CAPSTONE PROJECT (Real Estate Case Study)

PROBLEM STATEMENT

A banking institution requires actionable insights into mortgage-backed securities, geographic business investment, and real estate analysis. The mortgage bank would like to identify potential monthly mortgage expenses for each region based on monthly family income and rental of the real estate. A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental income, monthly mortgage and owner's cost, family income vs mortgage cost comparison across different regions. The metrics described here do not limit the dashboard to these few.

1.Import data

In [1]:

```
import os

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
os.getcwd()
```

Out[2]:

```
'C:\\Users\\User\\Downloads\\sda\\STEP 6\\Capstone Project REALESTATE-master'
```

In [3]:

```
csv_s = []

for file in os.listdir():
    if file.endswith('.csv'):
```

```
        print(file)
        csv_s.append(file)

print(csv_s)
```

```
test.csv
train.csv
['test.csv', 'train.csv']
```

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from itertools import cycle

pd.set_option('max_columns', 90)
pd.set_option('max_rows', 90)
plt.style.use('bmh')
color_pal = plt.rcParams['axes.prop_cycle'].by_key()['color']
color_cycle = cycle(plt.rcParams['axes.prop_cycle'].by_key()['color'])
```

In [5]:

```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

In [6]:

```
train_df.head()
```

Out[6]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_co
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	

In [7]:

```
test_df.head()
```

Out[7]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	48239
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	4210
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract	14871

3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	tract	42633
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract	78410

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

UID is the primary Key

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 [8]:

```
train_df.columns
```

Out[8]:

```
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 [9]:

```
train_df.dtypes
```

Out[9]:

```
UID                                int64
BLOCKID                          float64
SUMLEVEL                           int64
COUNTYID                           int64
STATEID                            int64
state                             object
state_ab                          object
city                              object
```

```
place                            object
type                             object
primary                          object
zip_code                          int64
area_code                         int64
lat                             float64
lng                             float64
ALand                           float64
AWater                            int64
pop                               int64
male_pop                          int64
female_pop                        int64
rent_mean                       float64
rent_median                     float64
rent_stdev                      float64
rent_sample_weight              float64
rent_samples                    float64
rent_gt_10                      float64
rent_gt_15                      float64
rent_gt_20                      float64
rent_gt_25                      float64
rent_gt_30                      float64
rent_gt_35                      float64
rent_gt_40                      float64
rent_gt_50                      float64
universe_samples                  int64
used_samples                      int64
hi_mean                         float64
hi_median                       float64
hi_stdev                        float64
hi_sample_weight                float64
hi_samples                      float64
family_mean                     float64
family_median                   float64
family_stdev                    float64
family_sample_weight            float64
family_samples                  float64
hc_mortgage_mean                float64
hc_mortgage_median              float64
hc_mortgage_stdev               float64
hc_mortgage_sample_weight       float64
hc_mortgage_samples             float64
hc_mean                         float64
hc_median                       float64
hc_stdev                        float64
hc_samples                      float64
hc_sample_weight                float64
home_equity_second_mortgage     float64
second_mortgage                 float64
home_equity                     float64
debt                            float64
second_mortgage_cdf             float64
home_equity_cdf                 float64
debt_cdf                        float64
hs_degree                       float64
hs_degree_male                  float64
hs_degree_female                float64
male_age_mean                   float64
male_age_median                 float64
male_age_stdev                  float64
male_age_sample_weight          float64
male_age_samples                float64
female_age_mean                 float64
female_age_median               float64
female_age_stdev                float64
female_age_sample_weight        float64
female_age_samples              float64
pct_own                         float64
married                         float64
```

```
married_snp                       float64
separated                         float64
divorced                          float64
dtype: object
```

In [10]:

```
train_df.columns[:5]
```

Out[10]:

```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID'], dtype='object')
```

In [11]:

```
for i in range(0, len(np.array_split(train_df.dtypes, 5))):
    print((np.array_split(train_df.dtypes, 5)[i]))
    print()
```

```
UID            int64
BLOCKID      float64
SUMLEVEL       int64
COUNTYID       int64
STATEID        int64
state         object
state_ab      object
city          object
place         object
type          object
primary       object
zip_code       int64
area_code      int64
lat          float64
lng          float64
ALand        float64
dtype: object

AWater                   int64
pop                      int64
male_pop                 int64
female_pop               int64
rent_mean              float64
rent_median            float64
rent_stdev             float64
rent_sample_weight     float64
rent_samples           float64
rent_gt_10             float64
rent_gt_15             float64
rent_gt_20             float64
rent_gt_25             float64
rent_gt_30             float64
rent_gt_35             float64
rent_gt_40             float64
dtype: object

rent_gt_50               float64
universe_samples           int64
used_samples               int64
hi_mean                  float64
hi_median                float64
hi_stdev                 float64
hi_sample_weight         float64
hi_samples               float64
family_mean              float64
family_median            float64
family_stdev             float64
family_sample_weight     float64
family_samples           float64
hc_mortgage_mean         float64
hc_mortgage_median       float64
hc_mortgage_stdev        float64
dtype: object
```

```
hc_mortgage_sample_weight      float64
hc_mortgage_samples            float64
hc_mean                        float64
hc_median                      float64
hc_stdev                       float64
hc_samples                     float64
hc_sample_weight               float64
home_equity_second_mortgage    float64
second_mortgage                float64
home_equity                    float64
debt                           float64
second_mortgage_cdf            float64
home_equity_cdf                float64
debt_cdf                       float64
hs_degree                      float64
hs_degree_male                 float64
dtype: object

hs_degree_female            float64
male_age_mean               float64
male_age_median             float64
male_age_stdev              float64
male_age_sample_weight      float64
male_age_samples            float64
female_age_mean             float64
female_age_median           float64
female_age_stdev            float64
female_age_sample_weight    float64
female_age_samples          float64
pct_own                     float64
married                     float64
married_snp                 float64
separated                   float64
divorced                    float64
dtype: object
```

In [12]:

```
train_df[train_df.columns[0:20]].head()
```

Out[12]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_c
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	

In [13]:

```
for i in range(0, len(train_df.columns), 20):
    print(train_df[train_df.columns[i:i+20]].head())
    print()
```

```
      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 primary  zip_code  area_code        lat  \
0    Hamilton        Hamilton   City   tract     13346        315  42.840812
1  South Bend        Roseland   City   tract     46616        574  41.701441
2    Danville        Danville   City   tract     46122        317  39.792202
3    San Juan        Guaynabo  Urban   tract       927        787  18.396103
4   Manhattan  Manhattan City   City   tract     66502        785  39.195573

        lng        ALand   AWater   pop  male_pop  female_pop
0 -75.501524  202183361.0  1699120  5230      2612        2618
1 -86.266614    1560828.0   100363  2633      1349        1284
2 -86.515246   69561595.0   284193  6881      3643        3238
3 -66.104169    1105793.0        0  2700      1141        1559
4 -96.569366    2554403.0        0  5637      2586        3051

   rent_mean  rent_median  rent_stdev  rent_sample_weight  rent_samples  \
0  769.38638        784.0   232.63967           272.34441         362.0
1  804.87924        848.0   253.46747           312.58622         513.0
2  742.77365        703.0   323.39011           291.85520         378.0
3  803.42018        782.0   297.39258           259.30316         368.0
4  938.56493        881.0   392.44096          1005.42886        1704.0

   rent_gt_10  rent_gt_15  rent_gt_20  rent_gt_25  rent_gt_30  rent_gt_35  \
0     0.86761     0.79155     0.59155     0.45634     0.42817     0.18592
1     0.97410     0.93227     0.69920     0.69920     0.55179     0.41235
2     0.95238     0.88624     0.79630     0.66667     0.39153     0.39153
3     0.94693     0.87151     0.69832     0.61732     0.51397     0.46927
4     0.99286     0.98247     0.91688     0.84740     0.78247     0.60974

   rent_gt_40  rent_gt_50  universe_samples  used_samples      hi_mean  \
0     0.15493     0.12958               387           355  63125.28406
1     0.39044     0.27888               542           502  41931.92593
2     0.28307     0.15873               459           378  84942.68317
3     0.35754     0.32961               438           358  48733.67116
4     0.55455     0.44416              1725          1540  31834.15466

   hi_median     hi_stdev  hi_sample_weight  hi_samples
0    48120.0  49042.01206        1290.96240      2024.0
1    35186.0  31639.50203         838.74664      1127.0
2    74964.0  56811.62186        1155.20980      2488.0
3    37845.0  45100.54010         928.32193      1267.0
4    22497.0  34046.50907        1548.67477      1983.0

   family_mean  family_median  family_stdev  family_sample_weight  \
0  67994.14790        53245.0   47667.30119             884.33516
1  50670.10337        43023.0   34715.57548             375.28798
2  95262.51431        85395.0   49292.67664             709.74925
3  56401.68133        44399.0   41082.90515             490.18479
4  54053.42396        50272.0   39609.12605             244.08903

   family_samples  hc_mortgage_mean  hc_mortgage_median  hc_mortgage_stdev  \
0          1491.0        1414.80295              1223.0          641.22898
1           554.0         864.41390               784.0          482.27020
2          1889.0        1506.06758              1361.0          731.89394
3           729.0        1175.28642              1101.0          428.98751
4           395.0        1192.58759              1125.0          327.49674

   hc_mortgage_sample_weight  hc_mortgage_samples    hc_mean  hc_median  \
0                  377.83135                867.0  570.01530      558.0
1                  316.88320                356.0  351.98293      336.0
2                  699.41354               1491.0  556.45986      532.0
3                  261.28471                437.0  288.04047      247.0
4                   76.61052                134.0  443.68855      444.0

    hc_stdev  hc_samples  hc_sample_weight  home_equity_second_mortgage  \
0  270.11299       770.0         499.29293                      0.01588
```

```
1  125.40457       229.0       189.60606                         0.02222
2  184.42175       538.0       323.35354                         0.00000
3  185.55887       392.0       314.90566                         0.01086
4   76.12674       124.0        79.55556                         0.05426

   second_mortgage  home_equity     debt  second_mortgage_cdf
0          0.02077      0.08919  0.52963              0.43658
1          0.02222      0.04274  0.60855              0.42174
2          0.00000      0.09512  0.73484              1.00000
3          0.01086      0.01086  0.52714              0.53057
4          0.05426      0.05426  0.51938              0.18332

   home_equity_cdf  debt_cdf  hs_degree  hs_degree_male  hs_degree_female  \
0          0.49087   0.73341    0.89288         0.85880           0.92434
1          0.70823   0.58120    0.90487         0.86947           0.94187
2          0.46332   0.28704    0.94288         0.94616           0.93952
3          0.82530   0.73727    0.91500         0.90755           0.92043
4          0.65545   0.74967    1.00000         1.00000           1.00000

   male_age_mean  male_age_median  male_age_stdev  male_age_sample_weight  \
0       42.48574         44.00000        22.97306               696.42136
1       34.84728         32.00000        20.37452               323.90204
2       39.38154         40.83333        22.89769               888.29730
3       48.64749         48.91667        23.05968               274.98956
4       26.07533         22.41667        11.84399              1296.89877

   male_age_samples  female_age_mean  female_age_median  female_age_stdev  \
0            2612.0         44.48629           45.33333          22.51276
1            1349.0         36.48391           37.58333          23.43353
2            3643.0         42.15810           42.83333          23.94119
3            1141.0         47.77526           50.58333          24.32015
4            2586.0         24.17693           21.58333          11.10484

   female_age_sample_weight  female_age_samples  pct_own  married  \
0                 685.33845              2618.0  0.79046  0.57851
1                 267.23367              1284.0  0.52483  0.34886
2                 707.01963              3238.0  0.85331  0.64745
3                 362.20193              1559.0  0.65037  0.47257
4                1854.48652              3051.0  0.13046  0.12356

   married_snp  separated  divorced
0      0.01882    0.01240   0.08770
1      0.01426    0.01426   0.09030
2      0.02830    0.01607   0.10657
3      0.02021    0.02021   0.10106
4      0.00000    0.00000   0.03109
```

In [14]:

```
cat_columns = ['UID', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
'primary', 'zip_code', 'area_code']
```

In [15]:

```
train_df[cat_columns].dtypes
```

Out[15]:

```
UID           int64
COUNTYID      int64
STATEID       int64
state        object
state_ab     object
city         object
place        object
type         object
primary      object
zip_code      int64
area_code     int64
dtype: object
```

In [16]:

```
for col in cat_columns:
    print(col)
    print(train_df[col].nunique())
    print(train_df[col].unique())
    print()
```

```
UID
27161
[267822 246444 245683 ... 233000 287425 265371]

COUNTYID
296
[ 53 141  63 127 161  79 337  45  81  37  73  51  25 121  99 153  19  47
 209   3  97  69   7  89   1   5  13  86   9 101 183  67  35 115  29  17
  77  65  93  41 109 155  59 439 133 117 215  33  71  15  11  21 291  31
  95  75  91 163 491  27 129 113  55 111  49  57 105 123 241 197 290  83
 157 135  20  43  39 145 245 329 201 191 143  61 361 103 171 227 137 119
 449 131  85 231 221 147 740 810 189 213 670 177 257 477 317 159 169 173
 151  87 165 355 107 453 590 650 125 193  23 510 267 217 710 187 175 251
 167 139 347 233 179 479 321 313 149 339 427 680 277 325 770  78 459 195
 820 463 700 287 600 341 150 293 375 540 185 281 199 181 170 423 255 219
 373 481 305 261 405 122 265  14 282 800 349  90 401 730 247 307 379 445
 387 760 110 457  28 550 451 499 295 203 467 630 309 223 465 303 381 363
 235 301 207 473 485 333 455 237 367 253 353 158 229 259 441 505 263 471
 683 489 409 297 397 775 205 335 299 285 225 198 239   6 415 437 425 497
 507 580 130 520 220 357 475  50 391 365 311 275 417 595 735 493 369 283
  12 530 750 469 249 211 186 790 431 269 399 315 279 323 495 271 421 570
 411 343 403 389 371 395 610 503 461  68 620 230 351  54 840 720 487 273
 429 640 393 660 331 377 164 180]

STATEID
52
[36 18 72 20  1 48 45  6  5 24 17 19 47 32 22  8 44 28 34 41  4 12 55 42
 37 51 26 39 40 13 16 46 27 29 53 56  9 54 21 25 11 15 30  2 33 49 50 31
 38 35 23 10]

state
52
['New York' 'Indiana' 'Puerto Rico' 'Kansas' 'Alabama' 'Texas'
 'South Carolina' 'California' 'Arkansas' 'Maryland' 'Illinois' 'Iowa'
 'Tennessee' 'Nevada' 'Louisiana' 'Colorado' 'Rhode Island' 'Mississippi'
 'New Jersey' 'Oregon' 'Arizona' 'Florida' 'Wisconsin' 'Pennsylvania'
 'North Carolina' 'Virginia' 'Michigan' 'Ohio' 'Oklahoma' 'Georgia'
 'Idaho' 'South Dakota' 'Minnesota' 'Missouri' 'Washington' 'Wyoming'
 'Connecticut' 'West Virginia' 'Kentucky' 'Massachusetts'
 'District of Columbia' 'Hawaii' 'Montana' 'Alaska' 'New Hampshire' 'Utah'
 'Vermont' 'Nebraska' 'North Dakota' 'New Mexico' 'Maine' 'Delaware']

state_ab
52
['NY' 'IN' 'PR' 'KS' 'AL' 'TX' 'SC' 'CA' 'AR' 'MD' 'IL' 'IA' 'TN' 'NV'
 'LA' 'CO' 'RI' 'MS' 'NJ' 'OR' 'AZ' 'FL' 'WI' 'PA' 'NC' 'VA' 'MI' 'OH'
 'OK' 'GA' 'ID' 'SD' 'MN' 'MO' 'WA' 'WY' 'CT' 'WV' 'KY' 'MA' 'DC' 'HI'
 'MT' 'AK' 'NH' 'UT' 'VT' 'NE' 'ND' 'NM' 'ME' 'DE']

city
6916
['Hamilton' 'South Bend' 'Danville' ... 'Blue Bell' 'Weldona'
 'Colleyville']

place
9912
['Hamilton' 'Roseland' 'Danville' ... 'Cresco City' 'Saddle Ridge'
 'Colleyville City']

type
6
['City' 'Urban' 'Town' 'CDP' 'Village' 'Borough']
```

```
primary
1
['tract']

zip_code
12744
[13346 46616 46122 ... 19422 80653 76034]

area_code
274
[315 574 317 787 785 256 940 864 718 310 323 619 501 410 469 815 515 615
 217 512 702 337 970 401 662 609 503 661 480 305 920 215 919 540 843 734
 937 210 504 405 989 334 607 760 209 951 336 865 520 208 870 605 928 910
 714 626 507 417 682 909 601 510 931 218 541 918 916 956 206 239 307 754
 925 484 203 304 828 330 719 720 765 419 773 859 856 413 202 415 518 812
 530 508 716 434 513 707 803 808 406 810 770 360 614 303 509 409 630 423
 907 973 252 201 732 440 228 603 651 281 386 801 352 802 425 806 717 318
 432 618 412 724 254 772 602 502 308 610 813 402 775 678 817 913 701 216
 713 580 361 706 562 325 251 214 631 915 818 570 270 727 972 248 980 573
 301 517 740 850 559 914 903 941 708 586 262 505 650 912 617 585 435 408
 660 757 800 205 608 860 207 863 213 314 479 309 606 804 901 212 612 385
 908 979 260 704 253 319 331 316 414 858 805 715 269 816 954 832 985 219
 845 731 321 952 814 320 949 231 712 516 904 347 302 225 906 847 763 404
 561 978 478 831 781 563 646 936 703 636 575 407 313 623 641 229 616 424
 830 620 276 774 267 475 443 877 571 888 240 866 555 917 786 862 224 312
 481 855 857 848]
```

In [17]:

```
train_df.isnull().sum(axis = 0)
```

Out[17]:

```
UID                          0
BLOCKID                  27321
SUMLEVEL                     0
COUNTYID                     0
STATEID                      0
state                        0
state_ab                     0
city                         0
place                        0
type                         0
primary                      0
zip_code                     0
area_code                    0
lat                          0
lng                          0
ALand                        0
AWater                       0
pop                          0
male_pop                     0
female_pop                   0
rent_mean                  314
rent_median                314
rent_stdev                 314
rent_sample_weight         314
rent_samples               314
rent_gt_10                 314
rent_gt_15                 314
rent_gt_20                 314
rent_gt_25                 314
rent_gt_30                 314
rent_gt_35                 314
rent_gt_40                 314
rent_gt_50                 314
universe_samples             0
used_samples                 0
hi_mean                    268
```

```
hi_median                        268
hi_stdev                         268
hi_sample_weight                 268
hi_samples                       268
family_mean                      298
family_median                    298
family_stdev                     298
family_sample_weight             298
family_samples                   298
hc_mortgage_mean                 573
hc_mortgage_median               573
hc_mortgage_stdev                573
hc_mortgage_sample_weight        573
hc_mortgage_samples              573
hc_mean                          600
hc_median                        600
hc_stdev                         600
hc_samples                       600
hc_sample_weight                 600
home_equity_second_mortgage      457
second_mortgage                  457
home_equity                      457
debt                             457
second_mortgage_cdf              457
home_equity_cdf                  457
debt_cdf                         457
hs_degree                        190
hs_degree_male                   200
hs_degree_female                 223
male_age_mean                    189
male_age_median                  189
male_age_stdev                   189
male_age_sample_weight           189
male_age_samples                 189
female_age_mean                  206
female_age_median                206
female_age_stdev                 206
female_age_sample_weight         206
female_age_samples               206
pct_own                          268
married                          191
married_snp                      191
separated                        191
divorced                         191
dtype: int64
```

In [18]:

```
train_df.isnull().sum(axis = 0)[20:30]
```

Out[18]:

```
rent_mean              314
rent_median            314
rent_stdev             314
rent_sample_weight     314
rent_samples           314
rent_gt_10             314
rent_gt_15             314
rent_gt_20             314
rent_gt_25             314
rent_gt_30             314
dtype: int64
```

In [19]:

```
train_df.shape
```

Out[19]:

```
(27321, 80)
```

Columns : ['BLOCKID', 'Primary'] can be removed as "BLOCKID" is missing values in all rows and "Primary" has no variance as it has only 1 value.

In [20]:

```
len(train_df.columns[train_df.isnull().sum(axis = 0) > 0])
```

Out[20]:

```
59
```

In [21]:

```
import helpers_py as hf
```

In [22]:

```
hf.miss_df(train_df)
```

Out[22]:

	count	percentage
UID	0	0.00
BLOCKID	27321	100.00
SUMLEVEL	0	0.00
COUNTYID	0	0.00
STATEID	0	0.00
state	0	0.00
state_ab	0	0.00
city	0	0.00
place	0	0.00
type	0	0.00
primary	0	0.00
zip_code	0	0.00
area_code	0	0.00
lat	0	0.00
lng	0	0.00
ALand	0	0.00
AWater	0	0.00
pop	0	0.00

male_pop	0	0.00
female_pop	0	0.00
rent_mean	314	1.15
rent_median	314	1.15
rent_stdev	314	1.15
rent_sample_weight	314	1.15
rent_samples	314	1.15
rent_gt_10	314	1.15
rent_gt_15	314	1.15
rent_gt_20	314	1.15
rent_gt_25	314	1.15
rent_gt_30	314	1.15
rent_gt_35	314	1.15
rent_gt_40	314	1.15
rent_gt_50	314	1.15
universe_samples	0	0.00
used_samples	0	0.00
hi_mean	268	0.98
hi_median	268	0.98
hi_stdev	268	0.98
hi_sample_weight	268	0.98
hi_samples	268	0.98
family_mean	298	1.09
family_median	298	1.09
family_stdev	298	1.09
family_sample_weight	298	1.09
family_samples	298	1.09
hc_mortgage_mean	573	2.10
hc_mortgage_median	573	2.10
hc_mortgage_stdev	573	2.10
hc_mortgage_sample_weight	573	2.10
hc_mortgage_samples	573	2.10
hc_mean	600	2.20
hc_median	600	2.20
hc_stdev	600	2.20

hc_samples	600	2.20
hc_sample_weight	600	2.20
home_equity_second_mortgage	457	1.67
second_mortgage	457	1.67
home_equity	457	1.67
debt	457	1.67
second_mortgage_cdf	457	1.67
home_equity_cdf	457	1.67
debt_cdf	457	1.67
hs_degree	190	0.70
hs_degree_male	200	0.73
hs_degree_female	223	0.82
male_age_mean	189	0.69
male_age_median	189	0.69
male_age_stdev	189	0.69
male_age_sample_weight	189	0.69
male_age_samples	189	0.69
female_age_mean	206	0.75
female_age_median	206	0.75
female_age_stdev	206	0.75
female_age_sample_weight	206	0.75
female_age_samples	206	0.75
pct_own	268	0.98
married	191	0.70
married_snp	191	0.70
separated	191	0.70
divorced	191	0.70

In [23]:

```
train_df.drop(['BLOCKID', 'primary'], axis=1, inplace=True)
```

In [24]:

```
null_data = train_df[train_df.isnull().any(axis=1)]
null_data
```

Out[24]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	
51	223593	140	19	4	Arizona	AZ	Tucson	Littletown	CDP	85734	520	32.06

94	233040	140	101	8	Colorado	CO	Pueblo	Pueblo City	City	81001	719	38.30
153	263292	140	13	34	New Jersey	NJ	Newark	Silver Lake	City	7107	973	40.77
302	267158	140	47	36	New York	NY	Brooklyn	New York City	City	11215	718	40.65
340	292484	140	25	55	Wisconsin	WI	Madison	Madison City	City	53703	608	43.07
...	...	...	...	...	...	...	...	...	...	...	...	
27127	266321	140	5	36	New York	NY	Bronx	Mount Vernon City	City	10458	718	40.87
27175	235725	140	57	12	Florida	FL	Tampa	Pebble Creek	City	33647	813	28.14
27176	247777	140	61	21	Kentucky	KY	Brownsville	Brownsville City	City	42210	270	37.19
27216	266166	140	5	36	New York	NY	Bronx	Pelham Manor	City	10462	718	40.85
27240	251078	140	25	25	Massachusetts	MA	Boston	Brookline	City	2124	617	42.30

736 rows × 78 columns

In [25]:

```
round((736 / 27321)*100, 2)
```

Out[25]:

```
2.69
```

Since we only have 2.69% data missing, we can safely delete these rows, without loosing much information.

In [26]:

```
train_df.shape
```

Out[26]:

```
(27321, 78)
```

In [27]:

```
train_df = pd.concat([train_df, null_data, null_data]).drop_duplicates(keep=False)
```

In [28]:

```
train_df.shape
```

Out[28]:

```
(26585, 78)
```

In [29]:

```
len(train_df.columns[train_df.isnull().sum(axis = 0) > 0])
```

Out[29]:

```
0
```

In [30]:

```
cat_columns = ['UID', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
'zip_code', 'area_code']
```

In [31]:

```
## doing a loop
for col in cat_columns:
    train_df[col] = train_df[col].astype('category')
```

In [32]:

```
train_df.dtypes
```

Out[32]:

```
UID                               category
SUMLEVEL                             int64
COUNTYID                          category
STATEID                           category
state                             category
state_ab                          category
city                              category
place                             category
type                              category
zip_code                          category
area_code                         category
lat                                float64
lng                                float64
ALand                              float64
AWater                               int64
pop                                  int64
male_pop                             int64
female_pop                           int64
rent_mean                          float64
rent_median                        float64
rent_stdev                         float64
rent_sample_weight                 float64
rent_samples                       float64
rent_gt_10                         float64
rent_gt_15                         float64
rent_gt_20                         float64
rent_gt_25                         float64
rent_gt_30                         float64
rent_gt_35                         float64
rent_gt_40                         float64
rent_gt_50                         float64
universe_samples                     int64
used_samples                         int64
hi_mean                            float64
hi_median                          float64
hi_stdev                           float64
hi_sample_weight                   float64
hi_samples                         float64
family_mean                        float64
family_median                      float64
family_stdev                       float64
family_sample_weight               float64
family_samples                     float64
hc_mortgage_mean                   float64
hc_mortgage_median                 float64
hc_mortgage_stdev                  float64
```

```
hc_mortgage_sample_weight       float64
hc_mortgage_samples             float64
hc_mean                         float64
hc_median                       float64
hc_stdev                        float64
hc_samples                      float64
hc_sample_weight                float64
home_equity_second_mortgage     float64
second_mortgage                 float64
home_equity                     float64
debt                            float64
second_mortgage_cdf             float64
home_equity_cdf                 float64
debt_cdf                        float64
hs_degree                       float64
hs_degree_male                  float64
hs_degree_female                float64
male_age_mean                   float64
male_age_median                 float64
male_age_stdev                  float64
male_age_sample_weight          float64
male_age_samples                float64
female_age_mean                 float64
female_age_median               float64
female_age_stdev                float64
female_age_sample_weight        float64
female_age_samples              float64
pct_own                         float64
married                         float64
married_snp                     float64
separated                       float64
divorced                        float64
dtype: object
```

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 [33]:

```
train_df.nlargest(2500, ['second_mortgage', 'pct_own'])
```

Out[33]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code
14014	264403	140	31	34	New Jersey	NJ	Passaic	Garfield City	City	7055	973

3285	289712	140	147	51	Virginia	VA	Farmville	Farmville	Town	23901	434	3
21706	222830	140	13	4	Arizona	AZ	Scottsdale	Tempe City	CDP	85257	480	3
11980	251185	140	27	25	Massachusetts	MA	Worcester	Worcester City	City	1610	508	4
12896	278178	140	101	42	Pennsylvania	PA	Philadelphia	Millbourne	Borough	19104	215	3
...	...	...	...	...	...	...	...	...	...	...	...	
9223	245335	140	3	18	Indiana	IN	Fort Wayne	Fort Wayne City	City	46814	260	4
24579	260417	140	81	37	North Carolina	NC	High Point	Jamestown	Village	27265	336	3
19475	286364	140	257	48	Texas	TX	Crandall	Talty	Town	75114	972	3
13270	287041	140	397	48	Texas	TX	Royse City	Fate City	Town	75189	972	3
22594	225435	140	37	6	California	CA	Los Angeles	South Pasadena City	City	90042	323	

2500 rows × 78 columns

In [34]:

```
top_2500 = train_df[['state', 'lat', 'lng', 'second_mortgage', 'pct_own', 'place',
'state', 'city', 'COUNTYID', 'STATEID', 'home_equity', 'home_equity_second_mortgage',
'debt', 'hi_median', 'family_median']].nlargest(2563, ['second_mortgage', 'pct_own'])
top_2500
```

Out[34]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STAT
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	
21706	Arizona	33.458658	-111.955104	0.43750	0.05660	Tempe City	Arizona	Scottsdale	13	
11980	Massachusetts	42.254262	-71.800347	0.43363	0.20247	Worcester City	Massachusetts	Worcester	27	
12896	Pennsylvania	39.952954	-75.202767	0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	
...	...	...	...	...	...	...	...	...	...	
24443	California	37.732143	-121.242902	0.06814	0.67116	Manteca City	California	Manteca	77	
8377	Florida	25.550391	-80.347791	0.06813	0.50519	Cutler Bay	Florida	Cutler Bay	86	
16621	Texas	32.913822	-97.204310	0.06812	0.97987	Keller City	Texas	Keller	439	
13987	Ohio	39.556756	-84.443252	0.06812	0.92888	Jacksonburg	Ohio	Middletown	17	
14857	New Jersey	39.432879	-74.686137	0.06810	0.70642	Mays Landing	New Jersey	Mays Landing	1	

2563 rows × 15 columns

In [35]:

```
top_2500.pct_own.unique
```

Out[35]:

```
<bound method Series.unique of 14014    0.01157
3285     0.62069
21706    0.05660
11980    0.20247
12896    0.05041
          ...
24443    0.67116
8377     0.50519
16621    0.97987
13987    0.92888
14857    0.70642
Name: pct own, Length: 2563, dtype: float64>
```

In [36]:

```
train_df[train_df.pct_own > 0.1]
```

Out[36]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.8
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.7
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.3
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.1
...	...	...	...	...	...	...	...	...	...	...	...	
27316	279212	140	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	769	787	18.0
27317	277856	140	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	19422	215	40.
27318	233000	140	87	8	Colorado	CO	Weldona	Saddle Ridge	City	80653	970	40.
27319	287425	140	439	48	Texas	TX	Colleyville	Colleyville City	Town	76034	817	32.9
27320	265371	140	3	32	Nevada	NV	Las Vegas	Paradise	City	89123	702	36.0

26215 rows × 78 columns

In [37]:

```
top_2500[top_2500.pct_own > 0.1].head()
```

Out[37]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEID
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	51
11980	Massachusetts	42.254262	-71.800347	0.43363	0.20247	Worcester	Massachusetts	Worcester	27	25

						City				
26018	New York	40.751809	-73.853582	0.31818	0.15618	Harbor Hills	New York	Corona	81	36
7829	Maryland	39.127273	-76.635265	0.30212	0.22380	Glen Burnie	Maryland	Glen Burnie	3	24
2077	Florida	28.029063	-82.495395	0.28972	0.11618	Egypt Lake-leto	Florida	Tampa	57	12

In [38]:

```
import plotly.graph_objects as go
import plotly.figure_factory as ff
```

In [39]:

```
scope = ["USA"]

values = top_2500['second_mortgage'].tolist()

place = top_2500['place'].tolist()
```

In [40]:

```
def zero_prefix(str_list):
    ''' prefixing 0's to numbers. Define the target length of your final number
     Function will add required no. of 0's to meet the target length'''

    str_list = list(map(str, str_list))

    target_length = int(input("Enter Target Length of String: "))

    for i in range(len(str_list)):
        if len(str_list[i]) < target_length:
            str_list[i] = (target_length - len(str_list[i])) * '0'+ str_list[i]

    return str_list

        #elif len(str_list[i]) <= 1:
                #str_list[i] = '00'+ str_list[i]
```

In [41]:

```
z_COUNTYID = zero_prefix(top_2500.COUNTYID)
```

```
Enter Target Length of String: 1
```

In [42]:

```
z_STATEID = zero_prefix(top_2500.STATEID)
```

```
Enter Target Length of String: 1
```

In [43]:

```
top_2500['FIPSID'] = [a + b for a, b in zip(z_STATEID,z_COUNTYID)]
```

In [44]:

```
top_2500.head()
```

Out[44]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEII
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	3

3285	Virginia	37.297357	-78.396452		0.50000	0.62069	Farmville	Virginia	Farmville	147	5
21706	Arizona	33.458658	-111.955104		0.43750	0.05660	Tempe City	Arizona	Scottsdale	13	4
11980	Massachusetts	42.254262	-71.800347		0.43363	0.20247	Worcester City	Massachusetts	Worcester	27	2:
12896	Pennsylvania	39.952954	-75.202767		0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	4:

In [45]:

```
top_2500.dtypes
```

Out[45]:

```
state                          category
lat                             float64
lng                             float64
second_mortgage                 float64
pct_own                         float64
place                          category
state                          category
city                           category
COUNTYID                       category
STATEID                        category
home_equity                     float64
home_equity_second_mortgage     float64
debt                            float64
hi_median                       float64
family_median                   float64
FIPSID                           object
dtype: object
```

In [46]:

```
train_df[col] = train_df[col].astype('category')
```

In [47]:

```
top_2500['FIPSID'] = top_2500['FIPSID'].astype('int64')
```

In [48]:

```
scope = ["USA"]

values = top_2500['second_mortgage'].tolist()

fips = top_2500['FIPSID'].tolist()
```

In [49]:

```
colorscale = ["#8dd3c7", "#ffffb3", "#bebada", "#fb8072",
              "#80b1d3", "#fdb462", "#b3de69", "#fccde5",
              "#d9d9d9", "#bc80bd", "#ccebc5", "#ffed6f",
              "#8dd3c7", "#ffffb3", "#bebada", "#fb8072",
              "#80b1d3", "#fdb462", "#b3de69", "#fccde5",
              "#d9d9d9", "#bc80bd", "#ccebc5", "#ffed6f",
              "#8dd3c7", "#ffffb3", "#bebada", "#fb8072",
              "#80b1d3", "#fdb462", "#b3de69", "#fccde5",
              "#d9d9d9", "#bc80bd", "#ccebc5", "#ffed6f"]

endpts = list(np.linspace(1, 12, len(colorscale) - 1))
```

In [50]:

```
from bokeh.io import output_file, output_notebook, show
from bokeh.models import (
  GMapPlot, GMapOptions, ColumnDataSource, Circle, LogColorMapper, BasicTicker, ColorBar,
```

```
    DataRange1d, PanTool, WheelZoomTool, BoxSelectTool
)

from bokeh.plotting import gmap

from bokeh.models.mappers import ColorMapper, LinearColorMapper
from bokeh.palettes import Viridis5
```

In [51]:

```
map_options = GMapOptions(lat=37.88, lng=-122.23, map_type="roadmap", zoom=6)

plot = gmap( "AIzaSyBYrbp34OohAHsX1cub8ZeHlMEFajv15fY" , map_options=map_options,
                        title = 'Top 2500 Locations'
)

# source = ColumnDataSource(
#     data=dict(lat=[ 30.29,  30.20,  30.29],
#               lon=[-97.70, -97.74, -97.78])
# )

# p.circle(x="lon", y="lat", size=15, fill_color="blue", fill_alpha=0.8, source=source)

# show(p)

source = ColumnDataSource(
    data=dict(
        lat=top_2500.lat.tolist(),
        lon=top_2500.lng.tolist(),
        size=top_2500.second_mortgage.tolist(),
        color=top_2500.pct_own.tolist()
    )
)
max_pct_own = top_2500.loc[top_2500['pct_own'].idxmax()]['pct_own']
min_pct_own = top_2500.loc[top_2500['pct_own'].idxmin()]['pct_own']

#color_mapper = CategoricalColorMapper(factors=['hi', 'lo'], palette=[RdBu3[2], RdBu3[0]])
#color_mapper = LogColorMapper(palette="Viridis5", low=min_median_house_value,
high=max_median_house_value)
color_mapper = LinearColorMapper(palette=Viridis5)

circle = Circle(x="lon", y="lat", size="size", fill_color={'field': 'color', 'transform':
color_mapper}, fill_alpha=0.5, line_color=None)
plot.add_glyph(source, circle)

color_bar = ColorBar(color_mapper=color_mapper, ticker=BasicTicker(),
                     label_standoff=12, border_line_color=None, location=(0,0))
plot.add_layout(color_bar, 'right')

plot.add_tools(PanTool(), WheelZoomTool(), BoxSelectTool())
#output_file("gmap_plot.html")
output_notebook()

show(plot)
```

Loading BokehJS ...

Facing Issues with plotting maps. Will get back at it later

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

In [52]:

```
top_2500['Bad_Debt'] = top_2500['second_mortgage'] + top_2500['home_equity'] -
top_2500['home_equity_second_mortgage']
top_2500['Good_Debt'] = top_2500['debt'] - top_2500['Bad_Debt']
```

In [53]:

```
top_2500['Good_Debt'] = top_2500['debt'] - top_2500['Bad_Debt']
```

In [54]:

```
top_2500.head(15)
```

Out[54]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STAT
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	
21706	Arizona	33.458658	-111.955104	0.43750	0.05660	Tempe City	Arizona	Scottsdale	13	
11980	Massachusetts	42.254262	-71.800347	0.43363	0.20247	Worcester City	Massachusetts	Worcester	27	
12896	Pennsylvania	39.952954	-75.202767	0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	
7453	Texas	30.285534	-97.747727	0.36364	0.01737	Austin City	Texas	Austin	453	
15589	Georgia	33.740759	-84.401777	0.34783	0.04026	Atlanta City	Georgia	Atlanta	121	
1680	Illinois	41.782569	-87.579504	0.33333	0.05267	Chicago City	Illinois	Chicago	31	
26018	New York	40.751809	-73.853582	0.31818	0.15618	Harbor Hills	New York	Corona	81	
23547	California	34.066049	-118.274164	0.31148	0.06960	Vernon City	California	Los Angeles	37	
7829	Maryland	39.127273	-76.635265	0.30212	0.22380	Glen Burnie	Maryland	Glen Burnie	3	
21880	Michigan	42.290397	-85.584144	0.30159	0.07085	Kalamazoo City	Michigan	Kalamazoo	77	
2077	Florida	28.029063	-82.495395	0.28972	0.11618	Egypt Lake-leto	Florida	Tampa	57	
1701	Illinois	41.967289	-87.652434	0.28899	0.14228	Lincolnwood	Illinois	Chicago	31	

11839	Illinois	41.906640	-87.689580	0.27431	0.29468	Chicago City	Illinois	Chicago	31

c) Create pie charts to show overall debt and bad debt.

In [55]:

```
size = 10
explode = [0.4] * size
explode = tuple(explode)
explode

explode_bd = [0.5] * size*2
explode_bd = tuple(explode_bd)
explode_bd

labels_D = ['GD', 'BD'] * size
labels_D = tuple(labels_D)
labels_D
```

Out[55]:

```
('GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD')
```

In [56]:

```
l1 = list(top_2500['Bad_Debt'] )
l1[:5]
```

Out[56]:

```
[0.6087, 0.5, 0.4375, 0.43363, 0.60975]
```

In [57]:

```
l2 = list(top_2500['Good_Debt'] )
l2[:5]
```

Out[57]:

```
[0.0, 0.0, 0.10938000000000003, 0.41592999999999997, 0.32927000000000006]
```

In [58]:

```
l3 = sum(zip(l1, l2+[0]), ())
```

In [59]:

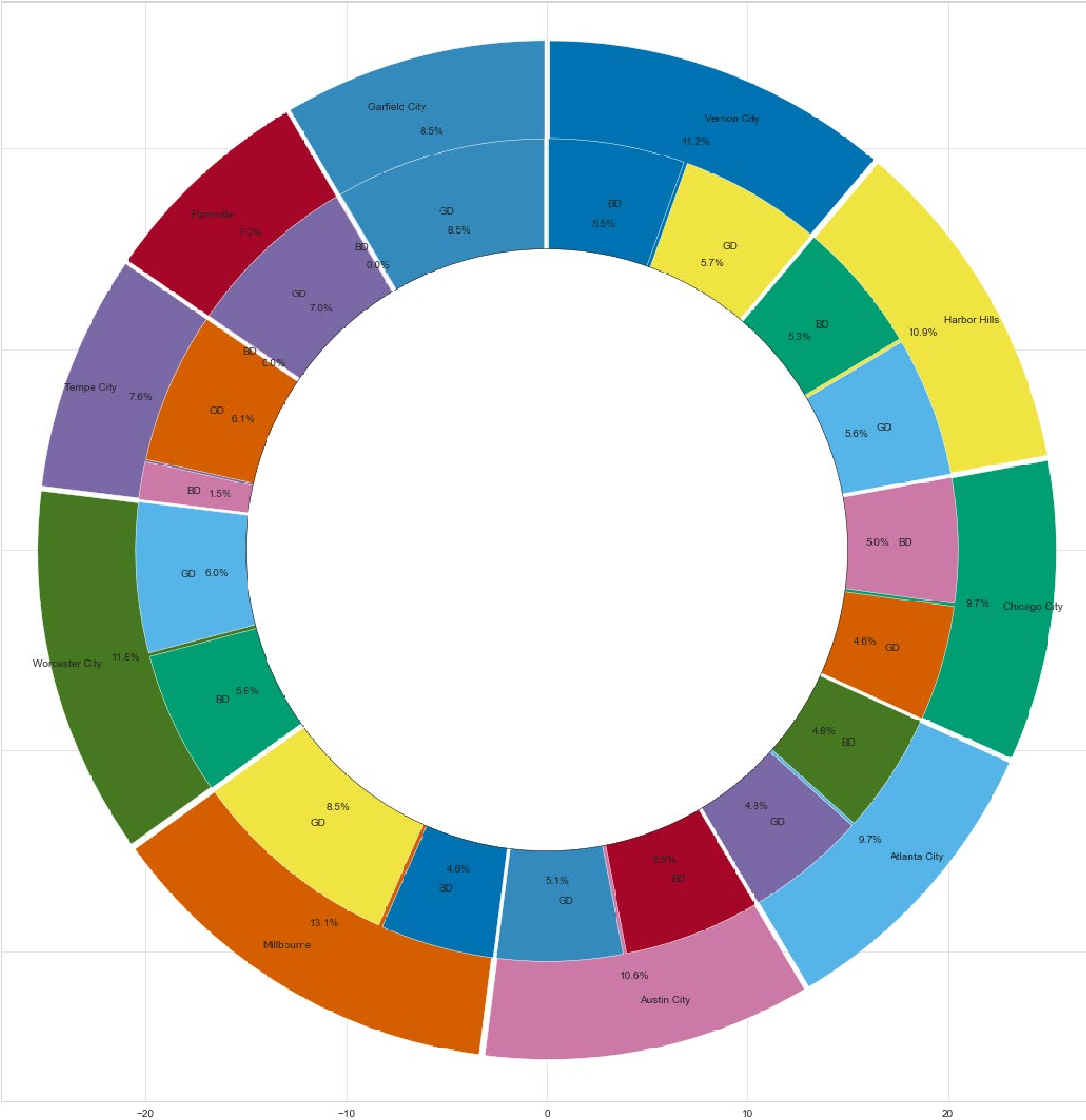
```
l3[:10]
```

Out[59]:

```
(0.6087,
 0.0,
 0.5,
 0.0,
 0.4375,
 0.10938000000000003,
 0.43363,
 0.41592999999999997,
 0.60975,
 0.32927000000000006)
```

In [60]:

```
labels = list(top_2500.place[:10])
debt = list(top_2500.debt[:10])

sns.set_style("whitegrid")

gd_bd = l3[:20]

plt.figure(figsize = (15, 15))

color_pal = plt.rcParams['axes.prop_cycle'].by_key()['color']
#color_cycle = cycle(plt.rcParams['axes.prop_cycle'].by_key()['color'])

plt.pie(debt, labels = labels, startangle = 90, frame = True, radius =25,
autopct='%1.1f%%', pctdistance=0.85, labeldistance = 0.9, colors = color_pal, explode =
explode)
plt.pie(gd_bd, labels = labels_D, startangle = 90, frame = True, radius = 20,
autopct='%1.1f%%', pctdistance=0.80,  labeldistance = 0.85, colors = color_pal, explode =
explode_bd)
centre_circle = plt.Circle((0,0),15,color='black', fc='white',linewidth=0.5)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.tight_layout()
plt.show()
```



Since it is difficult to show all 2500 locations, without compromising readability, I have limited my

selection to "Top 10" cities.

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

In [61]:

```
second_mortgage = list(top_2500.second_mortgage)
home_equity = list(top_2500.home_equity)

Good_Debt = list(top_2500.Good_Debt)
Bad_Debt = list(top_2500.Bad_Debt)
```

In [62]:

```
top_2500['city'].value_counts()[:31].index
```

Out[62]:

```
CategoricalIndex(['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                  'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                  'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                  'Littleton', 'Miami', 'Oakland', 'Houston'],
                 categories=['Abbeville', 'Aberdeen', 'Abilene', 'Abingdon', 'Abington', 'Ac
cokeek', 'Acton', 'Acushnet', ...], ordered=False, dtype='category')
```

In [63]:

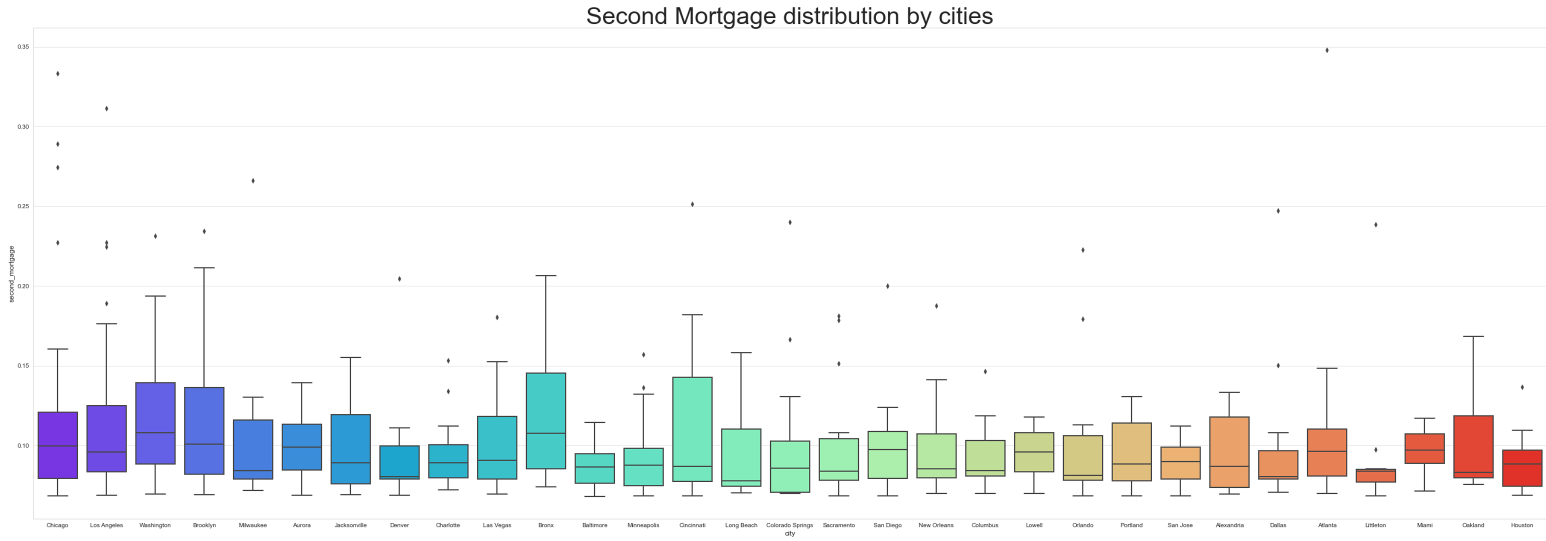
```
cities = ['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                  'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                  'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                  'Littleton', 'Miami', 'Oakland', 'Houston']
```

In [64]:

```
boxplot_df = top_2500[top_2500['city'].isin (cities)]
#rpt[rpt['STK_ID'].isin(stk_list)]
```

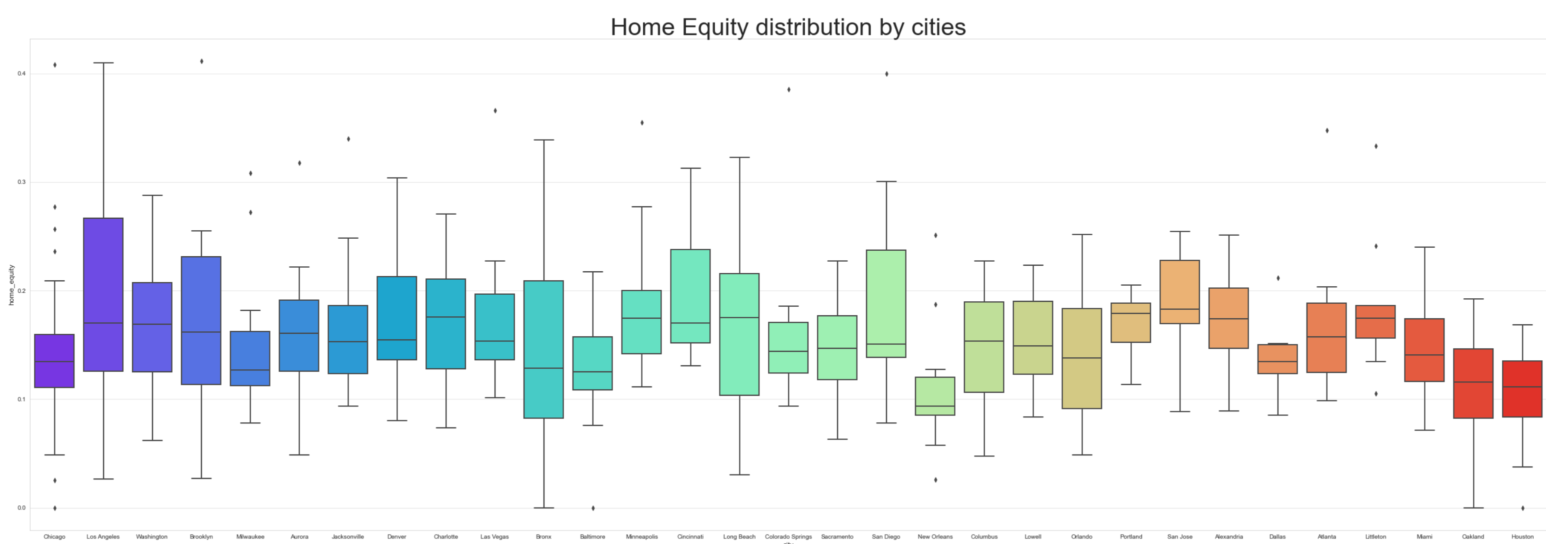
In [65]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 15))
sns.boxplot(x='city',y='second_mortgage',data=boxplot_df,palette='rainbow', order =
['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                  'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                  'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                  'Littleton', 'Miami', 'Oakland', 'Houston']).set_title('Second Mortgage
distribution by cities', fontsize = 40)
plt.show()
```



In [66]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 15))
sns.boxplot(x='city',y='home_equity',data=boxplot_df,palette='rainbow', order =
['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                  'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                  'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                  'Littleton', 'Miami', 'Oakland', 'Houston']).set_title('Home Equity
distribution by cities', fontsize = 40)
plt.show()
```



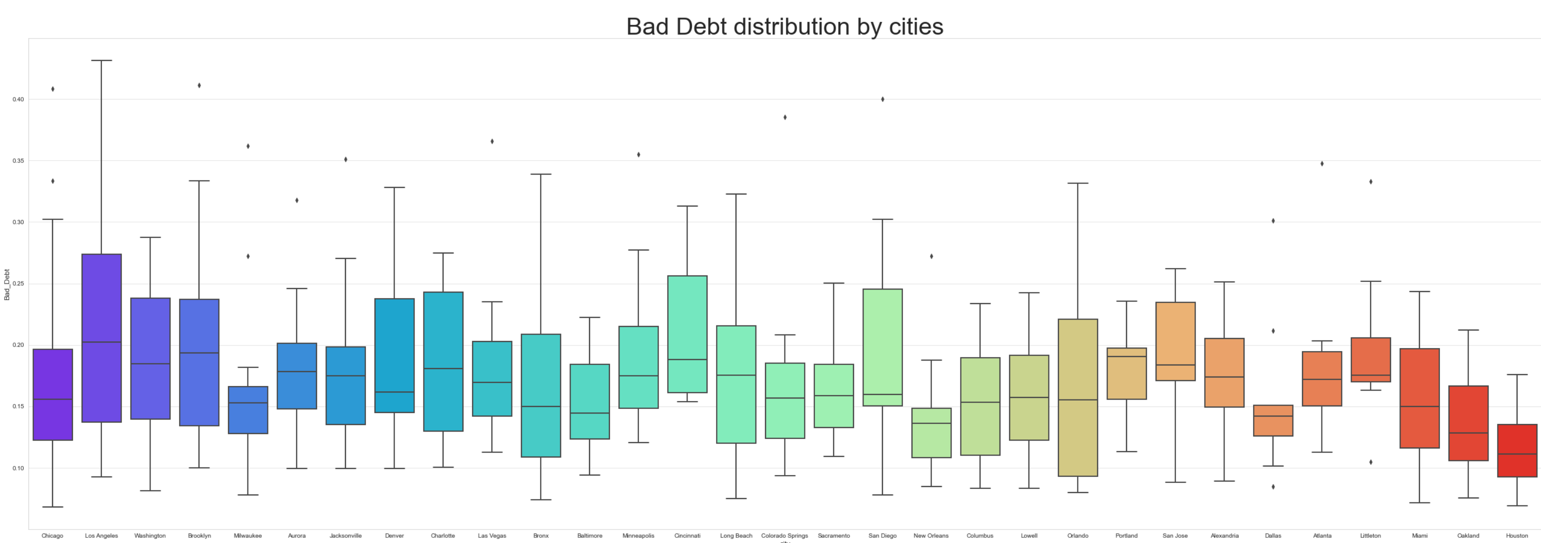
In [67]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 10))
sns.boxplot(x='city',y='Good_Debt',data=boxplot_df,palette='rainbow', order = ['Chicago',
'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
```

```
                    'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                    'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                    'Littleton', 'Miami', 'Oakland', 'Houston']).set_title('Good Debt
distribution by cities', fontsize = 40)
plt.show()
```

In [68]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 15))
sns.boxplot(x='city',y='Bad_Debt',data=boxplot_df,palette='rainbow', order = ['Chicago',
'Los Angeles', 'Washington', 'Brooklyn',
                    'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                    'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                    'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                    'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
                    'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                    'Littleton', 'Miami', 'Oakland', 'Houston']).set_title('Bad Debt
distribution by cities', fontsize = 40)
plt.show()
```



Since it is difficult to show all 2500 locations, without compromising

readability, I have limited my selection to "Top 31" cities.

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

In [69]:

```
top_2500['remaining_income'] = top_2500['family_median'] - top_2500['hi_median']
```

In [70]:

```
income_chart = round(top_2500[['city', 'hi_median', 'family_median', 'remaining_income']],
2)
income_chart
```

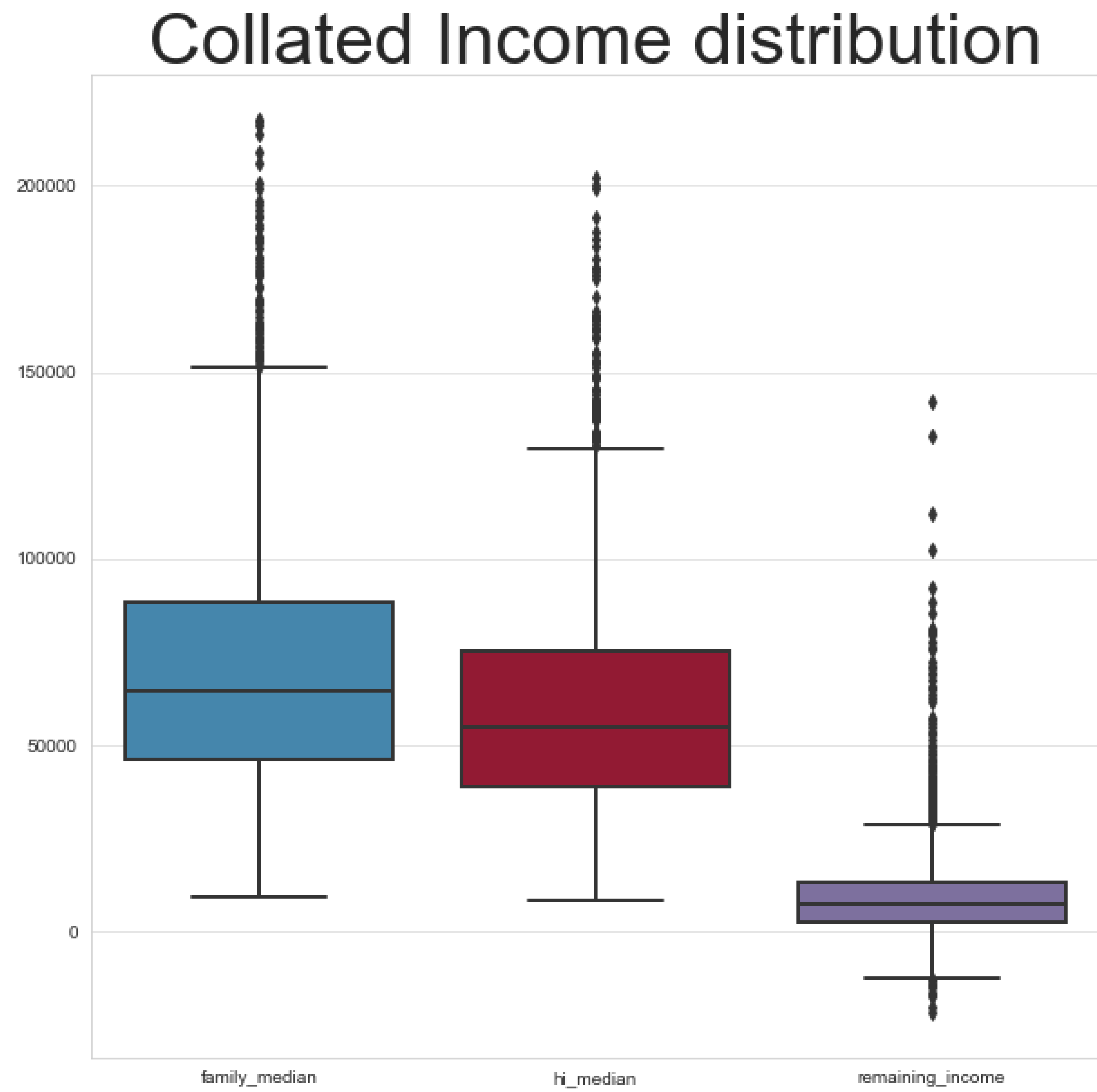
Out[70]:

	city	hi_median	family_median	remaining_income
14014	Passaic	28053.0	29340.0	1287.0
3285	Farmville	23236.0	59954.0	36718.0
21706	Scottsdale	40883.0	59657.0	18774.0
11980	Worcester	29037.0	40476.0	11439.0
12896	Philadelphia	12881.0	50622.0	37741.0
...	...	...	...	...
24443	Manteca	74648.0	76881.0	2233.0
8377	Cutler Bay	50832.0	52547.0	1715.0
16621	Keller	177847.0	177067.0	-780.0
13987	Middletown	72585.0	77338.0	4753.0
14857	Mays Landing	52393.0	61947.0	9554.0

2563 rows × 4 columns

In [71]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (10, 10))
sns.boxplot(data=top_2500[['family_median', 'hi_median', 'remaining_income']],
palette=color_pal).set_title('Collated Income distribution', fontsize = 40)
plt.show()
```



Exploratory Data Analysis (EDA) ...Contd.,

Project Task: Week 2

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):

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

population 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

In [72]:

```
train_df.head()
```

Out[72]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.!
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.:
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	-86.!
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	-66.
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	-96.!

In [73]:

```
density_eda_df = train_df[['state', 'city', 'place', 'ALand', 'pop', 'male_age_median',
'female_age_median', 'male_pop', 'female_pop']]
density_eda_df.head()
```

Out[73]:

	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop
0	New York	Hamilton	Hamilton	202183361.0	5230	44.00000	45.33333	2612	2618
1	Indiana	South Bend	Roseland	1560828.0	2633	32.00000	37.58333	1349	1284
2	Indiana	Danville	Danville	69561595.0	6881	40.83333	42.83333	3643	3238
3	Puerto Rico	San Juan	Guaynabo	1105793.0	2700	48.91667	50.58333	1141	1559
4	Kansas	Manhattan	Manhattan City	2554403.0	5637	22.41667	21.58333	2586	3051

In [74]:

```
density_eda_df['pop_density'] = density_eda_df['pop'] / density_eda_df['ALand']
density_eda_df.head()
```

Out[74]:

	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop	pop_density
0	New York	Hamilton	Hamilton	202183361.0	5230	44.00000	45.33333	2612	2618	0.000026

1	Indiana	South Bend	Roseland	1560828.0	2633	32.00000	37.58333	1349	1284	0.001687
2	Indiana	Danville	Danville	69561595.0	6881	40.83333	42.83333	3643	3238	0.000099
3	Puerto Rico	San Juan	Guaynabo	1105793.0	2700	48.91667	50.58333	1141	1559	0.002442
4	Kansas	Manhattan	Manhattan City	2554403.0	5637	22.41667	21.58333	2586	3051	0.002207

In [75]:

```
density_eda_df['median_age'] = (density_eda_df['male_age_median'] *
density_eda_df['male_pop'] + density_eda_df['female_age_median'] *
density_eda_df['female_pop'])  / density_eda_df['pop']
density_eda_df.head()
```

Out[75]:

	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop	pop_density	n
0	New York	Hamilton	Hamilton	202183361.0	5230	44.00000	45.33333	2612	2618	0.000026	
1	Indiana	South Bend	Roseland	1560828.0	2633	32.00000	37.58333	1349	1284	0.001687	
2	Indiana	Danville	Danville	69561595.0	6881	40.83333	42.83333	3643	3238	0.000099	
3	Puerto Rico	San Juan	Guaynabo	1105793.0	2700	48.91667	50.58333	1141	1559	0.002442	
4	Kansas	Manhattan	Manhattan City	2554403.0	5637	22.41667	21.58333	2586	3051	0.002207	

In [76]:

```
density_eda_df.nlargest(300, 'pop_density')
```

Out[76]:

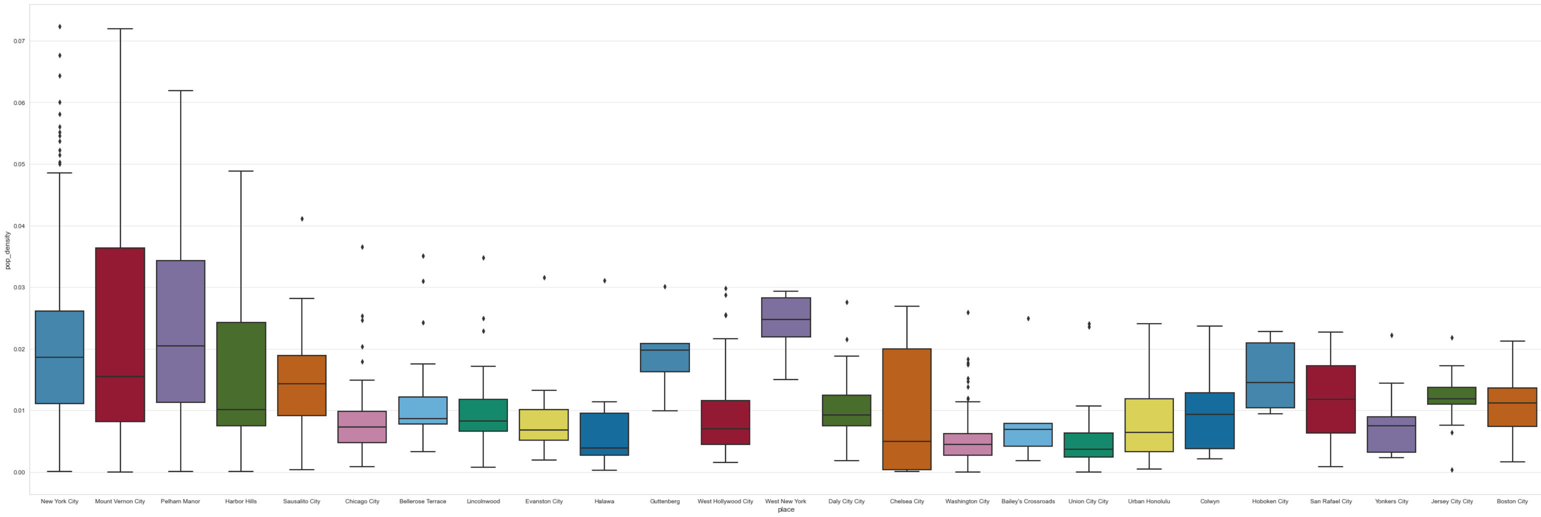
	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop	pop_density
21050	New York	New York	New York City	182091.0	13162	38.83333	34.66667	5597	7565	0.072283
10251	New York	New York	Mount Vernon City	169349.0	12189	33.25000	35.33333	6110	6079	0.071976
1546	New York	New York	New York City	183653.0	12427	37.00000	41.83333	5425	7002	0.067666
23760	New York	New York	New York City	181779.0	11688	39.25000	41.50000	5011	6677	0.064298
13022	New York	Bronx	Mount Vernon City	67355.0	4229	27.75000	26.66667	1932	2297	0.062787
...	...	...	...	...	...	...	...	...	...	...
14705	New Jersey	Guttenberg	Guttenberg	178469.0	3715	33.66667	34.00000	1893	1822	0.020816

706	New York	Brooklyn	New York City	184193.0	3829	29.58333	34.66667	1824	2005	0.020788
16852	New Jersey	Jersey City	Hoboken City	219021.0	4545	30.50000	32.41667	2330	2215	0.020751
8015	New York	Brooklyn	New York City	207813.0	4304	44.00000	47.00000	2196	2108	0.020711
19946	New York	Brooklyn	New York City	165897.0	3418	42.33333	38.83333	1468	1950	0.020603

300 rows × 11 columns

In [77]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df.nlargest(26585,
'pop_density'), palette=color_pal, order = ['New York City',
 'Mount Vernon City',
 'Pelham Manor',
 'Harbor Hills',
 'Sausalito City',
 'Chicago City',
 'Bellerose Terrace',
 'Lincolnwood',
 'Evanston City',
 'Halawa',
 'Guttenberg',
 'West Hollywood City',
 'West New York',
 'Daly City City',
 'Chelsea City',
 'Washington City',
 "Bailey's Crossroads",
 'Union City City',
 'Urban Honolulu',
 'Colwyn',
 'Hoboken City',
 'San Rafael City',
 'Yonkers City',
 'Jersey City City',
 'Boston City'])
plt.show()
```



In [78]:

```
list(density_eda_df.nsmallest(450, 'pop_density').state.unique())
```

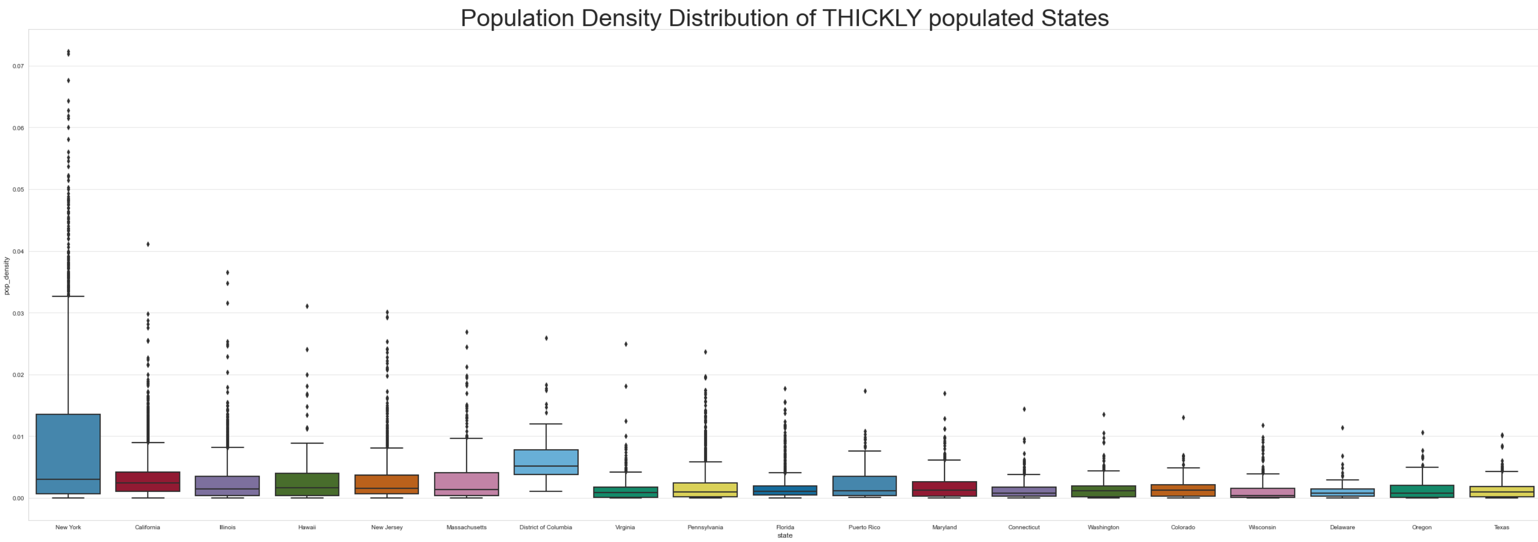
Out[78]:

```
['Alaska',
 'Montana',
 'Utah',
 'Oregon',
 'Nevada',
 'Colorado',
 'Idaho',
 'California',
 'New Mexico',
 'Maine',
 'South Dakota',
 'Wyoming',
 'Nebraska',
 'Texas',
 'Kansas',
 'North Dakota',
 'Arizona',
 'Washington',
 'New York',
 'Oklahoma',
 'Minnesota',
 'Louisiana',
 'Michigan',
 'Florida',
 'Wisconsin',
 'Mississippi',
 'New Hampshire',
 'Georgia',
 'Missouri',
 'Virginia',
 'Alabama',
 'Arkansas']
```

In [79]:

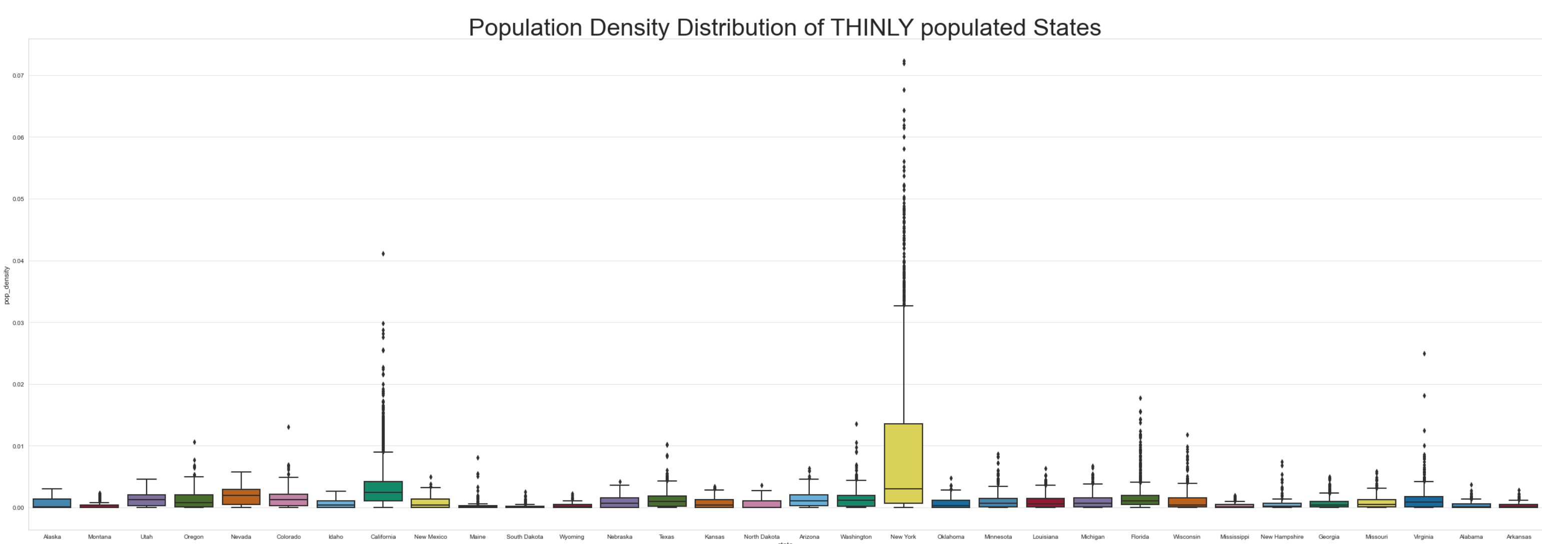
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
sns.boxplot(x = 'state', y = 'pop_density', data=density_eda_df.nlargest(26585,
'pop_density'), palette=color_pal, order = ['New York', 'California', 'Illinois',
'Hawaii', 'New Jersey', 'Massachusetts', 'District of Columbia', 'Virginia',

'Pennsylvania', 'Florida', 'Puerto Rico', 'Maryland', 'Connecticut', 'Washington',
'Colorado', 'Wisconsin',

'Delaware', 'Oregon', 'Texas']).set_title('Population Density Distribution of THICKLY
```

```
populated States', fontsize = 40)
plt.show()
```



In [80]:

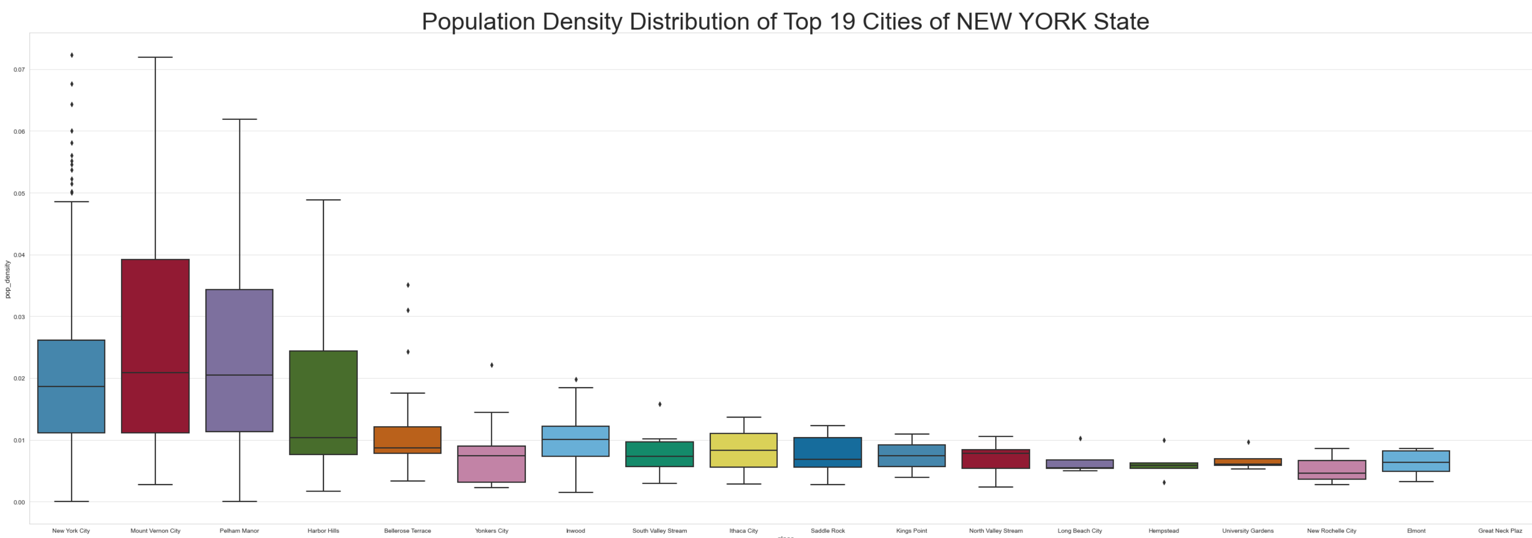
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
sns.boxplot(x = 'state', y = 'pop_density', data=density_eda_df.nsmallest(26585,
'pop_density'), palette=color_pal, order = ['Alaska', 'Montana', 'Utah', 'Oregon',
'Nevada', 'Colorado', 'Idaho', 'California', 'New Mexico',

'Maine', 'South Dakota', 'Wyoming', 'Nebraska', 'Texas', 'Kansas', 'North Dakota',
'Arizona',

'Washington', 'New York', 'Oklahoma', 'Minnesota', 'Louisiana', 'Michigan', 'Florida',
'Wisconsin', 'Mississippi',

'New Hampshire', 'Georgia', 'Missouri', 'Virginia', 'Alabama',
'Arkansas']).set_title('Population Density Distribution of THINLY populated States',
fontsize = 40)
plt.show()
```



In [81]:

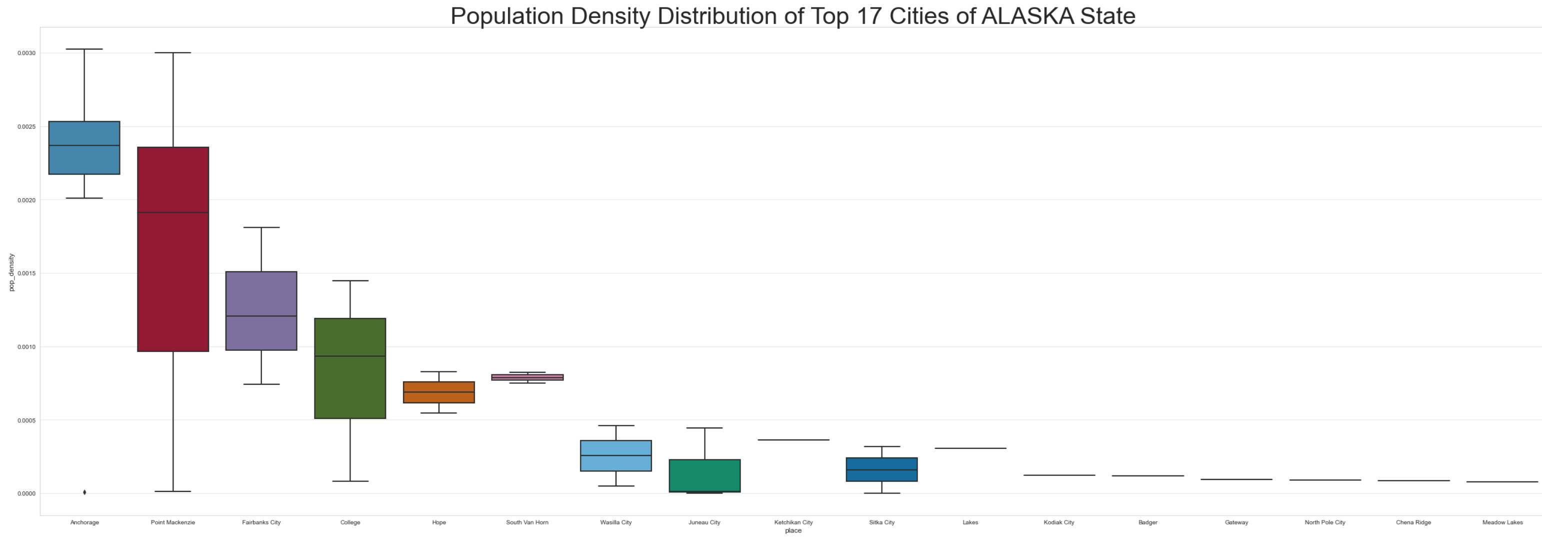
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df[density_eda_df['state'] ==
```

```
'New York'].nlargest(26585, 'pop_density'), palette=color_pal, order = ['New York City',
 'Mount Vernon City',
 'Pelham Manor',
 'Harbor Hills',
 'Bellerose Terrace',
 'Yonkers City',
 'Inwood',
 'South Valley Stream',
 'Ithaca City',
 'Saddle Rock',
 'Kings Point',
 'North Valley Stream',
 'Long Beach City',
 'Hempstead',
 'University Gardens',
 'New Rochelle City',
 'Elmont',
 'Great Neck Plaz']
).set_title('Population Density Distribution of Top 19 Cities of NEW YORK State', fontsize
= 40)
plt.show()
```



In [82]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df[density_eda_df['state'] ==
'Alaska'].nlargest(26585, 'pop_density'), palette=color_pal, order = ['Anchorage', 'Point
Mackenzie', 'Fairbanks City', 'College', 'Hope', 'South Van Horn',

'Wasilla City', 'Juneau City', 'Ketchikan City', 'Sitka City', 'Lakes', 'Kodiak City',
'Badger', 'Gateway', 'North Pole City', 'Chena Ridge', 'Meadow Lakes']
).set_title('Population Density Distribution of Top 17 Cities of ALASKA State', fontsize =
40)
plt.show()
```



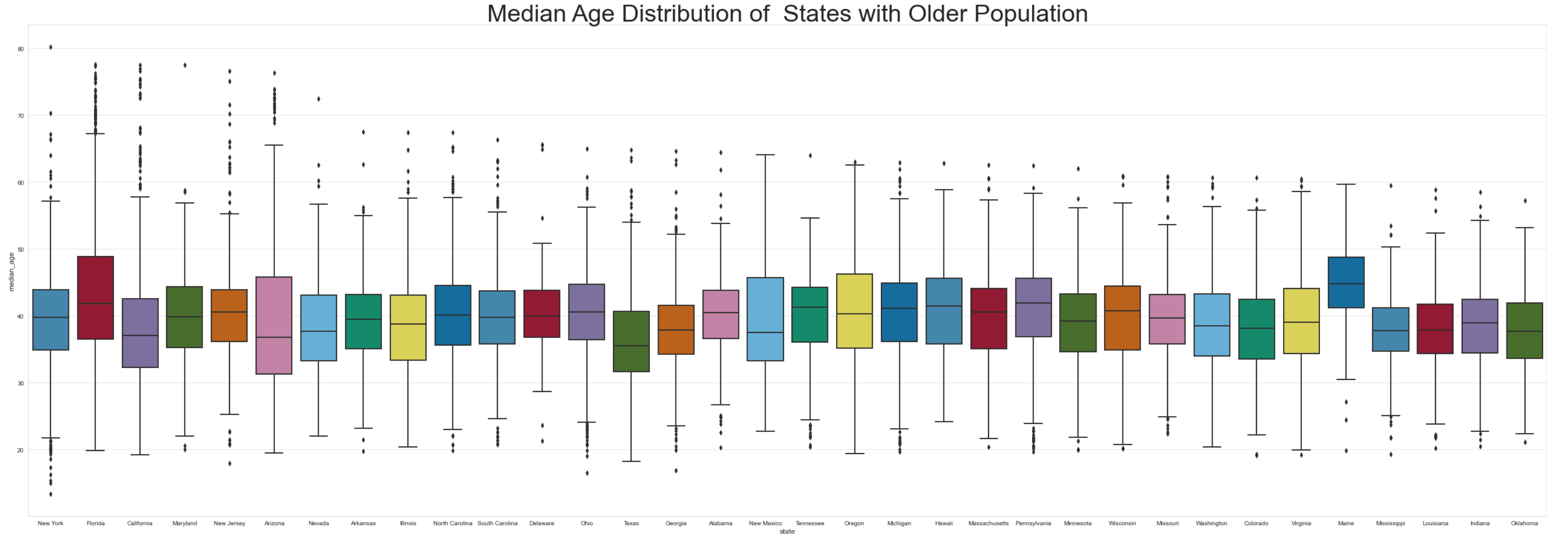
In [83]:

```
print(list(density_eda_df.nlargest(450, 'median_age').state.unique()))
print(len(list(density_eda_df.nlargest(450, 'median_age').state.unique())))
```

```
['New York', 'Florida', 'California', 'Maryland', 'New Jersey', 'Arizona', 'Nevada', 'Arkans
as', 'Illinois', 'North Carolina', 'South Carolina', 'Delaware', 'Ohio', 'Texas', 'Georgia',
 'Alabama', 'New Mexico', 'Tennessee', 'Oregon', 'Michigan', 'Hawaii', 'Massachusetts', 'Pen
nsylvania', 'Minnesota', 'Wisconsin', 'Missouri', 'Washington', 'Colorado', 'Virginia', 'Mai
ne', 'Mississippi', 'Louisiana', 'Indiana', 'Oklahoma']
34
```

In [84]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))


ax = sns.boxplot(x = 'state', y = 'median_age', data=density_eda_df.nlargest(26585,
'median_age'), palette=color_pal,
             order = ['New York', 'Florida', 'California', 'Maryland', 'New Jersey',
'Arizona', 'Nevada', 'Arkansas', 'Illinois', 'North Carolina', 'South Carolina',
'Delaware', 'Ohio', 'Texas', 'Georgia', 'Alabama', 'New Mexico', 'Tennessee',
             'Oregon', 'Michigan', 'Hawaii', 'Massachusetts', 'Pennsylvania', 'Minnesota',
'Wisconsin', 'Missouri', 'Washington', 'Colorado', 'Virginia', 'Maine', 'Mississippi',
'Louisiana', 'Indiana', 'Oklahoma']
            ).set_title('Median Age Distribution of  States with Older Population',
fontsize = 40)

#ax.set(ylim=(0, 100))

plt.show()
```



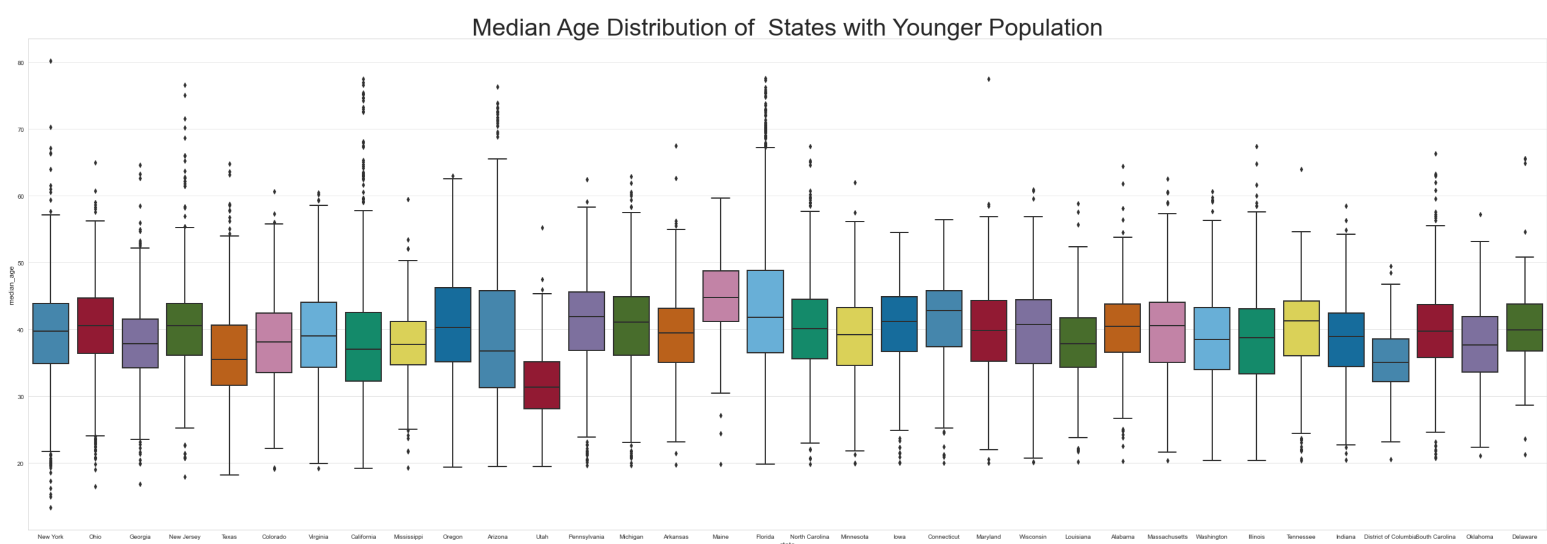
In [85]:

```
print(list(density_eda_df.nsmallest(150, 'median_age').state.unique()))
print(len(list(density_eda_df.nsmallest(150, 'median_age').state.unique())))
```

```
['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California',
 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania', 'Michigan', 'Arkansas', 'Maine'
, 'Florida', 'North Carolina', 'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana', '
District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware']
34
```

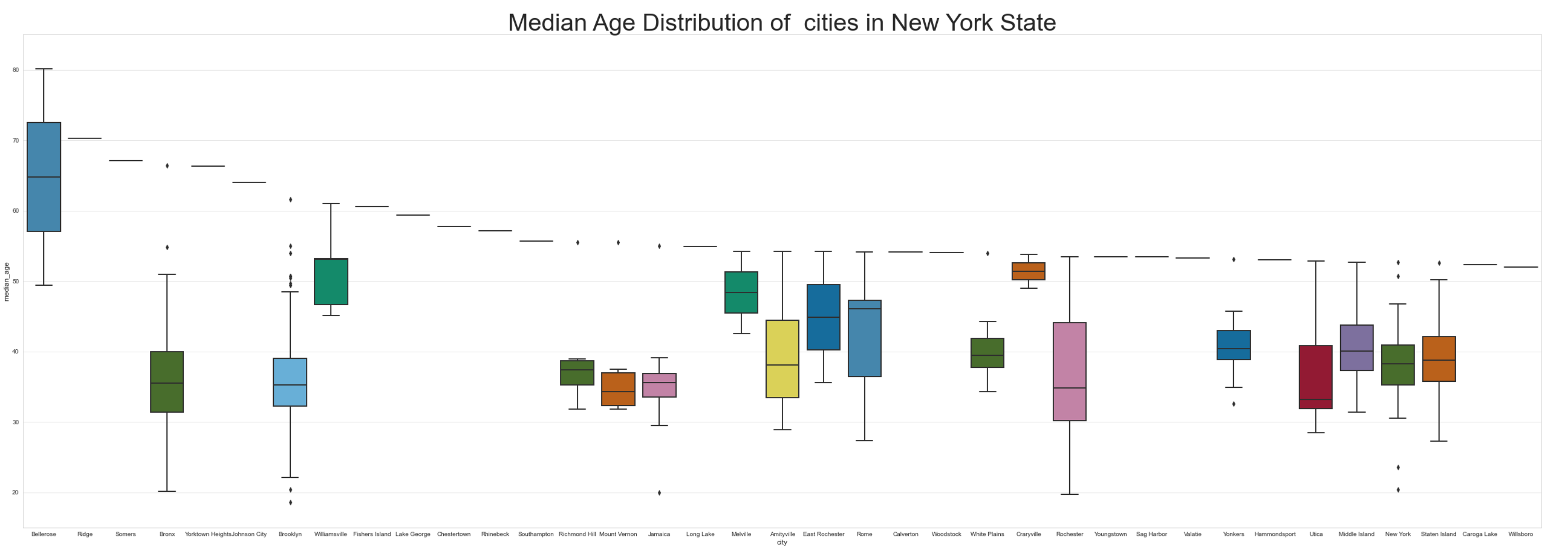
In [86]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
ax = sns.boxplot(x = 'state', y = 'median_age', data=density_eda_df.nsmallest(26585,
'median_age'), palette=color_pal,
              order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                       'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware']
             ).set_title('Median Age Distribution of  States with Younger Population',
fontsize = 40)
#ax.set(ylim=(0, 100))
plt.show()
```



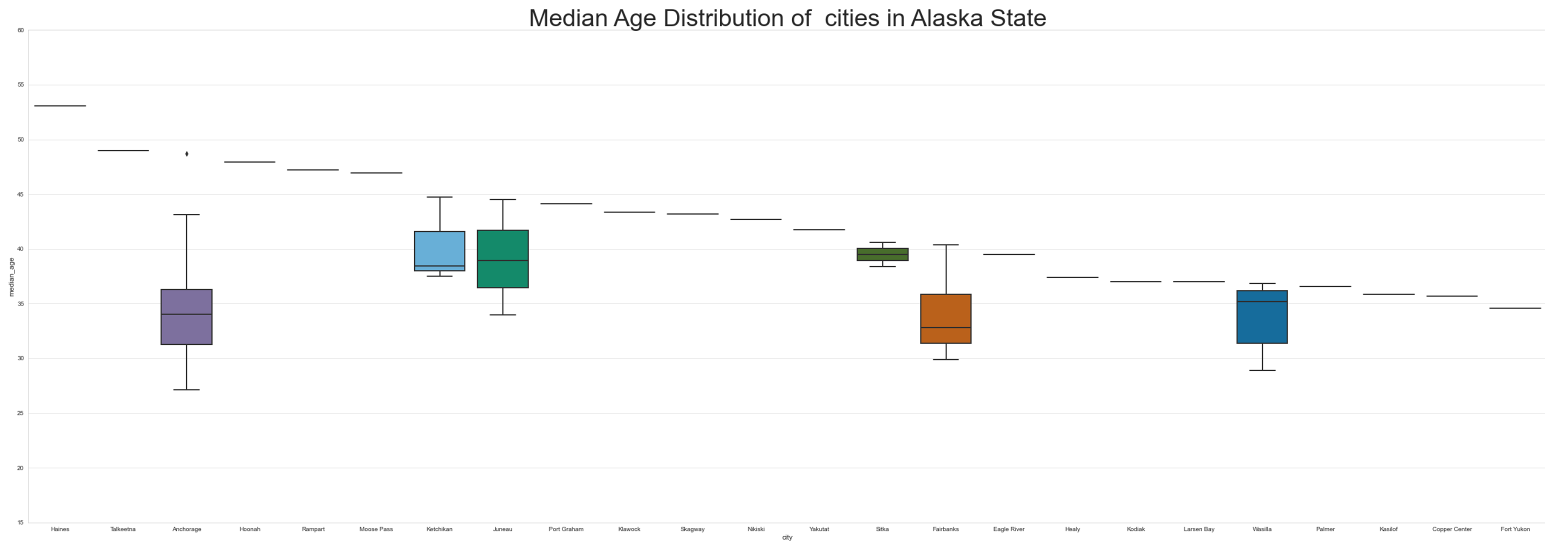
In [87]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
ax = sns.boxplot(x = 'city', y = 'median_age', data=density_eda_df[density_eda_df['state']
== 'New York'].nlargest(26585, 'median_age'), palette=color_pal,
            order =['Bellerose', 'Ridge', 'Somers', 'Bronx', 'Yorktown Heights', 'Johnson
City', 'Brooklyn', 'Williamsville', 'Fishers Island', 'Lake George', 'Chestertown',
'Rhinebeck',
                    'Southampton', 'Richmond Hill', 'Mount Vernon', 'Jamaica', 'Long
Lake', 'Melville', 'Amityville', 'East Rochester', 'Rome', 'Calverton', 'Woodstock',
'White Plains', 'Craryville',
                    'Rochester', 'Youngstown', 'Sag Harbor', 'Valatie', 'Yonkers',
'Hammondsport', 'Utica', 'Middle Island', 'New York', 'Staten Island', 'Caroga Lake',
'Willsboro']
)
ax.set_title('Median Age Distribution of  cities in New York State', fontsize = 40)
ax.set(ylim=(15, 85))
plt.show()
```



In [88]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
ax = sns.boxplot(x = 'city', y = 'median_age', data=density_eda_df[density_eda_df['state']
== 'Alaska'].nlargest(26585, 'median_age'), palette=color_pal,
            order =['Haines', 'Talkeetna', 'Anchorage', 'Hoonah', 'Rampart', 'Moose Pass',
'Ketchikan', 'Juneau', 'Port Graham', 'Klawock', 'Skagway', 'Nikiski', 'Yakutat', 'Sitka',
'Fairbanks',
                    'Eagle River', 'Healy', 'Kodiak', 'Larsen Bay', 'Wasilla', 'Palmer',
'Kasilof', 'Copper Center', 'Fort Yukon']
)
ax.set_title('Median Age Distribution of  cities in Alaska State', fontsize = 40)
ax.set(ylim=(15, 60))
plt.show()
```



In [89]:

```
list(density_eda_df[density_eda_df['state'] == 'New York'].nlargest(600,
'pop_density').place.unique())
print(len(list(density_eda_df[density_eda_df['state'] == 'New York'].nlargest(600,
'pop_density').place.unique())))
```

```
19
```

In [90]:

```
print(list(density_eda_df[density_eda_df['state'] == 'Alaska'].nlargest(42,
'median_age').city.unique()))
print(len(list(density_eda_df[density_eda_df['state'] == 'Alaska'].nlargest(42,
'median_age').city.unique())))
```

```
['Haines', 'Talkeetna', 'Anchorage', 'Hoonah', 'Rampart', 'Moose Pass', 'Ketchikan', 'Juneau
', 'Port Graham', 'Klawock', 'Skagway', 'Nikiski', 'Yakutat', 'Sitka', 'Fairbanks', 'Eagle R
iver', 'Healy', 'Kodiak', 'Larsen Bay', 'Wasilla', 'Palmer', 'Kasilof', 'Copper Center', 'Fo
rt Yukon']
24
```

In [91]:

```
train_df.head()
```

Out[91]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.!
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.:
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	-86.!
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	-66.
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	-96.!

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.

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

b) Visualize using appropriate chart type.

In [92]:

```
age_df = train_df[['state', 'city', 'place', 'pop', 'male_pop', 'female_pop',
'male_age_median', 'female_age_median', 'married', 'separated', 'divorced']]
```

In [93]:

```
train_df.male_age_median.unique()
```

Out[93]:

```
array([44.     , 32.     , 40.83333, 48.91667, 22.41667, 41.41667,
       40.     , 53.08333, 30.66667, 47.33333, 34.33333, 46.91667,
       49.75   , 34.66667, 42.58333, 45.83333, 44.16667, 32.5    ,
       30.41667, 27.41667, 30.08333, 41.16667, 38.75   , 30.     ,
       31.16667, 46.75   , 36.66667, 38.16667, 34.91667, 40.16667,
       27.66667, 39.33333, 42.83333, 36.41667, 41.91667, 44.5   ,
       51.75   , 43.41667, 51.66667, 34.     , 64.08333, 51.41667,
       20.25   , 29.     , 28.     , 41.25   , 49.83333, 24.91667,
       45.41667, 28.16667, 34.08333, 36.91667, 46.66667, 36.16667,
       36.75   , 38.5    , 36.08333, 47.5   , 51.16667, 48.16667,
       33.     , 25.25   , 37.08333, 42.66667, 40.25   , 29.75   ,
       38.41667, 37.41667, 42.     , 44.08333, 36.5    , 32.16667,
       35.91667, 39.5    , 37.75   , 38.58333, 21.25   , 35.33333,
       40.41667, 46.08333, 54.41667, 41.5    , 37.83333, 31.41667,
       41.75   , 32.41667, 26.66667, 39.83333, 31.91667, 34.58333,
       35.58333, 52.58333, 40.75   , 37.33333, 33.08333, 40.58333,
       36.25   , 42.16667, 32.91667, 45.58333, 46.16667, 25.     ,
       29.66667, 42.33333, 45.08333, 31.     , 52.66667, 37.66667,
       39.25   , 34.83333, 32.58333, 46.5    , 22.66667, 49.16667,
       35.5    , 32.66667, 37.25   , 22.     , 51.33333, 41.33333,
       43.25   , 35.83333, 37.     , 41.08333, 29.33333, 34.75   ,
       45.25   , 35.75   , 74.16667, 30.5    , 52.83333, 23.08333,
       30.75   , 36.33333, 33.41667, 46.     , 37.5    , 30.58333,
       36.     , 35.25   , 34.25   , 36.83333, 24.83333, 48.     ,
       33.83333, 44.83333, 42.5    , 43.66667, 43.33333, 33.75   ,
       46.25   , 57.83333, 31.83333, 49.08333, 43.5    , 30.33333,
       47.08333, 52.5    , 27.     , 29.16667, 44.41667, 38.     ,
       36.58333, 21.83333, 42.75   , 29.5    , 29.58333, 34.16667,
       35.08333, 33.16667, 39.58333, 23.5    , 33.66667, 43.16667,
       31.58333, 40.91667, 27.75   , 48.41667, 26.75   , 45.16667,
       44.33333, 41.66667, 38.33333, 33.5    , 25.83333, 39.66667,
       55.58333, 25.08333, 53.33333, 45.75   , 48.66667, 26.83333,
       45.91667, 28.66667, 35.41667, 26.08333, 26.16667, 31.66667,
       43.     , 31.33333, 35.     , 29.41667, 44.66667, 42.41667,
       23.83333, 38.91667, 40.5    , 44.58333, 28.75   , 26.58333,
       30.16667, 33.91667, 43.83333, 39.08333, 32.08333, 39.16667,
       24.     , 32.25   , 28.08333, 29.08333, 51.58333, 48.75   ,
       20.91667, 24.5    , 42.91667, 43.75   , 50.33333, 67.16667,
       48.83333, 59.08333, 22.5    , 29.91667, 46.83333, 50.66667,
       32.33333, 39.     , 44.91667, 56.     , 38.25   , 46.41667,
       48.58333, 52.25   , 41.58333, 45.66667, 25.91667, 33.33333,
       30.83333, 34.41667, 19.75   , 39.91667, 33.58333, 49.58333,
       47.16667, 32.75   , 70.5    , 40.33333, 54.08333, 35.66667,
       52.     , 31.5    , 25.16667, 22.16667, 45.5    , 48.5    ,
       23.58333, 43.91667, 47.58333, 24.08333, 21.08333, 43.58333,
```

```
       37.91667, 37.16667, 33.25   , 47.25   , 37.58333, 47.75   ,
       56.91667, 50.5    , 24.33333, 38.08333, 28.91667, 43.08333,
       44.25   , 27.83333, 40.08333, 42.08333, 29.25   , 27.08333,
       56.25   , 80.16667, 38.83333, 47.66667, 41.83333, 57.     ,
       40.66667, 47.91667, 30.25   , 27.25   , 39.75   , 27.91667,
       49.25   , 46.58333, 45.     , 50.41667, 54.16667, 71.41667,
       31.25   , 50.75   , 18.91667, 50.58333, 35.16667, 38.66667,
       44.75   , 28.25   , 20.33333, 49.66667, 16.16667, 41.     ,
       65.     , 23.41667, 28.58333, 32.83333, 34.5    , 22.33333,
       51.     , 25.33333, 52.08333, 26.33333, 24.25   , 39.41667,
       53.41667, 51.91667, 26.25   , 28.41667, 50.91667, 21.91667,
       62.33333, 48.25   , 23.     , 62.08333, 28.33333, 31.75   ,
       54.83333, 53.     , 49.     , 15.58333, 56.66667, 24.75   ,
       24.58333, 46.33333, 30.91667, 23.91667, 23.25   , 49.91667,
       51.5    , 27.5    , 55.75   , 17.83333, 67.75   , 27.33333,
       51.25   , 42.25   , 22.83333, 21.16667, 47.     , 29.83333,
       24.66667, 23.66667, 21.5    , 19.41667, 63.83333, 52.41667,
       24.41667, 26.5    , 25.41667, 21.75   , 45.33333, 53.66667,
       52.91667, 67.08333, 47.83333, 58.83333, 50.16667, 15.25   ,
       49.41667, 28.5    , 53.83333, 21.33333, 63.58333, 50.83333,
       51.83333, 64.58333, 27.16667, 22.25   , 71.33333, 62.5    ,
       58.5    , 18.41667, 52.75   , 50.25   , 58.16667, 22.91667,
       31.08333, 28.83333, 51.08333, 20.     , 57.91667, 27.58333,
       26.41667, 59.25   , 47.41667, 60.75   , 25.58333, 20.58333,
       19.91667, 26.     , 19.33333, 56.75   , 54.25   , 24.16667,
       65.16667, 19.58333, 60.66667, 21.66667, 63.33333, 25.5    ,
       53.5    , 22.58333, 49.33333, 55.25   , 22.75   , 21.41667,
       52.33333, 17.66667, 62.91667, 49.5    , 50.08333, 71.91667,
       57.66667, 60.08333, 75.58333, 59.66667, 53.75   , 61.41667,
       60.33333, 23.75   , 53.25   , 50.     , 56.83333, 69.83333,
       68.83333, 48.08333, 54.58333, 19.25   , 58.     , 57.25   ,
       25.66667, 60.5    , 57.08333, 54.     , 55.16667, 20.83333,
       48.33333, 73.     , 26.91667, 57.75   , 54.33333, 62.75   ,
       64.66667, 64.75   , 60.25   , 56.16667, 55.     , 61.33333,
       20.5    , 52.16667, 19.5    , 64.41667, 59.91667, 15.16667,
       23.16667, 55.41667, 57.58333, 61.25   , 54.75   , 23.33333,
       59.58333, 70.91667, 58.66667, 70.08333, 17.91667, 55.83333,
       60.41667, 60.58333, 73.41667, 20.08333, 56.41667, 67.83333,
       53.58333, 55.08333, 70.33333, 55.66667, 64.83333, 15.08333,
       62.58333, 71.58333, 25.75   , 71.83333, 21.58333, 17.     ,
       64.33333, 56.33333, 54.66667, 69.08333, 65.58333, 66.33333,
       66.     , 61.75   , 17.58333, 54.5    , 67.41667, 71.25   ,
       63.25   , 66.75   , 21.     , 76.83333, 59.16667, 65.41667,
       57.5    , 20.66667, 18.33333, 66.25   , 53.16667, 74.66667,
       66.08333, 73.91667, 22.08333, 77.08333, 20.41667, 58.08333,
       59.33333, 73.16667, 77.83333, 65.33333, 13.58333, 65.5    ,
       15.91667, 56.58333, 76.33333, 19.66667, 63.5    , 72.75   ,
       59.83333, 63.91667, 56.08333, 74.41667, 65.66667, 55.33333,
       55.91667, 20.75   , 61.83333, 59.     , 57.33333, 60.16667,
       68.25   , 16.83333, 19.83333, 10.     , 20.16667, 64.     ,
       70.25   , 68.66667, 59.41667, 66.5    , 62.     , 63.08333,
       59.5    , 67.58333, 62.83333, 53.91667, 62.25   , 78.     ,
       58.75   , 62.16667, 73.66667, 18.75   , 66.41667, 72.33333,
       75.     , 67.     , 62.66667, 54.91667, 18.08333, 61.16667,
       72.66667, 62.41667, 18.16667, 77.75   , 60.91667, 19.16667,
       68.91667, 19.08333, 65.75   , 77.25   , 75.16667, 74.83333,
       18.66667, 56.5    , 58.91667, 69.25   , 17.41667, 13.5    ,
       67.66667, 68.41667, 57.16667, 58.33333, 58.41667, 72.58333,
       78.25   , 67.5    , 19.     , 14.75   , 61.08333, 55.5    ,
       71.08333, 64.91667, 58.58333, 70.     , 68.33333, 16.75   ,
       61.     , 14.91667, 72.91667, 17.33333, 69.91667, 59.75   ,
       69.58333, 69.16667, 66.16667, 63.     , 13.16667, 67.33333,
       71.     , 65.08333, 63.75   , 69.33333, 68.     , 75.66667,
       77.66667, 65.91667, 74.5    , 60.83333, 73.33333, 60.     ,
       63.41667, 73.25   , 64.25   , 75.5    , 76.91667, 71.75   ,
       70.75   , 73.58333, 16.     ,  9.75   , 74.     , 63.66667,
       16.33333, 68.16667, 65.25   , 18.25   , 71.66667, 61.58333,
       17.5    , 17.25   ])
```

The IntelliSense Age Group defaults are:

Youth (<18)

Young Adult (18 to 35)

Adult (36 to 55)

Senior (56 and up)

In [94]:

```
bins = [0, 12,18, 35, 55, 100]
labels = ['kids', 'Youth', 'Young Adult', 'Adult', 'Senior']
#df['binned'] = pd.cut(df['percentage'], bins, labels = labels)
```

In [95]:

```
age_df['male_population_bracket'] = pd.cut(age_df['male_age_median'], bins, labels =
labels)
```

In [96]:

```
age_df['female_population_bracket'] = pd.cut(age_df['female_age_median'], bins, labels =
labels)
```

In [97]:

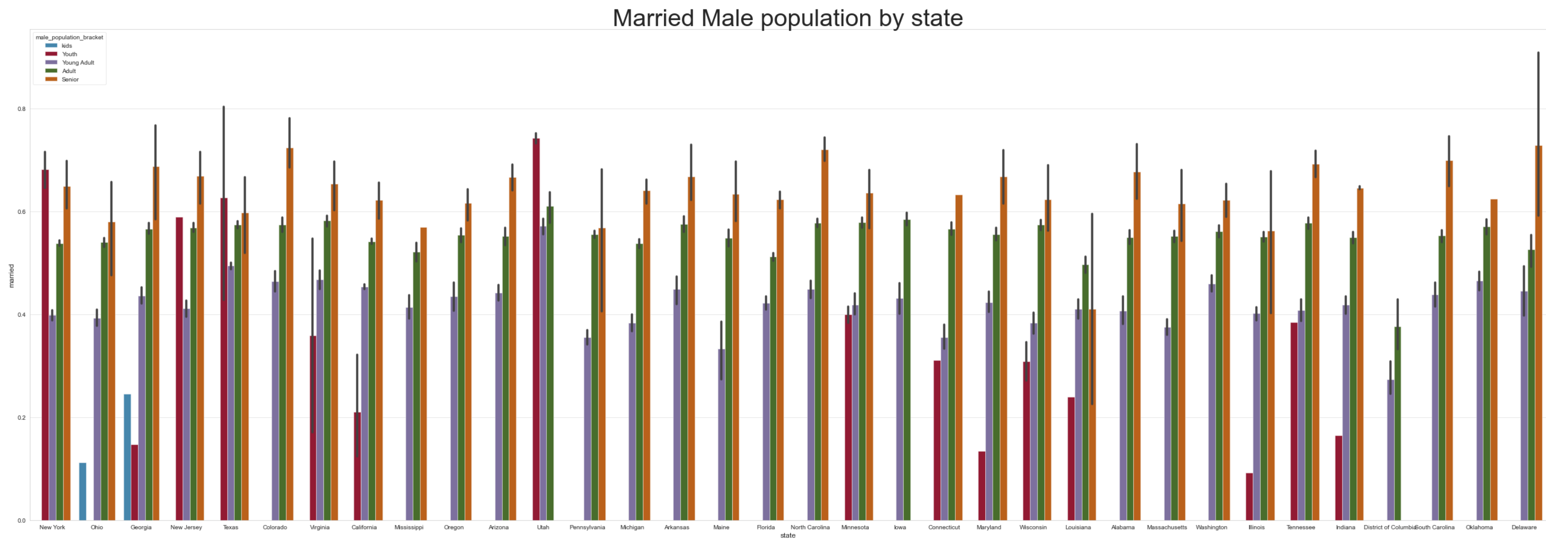
```
age_df.head()
```

Out[97]:

	state	city	place	pop	male_pop	female_pop	male_age_median	female_age_median	married	separated	divorc
0	New York	Hamilton	Hamilton	5230	2612	2618	44.00000	45.33333	0.57851	0.01240	0.087
1	Indiana	South Bend	Roseland	2633	1349	1284	32.00000	37.58333	0.34886	0.01426	0.090
2	Indiana	Danville	Danville	6881	3643	3238	40.83333	42.83333	0.64745	0.01607	0.106
3	Puerto Rico	San Juan	Guaynabo	2700	1141	1559	48.91667	50.58333	0.47257	0.02021	0.101
4	Kansas	Manhattan	Manhattan City	5637	2586	3051	22.41667	21.58333	0.12356	0.00000	0.031

In [98]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'married', hue = 'male_population_bracket', data =
age_df, palette=color_pal,
            order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                      'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Married Male population by state', fontsize = 40)

plt.show()
```



Surprisingly, "Ohio & Georgia" have Married Male KIDS

In [99]:

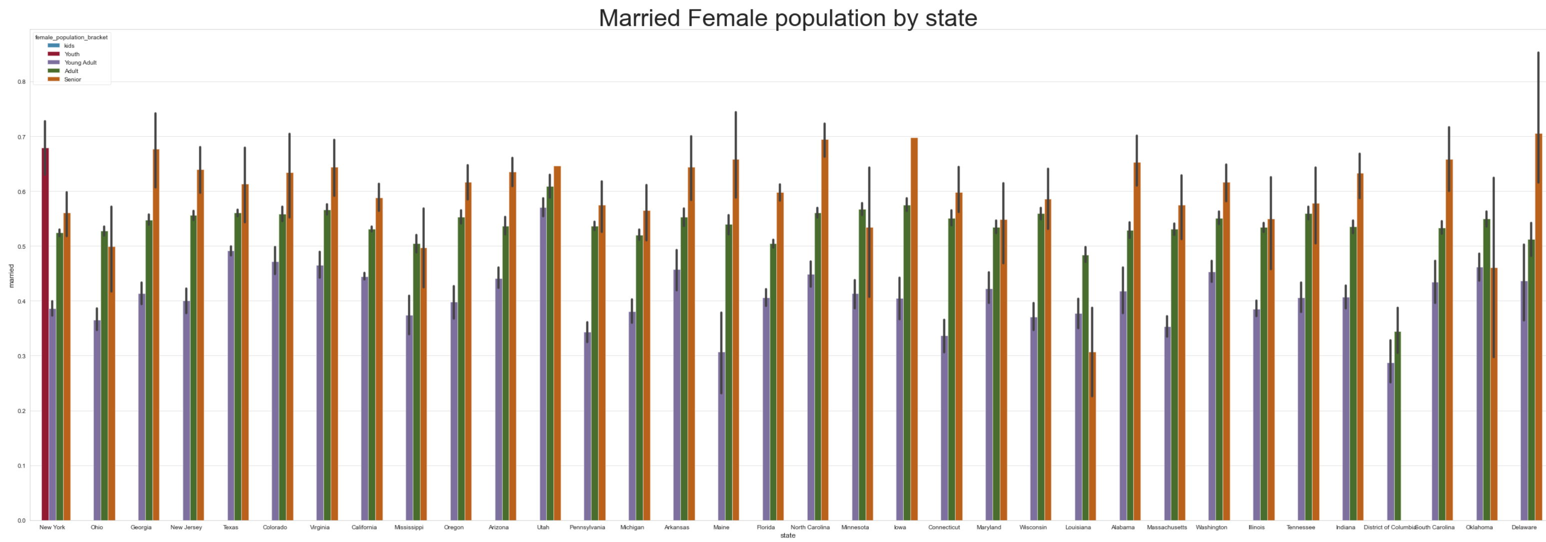
```
age_df.city.unique()
```

Out[99]:

```
['Hamilton', 'South Bend', 'Danville', 'San Juan', 'Manhattan', ..., 'Cresco', 'Wittensville
', 'Blue Bell', 'Weldona', 'Colleyville']
Length: 6876
Categories (6876, object): ['Hamilton', 'South Bend', 'Danville', 'San Juan', ..., 'Wittensv
ille', 'Blue Bell', 'Weldona', 'Colleyville']
```

In [100]:

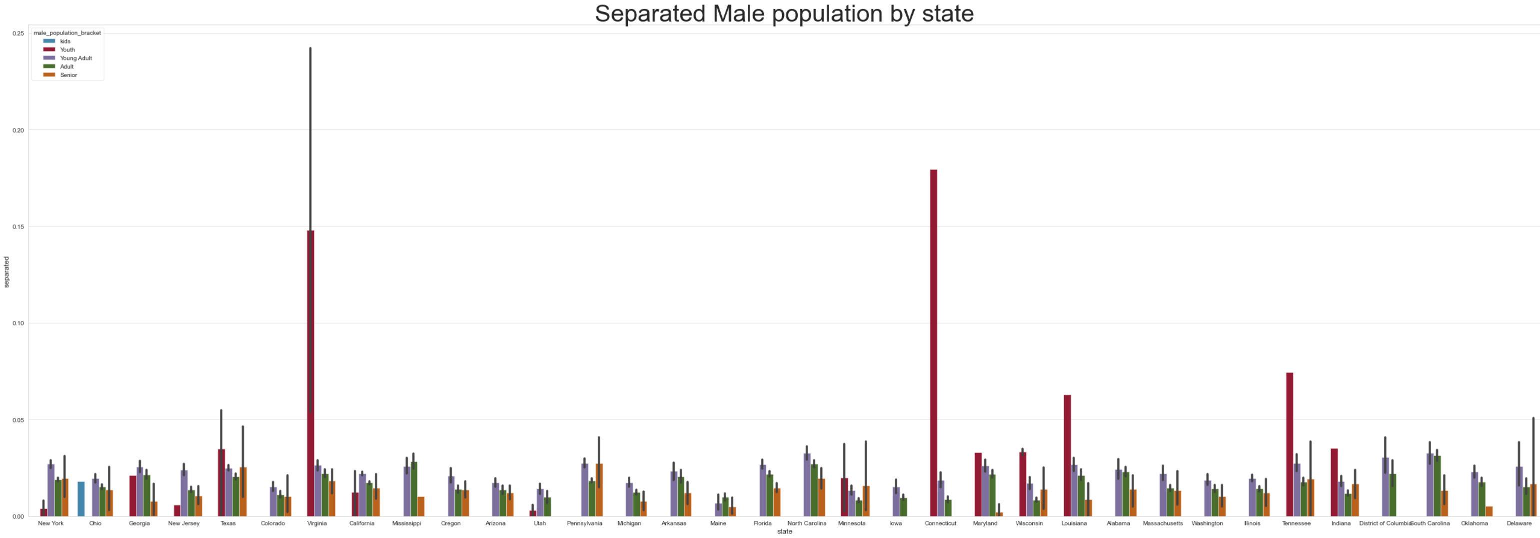
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'married', hue = 'female_population_bracket', data =
age_df, palette=color_pal,
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                     'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Married Female population by state', fontsize = 40)

plt.show()
```



Except for "Newyork", NO other state has Married Female KIDS or Youth

In [101]:

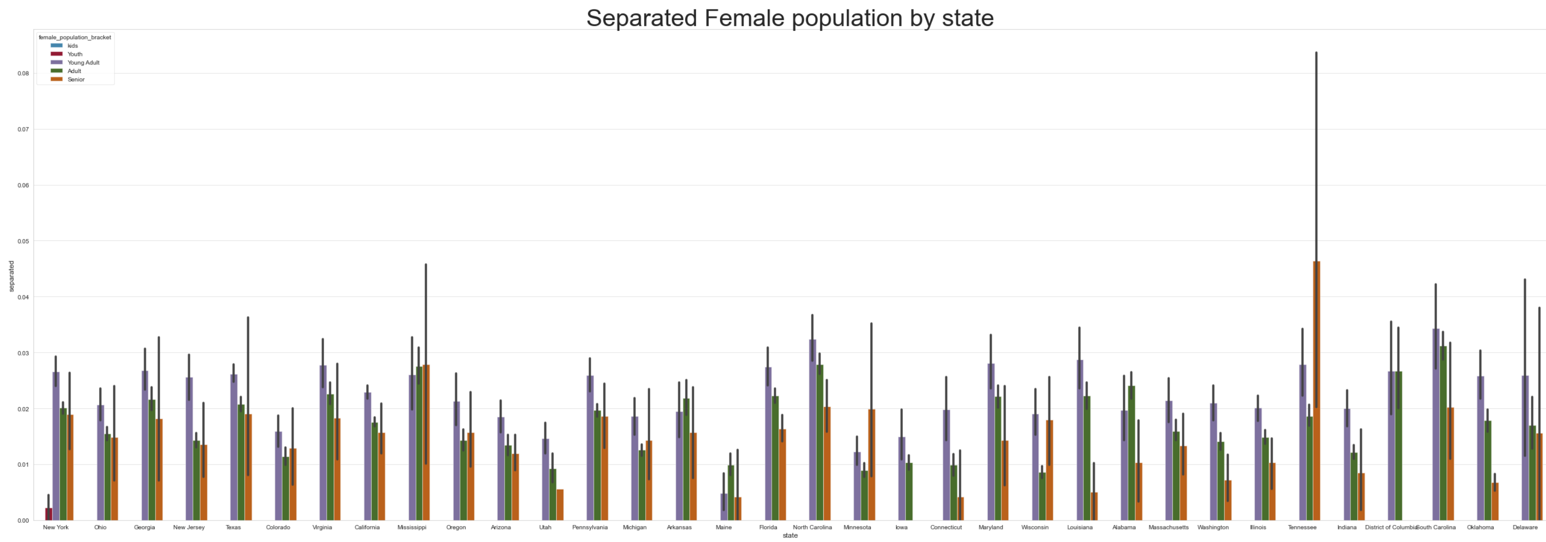
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'separated', hue = 'male_population_bracket', data =
age_df, palette=color_pal,
            order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                        'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Separated Male population by state', fontsize = 40)

plt.show()
```



"Connecticut, followed by Virginia", has Highest Separated Male Youth population

In [102]:

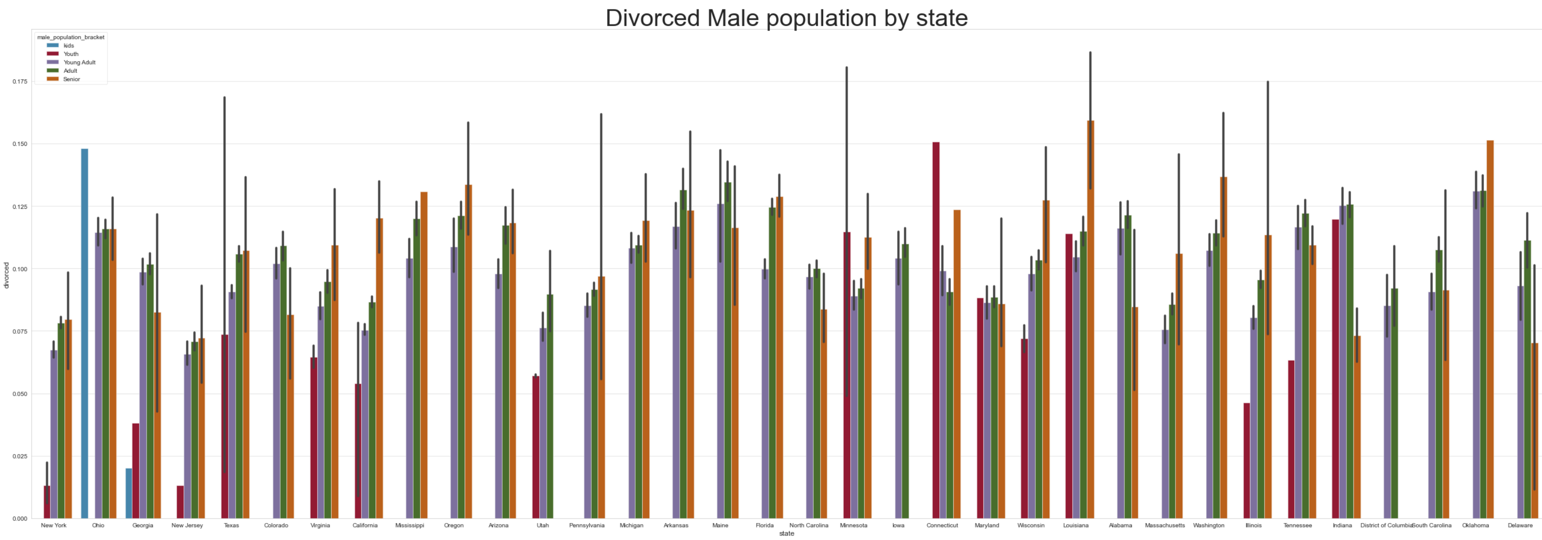
```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'separated', hue = 'female_population_bracket', data =
age_df, palette=color_pal,
            order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                      'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Separated Female population by state', fontsize = 40)

plt.show()
```



Except for "Newyork", No other state has Separated Female Youth population "Tennessee" has the Highest Separated Female SENIOR population

In [103]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'divorced', hue = 'male_population_bracket', data =
age_df, palette=color_pal,
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                     'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Divorced Male population by state', fontsize = 40)

plt.show()
```



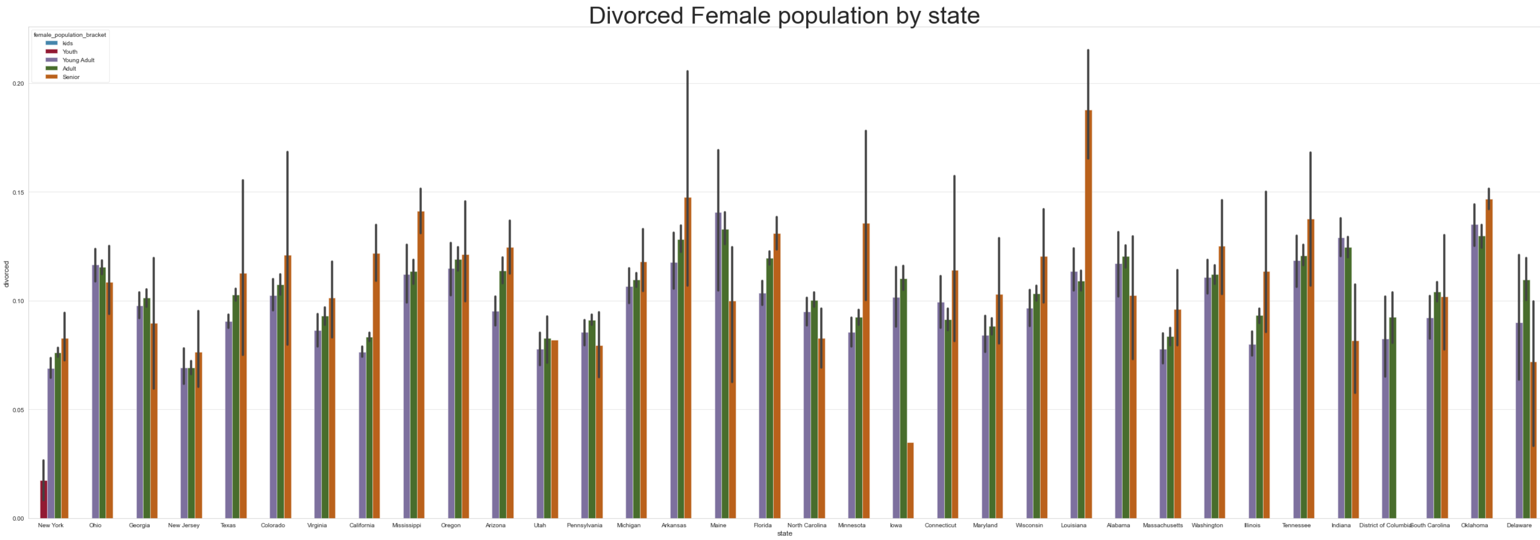
"Ohio", has Largest number of Divorced Male KIDS.
"Connecticut", has Largest number of Divorced Male YOUTH.
"Maine, Indiana & Oklahoma", has Largest number of Divorced Male YOUNG ADULTS
"Arkansas, Maine, Indiana & Oklahoma", has Largest number of Divorced Male ADULTS
"Louisiana & OKlahoma", has Largest number of Divorced Male SENIORS.

Looks like "OKlahoma", is the Divorce Capital for MALE population.

In [104]:

```
sns.set_style("whitegrid")

plt.figure(figsize = (45, 15))

ax = sns.barplot(x = 'state', y = 'divorced', hue = 'female_population_bracket', data =
age_df, palette=color_pal,
             order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado',
'Virginia', 'California', 'Mississippi', 'Oregon', 'Arizona', 'Utah', 'Pennsylvania',
```

```
'Michigan', 'Arkansas', 'Maine', 'Florida', 'North Carolina',
                       'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin',
'Louisiana', 'Alabama', 'Massachusetts', 'Washington', 'Illinois', 'Tennessee', 'Indiana',
'District of Columbia', 'South Carolina', 'Oklahoma', 'Delaware'])

ax.set_title('Divorced Female population by state', fontsize = 40)

plt.show()
```



"Newyork", is the only state that has Divorced Female YOUTH.
"Maine", has Largest number of Divorced Female YOUNG ADULTS
"Maine", has Largest number of Divorced Female ADULTS
"Louisiana", has Largest number of Divorced Female SENIORS.

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

In [105]:

```
train_df.head()
```

Out[105]:

UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat

0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.!
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.:
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	-86.!
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	-66.
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	-96.!

In [106]:

```
rent_df = train_df[['state', 'city', 'rent_median', 'hi_median', 'family_median']]
```

In [107]:

```
Overall_rent_percentage = (rent_df['rent_median'].sum() / rent_df['hi_median'].sum()) *
100
round(Overall_rent_percentage, 2)
```

Out[107]:

```
1.74
```

Overall Rent as a percentage of Overall House Hold Income is around 1.74%.

In [108]:

```
rent_df['ov_rent_pcnt'] = round((rent_df['rent_median'] / rent_df['hi_median']) * 100, 2)
```

In [109]:

```
rent_df.head()
```

Out[109]:

	state	city	rent_median	hi_median	family_median	ov_rent_pcnt
0	New York	Hamilton	784.0	48120.0	53245.0	1.63
1	Indiana	South Bend	848.0	35186.0	43023.0	2.41
2	Indiana	Danville	703.0	74964.0	85395.0	0.94
3	Puerto Rico	San Juan	782.0	37845.0	44399.0	2.07
4	Kansas	Manhattan	881.0	22497.0	50272.0	3.92

In [110]:

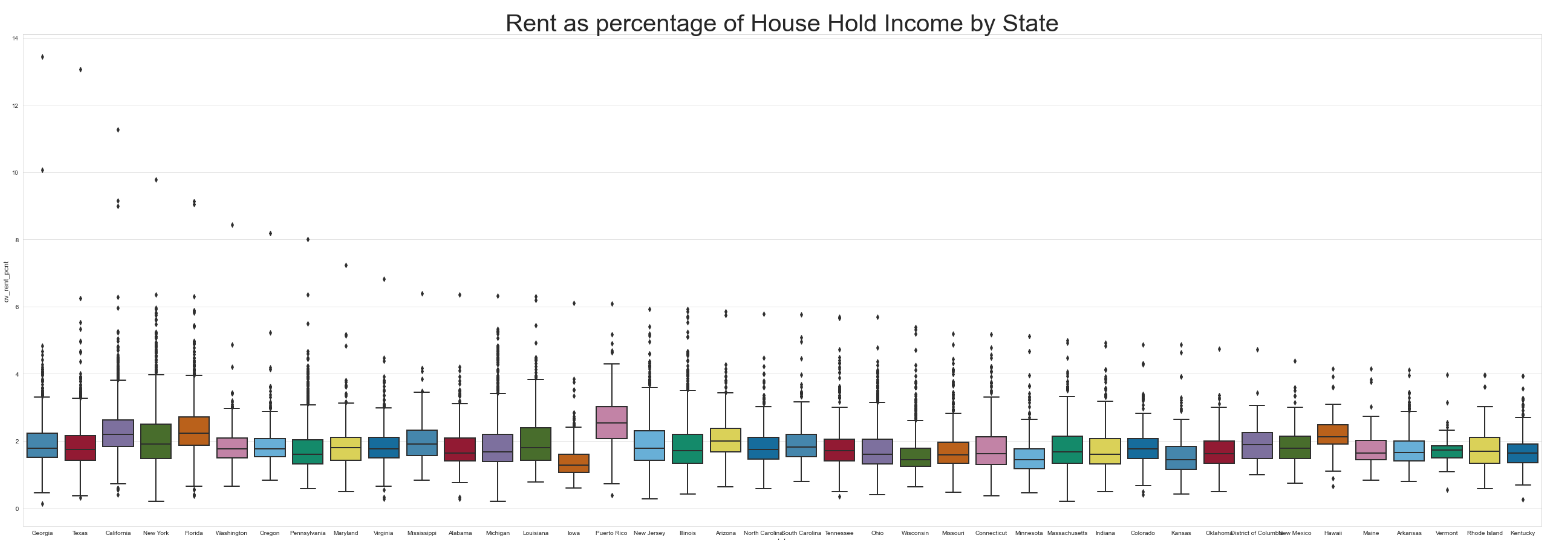
```
print(list(rent_df.nlargest(500, 'ov_rent_pcnt').state.unique()))
print(len(list(rent_df.nlargest(500, 'ov_rent_pcnt').state.unique())))
```

```
['Georgia', 'Texas', 'California', 'New York', 'Florida', 'Washington', 'Oregon', 'Pennsylva
```

```
nia', 'Maryland', 'Virginia', 'Mississippi', 'Alabama', 'Michigan', 'Louisiana', 'Iowa', 'Pu
erto Rico', 'New Jersey', 'Illinois', 'Arizona', 'North Carolina', 'South Carolina', 'Tennes
see', 'Ohio', 'Wisconsin', 'Missouri', 'Connecticut', 'Minnesota', 'Massachusetts', 'Indiana
', 'Colorado', 'Kansas', 'Oklahoma', 'District of Columbia', 'New Mexico', 'Hawaii', 'Maine'
, 'Arkansas', 'Vermont', 'Rhode Island', 'Kentucky']
40
```

In [111]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (45, 15))
ax = sns.boxplot(x = 'state', y = 'ov_rent_pcnt', data=rent_df.nlargest(26585,
'ov_rent_pcnt'), palette=color_pal,
            order = ['Georgia', 'Texas', 'California', 'New York', 'Florida',
'Washington', 'Oregon', 'Pennsylvania', 'Maryland', 'Virginia', 'Mississippi', 'Alabama',
'Michigan', 'Louisiana',
                     'Iowa', 'Puerto Rico', 'New Jersey', 'Illinois', 'Arizona', 'North
Carolina', 'South Carolina', 'Tennessee', 'Ohio', 'Wisconsin', 'Missouri', 'Connecticut',
'Minnesota',
                     'Massachusetts', 'Indiana', 'Colorado', 'Kansas', 'Oklahoma',
'District of Columbia', 'New Mexico', 'Hawaii', 'Maine', 'Arkansas', 'Vermont', 'Rhode
Island', 'Kentucky']
           ).set_title('Rent as percentage of House Hold Income by State', fontsize = 40)
#ax.set(ylim=(0, 100))
plt.show()
```

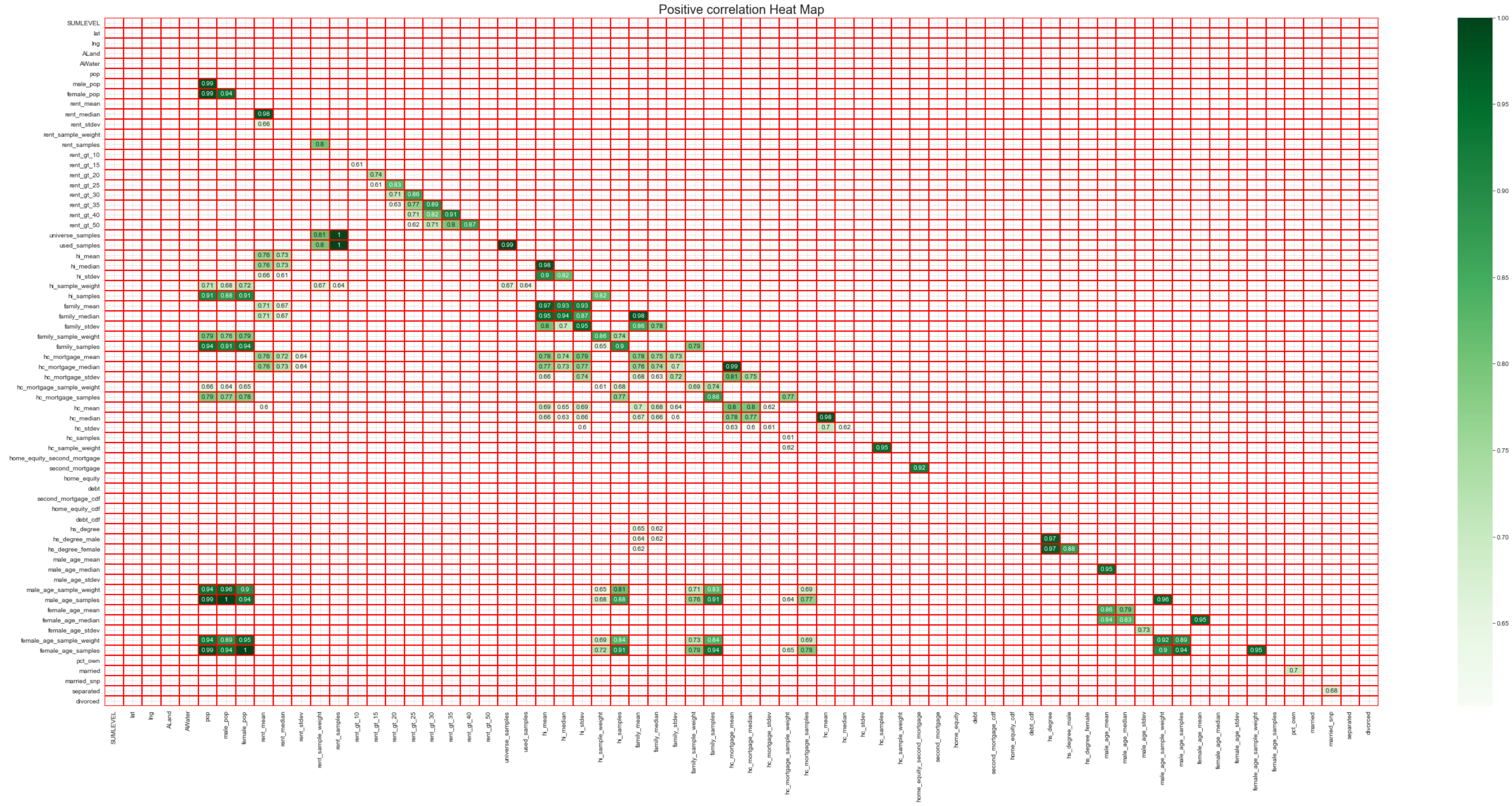


4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

In [112]:

```
sns.set_style("whitegrid")

corr = train_df.corr()

mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

kot = corr[corr>=.6]
plt.figure(figsize=(45,20))
sns.heatmap(kot, cmap="Greens", annot = True, mask = mask, linewidths=1,
linecolor='red').set_title('Positive correlation Heat Map', fontsize = 20)
```

```
plt.grid('on', )
plt.show()
```



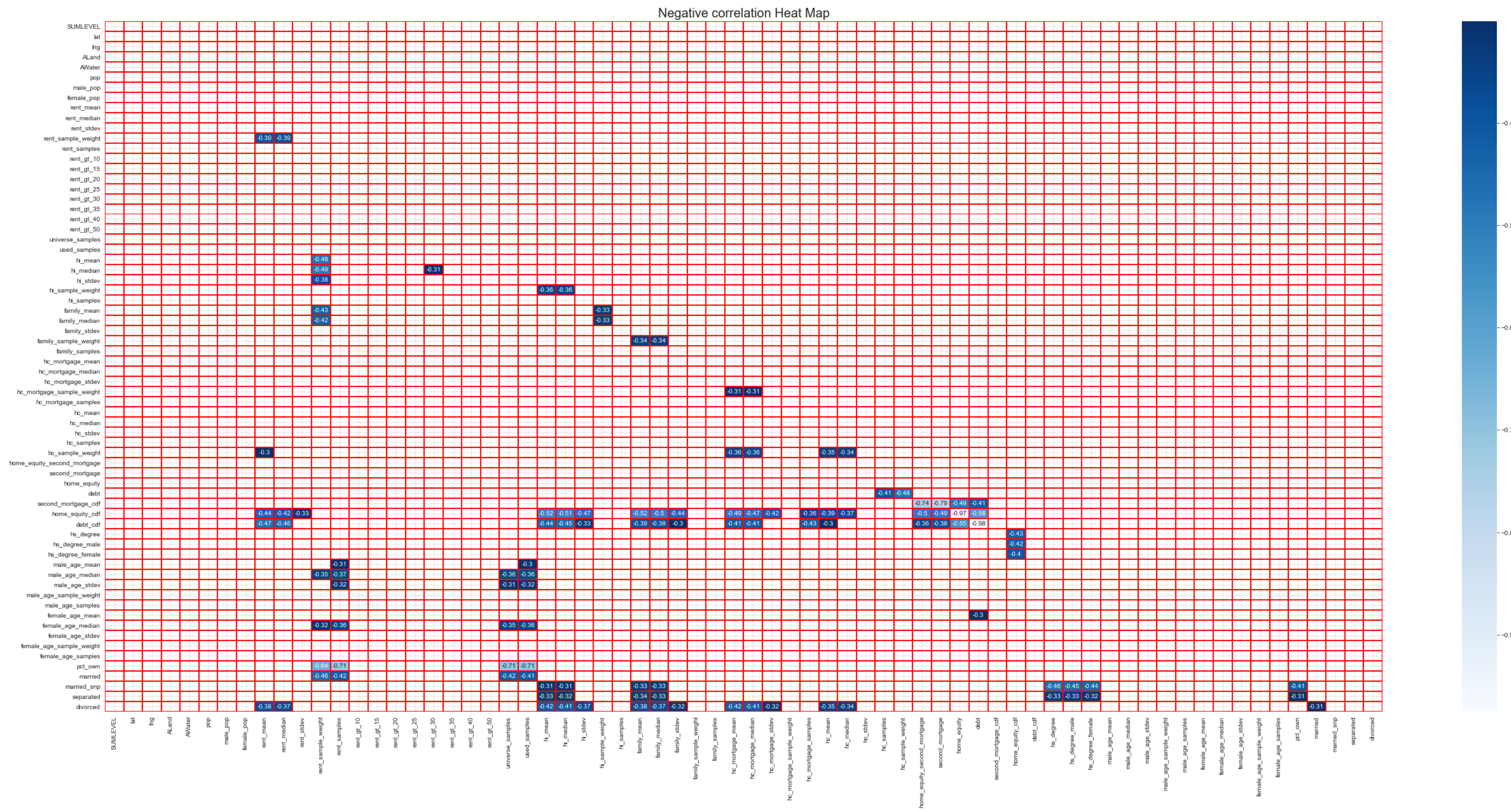
- "Population parameters" have Strong positive correlation wih "Sample Parameters".

```
- "Male Population is highly correlated with Female population. <br/>

<br/>

- "rent Mean & Median" has  high positive correlation with "House hold income Mean, Median and Standard Deviation",  <br/>

<br/>

where as "rent Standard Deviation has positive correlatioin with "hc mortgage mean & median". <br/>

<br/>

- "House hold income and Family income are highly positively correlated. <br/>

<br/>

- "Family Income"  and "hc_mortgage" are positively correlated. <br/>
```


- "pct_own" is positively correlated with "Married" marital status </h1>

In [113]:

```
sns.set_style("whitegrid")
kot = corr[corr <=-.3]
plt.figure(figsize=(45,20))
sns.heatmap(kot, cmap="Blues", annot = True, mask = mask, linewidths=1,
linecolor='red').set_title('Negative correlation Heat Map', fontsize = 20)
plt.grid('on', )
plt.show()
```



- "House hold income and Family Income" has Strong negative correlation with ["married_snp", "separated", "divorced"].

- "High School Degree in both "Males and Females" have Strong negative correlation with ["married_snp", "separated"]

- "pct_own" has Strong negative correlation with ["married_snp", "separated"]

```
<br/>

- "hi_median" has Strong negative correlation with
"rent_gt_30", indicating that most households look for
properties with rent less than 30% of their house hold
income.. <br/>
</h1>
```

Data Pre-processing:

Project Task: Week 3

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.

1. **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
</h1>

In [114]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26585 entries, 0 to 27320
Data columns (total 78 columns):
 #   Column                       Non-Null Count  Dtype   
---  ------                       --------------  -----   
 0   UID                          26585 non-null  category
 1   SUMLEVEL                     26585 non-null  int64   
 2   COUNTYID                     26585 non-null  category
 3   STATEID                      26585 non-null  category
 4   state                        26585 non-null  category
 5   state_ab                     26585 non-null  category
 6   city                         26585 non-null  category
 7   place                        26585 non-null  category
 8   type                         26585 non-null  category
 9   zip_code                     26585 non-null  category
 10  area_code                    26585 non-null  category
 11  lat                          26585 non-null  float64 
 12  lng                          26585 non-null  float64 
 13  ALand                        26585 non-null  float64 
 14  AWater                       26585 non-null  int64   
 15  pop                          26585 non-null  int64   
 16  male_pop                     26585 non-null  int64   
 17  female_pop                   26585 non-null  int64   
 18  rent_mean                    26585 non-null  float64 
 19  rent_median                  26585 non-null  float64 
 20  rent_stdev                   26585 non-null  float64 
 21  rent_sample_weight           26585 non-null  float64 
 22  rent_samples                 26585 non-null  float64 
 23  rent_gt_10                   26585 non-null  float64 
 24  rent_gt_15                   26585 non-null  float64 
 25  rent_gt_20                   26585 non-null  float64 
 26  rent_gt_25                   26585 non-null  float64 
 27  rent_gt_30                   26585 non-null  float64 
 28  rent_gt_35                   26585 non-null  float64 
 29  rent_gt_40                   26585 non-null  float64 
 30  rent_gt_50                   26585 non-null  float64 
 31  universe_samples             26585 non-null  int64   
 32  used_samples                 26585 non-null  int64   
 33  hi_mean                      26585 non-null  float64 
 34  hi_median                    26585 non-null  float64 
 35  hi_stdev                     26585 non-null  float64 
 36  hi_sample_weight             26585 non-null  float64 
 37  hi_samples                   26585 non-null  float64 
 38  family_mean                  26585 non-null  float64 
 39  family_median                26585 non-null  float64 
 40  family_stdev                 26585 non-null  float64 
 41  family_sample_weight         26585 non-null  float64 
 42  family_samples               26585 non-null  float64 
 43  hc_mortgage_mean             26585 non-null  float64 
 44  hc_mortgage_median           26585 non-null  float64 
 45  hc_mortgage_stdev            26585 non-null  float64 
 46  hc_mortgage_sample_weight    26585 non-null  float64 
 47  hc_mortgage_samples          26585 non-null  float64 
 48  hc_mean                      26585 non-null  float64 
 49  hc_median                    26585 non-null  float64 
 50  hc_stdev                     26585 non-null  float64 
 51  hc_samples                   26585 non-null  float64 
```

```
 52  hc_sample_weight              26585 non-null  float64
 53  home_equity_second_mortgage   26585 non-null  float64
 54  second_mortgage               26585 non-null  float64
 55  home_equity                   26585 non-null  float64
 56  debt                          26585 non-null  float64
 57  second_mortgage_cdf           26585 non-null  float64
 58  home_equity_cdf               26585 non-null  float64
 59  debt_cdf                      26585 non-null  float64
 60  hs_degree                     26585 non-null  float64
 61  hs_degree_male                26585 non-null  float64
 62  hs_degree_female              26585 non-null  float64
 63  male_age_mean                 26585 non-null  float64
 64  male_age_median               26585 non-null  float64
 65  male_age_stdev                26585 non-null  float64
 66  male_age_sample_weight        26585 non-null  float64
 67  male_age_samples              26585 non-null  float64
 68  female_age_mean               26585 non-null  float64
 69  female_age_median             26585 non-null  float64
 70  female_age_stdev              26585 non-null  float64
 71  female_age_sample_weight      26585 non-null  float64
 72  female_age_samples            26585 non-null  float64
 73  pct_own                       26585 non-null  float64
 74  married                       26585 non-null  float64
 75  married_snp                   26585 non-null  float64
 76  separated                     26585 non-null  float64
 77  divorced                      26585 non-null  float64
dtypes: category(10), float64(61), int64(7)
memory usage: 18.6 MB
```

In [115]:

```
train_df['Bad_Debt'] = train_df['second_mortgage'] + train_df['home_equity'] -
train_df['home_equity_second_mortgage']
```

In [116]:

```
for col in train_df.columns:
    print(col,' = ' ,train_df[col].dtype)
```

```
UID  =  category
SUMLEVEL  =  int64
COUNTYID  =  category
STATEID  =  category
state  =  category
state_ab  =  category
city  =  category
place  =  category
type  =  category
zip_code  =  category
area_code  =  category
lat  =  float64
lng  =  float64
ALand  =  float64
AWater  =  int64
pop  =  int64
male_pop  =  int64
female_pop  =  int64
rent_mean  =  float64
rent_median  =  float64
rent_stdev  =  float64
rent_sample_weight  =  float64
rent_samples  =  float64
rent_gt_10  =  float64
rent_gt_15  =  float64
rent_gt_20  =  float64
rent_gt_25  =  float64
rent_gt_30  =  float64
rent_gt_35  =  float64
rent_gt_40  =  float64
rent_gt_50  =  float64
universe_samples  =  int64
```

```
used_samples  =  int64
hi_mean  =  float64
hi_median  =  float64
hi_stdev  =  float64
hi_sample_weight  =  float64
hi_samples  =  float64
family_mean  =  float64
family_median  =  float64
family_stdev  =  float64
family_sample_weight  =  float64
family_samples  =  float64
hc_mortgage_mean  =  float64
hc_mortgage_median  =  float64
hc_mortgage_stdev  =  float64
hc_mortgage_sample_weight  =  float64
hc_mortgage_samples  =  float64
hc_mean  =  float64
hc_median  =  float64
hc_stdev  =  float64
hc_samples  =  float64
hc_sample_weight  =  float64
home_equity_second_mortgage  =  float64
second_mortgage  =  float64
home_equity  =  float64
debt  =  float64
second_mortgage_cdf  =  float64
home_equity_cdf  =  float64
debt_cdf  =  float64
hs_degree  =  float64
hs_degree_male  =  float64
hs_degree_female  =  float64
male_age_mean  =  float64
male_age_median  =  float64
male_age_stdev  =  float64
male_age_sample_weight  =  float64
male_age_samples  =  float64
female_age_mean  =  float64
female_age_median  =  float64
female_age_stdev  =  float64
female_age_sample_weight  =  float64
female_age_samples  =  float64
pct_own  =  float64
married  =  float64
married_snp  =  float64
separated  =  float64
divorced  =  float64
Bad Debt  =  float64
```

In [117]:

```
def cat_variables(df):
    cat_variables = list(df.select_dtypes(exclude = ['int', 'float']).columns)
    return cat_variables
```

In [118]:

```
def num_variables(df):
    num_variables = list(df.select_dtypes(include = ['int', 'float']).columns)
    return num_variables
```

In [119]:

```
train_df.city.dtype
```

Out[119]:

```
CategoricalDtype(categories=['Abbeville', 'Aberdeen', 'Abilene', 'Abingdon', 'Abington',
                  'Accokeek', 'Acton', 'Acushnet', 'Acworth', 'Ada',
                  ...
                  'Zeeland', 'Zellwood', 'Zephyr Cove', 'Zephyrhills',
                  'Zieglerville', 'Zionsville', 'Zoarville', 'Zolfo Springs',
                  'Zumbrota', 'Zuni'],
```

```
                 ordered=False)
```

In [120]:

```
cat_variables(train_df)
```

Out[120]:

```
['UID',
 'SUMLEVEL',
 'COUNTYID',
 'STATEID',
 'state',
 'state_ab',
 'city',
 'place',
 'type',
 'zip_code',
 'area_code',
 'AWater',
 'pop',
 'male_pop',
 'female_pop',
 'universe_samples',
 'used samples']
```

In [121]:

```
num_variables(train_df)
```

Out[121]:

```
['lat',
 'lng',
 'ALand',
 'rent_mean',
 'rent_median',
 'rent_stdev',
 'rent_sample_weight',
 'rent_samples',
 'rent_gt_10',
 'rent_gt_15',
 'rent_gt_20',
 'rent_gt_25',
 'rent_gt_30',
 'rent_gt_35',
 'rent_gt_40',
 'rent_gt_50',
 'hi_mean',
 'hi_median',
 'hi_stdev',
 'hi_sample_weight',
 'hi_samples',
 'family_mean',
 'family_median',
 'family_stdev',
 'family_sample_weight',
 'family_samples',
 'hc_mortgage_mean',
 'hc_mortgage_median',
 'hc_mortgage_stdev',
 'hc_mortgage_sample_weight',
 'hc_mortgage_samples',
 'hc_mean',
 'hc_median',
 'hc_stdev',
 'hc_samples',
 'hc_sample_weight',
 'home_equity_second_mortgage',
 'second_mortgage',
 'home_equity',
 'debt',
 'second_mortgage_cdf',
```

```
 'home_equity_cdf',
 'debt_cdf',
 'hs_degree',
 'hs_degree_male',
 'hs_degree_female',
 'male_age_mean',
 'male_age_median',
 'male_age_stdev',
 'male_age_sample_weight',
 'male_age_samples',
 'female_age_mean',
 'female_age_median',
 'female_age_stdev',
 'female_age_sample_weight',
 'female_age_samples',
 'pct_own',
 'married',
 'married_snp',
 'separated',
 'divorced',
 'Bad Debt']
```

In [122]:

```
fa_train_df = train_df[num_variables(train_df)]
fa_train_df
```

Out[122]:

	lat	lng	ALand	rent_mean	rent_median	rent_stdev	rent_sample_weight	rent_samples	rent_gt_10
0	42.840812	-75.501524	2.021834e+08	769.38638	784.0	232.63967	272.34441	362.0	0.86761
1	41.701441	-86.266614	1.560828e+06	804.87924	848.0	253.46747	312.58622	513.0	0.97410
2	39.792202	-86.515246	6.956160e+07	742.77365	703.0	323.39011	291.85520	378.0	0.95238
3	18.396103	-66.104169	1.105793e+06	803.42018	782.0	297.39258	259.30316	368.0	0.94693
4	39.195573	-96.569366	2.554403e+06	938.56493	881.0	392.44096	1005.42886	1704.0	0.99286
...	...	...	...	...	...	...	...	...	...
27316	18.076060	-66.358379	6.970300e+05	439.42839	419.0	140.29970	170.00000	170.0	1.00000
27317	40.158138	-75.307271	5.077337e+06	1813.19253	1788.0	492.92300	64.84927	471.0	0.85435
27318	40.410316	-103.814003	1.323262e+09	849.39107	834.0	336.47530	120.91448	195.0	0.93846
27319	32.904866	-97.162151	1.865230e+07	1972.45746	1843.0	633.02173	19.16328	157.0	1.00000
27320	36.064754	-115.152237	7.796308e+06	949.84199	924.0	198.82109	555.87526	1031.0	0.94956

26585 rows × 62 columns

In [123]:

```
# exclude columns you don't want
fa_train_df = fa_train_df[fa_train_df.columns[~fa_train_df.columns.isin(['SUMLEVEL',
'lat', 'lng',

'ALand', # 'AWater'

])]]
```

In [124]:

```
from factor_analyzer import FactorAnalyzer
```

```
import warnings
warnings.filterwarnings('ignore')
```

In [125]:

```
# Create factor analysis object and perform factor analysis
fa = FactorAnalyzer( rotation=None, n_factors = 25)
fa.fit(fa_train_df)
# Check Eigenvalues
ev, v = fa.get_eigenvalues()
ev
```

Out[125]:

```
array([1.55329901e+01, 9.03576244e+00, 7.86051899e+00, 4.53795677e+00,
       3.69180130e+00, 2.40808907e+00, 2.03947300e+00, 1.39337603e+00,
       1.25745489e+00, 1.12774306e+00, 9.63292652e-01, 9.04911514e-01,
       8.04226650e-01, 7.14077812e-01, 6.47959933e-01, 5.70217633e-01,
       5.01418519e-01, 4.58711541e-01, 3.96025793e-01, 3.75410399e-01,
       3.44496779e-01, 3.12997351e-01, 3.03211509e-01, 2.55376538e-01,
       2.39060996e-01, 2.36269154e-01, 2.05141645e-01, 1.98653046e-01,
       1.85841411e-01, 1.66327418e-01, 1.38553254e-01, 1.33794794e-01,
       1.25738288e-01, 1.06952390e-01, 9.84162717e-02, 9.43682387e-02,
       9.04514665e-02, 9.01820921e-02, 8.44651399e-02, 5.82787474e-02,
       4.95456915e-02, 3.94433134e-02, 3.36086780e-02, 2.72376548e-02,
       2.42653627e-02, 2.15220732e-02, 2.04733148e-02, 1.64264823e-02,
       1.59691767e-02, 1.52654487e-02, 1.41132773e-02, 8.33322350e-03,
       8.04333272e-03, 6.60440597e-03, 4.29961720e-03, 3.11840604e-03,
       1.03023643e-03, 7.05646294e-04, 1.03184254e-16])
```

In [126]:

```
print(sorted(ev, reverse=True))
```

```
[15.532990144155242, 9.035762435984875, 7.860518985795112, 4.537956767800304, 3.691801300713
6117, 2.4080890699749076, 2.0394730011699096, 1.3933760288137695, 1.2574548850200566, 1.1277
430626632703, 0.9632926523858844, 0.9049115141001577, 0.8042266500474491, 0.7140778124540662
, 0.6479599326989954, 0.5702176328106986, 0.5014185190919357, 0.45871154127266056, 0.3960257
925613, 0.37541039930140746, 0.34449677919656485, 0.31299735117375677, 0.30321150867821456,
0.2553765382935662, 0.23906099594815539, 0.23626915397944842, 0.20514164452140324, 0.1986530
4619528312, 0.1858414108281487, 0.16632741841031118, 0.13855325396976337, 0.1337947937261519
7, 0.1257382884860582, 0.10695238989479233, 0.09841627172312306, 0.09436823866766242, 0.0904
5146651659794, 0.09018209214436154, 0.08446513994338599, 0.05827874742869898, 0.049545691498
353785, 0.039443313355127754, 0.03360867797746703, 0.027237654836407867, 0.02426536272518814
4, 0.021522073237485544, 0.020473314814945468, 0.01642648227807213, 0.015969176655571522, 0.
015265448650628528, 0.014113277291217526, 0.008333223495333216, 0.008043332719859462, 0.0066
044059670556975, 0.004299617200940275, 0.0031184060353282137, 0.001030236426329815, 0.000705
6462936197111, 1.0318425394203536e-16]
```

In [127]:

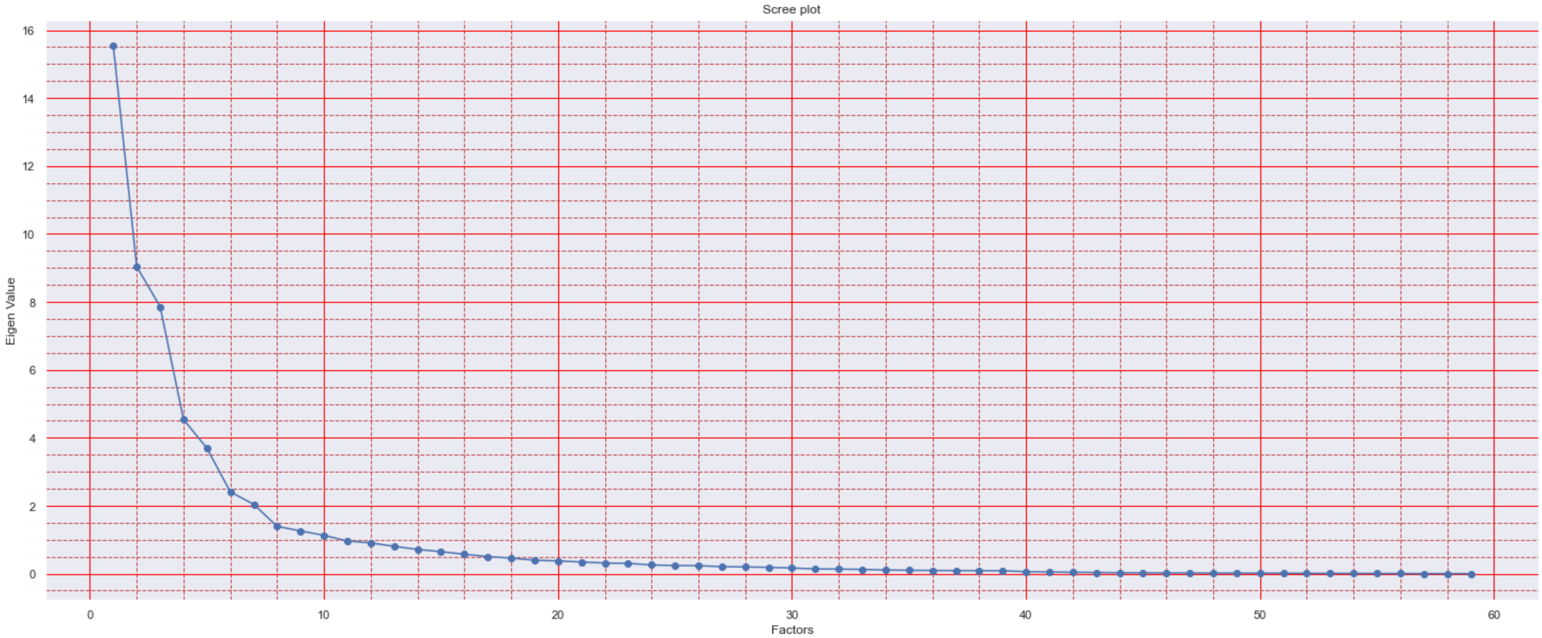
```
loadings = fa.loadings_
```

In [128]:

```
xvals = range(1, fa_train_df.shape[1]+1)
```

In [129]:

```
sns.set()
plt.figure(figsize = (25,10))
plt.scatter(xvals, ev)
plt.plot(xvals, ev)
plt.title('Scree plot')
plt.xlabel('Factors')
plt.ylabel('Eigen Value')
plt.grid(color = 'red', )
plt.grid(b=True, which='minor', color='r', linestyle='--')
plt.minorticks_on()
plt.show()
```



In [130]:

```
Factors  = pd.DataFrame.from_records(loadings)

Factors = Factors.add_prefix('Factor ')

Factors.index = fa_train_df.columns
Factors
```

Out[130]:

	Factor 0	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
rent_mean	0.760475	-0.063396	0.324753	0.157986	-0.135593	-0.147903	0.048739	0.144837	0.201937
rent_median	0.717884	-0.059129	0.319583	0.139694	-0.123943	-0.165580	0.053359	0.152174	0.204098
rent_stdev	0.576379	-0.067221	0.222542	0.187015	-0.179919	0.065701	0.071393	-0.005531	0.020369
rent_sample_weight	-0.442516	0.360121	0.258555	-0.124973	-0.179440	0.558241	-0.100781	-0.012571	-0.178091
rent_samples	-0.157643	0.409892	0.470609	-0.099507	-0.303518	0.613789	-0.099845	0.042845	-0.058124
rent_gt_10	-0.072528	0.072656	0.328015	0.272110	0.193265	-0.046130	-0.169811	0.360298	0.034659
rent_gt_15	-0.127099	0.055246	0.456564	0.429299	0.200224	-0.039068	-0.155473	0.295623	0.016905
rent_gt_20	-0.226160	0.016619	0.519126	0.583447	0.196329	-0.035268	-0.135473	0.201811	-0.008604
rent_gt_25	-0.283097	-0.006203	0.516834	0.627992	0.165774	-0.047905	-0.094716	0.023931	-0.023352
rent_gt_30	-0.310527	-0.018712	0.512883	0.679912	0.155216	-0.085214	-0.088437	-0.120305	-0.013475
rent_gt_35	-0.303332	-0.032649	0.489806	0.669785	0.124701	-0.102566	-0.084462	-0.199870	-0.007831
rent_gt_40	-0.299957	-0.041694	0.477595	0.662098	0.112784	-0.108338	-0.087700	-0.267211	-0.004273
rent_gt_50	-0.273391	-0.058043	0.428189	0.570540	0.077252	-0.083559	-0.073795	-0.249037	-0.004694
hi_mean	0.955017	-0.042475	-0.015172	0.002751	-0.111454	-0.163394	0.019603	0.003934	0.008861
hi_median	0.924299	-0.025099	-0.019047	-0.034585	-0.078977	-0.221515	0.019983	0.032594	0.017929
hi_stdev	0.894348	-0.085016	-0.003761	0.110077	-0.191813	0.019422	0.015756	-0.097577	-0.023820

hi_sample_weight	-0.281855	0.851851	-0.014944	0.105316	-0.060737	0.363291	-0.046057	0.013429	-0.031180
hi_samples	0.225212	0.915959	-0.018944	0.081977	-0.106297	0.233828	-0.054173	0.024860	0.001746
family_mean	0.951528	-0.071286	-0.041563	0.029775	-0.130450	-0.046964	-0.051531	-0.041000	0.010827
family_median	0.926729	-0.069712	-0.041594	0.016969	-0.124799	-0.083283	-0.043708	-0.033533	0.007342
family_stdev	0.826486	-0.080479	0.006611	0.108359	-0.182463	0.113900	-0.047109	-0.113731	-0.011419
family_sample_weight	-0.252024	0.889236	-0.043062	0.095819	-0.010307	-0.005668	0.137770	0.045704	-0.054491
family_samples	0.295321	0.913788	-0.078433	0.086047	-0.030528	-0.097544	0.082155	0.041502	-0.043413
hc_mortgage_mean	0.814993	-0.140474	0.325411	0.149810	-0.268517	0.089431	0.170773	-0.007925	-0.009386
hc_mortgage_median	0.795080	-0.141585	0.335093	0.140629	-0.267696	0.071978	0.165425	-0.002397	-0.014264
hc_mortgage_stdev	0.705367	-0.115090	0.116148	0.184739	-0.193527	0.149544	0.161691	-0.053032	0.001112
hc_mortgage_sample_weight	0.033047	0.778903	-0.334362	-0.005133	0.251696	-0.177235	-0.116068	0.076850	-0.010700
hc_mortgage_samples	0.513082	0.741510	-0.124765	0.040872	0.160367	-0.212380	-0.057590	0.101651	-0.010681
hc_mean	0.729933	-0.164216	0.228162	0.159747	-0.320547	0.136792	0.075345	-0.058224	-0.130386
hc_median	0.699403	-0.157526	0.232314	0.147746	-0.311856	0.125937	0.067263	-0.053286	-0.129474
hc_stdev	0.561939	-0.123664	0.080254	0.206635	-0.320587	0.162003	0.125380	-0.099886	-0.044677
hc_samples	0.040439	0.561232	-0.628977	0.326710	-0.046385	-0.026349	0.107329	-0.170841	0.120835
hc_sample_weight	-0.169444	0.545713	-0.643613	0.263688	0.008750	-0.061522	0.103571	-0.161872	0.148816
home_equity_second_mortgage	0.192335	0.096893	0.356434	-0.235778	0.603279	0.172285	0.307161	-0.202241	0.214858
second_mortgage	0.215819	0.088124	0.381661	-0.228729	0.619787	0.179319	0.335505	-0.220679	0.245391
home_equity	0.630631	0.019653	0.309765	-0.114517	0.488296	0.170592	0.126872	-0.029844	-0.178880
debt	0.505040	0.173117	0.509058	-0.268231	0.272997	-0.112529	-0.164514	0.307669	-0.112910
second_mortgage_cdf	-0.319169	-0.144912	-0.201991	0.165774	-0.599197	-0.084956	-0.223079	0.143555	-0.116589
home_equity_cdf	-0.650725	-0.041394	-0.272858	0.101310	-0.503442	-0.161047	-0.095260	0.018293	0.183264
debt_cdf	-0.494496	-0.162408	-0.530212	0.271567	-0.252608	0.133740	0.144241	-0.300614	0.080905
hs_degree	0.687511	0.003140	-0.254227	-0.042930	0.178590	0.205403	-0.545724	-0.054562	0.054694
hs_degree_male	0.680387	0.004870	-0.221397	-0.041693	0.158627	0.208617	-0.527774	-0.048292	0.041752
hs_degree_female	0.652225	0.001020	-0.274125	-0.043365	0.185522	0.189771	-0.525977	-0.055940	0.071324
male_age_mean	0.264362	-0.194492	-0.634425	0.430888	0.131008	0.325858	0.071820	0.143056	0.172480
male_age_median	0.335537	-0.171819	-0.637608	0.404396	0.159464	0.238244	0.099692	0.169336	0.175849
male_age_stdev	0.049833	-0.020338	-0.511379	0.293379	0.195964	-0.023146	0.198547	0.016471	-0.500323
male_age_sample_weight	0.153491	0.884079	0.167072	0.007167	-0.157445	-0.067401	0.025378	-0.066505	0.092770
male_age_samples	0.199813	0.921159	0.115816	0.019202	-0.135034	-0.059659	0.083009	0.009413	0.045143
female_age_mean	0.201879	-0.190332	-0.615134	0.465952	0.163001	0.376516	0.071278	0.162027	0.078768
female_age_median	0.271309	-0.167022	-0.636358	0.441313	0.197093	0.257314	0.100192	0.166995	0.076976
female_age_stdev	-0.057830	-0.020947	-0.433317	0.234013	0.158127	0.047137	0.201101	0.028094	-0.519360

female_age_sample_weight	0.152581	0.891817	0.174671	0.035905	-0.146199	-0.036469	-0.008760	-0.066913	0.031751
female_age_samples	0.204339	0.938734	0.114113	0.058287	-0.116408	-0.031715	0.054665	0.012574	-0.027062
pct_own	0.458917	0.073366	-0.592291	0.154113	0.281658	-0.430103	0.023573	-0.015681	0.000110
married	0.530469	0.126577	-0.475233	0.108788	0.126237	-0.278091	0.157451	0.121383	-0.115889
married_snp	-0.359319	-0.060522	0.291031	0.034920	-0.192854	0.125582	0.452835	0.370047	0.154781
separated	-0.358463	-0.051877	0.158742	0.028419	-0.073453	0.116475	0.255099	0.253754	0.082282
divorced	-0.393144	-0.042448	-0.205136	0.018705	0.162162	0.281798	-0.118419	0.059024	0.047672
Bad_Debt	0.628268	0.018681	0.323725	-0.114805	0.495392	0.170665	0.140063	-0.037879	-0.155709

In [131]:

```
fa = FactorAnalyzer( rotation="varimax", n_factors = 12)
fa.fit(fa_train_df)
loadings = fa.loadings_
```

In [132]:

```
Factors  = pd.DataFrame.from_records(loadings)

Factors = Factors.add_prefix('Factor ')

Factors.index = fa_train_df.columns
Factors
```

Out[132]:

	Factor 0	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
rent_mean	0.789021	0.040012	0.038941	-0.021709	0.069256	0.121732	0.088021	0.057797	0.145601
rent_median	0.742003	0.038226	0.031515	-0.035057	0.053643	0.134560	0.089465	0.052404	0.149594
rent_stdev	0.659926	0.018945	0.085863	0.045249	0.049560	-0.056153	0.034718	0.061515	0.051932
rent_sample_weight	-0.292390	0.250003	0.042910	-0.151727	-0.102631	-0.765664	-0.035708	-0.005133	-0.012322
rent_samples	0.039249	0.330013	0.052201	-0.180707	-0.094791	-0.865473	0.013352	-0.004131	0.062554
rent_gt_10	-0.030330	0.047236	0.208386	-0.042875	-0.004896	-0.023455	0.041221	0.041593	0.076880
rent_gt_15	-0.011144	0.032773	0.375047	-0.050537	-0.044303	-0.053456	0.043662	0.030643	0.049670
rent_gt_20	-0.035040	0.001215	0.585450	-0.038916	-0.106707	-0.081913	0.032065	0.014050	0.023625
rent_gt_25	-0.055213	-0.012471	0.746023	-0.037572	-0.137342	-0.081977	0.018918	0.001528	0.012203
rent_gt_30	-0.067113	-0.015637	0.865941	-0.039781	-0.135501	-0.056896	0.011162	-0.012296	0.003153
rent_gt_35	-0.057654	-0.021511	0.938392	-0.045885	-0.119311	-0.039949	-0.006243	-0.016248	-0.001093
rent_gt_40	-0.056545	-0.028555	0.941935	-0.052699	-0.107208	-0.038135	-0.010077	-0.014898	-0.009015
rent_gt_50	-0.044499	-0.047820	0.831141	-0.060800	-0.100731	-0.051270	-0.015498	-0.015243	-0.025855
hi_mean	0.829689	0.081603	-0.193967	0.041917	0.291149	0.295301	0.054728	0.103674	0.134906
hi_median	0.775399	0.091510	-0.214237	0.013129	0.278033	0.337109	0.062767	0.103761	0.162272
hi_stdev	0.846488	0.041184	-0.095019	0.113473	0.284510	0.128669	0.021573	0.088188	0.035394

hi_sample_weight	-0.302913	0.798058	0.057955	0.078790	-0.025504	-0.449996	-0.013072	-0.022945	-0.080585	(
hi_samples	0.110619	0.926326	-0.053162	0.076272	0.141768	-0.262349	0.027480	0.019410	0.023996	
family_mean	0.828524	0.049253	-0.176453	0.096145	0.369300	0.202573	0.033460	0.104102	0.089748	-(
family_median	0.801321	0.047836	-0.178996	0.074197	0.350845	0.224058	0.031119	0.097554	0.096293	-(
family_stdev	0.773963	0.029345	-0.078128	0.129495	0.315857	0.039352	0.020384	0.089643	0.021994	-(
family_sample_weight	-0.290470	0.860767	0.038449	-0.028340	-0.185132	-0.089482	0.010133	-0.036523	-0.039670	(
family_samples	0.139673	0.953192	-0.071167	0.003947	0.054655	0.089947	0.046805	0.026467	0.062615	(
hc_mortgage_mean	0.942393	-0.022254	0.003640	0.011175	0.004836	-0.032548	0.071605	0.130529	0.062018	(
hc_mortgage_median	0.925628	-0.026992	0.007762	-0.009202	-0.002076	-0.030055	0.067052	0.124466	0.071186	(
hc_mortgage_stdev	0.771064	-0.003261	-0.017650	0.154609	0.057231	-0.013570	0.057544	0.125013	-0.031623	-(
hc_mortgage_sample_weight	-0.300656	0.772373	-0.098838	0.117333	0.224082	0.246919	0.033721	0.012140	0.149346	-(
hc_mortgage_samples	0.209495	0.791455	-0.098561	0.069911	0.260278	0.293403	0.101181	0.092701	0.252832	(
hc_mean	0.872354	-0.051827	-0.008293	0.025025	0.081870	-0.049513	0.004439	0.016034	0.060164	(
hc_median	0.831823	-0.050818	-0.006428	0.010195	0.079423	-0.049352	0.002619	0.018487	0.063249	(
hc_stdev	0.697744	-0.018593	-0.000199	0.114021	0.038425	-0.058762	-0.020936	-0.030230	-0.077644	-(
hc_samples	-0.097223	0.629439	-0.068681	0.397885	0.121332	0.243475	-0.113148	-0.089472	-0.461786	-(
hc_sample_weight	-0.310923	0.578921	-0.062640	0.353524	0.062729	0.235259	-0.115807	-0.104263	-0.461142	-(
home_equity_second_mortgage	0.035851	0.030256	0.008863	-0.089052	-0.001548	-0.035698	0.905542	0.175228	0.065388	(
second_mortgage	0.067418	0.023287	0.020624	-0.085433	-0.010200	-0.033926	0.967571	0.147069	0.064568	(
home_equity	0.381937	0.026935	-0.024957	0.000623	0.172420	0.029206	0.361087	0.809731	0.156351	(
debt	0.310563	0.157140	-0.027205	-0.241165	0.145058	0.024562	0.249192	0.245264	0.750527	
second_mortgage_cdf	-0.088111	-0.107912	0.019417	0.007253	-0.117169	-0.090330	-0.774682	-0.183121	-0.135384	-(
home_equity_cdf	-0.381215	-0.050912	0.028408	-0.026020	-0.217667	-0.046808	-0.376446	-0.746963	-0.197439	-(
debt_cdf	-0.315979	-0.149139	0.022326	0.258084	-0.111850	-0.036980	-0.244838	-0.242864	-0.751708	-
hs_degree	0.330834	0.029097	-0.165616	0.225304	0.866310	-0.005369	0.066930	0.064640	0.125856	-(
hs_degree_male	0.347975	0.032828	-0.160559	0.207360	0.802019	-0.015616	0.062763	0.072629	0.130850	-(
hs_degree_female	0.300928	0.029297	-0.170518	0.235183	0.803285	0.028157	0.063811	0.068766	0.110956	-(
male_age_mean	0.122631	-0.085682	-0.065478	0.913235	0.130293	0.104775	-0.059422	-0.006027	-0.107807	-(
male_age_median	0.161157	-0.052976	-0.092536	0.872429	0.129297	0.204770	-0.040396	0.010207	-0.054947	-(
male_age_stdev	-0.040713	0.017978	-0.023633	0.295852	0.058230	0.177173	-0.029171	-0.014056	-0.086860	-(
male_age_sample_weight	0.108499	0.884899	0.019727	-0.199644	-0.005865	-0.055349	0.026812	0.034457	-0.024881	(
male_age_samples	0.133978	0.937625	-0.024449	-0.137432	-0.038063	-0.039567	0.043924	0.025528	0.036177	(
female_age_mean	0.075875	-0.095166	-0.029350	0.877510	0.126751	0.038853	-0.056413	0.002319	-0.109174	-(
female_age_median	0.106536	-0.060526	-0.055817	0.866467	0.125945	0.166771	-0.038758	0.017605	-0.067553	-(
female_age_stdev	-0.110127	-0.002930	-0.021278	0.247954	-0.000866	0.068087	-0.045231	0.004096	-0.088347	-(

female_age_sample_weight	0.110601	0.893191	0.045047	-0.203096	0.029983	-0.097395	0.026160	0.023393	-0.013725	(
female_age_samples	0.136863	0.954957	0.001722	-0.131260	-0.002110	-0.076364	0.043767	0.016854	0.052373	(
pct_own	0.123590	0.176666	-0.149570	0.328869	0.310353	0.749438	0.008302	0.039394	-0.004898	-
married	0.286620	0.224887	-0.233893	0.294934	0.166046	0.487008	0.006630	0.031694	0.065997	-(
married_snp	-0.089221	-0.072213	0.110052	-0.031297	-0.545451	-0.229939	0.002106	-0.083934	0.073016	(
separated	-0.181802	-0.076564	0.090799	0.019243	-0.395944	-0.194935	-0.001805	-0.073670	0.068161	(
divorced	-0.437719	-0.099307	0.011740	0.236724	0.034533	-0.208559	0.002501	-0.070071	-0.026605	(
Bad_Debt	0.386428	0.024802	-0.018312	-0.001947	0.163962	0.028109	0.399610	0.774981	0.160535	(

In [133]:

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

Factors_df = round(Factors.loc[['hs_degree', 'hs_degree_male',
'hs_degree_female',"male_age_median", "female_age_median", "home_equity_second_mortgage",
'second_mortgage', 'second_mortgage_cdf', 'pct_own', 'Bad_Debt'], :], 2)
```

In [134]:

```
def color_negative_red(value):
  """
  Colors elements in a dateframe
  green if positive and red if
  negative. Does not color NaN
  values.
  """

  if value < -0.6:
    color = 'brown'
  elif value > 0.6:
    color = 'green'
  else:
    color = 'blue'

  return 'color: %s' % color
```

In [135]:

```
Factors_df.style.applymap(color_negative_red)
```

Out[135]:

	Factor 0	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Fa
hs_degree	0.330000	0.030000	-0.170000	0.230000	0.870000	-0.010000	0.070000	0.060000	0.130000	-0.(
hs_degree_male	0.350000	0.030000	-0.160000	0.210000	0.800000	-0.020000	0.060000	0.070000	0.130000	-0.(
hs_degree_female	0.300000	0.030000	-0.170000	0.240000	0.800000	0.030000	0.060000	0.070000	0.110000	-0.(
male_age_median	0.160000	-0.050000	-0.090000	0.870000	0.130000	0.200000	-0.040000	0.010000	-0.050000	-0.(
female_age_median	0.110000	-0.060000	-0.060000	0.870000	0.130000	0.170000	-0.040000	0.020000	-0.070000	-0.(

home_equity_second_mortgage	0.040000	0.030000	0.010000	-0.090000	-0.000000	-0.040000	0.910000	0.180000	0.070000	0.(
second_mortgage	0.070000	0.020000	0.020000	-0.090000	-0.010000	-0.030000	0.970000	0.150000	0.060000	0.(
second_mortgage_cdf	-0.090000	-0.110000	0.020000	0.010000	-0.120000	-0.090000	-0.770000	-0.180000	-0.140000	-0.(
pct_own	0.120000	0.180000	-0.150000	0.330000	0.310000	0.750000	0.010000	0.040000	-0.000000	-0.(
Bad_Debt	0.390000	0.020000	-0.020000	-0.000000	0.160000	0.030000	0.400000	0.770000	0.160000	0.(

Looks like "Related parameters" are loading on Unique Factors.

In [136]:

```
len(fa_train_df.columns)
```

Out[136]:

```
59
```

In [137]:

```
# Get variance of each factors
fact_variance  = fa.get_factor_variance()
fact_variance
```

Out[137]:

```
(array([11.8093628 ,  8.87462738,  4.71407815,  4.40034679,  3.77193593,
         3.35575387,  3.02878697,  2.20823883,  2.02803283,  1.83414926,
         1.66211152,  0.78492779]),
 array([0.20015869, 0.15041741, 0.07989963, 0.07458215, 0.06393112,
        0.05687718, 0.05133537, 0.03742778, 0.03437344, 0.03108728,
        0.02817138, 0.01330386]),
 array([0.20015869, 0.3505761 , 0.43047573, 0.50505788, 0.568989  ,
        0.62586619, 0.67720156, 0.71462933, 0.74900277, 0.78009005,
        0.80826143, 0.82156529]))
```

In [138]:

```
Factor_variance  = pd.DataFrame.from_records(fact_variance)

Factor_variance = Factor_variance.add_prefix('Factor ')

Factor_variance.index = ['SS Loadings', 'Proportion Var', 'Cumulative Var']
round(Factor_variance, 2)
```

Out[138]:

	Factor 0	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Factor 9	Factor 10	Factor 11
SS Loadings	11.81	8.87	4.71	4.40	3.77	3.36	3.03	2.21	2.03	1.83	1.66	0.78
Proportion Var	0.20	0.15	0.08	0.07	0.06	0.06	0.05	0.04	0.03	0.03	0.03	0.01
Cumulative Var	0.20	0.35	0.43	0.51	0.57	0.63	0.68	0.71	0.75	0.78	0.81	0.82

Data Modeling :

Project Task: Week 4

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.

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

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

c) Keep below considerations while building a linear regression model. Data Modeling :

• Variables should have significant impact on predicting Monthly mortgage and owner costs

• Utilize all predictor variable to start with initial hypothesis

• R square of 60 percent and above should be achieved

• Ensure Multi-collinearity does not exist in dependent variables

• Test if predicted variable is normally distributed

In [139]:

```
train_df = pd.read_csv('train.csv')
```

In [140]:

```
train_df.head()
```

Out[140]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_co
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	

In [141]:

```
train_df.isna().sum()
```

Out[141]:

```
UID                           0
BLOCKID                   27321
SUMLEVEL                      0
COUNTYID                      0
STATEID                       0
state                         0
state_ab                      0
city                          0
place                         0
type                          0
primary                       0
zip_code                      0
area_code                     0
lat                           0
lng                           0
ALand                         0
AWater                        0
pop                           0
male_pop                      0
female_pop                    0
rent_mean                   314
rent_median                 314
rent_stdev                  314
rent_sample_weight          314
rent_samples                314
rent_gt_10                  314
rent_gt_15                  314
rent_gt_20                  314
rent_gt_25                  314
rent_gt_30                  314
rent_gt_35                  314
rent_gt_40                  314
rent_gt_50                  314
```

```
universe_samples                  0
used_samples                      0
hi_mean                         268
hi_median                       268
hi_stdev                        268
hi_sample_weight                268
hi_samples                      268
family_mean                     298
family_median                   298
family_stdev                    298
family_sample_weight            298
family_samples                  298
hc_mortgage_mean                573
hc_mortgage_median              573
hc_mortgage_stdev               573
hc_mortgage_sample_weight       573
hc_mortgage_samples             573
hc_mean                         600
hc_median                       600
hc_stdev                        600
hc_samples                      600
hc_sample_weight                600
home_equity_second_mortgage     457
second_mortgage                 457
home_equity                     457
debt                            457
second_mortgage_cdf             457
home_equity_cdf                 457
debt_cdf                        457
hs_degree                       190
hs_degree_male                  200
hs_degree_female                223
male_age_mean                   189
male_age_median                 189
male_age_stdev                  189
male_age_sample_weight          189
male_age_samples                189
female_age_mean                 206
female_age_median               206
female_age_stdev                206
female_age_sample_weight        206
female_age_samples              206
pct_own                         268
married                         191
married_snp                     191
separated                       191
divorced                        191
dtype: int64
```

In [142]:

```
hf.miss_df(train_df)
```

Out[142]:

	count	percentage
UID	0	0.00
BLOCKID	27321	100.00
SUMLEVEL	0	0.00
COUNTYID	0	0.00
STATEID	0	0.00
state	0	0.00
state_ab	0	0.00

city	0	0.00
place	0	0.00
type	0	0.00
primary	0	0.00
zip_code	0	0.00
area_code	0	0.00
lat	0	0.00
lng	0	0.00
ALand	0	0.00
AWater	0	0.00
pop	0	0.00
male_pop	0	0.00
female_pop	0	0.00
rent_mean	314	1.15
rent_median	314	1.15
rent_stdev	314	1.15
rent_sample_weight	314	1.15
rent_samples	314	1.15
rent_gt_10	314	1.15
rent_gt_15	314	1.15
rent_gt_20	314	1.15
rent_gt_25	314	1.15
rent_gt_30	314	1.15
rent_gt_35	314	1.15
rent_gt_40	314	1.15
rent_gt_50	314	1.15
universe_samples	0	0.00
used_samples	0	0.00
hi_mean	268	0.98
hi_median	268	0.98
hi_stdev	268	0.98
hi_sample_weight	268	0.98
hi_samples	268	0.98
family_mean	298	1.09
family_median	298	1.09

family_stdev	298	1.09
family_sample_weight	298	1.09
family_samples	298	1.09
hc_mortgage_mean	573	2.10
hc_mortgage_median	573	2.10
hc_mortgage_stdev	573	2.10
hc_mortgage_sample_weight	573	2.10
hc_mortgage_samples	573	2.10
hc_mean	600	2.20
hc_median	600	2.20
hc_stdev	600	2.20
hc_samples	600	2.20
hc_sample_weight	600	2.20
home_equity_second_mortgage	457	1.67
second_mortgage	457	1.67
home_equity	457	1.67
debt	457	1.67
second_mortgage_cdf	457	1.67
home_equity_cdf	457	1.67
debt_cdf	457	1.67
hs_degree	190	0.70
hs_degree_male	200	0.73
hs_degree_female	223	0.82
male_age_mean	189	0.69
male_age_median	189	0.69
male_age_stdev	189	0.69
male_age_sample_weight	189	0.69
male_age_samples	189	0.69
female_age_mean	206	0.75
female_age_median	206	0.75
female_age_stdev	206	0.75
female_age_sample_weight	206	0.75
female_age_samples	206	0.75
pct_own	268	0.98

married	191	0.70
married_snp	191	0.70
separated	191	0.70
divorced	191	0.70

In [143]:

```
hf.miss_df(train_df).sort_values(by='percentage', ascending=False)
```

Out[143]:

	count	percentage
BLOCKID	27321	100.00
hc_stdev	600	2.20
hc_sample_weight	600	2.20
hc_samples	600	2.20
hc_mean	600	2.20
hc_median	600	2.20
hc_mortgage_samples	573	2.10
hc_mortgage_stdev	573	2.10
hc_mortgage_median	573	2.10
hc_mortgage_mean	573	2.10
hc_mortgage_sample_weight	573	2.10
second_mortgage_cdf	457	1.67
second_mortgage	457	1.67
home_equity	457	1.67
debt	457	1.67
debt_cdf	457	1.67
home_equity_cdf	457	1.67
home_equity_second_mortgage	457	1.67
rent_gt_40	314	1.15
rent_gt_10	314	1.15
rent_gt_35	314	1.15
rent_gt_30	314	1.15
rent_gt_25	314	1.15
rent_gt_20	314	1.15
rent_gt_50	314	1.15
rent_gt_15	314	1.15
rent_samples	314	1.15

rent_sample_weight	314	1.15
rent_stdev	314	1.15
rent_median	314	1.15
rent_mean	314	1.15
family_samples	298	1.09
family_sample_weight	298	1.09
family_stdev	298	1.09
family_median	298	1.09
family_mean	298	1.09
hi_median	268	0.98
hi_stdev	268	0.98
pct_own	268	0.98
hi_samples	268	0.98
hi_mean	268	0.98
hi_sample_weight	268	0.98
hs_degree_female	223	0.82
female_age_median	206	0.75
female_age_mean	206	0.75
female_age_stdev	206	0.75
female_age_sample_weight	206	0.75
female_age_samples	206	0.75
hs_degree_male	200	0.73
married	191	0.70
married_snp	191	0.70
separated	191	0.70
hs_degree	190	0.70
divorced	191	0.70
male_age_stdev	189	0.69
male_age_sample_weight	189	0.69
male_age_mean	189	0.69
male_age_median	189	0.69
male_age_samples	189	0.69
SUMLEVEL	0	0.00
COUNTYID	0	0.00
STATEID	0	0.00

state	0	0.00
state_ab	0	0.00
city	0	0.00
place	0	0.00
type	0	0.00
primary	0	0.00
used_samples	0	0.00
universe_samples	0	0.00
zip_code	0	0.00
area_code	0	0.00
lat	0	0.00
lng	0	0.00
ALand	0	0.00
AWater	0	0.00
pop	0	0.00
male_pop	0	0.00
female_pop	0	0.00
UID	0	0.00

In [144]:

```
train_df.head()
```

Out[144]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_c
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	

In [145]:

```
null_data = train_df[train_df.isnull().any(axis=1)]
null_data
```

Out[145]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_c

0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66
...	...	...	...	...	...	...	...	...	...	...	...	
27316	279212	NaN	140	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	tract	
27317	277856	NaN	140	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	tract	19
27318	233000	NaN	140	87	8	Colorado	CO	Weldona	Saddle Ridge	City	tract	80
27319	287425	NaN	140	439	48	Texas	TX	Colleyville	Colleyville City	Town	tract	76
27320	265371	NaN	140	3	32	Nevada	NV	Las Vegas	Paradise	City	tract	89

27321 rows × 80 columns

In [146]:

```
train_df.drop('BLOCKID', axis=1, inplace=True)
```

In [147]:

```
test_df.drop('BLOCKID', axis=1, inplace=True)
```

In [148]:

```
train_df.isna().sum()
```

Out[148]:

```
UID                       0
SUMLEVEL                  0
COUNTYID                  0
STATEID                   0
state                     0
state_ab                  0
city                      0
place                     0
type                      0
primary                   0
zip_code                  0
area_code                 0
lat                       0
lng                       0
ALand                     0
AWater                    0
pop                       0
male_pop                  0
female_pop                0
rent_mean               314
rent_median             314
rent_stdev              314
rent_sample_weight      314
rent_samples            314
rent_gt_10              314
rent_gt_15              314
rent_gt_20              314
```

```
rent_gt_25                     314
rent_gt_30                     314
rent_gt_35                     314
rent_gt_40                     314
rent_gt_50                     314
universe_samples                 0
used_samples                     0
hi_mean                        268
hi_median                      268
hi_stdev                       268
hi_sample_weight               268
hi_samples                     268
family_mean                    298
family_median                  298
family_stdev                   298
family_sample_weight           298
family_samples                 298
hc_mortgage_mean               573
hc_mortgage_median             573
hc_mortgage_stdev              573
hc_mortgage_sample_weight      573
hc_mortgage_samples            573
hc_mean                        600
hc_median                      600
hc_stdev                       600
hc_samples                     600
hc_sample_weight               600
home_equity_second_mortgage    457
second_mortgage                457
home_equity                    457
debt                           457
second_mortgage_cdf            457
home_equity_cdf                457
debt_cdf                       457
hs_degree                      190
hs_degree_male                 200
hs_degree_female               223
male_age_mean                  189
male_age_median                189
male_age_stdev                 189
male_age_sample_weight         189
male_age_samples               189
female_age_mean                206
female_age_median              206
female_age_stdev               206
female_age_sample_weight       206
female_age_samples             206
pct_own                        268
married                        191
married_snp                    191
separated                      191
divorced                       191
dtype: int64
```

In [149]:

```
test_df.isna().sum()
```

Out[149]:

```
UID                              0
SUMLEVEL                         0
COUNTYID                         0
STATEID                          0
state                            0
state_ab                         0
city                             0
place                            0
type                             0
primary                          0
zip_code                         0
```

```
area_code                        0
lat                              0
lng                              0
ALand                            0
AWater                           0
pop                              0
male_pop                         0
female_pop                       0
rent_mean                      148
rent_median                    148
rent_stdev                     148
rent_sample_weight             148
rent_samples                   148
rent_gt_10                     149
rent_gt_15                     149
rent_gt_20                     149
rent_gt_25                     149
rent_gt_30                     149
rent_gt_35                     149
rent_gt_40                     149
rent_gt_50                     149
universe_samples                 0
used_samples                     0
hi_mean                        122
hi_median                      122
hi_stdev                       122
hi_sample_weight               122
hi_samples                     122
family_mean                    136
family_median                  136
family_stdev                   136
family_sample_weight           136
family_samples                 136
hc_mortgage_mean               268
hc_mortgage_median             268
hc_mortgage_stdev              268
hc_mortgage_sample_weight      268
hc_mortgage_samples            268
hc_mean                        290
hc_median                      290
hc_stdev                       290
hc_samples                     290
hc_sample_weight               290
home_equity_second_mortgage    220
second_mortgage                220
home_equity                    220
debt                           220
second_mortgage_cdf            220
home_equity_cdf                220
debt_cdf                       220
hs_degree                       85
hs_degree_male                  89
hs_degree_female               105
male_age_mean                   84
male_age_median                 84
male_age_stdev                  84
male_age_sample_weight          84
male_age_samples                84
female_age_mean                 96
female_age_median               96
female_age_stdev                96
female_age_sample_weight        96
female_age_samples              96
pct_own                        122
married                         84
married_snp                     84
separated                       84
divorced                        84
dtype: int64
```

In [150]:

```
train_df = train_df.dropna()
train_df = train_df.reset_index(drop=True)
```

In [151]:

```
test_df = test_df.dropna()
test_df = test_df.reset_index(drop=True)
```

In [152]:

```
train_df.shape
```

Out[152]:

```
(26585, 79)
```

In [153]:

```
test_df.shape
```

Out[153]:

```
(11355, 79)
```

In [154]:

```
train_df[cat_columns]
```

Out[154]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574
2	245683	63	18	Indiana	IN	Danville	Danville	City	46122	317
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787
4	247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785
...	...	...	...	...	...	...	...	...	...	...
26580	279212	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	769	787
26581	277856	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	19422	215
26582	233000	87	8	Colorado	CO	Weldona	Saddle Ridge	City	80653	970
26583	287425	439	48	Texas	TX	Colleyville	Colleyville City	Town	76034	817
26584	265371	3	32	Nevada	NV	Las Vegas	Paradise	City	89123	702

26585 rows × 10 columns

In [155]:

```
train_df[num_variables(train_df)]
```

Out[155]:

	lat	lng	ALand	rent_mean	rent_median	rent_stdev	rent_sample_weight	rent_samples	rent_gt_10	r
0	42.840812	-75.501524	2.021834e+08	769.38638	784.0	232.63967	272.34441	362.0	0.86761	
1	41.701441	-86.266614	1.560828e+06	804.87924	848.0	253.46747	312.58622	513.0	0.97410	
2	39.792202	-86.515246	6.956160e+07	742.77365	703.0	323.39011	291.85520	378.0	0.95238	
3	18.396103	-66.104169	1.105793e+06	803.42018	782.0	297.39258	259.30316	368.0	0.94693	
4	39.195573	-96.569366	2.554403e+06	938.56493	881.0	392.44096	1005.42886	1704.0	0.99286	

...	...	...	...	...	...	...	...	...	...
26580	18.076060	-66.358379	6.970300e+05	439.42839	419.0	140.29970	170.00000	170.0	1.00000
26581	40.158138	-75.307271	5.077337e+06	1813.19253	1788.0	492.92300	64.84927	471.0	0.85435
26582	40.410316	-103.814003	1.323262e+09	849.39107	834.0	336.47530	120.91448	195.0	0.93846
26583	32.904866	-97.162151	1.865230e+07	1972.45746	1843.0	633.02173	19.16328	157.0	1.00000
26584	36.064754	-115.152237	7.796308e+06	949.84199	924.0	198.82109	555.87526	1031.0	0.94956

26585 rows × 61 columns

In [156]:

```
train_df.drop('SUMLEVEL', inplace = True, axis = 1)
```

In [157]:

```
test_df.drop('SUMLEVEL', inplace = True, axis = 1)
```

In [158]:

```
train_df[num_variables(train_df)]
```

Out[158]:

	lat	lng	ALand	rent_mean	rent_median	rent_stdev	rent_sample_weight	rent_samples	rent_gt_10	r
0	42.840812	-75.501524	2.021834e+08	769.38638	784.0	232.63967	272.34441	362.0	0.86761	
1	41.701441	-86.266614	1.560828e+06	804.87924	848.0	253.46747	312.58622	513.0	0.97410	
2	39.792202	-86.515246	6.956160e+07	742.77365	703.0	323.39011	291.85520	378.0	0.95238	
3	18.396103	-66.104169	1.105793e+06	803.42018	782.0	297.39258	259.30316	368.0	0.94693	
4	39.195573	-96.569366	2.554403e+06	938.56493	881.0	392.44096	1005.42886	1704.0	0.99286	
...	...	...	...	...	...	...	...	...	...	
26580	18.076060	-66.358379	6.970300e+05	439.42839	419.0	140.29970	170.00000	170.0	1.00000	
26581	40.158138	-75.307271	5.077337e+06	1813.19253	1788.0	492.92300	64.84927	471.0	0.85435	
26582	40.410316	-103.814003	1.323262e+09	849.39107	834.0	336.47530	120.91448	195.0	0.93846	
26583	32.904866	-97.162151	1.865230e+07	1972.45746	1843.0	633.02173	19.16328	157.0	1.00000	
26584	36.064754	-115.152237	7.796308e+06	949.84199	924.0	198.82109	555.87526	1031.0	0.94956	

26585 rows × 61 columns

In [159]:

```
num_2_cat = ['UID','COUNTYID', 'STATEID', 'zip_code', 'area_code', 'lat', 'lng']
```

In [160]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26585 entries, 0 to 26584
Data columns (total 78 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   UID                           26585 non-null  int64
```

```
 1   COUNTYID                     26585 non-null  int64
 2   STATEID                      26585 non-null  int64
 3   state                        26585 non-null  object
 4   state_ab                     26585 non-null  object
 5   city                         26585 non-null  object
 6   place                        26585 non-null  object
 7   type                         26585 non-null  object
 8   primary                      26585 non-null  object
 9   zip_code                     26585 non-null  int64
 10  area_code                    26585 non-null  int64
 11  lat                          26585 non-null  float64
 12  lng                          26585 non-null  float64
 13  ALand                        26585 non-null  float64
 14  AWater                       26585 non-null  int64
 15  pop                          26585 non-null  int64
 16  male_pop                     26585 non-null  int64
 17  female_pop                   26585 non-null  int64
 18  rent_mean                    26585 non-null  float64
 19  rent_median                  26585 non-null  float64
 20  rent_stdev                   26585 non-null  float64
 21  rent_sample_weight           26585 non-null  float64
 22  rent_samples                 26585 non-null  float64
 23  rent_gt_10                   26585 non-null  float64
 24  rent_gt_15                   26585 non-null  float64
 25  rent_gt_20                   26585 non-null  float64
 26  rent_gt_25                   26585 non-null  float64
 27  rent_gt_30                   26585 non-null  float64
 28  rent_gt_35                   26585 non-null  float64
 29  rent_gt_40                   26585 non-null  float64
 30  rent_gt_50                   26585 non-null  float64
 31  universe_samples             26585 non-null  int64
 32  used_samples                 26585 non-null  int64
 33  hi_mean                      26585 non-null  float64
 34  hi_median                    26585 non-null  float64
 35  hi_stdev                     26585 non-null  float64
 36  hi_sample_weight             26585 non-null  float64
 37  hi_samples                   26585 non-null  float64
 38  family_mean                  26585 non-null  float64
 39  family_median                26585 non-null  float64
 40  family_stdev                 26585 non-null  float64
 41  family_sample_weight         26585 non-null  float64
 42  family_samples               26585 non-null  float64
 43  hc_mortgage_mean             26585 non-null  float64
 44  hc_mortgage_median           26585 non-null  float64
 45  hc_mortgage_stdev            26585 non-null  float64
 46  hc_mortgage_sample_weight    26585 non-null  float64
 47  hc_mortgage_samples          26585 non-null  float64
 48  hc_mean                      26585 non-null  float64
 49  hc_median                    26585 non-null  float64
 50  hc_stdev                     26585 non-null  float64
 51  hc_samples                   26585 non-null  float64
 52  hc_sample_weight             26585 non-null  float64
 53  home_equity_second_mortgage  26585 non-null  float64
 54  second_mortgage              26585 non-null  float64
 55  home_equity                  26585 non-null  float64
 56  debt                         26585 non-null  float64
 57  second_mortgage_cdf          26585 non-null  float64
 58  home_equity_cdf              26585 non-null  float64
 59  debt_cdf                     26585 non-null  float64
 60  hs_degree                    26585 non-null  float64
 61  hs_degree_male               26585 non-null  float64
 62  hs_degree_female             26585 non-null  float64
 63  male_age_mean                26585 non-null  float64
 64  male_age_median              26585 non-null  float64
 65  male_age_stdev               26585 non-null  float64
 66  male_age_sample_weight       26585 non-null  float64
 67  male_age_samples             26585 non-null  float64
 68  female_age_mean              26585 non-null  float64
 69  female_age_median            26585 non-null  float64
```

```
 70  female_age_stdev              26585 non-null  float64
 71  female_age_sample_weight      26585 non-null  float64
 72  female_age_samples            26585 non-null  float64
 73  pct_own                       26585 non-null  float64
 74  married                       26585 non-null  float64
 75  married_snp                   26585 non-null  float64
 76  separated                     26585 non-null  float64
 77  divorced                      26585 non-null  float64
dtypes: float64(61), int64(11), object(6)
memory usage: 15.8+ MB
```

In [161]:

```
for col in num_2_cat:
    train_df[col] = train_df[col].astype('category')
    test_df[col] = test_df[col].astype('category')
```

In [162]:

```
print(train_df.info())
print('-----------')
print(test_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26585 entries, 0 to 26584
Data columns (total 78 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   UID                           26585 non-null  category
 1   COUNTYID                      26585 non-null  category
 2   STATEID                       26585 non-null  category
 3   state                         26585 non-null  object
 4   state_ab                      26585 non-null  object
 5   city                          26585 non-null  object
 6   place                         26585 non-null  object
 7   type                          26585 non-null  object
 8   primary                       26585 non-null  object
 9   zip_code                      26585 non-null  category
 10  area_code                     26585 non-null  category
 11  lat                           26585 non-null  category
 12  lng                           26585 non-null  category
 13  ALand                         26585 non-null  float64
 14  AWater                        26585 non-null  int64
 15  pop                           26585 non-null  int64
 16  male_pop                      26585 non-null  int64
 17  female_pop                    26585 non-null  int64
 18  rent_mean                     26585 non-null  float64
 19  rent_median                   26585 non-null  float64
 20  rent_stdev                    26585 non-null  float64
 21  rent_sample_weight            26585 non-null  float64
 22  rent_samples                  26585 non-null  float64
 23  rent_gt_10                    26585 non-null  float64
 24  rent_gt_15                    26585 non-null  float64
 25  rent_gt_20                    26585 non-null  float64
 26  rent_gt_25                    26585 non-null  float64
 27  rent_gt_30                    26585 non-null  float64
 28  rent_gt_35                    26585 non-null  float64
 29  rent_gt_40                    26585 non-null  float64
 30  rent_gt_50                    26585 non-null  float64
 31  universe_samples              26585 non-null  int64
 32  used_samples                  26585 non-null  int64
 33  hi_mean                       26585 non-null  float64
 34  hi_median                     26585 non-null  float64
 35  hi_stdev                      26585 non-null  float64
 36  hi_sample_weight              26585 non-null  float64
 37  hi_samples                    26585 non-null  float64
 38  family_mean                   26585 non-null  float64
 39  family_median                 26585 non-null  float64
 40  family_stdev                  26585 non-null  float64
 41  family_sample_weight          26585 non-null  float64
```

```
 42  family_samples               26585 non-null  float64
 43  hc_mortgage_mean             26585 non-null  float64
 44  hc_mortgage_median           26585 non-null  float64
 45  hc_mortgage_stdev            26585 non-null  float64
 46  hc_mortgage_sample_weight    26585 non-null  float64
 47  hc_mortgage_samples          26585 non-null  float64
 48  hc_mean                      26585 non-null  float64
 49  hc_median                    26585 non-null  float64
 50  hc_stdev                     26585 non-null  float64
 51  hc_samples                   26585 non-null  float64
 52  hc_sample_weight             26585 non-null  float64
 53  home_equity_second_mortgage  26585 non-null  float64
 54  second_mortgage              26585 non-null  float64
 55  home_equity                  26585 non-null  float64
 56  debt                         26585 non-null  float64
 57  second_mortgage_cdf          26585 non-null  float64
 58  home_equity_cdf              26585 non-null  float64
 59  debt_cdf                     26585 non-null  float64
 60  hs_degree                    26585 non-null  float64
 61  hs_degree_male               26585 non-null  float64
 62  hs_degree_female             26585 non-null  float64
 63  male_age_mean                26585 non-null  float64
 64  male_age_median              26585 non-null  float64
 65  male_age_stdev               26585 non-null  float64
 66  male_age_sample_weight       26585 non-null  float64
 67  male_age_samples             26585 non-null  float64
 68  female_age_mean              26585 non-null  float64
 69  female_age_median            26585 non-null  float64
 70  female_age_stdev             26585 non-null  float64
 71  female_age_sample_weight     26585 non-null  float64
 72  female_age_samples           26585 non-null  float64
 73  pct_own                      26585 non-null  float64
 74  married                      26585 non-null  float64
 75  married_snp                  26585 non-null  float64
 76  separated                    26585 non-null  float64
 77  divorced                     26585 non-null  float64
dtypes: category(7), float64(59), int64(6), object(6)
memory usage: 19.8+ MB
None
-----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11355 entries, 0 to 11354
Data columns (total 78 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   UID                          11355 non-null  category
 1   COUNTYID                     11355 non-null  category
 2   STATEID                      11355 non-null  category
 3   state                        11355 non-null  object
 4   state_ab                     11355 non-null  object
 5   city                         11355 non-null  object
 6   place                        11355 non-null  object
 7   type                         11355 non-null  object
 8   primary                      11355 non-null  object
 9   zip_code                     11355 non-null  category
 10  area_code                    11355 non-null  category
 11  lat                          11355 non-null  category
 12  lng                          11355 non-null  category
 13  ALand                        11355 non-null  int64
 14  AWater                       11355 non-null  int64
 15  pop                          11355 non-null  int64
 16  male_pop                     11355 non-null  int64
 17  female_pop                   11355 non-null  int64
 18  rent_mean                    11355 non-null  float64
 19  rent_median                  11355 non-null  float64
 20  rent_stdev                   11355 non-null  float64
 21  rent_sample_weight           11355 non-null  float64
 22  rent_samples                 11355 non-null  float64
 23  rent_gt_10                   11355 non-null  float64
```

```
 24  rent_gt_15                   11355 non-null  float64
 25  rent_gt_20                   11355 non-null  float64
 26  rent_gt_25                   11355 non-null  float64
 27  rent_gt_30                   11355 non-null  float64
 28  rent_gt_35                   11355 non-null  float64
 29  rent_gt_40                   11355 non-null  float64
 30  rent_gt_50                   11355 non-null  float64
 31  universe_samples             11355 non-null  int64
 32  used_samples                 11355 non-null  int64
 33  hi_mean                      11355 non-null  float64
 34  hi_median                    11355 non-null  float64
 35  hi_stdev                     11355 non-null  float64
 36  hi_sample_weight             11355 non-null  float64
 37  hi_samples                   11355 non-null  float64
 38  family_mean                  11355 non-null  float64
 39  family_median                11355 non-null  float64
 40  family_stdev                 11355 non-null  float64
 41  family_sample_weight         11355 non-null  float64
 42  family_samples               11355 non-null  float64
 43  hc_mortgage_mean             11355 non-null  float64
 44  hc_mortgage_median           11355 non-null  float64
 45  hc_mortgage_stdev            11355 non-null  float64
 46  hc_mortgage_sample_weight    11355 non-null  float64
 47  hc_mortgage_samples          11355 non-null  float64
 48  hc_mean                      11355 non-null  float64
 49  hc_median                    11355 non-null  float64
 50  hc_stdev                     11355 non-null  float64
 51  hc_samples                   11355 non-null  float64
 52  hc_sample_weight             11355 non-null  float64
 53  home_equity_second_mortgage  11355 non-null  float64
 54  second_mortgage              11355 non-null  float64
 55  home_equity                  11355 non-null  float64
 56  debt                         11355 non-null  float64
 57  second_mortgage_cdf          11355 non-null  float64
 58  home_equity_cdf              11355 non-null  float64
 59  debt_cdf                     11355 non-null  float64
 60  hs_degree                    11355 non-null  float64
 61  hs_degree_male               11355 non-null  float64
 62  hs_degree_female             11355 non-null  float64
 63  male_age_mean                11355 non-null  float64
 64  male_age_median              11355 non-null  float64
 65  male_age_stdev               11355 non-null  float64
 66  male_age_sample_weight       11355 non-null  float64
 67  male_age_samples             11355 non-null  float64
 68  female_age_mean              11355 non-null  float64
 69  female_age_median            11355 non-null  float64
 70  female_age_stdev             11355 non-null  float64
 71  female_age_sample_weight     11355 non-null  float64
 72  female_age_samples           11355 non-null  float64
 73  pct_own                      11355 non-null  float64
 74  married                      11355 non-null  float64
 75  married_snp                  11355 non-null  float64
 76  separated                    11355 non-null  float64
 77  divorced                     11355 non-null  float64
dtypes: category(7), float64(58), int64(7), object(6)
memory usage: 7.9+ MB
None
```

In [163]:

```
train_df[cat_variables(train_df)]
```

Out[163]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.70

2	245683	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396
4	247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195
...	...	...	...	...	...	...	...	...	...	...	...	
26580	279212	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	tract	769	787	18.076
26581	277856	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	tract	19422	215	40.158
26582	233000	87	8	Colorado	CO	Weldona	Saddle Ridge	City	tract	80653	970	40.410
26583	287425	439	48	Texas	TX	Colleyville	Colleyville City	Town	tract	76034	817	32.904
26584	265371	3	32	Nevada	NV	Las Vegas	Paradise	City	tract	89123	702	36.064

26585 rows × 19 columns

In [164]:

```
obj_2_cat = ['state', 'state_ab', 'city', 'place', 'type', 'primary']
```

In [165]:

```
for col in obj_2_cat:
    train_df[col] = train_df[col].astype('category')
    test_df[col] = test_df[col].astype('category')
```

In [166]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26585 entries, 0 to 26584
Data columns (total 78 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   UID                          26585 non-null  category
 1   COUNTYID                     26585 non-null  category
 2   STATEID                      26585 non-null  category
 3   state                        26585 non-null  category
 4   state_ab                     26585 non-null  category
 5   city                         26585 non-null  category
 6   place                        26585 non-null  category
 7   type                         26585 non-null  category
 8   primary                      26585 non-null  category
 9   zip_code                     26585 non-null  category
 10  area_code                    26585 non-null  category
 11  lat                          26585 non-null  category
 12  lng                          26585 non-null  category
 13  ALand                        26585 non-null  float64
 14  AWater                       26585 non-null  int64
 15  pop                          26585 non-null  int64
 16  male_pop                     26585 non-null  int64
 17  female_pop                   26585 non-null  int64
 18  rent_mean                    26585 non-null  float64
 19  rent_median                  26585 non-null  float64
 20  rent_stdev                   26585 non-null  float64
 21  rent_sample_weight           26585 non-null  float64
 22  rent_samples                 26585 non-null  float64
 23  rent_gt_10                   26585 non-null  float64
 24  rent_gt_15                   26585 non-null  float64
 25  rent_gt_20                   26585 non-null  float64
```

```
 26  rent_gt_25                    26585 non-null  float64
 27  rent_gt_30                    26585 non-null  float64
 28  rent_gt_35                    26585 non-null  float64
 29  rent_gt_40                    26585 non-null  float64
 30  rent_gt_50                    26585 non-null  float64
 31  universe_samples              26585 non-null  int64
 32  used_samples                  26585 non-null  int64
 33  hi_mean                       26585 non-null  float64
 34  hi_median                     26585 non-null  float64
 35  hi_stdev                      26585 non-null  float64
 36  hi_sample_weight              26585 non-null  float64
 37  hi_samples                    26585 non-null  float64
 38  family_mean                   26585 non-null  float64
 39  family_median                 26585 non-null  float64
 40  family_stdev                  26585 non-null  float64
 41  family_sample_weight          26585 non-null  float64
 42  family_samples                26585 non-null  float64
 43  hc_mortgage_mean              26585 non-null  float64
 44  hc_mortgage_median            26585 non-null  float64
 45  hc_mortgage_stdev             26585 non-null  float64
 46  hc_mortgage_sample_weight     26585 non-null  float64
 47  hc_mortgage_samples           26585 non-null  float64
 48  hc_mean                       26585 non-null  float64
 49  hc_median                     26585 non-null  float64
 50  hc_stdev                      26585 non-null  float64
 51  hc_samples                    26585 non-null  float64
 52  hc_sample_weight              26585 non-null  float64
 53  home_equity_second_mortgage   26585 non-null  float64
 54  second_mortgage               26585 non-null  float64
 55  home_equity                   26585 non-null  float64
 56  debt                          26585 non-null  float64
 57  second_mortgage_cdf           26585 non-null  float64
 58  home_equity_cdf               26585 non-null  float64
 59  debt_cdf                      26585 non-null  float64
 60  hs_degree                     26585 non-null  float64
 61  hs_degree_male                26585 non-null  float64
 62  hs_degree_female              26585 non-null  float64
 63  male_age_mean                 26585 non-null  float64
 64  male_age_median               26585 non-null  float64
 65  male_age_stdev                26585 non-null  float64
 66  male_age_sample_weight        26585 non-null  float64
 67  male_age_samples              26585 non-null  float64
 68  female_age_mean               26585 non-null  float64
 69  female_age_median             26585 non-null  float64
 70  female_age_stdev              26585 non-null  float64
 71  female_age_sample_weight      26585 non-null  float64
 72  female_age_samples            26585 non-null  float64
 73  pct_own                       26585 non-null  float64
 74  married                       26585 non-null  float64
 75  married_snp                   26585 non-null  float64
 76  separated                     26585 non-null  float64
 77  divorced                      26585 non-null  float64
dtypes: category(13), float64(59), int64(6)
memory usage: 19.6 MB
```

In [167]:

```
train_df[['hc_mortgage_mean']]
```

Out[167]:

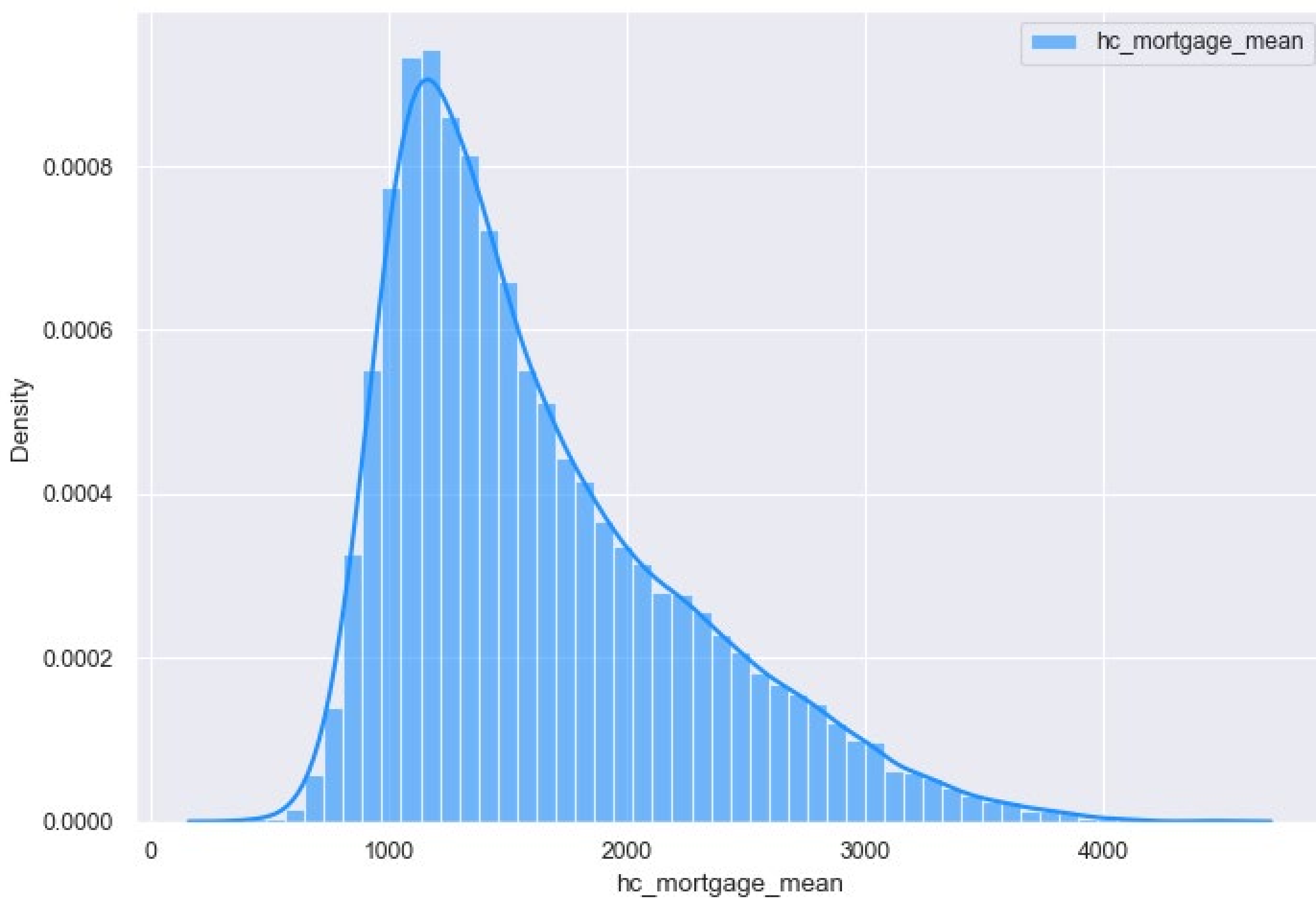
	hc_mortgage_mean
0	1414.80295
1	864.41390
2	1506.06758
3	1175.28642

4	1192.58759
...	...
26580	770.11560
26581	2210.84055
26582	1671.07908
26583	3074.83088
26584	1455.42340

26585 rows × 1 columns

In [168]:

```
# Plot
kwargs = dict(hist_kws={'alpha':.6}, kde_kws={'linewidth':2})

plt.figure(figsize=(10,7), dpi= 80)
sns.distplot(train_df.hc_mortgage_mean, color="dodgerblue", label="hc_mortgage_mean",
**kwargs)
# sns.distplot(x2, color="orange", label="SUV", **kwargs)
# sns.distplot(x3, color="deeppink", label="minivan", **kwargs)
# plt.xlim(50,75)
plt.legend();
```



Target Variable "hc_mortgage_mean" has a Positive Skew.

In [169]:

```
from sklearn.linear_model import LinearRegression
```

In [170]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, SCORERS
```

In [171]:

```
lr = LinearRegression()
```

Adj r2 = 1-(1-R2)*(n-1)/(n-p-1)

In [172]:

```
def adj_rsqrd(df, r2):
        # adjusted r2 using formula adj_r2 = 1 - (1- r2) * (n-1) / (n - k - 1)
    # k = number of predictors = data.shape[1] - 1
    adj_rsqrd = 1 - (1-r2)*(len(df) - 1) / (len(df) - (df.shape[1] - 1) - 1)
    return round(adj_rsqrd, 3)
```

In [173]:

```
cat_cols_2_drop = ['UID', 'state', 'state_ab', 'city', 'place', 'type', 'primary',
'zip_code', 'area_code', 'lat', 'lng']
```

In [174]:

```
train_df.drop(cat_cols_2_drop, axis=1, inplace=True)
```

In [175]:

```
test_df.drop(cat_cols_2_drop, axis=1, inplace=True)
```

In [176]:

```
train_df.drop(['COUNTYID', 'STATEID'], axis=1, inplace=True)
```

In [177]:

```
test_y = test_df['hc_mortgage_mean']
```

In [178]:

```
test_df.drop(['COUNTYID', 'STATEID', 'hc_mortgage_mean'], axis=1, inplace=True)
```

In [179]:

```
print(train_df.shape, test_df.shape)
```

```
(26585, 65) (11355, 64)
```

In [180]:

```
train_X = train_df.drop(columns=['hc_mortgage_mean'])
train_y = train_df['hc_mortgage_mean']
```

In [181]:

```
lr.fit(train_X, train_y)
```

Out[181]:

```
LinearRegression()
```

In [182]:

```
predict_train = lr.predict(train_X)
predict_test = lr.predict(test_df)
```

In [183]:

```
# model evaluation for testing set
```

```
mae = mean_absolute_error(test_y, predict_test)
mse = mean_squared_error(test_y, predict_test)
r2 = r2_score(test_y, predict_test)

print("The model performance for test set")
print("--------------------------------------")
print('MAE is {}'.format(round(mae, 3)))
print('MSE is {}'.format(round(mse, 3)))
print('RMSE is {}'.format(round(mse**(0.5), 3)))
print('R2 score is {}'.format(round(r2, 3)))

print('Adjusted R2 score is {}'.format(adj_rsqrd(test_df, r2)))
```

```
The model performance for test set
--------------------------------------
MAE is 43.675
MSE is 4673.486
RMSE is 68.363
R2 score is 0.988
Adjusted R2 score is 0.988
```

Regression Model with all dependent numeric variables @ Country level is giving R SQUARED metric of 98.8%. So skipping state level Regression Model

In [184]:

```
correlated_features = set()
correlation_matrix = train_df.drop('hc_mortgage_mean', axis=1).corr()

for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.8:
            colname = correlation_matrix.columns[i]
            correlated_features.add(colname)
```

In [185]:

```
correlated_features
```

Out[185]:

```
{'debt_cdf',
 'family_mean',
 'family_median',
 'family_sample_weight',
 'family_samples',
 'family_stdev',
 'female_age_mean',
 'female_age_median',
 'female_age_sample_weight',
 'female_age_samples',
 'female_pop',
 'hc_median',
 'hc_mortgage_samples',
 'hc_sample_weight',
 'hi_median',
 'hi_samples',
```

```
 'hi_stdev',
 'home_equity_cdf',
 'hs_degree_female',
 'hs_degree_male',
 'male_age_median',
 'male_age_sample_weight',
 'male_age_samples',
 'male_pop',
 'rent_gt_25',
 'rent_gt_30',
 'rent_gt_35',
 'rent_gt_40',
 'rent_gt_50',
 'rent_median',
 'rent_samples',
 'second_mortgage',
 'universe_samples',
 'used_samples'}
```

In [186]:

```
corr_list = ['debt_cdf', 'family_mean', 'family_median', 'family_sample_weight',
'family_samples', 'family_stdev', 'female_age_mean', 'female_age_median',
                     'female_age_sample_weight', 'female_age_samples', 'female_pop',
'hc_median', 'hc_mortgage_samples', 'hc_sample_weight', 'hi_median',
                     'hi_samples', 'hi_stdev', 'home_equity_cdf', 'hs_degree_female',
'hs_degree_male', 'male_age_median', 'male_age_sample_weight',
                 'male_age_samples', 'male_pop', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
'rent_gt_40', 'rent_gt_50', 'rent_median', 'rent_samples', 'second_mortgage',
'universe_samples', 'used_samples']
```

In [187]:

```
train_df.drop(corr_list, axis=1, inplace=True)
```

In [188]:

```
test_df.drop(corr_list, axis=1, inplace=True)
```

In [189]:

```
print(train_df.shape, test_df.shape)
```

```
(26585, 31) (11355, 30)
```

Dropped MultiCollinear variables and ran the Regression Model.

In [190]:

```
train_df.head()
```

Out[190]:

	ALand	AWater	pop	rent_mean	rent_stdev	rent_sample_weight	rent_gt_10	rent_gt_15	rent_gt_20	hi_mean	hi_samp
0	202183361.0	1699120	5230	769.38638	232.63967	272.34441	0.86761	0.79155	0.59155	63125.28406	1
1	1560828.0	100363	2633	804.87924	253.46747	312.58622	0.97410	0.93227	0.69920	41931.92593	
2	69561595.0	284193	6881	742.77365	323.39011	291.85520	0.95238	0.88624	0.79630	84942.68317	
3	1105793.0	0	2700	803.42018	297.39258	259.30316	0.94693	0.87151	0.69832	48733.67116	
4	2554403.0	0	5637	938.56493	392.44096	1005.42886	0.99286	0.98247	0.91688	31834.15466	

In [191]:

```
train_X = train_df.drop(columns=['hc_mortgage_mean'])
train_y = train_df['hc_mortgage_mean']
```

In [192]:

```
lr.fit(train_X, train_y)
```

Out[192]:

```
LinearRegression()
```

In [193]:

```
predict_train = lr.predict(train_X)
predict_test = lr.predict(test_df)
```

In [194]:

```
# model evaluation for testing set

mae = mean_absolute_error(test_y, predict_test)
mse = mean_squared_error(test_y, predict_test)
r2 = r2_score(test_y, predict_test)

print("The model performance for test set")
print("--------------------------------------")
print('MAE is {}'.format(round(mae, 3)))
print('MSE is {}'.format(round(mse, 3)))
print('RMSE is {}'.format(round(mse**(0.5), 3)))
print('R2 score is {}'.format(round(r2, 3)))

print('Adjusted R2 score is {}'.format(adj_rsqrd(test_df, r2)))
```

```
The model performance for test set
--------------------------------------
MAE is 44.001
MSE is 4787.228
RMSE is 69.19
R2 score is 0.988
Adjusted R2 score is 0.988
```

In [195]:

```
sorted(SCORERS.keys())
```

Out[195]:

```
['accuracy',
 'adjusted_mutual_info_score',
 'adjusted_rand_score',
 'average_precision',
 'balanced_accuracy',
 'completeness_score',
 'explained_variance',
 'f1',
 'f1_macro',
 'f1_micro',
 'f1_samples',
 'f1_weighted',
 'fowlkes_mallows_score',
 'homogeneity_score',
 'jaccard',
 'jaccard_macro',
 'jaccard_micro',
 'jaccard_samples',
 'jaccard_weighted',
 'max_error',
 'mutual_info_score',
 'neg_brier_score',
 'neg_log_loss',
```

```
 'neg_mean_absolute_error',
 'neg_mean_gamma_deviance',
 'neg_mean_poisson_deviance',
 'neg_mean_squared_error',
 'neg_mean_squared_log_error',
 'neg_median_absolute_error',
 'neg_root_mean_squared_error',
 'normalized_mutual_info_score',
 'precision',
 'precision_macro',
 'precision_micro',
 'precision_samples',
 'precision_weighted',
 'r2',
 'recall',
 'recall_macro',
 'recall_micro',
 'recall_samples',
 'recall_weighted',
 'roc_auc',
 'roc_auc_ovo',
 'roc_auc_ovo_weighted',
 'roc_auc_ovr',
 'roc_auc_ovr_weighted',
 'v_measure_score']
```

Let's Check how close our algorithm is predicting, by passing the inputs from our test set and compare them to the target values.

In [196]:

```
import random
randomlist = []
for i in range(0,100):
    n = random.randint(1,len(test_df))
    randomlist.append(n)
print(randomlist)
```

```
[5017, 11339, 1838, 10169, 2484, 6124, 8393, 6118, 8550, 1959, 620, 2689, 10028, 6120, 5706,
 996, 8517, 7407, 3330, 2886, 4942, 581, 10109, 8907, 4576, 10204, 10091, 204, 881, 9369, 11
034, 6443, 6263, 7682, 5958, 8756, 9097, 10095, 6567, 3048, 2593, 7036, 3693, 6401, 3900, 24
61, 5752, 2203, 6093, 2546, 6987, 9917, 105, 8258, 9934, 7572, 6221, 1237, 1659, 4223, 1290,
 8914, 2464, 9241, 8583, 5269, 1472, 2653, 7613, 5829, 322, 6314, 9125, 10258, 6619, 4295, 1
0807, 5834, 8408, 7668, 2617, 1420, 1605, 9286, 9977, 8338, 8943, 8243, 6669, 9479, 6538, 57
5, 7467, 4147, 7795, 4895, 7999, 4464, 177, 1510]
```

In [197]:

```
pre_out = []
out = []

for i in randomlist:
    data_in = [list(test_df.iloc[i])]
    pre_data_out = lr.predict(data_in)
    data_out = test_y .iloc[i]

    print(i, pre_data_out, data_out)

    pre_out.append(pre_data_out)
    out.append(data_out)
```

```
5017 [980.98840874] 1020.71136
11339 [2011.05709176] 1944.7597399999997
1838 [1513.01157044] 1422.99748
10169 [1503.40027775] 1515.92725
2484 [1043.69550703] 1060.44108
6124 [1125.01101387] 1184.84169
8393 [1050.86060956] 1076.9534800000001
6118 [1463.88380511] 1509.65504
8550 [948.86439967] 936.08394
1959 [1023.01320305] 1061.20854
620 [1782.09191058] 1797.0207
2689 [975.20114401] 972.21377
10028 [2674.40093178] 2691.21003
6120 [1198.73905501] 1155.4516199999998
5706 [1421.80680715] 1437.37567
996 [2356.73740576] 2310.54427
8517 [906.14949766] 924.10999
7407 [850.12201694] 849.5
3330 [1214.04574675] 1188.21001
2886 [1047.01167506] 1201.88341
4942 [1762.31028658] 1768.26761
581 [1163.32610553] 1175.23317
10109 [1013.86570337] 968.3770199999999
8907 [1036.56970692] 1046.06256
4576 [1160.951397] 1158.09756
10204 [2571.74372218] 2681.4611
10091 [1246.66363903] 1234.07312
204 [1778.82426126] 1786.19902
881 [1383.01799477] 1374.25792
9369 [3089.08901299] 3046.55486
11034 [1265.6220323] 1341.66401
6443 [1231.2480032] 1232.50782
6263 [1441.44304353] 1397.45742
7682 [1244.69722159] 1221.61521
5958 [2373.17740923] 2418.71823
8756 [1097.25465308] 1113.59999
9097 [3694.84270759] 3779.24825
10095 [1623.34850522] 1643.7051199999999
6567 [1618.42087368] 1558.1212699999999
3048 [1988.93276008] 2138.74684
2593 [1762.38446276] 1751.8533
7036 [1798.39253318] 1749.76635
3693 [1685.17410914] 1786.2229300000001
6401 [1147.92155466] 1125.8965
3900 [1523.17076763] 1423.37861
2461 [1850.96938825] 1866.10475
5752 [3140.96039438] 3225.32585
2203 [2182.61980053] 2464.9775600000003
6093 [2571.77711889] 2620.05712
2546 [887.46787702] 856.14587
6987 [2228.11601474] 2329.49476
9917 [1217.56156348] 1249.78824
105 [1599.81869881] 1688.66997
8258 [1668.44266453] 1729.5769
9934 [1971.55968422] 1911.39198
7572 [2681.70063186] 2504.93195
6221 [923.5774016] 909.26085
1237 [1131.83210306] 1069.15464
1659 [1119.42675633] 1128.76383
4223 [1411.71552387] 1390.2223
1290 [3557.33881532] 3591.3009399999996
8914 [1943.6713287] 1933.62743
2464 [2669.51218228] 2630.71249
9241 [1340.52883717] 1303.85277
8583 [1034.64039176] 1053.44012
5269 [3920.33754183] 3803.7822100000003
1472 [1347.22922625] 1334.8972099999999
2653 [1075.58590919] 1075.5807300000001
7613 [1776.14069517] 1736.6608800000001
```

```
5829 [1844.3184397] 1860.9503
322 [1250.45096227] 1301.00008
6314 [1692.64578314] 1659.34076
9125 [1815.02240931] 1839.38535
10258 [1380.3770052] 1438.4952
6619 [2720.78898457] 2895.2521899999997
4295 [1505.66469606] 1460.28404
10807 [993.77221757] 970.6320699999999
5834 [1783.00250909] 1725.04117
8408 [1158.90178204] 1209.91761
7668 [1717.36476011] 1784.81772
2617 [1829.58280703] 1833.3834399999998
1420 [1472.14942893] 1472.05407
1605 [912.81300553] 891.1456400000001
9286 [1029.46782928] 978.80337
9977 [950.90597378] 954.0540199999999
8338 [1650.55827343] 1775.7124199999998
8943 [1457.26319166] 1482.9299800000001
8243 [1107.85906521] 1145.9904199999999
6669 [2717.25495993] 2804.97875
9479 [1158.48035192] 1213.6245
6538 [1534.71616142] 1495.7499
575 [1635.5267134] 1644.8542300000001
7467 [1269.85854882] 1298.6027900000001
4147 [2303.14517593] 2253.07306
7795 [2037.42887731] 2031.97495
4895 [1202.321266] 1046.6097
7999 [1391.39987093] 1388.5344699999998
4464 [980.01645256] 984.28444
177 [1620.0276968] 1543.4494
1510 [974.56582613] 974.1840800000001
```

In [198]:

```
pre_out
```

Out[198]:

```
[array([980.98840874]),
 array([2011.05709176]),
 array([1513.01157044]),
 array([1503.40027775]),
 array([1043.69550703]),
 array([1125.01101387]),
 array([1050.86060956]),
 array([1463.88380511]),
 array([948.86439967]),
 array([1023.01320305]),
 array([1782.09191058]),
 array([975.20114401]),
 array([2674.40093178]),
 array([1198.73905501]),
 array([1421.80680715]),
 array([2356.73740576]),
 array([906.14949766]),
 array([850.12201694]),
 array([1214.04574675]),
 array([1047.01167506]),
 array([1762.31028658]),
 array([1163.32610553]),
 array([1013.86570337]),
 array([1036.56970692]),
 array([1160.951397]),
 array([2571.74372218]),
 array([1246.66363903]),
 array([1778.82426126]),
 array([1383.01799477]),
 array([3089.08901299]),
 array([1265.6220323]),
 array([1231.2480032]),
 array([1441.44304353]),
```

```
array([1244.69722159]),
array([2373.17740923]),
array([1097.25465308]),
array([3694.84270759]),
array([1623.34850522]),
array([1618.42087368]),
array([1988.93276008]),
array([1762.38446276]),
array([1798.39253318]),
array([1685.17410914]),
array([1147.92155466]),
array([1523.17076763]),
array([1850.96938825]),
array([3140.96039438]),
array([2182.61980053]),
array([2571.77711889]),
array([887.46787702]),
array([2228.11601474]),
array([1217.56156348]),
array([1599.81869881]),
array([1668.44266453]),
array([1971.55968422]),
array([2681.70063186]),
array([923.5774016]),
array([1131.83210306]),
array([1119.42675633]),
array([1411.71552387]),
array([3557.33881532]),
array([1943.6713287]),
array([2669.51218228]),
array([1340.52883717]),
array([1034.64039176]),
array([3920.33754183]),
array([1347.22922625]),
array([1075.58590919]),
array([1776.14069517]),
array([1844.3184397]),
array([1250.45096227]),
array([1692.64578314]),
array([1815.02240931]),
array([1380.3770052]),
array([2720.78898457]),
array([1505.66469606]),
array([993.77221757]),
array([1783.00250909]),
array([1158.90178204]),
array([1717.36476011]),
array([1829.58280703]),
array([1472.14942893]),
array([912.81300553]),
array([1029.46782928]),
array([950.90597378]),
array([1650.55827343]),
array([1457.26319166]),
array([1107.85906521]),
array([2717.25495993]),
array([1158.48035192]),
array([1534.71616142]),
array([1635.5267134]),
array([1269.85854882]),
array([2303.14517593]),
array([2037.42887731]),
array([1202.321266]),
array([1391.39987093]),
array([980.01645256]),
array([1620.0276968]),
array([974.56582613])]
```

In [199]:

```
x = [2,3,5,9,1,0,2,3]

def my_min(sequence):
    """return the minimum element of sequence"""
    low = sequence[0] # need to start with some value
    for i in sequence:
        if i < low:
            low = i
    return low

print(my_min(x))
```

0

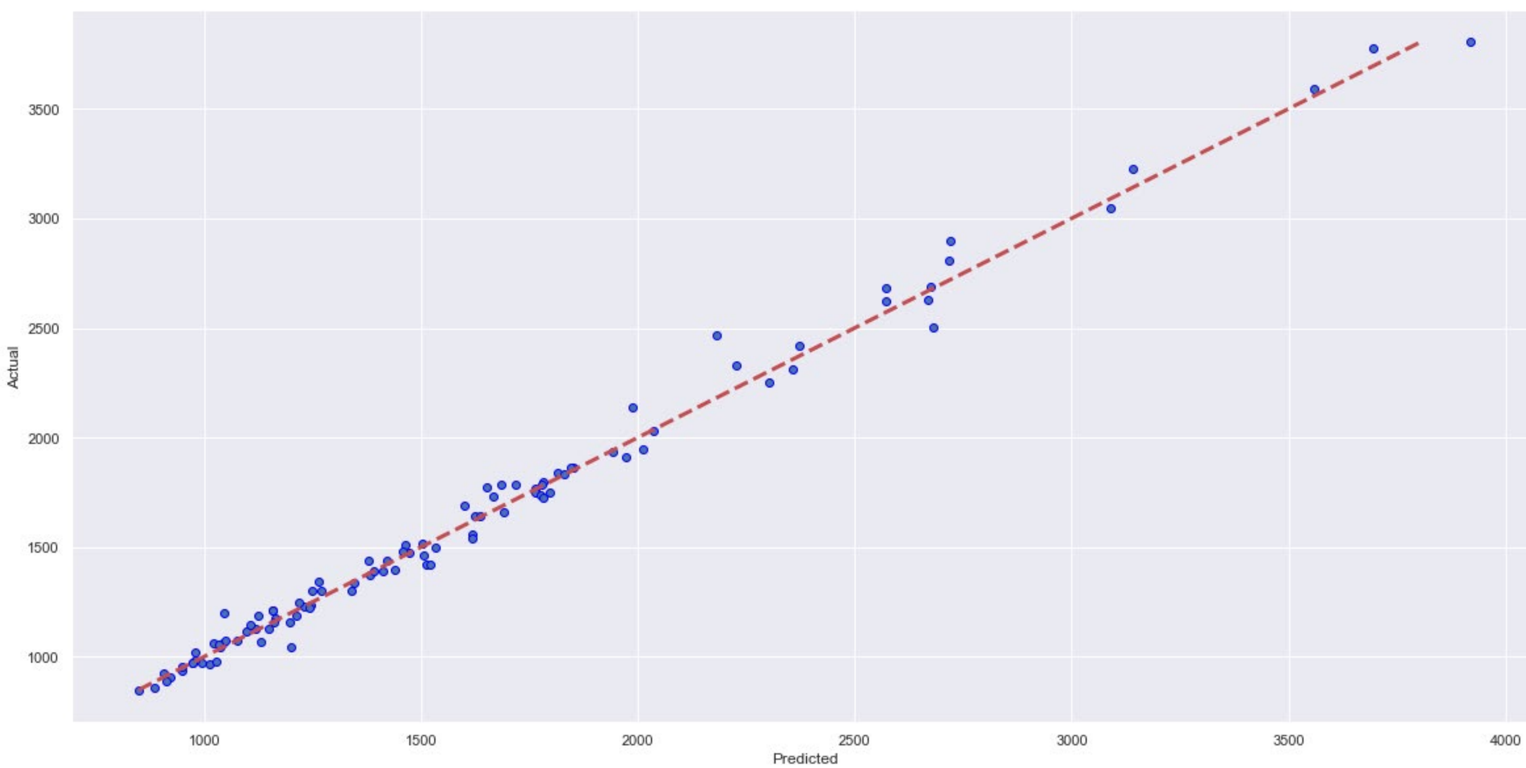
In [200]:

```
x = [2,3,5,9,1,0,2,3]

def my_maxi(sequence):
    """return the minimum element of sequence"""
    maxi = sequence[0] # need to start with some value
    for i in sequence:
        if i > maxi:
            maxi = i
    return maxi

print(my_maxi(x))
```

9

In [201]:

```
fig, ax = plt.subplots(figsize=(20,10))
ax.scatter(pre_out, out, edgecolors=(0, 0, 1))
ax.plot([my_min(out), my_maxi(out)], [my_min(out), my_maxi(out)], 'r--', lw=3)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
plt.show()
```



In [202]:

```
# model evaluation for testing set

mae = mean_absolute_error(test_y, predict_test)
mse = mean_squared_error(test_y, predict_test)
r2 = r2_score(test_y, predict_test)

print("The model performance for test set")
print("--------------------------------------")
print('MAE is {}'.format(round(mae, 3)))
print('MSE is {}'.format(round(mse, 3)))
print('RMSE is {}'.format(round(mse**(0.5), 3)))
print('R2 score is {}'.format(round(r2, 3)))

print('Adjusted R2 score is {}'.format(adj_rsqrd(test_df, r2)))
```

```
The model performance for test set
--------------------------------------
MAE is 44.001
MSE is 4787.228
RMSE is 69.19
R2 score is 0.988
Adjusted R2 score is 0.988
```

We have achieved an adjusted R Squared value of 98.8% which is pretty close to 1, indicating our selected "Independent Variables" are highly correlated to our "Dependent Variable" and our model is able to predict very accurately.

By : Abdullah

Alwabel