ACTIVITY 3.1

In this notebook, we are analysing Winter period in the year 2021 starting from January ("01-01-2021") to ("03-31-2021")¶

In [1]:

```
# Importing necessary libraries needed for the analysis
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
```

In [2]:

```
root = 'C:/Users/bidam/Downloads/Covid/'
recent_date = "03-31-2021"
previous_date = "01-01-2021"

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]:

```
# Viewing the two dataframes:
recent_df.head()
```

Out[3]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confir
0	NaN	NaN	NaN	Afghanistan	2021-04-01 04:27:05	33.93911	67.709953	5
1	NaN	NaN	NaN	Albania	2021-04-01 04:27:05	41.15330	20.168300	12
2	NaN	NaN	NaN	Algeria	2021-04-01 04:27:05	28.03390	1.659600	11
3	NaN	NaN	NaN	Andorra	2021-04-01 04:27:05	42.50630	1.521800	1
4	NaN	NaN	NaN	Angola	2021-04-01 04:27:05	-11.20270	17.873900	2

In [4]:

```
previous_df.head()
```

Out[4]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Latitude	Longitude	Confir
0	NaN	NaN	NaN	Afghanistan	02/01/2021 05:22	33.93911	67.709953	5
1	NaN	NaN	NaN	Albania	02/01/2021 05:22	41.15330	20.168300	5
2	NaN	NaN	NaN	Algeria	02/01/2021 05:22	28.03390	1.659600	9
3	NaN	NaN	NaN	Andorra	02/01/2021 05:22	42.50630	1.521800	
4	NaN	NaN	NaN	Angola	02/01/2021 05:22	-11.20270	17.873900	1

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	3941.0	283.0
1	NaN	Albania	66841.0	1054.0
2	NaN	Algeria	17295.0	331.0
3	NaN	Andorra	3893.0	31.0
4	NaN	Angola	4743.0	132.0

In [8]:

```
name_number = 'AbidemiAleem-2128712A.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	3941.0	283.0
1	NaN	Albania	66841.0	1054.0
2	NaN	Algeria	17295.0	331.0
3	NaN	Andorra	3893.0	31.0
4	NaN	Angola	4743.0	132.0

In [11]:

```
print(data.shape)
```

```
(4014, 4)
```

In [12]:

```
print(data.count())
```

```
Province_State    3835
Country_Region    4014
Confirmed         4011
Deaths            4011
dtype: int64
```

In [13]:

```
# Question 1: generating null vallues in the dataset.
print(data.isnull())
```

```
      Province_State  Country_Region  Confirmed  Deaths
0               True           False      False   False
1               True           False      False   False
2               True           False      False   False
3               True           False      False   False
4               True           False      False   False
...              ...             ...        ...     ...
4009           False           False      False   False
4010            True           False      False   False
4011           False           False       True    True
4012            True           False       True    True
4013           False           False       True    True

[4014 rows x 4 columns]
```

Answer: Corresponding with the original dataset's rows and columns, the table comprises 4014 rows and 4 columns. Following that, this output will be useful in determining which columns and rows contain missing or null values, so that they may be addressed appropriately.

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	3941.0	283.0
1	Albania	Albania	66841.0	1054.0
2	Algeria	Algeria	17295.0	331.0
3	Andorra	Andorra	3893.0	31.0
4	Angola	Angola	4743.0	132.0

In [16]:

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

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

In [17]:

```
#Question 2: Print number of unique countries
country = data['Country_Region'].unique()
print("Number of unique countries - ", len(country))
```

```
Number of unique countries -  201
```

As seen above, 201 unique countries exist in the dataset

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 [06:27<00:00,  2.00it/s]

Locations are not found for -  ['Repatriated Travellers', 'Sakha (Yakutiy
a) 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	3941.0	283.0	33.768006	66.238514
1	Albania	Albania	66841.0	1054.0	41.000028	19.999962
2	Algeria	Algeria	17295.0	331.0	28.000027	2.999983
3	Andorra	Andorra	3893.0	31.0	42.540717	1.573203
4	Angola	Angola	4743.0	132.0	-11.877577	17.569124

Question 3:

The Latitude and longitude values from geopy is just a bit low (almost same value) to the values from our dataset which might not be statiscally significant. In our dataset, the first roll record [33.93911 & 67.709953] respectively. while our geopy record [33.768006 & 66.238514].

