CAPSTONE PROJECT1: Real Estate

Problem Statement:

1)A banking institution requires actionable insights into mortgage-backed securities, geographic business investment, and real estate analysis.

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

3)A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. 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')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from itertools import cycle
```

In [3]:
```
train.head()
```

Out[3]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_me
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.3
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.5
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.8
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.5
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.5

5 rows × 80 columns

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

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 [4]:
```
train.columns
```

Out[4]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state',
       'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_code',
       'lat', 'lng', 'ALand', 'AWater', 'pop', 'male_pop', 'female_pop',
       'rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight',
       'rent_samples', 'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25',
       'rent_gt_30', 'rent_gt_35', 'rent_gt_40', 'rent_gt_50',
       'universe_samples', 'used_samples', 'hi_mean', 'hi_median', 'hi_stdev',
       'hi_sample_weight', 'hi_samples', 'family_mean', 'family_median',
       'family_stdev', 'family_sample_weight', 'family_samples',
       'hc_mortgage_mean', 'hc_mortgage_median', 'hc_mortgage_stdev',
       'hc_mortgage_sample_weight', 'hc_mortgage_samples', 'hc_mean',
       'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced'],
      dtype='object')
```

In [5]:
```
train.dtypes
```

Out[5]:
```
UID              int64
BLOCKID        float64
SUMLEVEL         int64
COUNTYID         int64
STATEID          int64
                 ...
pct_own        float64
married        float64
married_snp    float64
separated      float64
divorced       float64
Length: 80, dtype: object
```

In [6]:
```
train.columns[:5]
```

Out[6]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', 'COUNTYID', 'STATEID'], dtype='object')
```

In [7]:
```
for i in range(0, len(np.array_split(train.dtypes, 5))):
    print((np.array_split(train.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 [8]:
```
train[train.columns[0:20]].head()
```

Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.84
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.70
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.79
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.39
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.19

In [9]:
```
for i in range(0, len(train.columns), 20):
    print(train[train.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 [10]:
```
cat_columns = ['UID', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type', 'primary', 'zip_code
```

In [11]:
```
train[cat_columns].dtypes
```

Out[11]:
```
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 [12]:
```
for col in cat_columns:
    print(col)
    print(train[col].nunique())
    print(train[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 [13]:
```
train.isnull().sum(axis = 0)
```

Out[13]:
```
UID               0
BLOCKID       27321
SUMLEVEL          0
COUNTYID          0
STATEID           0
              ...
pct_own         268
married         191
married_snp     191
separated       191
divorced        191
Length: 80, dtype: int64
```

In [14]:
```
train.isnull().sum(axis = 0)[20:30]
```

Out[14]:
```
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 [15]:
```
train.shape
```

Out[15]:
```
(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 [16]:
```
len(train.columns[train.isnull().sum(axis = 0) > 0])
```

Out[16]:
```
59
```

In [17]:
```
train.drop(['BLOCKID', 'primary'], axis=1, inplace=True)
```

In [18]:
```
null_data = train[train.isnull().any(axis=1)]
null_data
```

Out[18]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean	fem
51	223593	140	19	4	Arizona	AZ	Tucson	Littletown	CDP	85734	...	40.02370	
94	233040	140	101	8	Colorado	CO	Pueblo	Pueblo City	City	81001	...	20.00784	
153	263292	140	13	34	New Jersey	NJ	Newark	Silver Lake	City	7107	...	35.47667	
302	267158	140	47	36	New York	NY	Brooklyn	New York City	City	11215	...	NaN	
340	292484	140	25	55	Wisconsin	WI	Madison	Madison City	City	53703	...	22.03226	
...	...	...	...	...	...	...	...	...	...	...	...	...	
27127	266321	140	5	36	New York	NY	Bronx	Mount Vernon City	City	10458	...	37.43584	
27175	235725	140	57	12	Florida	FL	Tampa	Pebble Creek	City	33647	...	29.08800	
27176	247777	140	61	21	Kentucky	KY	Brownsville	Brownsville City	City	42210	...	19.39847	
27216	266166	140	5	36	New York	NY	Bronx	Pelham Manor	City	10462	...	37.70543	
27240	251078	140	25	25	Massachusetts	MA	Boston	Brookline	City	2124	...	38.36136	

736 rows × 78 columns

In [19]:
```
round((736 / 27321)*100, 2)
```

Out[19]:
```
2.69
```

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

In [20]:
```
train.shape
```

Out[20]:
```
(27321, 78)
```

In [21]:
```
train = pd.concat([train, null_data, null_data]).drop_duplicates(keep=False)
```

In [22]:
```
train.shape
```

Out[22]:
```
(26585, 78)
```

In [23]:
```
len(train.columns[train.isnull().sum(axis = 0) > 0])
```

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

In [24]:
```
cat_columns = ['UID', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type', 'zip_code', 'area_co
```

In [25]:
```
for col in cat_columns:
    train[col] = train[col].astype('category')
```

In [26]:
```
train.dtypes
```

Out[26]:
```
UID            category
SUMLEVEL          int64
COUNTYID       category
STATEID        category
state          category
                  ...
pct_own         float64
married         float64
married_snp     float64
separated       float64
divorced        float64
Length: 78, 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 [27]:
```
train.nlargest(2500, ['second_mortgage', 'pct_own'])
```

Out[27]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean
14014	264403	140	31	34	New Jersey	NJ	Passaic	Garfield City	City	7055	...	26.57222
3285	289712	140	147	51	Virginia	VA	Farmville	Farmville	Town	23901	...	19.58762
21706	222830	140	13	4	Arizona	AZ	Scottsdale	Tempe City	CDP	85257	...	31.91429
11980	251185	140	27	25	Massachusetts	MA	Worcester	Worcester City	City	1610	...	30.60147
12896	278178	140	101	42	Pennsylvania	PA	Philadelphia	Millbourne	Borough	19104	...	22.42708
...	...	...	...	...	...	...	...	...	...	...	...	...
9223	245335	140	3	18	Indiana	IN	Fort Wayne	Fort Wayne City	City	46814	...	36.26617
24579	260417	140	81	37	North Carolina	NC	High Point	Jamestown	Village	27265	...	37.43896
19475	286364	140	257	48	Texas	TX	Crandall	Talty	Town	75114	...	32.84647
13270	287041	140	397	48	Texas	TX	Royse City	Fate City	Town	75189	...	32.19408
22594	225435	140	37	6	California	CA	Los Angeles	South Pasadena City	City	90042	...	32.26462

2500 rows × 78 columns

In [28]:
```
top_2500 = train[['state', 'lat', 'lng', 'second_mortgage', 'pct_own', 'place', 'state', 'city', 'COUNTYID', 'S
top_2500
```

Out[28]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEID	hon
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	34	
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	51	
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	25	
12896	Pennsylvania	39.952954	-75.202767	0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	42	
...	...	...	...	...	...	...	...	...	...	...	...
24443	California	37.732143	-121.242902	0.06814	0.67116	Manteca City	California	Manteca	77	6	
8377	Florida	25.550391	-80.347791	0.06813	0.50519	Cutler Bay	Florida	Cutler Bay	86	12	
16621	Texas	32.913822	-97.204310	0.06812	0.97987	Keller City	Texas	Keller	439	48	
13987	Ohio	39.556756	-84.443252	0.06812	0.92888	Jacksonburg	Ohio	Middletown	17	39	
14857	New Jersey	39.432879	-74.686137	0.06810	0.70642	Mays Landing	New Jersey	Mays Landing	1	34	

2563 rows × 15 columns

In [29]:
```
top_2500.pct_own.unique
```

Out[29]:
```
<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 [30]:
```
train[train.pct_own > 0.1]
```

Out[30]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean	fem
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	...	44.48629	
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	...	36.48391	
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	...	42.15810	
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	...	47.77526	
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	...	24.17693	
...	...	...	...	...	...	...	...	...	...	...	...	...	
27316	279212	140	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	769	...	42.73154	
27317	277856	140	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	19422	...	38.21269	
27318	233000	140	87	8	Colorado	CO	Weldona	Saddle Ridge	City	80653	...	43.40218	
27319	287425	140	439	48	Texas	TX	Colleyville	Colleyville City	Town	76034	...	39.25921	
27320	265371	140	3	32	Nevada	NV	Las Vegas	Paradise	City	89123	...	34.45345	

26215 rows × 78 columns

In [31]:
```
top_2500[top_2500.pct_own > 0.1].head()
```

Out[31]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEID	home_eq
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	51	0.00
11980	Massachusetts	42.254262	-71.800347	0.43363	0.20247	Worcester City	Massachusetts	Worcester	27	25	0.43
26018	New York	40.751809	-73.853582	0.31818	0.15618	Harbor Hills	New York	Corona	81	36	0.40
7829	Maryland	39.127273	-76.635265	0.30212	0.22380	Glen Burnie	Maryland	Glen Burnie	3	24	0.35
2077	Florida	28.029063	-82.495395	0.28972	0.11618	Egypt Lake-leto	Florida	Tampa	57	12	0.38

In [32]:
```
import plotly.graph_objects as go
import plotly.figure_factory as ff
```

In [33]:
```
scope = ["USA"]

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

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

In [34]:
```
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
```

In [35]:
```
z_COUNTYID = zero_prefix(top_2500.COUNTYID)
```

```
Enter Target Length of String: 4
```

In [36]:
```
z_STATEID = zero_prefix(top_2500.STATEID)
```

```
Enter Target Length of String: 3
```

In [37]:
```
top_2500['FIPSID'] = [a + b for a, b in zip(z_STATEID,z_COUNTYID)]
```

In [38]:
```
top_2500.head()
```

Out[38]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEID	home_
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	34	(
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	51	(
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	25	(
12896	Pennsylvania	39.952954	-75.202767	0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	42	(

In [39]:
```
top_2500.dtypes
```

Out[39]:
```
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 [40]:
```
train[col] = train[col].astype('category')
```

In [41]:
```
top_2500['FIPSID'] = top_2500['FIPSID'].astype('int64')
```

In [42]:
```
scope = ["USA"]

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

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

In [43]:
```
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 [44]:
```
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 [45]:
```
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=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 = LinearColorMapper(palette=Viridis5)

circle = Circle(x="lon", y="lat", size="size", fill_color={'field': 'color', 'transform': color_mapper}, fill_a
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_notebook()

show(plot)
```

Loading BokehJS ...

b) Use the following bad debt equation:

1)Bad Debt = P (Second Mortgage ∩ Home Equity Loan) 2)Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage

In [46]:
```
top_2500['Bad_Debt'] = top_2500['second_mortgage'] + top_2500['home_equity'] - top_2500['home_equity_second_mor
top_2500['Good_Debt'] = top_2500['debt'] - top_2500['Bad_Debt']
```

In [47]:
```
top_2500['Good_Debt'] = top_2500['debt'] - top_2500['Bad_Debt']
```

In [48]:
```
top_2500.head(15)
```

Out[48]:

	state	lat	lng	second_mortgage	pct_own	place	state	city	COUNTYID	STATEID	hom
14014	New Jersey	40.867944	-74.114633	0.60870	0.01157	Garfield City	New Jersey	Passaic	31	34	
3285	Virginia	37.297357	-78.396452	0.50000	0.62069	Farmville	Virginia	Farmville	147	51	
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	25	
12896	Pennsylvania	39.952954	-75.202767	0.39024	0.05041	Millbourne	Pennsylvania	Philadelphia	101	42	
7453	Texas	30.285534	-97.747727	0.36364	0.01737	Austin City	Texas	Austin	453	48	
15589	Georgia	33.740758	-84.401777	0.34783	0.04026	Atlanta City	Georgia	Atlanta	121	13	
1680	Illinois	41.782569	-87.579504	0.33333	0.05267	Chicago City	Illinois	Chicago	31	17	
26018	New York	40.751809	-73.853582	0.31818	0.15618	Harbor Hills	New York	Corona	81	36	
23547	California	34.066049	-118.274164	0.31148	0.06960	Vernon City	California	Los Angeles	37	6	
7829	Maryland	39.127273	-76.635265	0.30212	0.22380	Glen Burnie	Maryland	Glen Burnie	3	24	
21880	Michigan	42.290397	-85.584143	0.30159	0.07085	Kalamazoo City	Michigan	Kalamazoo	77	26	
2077	Florida	28.029063	-82.495395	0.28972	0.11618	Egypt Lake-leto	Florida	Tampa	57	12	
1701	Illinois	41.967289	-87.652434	0.28899	0.14228	Lincolnwood	Illinois	Chicago	31	17	
11839	Illinois	41.906640	-87.689580	0.27431	0.29468	Chicago City	Illinois	Chicago	31	17	

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

In [49]:
```
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[49]:
```
('GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD',
 'GD',
 'BD')
```

In [50]:
```
l1 = list(top_2500['Bad_Debt'] )
l1[:5]
```

Out[50]:
```
[0.6087, 0.5, 0.4375, 0.43363, 0.60975]
```

In [51]:
```
l2 = list(top_2500['Good_Debt'] )
l2[:5]
```

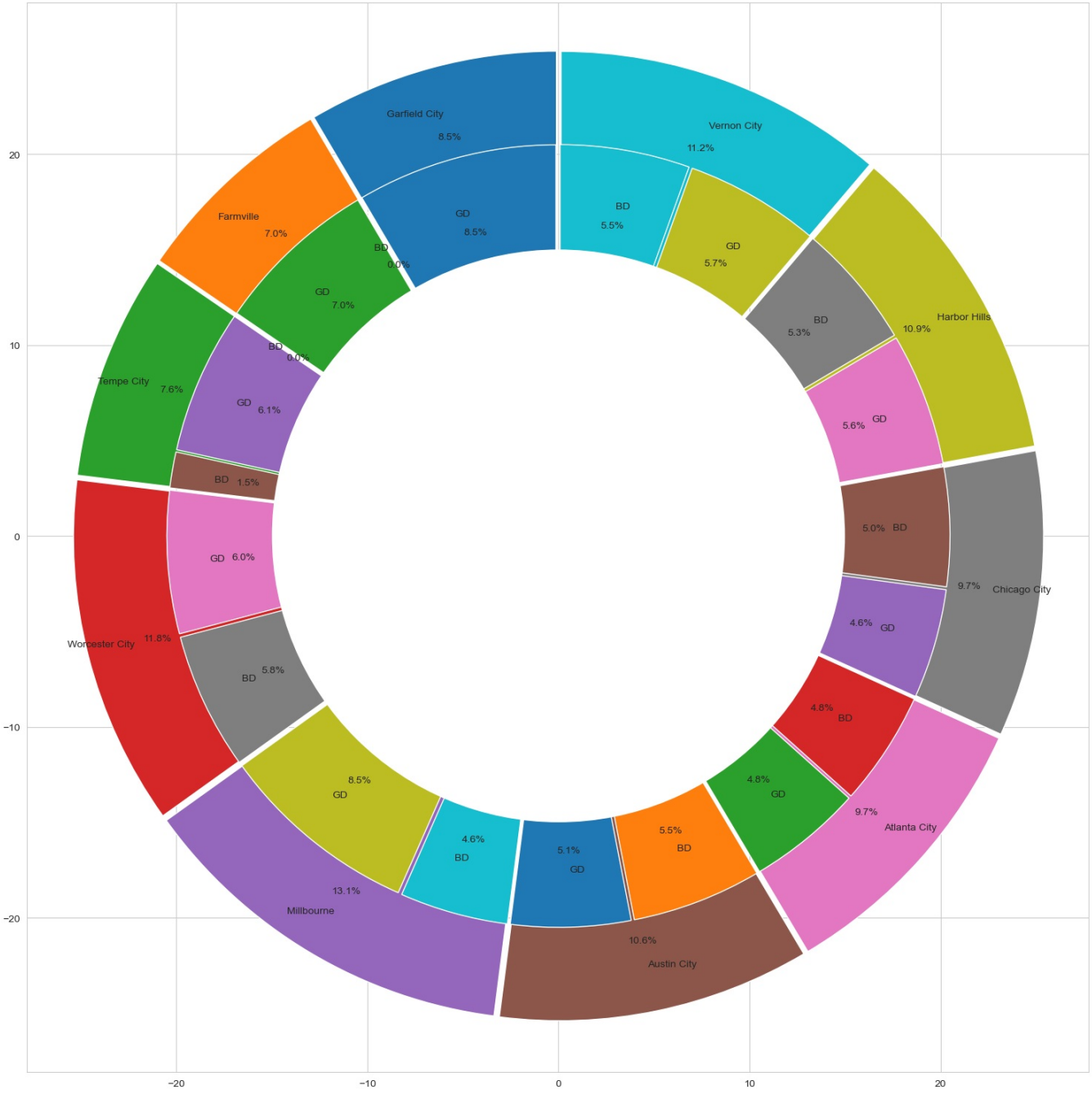
Out[51]:
```
[0.0, 0.0, 0.10938000000000003, 0.41592999999999997, 0.32926999999999995]
```

In [52]:
```
l3 = sum(zip(l1, l2+[0]), ())
```

In [53]:
```
l3[:10]
```

Out[53]:
```
(0.6087,
 0.0,
 0.5,
 0.0,
 0.4375,
 0.10938000000000003,
 0.43363,
 0.41592999999999997,
 0.60975,
 0.32926999999999995)
```

In [54]:
```
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']
plt.pie(debt, labels = labels, startangle = 90, frame = True, radius =25, autopct='%1.1f%%', pctdistance=0.85,
plt.pie(gd_bd, labels = labels_D, startangle = 90, frame = True, radius = 20, autopct='%1.1f%%', pctdistance=0.
centre_circle = plt.Circle((0,0),15,color='black', fc='white',linewidth=0)
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 [55]:
```
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 [56]:
```
top_2500['city'].value_counts()[:31].index
```

Out[56]:
```
CategoricalIndex(['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Sacramento', 'Long Beach', 'Colorado Springs', 'Cincinnati',
                  'Columbus', 'San Diego', 'New Orleans', 'Lowell', 'Orlando',
                  'Dallas', 'San Jose', 'Atlanta', 'Alexandria', 'Portland',
                  'Oakland', 'Houston', 'Littleton', 'Miami'],
                 categories=['Abbeville', 'Aberdeen', 'Abilene', 'Abingdon', 'Abington', 'Accokeek', 'Acton', '
Acushnet', ...], ordered=False, dtype='category')
```

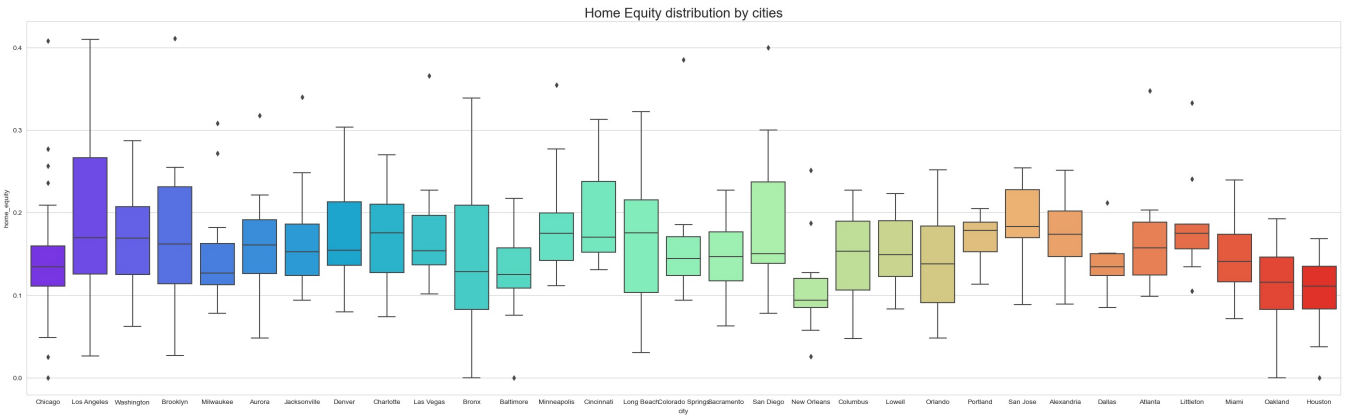
In [57]:
```
cities = ['Chicago', 'Los Angeles', 'Washington', 'Brooklyn',
                  'Milwaukee', 'Aurora', 'Jacksonville', 'Denver', 'Charlotte',
                  'Las Vegas', 'Bronx', 'Baltimore', 'Minneapolis',
                  'Cincinnati', 'Long Beach', 'Colorado Springs', 'Sacramento',
                  'San Diego', 'New Orleans', 'Columbus', 'Lowell', 'Orlando',
```

```
                  'Portland', 'San Jose', 'Alexandria', 'Dallas', 'Atlanta',
                  'Littleton', 'Miami', 'Oakland', 'Houston']
```

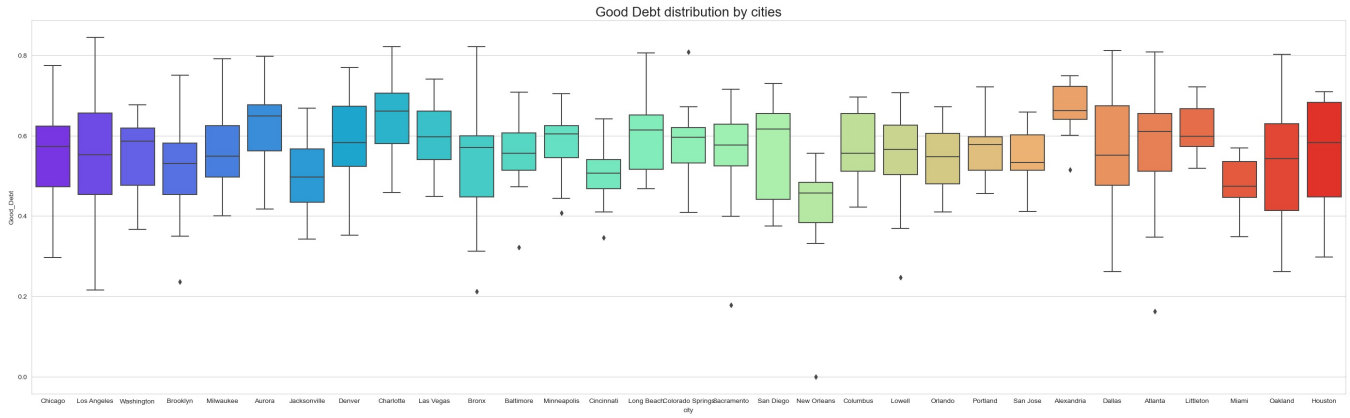
In [58]:
```
boxplot_df = top_2500[top_2500['city'].isin (cities)]
```

In [59]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))
sns.boxplot(x='city',y='second_mortgage',data=boxplot_df,palette='rainbow', order = ['Chicago', 'Los Angeles',
                  '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 citie
plt.show()
```

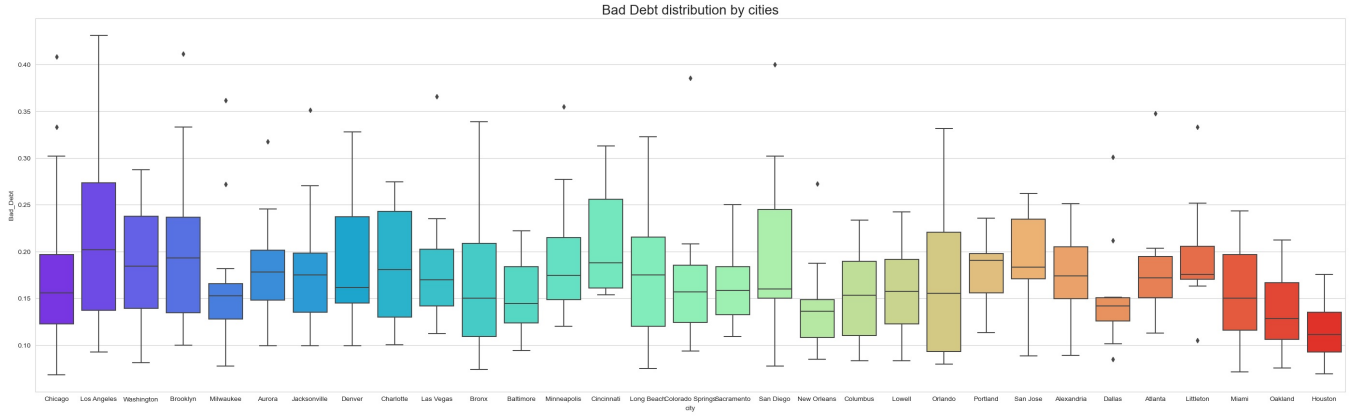
In [60]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))
sns.boxplot(x='city',y='home_equity',data=boxplot_df,palette='rainbow', order = ['Chicago', 'Los Angeles', 'Was
                  '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',
plt.show()
```



In [61]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))
sns.boxplot(x='city',y='Good_Debt',data=boxplot_df,palette='rainbow', order = ['Chicago', 'Los Angeles', 'Washi
                  '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', fo
plt.show()
```



In [62]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))
sns.boxplot(x='city',y='Bad_Debt',data=boxplot_df,palette='rainbow', order = ['Chicago', 'Los Angeles', 'Washin
                  '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', fon
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 [63]:
```
top_2500['remaining_income'] = top_2500['family_median'] - top_2500['hi_median']
```

In [64]:
```
income_chart = round(top_2500[['city', 'hi_median', 'family_median', 'remaining_income']], 2)
income_chart
```

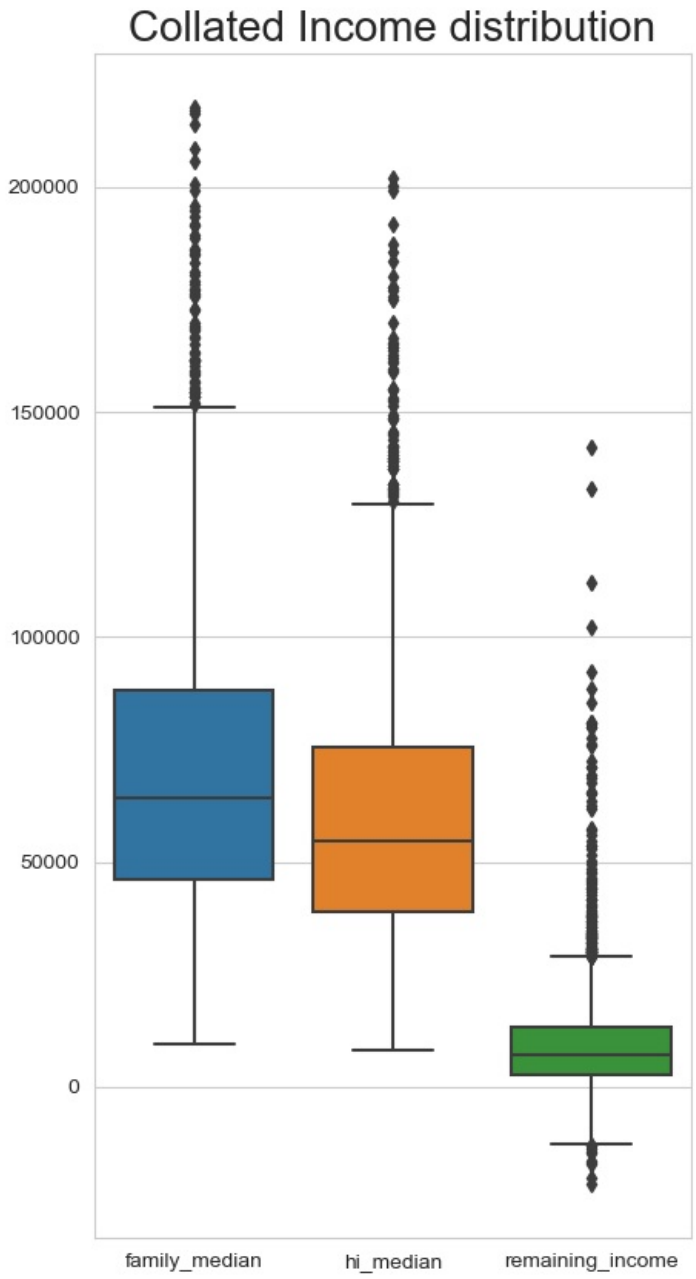
Out[64]:

	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 [65]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (5, 10))
```

```
sns.boxplot(data=top_2500[['family_median', 'hi_median', 'remaining_income']], palette=color_pal).set_title('Co
plt.show()
```



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

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 [66]:
```
train.head()
```

Out[66]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean	female_age_me
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	...	44.48629	45.3
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	...	36.48391	37.5
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	...	42.15810	42.8
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	...	47.77526	50.5
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	...	24.17693	21.5

5 rows × 78 columns

In [67]:
```
density_eda_df = train[['state', 'city', 'place', 'ALand', 'pop', 'male_age_median', 'female_age_median', 'male
density_eda_df.head()
```

Out[67]:

	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 [68]:
```
density_eda_df['pop_density'] = density_eda_df['pop'] / density_eda_df['ALand']
density_eda_df.head()
```

Out[68]:

	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 [69]:
```
density_eda_df['median_age'] = (density_eda_df['male_age_median'] *  density_eda_df['male_pop'] + density_eda_d
density_eda_df.head()
```

Out[69]:

	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop	pop_density	median_age
0	New York	Hamilton	Hamilton	202183361.0	5230	44.00000	45.33333	2612	2618	0.000026	44.667430
1	Indiana	South Bend	Roseland	1560828.0	2633	32.00000	37.58333	1349	1284	0.001687	34.722748
2	Indiana	Danville	Danville	69561595.0	6881	40.83333	42.83333	3643	3238	0.000099	41.774472
3	Puerto Rico	San Juan	Guaynabo	1105793.0	2700	48.91667	50.58333	1141	1559	0.002442	49.879012
4	Kansas	Manhattan	Manhattan City	2554403.0	5637	22.41667	21.58333	2586	3051	0.002207	21.965629

In [70]:
```
density_eda_df.nlargest(300, 'pop_density')
```

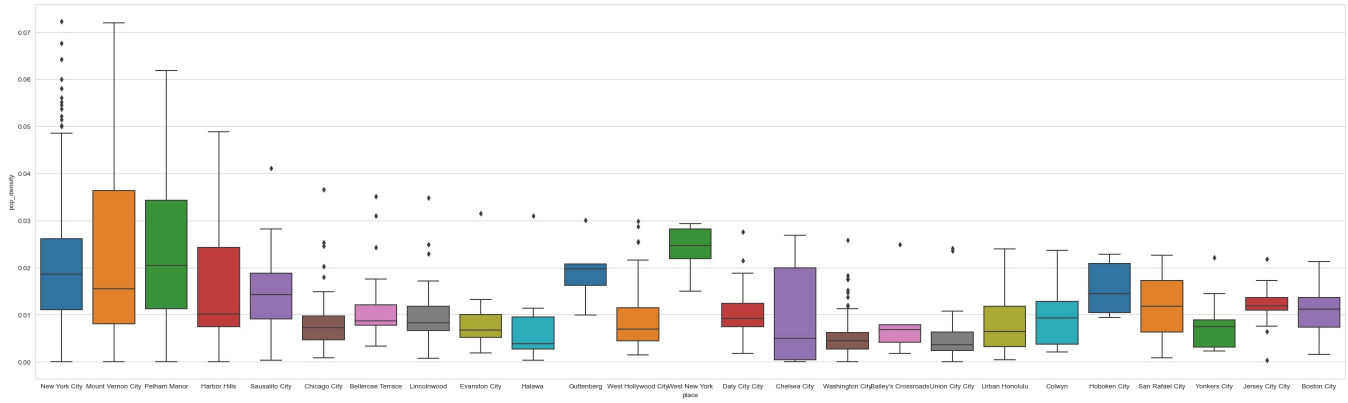
Out[70]:

	state	city	place	ALand	pop	male_age_median	female_age_median	male_pop	female_pop	pop_density	median_a
21050	New York	New York	New York City	182091.0	13162	38.83333	34.66667	5597	7565	0.072283	36.4384
10251	New York	New York	Mount Vernon City	169349.0	12189	33.25000	35.33333	6110	6079	0.071976	34.2890
1546	New York	New York	New York City	183653.0	12427	37.00000	41.83333	5425	7002	0.067666	39.7233
23760	New York	New York	New York City	181779.0	11688	39.25000	41.50000	5011	6677	0.064298	40.5353
13022	New York	Bronx	Mount Vernon City	67355.0	4229	27.75000	26.66667	1932	2297	0.062787	27.1615
...	...	...	...	...	...	...	...	...	...	...	
14705	New Jersey	Guttenberg	Guttenberg	178469.0	3715	33.66667	34.00000	1893	1822	0.020816	33.8301
706	New York	Brooklyn	New York City	184193.0	3829	29.58333	34.66667	1824	2005	0.020788	32.2451
16852	New Jersey	Jersey City	Hoboken City	219021.0	4545	30.50000	32.41667	2330	2215	0.020751	31.4340
8015	New York	Brooklyn	New York City	207813.0	4304	44.00000	47.00000	2196	2108	0.020711	45.4693
19946	New York	Brooklyn	New York City	165897.0	3418	42.33333	38.83333	1468	1950	0.020603	40.3365

300 rows × 11 columns

In [71]:

```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df.nlargest(26585, 'pop_density'), palette=color_p
 '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()
```

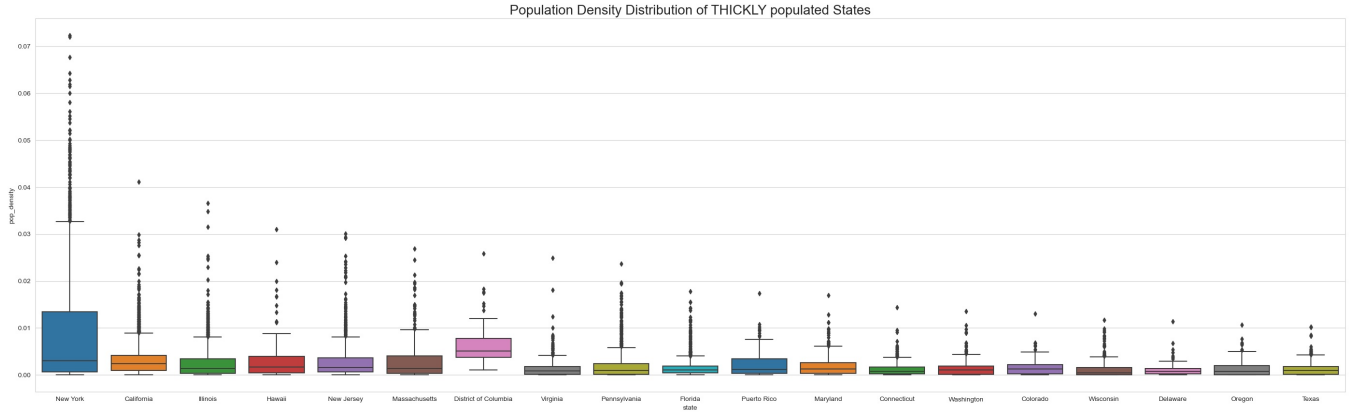


In [72]:

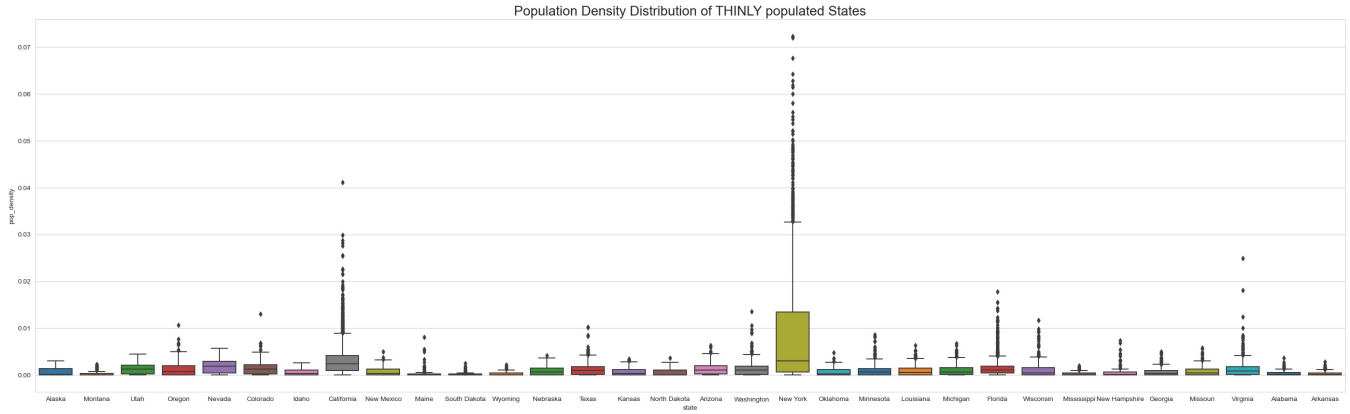
```
list(density_eda_df.nsmallest(450, 'pop_density').state.unique())
```

Out[72]:
```
['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 [73]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
sns.boxplot(x = 'state', y = 'pop_density', data=density_eda_df.nlargest(26585, 'pop_density'), palette=color_p


plt.show()
```

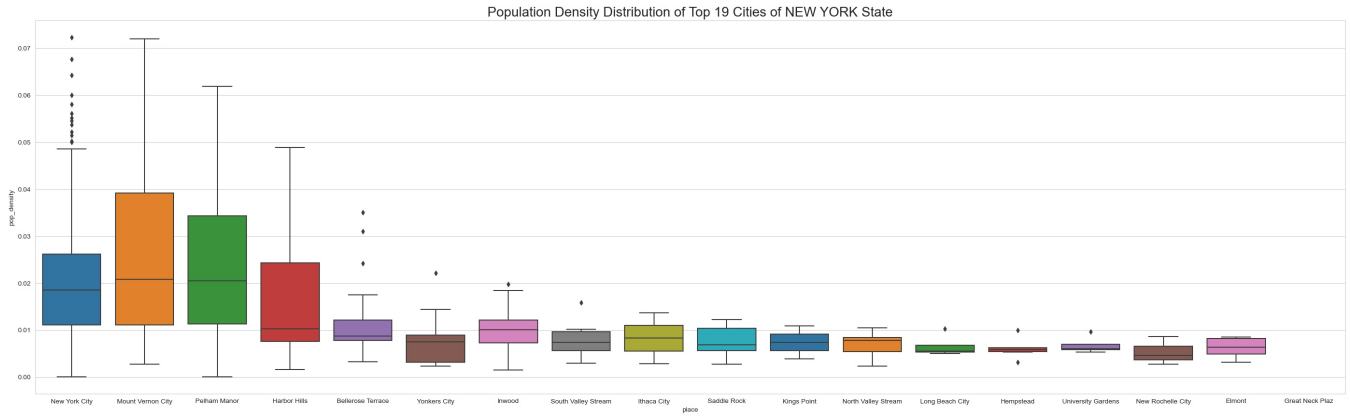


In [74]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
sns.boxplot(x = 'state', y = 'pop_density', data=density_eda_df.nsmallest(26585, 'pop_density'), palette=color_



plt.show()
```

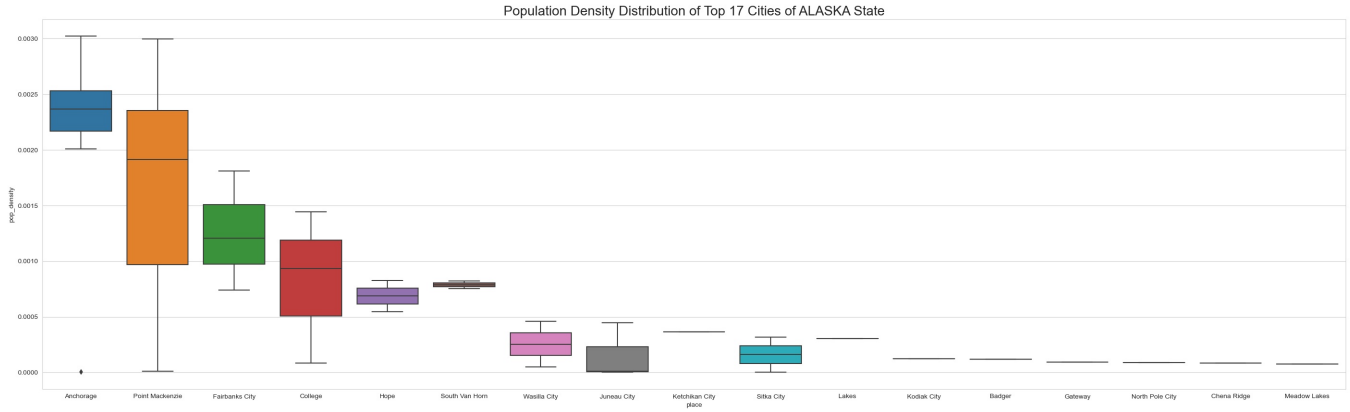


In [75]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df[density_eda_df['state'] == 'New York'].nlargest
 'Mount Vernon City',
 'Pelham Manor',
```

```
 '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 = 20)
plt.show()
```



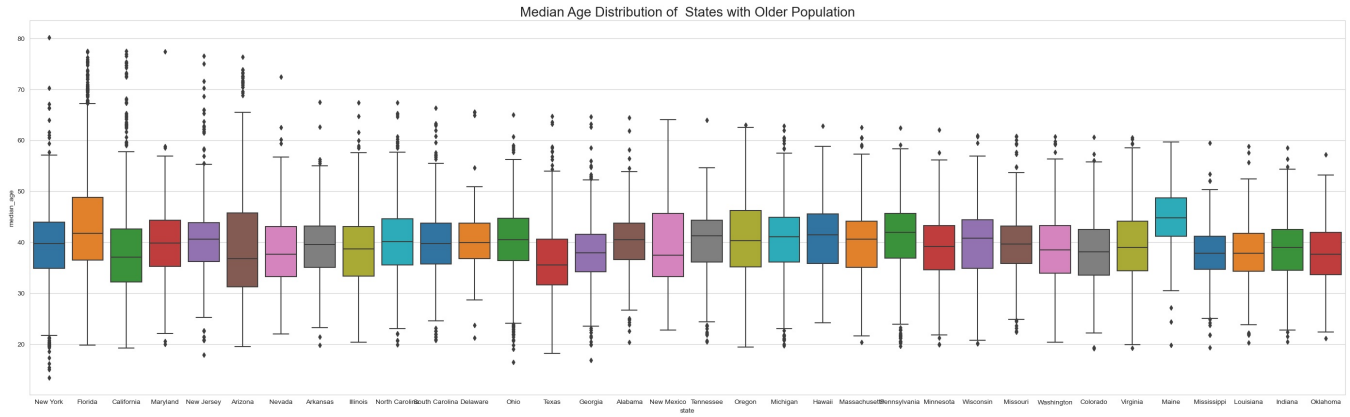
In [76]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
sns.boxplot(x = 'place', y = 'pop_density', data=density_eda_df[density_eda_df['state'] == 'Alaska'].nlargest(2

).set_title('Population Density Distribution of Top 17 Cities of ALASKA State', fontsize = 20)
plt.show()
```



In [77]:
```
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', 'Arkansas', 'Illinois', 'N
orth Carolina', 'South Carolina', 'Delaware', 'Ohio', 'Texas', 'Georgia', 'Alabama', 'New Mexico', 'Tennessee',
'Oregon', 'Michigan', 'Hawaii', 'Massachusetts', 'Pennsylvania', 'Minnesota', 'Wisconsin', 'Missouri', 'Washing
ton', 'Colorado', 'Virginia', 'Maine', 'Mississippi', 'Louisiana', 'Indiana', 'Oklahoma']
34
```

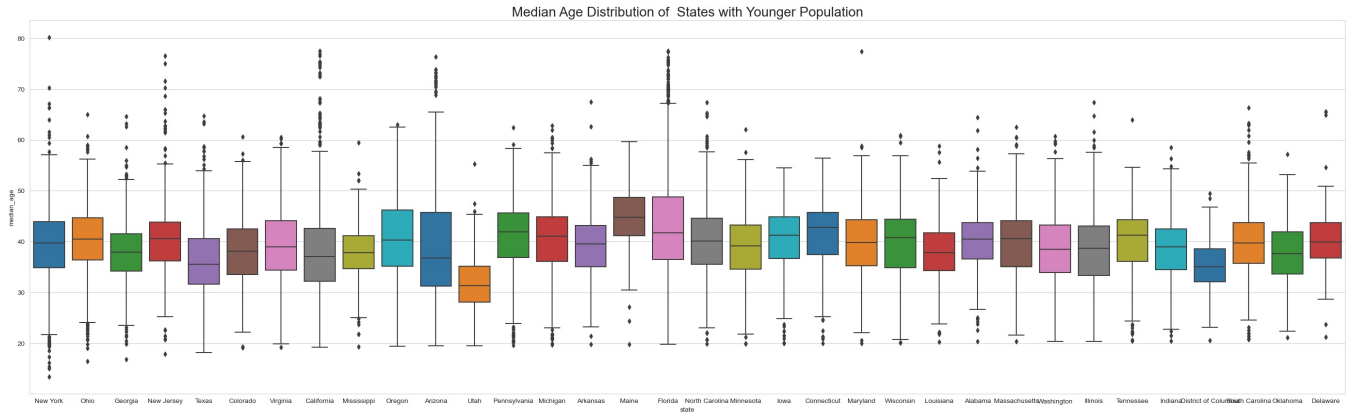
In [78]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))


ax = sns.boxplot(x = 'state', y = 'median_age', data=density_eda_df.nlargest(26585, 'median_age'), palette=colo
            order = ['New York', 'Florida', 'California', 'Maryland', 'New Jersey', 'Arizona', 'Nevada', 'Arkan
            'Oregon', 'Michigan', 'Hawaii', 'Massachusetts', 'Pennsylvania', 'Minnesota', 'Wisconsin', 'Missour
           ).set_title('Median Age Distribution of  States with Older Population', fontsize = 20)

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

plt.show()
```



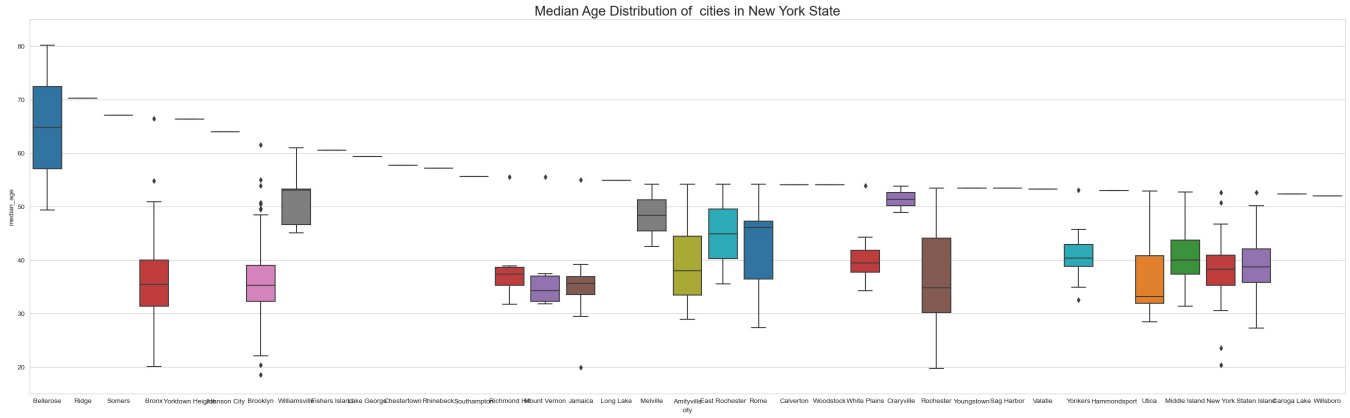
In [79]:
```
print(list(density_eda_df.nsmallest(150, 'median_age').state.unique()))
print(len(list(density_eda_df.nsmallest(150, 'median_age').state.unique())))
```

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

In [80]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
ax = sns.boxplot(x = 'state', y = 'median_age', data=density_eda_df.nsmallest(26585, 'median_age'), palette=col
            order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California'
                     'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass
           ).set_title('Median Age Distribution of  States with Younger Population', fontsize = 20)
#ax.set(ylim=(0, 100))
plt.show()
```

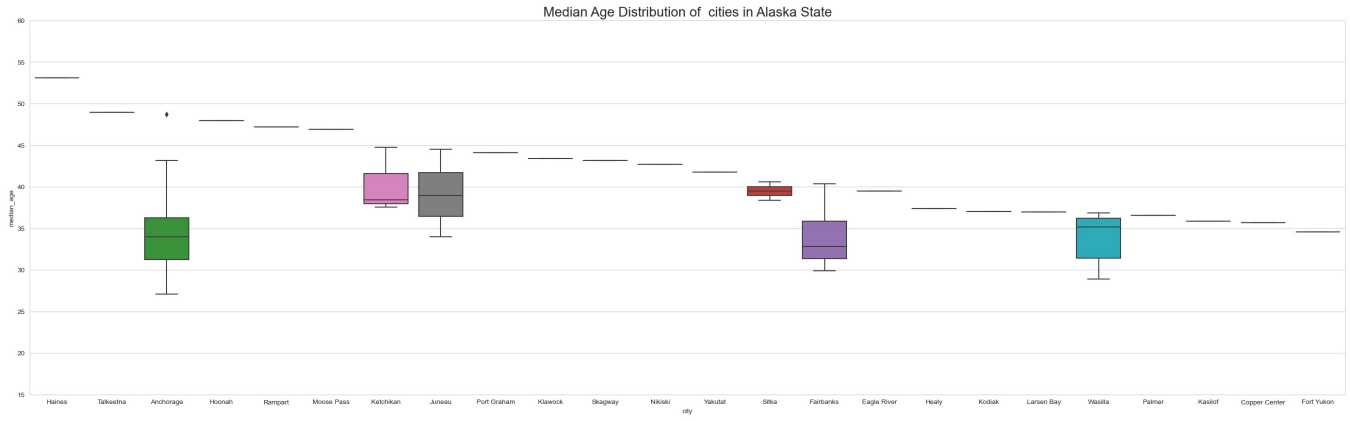


In [81]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
ax = sns.boxplot(x = 'city', y = 'median_age', data=density_eda_df[density_eda_df['state'] == 'New York'].nlarg
            order =['Bellerose', 'Ridge', 'Somers', 'Bronx', 'Yorktown Heights', 'Johnson City', 'Brooklyn', 'W
                    'Southampton', 'Richmond Hill', 'Mount Vernon', 'Jamaica', 'Long Lake', 'Melville', 'Amityv
                    'Rochester', 'Youngstown', 'Sag Harbor', 'Valatie', 'Yonkers', 'Hammondsport', 'Utica', 'Mi
)
ax.set_title('Median Age Distribution of  cities in New York State', fontsize = 20)
ax.set(ylim=(15, 85))
plt.show()
```



In [82]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
ax = sns.boxplot(x = 'city', y = 'median_age', data=density_eda_df[density_eda_df['state'] == 'Alaska'].nlarges
            order =['Haines', 'Talkeetna', 'Anchorage', 'Hoonah', 'Rampart', 'Moose Pass', 'Ketchikan', 'Juneau
                    'Eagle River', 'Healy', 'Kodiak', 'Larsen Bay', 'Wasilla', 'Palmer', 'Kasilof', 'Copper Cen
)
```

```
ax.set_title('Median Age Distribution of  cities in Alaska State', fontsize = 20)
ax.set(ylim=(15, 60))
plt.show()
```



In [83]:
```
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 [84]:
```
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 River', 'Healy', 'Kodiak', 'Larsen Bay'
, 'Wasilla', 'Palmer', 'Kasilof', 'Copper Center', 'Fort Yukon']
24
```

In [85]:
```
train.head()
```

Out[85]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean	female_age_me
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	...	44.48629	45.3
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	...	36.48391	37.5
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	...	42.15810	42.8
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	...	47.77526	50.5
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	...	24.17693	21.5

5 rows × 78 columns

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 [86]:
```
age_df = train[['state', 'city', 'place', 'pop', 'male_pop', 'female_pop', 'male_age_median', 'female_age_media
```

In [87]:
```
train.male_age_median.unique()
```

Out[87]:
```
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   ])
```

In [88]:
```
bins = [0, 12,18, 35, 55, 100]
labels = ['kids', 'Youth', 'Young Adult', 'Adult', 'Senior']
```

In [89]:
```
age_df['male_population_bracket'] = pd.cut(age_df['male_age_median'], bins, labels = labels)
```

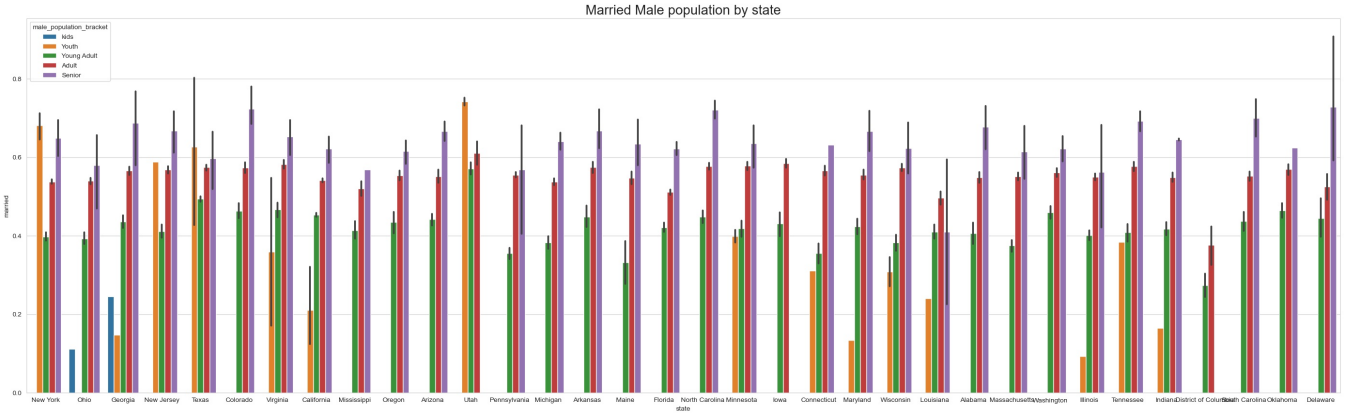
In [90]:
```
age_df['female_population_bracket'] = pd.cut(age_df['female_age_median'], bins, labels = labels)
```

In [91]:
```
age_df.head()
```

Out[91]:

	state	city	place	pop	male_pop	female_pop	male_age_median	female_age_median	married	separated	divorced	male_po
0	New York	Hamilton	Hamilton	5230	2612	2618	44.00000	45.33333	0.57851	0.01240	0.08770	
1	Indiana	South Bend	Roseland	2633	1349	1284	32.00000	37.58333	0.34886	0.01426	0.09030	
2	Indiana	Danville	Danville	6881	3643	3238	40.83333	42.83333	0.64745	0.01607	0.10657	
3	Puerto Rico	San Juan	Guaynabo	2700	1141	1559	48.91667	50.58333	0.47257	0.02021	0.10106	
4	Kansas	Manhattan	Manhattan City	5637	2586	3051	22.41667	21.58333	0.12356	0.00000	0.03109	

In [92]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))

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',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Married Male population by state', fontsize = 20)

plt.show()
```

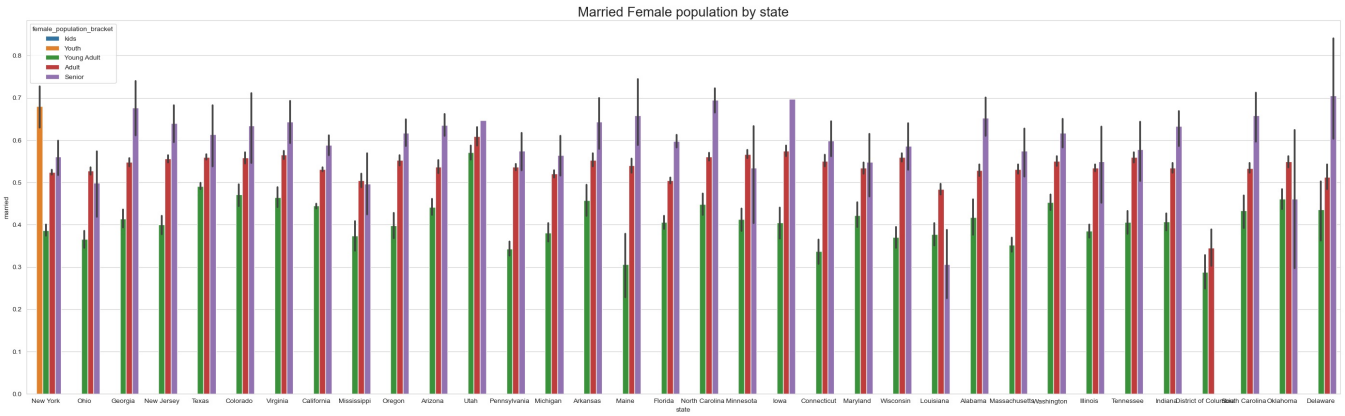


Surprisingly, "Ohio & Georgia" have Married Male KIDS

In [93]:
```
age_df.city.unique()
```

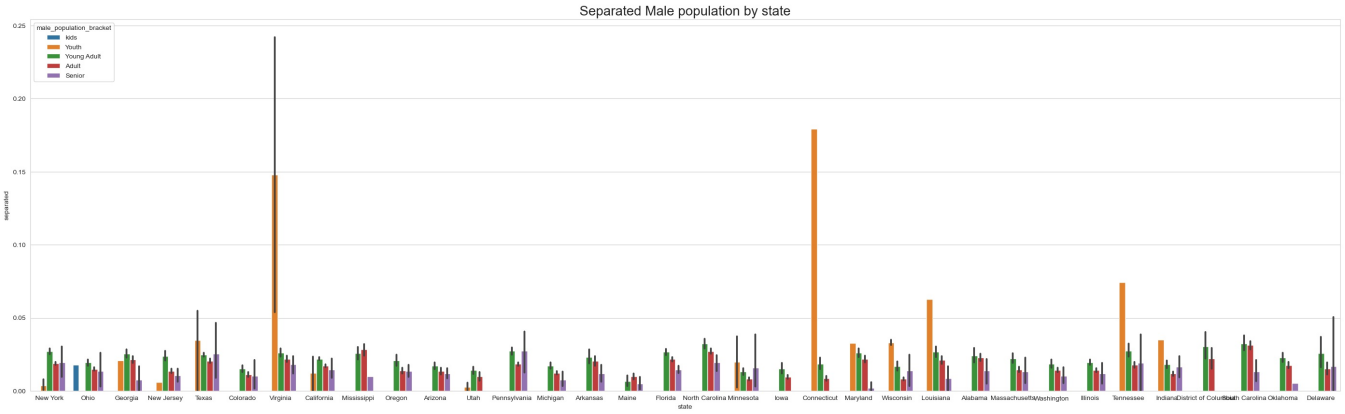
Out[93]:
```
['Hamilton', 'South Bend', 'Danville', 'San Juan', 'Manhattan', ..., 'Cresco', 'Wittensville', 'Blue Bell', 'We
ldona', 'Colleyville']
Length: 6876
Categories (6876, object): ['Abbeville', 'Aberdeen', 'Abilene', 'Abingdon', ..., 'Zoarville', 'Zolfo Springs',
'Zumbrota', 'Zuni']
```

In [94]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))

ax = sns.barplot(x = 'state', y = 'married', hue = 'female_population_bracket', data = age_df, palette=color_pa
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Married Female population by state', fontsize = 20)

plt.show()
```



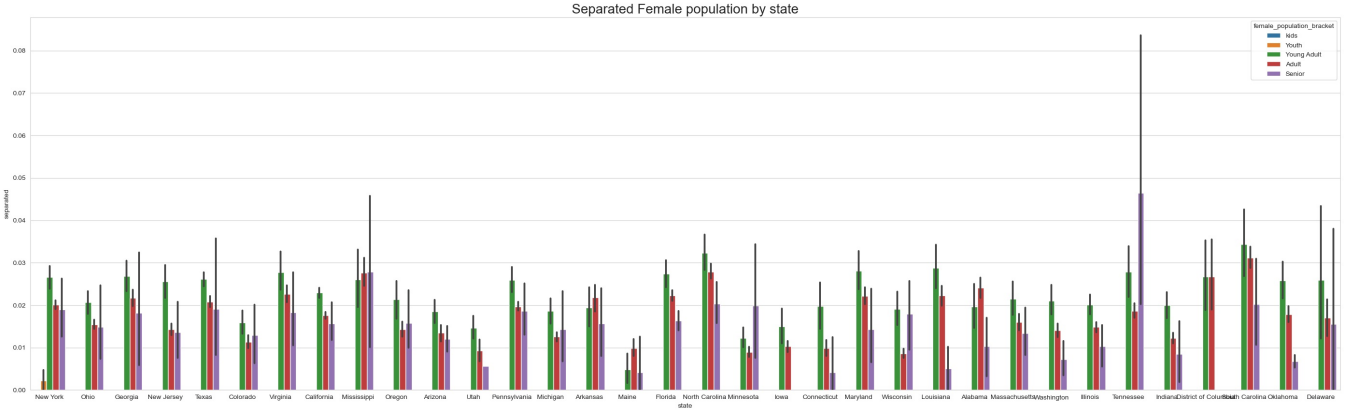
Except for "Newyork", NO other state has Married Female KIDS or Youth

In [95]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))

ax = sns.barplot(x = 'state', y = 'separated', hue = 'male_population_bracket', data = age_df, palette=color_pa
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Separated Male population by state', fontsize = 20)

plt.show()
```



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

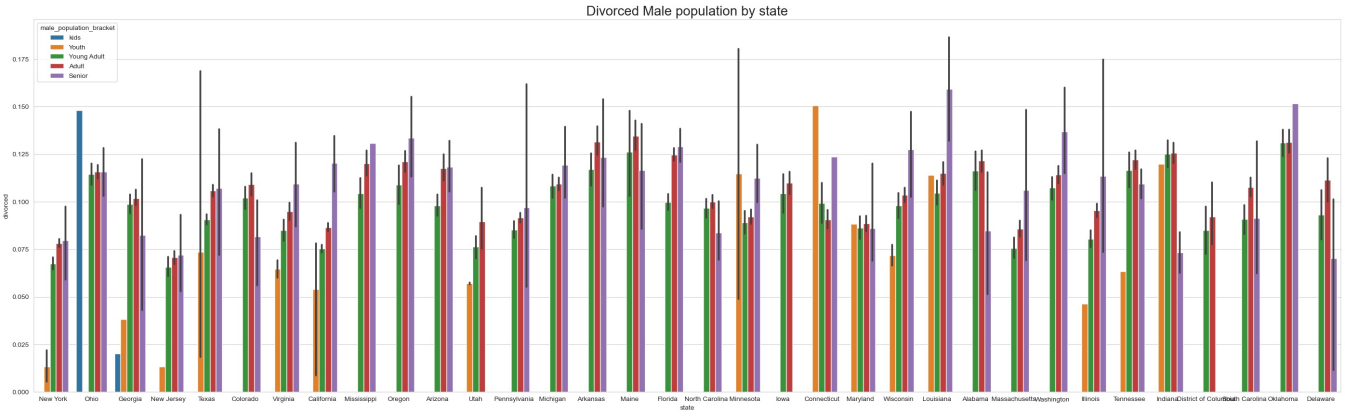
In [96]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))

ax = sns.barplot(x = 'state', y = 'separated', hue = 'female_population_bracket', data = age_df, palette=color_
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Separated Female population by state', fontsize = 20)

plt.show()
```



Except for "Newyork", No other state has Separated Female Youth population

"Tennessee" has the Highest Separated Female SENIOR population

In [97]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))

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',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Divorced Male population by state', fontsize = 20)

plt.show()
```

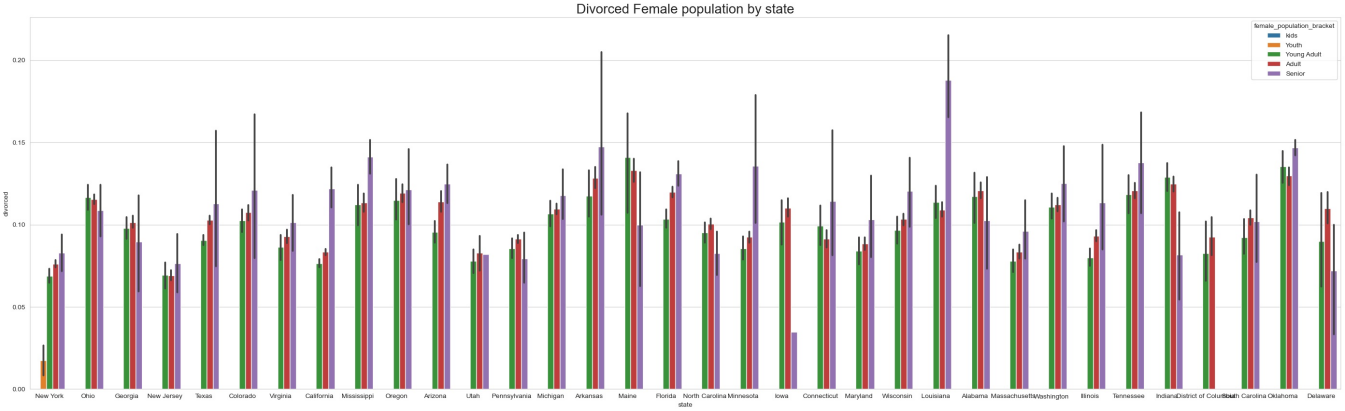


"Ohio", has Largest number of Divorced Male KIDS.

"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 [98]:
```
sns.set_style("whitegrid")

plt.figure(figsize = (35, 10))

ax = sns.barplot(x = 'state', y = 'divorced', hue = 'female_population_bracket', data = age_df, palette=color_p
           order = ['New York', 'Ohio', 'Georgia', 'New Jersey', 'Texas', 'Colorado', 'Virginia', 'California',
                    'Minnesota', 'Iowa', 'Connecticut', 'Maryland', 'Wisconsin', 'Louisiana', 'Alabama', 'Mass

ax.set_title('Divorced Female population by state', fontsize = 20)

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 [99]:
```
train.head()
```

Out[99]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	...	female_age_mean	female_age_me
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	13346	...	44.48629	45.3
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	46616	...	36.48391	37.5
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	46122	...	42.15810	42.8
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	...	47.77526	50.5
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	...	24.17693	21.5

5 rows × 78 columns

In [100…
```
rent_df = train[['state', 'city', 'rent_median', 'hi_median', 'family_median']]
```

In [101…
```
Overall_rent_percentage = (rent_df['rent_median'].sum() / rent_df['hi_median'].sum()) * 100
round(Overall_rent_percentage, 2)
```

Out[101]: 1.74

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

In [102…
```
rent_df['ov_rent_pcnt'] = round((rent_df['rent_median'] / rent_df['hi_median']) * 100, 2)
```

In [103…
```
rent_df.head()
```

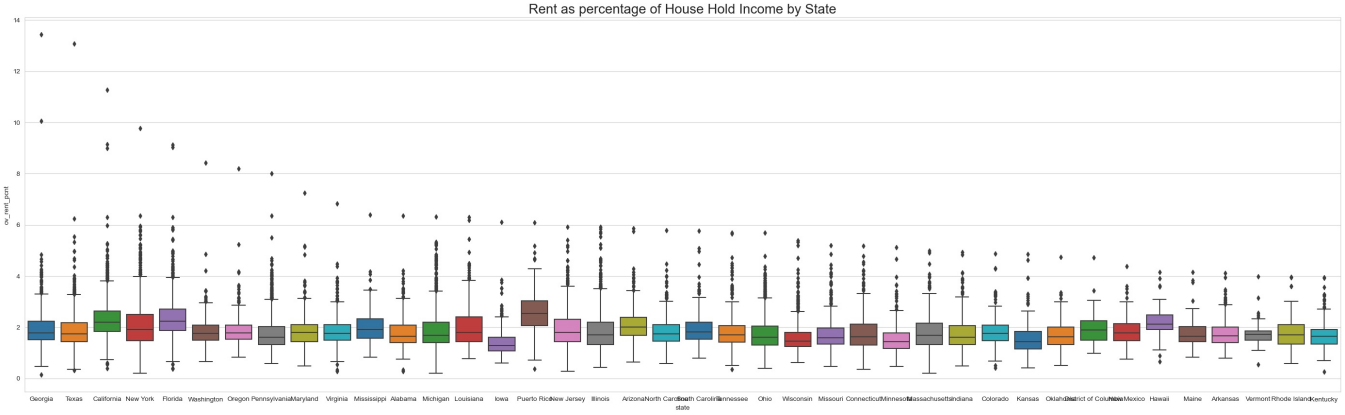
Out[103]:

	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 [104…
```
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', 'Pennsylvania', 'Maryland', '
Virginia', 'Mississippi', 'Alabama', 'Michigan', 'Louisiana', 'Iowa', 'Puerto Rico', 'New Jersey', 'Illinois',
'Arizona', 'North Carolina', 'South Carolina', 'Tennessee', 'Ohio', 'Wisconsin', 'Missouri', 'Connecticut', 'Mi
nnesota', 'Massachusetts', 'Indiana', 'Colorado', 'Kansas', 'Oklahoma', 'District of Columbia', 'New Mexico', '
Hawaii', 'Maine', 'Arkansas', 'Vermont', 'Rhode Island', 'Kentucky']
40
```

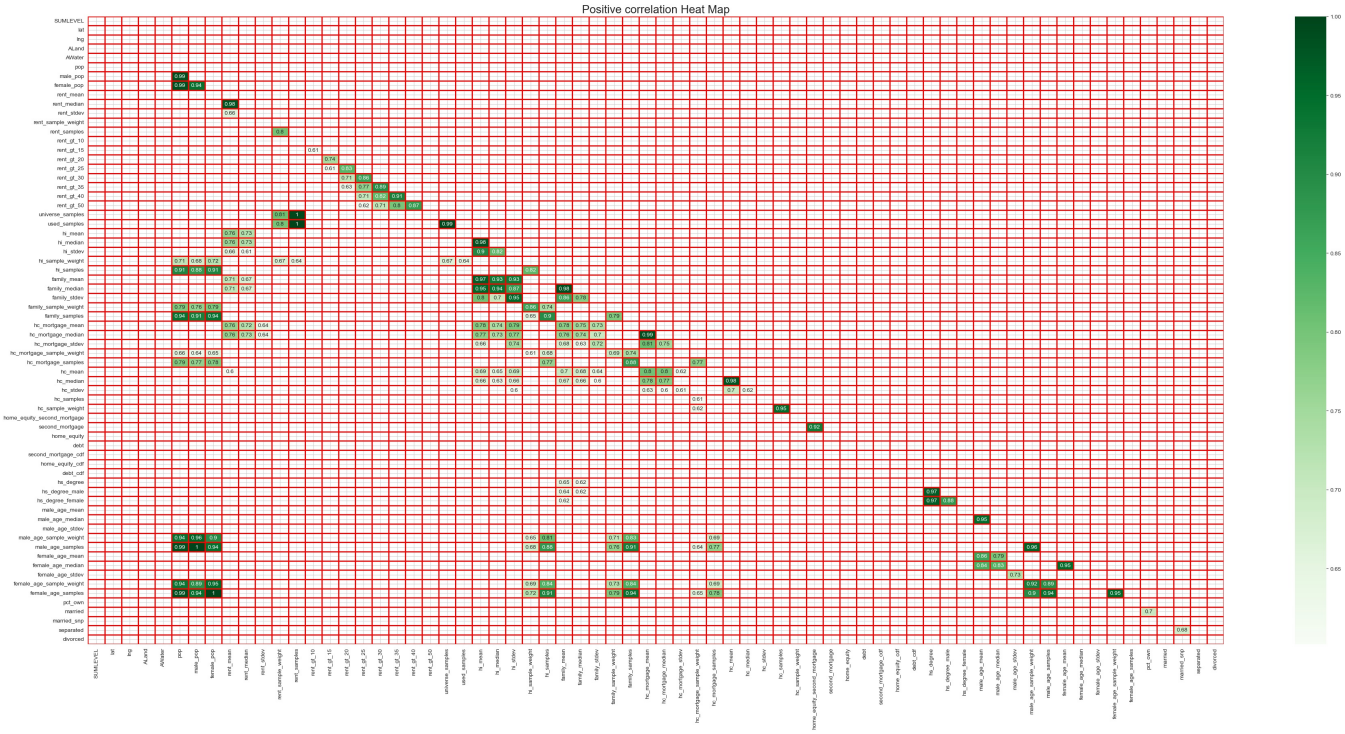
In [105…
```
sns.set_style("whitegrid")
plt.figure(figsize = (35, 10))
ax = sns.boxplot(x = 'state', y = 'ov_rent_pcnt', data=rent_df.nlargest(26585, 'ov_rent_pcnt'), palette=color_p
            order = ['Georgia', 'Texas', 'California', 'New York', 'Florida', 'Washington', 'Oregon', 'Pennsylv
                     'Iowa', 'Puerto Rico', 'New Jersey', 'Illinois', 'Arizona', 'North Carolina', 'South Carol
                     'Massachusetts', 'Indiana', 'Colorado', 'Kansas', 'Oklahoma', 'District of Columbia', 'New
           ).set_title('Rent as percentage of House Hold Income by State', fontsize = 20)
#ax.set(ylim=(0, 100))
plt.show()
```



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

In [106…
```
sns.set_style("whitegrid")
```

```
corr = train.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))S
sns.heatmap(kot, cmap="Greens", annot = True, mask = mask, linewidths=1, linecolor='red').set_title('Positive c
plt.grid('on', )
plt.show()
```



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

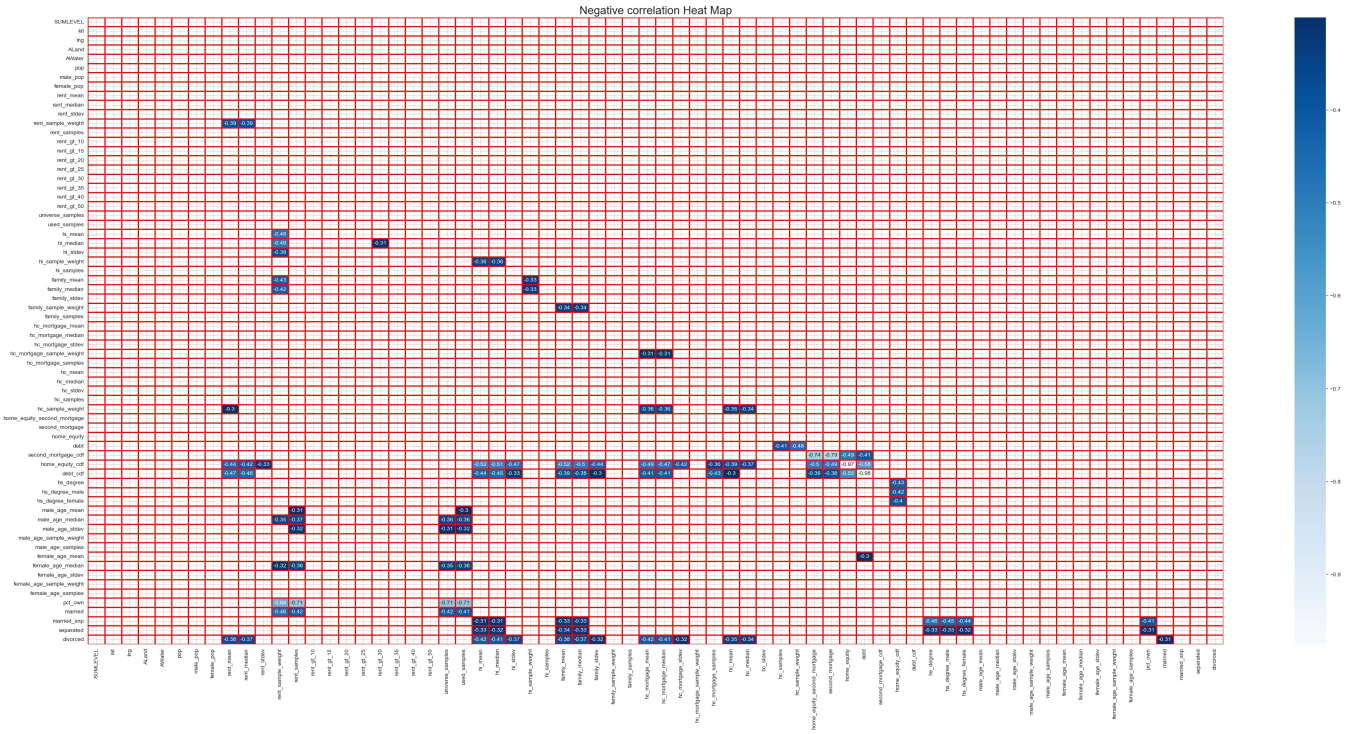
- "Male Population is highly correlated with Female population.
- "rent Mean & Median" has high positive correlation with "House hold income Mean, Median and Standard Deviation",

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

- "House hold income and Family income are highly positively correlated.
- "Family Income" and "hc_mortgage" are positively correlated.
- "pct_own" is positively correlated with "Married" marital status

In [107…

```
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 co
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"]
- "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..

Data Pre-processing:

Project Task: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.
2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings. Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable. Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data. Following are the list of latent variables:

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

In [108…

```
train.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: 16.9 MB
```

In [109…

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

In [110

```
for col in train.columns:
```

```
    print(col,' = ' ,train[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 [111…

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

In [112…

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

```
    return num_variables
```

In [113…
```
train.city.dtype
```

Out[113]:
```
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 [114…
```
cat_variables(train)
```

Out[114]:
```
['UID',
 'COUNTYID',
 'STATEID',
 'state',
 'state_ab',
 'city',
 'place',
 'type',
 'zip_code',
 'area_code']
```

In [115…
```
num_variables(train)
```

Out[115]:
```
['SUMLEVEL',
 'lat',
 'lng',
 'ALand',
 'AWater',
 'pop',
 'male_pop',
 'female_pop',
 'rent_mean',
 'rent_median',
 'rent_stdev',
 'rent_sample_weight',
 'rent_samples',
 'rent_gt_10',
 'rent_gt_15',
 'rent_gt_20',
 'rent_gt_25',
 'rent_gt_30',
 'rent_gt_35',
 'rent_gt_40',
 'rent_gt_50',
 'universe_samples',
 'used_samples',
 'hi_mean',
 'hi_median',
 'hi_stdev',
 'hi_sample_weight',
 'hi_samples',
 'family_mean',
 'family_median',
 'family_stdev',
 'family_sample_weight',
 'family_samples',
 'hc_mortgage_mean',
 'hc_mortgage_median',
 'hc_mortgage_stdev',
 'hc_mortgage_sample_weight',
 'hc_mortgage_samples',
 'hc_mean',
 'hc_median',
 'hc_stdev',
 'hc_samples',
 'hc_sample_weight',
 'home_equity_second_mortgage',
 'second_mortgage',
 'home_equity',
 'debt',
 'second_mortgage_cdf',
 'home_equity_cdf',
 'debt_cdf',
 'hs_degree',
 'hs_degree_male',
 'hs_degree_female',
 'male_age_mean',
 'male_age_median',
 'male_age_stdev',
 'male_age_sample_weight',
 'male_age_samples',
 'female_age_mean',
 'female_age_median',
 'female_age_stdev',
 'female_age_sample_weight',
 'female_age_samples',
 'pct_own',
 'married',
 'married_snp',
 'separated',
 'divorced',
 'Bad_Debt']
```

In [116…
```
fa_train = train[num_variables(train)]
fa_train
```

Out[116]:

	SUMLEVEL	lat	lng	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	...	female_age
0	140	42.840812	-75.501524	2.021834e+08	1699120	5230	2612	2618	769.38638	784.0	...	
1	140	41.701441	-86.266614	1.560828e+06	100363	2633	1349	1284	804.87924	848.0	...	
2	140	39.792202	-86.515246	6.956160e+07	284193	6881	3643	3238	742.77365	703.0	...	
3	140	18.396103	-66.104169	1.105793e+06	0	2700	1141	1559	803.42018	782.0	...	
4	140	39.195573	-96.569366	2.554403e+06	0	5637	2586	3051	938.56493	881.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
27316	140	18.076060	-66.358379	6.970300e+05	0	1847	909	938	439.42839	419.0	...	
27317	140	40.158138	-75.307271	5.077337e+06	11786	4155	2116	2039	1813.19253	1788.0	...	
27318	140	40.410316	-103.814003	1.323262e+09	17577610	2829	1465	1364	849.39107	834.0	...	
27319	140	32.904866	-97.162151	1.865230e+07	158882	11542	5727	5815	1972.45746	1843.0	...	
27320	140	36.064754	-115.152237	7.796308e+06	0	3726	1815	1911	949.84199	924.0	...	

26585 rows × 69 columns

In [117…
```
fa_train = fa_train[fa_train.columns[~fa_train.columns.isin(['SUMLEVEL', 'lat', 'lng',
                                                             'AL
                                                            ])]]
```

In [118…
```
from factor_analyzer import FactorAnalyzer
import warnings
warnings.filterwarnings('ignore')
```

In [120…
```
xvals = range(1, fa_train.shape[1]+1)
```

In [121…
```
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 = 'red'
  elif value > 0.6:
    color = 'green'
  else:
    color = 'black'

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

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

In [122…
```
len(fa_train.columns)
```

Out[122]: 65

Data Modeling :

Project Task: 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 [123…
```
train = pd.read_csv('train.csv1')
```

In [124…
```
train.head()
```

Out[124]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_m
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.

5 rows × 80 columns

In [125…
```
train.isna().sum()
```

Out[125]:
```
UID                0
BLOCKID        27321
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...
pct_own          268
married          191
married_snp      191
separated        191
divorced         191
Length: 80, dtype: int64
```

In [126…
```
train.head()
```

Out[126]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_m
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.

5 rows × 80 columns

In [127…
```
null_data = train[train.isnull().any(axis=1)]
null_data
```

Out[127]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	fe
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	
...	...	...	...	...	...	...	...	...	...	...	...	...	
27316	279212	NaN	140	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	...	42.73154	
27317	277856	NaN	140	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	...	38.21269	
27318	233000	NaN	140	87	8	Colorado	CO	Weldona	Saddle Ridge	City	...	43.40218	
27319	287425	NaN	140	439	48	Texas	TX	Colleyville	Colleyville City	Town	...	39.25921	
27320	265371	NaN	140	3	32	Nevada	NV	Las Vegas	Paradise	City	...	34.45345	

27321 rows × 80 columns

In [128…
```
train.drop('BLOCKID', axis=1, inplace=True)
```

In [129…
```
test.drop('BLOCKID', axis=1, inplace=True)
```

In [130…
```
train.isna().sum()
```

Out[130]:
```
UID              0
SUMLEVEL         0
COUNTYID         0
STATEID          0
state            0
                ...
pct_own        268
married        191
married_snp    191
separated      191
divorced       191
Length: 79, dtype: int64
```

In [131…
```
test.isna().sum()
```

Out[131]:
```
UID              0
SUMLEVEL         0
COUNTYID         0
STATEID          0
state            0
                ...
pct_own        122
married         84
married_snp     84
separated       84
divorced        84
Length: 79, dtype: int64
```

In [132…
```
train = train.dropna()
train = train.reset_index(drop=True)
```

In [133…
```
test = test.dropna()
test = test.reset_index(drop=True)
```

In [134…
```
train.shape
```

Out[134]:
```
(26585, 79)
```

In [135…
```
test.shape
```

Out[135]:
```
(11355, 79)
```

In [136…
```
train[cat_columns]
```

Out[136]:

	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 [137…
```
train[num_variables(train)]
```

Out[137]:

	UID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AWater	...	female_age_mea
0	267822	140	53	36	13346	315	42.840812	-75.501524	2.021834e+08	1699120	...	44.486
1	246444	140	141	18	46616	574	41.701441	-86.266614	1.560828e+06	100363	...	36.483
2	245683	140	63	18	46122	317	39.792202	-86.515246	6.956160e+07	284193	...	42.158
3	279653	140	127	72	927	787	18.396103	-66.104169	1.105793e+06	0	...	47.775
4	247218	140	161	20	66502	785	39.195573	-96.569366	2.554403e+06	0	...	24.176
...	...	...	...	...	...	...	...	...	...	...	...	
26580	279212	140	43	72	769	787	18.076060	-66.358379	6.970300e+05	0	...	42.731
26581	277856	140	91	42	19422	215	40.158138	-75.307271	5.077337e+06	11786	...	38.212
26582	233000	140	87	8	80653	970	40.410316	-103.814003	1.323262e+09	17577610	...	43.402
26583	287425	140	439	48	76034	817	32.904866	-97.162151	1.865230e+07	158882	...	39.259
26584	265371	140	3	32	89123	702	36.064754	-115.152237	7.796308e+06	0	...	34.453

26585 rows × 73 columns

In [138…
```
train.drop('SUMLEVEL', inplace = True, axis = 1)
```

In [139…
```
test.drop('SUMLEVEL', inplace = True, axis = 1)
```

In [140…
```
train[num_variables(train)]
```

Out[140]:

	UID	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AWater	pop	...	female_age_mean	fe
0	267822	53	36	13346	315	42.840812	-75.501524	2.021834e+08	1699120	5230	...	44.48629	
1	246444	141	18	46616	574	41.701441	-86.266614	1.560828e+06	100363	2633	...	36.48391	
2	245683	63	18	46122	317	39.792202	-86.515246	6.956160e+07	284193	6881	...	42.15810	
3	279653	127	72	927	787	18.396103	-66.104169	1.105793e+06	0	2700	...	47.77526	
4	247218	161	20	66502	785	39.195573	-96.569366	2.554403e+06	0	5637	...	24.17693	
...	...	...	...	...	...	...	...	...	...	...	...	...	
26580	279212	43	72	769	787	18.076060	-66.358379	6.970300e+05	0	1847	...	42.73154	
26581	277856	91	42	19422	215	40.158138	-75.307271	5.077337e+06	11786	4155	...	38.21269	
26582	233000	87	8	80653	970	40.410316	-103.814003	1.323262e+09	17577610	2829	...	43.40218	
26583	287425	439	48	76034	817	32.904866	-97.162151	1.865230e+07	158882	11542	...	39.25921	
26584	265371	3	32	89123	702	36.064754	-115.152237	7.796308e+06	0	3726	...	34.45345	

26585 rows × 72 columns

In [141…
```
num_2_cat = ['UID','COUNTYID', 'STATEID', 'zip_code', 'area_code', 'lat', 'lng']
```

In [142…
```
train.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 [143…

```
for col in num_2_cat:
    train[col] = train[col].astype('category')
    test[col] = test[col].astype('category')
```

In [144…

```
print(train.info())
print('-----------')
print(test.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: 19.0+ 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: 7.6+ MB
None
```

In [145…
```
train[cat_variables(train)]
```

Out[145]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2
2	245683	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1
4	247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
26580	279212	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	tract	769	787	18.076060	-66.3
26581	277856	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	tract	19422	215	40.158138	-75.3
26582	233000	87	8	Colorado	CO	Weldona	Saddle Ridge	City	tract	80653	970	40.410316	-103.8
26583	287425	439	48	Texas	TX	Colleyville	Colleyville City	Town	tract	76034	817	32.904866	-97.1
26584	265371	3	32	Nevada	NV	Las Vegas	Paradise	City	tract	89123	702	36.064754	-115.1

26585 rows × 13 columns

In [146…
```
obj_2_cat = ['state', 'state_ab', 'city', 'place', 'type', 'primary']
```

In [147…
```
for col in obj_2_cat:
    train[col] = train[col].astype('category')
    test[col] = test[col].astype('category')
```

In [148…
```
train.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: 18.6 MB
```

In [149…

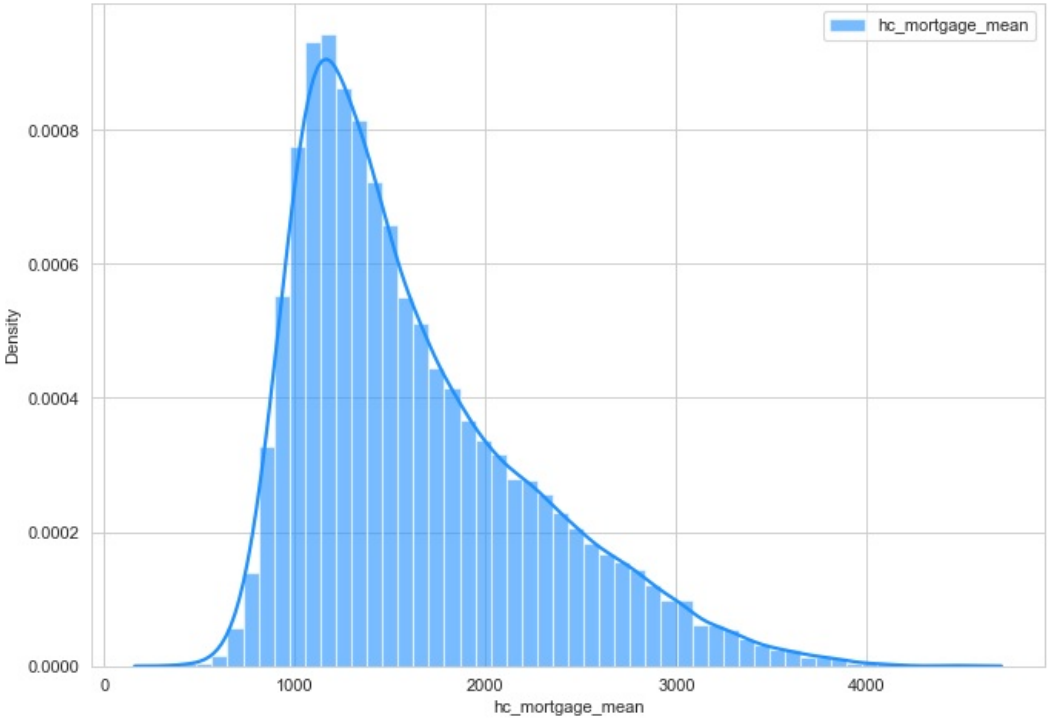
```
train[['hc_mortgage_mean']]
```

Out[149]:

	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 [150…
```
kwargs = dict(hist_kws={'alpha':.6}, kde_kws={'linewidth':2})

plt.figure(figsize=(10,7), dpi= 80)
sns.distplot(train.hc_mortgage_mean, color="dodgerblue", label="hc_mortgage_mean", **kwargs)
plt.legend();
```



Target Variable "hc_mortgage_mean" has a Positive Skew.

In [151…
```
from sklearn.linear_model import LinearRegression
```

In [152…
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, SCORERS
```

In [153…
```
lr = LinearRegression()
```

In [154…
```
def adj_rsqrd(df, r2):
    adj_rsqrd = 1 - (1-r2)*(len(df) - 1) / (len(df) - (df.shape[1] - 1) - 1)
    return round(adj_rsqrd, 3)
```

In [155…
```
cat_cols_2_drop = ['UID', 'state', 'state_ab', 'city', 'place', 'type', 'primary', 'zip_code', 'area_code', 'la
```

In [156…
```
train.drop(cat_cols_2_drop, axis=1, inplace=True)
```

In [157…
```
test.drop(cat_cols_2_drop, axis=1, inplace=True)
```

In [158…
```
train.drop(['COUNTYID', 'STATEID'], axis=1, inplace=True)
```

In [159…
```
test_y = test['hc_mortgage_mean']
```

In [160…
```
test.drop(['COUNTYID', 'STATEID', 'hc_mortgage_mean'], axis=1, inplace=True)
```

In [161…
```
print(train.shape, test.shape)
```

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

In [162…
```
train_X = train.drop(columns=['hc_mortgage_mean'])
train_y = train['hc_mortgage_mean']
```

In [163…
```
lr.fit(train_X, train_y)
```

Out[163]:
```
LinearRegression()
```

In [164…
```
predict_train = lr.predict(train_X)
predict_test = lr.predict(test)
```

In [165…
```
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, 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 [166…
```
import random
randomlist = []
for i in range(0,100):
    n = random.randint(1,len(test))
    randomlist.append(n)
print(randomlist)
```

```
[7206, 8932, 7744, 4638, 4814, 10579, 531, 9875, 6106, 719, 232, 1041, 8256, 5935, 1380, 4871, 6673, 6072, 729,
2729, 1567, 6230, 10206, 3853, 4523, 6741, 1652, 3787, 589, 7402, 6341, 9386, 172, 2245, 2358, 8098, 4615, 5200
, 49, 1885, 3303, 10898, 6047, 7857, 8173, 9256, 7375, 3871, 7477, 3908, 7129, 7920, 6323, 9467, 4209, 3915, 73
73, 6379, 10351, 6974, 3540, 3881, 1074, 2121, 8351, 10839, 3408, 9919, 4515, 2286, 1379, 7686, 8705, 5540, 319
8, 9896, 1137, 6441, 5151, 10909, 5169, 6328, 4469, 9524, 6762, 3020, 3252, 8017, 5341, 7361, 10601, 2798, 1205
, 7317, 1851, 6888, 3973, 342, 9959, 5967]
```

In [167…
```
pre_out = []
out = []

for i in randomlist:
    data_in = [list(test.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)
```

```
7206 [2417.50776858] 2566.80418
8932 [902.94955408] 874.75227
7744 [2381.17677508] 2307.55364
4638 [1598.90819628] 1608.32371
4814 [960.3736031] 962.83068
10579 [1671.1036757] 1620.98242
531 [1266.43847014] 1260.42351
9875 [2428.624757] 2348.76286
6106 [1891.98599763] 1870.86356
719 [1704.49578317] 1731.9755
232 [1488.25325093] 1488.21425
1041 [1637.74782599] 1647.84921
8256 [1095.7794827] 1155.08076
5935 [1822.31769818] 1848.07685
1380 [1177.45072379] 1168.58953
4871 [1295.18352833] 1288.73676
6673 [2053.9544381] 2122.93967
```

```
6072 [1536.3333845] 1517.41049
729 [1194.46947896] 1187.52238
2729 [1493.38211437] 1521.12116
1567 [1721.14054498] 1736.05174
6230 [1001.84660958] 978.75851
10206 [1181.55883763] 1125.16923
3853 [2389.27665352] 2514.64694
4523 [1011.82150958] 1025.5409
6741 [988.16380043] 986.08498
1652 [1267.93234696] 1242.43095
3787 [2001.10148414] 1934.18753
589 [1005.12543097] 1046.55344
7402 [3550.81178356] 3396.71526
6341 [2246.51305108] 2154.11
9386 [2264.89825742] 2275.04507
172 [1520.64338054] 1538.32005
2245 [1666.51173292] 1693.23975
2358 [2450.22820533] 2447.91008
8098 [2373.23839781] 2327.52068
4615 [857.75066331] 854.83709
5200 [1874.71353519] 1854.6625
49 [1793.64874183] 1765.78009
1885 [1502.68190488] 1504.37406
3303 [2529.92964041] 2581.18774
10898 [1456.59970103] 1456.94923
6047 [1055.97882605] 1058.12115
7857 [2525.27954807] 2574.9143
8173 [1165.69746527] 1169.76844
9256 [1157.2736874] 1169.86046
7375 [1917.3416156] 1943.04375
3871 [917.57945781] 886.29183
7477 [2251.71651182] 2208.68045
3908 [1260.08793708] 1261.24826
7129 [1531.71810524] 1536.91144
7920 [2319.91821856] 2337.0859
6323 [2115.0791342] 2101.30215
9467 [1515.47867122] 1566.35898
4209 [927.38521106] 915.28106
3915 [1647.88752325] 1654.9173
7373 [1673.77742695] 1656.14359
6379 [1978.54370158] 1993.36151
10351 [1787.51154772] 1770.73009
6974 [2087.42069431] 2177.70864
3540 [893.77070904] 932.64226
3881 [1031.94825664] 1035.50786
1074 [1831.08499844] 1830.8505
2121 [2087.76173445] 2065.0721
8351 [1088.16315719] 1092.50397
10839 [1310.80904852] 1306.44266
3408 [1052.42161966] 1130.6576
9919 [1553.84651811] 1572.8478
4515 [1023.53747759] 1080.64719
2286 [1138.76326377] 1161.17819
1379 [1055.06957019] 955.0416
7686 [1691.02671818] 1696.93693
8705 [1221.11307371] 1210.98089
5540 [1302.69653023] 1298.21452
3198 [1922.9253142] 2012.58487
9896 [2253.79054502] 2216.98067
1137 [1378.19263711] 1387.14953
6441 [1806.3577762] 1827.7902
5151 [1544.84240907] 1555.84307
10909 [1360.94131511] 1283.4328
5169 [983.74426747] 1026.39541
6328 [3074.96386285] 3050.71119
4469 [1660.59814905] 1646.65575
9524 [2202.95741704] 2131.48222
6762 [1991.25461244] 1991.70409
3020 [1760.77899287] 1761.66237
3252 [2410.6326154] 2515.02947
8017 [820.19492844] 764.60873
5341 [2240.66152276] 2270.39927
7361 [1999.67605672] 2044.0433
10601 [1732.66724719] 1794.18839
2798 [2007.14745462] 1973.74047
1205 [1006.35121633] 997.42279
7317 [988.76371295] 940.76062
1851 [764.27939536] 788.99863
6888 [1100.18459082] 1097.52649
3973 [1819.54690664] 1851.28963
342 [1344.11221525] 1303.03434
9959 [1568.39947006] 1556.29254
5967 [1419.13157411] 1329.62193
```

In [168…
```
pre_out
```

Out[168]:
```
[array([2417.50776858]),
 array([902.94955408]),
 array([2381.17677508]),
```

```
array([1598.90819628]),
array([960.3736031]),
array([1671.1036757]),
array([1266.43847014]),
array([2428.624757]),
array([1891.98599763]),
array([1704.49578317]),
array([1488.25325093]),
array([1637.74782599]),
array([1095.7794827]),
array([1822.31769818]),
array([1177.45072379]),
array([1295.18352833]),
array([2053.9544381]),
array([1536.3333845]),
array([1194.46947896]),
array([1493.38211437]),
array([1721.14054498]),
array([1001.84660958]),
array([1181.55883763]),
array([2389.27665352]),
array([1011.82150958]),
array([988.16380043]),
array([1267.93234696]),
array([2001.10148414]),
array([1005.12543097]),
array([3550.81178356]),
array([2246.51305108]),
array([2264.89825742]),
array([1520.64338054]),
array([1666.51173292]),
array([2450.22820533]),
array([2373.23839781]),
array([857.75066331]),
array([1874.71353519]),
array([1793.64874183]),
array([1502.68190488]),
array([2529.92964041]),
array([1456.59970103]),
array([1055.97882605]),
array([2525.27954807]),
array([1165.69746527]),
array([1157.2736874]),
array([1917.3416156]),
array([917.57945781]),
array([2251.71651182]),
array([1260.08793708]),
array([1531.71810524]),
array([2319.91821856]),
array([2115.0791342]),
array([1515.47867122]),
array([927.38521106]),
array([1647.88752325]),
array([1673.77742695]),
array([1978.54370158]),
array([1787.51154772]),
array([2087.42069431]),
array([893.77070904]),
array([1031.94825664]),
array([1831.08499844]),
array([2087.76173445]),
array([1088.16315719]),
array([1310.80904852]),
array([1052.42161966]),
array([1553.84651811]),
array([1023.53747759]),
array([1138.76326377]),
array([1055.06957019]),
array([1691.02671818]),
array([1221.11307371]),
array([1302.69653023]),
array([1922.9253142]),
array([2253.79054502]),
array([1378.19263711]),
array([1806.3577762]),
array([1544.84240907]),
array([1360.94131511]),
array([983.74426747]),
array([3074.96386285]),
array([1660.59814905]),
array([2202.95741704]),
array([1991.25461244]),
array([1760.77899287]),
array([2410.6326154]),
array([820.19492844]),
array([2240.66152276]),
array([1999.67605672]),
array([1732.66724719]),
array([2007.14745462]),
```

```
 array([1006.35121633]),
 array([988.76371295]),
 array([764.27939536]),
 array([1100.18459082]),
 array([1819.54690664]),
 array([1344.11221525]),
 array([1568.39947006]),
 array([1419.13157411])]
```

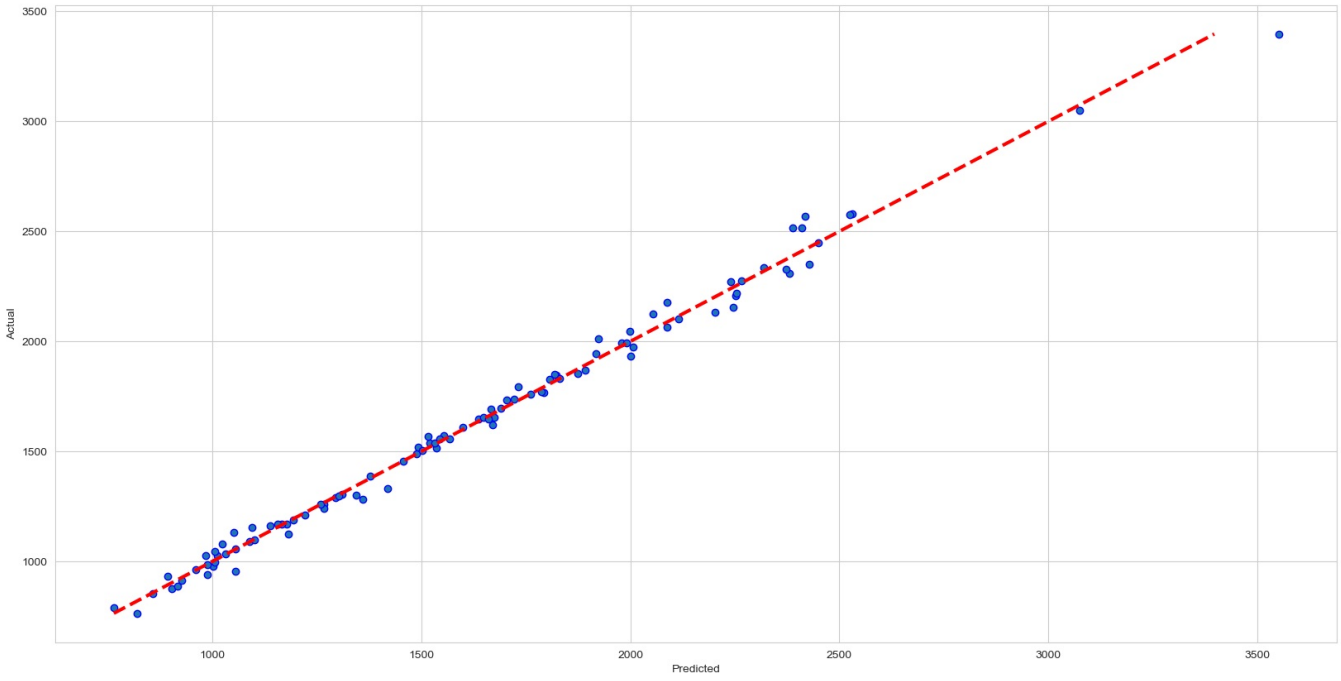
In [169…
```python
x = [2,3,5,9,1,0,2,3]

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 [170…
```python
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 [171…
```python
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 [172…
```python
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, 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
```