canada_covid_datapreprocessing

October 14, 2020

1 Context

1.0.1 Novel Coronavirus 2019 (nCoV-2019) is a virus which affects respiratory system and was first discovered in wuhan, China. Some early reports suggested that virus may have been transmitted from animal to person. As we know whole world has been shutdown because of the widespread cases. At this time it's unclear how easily or sustainably this virus is spreading between people.

2 Current Cases (WorldWide)

2.0.1 To know how bad the world has been affected lets get some information on current situation.

Lets import all the dependencies for scrapping the website

[36]:
```
import bs4
from urllib.request import Request, urlopen
from urllib.request import urlopen as uReq
from bs4 import BeautifulSoup as soup
import pandas as pd
import plotly.graph_objects as go
```

[37]:
```
# grabbing the url

url = "https://ncov2019.live/"
req = Request(url, headers={"User-Agent" : "Mozilla/5.0"})

webpage = urlopen(req).read()

#parsing it as lxml
pagesoup = soup(webpage,"lxml")
```

Website Information

1. Website Name
2. Link to Website

[38]:
```
from IPython.display import display, Markdown
```

[39]:
```
#finding the relevant tags to scrap the data from website

website_name = pagesoup.find('a',class_ = "navbar-brand")
link = "https://ncov2019.live/"
Markdown('<strong>{}</strong>{}'.format(website_name.text,link))
```

[39]:

nCoV2019.live

https://ncov2019.live/

[40]:
```
#some quick facts from the website

quickfacts = pagesoup.find('div', class_ = "container--wrap bg-navy-4")
Markdown('<strong align="center">{}</strong>'.format(quickfacts))
```

[40]:

Quick Facts

updated: A few minutes ago

38,727,284

Total Confirmed

70,065

Total Critical

1,096,292

Total Deceased

8,827,582

Total Active

28,624,301

Total Recovered

177

Total Vaccines In Development

2.1 World COVID-19 Stats

We will scrap worldwide covid cases.

1. We'll use pandas read.html which lets us read the webpage table without much of complexity.
2. Convert the table into dataframe for further processing.
3. In the header of the list generated you see a number ``1'', which was used in the original website as a filter for arranging data in ascending or descending order.

[41]:
```
import pandas as pd
import requests
```

[42]:
```
# grabbing latest worldwide data

url = "https://ncov2019.live/data/world"

r = requests.get(url)
df_list = pd.read_html(r.text)            #this parse all html tables from a webpage to alist
world_df = df_list[2]
world_df
```

[42]:

	Name	Confirmed	Per Million	Changes Today	Percentage Day Change	Critical	Deceased	Per Million.1	Changes Today.1	Percentage Death Change	Tests	Active	Recovered	Per Million.2
0	TOTAL	38727284	4978	378470	0.99%	70065	1096292	141	6052	0.56%	718000366	8827582	28624301	3679
1	Afghanistan	39994	1021	66	0.17%	93	1481	38	1	0.07%	115720	5159	33354	851
2	Albania	15955	5546	203	1.29%	16	434	151	5	1.17%	97605	5759	9762	3393
3	Algeria	53584	1216	185	0.35%	38	1827	41	9	0.5%	Unknown	Unknown	37603	853
4	Andorra	3190	0	195	6.51%	25	59	0	2	3.51%	137457	1120	2011	0
..	…	…	…	…	…	…	…	…	…	…	…	…	…	…
211	Saint Pierre and Miquelon	16	0	0	0%	Unknown	Unknown	Unknown	0	0%	2222	Unknown	12	0
212	Montserrat	13	0	0	0%	Unknown	1	0	0					
213	Falkland Islands	13	0	0	0%	Unknown	Unknown	Unknown	0					
214	Anguilla	3	0	0	0%	Unknown	Unknown	Unknown	0					
215	China	85611	59	20	0.02%	4	4634	3	0					

```
212                    0%        483        0        12             0
213                    0%       2682  Unknown        13             0
214                    0%       1329  Unknown         3             0
215                    0%  160000000      241     80736            56

     Population
0    7780416607
1      39172730
2       2876889
3      44073388
4         77301
..          ...
211        5786
212        4993
213        3508
214       15041
215  1439323776

[216 rows x 15 columns]
```

Sorting the data on number of confirmed cases

[43]:
```
# We will now sort the countries based on total confirmed cases column

world_df = world_df.sort_values("Confirmed" , ascending = False)



#Lets get top 10 affected countries

world_df.head(10)
```

[43]:
```
                Name  Confirmed Per Million  Changes Today  \
0              TOTAL   38727284        4978         378470
170    United States    8148083       24575          57733
171            India    7305070        5279          67988
172           Brazil    5141498       24140          26675
173           Russia    1340409        9184          14231
7          Argentina     931967       20567          14932
175         Colombia     930159       18225           6061
174            Spain     908056       19420          11970
176             Peru     856951       25888           2977
177           Mexico     825340        6382           4295

    Percentage Day Change Critical Deceased Per Million.1  Changes Today.1  \
0                   0.99%    70065  1096292           141             6052
170                 0.71%    15143   221818           669              945
```

```
171                  0.94%     8944   111311            80              694
172                  0.52%     8318   151779           713              716
173                  1.07%     2300    23205           159              239
7                    1.63%     4316    24921           550              349
175                  0.66%     2220    28306           555              165
174                  1.34%     1652    33413           715              209
176                  0.35%     1163    33512          1012               93
177                  0.52%     2379    84420           653              475

    Percentage Death Change      Tests   Active Recovered Per Million.2  \
0                     0.56%  718000366  8827582  28624301          3679
170                   0.43%  121368954  2656624   5269641         15894
171                   0.63%   90090122   813303   6380456          4611
172                   0.47%   17900000   420906   4568813         21451
173                   1.04%   51800000   277499   1039705          7124
7                     1.42%    2283577   155900    751146         16577
175                   0.59%    4270936    85186    816667         16001
174                   0.63%   14590713   724267    150376          3216
176                   0.28%    4135223    63842    759597         22947
177                   0.57%    2109456   139349    601571          4652

     Population
0    7780416607
170   331558077
171  1383863737
172   212990988
173   145952510
7      45313862
175    51036966
174    46760002
176    33101676
177   129317680
```

Lets see many coulmns are missing values.

[44]:
```
world_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 216 entries, 0 to 141
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Name                    216 non-null    object
 1   Confirmed               216 non-null    int64
 2   Per Million             216 non-null    object
 3   Changes Today           216 non-null    int64
 4   Percentage Day Change   216 non-null    object
 5   Critical                216 non-null    object
```

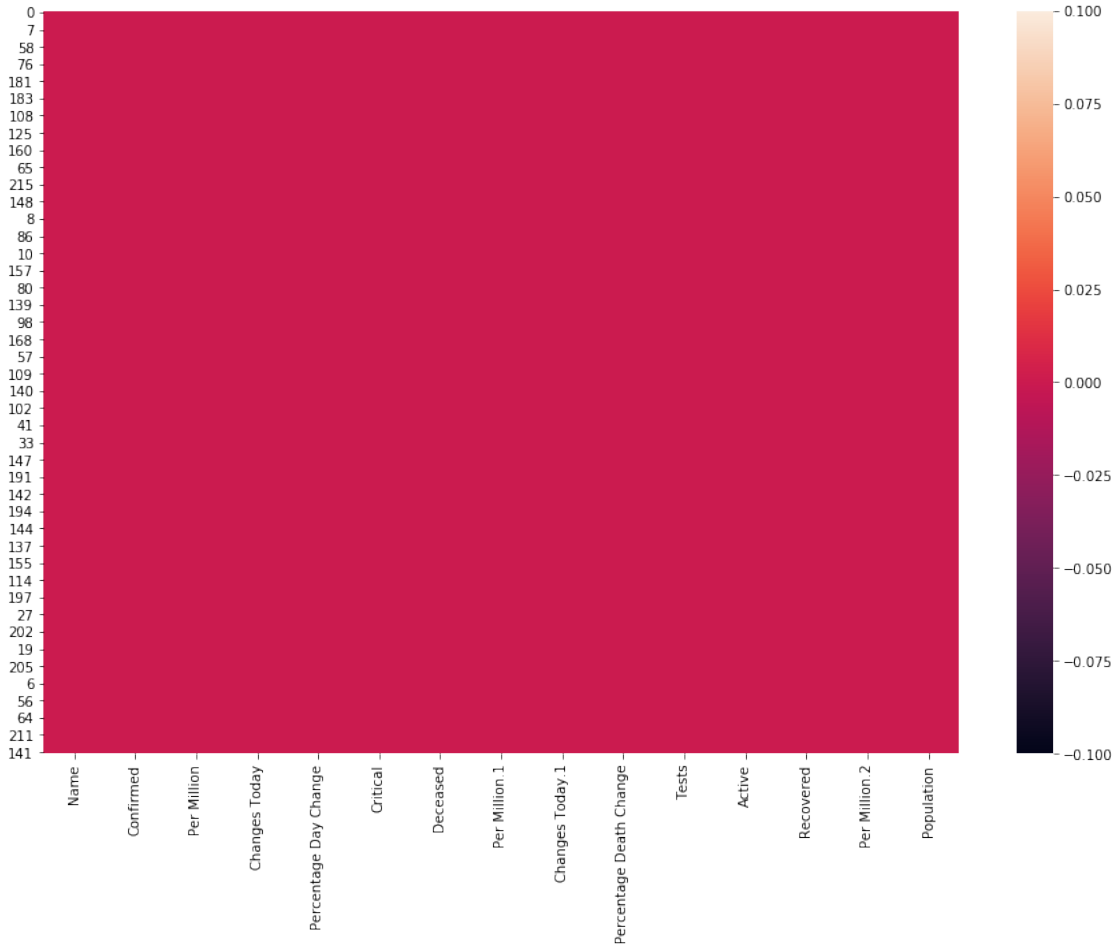
```
 6   Deceased                  216 non-null    object
 7   Per Million.1             216 non-null    object
 8   Changes Today.1           216 non-null    int64
 9   Percentage Death Change   216 non-null    object
 10  Tests                     216 non-null    object
 11  Active                    216 non-null    object
 12  Recovered                 216 non-null    object
 13  Per Million.2             216 non-null    object
 14  Population                216 non-null    object
dtypes: int64(3), object(12)
memory usage: 27.0+ KB
```

Lets import seaborn as well as matplotlib

[45]:
```
#We can also visualize the same using seaborn

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[46]:
```
plt.figure(figsize=(15,10))
sns.heatmap(world_df.isnull())
```

[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25563557408>
```



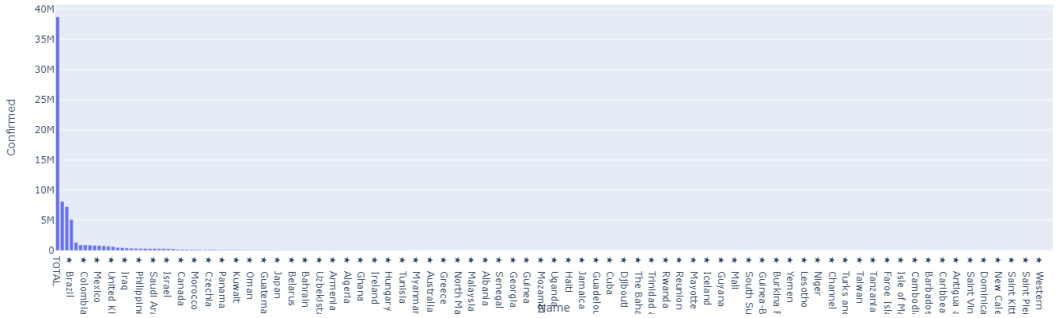
We'll use plotly express for visualization.

1. It generates graphs which are interactive and user friendly.
2. We can use zoom in and zoom out feature for proper understanding to a specific part of graph.

[47]:
```
import plotly.express as px
import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

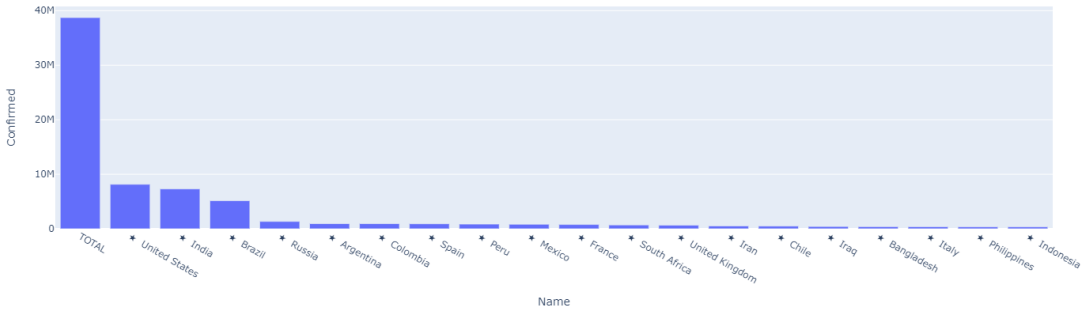
Plot number of confirmed cases.

[48]:
```
# plotting world_df based on confirmed cases by country names.

world_fig = px.bar(world_df, x = 'Name' , y = 'Confirmed')
world_fig.show()
```



- We can zoom in the graph, thats the beauty of plotly.

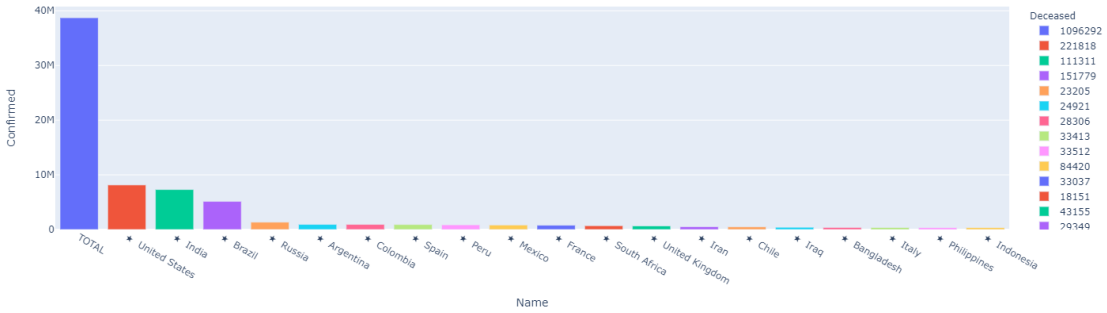
[49]:
```
# Lets plot top 20 countries based on confirmed cases.

world_fig = px.bar(world_df.head(20), x = "Name" , y = 'Confirmed')
world_fig.show()
```



- Now we can see United states holds number 1 position. (cough cough ``we don't wear masks'' - americans)
- Brazil and India comes at the second and third position surpassing Russia respectively.

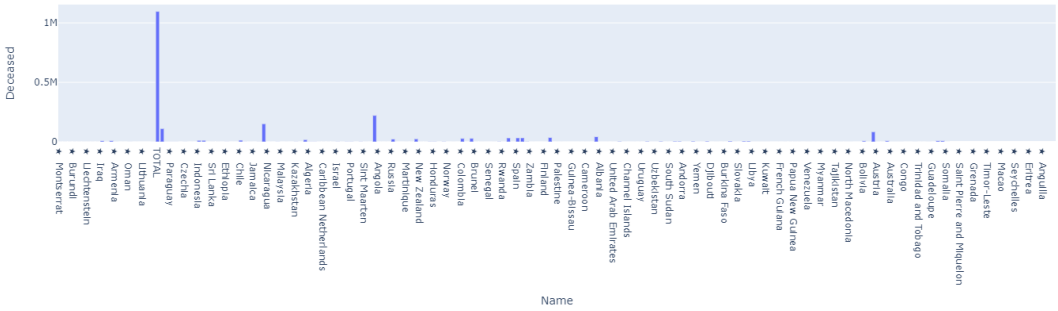
Now we'll try to explore the world_df in more details.(based on number of Deceased People)

[50]:
```
# Lets see how many people have died with respect to countries. (For top 20 countries)

world_fig = px.bar(world_df.head(20), x = 'Name', y = 'Confirmed', color = "Deceased")
world_fig.show()
```



- Here the color of each bar corrosponds to how many people have died.
- We cannot make out which country has most number of deceased people in a descending order.

[51]:
```
# lets visualize world_df based on death poll for top 20 countries.

world_fig = px.bar(world_df.sort_values('Deceased'), x = 'Name', y = 'Deceased')
world_fig.show()
```



- As you see here also we cannot make countres based on number of deceased people.
- There is some some problem with world_df[``Deceased''] column, let's fix that.

[52]:
```
# lets grab the world_df based on deceased column.
world_df.sort_values('Deceased',ascending = False)
```

[52]:

	Name	Confirmed	Per Million	Changes Today	\
141	Solomon Islands	2	0	0	
129	Saint Lucia	29	0	0	
200	Gibraltar	516	0	17	
202	Faroe Islands	478	0	1	

```
52              Eritrea        414         116              0
..                  …        …         …            …
199             Curacao        645           0             26
27              Burundi        529          44              0
212         Montserrat         13           0              0
166     Western Sahara         10           0              0
204     Cayman Islands        225           0              4

    Percentage Day Change Critical Deceased Per Million.1  Changes Today.1  \
141                    0%  Unknown  Unknown       Unknown                 0
129                    0%  Unknown  Unknown       Unknown                 0
200                 3.41%        2  Unknown       Unknown                 0
202                 0.21%  Unknown  Unknown       Unknown                 0
52                     0%  Unknown  Unknown       Unknown                 0
..                    …      …      …            …               …
199                  4.2%        2        1             0                 0
27                     0%  Unknown        1             0                 0
212                    0%  Unknown        1             0                 0
166                    0%  Unknown        1             0                 0
204                 1.81%  Unknown        1             0                 0

    Percentage Death Change    Tests   Active Recovered Per Million.2  \
141                      0%       96  Unknown   Unknown       Unknown
129                      0%     8827  Unknown        27             0
200                      0%    55109  Unknown       435             0
202                      0%   141775  Unknown       467             0
52                       0%  Unknown  Unknown       372           104
..                      …       …       …        …           …
199                      0%     9541      277       367             0
27                       0%    44526       31       497            41
212                      0%      483        0        12             0
166                      0%  Unknown  Unknown         8             0
204                      0%    42800       12       212             0

    Population
141     691564
129     183868
200      33688
202      48916
52     3560296
..         …
199     164286
27    11988298
212       4993
166     601419
204      65941
```

[216 rows x 15 columns]

- The column contains many unknown values.
- We'll replace all the unknown values with zero.
- Then we will arrange the column in descending order for visualization purpose.

[53]:
```
# lets replace unknown values to 0.

world_df['Deceased'].replace("Unknown", 0,inplace=True)
world_df['Deceased'] = pd.to_numeric(world_df['Deceased'])          #convert␣
 ↪column from type object to int64
world_df['Deceased']
```

[53]:
```
0      1096292
170     221818
171     111311
172     151779
173      23205
        ...
212          1
213          0
166          1
214          0
141          0
Name: Deceased, Length: 216, dtype: int64
```

[54]:
```
# now again lets grab world_df based on deceased column.

world_df.sort_values('Deceased',ascending = False)
```

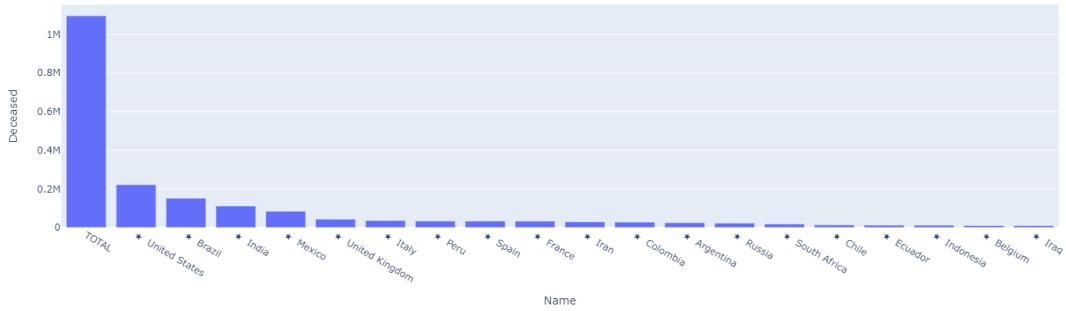
[54]:

	Name	Confirmed	Per Million	Changes Today	Percentage Day Change	Critical	Deceased	Per Million.1	Changes Today.1
0	TOTAL	38727284	4978	378470	0.99%	70065	1096292	141	6052
170	United States	8148083	24575	57733	0.71%	15143	221818	669	945
172	Brazil	5141498	24140	26675					
171	India	7305070	5279	67988					
177	Mexico	825340	6382	4295					
..	...	...	...	...					
52	Eritrea	414	116	0					
106	Mongolia	320	97	0					
136	Seychelles	148	0	0					
19	Bhutan	313	0	4					
141	Solomon Islands	2	0	0					

```
172                  0.52%     8318    151779           713              716
171                  0.94%     8944    111311            80              694
177                  0.52%     2379     84420           653              475
..                     …        …         …             …                …
52                     0%  Unknown         0       Unknown                0
106                    0%        1         0       Unknown                0
136                    0%  Unknown         0       Unknown                0
19                  1.29%  Unknown         0       Unknown                0
141                    0%  Unknown         0       Unknown                0

    Percentage Death Change      Tests   Active Recovered Per Million.2  \
0                     0.56%  718000366  8827582  28624301          3679
170                   0.43%  121368954  2656624   5269641         15894
172                   0.47%   17900000   420906   4568813         21451
171                   0.63%   90090122   813303   6380456          4611
177                   0.57%    2109456   139349    601571          4652
..                      …        …       …        …            …
52                      0%    Unknown  Unknown       372           104
106                     0%      79151  Unknown       311            94
136                     0%       5200  Unknown       144             0
19                      0%     153831  Unknown       293             0
141                     0%         96  Unknown   Unknown       Unknown

     Population
0    7780416607
170   331558077
172   212990988
171  1383863737
177   129317680
..          …
52      3560296
106     3293161
136       98521
19       774020
141      691564

[216 rows x 15 columns]
```

- Perfecto!. Now we can see that column has been cleared off all the ``Unknown''.

[55]:
```
# lets again try ro visualize world_df based on death poll for top 20 countries.

world_fig = px.bar(world_df.sort_values('Deceased', ascending=False).head(20),␣
↪x = 'Name' , y = 'Deceased')
world_fig.show()
```



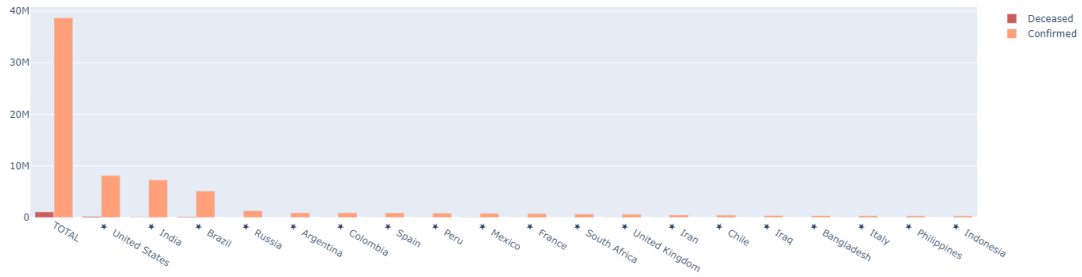
Click on the link for more information.

- United States tops the chart. If you want to know why United States leads in coronvirus cases, but not pandemic response
- Brazil also surpasses 100,000 deaths and becomes the one of the worst affected countries. `Death became normal': Brazil surpasses 100,000 deaths from COVID-19
- Mexico's death toll also reached 59.106k and many young people are dying of COVID-19 Why Are So Many Young People Dying Of Covid-19 In Mexico City?
- India has also reached 56k and there are many questions about India's rising COVID-19 infection Five key questions about India's rising Covid-19 infections

Lets visualize the death toll in relation to total confirmed case

[56]:
```
# lets visualize the death toll based on total confirmed case

import plotly.graph_objects as go


# for grouped barplot using Deceased numbers per country and total number of␣
 ↪cases per country.

fig = go.Figure(data = [
go.Bar(
    x = world_df['Name'],
    y = world_df["Deceased"].head(20),
    name = "Deceased",
    marker_color = "indianred"
),
go.Bar(
    x = world_df['Name'],
    y = world_df['Confirmed'].head(20),
    name = 'Confirmed',
    marker_color = "lightsalmon"
```

```
)
])

# Here we modify the tickangle of the xaxis, resulting in rotated labels.
fig.update_layout(barmode='group')
fig.show()
```

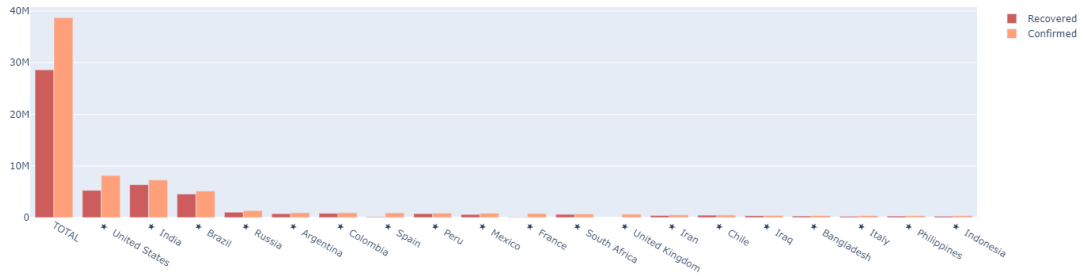


- Here we can see the Death toll is very low as compared to confirmed cases, which is because most of the people recover from COVID-19. Early estimates predicted that the overall COVID-19 recovery rate is between 97% and 99.75%.
- Mortality rate calculated = 3.4% (802.318k/23.09665M)

lets visualize the recovered cases based on total confirmed case

[57]:
```
# lets visualize the recovered case based in relation to total confirmed case

import plotly.graph_objects as go


# for grouped barplot using recovered cases per country and total number of␣
 ↪cases per country.

fig = go.Figure(data = [
go.Bar(
    x = world_df['Name'],
    y = world_df["Recovered"].head(20),
    name = "Recovered",
    marker_color = "indianred"
),
go.Bar(
    x = world_df['Name'],
    y = world_df['Confirmed'].head(20),
    name = 'Confirmed',
    marker_color = "lightsalmon"
)
```

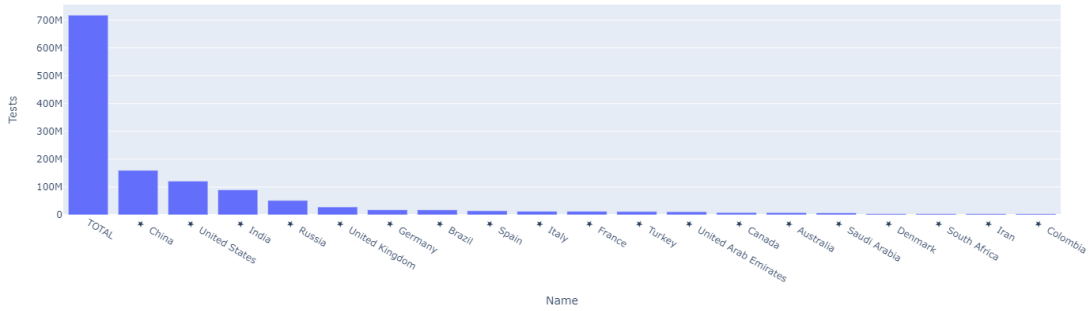
```
])

# Here we modify the tickangle of the xaxis, resulting in rotated labels.
fig.update_layout(barmode='group')
fig.show()
```



- Here we can see how many person recovered in relation to total cases registered.
- Recovery rate = 67% (15.4827M/23.09665M), this contradicts early predicted value of recovery rate which was 97%.
- Recovery rate and mortality rate are based on how well a country is implementing the testing of its people. Estimating mortality from COVID-19

Lets see who has implemented testing vastly.

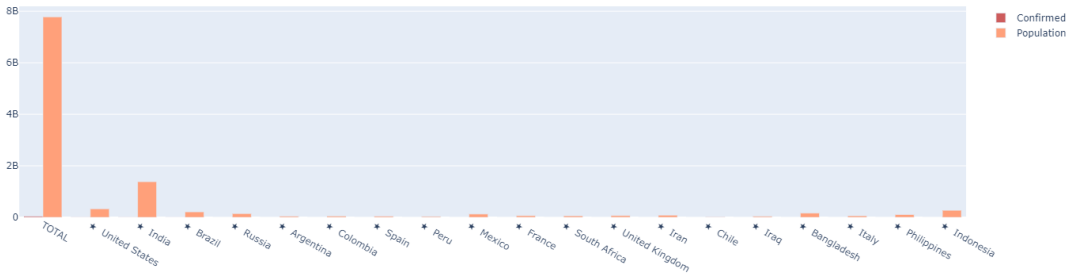
[58]:
```
# replace unknown values from the column

world_df['Tests'].replace("Unknown", 0, inplace=True)
world_df['Tests'] = pd.to_numeric(world_df['Tests'])          #convert column␣
↪from type object to int64



#Now lets plot the data

world_fig = px.bar(world_df.sort_values('Tests', ascending=False).head(20), x =␣
↪'Name' , y = 'Tests')
world_fig.show()
```



- China on first position that was unexpected. I was expecting United States.
- As you can see the countries who are vastly testing their people have a upper hand on curbing the spread of virus by implementing policies.

lets explore the Confirmed cases in relation to total population

[59]:
```
# lets visualize the confirmed case based in relation to total population

import plotly.graph_objects as go


# for grouped barplot using confirmed cases per country and population per␣
 ↪country.

fig = go.Figure(data = [
go.Bar(
    x = world_df['Name'],
    y = world_df["Confirmed"].head(20),
    name = "Confirmed",
    marker_color = "indianred"
),
go.Bar(
    x = world_df['Name'],
    y = world_df['Population'].head(20),
    name = 'Population',
    marker_color = "lightsalmon"
)
])

# Here we modify the tickangle of the xaxis, resulting in rotated labels.
fig.update_layout(barmode='group')
fig.show()
```



- This graph shows a small percentage of people are affected by the novel coronavirus. People who are at high risk for severe illness from COVID-19

Lets plot world data using Choropleth Map

[60]:
```
world_df.iloc[1:]['Name']
```

[60]:
```
170          United States
171                  India
172                 Brazil
173                 Russia
7                Argentina
               …
212              Montserrat
213       Falkland Islands
166         Western Sahara
214               Anguilla
141        Solomon Islands
Name: Name, Length: 215, dtype: object
```

[61]:
```
# For using choropleth first we have to make a dictionary

data = dict(
        type = 'choropleth',
        locations = world_df.iloc[1:]['Name'],
        z = world_df.iloc[1:]['Confirmed'],
        text = world_df.iloc[1:]['Deceased'],
        marker = dict(line = dict(color = 'rgb(255,255,255)',width = 2)),
        colorbar = {'title' : "Confirmed Cases"}
        )
```

[62]:
```
# Now create a layout for the graph

layout = dict(
    title = 'World COVID-19 Stats',
```

```
    geo = dict(
        showframe = False,
        projection = {'type':'mercator'}
    )
)
```

[63]:
```
# Finally we will pass both layout and data dictionary to generate the map.

choromap = go.Figure(data = [data],layout = layout)
iplot(choromap)
```

[64]:
```
#something worng with the country names. plotly uses standard ISO-3_codes. Lets␣
↪try to create a column for country codes

print("{} countries in the list.". format(world_df['Name'].nunique()))
```

```
216 countries in the list.
```

[65]:
```
world_df['Name']
```

[65]:
```
0                    TOTAL
170          United States
171                  India
172                 Brazil
173                 Russia
                 ...
212             Montserrat
213       Falkland Islands
166         Western Sahara
214               Anguilla
141        Solomon Islands
Name: Name, Length: 216, dtype: object
```

The country converter (coco) - a Python package for converting country names between different classifications schemes. For more info please click here.

[66]:
```
import country_converter as coco
```

[67]:
```
# Creating a list and appending all the names from world_df column.

Names = []
for i in range(1,215):
    Names.append(world_df.iloc[i]['Name'][3:])

# Insert Total at index 0. we left that because it doesn't contain any start in
 ↪it.

Names.insert(0,'TOTAL')
```

[68]:
```
Names
```

[68]:
```
['TOTAL',
 'United States',
 'India',
 'Brazil',
 'Russia',
 'Argentina',
 'Colombia',
 'Spain',
 'Peru',
 'Mexico',
 'France',
 'South Africa',
 'United Kingdom',
 'Iran',
 'Chile',
 'Iraq',
 'Bangladesh',
 'Italy',
 'Philippines',
 'Indonesia',
 'Germany',
 'Saudi Arabia',
 'Turkey',
 'Pakistan',
 'Israel',
 'Ukraine',
 'Netherlands',
 'Canada',
 'Belgium',
```

```
'Romania',
'Morocco',
'Ecuador',
'Poland',
'Czechia',
'Bolivia',
'Qatar',
'Panama',
'Dominican Republic',
'Nepal',
'Kuwait',
'United Arab Emirates',
'Kazakhstan',
'Oman',
'Egypt',
'Sweden',
'Guatemala',
'Costa Rica',
'Portugal',
'Japan',
'Ethiopia',
'China',
'Belarus',
'Honduras',
'Venezuela',
'Bahrain',
'Switzerland',
'Moldova',
'Uzbekistan',
'Nigeria',
'Austria',
'Armenia',
'Singapore',
'Lebanon',
'Algeria',
'Paraguay',
'Kyrgyzstan',
'Ghana',
'Libya',
'Palestine',
'Ireland',
'Azerbaijan',
'Kenya',
'Hungary',
'Afghanistan',
'Serbia',
'Tunisia',
```

```
'Denmark',
'Bosnia and Herzegovina',
'Myanmar',
'El Salvador',
'Jordan',
'Australia',
'Bulgaria',
'South Korea',
'Greece',
'Slovakia',
'Croatia',
'North Macedonia',
'Cameroon',
'Ivory Coast',
'Malaysia',
'Madagascar',
'Kosovo',
'Albania',
'Norway',
'Zambia',
'Senegal',
'Montenegro',
'Sudan',
'Georgia',
'Finland',
'Namibia',
'Guinea',
'Maldives',
'DR Congo',
'Mozambique',
'Tajikistan',
'French Guiana',
'Uganda',
'Luxembourg',
'Slovenia',
'Haiti',
'Gabon',
'Zimbabwe',
'Jamaica',
'Mauritania',
'Cape Verde',
'Guadeloupe',
'Angola',
'Lithuania',
'Cuba',
'Malawi',
'Eswatini',
```

'Djibouti',
'Nicaragua',
'Hong Kong',
'The Bahamas',
'Sri Lanka',
'Congo',
'Trinidad and Tobago',
'Suriname',
'Equatorial Guinea',
'Rwanda',
'Syria',
'Central African Republic',
'Reunion',
'Aruba',
'Malta',
'Mayotte',
'Estonia',
'Somalia',
'Iceland',
'Thailand',
'The Gambia',
'Guyana',
'French Polynesia',
'Botswana',
'Mali',
'Andorra',
'Latvia',
'South Sudan',
'Belize',
'Benin',
'Guinea-Bissau',
'Uruguay',
'Sierra Leone',
'Burkina Faso',
'Martinique',
'Cyprus',
'Yemen',
'Togo',
'New Zealand',
'Lesotho',
'Liberia',
'Chad',
'Niger',
'Vietnam',
'SÃ£o TomÃ© and PrÃ\xadncipe',
'Channel Islands',
'San Marino',

```
'Sint Maarten',
'Turks and Caicos Islands',
'Curacao',
'Papua New Guinea',
'Taiwan',
'Burundi',
'Gibraltar',
'Tanzania',
'Saint Martin',
'Comoros',
'Faroe Islands',
'Eritrea',
'Mauritius',
'Isle of Man',
'Mongolia',
'Bhutan',
'Cambodia',
'Monaco',
'Cayman Islands',
'Barbados',
'Bermuda',
'Liechtenstein',
'Caribbean Netherlands',
'Seychelles',
'Brunei',
'Antigua and Barbuda',
'British Virgin Islands',
'Saint Barthelemy',
'Saint Vincent and the Grenadines',
'Macao',
'Fiji',
'Dominica',
'Saint Lucia',
'Timor-Leste',
'New Caledonia',
'Grenada',
'Laos',
'Saint Kitts and Nevis',
'Vatican City',
'Greenland',
'Saint Pierre and Miquelon',
'Montserrat',
'Falkland Islands',
'Western Sahara',
'Anguilla']
```

[69]:
```
standard_names = coco.convert(names= Names, to='ISO3')
print(len(standard_names))
```

```
WARNING:root:TOTAL not found in regex
WARNING:root:SÃ£o TomÃ© and PrÃncipe not found in regex
WARNING:root:Channel Islands not found in regex
```

```
215
```

[70]:
```
map_data = world_df[world_df['Name']!='TOTAL']
print(map_data.nunique())
print(len(standard_names))

# Adding the ISO3 code in a new world_df['Code'] column.

map_data['code'] = standard_names
map_data['code'] = map_data['code'].shift(-1)
map_data.head()


map_data = map_data[:213]
map_data

# Removing countries of which ISO3 code is not available

choropleth_data = map_data[map_data['code'] != "not found"]
choropleth_data
```

```
Name                     215
Confirmed                210
Per Million              155
Changes Today            138
Percentage Day Change    127
Critical                  86
Deceased                 163
Per Million.1            101
Changes Today.1           49
Percentage Death Change   81
Tests                    197
Active                   165
Recovered                207
Per Million.2            153
Population               215
dtype: int64
215
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:7:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:8:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[70]:

	Name	Confirmed	Per Million	Changes Today	\
170	United States	8148083	24575	57733	
171	India	7305070	5279	67988	
172	Brazil	5141498	24140	26675	
173	Russia	1340409	9184	14231	
7	Argentina	931967	20567	14932	
..	…	…	…	…	
210	Greenland	16	0	0	
211	Saint Pierre and Miquelon	16	0	0	
212	Montserrat	13	0	0	
213	Falkland Islands	13	0	0	
166	Western Sahara	10	0	0	

	Percentage Day Change	Critical	Deceased	Per Million.1	Changes Today.1	\
170	0.71%	15143	221818	669	945	
171	0.94%	8944	111311	80	694	
172	0.52%	8318	151779	713	716	
173	1.07%	2300	23205	159	239	
7	1.63%	4316	24921	550	349	
..	…	…	…	…	…	
210	0%	Unknown	0	Unknown	0	
211	0%	Unknown	0	Unknown	0	
212	0%	Unknown	1	0	0	
213	0%	Unknown	0	Unknown	0	
166	0%	Unknown	1	0	0	

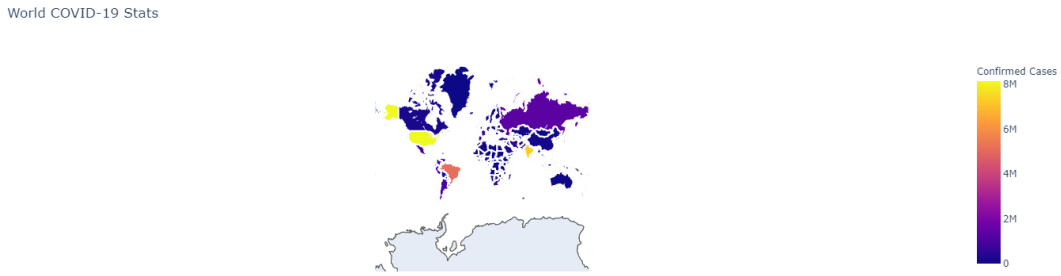
	Percentage Death Change	Tests	Active	Recovered	Per Million.2	\
170	0.43%	121368954	2656624	5269641	15894	
171	0.63%	90090122	813303	6380456	4611	

```
172                  0.47%   17900000   420906   4568813          21451
173                  1.04%   51800000   277499   1039705           7124
7                    1.42%    2283577   155900    751146          16577
..                     …        …       …        …              …
210                    0%        8879  Unknown        14              0
211                    0%        2222  Unknown        12              0
212                    0%         483        0        12              0
213                    0%        2682  Unknown        13              0
166                    0%           0  Unknown         8              0

     Population code
170   331558077  USA
171  1383863737  IND
172   212990988  BRA
173   145952510  RUS
7      45313862  ARG
..          …  …
210        56798  GRL
211         5786  SPM
212         4993  MSR
213         3508  FLK
166       601419  ESH

[211 rows x 16 columns]
```

[71]:
```
#lets again try to plot the data using choropleth dataframe.

# For using choropleth first we have to make a dictionary

data = dict(
        type = 'choropleth',
        locations = choropleth_data['code'],
        z = choropleth_data['Confirmed'],
        text = choropleth_data['Deceased'],
        marker = dict(line = dict(color = 'rgb(255,255,255)',width = 2)),
        colorbar = {'title' : "Confirmed Cases"}
        )
```

[72]:
```
# Now create a layout for the graph

layout = dict(
    title = 'World COVID-19 Stats',
    geo = dict(
        showframe = False,
        projection = {'type':'mercator'}
    )
)
```

[73]:
```
# Finally we will pass both layout and data dictionary to generate the map.

choromap = go.Figure(data = [data],layout = layout)
iplot(choromap)
```



2.2 Canada COVID-19 Stats

Lets get Latest Canada's information.

1. We'll use the pandas read.html which lets us read the webpage table without much of complexity.
2. We can also use the lsit to convert it to a dataframe.
3. In the header of the list generated you see a number ``1'', which was used in the original website as a filter for arranging data in ascending or descending order.

[74]:
```
#grabbing latest canada specific data

url = "https://ncov2019.live/data/canada"

r = requests.get(url)
df_list = pd.read_html(r.text) # this parses all the tables in webpages to a␣
↪list
canada_df = df_list[2]
canada_df
```

[74]:

	Name	Confirmed	Per Million	Changes Today	\
0	TOTAL	189476	Unknown	0	
1	Alberta	20956	Unknown	0	
2	British Columbia	10734	Unknown	0	
3	Manitoba	2779	Unknown	0	
4	New Brunswick	284	Unknown	0	
5	Newfoundland and Labrador	283	Unknown	0	
6	Northwest Territories	5	Unknown	0	
7	Nova Scotia	1092	Unknown	0	

```
8                        Ontario      63300     Unknown              0
9          Prince Edward Island         63     Unknown              0
10                         Quebec      87791     Unknown              0
11                   Saskatchewan       2174     Unknown              0
12                         Yukon         15     Unknown              0

   Percentage Day Change Critical Deceased Per Million.1  Changes Today.1  \
0                     0%      172     9706       Unknown                0
1                     0%  Unknown      286       Unknown                0
2                     0%  Unknown      250       Unknown                0
3                     0%  Unknown       35       Unknown                0
4                     0%  Unknown        2       Unknown                0
5                     0%  Unknown        4       Unknown                0
6                     0%  Unknown  Unknown       Unknown                0
7                     0%  Unknown       65       Unknown                0
8                     0%  Unknown     3069       Unknown                0
9                     0%  Unknown  Unknown       Unknown                0
10                    0%  Unknown     5970       Unknown                0
11                    0%  Unknown       25       Unknown                0
12                    0%  Unknown  Unknown       Unknown                0

   Percentage Death Change    Tests   Active  Recovered Per Million.2  \
0                       0%  8269676    19559     160210       Unknown
1                       0%  Unknown  Unknown      18055       Unknown
2                       0%  Unknown  Unknown       9008       Unknown
3                       0%  Unknown  Unknown       1496       Unknown
4                       0%  Unknown  Unknown        200       Unknown
5                       0%  Unknown  Unknown        271       Unknown
6                       0%  Unknown  Unknown          5       Unknown
7                       0%  Unknown  Unknown       1023       Unknown
8                       0%  Unknown  Unknown      54432       Unknown
9                       0%  Unknown  Unknown         60       Unknown
10                      0%  Unknown  Unknown      73734       Unknown
11                      0%  Unknown  Unknown       1911       Unknown
12                      0%  Unknown  Unknown         15       Unknown

   Population
0     Unknown
1     Unknown
2     Unknown
3     Unknown
4     Unknown
5     Unknown
6     Unknown
7     Unknown
8     Unknown
9     Unknown
```

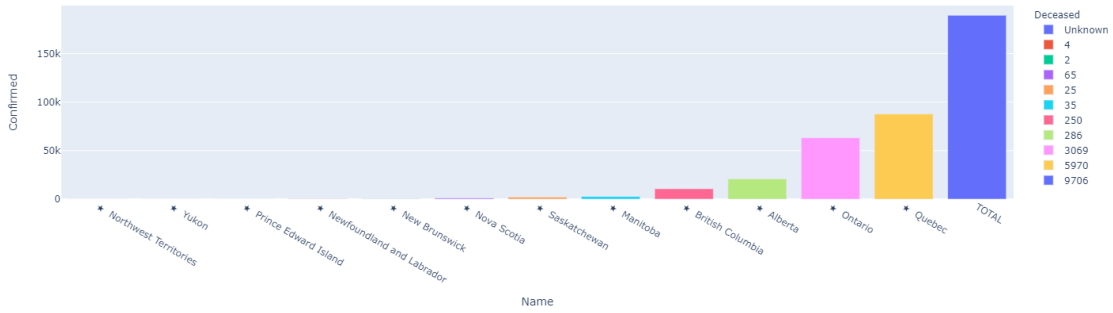
```
10    Unknown
11    Unknown
12    Unknown
```

2.3 Canada COVID-19 Stats

Lets visualize Canada's Data and see which province has been worst effected.

1. We'll use the same above canada_df for visualization purpose.
2. We are going to use this dataframe because it's the latest data and our script we'll update the data every time we run the cell based on the website mentioned above.
3. I'm going to use plotly for visualization purpose as it generates graphs which are interactive and user friendly.

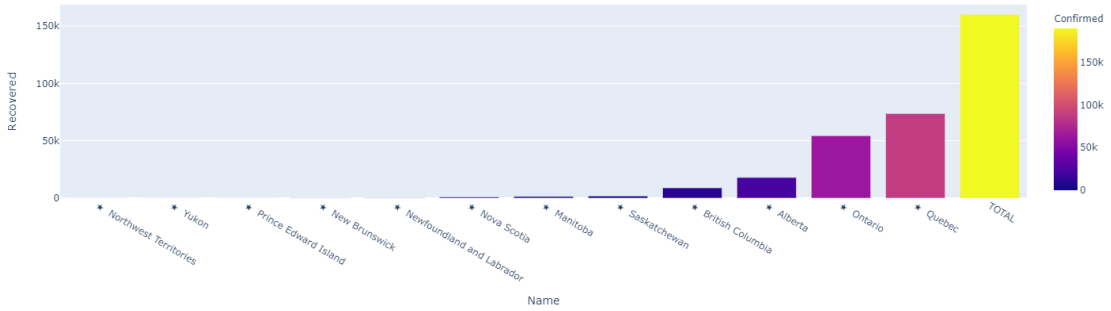
[75]:
```
canada_fig = px.bar(canada_df.
↪sort_values('Confirmed'),x='Name',y='Confirmed',color="Deceased")
canada_fig.show()
```



- Quebec has maximum number of confirmed cases and twice as many deceased people than ontario. Quebec leads Canada in Coronavirus deaths
- In this article I also found one more interesting thing that Alberta has done more testing per capita, and along with good policies the death polls remains below 500.
- There a some provinces where there were less to no cases, and no death has been reported, because quite a few people live there.

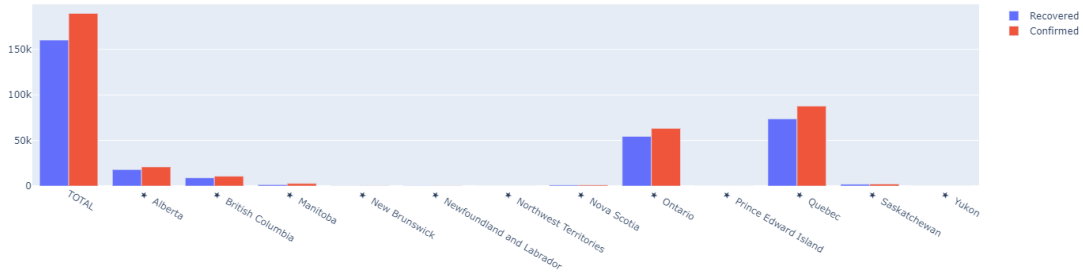
Lets see relation between total confirmed cases to recovered cases.

[76]:
```
canada_fig = px.bar(canada_df.sort_values('Recovered'), x = 'Name', y =␣
↪'Recovered',color='Confirmed')
canada_fig.show()
```



Lets calculate recovery rate in Canada and Alberta specifically

[77]:
```
fig = go.Figure(data = [
    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Recovered'],
    name = "Recovered"
    ),

    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Confirmed'],
    name = "Confirmed"
    )
])

fig.update_layout(barmode = "group")
fig.show()
```

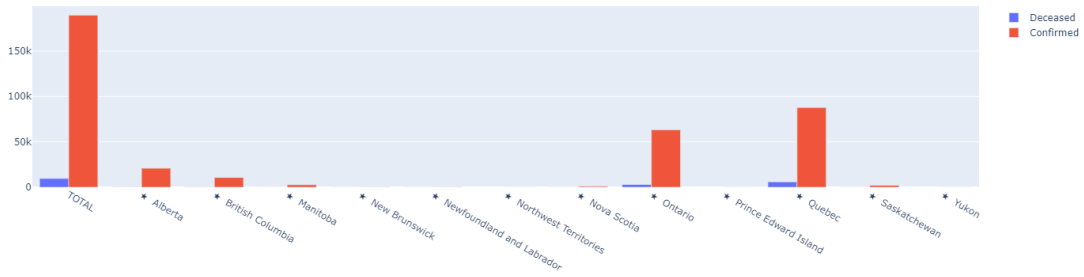


- Recovery rate canada wide is 88% which is 21% higher than the worldwide recovery rate. This also brings in another factor the geographical location a patient is in and how is the healthcare system there.

- Alberta's recovery rate is also 89% which is close to overall recovery rate.

lets calculate mortality rate.

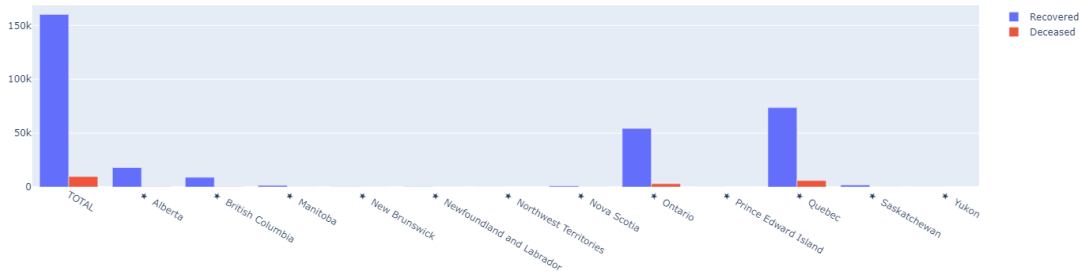
[78]:
```
fig = go.Figure(data = [
    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Deceased'],
    name = "Deceased"
    ),

    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Confirmed'],
    name = "Confirmed"
    )
])

fig.update_layout(barmode = "group")
fig.show()
```



- Mortality rate of overall canada is 7% (9118/126.804k)
- Mortality rate of Alberta is 1.8% which is quite astounding. Alberta is implementing policies very efficiently and because of that it has such a low mortality rate.
- Highest mortality rate is of Quebec 8.9%.
- Second highest mortality rate is of ontario 6.5%

[79]:
```
fig = go.Figure(data = [
    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Recovered'],
    name = "Recovered"
    ),
```

```
    go.Bar(
    x = canada_df['Name'],
    y = canada_df['Deceased'],
    name = "Deceased"
    )
])

fig.update_layout(barmode = "group")
fig.show()
```



2.4 Model for predicting the number of confirmed cases.

[80]:
```
# import confirmed cases data

confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/
 ↪COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_confirmed_global.csv')

#Getting all the dates
cols = confirmed_df.keys()
confirmed = confirmed_df.loc[:4,cols[4]:cols[-1]]

dates = confirmed.keys()
```

[81]:
```
worldcases = []

for i in ((dates)):

    confirmed_sum = confirmed[i].sum()

    worldcases.append(confirmed_sum)
```

[82]:
```
import numpy as np
import random
import math
import time
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime
```

[83]:
```
days_in_future = 10
future_forcast = np.array([i for i in range(len(dates)+days_in_future)]).
 ↪reshape(-1, 1)
adjusted_dates = future_forcast[:-10]
```

[84]:
```
start = '1/20/2020'
start_date = datetime.datetime.strptime(start, '%m/%d/%Y')
future_forcast_dates = []
for i in range(len(future_forcast)):
    future_forcast_dates.append((start_date + datetime.timedelta(days=i)).
 ↪strftime('%m/%d/%Y'))
```

[85]:
```
days_from_1_20 = np.array([i for i in range(len(dates))]).reshape(-1,1)
```

Train Test Split

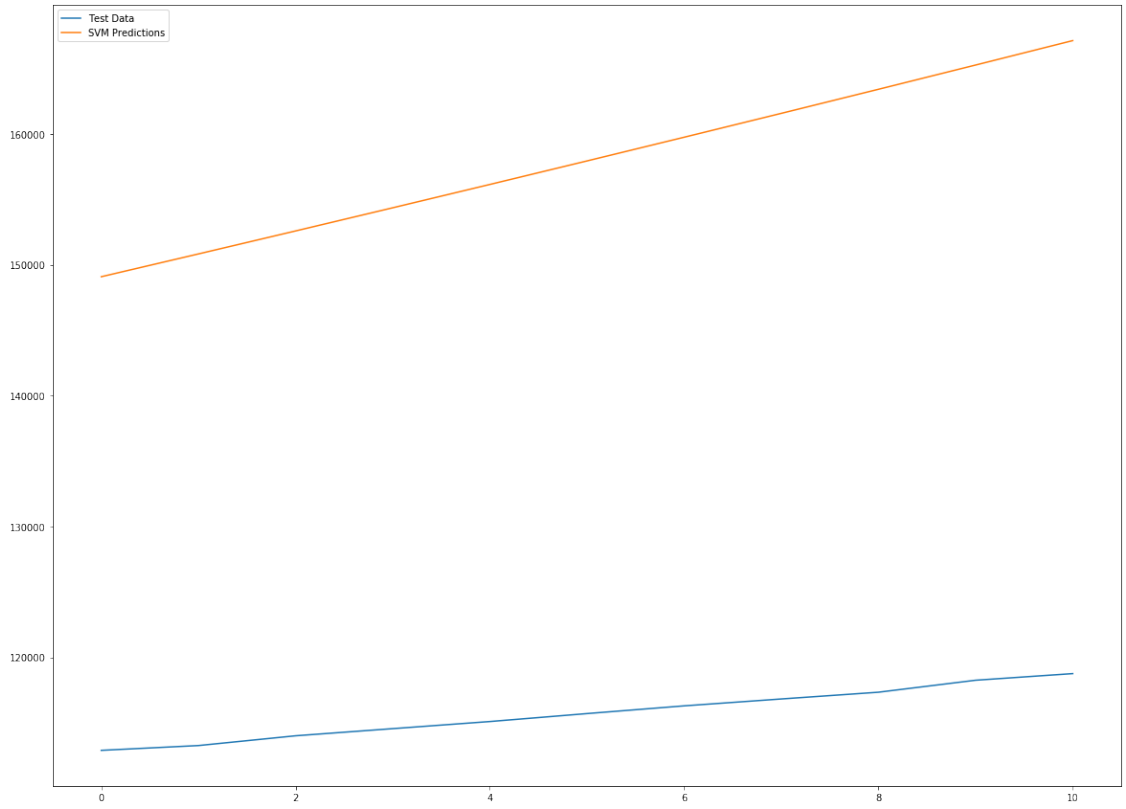
[86]:
```
X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed =␣
 ↪train_test_split(days_from_1_20[50:], worldcases[50:], test_size=0.05,␣
 ↪shuffle=False)
```

Support Vector Machine Model

[87]:
```
svm_confirmed = SVR(shrinking=True, kernel='poly',gamma=0.
 ↪01,epsilon=1,degree=3,C=0.1)
svm_confirmed.fit(X_train_confirmed,y_train_confirmed)
svm_pred = svm_confirmed.predict(future_forcast)
```

[88]:
```
svm_test_pred = svm_confirmed.predict(X_test_confirmed)
plt.figure(figsize=(20,15))
plt.plot(y_test_confirmed)
plt.plot(svm_test_pred)
plt.legend(['Test Data', 'SVM Predictions'])
print('MAE:', mean_absolute_error(svm_test_pred, y_test_confirmed))
print('MSE:',mean_squared_error(svm_test_pred, y_test_confirmed))
```

```
MAE: 42293.30619908933
MSE: 1803459408.5866432
```



Linear Regression model

[89]:
```
# transform our data for polynomial regression
poly = PolynomialFeatures(degree=5)
poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)
poly_X_test_confirmed = poly.fit_transform(X_test_confirmed)
poly_future_forcast = poly.fit_transform(future_forcast)
```

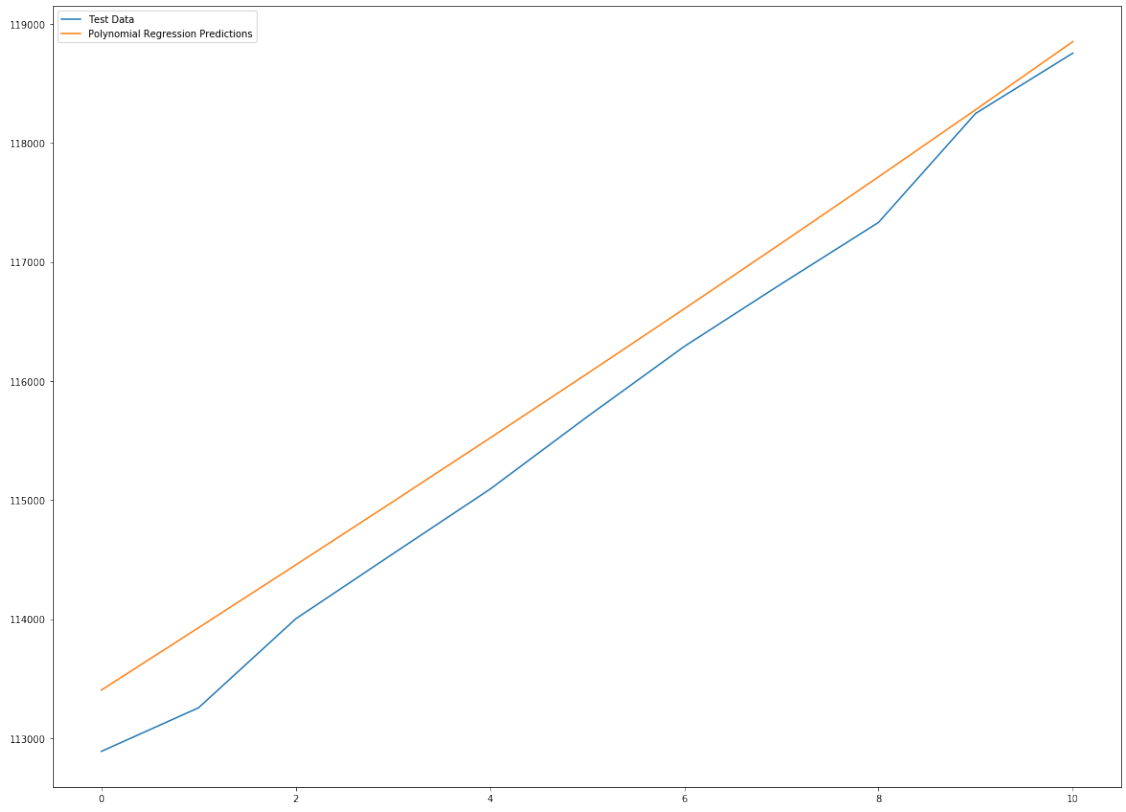
[90]:
```
# polynomial regression
linear_model = LinearRegression(normalize=True, fit_intercept=False)
linear_model.fit(poly_X_train_confirmed, y_train_confirmed)
test_linear_pred = linear_model.predict(poly_X_test_confirmed)
linear_pred = linear_model.predict(poly_future_forcast)
print('MAE:', mean_absolute_error(test_linear_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_linear_pred, y_test_confirmed))
```

```
MAE: 367.10495157956825
MSE: 163937.2934095679
```

[91]:
```
plt.figure(figsize=(20,15))
plt.plot(y_test_confirmed)
plt.plot(test_linear_pred)
```

```
plt.legend(['Test Data', 'Polynomial Regression Predictions'])
```

[91]: <matplotlib.legend.Legend at 0x2554537a988>



[92]:
```
confirmed_df.head()
```

[92]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	\
0	NaN	Afghanistan	33.93911	67.709953	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	

	1/24/20	1/25/20	1/26/20	1/27/20	…	10/4/20	10/5/20	10/6/20	\
0	0	0	0	0	…	39341	39422	39486	
1	0	0	0	0	…	14266	14410	14568	
2	0	0	0	0	…	52136	52270	52399	
3	0	0	0	0	…	2110	2370	2370	
4	0	0	0	0	…	5402	5530	5725	

	10/7/20	10/8/20	10/9/20	10/10/20	10/11/20	10/12/20	10/13/20

```
0    39548    39616    39693     39703     39799     39870     39928
1    14730    14899    15066     15231     15399     15570     15752
2    52520    52658    52804     52940     53072     53325     53399
3     2568     2568     2696      2696      2696      2995      2995
4     5725     5958     6031      6246      6366      6488      6680

[5 rows x 270 columns]
```

[93]:
```
# Transposing the row for time series analysis

confirmed_df = confirmed_df.T
confirmed_df = confirmed_df.rename(columns=confirmed_df.iloc[1])
confirmed_df
```

[93]:
```
                Afghanistan  Albania  Algeria  Andorra   Angola  \
Province/State          NaN      NaN      NaN      NaN      NaN
Country/Region  Afghanistan  Albania  Algeria  Andorra   Angola
Lat                 33.9391  41.1533  28.0339  42.5063 -11.2027
Long                  67.71  20.1683   1.6596   1.5218  17.8739
1/22/20                   0        0        0        0        0
…                         …      …        …        …      …
10/9/20               39693    15066    52804     2696     6031
10/10/20              39703    15231    52940     2696     6246
10/11/20              39799    15399    53072     2696     6366
10/12/20              39870    15570    53325     2995     6488
10/13/20              39928    15752    53399     2995     6680

                Antigua and Barbuda  Argentina  Armenia  \
Province/State                  NaN        NaN      NaN
Country/Region  Antigua and Barbuda  Argentina  Armenia
Lat                         17.0608   -38.4161  40.0691
Long                       -61.7964   -63.6167  45.0382
1/22/20                           0          0        0
…                               …        …      …
10/9/20                         111     871468    55087
10/10/20                        111     883882    55736
10/11/20                        111     894206    56451
10/12/20                        111     903730    56821
10/13/20                        111     917035    57566

                                  Australia        Australia  …  \
Province/State  Australian Capital Territory  New South Wales  …
Country/Region                     Australia        Australia  …
Lat                                -35.4735         -33.8688  …
Long                                149.012           151.209  …
1/22/20                                   0                 0  …
…                                       …                 …  …
```

```
10/9/20                                 113             4273  …
10/10/20                                113             4278  …
10/11/20                                113             4284  …
10/12/20                                113             4295  …
10/13/20                                113             4310  …

                United Kingdom  Uruguay  Uzbekistan  Venezuela  Vietnam  \
Province/State             NaN      NaN         NaN        NaN      NaN
Country/Region  United Kingdom  Uruguay  Uzbekistan  Venezuela  Vietnam
Lat                   55.3781 -32.5228     41.3775     6.4238  14.0583
Long                   -3.436 -55.7658     64.5853   -66.5897  108.277
1/22/20                     0        0           0          0        0
…                           …        …           …          …        …
10/9/20                575679     2251       60342      81696     1105
10/10/20               590844     2268       60776      82453     1107
10/11/20               603716     2294       61098      83137     1109
10/12/20               617688     2313       61319      83756     1110
10/13/20               634920     2337       61642      84391     1113

                West Bank and Gaza  Western Sahara    Yemen   Zambia  Zimbabwe
Province/State                 NaN             NaN      NaN      NaN       NaN
Country/Region  West Bank and Gaza  Western Sahara    Yemen   Zambia  Zimbabwe
Lat                        31.9522         24.2155  15.5527 -13.1339 -19.0154
Long                       35.2332        -12.8858  48.5164  27.8493  29.1549
1/22/20                          0               0        0        0        0
…                                …               …        …        …        …
10/9/20                      43664              10     2051    15339     7994
10/10/20                     43945              10     2051    15415     8010
10/11/20                     44299              10     2052    15458     8011
10/12/20                     44684              10     2052    15549     8021
10/13/20                     45200              10     2053    15587     8036

[270 rows x 267 columns]
```

[94]:
```
confirmed_df = confirmed_df[4:]
confirmed_df
```

[94]:
```
          Afghanistan Albania Algeria Andorra Angola Antigua and Barbuda  \
1/22/20             0       0       0       0      0                   0
1/23/20             0       0       0       0      0                   0
1/24/20             0       0       0       0      0                   0
1/25/20             0       0       0       0      0                   0
1/26/20             0       0       0       0      0                   0
…                   …       …       …       …      …                   …
10/9/20         39693   15066   52804    2696   6031                 111
10/10/20        39703   15231   52940    2696   6246                 111
10/11/20        39799   15399   53072    2696   6366                 111
```

```
10/12/20       39870   15570   53325    2995   6488                 111
10/13/20       39928   15752   53399    2995   6680                 111

          Argentina Armenia Australia Australia  … United Kingdom Uruguay  \
1/22/20           0       0         0         0  …              0       0
1/23/20           0       0         0         0  …              0       0
1/24/20           0       0         0         0  …              0       0
1/25/20           0       0         0         0  …              0       0
1/26/20           0       0         0         3  …              0       0
…               …     …       …       …  …            …     …
10/9/20      871468   55087       113      4273  …         575679    2251
10/10/20     883882   55736       113      4278  …         590844    2268
10/11/20     894206   56451       113      4284  …         603716    2294
10/12/20     903730   56821       113      4295  …         617688    2313
10/13/20     917035   57566       113      4310  …         634920    2337

          Uzbekistan Venezuela Vietnam West Bank and Gaza Western Sahara Yemen  \
1/22/20           0         0       0                  0              0     0
1/23/20           0         0       2                  0              0     0
1/24/20           0         0       2                  0              0     0
1/25/20           0         0       2                  0              0     0
1/26/20           0         0       2                  0              0     0
…               …       …     …                  …            …   …
10/9/20       60342     81696    1105              43664             10  2051
10/10/20      60776     82453    1107              43945             10  2051
10/11/20      61098     83137    1109              44299             10  2052
10/12/20      61319     83756    1110              44684             10  2052
10/13/20      61642     84391    1113              45200             10  2053

          Zambia Zimbabwe
1/22/20        0        0
1/23/20        0        0
1/24/20        0        0
1/25/20        0        0
1/26/20        0        0
…            …      …
10/9/20    15339     7994
10/10/20   15415     8010
10/11/20   15458     8011
10/12/20   15549     8021
10/13/20   15587     8036

[266 rows x 267 columns]
```

[95]:
```
confirmed_df['Total_cases'] = confirmed_df.sum(axis=1)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
```

```
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[96]:
```
# converting the index column to date

confirmed_df.reset_index(level=0,inplace=True)
confirmed_df
```

[96]:

	index	Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	\
0	1/22/20	0	0	0	0	0	0	
1	1/23/20	0	0	0	0	0	0	
2	1/24/20	0	0	0	0	0	0	
3	1/25/20	0	0	0	0	0	0	
4	1/26/20	0	0	0	0	0	0	
..	…	…	…	…	…	…	…	
261	10/9/20	39693	15066	52804	2696	6031	111	
262	10/10/20	39703	15231	52940	2696	6246	111	
263	10/11/20	39799	15399	53072	2696	6366	111	
264	10/12/20	39870	15570	53325	2995	6488	111	
265	10/13/20	39928	15752	53399	2995	6680	111	

	Argentina	Armenia	Australia	…	Uruguay	Uzbekistan	Venezuela	Vietnam	\
0	0	0	0	…	0	0	0	0	
1	0	0	0	…	0	0	0	2	
2	0	0	0	…	0	0	0	2	
3	0	0	0	…	0	0	0	2	
4	0	0	0	…	0	0	0	2	
..	…	…	…	…	…	…	…	…	
261	871468	55087	113	…	2251	60342	81696	1105	
262	883882	55736	113	…	2268	60776	82453	1107	
263	894206	56451	113	…	2294	61098	83137	1109	
264	903730	56821	113	…	2313	61319	83756	1110	
265	917035	57566	113	…	2337	61642	84391	1113	

	West Bank and Gaza	Western Sahara	Yemen	Zambia	Zimbabwe	Total_cases
0	0	0	0	0	0	555.0
1	0	0	0	0	0	654.0
2	0	0	0	0	0	941.0
3	0	0	0	0	0	1434.0
4	0	0	0	0	0	2118.0

```
..                 …              …   …     …      …         …
261              43664              10  2051  15339     7994  36876248.0
262              43945              10  2051  15415     8010  37207057.0
263              44299              10  2052  15458     8011  37475325.0
264              44684              10  2052  15549     8021  37801526.0
265              45200              10  2053  15587     8036  38129806.0

[266 rows x 269 columns]
```

[97]:
```
confirmed_df['dates'] = pd.to_datetime(confirmed_df['index'])
confirmed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 270 entries, index to dates
dtypes: datetime64[ns](1), float64(1), object(268)
memory usage: 561.2+ KB

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[98]:
```
time_series_analysis_df = confirmed_df[['Total_cases','dates']]
time_series_analysis_df
```

[98]:
```
     Total_cases      dates
0          555.0 2020-01-22
1          654.0 2020-01-23
2          941.0 2020-01-24
3         1434.0 2020-01-25
4         2118.0 2020-01-26
..           …        …
261   36876248.0 2020-10-09
262   37207057.0 2020-10-10
263   37475325.0 2020-10-11
264   37801526.0 2020-10-12
265   38129806.0 2020-10-13

[266 rows x 2 columns]
```

[99]:
```
# Now we will set the dates column as the index of the dataframe to allow us␣
↪really explore the our data.

time_series_analysis_df = time_series_analysis_df.set_index('dates')
time_series_analysis_df
```

[99]:

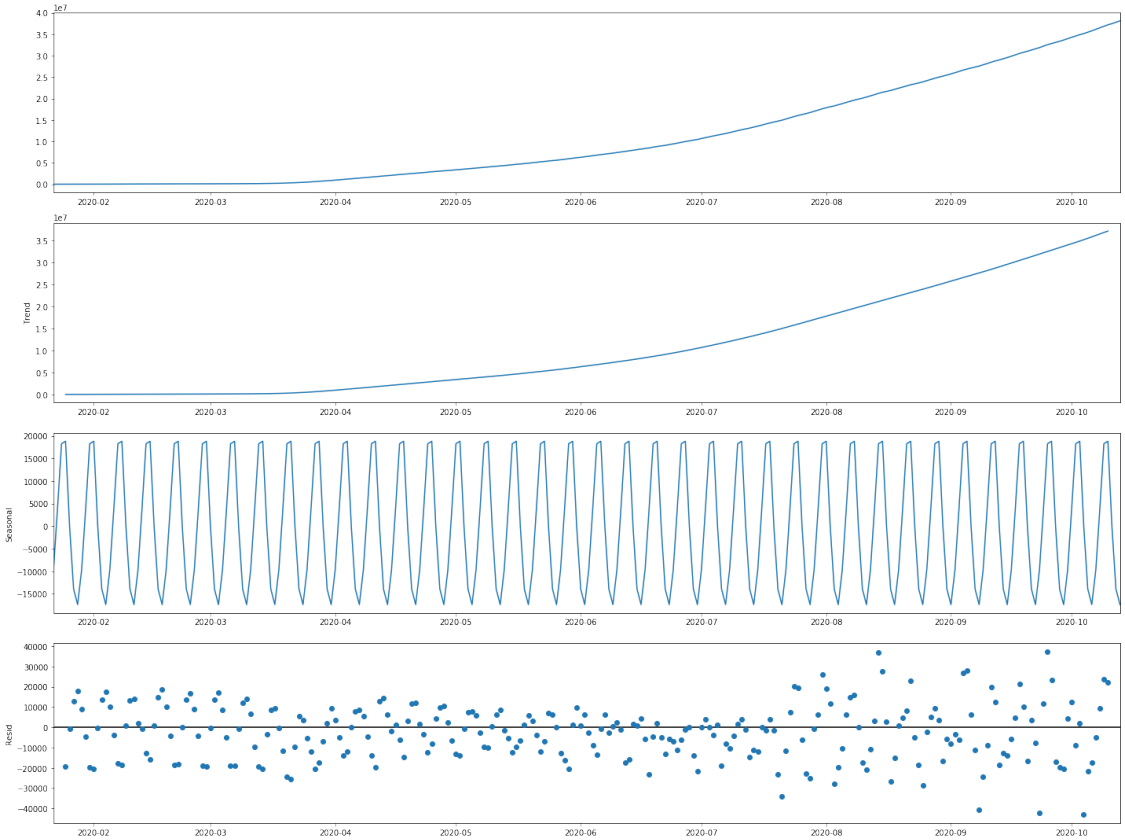
dates	Total_cases
2020-01-22	555.0
2020-01-23	654.0
2020-01-24	941.0
2020-01-25	1434.0
2020-01-26	2118.0
…	…
2020-10-09	36876248.0
2020-10-10	37207057.0
2020-10-11	37475325.0
2020-10-12	37801526.0
2020-10-13	38129806.0

[266 rows x 1 columns]

Additive model

1. This model is used when the time series level does not vary with the variations around the trend. Here, the time series components are simply added together using the formula:
   - y(t) = Level(t) + Trend(t) + Seasonality(t) + Noise(t)

[101]:
```
import matplotlib
import statsmodels.api as sm
decomposition = sm.tsa.
↪seasonal_decompose(time_series_analysis_df,model='additive')
fir = decomposition.plot()
matplotlib.rcParams['figure.figsize']=[20.0,15.0]
```



- The additive model predicted the total confirmed cases value with an accuracy of more than ~92%.
- This model can be used for forecasting values in future pandemic