In [23]:

```
#Selected rows without NaN
data.dropna(inplace=True)
```

In [24]:

```
data.shape
```

Out[24]:

```
(4007, 6)
```

observation: our dataframe row is reduced from [4014] to [4007] after removing null-value from our

datase

In [25]:

```
clustering_data = data[["Confirmed", "Deaths"]]
```

In [26]:

```
clustering_data.head()
```

Out[26]:

	Confirmed	Deaths
0	3941.0	283.0
1	66841.0	1054.0
2	17295.0	331.0
3	3893.0	31.0
4	4743.0	132.0

In [27]:

```
scaler = StandardScaler()
X_scaled = scaler.fit(clustering_data).transform(clustering_data.astype(np.float))
```

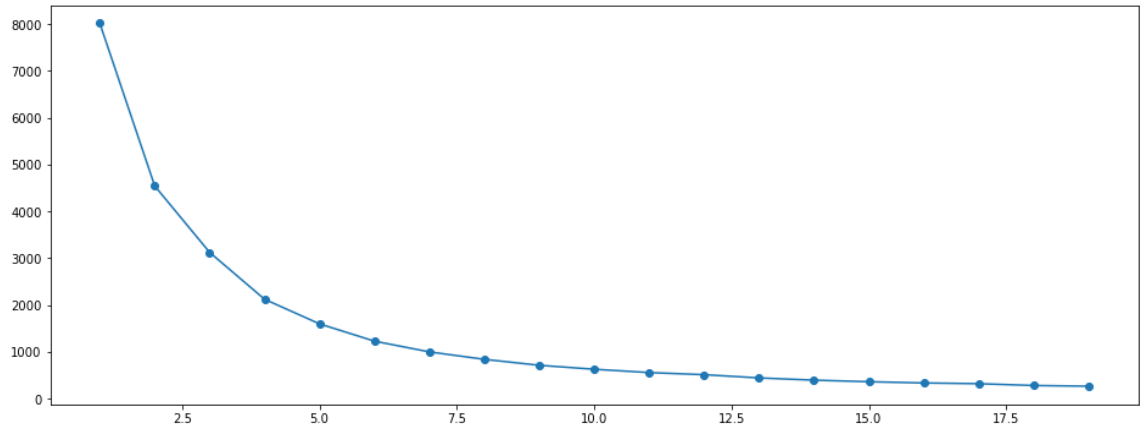
In [28]:

```
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" );
```



K- Means Clustering:

Based on how far apart our data points are from one another, it clusters related data points into a set number of clusters (K). In the plot above, we choose the optimal value for K; K=4 which is our x-axis and Y-axis is our cluster error.

In [29]:

```
# Fitting K-Means to the dataset
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	3941.0	283.0	1
1	66841.0	1054.0	1
2	17295.0	331.0	1
3	3893.0	31.0	1
4	4743.0	132.0	1
5	977.0	23.0	1
6	719227.0	12539.0	3
7	32901.0	687.0	1
8	5.0	0.0	1
9	349.0	0.0	1

In [31]:

```
kmeans_mean_cluster = pd.DataFrame(round(clustering_data.groupby('cluster').mean(),1))
kmeans_mean_cluster
```

Out[31]:

cluster	Confirmed	Deaths
1	3006.0	78.7
2	3263106.6	74613.0
3	723291.0	17239.0
4	-1518821.6	-43858.8

In [32]:

```
print(abs(kmeans_mean_cluster))
```

```
         Confirmed   Deaths
cluster
1           3006.0     78.7
2        3263106.6  74613.0
3         723291.0  17239.0
4        1518821.6  43858.8
```

As seen above cluster 2 has the highest number of deaths and confirmed cases while cluster 1 has the lowset.

In [34]:

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

Out[34]:

	Province_State	cluster
216	France	2
269	Maharashtra	2
287	Iran	2
650	Turkey	2
3960	England	2

In [35]:

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

Out[35]:

	Province_State	cluster
6	Argentina	3
53	Minas Gerais	3
56	Parana	3
60	Rio Grande do Sul	3
65	Sao Paulo	3
152	Capital District	3
187	Czechia	3
221	Bayern	3
229	Nordrhein-Westfalen	3
247	Hungary	3
250	Andhra Pradesh	3
264	Karnataka	3
265	Kerala	3
279	Tamil Nadu	3
286	Indonesia	3
288	Iraq	3
290	Israel	3
299	Lombardia	3
311	Veneto	3
362	Jordan	3
371	Lebanon	3
393	Ciudad de Mexico	3
401	Mexico	3
477	Lima	3
489	Philippines	3
490	Poland	3
491	Portugal	3
493	Romania	3
536	Moscow	3
553	Saint Petersburg	3
586	Serbia	3
590	Slovakia	3
594	South Africa	3
596	Andalusia	3
605	Catalonia	3
610	Madrid	3
640	Switzerland	3

	Province_State	cluster
756	Arizona	3
860	California	3
1279	Illinois	3

In [36]:

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

Out[36]:

	Province_State	cluster
214	St Martin	4
267	Lakshadweep	4
285	West Bengal	4
474	Junin	4
487	Ucayali	4
591	Slovenia	4
647	Togo	4
3956	Bermuda	4

In [37]:

```
data['cluster'] = cluster
clusters = data[['Province_State', 'cluster']]
clusters.loc[clusters['cluster'] == 1]
```

Out[37]:

	Province_State	cluster
0	Afghanistan	1
1	Albania	1
2	Algeria	1
3	Andorra	1
4	Angola	1
...	...	...
4006	Jersey	1
4007	Guernsey	1
4008	Korea, North	1
4009	Unknown	1
4010	Nauru	1

3954 rows × 2 columns

Question 4:

It is interesting to know that cluster 1 with the lowest number of deaths and confirmed cases has the highest number of province_state with (3954 rows), while cluster 2 shows a smaller set of countries (5) that are all assigned to it. Cluster 2 records highest number of death and confirm cases.

In [38]:

```
data.head()
```

Out[38]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude	cluster
0	Afghanistan	Afghanistan	3941.0	283.0	33.768006	66.238514	1
1	Albania	Albania	66841.0	1054.0	41.000028	19.999962	1
2	Algeria	Algeria	17295.0	331.0	28.000027	2.999983	1
3	Andorra	Andorra	3893.0	31.0	42.540717	1.573203	1
4	Angola	Angola	4743.0	132.0	-11.877577	17.569124	1

In [39]:

```
def get_color(cluster_id):
    if cluster_id == 2:
        return 'darkred'
    if cluster_id == 1:
        return 'green'
    if cluster_id == 3:
        return 'yellow'
    if cluster_id == 4:
        return 'blue'
data["color"] = data["cluster"].apply(lambda x: get_color(x))
```

In [40]:

```
data.head(10)
```

Out[40]:

	Province_State	Country_Region	Confirmed	Deaths	Latitude	Longitude	cluster	col
0	Afghanistan	Afghanistan	3941.0	283.0	33.768006	66.238514	1	gre
1	Albania	Albania	66841.0	1054.0	41.000028	19.999962	1	gre
2	Algeria	Algeria	17295.0	331.0	28.000027	2.999983	1	gre
3	Andorra	Andorra	3893.0	31.0	42.540717	1.573203	1	gre
4	Angola	Angola	4743.0	132.0	-11.877577	17.569124	1	gre
5	Antigua and Barbuda	Antigua and Barbuda	977.0	23.0	17.223472	-61.955461	1	gre
6	Argentina	Argentina	719227.0	12539.0	-34.996496	-64.967282	3	yell
7	Armenia	Armenia	32901.0	687.0	40.769627	44.673665	1	gre
8	Australian Capital Territory	Australia	5.0	0.0	-35.488350	149.002694	1	gre
9	New South Wales	Australia	349.0	0.0	-31.875984	147.286949	1	gre

In [41]:

```
#create a map
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 longitud
    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_map3a.html'))
```

REPORT:

In this note book, we considered winter period between January to March year 2021 (after the covid outbreak), to analyse the covid dataset and determine whether the negative effect is more during this period than during summer.

K-Means clustering was used to visualise covid affected areas on the world map based on geo-location. The following tasks were performed to achieve this:

1. Grouping similar data points together by creating a seperate dataset for the preferred period where we subtract the number of confirmed cases and number of deaths in previous_df from recent_df.
2. Clustering dataset were normalised
3. The number of clusters was determined from our graph at optimal value of K=4.
4. We assign diferent colors to our cluster to visualise on the map. Red to cluster 2 because it represent high risk areas, green to cluster 1 low risk areas, yellow to cluster 3 it represent moderately high risk areas and blue to cluster 4 mild risks areas.

Data clustering is one of the most essential and interesting tasks to classify patterns in different similar entities. According to [1] the absence of truth complicates assessing quality of clustering because is unsupervised and real-world datasets typically do not fall into obvious clusters. Base on the results from analysising covid dataset i disagree because it is possible to improve the quality of clustering and this was achieved on the dataset analysed following this methods;

1. Ensuring my data is scaled by perform a visual check that similar entities appears in the same clusters.
2. Performing similiarity measure
3. Optimum number of clusters was determined choosing k=3.
4. Fitting k-mans to the dataset.

Therefore, as seen in covid_map3a.html, the dataset fall into clusters with different colors as selected. Dark-red represents cluster 2 with highest number of death and confirmed cases although few states fall in this category while blue represent cluster1 with lowest number of death and confirmed cases majority of the states fall in this category and yellow cluster 3.

REFRENCES

[1] Dr. Vinita Nahar Workshop 3-K-Means clustering, covid-19 Notebook

[2] Google Developers. (n.d.). Interpret Results and Adjust Clustering. [online] Available at: https://developers.google.com/machine-learning/clustering/interpret (https://developers.google.com/machine-learning/clustering/interpret).

[3] Silveira, O.S. (2023). K-Means Clustering Explained in 2023 (with 12 Code Examples). [online] Dataquest. Available at: https://www.dataquest.io/blog/tutorial-k-means-clustering/ (https://www.dataquest.io/blog/tutorial-k-means-clustering/) [Accessed 1 Mar. 2023].

In [ ]: