VerveBridge TASK-2

Big Game Census Data Visualization.

IMPORT ALL REQUIRED PACKAGES

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [2]:
```
path=r"C:\Users\Sruth\Documents\Naresh it\EDA\Datafiles\uscensusbureau-big-game-
```

In [3]:
```
df=pd.read_excel(path)
df
```

Out[3]:

	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Popula Estir (as of 1) - 2
0	1620000US0100124	100124	Abbeville city, Alabama	2688	2688	2683	
1	1620000US0100460	100460	Adamsville city, Alabama	4522	4522	4517	
2	1620000US0100484	100484	Addison town, Alabama	758	756	754	
3	1620000US0100676	100676	Akron town, Alabama	356	356	355	
4	1620000US0100820	100820	Alabaster city, Alabama	30352	31066	31176	
...	...	...	...	...	...	...	
19505	1620000US5681300	5681300	Wamsutter town, Wyoming	451	451	450	
19506	1620000US5683040	5683040	Wheatland town, Wyoming	3627	3627	3629	
19507	1620000US5684925	5684925	Worland city, Wyoming	5487	5487	5494	
19508	1620000US5685015	5685015	Wright town, Wyoming	1807	1807	1807	
19509	1620000US5686665	5686665	Yoder town, Wyoming	151	151	152	

19510 rows × 12 columns

DATA QUCIK CHECKS

shape

In [4]:
```
# gives number of columns(12)  and rows(19510) present in dataset

df.shape
```

Out[4]:
```
(19510, 12)
```

len

In [5]:
```
len(df)
```
Out[5]:
```
19510
```
size

In [6]:
```
#Gives total elements in dataframe

df.size
```
Out[6]:
```
234120
```
dtypes

In [7]:
```
# dtypes reprecnts the datatype of each column categorical columns(object),numer
df.dtypes
```
Out[7]:
```
Geographic ID                                  object
GEOID 2                                         int64
Geography, full name (City, State)             object
April 1, 2010 - Census                         object
April 1, 2010 - Estimates Base                  int64
Population Estimate (as of July 1) - 2010       int64
Population Estimate (as of July 1) - 2011       int64
Population Estimate (as of July 1) - 2012       int64
Population Estimate (as of July 1) - 2013       int64
Population Estimate (as of July 1) - 2014       int64
Population Estimate (as of July 1) - 2015       int64
Population Estimate (as of July 1) - 2016       int64
dtype: object
```
info

In [8]:
```
# gives overall information about a dataframe
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19510 entries, 0 to 19509
Data columns (total 12 columns):
 #   Column                                     Non-Null Count  Dtype 
---  ------                                     --------------  ----- 
 0   Geographic ID                              19510 non-null  object
 1   GEOID 2                                    19510 non-null  int64 
 2   Geography, full name (City, State)         19510 non-null  object
 3   April 1, 2010 - Census                     19510 non-null  object
 4   April 1, 2010 - Estimates Base             19510 non-null  int64 
 5   Population Estimate (as of July 1) - 2010  19510 non-null  int64 
 6   Population Estimate (as of July 1) - 2011  19510 non-null  int64 
 7   Population Estimate (as of July 1) - 2012  19510 non-null  int64 
 8   Population Estimate (as of July 1) - 2013  19510 non-null  int64 
 9   Population Estimate (as of July 1) - 2014  19510 non-null  int64 
 10  Population Estimate (as of July 1) - 2015  19510 non-null  int64 
 11  Population Estimate (as of July 1) - 2016  19510 non-null  int64 
dtypes: int64(9), object(3)
memory usage: 1.8+ MB
```
head

In [9]:
```
# Gives first 5 rows of dataframe

df.head()
```

Out[9]:

	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011
0	1620000US0100124	100124	Abbeville city, Alabama	2688	2688	2683	2685
1	1620000US0100460	100460	Adamsville city, Alabama	4522	4522	4517	4495
2	1620000US0100484	100484	Addison town, Alabama	758	756	754	753
3	1620000US0100676	100676	Akron town, Alabama	356	356	355	345
4	1620000US0100820	100820	Alabaster city, Alabama	30352	31066	31176	31362

tail

In [10]:
```
# Gives last 5 rows of dataframe

df.tail()
```

Out[10]:

	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Popula Estir (as of 1) - 2
19505	1620000US5681300	5681300	Wamsutter town, Wyoming	451	451	450	
19506	1620000US5683040	5683040	Wheatland town, Wyoming	3627	3627	3629	
19507	1620000US5684925	5684925	Worland city, Wyoming	5487	5487	5494	
19508	1620000US5685015	5685015	Wright town, Wyoming	1807	1807	1807	
19509	1620000US5686665	5686665	Yoder town, Wyoming	151	151	152	

checking missing values are present in dataframe

In [11]:
```
# There is no missing values present in dataframe

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

Out[11]:
```
Geographic ID                                   0
GEOID 2                                         0
Geography, full name (City, State)              0
April 1, 2010 - Census                          0
April 1, 2010 - Estimates Base                  0
Population Estimate (as of July 1) - 2010       0
Population Estimate (as of July 1) - 2011       0
Population Estimate (as of July 1) - 2012       0
Population Estimate (as of July 1) - 2013       0
Population Estimate (as of July 1) - 2014       0
Population Estimate (as of July 1) - 2015       0
Population Estimate (as of July 1) - 2016       0
dtype: int64
```

Conversion of categorical columns

In [12]:
```
cat_cols=df.select_dtypes(include='object').columns
cat_cols
```

Out[12]:
```
Index(['Geographic ID', 'Geography, full name (City, State)',
       'April 1, 2010 - Census'],
      dtype='object')
```

In [13]:
```
len(cat_cols)
```

Out[13]:
```
3
```

Conversion of Numerical Columns

In [14]:
```python
num_cols=df.select_dtypes(exclude='object').columns
num_cols
```

Out[14]:
```
Index(['GEOID 2', 'April 1, 2010 - Estimates Base',
       'Population Estimate (as of July 1) - 2010',
       'Population Estimate (as of July 1) - 2011',
       'Population Estimate (as of July 1) - 2012',
       'Population Estimate (as of July 1) - 2013',
       'Population Estimate (as of July 1) - 2014',
       'Population Estimate (as of July 1) - 2015',
       'Population Estimate (as of July 1) - 2016'],
      dtype='object')
```

In [15]:
```python
len(num_cols)
```

Out[15]:
```
9
```

Categorical column analysis

In [16]:
```python
cat_cols
```

Out[16]:
```
Index(['Geographic ID', 'Geography, full name (City, State)',
       'April 1, 2010 - Census'],
      dtype='object')
```

In [17]:
```python
len(cat_cols)
```

Out[17]:
```
3
```

In [18]:
```python
df['Geographic ID'].value_counts()
```

Out[18]:
```
Geographic ID
1620000US0100124    1
1620000US3913778    1
1620000US3914156    1
1620000US3914128    1
1620000US3914114    1
                   ..
1620000US2060900    1
1620000US2060825    1
1620000US2060325    1
1620000US2059875    1
1620000US5686665    1
Name: count, Length: 19510, dtype: int64
```

In [19]:
```python
# nunique function  gives the unique items in the specify column

df['Geographic ID'].nunique()
```

Out[19]:
```
19510
```

In [20]:
```python
df['Geography, full name (City, State)'].value_counts()
```

Out[20]:
```
Geography, full name (City, State)
Abbeville city, Alabama       1
Chauncey village, Ohio        1
Chickasaw village, Ohio       1
Cheviot city, Ohio            1
Chesterville village, Ohio    1
                             ..
Rolla city, Kansas            1
Roeland Park city, Kansas     1
Robinson city, Kansas         1
Riley city, Kansas            1
Yoder town, Wyoming           1
Name: count, Length: 19510, dtype: int64
```

In [21]:
```
# there are 19510 items are unique in the specify column

df['Geography, full name (City, State)'].nunique()
```

Out[21]:
```
19510
```

In [22]:
```
df['April 1, 2010 - Census'].value_counts()
```

Out[22]:
```
April 1, 2010 - Census
(X)      46
94       27
139      26
112      26
63       24
         ..
2571      1
3931      1
6835      1
21447     1
5487      1
Name: count, Length: 7973, dtype: int64
```

In [23]:
```
# there are 7973 unique values present in the column

df['April 1, 2010 - Census'].nunique()
```

Out[23]:
```
7973
```

Numerical Column analysis

In [25]:
```
num_cols
```

Out[25]:
```
Index(['GEOID 2', 'April 1, 2010 - Estimates Base',
       'Population Estimate (as of July 1) - 2010',
       'Population Estimate (as of July 1) - 2011',
       'Population Estimate (as of July 1) - 2012',
       'Population Estimate (as of July 1) - 2013',
       'Population Estimate (as of July 1) - 2014',
       'Population Estimate (as of July 1) - 2015',
       'Population Estimate (as of July 1) - 2016'],
      dtype='object')
```

In [26]:
```
len(num_cols)
```

Out[26]: 9

predefined Function

```
In [27]: df.describe()
```

Out[27]:

	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012	Popul Estima of Jul
count	1.951000e+04	1.951000e+04	1.951000e+04	1.951000e+04	1.951000e+04	1.95100(
mean	3.008271e+06	9.909960e+03	9.930210e+03	1.001311e+04	1.009960e+04	1.018003
std	1.461650e+06	8.012455e+04	8.028673e+04	8.113065e+04	8.195162e+04	8.265639
min	1.001240e+05	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000(
25%	1.857141e+06	3.690000e+02	3.690000e+02	3.682500e+02	3.660000e+02	3.66250(
50%	2.940709e+06	1.147000e+03	1.148000e+03	1.148000e+03	1.147000e+03	1.14650(
75%	4.202612e+06	4.603250e+03	4.606000e+03	4.620000e+03	4.632500e+03	4.64800(
max	5.686665e+06	8.174962e+06	8.192026e+06	8.284098e+06	8.361179e+06	8.42246(

skewness for numerical columns

```
In [28]: for i in num_cols:
             skew=round(df[i].skew(),2)
             print(f"skew of  column {i} is '{skew}'")
```

```
skew of  column GEOID 2 is '-0.08'
skew of  column April 1, 2010 - Estimates Base is '64.76'
skew of  column Population Estimate (as of July 1) - 2010 is '64.75'
skew of  column Population Estimate (as of July 1) - 2011 is '64.8'
skew of  column Population Estimate (as of July 1) - 2012 is '64.65'
skew of  column Population Estimate (as of July 1) - 2013 is '64.46'
skew of  column Population Estimate (as of July 1) - 2014 is '64.18'
skew of  column Population Estimate (as of July 1) - 2015 is '63.89'
skew of  column Population Estimate (as of July 1) - 2016 is '63.52'
```

kurtosis of numerical columns

```
In [29]: for i in num_cols[1:]:
             kurt=round(df[i].kurt(),2)
             print(f"kurtosis of  column {i} is '{kurt}'")
```

```
kurtosis of  column April 1, 2010 - Estimates Base is '5898.9'
kurtosis of  column Population Estimate (as of July 1) - 2010 is '5899.03'
kurtosis of  column Population Estimate (as of July 1) - 2011 is '5910.7'
kurtosis of  column Population Estimate (as of July 1) - 2012 is '5890.76'
kurtosis of  column Population Estimate (as of July 1) - 2013 is '5862.48'
kurtosis of  column Population Estimate (as of July 1) - 2014 is '5821.8'
kurtosis of  column Population Estimate (as of July 1) - 2015 is '5778.92'
kurtosis of  column Population Estimate (as of July 1) - 2016 is '5724.54'
```

Covariance matrix

In [30]:
```
df.cov(numeric_only=True)
```

Out[30]:

	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012
GEOID 2	2.136420e+12	-2.967754e+09	-2.973300e+09	-2.995470e+09	-3.012286e+09
April 1, 2010 - Estimates Base	-2.967754e+09	6.419943e+09	6.432933e+09	6.500447e+09	6.565984e+09
Population Estimate (as of July 1) - 2010	-2.973300e+09	6.432933e+09	6.445959e+09	6.513641e+09	6.579339e+09
Population Estimate (as of July 1) - 2011	-2.995470e+09	6.500447e+09	6.513641e+09	6.582183e+09	6.648704e+09
Population Estimate (as of July 1) - 2012	-3.012286e+09	6.565984e+09	6.579339e+09	6.648704e+09	6.716067e+09
Population Estimate (as of July 1) - 2013	-3.030919e+09	6.622052e+09	6.635548e+09	6.705629e+09	6.773728e+09
Population Estimate (as of July 1) - 2014	-3.051341e+09	6.672641e+09	6.686271e+09	6.757022e+09	6.825825e+09
Population Estimate (as of July 1) - 2015	-3.067985e+09	6.720005e+09	6.733761e+09	6.805148e+09	6.874622e+09
Population Estimate (as of July 1) - 2016	-3.081982e+09	6.752975e+09	6.766827e+09	6.838687e+09	6.908675e+09

Correlation

- correlation tells about how much relation between two variables
- denotes with r , r varies **-1 to +1**
- **-1 to 0** indicates **negative relation**
- **0 to 1** indivates **positive relation**
- **0** indicates **no relation**

In [31]:
```
correlation_data=df.corr(numeric_only=True)
correlation_data
```

Out[31]:

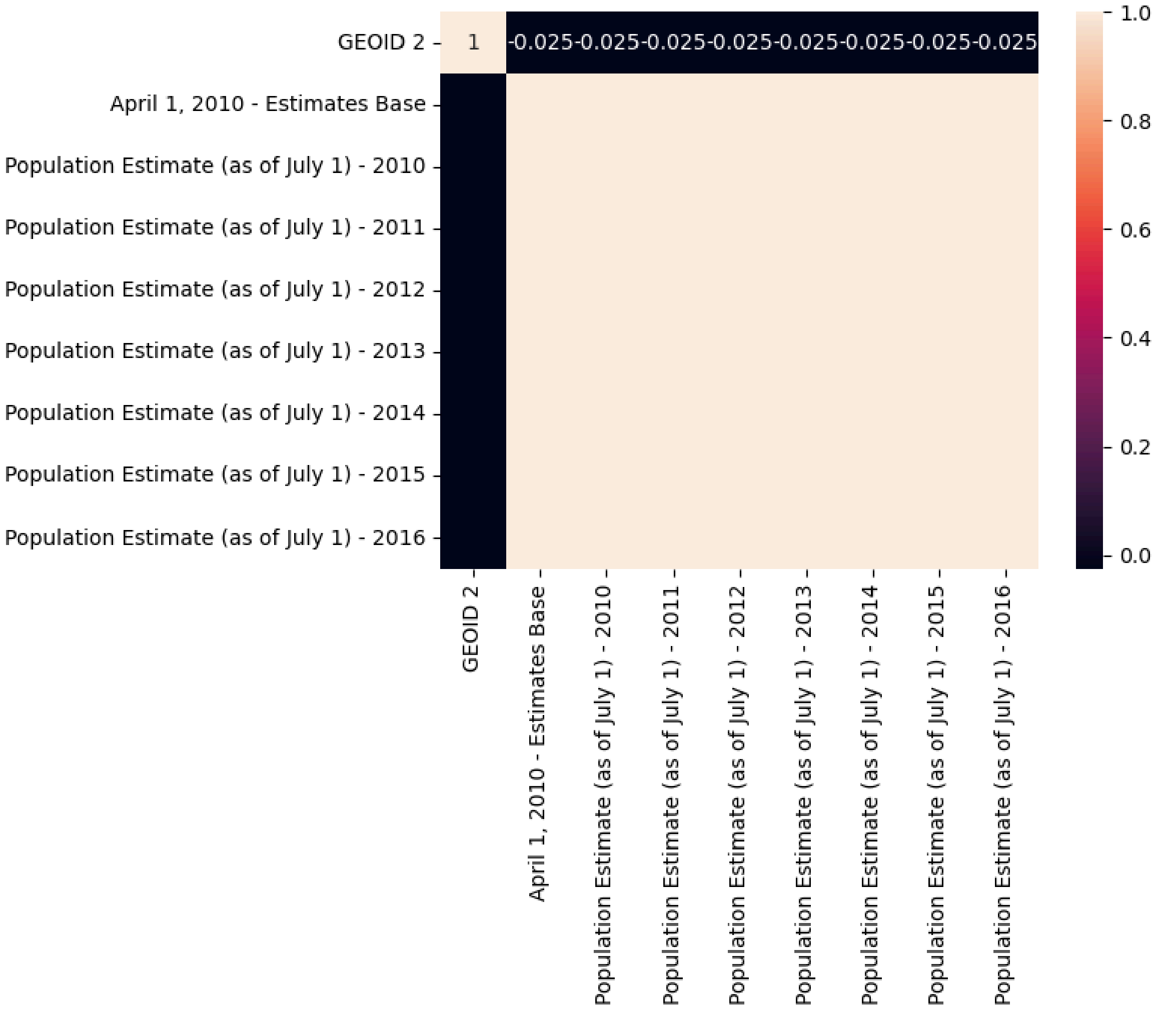
	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012	Population Estimate (as of July 1) - 2013	Popu Est (as c 1) -
GEOID 2	1.000000	-0.025341	-0.025337	-0.025260	-0.025148	-0.025087	-0.0
April 1, 2010 - Estimates Base	-0.025341	1.000000	0.999999	0.999983	0.999946	0.999886	0.9
Population Estimate (as of July 1) - 2010	-0.025337	0.999999	1.000000	0.999989	0.999956	0.999900	0.9
Population Estimate (as of July 1) - 2011	-0.025260	0.999983	0.999989	1.000000	0.999987	0.999950	0.9
Population Estimate (as of July 1) - 2012	-0.025148	0.999946	0.999956	0.999987	1.000000	0.999986	0.9
Population Estimate (as of July 1) - 2013	-0.025087	0.999886	0.999900	0.999950	0.999986	1.000000	0.9
Population Estimate (as of July 1) - 2014	-0.025062	0.999788	0.999806	0.999875	0.999938	0.999982	1.0
Population Estimate (as of July 1) - 2015	-0.025018	0.999654	0.999677	0.999766	0.999855	0.999927	0.9
Population Estimate (as of July 1) - 2016	-0.025005	0.999489	0.999516	0.999623	0.999737	0.999836	0.9

Heat map

- Heat map is one of the important **visulization method**, to show the matxice
- It is under seaborn package
- In very matrix we have highest and lowest values
- Heat map will give color visulazation

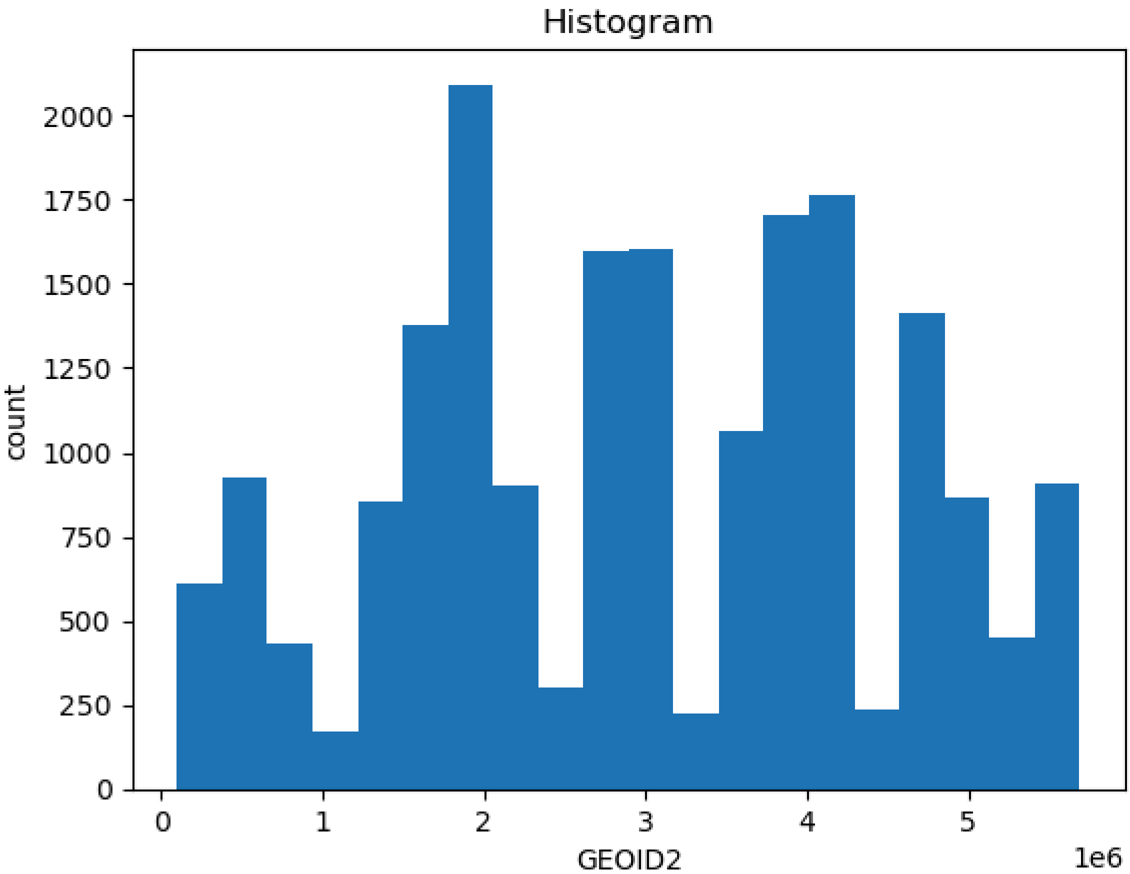
In [32]:
```
# Here there exist a positive relation between the variables ao each column

corr_data=df.corr(numeric_only=True)
sns.heatmap(corr_data,annot=True)
plt.show()
```



Histogram Analysis

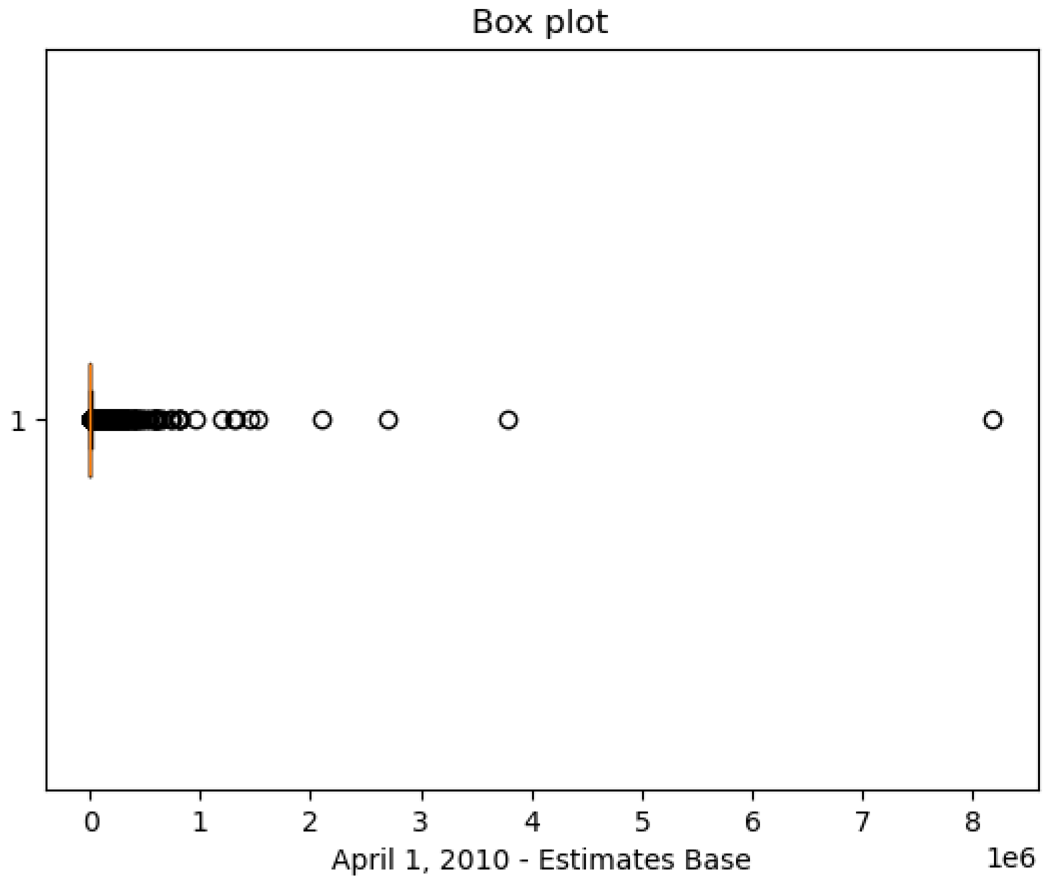
In [33]:
```
data=df['GEOID 2']
count,bins,x=plt.hist(data,bins=20)
plt.xlabel('GEOID2')
plt.ylabel('count')
plt.title('Histogram')
plt.show()
```

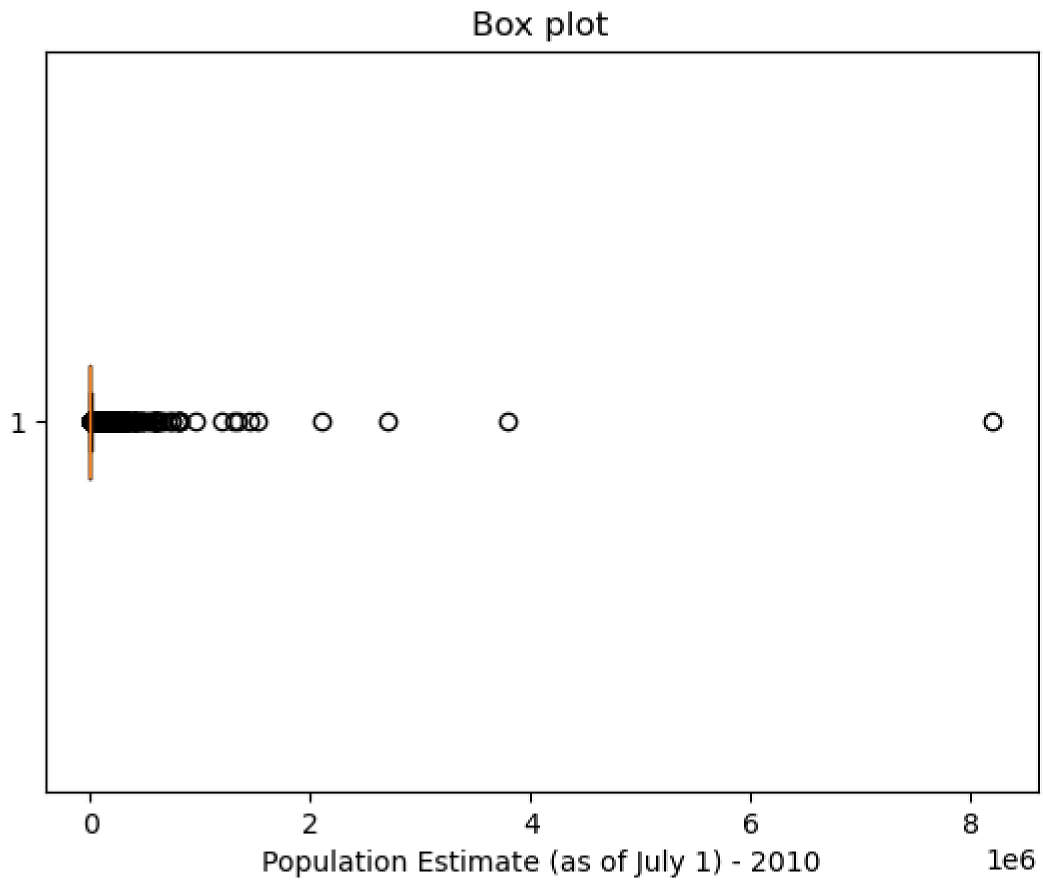


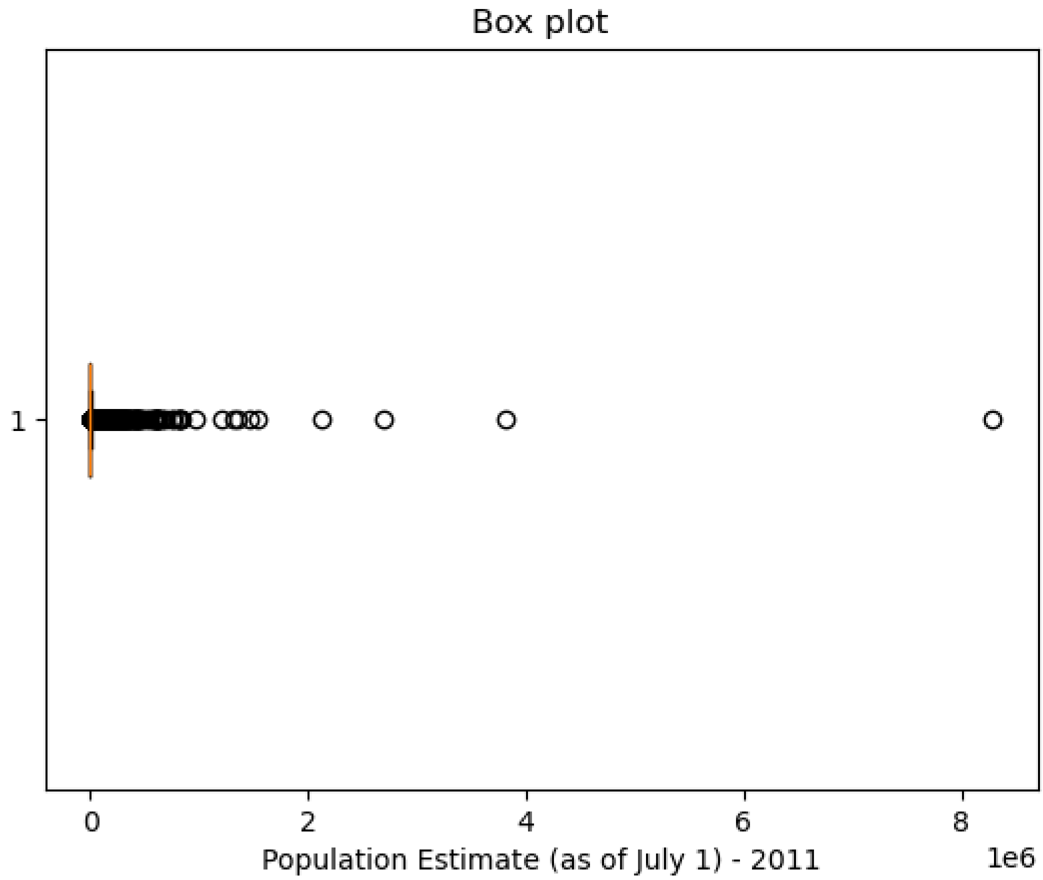
Outliers Analysis

num_cols[1:]-->it means removes first column (that not contain any outliers)

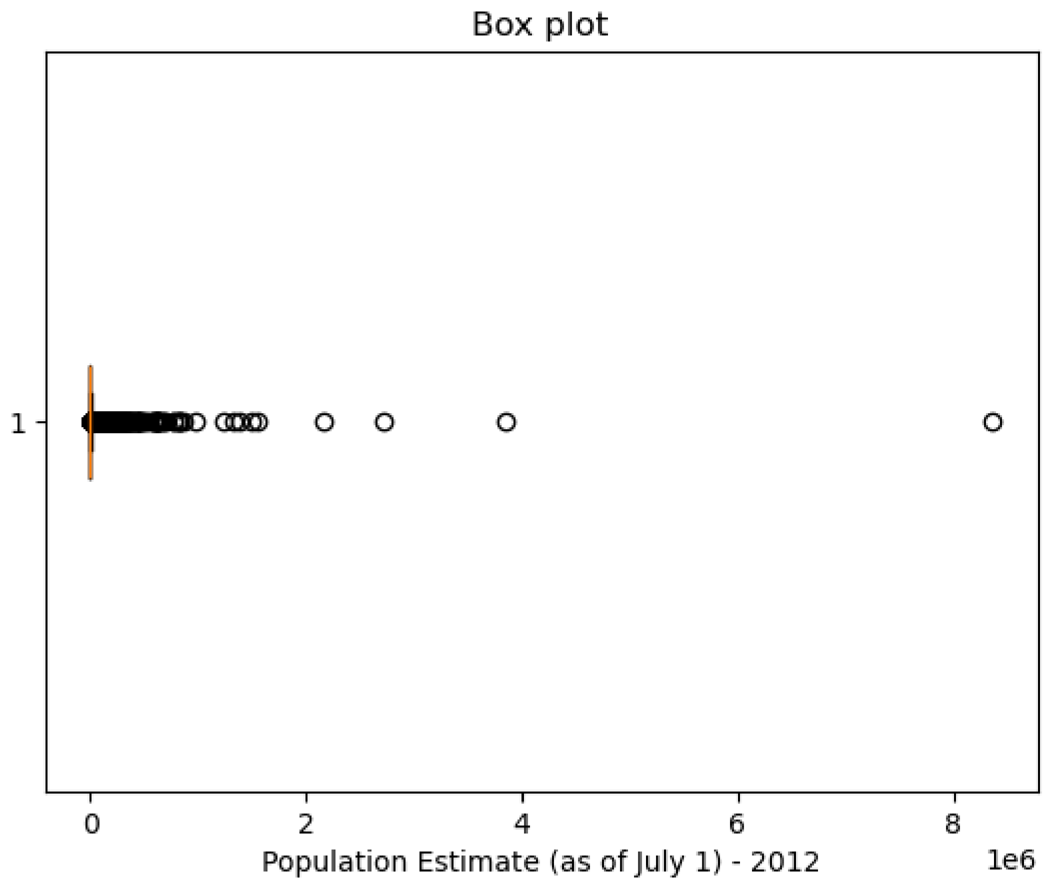
In [34]:
```
for i in num_cols[1:]:
    data=df[i]
    plt.boxplot(data,vert=False)
    plt.title("Box plot")
    plt.xlabel(i)
    plt.show()
```


Box plot
1
0
1
2
3
4
5
6
7
8
1e6
April 1, 2010 - Estimates Base

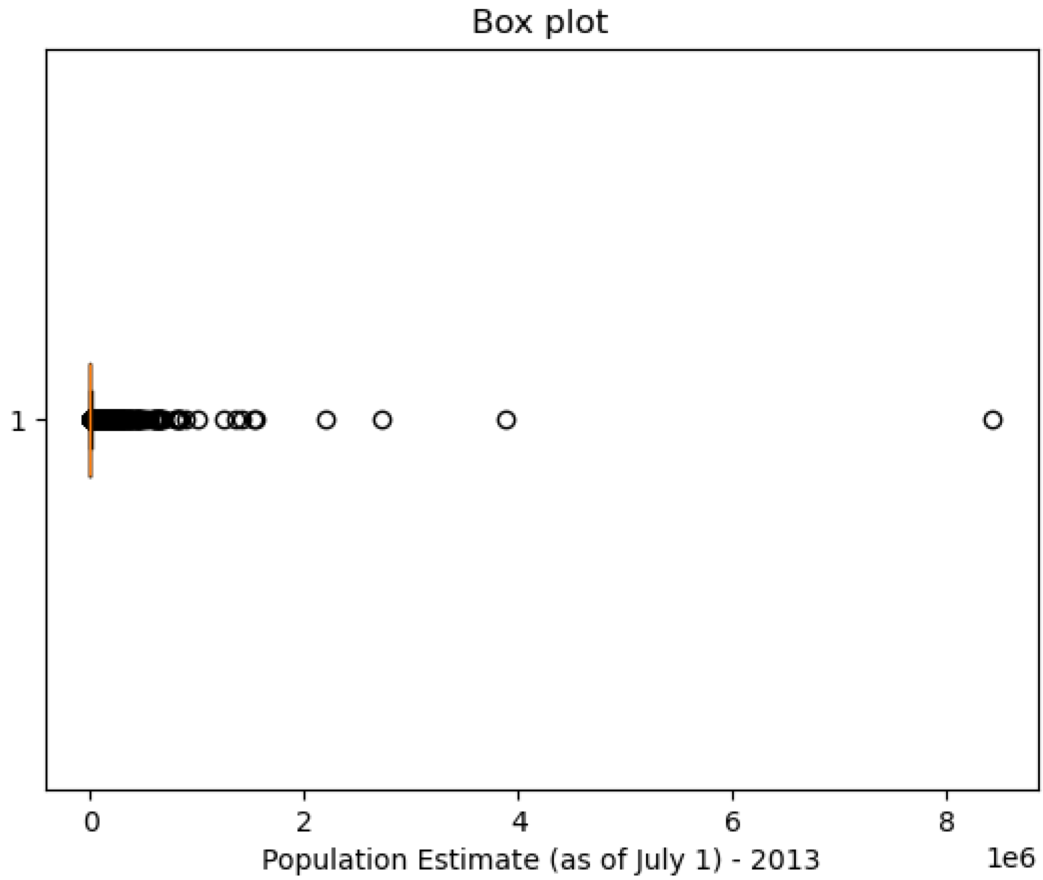

Box plot
1
0
2
4
6
8
1e6
Population Estimate (as of July 1) - 2010

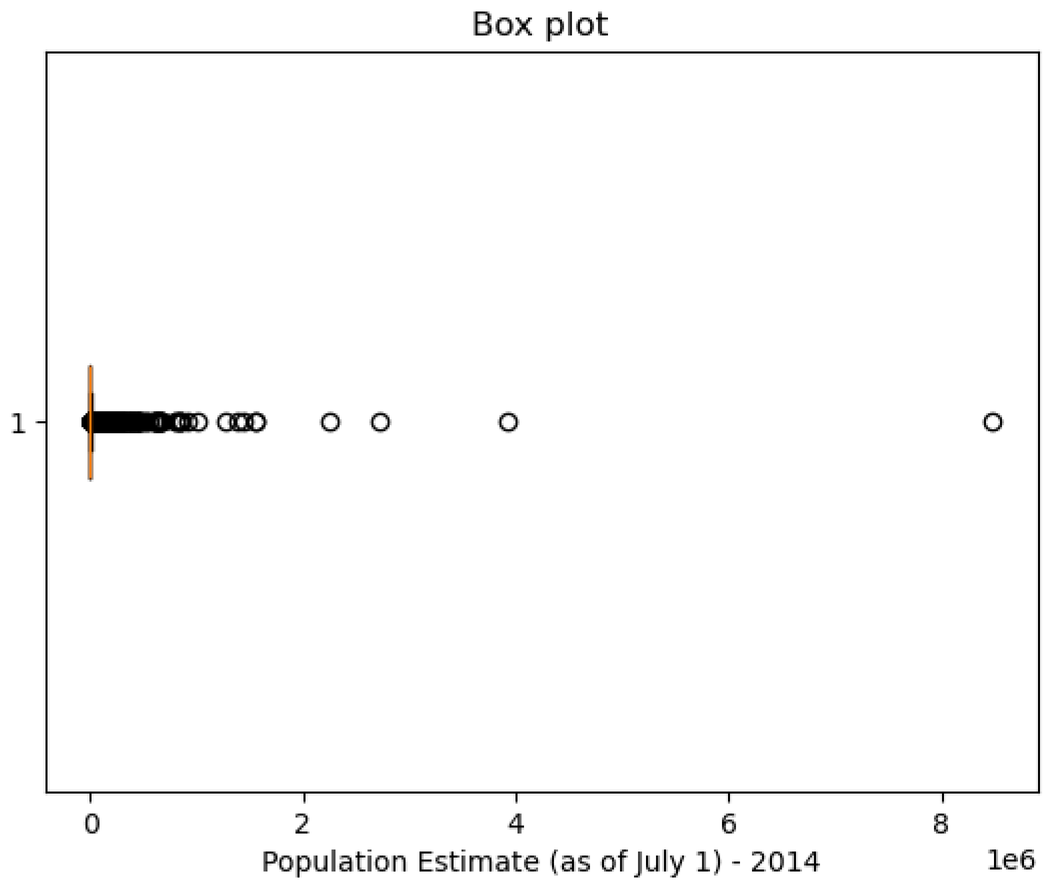


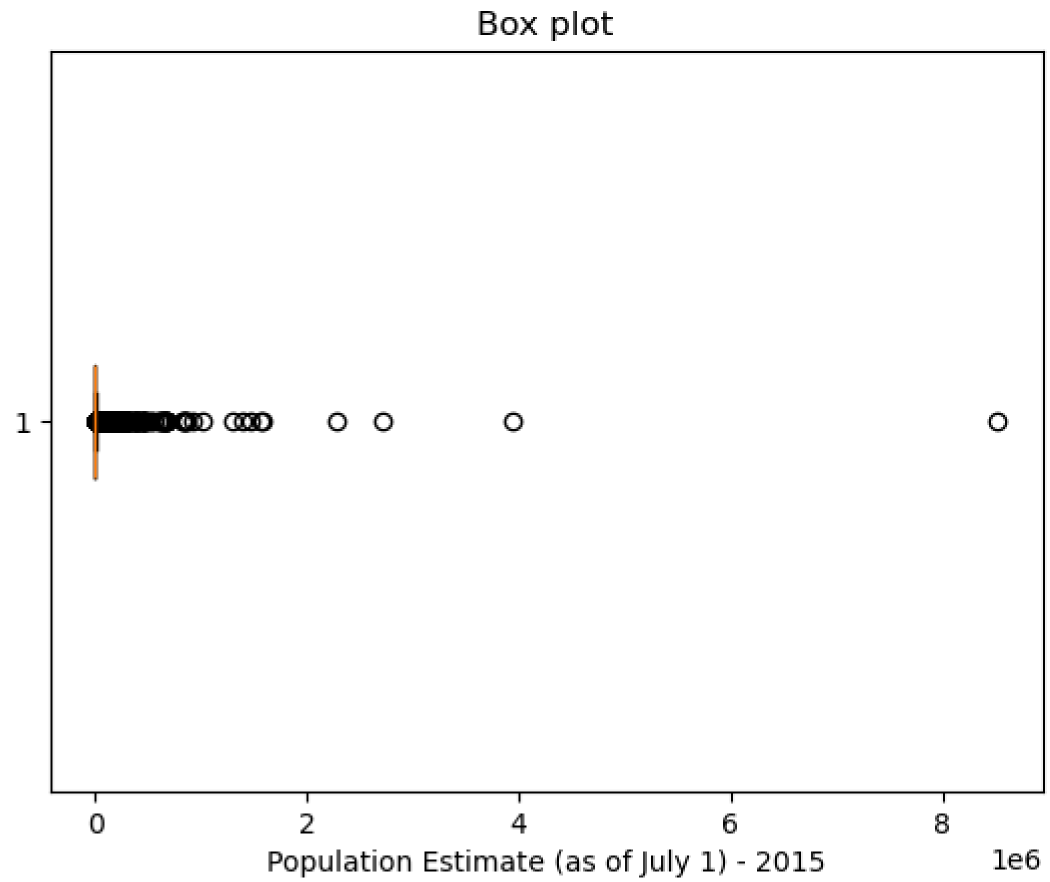
Box plot
1
0
2
4
6
8
Population Estimate (as of July 1) - 2011
1e6

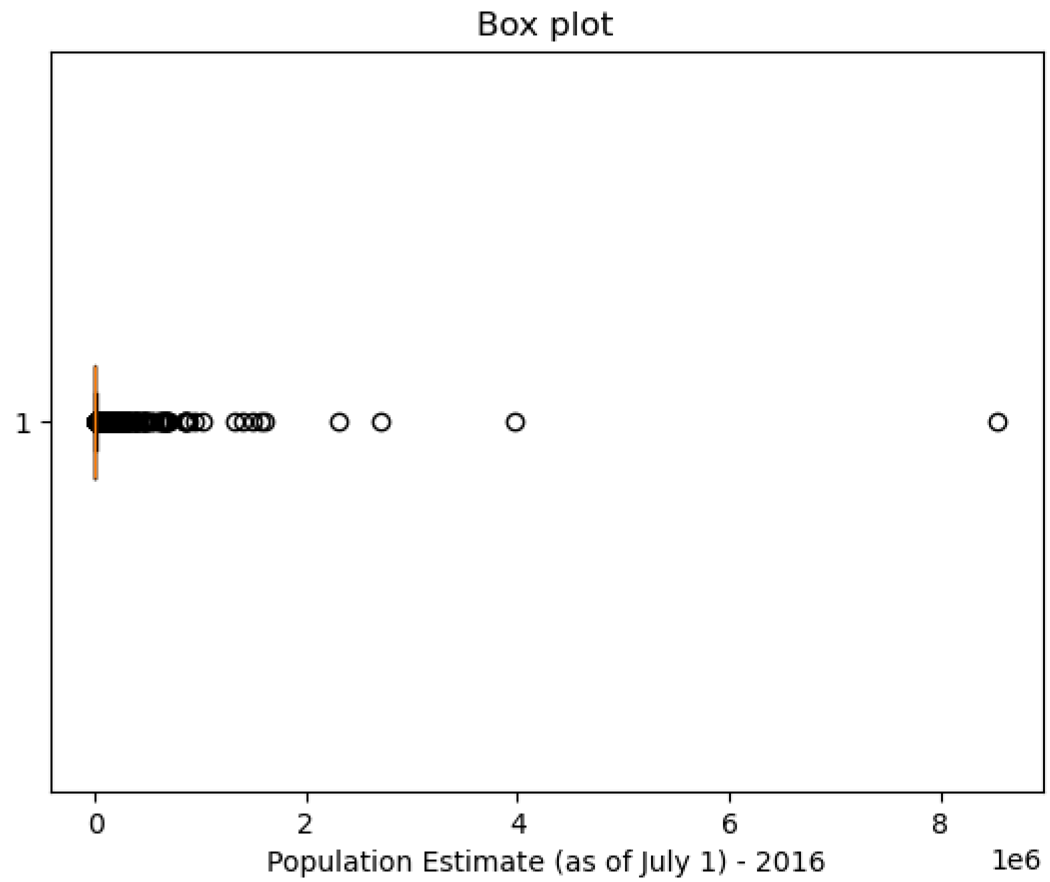


Box plot
1
0
2
4
6
8
Population Estimate (as of July 1) - 2012
1e6


Box plot
1
0
2
4
6
8
Population Estimate (as of July 1) - 2013
1e6


Box plot
1
0
2
4
6
8
Population Estimate (as of July 1) - 2014
1e6





Removal Of Outliers

By Uisng IQR(Inter Quartile Range) techinque

In [35]:
```python
# by using IQR we are able to remove outliers in data
data=df['April 1, 2010 - Estimates Base']
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1

lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data
```

Out[35]:
```
0        2688
1        4522
2         756
3         356
7        2481
         ...
19505     451
19506    3627
19507    5487
19508    1807
19509     151
Name: April 1, 2010 - Estimates Base, Length: 16724, dtype: int64
```

In [36]:
```python
# data frame without any outliers
import warnings
warnings.filterwarnings('ignore')
non_outliers_df=df[con3]
non_outliers_df.dropna(inplace=True)
non_outliers_df
```

Out[36]:

	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Popula Estir (as of 1) - 2
0	1620000US0100124	100124	Abbeville city, Alabama	2688	2688	2683	
1	1620000US0100460	100460	Adamsville city, Alabama	4522	4522	4517	
2	1620000US0100484	100484	Addison town, Alabama	758	756	754	
3	1620000US0100676	100676	Akron town, Alabama	356	356	355	
7	1620000US0101228	101228	Aliceville city, Alabama	2486	2481	2478	
...	...	...	...	...	...	...	
19505	1620000US5681300	5681300	Wamsutter town, Wyoming	451	451	450	
19506	1620000US5683040	5683040	Wheatland town, Wyoming	3627	3627	3629	
19507	1620000US5684925	5684925	Worland city, Wyoming	5487	5487	5494	
19508	1620000US5685015	5685015	Wright town, Wyoming	1807	1807	1807	
19509	1620000US5686665	5686665	Yoder town, Wyoming	151	151	152	

16724 rows × 12 columns

In [37]:

```
len(non_outliers_df)
```

Out[37]: 16724

In [38]:

```
# by using IQR we are able to remove outliers in data
data=df[i]
q1=np.percentile(data,25)
q2=np.percentile(data,50)
q3=np.percentile(data,75)

IQR=q3-q1
```

```
lb=(q1-(1.5*IQR))
ub=(q3+(1.5*IQR))

con1=data>lb
con2=data<ub
con3=con1&con2
non_outliers_data=data[con3]
non_outliers_data
```

Out[38]:

```
0        2603
1        4360
2         738
3         334
7        2357
         ... 
19505     484
19506    3606
19507    5316
19508    1834
19509     159
Name: Population Estimate (as of July 1) - 2016, Length: 16703, dtype: int64
```

Extracted dataframe without any outliers

In [39]:

```
import warnings
warnings.filterwarnings('ignore')
non_outliers_df=df[con3]
non_outliers_df.dropna(inplace=True)
non_outliers_df
```

Out[39]:

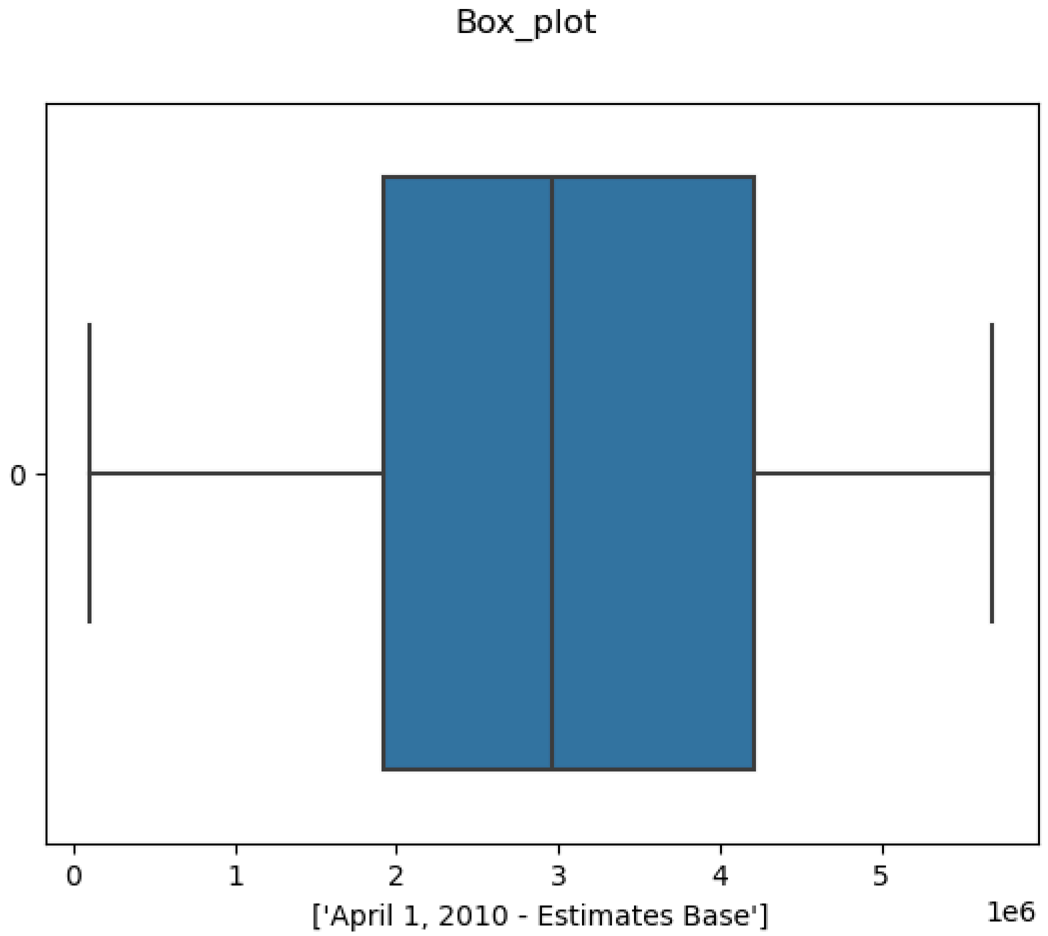
	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Popula Estir (as of 1) - 2
0	1620000US0100124	100124	Abbeville city, Alabama	2688	2688	2683	2
1	1620000US0100460	100460	Adamsville city, Alabama	4522	4522	4517	4
2	1620000US0100484	100484	Addison town, Alabama	758	756	754	
3	1620000US0100676	100676	Akron town, Alabama	356	356	355	
7	1620000US0101228	101228	Aliceville city, Alabama	2486	2481	2478	2
...	...	...	...	...	...	...	
19505	1620000US5681300	5681300	Wamsutter town, Wyoming	451	451	450	
19506	1620000US5683040	5683040	Wheatland town, Wyoming	3627	3627	3629	3
19507	1620000US5684925	5684925	Worland city, Wyoming	5487	5487	5494	5
19508	1620000US5685015	5685015	Wright town, Wyoming	1807	1807	1807	1
19509	1620000US5686665	5686665	Yoder town, Wyoming	151	151	152	

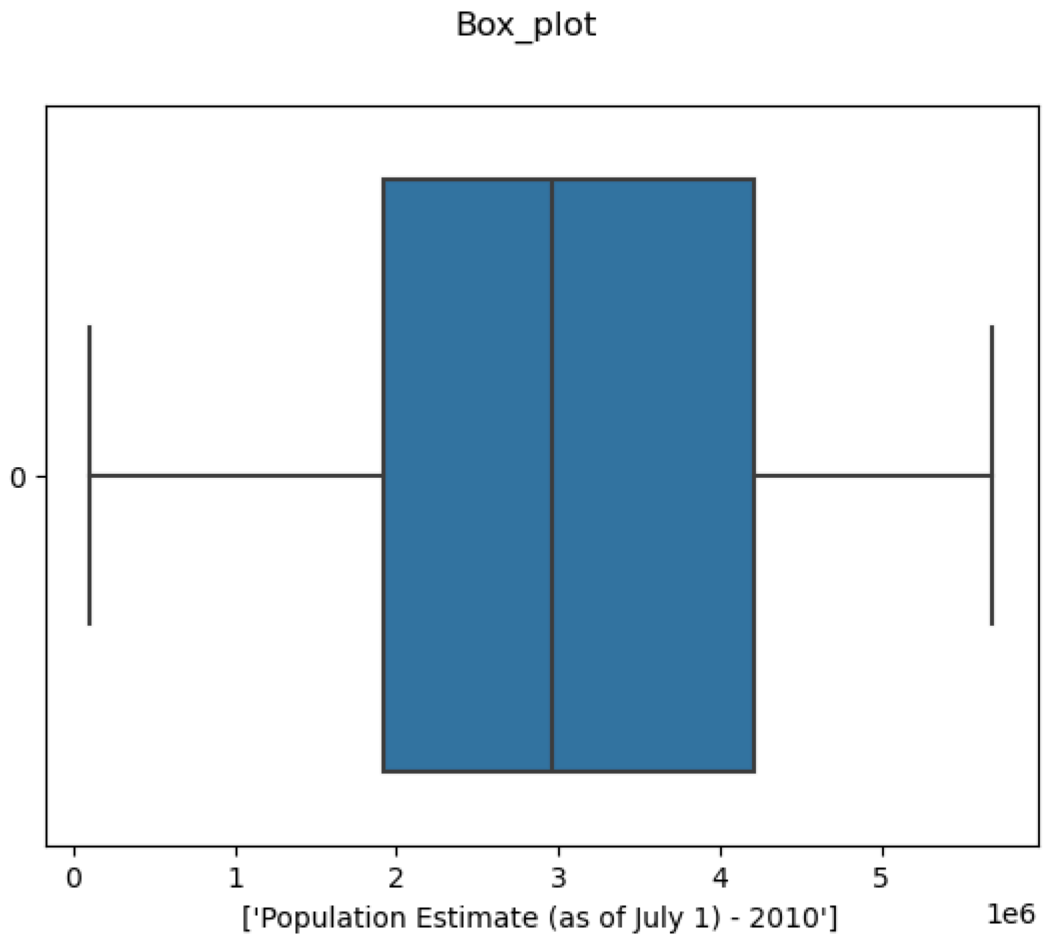
16703 rows × 12 columns

Box-plot Analysis without any outliers

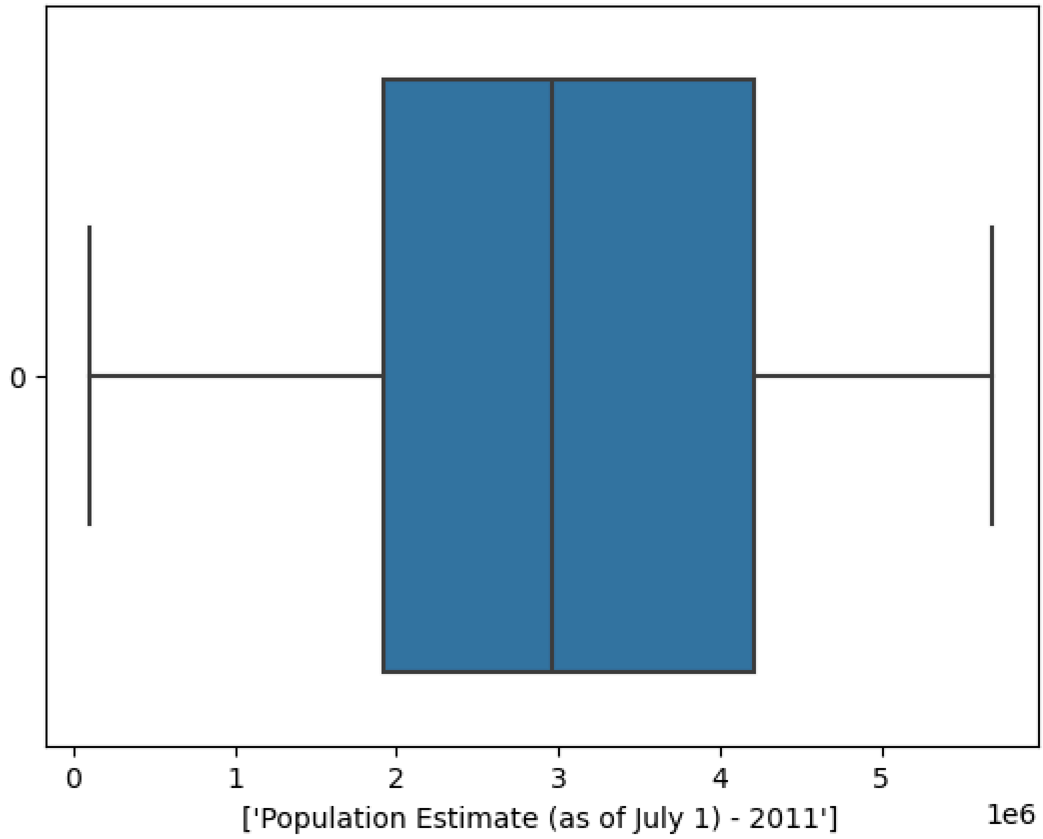
In [40]:

```
for i in num_cols[1:]:
    height=non_outliers_df['GEOID 2']
    sns.boxplot(height,orient='h')
    plt.xlabel([i])
    plt.suptitle('Box_plot')
    plt.show()
```

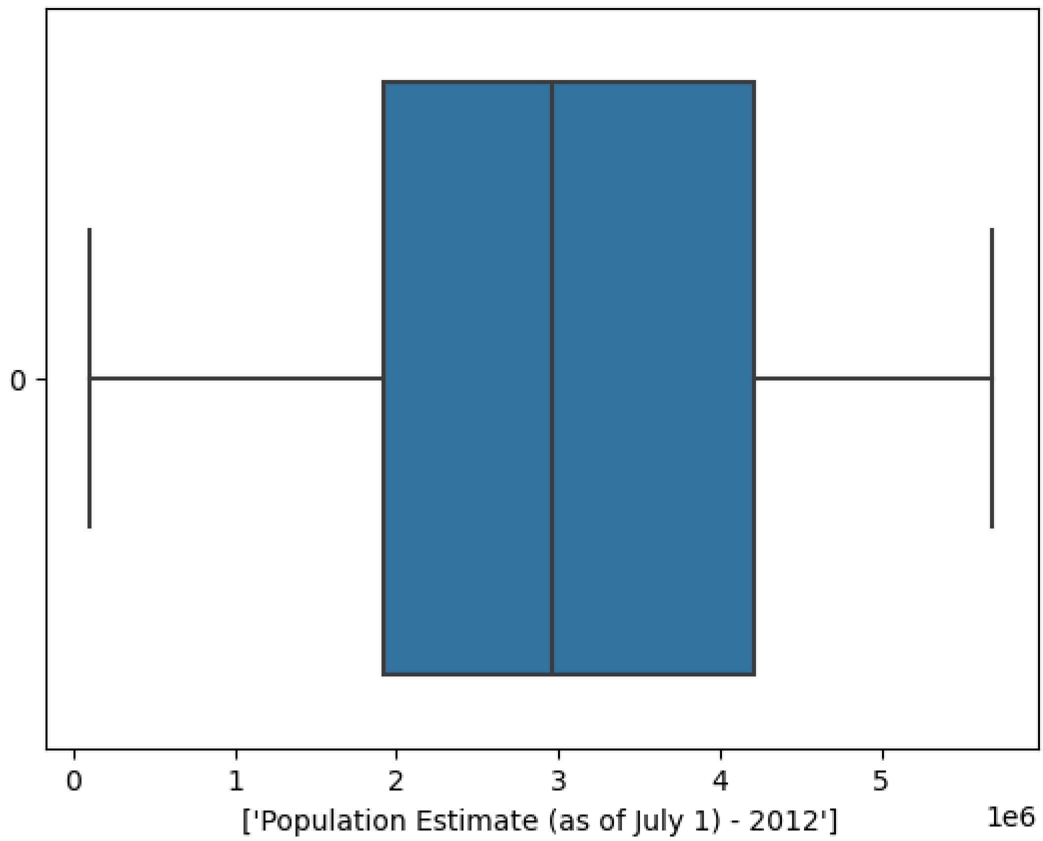

Box_plot
0
0
1
2
3
4
5
['April 1, 2010 - Estimates Base']
1e6

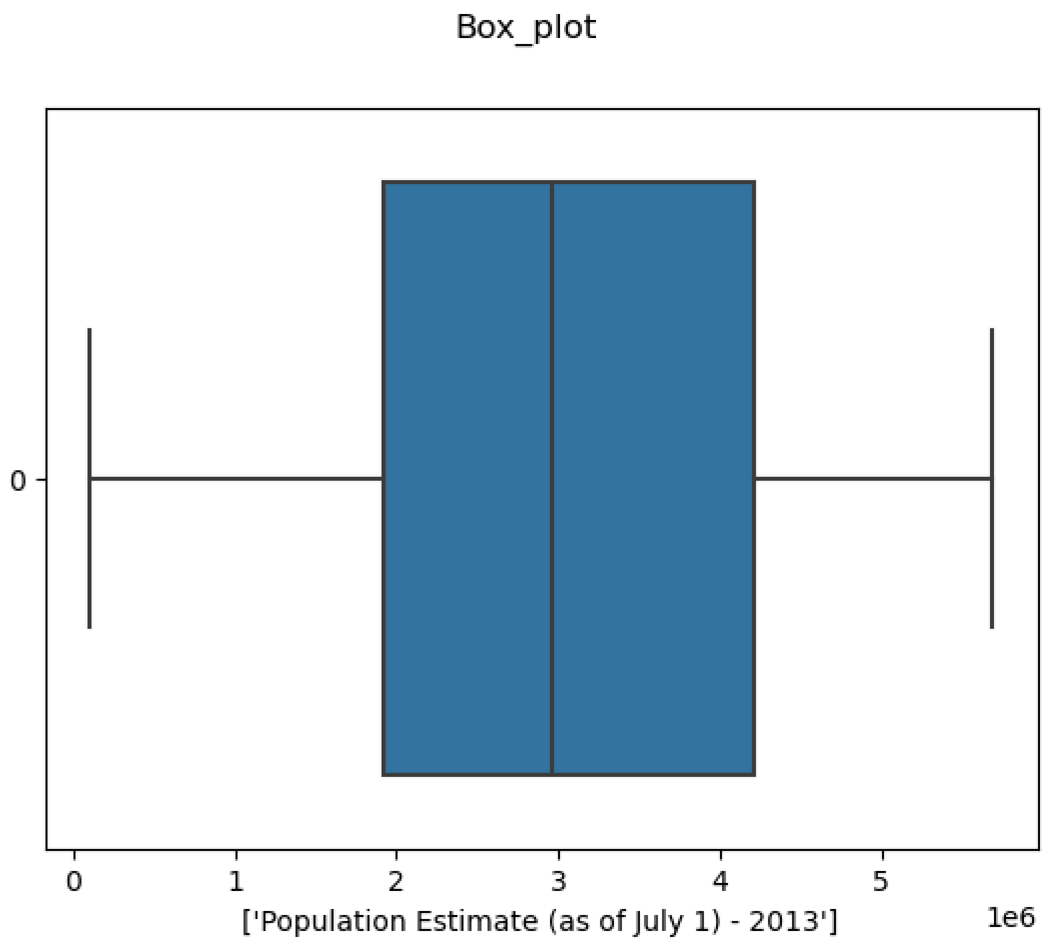

Box_plot
0
0
1
2
3
4
5
['Population Estimate (as of July 1) - 2010']
1e6

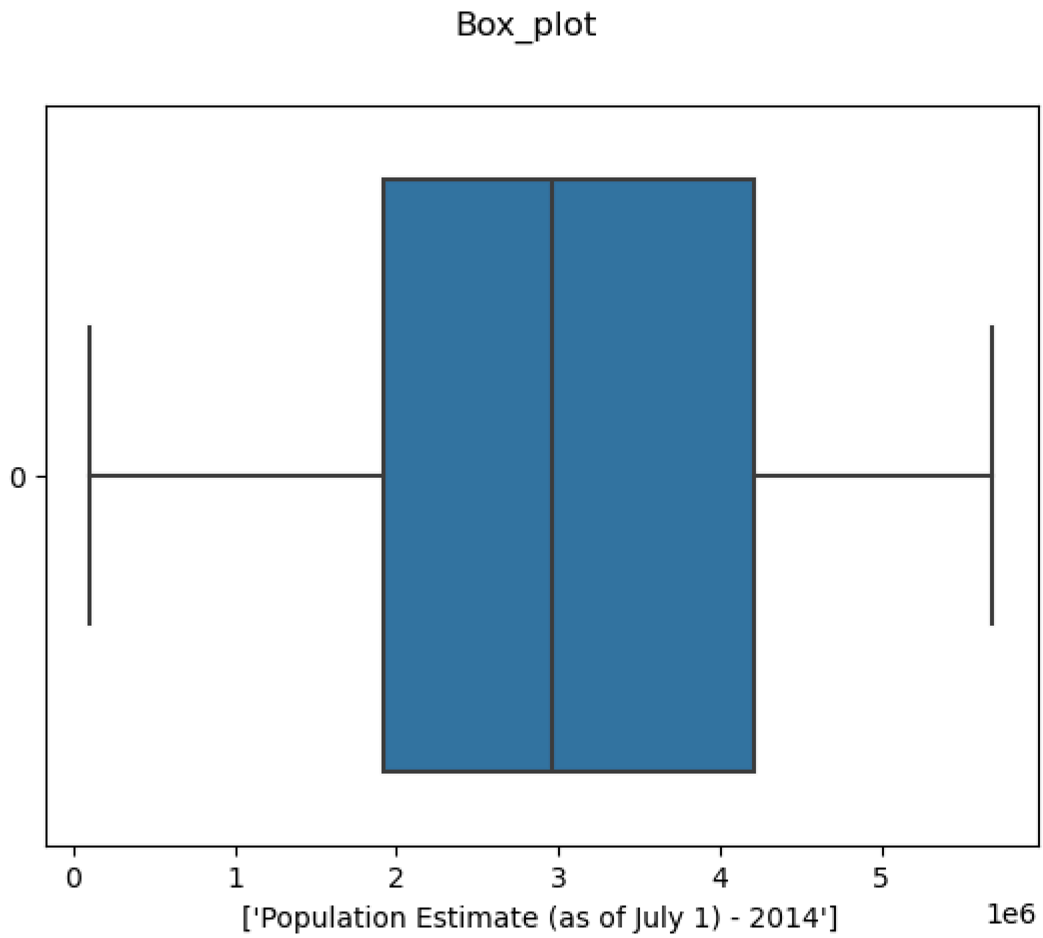
Box_plot

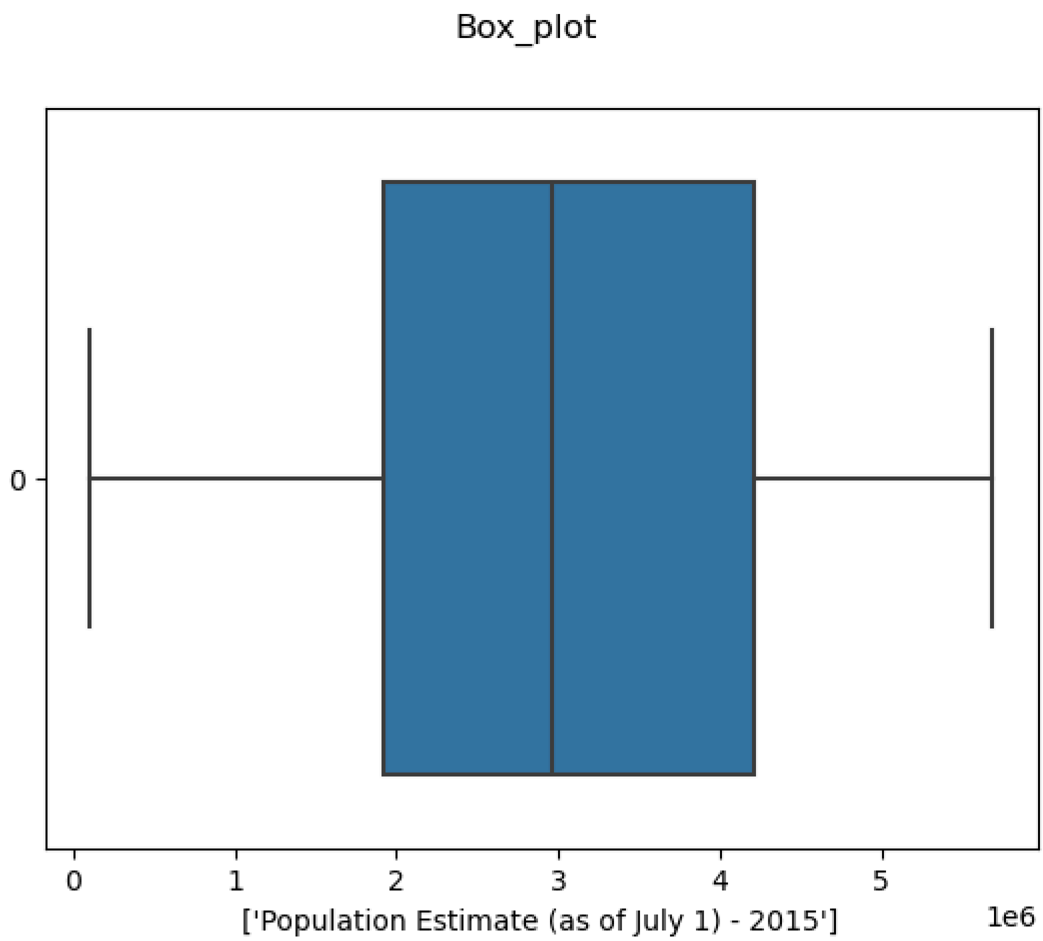


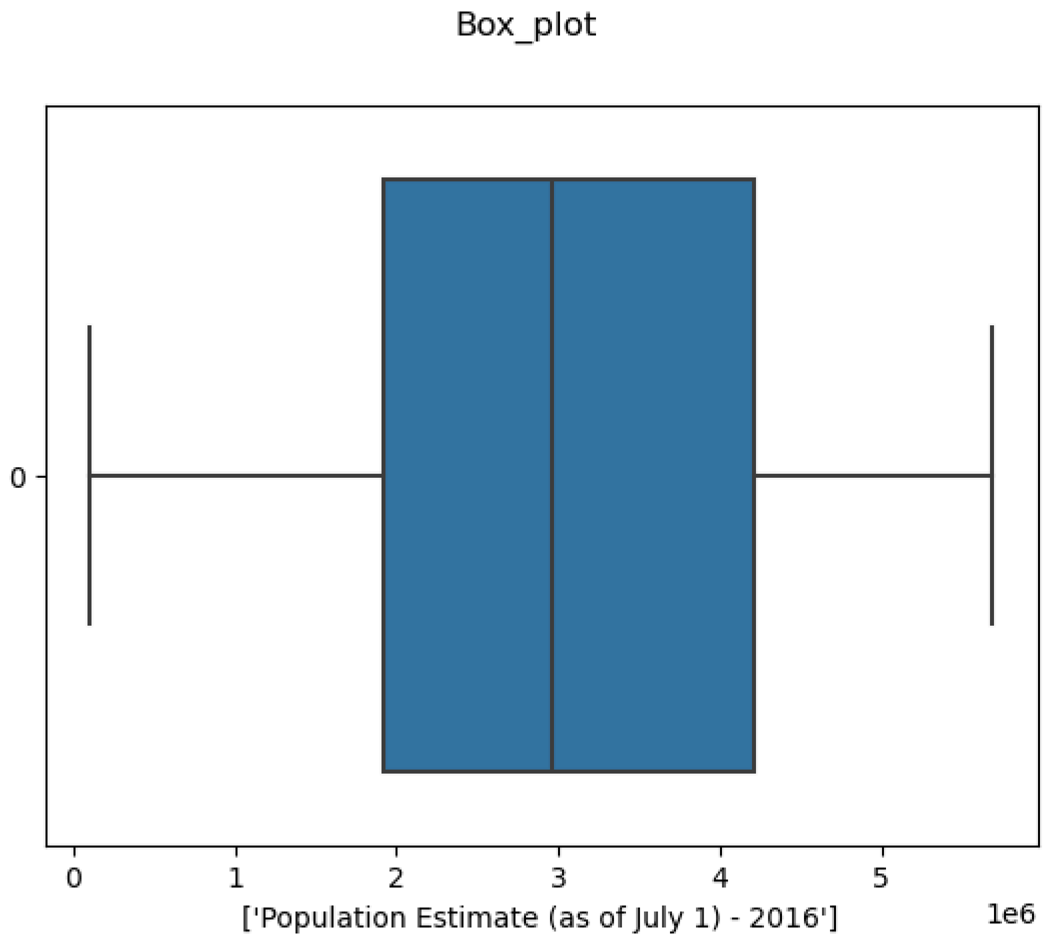
Box_plot




Box_plot
0
0
1
2
3
4
5
1e6
['Population Estimate (as of July 1) - 2013']


Box_plot
0
0
1
2
3
4
5
1e6
['Population Estimate (as of July 1) - 2014']





In [41]:
```
len(non_outliers_df)
```

Out[41]: 16703

Difference of dataframe with outliers and without outliers

In [42]:
```
count=len(df)-len(non_outliers_df)
count
```

Out[42]: 2807

In [43]:
```
print(f"The number of outliers in data is :{count}")
```

```
The number of outliers in data is :2807
```

Predefined function applied for non_outliers_data_frame

In [44]:
```
non_outliers_df.describe()
```

Out[44]:

	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012	Populati Estim (as of J 1) - 20
count	1.670300e+04	16703.000000	16703.000000	16703.000000	16703.000000	16703.00(
mean	3.052889e+06	1839.332036	1840.943962	1845.300665	1848.416572	1852.27(
std	1.429188e+06	2346.114447	2348.576636	2355.974554	2363.361227	2371.45(
min	1.001240e+05	0.000000	0.000000	0.000000	0.000000	0.00(
25%	1.922912e+06	307.000000	307.500000	307.000000	306.000000	305.00(
50%	2.956396e+06	828.000000	828.000000	829.000000	826.000000	825.00(
75%	4.207080e+06	2349.000000	2347.500000	2357.000000	2355.000000	2359.50(
max	5.686665e+06	12282.000000	12234.000000	12095.000000	12632.000000	12436.00(

Covariance Matrix

In [45]:
```
non_outliers_df.cov(numeric_only=True)
```

Out[45]:

	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012	I E
GEOID 2	2.042580e+12	1.233883e+08	1.234193e+08	1.254885e+08	1.275953e+08	1.3
April 1, 2010 - Estimates Base	1.233883e+08	5.504253e+06	5.509816e+06	5.526354e+06	5.542028e+06	5.5
Population Estimate (as of July 1) - 2010	1.234193e+08	5.509816e+06	5.515812e+06	5.532594e+06	5.548434e+06	5.5
Population Estimate (as of July 1) - 2011	1.254885e+08	5.526354e+06	5.532594e+06	5.550616e+06	5.567235e+06	5.5
Population Estimate (as of July 1) - 2012	1.275953e+08	5.542028e+06	5.548434e+06	5.567235e+06	5.585476e+06	5.6
Population Estimate (as of July 1) - 2013	1.306147e+08	5.558179e+06	5.564766e+06	5.584369e+06	5.603681e+06	5.6
Population Estimate (as of July 1) - 2014	1.330251e+08	5.578533e+06	5.585323e+06	5.605784e+06	5.626084e+06	5.6
Population Estimate (as of July 1) - 2015	1.359587e+08	5.596147e+06	5.603160e+06	5.624525e+06	5.645864e+06	5.6
Population Estimate (as of July 1) - 2016	1.382116e+08	5.617379e+06	5.624606e+06	5.646832e+06	5.669119e+06	5.6

correlation matrix

In [46]:

```
correlation_data=non_outliers_df.corr(numeric_only=True)
correlation_data
```

Out[46]:

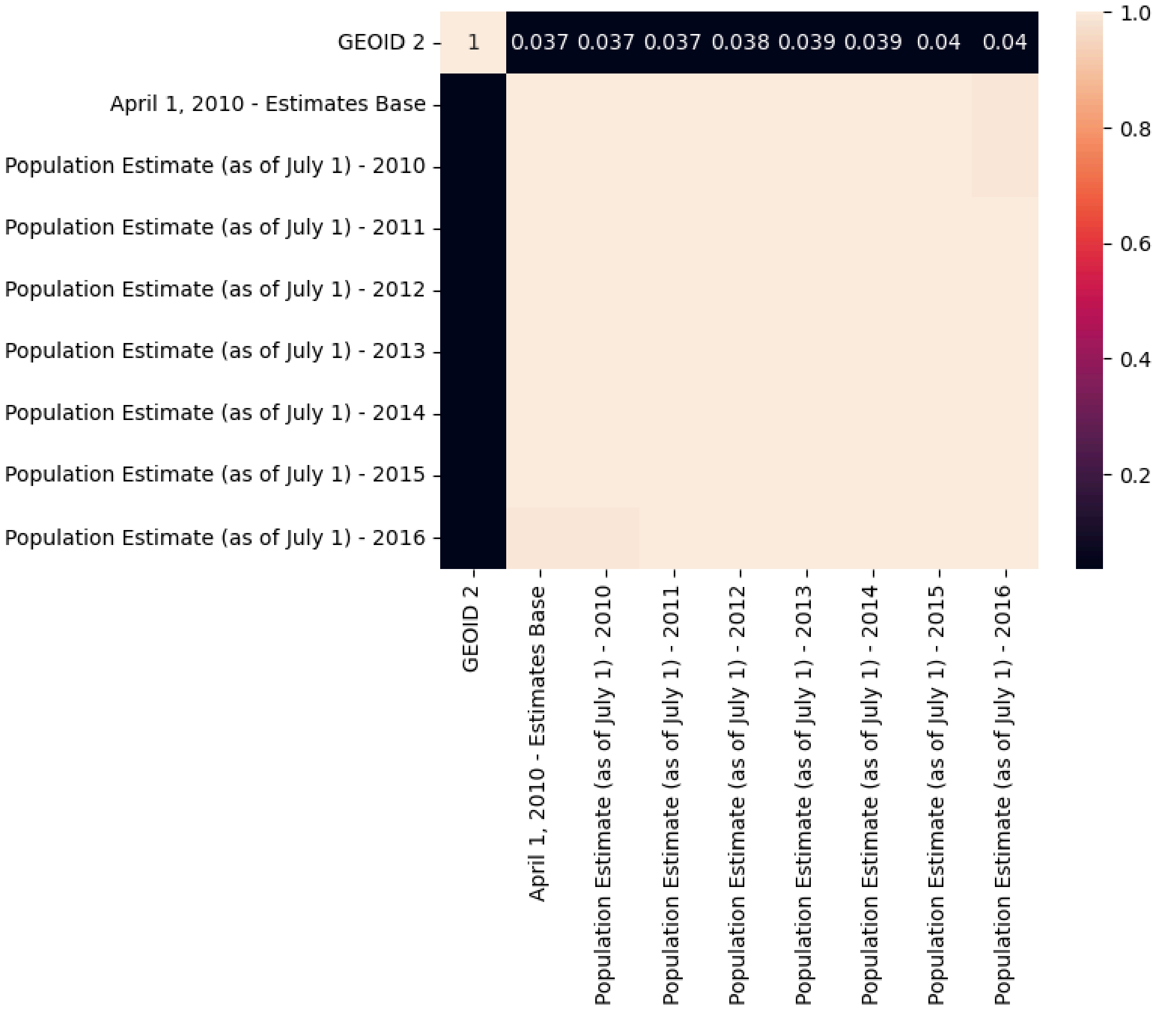
	GEOID 2	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Population Estimate (as of July 1) - 2011	Population Estimate (as of July 1) - 2012	Population Estimate (as of July 1) - 2013	Popul Esti (as o 1) -
GEOID 2	1.000000	0.036799	0.036770	0.037269	0.037776	0.038538	0.03
April 1, 2010 - Estimates Base	0.036799	1.000000	0.999961	0.999813	0.999515	0.999005	0.99
Population Estimate (as of July 1) - 2010	0.036770	0.999961	1.000000	0.999893	0.999621	0.999140	0.99
Population Estimate (as of July 1) - 2011	0.037269	0.999813	0.999893	1.000000	0.999859	0.999511	0.99
Population Estimate (as of July 1) - 2012	0.037776	0.999515	0.999621	0.999859	1.000000	0.999833	0.99
Population Estimate (as of July 1) - 2013	0.038538	0.999005	0.999140	0.999511	0.999833	1.000000	0.99
Population Estimate (as of July 1) - 2014	0.039076	0.998242	0.998410	0.998921	0.999405	0.999802	1.00
Population Estimate (as of July 1) - 2015	0.039765	0.997063	0.997266	0.997925	0.998580	0.999225	0.99
Population Estimate (as of July 1) - 2016	0.040213	0.995624	0.995860	0.996656	0.997462	0.998320	0.99

heat map visualization

In [47]:

```
# Here there exist a positive relation between the variables ao each column

corr_data=non_outliers_df.corr(numeric_only=True)
sns.heatmap(corr_data,annot=True)
plt.show()
```



Clean Data Without Any Outliers And Missing Values

In [48]:
```
non_outliers_df
```

Out[48]:

	Geographic ID	GEOID 2	Geography, full name (City, State)	April 1, 2010 - Census	April 1, 2010 - Estimates Base	Population Estimate (as of July 1) - 2010	Popula Estir (as of 1) - 2
0	1620000US0100124	100124	Abbeville city, Alabama	2688	2688	2683	
1	1620000US0100460	100460	Adamsville city, Alabama	4522	4522	4517	
2	1620000US0100484	100484	Addison town, Alabama	758	756	754	
3	1620000US0100676	100676	Akron town, Alabama	356	356	355	
7	1620000US0101228	101228	Aliceville city, Alabama	2486	2481	2478	
...	...	...	...	...	...	...	
19505	1620000US5681300	5681300	Wamsutter town, Wyoming	451	451	450	
19506	1620000US5683040	5683040	Wheatland town, Wyoming	3627	3627	3629	
19507	1620000US5684925	5684925	Worland city, Wyoming	5487	5487	5494	
19508	1620000US5685015	5685015	Wright town, Wyoming	1807	1807	1807	
19509	1620000US5686665	5686665	Yoder town, Wyoming	151	151	152	

16703 rows × 12 columns

In [ ]: