```
import pandas as pd
import numpy as np

# Load the CSV files into DataFrames
df1 = pd.read_csv('/content/indian_cities - kaggle dataset (2).csv')
df2 = pd.read_csv('/content/total-final-2 (1).csv')
unique_values=df2['city'].value_counts()

# Print or use the unique values as needed
print("Unique values in 'column_name':", unique_values)
# Merge the DataFrames based on a common column
merged_df = pd.merge(df1, df2, on='city', how='inner')


# # Save the merged DataFrame to a new CSV file
# merged_df.to_csv('merged_file.csv', index=False)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-2-dafb614e6e2c> in <cell line: 5>()
      3 
      4 # Load the CSV files into DataFrames
----> 5 df1 = pd.read_csv('/content/indian_cities - kaggle dataset (2).csv')
      6 df2 = pd.read_csv('/content/total-final-2 (1).csv')
      7 unique_values=df2['city'].value_counts()

                                6 frames
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding,
compression, memory_map, is_text, errors, storage_options)
    854         if ioargs.encoding and "b" not in ioargs.mode:
    855             # Encoding
--> 856             handle = open(
    857                 handle,
    858                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: '/content/indian_cities - kaggle dataset (2).csv'
```

```
import numpy as np
import pandas as pd


df= pd.read_csv("/content/Sheet 1-1-merged_file.xlsx - Sheet 1-1-merged_file (2).csv")
```

1. Basic Exploratory Data Analysis and inferences.

[ ] ↳ *7 cells hidden*

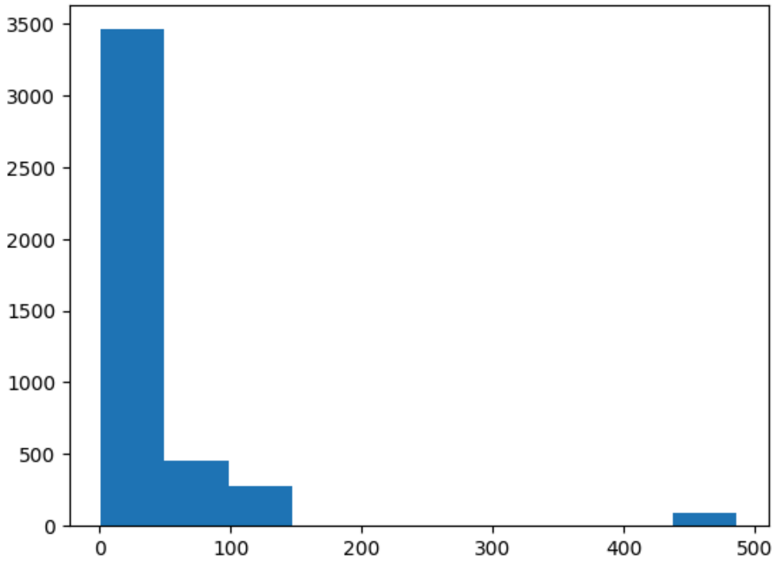
2. Visualization using various plots and their inferences.

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

a) Histogram

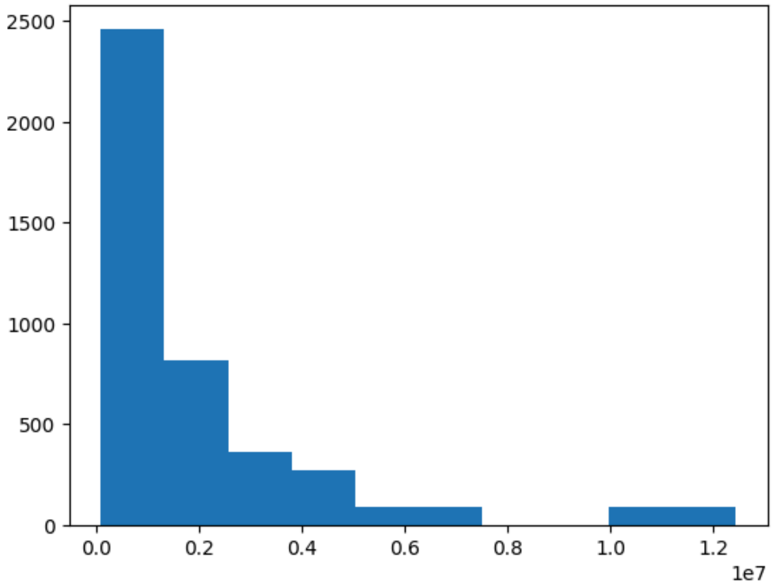
```
plt.hist(df['Rank'])
```

```
(array([3458.,  455.,  273.,    0.,    0.,    0.,    0.,    0.,    0.,
          91.]),
 array([  1. ,  49.5,  98. , 146.5, 195. , 243.5, 292. , 340.5, 389. ,
        437.5, 486. ]),
 <BarContainer object of 10 artists>)
```



