PHASE3:DEVELOPMENT PART1

Abstract:

This data analysis project focuses on COVID-19 cases and deaths in the European Union/European Economic Area (EU/EEA). The primary objectives are to compare and contrast the mean values and standard deviations of daily cases and associated deaths, ultimately deriving insights from this data. The project begins with clear definition of analysis objectives, followed by the acquisition of a COVID-19 dataset. Data preprocessing and cleaning steps ensure data accuracy and reliability, enabling meaningful analysis and visualization. The project employs Python for data manipulation and Matplotlib for visualization, allowing for the creation of informative charts and graphs. The resulting insights contribute to our understanding of the COVID-19 situation in the EU/EEA region and may inform public health decisions.

1. Define Analysis Objectives

Clearly define the objectives of your COVID-19 analysis. What insights are you looking to gain from the data? For example, your objectives might include:

- Analyzing the daily trend of COVID-19 cases and deaths.

- Comparing the situation across different countries in the EU/EEA.

- Identifying spikes or patterns in the data.

- Assessing the impact of various measures and interventions on the pandemic.

2. Obtain COVID-19 Data:

Acquire a reliable COVID-19 dataset that aligns with your analysis objectives. Consider using trusted sources like government health agencies, the World Health Organization (WHO), or reputable data providers. You can typically find datasets in formats like CSV or Excel. Ensure that the dataset includes relevant fields such as date, cases, deaths, and countries/territories.

3. Load the Data:

Load the obtained dataset into a data analysis tool or programming environment like Python. You can use Pandas, a popular data manipulation library, to work with the data.

```
python

import pandas as pd

# Load the data into a Pandas DataFrame

df = pd.read_csv(' /content/Covid_19_cases4..csv')
```

4. Initial Data Exploration:

Begin by exploring the dataset to understand its structure:

- Use `df.head()` to view the first few rows.

- Check for missing values with `df.isnull().sum()`.

- Inspect data types and statistics with `df.info()`.

5. Data Preprocessing:

Clean and preprocess the data to ensure accuracy and reliability. Common preprocessing steps include:

- Converting date columns to datetime objects.

- Handling missing values by filling, interpolating, or removing them.

- Addressing data inconsistencies or outliers.

- Aggregating data by day or other relevant time intervals.

6. Save Preprocessed Data:

After cleaning and processing, save the preprocessed data to a new file:

python

```
df.to_csv('preprocessed_covid_data.csv', index=False)
```

This preprocessed dataset will be more suitable for analysis, visualization, and drawing insights while ensuring data accuracy and reliability.

The steps mentioned here provide a starting point for your data analysis project. Adjust the objectives and data processing steps according to your specific research goals and dataset.

Analysis Objectives:

The analysis objectives for this COVID-19 data project are as follows:

1. Daily Trends Analysis:To understand the daily trends of COVID-19 cases and associated deaths in the European Union/European Economic Area (EU/EEA) region. This analysis will provide insights into the progression of the pandemic over time.

2. Comparative Analysis: To compare and contrast the mean values and standard deviations of daily COVID-19 cases and deaths across different countries and territories within the EU/EEA. This will allow for the identification of countries that experienced unique trends and variations.

3. Pattern Detection: To detect any significant patterns, spikes, or anomalies in the data that may require further investigation. Identifying these patterns can provide insights into the impact of interventions, vaccination campaigns, or public health measures.

4. Impact Assessment: To assess the potential impact of various measures and interventions, such as lockdowns, mask mandates, and vaccination campaigns, on the trajectory of the pandemic. This analysis will help in understanding the effectiveness of these measures.

Data Collection and Processing:

1. Data Source: Obtain a reliable and up-to-date COVID-19 dataset from trusted sources such as government health agencies, international organizations, or reputable data providers. This dataset should include key information such as date, daily cases, daily deaths, and the corresponding countries and territories.

2. Data Loading: Utilize data manipulation tools like Pandas in Python to load the dataset into a structured format for analysis.

3. Data Cleaning: Clean the data to ensure accuracy and reliability. Steps may include:

- Converting date columns to the correct datetime format.
- Handling missing data through imputation or removal.
- Addressing data inconsistencies, outliers, and anomalies.

4. Data Aggregation: Depending on the analysis objectives, aggregate the data into relevant time intervals (e.g., daily, weekly, or monthly) to identify overarching trends and patterns.

5. Data Validation:Cross-verify the dataset with multiple sources to validate data accuracy and consistency.

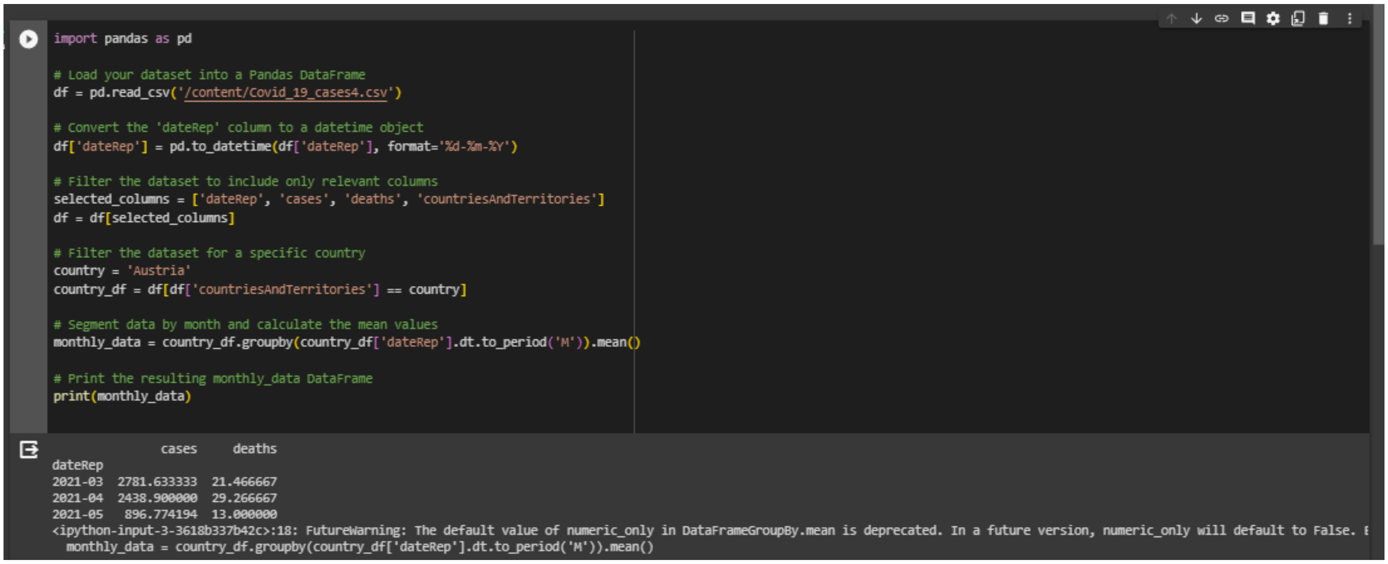
6. Data Version Control:Keep records of different versions of the dataset, especially if the data evolves over time due to updates.

By setting clear analysis objectives and ensuring data accuracy and reliability through proper data collection and cleaning, this project will facilitate a meaningful exploration of the COVID-19 situation in the EU/EEA region and contribute to our understanding of the pandemic's trajectory and the effectiveness of interventions.

Program

```
import pandas as pd

# Load your dataset into a Pandas DataFrame
df = pd.read_csv('/content/Covid_19_cases4.csv')

# Convert the 'dateRep' column to a datetime object
df['dateRep'] = pd.to_datetime(df['dateRep'], format='%d-%m-%Y')

# Filter the dataset to include only relevant columns
selected_columns = ['dateRep', 'cases', 'deaths', 'countriesAndTerritories']
df = df[selected_columns]

# Filter the dataset for a specific country
country = 'Austria'
country_df = df[df['countriesAndTerritories'] == country]

# Segment data by month and calculate the mean values
monthly_data = country_df.groupby(country_df['dateRep'].dt.to_period('M')).mean()

# Print the resulting monthly_data DataFrame
print(monthly_data)
```

output



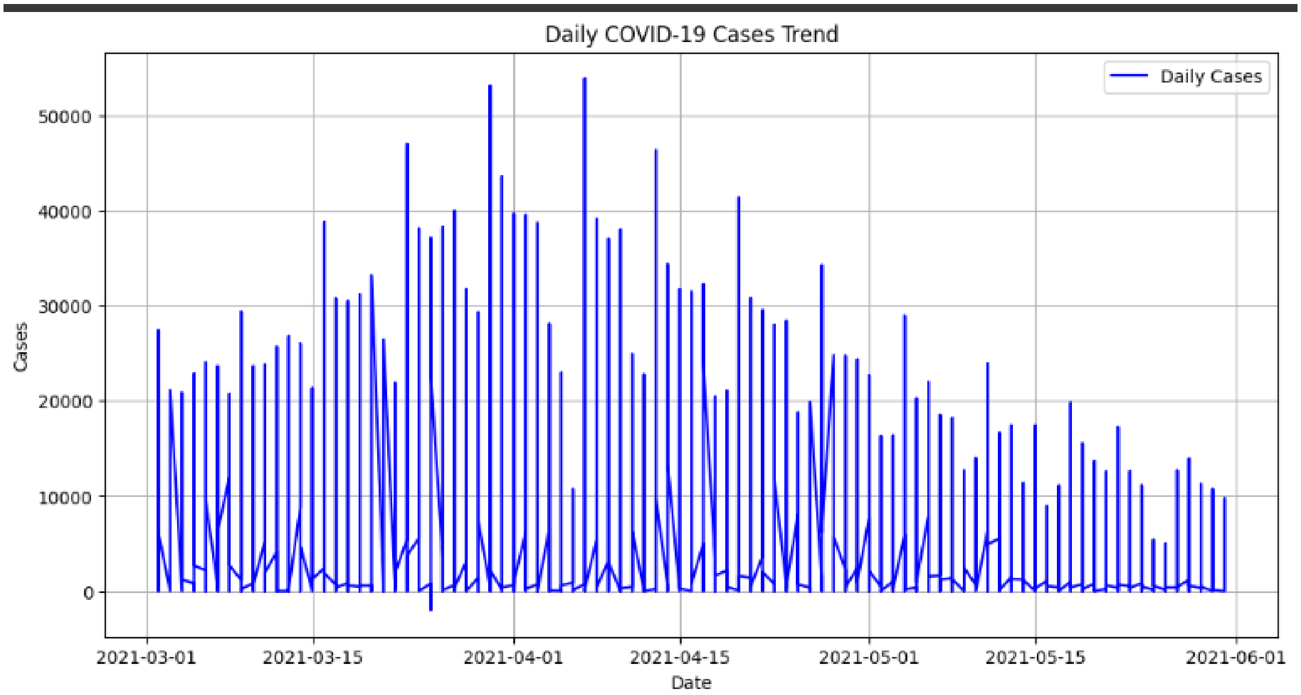
```
import pandas as pd

# Load your dataset into a Pandas DataFrame
df = pd.read_csv('/content/Covid_19_cases4.csv')

# Convert the 'dateRep' column to a datetime object
df['dateRep'] = pd.to_datetime(df['dateRep'], format='%d-%m-%Y')

# Filter the dataset to include only relevant columns
selected_columns = ['dateRep', 'cases', 'deaths', 'countriesAndTerritories']
df = df[selected_columns]

# Filter the dataset for a specific country
country = 'Austria'
country_df = df[df['countriesAndTerritories'] == country]

# Segment data by month and calculate the mean values
monthly_data = country_df.groupby(country_df['dateRep'].dt.to_period('M')).mean()

# Print the resulting monthly_data DataFrame
print(monthly_data)
```

```
               cases     deaths
dateRep
2021-03  2781.633333  21.466667
2021-04  2438.900000  29.266667
2021-05   896.774194  13.000000
<ipython-input-3-3618b337b42c>:18: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. E
  monthly_data = country_df.groupby(country_df['dateRep'].dt.to_period('M')).mean()
```

Program 2

```
import pandas as pd
import matplotlib.pyplot as plt


# Load your dataset
df = pd.read_csv('/content/Covid_19_cases4.csv')


# Convert 'dateRep' to a datetime object
df['dateRep'] = pd.to_datetime(df['dateRep'], format='%d-%m-%Y')


# Sort the data by date
df = df.sort_values(by='dateRep')


# Create a line chart for daily COVID-19 cases
plt.figure(figsize=(12, 6))
plt.plot(df['dateRep'], df['cases'], label='Daily Cases', color='blue')
plt.xlabel('Date')
plt.ylabel('Cases')
plt.title('Daily COVID-19 Cases Trend')
plt.legend()
plt.grid(True)

# Display the chart
plt.show()
```

output


Daily COVID-19 Cases Trend
Daily Cases
Cases
Date