COVID-19 Notebook: 🦠 Worldwide Cases and Deaths 🌍

Importing Libraries

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')
import plotly.express as px
import plotly.graph_objects as go
from plotly import tools
from plotly.subplots import make_subplots
from plotly.offline import iplot,init_notebook_mode
init_notebook_mode()
import warnings
warnings.filterwarnings('ignore')
from wordcloud import WordCloud,STOPWORDS
```

Reading Data

In [8]:

```
df=pd.read_csv('covid_worldwide.csv')
```

In [11]:

```
df
```

Out[11]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
1	2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
2	3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
3	4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
4	5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593
...	...	...	...	...	...	...	...	...
226	227	Diamond Princess	712	13	699	0	NaN	NaN
227	228	Vatican City	29	NaN	29	0	NaN	799
228	229	Western Sahara	10	1	9	0	NaN	626,161
229	230	MS Zaandam	9	2	7	0	NaN	NaN
230	231	Tokelau	5	NaN	NaN	5	NaN	1,378

231 rows × 8 columns

In [9]:

```
df.head()
```

Out[9]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
1	2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
2	3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
3	4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
4	5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593

In [10]:

```
df.tail()
```

Out[10]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
226	227	Diamond Princess	712	13	699	0	NaN	NaN
227	228	Vatican City	29	NaN	29	0	NaN	799
228	229	Western Sahara	10	1	9	0	NaN	626,161
229	230	MS Zaandam	9	2	7	0	NaN	NaN
230	231	Tokelau	5	NaN	NaN	5	NaN	1,378

In [12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 231 entries, 0 to 230
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Serial Number    231 non-null    int64 
 1   Country          231 non-null    object
 2   Total Cases      231 non-null    object
 3   Total Deaths     225 non-null    object
 4   Total Recovered  210 non-null    object
 5   Active Cases     212 non-null    object
 6   Total Test       213 non-null    object
 7   Population       228 non-null    object
dtypes: int64(1), object(7)
memory usage: 14.6+ KB
```

In [15]:

```
df.isnull().sum()
```

Out[15]:

```
Serial Number       0
Country             0
Total Cases         0
Total Deaths        6
Total Recovered    21
Active Cases       19
Total Test         18
Population          3
dtype: int64
```

In [16]:

```
#another way
df.isna().sum()
```

Out[16]:

```
Serial Number       0
Country             0
Total Cases         0
Total Deaths        6
Total Recovered    21
Active Cases       19
Total Test         18
Population          3
dtype: int64
```

In [17]:

```
df=df.fillna('0')
```

In [18]:

```
df.isna().sum()
```

Out[18]:

```
Serial Number      0
Country            0
Total Cases        0
Total Deaths       0
Total Recovered    0
Active Cases       0
Total Test         0
Population         0
dtype: int64
```

In [19]:

```
df.columns
```

Out[19]:

```
Index(['Serial Number', 'Country', 'Total Cases', 'Total Deaths',
       'Total Recovered', 'Active Cases', 'Total Test', 'Population'],
      dtype='object')
```

In [20]:

```
df['Total Cases']=df['Total Cases'].str.replace(',','',regex=True).astype('float')
```

In [21]:

```
df['Total Deaths']=df['Total Deaths'].str.replace(',','',regex=True).astype('float')
```

In [22]:

```
df['Total Recovered']=df['Total Recovered'].str.replace(',','',regex=True).astype('float')
```

In [23]:

```
df['Active Cases']=df['Active Cases'].str.replace(',','',regex=True).astype('float')
```

In [24]:

```
df['Total Test']=df['Total Test'].str.replace(',','',regex=True).astype('float')
```

In [25]:

```
df['Population']=df['Population'].str.replace(',','',regex=True).astype('float')
```

In [26]:

```
df
```

Out[26]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1	USA	104196861.0	1132935.0	101322779.0	1741147.0	1.159833e+09	3.348053e+08
1	2	India	44682784.0	530740.0	44150289.0	1755.0	9.152658e+08	1.406632e+09
2	3	France	39524311.0	164233.0	39264546.0	95532.0	2.714902e+08	6.558452e+07
3	4	Germany	37779833.0	165711.0	37398100.0	216022.0	1.223324e+08	8.388360e+07
4	5	Brazil	36824580.0	697074.0	35919372.0	208134.0	6.377617e+07	2.153536e+08
...	...	...	...	...	...	...	...	...
226	227	Diamond Princess	712.0	13.0	699.0	0.0	0.000000e+00	0.000000e+00
227	228	Vatican City	29.0	0.0	29.0	0.0	0.000000e+00	7.990000e+02
228	229	Western Sahara	10.0	1.0	9.0	0.0	0.000000e+00	6.261610e+05
229	230	MS Zaandam	9.0	2.0	7.0	0.0	0.000000e+00	0.000000e+00
230	231	Tokelau	5.0	0.0	0.0	5.0	0.000000e+00	1.378000e+03

231 rows × 8 columns

In [27]:

```
df.describe()
```

Out[27]:

	Serial Number	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
count	231.000000	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02
mean	116.000000	2.923460e+06	2.927706e+04	2.721732e+06	8.351410e+04	2.996123e+07	2.812322e+07
std	66.828138	9.479286e+06	1.041073e+05	9.116089e+06	7.344789e+05	1.133726e+08	1.016625e+08
min	1.000000	5.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	58.500000	2.400100e+04	1.795000e+02	1.208250e+04	1.850000e+01	2.260585e+05	4.063530e+05
50%	116.000000	2.065920e+05	1.965000e+03	1.315590e+05	7.390000e+02	1.671684e+06	5.511370e+06
75%	173.500000	1.296146e+06	1.390850e+04	1.255186e+06	9.328500e+03	1.148478e+07	2.152480e+07
max	231.000000	1.041969e+08	1.132935e+06	1.013228e+08	1.095262e+07	1.159833e+09	1.406632e+09

In [28]:

```
df.describe(include='all')
```

Out[28]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
count	231.000000	231	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02	2.310000e+02
unique	NaN	231	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	USA	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	1	NaN	NaN	NaN	NaN	NaN	NaN
mean	116.000000	NaN	2.923460e+06	2.927706e+04	2.721732e+06	8.351410e+04	2.996123e+07	2.812322e+07
std	66.828138	NaN	9.479286e+06	1.041073e+05	9.116089e+06	7.344789e+05	1.133726e+08	1.016625e+08
min	1.000000	NaN	5.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	58.500000	NaN	2.400100e+04	1.795000e+02	1.208250e+04	1.850000e+01	2.260585e+05	4.063530e+05
50%	116.000000	NaN	2.065920e+05	1.965000e+03	1.315590e+05	7.390000e+02	1.671684e+06	5.511370e+06
75%	173.500000	NaN	1.296146e+06	1.390850e+04	1.255186e+06	9.328500e+03	1.148478e+07	2.152480e+07
max	231.000000	NaN	1.041969e+08	1.132935e+06	1.013228e+08	1.095262e+07	1.159833e+09	1.406632e+09

The number of countries in which the virus was detected

In [30]:

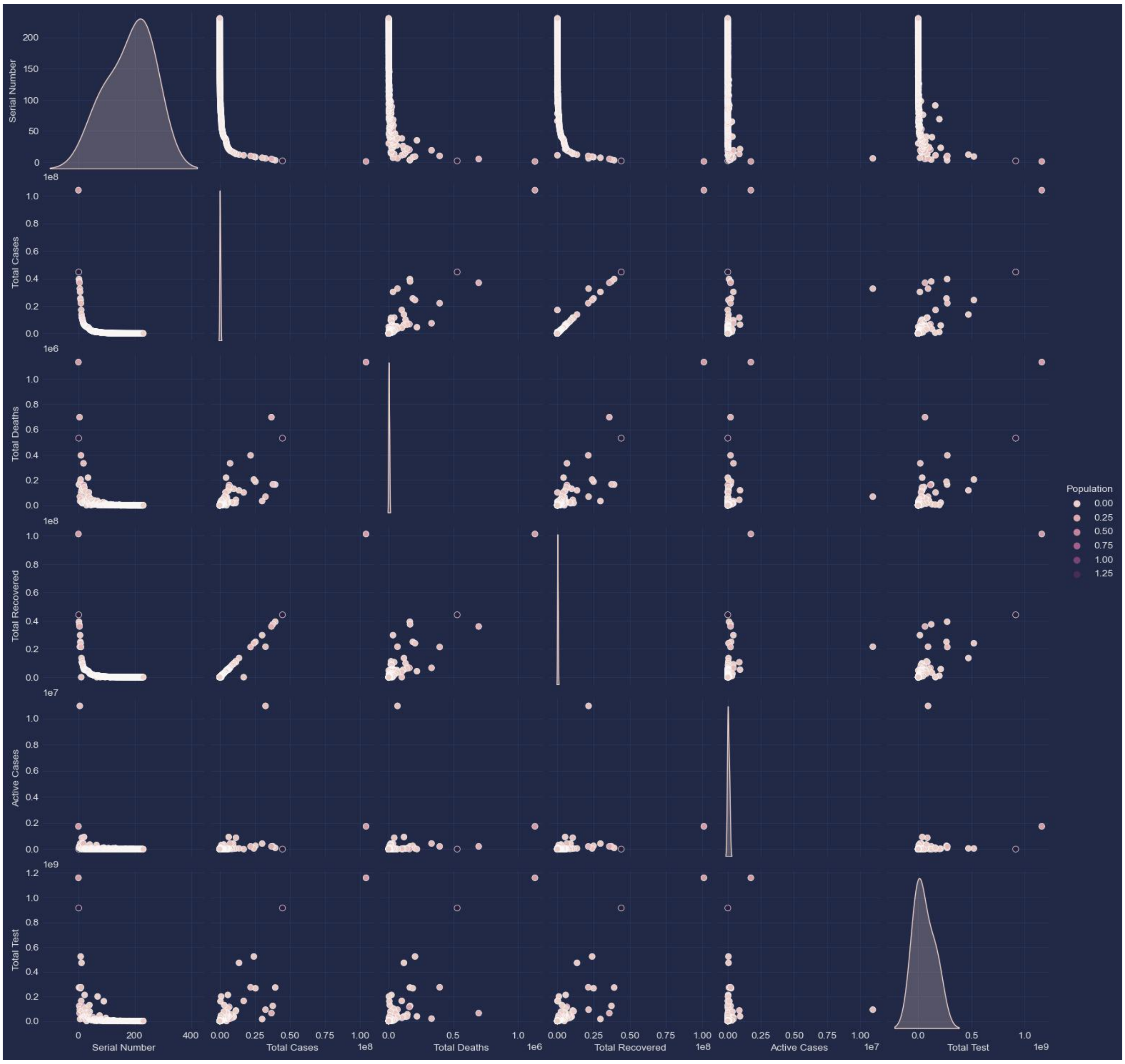
```
df['Country'].nunique()
```

Out[30]:

```
231
```

In [31]:

```
sns.pairplot(df,hue='Population')
plt.show()
```

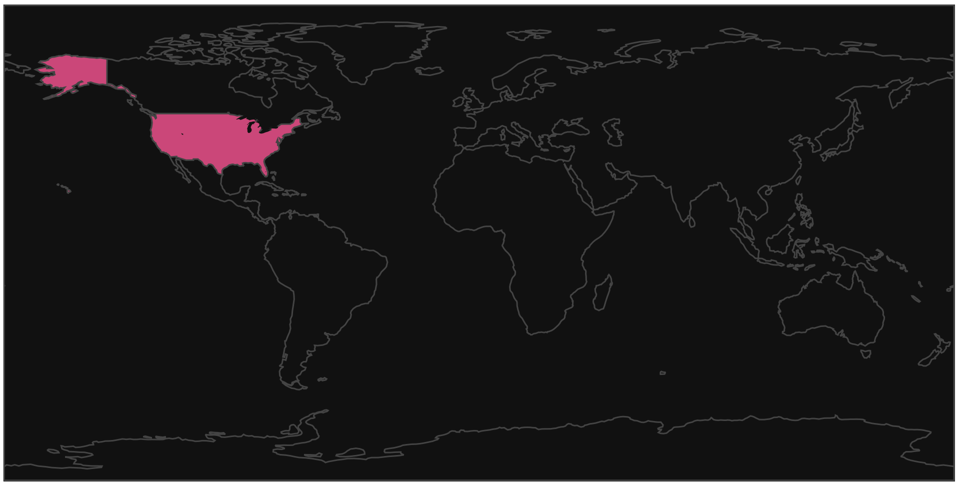


Distribution of the number of people who could not cope with the disease

In [32]:

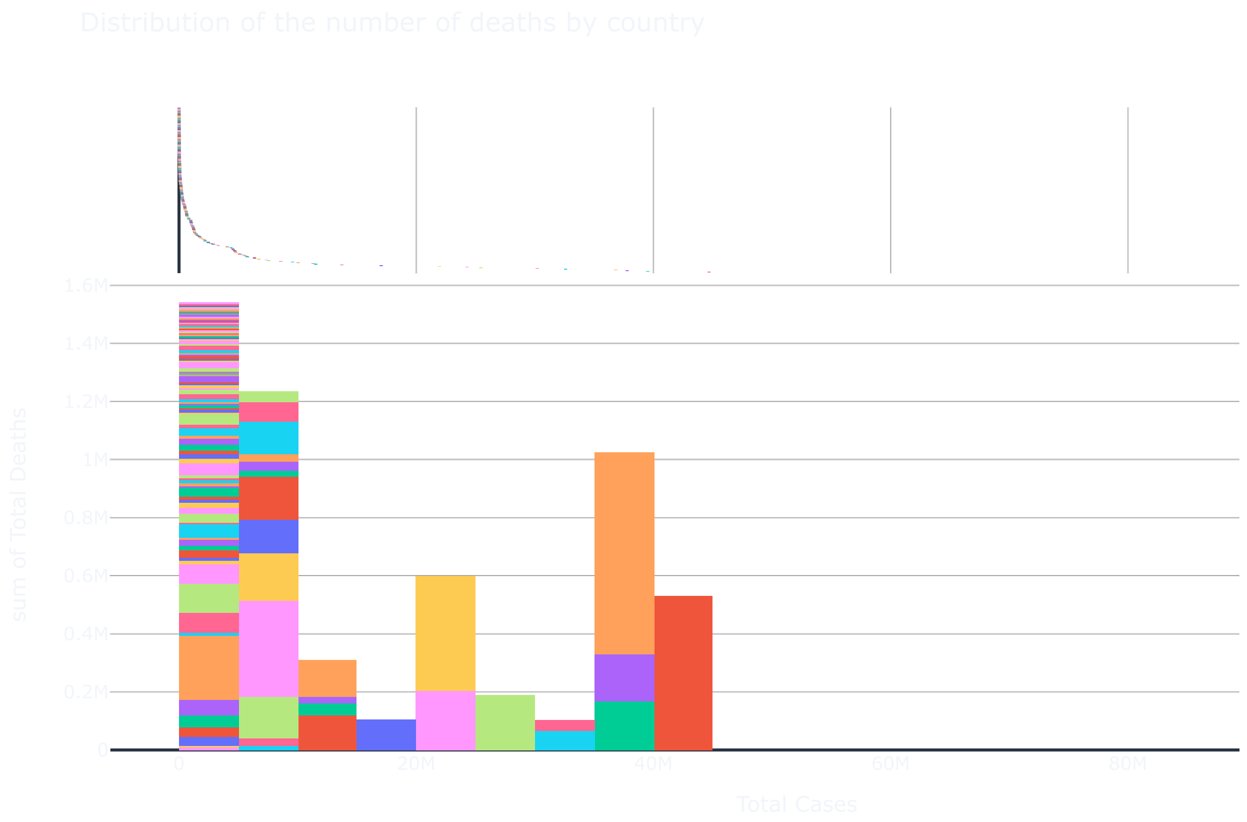
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Total Deaths']
                  ,animation_frame=df['Total Cases'],animation_group=df['Total Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of deaths by country'})
fig.show()
```

Distribution of the number of deaths by country



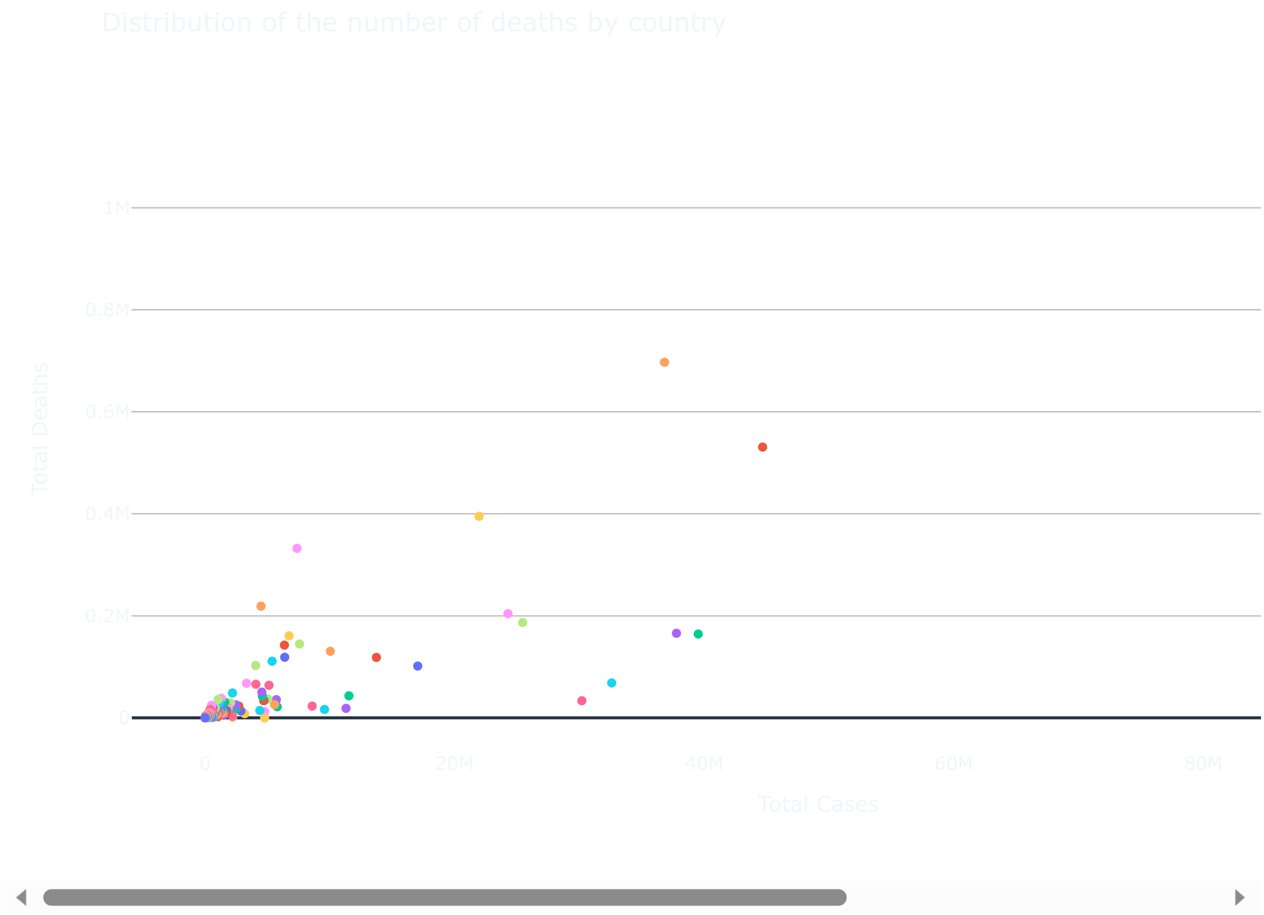
In [40]:

```
fig=px.histogram(df,x='Total Cases',color='Country',y='Total Deaths',marginal='box',template='plotly_dark',
                hover_data=df.columns,title="Distribution of the number of deaths by country",width=1200,height=600)
fig.show()
```



In [50]:

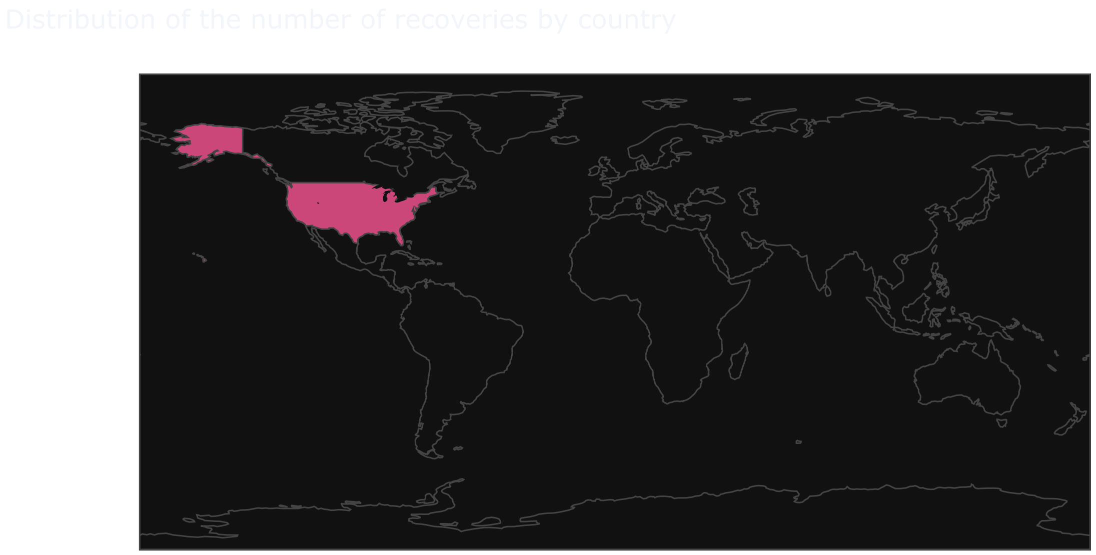
```
fig=px.violin(df,x='Total Cases',color='Country',y='Total Deaths',hover_data=df.columns,box=True,points='all',
             width=1200,height=600,template='plotly_dark',title='Distribution of the number of deaths by country')
fig.show()
```



Distribution the number of people who were able to recover by country

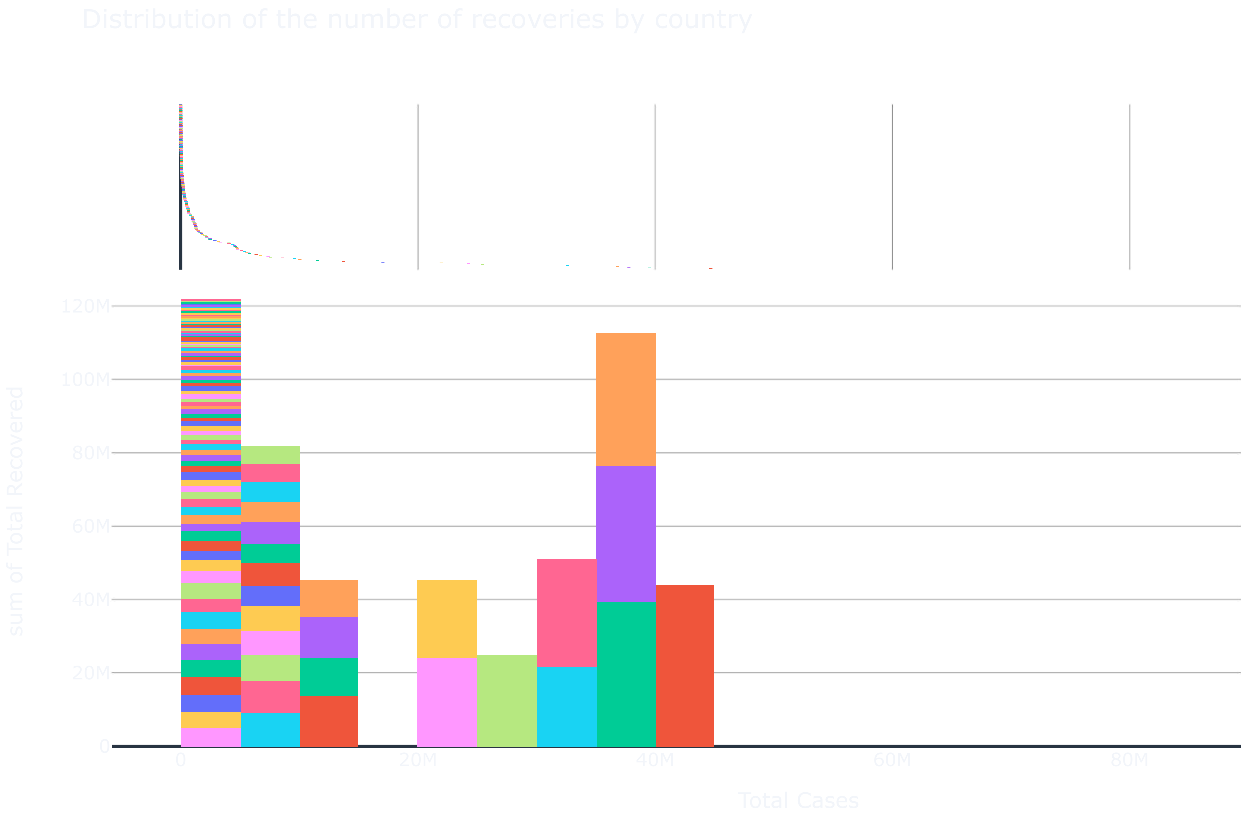
In [53]:

```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Total Recovered'],
                 animation_frame=df['Total Cases'],animation_group=df['Total Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of recoveries by country'})
fig.show()
```



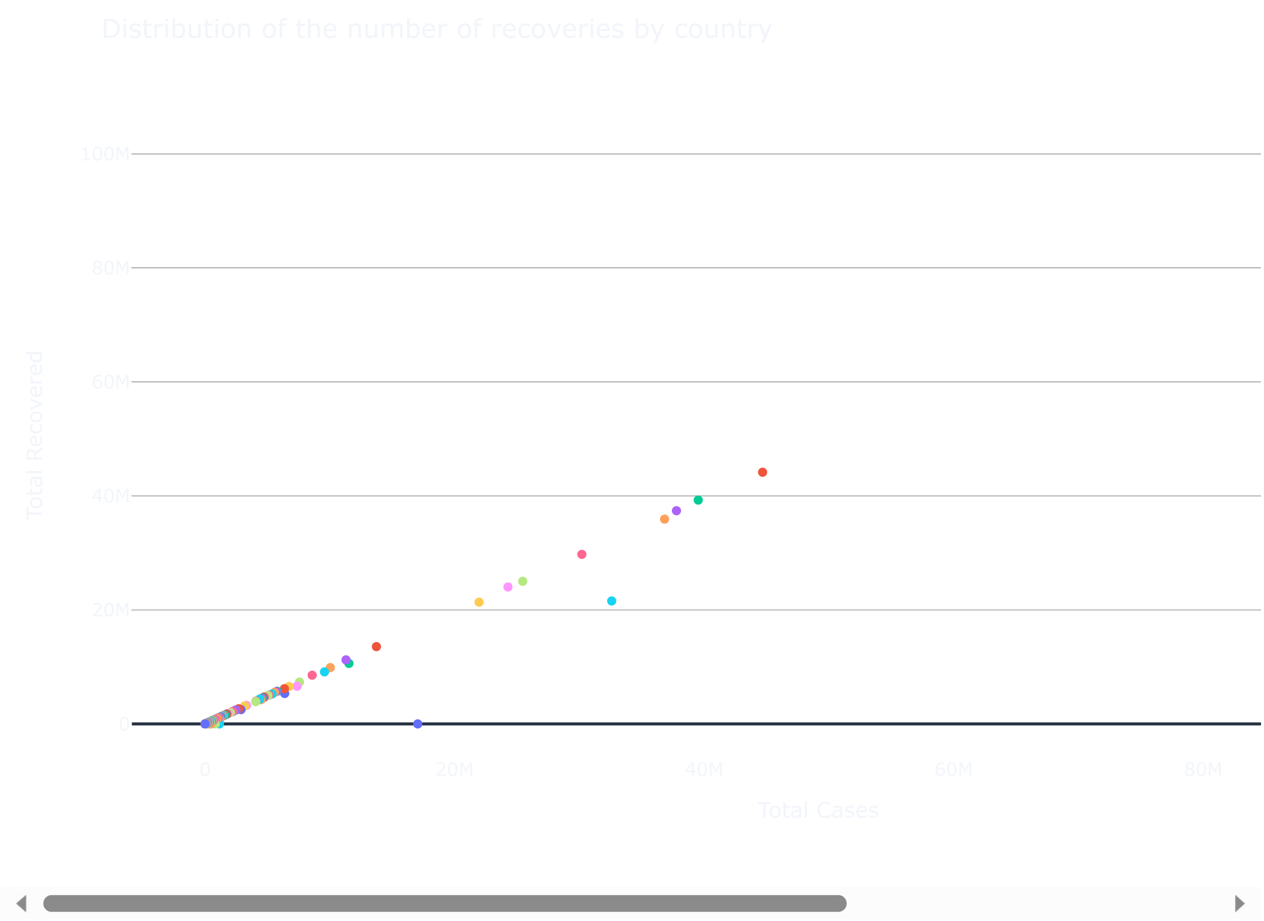
In [54]:

```
fig=px.histogram(df,x='Total Cases',y='Total Recovered',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of the number of recoveries by country',width=1200,
                height=600)
fig.show()
```



In [55]:

```
fig=px.violin(df,x='Total Cases',y='Total Recovered',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of the number of recoveries by country',width=1200,
             height=600)
fig.show()
```

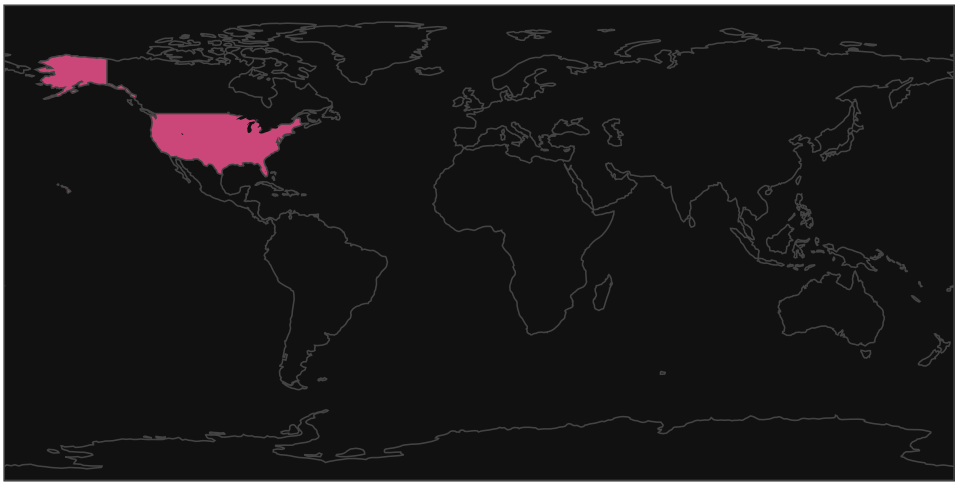


Distribution of the number of active cases by country

In [59]:

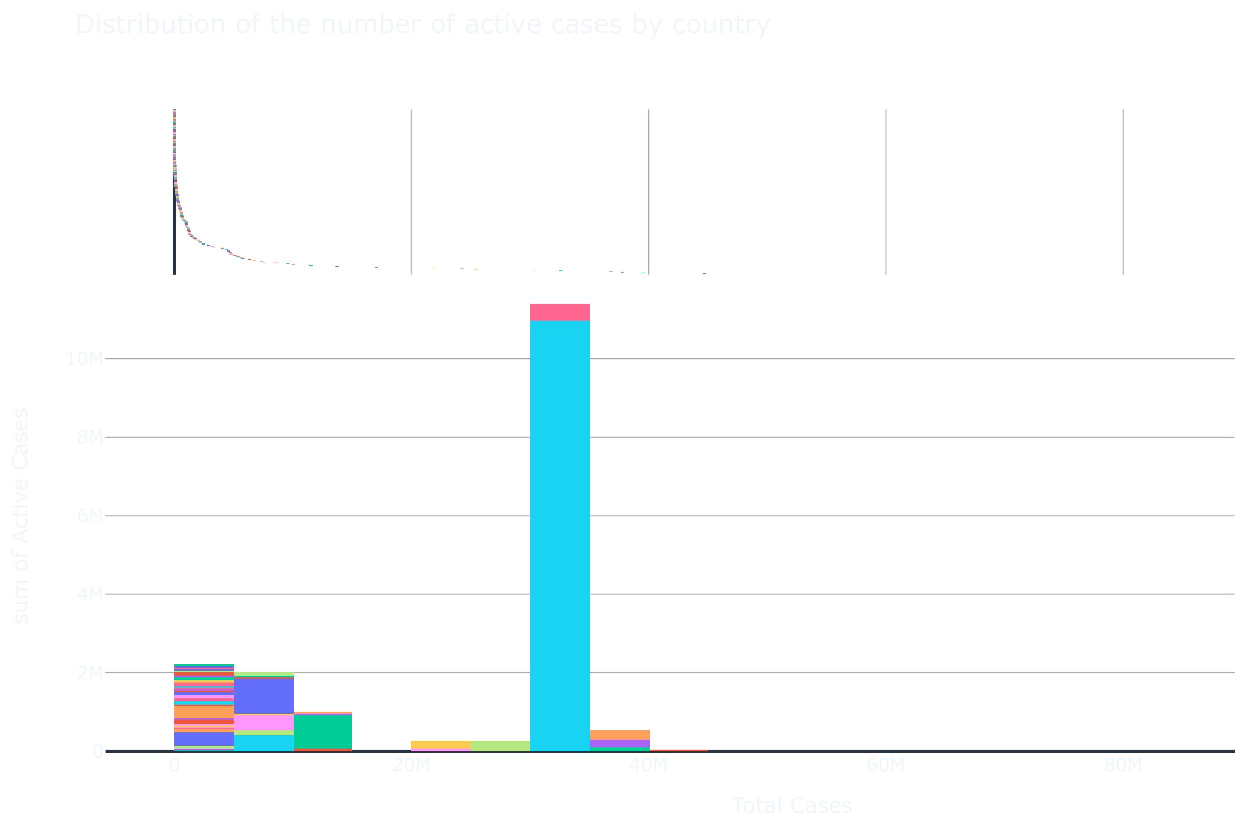
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Active Cases'],
                  animation_frame=df['Total Cases'],animation_group=df['Total Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of active cases by country'})
fig.show()
```

Distribution of the number of active cases by country



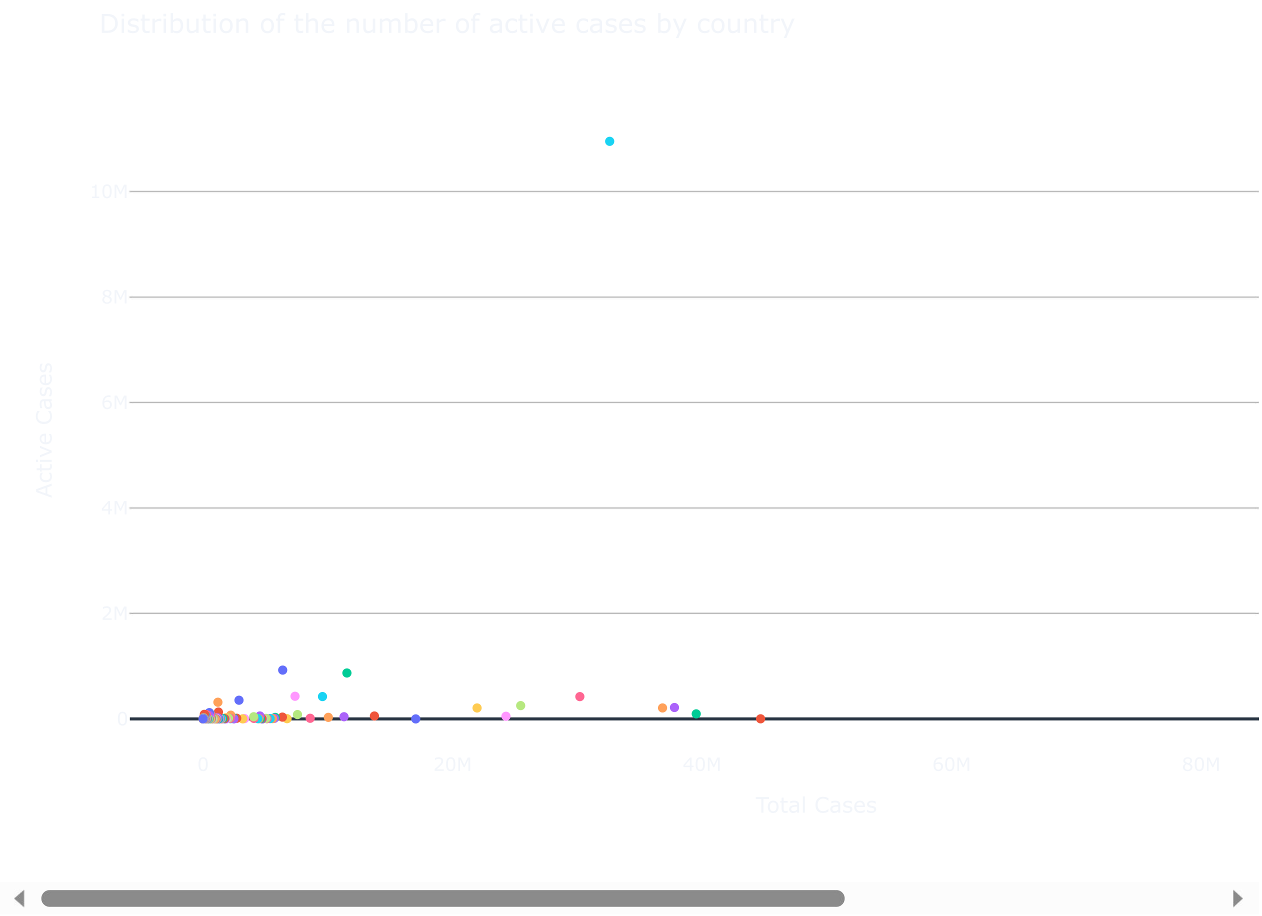
In [60]:

```
fig=px.histogram(df,x='Total Cases',y='Active Cases',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of the number of active cases by country',width=1200,
                height=600)
fig.show()
```



In [62]:

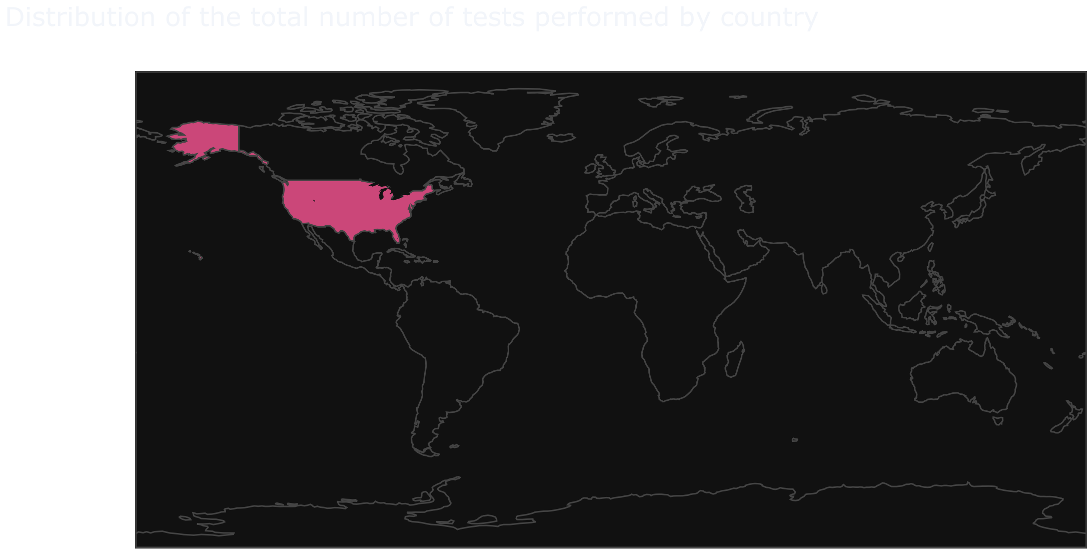
```
fig=px.violin(df,x='Total Cases',y='Active Cases',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of the number of active cases by country',width=1200,
             height=600)
fig.show()
```



Distribution of the total number of tests performed by country

In [63]:

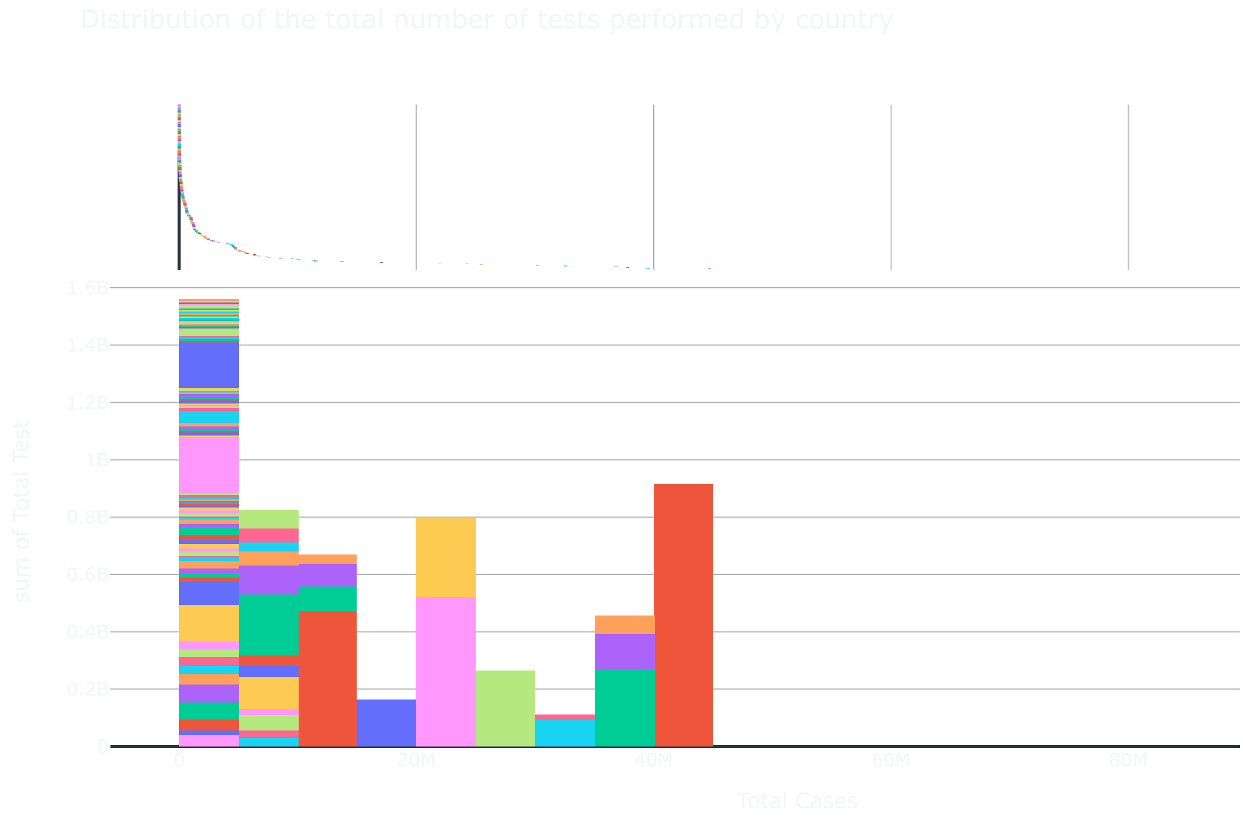
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Total Test'],
                  animation_frame=df['Total Cases'],animation_group=df['Total Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the total number of tests performed by country'})
fig.show()
```



Distribution of the total number of tests performed by country

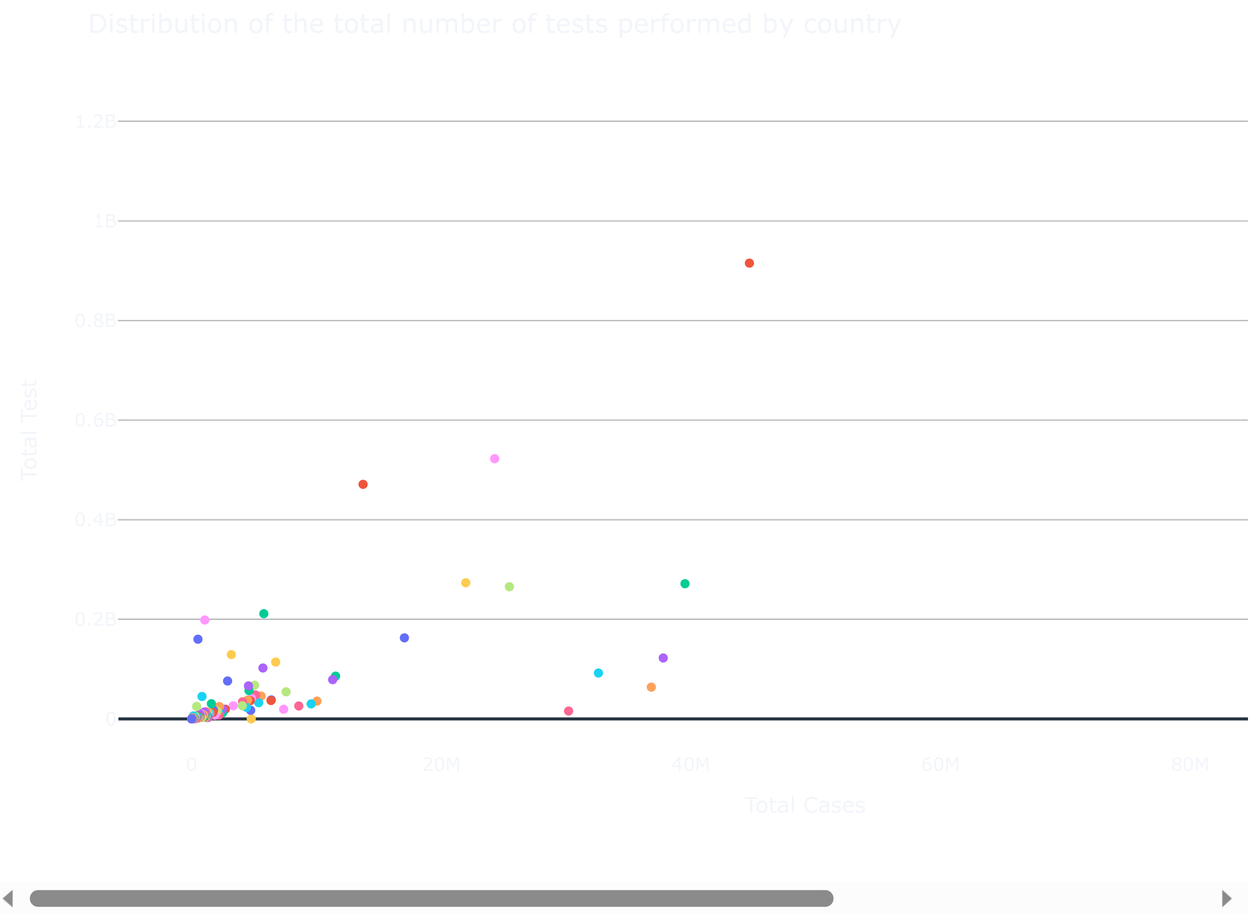
In [65]:

```
fig=px.histogram(df,x='Total Cases',y='Total Test',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of the total number of tests performed by country',width=1200
                height=600)
fig.show()
```



In [67]:

```
fig=px.violin(df,x='Total Cases',y='Total Test',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of the total number of tests performed by country',width=1200,
             height=600)
fig.show()
```

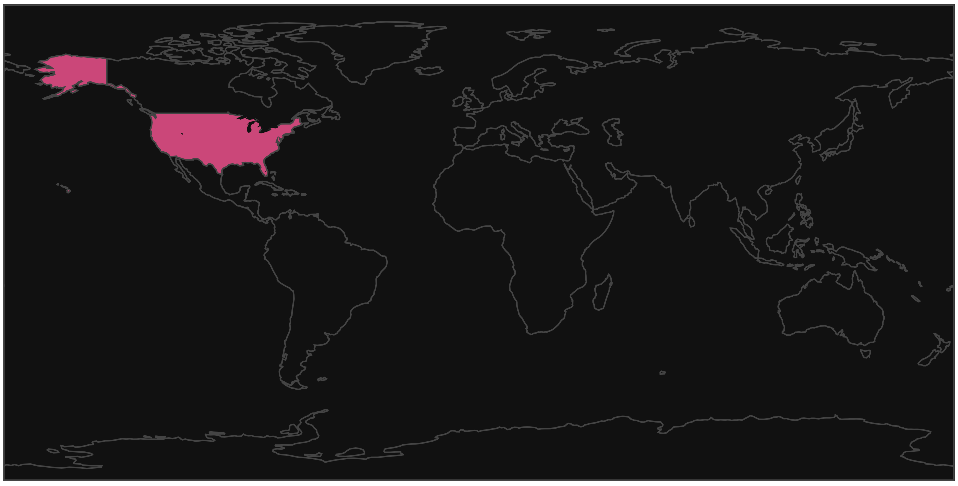


Distribution of the number of cases in countries depending on the population

In [70]:

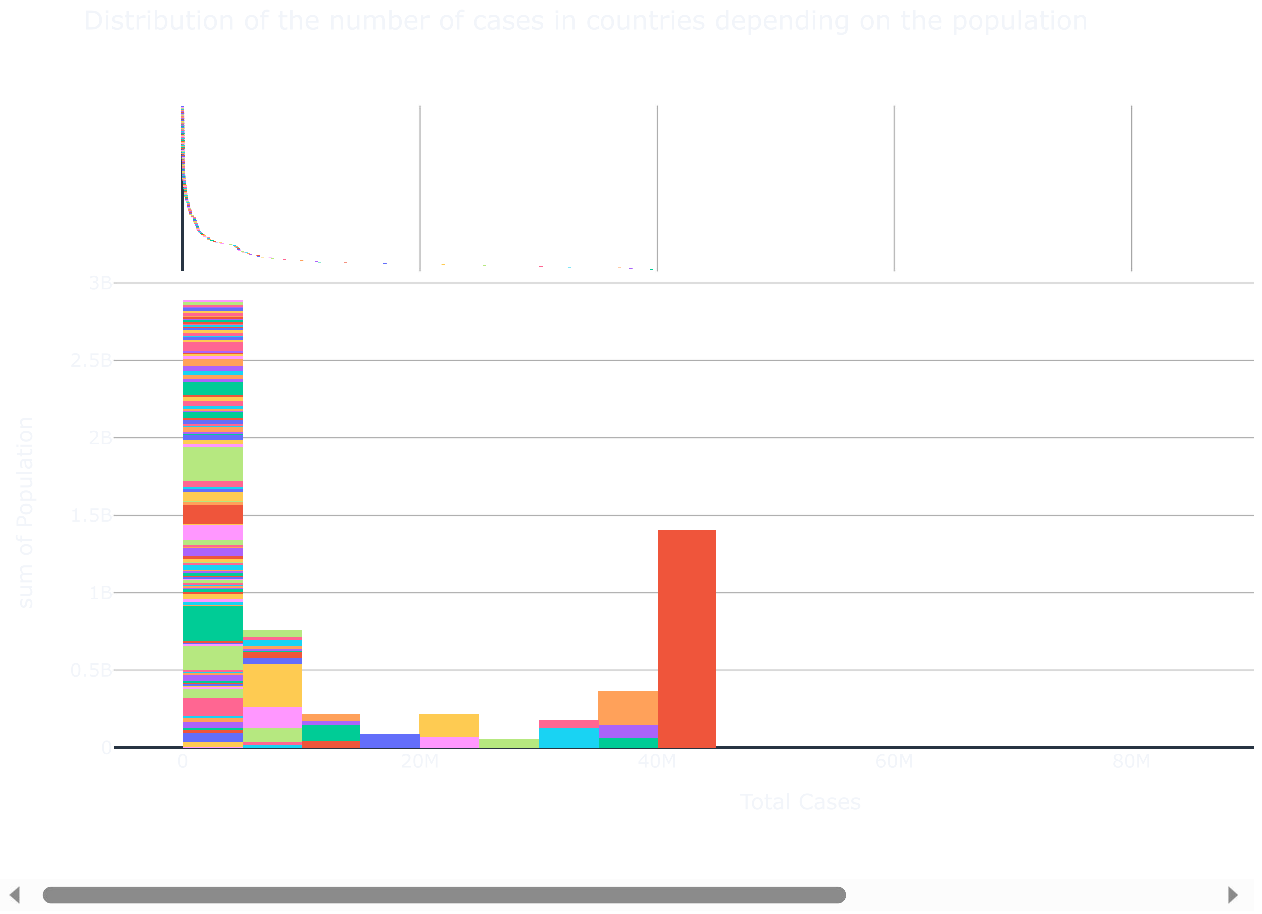
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Total Cases'],
                 animation_frame=df['Population'],animation_group=df['Total Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of cases in countries depending on the population'})
fig.show()
```

Distribution of the number of cases in countries depending on the population



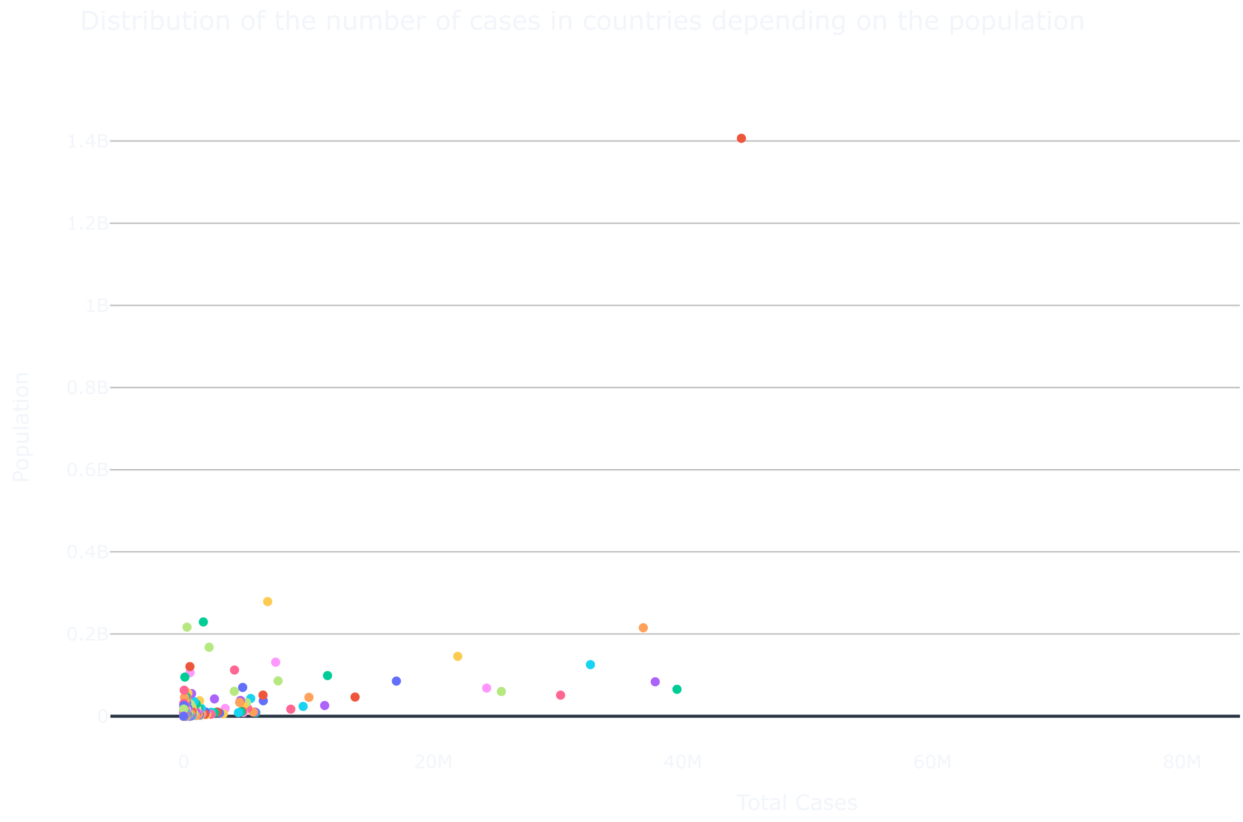
In [71]:

```
fig=px.histogram(df,x='Total Cases',y='Population',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of the number of cases in countries depending on the populati
                height=600)
fig.show()
```



In [72]:

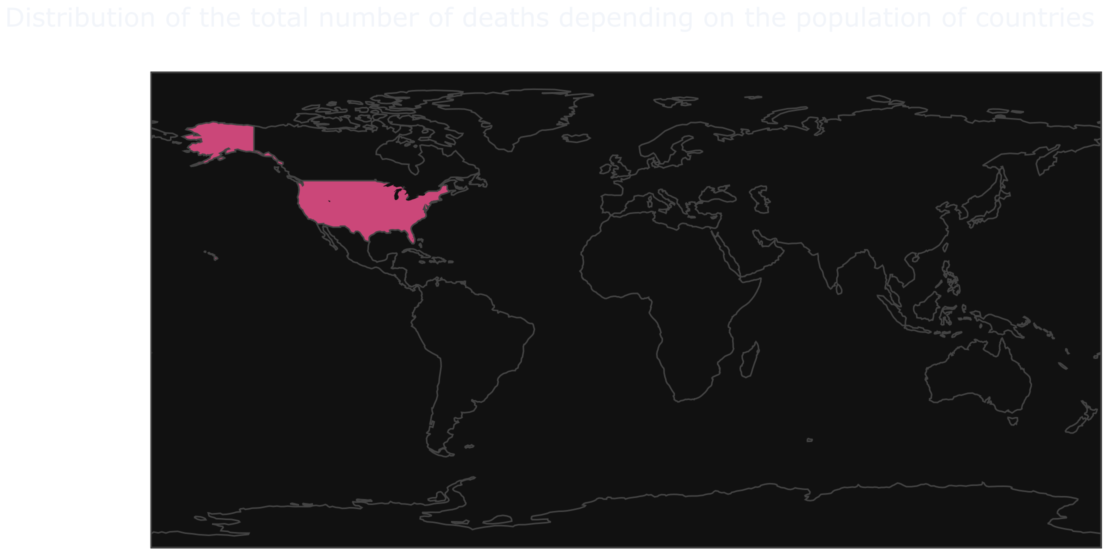
```
fig=px.violin(df,x='Total Cases',y='Population',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of the number of cases in countries depending on the population'
             height=600)
fig.show()
```



Distribution of the total number of deaths depending on the population of countries

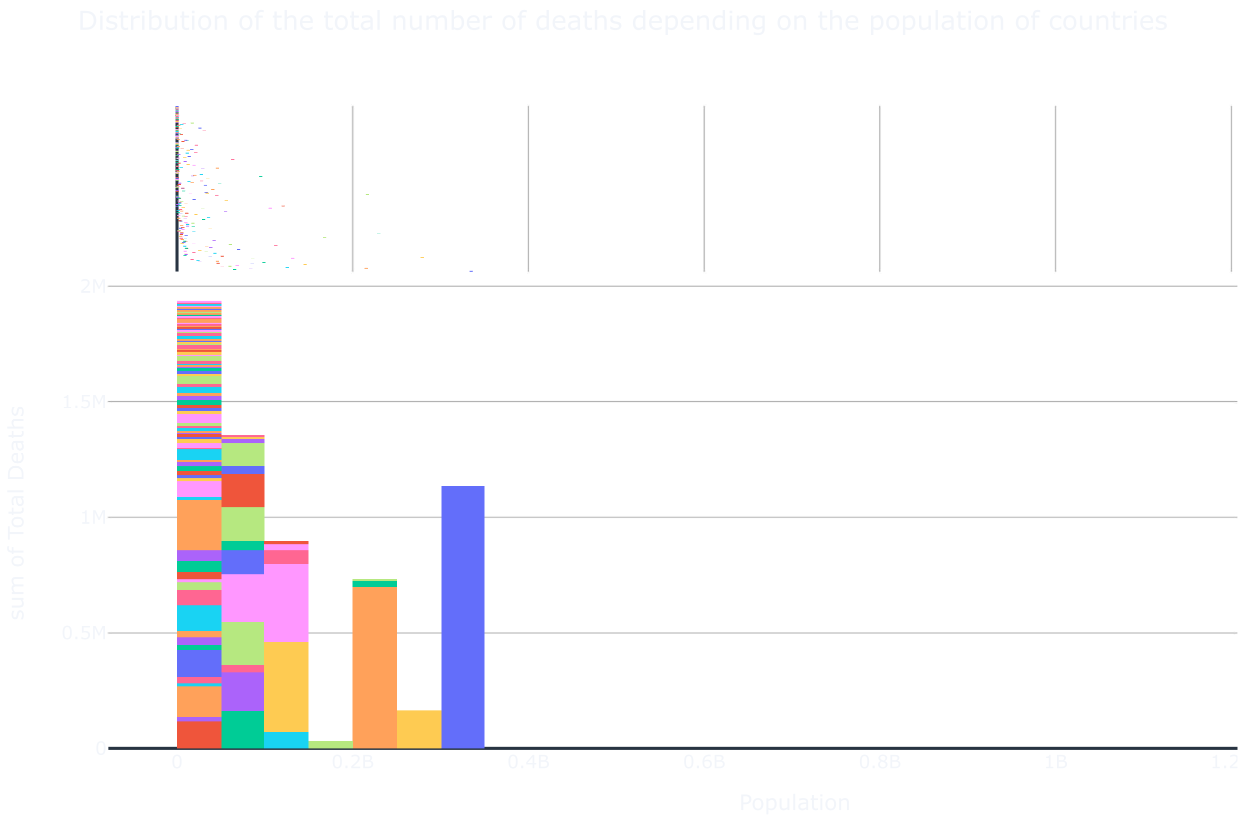
In [73]:

```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Total Deaths'],
                 animation_frame=df['Population'],animation_group=df['Total Deaths'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the total number of deaths depending on the population of countries'})
fig.show()
```



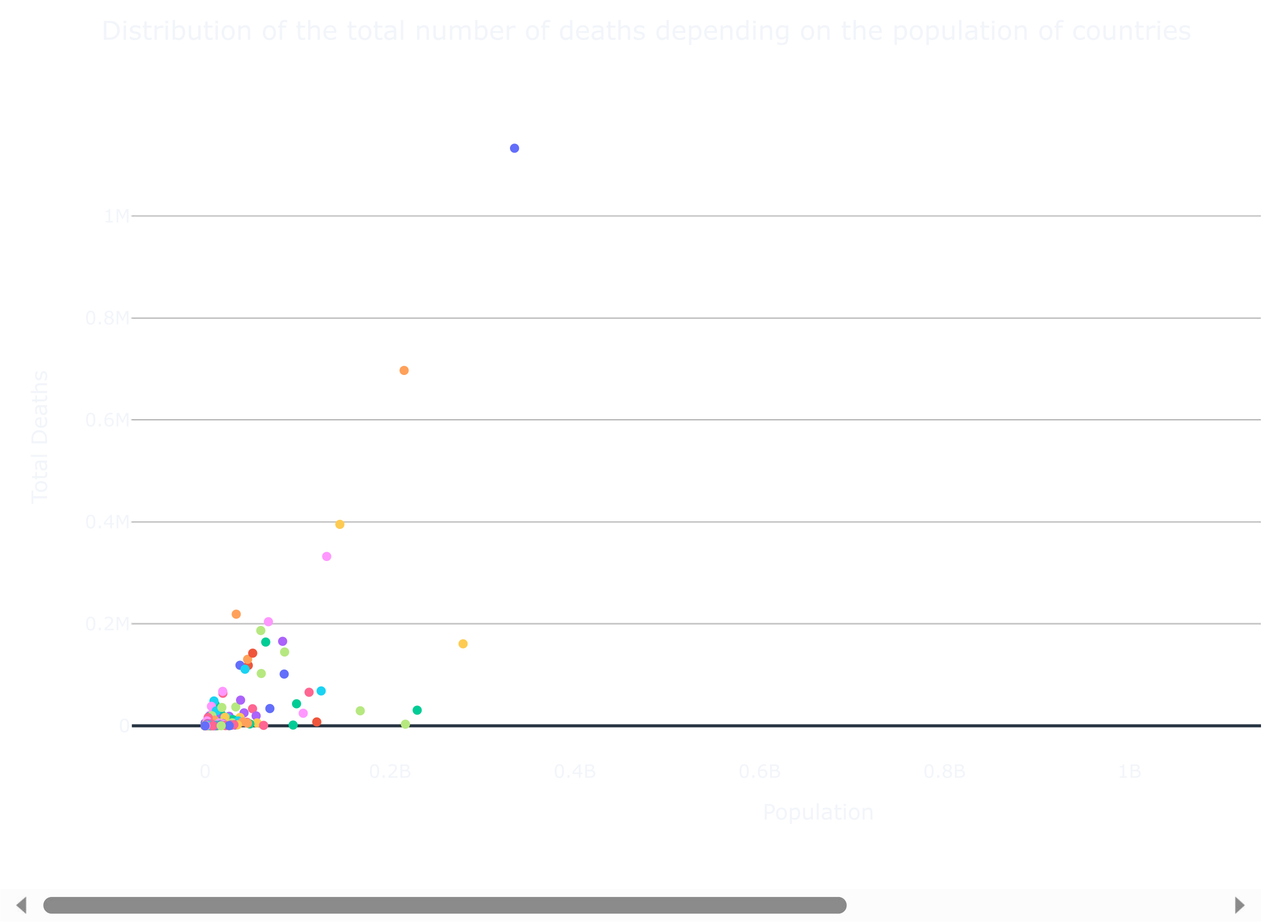
In [75]:

```
ig=px.histogram(df,x='Population',y='Total Deaths',color='Country',hover_data=df.columns,marginal='box',
               template='plotly_dark',title='Distribution of the total number of deaths depending on the population of c
               height=600)
ig.show()
```



In [76]:

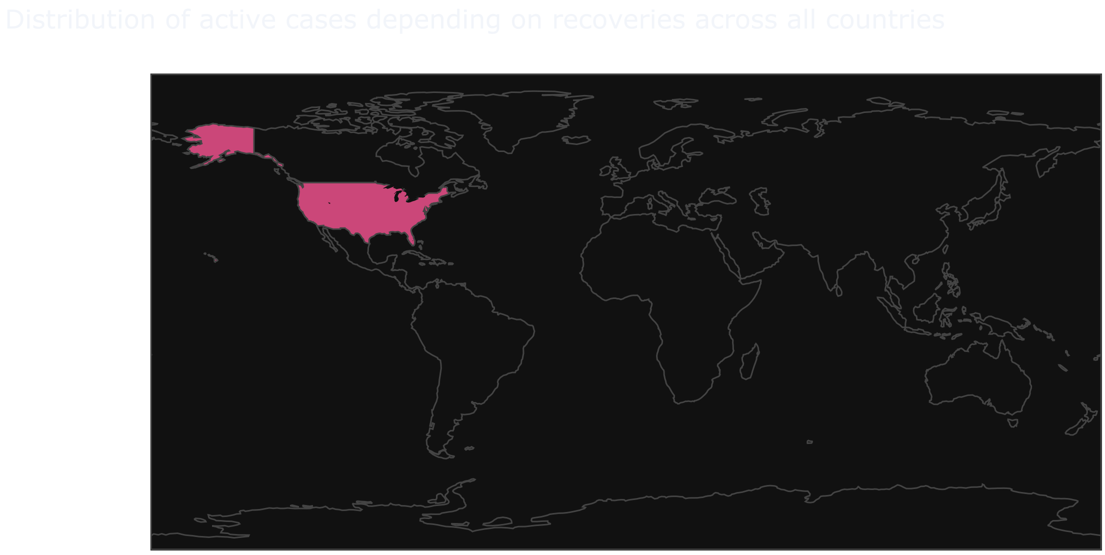
```
fig=px.violin(df,x='Population',y='Total Deaths',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of the total number of deaths depending on the population of cou
             height=600)
fig.show()
```



Distribution of active cases depending on recoveries across all countries

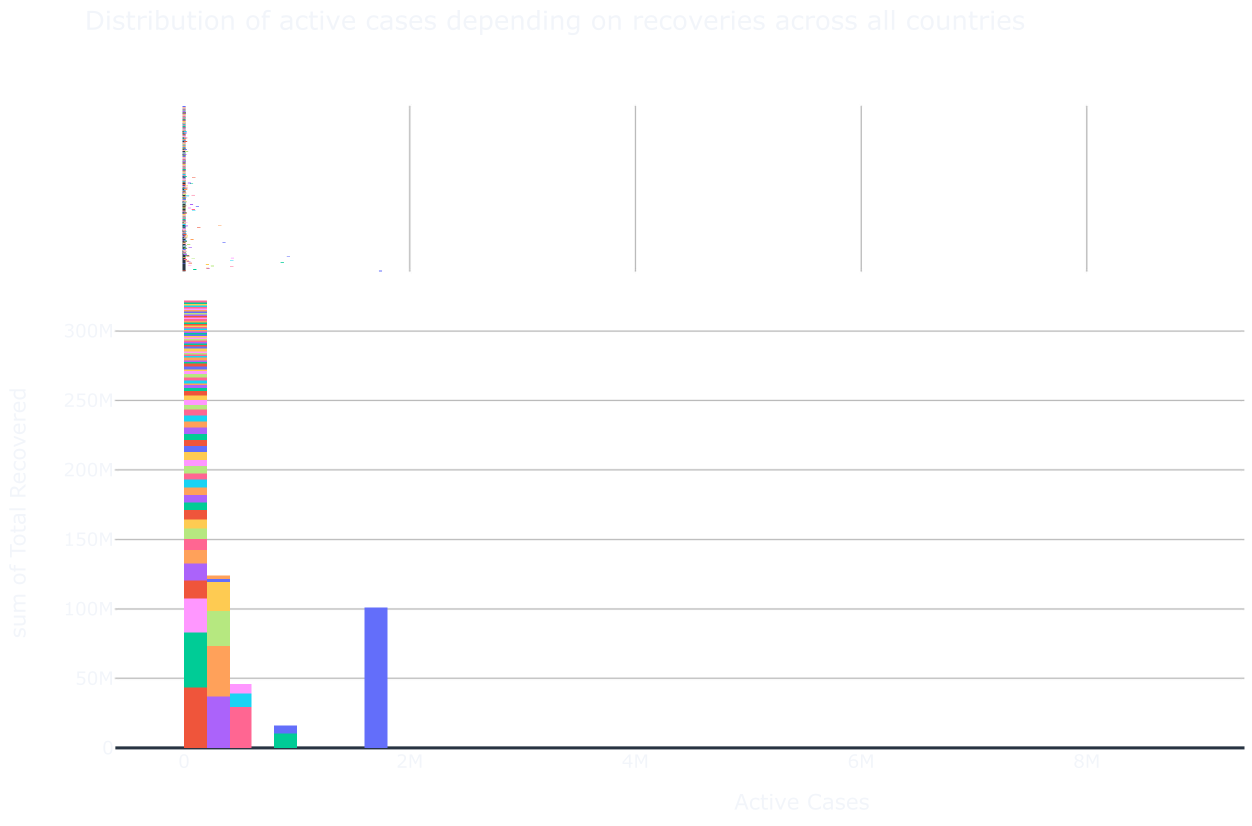
In [79]:

```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Active Cases'],
                  animation_frame=df['Total Recovered'],animation_group=df['Total Recovered'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of active cases depending on recoveries across all countries'})
fig.show()
```



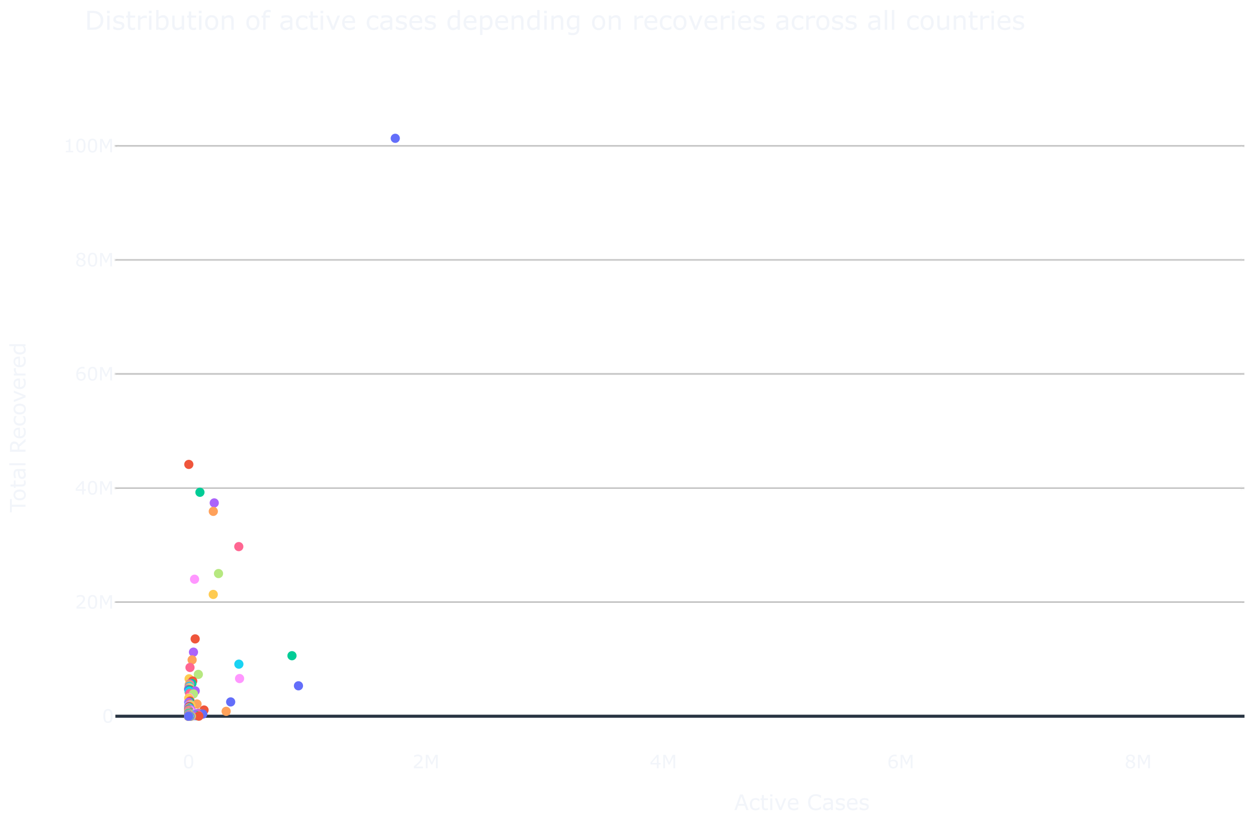
In [80]:

```
=px.histogram(df,x='Active Cases',y='Total Recovered',color='Country',hover_data=df.columns,marginal='box',
             template='plotly_dark',title='Distribution of active cases depending on recoveries across all countries',wi
             height=600)
.show()
```



In [81]:

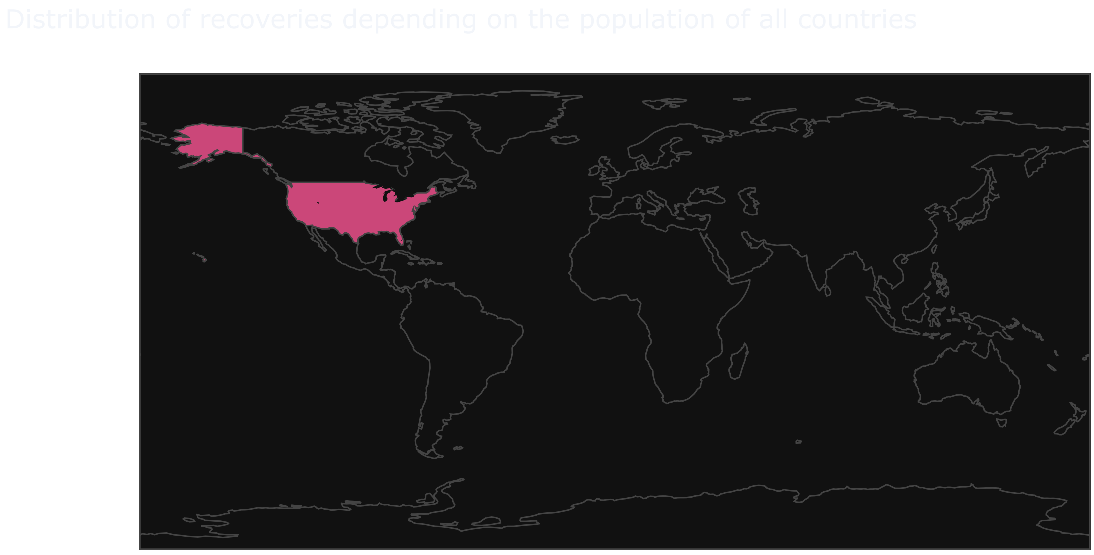
```
fig=px.violin(df,x='Active Cases',y='Total Recovered',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of active cases depending on recoveries across all countries',wi
             height=600)
fig.show()
```



Distribution of recoveries depending on the population of all countries

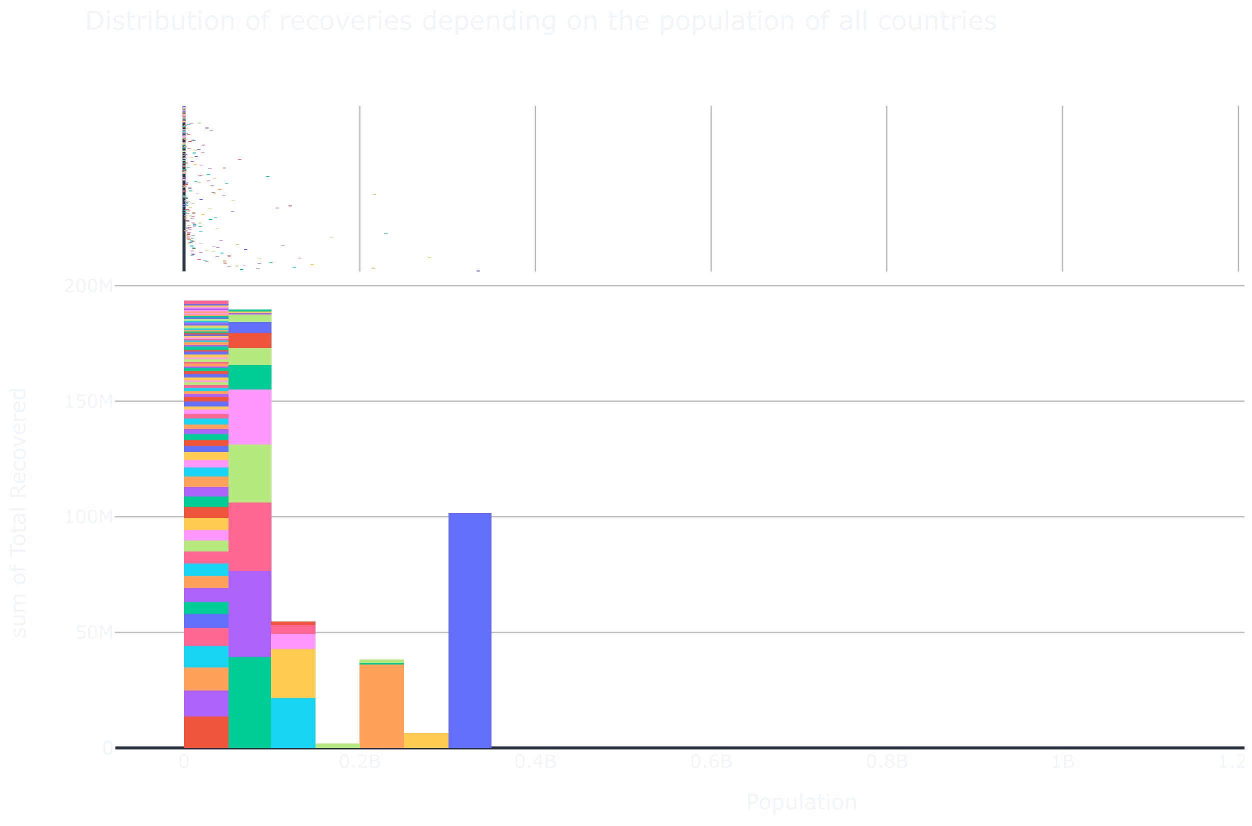
In [82]:

```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Population'],
                 animation_frame=df['Total Recovered'],animation_group=df['Total Recovered'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of recoveries depending on the population of all countries'})
fig.show()
```



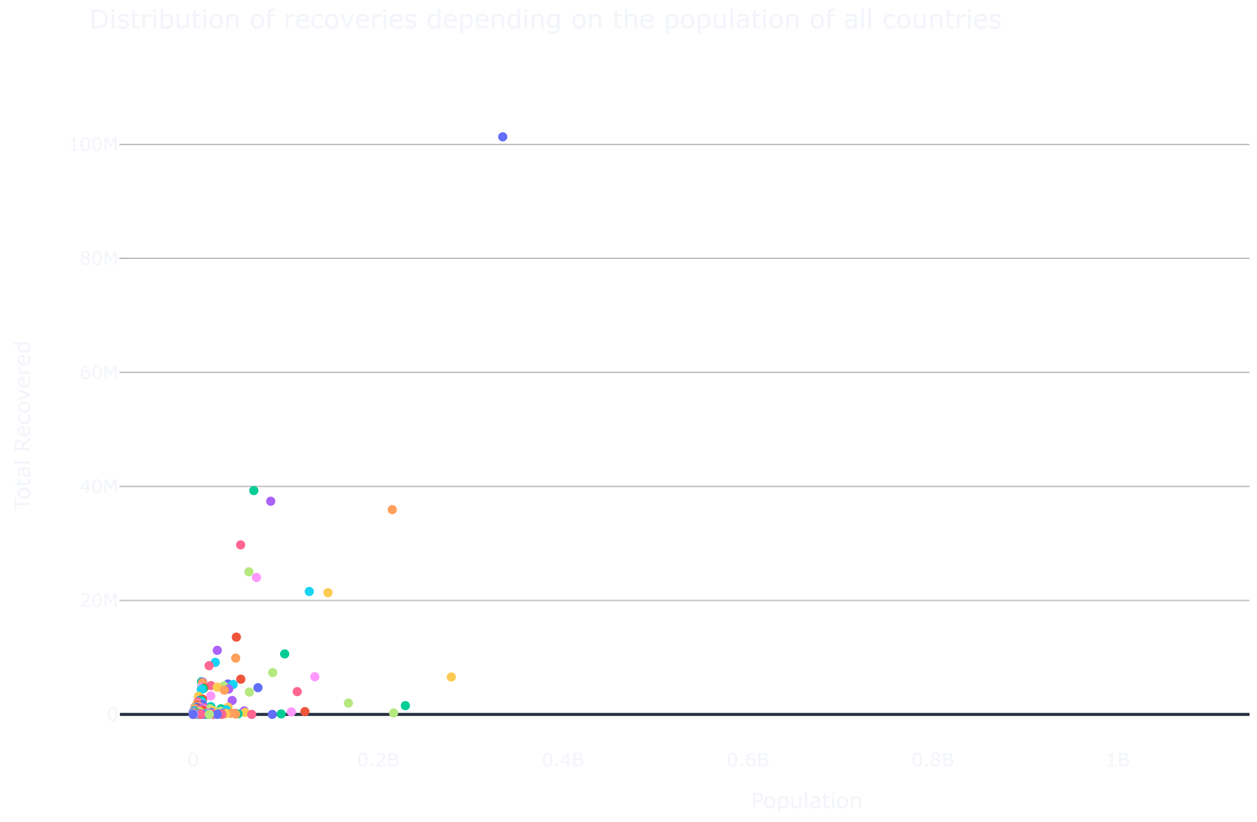
In [83]:

```
fig=px.histogram(df,x='Population',y='Total Recovered',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of recoveries depending on the population of all countries',w
                height=600)
fig.show()
```



In [84]:

```
fig=px.violin(df,x='Population',y='Total Recovered',color='Country',hover_data=df.columns,points='all',
             template='plotly_dark',title='Distribution of recoveries depending on the population of all countries',widt
             height=600)
fig.show()
```

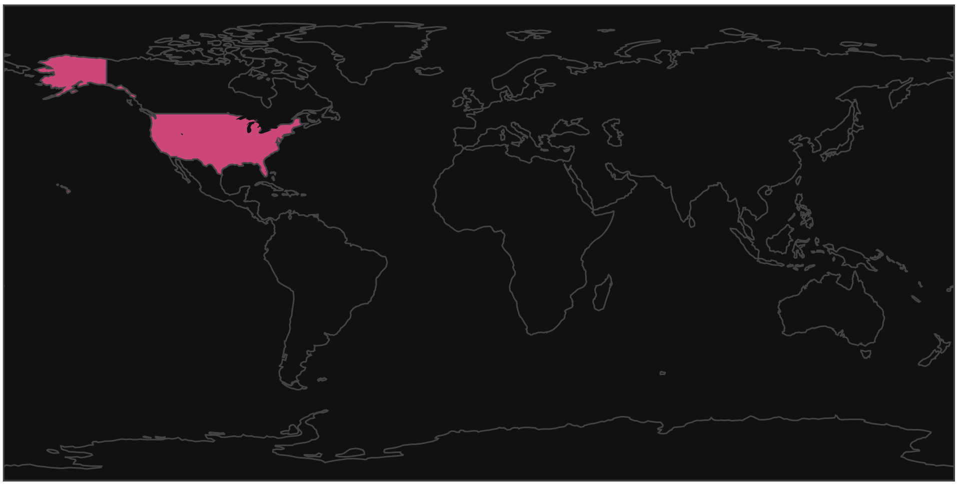


Distribution of the number of tests performed depending on the population of all countries

In [85]:

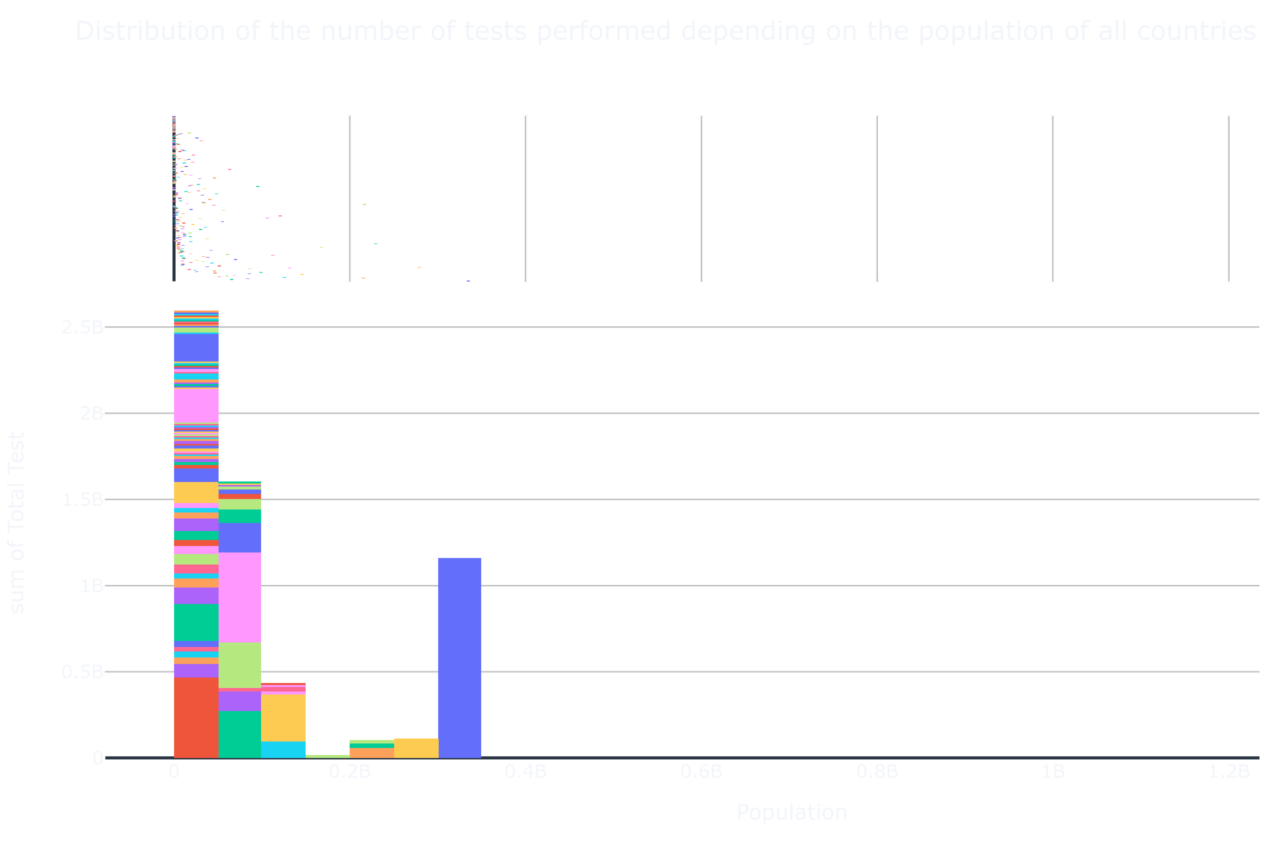
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Population'],
                  animation_frame=df['Total Test'],animation_group=df['Total Test'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of tests performed depending on the population of all count
fig.show()
```

Distribution of the number of tests performed depending on the population of all countrie



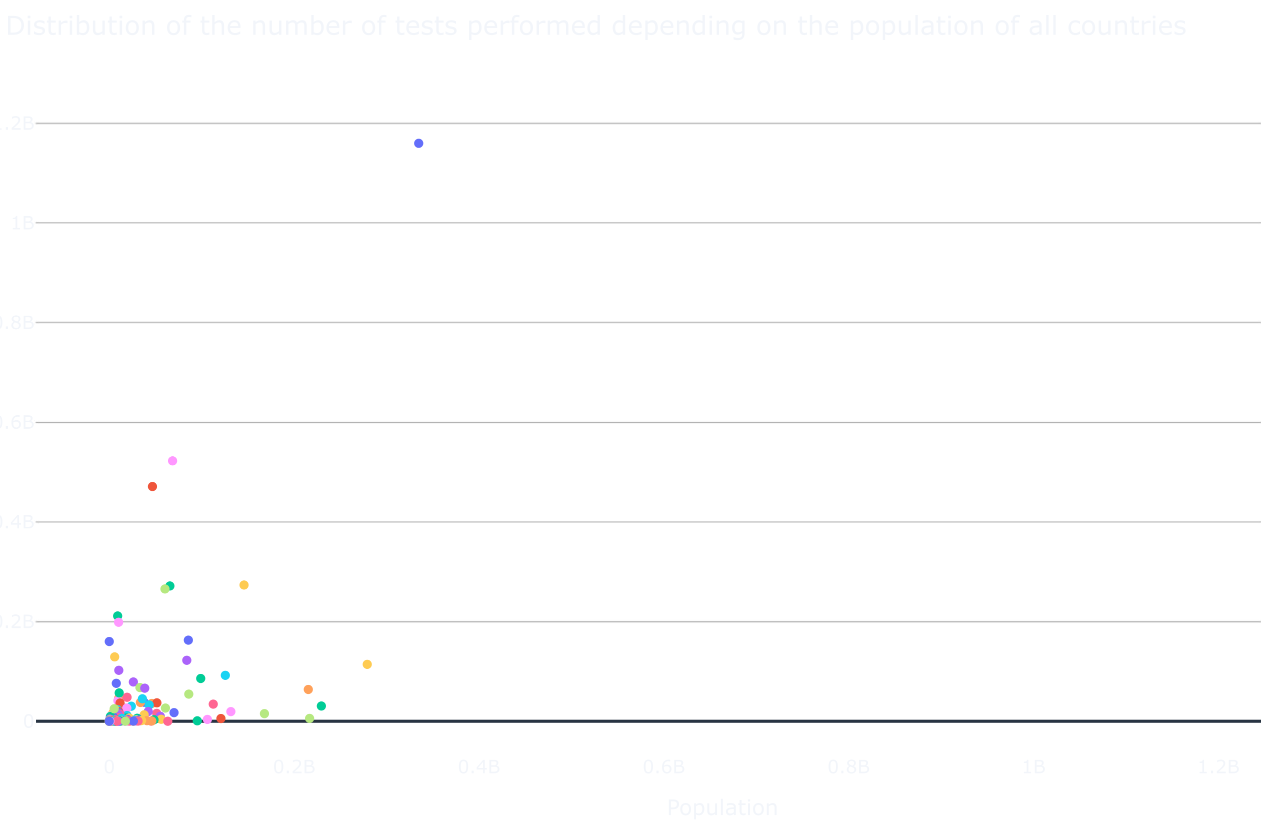
In [86]:

```
fig=px.histogram(df,x='Population',y='Total Test',color='Country',hover_data=df.columns,marginal='box',
                template='plotly_dark',title='Distribution of the number of tests performed depending on the population
                height=600)
fig.show()
```



In [87]:

```
x.violin(df,x='Population',y='Total Test',color='Country',hover_data=df.columns,points='all',
        template='plotly_dark',title='Distribution of the number of tests performed depending on the population of all co
        height=600)
how()
```

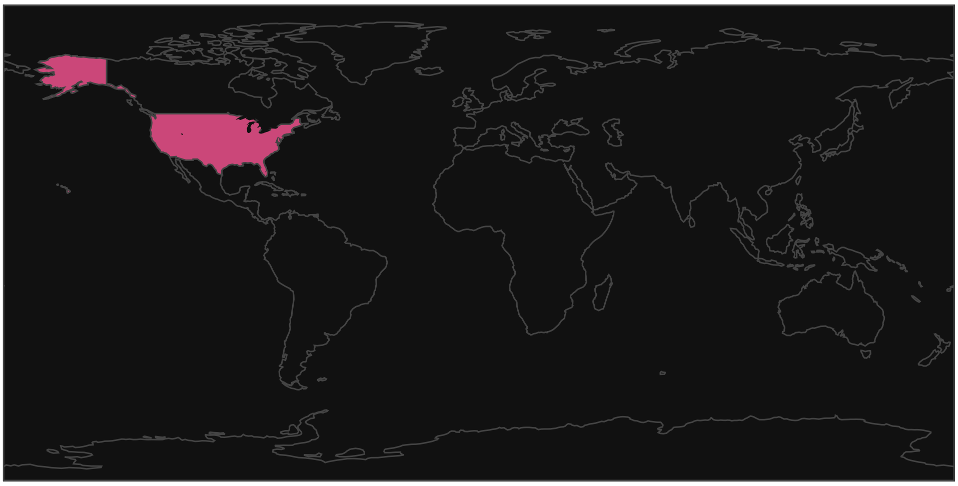


Distribution of the number of active cases depending on the population of countries

In [88]:

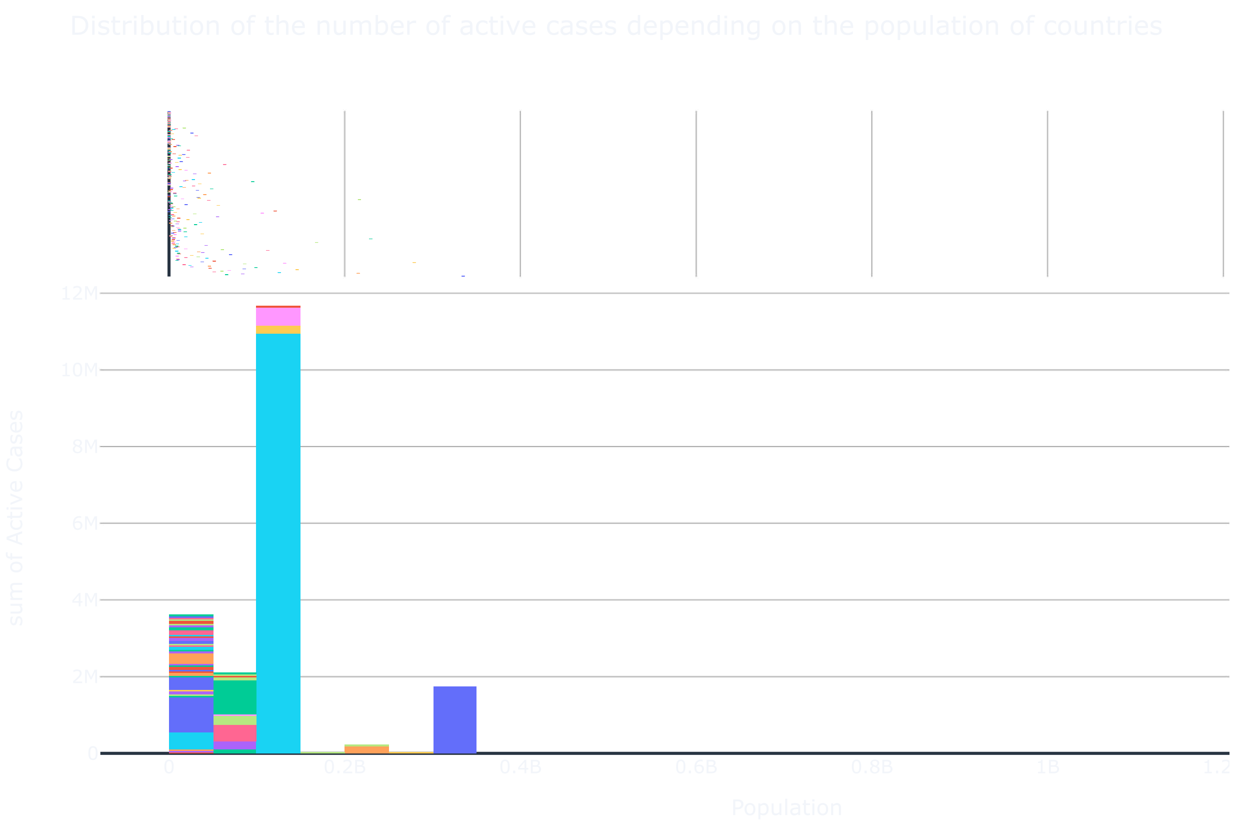
```
fig=px.choropleth(data_frame=df,locations=df['Country'],locationmode='country names',color=df['Population'],
                 animation_frame=df['Active Cases'],animation_group=df['Active Cases'],template='plotly_dark')
fig.update_layout(dict1={'title':'Distribution of the number of active cases depending on the population of countries'})
fig.show()
```

Distribution of the number of active cases depending on the population of countries



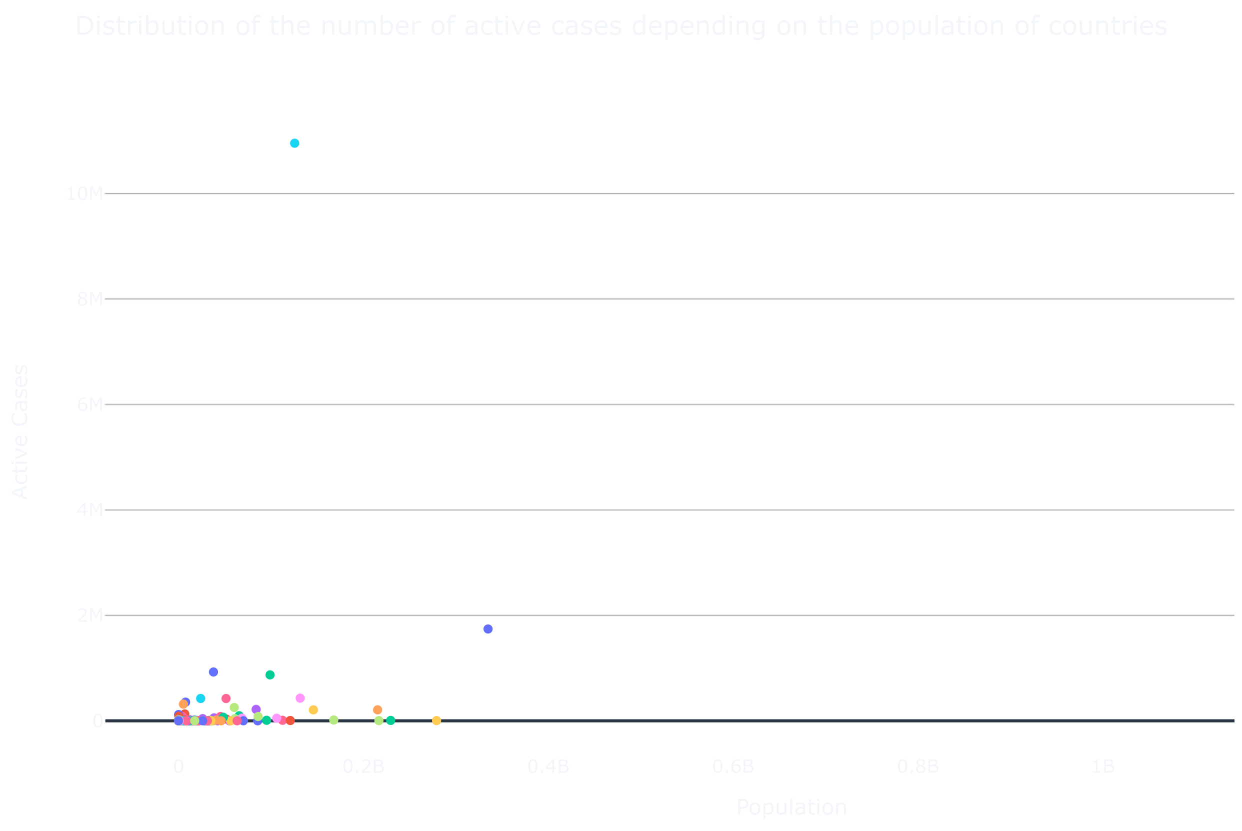
In [89]:

```
istogram(df,x='Population',y='Active Cases',color='Country',hover_data=df.columns,marginal='box',
        template='plotly_dark',title='Distribution of the number of active cases depending on the population of countrie
        height=600)
()
```



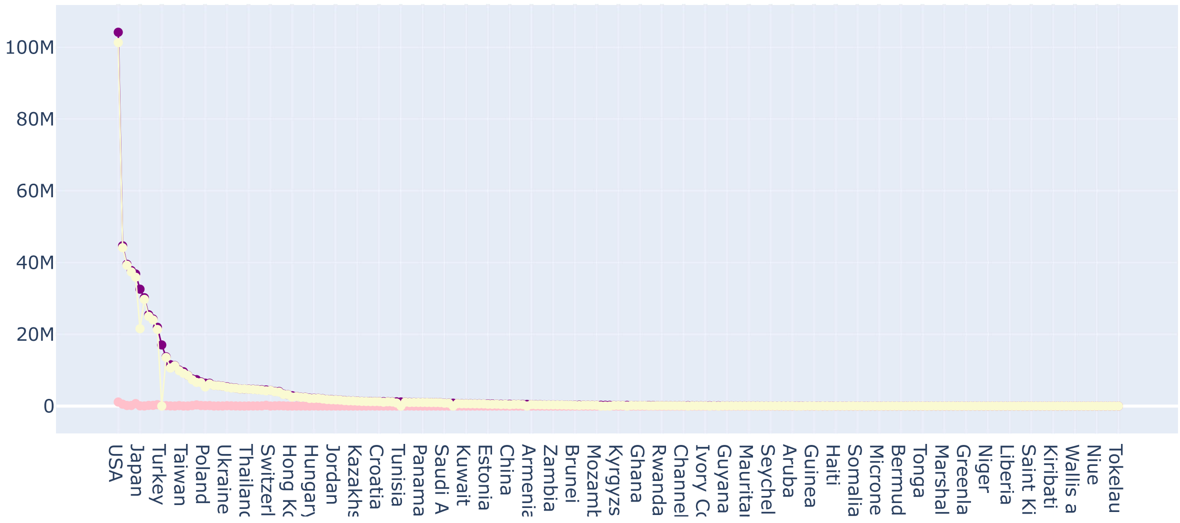
In [91]:

```
iolin(df,x='Population',y='Active Cases',color='Country',hover_data=df.columns,points='all',
     template='plotly_dark',title='Distribution of the number of active cases depending on the population of countries',
     height=600)
()
```



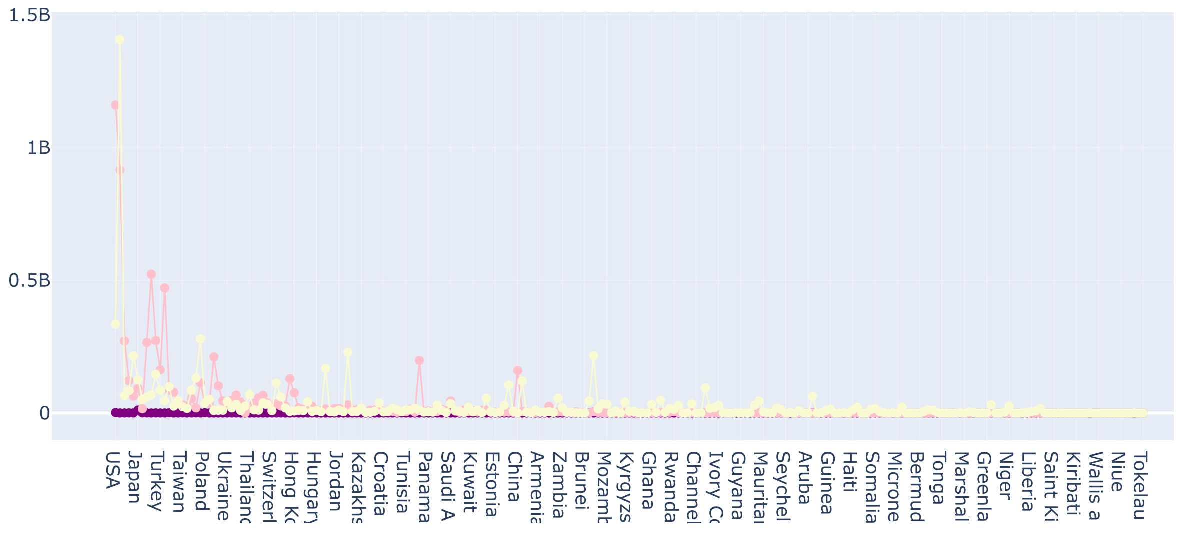
In [97]:

```
fig=go.Figure()
fig.add_trace(go.Scatter(x=df['Country'],y=df['Total Cases'],mode='lines+markers',name='Total Cases',
                          line=dict(color='purple',width=1)))
fig.add_trace(go.Scatter(x=df['Country'],y=df['Total Deaths'],mode='lines+markers',name='Total Deaths',
                        line=dict(color='pink',width=1)))
fig.add_trace(go.Scatter(x=df['Country'],y=df['Total Recovered'],mode='lines+markers',name='Total Recovered',
                        line=dict(color='lightgoldenrodyellow',width=1)))
fig.show()
```



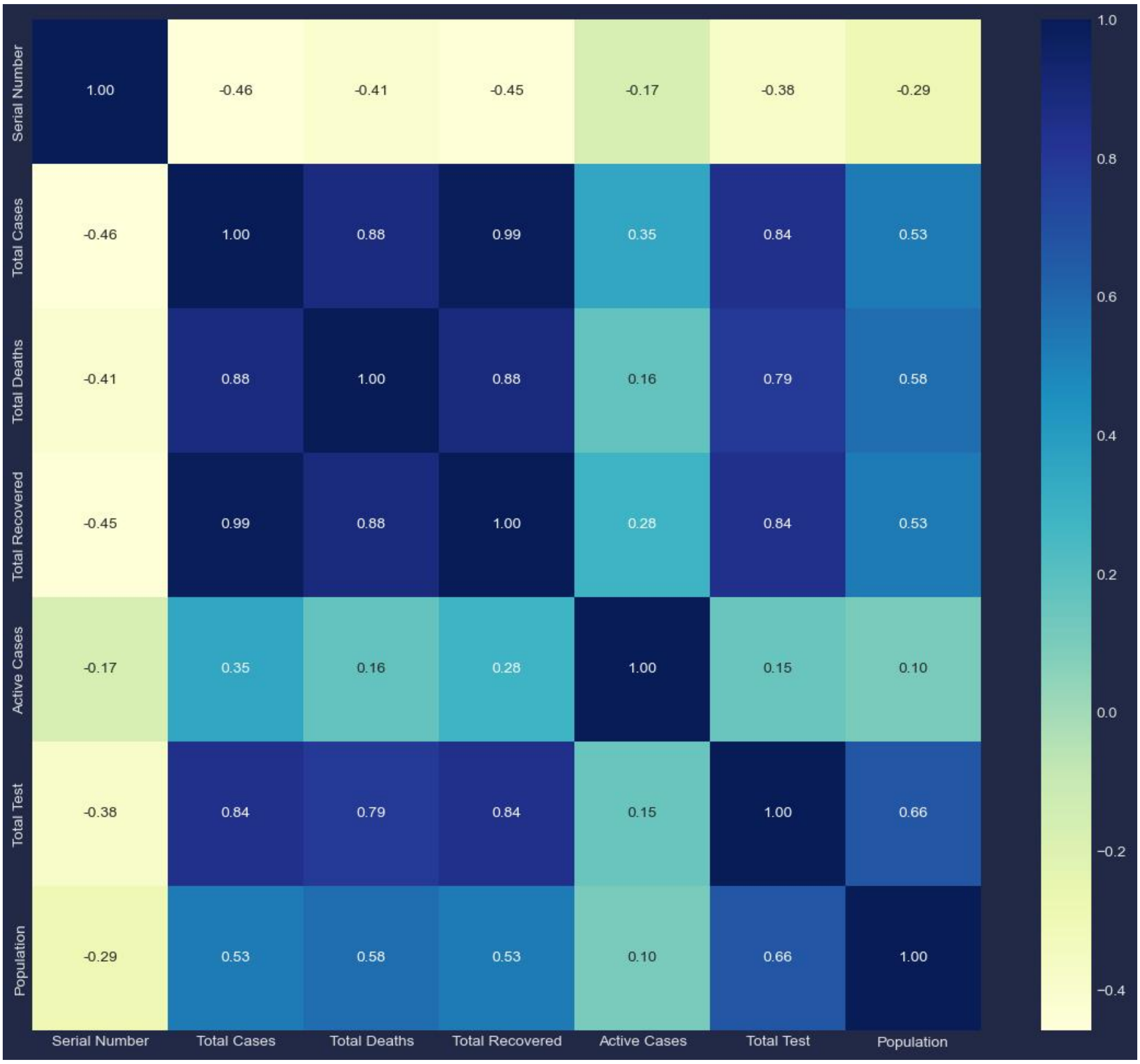
In [98]:

```
fig=go.Figure()
fig.add_trace(go.Scatter(x=df['Country'],y=df['Active Cases'],mode='lines+markers',name='Active Cases',
                          line=dict(color='purple',width=1)))
fig.add_trace(go.Scatter(x=df['Country'],y=df['Total Test'],mode='lines+markers',name='Total Test',
                        line=dict(color='pink',width=1)))
fig.add_trace(go.Scatter(x=df['Country'],y=df['Population'],mode='lines+markers',name='Population',
                        line=dict(color='lightgoldenrodyellow',width=1)))
fig.show()
```



In [99]:

```
plt.figure(figsize=(14,12))
sns.heatmap(df.corr(),annot=True,cmap="YlGnBu",fmt='.2f')
plt.show()
```



Cluster

In [100]:

```
df=df.drop(['Serial Number'],axis=1).set_index('Country')
```

In [101]:

```
df.head()
```

Out[101]:

	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
Country						
USA	104196861.0	1132935.0	101322779.0	1741147.0	1.159833e+09	3.348053e+08
India	44682784.0	530740.0	44150289.0	1755.0	9.152658e+08	1.406632e+09
France	39524311.0	164233.0	39264546.0	95532.0	2.714902e+08	6.558452e+07
Germany	37779833.0	165711.0	37398100.0	216022.0	1.223324e+08	8.388360e+07
Brazil	36824580.0	697074.0	35919372.0	208134.0	6.377617e+07	2.153536e+08

In [102]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [103]:

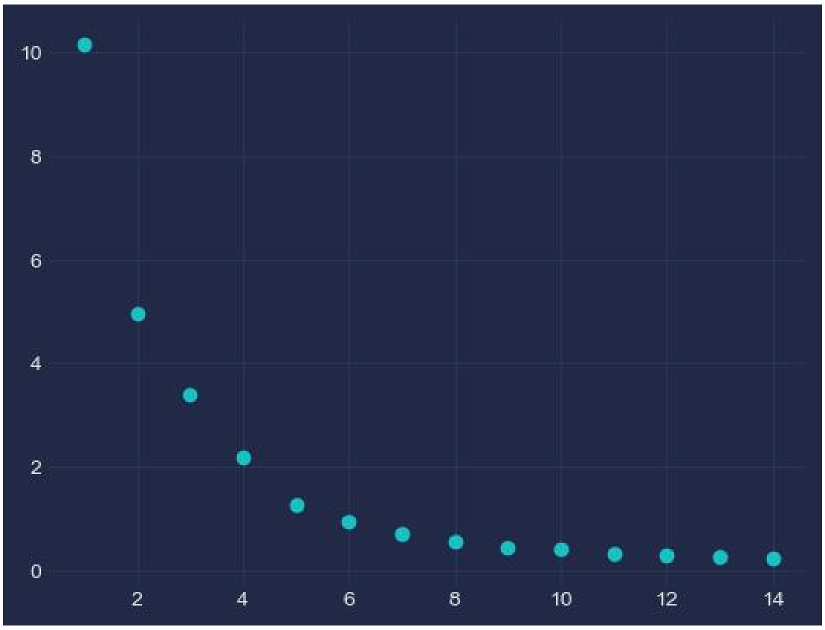
```
scaler=MinMaxScaler()
names=df.columns
d=scaler.fit_transform(df)
scaled_df=pd.DataFrame(d,columns=names)
scaled_df.head()
```

Out[103]:

	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1.000000	1.000000	1.000000	0.158971	1.000000	0.238019
1	0.428830	0.468465	0.435739	0.000160	0.789136	1.000000
2	0.379323	0.144962	0.387519	0.008722	0.234077	0.046625
3	0.362581	0.146267	0.369099	0.019723	0.105474	0.059634
4	0.353413	0.615282	0.354504	0.019003	0.054987	0.153099

In [105]:

```
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,15):
    km=KMeans(n_clusters=i)
    km.fit_predict(scaled_df)
    wcss.append(km.inertia_)
plt.scatter(range(1,15),wcss)
plt.show()
```



In [ ]: