Phase 3

COVID-19 Cases Analysis

To load and preprocess a COVID-19 dataset for analysis, you can follow these general steps using Python and Pandas. Make sure you have a COVID-19 dataset in a suitable format available.

1. Import Libraries: Start by importing the necessary Python libraries, including Pandas, to load and preprocess the dataset.

```
Import pandas as pd
```

2. Load the COVID-19 Dataset: Load the COVID-19 dataset into a Pandas DataFrame. You can use `pd.read_csv()` for CSV files, but the method may vary depending on the file format.

```
In [4]: import pandas as pd
```

```
In [5]: df = pd.read_csv(r"C:\Users\MUHILAN\OneDrive\Documents\Covid_19_cases.csv")
        df
```

Out[5]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories
0	31-05-2021	31	5	2021	366	5	Austria
1	30-05-2021	30	5	2021	570	6	Austria
2	29-05-2021	29	5	2021	538	11	Austria
3	28-05-2021	28	5	2021	639	4	Austria
4	27-05-2021	27	5	2021	405	19	Austria
...	...	...	...	...	...	...	...
2725	06-03-2021	6	3	2021	3455	17	Sweden
2726	05-03-2021	5	3	2021	4069	12	Sweden
2727	04-03-2021	4	3	2021	4884	14	Sweden
2728	03-03-2021	3	3	2021	4876	19	Sweden
2729	02-03-2021	2	3	2021	6191	19	Sweden

2730 rows × 7 columns

3. Data Inspection: Before preprocessing, inspect the data to understand its structure and identify any potential issues.

```
In [6]: print(df.head())
```

```
      dateRep  day  month  year  cases  deaths countriesAndTerritories
0  31-05-2021   31      5  2021    366       5                 Austria
1  30-05-2021   30      5  2021    570       6                 Austria
2  29-05-2021   29      5  2021    538      11                 Austria
3  28-05-2021   28      5  2021    639       4                 Austria
4  27-05-2021   27      5  2021    405      19                 Austria
```

```
In [7]: print(df.isnull().sum())
```

```
dateRep                   0
day                       0
month                     0
year                      0
cases                     0
deaths                    0
countriesAndTerritories   0
dtype: int64
```

```
In [8]: print(df.dtypes)
```

```
dateRep                   object
day                        int64
month                      int64
year                       int64
cases                      int64
deaths                     int64
countriesAndTerritories   object
dtype: object
```

4. Data Preprocessing:

a. **Data Cleaning**:

- Handle missing values by either imputing them or removing rows with missing data.

In [10]:

```
df.fillna(0, inplace=True)
df.dropna(inplace=True)
df
```

Out[10]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories
0	31-05-2021	31	5	2021	366	5	Austria
1	30-05-2021	30	5	2021	570	6	Austria
2	29-05-2021	29	5	2021	538	11	Austria
3	28-05-2021	28	5	2021	639	4	Austria
4	27-05-2021	27	5	2021	405	19	Austria
...	...	...	...	...	...	...	...
2725	06-03-2021	6	3	2021	3455	17	Sweden
2726	05-03-2021	5	3	2021	4069	12	Sweden
2727	04-03-2021	4	3	2021	4884	14	Sweden
2728	03-03-2021	3	3	2021	4876	19	Sweden
2729	02-03-2021	2	3	2021	6191	19	Sweden

2730 rows × 7 columns

b. **Data Transformation**:

- If necessary, transform the data to suit your analysis objectives. For instance, you may want to aggregate data by date or region.

In [13]:
```
df['dateRep'] = pd.to_datetime(df['dateRep'])
df = df.groupby('dateRep').agg({'cases': 'sum', 'deaths': 'sum'}).reset_index()
df
```

Out[13]:

	dateRep	cases	deaths
0	2021-01-04	194036	2746
1	2021-01-05	96455	1884
2	2021-02-03	101010	1996
3	2021-02-04	183224	2731
4	2021-02-05	65030	1511
...	...	...	...
86	2021-11-04	127050	2188
87	2021-11-05	69278	1532
88	2021-12-03	150606	2349
89	2021-12-04	115514	1957
90	2021-12-05	77429	1735

91 rows × 3 columns

c. **Data Filtering**:

- Filter the data to focus on a specific time frame or specific regions of interest.

In [24]:
```
start_date = '2021-06-01'
end_date = '2021-12-31'
df = df[(df['dateRep'] >= start_date) & (df['dateRep'] <= end_date)]
df
```

Out[24]:

	dateRep	cases	deaths
70	2021-06-03	126808	2164
71	2021-06-04	81837	1688
72	2021-06-05	98484	2080
73	2021-07-03	106324	1490
74	2021-07-04	138694	2400
75	2021-07-05	96246	1938
76	2021-08-03	86795	1502
77	2021-08-04	155478	3293
78	2021-08-05	82359	1741
79	2021-09-03	107052	2174
80	2021-09-04	178293	3355
81	2021-09-05	56020	1465
82	2021-10-03	130530	2587
83	2021-10-04	163349	3150
84	2021-10-05	58466	989
85	2021-11-03	148377	2431
86	2021-11-04	127050	2188
87	2021-11-05	69278	1532
88	2021-12-03	150606	2349
89	2021-12-04	115514	1957
90	2021-12-05	77429	1735

5. Save the Preprocessed Data (Optional): If you want to save the preprocessed data for future analysis, you can use Pandas to save it to a new CSV file.

```
In [19]: df.to_csv('preprocessed_covid_data.csv', index=False)
```

These are the general steps to load and preprocess a COVID-19 dataset using Python and Pandas. Remember that the specific preprocessing steps and operations may vary depending on the structure of your dataset and your analysis objectives.