##CAPSTONE PROJECT-1 Real Estate Case Study

Submitted By - AbdulRajak J Bhavikatti

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as plt
```

##1.Import Data

```
data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

```
data_train.head()
```

```
      UID  BLOCKID  SUMLEVEL  COUNTYID  STATEID        state state_ab  
\
0  267822      NaN       140        53       36     New York       NY   

1  246444      NaN       140       141       18      Indiana       IN   

2  245683      NaN       140        63       18      Indiana       IN   

3  279653      NaN       140       127       72  Puerto Rico       PR   

4  247218      NaN       140       161       20       Kansas       KS   


         city           place   type  ... female_age_mean  
female_age_median  \
0    Hamilton        Hamilton   City  ...        44.48629  
45.33333   
1  South Bend        Roseland   City  ...        36.48391  
37.58333   
2    Danville        Danville   City  ...        42.15810  
42.83333   
3    San Juan        Guaynabo  Urban  ...        47.77526  
50.58333   
4   Manhattan  Manhattan City   City  ...        24.17693  
21.58333   

   female_age_stdev  female_age_sample_weight  female_age_samples  
pct_own  \
0          22.51276                 685.33845              2618.0  
0.79046   
1          23.43353                 267.23367              1284.0  
0.52483   
2          23.94119                 707.01963              3238.0  
0.85331
```

```
3          24.32015                362.20193             1559.0
0.65037
4          11.10484               1854.48652             3051.0
0.13046

   married  married_snp  separated  divorced
0  0.57851      0.01882    0.01240   0.08770
1  0.34886      0.01426    0.01426   0.09030
2  0.64745      0.02830    0.01607   0.10657
3  0.47257      0.02021    0.02021   0.10106
4  0.12356      0.00000    0.00000   0.03109

[5 rows x 80 columns]
```

```
data_test.head()
```

```
      UID  BLOCKID  SUMLEVEL  COUNTYID  STATEID         state state_ab
\
0  255504      NaN       140       163       26      Michigan       MI

1  252676      NaN       140         1       23         Maine       ME

2  276314      NaN       140        15       42  Pennsylvania       PA

3  248614      NaN       140       231       21      Kentucky       KY

4  286865      NaN       140       355       48         Texas       TX


             city                  place     type  ... female_age_mean
\
0         Detroit  Dearborn Heights City      CDP  ...        34.78682

1          Auburn            Auburn City     City  ...        44.23451

2       Pine City              Millerton  Borough  ...        41.62426

3      Monticello        Monticello City     City  ...        44.81200

4  Corpus Christi                  Edroy     Town  ...        40.66618


   female_age_median  female_age_stdev  female_age_sample_weight  \
0           33.75000          21.58531                 416.48097
1           46.66667          22.37036                 532.03505
2           44.50000          22.86213                 453.11959
3           48.00000          21.03155                 263.94320
4           42.66667          21.30900                 709.90829
```

```
   female_age_samples  pct_own  married  married_snp  separated
divorced
0              1938.0  0.70252  0.28217      0.05910    0.03813
0.14299
1              1950.0  0.85128  0.64221      0.02338    0.00000
0.13377
2              1879.0  0.81897  0.59961      0.01746    0.01358
0.10026
3              1081.0  0.84609  0.56953      0.05492    0.04694
0.12489
4              2956.0  0.79077  0.57620      0.01726    0.00588
0.16379

[5 rows x 80 columns]
```

```
data_train.shape
```

```
(27321, 80)
```

```
data_test.shape
```

```
(11709, 80)
```

```
data_train.columns
```

```
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')
```

```
data_test.columns
```

```
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')
```

```
data_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
```

```
data_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
```

```
data_train.describe()
```

```
                 UID  BLOCKID  SUMLEVEL      COUNTYID       STATEID  \
count   27321.000000      0.0   27321.0  27321.000000  27321.000000
mean   257331.996303      NaN     140.0     85.646426     28.271806
std     21343.859725      NaN       0.0     98.333097     16.392846
min    220342.000000      NaN     140.0      1.000000      1.000000
25%    238816.000000      NaN     140.0     29.000000     13.000000
50%    257220.000000      NaN     140.0     63.000000     28.000000
75%    275818.000000      NaN     140.0    109.000000     42.000000
max    294334.000000      NaN     140.0    840.000000     72.000000

           zip_code     area_code           lat           lng
ALand  \
count  27321.000000  27321.000000  27321.000000  27321.000000
2.732100e+04
mean   50081.999524    596.507668     37.508813    -91.288394
1.295106e+08
std    29558.115660    232.497482      5.588268     16.343816
1.275531e+09
min      602.000000    201.000000     17.929085   -165.453872
4.113400e+04
25%    26554.000000    405.000000     33.899064    -97.816067
1.799408e+06
50%    47715.000000    614.000000     38.755183    -86.554374
4.866940e+06
75%    77093.000000    801.000000     41.380606    -79.782503
3.359820e+07
max    99925.000000    989.000000     67.074017    -65.379332
1.039510e+11

       ...  female_age_mean  female_age_median  female_age_stdev  \
count  ...     27115.000000       27115.000000      27115.000000
mean   ...        40.319803          40.355099         22.178745
std    ...         5.886317           8.039585          2.540257
min    ...        16.008330          13.250000          0.556780
25%    ...        36.892050          34.916670         21.312135
50%    ...        40.373320          40.583330         22.514410
```

```
75%    ...      43.567120       45.416670       23.575260
max    ...      79.837390       82.250000       30.241270

       female_age_sample_weight  female_age_samples       pct_own  \
count              27115.000000        27115.000000  27053.000000
mean                 544.238432         2208.761903      0.640434
std                  283.546896         1089.316999      0.226640
min                    0.664700            2.000000      0.000000
25%                  355.995825         1471.000000      0.502780
50%                  503.643890         2066.000000      0.690840
75%                  680.275055         2772.000000      0.817460
max                 6197.995200        27250.000000      1.000000

            married   married_snp     separated      divorced
count  27130.000000  27130.000000  27130.000000  27130.000000
mean       0.508300      0.047537      0.019089      0.100248
std        0.136860      0.037640      0.020796      0.049055
min        0.000000      0.000000      0.000000      0.000000
25%        0.425102      0.020810      0.004530      0.065800
50%        0.526665      0.038840      0.013460      0.095205
75%        0.605760      0.065100      0.027488      0.129000
max        1.000000      0.714290      0.714290      1.000000

[8 rows x 74 columns]
```

```
data_test.describe()
```

```
                 UID  BLOCKID  SUMLEVEL      COUNTYID       STATEID  \
count   11709.000000      0.0   11709.0  11709.000000  11709.000000
mean   257525.004783      NaN     140.0     85.710650     28.489196
std     21466.372658      NaN       0.0     99.304334     16.607262
min    220336.000000      NaN     140.0      1.000000      1.000000
25%    238819.000000      NaN     140.0     29.000000     13.000000
50%    257651.000000      NaN     140.0     61.000000     28.000000
75%    276300.000000      NaN     140.0    109.000000     42.000000
max    294333.000000      NaN     140.0    810.000000     72.000000

           zip_code     area_code           lat           lng
ALand  \
count  11709.000000  11709.000000  11709.000000  11709.000000
1.170900e+04
mean   50123.418396    593.598514     37.405491    -91.340229
1.095500e+08
std    29775.134038    232.074263      5.625904     16.407818
7.624940e+08
min      601.000000    201.000000     17.965835   -166.770979
8.299000e+03
25%    25570.000000    404.000000     33.919813    -97.816561
1.718660e+06
50%    47362.000000    612.000000     38.618093    -86.643344
```

```
4.835000e+06
75%    77406.000000    787.000000     41.232973    -79.697311
3.204540e+07
max    99929.000000    989.000000     64.804269    -65.695344
5.520166e+10

       ...  female_age_mean  female_age_median  female_age_stdev  \
count  ...     11613.000000       11613.000000      11613.000000
mean   ...        40.111999          40.131864         22.148145
std    ...         5.851192           7.972026          2.554907
min    ...        15.360240          12.833330          0.737110
25%    ...        36.729210          34.750000         21.270920
50%    ...        40.196960          40.333330         22.472990
75%    ...        43.496490          45.333330         23.549450
max    ...        90.107940          90.166670         29.626680

       female_age_sample_weight  female_age_samples       pct_own  \
count              11613.000000        11613.000000  11587.000000
mean                 550.411243         2233.003186      0.634194
std                  280.992521         1072.017063      0.232232
min                    0.251910            3.000000      0.000000
25%                  363.225840         1499.000000      0.492500
50%                  509.103610         2099.000000      0.687640
75%                  685.883910         2800.000000      0.815235
max                 4145.557870        15466.000000      1.000000

            married   married_snp     separated      divorced
count  11625.000000  11625.000000  11625.000000  11625.000000
mean       0.505632      0.047960      0.019346      0.099191
std        0.139774      0.038693      0.021428      0.048525
min        0.000000      0.000000      0.000000      0.000000
25%        0.422020      0.020890      0.004500      0.064590
50%        0.525270      0.038680      0.013870      0.094350
75%        0.605660      0.065340      0.027910      0.128400
max        1.000000      0.714290      0.714290      0.362750

[8 rows x 74 columns]
```

```
data_train.isnull().sum()
```

```
UID                0
BLOCKID        27321
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...
pct_own          268
married          191
married_snp      191
separated        191
```

```
divorced         191
Length: 80, dtype: int64
```

```
data_train.isnull().any().value_counts()
```

```
True     59
False    21
dtype: int64
```

```
data_test.isnull().sum()
```

```
UID                0
BLOCKID        11709
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...
pct_own          122
married           84
married_snp       84
separated         84
divorced          84
Length: 80, dtype: int64
```

```
data_test.isnull().any().value_counts()
```

```
True     59
False    21
dtype: int64
```

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

```
data_train.nunique()
```

