**\*\*Data Analysis using Python on Covid 19 Dataset\*\***

**Summary:**

This analysis of the COVID-19 dataset explored key trends in confirmed cases, deaths, and recoveries across different countries and regions. We cleaned the data by handling duplicates and missing values, focusing on ensuring accuracy for analysis. Daily time series were created to track changes in confirmed cases and deaths, while recovery trends were analyzed for the top affected countries. Regional death trends were examined, with a focus on continents like Europe, Asia, and America. Correlation analysis highlighted strong relationships between confirmed cases and deaths. Overall, this analysis provided a comprehensive view of the global impact of COVID-19 over time.

Covid 19 Dataset: 

**Import required libraries**

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

**Load COVID-19 Dataset**

df = pd.read\_csv('covid\_19\_clean\_complete.csv')

**Configure visualization style**

sns.set(style="whitegrid")

plt.rcParams["date.autoformatter.year"] = "%Y"

**Set the plot size for better visuals**

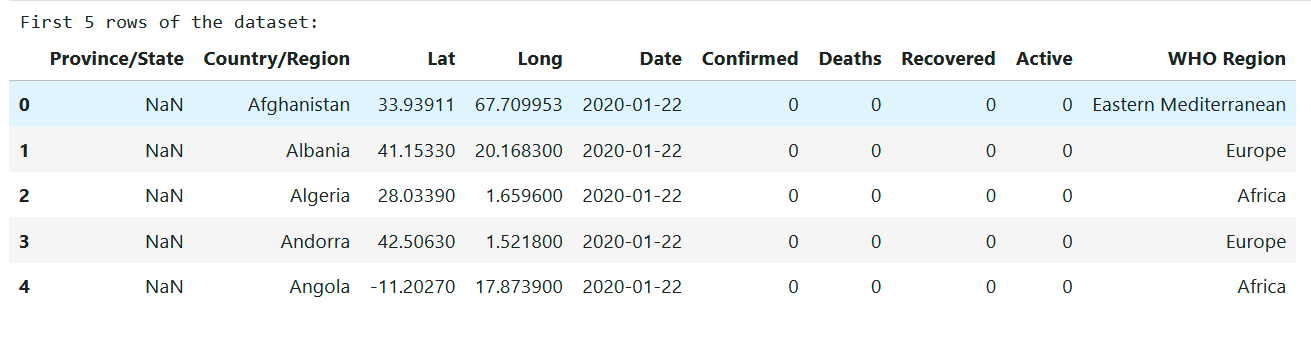
plt.figure(figsize=(12, 8))

**Display first few rows of the dataset to check structure**

print("First 5 rows of the dataset:")

df.head()

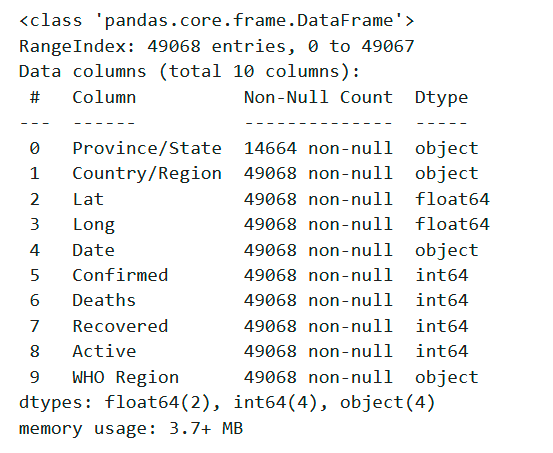
**O/P:**



**Basic information about the dataset**

df.info()

**O/P:**



**Handle duplicates by removing redundant rows**

Checking the number of rows

df.shape[0]

**O/P:**

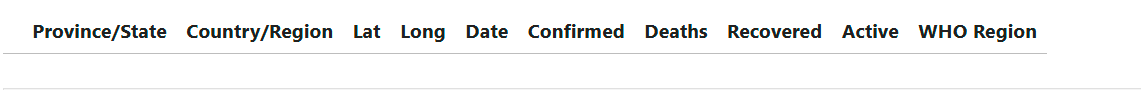
**A close-up of numbers

Description automatically generated**

Checking the duplicate rows

df[df.duplicated()]

**O/P:**



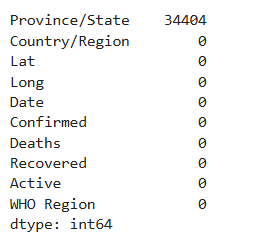
If any duplicates found, then will drop it

df.drop\_duplicates(inplace=True)

**Handle null values by filling them appropriately**

(df.isnull()).sum()

**O/P:**

****

**To display the null values in the dataframe**

df[df['Province/State'].isnull()]

**O/P:**

**A screenshot of a computer screen

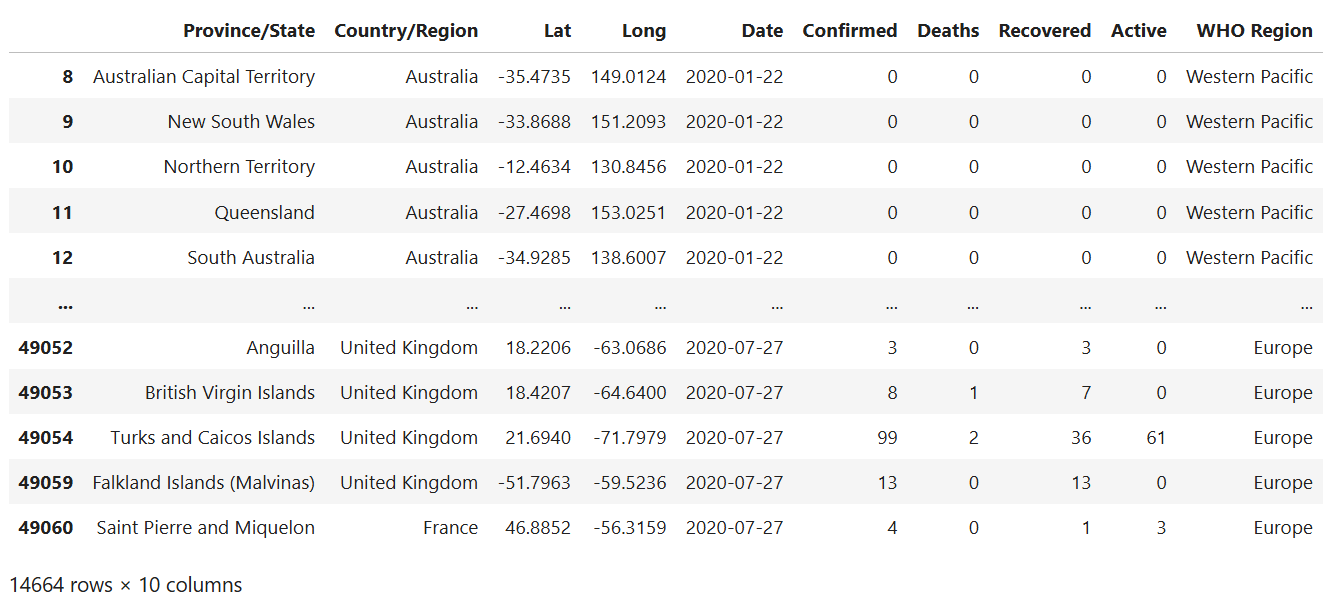
Description automatically generated**

**Note:** There is no duplicate values in any other columns except in the Province/State column, which had missing values, I chose not to replace them, as it’s a non-numerical field and imputation could lead to inaccuracies. Instead, I filtered out the rows with null values to ensure the analysis remained reliable and meaningful. This approach to adapt the data cleaning methods to the type of data while maintaining accuracy and relevance in the analysis.

**Displaying the not null states in the dataframes**

df[df['Province/State'].notnull()]

**O/P:**



**Explore unique countries in the dataset to clean inconsistencies**

df['Country/Region'].unique().tolist()

**O/P:**

**A screenshot of a computer program

Description automatically generated**

**Create a daily time series of new cases**

df['Date'] = pd.to\_datetime(df['Date'])

df['Daily\_Confirmed'] = df['Confirmed'].diff()

df['Daily\_Deaths'] = df['Deaths'].diff()

df[df['Daily\_Deaths'] >0]

**O/P:**

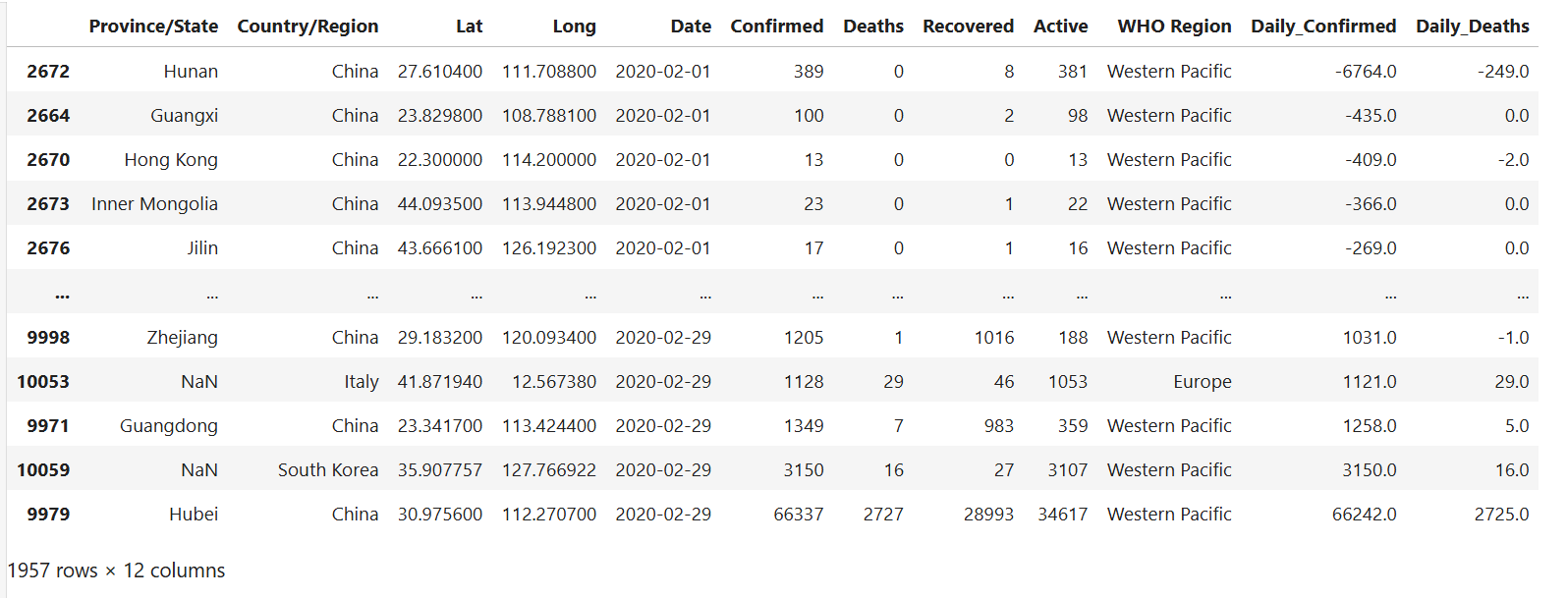
A screenshot of a computer

Description automatically generated

**Deaths in the Month of February**

df[(df['Date'].dt.month == 2) & (df['Confirmed'] != 0)].sort\_values(by=['Date','Daily\_Confirmed'])

**O/P:**



**Aggregate data by country and date**

df.groupby(['Country/Region','Date']).agg({'Confirmed' : 'sum', 'Deaths': 'sum', 'Recovered': 'sum'}).reset\_index()

**O/P:**

A screenshot of a table

Description automatically generated

**Generate insights on top countries with the highest confirmed COVID-19 cases**

df.groupby('Country/Region').agg({'Confirmed':'max'}).sort\_values(by = 'Confirmed', ascending = False).head(10)

**O/P:**

**A screenshot of a computer

Description automatically generated**

**Filtering the cases count by Region and Date**

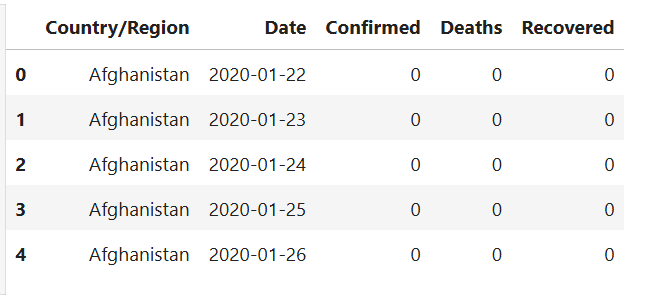
country\_group = df.groupby(['Country/Region', 'Date']).agg({'Confirmed': 'sum',

'Deaths': 'sum',

'Recovered': 'sum',}).reset\_index()

country\_group.head()

**O/P:**

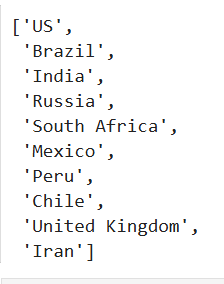


**Filtering the data to analyze the top 10 countries affected by Covid-19**

top\_countries = country\_group.groupby('Country/Region')['Confirmed'].max().nlargest(10).index.tolist()

top\_countries

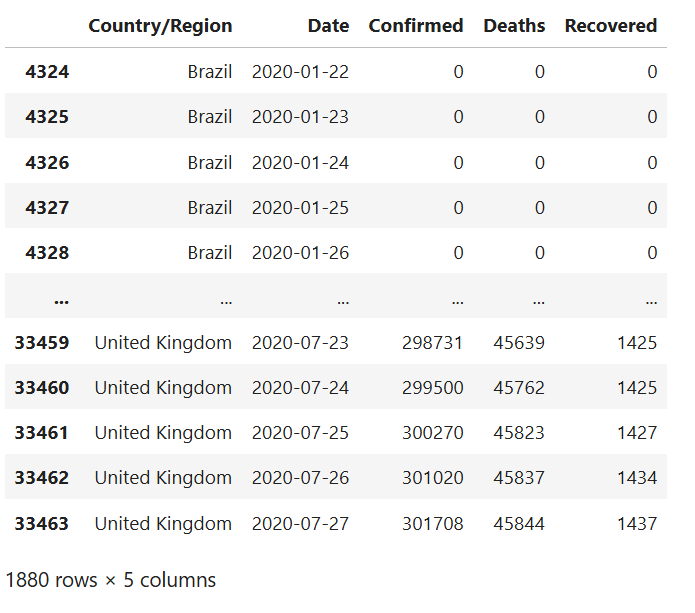
**O/P:**



**Displaying the dataset with only top 10 countries**

country\_group[country\_group['Country/Region'].isin(top\_countries)]

**O/P:**



**Plotting the cases count increase over the time for the top countries affected by Covid-19**

top\_countries\_df = country\_group[country\_group['Country/Region'].isin(top\_countries)]

plt.figure(figsize=(12, 8))

sns.lineplot(data=top\_countries\_df, x='Date', y='Confirmed', hue='Country/Region')

plt.title("Confirmed Cases in Top 10 Countries")

plt.xlabel('Date')

plt.ylabel('Confirmed Cases')

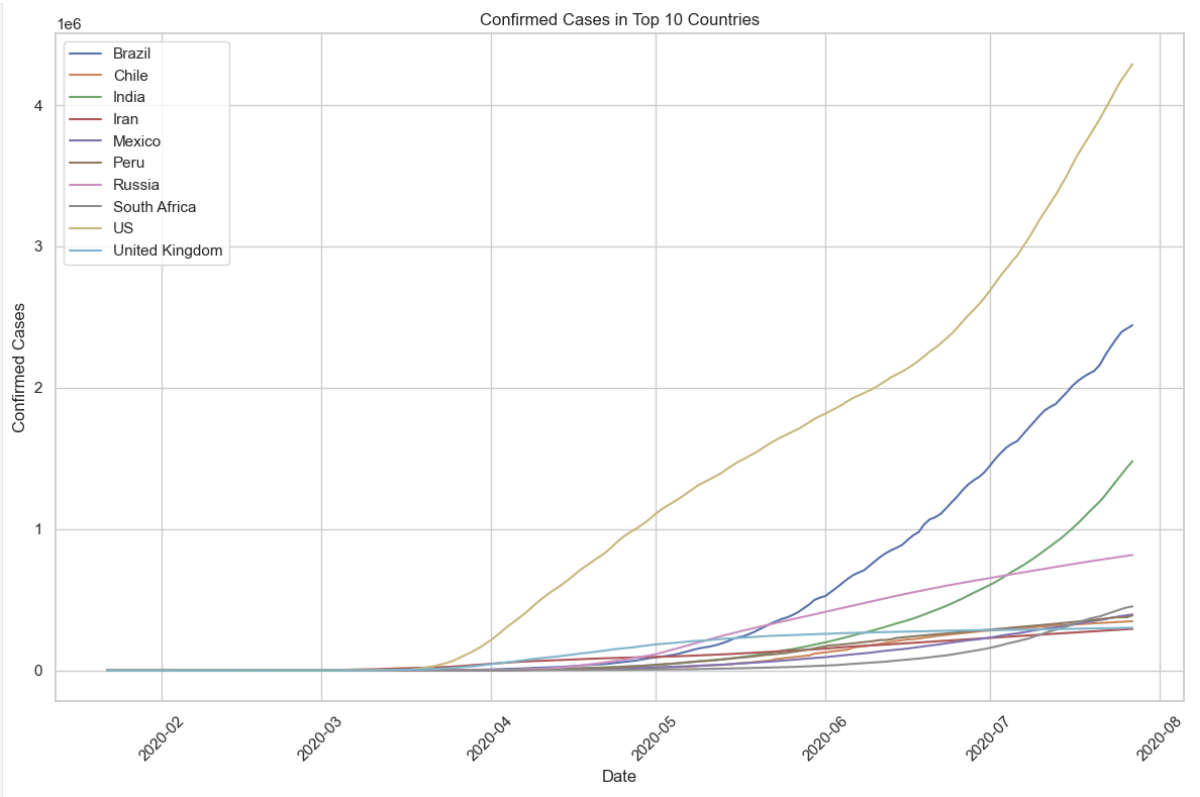
plt.legend(loc='upper left')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**O/P:**



**Plotting the recovery rate increase over the time for the top countries affected by Covid-19**

plt.figure(figsize=(12, 8))

sns.lineplot(data=country\_group, x='Date', y='Recovered', hue='Country/Region')

plt.title("Daily Recovery Rate Across Top COVID-19 Affected Countries")

plt.xlabel('Date')

plt.ylabel('Recovery Rate')

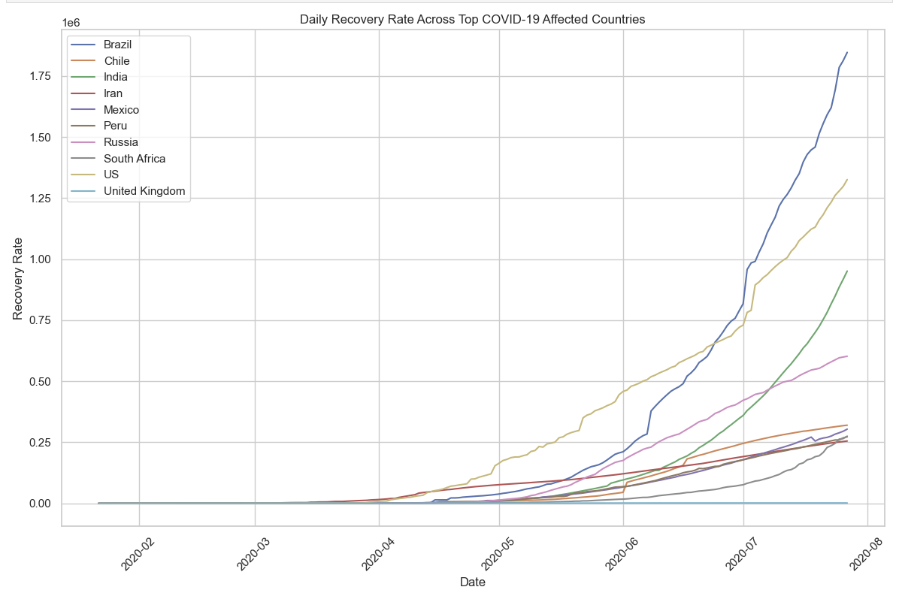
plt.legend(loc='upper left')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**O/P:**

****

**Regional analysis of deaths across major continents**

df['Continent'] = df['Country/Region'].apply(lambda x: 'Europe' if x in ['Italy', 'Spain', 'Germany'] else

('Asia' if x in ['China', 'India', 'Japan'] else

('America' if x in ['USA', 'Brazil', 'Argentina'] else 'Others')))

**Aggregate data by continent over time**

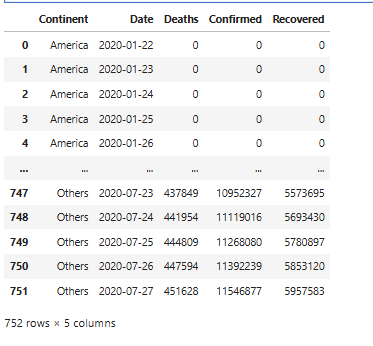
continent\_group\_region= df.groupby(['Continent', 'Date']).agg({

'Deaths': 'sum',

}).reset\_index()

continent\_group\_region

**O/P:**



**Visualize the death trends across regions**

plt.figure(figsize=(12, 8))

sns.lineplot(data=continent\_group\_region, x='Date', y='Deaths', hue='Continent')

plt.title("Trends in Deaths by Continent")

plt.xlabel('Date')

plt.ylabel('Number of Deaths')

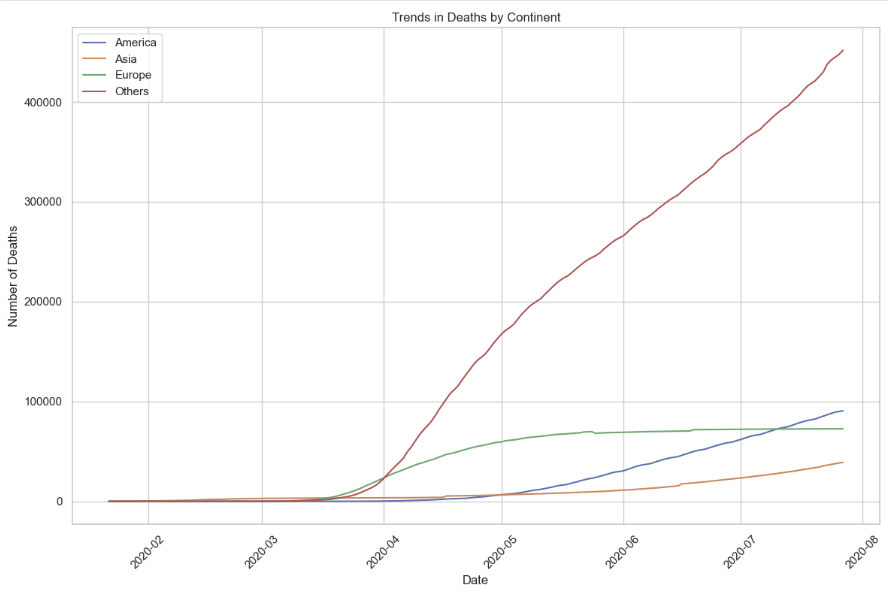
plt.legend(loc='upper left')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**O/P:**



**Statistical insights: Correlation analysis**

correlation\_df = df[['Confirmed', 'Deaths', 'Recovered']].copy()

correlation\_matrix = correlation\_df.corr()

**Visualizing correlations between these parameters**

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

plt.title("Correlation Matrix Between Confirmed, Deaths, and Recovered")

plt.show()

**O/P:**

