Real Estate

Problem Statement:

A banking institution requires actionable insights from the perspective of Mortgage-Backed Securities, Geographic Business Investment and Real Estate Analysis.

The objective is to identify white spaces/potential business in the mortgage loan.

The mortgage bank would like to identify potential monthly mortgage expenses for each region based on monthly family income and rental of the real estate.

A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. This would help to monitor the key metrics and trends.

The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental income, monthly mortgage and owner's cost, family income vs mortgage cost comparison across different regions. The metrics are described not to limit the dashboard to these few only.

Dataset Description :-

Following are the themes the fields fall under Home Owner Costs: Sum of utilities, property taxes.

□1. Second Mortgage: Households with a second mortgage statistics.

□2. Home Equity Loan: Households with a Home equity Loan statistics.

□3. Debt: Households with any type of debt statistics.

□4. Mortgage Costs: Statistics regarding mortgage payments, home equity loans, utilities and property taxes

□5. Home Owner Costs: Sum of utilities, property taxes statistics

□6. Gross Rent: Contract rent plus the estimated average monthly cost of utility features

□7. Gross Rent as Percent of Income Gross rent as the percent of income very interesting

□8. High school Graduation: High school graduation statistics.

□9. Population Demographics: Population demographic statistics.

□10. Age Demographics: Age demographic statistics.

□11.Household Income: Total income of people residing in the household.

□12.Family Income: Total income of people related to the householder.

In [1]:

```
import time
import random
from math import *
import operator
import pandas as pd
import numpy as np

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

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

```
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The text.latex.preview rcparam was deprecated in Matplotlib 3.3 and will b
e removed two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The mathtext.fallback_to_cm rcparam was deprecated in Matplotlib 3.3 and w
ill be removed two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle: Support for setting the 'mathtext.fallback_to_cm'
rcParam is deprecated since 3.3 and will be removed two minor releases lat
er; use 'mathtext.fallback : 'cm' instead.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The validate_bool_maybe_none function was deprecated in Matplotlib 3.3 and
will be removed two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The savefig.jpeg_quality rcparam was deprecated in Matplotlib 3.3 and will
be removed two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The keymap.all_axes rcparam was deprecated in Matplotlib 3.3 and will be r
emoved two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The animation.avconv_path rcparam was deprecated in Matplotlib 3.3 and wil
l be removed two minor releases later.
In C:\Users\frees\Anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib
\_classic_test.mplstyle:
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and wil
l be removed two minor releases later.
C:\Users\frees\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: Futur
eWarning: pandas.util.testing is deprecated. Use the functions in the publ
ic API at pandas.testing instead.
  if sys.path[0] == '':
```

1. Import data

In [2]:

```
df_train=pd.read_csv("../Data/Input/train.csv")
```

In [3]:

```
df_test=pd.read_csv("../Data/Input/test.csv")
```

In [4]:

```
df_train.head()
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	pl
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hami
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Rosel
2	245683	NaN	140	63	18	Indiana	IN	Danville	Dan
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guayn
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manha (

5 rows × 80 columns

In [5]:

```
df_test.head()
```

Out[5]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	
0	255504	NaN	140	163	26	Michigan	MI	Detroit	D
1	252676	NaN	140	1	23	Maine	ME	Auburn	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	M
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	

5 rows × 80 columns

In [6]:

```
df_train.columns
```

Out[6]:

```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state',
       'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_c
ode',
       '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_std
ev',
       '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', 'd
ebt',
       '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_sample
s',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced'],
      dtype='object')
```

In [7]:

```
df_test.columns
```

Out[7]:

```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state',
       'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_c
ode',
       '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_std
ev',
       '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', 'd
ebt',
       '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_sample
s',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced'],
      dtype='object')
```

In [8]:

```
len(df_train)
```

Out[8]:

```
27321
```

In [9]:

```
len(df_test)
```

Out[9]:

```
11709
```

In [10]:

```
df_test.head()
```

Out[10]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	
0	255504	NaN	140	163	26	Michigan	MI	Detroit	D
1	252676	NaN	140	1	23	Maine	ME	Auburn	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	M
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	

5 rows × 80 columns

In [11]:

```
df_train.describe()
```

Out[11]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	a
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	273
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	5
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	2
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	2
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	4
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	6
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	8
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	9

8 rows × 74 columns

In [12]:

```
df_test.describe()
```

Out[12]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	a
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	1170
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	59
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	23
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	20
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	40
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	61
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	78
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	98

8 rows × 74 columns

In [13]:

```
df_train.info()
df_test.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
<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
```

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

In [14]:

```
#UID is unique userID value in the train and test dataset. So an index can be created f
rom 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 [15]:

```
df_train.head(2)
```

Out[15]:

	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	t
UID									
267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	
246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	

2 rows × 79 columns

In [16]:

```
df_test.head(2)
```

Out[16]:

	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	t
UID									
255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	(
252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	

2 rows × 79 columns

1. 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 [89]:

```
df_train.describe().T
```

Out[89]:

	count	mean	std	min	25%	50%	
COUNTYID	27321.0	85.646426	98.333097	1.000	29.000000	63.000000	10
STATEID	27321.0	28.271806	16.392846	1.000	13.000000	28.000000	4
type	27321.0	2.376890	1.692008	1.000	1.000000	1.000000	
zip_code	27321.0	50081.999524	29558.115660	602.000	26554.000000	47715.000000	7709
area_code	27321.0	596.507668	232.497482	201.000	405.000000	614.000000	80
...	...	...	...	...	...	...	
separated	27321.0	0.019089	0.020723	0.000	0.004600	0.013620	
divorced	27321.0	0.100248	0.048883	0.000	0.066080	0.095640	
bad_debt	27321.0	0.105099	0.070907	0.000	0.052790	0.100330	
pop_density	27321.0	0.002067	0.004597	0.000	0.000120	0.000851	
age_median	27321.0	39.214646	7.585480	13.375	34.166665	39.214646	4

75 rows × 8 columns

In [91]:

```
from pandas_profiling import ProfileReport
#Perform descriptive analytics on the given data
profile = ProfileReport(df_train, title='Realestate_Profiling_Report')
```

In [93]:

```
profile.to_file("Realestate_Profiling_Report.html")
```

In [19]:

```
#percantage of missing values in train set
d_miss_list_train = df_train.isnull().sum()*100 / len(df_train)
d_miss_values_df_train=pd.DataFrame(d_miss_list_train, columns=["% missing values"])
d_miss_values_df_train.sort_values(by=['% missing values'],inplace=True,ascending=False
)
d_miss_values_df_train[d_miss_values_df_train['% missing values'] > 0] [:10]
#BLOCKID can be dropped, since it is 100%missing values
```

Out[19]:

	% 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

In [20]:

```
#percantage of missing values in train set
d_miss_list_test = df_test.isnull().sum()*100 / len(df_test)
d_miss_values_df_test=pd.DataFrame(d_miss_list_test, columns=["% missing values"])
d_miss_values_df_test.sort_values(by=['% missing values'],inplace=True,ascending=False)
d_miss_values_df_test[d_miss_values_df_test['% missing values'] > 0] [:10]
#BLOCKID can be dropped, since it is 100%missing values
```

Out[20]:

	% missing values
BLOCKID	100.000000
hc_samples	2.476727
hc_mean	2.476727
hc_median	2.476727
hc_stdev	2.476727
hc_sample_weight	2.476727
hc_mortgage_mean	2.288838
hc_mortgage_stdev	2.288838
hc_mortgage_sample_weight	2.288838
hc_mortgage_samples	2.288838

In [21]:

```
#SUMLEVEL doest not have any predictive power and no variance
df_train .drop(columns=['BLOCKID','SUMLEVEL'],inplace=True)
```

In [22]:

```
#SUMLEVEL doest not have any predictive power
df_test .drop(columns=['BLOCKID','SUMLEVEL'],inplace=True)
```

In [23]:

```
# Input  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_sam
ples', 'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_3
0', '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_mea
n', 'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weigh
t', 'hc_mortgage_samples', 'hc_mean', 'hc_median', 'hc_stdev', 'hc_sample
s', 'hc_sample_weight', 'home_equity_second_mortgage', 'second_mortgage',
'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cdf', 'debt_cd
f', 'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mean', 'm
ale_age_median', 'male_age_stdev', 'male_age_sample_weight', 'male_age_sam
ples', '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]:

```
# Input  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_sam
ples', 'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_3
0', '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_mea
n', 'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weigh
t', 'hc_mortgage_samples', 'hc_mean', 'hc_median', 'hc_stdev', 'hc_sample
s', 'hc_sample_weight', 'home_equity_second_mortgage', 'second_mortgage',
'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cdf', 'debt_cd
f', 'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mean', 'm
ale_age_median', 'male_age_stdev', 'male_age_sample_weight', 'male_age_sam
ples', 'female_age_mean', 'female_age_median', 'female_age_stdev', 'female
_age_sample_weight', 'female_age_samples', 'pct_own', 'married', 'married_
snp', 'separated', 'divorced']
```

In [25]:

```
  # 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 [26]:

```python
# 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 [27]:

```python
df_train.isna().sum().sum()
```

Out[27]:

```
0
```

In [28]:

```python
df_test.isna().sum().sum()
```

Out[28]:

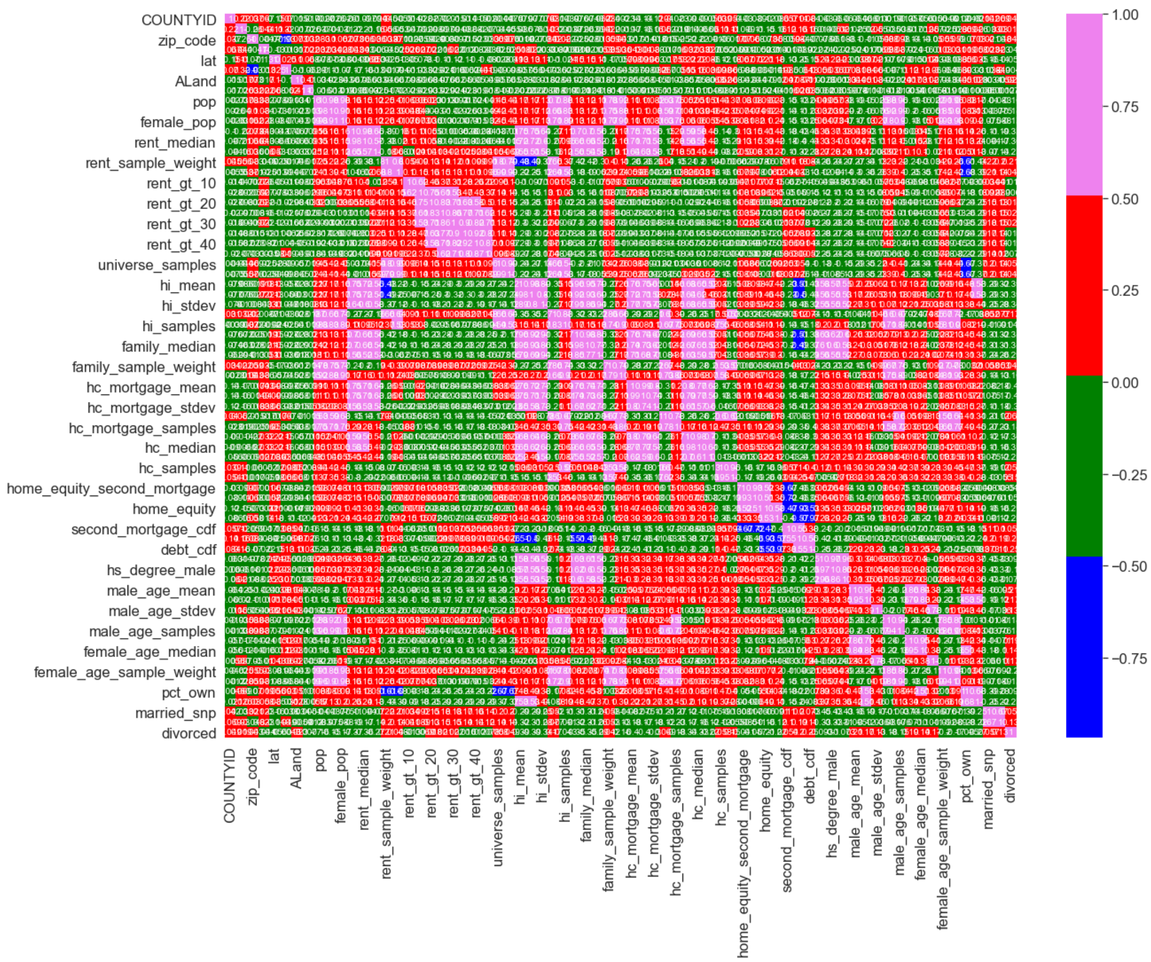
```
0
```

In [29]:

```python
plt.subplots(figsize=(20,15))
sns.heatmap(df_train.corr(), annot=True, cmap = ['blue', 'green', 'red', 'violet'] )
```

Out[29]:

```
<AxesSubplot:>
```



In [ ]:

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 [30]:

```
# Best would be to use Pandasql library for this, it would make life easier
# Similarly using Plotly along with Matplotlib would help plat data well
#!pip install pandasql
```

In [31]:

```
from pandasql import sqldf
q1 = "select place,pct_own,second_mortgage,lat,lng from df_train where pct_own >0.10 an
d 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 [32]:

```
df_train_location_mort_pct.head()
```

Out[32]:

	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 [33]:

```
!pip install jupyter-dash
import plotly.express as px
import plotly.graph_objects as go
```

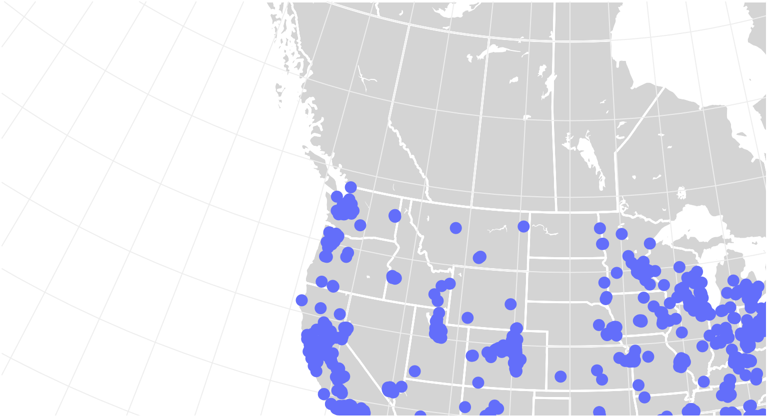
```
Requirement already satisfied: jupyter-dash in c:\users\frees\anaconda3\li
b\site-packages (0.4.0)
Requirement already satisfied: requests in c:\users\frees\anaconda3\lib\si
te-packages (from jupyter-dash) (2.25.1)
Requirement already satisfied: flask in c:\users\frees\anaconda3\lib\site-
packages (from jupyter-dash) (1.1.1)
Requirement already satisfied: ipython in c:\users\frees\anaconda3\lib\sit
e-packages (from jupyter-dash) (7.8.0)
Requirement already satisfied: dash in c:\users\frees\anaconda3\lib\site-p
ackages (from jupyter-dash) (1.19.0)
Requirement already satisfied: retrying in c:\users\frees\anaconda3\lib\si
te-packages (from jupyter-dash) (1.3.3)
Requirement already satisfied: ipykernel in c:\users\frees\anaconda3\lib\s
ite-packages (from jupyter-dash) (5.1.2)
Requirement already satisfied: ansi2html in c:\users\frees\anaconda3\lib\s
ite-packages (from jupyter-dash) (1.6.0)
Requirement already satisfied: dash-core-components==1.15.0 in c:\users\fr
ees\anaconda3\lib\site-packages (from dash->jupyter-dash) (1.15.0)
Requirement already satisfied: dash-renderer==1.9.0 in c:\users\frees\anac
onda3\lib\site-packages (from dash->jupyter-dash) (1.9.0)
Requirement already satisfied: future in c:\users\frees\anaconda3\lib\site
-packages (from dash->jupyter-dash) (0.17.1)
Requirement already satisfied: dash-table==4.11.2 in c:\users\frees\anacon
da3\lib\site-packages (from dash->jupyter-dash) (4.11.2)
Requirement already satisfied: flask-compress in c:\users\frees\anaconda3
\lib\site-packages (from dash->jupyter-dash) (1.8.0)
Requirement already satisfied: plotly in c:\users\frees\anaconda3\lib\site
-packages (from dash->jupyter-dash) (4.14.3)
Requirement already satisfied: dash-html-components==1.1.2 in c:\users\fre
es\anaconda3\lib\site-packages (from dash->jupyter-dash) (1.1.2)
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\frees\anaconda3
\lib\site-packages (from flask->jupyter-dash) (2.11.2)
Requirement already satisfied: click>=5.1 in c:\users\frees\anaconda3\lib
\site-packages (from flask->jupyter-dash) (7.0)
Requirement already satisfied: Werkzeug>=0.15 in c:\users\frees\anaconda3
\lib\site-packages (from flask->jupyter-dash) (0.16.0)
Requirement already satisfied: itsdangerous>=0.24 in c:\users\frees\anacon
da3\lib\site-packages (from flask->jupyter-dash) (1.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\frees\anaconda
3\lib\site-packages (from Jinja2>=2.10.1->flask->jupyter-dash) (1.1.1)
Requirement already satisfied: brotli in c:\users\frees\anaconda3\lib\site
-packages (from flask-compress->dash->jupyter-dash) (1.0.9)
Requirement already satisfied: tornado>=4.2 in c:\users\frees\anaconda3\li
b\site-packages (from ipykernel->jupyter-dash) (6.0.3)
Requirement already satisfied: traitlets>=4.1.0 in c:\users\frees\anaconda
3\lib\site-packages (from ipykernel->jupyter-dash) (4.3.2)
Requirement already satisfied: jupyter-client in c:\users\frees\anaconda3
\lib\site-packages (from ipykernel->jupyter-dash) (6.1.6)
Requirement already satisfied: backcall in c:\users\frees\anaconda3\lib\si
te-packages (from ipython->jupyter-dash) (0.1.0)
Requirement already satisfied: jedi>=0.10 in c:\users\frees\anaconda3\lib
\site-packages (from ipython->jupyter-dash) (0.15.1)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in c:\users\fr
ees\anaconda3\lib\site-packages (from ipython->jupyter-dash) (2.0.9)
Requirement already satisfied: setuptools>=18.5 in c:\users\frees\anaconda
3\lib\site-packages (from ipython->jupyter-dash) (41.2.0)
Requirement already satisfied: pygments in c:\users\frees\anaconda3\lib\si
te-packages (from ipython->jupyter-dash) (2.4.2)
Requirement already satisfied: decorator in c:\users\frees\anaconda3\lib\s
ite-packages (from ipython->jupyter-dash) (4.4.0)
Requirement already satisfied: colorama in c:\users\frees\anaconda3\lib\si
```

```
te-packages (from ipython->jupyter-dash) (0.4.1)
Requirement already satisfied: pickleshare in c:\users\frees\anaconda3\lib
\site-packages (from ipython->jupyter-dash) (0.7.5)
Requirement already satisfied: parso>=0.5.0 in c:\users\frees\anaconda3\li
b\site-packages (from jedi>=0.10->ipython->jupyter-dash) (0.5.1)
Requirement already satisfied: wcwidth in c:\users\frees\anaconda3\lib\sit
e-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython->jupyter-dash) (0.
1.7)
Requirement already satisfied: six>=1.9.0 in c:\users\frees\appdata\roamin
g\python\python37\site-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipytho
n->jupyter-dash) (1.13.0)
Requirement already satisfied: ipython-genutils in c:\users\frees\anaconda
3\lib\site-packages (from traitlets>=4.1.0->ipykernel->jupyter-dash) (0.2.
0)
Requirement already satisfied: jupyter-core>=4.6.0 in c:\users\frees\anaco
nda3\lib\site-packages (from jupyter-client->ipykernel->jupyter-dash) (4.
6.3)
Requirement already satisfied: pyzmq>=13 in c:\users\frees\anaconda3\lib\s
ite-packages (from jupyter-client->ipykernel->jupyter-dash) (18.1.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\frees\appd
ata\roaming\python\python37\site-packages (from jupyter-client->ipykernel-
>jupyter-dash) (2.8.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\frees\anaconda3\li
b\site-packages (from jupyter-core>=4.6.0->jupyter-client->ipykernel->jupy
ter-dash) (223)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\frees\app
data\roaming\python\python37\site-packages (from requests->jupyter-dash)
(1.25.7)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\frees\anacond
a3\lib\site-packages (from requests->jupyter-dash) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\frees\appdat
a\roaming\python\python37\site-packages (from requests->jupyter-dash) (201
9.11.28)
Requirement already satisfied: idna<3,>=2.5 in c:\users\frees\anaconda3\li
b\site-packages (from requests->jupyter-dash) (2.8)
```

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 ownershi
p is above 10 percent')
fig.show()
```

Top 2,500 locations with second mortgage is the highest and



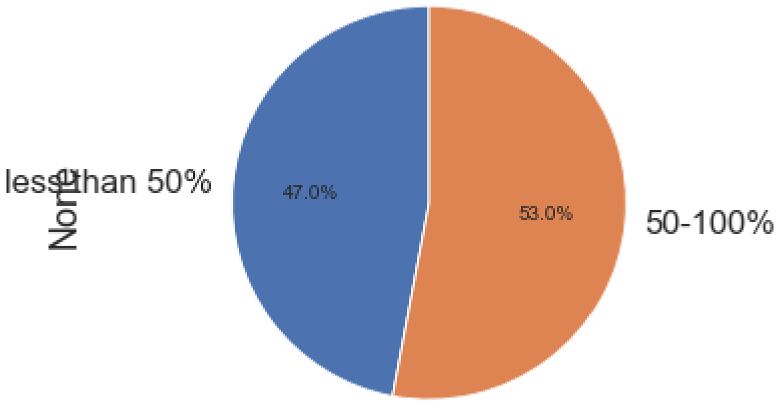
Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage c) Create pie charts to show overall debt and bad debt

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()
```



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', 'typ
e',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWate
r',
       '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_3
5',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_sample
s',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weig
ht',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samp
les',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weigh
t',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'd
ebt',
       '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_sample
s',
       '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)
```

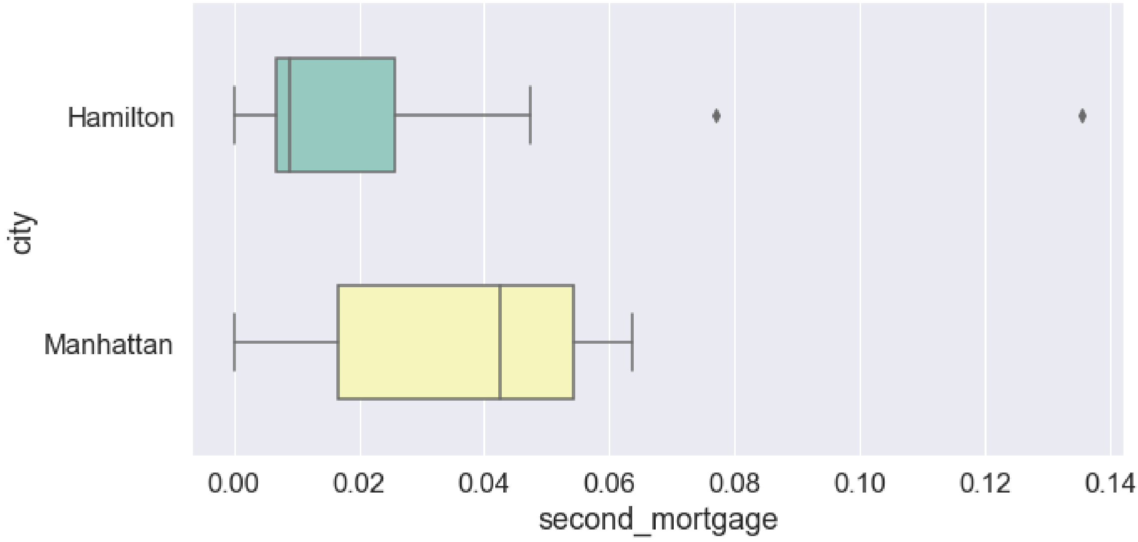
Out[38]:

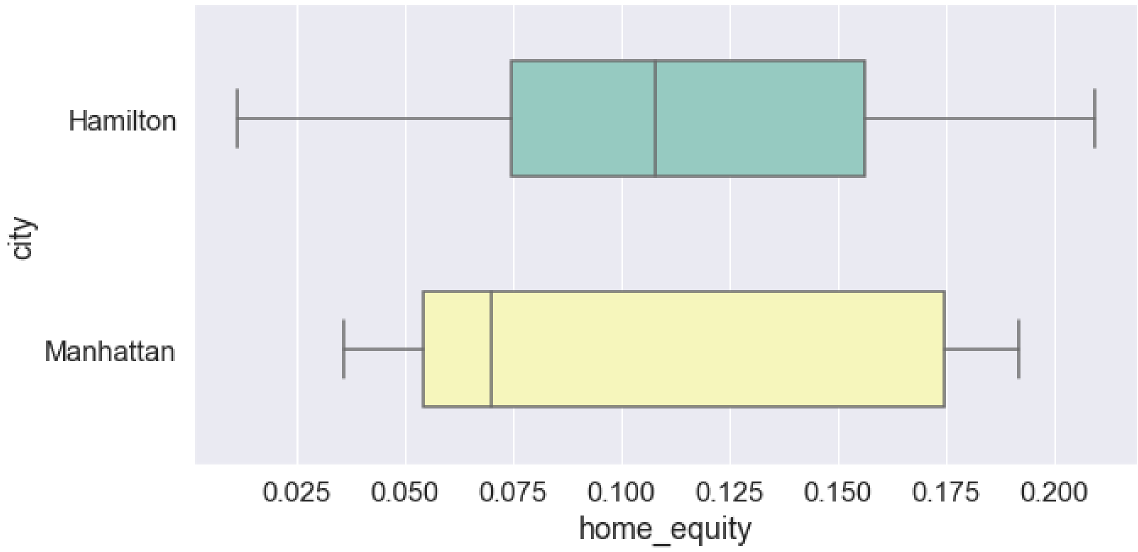
	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_c
UID									
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	1:
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	ε
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	45
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	39

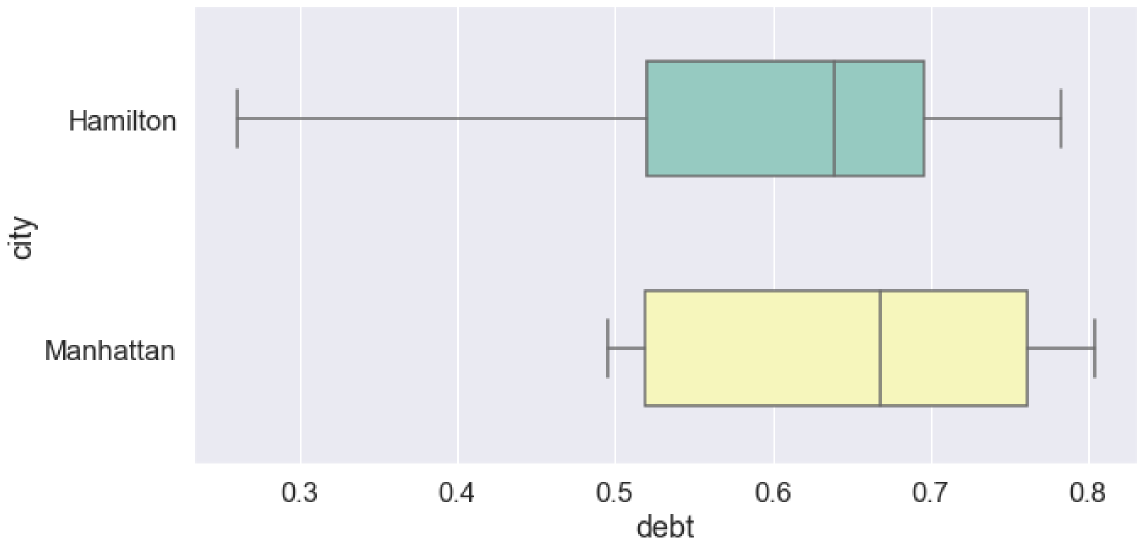
4 rows × 79 columns

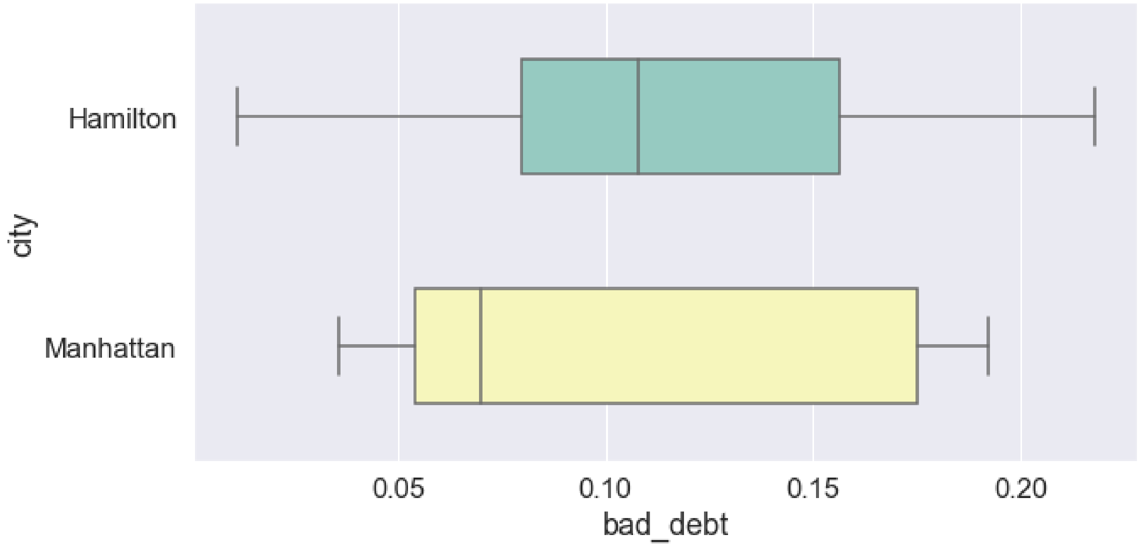
In [39]:

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

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

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

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


Hamilton
Manhattan
city
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
second_mortgage


Hamilton
Manhattan
city
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
home_equity


Hamilton
Manhattan
city
0.3
0.4
0.5
0.6
0.7
0.8
debt

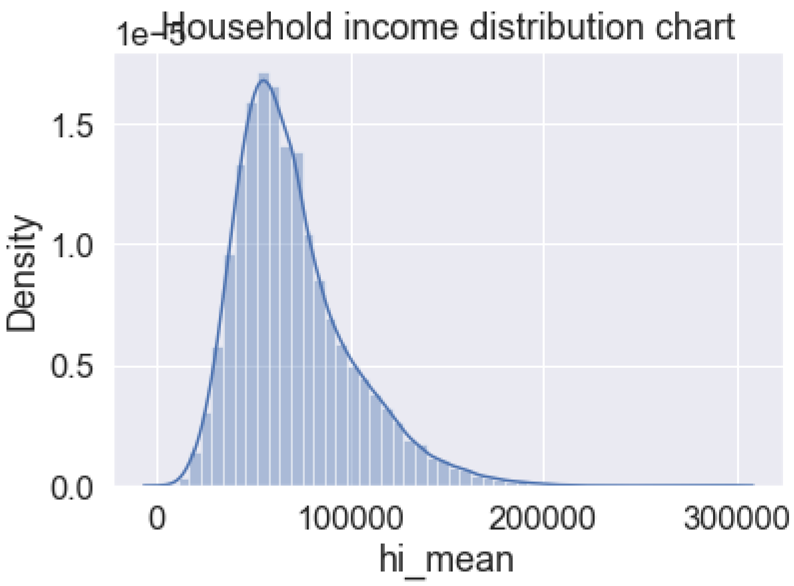


Create a collated income distribution chart for family income, house hold income, and remaining income

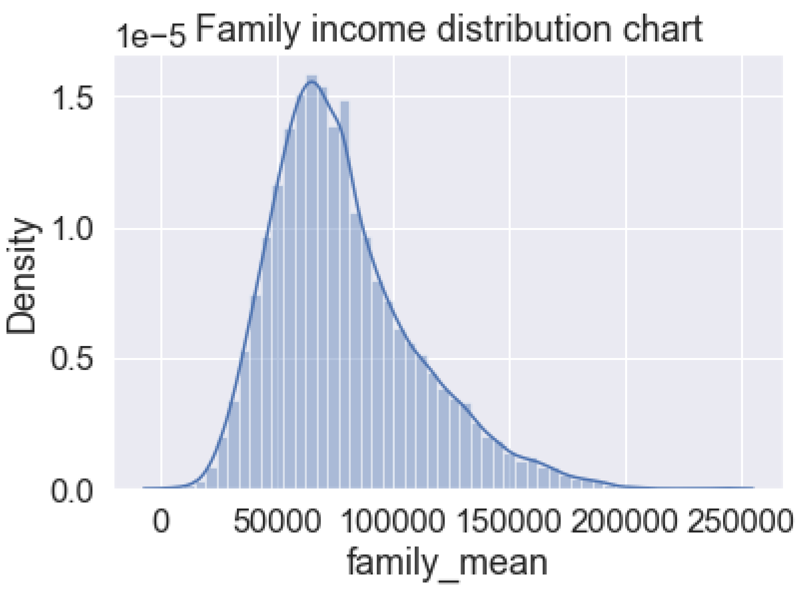
In [40]:

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

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

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

#As can be seen below the income distribution has a good normal distribution, without m
uch SKEW
```

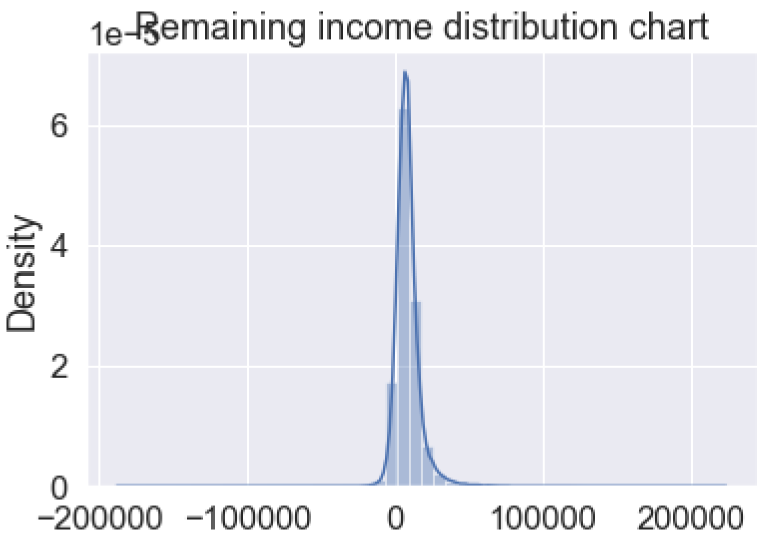
```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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




```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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



```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

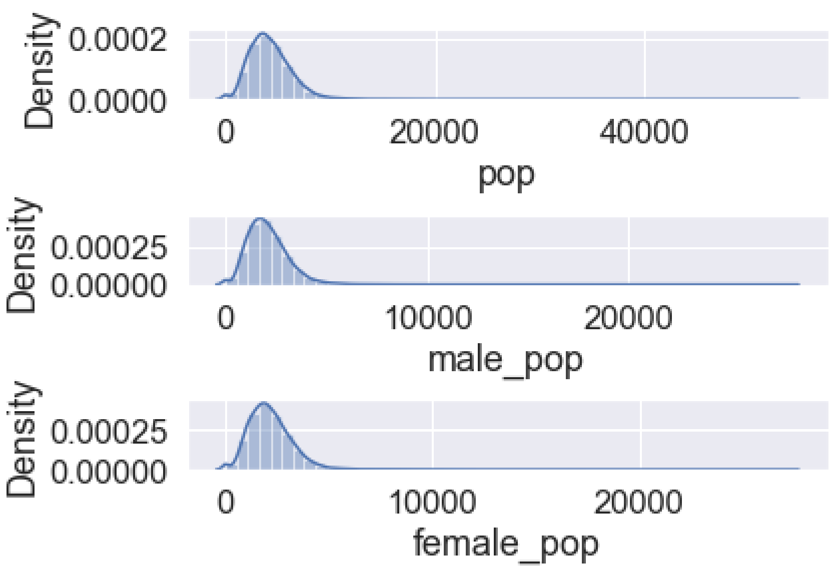


Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):¶

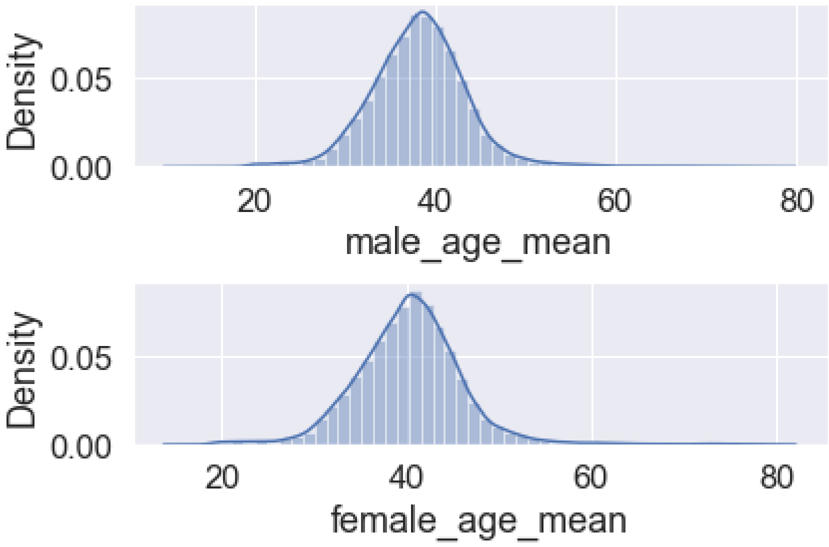
In [41]:

```
#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()

#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()

#The population data based on the plot shows that for male and female the distribution
 is very similar and closely identical
```

```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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




```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

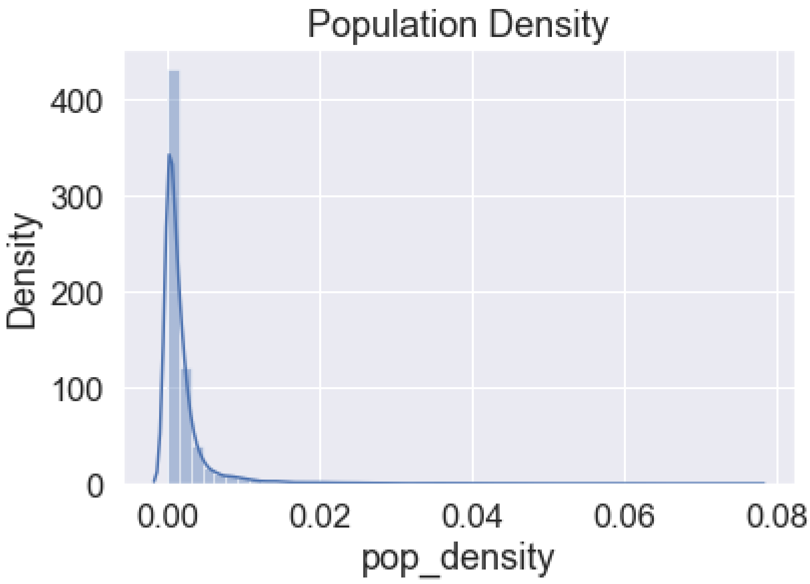


a) Use pop and ALand variables to create a new field called population density

In [42]:

```
df_train['pop_density']=df_train['pop']/df_train['ALand']

df_test['pop_density']=df_test['pop']/df_test['ALand']

sns.distplot(df_train['pop_density'])
plt.title('Population Density')
plt.show()

# Based on the plot very less density is noticed
```

```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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



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 [43]:

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

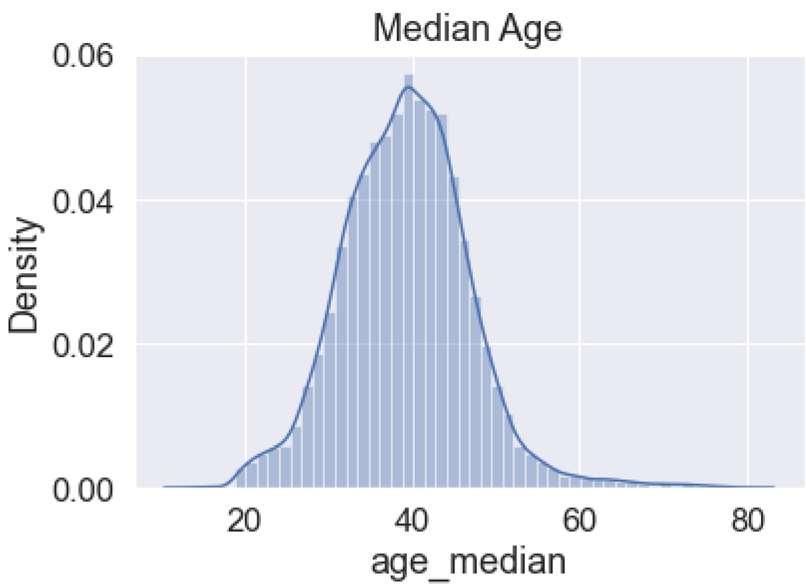
Out[43]:

	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 [44]:

```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
# Age of population
# 1 Sigma deviation of age is between 25 & 45 and the overall age range is mostly betwe
en 20 and 60
# The peak being at 40 signifies that the majority of people are at 40 years
# The distribution seems perfectly Normal (while not) and the Median age has a seemingl
y gaussian distribution
# Although some right skewness is noticed ths is very mild and shows that there are not
too many
# retired / Sr. Citizens
# In fact it is also evident from graph that the slope to the Right of the distribution
is steeper
# meaning that the prime density of age group of people is  in the middle ages of 25 to
40
```

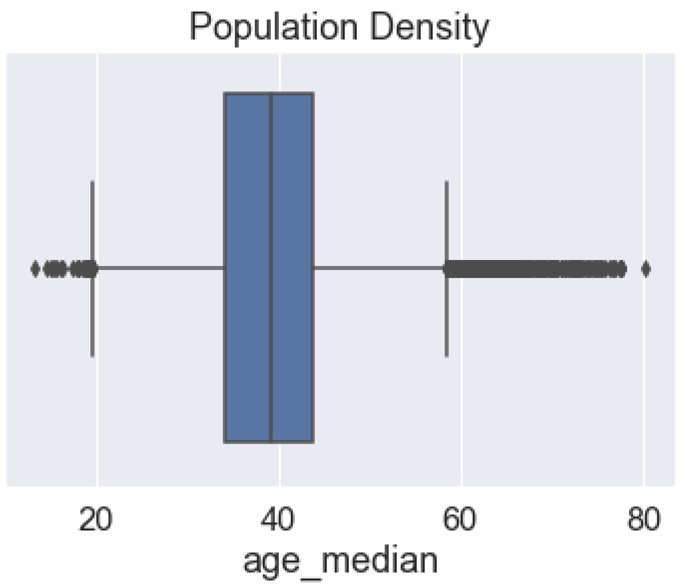
```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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



In [45]:

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

```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: Futu
reWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments w
ithout 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 [46]:

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

Out[46]:

```
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 [47]:

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

In [48]:

```
df_train[['pop','pop_bins']]
```

Out[48]:

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 [49]:

```
df_train['pop_bins'].value_counts()
```

Out[49]:

```
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 [50]:

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

Out[50]:

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

In [51]:

```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "media
n"])
```

Out[51]:

	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

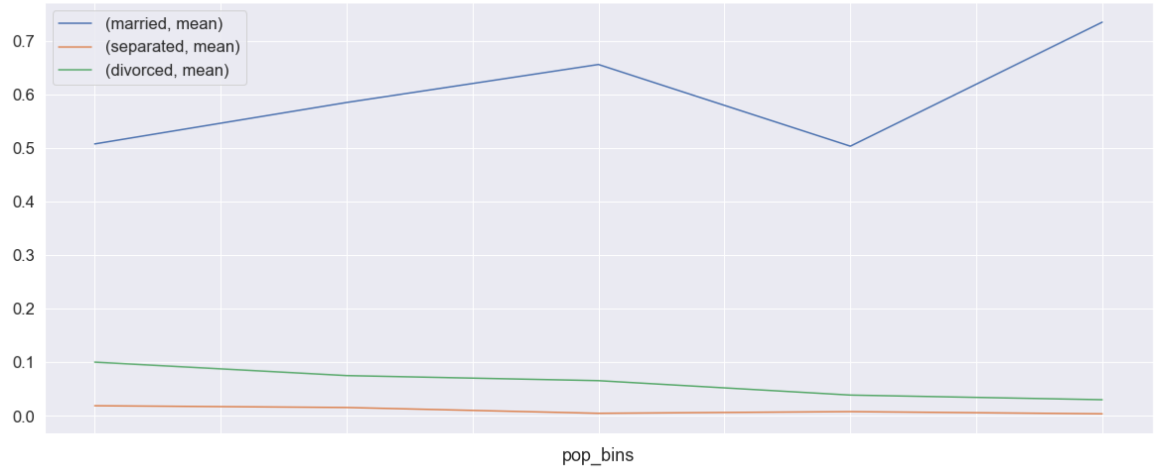
1. Very high population group has more married people & less percantage of separated and divorced couples
2. Very low population groups has increased divorce people

Visualize using appropriate chart type

In [52]:

```
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()
```

```
C:\Users\frees\AppData\Roaming\Python\Python37\site-packages\pandas\plotti
ng\_matplotlib\core.py:1192: UserWarning:

FixedFormatter should only be used together with FixedLocator


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

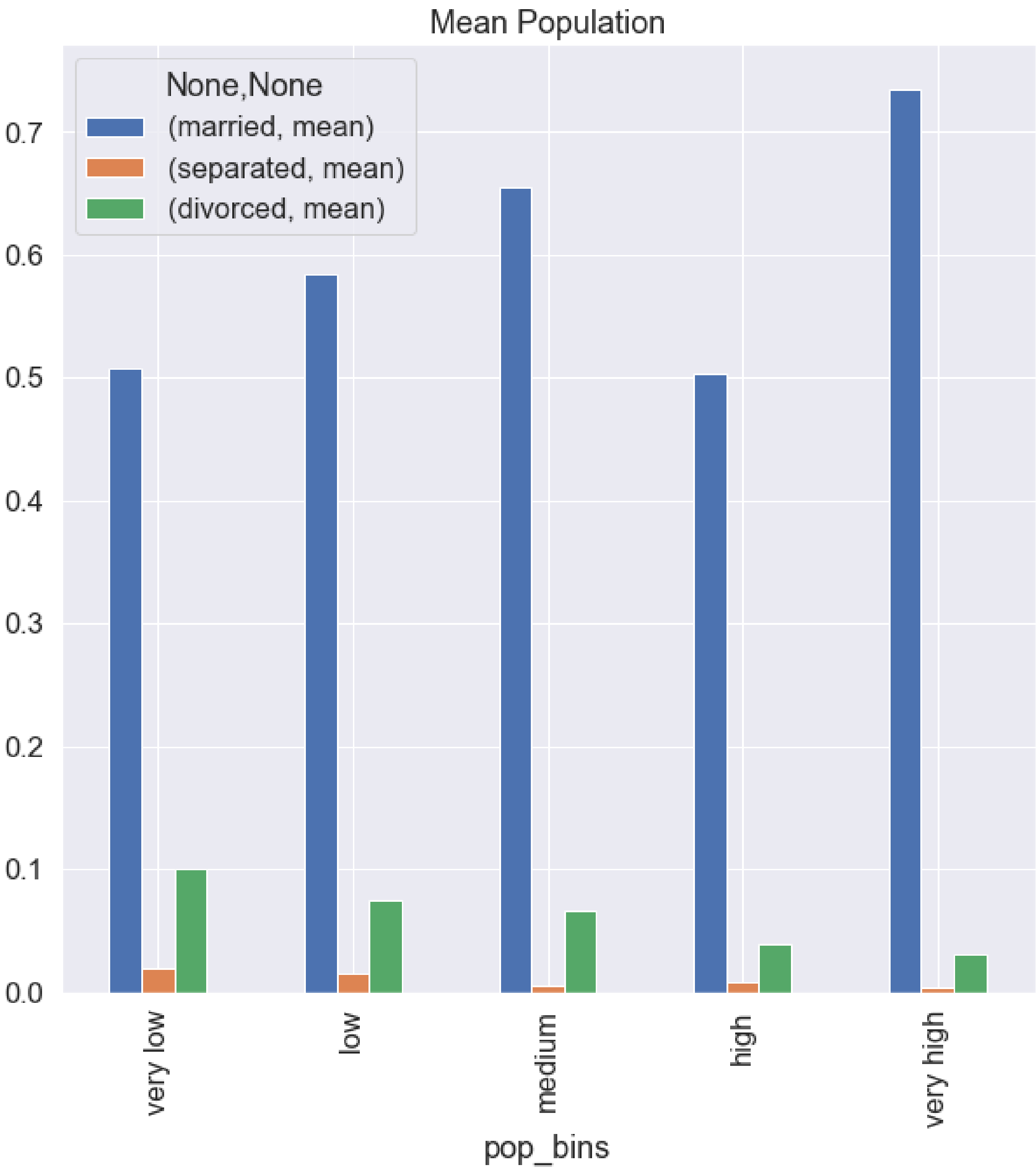


In [53]:

```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean"]).plot(
kind = 'bar', figsize = (10, 10), title = 'Mean Population')
```

Out[53]:

```
<AxesSubplot:title={'center':'Mean Population'}, xlabel='pop_bins'>
```


Mean Population
None,None
(married, mean)
(separated, mean)
(divorced, mean)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
very low
low
medium
high
very high
pop_bins

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

In [54]:

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

Out[54]:

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

In [55]:

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

Out[55]:

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

In [56]:

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

Out[56]:

```
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 [57]:

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

Out[57]:

```
0.013358170721473864
```

In [58]:

```
df_train.columns
```

Out[58]:

```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'typ
e',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWate
r',
       '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_3
5',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_sample
s',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weig
ht',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samp
les',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weigh
t',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'd
ebt',
       '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_sample
s',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'bad_debt', 'bins', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

In [59]:

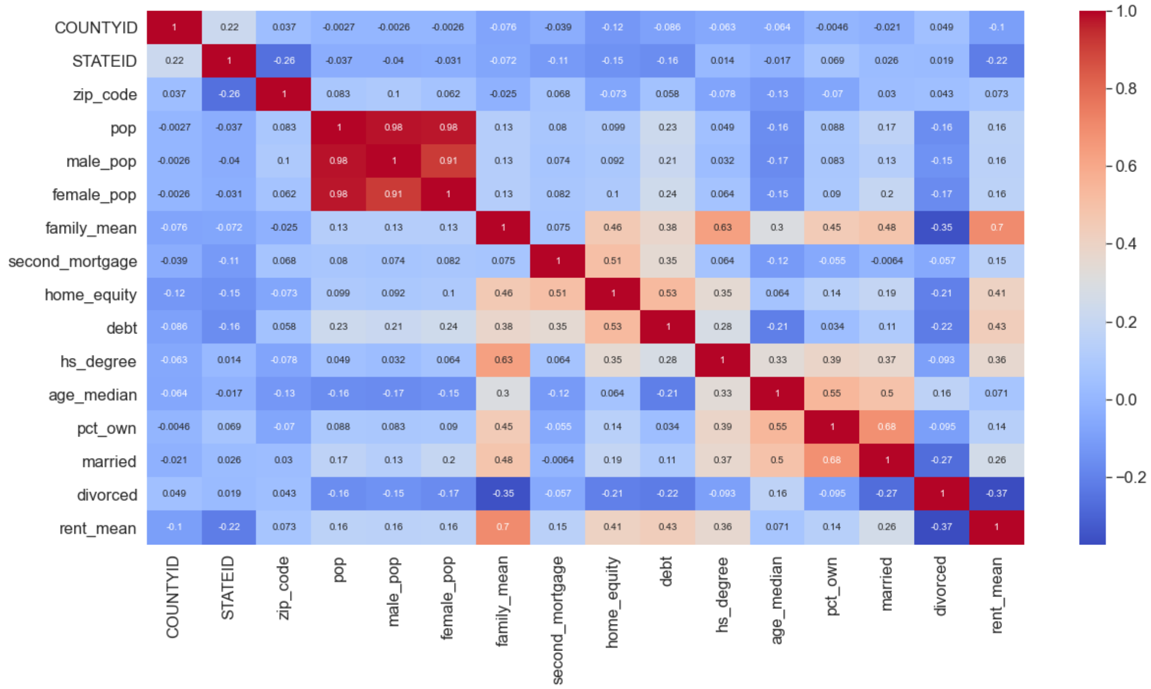
```
cor=df_train[['COUNTYID','STATEID','zip_code','type','pop', 'male_pop', 'female_pop','f
amily_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','divorced','rent_mean', ]].corr()
```

In [60]:

```
cor.to_excel("output.xlsx",
             sheet_name='Sheet_name_1')
```

In [61]:

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



In [ ]:

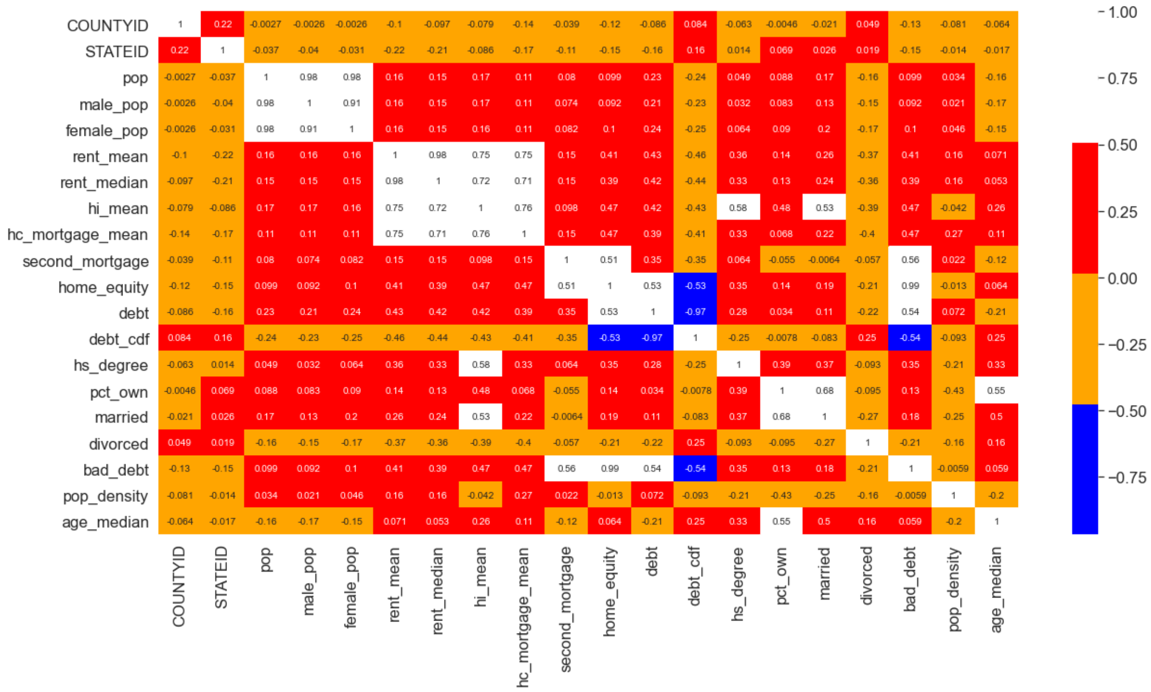
```

```

In [62]:

```
cor=df_train[['COUNTYID', 'STATEID', 'state', 'city',  'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'hi_mean', 'hc_mortgage_mean', 'second_mortgage', 'home_equity', 'debt',
       'debt_cdf', 'hs_degree', 'pct_own', 'married',  'divorced',
       'bad_debt', 'bins', 'pop_density', 'age_median', 'pop_bins']].corr()
```

In [63]:

```
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True,cmap= ['blue', 'orange', 'red', 'white'])
plt.show()
```



In [ ]:

1. Very High positive correaltion is noticed between pop, male_pop and female_pop - but this is not a valid inference
2. High positive correaltion is noticed between rent_mean,hi_mean, family_mean,hc_mean - This is not a valid inference as its all related The key is that there is good correlation btween married & pct_owners; home equity owners and debt etc

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:

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

In [64]:

```
#!pip install factor_analyzer
```

In [65]:

```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [66]:

```
fa=FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

Out[66]:

```
array([[-1.12589167e-01,  1.95646468e-02, -2.39331082e-02,
        -6.27632620e-02,  4.23474743e-02],
       [-1.10186764e-01,  1.33506222e-02,  2.79651231e-02,
        -1.49825870e-01,  1.10838810e-01],
       [-8.28678631e-02,  5.16372374e-02, -1.36451870e-01,
        -4.98918623e-02, -1.04024838e-01],
       [ 1.80961150e-02,  1.92013752e-02,  5.81329824e-03,
         2.64842736e-02, -6.12442447e-03],
       [ 9.02324724e-02, -9.72544298e-02, -6.54601333e-02,
        -1.33145902e-01, -1.48594600e-01],
       [-1.07335688e-02, -4.12376818e-02,  1.45853490e-01,
         8.80433420e-03,  1.08227570e-01],
       [-4.28796978e-02, -2.09780215e-02,  3.66726845e-02,
        -9.45597403e-02,  5.91380521e-02],
       [-2.44243063e-03, -1.53245406e-02, -2.68300832e-03,
        -4.52473018e-02,  2.37240641e-02],
       [ 7.92164326e-02,  9.57453314e-01, -8.71151642e-02,
        -6.59923918e-03, -3.97273184e-02],
       [ 7.39808199e-02,  9.18750509e-01, -1.08834840e-01,
        -2.79371588e-02, -3.93153639e-02],
       [ 8.06598896e-02,  9.47839225e-01, -6.08006514e-02,
         1.53627099e-02, -3.86977270e-02],
       [ 7.70052134e-01,  9.84675308e-03, -3.71249747e-02,
         1.14949043e-01, -1.23784679e-01],
       [ 7.18615885e-01,  6.24980423e-03, -4.59787405e-02,
         1.09109689e-01, -1.35301905e-01],
       [ 7.07647245e-01,  2.46625389e-02, -1.00860843e-02,
         1.04472487e-01,  7.72381260e-02],
       [-1.34545490e-01,  3.36809300e-01, -4.87894968e-01,
        -4.15446203e-02,  3.17608540e-01],
       [ 2.31079708e-01,  4.37729791e-01, -6.40209210e-01,
        -2.52310964e-02,  3.47216232e-01],
       [-4.52068132e-02,  3.51263845e-02,  3.07537067e-02,
         4.44793513e-01, -1.63273413e-01],
       [-2.50717052e-02,  1.70166797e-02,  4.57227249e-02,
         6.76083891e-01, -1.55256763e-01],
       [-3.90694437e-02, -1.67460863e-02,  8.13962615e-02,
         8.36389085e-01, -9.18259771e-02],
       [-5.14161942e-02, -3.57207133e-02,  1.10795179e-01,
         9.25123759e-01, -4.44866514e-02],
       [-6.08589975e-02, -4.41860613e-02,  1.35794037e-01,
         9.53019942e-01, -2.21548688e-02],
       [-4.57771158e-02, -5.25526113e-02,  1.41019872e-01,
         9.32702623e-01, -5.86157419e-07],
       [-4.19486066e-02, -5.90387619e-02,  1.28851761e-01,
         8.87316629e-01,  1.05894303e-02],
       [-2.47894655e-02, -7.29670537e-02,  9.41510352e-02,
         7.79023641e-01,  2.95352807e-02],
       [ 2.12258449e-01,  4.65992338e-01, -6.14495941e-01,
        -2.47659966e-02,  3.66644525e-01],
       [ 2.33057242e-01,  4.47057844e-01, -6.28263419e-01,
        -2.71547670e-02,  3.43419616e-01],
       [ 7.85157088e-01,  4.91249251e-02,  1.44540482e-01,
        -2.05217628e-01, -1.54523357e-01],
       [ 7.10324887e-01,  4.99730438e-02,  1.32239988e-01,
        -2.19171867e-01, -2.10505569e-01],
       [ 8.61780945e-01,  4.35044821e-02,  1.65839097e-01,
        -1.19850815e-01,  3.16733600e-02],
       [-2.23443277e-01,  8.46259552e-01, -4.61177177e-02,
```

```
       6.88599284e-02,  2.27742314e-01],
      [ 1.43837555e-01,  9.53197417e-01,  2.27887457e-02,
       -4.57890445e-02,  1.00796447e-01],
      [ 8.30286485e-01,  3.42025995e-02,  1.61105999e-01,
       -2.04570326e-01, -7.48710465e-02],
      [ 7.94476578e-01,  2.83818590e-02,  1.51219546e-01,
       -2.07681495e-01, -9.12497069e-02],
      [ 8.11481639e-01,  4.32314872e-02,  1.43645560e-01,
       -1.07778261e-01,  5.79540095e-02],
      [-3.37741911e-01,  8.64927626e-01,  3.58933715e-02,
        9.07183983e-02,  4.46327232e-02],
      [ 5.03572659e-02,  9.35515340e-01,  1.51475399e-01,
       -2.51501272e-02, -9.34471612e-02],
      [ 9.78242248e-01, -3.31490300e-02, -1.05261172e-01,
        4.50364262e-02,  7.37362101e-02],
      [ 9.59137175e-01, -3.90023011e-02, -1.20630333e-01,
        4.52591414e-02,  6.64877172e-02],
      [ 8.14087200e-01,  2.23057212e-03,  7.66518546e-02,
        2.02747481e-02,  1.27634839e-01],
      [-4.15353985e-01,  7.18339583e-01,  3.40068063e-01,
       -7.18402786e-02, -2.77950514e-01],
      [ 7.64912662e-02,  7.24900629e-01,  2.74193203e-01,
       -4.83952656e-02, -3.52988281e-01],
      [ 9.10390845e-01, -5.36541224e-02, -4.68641832e-02,
       -7.64182799e-04,  1.63870450e-01],
      [ 8.73011847e-01, -5.30302305e-02, -5.89943080e-02,
       -1.58989796e-03,  1.52417530e-01],
      [ 7.55087658e-01, -3.56133951e-03,  5.39542589e-02,
        4.24181467e-03,  2.58043472e-01],
      [-1.23469887e-01,  6.07438129e-01,  6.33039226e-01,
       -2.14798836e-02,  2.47973915e-01],
      [-3.42866890e-01,  5.59526271e-01,  5.88212996e-01,
       -2.51533561e-02,  2.18419876e-01],
      [-1.60867221e-01, -1.53062610e-02, -1.57026585e-01,
        1.09243760e-01, -6.61660836e-01],
      [-1.37306768e-01, -2.17250646e-02, -1.58408932e-01,
        1.25156198e-01, -6.71630814e-01],
      [ 2.45096178e-01, -2.54584609e-02, -2.66691442e-02,
        9.53148560e-02, -6.42510860e-01],
      [ 2.03988664e-01,  7.85172850e-02, -3.01656223e-01,
        2.28379473e-02, -6.29223340e-01],
      [ 1.08926092e-01, -6.34332397e-02, -3.36565212e-02,
       -9.49480516e-02,  6.81473857e-01],
      [-2.63787620e-01, -6.43281032e-03, -3.58792167e-02,
       -9.37962507e-02,  6.47817015e-01],
      [-2.15717050e-01, -7.36588969e-02,  3.50113235e-01,
       -1.95201616e-02,  6.36783756e-01],
      [ 3.94306147e-01,  6.09565684e-02,  2.55337862e-01,
       -2.20362099e-01, -1.84248078e-01],
      [ 4.07877888e-01,  6.27256514e-02,  2.23926906e-01,
       -2.10028736e-01, -1.71989220e-01],
      [ 3.53156877e-01,  5.36715658e-02,  2.69603567e-01,
       -2.16933221e-01, -1.80072068e-01],
      [ 2.33537262e-01, -4.91732970e-02,  8.14450800e-01,
        9.36688962e-02,  3.27131930e-01],
      [ 2.40298203e-01, -3.38140121e-02,  8.31496969e-01,
        7.52417556e-02,  2.46323596e-01],
      [-6.71839509e-02,  6.58504532e-02,  5.86207682e-01,
        8.72955202e-02,  9.12541305e-02],
      [ 5.59835555e-02,  8.17918706e-01, -1.78458352e-01,
       -1.55949442e-02, -3.34299720e-02],
```

```
       [ 7.16426409e-02,  9.23428558e-01, -1.07142698e-01,
        -2.78635385e-02, -4.35991107e-02],
       [ 1.92496946e-01, -4.75870404e-02,  8.03173209e-01,
         1.43492718e-01,  3.33862153e-01],
       [ 1.87644431e-01, -3.29941019e-02,  8.58024507e-01,
         1.31329962e-01,  2.55679722e-01],
       [-1.02263659e-01,  6.03984260e-02,  4.72982259e-01,
         7.36848397e-02,  1.12273904e-01],
       [ 6.14776657e-02,  8.77962762e-01, -1.50410290e-01,
         2.20991041e-02, -4.17158165e-02],
       [ 7.83728215e-02,  9.54508789e-01, -5.91095915e-02,
         1.64800933e-02, -4.32590993e-02],
       [-3.24381775e-02,  1.11167163e-01,  7.84467373e-01,
        -4.37718673e-02, -2.80931215e-01],
       [ 1.76682390e-01,  1.90494236e-01,  5.61405479e-01,
        -1.20746168e-01, -1.32570784e-01],
       [-6.37386635e-02, -7.03047924e-02, -2.68934064e-01,
         1.28589795e-01,  1.88507857e-01],
       [-1.56051274e-01, -7.08033933e-02, -1.45964496e-01,
         1.24253733e-01,  1.46293106e-01],
       [-3.56716303e-01, -5.29910749e-02,  1.47771611e-01,
         2.87196217e-02,  1.13159572e-01],
       [ 2.42173831e-01, -2.86199097e-02, -3.25958378e-02,
         1.05027815e-01, -6.55406058e-01],
       [ 3.50196757e-01, -1.05016420e-02, -3.95274122e-01,
         5.92876802e-02,  2.91651800e-01],
       [ 2.25671537e-01, -3.42672779e-02,  8.92876610e-01,
         1.12426805e-01,  2.67065185e-01]])
```

Data Modeling : Linear Regression

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 [67]:

```
df_train.columns
```

Out[67]:

```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'typ
e',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWate
r',
       '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_3
5',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_sample
s',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weig
ht',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samp
les',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weigh
t',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'd
ebt',
       '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_sample
s',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'bad_debt', 'bins', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

In [68]:

```
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 [69]:

```
df_train['type'].unique()
```

Out[69]:

```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [70]:

```
df_test.replace(type_dict,inplace=True)
```

In [71]:

```
df_test['type'].unique()
```

Out[71]:

```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

In [72]:

```
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 [73]:

```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

In [74]:

```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

In [75]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error,accuracy_s
core
```

In [76]:

```
x_train.head()
```

Out[76]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_e
UID								
267822	53	36	13346	1	5230	67994.14790	0.02077	0.(
246444	141	18	46616	1	2633	50670.10337	0.02222	0.(
245683	63	18	46122	1	6881	95262.51431	0.00000	0.(
279653	127	72	927	2	2700	56401.68133	0.01086	0.(
247218	161	20	66502	1	5637	54053.42396	0.05426	0.(

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

In [77]:

```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

In [78]:

```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```

Out[78]:

```
LinearRegression()
```

In [79]:

```
y_pred=linereg.predict(x_test_scaled)
```

In [80]:

```
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_pr
ed)))
```

```
Overall R2 score of linear regression model 0.7348210754610929
Overall RMSE of linear regression model 323.1018894984635
```

From the R2 score and RMSE the accuracy and R2 score are good,we will further investigate the model performance at state level

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

In [81]:

```
state=df_train['STATEID'].unique()
state[0:5]
#Picking a few iDs 7,35,42,11
```

Out[81]:

```
array([36, 18, 72, 20,  1], dtype=int64)
```

In [82]:

```
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_t
est_nation,y_pred_nation))
    print("Overall RMSE of linear regression model for state,",i,":-" ,np.sqrt(mean_squ
ared_error(y_test_nation,y_pred_nation)))
    print("\n")
```

```
State ID- 20
Overall R2 score of linear regression model for state, 20 :- 0.60466037664
61809
Overall RMSE of linear regression model for state, 20 :- 307.9718899931472


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.810438247548
4617
Overall RMSE of linear regression model for state, 1 :- 307.8275861848435


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.78874464978
55252
Overall RMSE of linear regression model for state, 45 :- 225.6961542072413
4
```

Checking residuals

In [83]:

```
residuals=y_test-y_pred
residuals
```

Out[83]:

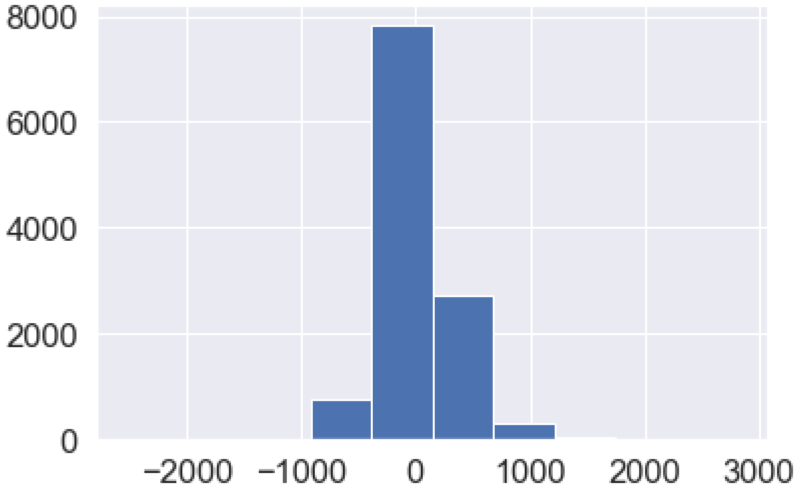
```
UID
255504    281.969088
252676    -69.935775
276314    190.761969
248614   -157.290627
286865     -9.887017
             ...
238088    -67.541646
242811    -41.578757
250127   -127.427569
241096   -330.820475
287763    217.760642
Name: hc_mortgage_mean, Length: 11709, dtype: float64
```

In [84]:

```
    plt.hist(residuals) # Normal distribution of residuals
```

Out[84]:

```
(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>)
```



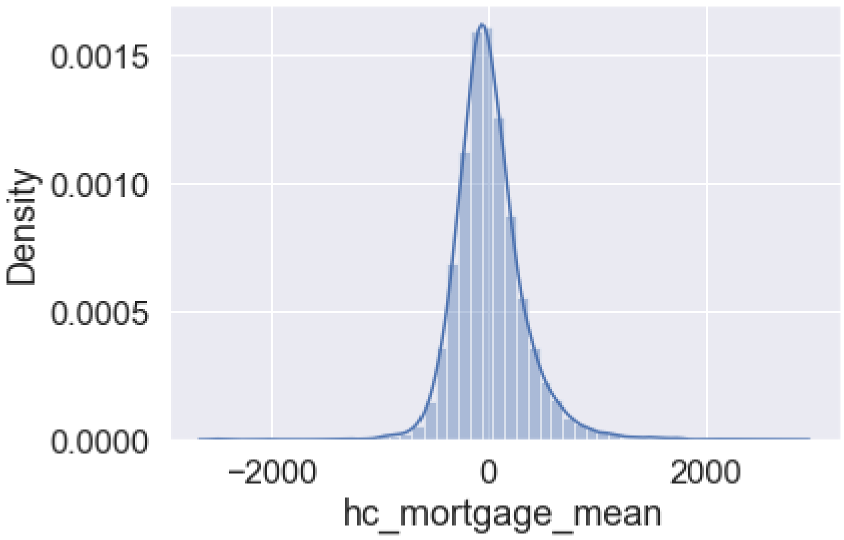
In [85]:

```
sns.distplot(residuals)
```

```
C:\Users\frees\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning:

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

Out[85]:

```
<AxesSubplot:xlabel='hc_mortgage_mean', ylabel='Density'>
```



In [86]:

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

Out[86]:

```
<matplotlib.collections.PathCollection at 0x184bd5e2c88>
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```