COVID-19 CASES ANALYSIS USING COGNOS

Phase 3 Submission Document

Project Name: COVID-19 CASES ANALYSIS

Phase 3: Development Part 1

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

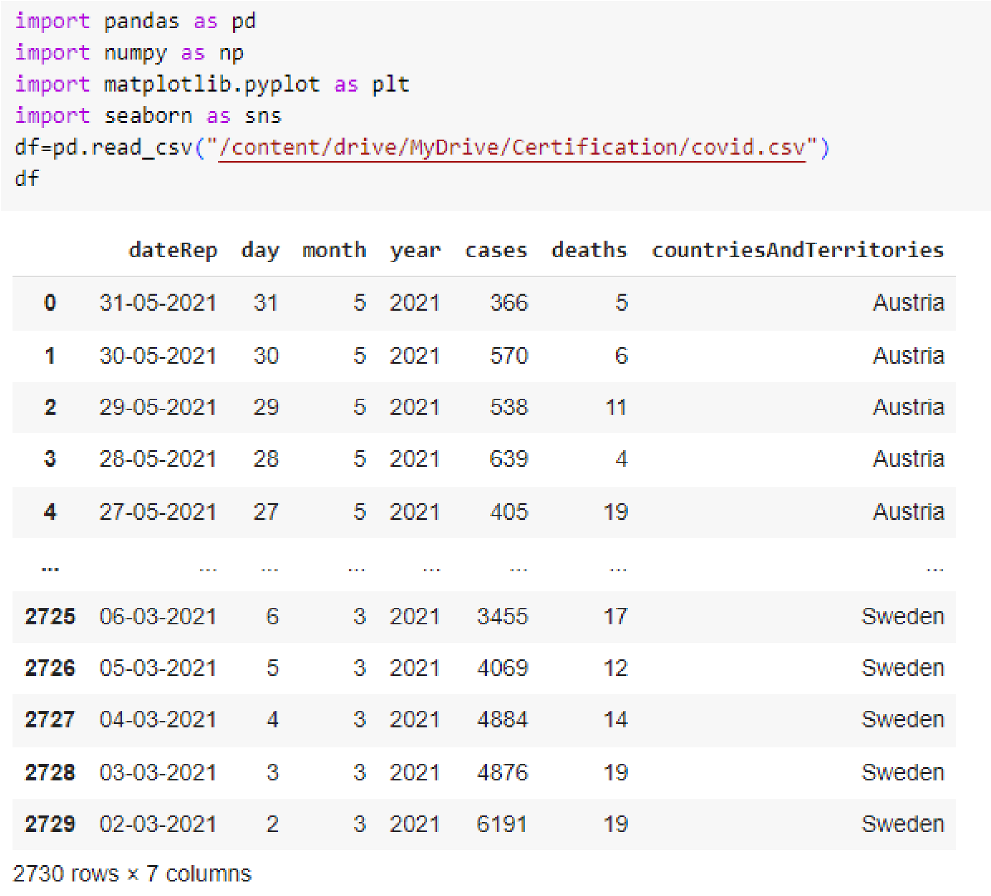
Step 1: Dataset Loading and Preprocessing

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
```

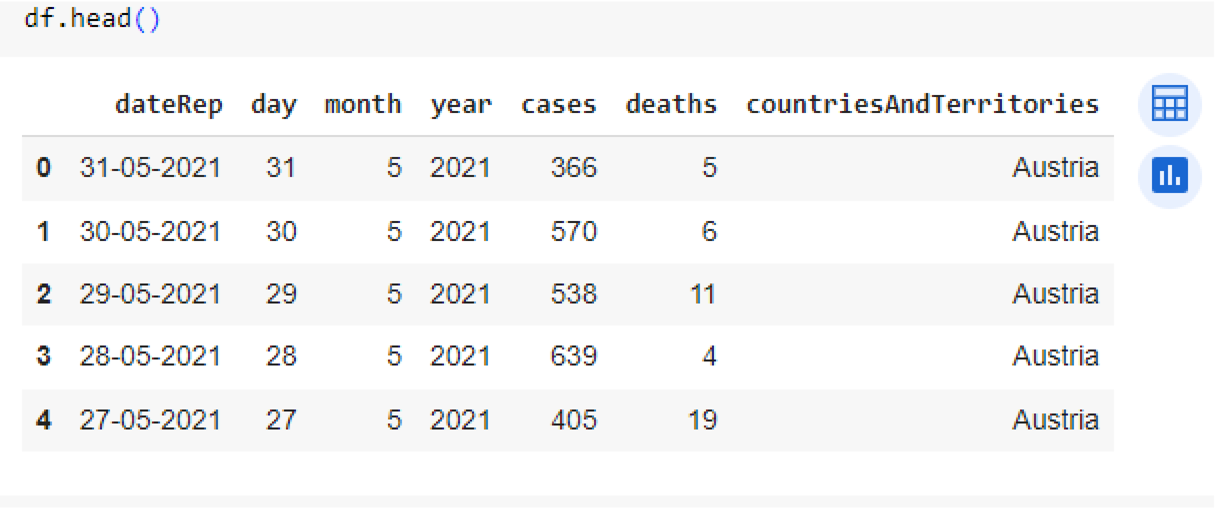
This code reads the dataset from the "your_dataset.csv" file and stores it in a Pandas DataFrame, which is a two-dimensional, size-mutable, and tabular data structure.

2. 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:

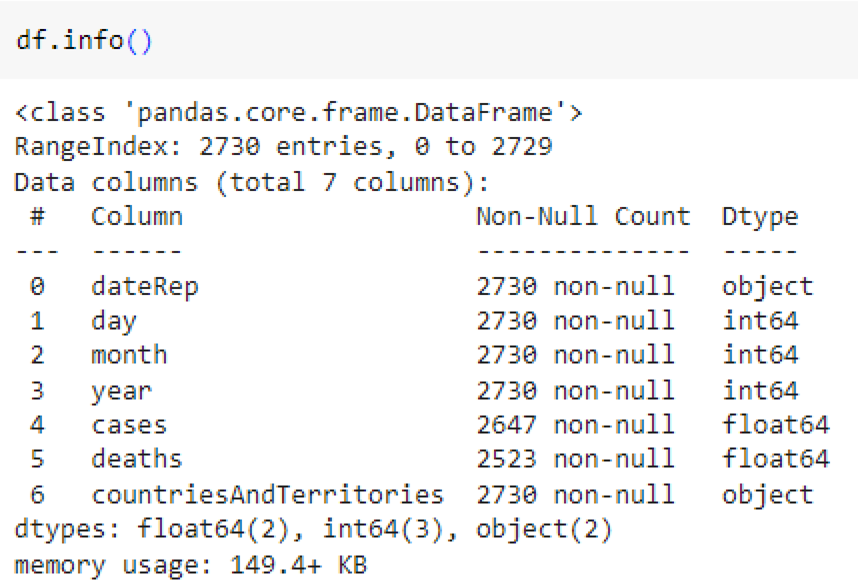
# Display the first few rows of the dataset



```
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

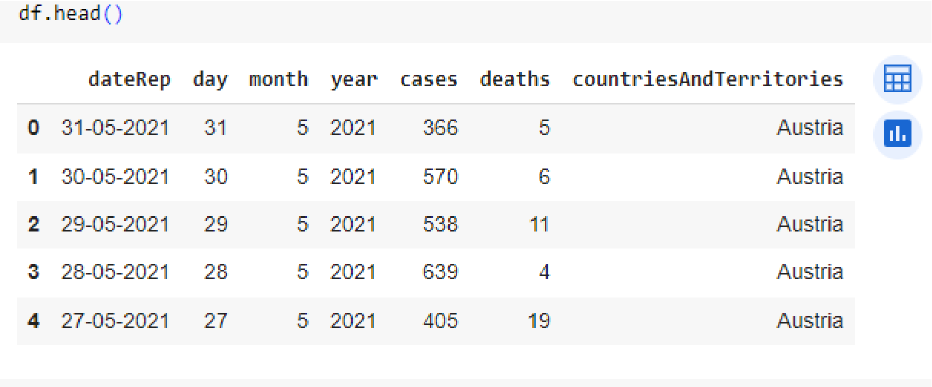
# Get information about the dataset, including data types and missing values



```
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



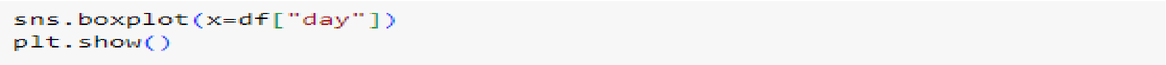
```
df.head()
```

	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

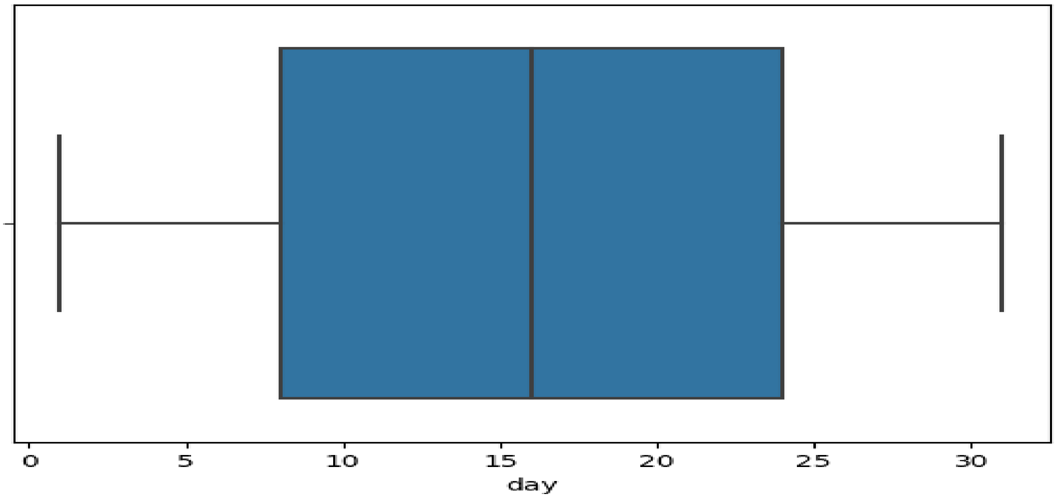
These steps help you identify any missing values, outliers, or data quality issues that need to be addressed during the data preprocessing phase.

3. Data Preprocessing:

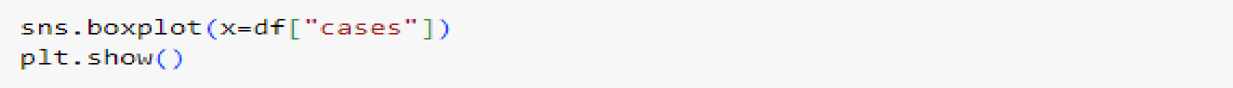
- Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. Common preprocessing tasks include:
- Handling missing values: Decide whether to impute, remove, or ignore missing data based on the nature of the problem.
- Removing duplicates: Identify and remove duplicate 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.



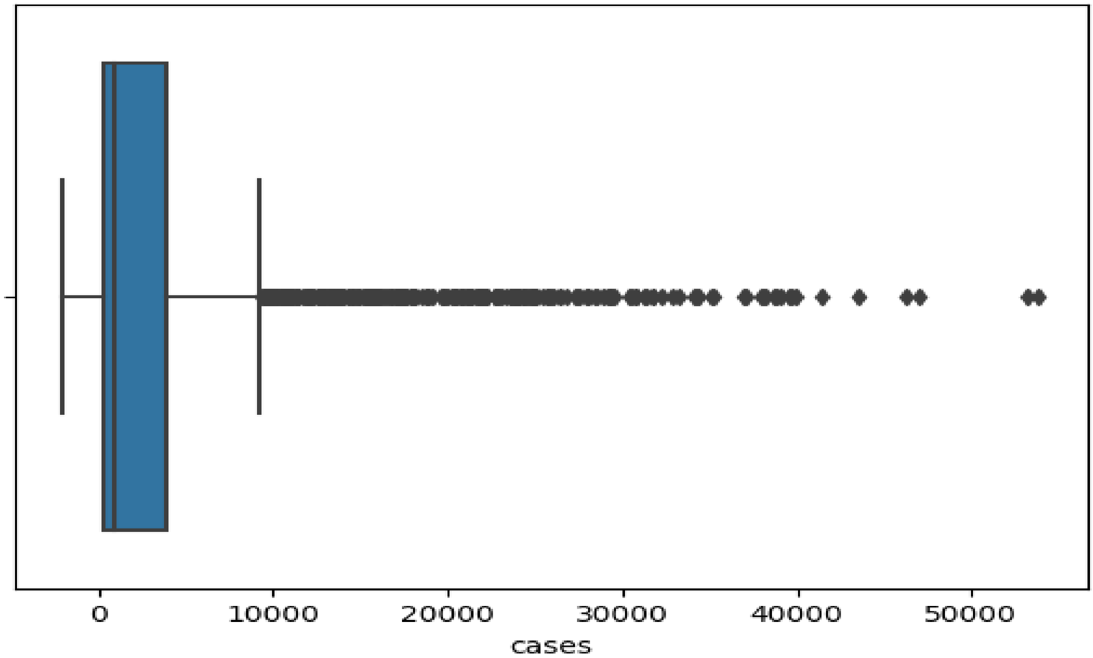
```
sns.boxplot(x=df["day"])
plt.show()
```



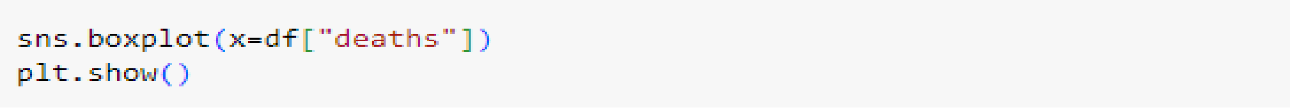
- In the above there is no outliers present in the day column similarly check for an outlier in the column cases and deaths



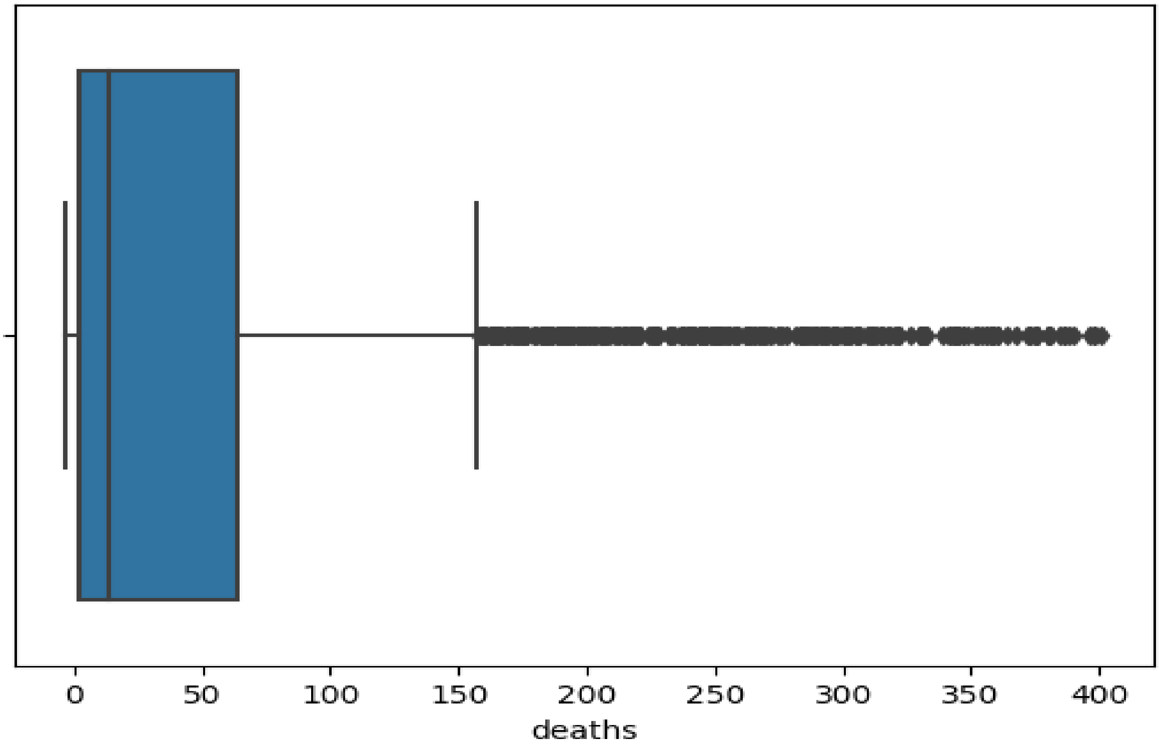
```
sns.boxplot(x=df["cases"])
plt.show()
```



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



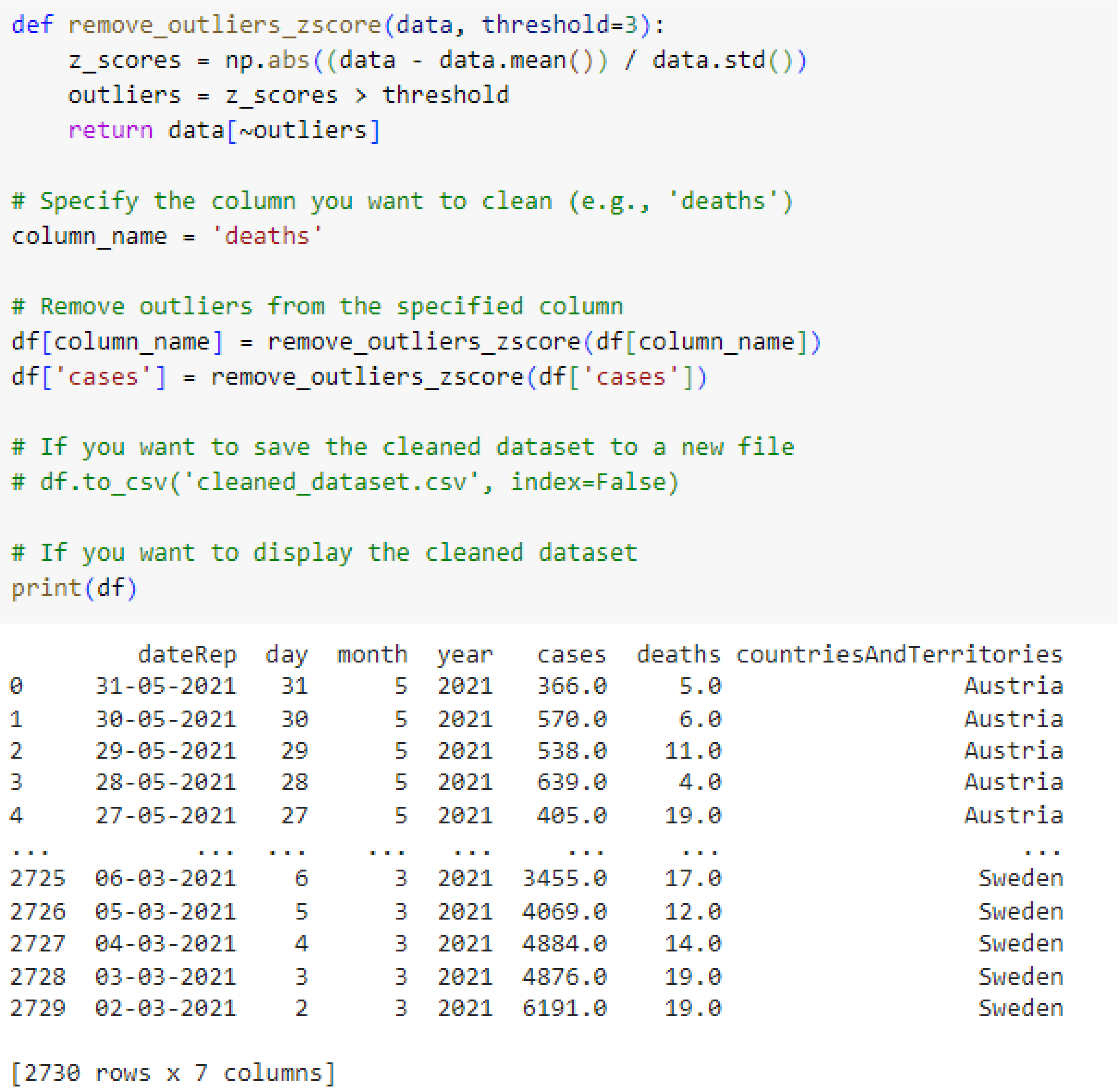
```
sns.boxplot(x=df["deaths"])
plt.show()
```



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.



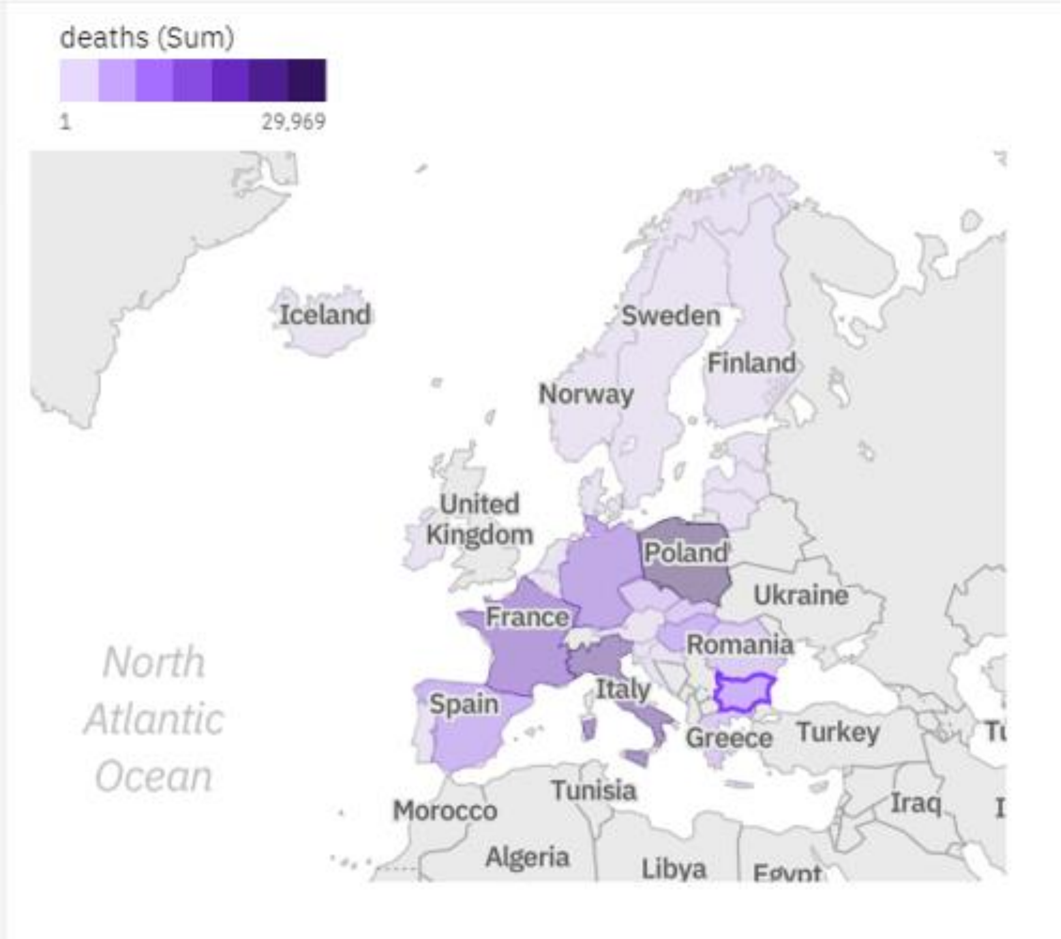
```
def remove_outliers_zscore(data, threshold=3):
    z_scores = np.abs((data - data.mean()) / data.std())
    outliers = z_scores > threshold
    return data[~outliers]

# 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)
```

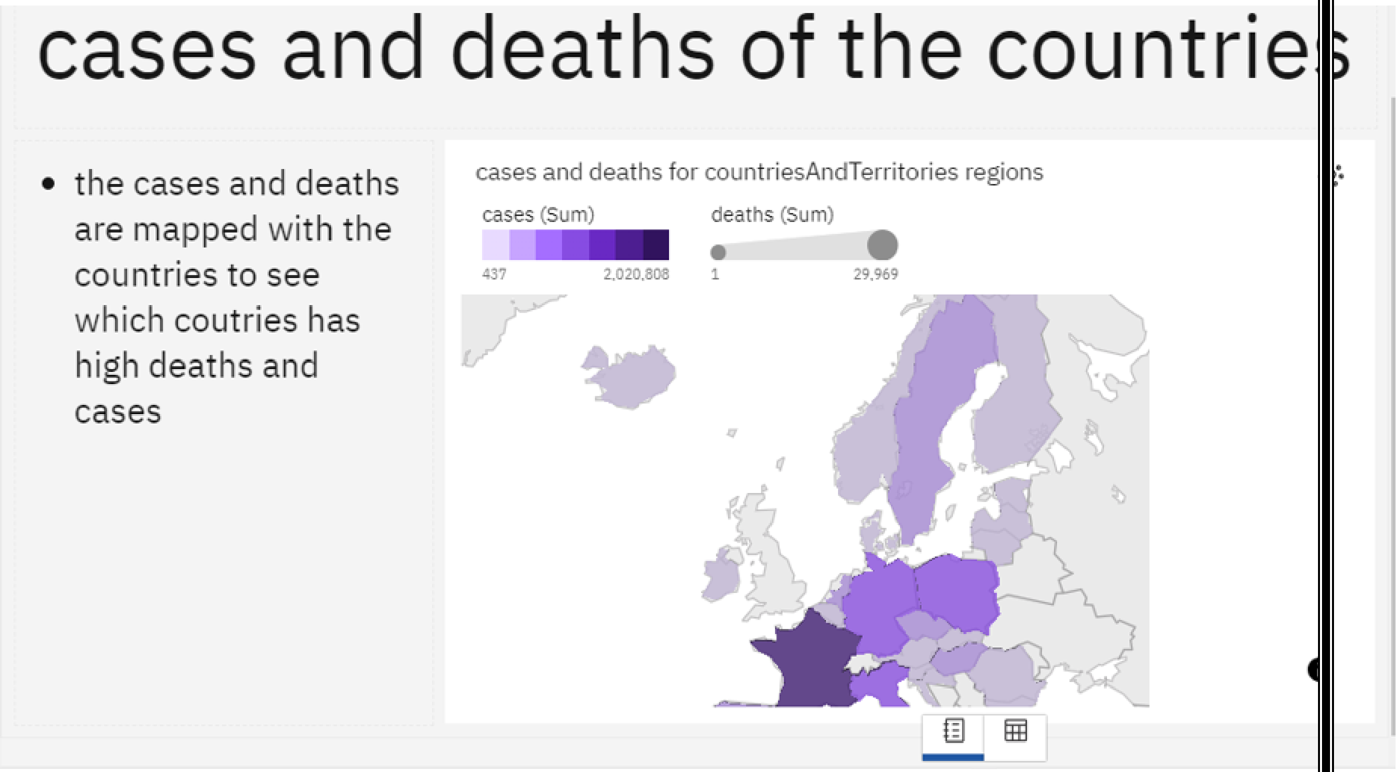
```
        dateRep  day  month  year   cases  deaths countriesAndTerritories
0     31-05-2021   31      5  2021   366.0     5.0                 Austria
1     30-05-2021   30      5  2021   570.0     6.0                 Austria
2     29-05-2021   29      5  2021   538.0    11.0                 Austria
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.

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



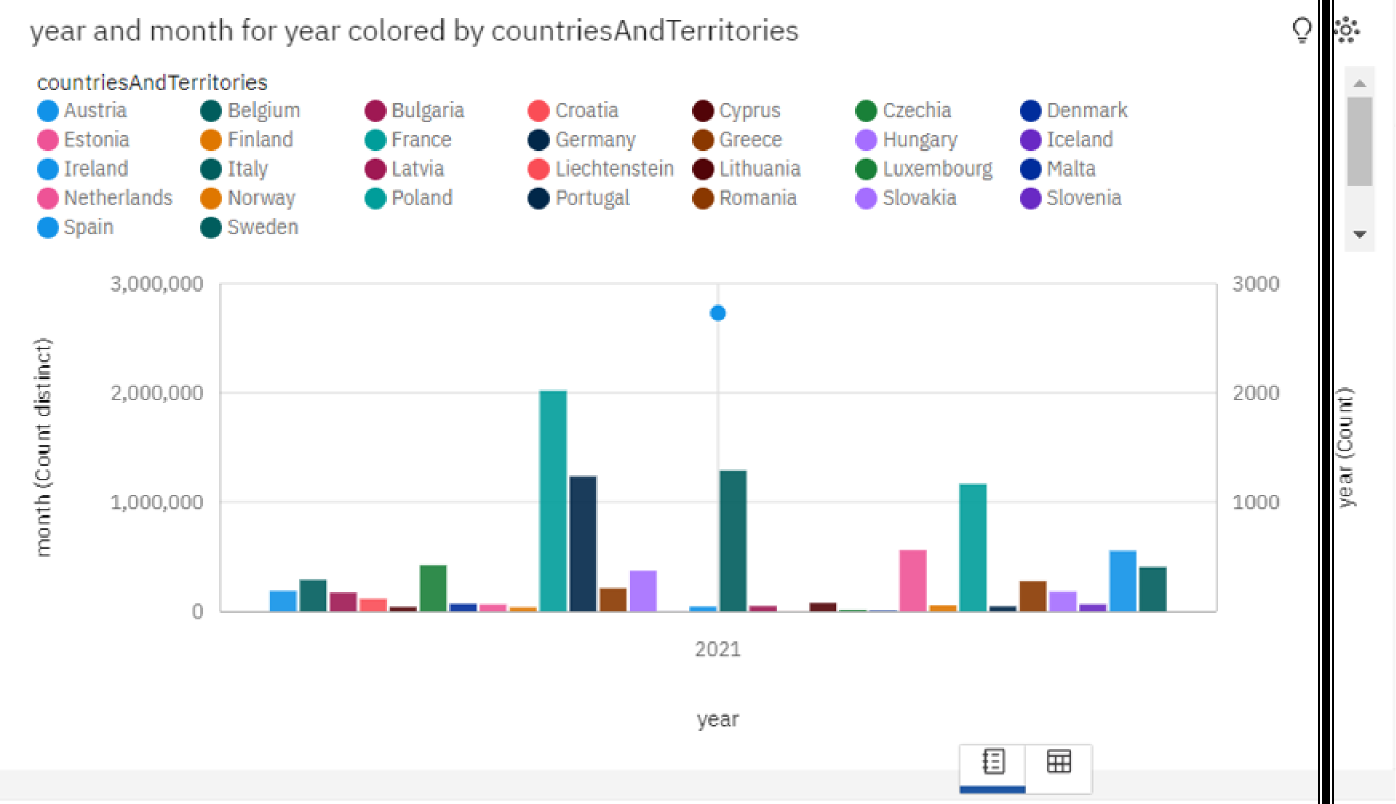


cases and deaths of the countries

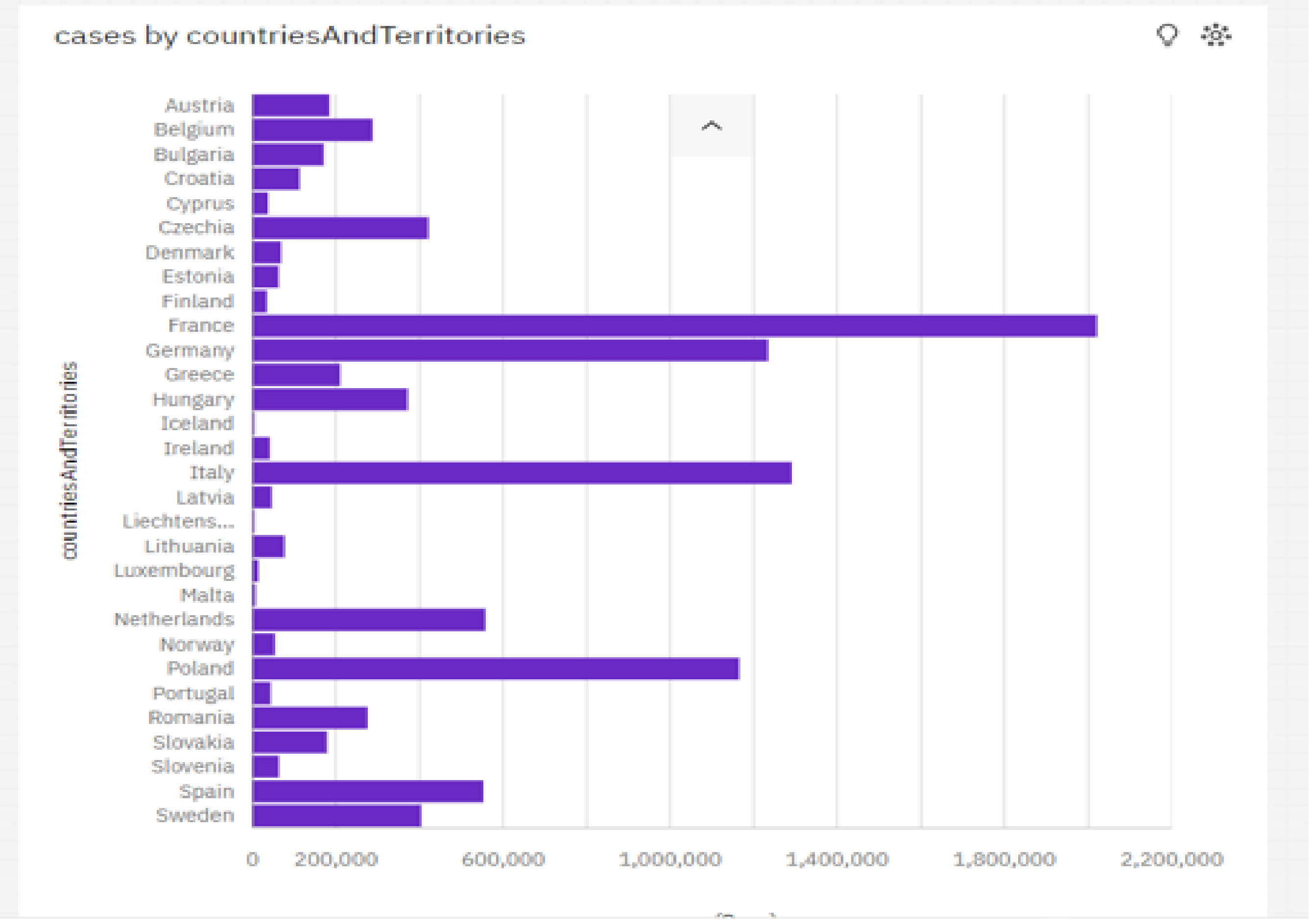
- the cases and deaths are mapped with the countries to see which coutries has high deaths and cases

cases and deaths for countriesAndTerritories regions

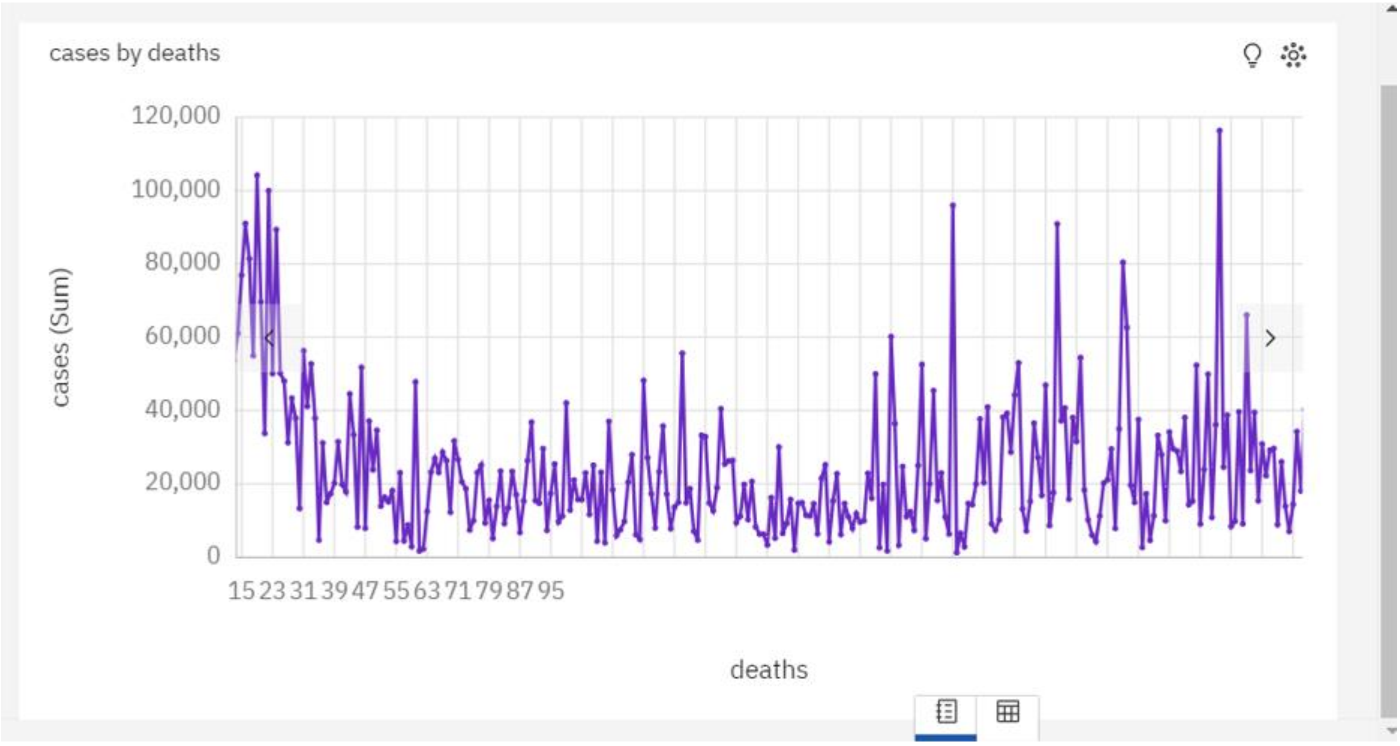
#visualizing the years and the months in which the countries are affected and from that we can see which country is affected high using IBM Cognos



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



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



Step 3: Define Analysis Objectives

1. Understand Why You're Doing This:

- Start by figuring out why you're working on this project. What's the big reason behind it? For instance, are you trying to figureout the covid cases and deaths pattern.

2. Break It Down into Specific Goals:

- Next, take that big reason and break it into smaller, clear goals. These goals will guide your work.

Simple Goals Example:

- To tract and understand the Eu countries temporal trends in COVID-19 mortality
- To examine the death case in various countries over time period .
- To calculate the death rate
- To analyse the public health

3. Make Goals Easy to Measure:

- Your goals should be easy to measure. This means you can see if you achieved them.

Measurable Goals Example:

- To tract and understand the Eu countries temporal trends in COVID-19 mortality
- To examine the death case in various countries over time period .
- To calculate the death rate
-

4. Connect Goals to the Bigger Picture:

- Make sure your goals help the company or project. Your work should lead to decisions that help the business.

5. Write Down Your Goals:

- Lastly, write your goals down in a clear way. This keeps you on track and helps others understand what you're doing.

Conclusion:

- Thus we loaded and processed the data and we made preprocessing
- Covid 19 cases analysis using IBM visualization is done and from that we can observe the cases and desth pattern and we can also able to know whick country has high deaths and cases rate.