In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import plotly.express as px
```

In [ ]:
```
data = pd.read_csv('data.csv')
data.head()
```

Out[ ]:

	iso_code	location	date	total_cases	total_deaths	stringency_index	population	gdp_per_capi
0	AFG	Afghanistan	2019-12-31	0.0	0.0	0.0	38928341	1803.9
1	AFG	Afghanistan	2020-01-01	0.0	0.0	0.0	38928341	1803.9
2	AFG	Afghanistan	2020-01-02	0.0	0.0	0.0	38928341	1803.9
3	AFG	Afghanistan	2020-01-03	0.0	0.0	0.0	38928341	1803.9
4	AFG	Afghanistan	2020-01-04	0.0	0.0	0.0	38928341	1803.9

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50418 entries, 0 to 50417
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   iso_code                 50418 non-null  object
 1   location                 50418 non-null  object
 2   date                     50418 non-null  object
 3   total_cases              47324 non-null  float64
 4   total_deaths             39228 non-null  float64
 5   stringency_index         43292 non-null  float64
 6   population               50418 non-null  int64
 7   gdp_per_capita           44706 non-null  float64
 8   human_development_index  44216 non-null  float64
 9   Unnamed: 9               50418 non-null  object
 10  Unnamed: 10              50418 non-null  object
 11  Unnamed: 11              50418 non-null  object
 12  Unnamed: 12              50418 non-null  float64
 13  Unnamed: 13              50418 non-null  object
dtypes: float64(6), int64(1), object(7)
memory usage: 5.4+ MB
```

In [ ]:
```
data.describe()
```

Out[ ]:

	total_cases	total_deaths	stringency_index	population	gdp_per_capita	human_develop
count	4.732400e+04	39228.000000	43292.000000	5.041800e+04	44706.000000	4
mean	6.621927e+04	2978.767819	56.162022	4.251601e+07	20818.706240	

	total_cases	total_deaths	stringency_index	population	gdp_per_capita	human_develop
std	4.045582e+05	13836.644013	27.532685	1.564607e+08	20441.365392	
min	0.000000e+00	0.000000	0.000000	8.090000e+02	661.240000	
25%	1.260000e+02	10.000000	37.960000	1.399491e+06	5338.454000	
50%	1.594000e+03	64.000000	61.110000	8.278737e+06	13913.839000	
75%	1.584775e+04	564.000000	78.700000	2.913681e+07	31400.840000	
max	8.154595e+06	219674.000000	100.000000	1.439324e+09	116935.600000	