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

```
data_test.nunique()
```

```
UID            11677
BLOCKID            0
SUMLEVEL           1
```

```
COUNTYID         246
STATEID           52
               ...
pct_own        10578
married        10215
married_snp     6829
separated       4512
divorced        8273
Length: 80, dtype: int64
```

```python
###UID is unique userID value in the train and test dataset and it as
more number of unique values So an index can be created from the UID
feature

#Set the DataFrame index using existing columns.
data_train.set_index(keys=['UID'],inplace=True)
data_test.set_index(keys=['UID'],inplace=True)

data_train.head()
```

```
        BLOCKID  SUMLEVEL  COUNTYID  STATEID        state state_ab  \
UID
267822      NaN       140        53       36     New York       NY
246444      NaN       140       141       18      Indiana       IN
245683      NaN       140        63       18      Indiana       IN
279653      NaN       140       127       72  Puerto Rico       PR
247218      NaN       140       161       20       Kansas       KS

              city           place   type primary  ...
female_age_mean  \
UID                                                ...

267822    Hamilton        Hamilton   City   tract  ...
44.48629
246444  South Bend        Roseland   City   tract  ...
36.48391
245683    Danville        Danville   City   tract  ...
42.15810
279653    San Juan        Guaynabo  Urban   tract  ...
47.77526
247218   Manhattan  Manhattan City   City   tract  ...
24.17693

        female_age_median  female_age_stdev  female_age_sample_weight
\
UID

267822           45.33333          22.51276                 685.33845

246444           37.58333          23.43353                 267.23367
```

```
245683           42.83333         23.94119                 707.01963

279653           50.58333         24.32015                 362.20193

247218           21.58333         11.10484                1854.48652


        female_age_samples  pct_own  married  married_snp  separated
divorced
UID

267822              2618.0  0.79046  0.57851      0.01882    0.01240
0.08770
246444              1284.0  0.52483  0.34886      0.01426    0.01426
0.09030
245683              3238.0  0.85331  0.64745      0.02830    0.01607
0.10657
279653              1559.0  0.65037  0.47257      0.02021    0.02021
0.10106
247218              3051.0  0.13046  0.12356      0.00000    0.00000
0.03109

[5 rows x 79 columns]
```

```
data_test.head()
```

```
        BLOCKID  SUMLEVEL  COUNTYID  STATEID         state state_ab  \
UID
255504      NaN       140       163       26      Michigan       MI
252676      NaN       140         1       23         Maine       ME
276314      NaN       140        15       42  Pennsylvania       PA
248614      NaN       140       231       21      Kentucky       KY
286865      NaN       140       355       48         Texas       TX

                  city                  place     type primary  ...  \
UID                                                             ...
255504         Detroit  Dearborn Heights City      CDP   tract  ...
252676          Auburn            Auburn City     City   tract  ...
276314       Pine City              Millerton  Borough   tract  ...
248614      Monticello        Monticello City     City   tract  ...
286865  Corpus Christi                  Edroy     Town   tract  ...

        female_age_mean  female_age_median  female_age_stdev  \
UID
255504         34.78682           33.75000          21.58531
252676         44.23451           46.66667          22.37036
276314         41.62426           44.50000          22.86213
248614         44.81200           48.00000          21.03155
286865         40.66618           42.66667          21.30900
```

```
        female_age_sample_weight  female_age_samples  pct_own  married
\
UID

255504                 416.48097              1938.0  0.70252  0.28217

252676                 532.03505              1950.0  0.85128  0.64221

276314                 453.11959              1879.0  0.81897  0.59961

248614                 263.94320              1081.0  0.84609  0.56953

286865                 709.90829              2956.0  0.79077  0.57620


        married_snp  separated  divorced
UID
255504      0.05910    0.03813   0.14299
252676      0.02338    0.00000   0.13377
276314      0.01746    0.01358   0.10026
248614      0.05492    0.04694   0.12489
286865      0.01726    0.00588   0.16379

[5 rows x 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.

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

```
                           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
```

```
#BLOCKID can be dropped, since it is 43%missing values
#SUMLEVEL doest not have any predictive power and no variance
data_train.drop(columns=['BLOCKID','SUMLEVEL'],axis=1,inplace=True)
data_test .drop(columns=['BLOCKID','SUMLEVEL'],axis=1,inplace=True)
```

```
print('No.of missing value in pop in train dataset = ',
(data_train['pop']==0).sum())
print('No.of missing value in pop in test  dataset = ',
(data_test['pop']==0).sum())
```

```
No.of missing value in pop in train dataset =  182
No.of missing value in pop in test  dataset =  81
```

##There are 182 records with population as zero.So remove them

```
data_train =
data_train.drop(data_train[data_train['pop']==0].index).reset_index(dr
op=True)
data_test =
data_test.drop(data_test[data_test['pop']==0].index).reset_index(drop=
True)
```

```
print('No.of missing value in pop in train dataset = ',
(data_train['pop']==0).sum())
print('No.of missing value in pop in test  dataset = ',
(data_test['pop']==0).sum())
```

```
No.of missing value in pop in train dataset =  0
No.of missing value in pop in test  dataset =  0
```

##Finding the remaining missing values in the datasets

```
print('Remaining missing values for train dataset :')
print(data_train.isnull().any().value_counts(), '\n')
print('Remaining missing values for test dataset :')
print(data_test.isnull().any().value_counts())
```

```
Remaining missing values for train dataset :
True     58
False    19
dtype: int64

Remaining missing values for test dataset :
True     58
False    19
dtype: int64
```

So there are 58 columns in train and test datasets with missing values

Imputing missing values with median value of corresponding columns because median is independent of outliers.

```
records = len(data_train)
columns = data_train.columns
for i in columns:
    counts = data_train[i].count()
    if counts < records:
        data_train[i].fillna(data_train[i].median(), inplace=True)
```

```
records = len(data_test)
columns = data_test.columns
for i in columns:
    counts = data_test[i].count()
    if counts < records:
        data_test[i].fillna(data_test[i].median(), inplace=True)
```

```
print('Is there any more missing values for train dataset :')
print(data_train.isnull().any().value_counts(), '\n')
print('Is there any more missing values for test dataset :')
print(data_test.isnull().any().value_counts())
```

```
Is there any more missing values for train dataset :
False    77
dtype: int64

Is there any more missing values for test dataset :
False    77
dtype: int64
```

```
##Missing value treatment complete. Saving the data to csv file for
tableau dashboard
```

```
data_train.to_csv('RealEstate.csv')
```

Exploratory Data Analysis (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

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

```
data_train_location_mort_pct.head()
```

```
             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
```

```
import plotly.express as px
import plotly.graph_objects as go

subset1 = []
subset1 = data_train.loc[:, ['STATEID', 'state', 'city', 'place',
'lat', 'lng', 'home_equity_second_mortgage',
                             'second_mortgage', 'home_equity', 'debt',
'pct_own']]

subset1['bad_debt'] = subset1['second_mortgage'] +
subset1['home_equity'] - subset1['home_equity_second_mortgage']
subset1
```

```
       STATEID         state         city              place        lat
\
0           36      New York     Hamilton           Hamilton  42.840812

1           18       Indiana   South Bend           Roseland  41.701441

2           18       Indiana     Danville           Danville  39.792202

3           72   Puerto Rico     San Juan           Guaynabo  18.396103

4           20        Kansas    Manhattan     Manhattan City  39.195573

...        ...           ...          ...                ...        ...

27134       72   Puerto Rico        Coamo              Coamo  18.076060

27135       42  Pennsylvania    Blue Bell          Blue Bell  40.158138

27136        8      Colorado      Weldona      Saddle Ridge  40.410316

27137       48         Texas  Colleyville  Colleyville City  32.904866

27138       32        Nevada    Las Vegas           Paradise  36.064754


              lng  home_equity_second_mortgage  second_mortgage
home_equity  \
0      -75.501524                      0.01588          0.02077
0.08919
1      -86.266614                      0.02222          0.02222
0.04274
2      -86.515246                      0.00000          0.00000
0.09512
3      -66.104169                      0.01086          0.01086
```

```
0.01086
4      -96.569366                          0.05426          0.05426
0.05426
...           ...                              ...              ...
...
27134  -66.358379                          0.00000          0.00000
0.00000
27135  -75.307271                          0.00845          0.02112
0.19641
27136 -103.814003                          0.02024          0.02024
0.07857
27137  -97.162151                          0.05801          0.07550
0.12556
27138 -115.152237                          0.01412          0.01412
0.18362

          debt  pct_own  bad_debt
0      0.52963  0.79046   0.09408
1      0.60855  0.52483   0.04274
2      0.73484  0.85331   0.09512
3      0.52714  0.65037   0.01086
4      0.51938  0.13046   0.05426
...        ...      ...       ...
27134  0.11694  0.60422   0.00000
27135  0.65364  0.68072   0.20908
27136  0.58095  0.78508   0.07857
27137  0.65722  0.93970   0.14305
27138  0.65537  0.27912   0.18362

[27139 rows x 12 columns]
```

```
subset2 = subset1.loc[(subset1['second_mortgage'] > 0.2) &
(subset1['pct_own'] > 0.1)]
subset2
```

```
       STATEID            state            city  \
593          9      Connecticut        Hartford
1144         8         Colorado     Westminster
1690        17         Illinois         Chicago
1744        41           Oregon    Happy Valley
2062        12          Florida           Tampa
3260        51         Virginia       Farmville
4972        47        Tennessee         Memphis
5154        53       Washington          Tacoma
6431        55        Wisconsin       Milwaukee
7235        24         Maryland     Hyattsville
7667        12          Florida         Orlando
7767        24         Maryland     Glen Burnie
8036        39             Ohio      Cincinnati
8378        39             Ohio   East Cleveland
```

```
8705        17              Illinois              Chicago
8797        26              Michigan              Lansing
9998         6            California                 Napa
10249        8              Colorado            Littleton
10848        8              Colorado     Colorado Springs
11577        6            California             Etiwanda
11756       17              Illinois              Chicago
11893       25         Massachusetts            Worcester
12342        6            California            Fairfield
14391        6            California          Los Angeles
15335        6            California             Murrieta
16673       36              New York              Yonkers
17013       10              Delaware           New Castle
17359       36              New York            Watertown
18527        6            California            San Pedro
19335       12               Florida            Hollywood
21048        6            California  South San Francisco
21685        8              Colorado               Denver
22541       12               Florida              Oldsmar
23456       13               Georgia               Winder
23460       28           Mississippi              Jackson
23613       48                 Texas               Dallas
23824       26              Michigan        Highland Park
25843       36              New York               Corona
26408       11  District of Columbia           Washington
26902       28           Mississippi               Oxford

                        place        lat         lng  \
593             Hartford City  41.767728  -72.706646
1144             Shaw Heights  39.859951 -105.038811
1690              Lincolnwood  41.967289  -87.652434
1744           Milwaukie City  45.445405 -122.574608
2062          Egypt Lake-leto  28.029063  -82.495395
3260               Farmville  37.297357  -78.396452
4972             Memphis City  35.128588  -90.039448
5154              Tacoma City  47.240148 -122.437743
6431           Milwaukee City  43.067063  -87.953378
7235                  Chillum  38.971338  -76.985846
7667               Orlovista  28.533263  -81.468627
7767              Glen Burnie  39.127273  -76.635265
8036          Cincinnati City  39.121316  -84.511896
8378      East Cleveland City  41.527016  -81.572525
8705             Chicago City  41.783468  -87.593521
8797             Lansing City  42.714208  -84.519179
9998                Napa City  38.294765 -122.287773
10249                Louviers  39.509438 -105.063203
10848  Colorado Springs City  38.819171 -104.750161
11577  Rancho Cucamonga City  34.093184 -117.531484
11756            Chicago City  41.906640  -87.689580
11893          Worcester City  42.254262  -71.800347
```

```
12342         Fairfield City  38.247035 -122.044933
14391    South Pasadena City  34.103959 -118.197125
15335          Murrieta City  33.565123 -117.191894
16673           Yonkers City  40.919967  -73.898818
17013       Wilmington Manor  39.659917  -75.622893
17359         Watertown City  43.972857  -75.906025
18527     Rolling Hills City  33.724904 -118.291416
19335          Pembroke Park  26.007309  -80.180970
21048         San Bruno City  37.656229 -122.417568
21685          Aetna Estates  39.775653 -104.744413
22541           Oldsmar City  28.023056  -82.645981
23456                   Carl  33.955590  -83.780511
23460           Jackson City  32.295589  -90.170816
23613            Dallas City  32.800348  -96.773713
23824     Highland Park City  42.397127  -83.105436
25843           Harbor Hills  40.751809  -73.853582
26408        Washington City  38.849942  -77.000410
26902             University  34.363620  -89.544511

       home_equity_second_mortgage  second_mortgage  home_equity
debt  \
593                        0.22997          0.22997      0.48780
0.94774
1144                       0.17532          0.20022      0.30303
0.76732
1690                       0.28899          0.28899      0.40826
0.83945
1744                       0.22464          0.22464      0.26570
0.53140
2062                       0.28972          0.28972      0.38785
0.78972
3260                       0.00000          0.50000      0.00000
0.50000
4972                       0.21875          0.21875      0.21875
1.00000
5154                       0.21429          0.21429      0.21429
0.78571
6431                       0.21277          0.26596      0.30851
0.86170
7235                       0.08614          0.21348      0.29213
0.91760
7667                       0.04604          0.22284      0.15470
0.80479
7767                       0.27739          0.30212      0.35689
0.87633
8036                       0.16168          0.25150      0.19162
0.74850
8378                       0.23529          0.23529      0.40588
0.77059
8705                       0.18182          0.22727      0.23636
```

0.81818			
8797	0.26667	0.26667	0.26667
1.00000			
9998	0.18707	0.23810	0.28571
0.74490			
10249	0.23872	0.23872	0.33300
0.92076			
10848	0.24011	0.24011	0.38522
0.82586			
11577	0.26154	0.26154	0.40000
0.61538			
11756	0.25686	0.27431	0.25686
0.84788			
11893	0.43363	0.43363	0.43363
0.84956			
12342	0.23148	0.23148	0.23148
0.62037			
14391	0.15138	0.22477	0.15138
0.66972			
15335	0.16268	0.21222	0.29727
0.85054			
16673	0.20144	0.20144	0.25540
0.86331			
17013	0.11272	0.21176	0.21417
0.82126			
17359	0.19643	0.21429	0.25000
0.79018			
18527	0.16800	0.22000	0.31400
0.78400			
19335	0.20712	0.20712	0.23087
0.72559			
21048	0.21866	0.25948	0.27114
0.63848			
21685	0.20464	0.20464	0.23742
0.94178			
22541	0.21053	0.21053	0.21053
0.87719			
23456	0.20802	0.22363	0.23774
0.85438			
23460	0.21495	0.21495	0.21495
0.85047			
23613	0.08065	0.24731	0.13441
0.75806			
23824	0.23404	0.23404	0.23404
0.50000			
25843	0.31818	0.31818	0.40341
0.78409			
26408	0.16024	0.23124	0.16024
0.59838			
26902	0.21277	0.21277	0.21277

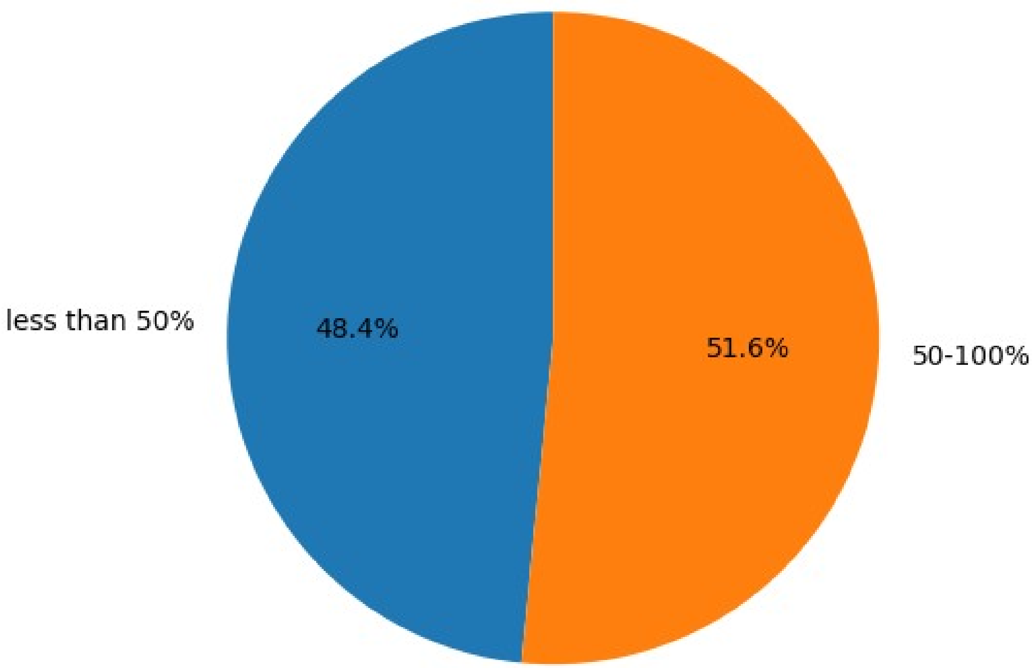
```
0.87234
```

```
       pct_own  bad_debt
593    0.14086   0.48780
1144   0.41785   0.32793
1690   0.14228   0.40826
1744   0.23231   0.26570
2062   0.11618   0.38785
3260   0.62069   0.50000
4972   0.20240   0.21875
5154   0.10670   0.21429
6431   0.26531   0.36170
7235   0.27656   0.41947
7667   0.32188   0.33150
7767   0.22380   0.38162
8036   0.29194   0.28144
8378   0.33607   0.40588
8705   0.12405   0.28181
8797   1.00000   0.26667
9998   0.20493   0.33674
10249  0.91609   0.33300
10848  0.16658   0.38522
11577  0.16188   0.40000
11756  0.29468   0.27431
11893  0.20247   0.43363
12342  0.15045   0.23148
14391  0.26450   0.22477
15335  0.50277   0.34681
16673  0.14240   0.25540
17013  0.52709   0.31321
17359  0.27999   0.26786
18527  0.41784   0.36600
19335  0.45600   0.23087
21048  0.17064   0.31196
21685  0.68714   0.23742
22541  0.32185   0.21053
23456  0.75831   0.25335
23460  0.37841   0.21495
23613  0.20517   0.30107
23824  0.27573   0.23404
25843  0.15618   0.40341
26408  0.31537   0.23124
26902  0.13256   0.21277
```

```
subset2.to_csv('Locations.csv')
```

##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

```
from statsmodels.graphics.gofplots import qqplot
import seaborn as sns

from factor_analyzer.factor_analyzer import FactorAnalyzer,
calculate_bartlett_sphericity, calculate_kmo
import matplotlib.pyplot as plt

data_train['bad_debt']=data_train['second_mortgage']
+data_train['home_equity']-data_train['home_equity_second_mortgage']

data_train['bins'] = pd.cut(data_train['bad_debt'],bins=[0,0.10,1],
labels=["less than 50%","50-100%"])
data_train.groupby(['bins']).size().plot(kind='pie',subplots=True,star
tangle=90, autopct='%1.1f%%')
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

```
cols=[]
data_train.columns

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',
```

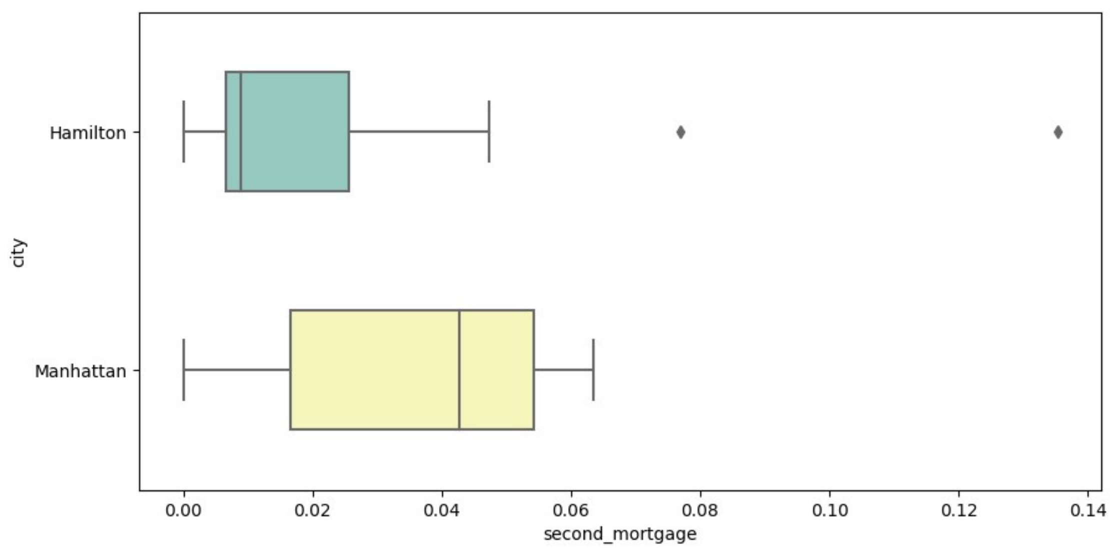
```
'rent_gt_10',
       'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30',
'rent_gt_35',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight',
'hi_samples',
       'family_mean', 'family_median', 'family_stdev',
'family_sample_weight',
       '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')
```

```
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
data_box_hamilton=data_train.loc[data_train['city'] == 'Hamilton']
data_box_manhattan=data_train.loc[data_train['city'] == 'Manhattan']
data_box_city=pd.concat([data_box_hamilton,data_box_manhattan])
data_box_city.head(4)
```

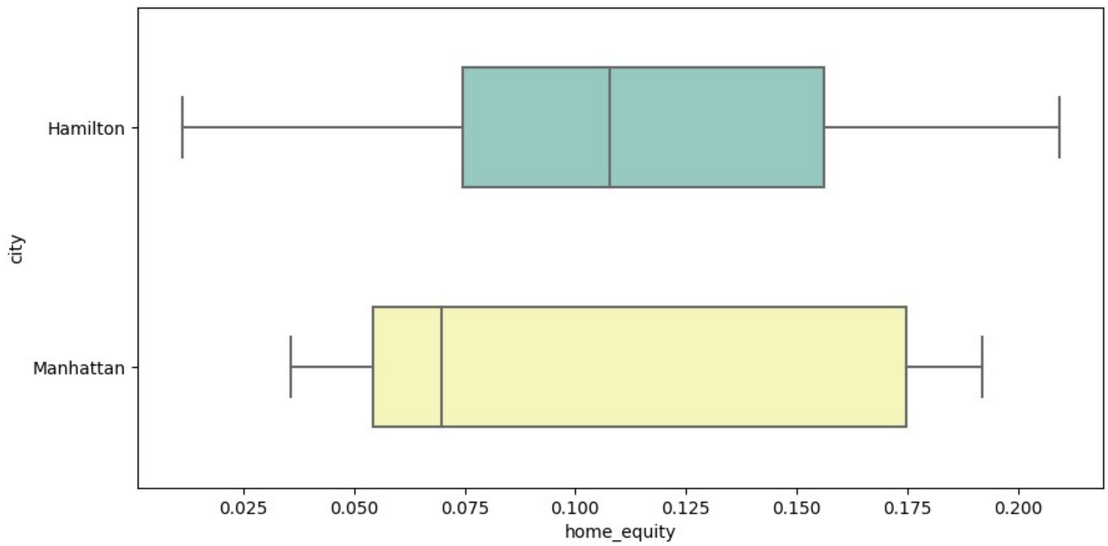
```
      COUNTYID  STATEID        state state_ab      city          place
\
0           53       36     New York       NY  Hamilton       Hamilton

390         21       34   New Jersey       NJ  Hamilton      Yardville

1368        17       39         Ohio       OH  Hamilton  Hamilton City

1396        95       28  Mississippi       MS  Hamilton       Hamilton


         type primary  zip_code  area_code  ...  female_age_stdev  \
0        City   tract     13346        315  ...          22.51276
390      City   tract      8610        609  ...          24.05831
1368  Village   tract     45015        513  ...          22.66500
1396      CDP   tract     39746        662  ...          22.79602
```

```
      female_age_sample_weight  female_age_samples  pct_own
married  \
0                    685.33845              2618.0  0.79046  0.57851

390                  732.58443              3124.0  0.64400  0.56377

1368                 565.32725              2528.0  0.61278  0.47397

1396                 483.01311              1954.0  0.83241  0.58678


      married_snp  separated  divorced  bad_debt           bins
0         0.01882    0.01240   0.08770   0.09408  less than 50%
390       0.01980    0.00990   0.04892   0.18071        50-100%
1368      0.04419    0.02663   0.13741   0.15005        50-100%
1396      0.01052    0.00000   0.11721   0.02130  less than 50%

[4 rows x 79 columns]
```

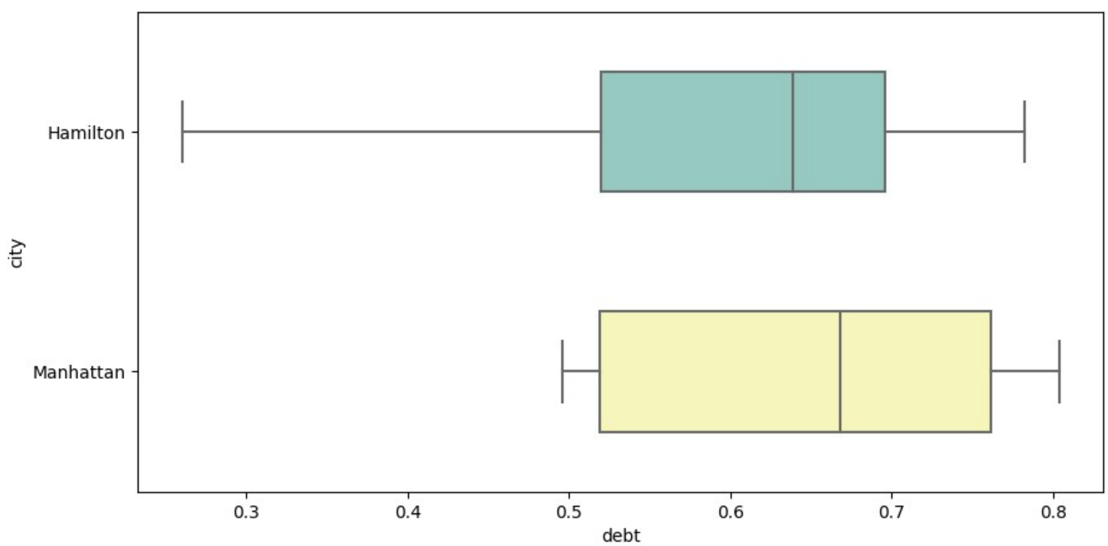
```
plt.figure(figsize=(10,5))
sns.boxplot(data=data_box_city,x='second_mortgage',
y='city',width=0.5,palette="Set3")
plt.show()
```



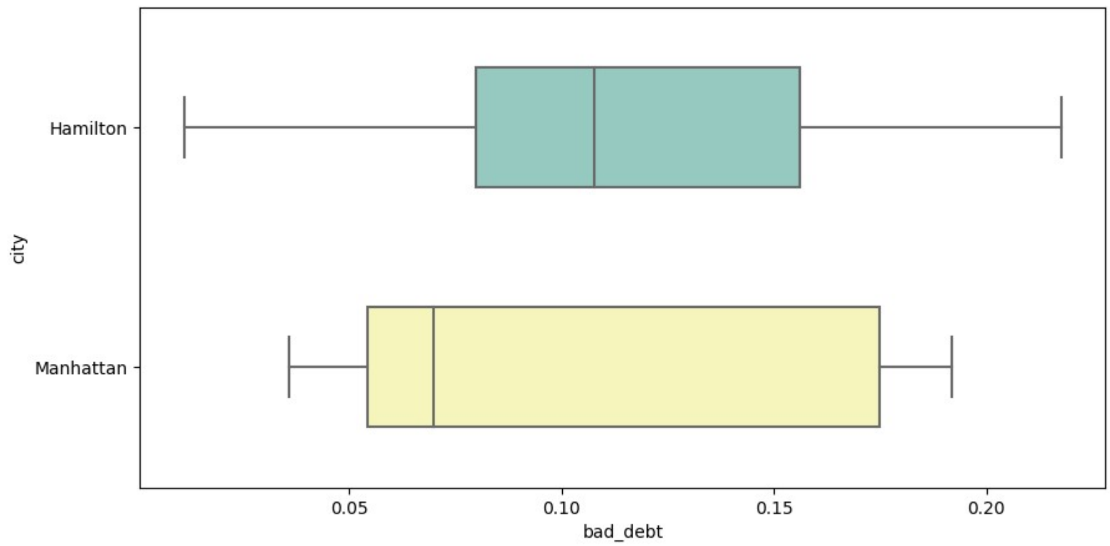

```
plt.figure(figsize=(10,5))
sns.boxplot(data=data_box_city,x='home_equity',
y='city',width=0.5,palette="Set3")
plt.show()
```




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




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

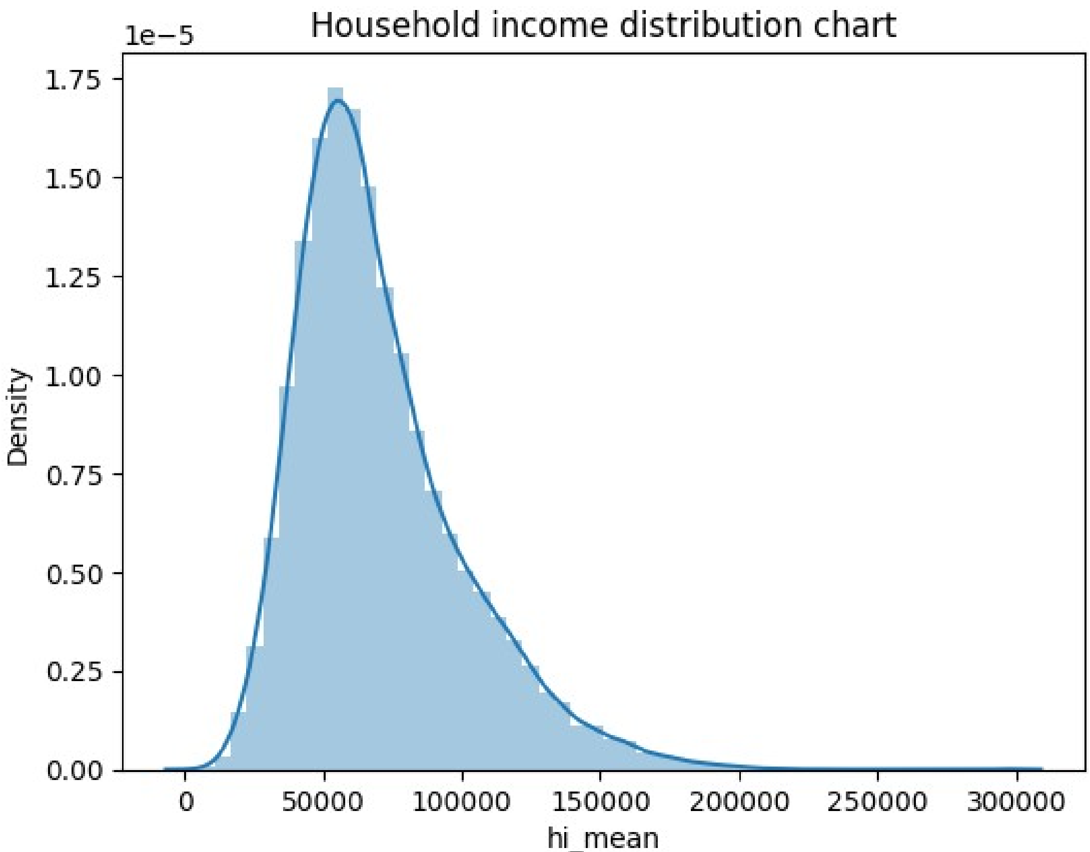


##Manhattan has higher metrics compared to Hamilton

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

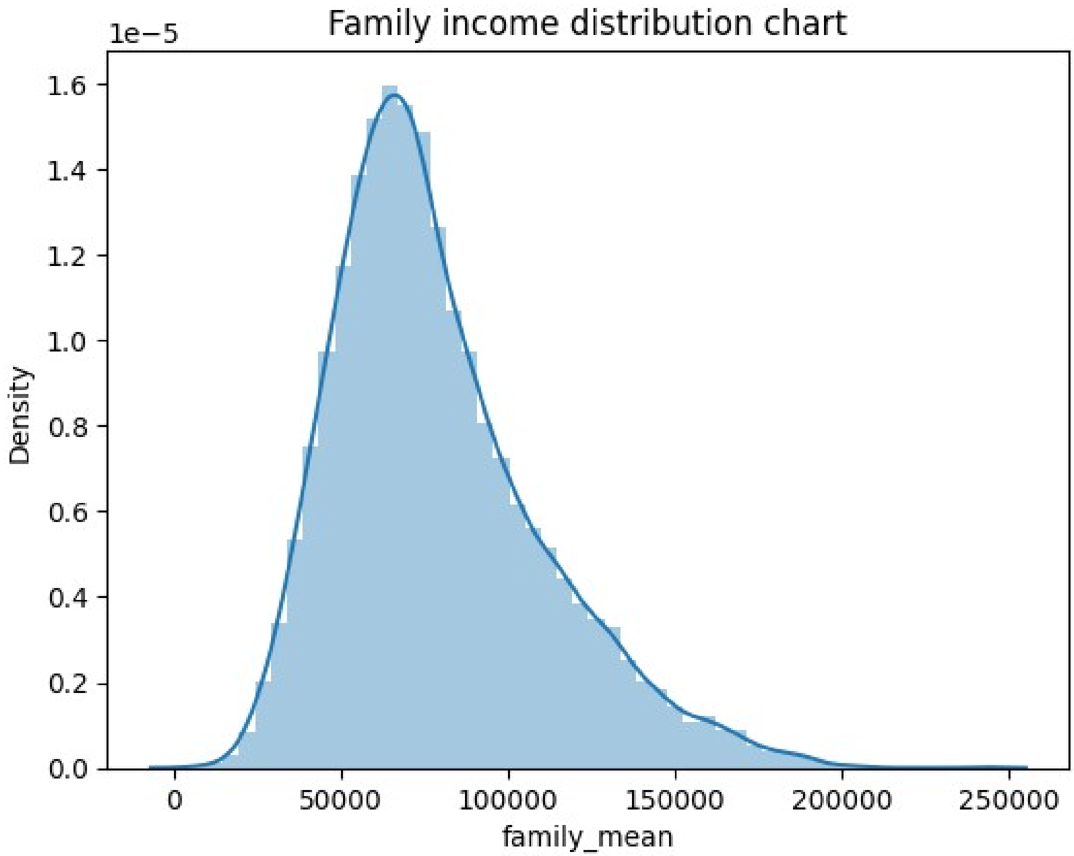
```
sns.distplot(data_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\4142743480.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['hi_mean'])
```



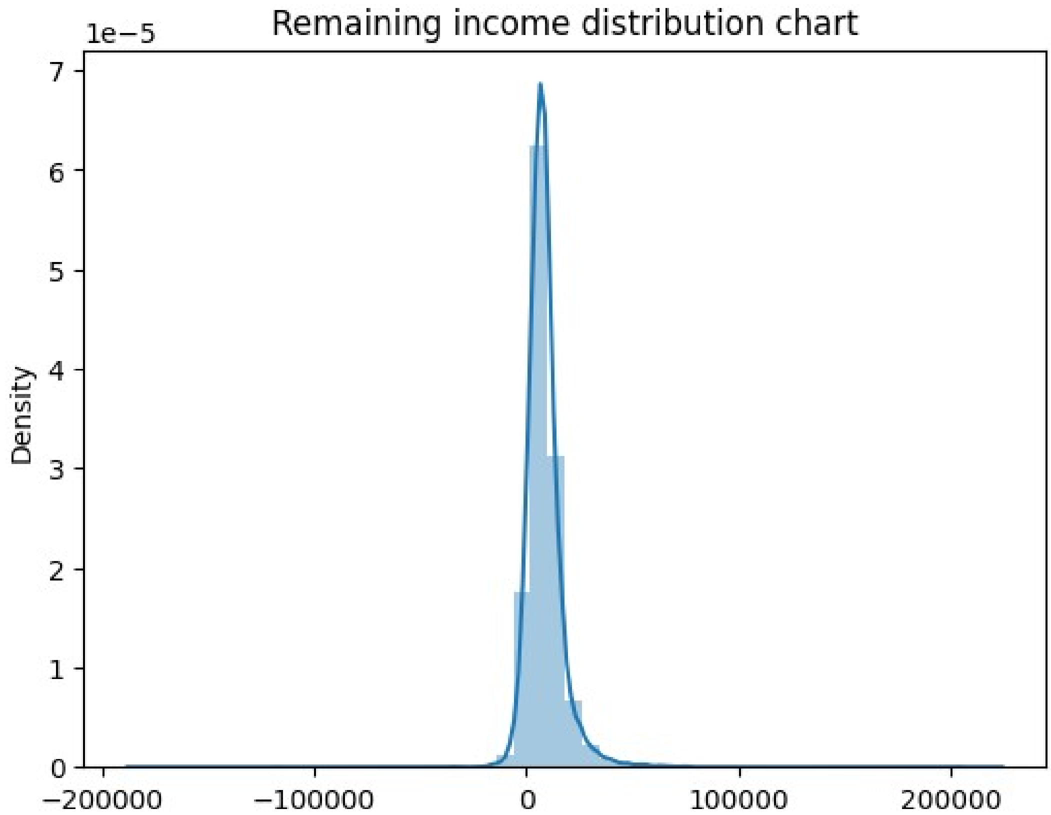

```
sns.distplot(data_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\411751307.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['family_mean'])
```




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

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\413459647.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['family_mean']-data_train['hi_mean'])
```



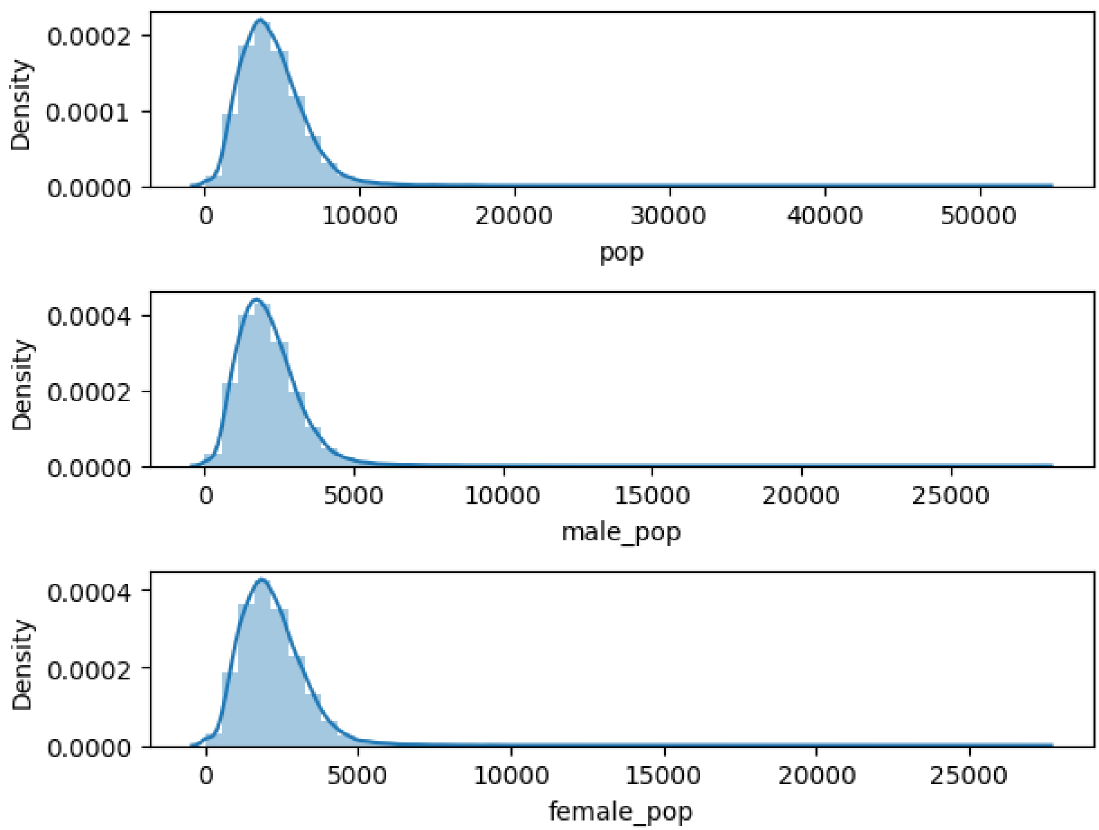
##Exploratory Data Analysis (EDA):

##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):

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

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\4212259064.py:3:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).
```

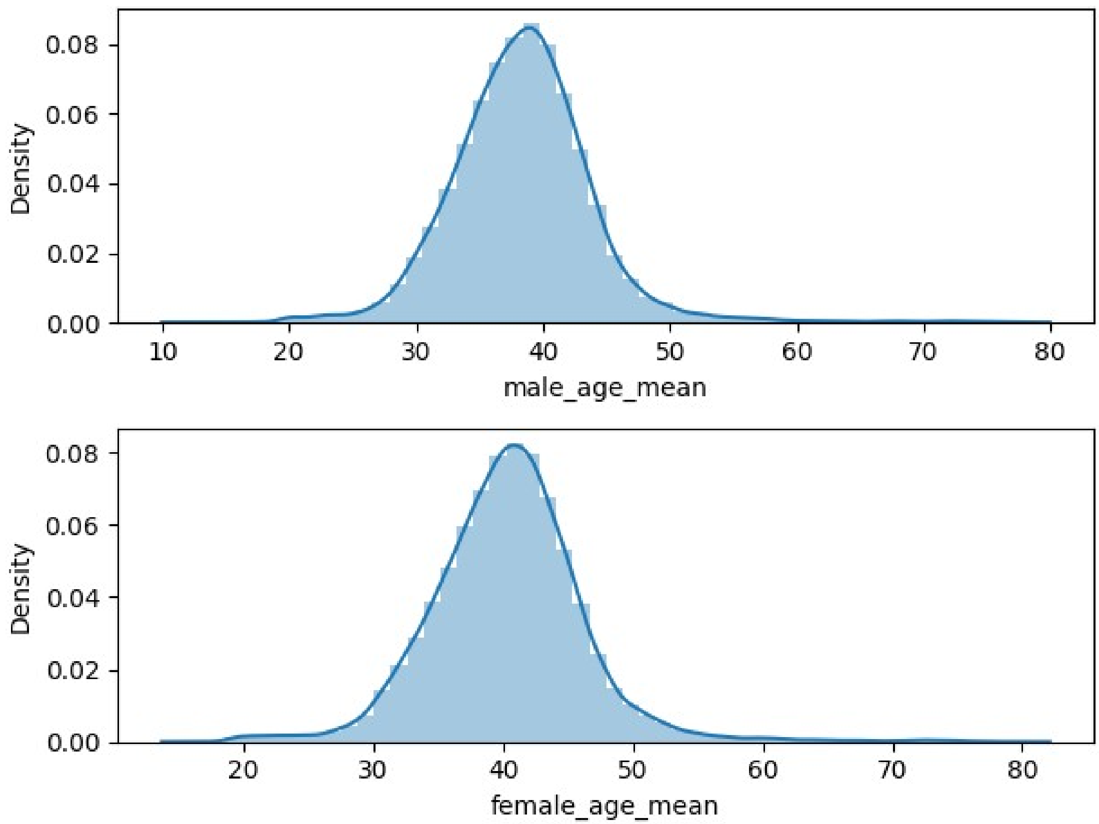
```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['pop'],ax=ax1)
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\4212259064.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['male_pop'],ax=ax2)
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\4212259064.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['female_pop'],ax=ax3)
```




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

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\345606694.py:3:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['male_age_mean'],ax=ax1)
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\345606694.py:4:
```

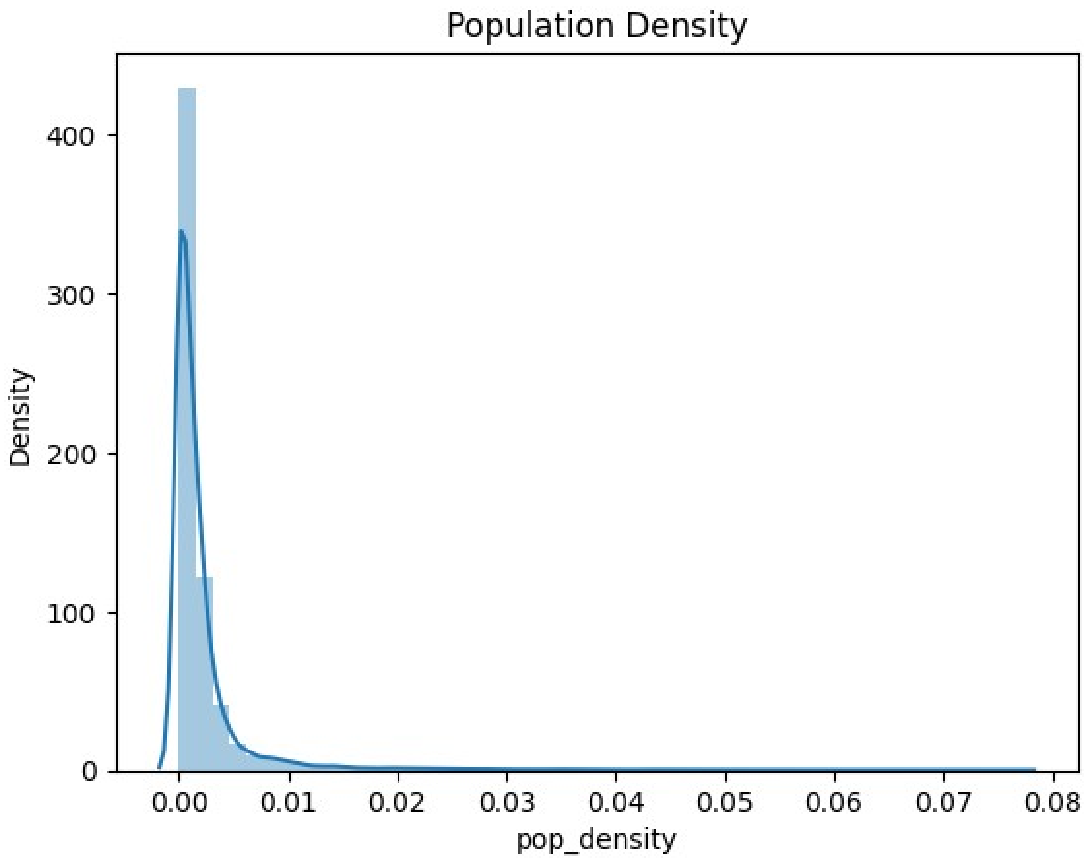
```
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['female_age_mean'],ax=ax2)
```



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

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

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

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

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\2516561165.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['pop_density'])
```



##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

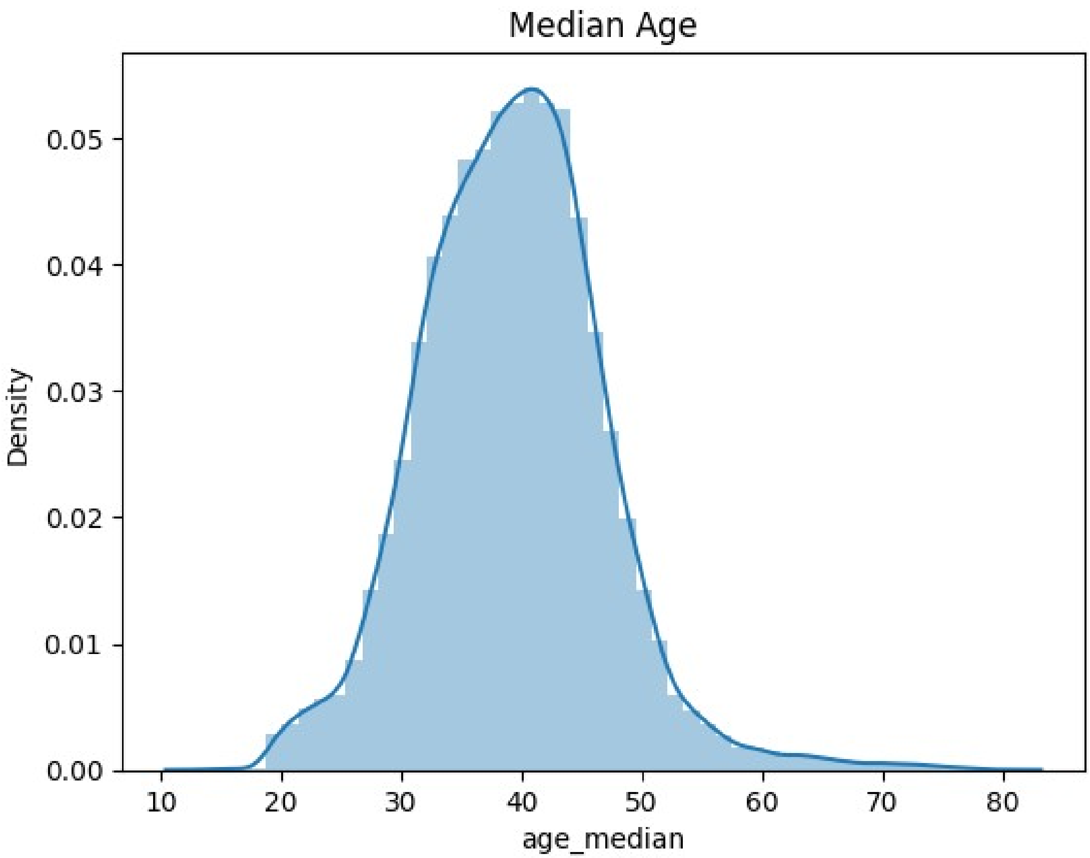
```
data_train['age_median']=(data_train['male_age_median']
+data_train['female_age_median'])/2
data_test['age_median']=(data_test['male_age_median']
+data_test['female_age_median'])/2
```

```
data_train[['male_age_median','female_age_median','male_pop','female_p
op','age_median']].head()
```

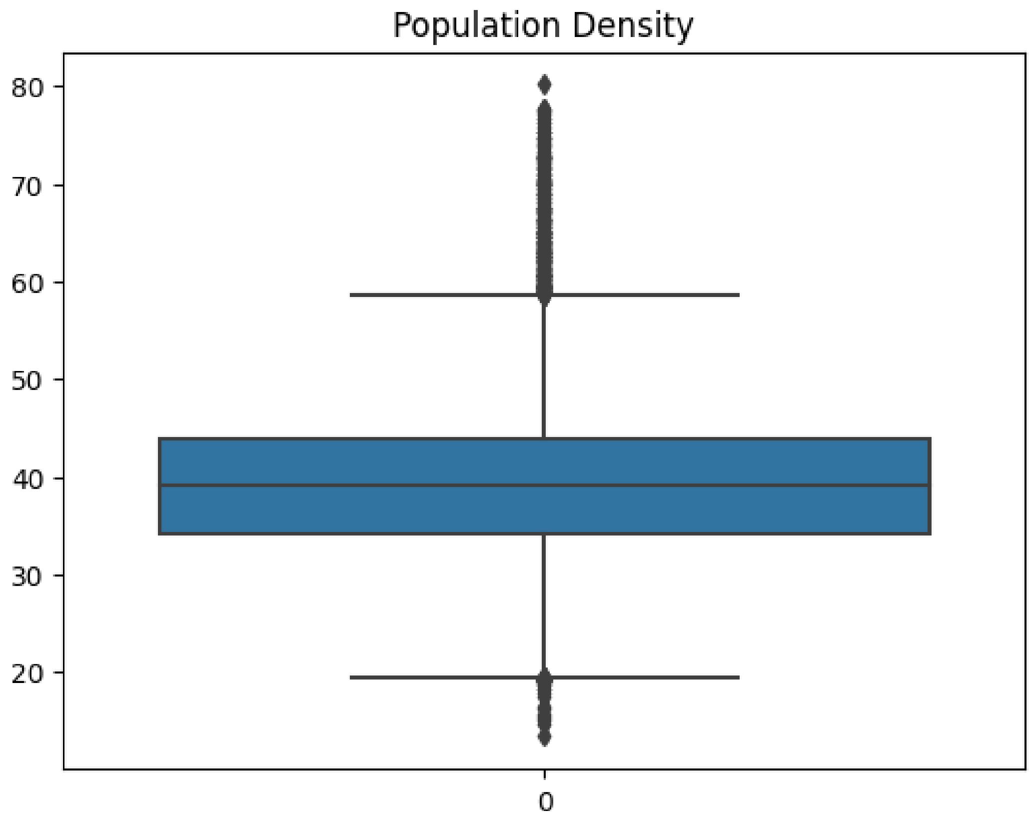
```
   male_age_median  female_age_median  male_pop  female_pop
age_median
0         44.00000           45.33333      2612        2618
44.666665
1         32.00000           37.58333      1349        1284
34.791665
2         40.83333           42.83333      3643        3238
41.833330
3         48.91667           50.58333      1141        1559
49.750000
4         22.41667           21.58333      2586        3051
22.000000
```

```
sns.distplot(data_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
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\2979765218.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_train['age_median'])
```




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



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.

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

count    27139.000000
mean      4344.976934
std       2147.401455
min          3.000000
25%       2910.500000
50%       4058.000000
75%       5441.000000
max      53812.000000
Name: pop, dtype: float64

data_train['pop_bins']=pd.cut(data_train['pop'],bins=5,labels=['very low','low','medium','high','very high'])

data_train[['pop','pop_bins']]

         pop  pop_bins
0       5230  very low
1       2633  very low
2       6881  very low
3       2700  very low
4       5637  very low
```

```
...      ...       ...
27134   1847  very low
27135   4155  very low
27136   2829  very low
27137  11542       low
27138   3726  very low

[27139 rows x 2 columns]
```

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

```
very low     26877
low            245
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

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

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

```
           married  separated  divorced
pop_bins                               
very low     26877      26877     26877
low            245        245       245
medium           9          9         9
high             7          7         7
very high        1          1         1
```

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

```
            married           separated           divorced         
               mean   median       mean   median      mean   median
pop_bins                                                           
very low   0.507554  0.52579   0.019125  0.01349  0.100502  0.09552
low        0.584680  0.59275   0.015831  0.01108  0.075402  0.07055
medium     0.655737  0.61871   0.005003  0.00412  0.065927  0.06489
high       0.503359  0.33566   0.008141  0.00250  0.039030  0.01032
very high  0.734740  0.73474   0.004050  0.00405  0.030360  0.03036
```

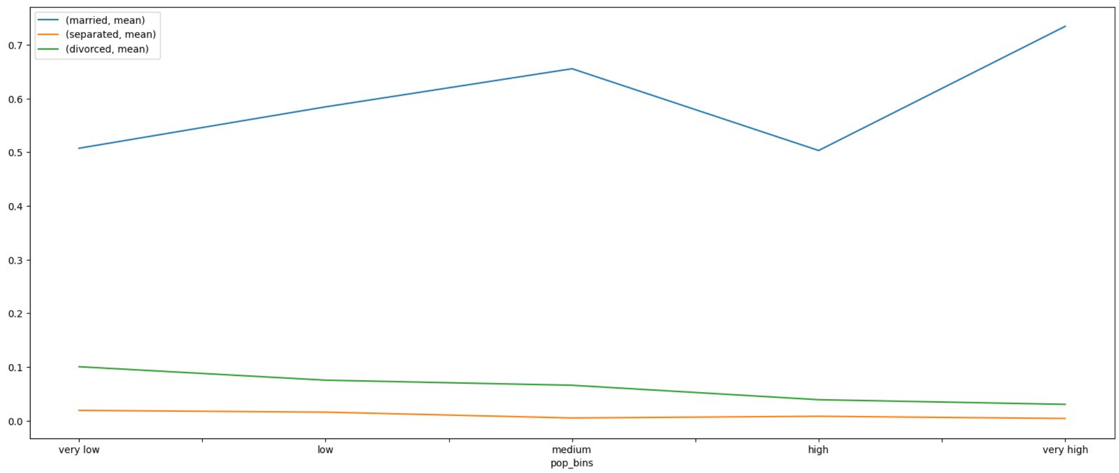
1.Very high population group has more married people and less percantage of separated and divorced couples

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

##b) Visualize using appropriate chart type

```
plt.figure(figsize=(10,5))
pop_bin_married=data_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>
```



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

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

```
                   mean
state
Alabama      772.384968
Alaska      1185.763570
Arizona     1096.933599
Arkansas     719.785963
California  1472.752217
```

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

```
                    mean
state
Alabama     66956.829961
Alaska      92136.545109
Arizona     73240.726576
Arkansas    64765.377850
California  87670.745181
```

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

```
state
Alabama                 0.011536
Alaska                  0.012870
Arizona                 0.014977
Arkansas                0.011114
California              0.016799
Colorado                0.013524
Connecticut             0.012636
Delaware                0.012929
District of Columbia    0.013192
Florida                 0.015798
Name: mean, dtype: float64
```

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

```
0.013356310653870832
```

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

```
data_train.columns
```

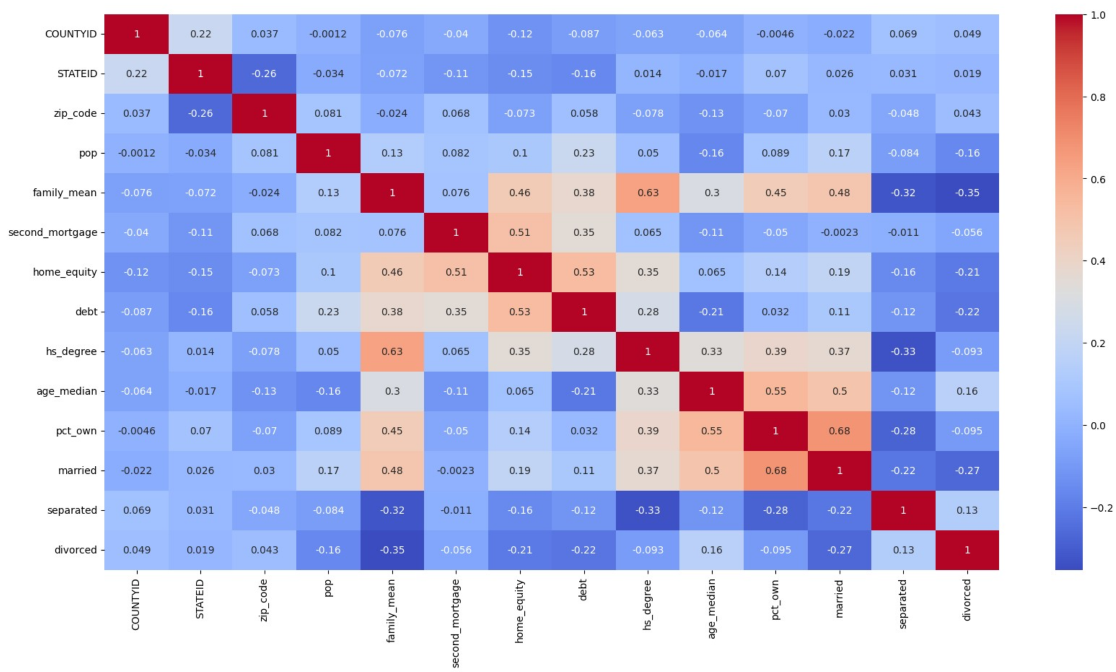
```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place',
'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand',
'AWater',
       'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'rent_stdev', 'rent_sample_weight', 'rent_samples',
'rent_gt_10',
       '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')
```

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

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\2033563093.py:3:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  'age_median','pct_own', 'married','separated', 'divorced']].corr()
```

```
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

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

```
array([[-1.13616024e-01,  2.02504992e-02, -2.36315849e-02,
        -6.28784671e-02,  4.22746298e-02],
       [-1.10874031e-01,  1.42490587e-02,  2.84831890e-02,
        -1.50689611e-01,  1.11194725e-01],
       [-8.33621628e-02,  5.14025370e-02, -1.37573750e-01,
        -4.99701837e-02, -1.04723735e-01],
       [ 1.79991203e-02,  2.01088361e-02,  5.84313104e-03,
         2.67940029e-02, -6.63178228e-03],
       [ 9.22853555e-02, -9.92154593e-02, -6.72568359e-02,
        -1.34390774e-01, -1.47354820e-01],
       [-1.07622187e-02, -4.12044377e-02,  1.47338914e-01,
         8.87958561e-03,  1.08702086e-01],
       [-4.34278445e-02, -2.12207008e-02,  3.71082296e-02,
        -9.45019155e-02,  5.83214612e-02],
       [-2.69234785e-03, -1.52718099e-02, -2.58118971e-03,
        -4.52313239e-02,  2.34474559e-02],
       [ 7.75998265e-02,  9.65109279e-01, -8.41032222e-02,
        -6.84127782e-03, -4.47639251e-02],
       [ 7.11071495e-02,  9.24512302e-01, -1.06099590e-01,
        -2.83508855e-02, -4.49944986e-02],
       [ 8.03753713e-02,  9.55681645e-01, -5.75144158e-02,
         1.53421552e-02, -4.26911847e-02],
       [ 7.68105176e-01,  1.00187525e-02, -3.77164045e-02,
         1.14844262e-01, -1.26385207e-01],
       [ 7.16660396e-01,  6.57937451e-03, -4.64890503e-02,
```

```
  1.09138460e-01, -1.37867232e-01],
[ 7.07482933e-01,  2.51688366e-02, -9.12817360e-03,
  1.04053895e-01,  7.70855484e-02],
[-1.29878836e-01,  3.39496357e-01, -4.85394519e-01,
 -4.36195869e-02,  3.27526152e-01],
[ 2.36016256e-01,  4.40357434e-01, -6.38437531e-01,
 -2.81920203e-02,  3.58107292e-01],
[-4.41975545e-02,  3.41195545e-02,  2.91439299e-02,
  4.44562984e-01, -1.63286499e-01],
[-2.42774277e-02,  1.65081016e-02,  4.50134340e-02,
  6.76178786e-01, -1.55445169e-01],
[-3.85765490e-02, -1.69888858e-02,  8.18832307e-02,
  8.36850950e-01, -9.23101412e-02],
[-5.13714372e-02, -3.57436480e-02,  1.12222196e-01,
  9.25948933e-01, -4.54719445e-02],
[-6.10962207e-02, -4.39648473e-02,  1.37889880e-01,
  9.54226989e-01, -2.36644341e-02],
[-4.62099143e-02, -5.21758017e-02,  1.43490512e-01,
  9.34077717e-01, -1.79047980e-03],
[-4.24455760e-02, -5.85402343e-02,  1.31444705e-01,
  8.88701389e-01,  8.86159204e-03],
[-2.52336512e-02, -7.23141245e-02,  9.67650749e-02,
  7.80281341e-01,  2.81970964e-02],
[ 2.17520607e-01,  4.65718558e-01, -6.14995593e-01,
 -2.79288982e-02,  3.77688953e-01],
[ 2.38446446e-01,  4.46646274e-01, -6.28879872e-01,
 -3.03502289e-02,  3.54342496e-01],
[ 7.83528761e-01,  4.83622857e-02,  1.43109374e-01,
 -2.05364286e-01, -1.58168153e-01],
[ 7.08353623e-01,  4.92967466e-02,  1.30571547e-01,
 -2.19092655e-01, -2.14567252e-01],
[ 8.62142716e-01,  4.27330846e-02,  1.65279979e-01,
 -1.20649577e-01,  3.02280389e-02],
[-2.19672087e-01,  8.47909614e-01, -4.34210614e-02,
  6.70081191e-02,  2.32806756e-01],
[ 1.46498203e-01,  9.54259061e-01,  2.40464768e-02,
 -4.79013450e-02,  1.03272487e-01],
[ 8.29251782e-01,  3.34770564e-02,  1.60008592e-01,
 -2.04981560e-01, -7.76321535e-02],
[ 7.93148245e-01,  2.77348521e-02,  1.50207698e-01,
 -2.07905181e-01, -9.42886210e-02],
[ 8.12438294e-01,  4.24541829e-02,  1.43014397e-01,
 -1.08795277e-01,  5.75482353e-02],
[-3.36704161e-01,  8.66691033e-01,  3.87752967e-02,
  9.05620178e-02,  4.50579246e-02],
[ 5.03910120e-02,  9.36542258e-01,  1.52607160e-01,
 -2.56616942e-02, -9.56073572e-02],
[ 9.76907339e-01, -3.20697540e-02, -9.77183052e-02,
  4.70483431e-02,  7.15365887e-02],
[ 9.57707833e-01, -3.78300992e-02, -1.12943717e-01,
```

```
  4.73778232e-02,  6.42169233e-02],
[ 8.14070143e-01,  2.21221367e-03,  8.06507655e-02,
  2.07353275e-02,  1.26693822e-01],
[-4.16282568e-01,  7.19886159e-01,  3.43576632e-01,
 -7.05087214e-02, -2.82613987e-01],
[ 7.49820795e-02,  7.26095184e-01,  2.76800765e-01,
 -4.72481218e-02, -3.58625987e-01],
[ 9.09978828e-01, -5.12232557e-02, -3.78040035e-02,
  1.15405733e-04,  1.65536712e-01],
[ 8.72608288e-01, -5.05484704e-02, -5.00135497e-02,
 -6.80611571e-04,  1.54115498e-01],
[ 7.55092216e-01, -1.90669904e-03,  6.17625362e-02,
  4.69959146e-03,  2.59107316e-01],
[-1.26115413e-01,  6.08598592e-01,  6.41437465e-01,
 -2.07202474e-02,  2.43226379e-01],
[-3.45172027e-01,  5.61579306e-01,  5.97889561e-01,
 -2.38456495e-02,  2.14475655e-01],
[-1.54752864e-01, -1.17108297e-02, -1.57730634e-01,
  1.10558287e-01, -6.53537606e-01],
[-1.31156114e-01, -1.81961013e-02, -1.59360462e-01,
  1.26404044e-01, -6.63482406e-01],
[ 2.49696931e-01, -2.39680187e-02, -3.08166394e-02,
  9.52012565e-02, -6.37270385e-01],
[ 2.05952484e-01,  7.84381656e-02, -3.09538572e-01,
  2.18950691e-02, -6.27901056e-01],
[ 1.03736524e-01, -6.36718323e-02, -2.55938347e-02,
 -9.39465671e-02,  6.76184195e-01],
[-2.68822610e-01, -6.92582090e-03, -2.91775141e-02,
 -9.27363765e-02,  6.42693466e-01],
[-2.17375248e-01, -7.42541120e-02,  3.56286606e-01,
 -1.91582567e-02,  6.35328771e-01],
[ 3.95835768e-01,  5.97448187e-02,  2.50772999e-01,
 -2.22532905e-01, -1.84187009e-01],
[ 4.09255136e-01,  6.14126079e-02,  2.19455853e-01,
 -2.12231090e-01, -1.71890450e-01],
[ 3.54148661e-01,  5.20948415e-02,  2.64916893e-01,
 -2.19014935e-01, -1.80545614e-01],
[ 2.35915510e-01, -5.06673518e-02,  8.15372166e-01,
  9.25526029e-02,  3.26502171e-01],
[ 2.41710266e-01, -3.54564965e-02,  8.31759880e-01,
  7.42434162e-02,  2.44307955e-01],
[-5.76796966e-02,  6.78860803e-02,  5.85906589e-01,
  8.65506611e-02,  9.56837563e-02],
[ 5.35983461e-02,  8.18456517e-01, -1.77592587e-01,
 -1.64039311e-02, -3.59193199e-02],
[ 7.08165224e-02,  9.24343156e-01, -1.06356948e-01,
 -2.88272820e-02, -4.52533770e-02],
[ 1.95677257e-01, -4.88705499e-02,  8.04227167e-01,
  1.42336208e-01,  3.34261663e-01],
[ 1.89554212e-01, -3.45664717e-02,  8.58413815e-01,
```

```
         1.30304393e-01,  2.54231824e-01],
       [-9.22402119e-02,  6.26483425e-02,  4.72782204e-01,
         7.28205375e-02,  1.17939199e-01],
       [ 6.14879138e-02,  8.79225944e-01, -1.49583032e-01,
         2.10415053e-02, -4.24408637e-02],
       [ 7.97987129e-02,  9.56023283e-01, -5.84306404e-02,
         1.52210508e-02, -4.31904892e-02],
       [-3.34346137e-02,  1.09859776e-01,  7.82459522e-01,
        -4.22860227e-02, -2.88853750e-01],
       [ 1.78709150e-01,  1.90385860e-01,  5.59586048e-01,
        -1.21043100e-01, -1.35604880e-01],
       [-6.61672453e-02, -7.03683880e-02, -2.66077409e-01,
         1.29450389e-01,  1.89027600e-01],
       [-1.55689692e-01, -7.01716329e-02, -1.43941171e-01,
         1.24770271e-01,  1.48287738e-01],
       [-3.53463766e-01, -5.25112938e-02,  1.47986271e-01,
         2.82037030e-02,  1.16839292e-01],
       [ 2.46911343e-01, -2.70624903e-02, -3.67801560e-02,
         1.04937229e-01, -6.49965661e-01],
       [ 3.49133883e-01, -1.23293081e-02, -3.92754232e-01,
         5.86445364e-02,  2.93855416e-01],
       [ 2.27422309e-01, -3.59713627e-02,  8.93211133e-01,
         1.11330915e-01,  2.65251828e-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.

```
data_train.columns
```

```
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')
```

```
data_train['type'].unique()
type_dict={'type':{'City':1,
                   'Urban':2,
                   'Town':3,
                   'CDP':4,
                   'Village':5,
                   'Borough':6}
          }
data_train.replace(type_dict,inplace=True)
```

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

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

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

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

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

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

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

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

```
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
```

```
x_train.head()
```

```
   COUNTYID  STATEID  zip_code  type   pop  family_mean
second_mortgage  \
0        53       36     13346     1  5230  67994.14790
0.02077
1       141       18     46616     1  2633  50670.10337
0.02222
2        63       18     46122     1  6881  95262.51431
0.00000
3       127       72       927     2  2700  56401.68133
0.01086
4       161       20     66502     1  5637  54053.42396
0.05426

   home_equity     debt  hs_degree  age_median  pct_own  married
separated  \
0      0.08919  0.52963    0.89288   44.666665  0.79046  0.57851
0.01240
1      0.04274  0.60855    0.90487   34.791665  0.52483  0.34886
0.01426
2      0.09512  0.73484    0.94288   41.833330  0.85331  0.64745
0.01607
3      0.01086  0.52714    0.91500   49.750000  0.65037  0.47257
0.02021
4      0.05426  0.51938    1.00000   22.000000  0.13046  0.12356
0.00000

   divorced
0   0.08770
1   0.09030
2   0.10657
3   0.10106
4   0.03109
```

```
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.
```

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

```
LinearRegression()
```

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

```
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 0.7339302514983688
Overall RMSE of linear regression model 324.96235293727136
```

##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.

```
state=data_train['STATEID'].unique()
state[0:5]
```

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

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

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

    x_test_nation=data_test[data_test['COUNTYID']==i][feature_cols]
    y_test_nation=data_test[data_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 :-
0.6095965086676078
Overall RMSE of linear regression model for state, 20 :-
325.91249059701744



State ID- 1
Overall R2 score of linear regression model for state, 1 :-
0.8106360175689272
Overall RMSE of linear regression model for state, 1 :-
308.6915410379323
```

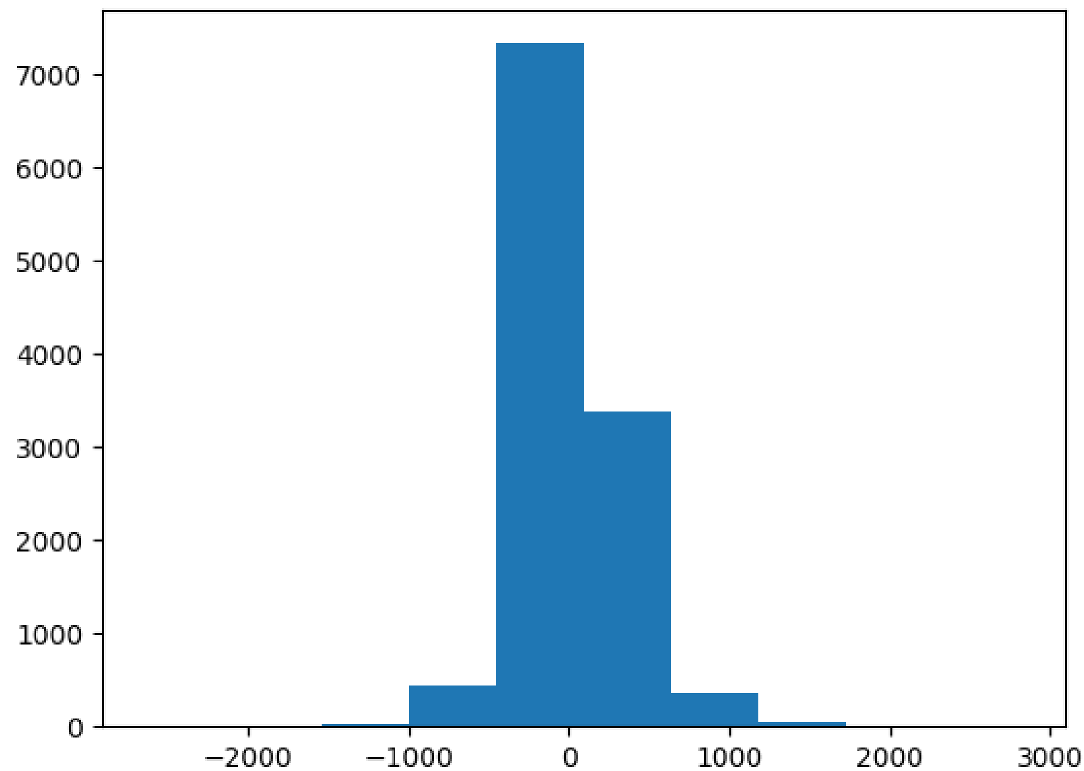
```
State ID- 45
Overall R2 score of linear regression model for state, 45 :-
0.7881168171591661
Overall RMSE of linear regression model for state, 45 :-
226.03128012397102
```

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

```
0        279.618413
1        -73.281616
2        193.217902
3       -162.514964
4        -11.732014
            ...
11623    -70.549478
11624    -35.388334
11625   -130.726446
11626   -332.316613
11627    222.599269
Name: hc_mortgage_mean, Length: 11628, dtype: float64
```

```
plt.hist(residuals)
```

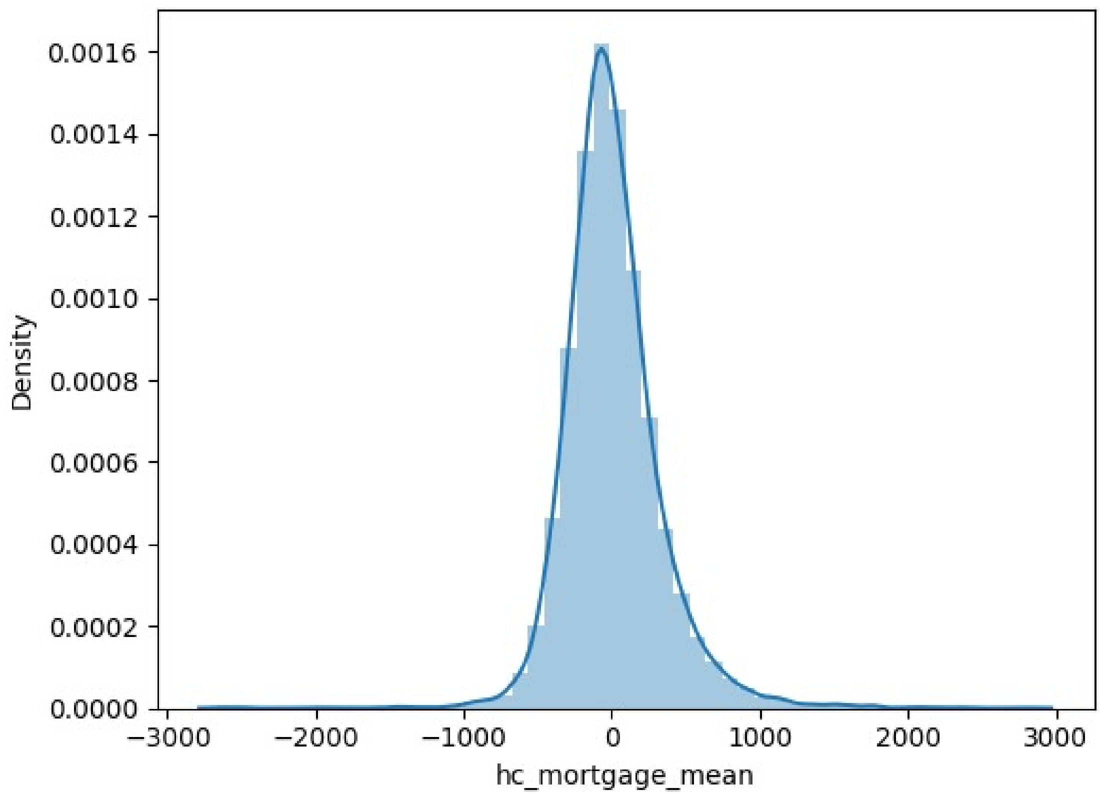
```
(array([4.000e+00, 4.000e+00, 1.700e+01, 4.530e+02, 7.336e+03,
3.380e+03,
        3.620e+02, 5.500e+01, 1.400e+01, 3.000e+00]),
 array([-2635.86777826, -2090.42179388, -1544.97580951,  -
999.52982514,
         -454.08384076,    91.36214361,   636.80812799,
1182.25411236,
         1727.70009674,  2273.14608111,  2818.59206549]),
 <BarContainer object of 10 artists>)
```




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

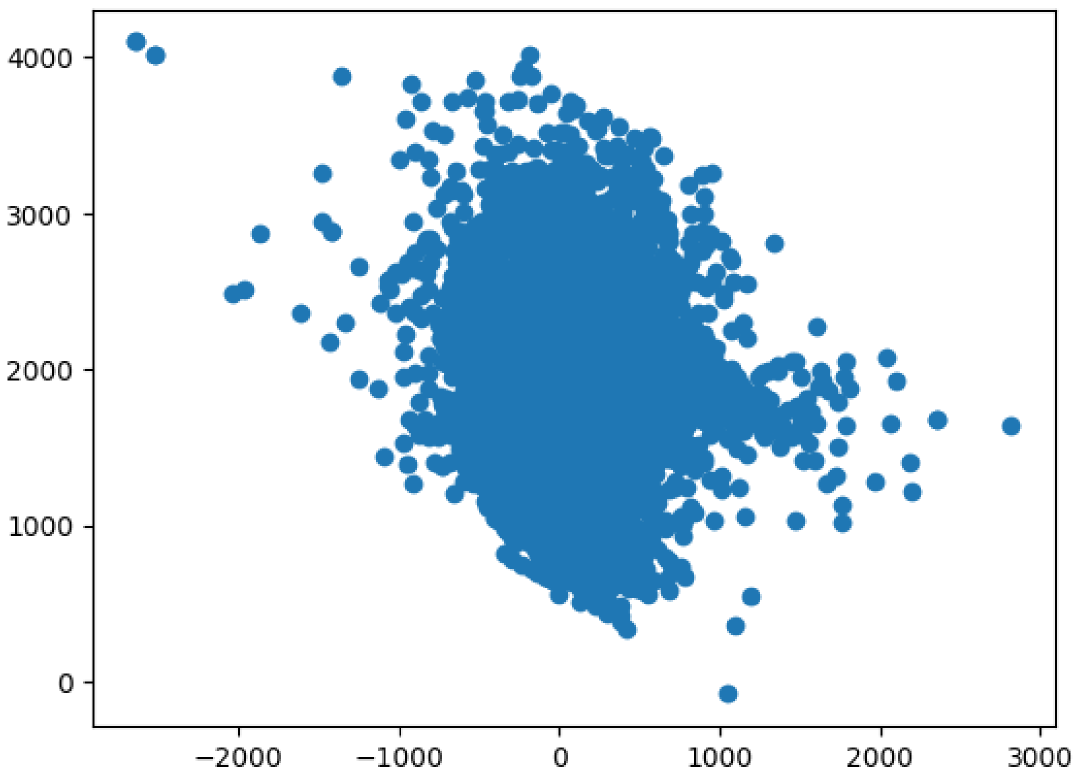
```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_1092\2665350104.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(residuals)
```

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




```
plt.scatter(residuals,y_pred)
```

```
<matplotlib.collections.PathCollection at 0x1a8b9536590>
```


4000
3000
2000
1000
0
−2000
−1000
0
1000
2000
3000