import modules for data visualizations

In [66]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.animation as animation
from IPython.display import HTML
import plotly.graph_objects as go
import seaborn as sns
%matplotlib inline
```

Dataset Detail Information

The world is disturbed with covid diseases. So Our data Scienetist want to figure out some explanation about covid 19 and also find their way. So I Choose Covid 19 for my 1st Assignment for data visualization

Our dataset contains at first 6 columns

- Id
- Province_State
- Country_Region
- Date
- ConfirmedCases
- Fatalities

There is the information before April 11 2020

**** I create the index of Id attribute for indexing ***

In [67]:
```
df = pd.read_csv('train.csv',index_col = 'Id')
```

In [68]:
```
df.head(10)
```

Out[68]:

Id	Province_State	Country_Region	Date	ConfirmedCases	Fatalities
1	NaN	Afghanistan	2020-01-22	0.0	0.0
2	NaN	Afghanistan	2020-01-23	0.0	0.0
3	NaN	Afghanistan	2020-01-24	0.0	0.0
4	NaN	Afghanistan	2020-01-25	0.0	0.0
5	NaN	Afghanistan	2020-01-26	0.0	0.0
6	NaN	Afghanistan	2020-01-27	0.0	0.0
7	NaN	Afghanistan	2020-01-28	0.0	0.0
8	NaN	Afghanistan	2020-01-29	0.0	0.0
9	NaN	Afghanistan	2020-01-30	0.0	0.0
10	NaN	Afghanistan	2020-01-31	0.0	0.0

So the column name is so large . I want to redundate the column name. So i use **rename** method to easily write the columns name.

In [70]:
```
df.rename(columns={"Country_Region": "country", "Province_State": "province"}, inplace=True, errors="raise")
df
```

Out[70]:

Id	province	country	Date	ConfirmedCases	Fatalities
1	NaN	Afghanistan	2020-01-22	0.0	0.0
2	NaN	Afghanistan	2020-01-23	0.0	0.0
3	NaN	Afghanistan	2020-01-24	0.0	0.0
4	NaN	Afghanistan	2020-01-25	0.0	0.0
5	NaN	Afghanistan	2020-01-26	0.0	0.0
...	...	...	...	...	...
35645	NaN	Zimbabwe	2020-04-07	11.0	2.0
35646	NaN	Zimbabwe	2020-04-08	11.0	3.0
35647	NaN	Zimbabwe	2020-04-09	11.0	3.0
35648	NaN	Zimbabwe	2020-04-10	13.0	3.0
35649	NaN	Zimbabwe	2020-04-11	14.0	3.0

25353 rows × 5 columns

use of data manipulations with numpy and pandas

I use groupby method mainly to get all the country's covid 19 confirm cases and death cases

and also i use the sort function to sort desecnding according to confirmed cases

In [5]:
```
df_groupby_country = df.groupby("country")["ConfirmedCases","Fatalities"].sum().sort_values("ConfirmedCases",ascending = False).reset_index()
df_groupby_country
```

Out[5]:

	country	ConfirmedCases	Fatalities
0	US	5135445.0	147545.0
1	China	5096274.0	182450.0
2	Italy	2661341.0	297444.0
3	Spain	2237252.0	200412.0
4	Germany	1728391.0	24491.0
...	...	...	...
179	Papua New Guinea	29.0	0.0
180	Western Sahara	28.0	0.0
181	Sao Tome and Principe	24.0	0.0
182	Timor-Leste	23.0	0.0
183	South Sudan	17.0	0.0

184 rows × 3 columns

Add a new column to find the recoveries in here

just subtract ConfirmCases column to Fatalities column get recoveries

In [6]:
```
df_groupby_country["Recoveries"] = df_groupby_country["ConfirmedCases"] - df_groupby_country["Fatalities"]
```

In [7]:
```
df_groupby_country.head(10)
```

Out[7]:

	country	ConfirmedCases	Fatalities	Recoveries
0	US	5135445.0	147545.0	4987900.0
1	China	5096274.0	182450.0	4913824.0
2	Italy	2661341.0	297444.0	2363897.0
3	Spain	2237252.0	200412.0	2036840.0
4	Germany	1728391.0	24491.0	1703900.0
5	France	1476739.0	122427.0	1354312.0
6	Iran	1184893.0	75219.0	1109674.0
7	United Kingdom	749434.0	72805.0	676629.0
8	Turkey	408544.0	8310.0	400234.0
9	Switzerland	393842.0	11029.0	382813.0

Add two new column for further data analysis

Though We have less column so want to increase more , that's why we add two new column **survival rate** and **fatalities rate**

In [8]:
```
df_groupby_country['Survival Rate'] = round(df_groupby_country['Recoveries']/df_groupby_country['ConfirmedCases']*100,2)
df_groupby_country['Fatalities Rate'] = round(df_groupby_country['Fatalities']/df_groupby_country['ConfirmedCases']*100,2)
df_groupby_country.head(10)
```

Out[8]:

	country	ConfirmedCases	Fatalities	Recoveries	Survival Rate	Fatalities Rate
0	US	5135445.0	147545.0	4987900.0	97.13	2.87
1	China	5096274.0	182450.0	4913824.0	96.42	3.58
2	Italy	2661341.0	297444.0	2363897.0	88.82	11.18
3	Spain	2237252.0	200412.0	2036840.0	91.04	8.96
4	Germany	1728391.0	24491.0	1703900.0	98.58	1.42
5	France	1476739.0	122427.0	1354312.0	91.71	8.29
6	Iran	1184893.0	75219.0	1109674.0	93.65	6.35
7	United Kingdom	749434.0	72805.0	676629.0	90.29	9.71
8	Turkey	408544.0	8310.0	400234.0	97.97	2.03
9	Switzerland	393842.0	11029.0	382813.0	97.20	2.80

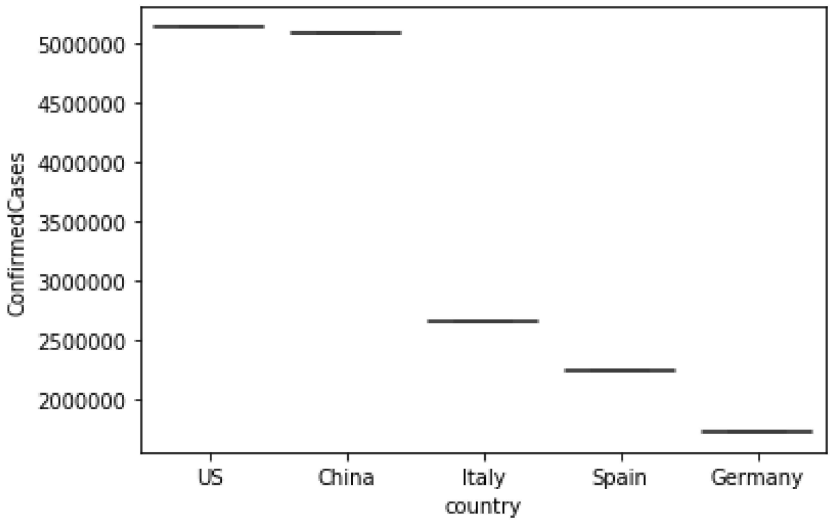
Data Visualization part

So our data visualization part is begin here

at first we plot a box plot Confirmed cases against country

In [9]:
```
sns.boxplot(x=df_groupby_country.head(5)["country"], y=df_groupby_country.head(5)["ConfirmedCases"])
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2090e0b8288>
```



Box plots are useful as they provide a visual summary of the data enabling researchers to quickly identify mean values, the dispersion of the data set, and signs of skewness.

So we plot our data to a box to analyze what's going on there .

The summary is below there. We got US max Confirmed Cases and also max fatalitiesnumbers. South Sudan has low in this cases.

and also we got the every column mean value, median, 1st qurtile value and 3rd qurtile value

In [74]:
```
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

In [11]:
```
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

In [12]:
```
df_groupby_country.iplot(kind='box')
```

In [13]:
```
sns.set_palette('Set2')
```

Describe the data with describe method

In [14]:
```
df_groupby_country.describe()
```

Out[14]:

	ConfirmedCases	Fatalities	Recoveries	Survival Rate	Fatalities Rate
count	1.840000e+02	184.000000	1.840000e+02	184.000000	184.000000
mean	1.383150e+05	6943.532609	1.313715e+05	96.569891	3.430109
std	6.141391e+05	33315.452596	5.863846e+05	4.398191	4.398191
min	1.700000e+01	0.000000	1.700000e+01	76.190000	0.000000
25%	3.727500e+02	11.500000	3.702500e+02	95.607500	0.487500
50%	5.351000e+03	76.000000	5.209000e+03	98.010000	1.990000
75%	2.943100e+04	623.250000	2.913725e+04	99.512500	4.392500
max	5.135445e+06	297444.000000	4.987900e+06	100.000000	23.810000

Playing with data manipulation to see how different parametes works there

In [15]:
```
df_groupby_country.describe(include=['O'])
```

Out[15]:

	country
count	184
unique	184
top	Fiji
freq	1

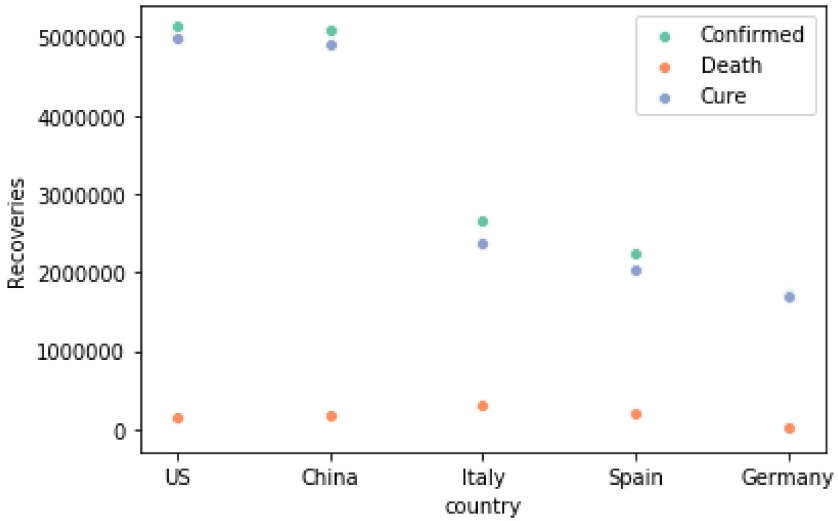
In [16]:
```
df_groupby_country_10 = df_groupby_country.head(5)
```

SeaBorn Scatter Plot

multiple columns plotting with one axis . the columns here confirmedcases , Fatalities, Recoveries

In [17]:
```
p = sns.scatterplot(data=df_groupby_country_10, x = df_groupby_country_10['country'], y = df_groupby_country_10['ConfirmedCases'], label = 'Confirmed')
p = sns.scatterplot(data=df_groupby_country_10, x = df_groupby_country_10['country'], y = df_groupby_country_10['Fatalities'], label = 'Death')
p = sns.scatterplot(data=df_groupby_country_10, x = df_groupby_country_10['country'], y = df_groupby_country_10['Recoveries'], label = 'Cure')
plt.legend()
```

Out[17]: <matplotlib.legend.Legend at 0x20910758408>



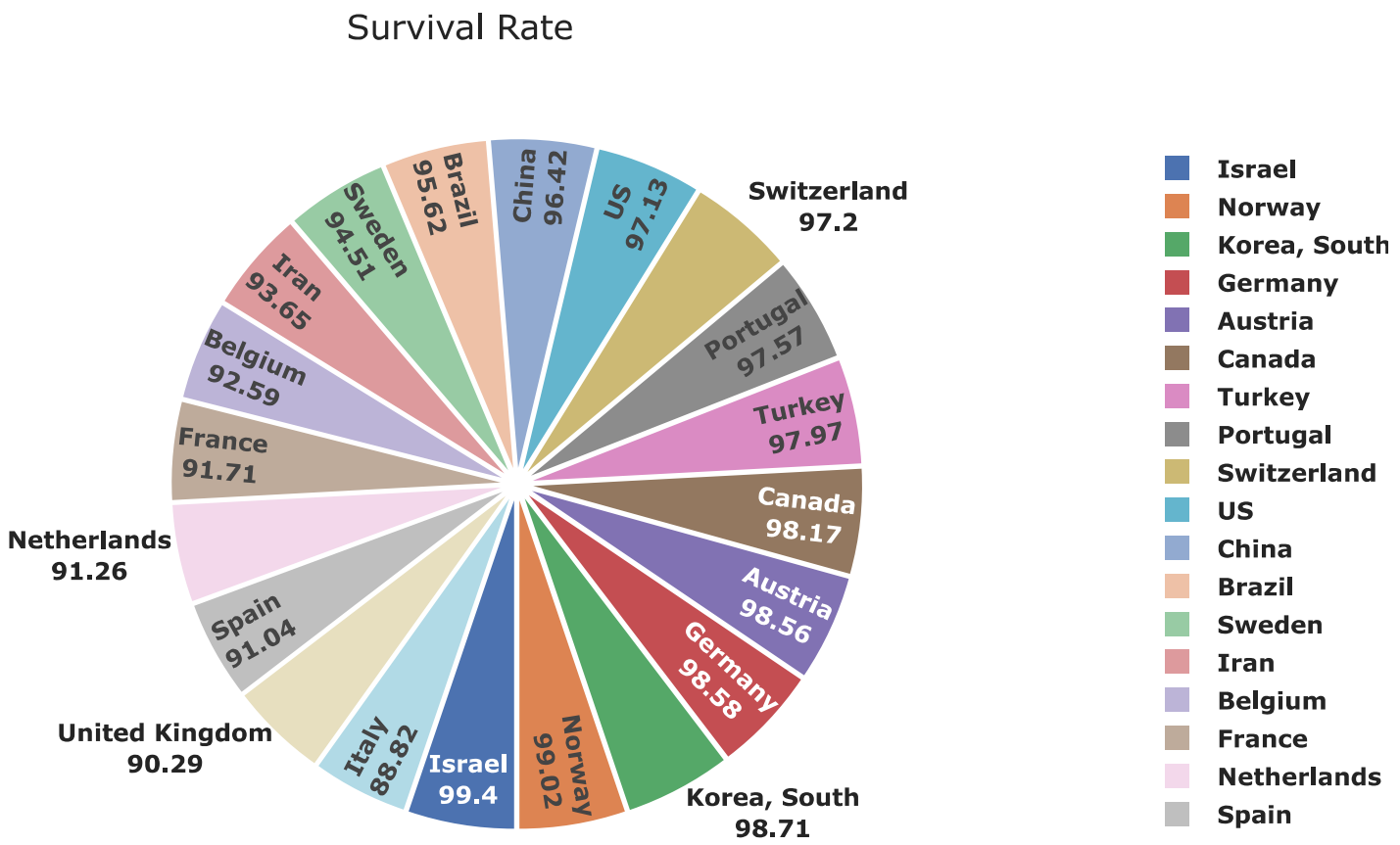
plotly Pie Chart

A pie chart shows how a total amount is divided between levels of a categorical variable as a circle divided into radial slices. Each categorical value corresponds with a single slice of the circle, and the size of each slice (both in area and arc length) indicates what proportion of the whole each category level takes.

Use plotly becasue of better good visualization with graphics

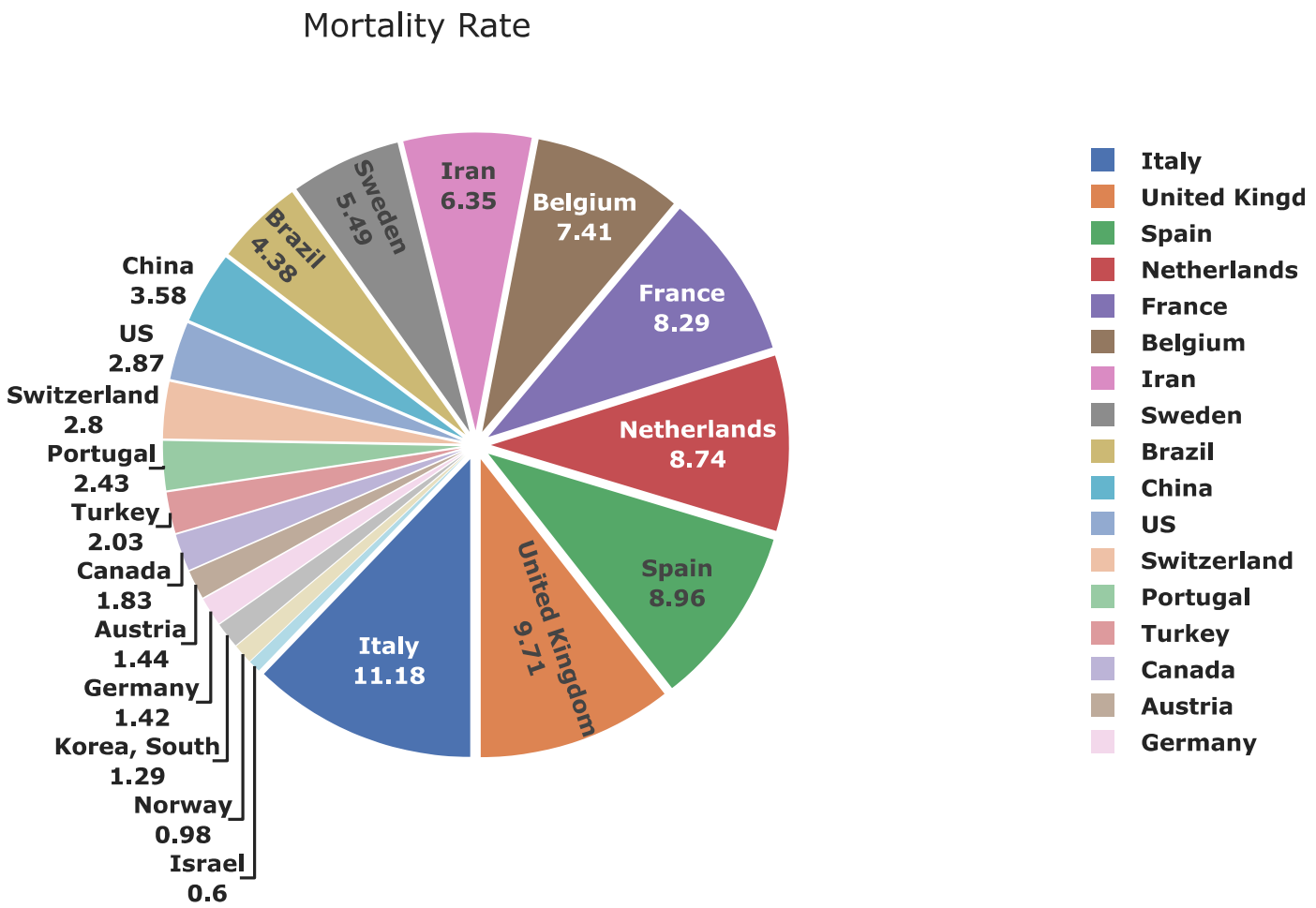
1st analysis of pie chart is Survival Rate . We want to see which country has most survival rate. So we plot our 1st 20 countries to shows the survival rates.

In [78]:
```
fig = px.pie(df_groupby_country.head(20).sort_values("Survival Rate",ascending = False),values="Survival Rate",names="countr
y",title="Survival Rate",template="seaborn")
fig.update_traces(rotation=180, pull=0.05, textinfo='value+label')
fig.show()
```



2nd analysis of pie chart is Mortality Rate . We want to see which country has most survival rate. So we plot our 1st 20 countries to shows the survival rates.

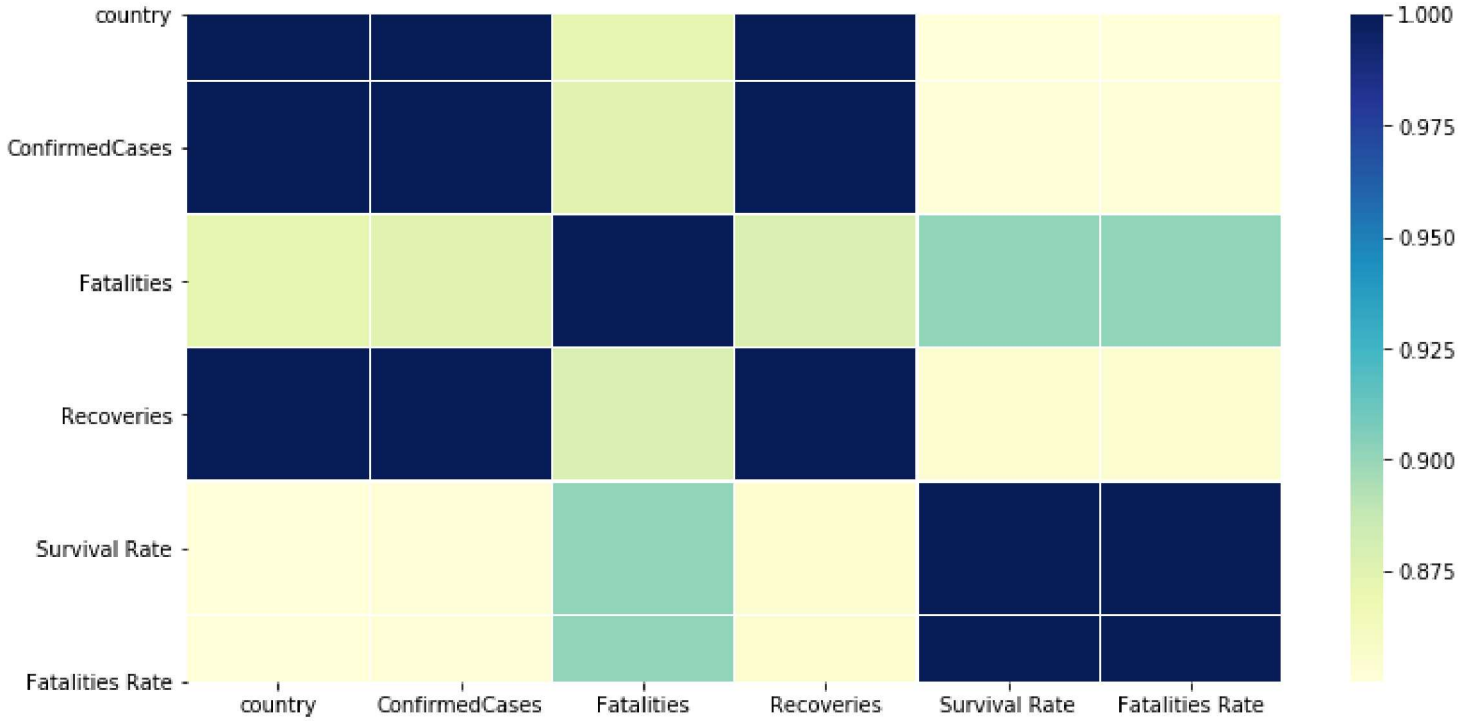
In [79]:
```
fig = px.pie(df_groupby_country.head(20).sort_values("Fatalities Rate",ascending = False),values="Fatalities Rate",names="country",title="Mortality Rate",template="seaborn")
fig.update_traces(rotation=180, pull=0.05, textinfo='value+label')
fig.show()
```



Co-relation between columns of covid 19

Correlation can tell you just how much of our data is strongly co- related with each others. So we use Heat Map to show our analysis

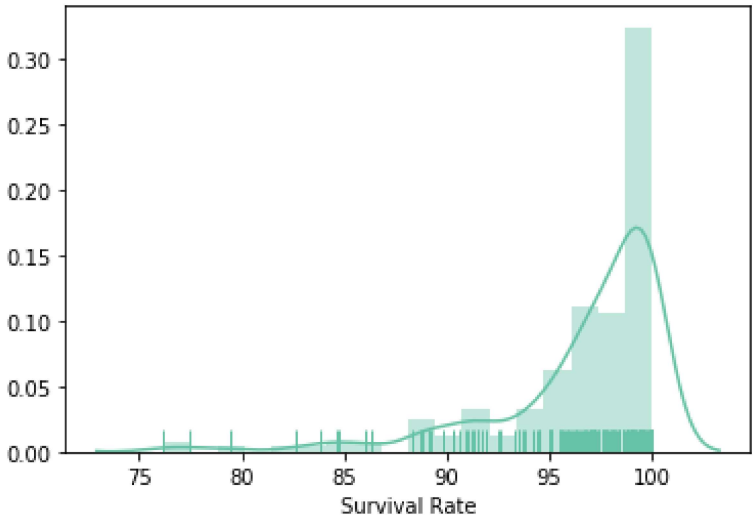
In [20]:
```
plt.figure(figsize=(12, 6))
corr = df_groupby_country.apply(lambda x: pd.factorize(x)[0]).corr()
ax = sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,
                 linewidths=.2, cmap="YlGnBu")
```



Sea Born histogram Plot

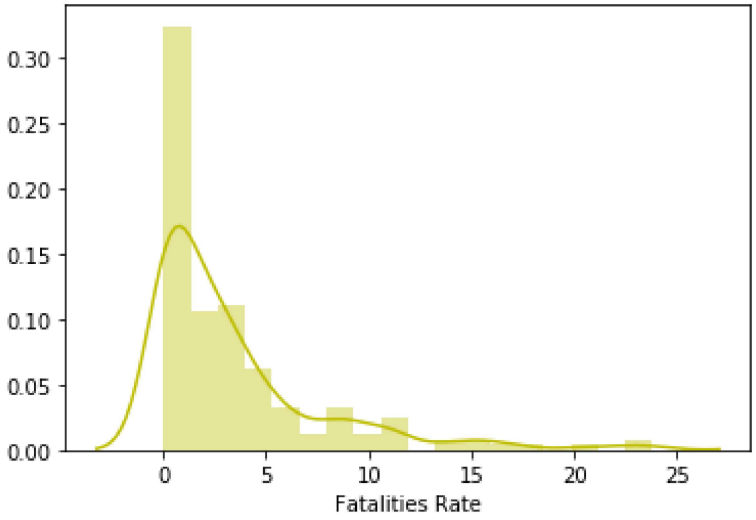
We plot the histogram of Survival rate here, this can be shown in all kinds of variations.

In [21]:
```
sns.distplot(df_groupby_country['Survival Rate'], rug=True)

plt.show()
```



We plot the histogram of Fatalities rate here, this can be shown in all kinds of variations.

In [22]:
```
sns.distplot(df_groupby_country['Fatalities Rate'], kde=True, color = 'y')

plt.show()
```

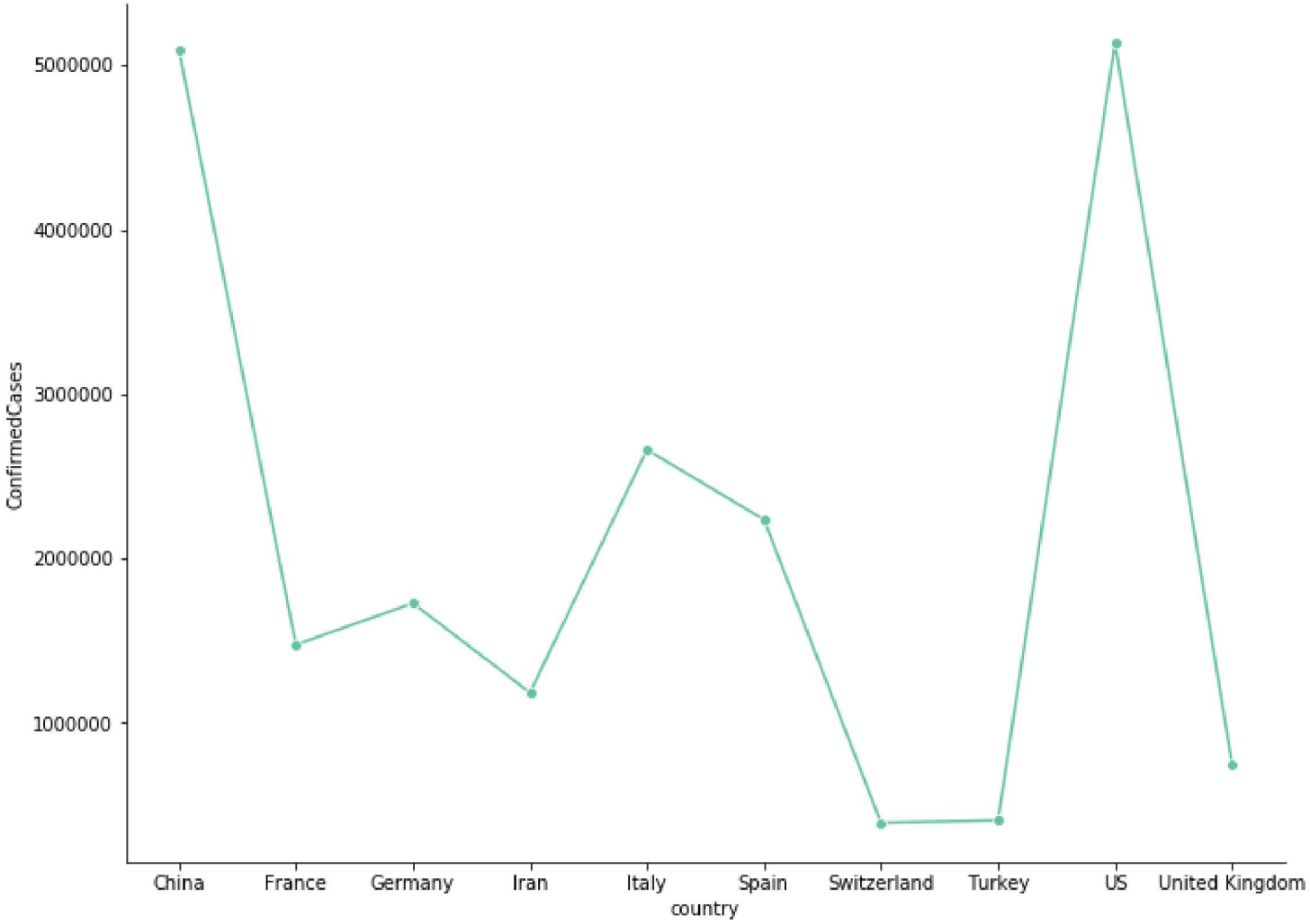


SeaBorn RelPlot Uses

We use the kind=line to see the scatter plot with a line Country versus Confirmed Cases

In [23]:
```
sns.relplot(x='country', y='ConfirmedCases',kind = 'line', data=df_groupby_country.head(10), height = 7.2, aspect = 1.4, mar
ker = 'o')
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x20910e9d6c8>



Uses np.log to plot the data with best visualization

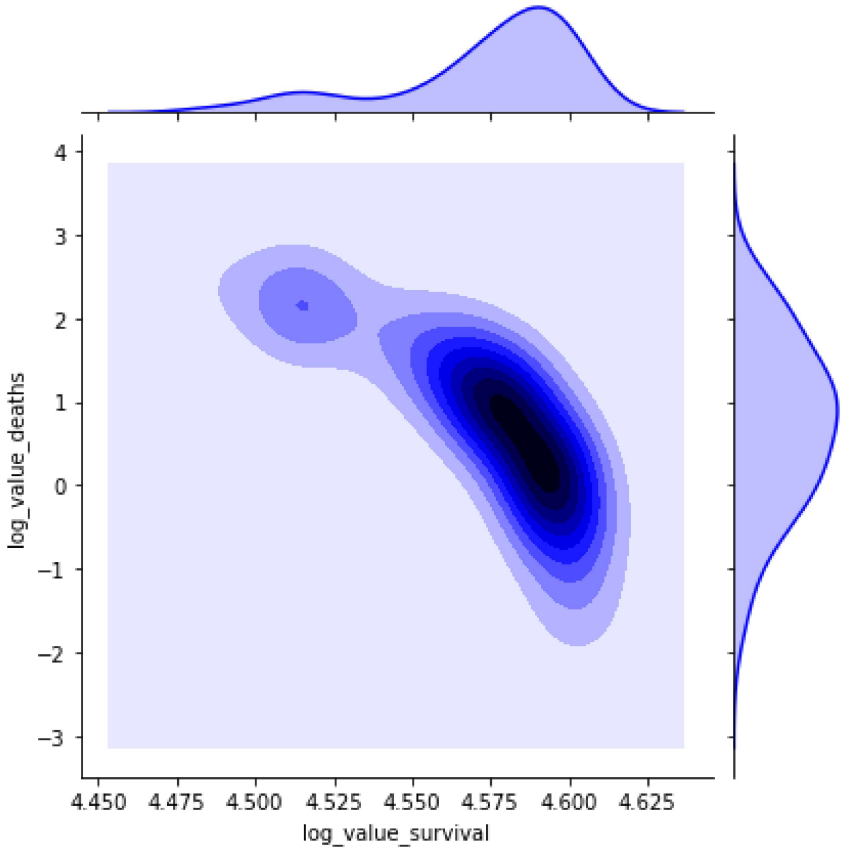
In [24]:
```
df_groupby_country['log_value_survival'] = np.log(df_groupby_country['Survival Rate'])
df_groupby_country['log_value_deaths'] = np.log(df_groupby_country['Fatalities Rate'])
```

```
C:\Users\inzim\Anaconda3\lib\site-packages\pandas\core\series.py:853: RuntimeWarning:

divide by zero encountered in log
```

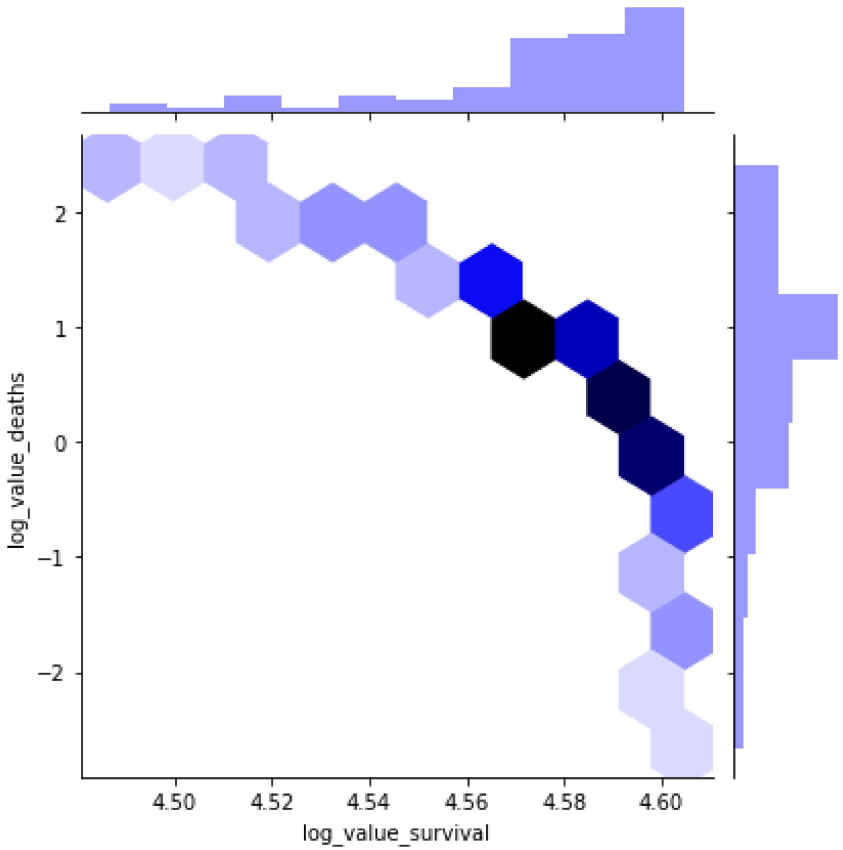
SeaBorn joint Plot

A jointplot is seaborn's method of displaying a bivariate relationship at the same time as a univariate profile.

In [25]:
```
sns.jointplot(x='log_value_survival',y='log_value_deaths',data=df_groupby_country.head(50), kind='kde', color='b')
plt.show()
```

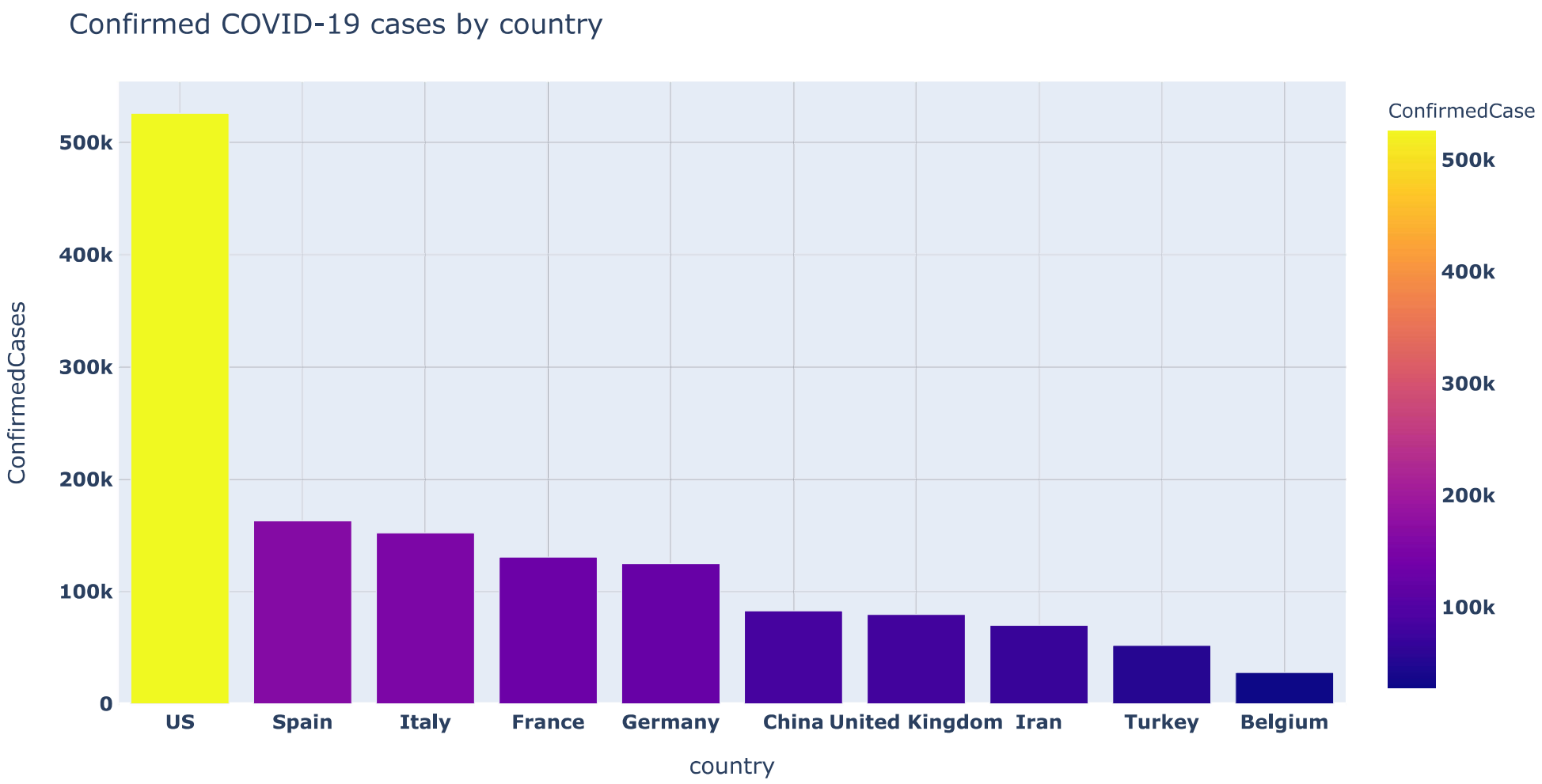


In [29]:
```
sns.jointplot(x='log_value_survival',y='log_value_deaths',data=df_groupby_country.head(80), kind='hex', color='b')
plt.show()
```



Top Confirmed Cases Against the Country

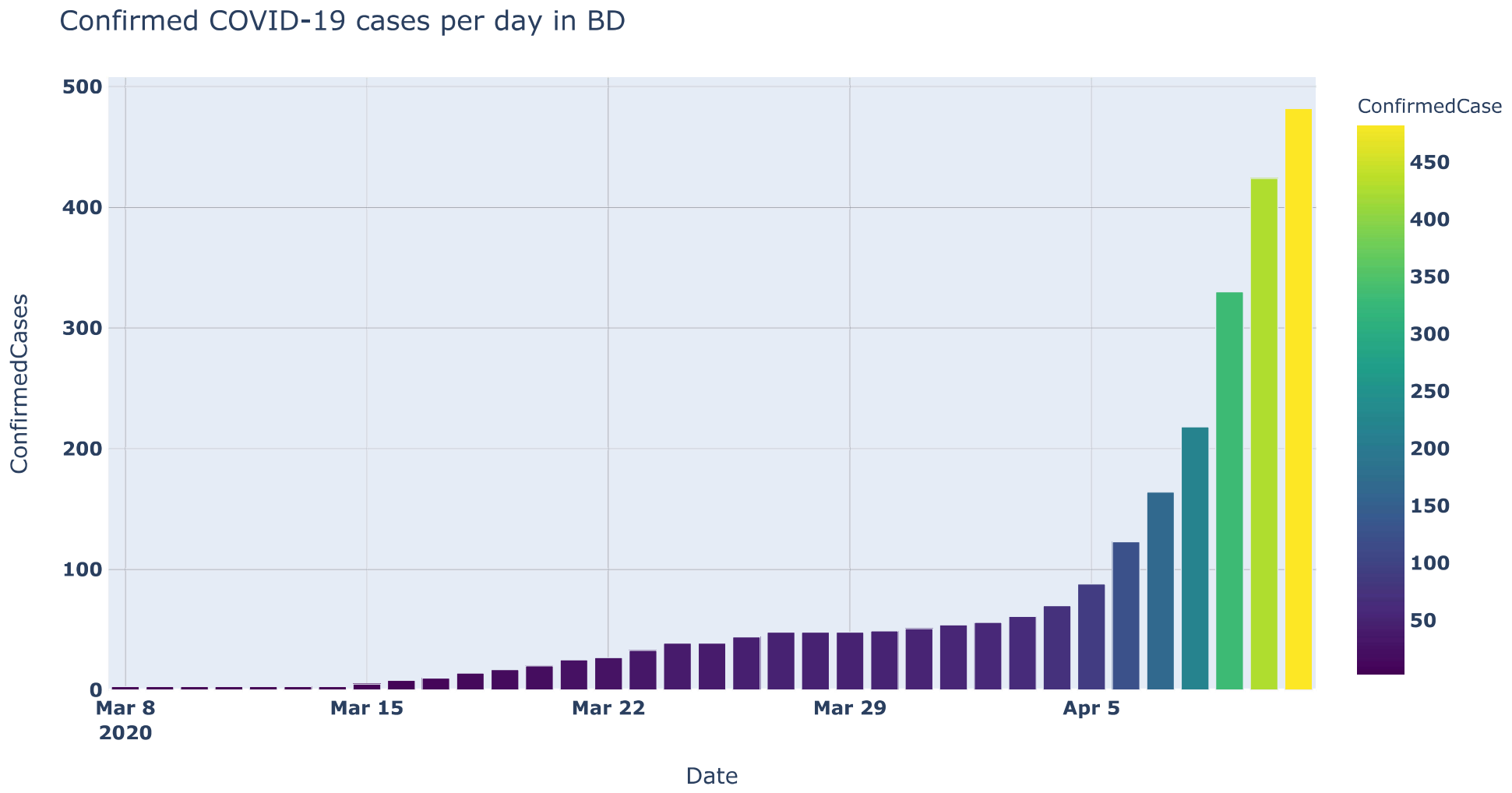
In [30]:
```
df_groupby = df.fillna('NA').groupby(['country','province','Date'])['ConfirmedCases'].sum() \
                            .groupby(['country','province']).max().sort_values() \
                            .groupby(['country']).sum().sort_values(ascending = False)

top10 = pd.DataFrame(df_groupby).head(10)
fig = px.bar(top10, x=top10.index, y='ConfirmedCases', labels={'x':'Country'},
             color="ConfirmedCases", color_continuous_scale=px.colors.sequential.Plasma)
fig.update_layout(title_text='Confirmed COVID-19 cases by country')
fig.show()
```



In [33]:
```
df_by_date_country = pd.DataFrame(df.fillna('NA').groupby(['country','Date'])['ConfirmedCases'].sum().sort_values().reset_index())
```

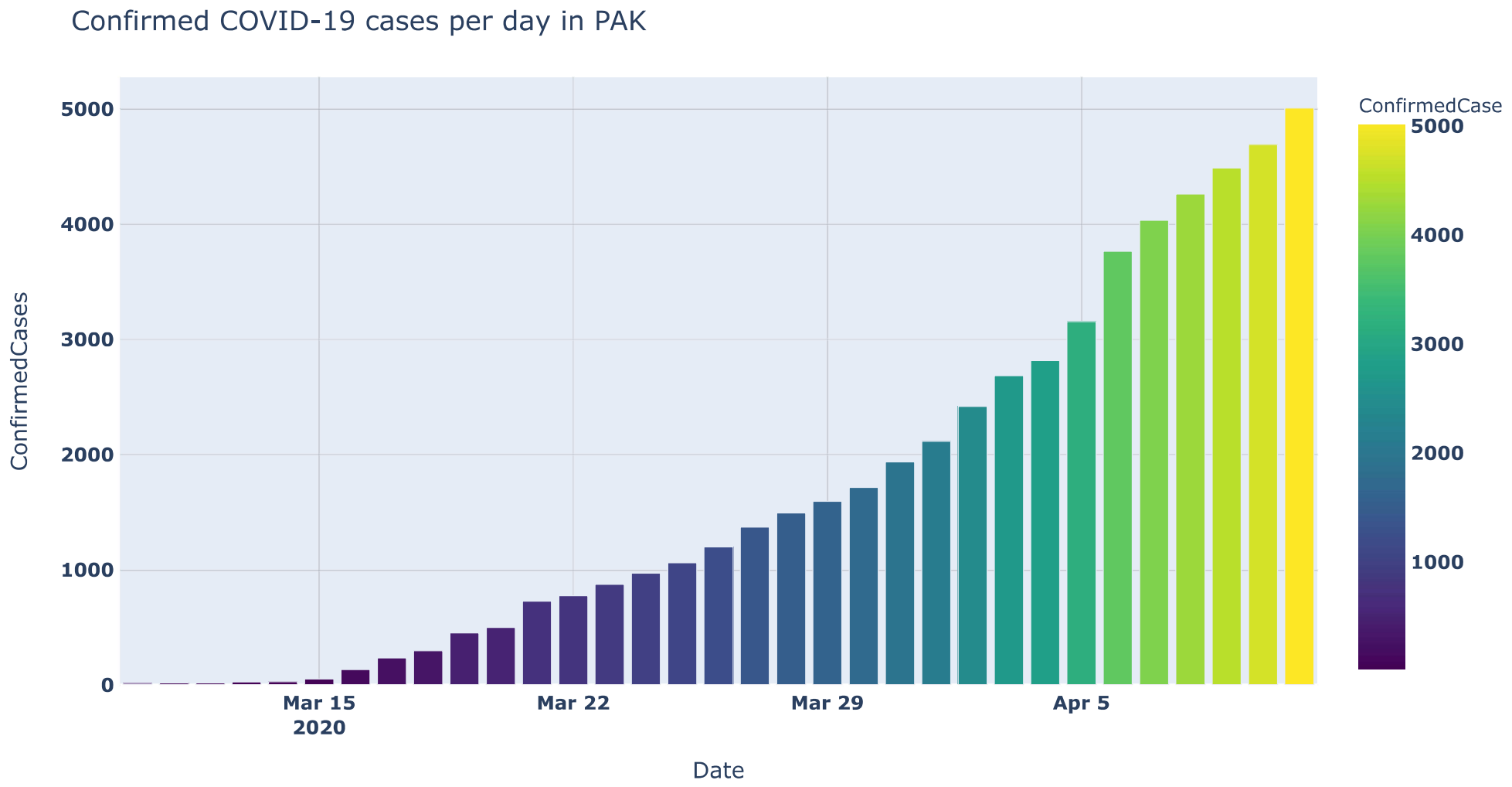
Distribution of Confirmed Cases of Bangladesh After March 08, 2020

In [34]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'Bangladesh') &(df_by_date_country.Date >= '2020-03-0
8')].sort_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.Viridis)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in BD')
fig.show()
```



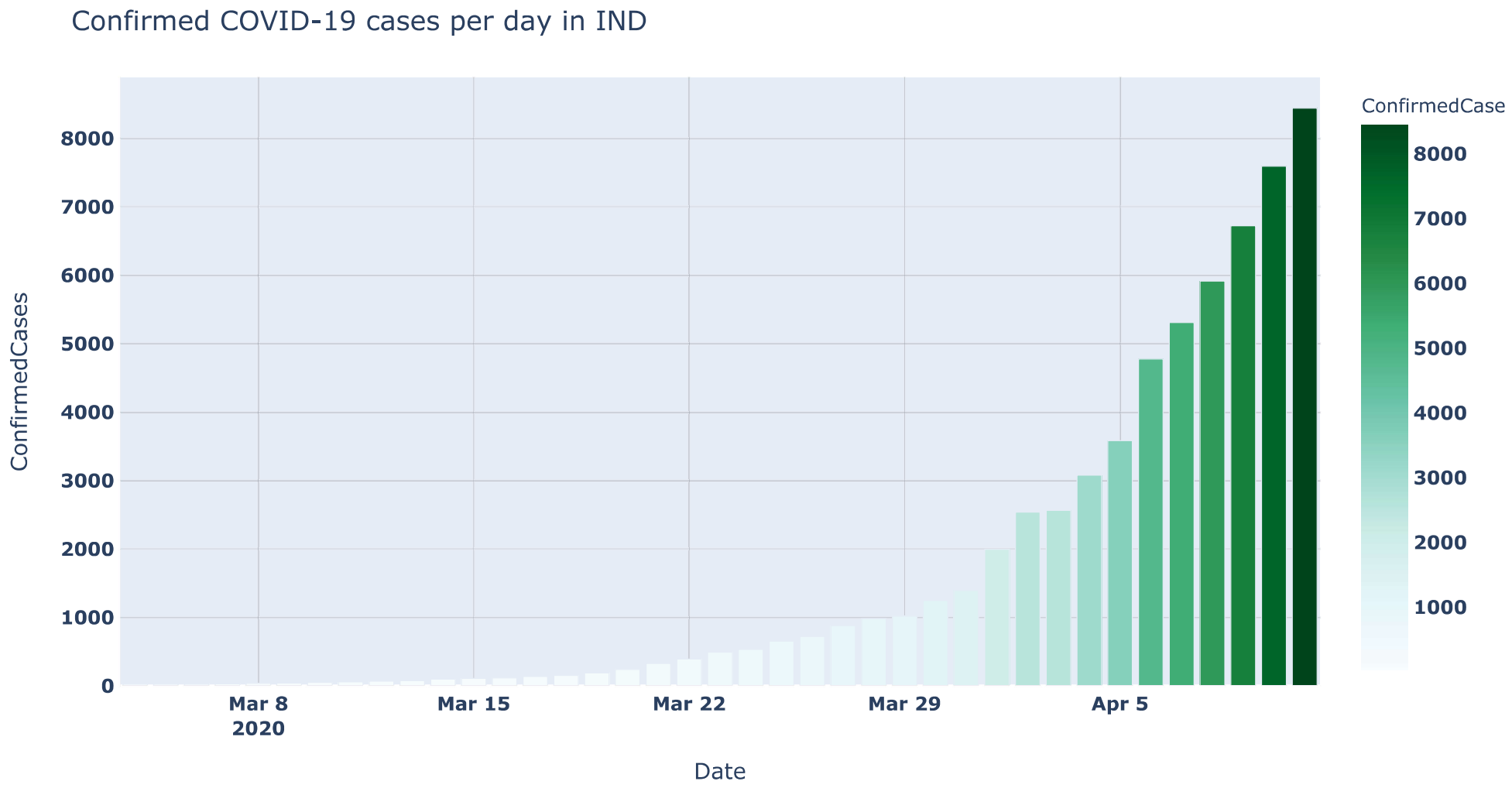
Distribution of Confirmed Cases of pakistan After March 10, 2020

In [36]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'Pakistan') &(df_by_date_country.Date >= '2020-03-10'
)].sort_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.Viridis)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in PAK')
fig.show()
```



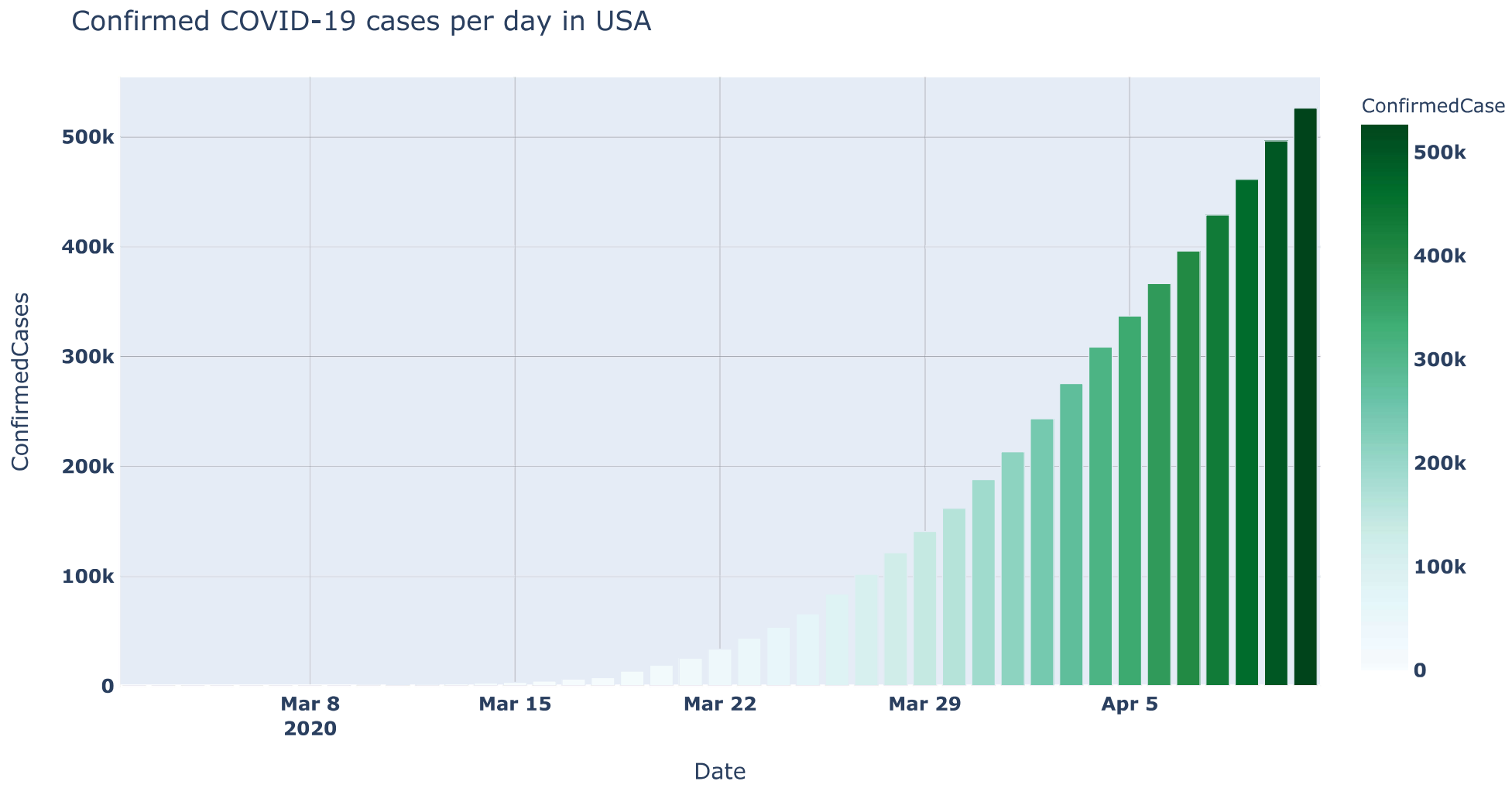
Distribution of Confirmed Cases of India After March 04, 2020

In [37]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'India') &(df_by_date_country.Date >= '2020-03-04')].s
ort_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.BuGn)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in IND')
fig.show()
```



Distribution of Confirmed Cases of United States After March 02, 2020

In [38]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'US') &(df_by_date_country.Date >= '2020-03-02')].sort
_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.BuGn)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in USA')
fig.show()
```

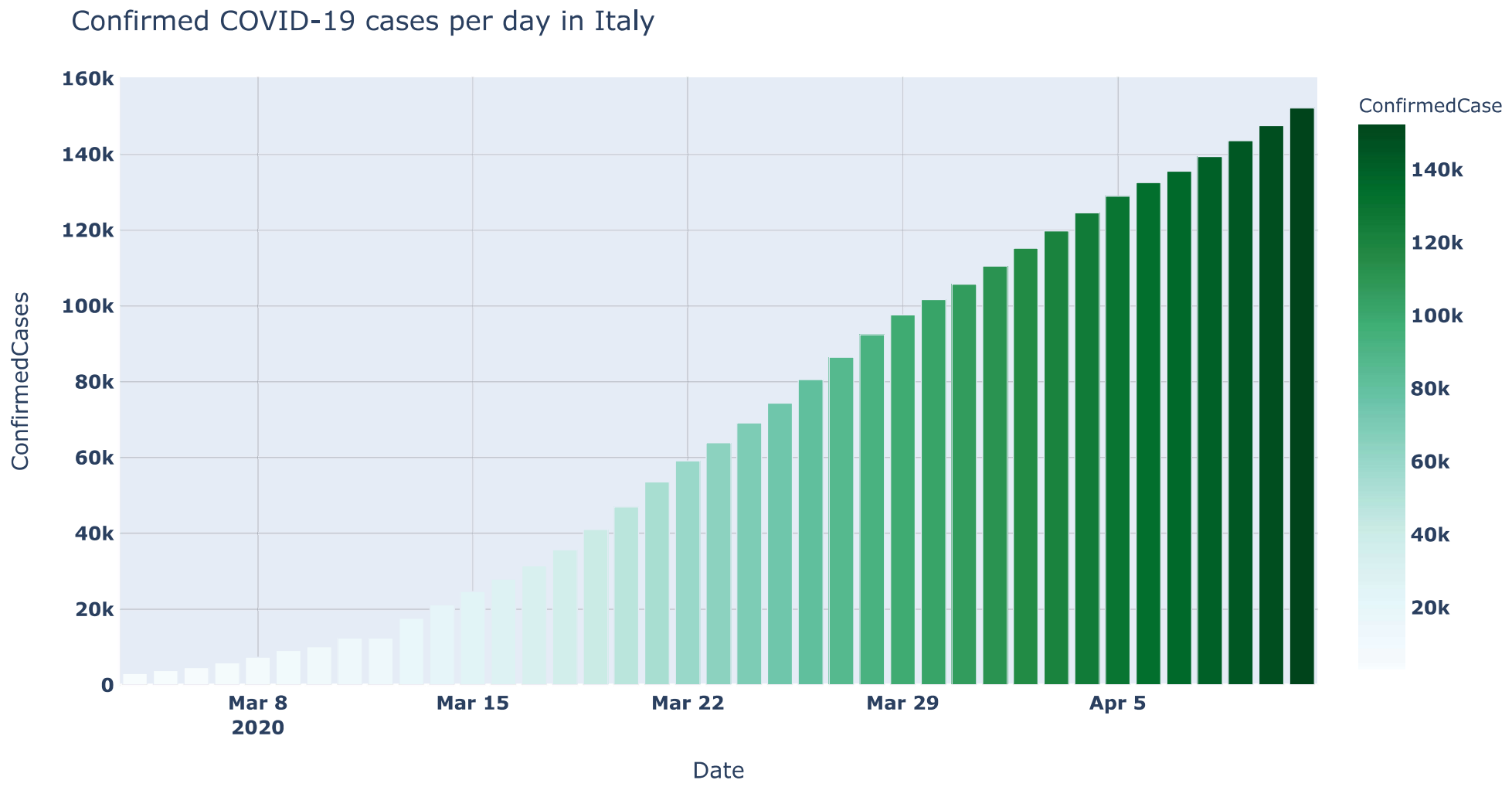


Distribution of Confirmed Cases of China After january 01, 2020

In [83]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'China') &(df_by_date_country.Date >= '2020-01-22')].s
ort_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.BuGn)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in China')
fig.show()
```

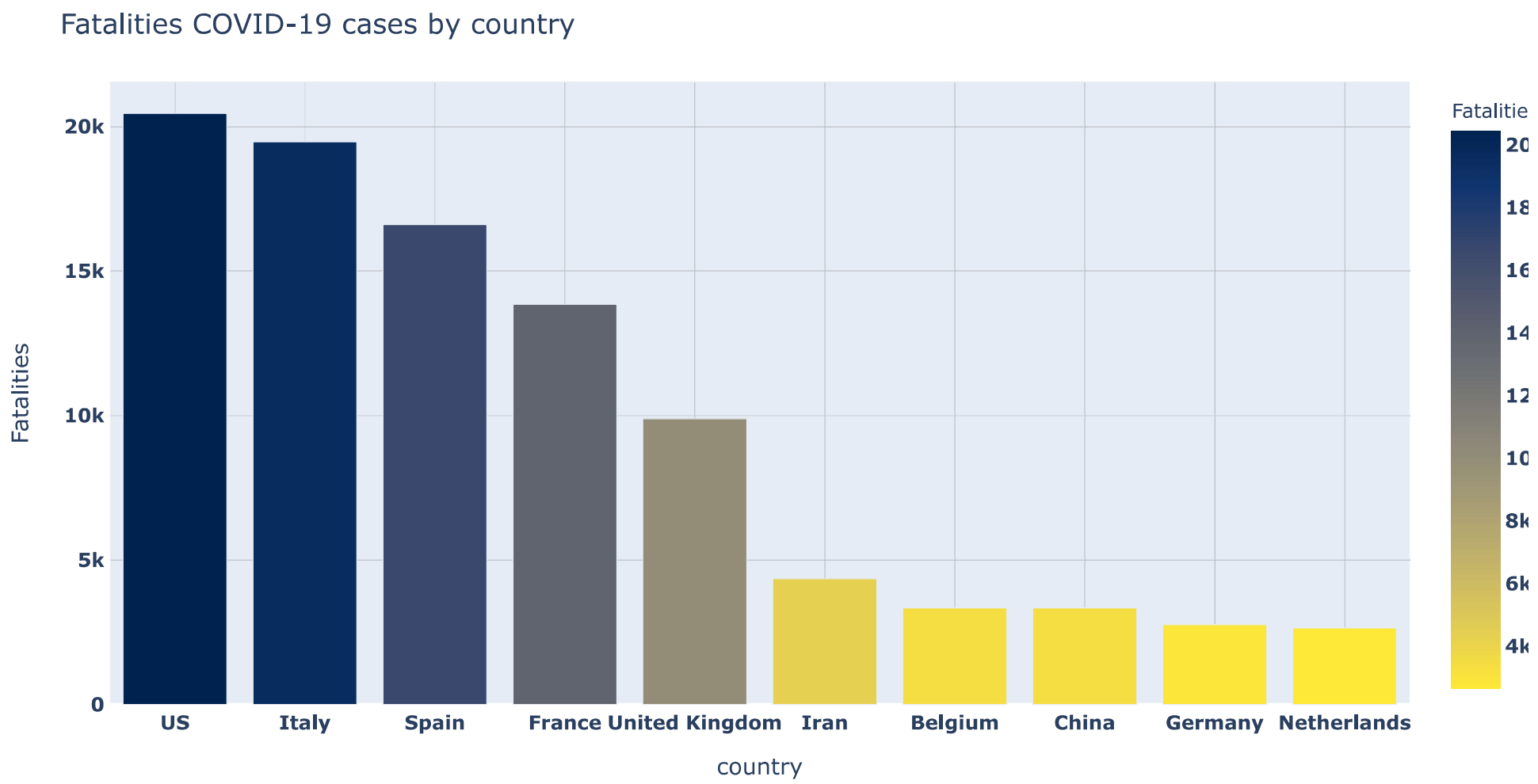
Distribution of Confirmed Cases of Italy After March 04, 2020

In [40]:
```
fig = px.bar(df_by_date_country.loc[(df_by_date_country['country'] == 'Italy') &(df_by_date_country.Date >= '2020-03-04')].s
ort_values('ConfirmedCases',ascending = False),
             x='Date', y='ConfirmedCases', color="ConfirmedCases", color_continuous_scale=px.colors.sequential.BuGn)
fig.update_layout(title_text='Confirmed COVID-19 cases per day in Italy')
fig.show()
```



Top Confirmed Cases Against the Country

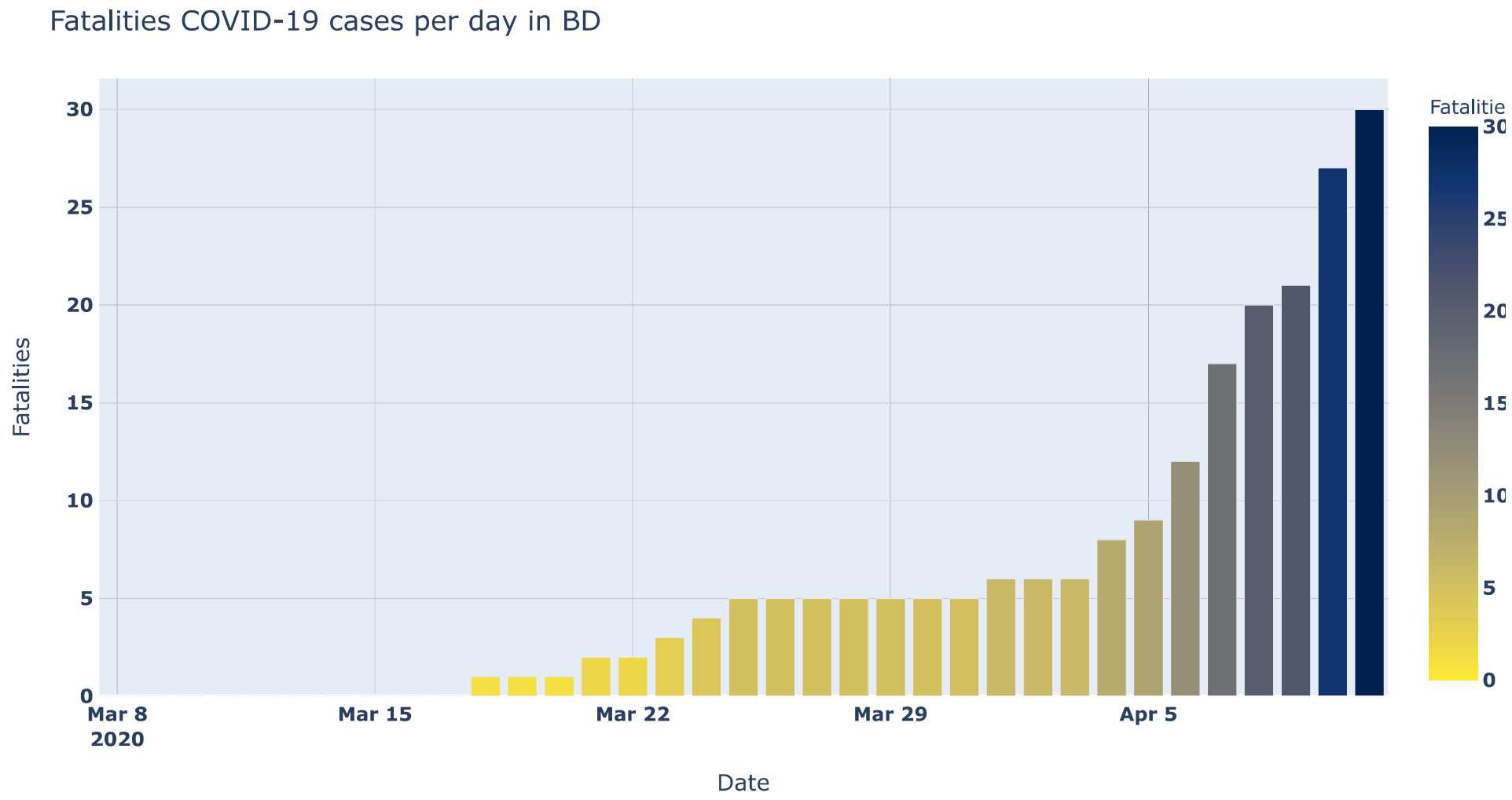
In [84]:
```
df_groupby_death = df.fillna('NA').groupby(['country','province','Date'])['Fatalities'].sum() \
                           .groupby(['country','province']).max().sort_values() \
                           .groupby(['country']).sum().sort_values(ascending = False)

top10_death = pd.DataFrame(df_groupby_death).head(10)
fig = px.bar(top10_death, x=top10_death.index, y='Fatalities', labels={'x':'Country'},
             color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases by country')
fig.show()
```



In [87]:
```
df_death_by_date = pd.DataFrame(df.fillna('NA').groupby(['country','Date'])['Fatalities'].sum().sort_values().reset_index())
```

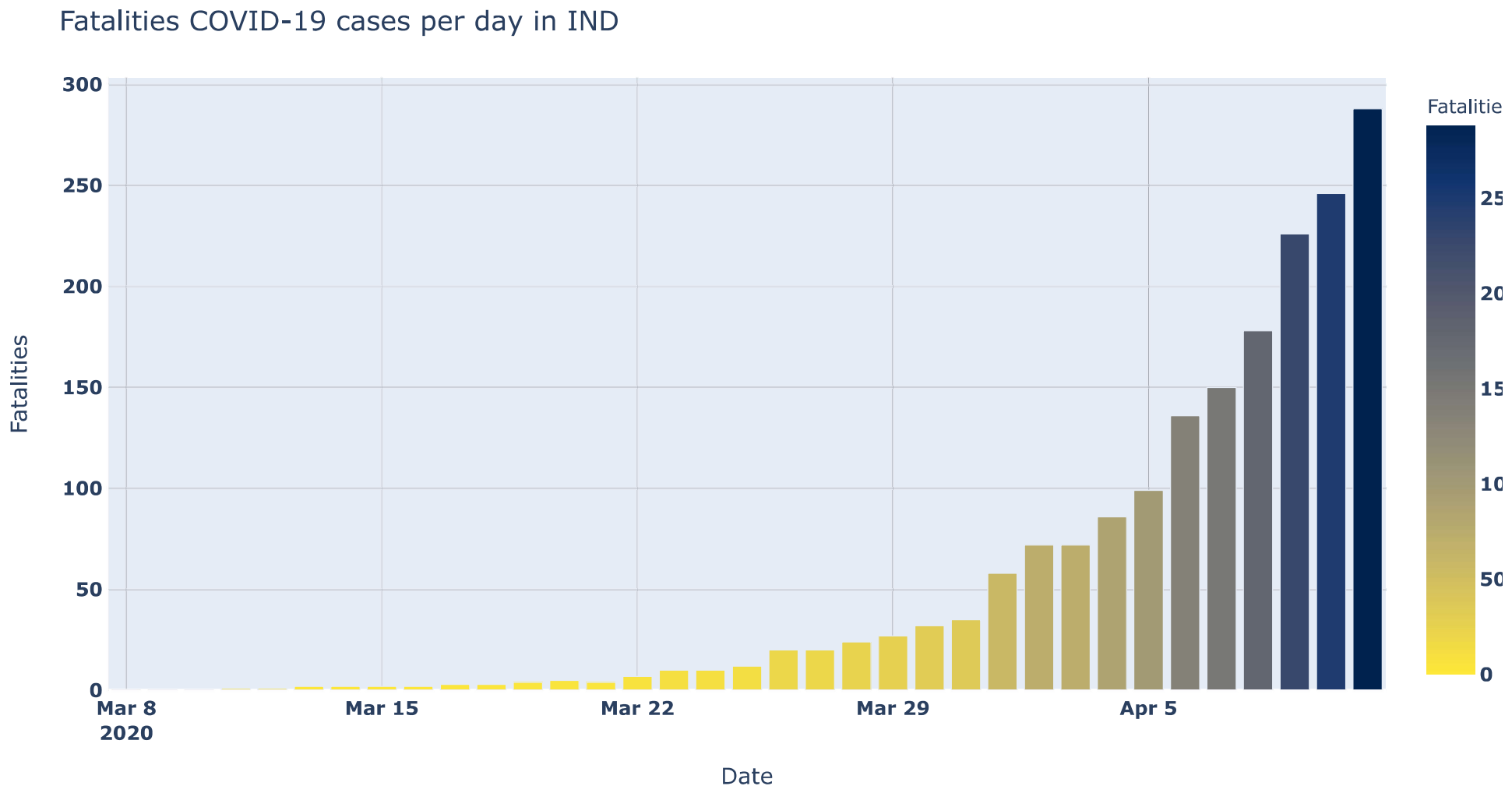
Distribution of Fatalities Cases of Bangladesh After March 08, 2020

In [86]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'Bangladesh') &(df_death_by_date.Date >= '2020-03-08')].so
rt_values('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in BD')
fig.show()
```



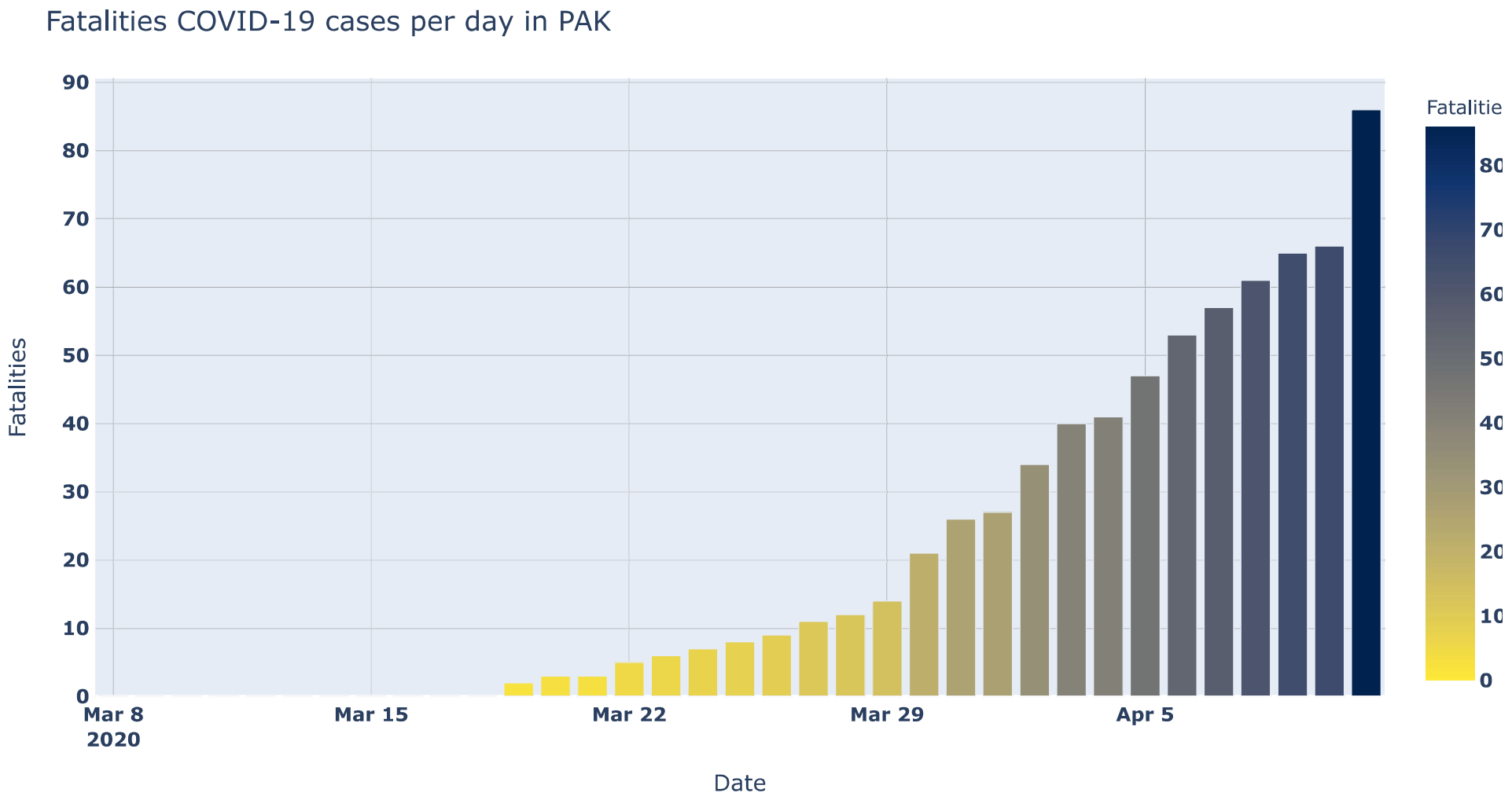
Distribution of Fatalities Cases of India After March 08, 2020

In [44]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'India') &(df_death_by_date.Date >= '2020-03-08')].sort_va
lues('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in IND')
fig.show()
```



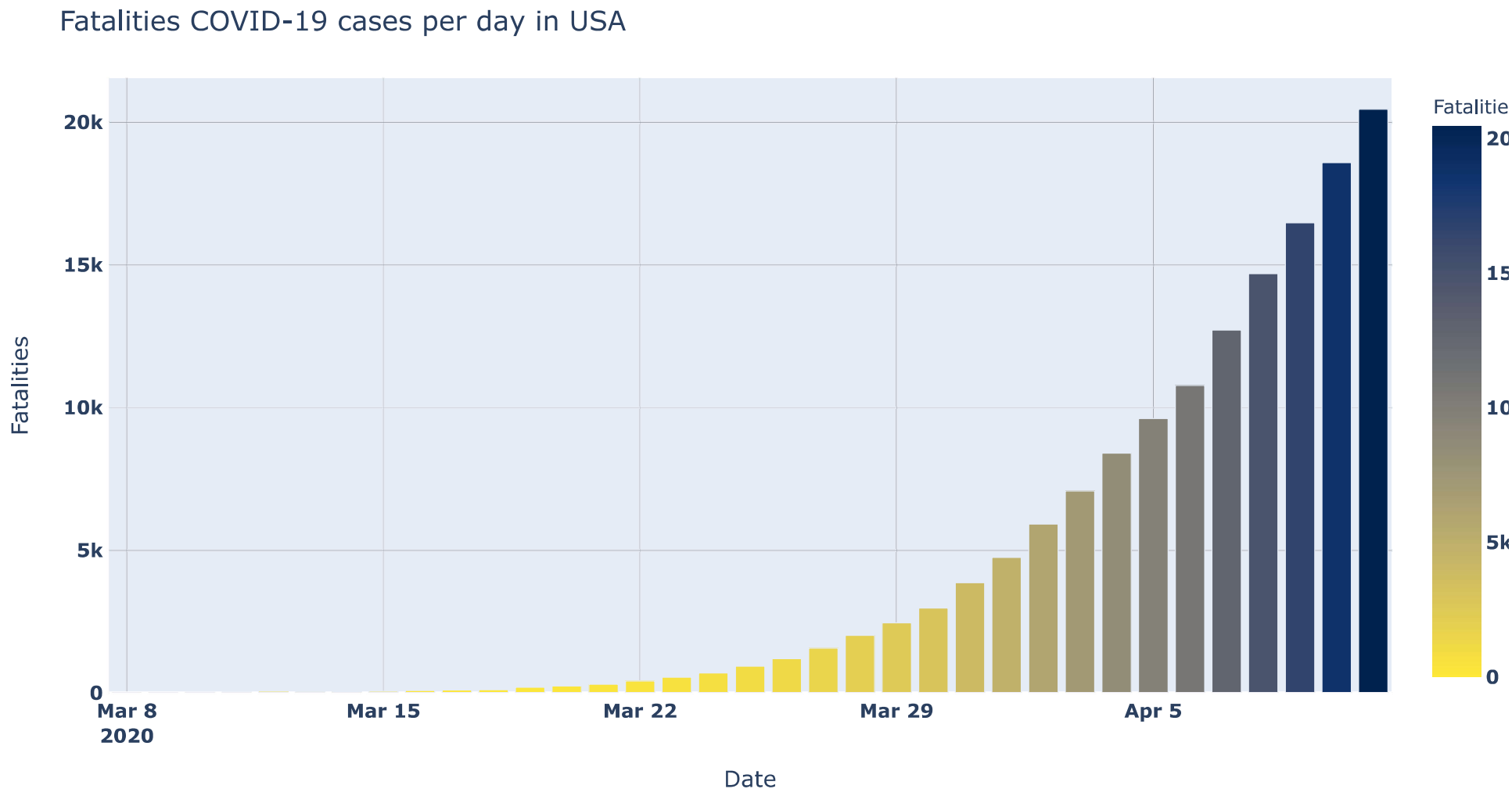
Distribution of Fatalities Cases of Pakistan After March 08, 2020

In [45]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'Pakistan') &(df_death_by_date.Date >= '2020-03-08')].sort
_values('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in PAK')
fig.show()
```



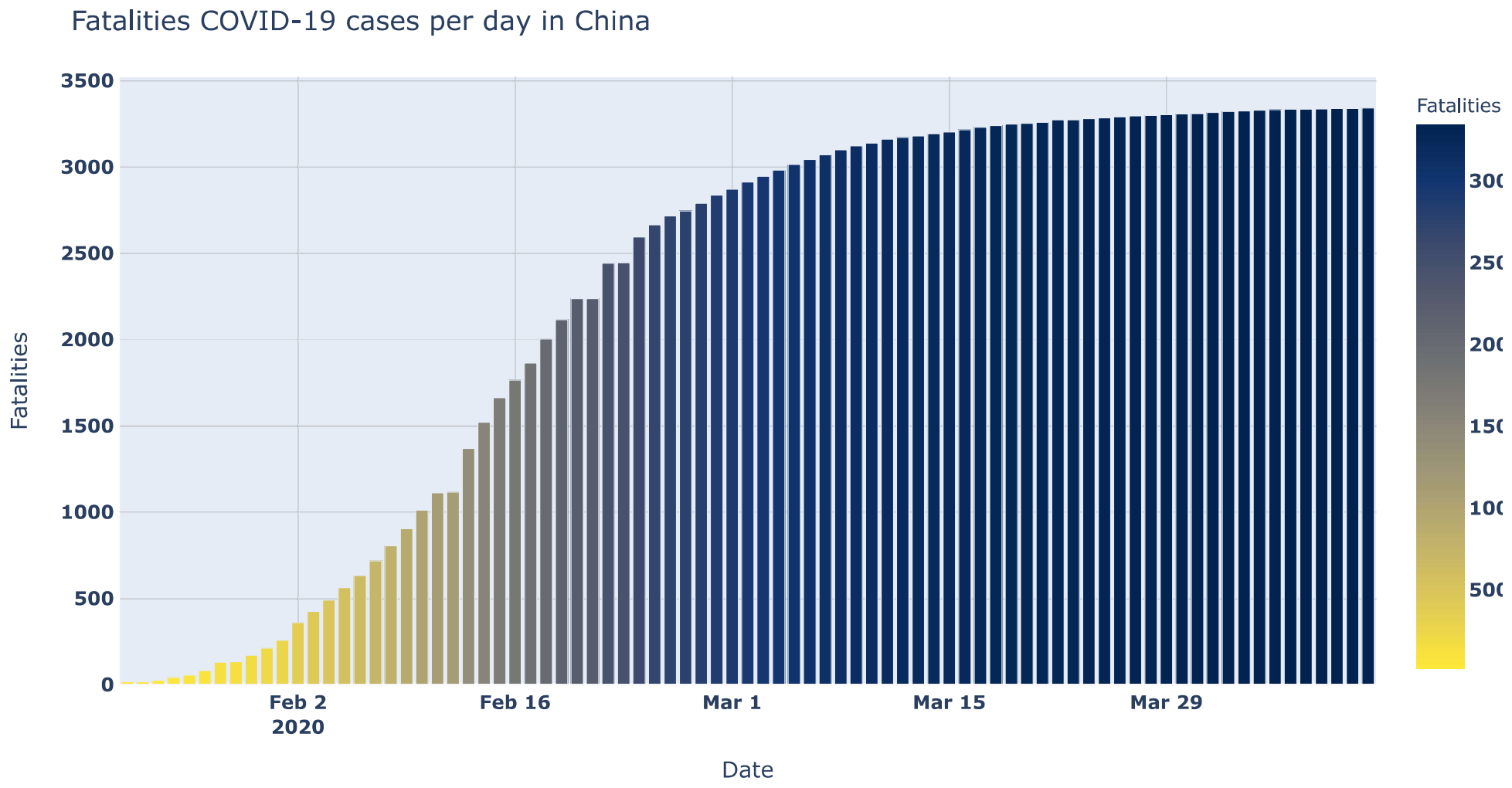
Distribution of Fatalities Cases of united States After March 08, 2020

In [46]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'US') &(df_death_by_date.Date >= '2020-03-08')].sort_value
s('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in USA')
fig.show()
```



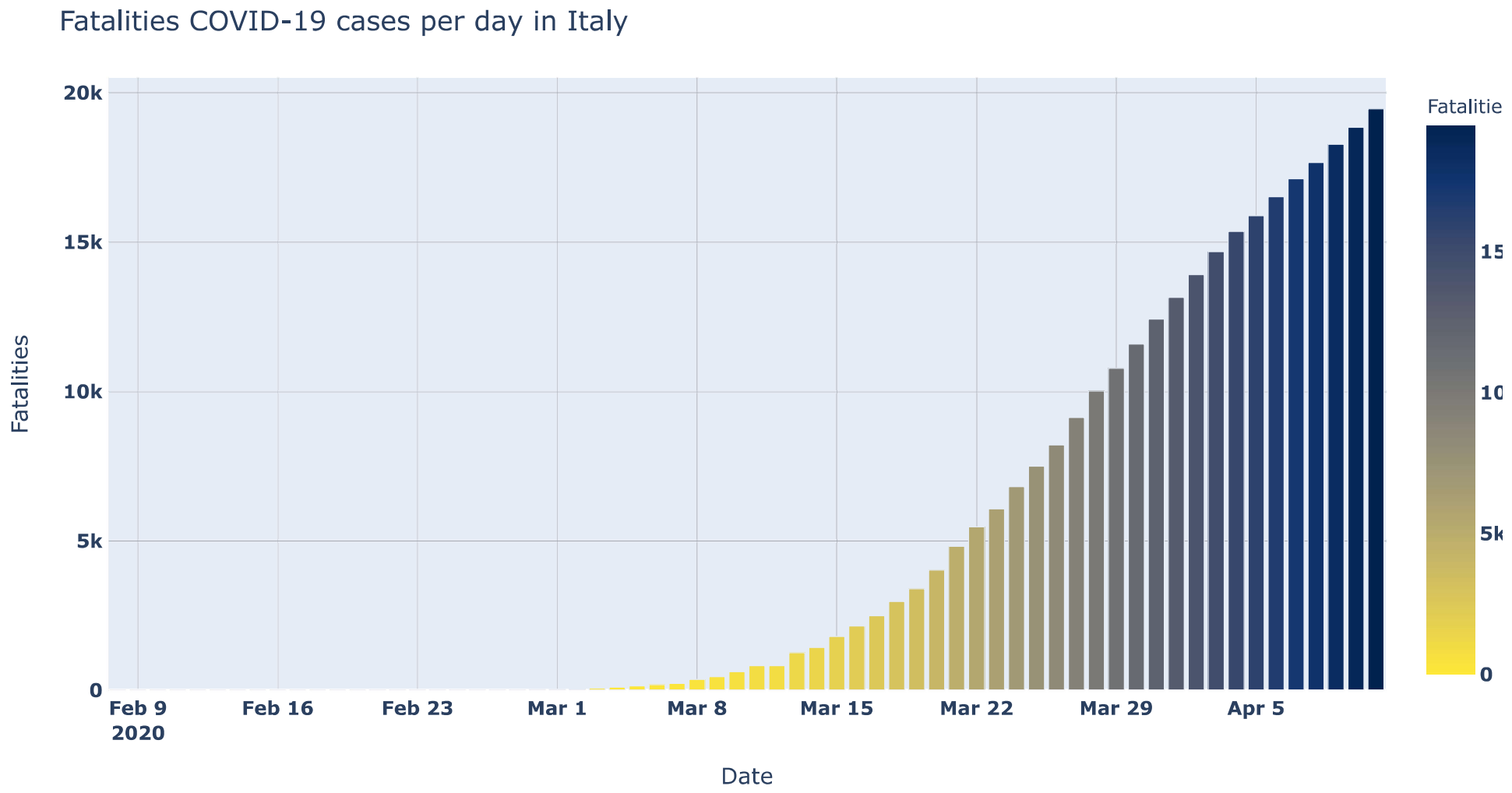
Distribution of Fatalities Cases of China After November 01, 2020

In [47]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'China') &(df_death_by_date.Date >= '2020-01-01')].sort_va
lues('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in China')
fig.show()
```



Distribution of Fatalities Cases of Italy After February 08, 2020

In [48]:
```
fig = px.bar(df_death_by_date.loc[(df_death_by_date['country'] == 'Italy') &(df_death_by_date.Date >= '2020-02-08')].sort_va
lues('Fatalities',ascending = False),
             x='Date', y='Fatalities', color="Fatalities", color_continuous_scale=px.colors.sequential.Cividis_r)
fig.update_layout(title_text='Fatalities COVID-19 cases per day in Italy')
fig.show()
```



Compare the Distribution of Fatalities Cases and Confirmed Cases of Top Countries with due to Covide 19 diseases

And also find the new cases of Infected and Death using the data manipulation techniques.

In [49]:
```
df_compare = df.fillna('NA').groupby(['country','province','Date'])['ConfirmedCases','Fatalities'].sum() \
                            .groupby(['country','province']).max().sort_values(by='ConfirmedCases') \
                            .groupby(['country']).sum().sort_values(by='ConfirmedCases',ascending = False)

df_compare = pd.DataFrame(df_compare).reset_index()


df_compare = pd.DataFrame(df_compare)

df_new_cases = pd.DataFrame(df.fillna('NA').groupby(['country','Date'])['ConfirmedCases'].sum() \
                            .reset_index()).sort_values(['country','Date'])
df_new_cases.ConfirmedCases = df_new_cases.ConfirmedCases.diff().fillna(0)
df_new_cases = df_new_cases.loc[df_new_cases['Date'] == max(df_new_cases['Date']),['country','ConfirmedCases']]
df_new_cases.rename(columns={"ConfirmedCases": "NewCases"}, inplace=True, errors="raise")

df_new_deaths = pd.DataFrame(df.fillna('NA').groupby(['country','Date'])['Fatalities'].sum() \
                            .reset_index()).sort_values(['country','Date'])

df_new_deaths.Fatalities = df_new_deaths.Fatalities.diff().fillna(0)
df_new_deaths = df_new_deaths.loc[df_new_deaths['Date'] == max(df_new_deaths['Date']),['country','Fatalities']]

df_new_deaths.rename(columns={"Fatalities": "NewFatalities"}, inplace=True, errors="raise")

merged = df_compare.merge(df_new_cases, left_on='country', right_on='country')\
                   .merge(df_new_deaths, left_on='country', right_on='country')

new = pd.DataFrame(merged)

new_df=new.head(10)
new_df
```

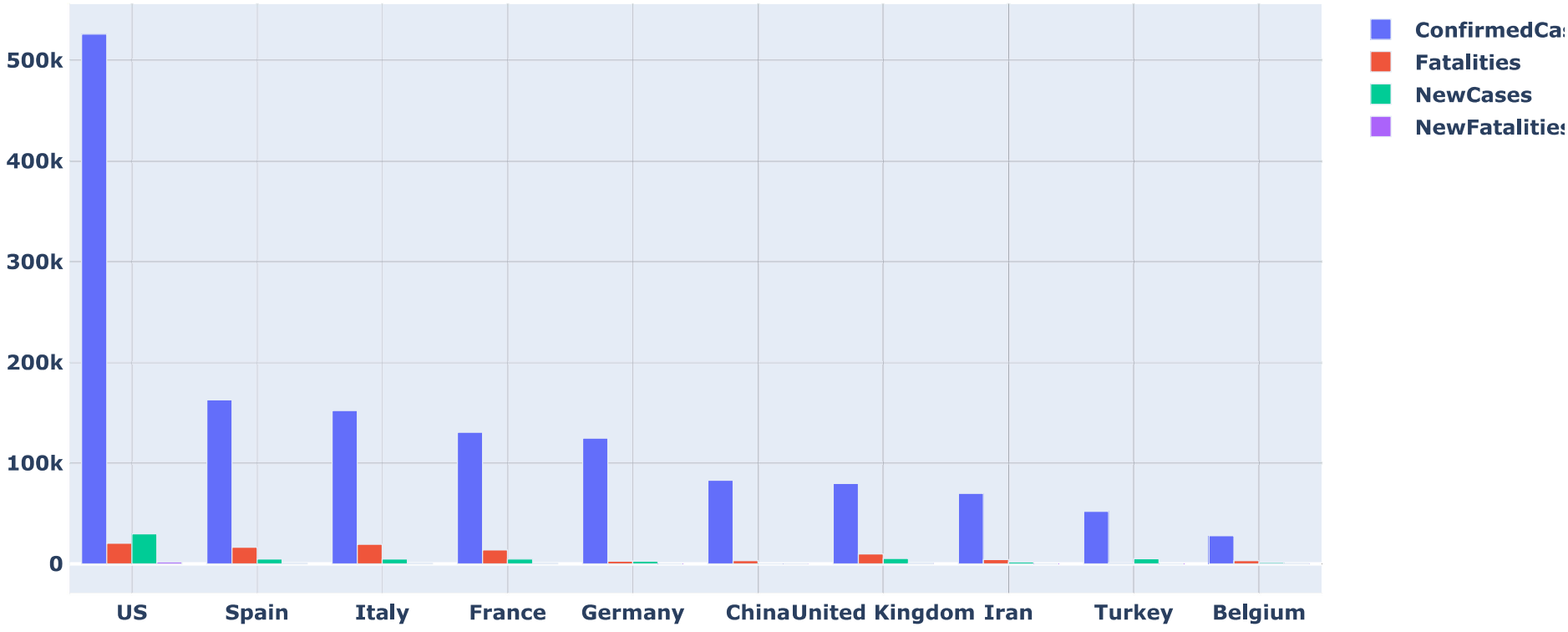
Out[49]:

	country	ConfirmedCases	Fatalities	NewCases	NewFatalities
0	US	526253.0	20458.0	29861.0	1877.0
1	Spain	163027.0	16606.0	4754.0	525.0
2	Italy	152271.0	19468.0	4694.0	619.0
3	France	130727.0	13851.0	4796.0	636.0
4	Germany	124908.0	2767.0	2737.0	-31.0
5	China	83015.0	3343.0	73.0	3.0
6	United Kingdom	79874.0	9892.0	5269.0	918.0
7	Iran	70029.0	4357.0	1837.0	125.0
8	Turkey	52167.0	1101.0	5138.0	95.0
9	Belgium	28018.0	3346.0	1351.0	327.0

Plot the graph of Confirm Cases , Fatalities Cases, new Cases and New Death Cases of top 10 countries

In [51]:

```
countries=new_df.country.to_list()
ConfirmedCases = new_df.ConfirmedCases.to_list()
Fatalities = new_df.Fatalities.to_list()
NewCases = new_df.NewCases.to_list()
NewFatalities = new_df.NewFatalities.to_list()

fig = go.Figure(data=[
    go.Bar(name='ConfirmedCases', x=countries, y=ConfirmedCases),
    go.Bar(name='Fatalities', x=countries, y=Fatalities),
    go.Bar(name='NewCases', x=countries, y=NewCases),
    go.Bar(name='NewFatalities', x=countries, y=NewFatalities)
])

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


500k
400k
300k
200k
100k
0
US
Spain
Italy
France
Germany
China
United Kingdom
Iran
Turkey
Belgium
ConfirmedCa
Fatalities
NewCases
NewFatalitie

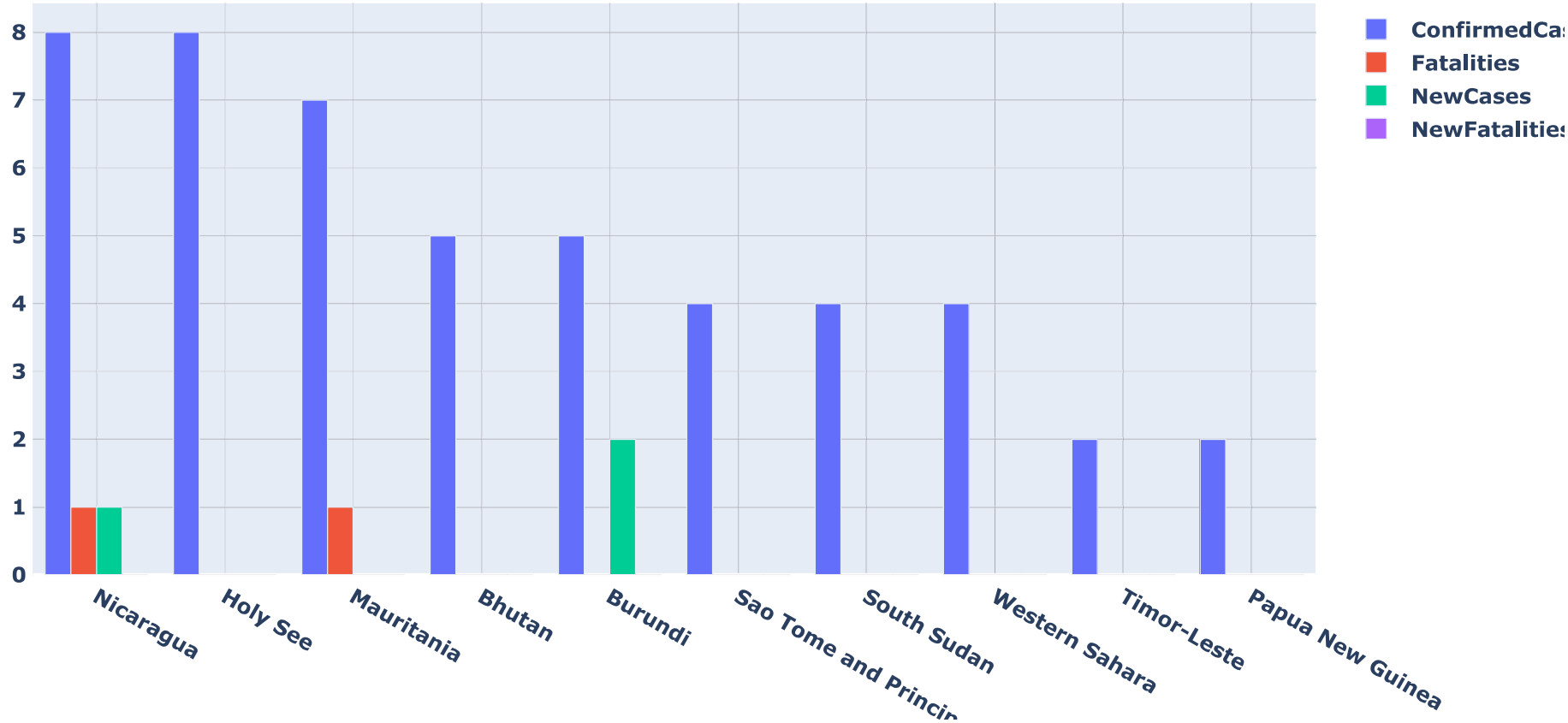
In [52]:
```
new_df_last=new.tail(10)
new_df_last
```

Out[52]:

	country	ConfirmedCases	Fatalities	NewCases	NewFatalities
174	Nicaragua	8.0	1.0	1.0	0.0
175	Holy See	8.0	0.0	0.0	0.0
176	Mauritania	7.0	1.0	0.0	0.0
177	Bhutan	5.0	0.0	0.0	0.0
178	Burundi	5.0	0.0	2.0	0.0
179	Sao Tome and Principe	4.0	0.0	0.0	0.0
180	South Sudan	4.0	0.0	0.0	0.0
181	Western Sahara	4.0	0.0	0.0	0.0
182	Timor-Leste	2.0	0.0	0.0	0.0
183	Papua New Guinea	2.0	0.0	0.0	0.0

Plot the graph of Confirm Cases , Fatalities Cases, new Cases and New Death Cases of min 10 countries

In [53]:
```
countries=new_df_last.country.to_list()
ConfirmedCases = new_df_last.ConfirmedCases.to_list()
Fatalities = new_df_last.Fatalities.to_list()
NewCases = new_df_last.NewCases.to_list()
NewFatalities = new_df_last.NewFatalities.to_list()

fig = go.Figure(data=[
    go.Bar(name='ConfirmedCases', x=countries, y=ConfirmedCases),
    go.Bar(name='Fatalities', x=countries, y=Fatalities),
    go.Bar(name='NewCases', x=countries, y=NewCases),
    go.Bar(name='NewFatalities', x=countries, y=NewFatalities)
])

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


8
7
6
5
4
3
2
1
0
Nicaragua
Holy See
Mauritania
Bhutan
Burundi
Sao Tome and Princi
South Sudan
Western Sahara
Timor-Leste
Papua New Guinea
ConfirmedCa
Fatalities
NewCases
NewFatalitie

End Of Assignment - 01

hash Analytics
Inzamamul Alam Munna
inzimunna@gmail.com
Bangladesh
Data Analyst

In [ ]: