PSM Final Assessment - COVID 19 dataset

Sowrabha Ravishankar

In [1]:

```
#Importing Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [2]:

```
#Reading the original file using pandas
df_original = pd.read_csv("./owid-covid-data.csv")
```

In [3]:

```
original = df_original.head()
```

In [4]:

```
#Converting it to csv and saving the file for further use
original.to_csv("./original.csv")
```

In [5]:

```
#Getting to know number of columns and rows
df_original.shape
```

Out[5]:

```
(20976, 32)
```

In [6]:

```
#To know the entire dataset
describe = df_original.describe()
#describe
```

In [7]:

```
#List of missing values in each column
print(df_original.isnull().sum())
df_original.shape
```

```
iso_code                              64
location                               0
date                                   0
total_cases                            0
new_cases                              0
total_deaths                           0
new_deaths                             0
total_cases_per_million              385
new_cases_per_million                385
total_deaths_per_million             385
new_deaths_per_million               385
total_tests                        15345
new_tests                          15940
total_tests_per_thousand           15345
new_tests_per_thousand             15940
new_tests_smoothed                 14838
new_tests_smoothed_per_thousand    14838
tests_units                        14240
stringency_index                    4543
population                            64
population_density                   934
median_age                          1911
aged_65_older                       2169
aged_70_older                       2007
gdp_per_capita                      2178
extreme_poverty                     8509
cvd_death_rate                      1992
diabetes_prevalence                 1293
female_smokers                      5543
male_smokers                        5711
handwashing_facilities             12657
hospital_beds_per_100k              3482
dtype: int64
```

Out[7]:

```
(20976, 32)
```

In [8]:

```
#Sum of missing values in the dataset
missing_values = df_original.isnull().sum()
missing_values.sum()
#print(missing_values)
```

Out[8]:

```
161083
```

In [9]:

```
#After cleaning and dropping off columns using df.drop, reading new file
df = pd.read_csv("./Covid - 19 - missing values.csv")
```

In [10]:

```
df.head() #Sample of our dataset
```

Out[10]:

	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	r
0	Aruba	2020-03-13	2	2	0	0	18.733	
1	Aruba	2020-03-20	4	2	0	0	37.465	
2	Aruba	2020-03-24	12	8	0	0	112.395	
3	Aruba	2020-03-25	17	5	0	0	159.227	
4	Aruba	2020-03-26	19	2	0	0	177.959	

In [11]:

```
print(df.isnull().sum()) #No missing values found after dropping all the unwanted columns
```

```
location                    0
date                        0
total_cases                 0
new_cases                   0
total_deaths                0
new_deaths                  0
total_cases_per_million     0
new_cases_per_million       0
total_deaths_per_million    0
new_deaths_per_million      0
population                  0
median_age                  0
aged_65_older               0
aged_70_older               0
diabetes_prevalence         0
dtype: int64
```

In [12]:

```
df.shape #gettimg number of columns and rows
```

Out[12]:

```
(19356, 15)
```

In [13]:

```
df.columns #list of columns in the data set
```

Out[13]:

```
Index(['location', 'date', 'total_cases', 'new_cases', 'total_deaths',
       'new_deaths', 'total_cases_per_million', 'new_cases_per_million',
       'total_deaths_per_million', 'new_deaths_per_million', 'population',
       'median_age', 'aged_65_older', 'aged_70_older', 'diabetes_prevalenc
e'],
      dtype='object')
```

In [14]:

```
#Latest file with Month column added to it!
df = pd.read_csv("./Covid - 19 - Latest.csv")
df.tail()
```

Out[14]:

	location	date	Month	total_cases	new_cases	total_deaths	new_deaths	total_cases_p
19351	World	2020-05-30	May	5900530	123018	364895	4804	
19352	World	2020-05-31	May	6028326	127796	368945	4050	
19353	World	2020-06-01	Jun	6136294	107968	372668	3723	
19354	World	2020-06-02	Jun	6245638	109344	376427	3759	
19355	World	2020-06-03	Jun	6348900	103262	380810	4383	

In [15]:

```
group = df.groupby('Month')#Groupping by month and finding its mean
group.mean().head()
```

Out[15]:

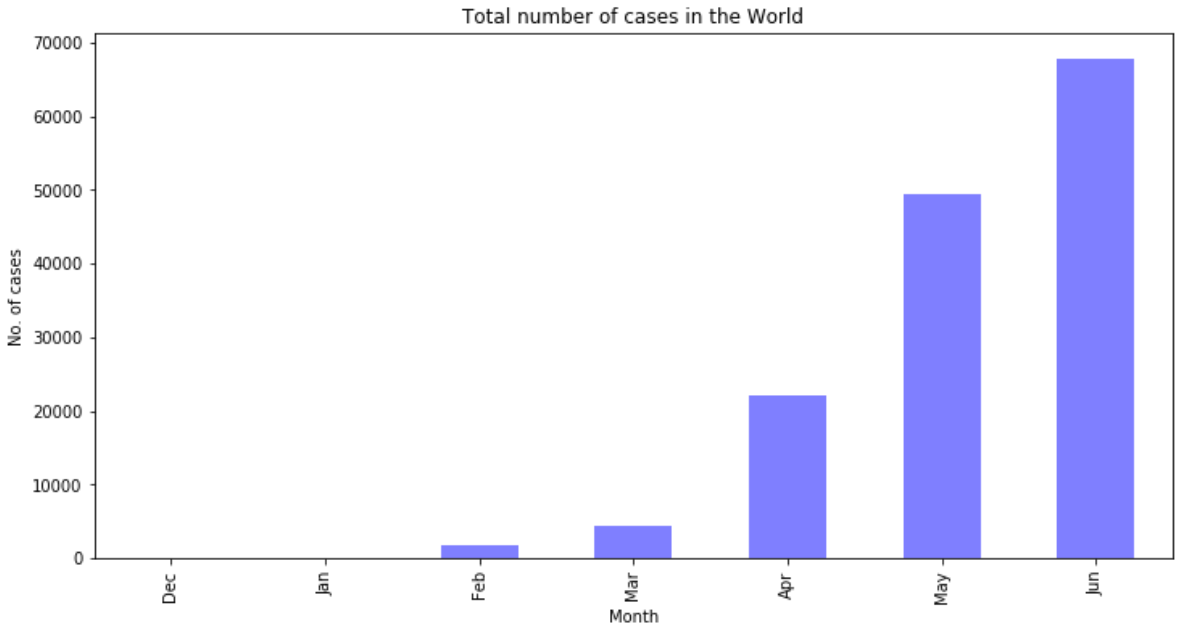
Month	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases
Apr	22218.429417	864.242738	1493.568918	70.061610	385.213334	
Dec	0.843750	0.843750	0.000000	0.000000	0.000344	
Feb	1680.044938	77.508781	46.559401	2.793905	1.118183	
Jan	38.679597	9.868514	0.889673	0.214610	0.024168	
Jun	67891.357664	1169.748175	4071.412409	43.297445	1114.475511	

In [16]:

```
#Plotting by grouping month with respect to months
group = df.groupby('Month')
total_cases = group['total_cases'].mean().sort_values(ascending=True)

# Equivalent way to get average_salary
group = df['total_cases'].groupby(df['Month'])
total_cases = group.mean().sort_values(ascending=True)

total_cases.plot(kind='bar', figsize=(12, 6), color='blue', alpha=0.5);
plt.xlabel('Month')
plt.ylabel('No. of cases')
plt.title('Total number of cases in the World')
#https://janakiev.com/blog/pandas-groupby/
```

Out[16]:

```
Text(0.5, 1.0, 'Total number of cases in the World')
```



In [17]:

```
#reading only the world data from the dataset
df_world = (df['location'] == 'World')
df_world.tail()
```

Out[17]:

```
19351    True
19352    True
19353    True
19354    True
19355    True
Name: location, dtype: bool
```

In [18]:

```
df_world = pd.read_csv("./World data.csv") # reading the world data csv file after saving i
```

In [19]:

```
df_world.shape
```

Out[19]:

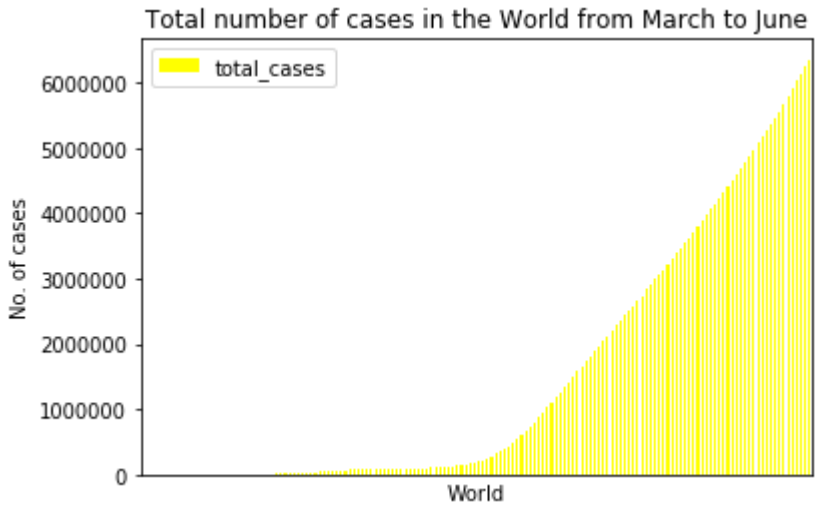
```
(156, 16)
```

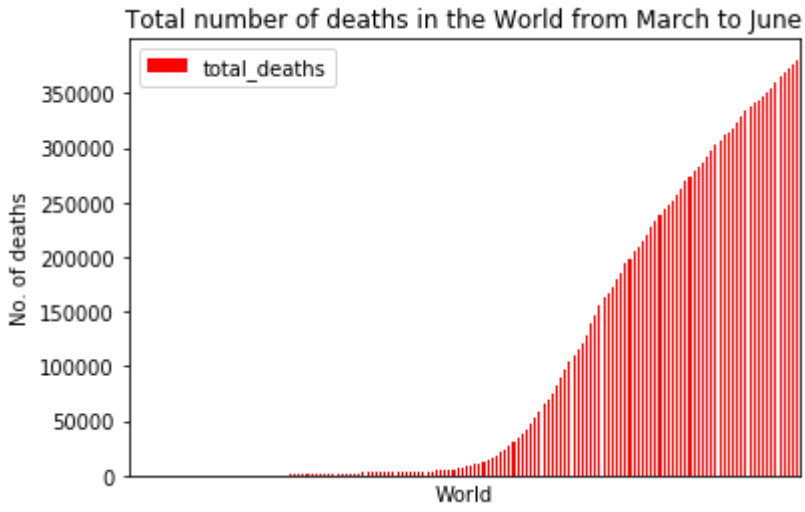
In [20]:

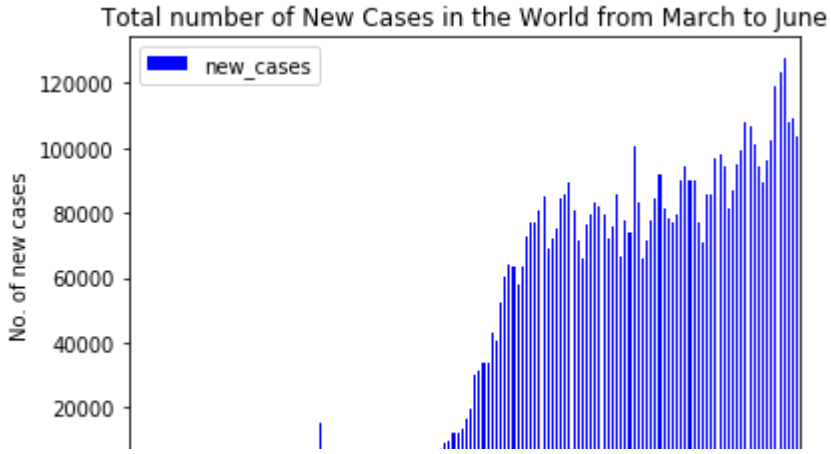
```
#Plotting Corona cases/deaths in the world
df_world = pd.read_csv("./World data.csv", parse_dates = ['date'], index_col = 'date')
#df_world.head()
df_world.plot(kind = 'bar', x = 'location', y = 'total_cases', color = 'yellow')
plt.xticks([])
plt.xlabel('World')
plt.ylabel('No. of cases')
plt.title('Total number of cases in the World from March to June')
df_world.plot(kind = 'bar', x = 'location', y = 'total_deaths', color = 'red')
plt.xticks([])
plt.xlabel('World')
plt.ylabel('No. of deaths')
plt.title('Total number of deaths in the World from March to June')
df_world.plot(kind = 'bar', x = 'location', y = 'new_cases', color = 'blue')
plt.xticks([])
plt.xlabel('World')
plt.ylabel('No. of new cases')
plt.title('Total number of New Cases in the World from March to June')
```

Out[20]:

```
Text(0.5, 1.0, 'Total number of New Cases in the World from March to June')
```







In [21]:

```
#reading only the last reading of every location in the dataset
Total_cases_World = pd.read_csv("./Covid - 19 - June 3rd data.csv")
```

In [22]:

```
Total_cases_World.shape
```

Out[22]:

```
(181, 16)
```

In [23]:

```
#Top_20 countries
Total_cases_World = pd.read_csv("./Covid - 19 - June 3rd data - latest.csv")
Top_20 = Total_cases_World.sort_values(by=['total_cases'], ascending=False).head(20)
Top_20
Top_20.head(10)
```

Out[23]:

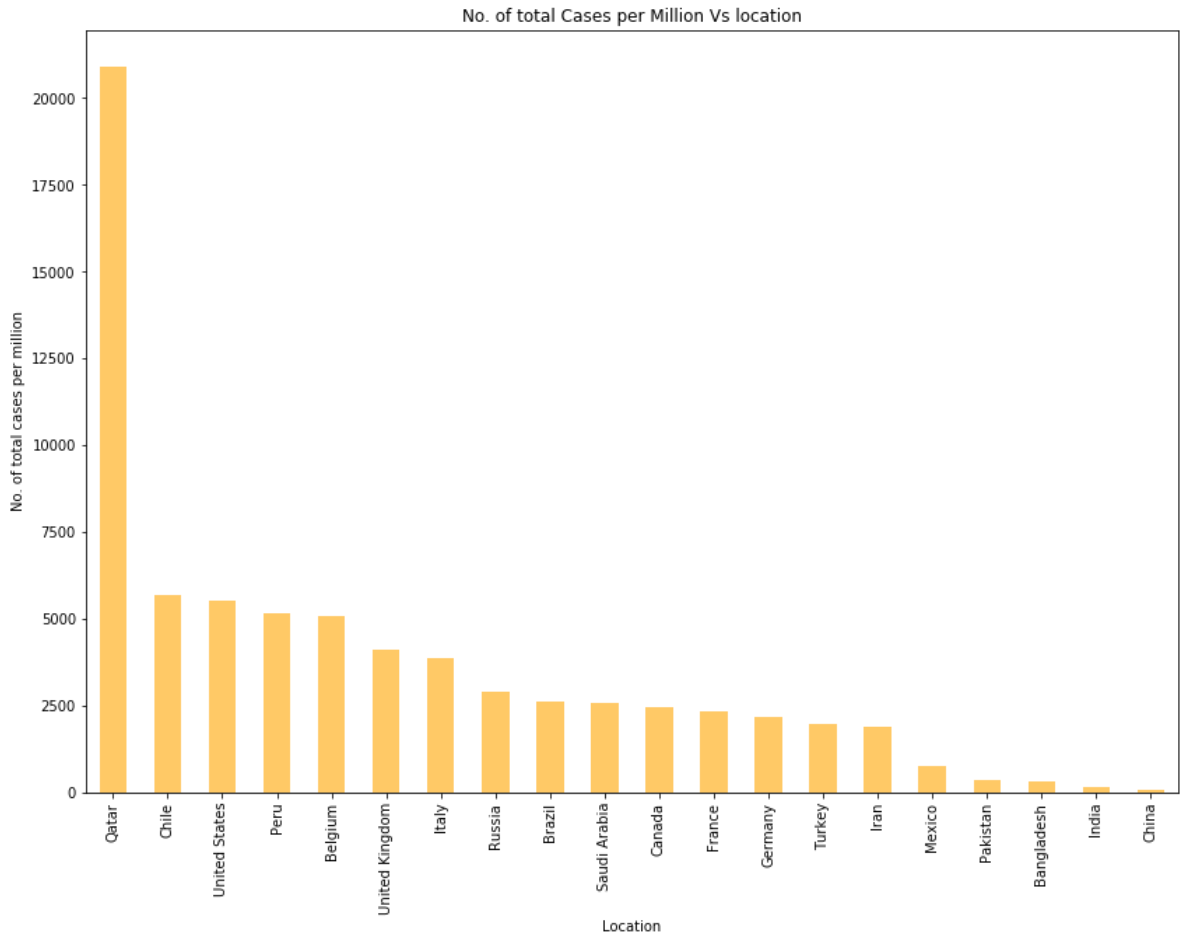
	location	date	Month	total_cases	new_cases	total_deaths	new_deaths	total_cases_pe
171	United States	2020-06-03	Jun	1831821	20544	106181	1034	
23	Brazil	2020-06-03	Jun	555383	28936	31199	1262	
141	Russia	2020-06-03	Jun	423741	0	5037	0	
59	United Kingdom	2020-06-03	Jun	277985	1653	39369	324	
82	Italy	2020-06-03	Jun	233515	318	33530	55	
76	India	2020-06-03	Jun	207615	8909	5815	217	
45	Germany	2020-06-03	Jun	182370	342	8551	29	
130	Peru	2020-06-03	Jun	170039	0	4634	0	
166	Turkey	2020-06-03	Jun	165555	786	4585	22	
78	Iran	2020-06-03	Jun	157562	3117	7942	64	

In [24]:

```
#Plotting total number of cases per milllion by location
group = Top_20.groupby('location')
total_cases = group['total_cases_per_million'].mean().sort_values(ascending=False)

# Equivalent way to get average_salary
group = Top_20['total_cases_per_million'].groupby(Top_20['location'])
total_cases = group.mean().sort_values(ascending=False)

total_cases.plot(kind='bar', figsize=(14, 10), color='orange', alpha=0.6);
plt.xlabel('Location')
plt.ylabel('No. of total cases per million')
plt.title('No. of total Cases per Million Vs location')
```

Out[24]:

```
Text(0.5, 1.0, 'No. of total Cases per Million Vs location')
```

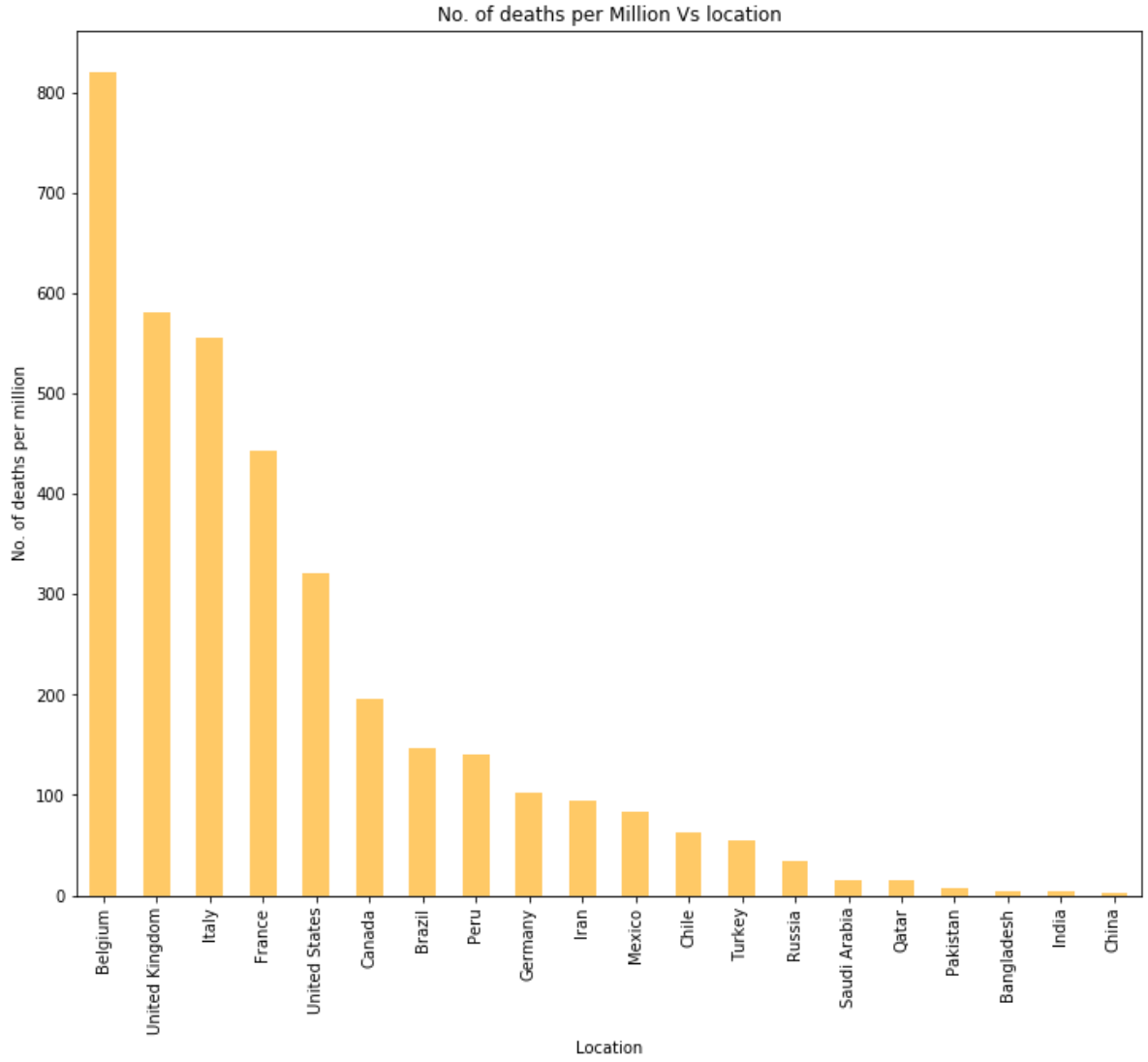


In [25]:

```
# Graph of total number of deaths per million by location
group = Top_20.groupby('location')
total_cases = group['total_deaths_per_million'].mean().sort_values(ascending=False)

# Equivalent way to get average_salary
group = Top_20['total_deaths_per_million'].groupby(Top_20['location'])
total_cases = group.mean().sort_values(ascending=False)

total_cases.plot(kind='bar', figsize=(12, 10), color='orange', alpha=0.6);
plt.xlabel('Location')
plt.ylabel('No. of deaths per million')
plt.title('No. of deaths per Million Vs location')
```

Out[25]:

```
Text(0.5, 1.0, 'No. of deaths per Million Vs location')
```

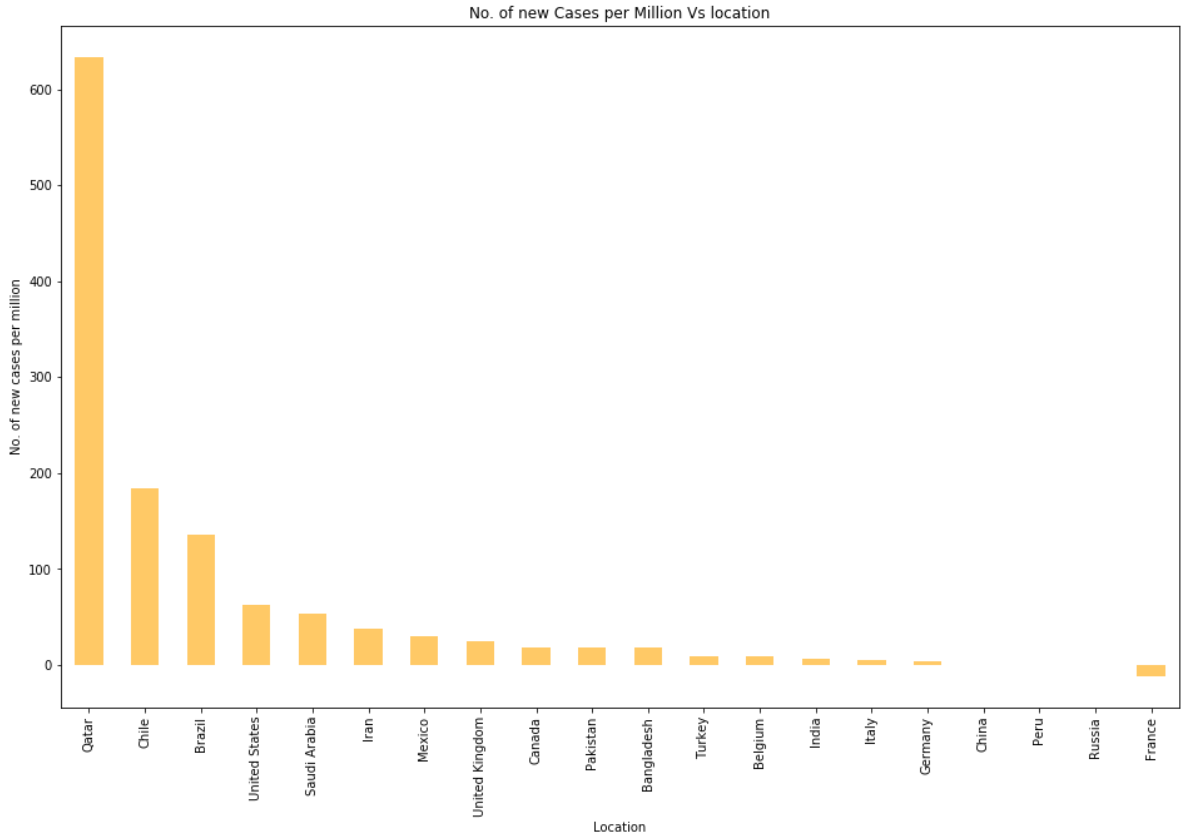


In [26]:

```
#Graph of total number of new cases per million by location
group = Top_20.groupby('location')
total_cases = group['new_cases_per_million'].mean().sort_values(ascending=False)

# Equivalent way to get average_salary
group = Top_20['new_cases_per_million'].groupby(Top_20['location'])
total_cases = group.mean().sort_values(ascending=False)

total_cases.plot(kind='bar', figsize=(16, 10), color='orange', alpha=0.6);
plt.xlabel('Location')
plt.ylabel('No. of new cases per million')
plt.title('No. of new Cases per Million Vs location')
```

Out[26]:

```
Text(0.5, 1.0, 'No. of new Cases per Million Vs location')
```

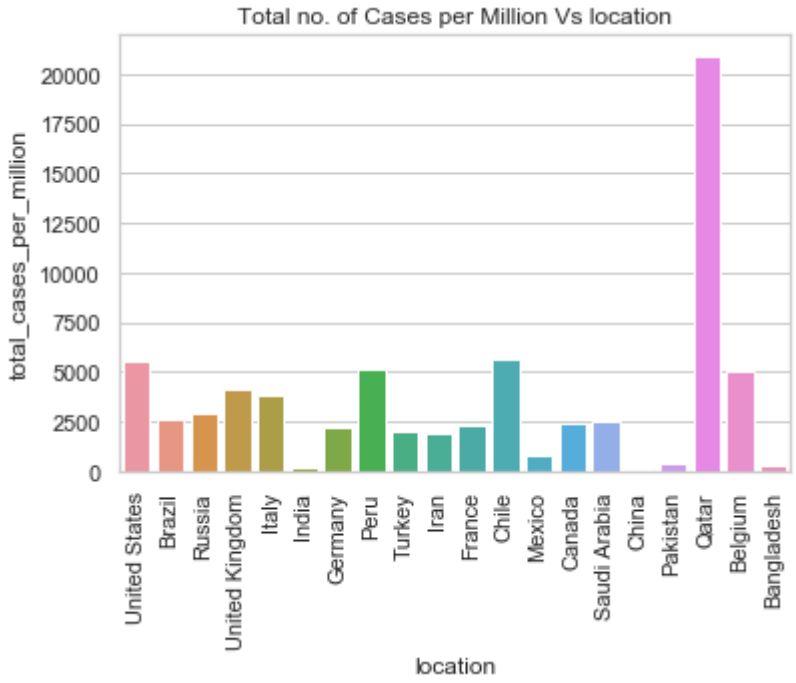


In [27]:

```
#Plotting total number of cases per milllion by location
import seaborn as sns
sns.set(style = "whitegrid")
ax = sns.barplot(x = "location", y = "total_cases_per_million",data = Top_20)
plt.xticks(rotation = 90)
plt.title('Total no. of Cases per Million Vs location')
```

Out[27]:

```
Text(0.5, 1.0, 'Total no. of Cases per Million Vs location')
```

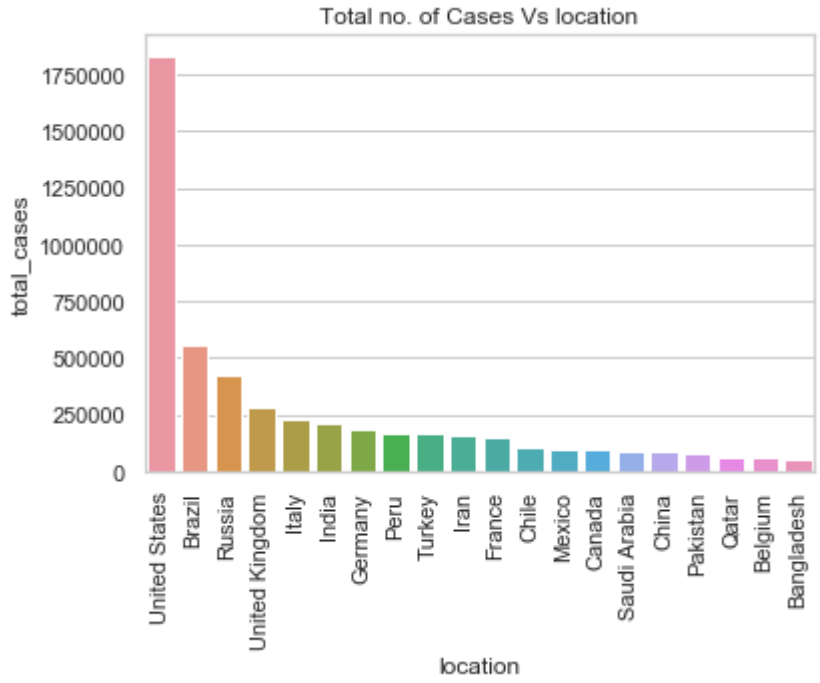


In [28]:

```
#Plot of Total number of cases bylocation
import seaborn as sns
sns.set(style = "whitegrid")
ax = sns.barplot(x = "location", y = "total_cases",data = Top_20)
plt.xticks(rotation = 90)
plt.title('Total no. of Cases Vs location')
```

Out[28]:

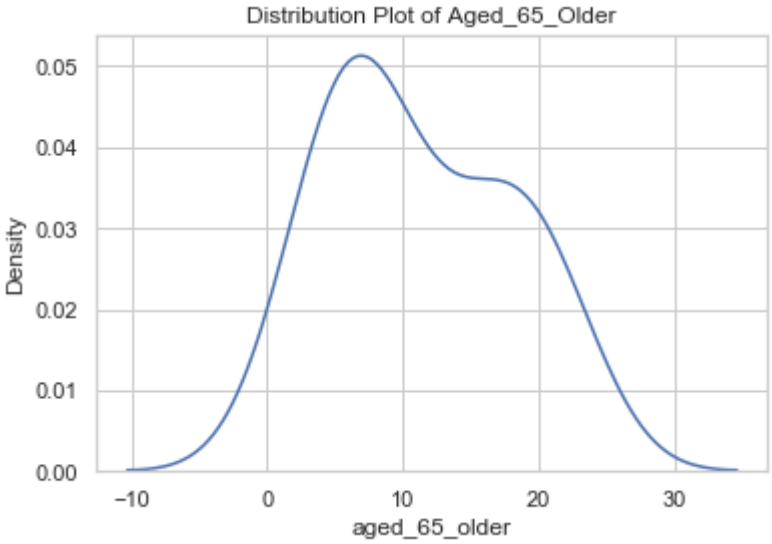
```
Text(0.5, 1.0, 'Total no. of Cases Vs location')
```



Distribution plots

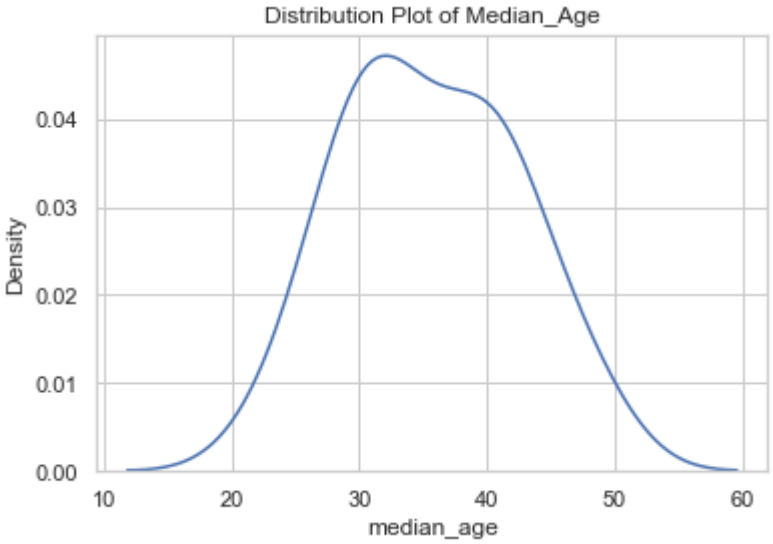
In [29]:

```
#Distribution plot of aged_65_older
sns.distplot(Top_20['aged_65_older'], bins = 0.5, hist = False)
plt.title('Distribution Plot of Aged_65_Older')
plt.ylabel('Density')
plt.savefig('aged_65.png', dpi = 300)
plt.show()
```



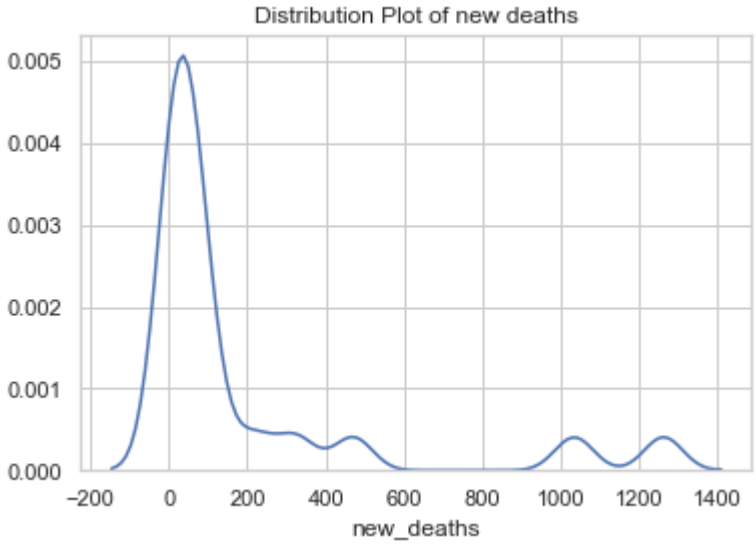
In [30]:

```
#Distribution plot of mEdian Age
sns.distplot(Top_20['median_age'], bins = 0.5, hist = False)
plt.title('Distribution Plot of Median_Age')
plt.ylabel('Density')
plt.savefig('median age.png', dpi = 300)
plt.show()
```



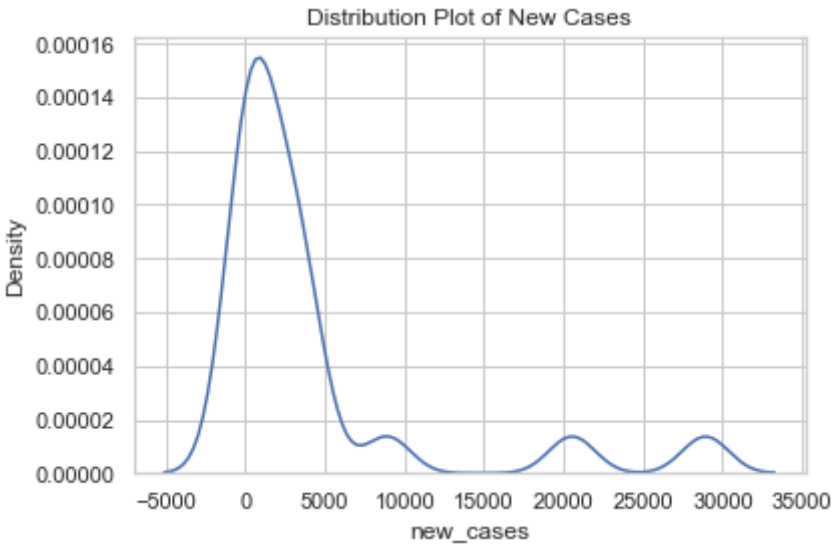
In [31]:

```
# Distribution plot of new deaths
sns.distplot(Top_20['new_deaths'], hist = False)
plt.title('Distribution Plot of new deaths')

plt.savefig('death_total.png', dpi = 300)
plt.show()
```



In [32]:

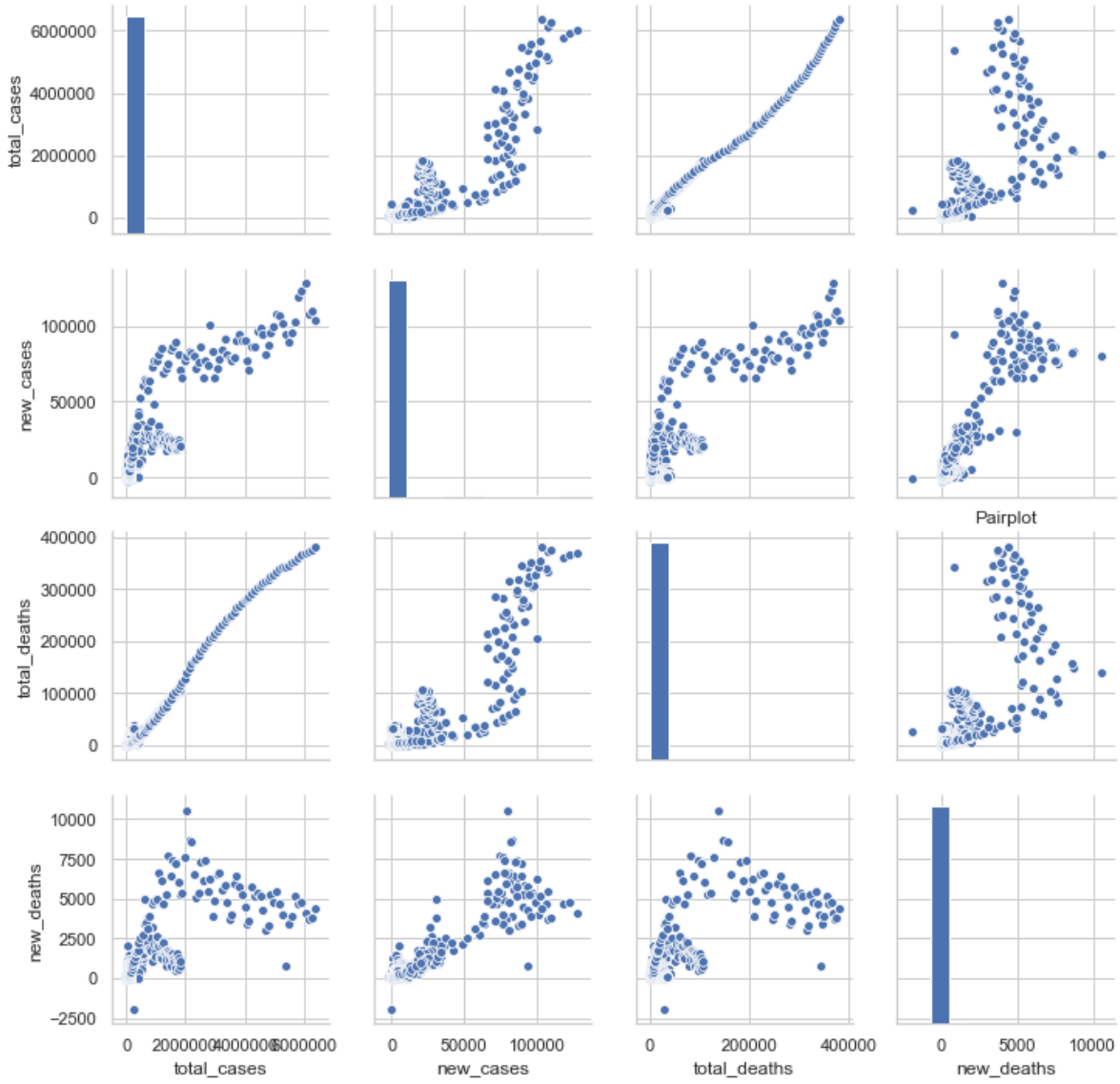
```
#Distribution plot of new cases
sns.distplot(Top_20['new_cases'], hist = False)
plt.title('Distribution Plot of New Cases')
plt.ylabel('Density')
plt.savefig('new-cases_density.png', dpi = 300)
plt.show()
# more of data points having 0 - 5000 cases per day - more no. of days having 0-5000 cases
```



Pairplot

In [33]:

```
#Pairplot
import seaborn as sns
import matplotlib.pyplot as plt

# Basic correlogram
sns.pairplot(df, vars = ["total_cases", "new_cases", "total_deaths", "new_deaths"])
plt.title('Pairplot')
plt.show()
#fig.savefig('distplot.png')
```



In [34]:

```
#Reading the file after all modifications
df = pd.read_csv("./Covid - 19 - Final.csv", parse_dates = ['date'], index_col = 'date')
```

In [35]:

```
df_world = pd.read_csv("./World data.csv")
```

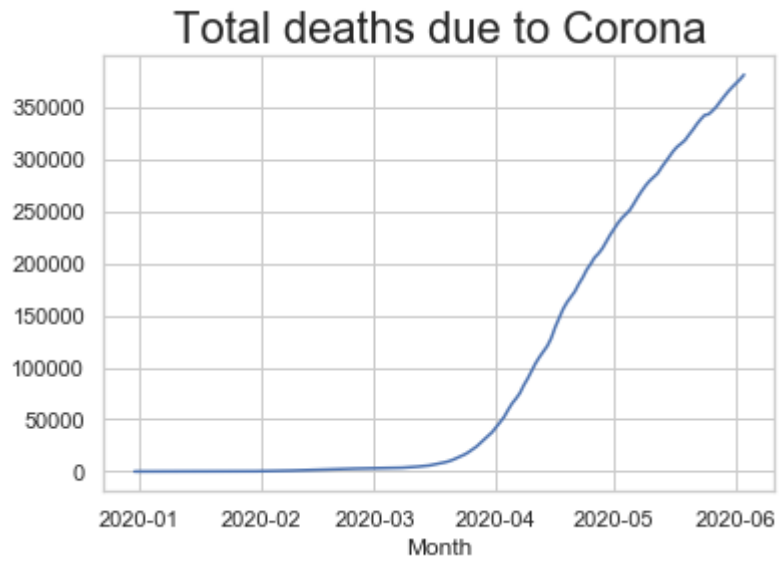
In [36]:

```
#Parsing dates and making date as index
df_world = pd.read_csv("./World data.csv", parse_dates = ['date'], index_col = 'date')
```

Time Series Analysis

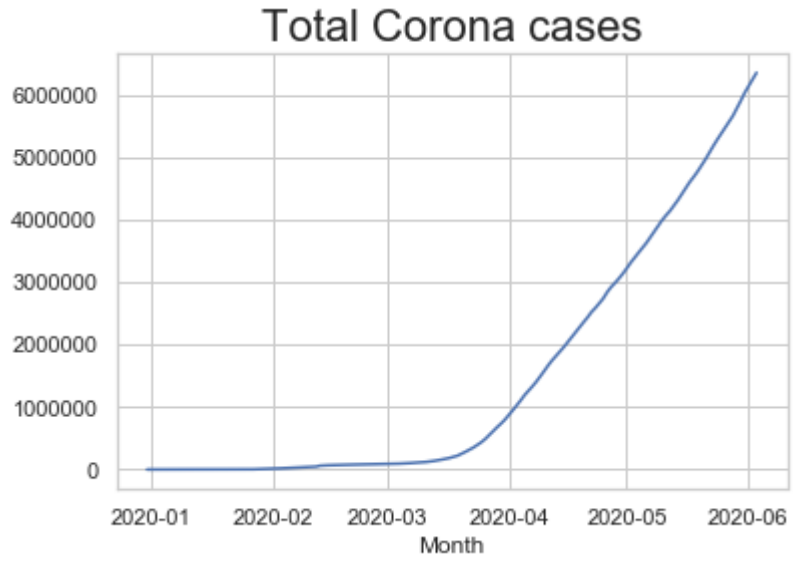
In [37]:

```
#Time series plots
plt.title('Total deaths due to Corona', fontsize=22)
plt.xlabel('Month')
plt.plot(df_world['total_deaths'])
plt.savefig('total deaths.png', dpi = 300)
```



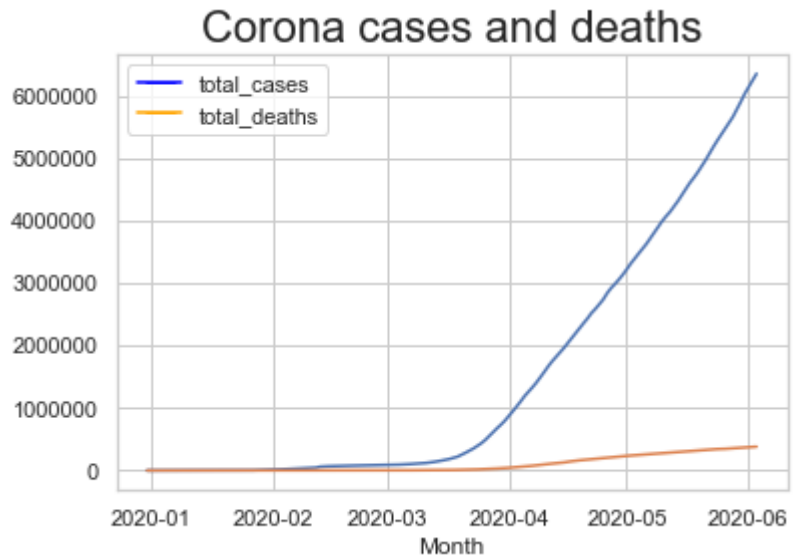
In [38]:

```
#Total Corona cases plot
plt.title('Total Corona cases', fontsize=22)
plt.xlabel('Month')
plt.plot(df_world['total_cases'])
plt.savefig('total cases.png', dpi = 300)
```



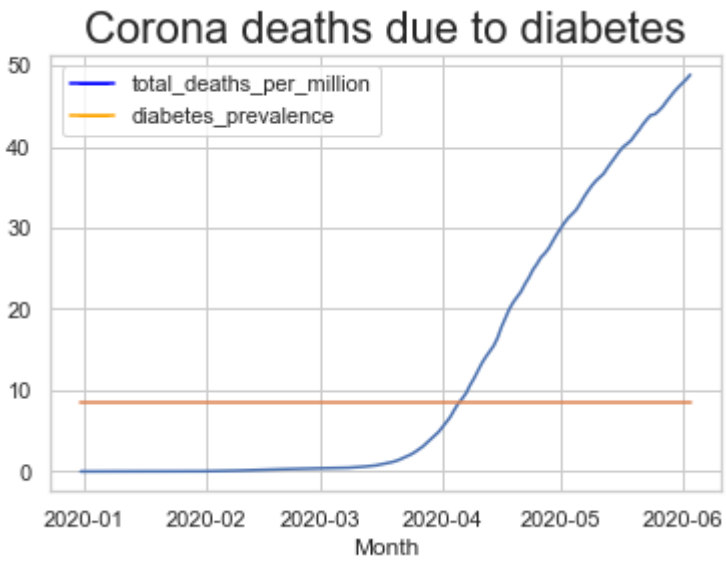
In [39]:

```
#Total corona cases and deaths
import matplotlib.lines as mlines
blue_line = mlines.Line2D([], [], color='blue', marker='_',
                          markersize=15, label='total_cases')
orange_line = mlines.Line2D([], [], color='orange', marker='_',
                          markersize=15, label='total_deaths')


plt.plot(df_world[['total_cases', 'total_deaths']])
plt.title('Corona cases and deaths', fontsize=22)
plt.xlabel('Month')
plt.legend(handles=[blue_line, orange_line])
plt.savefig('cases and deaths.png', dpi = 300)
```



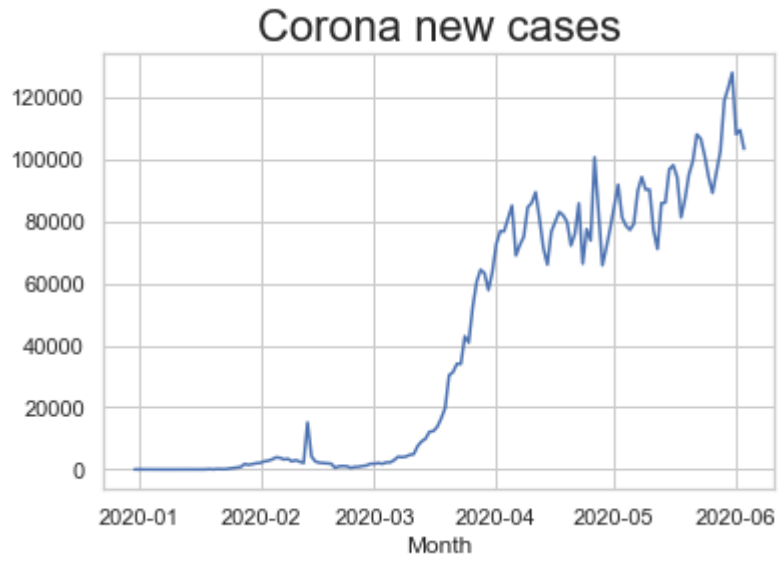
In [40]:

```
#Deaths due to diabetes
import matplotlib.lines as mlines
blue_line = mlines.Line2D([], [], color='blue', marker='_',
                          markersize=15, label='total_deaths_per_million')
orange_line = mlines.Line2D([], [], color='orange', marker='_',
                          markersize=15, label='diabetes_prevalence')


plt.plot(df_world[['total_deaths_per_million', 'diabetes_prevalence']])
plt.title('Corona deaths due to diabetes', fontsize=22)
plt.xlabel('Month')
plt.legend(handles=[blue_line, orange_line])
plt.figsize = [20, 4]
plt.savefig('deaths due to diabetes.png', dpi = 300)
```



In [41]:

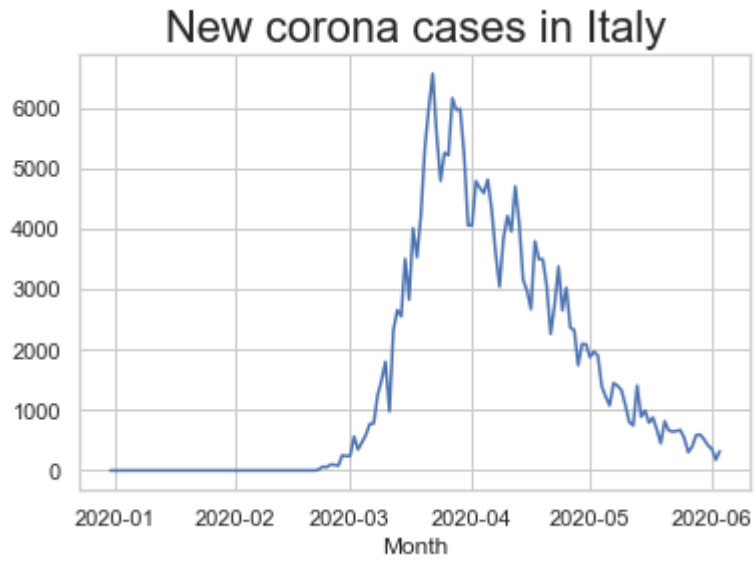
```
#Corona new cases
plt.title('Corona new cases', fontsize=22)
plt.xlabel('Month')
plt.plot(df_world['new_cases'])
plt.figsize = [16,4]
plt.savefig('new cases.png', dpi = 300)
```



New cases by countries

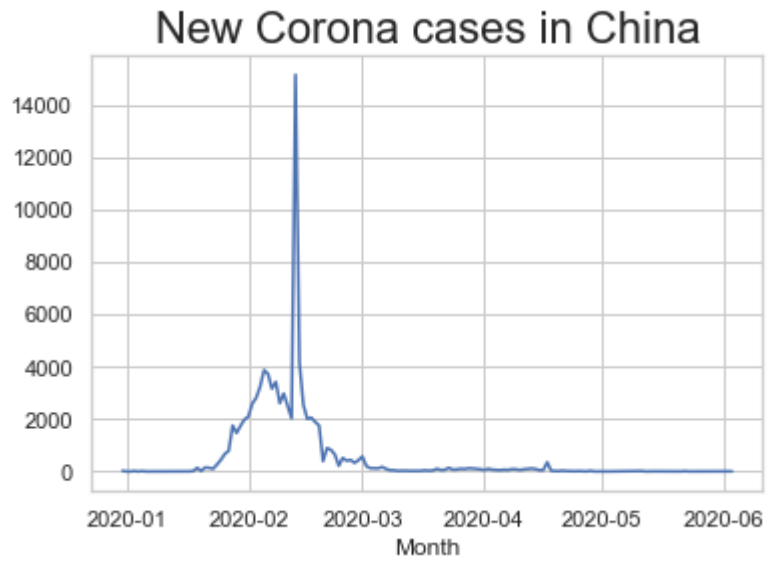
In [42]:

```
plt.title('New corona cases in Italy', fontsize=22)
plt.xlabel('Month')
df_Italy = df[df['location'] == 'Italy']
#print(df_Italy)
plt.plot(df_Italy['new_cases'])
plt.savefig('Italy cases.png', dpi = 300)
```



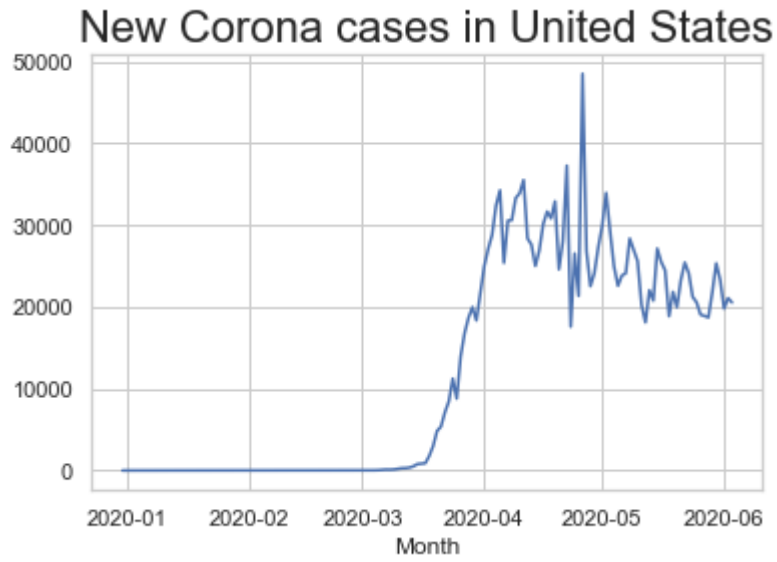
In [43]:

```
plt.title('New Corona cases in China', fontsize=22)
plt.xlabel('Month')
df_China = df[df['location'] == 'China']

plt.plot(df_China['new_cases'])
plt.savefig('China cases.png', dpi = 300)
```



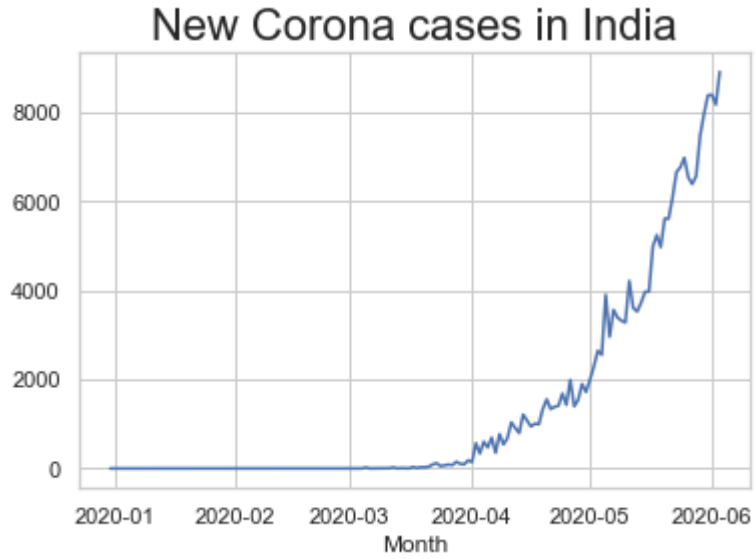
In [44]:

```
plt.title('New Corona cases in United States', fontsize=22)
plt.xlabel('Month')
df_United_States = df[df['location'] == 'United States']
#print(df_United_States).head()
plt.plot(df_United_States['new_cases'])
plt.savefig('UScases.png', dpi = 300)
```



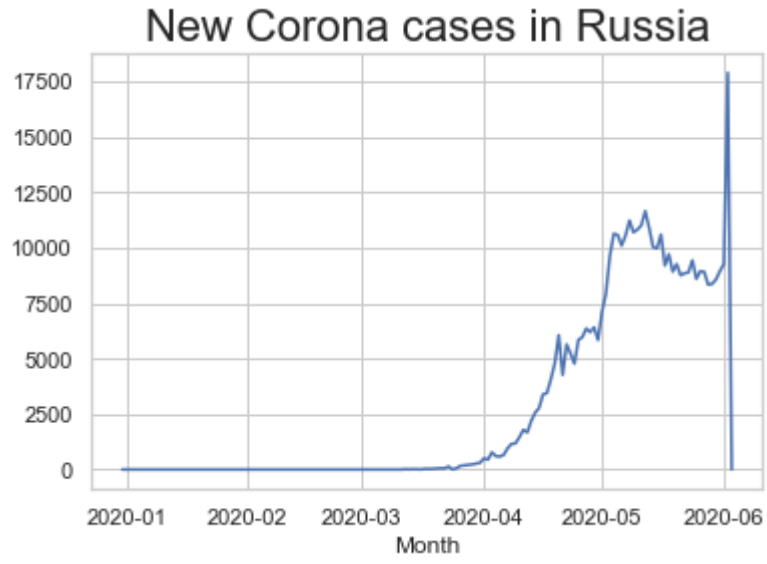
In [45]:

```
plt.title('New Corona cases in India', fontsize=22)
plt.xlabel('Month')
df_India = df[df['location'] == 'India']
#print(df_United_States).head()
plt.plot(df_India['new_cases'])
plt.savefig('India cases.png', dpi = 300)
```



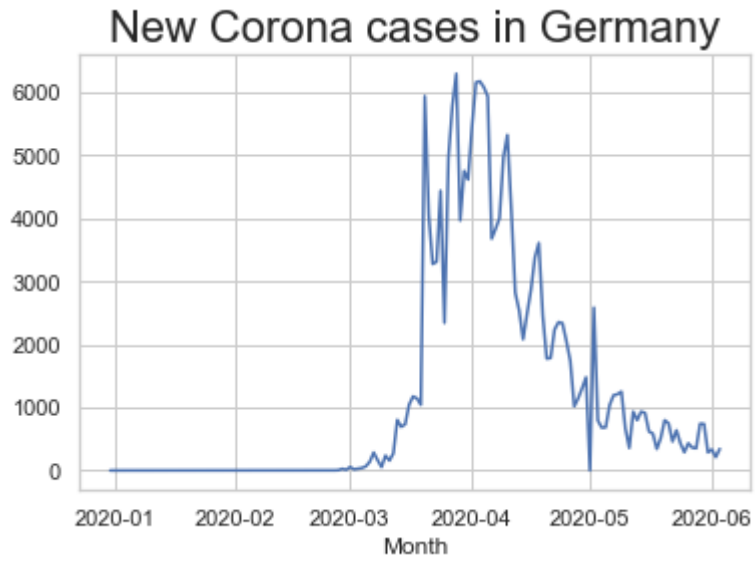
In [46]:

```
plt.title('New Corona cases in Russia', fontsize=22)
plt.xlabel('Month')
df_Russia = df[df['location'] == 'Russia']
#print(df_United_States).head()
plt.plot(df_Russia['new_cases'])
plt.savefig('Russia cases.png', dpi = 300)
```



In [47]:

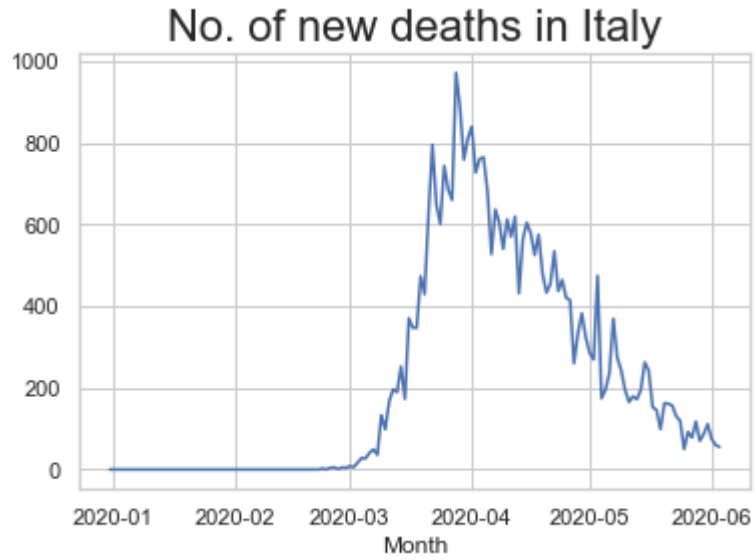
```
plt.title('New Corona cases in Germany', fontsize=22)
plt.xlabel('Month')
df_Germany = df[df['location'] == 'Germany']
#print(df_United_States).head()
plt.plot(df_Germany['new_cases'])
plt.savefig('Germany cases.png', dpi = 300)
```



Deaths by Countries

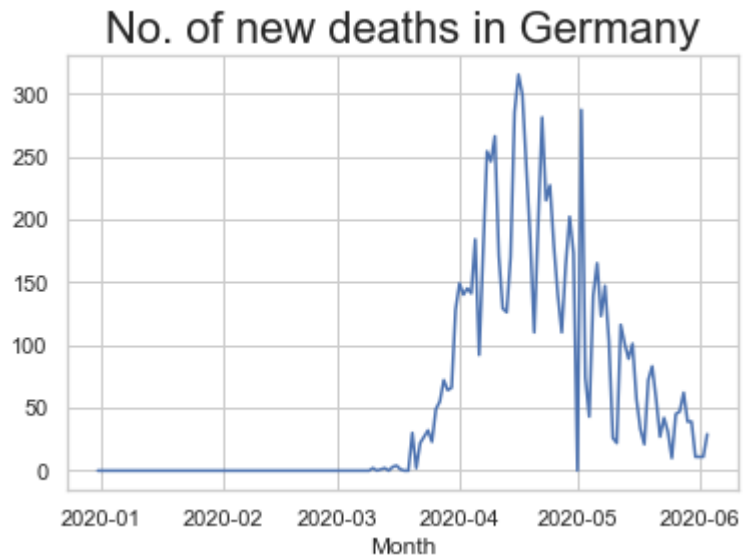
In [48]:

```
plt.title('No. of new deaths in Italy', fontsize=22)
plt.xlabel('Month')
df_Italy = df[df['location'] == 'Italy']
#print(df_Italy)
plt.plot(df_Italy['new_deaths'])
plt.savefig('Italy deaths.png', dpi = 300)
```



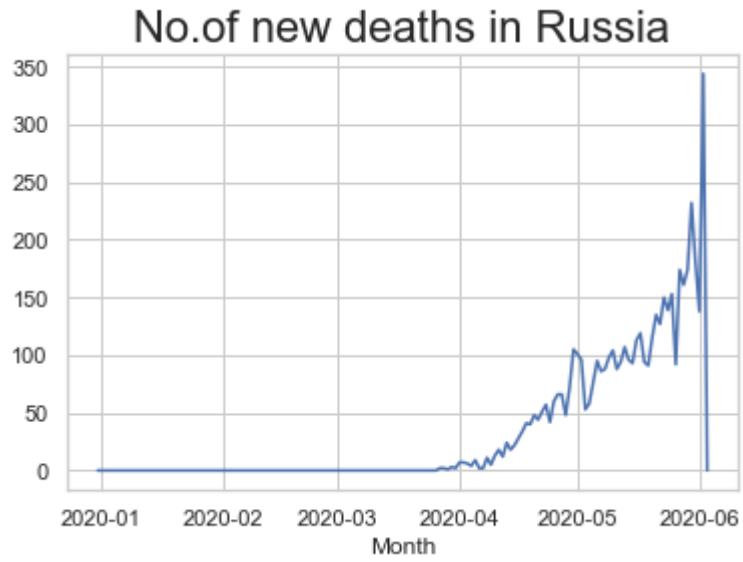
In [49]:

```
plt.title('No. of new deaths in Germany', fontsize=22)
plt.xlabel('Month')
df_Germany = df[df['location'] == 'Germany']
#print(df_United_States).head()
plt.plot(df_Germany['new_deaths'])
plt.savefig('Germany deaths.png', dpi = 300)
```



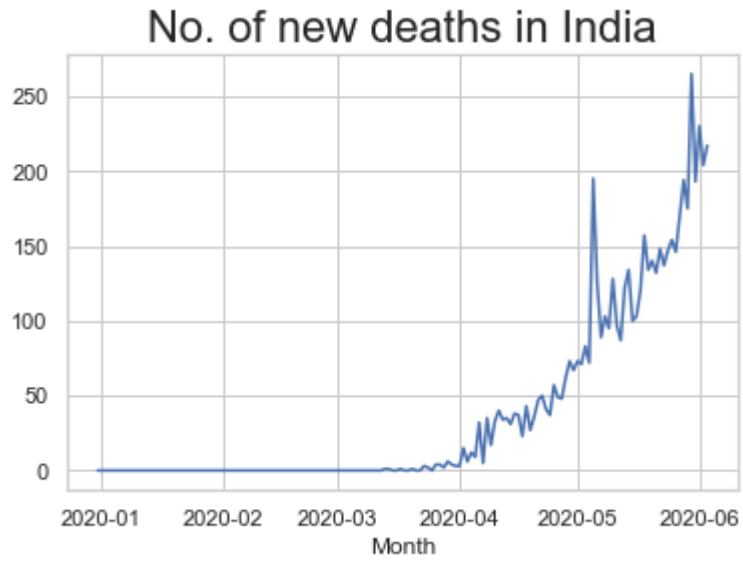
In [50]:

```
plt.title('No.of new deaths in Russia', fontsize=22)
plt.xlabel('Month')
df_Russia = df[df['location'] == 'Russia']
plt.plot(df_Russia['new_deaths'])
plt.savefig('Russia deaths.png', dpi = 300)
```



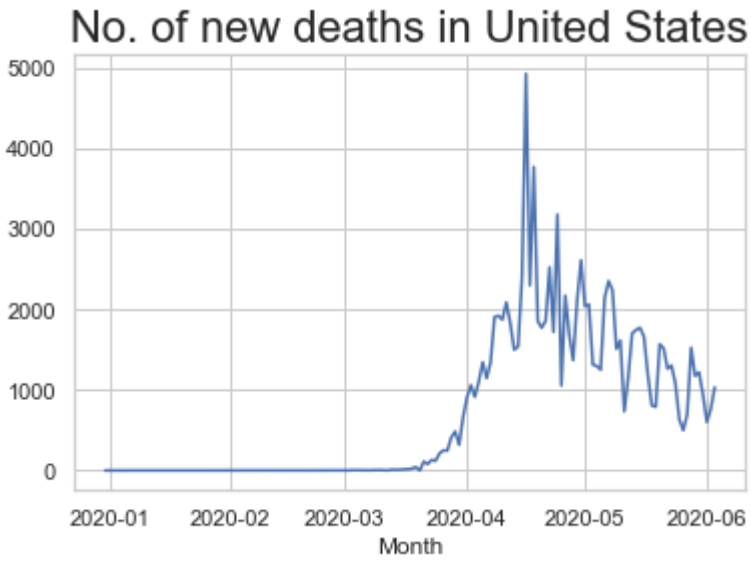
In [51]:

```
plt.title('No. of new deaths in India', fontsize=22)
plt.xlabel('Month')
df_India = df[df['location'] == 'India']
#print(df_United_States).head()
plt.plot(df_India['new_deaths'])
plt.savefig('Indiadeaths.png', dpi = 300)
```



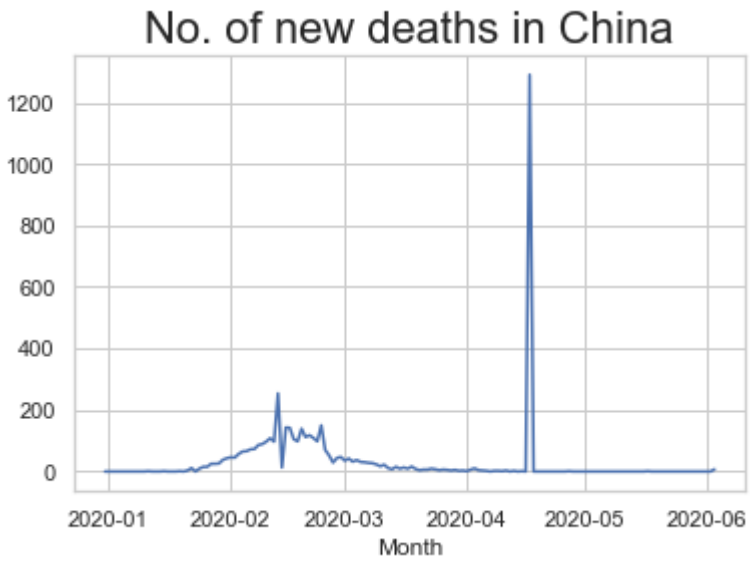
In [52]:

```
plt.title('No. of new deaths in United States', fontsize=22)
plt.xlabel('Month')
df_United_States = df[df['location'] == 'United States']
#print(df_United_States).head()
plt.plot(df_United_States['new_deaths'])
plt.savefig('USdeaths.png', dpi = 300)
```



In [53]:

```
plt.title('No. of new deaths in China', fontsize=22)
plt.xlabel('Month')
df_China = df[df['location'] == 'China']

plt.plot(df_China['new_deaths'])
plt.savefig('China deaths.png', dpi = 300)
```



In [54]:

```
#parsing dates and adding date as index
df = pd.read_csv("./Covid - 19 - Final.csv", parse_dates = ['date'], index_col = 'date')
```

In [55]:

```
df.head()
```

Out[55]:

date	location	Month	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_milli
2020-03-13	Aruba	Mar	2	2	0	0	18.7
2020-03-20	Aruba	Mar	4	2	0	0	37.4
2020-03-24	Aruba	Mar	12	8	0	0	112.3
2020-03-25	Aruba	Mar	17	5	0	0	159.2
2020-03-26	Aruba	Mar	19	2	0	0	177.9

In [56]:

```
Top_20.set_index('date', inplace = True) #Setting date as index
```

Pearson Correlation

In [57]:

```
#Reading few columns which needs to be correlated
data = pd.DataFrame(df, columns = ['total_cases', 'new_cases', 'total_deaths', 'new_deaths'
data.head()
```

Out[57]:

date	total_cases	new_cases	total_deaths	new_deaths
2020-03-13	2	2	0	0
2020-03-20	4	2	0	0
2020-03-24	12	8	0	0
2020-03-25	17	5	0	0
2020-03-26	19	2	0	0

In [58]:

```
#Pearson Correlation
pearsoncorr = data.corr(method = 'pearson')
pearsoncorr
```

Out[58]:

	total_cases	new_cases	total_deaths	new_deaths
total_cases	1.000000	0.785899	0.924879	0.700264
new_cases	0.785899	1.000000	0.655935	0.836401
total_deaths	0.924879	0.655935	1.000000	0.677768
new_deaths	0.700264	0.836401	0.677768	1.000000

In [59]:

```
#Plotting Pearson Correlation
plt.title('Pearson Correlation', fontsize=20)
sns.heatmap(pearsoncorr, xticklabels = pearsoncorr.columns, yticklabels = pearsoncorr.colum
plt.yticks(rotation = 60, fontsize=10)
plt.xticks(fontsize = 10)
#left,right = ax.get_xlim()
#ax.set_xlim(left +1.0, right -1.0)
plt.savefig('Pearson.png', dpi = 300)
```