In [1]:
```
import os, re, glob
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import folium
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from IPython.display import IFrame
from IPython.display import Image
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

This workshop would analyse COVID cases in Summer 2021. Summer months between July and August are considered to be the hottest time of the year, and it is believed that there is usually an upsurge of COVID cases during summer months. This report would help us know if there was an increase in the number of COVID-19 cases in Summer 2021.

In [2]:
```
root = 'C:\\Users\\User\\Documents\\COVID'
recent_date = "08-01-2021"
previous_date = "07-01-2021"

duplicate_columns = {"Lat": "Latitude",
                     "Long_": "Longitude",
                     "Incidence_Rate": "Incident_Rate",
                     "Case-Fatality_Ratio": "Case_Fatality_Ratio",
                     "Province/State": "Province_State",
                     "Country/Region": "Country_Region",
                     "Last Update": "Last_Update"}

recent_df = pd.read_csv(os.path.join(root, (recent_date + ".csv")))
previous_df = pd.read_csv(os.path.join(root, (previous_date + ".csv")))

for key, value in duplicate_columns.items():
    if key in recent_df.columns:
        recent_df = recent_df.rename(columns={key: value})
    if key in previous_df.columns:
        previous_df = previous_df.rename(columns={key: value})
```

In [3]:
```
recent_df.head()
```

Out[3]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confirmed	Deaths	Recovered	Active	Combined_Key	Inc
0	NaN	NaN	NaN	Afghanistan	2021-08-02 04:21:36	33.93911	67.709953	147501	6737	82586.0	58178.0	Afghanistan	
1	NaN	NaN	NaN	Albania	2021-08-02 04:21:36	41.15330	20.168300	133121	2457	130243.0	421.0	Albania	4
2	NaN	NaN	NaN	Algeria	2021-08-02 04:21:36	28.03390	1.659600	172564	4291	116009.0	52264.0	Algeria	
3	NaN	NaN	NaN	Andorra	2021-08-02 04:21:36	42.50630	1.521800	14678	128	14210.0	340.0	Andorra	18
4	NaN	NaN	NaN	Angola	2021-08-02 04:21:36	-11.20270	17.873900	42815	1016	37397.0	4402.0	Angola	

In [4]:
```
previous_df.head()
```

Out[4]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confirmed	Deaths	Recovered	Active	Combined_Key	Inc
0	NaN	NaN	NaN	Afghanistan	2021-07-02 04:21:47	33.93911	67.709953	120216	4962	71924.0	43330.0	Afghanistan	
1	NaN	NaN	NaN	Albania	2021-07-02 04:21:47	41.15330	20.168300	132523	2456	130014.0	53.0	Albania	4
2	NaN	NaN	NaN	Algeria	2021-07-02 04:21:47	28.03390	1.659600	140075	3726	97380.0	38969.0	Algeria	
3	NaN	NaN	NaN	Andorra	2021-07-02 04:21:47	42.50630	1.521800	13918	127	13721.0	70.0	Andorra	18
4	NaN	NaN	NaN	Angola	2021-07-02 04:21:47	-11.20270	17.873900	38965	903	33271.0	4791.0	Angola	

In [5]:
```
current_df = pd.DataFrame(columns=['Province_State','Country_Region','Confirmed','Deaths'])
current_df['Province_State'] = recent_df['Province_State']
current_df['Country_Region'] = recent_df['Country_Region']
current_df['Confirmed'] = recent_df['Confirmed'] - previous_df['Confirmed']
current_df['Deaths'] = recent_df['Deaths'] - previous_df['Deaths']
```

In [6]:
```
current_df.shape
```

Out[6]:
```
(4014, 4)
```

In [7]:
```
current_df.head()
```

Out[7]:

	Province_State	Country_Region	Confirmed	Deaths
0	NaN	Afghanistan	27285	1775
1	NaN	Albania	598	1
2	NaN	Algeria	32489	565
3	NaN	Andorra	760	1
4	NaN	Angola	3850	113

In [8]:
```
name_number = 'DekeAdeleye-2229810a.csv'
current_df.to_csv(name_number, index=False)
```

In [9]:
```
data = pd.read_csv(name_number)
```

In [10]:
```
data.head()
```

Out[10]:

	Province_State	Country_Region	Confirmed	Deaths
0	NaN	Afghanistan	27285	1775
1	NaN	Albania	598	1
2	NaN	Algeria	32489	565
3	NaN	Andorra	760	1
4	NaN	Angola	3850	113

In [11]:
```
print(data.shape)
```

```
(4014, 4)
```

In [12]:
```
print(data.count())
```

```
Province_State    3835
Country_Region    4014
Confirmed         4014
Deaths            4014
dtype: int64
```

In [13]:
```
#Q1. Print how many null values occur in the dataset . From line 13, 179 null vlues
data.apply(lambda x: sum(x.isnull()), axis = 0)
```

Out[13]:
```
Province_State    179
Country_Region      0
Confirmed           0
Deaths              0
dtype: int64
```

In [14]:
```
data.loc[data['Province_State'].isnull(),'Province_State'] = data['Country_Region']
```

In [15]:
```
data.head()
```

Out[15]:

	Province_State	Country_Region	Confirmed	Deaths
0	Afghanistan	Afghanistan	27285	1775
1	Albania	Albania	598	1
2	Algeria	Algeria	32489	565
3	Andorra	Andorra	760	1
4	Angola	Angola	3850	113

In [16]:
```
states = data['Province_State'].unique()
print("Number of unique States - ", len(states))
```

```
Number of unique States -  773
```

Q2. Print how many unique countries exist in the dataset using a similar approach. From line #17,201 Unique countries exist in the daabase.

In [17]:
```
Countries = data['Country_Region'].unique()
print("Number of unique Countries - ", len(Countries))
```

```
Number of unique Countries -  201
```

In [18]:
```
import datetime, time, requests
from time import sleep
from geopy.geocoders import Nominatim


def get_lat_lon(place):
    geolocator = Nominatim(user_agent=name_number)
    location = geolocator.geocode(place)
    lat_lon = location.latitude, location.longitude

    output = [float(i) for i in lat_lon]
    return output
```

In [19]:
```
data['Province_State'].value_counts()
```

Out[19]:
```
Texas                255
Georgia              162
Virginia             134
Kentucky             121
Missouri             117
                    ...
Manipur                1
Meghalaya              1
Mizoram                1
Nagaland               1
Pitcairn Islands       1
Name: Province_State, Length: 773, dtype: int64
```

In [20]:
```
from tqdm import tqdm

geo_lat = []
geo_lon = []

not_found = []
found = []
for state in tqdm(states):
    time.sleep(0.2)
    lat_lon = [None, None]
    try:
        lat_lon = get_lat_lon(state)
        found.append(state)
    except:
        not_found.append(state)

    geo_lat.append(lat_lon[0])
    geo_lon.append(lat_lon[1])

if len(not_found) > 0:
    print("Locations are not found for - ", not_found)
else:
    print("Found all the locations")
```

```
100%|██████████████████████████████████████████████████████████████████████████| 773/773 [06:35<00:00,  1.96it/
s]

Locations are not found for -  ['Repatriated Travellers', 'Sakha (Yakutiya) Republic', 'Summer Olympics 2020', 'W.P.
Kuala Lumpur']
```

In [21]:
```
states_list = states.tolist() #converting states to list to index list's items
lats = []
lons = []
for i, r in data.iterrows():
    state = r['Province_State']
    index_list = states_list.index(state)
    lats.append(geo_lat[index_list])
    lons.append(geo_lon[index_list])


data['Latitude'] = lats
data['Longitude'] = lons
```

In [22]:
```
data.head()
```

Out[22]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude
0	Afghanistan	Afghanistan	27285	1775	33.768006	66.238514
1	Albania	Albania	598	1	41.000028	19.999962
2	Algeria	Algeria	32489	565	28.000027	2.999983
3	Andorra	Andorra	760	1	42.540717	1.573203
4	Angola	Angola	3850	113	-11.877577	17.569124

Q3. Check whether the latitude and longitude values we retrieved from geopy are same as the latitude and longitude given in the dataset. Identify and report differences in values. The longitude and latitude values retrieved from geopy is slightly different from the one on the data. This is most likely due to differences in precision. Geopy might round up numbers to a different decimal place from what is in my dataset.

In [23]:
```
data = data[data['Latitude'].notna()]
```

In [24]:
```
data.shape
```

Out[24]:
```
(4010, 6)
```

In [25]:
```
clustering_data = data[["Confirmed", "Deaths"]]
```

In [26]:
```
clustering_data.head()
```

Out[26]:

	Confirmed	Deaths
0	27285	1775
1	598	1
2	32489	565
3	760	1
4	3850	113

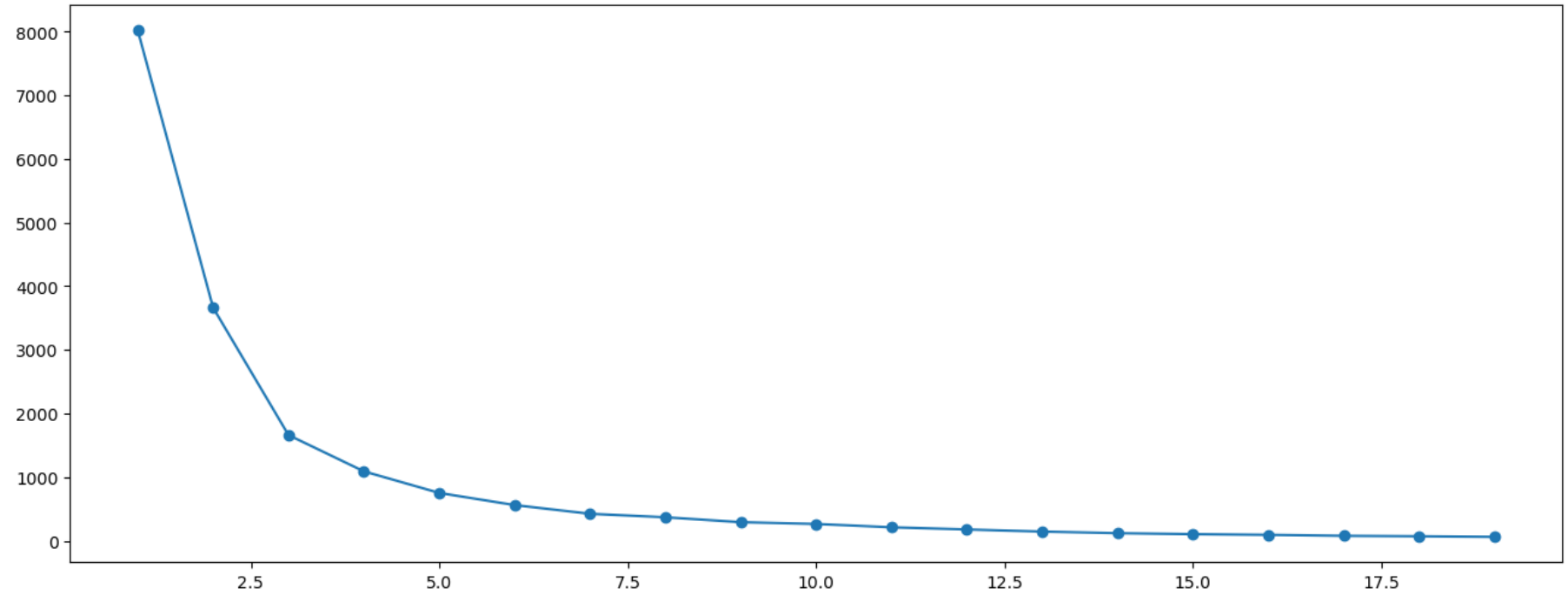
In [27]:
```
scaler = StandardScaler()
X_scaled = scaler.fit(clustering_data).transform(clustering_data.astype(np.float))
```

In [28]:
```
cluster_range = range( 1, 20 )
cluster_errors = []

for num_clusters in cluster_range:
    clusters = KMeans( num_clusters )
    clusters.fit( X_scaled )
    cluster_errors.append( clusters.inertia_ )

clusters_df = pd.DataFrame( { "num_clusters":cluster_range,
                              "cluster_errors": cluster_errors } )

plt.figure(figsize=(16,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" );
```



In [29]:
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 10)
y_kmeans = kmeans.fit_predict(X_scaled)

#beginning of  the cluster numbering with 1 instead of 0
y_kmeans1=y_kmeans+1

# New list called cluster
cluster = list(y_kmeans1)
# Adding cluster to our data set
clustering_data['cluster'] = cluster
```

In [30]:
```
clustering_data.head(10)
```

Out[30]:

	Confirmed	Deaths	cluster
0	27285	1775	1
1	598	1	1
2	32489	565	1
3	760	1	1
4	3850	113	1
5	39	1	1
6	444296	11000	3
7	5118	102	1
8	0	0	1
9	3474	15	1

In [31]:
```
kmeans_mean_cluster = pd.DataFrame(round(clustering_data.groupby('cluster').mean(),1))
kmeans_mean_cluster
```

Out[31]:

cluster	Confirmed	Deaths
1	1768.9	27.6
2	1237288.0	36728.0
3	487924.8	7691.4
4	174485.2	2993.4

In [32]:
```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 2]
```

Out[32]:

	Province_State	cluster
286	Indonesia	2

In [33]:
```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 3]
```

Out[33]:

	Province_State	cluster
6	Argentina	3
20	Bangladesh	3
65	Sao Paulo	3
265	Kerala	3
269	Maharashtra	3
287	Iran	3
597	South Africa	3
3966	England	3

In [34]:
```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 4]
```

Out[34]:

	Province_State	cluster
53	Minas Gerais	4
56	Parana	4
60	Rio Grande do Sul	4
61	Rio de Janeiro	4
71	Burma	4
152	Capital District	4
185	Cuba	4
195	Ecuador	4
216	France	4
288	Iraq	4
363	Kazakhstan	4
492	Philippines	4
539	Moscow	4
556	Saint Petersburg	4
608	Catalonia	4
649	Thailand	4
653	Tunisia	4
654	Turkey	4
3981	Vietnam	4
3997	Selangor	4

In [35]:
```
data.head()
```

Out[35]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude	cluster
0	Afghanistan	Afghanistan	27285	1775	33.768006	66.238514	1
1	Albania	Albania	598	1	41.000028	19.999962	1
2	Algeria	Algeria	32489	565	28.000027	2.999983	1
3	Andorra	Andorra	760	1	42.540717	1.573203	1
4	Angola	Angola	3850	113	-11.877577	17.569124	1

In [36]:
```
def get_color(cluster_id):
    if cluster_id == 2:
        return 'darkred'
    if cluster_id == 1:
        return 'green'
    if cluster_id == 3:
        return 'orange'
    if cluster_id == 4:
        return 'yellow'

data["color"] = data["cluster"].apply(lambda x: get_color(x))
```

In [37]:
```
data.head(10)
```

Out[37]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude	cluster	color
0	Afghanistan	Afghanistan	27285	1775	33.768006	66.238514	1	green
1	Albania	Albania	598	1	41.000028	19.999962	1	green
2	Algeria	Algeria	32489	565	28.000027	2.999983	1	green
3	Andorra	Andorra	760	1	42.540717	1.573203	1	green
4	Angola	Angola	3850	113	-11.877577	17.569124	1	green
5	Antigua and Barbuda	Antigua and Barbuda	39	1	17.223472	-61.955461	1	green
6	Argentina	Argentina	444296	11000	-34.996496	-64.967282	3	orange
7	Armenia	Armenia	5118	102	40.769627	44.673665	1	green
8	Australian Capital Territory	Australia	0	0	-35.488350	149.002694	1	green
9	New South Wales	Australia	3474	15	-31.875984	147.286949	1	green

In [38]:
```
this_map = folium.Map(location =[data["Latitude"].mean(),
                                 data["Longitude"].mean()], zoom_start=5)

def plot_dot(point):
    '''input: series that contains a numeric named latitude and a numeric named longitude
    this function creates a CircleMarker and adds it to your this_map'''
    folium.CircleMarker(location=[point.Latitude, point.Longitude],
                        radius=2,
                        color=point.color,
                        weight=1).add_to(this_map)




#clustered_full.apply(,axis=1) #use this to iterate through every row in your dataframe
data.apply(plot_dot, axis = 1)


#Set the zoom to the maximum possible
this_map.fit_bounds(this_map.get_bounds())

#Save the map to an HTML file
this_map.save(os.path.join('covid_map.html'))
```

According to the result, cluster 1 has a relatively low number of confirmed cases and deaths in summer 2021. Countries in the cluster include Afghanistan, Albania, Algeria,Andora. According to IMF (2022), COVID infectons in the countries in cluster 1 reduced significantly and they all relaxed their restrictions and their borders were even opened as at summer 2021.
Even though their are not many countries in cluster 2, the countries in cluster 2 had an upsurge of confirmed cases and deaths in summer 2021. For instance, in the summer of 2021, from June 2021 until August 2021, Indonesia experienced an unprecedented surge in the number of daily new confirmed coronavirus disease 2019 (COVID-19) cases. Indonesia's previous record of the highest number of daily new confirmed cases was 14, 518 cases, recorded on 30 January 2021, by 15 July 2021, the number had increased almost four-fold to 56, 757 cases (Johns Hopkins University & Medicine, 2021). This is responsible for the high number of confirmed cases and deaths in cluster 2.
Cluters 3 and 4 have relatively average cases of confirmed cases and few deaths in summer 2021. Cluster 3 had an average of 487925 confirmed cases and 7691 deaths. The number of confirmed cases reduced further in cluster 4 with 174485 cases and 2993 deaths.

From the maps, it also shows that COVID 19 cases is not restricted to a particular region or continent of the world. The countries in the clusters are in different parts of the world.

References

Johns Hopkins University & Medicine (2021) . COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) [Accessed 28 February 2023]. Available at: https://coronavirus.•jhu.edu/

International Monetary Fund (IMF)(2022). Policy Responses to COVID-19 [Accessed 28 February 2023] Available at:https://www.imf.org/en/Topics/imf-and-covid19/Policy-Responses-to-COVID-19