In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
pd.set_option('display.max_rows', None)
from plotly.subplots import make_subplots
import seaborn as sns
import datetime

#Plotly Libraris
import plotly.express as px

# Minmax scaler
from sklearn.preprocessing import MinMaxScaler

#itertools
import itertools

#dataframe display settings
pd.set_option('display.max_columns', 5000000)
pd.set_option('display.max_rows', 50000000)

#to suppress un-necessary warnings
import warnings
warnings.filterwarnings('ignore')


#Package to flatten python lists
from pandas.core.common import flatten
```

In [2]:
```
covid_data = pd.read_csv('covid_19_data.csv')
covid_confirmed = pd.read_csv('time_series_covid_19_confirmed.csv')
covid_deaths = pd.read_csv('time_series_covid_19_deaths.csv')
covid_recovered = pd.read_csv('time_series_covid_19_recovered.csv')
```

In [3]:
```
covid_confirmed.describe()
```

Out[3]:

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20
count	274.000000	274.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000
mean	20.447559	22.328281	2.018116	2.373188	3.409420	5.192029	7.673913	10.605072	20.210145	22.344203	29.836957	35.967391	43.615942	60.822464	72.054348	86.586957
std	25.189838	74.369096	26.781738	26.879567	33.464159	46.575328	65.089830	87.699030	215.201418	216.521511	298.377294	353.754834	435.313401	676.494833	817.788348	1007.884613
min	-51.796300	-178.116500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.933349	-22.036550	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	21.607878	20.921188	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	40.950592	83.380449	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	71.706900	178.065000	444.000000	444.000000	549.000000	761.000000	1058.000000	1423.000000	3554.000000	3554.000000	4903.000000	5806.000000	7153.000000	11177.000000	13522.000000	16678.000000

In [4]:
```
NAN = [(c, covid_data[c].isna().mean()*100) for c in covid_data]
NAN = pd.DataFrame(NAN, columns=["column_name", "percentage"])
NAN
```

Out[4]:

	column_name	percentage
0	SNo	0.000000
1	ObservationDate	0.000000
2	Province/State	25.487144
3	Country/Region	0.000000
4	Last Update	0.000000
5	Confirmed	0.000000
6	Deaths	0.000000
7	Recovered	0.000000

34 % of Province/State are unknown we will fill nan values with Unknown. 0

In [5]:
```
covid_data["Province/State"]= covid_data["Province/State"].fillna('Unknown')
```

Change Data Type for "Confirmed","Deaths" and "Recovered" columns to int

In [6]:
```
covid_data[["Confirmed","Deaths","Recovered"]] =covid_data[["Confirmed","Deaths","Recovered"]].astype(int)
```

Replacing "Mainland China" with "China"

In [7]:
```
covid_data['Country/Region'] = covid_data['Country/Region'].replace('Mainland China', 'China')
```

- Creating new feature "Active_case"
- Active_case = Confirmed - Deaths - Recovered

In [8]:
```
covid_data['Active_case'] = covid_data['Confirmed'] - covid_data['Deaths'] - covid_data['Recovered']
```

In [9]:
```
covid_data.head()
```

Out[9]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	Active_case
0	1	01/22/2020	Anhui	China	1/22/2020 17:00	1	0	0	1
1	2	01/22/2020	Beijing	China	1/22/2020 17:00	14	0	0	14
2	3	01/22/2020	Chongqing	China	1/22/2020 17:00	6	0	0	6
3	4	01/22/2020	Fujian	China	1/22/2020 17:00	1	0	0	1
4	5	01/22/2020	Gansu	China	1/22/2020 17:00	0	0	0	0

In [10]:
```
Data = covid_data[covid_data['ObservationDate'] == max(covid_data['ObservationDate'])].reset_index()
```

Time series stats

In [11]:
```
base_stats = pd.DataFrame(columns=['Dates','Confirmed','Deaths','Recovered','Active'])
base_stats['Dates'] = covid_confirmed.columns[4:]

base_stats['Confirmed'] = base_stats['Dates'].apply(lambda x: covid_confirmed[x].sum())
base_stats['Deaths'] = base_stats['Dates'].apply(lambda x: covid_deaths[x].sum())
base_stats['Recovered'] = base_stats['Dates'].apply(lambda x: covid_recovered[x].sum())
base_stats.reset_index(drop=False, inplace=True)
base_stats['Active'] = base_stats['index'].apply(lambda x: (base_stats['Confirmed'][x]-(base_stats['Deaths'][x]+base_stats['Recovered'][x])))
base_stats.head()
```

Out[11]:

	index	Dates	Confirmed	Deaths	Recovered	Active
0	0	1/22/20	557	17	30	510
1	1	1/23/20	655	18	32	605
2	2	1/24/20	941	26	39	876
3	3	1/25/20	1433	42	42	1349
4	4	1/26/20	2118	56	56	2006

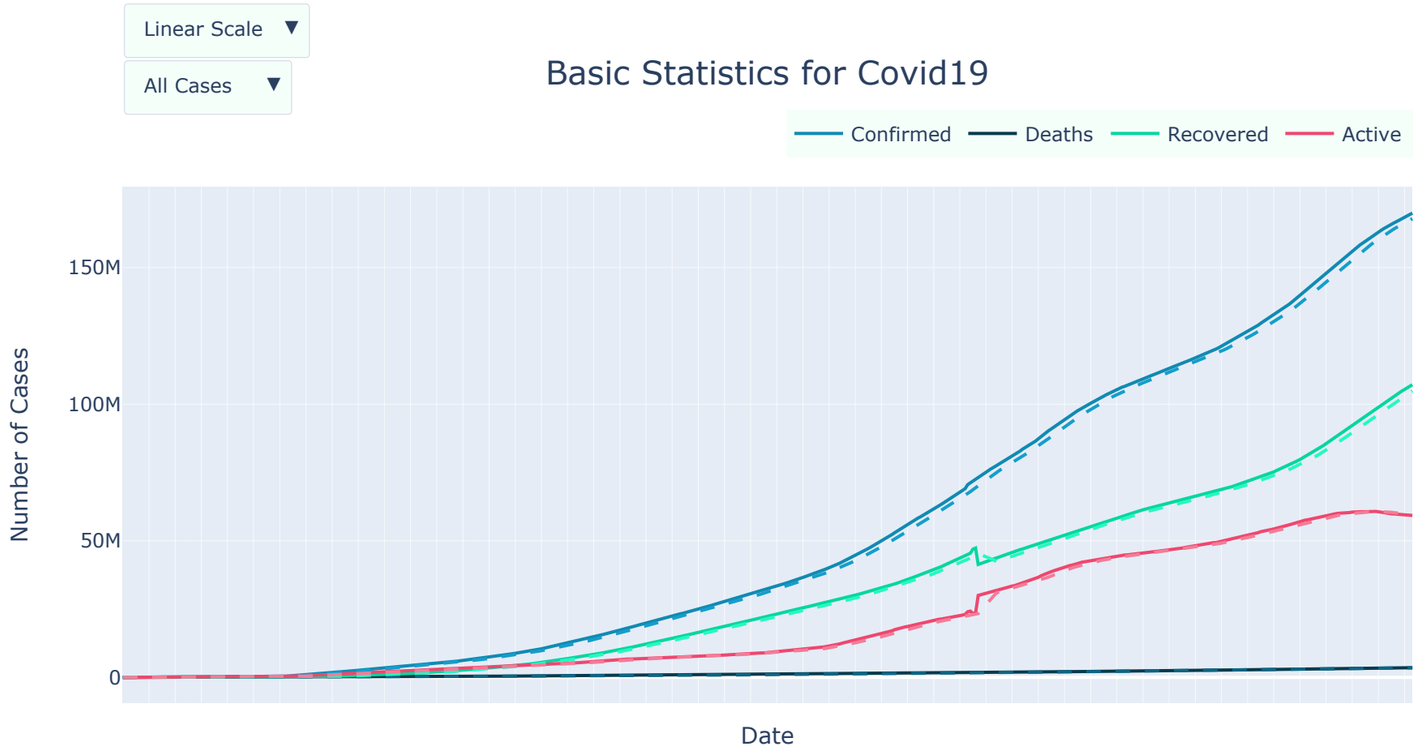
In [12]:
```
latest_stats_fig = go.Figure()
latest_stats_fig.add_trace(go.Treemap(labels = ['Confirmed','Active','Recovered','Deaths'],
                                      parents = ['','Confirmed','Confirmed','Confirmed'],
                                      values = [base_stats['Confirmed'].sum(), base_stats['Active'].sum(), base_stats['Recovered'].sum(), base_stats['Deaths'].sum()],
                                       branchvalues="total", marker_colors = ['#118ab2','#ef476f','#06d6a0','#073b4c'],
                                       textinfo = "label+text+value",
                                       outsidetextfont = {"size": 30, "color": "darkblue"},
                                       marker = {"line": {"width": 2}},
                                         pathbar = {"visible": False}
                                      ))
latest_stats_fig.update_layout(#width=1000,
                               height=300)
latest_stats_fig.show()
```

Cases over time around the world

In [13]:

```
base_stats_fig = go.Figure()

for column in base_stats.columns.to_list()[2:6]:
    color_dict = {
      "Confirmed": "#118ab2",
      "Active": "#ef476f",
      "Recovered": "#06d6a0",
      "Deaths": "#073b4c"
        }
    base_stats_fig.add_trace(
        go.Scatter(
            x = base_stats['Dates'],
            y = base_stats[column],
            name = column,
            line = dict(color=color_dict[column]),
            hovertemplate ='<br><b>Date</b>: %{x}'+'<br><i>Count</i>:'+'%{y}',
        )
    )

for column in base_stats.columns.to_list()[2:6]:
    color_dict = {
      "Confirmed": "#149ECC",
      "Active": "#F47C98",
      "Recovered": "#24F9C1",
      "Deaths": "#0C6583"
        }
    base_stats_fig.add_trace(
        go.Scatter(
            x = base_stats['Dates'],
            y = base_stats['index'].apply(lambda x: (base_stats[column][x-7:x].sum())/7 if x>7 else (base_stats[column][0:x].sum())/7),
            name = column+" 7-day Moving Avg.",
            line = dict(dash="dash", color=color_dict[column]), showlegend=False,
            hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>7-day moving avg.</i>: %{y}'
        )
    )

base_stats_fig.update_layout(
    updatemenus=[
        dict(
        buttons=list(
            [dict(label = 'All Cases',
                  method = 'update',
                  args = [{'visible': [True, True, True, True, True, True, True, True]},
                          {'title': 'All Cases',
                           'showlegend':True}]),
             dict(label = 'Confirmed',
                  method = 'update',
                  args = [{'visible': [True, False, False, False, True, False, False, False]},
                          {'title': 'Confirmed',
                           'showlegend':True}]),
             dict(label = 'Active',
                  method = 'update',
                  args = [{'visible': [False, False, False, True, False, False, False, True]},
                          {'title': 'Active',
                           'showlegend':True}]),
             dict(label = 'Recovered',
                  method = 'update',
                  args = [{'visible': [False, False, True, False, False, False, True, False]},
                          {'title': 'Recovered',
                           'showlegend':True}]),
             dict(label = 'Deaths',
```

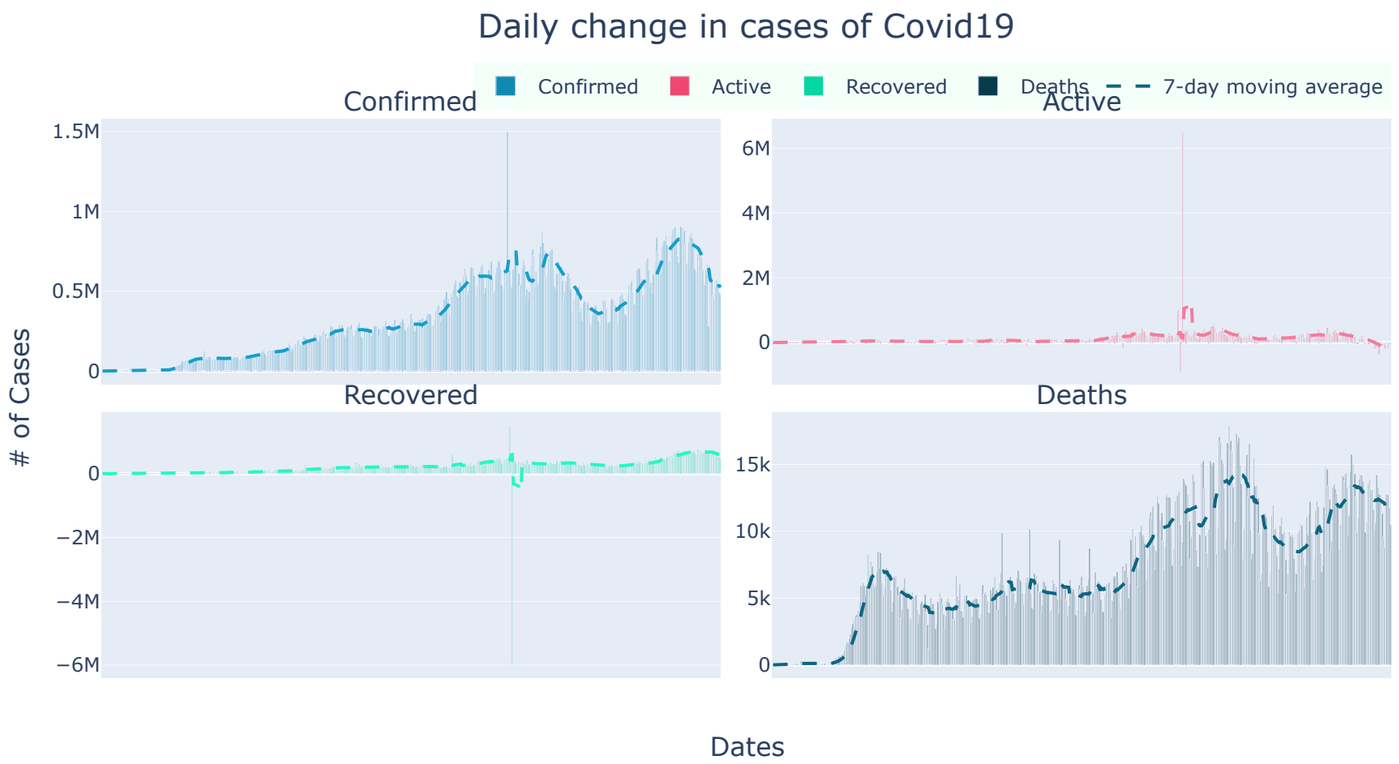
```
                method = 'update',
                args = [{'visible': [False, True, False, False, False, True, False, False]},
                        {'title': 'Deaths',
                         'showlegend':True}]),
            ]),
             type = "dropdown",
             direction="down",
#             pad={"r": 10, "t": 40},
             showactive=True,
             x=0,
             xanchor="left",
             y=1.25,
             yanchor="top"
        ),
        dict(
        buttons=list(
            [dict(label = 'Linear Scale',
                method = 'relayout',
                args = [{'yaxis': {'type': 'linear'}},
                        {'title': 'All Cases',
                         'showlegend':True}]),
             dict(label = 'Log Scale',
                method = 'relayout',
                args = [{'yaxis': {'type': 'log'}},
                        {'title': 'Confirmed',
                         'showlegend':True}]),
            ]),
             type = "dropdown",
             direction="down",
#             pad={"r": 10, "t": 10},
             showactive=True,
             x=0,
             xanchor="left",
             y=1.36,
             yanchor="top"
        )
    ])

base_stats_fig.update_xaxes(showticklabels=False)
base_stats_fig.update_layout(
    #height=600, width=600,
    title_text="Basic Statistics for Covid19", title_x=0.5, title_font_size=20,
                            legend=dict(orientation='h',yanchor='top',y=1.15,xanchor='right',x=1), paper_bgcolor="mintcream",
                            xaxis_title="Date", yaxis_title="Number of Cases")
base_stats_fig.show()
```


Linear Scale
All Cases
Basic Statistics for Covid19
Confirmed
Deaths
Recovered
Active
150M
100M
50M
0
Number of Cases
Date

Increase in cases

In [14]:

```
daily_case_fig = make_subplots(rows=2, cols=2, vertical_spacing=0.05, horizontal_spacing=0.04, # shared_yaxes=True,
                           subplot_titles=('Confirmed','Active','Recovered','Deaths'),
                            x_title='Dates', y_title='# of Cases',)

daily_case_fig.add_trace(go.Bar(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: base_stats['Confirmed'][x]-base_stats['Confirmed'][x-1:x].sum()),
                                name='Confirmed',hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>Confirmed Count</i>: %{y}',
                                  marker=dict(color='#118ab2')),row=1, col=1)
daily_case_fig.add_trace(go.Scatter(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: (base_stats['Confirmed'][x-7:x].sum()-base_stats['Confirmed'][x-8:x-1].sum())/
                                name='7-day moving average', hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>7-day average</i>: %{y}', showlegend=False,
                                      line=dict(dash="dash", color='#149ECC')),row=1, col=1)

daily_case_fig.add_trace(go.Bar(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: base_stats['Active'][x]-base_stats['Active'][x-1:x].sum()),
                                name='Active',hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>Active Count</i>: %{y}',
                                  marker=dict(color='#ef476f')),row=1, col=2)
daily_case_fig.add_trace(go.Scatter(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: (base_stats['Active'][x-7:x].sum()-base_stats['Active'][x-8:x-1].sum())/7 if x
                                name='7-day moving average', hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>7-day average</i>: %{y}', showlegend=False,
                                      line=dict(dash="dash", color='#F47C98')),row=1, col=2)

daily_case_fig.add_trace(go.Bar(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: base_stats['Recovered'][x]-base_stats['Recovered'][x-1:x].sum()),
                                 name='Recovered',hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>Recovered Count</i>: %{y}',
                                  marker=dict(color='#06d6a0')),row=2, col=1)
daily_case_fig.add_trace(go.Scatter(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: (base_stats['Recovered'][x-7:x].sum()-base_stats['Recovered'][x-8:x-1].sum())/
                                name='7-day moving average', hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>7-day average</i>: %{y}', showlegend=False,
                                      line=dict(dash="dash", color='#24F9C1')),row=2, col=1)

daily_case_fig.add_trace(go.Bar(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: base_stats['Deaths'][x]-base_stats['Deaths'][x-1:x].sum()),
                                 name='Deaths',hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>Death Count</i>: %{y}',
                                  marker=dict(color='#073b4c')),row=2, col=2)
daily_case_fig.add_trace(go.Scatter(x=base_stats['Dates'], y=base_stats['index'].apply(lambda x: (base_stats['Deaths'][x-7:x].sum()-base_stats['Deaths'][x-8:x-1].sum())/7 if x
                                name='7-day moving average', hovertemplate = '<br><b>Date</b>: %{x}'+'<br><i>7-day average</i>: %{y}', line=dict(dash="dash", color='#0C6583')),ro



daily_case_fig.update_xaxes(showticklabels=False)
daily_case_fig.update_layout(
    #height=600, width=1100,
    title_text="Daily change in cases of Covid19", title_x=0.5, title_font_size=20,
                               legend=dict(orientation='h',yanchor='top',y=1.1,xanchor='right',x=1), paper_bgcolor="mintcream")


daily_case_fig.show()
```



Confirmed cases in each Country

In [15]:
```
Data_per_country = covid_data.groupby(["Country/Region"])["Confirmed","Active_case","Recovered","Deaths"].sum().reset_index().sort_values("Confirmed",ascending=False).reset_in
```

In [16]:

```
headerColor = 'grey'
rowEvenColor = 'lightgrey'
rowOddColor = 'white'

fig = go.Figure(data=[go.Table(
  header=dict(
    values=['<b>Country</b>','<b>Confirmed Cases</b>'],
    line_color='darkslategray',
    fill_color=headerColor,
    align=['left','center'],

    font=dict(color='white', size=12)
  ),
  cells=dict(
    values=[
      Data_per_country['Country/Region'],
      Data_per_country['Confirmed'],
      ],
    line_color='darkslategray',
    # 2-D list of colors for alternating rows
    fill_color = [[rowOddColor,rowEvenColor,rowOddColor, rowEvenColor,rowOddColor]*len(Data_per_country)],
    align = ['left', 'center'],
    font = dict(color = 'darkslategray', size = 11)
    ))
])
fig.update_layout(
    title='Confirmed Cases In Each Country',
)
fig.show()
```

Confirmed Cases In Each Country

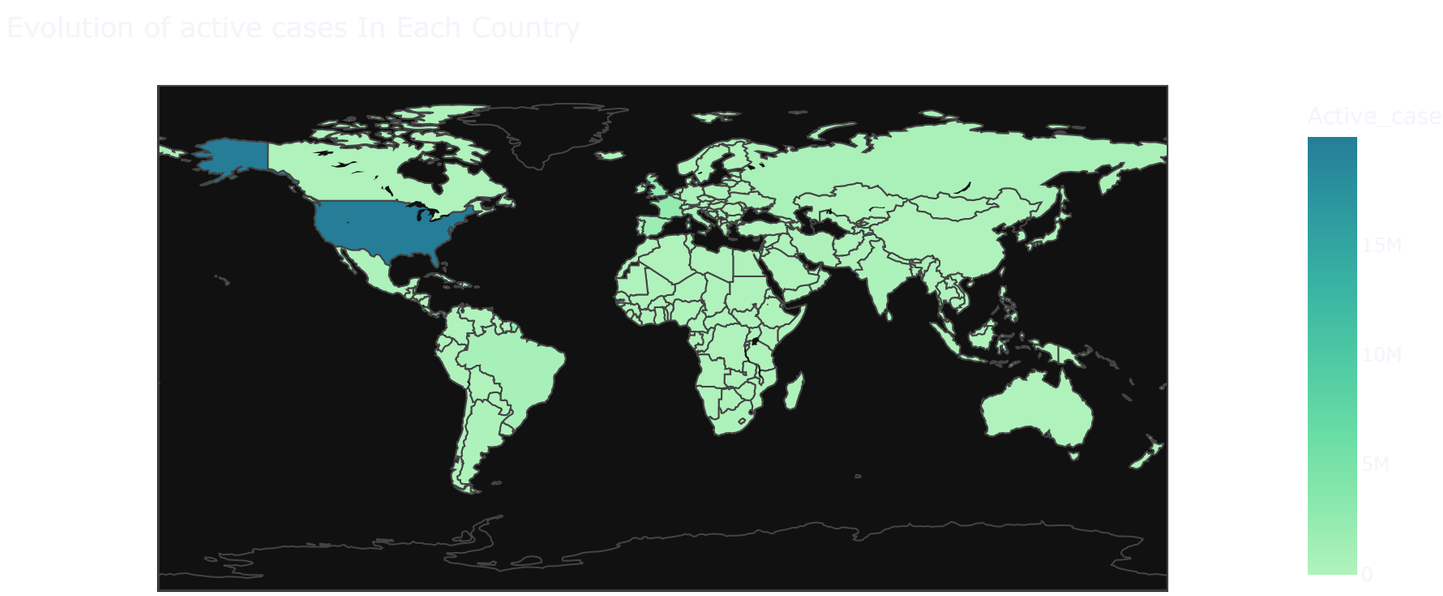
Country	Confirmed Cases
US	6049145667
India	3226768088
Brazil	2653587540
Russia	930548849
France	855188962
UK	783794384
Spain	649111763
Italy	636694305
Turkey	618940956
Germany	524166833
Colombia	515307146
Argentina	504802880
Mexico	460463678
Iran	400909778
Poland	380680836
Peru	361150607

Evolution of coronavirus over time.

In [17]:
```
data_per_country = covid_data.groupby(["Country/Region","ObservationDate"])[["Confirmed","Active_case","Recovered","Deaths"]].sum().reset_index().sort_values("ObservationDate"
```

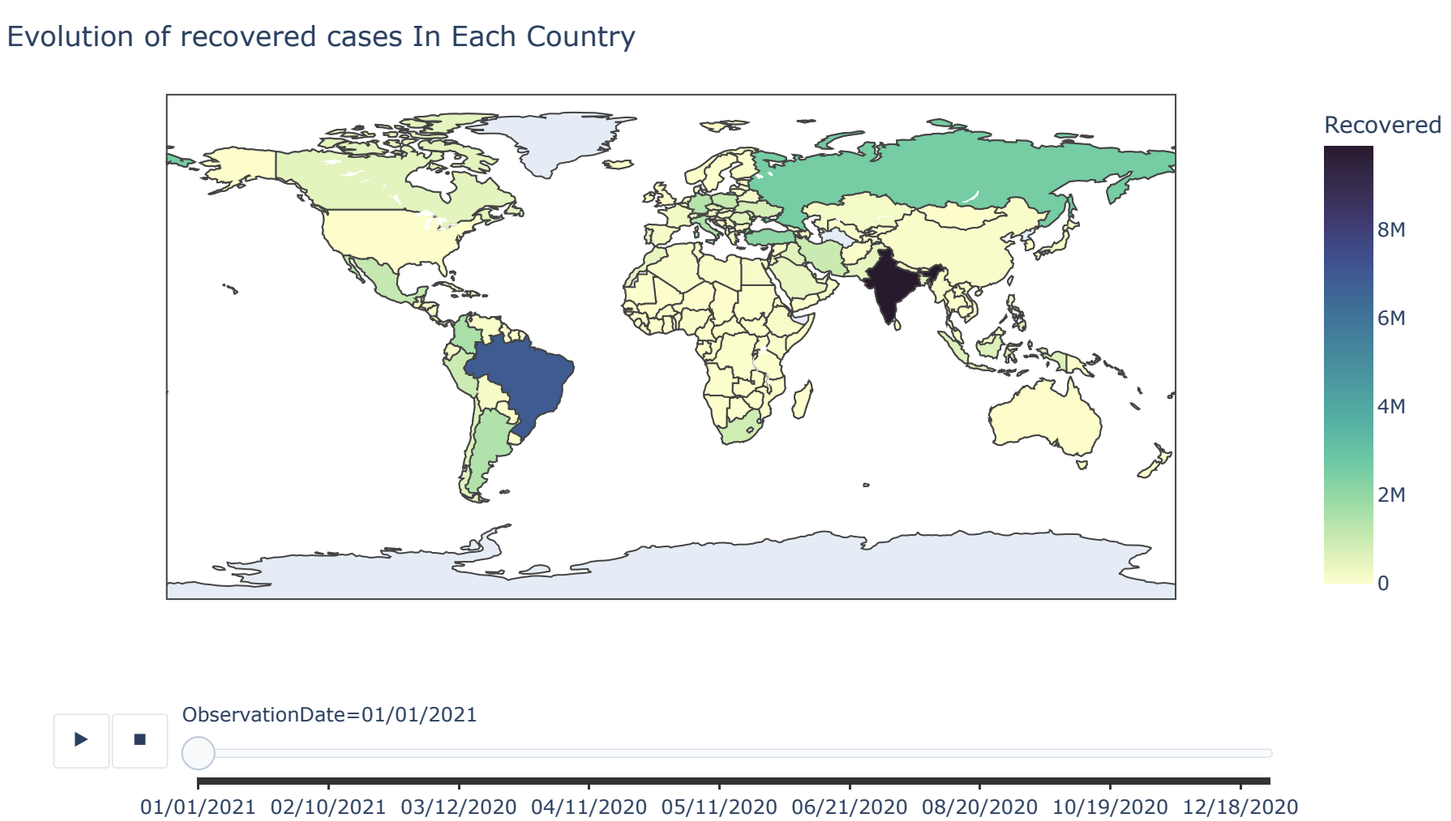
In [18]:

```
fig = px.choropleth(data_per_country, locations=data_per_country['Country/Region'],
                    color=data_per_country['Active_case'],locationmode='country names',
                    hover_name=data_per_country['Country/Region'],
                    color_continuous_scale=px.colors.sequential.Tealgrn,
                    animation_frame="ObservationDate")
fig.update_layout(

    title='Evolution of active cases In Each Country',
    template='plotly_dark'
)
fig.show()
```



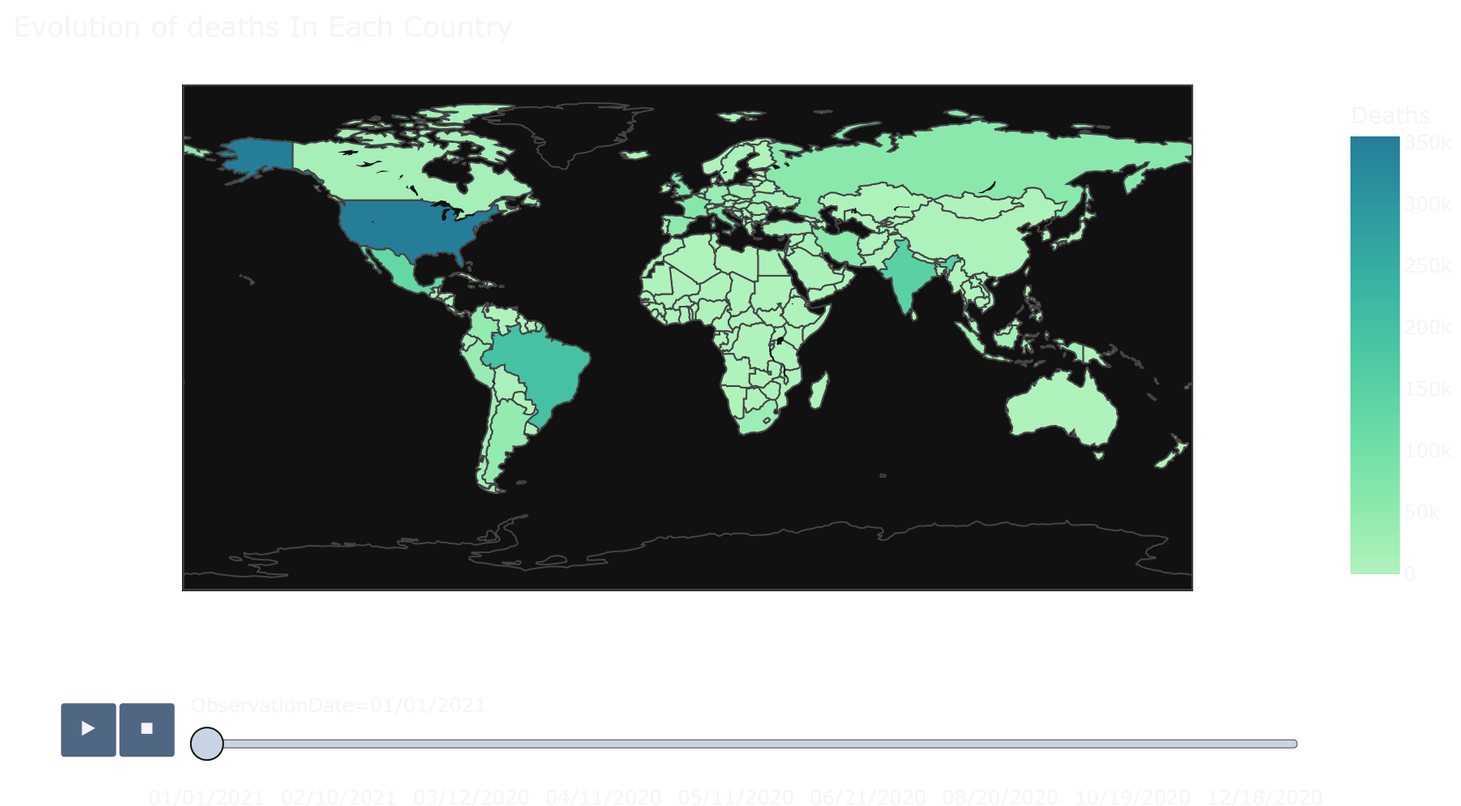
In [19]:

```
fig = px.choropleth(data_per_country, locations=data_per_country['Country/Region'],
                    color=data_per_country['Recovered'],locationmode='country names',
                    hover_name=data_per_country['Country/Region'],
                    color_continuous_scale=px.colors.sequential.deep,
                    animation_frame="ObservationDate")
fig.update_layout(
    title='Evolution of recovered cases In Each Country',
)
fig.show()
```

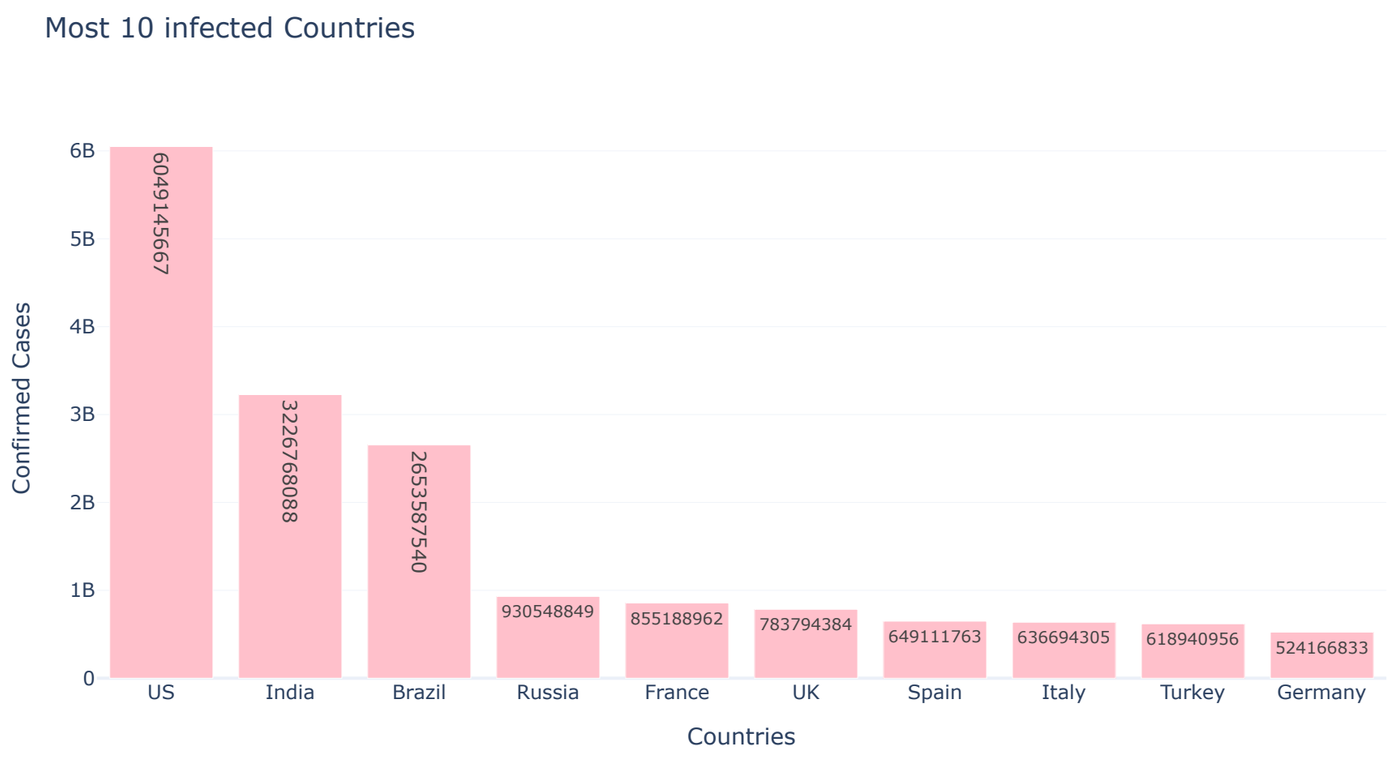


In [20]:

```
fig = px.choropleth(data_per_country, locations=data_per_country['Country/Region'],
                    color=data_per_country['Deaths'],locationmode='country names',
                    hover_name=data_per_country['Country/Region'],
                    color_continuous_scale=px.colors.sequential.Tealgrn,
                    animation_frame="ObservationDate")
fig.update_layout(
    title='Evolution of deaths In Each Country',
    template='plotly_dark'
)
fig.show()
```



In [21]:
```
fig = go.Figure(data=[go.Bar(
            x=Data_per_country['Country/Region'][0:10], y=Data_per_country['Confirmed'][0:10],
            text=Data_per_country['Confirmed'][0:10],
            textposition='auto',
            marker_color='pink',


        )])
fig.update_layout(
    title='Most 10 infected Countries',
    xaxis_title="Countries",
    yaxis_title="Confirmed Cases",
        template='plotly_white'

)
fig.show()
```



Recorvered cases in each Country

In [22]:
```
Recovered_per_country = covid_data.groupby(["Country/Region"])["Recovered"].sum().reset_index().sort_values("Recovered",ascending=False).reset_index(drop=True)
```

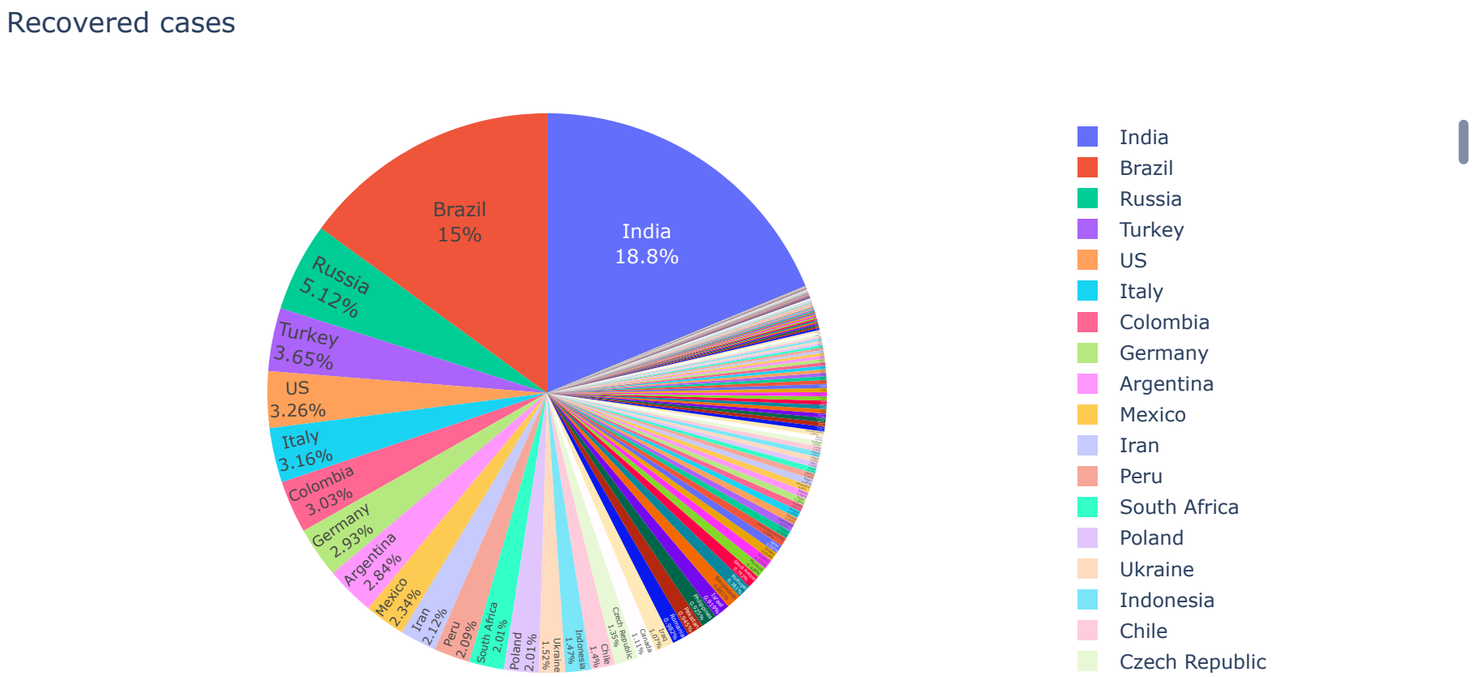
In [23]:

```
headerColor = 'grey'
rowEvenColor = 'lightgrey'
rowOddColor = 'white'

fig = go.Figure(data=[go.Table(
  header=dict(
    values=['<b>Country</b>','<b>Recovered Cases</b>'],
    line_color='darkslategray',
    fill_color=headerColor,
    align=['left','center'],
    font=dict(color='white', size=12)
  ),
  cells=dict(
    values=[
      Recovered_per_country['Country/Region'],
      Recovered_per_country['Recovered'],
      ],
    line_color='darkslategray',
    # 2-D list of colors for alternating rows
    fill_color = [[rowOddColor,rowEvenColor,rowOddColor, rowEvenColor,rowOddColor]*len(Data_per_country)],
    align = ['left', 'center'],
    font = dict(color = 'darkslategray', size = 11)
    ))
])
fig.update_layout(
    title='Recovered Cases In Each Country',
)
fig.show()
```

Recovered Cases In Each Country

Country	Recovered Cases
India	2900589824
Brazil	2313677028
Russia	790705716
Turkey	564170558
US	503370956
Italy	487799849
Colombia	468747010
Germany	453383253
Argentina	438750295
Mexico	361780181
Iran	326813025
Peru	323672775
South Africa	310037579
Poland	309835430
Ukraine	234410665
Indonesia	226416174

In [24]:
```
fig = px.pie(Recovered_per_country, values=Recovered_per_country['Recovered'], names=Recovered_per_country['Country/Region'],
             title='Recovered cases',
            )
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(
    template='plotly_white'
)
fig.show()
```



Active cases in each Country

In [25]:
```
Active_per_country = covid_data.groupby(["Country/Region"])["Active_case"].sum().reset_index().sort_values("Active_case",ascending=False).reset_index(drop=True)
```

In [26]:

```
headerColor = 'grey'
rowEvenColor = 'lightgrey'
rowOddColor = 'white'

fig = go.Figure(data=[go.Table(
  header=dict(
    values=['<b>Country</b>','<b>Active Cases</b>'],
    line_color='darkslategray',
    fill_color=headerColor,
    align=['left','center'],
    font=dict(color='white', size=12)
  ),
  cells=dict(
    values=[
      Active_per_country['Country/Region'],
      Active_per_country['Active_case'],
      ],
    line_color='darkslategray',
    # 2-D list of colors for alternating rows
    fill_color = [[rowOddColor,rowEvenColor,rowOddColor, rowEvenColor,rowOddColor]*len(Data_per_country)],
    align = ['left', 'center'],
    font = dict(color = 'darkslategray', size = 11)
    ))
])
fig.update_layout(
    title='Active Cases In Each Country',
)
fig.show()
```

Active Cases In Each Country

Country	Active Cases
US	5422470949
France	763060711
UK	752294828
Spain	568853041
India	281753541
Brazil	267285902
Netherlands	227486544
Belgium	168919413
Sweden	138768695
Italy	122893754
Russia	121479414
Serbia	93932825
Ukraine	70271100
Poland	61893730
Iran	58352346
Germany	57419364

Deaths cases in each Country

In [27]:
```
Deaths_per_country = covid_data.groupby(["Country/Region"])["Deaths"].sum().reset_index().sort_values("Deaths",ascending=False).reset_index(drop=True)
```

In [28]:

```
headerColor = 'grey'
rowEvenColor = 'lightgrey'
rowOddColor = 'white'

fig = go.Figure(data=[go.Table(
  header=dict(
    values=['<b>Country</b>','<b>Deaths</b>'],
    line_color='darkslategray',
    fill_color=headerColor,
    align=['left','center'],
    font=dict(color='white', size=12)
  ),
  cells=dict(
    values=[
      Deaths_per_country['Country/Region'],
      Deaths_per_country['Deaths'],
      ],
    line_color='darkslategray',
    # 2-D list of colors for alternating rows
    fill_color = [[rowOddColor,rowEvenColor,rowOddColor, rowEvenColor,rowOddColor]*len(Data_per_country)],
    align = ['left', 'center'],
    font = dict(color = 'darkslategray', size = 11)
    ))
])
fig.update_layout(
    title='Deaths In Each Country',
)
fig.show()
```

Deaths In Each Country

Country	Deaths
US	123303762
Brazil	72624610
India	44424723
Mexico	43005509
UK	29171984
Italy	26000702
France	22720818
Spain	19065104
Russia	18363719
Iran	15744407
Colombia	13981703
Germany	13364216
Peru	13194771
Argentina	12112441
South Africa	10250036
Poland	8951676

Predictions

In [29]:
```
base_stats_inc_df = pd.DataFrame(columns=['Index', 'Dates', 'Confirmed', 'Deaths', 'Recovered', 'Active', 'Daily Inc.'])
base_stats_inc_df[['Index', 'Dates', 'Confirmed', 'Deaths', 'Recovered', 'Active']] = base_stats[['index', 'Dates', 'Confirmed', 'Deaths', 'Recovered', 'Active']]
base_stats_inc_df['Daily Inc.'] = base_stats['index'].apply(lambda x: base_stats['Confirmed'][x]-base_stats['Confirmed'][x-1:x].sum())
```

In [30]:
```
days = np.array(base_stats_inc_df[['Index']]).reshape(-1, 1)
days_ex = []
for i in range(len(days)+30):
    days_ex = days_ex+[[i]]
```

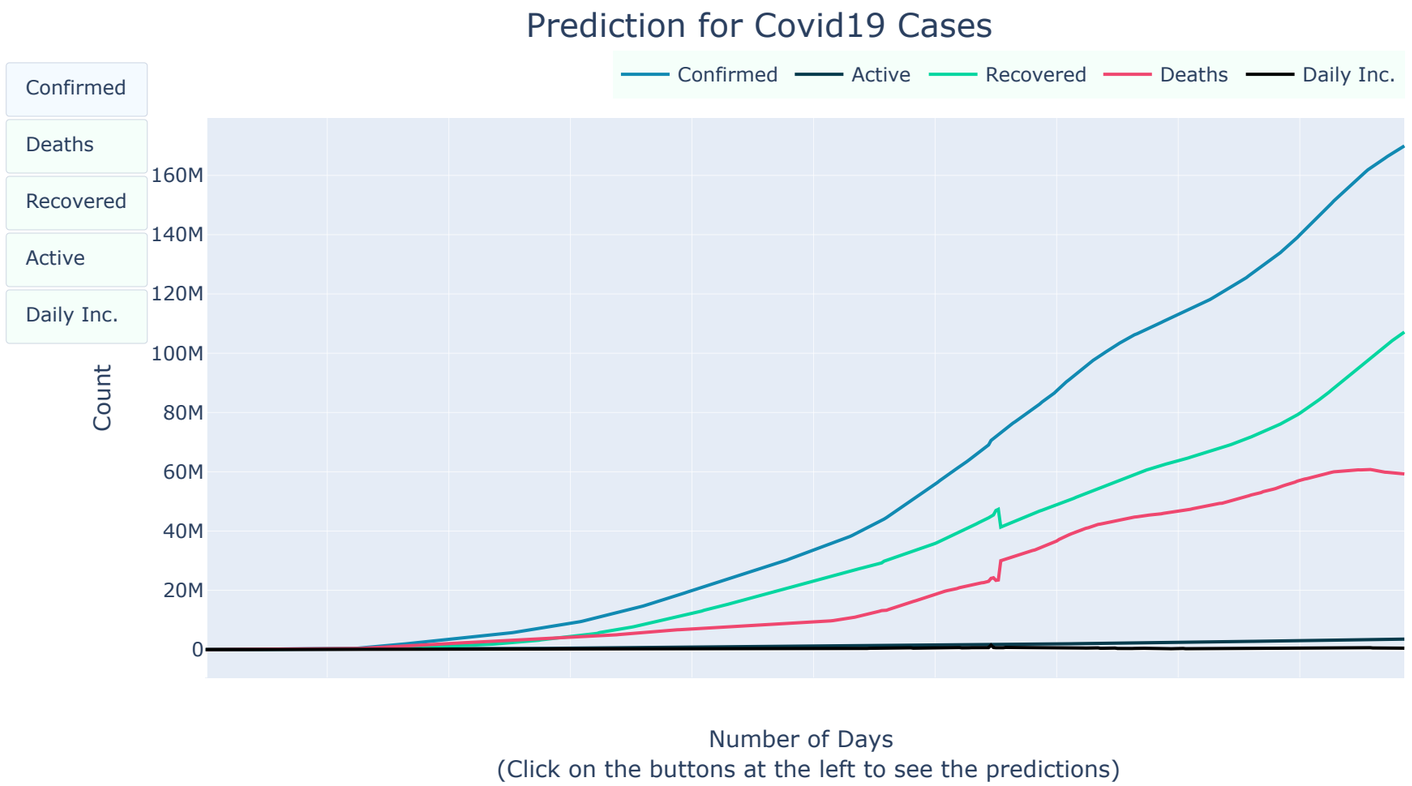
In [31]:
```
prediction_df = pd.DataFrame(columns=['Index', 'Confirmed Pred', 'Deaths Pred', 'Recovered Pred', 'Active Pred', 'Daily Inc. Pred'])
prediction_df['Index'] = list(flatten(days_ex))
```

In [32]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.metrics import r2_score


for col in base_stats_inc_df.columns[2:]:

    count = np.array(base_stats_inc_df[[col]]).reshape(-1, 1)

    X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed = train_test_split(
                                                    days[50:], count[50:],
                                                    test_size=0.05, shuffle=False)

MAE, RSE, R2 = [], [], []
for j in range(1,10):
    #creating the model
    poly = PolynomialFeatures(degree=j)
    train_x_poly = poly.fit_transform(X_train_confirmed)

    regr_poly = linear_model.LinearRegression()
    regr_poly.fit(train_x_poly, y_train_confirmed)

    y_pred_poly = regr_poly.predict(poly.fit_transform(X_test_confirmed))
    MAE.append(np.mean(np.absolute(y_pred_poly - y_test_confirmed)))
    RSE.append(np.mean((y_pred_poly - list(flatten(y_test_confirmed))) ** 2))
    R2.append(r2_score(y_pred_poly, list(flatten(y_test_confirmed))))

deg = RSE.index(min(RSE))+1
#print("best deg for column {} is {}".format(col, deg))

poly = PolynomialFeatures(degree=deg)
train_x_poly = poly.fit_transform(X_train_confirmed)

regr_poly = linear_model.LinearRegression()
regr_poly.fit(train_x_poly, y_train_confirmed)
col_name = col+' Pred'
prediction_df[col_name] = list(flatten(regr_poly.predict(poly.fit_transform(days_ex))))
```

In [33]:

```
prediction_fig = go.Figure()
pred_dict = {
  "Confirmed": ["#118ab2", 'Confirmed', 'Predicted Confirmed','#149ECC'],
  "Active": ["#ef476f", 'Deaths', 'Predicted Deaths', '#F47C98'],
  "Recovered": ["#06d6a0", 'Recovered', 'Predicted Recovered','#24F9C1'],
  "Deaths": ["#073b4c", 'Active', 'Predicted Active','#0C6583'],
  "Daily Inc.": ["black", 'Daily Inc.', 'Predicted Daily Inc.','grey']
    }

for z in base_stats_inc_df.columns[2:]:

    z_pred = z+' Pred'
    prediction_fig.add_trace(go.Scatter(x=list(flatten(days)), y=base_stats_inc_df[z],
                                        line=dict(color=pred_dict[z][0]), name = pred_dict[z][1],
                                        hovertemplate ='<br><b>Day number</b>: %{x}'+'<br><i>No.of cases </i>:'+'%{y}'))

    prediction_fig.add_trace(go.Scatter(x=list(flatten(days_ex))[50:], y=prediction_df[z_pred][50:],
                                        line=dict(dash="dash", color='black'), visible=False, name = pred_dict[z][2],
                                        hovertemplate ='<br><b>Day number</b>: %{x}'+'<br><i>Predicted no.of cases </i>:'+'%{y}'))




prediction_fig.update_layout(
    updatemenus=[
        dict(
        buttons=list(
            [dict(label = 'Confirmed',
                  method = 'update',
                  args = [{'visible': [True, True, False, False, False, False, False, False, False, False]},
                          {'title': 'Confirmed Cases',
                           'showlegend':True}]),
             dict(label = 'Deaths',
                  method = 'update',
                  args = [{'visible': [False, False, True, True, False, False, False, False, False, False]},
                          {'title': 'Deaths Cases',
                           'showlegend':True}]),
             dict(label = 'Recovered',
                  method = 'update',
                  args = [{'visible': [False, False, False, False, True, True, False, False, False, False]},
                          {'title': 'Recovered Cases',
                           'showlegend':True}]),
             dict(label = 'Active',
                  method = 'update',
                  args = [{'visible': [False, False, False, False, False, False, True, True, False, False]},
                          {'title': 'Active Cases',
                           'showlegend':True}]),
             dict(label = 'Daily Inc.',
                  method = 'update',
                  args = [{'visible': [False, False, False, False, False, False, False, False, True, True]},
                          {'title': 'Daily Inc. Cases',
                           'showlegend':True}]),
            ]),
             type = "buttons",
             direction="down",
#             pad={"r": 10, "t": 40},
             showactive=True,
#             x=1.01,
#             xanchor="left",
             y=1.1,
             yanchor="top"
```

```
        )
    ])


prediction_fig.update_xaxes(showticklabels=False)
prediction_fig.update_layout(
    #height=500, width=1100,
    title_text="Prediction for Covid19 Cases", title_x=0.5, title_font_size=20,
                            legend=dict(orientation='h',yanchor='top',y=1.12,xanchor='right',x=1), paper_bgcolor="mintcream",
                            xaxis_title="Number of Days <br> (Click on the buttons at the left to see the predictions)", yaxis_title="Count")
prediction_fig.show()
```



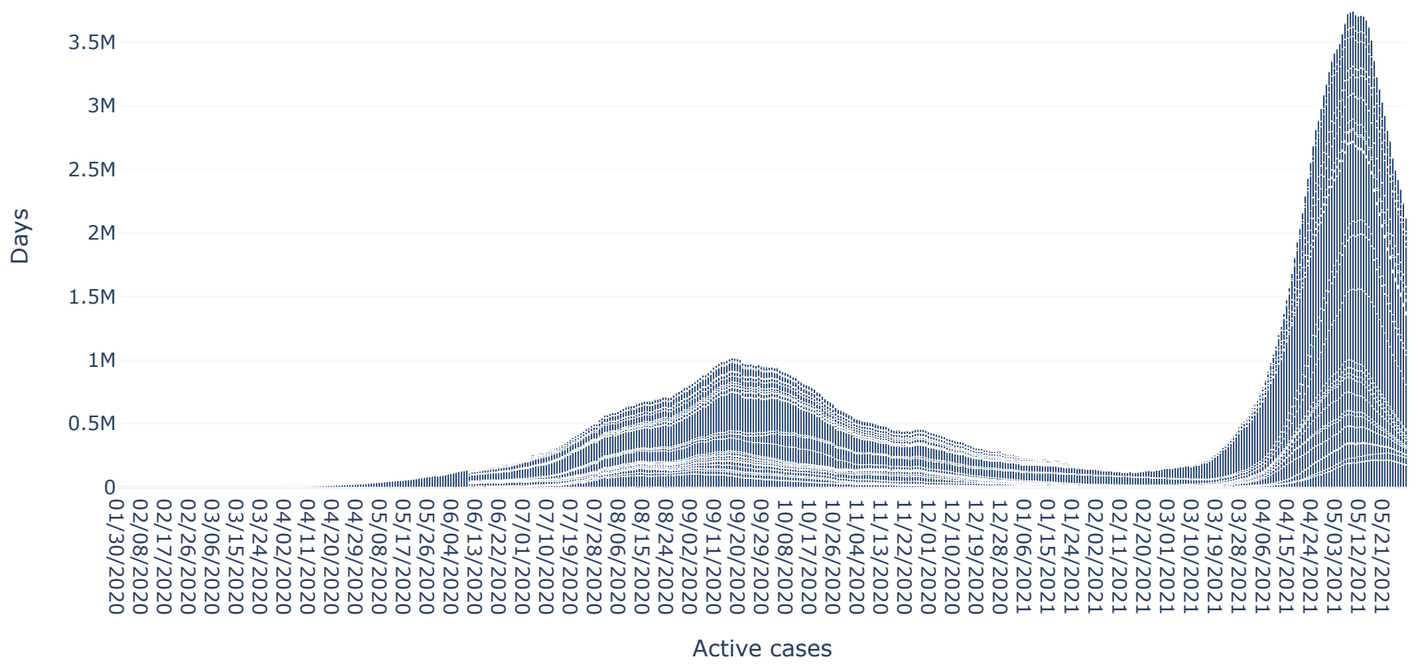
Coronavirus in my country India

In [34]:
```
Data_india = covid_data [(covid_data['Country/Region'] == 'India') ].reset_index(drop=True)
Data_india.head()
```

Out[34]:

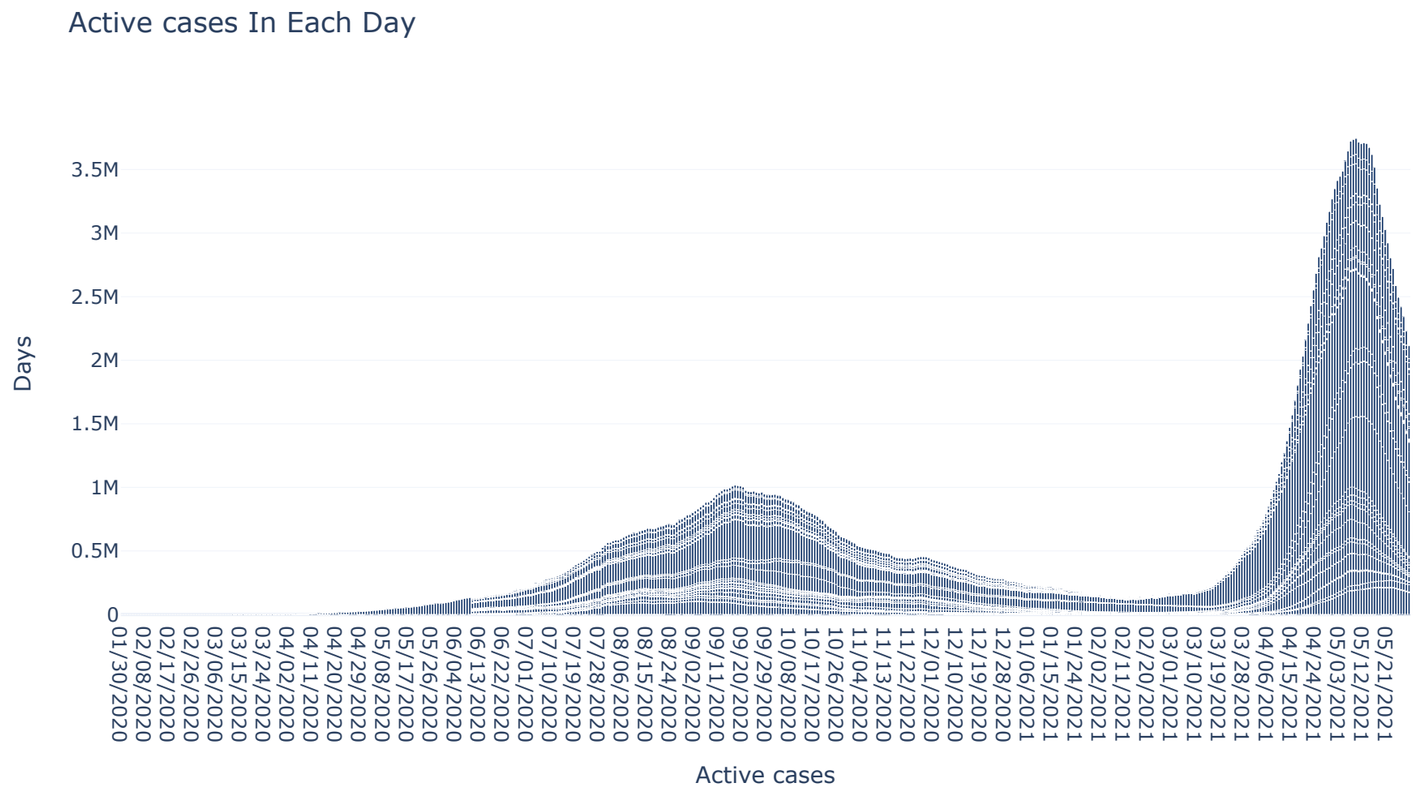
	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	Active_case
0	447	01/30/2020	Unknown	India	1/30/20 16:00	1	0	0	1
1	510	01/31/2020	Unknown	India	1/31/2020 23:59	1	0	0	1
2	568	02/01/2020	Unknown	India	1/31/2020 8:15	1	0	0	1
3	630	02/02/2020	Unknown	India	2020-02-02T06:03:08	2	0	0	2
4	697	02/03/2020	Unknown	India	2020-02-03T21:43:02	3	0	0	3

In [35]:
```
fig = go.Figure(go.Bar(
            x=Data_india['ObservationDate'],
            y=Data_india['Active_case'],
    marker_color='rgb(13,48,100)'
           ))
fig.update_layout(
    title='Active cases In Each Day',
    template='plotly_white',
     xaxis_title="Active cases",
    yaxis_title="Days",
)
fig.show()
```



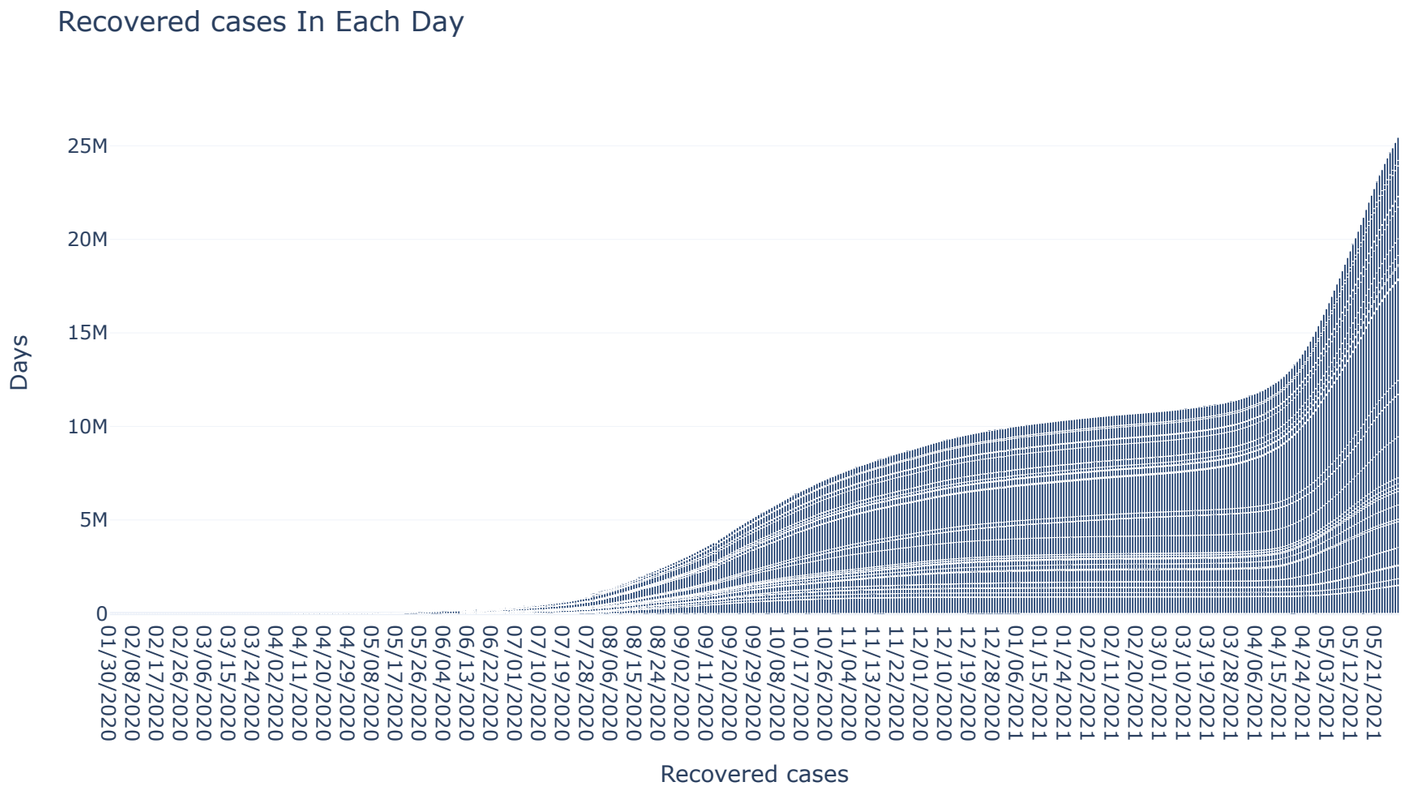
In [36]:

```
fig = go.Figure(go.Bar(
            x=Data_india['ObservationDate'],
            y=Data_india['Active_case'],
    marker_color='rgb(13,48,100)'
           ))
fig.update_layout(
    title='Active cases In Each Day',
    template='plotly_white',
     xaxis_title="Active cases",
    yaxis_title="Days",
)
fig.show()
```

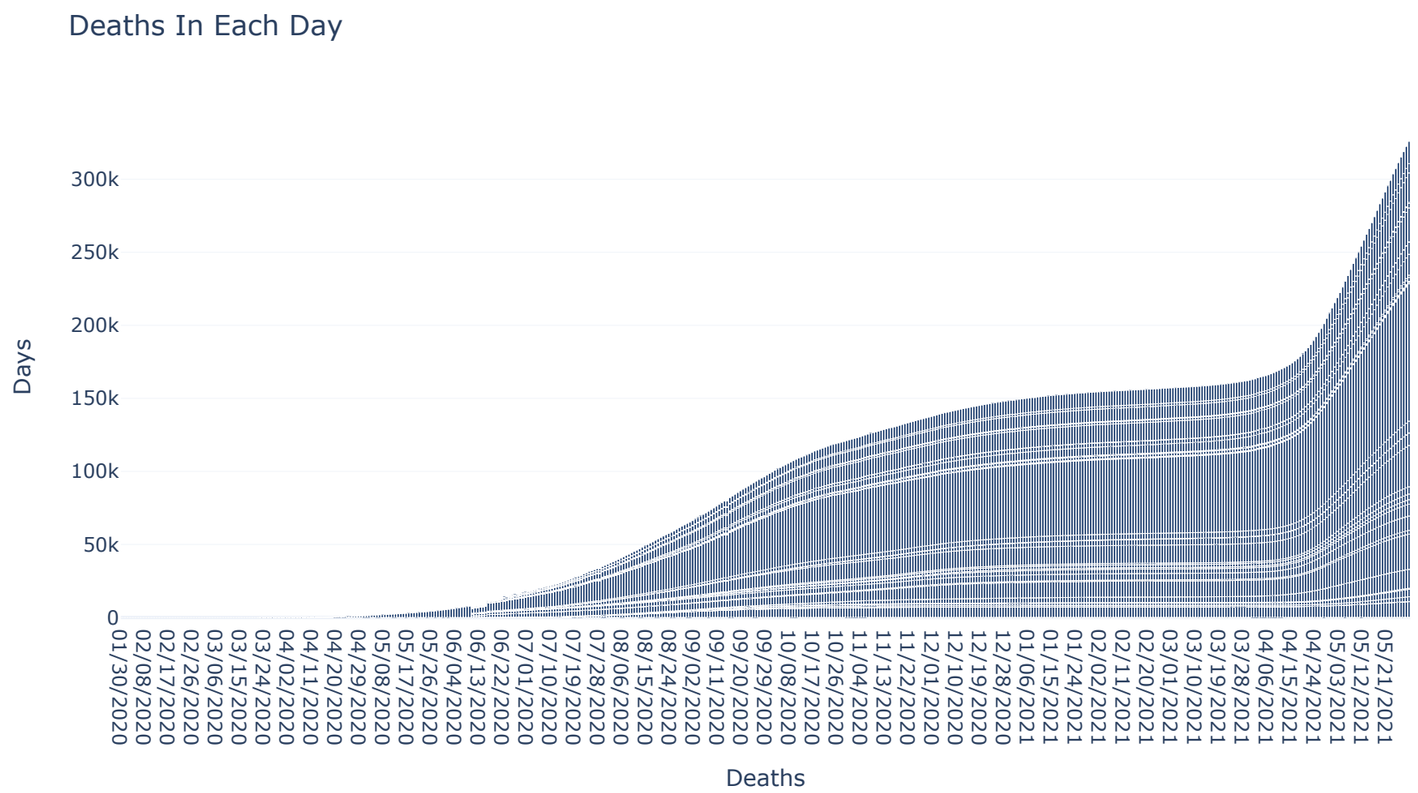


In [37]:

```
fig = go.Figure(go.Bar(
            x=Data_india['ObservationDate'],
            y=Data_india['Recovered'],
    marker_color='rgb(13,48,100)'
           ))
fig.update_layout(
    title='Recovered cases In Each Day',
    template='plotly_white',
     xaxis_title="Recovered cases",
    yaxis_title="Days",
)
fig.show()
```

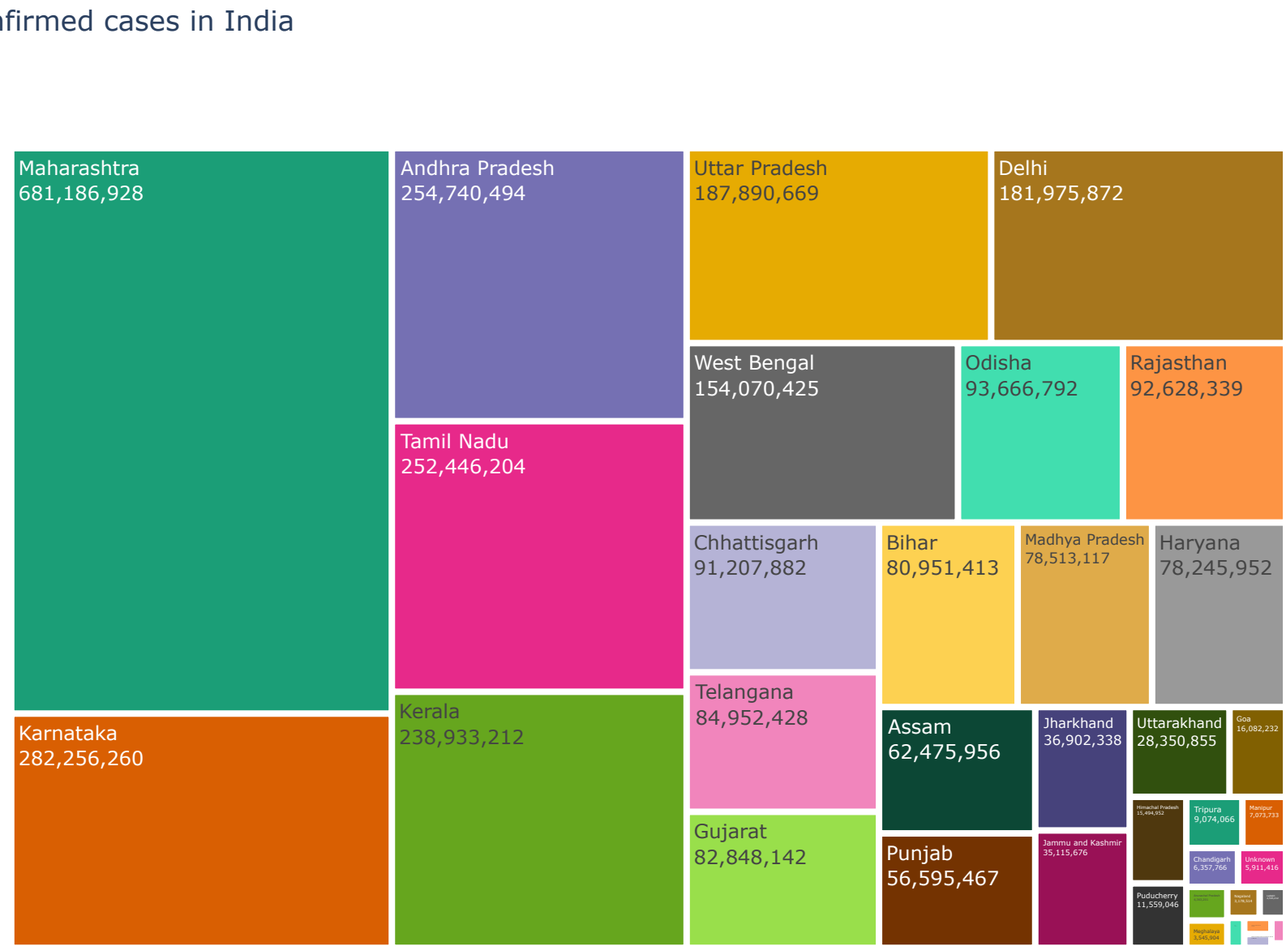


In [38]:
```
fig = go.Figure(go.Bar(
            x=Data_india['ObservationDate'],
            y=Data_india['Deaths'],
    marker_color='rgb(13,48,100)'
           ))
fig.update_layout(
    title='Deaths In Each Day',
    template='plotly_white',
     xaxis_title="Deaths",
    yaxis_title="Days",
)
fig.show()
```

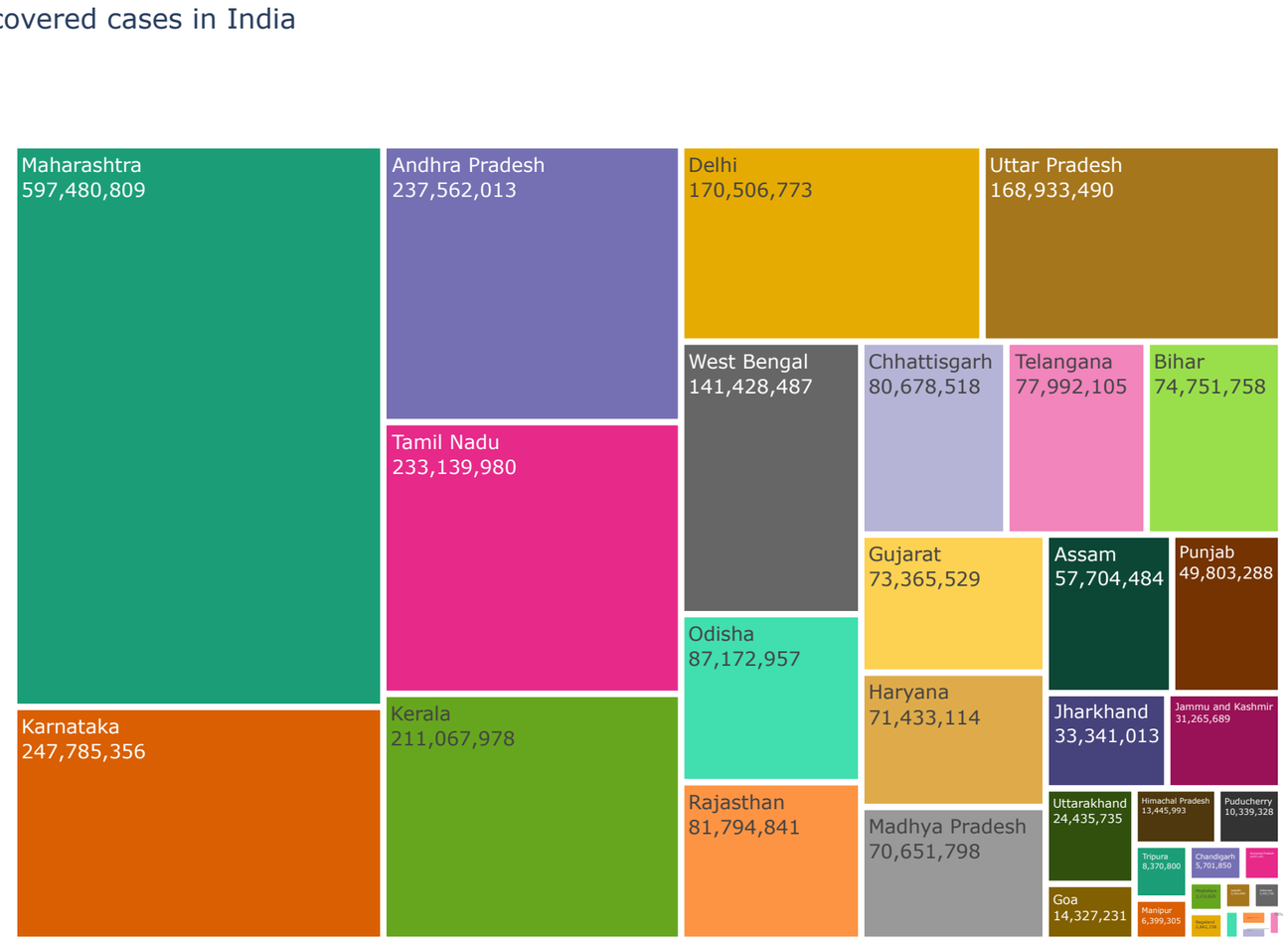


In [39]:
```
Data_India_per_state= Data_india.groupby(["Province/State"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().sort_values("Confirmed",ascending=False).reset
```

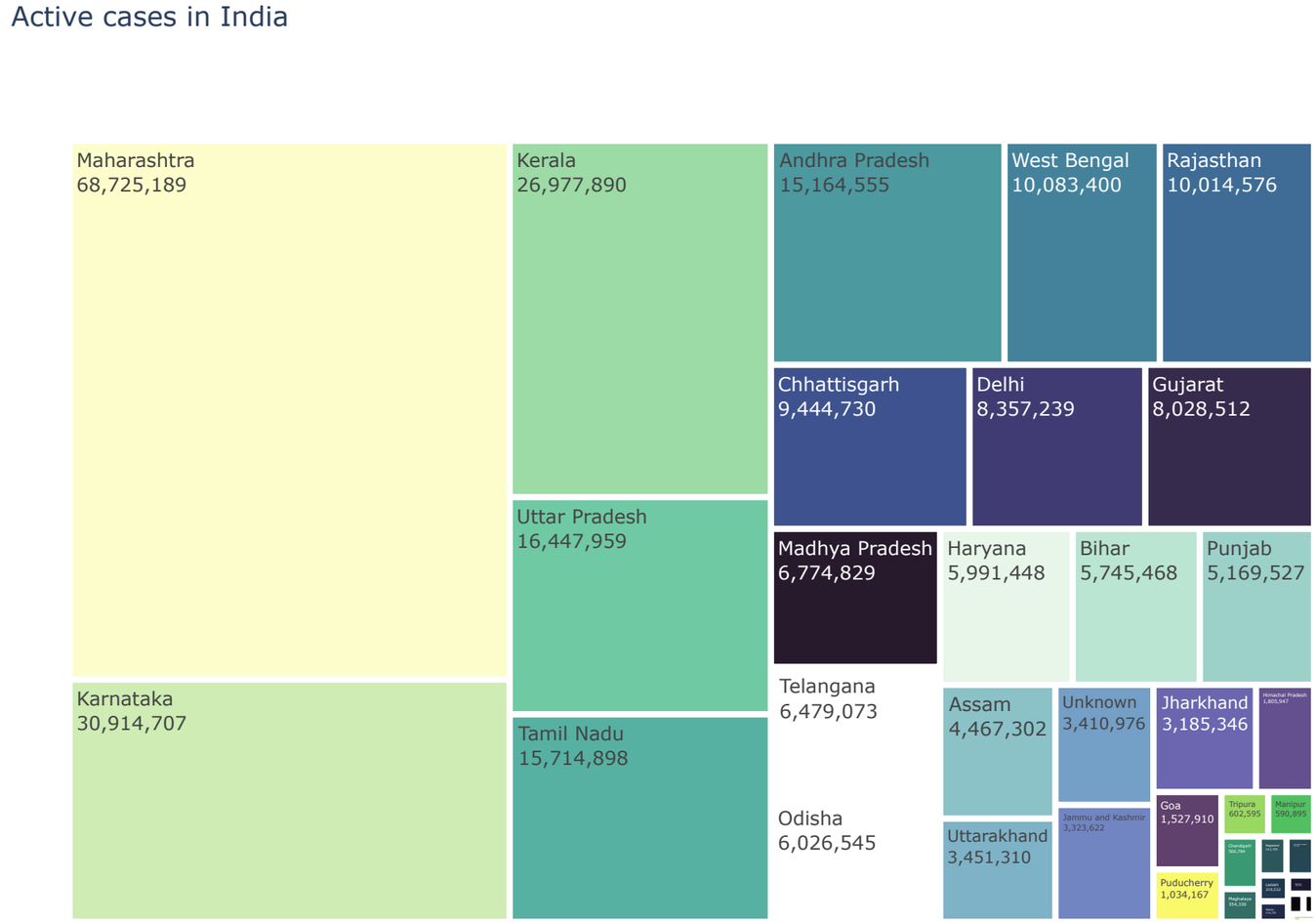
In [40]:
```
fig = px.treemap(Data_India_per_state, path=['Province/State'], values=Data_India_per_state['Confirmed'], height=700,
                 title='Confirmed cases in India', color_discrete_sequence = px.colors.qualitative.Dark2)
fig.data[0].textinfo = 'label+text+value'
fig.show()
```



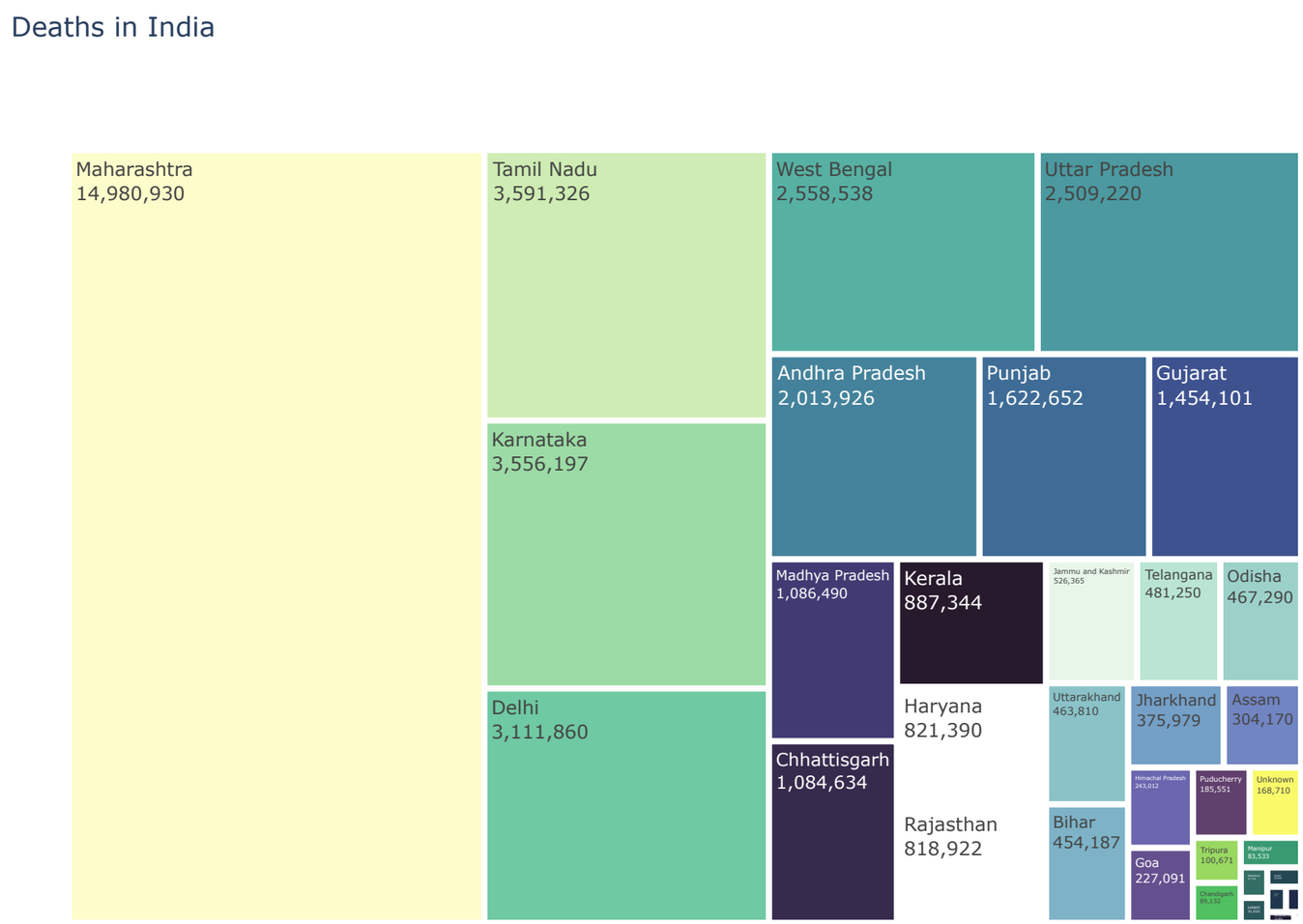
In [41]:
```
fig = px.treemap(Data_India_per_state, path=['Province/State'], values=Data_India_per_state['Recovered'], height=700,
                 title='Recovered cases in India', color_discrete_sequence = px.colors.qualitative.Dark2)
fig.data[0].textinfo = 'label+text+value'
fig.show()
```



In [42]:
```
fig = px.treemap(Data_India_per_state, path=['Province/State'], values=Data_India_per_state['Active_case'], height=700,
                 title='Active cases in India', color_discrete_sequence = px.colors.sequential.deep)
fig.data[0].textinfo = 'label+text+value'
fig.show()
```



In [43]:
```
fig = px.treemap(Data_India_per_state, path=['Province/State'], values=Data_India_per_state['Deaths'], height=700,
                 title='Deaths in India', color_discrete_sequence = px.colors.sequential.deep)
fig.data[0].textinfo = 'label+text+value'
fig.show()
```



Get Last Update

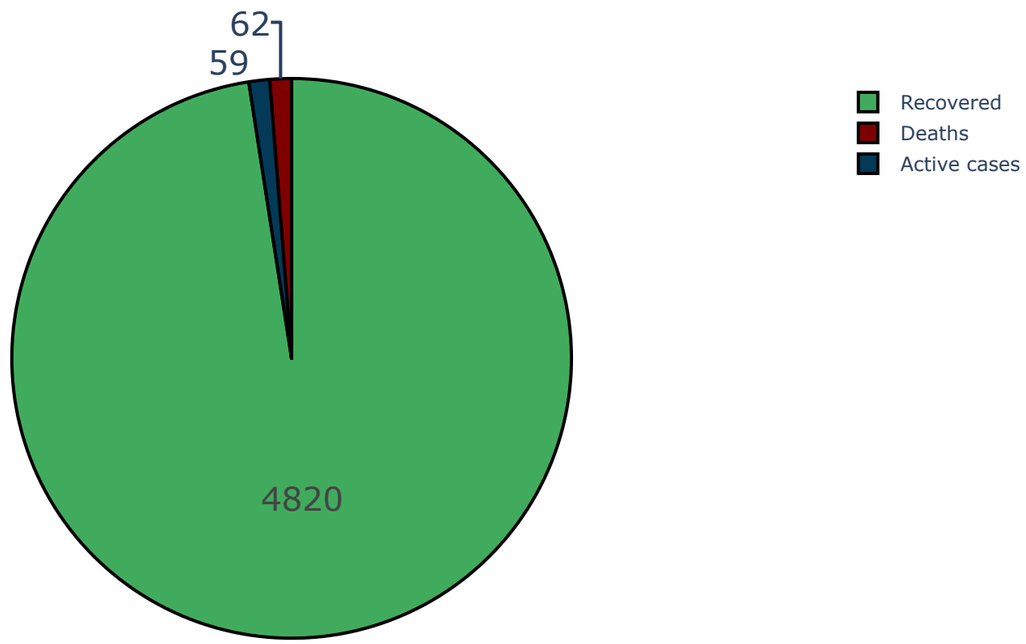
In [44]:

```
Data_india_last = Data_india[Data_india['ObservationDate'] == max(Data_india['ObservationDate'])].reset_index()
Data_india_last
```

Out[44]:

	index	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	Active_case
0	7632	191894	12/31/2020	Andaman and Nicobar Islands	India	2021-04-02 15:13:53	4941	62	4820	59
1	7633	191895	12/31/2020	Andhra Pradesh	India	2021-04-02 15:13:53	881948	7104	871588	3256
2	7634	191912	12/31/2020	Arunachal Pradesh	India	2021-04-02 15:13:53	16711	56	16549	106
3	7635	191913	12/31/2020	Assam	India	2021-04-02 15:13:53	216139	1043	211838	3258
4	7636	191935	12/31/2020	Bihar	India	2021-04-02 15:13:53	251348	1393	245156	4799
5	7637	191969	12/31/2020	Chandigarh	India	2021-04-02 15:13:53	19682	316	18967	399
6	7638	191976	12/31/2020	Chhattisgarh	India	2021-04-02 15:13:53	278540	3350	263251	11939
7	7639	191995	12/31/2020	Dadra and Nagar Haveli and Daman and Diu	India	2021-04-02 15:13:53	3375	2	3364	9
8	7640	191999	12/31/2020	Delhi	India	2021-04-02 15:13:53	624795	10523	608434	5838
9	7641	192035	12/31/2020	Goa	India	2021-04-02 15:13:53	50981	737	49313	931
10	7642	192051	12/31/2020	Gujarat	India	2021-04-02 15:13:53	244258	4302	229977	9979
11	7643	192057	12/31/2020	Haryana	India	2021-04-02 15:13:53	262054	2899	255356	3799
12	7644	192064	12/31/2020	Himachal Pradesh	India	2021-04-02 15:13:53	55114	931	51387	2796
13	7645	192090	12/31/2020	Jammu and Kashmir	India	2021-04-02 15:13:53	120744	1880	115830	3034
14	7646	192093	12/31/2020	Jharkhand	India	2021-04-02 15:13:53	114873	1027	112206	1640
15	7647	192111	12/31/2020	Karnataka	India	2021-04-02 15:13:53	918544	12081	894834	11629
16	7648	192114	12/31/2020	Kerala	India	2021-04-02 15:13:53	755718	3042	687104	65572
17	7649	192139	12/31/2020	Ladakh	India	2021-04-02 15:13:53	9447	127	9132	188
18	7650	192140	12/31/2020	Lakshadweep	India	2021-04-02 15:13:53	0	0	0	0
19	7651	192160	12/31/2020	Madhya Pradesh	India	2021-04-02 15:13:53	240947	3595	227965	9387
20	7652	192166	12/31/2020	Maharashtra	India	2021-04-02 15:13:53	1928603	49463	1824934	54206
21	7653	192168	12/31/2020	Manipur	India	2021-04-02 15:13:53	28137	354	26601	1182
22	7654	192181	12/31/2020	Meghalaya	India	2021-04-02 15:13:53	13408	139	13085	184
23	7655	192195	12/31/2020	Mizoram	India	2021-04-02 15:13:53	4204	8	4091	105
24	7656	192207	12/31/2020	Nagaland	India	2021-04-02 15:13:53	11921	79	11624	218
25	7657	192251	12/31/2020	Odisha	India	2021-04-02 15:13:53	329306	1871	325103	2332
26	7658	192285	12/31/2020	Puducherry	India	2021-04-02 15:13:53	38096	633	37100	363
27	7659	192289	12/31/2020	Punjab	India	2021-04-02 15:13:53	166239	5331	157043	3865
28	7660	192299	12/31/2020	Rajasthan	India	2021-04-02 15:13:53	307554	2689	295030	9835
29	7661	192348	12/31/2020	Sikkim	India	2021-04-02 15:13:53	5877	127	5218	532
30	7662	192369	12/31/2020	Tamil Nadu	India	2021-04-02 15:13:53	817077	12109	796353	8615
31	7663	192373	12/31/2020	Telangana	India	2021-04-02 15:13:53	286354	1541	278839	5974
32	7664	192390	12/31/2020	Tripura	India	2021-04-02 15:13:53	33264	385	32751	128
33	7665	192405	12/31/2020	Unknown	India	2021-04-02 15:13:53	0	0	0	0
34	7666	192416	12/31/2020	Uttar Pradesh	India	2021-04-02 15:13:53	584966	8352	562459	14155
35	7667	192417	12/31/2020	Uttarakhand	India	2021-04-02 15:13:53	90616	1504	84149	4963
36	7668	192444	12/31/2020	West Bengal	India	2021-04-02 15:13:53	550893	9683	528829	12381

In [45]:
```
colors = ['rgb(2,58,88)','rgb(65,171,93)', 'rgb(127,0,0)']
labels = ["Active cases","Recovered","Deaths"]
values = Data_india_last.loc[0, ["Active_case","Recovered","Deaths"]]

fig = go.Figure(data=[go.Pie(labels=labels,
                             values=values)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```



Coronavirus in China

In [46]:
```
Data_China = covid_data [(covid_data['Country/Region'] == 'China') ].reset_index(drop=True)
Data_China.head()
```

Out[46]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	Active_case
0	1	01/22/2020	Anhui	China	1/22/2020 17:00	1	0	0	1
1	2	01/22/2020	Beijing	China	1/22/2020 17:00	14	0	0	14
2	3	01/22/2020	Chongqing	China	1/22/2020 17:00	6	0	0	6
3	4	01/22/2020	Fujian	China	1/22/2020 17:00	1	0	0	1
4	5	01/22/2020	Gansu	China	1/22/2020 17:00	0	0	0	0

Get last update in china

In [47]:
```
Data_china_last = Data_China[Data_China['ObservationDate'] == max(Data_China['ObservationDate'])].reset_index()
Data_china_last.head()
```

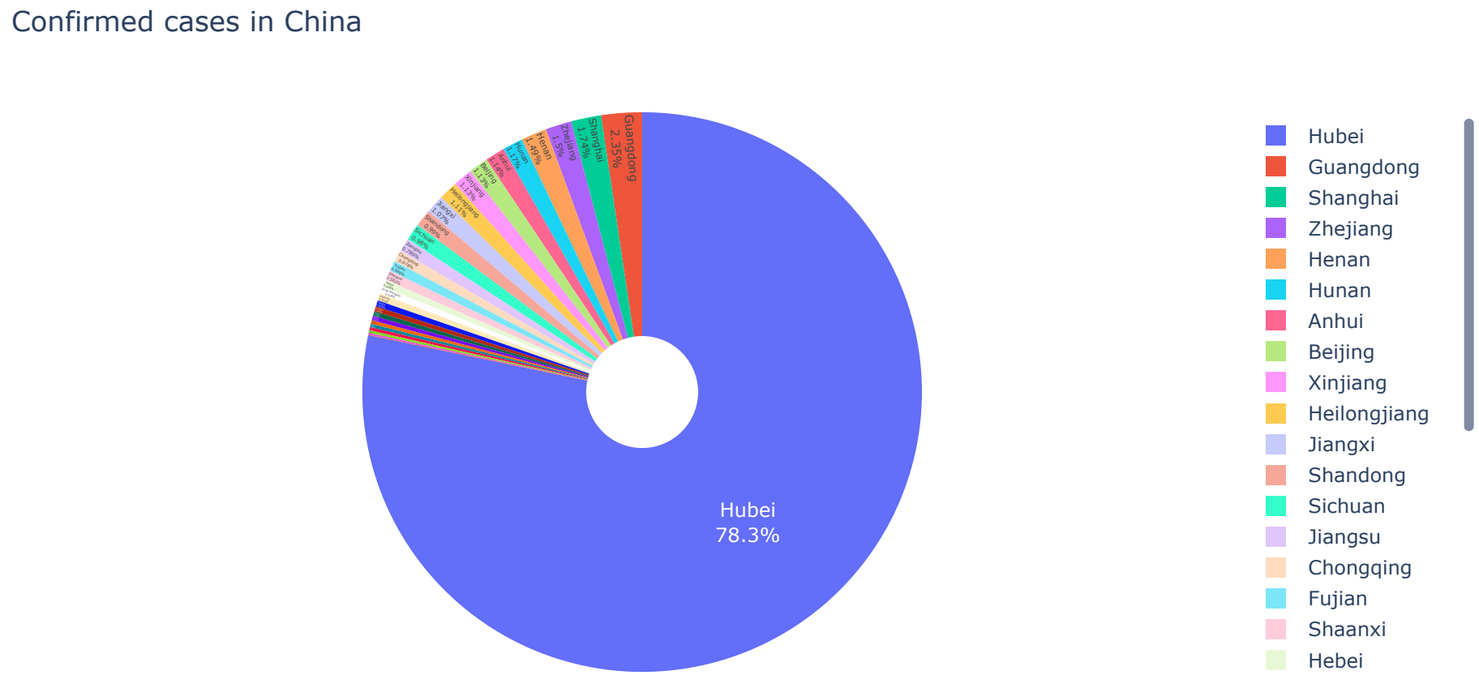
Out[47]:

	index	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	Active_case
0	11006	191897	12/31/2020	Anhui	China	2021-04-02 15:13:53	993	6	986	1
1	11007	191931	12/31/2020	Beijing	China	2021-04-02 15:13:53	987	9	944	34
2	11008	191981	12/31/2020	Chongqing	China	2021-04-02 15:13:53	590	6	584	0
3	11009	192023	12/31/2020	Fujian	China	2021-04-02 15:13:53	513	1	488	24
4	11010	192028	12/31/2020	Gansu	China	2021-04-02 15:13:53	182	2	180	0

Confirmed cases in every Province/State in china

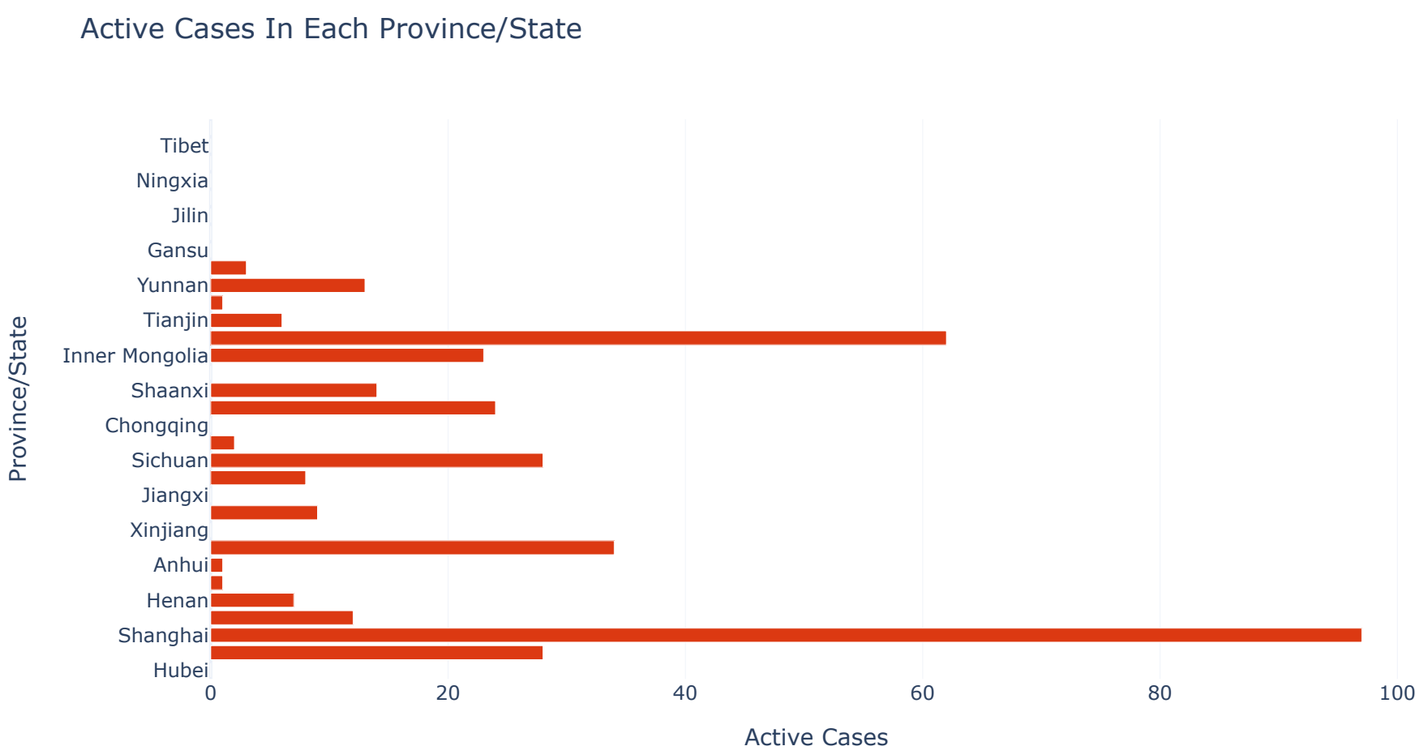
In [48]:
```
Data_china_per_state= Data_china_last.groupby(["Province/State"])["Confirmed","Active_case","Recovered","Deaths"].sum().reset_index().sort_values("Confirmed",ascending=False).
```

In [49]:
```
fig = px.pie(Data_china_per_state, values=Data_china_per_state['Confirmed'], names=Data_china_per_state['Province/State'],
             title='Confirmed cases in China',
            hole=.2)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



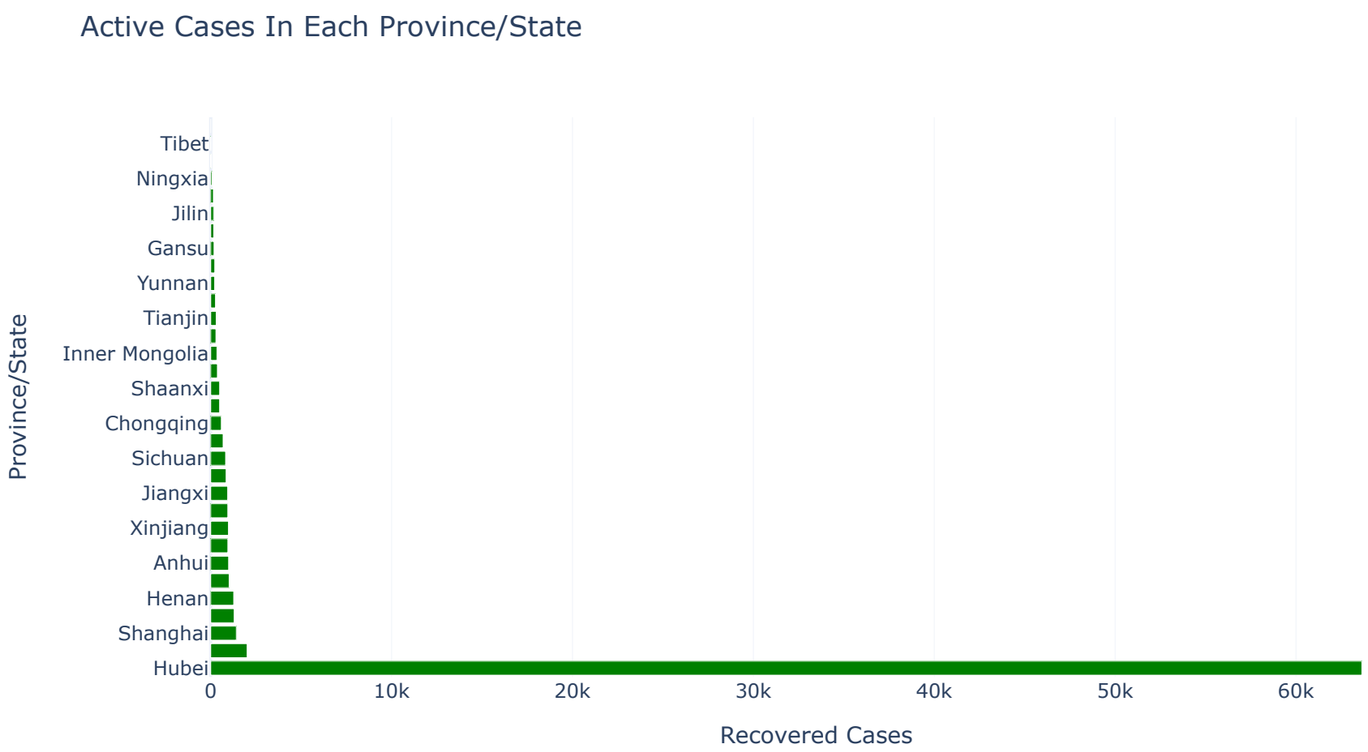
In [50]:

```
fig = go.Figure(go.Bar(
            x=Data_china_per_state['Active_case'],
            y=Data_china_per_state['Province/State'],
            orientation='h',
            marker_color='#DC3912',))
fig.update_layout(
    title='Active Cases In Each Province/State',
    template='plotly_white',
    xaxis_title="Active Cases",
    yaxis_title="Province/State",
)
fig.show()
```

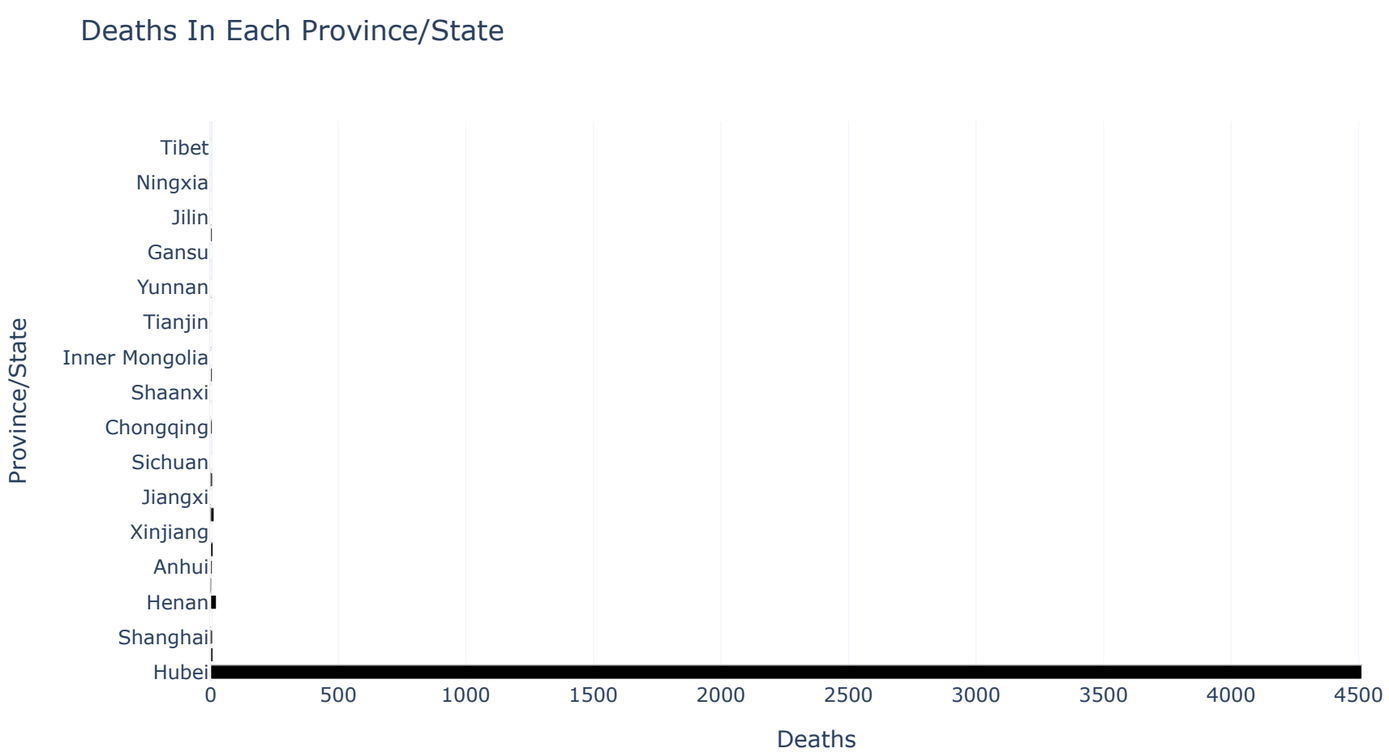


In [51]:

```
fig = go.Figure(go.Bar(
            x=Data_china_per_state['Recovered'],
            y=Data_china_per_state['Province/State'],
            orientation='h',
            marker_color='green',))
fig.update_layout(
    title='Active Cases In Each Province/State',
    template='plotly_white',
    xaxis_title="Recovered Cases",
    yaxis_title="Province/State",
)
fig.show()
```



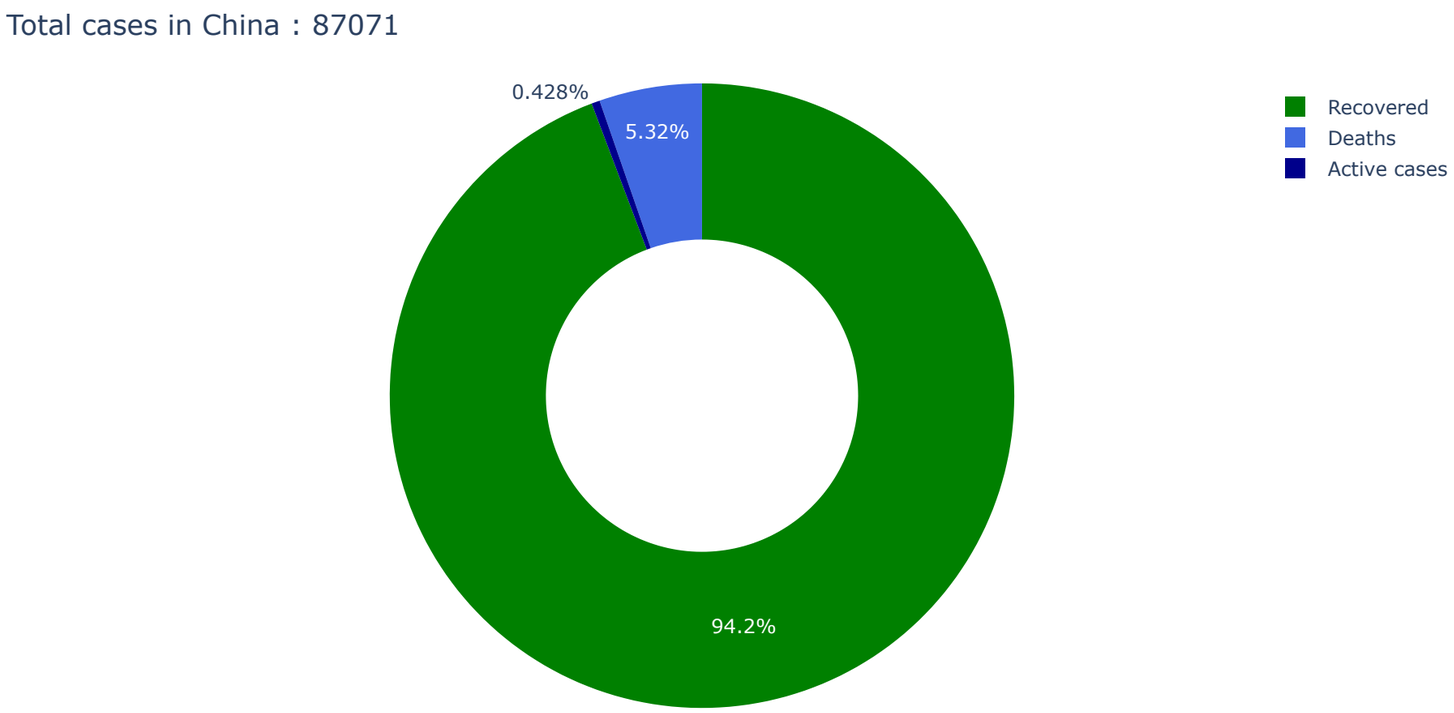
In [52]:
```
fig = go.Figure(go.Bar(
            x=Data_china_per_state['Deaths'],
            y=Data_china_per_state['Province/State'],
            orientation='h',
            marker_color='black',))
fig.update_layout(
    title='Deaths In Each Province/State',
    template='plotly_white',
    xaxis_title="Deaths",
    yaxis_title="Province/State",
)
fig.show()
```



Get total cases in China

In [53]:
```
Data_china_total= Data_china_last.groupby(["Country/Region"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)
```

In [54]:
```
labels = ["Active cases","Recovered","Deaths"]
values = Data_china_total.loc[0, ["Active_case","Recovered","Deaths"]]
df = px.data.tips()
fig = px.pie(Data_china_total, values=values, names=labels, color_discrete_sequence=['green','royalblue','darkblue'], hole=0.5)
fig.update_layout(
    title='Total cases in China : '+str(Data_china_total["Confirmed"][0]),
)
fig.show()
```



Coronavirus in United States

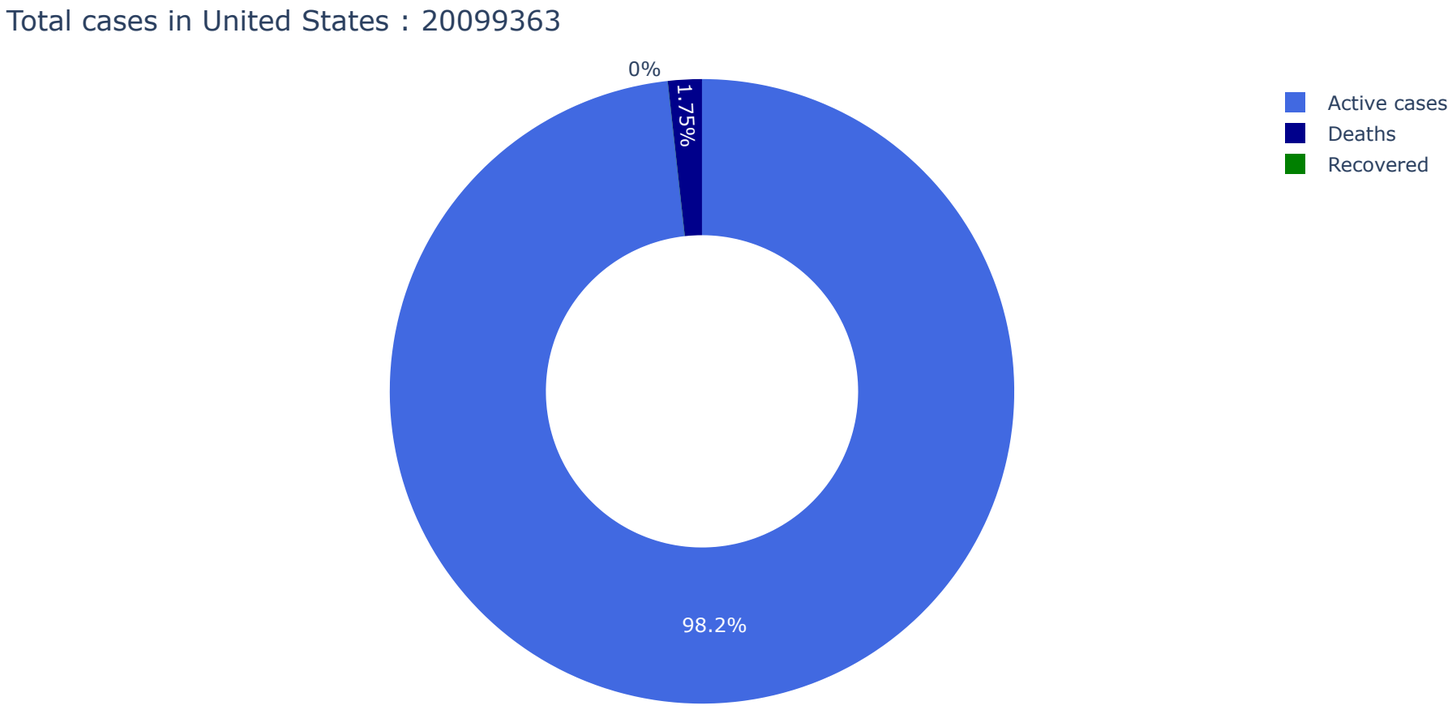
In [55]:
```
Data_US = covid_data [(covid_data['Country/Region'] == 'US') ].reset_index(drop=True)
```

Get last update in US

In [56]:
```
Data_us_last = Data_US[Data_US['ObservationDate'] == max(Data_US['ObservationDate'])].reset_index()
```

In [57]:
```
Data_us_total= Data_us_last.groupby(["Country/Region"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)
```

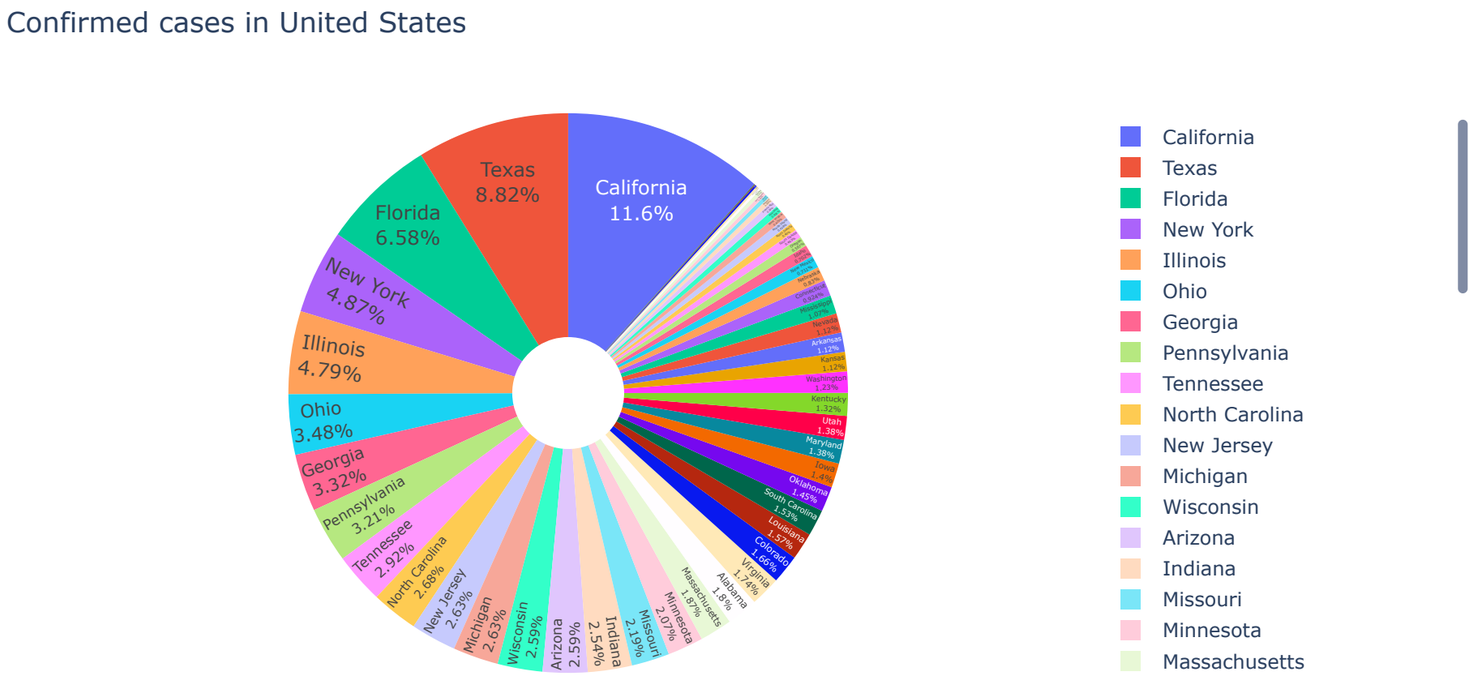
In [58]:
```
labels = ["Active cases","Recovered","Deaths"]
values = Data_us_total.loc[0, ["Active_case","Recovered","Deaths"]]
df = px.data.tips()
fig = px.pie(Data_us_total, values=values, names=labels, color_discrete_sequence=['royalblue','darkblue','green'], hole=0.5)
fig.update_layout(
    title='Total cases in United States : '+str(Data_us_total["Confirmed"][0]),
)
fig.show()
```



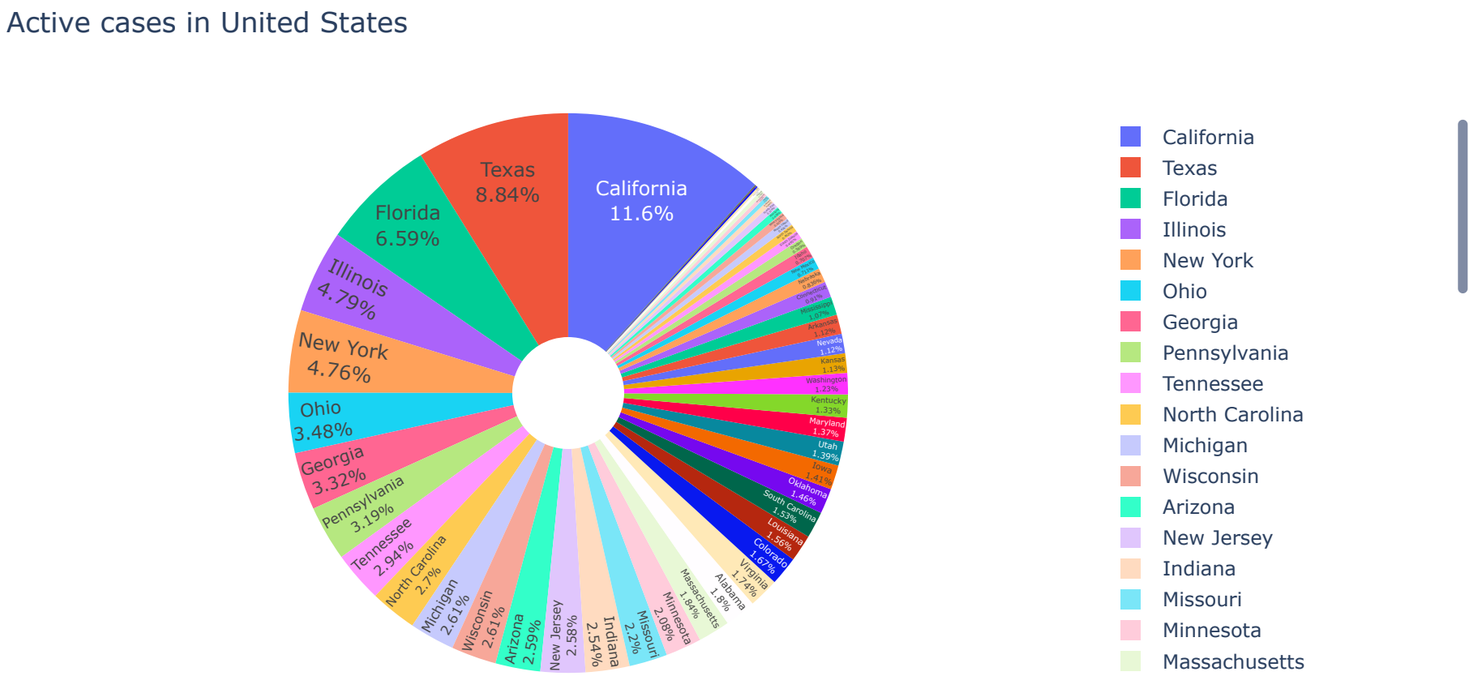
Cases in every Province/State in US

In [59]:
```
Data_us_per_state= Data_us_last.groupby(["Province/State"])["Confirmed","Active_case","Deaths"].sum().reset_index().sort_values("Confirmed",ascending=False).reset_index(drop=T
```

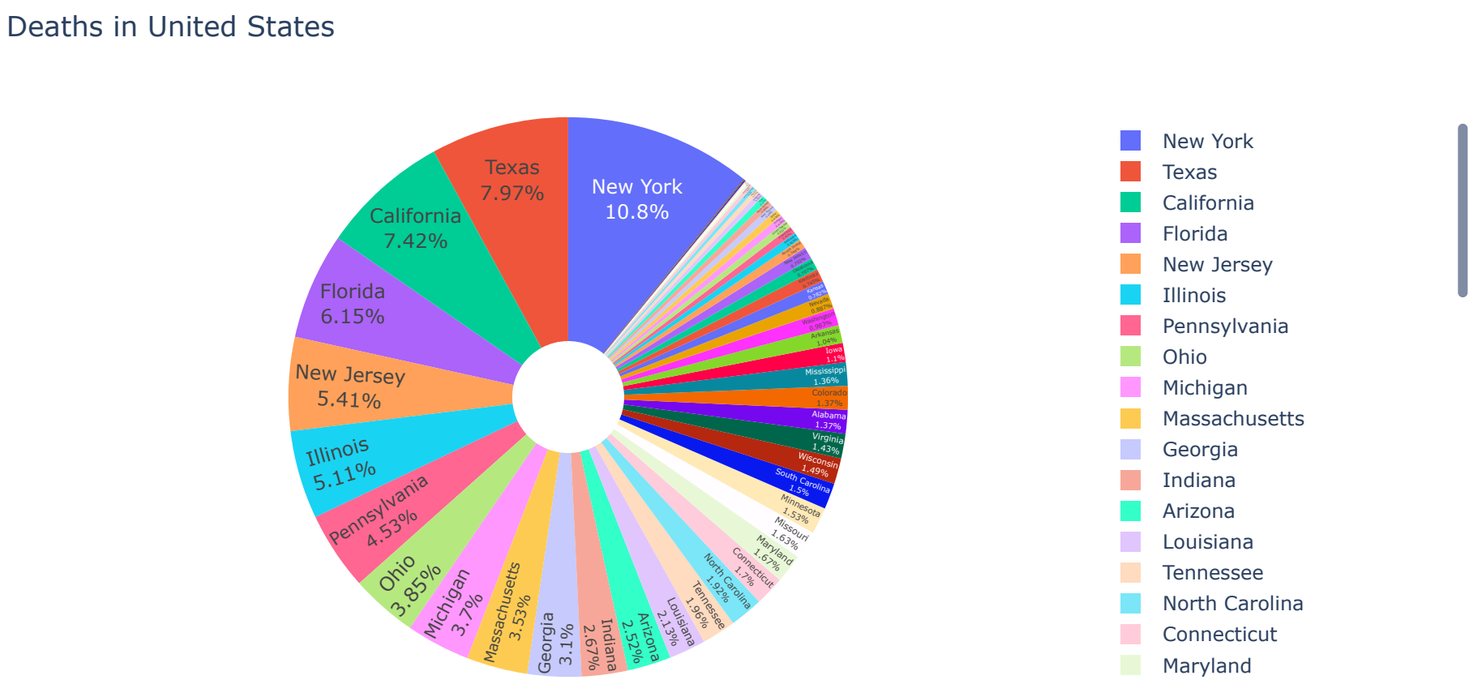
In [60]:
```
fig = px.pie(Data_us_per_state, values=Data_us_per_state['Confirmed'], names=Data_us_per_state['Province/State'],
             title='Confirmed cases in United States',
             hole=.2)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



In [61]:
```
fig = px.pie(Data_us_per_state, values=Data_us_per_state['Active_case'], names=Data_us_per_state['Province/State'],
             title='Active cases in United States',
             hole=.2)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



In [62]:
```
fig = px.pie(Data_us_per_state, values=Data_us_per_state['Deaths'], names=Data_us_per_state['Province/State'],
             title='Deaths in United States',
            hole=.2)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



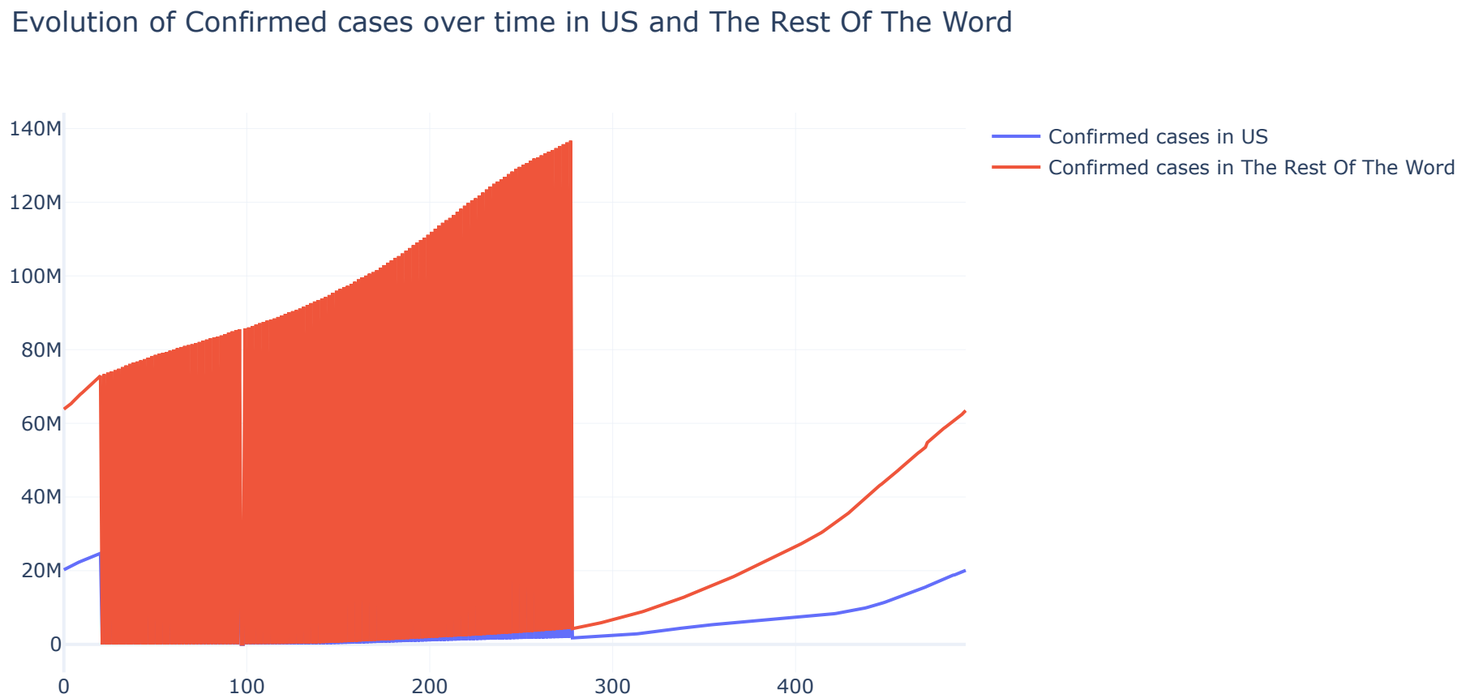
US X The rest of the word

In [63]:
```
Data_US_op= Data_US.groupby(["ObservationDate","Country/Region"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)
```

In [64]:
```
Data_Word = covid_data [(covid_data['Country/Region'] != 'US') ].reset_index(drop=True)
Data_WORD_last = Data_Word[Data_Word['ObservationDate'] == max(Data_Word['ObservationDate'])].reset_index()
Data_us_total= Data_us_last.groupby(["Country/Region"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)
Data_word_total= Data_WORD_last.groupby(["ObservationDate"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)

Data_Word_op= Data_Word.groupby(["ObservationDate"])["Confirmed","Deaths","Recovered","Active_case"].sum().reset_index().reset_index(drop=True)
```

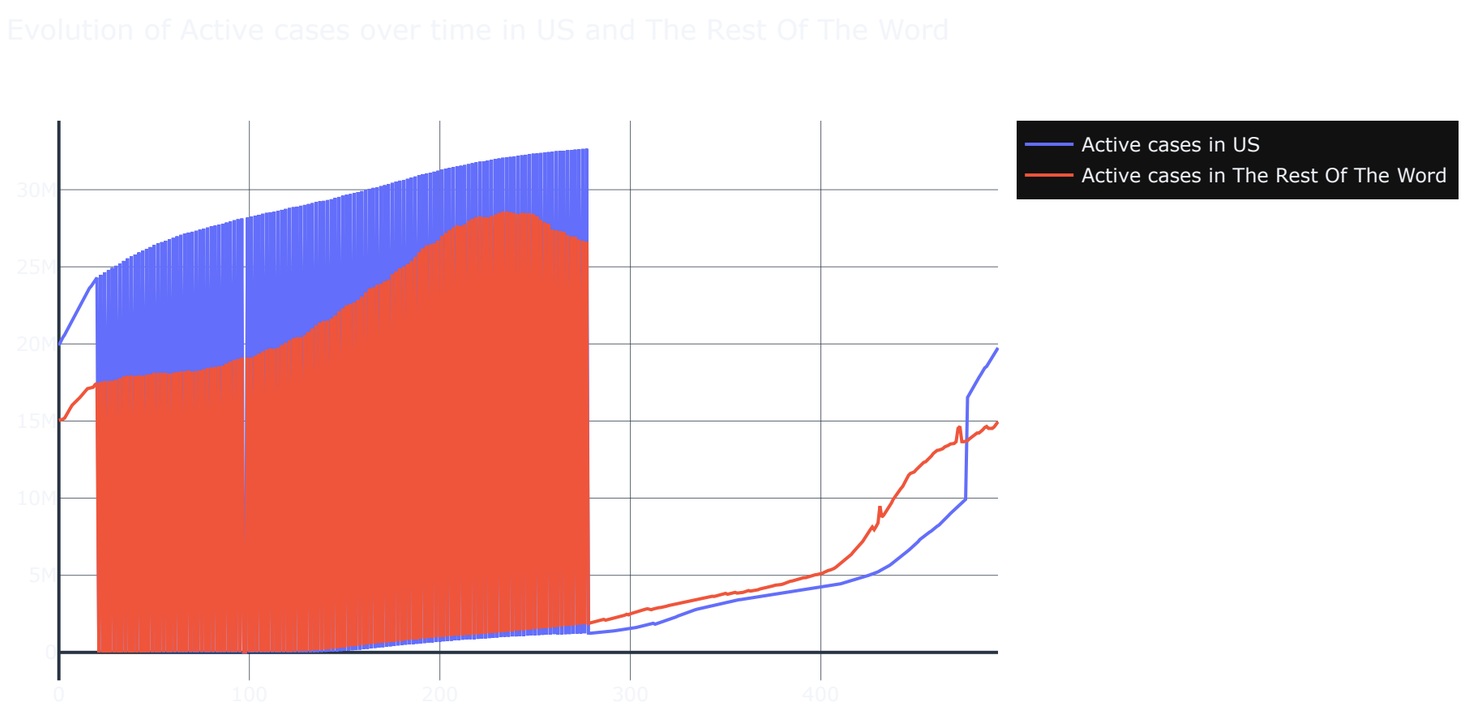
In [65]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=Data_US_op.index, y=Data_US_op['Confirmed'],
                    mode='lines',
                    name='Confirmed cases in US'))


fig.add_trace(go.Scatter(x=Data_Word_op.index, y=Data_Word_op['Confirmed'],
                    mode='lines',
                    name='Confirmed cases in The Rest Of The Word'))

fig.update_layout(
    title='Evolution of Confirmed cases over time in US and The Rest Of The Word',
        template='plotly_white'

)

fig.show()
```



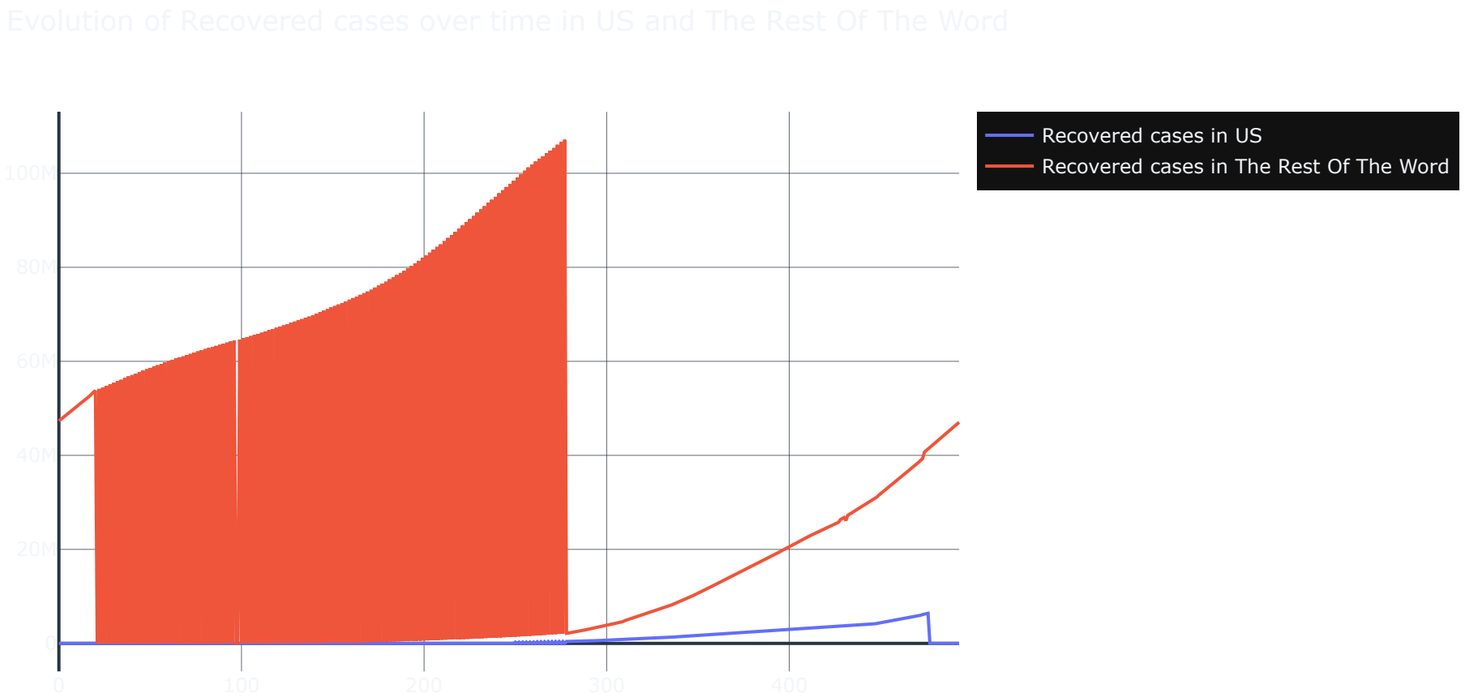
In [66]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=Data_US_op.index, y=Data_US_op['Active_case'],
                    mode='lines',
                    name='Active cases in US'))


fig.add_trace(go.Scatter(x=Data_Word_op.index, y=Data_Word_op['Active_case'],
                    mode='lines',
                    name='Active cases in The Rest Of The Word'))

fig.update_layout(
    title='Evolution of Active cases over time in US and The Rest Of The Word',
        template='plotly_dark'

)

fig.show()
```

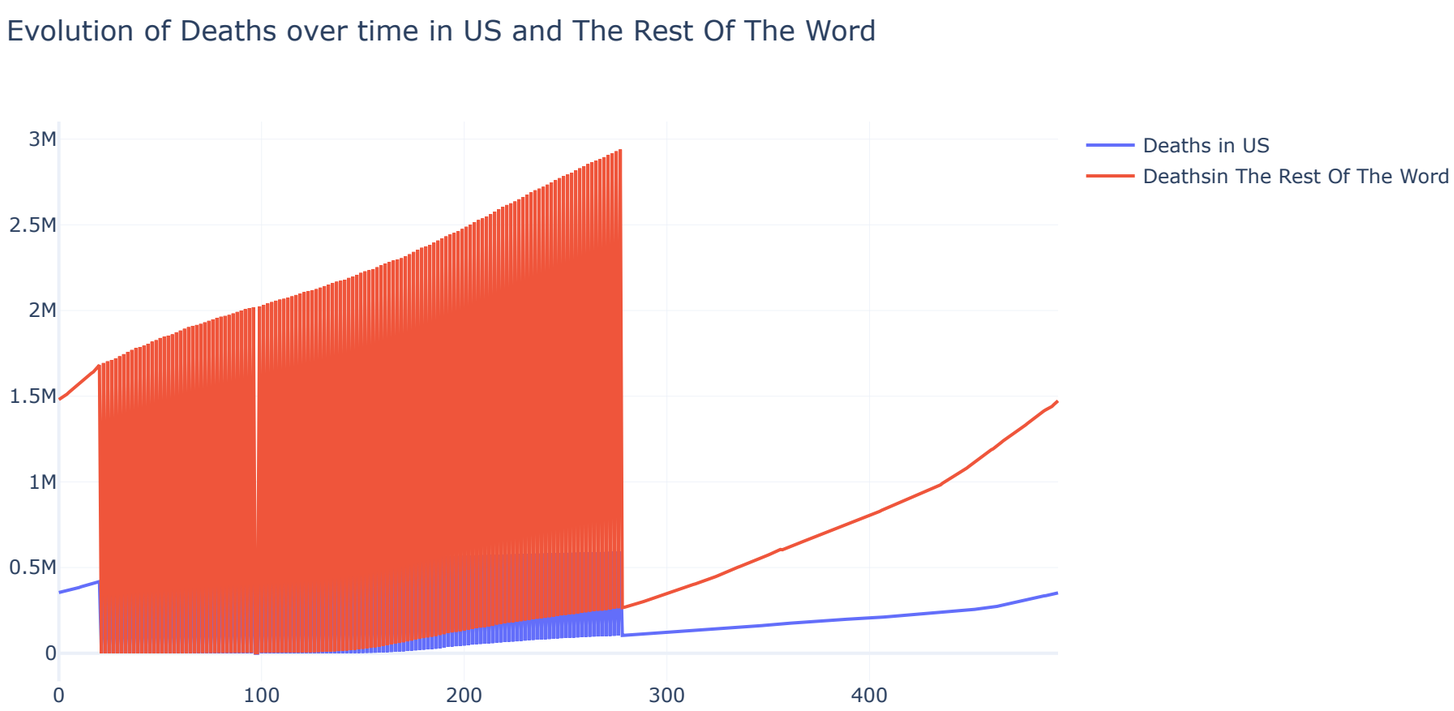


In [67]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=Data_US_op.index, y=Data_US_op['Recovered'],
                    mode='lines',
                    name='Recovered cases in US'))


fig.add_trace(go.Scatter(x=Data_Word_op.index, y=Data_Word_op['Recovered'],
                    mode='lines',
                    name='Recovered cases in The Rest Of The Word'))

fig.update_layout(
    title='Evolution of Recovered cases over time in US and The Rest Of The Word',
        template='plotly_dark'

)

fig.show()
```



In [68]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=Data_US_op.index, y=Data_US_op['Deaths'],
                    mode='lines',
                    name='Deaths in US'))


fig.add_trace(go.Scatter(x=Data_Word_op.index, y=Data_Word_op['Deaths'],
                    mode='lines',
                    name='Deathsin The Rest Of The Word'))
fig.update_layout(
    title='Evolution of Deaths over time in US and The Rest Of The Word',
        template='plotly_white'

)

fig.show()
```



In [ ]:

In [ ]: