayotte
140              Montserrat
144                   Nauru
153      N. Mariana Islands
171             Saint Helena
174    St Pierre & Miquelon
177              San Marino
208       Turks & Caicos Is
221       Wallis and Futuna
223          Western Sahara
Name: Country, dtype: object
```

TASK: You should have noticed most of these countries are tiny islands, with the exception of Greenland and Western Sahara. Go ahead and fill any of these countries missing NaN values with 0, since they are so small or essentially non-existant. There should be 15 countries in total you do this for. For a hint on how to do this, recall you can do the following:

```
df[df['feature'].isnull()]
```

In [63]:

```
df[df['Agriculture'].isnull()] = df[df['Agriculture'].isnull()].fillna(0)
```

TASK: Now check to see what is still missing by counting number of missing elements again per feature:

In [65]:

```
#CODE HERE
df.isnull().sum()
```

Out[65]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                         13
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                              18
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

In [726]:

```

```

Out[726]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                         13
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                              18
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

TASK: Notice climate is missing for a few countries, but not the Region! Let's use this to our advantage. Fill in the missing Climate values based on the mean climate value for its region.

Hints on how to do this: https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group (https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group)

In [69]:

```
# CODE HERE
df['Climate'] =df['Climate'].fillna(df.groupby('Region')['Climate'].transform('mean'))
```

In [728]:

```

```

TASK: Check again on many elements are missing:

In [70]:

```
#CODE HERE
df.isnull().sum()
```

Out[70]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                         13
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                               0
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

In [730]:

```
```

Out[730]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                         13
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                               0
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

TASK: It looks like Literacy percentage is missing. Use the same tactic as we did with Climate missing values and fill in any missing Literacy % values with the mean Literacy % of the Region.

In [71]:

```
#CODE HERE
df[df['Literacy (%)'].isnull()]
```

Out[71]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	G ca
25	Bosnia & Herzegovina	EASTERN EUROPE	4498976	51129	88.0	0.04	0.31	21.05	6
66	Faroe Islands	WESTERN EUROPE	47246	1399	33.8	79.84	1.41	6.24	22
74	Gaza Strip	NEAR EAST	1428757	360	3968.8	11.11	1.60	22.93	
85	Guernsey	WESTERN EUROPE	65409	78	838.6	64.10	3.84	4.71	20
99	Isle of Man	WESTERN EUROPE	75441	572	131.9	27.97	5.36	5.93	21
104	Jersey	WESTERN EUROPE	91084	116	785.2	60.34	2.76	5.24	24
108	Kiribati	OCEANIA	105432	811	130.0	140.94	0.00	48.52	
123	Macedonia	EASTERN EUROPE	2050554	25333	80.9	0.00	-1.45	10.09	6
185	Slovakia	EASTERN EUROPE	5439448	48845	111.4	0.00	0.30	7.41	13
187	Solomon Islands	OCEANIA	552438	28450	19.4	18.67	0.00	21.29	1
209	Tuvalu	OCEANIA	11810	26	454.2	92.31	0.00	20.03	1
220	Virgin Islands	LATIN AMER. & CARIB	108605	1910	56.9	9.84	-8.94	8.03	17
222	West Bank	NEAR EAST	2460492	5860	419.9	0.00	2.98	19.62	

In [732]:

```
```

Out[732]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	G c
25	Bosnia & Herzegovina	EASTERN EUROPE	4498976	51129	88.0	0.04	0.31	21.05	6
66	Faroe Islands	WESTERN EUROPE	47246	1399	33.8	79.84	1.41	6.24	22
74	Gaza Strip	NEAR EAST	1428757	360	3968.8	11.11	1.60	22.93	
85	Guernsey	WESTERN EUROPE	65409	78	838.6	64.10	3.84	4.71	20
99	Isle of Man	WESTERN EUROPE	75441	572	131.9	27.97	5.36	5.93	21
104	Jersey	WESTERN EUROPE	91084	116	785.2	60.34	2.76	5.24	24
108	Kiribati	OCEANIA	105432	811	130.0	140.94	0.00	48.52	
123	Macedonia	EASTERN EUROPE	2050554	25333	80.9	0.00	-1.45	10.09	6
185	Slovakia	EASTERN EUROPE	5439448	48845	111.4	0.00	0.30	7.41	13
187	Solomon Islands	OCEANIA	552438	28450	19.4	18.67	0.00	21.29	1
209	Tuvalu	OCEANIA	11810	26	454.2	92.31	0.00	20.03	1
220	Virgin Islands	LATIN AMER. & CARIB	108605	1910	56.9	9.84	-8.94	8.03	17
222	West Bank	NEAR EAST	2460492	5860	419.9	0.00	2.98	19.62	

In [72]:

```
df['Literacy (%)'] = df['Literacy (%)'].fillna(df.groupby('Region')['Literacy (%)'].tra
nsform('mean'))
```

TASK: Check again on the remaining missing values:

In [73]:

```
df.isnull().sum()
```

Out[73]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                          0
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                               0
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

In [734]:

```

```

Out[734]:

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         1
Infant mortality (per 1000 births)    1
GDP ($ per capita)                    0
Literacy (%)                          0
Phones (per 1000)                     2
Arable (%)                            1
Crops (%)                             1
Other (%)                             1
Climate                               0
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

TASK: Optional: We are now missing values for only a few countries. Go ahead and drop these countries OR feel free to fill in these last few remaining values with any preferred methodology. For simplicity, we will drop these.

In [74]:

```
# CODE HERE

df = df.dropna()
```

In [736]:

```

```

Data Feature Preparation

TASK: It is now time to prepare the data for clustering. The Country column is still a unique identifier string, so it won't be useful for clustering, since its unique for each point. Go ahead and drop this Country column.

In [737]:

```
#CODE HERE
```

In [75]:

```
X = df.drop('Country',axis=1)
```

TASK: Now let's create the X array of features, the Region column is still categorical strings, use Pandas to create dummy variables from this column to create a finalzed X matrix of continuous features along with the dummy variables for the Regions.

In [76]:

```
#COde here
X= pd.get_dummies(X,drop_first=True)
```

In [77]:

```
X.head()
```

Out[77]:

	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	Phones (per 1000)
0	31056997	647500	48.0	0.00	23.06	163.07	700.0	36.0	3.2
1	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	86.5	71.2
2	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	70.0	78.1
3	57794	199	290.4	58.29	-20.71	9.27	8000.0	97.0	259.5
4	71201	468	152.1	0.00	6.60	4.05	19000.0	100.0	497.2

5 rows × 28 columns

In [741]:

```
```

Out[741]:

	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	Phones (per 1000)
0	31056997	647500	48.0	0.00	23.06	163.07	700.0	36.0	3.2
1	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	86.5	71.2
2	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	70.0	78.1
3	57794	199	290.4	58.29	-20.71	9.27	8000.0	97.0	259.5
4	71201	468	152.1	0.00	6.60	4.05	19000.0	100.0	497.2

5 rows × 29 columns

Scaling

TASK: Due to some measurements being in terms of percentages and other metrics being total counts (population), we should scale this data first. Use Sklearn to scale the X feature matrics.

In [79]:

```
#CODE HERE
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [80]:

```
scaler_x = scaler.fit_transform(X)
```

In [81]:

```
scaler_x
```

Out[81]:

```
array([[ 0.0133285 ,  0.01855412, -0.20308668, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.21730118, -0.32370888, -0.14378531, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [ 0.02905136,  0.97784988, -0.22956327, ..., -0.31544015,
        -0.54772256, -0.36514837],
       ...,
       [-0.06726127, -0.04756396, -0.20881553, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.15081724,  0.07669798, -0.22840201, ..., -0.31544015,
         1.82574186, -0.36514837],
       [-0.14464933, -0.12356132, -0.2160153 , ..., -0.31544015,
         1.82574186, -0.36514837]])
```

In [745]:

```
```

Out[745]:

```
array([[ 0.0133285 ,  0.01855412, -0.20308668, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.21730118, -0.32370888, -0.14378531, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [ 0.02905136,  0.97784988, -0.22956327, ..., -0.31544015,
        -0.54772256, -0.36514837],
       ...,
       [-0.06726127, -0.04756396, -0.20881553, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.15081724,  0.07669798, -0.22840201, ..., -0.31544015,
         1.82574186, -0.36514837],
       [-0.14464933, -0.12356132, -0.2160153 , ..., -0.31544015,
         1.82574186, -0.36514837]])
```

Creating and Fitting Kmeans Model

TASK: Use a for loop to create and fit multiple KMeans models, testing from K=2-30 clusters. Keep track of the Sum of Squared Distances for each K value, then plot this out to create an "elbow" plot of K versus SSD. Optional: You may also want to create a bar plot showing the SSD difference from the previous cluster.

In [82]:

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

In [ ]:

```
```

In [83]:

```
#CODE HERE
ssd = []

for k in range(2,30):
    model = KMeans(n_clusters=k)
    model.fit(scaler_x)

    ssd.append(model.inertia_)
```

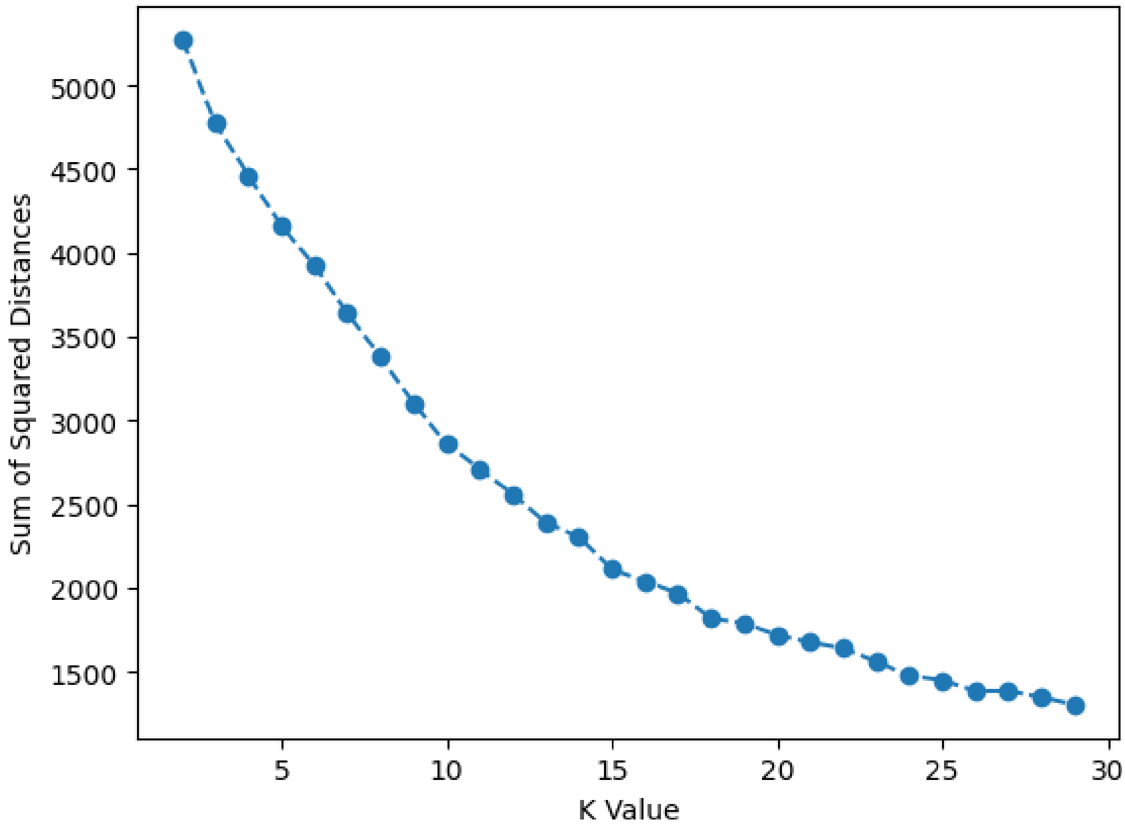
```
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
```

```
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
```

```
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

In [85]:

```
plt.plot(range(2,30),ssd,'o--')
plt.xlabel('K Value')
plt.ylabel('Sum of Squared Distances');
```



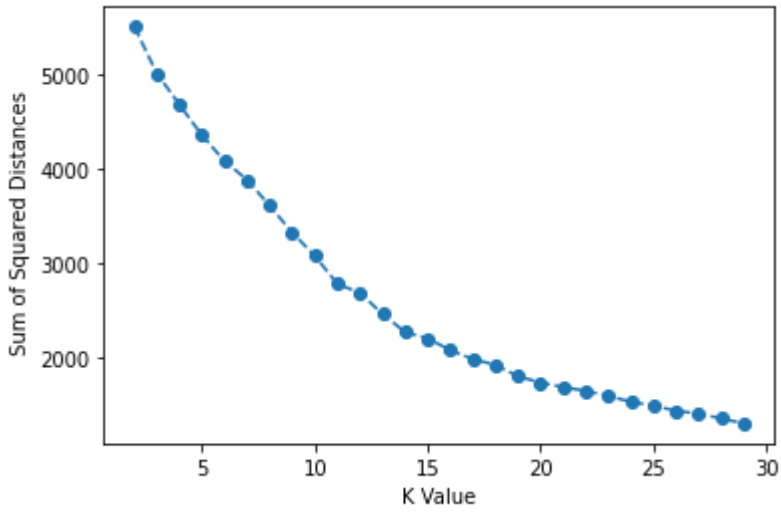
In [748]:

In [749]:

```
```

Out[749]:

```
Text(0, 0.5, ' Sum of Squared Distances')
```

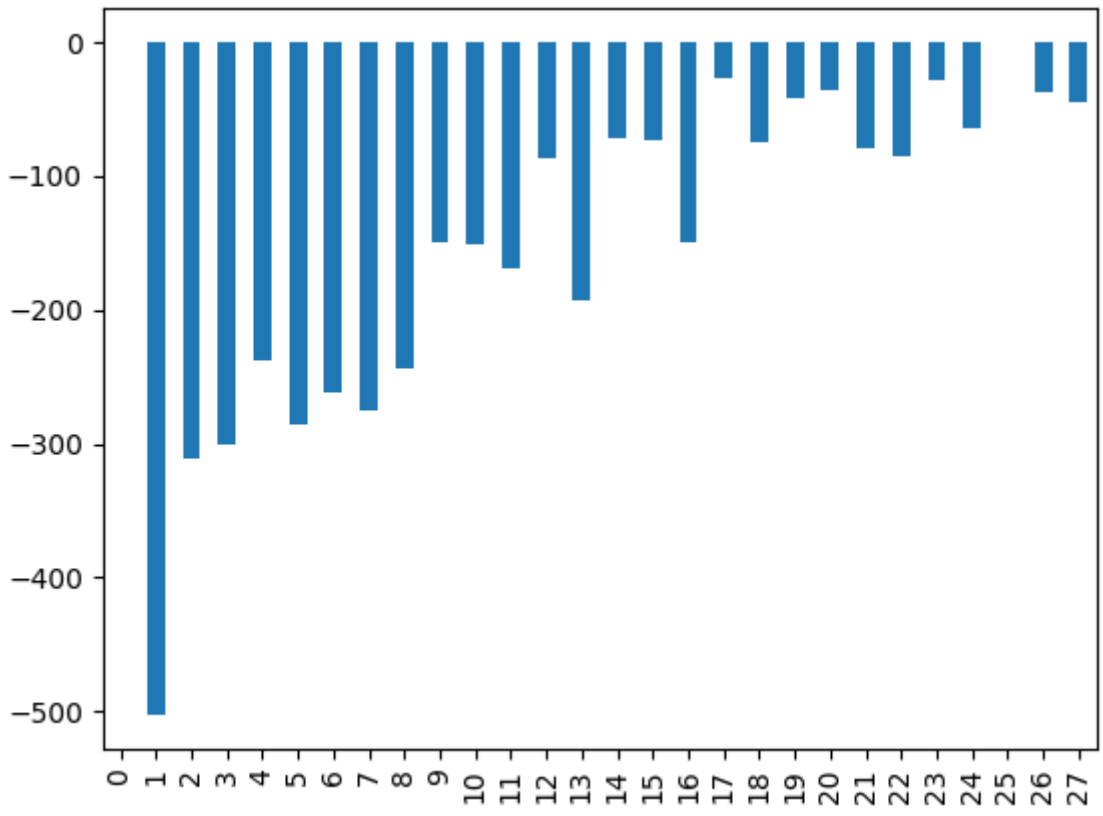


In [87]:

```
pd.Series(ssd).diff().plot(kind='bar')
```

Out[87]:

```
<AxesSubplot: >
```

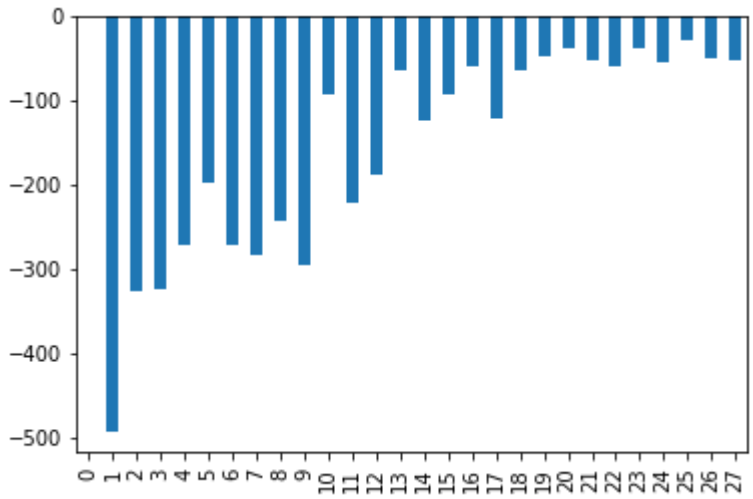


In [750]:

```
```

Out[750]:

<AxesSubplot:>



Model Interpretation

TASK: What K value do you think is a good choice? Are there multiple reasonable choices? What features are helping define these cluster choices. As this is unsupervised learning, there is no 100% correct answer here. Please feel free to jump to the solutions for a full discussion on this!.

In [751]:

```
# Nothing to really code here, but choose a K value and see what features
# are most correlated to belonging to a particular cluster!

# Remember, there is no 100% correct answer here!
```

Example Interpretation: Choosing K=3

One could say that there is a significant drop off in SSD difference at K=3 (although we can see it continues to drop off past this). What would an analysis look like for K=3? Let's explore which features are important in the decision of 3 clusters!

In [89]:

```
model = KMeans(n_clusters=3)
model.fit(scaler_x)
```

```
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

Out[89]:

```
KMeans(n_clusters=3)
```

In [91]:

```
model.labels_
```

Out[91]:

```
array([1, 0, 0, 0, 2, 1, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 1, 2, 2, 2, 0, 1,
       2, 1, 0, 2, 1, 0, 2, 0, 2, 1, 1, 1, 1, 1, 2, 0, 2, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 2, 0, 2, 2, 1, 0, 0, 0, 0, 0, 1, 1, 2, 1, 2, 0, 2,
       2, 0, 0, 1, 1, 0, 0, 2, 1, 2, 2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 2,
       2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2, 2, 0, 0, 1, 0, 0, 2, 0, 0, 1,
       2, 0, 1, 1, 0, 2, 2, 2, 2, 2, 1, 1, 0, 0, 1, 2, 0, 0, 1, 2, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 0, 0, 2, 0, 1, 1, 0, 2, 0, 1, 0, 0,
       0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1,
       0, 1, 2, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1, 2, 2, 0, 2, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 2, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1])
```

In [754]:

```

```

Out[754]:

```
array([2, 0, 0, 0, 1, 2, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 2, 1, 1, 1, 0, 2,
       1, 2, 0, 1, 2, 0, 1, 0, 1, 2, 2, 2, 2, 2, 1, 0, 1, 2, 2, 0, 0, 0,
       2, 2, 2, 0, 2, 1, 0, 1, 1, 2, 0, 0, 0, 0, 0, 2, 2, 1, 2, 1, 0, 1,
       1, 0, 0, 2, 2, 0, 0, 1, 2, 1, 1, 0, 0, 0, 0, 0, 2, 2, 0, 2, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 0, 0, 1, 0, 0, 2,
       1, 0, 2, 2, 0, 1, 1, 1, 1, 1, 2, 2, 0, 0, 2, 1, 0, 0, 2, 0, 2, 0,
       0, 0, 0, 0, 0, 2, 2, 0, 2, 1, 0, 0, 1, 0, 2, 2, 0, 1, 0, 2, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2,
       0, 2, 1, 1, 1, 0, 2, 2, 1, 0, 2, 0, 2, 1, 1, 0, 1, 0, 2, 0, 2, 0,
       0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2,
       2])
```

In [96]:

```
X['K=3 Clusters'] = model.labels_
```

In [97]:

```
X.corr()['K=3 Clusters'].sort_values()
```

Out[97]:

```
Region_LATIN AMER. & CARIB                -0.346773
Crops (%)                                 -0.282076
Birthrate                                 -0.226310
Region_OCEANIA                            -0.216328
Region_NEAR EAST                          -0.185058
Region_NORTHERN AFRICA                    -0.154500
Region_C.W. OF IND. STATES                -0.126141
Other (%)                                 -0.106341
Agriculture                               -0.101606
Coastline (coast/area ratio)              -0.085683
Infant mortality (per 1000 births)        -0.075079
Industry                                  -0.070617
Population                                -0.069697
Area (sq. mi.)                            -0.028514
Region_NORTHERN AMERICA                    0.077515
Region_SUB-SAHARAN AFRICA                  0.096770
Literacy (%)                               0.100870
Region_BALTICS                             0.171940
Pop. Density (per sq. mi.)                 0.175431
Service                                    0.235340
Region_EASTERN EUROPE                      0.285715
Arable (%)                                 0.310984
Net migration                              0.345777
Deathrate                                  0.353643
Climate                                    0.403869
Phones (per 1000)                          0.470176
Region_WESTERN EUROPE                      0.535198
GDP ($ per capita)                         0.563169
K=3 Clusters                               1.000000
Name: K=3 Clusters, dtype: float64
```

In [757]:

```

```

Out[757]:

```
Literacy (%)                          -0.419453
Region_LATIN AMER. & CARIB            -0.377533
Region_OCEANIA                        -0.248224
Crops (%)                             -0.245934
Phones (per 1000)                     -0.198737
Region_C.W. OF IND. STATES            -0.193384
Region_NEAR EAST                      -0.179732
Coastline (coast/area ratio)          -0.158318
Region_NORTHERN AFRICA                -0.151646
Service                               -0.117898
Population                            -0.062404
GDP ($ per capita)                    -0.060568
Industry                              -0.048420
Area (sq. mi.)                        -0.039735
Region_NORTHERN AMERICA               -0.027789
Pop. Density (per sq. mi.)             0.013816
Other (%)                              0.016429
Climate                                0.024573
Region_ASIA (EX. NEAR EAST)            0.028712
Region_BALTICS                         0.035283
Region_EASTERN EUROPE                  0.043691
Arable (%)                             0.084553
Region_WESTERN EUROPE                  0.109824
Net migration                          0.208539
Agriculture                            0.440815
Birthrate                              0.494413
Infant mortality (per 1000 births)     0.614130
Region_SUB-SAHARAN AFRICA              0.670927
Deathrate                              0.727801
K=3 Clusters                           1.000000
Name: K=3 Clusters, dtype: float64
```

BONUS CHALLGENGE:

Geographical Model Interpretation

The best way to interpret this model is through visualizing the clusters of countries on a map! **NOTE: THIS IS A BONUS SECTION. YOU MAY WANT TO JUMP TO THE SOLUTIONS LECTURE FOR A FULL GUIDE, SINCE WE WILL COVER TOPICS NOT PREVIOUSLY DISCUSSED AND BE HAVING A NUANCED DISCUSSION ON PERFORMANCE!**

IF YOU GET STUCK, PLEASE CHECK OUT THE SOLUTIONS LECTURE. AS THIS IS OPTIONAL AND COVERS MANY TOPICS NOT SHOWN IN ANY PREVIOUS LECTURE

TASK: Create cluster labels for a chosen K value. Based on the solutions, we believe either K=3 or K=15 are reasonable choices. But feel free to choose differently and explore.

In [98]:

```
model = KMeans(n_clusters=15)

model.fit(scaler_x)
```

```
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

Out[98]:

```
KMeans(n_clusters=15)
```

In [99]:

```
model = KMeans(n_clusters=3)

model.fit(scaler_x)
```

```
C:\Users\Chromsy\AppData\Roaming\Python\Python39\site-packages\sklearn\clu
ster\_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
nge from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

Out[99]:

```
KMeans
KMeans(n_clusters=3)
```

TASK: Let's put you in the real world! Your boss just asked you to plot out these clusters on a country level choropleth map, can you figure out how to do this? We won't step by step guide you at all on this, just show you an example result. You'll need to do the following:

1. Figure out how to install plotly library: https://plotly.com/python/getting-started/ (https://plotly.com/python/getting-started/)
2. Figure out how to create a geographical choropleth map using plotly: https://plotly.com/python/choropleth-maps/#using-builtin-country-and-state-geometries (https://plotly.com/python/choropleth-maps/#using-builtin-country-and-state-geometries)
3. You will need ISO Codes for this. Either use the wikipedia page, or use our provided file for this: **"../DATA/country_iso_codes.csv"**
4. Combine the cluster labels, ISO Codes, and Country Names to create a world map plot with plotly given what you learned in Step 1 and Step 2.

Note: This is meant to be a more realistic project, where you have a clear objective of what you need to create and accomplish and the necessary online documentation. It's up to you to piece everything together to figure it out! If you get stuck, no worries! Check out the solution lecture.

In [102]:

```
iso_codes = pd.read_csv("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\22 - K-Means Clustering\\32407460-country-iso-codes.csv")
iso_codes
```

Out[102]:

	Country	ISO Code
0	Afghanistan	AFG
1	Akrotiri and Dhekelia – See United Kingdom, The	Akrotiri and Dhekelia – See United Kingdom, The
2	Åland Islands	ALA
3	Albania	ALB
4	Algeria	DZA
...	...	...
296	Congo, Dem. Rep.	COD
297	Congo, Repub. of the	COG
298	Tanzania	TZA
299	Central African Rep.	CAF
300	Cote d'Ivoire	CIV

301 rows × 2 columns

In [768]:

```

```

Out[768]:

	Country	ISO Code
0	Afghanistan	AFG
1	Akrotiri and Dhekelia – See United Kingdom, The	Akrotiri and Dhekelia – See United Kingdom, The
2	Åland Islands	ALA
3	Albania	ALB
4	Algeria	DZA
...	...	...
296	Congo, Dem. Rep.	COD
297	Congo, Repub. of the	COG
298	Tanzania	TZA
299	Central African Rep.	CAF
300	Cote d'Ivoire	CIV

301 rows × 2 columns

In [103]:

```
iso_mapping = iso_codes.set_index('Country')['ISO Code'].to_dict()
```

In [104]:

```
iso_mapping
```

Out[104]:

```
{'Afghanistan': 'AFG',
 'Akrotiri and Dhekelia – See United Kingdom, The': 'Akrotiri and Dhekelia
– See United Kingdom, The',
 'Åland Islands': 'ALA',
 'Albania': 'ALB',
 'Algeria': 'DZA',
 'American Samoa': 'ASM',
 'Andorra': 'AND',
 'Angola': 'AGO',
 'Anguilla': 'AIA',
 'Antarctica\u200a[a]': 'ATA',
 'Antigua and Barbuda': 'ATG',
 'Argentina': 'ARG',
 'Armenia': 'ARM',
 'Aruba': 'ABW',
 'Ashmore and Cartier Islands – See Australia.': 'Ashmore and Cartier Isla
nds – See Australia.',
 'Australia\u200a[b]': 'AUS',
 'Austria': 'AUT',
 'Azerbaijan': 'AZE',
 'Bahamas (the)': 'BHS',
 'Bahrain': 'BHR',
 'Bangladesh': 'BGD',
 'Barbados': 'BRB',
 'Belarus': 'BLR',
 'Belgium': 'BEL',
 'Belize': 'BLZ',
 'Benin': 'BEN',
 'Bermuda': 'BMU',
 'Bhutan': 'BTN',
 'Bolivia (Plurinational State of)': 'BOL',
 'Bonaire\xa0Sint Eustatius\xa0Saba': 'BES',
 'Bosnia and Herzegovina': 'BIH',
 'Botswana': 'BWA',
 'Bouvet Island': 'BVT',
 'Brazil': 'BRA',
 'British Indian Ocean Territory (the)': 'IOT',
 'British Virgin Islands – See Virgin Islands (British).': 'British Virgin
Islands – See Virgin Islands (British).',
 'Brunei Darussalam\u200a[e]': 'BRN',
 'Bulgaria': 'BGR',
 'Burkina Faso': 'BFA',
 'Burma – See Myanmar.': 'Burma – See Myanmar.',
 'Burundi': 'BDI',
 'Cabo Verde\u200a[f]': 'CPV',
 'Cambodia': 'KHM',
 'Cameroon': 'CMR',
 'Canada': 'CAN',
 'Cape Verde – See Cabo Verde.': 'Cape Verde – See Cabo Verde.',
 'Caribbean Netherlands – See Bonaire, Sint Eustatius and Saba.': 'Caribbe
an Netherlands – See Bonaire, Sint Eustatius and Saba.',
 'Cayman Islands (the)': 'CYM',
 'Central African Republic (the)': 'CAF',
 'Chad': 'TCD',
 'Chile': 'CHL',
 'China': 'CHN',
 'China, The Republic of – See Taiwan (Province of China).': 'China, The R
epublic of – See Taiwan (Province of China).',
 'Christmas Island': 'CXR',
 'Clipperton Island – See France.': 'Clipperton Island – See France.',
 'Cocos (Keeling) Islands (the)': 'CCK',
```

```
 'Colombia': 'COL',
 'Comoros (the)': 'COM',
 'Congo (the Democratic Republic of the)': 'COD',
 'Congo (the)\u200a[g]': 'COG',
 'Cook Islands (the)': 'COK',
 'Coral Sea Islands – See Australia.': 'Coral Sea Islands – See Australi
a.',
 'Costa Rica': 'CRI',
 "Côte d'Ivoire\u200a[h]": 'CIV',
 'Croatia': 'HRV',
 'Cuba': 'CUB',
 'Curaçao': 'CUW',
 'Cyprus': 'CYP',
 'Czechia\u200a[i]': 'CZE',
 "Democratic People's Republic of Korea – See Korea, The Democratic Peopl
e's Republic of.": "Democratic People's Republic of Korea – See Korea, The
Democratic People's Republic of.",
 'Democratic Republic of the Congo – See Congo, The Democratic Republic of
the.': 'Democratic Republic of the Congo – See Congo, The Democratic Repub
lic of the.',
 'Denmark': 'DNK',
 'Djibouti': 'DJI',
 'Dominica': 'DMA',
 'Dominican Republic (the)': 'DOM',
 'East Timor – See Timor-Leste.': 'East Timor – See Timor-Leste.',
 'Ecuador': 'ECU',
 'Egypt': 'EGY',
 'El Salvador': 'SLV',
 'England – See United Kingdom, The.': 'England – See United Kingdom, Th
e.',
 'Equatorial Guinea': 'GNQ',
 'Eritrea': 'ERI',
 'Estonia': 'EST',
 'Eswatini\u200a[j]': 'SWZ',
 'Ethiopia': 'ETH',
 'Falkland Islands (the) [Malvinas]\u200a[k]': 'FLK',
 'Faroe Islands (the)': 'FRO',
 'Fiji': 'FJI',
 'Finland': 'FIN',
 'France\u200a[l]': 'FRA',
 'French Guiana': 'GUF',
 'French Polynesia': 'PYF',
 'French Southern Territories (the)\u200a[m]': 'ATF',
 'Gabon': 'GAB',
 'Gambia (the)': 'GMB',
 'Georgia': 'GEO',
 'Germany': 'DEU',
 'Ghana': 'GHA',
 'Gibraltar': 'GIB',
 'Great Britain – See United Kingdom, The.': 'Great Britain – See United K
ingdom, The.',
 'Greece': 'GRC',
 'Greenland': 'GRL',
 'Grenada': 'GRD',
 'Guadeloupe': 'GLP',
 'Guam': 'GUM',
 'Guatemala': 'GTM',
 'Guernsey': 'GGY',
 'Guinea': 'GIN',
 'Guinea-Bissau': 'GNB',
 'Guyana': 'GUY',
```

```
    'Haiti': 'HTI',
    'Hawaiian Islands – See United States of America, The.': 'Hawaiian Island
  s – See United States of America, The.',
    'Heard Island and McDonald Islands': 'HMD',
    'Holy See (the)\u200a[n]': 'VAT',
    'Honduras': 'HND',
    'Hong Kong': 'HKG',
    'Hungary': 'HUN',
    'Iceland': 'ISL',
    'India': 'IND',
    'Indonesia': 'IDN',
    'Iran (Islamic Republic of)': 'IRN',
    'Iraq': 'IRQ',
    'Ireland': 'IRL',
    'Isle of Man': 'IMN',
    'Israel': 'ISR',
    'Italy': 'ITA',
    "Ivory Coast – See Côte d'Ivoire.": "Ivory Coast – See Côte d'Ivoire.",
    'Jamaica': 'JAM',
    'Jan Mayen – See Svalbard and Jan Mayen.': 'Jan Mayen – See Svalbard and
  Jan Mayen.',
    'Japan': 'JPN',
    'Jersey': 'JEY',
    'Jordan': 'JOR',
    'Kazakhstan': 'KAZ',
    'Kenya': 'KEN',
    'Kiribati': 'KIR',
    "Korea (the Democratic People's Republic of)\u200a[o]": 'PRK',
    'Korea (the Republic of)\u200a[p]': 'KOR',
    'Kuwait': 'KWT',
    'Kyrgyzstan': 'KGZ',
    "Lao People's Democratic Republic (the)\u200a[q]": 'LAO',
    'Latvia': 'LVA',
    'Lebanon': 'LBN',
    'Lesotho': 'LSO',
    'Liberia': 'LBR',
    'Libya': 'LBY',
    'Liechtenstein': 'LIE',
    'Lithuania': 'LTU',
    'Luxembourg': 'LUX',
    'Macao\u200a[r]': 'MAC',
    'North Macedonia\u200a[s]': 'MKD',
    'Madagascar': 'MDG',
    'Malawi': 'MWI',
    'Malaysia': 'MYS',
    'Maldives': 'MDV',
    'Mali': 'MLI',
    'Malta': 'MLT',
    'Marshall Islands (the)': 'MHL',
    'Martinique': 'MTQ',
    'Mauritania': 'MRT',
    'Mauritius': 'MUS',
    'Mayotte': 'MYT',
    'Mexico': 'MEX',
    'Micronesia (Federated States of)': 'FSM',
    'Moldova (the Republic of)': 'MDA',
    'Monaco': 'MCO',
    'Mongolia': 'MNG',
    'Montenegro': 'MNE',
    'Montserrat': 'MSR',
    'Morocco': 'MAR',
```

```
 'Mozambique': 'MOZ',
 'Myanmar\u200a[t]': 'MMR',
 'Namibia': 'NAM',
 'Nauru': 'NRU',
 'Nepal': 'NPL',
 'Netherlands (the)': 'NLD',
 'New Caledonia': 'NCL',
 'New Zealand': 'NZL',
 'Nicaragua': 'NIC',
 'Niger (the)': 'NER',
 'Nigeria': 'NGA',
 'Niue': 'NIU',
 'Norfolk Island': 'NFK',
 "North Korea – See Korea, The Democratic People's Republic of.": "North K
orea – See Korea, The Democratic People's Republic of.",
 'Northern Ireland – See United Kingdom, The.': 'Northern Ireland – See Un
ited Kingdom, The.',
 'Northern Mariana Islands (the)': 'MNP',
 'Norway': 'NOR',
 'Oman': 'OMN',
 'Pakistan': 'PAK',
 'Palau': 'PLW',
 'Palestine, State of': 'PSE',
 'Panama': 'PAN',
 'Papua New Guinea': 'PNG',
 'Paraguay': 'PRY',
 "People's Republic of China – See China.": "People's Republic of China –
See China.",
 'Peru': 'PER',
 'Philippines (the)': 'PHL',
 'Pitcairn\u200a[u]': 'PCN',
 'Poland': 'POL',
 'Portugal': 'PRT',
 'Puerto Rico': 'PRI',
 'Qatar': 'QAT',
 'Republic of China – See Taiwan (Province of China).': 'Republic of China
– See Taiwan (Province of China).',
 'Republic of Korea – See Korea, The Republic of.': 'Republic of Korea – S
ee Korea, The Republic of.',
 'Republic of the Congo – See Congo, The.': 'Republic of the Congo – See C
ongo, The.',
 'Réunion': 'REU',
 'Romania': 'ROU',
 'Russian Federation (the)\u200a[v]': 'RUS',
 'Rwanda': 'RWA',
 'Saba – See Bonaire, Sint Eustatius and Saba.': 'Saba – See Bonaire, Sint
Eustatius and Saba.',
 'Sahrawi Arab Democratic Republic – See Western Sahara.': 'Sahrawi Arab D
emocratic Republic – See Western Sahara.',
 'Saint Barthélemy': 'BLM',
 'Saint Helena\xa0Ascension Island\xa0Tristan da Cunha': 'SHN',
 'Saint Kitts and Nevis': 'KNA',
 'Saint Lucia': 'LCA',
 'Saint Martin (French part)': 'MAF',
 'Saint Pierre and Miquelon': 'SPM',
 'Saint Vincent and the Grenadines': 'VCT',
 'Samoa': 'WSM',
 'San Marino': 'SMR',
 'Sao Tome and Principe': 'STP',
 'Saudi Arabia': 'SAU',
 'Scotland – See United Kingdom, The.': 'Scotland – See United Kingdom, Th
```

```
e.',
 'Senegal': 'SEN',
 'Serbia': 'SRB',
 'Seychelles': 'SYC',
 'Sierra Leone': 'SLE',
 'Singapore': 'SGP',
 'Sint Eustatius – See Bonaire, Sint Eustatius and Saba.': 'Sint Eustatius
– See Bonaire, Sint Eustatius and Saba.',
 'Sint Maarten (Dutch part)': 'SXM',
 'Slovakia': 'SVK',
 'Slovenia': 'SVN',
 'Solomon Islands': 'SLB',
 'Somalia': 'SOM',
 'South Africa': 'ZAF',
 'South Georgia and the South Sandwich Islands': 'SGS',
 'South Korea – See Korea, The Republic of.': 'South Korea – See Korea, Th
e Republic of.',
 'South Sudan': 'SSD',
 'Spain': 'ESP',
 'Sri Lanka': 'LKA',
 'Sudan (the)': 'SDN',
 'Suriname': 'SUR',
 'Svalbard\xa0Jan Mayen': 'SJM',
 'Sweden': 'SWE',
 'Switzerland': 'CHE',
 'Syrian Arab Republic (the)\u200a[x]': 'SYR',
 'Taiwan (Province of China)\u200a[y]': 'TWN',
 'Tajikistan': 'TJK',
 'Tanzania, the United Republic of': 'TZA',
 'Thailand': 'THA',
 'Timor-Leste\u200a[aa]': 'TLS',
 'Togo': 'TGO',
 'Tokelau': 'TKL',
 'Tonga': 'TON',
 'Trinidad and Tobago': 'TTO',
 'Tunisia': 'TUN',
 'Turkey': 'TUR',
 'Turkmenistan': 'TKM',
 'Turks and Caicos Islands (the)': 'TCA',
 'Tuvalu': 'TUV',
 'Uganda': 'UGA',
 'Ukraine': 'UKR',
 'United Arab Emirates (the)': 'ARE',
 'United Kingdom of Great Britain and Northern Ireland (the)': 'GBR',
 'United States Minor Outlying Islands (the)\u200a[ac]': 'UMI',
 'United States of America (the)': 'USA',
 'United States Virgin Islands – See Virgin Islands (U.S.).': 'United Stat
es Virgin Islands – See Virgin Islands (U.S.).',
 'Uruguay': 'URY',
 'Uzbekistan': 'UZB',
 'Vanuatu': 'VUT',
 'Vatican City – See Holy See, The.': 'Vatican City – See Holy See, The.',
 'Venezuela (Bolivarian Republic of)': 'VEN',
 'Viet Nam\u200a[ae]': 'VNM',
 'Virgin Islands (British)\u200a[af]': 'VGB',
 'Virgin Islands (U.S.)\u200a[ag]': 'VIR',
 'Wales – See United Kingdom, The.': 'Wales – See United Kingdom, The.',
 'Wallis and Futuna': 'WLF',
 'Western Sahara\u200a[ah]': 'ESH',
 'Yemen': 'YEM',
 'Zambia': 'ZMB',
```

```
    'Zimbabwe': 'ZWE',
    'United States': 'USA',
    'United Kingdom': 'GBR',
    'Venezuela': 'VEN',
    'Australia': 'AUS',
    'Iran': 'IRN',
    'France': 'FRA',
    'Russia': 'RUS',
    'Korea, North': 'PRK',
    'Korea, South': 'KOR',
    'Myanmar': 'MMR',
    'Burma': 'MMR',
    'Vietnam': 'VNM',
    'Laos': 'LAO',
    'Bolivia': 'BOL',
    'Niger': 'NER',
    'Sudan': 'SDN',
    'Congo, Dem. Rep.': 'COD',
    'Congo, Repub. of the': 'COG',
    'Tanzania': 'TZA',
    'Central African Rep.': 'CAF',
    "Cote d'Ivoire": 'CIV'}
```

In [770]:

Out[770]:

```
{'Afghanistan': 'AFG',
 'Akrotiri and Dhekelia – See United Kingdom, The': 'Akrotiri and Dhekelia
– See United Kingdom, The',
 'Åland Islands': 'ALA',
 'Albania': 'ALB',
 'Algeria': 'DZA',
 'American Samoa': 'ASM',
 'Andorra': 'AND',
 'Angola': 'AGO',
 'Anguilla': 'AIA',
 'Antarctica\u200a[a]': 'ATA',
 'Antigua and Barbuda': 'ATG',
 'Argentina': 'ARG',
 'Armenia': 'ARM',
 'Aruba': 'ABW',
 'Ashmore and Cartier Islands – See Australia.': 'Ashmore and Cartier Isla
nds – See Australia.',
 'Australia\u200a[b]': 'AUS',
 'Austria': 'AUT',
 'Azerbaijan': 'AZE',
 'Bahamas (the)': 'BHS',
 'Bahrain': 'BHR',
 'Bangladesh': 'BGD',
 'Barbados': 'BRB',
 'Belarus': 'BLR',
 'Belgium': 'BEL',
 'Belize': 'BLZ',
 'Benin': 'BEN',
 'Bermuda': 'BMU',
 'Bhutan': 'BTN',
 'Bolivia (Plurinational State of)': 'BOL',
 'Bonaire\xa0Sint Eustatius\xa0Saba': 'BES',
 'Bosnia and Herzegovina': 'BIH',
 'Botswana': 'BWA',
 'Bouvet Island': 'BVT',
 'Brazil': 'BRA',
 'British Indian Ocean Territory (the)': 'IOT',
 'British Virgin Islands – See Virgin Islands (British).': 'British Virgin
Islands – See Virgin Islands (British).',
 'Brunei Darussalam\u200a[e]': 'BRN',
 'Bulgaria': 'BGR',
 'Burkina Faso': 'BFA',
 'Burma – See Myanmar.': 'Burma – See Myanmar.',
 'Burundi': 'BDI',
 'Cabo Verde\u200a[f]': 'CPV',
 'Cambodia': 'KHM',
 'Cameroon': 'CMR',
 'Canada': 'CAN',
 'Cape Verde – See Cabo Verde.': 'Cape Verde – See Cabo Verde.',
 'Caribbean Netherlands – See Bonaire, Sint Eustatius and Saba.': 'Caribbe
an Netherlands – See Bonaire, Sint Eustatius and Saba.',
 'Cayman Islands (the)': 'CYM',
 'Central African Republic (the)': 'CAF',
 'Chad': 'TCD',
 'Chile': 'CHL',
 'China': 'CHN',
 'China, The Republic of – See Taiwan (Province of China).': 'China, The R
epublic of – See Taiwan (Province of China).',
 'Christmas Island': 'CXR',
 'Clipperton Island – See France.': 'Clipperton Island – See France.',
 'Cocos (Keeling) Islands (the)': 'CCK',
```

```
 'Colombia': 'COL',
 'Comoros (the)': 'COM',
 'Congo (the Democratic Republic of the)': 'COD',
 'Congo (the)\u200a[g]': 'COG',
 'Cook Islands (the)': 'COK',
 'Coral Sea Islands – See Australia.': 'Coral Sea Islands – See Australi
a.',
 'Costa Rica': 'CRI',
 "Côte d'Ivoire\u200a[h]": 'CIV',
 'Croatia': 'HRV',
 'Cuba': 'CUB',
 'Curaçao': 'CUW',
 'Cyprus': 'CYP',
 'Czechia\u200a[i]': 'CZE',
 "Democratic People's Republic of Korea – See Korea, The Democratic Peopl
e's Republic of.": "Democratic People's Republic of Korea – See Korea, The
Democratic People's Republic of.",
 'Democratic Republic of the Congo – See Congo, The Democratic Republic of
the.': 'Democratic Republic of the Congo – See Congo, The Democratic Repub
lic of the.',
 'Denmark': 'DNK',
 'Djibouti': 'DJI',
 'Dominica': 'DMA',
 'Dominican Republic (the)': 'DOM',
 'East Timor – See Timor-Leste.': 'East Timor – See Timor-Leste.',
 'Ecuador': 'ECU',
 'Egypt': 'EGY',
 'El Salvador': 'SLV',
 'England – See United Kingdom, The.': 'England – See United Kingdom, Th
e.',
 'Equatorial Guinea': 'GNQ',
 'Eritrea': 'ERI',
 'Estonia': 'EST',
 'Eswatini\u200a[j]': 'SWZ',
 'Ethiopia': 'ETH',
 'Falkland Islands (the) [Malvinas]\u200a[k]': 'FLK',
 'Faroe Islands (the)': 'FRO',
 'Fiji': 'FJI',
 'Finland': 'FIN',
 'France\u200a[l]': 'FRA',
 'French Guiana': 'GUF',
 'French Polynesia': 'PYF',
 'French Southern Territories (the)\u200a[m]': 'ATF',
 'Gabon': 'GAB',
 'Gambia (the)': 'GMB',
 'Georgia': 'GEO',
 'Germany': 'DEU',
 'Ghana': 'GHA',
 'Gibraltar': 'GIB',
 'Great Britain – See United Kingdom, The.': 'Great Britain – See United K
ingdom, The.',
 'Greece': 'GRC',
 'Greenland': 'GRL',
 'Grenada': 'GRD',
 'Guadeloupe': 'GLP',
 'Guam': 'GUM',
 'Guatemala': 'GTM',
 'Guernsey': 'GGY',
 'Guinea': 'GIN',
 'Guinea-Bissau': 'GNB',
 'Guyana': 'GUY',
```

```
 'Haiti': 'HTI',
 'Hawaiian Islands – See United States of America, The.': 'Hawaiian Island
s – See United States of America, The.',
 'Heard Island and McDonald Islands': 'HMD',
 'Holy See (the)\u200a[n]': 'VAT',
 'Honduras': 'HND',
 'Hong Kong': 'HKG',
 'Hungary': 'HUN',
 'Iceland': 'ISL',
 'India': 'IND',
 'Indonesia': 'IDN',
 'Iran (Islamic Republic of)': 'IRN',
 'Iraq': 'IRQ',
 'Ireland': 'IRL',
 'Isle of Man': 'IMN',
 'Israel': 'ISR',
 'Italy': 'ITA',
 "Ivory Coast – See Côte d'Ivoire.": "Ivory Coast – See Côte d'Ivoire.",
 'Jamaica': 'JAM',
 'Jan Mayen – See Svalbard and Jan Mayen.': 'Jan Mayen – See Svalbard and
Jan Mayen.',
 'Japan': 'JPN',
 'Jersey': 'JEY',
 'Jordan': 'JOR',
 'Kazakhstan': 'KAZ',
 'Kenya': 'KEN',
 'Kiribati': 'KIR',
 "Korea (the Democratic People's Republic of)\u200a[o]": 'PRK',
 'Korea (the Republic of)\u200a[p]': 'KOR',
 'Kuwait': 'KWT',
 'Kyrgyzstan': 'KGZ',
 "Lao People's Democratic Republic (the)\u200a[q]": 'LAO',
 'Latvia': 'LVA',
 'Lebanon': 'LBN',
 'Lesotho': 'LSO',
 'Liberia': 'LBR',
 'Libya': 'LBY',
 'Liechtenstein': 'LIE',
 'Lithuania': 'LTU',
 'Luxembourg': 'LUX',
 'Macao\u200a[r]': 'MAC',
 'North Macedonia\u200a[s]': 'MKD',
 'Madagascar': 'MDG',
 'Malawi': 'MWI',
 'Malaysia': 'MYS',
 'Maldives': 'MDV',
 'Mali': 'MLI',
 'Malta': 'MLT',
 'Marshall Islands (the)': 'MHL',
 'Martinique': 'MTQ',
 'Mauritania': 'MRT',
 'Mauritius': 'MUS',
 'Mayotte': 'MYT',
 'Mexico': 'MEX',
 'Micronesia (Federated States of)': 'FSM',
 'Moldova (the Republic of)': 'MDA',
 'Monaco': 'MCO',
 'Mongolia': 'MNG',
 'Montenegro': 'MNE',
 'Montserrat': 'MSR',
 'Morocco': 'MAR',
```

```
 'Mozambique': 'MOZ',
 'Myanmar\u200a[t]': 'MMR',
 'Namibia': 'NAM',
 'Nauru': 'NRU',
 'Nepal': 'NPL',
 'Netherlands (the)': 'NLD',
 'New Caledonia': 'NCL',
 'New Zealand': 'NZL',
 'Nicaragua': 'NIC',
 'Niger (the)': 'NER',
 'Nigeria': 'NGA',
 'Niue': 'NIU',
 'Norfolk Island': 'NFK',
 "North Korea – See Korea, The Democratic People's Republic of.": "North K
orea – See Korea, The Democratic People's Republic of.",
 'Northern Ireland – See United Kingdom, The.': 'Northern Ireland – See Un
ited Kingdom, The.',
 'Northern Mariana Islands (the)': 'MNP',
 'Norway': 'NOR',
 'Oman': 'OMN',
 'Pakistan': 'PAK',
 'Palau': 'PLW',
 'Palestine, State of': 'PSE',
 'Panama': 'PAN',
 'Papua New Guinea': 'PNG',
 'Paraguay': 'PRY',
 "People's Republic of China – See China.": "People's Republic of China –
See China.",
 'Peru': 'PER',
 'Philippines (the)': 'PHL',
 'Pitcairn\u200a[u]': 'PCN',
 'Poland': 'POL',
 'Portugal': 'PRT',
 'Puerto Rico': 'PRI',
 'Qatar': 'QAT',
 'Republic of China – See Taiwan (Province of China).': 'Republic of China
– See Taiwan (Province of China).',
 'Republic of Korea – See Korea, The Republic of.': 'Republic of Korea – S
ee Korea, The Republic of.',
 'Republic of the Congo – See Congo, The.': 'Republic of the Congo – See C
ongo, The.',
 'Réunion': 'REU',
 'Romania': 'ROU',
 'Russian Federation (the)\u200a[v]': 'RUS',
 'Rwanda': 'RWA',
 'Saba – See Bonaire, Sint Eustatius and Saba.': 'Saba – See Bonaire, Sint
Eustatius and Saba.',
 'Sahrawi Arab Democratic Republic – See Western Sahara.': 'Sahrawi Arab D
emocratic Republic – See Western Sahara.',
 'Saint Barthélemy': 'BLM',
 'Saint Helena\xa0Ascension Island\xa0Tristan da Cunha': 'SHN',
 'Saint Kitts and Nevis': 'KNA',
 'Saint Lucia': 'LCA',
 'Saint Martin (French part)': 'MAF',
 'Saint Pierre and Miquelon': 'SPM',
 'Saint Vincent and the Grenadines': 'VCT',
 'Samoa': 'WSM',
 'San Marino': 'SMR',
 'Sao Tome and Principe': 'STP',
 'Saudi Arabia': 'SAU',
 'Scotland – See United Kingdom, The.': 'Scotland – See United Kingdom, Th
```

```
e.',
 'Senegal': 'SEN',
 'Serbia': 'SRB',
 'Seychelles': 'SYC',
 'Sierra Leone': 'SLE',
 'Singapore': 'SGP',
 'Sint Eustatius – See Bonaire, Sint Eustatius and Saba.': 'Sint Eustatius
– See Bonaire, Sint Eustatius and Saba.',
 'Sint Maarten (Dutch part)': 'SXM',
 'Slovakia': 'SVK',
 'Slovenia': 'SVN',
 'Solomon Islands': 'SLB',
 'Somalia': 'SOM',
 'South Africa': 'ZAF',
 'South Georgia and the South Sandwich Islands': 'SGS',
 'South Korea – See Korea, The Republic of.': 'South Korea – See Korea, Th
e Republic of.',
 'South Sudan': 'SSD',
 'Spain': 'ESP',
 'Sri Lanka': 'LKA',
 'Sudan (the)': 'SDN',
 'Suriname': 'SUR',
 'Svalbard\xa0Jan Mayen': 'SJM',
 'Sweden': 'SWE',
 'Switzerland': 'CHE',
 'Syrian Arab Republic (the)\u200a[x]': 'SYR',
 'Taiwan (Province of China)\u200a[y]': 'TWN',
 'Tajikistan': 'TJK',
 'Tanzania, the United Republic of': 'TZA',
 'Thailand': 'THA',
 'Timor-Leste\u200a[aa]': 'TLS',
 'Togo': 'TGO',
 'Tokelau': 'TKL',
 'Tonga': 'TON',
 'Trinidad and Tobago': 'TTO',
 'Tunisia': 'TUN',
 'Turkey': 'TUR',
 'Turkmenistan': 'TKM',
 'Turks and Caicos Islands (the)': 'TCA',
 'Tuvalu': 'TUV',
 'Uganda': 'UGA',
 'Ukraine': 'UKR',
 'United Arab Emirates (the)': 'ARE',
 'United Kingdom of Great Britain and Northern Ireland (the)': 'GBR',
 'United States Minor Outlying Islands (the)\u200a[ac]': 'UMI',
 'United States of America (the)': 'USA',
 'United States Virgin Islands – See Virgin Islands (U.S.).': 'United Stat
es Virgin Islands – See Virgin Islands (U.S.).',
 'Uruguay': 'URY',
 'Uzbekistan': 'UZB',
 'Vanuatu': 'VUT',
 'Vatican City – See Holy See, The.': 'Vatican City – See Holy See, The.',
 'Venezuela (Bolivarian Republic of)': 'VEN',
 'Viet Nam\u200a[ae]': 'VNM',
 'Virgin Islands (British)\u200a[af]': 'VGB',
 'Virgin Islands (U.S.)\u200a[ag]': 'VIR',
 'Wales – See United Kingdom, The.': 'Wales – See United Kingdom, The.',
 'Wallis and Futuna': 'WLF',
 'Western Sahara\u200a[ah]': 'ESH',
 'Yemen': 'YEM',
 'Zambia': 'ZMB',
```

```
 'Zimbabwe': 'ZWE',
 'United States': 'USA',
 'United Kingdom': 'GBR',
 'Venezuela': 'VEN',
 'Australia': 'AUS',
 'Iran': 'IRN',
 'France': 'FRA',
 'Russia': 'RUS',
 'Korea, North': 'PRK',
 'Korea, South': 'KOR',
 'Myanmar': 'MMR',
 'Burma': 'MMR',
 'Vietnam': 'VNM',
 'Laos': 'LAO',
 'Bolivia': 'BOL',
 'Niger': 'NER',
 'Sudan': 'SDN',
 'Congo, Dem. Rep.': 'COD',
 'Congo, Repub. of the': 'COG',
 'Tanzania': 'TZA',
 'Central African Rep.': 'CAF',
 "Cote d'Ivoire": 'CIV'}
```

In [105]:

```
df['ISO Code'] = df['Country'].map(iso_mapping)
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_10852\1751201090.py:1: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['ISO Code'] = df['Country'].map(iso_mapping)
```

In [106]:

```
df['Cluster'] = model.labels_
```

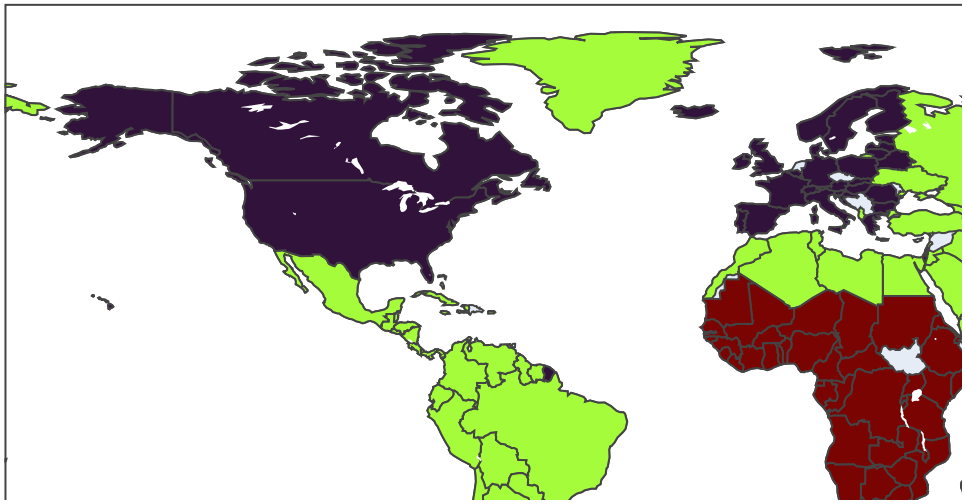
```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_10852\963081469.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Cluster'] = model.labels_
```

In [108]:

```
import plotly.express as px

fig = px.choropleth(df, locations="ISO Code",
                    color="Cluster", # lifeExp is a column of gapminder
                    hover_name="Country", # column to add to hover information
                    color_continuous_scale='Turbo'
                    )
fig.show()
```



In [773]: