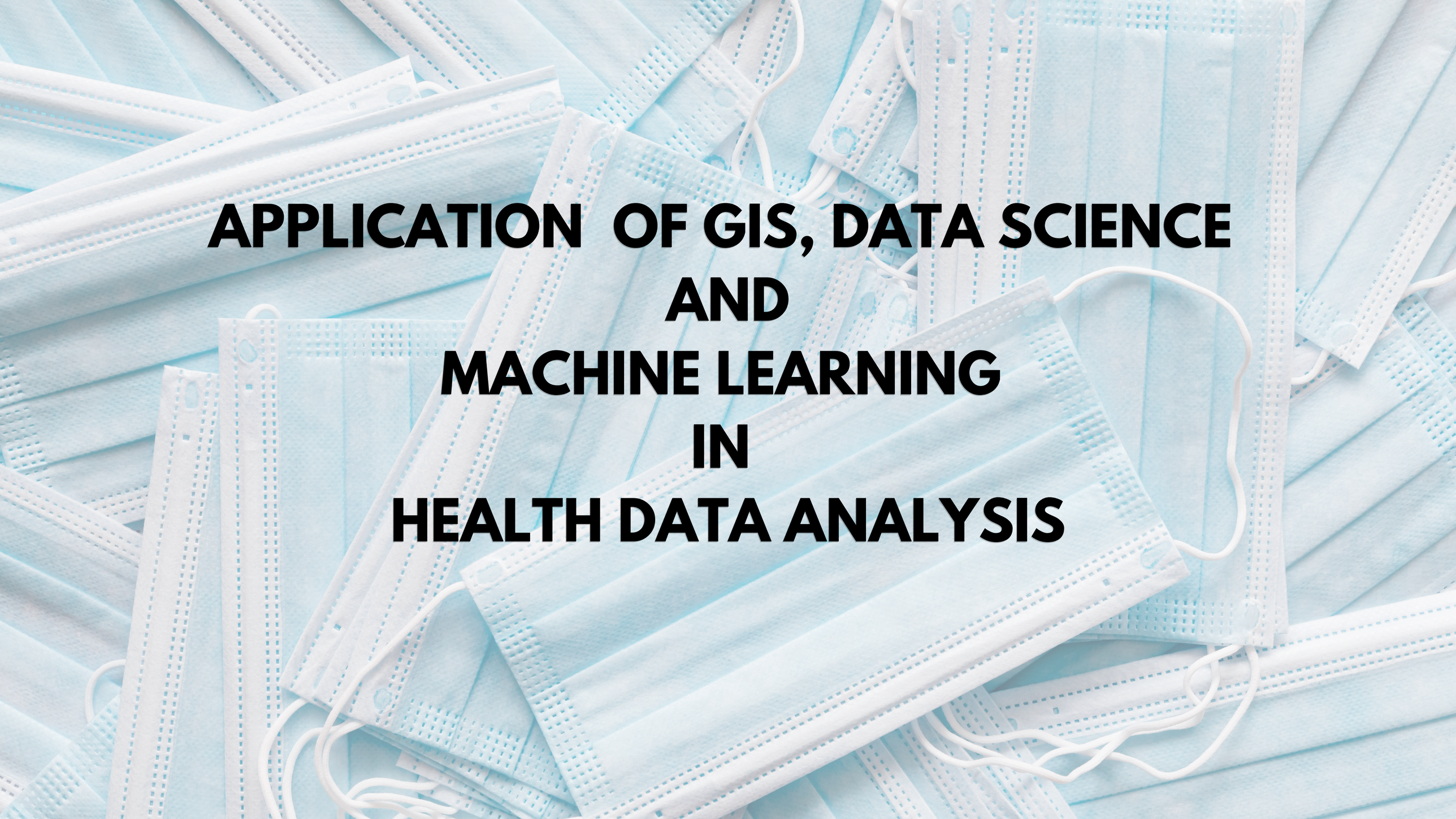
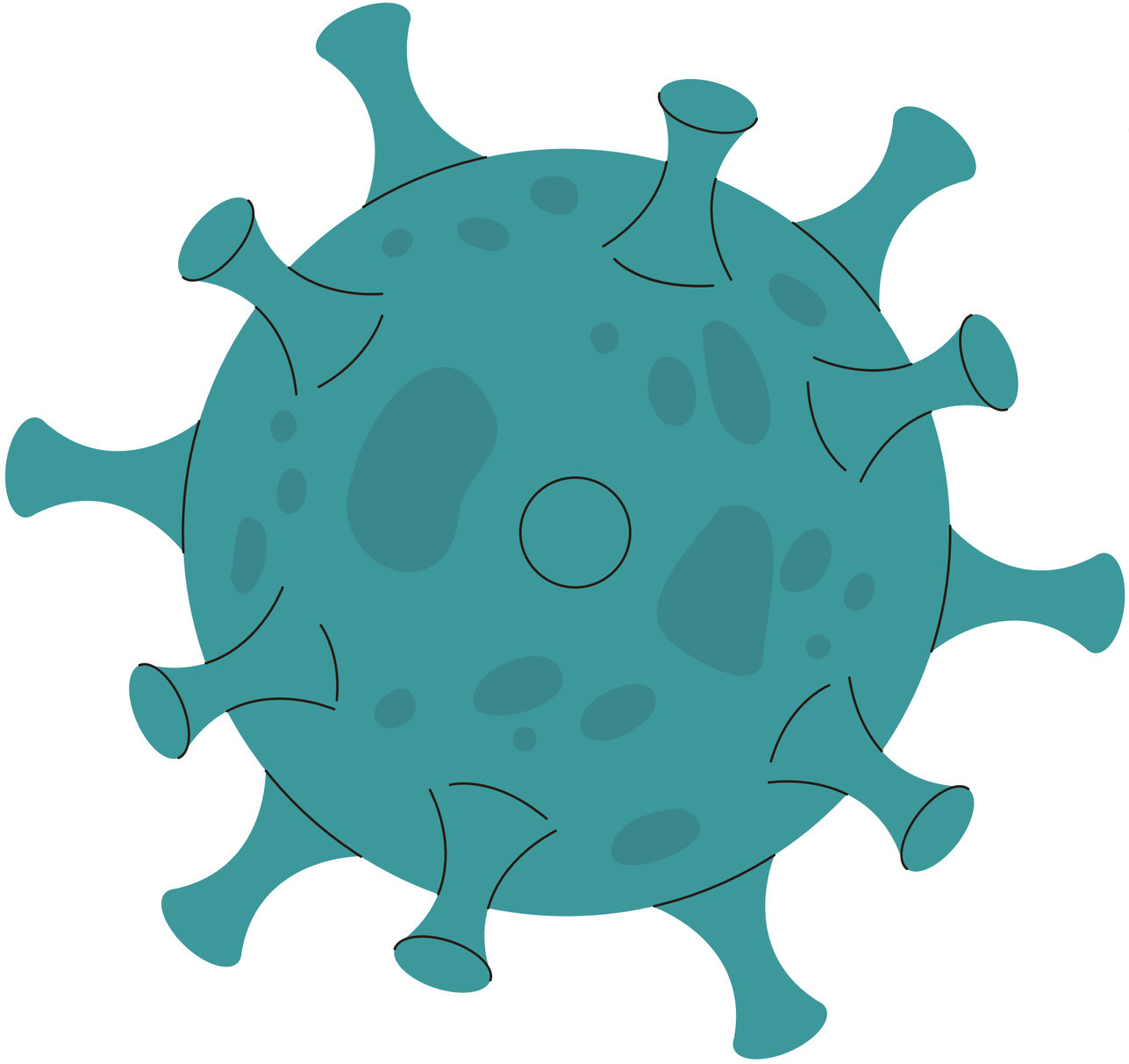


APPLICATION OF GIS, DATA SCIENCE AND MACHINE LEARNING IN HEALTH DATA ANALYSIS



Covid-19 Outbreak:

Presented BY:

Esther Wambui

Emily Kimani

Brenda Onyango

PRESENTERS' PROFILES

Esther Wambui

Data Scientist,
Data Analyst,
Python Guru

Emily Kimani

GIS Analyst,
Data Scientist,
Data Analyst

Brenda Onyango

Application Administrator,
SQL/ Database Analyst

What is Data Science?

It refers to the combination of programming (in our case python), preparation, as well as application of statistical methods to extract meaningful deductions and insights from data.

Machine Learning

Refers to the integration of statistics and computer science or simply put, AI and giving the computer the ability to learn and improve without necessarily following a programming code. The computer learns from instructions, and learns by identifying patterns.

GIS - Geographic Information System

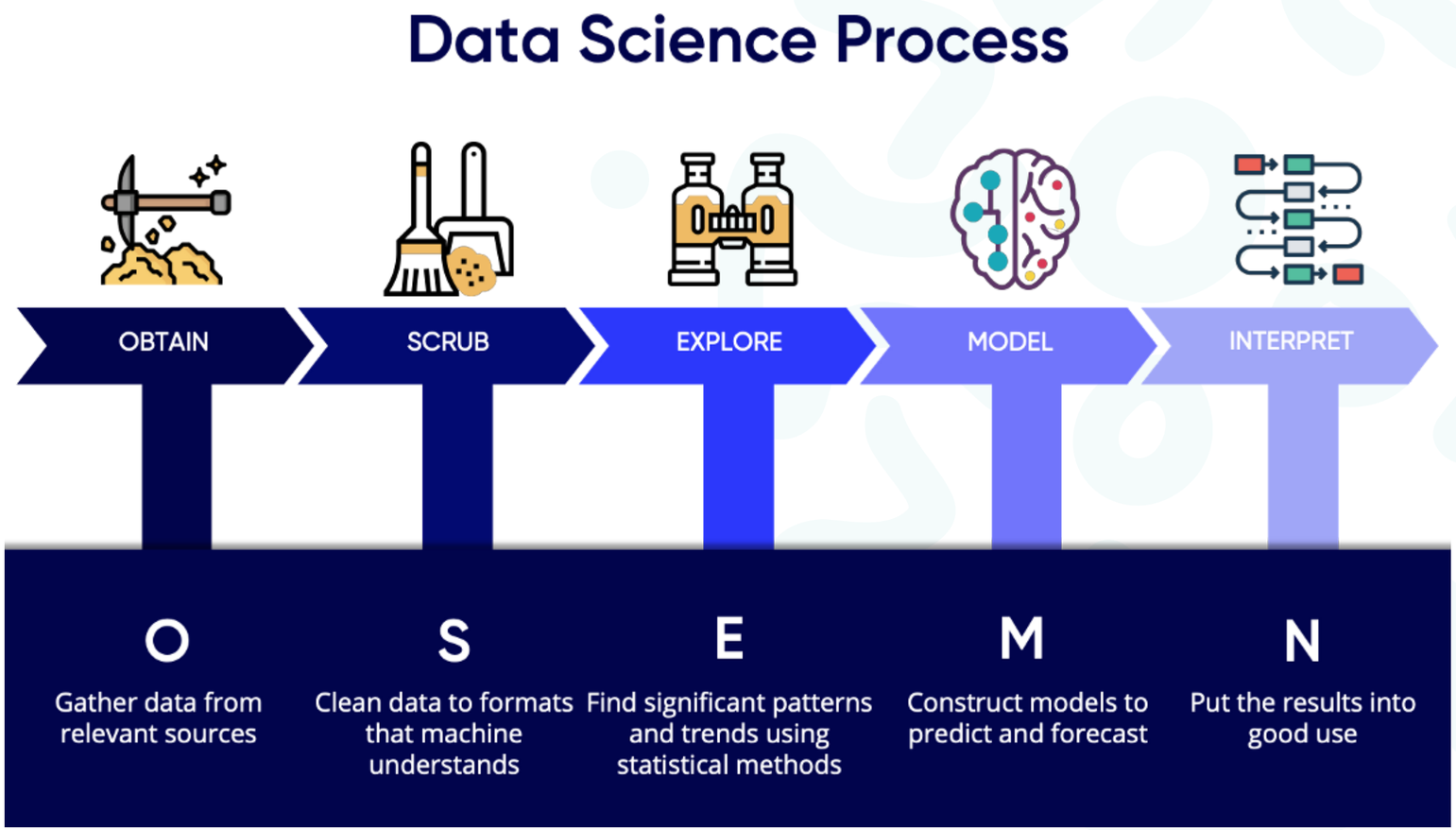
It is a computer system that allows us to capture, analyze and present data on a map. It allows for better informational use based on geography.

Health Data Analysis

Refers to the analysis of current and historical data to make predictions and provide actional insights that help improve health data management.

UNDERSTANDING DATA ANALYSIS

What is it: It is the systematic application of both logical and statistical methods to describe, condense and evaulate cleaned data.

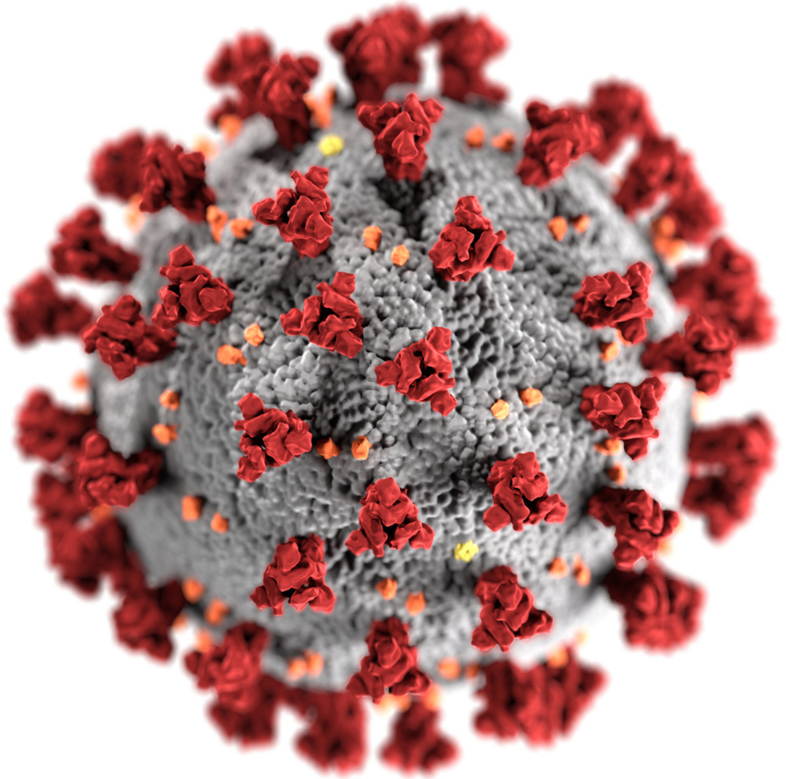


SOME OF THE LIBRARIES WE USED

- Pandas
- Numpy
- Seaborn
- Matplotlib
- Folium
- Geopandas
- Shapely geometry
- Scikit Learn

OBJECTIVES

- To determine the prevalence and spread of Covid-19 at a global and National scale.
- To assess the trend of the current death and infection rate.
- To present an in-depth analysis on the vaccine administration by regions.
- To predict the vaccination coverage in the US.



Data Source

Data used in this project was retrieved from:
owid-datasets

https://covid.ourworldindata.org/data/owid-covid-data.csv

Disclaimer

All information was sourced from owid-datasets in good faith, however we make no representation or warranty of any kind, express or implied, regarding the accuracy, adequacy, validity, reliability, availability or completeness of any information on their site.

Data Preparation

- The original Dataset had: **df.shape** -> **97585 Rows, 60 Columns**
- Dropped the Unwanted Columns:

df.drop(columns=['List Column_names - 30 columns'], inplace= True)

- Checked for missing Values : **df.apply(lambda x: sum(x.isnull()),axis=0)**
- Replaced missing values with Zeros:

df.update(df[['List Column_names with missing values']].fillna(0))

- Renamed some columns: **df = df.rename(columns={'location':'country'})**
- Corrected Inaccurate data - rows that had Country name as the Continent name:

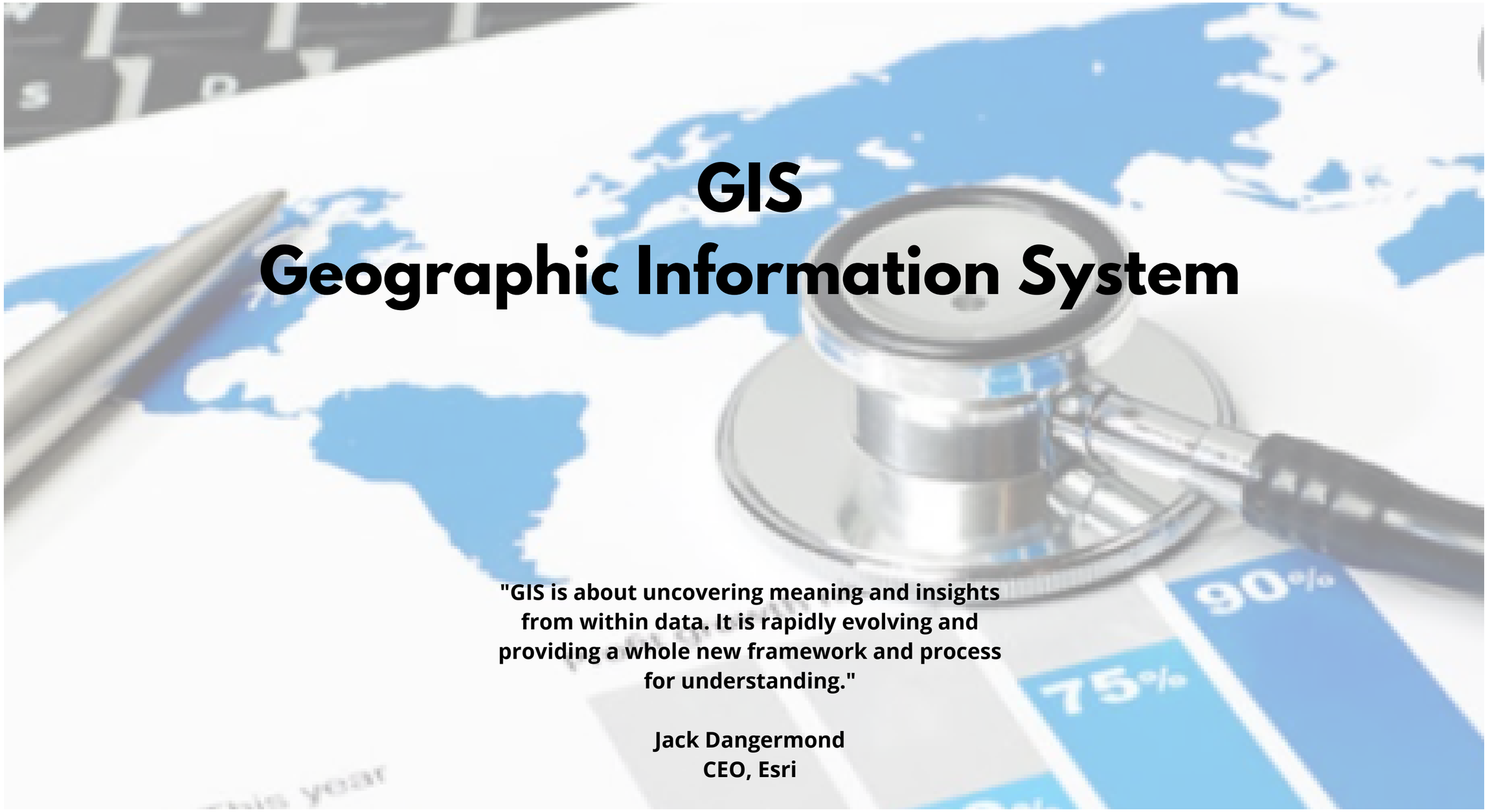
flt =(df['country'] == 'Asia') - 4588 Rows

- Deleted rows with wrong country names:

df.drop(df_nan_continents.index, axis=0, inplace = True)

- Checked again to make sure all the missing values have been filled.
- Finally, after all the data cleaning, our Dataset has: **92997 Rows, 30 Columns.**


EMILY KIMANI
Profit growth ra
90%
75%



GIS
Geographic Information System

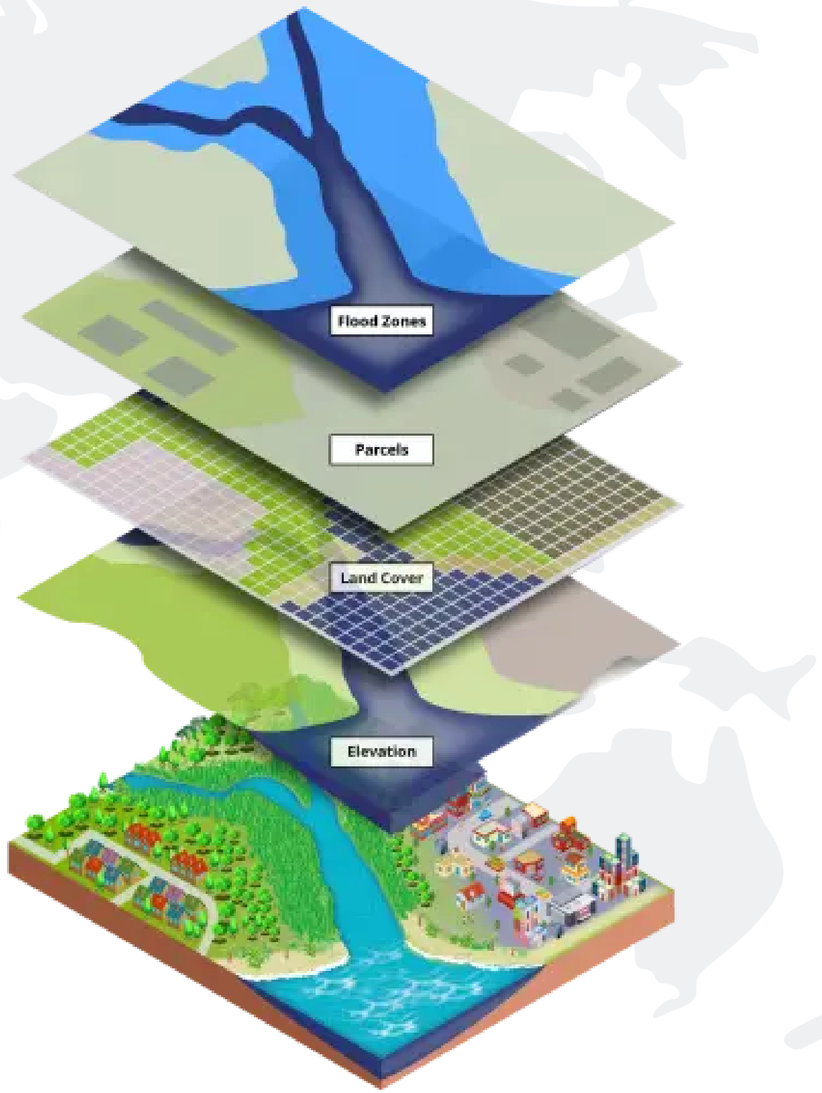
"GIS is about uncovering meaning and insights from within data. It is rapidly evolving and providing a whole new framework and process for understanding."

Jack Dangermond
CEO, Esri

What is GIS?

- A spatial system that creates, manages, analyzes, and visualizes data for geographic positions on Earth's surface.

- GIS connects data to a map, integrating location data (where things are)
with all types of descriptive information (what things are like there).



History of GIS

1854 - Cholera outbreak hit the city of London. British physician John Snow started mapping out the reported cases, roads, property boundaries, and water lines on a map.

1960 - computers and early concepts of quantitative and computational geography started to emerge.

1963 - Roger Tomlinson (aka 'the Father of GIS') - pioneered work to initiate, plan, and develop the Canada Natural Resource GIS which became the 1st computerized GIS.

1965 - Howard Fisher - created one of the first computer mapping software program. He also established the Harvard Laboratory for Computer Graphics.

1969 - Jack Dangermond - founded Environmental Systems Research Institute, Inc. (Esri). The firm applied computer mapping and spatial analysis to help land use planners and land resource managers make informed decisions.

1981 - Esri developed ARC/INFO—the first commercial GIS product and this started it's evolution into a Software Company.

Where is GIS Being Used?

GIS is being used in agriculture, conservation, natural resources, public safety, transportation, urban planning and more...

The Future of GIS

Its move toward web and cloud computing, and its integration with real-time information via the Internet of Things, has become a platform relevant to almost every human endeavor.

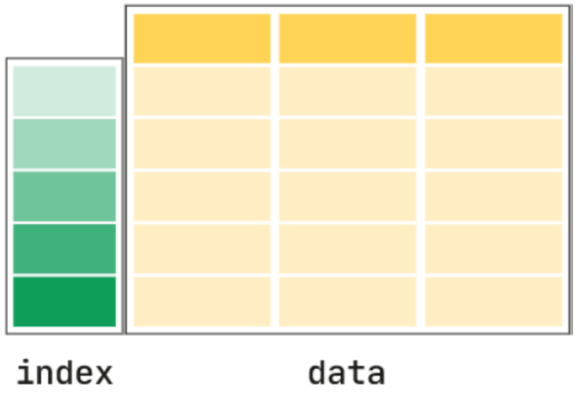
GeoPandas

- It is a free open-sourced library which enables the use and manipulation of geospatial data in Python.

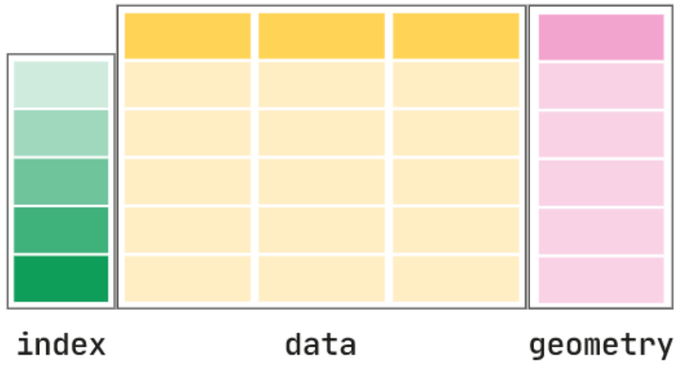
- It extends the common datatype used in pandas to allow for the many and unique geometric operations: GeoSeries and GeoDataFrame.

- This operations are performed by shapely library. It also depends on fiona for file access & matplotlib for plotting.

Pandas DataFrame



GeoPandas DataFrame

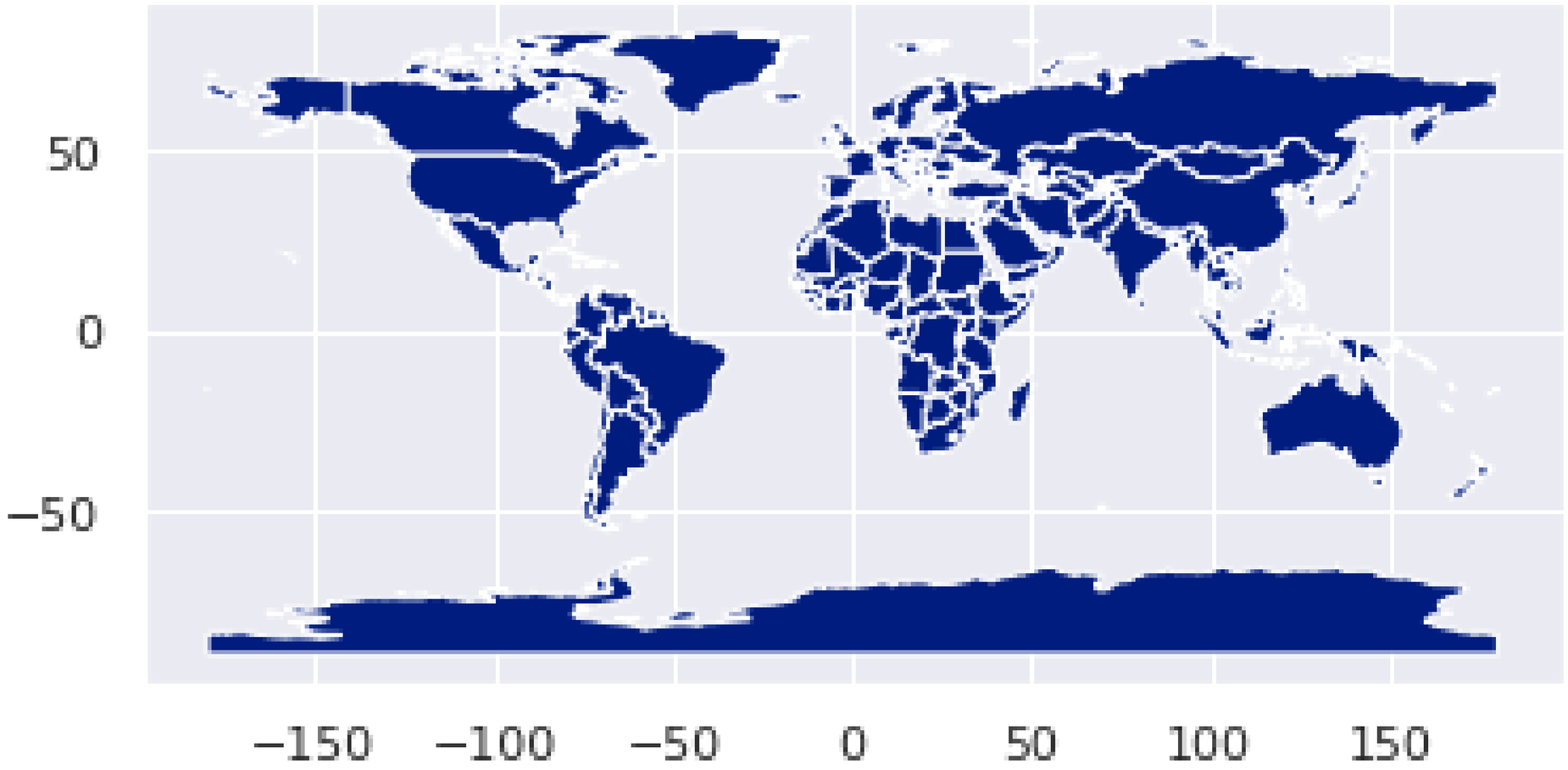


Plotting Using GeoPandas

```
# World Shapefile
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
# Cities Shapefile
cities = geopandas.read_file(geopandas.datasets.get_path('naturalearth_cities'))
world.head()
```

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	603253	Africa	W. Sahara	ESH	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...

world.plot();



Merge world shapefile to covid dataset

- We performed a Left-Join to merge the Covid Data to the World Data Shapefile from GeoPandas so the we can visualize the data geographically.

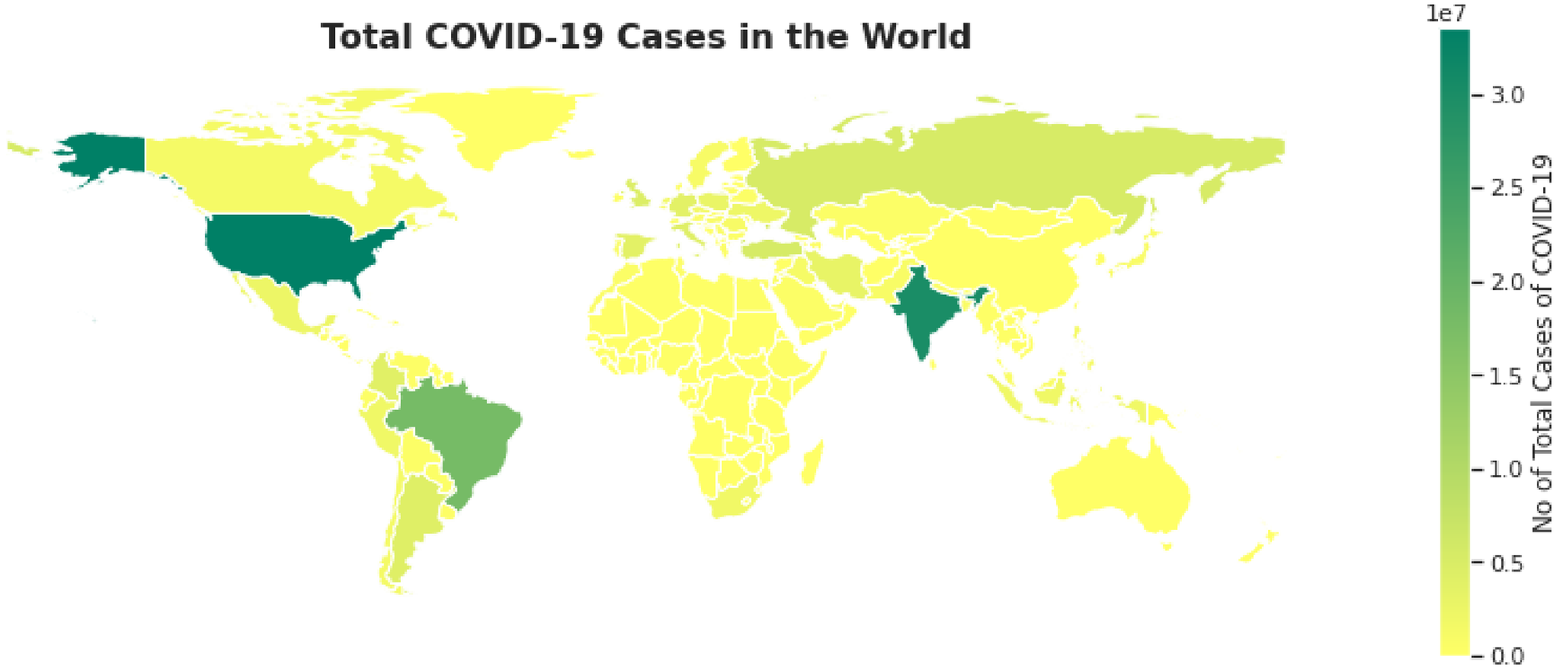
geo_df = df.merge(world, on='iso_code', how='left')

The columns added were:

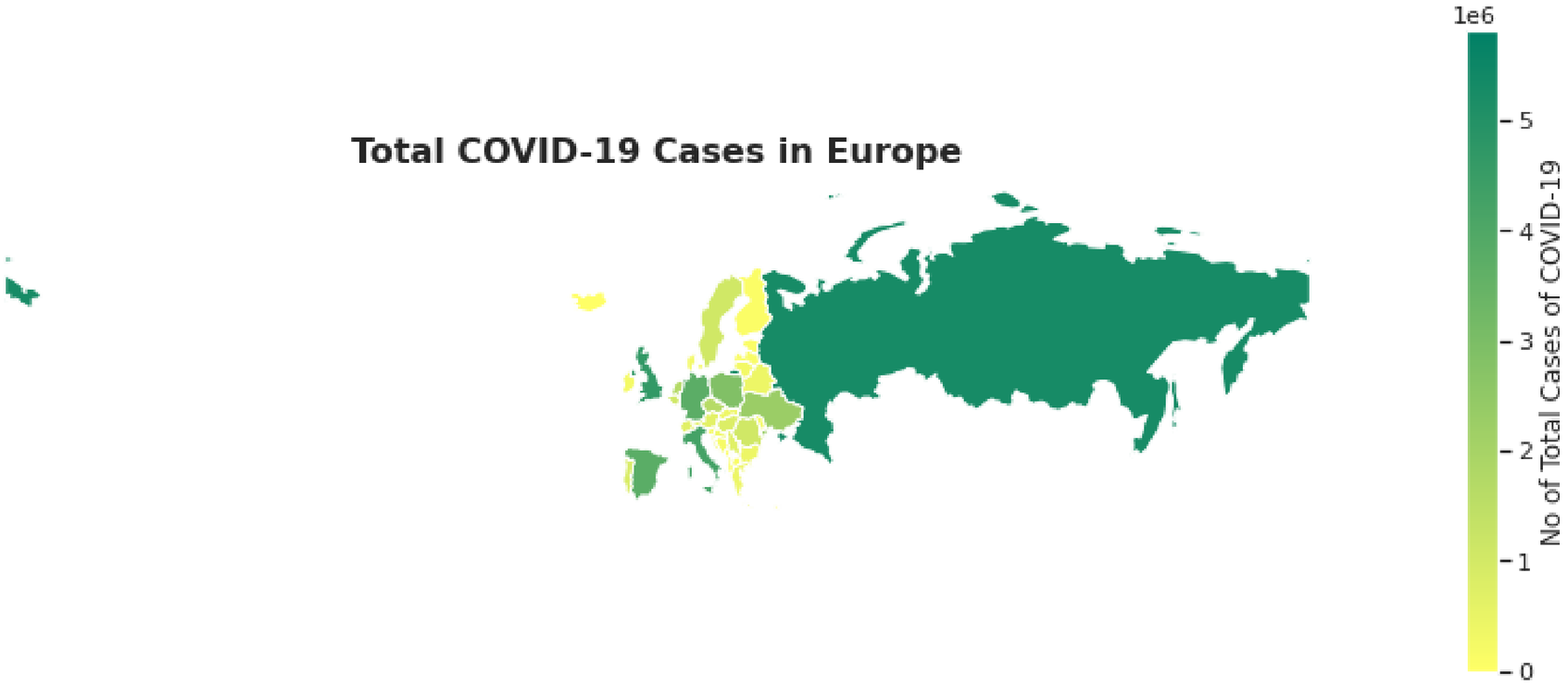
- Continent_y
- Name
- Geometry

Folium Time Series

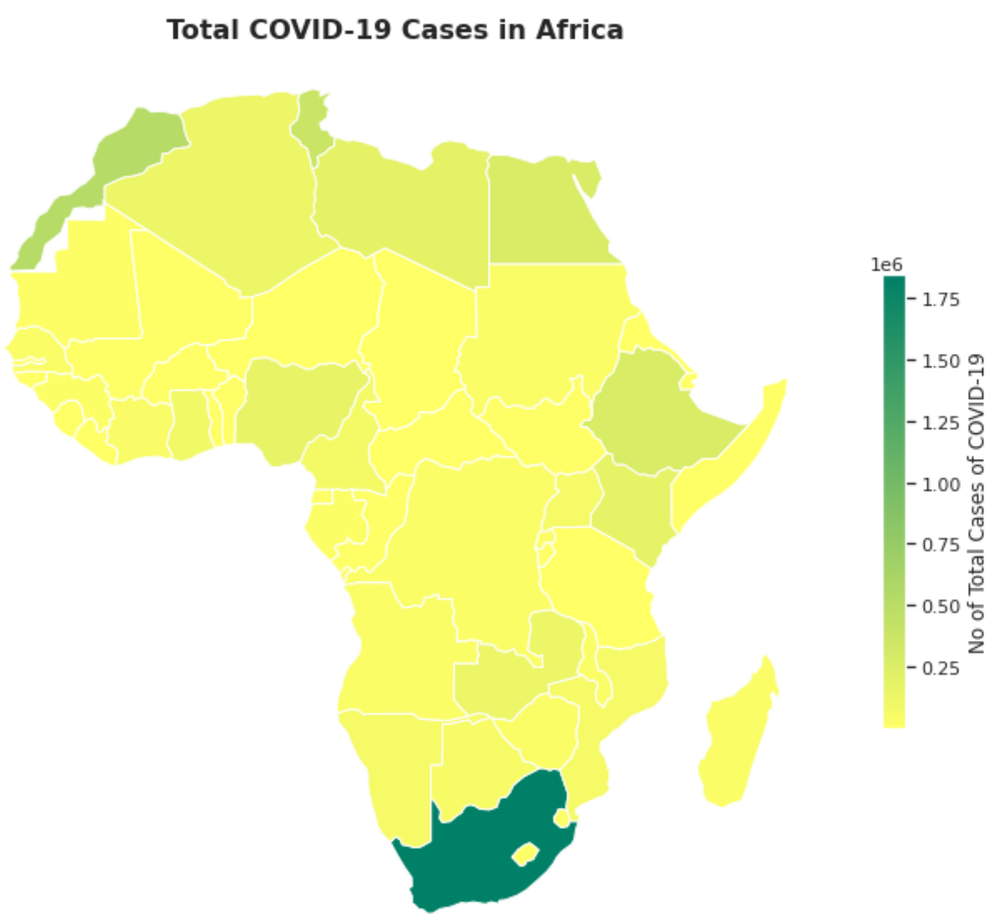
```
from folium.plugins import TimeSliderChoropleth

slider_map = folium.Map(min_zoom=2, max_bounds=True,tiles='cartodbpositron')

_ = TimeSliderChoropleth(
    data=countries_gdf.to_json(),
    styledict=style_dict,

).add_to(slider_map)

_ = cmap.add_to(slider_map)
cmap.caption = "Log of number of Covid Total cases"
slider_map.save(outfile='TotalCasesTimeSliderChoropleth.html')

slider_map
```

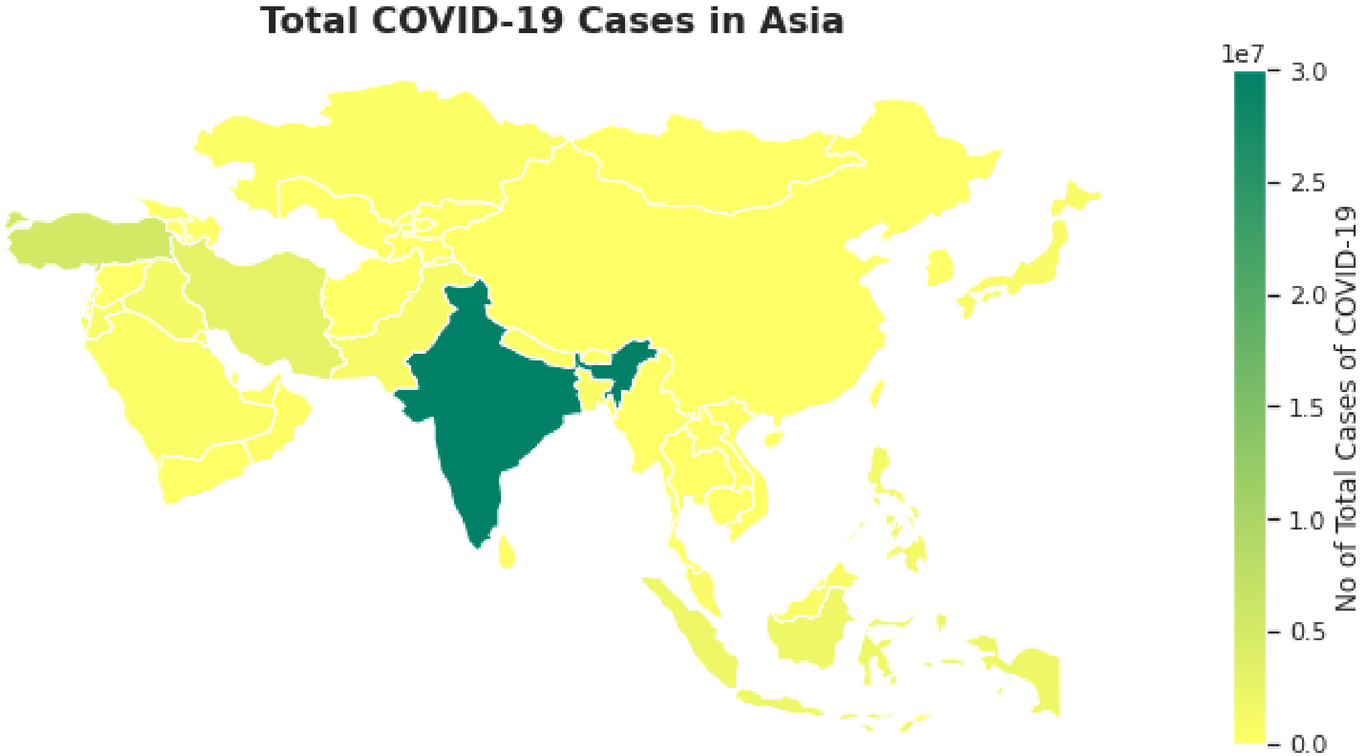



```
# Generate the choropleth map using gdf plot function on the Total Cases

columnfig = june_total_cases_df.plot(column='total_cases',cmap='summer_r',figsize=(15,10), legend = True,
                                     legend_kwds={'label': "No of Total Cases of COVID-19",'orientation': "vertical", 'shrink': 0.55})

# Removing axis ticks
plt.axis('off')

# Add the title
plt.title("Total COVID-19 Cases in the World",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```

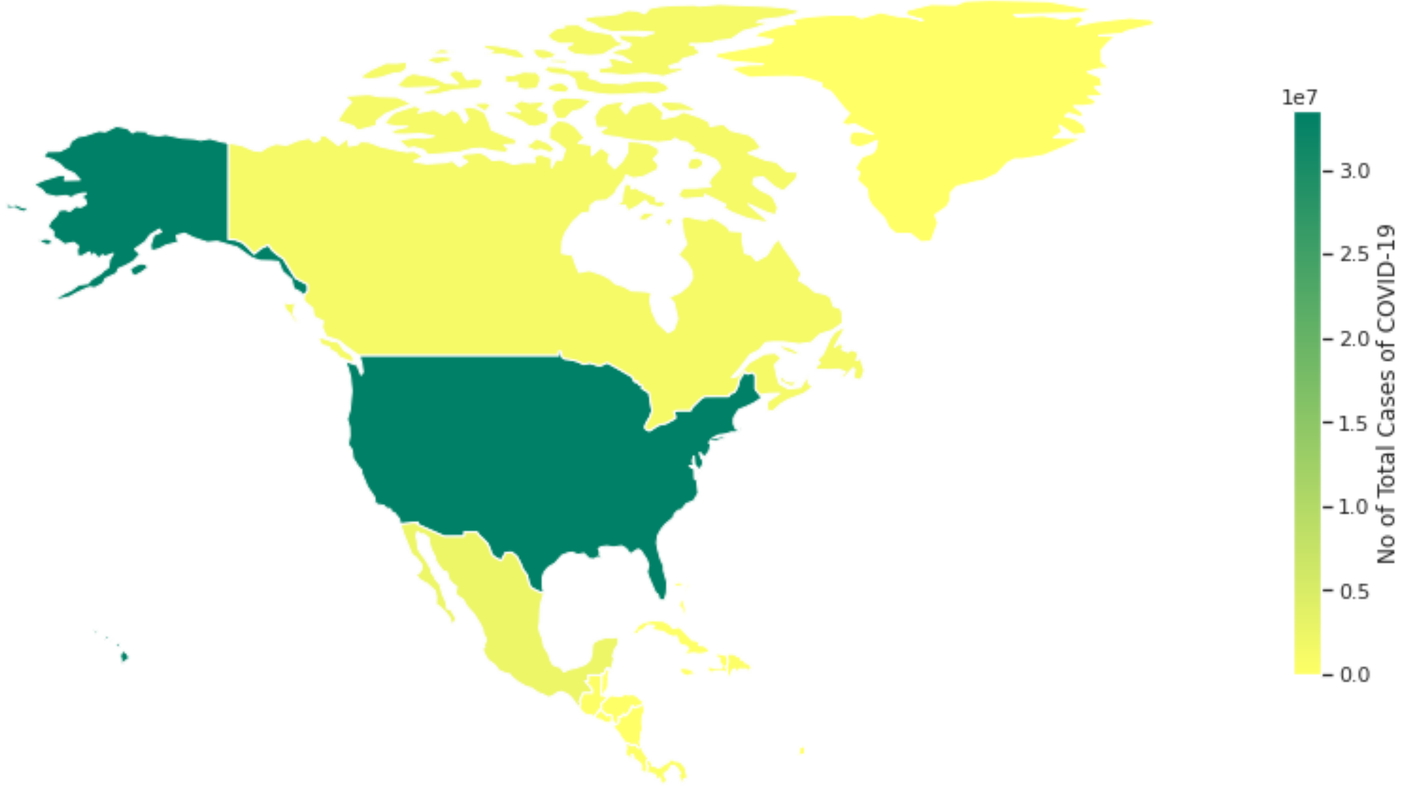



```
map_europe = europe.plot(column='total_cases',cmap='summer_r',figsize=(15,10), legend = True,
                                legend_kwds={'label': "No of Total Cases of COVID-19",'orientation': "vertical", 'shrink': 0.55})

#removing axis ticks
plt.axis('off')

#Add the title
plt.title("Total COVID-19 Cases in Europe",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```

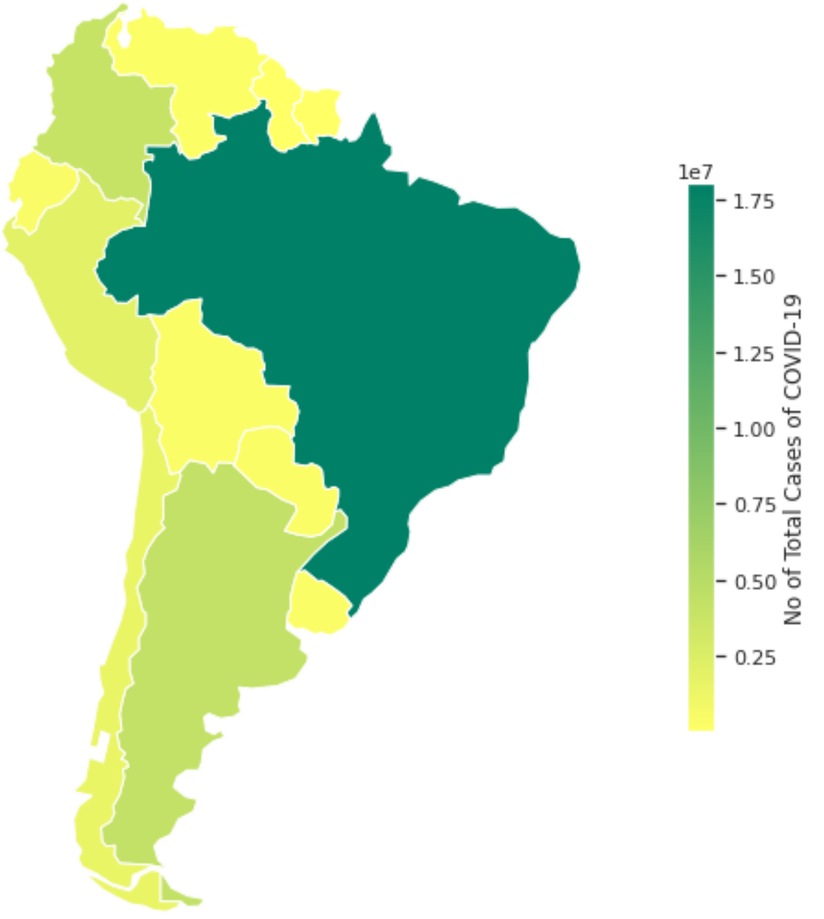

Total COVID-19 Cases in Africa
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
No of Total Cases of COVID-19

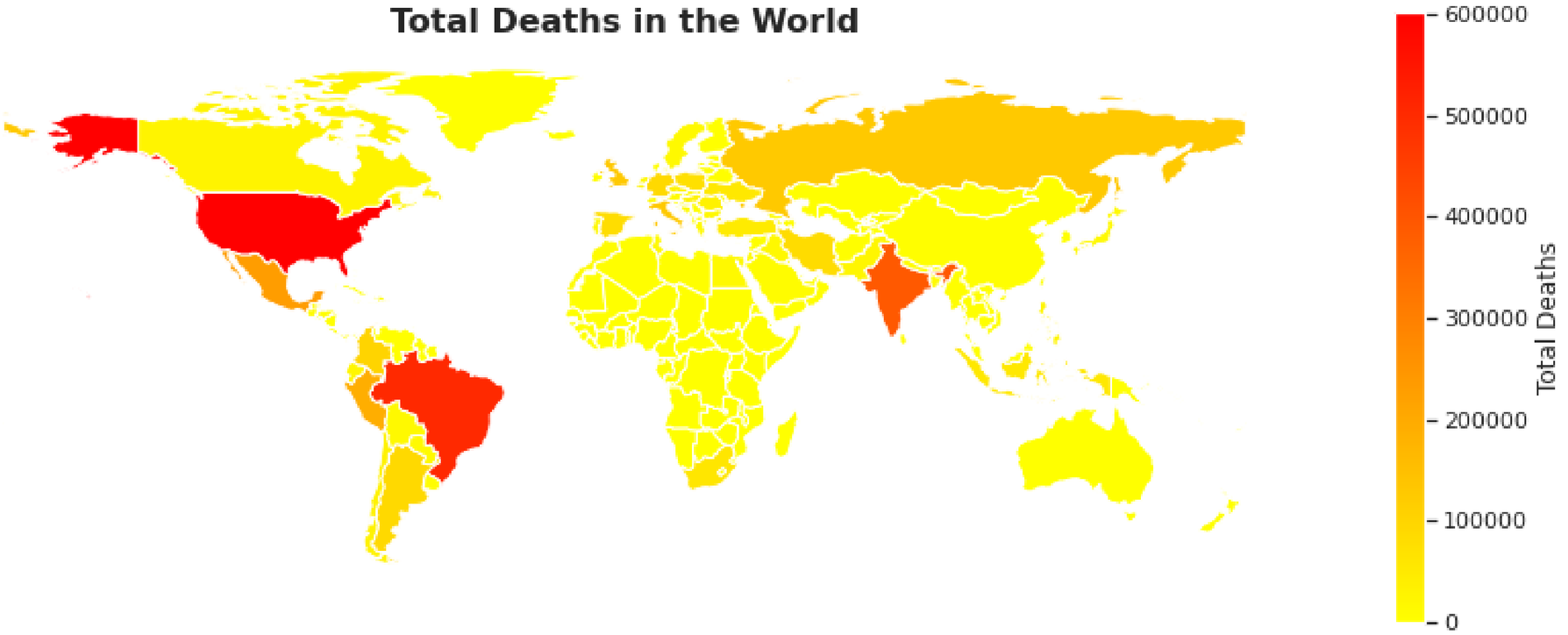

Total COVID-19 Cases in Asia
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
No of Total Cases of COVID-19

Total COVID-19 Cases in North America



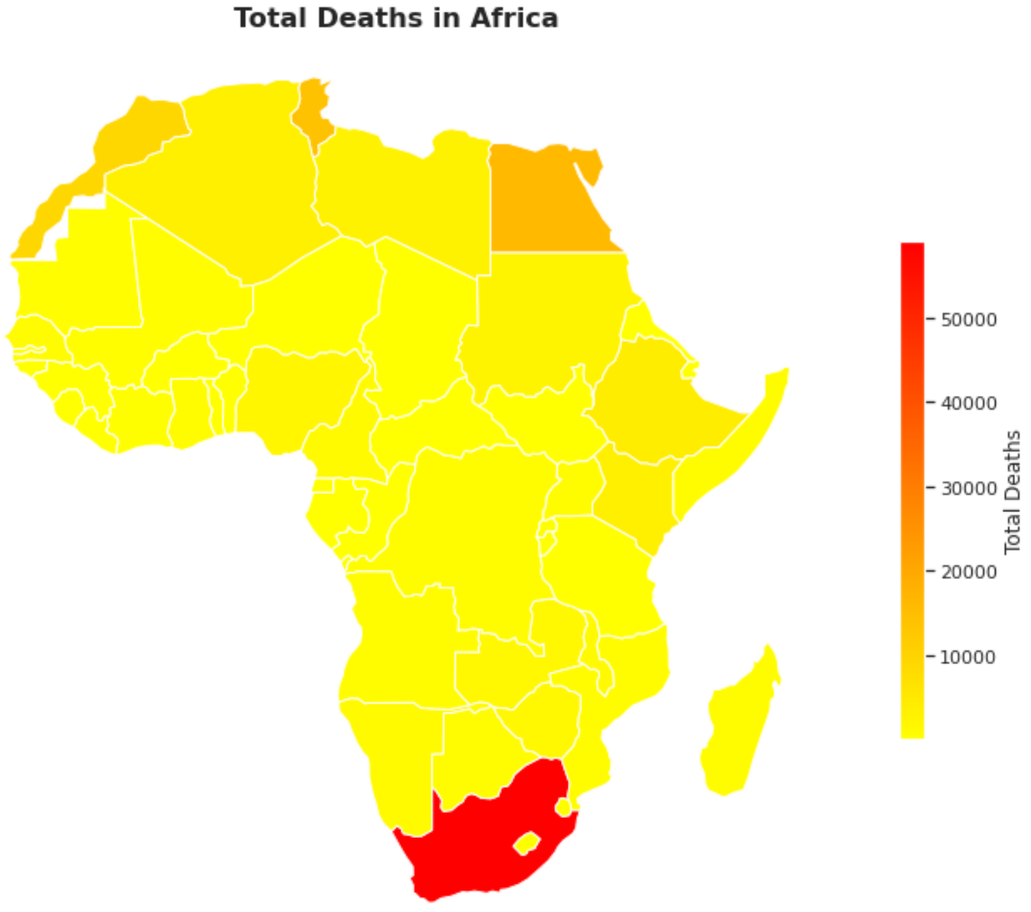
Total COVID-19 Cases in South America

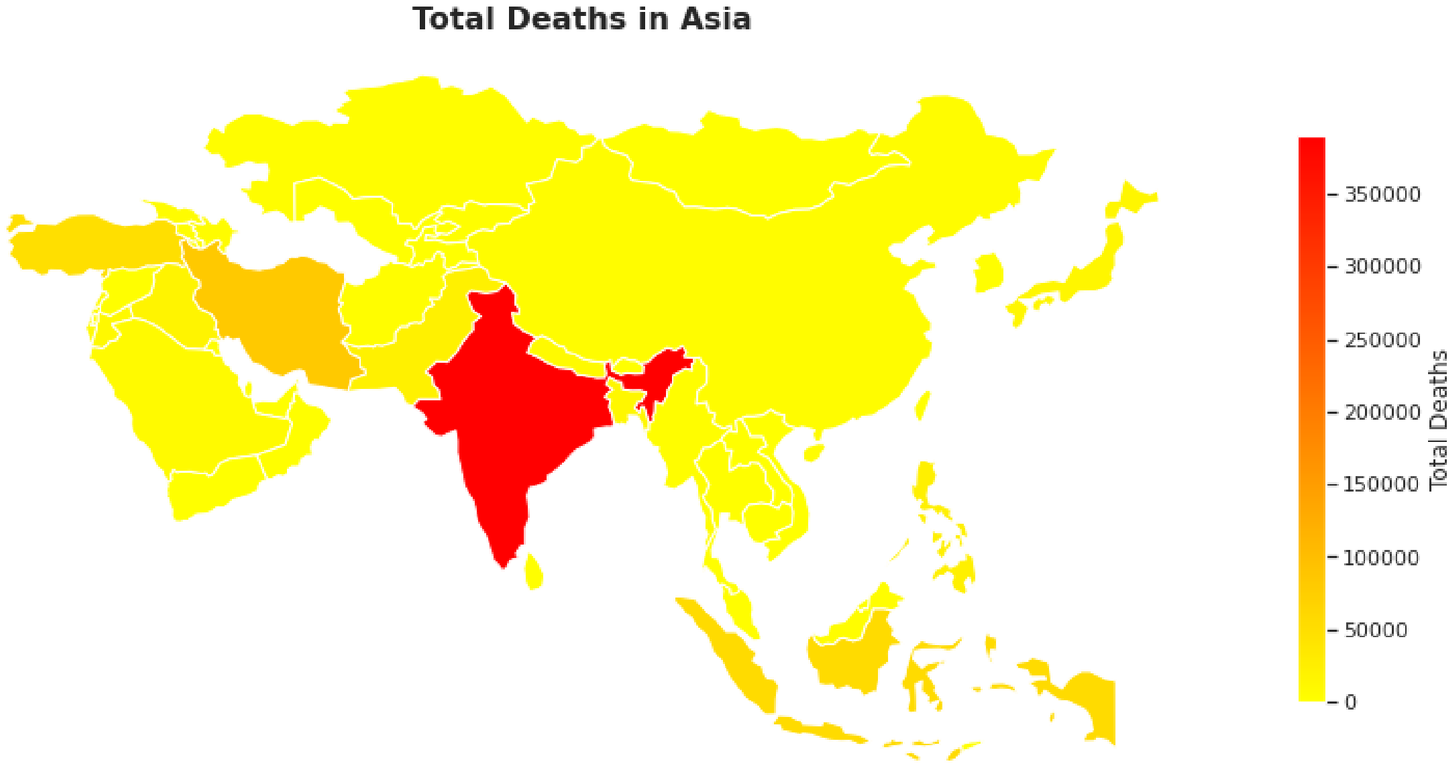




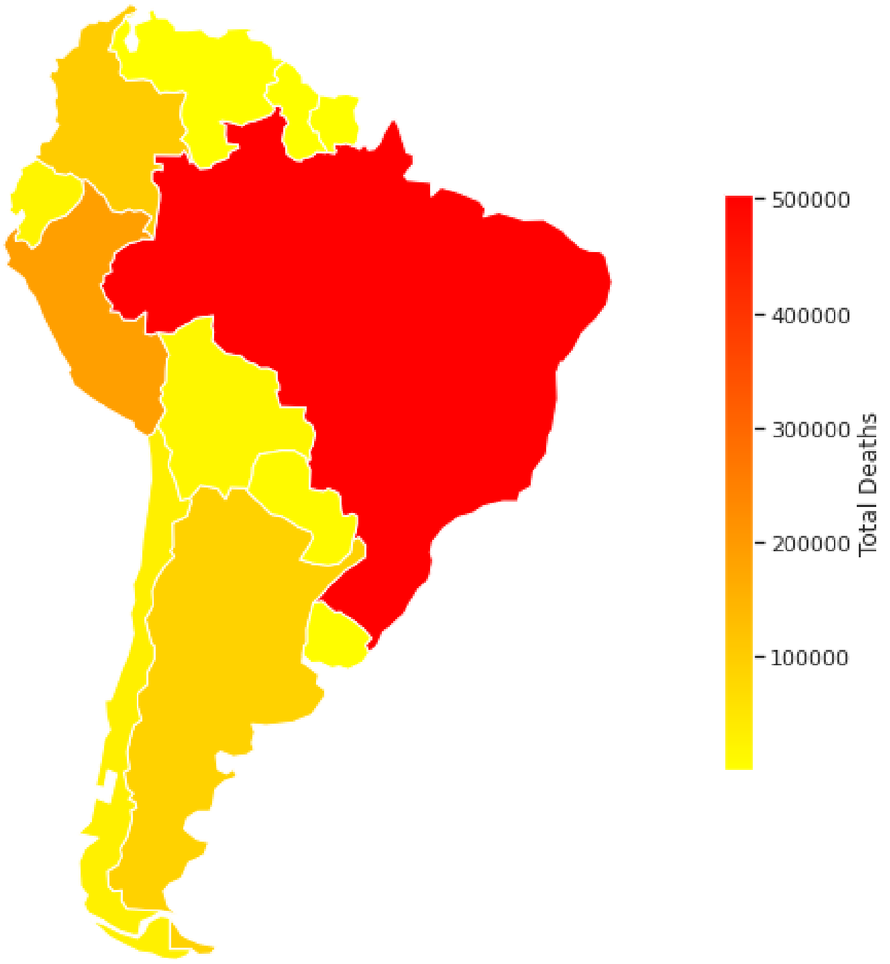

```
#Generate the choropleth map using gdf plot function on the Total Deaths

columnfig = june_deaths_df.plot(column='total_deaths',cmap='autumn_r',figsize=(15,10), legend = True,
                                legend_kwds={'label': "Total Deaths", 'orientation': "vertical", 'shrink': 0.55})

#removing axis ticks
plt.axis('off')

#Add the title
plt.title("Total Deaths in the World",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```


```
map_europe = europe.plot(column='total_deaths',cmap='autumn_r',figsize=(15,10), legend = True,
                         legend_kwds={'label': "Total Fatalities from COVID-19",'orientation': "vertical", 'shrink': 0.55})

#removing axis ticks
plt.axis('off')

#Add the title
plt.title("Total Fatalities in Europe",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```

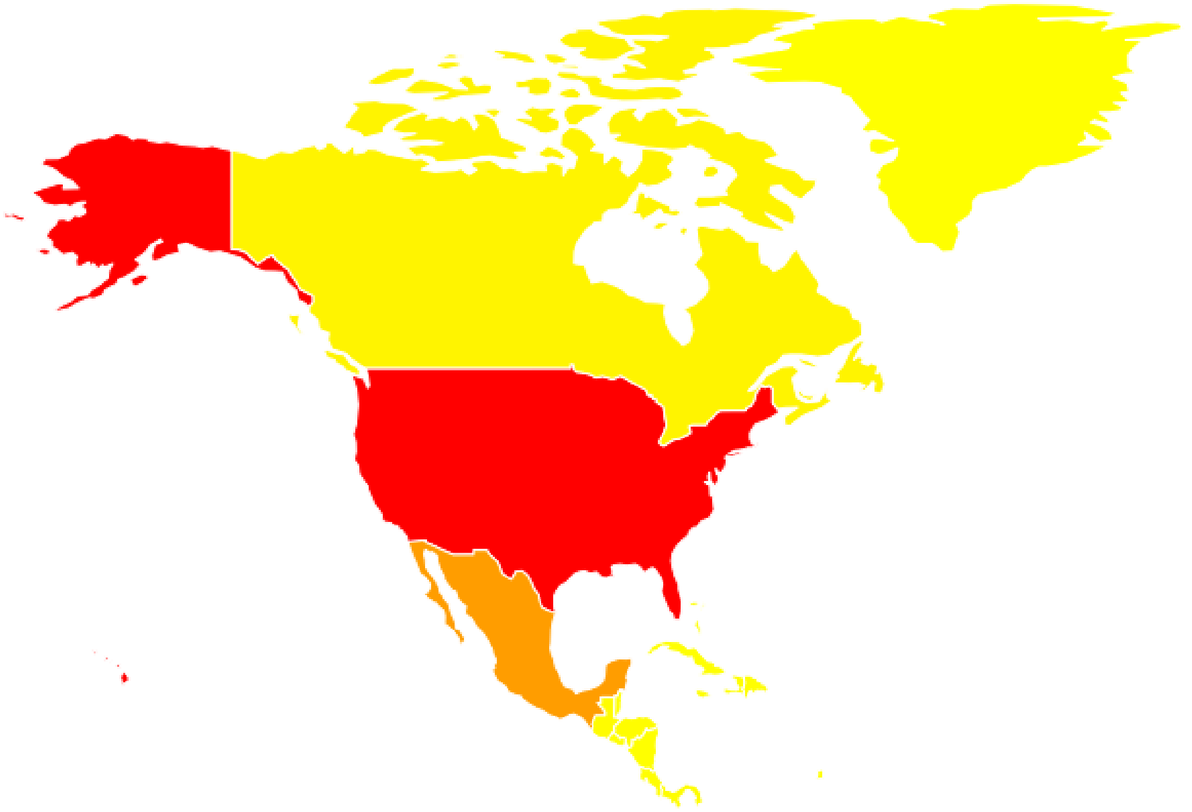

Total Deaths in Africa
50000
40000
30000
20000
10000
Total Deaths

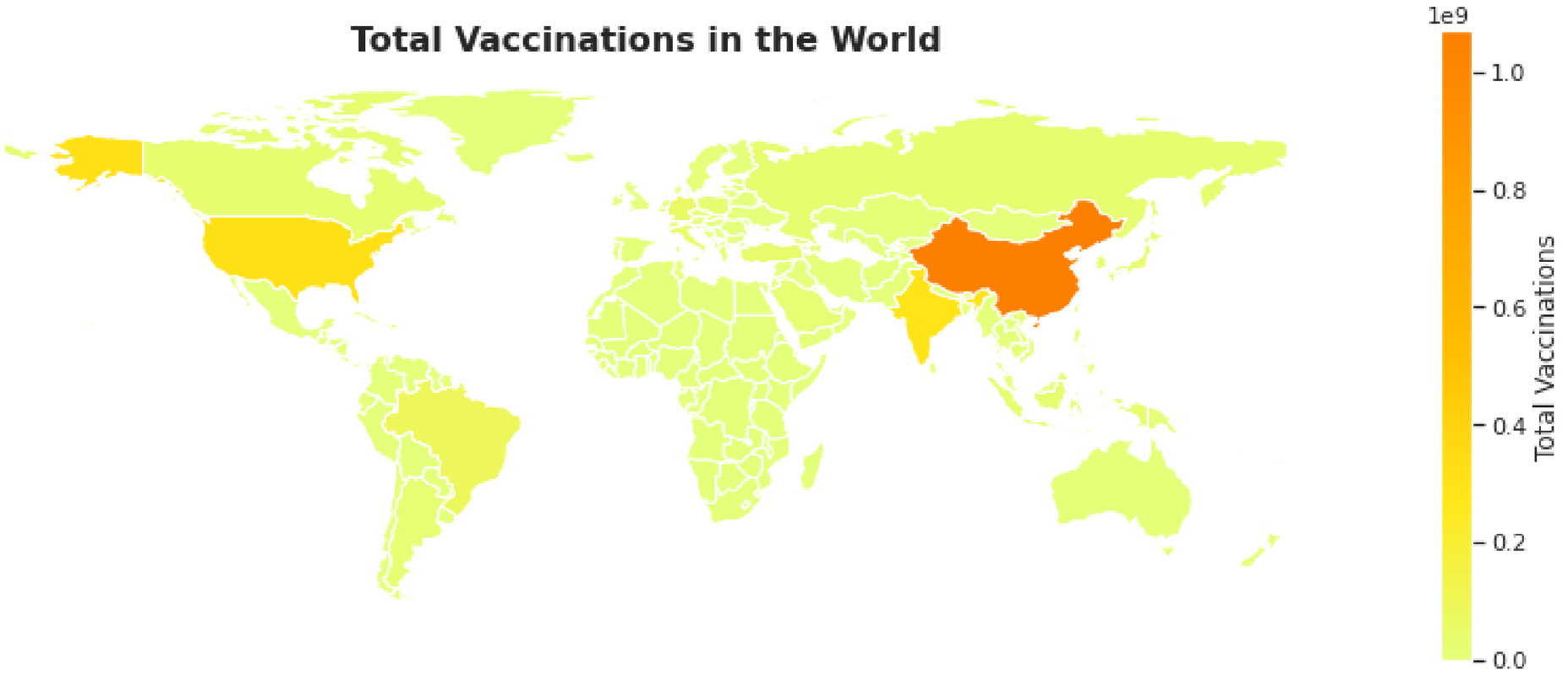

Total Deaths in Asia
350000
300000
250000
200000
150000
100000
50000
0
Total Deaths

Total Deaths in South America



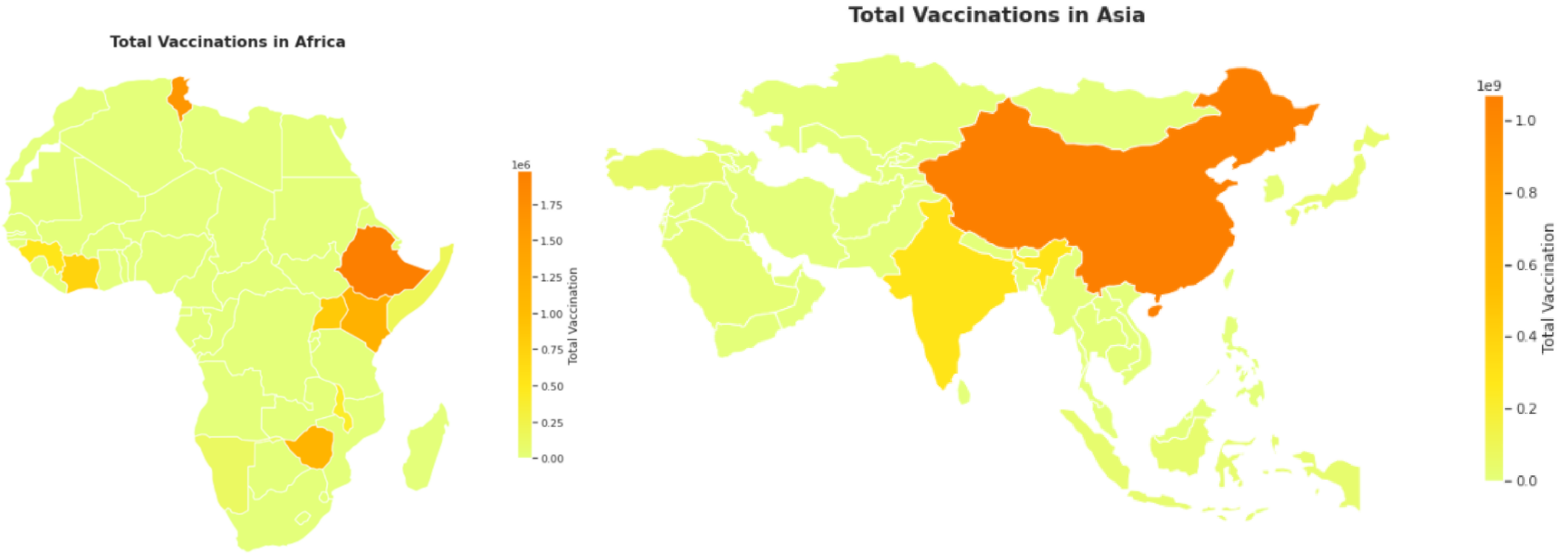
Total Deaths in North America

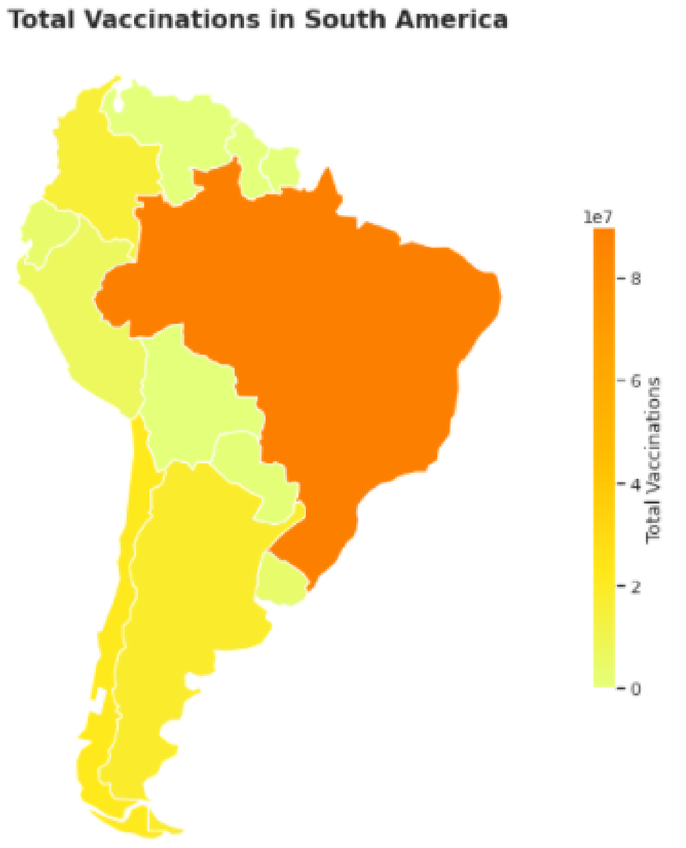


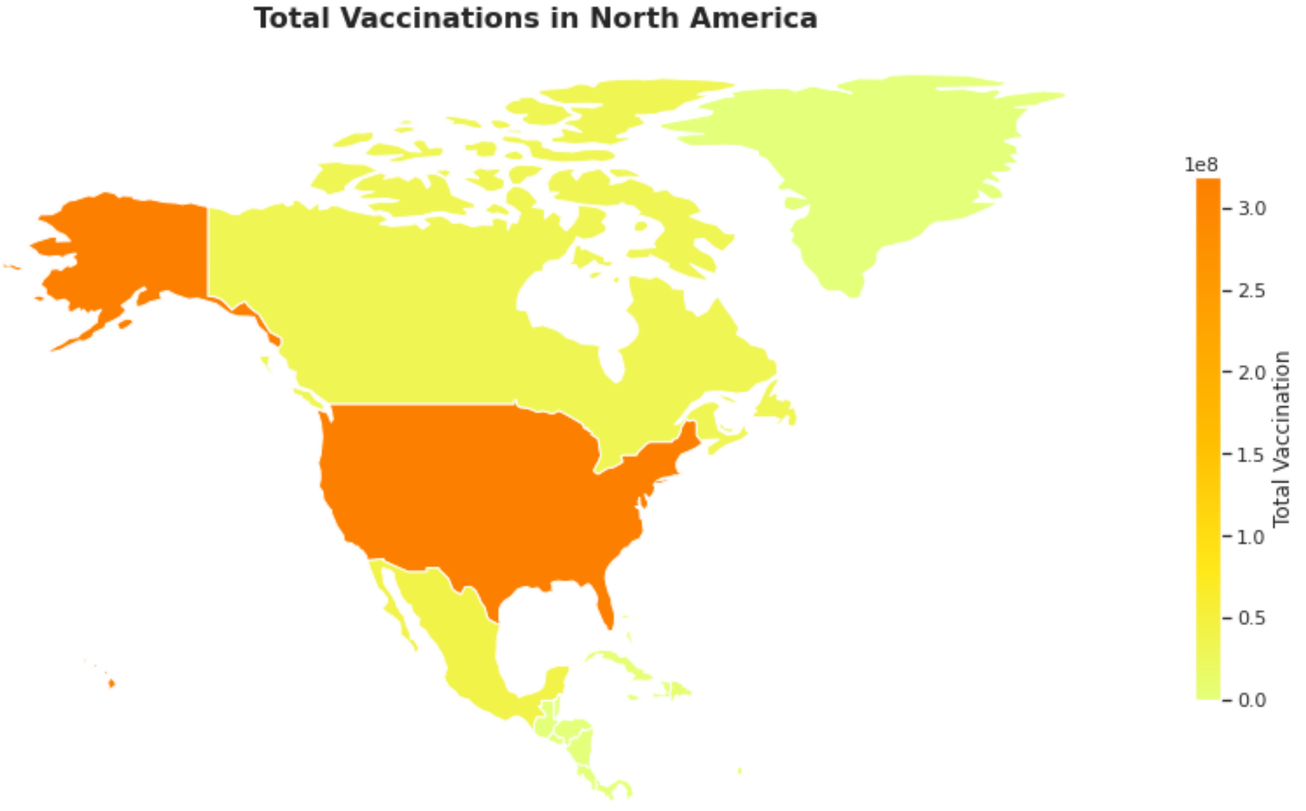


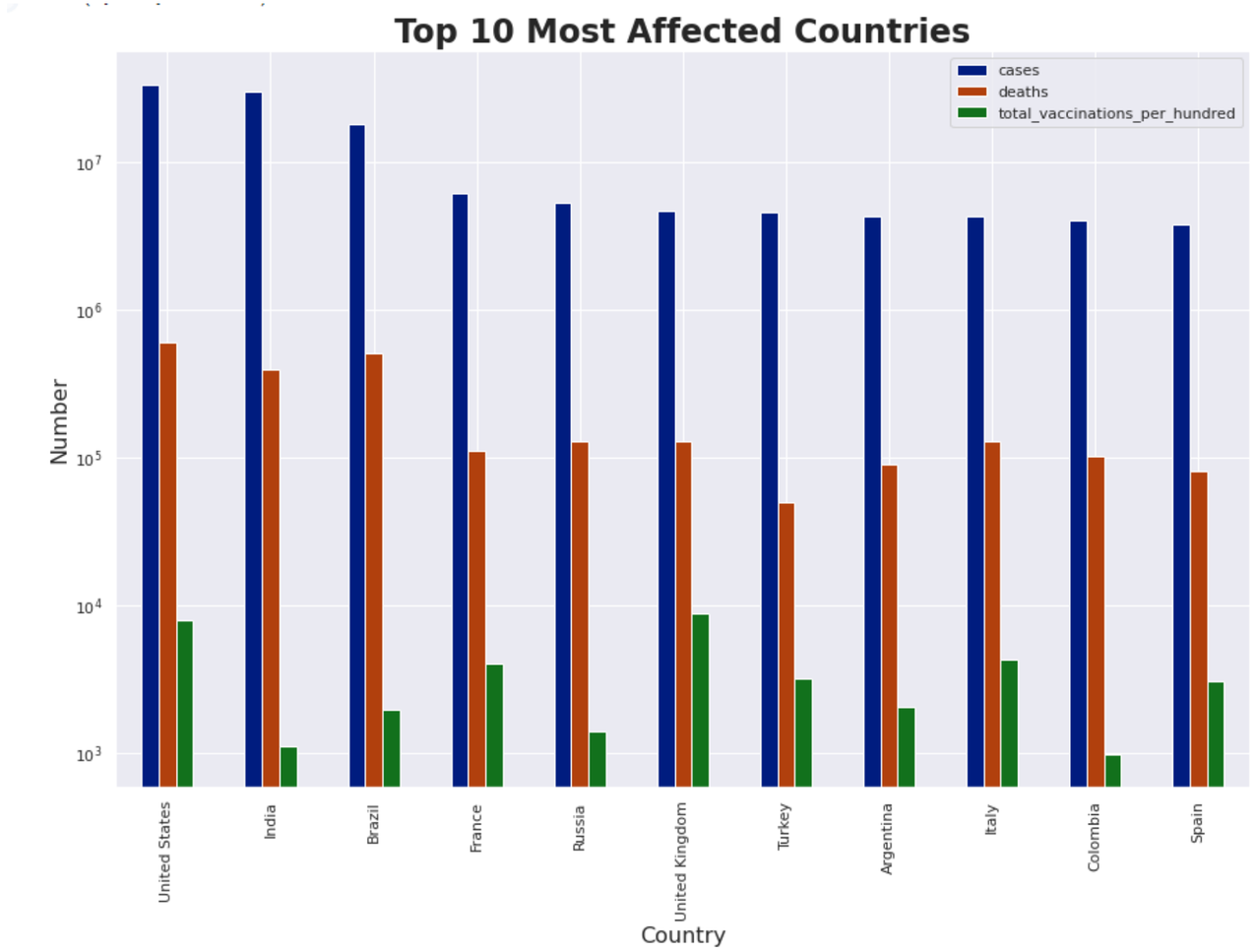

```
#Generate the choropleth map using gdf plot function on the Total Vaccinations

columnfig = june_people_vaccinated.plot(column='total_vaccinations',cmap='Wistia',figsize=(15,10), legend = True,
                                legend_kwds={'label': "Total Vaccinations",'orientation': "vertical", 'shrink': 0.55})

#removing axis ticks
plt.axis('off')

#Add the title
plt.title("Total Vaccinations in the World",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```


```
map_europe = europe.plot(column='total_vaccinations',cmap='Wistia',figsize=(15,10), legend = True,
                                legend_kwds={'label': "Total Vaccination",'orientation': "vertical", 'shrink': 0.55})

#removing axis ticks
plt.axis('off')

#Add the title
plt.title("Total Vaccinations in Europe",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
plt.show()
```

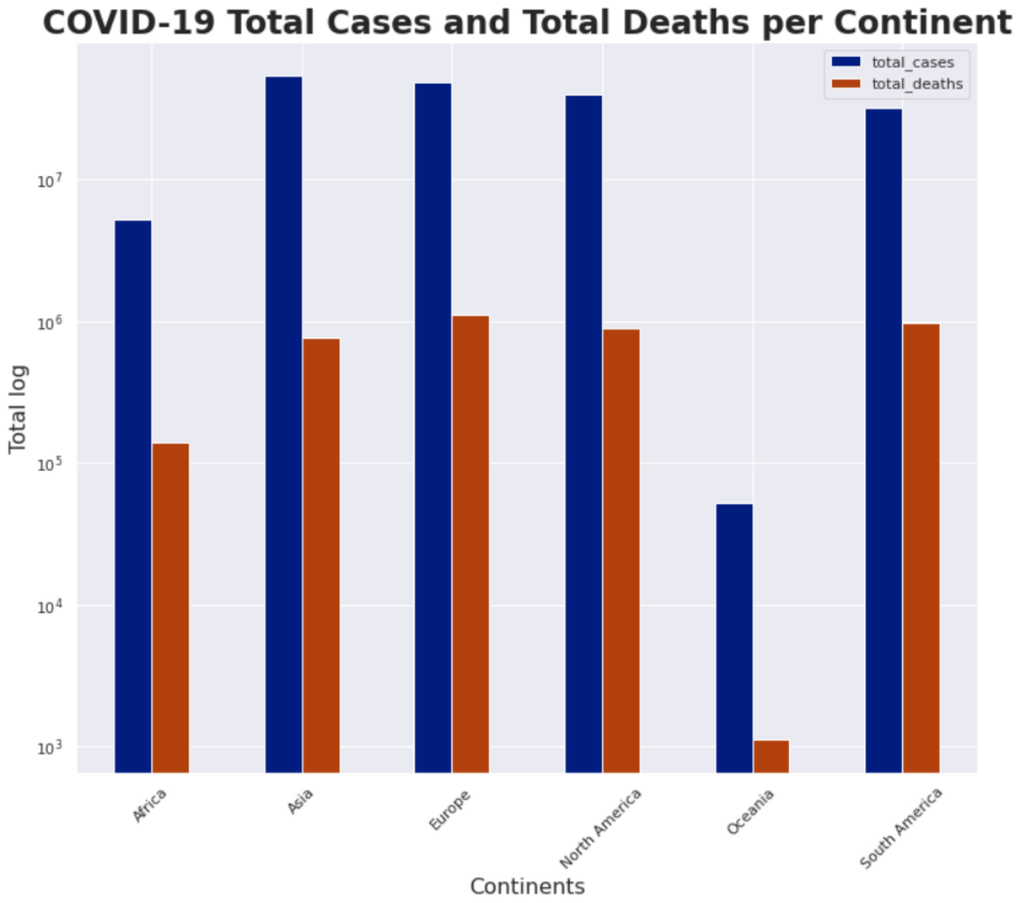

Total Vaccinations in Africa
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Total Vaccination
Total Vaccinations in Asia
1e9
1.0
0.8
0.6
0.4
0.2
0.0
Total Vaccination


Total Vaccinations in South America
1e7
8
6
4
2
0
Total Vaccinations

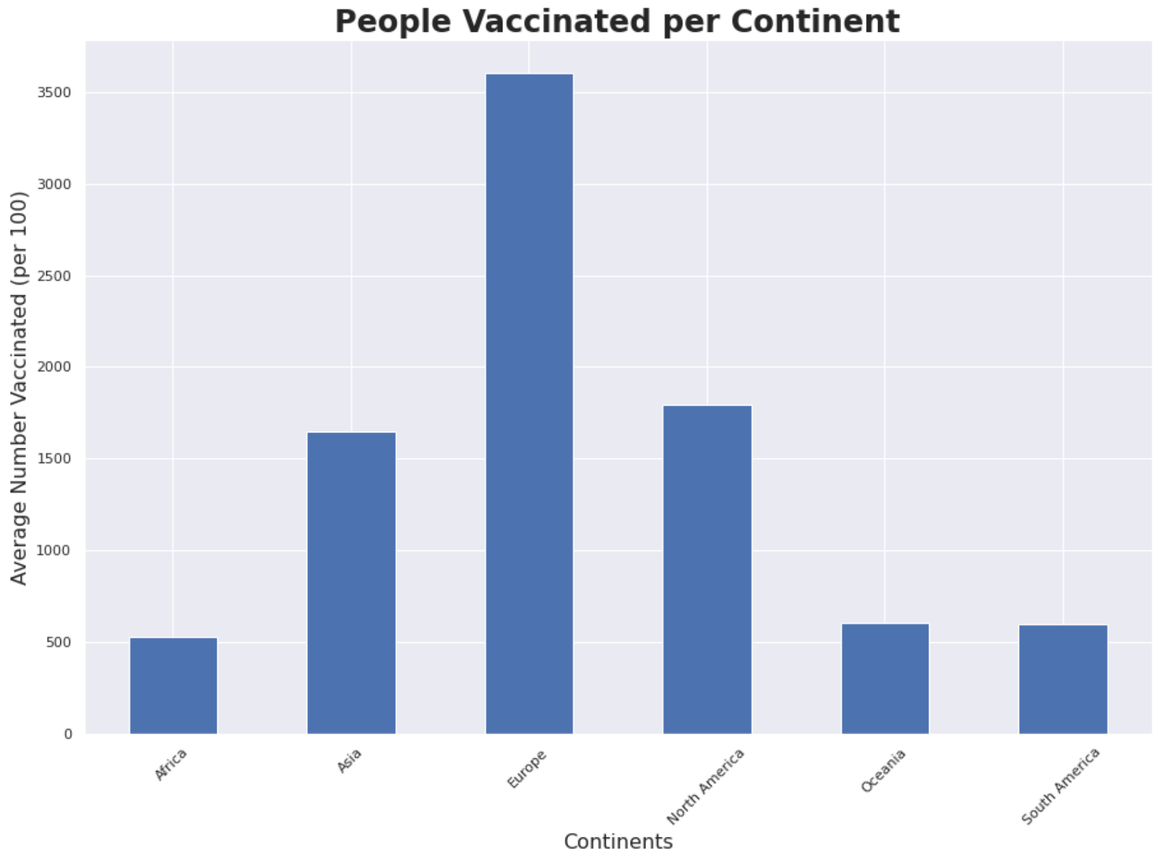

Total Vaccinations in North America
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Vaccination




```
ax_ct=country_df.iloc[:11][['cases','deaths','total_vaccinations_per_hundred']].plot(kind='bar',figsize=(12,10),logy=True)

ax_ct.set_title("Top 10 Most Affected Countries",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_ct.set_xlabel('Country',fontsize = 16, )
ax_ct.set_ylabel('Number',fontsize = 16)
```

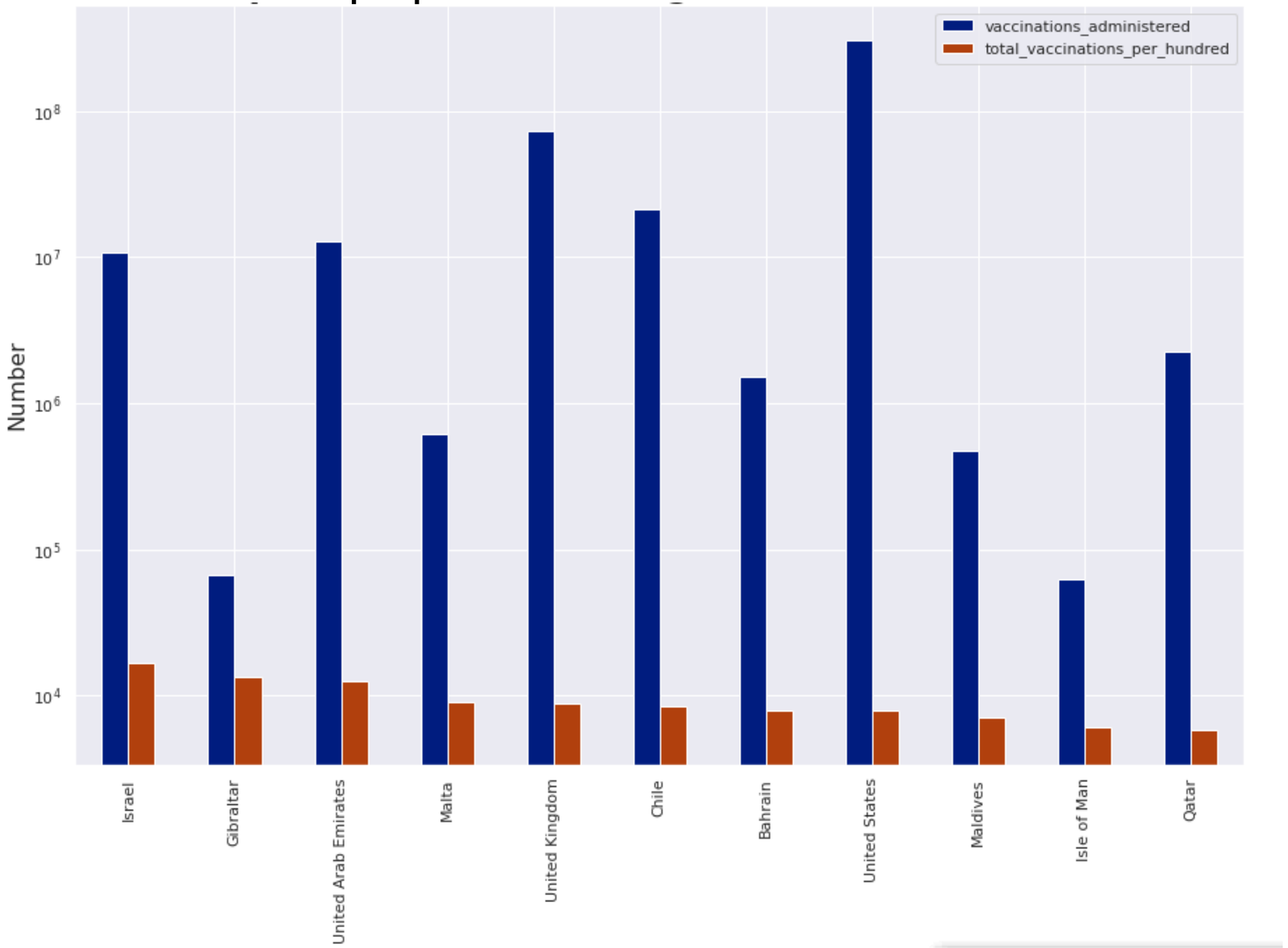



```
ax_cont=total_continents_df[['total_cases','total_deaths']].plot(kind='bar',figsize=(12,10),logy=True)
ax_cont.set_title("COVID-19 Total Cases and Total Deaths per Continent",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_cont.set_xlabel('Continents',fontsize = 16, )
ax_cont.set_ylabel('Total log',fontsize = 16)
ax_cont.set_xticklabels(total_continents_df.index, rotation=45 );
```

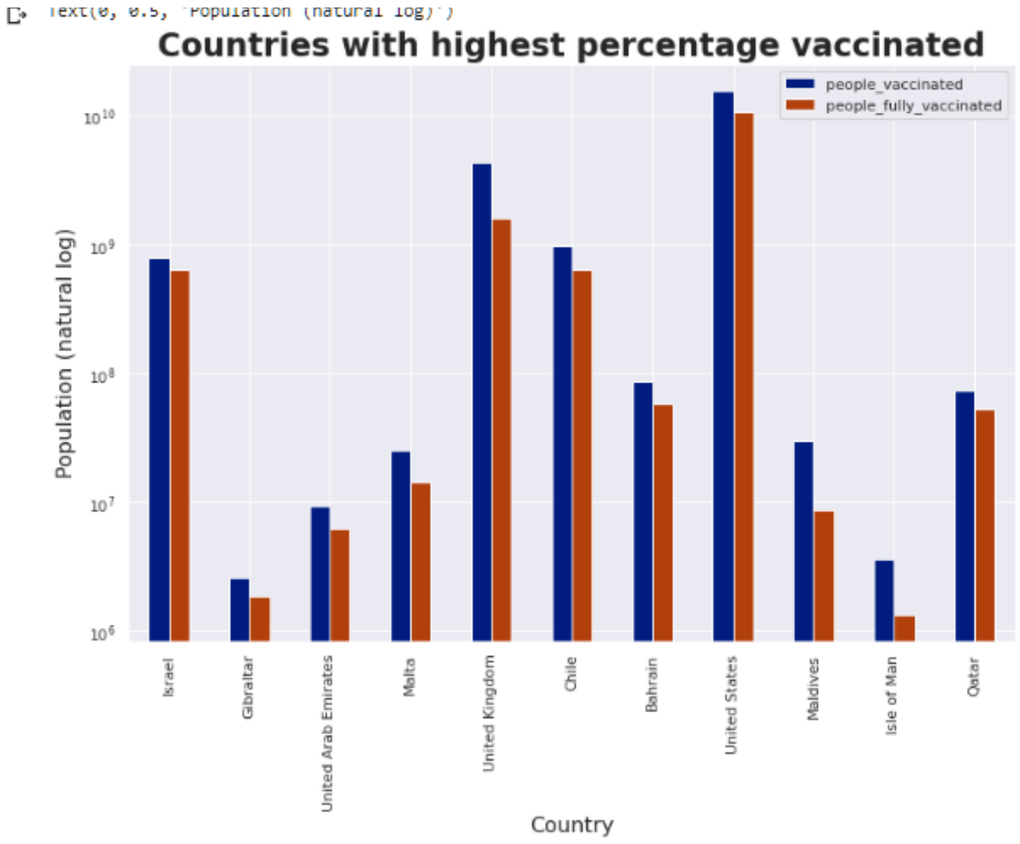


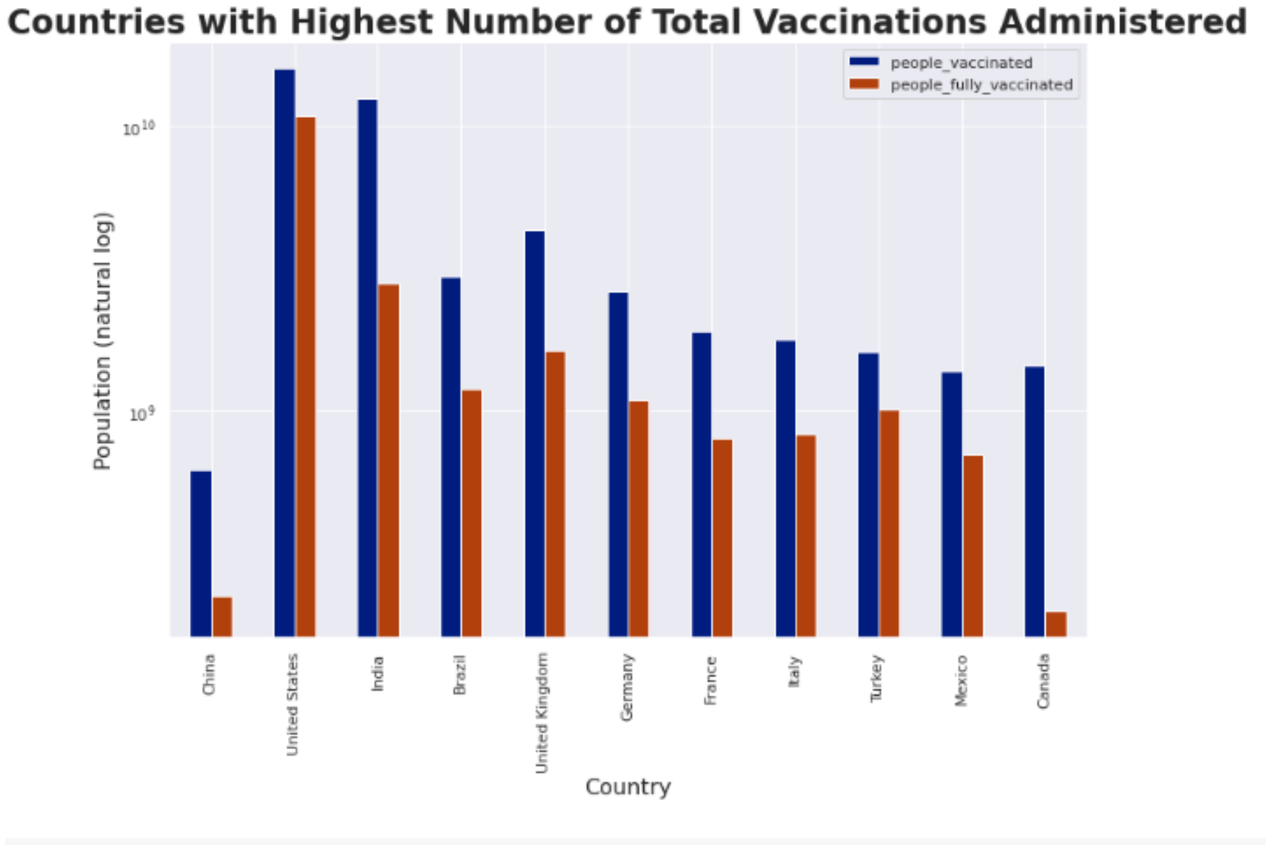

```
ax_cont=cont_vax_df['total_vaccinations_per_hundred'].plot(kind='bar',figsize=(15,10),logy=False)
ax_cont.set_title("People Vaccinated per Continent",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_cont.set_xlabel('Continents',fontsize = 16, )
ax_cont.set_ylabel('Average Number Vaccinated (per 100)',fontsize = 16)
ax_cont.set_xticklabels(total_continents_df.index, rotation=45 );
```

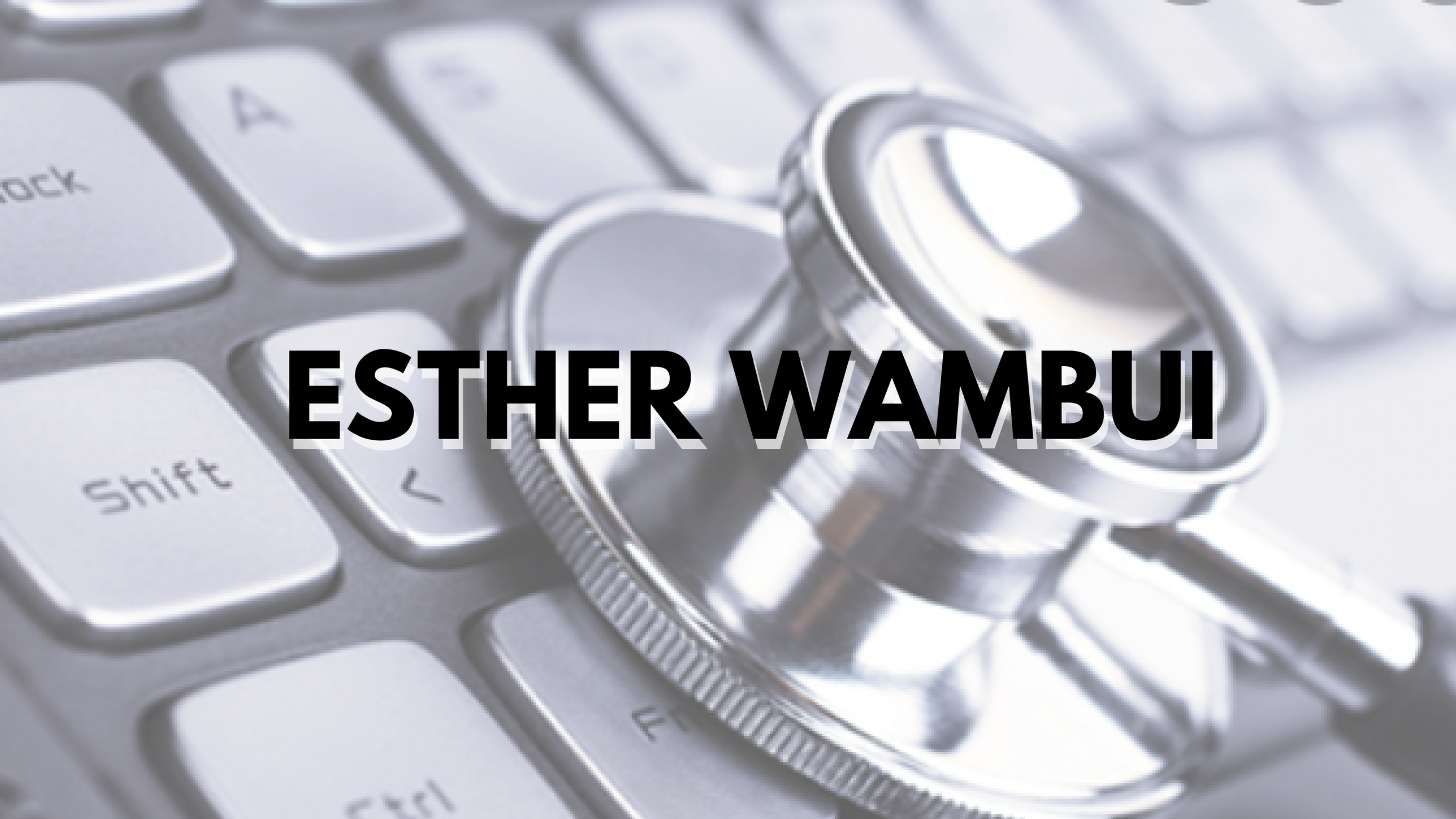
Top 10 Countries with the Highest % of population vaccinated



We compared people who received one vaccination and those who are fully vaccinated






ESTHER WAMBUI

Visualising COVID-19 as a pandemic

Let's Define some terms

A **pandemic** is an epidemic that spreads across international borders.

An **epidemic** is an outbreak of disease that spreads quickly and affects many individuals at the same time.

According to the World Health Organization (WHO), a pandemic can start when the following conditions have been met:

- A new virus subtype emerges.
- It infects humans and causes serious illness.
- It spreads easily and sustainably among humans.

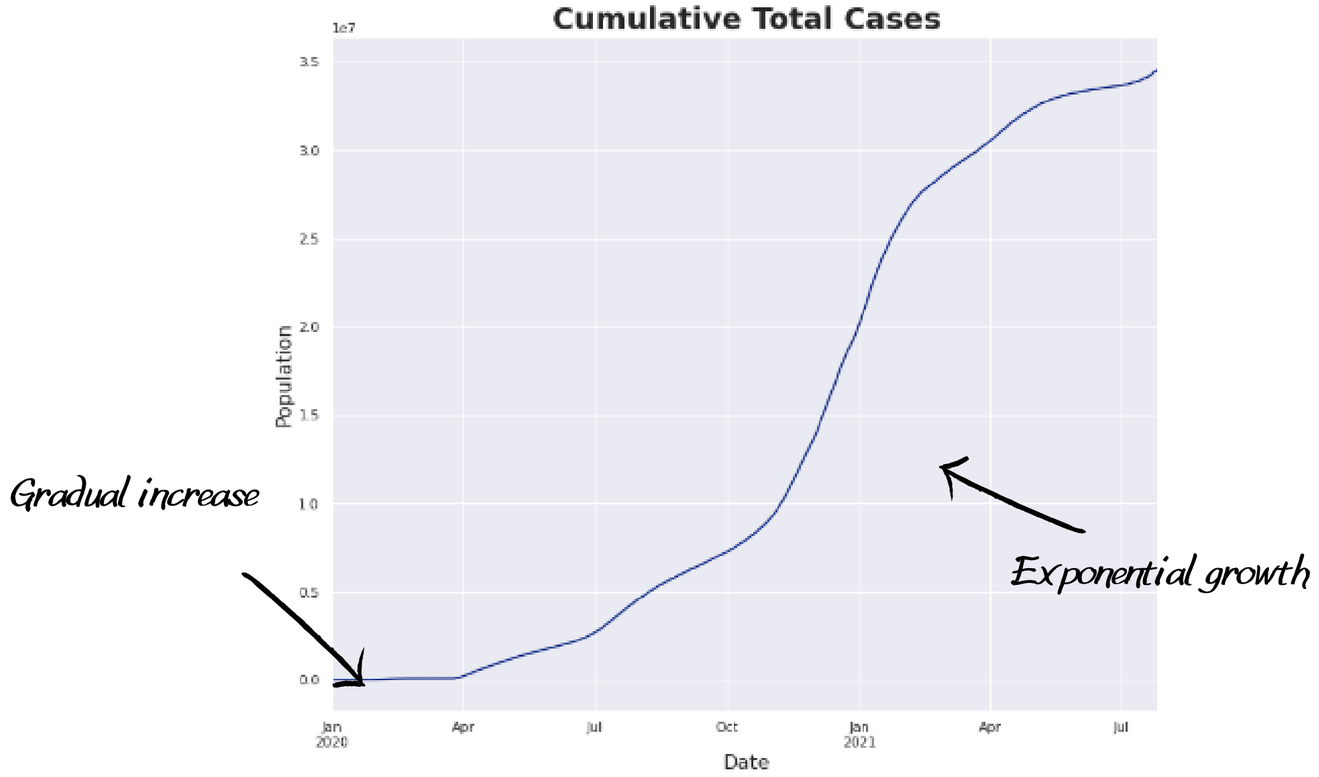
The Global COVID-19 Logistic Curve

This is a sigmoid (an S-shaped) curve that illustrates the growth of the number of people infected by the virus.

```
# Group values in the dataframe by date
grouped_dates = df.groupby('date')
grouped_dates
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fb4ef3409d0>
```

```
# Create a dataframe with the maximun values for each date
grouped_dates_max = grouped_dates.max()
# Create a dataframe with the sum of values for each date
grouped_dates_sum = grouped_dates.sum()
```

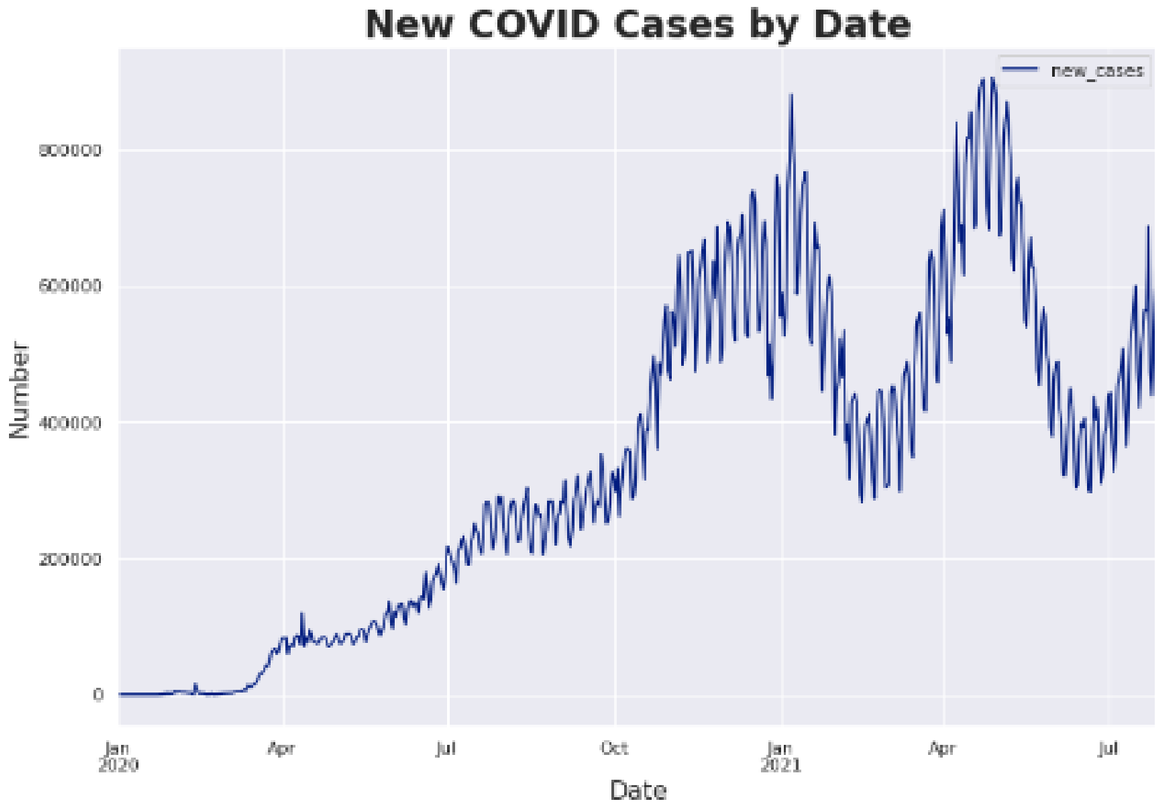



```
# Plot the total cases column
ax_dt=grouped_dates_max['total_cases'].plot(kind='line',figsize=(12,10))

ax_dt.set_title("Cumulative Total Cases",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_dt.set_xlabel('Date',fontsize = 16, )
ax_dt.set_ylabel('Population',fontsize = 16)
```

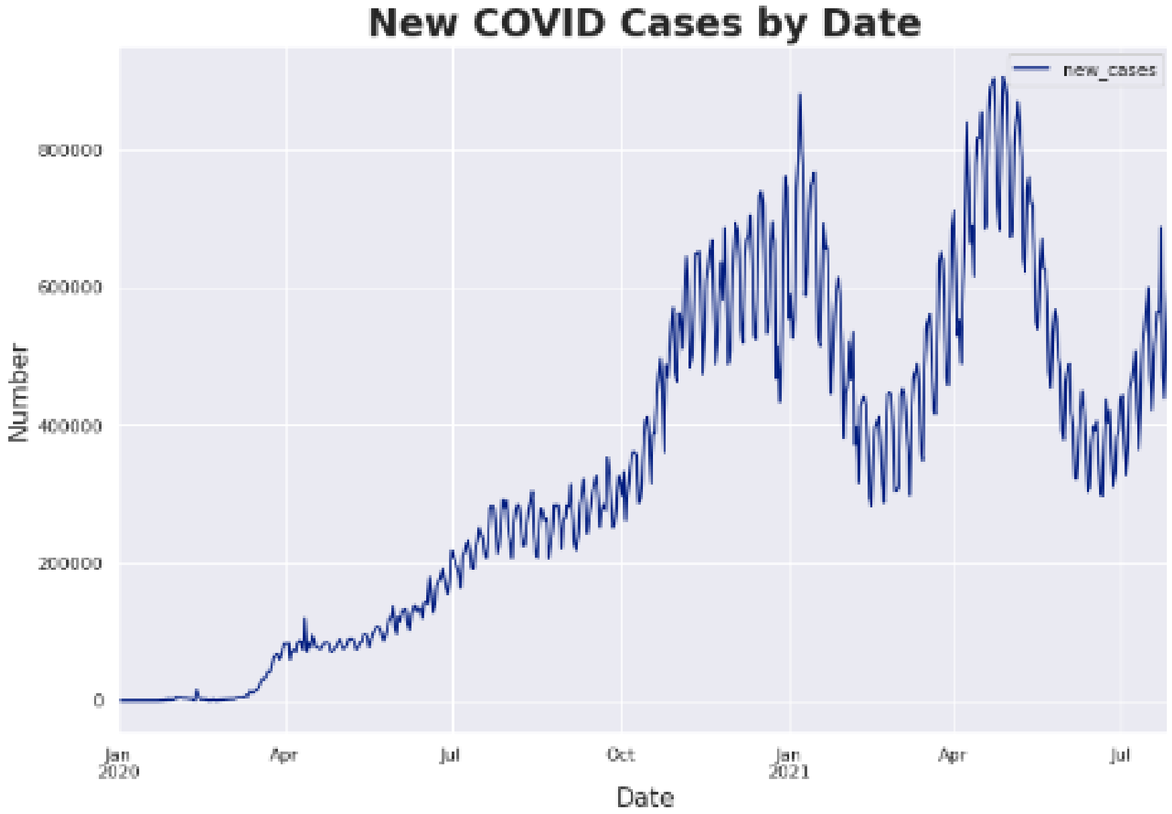
The Global COVID-19 Epidemic Curve

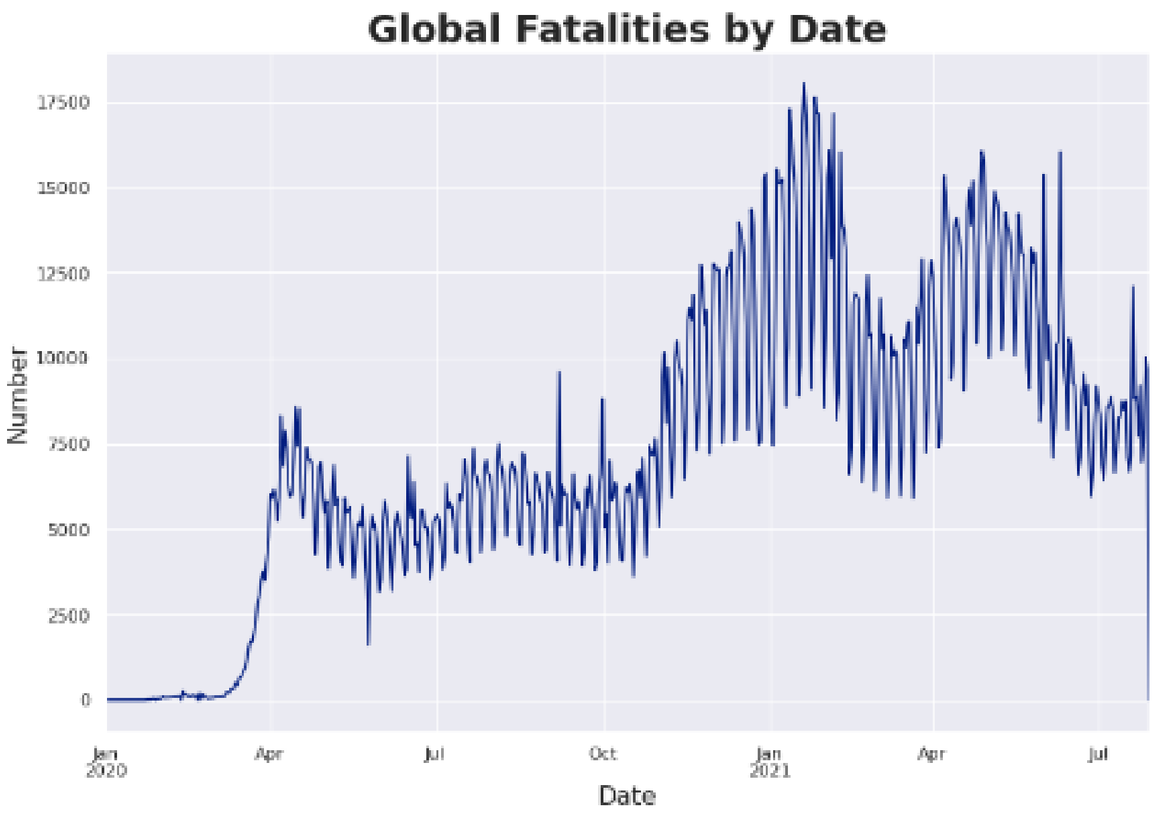
This curve shows the frequency of new cases over time (infection rate) based on the date of onset of disease.

The variance of data points in relation to time can give a clue on the incubation period of the disease.



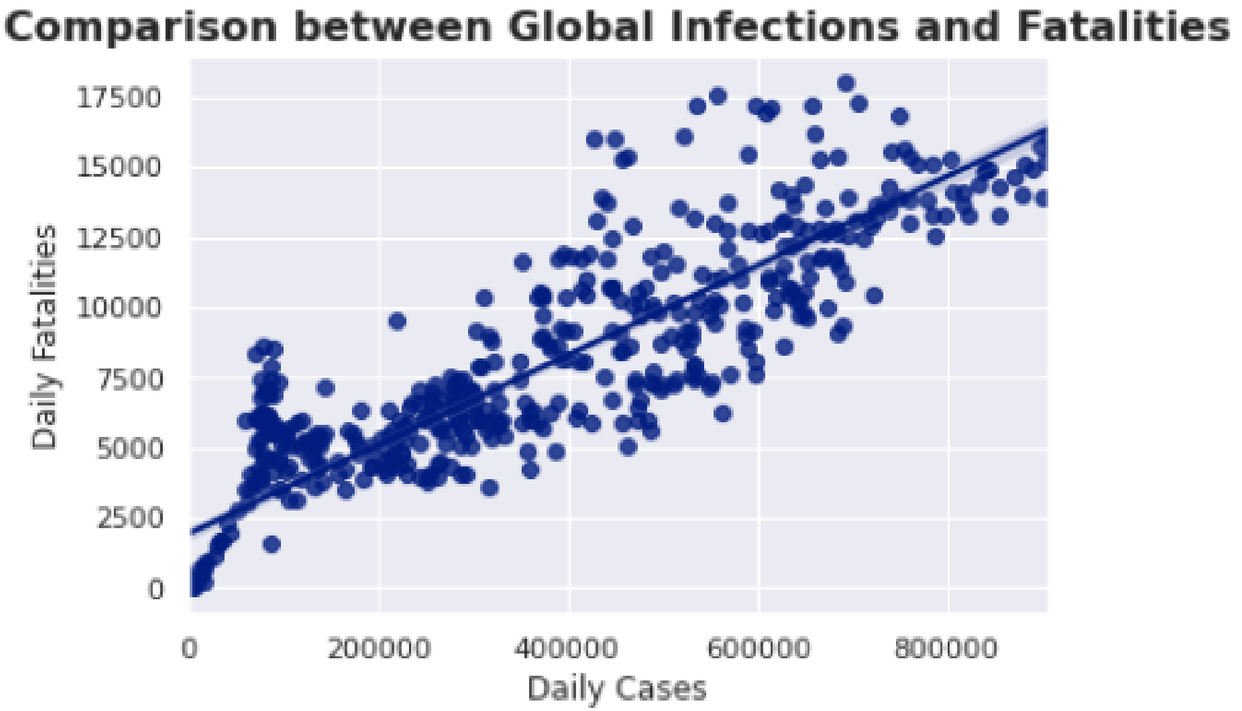

```
ax_dt=grouped_dates_sum[['new_cases']].plot(kind='line',figsize=(12,8))
ax_dt.set_title("New COVID Cases by Date",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_dt.set_xlabel('Date',fontsize = 16, )
ax_dt.set_ylabel('Number',fontsize = 16)
```





Notice that the shape of the curves representing the infection rate and the death rate are almost similar.

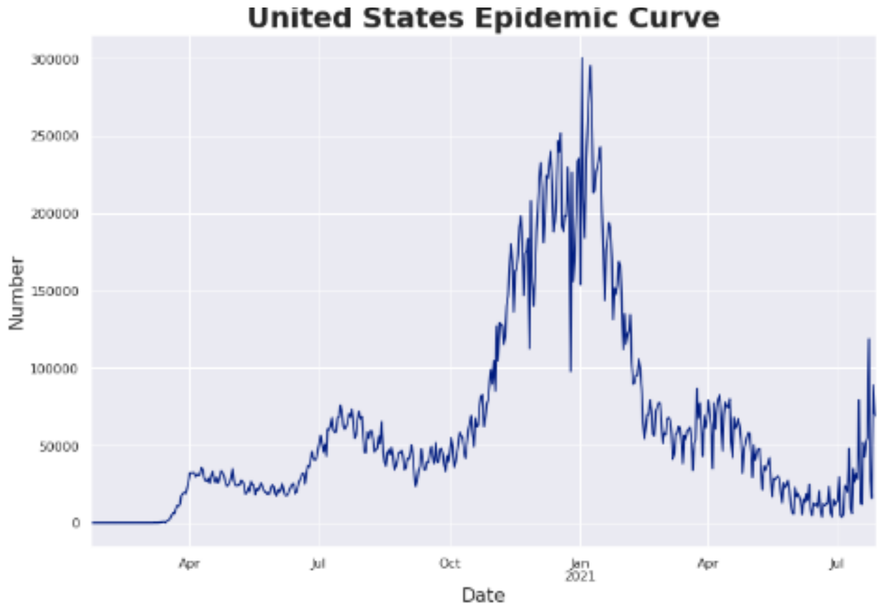
This insinuates a correlation between the COVID -19 cases and fatalities.

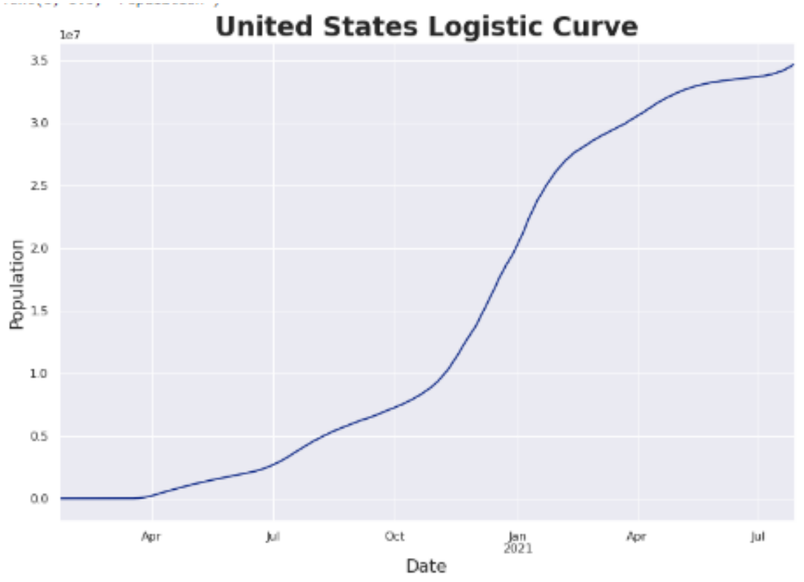


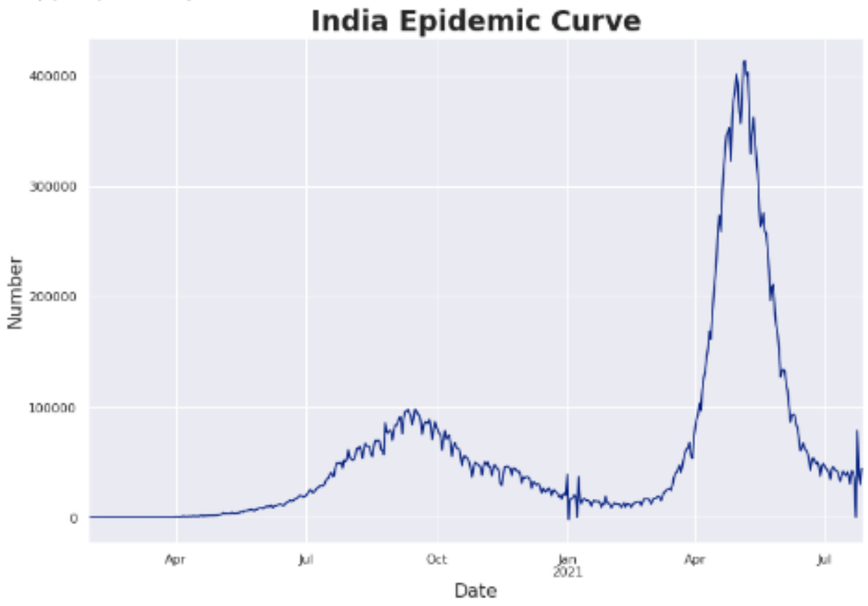
This high positive correlation of 0.89 indicates that COVID-19 is a fatal disease.

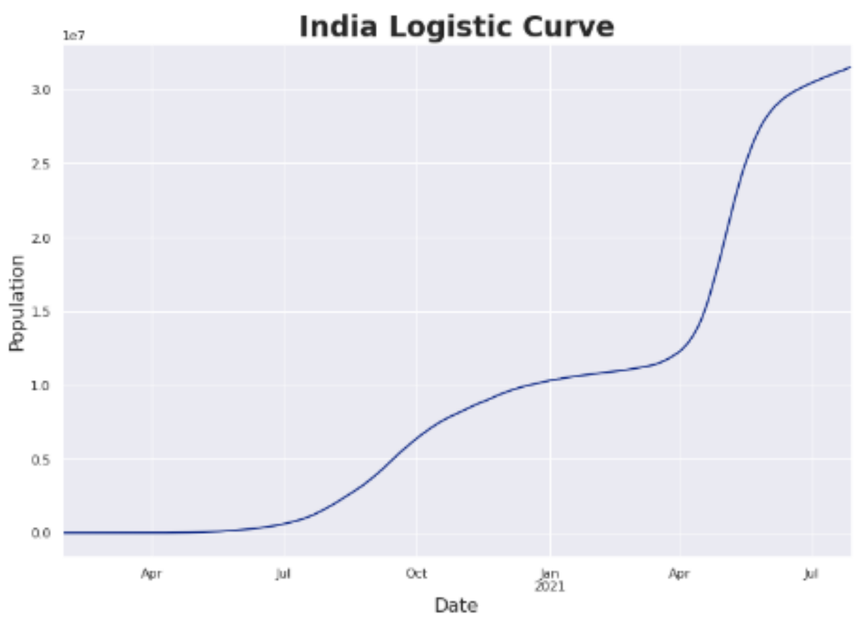
```
ax_c=sns.regplot(x='new_cases', y='new_deaths', data= grouped_dates_sum)
ax_c.set_title("Comparison between Global Infections and Fatalities",fontdict= { 'fontsize': 16, 'fontweight':'bold'})
ax_c.set_xlabel('Daily Cases',fontsize = 12 )
ax_c.set_ylabel('Daily Fatalities',fontsize = 12)
```

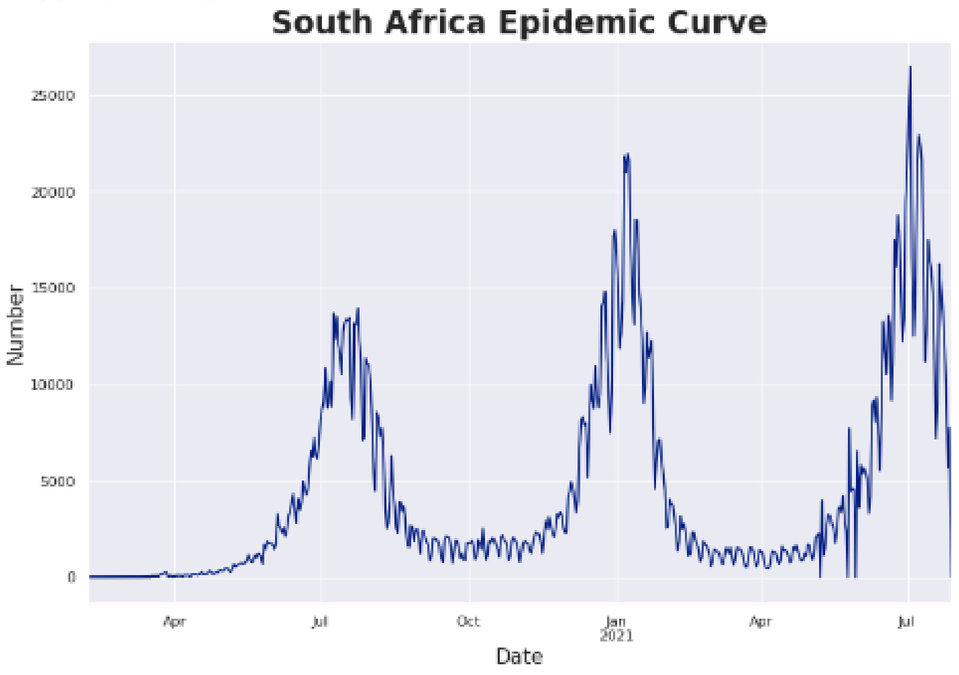
Let's take a look at some countries

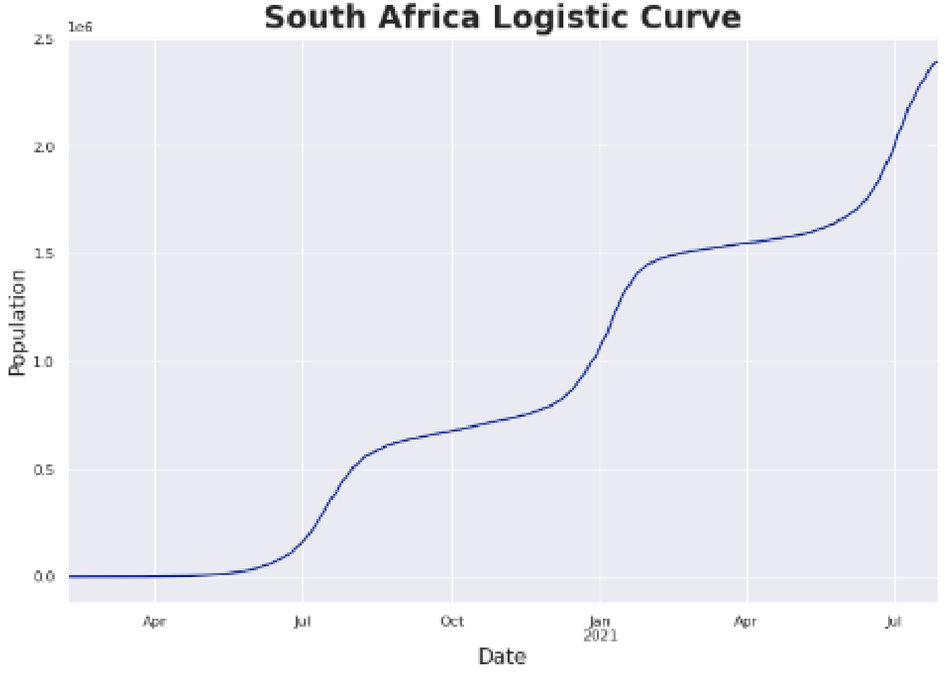


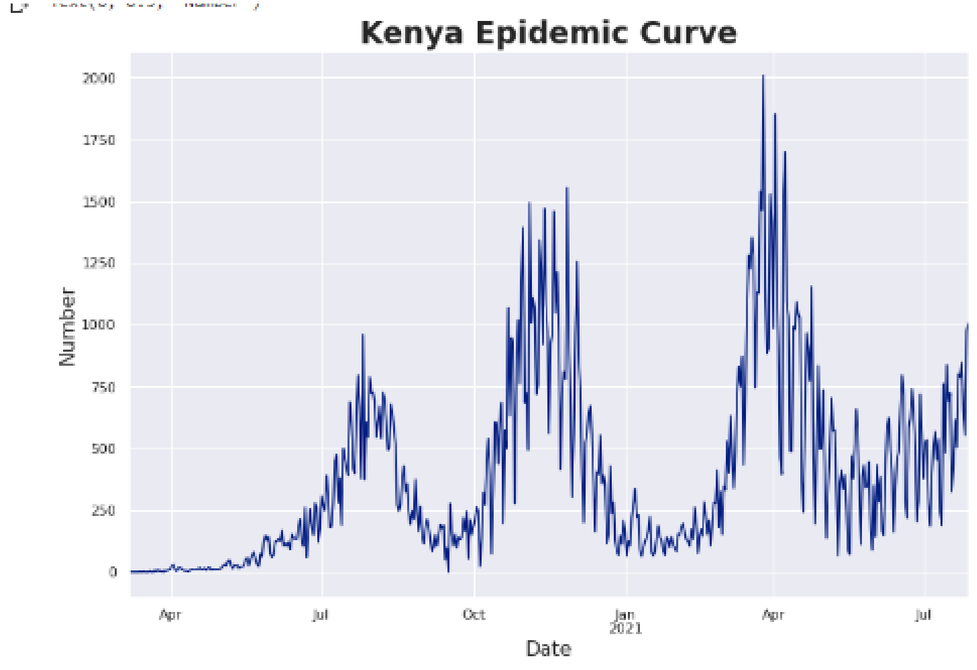


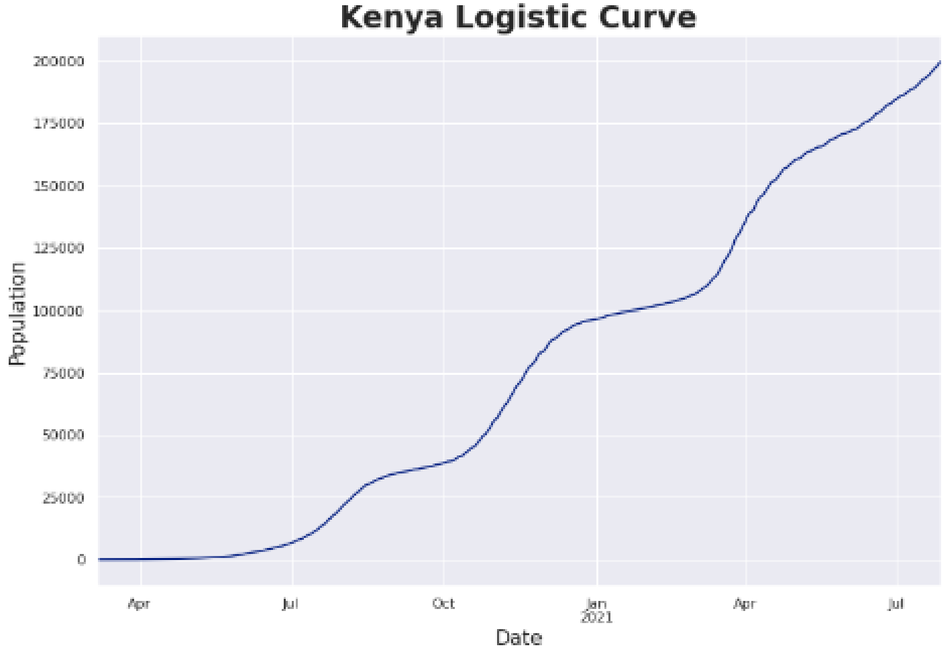





South Africa Epidemic Curve
Number
Date
Apr
Jul
Oct
Jan
2021
Apr
Jul
0
5000
10000
15000
20000
25000


South Africa Logistic Curve
1e6
Population
Date
Apr
Jul
Oct
Jan
2021
Apr
Jul
0.0
0.5
1.0
1.5
2.0
2.5

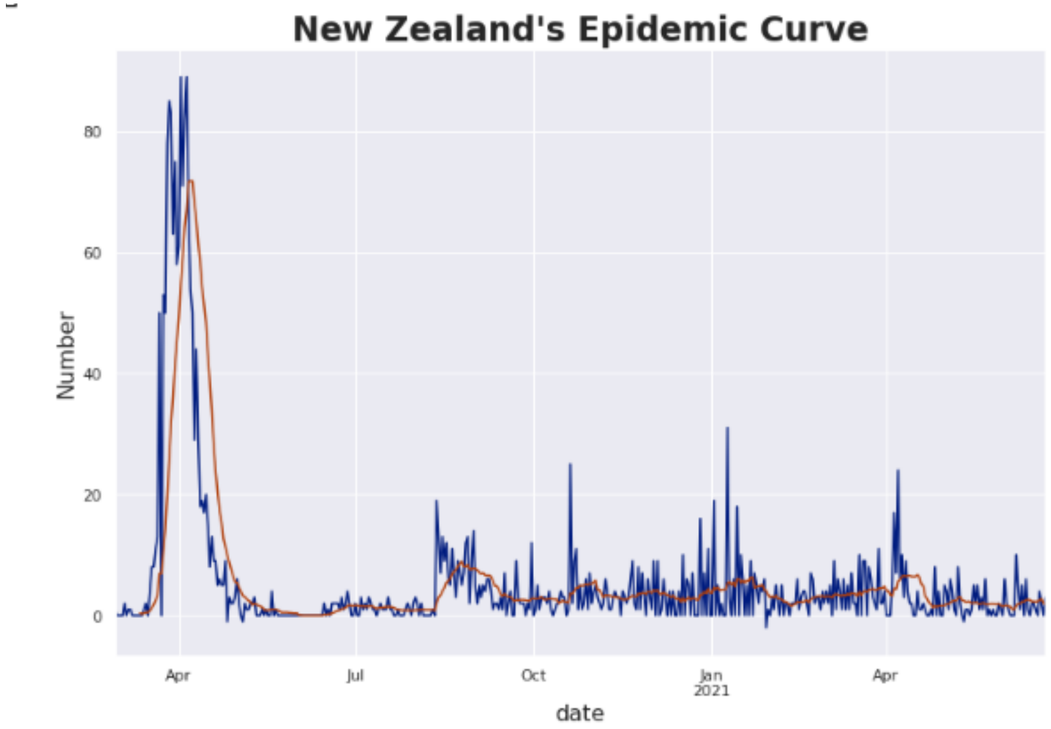

Kenya Epidemic Curve
Number
Date
Apr
Jul
Oct
Jan
2021
Apr
Jul
0
250
500
750
1000
1250
1500
1750
2000

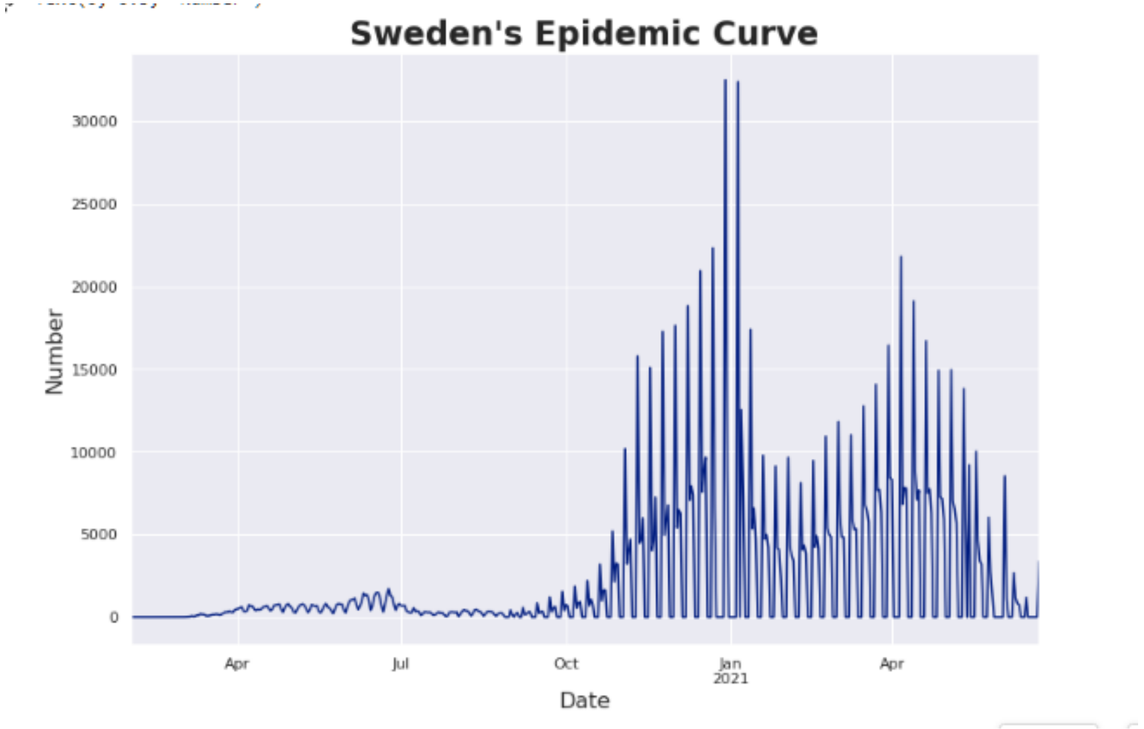

Kenya Logistic Curve
Population
Date
Apr
Jul
Oct
Jan
2021
Apr
Jul
0
25000
50000
75000
100000
125000
150000
175000
200000

Does lockdown work?

To answer this question, we compared a country that went into lockdown and one that did not.

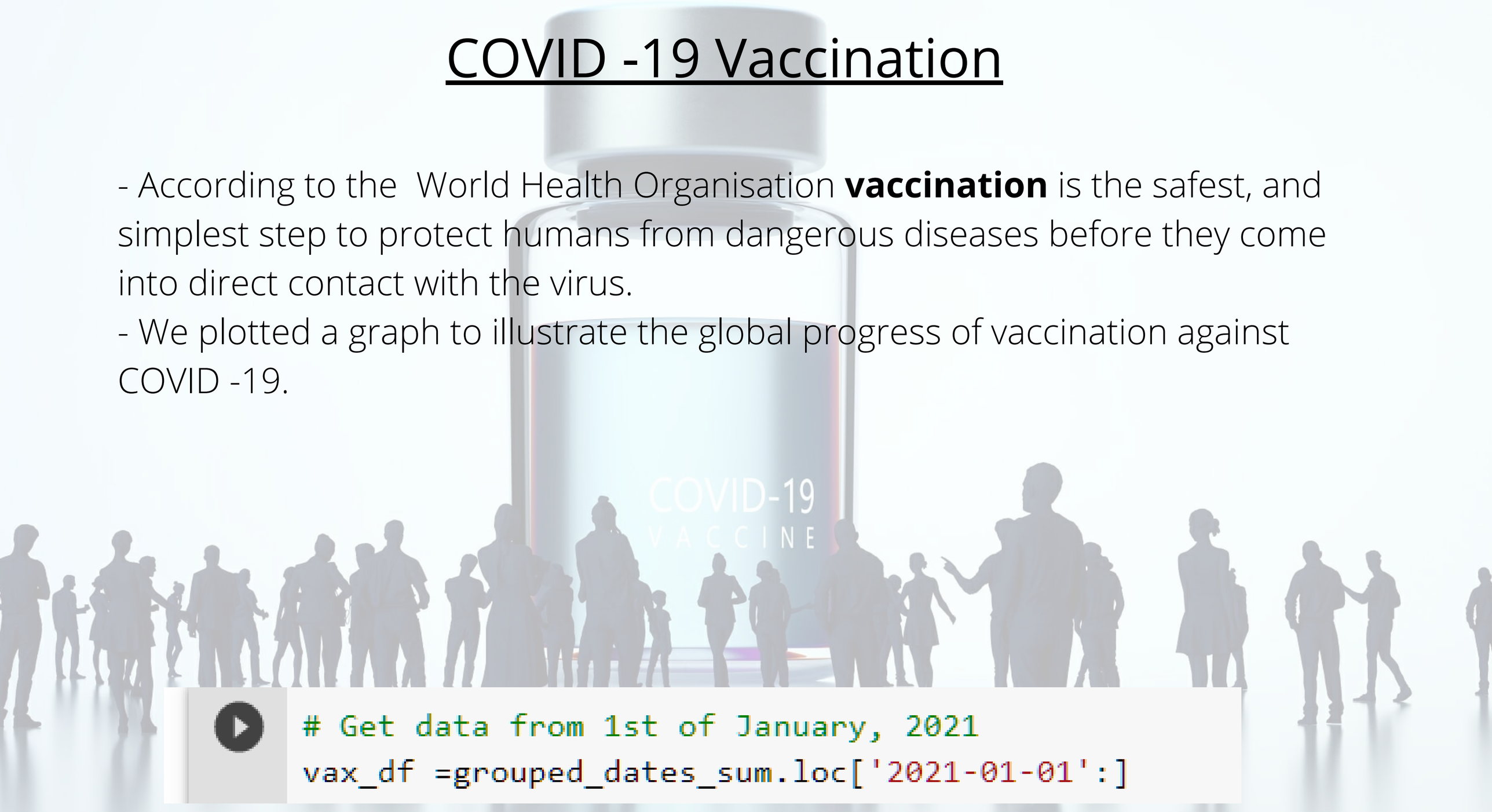
New Zealand went into lockdown during the early onset of the pandemic whereas ***Sweden*** has never locked down.



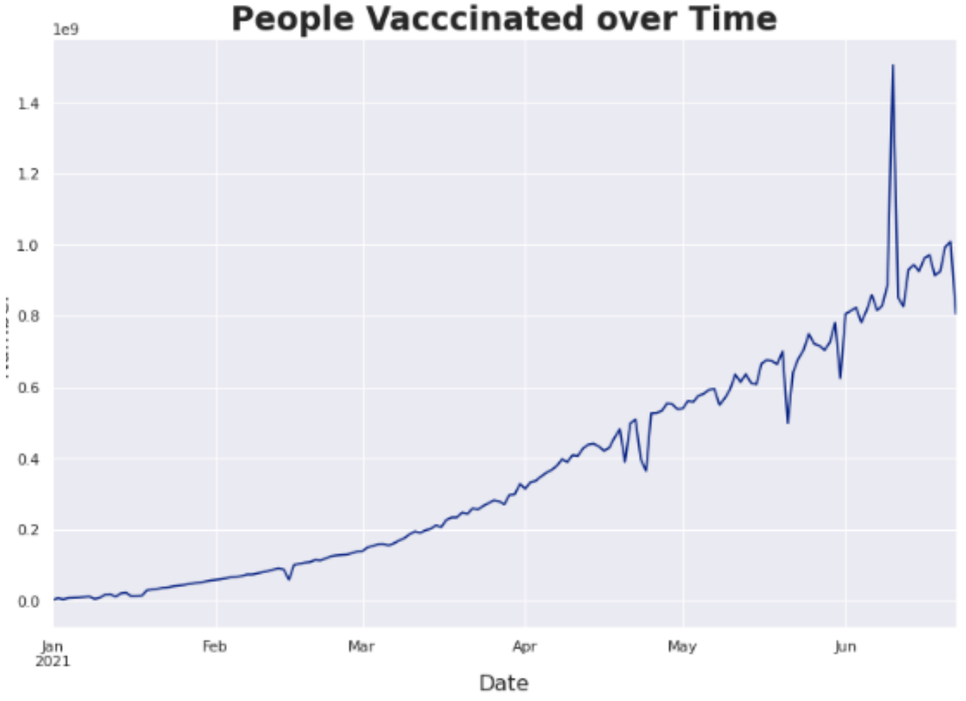


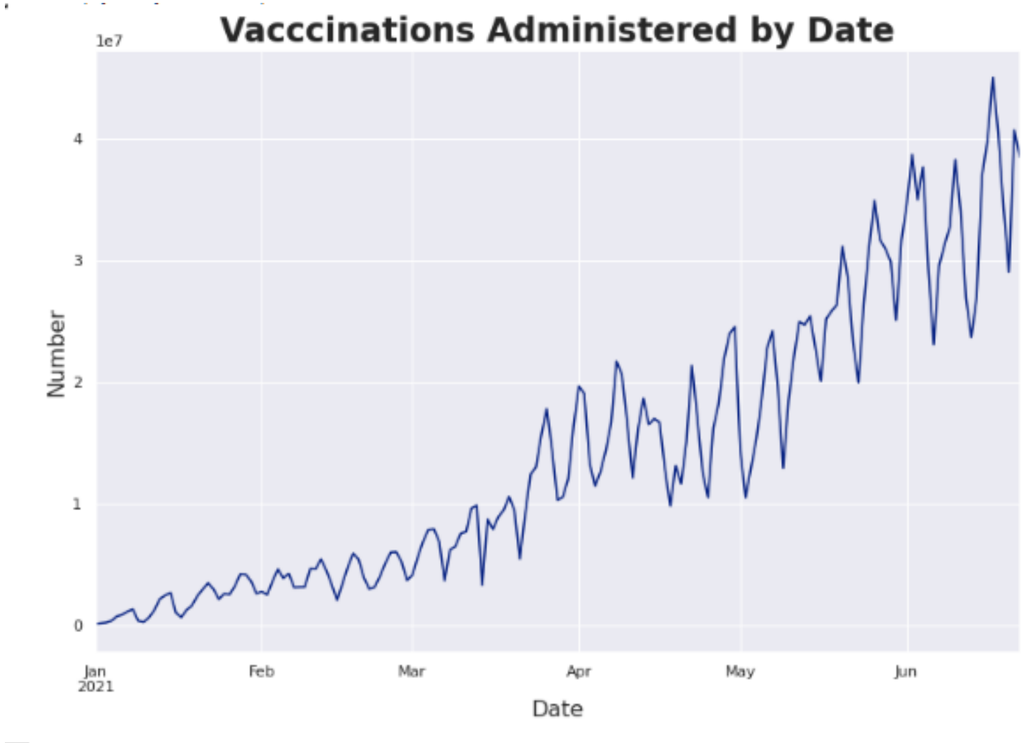
COVID -19 Vaccination

- According to the World Health Organisation **vaccination** is the safest, and simplest step to protect humans from dangerous diseases before they come into direct contact with the virus.
- We plotted a graph to illustrate the global progress of vaccination against COVID -19.




```
# Get data from 1st of January, 2021
vax_df =grouped_dates_sum.loc['2021-01-01':]
```






```
ax_vx=vax_df['people_vaccinated'].plot(kind='line',figsize=(12,8))

ax_vx.set_title("People Vacccinated over Time",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_vx.set_xlabel('Date',fontsize = 16, )
ax_vx.set_ylabel('Number',fontsize = 16)
```

```
_vx=vax_df['new_vaccinations'].plot(kind='line',figsize=(12,8))

_vx.set_title("Vacccinations Administered by Date",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
_vx.set_xlabel('Date',fontsize = 16, )
_vx.set_ylabel('Number',fontsize = 16)
```

Prediction of daily Vaccination in the United states.

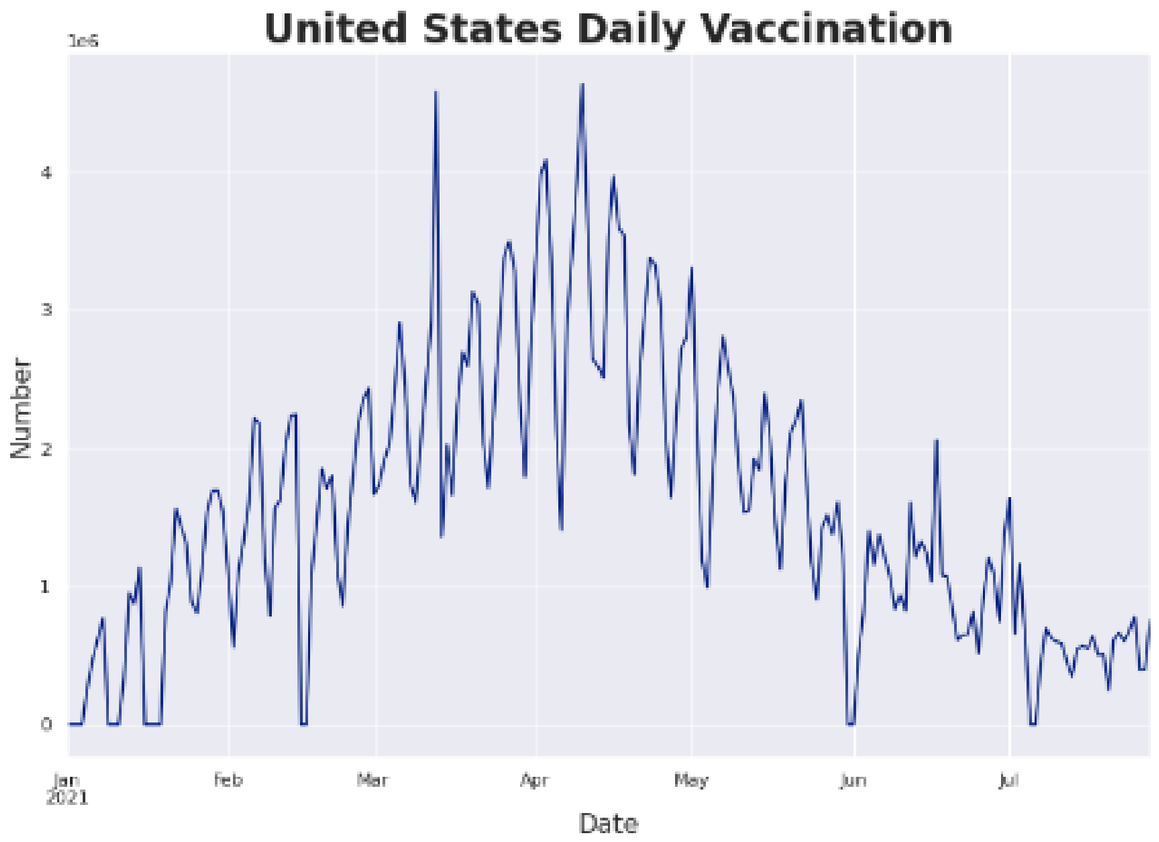
Our aim is to forecast the number of new vaccinations to be administered the following day.

This will useful in:

- Human resource allocation in health facilities.
- Procurement in health facilities.
- Operations management in manufacturing companies
- Formulation of health promotion activities.

Exploring our Data

date	vaccinations_administered
2021-01-05	273209.0
2021-01-06	470328.0
2021-01-07	612621.0
2021-01-08	768813.0
2021-01-09	0.0



The graph above is a typical example of ***time series data.***

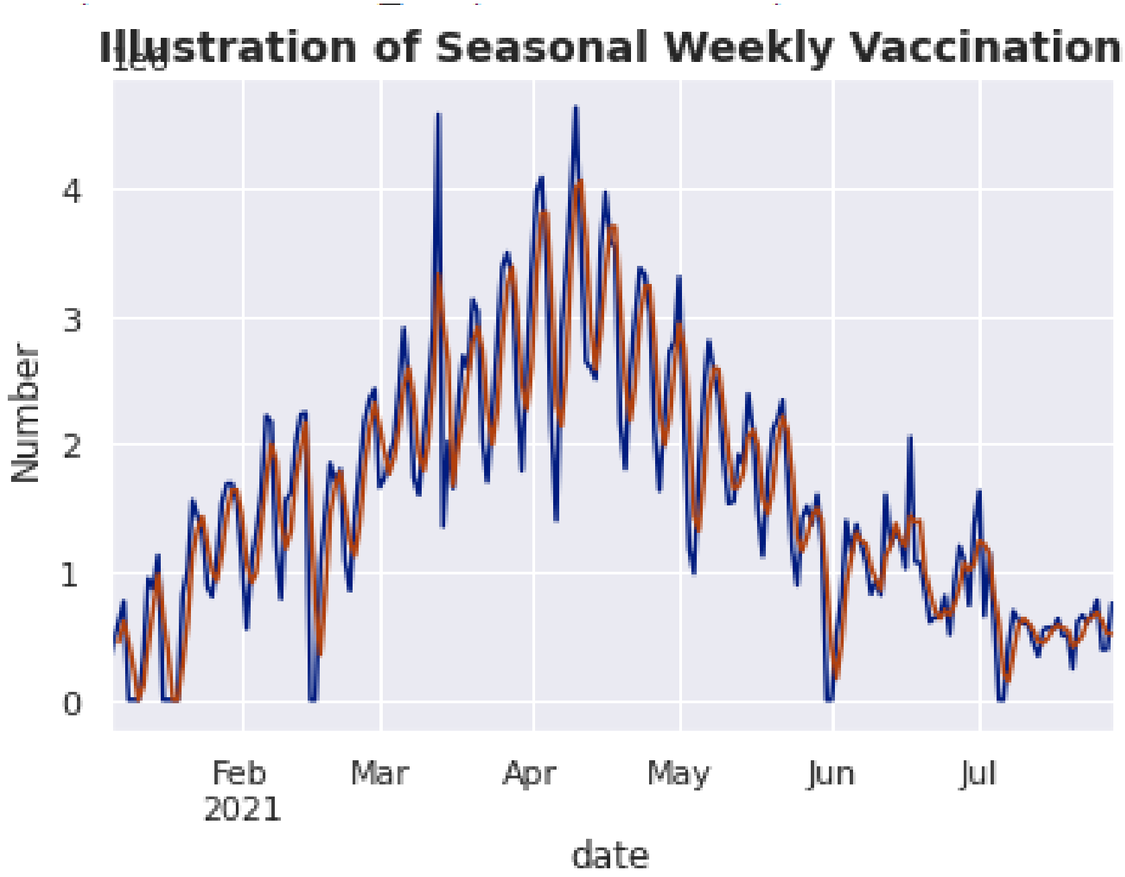
```
ax_dt=usa_df.loc['2021-01-01':]['new_vaccinations'].plot(kind='line',figsize=(12,8),logy=False)

ax_dt.set_title("United States Daily Vaccination",fontdict= { 'fontsize': 24, 'fontweight':'bold'})
ax_dt.set_xlabel('Date',fontsize = 16, )
```

Understanding Time Series Data

Time series data is a collection of quantities such as patient vital signs, sales, stock prices or weather data, that are assembled over even intervals in time and ordered chronologically.

It is characterized by:

- Mean reverting data.
- Time trending data.
- Seasonality
- Structured btraks.




```
tion to plot for stationarity test
ationary_test_plot(series):
eries.plot()
et_title("United States Daily Vaccination",fontdict= { 'fontsize':
et_xlabel('Date',fontsize = 12, )
et_ylabel('Number',fontsize = 12)
tting the moving average
es.rolling(window=7).mean().plot()
tting the moving standard deviation.
es.rolling(window=7).std().plot()
```

From the above graph we observe that:

- There's an upward trend between January and May with peaks in mid-March and mid-April followed by a downward trend from May to June.
- The mean is not constant., making the series is non-stationary.
- There's a weekly seasonality.

The Prediction Process

Step 1:- Learning

Step 2:- Prediction

Testing Stationarity.

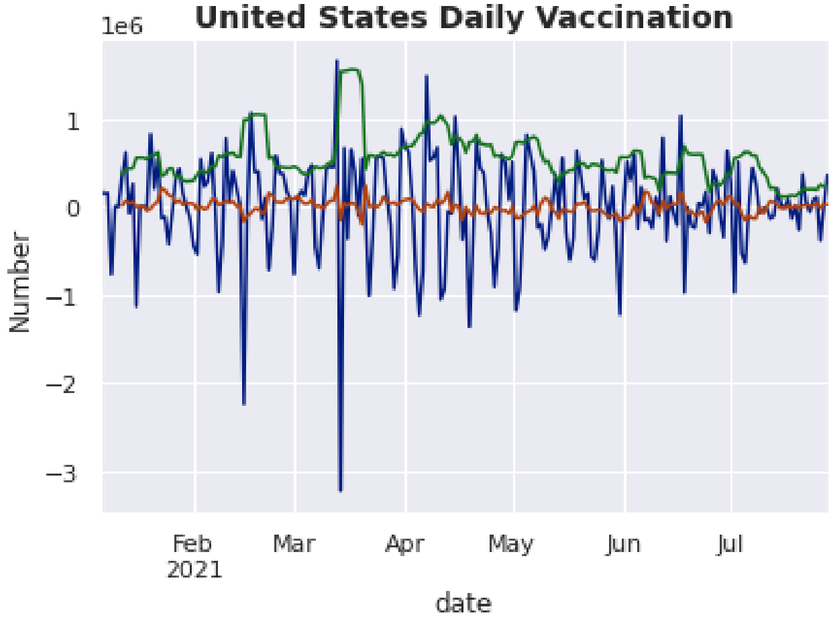
```
]  # Create afunction for the dickey fuller test.
   def adfuller_test(sales):
       result=adfuller(sales)
       labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Obser
       for value,label in zip(result,labels):
           print(label+' : '+str(value) )
       if result[1] <= 0.05:
           print("The series is stationary")
       else:
           print("The series is non-stationary ")
```

```
usa_vax['diff_vaccinations']=usa_vax['vaccinations_administered'].diff()
usa_vax.dropna(axis=0,inplace=True)
usa_vax.tail(3)
```

date	vaccinations_administered	diff_vaccinations
2021-07-26	393083.0	-385913.0
2021-07-27	395489.0	2406.0
2021-07-28	753984.0	358495.0

```
adfuller_test(usa_vax['diff_vaccinations'])
```

```
ADF Test Statistic : -3.008376842900737
p-value : 0.03411947947141374
#Lags Used : 15
Number of Observations Used : 188
The series is stationary
```



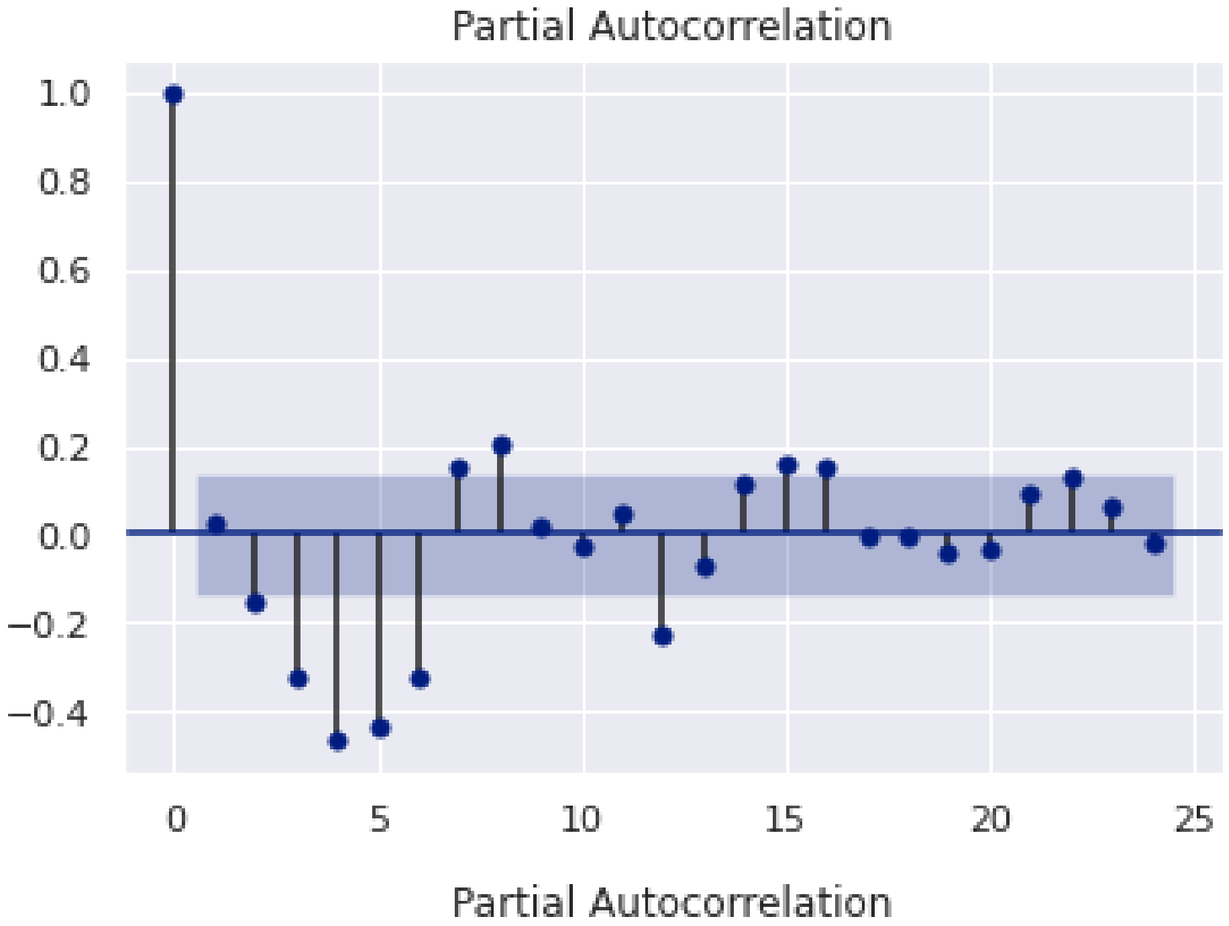
The autoregressive Value

Identification of an AR model is often best done with the PACF.

- For an AR model, the theoretical PACF “shuts off” past the order of the model.

- The phrase “shuts off” means that in theory the partial autocorrelations are equal to 0 beyond that point. Put another way, the number of non-zero partial autocorrelations gives the order of the AR model.

- By the “order of the model” we mean the most extreme lag of x that is used as a predictor.

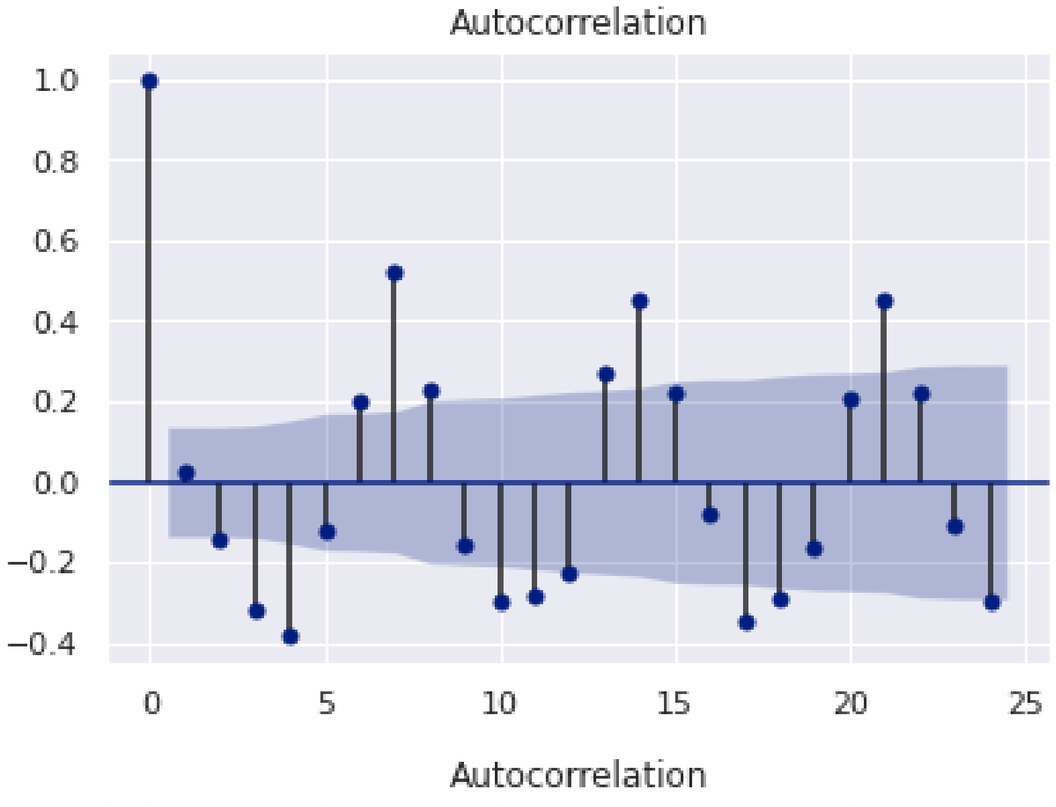

Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20
25
Partial Autocorrelation

The Moving Average (MA) Value

Identification of an MA model is often best done with the ACF rather than the PACF.

- For an MA model, the theoretical PACF does not shut off, but instead tapers toward 0 in some manner.

- A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20
25
Autocorrelation

Statistical Models

We use the dependent variable
to make predictions.

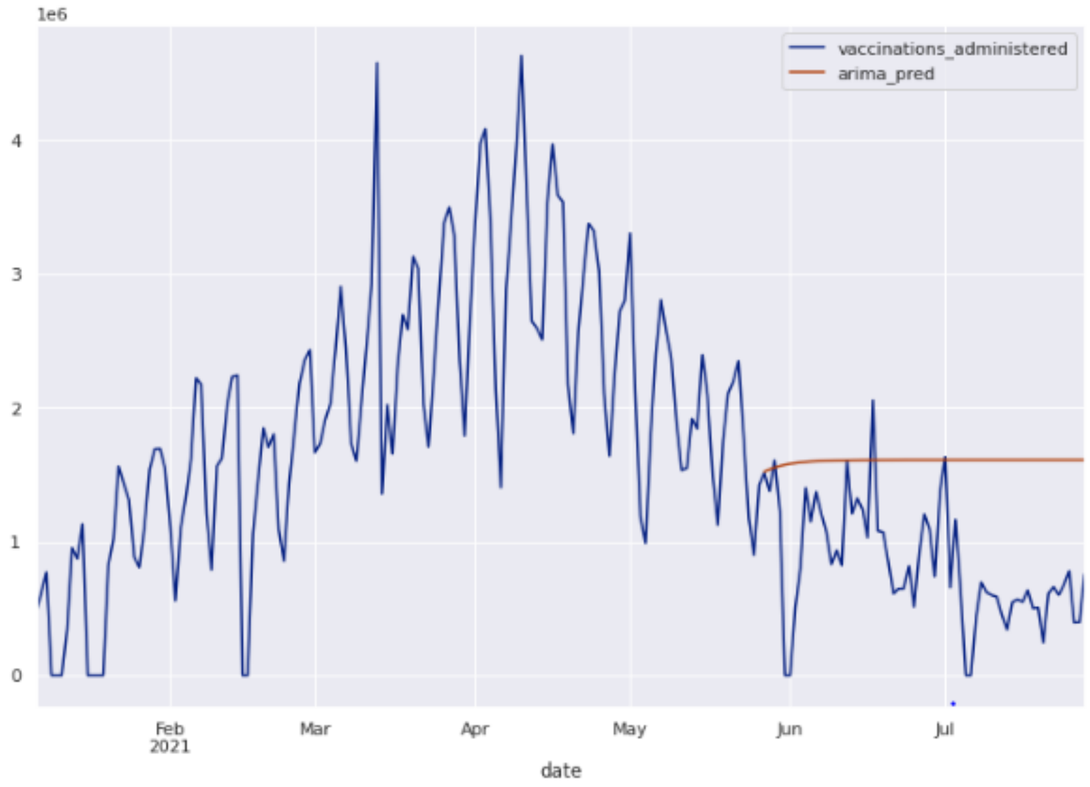
Requires the following parameters:

- Autoregression value (p)
- Order of differencing (d)
- Moving Average value (q)

ARIMA -

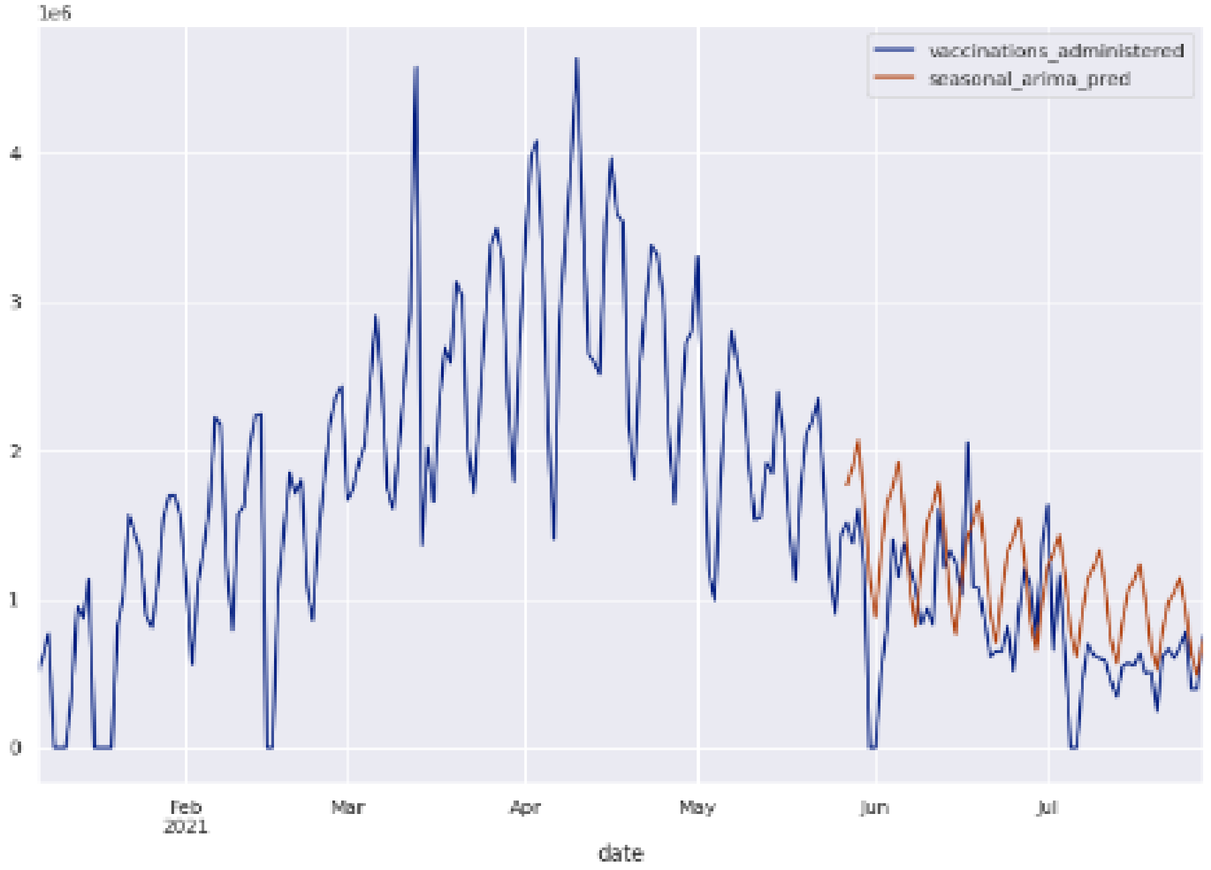
Autoregressive Integrated Moving Average

```
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(usa_vax['vaccinations_administered'], order=(1,0,1))
arima_model_fit = model.fit()

usa_vax['arima_pred']=arima_model_fit.predict(start=141,end=204,dynamic=True)
usa_vax[['vaccinations_administered','arima_pred']].plot(figsize=(12,8))
```



Seasonal ARIMA

```
import statsmodels.api as sm

model=sm.tsa.statespace.SARIMAX(usa_vax['vaccinations_administered'],order=(1, 0, 1),seasonal_order=(1,0,1,7))
sarima_model_fit=model.fit()

usa_vax['seasonal_arima_pred']=sarima_model_fit.predict(start=141,end=204,dynamic=True)
usa_vax[['vaccinations_administered','seasonal_arima_pred']].plot(figsize=(12,8))
```



Supervised Machine Learning Models

We use labelled data to make predictions.

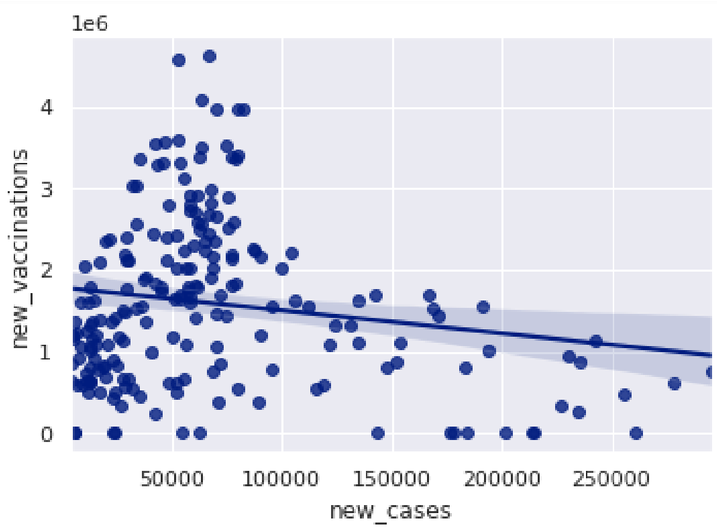
Requires the following parameters:

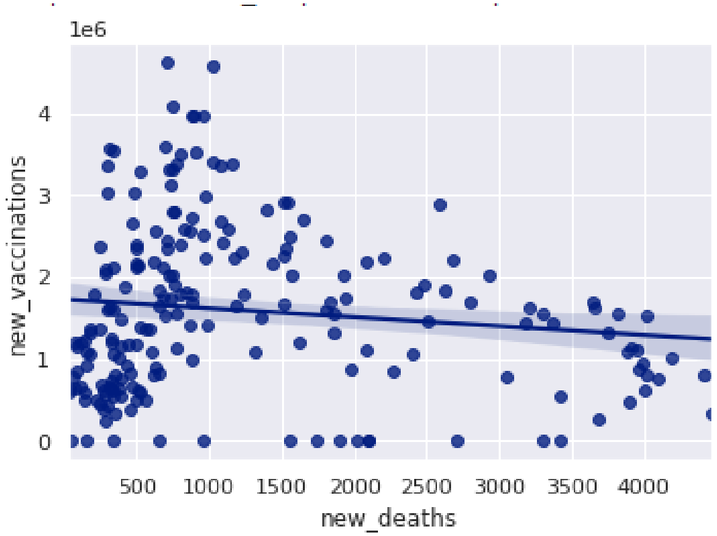
- Features as input
- Labels as output

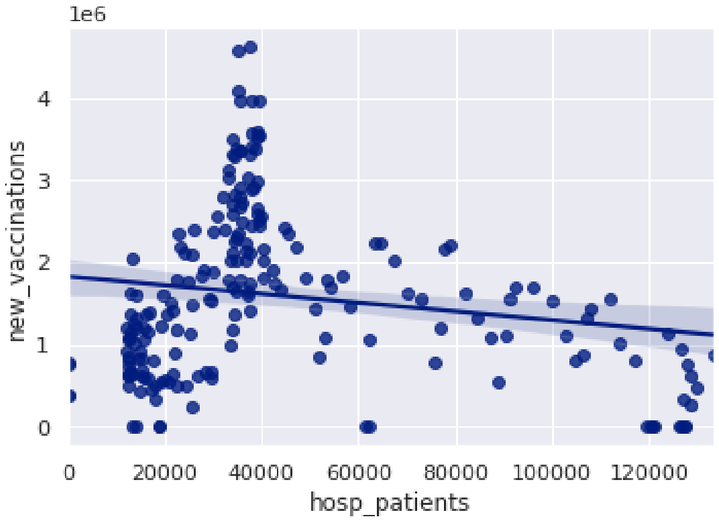
Pre-processimg

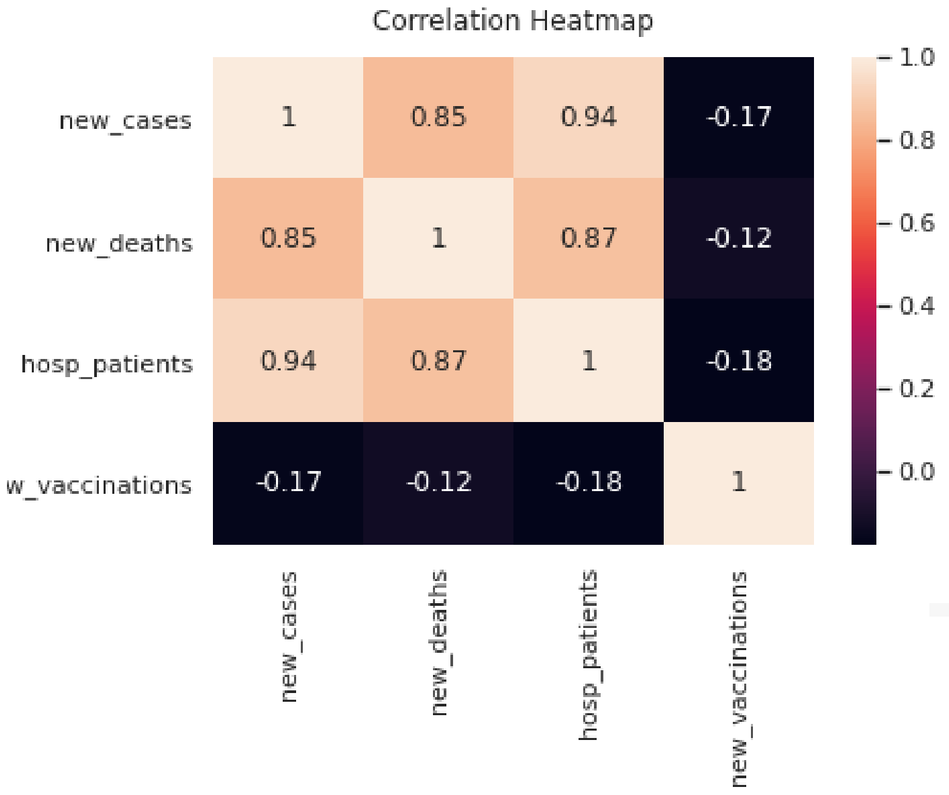
Feature Selection

We selected features that correlated to the daily vaccinations





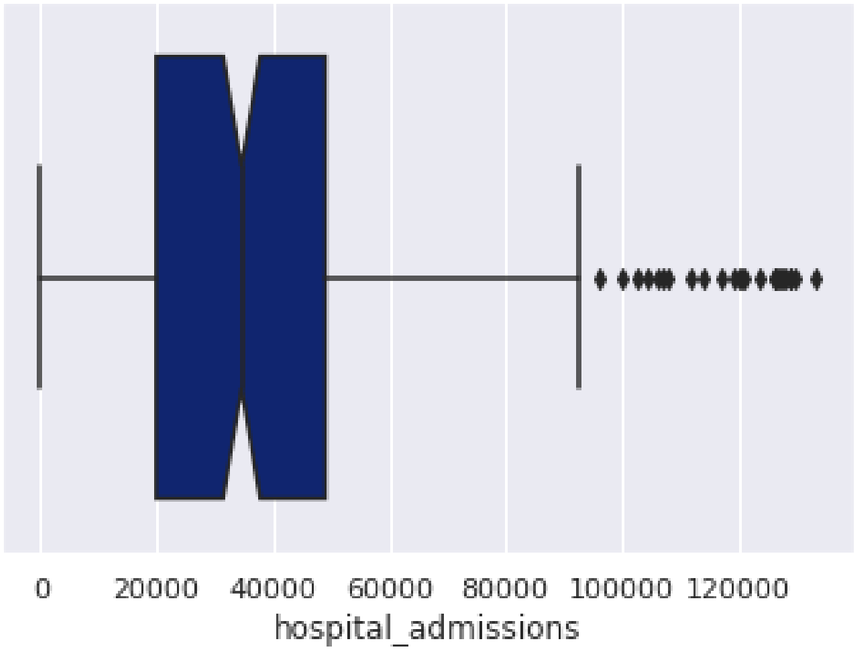


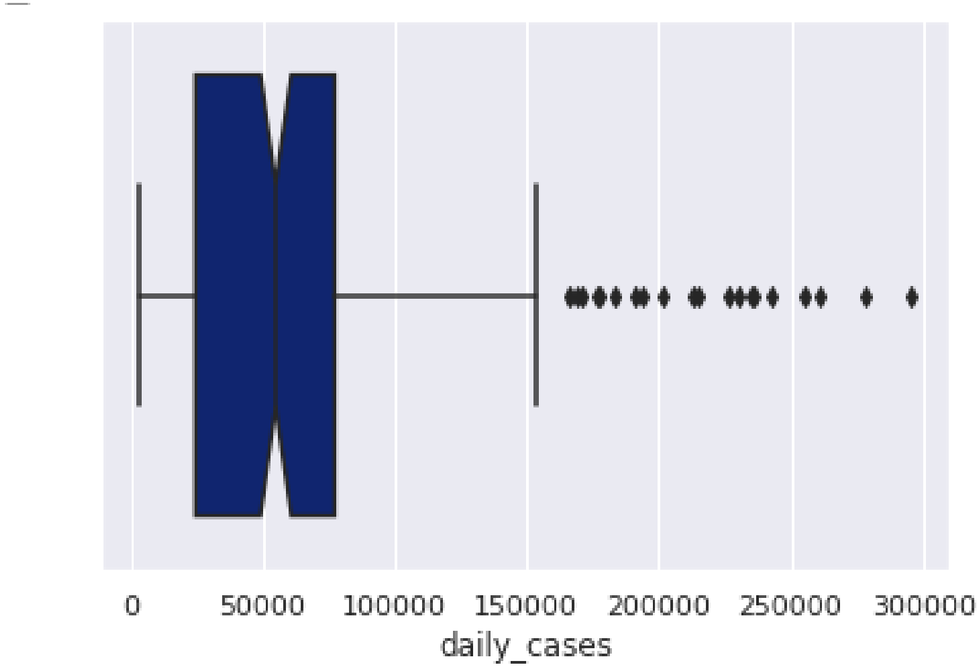



```
Cases corr coefficient:  -0.16520956842380316
Deaths corr coefficient:  -0.12404115574846716
Hospitalised patients corr coefficient:  -0.17521577183857312
```

date	id	hospital_admissions	daily_cases	vaccinations_administered
2021-01-05	1	128826.0	235042.0	273209.0
2021-01-06	2	129769.0	255637.0	470328.0
2021-01-07	3	128782.0	278337.0	612621.0
2021-01-08	4	127854.0	295257.0	768813.0
2021-01-09	5	126384.0	260967.0	0.0

It's always a good idea to check for outliers.





Feature Engineering

This is necessary for:

- Preparing the proper input dataset, compatible with the machine learning algorithm requirements.
- Improving the performance of machine learning models.

```
# Stationary vaccination data
usa_vax['diff_vaccinations']=usa_vax['vaccinations_administered'].diff(periods=1)
# Adding lag variable
usa_vax['prev_day_vaccination']=usa_vax['vaccinations_administered'].shift(1)
usa_vax['prev_3_day_vaccination']=usa_vax['vaccinations_administered'].shift(3)
usa_vax['prev_6_day_vaccination']=usa_vax['vaccinations_administered'].shift(6)

# Adding smoothed variable
usa_vax['3_day_average_vaccination']=usa_vax['vaccinations_administered'].rolling(window=3).mean()

# Adding lag variable for the next day
usa_vax['next_day_vaccinations']=usa_vax['vaccinations_administered'].shift(-1)
```

Feature Engineering

date	id	hospital_admissions	daily_cases	vaccinations_administered
2021-07-23	200	29607.0	118797.0	600157.0
2021-07-24	201	29596.0	27395.0	676050.0
2021-07-25	202	0.0	16008.0	778996.0
2021-07-26	203	0.0	89096.0	393083.0
2021-07-27	204	0.0	70740.0	395489.0

diff_vaccinations	prev_day_vaccination	prev_3_day_vaccination	prev_6_day_vaccination	3_day_average_vaccination	next_day_vaccinations
-60741.0	660898.0	243940.0	635290.0	624182.7	676050.0
75893.0	600157.0	611493.0	500910.0	645701.7	778996.0
102946.0	676050.0	660898.0	507076.0	685067.7	393083.0
-385913.0	778996.0	600157.0	243940.0	616043.0	395489.0
2406.0	393083.0	676050.0	611493.0	522522.7	753984.0

We shall use the following features:

- id
- daily_cases
- hospital_admissions
- vaccinations_administered
- prev_day_vaccination
- prev_3_day_vaccination
- prev_6_day_vaccination
- 3_day_average_vaccination

The **Label (target)** is:

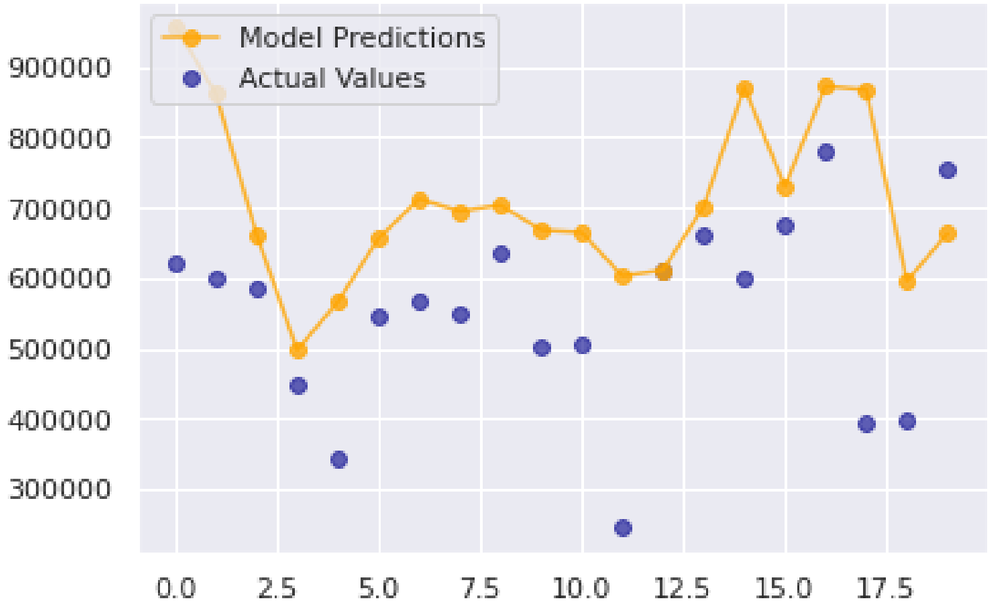
- next_day_vaccinations

Then **split the data into training and test** datasets.

```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size = 0.10, shuffle=False)
```

Decision Tree Model

```
# Import the model we are using
from sklearn.tree import DecisionTreeRegressor
# Initialise model
dtr = RandomForestRegressor(random_state = 42)
# Train the model on training data
dtr_fit=rf.fit(train_features, train_labels)
# Make predictions
dtr_pred = dtr_fit.predict(test_features)
```

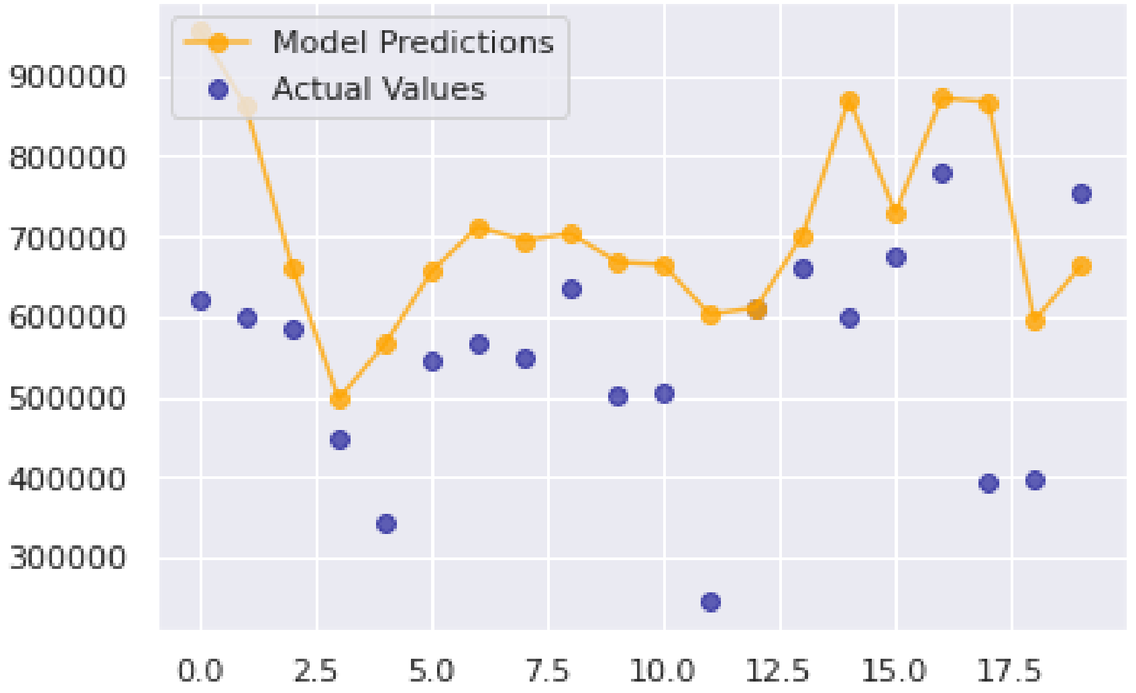


Error rate: 0.402

Accuracy: 59.8%

Random Forest Model

```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf_fit=rf.fit(train_features, train_labels)
# Use the forest's predict method on the test data
rf_pred = rf_fit.predict(test_features)
```

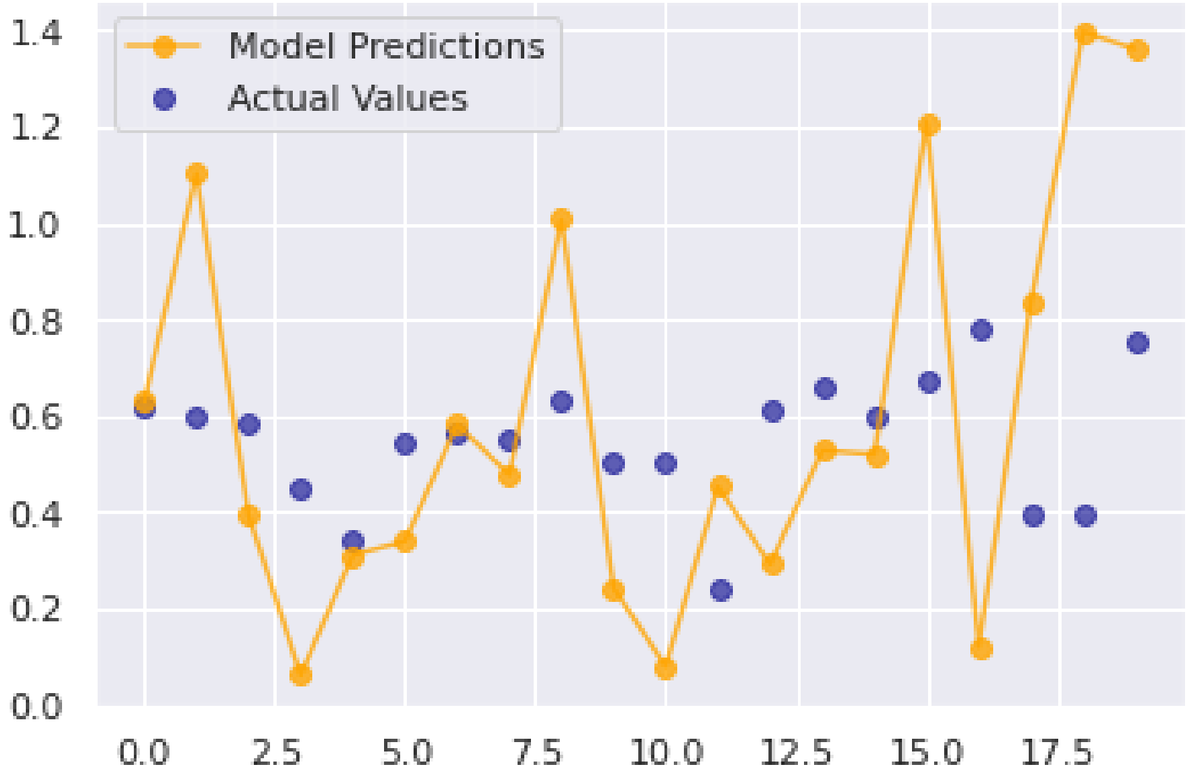


Error rate: 0.402

Accuracy: 59.8%

Linear Regression Model

```
] from sklearn import linear_model

  model = linear_model.LinearRegression()
  model.fit(train_features,train_labels)
  lreg_pred =model.predict(test_features)

  visualize_model(lreg_pred)
```



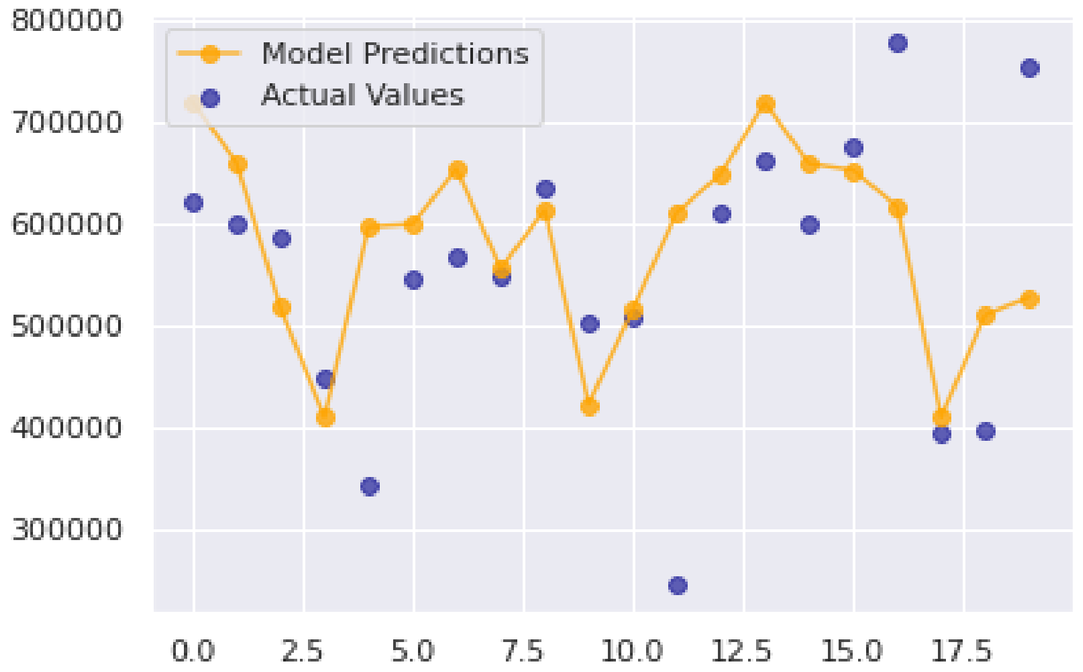
Error rate: 0.799

Accuracy: 20.1%

XG Boost Model

```python
import xgboost as xg

# Instantiation
xgb_r = xg.XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7, colsample_bytree=0.8)

# Fitting the model
xgb_r.fit(train_features, train_labels)

# Make predictions
xgb_pred = xgb_r.predict(test_features)
```

```python
visualize_model(xgb_pred)
```



Error rate: 0.254

Accuracy: 74.6%

Evaluating the Models

We used the following metrics to evaluate our models

- Mean Squared Error
- Mean Absolute Percentage Error.
- Error Rate
- Accuracy

```
def evaluate_model(prediction):

  from sklearn.metrics import mean_squared_error as MSE

  # RMSE Computation
  rmse = np.sqrt(MSE(test_labels, lreg_pred))
  print("Root Mean Squared Eroor : % f" %(rmse))

  # Calculate mean absolute percentage error (MAPE)
  mape = (rmse / test_labels)
  # Calculate and display accuracy
  err = np.mean(mape[~np.isinf(mape)])
  print('Error rate:',round(err,3))
  accuracy = (1-(np.mean(mape[~np.isinf(mape)])))*100
  print('Accuracy: ',round(accuracy,1))
```

We chose the XG Boost Model since it had the best accuracy.

Figures generated by the XG Boost Model

date	Vaccinations_Administered	ACTUAL_next_day_vaccinations	PREDICTED_next_day_vaccinations
2021-07-08	694333.0	620612.0	719786.8
2021-07-09	620612.0	598995.0	660449.4
2021-07-10	598995.0	586244.0	517309.5
2021-07-11	586244.0	449122.0	409753.0
2021-07-12	449122.0	341466.0	596564.8
2021-07-13	341466.0	545543.0	599135.9
2021-07-14	545543.0	567174.0	653763.2
2021-07-15	567174.0	549205.0	555358.4
2021-07-16	549205.0	635290.0	613190.6
2021-07-17	635290.0	500910.0	422255.7

date	Vaccinations_Administered	ACTUAL_next_day_vaccinations	PREDICTED_next_day_vaccinations
2021-07-18	500910.0	507076.0	515303.4
2021-07-19	507076.0	243940.0	610796.1
2021-07-20	243940.0	611493.0	648717.8
2021-07-21	611493.0	660898.0	718655.4
2021-07-22	660898.0	600157.0	658678.3
2021-07-23	600157.0	676050.0	652334.1
2021-07-24	676050.0	778996.0	616641.2
2021-07-25	778996.0	393083.0	409384.1
2021-07-26	393083.0	395489.0	510174.8
2021-07-27	395489.0	nan	527337.1

On the 28th of July 753984 vaccinations were administered.

Data is key.

All models are wrong but some are useful

You can always retrain andoptimize your model.

Take Aways

- Some regions had missing data and we would recommend better reporting systems for analysis and prediction.

- There is a big disparity in the resources available, which controls the adversity of the effects.

- Tracking reinfection has been a challenge. This is a major factor to take into consideration to determine the rate of economic recovery of many countries.

Questions?