Import necessary libraries (e.g., pandas, matplotlib, seaborn).  
Load the COVID-19 dataset.  
Display basic information about the dataset (e.g., data types, missing values).

In [1]:

*# load required libraries/modules*

2

**import** pandas **as** pd

3

**import** datetime

4

**import** matplotlib.pyplot **as** plt

5

**import** seaborn **as** sns

6

​

7

​

8

*# Load the dataset*

9

df**=**pd.read\_csv(r"C:\Users\ashis\Downloads\covid\_19\_data (1).csv")

10

​

11

df.head()

12

​

13

*# Display basic information about the dataset*

14

​

15

​

Out[1]:

|  | **SNo** | **ObservationDate** | **Province/State** | **Country/Region** | **Last Update** | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 01/22/2020 | Anhui | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **1** | 2 | 01/22/2020 | Beijing | Mainland China | 1/22/2020 17:00 | 14.0 | 0.0 | 0.0 |
| **2** | 3 | 01/22/2020 | Chongqing | Mainland China | 1/22/2020 17:00 | 6.0 | 0.0 | 0.0 |
| **3** | 4 | 01/22/2020 | Fujian | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **4** | 5 | 01/22/2020 | Gansu | Mainland China | 1/22/2020 17:00 | 0.0 | 0.0 | 0.0 |

In [2]:

df.shape

Out[2]:

(306429, 8)

In [3]:

df.columns

Out[3]:

Index(['SNo', 'ObservationDate', 'Province/State', 'Country/Region',

'Last Update', 'Confirmed', 'Deaths', 'Recovered'],

dtype='object')

In [4]:

df.isna().sum()

Out[4]:

SNo 0

ObservationDate 0

Province/State 78103

Country/Region 0

Last Update 0

Confirmed 0

Deaths 0

Recovered 0

dtype: int64

In [5]:

df.dtypes

Out[5]:

SNo int64

ObservationDate object

Province/State object

Country/Region object

Last Update object

Confirmed float64

Deaths float64

Recovered float64

dtype: object

**2. Data Cleaning:**

In [6]:

Handle missing values.

2

Convert data types **if** necessary.

3

Check **for** duplicates.

**Cell In[6], line 1**

**Handle missing values.**

**^**

**SyntaxError:** invalid syntax

In [7]:

*# Handle missing values*

2

**for** i **in** df:

3

**if** df[i].dtypes**==**"object":

4

m**=**df[i].mode()[0]

5

df[i]**=**df[i].fillna(m)

6

**else**:

7

avg**=**df[i].mean()

8

df[i]**=**df[i].fillna(avg)

9

​

10

*# Convert data types if necessary ( ObservationDate datetime64[ns]) change to datetime*

11

*#df['observationDate'] = pd.to\_datetimedf['odservationDate'],*

12

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

13

​

14

*# Check for duplicates*

15

*# print("Number of duplicates:", df.duplicated().sum())*

16

​

In [8]:

df.head()

Out[8]:

|  | **SNo** | **ObservationDate** | **Province/State** | **Country/Region** | **Last Update** | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 2020-01-22 | Anhui | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **1** | 2 | 2020-01-22 | Beijing | Mainland China | 1/22/2020 17:00 | 14.0 | 0.0 | 0.0 |
| **2** | 3 | 2020-01-22 | Chongqing | Mainland China | 1/22/2020 17:00 | 6.0 | 0.0 | 0.0 |
| **3** | 4 | 2020-01-22 | Fujian | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **4** | 5 | 2020-01-22 | Gansu | Mainland China | 1/22/2020 17:00 | 0.0 | 0.0 | 0.0 |

In [9]:

df.isna().sum()

Out[9]:

SNo 0

ObservationDate 0

Province/State 0

Country/Region 0

Last Update 0

Confirmed 0

Deaths 0

Recovered 0

dtype: int64

**3. Descriptive Statistics:**

Generate descriptive statistics (mean, median, min, max) for numeric columns.  
Explore summary statistics for categorical columns.

In [10]:

*# Descriptive statistics*

2

medians **=** df.select\_dtypes(include**=**['float64', 'int64']).median()

3

​

4

*# Descriptive statistics for numeric columns*

5

numeric\_stats **=** df.describe()

6

​

7

*# Add median to the numeric\_stats DataFrame*

8

numeric\_stats.loc['median'] **=** medians

9

​

10

*# Explore summary statistics for categorical columns*

11

categorical\_stats **=** {}

12

**for** column **in** df.select\_dtypes(include**=**'object'):

13

categorical\_stats[column] **=** df[column].value\_counts()

14

​

15

*# Display the results*

16

print("Descriptive Statistics for Numeric Columns:")

17

print(numeric\_stats)

18

​

19

print("\nSummary Statistics for Categorical Columns:")

20

**for** column, stats **in** categorical\_stats.items():

21

print(f"\nColumn: {column}")

22

print(stats)

23

​

24

25

​

26

*# Summary statistics for categorical columns*

27

​

28

​

Descriptive Statistics for Numeric Columns:

SNo ObservationDate Confirmed \

count 306429.000000 306429 3.064290e+05

mean 153215.000000 2020-11-06 01:54:54.146441728 8.567091e+04

min 1.000000 2020-01-22 00:00:00 -3.028440e+05

25% 76608.000000 2020-07-30 00:00:00 1.042000e+03

50% 153215.000000 2020-11-10 00:00:00 1.037500e+04

75% 229822.000000 2021-02-18 00:00:00 5.075200e+04

max 306429.000000 2021-05-29 00:00:00 5.863138e+06

std 88458.577156 NaN 2.775516e+05

median 153215.000000 NaN 1.037500e+04

Deaths Recovered

count 306429.000000 3.064290e+05

mean 2036.403268 5.042029e+04

min -178.000000 -8.544050e+05

25% 13.000000 1.100000e+01

50% 192.000000 1.751000e+03

75% 1322.000000 2.027000e+04

max 112385.000000 6.399531e+06

std 6410.938048 2.015124e+05

median 192.000000 1.751000e+03

Summary Statistics for Categorical Columns:

Column: Province/State

Province/State

Unknown 82226

Amazonas 1109

Diamond Princess cruise ship 924

Grand Princess 882

Punjab 708

...

Harrison County, KY 1

Queens County, NY 1

Johnson County, IA 1

Unassigned Location, VT 1

Charlotte County, FL 1

Name: count, Length: 736, dtype: int64

Column: Country/Region

Country/Region

Russia 30251

US 26740

Japan 18059

Mainland China 15758

India 13182

...

Azerbaijan 1

North Ireland 1

Republic of Ireland 1

Cape Verde 1

East Timor 1

Name: count, Length: 229, dtype: int64

Column: Last Update

Last Update

2021-04-02 15:13:53 239885

3/8/20 5:31 1252

2021-05-30 04:20:55 765

2021-04-26 04:21:04 765

2021-04-20 04:20:41 765

...

2020-03-10T10:33:02 1

2020-02-18T01:33:01 1

2020-03-10T01:33:02 1

2020-03-10T07:33:03 1

2020-02-04T03:03:07 1

Name: count, Length: 1905, dtype: int64

**4. Time Series Analysis:**

Analyze the trends in confirmed, deaths, and recovered cases over time.  
Create time-based visualizations.

In [11]:

*# Time series analysis*

2

*#groupby 'ObservationDate' and apply aggregation functions like sum on deaths,confirmed ,recovered*

3

df\_time **=** df.groupby('ObservationDate').agg({'Confirmed': 'sum', 'Deaths': 'sum', 'Recovered': 'sum'})

4

​

5

*# Plot time series data using line plot*

6

​

7

*# syntax:*

8

sns.lineplot(data**=**df\_time, x**=**'ObservationDate', y**=**'Confirmed', label**=**'Confirmed')

9

sns.lineplot(data**=**df\_time, x**=**'ObservationDate', y**=**'Deaths', label**=**'Deaths')

10

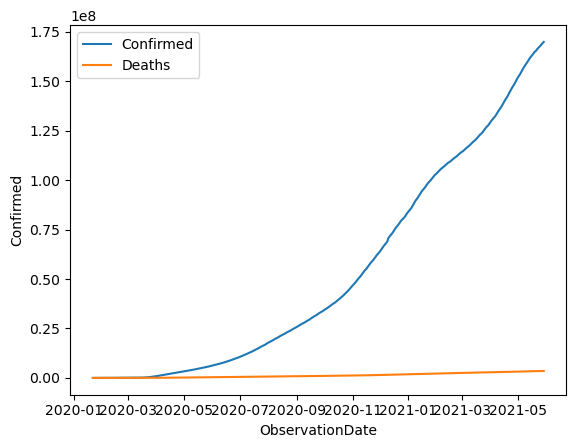
​

11

​

Out[11]:

<Axes: xlabel='ObservationDate', ylabel='Confirmed'>



In [12]:

df.head()

Out[12]:

|  | **SNo** | **ObservationDate** | **Province/State** | **Country/Region** | **Last Update** | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 2020-01-22 | Anhui | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **1** | 2 | 2020-01-22 | Beijing | Mainland China | 1/22/2020 17:00 | 14.0 | 0.0 | 0.0 |
| **2** | 3 | 2020-01-22 | Chongqing | Mainland China | 1/22/2020 17:00 | 6.0 | 0.0 | 0.0 |
| **3** | 4 | 2020-01-22 | Fujian | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **4** | 5 | 2020-01-22 | Gansu | Mainland China | 1/22/2020 17:00 | 0.0 | 0.0 | 0.0 |

**5. Geographical Analysis:**

Analyze the distribution of cases across different countries/regions.  
Create geographical visualizations (e.g., world map).

In [13]:



1

*# Geographical analysis*

2

df\_country **=** df.groupby('Country/Region').agg({'Confirmed': 'sum', 'Deaths': 'sum', 'Recovered': 'sum'})

3

​

4

*# Plot geographical data*

5

sns.barplot(x**=**df\_country.index, y**=**'Confirmed', data**=**df\_country)

6

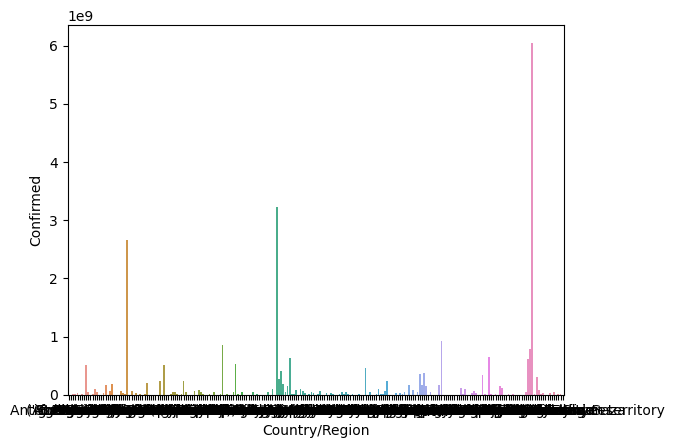
​

7

​

Out[13]:

<Axes: xlabel='Country/Region', ylabel='Confirmed'>



**6. Correlation Analysis:**

Examine the correlation between different numeric variables.

In [14]:



1

*# Correlaion analysis between confirmed,deaths and recovered*

2

correlation\_matrix **=** df[['Confirmed', 'Deaths', 'Recovered']].corr()

3

*#plot crrelation on map*

4

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

5

plt.title('Correlation Matrix')

6

plt.show()

7

​

8

​

9

df[['Confirmed','Deaths']].corr()

10

​

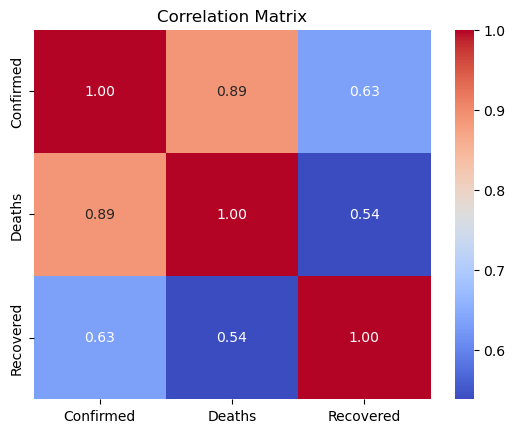
11

​

12

​

13

​

Out[14]:

|  | **Confirmed** | **Deaths** |
| --- | --- | --- |
| **Confirmed** | 1.000000 | 0.888416 |
| **Deaths** | 0.888416 | 1.000000 |

In [15]:

df.head()

Out[15]:

|  | **SNo** | **ObservationDate** | **Province/State** | **Country/Region** | **Last Update** | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 2020-01-22 | Anhui | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **1** | 2 | 2020-01-22 | Beijing | Mainland China | 1/22/2020 17:00 | 14.0 | 0.0 | 0.0 |
| **2** | 3 | 2020-01-22 | Chongqing | Mainland China | 1/22/2020 17:00 | 6.0 | 0.0 | 0.0 |
| **3** | 4 | 2020-01-22 | Fujian | Mainland China | 1/22/2020 17:00 | 1.0 | 0.0 | 0.0 |
| **4** | 5 | 2020-01-22 | Gansu | Mainland China | 1/22/2020 17:00 | 0.0 | 0.0 | 0.0 |

**7. Advanced Analysis:**

Explore advanced visualizations or statistical tests based on the specific questions you want to answer.

In [16]:

*# Calculate the total number of confirmed cases, deaths, and recoveries for each country/region.*

2

covid\_country\_region **=** df.groupby('Country/Region').agg({'Confirmed': 'sum', 'Deaths': 'sum', 'Recovered': 'sum'})

3

covid\_country\_region

Out[16]:

|  | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- |
| **Country/Region** |  |  |  |
| **Azerbaijan** | 1.0 | 0.0 | 0.0 |
| **('St. Martin',)** | 2.0 | 0.0 | 0.0 |
| **Afghanistan** | 17026442.0 | 669075.0 | 13464399.0 |
| **Albania** | 19768869.0 | 375955.0 | 13945256.0 |
| **Algeria** | 27684358.0 | 834464.0 | 18959299.0 |
| **...** | ... | ... | ... |
| **West Bank and Gaza** | 41819444.0 | 440378.0 | 37003116.0 |
| **Yemen** | 962066.0 | 237613.0 | 506523.0 |
| **Zambia** | 13493953.0 | 205990.0 | 12625626.0 |
| **Zimbabwe** | 6484581.0 | 237234.0 | 5594887.0 |
| **occupied Palestinian territory** | 25.0 | 0.0 | 0.0 |

229 rows × 3 columns

In [17]:

*# Identify the countries/regions with the highest number of confirmed cases, deaths, and recoveries.*

2

​

3

Highest\_number **=** df.sort\_values(by**=**['Country/Region','Confirmed','Deaths','Recovered'],ascending**=False**)

4

Highest\_number

Out[17]:

|  | **SNo** | **ObservationDate** | **Province/State** | **Country/Region** | **Last Update** | **Confirmed** | **Deaths** | **Recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4688** | 4689 | 2020-03-10 | Unknown | occupied Palestinian territory | 2020-03-10T19:13:21 | 25.0 | 0.0 | 0.0 |
| **5032** | 5033 | 2020-03-11 | Unknown | occupied Palestinian territory | 2020-03-11T20:53:02 | 0.0 | 0.0 | 0.0 |
| **5252** | 5253 | 2020-03-12 | Unknown | occupied Palestinian territory | 2020-03-11T20:53:02 | 0.0 | 0.0 | 0.0 |
| **5735** | 5736 | 2020-03-14 | Unknown | occupied Palestinian territory | 2020-03-11T20:53:02 | 0.0 | 0.0 | 0.0 |
| **5995** | 5996 | 2020-03-15 | Unknown | occupied Palestinian territory | 2020-03-11T20:53:02 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2858** | 2859 | 2020-02-29 | Unknown | Afghanistan | 2020-02-24T23:33:02 | 1.0 | 0.0 | 0.0 |
| **2988** | 2989 | 2020-03-01 | Unknown | Afghanistan | 2020-02-24T23:33:02 | 1.0 | 0.0 | 0.0 |
| **3121** | 3122 | 2020-03-02 | Unknown | Afghanistan | 2020-02-24T23:33:02 | 1.0 | 0.0 | 0.0 |
| **4770** | 4771 | 2020-03-10 | Unknown | ('St. Martin',) | 2020-03-10T05:33:02 | 2.0 | 0.0 | 0.0 |
| **2737** | 2738 | 2020-02-28 | Unknown | Azerbaijan | 2020-02-28T15:03:26 | 1.0 | 0.0 | 0.0 |

306429 rows × 8 columns

In [18]:



1

*# Visualize the distribution of deaths using bar chart*

2

covid\_death **=** df.groupby('Country/Region').agg({'Deaths': 'max'})

3

covid\_death

4

​

Out[18]:

|  | **Deaths** |
| --- | --- |
| **Country/Region** |  |
| **Azerbaijan** | 0.0 |
| **('St. Martin',)** | 0.0 |
| **Afghanistan** | 2899.0 |
| **Albania** | 2449.0 |
| **Algeria** | 3460.0 |
| **...** | ... |
| **West Bank and Gaza** | 3492.0 |
| **Yemen** | 1319.0 |
| **Zambia** | 1276.0 |
| **Zimbabwe** | 1594.0 |
| **occupied Palestinian territory** | 0.0 |

229 rows × 1 columns