Clustering Problem

Find similar countries based on different attributes like Income per capita, phones per 1000, GDP, etc. and create clusters for the same. Plot the cluster on the world map using the ISO codes provided, depicting the clusters created.

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

```
df= pd.read_csv("/content/drive/MyDrive/MSBA/Semester 1/IDS-572-DataMiningProfNegar/project winter break/Country_Facts.csv")
df.head()
```

	Country	Region	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)	Arabl (%
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	700.0	36.0	3.2	12.1
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	86.5	71.2	21.0
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	70.0	78.1	3.2
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	8000.0	97.0	259.5	10.0
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	19000.0	100.0	497.2	2.2

```
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
```

```
df.columns
```

```
Index(['Country', 'Region', '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)', 'Arable (%)',
       'Crops (%)', 'Other (%)', 'Climate', 'Birthrate', 'Deathrate',
       'Agriculture', 'Industry', 'Service'],
      dtype='object')
```

Double-click (or enter) to edit

```
df.shape
```

```
(227, 20)
```

```
len(df['Country'].unique())
```

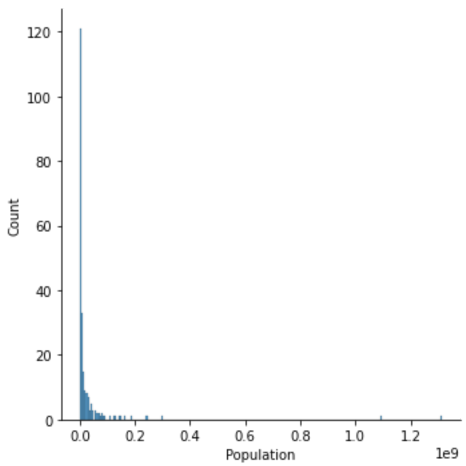
```
227
```

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

	count	mean	std	min	25%	50%	75%	max
Population	227.0	2.874028e+07	1.178913e+08	7026.000	437624.00000	4786994.000	1.749777e+07	1.313974e+09
Area (sq. mi.)	227.0	5.982270e+05	1.790282e+06	2.000	4647.50000	86600.000	4.418110e+05	1.707520e+07
Pop. Density (per sq. mi.)	227.0	3.790471e+02	1.660186e+03	0.000	29.15000	78.800	1.901500e+02	1.627150e+04
Coastline (coast/area ratio)	227.0	2.116533e+01	7.228686e+01	0.000	0.10000	0.730	1.034500e+01	8.706600e+02
Net migration	224.0	3.812500e-02	4.889269e+00	-20.990	-0.92750	0.000	9.975000e-01	2.306000e+01
Infant mortality (per 1000 births)	224.0	3.550696e+01	3.538990e+01	2.290	8.15000	21.000	5.570500e+01	1.911900e+02
GDP ($ per capita)	226.0	9.689823e+03	1.004914e+04	500.000	1900.00000	5550.000	1.570000e+04	5.510000e+04
Literacy (%)	209.0	8.283828e+01	1.972217e+01	17.600	70.60000	92.500	9.800000e+01	1.000000e+02
Phones (per 1000)	223.0	2.360614e+02	2.279918e+02	0.200	37.80000	176.200	3.896500e+02	1.035600e+03

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

```
<seaborn.axisgrid.FacetGrid at 0x7fc41d7d2d30>
```



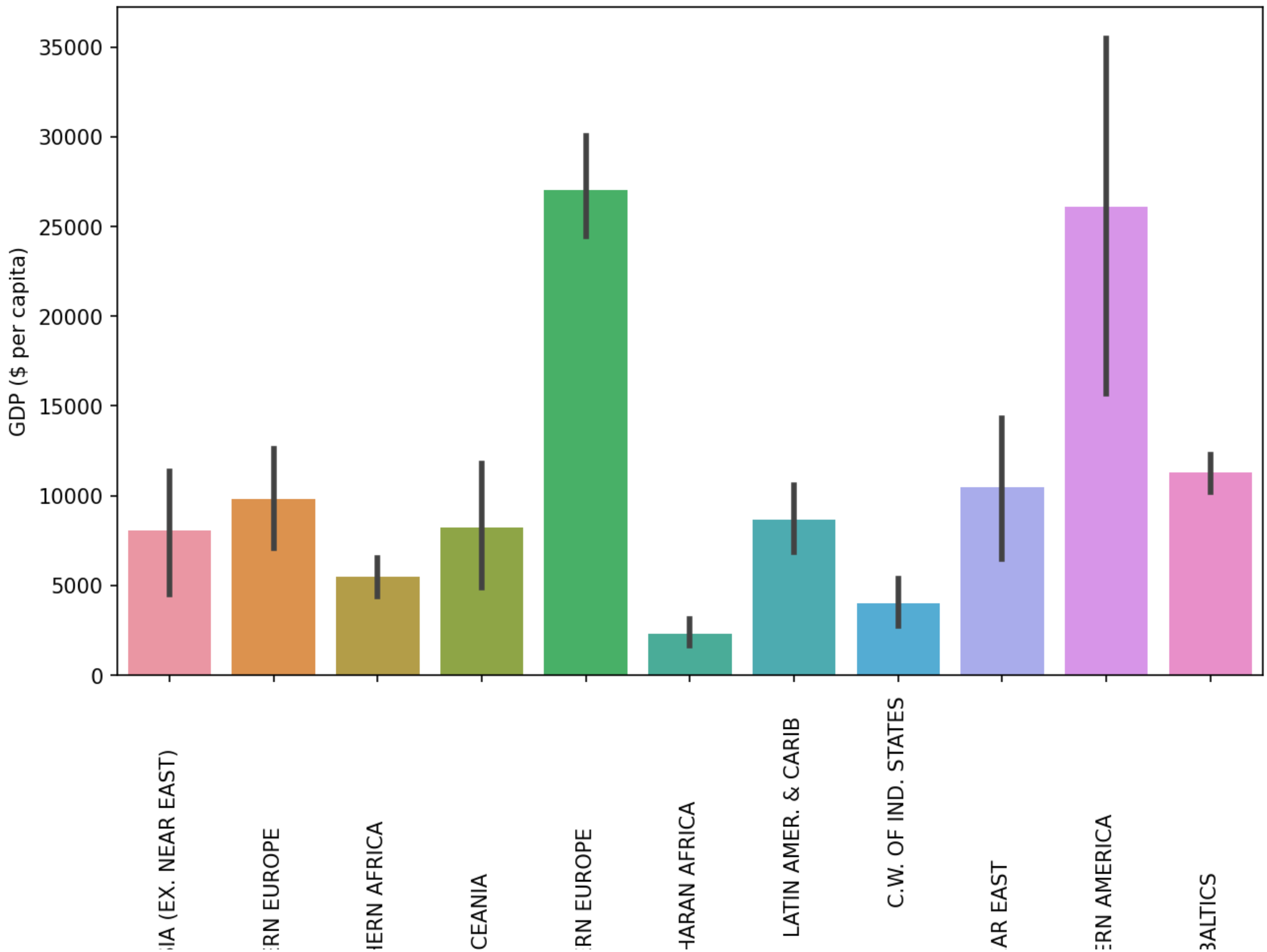
Population is data is very skewed and not normally distributed

```
df['Region'].value_counts()
```

```
SUB-SAHARAN AFRICA                     51
LATIN AMER. & CARIB                    45
ASIA (EX. NEAR EAST)                   28
WESTERN EUROPE                         28
OCEANIA                                21
NEAR EAST                              16
EASTERN EUROPE                         12
C.W. OF IND. STATES                    12
NORTHERN AFRICA                         6
NORTHERN AMERICA                        5
BALTICS                                 3
Name: Region, dtype: int64
```

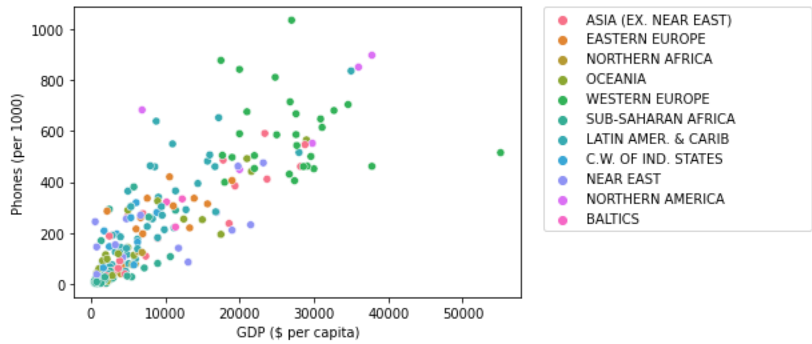
```
plt.figure(figsize=(10,6),dpi=150)
sns.barplot(data=df,y='GDP ($ per capita)',x='Region',estimator=np.mean)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10]),
 <a list of 11 Text major ticklabel objects>)
```



```
sns.scatterplot(data=df,x= 'GDP ($ per capita)',y= 'Phones (per 1000)',hue='Region')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
```

```
<matplotlib.legend.Legend at 0x7fc41a72c9a0>
```



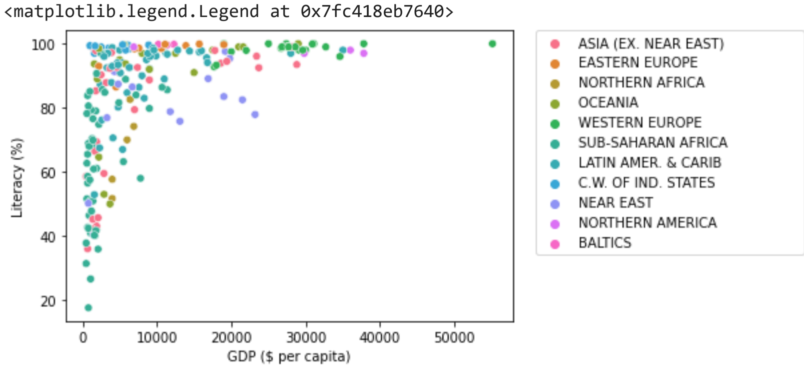

```
df[(df['GDP ($ per capita)'] > 30000) & (df['Phones (per 1000)'] < 600)]['Country']
```

```
121    Luxembourg
154        Norway
Name: Country, dtype: object
```

These 2 countries have high GDP but less Number of Phones. An interesting finding

```
sns.scatterplot(data=df,x='GDP ($ per capita)',y='Literacy (%)',hue='Region')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
```

```
<matplotlib.legend.Legend at 0x7fc418eb7640>
```




```
df = df.drop('Other (%)',axis=1)
```

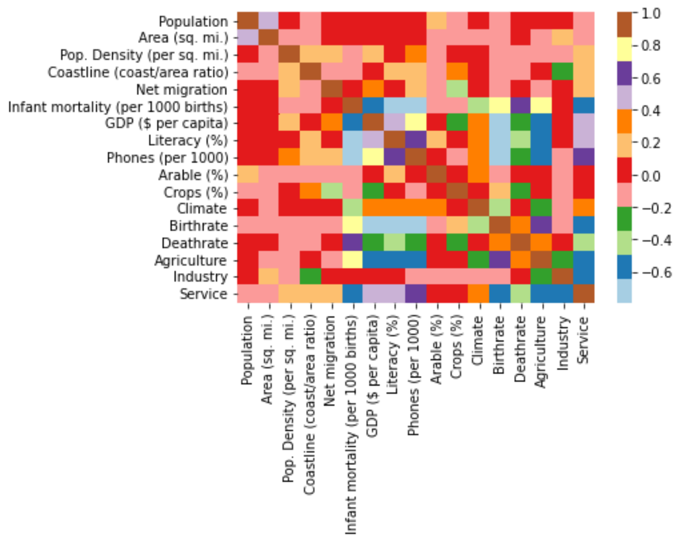
```
df.isnull().sum()
```

```
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
Climate                             22
Birthrate                            3
Deathrate                            4
Agriculture                         15
Industry                            16
Service                             15
dtype: int64
```

```
sns.heatmap(df.corr(), cmap='Paired')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc418e53bb0>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 19 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  Climate                            205 non-null    float64
 14  Birthrate                          224 non-null    float64
 15  Deathrate                          223 non-null    float64
 16  Agriculture                        212 non-null    float64
 17  Industry                           211 non-null    float64
 18  Service                            212 non-null    float64
dtypes: float64(15), int64(2), object(2)
memory usage: 33.8+ KB
```

```
df[df['Agriculture'].isnull()][['Country','GDP ($ per capita)']]
```

	Country	GDP ($ per capita)
3	American Samoa	8000.0
4	Andorra	19000.0
78	Gibraltar	17500.0
80	Greenland	20000.0
83	Guam	21000.0
134	Mayotte	2600.0
140	Montserrat	3400.0
144	Nauru	5000.0
153	N. Mariana Islands	12500.0
171	Saint Helena	2500.0
174	St Pierre & Miquelon	6900.0
177	San Marino	34600.0
208	Turks & Caicos Is	9600.0
221	Wallis and Futuna	3700.0
223	Western Sahara	NaN

```
df[df['Industry'].isnull()][['Country','GDP ($ per capita)']]
```

	Country	GDP ($ per capita)
3	American Samoa	8000.0
4	Andorra	19000.0
78	Gibraltar	17500.0
80	Greenland	20000.0
83	Guam	21000.0
134	Mayotte	2600.0
138	Monaco	27000.0
140	Montserrat	3400.0
153	N. Mariana Islands	12500.0

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

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

```
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
Climate                             18
Birthrate                            1
Deathrate                            2
Agriculture                          0
Industry                             1
Service                              1
dtype: int64
```

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

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

```
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
Climate                              0
```

```
Birthrate                             1
Deathrate                             2
Agriculture                           0
Industry                              1
Service                               1
dtype: int64
```

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

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

```
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
Climate                              0
Birthrate                            1
Deathrate                            2
Agriculture                          0
Industry                             1
Service                              1
dtype: int64
```

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

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

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

All missing values handled!

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221 entries, 0 to 226
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Country                             221 non-null    object 
 1   Region                              221 non-null    object 
 2   Population                          221 non-null    int64  
 3   Area (sq. mi.)                      221 non-null    int64  
 4   Pop. Density (per sq. mi.)          221 non-null    float64
 5   Coastline (coast/area ratio)        221 non-null    float64
 6   Net migration                       221 non-null    float64
 7   Infant mortality (per 1000 births)  221 non-null    float64
 8   GDP ($ per capita)                  221 non-null    float64
 9   Literacy (%)                        221 non-null    float64
 10  Phones (per 1000)                   221 non-null    float64
 11  Arable (%)                          221 non-null    float64
 12  Crops (%)                           221 non-null    float64
 13  Climate                             221 non-null    float64
 14  Birthrate                           221 non-null    float64
 15  Deathrate                           221 non-null    float64
 16  Agriculture                         221 non-null    float64
 17  Industry                            221 non-null    float64
 18  Service                             221 non-null    float64
dtypes: float64(15), int64(2), object(2)
memory usage: 34.5+ KB
```

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

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

```
X.head()
```

	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)	Arable (%)	...	Region_BALTICS	Region_C.W. OF IND. STATES	Region_EASTERN EUROPE	Region_LATIN AMER. & CARIB	Region_NEAR EAST	Regi
0	31056997	647500	48.0	0.00	23.06	163.07	700.0	36.0	3.2	12.13	...	0	0	0	0	0	
1	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	86.5	71.2	21.09	...	0	0	1	0	0	
2	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	70.0	78.1	3.22	...	0	0	0	0	0	
3	57794	199	290.4	58.29	-20.71	9.27	8000.0	97.0	259.5	10.00	...	0	0	0	0	0	
4	71201	468	152.1	0.00	6.60	4.05	19000.0	100.0	497.2	2.22	...	0	0	0	0	0	

5 rows × 28 columns

Scaling the features of X dataframe

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

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

```
scaled_X
```

```
array([[2.36307140e-02, 3.79200985e-02, 2.96607551e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.72048678e-03, 1.68320206e-03, 7.69943768e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.50562403e-02, 1.39484983e-01, 8.52746710e-04, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.63239770e-02, 3.09198848e-02, 2.50880554e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [8.74830703e-03, 4.40760465e-02, 9.45436569e-04, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [9.30752592e-03, 2.28737092e-02, 1.93412841e-03, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

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

```
inertias = []
```

```
for i in range(2,30):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(scaled_X)
    inertias.append(kmeans.inertia_)
```

```
plt.plot(range(2,30), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Sum of Squared Distances')
plt.show()
```

```
kmeans = KMeans(n_clusters=15)
kmeans.fit(scaled_X)
```

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

```
kmeans.labels_
```

```
array([13, 10,  9,  2,  4,  1,  0,  0, 12,  3,  0,  2,  4,  3,  0,  5, 13,
        0,  3,  4, 12, 11,  8, 13, 12, 10,  6, 12,  0, 13, 10,  1, 13, 11,
       13, 11,  8,  6,  0,  1,  1, 12, 13, 12, 11,  1,  6, 12, 11, 10, 12,
       10,  4,  6, 12, 12, 12,  9, 12,  6,  6, 14,  1,  4,  2,  4,  4,  0,
        2,  6, 11,  5,  3,  4, 11,  4,  4,  8, 12,  0,  2, 12,  1,  1, 12,
       12, 12,  7, 10,  4, 13, 13, 13,  5,  4,  4,  5,  4, 12,  7,  4,  5,
        3,  6,  2, 13,  7,  5,  3, 13, 14,  5,  6,  1,  9,  4, 14,  4,  7,
       10, 11, 11, 13, 13,  1,  4,  2,  0,  1, 11,  1, 12,  2,  3, 13, 12,
        9,  1,  6,  2, 13,  4,  0,  2,  2, 12,  1, 11,  2,  4,  5, 13,  2,
       12,  2, 12, 12, 13, 10,  4, 12,  5,  6, 10,  3, 11,  6,  0, 12,  8,
       12,  2,  4, 11,  5, 11,  6,  1,  7, 10, 10,  2,  1,  6,  4, 13, 11,
       12,  6,  4,  4,  5,  7,  3,  1, 13, 11,  2, 12,  9,  5,  3, 12,  2,
       11,  3,  5,  4,  8, 12,  3,  2, 12, 13,  0,  2,  5,  9,  5,  6,  6],
      dtype=int32)
```

```
iso_codes = pd.read_csv('/content/drive/MyDrive/MSBA/Semester 1/IDS-572-DataMiningProfNegar/project winter break/country_iso_codes.csv')
```

```
iso_codes.head()
```

	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

```
iso_mapping = iso_codes.set_index('Country')
iso_mapping.head()
```

Country	ISO Code
Afghanistan	AFG
Akrotiri and Dhekelia – See United Kingdom, The	Akrotiri and Dhekelia – See United Kingdom, The
Åland Islands	ALA
Albania	ALB
Algeria	DZA

```
iso_mapping_dict = iso_mapping['ISO Code'].to_dict()
```

```
iso_mapping_dict
```

```
 '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 States 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'}
```

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

```
df[['Country','ISO Code']].head()
```

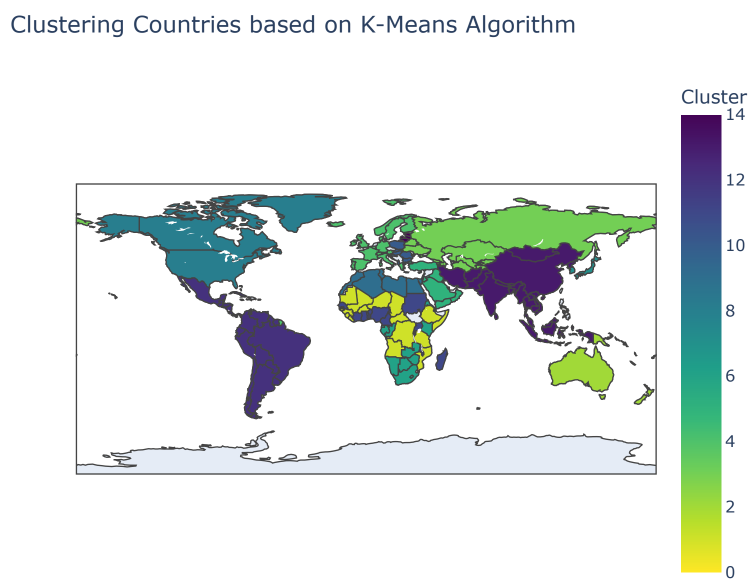
	Country	ISO Code
0	Afghanistan	AFG
1	Albania	ALB
2	Algeria	DZA
3	American Samoa	ASM
4	Andorra	AND

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

```
import plotly.express as px
```

```
fig = px.choropleth(df,locations='ISO Code',color='Cluster',hover_name='Country',
                    color_continuous_scale='Viridis_r')
```

```
fig.update_layout(title_text="Clustering Countries based on K-Means Algorithm")
fig.show()
```



1s completed at 9:28 PM