In [9]:

```
# Assignment - A1     |     Name : Pratik Pingale     |     Roll No : 19CO056
```

In [1]:

```
#subtask 1 - importing libraries
import pandas as pd
import numpy as np
```

In [2]:

```
#subtask 2 - Dataset url
# Name - COVID -19 Global Reports early March 2022
# URL - https://www.kaggle.com/danielfesalbon/covid-19-global-reports-early-mar
# local machine relative address - /covid_19_clean_complete_2022.csv/
```

In [3]:

```
#subtask 3 - Loading dataset
df = pd.read_csv("covid_19_clean_complete_2022.csv")
df = df.drop('Province/State', axis=1)
df.head()
```

Out[3]:

| | Country/Region | Lat | Long | Date | Confirmed | Deaths | Recovered | Active | WHO Regio |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 33.93911 | 67.709953 | 2020-01-22 | 0 | 0 | 0 | 0 | Easter Mediterranea |
| **1** | Albania | 41.15330 | 20.168300 | 2020-01-22 | 0 | 0 | 0 | 0 | Europ |
| **2** | Algeria | 28.03390 | 1.659600 | 2020-01-22 | 0 | 0 | 0 | 0 | Afric |
| **3** | Andorra | 42.50630 | 1.521800 | 2020-01-22 | 0 | 0 | 0 | 0 | Europ |
| **4** | Angola | -11.20270 | 17.873900 | 2020-01-22 | 0 | 0 | 0 | 0 | Afric |

In [4]:

```
#subtask 4 - data preprocessing - detecting NaN values and using describe() fun
column={}
for x in df.columns:
    column[x] = df[x].isnull().any()
print(column)

df.describe()
```

```
{'Country/Region': False, 'Lat': True, 'Long': True, 'Date': False, 'Confirme
d': False, 'Deaths': False, 'Recovered': False, 'Active': False, 'WHO Region':
True}
```

Out[4]:

| | Lat | Long | Confirmed | Deaths | Recovered | Active |
|---|---|---|---|---|---|---|
| **count** | 213348.000000 | 213348.000000 | 2.148940e+05 | 214894.000000 | 2.148940e+05 | 2.148940e+05 |
| **mean** | 20.528131 | 22.735337 | 4.578132e+05 | 9310.764693 | 1.079987e+05 | 3.405037e+05 |
| **std** | 25.899139 | 76.304185 | 2.708770e+06 | 47497.835275 | 8.470111e+05 | 2.516382e+06 |
| **min** | -71.949900 | -178.116500 | 0.000000e+00 | 0.000000 | 0.000000e+00 | -1.638280e+05 |
| **25%** | 6.426991 | -27.932425 | 2.530000e+02 | 2.000000 | 0.000000e+00 | 1.600000e+01 |
| **50%** | 22.233350 | 21.752000 | 5.223000e+03 | 71.000000 | 4.500000e+01 | 1.243000e+03 |
| **75%** | 41.166070 | 88.658375 | 9.892275e+04 | 1675.000000 | 5.115750e+03 | 2.644675e+04 |
| **max** | 71.706900 | 178.065000 | 7.925051e+07 | 958144.000000 | 3.097475e+07 | 7.829236e+07 |

In [5]:

```
#shape of dataset (dimensions)
df.shape
```

Out[5]:

```
(214894, 9)
```

In [6]:

```
#subtask 5 - data formatting and normalization
df.dtypes
```

Out[6]:

```
Country/Region     object
Lat               float64
Long              float64
Date               object
Confirmed           int64
Deaths              int64
Recovered           int64
Active              int64
WHO Region         object
dtype: object
```

In [7]:

```
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Out[7]:

```
Country/Region            object
Lat                      float64
Long                     float64
Date              datetime64[ns]
Confirmed                  int64
Deaths                     int64
Recovered                  int64
Active                     int64
WHO Region                object
dtype: object
```

In [8]:

```
#subtask 6 - handling categorical values
#dropping the categorical variable column
df = df.drop(['WHO Region','Country/Region'], axis=1)
df.head()
```

Out[8]:

| | Lat | Long | Date | Confirmed | Deaths | Recovered | Active |
|---|---|---|---|---|---|---|---|
| **0** | 33.93911 | 67.709953 | 2020-01-22 | 0 | 0 | 0 | 0 |
| **1** | 41.15330 | 20.168300 | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 28.03390 | 1.659600 | 2020-01-22 | 0 | 0 | 0 | 0 |
| **3** | 42.50630 | 1.521800 | 2020-01-22 | 0 | 0 | 0 | 0 |
| **4** | -11.20270 | 17.873900 | 2020-01-22 | 0 | 0 | 0 | 0 |