CAPSTONE PROJECT 1:- Real Estate Case Study

Submitted by:- Ayub S Bijapur

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

1.Import data

In [2]:
```
df_train = pd.read_csv("train.csv")
df_train.head()
```

Out[2]:

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

5 rows × 80 columns

In [3]:
```
df_test = pd.read_csv("test.csv")
df_test.head()
```

Out[3]:

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

5 rows × 80 columns

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.describe()
```

Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_co
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	27321.000
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	596.5076
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	232.4974
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	201.0000
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	405.0000
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	614.0000
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	801.0000
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	989.0000

8 rows × 74 columns

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

Out[9]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_co
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	11709.000
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	593.5985
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	232.0742
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	201.0000
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	404.0000
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	612.0000
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	787.0000
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	989.0000

8 rows × 74 columns

In [10]:
```
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 [11]:
```
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

UID is unique userID value in the train and test dataset. So an index can be created from the UID feature

In [12]:
```
#Set the DataFrame index using existing columns.
df_train.set_index(keys=['UID'],inplace=True)
df_test.set_index(keys=['UID'],inplace=True)
```

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

Out[13]:

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

2 rows × 79 columns

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

Out[14]:

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

2 rows × 79 columns

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

In [15]:
```
#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 mi
missing_values_df_train.sort_values(by=['Percantage of missing values'],inplace=Tru
missing_values_df_train[missing_values_df_train['Percantage of missing values'] >0
#BLOCKID can be dropped, since it is 100%missing values
```

Out[15]:

	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 [16]:
```
#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 miss
missing_values_df_test.sort_values(by=['Percantage of missing values'],inplace=True
missing_values_df_test[missing_values_df_test['Percantage of missing values'] >0][
#BLOCKID can be dropped, since it is 43%missing values
```

Out[16]:

	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 [17]:
```
df_train.drop(columns=['BLOCKID','SUMLEVEL'],axis=1,inplace=True)
#SUMLEVEL doest not have any predictive power and no variance
```

In [18]:
```
df_test .drop(columns=['BLOCKID','SUMLEVEL'],axis=1,inplace=True)
#SUMLEVEL doest not have any predictive power
```

In [19]:
```
# Imputing  missing values with mean
missing_train_cols=[]
for col in df_train.columns:
    if df_train[col].isna().sum() !=0:
         missing_train_cols.append(col)
print(missing_train_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples',
'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_3
5', 'rent_gt_40', 'rent_gt_50', 'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_wei
ght', '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_media
n', 'hc_stdev', 'hc_samples', 'hc_sample_weight', 'home_equity_second_mortgage',
'second_mortgage', 'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cd
f', 'debt_cdf', 'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mea
n', 'male_age_median', 'male_age_stdev', 'male_age_sample_weight', 'male_age_sampl
es', 'female_age_mean', 'female_age_median', 'female_age_stdev', 'female_age_sampl
e_weight', 'female_age_samples', 'pct_own', 'married', 'married_snp', 'separated',
'divorced']
```

In [20]:
```
# Imputing  missing values with mean
missing_test_cols=[]
for col in df_test.columns:
    if df_test[col].isna().sum() !=0:
         missing_test_cols.append(col)
print(missing_test_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples',
'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_3
5', 'rent_gt_40', 'rent_gt_50', 'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_wei
ght', '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_media
n', 'hc_stdev', 'hc_samples', 'hc_sample_weight', 'home_equity_second_mortgage',
'second_mortgage', 'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cd
f', 'debt_cdf', 'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mea
n', 'male_age_median', 'male_age_stdev', 'male_age_sample_weight', 'male_age_sampl
es', 'female_age_mean', 'female_age_median', 'female_age_stdev', 'female_age_sampl
e_weight', 'female_age_samples', 'pct_own', 'married', 'married_snp', 'separated',
'divorced']
```

In [21]:
```
# 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 [22]:
```
# 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 [23]:
```
df_train.isna().sum().sum()
```

Out[23]:
```
0
```

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

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

Exploratory Data Analysis (EDA):

4.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 [25]:
```
from pandasql import sqldf
q1 = "select place,pct_own,second_mortgage,lat,lng from df_train where pct_own >0.1
pysqldf = lambda q: sqldf(q, globals())
df_train_location_mort_pct=pysqldf(q1)
```

In [26]:
```
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
```

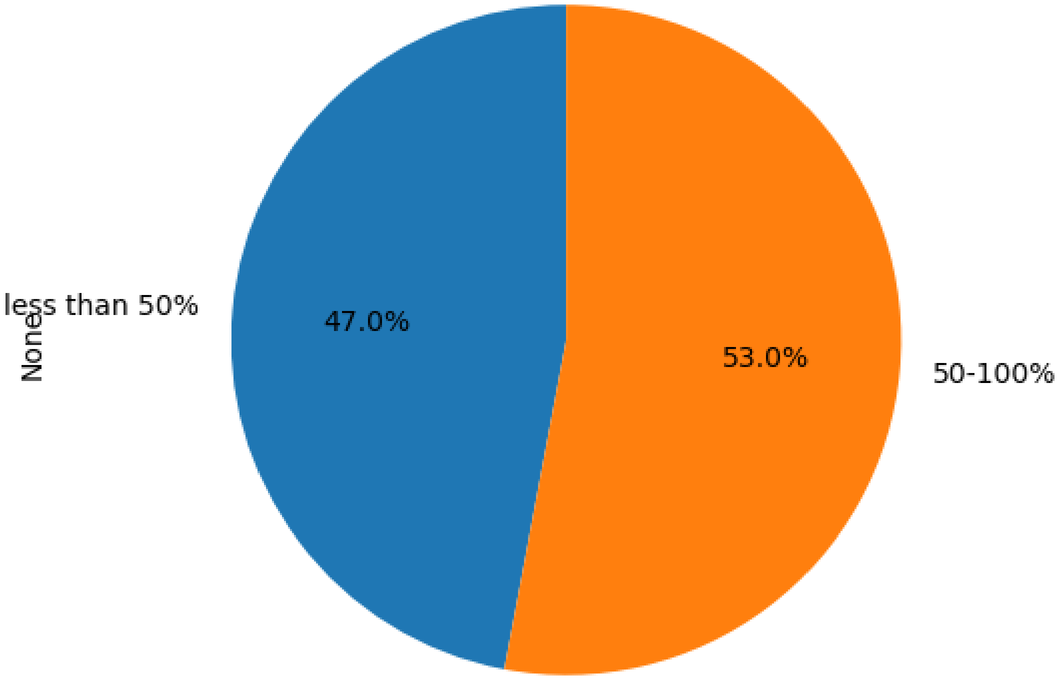
In [28]:

```
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 owne
fig.show()
```

b) 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 over

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

In [30]:
```
df_train['bins'] = pd.cut(df_train['bad_debt'],bins=[0,0.10,1], labels=["less than
df_train.groupby(['bins']).size().plot(kind='pie',subplots=True,startangle=90, auto
plt.axis('equal')
plt.show()
```



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

In [31]:
```
cols=[]
df_train.columns
```

Out[31]:
```
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')
```

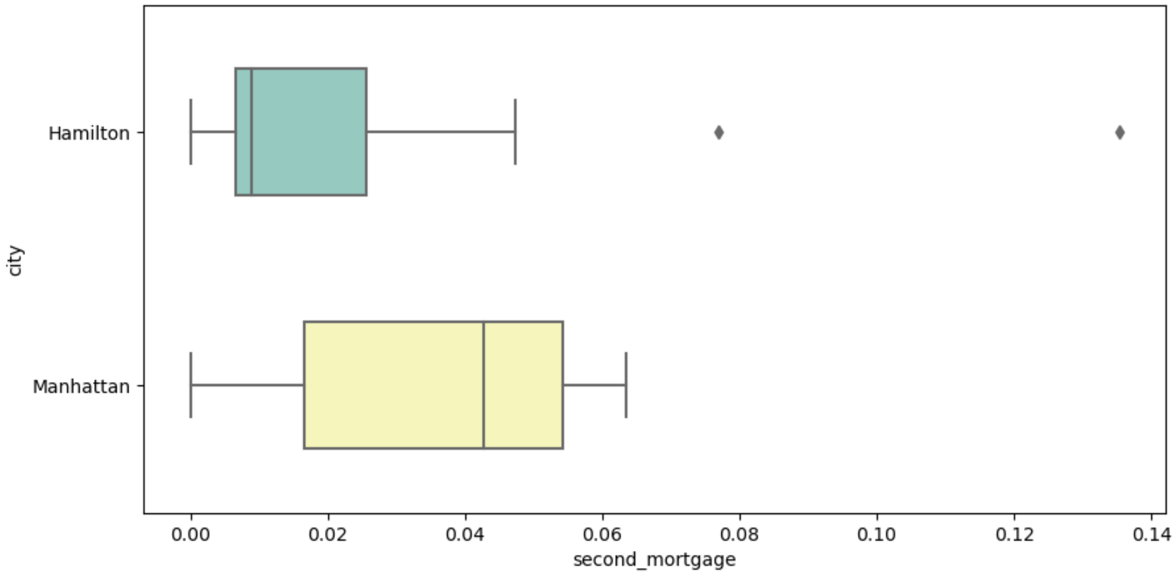
In [32]:
```
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
df_box_hamilton=df_train.loc[df_train['city'] == 'Hamilton']
df_box_manhattan=df_train.loc[df_train['city'] == 'Manhattan']
df_box_city=pd.concat([df_box_hamilton,df_box_manhattan])
df_box_city.head(4)
```

Out[32]:

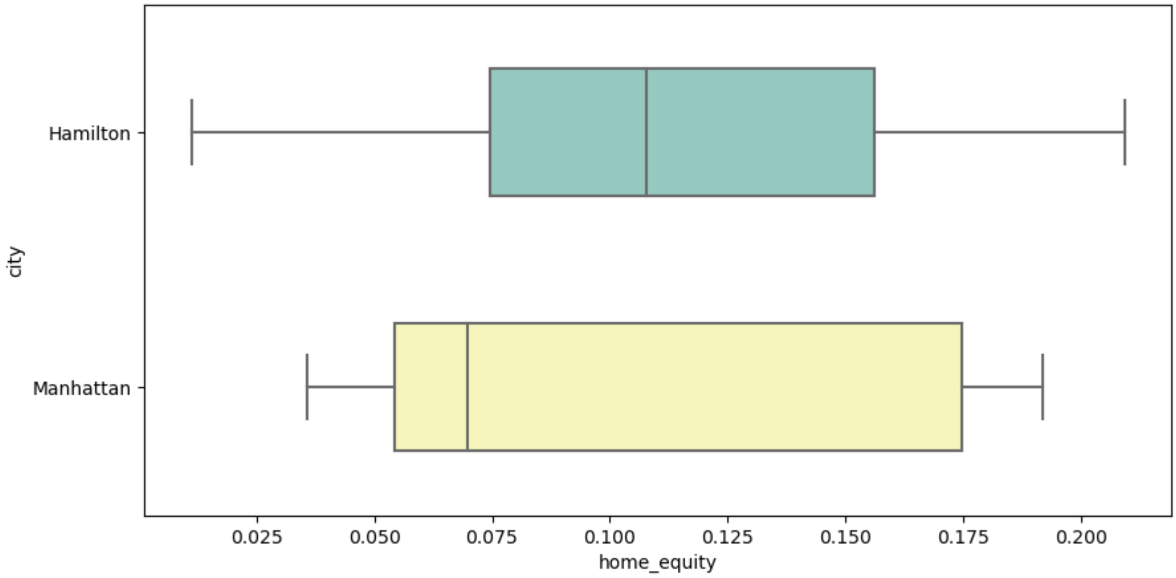
UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	1334€
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	861C
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	4501E
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	3974€

4 rows × 79 columns

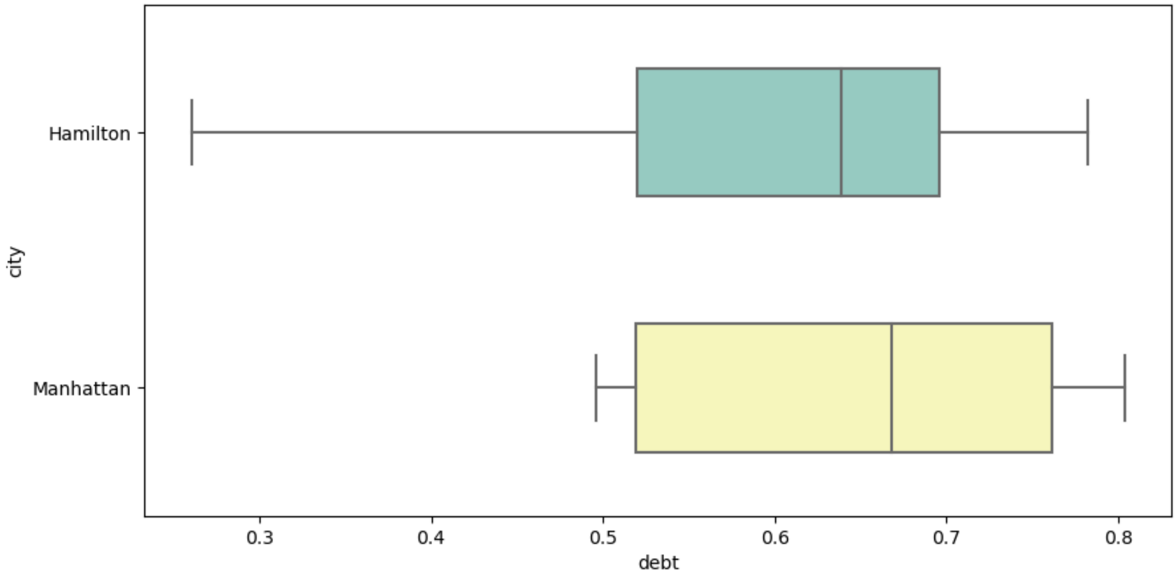
In [33]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.5,palette="Set3'
plt.show()
```



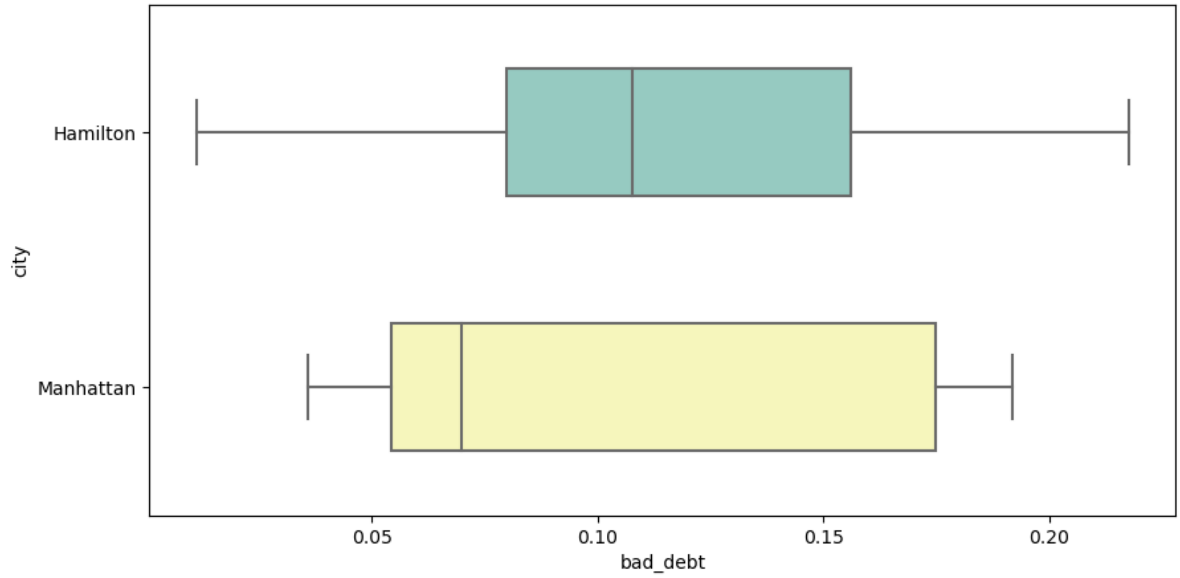
In [34]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



In [35]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```



In [36]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='bad_debt', y='city',width=0.5,palette="Set3")
plt.show()
```

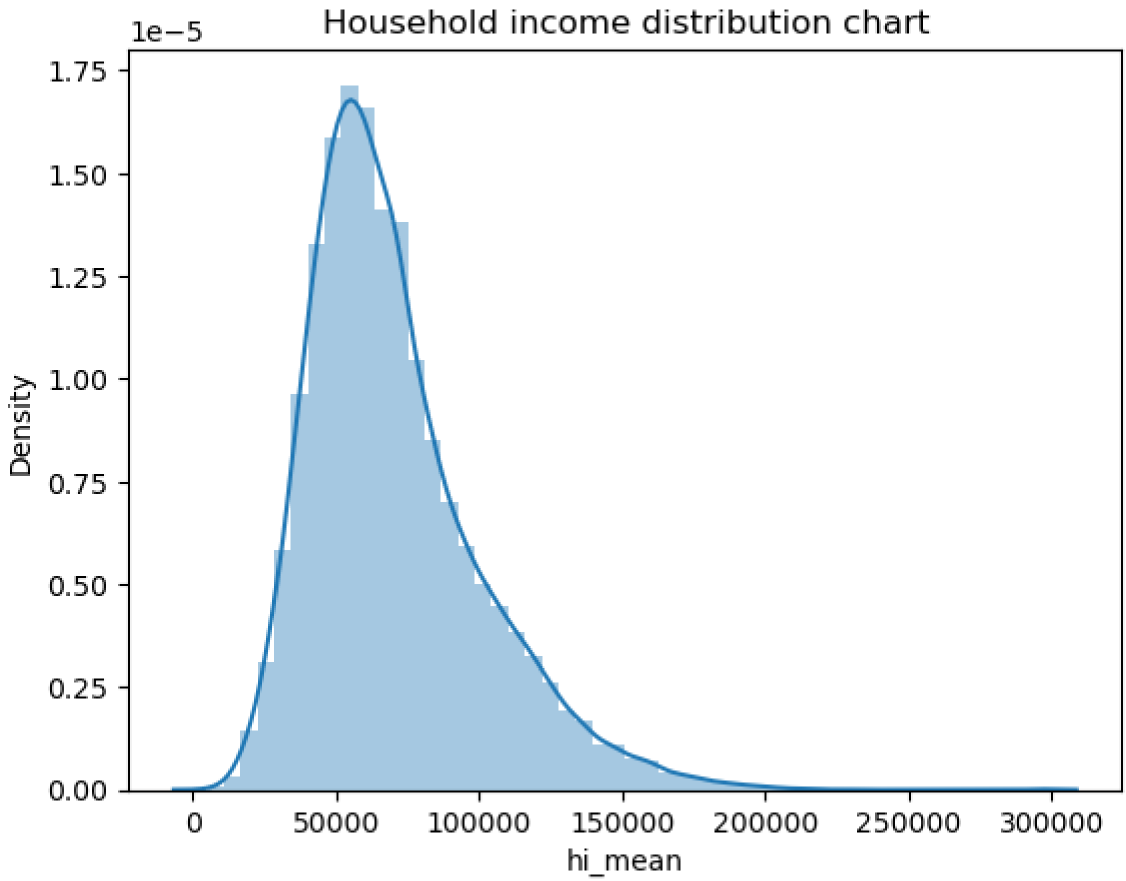


Manhattan has higher metrics compared to Hamilton

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

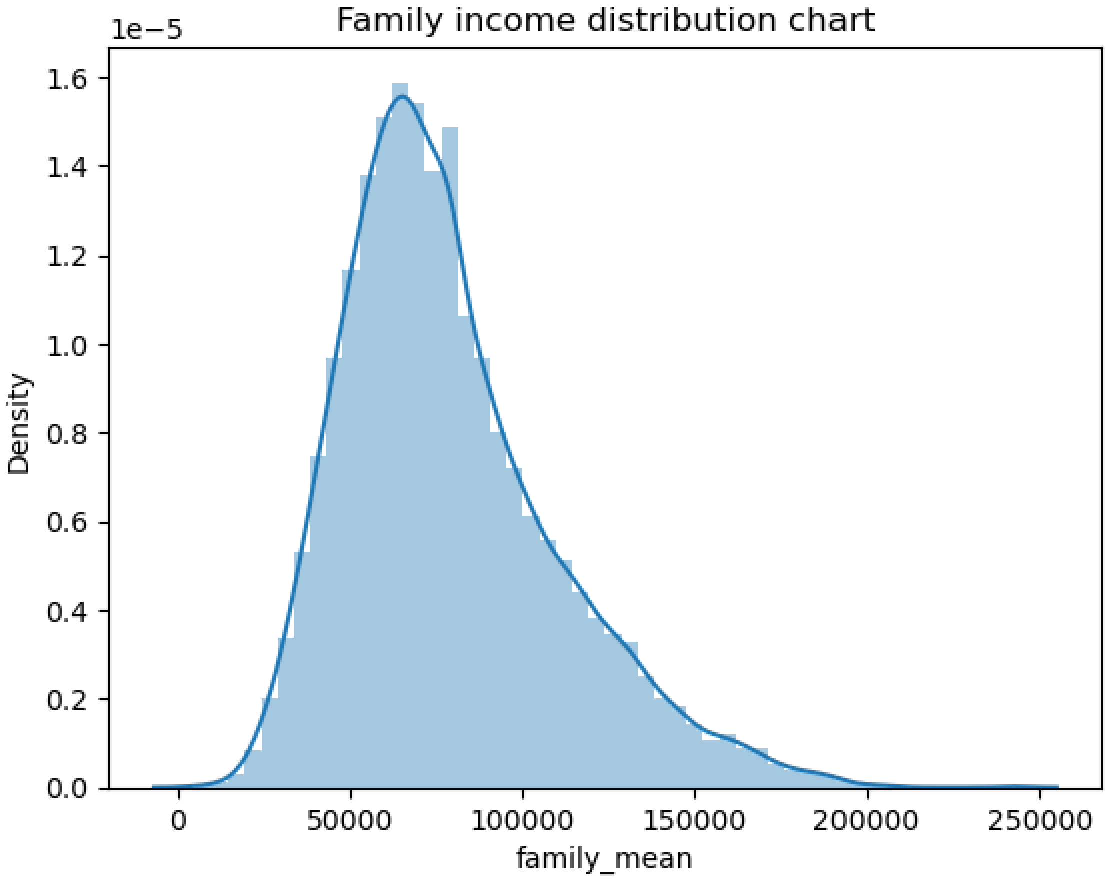
In [37]:
```
sns.distplot(df_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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



In [38]:
```
sns.distplot(df_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

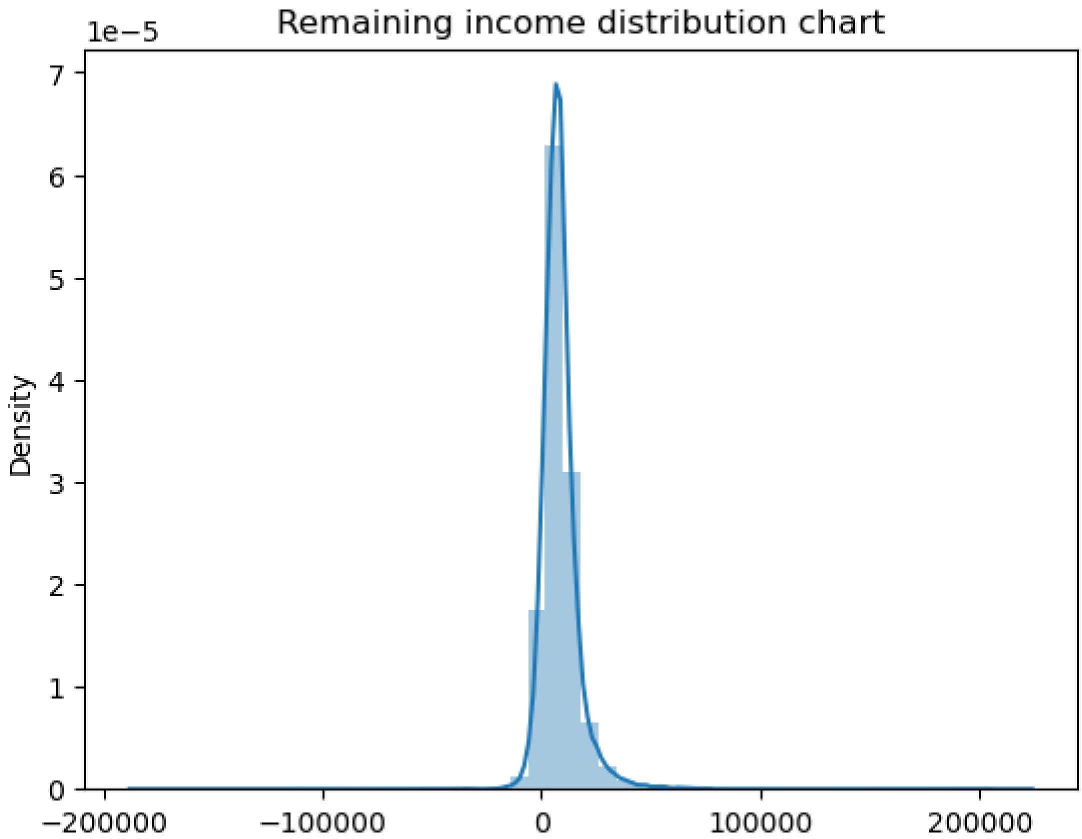
```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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



In [39]:

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

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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

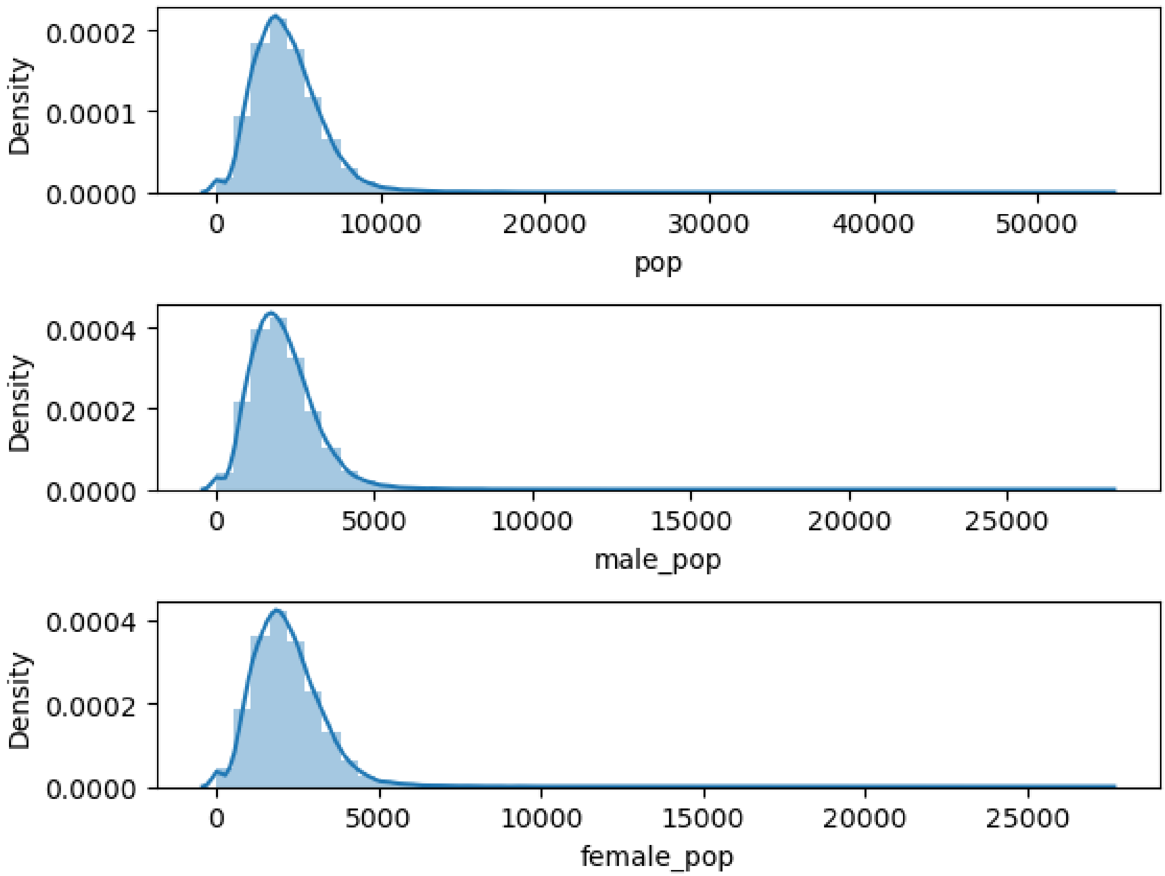


Exploratory Data Analysis (EDA):

1. 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):

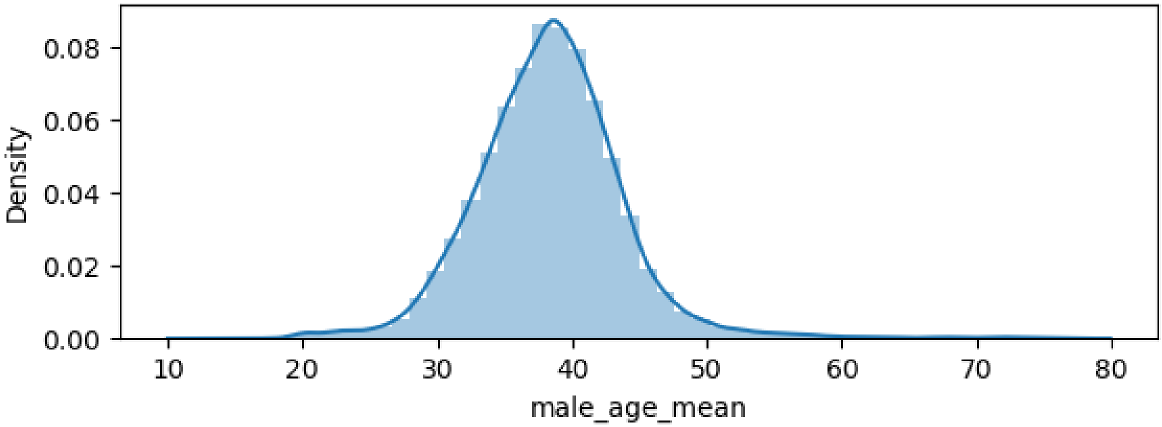
In [40]:
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2,ax3)=plt.subplots(3,1)
sns.distplot(df_train['pop'],ax=ax1)
sns.distplot(df_train['male_pop'],ax=ax2)
sns.distplot(df_train['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

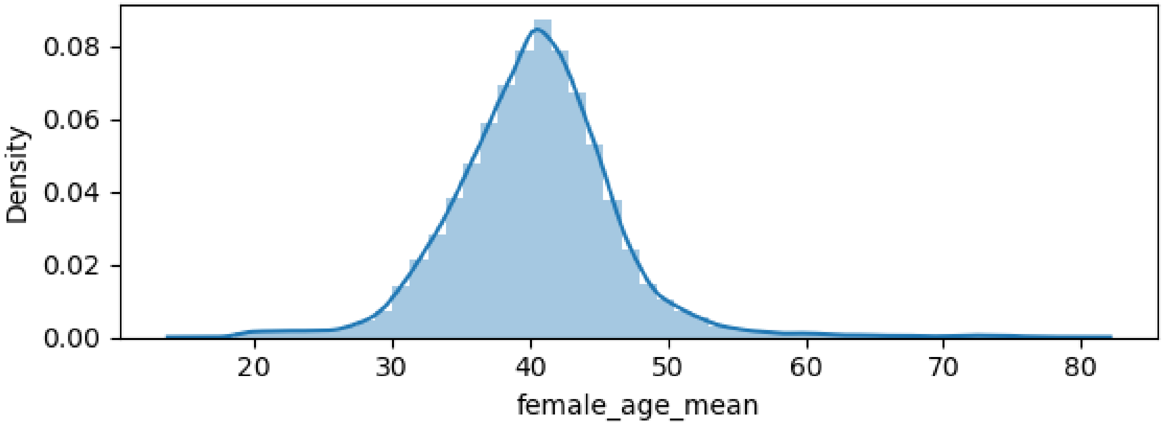
```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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

/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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

/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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



In [41]:
```
#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()
```

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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

/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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





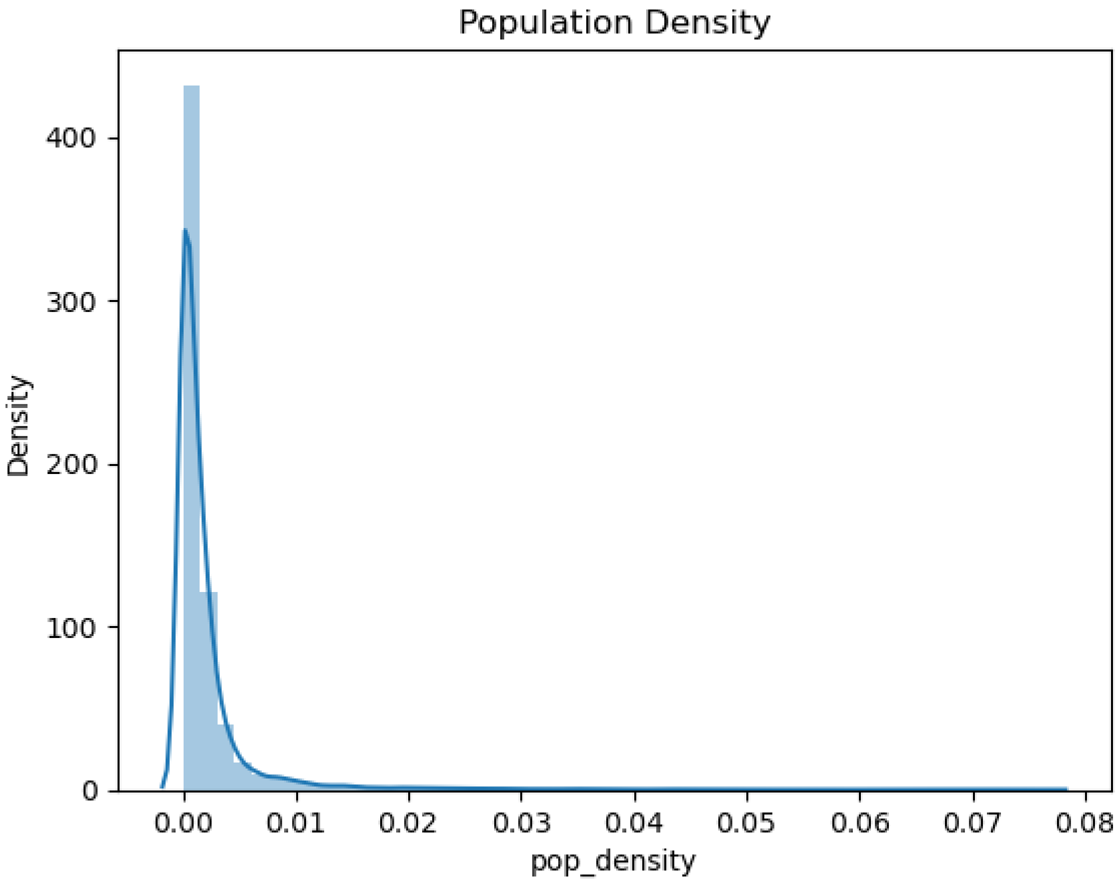
a) Use pop and ALand variables to create a new field called population density

In [42]:
```
df_train['pop_density']=df_train['pop']/df_train['ALand']
```

In [43]:
```
df_test['pop_density']=df_test['pop']/df_test['ALand']
```

In [44]:
```
sns.distplot(df_train['pop_density'])
plt.title('Population Density')
plt.show() # Very less density is noticed
```

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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



b) 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 [45]:
```
df_train['age_median']=(df_train['male_age_median']+df_train['female_age_median'])/
df_test['age_median']=(df_test['male_age_median']+df_test['female_age_median'])/2
```

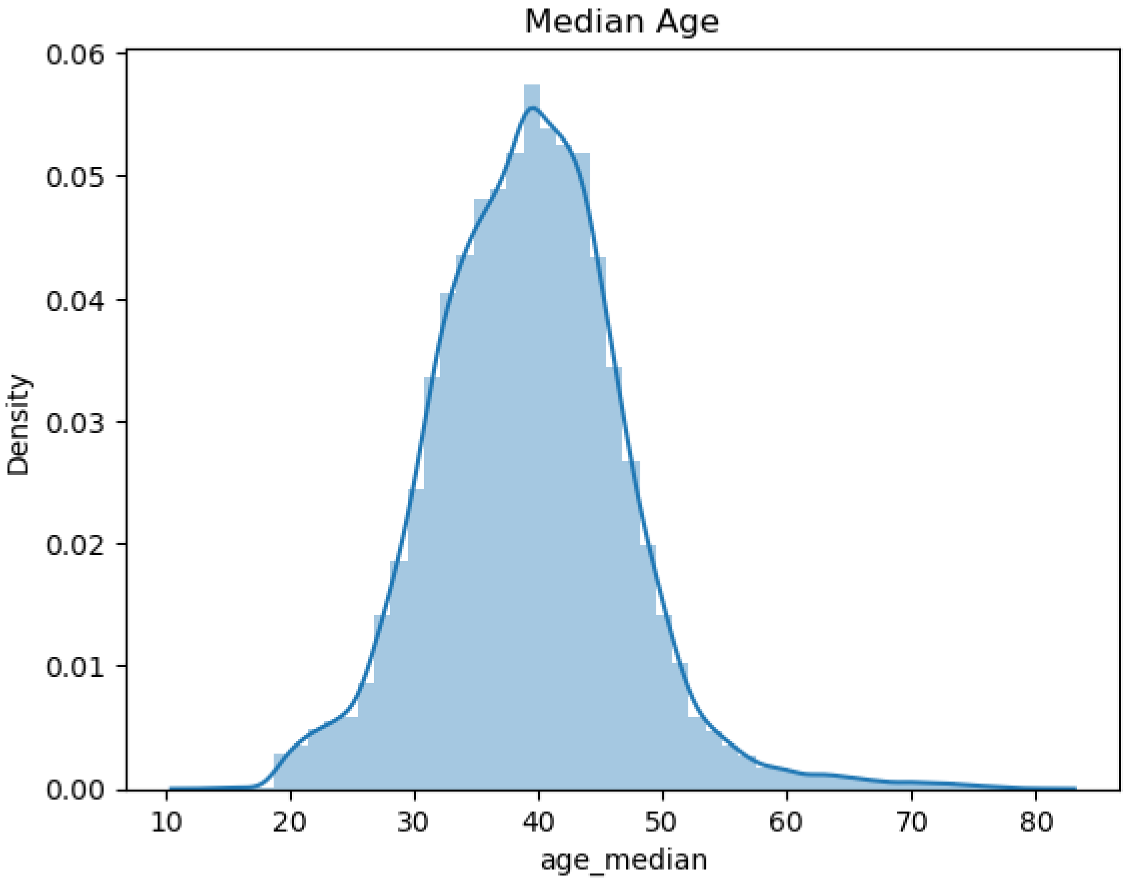
In [46]:
```
df_train[['male_age_median','female_age_median','male_pop','female_pop','age_media
```

Out[46]:

	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

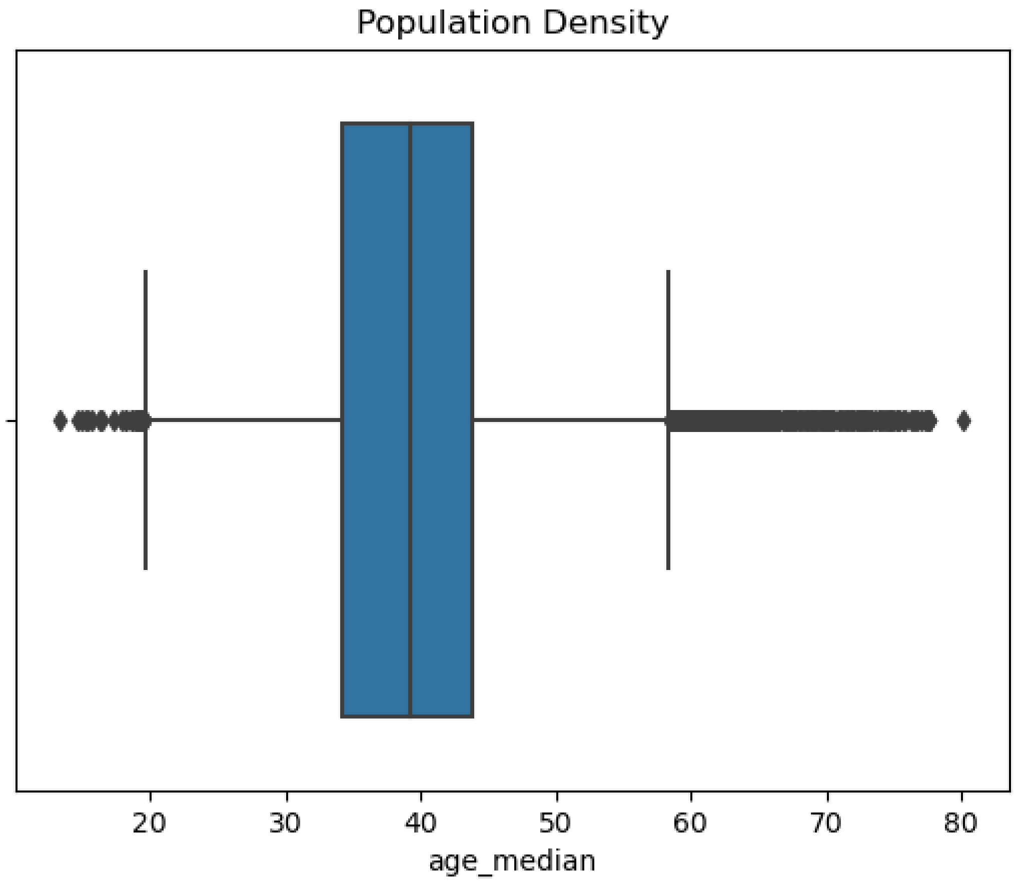
In [47]:
```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
# Age of population is mostly between 20 and 60
# Majority are of age around 40
# Median age distribution has a gaussian distribution
# Some right skewness is noticed
```

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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



In [48]:
```python
sns.boxplot(df_train['age_median'])
plt.title('Population Density')
plt.show()
```

```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/_decorators.py:36: Fut
ureWarning:

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



2. 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 [49]:
```
df_train['pop'].describe()
```

Out[49]:
```
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 [50]:
```
df_train['pop_bins']=pd.cut(df_train['pop'],bins=5,labels=['very low','low','mediu
```

In [51]:
```
df_train[['pop','pop_bins']]
```

Out[51]:

	pop	pop_bins
UID		
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 [52]:
```
df_train['pop_bins'].value_counts()
```

Out[52]:
```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

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

In [53]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

Out[53]:

	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 [54]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "ı
```

Out[54]:

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

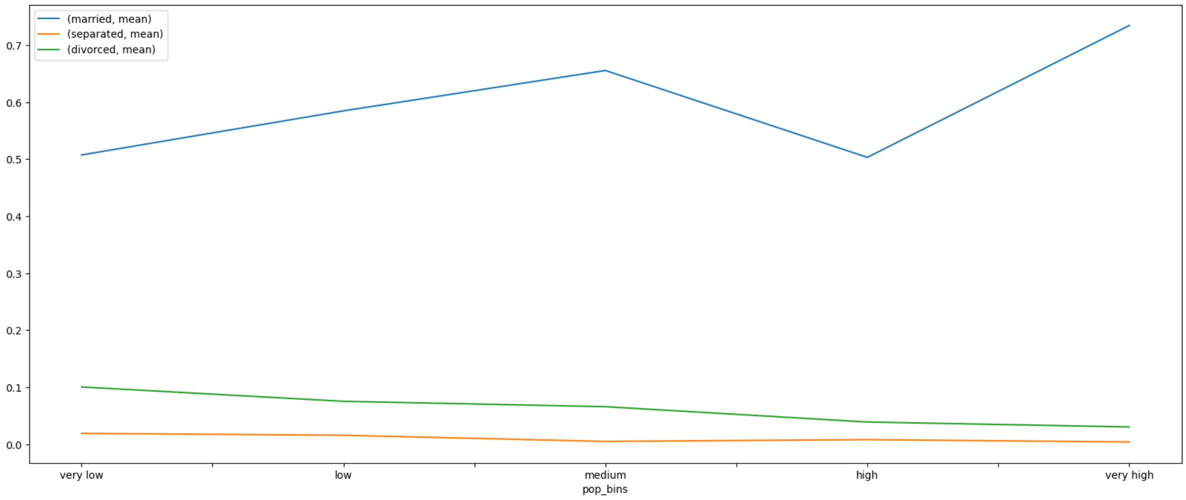
1.Very high population group has more married people and less percantage of separated and divorced couples

2.In very low population groups, there are more divorced people

b) Visualize using appropriate chart type

In [55]:
```
plt.figure(figsize=(10,5))
pop_bin_married=df_train.groupby(by='pop_bins')[['married','separated','divorced']
pop_bin_married.plot(figsize=(20,8))
plt.legend(loc='best')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



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

In [56]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

Out[56]:

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

In [57]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

Out[57]:

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

In [58]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

Out[58]:
```
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 [59]:
```
#overall level rent as a percentage of income
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

Out[59]:
```
0.013358170721473864
```

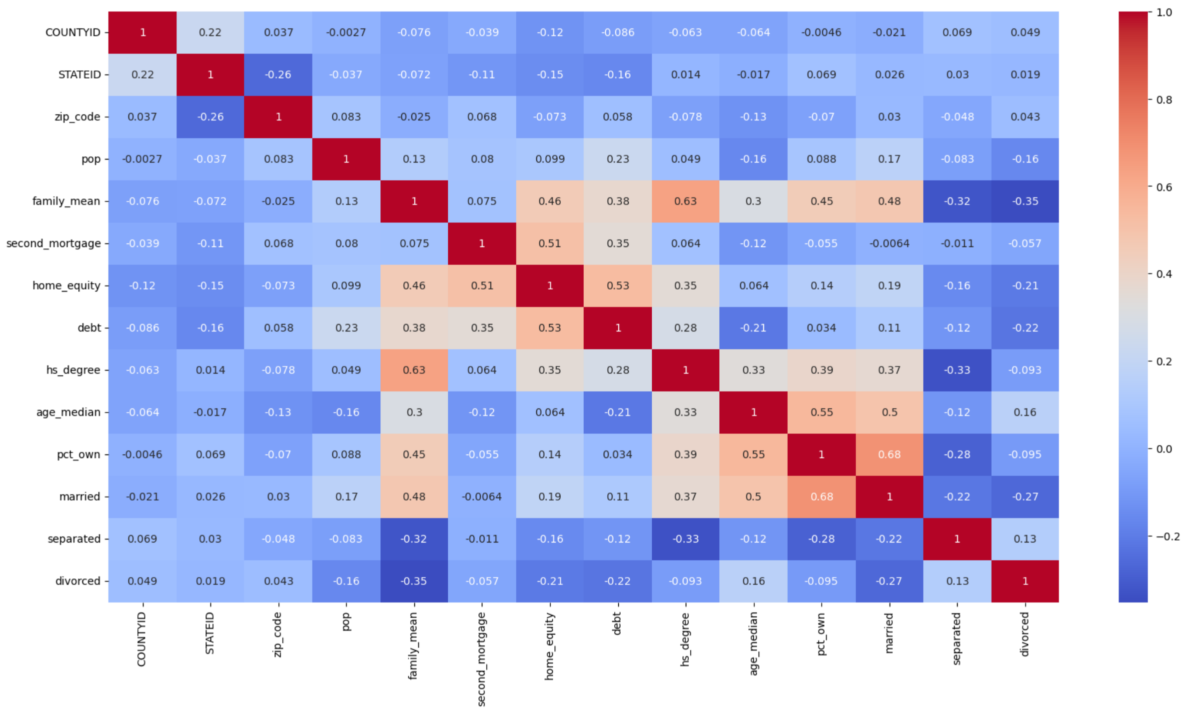
4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

In [60]:
```
df_train.columns
```

Out[60]:
```
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 [61]:
```python
cor=df_train[['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']].corr()
```

In [62]:
```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

Data Pre-processing:

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 [63]:
```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [64]:
```
fa=FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

Out[64]:
```
array([[-1.12589168e-01,  1.95646473e-02, -2.39331081e-02,
        -6.27632630e-02,  4.23474744e-02],
       [-1.10186764e-01,  1.33506220e-02,  2.79651243e-02,
        -1.49825864e-01,  1.10838806e-01],
       [-8.28678626e-02,  5.16372371e-02, -1.36451864e-01,
        -4.98918600e-02, -1.04024838e-01],
       [ 1.80961136e-02,  1.92013749e-02,  5.81329912e-03,
         2.64842745e-02, -6.12442620e-03],
       [ 9.02324727e-02, -9.72544310e-02, -6.54601334e-02,
        -1.33145905e-01, -1.48594602e-01],
       [-1.07335697e-02, -4.12376813e-02,  1.45853480e-01,
         8.80433214e-03,  1.08227564e-01],
       [-4.28796975e-02, -2.09780214e-02,  3.66726857e-02,
        -9.45597367e-02,  5.91380506e-02],
       [-2.44243043e-03, -1.53245408e-02, -2.68300838e-03,
        -4.52473026e-02,  2.37240645e-02],
       [ 7.92164309e-02,  9.57453307e-01, -8.71151634e-02,
        -6.59923727e-03, -3.97273224e-02],
       [ 7.39808182e-02,  9.18750508e-01, -1.08834838e-01,
        -2.79371563e-02, -3.93153685e-02],
       [ 8.06598878e-02,  9.47839209e-01, -6.08006517e-02,
         1.53627100e-02, -3.86977301e-02],
       [ 7.70052115e-01,  9.84675280e-03, -3.71249731e-02,
         1.14949040e-01, -1.23784691e-01],
       [ 7.18615876e-01,  6.24980344e-03, -4.59787391e-02,
         1.09109690e-01, -1.35301916e-01],
       [ 7.07647251e-01,  2.46625398e-02, -1.00860886e-02,
         1.04472489e-01,  7.72381273e-02],
       [-1.34545479e-01,  3.36809304e-01, -4.87894982e-01,
        -4.15446298e-02,  3.17608549e-01],
       [ 2.31079719e-01,  4.37729796e-01, -6.40209229e-01,
        -2.52311077e-02,  3.47216239e-01],
       [-4.52068105e-02,  3.51263841e-02,  3.07536976e-02,
         4.44793489e-01, -1.63273400e-01],
       [-2.50717045e-02,  1.70166794e-02,  4.57227166e-02,
         6.76083874e-01, -1.55256754e-01],
       [-3.90694430e-02, -1.67460870e-02,  8.13962648e-02,
         8.36389105e-01, -9.18259738e-02],
       [-5.14161938e-02, -3.57207133e-02,  1.10795164e-01,
         9.25123732e-01, -4.44866412e-02],
       [-6.08589976e-02, -4.41860612e-02,  1.35794020e-01,
         9.53019912e-01, -2.21548589e-02],
       [-4.57771169e-02, -5.25526111e-02,  1.41019853e-01,
         9.32702589e-01, -5.77563331e-07],
       [-4.19486035e-02, -5.90387634e-02,  1.28851777e-01,
         8.87316677e-01,  1.05894354e-02],
       [-2.47894631e-02, -7.29670549e-02,  9.41510411e-02,
         7.79023667e-01,  2.95352863e-02],
       [ 2.12258453e-01,  4.65992340e-01, -6.14495953e-01,
        -2.47660051e-02,  3.66644524e-01],
       [ 2.33057257e-01,  4.47057851e-01, -6.28263442e-01,
        -2.71547799e-02,  3.43419628e-01],
       [ 7.85157099e-01,  4.91249250e-02,  1.44540487e-01,
        -2.05217630e-01, -1.54523367e-01],
       [ 7.10324868e-01,  4.99730429e-02,  1.32239995e-01,
        -2.19171855e-01, -2.10505576e-01],
       [ 8.61780950e-01,  4.35044832e-02,  1.65839097e-01,
        -1.19850814e-01,  3.16733597e-02],
       [-2.23443268e-01,  8.46259548e-01, -4.61177284e-02,
         6.88599220e-02,  2.27742319e-01],
       [ 1.43837565e-01,  9.53197441e-01,  2.27887416e-02,
        -4.57890474e-02,  1.00796460e-01],
       [ 8.30286493e-01,  3.42026000e-02,  1.61106002e-01,
        -2.04570328e-01, -7.48710501e-02],
```

```
[ 7.94476579e-01,  2.83818591e-02,  1.51219549e-01,
 -2.07681494e-01, -9.12497137e-02],
[ 8.11481659e-01,  4.32314894e-02,  1.43645559e-01,
 -1.07778262e-01,  5.79540197e-02],
[-3.37741899e-01,  8.64927608e-01,  3.58933641e-02,
  9.07183932e-02,  4.46327272e-02],
[ 5.03572673e-02,  9.35515325e-01,  1.51475395e-01,
 -2.51501261e-02, -9.34471589e-02],
[ 9.78242233e-01, -3.31490295e-02, -1.05261175e-01,
  4.50364242e-02,  7.37361956e-02],
[ 9.59137174e-01, -3.90023011e-02, -1.20630338e-01,
  4.52591412e-02,  6.64877103e-02],
[ 8.14087183e-01,  2.23057256e-03,  7.66518490e-02,
  2.02747449e-02,  1.27634826e-01],
[-4.15353978e-01,  7.18339580e-01,  3.40068066e-01,
 -7.18402751e-02, -2.77950503e-01],
[ 7.64912569e-02,  7.24900638e-01,  2.74193216e-01,
 -4.83952593e-02, -3.52988295e-01],
[ 9.10390848e-01, -5.36541221e-02, -4.68641899e-02,
 -7.64182970e-04,  1.63870449e-01],
[ 8.73011851e-01, -5.30302303e-02, -5.89943146e-02,
 -1.58989810e-03,  1.52417529e-01],
[ 7.55087669e-01, -3.56133794e-03,  5.39542523e-02,
  4.24181518e-03,  2.58043480e-01],
[-1.23469883e-01,  6.07438118e-01,  6.33039205e-01,
 -2.14798827e-02,  2.47973912e-01],
[-3.42866886e-01,  5.59526274e-01,  5.88212998e-01,
 -2.51533511e-02,  2.18419886e-01],
[-1.60867227e-01, -1.53062640e-02, -1.57026582e-01,
  1.09243755e-01, -6.61660846e-01],
[-1.37306742e-01, -2.17250612e-02, -1.58408924e-01,
  1.25156181e-01, -6.71630758e-01],
[ 2.45096185e-01, -2.54584594e-02, -2.66691412e-02,
  9.53148433e-02, -6.42510831e-01],
[ 2.03988660e-01,  7.85172838e-02, -3.01656215e-01,
  2.28379441e-02, -6.29223343e-01],
[ 1.08926076e-01, -6.34332399e-02, -3.36565249e-02,
 -9.49480406e-02,  6.81473821e-01],
[-2.63787628e-01, -6.43281164e-03, -3.58792183e-02,
 -9.37962372e-02,  6.47816982e-01],
[-2.15717041e-01, -7.36588946e-02,  3.50113231e-01,
 -1.95201594e-02,  6.36783774e-01],
[ 3.94306145e-01,  6.09565684e-02,  2.55337859e-01,
 -2.20362094e-01, -1.84248070e-01],
[ 4.07877890e-01,  6.27256523e-02,  2.23926908e-01,
 -2.10028737e-01, -1.71989219e-01],
[ 3.53156877e-01,  5.36715662e-02,  2.69603568e-01,
 -2.16933220e-01, -1.80072063e-01],
[ 2.33537261e-01, -4.91732966e-02,  8.14450766e-01,
  9.36688826e-02,  3.27131932e-01],
[ 2.40298209e-01, -3.38140105e-02,  8.31496963e-01,
  7.52417537e-02,  2.46323612e-01],
[-6.71839466e-02,  6.58504547e-02,  5.86207684e-01,
  8.72955212e-02,  9.12541435e-02],
[ 5.59835523e-02,  8.17918693e-01, -1.78458348e-01,
 -1.55949404e-02, -3.34299788e-02],
[ 7.16426383e-02,  9.23428540e-01, -1.07142695e-01,
 -2.78635352e-02, -4.35991157e-02],
[ 1.92496949e-01, -4.75870395e-02,  8.03173180e-01,
  1.43492705e-01,  3.33862161e-01],
[ 1.87644433e-01, -3.29941019e-02,  8.58024474e-01,
  1.31329947e-01,  2.55679728e-01],
[-1.02263653e-01,  6.03984268e-02,  4.72982253e-01,
  7.36848375e-02,  1.12273916e-01],
```

```
       [ 6.14776635e-02,  8.77962755e-01, -1.50410288e-01,
         2.20991060e-02, -4.17158213e-02],
       [ 7.83728204e-02,  9.54508785e-01, -5.91095915e-02,
         1.64800945e-02, -4.32591020e-02],
       [-3.24381831e-02,  1.11167162e-01,  7.84467387e-01,
        -4.37718583e-02, -2.80931215e-01],
       [ 1.76682383e-01,  1.90494242e-01,  5.61405509e-01,
        -1.20746162e-01, -1.32570792e-01],
       [-6.37386607e-02, -7.03047925e-02, -2.68934071e-01,
         1.28589792e-01,  1.88507859e-01],
       [-1.56051269e-01, -7.08033944e-02, -1.45964505e-01,
         1.24253734e-01,  1.46293119e-01],
       [-3.56716288e-01, -5.29910735e-02,  1.47771600e-01,
         2.87196156e-02,  1.13159584e-01],
       [ 2.42173825e-01, -2.86199134e-02, -3.25958343e-02,
         1.05027814e-01, -6.55406075e-01],
       [ 3.50196742e-01, -1.05016412e-02, -3.95274120e-01,
         5.92876774e-02,  2.91651778e-01],
       [ 2.25671545e-01, -3.42672758e-02,  8.92876608e-01,
         1.12426805e-01,  2.67065206e-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 [65]:
```
df_train.columns
```

Out[65]:
```
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 [66]:
```
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 [67]:
```
df_train['type'].unique()
```
Out[67]:
```
array([1, 2, 3, 4, 5, 6])
```

In [68]:
```
df_test.replace(type_dict,inplace=True)
```

In [69]:
```
df_test['type'].unique()
```
Out[69]:
```
array([4, 1, 6, 3, 5, 2])
```

In [70]:
```
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 [71]:
```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

In [72]:
```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

In [73]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error,accura
```

In [74]:
```
x_train.head()
```
Out[74]:

UID	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity
267822	53	36	13346	1	5230	67994.14790	0.02077	0.0891
246444	141	18	46616	1	2633	50670.10337	0.02222	0.0427
245683	63	18	46122	1	6881	95262.51431	0.00000	0.0951
279653	127	72	927	2	2700	56401.68133	0.01086	0.0108
247218	161	20	66502	1	5637	54053.42396	0.05426	0.0542

In [75]:
```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

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

In [76]:
```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```
Out[76]:
```
LinearRegression()
```

In [77]:
```
y_pred=linereg.predict(x_test_scaled)
```

In [78]:
```
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
```

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

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

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

In [79]:
```
state=df_train['STATEID'].unique()
state[0:5]
```

Out[79]:
```
array([36, 18, 72, 20,  1])
```

In [80]:
```
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
    print("Overall RMSE of linear regression model for state,",i,":-" ,np.sqrt(mean
    print("\n")
```

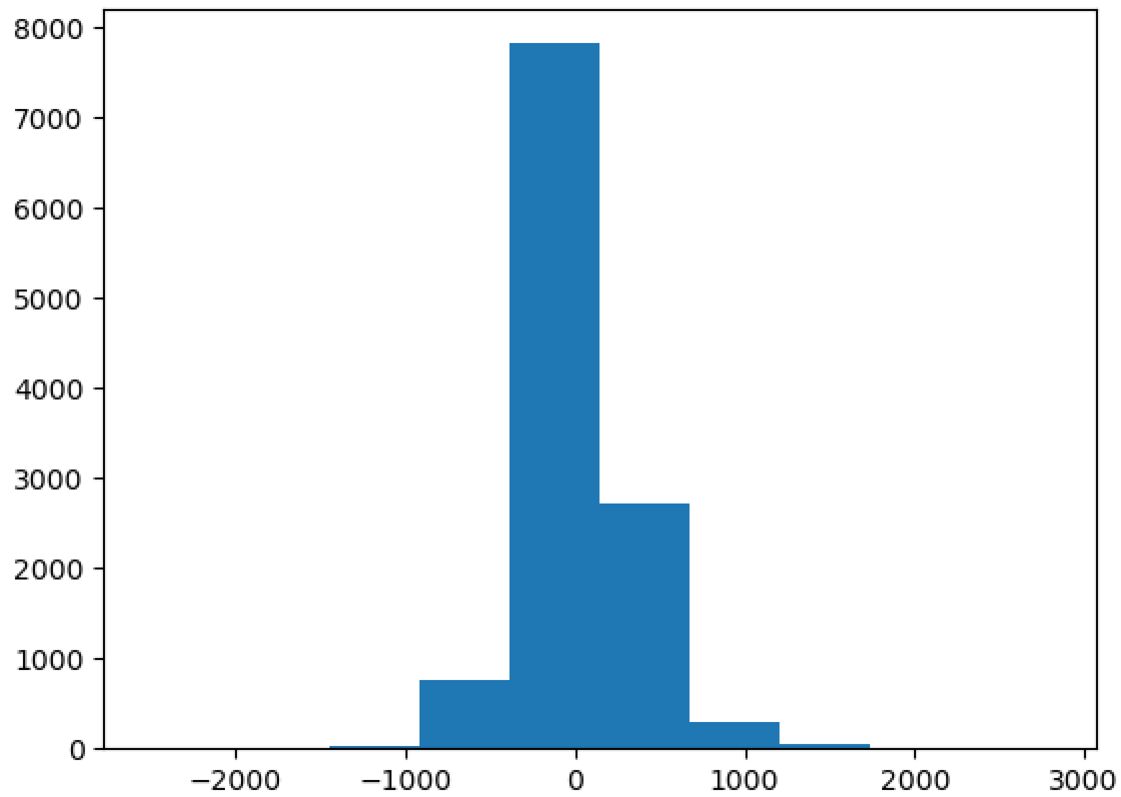
```
State ID- 20
Overall R2 score of linear regression model for state, 20 :- 0.6046603766461809
Overall RMSE of linear regression model for state, 20 :- 307.9718899931472


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8104382475484617
Overall RMSE of linear regression model for state, 1 :- 307.8275861848435


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.788744649785525
Overall RMSE of linear regression model for state, 45 :- 225.6961542072414
```

In [81]:
```
residuals=y_test-y_pred
residuals
```

Out[81]:
```
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 [82]:
```
plt.hist(residuals)
```

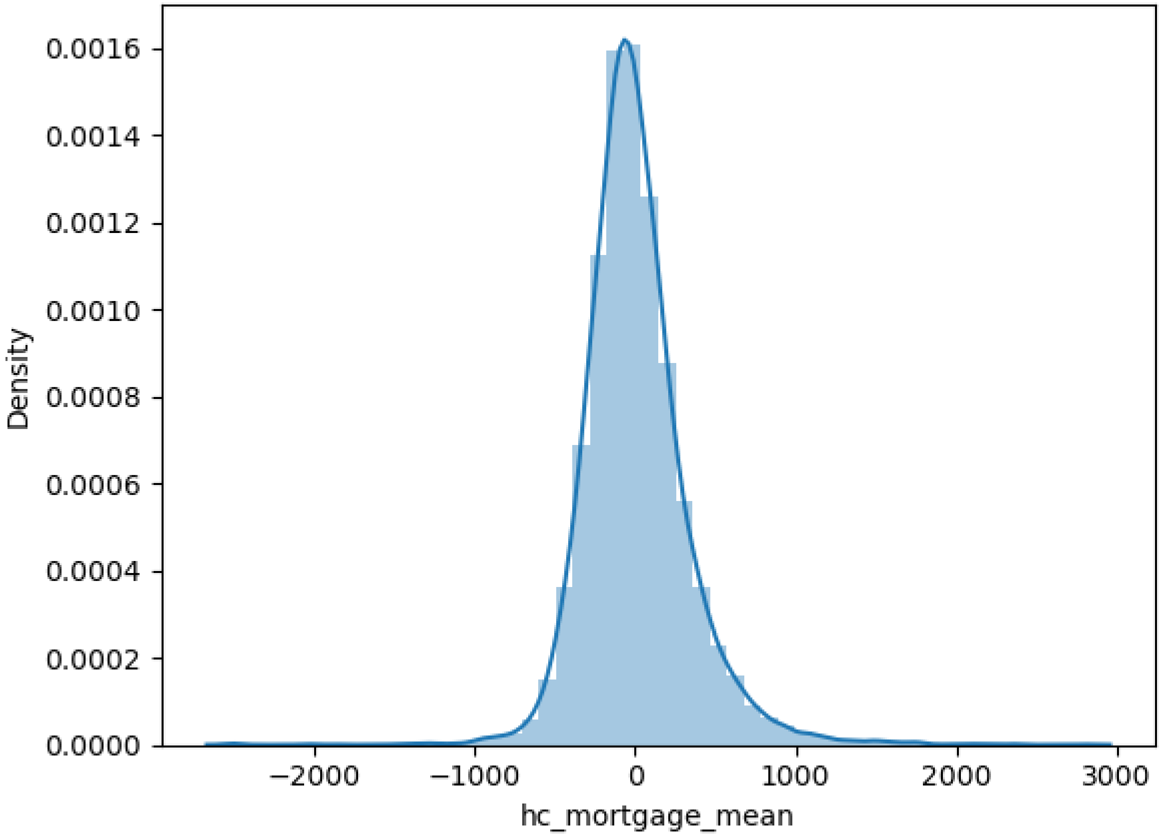
Out[82]:
```
(array([6.000e+00, 3.000e+00, 2.900e+01, 7.670e+02, 7.823e+03, 2.716e+03,
        3.010e+02, 4.900e+01, 1.200e+01, 3.000e+00]),
 array([-2515.04284233, -1982.92661329, -1450.81038425,  -918.69415521,
         -386.57792617,   145.53830287,   677.65453191,  1209.77076095,
         1741.88698999,  2274.00321903,  2806.11944807]),
 <BarContainer object of 10 artists>)
```



In [83]:
```
sns.distplot(residuals)
```

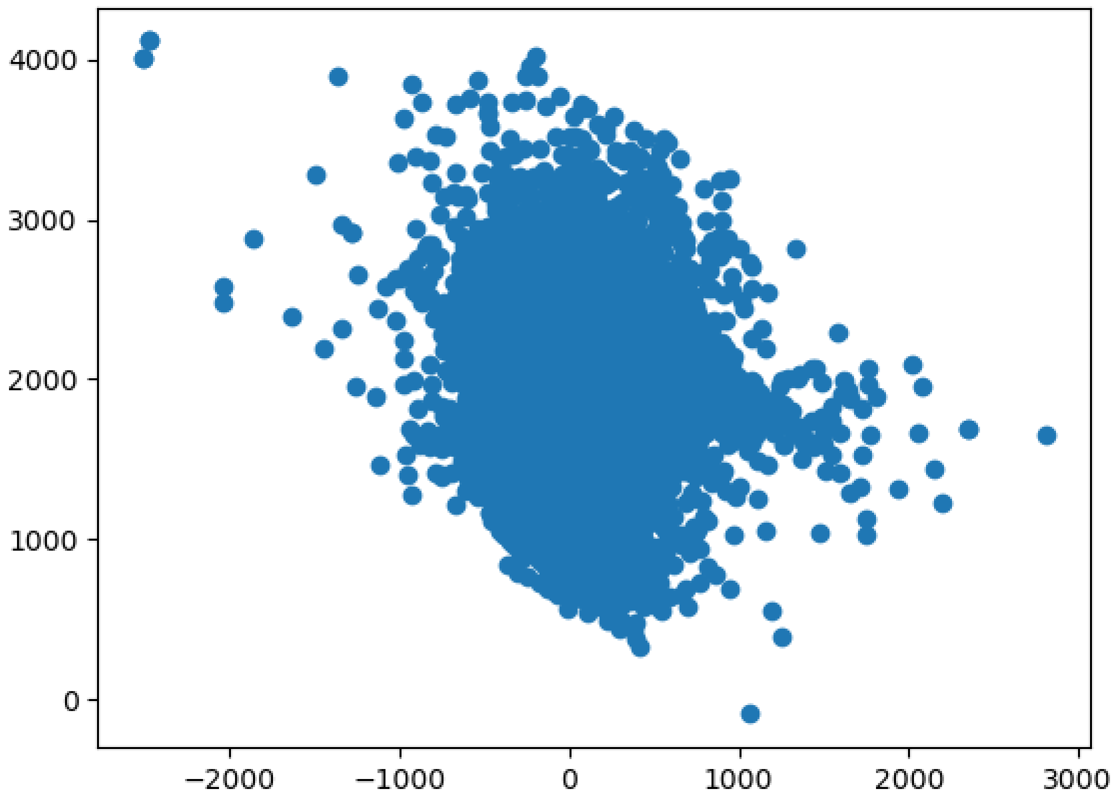
```
/home/spx072/anaconda3/lib/python3.10/site-packages/seaborn/distributions.py:2619:
FutureWarning:

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

Out[83]:
```
<AxesSubplot:xlabel='hc_mortgage_mean', ylabel='Density'>
```



In [84]:
```
plt.scatter(residuals,y_pred)
```

Out[84]: <matplotlib.collections.PathCollection at 0x7fc848a137f0>



In [ ]: