In [2]:
```
'''
DESCRIPTION:

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

'''
```

Out[2]:
```
'\nDESCRIPTION:\n\n* A banking institution requires actionable insights into mortgage-backed securities, geographic business investment, and \nreal estate analysis. \n* The mortgage bank would like to identify potential monthly mortgage expenses for each region based on monthly family income\nand rental of the real estate.\n* A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region \nin the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies.\n* The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental \nincome, monthly mortgage and owner’s cost, family income vs mortgage cost comparison across different regions. The metrics \ndescribed here do not limit the dashboard to these few.\n\n'
```

In [3]:
```
'''
Data Import and Preparation:

Import data.
Figure out the primary key and look for the requirement of indexing.
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.
'''
```

Out[3]:
```
'\nData Import and Preparation:\n\nImport data.\nFigure out the primary key and look for the requirement of indexing.\nGauge the fill rate of the variables and devise plans for missing value treatment. \nPlease explain explicitly the reason for the treatment chosen for each variable.\n'
```

In [4]:
```
# Importing required libraries and dataset
import pandas as pd
train_url = "https://raw.githubusercontent.com/PraveenBandla/Data-Science-Projects/master/Data-Science-Capstone-Projects/Project%201/train.csv"
real_train_df = pd.read_csv(train_url)
real_train_df
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	marrie
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.5785
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.58333	23.43353	267.23367	1284.0	0.52483	0.3488
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.83333	23.94119	707.01963	3238.0	0.85331	0.6474
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.58333	24.32015	362.20193	1559.0	0.65037	0.4725
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.58333	11.10484	1854.48652	3051.0	0.13046	0.1235
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
27316	279212	NaN	140	43	72	Puerto Rico	PR	Coamo	Coamo	Urban	...	42.73154	40.16667	24.79821	230.87898	938.0	0.60422	0.2460
27317	277856	NaN	140	91	42	Pennsylvania	PA	Blue Bell	Blue Bell	Borough	...	38.21269	39.50000	21.84826	496.20427	2039.0	0.68072	0.6112
27318	233000	NaN	140	87	8	Colorado	CO	Weldona	Saddle Ridge	City	...	43.40218	46.33333	23.40858	316.52078	1364.0	0.78508	0.7045
27319	287425	NaN	140	439	48	Texas	TX	Colleyville	Colleyville City	Town	...	39.25921	43.41667	21.36235	1373.94120	5815.0	0.93970	0.7550
27320	265371	NaN	140	3	32	Nevada	NV	Las Vegas	Paradise	City	...	34.45345	29.83333	19.77208	526.73261	1911.0	0.27912	0.3442

27321 rows × 80 columns

In [5]:
```python
# Data Preparation
# Figure out the primary key and look for the requirement of indexing.
real_train_df.info()
# Observations:
# We can consider UID (location ID) as the primary key as it is unique across different locations
# UID can be moved to dataframe index since it is used for identifying a record and not for prediction
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27321 entries, 0 to 27320
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   UID                          27321 non-null  int64  
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     27321 non-null  int64  
 3   COUNTYID                     27321 non-null  int64  
 4   STATEID                      27321 non-null  int64  
 5   state                        27321 non-null  object 
 6   state_ab                     27321 non-null  object 
 7   city                         27321 non-null  object 
 8   place                        27321 non-null  object 
 9   type                         27321 non-null  object 
 10  primary                      27321 non-null  object 
 11  zip_code                     27321 non-null  int64  
 12  area_code                    27321 non-null  int64  
 13  lat                          27321 non-null  float64
 14  lng                          27321 non-null  float64
 15  ALand                        27321 non-null  float64
 16  AWater                       27321 non-null  int64  
 17  pop                          27321 non-null  int64  
 18  male_pop                     27321 non-null  int64  
 19  female_pop                   27321 non-null  int64  
 20  rent_mean                    27007 non-null  float64
 21  rent_median                  27007 non-null  float64
 22  rent_stdev                   27007 non-null  float64
 23  rent_sample_weight           27007 non-null  float64
 24  rent_samples                 27007 non-null  float64
 25  rent_gt_10                   27007 non-null  float64
 26  rent_gt_15                   27007 non-null  float64
 27  rent_gt_20                   27007 non-null  float64
 28  rent_gt_25                   27007 non-null  float64
 29  rent_gt_30                   27007 non-null  float64
 30  rent_gt_35                   27007 non-null  float64
 31  rent_gt_40                   27007 non-null  float64
 32  rent_gt_50                   27007 non-null  float64
 33  universe_samples             27321 non-null  int64  
 34  used_samples                 27321 non-null  int64  
 35  hi_mean                      27053 non-null  float64
 36  hi_median                    27053 non-null  float64
 37  hi_stdev                     27053 non-null  float64
 38  hi_sample_weight             27053 non-null  float64
 39  hi_samples                   27053 non-null  float64
 40  family_mean                  27023 non-null  float64
 41  family_median                27023 non-null  float64
 42  family_stdev                 27023 non-null  float64
 43  family_sample_weight         27023 non-null  float64
 44  family_samples               27023 non-null  float64
 45  hc_mortgage_mean             26748 non-null  float64
 46  hc_mortgage_median           26748 non-null  float64
 47  hc_mortgage_stdev            26748 non-null  float64
 48  hc_mortgage_sample_weight    26748 non-null  float64
```

```
 49  hc_mortgage_samples          26748 non-null  float64
 50  hc_mean                      26721 non-null  float64
 51  hc_median                    26721 non-null  float64
 52  hc_stdev                     26721 non-null  float64
 53  hc_samples                   26721 non-null  float64
 54  hc_sample_weight             26721 non-null  float64
 55  home_equity_second_mortgage  26864 non-null  float64
 56  second_mortgage              26864 non-null  float64
 57  home_equity                  26864 non-null  float64
 58  debt                         26864 non-null  float64
 59  second_mortgage_cdf          26864 non-null  float64
 60  home_equity_cdf              26864 non-null  float64
 61  debt_cdf                     26864 non-null  float64
 62  hs_degree                    27131 non-null  float64
 63  hs_degree_male               27121 non-null  float64
 64  hs_degree_female             27098 non-null  float64
 65  male_age_mean                27132 non-null  float64
 66  male_age_median              27132 non-null  float64
 67  male_age_stdev               27132 non-null  float64
 68  male_age_sample_weight       27132 non-null  float64
 69  male_age_samples             27132 non-null  float64
 70  female_age_mean              27115 non-null  float64
 71  female_age_median            27115 non-null  float64
 72  female_age_stdev             27115 non-null  float64
 73  female_age_sample_weight     27115 non-null  float64
 74  female_age_samples           27115 non-null  float64
 75  pct_own                      27053 non-null  float64
 76  married                      27130 non-null  float64
 77  married_snp                  27130 non-null  float64
 78  separated                    27130 non-null  float64
 79  divorced                     27130 non-null  float64
dtypes: float64(62), int64(12), object(6)
memory usage: 16.7+ MB
```

In [6]:
```python
#Checking for duplicates
real_train_df[real_train_df.duplicated()]
# Observations:
# 160 duplicate records are present in our dataset
```

Out[6]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married
1623	230058	NaN	140	73	6	California	CA	Oceanside	Camp Pendleton North	City	...	19.99315	22.41667	11.62088	3406.53918	11492.0	0.00107	0.33566
1907	292484	NaN	140	25	55	Wisconsin	WI	Madison	Madison City	City	...	22.03226	21.08333	5.13435	1365.86300	1981.0	0.00000	0.00773
2447	268401	NaN	140	61	36	New York	NY	Long Island City	New York City	City	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4161	284060	NaN	140	113	48	Texas	TX	Dallas	University Park City	Town	...	35.57082	31.00000	14.89626	248.71488	1066.0	0.00419	0.39327
5066	274254	NaN	140	109	40	Oklahoma	OK	Oklahoma City	Oklahoma City City	CDP	...	36.16616	29.83333	13.14478	23.22171	113.0	0.00000	0.22881
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26769	252187	NaN	140	33	24	Maryland	MD	Morningside	Andrews Afb	CDP	...	21.92741	22.50000	15.50144	375.14523	1687.0	0.00000	0.78735
26872	293566	NaN	140	133	55	Wisconsin	WI	Brookfield	Pewaukee City	City	...	39.92907	44.33333	22.25252	593.35393	2424.0	0.99468	0.77148
26910	222470	NaN	140	11	4	Arizona	AZ	Morenci	Clifton	CDP	...	28.24603	27.83333	17.42918	392.61849	1710.0	0.00517	0.46198
27175	235725	NaN	140	57	12	Florida	FL	Tampa	Pebble Creek	City	...	29.08800	28.08333	14.65116	144.78344	648.0	0.00000	0.25806
27176	247777	NaN	140	61	21	Kentucky	KY	Brownsville	Brownsville City	City	...	19.39847	19.00000	1.49474	3.39130	6.0	NaN	0.00000

160 rows × 80 columns

In [7]:
```python
#Removing duplicates
real_train_df = real_train_df.drop_duplicates()
real_train_df.shape
```

Out[7]: (27161, 80)

In [8]:
```python
# 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.
# Checking for NULL Values
real_train_df.isna().sum()
# Observations:
# Block ID is NULL for all the records => It can be removed from the dataset
# There are NULL Values across rent, income, mortgage, equity, age and marital status columns
# Since the NULL values are only ~2%, We can drop the records with NULL Values
```

Out[8]:
```
UID                0
BLOCKID        27161
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...  
pct_own          207
married          150
married_snp      150
separated        150
divorced         150
Length: 80, dtype: int64
```

In [9]:
```python
# Moving UID to Index
real_train_df.index = real_train_df['UID']
real_train_df.head()
```

Out[9]:

UID	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	ma
267822	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.57851	
246444	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.58333	23.43353	267.23367	1284.0	0.52483	0.34886	
245683	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.83333	23.94119	707.01963	3238.0	0.85331	0.64745	
279653	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.58333	24.32015	362.20193	1559.0	0.65037	0.47257	
247218	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.58333	11.10484	1854.48652	3051.0	0.13046	0.12356	

5 rows × 80 columns

In [10]:
```python
# Dropping UID, BlockID columns
real_train_df.drop(['UID','BLOCKID','SUMLEVEL'],axis=1,inplace=True)
real_train_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\pandas\core\frame.py:4312: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[10]: (27161, 77)

In [11]:
```
# Checking columns with a constant value
real_train_df.columns[real_train_df.nunique()<=1]
```

Out[11]:
```
Index(['primary'], dtype='object')
```

In [12]:
```
# Dropping columns with constant value
real_train_df.drop(real_train_df.columns[real_train_df.nunique()<=1],axis=1,inplace=True)
real_train_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\pandas\core\frame.py:4312: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[12]:
```
(27161, 76)
```

In [13]:
```
# Dropping records with NULL Values
real_train_df.dropna(inplace=True)
real_train_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy)
```

Out[13]:
```
(26585, 76)
```

In [14]:
```python
# Importing the test dataset
import pandas as pd
test_url = "https://raw.githubusercontent.com/PraveenBandla/Data-Science-Projects/master/Data-Science-Capstone-Projects/Project%201/test.csv"
real_test_df = pd.read_csv(test_url)
real_test_df
```

Out[14]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	marri
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...	34.78682	33.75000	21.58531	416.48097	1938.0	0.70252	0.282
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...	44.23451	46.66667	22.37036	532.03505	1950.0	0.85128	0.642
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...	41.62426	44.50000	22.86213	453.11959	1879.0	0.81897	0.599
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...	44.81200	48.00000	21.03155	263.94320	1081.0	0.84609	0.569
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...	40.66618	42.66667	21.30900	709.90829	2956.0	0.79077	0.576
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11704	238088	NaN	140	105	12	Florida	FL	Lakeland	Crystal Springs	City	...	53.51255	59.58333	23.23426	699.33353	2914.0	0.93121	0.659
11705	242811	NaN	140	31	17	Illinois	IL	Chicago	Chicago City	Village	...	33.14169	32.83333	20.24698	306.63915	1191.0	0.33122	0.428
11706	250127	NaN	140	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	...	43.53905	43.66667	23.17995	900.13903	3723.0	0.84372	0.502
11707	241096	NaN	140	27	19	Iowa	IA	Carroll	Carroll City	City	...	45.63179	48.16667	24.84209	693.82905	3213.0	0.83330	0.666
11708	287763	NaN	140	453	48	Texas	TX	Austin	Sunset Valley City	Town	...	35.99955	35.41667	20.68049	559.30291	2047.0	0.52587	0.519

11709 rows × 80 columns

In [15]:
```
#Data Preparation for test dataset
real_test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11709 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   UID                         11709 non-null  int64
 1   BLOCKID                     0 non-null      float64
 2   SUMLEVEL                    11709 non-null  int64
 3   COUNTYID                    11709 non-null  int64
 4   STATEID                     11709 non-null  int64
 5   state                       11709 non-null  object
 6   state_ab                    11709 non-null  object
 7   city                        11709 non-null  object
 8   place                       11709 non-null  object
 9   type                        11709 non-null  object
 10  primary                     11709 non-null  object
 11  zip_code                    11709 non-null  int64
 12  area_code                   11709 non-null  int64
 13  lat                         11709 non-null  float64
 14  lng                         11709 non-null  float64
 15  ALand                       11709 non-null  int64
 16  AWater                      11709 non-null  int64
 17  pop                         11709 non-null  int64
 18  male_pop                    11709 non-null  int64
 19  female_pop                  11709 non-null  int64
 20  rent_mean                   11561 non-null  float64
 21  rent_median                 11561 non-null  float64
 22  rent_stdev                  11561 non-null  float64
 23  rent_sample_weight          11561 non-null  float64
 24  rent_samples                11561 non-null  float64
 25  rent_gt_10                  11560 non-null  float64
 26  rent_gt_15                  11560 non-null  float64
 27  rent_gt_20                  11560 non-null  float64
 28  rent_gt_25                  11560 non-null  float64
 29  rent_gt_30                  11560 non-null  float64
 30  rent_gt_35                  11560 non-null  float64
 31  rent_gt_40                  11560 non-null  float64
 32  rent_gt_50                  11560 non-null  float64
 33  universe_samples            11709 non-null  int64
 34  used_samples                11709 non-null  int64
 35  hi_mean                     11587 non-null  float64
 36  hi_median                   11587 non-null  float64
 37  hi_stdev                    11587 non-null  float64
 38  hi_sample_weight            11587 non-null  float64
 39  hi_samples                  11587 non-null  float64
 40  family_mean                 11573 non-null  float64
 41  family_median               11573 non-null  float64
 42  family_stdev                11573 non-null  float64
 43  family_sample_weight        11573 non-null  float64
 44  family_samples              11573 non-null  float64
 45  hc_mortgage_mean            11441 non-null  float64
 46  hc_mortgage_median          11441 non-null  float64
 47  hc_mortgage_stdev           11441 non-null  float64
 48  hc_mortgage_sample_weight   11441 non-null  float64
 49  hc_mortgage_samples         11441 non-null  float64
 50  hc_mean                     11419 non-null  float64
 51  hc_median                   11419 non-null  float64
 52  hc_stdev                    11419 non-null  float64
```

```
 53  hc_samples                  11419 non-null  float64
 54  hc_sample_weight            11419 non-null  float64
 55  home_equity_second_mortgage 11489 non-null  float64
 56  second_mortgage             11489 non-null  float64
 57  home_equity                 11489 non-null  float64
 58  debt                        11489 non-null  float64
 59  second_mortgage_cdf         11489 non-null  float64
 60  home_equity_cdf             11489 non-null  float64
 61  debt_cdf                    11489 non-null  float64
 62  hs_degree                   11624 non-null  float64
 63  hs_degree_male              11620 non-null  float64
 64  hs_degree_female            11604 non-null  float64
 65  male_age_mean               11625 non-null  float64
 66  male_age_median             11625 non-null  float64
 67  male_age_stdev              11625 non-null  float64
 68  male_age_sample_weight      11625 non-null  float64
 69  male_age_samples            11625 non-null  float64
 70  female_age_mean             11613 non-null  float64
 71  female_age_median           11613 non-null  float64
 72  female_age_stdev            11613 non-null  float64
 73  female_age_sample_weight    11613 non-null  float64
 74  female_age_samples          11613 non-null  float64
 75  pct_own                     11587 non-null  float64
 76  married                     11625 non-null  float64
 77  married_snp                 11625 non-null  float64
 78  separated                   11625 non-null  float64
 79  divorced                    11625 non-null  float64
dtypes: float64(61), int64(13), object(6)
memory usage: 7.1+ MB
```

In [16]:
```
# Checking for duplicates
real_test_df[real_test_df.duplicated()].shape
# Observations:
# 32 duplicate records are present in our dataset
```

Out[16]:
```
(32, 80)
```

In [17]:
```
#Removing duplicates
real_test_df = real_test_df.drop_duplicates()
real_test_df.shape
```

Out[17]:
```
(11677, 80)
```

In [18]:
```python
#Checking for NULL values
real_test_df.isna().sum()
# Observations:
# Block ID is NULL for all the records => It can be removed from the dataset
# There are NULL Values across rent, income, mortgage, equity, age and marital status columns
# Since the NULL values are only ~2%, We can drop the records with NULL Values
```

Out[18]:
```
UID                0
BLOCKID        11677
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...
pct_own          112
married           77
married_snp       77
separated         77
divorced          77
Length: 80, dtype: int64
```

In [19]:
```python
# Moving UID to Index
real_test_df.set_index('UID')
real_test_df.head()
```

Out[19]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	m
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...	34.78682	33.75000	21.58531	416.48097	1938.0	0.70252	0.28217	
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...	44.23451	46.66667	22.37036	532.03505	1950.0	0.85128	0.64221	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...	41.62426	44.50000	22.86213	453.11959	1879.0	0.81897	0.59961	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...	44.81200	48.00000	21.03155	263.94320	1081.0	0.84609	0.56953	
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...	40.66618	42.66667	21.30900	709.90829	2956.0	0.79077	0.57620	

5 rows × 80 columns

In [20]:
```python
# Dropping UID and BlockID columns
real_test_df.drop(['UID','BLOCKID'],axis=1,inplace=True)
real_test_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\pandas\core\frame.py:4312: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[20]: (11677, 78)

In [21]:
```python
# Checking for columns with constant value
real_test_df.columns[real_test_df.nunique()<=1]
```

Out[21]: Index(['SUMLEVEL', 'primary'], dtype='object')

In [22]:
```
# Dropping columns with constant value
real_test_df.drop(real_test_df.columns[real_test_df.nunique()<=1],axis=1,inplace=True)
real_test_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\pandas\core\frame.py:4312: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[22]:
```
(11677, 76)
```

In [23]:
```
# Dropping records with NULL Values
real_test_df.dropna(inplace=True)
real_test_df.shape
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy)
```

Out[23]:
```
(11355, 76)
```

In [24]:
```
'''
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

b) Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage

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

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

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

Out[24]:
```
'\nExploratory Data Analysis (EDA):\n\n4.Perform debt analysis. You may take the following steps:\n\na) Explore the top 2,500 locations where the percentage of households with a second mortgage i
s the highest and percent ownership is above 10 percent. Visualize using geo-map. \n   You may keep the upper limit for the percent of households with a second mortgage to 50 percent\n\nb) Use th
e following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage \n\nc) Create pie charts to show overall de
bt and bad debt\n\nd) Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities\n\ne) Create a collated income distribut
ion chart for family income, house hold income, and remaining income\n'
```

In [25]:
```
# Combining train and test datasets for EDA
real_df = real_train_df.append(real_test_df)
real_df
```

Out[25]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	mar
267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	...	44.48629	45.33333	22.51276	685.33845	2618.0	0.79046	0.57
246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	...	36.48391	37.58333	23.43353	267.23367	1284.0	0.52483	0.34
245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	...	42.15810	42.83333	23.94119	707.01963	3238.0	0.85331	0.64
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	...	47.77526	50.58333	24.32015	362.20193	1559.0	0.65037	0.47
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	...	24.17693	21.58333	11.10484	1854.48652	3051.0	0.13046	0.12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11704	105	12	Florida	FL	Lakeland	Crystal Springs	City	33810	863	28.226068	...	53.51255	59.58333	23.23426	699.33353	2914.0	0.93121	0.65
11705	31	17	Illinois	IL	Chicago	Chicago City	Village	60609	773	41.804936	...	33.14169	32.83333	20.24698	306.63915	1191.0	0.33122	0.42
11706	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	1841	978	42.737778	...	43.53905	43.66667	23.17995	900.13903	3723.0	0.84372	0.50
11707	27	19	Iowa	IA	Carroll	Carroll City	City	51401	712	42.081366	...	45.63179	48.16667	24.84209	693.82905	3213.0	0.83330	0.66
11708	453	48	Texas	TX	Austin	Sunset Valley City	Town	78745	512	30.219013	...	35.99955	35.41667	20.68049	559.30291	2047.0	0.52587	0.51

37940 rows × 76 columns

In [26]:
```
# 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
geo_df = real_df[real_df.home_equity > 0.1].sort_values('second_mortgage',ascending=False).head(2500)
geo_df
```

Out[26]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	female_age_mean	female_age_median	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own
222830	13	4	Arizona	AZ	Scottsdale	Tempe City	CDP	85257	480	33.458658	...	31.91429	30.83333	14.45269	229.39846	981.0	0.05660
251185	27	25	Massachusetts	MA	Worcester	Worcester City	City	1610	508	42.254262	...	30.60147	26.16667	19.21553	262.09529	994.0	0.20247
278178	101	42	Pennsylvania	PA	Philadelphia	Millbourne	Borough	19104	215	39.952954	...	22.42708	21.08333	7.39823	2280.04214	3446.0	0.05041
9088	33	22	Louisiana	LA	Baton Rouge	Port Allen City	City	70802	225	30.414676	...	23.22094	21.50000	8.51933	711.05609	1640.0	0.03976
287621	453	48	Texas	TX	Austin	Austin City	Town	78705	512	30.285534	...	21.71204	20.50000	5.97345	2119.00876	3203.0	0.01737
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1501	31	12	Florida	FL	Jacksonville	Orange Park	City	32257	904	30.180957	...	45.61859	46.16667	24.67458	428.88514	1722.0	0.61322
278643	125	42	Pennsylvania	PA	Canonsburg	Canonsburg	Borough	15317	724	40.259249	...	43.40131	44.50000	23.89332	344.10155	1572.0	0.58465
10563	31	17	Illinois	IL	Cicero	Cicero	Village	60804	708	41.834200	...	29.09346	25.25000	19.76185	679.19639	2650.0	0.56228
3017	121	13	Georgia	GA	Atlanta	Hapeville City	City	30310	404	33.693486	...	39.54266	39.08333	20.79632	448.10167	1807.0	0.27609
7843	163	26	Michigan	MI	Woodhaven	Woodhaven City	CDP	48183	734	42.146180	...	42.87377	43.25000	21.33776	467.16762	1911.0	0.80903

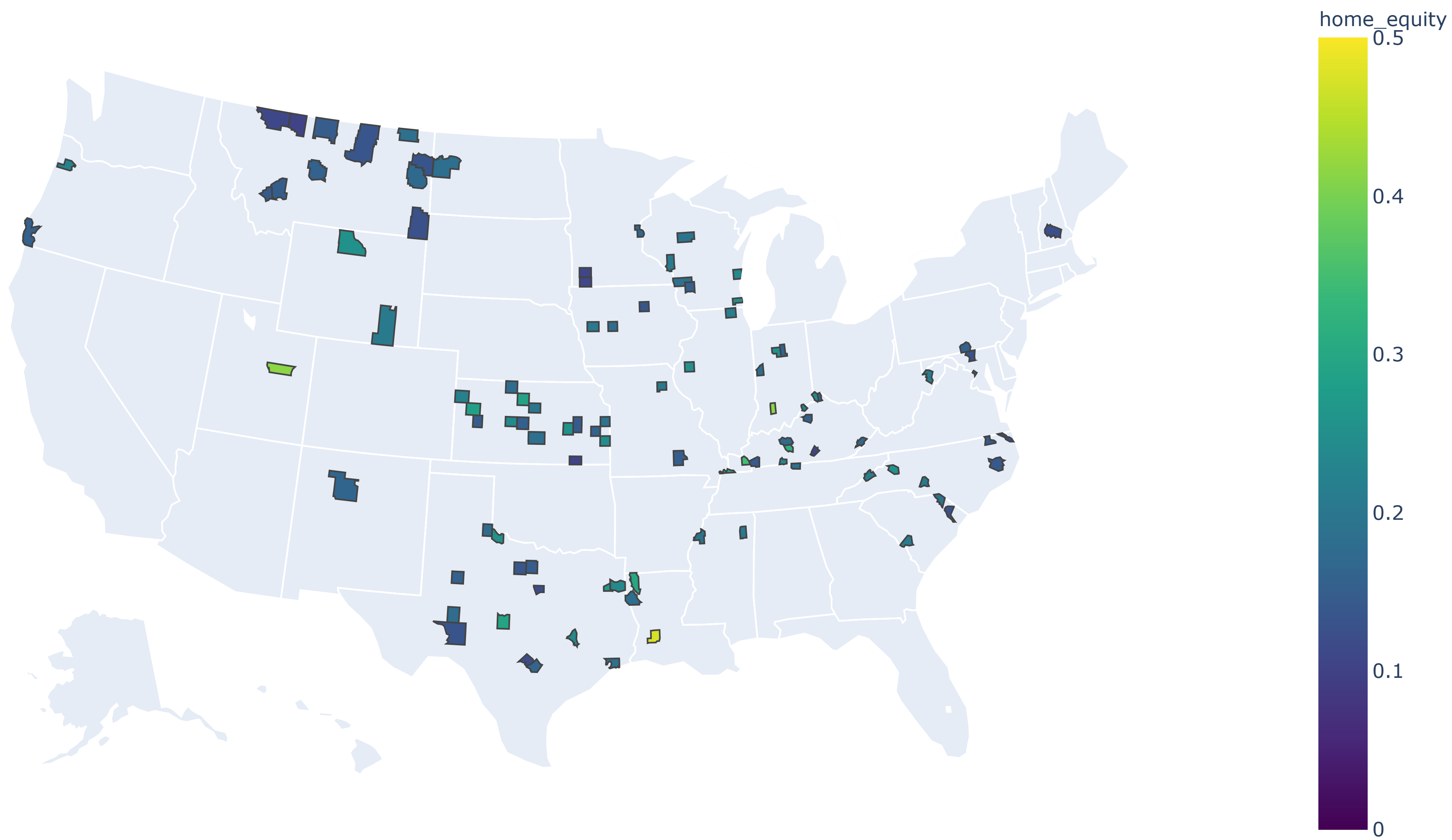
2500 rows × 76 columns

In [27]:
```
!pip install -U plotly
```

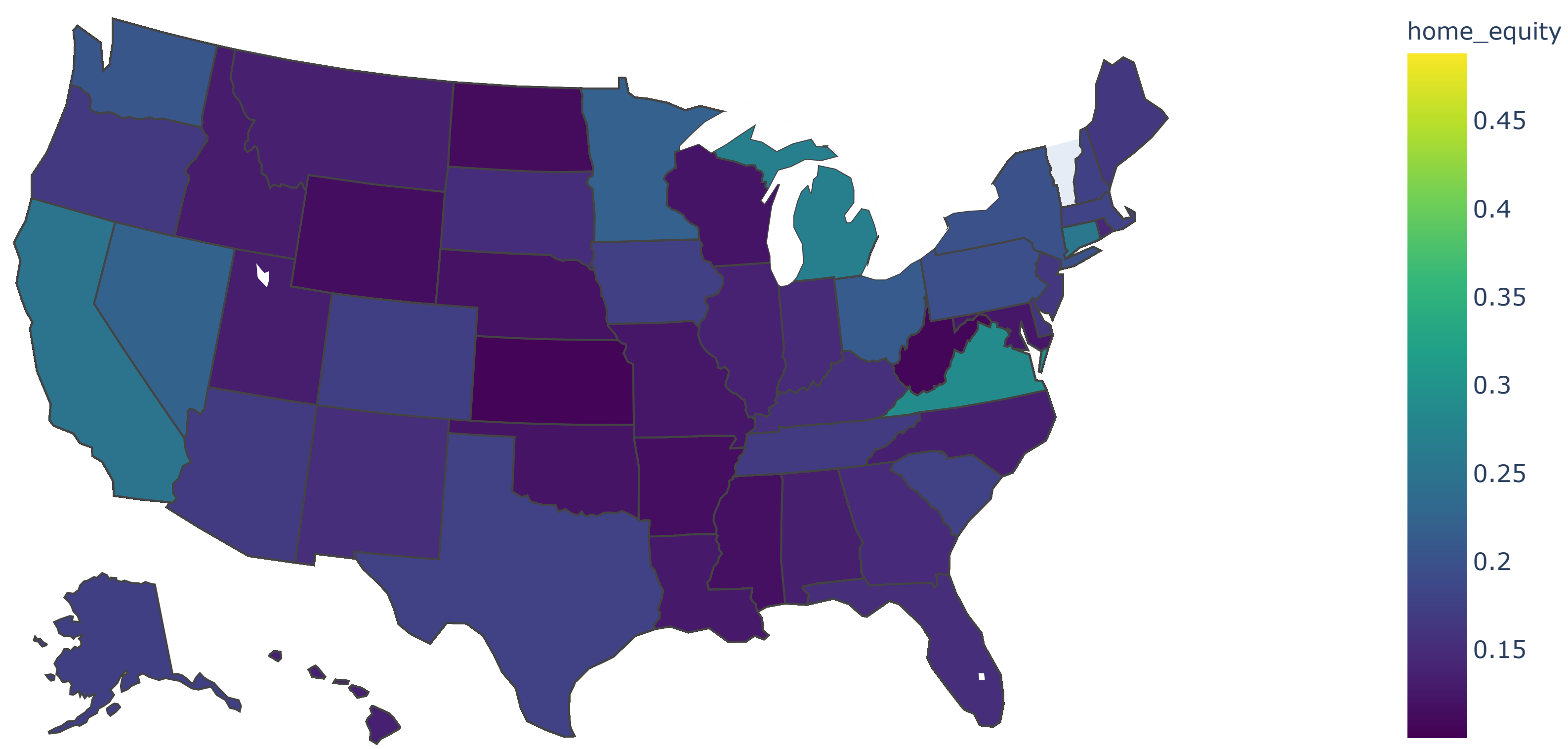
```
Requirement already satisfied: plotly in c:\users\bpk20\anaconda3\lib\site-packages (4.13.0)
Collecting plotly
  Downloading plotly-4.14.3-py2.py3-none-any.whl (13.2 MB)
Requirement already satisfied: retrying>=1.3.3 in c:\users\bpk20\anaconda3\lib\site-packages (from plotly) (1.3.3)
Requirement already satisfied: six in c:\users\bpk20\anaconda3\lib\site-packages (from plotly) (1.15.0)
Installing collected packages: plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 4.13.0
    Uninstalling plotly-4.13.0:
      Successfully uninstalled plotly-4.13.0
Successfully installed plotly-4.14.3
```

In [28]:

```
# Geo Map for top 2500 Locations --By zip-code
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

import plotly.express as px

fig = px.choropleth(geo_df, geojson=counties, locations='zip_code', color='home_equity',
                           color_continuous_scale="Viridis",
                           range_color=(0, 0.5),
                           hover_name='state',
                           hover_data=['second_mortgage'],
                           scope="usa"
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



In [29]:
```
#Geo Map for top 2500 locations --By State
import plotly.express as px

fig = px.choropleth(geo_df, locations="state_ab",
                    locationmode="USA-states",
                    color="home_equity",
                    hover_name="state",
                    hover_data=['second_mortgage'],
                    scope='usa',
                    color_continuous_scale="Viridis")
fig.show()
```



In [30]:
```python
# b) Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage
# c) Create pie charts to show overall debt and bad debt
real_df['bad_debt'] = real_df.second_mortgage + real_df.home_equity - real_df.home_equity_second_mortgage
real_df['good_debt'] = real_df.debt - real_df.bad_debt
real_df
```

Out[30]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced
267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	...	22.51276	685.33845	2618.0	0.79046	0.57851	0.01882	0.01240	0.08770
246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	...	23.43353	267.23367	1284.0	0.52483	0.34886	0.01426	0.01426	0.09030
245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	...	23.94119	707.01963	3238.0	0.85331	0.64745	0.02830	0.01607	0.10657
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	...	24.32015	362.20193	1559.0	0.65037	0.47257	0.02021	0.02021	0.10106
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	...	11.10484	1854.48652	3051.0	0.13046	0.12356	0.00000	0.00000	0.03109
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11704	105	12	Florida	FL	Lakeland	Crystal Springs	City	33810	863	28.226068	...	23.23426	699.33353	2914.0	0.93121	0.65969	0.02135	0.02135	0.08780
11705	31	17	Illinois	IL	Chicago	Chicago City	Village	60609	773	41.804936	...	20.24698	306.63915	1191.0	0.33122	0.42882	0.07781	0.02829	0.05305
11706	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	1841	978	42.737778	...	23.17995	900.13903	3723.0	0.84372	0.50269	0.00108	0.00108	0.07294
11707	27	19	Iowa	IA	Carroll	Carroll City	City	51401	712	42.081366	...	24.84209	693.82905	3213.0	0.83330	0.66699	0.02738	0.00000	0.04694
11708	453	48	Texas	TX	Austin	Sunset Valley City	Town	78745	512	30.219013	...	20.68049	559.30291	2047.0	0.52587	0.51922	0.08066	0.02520	0.10586

37940 rows × 78 columns

In [31]:
```
#Calculating number of households with bad debt
debt_df = real_df[['type','debt','good_debt','bad_debt','hi_samples']]
debt_df['debt_num'] = debt_df.debt * debt_df.hi_samples
debt_df['good_debt_num'] = debt_df.good_debt * debt_df.hi_samples
debt_df['bad_debt_num'] = debt_df.bad_debt * debt_df.hi_samples
debt_df
```

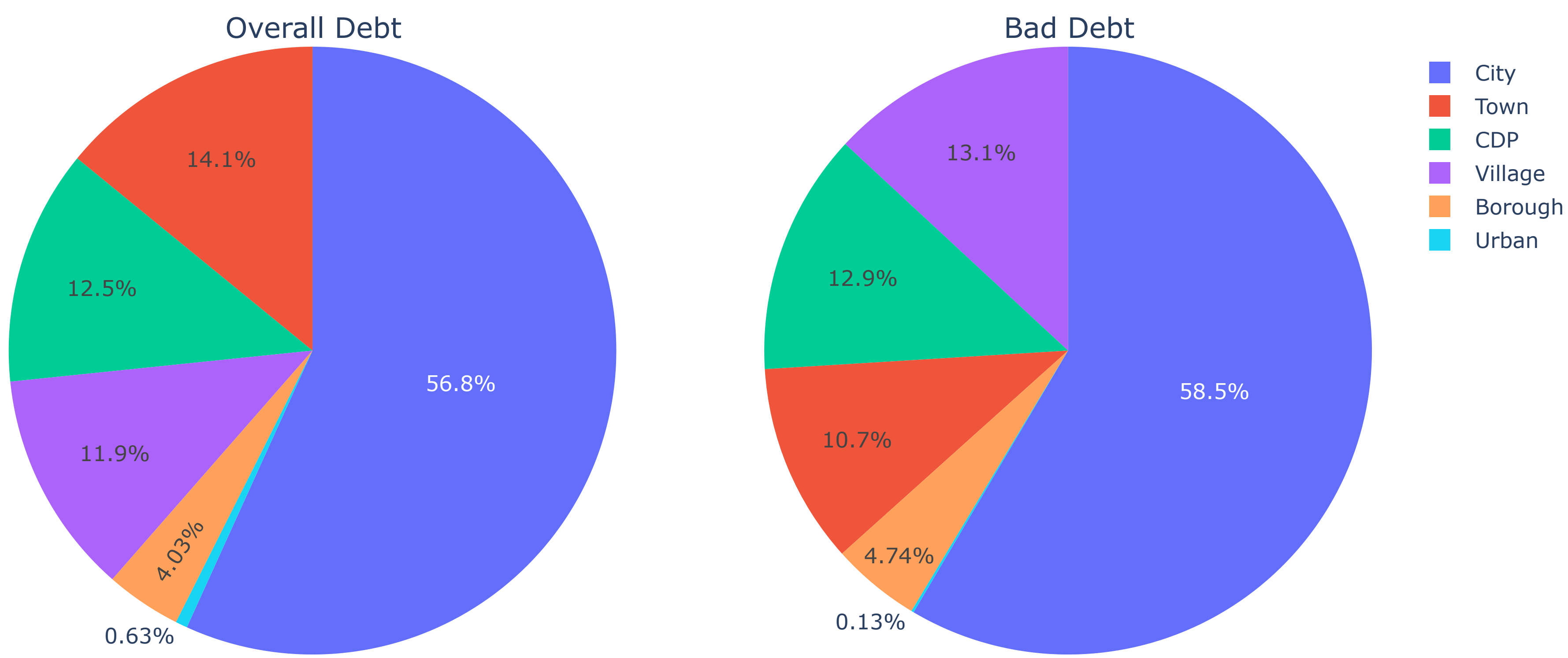
```
C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[31]:

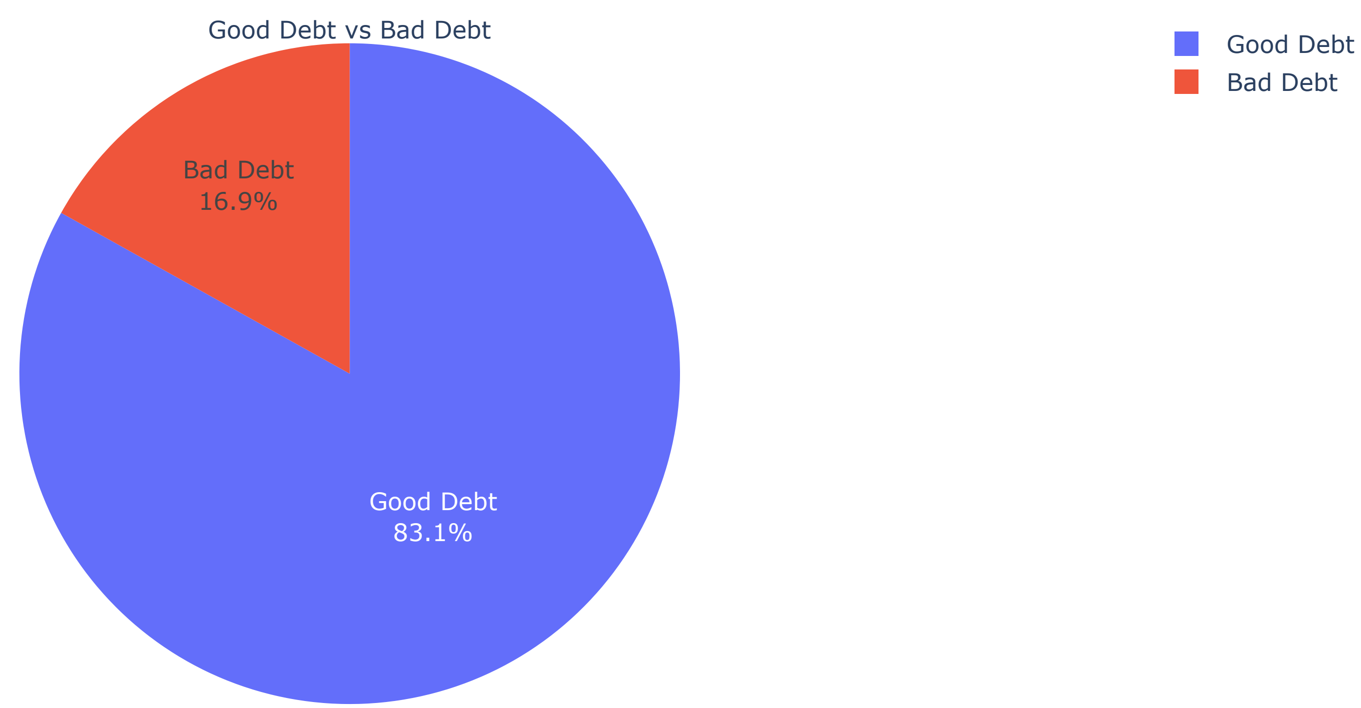
	type	debt	good_debt	bad_debt	hi_samples	debt_num	good_debt_num	bad_debt_num
267822	City	0.52963	0.43555	0.09408	2024.0	1071.97112	881.55320	190.41792
246444	City	0.60855	0.56581	0.04274	1127.0	685.83585	637.66787	48.16798
245683	City	0.73484	0.63972	0.09512	2488.0	1828.28192	1591.62336	236.65856
279653	Urban	0.52714	0.51628	0.01086	1267.0	667.88638	654.12676	13.75962
247218	City	0.51938	0.46512	0.05426	1983.0	1029.93054	922.33296	107.59758
...	...	...	...	...	...	...	...	...
11704	City	0.43593	0.37973	0.05620	2496.0	1088.08128	947.80608	140.27520
11705	Village	0.63182	0.55000	0.08182	838.0	529.46516	460.90000	68.56516
11706	City	0.74273	0.60728	0.13545	2739.0	2034.33747	1663.33992	370.99755
11707	City	0.65546	0.57579	0.07967	2596.0	1701.57416	1494.75084	206.82332
11708	Town	0.63866	0.58824	0.05042	1396.0	891.56936	821.18304	70.38632

37940 rows × 8 columns

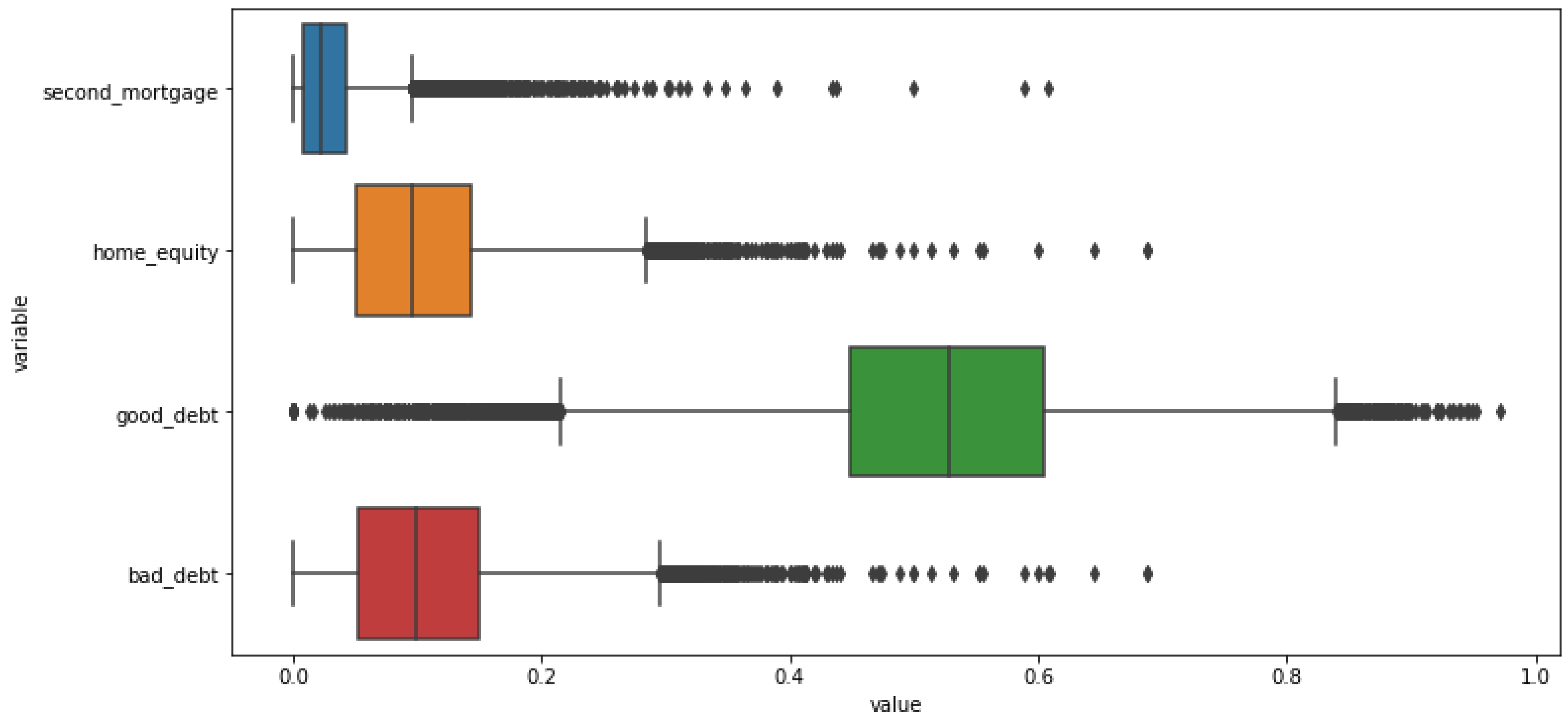
In [32]:
```
# Pie Chart --shows debt by place type
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]],subplot_titles=['Overall Debt', 'Bad Debt'])
fig.add_trace(go.Pie(labels=debt_df.type, values=debt_df.debt_num, name="Overall Debt"),
              1, 1)
fig.add_trace(go.Pie(labels=debt_df.type, values=debt_df.bad_debt_num, name="Bad Debt"),
              1, 2)
fig.show()
# Observations:
# Cities have a higher overall debt followed by towns, CDP, villages, boroughs and urban places
# Cities and villages have 71.6% of bad debt followed by CDP, towns, boroughs and urban places
```



In [33]:
```
#Good Debt vs Bad Debt
fig = go.Figure(data=[go.Pie(labels=['Good Debt','Bad Debt'], values=[sum(debt_df.good_debt_num),sum(debt_df.bad_debt_num)], textinfo='label+percent',
                            title="Good Debt vs Bad Debt")])
fig.show()
# Observations:
# Bad Debt is ~17% across all the states in USA combined
```

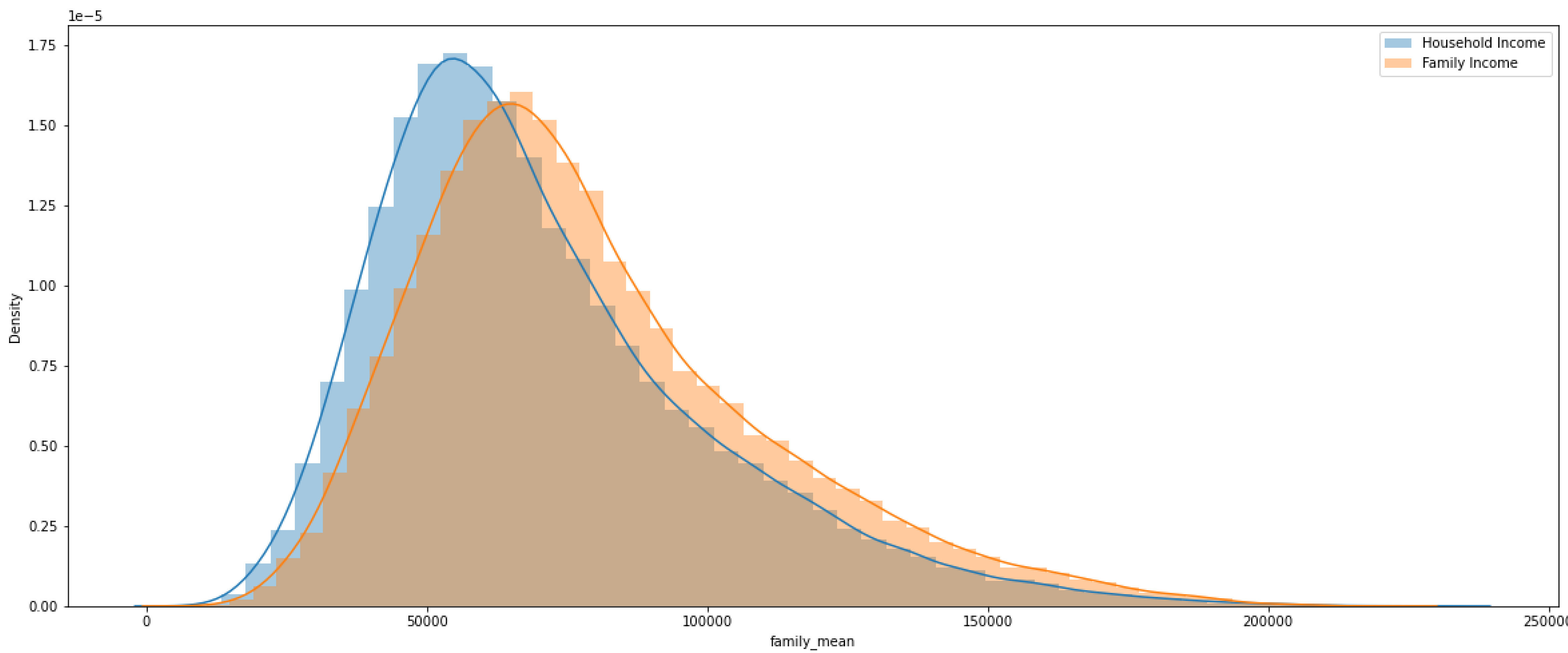


In [34]:
```
# d) Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities
import matplotlib.pyplot as plt
import seaborn as sns

data = real_df[['second_mortgage','home_equity','good_debt','bad_debt']]
plt.figure(figsize=(12,6))
sns.boxplot(y='variable',x='value',data=pd.melt(data))
plt.show()

# Observations:
# Second Mortgage has a smaller range and is silghtly right skewed with outliers towards right
# Home Equity and Bad debt has moderate range and is normally distributed with outliers towards right
# Good Debt has wide range of values and is normally distributed with outliers on both ends
```



In [35]:
```
# Create a collated income distribution chart for family income, house hold income, and remaining income
plt.figure(figsize=(20,8))
sns.distplot(real_df.hi_mean,label='Household Income')
sns.distplot(real_df.family_mean,label='Family Income')
plt.legend()
plt.show()
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).

C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
```



In [36]:

```
'''
Exploratory Data Analysis (EDA):

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

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

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

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

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

Out[36]:

```
'\nExploratory Data Analysis (EDA):\n\n1. 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 meas
urements):\n\na) Use pop and ALand variables to create a new field called population density\n\nb) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called me
dian age c) Visualize the findings using appropriate chart type\n\n2. 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.\n\na) Analyze the married, separated, and divorced population for these population brackets\n\nb) Visualize using appropriate chart type\n\n3. Please detail you
r observations for rent as a percentage of income at an overall level, and for different states.\n\n4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe
your findings.\n'
```

In [37]:
```python
# a) Use pop and ALand variables to create a new field called population density
real_df['pop_den'] = real_df['pop']/real_df['ALand']
real_df
```

Out[37]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	female_age_sample_weight	female_age_samples	pct_own	married	married_snp	separated	divorced	bad_debt	good_d
267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	...	685.33845	2618.0	0.79046	0.57851	0.01882	0.01240	0.08770	0.09408	0.43
246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	...	267.23367	1284.0	0.52483	0.34886	0.01426	0.01426	0.09030	0.04274	0.56
245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	...	707.01963	3238.0	0.85331	0.64745	0.02830	0.01607	0.10657	0.09512	0.63
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	...	362.20193	1559.0	0.65037	0.47257	0.02021	0.02021	0.10106	0.01086	0.51
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	...	1854.48652	3051.0	0.13046	0.12356	0.00000	0.00000	0.03109	0.05426	0.46
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11704	105	12	Florida	FL	Lakeland	Crystal Springs	City	33810	863	28.226068	...	699.33353	2914.0	0.93121	0.65969	0.02135	0.02135	0.08780	0.05620	0.37
11705	31	17	Illinois	IL	Chicago	Chicago City	Village	60609	773	41.804936	...	306.63915	1191.0	0.33122	0.42882	0.07781	0.02829	0.05305	0.08182	0.55
11706	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	1841	978	42.737778	...	900.13903	3723.0	0.84372	0.50269	0.00108	0.00108	0.07294	0.13545	0.60
11707	27	19	Iowa	IA	Carroll	Carroll City	City	51401	712	42.081366	...	693.82905	3213.0	0.83330	0.66699	0.02738	0.00000	0.04694	0.07967	0.57
11708	453	48	Texas	TX	Austin	Sunset Valley City	Town	78745	512	30.219013	...	559.30291	2047.0	0.52587	0.51922	0.08066	0.02520	0.10586	0.05042	0.58

37940 rows × 79 columns

In [38]:

```
# b) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age
real_df['median_age'] = (real_df['male_age_median']*real_df['male_pop'] + real_df['female_age_median']*real_df['female_pop'])/(real_df['male_pop']+real_df['female_pop'])
real_df
```

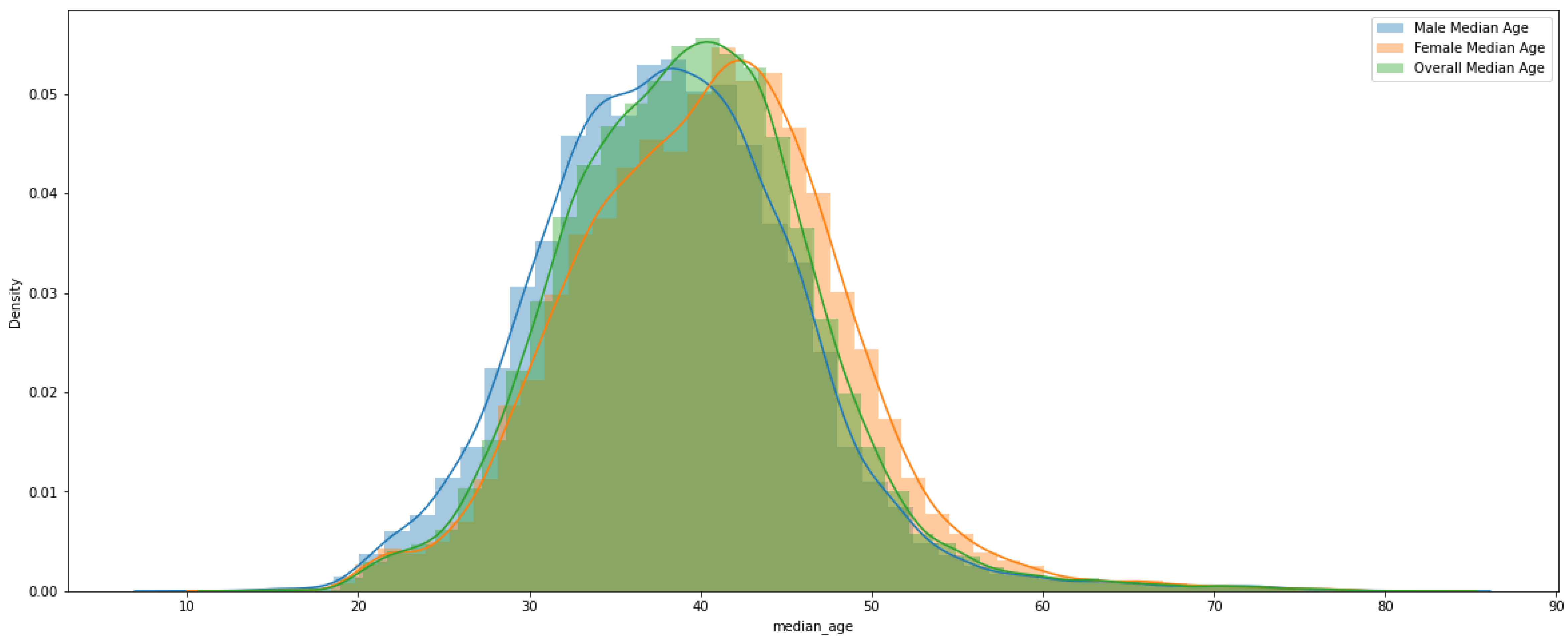
Out[38]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	female_age_samples	pct_own	married	married_snp	separated	divorced	bad_debt	good_debt	pop_den	median_age
267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	...	2618.0	0.79046	0.57851	0.01882	0.01240	0.08770	0.09408	0.43555	0.000026	44.667430
246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	...	1284.0	0.52483	0.34886	0.01426	0.01426	0.09030	0.04274	0.56581	0.001687	34.722748
245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	...	3238.0	0.85331	0.64745	0.02830	0.01607	0.10657	0.09512	0.63972	0.000099	41.774472
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	...	1559.0	0.65037	0.47257	0.02021	0.02021	0.10106	0.01086	0.51628	0.002442	49.879012
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	...	3051.0	0.13046	0.12356	0.00000	0.00000	0.03109	0.05426	0.46512	0.002207	21.965629
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11704	105	12	Florida	FL	Lakeland	Crystal Springs	City	33810	863	28.226068	...	2914.0	0.93121	0.65969	0.02135	0.02135	0.08780	0.05620	0.37973	0.000061	57.620624
11705	31	17	Illinois	IL	Chicago	Chicago City	Village	60609	773	41.804936	...	1191.0	0.33122	0.42882	0.07781	0.02829	0.05305	0.08182	0.55000	0.008241	31.159118
11706	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	1841	978	42.737778	...	3723.0	0.84372	0.50269	0.00108	0.00108	0.07294	0.13545	0.60728	0.001415	39.323630
11707	27	19	Iowa	IA	Carroll	Carroll City	City	51401	712	42.081366	...	3213.0	0.83330	0.66699	0.02738	0.00000	0.04694	0.07967	0.57579	0.000537	44.528597
11708	453	48	Texas	TX	Austin	Sunset Valley City	Town	78745	512	30.219013	...	2047.0	0.52587	0.51922	0.08066	0.02520	0.10586	0.05042	0.58824	0.002069	35.207171

37940 rows × 80 columns

In [39]:
```
# c) Visualize the findings using appropriate chart type
plt.figure(figsize=(20,8))
sns.distplot(real_df.male_age_median,label='Male Median Age')
sns.distplot(real_df.female_age_median,label='Female Median Age')
sns.distplot(real_df.median_age,label='Overall Median Age')
plt.legend()
plt.show()

# Observations:
# Female median age (>40 years) is greater than male median age (<40 years) on an average
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).

C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).

C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
```



In [40]:
```
# 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.
import numpy as np
real_df['pop_bins'] = pd.cut(real_df['pop'],bins=np.linspace(0,54000,6),labels=["0-10800","10800-21600","21600-32400","32400-43200","43200-54000"])
real_df[['pop','pop_bins']]
```

Out[40]:

	pop	pop_bins
267822	5230	0-10800
246444	2633	0-10800
245683	6881	0-10800
279653	2700	0-10800
247218	5637	0-10800
...	...	...
11704	5611	0-10800
11705	2695	0-10800
11706	7392	0-10800
11707	5945	0-10800
11708	4117	0-10800

37940 rows × 2 columns

In [41]:
```
# Checking the count of population for the bins created
real_df['pop_bins'].value_counts()
```

Out[41]:
```
0-10800        37585
10800-21600      339
21600-32400       12
32400-43200        3
43200-54000        1
Name: pop_bins, dtype: int64
```

In [42]:
```
# a) Analyze the married, separated, and divorced population for these population brackets
marital_df = real_df[['pop','married','separated','divorced','pop_bins']]
marital_df['married_num'] = marital_df['pop'] * marital_df['married']
marital_df['separated_num'] = marital_df['pop'] * marital_df['separated']
marital_df['divorced_num'] = marital_df['pop'] * marital_df['divorced']
marital_df
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[42]:

	pop	married	separated	divorced	pop_bins	married_num	separated_num	divorced_num
267822	5230	0.57851	0.01240	0.08770	0-10800	3025.60730	64.85200	458.67100
246444	2633	0.34886	0.01426	0.09030	0-10800	918.54838	37.54658	237.75990
245683	6881	0.64745	0.01607	0.10657	0-10800	4455.10345	110.57767	733.30817
279653	2700	0.47257	0.02021	0.10106	0-10800	1275.93900	54.56700	272.86200
247218	5637	0.12356	0.00000	0.03109	0-10800	696.50772	0.00000	175.25433
...	...	...	...	...	...	...	...	...
11704	5611	0.65969	0.02135	0.08780	0-10800	3701.52059	119.79485	492.64580
11705	2695	0.42882	0.02829	0.05305	0-10800	1155.66990	76.24155	142.96975
11706	7392	0.50269	0.00108	0.07294	0-10800	3715.88448	7.98336	539.17248
11707	5945	0.66699	0.00000	0.04694	0-10800	3965.25555	0.00000	279.05830
11708	4117	0.51922	0.02520	0.10586	0-10800	2137.62874	103.74840	435.82562

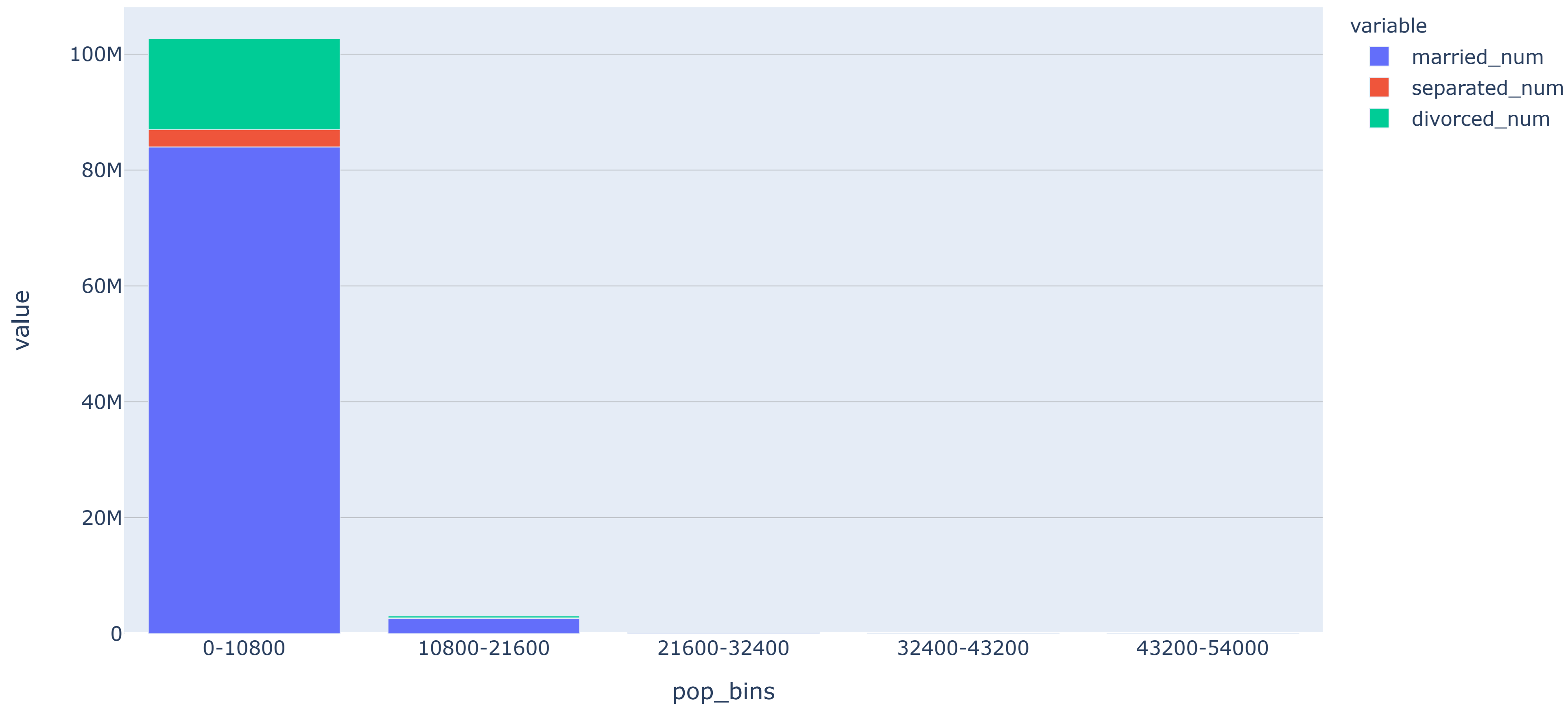
37940 rows × 8 columns

In [43]:

```
group = marital_df.groupby('pop_bins')
group = pd.DataFrame(group['married_num','separated_num','divorced_num'].agg(np.sum))
print(group)

print("\nTotal Population:",np.sum(marital_df['pop']))
print("Total Married:",np.sum(marital_df.married_num))
print("Total Separated:",np.sum(marital_df.separated_num))
print("Total divorced:",np.sum(marital_df.divorced_num))
# Observations:
# Out of ~166Mn population, ~87Mn are married, ~16Mn are divorced and ~3Mn are separated
# Majority of the population is falling in first (0-10800) and second (10800-21600) bins
# ~84Mn are married, ~15.7Mn divorced and ~3Mn separated people fall in locations having <10800 population
```

```
              married_num  separated_num  divorced_num
pop_bins
0-10800      8.397593e+07   2.966350e+06  1.571751e+07
10800-21600  2.675873e+06   6.521666e+04  3.349785e+05
21600-32400  1.771719e+05   3.424042e+03  2.112078e+04
32400-43200  7.829652e+04   1.666850e+03  8.300232e+03
43200-54000  3.953783e+04   2.179386e+02  1.633732e+03

Total Population: 166403989
Total Married: 86946807.97670999
Total Separated: 3036875.1934
Total divorced: 16083544.64636

C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [44]:
```
# b) Visualize using appropriate chart type
plt.figure(figsize=(12,8))
fig = px.bar(group, x=group.index, y=["married_num", "separated_num", "divorced_num"])
fig.show()
```




```
&lt;Figure size 864x576 with 0 Axes&gt;
```

In [45]:
```python
# Calculating average rent as percentage of average household income
real_df['rent_pct'] = real_df['rent_mean']/real_df['hi_mean']
real_df
```

Out[45]:

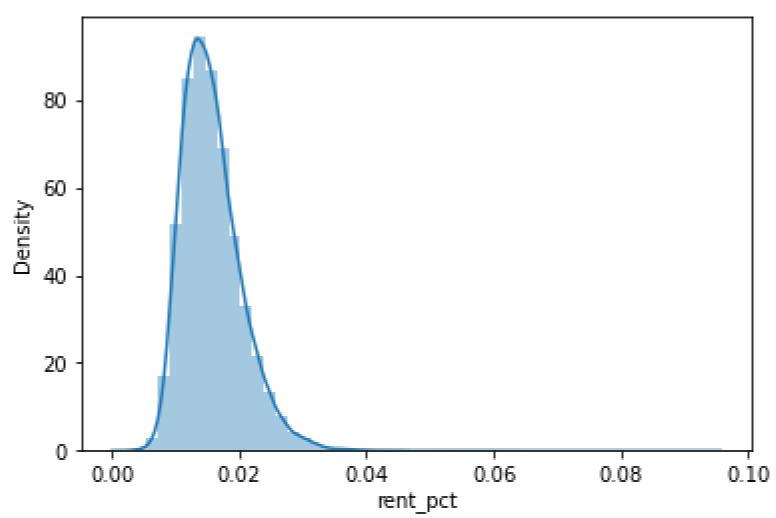
	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	married	married_snp	separated	divorced	bad_debt	good_debt	pop_den	median_age	pop_bins	rent_pct
267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	...	0.57851	0.01882	0.01240	0.08770	0.09408	0.43555	0.000026	44.667430	0-10800	0.012188
246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	...	0.34886	0.01426	0.01426	0.09030	0.04274	0.56581	0.001687	34.722748	0-10800	0.019195
245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202	...	0.64745	0.02830	0.01607	0.10657	0.09512	0.63972	0.000099	41.774472	0-10800	0.008744
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103	...	0.47257	0.02021	0.02021	0.10106	0.01086	0.51628	0.002442	49.879012	0-10800	0.016486
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	66502	785	39.195573	...	0.12356	0.00000	0.00000	0.03109	0.05426	0.46512	0.002207	21.965629	0-10800	0.029483
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11704	105	12	Florida	FL	Lakeland	Crystal Springs	City	33810	863	28.226068	...	0.65969	0.02135	0.02135	0.08780	0.05620	0.37973	0.000061	57.620624	0-10800	0.025273
11705	31	17	Illinois	IL	Chicago	Chicago City	Village	60609	773	41.804936	...	0.42882	0.07781	0.02829	0.05305	0.08182	0.55000	0.008241	31.159118	0-10800	0.019873
11706	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	1841	978	42.737778	...	0.50269	0.00108	0.00108	0.07294	0.13545	0.60728	0.001415	39.323630	0-10800	0.011945
11707	27	19	Iowa	IA	Carroll	Carroll City	City	51401	712	42.081366	...	0.66699	0.02738	0.00000	0.04694	0.07967	0.57579	0.000537	44.528597	0-10800	0.012042
11708	453	48	Texas	TX	Austin	Sunset Valley City	Town	78745	512	30.219013	...	0.51922	0.08066	0.02520	0.10586	0.05042	0.58824	0.002069	35.207171	0-10800	0.016379

37940 rows × 82 columns

In [46]:
```python
# 3. Please detail your observations for rent as a percentage of income at an overall level, and for different states.
sns.distplot(real_df['rent_pct'])
# Observations:
# Overall, Average rent is approximately slightly less than 2% of average household income
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
```

Out[46]: <AxesSubplot:xlabel='rent_pct', ylabel='Density'>



In [47]:
```python
# Grouping rent and household income by states
rent = real_df[['state','state_ab','rent_mean','hi_mean']]
rent = rent.groupby('state_ab')
rent_df = pd.DataFrame(rent['rent_mean','hi_mean'].agg(np.sum))
rent_df['rent_pct'] = rent_df['rent_mean']/rent_df['hi_mean']
print(rent_df)
```

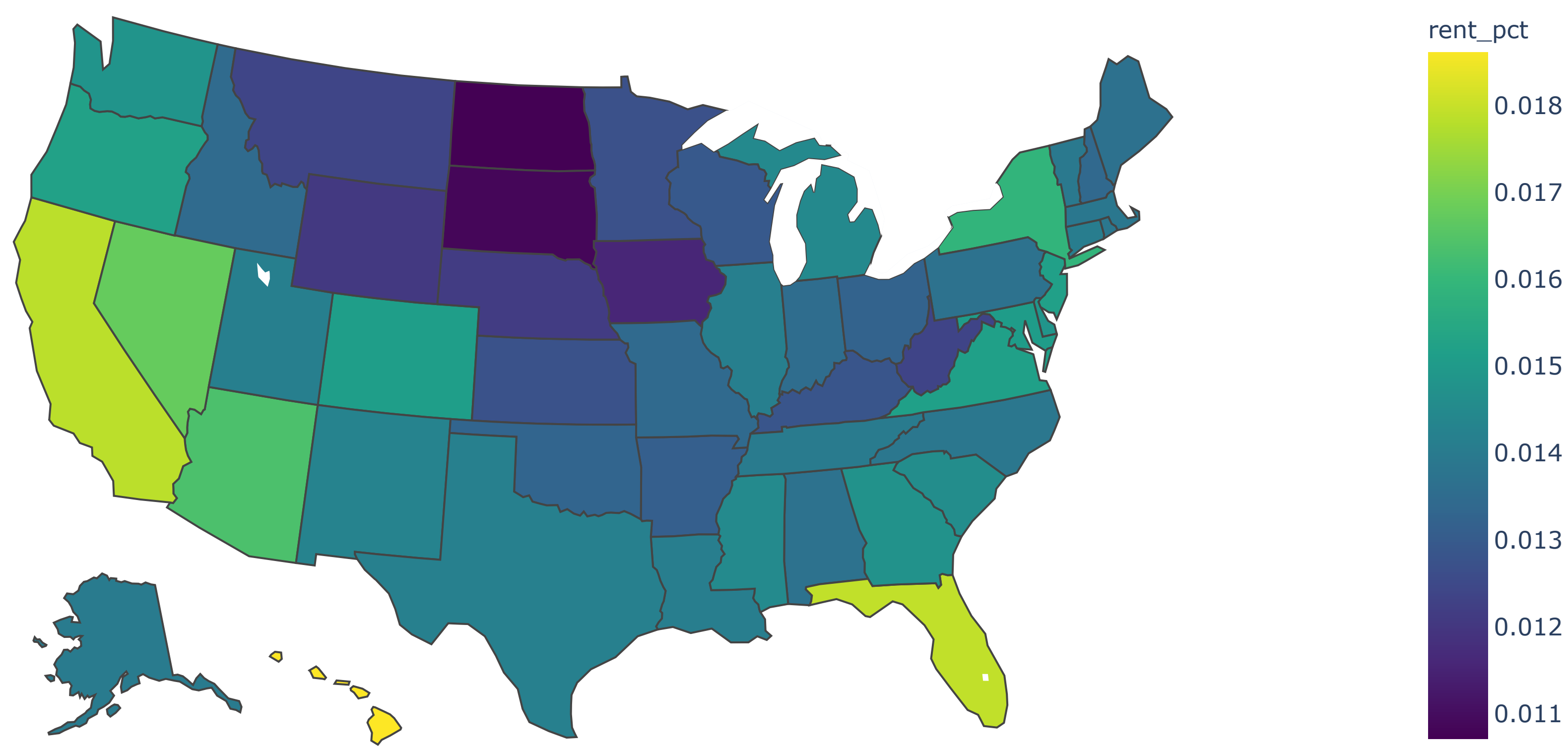
```
             rent_mean       hi_mean  rent_pct
state_ab
AK        1.208197e+05  8.637785e+06  0.013987
AL        4.630363e+05  3.386336e+07  0.013674
AR        2.541412e+05  1.939590e+07  0.013103
AZ        8.402896e+05  5.147532e+07  0.016324
CA        5.962284e+06  3.348984e+08  0.017803
CO        7.692085e+05  5.096446e+07  0.015093
CT        5.738708e+05  4.078962e+07  0.014069
DC        1.350433e+05  8.539456e+06  0.015814
DE        1.191767e+05  8.039070e+06  0.014825
FL        2.562324e+06  1.433095e+08  0.017880
GA        9.957610e+05  6.766986e+07  0.014715
HI        2.493010e+05  1.339747e+07  0.018608
IA        3.049067e+05  2.638637e+07  0.011555
ID        1.194074e+05  8.883005e+06  0.013442
IL        1.614683e+06  1.143652e+08  0.014119
IN        6.405490e+05  4.744462e+07  0.013501
KS        3.523778e+05  2.772092e+07  0.012712
KY        4.069758e+05  3.175204e+07  0.012817
LA        4.960024e+05  3.520436e+07  0.014089
MA        9.210372e+05  6.642643e+07  0.013866
MD        1.034573e+06  6.879901e+07  0.015038
ME        1.607734e+05  1.177583e+07  0.013653
MI        1.314129e+06  9.107831e+07  0.014429
MN        6.655876e+05  5.246227e+07  0.012687
MO        5.989782e+05  4.460168e+07  0.013429
MS        2.562751e+05  1.769839e+07  0.014480
MT        1.177861e+05  9.523003e+06  0.012369
NC        1.005367e+06  7.252183e+07  0.013863
ND        8.098510e+04  7.558809e+06  0.010714
NE        2.237015e+05  1.843371e+07  0.012135
NH        1.722034e+05  1.288359e+07  0.013366
NJ        1.371290e+06  9.043635e+07  0.015163
NM        2.488255e+05  1.744334e+07  0.014265
NV        3.946941e+05  2.360372e+07  0.016722
NY        3.033738e+06  1.911373e+08  0.015872
OH        1.243341e+06  9.422098e+07  0.013196
OK        4.286365e+05  3.230959e+07  0.013267
OR        4.642075e+05  3.052355e+07  0.015208
PA        1.609767e+06  1.176124e+08  0.013687
PR        2.413552e+05  1.336698e+07  0.018056
RI        1.305354e+05  9.423114e+06  0.013853
SC        4.707948e+05  3.229436e+07  0.014578
SD        8.527400e+04  7.874415e+06  0.010829
TN        6.377417e+05  4.558342e+07  0.013991
TX        2.638443e+06  1.862740e+08  0.014164
UT        3.459038e+05  2.450165e+07  0.014118
VA        1.275800e+06  8.418195e+07  0.015155
VT        8.784145e+04  6.308668e+06  0.013924
WA        8.999345e+05  6.083478e+07  0.014793
WI        6.064904e+05  4.690733e+07  0.012930
```

```
WV       1.781120e+05  1.443400e+07  0.012340
WY       6.296093e+04  5.226764e+06  0.012046
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [48]:

```python
#Visualizing rent by states
fig = px.choropleth(rent_df, locations=rent_df.index,
                    locationmode="USA-states",
                    color="rent_pct",
                    scope='usa',
                    color_continuous_scale="Viridis")
fig.show()

# Observations;
# Hawai, California and Florida are the top 3 states with highest rent as a percentage of household income
```



In [49]:
```python
# 4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.
rel_var = real_df[['ALand','AWater','pop','male_pop','female_pop','rent_mean','hi_mean','family_mean','hc_mortgage_mean','hc_mean','home_equity','second_mortgage',
        'home_equity_second_mortgage','debt','bad_debt','hs_degree','hs_degree_male','hs_degree_female','male_age_mean','female_age_mean','pct_own','married',
        'married_snp','separated','divorced']]
rel_var
```

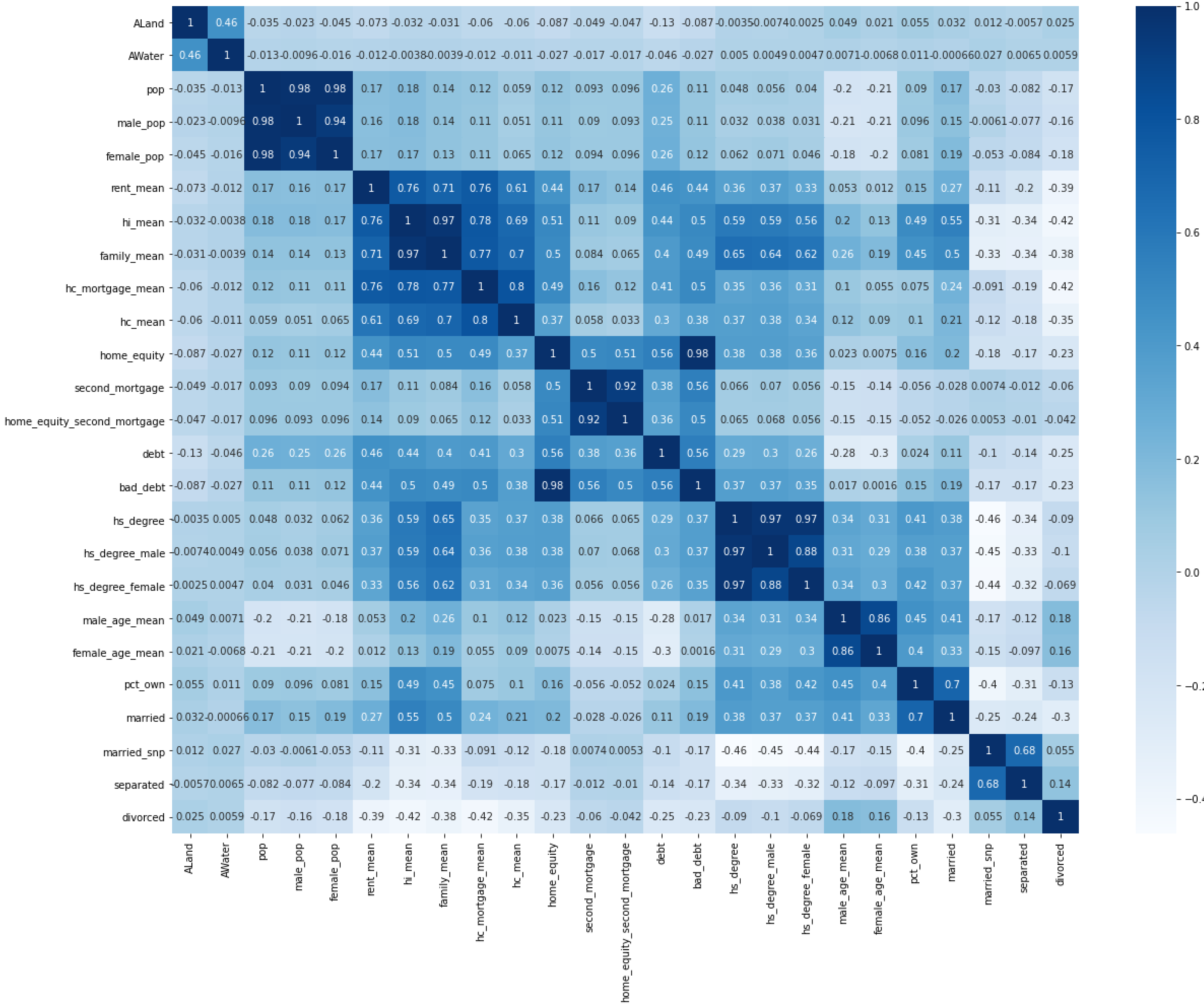
Out[49]:

	ALand	AWater	pop	male_pop	female_pop	rent_mean	hi_mean	family_mean	hc_mortgage_mean	hc_mean	...	hs_degree	hs_degree_male	hs_degree_female	male_age_mean	female_age_mean	pct_own	married	mar
267822	202183361.0	1699120	5230	2612	2618	769.38638	63125.28406	67994.14790	1414.80295	570.01530	...	0.89288	0.85880	0.92434	42.48574	44.48629	0.79046	0.57851	
246444	1560828.0	100363	2633	1349	1284	804.87924	41931.92593	50670.10337	864.41390	351.98293	...	0.90487	0.86947	0.94187	34.84728	36.48391	0.52483	0.34886	
245683	69561595.0	284193	6881	3643	3238	742.77365	84942.68317	95262.51431	1506.06758	556.45986	...	0.94288	0.94616	0.93952	39.38154	42.15810	0.85331	0.64745	
279653	1105793.0	0	2700	1141	1559	803.42018	48733.67116	56401.68133	1175.28642	288.04047	...	0.91500	0.90755	0.92043	48.64749	47.77526	0.65037	0.47257	
247218	2554403.0	0	5637	2586	3051	938.56493	31834.15466	54053.42396	1192.58759	443.68855	...	1.00000	1.00000	1.00000	26.07533	24.17693	0.13046	0.12356	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11704	92582775.0	1166617	5611	2697	2914	1458.82449	57723.48180	70786.81912	1269.83033	536.66053	...	0.92097	0.95007	0.89480	51.03535	53.51255	0.93121	0.65969	
11705	327029.0	0	2695	1504	1191	700.53513	35249.76522	38912.54156	1406.83478	487.66419	...	0.54890	0.49817	0.60965	32.94145	33.14169	0.33122	0.42882	
11706	5225804.0	393810	7392	3669	3723	1069.70567	89549.15374	99484.96572	1791.63902	654.78088	...	0.94057	0.94000	0.94105	35.85743	43.53905	0.84372	0.50269	
11707	11066759.0	0	5945	2732	3213	696.93368	57877.26387	75066.29009	1182.30365	369.29903	...	0.91407	0.92428	0.90634	39.18219	45.63179	0.83330	0.66699	
11708	1990126.0	0	4117	2070	2047	950.09294	58006.33817	54913.24441	1364.17379	550.78197	...	0.78685	0.80615	0.76820	35.56404	35.99955	0.52587	0.51922	

37940 rows × 25 columns

In [50]:

```
#Correlation heat map
plt.figure(figsize=(20,15))
sns.heatmap(rel_var.corr(),cmap='Blues',annot=True)
plt.show()

# Observations:
# Household and family Income is highly correlated with high school degree, percent of owned houses, marriage, average rent, monthly mortgage and owner costs,
# debt and bad debt
# Home equity loan is highly correlated with bad debt
# Home equity and second mortgage is highly correlated with second mortgage
# Bad debt is more likely due to home equity loans and second mortgage
# Age is directly influencing marriage, divorce and percent of home ownership
# Higher houselhold and family income => less divorces and separations
```


ALand
AWater
pop
male_pop
female_pop
rent_mean
hi_mean
family_mean
hc_mortgage_mean
hc_mean
home_equity
second_mortgage
home_equity_second_mortgage
debt
bad_debt
hs_degree
hs_degree_male
hs_degree_female
male_age_mean
female_age_mean
pct_own
married
married_snp
separated
divorced
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [51]:
```
# Dropping unnecessary variables
real_df.drop(['pop_bins'],axis=1,inplace=True)
```

In [52]:
```
# Finding categorical columns
cat_col = real_df.select_dtypes(include="object").columns
cat_col
```

Out[52]:
```
Index(['state', 'state_ab', 'city', 'place', 'type'], dtype='object')
```

In [53]:
```
# Label Encoding
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in cat_col:
  real_df[col] = le.fit_transform(real_df[col].astype(str))

real_df
```

Out[53]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	pct_own	married	married_snp	separated	divorced	bad_debt	good_debt	pop_den	median_age	rent_pct
267822	53	36	32	34	2933	4296	2	13346	315	42.840812	...	0.79046	0.57851	0.01882	0.01240	0.08770	0.09408	0.43555	0.000026	44.667430	0.012188
246444	141	18	14	15	6773	9032	2	46616	574	41.701441	...	0.52483	0.34886	0.01426	0.01426	0.09030	0.04274	0.56581	0.001687	34.722748	0.019195
245683	63	18	14	15	1700	2457	2	46122	317	39.792202	...	0.85331	0.64745	0.02830	0.01607	0.10657	0.09512	0.63972	0.000099	41.774472	0.008744
279653	127	72	39	39	6378	4227	4	927	787	18.396103	...	0.65037	0.47257	0.02021	0.02021	0.10106	0.01086	0.51628	0.002442	49.879012	0.016486
247218	161	20	16	16	4235	6241	2	66502	785	39.195573	...	0.13046	0.12356	0.00000	0.00000	0.03109	0.05426	0.46512	0.002207	21.965629	0.029483
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11704	105	12	9	9	3797	2375	2	33810	863	28.226068	...	0.93121	0.65969	0.02135	0.02135	0.08780	0.05620	0.37973	0.000061	57.620624	0.025273
11705	31	17	13	14	1240	1845	5	60609	773	41.804936	...	0.33122	0.42882	0.07781	0.02829	0.05305	0.08182	0.55000	0.008241	31.159118	0.019873
11706	9	25	21	19	3877	6607	2	1841	978	42.737778	...	0.84372	0.50269	0.00108	0.00108	0.07294	0.13545	0.60728	0.001415	39.323630	0.011945
11707	27	19	15	12	1044	1560	2	51401	712	42.081366	...	0.83330	0.66699	0.02738	0.00000	0.04694	0.07967	0.57579	0.000537	44.528597	0.012042
11708	453	48	44	44	275	10266	3	78745	512	30.219013	...	0.52587	0.51922	0.08066	0.02520	0.10586	0.05042	0.58824	0.002069	35.207171	0.016379

37940 rows × 81 columns

In [54]:
```
# Splitting into train and test sets
real_train_df1 = real_df.iloc[0:26585,:]
real_test_df1 = real_df.iloc[26585:,:]
print(real_train_df1.shape)
print(real_test_df1.shape)
```

```
(26585, 81)
(11355, 81)
```

In [55]:
```
'''
Data Pre-processing:

1. The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a
number of smaller unobserved common factors or latent variables.
2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings.
Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable.
Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships
in the data.

Following are the list of latent variables:

• Highschool graduation rates

• Median population age

• Second mortgage statistics

• Percent own

• Bad debt expense
'''
```

Out[55]:
```
'\nData Pre-processing:\n\n1. The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a \nnumber of smaller unobserved common factors or latent variables. \n2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings. \nEach measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable.\nObtain 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\nin the data. \n\nFollowing are the list of latent variables:\n\n• Highschool graduation rates\n\n• Median population age\n\n• Second mortgage statistics\n\n• Percent own\n\n• Bad debt expense\n'
```

In [56]:
```
!pip install factor_analyzer
```

```
Collecting factor_analyzer
  Downloading factor_analyzer-0.3.2.tar.gz (40 kB)
Requirement already satisfied: pandas in c:\users\bpk20\anaconda3\lib\site-packages (from factor_analyzer) (1.2.0)
Requirement already satisfied: scipy in c:\users\bpk20\anaconda3\lib\site-packages (from factor_analyzer) (1.5.2)
Requirement already satisfied: numpy in c:\users\bpk20\anaconda3\lib\site-packages (from factor_analyzer) (1.19.2)
Requirement already satisfied: scikit-learn in c:\users\bpk20\anaconda3\lib\site-packages (from factor_analyzer) (0.23.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\bpk20\anaconda3\lib\site-packages (from pandas->factor_analyzer) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\bpk20\anaconda3\lib\site-packages (from pandas->factor_analyzer) (2020.5)
Requirement already satisfied: six>=1.5 in c:\users\bpk20\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas->factor_analyzer) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\bpk20\anaconda3\lib\site-packages (from scikit-learn->factor_analyzer) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\bpk20\anaconda3\lib\site-packages (from scikit-learn->factor_analyzer) (1.0.0)
Building wheels for collected packages: factor-analyzer
  Building wheel for factor-analyzer (setup.py): started
  Building wheel for factor-analyzer (setup.py): finished with status 'done'
  Created wheel for factor-analyzer: filename=factor_analyzer-0.3.2-py3-none-any.whl size=40380 sha256=1cfbb77d6535997e0a5a4756f550844d3572cd411b10483b41f3539db749aa84
  Stored in directory: c:\users\bpk20\appdata\local\pip\cache\wheels\8d\9e\4c\fd4cb92cecf157b13702cc0907e5c56ddc48e5388134dc9f1a
Successfully built factor-analyzer
Installing collected packages: factor-analyzer
Successfully installed factor-analyzer-0.3.2
```

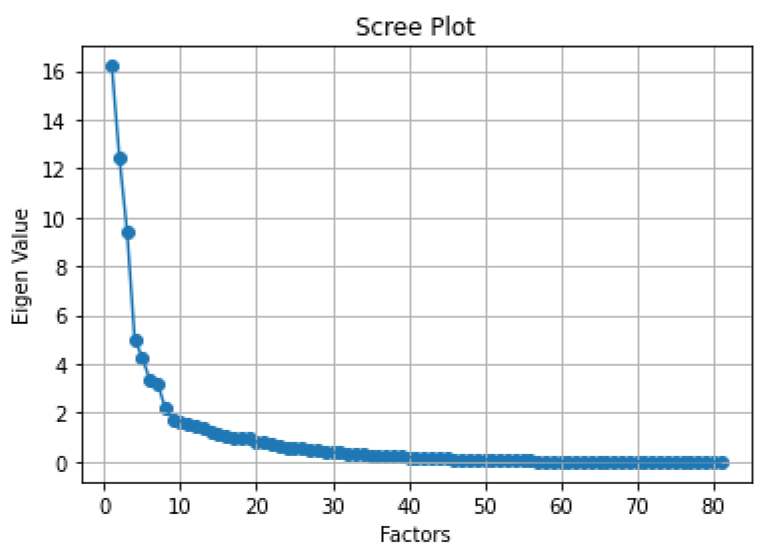
In [57]:
```
# Bartlett’s Test --checks whether the correlation is present in the given data
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity

chi2,p = calculate_bartlett_sphericity(real_df)
print("Chi squared value : ",chi2)
print("p value : ",p)

# Observations:
# Since p-value < 0.05 => correlation is present among the variables in the dataset
```

```
Chi squared value :  13574065.536081182
p value :  0.0
```

In [58]:
```
# KMO test --Measures the proportion of common variance among the variables
from factor_analyzer.factor_analyzer import calculate_kmo

kmo_vars,kmo_model = calculate_kmo(real_df)
print(kmo_model)
```

```
C:\Users\bpk20\anaconda3\lib\site-packages\factor_analyzer\utils.py:248: UserWarning:

The inverse of the variance-covariance matrix was calculated using the Moore-Penrose generalized matrix inversion, due to its determinant being at or very close to zero.


0.41338612846626166
```

In [59]:
```
# Factor Analysis
from factor_analyzer import FactorAnalyzer

fa = FactorAnalyzer()
fa.fit(real_df,10)

#Get Eigen values and plot them
ev, v = fa.get_eigenvalues()
ev
plt.scatter(range(1,real_df.shape[1]+1),ev)
plt.plot(range(1,real_df.shape[1]+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigen Value')
plt.grid()

# Observations:
# Eigen Values are dropping below 3 after 7th factore an ddropping below 2 after 8th factor
# Optimal number of factors = 7 or 8
```



In [60]:
```python
# Interpreting the Factors by plotting loadings
fa = FactorAnalyzer(n_factors=7,rotation='varimax')
fa.fit(real_df)

# Settings to see all rows and columns
pd.set_option('display.max_columns', None,'display.max_rows',None)
print(pd.DataFrame(fa.loadings_,index=real_df.columns))

# Observations:
# Factor1 - Higher loadings on income, degree, rent, mortgage costs and debt
# Factor2 - Higher loadings on population and number of samples used
# Factor3 - Higher loadings on age related variables
# Factor4 - Higher loadings on rent as percent of household income
# Factor5 - Higher loadings on mortgage, home equity loan and bad debt
# Factor6 - Higher loadings on number of smples used for rent, universe samples, pct_own and married
# Factor7 - Higher loadings on state and city (location)
```

```
                               0         1         2         3         4  \
COUNTYID               -0.097256  0.025290 -0.062471 -0.029951 -0.095203
STATEID                -0.091589  0.012910 -0.020031 -0.074374 -0.087602
state                  -0.065399  0.013697 -0.032208 -0.092287 -0.065336
state_ab               -0.054328  0.009350 -0.034863 -0.086600 -0.074816
city                    0.000412  0.015274  0.029055  0.005352  0.028644
place                   0.025166  0.004227  0.001258  0.008086  0.008176
type                   -0.087404  0.034828 -0.052067 -0.041208 -0.010199
zip_code               -0.064484  0.061158 -0.173057 -0.084296 -0.019140
area_code               0.034358  0.028656 -0.036099  0.036619 -0.033603
lat                     0.116377 -0.104782 -0.002249 -0.138891  0.256577
lng                    -0.018869 -0.049593  0.165753  0.048882 -0.014724
ALand                  -0.050740 -0.023625  0.040989 -0.107808 -0.096798
AWater                 -0.007421 -0.014175  0.000352 -0.048487 -0.044878
pop                     0.114414  0.972192 -0.125582  0.014183  0.046952
male_pop                0.109631  0.945854 -0.143080 -0.003806  0.036450
female_pop              0.114915  0.962500 -0.103314  0.031001  0.057126
rent_mean               0.813326  0.056810 -0.120581  0.106553  0.147375
rent_median             0.767098  0.054334 -0.133793  0.104835  0.140994
```

In [61]:
```python
# Variance captured by each factor
print(pd.DataFrame(fa.get_factor_variance(),index=['Variance','Proportional Var','Cumulative Var']))
```

```
                          0          1         2         3         4  \
Variance          12.949905  11.918745  6.889901  6.266754  5.376851
Proportional Var   0.159875   0.147145  0.085061  0.077367  0.066381
Cumulative Var     0.159875   0.307020  0.392081  0.469448  0.535829

                         5         6
Variance          5.333653  3.409919
Proportional Var  0.065848  0.042098
Cumulative Var    0.601677  0.643774
```

In [62]:
```python
# Communality - Proportion of each variable's variance explained by each factor
print(pd.DataFrame(fa.get_communalities(),index=real_df.columns,columns=['Communalities']))
```

```
                             Communalities
COUNTYID                          0.062154
STATEID                           0.872988
state                             0.879970
state_ab                          0.869175
city                              0.004555
place                             0.001624
type                              0.026250
zip_code                          0.214368
area_code                         0.008547
lat                               0.148296
lng                               0.248962
ALand                             0.030753
AWater                            0.006964
pop                               0.985176
male_pop                          0.935528
female_pop                        0.965196
rent_mean                         0.759268
rent_median                       0.686887
rent_stdev                        0.477813
rent_sample_weight                0.759403
rent_samples                      0.963845
rent_gt_10                        0.196571
rent_gt_15                        0.423013
rent_gt_20                        0.629172
rent_gt_25                        0.767717
rent_gt_30                        0.819245
rent_gt_35                        0.798159
rent_gt_40                        0.733241
rent_gt_50                        0.586214
universe_samples                  0.964491
used_samples                      0.958398
hi_mean                           0.956558
hi_median                         0.904637
hi_stdev                          0.850432
hi_sample_weight                  0.943161
hi_samples                        0.965814
family_mean                       0.934510
family_median                     0.884887
family_stdev                      0.721977
family_sample_weight              0.856723
family_samples                    0.949141
hc_mortgage_mean                  0.891354
hc_mortgage_median                0.864005
hc_mortgage_stdev                 0.618264
hc_mortgage_sample_weight         0.804096
hc_mortgage_samples               0.902672
hc_mean                           0.754688
hc_median                         0.700030
hc_stdev                          0.479048
hc_samples                        0.820641
hc_sample_weight                  0.800902
home_equity_second_mortgage       0.513724
second_mortgage                   0.525632
home_equity                       0.734844
debt                              0.729572
second_mortgage_cdf               0.522839
home_equity_cdf                   0.778317
```

```
debt_cdf                      0.734485
hs_degree                     0.591043
hs_degree_male                0.555473
hs_degree_female              0.549235
male_age_mean                 0.825634
male_age_median               0.788499
male_age_stdev                0.341424
male_age_sample_weight        0.853178
male_age_samples              0.935528
female_age_mean               0.822017
female_age_median             0.818427
female_age_stdev              0.242223
female_age_sample_weight      0.868804
female_age_samples            0.965196
pct_own                       0.884761
married                       0.583822
married_snp                   0.220001
separated                     0.162206
divorced                      0.284110
bad_debt                      0.753889
good_debt                     0.418790
pop_den                       0.299481
median_age                    0.898180
rent_pct                      0.556919
```

In [63]:

```
# Resetting display options
pd.reset_option('display.max_columns')
pd.reset_option('display.max_rows')
real_df
```

Out[63]:

	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	...	pct_own	married	married_snp	separated	divorced	bad_debt	good_debt	pop_den	median_age	rent_pct
267822	53	36	32	34	2933	4296	2	13346	315	42.840812	...	0.79046	0.57851	0.01882	0.01240	0.08770	0.09408	0.43555	0.000026	44.667430	0.012188
246444	141	18	14	15	6773	9032	2	46616	574	41.701441	...	0.52483	0.34886	0.01426	0.01426	0.09030	0.04274	0.56581	0.001687	34.722748	0.019195
245683	63	18	14	15	1700	2457	2	46122	317	39.792202	...	0.85331	0.64745	0.02830	0.01607	0.10657	0.09512	0.63972	0.000099	41.774472	0.008744
279653	127	72	39	39	6378	4227	4	927	787	18.396103	...	0.65037	0.47257	0.02021	0.02021	0.10106	0.01086	0.51628	0.002442	49.879012	0.016486
247218	161	20	16	16	4235	6241	2	66502	785	39.195573	...	0.13046	0.12356	0.00000	0.00000	0.03109	0.05426	0.46512	0.002207	21.965629	0.029483
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11704	105	12	9	9	3797	2375	2	33810	863	28.226068	...	0.93121	0.65969	0.02135	0.02135	0.08780	0.05620	0.37973	0.000061	57.620624	0.025273
11705	31	17	13	14	1240	1845	5	60609	773	41.804936	...	0.33122	0.42882	0.07781	0.02829	0.05305	0.08182	0.55000	0.008241	31.159118	0.019873
11706	9	25	21	19	3877	6607	2	1841	978	42.737778	...	0.84372	0.50269	0.00108	0.00108	0.07294	0.13545	0.60728	0.001415	39.323630	0.011945
11707	27	19	15	12	1044	1560	2	51401	712	42.081366	...	0.83330	0.66699	0.02738	0.00000	0.04694	0.07967	0.57579	0.000537	44.528597	0.012042
11708	453	48	44	44	275	10266	3	78745	512	30.219013	...	0.52587	0.51922	0.08066	0.02520	0.10586	0.05042	0.58824	0.002069	35.207171	0.016379

37940 rows × 81 columns

In [64]:
```
'''
Data Modeling :

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
'''
```

Out[64]:
```
'\nData Modeling :\n\n1. Build a linear Regression model to predict the total monthly expenditure for home mortgages loan. Please refer ‘deplotment_RE.xlsx’. \nColumn hc_mortgage_mean is predicte
d variable. This is the mean monthly mortgage and owner costs of specified geographical location. \n\nNote: Exclude loans from prediction model which have NaN (Not a Number) values for hc_mortgag
e_mean.\n\na) Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.\n\nb) Run another model at State level. There are 52 states in USA.\n
\nc) Keep below considerations while building a linear regression model. Data Modeling :\n\n• Variables should have significant impact on predicting Monthly mortgage and owner costs\n\n• Utilize
all predictor variable to start with initial hypothesis\n\n• R square of 60 percent and above should be achieved\n\n• Ensure Multi-collinearity does not exist in dependent variables\n\n• Test if
predicted variable is normally distributed\n'
```

In [65]:
```
# Splitting the dataset into x and y variables
x_train = real_train_df1.drop(['state','state_ab','hc_mortgage_median','hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples'],axis=1)
y_train = real_train_df1['hc_mortgage_mean']

print(x_train.shape)
print(y_train.shape)
```

```
(26585, 75)
(26585,)
```

In [66]:
```
# Splitting the dataset into x and y variables
x_test = real_test_df1.drop(['state','state_ab','hc_mortgage_median','hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples'],axis=1)
y_test = real_test_df1['hc_mortgage_mean']

print(x_test.shape)
print(y_test.shape)
```

```
(11355, 75)
(11355,)
```

In [67]:
```
# Feature Scaling --Standardizing our dataset
from sklearn.preprocessing import StandardScaler

# Initialization
sc = StandardScaler()

# Fitting and transforming on train data
x_train_std = sc.fit_transform(x_train)

# Transforming test data
x_test_std = sc.transform(x_test)
```

In [68]:
```
# Model Building
# Regression Model 1 ---Multiple Linear Regression
from sklearn.linear_model import LinearRegression

# Initialization
lr = LinearRegression()

# Fitting the model on train set
real_lin_reg = lr.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_lin_reg.predict(x_test_std)
```

In [69]:
```
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  1.0
MSE:  3.0404658380029244e-24
RMSE:  1.7436931605081568e-12
```

In [70]:
```
# Creating variables to capture evaluation metrics
r2s = []
mse = []
rmse = []
```

In [71]:
```
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [72]:
```python
# Model Building
# Regression Model 2 ---Ridge Regression
from sklearn.linear_model import Ridge

# Initialization
rdg = Ridge()

# Fitting the model on train set
real_rdg_reg = rdg.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_rdg_reg.predict(x_test_std)
```

In [73]:
```python
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.9999999888232182
MSE:  0.004453686657809074
RMSE:  0.06673594726838808
```

In [74]:
```python
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [75]:
```python
# Model Building
# Regression Model 3 ---Lasso Regression
from sklearn.linear_model import Lasso

# Initialization
lso = Lasso()

# Fitting the model on train set
real_lso_reg = lso.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_lso_reg.predict(x_test_std)
```

In [76]:
```python
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.9999974029216658
MSE:  1.0348750933186275
RMSE:  1.0172881073317566
```

In [77]:
```python
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [78]:
```
# Model Building
# Regression Model 4 ---Elastic Net Regression
from sklearn.linear_model import ElasticNet

# Initialization
en = ElasticNet()

# Fitting the model on train set
real_en_reg = en.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_en_reg.predict(x_test_std)
```

In [79]:
```
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.9243020218845118
MSE:  30163.877282578163
RMSE:  173.677509432218
```

In [80]:
```
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [81]:
```
# Model Building
# Regression Model 5 ---Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor

# Initialization
dtr = DecisionTreeRegressor()

# Fitting the model on train set
real_dtr_reg = dtr.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_dtr_reg.predict(x_test_std)
```

In [82]:
```
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.9999955608177532
MSE:  1.7689105027810774
RMSE:  1.330003948408078
```

In [83]:
```
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [84]:
```python
# Model Building
# Regression Model 6 ---Random Forest Regression
from sklearn.ensemble import RandomForestRegressor

# Initialization
rfr = RandomForestRegressor()

# Fitting the model on train set
real_rfr_reg = rfr.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_rfr_reg.predict(x_test_std)
```

In [85]:
```python
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.9999911712014465
MSE:  3.5180701354103006
RMSE:  1.8756519227751989
```

In [86]:
```python
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [87]:
```python
# Model Building
# Regression Model 7 ---Support Vector Regression
from sklearn.svm import SVR

# Initialization
svr = SVR()

# Fitting the model on train set
real_svr_reg = svr.fit(x_train_std,y_train)

# Predictions on test set
y_pred = real_svr_reg.predict(x_test_std)
```

In [88]:
```python
# Model Evaluation
from sklearn import metrics

print("Accuracy: ",metrics.r2_score(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Accuracy:  0.6889994642255626
MSE:  123926.45390877084
RMSE:  352.0318933119141
```

In [89]:
```python
# Appending results into the list
r2s.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

In [90]:

```python
# Summary Table
reg_models = ['Linear Regression','Ridge regression','Lasso Regression','Elastic Net Regression','Decision Tree Regression','Random Forest Regression',
              'Support Vector Regression']
results_df = pd.DataFrame({'Regression Model':reg_models,'R2_Score':r2s,'Mean Squared Error':mse,'Root Mean Squared Error':rmse})
results_df

# Observations:
# All the models have an r2_score > 60%
# Linear Regression model gives the best possible accuracy with low error values => Best Model
```

Out[90]:

	Regression Model	R2_Score	Mean Squared Error	Root Mean Squared Error
0	Linear Regression	1.000000	3.040466e-24	1.743693e-12
1	Ridge regression	1.000000	4.453687e-03	6.673595e-02
2	Lasso Regression	0.999997	1.034875e+00	1.017288e+00
3	Elastic Net Regression	0.924302	3.016388e+04	1.736775e+02
4	Decision Tree Regression	0.999996	1.768911e+00	1.330004e+00
5	Random Forest Regression	0.999991	3.518070e+00	1.875652e+00
6	Support Vector Regression	0.688999	1.239265e+05	3.520319e+02