
[(http://www.pieriandata.com)](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)](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('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
```

In [4]:

```
df.head()
```

Out[4]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GD ca
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	7
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	45
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	60
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	80
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	190

In [5]:

```
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 [6]:

```
df.describe().transpose()
```

Out[6]:

	count	mean	std	min	25%	50%	
Population	227.0	2.874028e+07	1.178913e+08	7026.000	437624.00000	4786994.000	1.749777e
Area (sq. mi.)	227.0	5.982270e+05	1.790282e+06	2.000	4647.50000	86600.000	4.418110e
Pop. Density (per sq. mi.)	227.0	3.790471e+02	1.660186e+03	0.000	29.15000	78.800	1.901500e
Coastline (coast/area ratio)	227.0	2.116533e+01	7.228686e+01	0.000	0.10000	0.730	1.034500e
Net migration	224.0	3.812500e-02	4.889269e+00	-20.990	-0.92750	0.000	9.975000
Infant mortality (per 1000 births)	224.0	3.550696e+01	3.538990e+01	2.290	8.15000	21.000	5.570500e
GDP ($ per capita)	226.0	9.689823e+03	1.004914e+04	500.000	1900.00000	5550.000	1.570000e
Literacy (%)	209.0	8.283828e+01	1.972217e+01	17.600	70.60000	92.500	9.800000e
Phones (per 1000)	223.0	2.360614e+02	2.279918e+02	0.200	37.80000	176.200	3.896500e
Arable (%)	225.0	1.379711e+01	1.304040e+01	0.000	3.22000	10.420	2.000000e
Crops (%)	225.0	4.564222e+00	8.361470e+00	0.000	0.19000	1.030	4.440000e
Other (%)	225.0	8.163831e+01	1.614083e+01	33.330	71.65000	85.700	9.544000e
Climate	205.0	2.139024e+00	6.993968e-01	1.000	2.00000	2.000	3.000000e
Birthrate	224.0	2.211473e+01	1.117672e+01	7.290	12.67250	18.790	2.982000e
Deathrate	223.0	9.241345e+00	4.990026e+00	2.290	5.91000	7.840	1.060500e
Agriculture	212.0	1.508443e-01	1.467980e-01	0.000	0.03775	0.099	2.210000
Industry	211.0	2.827109e-01	1.382722e-01	0.020	0.19300	0.272	3.410000
Service	212.0	5.652830e-01	1.658410e-01	0.062	0.42925	0.571	6.785000

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 [7]:

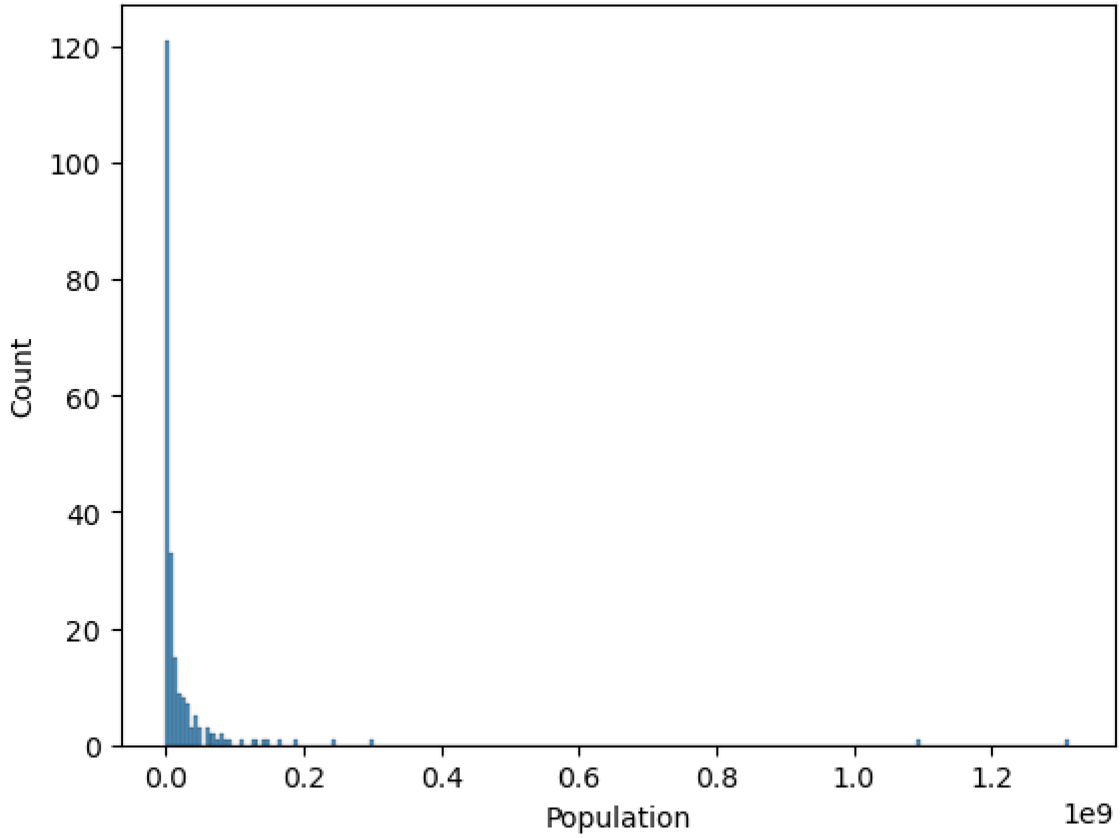
```
# CODE HERE
```

In [8]:

```
sns.histplot(data=df,x='Population')
```

Out[8]:

```
<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 [9]:

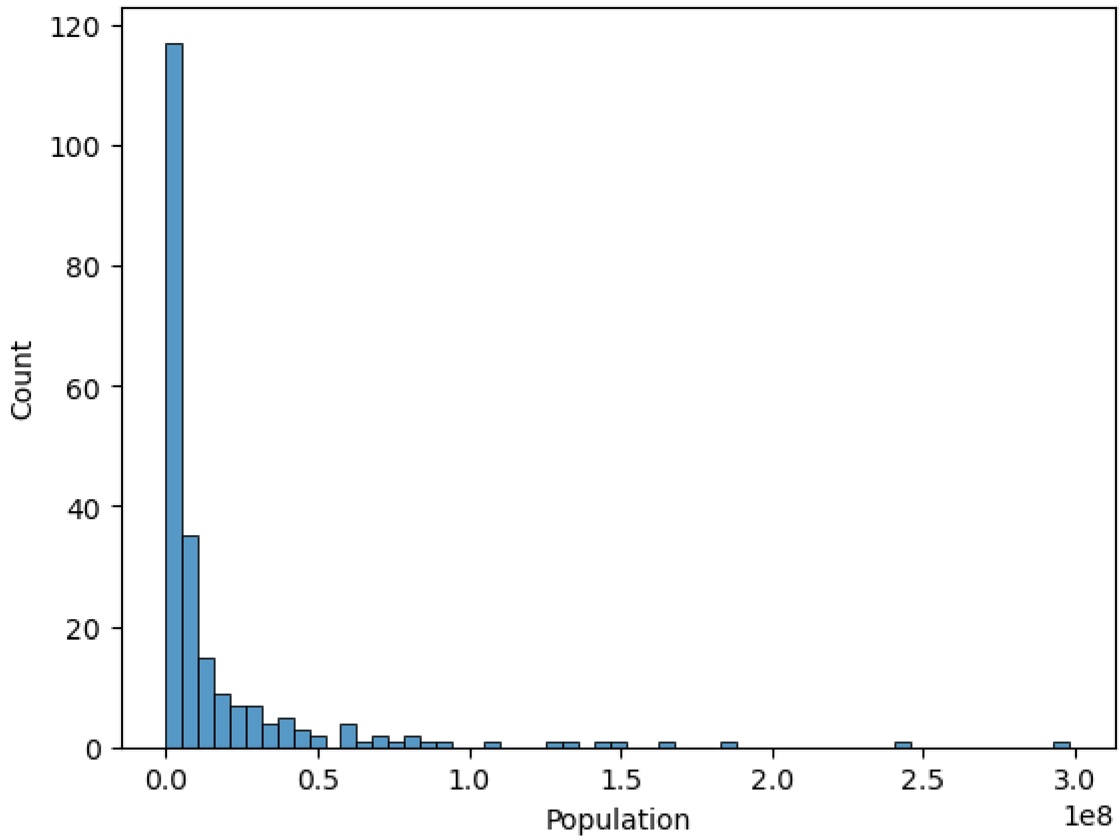
```
#CODE HERE
```

In [10]:

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

Out[10]:

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



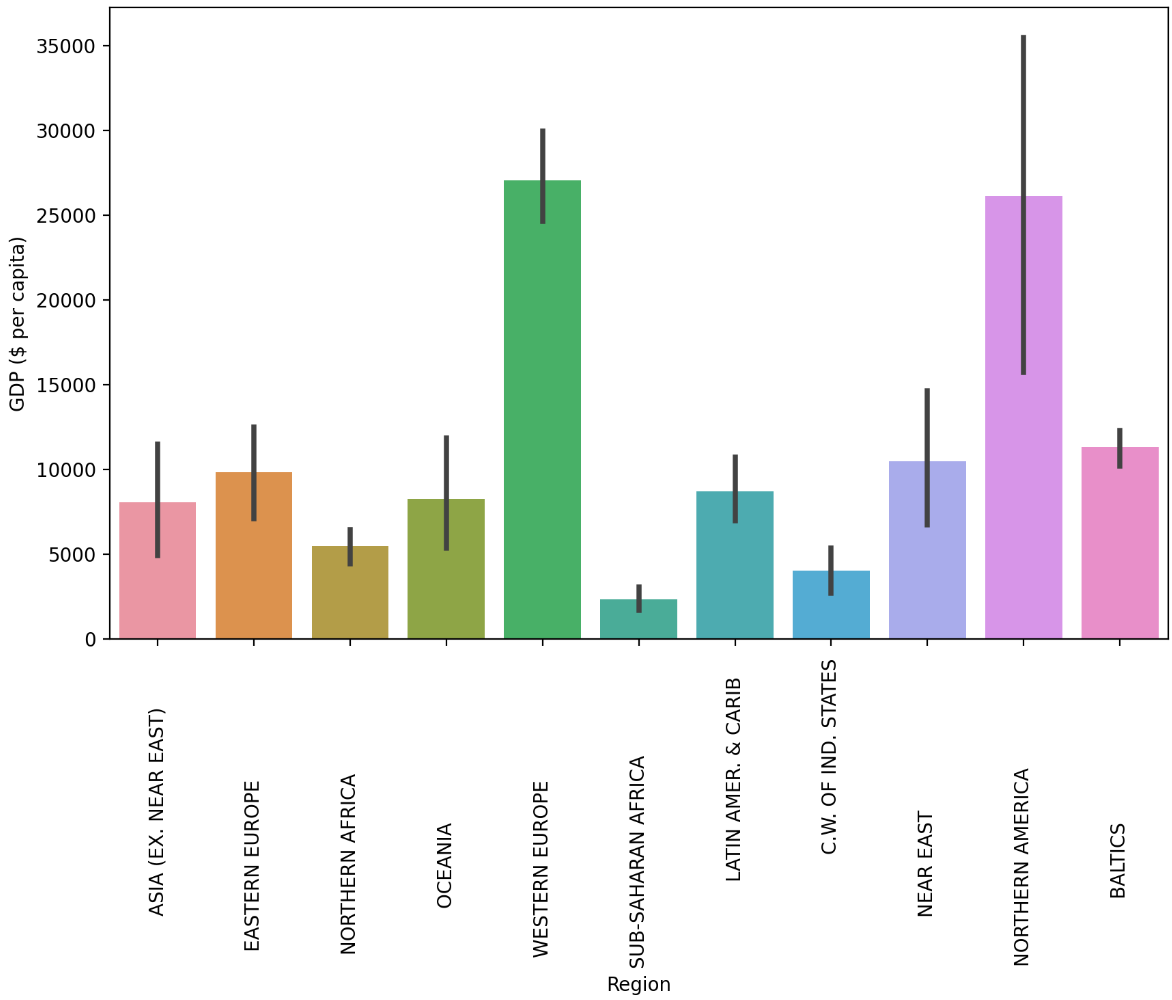
TASK: Now let's explore GDP and Regions. Create a bar chart showing the mean GDP per Capita per region (recall the black bar represents std).

In [11]:

