Data Science Capstone Project

Real Estate.

In [1]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Import data

In [2]:
```
train_df=pd.read_csv('train-RE.csv')
```

In [3]:
```
test_df=pd.read_csv('test-RE.csv')
```

In [4]:
```
train_df.head()
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	femal
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	

5 rows × 80 columns

In [5]:
```
test_df.head()
```

Out[5]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...

5 rows × 80 columns

In [6]:
```
#checking shape of dataset
train_df.shape, test_df.shape
```

Out[6]:
```
((27321, 80), (11709, 80))
```

Figure out the primary key and look for the requirement of indexing

In [7]:
```
#Checking duplicate records
print('Duplicates in training dataset :')
print(train_df.duplicated().value_counts(),'\n')

print('Duplicates in testing dataset :')
print(test_df.duplicated().value_counts(),'\n')
```

```
Duplicates in training dataset :
False    27161
True       160
dtype: int64

Duplicates in testing dataset :
False    11677
True        32
dtype: int64
```

In [8]:
```
# Removing the duplicates from the dataset

train_df.drop_duplicates(keep = 'first', inplace=True)
test_df.drop_duplicates(keep = 'first', inplace=True)
```

In [9]:
```
# checking shape of dataset after removing duplicates
train_df.shape, test_df.shape
```

Out[9]:
```
((27161, 80), (11677, 80))
```

In [10]:
```
#Checking Unique value for primary key
train_df.nunique()==train_df.shape[0]
```

Out[10]:
```
UID             True
BLOCKID        False
SUMLEVEL       False
COUNTYID       False
STATEID        False
               ...
pct_own        False
married        False
married_snp    False
separated      False
divorced       False
Length: 80, dtype: bool
```

In [11]:
```
test_df.nunique()==test_df.shape[0]
```

Out[11]:
```
UID             True
BLOCKID        False
SUMLEVEL       False
COUNTYID       False
STATEID        False
               ...
```

```
pct_own         False
married         False
married_snp     False
separated       False
divorced        False
Length: 80, dtype: bool
```

In [12]:
```
train_df.nunique()
```

Out[12]:
```
UID             27161
BLOCKID             0
SUMLEVEL            1
COUNTYID          296
STATEID            52
                ...
pct_own         22302
married         20282
married_snp     10350
separated        6190
divorced        13688
Length: 80, dtype: int64
```

Since UID has all unique values and it matches the number of rows, UID can be used as the primary key in the data set

Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.

In [13]:
```
train_df.isnull().sum()
```

Out[13]:
```
UID                 0
BLOCKID         27161
SUMLEVEL            0
COUNTYID            0
STATEID             0
                ...
pct_own           207
married           150
married_snp       150
separated         150
divorced          150
Length: 80, dtype: int64
```

In [14]:
```
test_df.isnull().sum()
```

Out[14]:
```
UID                 0
BLOCKID         11677
SUMLEVEL            0
COUNTYID            0
STATEID             0
                ...
pct_own           112
married            77
married_snp        77
separated          77
divorced           77
Length: 80, dtype: int64
```

In [15]:
```
# Block ID column has all missing values, and SUMLEVEL and primary each have single valu
```

```
train_df.drop(columns=['BLOCKID', 'SUMLEVEL','primary'], axis = 1, inplace=True)
test_df.drop(columns=['BLOCKID', 'SUMLEVEL','primary'], axis = 1, inplace=True)
```

In [16]:
```
train_df.shape, test_df.shape
```

Out[16]:
```
((27161, 77), (11677, 77))
```

In [17]:
```
train_df['data_type'] = 'Train'
test_df['data_type'] = 'Test'
```

In [18]:
```
#Combining datasets
combined_df = train_df.append(test_df, ignore_index=True)
```

In [19]:
```
combined_df.head()
```

Out[19]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	...	female
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	...	
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	...	
2	245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	...	
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	...	
4	247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	...	

5 rows × 78 columns

In [20]:
```
combined_df.shape
```

Out[20]:
```
(38838, 78)
```

In [21]:
```
# checking percentage of missing values

(combined_df.isna().sum()/len(combined_df))*100
```

Out[21]:
```
UID            0.000000
COUNTYID       0.000000
STATEID        0.000000
state          0.000000
state_ab       0.000000
                 ...
married        0.584479
married_snp    0.584479
separated      0.584479
divorced       0.584479
data_type      0.000000
Length: 78, dtype: float64
```

In [22]:
```
col_check = combined_df.isna().sum().to_frame().reset_index()
col_check
```

Out[22]:

	index	0
0	UID	0
1	COUNTYID	0
2	STATEID	0

3	state	0
4	state_ab	0
...	...	...
73	married	227
74	married_snp	227
75	separated	227
76	divorced	227
77	data_type	0

78 rows × 2 columns

In [23]:
```
#columns with null values

null_col = col_check[col_check[0]>0]['index'].tolist()
null_col
```

Out[23]:
```
['rent_mean',
 'rent_median',
 'rent_stdev',
 'rent_sample_weight',
 'rent_samples',
 'rent_gt_10',
 'rent_gt_15',
 'rent_gt_20',
 'rent_gt_25',
 'rent_gt_30',
 'rent_gt_35',
 'rent_gt_40',
 'rent_gt_50',
 'hi_mean',
 'hi_median',
 'hi_stdev',
 'hi_sample_weight',
 'hi_samples',
 'family_mean',
 'family_median',
 'family_stdev',
 'family_sample_weight',
 'family_samples',
 'hc_mortgage_mean',
 'hc_mortgage_median',
 'hc_mortgage_stdev',
 'hc_mortgage_sample_weight',
 'hc_mortgage_samples',
 'hc_mean',
 'hc_median',
 'hc_stdev',
 'hc_samples',
 'hc_sample_weight',
 'home_equity_second_mortgage',
 'second_mortgage',
 'home_equity',
 'debt',
 'second_mortgage_cdf',
 'home_equity_cdf',
 'debt_cdf',
 'hs_degree',
 'hs_degree_male',
 'hs_degree_female',
```

```
 'male_age_mean',
 'male_age_median',
 'male_age_stdev',
 'male_age_sample_weight',
 'male_age_samples',
 'female_age_mean',
 'female_age_median',
 'female_age_stdev',
 'female_age_sample_weight',
 'female_age_samples',
 'pct_own',
 'married',
 'married_snp',
 'separated',
 'divorced']
```

In [24]:
```
#Filling the missing value with Median value

for i in null_col:
    combined_df[i].fillna(combined_df[i].median(), inplace=True)
```

In [25]:
```
combined_df.isnull().sum().any()
```

Out[25]:
```
False
```

In [26]:
```
#In pop column, there are some records for which the value is 0 which need to be removed
print('Number of observations with 0 Population = ', (combined_df['pop']==0).sum())
```

```
Number of observations with 0 Population =  216
```

In [27]:
```
combined_df = combined_df.drop(combined_df[combined_df['pop']==0].index).reset_index(dro
```

Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map. You may keep the upper limit for the percent of households with a second mortgage to 50 percent

In [28]:
```
# Sorting the data in decending order for second mortgage
top_second_mortgage = combined_df.sort_values(by=['second_mortgage'],ascending=False)
```

In [29]:
```
top_second_mortgage[(top_second_mortgage['second_mortgage'] <= 0.5)
                    & (top_second_mortgage['pct_own'] > 0.1)][['state','city','place']].
```

Out[29]:

	state	city	place
3258	Virginia	Farmville	Farmville
11860	Massachusetts	Worcester	Worcester City
28218	Oklahoma	Edmond	Edmond City
25737	New York	Corona	Harbor Hills
7754	Maryland	Glen Burnie	Glen Burnie
2060	Florida	Tampa	Egypt Lake-leto
1689	Illinois	Chicago	Lincolnwood
31958	Maryland	Adelphi	Adelphi

11723	Illinois	Chicago	Chicago City
8781	Michigan	Lansing	Lansing City
6422	Wisconsin	Milwaukee	Milwaukee City
11544	California	Etiwanda	Rancho Cucamonga City
37971	Pennsylvania	Philadelphia	Millbourne
20978	California	South San Francisco	San Bruno City
29025	New York	Bronx	Mount Vernon City
8022	Ohio	Cincinnati	Cincinnati City
23527	Texas	Dallas	Dallas City
28469	Virginia	Annandale	Ravensworth
36471	California	Sacramento	Parkway
28765	Massachusetts	Dorchester	Milton
10822	Colorado	Colorado Springs	Colorado Springs City
10228	Colorado	Littleton	Louviers
9979	California	Napa	Napa City
38205	Colorado	Northglenn	Northglenn City
8362	Ohio	East Cleveland	East Cleveland City

Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage

In [30]:

```
# Equation for bad debt
combined_df['bad_debt'] = (combined_df['second_mortgage'] +
                           combined_df['home_equity'] -
                           combined_df['home_equity_second_mortgage'])

combined_df[['bad_debt']].head()
```

Out[30]:

	bad_debt
0	0.09408
1	0.04274
2	0.09512
3	0.01086
4	0.05426

Create pie charts to show overall debt and bad

In [31]:

```
overall_debt = []
debt = combined_df['debt'].sum()
overall_debt.append(debt)
```

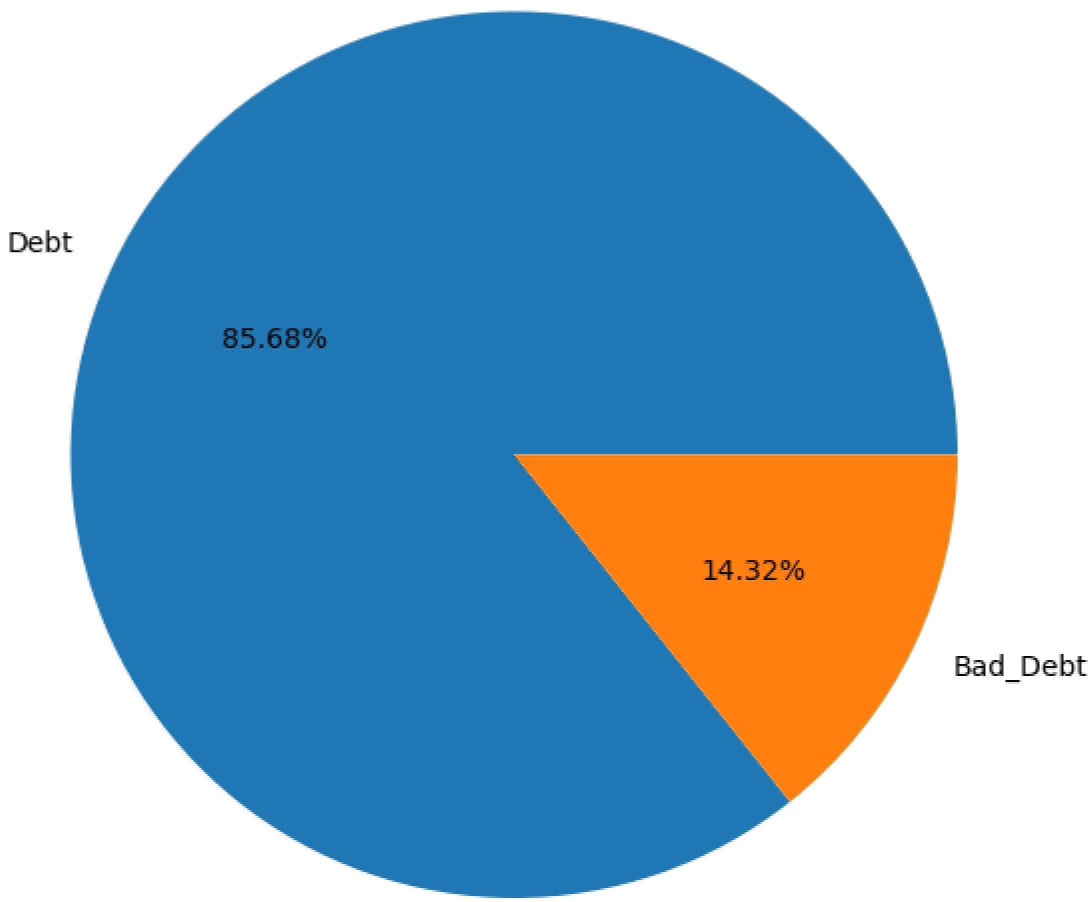
```
bad_debt = combined_df['bad_debt'].sum()
overall_debt.append(bad_debt)
```

In [32]:
```
overall_debt
```

Out[32]:
```
[24348.5801, 4068.6565]
```

In [33]:
```
print("Pie chart for overall debt and bad debt : \n")
plt.pie(overall_debt, labels=['Debt', 'Bad_Debt'], autopct='%1.2f%%', radius=1.5)
plt.show()
```

```
Pie chart for overall debt and bad debt :
```



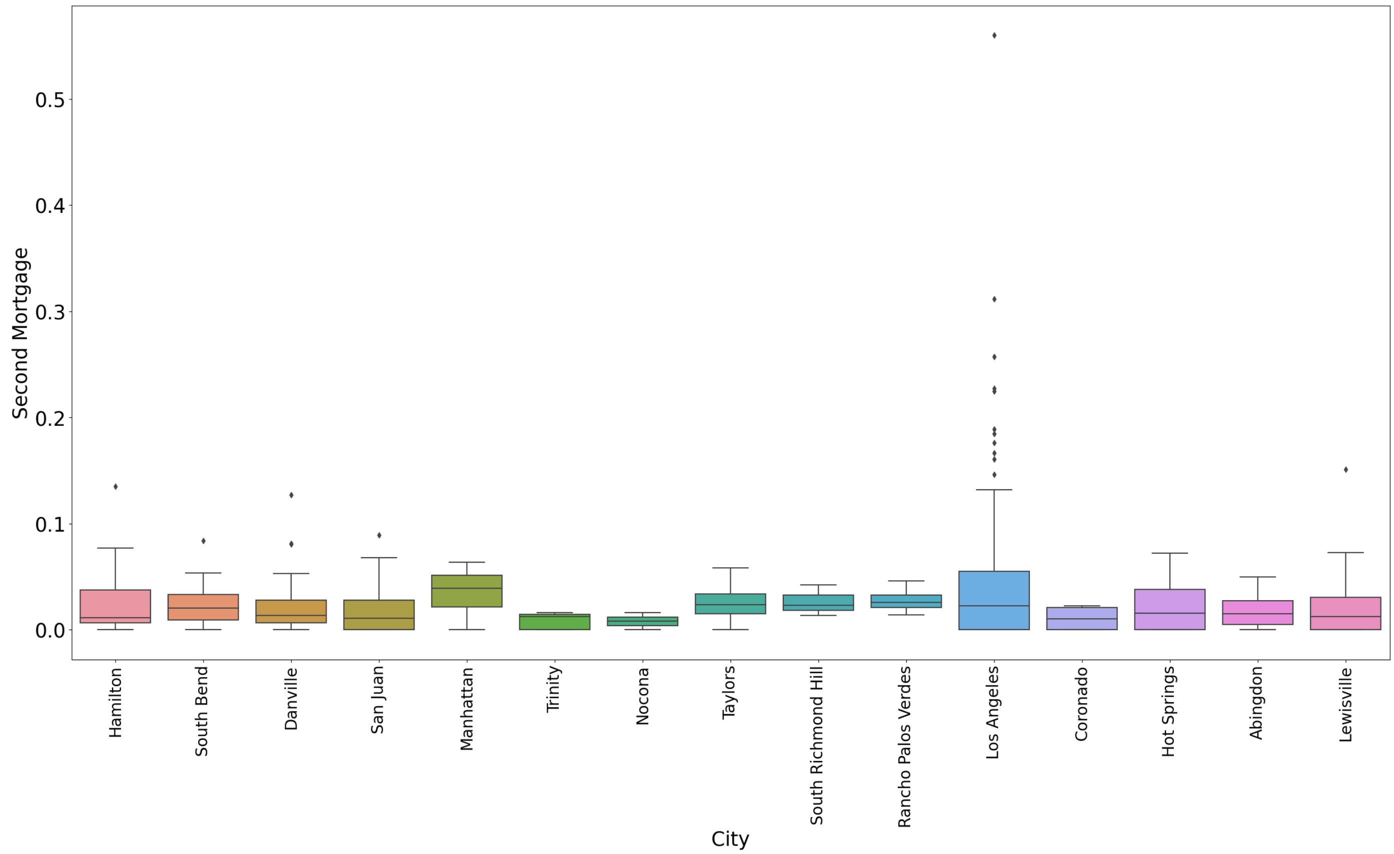
Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities

In [34]:
```
# Selecting 15 unique cities out of total cities
cities = combined_df['city'].unique()[0:15]
```

In [35]:
```
df = combined_df.loc[combined_df['city'].isin(cities)]
```

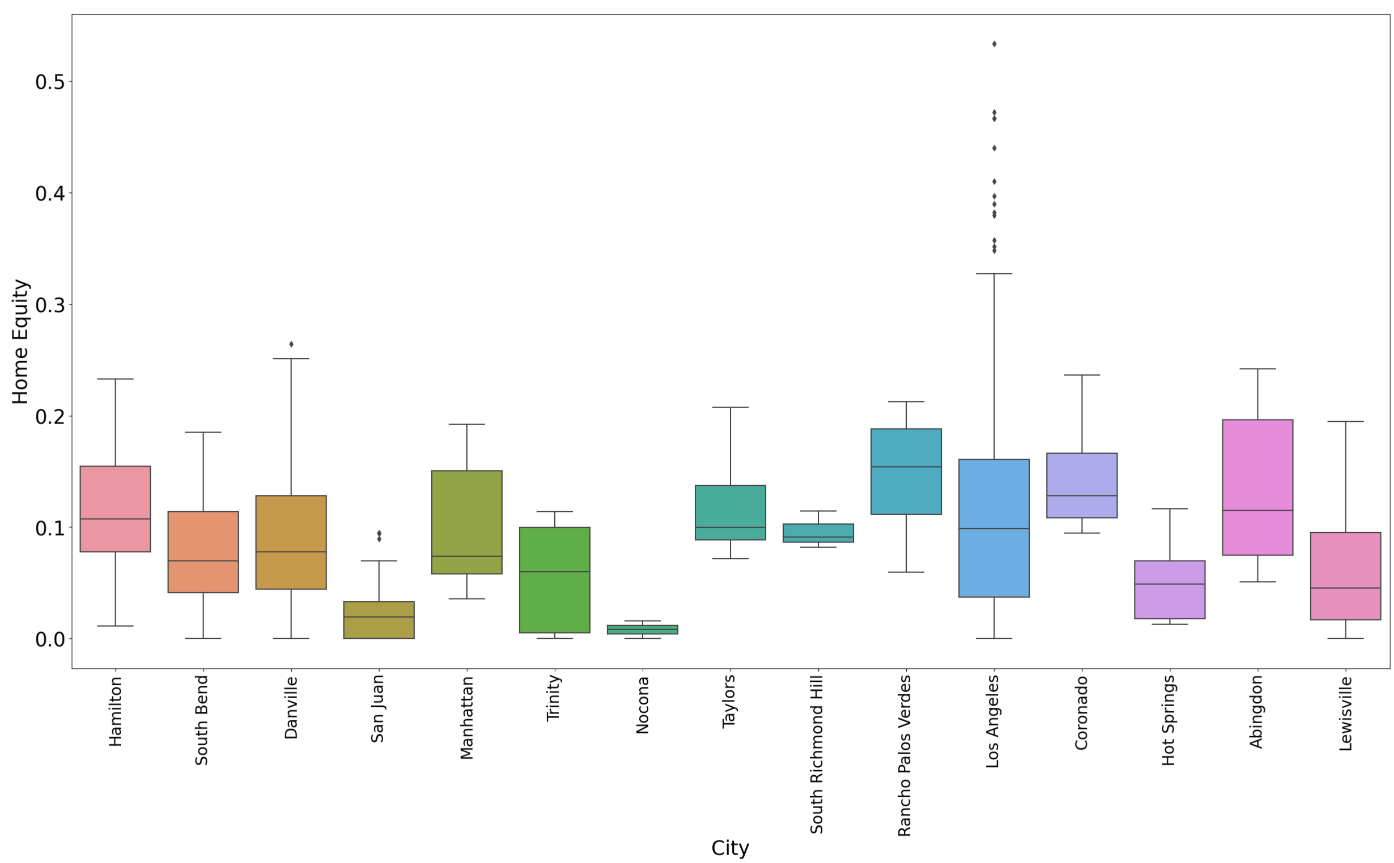
In [36]:
```
#Distribution for Second mortgage
plt.figure(figsize = (30, 15))
sns.boxplot(x = df['city'], y = df['second_mortgage'])
plt.xticks(rotation = 90, fontsize = 20)
plt.yticks(fontsize = 25)
plt.xlabel('City', fontsize = 25)
plt.ylabel('Second Mortgage', fontsize = 25)
```

Out[36]:
```
Text(0, 0.5, 'Second Mortgage')
```



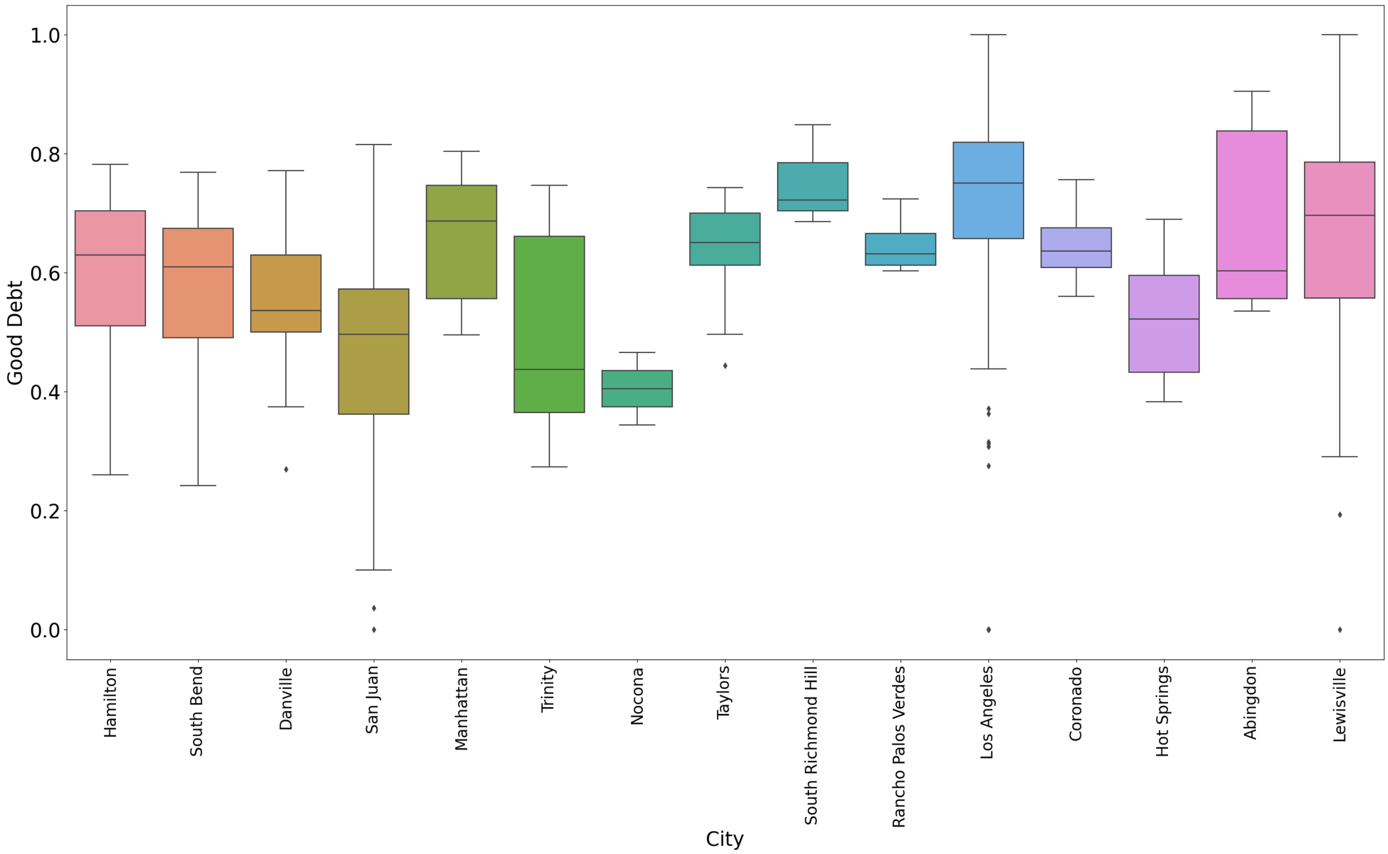
In [37]:
```
#Distribution for home equity
plt.figure(figsize = (30, 15))
sns.boxplot(x = df['city'], y = df['home_equity'])
plt.xticks(rotation = 90, fontsize = 20)
plt.yticks(fontsize = 25)
plt.xlabel('City', fontsize = 25)
plt.ylabel('Home Equity', fontsize = 25)
```

Out[37]:
```
Text(0, 0.5, 'Home Equity')
```



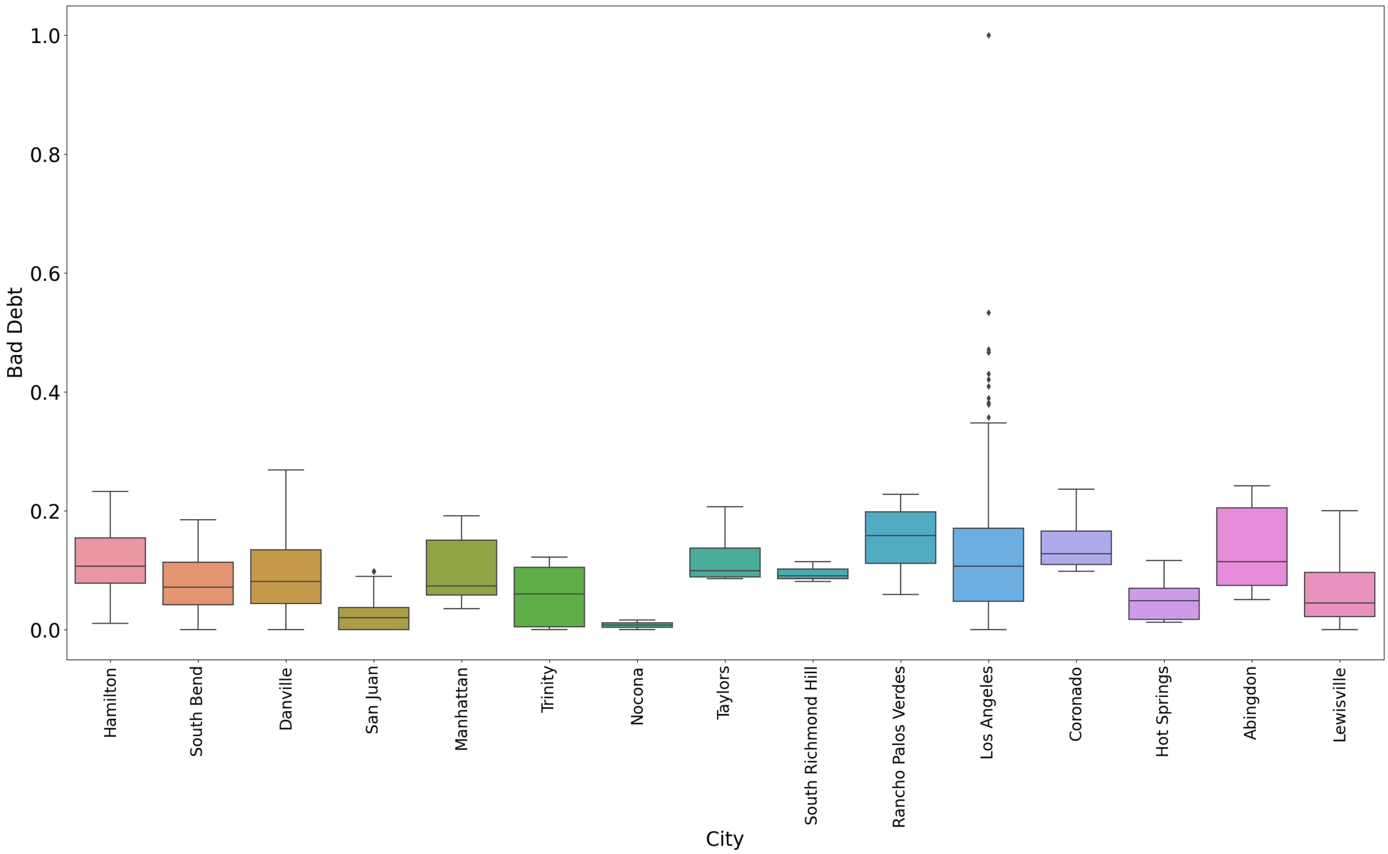
In [38]:
```
#Distribution for Good debt
plt.figure(figsize = (30, 15))
sns.boxplot(x = df['city'], y = df['debt'])
plt.xticks(rotation = 90, fontsize = 20)
plt.yticks(fontsize = 25)
plt.xlabel('City', fontsize = 25)
plt.ylabel('Good Debt', fontsize = 25)
```

Out[38]:
```
Text(0, 0.5, 'Good Debt')
```



In [39]:
```
#Distribution for bad debt
plt.figure(figsize = (30, 15))
sns.boxplot(x = df['city'], y = df['bad_debt'])
plt.xticks(rotation = 90, fontsize = 20)
plt.yticks(fontsize = 25)
plt.xlabel('City', fontsize = 25)
plt.ylabel('Bad Debt', fontsize = 25)
```

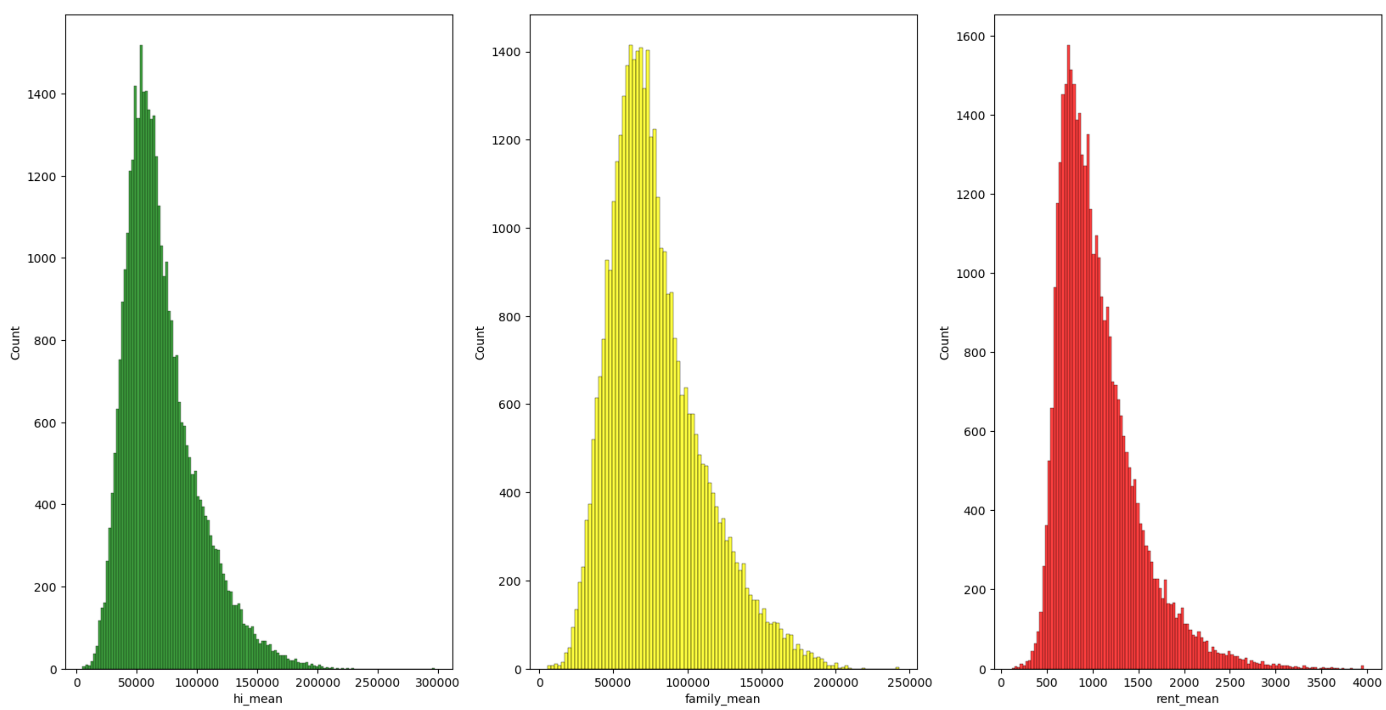
Out[39]:
```
Text(0, 0.5, 'Bad Debt')
```



Create a collated income distribution chart for family income, house hold income, and remaining income

In [40]:
```
f,axs = plt.subplots(1, 3, figsize = (20, 10))
sns.histplot(combined_df['hi_mean'], color='green', ax=axs[0])
sns.histplot(combined_df['family_mean'], color='yellow', ax=axs[1])
sns.histplot(combined_df['rent_mean'], color='red', ax=axs[2])
```

Out[40]:
```
<AxesSubplot:xlabel='rent_mean', ylabel='Count'>
```



Use pop and ALand variables to create a new field called

population density

In [41]:
```
combined_df['population_density'] = combined_df['pop'] / combined_df['ALand']
combined_df[['population_density']].head()
```

Out[41]:

	population_density
0	0.000026
1	0.001687
2	0.000099
3	0.002442
4	0.002207

Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age

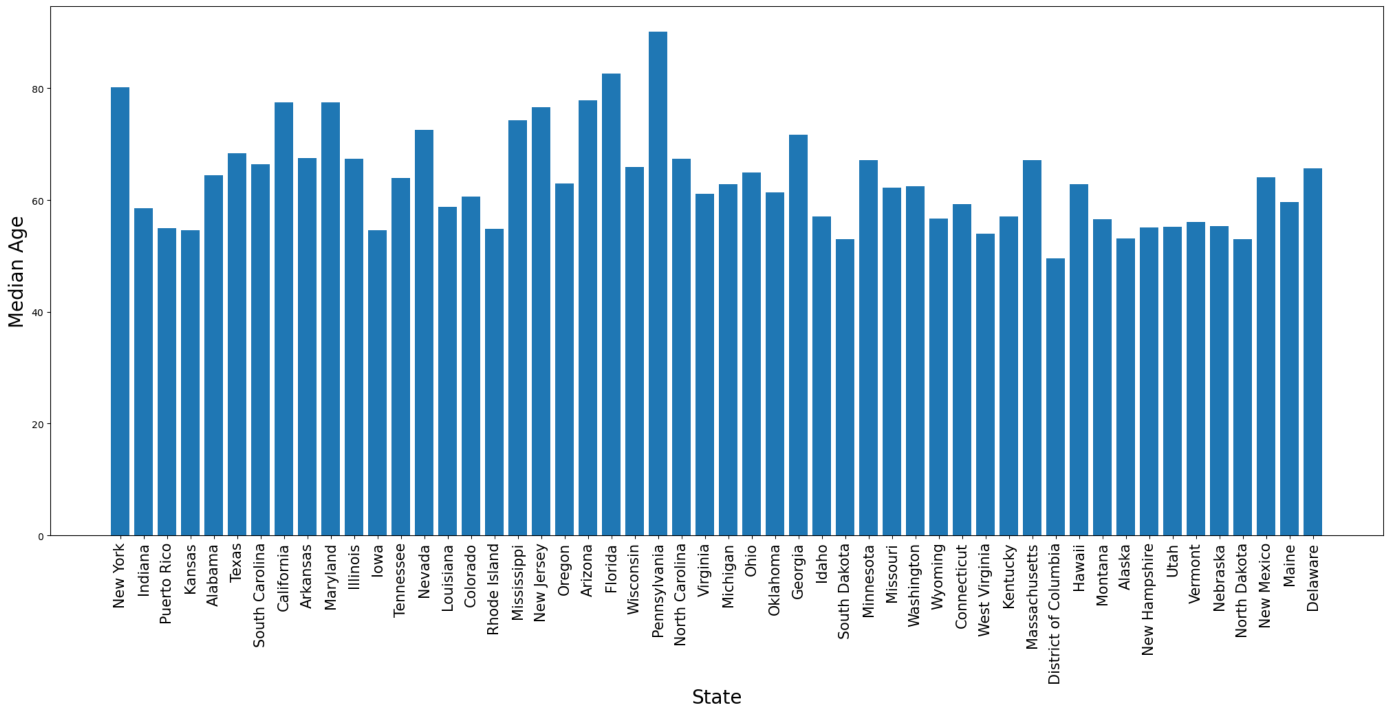
In [42]:
```
combined_df['median_age'] = (((combined_df['male_age_median']*combined_df['male_pop'])+
                               (combined_df['female_age_median']*combined_df['female_pop'
                              (combined_df['male_pop']+combined_df['female_pop']))
combined_df[['median_age']].head()
```

Out[42]:

	median_age
0	44.667430
1	34.722748
2	41.774472
3	49.879012
4	21.965629

Visualize the findings using appropriate chart type

In [43]:
```
plt.figure(figsize = (25, 10))
plt.bar('state', 'median_age', data=combined_df)
plt.xlabel('State', fontsize=20)
plt.ylabel('Median Age', fontsize=20)
plt.xticks(rotation=90, fontsize=15)
plt.show()
```



Create bins for population into a new variable by selecting appropriate class interval so that the number of categories don’t exceed 5 for the ease of analysis

In [44]:
```
#Creating bins for :
# 0-5000 in class 1
# 5000-10000 in class 2
# 10000-15000 in class 3
# 15000-25000 in class 4
# 25000-55000 in class 5

combined_df['pop_class'] = pd.cut(x = combined_df['pop'],
                                  bins = [0,5000,10000,15000,25000,55000],
                                  labels = ['1', '2', '3','4','5'])
```

In [45]:
```
combined_df['pop_class'].value_counts()
```

Out[45]:
```
1    26173
2    11919
3      439
4       82
5        9
Name: pop_class, dtype: int64
```

Analyze the married, separated, and divorced population for these population brackets

In [46]:
```
combined_df = combined_df.drop(combined_df[combined_df['pop']==0].index).reset_index(dro
```

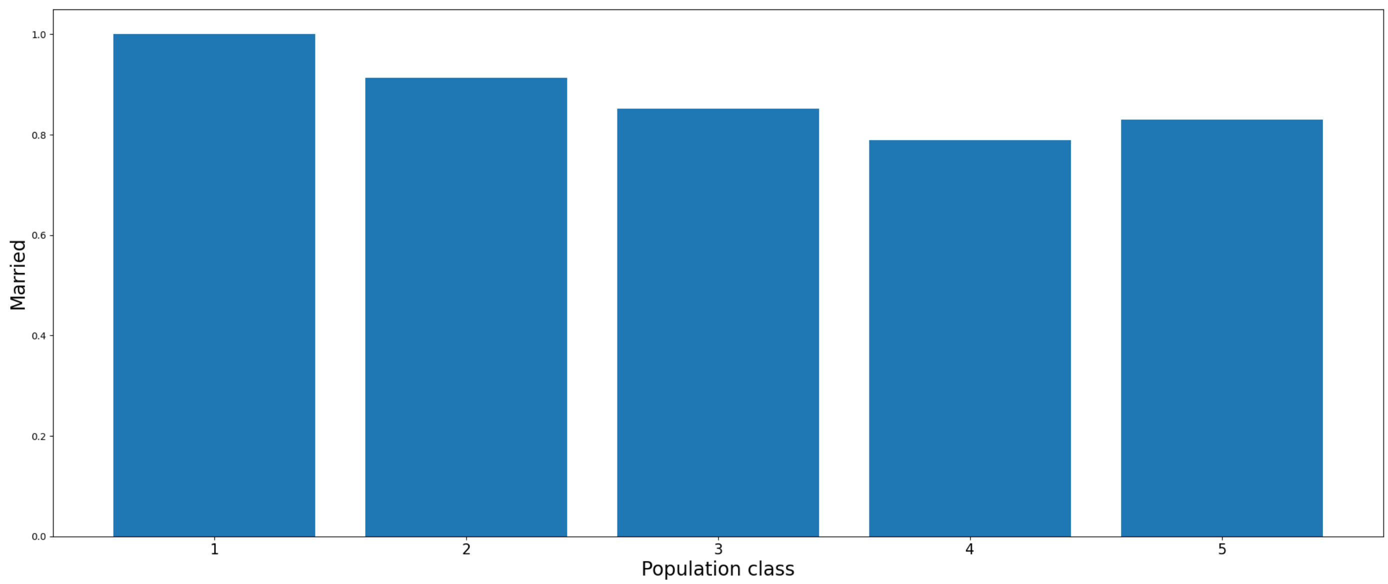
In [47]:
```
combined_df['pop_class']= combined_df['pop_class'].astype('int64')
```

In [48]:
```
for i in [1,2,3,4,5]:
        for j in ['married','separated','divorced']:
            print('Population Class:',i,'|',
                  'Mean:%.3f'%combined_df[combined_df['pop_class']==i][j].mean(),'|',
                  'Status:',j)
```

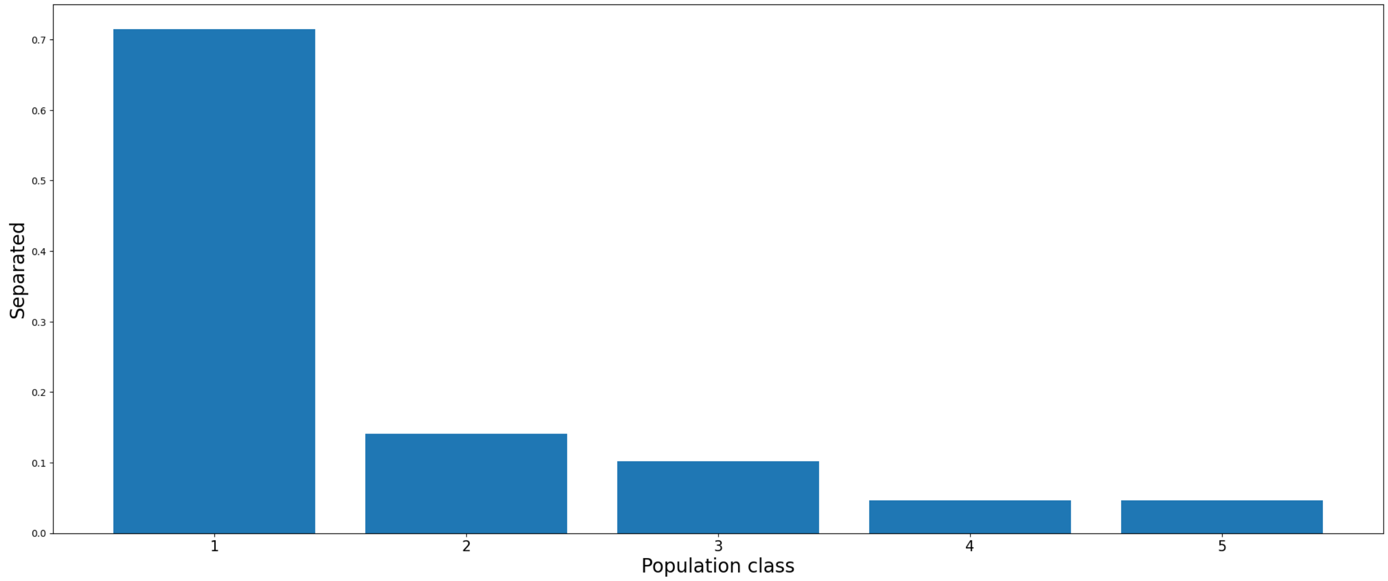
```
Population Class: 1 | Mean:0.496 | Status: married
Population Class: 1 | Mean:0.020 | Status: separated
Population Class: 1 | Mean:0.104 | Status: divorced
Population Class: 2 | Mean:0.531 | Status: married
Population Class: 2 | Mean:0.017 | Status: separated
Population Class: 2 | Mean:0.092 | Status: divorced
Population Class: 3 | Mean:0.575 | Status: married
Population Class: 3 | Mean:0.016 | Status: separated
Population Class: 3 | Mean:0.081 | Status: divorced
Population Class: 4 | Mean:0.606 | Status: married
Population Class: 4 | Mean:0.011 | Status: separated
Population Class: 4 | Mean:0.064 | Status: divorced
Population Class: 5 | Mean:0.588 | Status: married
Population Class: 5 | Mean:0.013 | Status: separated
Population Class: 5 | Mean:0.060 | Status: divorced
```

Visualize using appropriate chart type

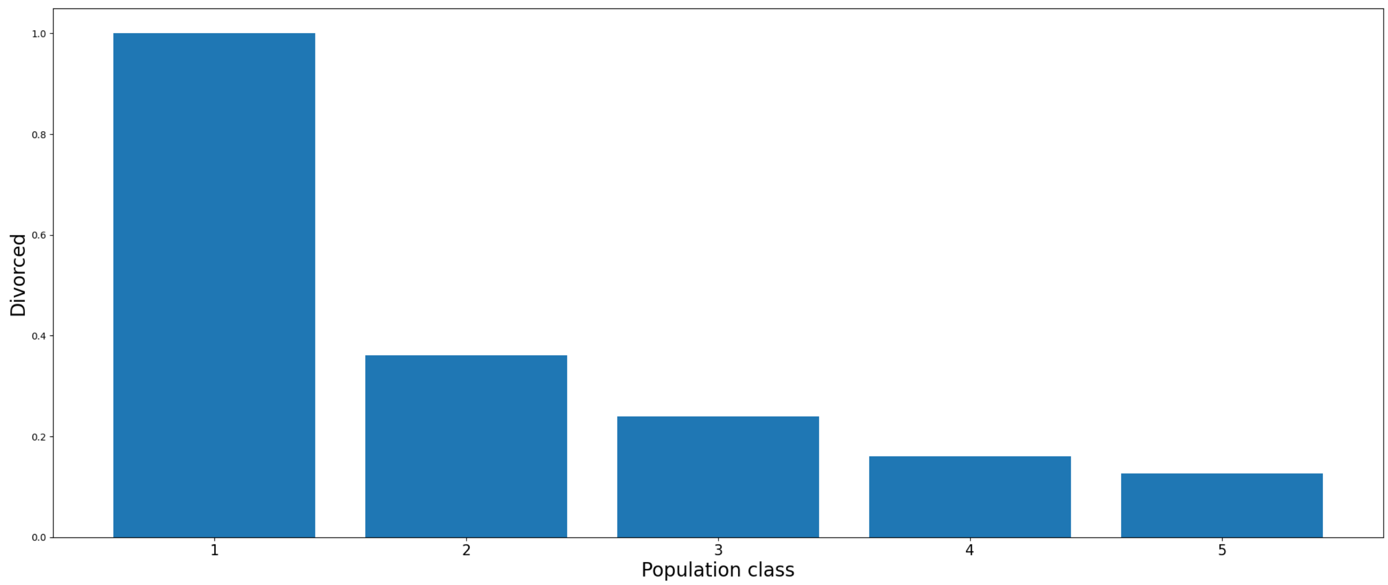
In [49]:
```
plt.figure(figsize = (25, 10))
plt.bar('pop_class', 'married', data=combined_df)
plt.xlabel('Population class', fontsize=20)
plt.ylabel('Married', fontsize=20)
plt.xticks(fontsize=15)
plt.show()
```



In [50]:
```
plt.figure(figsize = (25, 10))
plt.bar('pop_class', 'separated', data=combined_df)
plt.xlabel('Population class', fontsize=20)
plt.ylabel('Separated', fontsize=20)
plt.xticks(fontsize=15)
plt.show()
```



In [51]:
```
plt.figure(figsize = (25, 10))
plt.bar('pop_class', 'divorced', data=combined_df)
plt.xlabel('Population class', fontsize=20)
plt.ylabel('Divorced', fontsize=20)
plt.xticks(fontsize=15)
plt.show()
```



Please detail your observations for rent as a percentage of income at an overall level, and for different states

In [52]:
```
combined_df['%_rent'] = (combined_df['rent_mean']/combined_df['hi_mean'])*100
combined_df[['%_rent']].head()
```

Out[52]:

	%_rent
0	1.218824
1	1.919490
2	0.874441
3	1.648594
4	2.948295

In [53]:
```
states = combined_df['state'].unique().tolist()
states
```

Out[53]:
```
['New York',
 'Indiana',
 'Puerto Rico',
 'Kansas',
 'Alabama',
 'Texas',
 'South Carolina',
 'California',
 'Arkansas',
 'Maryland',
 'Illinois',
 'Iowa',
 'Tennessee',
 'Nevada',
 'Louisiana',
 'Colorado',
 'Rhode Island',
 'Mississippi',
 'New Jersey',
 'Oregon',
 'Arizona',
 'Florida',
 'Wisconsin',
 'Pennsylvania',
 'North Carolina',
 'Virginia',
 'Michigan',
 'Ohio',
 'Oklahoma',
 'Georgia',
 'Idaho',
 'South Dakota',
 'Minnesota',
 'Missouri',
 'Washington',
 'Wyoming',
 'Connecticut',
 'West Virginia',
 'Kentucky',
 'Massachusetts',
 'District of Columbia',
 'Hawaii',
 'Montana',
 'Alaska',
 'New Hampshire',
 'Utah',
 'Vermont',
 'Nebraska',
 'North Dakota',
 'New Mexico',
 'Maine',
 'Delaware']
```

In [54]:
```
for i in states:
            print(i,'=','%.3f'%combined_df[combined_df['state']==i]['%_rent'].mean(),'%'
            print("------------------------")
```

```
New York = 1.705 %
------------------------
Indiana = 1.469 %
------------------------
Puerto Rico = 1.958 %
------------------------
```

```
Kansas = 1.368 %
------------------------
Alabama = 1.468 %
------------------------
Texas = 1.526 %
------------------------
South Carolina = 1.532 %
------------------------
California = 1.897 %
------------------------
Arkansas = 1.384 %
------------------------
Maryland = 1.599 %
------------------------
Illinois = 1.542 %
------------------------
Iowa = 1.209 %
------------------------
Tennessee = 1.489 %
------------------------
Nevada = 1.764 %
------------------------
Louisiana = 1.555 %
------------------------
Colorado = 1.591 %
------------------------
Rhode Island = 1.474 %
------------------------
Mississippi = 1.527 %
------------------------
New Jersey = 1.644 %
------------------------
Oregon = 1.594 %
------------------------
Arizona = 1.725 %
------------------------
Florida = 1.919 %
------------------------
Wisconsin = 1.415 %
------------------------
Pennsylvania = 1.471 %
------------------------
North Carolina = 1.481 %
------------------------
Virginia = 1.593 %
------------------------
Michigan = 1.584 %
------------------------
Ohio = 1.451 %
------------------------
Oklahoma = 1.421 %
------------------------
Georgia = 1.599 %
------------------------
Idaho = 1.388 %
------------------------
South Dakota = 1.112 %
------------------------
Minnesota = 1.323 %
------------------------
Missouri = 1.439 %
------------------------
Washington = 1.553 %
------------------------
Wyoming = 1.260 %
------------------------
```

```
Connecticut = 1.607 %
------------------------
West Virginia = 1.300 %
------------------------
Kentucky = 1.372 %
------------------------
Massachusetts = 1.495 %
------------------------
District of Columbia = 1.703 %
------------------------
Hawaii = 2.035 %
------------------------
Montana = 1.280 %
------------------------
Alaska = 1.450 %
------------------------
New Hampshire = 1.397 %
------------------------
Utah = 1.480 %
------------------------
Vermont = 1.438 %
------------------------
Nebraska = 1.293 %
------------------------
North Dakota = 1.105 %
------------------------
New Mexico = 1.497 %
------------------------
Maine = 1.390 %
------------------------
Delaware = 1.562 %
------------------------
```

Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings

In [55]:

```
var = combined_df.iloc[:,12:77]
var.head()
```

Out[55]:

	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample_weight
0	202183361.0	1699120	5230	2612	2618	769.38638	784.0	232.63967	272.34441
1	1560828.0	100363	2633	1349	1284	804.87924	848.0	253.46747	312.58622
2	69561595.0	284193	6881	3643	3238	742.77365	703.0	323.39011	291.85520
3	1105793.0	0	2700	1141	1559	803.42018	782.0	297.39258	259.30316
4	2554403.0	0	5637	2586	3051	938.56493	881.0	392.44096	1005.42886

5 rows × 65 columns

In [56]:

```
var.corr()
```

Out[56]:

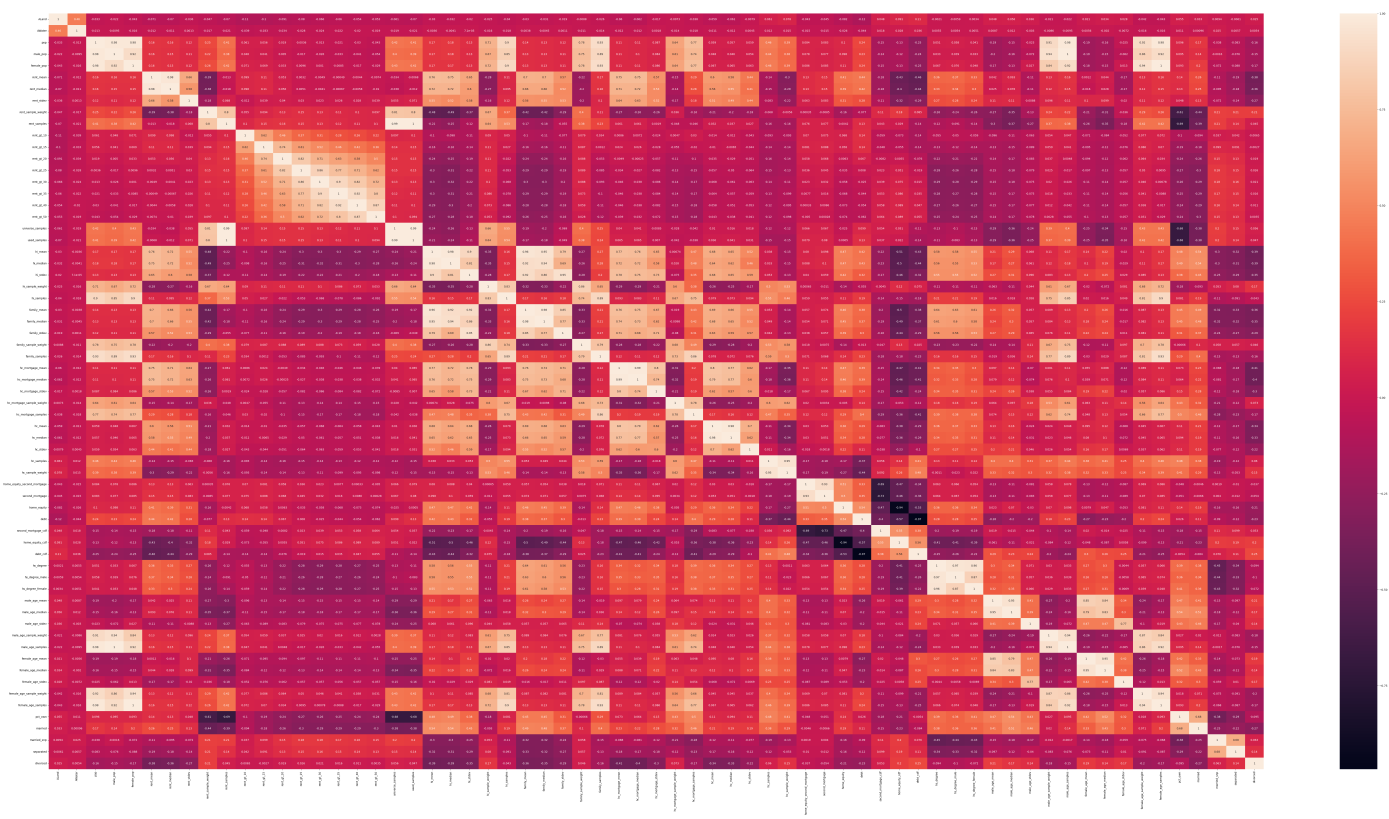
	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sa
ALand	1.000000	0.455449	-0.032923	-0.021729	-0.042678	-0.071482	-0.069624	-0.035939	
AWater	0.455449	1.000000	-0.013074	-0.009509	-0.016076	-0.011709	-0.011278	0.001320	
pop	-0.032923	-0.013074	1.000000	0.979398	0.979774	0.163460	0.157320	0.120056	
male_pop	-0.021729	-0.009509	0.979398	1.000000	0.919180	0.159282	0.153800	0.110374	

female_pop	-0.042678	-0.016076	0.979774	0.919180	1.000000	0.160958	0.154415	0.124771
...	...	...	...	...	...	...	...	...
pct_own	0.054511	0.010880	0.096191	0.095353	0.093112	0.135691	0.127671	0.048370
married	0.032989	0.000963	0.174286	0.141549	0.199644	0.258671	0.246623	0.133570
married_snp	0.009389	0.024880	-0.037980	-0.001637	-0.072450	-0.106860	-0.094693	-0.072198
separated	-0.006100	0.005697	-0.083472	-0.075661	-0.087820	-0.191482	-0.177771	-0.138700
divorced	0.024592	0.005445	-0.162148	-0.147472	-0.170101	-0.378392	-0.362280	-0.273227

65 rows × 65 columns

In [57]:
```
plt.figure(figsize = (100,50))
sns.heatmap(var.corr(),annot=True)
```

Out[57]:
```
<AxesSubplot:>
```



The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a number of smaller unobserved common factors or latent variables

In [58]:
```
pip install factor_analyzer
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: factor_analyzer in c:\users\amit singh\appdata\roaming\py
thon\python39\site-packages (0.4.1)
Requirement already satisfied: scikit-learn in d:\users\amit singh\anaconda2\lib\site-pa
ckages (from factor_analyzer) (1.0.2)
Requirement already satisfied: scipy in d:\users\amit singh\anaconda2\lib\site-packages
(from factor_analyzer) (1.9.1)
Requirement already satisfied: pandas in d:\users\amit singh\anaconda2\lib\site-packages
(from factor_analyzer) (1.4.4)
```

```
Requirement already satisfied: numpy in d:\users\amit singh\anaconda2\lib\site-packages
(from factor_analyzer) (1.21.5)
Requirement already satisfied: pre-commit in c:\users\amit singh\appdata\roaming\python
\python39\site-packages (from factor_analyzer) (3.2.1)
Requirement already satisfied: pytz>=2020.1 in d:\users\amit singh\anaconda2\lib\site-pa
ckages (from pandas->factor_analyzer) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in d:\users\amit singh\anaconda2\l
ib\site-packages (from pandas->factor_analyzer) (2.8.2)
Requirement already satisfied: virtualenv>=20.10.0 in c:\users\amit singh\appdata\roamin
g\python\python39\site-packages (from pre-commit->factor_analyzer) (20.21.0)
Requirement already satisfied: pyyaml>=5.1 in d:\users\amit singh\anaconda2\lib\site-pac
kages (from pre-commit->factor_analyzer) (6.0)
Requirement already satisfied: nodeenv>=0.11.1 in c:\users\amit singh\appdata\roaming\py
thon\python39\site-packages (from pre-commit->factor_analyzer) (1.7.0)
Requirement already satisfied: identify>=1.0.0 in c:\users\amit singh\appdata\roaming\py
thon\python39\site-packages (from pre-commit->factor_analyzer) (2.5.22)
Requirement already satisfied: cfgv>=2.0.0 in c:\users\amit singh\appdata\roaming\python
\python39\site-packages (from pre-commit->factor_analyzer) (3.3.1)
Requirement already satisfied: joblib>=0.11 in d:\users\amit singh\anaconda2\lib\site-pa
ckages (from scikit-learn->factor_analyzer) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\users\amit singh\anaconda2\lib
\site-packages (from scikit-learn->factor_analyzer) (2.2.0)
Requirement already satisfied: setuptools in d:\users\amit singh\anaconda2\lib\site-pack
ages (from nodeenv>=0.11.1->pre-commit->factor_analyzer) (63.4.1)
Requirement already satisfied: six>=1.5 in d:\users\amit singh\anaconda2\lib\site-packag
es (from python-dateutil>=2.8.1->pandas->factor_analyzer) (1.16.0)
Requirement already satisfied: platformdirs<4,>=2.4 in d:\users\amit singh\anaconda2\lib
\site-packages (from virtualenv>=20.10.0->pre-commit->factor_analyzer) (2.5.2)
Requirement already satisfied: distlib<1,>=0.3.6 in c:\users\amit singh\appdata\roaming
\python\python39\site-packages (from virtualenv>=20.10.0->pre-commit->factor_analyzer)
(0.3.6)
Requirement already satisfied: filelock<4,>=3.4.1 in d:\users\amit singh\anaconda2\lib\s
ite-packages (from virtualenv>=20.10.0->pre-commit->factor_analyzer) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
```

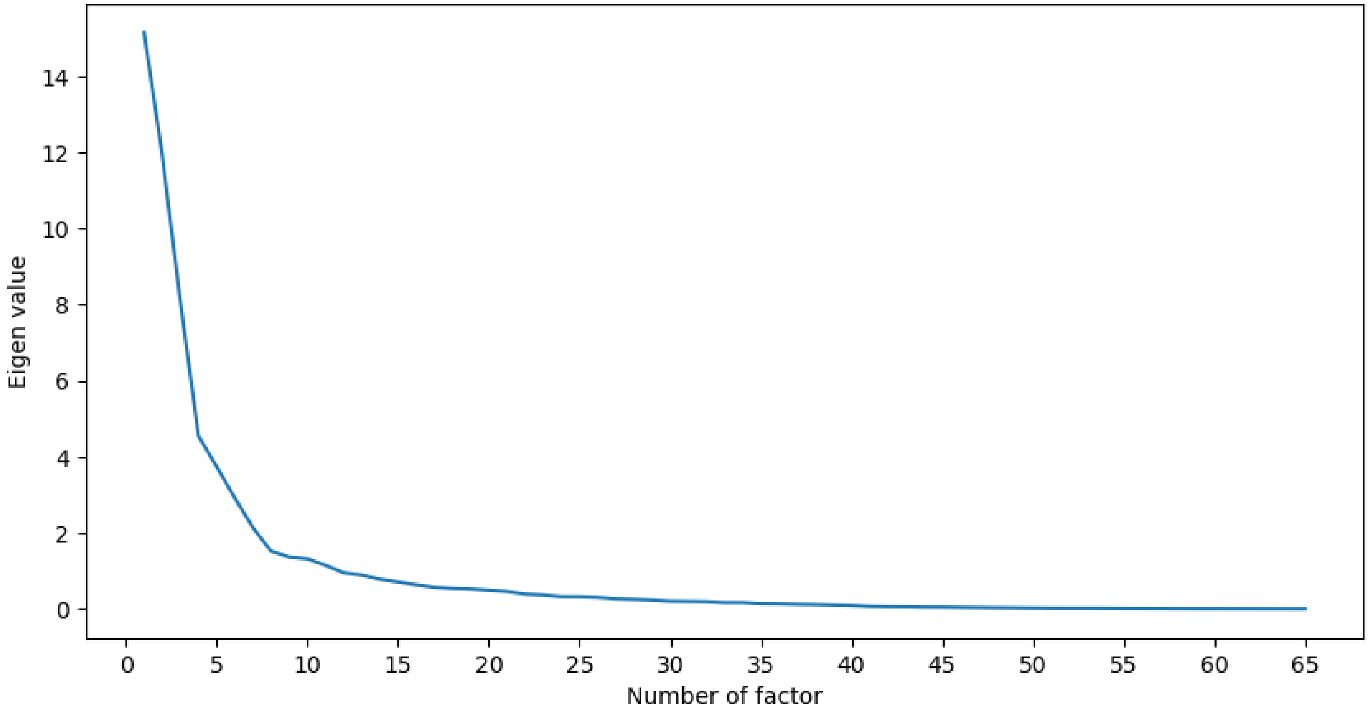
In [59]:
```python
from factor_analyzer.factor_analyzer import FactorAnalyzer
```

In [60]:
```python
fa = FactorAnalyzer()
fa.fit(var, 10)
```

Out[60]:
```
FactorAnalyzer(rotation_kwargs={})
```

In [61]:
```python
ev, v = fa.get_eigenvalues()
```

In [62]:
```python
plt.figure(figsize = (10,5))
plt.plot(range(1, var.shape[1]+1), ev)
plt.xticks(np.arange(0, 70, step=5))
plt.xlabel('Number of factor')
plt.ylabel('Eigen value')
plt.show()
```



In [63]:

```
# Elbow bend can be observed at 8, thus taking n=8
n = 8
fa = FactorAnalyzer(n)
fa.fit(var, 10)
loads = fa.loadings_

print(loads)
```

```
[[-3.33372954e-03 -1.49337266e-02 -4.35546140e-02 -4.18560809e-02
  -1.53763218e-02  4.75300955e-03  1.36201429e-02 -1.82857127e-01]
 [-1.89265860e-03  1.66228269e-02 -8.85199904e-03 -1.25827565e-02
  -3.42481104e-02  9.46073958e-03  1.96987414e-02 -1.16703953e-01]
 [ 9.88776300e-01  1.27974731e-01 -1.26830840e-03  1.03770950e-01
  -9.48633665e-02 -8.98557505e-02  6.54512025e-03  1.49651779e-02]
 [ 9.62452952e-01  1.31358319e-01 -1.51799154e-02  7.86534729e-02
  -1.30059159e-01 -1.03929597e-01  9.93021665e-03 -1.63611983e-02]
 [ 9.66058426e-01  1.16802093e-01  1.27411071e-02  1.26718879e-01
  -5.43058805e-02 -6.75111476e-02  3.02359252e-03  4.71512336e-02]
 [ 6.50140660e-02  7.79322827e-01  1.08530610e-01 -1.09592889e-01
  -7.62487399e-02 -2.95044328e-02 -3.20944886e-02  1.48493373e-01]
 [ 6.59051212e-02  7.32441873e-01  9.77722106e-02 -1.25280537e-01
  -8.77109154e-02 -4.54522410e-02 -3.17312694e-02  1.53613113e-01]
 [ 3.25993625e-02  7.23847931e-01  1.07796565e-01  8.00402647e-02
   6.30168054e-02 -3.41889947e-02 -4.29661820e-03  9.52691902e-03]
 [ 1.69973894e-01 -2.15518906e-01 -9.69612853e-02  8.32090263e-01
   2.25846033e-03  4.32536770e-02 -1.61327225e-03  4.88318519e-02]
 [ 2.46403825e-01  1.38457908e-01 -8.20389470e-02  9.91947301e-01
  -5.54724953e-02  5.59213378e-02 -6.79734491e-03  9.88047655e-02]
 [ 2.89526968e-02 -8.25910302e-02  3.58817990e-01 -1.22916636e-02
   3.34507683e-02  3.70945176e-02 -4.35565208e-02  3.42728330e-01]
 [ 2.23449621e-02 -4.08117264e-02  5.74532396e-01 -7.98522753e-03
   4.53183872e-02  1.04892316e-02 -4.73568050e-02  3.70474254e-01]
 [ 6.81227241e-03 -8.71351012e-03  7.51907026e-01 -1.62214161e-02
   5.57509929e-02 -1.89272976e-02 -2.93069133e-02  2.68671571e-01]
 [ 6.42822629e-03  1.74513093e-02  8.73942278e-01 -3.18217777e-02
   3.49021411e-02 -2.31075815e-02  3.83679212e-03  1.23835447e-01]
 [ 1.82670491e-02  4.29902037e-02  9.54674015e-01 -6.78472192e-02
  -9.36531770e-03  1.40636923e-03  3.70295940e-02 -1.80639308e-02]
 [ 1.89293228e-02  6.65865253e-02  9.78205995e-01 -8.34108999e-02
  -4.28066506e-02  2.72190604e-02  4.99871724e-02 -1.11144999e-01]
 [ 1.29478517e-02  7.42875994e-02  9.51213859e-01 -8.23869476e-02
```

```
 -6.37700793e-02  3.80554680e-02  5.60008445e-02 -1.56337222e-01]
[-4.69653276e-03  7.90229907e-02  8.44909387e-01 -6.10193009e-02
 -7.53893388e-02  3.45745063e-02  4.90455640e-02 -1.67061910e-01]
[ 2.72572028e-01  1.27194265e-01 -7.92651743e-02  9.87021290e-01
 -3.83854927e-02  4.49712249e-02 -7.74374527e-03  7.55947281e-02]
[ 2.51124836e-01  1.40865940e-01 -9.02191319e-02  9.90160385e-01
 -4.17311452e-02  5.01949179e-02 -5.31469527e-03  1.09962403e-01]
[ 9.08989255e-02  7.52970100e-01 -1.51909518e-01 -2.07767720e-01
  5.29263798e-03  1.08925439e-01 -3.01059742e-02  6.15723443e-02]
[ 1.03196925e-01  6.81570874e-01 -1.75305429e-01 -2.62480279e-01
 -2.94672100e-02  9.03748698e-02 -3.38633528e-02  1.00586114e-01]
[ 4.34357983e-02  8.29557189e-01 -4.88022991e-02 -1.84935280e-02
  1.06902313e-01  1.48397040e-01 -1.19900084e-02 -6.79681286e-02]
[ 7.20472345e-01 -2.72940749e-01  4.26149198e-02  5.47888696e-01
  1.91193191e-01  8.18846498e-02  1.88846149e-02  1.46354368e-03]
[ 8.55315502e-01  7.54155973e-02 -3.65767160e-02  4.03740638e-01
  1.45541119e-01  1.56810928e-01  1.39473978e-03  4.95844764e-02]
[ 3.42912204e-02  7.51782188e-01 -1.14583886e-01 -8.52648088e-02
  4.09833928e-02  2.42889816e-01 -3.51755167e-02  6.27584662e-03]
[ 3.77632038e-02  7.23560642e-01 -1.23419211e-01 -1.17491840e-01
  2.03684494e-02  2.17637156e-01 -4.32102325e-02  1.78267330e-02]
[ 1.12297238e-02  7.46123178e-01 -2.41080785e-02  8.12046803e-02
  1.14097370e-01  2.32798090e-01 -3.28537261e-03 -7.17661869e-02]
[ 8.67473924e-01 -2.62022119e-01  2.03170820e-02  1.07682658e-01
  1.00918992e-01 -2.19749796e-01  2.42428427e-02  2.93711923e-02]
[ 9.53714164e-01  8.29290937e-02 -4.25039062e-02 -2.45378671e-02
  8.19649041e-02 -5.82050607e-02  7.26176017e-03  7.64344065e-02]
[-1.78104418e-03  1.03117919e+00  3.05947709e-02  9.20027635e-02
  4.45481992e-02 -1.49450397e-01  3.74690658e-02 -4.67829420e-03]
[-4.71069357e-03  1.01309125e+00  3.00925980e-02  8.13508389e-02
  2.71069277e-02 -1.55731287e-01  2.64095693e-02  7.59804475e-03]
[ 2.96562466e-03  8.46412013e-01  2.83241265e-02  9.94409715e-02
  1.81990184e-01 -5.94459602e-02  7.26055379e-02 -9.47501338e-02]
[ 7.15540567e-01 -4.93021585e-01 -5.48260114e-02 -2.11669238e-01
  7.95595083e-02  2.22342508e-01 -6.40816872e-03  1.61543913e-01]
[ 7.56475113e-01  2.10892752e-02 -3.68790769e-02 -2.42757815e-01
  3.42940301e-02  1.65614817e-01  1.75760114e-02  2.22721450e-01]
[-4.54968374e-02  9.13923126e-01  2.86104370e-02  1.28507643e-01
  5.79429753e-02 -2.19283206e-02 -4.83731384e-02 -3.72930998e-02]
[-4.45744997e-02  8.75026528e-01  2.76543065e-02  1.22288575e-01
  4.17683593e-02 -2.20653297e-02 -5.18133641e-02 -2.78488512e-02]
[-2.68605148e-03  7.89509367e-01  3.94461091e-02  1.39723441e-01
  1.51882598e-01 -5.02844821e-02 -3.05715694e-02 -1.57641216e-01]
[ 6.24749125e-01 -5.41260263e-02  1.04900805e-01 -1.47554375e-01
  3.63177040e-01  1.04372320e-01 -2.57187310e-02 -4.27054466e-01]
[ 5.79522477e-01 -2.70091062e-01  8.24147098e-02 -1.70332470e-01
  3.20491523e-01  7.11559320e-02 -1.69412549e-02 -4.09465215e-01]
[-5.01109815e-04 -9.84423733e-02  4.13888058e-02  3.34087675e-02
 -2.65307944e-02 -7.77049818e-02  1.01701878e+00 -1.65324340e-01]
[-3.77126746e-03 -6.41236128e-02  5.09502760e-02  3.24231528e-02
 -2.21036982e-02 -9.68395652e-02  1.03478564e+00 -1.68933218e-01]
[-3.58702163e-02  2.25392554e-01 -6.81134632e-03  1.47797595e-02
  5.57081798e-02  1.01687618e-01  5.36048603e-01  1.99028148e-01]
[ 7.69828815e-02  6.24273474e-02 -1.36810099e-01 -3.59552484e-02
 -2.30345470e-01  7.90355672e-02  1.22646266e-01  6.87540463e-01]
[-7.66602488e-02  8.47146834e-02 -1.56825233e-02  8.33896343e-02
 -6.19534713e-02 -1.42662837e-02 -7.86633232e-01 -6.36750120e-03]
[ 1.38178587e-02 -2.17422813e-01  1.05730127e-02  2.59782441e-03
 -9.53633670e-02 -1.40132228e-01 -5.09807425e-01 -2.49520933e-01]
[-8.12679917e-02 -8.03789733e-02  1.38997636e-01  5.23121181e-02
  2.73636163e-01 -5.07125709e-02 -1.09688633e-01 -6.97362262e-01]
[-1.13328436e-01  2.60485816e-02  1.59554563e-02  1.69445558e-01
 -1.94632837e-03  1.05370870e+00 -1.78971337e-02  3.47759350e-02]
[-1.00862437e-01  6.78098897e-02  2.75240664e-03  1.71967500e-01
  2.60101013e-03  9.64691155e-01 -1.71902844e-02  4.79618306e-02]
[-1.03059023e-01  1.64778049e-02 -4.18998925e-04  1.28494034e-01
```

```
  1.57142109e-02  9.68656397e-01 -1.06778545e-02  2.56544834e-02]
 [-1.66208469e-01  1.76566478e-01  2.21231096e-02  4.99886730e-02
   8.66084716e-01  1.12584239e-01 -1.14979287e-03 -4.12525712e-02]
 [-1.25268900e-01  1.92386783e-01  2.44252154e-03 -5.62045039e-02
   8.06729334e-01  9.81750741e-02  4.11710560e-03 -2.30827341e-02]
 [ 4.70577411e-02 -1.61716663e-02 -4.24735506e-02 -1.50149205e-01
   6.40229196e-01 -1.73895946e-01  1.61416389e-02  9.60333470e-02]
 [ 8.84002245e-01  1.00430027e-01  6.72245580e-02  8.05102580e-02
  -2.76389353e-01  7.65363657e-03  8.65016377e-04 -1.21021308e-01]
 [ 9.62247552e-01  1.30947077e-01 -1.51823604e-02  7.84773046e-02
  -1.30703015e-01 -1.03176300e-01  9.78468650e-03 -1.66172065e-02]
 [-1.74526624e-01  1.35367538e-01  4.75740872e-02  9.70707703e-02
   9.15556077e-01  9.14707063e-02  7.36522252e-04  1.01722724e-02]
 [-1.29316668e-01  1.42556484e-01  3.57826889e-02 -3.48899223e-02
   8.74235765e-01  7.61786696e-02  9.27318871e-03  1.50088803e-02]
 [ 2.80795536e-02 -4.68537128e-02 -6.28634326e-02 -6.62994838e-02
   5.93676375e-01 -2.03535944e-01  1.50405444e-02  1.02243164e-01]
 [ 9.01111609e-01  8.91652589e-02  9.51798651e-02  1.34517717e-01
  -2.19752826e-01  4.49426996e-02 -1.37227101e-03 -7.12061210e-02]
 [ 9.66661556e-01  1.16450297e-01  1.26986837e-02  1.26111512e-01
  -5.55094337e-02 -6.79387717e-02  2.76121598e-03  4.73375297e-02]
 [ 1.97456246e-01 -2.09971297e-02 -1.50313152e-02 -6.78463453e-01
   3.07269203e-01  1.28219184e-01 -1.79222967e-03  7.22693274e-03]
 [ 2.27744573e-01  1.95229554e-01 -1.71908841e-01 -3.87087374e-01
   3.67671704e-01 -3.82979458e-02 -1.67569338e-02  1.08059534e-01]
 [-2.16846309e-02  1.02122276e-01 -3.52740887e-02  1.21650717e-01
   4.33693863e-02 -5.73045810e-01  1.52619509e-02  4.68593321e-02]
 [-5.42146922e-02 -4.76303470e-02 -2.08657376e-02  1.13553903e-01
   1.01690250e-01 -4.09492812e-01  1.39511201e-02  7.30608649e-02]
 [-1.60595701e-01 -4.35350987e-01 -2.21293390e-02  2.13631202e-01
   2.75849370e-01  1.19023135e-01  3.71738414e-02  4.28412684e-02]]
```

In [64]:
```
df = pd.DataFrame(loads)
df.set_index(var.columns, drop=True, inplace=True)
for i in range(n):
    s = 'Factor ' + str(i+1)
    df.rename(columns = {i : s}, inplace=True)

df
```

Out[64]:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
ALand	-0.003334	-0.014934	-0.043555	-0.041856	-0.015376	0.004753	0.013620	-0.182857
AWater	-0.001893	0.016623	-0.008852	-0.012583	-0.034248	0.009461	0.019699	-0.116704
pop	0.988776	0.127975	-0.001268	0.103771	-0.094863	-0.089856	0.006545	0.014965
male_pop	0.962453	0.131358	-0.015180	0.078653	-0.130059	-0.103930	0.009930	-0.016361
female_pop	0.966058	0.116802	0.012741	0.126719	-0.054306	-0.067511	0.003024	0.047151
...	...	...	...	...	...	...	...	...
pct_own	0.197456	-0.020997	-0.015031	-0.678463	0.307269	0.128219	-0.001792	0.007227
married	0.227745	0.195230	-0.171909	-0.387087	0.367672	-0.038298	-0.016757	0.108060
married_snp	-0.021685	0.102122	-0.035274	0.121651	0.043369	-0.573046	0.015262	0.046859
separated	-0.054215	-0.047630	-0.020866	0.113554	0.101690	-0.409493	0.013951	0.073061
divorced	-0.160596	-0.435351	-0.022129	0.213631	0.275849	0.119023	0.037174	0.042841

65 rows × 8 columns

Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data

- Highschool graduation rates
- Median population age
- Second mortgage statistics
- Percent own
- Bad debt expense

In [65]:
```
latent_variables = combined_df[['pct_own','median_age','second_mortgage','bad_debt','hs_
latent_variables.head()
```

Out[65]:

	pct_own	median_age	second_mortgage	bad_debt	hs_degree
0	0.79046	44.667430	0.02077	0.09408	0.89288
1	0.52483	34.722748	0.02222	0.04274	0.90487
2	0.85331	41.774472	0.00000	0.09512	0.94288
3	0.65037	49.879012	0.01086	0.01086	0.91500
4	0.13046	21.965629	0.05426	0.05426	1.00000

In [66]:
```
latent_variables.corr()
```

Out[66]:

	pct_own	median_age	second_mortgage	bad_debt	hs_degree
pct_own	1.000000	0.548747	-0.050761	0.133315	0.394067
median_age	0.548747	1.000000	-0.116364	0.056075	0.334217
second_mortgage	-0.050761	-0.116364	1.000000	0.559154	0.063609
bad_debt	0.133315	0.056075	0.559154	1.000000	0.350089
hs_degree	0.394067	0.334217	0.063609	0.350089	1.000000

Data Modeling

In [67]:
```
combined_df.head()
```

Out[67]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	...	marrie
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	...	0.5785
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	...	0.3488
2	245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	...	0.6474
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	...	0.4725
4	247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	...	0.1235

5 rows × 83 columns

In [68]:
```
model1 = combined_df.drop(columns=['UID','COUNTYID','STATEID','state_ab','zip_code','are
```

In [69]:
```
# Since we have few features that are categorical, will convert them into integer using

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
model1['state']=le.fit_transform(model1['state'])

le1 = LabelEncoder()
model1['city']=le.fit_transform(model1['city'])

le2 = LabelEncoder()
model1['place']=le2.fit_transform(model1['place'])

le3 = LabelEncoder()
model1['data_type']=le3.fit_transform(model1['data_type'])
```

In [70]:
```
model1_train = model1[model1['data_type']==1]
model1_test = model1[model1['data_type']==0]
```

In [71]:
```
model1_x_train = model1_train.drop(columns=['hc_mortgage_mean']).values
model1_x_train
```

Out[71]:
```
array([[3.20000000e+01, 2.95200000e+03, 4.32000000e+03, ...,
        4.46674298e+01, 2.00000000e+00, 1.21882442e+00],
       [1.40000000e+01, 6.79700000e+03, 9.07100000e+03, ...,
        3.47227481e+01, 1.00000000e+00, 1.91949027e+00],
       [1.40000000e+01, 1.70800000e+03, 2.47000000e+03, ...,
        4.17744723e+01, 2.00000000e+00, 8.74441002e-01],
       ...,
       [5.00000000e+00, 7.74200000e+03, 9.17800000e+03, ...,
        4.40893115e+01, 1.00000000e+00, 1.06319740e+00],
       [4.40000000e+01, 1.42500000e+03, 2.08800000e+03, ...,
        4.50292805e+01, 3.00000000e+00, 1.19174324e+00],
       [2.80000000e+01, 3.87500000e+03, 8.00500000e+03, ...,
        3.11323118e+01, 1.00000000e+00, 1.83906163e+00]])
```

In [72]:
```
model1_x_test = model1_test.drop(columns=['hc_mortgage_mean']).values
model1_x_test
```

Out[72]:
```
array([[2.20000000e+01, 1.81600000e+03, 2.52000000e+03, ...,
        3.11890533e+01, 1.00000000e+00, 1.75578752e+00],
       [1.90000000e+01, 2.69000000e+02, 3.91000000e+02, ...,
        4.63829910e+01, 1.00000000e+00, 1.15114733e+00],
       [3.80000000e+01, 5.67800000e+03, 6.72600000e+03, ...,
        4.31474198e+01, 1.00000000e+00, 1.39485508e+00],
       ...,
       [2.10000000e+01, 3.89700000e+03, 6.64000000e+03, ...,
        3.93236302e+01, 2.00000000e+00, 1.19454582e+00],
       [1.50000000e+01, 1.05000000e+03, 1.57000000e+03, ...,
        4.45285973e+01, 2.00000000e+00, 1.20415796e+00],
       [4.40000000e+01, 2.77000000e+02, 1.03090000e+04, ...,
        3.52071711e+01, 1.00000000e+00, 1.63791229e+00]])
```

In [73]:
```
model1_x_train.shape, model1_x_test.shape
```

Out[73]:
```
((27019, 73), (11603, 73))
```

In [74]:
```
model1_y_train = model1_train['hc_mortgage_mean'].values
```

```
model1_y_train
```

Out[74]:
```
array([1414.80295,  864.4139 , 1506.06758, ..., 1671.07908, 3074.83088,
       1455.4234 ])
```

In [75]:
```
model1_y_test = model1_test['hc_mortgage_mean'].values
model1_y_test
```

Out[75]:
```
array([1139.24548, 1533.25988, 1254.54462, ..., 1791.63902, 1182.30365,
       1364.17379])
```

Linear Regression

In [76]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

First model with all the features

In [77]:
```
model1_lr = LinearRegression()
```

In [78]:
```
model1_lr.fit(model1_x_train,model1_y_train)
```

Out[78]:
```
LinearRegression()
```

In [79]:
```
model1_y_pred = model1_lr.predict(model1_x_test)
```

In [80]:
```
r2_score(model1_y_test, model1_y_pred)
```

Out[80]:
```
0.9874477647134917
```

R square score for model1 is 0.987 i.e very high but R square value increases with the increased number of features and thus we will reduce number of features for model2

In [81]:
```
import math
```

In [82]:
```
RMSE =  math.sqrt(mean_squared_error(model1_y_test, model1_y_pred))
RMSE
```

Out[82]:
```
70.59062563607944
```

Root Mean Square Error for model1 is 70.59

In [83]:
```
combined_df.corr()
```

Out[83]:

	UID	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AWa
UID	1.000000	0.262508	0.977250	-0.224913	0.020760	0.177594	0.289902	-0.015847	-0.028
COUNTYID	0.262508	1.000000	0.224545	0.034504	0.064198	-0.150504	0.071227	0.011864	0.012
STATEID	0.977250	0.224545	1.000000	-0.263080	0.041162	0.106460	0.320501	-0.015467	-0.026
zip_code	-0.224913	0.034504	-0.263080	1.000000	-0.006866	-0.064274	-0.927673	0.072676	0.031
area_code	0.020760	0.064198	0.041162	-0.006866	1.000000	-0.123012	-0.012082	0.015327	0.021
...	...	...	...	...	...	...	...	...	
bad_debt	-0.129954	-0.125309	-0.150047	-0.061916	0.001796	0.213900	-0.012860	-0.082722	-0.025
population_density	-0.014908	-0.080217	-0.011986	-0.118788	-0.028577	0.052490	0.066848	-0.047295	-0.013

median_age	-0.018141	-0.060917	-0.014016	-0.130128	-0.013401	0.013277	0.109913	0.046250	0.005
pop_class	-0.016743	0.003548	-0.022874	0.062080	0.030310	-0.074146	-0.070071	-0.020830	-0.007
%_rent	-0.144161	-0.015744	-0.129989	0.043692	0.018369	-0.179452	-0.077973	-0.062773	-0.013

77 rows × 77 columns

Second Model with fewer features (Removing features which were insignificant based on correlation values)

In [84]:

```
model2 = model1[['state','city','place','ALand','pop','rent_mean','rent_stdev','hi_mean'
                 'hi_sample_weight','hc_stdev','second_mortgage',
                 'debt','debt_cdf','hs_degree','median_age','pct_own','median_age',
                  'home_equity','data_type','hc_mortgage_mean']]
model2.head()
```

Out[84]:

	state	city	place	ALand	pop	rent_mean	rent_stdev	hi_mean	hi_sample_weight	hc_stdev	secor
0	32	2952	4320	202183361.0	5230	769.38638	232.63967	63125.28406	1290.96240	270.11299	
1	14	6797	9071	1560828.0	2633	804.87924	253.46747	41931.92593	838.74664	125.40457	
2	14	1708	2470	69561595.0	6881	742.77365	323.39011	84942.68317	1155.20980	184.42175	
3	39	6402	4251	1105793.0	2700	803.42018	297.39258	48733.67116	928.32193	185.55887	
4	16	4255	6274	2554403.0	5637	938.56493	392.44096	31834.15466	1548.67477	76.12674	

In [85]:

```
model2.corr()
```

Out[85]:

	state	city	place	ALand	pop	rent_mean	rent_stdev	hi_mean	hi_sar
state	1.000000	-0.028575	-0.000981	-0.012449	-0.024165	-0.186634	-0.141012	-0.050857	
city	-0.028575	1.000000	0.524523	0.000731	0.014808	0.023054	0.023935	0.017941	
place	-0.000981	0.524523	1.000000	-0.001237	0.010081	0.027506	0.027797	0.014207	
ALand	-0.012449	0.000731	-0.001237	1.000000	-0.032923	-0.071482	-0.035939	-0.030496	
pop	-0.024165	0.014808	0.010081	-0.032923	1.000000	0.163460	0.120056	0.173600	
rent_mean	-0.186634	0.023054	0.027506	-0.071482	0.163460	1.000000	0.662229	0.755061	
rent_stdev	-0.141012	0.023935	0.027797	-0.035939	0.120056	0.662229	1.000000	0.552092	
hi_mean	-0.050857	0.017941	0.014207	-0.030496	0.173600	0.755061	0.552092	1.000000	
hi_sample_weight	0.028375	0.005275	0.007095	-0.025387	0.710603	-0.277523	-0.163111	-0.348184	
hc_stdev	-0.035079	-0.003947	0.020390	-0.007894	0.059366	0.444302	0.439052	0.519272	
second_mortgage	-0.105186	0.009321	0.004195	-0.044954	0.082808	0.150986	0.083304	0.098118	
debt	-0.131503	0.010578	0.016683	-0.122598	0.241463	0.436349	0.276172	0.419093	
debt_cdf	0.136978	-0.009674	-0.016895	0.111788	-0.249127	-0.458502	-0.286795	-0.425671	
hs_degree	0.039840	0.018611	-0.004280	-0.002124	0.051071	0.362281	0.269386	0.582384	
median_age	-0.018062	0.028612	0.003759	0.046250	-0.160976	0.070182	0.112328	0.262784	
pct_own	0.066441	0.037039	-0.016203	0.054511	0.096191	0.135691	0.048370	0.481934	
median_age	-0.018062	0.028612	0.003759	0.046250	-0.160976	0.070182	0.112328	0.262784	

home_equity	-0.123769	0.030183	0.016898	-0.082397	0.104680	0.411801	0.308239	0.473898
data_type	-0.004220	-0.000051	-0.008253	0.008880	-0.011458	0.001726	-0.000182	0.004762
hc_mortgage_mean	-0.138523	0.001830	0.030580	-0.059813	0.113466	0.751835	0.638806	0.767424

In [86]:
```
model2_train = model2[model2['data_type']==1]
model2_test = model2[model2['data_type']==0]
```

In [87]:
```
model2_x_train = model2_train.drop(columns=['hc_mortgage_mean']).values
model2_x_train
```

Out[87]:
```
array([[3.20000000e+01, 2.95200000e+03, 4.32000000e+03, ...,
        4.46674298e+01, 8.91900000e-02, 1.00000000e+00],
       [1.40000000e+01, 6.79700000e+03, 9.07100000e+03, ...,
        3.47227481e+01, 4.27400000e-02, 1.00000000e+00],
       [1.40000000e+01, 1.70800000e+03, 2.47000000e+03, ...,
        4.17744723e+01, 9.51200000e-02, 1.00000000e+00],
       ...,
       [5.00000000e+00, 7.74200000e+03, 9.17800000e+03, ...,
        4.40893115e+01, 7.85700000e-02, 1.00000000e+00],
       [4.40000000e+01, 1.42500000e+03, 2.08800000e+03, ...,
        4.50292805e+01, 1.25560000e-01, 1.00000000e+00],
       [2.80000000e+01, 3.87500000e+03, 8.00500000e+03, ...,
        3.11323118e+01, 1.83620000e-01, 1.00000000e+00]])
```

In [88]:
```
model2_x_test = model2_test.drop(columns=['hc_mortgage_mean']).values
model2_x_test
```

Out[88]:
```
array([[2.20000000e+01, 1.81600000e+03, 2.52000000e+03, ...,
        3.11890533e+01, 7.65100000e-02, 0.00000000e+00],
       [1.90000000e+01, 2.69000000e+02, 3.91000000e+02, ...,
        4.63829910e+01, 1.43750000e-01, 0.00000000e+00],
       [3.80000000e+01, 5.67800000e+03, 6.72600000e+03, ...,
        4.31474198e+01, 6.49700000e-02, 0.00000000e+00],
       ...,
       [2.10000000e+01, 3.89700000e+03, 6.64000000e+03, ...,
        3.93236302e+01, 1.35450000e-01, 0.00000000e+00],
       [1.50000000e+01, 1.05000000e+03, 1.57000000e+03, ...,
        4.45285973e+01, 7.96700000e-02, 0.00000000e+00],
       [4.40000000e+01, 2.77000000e+02, 1.03090000e+04, ...,
        3.52071711e+01, 5.04200000e-02, 0.00000000e+00]])
```

In [89]:
```
model2_y_train = model2_train['hc_mortgage_mean'].values
model2_y_train
```

Out[89]:
```
array([1414.80295,  864.4139 , 1506.06758, ..., 1671.07908, 3074.83088,
       1455.4234 ])
```

In [90]:
```
model2_y_test = model2_test['hc_mortgage_mean'].values
model2_y_test
```

Out[90]:
```
array([1139.24548, 1533.25988, 1254.54462, ..., 1791.63902, 1182.30365,
       1364.17379])
```

In [91]:
```
model2_lr = LinearRegression()
```

In [92]:
```
model2_lr.fit(model2_x_train,model2_y_train)
```

Out[92]:
```
LinearRegression()
```

In [93]:
```
model2_y_pred = model2_lr.predict(model2_x_test)
```

In [94]:
```
r2_score(model2_y_test, model2_y_pred)
```

Out[94]:
```
0.7887420001171161
```

R square value of model2 is 0.79

Since R Square value for both the models (i.e model1 = 98% & model2= 79%) is high, the model is satisfactory at Nation level.

In [ ]:
```

```