Real Estate Capstone Project – Shivam Sharma

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

1. The objective is to identify white spaces/potential business in the mortgage loan.
2. 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.
3. 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.
4. 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 n[illegible]mit 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
%matplotlib inline

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

Importing data

In [2]:

```
df_train=pd.read_csv("train.csv")
```

In [3]:

```
df_test=pd.read_csv("test.csv")
```

In [4]:

```
df_train.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]:

```
df_test.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]:
```
len(df_train)
```
Out[6]:
```
27321
```

In [7]:
```
len(df_test)
```
Out[7]:
```
11709
```

In [8]:
```
df_train.head()
```
Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.33333	22.51276	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.58333	23.43353	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.83333	23.94119	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.58333	24.32015	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.58333	11.10484	

5 rows × 80 columns

In [9]:
```
df_test.head()
```
Out[9]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	fema
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...	34.78682	33.75000	21.58531	
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...	44.23451	46.66667	22.37036	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...	41.62426	44.50000	22.86213	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...	44.81200	48.00000	21.03155	
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...	40.66618	42.66667	21.30900	

5 rows × 80 columns

In [10]:
```
df_test.describe()
```
Out[10]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	female_age_mean	female_ag
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	1.170900e+04	...	11613.000000	116
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	593.598514	37.405491	-91.340229	1.095500e+08	...	40.111999	
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	232.074263	5.625904	16.407818	7.624940e+08	...	5.851192	
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	201.000000	17.965835	-166.770979	8.299000e+03	...	15.360240	
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	404.000000	33.919813	-97.816561	1.718660e+06	...	36.729210	
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	612.000000	38.618093	-86.643344	4.835000e+06	...	40.196960	
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	787.000000	41.232973	-79.697311	3.204540e+07	...	43.496490	
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	989.000000	64.804269	-65.695344	5.520166e+10	...	90.107940	

8 rows × 74 columns

In [11]:
```
df_train.describe()
```
Out[11]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	female_age_mean	female_ag
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	2.732100e+04	...	27115.000000	271
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	596.507668	37.508813	-91.288394	1.295106e+08	...	40.319803	
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	232.497482	5.588268	16.343816	1.275531e+09	...	5.886317	
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	201.000000	17.929085	-165.453872	4.113400e+04	...	16.008330	
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	405.000000	33.899064	-97.816067	1.799408e+06	...	36.892050	
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	614.000000	38.755183	-86.554374	4.866940e+06	...	40.373320	
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	801.000000	41.380606	-79.782503	3.359820e+07	...	43.567120	
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	989.000000	67.074017	-65.379332	1.039510e+11	...	79.837390	

8 rows × 74 columns

In [12]:
```
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
```

In [13]:
```
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
```

2. 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 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)
df_train.head()#Check if the index has been set correctly
```

Out[14]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female_age_stdev	fem
267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	...	44.48629	45.33333	22.51276	
246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	...	36.48391	37.58333	23.43353	
245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	...	42.15810	42.83333	23.94119	
279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	...	47.77526	50.58333	24.32015	
247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	...	24.17693	21.58333	11.10484	

5 rows × 79 columns

In [15]:
```
df_test.head()#Check if the index has been set correctly
```

Out[15]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female_age_stdev
255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	...	34.78682	33.75000	21.58531
252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	...	44.23451	46.66667	22.37036
276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract	...	41.62426	44.50000	22.86213
248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	tract	...	44.81200	48.00000	21.03155
286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract	...	40.66618	42.66667	21.30900

5 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 [16]:
```
#percantage of missing values in train set
missing_list_train=df_train.isnull().sum() *100/len(df_train)
missing_values_df_train=pd.DataFrame(missing_list_train,columns=['Percantage of missing values'])
missing_values_df_train.sort_values(by=['Percantage of missing values'],inplace=True,ascending=False)
missing_values_df_train[missing_values_df_train['Percantage of missing values'] >0][:10]
```

Out[16]:

	Percantage 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

In [17]:
```
df_train.drop(['BLOCKID','SUMLEVEL'],axis=1,inplace=True) #Drop the BLOCKID feature since it is 100% missing values  and #SUMLEVEL doest not have an
```

In [18]:
```
#percantage 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=['Percantage of missing values'])
missing_values_df_test.sort_values(by=['Percantage of missing values'],inplace=True,ascending=False)
missing_values_df_test[missing_values_df_test['Percantage of missing values'] >0][:10]
```

Out[18]:

	Percantage 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

In [19]:
```
df_test.drop(['BLOCKID','SUMLEVEL'],axis=1,inplace=True) #Drop the BLOCKID feature since it is 43% missing values  and SUMLEVEL doest not have any p
```

In [20]:
```
# finding colums with missing values
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_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_mean', 'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weigh
t', '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', 'ma
le_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 [21]:
```
#Impute missing values with median
df_train[missing_train_cols]=df_train[missing_train_cols].fillna(df_train[missing_train_cols].median())
```

In [22]:
```
# find cols with missing values
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_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_mean', 'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weigh
t', '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', 'ma
le_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]:
```
#Impute missing values with median
df_test[missing_test_cols]=df_test[missing_test_cols].fillna(df_test[missing_test_cols].median())
```

In [24]:
```
df_train.isna().sum().sum()
```

Out[24]:
```
0
```

In [25]:
```
df_test.isna().sum().sum()
```

Out[25]:
```
0
```

EDA

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 [26]:
```
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 25
pysqldf = lambda q: sqldf(q, globals())
df_train_location_mort_pct=pysqldf(q1)
df_train_location_mort_pct.head()
```

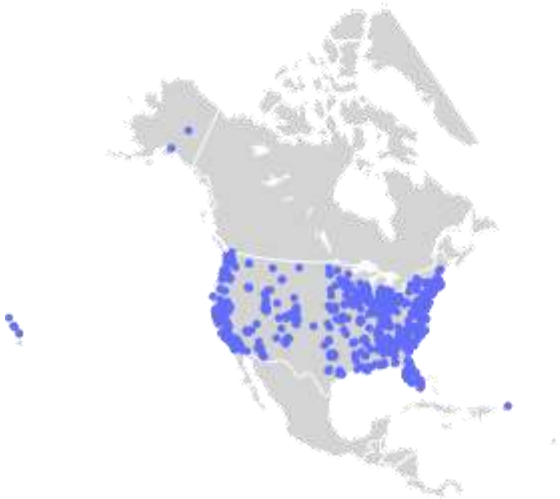
Out[26]:

	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 [27]:
```
import plotly.express as px
import plotly.graph_objects as go
```

In [28]:
```
import plotly.graph_objects as go
```

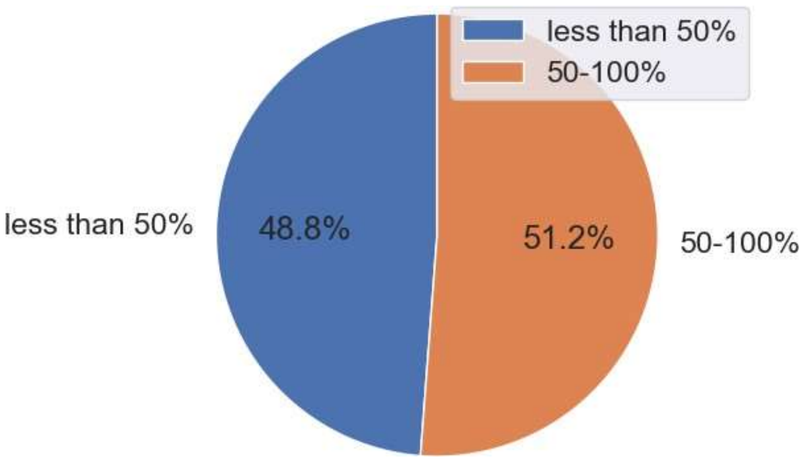
```python
fig = go.Figure(data=go.Scattergeo(
    lat=df_train_location_mort_pct['lat'],
    lon=df_train_location_mort_pct['lng']
))

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,
    ),
    width=800,  # Set the width of the plot
    height=600,  # Set the height of the plot
)
fig.show()
```



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 [29]:
```python
df_train['bad_debt']=df_train['second_mortgage']+df_train['home_equity']-df_train['home_equity_second_mortgage']
```

In [34]:
```python
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.ylabel('')
plt.legend()
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 [35]:
```
cols=[]
df_train.columns
```

Out[35]:
```
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')
```

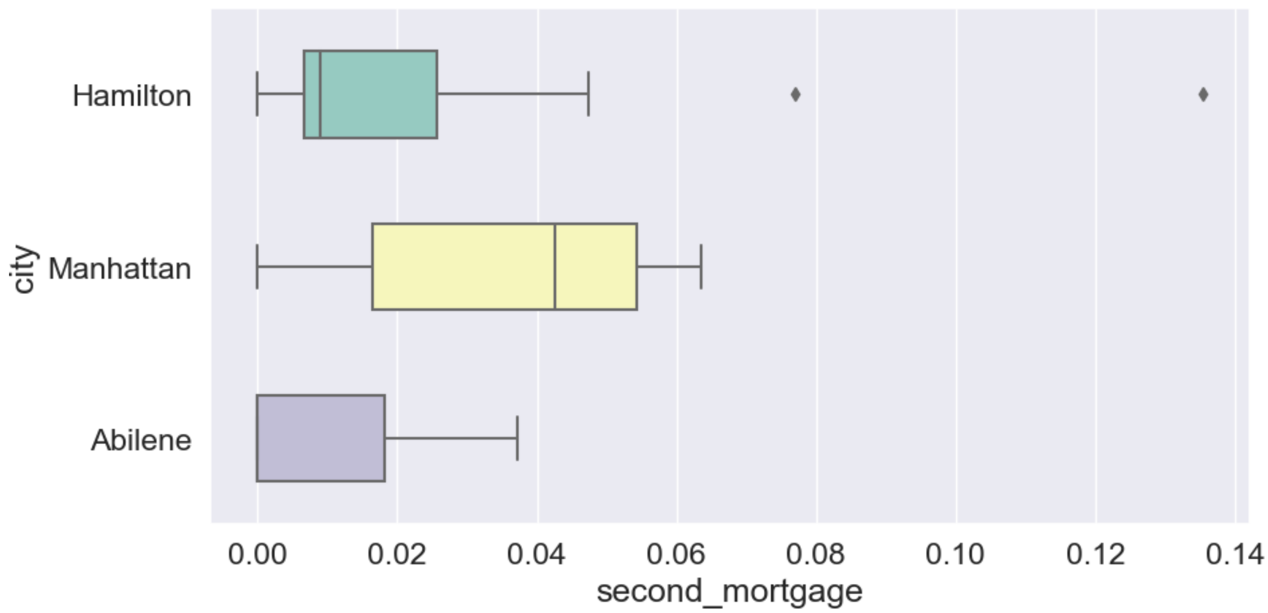
In [37]:
```
# there are too many cities we shall take a few as samples
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
df_box_hamilton=df_train.loc[df_train['city'] == 'Hamilton']
df_box_manhattan=df_train.loc[df_train['city'] == 'Manhattan']
df_box_abilene=df_train.loc[df_train['city'] == 'Abilene']
df_box_city=pd.concat([df_box_hamilton,df_box_manhattan,df_box_abilene])#Concatenate the dataframes
df_box_city.head(4)
```

Out[37]:

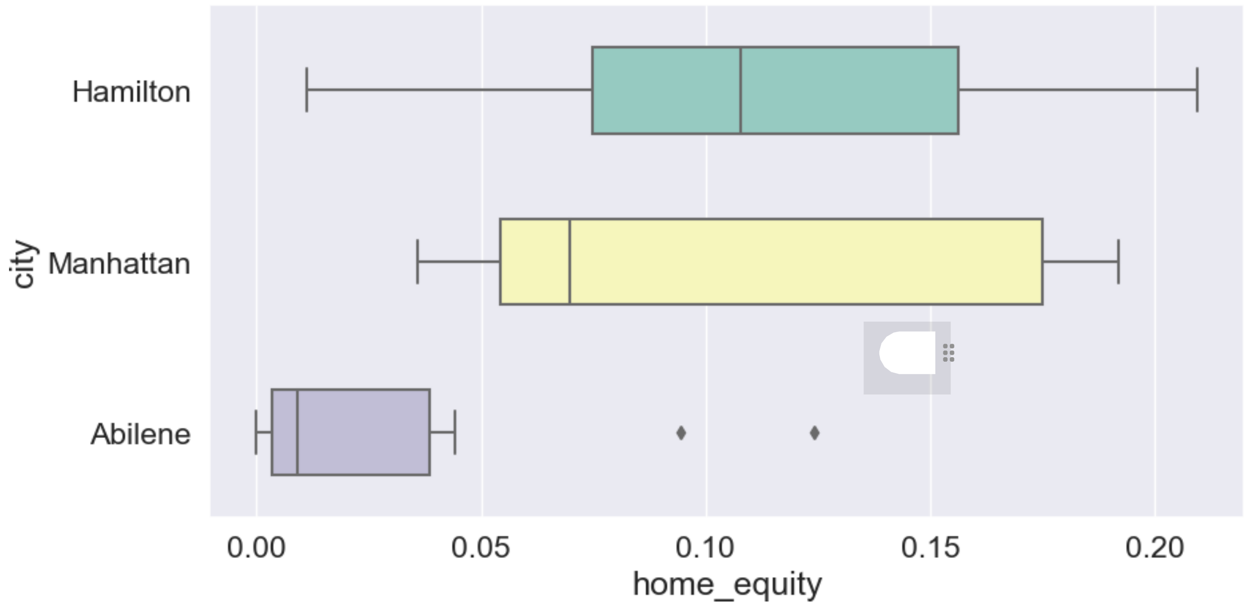
UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	...	female_age_stdev	female_age_sample_weight	female_age_samp
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	...	22.51276	685.33845	261
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	8610	609	...	24.05831	732.58443	312
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	45015	513	...	22.66500	565.32725	252
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	39746	662	...	22.79602	483.01311	195

4 rows × 79 columns

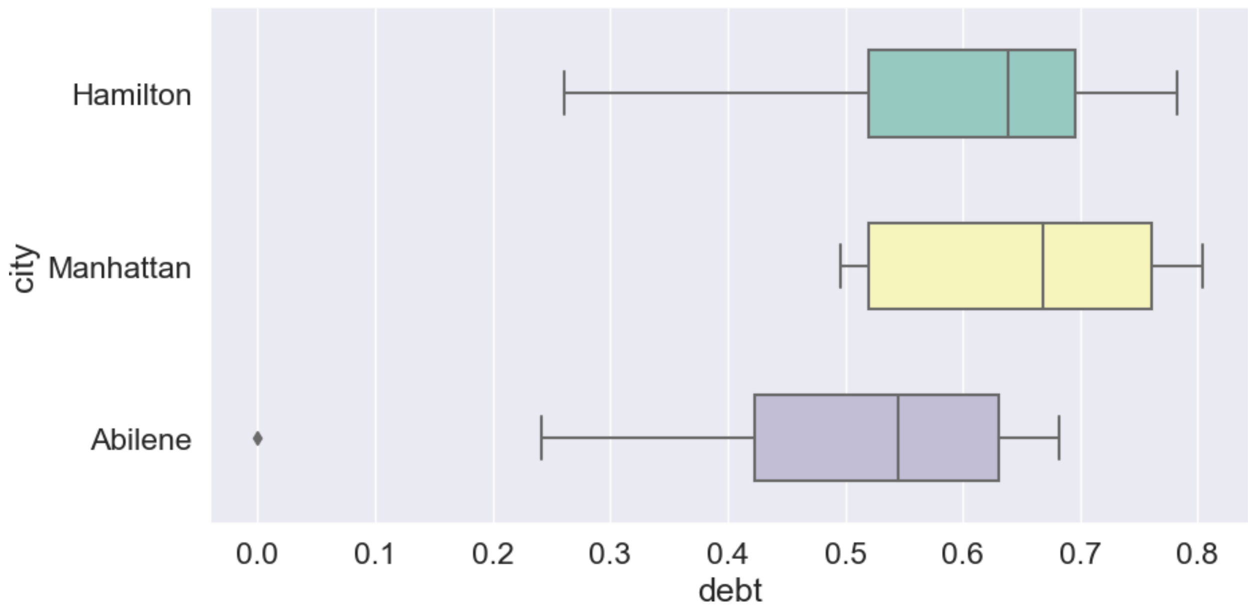
In [38]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.5,palette="Set3")
plt.show()
```



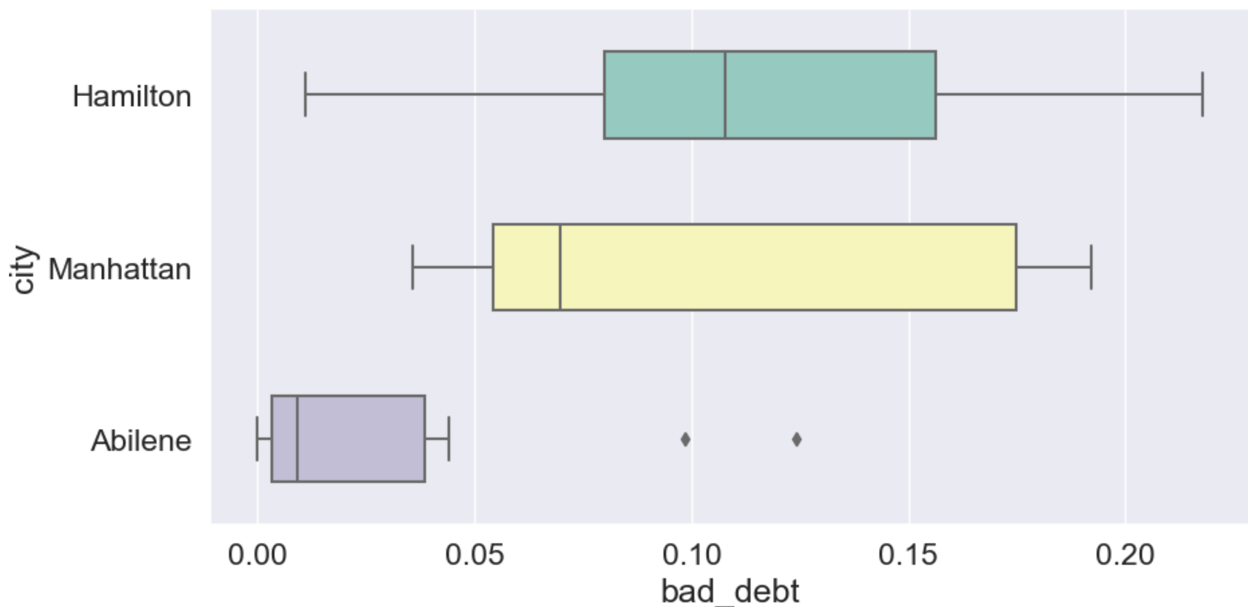
In [39]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



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



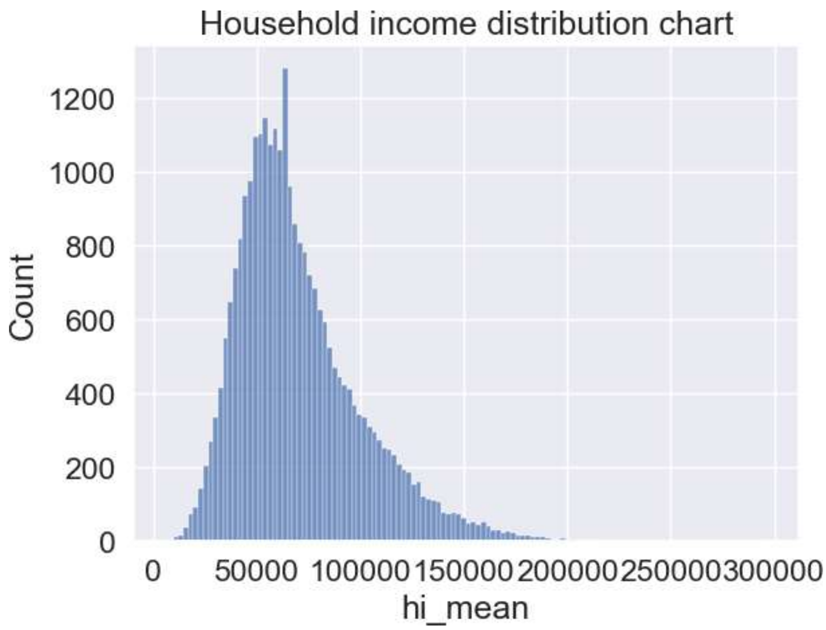
In [41]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='bad_debt', y='city',width=0.5,palette="Set3")
plt.show()
```



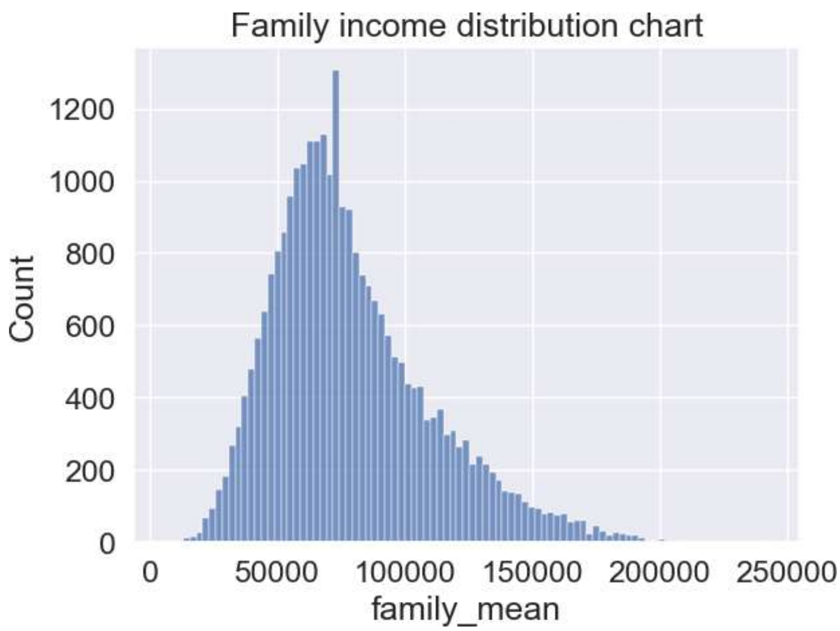
Manhattan has higher metrics as compared to Hamilton and Abilene

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

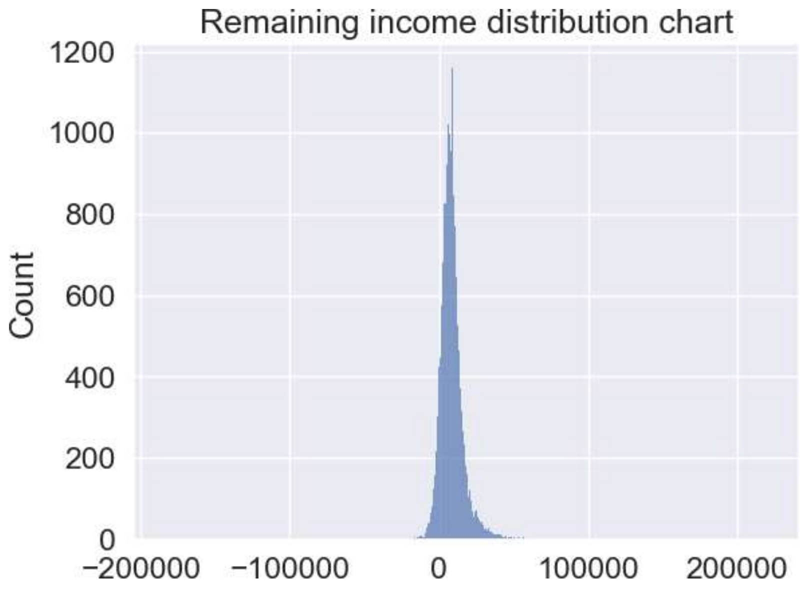
In [45]:
```
sns.histplot(df_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```



In [46]:
```
sns.histplot(df_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```



In [48]:
```
sns.histplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
```

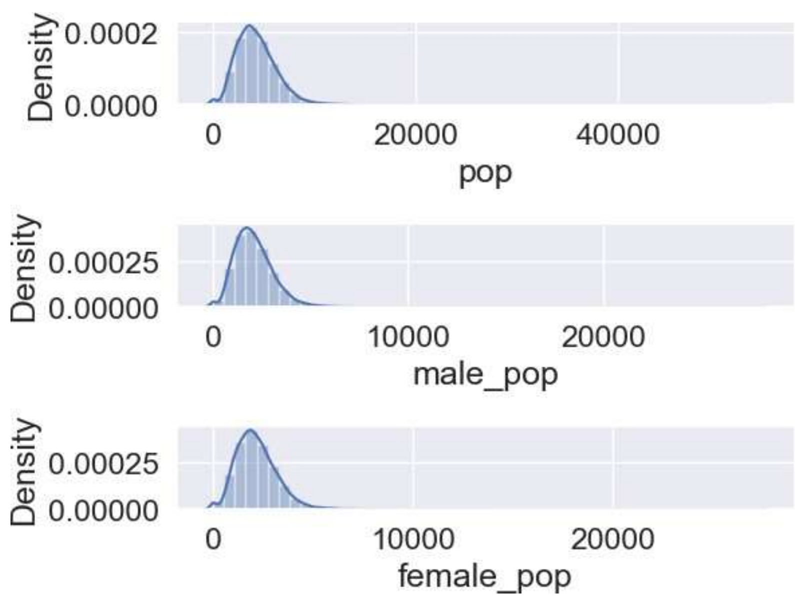


Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):

In [55]:

```
#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()
```

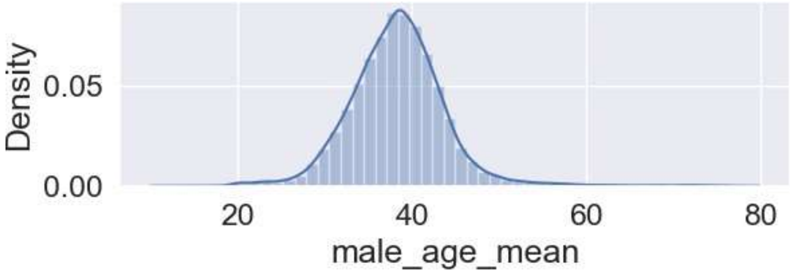
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`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).

C:\ProgramData\Anaconda3\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).

C:\ProgramData\Anaconda3\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).
```

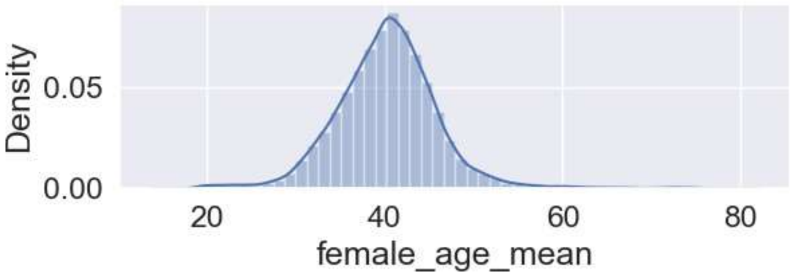


In [52]:

```
#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()
plt.figure(figsize=(25,10))#Plotting the age distribution
```

```
C:\ProgramData\Anaconda3\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).

C:\ProgramData\Anaconda3\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[52]:

```
<Figure size 2500x1000 with 0 Axes>

<Figure size 2500x1000 with 0 Axes>
```

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

In [56]:

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

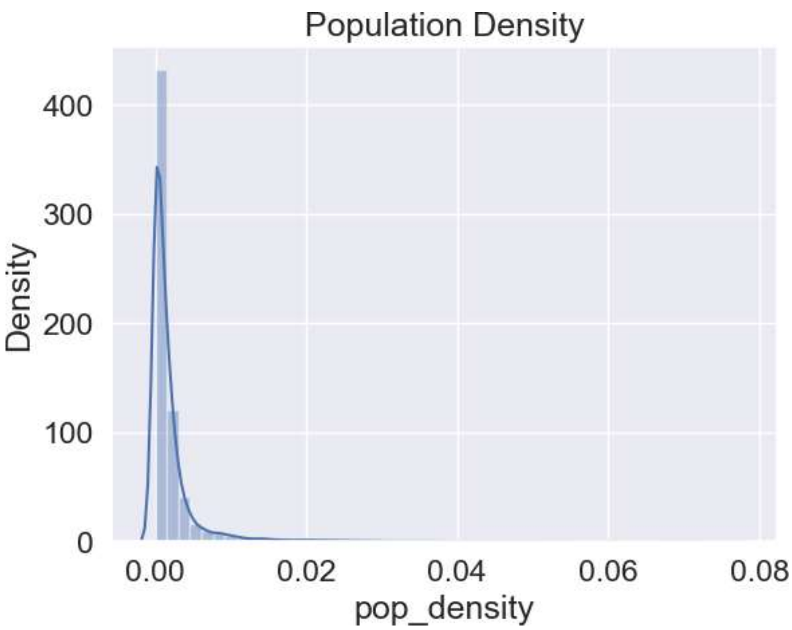
In [57]:

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

In [58]:

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

```
C:\ProgramData\Anaconda3\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 [59]:

```
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 [60]:

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

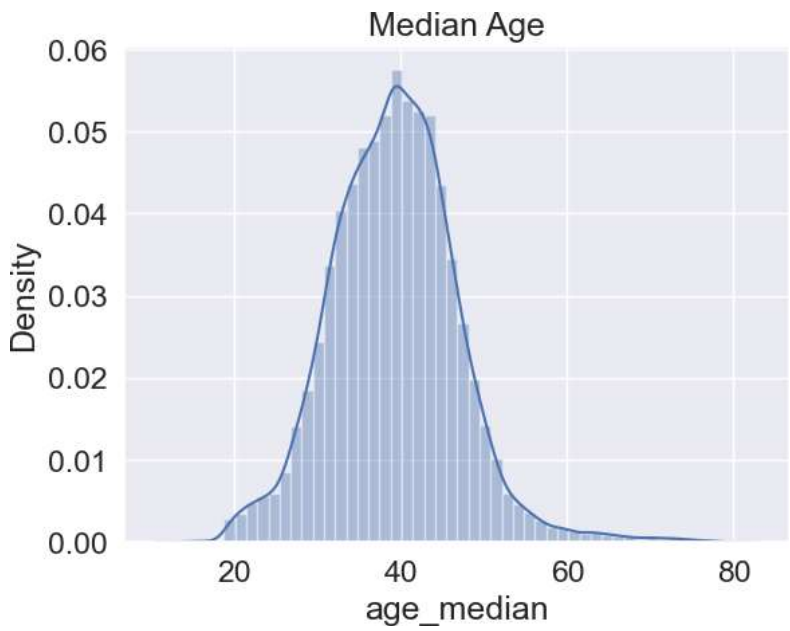
Out[60]:

UID	male_age_median	female_age_median	male_pop	female_pop	age_median
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 [61]:

```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
```

```
C:\ProgramData\Anaconda3\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).
```



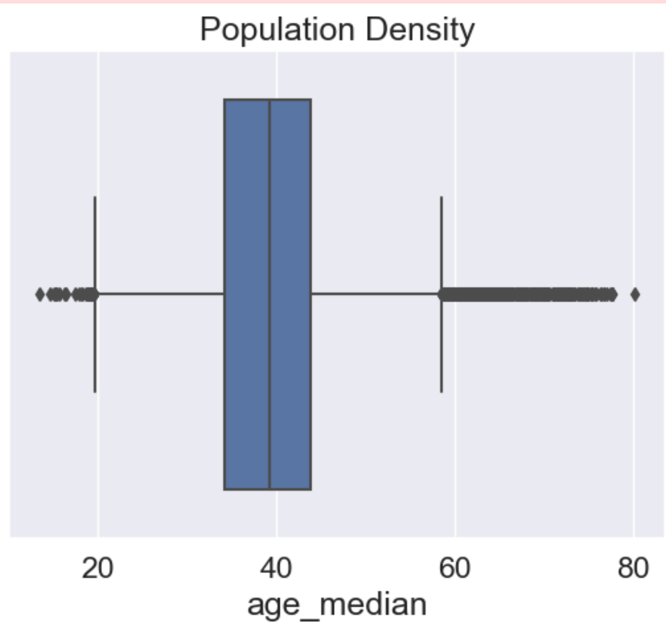
In [62]:

```
# 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
```

In [63]:

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

```
C:\ProgramData\Anaconda3\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 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 [64]:
```
df_train['pop'].describe()
```

Out[64]:
```
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 [65]:
```
df_train['pop_bins']=pd.cut(df_train['pop'],bins=5,labels=['very low','low','medium','high','very high'])#Binning the population
```

In [66]:
```
df_train[['pop','pop_bins']]
```

Out[66]:

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 [67]:
```
df_train['pop_bins'].value_counts()
```

Out[67]:
```
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 [68]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()#Counting the number of people in each category
```

Out[68]:

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 [69]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"])
```

Out[69]:

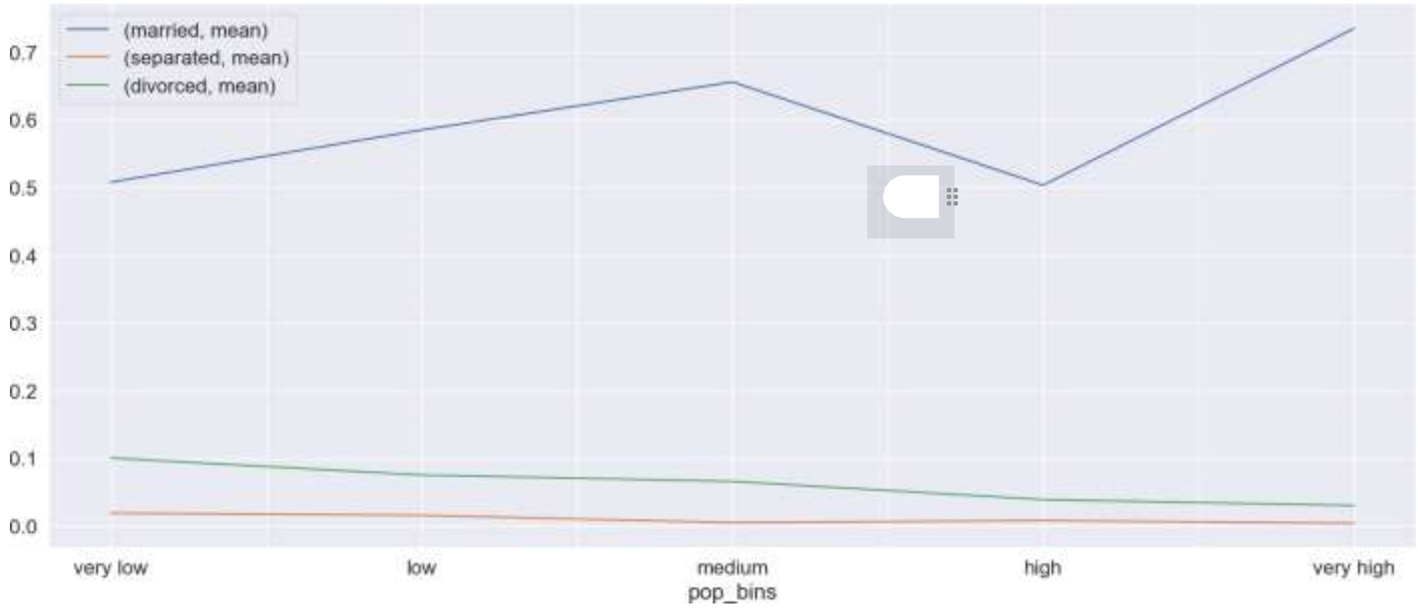
	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.507677	0.526665	0.019087	0.013460	0.100468	0.095205
low	0.584894	0.593135	0.015833	0.011195	0.075348	0.070045
medium	0.655737	0.618710	0.005003	0.004120	0.065927	0.064890
high	0.503359	0.335660	0.008141	0.002500	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 the findings using appropriate chart type

In [71]:
```
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 1000x500 with 0 Axes>
```



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

In [73]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
print(rent_state_mean)
```

```
                             mean
state
Alabama                773.262682
Alaska                1185.763570
Arizona               1095.872790
Arkansas               719.785963
California            1470.088726
Colorado              1196.914665
Connecticut           1316.787850
Delaware              1127.309811
District of Columbia  1414.055096
Florida               1140.108292
Georgia                963.788830
Hawaii                1707.393377
Idaho                  800.486650
Illinois              1034.704750
Indiana                810.553939
Iowa                   736.892211
Kansas                 828.860920
Kentucky               739.934539
Louisiana              844.949841
Maine                  829.941899
Maryland              1410.673337
Massachusetts         1210.703169
Michigan               927.143055
Minnesota              956.561021
Mississippi            737.694004
Missouri               827.981544
Montana                774.652428
Nebraska               834.623685
Nevada                1127.806232
New Hampshire         1082.179938
New Jersey            1378.110381
New Mexico             853.611858
New York              1246.691583
North Carolina         884.356353
North Dakota           771.423137
Ohio                   819.537597
Oklahoma               777.702422
Oregon                1024.300380
Pennsylvania           948.665551
Puerto Rico            544.950649
Rhode Island          1038.349457
South Carolina         859.661748
South Dakota           685.325569
Tennessee              852.817783
Texas                  976.080760
Utah                  1067.173019
Vermont                935.501922
Virginia              1303.648392
Washington            1126.461192
West Virginia          667.193267
Wisconsin              841.101778
Wyoming                859.637825
```

In [74]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
print(income_state_mean)
```

```
                              mean
state
Alabama                66985.565959
Alaska                 92136.545109
Arizona                73238.038156
Arkansas               64765.377850
California             87594.902907
Colorado               88511.653831
Connecticut           104209.698547
Delaware               87190.491455
District of Columbia  107189.896223
Florida                72300.712774
Georgia                73222.469051
Hawaii                 93674.291438
Idaho                  66949.755112
Illinois               81992.385579
Indiana                67433.323667
Iowa                   74146.568840
Kansas                 74975.441508
Kentucky               66908.412825
Louisiana              69547.216033
Maine                  71091.495286
Maryland              101171.509330
Massachusetts          98366.774298
Michigan               72632.674568
Minnesota              86526.043971
Mississippi            59365.463764
Missouri               71003.063200
Montana                71857.699842
Nebraska               76503.220795
Nevada                 73998.702963
New Hampshire          90584.451014
New Jersey            100791.708227
New Mexico             69229.674498
New York               86534.411140
North Carolina         72725.398431
North Dakota           82925.457953
Ohio                   71893.086190
Oklahoma               66860.468732
Oregon                 77295.773289
Pennsylvania           79730.649076
Puerto Rico            36062.250655
Rhode Island           84494.541412
South Carolina         67911.589554
South Dakota           74553.497414
Tennessee              69504.790609
Texas                  75696.181815
Utah                   80923.048435
Vermont                79704.476529
Virginia               92815.686527
Washington             84368.396258
West Virginia          64522.201637
Wisconsin              75201.386491
Wyoming                79763.594205
```

In [75]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
print(rent_perc_of_income)
```

```
state
Alabama                 0.011544
Alaska                  0.012870
Arizona                 0.014963
Arkansas                0.011114
California              0.016783
Colorado                0.013523
Connecticut             0.012636
Delaware                0.012929
District of Columbia    0.013192
Florida                 0.015769
Georgia                 0.013162
Hawaii                  0.018227
Idaho                   0.011957
Illinois                0.012620
Indiana                 0.012020
Iowa                    0.009938
Kansas                  0.011055
Kentucky                0.011059
Louisiana               0.012149
Maine                   0.011674
Maryland                0.013943
Massachusetts           0.012308
Michigan                0.012765
Minnesota               0.011055
Mississippi             0.012426
Missouri                0.011661
Montana                 0.010780
Nebraska                0.010910
Nevada                  0.015241
New Hampshire           0.011947
New Jersey              0.013673
New Mexico              0.012330
New York                0.014407
North Carolina          0.012160
North Dakota            0.009303
Ohio                    0.011399
Oklahoma                0.011632
Oregon                  0.013252
Pennsylvania            0.011898
Puerto Rico             0.015111
Rhode Island            0.012289
South Carolina          0.012659
South Dakota            0.009192
Tennessee               0.012270
Texas                   0.012895
Utah                    0.013188
Vermont                 0.011737
Virginia                0.014046
Washington              0.013352
West Virginia           0.010341
Wisconsin               0.011185
Wyoming                 0.010777
Name: mean, dtype: float64
```

In [76]:

```
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

Out[76]:

```
0.013354608441451008
```

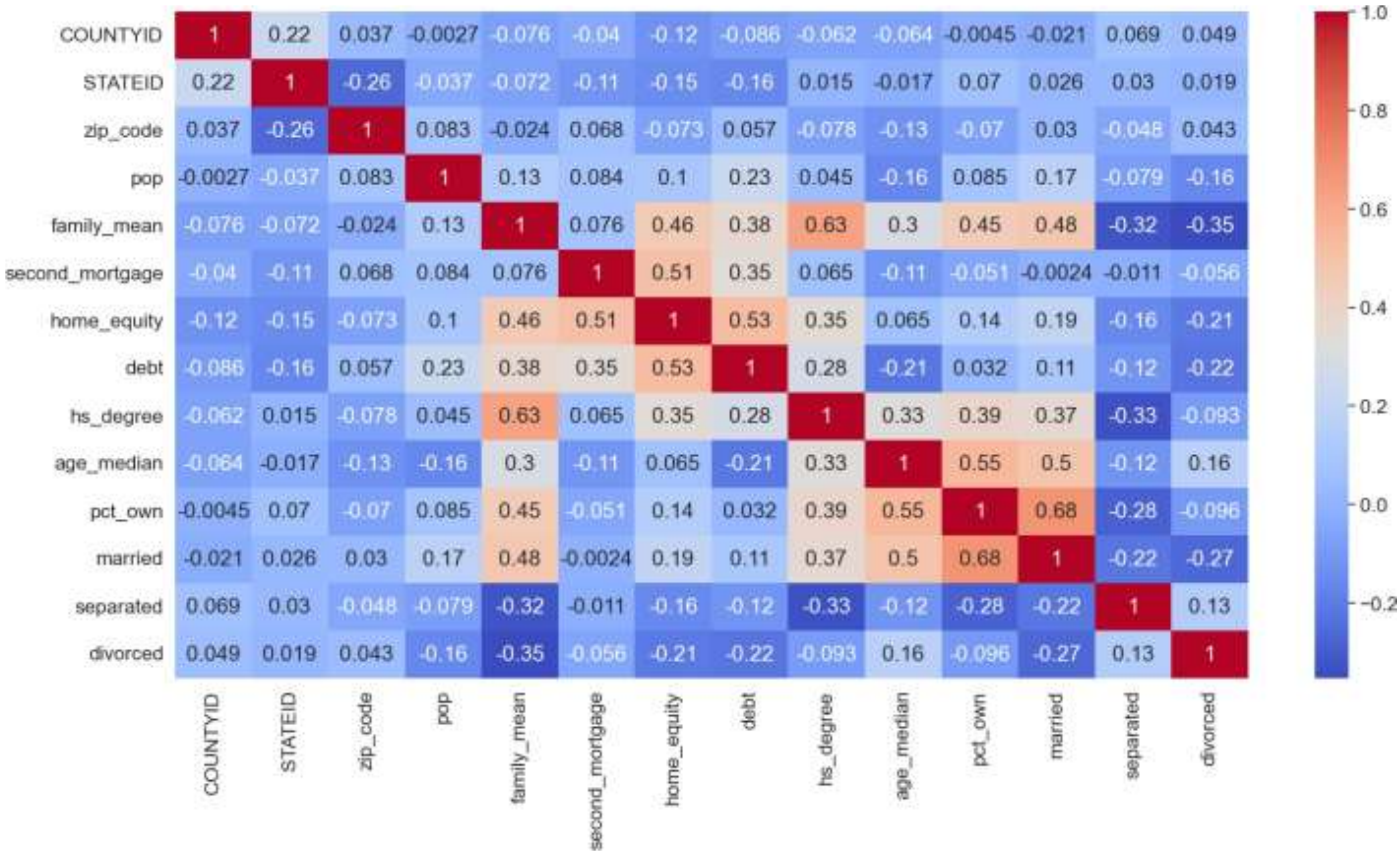
Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

In [77]:

```
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()
print(cor)
```

```
                COUNTYID   STATEID  zip_code       pop  family_mean  \
COUNTYID        1.000000  0.224549  0.036527 -0.002662    -0.075838
STATEID         0.224549  1.000000 -0.261465 -0.036599    -0.071887
zip_code        0.036527 -0.261465  1.000000  0.083058    -0.024101
pop            -0.002662 -0.036599  0.083058  1.000000     0.131625
family_mean    -0.075838 -0.071887 -0.024101  0.131625     1.000000
second_mortgage -0.039559 -0.112724  0.068167  0.084204     0.075992
home_equity    -0.124055 -0.145400 -0.072972  0.101290     0.459500
debt           -0.086057 -0.160368  0.057487  0.228425     0.378059
hs_degree      -0.062490  0.014590 -0.078077  0.045483     0.634316
age_median     -0.063522 -0.017173 -0.126153 -0.162498     0.300619
pct_own        -0.004451  0.069618 -0.070455  0.084742     0.450667
married        -0.021335  0.025984  0.029994  0.165773     0.481915
separated       0.068857  0.029950 -0.047565 -0.079384    -0.323305
divorced        0.048778  0.018576  0.043478 -0.159490    -0.353194

                second_mortgage  home_equity      debt  hs_degree  \
COUNTYID              -0.039559    -0.124055 -0.086057  -0.062490
STATEID               -0.112724    -0.145400 -0.160368   0.014590
zip_code               0.068167    -0.072972  0.057487  -0.078077
pop                    0.084204     0.101290  0.228425   0.045483
family_mean            0.075992     0.459500  0.378059   0.634316
second_mortgage        1.000000     0.510560  0.350668   0.064751
home_equity            0.510560     1.000000  0.531767   0.354612
debt                   0.350668     0.531767  1.000000   0.279636
hs_degree              0.064751     0.354612  0.279636   1.000000
age_median            -0.114125     0.064814 -0.214633   0.333946
pct_own               -0.050649     0.142537  0.032037   0.390549
married               -0.002445     0.191500  0.106256   0.370545
separated             -0.010906    -0.155348 -0.118963  -0.333738
divorced              -0.055915    -0.206778 -0.222918  -0.093316

                age_median   pct_own   married  separated  divorced
COUNTYID         -0.063522 -0.004451 -0.021335   0.068857  0.048778
STATEID          -0.017173  0.069618  0.025984   0.029950  0.018576
zip_code         -0.126153 -0.070455  0.029994  -0.047565  0.043478
pop              -0.162498  0.084742  0.165773  -0.079384 -0.159490
family_mean       0.300619  0.450667  0.481915  -0.323305 -0.353194
second_mortgage  -0.114125 -0.050649 -0.002445  -0.010906 -0.055915
home_equity       0.064814  0.142537  0.191500  -0.155348 -0.206778
debt             -0.214633  0.032037  0.106256  -0.118963 -0.222918
hs_degree         0.333946  0.390549  0.370545  -0.333738 -0.093316
age_median        1.000000  0.546013  0.495081  -0.116701  0.164222
pct_own           0.546013  1.000000  0.682076  -0.284650 -0.095555
married           0.495081  0.682076  1.000000  -0.219870 -0.267903
separated        -0.116701 -0.284650 -0.219870   1.000000  0.133399
divorced          0.164222 -0.095555 -0.267903   0.133399  1.000000
```

In [78]:
```python
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

We encounter numerous variables in economic data that must be measured. Our objective is to uncover how measured variables relate to lesser known, unobserved variables or common factors. To assess these common factors, we presume that every variable relies on a linear construction of them, known as the loadings.

Additionally, each measured variable is affected by specific changes, also called "specific variance," inflicting random impacts on a variable. After obtaining the list of common factors and depict the tabulated loadings, factor analysis allows us to identify alternatives latent variables present in our dataset, giving us a novel viewpoint at the linear connections embedded within the data. Below details the list of latent variables uncovered:

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

In [80]:

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

In [81]:

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

```
Out[81]: array([[-1.13298111e-01,  1.96255769e-02, -2.32690560e-02,
         -6.24938156e-02,  4.16684826e-02],
        [-1.11234732e-01,  1.32909194e-02,  2.94028566e-02,
         -1.49584365e-01,  1.09715332e-01],
        [-8.20860271e-02,  5.20290602e-02, -1.38169227e-01,
         -5.00788857e-02, -1.03169256e-01],
        [ 1.73102455e-02,  1.89624897e-02,  6.24329160e-03,
          2.67600027e-02, -7.17988183e-03],
        [ 9.22758000e-02, -9.67542739e-02, -6.80453714e-02,
         -1.33520873e-01, -1.46139321e-01],
        [-1.14015352e-02, -4.14310228e-02,  1.47478051e-01,
          9.05437712e-03,  1.07284781e-01],
        [-4.29446684e-02, -2.08391214e-02,  3.70211997e-02,
         -9.45716929e-02,  5.88388660e-02],
        [-2.64199155e-03, -1.53210235e-02, -2.48800809e-03,
         -4.52221002e-02,  2.35523813e-02],
        [ 8.21963740e-02,  9.58785505e-01, -8.60470483e-02,
         -7.80664647e-03, -3.93054848e-02],
        [ 7.56724520e-02,  9.19629146e-01, -1.07717191e-01,
         -2.89666293e-02, -3.97165713e-02],
        [ 8.48070843e-02,  9.49558497e-01, -5.98172996e-02,
          1.40572011e-02, -3.74578846e-02],
        [ 7.68234069e-01,  1.03321266e-02, -3.75681038e-02,
          1.14555193e-01, -1.25940511e-01],
        [ 7.16872869e-01,  6.97015312e-03, -4.65339547e-02,
          1.08807518e-01, -1.37296913e-01],
        [ 7.07945401e-01,  2.60450021e-02, -8.17256769e-03,
          1.03980147e-01,  7.76088520e-02],
        [-1.29671542e-01,  3.40800184e-01, -4.82968197e-01,
         -4.24535265e-02,  3.26857606e-01],
        [ 2.35489398e-01,  4.41121577e-01, -6.34624452e-01,
         -2.66848940e-02,  3.56702243e-01],
        [-4.55293869e-02,  3.21694983e-02,  2.93213347e-02,
          4.44616878e-01, -1.64431252e-01],
        [-2.49835242e-02,  1.49999234e-02,  4.48788265e-02,
          6.76128790e-01, -1.55882615e-01],
        [-3.86012815e-02, -1.80824068e-02,  8.16123510e-02,
          8.36685682e-01, -9.19913767e-02],
        [-5.08188895e-02, -3.64732707e-02,  1.11799167e-01,
          9.25674674e-01, -4.45026786e-02],
        [-6.01204862e-02, -4.43880861e-02,  1.37268437e-01,
          9.53830475e-01, -2.22201267e-02],
        [-4.49884285e-02, -5.23690474e-02,  1.42802792e-01,
          9.33605669e-01, -8.81671459e-05],
        [-4.11150777e-02, -5.85563708e-02,  1.30715177e-01,
          8.88204882e-01,  1.06635282e-02],
        [-2.38450318e-02, -7.20685666e-02,  9.60677098e-02,
          7.79799423e-01,  3.00114033e-02],
        [ 2.19375564e-01,  4.68684953e-01, -6.10668657e-01,
         -2.71044844e-02,  3.77760089e-01],
        [ 2.40060333e-01,  4.49638263e-01, -6.24711029e-01,
         -2.94747056e-02,  3.54497874e-01],
        [ 7.83083048e-01,  4.86686913e-02,  1.43254474e-01,
         -2.05511326e-01, -1.58638395e-01],
        [ 7.07930115e-01,  4.95456979e-02,  1.30314139e-01,
         -2.19307710e-01, -2.14927444e-01],
        [ 8.61622514e-01,  4.29562525e-02,  1.66667789e-01,
         -1.20545375e-01,  2.93879865e-02],
        [-2.20422624e-01,  8.47822645e-01, -4.08604119e-02,
          6.79875744e-02,  2.30883636e-01],
        [ 1.44926417e-01,  9.53444814e-01,  2.68729382e-02,
         -4.68742656e-02,  1.00487172e-01],
        [ 8.28622100e-01,  3.38785914e-02,  1.60744124e-01,
         -2.04923576e-01, -7.84943083e-02],
        [ 7.92619853e-01,  2.81796719e-02,  1.50728901e-01,
         -2.07906070e-01, -9.49846519e-02],
        [ 8.11632008e-01,  4.25175153e-02,  1.44729727e-01,
         -1.08498854e-01,  5.63105467e-02],
        [-3.36826943e-01,  8.66265710e-01,  3.97015555e-02,
          9.08423234e-02,  4.42460594e-02],
        [ 4.93930179e-02,  9.35495452e-01,  1.53733112e-01,
         -2.53290177e-02, -9.73664722e-02],
        [ 9.77198386e-01, -3.12035523e-02, -9.64438875e-02,
          4.70255034e-02,  7.20102904e-02],
        [ 9.58046809e-01, -3.69432469e-02, -1.11761259e-01,
          4.73361596e-02,  6.47730305e-02],
        [ 8.14135415e-01,  2.72855626e-03,  8.22175954e-02,
          2.07832585e-02,  1.26599477e-01],
        [-4.17138320e-01,  7.19543592e-01,  3.42627945e-01,
         -7.04981159e-02, -2.84146197e-01],
        [ 7.37525976e-02,  7.25138226e-01,  2.76353095e-01,
         -4.72065750e-02, -3.60354801e-01],
        [ 9.10177014e-01, -5.00945059e-02, -3.62915046e-02,
          2.56705367e-04,  1.65651371e-01],
        [ 8.72817017e-01, -4.94330675e-02, -4.85977340e-02,
         -5.42628884e-04,  1.54250510e-01],
        [ 7.55635172e-01, -8.25632031e-04,  6.34921826e-02,
          4.75264494e-03,  2.59414270e-01],
        [-1.25546493e-01,  6.09318423e-01,  6.42291409e-01,
         -2.09758135e-02,  2.42853374e-01],
        [-3.44164122e-01,  5.62607873e-01,  5.98002315e-01,
         -2.42421997e-02,  2.14609170e-01],
        [-1.54967825e-01, -1.14439765e-02, -1.60675602e-01,
          1.10364767e-01, -6.53106927e-01],
        [-1.31416487e-01, -1.79976628e-02, -1.62299788e-01,
          1.26203469e-01, -6.63071156e-01],
        [ 2.48215350e-01, -2.45028251e-02, -3.26359211e-02,
          9.53487771e-02, -6.38551295e-01],
        [ 2.03556805e-01,  7.64766515e-02, -3.10575802e-01,
          2.23499982e-02, -6.29949754e-01],
```

```
[ 1.05622414e-01, -6.20036356e-02, -2.36013491e-02,
 -9.40334652e-02,  6.77644851e-01],
[-2.66727113e-01, -5.72122133e-03, -2.77715949e-02,
 -9.30230779e-02,  6.44699033e-01],
[-2.15353612e-01, -7.26461572e-02,  3.57613478e-01,
 -1.95184801e-02,  6.37000528e-01],
[ 3.92584117e-01,  5.81025283e-02,  2.52179455e-01,
 -2.21583739e-01, -1.88017217e-01],
[ 4.06107247e-01,  5.98064849e-02,  2.20918992e-01,
 -2.11291928e-01, -1.75588184e-01],
[ 3.51035972e-01,  5.04840459e-02,  2.66181407e-01,
 -2.18136031e-01, -1.84206957e-01],
[ 2.36302953e-01, -4.95779551e-02,  8.16743077e-01,
  9.23414904e-02,  3.26192892e-01],
[ 2.42117283e-01, -3.44186923e-02,  8.32654326e-01,
  7.39018414e-02,  2.44088930e-01],
[-5.79462057e-02,  6.75601117e-02,  5.86353776e-01,
  8.63601640e-02,  9.50625276e-02],
[ 5.30849926e-02,  8.17277296e-01, -1.76316854e-01,
 -1.59912023e-02, -3.68571006e-02],
[ 7.01058125e-02,  9.23054071e-01, -1.04890855e-01,
 -2.83802817e-02, -4.65119953e-02],
[ 1.96005547e-01, -4.78271389e-02,  8.05660351e-01,
  1.42191809e-01,  3.33865801e-01],
[ 1.89799337e-01, -3.37772066e-02,  8.59444876e-01,
  1.30024910e-01,  2.53833843e-01],
[-9.23696925e-02,  6.24568769e-02,  4.73309395e-01,
  7.27217833e-02,  1.17491137e-01],
[ 6.06349187e-02,  8.78010880e-01, -1.48115478e-01,
  2.15485185e-02, -4.38833228e-02],
[ 7.87718081e-02,  9.54725865e-01, -5.67836225e-02,
  1.57428000e-02, -4.49269937e-02],
[-3.44102683e-02,  1.08436006e-01,  7.80930918e-01,
 -4.29856525e-02, -2.89890279e-01],
[ 1.77718043e-01,  1.89451749e-01,  5.59576970e-01,
 -1.21319632e-01, -1.36887744e-01],
[-6.40811130e-02, -6.89807964e-02, -2.66336158e-01,
  1.29069261e-01,  1.91442118e-01],
[-1.53795908e-01, -6.85388611e-02, -1.44514838e-01,
  1.24406596e-01,  1.50272715e-01],
[-3.52851786e-01, -5.13784296e-02,  1.47741155e-01,
  2.82495922e-02,  1.17176086e-01],
[ 2.45441928e-01, -2.76065037e-02, -3.86634626e-02,
  1.05072116e-01, -6.51205316e-01],
[ 3.50739682e-01, -1.00313539e-02, -3.91555232e-01,
  5.87290135e-02,  2.95442541e-01],
[ 2.27776511e-01, -3.50086250e-02,  8.94244741e-01,
  1.11005561e-01,  2.64936447e-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 [82]:
```
df_train.columns
```

Out[82]:
```
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 [83]:
```
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 [84]:
```
df_train['type'].unique()
```

Out[84]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [85]:
```
df_test.replace(type_dict,inplace=True)
```

In [86]:
```
df_test['type'].unique()
```

Out[86]:
```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

In [87]:
```
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 [88]:
```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

In [89]:
```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

In [90]:
```
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 [91]:
```
x_train.head()
```

Out[91]:

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 [92]:
```
# scaling the data
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 [93]:
```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```

Out[93]:
```
LinearRegression()
```

In [94]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Out[94]:
```
LinearRegression(normalize=False)
```

In [95]:
```
y_pred=linereg.predict(x_test_scaled)
```

In [96]:
```
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 0.7339854188499229
Overall RMSE of linear regression model 323.8841523769673
```

Overall R2 score for the linear regression model is 0.7348210754610929 and overall RMSE is 323.1018894984635.

Although the Accuracy and R2 score are satisfactory, further investigation of the model's performance at the state level is warranted. Therefore, we proceed to state level analysis.

In [97]:
```
state=df_train['STATEID'].unique()
state[0:5]
```

Out[97]:
```
array([36, 18, 72, 20,  1], dtype=int64)
```

In [99]:
```
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 : 0.6095965086676083
Overall RMSE of linear regression model for state, 20 : 325.91249059701727


State ID- 1
Overall R2 score of linear regression model for state, 1 : 0.8108586083394038
Overall RMSE of linear regression model for state, 1 : 308.15989834073866


State ID- 45
Overall R2 score of linear regression model for state, 45 : 0.7874995932643833
Overall RMSE of linear regression model for state, 45 : 226.36025963997875
```

In [100…

```
# Checking Residuals

residuals=y_test-y_pred
residuals
```

Out[100]:

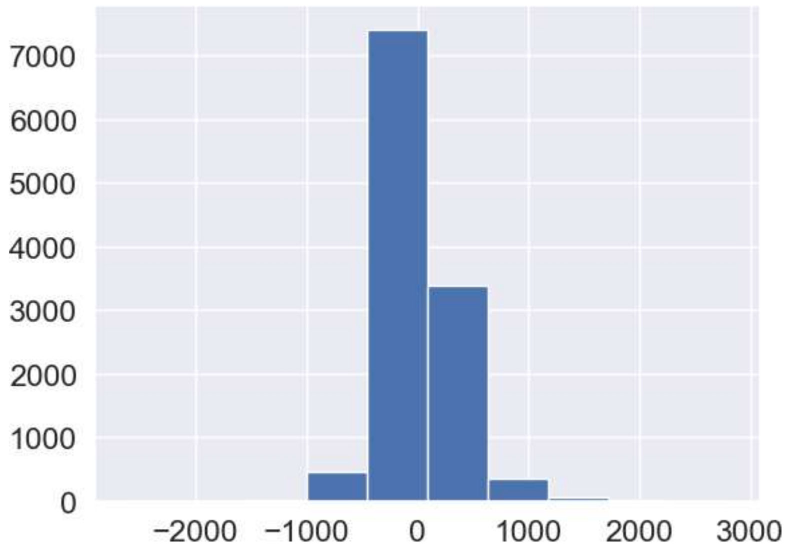
```
UID
255504    279.050341
252676    -73.065620
276314    192.730495
248614   -162.801962
286865    -12.575729
             ...
238088    -70.024997
242811    -36.023945
250127   -129.971952
241096   -332.004995
287763    220.967320
Name: hc_mortgage_mean, Length: 11709, dtype: float64
```

In [101…

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

Out[101]:

```
(array([4.000e+00, 4.000e+00, 1.700e+01, 4.530e+02, 7.409e+03, 3.388e+03,
        3.620e+02, 5.500e+01, 1.400e+01, 3.000e+00]),
 array([-2635.76699058, -2090.36503218, -1544.96307377,  -999.56111537,
         -454.15915697,    91.24280144,   636.64475984,  1182.04671825,
         1727.44867665,  2272.85063505,  2818.25259346]),
 <BarContainer object of 10 artists>)
```



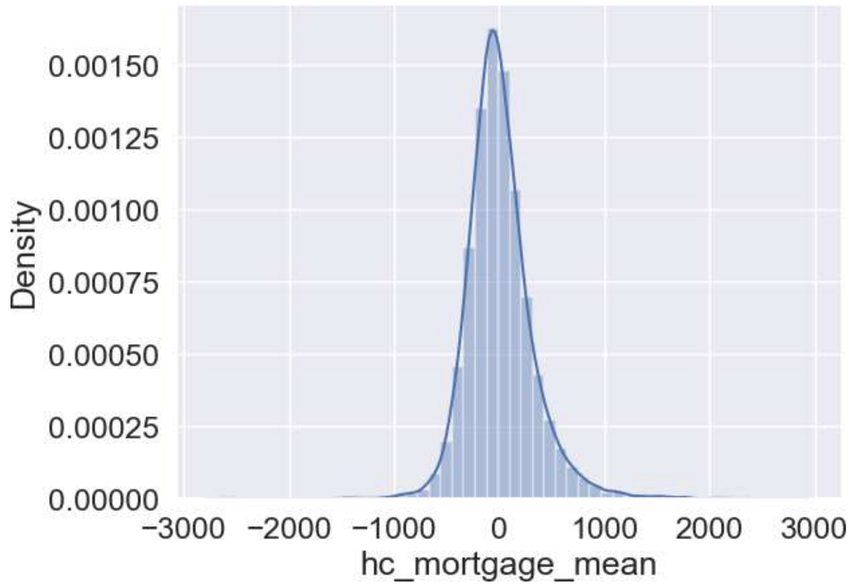
In [102…

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

```
C:\ProgramData\Anaconda3\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).
```

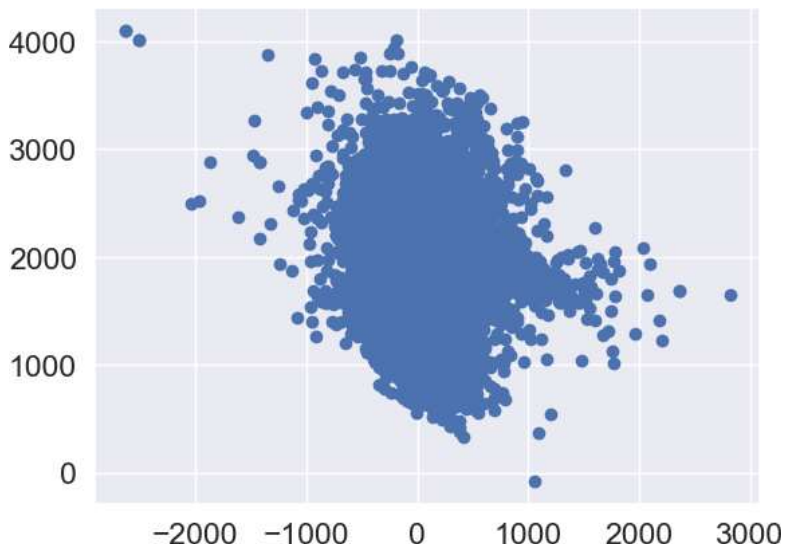
Out[102]:

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



In [103…
```
plt.scatter(residuals,y_pred) # Same variance and residuals does not have correlation with predictor
# Independance of residuals
```

Out[103]: <matplotlib.collections.PathCollection at 0x21de14a1820>



In [ ]: