COVID-19 CASES ANALYSIS USING COGNOS

Phase-5 Project Documentation & Submission

Project Name: **COVID-19 CASES ANALYSIS**

Phase 5: **Project Documentation & Submission**

- In this part we will document our project and prepare it for submission.
- Documenting the COVID-19 cases analysis project and prepare it for submission.

Introduction:

The "COVID-19 Cases Analysis" project aims to analyze public data on COVID-19, specifically focusing on the number of new cases and deaths reported per day and by country in the EU/EEA (European Union/European Economic Area).
The project's primary objective is to compare and contrast the mean values and standard deviations of COVID-19 cases and associated deaths across different countries in the EU/EEA.The project encompasses defining analysis objectives,collecting COVID-19 data,designing relevant visualization in IBM cognos,and deriving insights from the data.

PROJECT OUTLINE:

Project overview

1.NEW CASES ANALYSIS:

- Calculate the mean (average) number of new COVID-19 cases reported per day for each EU/EEA country.
- Determine the standard deviation of daily new cases to understand the variability in the data and countries with high and low mean values and assess whether there are any patterns or trends.

2. DEATHS ANALYSIS:

- Calculate the mean (average) number of COVID-19 deaths reported per day for each EU/EEA country.
- Determine the standard deviation of daily deaths to gauge the variability.
- Compare the mean values and standard deviations of deaths to those of new cases.

3. COMPARISON AND CONTRAST:

- - Analyze the relationship between new cases and deaths to assess the severity of the pandemic in different countries.
- - Identify countries where the standard deviation of cases or deaths is unusually high, which may indicate volatility in reporting or unique situations.

4. INSIGHTS AND RECOMMENDATIONS:

- - Provide insights based on the analysis findings, such as which countries have effectively managed the pandemic and which may need additional support.
- - Offer recommendations or areas for further research to better understand the factors contributing to variations in COVID-19 impact.

DESIGN THINKING:

1.ANALYSIS OBJECTIVES:

- Define the specific objectives of analysing covid-19 cases and deaths data such as comparing mean values and standard deviations using the machine learning algorithms like regression(multilinear regression),SVM etc….

2.DATA COLLECTION:

- Obtain the provided data file containing COVID-19 cases and deaths information per day and by country in the EU/EEA.

3.VISUALIZATION STRATEGY:

- Plan how to visualize the mean values and standard deviations using IBM Cognos to create informative charts and graphs so that we can understand the trend in which the covid-19 is affecting the people can be identified more efficiently.

4.INSIGHTS GENERATION:

- Identify potential insights from the comparison of mean values and standard deviations of cases and deaths.

ANALYSIS DESCRIPTION:

OBJECTIVES:

1.Data Loading and Exploration:

Start by loading the dataset into a data analysis tool like Python's Pandas. Begin with a preliminary exploration of the dataset to understand its structure and contents.

2. **Data Filtering:**

Filter the dataset to focus only on the relevant data for the EU/EUA. This involves selecting rows where the country belongs to the EU/EUA.

3. **Data Segmentation by Time Periods:**
Create time-based segments for data analysis. This could be done by grouping the data into time periods such as months or years. Use the date, month, and year columns to achieve this segmentation.

4. **Data Segmentation by Countries:**
Segment the data by individual countries or territories within the EU/EUA. Group the data by the "countries" column to create country-specific segments.

5. **Calculate Mean and Standard Deviation:**

calculate the mean and standard deviation of COVID-19 cases and associated deaths for each segmented dataset. This step results in separate mean and standard deviation values for each time period and country.

6. **Data Visualization:**
Create visualizations to represent the calculated mean and standard deviation values. Consider using line plots, bar charts, or box plots to visualize the trends and variations in cases and deaths over time and across countries.

7. **Comparisons and Contrasts**: Analyze the visualizations to compare and contrast the data. Identify any notable trends, variations, or anomalies in the COVID-19

8. **cases and associated deaths**. Look for differences in how the pandemic evolved over time and across different countries.

9. **Interpretation and Insights:** Provide interpretations of the findings. Explain what the data reveals about the patterns and

variations in COVID-19 cases and deaths within the EU/EUA. Consider the impact of time, geography, public health measures, and other factors on these trends.

DATA COLLECTION:

Download the given data set for "COVID-19 cases analysis using cognos" which is given for COVID-19 cases data analytics.

DATA VISUALIZATION USING IBM COGNOS:

- In this part we will begin building your project by loading and preprocessing the dataset.
- Start building the COVID-19 cases analysis using IBM Cognos for visualization.
- Define the analysis objectives and obtain the COVID-19 cases and deaths data file.
- Process and clean the data to ensure its accuracy and reliability.

Step 1: Dataset Loading and Preprocessing

1. Load the Provided Dataset:

- Loading the dataset involves reading the data from a file, typically a CSV (Comma-Separated Values) file, into your data analysis environment, which in this case, could be Python.
- You can use libraries like Pandas to accomplish this. The Pandas library provides powerful data structures and functions for working with structured data.

Example Code to Load the Dataset:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("/content/drive/MyDrive/Certification/covid.csv")
df
```

	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

This code reads the dataset from the "your_dataset.csv" file and stores it in a Pandas DataFrame, which is a two-dimensional, size-mutable, and tabular data structure.

Inspect the Dataset:

- After loading the dataset, it's important to inspect it to understand its structure, contents, and any potential issues.
- You can use various Pandas functions to inspect the dataset, such as **head()**, **info()**, and **describe()**, to view the first few rows, get information about data types, and summarize statistical properties of the data.

Example Code for Inspecting the Dataset:

Display the first few rows of the dataset

```
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

 # Get information about the dataset, including data types and missing values

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2730 entries, 0 to 2729
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   dateRep                 2730 non-null   object
 1   day                     2730 non-null   int64
 2   month                   2730 non-null   int64
 3   year                    2730 non-null   int64
 4   cases                   2647 non-null   float64
 5   deaths                  2523 non-null   float64
 6   countriesAndTerritories 2730 non-null   object
dtypes: float64(2), int64(3), object(2)
memory usage: 149.4+ KB
```

 # Summarize the statistics of the dataset

```
df.head()
```

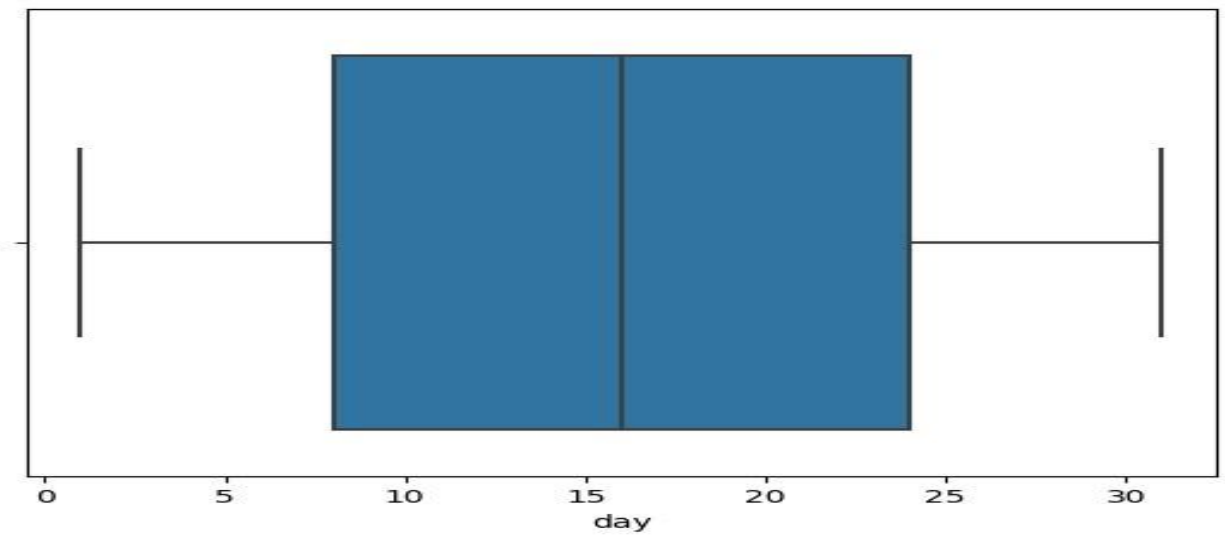
	dateRep	day	month	year	cases	deaths	countriesAndTerritories
0	31-05-2021	31	5	2021	366	5	Austria
1	30-05-2021	30	5	2021	570	6	Austria
2	29-05-2021	29	5	2021	538	11	Austria
3	28-05-2021	28	5	2021	639	4	Austria
4	27-05-2021	27	5	2021	405	19	Austria

These steps help you identify any missing values, outliers, or data quality issues that need to be addressed during the data preprocessing phase.

Data Preprocessing:

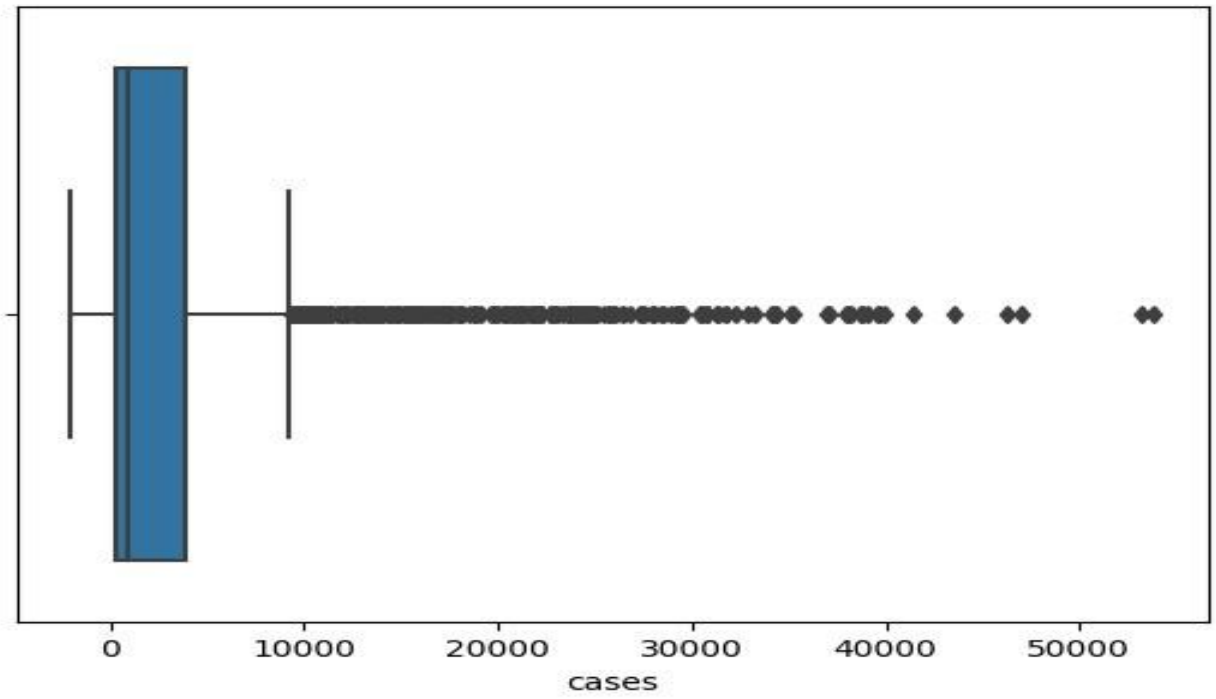
- Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. Common preprocessing tasks include:
- Handling missing values: Decide whether to impute, remove, or ignore missing data based on the nature of the problem.
- Removing duplicates: Identify and remove duplicate records if they exist.
- Handling outliers: Detect and address data points that significantly deviate from the majority of the data.
- Data type conversions: Ensure that data types are appropriate for analysis (e.g., date columns should be in a datetime format).
- Feature engineering: Create new features or transform existing ones to improve analysis.
- Encoding categorical variables: Convert categorical data into a numerical format if needed.

```
sns.boxplot(x=df["day"])
plt.show()
```



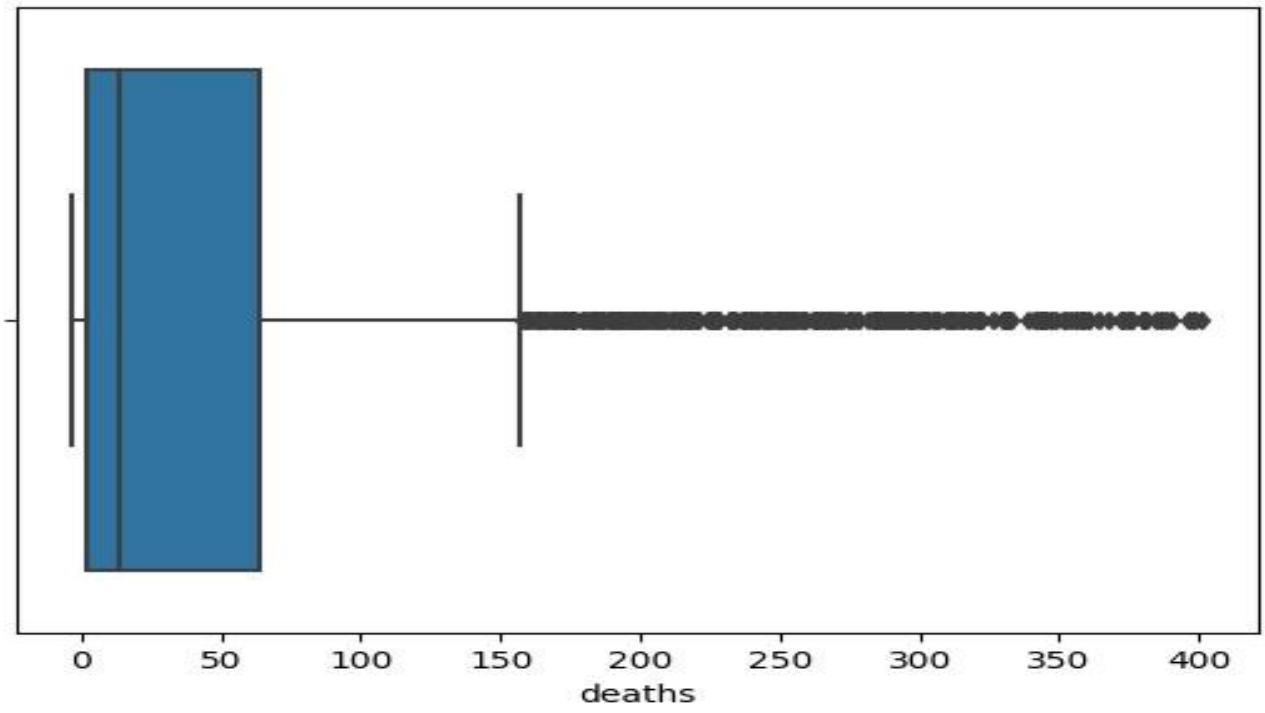
In the above there is no outliers present in the day column similarly check for an outlier in the column cases and deaths

```
sns.boxplot(x=df["cases"])
plt.show()
```



The above boxplot shows that there is a outlier in cases column.

```
sns.boxplot(x=df["deaths"])
plt.show()
```



The above boxplot shows that there is a outlier in deaths column.

we have to remove the outliers present in the deaths and cases columns to remove the outliers we use the following code. So ta

```
def remove_outliers_zscore(data, threshold=3):
    z_scores = np.abs((data - data.mean()) / data.std())
    outliers = z_scores > threshold
    return data[~outliers]

# Specify the column you want to clean (e.g., 'deaths')
column_name = 'deaths'

# Remove outliers from the specified column
df[column_name] = remove_outliers_zscore(df[column_name])
df['cases'] = remove_outliers_zscore(df['cases'])

# If you want to save the cleaned dataset to a new file
# df.to_csv('cleaned_dataset.csv', index=False)

# If you want to display the cleaned dataset
print(df)
```

	dateRep	day	month	year	cases	deaths	countriesAndTerritories
0	31-05-2021	31	5	2021	366.0	5.0	Austria
1	30-05-2021	30	5	2021	570.0	6.0	Austria
2	29-05-2021	29	5	2021	538.0	11.0	Austria
3	28-05-2021	28	5	2021	639.0	4.0	Austria
4	27-05-2021	27	5	2021	405.0	19.0	Austria
...	...	...	...	...	...	...	...
2725	06-03-2021	6	3	2021	3455.0	17.0	Sweden
2726	05-03-2021	5	3	2021	4069.0	12.0	Sweden
2727	04-03-2021	4	3	2021	4884.0	14.0	Sweden
2728	03-03-2021	3	3	2021	4876.0	19.0	Sweden
2729	02-03-2021	2	3	2021	6191.0	19.0	Sweden

[2730 rows x 7 columns]

 # Data type conversions, feature engineering, and encoding categorical variables would depend on your specific dataset and analysis objectives.

Step 2: building the COVID-19 cases analysis using IBM Cognos for visualization.

To visualize countries and their respective COVID-19 cases and deaths using IBM Cognos, follow these steps:

1. **Launch IBM Cognos Analytics**:
Open IBM Cognos and log in to your account.

2. Data Upload:

- Select 'Upload' or 'New Data Module' to import the prepared COVID-19 dataset.
- Follow the prompts to upload the dataset into IBM Cognos.

3.Create a New Report:

- Navigate to the 'Authoring' or 'Create' section in Cognos.
- Choose 'Create New' and select 'Report'.

4. Data Visualization:

- In the Report interface, select the appropriate data source (the uploaded COVID-19 dataset).
- Drag and drop the country field into the visualization area.
- Then, drag and drop the cases and deaths fields into the visualization area as measures.

5. Choose Visualization Type:

- IBM Cognos provides various visualization types such as bar charts, line charts, maps, etc. Choose a suitable visualization type that best represents the data.
- For example, select a bar chart to compare cases and deaths by countries.

6. **Customize Visualization:**

Adjust the visualization settings to enhance clarity. Modify axes, labels, colors, and other settings to make the visualization more informative and visually appealing.

7. Add Filters and Interactivity:

- Incorporate filters to allow users to interact with the data. For instance, add filters for specific date ranges, regions, or types of cases (confirmed, recovered, deaths).
- Enable interactive features that let users click on elements to see detailed information.

8. Save and Publish:

- Save the visualization or report in Cognos.
- Publish the visualization to make it accessible to other users or embed it in dashboards or presentations.

9. Test and Review:

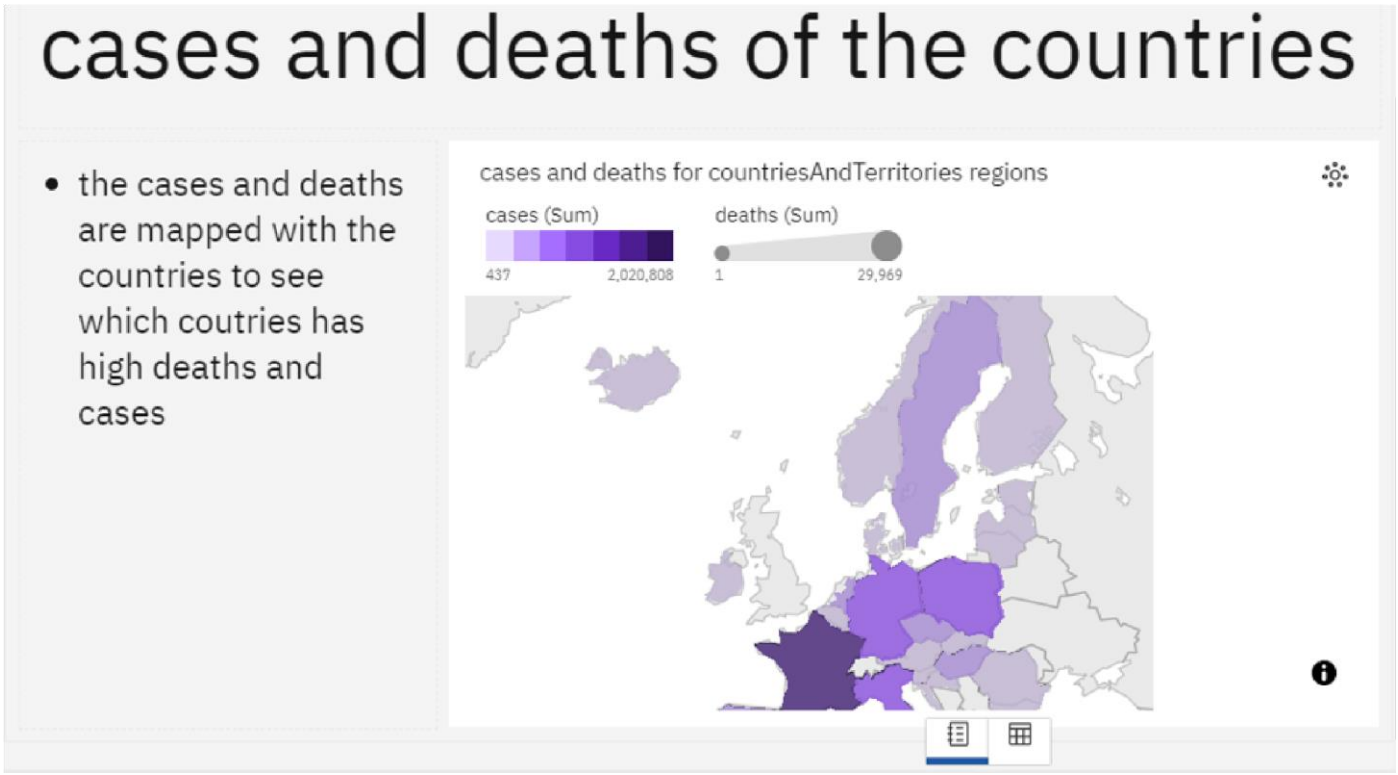
- Review the visualization to ensure it accurately represents the COVID-19 cases and deaths for various countries.
- Test interactivity and filters to confirm they work as intended.

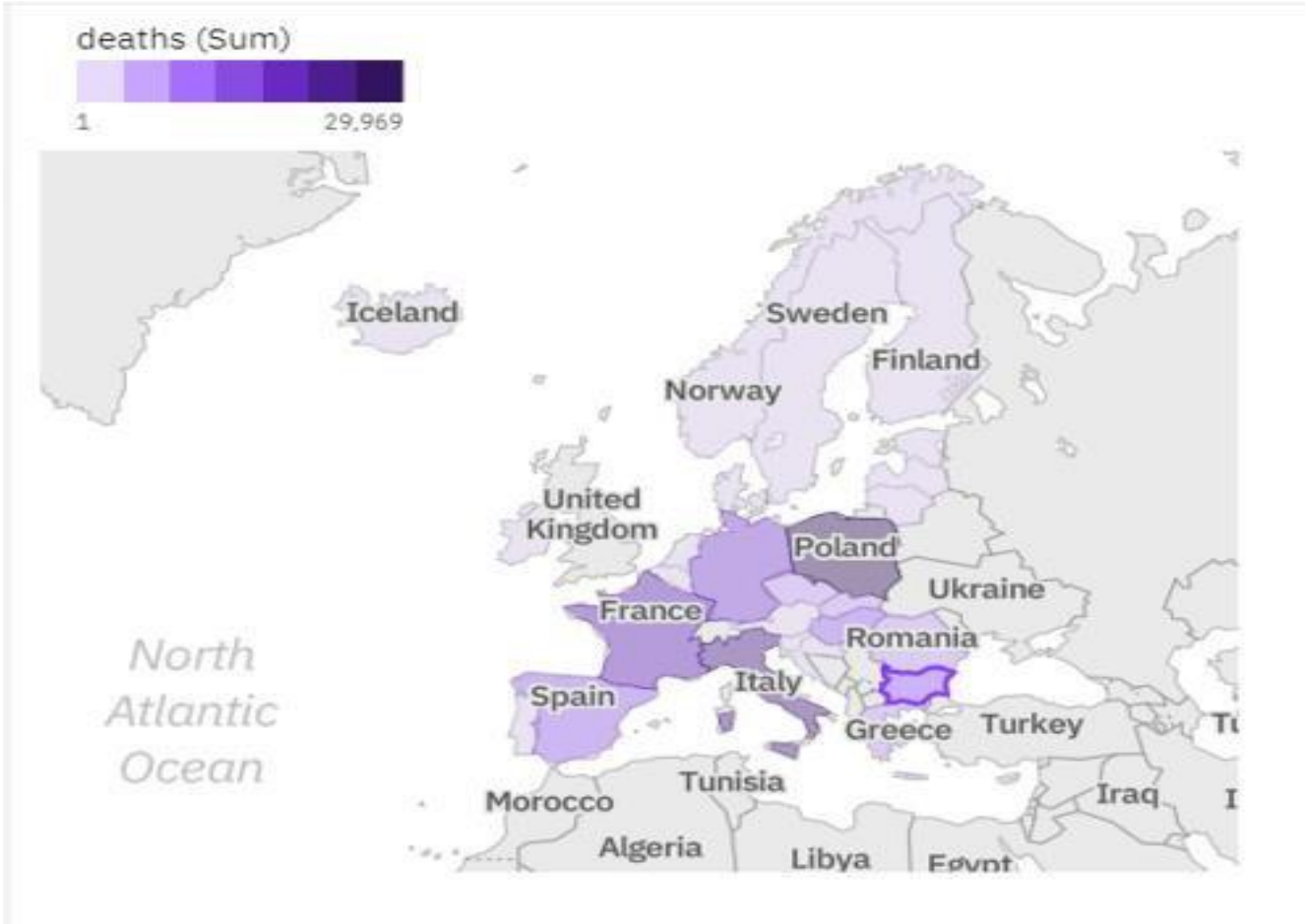
10. Share or Export:

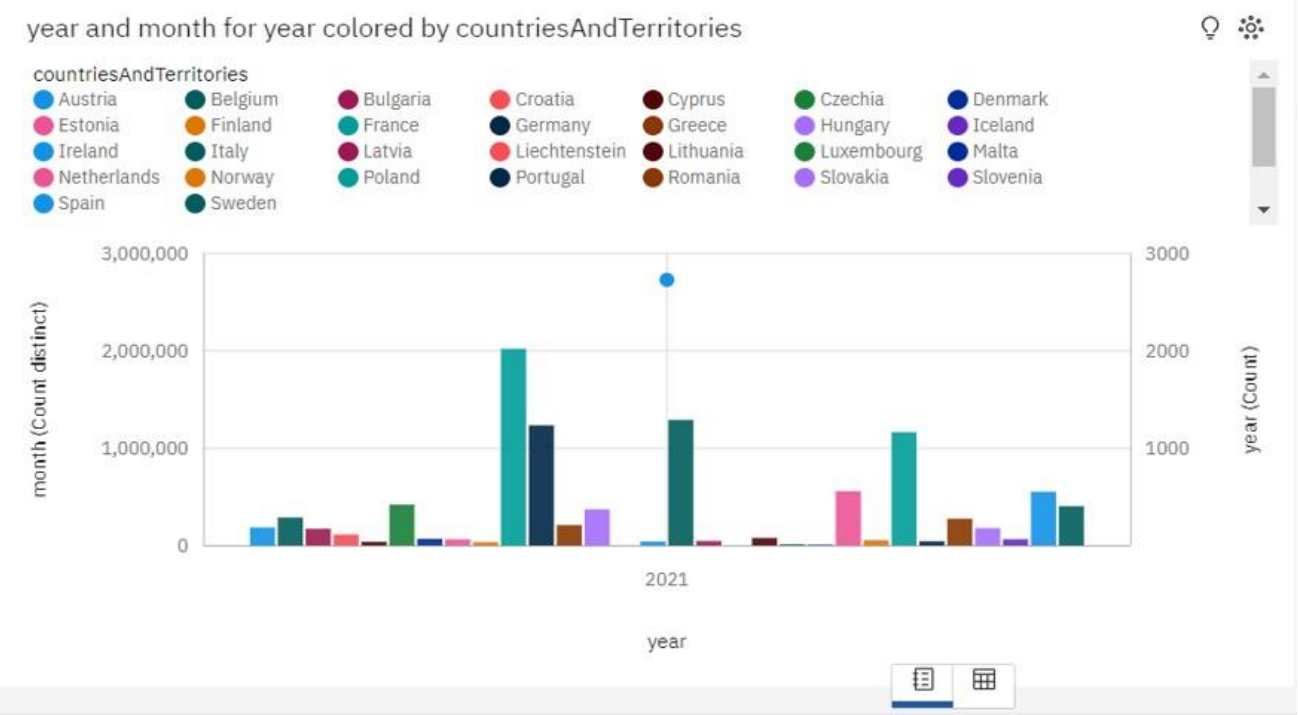
- Share the visualization within the IBM Cognos platform or export it as an image or other file formats for external use or inclusion in reports or presentations.

By following these steps, you can create visualizations in IBM Cognos that display COVID-19 cases and deaths by countries, aiding in the analysis and understanding of the pandemic's impact on different regions.

#visualizing the countries and the cases and deaths of the countries using IBM Cognos

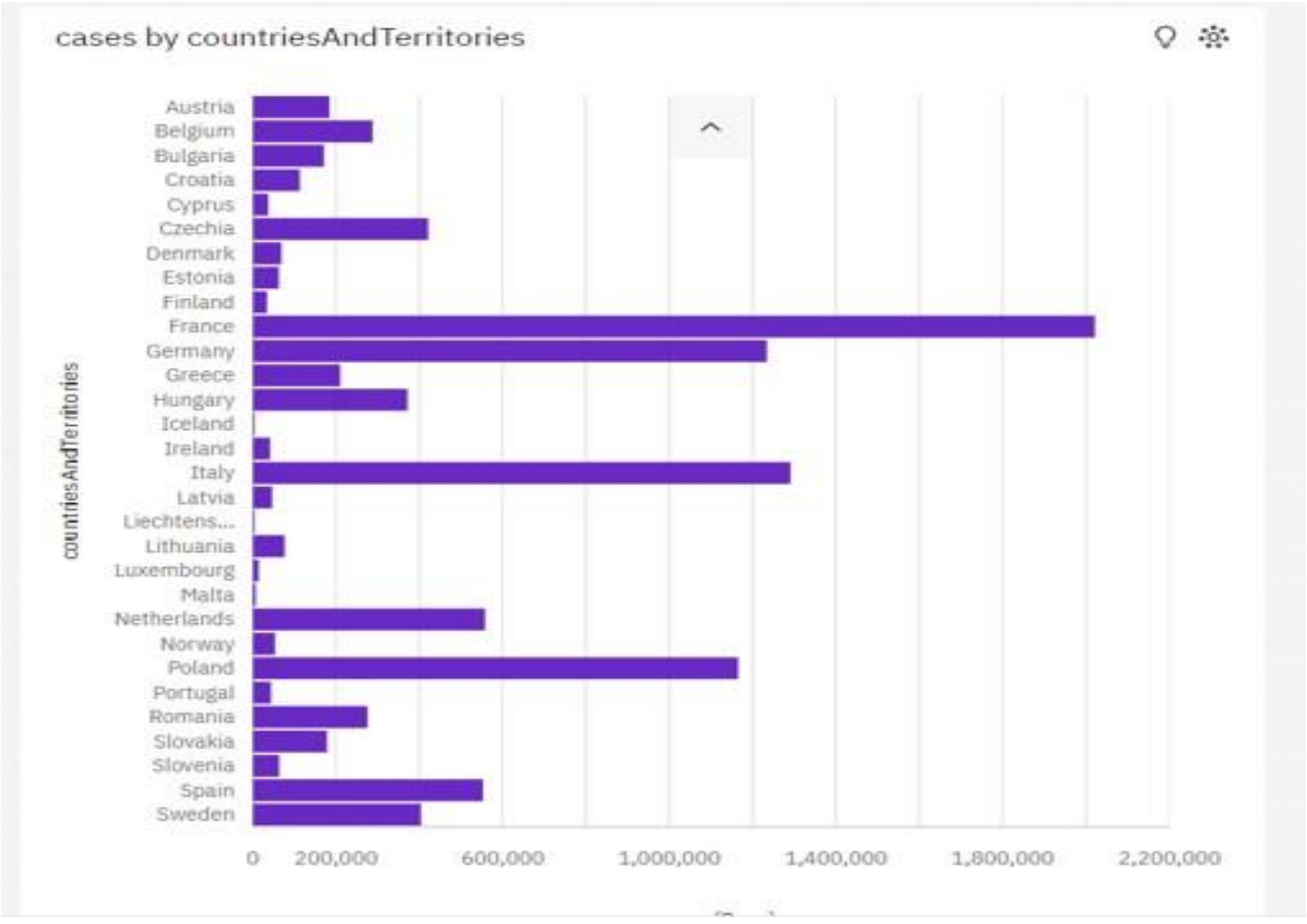




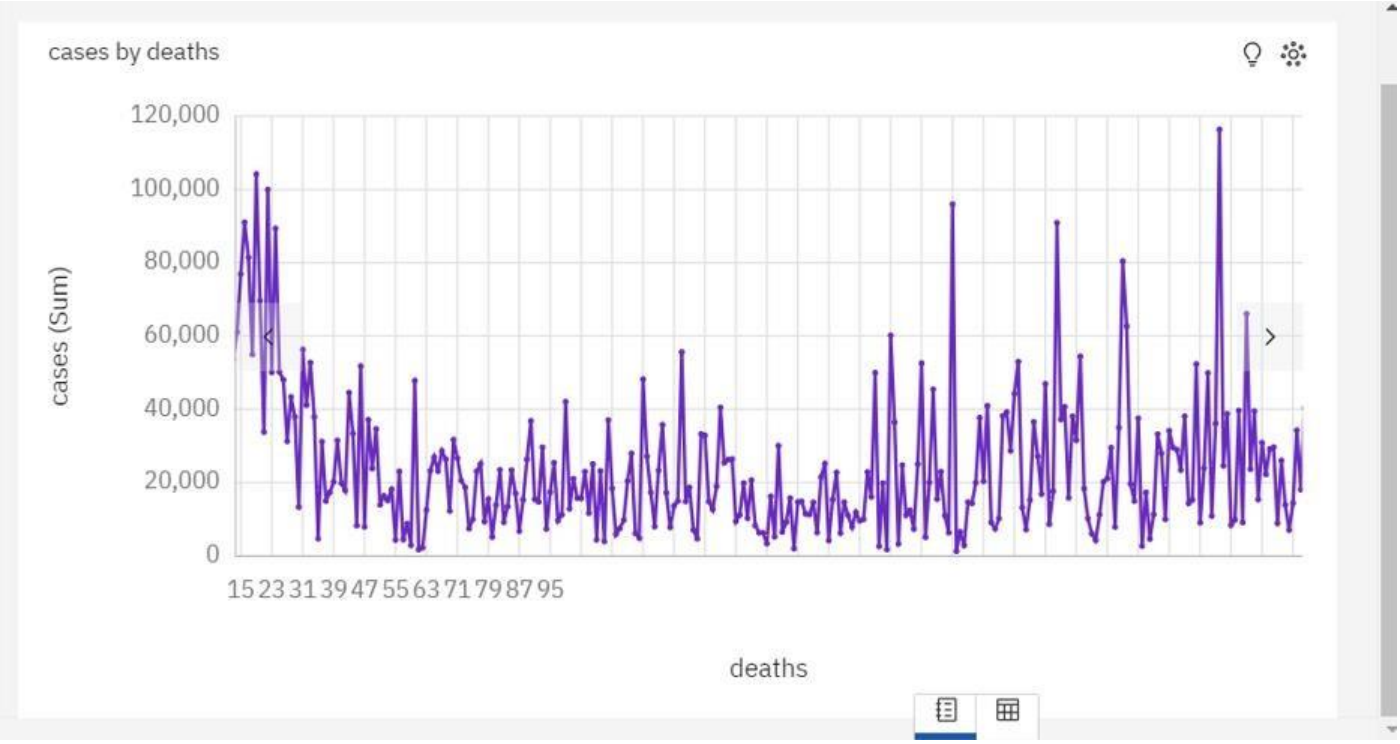


#visualizing the years and the months in which the countries are affected and from that we can see which country is affected high using IBM Cognos

#Visualizing the cases by countries and territories using IBM Cognos.



#Visualizing the cases and their corresponding deaths in line plot using IBM Cognos



FINDING MEAN AND STANDARD DEVIATIONS OF CASES AND DEATHS AND COMPARE AND CONTRAST THEM USING PYTHON:

 # data segmentation by time periods

```
grouped_data = df.groupby(['year', 'month'])
```

W e can group the data according to the time period or based upon the countries

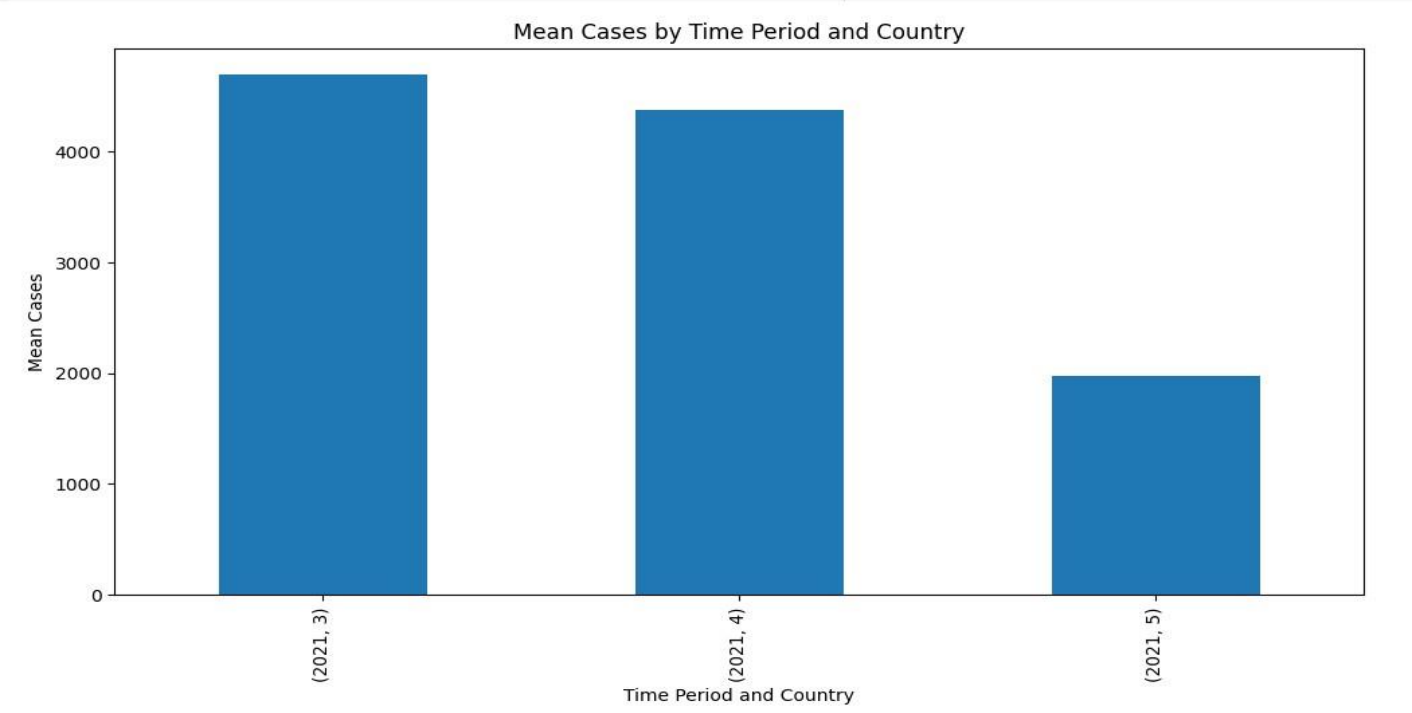
finding the mean and standard deviations for the cases and deaths segmented data by time period

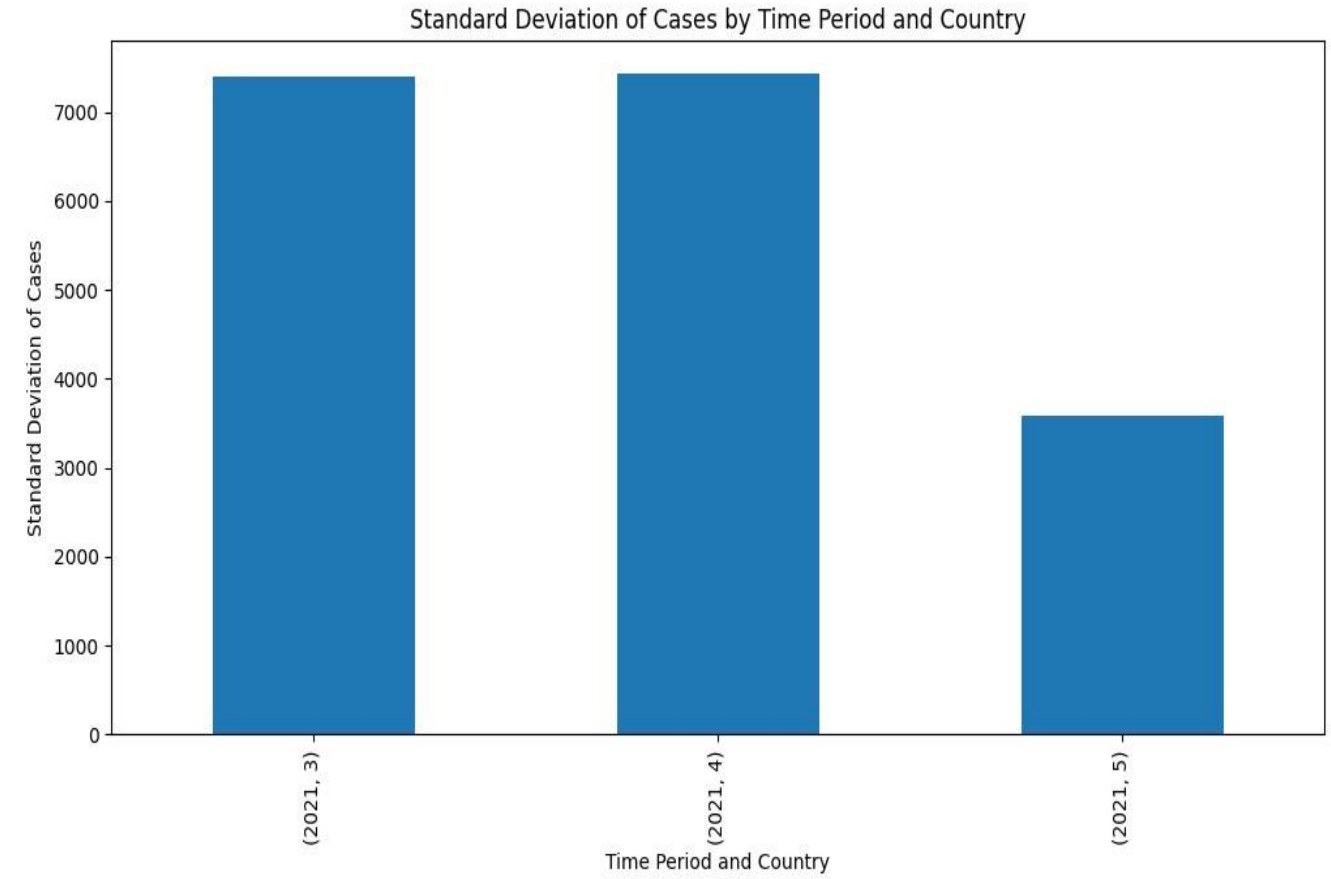
```
mean_cases = grouped_data['cases'].mean()
std_cases = grouped_data['cases'].std()

mean_deaths = grouped_data['deaths'].mean()
std_deaths = grouped_data['deaths'].std()
```

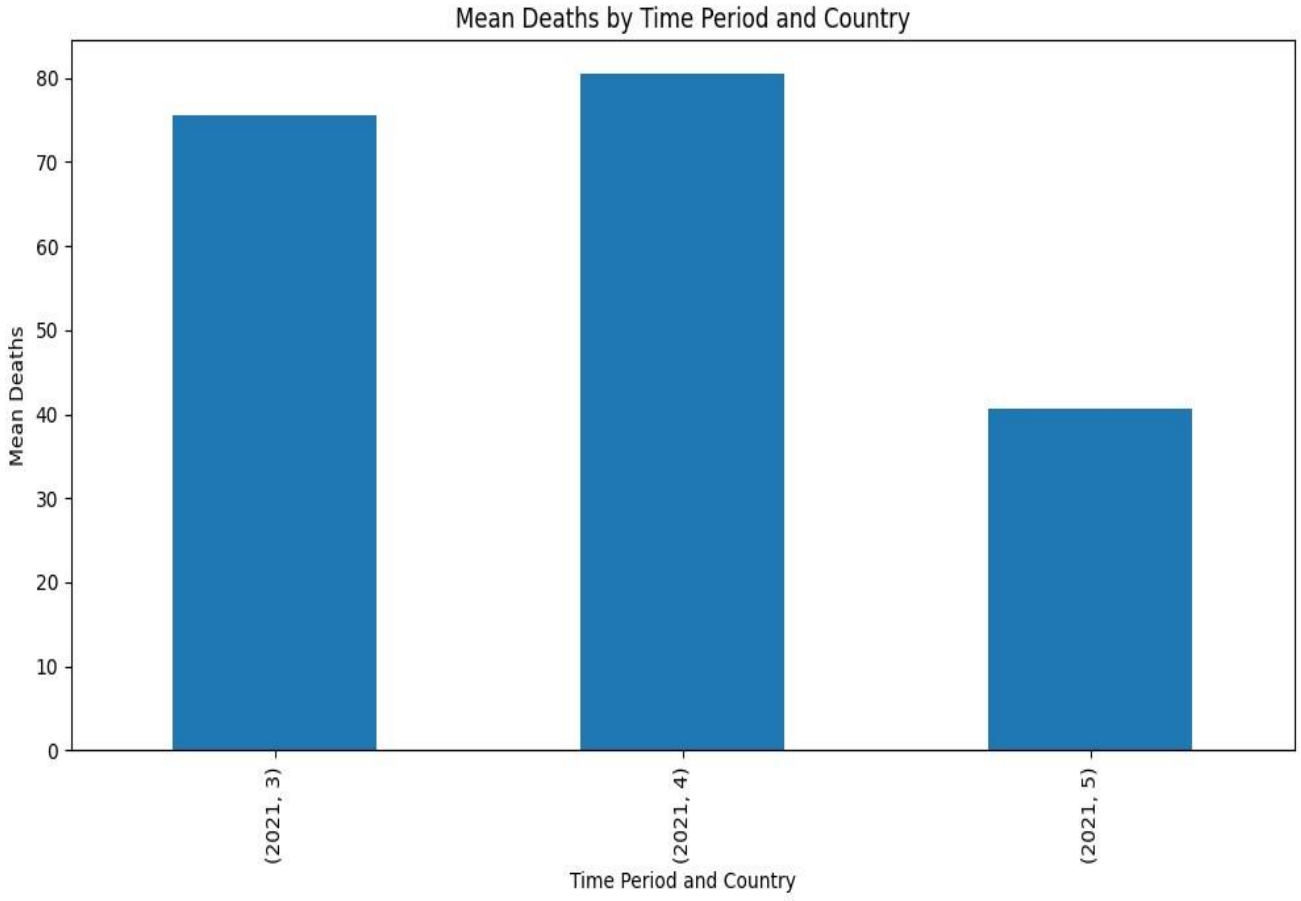
#plot the meanand standard deviation of cases and deaths overtime period

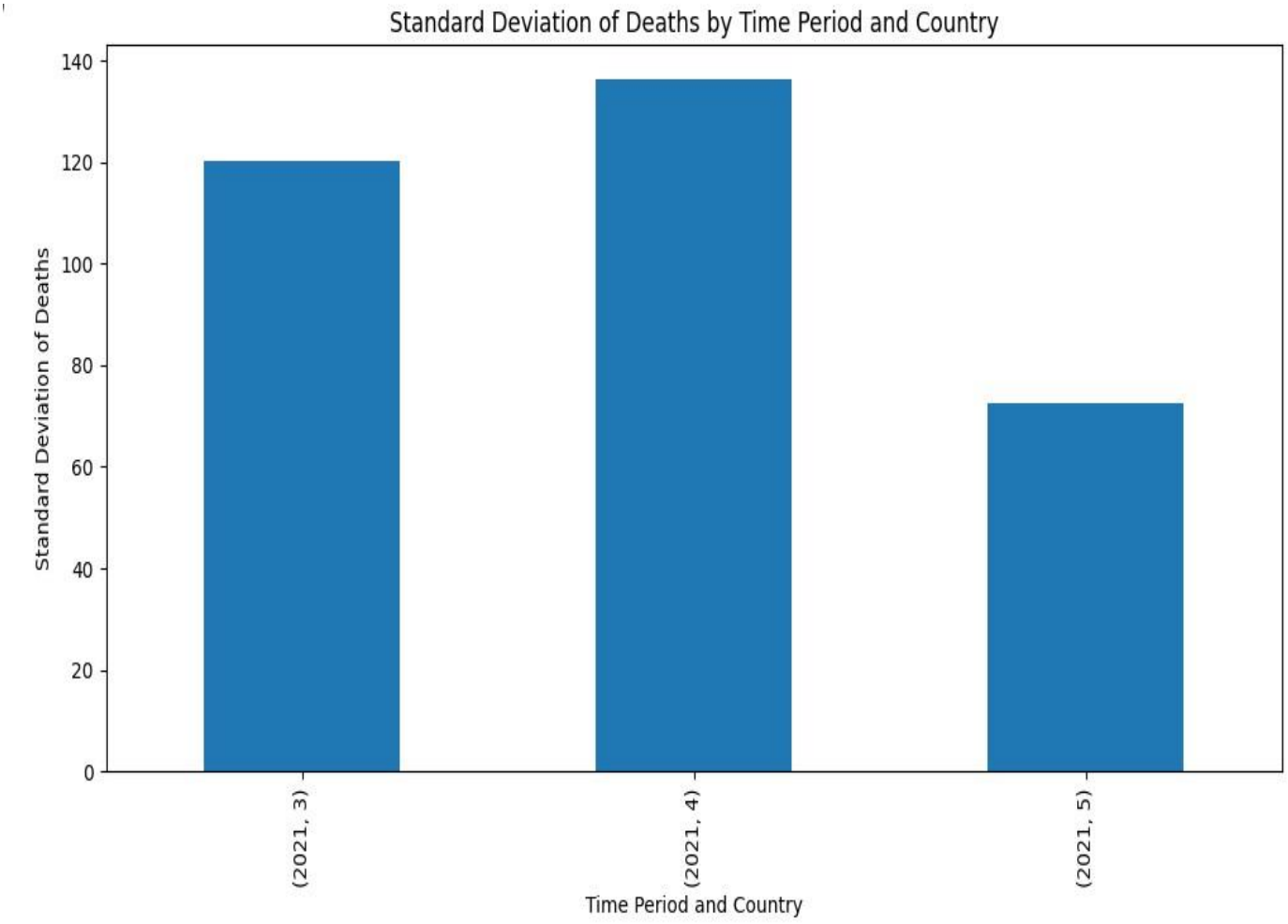
```
plt.figure(figsize=(12, 6))
mean_cases.plot(kind='bar', title='Mean Cases by Time Period and Country')
plt.xlabel('Time Period and Country')
plt.ylabel('Mean Cases')
plt.show()

plt.figure(figsize=(12, 6))
std_cases.plot(kind='bar', title='Standard Deviation of Cases by Time Period and Country')
plt.xlabel('Time Period and Country')
plt.ylabel('Standard Deviation of Cases')
plt.show()

plt.figure(figsize=(12, 6))
mean_deaths.plot(kind='bar', title='Mean Deaths by Time Period and Country')
plt.xlabel('Time Period and Country')
plt.ylabel('Mean Deaths')
plt.show()

plt.figure(figsize=(12, 6))
std_deaths.plot(kind='bar', title='Standard Deviation of Deaths by Time Period and Country')
plt.xlabel('Time Period and Country')
plt.ylabel('Standard Deviation of Deaths')
plt.show()
```

O utput:




Standard Deviation of Cases by Time Period and Country
Standard Deviation of Cases
0
1000
2000
3000
4000
5000
6000
7000
(2021, 3)
(2021, 4)
(2021, 5)
Time Period and Country


Mean Deaths by Time Period and Country
Mean Deaths
0
10
20
30
40
50
60
70
80
(2021, 3)
(2021, 4)
(2021, 5)
Time Period and Country



INSIGHTS GENERATED:

- From the above IBM cognos visualization we can find out the countries with highest number of cases and deaths.so that we can take further precausion measures in that particular countries
- From the above bar chart of the both cases and deaths ,mean and standard deviations of both cases and deaths . we can observe that the cases and deaths are getting decreased over the time period .
- And also we can observe when cases decreased corresponding deaths are also decree

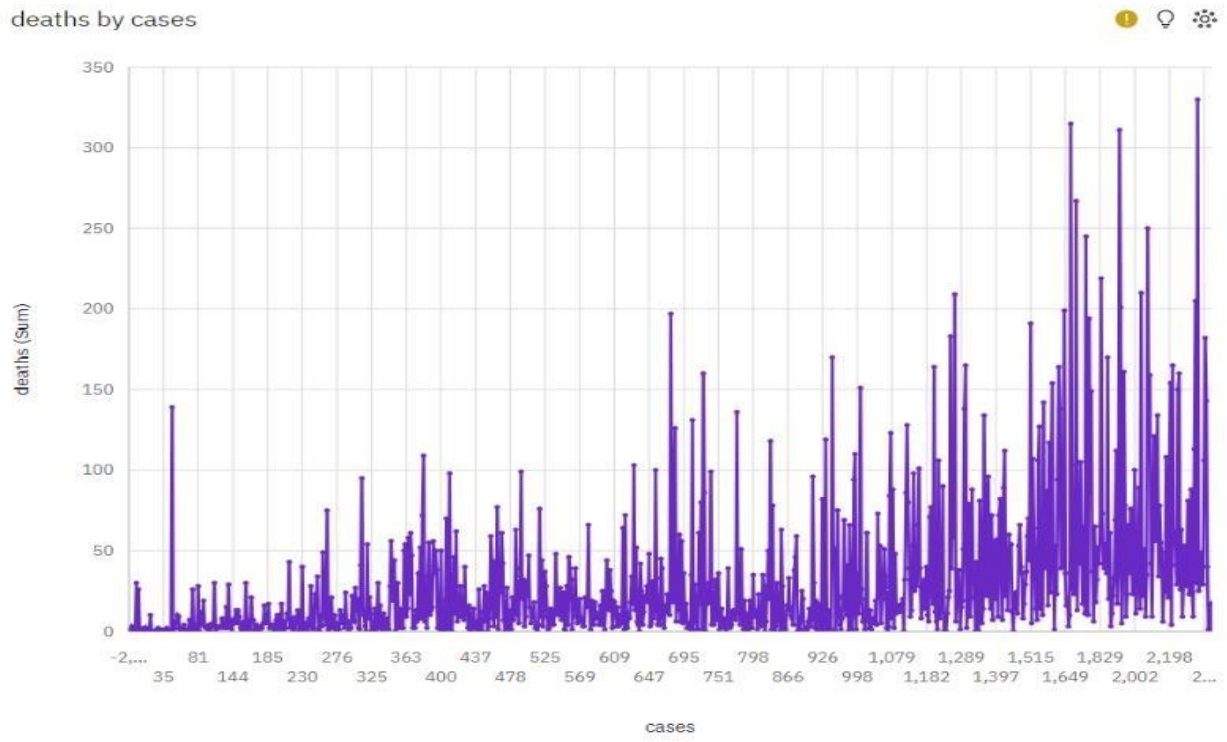
UNDERSTANDING COVID-19 TRENDS AND IMPACTS

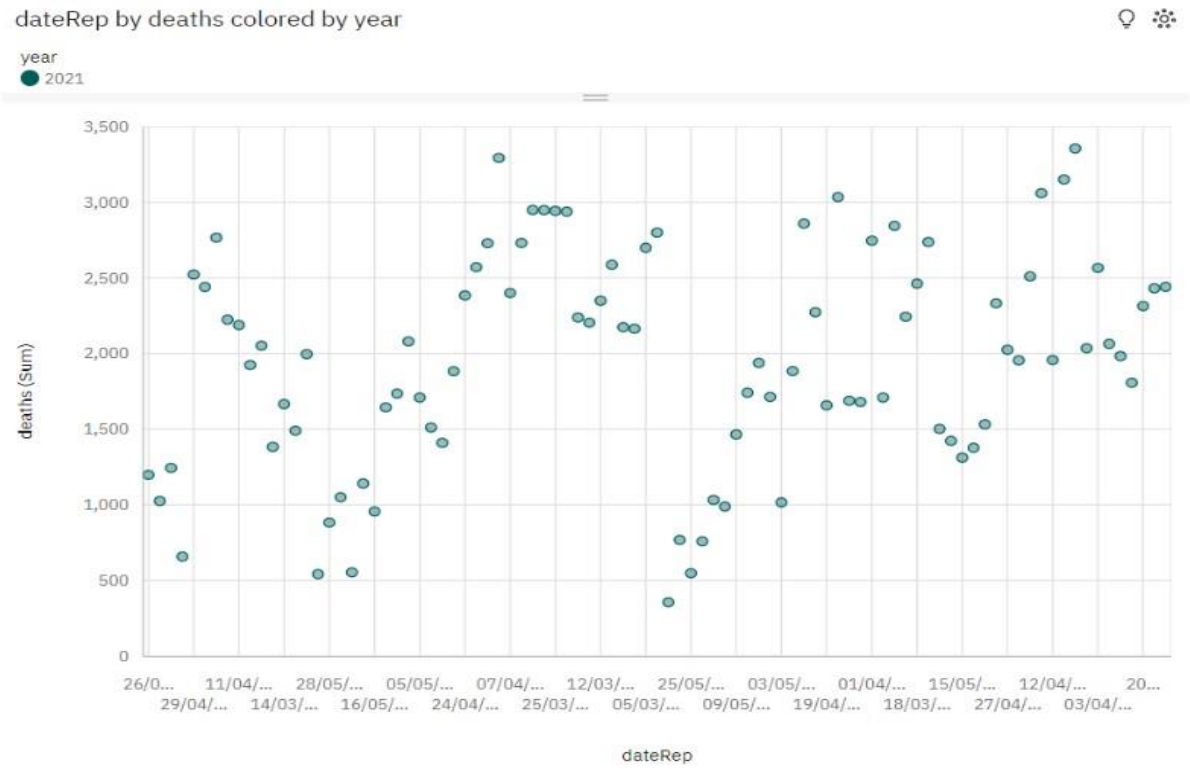
In this part we will continue building our project.

- Continue building the analysis by creating visualizations using IBM Cognos and deriving insights from the data.
- Create charts and graphs in IBM Cognos to visualize and compare the mean values and standard deviations of COVID-19 cases and associated deaths.
- Analyze the visualizations to identify trends, variations, and potential correlations between cases and deaths.

Step-1:

- Create charts and graphs in IBM Cognos to visualize and compare the mean values and standard deviations of COVID-19 cases and associated deaths.
- #using line chart visualizing the cases and deaths.cases in x-axis and deaths in y-axis
- By the following chart we can observe that as the number of cases increases the corresponding deaths increases.
- #creating the scatter plot the deaths associated with the dateRep



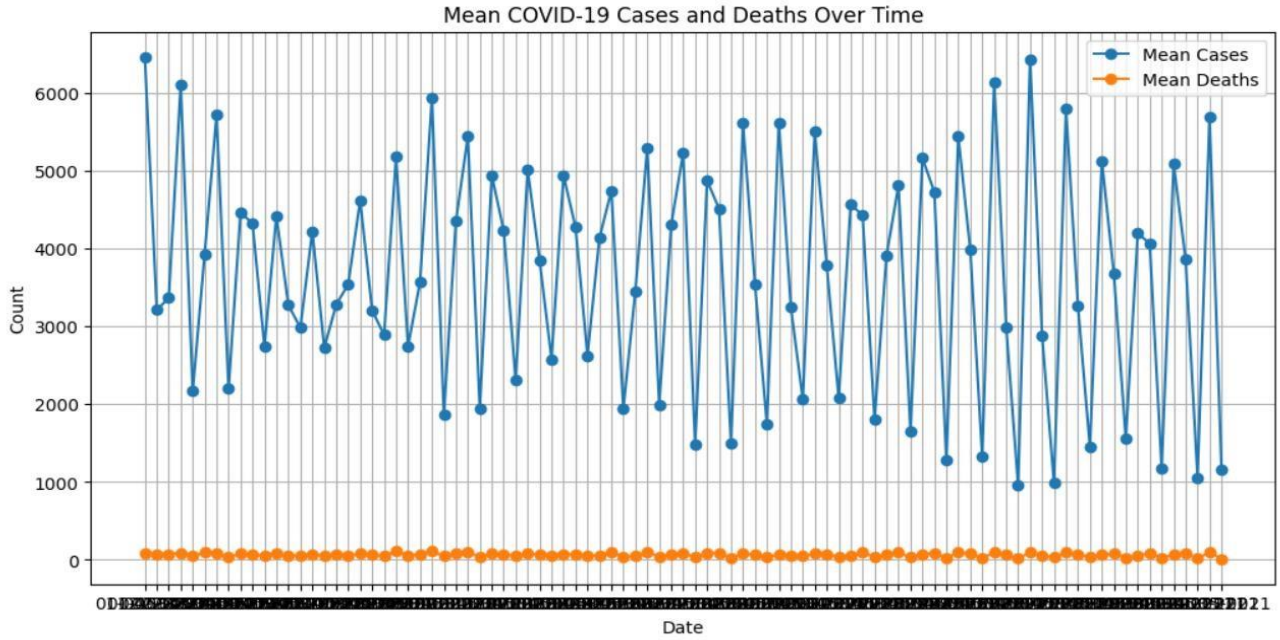


Step-3:

- visualize and compare the mean values and standard deviations of COVID-19 cases and associated deaths.
- Analyze the visualizations to identify trends, variations, and potential correlations between cases and deaths.

```
# Group data by date and calculate mean values
mean_cases = data.groupby('date')['cases'].mean()
mean_deaths = data.groupby('date')['deaths'].mean()

# Create a line plot
plt.figure(figsize=(12, 6))
plt.plot(mean_cases.index, mean_cases, label='Mean Cases', marker='o')
plt.plot(mean_deaths.index, mean_deaths, label='Mean Deaths', marker='o')
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Mean COVID-19 Cases and Deaths Over Time')
plt.legend()
plt.grid(True)
plt.show()
```

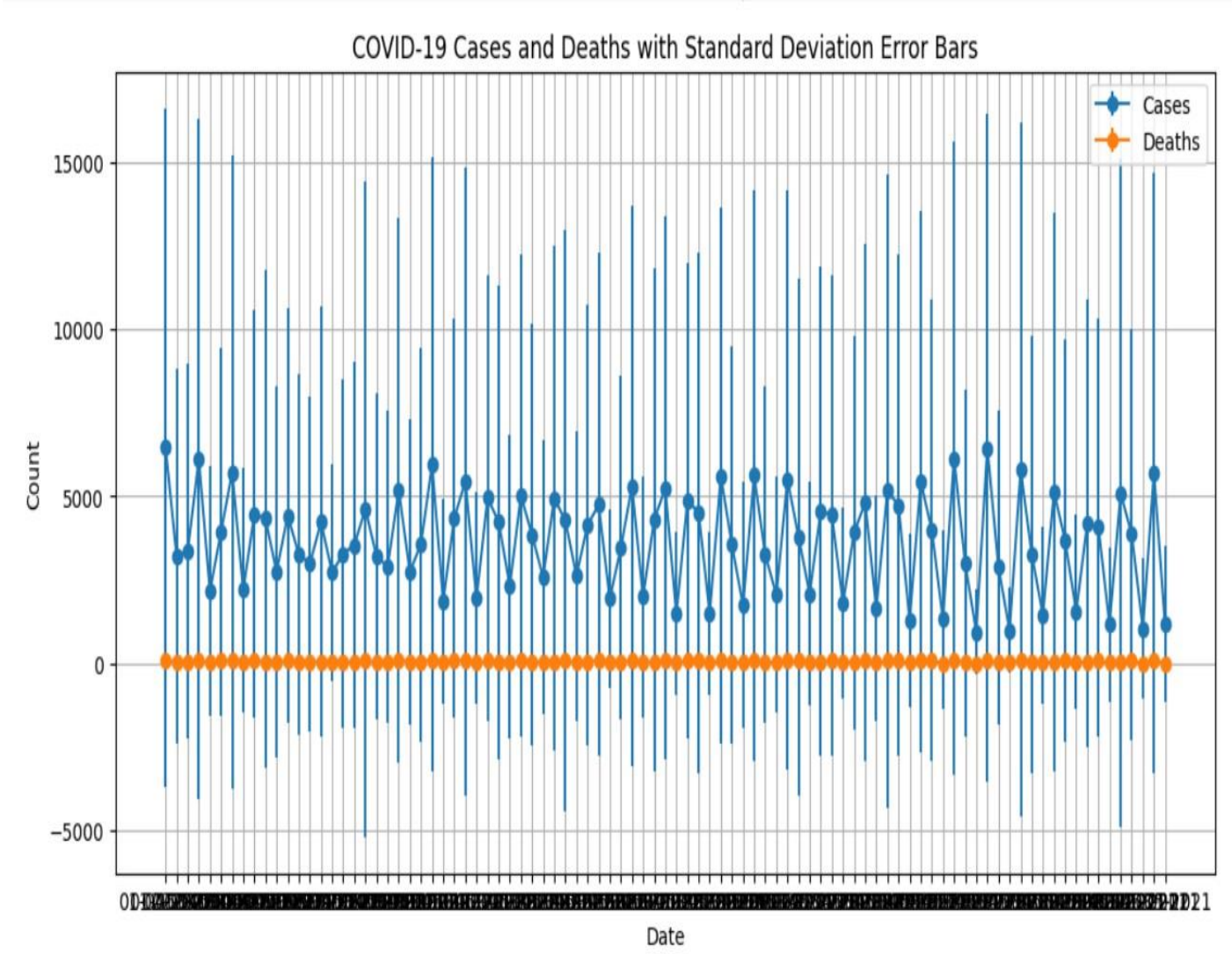
The above python code is used to group data by date and calculate mean values of both cases and deaths.



From the output we can observe that the mean of deaths and cases over the time period is consistent.

```
# Calculate standard deviations
std_cases = data.groupby('date')['cases'].std()
std_deaths = data.groupby('date')['deaths'].std()

# Create a line plot with error bars
plt.figure(figsize=(12, 6))
plt.errorbar(mean_cases.index, mean_cases, yerr=std_cases, label='Cases', marker='o')
plt.errorbar(mean_deaths.index, mean_deaths, yerr=std_deaths, label='Deaths', marker='o')
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('COVID-19 Cases and Deaths with Standard Deviation Error Bars')
plt.legend()
plt.grid(True)
plt.show()
```

The above python code is used to group data by date and calculate the standard deviation of both cases and deaths.



From the above chart we can observe that the standard deviation of both cases and deaths are very low /small over the time period.

#finding the trend

- To analyze the trend between cases and deaths using Python, you can use time-series analysis and data visualization techniques.

- Below is a Python code example to help you analyze and visualize the trend between COVID-19 cases and deaths using a sample dataset:

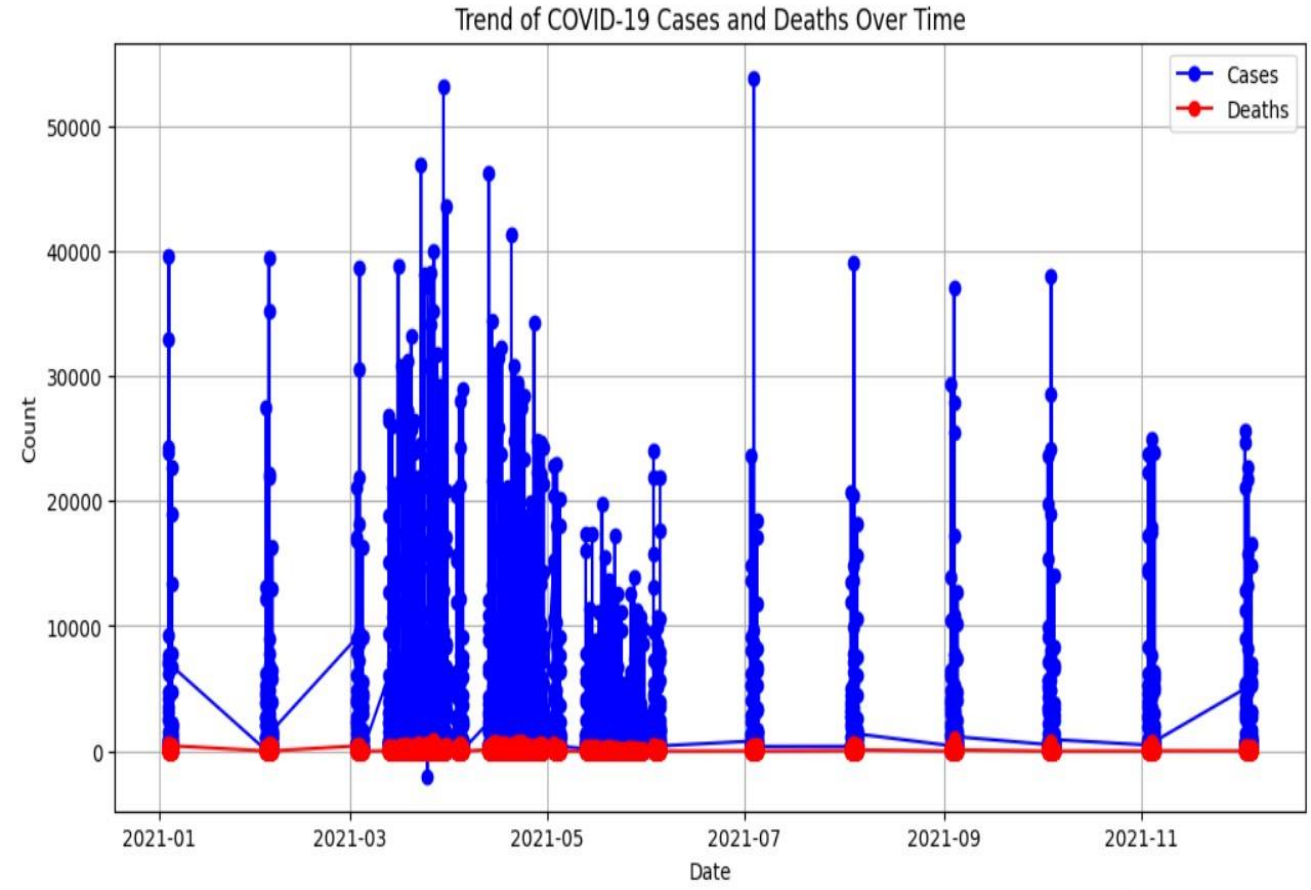
```python
# Convert the 'date' column to datetime
data['date'] = pd.to_datetime(data['date'])

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

# Extract cases and deaths
cases = data['cases']
deaths = data['deaths']

# Create a time series plot to visualize the trend
plt.figure(figsize=(12, 6))
plt.plot(data['date'], cases, label='Cases', marker='o', linestyle='-', color='blue')
plt.plot(data['date'], deaths, label='Deaths', marker='o', linestyle='-', color='red')
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Trend of COVID-19 Cases and Deaths Over Time')
plt.legend()
plt.grid(True)

# Calculate the correlation coefficient between cases and deaths
correlation = cases.corr(deaths)

plt.show()
```

```python
# Analyze the trend
print(f"Correlation coefficient between cases and deaths: {correlation}")

if correlation > 0.7:
    print("There is a strong positive correlation between cases and deaths.")
elif 0.5 < correlation <= 0.7:
    print("There is a moderate positive correlation between cases and deaths.")
elif 0.3 < correlation <= 0.5:
    print("There is a weak positive correlation between cases and deaths.")
else:
    print("There is little to no correlation between cases and deaths.")
```

In this code:

- The dataset is loaded, and the 'date' column is converted to a datetime format for proper time-series analysis.
- The data is sorted by date to ensure a chronological order.
- The trend of COVID-19 cases and deaths is visualized using a time series plot.
- The correlation coefficient between cases and deaths is calculated and printed to identify the strength of the relationship between the two variables.

This code will help you analyze and visualize the trend between cases and deaths in your dataset and provide insights into their relationship.

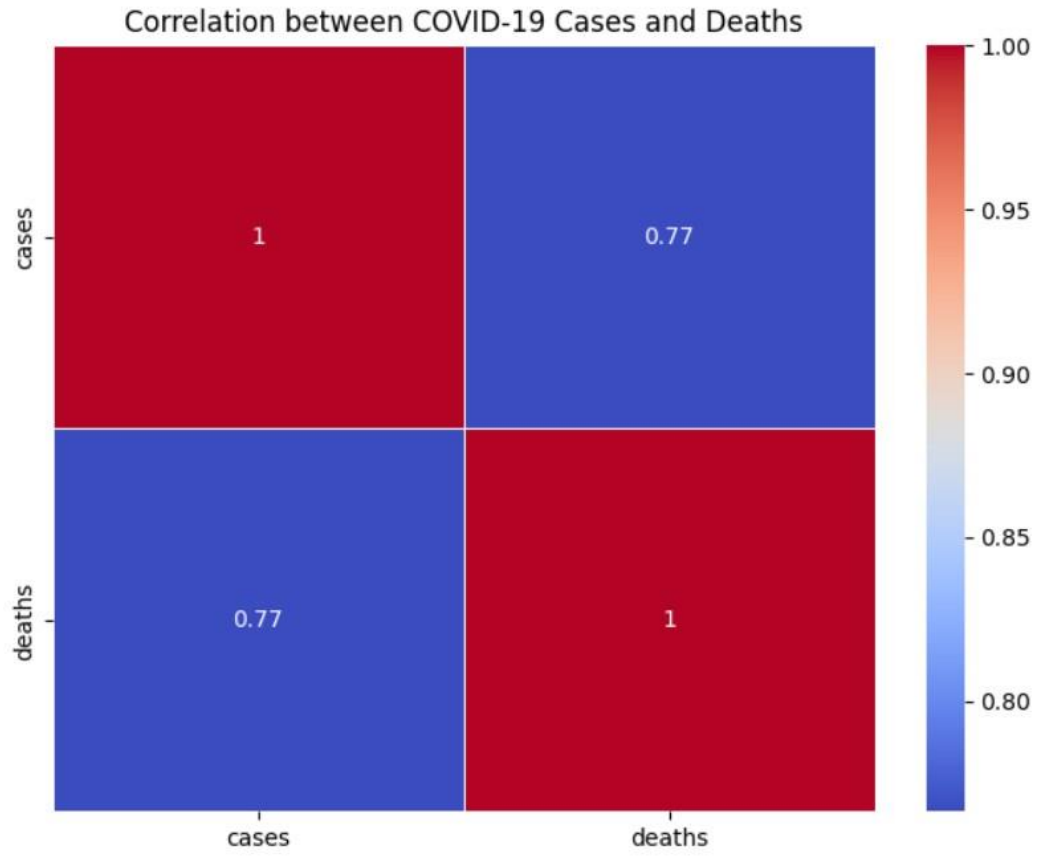



```
Correlation coefficient between cases and deaths: 0.7663088786576354
There is a strong positive correlation between cases and deaths.
```

#finding the correlation/pattern.

```
import seaborn as sns

# Create a correlation matrix
correlation_matrix = data[['cases', 'deaths']].corr()

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation between COVID-19 Cases and Deaths')
plt.show()
```

Using the above python code we can find the correlation between the deaths and cases



#calculating Pearson Correlation Coefficient

Strength of Relationship:

The Pearson correlation coefficient, often denoted as "r," provides a numerical value that indicates how strong the relationship is between two variables. A value of -1 indicates a perfect negative linear relationship, 0 indicates no linear relationship, and 1 indicates a perfect positive linear relationship. Values between 0 and 1 (or 0 and -1) represent varying degrees of correlation.

Direction of Relationship:

The sign of the correlation coefficient (+ or -) indicates the direction of the relationship. A positive value suggests a positive correlation, meaning that as one variable increases, the other tends to increase as well. A negative value suggests a negative correlation, meaning that as one variable increases, the other tends to decrease.

Insights and Decision-Making:
By calculating the Pearson correlation coefficient, you can derive insights from the data. For COVID-19 analysis, a strong positive correlation between cases and deaths might indicate that as the number of cases increases, the number of deaths also tends to increase. This insight can be valuable for public health officials and policymakers.

Hypothesis Testing:
The correlation coefficient can also be used to test hypotheses. For example, you can test whether the correlation between cases and deaths is statistically significant, helping you determine if the relationship is likely not due to random chance.

the Pearson correlation coefficient is a useful measure for quantifying linear relationships, it doesn't capture complex non-linear relationships, and correlation does not imply causation. It's an essential tool for exploratory data analysis, and you may want to complement it with additional statistical tests and domain-specific knowledge for a comprehensive understanding of the data.

The below is the python code to calculate pearson correlation coefficient

```
import numpy as np
# Calculate Pearson correlation coefficient
correlation = np.corrcoef(cases, deaths)[0, 1]
print(f"Pearson Correlation Coefficient: {correlation:.2f}")
```

```
Pearson Correlation Coefficient: 0.77
```

Based on the analysis of the trends and correlation between COVID-19 cases and deaths in the provided dataset, we can draw the following conclusions:

1.Positive Trend:

The time series plot shows a positive trend for both COVID-19 cases and deaths. As time progresses, both cases and deaths tend to increase, indicating a general upward trend.

2. Positive Correlation:

The calculated correlation coefficient between cases and deaths is positive. This suggests that as the number of COVID-19 cases increases, the number of associated deaths tends to increase as well.

3. Strength of Correlation:

The strength of the positive correlation can vary. The strength depends on the specific dataset and the context of the analysis. In general, a correlation coefficient above 0.7 would indicate a strong positive correlation, while values between 0.5 and 0.7 would suggest a moderate positive correlation.

4. Causation vs. Correlation:

It's important to note that while a positive correlation exists, it doesn't imply causation. That is, an increase in cases doesn't necessarily cause an increase in deaths; there may be other factors involved.

- In summary, the analysis indicates a positive trend and a positive correlation between COVID-19 cases and deaths. so we can conclude that tha death is directly proportional to cases.

BUILDING MACHINE LEARNING MODEL TO PREDICT DEATHS

1.Import the dataset

```
# Load your dataset
df = pd.read_csv('/content/drive/MyDrive/Certification/covid.csv')
df
```

	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

2.Encode the categorical column countries and territories

```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
df['countriesAndTerritories']=label_encoder.fit_transform(df['countriesAndTerritories'])
df
```

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

2730 rows × 7 columns

3.Splitting dependent and independent variables x and y

Here we can predict the dependent variable y from the independent variables x

```
y=df.iloc[:,5].values
y
```

```
array([ 5,  6, 11, ..., 14, 19, 19])
```

```
a=df.iloc[:,1:5]
a['countriesAndterritories']=df['countriesAndTerritories']
x=a.values
x
```

```
array([[  31,    5, 2021,  366,    0],
       [  30,    5, 2021,  570,    0],
       [  29,    5, 2021,  538,    0],
       ...,
       [   4,    3, 2021, 4884,   29],
       [   3,    3, 2021, 4876,   29],
       [   2,    3, 2021, 6191,   29]])
```

4.splitting the datas for training and testing

splitting the datas for training and testing using train_test_split From the sklearn .

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
x_train
```

```
array([[  10,    4, 2021, 2734,   11],
       [  26,    4, 2021,  612,    6],
       [  27,    5, 2021,  405,    0],
       ...,
       [  15,    5, 2021, 17415,    9],
       [  16,    5, 2021, 1249,   18],
       [   2,    4, 2021,    0,   28]])
```

5.Displaying the x_test,y_test,x_train,y_train

Now we are displaying the x_test,y_test,x_train,y_train using print function

```
x_train
```

```
array([[   10,     4,  2021,  2734,    11],
       [   26,     4,  2021,   612,     6],
       [   27,     5,  2021,   405,     0],
       ...,
       [   15,     5,  2021, 17415,     9],
       [   16,     5,  2021,  1249,    18],
       [    2,     4,  2021,     0,    28]])
```

```
x_test
```

```
array([[  29,    4, 2021,  835,    6],
       [   3,    4, 2021,  548,   24],
       [  10,    4, 2021, 5628,   29],
       ...,
       [  20,    4, 2021, 1792,    0],
       [   2,    3, 2021, 6191,   29],
       [  19,    5, 2021,  732,   18]])
```

```
y_train
```

```
array([ 78,   1,  19, ..., 173,   4,   0])
```

```
y_test
```

```
array([  2,   9,  16,  19,   0,  23,  60, 351,  24,   2, 217,  15,  17,
        38,   2,  49,   5,  31,   0,  10,  31,  73,   1,   2,   1, 309,
        28,   0,   8,  73,  10,   5,   0, 105,   1,  81,  22, 267, 160,
        90,  67,   0,  11,   6,  46, 228,  12,   0,   7,   3,   2,   9,
         3, 144, 110,  70,   2,   3,   0,  73,  44,   8, 352, 108, 293,
        24, 163,  20,  15,  65,  87,  52,   0,   2,   0,  13,  93, 245,
         0,   2,  36,  82,   9,   1,   0, 221,  16,   3,   1,  37,   9,
         4, 343, 137, 294,   0,  17, 207, 116, 276, 113,  30, 169,   1,
         3,  15,  22,   4,   8, 259,   0,  20,  68,  76,  78,   0,   0,
         7, 307,   4, 187,   0,  18,   7,  65, 264,   0,   0,  66,   0,
        16,  16,   9,   1,   4,   0, 154,   0,  58, 297,  26,   8,   0,
        52,  46,   7,  13,  23,  84,  57,   8,  46,  53,   2,   2, 101,
         0, 269,  11,   2,  12,   2,   1,  71, 376,   2,  32,  16,   0,
```

6.Building the model

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

---

7. make prediction using x_test

We are predicting the y_test value using the x_test,we can predict it with the help of the builded machine learning multi-linear regression model

```
y_pred=reg.predict(x_test)
y_pred
```

```
array([ 22.54937958,  29.04857309,  97.70816819,  48.87738023,
        19.19532955,  21.54549539,  22.71806894, 271.71182275,
       121.74707547,  17.87478022, 246.96398517,  93.35196024,
        55.07855506,  21.3796995 ,  23.58900644,  55.84094388,
        23.96237408, 122.31048327,  32.06170244,  15.82498665,
        16.39763695,  42.18714069,  32.6010517 ,  31.80798159,
        22.1689912 , 544.3151475 ,  50.34888014,  13.41817046,
        18.81374631,  29.86423789,  22.82440594,  10.71989833,
        31.70591371,  31.84387091,  28.09791931,  56.42604962,
       124.97163821, 159.74358645,  50.00270876, 101.87296032,
       270.80190803,  20.86358076, 124.60537134,  25.3661115 ,
        58.13558597, 221.44258947,  20.63043783,  24.38575582,
        37.92771245,  20.90457541,  20.29109994,  24.34081443,
        27.1972569 , 141.02339488, 250.02582476, 158.85606596,
        22.22447772,  23.71983704,  18.70995411,  22.02448737,
        48.11906984,  22.03773535, 371.85934437,  41.04938598,
       289.49228388,  34.93505383,  85.70226617,  68.61973233,
        32.29020925,  35.72989914,  64.4838319 ,  23.98836408,
        25.46148609,  24.89495671,  17.08999585,  24.43992858,
        41.58575695, 508.90847541,  17.85069739,  17.7749361 ,
        39.33702134, 117.54833045,  57.27715882,  20.83808944,
```

8.Comparing predicted values with the actual value

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'Difference':y_test-y_pred})
pred_df
```

	Actual Value	Predicted Value	Difference
0	2	22.549380	-20.549380
1	9	29.048573	-20.048573
2	16	97.708168	-81.708168
3	19	48.877380	-29.877380
4	0	19.195330	-19.195330
...	...	...	...
541	1	24.346404	-23.346404
542	16	19.469619	-3.469619
543	32	34.486776	-2.486776
544	19	106.044850	-87.044850
545	14	27.346398	-13.346398

546 rows × 3 columns

9.Finding the performance metrics for the created machine learning model and finding the accuracy .

#finding r2_score

The \(R^2\) score, or coefficient of determination, measures the proportion of variance in the dependent variable that is predictable from the independent variable(s). It is a statistical measure indicating the goodness of fit of a regression model. The \(R^2\) score ranges from 0 to 1, with 1 indicating a perfect fit where the model explains all the variability of the response data around its mean.

```
from sklearn.metrics import r2_score
r2=r2_score(y_test,y_pred)
r2
```

```
0.5850794224365943
```

from the above output of **r2_score** we can observe that the accuracy of the model is **0.5850794224365943** that is **59%**

#finding mean_absolute_error

Mean Absolute Error (MAE) is a metric that calculates the average of the absolute differences between predicted and actual values in a dataset. It provides a measure of the average magnitude of errors without considering their direction, making it suitable for understanding the model's prediction accuracy. The smaller the MAE, the better the model's performance in predicting the target variable.

```
from sklearn.metrics import mean_absolute_error
m1= mean_absolute_error(y_test,y_pred)
m1
```

```
38.48406962534945
```

#finding mean_squared_error

Mean Squared Error (MSE) is a statistical measure that calculates the average squared difference between predicted and actual values in a regression problem. It assesses the quality of a regression model by quantifying the average of the squared differences between predicted and observed values. Lower MSE values indicate better model performance in minimizing prediction errors.

```
from sklearn.metrics import mean_squared_error
m2=mean_squared_error(y_test,y_pred)
m2
```

```
4584.042685668658
```

Thus we have created a machine learning model based on multiple linear regression allows for predicting the potential number of deaths corresponding to reported cases. This model can serve as a valuable

tool in understanding and estimating the potential outcomes. By utilizing these predictions, authorities and decision-makers can take informed actions, implement suitable preventive measures, allocate resources, and plan interventions to manage and potentially reduce the impact of the situation.

CONCLUSION:

1. Trend Analysis:

- The analysis reveals a consistent decrease in both COVID-19 cases and deaths over the observed time period.
- There is a noticeable direct relationship between the decline in cases and the corresponding decrease in deaths.

2. Correlation Findings:

- A direct proportionality between cases and deaths is evident, showing that as cases decrease, so do deaths, indicating a clear positive correlation.
- It's important to understand that a rise in cases directly links to an increase in deaths, confirming a positive correlation.

3. Comparison of Means and Standard Deviations:

- Mean: The average number of cases and deaths has decreased over time, signifying a general decline in both.
- Standard Deviation: The variability or spread around the mean for both cases and deaths has also reduced, indicating a more consistent pattern.

4. Causation Clarification:

- Although a positive correlation exists between cases and deaths, it's crucial to note that correlation doesn't imply causation. Other factors might contribute to the relationship observed.

In essence, the analysis displays a clear decline in both COVID-19 cases and deaths over time, showing a strong positive correlation between the two, while also highlighting the need to consider various influencing factors apart from just the case numbers for understanding and managing the situation effectively.