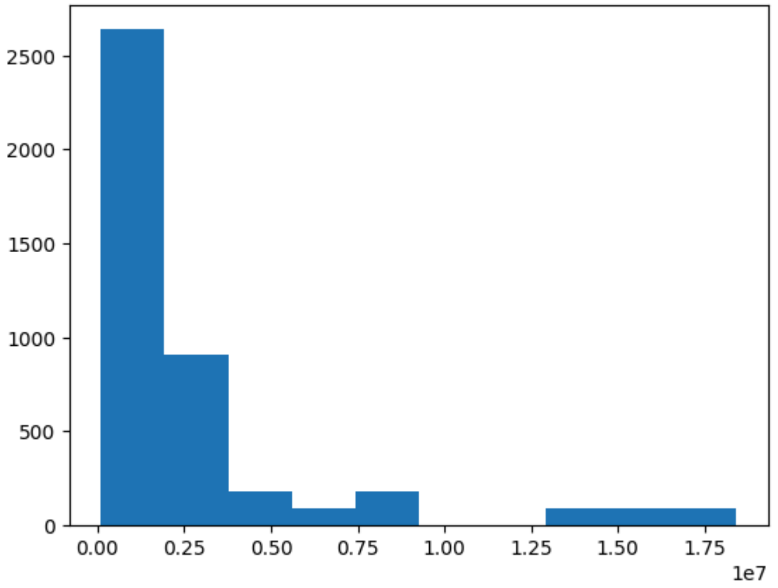
```
plt.hist(df['Population'])
```

```
(array([2457.,  819.,  364.,  273.,   91.,   91.,    0.,    0.,   91.,
          91.]),
 array([  102244. ,  1336256.9,  2570269.8,  3804282.7,  5038295.6,
         6272308.5,  7506321.4,  8740334.3,  9974347.2, 11208360.1,
        12442373. ]),
 <BarContainer object of 10 artists>)
```



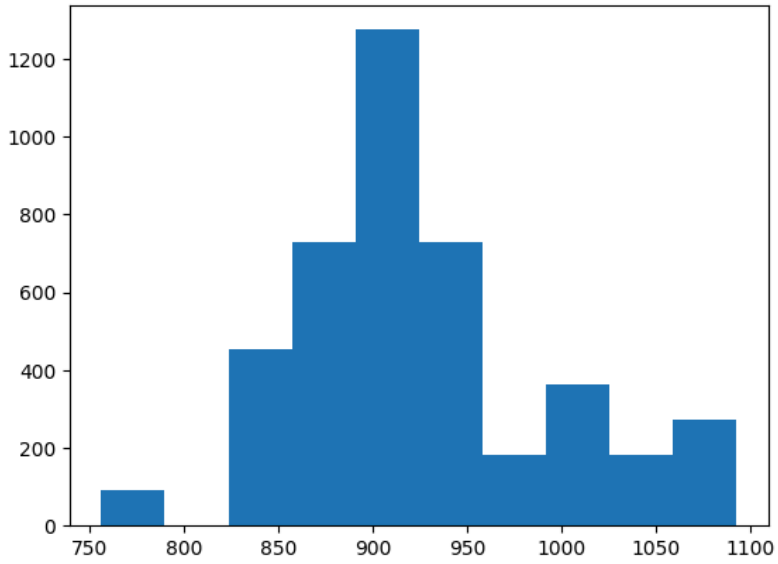
```
plt.hist(df['Metro_Population'])
```

```
    (array([2639.,  910.,  182.,   91.,  182.,    0.,    0.,   91.,   91.,
              91.]),
     array([  102244. ,  1933448.4,  3764652.8,  5595857.2,  7427061.6,
             9258266. , 11089470.4, 12920674.8, 14751879.2, 16583083.6,
            18414288. ]),
     <BarContainer object of 10 artists>)
```



```
plt.hist(df['Sexratio'])
```

```
    (array([  91.,    0.,  455.,  728., 1274.,  728.,  182.,  364.,  182.,
             273.]),
     array([ 756.  ,  789.7,  823.4,  857.1,  890.8,  924.5,  958.2,  991.9,
            1025.6, 1059.3, 1093. ]),
     <BarContainer object of 10 artists>)
```



› b) Scatter plot

[ ] ↳ 4 cells hidden

› c) pie chart

[ ] ↳ 1 cell hidden

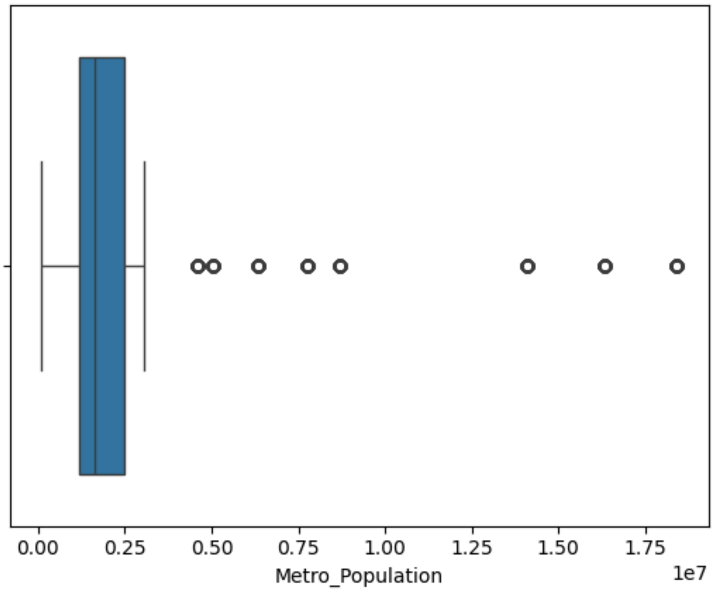
› d) barchart

[ ] ↳ 1 cell hidden

3. Handling outliers.

```
sns.boxplot(x='Metro_Population', data=df)
```

```
<Axes: xlabel='Metro_Population'>
```



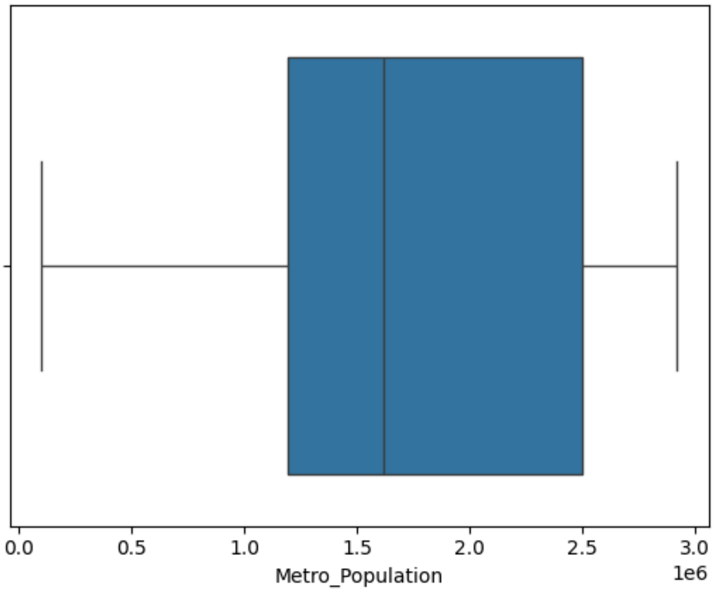
```
print(df['Metro_Population'].quantile(0.80))
```

```
2920067.0
```

```
df['Metro_Population'] = np.where(df['Metro_Population'] > 2920067, 2920067, df['Metro_Population'])
```

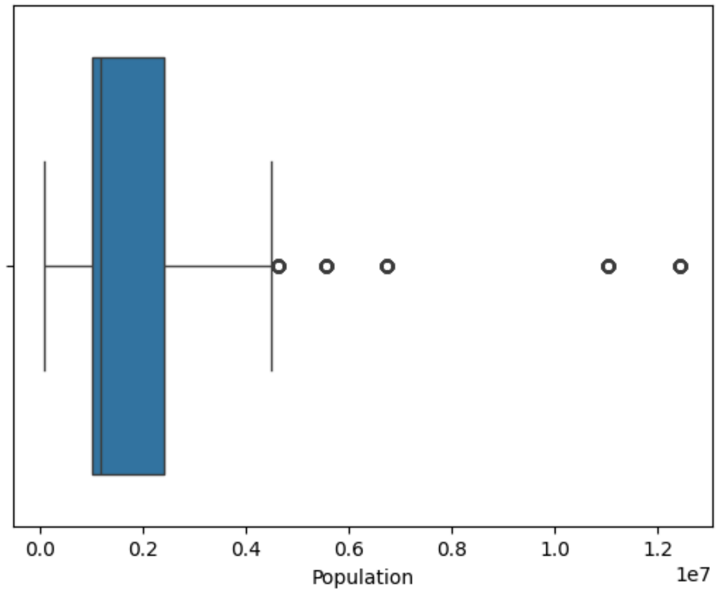
```
sns.boxplot(x='Metro_Population', data=df)
```

```
<Axes: xlabel='Metro_Population'>
```



```
sns.boxplot(x='Population', data=df)
```

```
<Axes: xlabel='Population'>
```



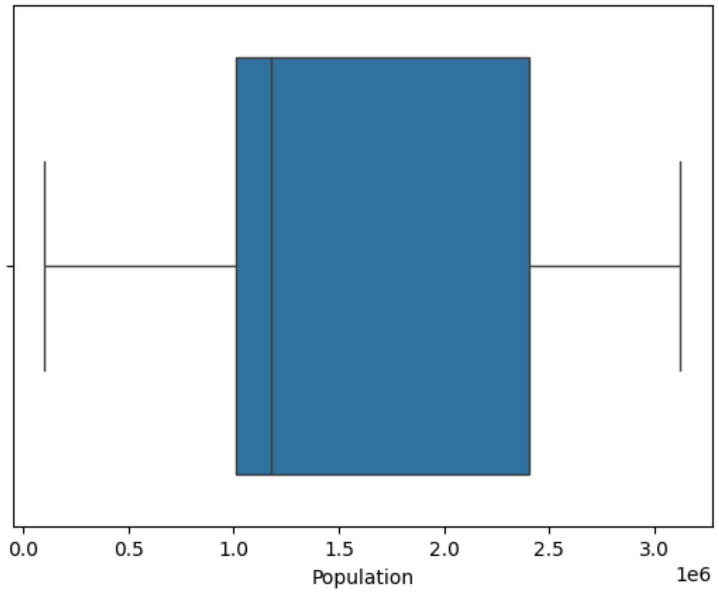
```
val = df['Population'].quantile(0.85)
print(val)
```

```
3124458.0
```

```
df['Population'] = np.where(df['Population'] > val, val, df['Population'])
```

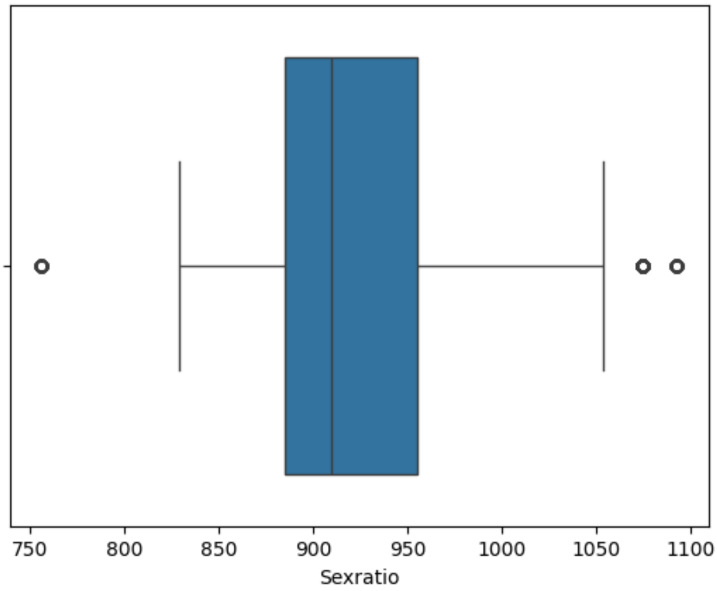
```
sns.boxplot(x='Population', data=df)
```

```
<Axes: xlabel='Population'>
```



```
sns.boxplot(x='Sexratio', data=df)
```

```
<Axes: xlabel='Sexratio'>
```



```
val = df['Sexratio'].quantile(0.93)
print(val)
```

```
1054.0
```

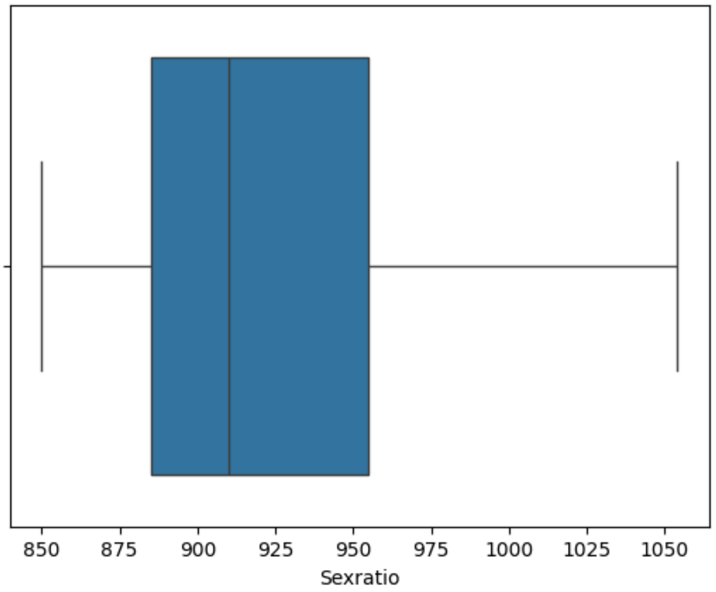
```
val_left = df['Sexratio'].quantile(0.05)
print(val_left)
```

```
850.0
```

```
df['Sexratio'] = np.where(df['Sexratio'] > val, val, df['Sexratio'])
df['Sexratio'] = np.where(df['Sexratio'] < val_left, val_left, df['Sexratio'])
```

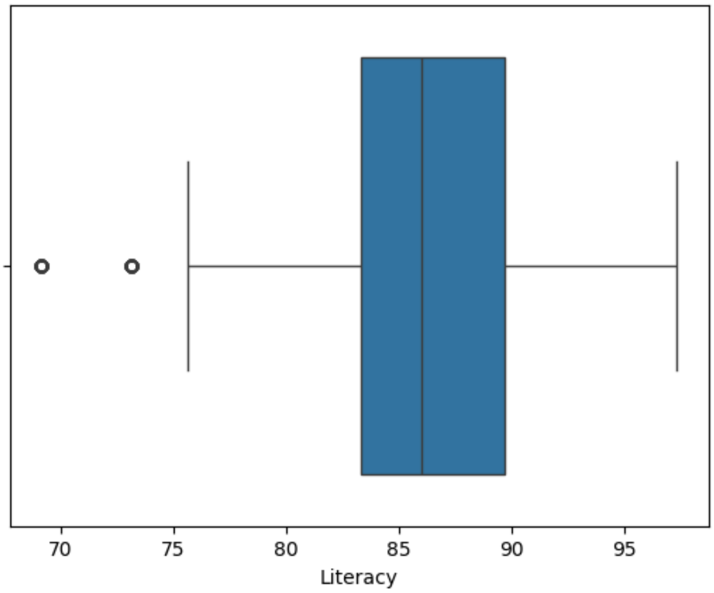
```
sns.boxplot(x='Sexratio', data=df)
```

```
<Axes: xlabel='Sexratio'>
```



```
sns.boxplot(x='Literacy', data=df)
```

```
<Axes: xlabel='Literacy'>
```



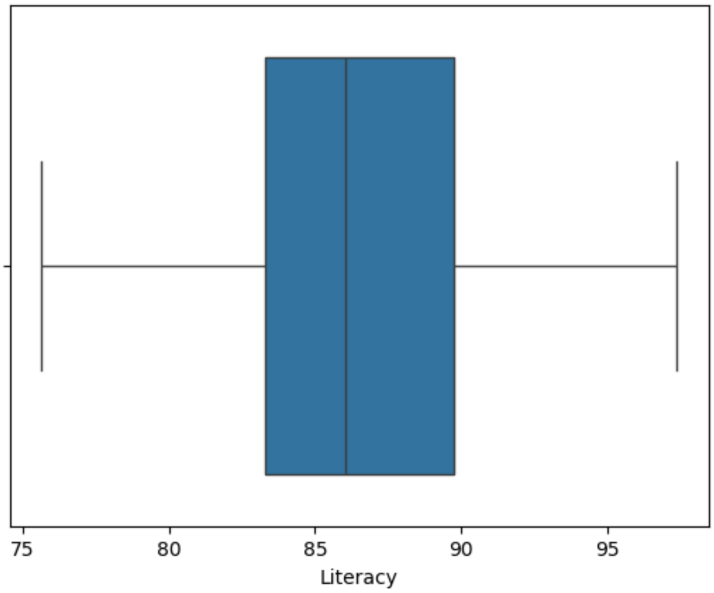
```
val_left = df['Literacy'].quantile(0.05)
print(val_left)
```

```
75.66
```

```
df['Literacy'] = np.where(df['Literacy'] < val_left, val_left, df['Literacy'])
```

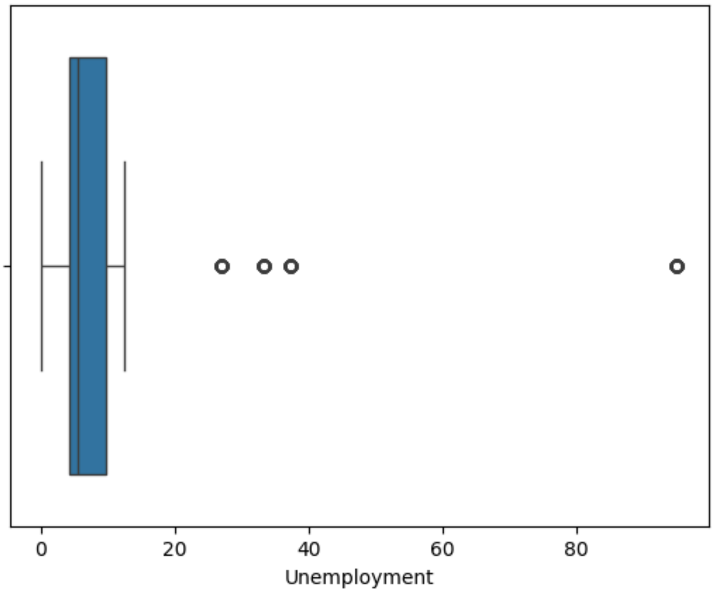
```
sns.boxplot(x='Literacy', data=df)
```

```
<Axes: xlabel='Literacy'>
```



```
sns.boxplot(x='Unemployment', data=df)
```

```
<Axes: xlabel='Unemployment'>
```



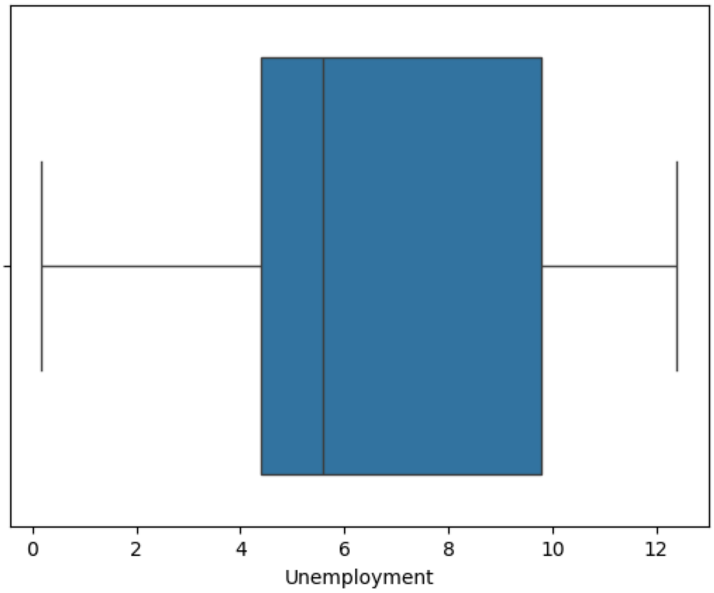
```
val = df['Unemployment'].quantile(0.80)
print(val)
```

```
12.4
```

```
df['Unemployment'] = np.where(df['Unemployment'] > val, val, df['Unemployment'])
```

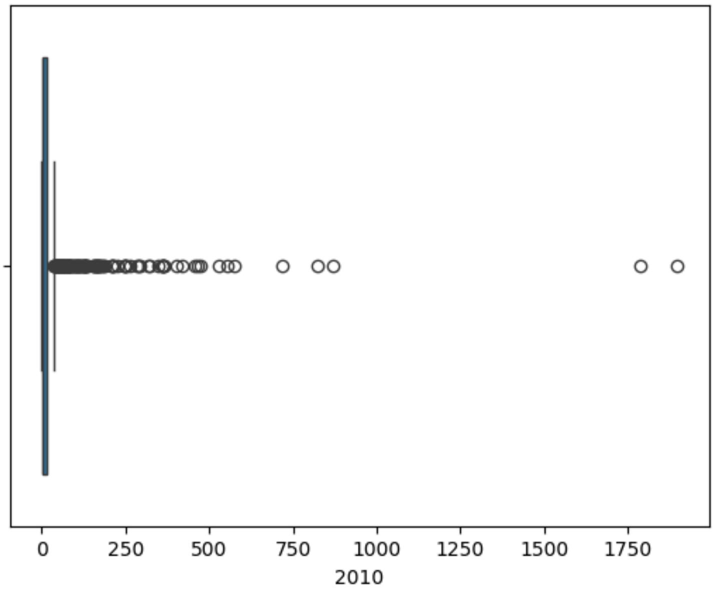
```
sns.boxplot(x='Unemployment', data=df)
```

```
<Axes: xlabel='Unemployment'>
```



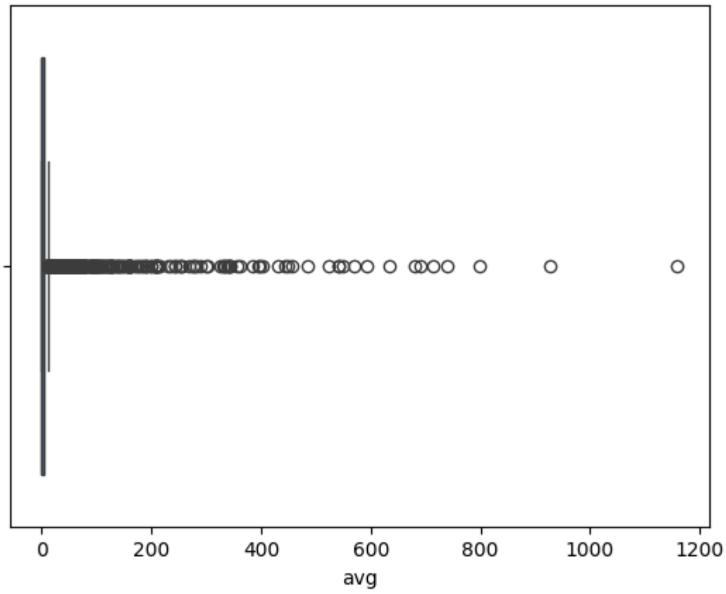
```
sns.boxplot(x='2010', data=df)
```

```
<Axes: xlabel='2010'>
```



```
sns.boxplot(x='avg', data=df)
```

```
<Axes: xlabel='avg'>
```



```
df.head()
```

	Rank	city	State	Population	Metro_Population	Sexratio	Literacy	Unemployment	Poverty (MPI)	crime_name	201
0	4	Hyderabad	Andhra Pradesh	6731790	2920067	955	83.26	95.0	NaN	Acid attack	Na
1	4	Hyderabad	Andhra Pradesh	6731790	2920067	955	83.26	95.0	NaN	Agrarian riots	Na
2	4	Hyderabad	Andhra Pradesh	6731790	2920067	955	83.26	95.0	NaN	Arson	0.
3	4	Hyderabad	Andhra Pradesh	6731790	2920067	955	83.26	95.0	NaN	Assault on women with intent to outrage her mo...	4.
4	4	Hyderabad	Andhra Pradesh	6731790	2920067	955	83.26	95.0	NaN	Attempt to acid attack	Na

Next steps: View recommended plots

4. Handling missing values.

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

```
Rank                   0
city                   0
State                  0
Population             0
Metro_Population       0
Sexratio               0
Literacy               0
Poverty (MPI)        364
crime_name             0
2010                3285
2011                2820
2012                2820
2013                2820
2014                   0
avg                    0
dtype: int64
```

```
df['Poverty (MPI)'].fillna(df['Poverty (MPI)'].mean(), inplace= True)
```

```
condition = (df['2010'].isnull()  & df['2011'].isnull()  & df['2012'].isnull()  & df['2013'].isnull())
```

```
filtered_df = df[~condition]
```

```
filtered_df
```

	Rank	city	State	Population	Metro_Population	Sexratio	Literacy	Poverty (MPI)	crime_name	2010	2011	2012	20
2	4	15	0	6731790	2920067	955	83.26	0.049	2	0.3	0.6	1.3	0
3	4	15	0	6731790	2920067	955	83.26	0.049	3	4.5	4.2	3.9	6
6	4	15	0	6731790	2920067	955	83.26	0.049	6	2.6	1.8	2.1	1
8	4	15	0	6731790	2920067	955	83.26	0.049	8	27.9	23.5	17.7	17
9	4	15	0	6731790	2920067	955	83.26	0.049	9	16.4	8.9	8.3	8
...	...	...	...	...	...	...	...	...	...	...	...	...	
4260	78	4	16	563917	1243008	929	83.30	0.049	74	4.5	8.0	17.1	2
4262	78	4	16	563917	1243008	929	83.30	0.049	76	0.5	1.9	6.0	3
4267	78	4	16	563917	1243008	929	83.30	0.049	81	2.2	6.2	5.9	12
4270	78	4	16	563917	1243008	929	83.30	0.049	84	29.6	51.6	76.2	59
4271	78	4	16	563917	1243008	929	83.30	0.049	85	168.7	272.0	313.5	385

1504 rows × 15 columns

```
filtered_df.isna().sum()
```

```
Rank                  0
city                  0
State                 0
Population            0
Metro_Population      0
Sexratio              0
Literacy              0
Poverty (MPI)         0
crime_name            0
2010                512
2011                 47
2012                 47
2013                 47
2014                  0
avg                   0
dtype: int64
```

5. Handling categorical features.

[ ] ↳ 8 cells hidden

6. Feature scaling.

a) Normalization

```
from sklearn.preprocessing import Normalizer

scaler = Normalizer()
scaled_data = scaler.fit_transform(filtered_df)
scaled_df = pd.DataFrame(scaled_data, columns=df.columns)
print(scaled_df.head())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-65-79a0a1928bd8> in <cell line: 4>()
      2
      3 scaler = Normalizer()
----> 4 scaled_data = scaler.fit_transform(filtered_df)
      5 scaled_df = pd.DataFrame(scaled_data,
      6                          columns=df.columns)

                                  5 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan,
msg_dtype, estimator_name, input_name)
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

ValueError: Input X contains NaN.
Normalizer does not accept missing values encoded as NaN natively. For supervised learning, you might want to
consider sklearn.ensemble.HistGradientBoostingClassifier and Regressor which accept missing values encoded as
```