In [ ]:
```
data.shape
```

Out[ ]:
```
(50418, 14)
```

Null Values

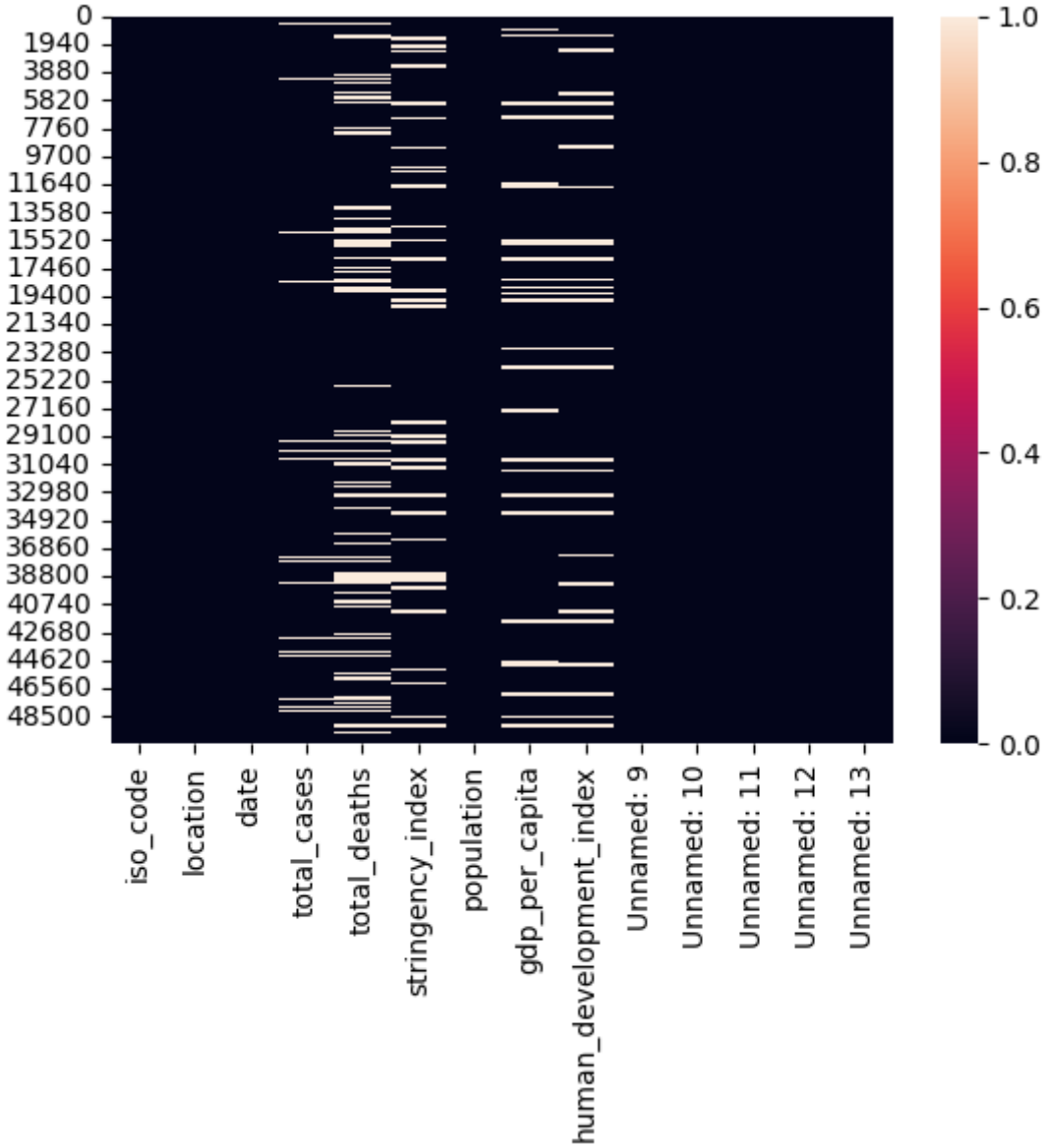
Checking Null Values

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
iso_code                    0
location                    0
date                        0
total_cases              3094
total_deaths            11190
stringency_index         7126
population                  0
gdp_per_capita           5712
human_development_index  6202
Unnamed: 9                  0
Unnamed: 10                 0
Unnamed: 11                 0
Unnamed: 12                 0
Unnamed: 13                 0
dtype: int64
```

In [ ]:
```
sns.heatmap(data.isnull())
```

Out[ ]:
```
<Axes: >
```



Dropping Null Values

In [ ]:
```
print("With Null Values : ", data.shape)
data = data.dropna()
```

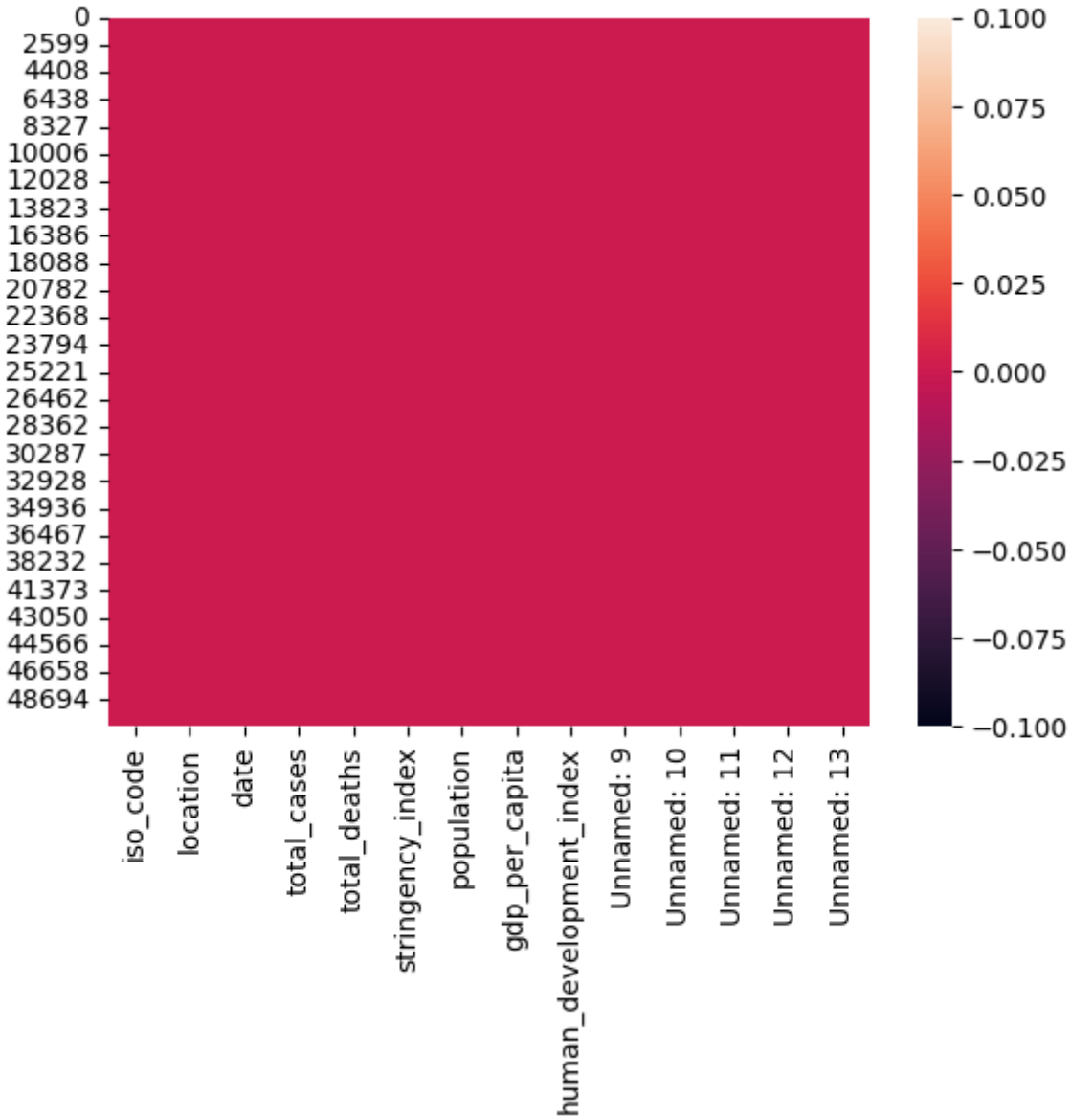
```
With Null Values :  (50418, 14)
```

In [ ]:
```
print("Without Null Values : ", data.shape)
```

```
Without Null Values :  (31518, 14)
```

In [ ]:
```
sns.heatmap(data.isnull())
```

Out[ ]:
```
<Axes: >
```



Shortening the data for proper Exploratory Data Analysis Report with Visualization

In [ ]:
```
data = data.iloc[0:7000,:]
```

Duplicate Values

Dropping Duplicate Values

In [ ]:
```
# data = data.drop_duplicates()
```

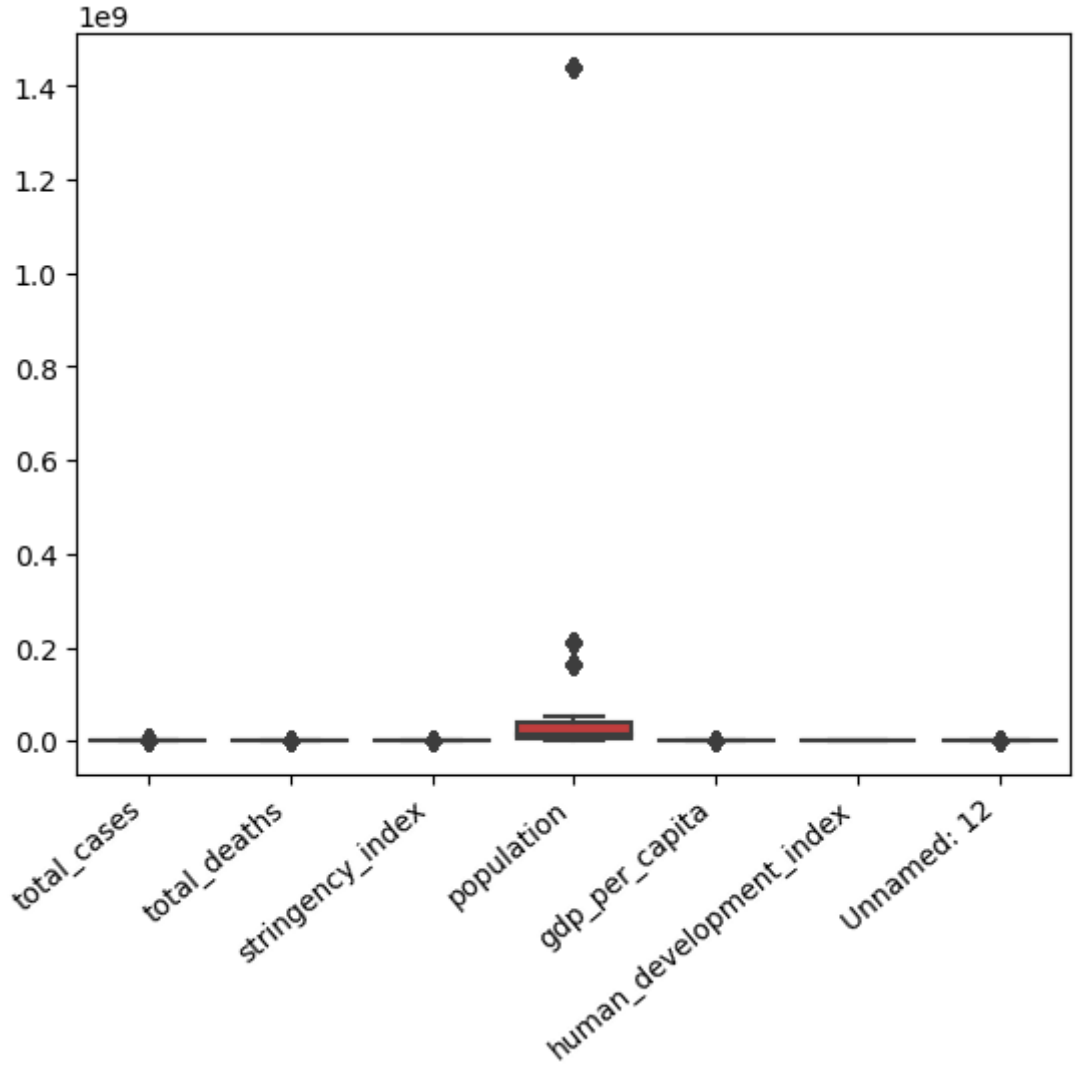
In [ ]:
```
# data.shape
```

Outliers

Checking Outliers

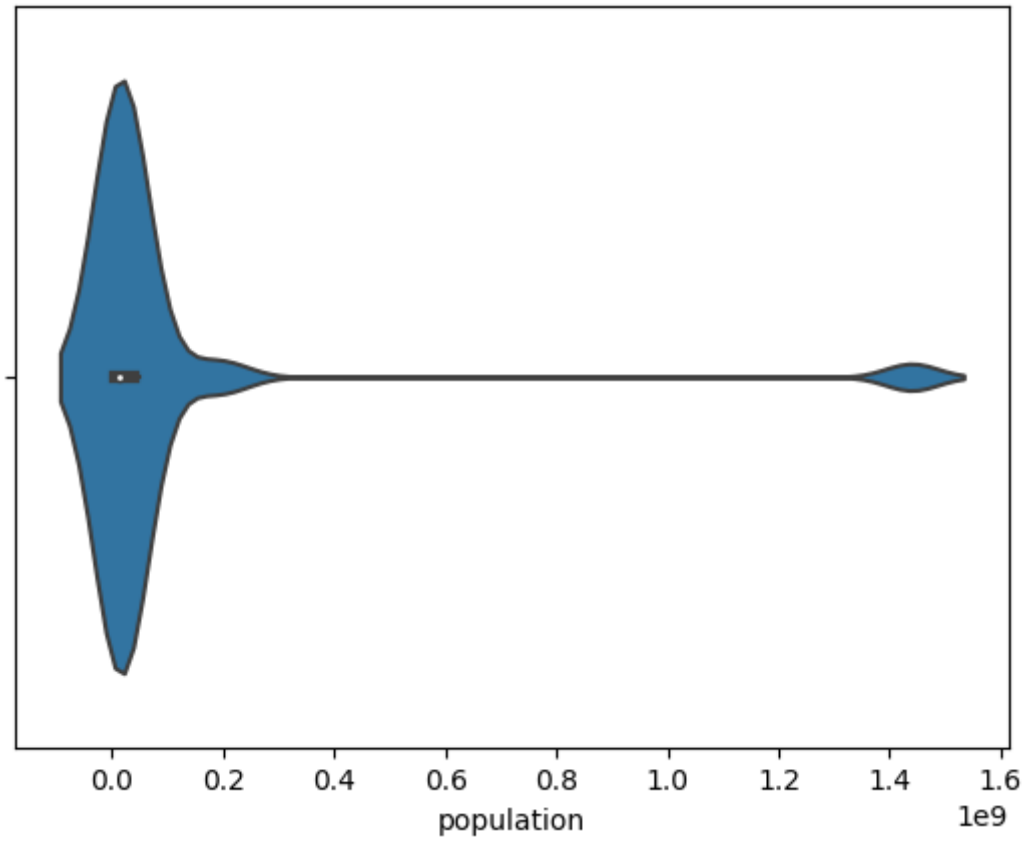
In [ ]:
```
ax = sns.boxplot(data=data)
ax.set_xticklabels(ax.get_xticklabels(),rotation = 40, ha = 'right')
```

Out[ ]:
```
[Text(0, 0, 'total_cases'),
 Text(1, 0, 'total_deaths'),
 Text(2, 0, 'stringency_index'),
 Text(3, 0, 'population'),
 Text(4, 0, 'gdp_per_capita'),
 Text(5, 0, 'human_development_index'),
 Text(6, 0, 'Unnamed: 12')]
```



In [ ]:
```
sns.violinplot(data=data,x = 'population')
```

Out[ ]:
```
<Axes: xlabel='population'>
```



Solving Outliers

In [ ]:

```
# Quantiles :
Q1 = data['population'].quantile(0.25)
Q3 = data['population'].quantile(0.75)
IQR = Q3 - Q1
print("Quantile 1 : ", Q1)
print("Quantile 3 : ", Q3)
print("IQR : ", IQR)

# Upper Quantile :
upper = Q3+1.5*IQR
print("Upper Quantile : ",upper)
# Lower Quantile :
lower = Q1-1.5*IQR
print("Lower Quantile : ",lower)
```

```
Quantile 1 :  4829764.0
Quantile 3 :  37742157.0
IQR :  32912393.0
Upper Quantile :  87110746.5
Lower Quantile :  -44538825.5
```

In [ ]:

```
upper_arr = np.where(data['population']>upper)[0]
lower_arr = np.where(data['population']<lower)[0]

print("Before Removing Outliers : ", data.shape)
# data = data.drop(index=upper_arr)
data = data.drop(index=lower_arr)
print("After Removing Outliers : ", data.shape )
```

```
Before Removing Outliers :  (7000, 14)
After Removing Outliers :  (7000, 14)
```

No Impact of Outliers Found

Skewness

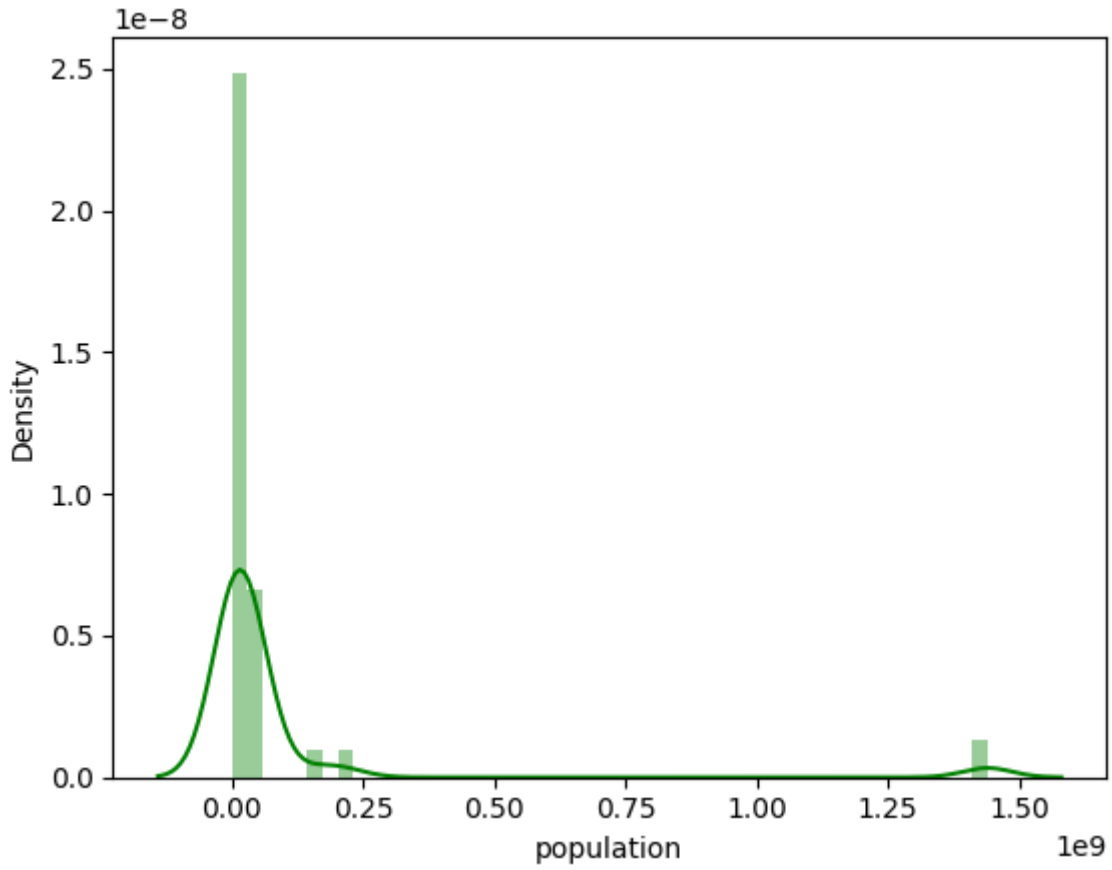
In [ ]:
```
skewness = data['population'].skew()
print(skewness)
```

```
4.614490840276978
```

In [ ]:
```
sns.distplot(data['population'],color='g')
```

```
C:\Users\Hunain\AppData\Local\Temp\ipykernel_12352\3399190770.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['population'],color='g')
```

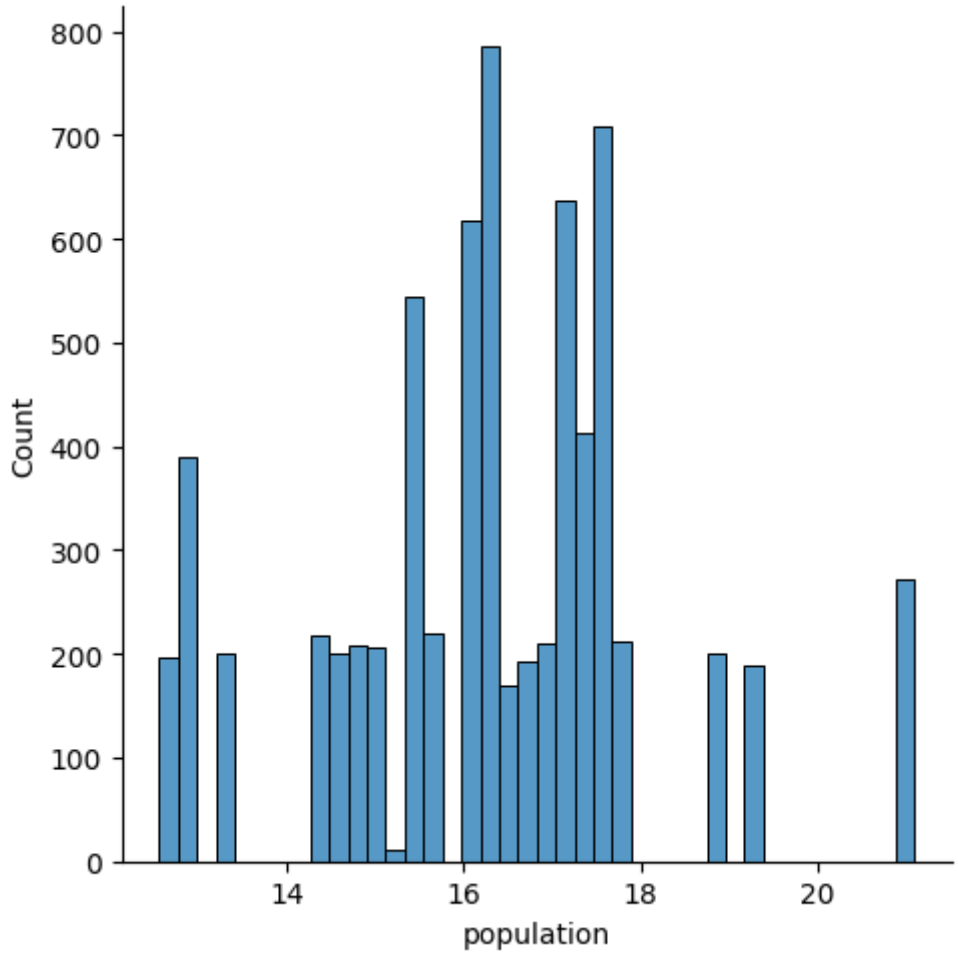
Out[ ]:
```
<Axes: xlabel='population', ylabel='Density'>
```



Normalize the Distribution

In [ ]:
```
log_y = np.log1p(data['population'])
sns.displot(log_y)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1aef98707d0>
```



In [ ]:
```
log_y.skew()
```

Out[ ]: 0.13195669059368823

Log1p Second time to Again and further normalize the distribution

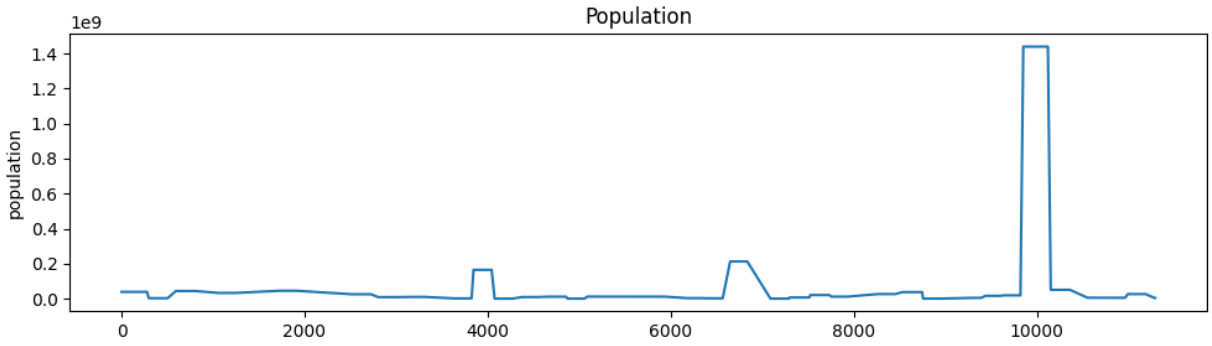
In [ ]:
```
log_y2 = np.log1p(log_y)
log_y2.skew()
```

Out[ ]: -0.25273868600002475

In [ ]:
```
log_y3 = np.log1p(log_y)
log_y3.skew()
# Same Skewness as upper shows that it is skewed till its last.
```

Out[ ]: -0.25273868600002475

In [ ]:
```
plt.figure(figsize=(10, 3))
data['population'].plot()
plt.ylabel('population')
plt.xlabel(None)
plt.title("Population")
plt.tight_layout()
```



to get years only

In [ ]:
```
data['date'] = pd.to_datetime(data['date'])
data['Year'] = data['date'].dt.year
```

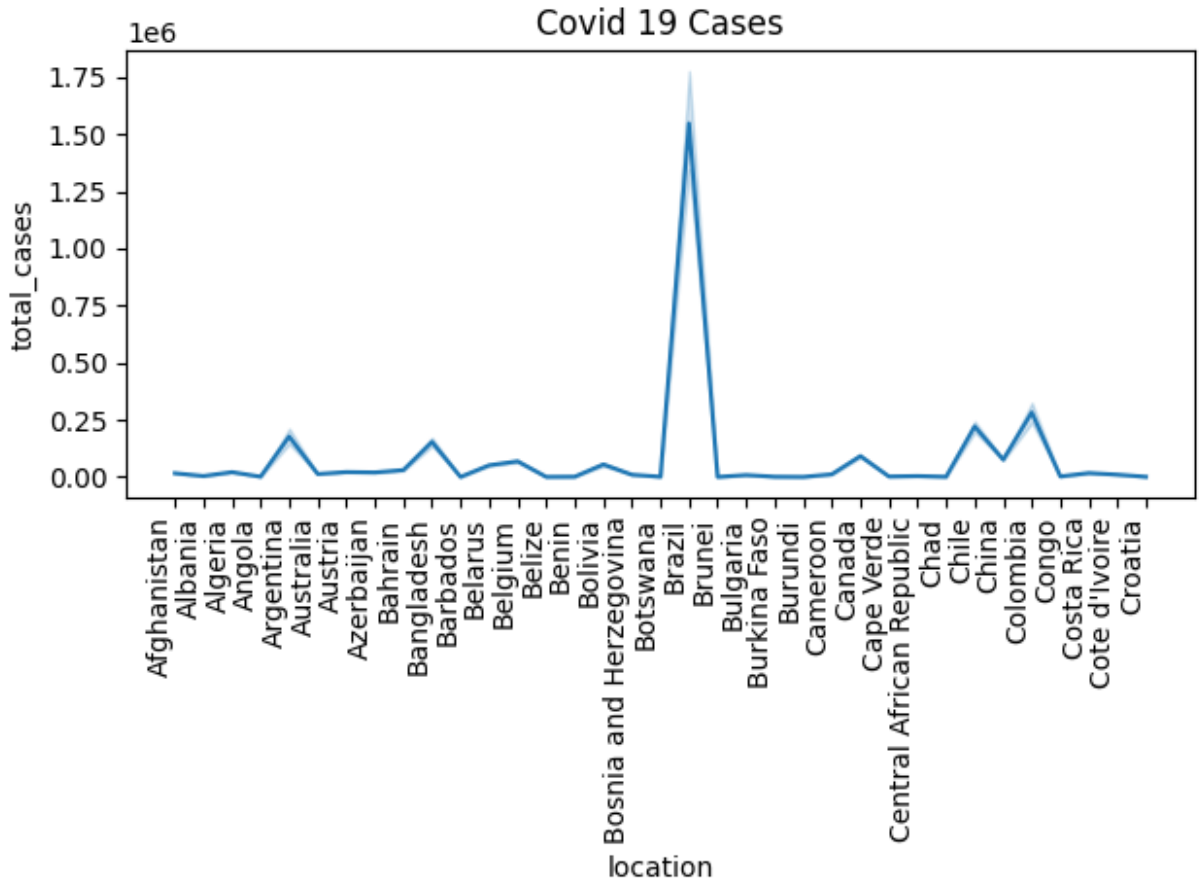
In [ ]:
```
data.head()
```

Out[ ]:

	iso_code	location	date	total_cases	total_deaths	stringency_index	population	gdp_per_capi
0	AFG	Afghanistan	2019-12-31	0.0	0.0	0.0	38928341	1803.9
1	AFG	Afghanistan	2020-01-01	0.0	0.0	0.0	38928341	1803.9
2	AFG	Afghanistan	2020-01-02	0.0	0.0	0.0	38928341	1803.9
3	AFG	Afghanistan	2020-01-03	0.0	0.0	0.0	38928341	1803.9
4	AFG	Afghanistan	2020-01-04	0.0	0.0	0.0	38928341	1803.9

In [ ]:
```
aax = sns.lineplot(data = data, x = 'location', y = 'total_cases')
aax.set_xticklabels(aax.get_xticklabels(),rotation = 90, ha = 'right')
plt.title('Covid 19 Cases')
plt.tight_layout()
```

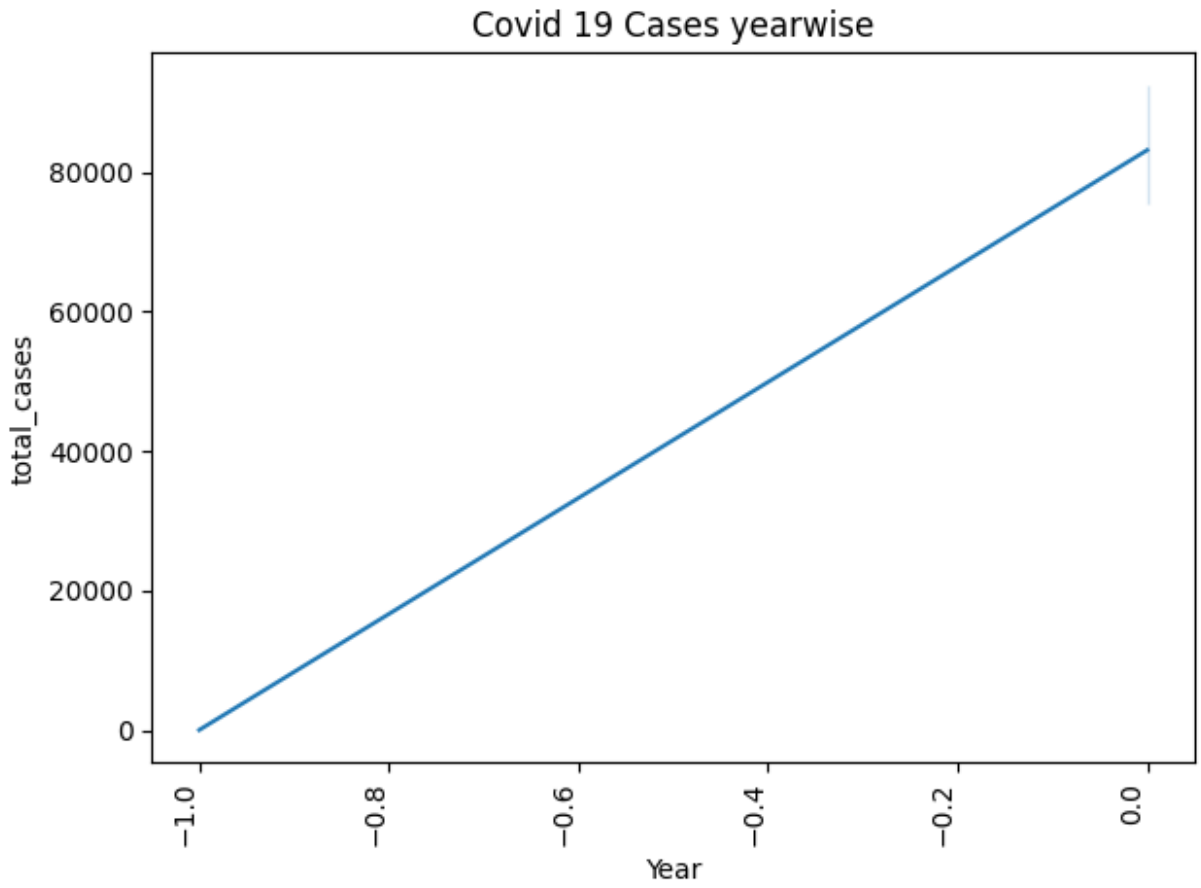
```
C:\Users\Hunain\AppData\Local\Temp\ipykernel_12352\462942659.py:2: UserWarning: Fixe
dFormatter should only be used together with FixedLocator
  aax.set_xticklabels(aax.get_xticklabels(),rotation = 90, ha = 'right')
```



In [ ]:

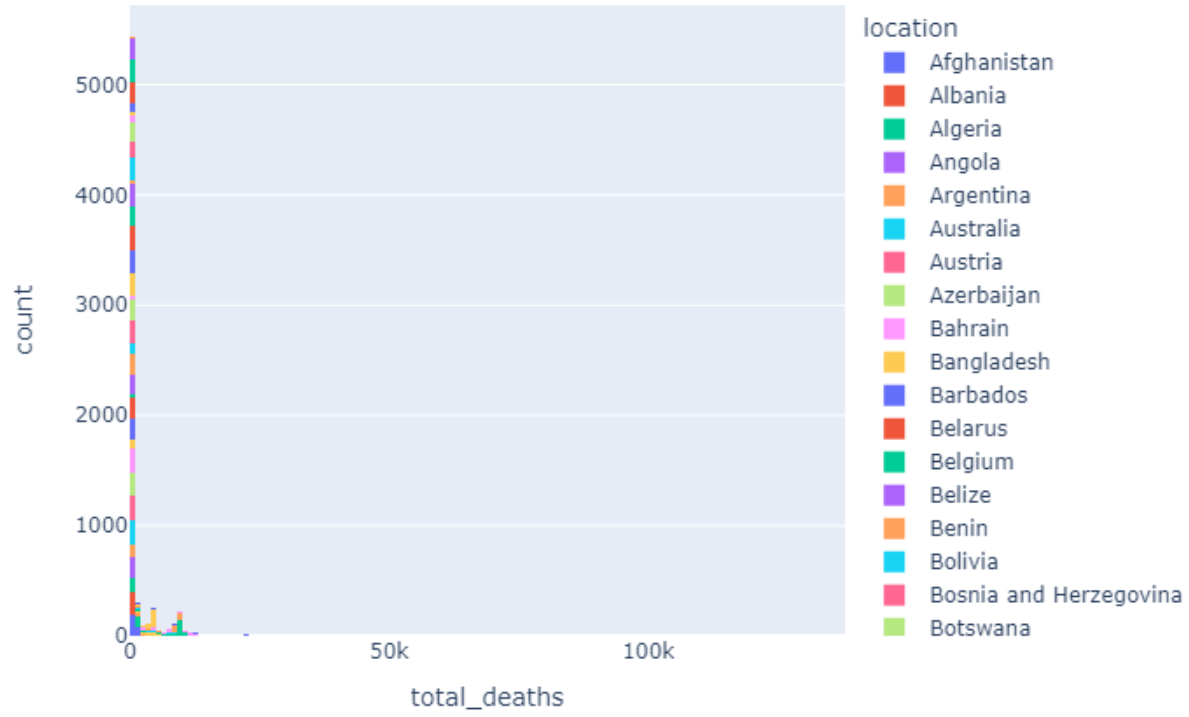
```
aax = sns.lineplot(data = data, x = 'Year', y = 'total_cases')
aax.set_xticklabels(aax.get_xticklabels(),rotation = 90, ha = 'right')
plt.title('Covid 19 Cases yearwise')
plt.tight_layout()
```

```
C:\Users\Hunain\AppData\Local\Temp\ipykernel_12352\4208503907.py:2: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [ ]:

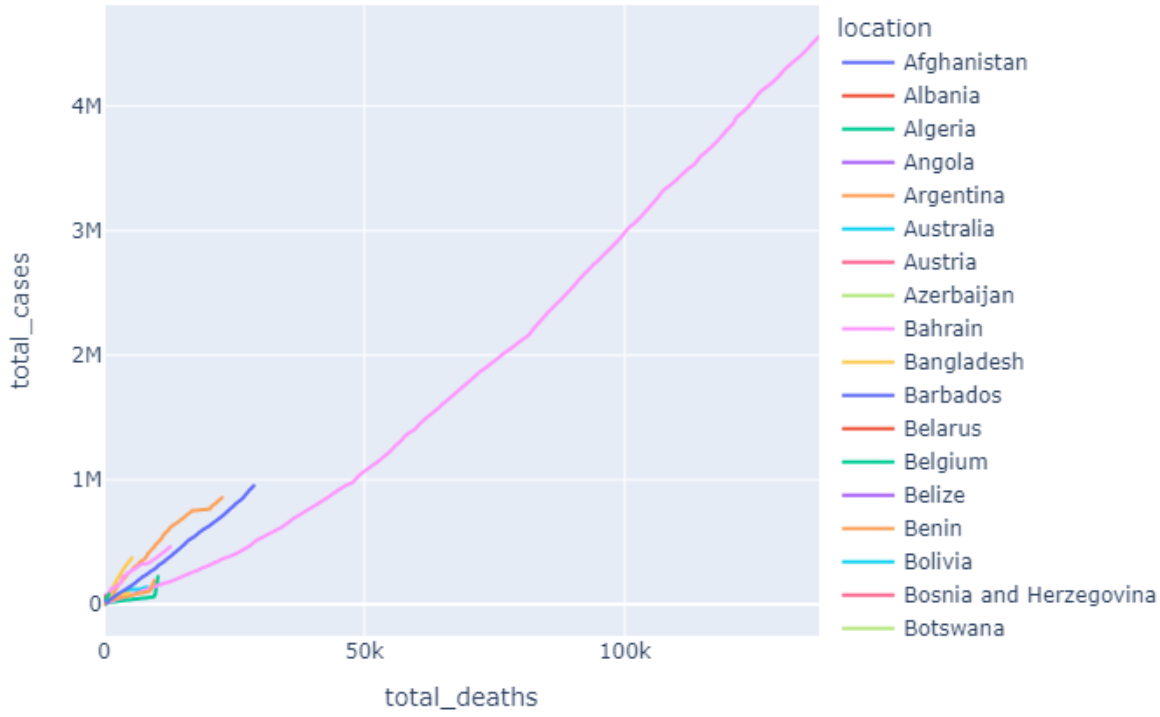
```
fig = px.histogram(data, x="total_deaths", color="location")
fig.show('png')
```



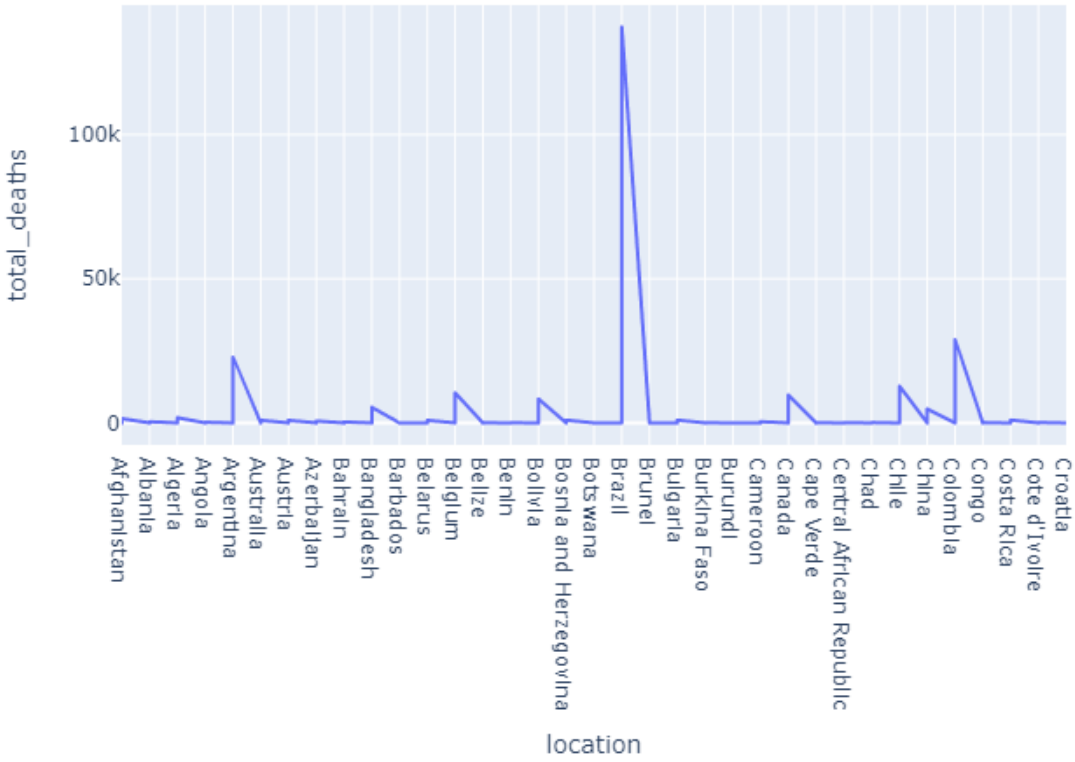
In [ ]:

```
fig = px.line(data, x="total_deaths", y="total_cases", color='location')
```

```
fig.show('png')
```



In [ ]:
```
fig = px.line(data, x='location', y="total_deaths")
fig.show('png')
```



In [ ]:
```
fig = px.scatter_geo(data, locations="iso_code", color="location",
                     size="total_deaths",hover_name='total_cases',
```

```
                    projection="natural earth")
fig.show('png')
```