In [1]:
```python
import time
import random
import math
import operator
import pandas as pd
import numpy as np

#import plotting libraries
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import matplotlib
%matplotlib inline

import seaborn as sns
sns.set(style='white',color_codes=True)
sns.set(font_scale=1.5)
```

1.Import Data

In [2]:
```python
df_test=pd.read_csv('real_estate_test.csv')
df_test.head()
```

Out[2]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...	34.78682	33.75000	21.58531	416.48097	1938.0	0.70252	0.28217	0.05910	0.03813	0.14299
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...	44.23451	46.66667	22.37036	532.03505	1950.0	0.85128	0.64221	0.02338	0.00000	0.13377
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...	41.62426	44.50000	22.86213	453.11959	1879.0	0.81897	0.59961	0.01746	0.01358	0.10026
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...	44.81200	48.00000	21.03155	263.94320	1081.0	0.84609	0.56953	0.05492	0.04694	0.12489
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...	40.66618	42.66667	21.30900	709.90829	2956.0	0.79077	0.57620	0.01726	0.00588	0.16379

5 rows × 80 columns

In [3]:
```python
df_train=pd.read_csv('real_estate_train.csv')
df_train.head()
```

Out[3]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.57851	0.01882	0.01240	0.08770
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.58333	23.43353	267.23367	1284.0	0.52483	0.34886	0.01426	0.01426	0.09030
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.83333	23.94119	707.01963	3238.0	0.85331	0.64745	0.02830	0.01607	0.10657
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.58333	24.32015	362.20193	1559.0	0.65037	0.47257	0.02021	0.02021	0.10106
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.58333	11.10484	1854.48652	3051.0	0.13046	0.12356	0.00000	0.00000	0.03109

5 rows × 80 columns

In [4]:
```python
df_test.columns
```

Out[4]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state',
       'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_code',
       'lat', 'lng', 'ALand', 'AWater', 'pop', 'male_pop', 'female_pop',
       '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',
       'universe_samples', 'used_samples', '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'],
      dtype='object')
```

In [5]:
```python
df_train.columns
```

Out[5]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state',
       'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_code',
       'lat', 'lng', 'ALand', 'AWater', 'pop', 'male_pop', 'female_pop',
       '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',
       'universe_samples', 'used_samples', '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'],
      dtype='object')
```

In [6]:
```
df_test.shape
```

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

In [7]:
```
df_train.shape
```

Out[7]:
```
(27321, 80)
```

In [8]:
```
df_test.describe()
```

Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	1.170900e+04	...	11613.000000	11613.000000	11613.000000	11613.000000	11613.000000	11587.000000	11625.000000	11625.000000	11625.000000	11625.000000
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	593.598514	37.405491	-91.340229	1.095500e+08	...	40.111999	40.131864	22.148145	550.411243	2233.003186	0.634194	0.505632	0.047960	0.019346	0.099191
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	232.074263	5.625904	16.407818	7.624940e+08	...	5.851192	7.972026	2.554907	280.992521	1072.017063	0.232232	0.139774	0.038693	0.021428	0.048525
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	201.000000	17.965835	-166.770979	8.299000e+03	...	15.360240	12.833330	0.737110	0.251910	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	404.000000	33.919813	-97.816561	1.718660e+06	...	36.729210	34.750000	21.270920	363.225840	1499.000000	0.492500	0.422020	0.020890	0.004500	0.064590
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	612.000000	38.618093	-86.643344	4.835000e+06	...	40.196960	40.333330	22.472990	509.103610	2099.000000	0.687640	0.525270	0.038680	0.013870	0.094350
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	787.000000	41.232973	-79.697311	3.204540e+07	...	43.496490	45.333330	23.549450	685.883910	2800.000000	0.815235	0.605660	0.065340	0.027910	0.128400
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	989.000000	64.804269	-65.695344	5.520166e+10	...	90.107940	90.166670	29.626680	4145.557870	15466.000000	1.000000	1.000000	0.714290	0.714290	0.362750

8 rows × 74 columns

In [9]:
```
df_train.describe()
```

Out[9]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	2.732100e+04	...	27115.000000	27115.000000	27115.000000	27115.000000	27115.000000	27053.000000	27130.000000	27130.000000	27130.000000	27130.000000
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	596.507668	37.508813	-91.288394	1.295106e+08	...	40.319803	40.355099	22.178745	544.238432	2208.761903	0.640434	0.508300	0.047537	0.019089	0.100248
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	232.497482	5.588268	16.343816	1.275531e+09	...	5.886317	8.039585	2.540257	283.546896	1089.316999	0.226640	0.136860	0.037640	0.020796	0.049055
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	201.000000	17.929085	-165.453872	4.113400e+04	...	16.008330	13.250000	0.556780	0.664700	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	405.000000	33.899064	-97.816067	1.799408e+06	...	36.892050	34.916670	21.312135	355.995825	1471.000000	0.502780	0.425102	0.020810	0.004530	0.065800
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	614.000000	38.755183	-86.554374	4.866940e+06	...	40.373320	40.583330	22.514410	503.643890	2066.000000	0.690840	0.526665	0.038840	0.013460	0.095205
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	801.000000	41.380606	-79.782503	3.359820e+07	...	43.567120	45.416670	23.575260	680.275055	2772.000000	0.817460	0.605760	0.065100	0.027488	0.129000
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	989.000000	67.074017	-65.379332	1.039510e+11	...	79.837390	82.250000	30.241270	6197.995200	27250.000000	1.000000	1.000000	0.714290	0.714290	1.000000

8 rows × 74 columns

In [10]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11709 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   UID                          11709 non-null  int64  
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     11709 non-null  int64  
 3   COUNTYID                     11709 non-null  int64  
 4   STATEID                      11709 non-null  int64  
 5   state                        11709 non-null  object 
 6   state_ab                     11709 non-null  object 
 7   city                         11709 non-null  object 
 8   place                        11709 non-null  object 
 9   type                         11709 non-null  object 
 10  primary                      11709 non-null  object 
 11  zip_code                     11709 non-null  int64  
 12  area_code                    11709 non-null  int64  
 13  lat                          11709 non-null  float64
 14  lng                          11709 non-null  float64
 15  ALand                        11709 non-null  int64  
 16  AWater                       11709 non-null  int64  
 17  pop                          11709 non-null  int64  
 18  male_pop                     11709 non-null  int64  
 19  female_pop                   11709 non-null  int64  
 20  rent_mean                    11561 non-null  float64
 21  rent_median                  11561 non-null  float64
 22  rent_stdev                   11561 non-null  float64
 23  rent_sample_weight           11561 non-null  float64
 24  rent_samples                 11561 non-null  float64
 25  rent_gt_10                   11560 non-null  float64
 26  rent_gt_15                   11560 non-null  float64
 27  rent_gt_20                   11560 non-null  float64
 28  rent_gt_25                   11560 non-null  float64
 29  rent_gt_30                   11560 non-null  float64
 30  rent_gt_35                   11560 non-null  float64
 31  rent_gt_40                   11560 non-null  float64
 32  rent_gt_50                   11560 non-null  float64
 33  universe_samples             11709 non-null  int64  
 34  used_samples                 11709 non-null  int64  
 35  hi_mean                      11587 non-null  float64
 36  hi_median                    11587 non-null  float64
 37  hi_stdev                     11587 non-null  float64
 38  hi_sample_weight             11587 non-null  float64
 39  hi_samples                   11587 non-null  float64
 40  family_mean                  11573 non-null  float64
 41  family_median                11573 non-null  float64
 42  family_stdev                 11573 non-null  float64
 43  family_sample_weight         11573 non-null  float64
 44  family_samples               11573 non-null  float64
 45  hc_mortgage_mean             11441 non-null  float64
 46  hc_mortgage_median           11441 non-null  float64
 47  hc_mortgage_stdev            11441 non-null  float64
 48  hc_mortgage_sample_weight    11441 non-null  float64
 49  hc_mortgage_samples          11441 non-null  float64
 50  hc_mean                      11419 non-null  float64
 51  hc_median                    11419 non-null  float64
 52  hc_stdev                     11419 non-null  float64
 53  hc_samples                   11419 non-null  float64
 54  hc_sample_weight             11419 non-null  float64
 55  home_equity_second_mortgage  11489 non-null  float64
 56  second_mortgage              11489 non-null  float64
 57  home_equity                  11489 non-null  float64
 58  debt                         11489 non-null  float64
 59  second_mortgage_cdf          11489 non-null  float64
 60  home_equity_cdf              11489 non-null  float64
 61  debt_cdf                     11489 non-null  float64
 62  hs_degree                    11624 non-null  float64
 63  hs_degree_male               11620 non-null  float64
 64  hs_degree_female             11604 non-null  float64
 65  male_age_mean                11625 non-null  float64
 66  male_age_median              11625 non-null  float64
 67  male_age_stdev               11625 non-null  float64
 68  male_age_sample_weight       11625 non-null  float64
 69  male_age_samples             11625 non-null  float64
 70  female_age_mean              11613 non-null  float64
 71  female_age_median            11613 non-null  float64
 72  female_age_stdev             11613 non-null  float64
 73  female_age_sample_weight     11613 non-null  float64
 74  female_age_samples           11613 non-null  float64
 75  pct_own                      11587 non-null  float64
 76  married                      11625 non-null  float64
 77  married_snp                  11625 non-null  float64
 78  separated                    11625 non-null  float64
 79  divorced                     11625 non-null  float64
dtypes: float64(61), int64(13), object(6)
memory usage: 7.1+ MB
```

In [11]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27321 entries, 0 to 27320
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   UID                          27321 non-null  int64  
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     27321 non-null  int64  
 3   COUNTYID                     27321 non-null  int64  
 4   STATEID                      27321 non-null  int64  
 5   state                        27321 non-null  object 
 6   state_ab                     27321 non-null  object 
 7   city                         27321 non-null  object 
 8   place                        27321 non-null  object 
 9   type                         27321 non-null  object 
 10  primary                      27321 non-null  object 
 11  zip_code                     27321 non-null  int64  
 12  area_code                    27321 non-null  int64  
 13  lat                          27321 non-null  float64
 14  lng                          27321 non-null  float64
 15  ALand                        27321 non-null  float64
 16  AWater                       27321 non-null  int64  
 17  pop                          27321 non-null  int64  
 18  male_pop                     27321 non-null  int64  
 19  female_pop                   27321 non-null  int64  
 20  rent_mean                    27007 non-null  float64
 21  rent_median                  27007 non-null  float64
 22  rent_stdev                   27007 non-null  float64
 23  rent_sample_weight           27007 non-null  float64
 24  rent_samples                 27007 non-null  float64
 25  rent_gt_10                   27007 non-null  float64
 26  rent_gt_15                   27007 non-null  float64
 27  rent_gt_20                   27007 non-null  float64
 28  rent_gt_25                   27007 non-null  float64
 29  rent_gt_30                   27007 non-null  float64
 30  rent_gt_35                   27007 non-null  float64
 31  rent_gt_40                   27007 non-null  float64
 32  rent_gt_50                   27007 non-null  float64
 33  universe_samples             27321 non-null  int64  
 34  used_samples                 27321 non-null  int64  
 35  hi_mean                      27053 non-null  float64
 36  hi_median                    27053 non-null  float64
 37  hi_stdev                     27053 non-null  float64
 38  hi_sample_weight             27053 non-null  float64
 39  hi_samples                   27053 non-null  float64
 40  family_mean                  27023 non-null  float64
 41  family_median                27023 non-null  float64
 42  family_stdev                 27023 non-null  float64
 43  family_sample_weight         27023 non-null  float64
 44  family_samples               27023 non-null  float64
 45  hc_mortgage_mean             26748 non-null  float64
 46  hc_mortgage_median           26748 non-null  float64
 47  hc_mortgage_stdev            26748 non-null  float64
 48  hc_mortgage_sample_weight    26748 non-null  float64
 49  hc_mortgage_samples          26748 non-null  float64
 50  hc_mean                      26721 non-null  float64
 51  hc_median                    26721 non-null  float64
 52  hc_stdev                     26721 non-null  float64
 53  hc_samples                   26721 non-null  float64
 54  hc_sample_weight             26721 non-null  float64
 55  home_equity_second_mortgage  26864 non-null  float64
 56  second_mortgage              26864 non-null  float64
 57  home_equity                  26864 non-null  float64
 58  debt                         26864 non-null  float64
 59  second_mortgage_cdf          26864 non-null  float64
 60  home_equity_cdf              26864 non-null  float64
 61  debt_cdf                     26864 non-null  float64
 62  hs_degree                    27131 non-null  float64
 63  hs_degree_male               27121 non-null  float64
 64  hs_degree_female             27098 non-null  float64
 65  male_age_mean                27132 non-null  float64
 66  male_age_median              27132 non-null  float64
 67  male_age_stdev               27132 non-null  float64
 68  male_age_sample_weight       27132 non-null  float64
 69  male_age_samples             27132 non-null  float64
 70  female_age_mean              27115 non-null  float64
 71  female_age_median            27115 non-null  float64
 72  female_age_stdev             27115 non-null  float64
 73  female_age_sample_weight     27115 non-null  float64
 74  female_age_samples           27115 non-null  float64
 75  pct_own                      27053 non-null  float64
 76  married                      27130 non-null  float64
 77  married_snp                  27130 non-null  float64
 78  separated                    27130 non-null  float64
 79  divorced                     27130 non-null  float64
dtypes: float64(62), int64(12), object(6)
memory usage: 16.7+ MB
```

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

In [12]:
```
#UID is unique userID value in the train and test dataset.So an index can be created from the UID feature
df_train.set_index(keys=['UID'],inplace=True) #set the DataFrame index using existing columns
df_test.set_index(keys=['UID'],inplace=True)
```

In [13]:
```
df_train.head(2)
```

Out[13]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.57851	0.01882	0.01240	0.0877
246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	...	36.48391	37.58333	23.43353	267.23367	1284.0	0.52483	0.34886	0.01426	0.01426	0.0903

2 rows × 79 columns

In [14]:
```
df_test.head(2)
```

Out[14]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	...	34.78682	33.75000	21.58531	416.48097	1938.0	0.70252	0.28217	0.05910	0.03813	0.14299
252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	...	44.23451	46.66667	22.37036	532.03505	1950.0	0.85128	0.64221	0.02338	0.00000	0.13377

2 rows × 79 columns

3. 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 [15]:
```
#percentage of missing values in train set
df_train.isnull().sum()
```

Out[15]:
```
BLOCKID        27321
SUMLEVEL           0
COUNTYID           0
STATEID            0
state              0
               ...
pct_own          268
married          191
married_snp      191
separated        191
divorced         191
Length: 79, dtype: int64
```

In [16]:
```
missing_list_train=df_train.isnull().sum() *100/len(df_train)
missing_values_df_train=pd.DataFrame(missing_list_train,columns=['Percentage of missing values'])
missing_values_df_train.sort_values(by=['Percentage of missing values'],inplace=True,ascending=False)
missing_values_df_train[missing_values_df_train['Percentage of missing values']>0][:10]
```

Out[16]:

	Percentage of missing values
BLOCKID	100.000000
hc_samples	2.196113
hc_mean	2.196113
hc_median	2.196113
hc_stdev	2.196113
hc_sample_weight	2.196113
hc_mortgage_mean	2.097288
hc_mortgage_stdev	2.097288
hc_mortgage_sample_weight	2.097288
hc_mortgage_samples	2.097288

BLOCKID can be dropped,since it is 100% missing values

In [17]:
```
#percentage of missing values in test set
missing_list_test=df_test.isnull().sum() *100/len(df_train)
missing_values_df_test=pd.DataFrame(missing_list_test,columns=['Percentage of missing values'])
missing_values_df_test.sort_values(by=['Percentage of missing values'],inplace=True,ascending=False)
missing_values_df_test[missing_values_df_test['Percentage of missing values']>0][:10]
```

Out[17]:

	Percentage of missing values
BLOCKID	42.857143
hc_samples	1.061455
hc_mean	1.061455
hc_median	1.061455
hc_stdev	1.061455
hc_sample_weight	1.061455
hc_mortgage_mean	0.980930
hc_mortgage_stdev	0.980930
hc_mortgage_sample_weight	0.980930
hc_mortgage_samples	0.980930

BLOCKID can be dropped, since it is 43%missing values

In [18]:
```
#since SUMLEVEL does not have any predictive power and no variance therefore we will droppeed it.
df_train.drop(columns=['BLOCKID','SUMLEVEL'],inplace=True)
```

In [19]:
```
#since SUMLEVEL does not have any predictive power and no variance therefore we will droppeed it.
df_test.drop(columns=['BLOCKID','SUMLEVEL'],inplace=True)
```

In [20]:
```
df_train.head(1)
```

Out[20]:

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.57851	0.01882	0.0124	0.0877

1 rows × 77 columns

In [21]:
```
df_test.head(1)
```

Out[21]:

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
255504	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	48239	313	...	34.78682	33.75	21.58531	416.48097	1938.0	0.70252	0.28217	0.0591	0.03813	0.14299

1 rows × 77 columns

In [22]:
```
#Imputing missing values with mean
missing_train_cols=[]
for col in df_train.columns:
    if df_train[col].isna().sum() !=0:
        missing_train_cols.append(col)
print(missing_train_cols)
```

```
['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 [23]:
```
#Imputing missing values with mean
missing_test_cols=[]
for col in df_test.columns:
    if df_test[col].isna().sum() !=0:
        missing_test_cols.append(col)
print(missing_test_cols)
```

```
['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]:
```
#Missing cols are all numerical variables
for col in df_train.columns:
    if col in (missing_train_cols):
        df_train[col].replace(np.nan, df_train[col].mean(),inplace=True)
```

In [25]:
```
#Missing cols are all numerical variables
for col in df_test.columns:
    if col in (missing_test_cols):
        df_test[col].replace(np.nan, df_test[col].mean(),inplace=True)
```

In [26]:
```
df_train.isna().sum().sum()
```

Out[26]: 0

In [27]:
```
df_train.isna().sum().sum()
```

Out[27]: 0

Exploratory Data Analysis (EDA):

Perform debt analysis. You may take the following steps:

a) 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 [29]:
```
!pip install pandasql
```

```
Collecting pandasql
  Using cached pandasql-0.7.3.tar.gz (26 kB)
Requirement already satisfied: numpy in d:\anaconda\lib\site-packages (from pandasql) (1.21.5)
Requirement already satisfied: pandas in d:\anaconda\lib\site-packages (from pandasql) (1.4.2)
Requirement already satisfied: sqlalchemy in d:\anaconda\lib\site-packages (from pandasql) (1.4.32)
Requirement already satisfied: pytz>=2020.1 in d:\anaconda\lib\site-packages (from pandas->pandasql) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in d:\anaconda\lib\site-packages (from pandas->pandasql) (2.8.2)
Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from python-dateutil>=2.8.1->pandas->pandasql) (1.16.0)
Requirement already satisfied: greenlet!=0.4.17 in d:\anaconda\lib\site-packages (from sqlalchemy->pandasql) (1.1.1)
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py): started
  Building wheel for pandasql (setup.py): finished with status 'done'
  Created wheel for pandasql: filename=pandasql-0.7.3-py3-none-any.whl size=26784 sha256=cb144b49e56a0834c47d1cbec46553e6175f70d60cfb1cb1ab1e47a22c6fdb3f
  Stored in directory: c:\users\chinm\appdata\local\pip\cache\wheels\63\e8\ec\75b1df467ecf57b6ececb32cb16f4e86697cbfe55cb0c51f07
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

In [30]:
```
from pandasql import sqldf
q1 = "select place,pct_own,second_mortgage,lat,lng from df_train where pct_own >0.10 and second_mortgage <0.5 order by second_mortgage DESC LIMIT 2500;"
pysqldf = lambda q: sqldf(q, globals())
df_train_location_mort_pct=pysqldf(q1)
```

In [31]:
```
df_train_location_mort_pct.head()
```

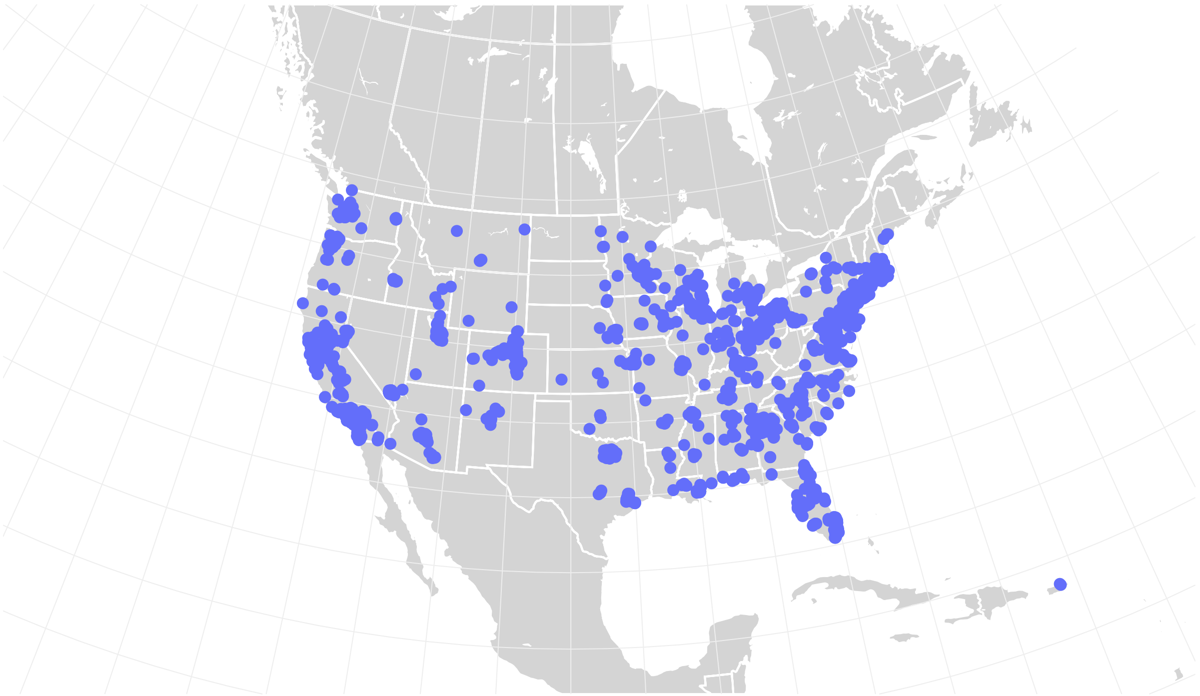
Out[31]:

	place	pct_own	second_mortgage	lat	lng
0	Worcester City	0.20247	0.43363	42.254262	-71.800347
1	Harbor Hills	0.15618	0.31818	40.751809	-73.853582
2	Glen Burnie	0.22380	0.30212	39.127273	-76.635265
3	Egypt Lake-leto	0.11618	0.28972	28.029063	-82.495395
4	Lincolnwood	0.14228	0.28899	41.967289	-87.652434

In [32]:
```
import plotly.express as px
import plotly.graph_objects as go
```

In [34]:
```
fig = go.Figure(data=go.Scattergeo(
    lat = df_train_location_mort_pct['lat'],
    lon = df_train_location_mort_pct['lng']),
    )
fig.update_layout(
    geo=dict(
        scope = 'north america',
        showland = True,
        landcolor = "rgb(212, 212, 212)",
        subunitcolor = "rgb(255, 255, 255)",
        countrycolor = "rgb(255, 255, 255)",
        showlakes = True,
        lakecolor = "rgb(255, 255, 255)",
        showsubunits = True,
        showcountries = True,
        resolution = 50,
        projection = dict(
            type = 'conic conformal',
            rotation_lon = -100
        ),
        lonaxis = dict(
            showgrid = True,
            gridwidth = 0.5,
            range= [ -140.0, -55.0 ],
            dtick = 5
        ),
        lataxis = dict (
            showgrid = True,
            gridwidth = 0.5,
            range= [ 20.0, 60.0 ],
            dtick = 5
        )
    ),
    title='Top 2,500 locations with second mortgage is the highest and percent ownership is above 10 percent')
fig.show()
```

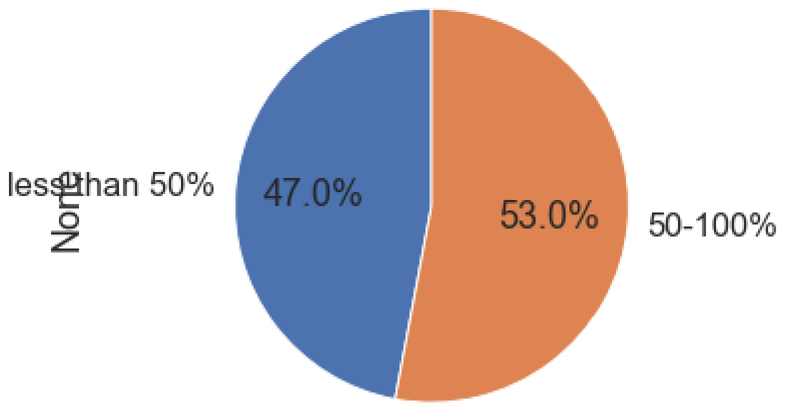
Top 2,500 locations with second mortgage is the highest and percent ownership is above 10 percent



Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage c) Create pie charts to show overall debt and bad debt

In [35]:
```
df_train['bad_debt']=df_train['second_mortgage']+df_train['home_equity']-df_train['home_equity_second_mortgage']
```

In [36]:
```
df_train['bins'] = pd.cut(df_train['bad_debt'],bins=[0,0.10,1], labels=["less than 50%","50-100%"])
df_train.groupby(['bins']).size().plot(kind='pie',subplots=True,startangle=90, autopct='%1.1f%%')
plt.axis('equal')

plt.show()
#df.plot.pie(subplots=True,figsize=(8, 3))
```



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

In [37]:
```
cols=[]
df_train.columns
```

Out[37]:
```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
       'pop', 'male_pop', 'female_pop', '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', 'universe_samples', 'used_samples',
       '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',
       'bad_debt', 'bins'],
      dtype='object')
```

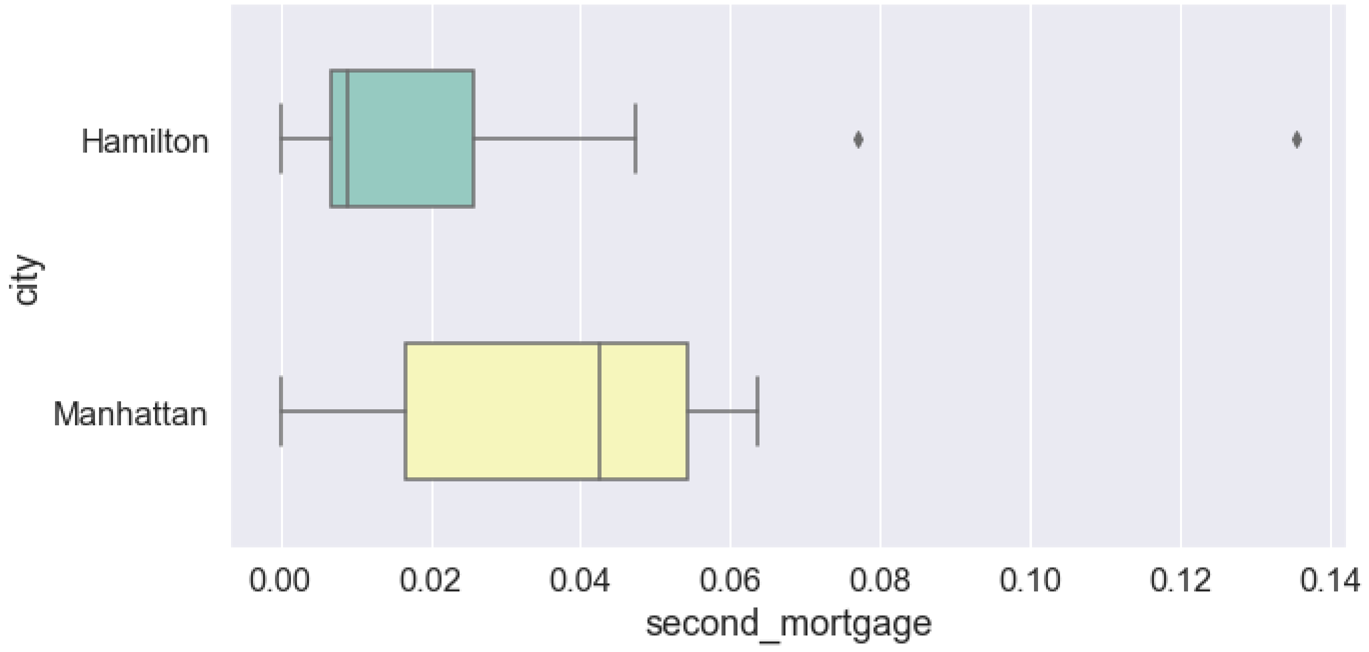
In [38]:
```
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
df_box_hamilton=df_train.loc[df_train['city'] == 'Hamilton']
df_box_manhattan=df_train.loc[df_train['city'] == 'Manhattan']
df_box_city=pd.concat([df_box_hamilton,df_box_manhattan])
df_box_city.head(4)
```

Out[38]:

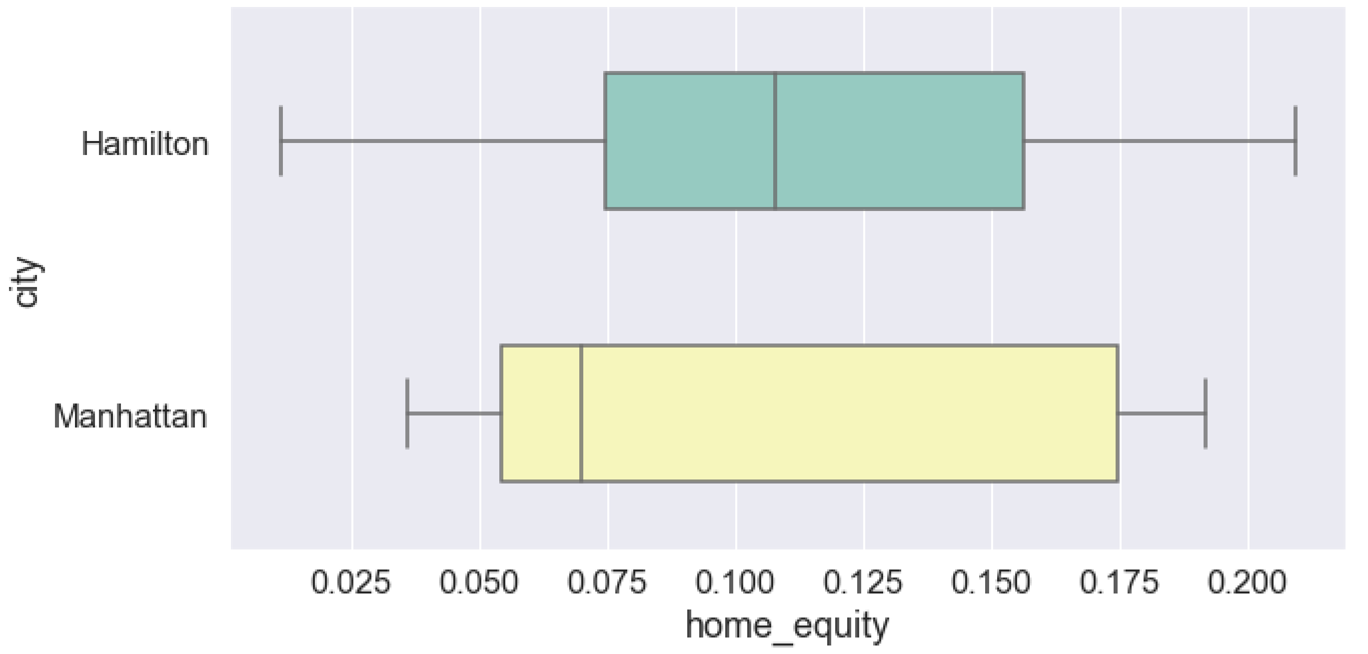
UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	...	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced	bad_debt	bins
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	...	22.51276	685.33845	2618.0	0.79046	0.57851	0.01882	0.01240	0.08770	0.09408	less than 50%
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	8610	609	...	24.05831	732.58443	3124.0	0.64400	0.56377	0.01980	0.00990	0.04892	0.18071	50-100%
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	45015	513	...	22.66500	565.32725	2528.0	0.61278	0.47397	0.04419	0.02663	0.13741	0.15005	50-100%
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	39746	662	...	22.79602	483.01311	1954.0	0.83241	0.58678	0.01052	0.00000	0.11721	0.02130	less than 50%

4 rows × 79 columns

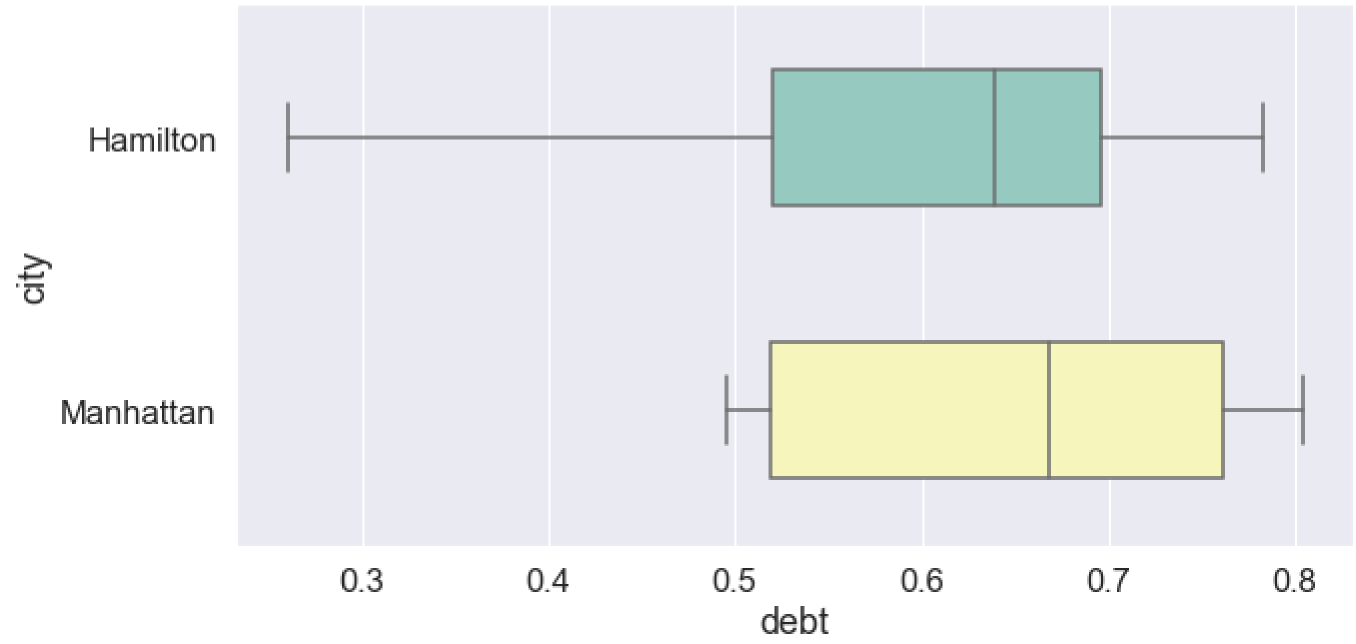
In [39]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.5,palette="Set3")
plt.show()
```



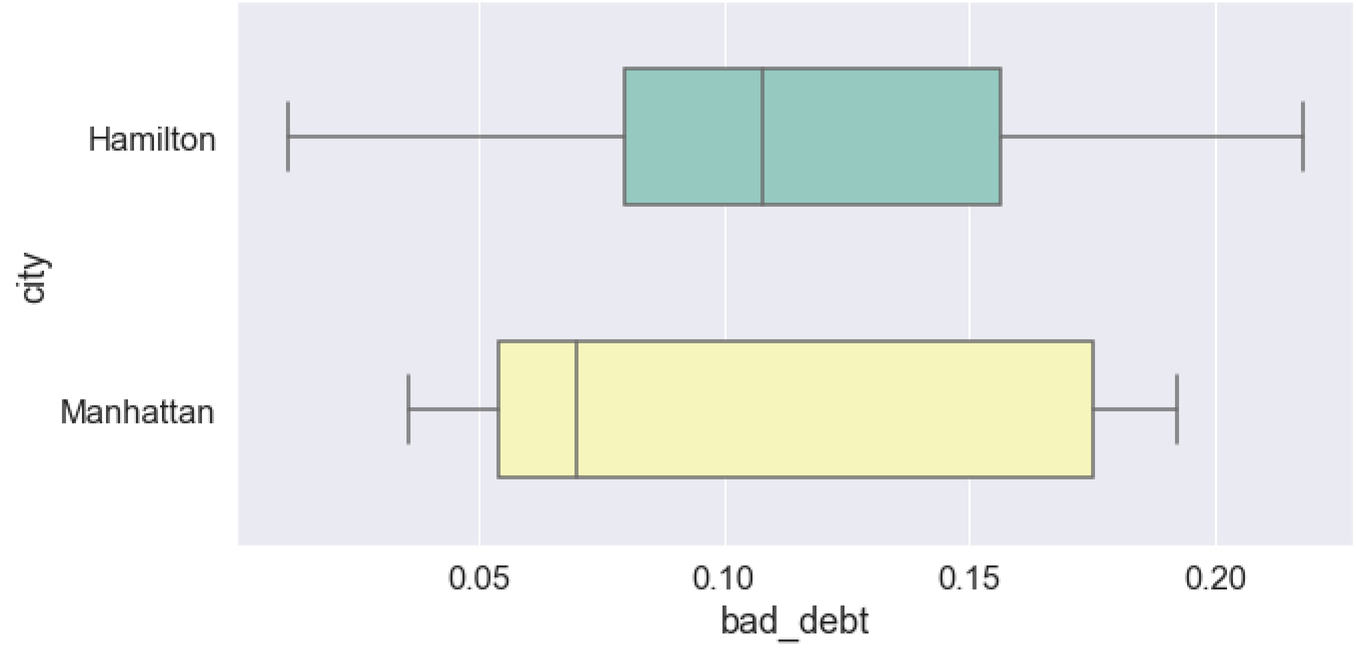
In [40]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



In [41]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```



In [42]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='bad_debt', y='city',width=0.5,palette="Set3")
plt.show()
```

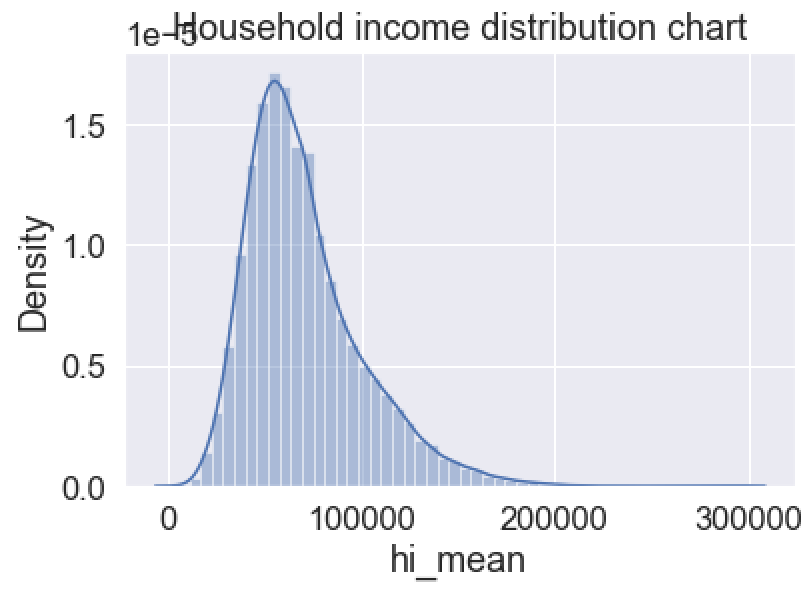


Manhattan has higher metrics compared to Hamilton

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

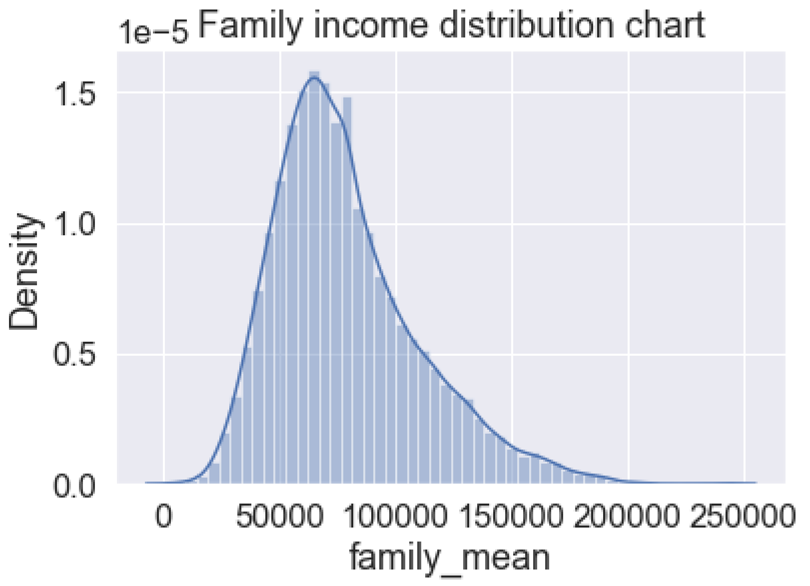
In [43]:
```
sns.distplot(df_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



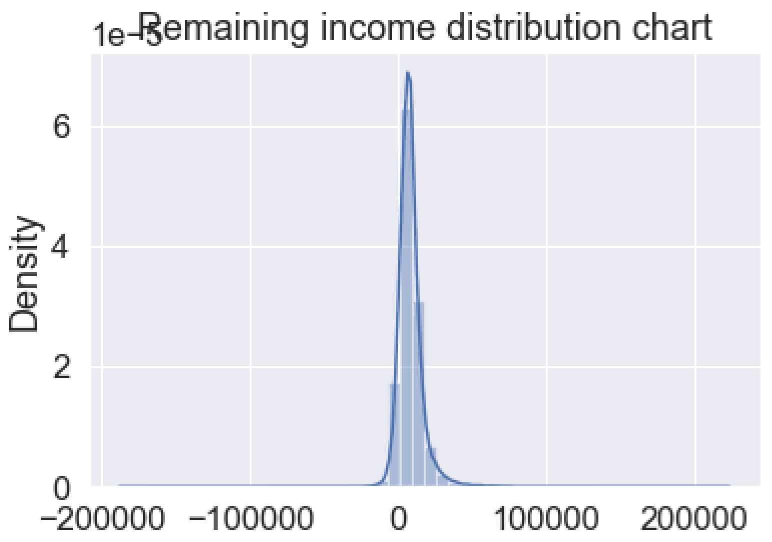
In [44]:
```
sns.distplot(df_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [45]:
```
sns.distplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

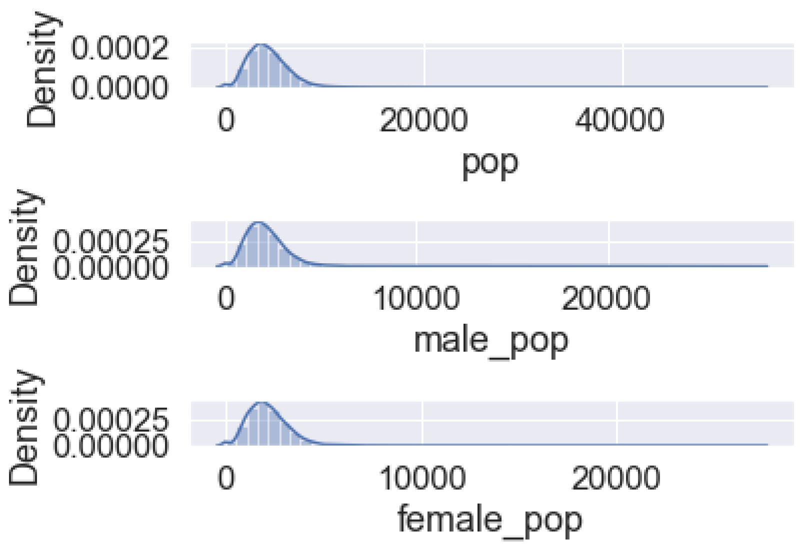


Income distribution almost has normality in its distrbution

Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):

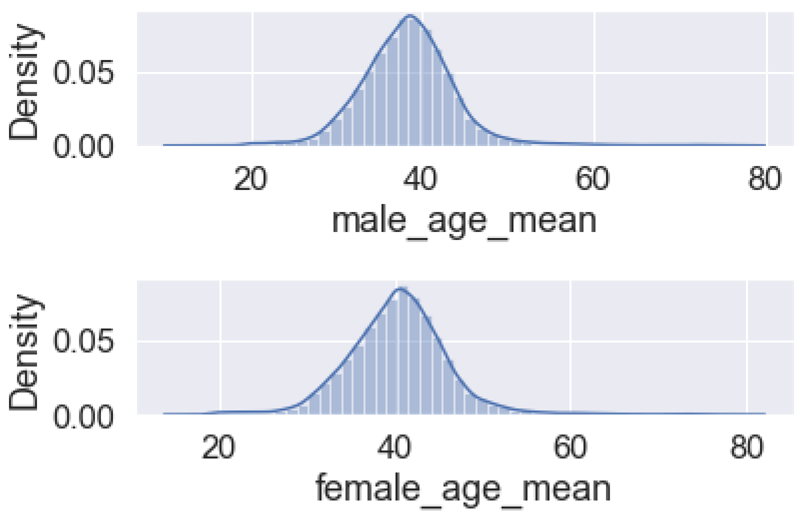
In [46]:
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2,ax3)=plt.subplots(3,1)
sns.distplot(df_train['pop'],ax=ax1)
sns.distplot(df_train['male_pop'],ax=ax2)
sns.distplot(df_train['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [47]:
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2)=plt.subplots(2,1)
sns.distplot(df_train['male_age_mean'],ax=ax1)
sns.distplot(df_train['female_age_mean'],ax=ax2)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



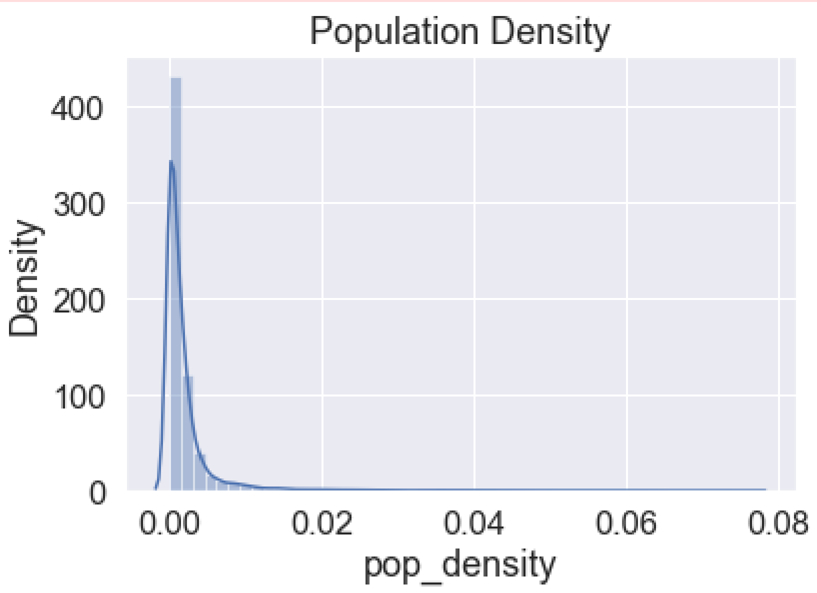
a) Use pop and ALand variables to create a new field called population density

In [50]:
```
df_train['pop_density']=df_train['pop']/df_train['ALand']
```

In [51]:
```
df_test['pop_density']=df_test['pop']/df_test['ALand']
```

In [52]:
```
sns.distplot(df_train['pop_density'])
plt.title('Population Density')
plt.show() # Very less density is noticed
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age c) Visualize the findings using appropriate chart type

In [55]:
```
df_train['age_median']=(df_train['male_age_median']+df_train['female_age_median'])/2
df_test['age_median']=(df_test['male_age_median']+df_test['female_age_median'])/2
```

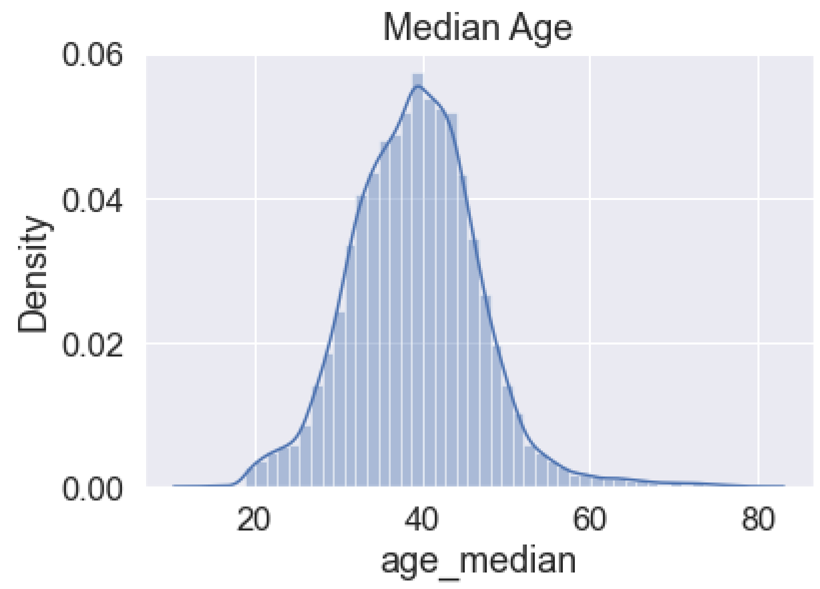
In [56]:
```
df_train[['male_age_median','female_age_median','male_pop','female_pop','age_median']].head()
```

Out[56]:

	male_age_median	female_age_median	male_pop	female_pop	age_median
UID					
267822	44.00000	45.33333	2612	2618	44.666665
246444	32.00000	37.58333	1349	1284	34.791665
245683	40.83333	42.83333	3643	3238	41.833330
279653	48.91667	50.58333	1141	1559	49.750000
247218	22.41667	21.58333	2586	3051	22.000000

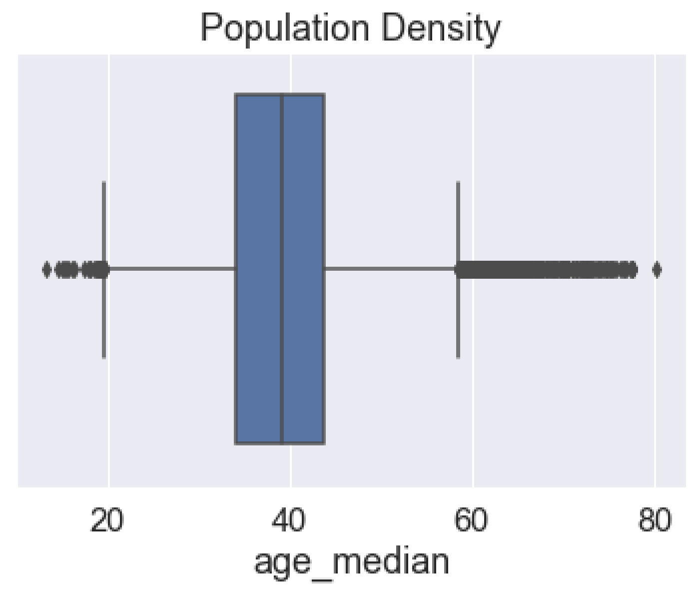
In [57]:
```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
# Age of population is mostly between 20 and 60
# Majority are of age around 40
# Median age distribution has a gaussian distribution
# Some right skewness is noticed
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [58]:
```
sns.boxplot(df_train['age_median'])
plt.title('Population Density')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



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 [59]:
```
df_train['pop'].describe()
```

Out[59]:
```
count    27321.000000
mean      4316.032685
std       2169.226173
min          0.000000
25%       2885.000000
50%       4042.000000
75%       5430.000000
max      53812.000000
Name: pop, dtype: float64
```

In [60]:
```
df_train['pop_bins']=pd.cut(df_train['pop'],bins=5,labels=['very low','low','medium','high','very high'])
```

In [61]:
```
df_train[['pop','pop_bins']]
```

Out[61]:

UID	pop	pop_bins
267822	5230	very low
246444	2633	very low
245683	6881	very low
279653	2700	very low
247218	5637	very low
...	...	...
279212	1847	very low
277856	4155	very low
233000	2829	very low
287425	11542	low
265371	3726	very low

27321 rows × 2 columns

In [63]:
```
df_train['pop_bins'].value_counts()
```

Out[63]:
```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

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

In [64]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

Out[64]:

pop_bins	married	separated	divorced
very low	27058	27058	27058
low	246	246	246
medium	9	9	9
high	7	7	7
very high	1	1	1

In [65]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"])
```

Out[65]:

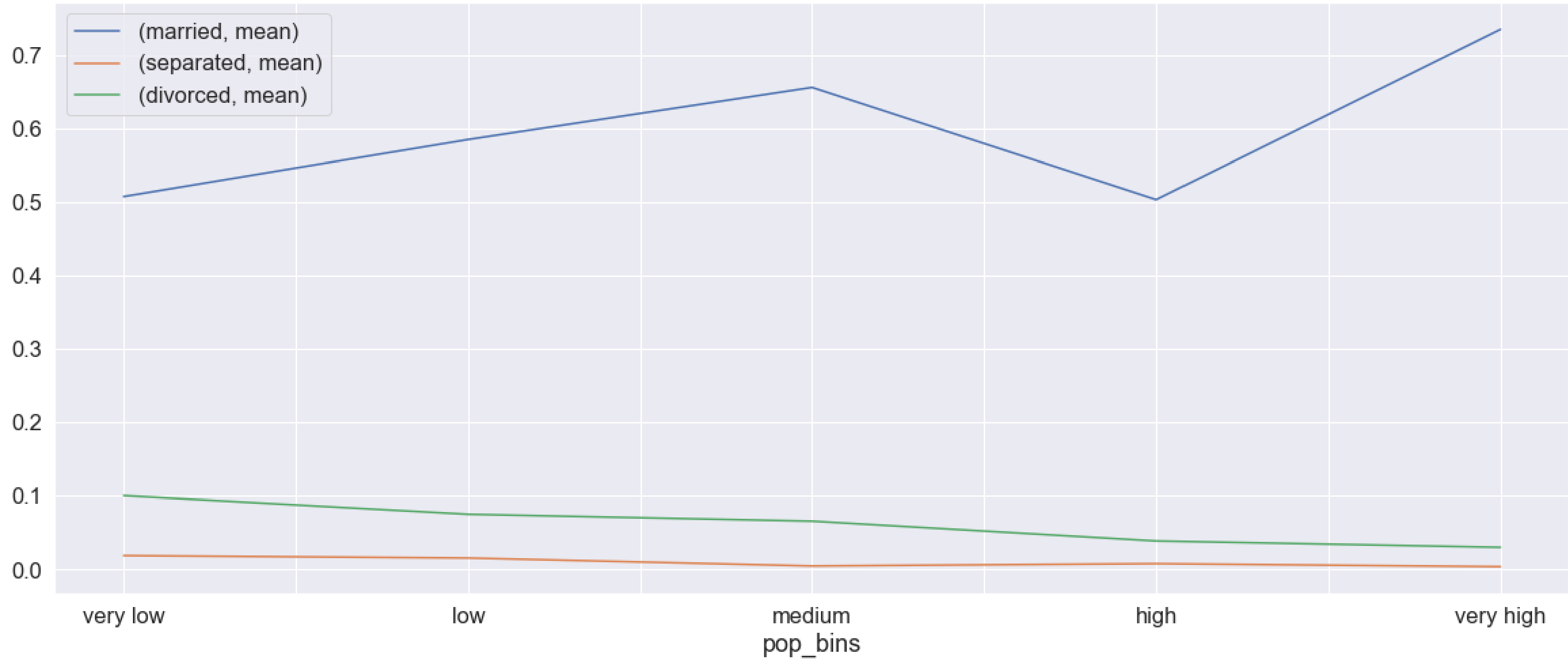
	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.507548	0.524680	0.019126	0.013650	0.100504	0.096020
low	0.584894	0.593135	0.015833	0.011195	0.075348	0.070045
medium	0.655737	0.618710	0.005003	0.004120	0.065927	0.064890
high	0.503359	0.335660	0.008141	0.002500	0.039030	0.010320
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.030360

1.Very high population group has more married people and less percantage of separated and divorced couples 2.In very low population groups, there are more divorced people

Visualize using appropriate chart type

In [66]:
```
plt.figure(figsize=(10,5))
pop_bin_married=df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean"])
pop_bin_married.plot(figsize=(20,8))
plt.legend(loc='best')
plt.show()
```

```
<Figure size 720x360 with 0 Axes>
```



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

In [67]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

Out[67]:

state	mean
Alabama	774.004927
Alaska	1185.763570
Arizona	1097.753511
Arkansas	720.918575
California	1471.133857

In [68]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

Out[68]:

state	mean
Alabama	67030.064213
Alaska	92136.545109
Arizona	73328.238798
Arkansas	64765.377850
California	87655.470820

In [69]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

Out[69]:
```
state
Alabama                 0.011547
Alaska                  0.012870
Arizona                 0.014970
Arkansas                0.011131
California              0.016783
Colorado                0.013529
Connecticut             0.012637
Delaware                0.012929
District of Columbia    0.013198
Florida                 0.015772
Name: mean, dtype: float64
```

In [70]:
```
#overall level rent as a percentage of income
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

Out[70]:
```
0.013358170721473864
```

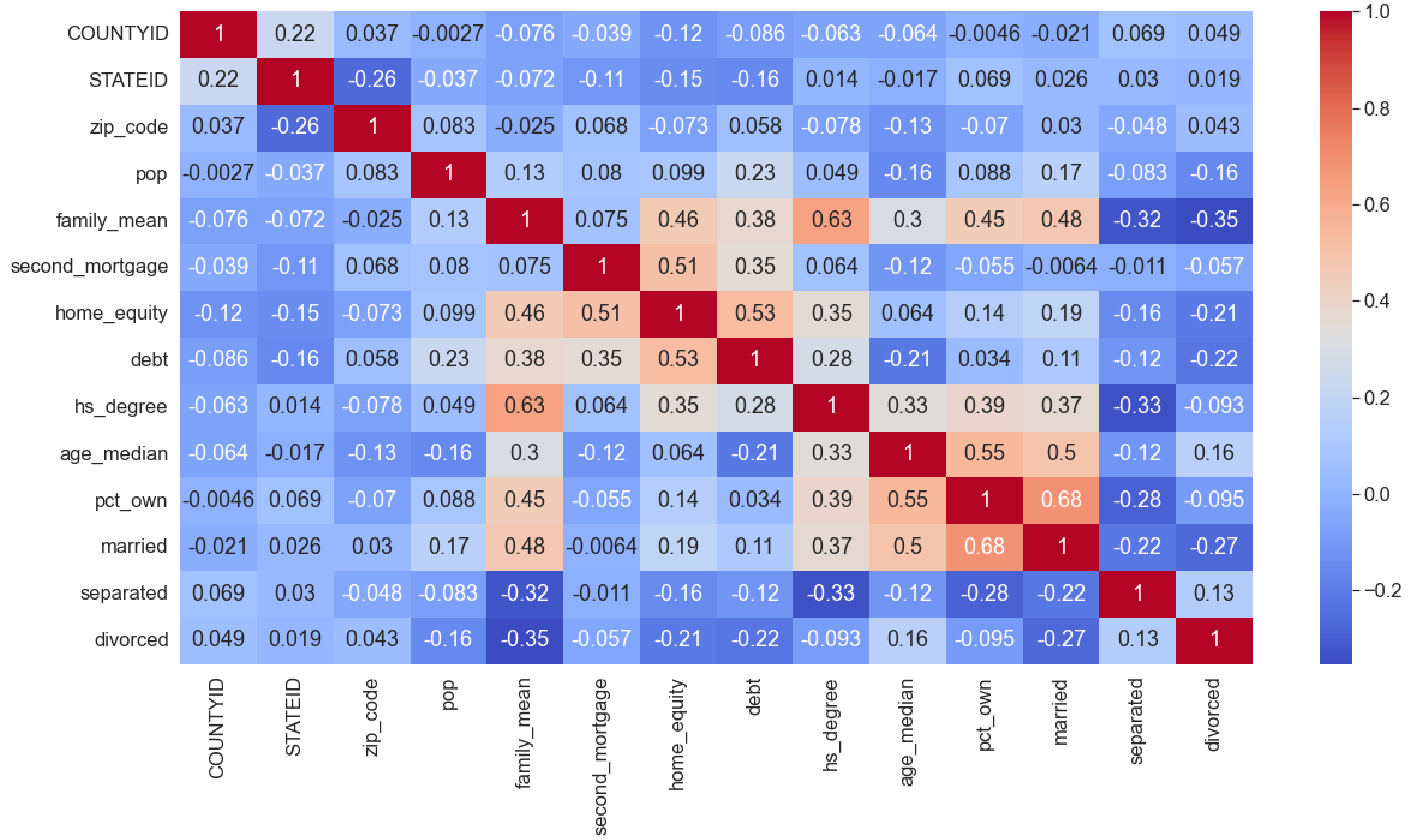
Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

In [71]:
```
df_train.columns
```

Out[71]:
```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
       'pop', 'male_pop', 'female_pop', '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', 'universe_samples', 'used_samples',
       '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',
       'bad_debt', 'bins', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

In [72]:
```
cor=df_train[['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']].corr()
```

In [73]:
```
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```



1.High positive correaltion is noticed between pop, male_pop and female_pop 2.High positive correaltion is noticed between rent_mean,hi_mean, family_mean,hc_mean

1. 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. 2. Each variable is assumed to be dependent upon a linear combination of the common

factors, and the coefficients are known as loadings. Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable. Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data. Following are the list of latent variables:

In [75]:
```
!pip install factor_analyzer
```

```
Collecting factor_analyzer
  Downloading factor_analyzer-0.4.1.tar.gz (41 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
    Preparing wheel metadata: started
    Preparing wheel metadata: finished with status 'done'
Requirement already satisfied: numpy in d:\anaconda\lib\site-packages (from factor_analyzer) (1.21.5)
Collecting pre-commit
  Downloading pre_commit-2.20.0-py2.py3-none-any.whl (199 kB)
Requirement already satisfied: pandas in d:\anaconda\lib\site-packages (from factor_analyzer) (1.4.2)
Requirement already satisfied: scipy in d:\anaconda\lib\site-packages (from factor_analyzer) (1.7.3)
Requirement already satisfied: scikit-learn in d:\anaconda\lib\site-packages (from factor_analyzer) (1.0.2)
Requirement already satisfied: pytz>=2020.1 in d:\anaconda\lib\site-packages (from pandas->factor_analyzer) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in d:\anaconda\lib\site-packages (from pandas->factor_analyzer) (2.8.2)
Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from python-dateutil>=2.8.1->pandas->factor_analyzer) (1.16.0)
Requirement already satisfied: toml in d:\anaconda\lib\site-packages (from pre-commit->factor_analyzer) (0.10.2)
Collecting identify>=1.0.0
  Downloading identify-2.5.9-py2.py3-none-any.whl (98 kB)
Requirement already satisfied: pyyaml>=5.1 in d:\anaconda\lib\site-packages (from pre-commit->factor_analyzer) (6.0)
Collecting cfgv>=2.0.0
  Downloading cfgv-3.3.1-py2.py3-none-any.whl (7.3 kB)
Collecting nodeenv>=0.11.1
  Downloading nodeenv-1.7.0-py2.py3-none-any.whl (21 kB)
Collecting virtualenv>=20.0.8
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
Requirement already satisfied: setuptools in d:\anaconda\lib\site-packages (from nodeenv>=0.11.1->pre-commit->factor_analyzer) (61.2.0)
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
Collecting platformdirs<3,>=2.4
  Downloading platformdirs-2.6.0-py3-none-any.whl (14 kB)
Requirement already satisfied: filelock<4,>=3.4.1 in d:\anaconda\lib\site-packages (from virtualenv>=20.0.8->pre-commit->factor_analyzer) (3.6.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda\lib\site-packages (from scikit-learn->factor_analyzer) (2.2.0)
Requirement already satisfied: joblib>=0.11 in d:\anaconda\lib\site-packages (from scikit-learn->factor_analyzer) (1.1.0)
Building wheels for collected packages: factor-analyzer
  Building wheel for factor-analyzer (PEP 517): started
  Building wheel for factor-analyzer (PEP 517): finished with status 'done'
  Created wheel for factor-analyzer: filename=factor_analyzer-0.4.1-py2.py3-none-any.whl size=42070 sha256=9196c13f91cc75f43672a9027543d91ff299790e41bea972b3cc5cb4c5b7111a
  Stored in directory: c:\users\chinm\appdata\local\pip\cache\wheels\6d\32\bd\460a71becd83f7d77152f437c2fd451f5c87bc19cfcdbfcd24
Successfully built factor-analyzer
Installing collected packages: platformdirs, distlib, virtualenv, nodeenv, identify, cfgv, pre-commit, factor-analyzer
Successfully installed cfgv-3.3.1 distlib-0.3.6 factor-analyzer-0.4.1 identify-2.5.9 nodeenv-1.7.0 platformdirs-2.6.0 pre-commit-2.20.0 virtualenv-20.17.1
```

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

In [76]:
```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [77]:
```
#pip install factor_analyzer
```

In [78]:
```
fa=FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

Out[78]:
```
array([[-1.12589169e-01,  1.95646471e-02, -2.39331085e-02,
        -6.27632640e-02,  4.23474734e-02],
       [-1.10186765e-01,  1.33506216e-02,  2.79651248e-02,
        -1.49825865e-01,  1.10838805e-01],
       [-8.28678643e-02,  5.16372375e-02, -1.36451870e-01,
        -4.98918621e-02, -1.04024841e-01],
       [ 1.80961144e-02,  1.92013754e-02,  5.81329911e-03,
         2.64842754e-02, -6.12442613e-03],
       [ 9.02324752e-02, -9.72544302e-02, -6.54601348e-02,
        -1.33145902e-01, -1.48594599e-01],
       [-1.07335694e-02, -4.12376817e-02,  1.45853485e-01,
         8.80433300e-03,  1.08227566e-01],
       [-4.28796983e-02, -2.09780216e-02,  3.66726852e-02,
        -9.45597414e-02,  5.91380522e-02],
       [-2.44243060e-03, -1.53245408e-02, -2.68300863e-03,
        -4.52473036e-02,  2.37240647e-02],
       [ 7.92164330e-02,  9.57453327e-01, -8.71151629e-02,
        -6.59923762e-03, -3.97273199e-02],
       [ 7.39808202e-02,  9.18750524e-01, -1.08834839e-01,
        -2.79371580e-02, -3.93153658e-02],
       [ 8.06598890e-02,  9.47839215e-01, -6.08006500e-02,
         1.53627099e-02, -3.86977286e-02],
       [ 7.70052111e-01,  9.84675379e-03, -3.71249730e-02,
         1.14949036e-01, -1.23784689e-01],
       [ 7.18615870e-01,  6.24980470e-03, -4.59787391e-02,
         1.09109686e-01, -1.35301914e-01],
       [ 7.07647228e-01,  2.46625391e-02, -1.00860835e-02,
         1.04472482e-01,  7.72381149e-02],
       [-1.34545482e-01,  3.36809304e-01, -4.87894972e-01,
        -4.15446259e-02,  3.17608557e-01],
       [ 2.31079711e-01,  4.37729793e-01, -6.40209208e-01,
        -2.52310994e-02,  3.47216241e-01],
       [-4.52068102e-02,  3.51263840e-02,  3.07536975e-02,
         4.44793493e-01, -1.63273402e-01],
       [-2.50717052e-02,  1.70166793e-02,  4.57227185e-02,
         6.76083880e-01, -1.55256757e-01],
       [-3.90694451e-02, -1.67460878e-02,  8.13962799e-02,
         8.36389142e-01, -9.18259833e-02],
       [-5.14161961e-02, -3.57207140e-02,  1.10795178e-01,
         9.25123762e-01, -4.44866488e-02],
       [-6.08590014e-02, -4.41860614e-02,  1.35794017e-01,
         9.53019900e-01, -2.21548635e-02],
       [-4.57771192e-02, -5.25526120e-02,  1.41019867e-01,
         9.32702618e-01, -5.84369519e-07],
       [-4.19486075e-02, -5.90387636e-02,  1.28851776e-01,
         8.87316670e-01,  1.05894303e-02],
       [-2.47894677e-02, -7.29670547e-02,  9.41510379e-02,
         7.79023652e-01,  2.95352817e-02],
       [ 2.12258458e-01,  4.65992344e-01, -6.14495945e-01,
        -2.47660022e-02,  3.66644543e-01],
       [ 2.33057252e-01,  4.47057849e-01, -6.28263424e-01,
        -2.71547728e-02,  3.43419633e-01],
       [ 7.85157101e-01,  4.91249252e-02,  1.44540484e-01,
        -2.05217631e-01, -1.54523366e-01],
       [ 7.10324888e-01,  4.99730434e-02,  1.32239990e-01,
        -2.19171866e-01, -2.10505580e-01],
       [ 8.61780947e-01,  4.35044827e-02,  1.65839098e-01,
        -1.19850814e-01,  3.16733580e-02],
       [-2.23443271e-01,  8.46259550e-01, -4.61177184e-02,
         6.88599251e-02,  2.27742322e-01],
       [ 1.43837558e-01,  9.53197416e-01,  2.27887461e-02,
        -4.57890454e-02,  1.00796451e-01],
       [ 8.30286504e-01,  3.42026000e-02,  1.61106001e-01,
        -2.04570331e-01, -7.48710468e-02],
       [ 7.94476573e-01,  2.83818589e-02,  1.51219547e-01,
        -2.07681492e-01, -9.12497145e-02],
       [ 8.11481641e-01,  4.32314878e-02,  1.43645560e-01,
        -1.07778260e-01,  5.79540090e-02],
       [-3.37741909e-01,  8.64927624e-01,  3.58933716e-02,
         9.07183972e-02,  4.46327258e-02],
       [ 5.03572647e-02,  9.35515353e-01,  1.51475405e-01,
        -2.51501245e-02, -9.34471652e-02],
       [ 9.78242259e-01, -3.31490292e-02, -1.05261174e-01,
         4.50364278e-02,  7.37362139e-02],
       [ 9.59137182e-01, -3.90023003e-02, -1.20630334e-01,
         4.52591426e-02,  6.64877184e-02],
       [ 8.14087200e-01,  2.23057300e-03,  7.66518549e-02,
         2.02747473e-02,  1.27634839e-01],
       [-4.15353990e-01,  7.18339587e-01,  3.40068068e-01,
        -7.18402763e-02, -2.77950522e-01],
       [ 7.64912665e-02,  7.24900629e-01,  2.74193203e-01,
        -4.83952627e-02, -3.52988286e-01],
       [ 9.10390829e-01, -5.36541209e-02, -4.68641801e-02,
        -7.64182945e-04,  1.63870438e-01],
       [ 8.73011859e-01, -5.30302299e-02, -5.89943093e-02,
        -1.58989714e-03,  1.52417538e-01],
       [ 7.55087682e-01, -3.56133752e-03,  5.39542598e-02,
         4.24181558e-03,  2.58043493e-01],
       [-1.23469887e-01,  6.07438129e-01,  6.33039230e-01,
        -2.14798834e-02,  2.47973916e-01],
       [-3.42866889e-01,  5.59526271e-01,  5.88212998e-01,
        -2.51533562e-02,  2.18419877e-01],
```

```
       [-1.60867224e-01, -1.53062632e-02, -1.57026591e-01,
         1.09243763e-01, -6.61660849e-01],
       [-1.37306756e-01, -2.17250639e-02, -1.58408936e-01,
         1.25156196e-01, -6.71630798e-01],
       [ 2.45096195e-01, -2.54584574e-02, -2.66691493e-02,
         9.53148481e-02, -6.42510821e-01],
       [ 2.03988665e-01,  7.85172846e-02, -3.01656227e-01,
         2.28379497e-02, -6.29223348e-01],
       [ 1.08926078e-01, -6.34332397e-02, -3.36565155e-02,
        -9.49480488e-02,  6.81473836e-01],
       [-2.63787634e-01, -6.43281165e-03, -3.58792108e-02,
        -9.37962462e-02,  6.47816991e-01],
       [-2.15717052e-01, -7.36588970e-02,  3.50113236e-01,
        -1.95201638e-02,  6.36783756e-01],
       [ 3.94306152e-01,  6.09565683e-02,  2.55337862e-01,
        -2.20362100e-01, -1.84248078e-01],
       [ 4.07877889e-01,  6.27256506e-02,  2.23926902e-01,
        -2.10028730e-01, -1.71989214e-01],
       [ 3.53156880e-01,  5.36715651e-02,  2.69603564e-01,
        -2.16933217e-01, -1.80072063e-01],
       [ 2.33537266e-01, -4.91732975e-02,  8.14450794e-01,
         9.36688907e-02,  3.27131938e-01],
       [ 2.40298207e-01, -3.38140127e-02,  8.31496967e-01,
         7.52417525e-02,  2.46323604e-01],
       [-6.71839458e-02,  6.58504508e-02,  5.86207669e-01,
         8.72955131e-02,  9.12541341e-02],
       [ 5.59835538e-02,  8.17918702e-01, -1.78458349e-01,
        -1.55949421e-02, -3.34299756e-02],
       [ 7.16426394e-02,  9.23428542e-01, -1.07142694e-01,
        -2.78635363e-02, -4.35991136e-02],
       [ 1.92496950e-01, -4.75870411e-02,  8.03173200e-01,
         1.43492711e-01,  3.33862159e-01],
       [ 1.87644438e-01, -3.29941019e-02,  8.58024513e-01,
         1.31329962e-01,  2.55679735e-01],
       [-1.02263658e-01,  6.03984282e-02,  4.72982273e-01,
         7.36848442e-02,  1.12273914e-01],
       [ 6.14776643e-02,  8.77962758e-01, -1.50410287e-01,
         2.20991054e-02, -4.17158193e-02],
       [ 7.83728221e-02,  9.54508804e-01, -5.91095897e-02,
         1.64800957e-02, -4.32591005e-02],
       [-3.24381874e-02,  1.11167162e-01,  7.84467391e-01,
        -4.37718597e-02, -2.80931234e-01],
       [ 1.76682388e-01,  1.90494238e-01,  5.61405491e-01,
        -1.20746166e-01, -1.32570792e-01],
       [-6.37386620e-02, -7.03047918e-02, -2.68934066e-01,
         1.28589794e-01,  1.88507864e-01],
       [-1.56051271e-01, -7.08033942e-02, -1.45964501e-01,
         1.24253736e-01,  1.46293121e-01],
       [-3.56716298e-01, -5.29910747e-02,  1.47771609e-01,
         2.87196191e-02,  1.13159582e-01],
       [ 2.42173836e-01, -2.86199110e-02, -3.25958426e-02,
         1.05027818e-01, -6.55406061e-01],
       [ 3.50196743e-01, -1.05016404e-02, -3.95274112e-01,
         5.92876795e-02,  2.91651787e-01],
       [ 2.25671548e-01, -3.42672769e-02,  8.92876631e-01,
         1.12426812e-01,  2.67065205e-01]])
```

Data Modeling : Linear Regression

1.Build a linear Regression model to predict the total monthly expenditure for home mortgages loan. Please refer 'deplotment_RE.xlsx'. Column hc_mortgage_mean is predicted variable. This is the mean monthly mortgage and owner costs of specified geographical location. Note: Exclude loans from prediction model which have NaN (Not a Number) values for hc_mortgage_mean.

In [79]:
```
df_train.columns
```

Out[79]:
```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
       'pop', 'male_pop', 'female_pop', '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', 'universe_samples', 'used_samples',
       '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',
       'bad_debt', 'bins', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

In [81]:
```
df_train['type'].unique()
type_dict={'type':{'City':1,
                   'Urban':2,
                   'Town':3,
                   'CDP':4,
```

```
                'Village':5,
                'Borough':6}
          }
df_train.replace(type_dict,inplace=True)
```

In [82]:
```
df_train['type'].unique()
```
Out[82]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [83]:
```
df_test.replace(type_dict,inplace=True)
```

In [84]:
```
df_test['type'].unique()
```
Out[84]:
```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

In [85]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']
```

In [86]:
```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

In [87]:
```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

In [88]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error,accuracy_score
```

In [89]:
```
x_train.head()
```
Out[89]:

UID	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_degree	age_median	pct_own	married	separated	divorced
267822	53	36	13346	1	5230	67994.14790	0.02077	0.08919	0.52963	0.89288	44.666665	0.79046	0.57851	0.01240	0.08770
246444	141	18	46616	1	2633	50670.10337	0.02222	0.04274	0.60855	0.90487	34.791665	0.52483	0.34886	0.01426	0.09030
245683	63	18	46122	1	6881	95262.51431	0.00000	0.09512	0.73484	0.94288	41.833330	0.85331	0.64745	0.01607	0.10657
279653	127	72	927	2	2700	56401.68133	0.01086	0.01086	0.52714	0.91500	49.750000	0.65037	0.47257	0.02021	0.10106
247218	161	20	66502	1	5637	54053.42396	0.05426	0.05426	0.51938	1.00000	22.000000	0.13046	0.12356	0.00000	0.03109

In [90]:
```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.

In [91]:
```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```
Out[91]:
```
LinearRegression()
```

In [92]:
```
y_pred=linereg.predict(x_test_scaled)
```

In [93]:
```
print("Overall R2 score of linear regression model", r2_score(y_test,y_pred))
print("Overall RMSE of linear regression model", np.sqrt(mean_squared_error(y_test,y_pred)))
```
```
Overall R2 score of linear regression model 0.7348210754610929
Overall RMSE of linear regression model 323.10188949846344
```

The Accuracy and R2 score are good, but still will investigate the model performance at state level

Run another model at State level. There are 52 states in USA.

In [94]:
```
state=df_train['STATEID'].unique()
state[0:5]
#Picking a few iDs 20,1,45,6
```
Out[94]:
```
array([36, 18, 72, 20,  1], dtype=int64)
```

In [95]:
```
for i in [20,1,45]:
    print("State ID-",i)

    x_train_nation=df_train[df_train['COUNTYID']==i][feature_cols]
    y_train_nation=df_train[df_train['COUNTYID']==i]['hc_mortgage_mean']

    x_test_nation=df_test[df_test['COUNTYID']==i][feature_cols]
    y_test_nation=df_test[df_test['COUNTYID']==i]['hc_mortgage_mean']
```

```
    x_train_scaled_nation=sc.fit_transform(x_train_nation)
    x_test_scaled_nation=sc.fit_transform(x_test_nation)

    linereg.fit(x_train_scaled_nation,y_train_nation)
    y_pred_nation=linereg.predict(x_test_scaled_nation)

    print("Overall R2 score of linear regression model for state,",i,":-" ,r2_score(y_test_nation,y_pred_nation))
    print("Overall RMSE of linear regression model for state,",i,":-" ,np.sqrt(mean_squared_error(y_test_nation,y_pred_nation)))
    print("\n")
```

```
State ID- 20
Overall R2 score of linear regression model for state, 20 :- 0.6046603766461807
Overall RMSE of linear regression model for state, 20 :- 307.9718899931473


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8104382475484617
Overall RMSE of linear regression model for state, 1 :- 307.8275861848434


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.7887446497855252
Overall RMSE of linear regression model for state, 45 :- 225.69615420724134
```

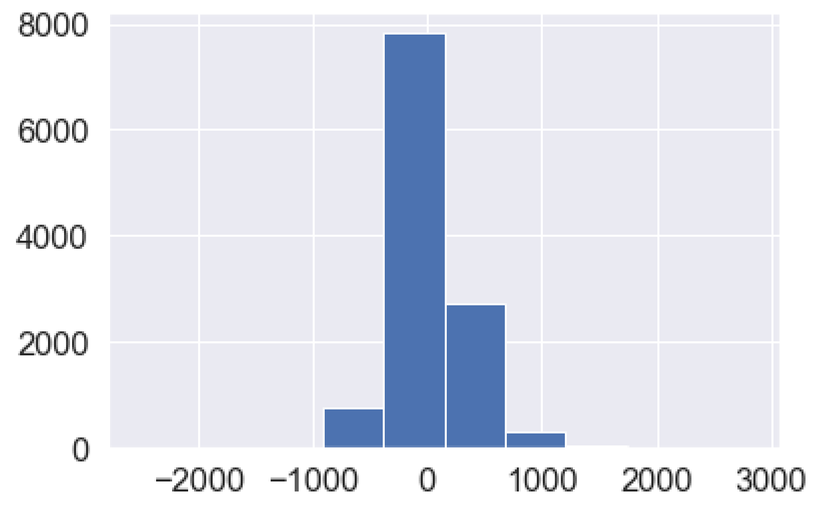
In [96]:
```
# To check the residuals
```

In [97]:
```
residuals=y_test-y_pred
residuals
```

Out[97]:
```
UID
255504    281.969088
252676    -69.935775
276314    190.761969
248614   -157.290627
286865     -9.887017
             ...
238088    -67.541646
242811    -41.578757
250127   -127.427569
241096   -330.820475
287763    217.760642
Name: hc_mortgage_mean, Length: 11709, dtype: float64
```

In [98]:
```
plt.hist(residuals) # Normal distribution of residuals
```

Out[98]:
```
(array([6.000e+00, 3.000e+00, 2.900e+01, 7.670e+02, 7.823e+03, 2.716e+03,
        3.010e+02, 4.900e+01, 1.200e+01, 3.000e+00]),
 array([-2515.04284233, -1982.92661329, -1450.81038425,  -918.69415521,
         -386.57792617,   145.53830287,   677.65453191,  1209.77076095,
         1741.88698999,  2274.00321903,  2806.11944807]),
 <BarContainer object of 10 artists>)
```



In [99]:
```
sns.distplot(residuals)
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[99]:
```
<AxesSubplot:xlabel='hc_mortgage_mean', ylabel='Density'>
```

In [100…

```
plt.scatter(residuals,y_pred) # Same variance and residuals does not have correlation with predictor
# Independance of residuals
```

Out[100]: <matplotlib.collections.PathCollection at 0x2dc4ea81880>

In [ ]: