Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/covid-19-cases (https://www.kaggle.com/datasets/chakradharmattapalli/covid-19-cases)

Importing Necessary Libraries

In [1]:

```
'''Importing Necessary Libraries'''
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
import seaborn as sns
```

Import and read dataset

In [2]:

```
'''Import and read dataset'''
covid_data=pd.read_csv("Covid_19_cases4.csv")
covid_data.head(10)
```

Out[2]:

	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
5	26-05-2021	26	5	2021	287	8	Austria
6	25-05-2021	25	5	2021	342	3	Austria
7	24-05-2021	24	5	2021	520	3	Austria
8	23-05-2021	23	5	2021	626	8	Austria
9	22-05-2021	22	5	2021	671	12	Austria

Click Here for the Hint

Data Cleaning

a. Missing Value

In [3]:

```
'''Data Cleaning missing values'''
covid_data.isnull().sum()
```

Out[3]:

```
dateRep                    0
day                        0
month                      0
year                       0
cases                      0
deaths                     0
countriesAndTerritories    0
dtype: int64
```

Click Here for the Hint

b. Duplicate data

In [4]:

```
'''Duplicate Data'''
covid_data.duplicated().sum()
```

Out[4]:

```
0
```

Click Here for the Hint

c. drop unecessary columns

In [5]:

```
'''Drop unnecessary columns'''
covid_data=covid_data.drop(['day', 'month', 'year'], axis=1)
'''Convert dateRep to datetime object and set it as index'''
covid_data['dateRep']=pd.to_datetime(covid_data['dateRep'])
covid_data=covid_data.set_index(['dateRep'])
```

Data Analysis

1. Count the total number of cases and deaths in the dataset.

In [6]:

```
'''count total cases and deaths'''
total_cases=covid_data['cases'].sum()
total_deaths=covid_data['deaths'].sum()
print("Total cases:",total_cases)
print("Total deaths:",total_deaths)
```

```
Total cases: 9994560
Total deaths: 178247
```

Click Here for the Hint

2. Calculate the percentage of cases and deaths by country.

In [7]:

```
'''group data by country and calculate the total cases and deaths for each country'''
grouped_data=covid_data.groupby('countriesAndTerritories')[['cases', 'deaths']].sum()
'''calculate the percentage of total cases and deaths for each country'''
grouped_data['case_percentage']=(grouped_data['cases']/total_cases)*100
grouped_data['death_percentage']=(grouped_data['deaths']/total_deaths)*100
'''select the columns containing the case and death percentages and print them'''
percentage_data=grouped_data[['case_percentage', 'death_percentage']]
print(percentage_data)
```

```
                         case_percentage  death_percentage
countriesAndTerritories
Austria                         1.845164          1.079962
Belgium                         2.882758          1.512508
Bulgaria                        1.713292          4.191375
Croatia                         1.132296          1.395816
Cyprus                          0.377205          0.072371
Czechia                         4.214503          5.407665
Denmark                         0.692257          0.086958
Estonia                         0.629502          0.366907
Finland                         0.347789          0.099300
France                         20.219079         12.890540
Germany                        12.347297         10.287410
Greece                          2.103154          3.113657
Hungary                         3.718153          8.232958
Iceland                         0.005273          0.000561
Ireland                         0.420799          0.348954
Italy                          12.914405         15.903213
Latvia                          0.469375          0.421886
Liechtenstein                   0.004372          0.002244
Lithuania                       0.770819          0.573362
Luxembourg                      0.144719          0.098739
Malta                           0.075901          0.058346
Netherlands                     5.582867          1.152895
Norway                          0.540244          0.090324
Poland                         11.655981         16.813186
Portugal                        0.441200          0.396080
Romania                         2.757400          5.568677
Slovakia                        1.785721          2.889249
Slovenia                        0.635846          0.326513
Spain                           5.530238          5.803183
Sweden                          4.042389          0.815161
```

Click Here for the Hint

3. Find the country with the highest number of cases and deaths.

In [8]:

```
'''Country has the highest number of cases and deaths'''
highest_cases_country=grouped_data['cases'].idxmax()
highest_deaths_country=grouped_data['deaths'].idxmax()
'''highest cases and deaths'''
highest_cases=grouped_data['cases'].max()
highest_deaths=grouped_data['deaths'].max()
'''printing the values'''
print("Country has the highest number of cases:",highest_cases_country,"-->",highest_cases)
print("Country has the highest number of deaths:",highest_deaths_country,"-->",highest_deaths
```

```
Country has the highest number of cases: France --> 2020808
Country has the highest number of deaths: Poland --> 29969
```

Click Here for the Hint

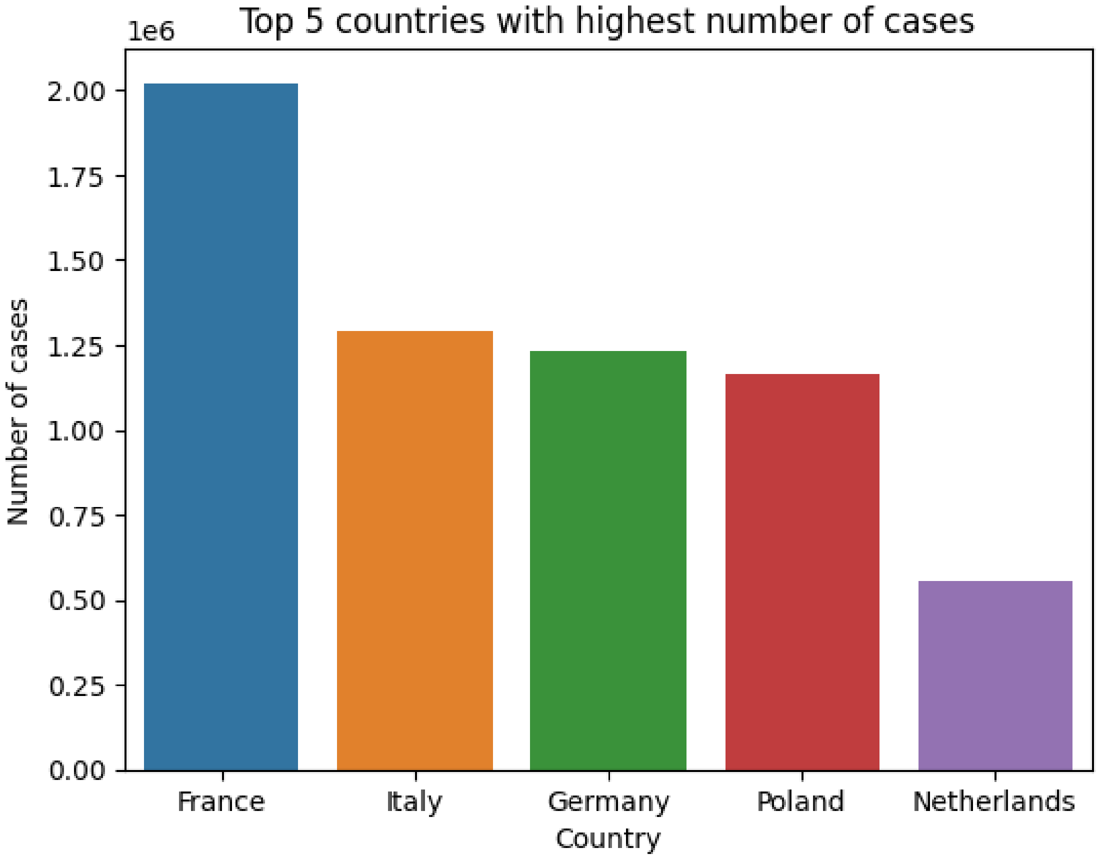
Data Visualization

1. Find top five countries in terms of cases, store them in a new dataframe and Visualize them

In [9]:

```
'''create new dataframe'''
new_df=covid_data[['cases', 'deaths', 'countriesAndTerritories']]
new_df.head(10)
'''group data by country and calculate the total cases and deaths for each country'''
group=new_df.groupby('countriesAndTerritories')[['cases', 'deaths']].sum()
'''sort the data by the number of cases in decending order and select the top 5 countries'''
top_5_countries_cases=group.sort_values(by='cases', ascending=False).head(5)
top_5_countries_cases=top_5_countries_cases.reset_index()
'''print the top 5 countries with the highest number of cases'''
print(top_5_countries_cases)
'''create barplot'''
sns.barplot(x='countriesAndTerritories', y='cases', data=top_5_countries_cases)
'''add labels to the plot'''
plt.title('Top 5 countries with highest number of cases')
plt.xlabel('Country')
plt.ylabel('Number of cases')
'''show the plot'''
plt.show()
```

```
  countriesAndTerritories    cases  deaths
0                  France  2020808   22977
1                   Italy  1290738   28347
2                 Germany  1234058   18337
3                  Poland  1164964   29969
4             Netherlands   557983    2055
```



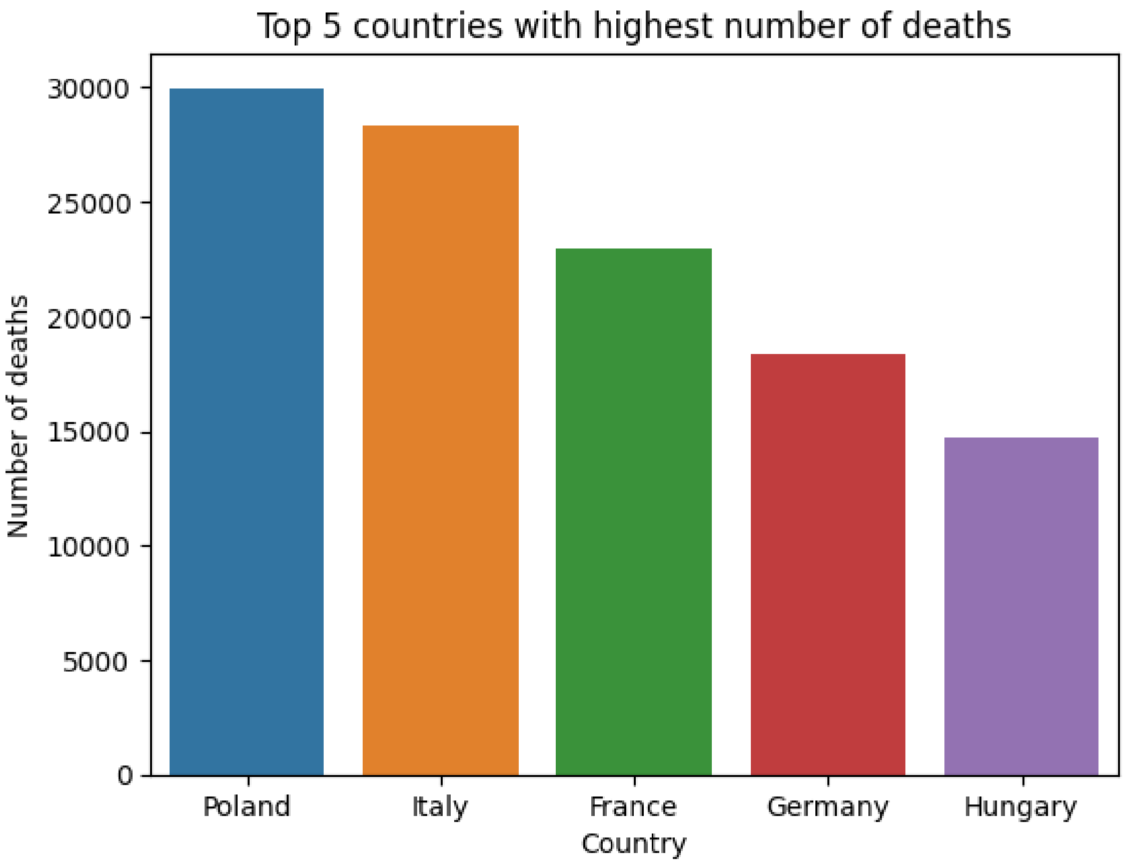
Click Here for the Hint

2. Find top five countries in terms of deaths, store them in a new dataframe and Visualize them

In [10]:

```
'''create new dataframe'''
new_df=covid_data[['cases', 'deaths', 'countriesAndTerritories']]
new_df.head(10)
'''group data by country and calculate the total cases and deaths for each country'''
group=new_df.groupby('countriesAndTerritories')[['cases', 'deaths']].sum()
'''sort the data by the number of deaths in decending order and select the top 5 countries'''
top_5_countries_deaths=group.sort_values(by='deaths', ascending=False).head(5)
top_5_countries_deaths=top_5_countries_deaths.reset_index()
'''print the top 5 countries with the highest number of deaths'''
print(top_5_countries_deaths)
'''create barplot'''
sns.barplot(x='countriesAndTerritories', y='deaths', data=top_5_countries_deaths)
'''add labels to the plot'''
plt.title('Top 5 countries with highest number of deaths')
plt.xlabel('Country')
plt.ylabel('Number of deaths')
'''show the plot'''
plt.show()
```

```
  countriesAndTerritories    cases  deaths
0                  Poland  1164964   29969
1                   Italy  1290738   28347
2                  France  2020808   22977
3                 Germany  1234058   18337
4                 Hungary   371613   14675
```



Click Here for the Hint

Model Development & Evaluation

In [15]:

```
'''Resample the data by date to get total cases and deaths by date'''
resampled_cases = covid_data.resample('D').sum()

'''Split the data into training and testing sets'''
X_train, X_test, y_train, y_test = train_test_split(resampled_cases[['cases', 'deaths']], resa

'''Scale the data using StandardScaler'''
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

'''Combine the scaled cases and deaths data into one array for both the training and testing c
train_data = pd.concat([pd.DataFrame(X_train), y_train.reset_index(drop=True)], axis=1).values
test_data = pd.concat([pd.DataFrame(X_test), y_test.reset_index(drop=True)], axis=1).values

'''Fit a GradientBoostingRegressor model on the training data'''
model = GradientBoostingRegressor()
model.fit(train_data[:, :-1], train_data[:, -1])

'''predictions using the trained model on the testing data'''
y_pred = model.predict(test_data[:, :-1])

'''model's performance using mean_squared_error and r2_score functions'''
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

'''Print the root mean squared error and R2 score to assess the model's performance'''
print('Root Mean Squared Error:', rmse)
print('R2 Score:', r2)
```

```
Root Mean Squared Error: 1597.5374964076198
R2 Score: 0.99853580309982
```

Click Here for the Hint