Covid-19 Dashboard's Dataset

By Abdullah Almuzaini

Table of Contents

Introduction

The first step in my project is to collect the data needed when building up the dashboard. Second, after gathering the data I need, I will have to prepare the datasets downloaded in the first step. By preparing the data I mean assessing and doing some cleaning process. Lastly, when all the data needed for the project is collected and be in the proper format and shape, it has to be extracted from this notebook and stored in the project dataset folder in the project directory. These Processes require using some python scripts I need to write and libraries, which I will be describing next.

In [1]:

```
import pandas as pd


from create_new_directory import new_folder
from get_dataset import get_dataset
from reshape_dataset import reshape
```

Gathering Data

The python scripts I will be using this part of the project are:

- create_new_directory.py: which contains the methodnew_folder() that takes a string variable as a folder name. The purpose of this little script is to create a new directory inside the current working directory. For now, it will be used to create a new directory in which the datasets will be stored.

- get_dataset.py: the function downloads the dataset from the internet using requests library and save it in the 'dataset' directory inside the current running direcroty. It takes two arguments one is the URL, and the second argument is file name which should includ the file format `csv` for example.

Collect the daily covid-19 confirmed cases dataset

In [2]:

```
# Create a new directory in the current running directory if it does not exist

folder_name = 'dataset'
new_folder(folder_name)
```

In [54]:

```
# Download the confirmed covid-19 cases from its source https://github.com/CSSEGISandData
/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_dat
a/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
file_name = 'covid19_confirmed_cases.csv'

# The functoin of get_dataset takes url and file name
data = get_dataset(URL,file_name)
confirmed_df = pd.read_csv('dataset'+ '/'+file_name)
```

In [4]:

```
confirmed_df.head()
```

Out[4]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	9/16/21	9/17
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	154361	154
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	160365	161
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	200989	201
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	15113	15
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	51827	52

5 rows × 617 columns

Collect the daily covid-19 death cases dataset

In [5]:

```
# Download the death covid-19 cases from its source https://github.com/CSSEGISandData/COV
ID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

URL = 'https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covi
d_19_time_series/time_series_covid19_deaths_global.csv'
file_name = 'covid19_death_cases.csv'

# The functoin of get_dataset takes url and file name
data = get_dataset(URL,file_name)
death_df = pd.read_csv('dataset'+ '/'+file_name)

death_df.head()
```

Out[5]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	9/16/21	9/17
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	7183	7
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	2563	2
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	5651	5
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	130	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	1371	1

5 rows × 617 columns

Collect the data of the daily recovery from covid19

In [6]:

```
URL = 'https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv'
file_name = 'covid19_recovered_cases.csv'

# The functoin of get_dataset takes url and file name
data = get_dataset(URL,file_name)
recovery_df = pd.read_csv('dataset'+ '/'+file_name)

recovery_df.head()
```

Out[6]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	9/16/21	9/17
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	0	

5 rows × 617 columns

> **Now, we have three datasets downloaded and stored in the dataset directory. The dataset are in the wide format as they appear above.**

Part II- Preparing Data

- In this part of the project, I will need to use the following python script:

- reshape_dataset.py: the function contains a method calledreshape() that transforms the dataset from the wide format into the long format using the pandas function melt(). The method takes three arguments. One is the dataframe. The second one is variable name, the column name.

Prepare the three datasets to be in an appropriate format and shape by converting them from the wide format to the long format.

In [7]:

```
confirmed = reshape(confirmed_df,
                    'date',
                    'confirmed')
confirmed.shape
```

Out[7]:

```
(171027, 6)
```

In [8]:

```
confirmed.sample(5)
```

Out[8]:

	Province/State	Country/Region	Lat	Long	date	confirmed
56436	Macau	China	22.1667	113.5500	8/11/20	46

	Province/State	Country/Region	Lat	Long	date	confirmed
33029	NaN	Dominica	15.4150	-61.3710	5/19/20	16
124840	Saint Pierre and Miquelon	France	46.8852	-56.3159	4/13/21	24
122055	NaN	Gambia	13.4432	-15.3101	4/3/21	5505
52223	Prince Edward Island	Canada	46.5107	-63.4168	7/27/20	36

In [9]:

```
death = reshape(death_df,
                'date',
                'deaths')
death.shape
```

Out[9]:

```
(171027, 6)
```

In [16]:

```
death.tail(5)
```

Out[16]:

	Province/State	Country/Region	Lat	Long	date	deaths
171022	NaN	Vietnam	14.058324	108.277199	9/25/21	18400
171023	NaN	West Bank and Gaza	31.952200	35.233200	9/25/21	4018
171024	NaN	Yemen	15.552727	48.516388	9/25/21	1682
171025	NaN	Zambia	-13.133897	27.849332	9/25/21	3645
171026	NaN	Zimbabwe	-19.015438	29.154857	9/25/21	4603

In [17]:

```
recovery = reshape(recovery_df,
                'date',
                'recovery')
recovery.shape
```

Out[17]:

```
(161832, 6)
```

In [19]:

```
recovery.sample(5)
```

Out[19]:

	Province/State	Country/Region	Lat	Long	date	recovery
64913	NaN	Thailand	15.870032	100.992541	9/23/20	3353
151751	NaN	Singapore	1.283300	103.833300	8/18/21	0
90078	Henan	China	33.882000	113.614000	12/28/20	1270
43692	NaN	India	20.593684	78.962880	7/5/20	424433
125727	Macau	China	22.166700	113.550000	5/12/21	49

> **As we can see above, the datasets are now looking in a suitable format. I have transformed the columns after the Long column, which represent the date of each day in the dataset from January 22, 2020, till September 25, 2021, into one column called date and stored all the values under each of the date columns in**

a new variable called recovery in the recovery dataset, deaths in the death dataset, and confirmed in the confirmed dataset.

Part III- Exploring the datasets

Next, I will explore the datasets taking advantage of the pandas method info(), which will return general information about each dataset such as the number of records, the name of the variables we have, the count of the non-null records, as well as the data type of each column.

In [20]:

```
confirmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171027 entries, 0 to 171026
Data columns (total 6 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Province/State  53331 non-null   object 
 1   Country/Region  171027 non-null  object 
 2   Lat             169801 non-null  float64
 3   Long            169801 non-null  float64
 4   date            171027 non-null  object 
 5   confirmed       171027 non-null  int64  
dtypes: float64(2), int64(1), object(3)
memory usage: 7.8+ MB
```

In [21]:

```
death.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171027 entries, 0 to 171026
Data columns (total 6 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Province/State  53331 non-null   object 
 1   Country/Region  171027 non-null  object 
 2   Lat             169801 non-null  float64
 3   Long            169801 non-null  float64
 4   date            171027 non-null  object 
 5   deaths          171027 non-null  int64  
dtypes: float64(2), int64(1), object(3)
memory usage: 7.8+ MB
```

In [22]:

```
recovery.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161832 entries, 0 to 161831
Data columns (total 6 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Province/State  43523 non-null   object 
 1   Country/Region  161832 non-null  object 
 2   Lat             161219 non-null  float64
 3   Long            161219 non-null  float64
 4   date            161832 non-null  object 
 5   recovery        161832 non-null  int64  
dtypes: float64(2), int64(1), object(3)
memory usage: 7.4+ MB
```

According to the above summary about the confirmed and death dataframes, we have some missing values in the Province/State, Lat, and Long columns. We also can see that the column data is not in the proper format. Furthermore, the confirmed and death datasets are a match. However, the recovery dataset has overall less records than the other two and has more missing values that needs to be inspected to find out what the issue is.

In the next cell, I will make copies of three datasets, and the copies will contain only records with NaN values in the `Province/State` to find out where the issue is.

In [23]:

```
# Storing only rows with null values in ['Province/State'] column

recovery_na = recovery[recovery['Province/State'].isna()]
confirmed_na = confirmed[confirmed['Province/State'].isna()]
death_na = death[death['Province/State'].isna()]
```

Next I will use pandas.DataFrame.sample to return a random samples form the above sub-datasets to invistigate the issue.

In [25]:

```
recovery_na.sample(20)
```

Out[25]:

	Province/State	Country/Region	Lat	Long	date	recovery
151722	NaN	Niger	17.607789	8.081666	8/18/21	0
127770	NaN	Venezuela	6.423800	-66.589700	5/19/21	202625
50922	NaN	Timor-Leste	-8.874200	125.727500	8/1/20	24
85013	NaN	Antigua and Barbuda	17.060800	-61.796400	12/9/20	138
118101	NaN	Dominican Republic	18.735700	-70.162700	4/13/21	217008
154958	NaN	United Kingdom	55.378100	-3.436000	8/30/21	0
71321	NaN	Chad	15.454200	18.732200	10/18/20	1181
138892	NaN	Bosnia and Herzegovina	43.915900	17.679100	7/1/21	183534
101086	NaN	Turkey	38.963700	35.243300	2/7/21	2420706
135252	NaN	Cuba	21.521757	-77.781167	6/17/21	154494
115076	NaN	Trinidad and Tobago	10.691800	-61.222500	4/1/21	7618
43947	NaN	Guatemala	15.783500	-90.230800	7/6/20	3429
66301	NaN	Cambodia	11.550000	104.916700	9/29/20	275
155532	NaN	Cabo Verde	16.538800	-23.041800	9/2/21	0
9100	NaN	Guinea	9.945600	-9.696600	2/25/20	0
84695	NaN	Singapore	1.283300	103.833300	12/7/20	58168

	Province/State	Country/Region	Lat	Long	date	recovery
47339	NaN	Croatia	45.100000	15.200000	7/19/20	3018
110834	NaN	Solomon Islands	-9.645700	160.156200	3/16/21	16
51092	NaN	Japan	36.204824	138.252924	8/2/20	25748
99001	NaN	Albania	41.153300	20.168300	1/31/21	47424

In [26]:

```
confirmed_na.sample(20)
```

Out[26]:

	Province/State	Country/Region	Lat	Long	date	confirmed
116939	NaN	Cameroon	3.848000	11.502100	3/16/21	40622
87248	NaN	Nicaragua	12.865416	-85.207229	11/29/20	5784
143423	NaN	Azerbaijan	40.143100	47.576900	6/19/21	335437
97366	NaN	Vietnam	14.058324	108.277199	1/4/21	1497
34155	NaN	Fiji	-17.713400	178.065000	5/23/20	18
22151	NaN	Egypt	26.820553	30.802498	4/10/20	1794
114966	NaN	Bahamas	25.025885	-78.035889	3/9/21	8642
149939	NaN	Ethiopia	9.145000	40.489700	7/12/21	277137
27095	NaN	Bulgaria	42.733900	25.485800	4/28/20	1399
54361	NaN	South Africa	-30.559500	22.937500	8/3/20	516862
116181	NaN	Fiji	-17.713400	178.065000	3/13/21	66
73612	NaN	South Africa	-30.559500	22.937500	10/11/20	692471
32058	NaN	Tunisia	33.886917	9.537499	5/15/20	1035
119630	NaN	Rwanda	-1.940300	29.873900	3/25/21	21210
44916	NaN	Yemen	15.552727	48.516388	6/30/20	1158
44777	NaN	Grenada	12.116500	-61.679000	6/30/20	23
129755	NaN	Bangladesh	23.685000	90.356300	5/1/21	760584
76561	NaN	Eswatini	-26.522500	31.465900	10/22/20	5814
118995	NaN	Guyana	4.860416	-58.930180	3/23/21	9732
48723	NaN	Maldives	3.202800	73.220700	7/14/20	2801

In [27]:

```
death_na.sample(20)
```

Out[27]:

	Province/State	Country/Region	Lat	Long	date	deaths
154817	NaN	Trinidad and Tobago	10.691800	-61.222500	7/29/21	1057
113704	NaN	Ireland	53.142400	-7.692100	3/4/21	4396
65941	NaN	Cote d'Ivoire	7.540000	-5.547100	9/14/20	120
117395	NaN	Qatar	25.354800	51.183900	3/17/21	270
135193	NaN	Kazakhstan	48.019600	66.923700	5/20/21	7149
91106	NaN	Israel	31.046051	34.851612	12/13/20	2999
153068	NaN	Malaysia	4.210484	101.975766	7/23/21	7718
132422	NaN	Malaysia	4.210484	101.975766	5/10/21	1700
122024	NaN	Czechia	49.817500	15.473000	4/3/21	26867
143932	NaN	Tanzania	-6.369028	34.888822	6/20/21	21

	Province/State	Country/Region	Lat	Long	date	deaths
86423	NaN	Philippines	12.879721	121.774017	11/26/20	8242
44848	NaN	Panama	8.538000	-80.782100	6/30/20	631
89240	NaN	Sudan	12.862800	30.217600	12/6/20	1295
12515	NaN	Sudan	12.862800	30.217600	3/6/20	0
35262	NaN	Dominican Republic	18.735700	-70.162700	5/27/20	474
149258	NaN	Vanuatu	-15.376700	166.959200	7/9/21	1
86373	NaN	Kuwait	29.311660	47.481766	11/26/20	872
160946	NaN	Sweden	60.128161	18.643501	8/20/21	14668
85509	NaN	Ghana	7.946500	-1.023200	11/23/20	323
152793	NaN	Marshall Islands	7.131500	171.184500	7/22/21	0

> **After Running the above three cells several times, it turns out that the reason for the null values in the column `Province/State` is that many countries did not report covid-19 cases by Province/State. Instead, they just counted the covid-19 cases for the entire country as a whole. However, there is one country that did not constantly report the cases. That country is Canada. Canada reports covid-19 deaths and confirmed cases by the Province/State, but they do not count the recovery cases in the same way. They report the recovery cases for the entire country as a whole without considering the Province/State. This issue would raise an issue when merging the three datasets.**

The following three cells will demonstrate the issue of Canada's covid-19 reporting method

In [28]:

```
death_na[death_na['Country/Region']== "Canada"]
```

Out[28]:

Province/State	Country/Region	Lat	Long	date	deaths

In [29]:

```
confirmed_na[confirmed_na['Country/Region']== "Canada"]
```

Out[29]:

Province/State	Country/Region	Lat	Long	date	confirmed

In [30]:

```
recovery_na[recovery_na['Country/Region']== "Canada"]
```

Out[30]:

	Province/State	Country/Region	Lat	Long	date	recovery
39	NaN	Canada	56.1304	-106.3468	1/22/20	0
303	NaN	Canada	56.1304	-106.3468	1/23/20	0

	Province/State	Country/Region	Lat	Long	date	recovery
567	NaN	Canada	56.1304	-106.3468	1/24/20	0
831	NaN	Canada	56.1304	-106.3468	1/25/20	0
1095	NaN	Canada	56.1304	-106.3468	1/26/20	0
...	...	...	...	...	...	...
160551	NaN	Canada	56.1304	-106.3468	9/21/21	0
160815	NaN	Canada	56.1304	-106.3468	9/22/21	0
161079	NaN	Canada	56.1304	-106.3468	9/23/21	0
161343	NaN	Canada	56.1304	-106.3468	9/24/21	0
161607	NaN	Canada	56.1304	-106.3468	9/25/21	0

613 rows × 6 columns

In order to resolve the above issue, I will recalculate the deaths and confirmed cases of Covid-19 in Canada by the Country to match the recovery dataset.

The first step we need to take is to fetch confirmed cases and deaths records of Canada and aggregate them by `date`, then store them in a separate sub-data frame

In [31]:

```
canada_conf = confirmed[confirmed['Country/Region'] == 'Canada'].groupby('date').sum()[[
'confirmed']]
canada_conf.head()
```

Out[31]:

date	confirmed
1/1/21	591149
1/10/21	666375
1/11/21	674624
1/12/21	681015
1/13/21	688097

In [32]:

```
canada_dth = death[death['Country/Region'] == 'Canada'].groupby('date').sum()[['deaths']
]
canada_dth.head()
```

Out[32]:

date	deaths
1/1/21	15806
1/10/21	17074
1/11/21	17199
1/12/21	17359

1/13/21	17539 deaths

The next step is to copy the recovery dataframe without including the `recovery` column. The reason is that I want to apply these columns on death and confirmed cases dataframes to make the death and confirmed cases calculated for the entire country as a whole instead of by Province/State, so they eventually match the recovery dataframe

In [33]:

```
canada_recovery = recovery[recovery['Country/Region'] == 'Canada'][recovery.columns[:-1]
].reset_index(drop=True)
canada_recovery.head()
```

Out[33]:

	Province/State	Country/Region	Lat	Long	date
0	NaN	Canada	56.1304	-106.3468	1/22/20
1	NaN	Canada	56.1304	-106.3468	1/23/20
2	NaN	Canada	56.1304	-106.3468	1/24/20
3	NaN	Canada	56.1304	-106.3468	1/25/20
4	NaN	Canada	56.1304	-106.3468	1/26/20
...	...	...	...	...	...
608	NaN	Canada	56.1304	-106.3468	9/21/21
609	NaN	Canada	56.1304	-106.3468	9/22/21
610	NaN	Canada	56.1304	-106.3468	9/23/21
611	NaN	Canada	56.1304	-106.3468	9/24/21
612	NaN	Canada	56.1304	-106.3468	9/25/21

613 rows × 5 columns

Now, we are set to join and apply `canada_recovery` on the `canada_conf` and `canada_dth`

In [34]:

```
canada_covid_19_conf = canada_recovery.merge(canada_conf, how='inner', left_on='date', r
ight_index=True)
canada_covid_19_conf.head()
```

Out[34]:

	Province/State	Country/Region	Lat	Long	date	confirmed
0	NaN	Canada	56.1304	-106.3468	1/22/20	0
1	NaN	Canada	56.1304	-106.3468	1/23/20	0
2	NaN	Canada	56.1304	-106.3468	1/24/20	0
3	NaN	Canada	56.1304	-106.3468	1/25/20	0
4	NaN	Canada	56.1304	-106.3468	1/26/20	1

In [35]:

```
canada_covid_19_conf.tail()
```

Out[35]:

	Province/State	Country/Region	Lat	Long	date	confirmed
608	NaN	Canada	56.1304	-106.3468	9/21/21	1593674
609	NaN	Canada	56.1304	-106.3468	9/22/21	1597789
610	NaN	Canada	56.1304	-106.3468	9/23/21	1602413
611	NaN	Canada	56.1304	-106.3468	9/24/21	1606974
612	NaN	Canada	56.1304	-106.3468	9/25/21	1608019

In [36]:

```
canada_covid_19_deaths = canada_recovery.merge(canada_dth, how='inner', left_on='date',
right_index=True)
canada_covid_19_deaths.head()
```

Out[36]:

	Province/State	Country/Region	Lat	Long	date	deaths
0	NaN	Canada	56.1304	-106.3468	1/22/20	0
1	NaN	Canada	56.1304	-106.3468	1/23/20	0
2	NaN	Canada	56.1304	-106.3468	1/24/20	0
3	NaN	Canada	56.1304	-106.3468	1/25/20	0
4	NaN	Canada	56.1304	-106.3468	1/26/20	0

In [37]:

```
canada_covid_19_deaths.tail()
```

Out[37]:

	Province/State	Country/Region	Lat	Long	date	deaths
608	NaN	Canada	56.1304	-106.3468	9/21/21	27539
609	NaN	Canada	56.1304	-106.3468	9/22/21	27590
610	NaN	Canada	56.1304	-106.3468	9/23/21	27636
611	NaN	Canada	56.1304	-106.3468	9/24/21	27677
612	NaN	Canada	56.1304	-106.3468	9/25/21	27690

> **Finally, we have Canada's Covid-19 information matched in the three dataframes**
>
> **Now, we need to put Canada's data back in the original dataframes (`confirmed`, `death`) by copying the dataframes excluding records of Canada, then insert Canada's data from (`canada_covid_19_deaths` and `canada_covid_19_conf`)**

In [40]:

```
confirmed = confirmed[confirmed['Country/Region'] != 'Canada'].append(canada_covid_19_con
```

```
f)
confirmed.head()
```

Out[40]:

	Province/State	Country/Region	Lat	Long	date	confirmed
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0
1	NaN	Albania	41.15330	20.168300	1/22/20	0
2	NaN	Algeria	28.03390	1.659600	1/22/20	0
3	NaN	Andorra	42.50630	1.521800	1/22/20	0
4	NaN	Angola	-11.20270	17.873900	1/22/20	0

In [44]:

```
death = death[death['Country/Region'] != 'Canada'].append(canada_covid_19_deaths)
death.head()
```

Out[44]:

	Province/State	Country/Region	Lat	Long	date	deaths
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0
1	NaN	Albania	41.15330	20.168300	1/22/20	0
2	NaN	Algeria	28.03390	1.659600	1/22/20	0
3	NaN	Andorra	42.50630	1.521800	1/22/20	0
4	NaN	Angola	-11.20270	17.873900	1/22/20	0

To make sure that the three dataframes match, I will run the same test I did here.

In [45]:

```
recovery_na = recovery[recovery['Province/State'].isna()]
confirmed_na = confirmed[confirmed['Province/State'].isna()]
death_na = death[death['Province/State'].isna()]
```

In [46]:

```
print(confirmed_na[confirmed_na['Country/Region']== "Canada"].shape)
print(recovery_na[recovery_na['Country/Region']== "Canada"].shape)
print(death_na[death_na['Country/Region']== "Canada"].shape)
```

```
(613, 6)
(613, 6)
(613, 6)
```

> **The three dataframes now are matched**

The Final Step is to combine the three datasets into one master dataset

In [47]:

```
columns = ['Country/Region','Province/State','date']
master_df = confirmed.merge(death, how='inner', on=columns)
master_df = master_df.merge(recovery, how='inner', on=columns)
master_df = master_df.drop(columns=['Lat_x', 'Long_x','Lat_y','Long_y'])
master_df = master_df[['Country/Region','Province/State','Lat','Long','date', 'confirmed
','recovery','deaths']]
master_df.head()
```

Out[47]:

	Country/Region	Province/State	Lat	Long	date	confirmed	recovery	deaths
0	Afghanistan	NaN	33.93911	67.709953	1/22/20	0	0	0
1	Albania	NaN	41.15330	20.168300	1/22/20	0	0	0
2	Algeria	NaN	28.03390	1.659600	1/22/20	0	0	0
3	Andorra	NaN	42.50630	1.521800	1/22/20	0	0	0
4	Angola	NaN	-11.20270	17.873900	1/22/20	0	0	0

> **Now, we have a dataframe containing all the covid-19 data we need and in the proper format.**

After having the covid-19 data downloaded and almost prepared, there is only one last step to make the all the data ready for my fueature analysis. The data we have so far needs the population information for each country. Thus, I will add the population information to the my dataset list, and the population dataset I will be using is Population by Country - 2020 by Tanu N Prabhu.

To download the dataset, I will need to use Kaggle API. Kaggle API is an API that can be used to interact with the Kaggle website through the command line. The use of the Kaggle can be downloading and uploading datasets or interacting with competitions.

In [48]:

```
# Download the population dataset from its source https://www.kaggle.com/tanuprabhu/popul
ation-by-country-2020
from kaggle.api.kaggle_api_extended import KaggleApi
api = KaggleApi()
api.authenticate()

api.dataset_download_files('tanuprabhu/population-by-country-2020', path = 'dataset',  un
zip=True)
population_df = pd.read_csv('dataset/population_by_country_2020.csv')
population_df.head()
```

Out[48]:

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
0	China	1440297825	0.39 %	5540090	153	9388211	-348399.0	1.7	38	61 %	18.47 %
1	India	1382345085	0.99 %	13586631	464	2973190	-532687.0	2.2	28	35 %	17.70 %

2	United States	331341050	0.59 %	1937734	36	9147420	954806.0	1.8	38	83 %	4.25 %
3	Indonesia	274021604	1.07 %	2898047	151	1811570	-98955.0	2.3	30	56 %	3.51 %
4	Pakistan	221612785	2.00 %	4327022	287	770880	-233379.0	3.6	23	35 %	2.83 %

After downloading the dataset, the first thing I would like to check is the matching of the countries names.

In [49]:

```
countries_covid = master_df['Country/Region'].unique()
countries_pop = population_df['Country (or dependency)'].unique()
unmatched = [x for x in countries_covid if x not in countries_pop]
print(unmatched)
print(len(unmatched))
print([x for x in countries_pop if x not in countries_covid])
```

```
['Burma', 'Congo (Brazzaville)', 'Congo (Kinshasa)', "Cote d'Ivoire", 'Czechia', 'Diamond
Princess', 'Korea, South', 'Kosovo', 'MS Zaandam', 'Saint Kitts and Nevis', 'Saint Vincen
t and the Grenadines', 'Sao Tome and Principe', 'Summer Olympics 2020', 'Taiwan*', 'US',
'West Bank and Gaza']
16
['United States', 'DR Congo', 'Myanmar', 'South Korea', "Côte d'Ivoire", 'North Korea', '
Taiwan', 'Czech Republic (Czechia)', 'Hong Kong', 'Turkmenistan', 'Congo', 'State of Pale
stine', 'Puerto Rico', 'Réunion', 'Macao', 'Western Sahara', 'Guadeloupe', 'Martinique',
'French Guiana', 'New Caledonia', 'French Polynesia', 'Mayotte', 'Sao Tome & Principe', '
Channel Islands', 'Guam', 'Curaçao', 'St. Vincent & Grenadines', 'Aruba', 'Tonga', 'U.S.
Virgin Islands', 'Isle of Man', 'Cayman Islands', 'Bermuda', 'Northern Mariana Islands',
'Greenland', 'American Samoa', 'Saint Kitts & Nevis', 'Faeroe Islands', 'Sint Maarten', '
Turks and Caicos', 'Saint Martin', 'Gibraltar', 'British Virgin Islands', 'Caribbean Neth
erlands', 'Cook Islands', 'Anguilla', 'Tuvalu', 'Wallis & Futuna', 'Nauru', 'Saint Barthe
lemy', 'Saint Helena', 'Saint Pierre & Miquelon', 'Montserrat', 'Falkland Islands', 'Niue
', 'Tokelau']
```

There is 13 unmatched countries and 2 Cruises (Diamond Princess and MS Zaandam) and the Summer Olympics 2020. The way to fix the unmatched countries names is by simply replacing them directly from the dataframe, and I will leave the other four for later analysis.

In [50]:

```
country_mapper = {
    'Congo (Brazzaville)': 'Congo',
    'Congo (Kinshasa)': 'Congo',
    "Cote d'Ivoire": "Côte d'Ivoire",
    'Czechia': 'Czech Republic (Czechia)',
    'Korea, South': 'South Korea',
    'Saint Vincent and the Grenadines': 'St. Vincent & Grenadines',
    'Taiwan*': 'Taiwan',
    'US': 'United States',
    'West Bank and Gaza': 'Israel',
    'Saint Kitts and Nevis': 'Saint Kitts & Nevis',
    'Burma': 'Myanmar',
    'Sao Tome and Principe': 'Sao Tome & Principe'
}
master_df['Country/Region'] = master_df['Country/Region'].replace(country_mapper)
```

In [51]:

```
countries_covid = master_df['Country/Region'].unique()
[x for x in countries_covid if x not in countries_pop]
```

Out[51]:

```
['Diamond Princess', 'Kosovo', 'MS Zaandam', 'Summer Olympics 2020']
```

> **Now we don't have any mismatchs within the countries names**

Now we don't have any mismatchs within the countries names.

Part IV- Exporting the datasets

The last remaining step is to export the datasets since they are almost ready to be uploaded on Tableau

In [53]:

```
master_df.to_csv('dataset/COVID-19.csv')
```