```
# CODE HERE
```

In [12]:

```
plt.figure(figsize=(10,6),dpi=200)
sns.barplot(data=df,y='GDP ($ per capita)',x='Region',estimator=np.mean)
plt.xticks(rotation=90);
```



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

In [13]:

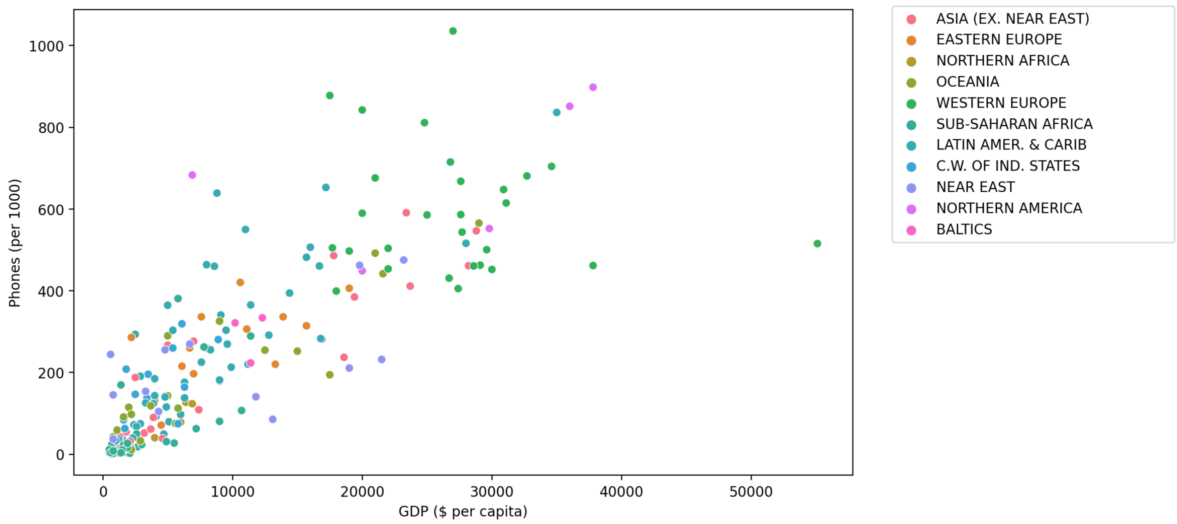
```
#CODE HERE
```

In [14]:

```
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df,x='GDP ($ per capita)',y='Phones (per 1000)',hue='Region')
plt.legend(loc=(1.05,0.5))
```

Out[14]:

```
<matplotlib.legend.Legend at 0x1fd0320b190>
```



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 [15]:

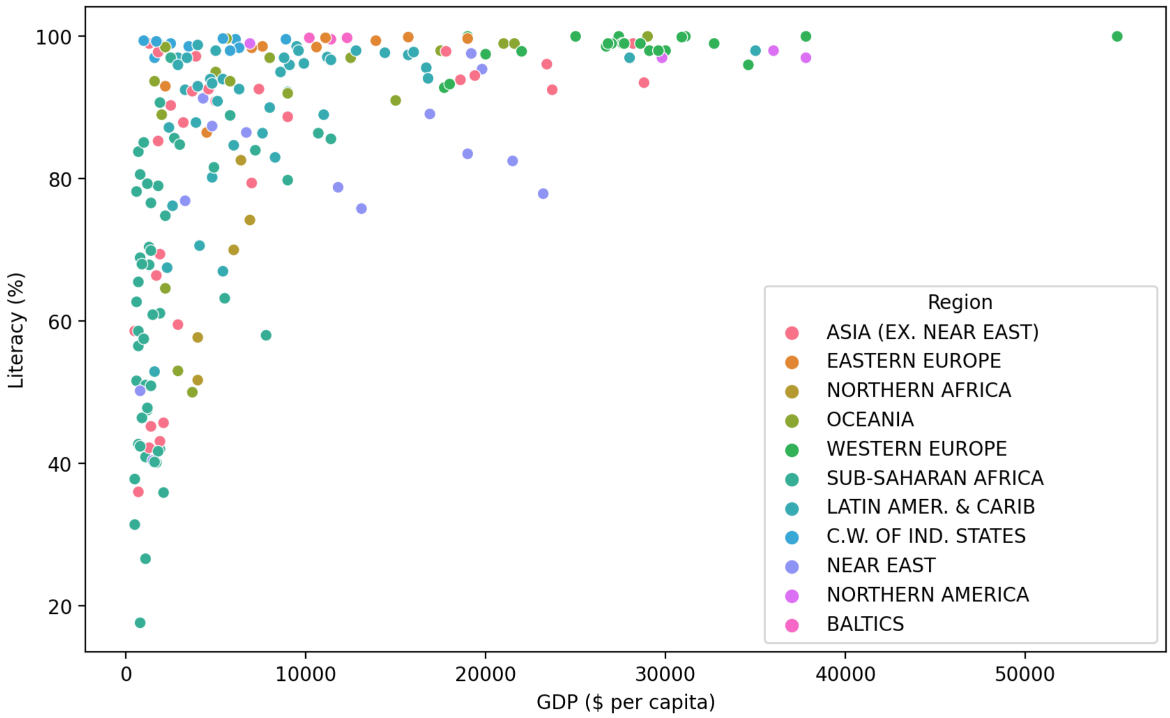
```
#CODE HERE
```

In [16]:

```
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df,x='GDP ($ per capita)',y='Literacy (%)',hue='Region')
```

Out[16]:

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



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

In [17]:

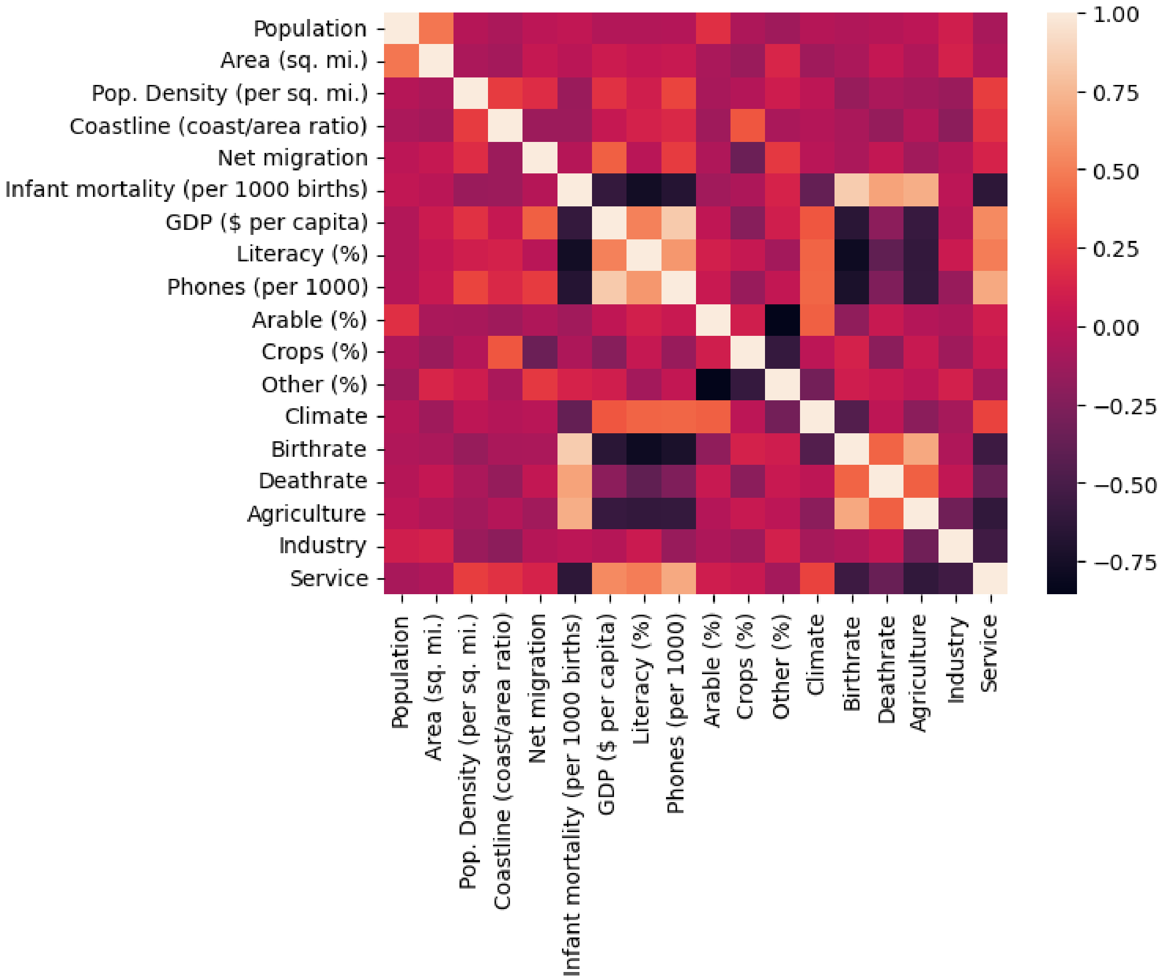
```
#CODE HERE
```

In [18]:

```
sns.heatmap(df.corr())
```

Out[18]:

```
<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 [19]:

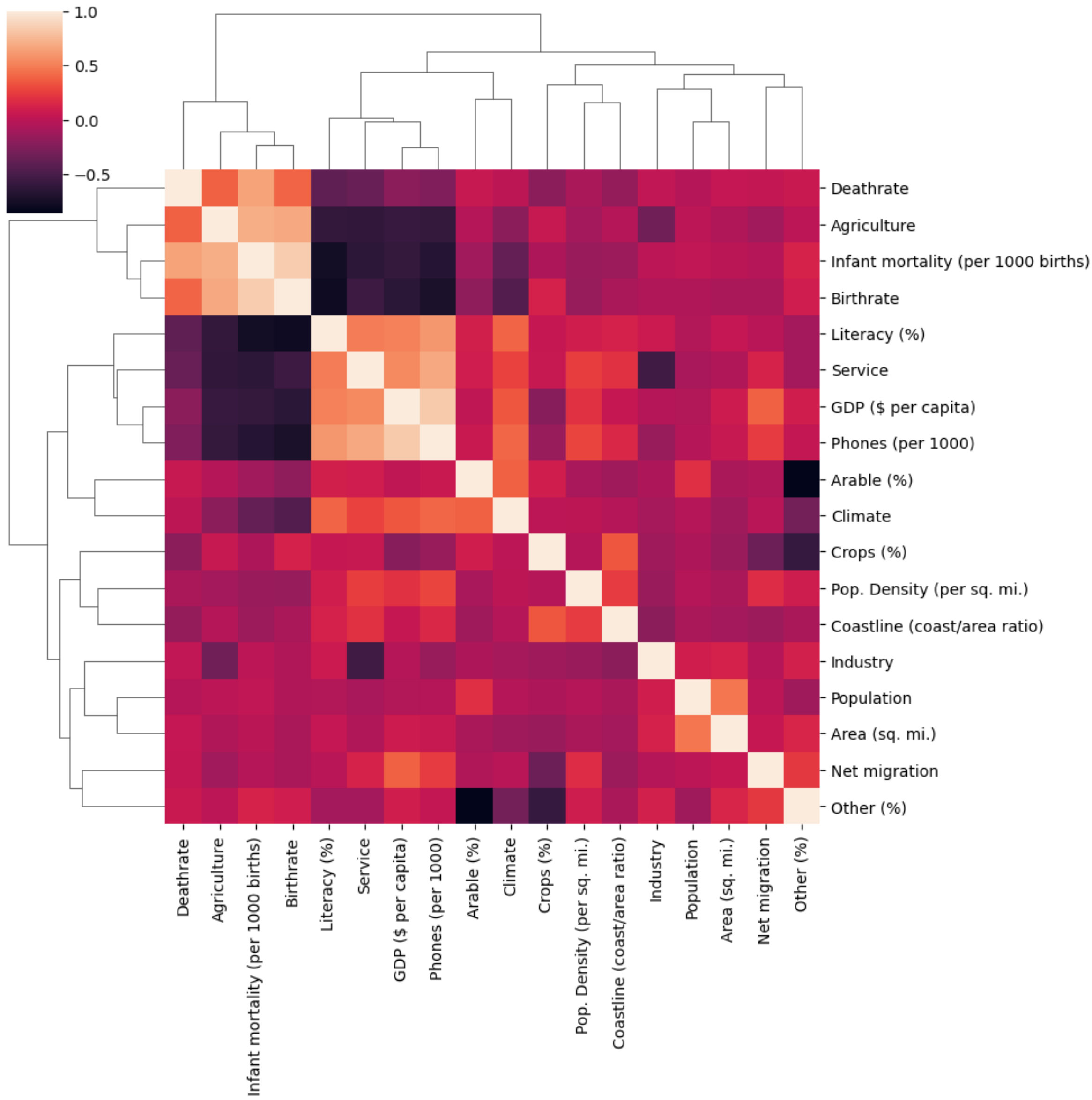
```
# CODE HERE
```

In [20]:

```
sns.clustermap(df.corr())
```

Out[20]:

```
<seaborn.matrix.ClusterGrid at 0x1fd030b8820>
```



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 [21]:

```
#CODE HERE
```

In [22]:

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

Out[22]:

```
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 [23]:

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

Out[23]:

```
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 [24]:

```
# REMOVAL OF TINY ISLANDS
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 [25]:

```
#CODE HERE
```

In [26]:

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

Out[26]:

```
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)](https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group)

In [27]:

```
# CODE HERE
```

In [28]:

```
# https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-eac
df['Climate'] = df['Climate'].fillna(df.groupby('Region')['Climate'].transform('mean'))
```

TASK: Check again on many elements are missing:

In [29]:

```
#CODE HERE
```

In [30]:

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

Out[30]:

```
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 [31]:

```
#CODE HERE
```

In [32]:

```
df[df['Literacy (%)'].isnull()]
```

Out[32]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDI cap
25	Bosnia & Herzegovina	EASTERN EUROPE	4498976	51129	88.0	0.04	0.31	21.05	610
66	Faroe Islands	WESTERN EUROPE	47246	1399	33.8	79.84	1.41	6.24	2200
74	Gaza Strip	NEAR EAST	1428757	360	3968.8	11.11	1.60	22.93	60
85	Guernsey	WESTERN EUROPE	65409	78	838.6	64.10	3.84	4.71	2000
99	Isle of Man	WESTERN EUROPE	75441	572	131.9	27.97	5.36	5.93	2100
104	Jersey	WESTERN EUROPE	91084	116	785.2	60.34	2.76	5.24	2480
108	Kiribati	OCEANIA	105432	811	130.0	140.94	0.00	48.52	80
123	Macedonia	EASTERN EUROPE	2050554	25333	80.9	0.00	-1.45	10.09	670
185	Slovakia	EASTERN EUROPE	5439448	48845	111.4	0.00	0.30	7.41	1330
187	Solomon Islands	OCEANIA	552438	28450	19.4	18.67	0.00	21.29	170
209	Tuvalu	OCEANIA	11810	26	454.2	92.31	0.00	20.03	110
220	Virgin Islands	LATIN AMER. & CARIB	108605	1910	56.9	9.84	-8.94	8.03	1720
222	West Bank	NEAR EAST	2460492	5860	419.9	0.00	2.98	19.62	80

In [33]:

```
# https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-eac
df['Literacy (%)'] = df['Literacy (%)'].fillna(df.groupby('Region')['Literacy (%)'].transfo
```

TASK: Check again on the remaining missing values:

In [34]:

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

Out[34]:

```
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 [35]:

```
# CODE HERE
```

In [36]:

```
df = df.dropna()
```

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 [37]:

```
#CODE HERE
```

In [38]:

```
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 [39]:

```
#COde here
```

In [40]:

```
X = pd.get_dummies(X)
```

In [41]:

```
X.head()
```

Out[41]:

	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)	A
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 [42]:

```
#CODE HERE
```

In [43]:

```
from sklearn.preprocessing import StandardScaler
```

In [44]:

```
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

In [45]:

```
scaled_X
```

Out[45]:

```
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 [46]:

```
#CODE HERE
```

In [47]:

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

In [48]:

```
ssd = []

for k in range(2,30):

    model = KMeans(n_clusters=k)


    model.fit(scaled_X)

    #Sum of squared distances of samples to their closest cluster center.
    ssd.append(model.inertia_)
```

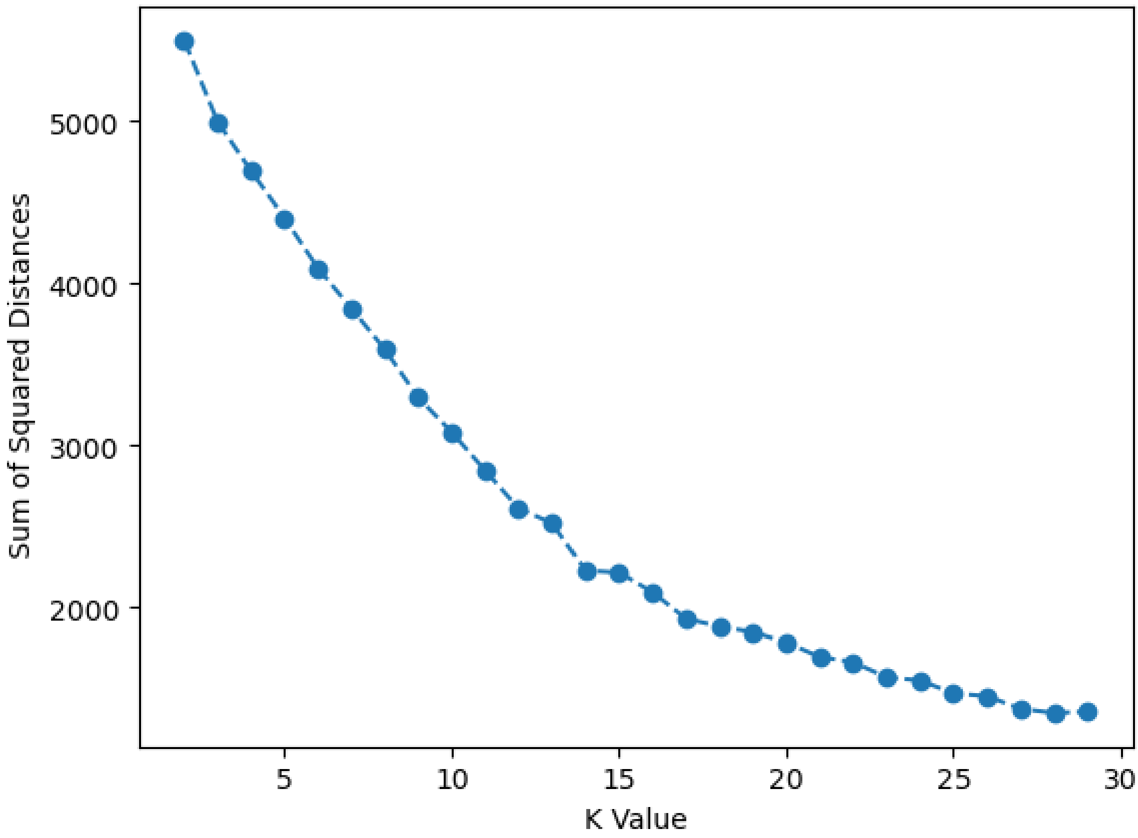
```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:13
34: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:13
34: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, 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\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:13
34: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, 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\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:13
34: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
```

In [50]:

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

Out[50]:

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

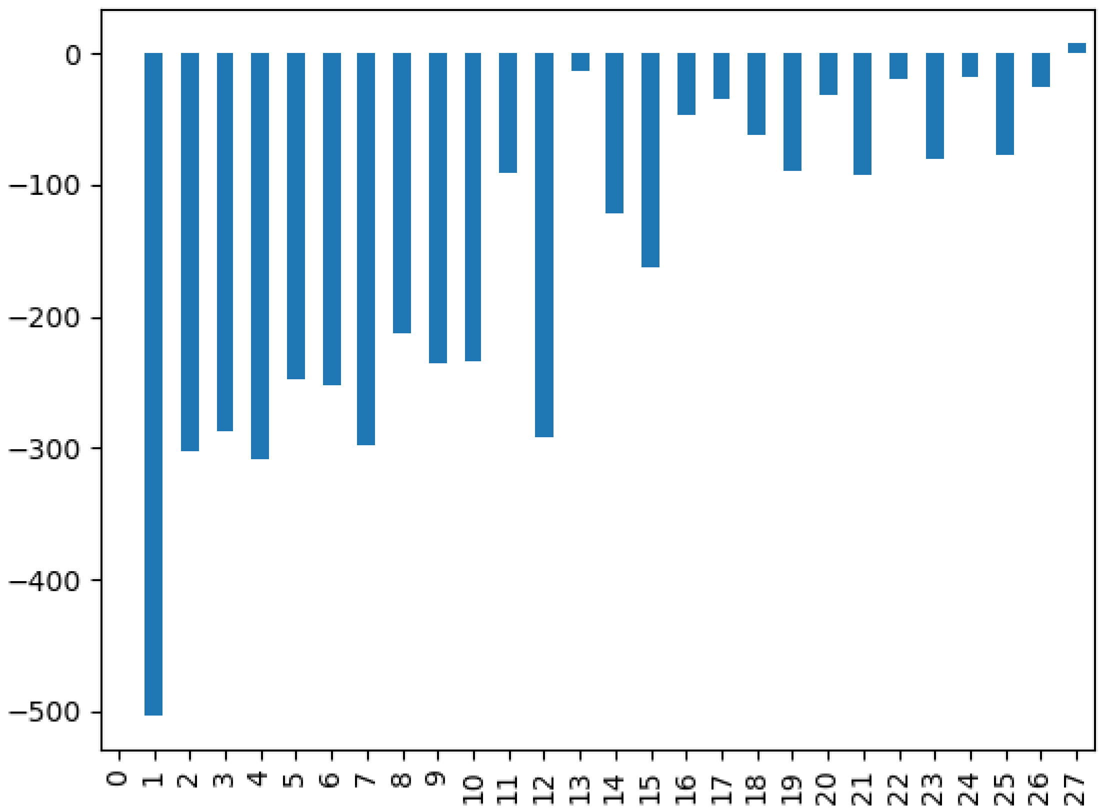


In [51]:

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

Out[51]:

```
<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 [52]:

```
# 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 [53]:

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

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
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 th
e environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[53]:

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

In [54]:

```
model.labels_
```

Out[54]:

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

In [55]:

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

In [56]:

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

Out[56]:

```
Deathrate                             -0.740749
Region_SUB-SAHARAN AFRICA             -0.734091
Infant mortality (per 1000 births)    -0.564161
Birthrate                             -0.452296
Agriculture                           -0.380670
Net migration                         -0.232335
Region_WESTERN EUROPE                 -0.134958
Climate                               -0.064159
Region_EASTERN EUROPE                 -0.058840
Other (%)                             -0.046626
Region_BALTICS                        -0.043357
Arable (%)                            -0.037717
Pop. Density (per sq. mi.)            -0.016705
GDP ($ per capita)                     0.008039
Region_NORTHERN AMERICA                0.019085
Area (sq. mi.)                         0.034405
Industry                               0.041552
Region_ASIA (EX. NEAR EAST)            0.084417
Population                             0.086608
Service                                0.096146
Phones (per 1000)                      0.137725
Region_NORTHERN AFRICA                 0.145002
Coastline (coast/area ratio)           0.145326
Region_C.W. OF IND. STATES             0.183274
Region_NEAR EAST                       0.211959
Region_OCEANIA                         0.234762
Crops (%)                              0.239658
Literacy (%)                           0.359238
Region_LATIN AMER. & CARIB             0.383277
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 [57]:

```
model = KMeans(n_clusters=15)

model.fit(scaled_X)
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
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 th
e environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[57]:

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

In [58]:

```
model = KMeans(n_clusters=3)

model.fit(scaled_X)
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
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 th
e environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[58]:

```
▾ 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 [59]:

```
iso_codes = pd.read_csv("country_iso_codes.csv")
```

In [60]:

```
iso_codes
```

Out[60]:

	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 [61]:

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

In [62]:

```
iso_mapping
```

Out[62]:

```
{'Afghanistan': 'AFG',
 'Akrotiri and Dhekelia – See United Kingdom, The': 'Akrotiri and Dhekeli
a – 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 Isl
ands – See Australia.',
 'Australia\u200a[b]': 'AUS',
```

In [63]:

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

In [64]:

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

In [65]:

```
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()
```