Clustering World Countries based on Covid-19 Prevelance & Vaccination Response

Punit Kumar

Abstract

Spread of Corona Virus in the last 2 years has affected all of our lives. With this project, we tried to find out how differently it has affected world countries and how differently countries have responded with vaccination campaigns. Two methods (explorative, cluster analysis) were used on WHO dataset to investigate and it was observed that the countries most affected by the virus had a stronger vaccine response and vice - versa.

Motivation

The Covid – 19 pandemic has changed our lives a lot in a short span of time. It has brought grief & distress to many and still seems far from over with new varients coming in from time to time. The only wheapon currently we have against it is the Covid Vaccine developed by various drug manufacturers. While the precautionary measures taken by various countries have helped in curtailing the exponential growth in spread of the virus, they have affected us both physically and mentally.

With this project I am trying to understand how different countries are responding to the situation with their Covid vaccination campaigns. This would help us to know which countries are doing better job so that other countries can learn from them.

Datasets

I have used data from 2 sources:

1. WHO:
    a) Overall Covid Cases & Deaths across countries
    b) Vaccination Coverage data at country level
2. World Bank
    a) Latest Population for world countries
    b) GDP Per Capita for world countries

Data Preparation and Cleaning

- Data was loaded from 4 different data files, cleaned separately and then merged together
- New fields were created based on calculation
- Null Value treatment was done based on data values.
  - Fill with Mode method was used to treat Null values for the field “No of different vaccines used”
  - Fill with Mean method was used for field “No. of days since First Vaccine”
  - Drop method was used for null vales in GDP per Capita
  - For Population field, it was calculated from other fields and inserted
- One dataset was not in standard csv format and had title and data information in first few lines. “skiprows” method was used while loading that file.
- In one dataset, country name was the key for joining and it had some country names different than the other dataset. Such country names were identified and fixed.
- Dummy columns were created for categorical variables

Research Questions

1. Which countries are lagging behind in the vaccination campaign
2. Clusters of countries based on prevalence of Covid-19 and counter response with vaccination

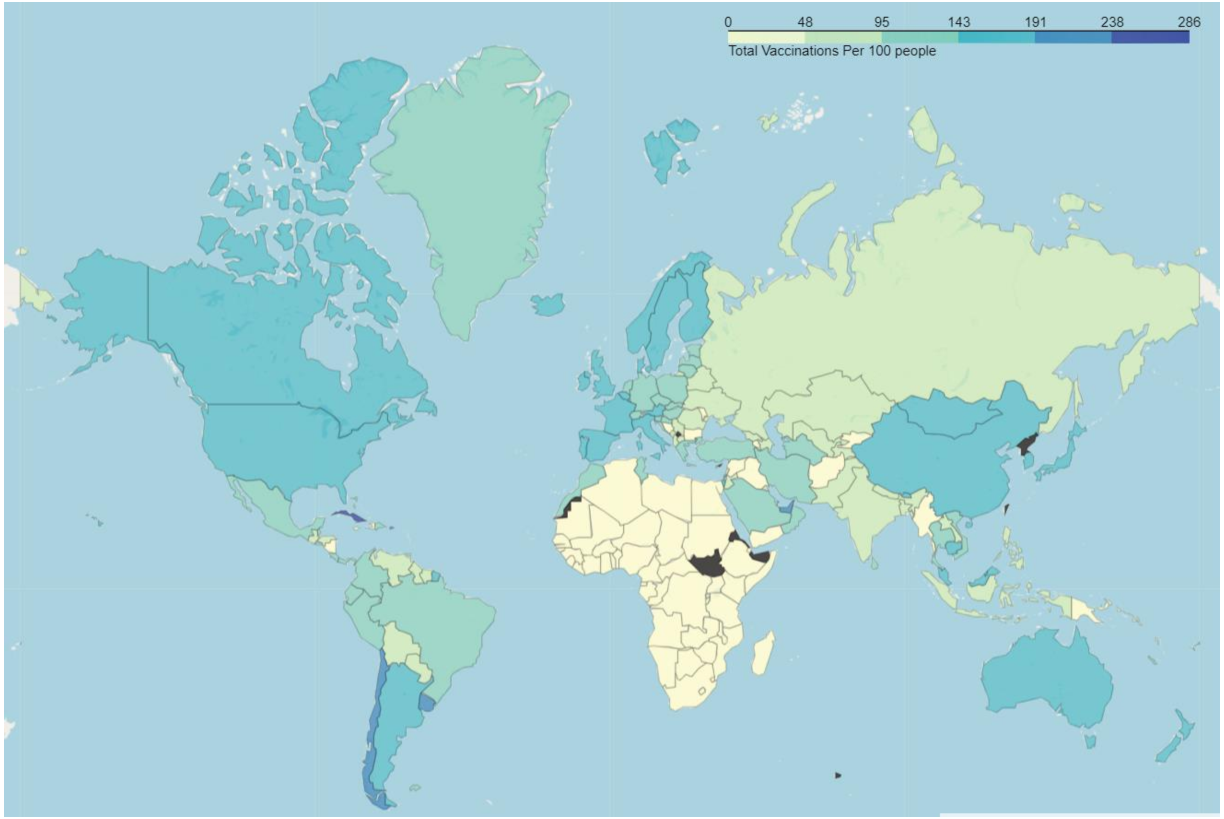
Methods

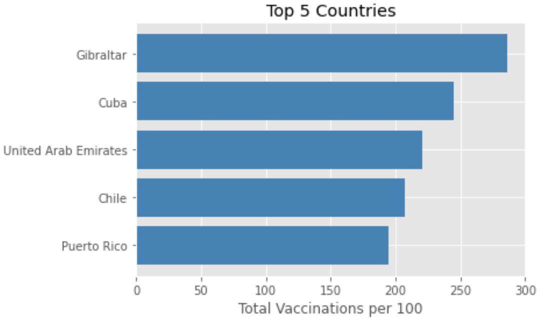
Two major methods were used:

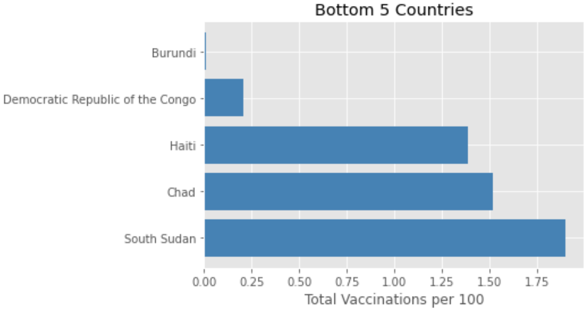
1. Exploratory
   a) World Map using follium for understanding data variations across countries
   b) Null value Treatment using Mean, Mode & drop
   c) Top & bottom values bar charts to get good and bad performing list
2. Cluster Analysis using K-Means Algorithm
   a) MaxMinScalar was used to scale both continuous and categorical variables
   b) Standard Scalar was used to scale continuous variables only
   c) Elbow method & Silhouette method were used to find optimum number of clusters in K-Means
   d) Parallel plot was used to visualize final clusters

Finding: Total Vaccination per 100 People shows African countries are lagging behind

Vaccination across world countries

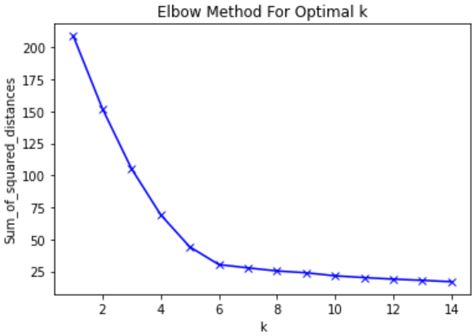


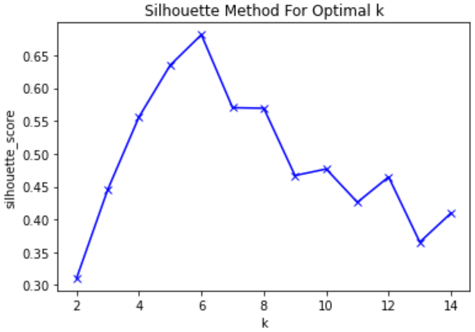


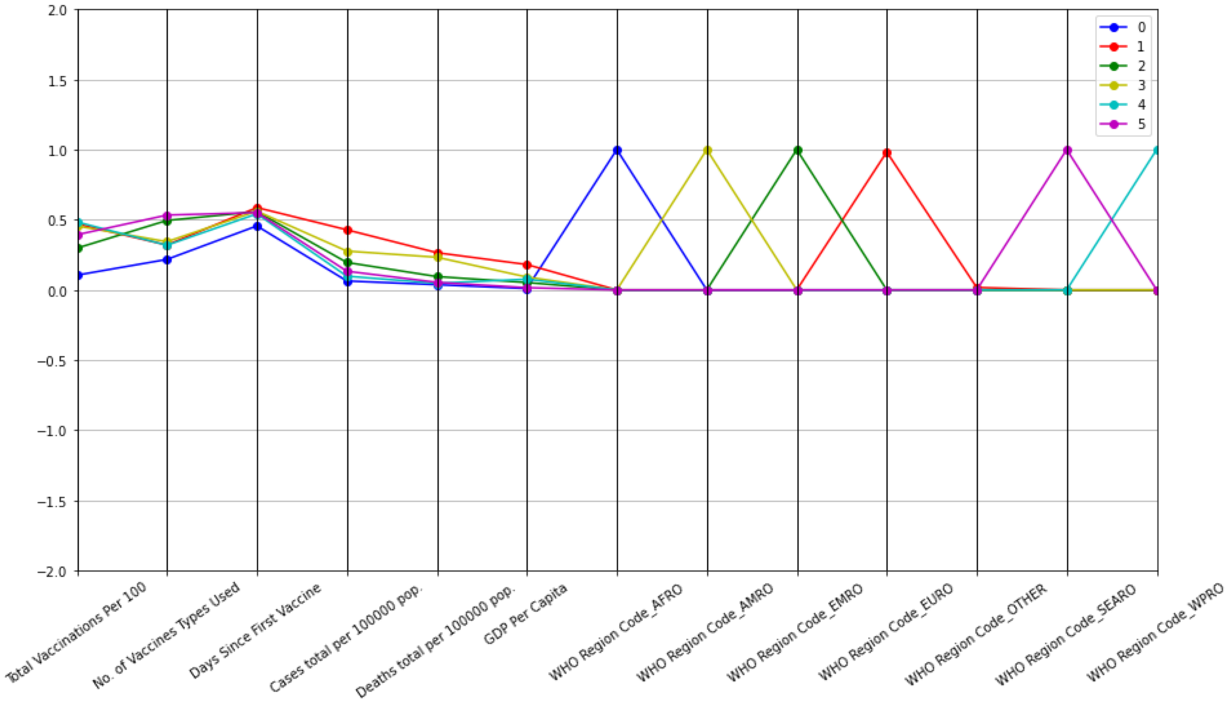


Finding: Categorical Features like WHO Region is overriding clustering

- All the features seen in the x-axis of the parallel plot were considered for clustering
- To arrive at the optimum value of k(no. of clusters), both elbow and Silhoutte methods were used
- Elbow curve shows a sharp turn while Silhoutte curve shows global maxima. In our case it is 6
- Parallel plot shows all other factors insignificant compared with the region related features as all clusters looks similar on all other features

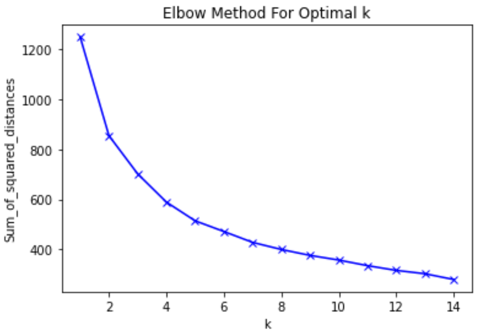


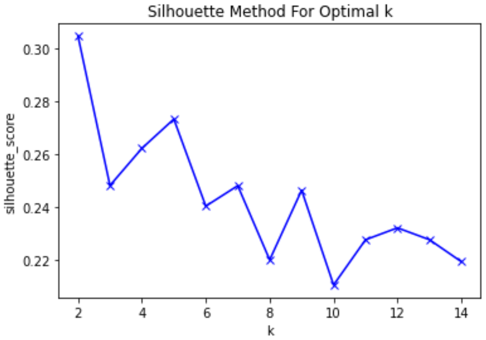


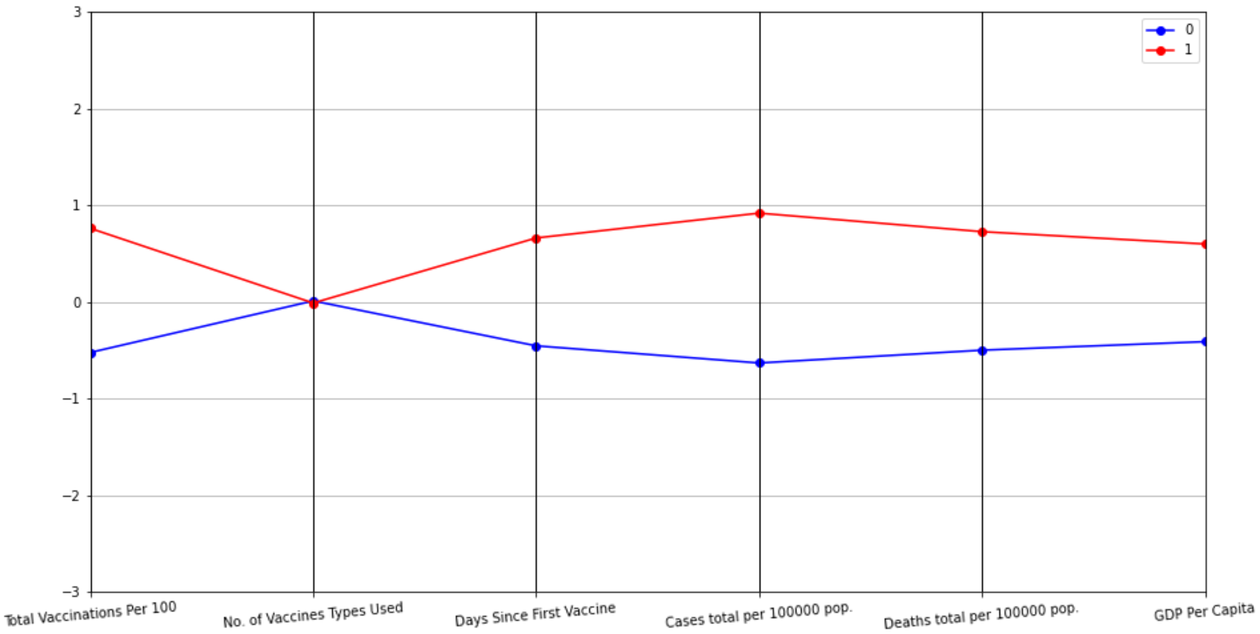


Finding: Countries are majorly divided into 2 clusters: more cases – more vaccinations & less cases – less vaccinations

- All the features seen in the x-axis of the parallel plot were considered for clustering
- To arrive at the optimum value of k(no. of clusters), both elbow and Silhoutte methods were used
- Elbow curve shows a sharp turn while Silhoutte curve shows global maxima. In our case it is 2
- Parallel plot shows 2 major clusters where **rich countries are more affected and more vaccinated while poor countries are less affected and also less vaccinated**







Limitations

Responses by countries against covid-19 can be even better captured by including data on various restrictive measures taken by countries.

Conclusions

1. Compared to other continents, African countries are lagging behind in the vaccination campaign.

2. With the cluster analysis done on the limited data on country response to covid 19, we understood that majorly **rich countries are more affected and more vaccinated while poor countries are less affected and also less vaccinated.** More cases could be due to more population density and travel requirements of the people in rich countries compared to poor countries. Also, these countries were able to afford the vaccines easily compared to the poor countries.

Acknowledgements

- Got data from WHO & World Bank websites
- Thanks to my wife for supporting and motivating me throughout

References

It’s my own work

In [1]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
from itertools import cycle, islice
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates
import datetime as dt
from sklearn.preprocessing import MinMaxScaler
import folium

%matplotlib inline
```

Importing First Dataset with Covid Cases and fatalities data

In [2]:

```
data1 = pd.read_csv('data/WHO-COVID-19-global-table-data.csv',index_col=False)
```

In [3]:

```
data1.head()
```

Out[3]:

	Name	WHO Region	Cases - cumulative total	Cases - cumulative total per 100000 population	Cases - newly reported in last 7 days	Cases - newly reported in last 7 days per 100000 population	Cases - newly reported in last 24 hours	Deaths - cumulative total	cu pc
0	Global	NaN	259502031	3329.279458	3980811	51.071786	611528	5183003	
1	United States of America	Americas	47802459	14441.715000	666617	201.393000	100455	771529	
2	India	South-East Asia	34555431	2504.009000	65808	4.769000	10549	467468	
3	Brazil	Americas	22043112	10370.330000	65451	30.792000	12930	613339	
4	The United Kingdom	Europe	10021501	14762.249000	299581	441.300000	46654	144433	

In [4]:

```
data1.shape
```

Out[4]:

```
(238, 12)
```

Slicing only required fields

In [5]:

```
data1_1 = data1.iloc[:,[0,1,2,3,7,8]]
```

In [6]:

```
data1_1.head()
```

Out[6]:

	Name	WHO Region	Cases - cumulative total	Cases - cumulative total per 100000 population	Deaths - cumulative total	Deaths - cumulative total per 100000 population
0	Global	NaN	259502031	3329.279458	5183003	66.4953
1	United States of America	Americas	47802459	14441.715000	771529	233.0880
2	India	South-East Asia	34555431	2504.009000	467468	33.8740
3	Brazil	Americas	22043112	10370.330000	613339	288.5490
4	The United Kingdom	Europe	10021501	14762.249000	144433	212.7580

Updating names to standard names for mapping

In [7]:

```
string1 = "Palest"
string2 = "Kos"
data1_1[(data1_1["Name"].str.contains(string1))
        | (data1_1["Name"].str.contains(string2))]
```

Out[7]:

	Name	WHO Region	Cases - cumulative total	Cases - cumulative total per 100000 population	Deaths - cumulative total	Deaths - cumulative total per 100000 population
68	occupied Palestinian territory, including east...	Eastern Mediterranean	459479	9006.895	4789	93.876
102	Kosovo[1]	Europe	161006	8966.367	2973	165.565

In [8]:

```
#dataframe.replace("old string", "new string")
data1_1["Name_New"] = np.where(data1_1["Name"].str.contains(string1),'occupied Palestinian
data1_1["Name_New"] = np.where(data1_1["Name"].str.contains(string2),'Kosovo', data1_1["Nam
```

```
<ipython-input-8-8e09a8812aad>:2: SettingWithCopyWarning:
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-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data1_1["Name_New"] = np.where(data1_1["Name"].str.contains(string1),'occu
pied Palestinian territory', data1_1["Name"])
<ipython-input-8-8e09a8812aad>:3: SettingWithCopyWarning:
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-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data1_1["Name_New"] = np.where(data1_1["Name"].str.contains(string2),'Koso
vo', data1_1["Name_New"])
```

In [9]:

```
data1_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238 entries, 0 to 237
Data columns (total 7 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Name                                          238 non-null    object
 1   WHO Region                                    237 non-null    object
 2   Cases - cumulative total                      238 non-null    int64
 3   Cases - cumulative total per 100000 population  237 non-null    float6
4
 4   Deaths - cumulative total                     238 non-null    int64
 5   Deaths - cumulative total per 100000 population  237 non-null    float6
4
 6   Name_New                                      238 non-null    object
dtypes: float64(2), int64(2), object(3)
memory usage: 13.1+ KB
```

Second Dataset Load - For Vaccination related data

In [10]:

```
data2 = pd.read_csv('data/vaccination-data.csv',index_col=False)
```

In [11]:

```
data2.head()
```

Out[11]:

	COUNTRY	ISO3	WHO_REGION	DATA_SOURCE	DATE_UPDATED	TOTAL_VACCINATIONS
0	Afghanistan	AFG	EMRO	REPORTING	2021-11-23	4331275
1	Albania	ALB	EURO	REPORTING	2021-11-21	2006988
2	Algeria	DZA	AFRO	REPORTING	2021-11-22	12032500
3	American Samoa	ASM	WPRO	REPORTING	2021-11-24	66691
4	Andorra	AND	EURO	REPORTING	2021-10-31	104534

In [12]:

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228 entries, 0 to 227
Data columns (total 14 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   COUNTRY                              228 non-null    object
 1   ISO3                                 228 non-null    object
 2   WHO_REGION                           228 non-null    object
 3   DATA_SOURCE                          228 non-null    object
 4   DATE_UPDATED                         228 non-null    object
 5   TOTAL_VACCINATIONS                   228 non-null    int64
 6   PERSONS_VACCINATED_1PLUS_DOSE        223 non-null    float64
 7   TOTAL_VACCINATIONS_PER100            228 non-null    float64
 8   PERSONS_VACCINATED_1PLUS_DOSE_PER100 223 non-null    float64
 9   PERSONS_FULLY_VACCINATED             224 non-null    float64
 10  PERSONS_FULLY_VACCINATED_PER100      224 non-null    float64
 11  VACCINES_USED                        225 non-null    object
 12  FIRST_VACCINE_DATE                   208 non-null    object
 13  NUMBER_VACCINES_TYPES_USED           225 non-null    float64
```

selecting required columns

In [13]:

```
data2_1 = data2.iloc[:,[0,1,2,5,7,12,13]]
```

In [14]:

```
data2_1.head()
```

Out[14]:

	COUNTRY	ISO3	WHO_REGION	TOTAL_VACCINATIONS	TOTAL_VACCINATIONS_PER100	FII
0	Afghanistan	AFG	EMRO	4331275	11.126	
1	Albania	ALB	EURO	2006988	69.700	
2	Algeria	DZA	AFRO	12032500	27.439	
3	American Samoa	ASM	WPRO	66691	120.824	
4	Andorra	AND	EURO	104534	135.300	

In [15]:

```
data2_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228 entries, 0 to 227
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   COUNTRY                     228 non-null    object
 1   ISO3                        228 non-null    object
 2   WHO_REGION                  228 non-null    object
 3   TOTAL_VACCINATIONS          228 non-null    int64
 4   TOTAL_VACCINATIONS_PER100   228 non-null    float64
 5   FIRST_VACCINE_DATE          208 non-null    object
 6   NUMBER_VACCINES_TYPES_USED  225 non-null    float64
dtypes: float64(2), int64(1), object(4)
memory usage: 12.6+ KB
```

changing to date format and calculating no. of days since first vaccine

In [16]:

```
data2_1["FIRST_VACCINE_DATE"] = pd.to_datetime(data2_1["FIRST_VACCINE_DATE"], format='%Y-%m
```

```
<ipython-input-16-e4e61d94b280>:1: SettingWithCopyWarning:
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-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data2_1["FIRST_VACCINE_DATE"] = pd.to_datetime(data2_1["FIRST_VACCINE_DAT
E"], format='%Y-%m-%d')
```

In [17]:

```
data2_1["Reference Date"] = pd.to_datetime('2021-11-25', format='%Y-%m-%d')
```

```
<ipython-input-17-2378ed4d2f10>:1: SettingWithCopyWarning:
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-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data2_1["Reference Date"] = pd.to_datetime('2021-11-25', format='%Y-%m-%
d')
```

In [18]:

```
data2_1["Days Since First Vaccine"] = (data2_1["Reference Date"] - data2_1["FIRST_VACCINE_D
```

```
<ipython-input-18-2897b985c892>:1: SettingWithCopyWarning:
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-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data2_1["Days Since First Vaccine"] = (data2_1["Reference Date"] - data2_1
["FIRST_VACCINE_DATE"]).dt.days
```

Merging 1st (Covid Cases & Fatalities) and 2nd Dataset (Vaccination)

In [19]:

```
mergedDf1 = data2_1.merge(data1_1, left_on='COUNTRY', right_on='Name_New', how= 'left')
```

In [20]:

```
mergedDf1.head()
```

Out[20]:

	COUNTRY	ISO3	WHO_REGION	TOTAL_VACCINATIONS	TOTAL_VACCINATIONS_PER100	FII
0	Afghanistan	AFG	EMRO	4331275	11.126	
1	Albania	ALB	EURO	2006988	69.700	
2	Algeria	DZA	AFRO	12032500	27.439	
3	American Samoa	ASM	WPRO	66691	120.824	
4	Andorra	AND	EURO	104534	135.300	

In [21]:

```
# Dropping unwanted columns
mergedDf1_1 = mergedDf1.drop(["Name","Name_New","Reference Date","FIRST_VACCINE_DATE"],axis
```

In [22]:

```
mergedDf1_1.head()
```

Out[22]:

	COUNTRY	ISO3	WHO_REGION	TOTAL_VACCINATIONS	TOTAL_VACCINATIONS_PER100	NU
0	Afghanistan	AFG	EMRO	4331275	11.126	
1	Albania	ALB	EURO	2006988	69.700	
2	Algeria	DZA	AFRO	12032500	27.439	
3	American Samoa	ASM	WPRO	66691	120.824	
4	Andorra	AND	EURO	104534	135.300	

In [23]:

```
# Renaming columns for better understanding
mergedDf1_1.rename(columns = {'COUNTRY':'Country Name'
                             ,'ISO3':'Country Code'
                             ,'WHO_REGION':'WHO Region Code'
                             ,'TOTAL_VACCINATIONS':'Total Vaccinations'
                             ,'TOTAL_VACCINATIONS_PER100':'Total Vaccinations Per 100'
                             ,'NUMBER_VACCINES_TYPES_USED':'No. of Vaccines Types Used'
                            }, inplace = True)
```

In [24]:

```
mergedDf1_1.head()
```

Out[24]:

	Country Name	Country Code	WHO Region Code	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	WHO Region
0	Afghanistan	AFG	EMRO	4331275	11.126	4.0	276.0	Eastern Mediterranean
1	Albania	ALB	EURO	2006988	69.700	5.0	316.0	Europe
2	Algeria	DZA	AFRO	12032500	27.439	4.0	299.0	Africa
3	American Samoa	ASM	WPRO	66691	120.824	3.0	339.0	Western Pacific
4	Andorra	AND	EURO	104534	135.300	3.0	309.0	Europe

In [25]:

```
mergedDf1_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 228 entries, 0 to 227
Data columns (total 12 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   Country Name                               228 non-null    object 
 1   Country Code                               228 non-null    object 
 2   WHO Region Code                            228 non-null    object 
 3   Total Vaccinations                         228 non-null    int64  
 4   Total Vaccinations Per 100                 228 non-null    float6
4
 5   No. of Vaccines Types Used                 225 non-null    float6
4
 6   Days Since First Vaccine                   208 non-null    float6
4
 7   WHO Region                                 227 non-null    object 
 8   Cases - cumulative total                   227 non-null    float6
4
 9   Cases - cumulative total per 100000 population  227 non-null    float6
4
 10  Deaths - cumulative total                  227 non-null    float6
4
 11  Deaths - cumulative total per 100000 population  227 non-null    float6
4
dtypes: float64(7), int64(1), object(4)
memory usage: 23.2+ KB
```

In [26]:

```
mergedDf1_1[mergedDf1_1["WHO Region"].isnull()==True]
```

Out[26]:

	Country Name	Country Code	WHO Region Code	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	WHO Region	Case cumulat to
26	Bonaire, Sint Eustatius and Saba	BES	AMRO	7391	28.441	2.0	276.0	NaN	N

In [27]:

```
#Removing data for BES country code as it is a combination of 3 countries
mergedDf1_1.drop(mergedDf1_1[mergedDf1_1["Country Code"]=="BES"].index, inplace = True)
```

Loading 3rd Dataset having latest country population information

In [28]:

```
data3 = pd.read_csv('data/API_SP.POP.TOTL_DS2_en_csv_v2_3158886.csv',index_col=False,skipro
```

In [29]:

```
data3.head()
```

Out[29]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54208.0	55434.0	56234.0	
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130836765.0	134159786.0	137614644.0	1412
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8996967.0	9169406.0	9351442.0	95
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	96396419.0	98407221.0	100506960.0	1026
4	Angola	AGO	Population, total	SP.POP.TOTL	5454938.0	5531451.0	5608499.0	56

5 rows × 66 columns

In [30]:

```
data3_1 = data3[["Country Name","Country Code","2020"]]
```

In [31]:

```
data3_1.head()
```

Out[31]:

	Country Name	Country Code	2020
0	Aruba	ABW	106766.0
1	Africa Eastern and Southern	AFE	677243299.0
2	Afghanistan	AFG	38928341.0
3	Africa Western and Central	AFW	458803476.0
4	Angola	AGO	32866268.0

In [32]:

```
data3_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Country Name  266 non-null    object 
 1   Country Code  266 non-null    object 
 2   2020          264 non-null    float64
dtypes: float64(1), object(2)
memory usage: 6.4+ KB
```

In [33]:

```
data3_1[data3_1["2020"].isnull()==True]
```

Out[33]:

	Country Name	Country Code	2020
69	Eritrea	ERI	NaN
110	Not classified	INX	NaN

In [34]:

```
del data3_1["Country Name"]
```

In [35]:

```
data3_1.columns = ["Country Code","Total Population(2020)"]
```

In [36]:

```
data3_1.head()
```

Out[36]:

	Country Code	Total Population(2020)
0	ABW	106766.0
1	AFE	677243299.0
2	AFG	38928341.0
3	AFW	458803476.0
4	AGO	32866268.0

Merging Cases & Vaccine data with population data

In [37]:

```
mergedDf1_2 = mergedDf1_1.merge(data3_1,on='Country Code', how= 'left')
```

In [38]:

```
mergedDf1_2.head()
```

Out[38]:

	Country Name	Country Code	WHO Region Code	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	WHO Region
0	Afghanistan	AFG	EMRO	4331275	11.126	4.0	276.0	Eastern Mediterranean
1	Albania	ALB	EURO	2006988	69.700	5.0	316.0	Europe
2	Algeria	DZA	AFRO	12032500	27.439	4.0	299.0	Africa
3	American Samoa	ASM	WPRO	66691	120.824	3.0	339.0	Western Pacific
4	Andorra	AND	EURO	104534	135.300	3.0	309.0	Europe

In [39]:

```
mergedDf1_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 227 entries, 0 to 226
Data columns (total 13 columns):
 #   Column                                         Non-Null Count  Dtype
---  ------                                         --------------  -----
 0   Country Name                                   227 non-null    object
 1   Country Code                                   227 non-null    object
 2   WHO Region Code                                227 non-null    object
 3   Total Vaccinations                             227 non-null    int64
 4   Total Vaccinations Per 100                     227 non-null    float6
4
 5   No. of Vaccines Types Used                     224 non-null    float6
4
 6   Days Since First Vaccine                       207 non-null    float6
4
 7   WHO Region                                     227 non-null    object
 8   Cases - cumulative total                       227 non-null    float6
4
 9   Cases - cumulative total per 100000 population  227 non-null    float6
4
 10  Deaths - cumulative total                      227 non-null    float6
4
 11  Deaths - cumulative total per 100000 population  227 non-null    float6
4
 12  Total Population(2020)                         210 non-null    float6
4
dtypes: float64(8), int64(1), object(4)
memory usage: 24.8+ KB
```

In [40]:

```
mergedDf1_2[mergedDf1_2["Total Population(2020)"].isnull()==True]
```

Out[40]:

	Country Name	Country Code	WHO Region Code	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	WHO Region	cu
6	Anguilla	AIA	AMRO	19066	127.090	2.0	293.0	Americas	
25	Bonaire	XAA	AMRO	31751	151.810	NaN	NaN	Americas	
46	Cook Islands	COK	WPRO	24346	138.613	1.0	192.0	Western Pacific	
66	Falkland Islands (Malvinas)	FLK	AMRO	4407	126.529	1.0	NaN	Americas	
71	French Guiana	GUF	AMRO	165026	55.251	2.0	316.0	Americas	
82	Guadeloupe	GLP	AMRO	266496	66.603	5.0	321.0	Americas	
85	Guernsey	GGY	EURO	104131	161.524	3.0	NaN	Europe	
103	Jersey	JEY	EURO	181102	168.004	3.0	NaN	Europe	
127	Martinique	MTQ	AMRO	267746	71.349	1.0	NaN	Americas	
135	Montserrat	MSR	AMRO	2949	58.992	1.0	290.0	Americas	
148	Niue	NIU	WPRO	2352	145.365	1.0	170.0	Western Pacific	
161	Pitcairn Islands	PCN	WPRO	74	148.000	1.0	192.0	Western Pacific	
171	Saba	XCA	AMRO	3131	161.976	NaN	NaN	Americas	
172	Saint Helena	SHN	AFRO	7892	129.995	1.0	NaN	Africa	
185	Sint Eustatius	XBA	AMRO	2963	94.393	NaN	NaN	Americas	
205	Tokelau	TKL	WPRO	1936	143.407	1.0	128.0	Western Pacific	
223	Wallis and Futuna	WLF	WPRO	11915	105.949	1.0	251.0	Western Pacific	

In [41]:

```
#Calculating population from covid data
mergedDf1_2["Total Population Calc"] = (mergedDf1_2["Cases - cumulative total"]
                                        /mergedDf1_2["Cases - cumulative total per 100000 population"]
```

In [42]:

```
#Inserting caculated population data for null values in Actual Population data
mergedDf1_2["Total Population(2020)_new"] = np.where((mergedDf1_2["Total Population(2020)"]
                                             & (mergedDf1_2["Total Population Calc"
                                            ,mergedDf1_2["Total Population Calc"],m
```

In [43]:

```
mergedDf1_2[mergedDf1_2["Total Population(2020)_new"].isnull()==True]
```

Out[43]:

	Country Name	Country Code	WHO Region Code	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	WHO Region	Cas cumula t
46	Cook Islands	COK	WPRO	24346	138.613	1.0	192.0	Western Pacific	
148	Niue	NIU	WPRO	2352	145.365	1.0	170.0	Western Pacific	
161	Pitcairn Islands	PCN	WPRO	74	148.000	1.0	192.0	Western Pacific	
172	Saint Helena	SHN	AFRO	7892	129.995	1.0	NaN	Africa	
205	Tokelau	TKL	WPRO	1936	143.407	1.0	128.0	Western Pacific	

inserting population values calculated from vaccine data into the null values of actual population data

In [44]:

```
mergedDf1_2["Total Population"] = np.where((mergedDf1_2["Total Population(2020)_new"].isnul
                                            ,100*(mergedDf1_2["Total Vaccinations"]
                                              ,mergedDf1_2["Total Population(2020)_
```

In [45]:

```
mergedDf1_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 227 entries, 0 to 226
Data columns (total 16 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Country Name                                227 non-null    object
 1   Country Code                                227 non-null    object
 2   WHO Region Code                             227 non-null    object
 3   Total Vaccinations                          227 non-null    int64
 4   Total Vaccinations Per 100                  227 non-null    float6
4
 5   No. of Vaccines Types Used                  224 non-null    float6
4
 6   Days Since First Vaccine                    207 non-null    float6
4
 7   WHO Region                                  227 non-null    object
 8   Cases - cumulative total                    227 non-null    float6
4
 9   Cases - cumulative total per 100000 population   227 non-null    float6
4
 10  Deaths - cumulative total                   227 non-null    float6
4
 11  Deaths - cumulative total per 100000 population  227 non-null    float6
4
 12  Total Population(2020)                      210 non-null    float6
4
 13  Total Population Calc                       217 non-null    float6
4
 14  Total Population(2020)_new                  222 non-null    float6
4
 15  Total Population                            227 non-null    float6
4
dtypes: float64(11), int64(1), object(4)
memory usage: 30.1+ KB
```

In [46]:

```
#dropping unnecessary columns
del mergedDf1_2["Total Population(2020)_new"]
del mergedDf1_2["Total Population(2020)"]
del mergedDf1_2["Total Population Calc"]
```

Loading GDP Per Capita (Current US $)

In [47]:

```
data4 = pd.read_csv('data/API_NY.GDP.PCAP.CD_DS2_en_csv_v2_3159040.csv',index_col=False,ski
data4.head()
```

Out[47]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	
0	Aruba	ABW	GDP per capita (current US$)	NY.GDP.PCAP.CD	NaN	NaN	NaN	
1	Africa Eastern and Southern	AFE	GDP per capita (current US$)	NY.GDP.PCAP.CD	147.836769	147.238537	156.426780	182.52
2	Afghanistan	AFG	GDP per capita (current US$)	NY.GDP.PCAP.CD	59.773234	59.860900	58.458009	78.70
3	Africa Western and Central	AFW	GDP per capita (current US$)	NY.GDP.PCAP.CD	107.963779	113.114697	118.865837	123.47
4	Angola	AGO	GDP per capita (current US$)	NY.GDP.PCAP.CD	NaN	NaN	NaN	

5 rows × 66 columns

In [48]:

```
#dropping unwanted columns
data4_1 = data4.drop(["Country Name","Indicator Name","Indicator Code","Unnamed: 65"],axis
```

In [49]:

```
data4_1.head()
```

Out[49]:

	Country Code	1960	1961	1962	1963	1964	1965	196
0	ABW	NaN	NaN	NaN	NaN	NaN	NaN	Nal
1	AFE	147.836769	147.238537	156.426780	182.521139	162.594548	180.489043	191.13579
2	AFG	59.773234	59.860900	58.458009	78.706429	82.095307	101.108325	137.59429
3	AFW	107.963779	113.114697	118.865837	123.478967	131.892939	138.566819	144.36839
4	AGO	NaN	NaN	NaN	NaN	NaN	NaN	Nal

5 rows × 62 columns

In [50]:

```
#Filling the latest non-null value from columns
data4_1['GDP Per Capita'] = data4_1.iloc[:, 1:].ffill(axis=1).iloc[:, -1]
```

In [51]:

```
data4_2= data4_1[["Country Code","GDP Per Capita"]]
data4_2.head()
```

Out[51]:

	Country Code	GDP Per Capita
0	ABW	30253.279358
1	AFE	1330.140232
2	AFG	508.808409
3	AFW	1714.426800
4	AGO	1895.770869

Merging Cases, vaccine, population and GDP per capita data

In [52]:

```
mergedDf1_3 = mergedDf1_2.merge(data4_2,on='Country Code', how= 'left')
```

In [53]:

```
mergedDf1_3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 227 entries, 0 to 226
Data columns (total 14 columns):
 #   Column                                           Non-Null Count  Dtyp
e
---  ------                                           --------------  ----
-
 0   Country Name                                     227 non-null    obje
ct
 1   Country Code                                     227 non-null    obje
ct
 2   WHO Region Code                                  227 non-null    obje
ct
 3   Total Vaccinations                               227 non-null    int6
4
 4   Total Vaccinations Per 100                       227 non-null    floa
t64
 5   No. of Vaccines Types Used                       224 non-null    floa
t64
```

In [54]:

```
mergedDf1_3.describe()
```

Out[54]:

	Total Vaccinations	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases - cumulative total	Cases - cumulative total per 100000 population	I cur
count	2.270000e+02	227.000000	224.000000	207.000000	2.270000e+02	227.000000	227
mean	3.393327e+07	93.116731	3.812500	281.545894	1.142735e+06	5780.326291	22829
std	1.840455e+08	58.317369	1.826714	50.977118	4.368314e+06	5713.329298	80382
min	7.400000e+01	0.012000	1.000000	38.000000	0.000000e+00	0.000000	0
25%	2.659260e+05	43.264000	2.000000	255.000000	1.282200e+04	457.888000	144
50%	1.730365e+06	96.657000	4.000000	281.000000	1.067940e+05	4683.340000	1732
75%	1.155442e+07	141.150000	5.000000	322.000000	5.633205e+05	9270.492500	10485
max	2.433028e+09	286.103000	10.000000	491.000000	4.780246e+07	24838.790000	771529

Null Value Treatment - Filling NAs with Mean, Mode or removal

In [55]:

```
#Filling NAs with mean value
mergedDf1_3['Days Since First Vaccine'] = mergedDf1_3['Days Since First Vaccine'].fillna(me
```

In [56]:

```
#Filling NAs with Mode (Most frequent value)
mergedDf1_3['No. of Vaccines Types Used'] = mergedDf1_3['No. of Vaccines Types Used'].filln
```

In [57]:

```
#Dropping NAs
mergedDf1_4 = mergedDf1_3.dropna().reset_index(drop=True)
```

In [58]:

```
mergedDf1_4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208 entries, 0 to 207
Data columns (total 14 columns):
 #   Column                                            Non-Null Count  Dtype
---  ------                                            --------------  -----
 0   Country Name                                      208 non-null    object
 1   Country Code                                      208 non-null    object
 2   WHO Region Code                                   208 non-null    object
 3   Total Vaccinations                                208 non-null    int64
 4   Total Vaccinations Per 100                        208 non-null    float6
4
 5   No. of Vaccines Types Used                        208 non-null    float6
4
 6   Days Since First Vaccine                          208 non-null    float6
4
 7   WHO Region                                        208 non-null    object
 8   Cases - cumulative total                          208 non-null    float6
4
 9   Cases - cumulative total per 100000 population    208 non-null    float6
4
 10  Deaths - cumulative total                         208 non-null    float6
4
 11  Deaths - cumulative total per 100000 population   208 non-null    float6
4
 12  Total Population                                  208 non-null    float6
4
 13  GDP Per Capita                                    208 non-null    float6
4
dtypes: float64(9), int64(1), object(4)
memory usage: 22.9+ KB
```

Checking statistics of dataset

In [59]:

```
mergedDf1_4.describe().transpose()
```

Out[59]:

	count	mean	std	min	25%	50%	
Total Vaccinations	208.0	3.702704e+07	1.920076e+08	1438.000000	4.379752e+05	2.613330e+06	1.
Total Vaccinations Per 100	208.0	8.981774e+01	5.782937e+01	0.012000	3.692750e+01	9.112750e+01	1.
No. of Vaccines Types Used	208.0	3.975962e+00	1.767774e+00	1.000000	3.000000e+00	4.000000e+00	5.
Days Since First Vaccine	208.0	2.833510e+02	4.808822e+01	38.000000	2.597500e+02	2.812729e+02	3.
Cases - cumulative total	208.0	1.246265e+06	4.550271e+06	0.000000	1.730775e+04	1.537575e+05	6.
Cases - cumulative total per 100000 population	208.0	5.729879e+03	5.634525e+03	0.000000	4.659310e+02	4.693081e+03	9.
Deaths - cumulative total	208.0	2.490438e+04	8.368180e+04	0.000000	2.360000e+02	2.206500e+03	1.
Deaths - cumulative total per 100000 population	208.0	9.063836e+01	1.003573e+02	0.000000	7.672500e+00	5.787600e+01	1.
Total Population	208.0	3.697797e+07	1.414036e+08	10834.000000	1.117124e+06	6.889756e+06	2.
GDP Per Capita	208.0	1.712117e+04	2.709607e+04	274.009523	2.233260e+03	6.269051e+03	2.

In [60]:

```
mergedDf1_4.columns = ['Country Name', 'Country Code', 'WHO Region Code', 'Total Vaccinatio
       'Total Vaccinations Per 100', 'No. of Vaccines Types Used',
       'Days Since First Vaccine', 'WHO Region', 'Cases total',
       'Cases total per 100000 pop.',
       'Deaths total',
       'Deaths total per 100000 pop.', 'Total Population',
       'GDP Per Capita']
```

In [61]:

```
continuous_features = ['Total Vaccinations Per 100',
                       'No. of Vaccines Types Used',
                       'Days Since First Vaccine',
                       'Cases total per 100000 pop.',
                       'Deaths total per 100000 pop.',
                       'GDP Per Capita']
```

In [62]:

```
categorical_features = ['WHO Region Code']
```

In [63]:

```
df_final = pd.concat([mergedDf1_4[continuous_features],mergedDf1_4[categorical_features]],a
```

In [64]:

```
df_final.head()
```

Out[64]:

	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases total per 100000 pop.	Deaths total per 100000 pop.	GDP Per Capita	WHO Region Code
0	11.126	4.0	276.0	403.675	18.770	508.808409	EMRO
1	69.700	5.0	316.0	6890.402	106.609	5215.276752	EURO
2	27.439	4.0	299.0	478.037	13.776	3310.386534	AFRO
3	120.824	3.0	339.0	9.058	0.000	11534.567544	WPRO
4	135.300	3.0	309.0	21440.497	169.546	40897.330873	EURO

To use categorical features, we need to convert them to binary

In [65]:

```
for col in categorical_features:
    dummies = pd.get_dummies(df_final[col],prefix=col)
    df_final = pd.concat([df_final,dummies], axis =1)
    df_final.drop(col, axis=1, inplace = True)
```

In [66]:

```
df_final.head()
```

Out[66]:

	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases total per 100000 pop.	Deaths total per 100000 pop.	GDP Per Capita	WHO Region Code_AFRO	WHO Regio Code_AMR(
0	11.126	4.0	276.0	403.675	18.770	508.808409	0	(
1	69.700	5.0	316.0	6890.402	106.609	5215.276752	0	(
2	27.439	4.0	299.0	478.037	13.776	3310.386534	1	(
3	120.824	3.0	339.0	9.058	0.000	11534.567544	0	(
4	135.300	3.0	309.0	21440.497	169.546	40897.330873	0	(

To give equal importance to all features, we need to scale the continuous features using scikit-learn's MinMaxScaler as the feature matrix is a mix of both binary and continuous variables

In [67]:

```
mms = MinMaxScaler()
mms.fit(df_final)
data_transformed = mms.transform(df_final)
```

In [68]:

```
features = df_final.columns
```

Calculate the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an elbow.

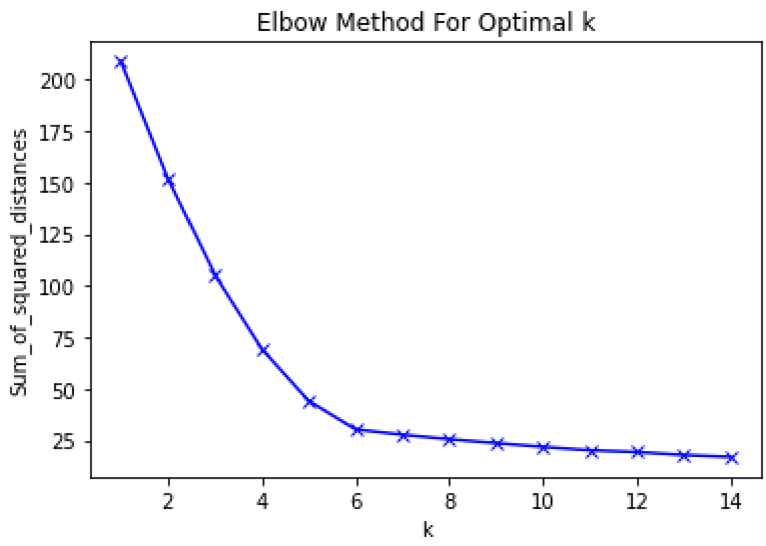
In [69]:

```
Sum_of_squared_distances = []
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data_transformed)
    Sum_of_squared_distances.append(km.inertia_)
```

```
C:\Users\punit\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: U
serWarning: 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(
```

In [70]:

```
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



From Elbow curve, k looks to be 6

Using silhouette method to confirm the optimum value of k

In [71]:

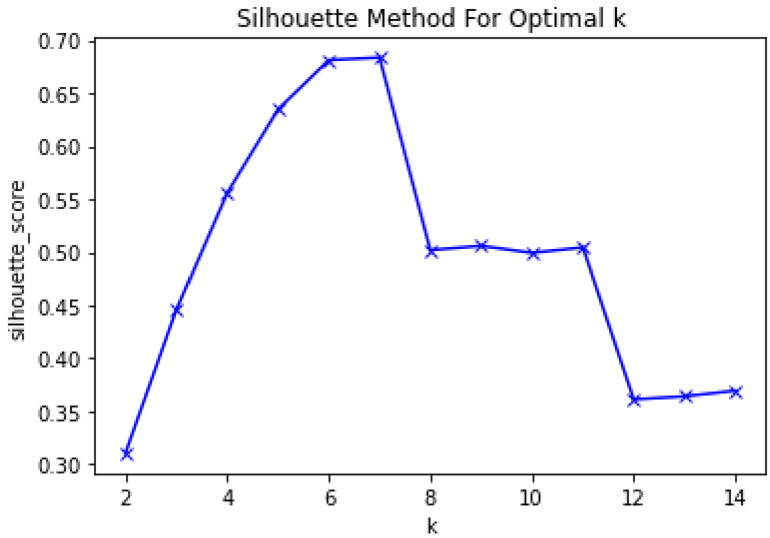
```
from sklearn.metrics import silhouette_score

sil = []
K = range(2,15)

# dissimilarity would not be defined for a single cluster, thus, minimum number of clusters
for k in K:
  kmeans = KMeans(n_clusters = k).fit(data_transformed)
  labels = kmeans.labels_
  sil.append(silhouette_score(data_transformed, labels, metric = 'euclidean'))
```

In [72]:

```
plt.plot(K,sil,'bx-')
plt.xlabel('k')
plt.ylabel('silhouette_score')
plt.title('Silhouette Method For Optimal k')
plt.show()
```



From Silhouette Method also, optimal k values looks like 6 as this looks like global maxima

In [73]:

```
kmeans = KMeans(n_clusters=6)
model = kmeans.fit(data_transformed)
print("model\n", model)
```

```
model
 KMeans(n_clusters=6)
```

In [74]:

```
centers = model.cluster_centers_
centers
```

Out[74]:

```
array([[ 4.55525678e-01,  3.43915344e-01,  5.61318074e-01,
         2.76838975e-01,  2.32821456e-01,  9.18062925e-02,
         5.55111512e-17,  1.00000000e+00, -4.16333634e-17,
        -5.55111512e-17, -2.60208521e-18, -2.77555756e-17,
         5.55111512e-17],
       [ 1.05206864e-01,  2.17391304e-01,  4.56089836e-01,
         6.37661846e-02,  3.72482075e-02,  1.00091595e-02,
         1.00000000e+00,  1.38777878e-16, -4.16333634e-17,
        -5.55111512e-17, -2.60208521e-18, -6.93889390e-18,
         2.77555756e-17],
       [ 4.84408196e-01,  3.18518519e-01,  5.42163355e-01,
         9.79380249e-02,  4.71587587e-02,  7.64327824e-02,
         8.32667268e-17,  5.55111512e-17, -4.16333634e-17,
        -5.55111512e-17,  8.67361738e-19, -2.77555756e-17,
         1.00000000e+00],
       [ 4.70501518e-01,  3.19923372e-01,  5.87020140e-01,
         4.28363719e-01,  2.65372033e-01,  1.80899300e-01,
        -1.11022302e-16,  1.66533454e-16, -5.55111512e-17,
         9.82758621e-01,  1.72413793e-02,  2.77555756e-17,
         0.00000000e+00],
       [ 2.97523631e-01,  4.94949495e-01,  5.57997190e-01,
         1.95697371e-01,  9.51339884e-02,  5.26778334e-02,
        -2.77555756e-17, -5.55111512e-17,  1.00000000e+00,
        -5.55111512e-17,  1.73472348e-18, -2.08166817e-17,
         2.77555756e-17],
       [ 3.93124962e-01,  5.33333333e-01,  5.51214128e-01,
         1.33294263e-01,  5.40695299e-02,  1.59719534e-02,
        -8.32667268e-17,  0.00000000e+00,  1.38777878e-17,
        -5.55111512e-17,  0.00000000e+00,  1.00000000e+00,
         2.77555756e-17]])
```

In [75]:

```
# Function that creates a DataFrame with a column for Cluster Number

def pd_centers(featuresUsed, centers):
    colNames = list(featuresUsed)
    colNames.append('prediction')

    # Zip with a column called 'prediction' (index)
    Z = [np.append(A, index) for index, A in enumerate(centers)]

    # Convert to pandas data frame for plotting
    P = pd.DataFrame(Z, columns=colNames)
    P['prediction'] = P['prediction'].astype(int)
    return P
```

In [76]:

```
# Function that creates Parallel Plots

def parallel_plot(data):
    my_colors = list(islice(cycle(['b', 'r', 'g', 'y', 'c','m','k']), None, len(data)))
    plt.figure(figsize=(15,8)).gca().axes.set_ylim([-2,+2])
    plt.xticks(rotation=35)
    parallel_coordinates(data,'prediction', color = my_colors, marker='o')
```

In [77]:

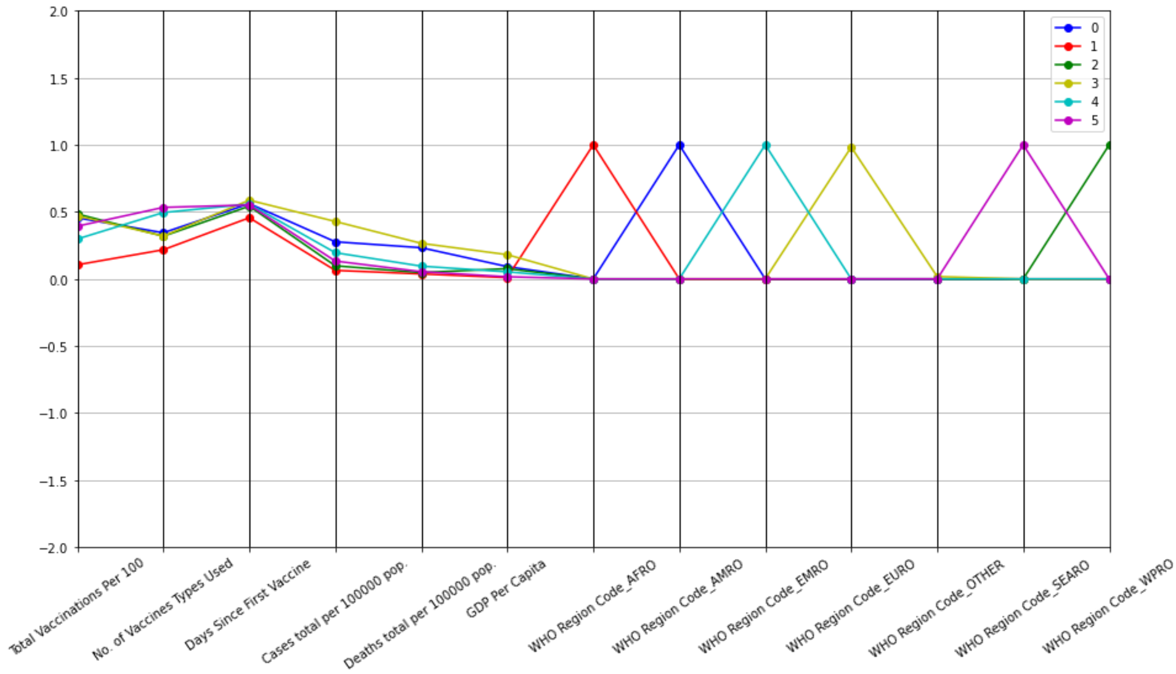
```
P = pd_centers(features,centers )
P
```

Out[77]:

	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases total per 100000 pop.	Deaths total per 100000 pop.	GDP Per Capita	WHO Region Code_AFRO	WHO Region Code_AMRO
0	0.455526	0.343915	0.561318	0.276839	0.232821	0.091806	5.551115e-17	1.000000e+00
1	0.105207	0.217391	0.456090	0.063766	0.037248	0.010009	1.000000e+00	1.387779e-16
2	0.484408	0.318519	0.542163	0.097938	0.047159	0.076433	8.326673e-17	5.551115e-17
3	0.470502	0.319923	0.587020	0.428364	0.265372	0.180899	-1.110223e-16	1.665335e-16
4	0.297524	0.494949	0.557997	0.195697	0.095134	0.052678	-2.775558e-17	-5.551115e-17
5	0.393125	0.533333	0.551214	0.133294	0.054070	0.015972	-8.326673e-17	0.000000e+00

In [78]:

```
parallel_plot(P)
```



From the graph above looks like the WHO Regions are coming out more significantly and creating seperate clusters. We need to again run the cluster analysis without the WHO Region feature

In [79]:

```
features = ['Total Vaccinations Per 100',
                       'No. of Vaccines Types Used',
                       'Days Since First Vaccine',
                       'Cases total per 100000 pop.',
                       'Deaths total per 100000 pop.',
                       'GDP Per Capita']
```

In [80]:

```
select_df = mergedDf1_4[features]
```

In [81]:

```
select_df.head()
```

Out[81]:

	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases total per 100000 pop.	Deaths total per 100000 pop.	GDP Per Capita
0	11.126	4.0	276.0	403.675	18.770	508.808409
1	69.700	5.0	316.0	6890.402	106.609	5215.276752
2	27.439	4.0	299.0	478.037	13.776	3310.386534
3	120.824	3.0	339.0	9.058	0.000	11534.567544
4	135.300	3.0	309.0	21440.497	169.546	40897.330873

Using Standard Scalar as there are only continuous features

In [82]:

```
X = StandardScaler().fit_transform(select_df)
X
```

Out[82]:

```
array([[-1.36404015,  0.01363096, -0.15323337, -0.94756064, -0.71785234,
        -0.61457026],
       [-0.34872022,  0.58067902,  0.68057802,  0.20646329,  0.15952173,
        -0.44045559],
       [-1.08127111,  0.01363096,  0.32620818, -0.93433124, -0.76773458,
        -0.51092655],
       ...,
       [-1.51276561, -0.5534171 , -1.3414146 , -1.0134255 , -0.84027059,
        -0.60290547],
       [-1.46398782, -0.5534171 , -1.21634289, -0.81604672, -0.70609595,
        -0.594515  ],
       [-0.80664903,  0.01363096, -0.06985223, -0.85925248, -0.5892113 ,
        -0.59165562]])
```

Calculate the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an elbow.

In [83]:

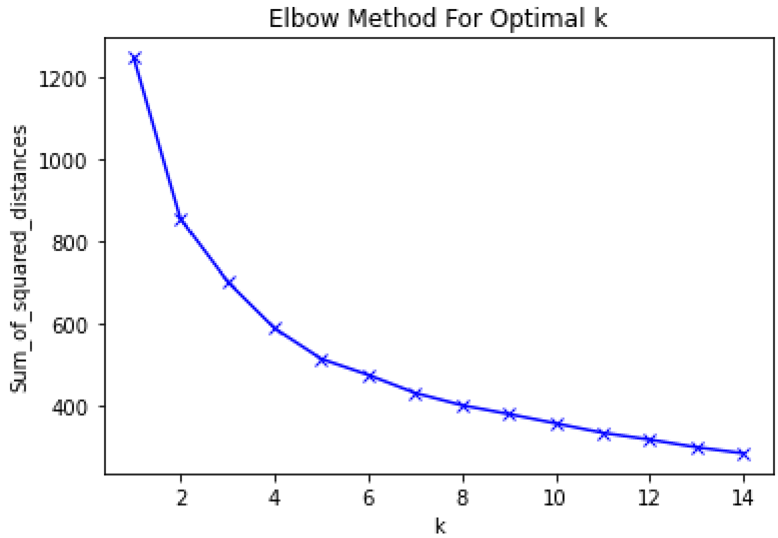
```
Sum_of_squared_distances = []
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(X)
    Sum_of_squared_distances.append(km.inertia_)
```

```
C:\Users\punit\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: U
serWarning: 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(
```

Plotting Elbow Curve for optimum value of k (Cluster Counts)

In [84]:

```
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



From above elbow curve looks like 2 is the optimum value of k

Using silhouette method to confirm the optimum value of k. The Silhouette Score reaches its global maximum at the optimal k

In [85]:

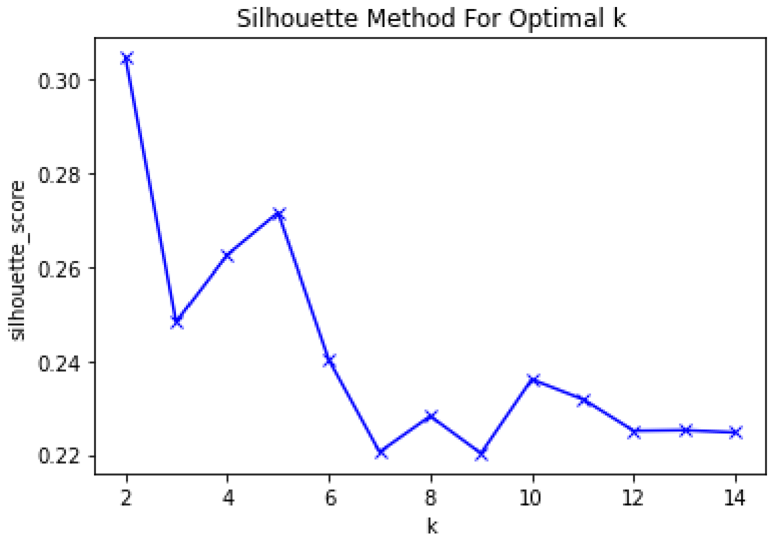
```
from sklearn.metrics import silhouette_score

sil = []
K = range(2,15)

# dissimilarity would not be defined for a single cluster, thus, minimum number of clusters
for k in K:
  kmeans = KMeans(n_clusters = k).fit(X)
  labels = kmeans.labels_
  sil.append(silhouette_score(X, labels, metric = 'euclidean'))
```

In [86]:

```
plt.plot(K,sil,'bx-')
plt.xlabel('k')
plt.ylabel('silhouette_score')
plt.title('Silhouette Method For Optimal k')
plt.show()
```



Global Maxima shows 2 as the optimum cluster numbers

Use k-Means Clustering

In [87]:

```
kmeans = KMeans(n_clusters=2)
model = kmeans.fit(X)
print("model\n", model)
```

```
model
 KMeans(n_clusters=2)
```

In [88]:

```
centers = model.cluster_centers_
centers
```

Out[88]:

```
array([[-0.52509914,  0.00902082, -0.45623708, -0.63365146, -0.50104753,
        -0.4130137 ],
       [ 0.75984934, -0.01305365,  0.66020189,  0.91693093,  0.72504525,
         0.59765511]])
```

In [89]:

```
# Function that creates a DataFrame with a column for Cluster Number

def pd_centers(featuresUsed, centers):
    colNames = list(featuresUsed)
    colNames.append('prediction')

    # Zip with a column called 'prediction' (index)
    Z = [np.append(A, index) for index, A in enumerate(centers)]

    # Convert to pandas data frame for plotting
    P = pd.DataFrame(Z, columns=colNames)
    P['prediction'] = P['prediction'].astype(int)
    return P
```

In [90]:

```
# Function that creates Parallel Plots

def parallel_plot(data):
    my_colors = list(islice(cycle(['b', 'r', 'g', 'y', 'k']), None, len(data)))
    plt.figure(figsize=(15,8)).gca().axes.set_ylim([-3,+3])
    plt.xticks(rotation=5)
    parallel_coordinates(data,'prediction', color = my_colors, marker='o')
```

In [91]:

```
features = select_df.columns
```

In [92]:

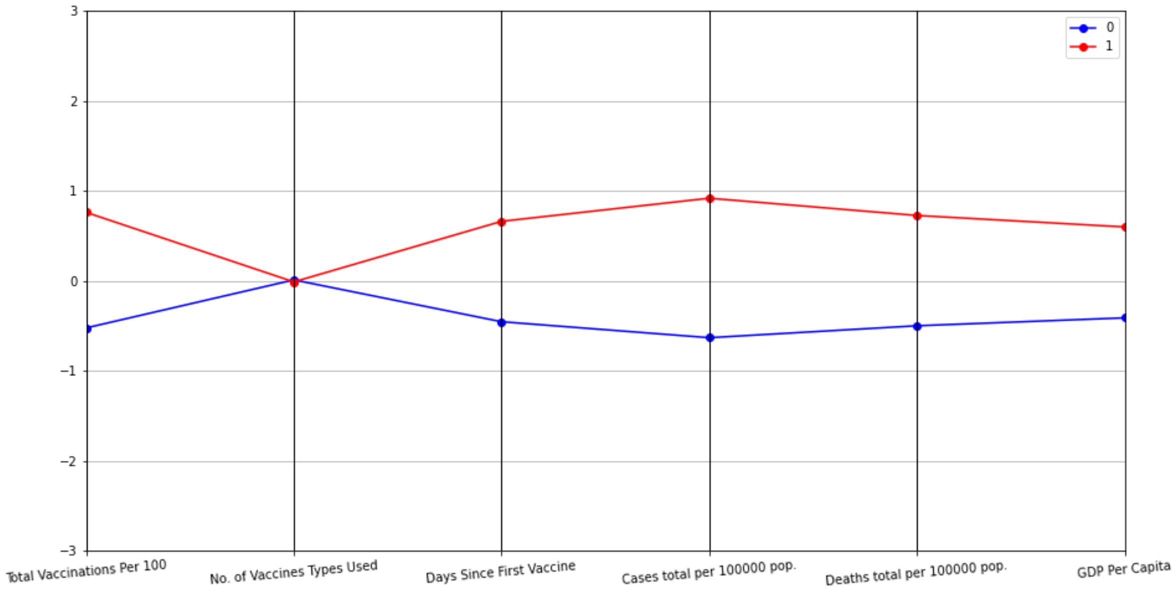
```
P = pd_centers(features,centers )
P
```

Out[92]:

	Total Vaccinations Per 100	No. of Vaccines Types Used	Days Since First Vaccine	Cases total per 100000 pop.	Deaths total per 100000 pop.	GDP Per Capita	prediction
0	-0.525099	0.009021	-0.456237	-0.633651	-0.501048	-0.413014	0
1	0.759849	-0.013054	0.660202	0.916931	0.725045	0.597655	1

In [93]:

```
parallel_plot(P)
```



In [94]:

```
country_geo = 'world-countries.json'
```

In [95]:

```
plot_data1 = data2.iloc[:,[1,7]]
```

In [96]:

```
data_indicator = 'Total Vaccinations Per 100 people'
```

In [97]:

```
plot_data1.columns = ['CountryCode', 'Value']
```

In [98]:

```
map = folium.Map(location=[50, 50], zoom_start=1.49)
```

In [99]:

```
# choropleth maps bind Pandas Data Frames and json geometries.  This allows us to quickly v
folium.Choropleth(geo_data=country_geo, data=plot_data1,
             columns=['CountryCode', 'Value'],
             key_on='feature.id',
             fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2,
             legend_name=data_indicator).add_to(map)
```

Out[99]:

```
<folium.features.Choropleth at 0x2bba5722070>
```

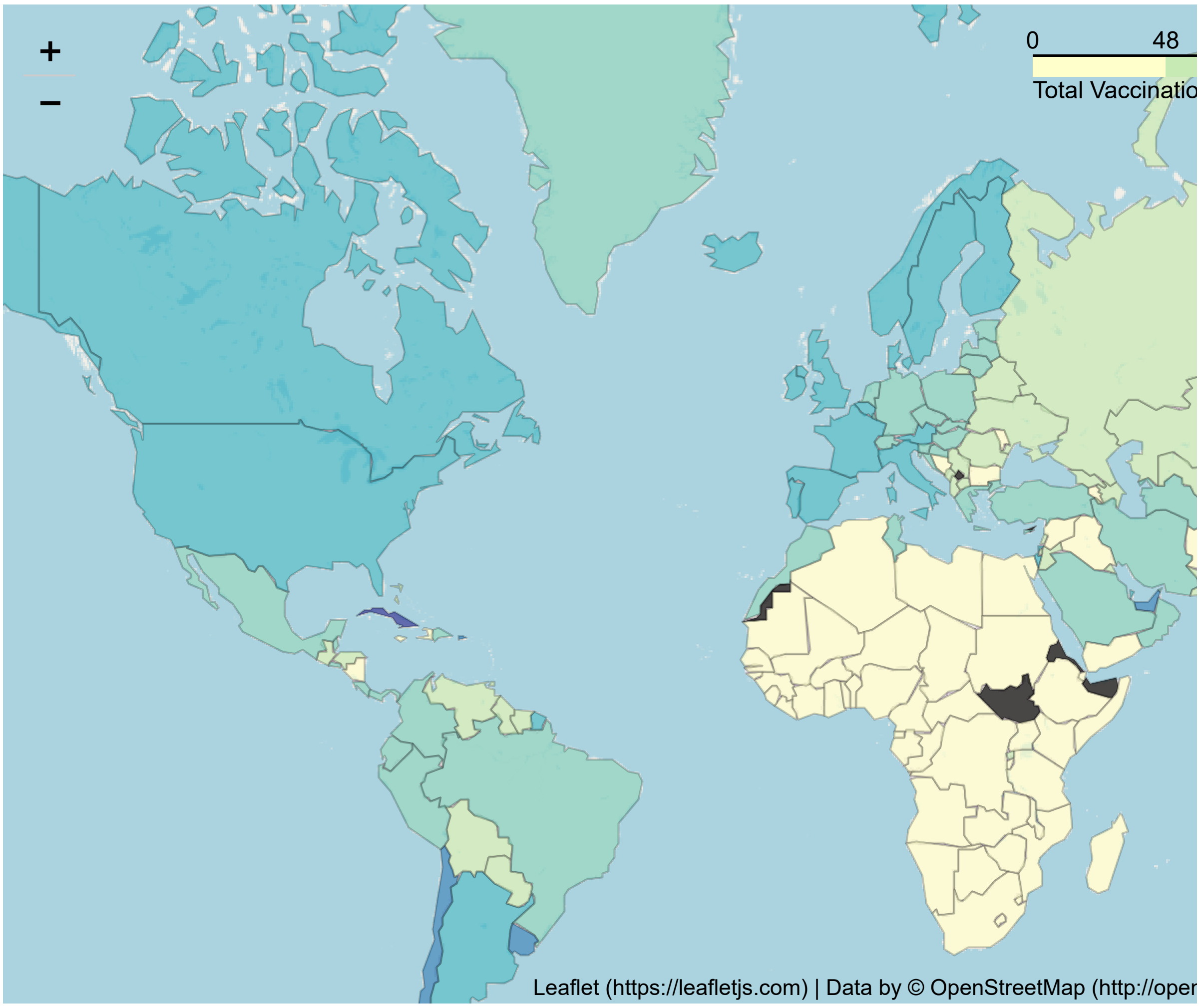
In [100]:

```
map.save('plot_data1.html')
```

In [101]:

```
# Import the Folium interactive html file
from IPython.display import IFrame
IFrame(src='plot_data1.html', width=950, height=500)
```

Out[101]:



In [102]:

```
plot_data2 = data2.iloc[:,[0,7]]
```

In [103]:

```
plot_data2 = plot_data2.sort_values(by='TOTAL_VACCINATIONS_PER100',ascending = False).reset
```

In [104]:

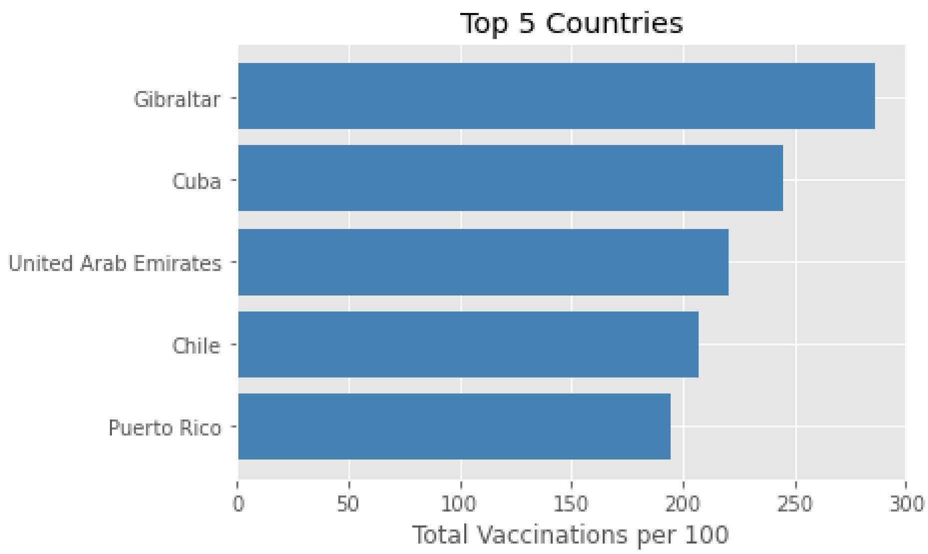
```
plot_top5 = plot_data2.head(5)
plot_top5.head()
```

Out[104]:

	COUNTRY	TOTAL_VACCINATIONS_PER100
0	Gibraltar	286.103
1	Cuba	244.927
2	United Arab Emirates	220.237
3	Chile	207.375
4	Puerto Rico	194.819

In [105]:

```
plt.style.use('ggplot')
plt.barh(plot_top5["COUNTRY"].values,plot_top5["TOTAL_VACCINATIONS_PER100"].values,color ='
plt.title('Top 5 Countries')
#plt.ylabel('Country')
plt.xlabel('Total Vaccinations per 100')
plt.gca().invert_yaxis()
plt.show()
```



In [106]:

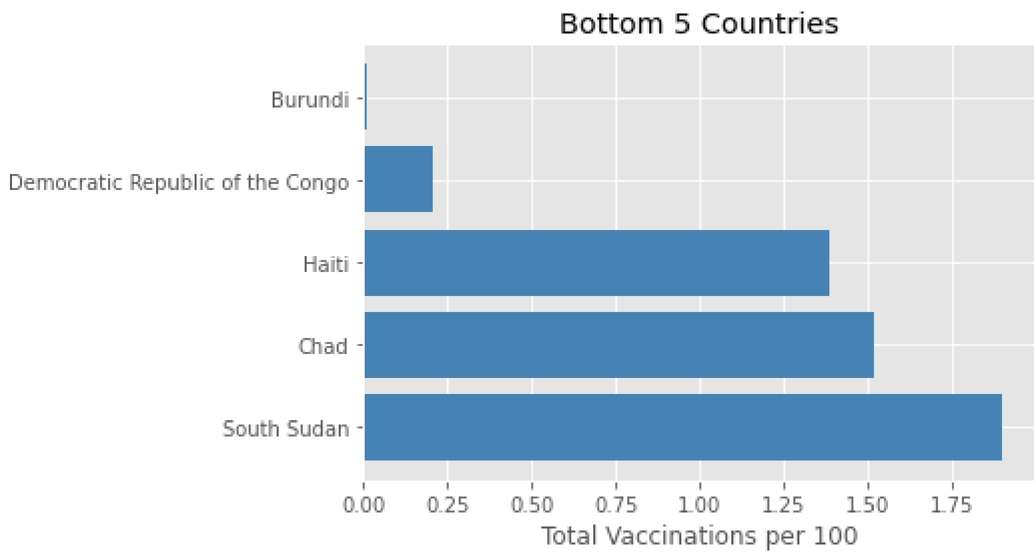
```
plot_bottom5 = plot_data2.tail(5)
plot_bottom5.head()
```

Out[106]:

	COUNTRY	TOTAL_VACCINATIONS_PER100
223	South Sudan	1.902
224	Chad	1.520
225	Haiti	1.390
226	Democratic Republic of the Congo	0.209
227	Burundi	0.012

In [107]:

```
plt.style.use('ggplot')
plt.barh(plot_bottom5["COUNTRY"].values,plot_bottom5["TOTAL_VACCINATIONS_PER100"].values,co
plt.title('Bottom 5 Countries')
#plt.ylabel('Country')
plt.xlabel('Total Vaccinations per 100')
plt.show()
```



In [ ]: