Project: Crawl and Analyze data Covid-19 from worldometers.info

In [1]:
```
import os
import requests
import pandas as pd
from datetime import datetime
from bs4 import BeautifulSoup as soup
import lxml.html as lh
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
import copy
pd.options.mode.chained_assignment = None  # default='warn'
```

I. Thu thập dữ liệu

In [2]:
```
url = "https://www.worldometers.info/coronavirus/"

response = requests.get(url)
response.status_code
```

Out[2]: 200

In [3]:
```
doc = lh.fromstring(response.content)
tr_elements = doc.xpath('//tr')
```

In [4]:
```
#For each row, store each first element (header) and an empty list
def find_col():
    col=[]
    col_name=[]
    i=0
    for t in tr_elements[0]:
        i+=1
        name=t.text_content()
        print('%d:"%s"'%(i,name))
        col_name.append(name)
        col.append((name,[]))
    return col,col_name
cols,cols_name= find_col()
```

```
1:"#"
2:"Country,Other"
3:"TotalCases"
4:"NewCases"
5:"TotalDeaths"
6:"NewDeaths"
7:"TotalRecovered"
8:"NewRecovered"
9:"ActiveCases"
10:"Serious,Critical"
11:"Tot Cases/1M pop"
12:"Deaths/1M pop"
13:"TotalTests"
14:"Tests/
1M pop
"
15:"Population"
16:"Continent"
17:"1 Caseevery X ppl"
18:"1 Deathevery X ppl"
19:"1 Testevery X ppl"
20:"New Cases/1M pop"
21:"New Deaths/1M pop"
22:"Active Cases/1M pop"
```

In [5]:
```
page = requests.get(url).text
page_soup = soup(page,"html.parser")
```

In [6]:
```
table = page_soup.find('tbody')
table_data = []
for row in table.findAll('tr'):
    row_data = []
    for cell in row.findAll('td'):
        row_data.append(cell.text)
    if(len(row_data) > 0):
        data_item=[]
        for i in range(len(cols)):
            data_item.append(row_data[i])
        table_data.append(data_item)
```

In [7]:
```
df_tmp= pd.DataFrame(table_data)
df_tmp.columns = cols_name
df_tmp
#df_2.set_index('#')
```

Out[7]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRec
0		\nNorth America\n	98,628,447	+70,217	1,461,680	+398	94,327,748	
1		\nAsia\n	148,372,557	+128,802	1,425,202	+297	126,347,228	+
2		\nSouth America\n	56,879,046	+22,752	1,294,971	+87	53,031,560	
3		\nEurope\n	192,058,836	+302,827	1,821,484	+977	178,127,670	+
4		\nOceania\n	7,337,184	+55,209	10,805	+79	6,849,320	
...	...	...	...	...	...	...	...	...
231	224	Niue	9				8	
232	225	Micronesia	7				1	
233	226	Nauru	5				3	
234	227	Saint Helena	2				2	
235	228	China	218,198	+362	5,128	+16	198,550	

236 rows × 22 columns

In [8]:
```
print(str(datetime.today())[:11])
```

```
2022-05-05
```

In [9]:
```
df_tmp.to_csv("data-"+str(datetime.today())[:10]+".csv",sep=',',mode = 'w', index
```

Ở đây, nhóm em chọn phân tích dữ liệu dựa trên ngày 1/5/2022

In [10]:
```
df = pd.read_csv("data-2022-05-01.csv")
df.head(10)
```

Out[10]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewReco
0	NaN	\nNorth America\n	98,326,638	222	1,459,866	1.0	94,167,720	
1	NaN	\nAsia\n	148,022,932	99,565	1,424,244	327.0	126,044,566	7
2	NaN	\nSouth America\n	56,802,343	NaN	1,294,462	NaN	52,927,991	
3	NaN	\nEurope\n	191,147,706	24,563	1,818,073	220.0	176,452,396	17
4	NaN	\nOceania\n	7,194,504	38,293	10,627	20.0	6,716,689	
5	NaN	\nAfrica\n	11,902,249	51	253,795	NaN	11,103,567	
6	NaN	\n\n	721	NaN	15	NaN	706	
7	NaN	World	513,397,093	162,694	6,261,082	568.0	467,413,635	25
8	1.0	USA	83,066,907	NaN	1,020,833	NaN	80,684,899	
9	2.0	India	43,079,188	NaN	523,843	NaN	42,536,253	

10 rows × 22 columns

II. Tiền xử lý dữ liệu

Tách các dòng số liệu của tổng khu vực và thế giới ra 1 dataframe khác, đặt tên là `df_continent` ."

In [11]:
```
df_continent = df[:8]
df_continent
```

Out[11]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewReco
0	NaN	\nNorth America\n	98,326,638	222	1,459,866	1.0	94,167,720	
1	NaN	\nAsia\n	148,022,932	99,565	1,424,244	327.0	126,044,566	7
2	NaN	\nSouth America\n	56,802,343	NaN	1,294,462	NaN	52,927,991	
3	NaN	\nEurope\n	191,147,706	24,563	1,818,073	220.0	176,452,396	17
4	NaN	\nOceania\n	7,194,504	38,293	10,627	20.0	6,716,689	
5	NaN	\nAfrica\n	11,902,249	51	253,795	NaN	11,103,567	
6	NaN	\n\n	721	NaN	15	NaN	706	
7	NaN	World	513,397,093	162,694	6,261,082	568.0	467,413,635	25

8 rows × 22 columns

In [12]:
```
df = df.iloc[8:,:].reset_index(drop = True)
```

In [13]:
```
df['#'] = df['#'].replace('',np.nan).fillna(0)
df['#'] =df['#'].astype('int64')
```

In [14]:
```
df
```

Out[14]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRec
0	1	USA	83,066,907	NaN	1,020,833	NaN	80,684,899	
1	2	India	43,079,188	NaN	523,843	NaN	42,536,253	
2	3	Brazil	30,448,236	NaN	663,551	NaN	29,519,204	
3	4	France	28,645,285	NaN	145,930	NaN	26,801,445	
4	5	Germany	24,770,313	8,713	135,913	13.0	22,356,000	
...	...	...	...	...	...	...	...	...
223	224	Niue	9	NaN		NaN	7	
224	225	Nauru	5	NaN		NaN	3	
225	226	Micronesia	3	NaN		NaN	1	
226	227	Saint Helena	2	NaN		NaN	2	
227	228	China	216,587	920	5,060	38.0	187,525	

228 rows × 22 columns

Kiểm tra giá trị các trường dữ liệu xem có hợp lý hay không?

In [15]:
```
df.dtypes
```

Out[15]:
```
#                       int64
Country,Other          object
TotalCases             object
NewCases               object
TotalDeaths            object
NewDeaths             float64
TotalRecovered         object
NewRecovered           object
ActiveCases            object
Serious,Critical       object
Tot Cases/1M pop       object
Deaths/1M pop          object
TotalTests             object
Tests/\n1M pop\n       object
Population             object
Continent              object
1 Caseevery X ppl      object
1 Deathevery X ppl     object
1 Testevery X ppl     float64
New Cases/1M pop       object
New Deaths/1M pop     float64
Active Cases/1M pop    object
dtype: object
```

Kiểm tra số lượng giá trị thiếu của từng trường thông tin

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
#                       0
Country,Other           0
TotalCases              0
NewCases              179
TotalDeaths             0
NewDeaths             201
TotalRecovered         15
NewRecovered          179
ActiveCases            15
Serious,Critical       77
Tot Cases/1M pop        2
Deaths/1M pop          10
TotalTests             16
Tests/\n1M pop\n       16
Population              0
Continent               2
1 Caseevery X ppl       2
1 Deathevery X ppl     10
1 Testevery X ppl      16
New Cases/1M pop      179
New Deaths/1M pop     201
Active Cases/1M pop     8
dtype: int64
```

Thực hiện chuyển kiểu dữ liệu do đa phần các cột cần sử dụng là kiểu dữ liệu số, nhưng ở dữ liệu gốc chúng lại mang kiểu dữ liệu khác.

In [17]:
```
def convertToFloat(df,col:str):
    #print(col)
    if col in ['Population','TotalDeaths']:
        df[col]= df[col].str.replace(',','')
        df[col]= df[col].str.replace(' ','')
        df[col]= pd.to_numeric(df[col])
    else: df[col]= [float(str(i).replace(",","")) for i in df[col]]
    return df[col]
```

In [18]:
```
def preprocessing(df):
    tmp = ['#','Country,Other','Continent']
    for c in df.columns:
        #print(c)
        if (c not in tmp) and (df[c].dtype == 'object'):
            #print(c)
            df[c] = convertToFloat(df,c)
    return df
```

In [19]:
```
df =  preprocessing(df)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228 entries, 0 to 227
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   #                    228 non-null    int64
 1   Country,Other        228 non-null    object
 2   TotalCases           228 non-null    float64
 3   NewCases             49 non-null     float64
 4   TotalDeaths          220 non-null    float64
 5   NewDeaths            27 non-null     float64
 6   TotalRecovered       213 non-null    float64
 7   NewRecovered         49 non-null     float64
 8   ActiveCases          213 non-null    float64
 9   Serious,Critical     151 non-null    float64
 10  Tot Cases/1M pop     226 non-null    float64
 11  Deaths/1M pop        218 non-null    float64
 12  TotalTests           212 non-null    float64
 13  Tests/
1M pop
       212 non-null    float64
 14  Population           226 non-null    float64
 15  Continent            226 non-null    object
 16  1 Caseevery X ppl    226 non-null    float64
 17  1 Deathevery X ppl   218 non-null    float64
 18  1 Testevery X ppl    212 non-null    float64
 19  New Cases/1M pop     49 non-null     float64
 20  New Deaths/1M pop    27 non-null     float64
 21  Active Cases/1M pop  220 non-null    float64
dtypes: float64(19), int64(1), object(2)
memory usage: 39.3+ KB
```

III. Phân tích dữ liệu

1. Phân tích đơn biến

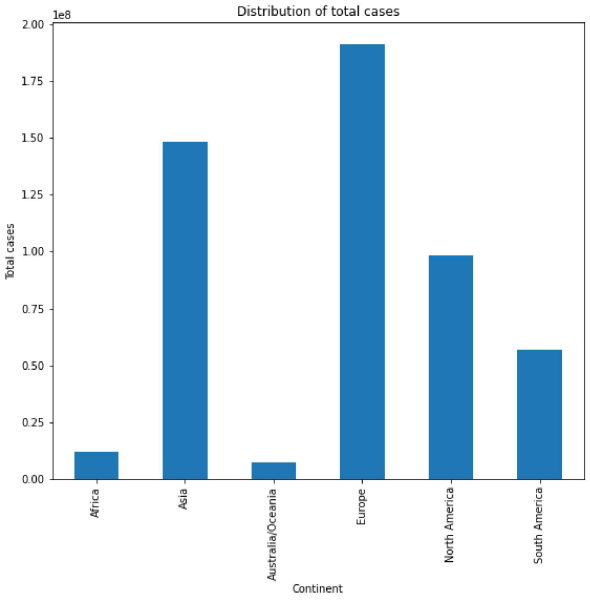
Tổng số ca nhiễm của các châu lục

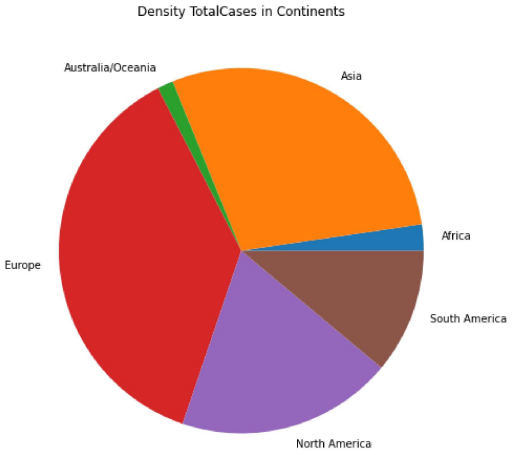
Ta thực hiện nhóm các nước theo tổng số ca nhiễm

In [20]:
```
df_total = df.groupby('Continent')['TotalCases'].agg('sum')
df_total
```

Out[20]:
```
Continent
Africa                11902249.0
Asia                 148022932.0
Australia/Oceania      7194504.0
Europe               191147706.0
North America         98326638.0
South America         56802343.0
Name: TotalCases, dtype: float64
```

In [21]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(20,8))
df_total.plot(kind='bar',ax=ax1)
ax1.set_ylabel('Total cases')
ax1.set_title('Distribution of total cases')
y = list(df_total.values)
mylabels = list(df_total.keys())
plt.pie(y, labels = mylabels)
plt.title("Density TotalCases in Continents")
plt.show()
plt.show()
```

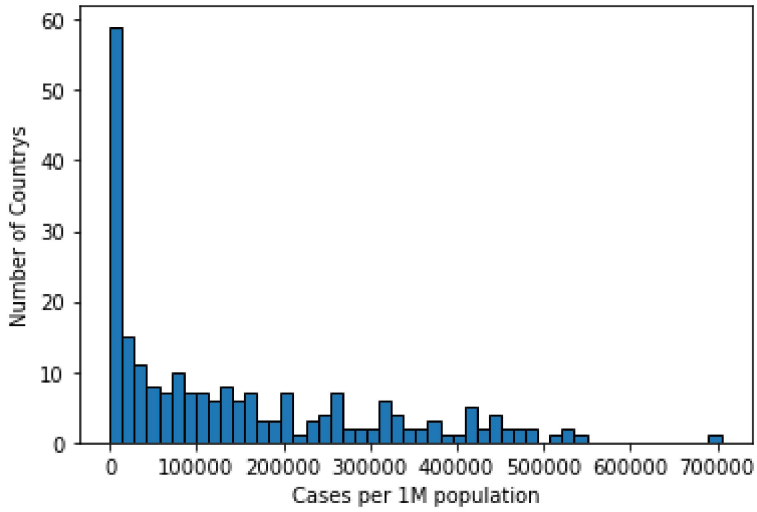




Nhận xét: Dựa vào biểu đồ trên ta thấy được tổng số ca nhiễm ở Châu Âu, theo sau là Châu Á và Bắc Mỹ. 3 Khu vực này hầu như chiếm toàn bộ tỉ trọng số ca nhiễm trên thế giới

Số ca nhiễm trên 1 triệu dân của các quốc gia

In [22]:
```
df_sub = copy.copy(df)
df_sub['Total_cases_per_1M_pop'] = round((df_sub["TotalCases"] / df_sub["Populati
data = list(df_sub['Total_cases_per_1M_pop'].dropna().astype(int))
plt.hist(data, bins = 50,edgecolor = "black")
plt.xlabel("Cases per 1M population")
plt.ylabel("Number of Countrys")
plt.show()
```



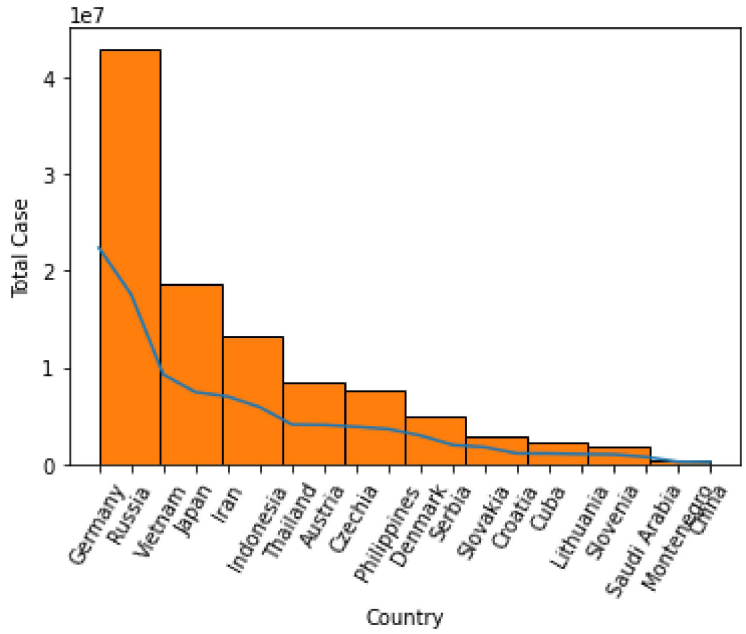
Nhận xét: Số ca nhiễm trên 1 triệu dân ở các quốc gia tập trung ở quảng dưới 100000, nhưng vẫn có 1 số quốc gia có lên đến 700000, các quốc gia này đáng báo động

2. Quan hệ đa biến

Tương quan giữa `TotalDeaths` , `TotalTests` và `TotalRecovered` :

Chuẩn hóa dữ liệu thành số thực và loại bỏ giá trị NaN

In [23]:
```
df1= copy.copy(df)
df1=df1.dropna()
plt.xticks(rotation=60)
plt.xlabel("Country")
plt.ylabel("Total Case")
# plt.plot(df1["Country,Other"],df1["TotalTests"])
plt.plot(df1["Country,Other"],df1["TotalRecovered"])
plt.hist(df1["Country,Other"],weights=df1["TotalCases"],edgecolor = "black")
plt.show()
```

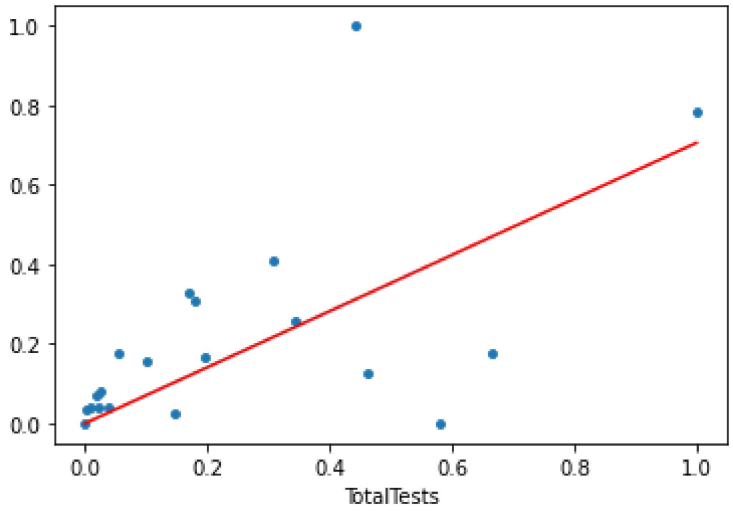


Sử dụng phép chuẩn hoá **Max - Min** để chuẩn hoá dữ liệu

In [24]:
```
def stMinMax(data):
    i=min(data)
    a=max(data)
    return [(k-i)/(a-i) for k in data]
```

Phân tích sự tương quan tổng số ca test `TotalTests` và tổng số ca phục hồi `TotalRecovered`

In [25]:
```
df_ln = df1[["TotalTests","TotalRecovered"]]
x=list(df_ln['TotalTests'].values)
y=list(df_ln['TotalRecovered'].values)
x=stMinMax(x)
y=stMinMax(y)
z=np.linspace(0,1,len(x))
model = sm.OLS(y,x).fit()
plt.plot(z,model.predict(z),c="red")
plt.scatter(x,y, s=15)
plt.xlabel("TotalTests")
plt.show()
```



Nhận xét:

Mối liên hệ giữa cột Tổng số ca nhiễm và tổng số test và tổng số ca phục hồi: Các quốc gia khi được tiến hàng test diện rộng sẽ có khả năng phát hiện được nhiều ca nhiễm hơn, chính thế có thể nhìn thấy số lượng test càng cao thì số lượng ca nhiễm phát hiện được càng nhiều. Điều này cũng giúp ích không ít trong việc điều trị bệnh. Vì nếu được phát hiện sớm, mức độ nhiễm bệnh của người dân còn thấp thì chính phủ dễ dàng điều trị hơn. Vì thế số ca phục hồi cũng sẽ cao hơn.

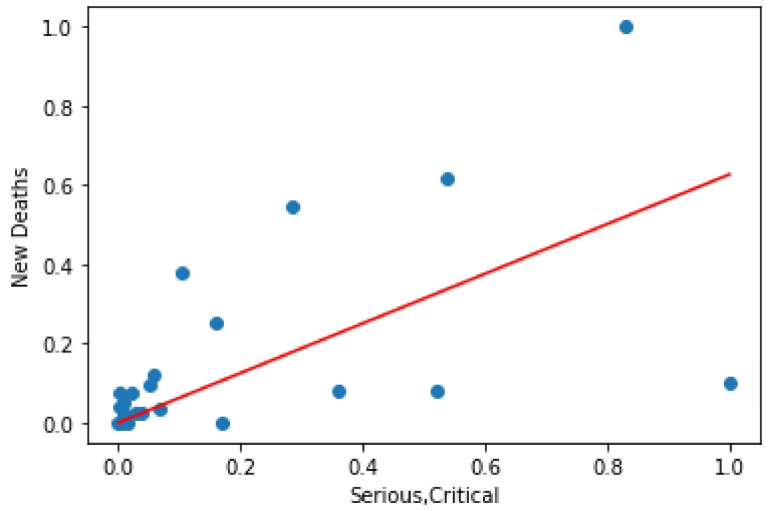
Tuy nhiên, điều này không hẳn là đúng 100%. Không thể nói các quốc gia có số lượng ca nhiễm ít là vì chính phủ chưa tiến hàng test diện rộng. mà là vì các quốc gia này thực hiện tốt biện pháp phòng bệnh, nên dù đã tiến hàng test rộng rãi thì số ca nhiễm vẫn ở mức thấp có thể kiểm soát được.

Phân tích sự tương quan giữa 2 trường dữ liệu `NewDeaths` và `Serious,Critical`

Lọc thông tin các cột cần sử dụng vào `df_sc`

In [26]:
```
df_sc = df[['Serious,Critical','NewDeaths']]
df_sc = df_sc.dropna()
df_sc = (df_sc-df_sc.min())/(df_sc.max()-df_sc.min())
```

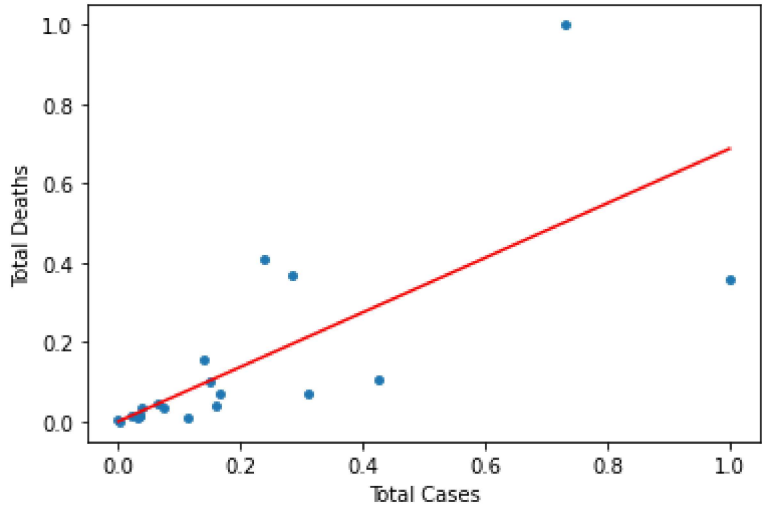
In [27]:
```
x = list(df_sc['Serious,Critical'].values)
y = list(df_sc['NewDeaths'].values)
##z = np.linspace(0,max(x),len(x))
z = np.linspace(0,1,len(df_sc['Serious,Critical']))
results=sm.OLS(y,x).fit()
plt.plot(z,results.predict(z),c='red')
plt.scatter(df_sc['Serious,Critical'],df_sc['NewDeaths'])
plt.xlabel('Serious,Critical')
plt.ylabel('New Deaths')
plt.show()
```



Nhận xét: Dựa vào kết quả mô hình hồi quy tuyến tính OLS xây dựng được, ta thấy được sự tương quan giữa 2 biến `Serious,Critical` và `NewDeaths` , số lượng trường hợp bệnh nhân gặp nguy kịch càng tăng thì số lượng người chết cũng tỉ lệ thuận theo.

Tương quan giữa tổng số ca nhiễm `TotalCases` và tổng số người chết `TotalDeaths`

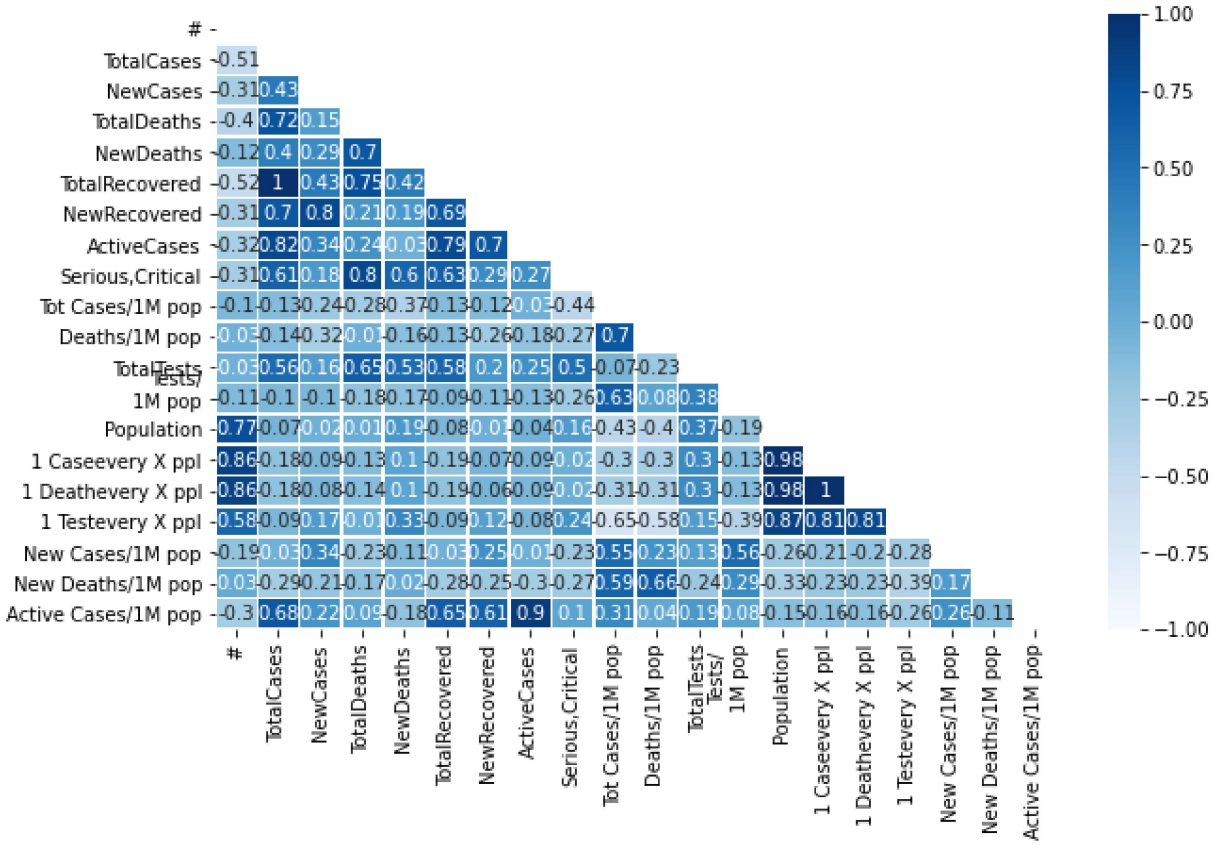
In [28]:
```
df_cd = copy.copy(df)
df_cd = df1[["TotalCases","TotalDeaths"]]
df_cd = df_cd.dropna()
x = list(df_cd.TotalCases.values)
y = list(df_cd.TotalDeaths.values)
x = stMinMax(x)
y = stMinMax(y)
z = np.linspace(0,1,len(x))
model = sm.OLS(y,x).fit()
plt.plot(z,model.predict(z),c= "red")
plt.xlabel("Total Cases")
plt.ylabel("Total Deaths")
plt.scatter(x,y, s=15)
plt.show()
```



Tương quan giữa các biến trong dataframe

In [29]:

```
df1_corr = df1.corr()
ones_corr = np.ones_like(df1_corr, dtype= bool)
mask = np.triu(ones_corr)
fig , ax = plt.subplots(figsize =(10,6))
sns.heatmap(round(df1_corr,2), mask = mask,annot = True,
            cmap ="Blues", linecolor="white", linewidths = 0.5,
            vmin = -1 , vmax = 1)
plt.show()
```



Nhận xét: Qua heatmap ma trận tương quan giữa các biến trong dataset, ta có thể nhận ra các mối quan hệ giữa các trường dữ liệu hầu như là độc lập và đồng biến