Project Task: Week 1

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.
```

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

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Import data.

In [2]:
```
df_train = pd.read_csv('train.csv')
df_test = pd. read_csv('test.csv')
```

In [3]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(27321, 80)
(11709, 80)
```

In [4]:
```
27321 + 11709
```

Out[4]:
```
39030
```

In [5]:
```
df = pd.concat([df_train,df_test])
df.shape
```

Out[5]:
```
(39030, 80)
```

In [6]:
```
# Checking and removing duplicates if any
if df.duplicated().sum().any()==True:
    print('There are duplicate values in the dataset')
else:
    print('There are no duplicate values in the dataset')
```

```
There are duplicate values in the dataset
```

In [7]:
```
df = df.drop_duplicates(keep=False)
```

In [8]:
```
df.shape
```

Out[8]:
```
(38466, 80)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38466 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   UID                   38466 non-null  int64
 1   BLOCKID               0 non-null      float64
 2   SUMLEVEL              38466 non-null  int64
 3   COUNTYID              38466 non-null  int64
 4   STATEID               38466 non-null  int64
 5   state                 38466 non-null  object
 6   state_ab              38466 non-null  object
 7   city                  38466 non-null  object
 8   place                 38466 non-null  object
 9   type                  38466 non-null  object
 10  primary               38466 non-null  object
 11  zip_code              38466 non-null  int64
 12  area_code             38466 non-null  int64
 13  lat                   38466 non-null  float64
 14  lng                   38466 non-null  float64
 15  ALand                 38466 non-null  float64
 16  AWater                38466 non-null  int64
 17  pop                   38466 non-null  int64
 18  male_pop              38466 non-null  int64
 19  female_pop            38466 non-null  int64
 20  rent_mean             38254 non-null  float64
 21  rent_median           38254 non-null  float64
 22  rent_stdev            38254 non-null  float64
 23  rent_sample_weight    38254 non-null  float64
 24  rent_samples          38254 non-null  float64
 25  rent_gt_10            38253 non-null  float64
 26  rent_gt_15            38253 non-null  float64
 27  rent_gt_20            38253 non-null  float64
 28  rent_gt_25            38253 non-null  float64
 29  rent_gt_30            38253 non-null  float64
 30  rent_gt_35            38253 non-null  float64
 31  rent_gt_40            38253 non-null  float64
 32  rent_gt_50            38253 non-null  float64
```

```
 33  universe_samples             38466 non-null  int64
 34  used_samples                 38466 non-null  int64
 35  hi_mean                      38287 non-null  float64
 36  hi_median                    38287 non-null  float64
 37  hi_stdev                     38287 non-null  float64
 38  hi_sample_weight             38287 non-null  float64
 39  hi_samples                   38287 non-null  float64
 40  family_mean                  38265 non-null  float64
 41  family_median                38265 non-null  float64
 42  family_stdev                 38265 non-null  float64
 43  family_sample_weight         38265 non-null  float64
 44  family_samples               38265 non-null  float64
 45  hc_mortgage_mean             38062 non-null  float64
 46  hc_mortgage_median           38062 non-null  float64
 47  hc_mortgage_stdev            38062 non-null  float64
 48  hc_mortgage_sample_weight    38062 non-null  float64
 49  hc_mortgage_samples          38062 non-null  float64
 50  hc_mean                      38030 non-null  float64
 51  hc_median                    38030 non-null  float64
 52  hc_stdev                     38030 non-null  float64
 53  hc_samples                   38030 non-null  float64
 54  hc_sample_weight             38030 non-null  float64
 55  home_equity_second_mortgage  38131 non-null  float64
 56  second_mortgage              38131 non-null  float64
 57  home_equity                  38131 non-null  float64
 58  debt                         38131 non-null  float64
 59  second_mortgage_cdf          38131 non-null  float64
 60  home_equity_cdf              38131 non-null  float64
 61  debt_cdf                     38131 non-null  float64
 62  hs_degree                    38337 non-null  float64
 63  hs_degree_male               38325 non-null  float64
 64  hs_degree_female             38313 non-null  float64
 65  male_age_mean                38334 non-null  float64
 66  male_age_median              38334 non-null  float64
 67  male_age_stdev               38334 non-null  float64
 68  male_age_sample_weight       38334 non-null  float64
 69  male_age_samples             38334 non-null  float64
 70  female_age_mean              38327 non-null  float64
 71  female_age_median            38327 non-null  float64
 72  female_age_stdev             38327 non-null  float64
 73  female_age_sample_weight     38327 non-null  float64
 74  female_age_samples           38327 non-null  float64
 75  pct_own                      38287 non-null  float64
 76  married                      38332 non-null  float64
 77  married_snp                  38332 non-null  float64
 78  separated                    38332 non-null  float64
 79  divorced                     38332 non-null  float64
dtypes: float64(62), int64(12), object(6)
```

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

In [10]:
```
# Checking and removing duplicates if any
if df['UID'].duplicated().sum().any()==True:
    print('There are duplicate values in the dataset')
else:
    print('There are no duplicate values in the dataset')
```

```
There are no duplicate values in the dataset
```

In [11]:
```
# UID here appears to be the primaray key but it is as int type, we will convert the dat

df['UID'] = df['UID'].apply(str)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38466 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   UID                            38466 non-null  object
 1   BLOCKID                        0 non-null      float64
 2   SUMLEVEL                       38466 non-null  int64
 3   COUNTYID                       38466 non-null  int64
 4   STATEID                        38466 non-null  int64
 5   state                          38466 non-null  object
 6   state_ab                       38466 non-null  object
 7   city                           38466 non-null  object
 8   place                          38466 non-null  object
 9   type                           38466 non-null  object
 10  primary                        38466 non-null  object
 11  zip_code                       38466 non-null  int64
 12  area_code                      38466 non-null  int64
 13  lat                            38466 non-null  float64
 14  lng                            38466 non-null  float64
 15  ALand                          38466 non-null  float64
 16  AWater                         38466 non-null  int64
 17  pop                            38466 non-null  int64
 18  male_pop                       38466 non-null  int64
 19  female_pop                     38466 non-null  int64
 20  rent_mean                      38254 non-null  float64
 21  rent_median                    38254 non-null  float64
 22  rent_stdev                     38254 non-null  float64
 23  rent_sample_weight             38254 non-null  float64
 24  rent_samples                   38254 non-null  float64
 25  rent_gt_10                     38253 non-null  float64
 26  rent_gt_15                     38253 non-null  float64
 27  rent_gt_20                     38253 non-null  float64
 28  rent_gt_25                     38253 non-null  float64
 29  rent_gt_30                     38253 non-null  float64
 30  rent_gt_35                     38253 non-null  float64
 31  rent_gt_40                     38253 non-null  float64
 32  rent_gt_50                     38253 non-null  float64
 33  universe_samples               38466 non-null  int64
 34  used_samples                   38466 non-null  int64
 35  hi_mean                        38287 non-null  float64
 36  hi_median                      38287 non-null  float64
 37  hi_stdev                       38287 non-null  float64
 38  hi_sample_weight               38287 non-null  float64
 39  hi_samples                     38287 non-null  float64
 40  family_mean                    38265 non-null  float64
 41  family_median                  38265 non-null  float64
 42  family_stdev                   38265 non-null  float64
 43  family_sample_weight           38265 non-null  float64
 44  family_samples                 38265 non-null  float64
 45  hc_mortgage_mean               38062 non-null  float64
 46  hc_mortgage_median             38062 non-null  float64
 47  hc_mortgage_stdev              38062 non-null  float64
 48  hc_mortgage_sample_weight      38062 non-null  float64
 49  hc_mortgage_samples            38062 non-null  float64
 50  hc_mean                        38030 non-null  float64
 51  hc_median                      38030 non-null  float64
 52  hc_stdev                       38030 non-null  float64
 53  hc_samples                     38030 non-null  float64
 54  hc_sample_weight               38030 non-null  float64
 55  home_equity_second_mortgage    38131 non-null  float64
 56  second_mortgage                38131 non-null  float64
 57  home_equity                    38131 non-null  float64
 58  debt                           38131 non-null  float64
 59  second_mortgage_cdf            38131 non-null  float64
```

```
 60  home_equity_cdf              38131 non-null  float64
 61  debt_cdf                     38131 non-null  float64
 62  hs_degree                    38337 non-null  float64
 63  hs_degree_male               38325 non-null  float64
 64  hs_degree_female             38313 non-null  float64
 65  male_age_mean                38334 non-null  float64
 66  male_age_median              38334 non-null  float64
 67  male_age_stdev               38334 non-null  float64
 68  male_age_sample_weight       38334 non-null  float64
 69  male_age_samples             38334 non-null  float64
 70  female_age_mean              38327 non-null  float64
 71  female_age_median            38327 non-null  float64
 72  female_age_stdev             38327 non-null  float64
 73  female_age_sample_weight     38327 non-null  float64
 74  female_age_samples           38327 non-null  float64
 75  pct_own                      38287 non-null  float64
 76  married                      38332 non-null  float64
 77  married_snp                  38332 non-null  float64
 78  separated                    38332 non-null  float64
 79  divorced                     38332 non-null  float64
dtypes: float64(62), int64(11), object(7)
```

In [12]:
```
# checking columns with zero variance
df_var=df.var().reset_index()
df_var[df_var[0]<=0]
```

Out[12]:

	index	0
1	SUMLEVEL	0.0

In [13]:
```
df.describe()
```

Out[13]:

	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng
count	0.0	38466.0	38466.000000	38466.000000	38466.000000	38466.000000	38466.000000	38466.000000
mean	NaN	140.0	85.772916	28.318073	50140.179717	595.617402	37.497225	-91.313079
std	NaN	0.0	98.718178	16.446219	29584.902391	232.291393	5.595144	16.335090
min	NaN	140.0	1.000000	1.000000	601.000000	201.000000	17.929085	-166.770979
25%	NaN	140.0	29.000000	13.000000	26612.750000	405.000000	33.913083	-97.814294
50%	NaN	140.0	63.000000	28.000000	47725.000000	614.000000	38.725800	-86.616366
75%	NaN	140.0	109.000000	42.000000	77304.000000	801.000000	41.356422	-79.819905
max	NaN	140.0	840.000000	72.000000	99929.000000	989.000000	67.074017	-65.379332

8 rows × 73 columns

In [14]:
```
df.describe(include='object')
```

Out[14]:

	UID	state	state_ab	city	place	type	primary
count	38466	38466	38466	38466	38466	38466	38466
unique	38466	52	52	8163	11842	6	1
top	267822	California	CA	Chicago	New York City	City	tract
freq	1	4119	4119	403	679	21378	38466

Observation 1:

- col BLOCKID is all blank,s which we can remove
- col SUMLEVEL has Zero variance, which can also be dropped
- col primary has only one value = 'tract', so basically a zero variance, which can be dropped

In [15]:
```
df.drop(['BLOCKID','SUMLEVEL','primary'], axis=1, inplace=True)
```

In [16]:
```
df.shape
```

Out[16]:
```
(38466, 77)
```

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 [17]:
```
#rows with missing values
print('rows with missing values: ',df.shape[0] - df.dropna().shape[0])
print('% of missing data:',(df.shape[0] - df.dropna().shape[0])/df.shape[0] )
```

```
rows with missing values:  526
% of missing data: 0.013674413768002911
```

In [18]:
```
# checking missing data
missing_data=pd.DataFrame(df.isnull().sum())
missing_data['missing %'] = missing_data[0]/df.shape[0]
print(missing_data[missing_data[0]!=0].shape)
missing_data[missing_data[0]!=0]
```

```
(58, 2)
```

Out[18]:

	0	missing %
rent_mean	212	0.005511
rent_median	212	0.005511
rent_stdev	212	0.005511
rent_sample_weight	212	0.005511
rent_samples	212	0.005511
rent_gt_10	213	0.005537
rent_gt_15	213	0.005537
rent_gt_20	213	0.005537
rent_gt_25	213	0.005537
rent_gt_30	213	0.005537
rent_gt_35	213	0.005537
rent_gt_40	213	0.005537
rent_gt_50	213	0.005537
hi_mean	179	0.004653

	0	missing %
hi_median	179	0.004653
hi_stdev	179	0.004653
hi_sample_weight	179	0.004653
hi_samples	179	0.004653
family_mean	201	0.005225
family_median	201	0.005225
family_stdev	201	0.005225
family_sample_weight	201	0.005225
family_samples	201	0.005225
hc_mortgage_mean	404	0.010503
hc_mortgage_median	404	0.010503
hc_mortgage_stdev	404	0.010503
hc_mortgage_sample_weight	404	0.010503
hc_mortgage_samples	404	0.010503
hc_mean	436	0.011335
hc_median	436	0.011335
hc_stdev	436	0.011335
hc_samples	436	0.011335
hc_sample_weight	436	0.011335
home_equity_second_mortgage	335	0.008709
second_mortgage	335	0.008709
home_equity	335	0.008709
debt	335	0.008709
second_mortgage_cdf	335	0.008709
home_equity_cdf	335	0.008709
debt_cdf	335	0.008709
hs_degree	129	0.003354
hs_degree_male	141	0.003666
hs_degree_female	153	0.003978
male_age_mean	132	0.003432
male_age_median	132	0.003432
male_age_stdev	132	0.003432
male_age_sample_weight	132	0.003432
male_age_samples	132	0.003432
female_age_mean	139	0.003614
female_age_median	139	0.003614
female_age_stdev	139	0.003614

	0	missing %
female_age_sample_weight	139	0.003614
female_age_samples	139	0.003614
pct_own	179	0.004653
married	134	0.003484
married_snp	134	0.003484

Observation 2:

- we see 59 columns has missing data
- there are only around 1% of the data that are missing from the given dataset, which is compartively very less to consider given the size of the dataset
- for simplicity, we can drop all these rows with missing data

In [19]:
```
#dropping rows with missing data
df_new = df.dropna()
df_new.shape
```

Out[19]:
```
(37940, 77)
```

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 [20]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [21]:
```
df_new['pct_own'].describe()
```

Out[21]:
```
count    37940.000000
mean         0.647870
std          0.215935
min          0.003530
25%          0.511232
50%          0.694130
75%          0.818025
max          0.998560
Name: pct_own, dtype: float64
```

In [22]:
```
#dataset withpct_own > 10%
df_1= df_new[df_new['pct_own']>=0.1]
df_1.shape
```

Out[22]:
```
(37387, 77)
```

In [23]:
```
df_new['second_mortgage'].describe()
```

Out[23]:
```
count    37940.000000
mean         0.029990
std          0.030713
min          0.000000
25%          0.008120
50%          0.022770
75%          0.043023
max          0.608700
Name: second_mortgage, dtype: float64
```

In [24]:
```
df_2 = df_1.sort_values('second_mortgage', ascending=False)
df_2.shape
```

Out[24]:
```
(37387, 77)
```

In [25]:
```
df_top2500=df_2.head(2500)
pd.set_option('display.max_columns', None)
df_top2500
```

Out[25]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code
3285	289712	147	51	Virginia	VA	Farmville	Farmville	Town	23901	434
11980	251185	27	25	Massachusetts	MA	Worcester	Worcester City	City	1610	508
26018	269323	81	36	New York	NY	Corona	Harbor Hills	City	11368	718
7829	251324	3	24	Maryland	MD	Glen Burnie	Glen Burnie	CDP	21061	410
2077	235788	57	12	Florida	FL	Tampa	Egypt Lake-leto	City	33614	813
...	...	...	...	...	...	...	...	...	...	...
24727	293454	117	55	Wisconsin	WI	Sheboygan	Sheboygan City	City	53081	920
12048	294317	37	56	Wyoming	WY	Rock Srpings	Rock Springs City	City	82902	307
885	238703	13	13	Georgia	GA	Auburn	Auburn City	City	30011	770
2830	230409	77	6	California	CA	Stockton	Lincoln Village	City	95219	209
26539	271985	49	39	Ohio	OH	Columbus	Bexley City	Village	43207	614

2500 rows × 77 columns

In [26]:
```
import plotly.express as px
import pandas as pd
fig = px.scatter_geo(df_top2500,lat='lat',lon='lng', scope='usa', color=df_top2500['stat
fig.update_layout(title = 'Top 2,500 locations with second mortgage is the highest and p
fig.show()
```

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

In [27]:
```
df_new['bad_debt'] = df_new['second_mortgage'] + df_new['home_equity'] - df_new['home_ec
df_new.head()
```

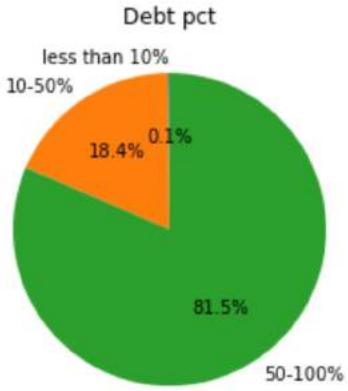
Out[27]:

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

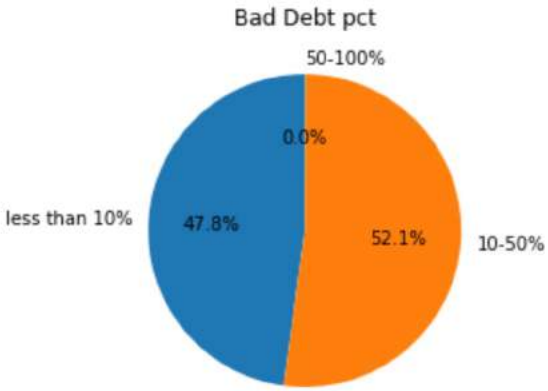
c) Create pie charts to show overall debt and bad debt

In [28]:
```
df_new['bins_debt'] = pd.cut(df_new['debt'],bins=[0,0.1,.5,1], labels=["less than 10%",'
df_new['bins_bad_debt'] = pd.cut(df_new['bad_debt'],bins=[0,0.1,.5,1], labels=["less tha
```

In [29]:
```
df_new.groupby(['bins_debt']).size().plot(kind='pie',subplots=True,startangle=90, autopc
plt.title('Debt pct')
plt.ylabel("")
plt.show()
```



In [30]:
```
df_new.groupby(['bins_bad_debt']).size().plot(kind='pie',subplots=True,startangle=90, au
plt.title('Bad Debt pct')
plt.ylabel("")
plt.show()
```

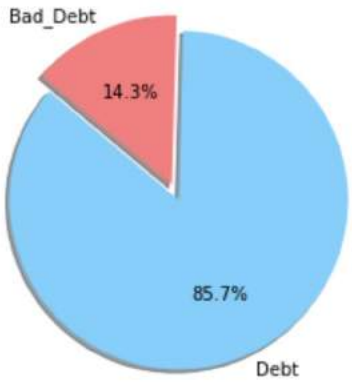


In [31]:
```
import matplotlib.pyplot as plt

# Data to plot
labels = 'Debt', 'Bad_Debt'
sizes = [df_new['debt'].mean()*100, df_new['bad_debt'].mean()*100]
colors = [ 'lightskyblue','lightcoral']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes,explode=explode,labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

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



In [32]:
```
df_new['bad_debt'].mean()*100
```

Out[32]:
```
10.543681128096999
```

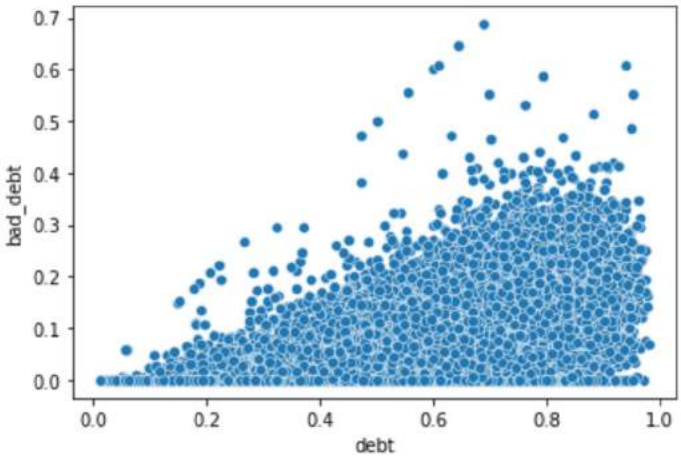
In [33]:
```
df_test=df_new[['UID','bins_debt','bins_bad_debt']]
df_test.groupby(['bins_debt','bins_bad_debt'])[['UID']].size().reset_index()
```

Out[33]:

	bins_debt	bins_bad_debt	0
0	less than 10%	less than 10%	9
1	less than 10%	10-50%	0
2	less than 10%	50-100%	0
3	10-50%	less than 10%	5323
4	10-50%	10-50%	555
5	10-50%	50-100%	0
6	50-100%	less than 10%	11983
7	50-100%	10-50%	18310
8	50-100%	50-100%	12

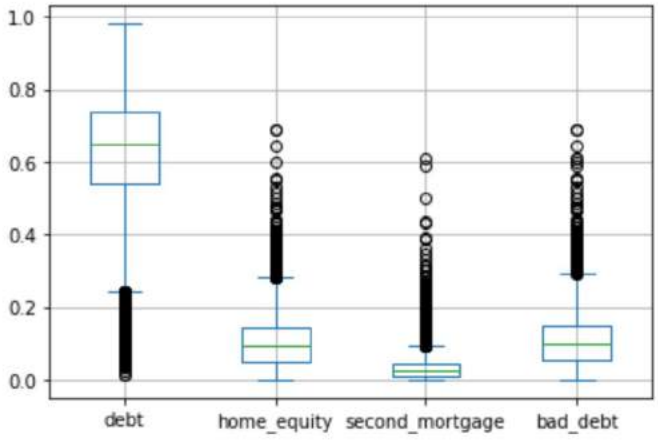
In [34]:
```
sns.scatterplot(df_new['debt'],df_new['bad_debt'])
```

Out[34]:
```
<AxesSubplot:xlabel='debt', ylabel='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

In [35]:
```
df_box_plot = df_new[['debt','home_equity','second_mortgage','bad_debt']]
df_box_plot.plot.box(grid='False')
plt.show()
```

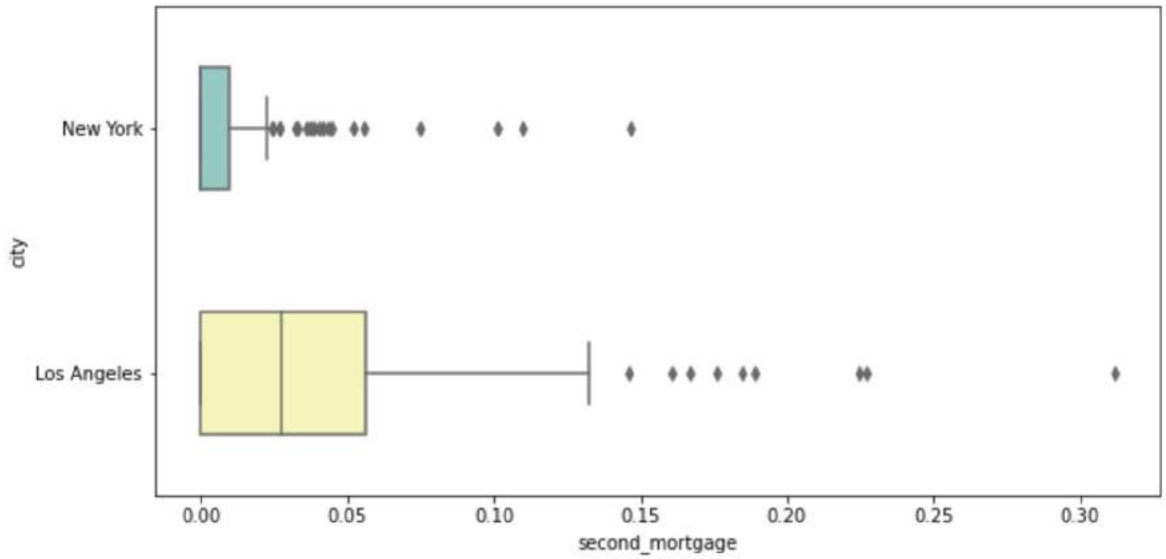


In [36]:
```
df_box_New_York=df_new.loc[df_new['city'] == 'New York']
df_box_Los_Angeles=df_new.loc[df_new['city'] == 'Los Angeles']
df_box_city=pd.concat([df_box_New_York,df_box_Los_Angeles])
df_box_city.head(4)
```

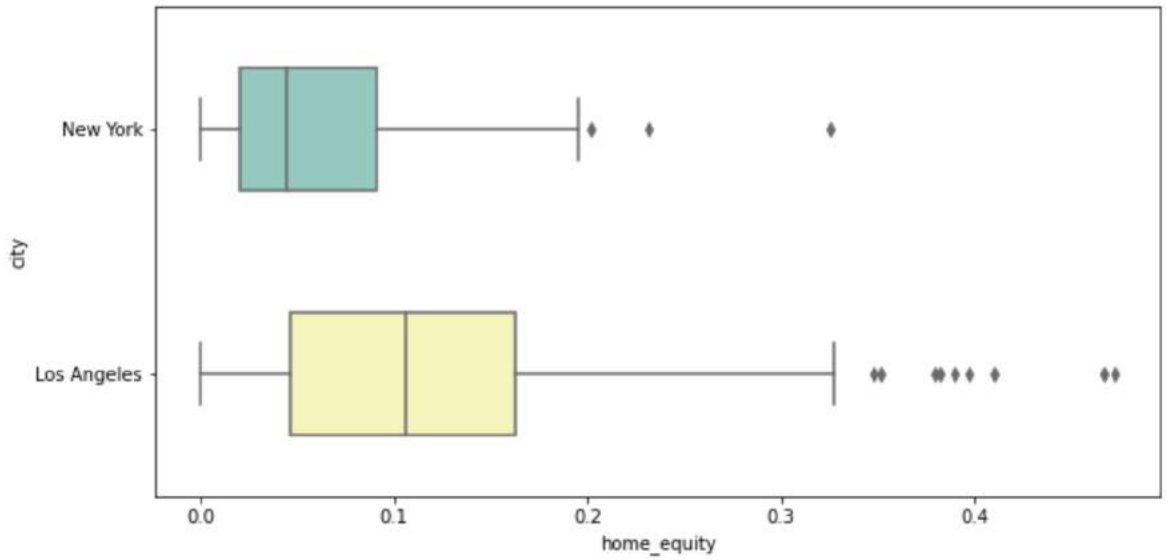
Out[36]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	lr
415	268583	61	36	New York	NY	New York	Mount Vernon City	City	10033	212	40.850557	-73.93097
554	268321	61	36	New York	NY	New York	New York City	City	10002	212	40.710258	-73.9880
770	268552	61	36	New York	NY	New York	Pelham Manor	City	10030	212	40.815930	-73.9432
1255	268565	61	36	New York	NY	New York	Mount Vernon City	City	10032	212	40.832338	-73.9410

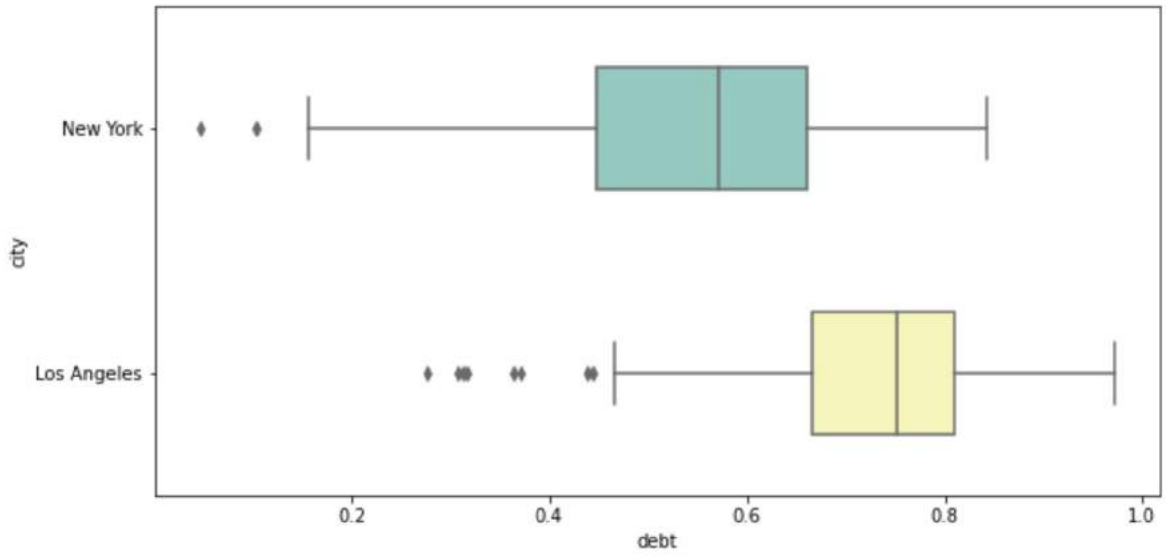
In [37]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.5,palette="Set3")
plt.show()
```



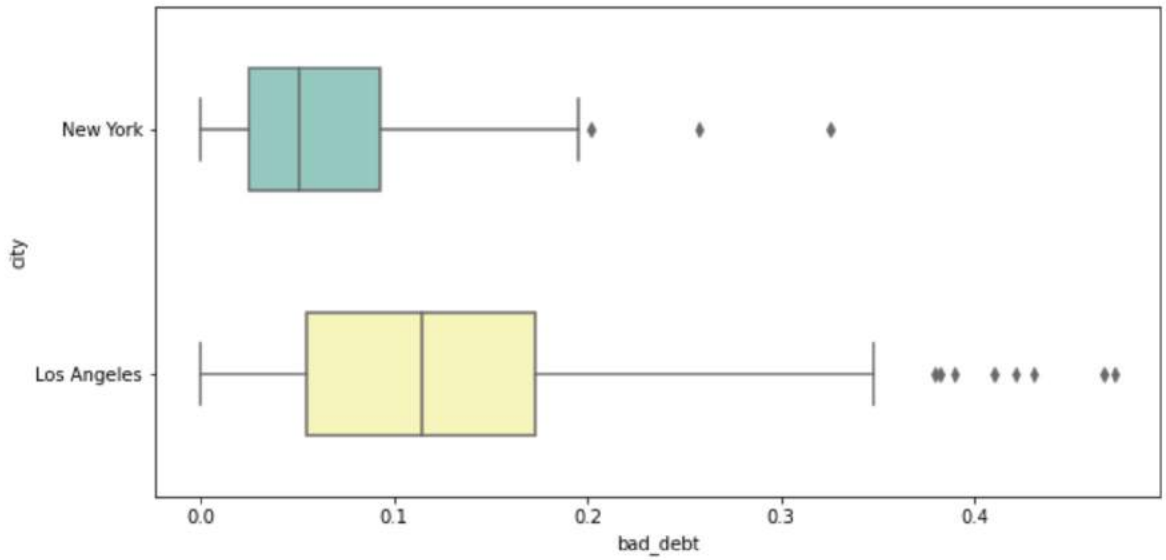
In [38]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



In [39]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```

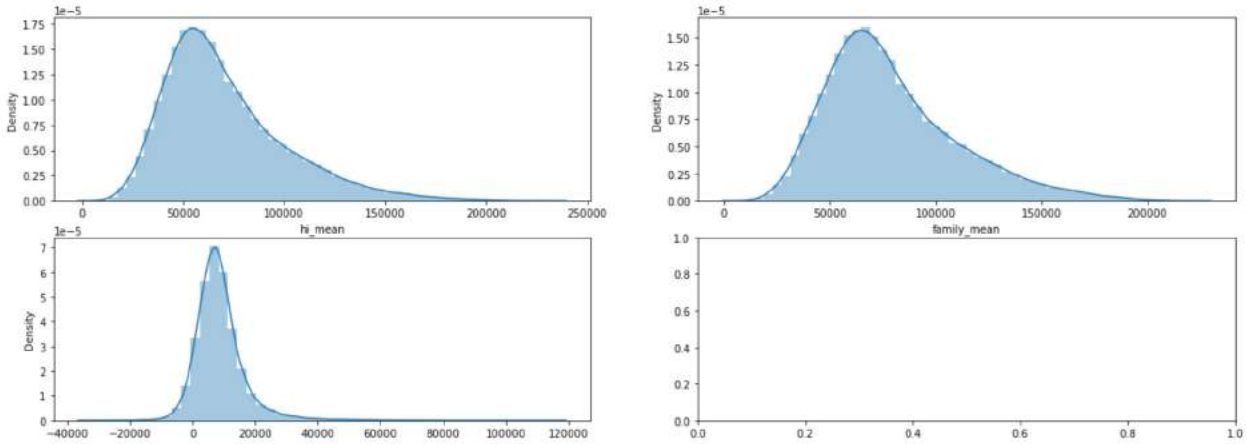


In [40]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='bad_debt', y='city',width=0.5,palette="Set3")
plt.show()
```



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

In [41]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(20,7))
sns.distplot(df_new['hi_mean'],ax=axes[0,0])
sns.distplot(df_new['family_mean'],ax=axes[0,1])
sns.distplot(df_new['family_mean']-df_new['hi_mean'],ax=axes[1,0])
plt.show()
```



In [ ]:

Project Task: Week 2

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.

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

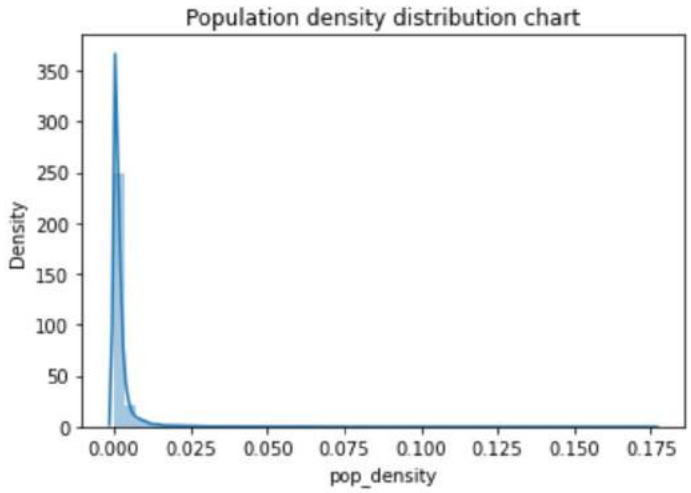
In [42]:
```
df_new['pop_density'] = df_new['pop']/df_new['ALand']
df_new.head()
```

Out[42]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat
2	245683	63	18	Indiana	IN	Danville	Danville	City	46122	317	39.792202
3	279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	927	787	18.396103

In [43]:
```
sns.distplot(df_new['pop_density'])
plt.title('Population density distribution chart')
plt.show()
```



b) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age

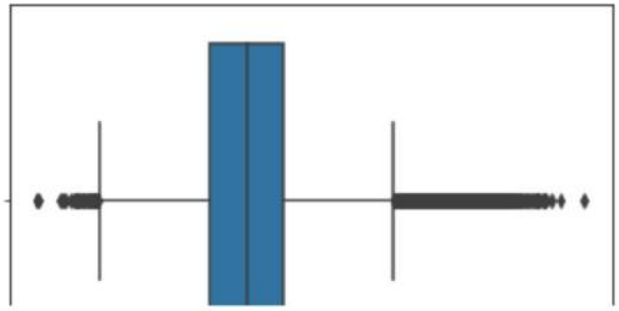
In [44]:
```
x=df_new['male_pop']+df_new['female_pop']
y=df_new['male_age_median']+df_new['female_age_median']
df_new['median_age']=y/2
df_new.head()
```

Out[44]:

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

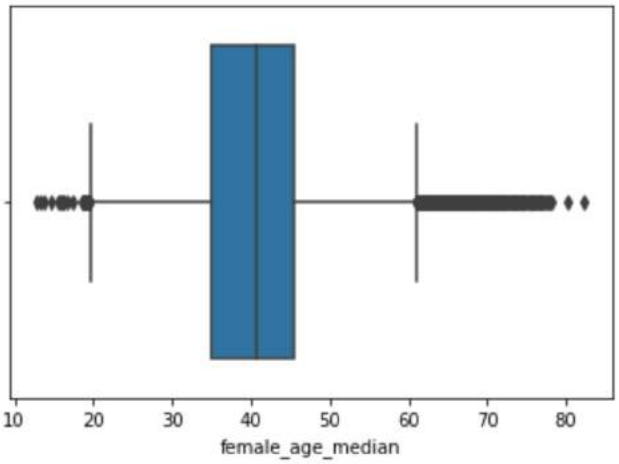
In [45]:
```
sns.boxplot(df_new['male_age_median'])
```

Out[45]:
```
<AxesSubplot:xlabel='male_age_median'>
```



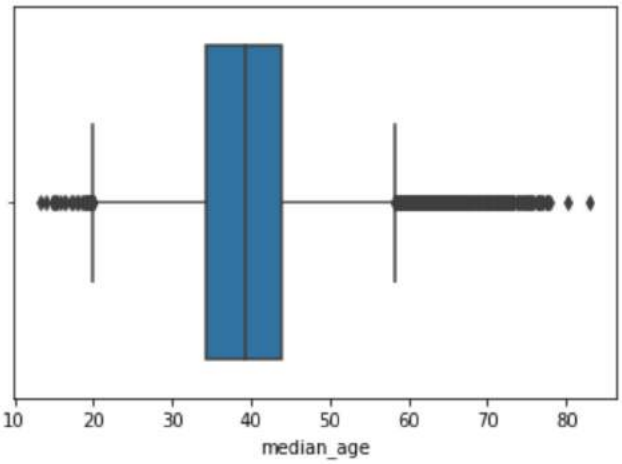
In [46]:
```
sns.boxplot(df_new['female_age_median'])
```

Out[46]: <AxesSubplot:xlabel='female_age_median'>

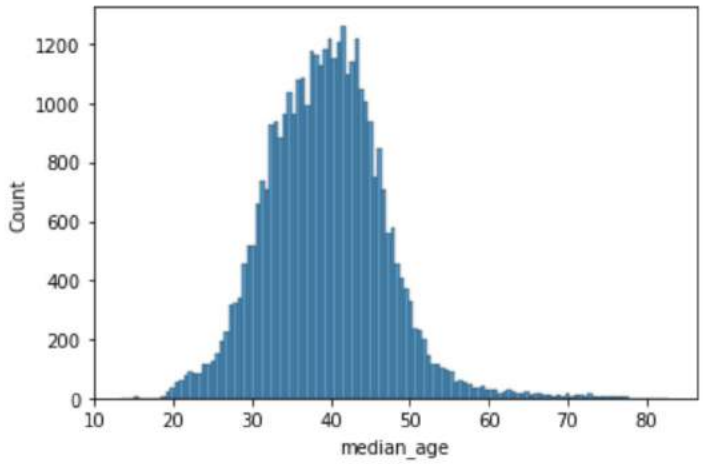


In [47]:
```
sns.boxplot(df_new['median_age'])
```

Out[47]: <AxesSubplot:xlabel='median_age'>



In [48]:
```
sns.histplot(df_new['median_age'])
plt.show()
```



1. Create bins for population into a new variable by selecting appropriate class interval so that the number of categories don't exceed 5 for the ease of analysis.

In [49]:
```
df_new['pop'].describe()
```

Out[49]:
```
count    37940.000000
mean      4385.977570
std       2084.057931
min         38.000000
25%       2956.000000
50%       4106.000000
75%       5470.250000
max      53812.000000
Name: pop, dtype: float64
```

In [50]:
```
df_new.set_index(keys=['UID'],inplace=True)
```

In [51]:
```
df_new.head()
```

Out[51]:

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

In [52]:
```
df_new.shape
```

Out[52]:
```
(37940, 81)
```

In [53]:
```
#df_new.to_csv('df_new.csv')
```

In [54]:
```
df_new.describe()
```

Out[54]:

	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	A
count	37940.000000	37940.000000	37940.000000	37940.000000	37940.000000	37940.000000	3.794000e+04	3.79400
mean	85.756616	28.315604	50162.427860	595.396916	37.508399	-91.303479	1.251229e+08	6.14181
std	98.400480	16.439228	29546.440992	232.356014	5.587125	16.278821	1.158857e+09	2.03641
min	1.000000	1.000000	601.000000	201.000000	17.929085	-166.770979	8.299000e+03	0.00000
25%	29.000000	13.000000	27030.000000	405.000000	33.916015	-97.797392	1.824246e+06	0.00000
50%	63.000000	28.000000	47803.000000	614.000000	38.729103	-86.627658	4.951182e+06	2.72960
75%	109.000000	42.000000	77098.000000	801.000000	41.374493	-79.863073	3.453241e+07	5.22575
max	840.000000	72.000000	99929.000000	989.000000	67.074017	-65.379332	1.039510e+11	2.45322

In [55]:
```
df_new['pop_bins']=pd.cut(df_new['pop'],bins=5,labels=['very low','low','medium','high',
```

In [56]:
```
df_new['pop_bins'].value_counts()
```

Out[56]:
```
very low     37582
low            342
medium          12
high             3
very high        1
Name: pop_bins, dtype: int64
```

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

In [57]:
```
df_new.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

Out[57]:

pop_bins	married	separated	divorced
very low	37582	37582	37582
low	342	342	342
medium	12	12	12
high	3	3	3
very high	1	1	1

In [ ]:
```

```

In [58]:
```
df_new.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"]
```

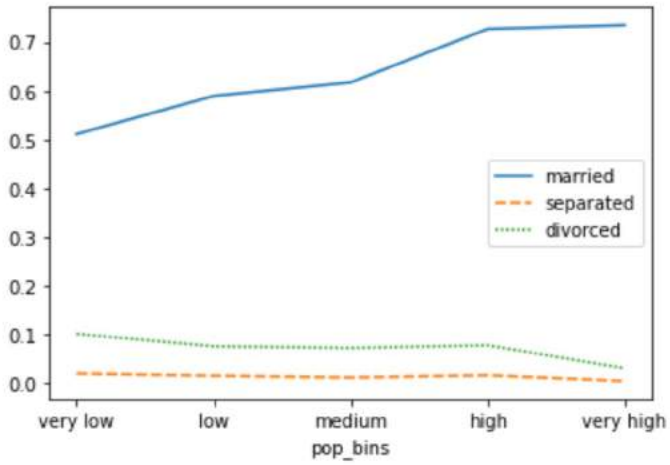
Out[58]:

	married		separated		divorced	
	mean	**median**	**mean**	**median**	**mean**	**median**

pop_bins						
very low	0.511088	0.527645	0.019074	0.013640	0.100761	0.095775
low	0.589286	0.601385	0.014721	0.010125	0.075311	0.069340
medium	0.617047	0.605765	0.011203	0.007745	0.071870	0.069090
high	0.726957	0.736060	0.015663	0.009160	0.077310	0.063050
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.030360

b) Visualize using appropriate chart type

In [59]:
```
pop_bin_married=df_new.groupby(by='pop_bins')[['married','separated','divorced']].agg(['
sns.lineplot(data=pop_bin_married)
plt.show()
```

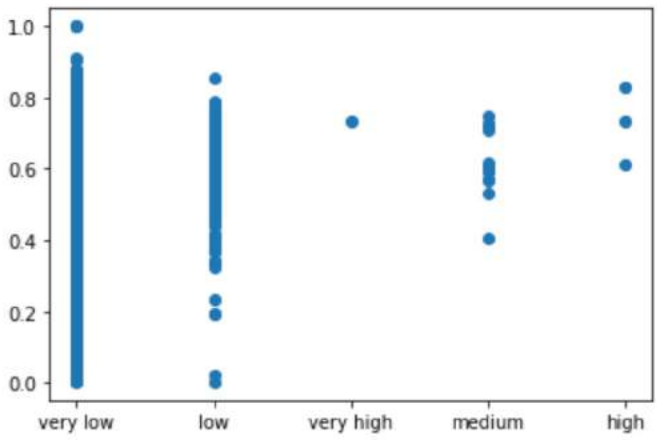


In [60]:
```
pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']).unique()
```

Out[60]:
```
['very low', 'low', 'very high', 'medium', 'high']
Categories (5, object): ['very low' < 'low' < 'medium' < 'high' < 'very high']
```

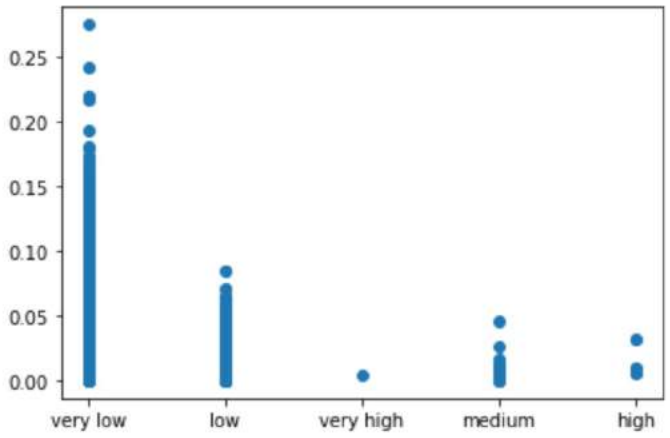
In [61]:
```
plt.scatter(pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']
```

Out[61]:
```
<matplotlib.collections.PathCollection at 0x1cf9b9378b0>
```



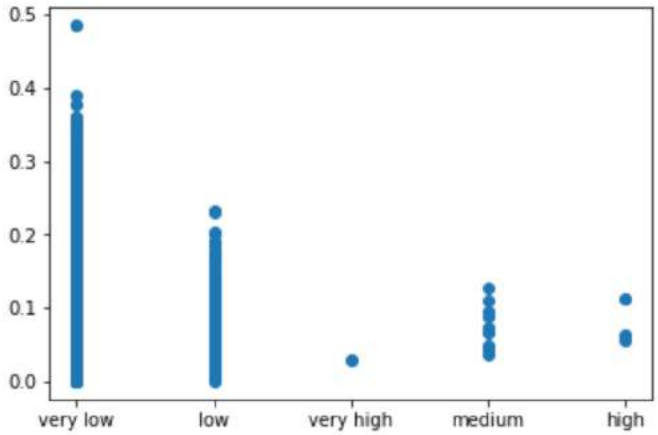
In [62]:
```
plt.scatter(pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']
```

Out[62]:
```
<matplotlib.collections.PathCollection at 0x1cfa89a0f40>
```



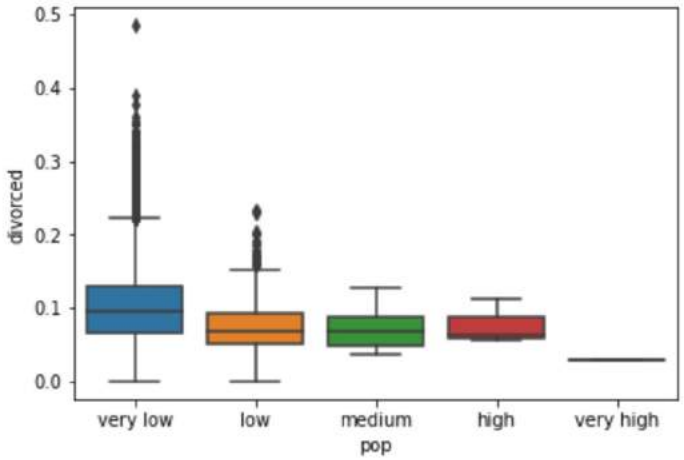
In [63]:
```
plt.scatter(pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']
```

Out[63]:
```
<matplotlib.collections.PathCollection at 0x1cfa89dd6d0>
```



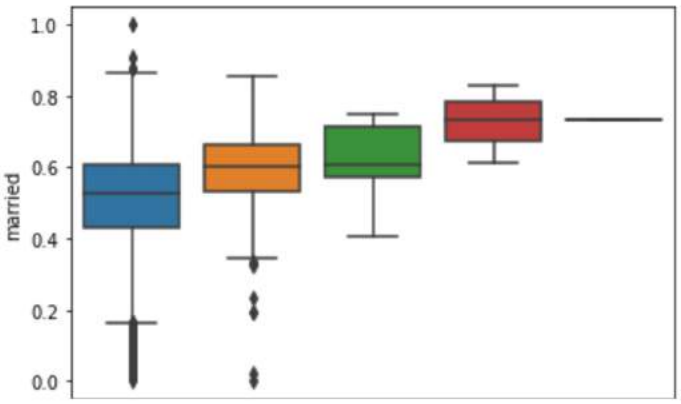
In [64]:
```
sns.boxplot(pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']
```

Out[64]:
```
<AxesSubplot:xlabel='pop', ylabel='divorced'>
```



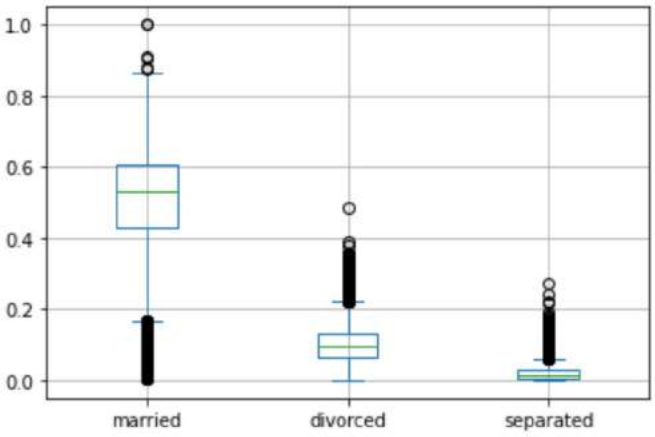
In [65]:
```
sns.boxplot(pd.cut(df_new['pop'],5,labels=['very low','low','medium','high','very high']
```

Out[65]:
```
<AxesSubplot:xlabel='pop', ylabel='married'>
```



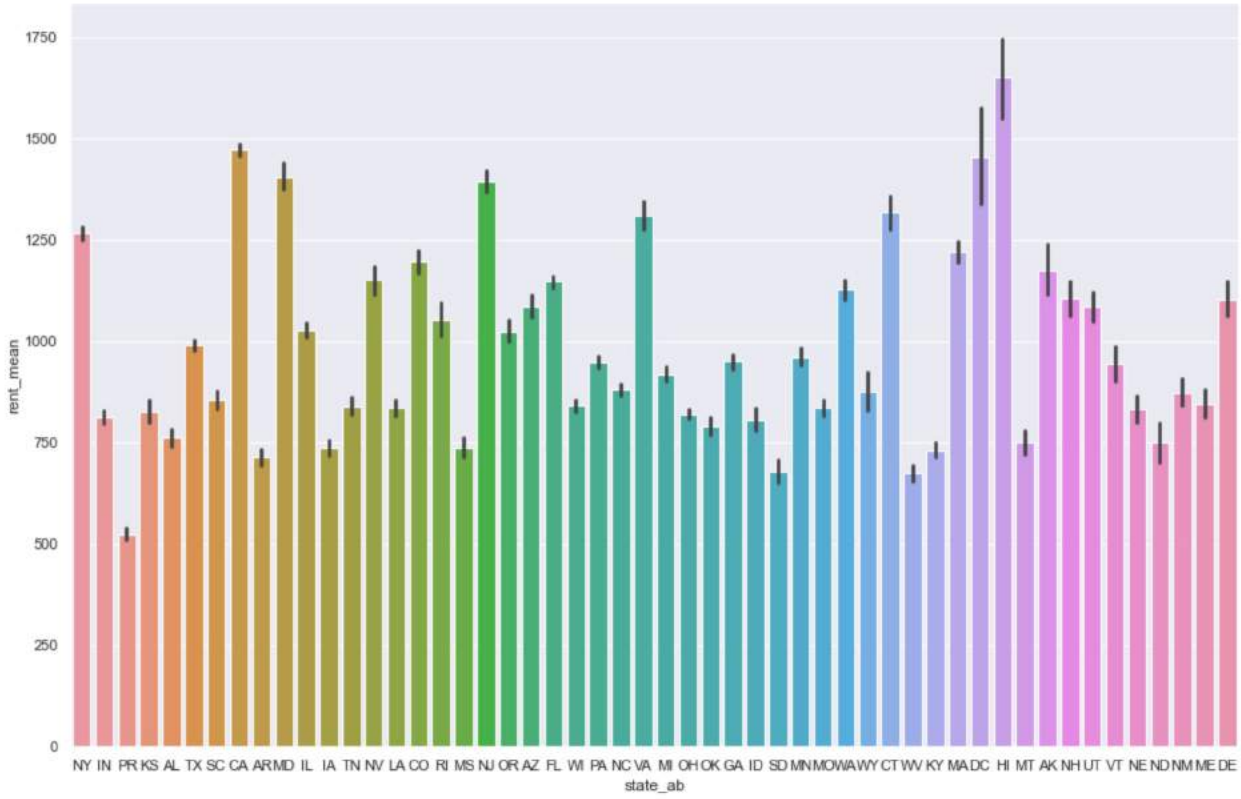
In [66]:
```
df_new[['married','divorced','separated']].plot.box(grid='True')
```

Out[66]:
```
<AxesSubplot:>
```



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

In [67]:
```
sns.set(rc={'figure.figsize':(15.7,10.27)})
sns.barplot(x='state_ab', y="rent_mean",data=df_new)
plt.show()
```



In [68]:
```
#HI,DC,CT,VA,NJ,MD,CA,NY has the top mean rent
```

In [69]:
```
rent_state_mean=df_new.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

Out[69]:

state	mean
Alabama	762.827471
Alaska	1173.006814
Arizona	1087.049918
Arkansas	713.879645
California	1472.168818

In [70]:
```
income_state_mean=df_new.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

Out[70]:

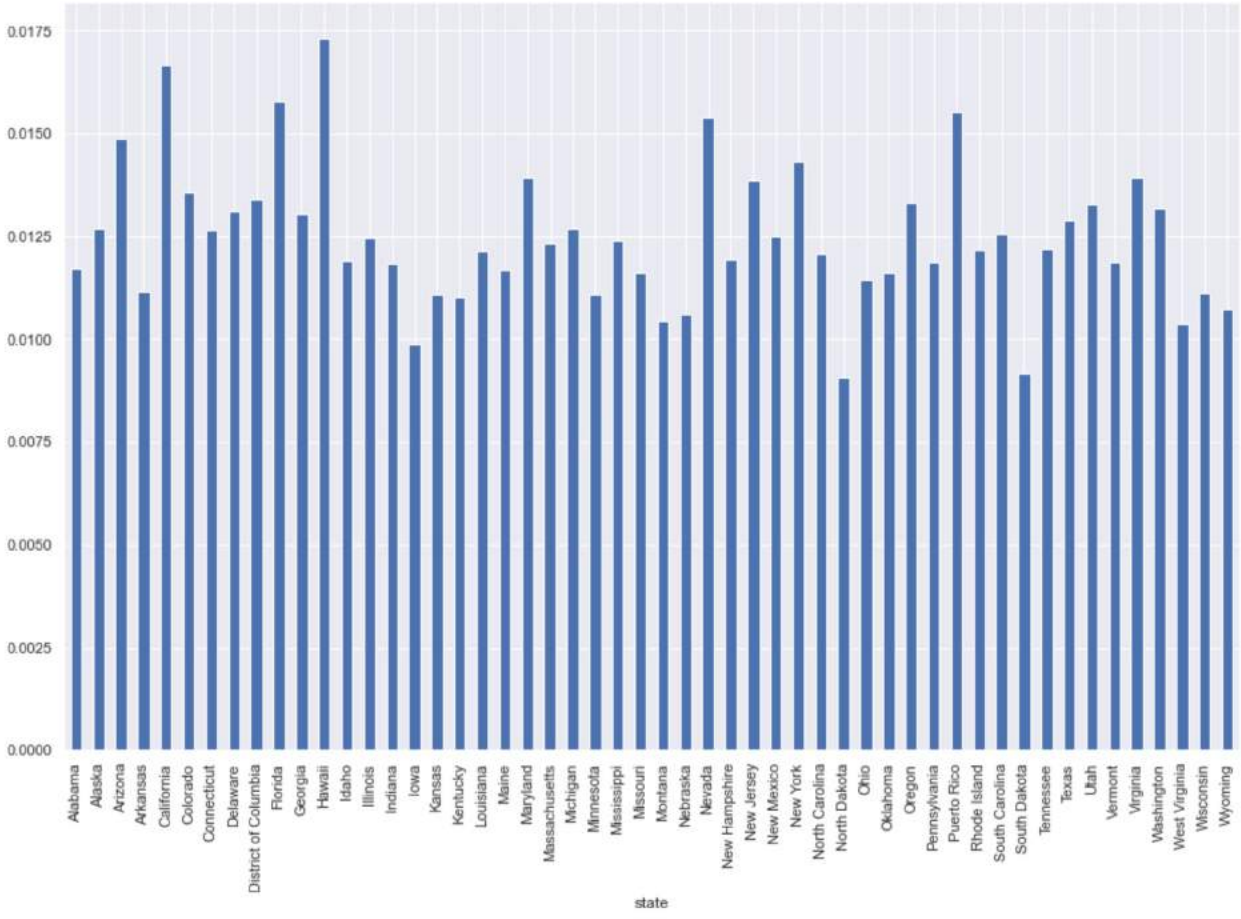
state	mean
Alabama	65183.236790
Alaska	92408.808068
Arizona	73134.038870
Arkansas	63939.536360
California	88315.668424

In [71]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

Out[71]:
```
state
Alabama                 0.011703
Alaska                  0.012694
Arizona                 0.014864
Arkansas                0.011165
California              0.016669
Colorado                0.013581
Connecticut             0.012656
Delaware                0.013116
District of Columbia    0.013414
Florida                 0.015775
Name: mean, dtype: float64
```

In [72]:
```
rent_perc_of_income.plot(kind='bar')
```

Out[72]:
```
<AxesSubplot:xlabel='state'>
```



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

In [73]:
```
df_new.describe()
```

Out[73]:

	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	A
count	37940.000000	37940.000000	37940.000000	37940.000000	37940.000000	37940.000000	3.794000e+04	3.79400

	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	A
mean	85.756616	28.315604	50162.427860	595.396916	37.508399	-91.303479	1.251229e+08	6.14181
std	98.400480	16.439228	29546.440992	232.356014	5.587125	16.278821	1.158857e+09	2.03641
min	1.000000	1.000000	601.000000	201.000000	17.929085	-166.770979	8.299000e+03	0.00000
25%	29.000000	13.000000	27030.000000	405.000000	33.916015	-97.797392	1.824246e+06	0.00000
50%	63.000000	28.000000	47803.000000	614.000000	38.729103	-86.627658	4.951182e+06	2.72960
75%	100.000000	42.000000	77008.000000	801.000000	41.274402	-79.863073	3.453241e+07	5.23575

In [74]:
```
pd.set_option('display.max_rows', None)
df_new.corr()
```

Out[74]:

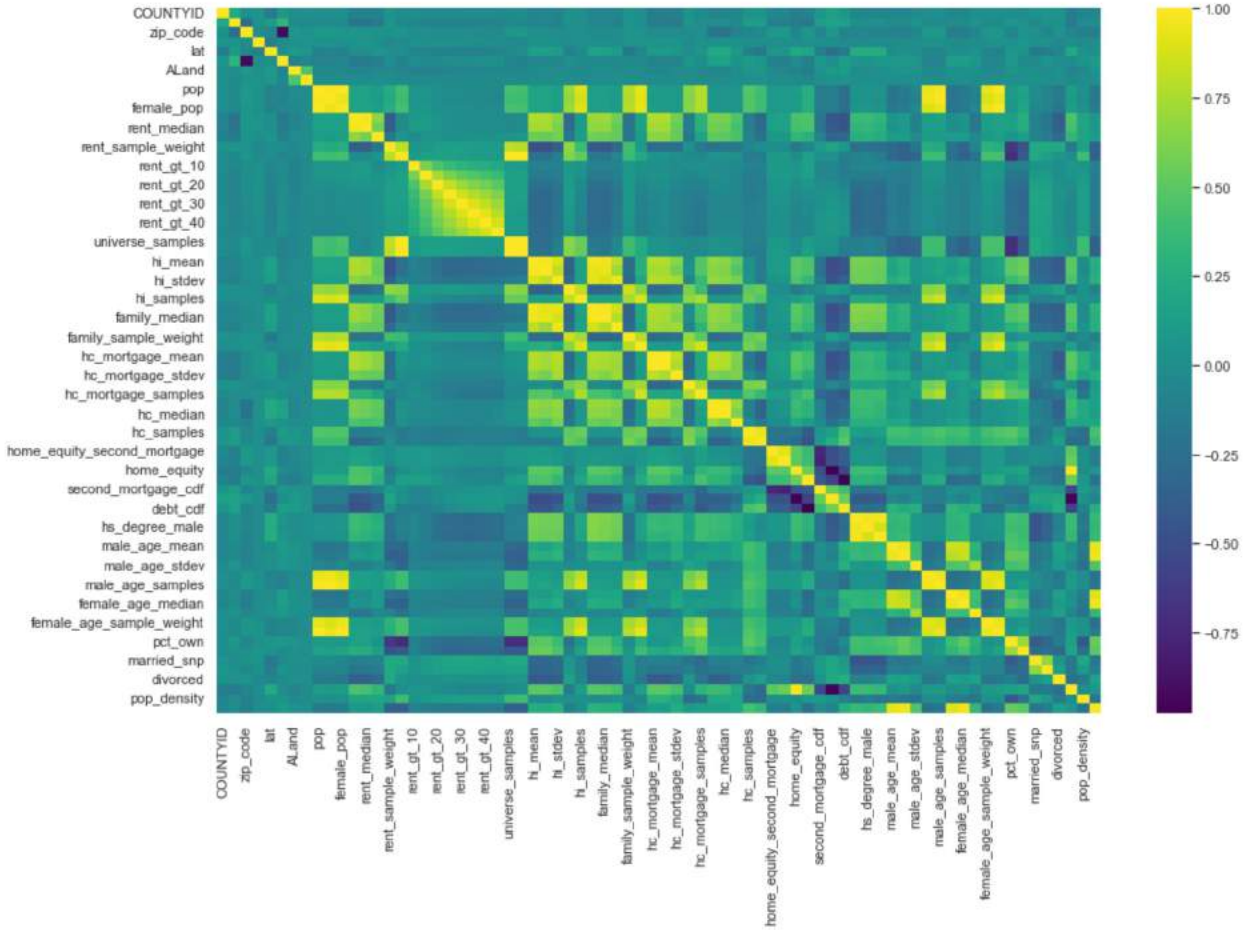
	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AWa
COUNTYID	1.000000	0.223249	0.034647	0.064306	-0.151373	0.070489	0.012028	0.012
STATEID	0.223249	1.000000	-0.260462	0.039777	0.106949	0.317457	-0.015365	-0.026
zip_code	0.034647	-0.260462	1.000000	-0.006592	-0.059264	-0.928379	0.073002	0.031
area_code	0.064306	0.039777	-0.006592	1.000000	-0.123332	-0.011907	0.015436	0.021
lat	-0.151373	0.106949	-0.059264	-0.123332	1.000000	0.010132	0.097511	0.069
lng	0.070489	0.317457	-0.928379	-0.011907	0.010132	1.000000	-0.104591	-0.064
ALand	0.012028	-0.015365	0.073002	0.015436	0.097511	-0.104591	1.000000	0.455
AWater	0.012614	-0.026277	0.031520	0.021942	0.069862	-0.064298	0.455609	1.000
pop	0.001766	-0.029941	0.074612	0.032150	-0.088813	-0.083194	-0.034796	-0.013
male_pop	0.002717	-0.031380	0.091773	0.034063	-0.082497	-0.099852	-0.023487	-0.009
female_pop	0.000783	-0.027616	0.055551	0.029293	-0.092258	-0.064361	-0.044777	-0.016
rent_mean	-0.093614	-0.210439	0.073278	0.043830	0.004122	-0.165121	-0.072570	-0.011
rent_median	-0.091566	-0.204581	0.065961	0.044061	0.001497	-0.155327	-0.070742	-0.011
rent_stdev	-0.088318	-0.156059	0.036994	0.007035	0.059381	-0.124145	-0.036316	0.001
rent_sample_weight	0.044406	0.054716	0.031241	-0.048235	-0.001603	0.000119	-0.047634	-0.017
rent_samples	-0.006799	-0.053043	0.066010	-0.027203	-0.014844	-0.087487	-0.071464	-0.021
rent_gt_10	-0.014474	-0.051764	-0.003307	0.030610	-0.058092	-0.009950	-0.110126	-0.040
rent_gt_15	-0.025461	-0.088639	0.001484	0.031531	-0.082439	-0.026907	-0.104851	-0.033
rent_gt_20	-0.024658	-0.097527	-0.008848	0.025441	-0.107223	-0.021917	-0.091837	-0.033
rent_gt_25	-0.022640	-0.098503	-0.016357	0.017678	-0.109509	-0.012286	-0.080626	-0.028
rent_gt_30	-0.018759	-0.096224	-0.016357	0.019724	-0.119554	-0.008516	-0.066575	-0.023
rent_gt_35	-0.017825	-0.090323	-0.020527	0.010909	-0.116309	-0.000831	-0.059983	-0.022
rent_gt_40	-0.017169	-0.084349	-0.027987	0.008683	-0.107017	0.009996	-0.054802	-0.020
rent_gt_50	-0.018913	-0.069097	-0.039080	-0.001885	-0.086229	0.025061	-0.053918	-0.018
universe_samples	-0.004419	-0.040684	0.059740	-0.025807	-0.035310	-0.080853	-0.062867	-0.018
used_samples	-0.007925	-0.053825	0.068371	-0.026476	-0.010152	-0.090665	-0.071255	-0.020
hi_mean	-0.072582	-0.080783	0.005896	0.018293	0.137150	-0.060503	-0.031739	-0.003

	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AW
hi_median	-0.070316	-0.070412	0.005138	0.021228	0.142288	-0.057636	-0.032844	-0.004
hi_stdev	-0.073695	-0.099050	-0.003233	0.005173	0.114848	-0.052814	-0.021407	0.000
hi_sample_weight	0.029897	0.034544	0.011067	-0.010805	-0.109987	0.012151	-0.026722	-0.016
hi_samples	-0.002144	0.002648	0.016417	-0.002708	-0.025885	-0.018682	-0.042171	-0.018
family_mean	-0.070550	-0.067865	-0.018989	0.003581	0.160362	-0.031893	-0.031126	-0.003
family_median	-0.068293	-0.057443	-0.023256	0.002556	0.159568	-0.025452	-0.032488	-0.004
family_stdev	-0.062086	-0.093832	-0.004973	0.000017	0.119506	-0.045155	-0.020480	0.001
family_sample_weight	0.036300	0.025193	0.051028	0.032747	-0.180237	-0.027357	-0.010018	-0.011
family_samples	0.002387	0.001553	0.038734	0.035338	-0.058296	-0.039166	-0.027413	-0.014
hc_mortgage_mean	-0.135312	-0.160032	-0.017303	0.046836	0.105801	-0.094536	-0.060174	-0.012
hc_mortgage_median	-0.133263	-0.156565	-0.014667	0.044656	0.106065	-0.095320	-0.062065	-0.012
hc_mortgage_stdev	-0.122337	-0.159612	-0.015908	0.043832	0.067421	-0.085563	-0.017309	0.001
hc_mortgage_sample_weight	0.039158	0.051867	-0.016402	-0.018791	-0.014926	0.065753	-0.008205	-0.014
hc_mortgage_samples	-0.019564	-0.017696	0.004034	0.018151	0.056631	-0.007424	-0.038551	-0.017
hc_mean	-0.086999	-0.011053	-0.216183	0.034864	0.223825	0.154541	-0.059961	-0.011
hc_median	-0.086785	-0.002390	-0.219966	0.034059	0.222456	0.161209	-0.061422	-0.011
hc_stdev	-0.052265	-0.056696	-0.093761	0.028254	0.090117	0.040519	-0.008336	0.004
hc_samples	0.037913	0.114769	-0.067723	0.005427	-0.117158	0.105772	0.060981	0.012
hc_sample_weight	0.051545	0.112906	-0.024162	-0.000318	-0.183389	0.072464	0.077688	0.014
home_equity_second_mortgage	-0.042339	-0.112528	0.087592	0.003876	0.076372	-0.109808	-0.047185	-0.016
second_mortgage	-0.051008	-0.124517	0.082149	0.002798	0.080837	-0.112018	-0.049226	-0.016
home_equity	-0.135867	-0.152960	-0.070119	0.001123	0.230329	-0.008446	-0.086985	-0.027
debt	-0.085225	-0.165843	0.068039	0.015074	0.192351	-0.123056	-0.127466	-0.045
second_mortgage_cdf	0.060659	0.123858	-0.066577	0.002992	-0.132694	0.094711	0.049121	0.018
home_equity_cdf	0.147356	0.161748	0.077658	0.005976	-0.262645	0.003090	0.092983	0.029
debt_cdf	0.080287	0.161463	-0.077652	-0.023463	-0.162840	0.133810	0.112945	0.036
hs_degree	-0.059736	0.018147	-0.078213	-0.029935	0.248925	0.057308	-0.003536	0.005
hs_degree_male	-0.057903	0.008675	-0.060172	-0.024961	0.237814	0.036742	-0.007391	0.004
hs_degree_female	-0.057060	0.027360	-0.090762	-0.032326	0.244635	0.074052	0.002452	0.004
male_age_mean	-0.066640	-0.024432	-0.109642	-0.022248	-0.003477	0.085288	0.049007	0.007
male_age_median	-0.066135	-0.018611	-0.110347	-0.013158	0.019630	0.087698	0.056136	0.010
male_age_stdev	-0.010973	0.056192	-0.062852	-0.004249	0.011691	0.071159	0.038950	-0.002
male_age_sample_weight	0.005001	-0.029789	0.088005	0.034673	-0.077975	-0.096417	-0.023281	-0.008
male_age_samples	0.002717	-0.031380	0.091773	0.034063	-0.082497	-0.099852	-0.023487	-0.009
female_age_mean	-0.057982	-0.016216	-0.146269	-0.020117	-0.013343	0.122589	0.020762	-0.006
female_age_median	-0.055959	-0.010214	-0.141619	-0.010217	0.000999	0.121505	0.034206	-0.002
female_age_stdev	-0.000478	0.064272	-0.063457	-0.011792	0.038302	0.073058	0.028911	-0.007

	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	AW
female_age_sample_weight	0.003390	-0.026547	0.054365	0.026518	-0.086969	-0.063297	-0.043595	-0.016
female_age_samples	0.000783	-0.027616	0.055551	0.029293	-0.092258	-0.064361	-0.044777	-0.016
pct_own	-0.004437	0.073040	-0.071671	0.019443	0.061400	0.093341	0.054980	0.011
married	-0.017703	0.028886	0.024862	0.060701	0.044538	-0.025699	0.031719	-0.000
married_snp	0.038756	-0.032214	0.016023	0.028576	-0.167407	-0.034053	0.011661	0.027
separated	0.065830	0.031420	-0.056039	0.020255	-0.147699	0.057109	-0.005661	0.006
divorced	0.048039	0.016948	0.046103	-0.044143	-0.063114	-0.006857	0.025018	0.005

In [75]:
```
sns.heatmap(df_new.corr(),cmap='viridis')
```

Out[75]: <AxesSubplot:>



Observation 3:

- by looking at the correlation heatmap, we do see that there are some strong correlation between variables and there could some multicolinearity

In [76]:
```
#checking population
df_new[['pop','male_pop','female_pop']].corr()
```

Out[76]:

	pop	male_pop	female_pop

	pop	male_pop	female_pop
pop	1.000000	0.984100	0.984823
male_pop	0.984100	1.000000	0.938337

In [77]:
```
# we see there is a very strong multicolinearity among pop, male_pop and female_pop
```

In [78]:
```
#checking rent
df_new[['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']].corr()
```

Out[78]:

	rent_mean	rent_median	rent_stdev	rent_sample_weight	rent_samples	rent_gt_10	rent_gt_15
rent_mean	1.000000	0.975979	0.665785	-0.391182	-0.014559	0.095773	0.106060
rent_median	0.975979	1.000000	0.580078	-0.385243	-0.019736	0.095269	0.107023
rent_stdev	0.665785	0.580078	1.000000	-0.185833	0.064933	-0.018841	0.035956
rent_sample_weight	-0.391182	-0.385243	-0.185833	1.000000	0.804153	0.055492	0.094917
rent_samples	-0.014559	-0.019736	0.064933	0.804153	1.000000	0.104436	0.149063
rent_gt_10	0.095773	0.095269	-0.018841	0.055492	0.104436	1.000000	0.609580
rent_gt_15	0.106060	0.107023	0.035956	0.094917	0.149063	0.609580	1.000000
rent_gt_20	0.050094	0.053387	0.037596	0.127686	0.156110	0.450762	0.743384
rent_gt_25	-0.000114	0.001855	0.028073	0.147011	0.152015	0.362383	0.605665
rent_gt_30	-0.008527	-0.007827	0.020477	0.129381	0.131357	0.311816	0.520556
rent_gt_35	-0.004879	-0.005325	0.023289	0.113833	0.116217	0.279606	0.461483
rent_gt_40	-0.008183	-0.009851	0.025984	0.106772	0.106509	0.254925	0.420216
rent_gt_50	-0.010033	-0.012691	0.037028	0.099440	0.100129	0.216948	0.363503

In [79]:
```
# we see there is a very strong multicolinearity among rent_mean, rent_median, rent_stde
```

This shows that there are some issue on multi colinearity and we will have to apply dimensional reduction technique

Project Task: Week 3

Data Pre-processing:

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

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

In [80]:
```
df_new[['hs_degree','hs_degree_male','hs_degree_female']].corr()
```

Out[80]:

	hs_degree	hs_degree_male	hs_degree_female
hs_degree	1.000000	0.968482	0.966934
hs_degree_male	0.968482	1.000000	0.875508
hs_degree_female	0.966934	0.875508	1.000000

In [81]:
```
df_new[['male_age_mean','male_age_median','male_age_stdev','female_age_mean','female_age
```

Out[81]:

	male_age_mean	male_age_median	male_age_stdev	female_age_mean	female_age_median	fem
male_age_mean	1.000000	0.948182	0.390410	0.857487	0.839516	
male_age_median	0.948182	1.000000	0.393806	0.792040	0.832776	
male_age_stdev	0.390410	0.393806	1.000000	0.453226	0.473152	
female_age_mean	0.857487	0.792040	0.453226	1.000000	0.949699	
female_age_median	0.839516	0.832776	0.473152	0.949699	1.000000	
female_age_stdev	0.308847	0.279515	0.726450	0.398555	0.378614	

In [82]:
```
df_new[['home_equity_second_mortgage','second_mortgage']].corr()
```

Out[82]:

	home_equity_second_mortgage	second_mortgage
home_equity_second_mortgage	1.000000	0.920907
second_mortgage	0.920907	1.000000

In [83]:
```
df_new[['debt','debt_cdf','bad_debt']].corr()
```

Out[83]:

	debt	debt_cdf	bad_debt
debt	1.000000	-0.974812	0.563814
debt_cdf	-0.974812	1.000000	-0.561652
bad_debt	0.563814	-0.561652	1.000000

Observations:

- all these variables shows that the variables has multi co-linearity and are latent variables

In [84]:
```
from factor_analyzer import FactorAnalyzer
```

In [85]:
```
df_copy=df_new.copy()
df_copy.reset_index(inplace=True)
```

In [86]:
```
df_copy.head(2)
```

Out[86]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.5
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.2

In [87]:
```
df_factor_analysis=df_copy.drop(['UID','state','state_ab','city','place','type',
                                 'pop_bins','bins_debt','bins_bad_debt','median_age','pc
                                ,'AWater','ALand','lat','lng'], axis=1)
```

In [88]:
```
df_factor_analysis.head(2)
```

Out[88]:

	COUNTYID	STATEID	zip_code	area_code	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	
0	53	36	13346	315	5230	2612	2618	769.38638	784.0	232.63967	
1	141	18	46616	574	2633	1349	1284	804.87924	848.0	253.46747	

In [89]:
```
df_factor_analysis.shape
```

Out[89]:
```
(37940, 67)
```

In [90]:
```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value,p_value=calculate_bartlett_sphericity(df_factor_analysis)
chi_square_value, p_value
```

Out[90]:
```
(9945914.596384186, 0.0)
```

In [91]:
```
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(df_factor_analysis)
kmo_model
```

Out[91]:
```
0.5084609945441668
```

In [92]:
```
df_factor_analysis.shape
```

Out[92]:
```
(37940, 67)
```

In [93]:
```
fa = FactorAnalyzer()
fa.fit(df_factor_analysis)
eigen_values, vectors = fa.get_eigenvalues()
print(eigen_values)
```

```
[ 1.54250725e+01  1.21743863e+01  8.24933454e+00  4.58566295e+00
  3.81656205e+00  2.93348721e+00  2.08089321e+00  1.50048049e+00
  1.38890628e+00  1.23500044e+00  1.14100969e+00  1.12512367e+00
  9.66116268e-01  8.61413571e-01  7.70019789e-01  7.59482062e-01
  6.92100518e-01  6.31497456e-01  5.73237814e-01  5.32794342e-01
  4.78294287e-01  4.43871425e-01  4.10328016e-01  3.71157876e-01
  3.43547660e-01  3.14303356e-01  3.03583046e-01  2.53365279e-01
  2.43783279e-01  2.35219169e-01  2.05388262e-01  2.00334431e-01
  1.84742235e-01  1.70947536e-01  1.53712067e-01  1.44640274e-01
  1.33596543e-01  1.16312564e-01  1.08432039e-01  9.74002917e-02
  9.25719739e-02  9.04591681e-02  7.24438683e-02  5.81721739e-02
  5.08811244e-02  4.07950865e-02  3.32753717e-02  2.91054497e-02
  2.71852187e-02  2.26752680e-02  2.07341090e-02  1.94091287e-02
  1.62150394e-02  1.55865413e-02  1.43567720e-02  1.20084096e-02
  7.95970238e-03  6.63569531e-03  5.40022385e-03  3.77892202e-03
  3.11734032e-03  9.72704194e-04  7.19872292e-04  3.66201455e-17
 -2.38284618e-17 -1.47634317e-16 -1.80440488e-16]
```

In [94]:
```
pd.DataFrame(eigen_values.round(5))
```

Out[94]:

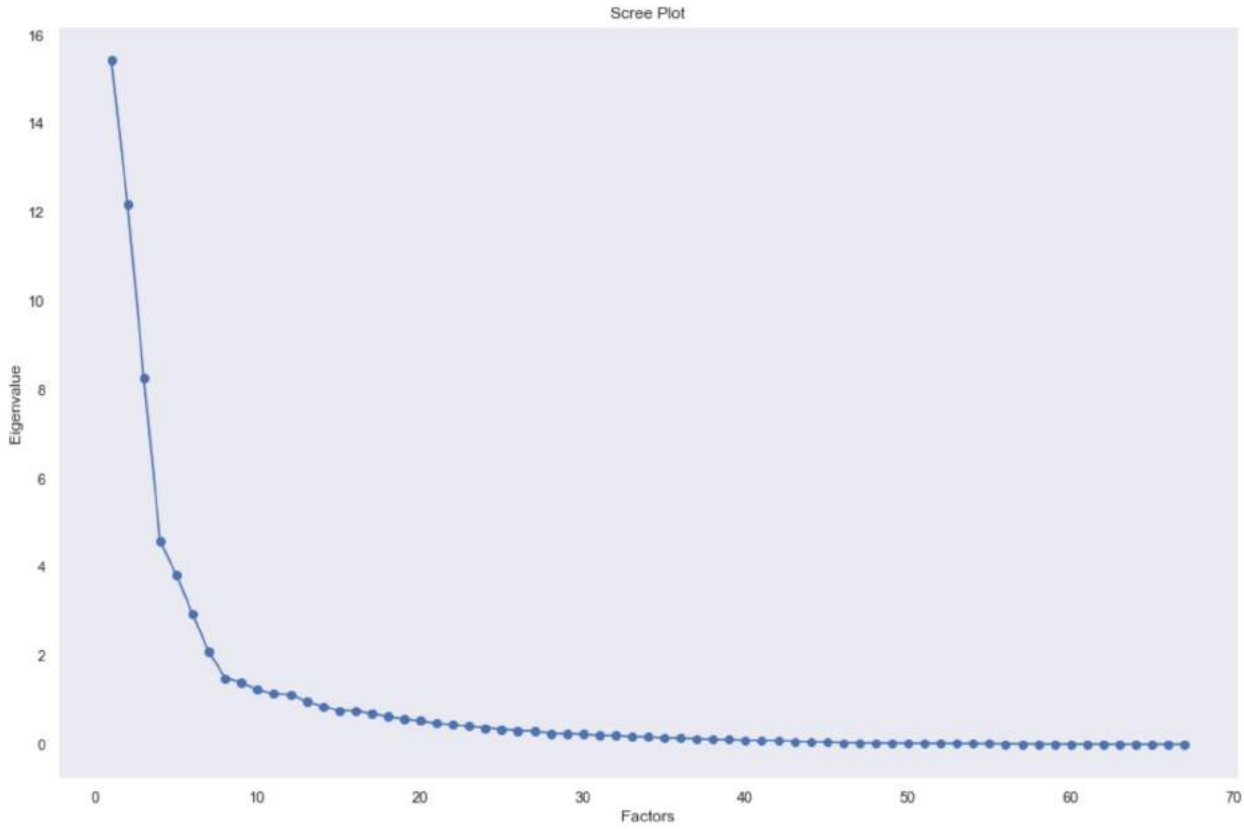
	0
0	15.42507
1	12.17439
2	8.24933
3	4.58566
4	3.81656
5	2.93349
6	2.08089
7	1.50048
8	1.38891
9	1.23500
10	1.14101
11	1.12512
12	0.96612
13	0.86141
14	0.77002
15	0.75948
16	0.69210
17	0.63150
18	0.57324
19	0.53279
20	0.47829
21	0.44387
22	0.41033
23	0.37116
24	0.34355

	0
25	0.31430
26	0.30358
27	0.25337
28	0.24378
29	0.23522
30	0.20539
31	0.20033
32	0.18474
33	0.17095
34	0.15371
35	0.14464
36	0.13360
37	0.11631
38	0.10843
39	0.09740
40	0.09257
41	0.09046
42	0.07244
43	0.05817
44	0.05088
45	0.04080
46	0.03328
47	0.02911
48	0.02719
49	0.02268
50	0.02073
51	0.01941
52	0.01622
53	0.01559
54	0.01436
55	0.01201
56	0.00796
57	0.00664
58	0.00540
59	0.00378
60	0.00312
61	0.00097

	0
62	0.00072
63	0.00000
64	-0.00000

Here, you can see only for 12-factors eigenvalues are greater than one. It means we need to choose only 12 factors (or unobserved variables).

In [95]:
```
# Create scree plot using matplotlib
plt.scatter(range(1,df_factor_analysis.shape[1]+1),eigen_values)
plt.plot(range(1,df_factor_analysis.shape[1]+1),eigen_values)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigenvalue')
plt.grid()
plt.show()
```



In [96]:
```
fa = FactorAnalyzer(12,rotation="varimax")
fa.fit(df_factor_analysis)
loadings=fa.loadings_
```

In [97]:
```
df_loading=pd.DataFrame.from_records(loadings)
df_loading
```

Out[97]:

	0	1	2	3	4	5	6	7	8	9	
0	-0.100492	0.018485	-0.032377	-0.039335	-0.096324	-0.063308	0.037089	-0.101302	-0.000779	0.009051	0.(
1	-0.127602	0.008444	-0.052866	-0.100782	-0.081439	-0.108566	0.084919	-0.131926	-0.029557	0.050318	0.(

	0	1	2	3	4	5	6	7	8	9	
2	-0.030153	0.042685	0.026776	-0.042604	-0.107495	0.031581	-0.028655	-0.017374	-0.010419	-0.011520	-0.(
3	0.040163	0.029465	-0.050610	-0.000255	-0.027493	-0.004941	-0.051934	-0.014269	0.053315	-0.002254	0.(
4	0.120668	0.973098	0.108268	-0.009805	-0.103072	0.040101	-0.025766	0.029929	0.019684	-0.022248	0.(
5	0.117292	0.950125	0.085334	-0.024260	-0.105939	0.040076	-0.038331	0.021088	0.005266	-0.066794	0.(
6	0.119499	0.960287	0.128358	0.004535	-0.096487	0.039243	-0.010894	0.037981	0.033730	0.023364	-0.(
7	0.813714	0.063558	-0.106256	0.036579	-0.024556	0.073627	0.074251	0.135549	0.156613	-0.106941	0.(
8	0.767119	0.061974	-0.115436	0.028825	-0.036875	0.073212	0.057327	0.137297	0.160905	-0.111215	0.(
9	0.673231	0.028013	0.057340	0.087372	0.051607	0.038542	0.039704	0.055287	0.007866	0.008792	0.(
10	-0.327135	0.210200	0.785844	0.050875	-0.125159	-0.028292	-0.026715	-0.016930	-0.010421	0.030868	0.(
11	0.013485	0.293756	0.923495	0.057358	-0.141315	0.005102	-0.006289	0.043718	0.047172	-0.089342	0.(
12	-0.030470	0.047988	0.037730	0.220528	-0.037617	0.052392	0.007186	0.084416	0.562990	-0.039890	0.(
13	-0.018333	0.032125	0.073523	0.393147	-0.043577	0.055336	-0.023082	0.052298	0.787274	-0.025368	0.(
14	-0.046469	-0.001095	0.093093	0.604197	-0.035744	0.036606	-0.073308	0.029372	0.616040	-0.005124	0.(
15	-0.072305	-0.016281	0.094618	0.761936	-0.034018	0.019784	-0.099036	0.016856	0.414797	0.003657	0.(
16	-0.076028	-0.016895	0.067077	0.875174	-0.041360	0.005557	-0.091838	0.003874	0.241192	-0.002275	0.(
17	-0.067778	-0.021349	0.049913	0.942446	-0.045951	-0.011101	-0.075681	-0.000229	0.101568	-0.012032	0.(
18	-0.063655	-0.029092	0.044279	0.941958	-0.053136	-0.015950	-0.067665	-0.007837	0.027482	-0.022279	0.(
19	-0.053245	-0.045310	0.053614	0.834379	-0.060288	-0.019427	-0.068406	-0.020360	0.009715	-0.029579	0.(
20	-0.007541	0.314219	0.918787	0.057358	-0.130626	-0.005368	-0.015532	0.018232	0.040321	-0.083984	0.(
21	0.017887	0.297424	0.920790	0.051088	-0.137457	0.007208	-0.005310	0.049202	0.048562	-0.084514	0.(
22	0.860698	0.108942	-0.275225	-0.208042	0.029026	0.070010	0.233407	0.116961	-0.062484	0.034662	-0.(
23	0.808916	0.120639	-0.316403	-0.227735	0.000064	0.075098	0.220450	0.145561	-0.048315	0.027441	-0.(
24	0.865675	0.057674	-0.116141	-0.106288	0.107446	0.042683	0.229437	0.021211	-0.113222	0.047237	-0.(
25	-0.335801	0.758219	0.478103	0.063405	0.122114	-0.014623	0.012549	-0.076488	0.039472	0.040419	-0.(
26	0.094889	0.908687	0.303081	-0.055809	0.111669	0.028845	0.152147	0.007517	0.009681	0.022593	-0.(
27	0.858056	0.071520	-0.190801	-0.190082	0.084346	0.055496	0.309963	0.074393	-0.079227	0.018146	-0.(
28	0.830823	0.070483	-0.211780	-0.193619	0.062196	0.049702	0.290759	0.081708	-0.076308	0.023086	-0.(
29	0.789796	0.041707	-0.032721	-0.088201	0.125204	0.045809	0.268145	0.010816	-0.096854	0.014492	-0.(
30	-0.323769	0.842779	0.138154	0.044184	0.009784	-0.003697	-0.175582	-0.036334	0.048461	0.101048	0.(
31	0.127968	0.956916	-0.044473	-0.074072	0.030498	0.045504	0.030795	0.049509	0.013500	0.105106	-0.(
32	0.945071	-0.010661	0.062085	0.010316	0.025899	0.098447	-0.049805	0.086533	0.027786	-0.051770	-0.(
33	0.929379	-0.015507	0.058189	0.015015	0.005011	0.090946	-0.053362	0.092729	0.030944	-0.050011	-0.(
34	0.773789	0.003266	0.036292	-0.013019	0.171980	0.091231	0.003896	0.005468	-0.027497	0.008934	-0.(
35	-0.298904	0.766941	-0.245273	-0.109350	0.124782	0.036008	0.208517	0.137629	-0.022223	0.091128	-0.(
36	0.217896	0.807882	-0.281338	-0.107404	0.076173	0.116788	0.224234	0.239395	0.020117	0.052563	-0.(
37	0.856241	-0.039253	0.065650	-0.006528	0.021384	0.017403	-0.012844	0.054368	0.027827	-0.036737	-0.(
38	0.819400	-0.039036	0.064744	-0.004013	0.006286	0.014273	-0.011982	0.058863	0.031256	-0.040141	-0.(

	0	1	2	3	4	5	6	7	8	9	
39	0.678270	-0.011992	0.070565	-0.000544	0.115612	-0.024454	-0.011277	-0.091566	-0.030732	0.016697	-0.(
40	-0.106899	0.614185	-0.242053	-0.072906	0.423725	-0.122658	0.061831	-0.446114	-0.083456	0.128969	-0.(
41	-0.318365	0.558830	-0.235313	-0.067236	0.377577	-0.131630	0.028031	-0.442768	-0.095011	0.123000	-0.(
42	0.034485	0.035701	0.038478	0.008625	-0.093351	0.921199	-0.000310	0.037435	0.034454	-0.057118	0.(
43	0.064580	0.028764	0.040282	0.017529	-0.093020	0.933763	-0.011669	0.032204	0.040790	-0.060396	0.(
44	0.430409	0.043332	-0.023992	-0.014107	0.036743	0.563552	0.136421	0.373014	0.025936	0.008808	-0.(
45	0.329740	0.173341	0.003998	-0.034700	-0.246939	0.294681	0.156139	0.748708	0.116562	-0.105297	-0.(
46	-0.094819	-0.117949	0.087741	0.025939	0.007540	-0.795231	-0.086434	-0.112476	-0.024998	-0.044913	0.(
47	-0.426184	-0.066482	0.041508	0.019753	-0.062443	-0.571732	-0.165902	-0.397226	-0.026905	-0.035509	0.'
48	-0.334913	-0.167583	0.005701	0.029147	0.265859	-0.288187	-0.129576	-0.748252	-0.118820	0.138437	0.(
49	0.376211	0.044079	-0.069694	-0.186167	0.190722	0.090107	0.852485	0.067197	-0.021365	0.002971	-0.'
50	0.391266	0.045547	-0.048543	-0.180505	0.179549	0.086957	0.772052	0.082162	-0.019707	0.007010	-0.'
51	0.342869	0.044562	-0.096006	-0.189479	0.203396	0.087660	0.787429	0.062742	-0.030571	-0.006947	-0.'
52	0.126619	-0.103021	-0.144241	-0.074711	0.905372	-0.060588	0.122956	-0.096707	-0.046277	0.089740	-0.(
53	0.169842	-0.064800	-0.237541	-0.103960	0.855665	-0.040837	0.117201	-0.049880	-0.053423	0.092979	-0.(
54	-0.040301	0.013361	-0.211946	-0.029166	0.299334	-0.026067	0.011370	-0.082405	-0.028472	0.837600	-0.(
55	0.093262	0.898373	0.096125	0.023522	-0.164837	0.028587	-0.019501	-0.026425	-0.000587	-0.153787	-0.(
56	0.117292	0.950125	0.085334	-0.024260	-0.105939	0.040076	-0.038331	0.021088	0.005266	-0.066794	0.(
57	0.080059	-0.111876	-0.088632	-0.035061	0.873978	-0.051565	0.118257	-0.090420	-0.012515	0.199178	-0.(
58	0.114805	-0.069522	-0.209759	-0.063827	0.857777	-0.032890	0.105980	-0.051376	-0.021530	0.196344	-0.(
59	-0.109759	-0.010521	-0.110411	-0.019062	0.256036	-0.033757	-0.013601	-0.081051	-0.041715	0.709563	0.(
60	0.094493	0.899435	0.145685	0.051694	-0.164354	0.026339	0.010316	-0.015636	0.023659	-0.087648	-0.(
61	0.119499	0.960287	0.128358	0.004535	-0.096487	0.039243	-0.010894	0.037981	0.033730	0.023364	-0.(
62	0.152435	0.203582	-0.751715	-0.168608	0.309577	0.021688	0.209134	-0.000592	-0.070394	0.197175	-0.'
63	0.307484	0.238897	-0.470101	-0.247829	0.272582	0.008372	0.127383	0.042257	-0.055632	0.299076	-0.(
64	-0.110683	-0.063988	0.175799	0.117159	-0.071910	-0.023446	-0.272379	-0.013439	0.021734	-0.051118	0.(

In [98]:
```
var=fa.get_factor_variance()
df_var=pd.DataFrame.from_records(var)
df_var.rename(index={0: 'SS Loadings',1:'Proportion Var',2:'Cumulative Var'})
```

Out[98]:

	0	1	2	3	4	5	6	7	8	9
SS Loadings	12.326217	11.793199	5.286944	4.905650	4.381709	3.348579	2.879746	2.138547	1.740513	1.648562
Proportion Var	0.183973	0.176018	0.078910	0.073219	0.065399	0.049979	0.042981	0.031919	0.025978	0.024605
Cumulative Var	0.183973	0.359991	0.438901	0.512120	0.577518	0.627497	0.670478	0.702397	0.728375	0.752980

Total 79% cumulative Variance explained by the 12 factors.

In [99]:
```
df_copy.head(2)
```

Out[99]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.5
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.2

In [100...
```
X = df_copy.drop('hc_mortgage_mean',axis=1)
y=df_copy['hc_mortgage_mean']
```

In [101...
```
X_transformed = fa.fit_transform(X.select_dtypes(exclude=('object','category')))
```

In [102...
```
X_transformed.shape
```

Out[102...
```
(37940, 12)
```

In [103...
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_transformed,y,train_size=0.7, ranc
```

In [104...
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
```

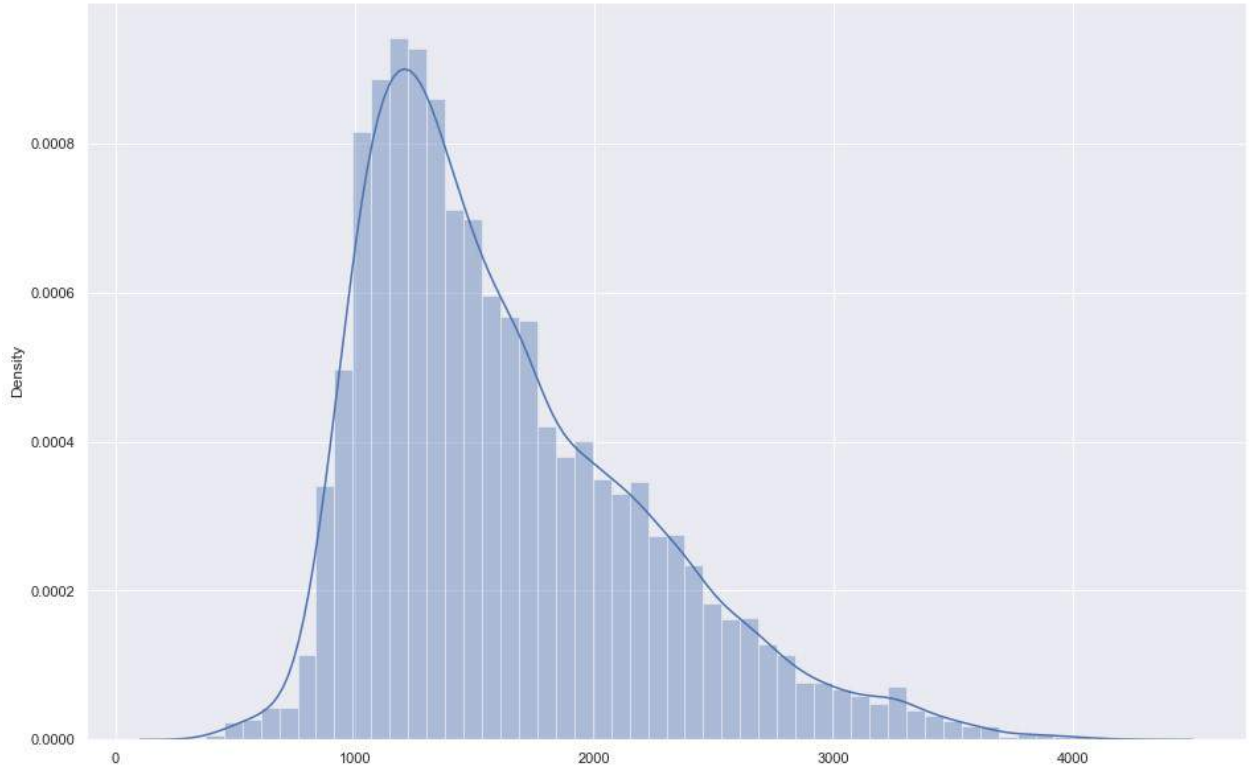
In [105...
```
from sklearn.metrics import r2_score,mean_squared_error
print('r-squared value: ',r2_score(y_pred = y_pred_train, y_true = y_train ))
print('mean suared error: ',mean_squared_error(y_pred = y_pred_train, y_true = y_train))
```

```
r-squared value:  0.8927422339028221
mean suared error:  41770.87529718914
```

In [106...
```
print('r-squared value: ',r2_score(y_pred=y_pred_test, y_true=y_test))
print('mean suared error: ',mean_squared_error(y_pred=y_pred_test, y_true=y_test))
```

```
r-squared value:  0.892977825574893
mean suared error:  41553.74290054049
```

In [107...
```
sns.distplot(y_pred_test)
plt.show()
```



Project Task: Week 4

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

Applying PCA

In [108...
```
# At nation level
```

In [109...
```
df_modelling = df_copy.copy()
df_modelling.head()
```

Out[109...

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

In [110...
```
#df_modelling.to_csv('df_modelling.csv')
```

In [111...
```
X_pca = df_modelling.drop(['type','zip_code','area_code','UID','COUNTYID','STATEID','sta
                           'pop_bins','pop_density','median_age','lat','lng','hc_mortgage

y=df_modelling['hc_mortgage_mean']
X_pca.head()
```

Out[111...

	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample_weight	r
0	202183361.0	1699120	5230	2612	2618	769.38638	784.0	232.63967	272.34441	
1	1560828.0	100363	2633	1349	1284	804.87924	848.0	253.46747	312.58622	
2	69561595.0	284193	6881	3643	3238	742.77365	703.0	323.39011	291.85520	
3	1105793.0	0	2700	1141	1559	803.42018	782.0	297.39258	259.30316	
4	2554403.0	0	5637	2586	3051	938.56493	881.0	392.44096	1005.42886	

In [112...
```
#X_pca['type'].unique()
```

In [ ]:
```
```

In [113...
```
X_pca.head()
```

Out[113...

	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample_weight	r
0	202183361.0	1699120	5230	2612	2618	769.38638	784.0	232.63967	272.34441	
1	1560828.0	100363	2633	1349	1284	804.87924	848.0	253.46747	312.58622	
2	69561595.0	284193	6881	3643	3238	742.77365	703.0	323.39011	291.85520	
3	1105793.0	0	2700	1141	1559	803.42018	782.0	297.39258	259.30316	
4	2554403.0	0	5637	2586	3051	938.56493	881.0	392.44096	1005.42886	

In [114...
```
from sklearn.decomposition import PCA
pca_12 = PCA(n_components=12)
pca_12.fit(X_pca)
X_pca_12 = pca_12.transform(X_pca)
X_pca_12.shape
```

Out[114...
```
(37940, 12)
```

In [115...
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_pca_12,y,train_size=0.7, random_st
```

In [ ]:
```
```

In [116...
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
```

In [117...
```
from sklearn.metrics import r2_score,mean_squared_error
```

In [118...
```
r2_score(y_pred = y_pred_train, y_true = y_train )
```

Out[118...
```
0.8993880981551801
```

In [119...
```
mean_squared_error(y_pred = y_pred_train, y_true = y_train)
```

Out[119...
```
39182.684464687794
```

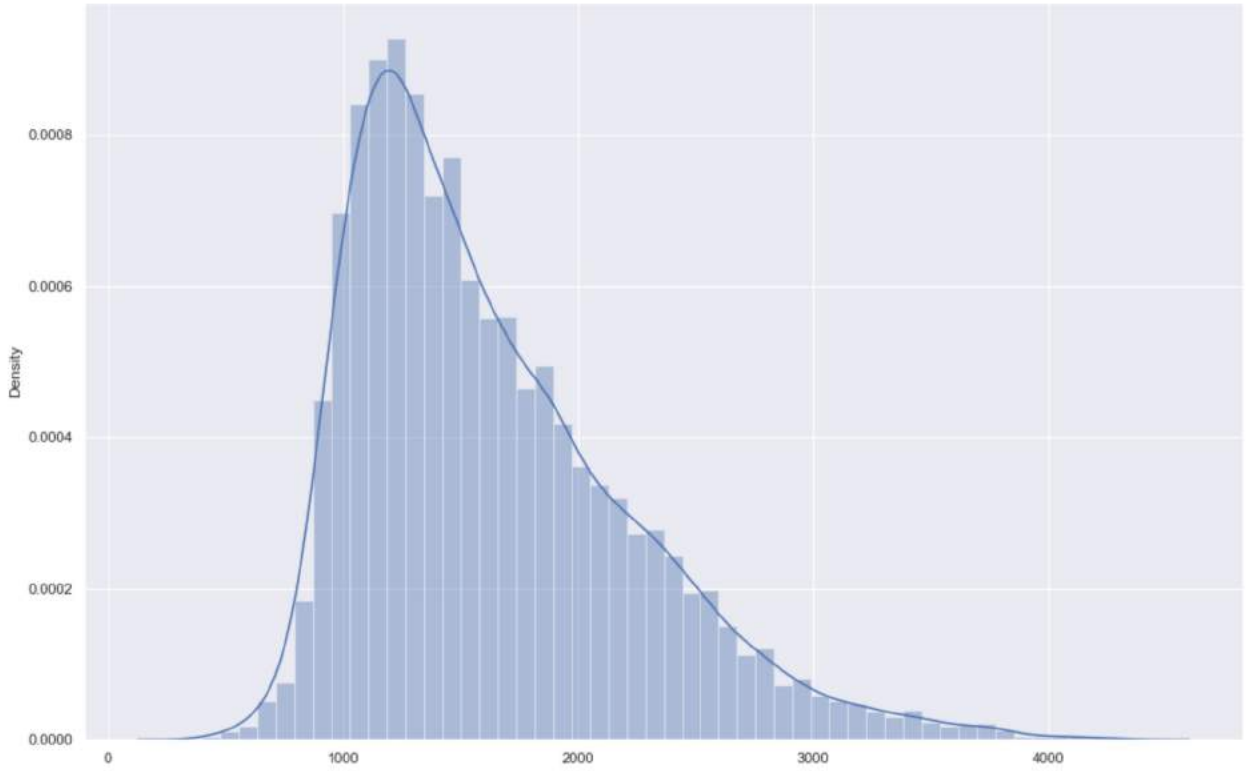
In [120...
```
r2_score(y_pred=y_pred_test, y_true=y_test)
```

Out[120...
```
0.8997550694922984
```

In [121...
```
mean_squared_error(y_pred=y_pred_test, y_true=y_test)
```

Out[121...
```
38922.327001630765
```

In [122...
```
sns.distplot(y_pred_test)
plt.show()
```



Observations:

- R squared value of 89% is achieved
- the predicted variable is somewhat normally distriuted

In [123...

```
# State level
```

In [124...

```
X_pca = df_modelling.drop(['type','zip_code','area_code','UID','COUNTYID','STATEID','sta
                           'pop_bins','pop_density','median_age','lat','lng','hc_mortgage

y=df_modelling['hc_mortgage_mean']
X_pca.head()
```

Out[124...

	state	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample_v
0	New York	202183361.0	1699120	5230	2612	2618	769.38638	784.0	232.63967	272
1	Indiana	1560828.0	100363	2633	1349	1284	804.87924	848.0	253.46747	312
2	Indiana	69561595.0	284193	6881	3643	3238	742.77365	703.0	323.39011	291
3	Puerto Rico	1105793.0	0	2700	1141	1559	803.42018	782.0	297.39258	259
4	Kansas	2554403.0	0	5637	2586	3051	938.56493	881.0	392.44096	1005

In [125...

```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
X_pca['state']=lb.fit_transform(X_pca['state'])
```

In [126...
```
X_pca.head()
```

Out[126...

	state	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample_we
0	32	202183361.0	1699120	5230	2612	2618	769.38638	784.0	232.63967	272.3
1	14	1560828.0	100363	2633	1349	1284	804.87924	848.0	253.46747	312.5
2	14	69561595.0	284193	6881	3643	3238	742.77365	703.0	323.39011	291.8
3	39	1105793.0	0	2700	1141	1559	803.42018	782.0	297.39258	259.3
4	16	2554403.0	0	5637	2586	3051	938.56493	881.0	392.44096	1005.4

In [127...
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_pca_12,y,train_size=0.7, random_st
```

In [128...
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
```

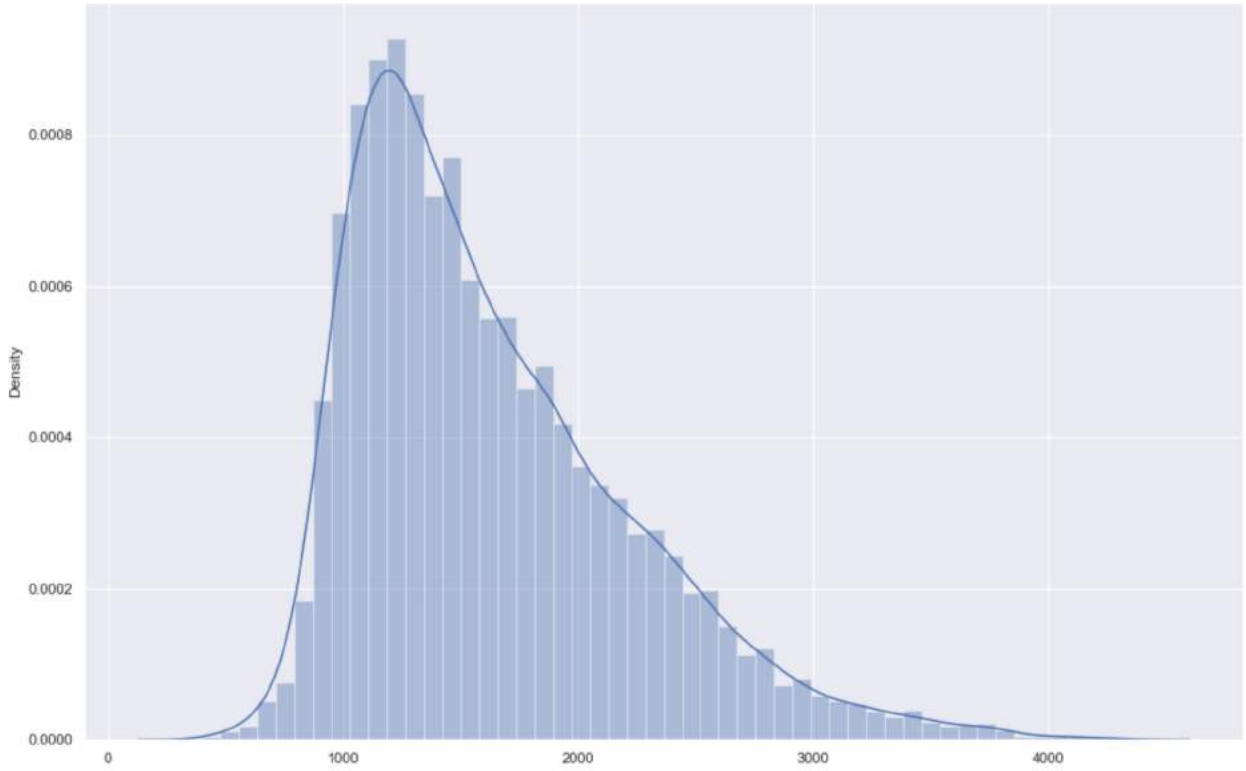
In [129...
```
from sklearn.metrics import r2_score,mean_squared_error
print('r-squared value: ',r2_score(y_pred = y_pred_train, y_true = y_train ))
print('mean suared error: ',mean_squared_error(y_pred = y_pred_train, y_true = y_train))
```

```
r-squared value:  0.8993880981551801
mean suared error:  39182.684464687794
```

In [130...
```
print('r-squared value: ',r2_score(y_pred=y_pred_test, y_true=y_test))
print('mean suared error: ',mean_squared_error(y_pred=y_pred_test, y_true=y_test))
```

```
r-squared value:  0.8997550694922984
mean suared error:  38922.327001630765
```

In [131...
```
sns.distplot(y_pred_test)
plt.show()
```



Observations:

- R squared value of 89% is achieved
- the predicted variable is somewhat normally distriuted
- It appears if we consider the state or without state, the R suqred is almost the same

In [132...
```
# city level
```

In [133...
```
X_pca = df_modelling.drop(['UID','COUNTYID','STATEID','state_ab','bins_debt','bins_bad_d
                           'pop_bins','pop_density','median_age','lat','lng','hc_mortgage

y=df_modelling['hc_mortgage_mean']
X_pca.head()
```

Out[133...

	state	city	place	type	zip_code	area_code	ALand	AWater	pop	male_pop	female_pop
0	New York	Hamilton	Hamilton	City	13346	315	202183361.0	1699120	5230	2612	2618
1	Indiana	South Bend	Roseland	City	46616	574	1560828.0	100363	2633	1349	1284
2	Indiana	Danville	Danville	City	46122	317	69561595.0	284193	6881	3643	3238
3	Puerto Rico	San Juan	Guaynabo	Urban	927	787	1105793.0	0	2700	1141	1559
4	Kansas	Manhattan	Manhattan City	City	66502	785	2554403.0	0	5637	2586	3051

In [134...
```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
X_pca['state']=lb.fit_transform(X_pca['state'])
X_pca['city']=lb.fit_transform(X_pca['city'])
X_pca['place']=lb.fit_transform(X_pca['place'])
X_pca['type']=lb.fit_transform(X_pca['type'])
X_pca.head()
```

Out[134...

	state	city	place	type	zip_code	area_code	ALand	AWater	pop	male_pop	female_pop	rent_mean
0	32	2933	4296	2	13346	315	202183361.0	1699120	5230	2612	2618	769.38638
1	14	6773	9032	2	46616	574	1560828.0	100363	2633	1349	1284	804.87924
2	14	1700	2457	2	46122	317	69561595.0	284193	6881	3643	3238	742.77365
3	39	6378	4227	4	927	787	1105793.0	0	2700	1141	1559	803.42018
4	16	4235	6241	2	66502	785	2554403.0	0	5637	2586	3051	938.56493

In [135...
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_pca_12,y,train_size=0.7, random_st
```

In [136...
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
```

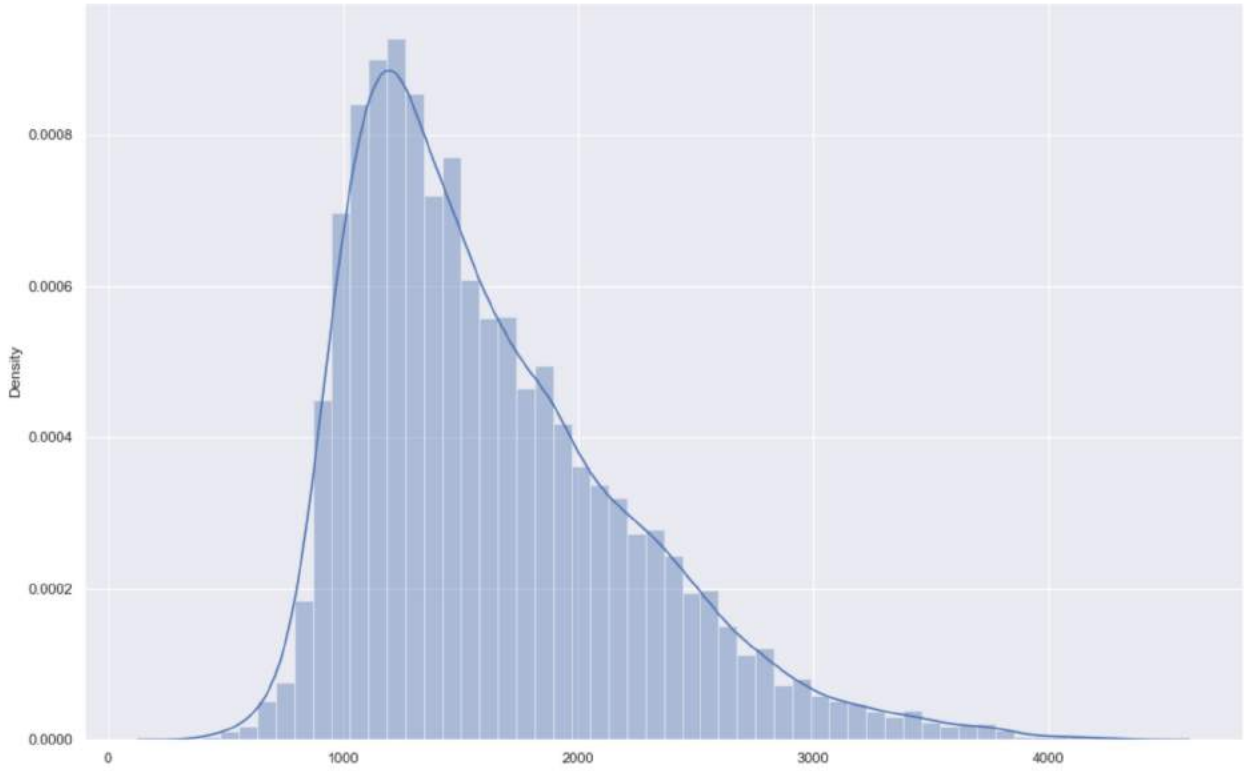
In [137...
```
from sklearn.metrics import r2_score,mean_squared_error
print('r-squared value: ',r2_score(y_pred = y_pred_train, y_true = y_train ))
print('mean suared error: ',mean_squared_error(y_pred = y_pred_train, y_true = y_train))
```

```
r-squared value:  0.8993880981551801
mean suared error:  39182.684464687794
```

In [138...
```
print('r-squared value: ',r2_score(y_pred=y_pred_test, y_true=y_test))
print('mean suared error: ',mean_squared_error(y_pred=y_pred_test, y_true=y_test))
```

```
r-squared value:  0.8997550694922984
mean suared error:  38922.327001630765
```

In [139...
```
sns.distplot(y_pred_test)
plt.show()
```



Observations:

- the variables, state,city, place, type is not making any diference to the Rsquared value

Doing PCA on set of variables

In [140...

```
df_copy.head()
```

Out[140...

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

In [141...
```
from sklearn.decomposition import PCA
pca=PCA(n_components=1)

Highschool_graduation_rates=pca.fit_transform(df_copy[['hs_degree','hs_degree_male','hs_
df_copy['Highschool_graduation_rates']=Highschool_graduation_rates


Median_population_age=pca.fit_transform(df_copy[['male_age_mean','male_age_median','male
df_copy['Median_population_age']=Median_population_age

Second_mortgage_statistics=pca.fit_transform(df_copy[['home_equity_second_mortgage','sec
df_copy['Second_mortgage_statistics']=Second_mortgage_statistics

Bad_debt_expense=pca.fit_transform(df_copy[['debt','debt_cdf','bad_debt']])
df_copy['Bad_debt_expense']=Bad_debt_expense
```

In [142...
```
df_copy.head(2)
```

Out[142...

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	zip_code	area_code	lat	
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	13346	315	42.840812	-75.5
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	46616	574	41.701441	-86.2

In [143...
```
df_copy_1=df_copy.drop(['state_ab','UID',
                        'hs_degree','hs_degree_male',
                        'hs_degree_female',
                        'male_age_mean','male_age_median','male_age_stdev',
                        'female_age_mean',
                        'female_age_median','female_age_stdev',
                        'home_equity_second_mortgage','second_mortgage','debt',
                        'debt_cdf',
                        'bad_debt','male_pop','female_pop','pop_bins','bins_bad_debt','bins_c
```

In [144...
```
df_copy_1.head(2)
```

Out[144...

	COUNTYID	STATEID	state	city	place	type	zip_code	area_code	lat	lng	ALan
0	53	36	New York	Hamilton	Hamilton	City	13346	315	42.840812	-75.501524	202183361.
1	141	18	Indiana	South Bend	Roseland	City	46616	574	41.701441	-86.266614	1560828.

In [145...
```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()

df_copy_1['state']=lb.fit_transform(df_copy_1['state'])
df_copy_1['city']=lb.fit_transform(df_copy_1['city'])
df_copy_1['place']=lb.fit_transform(df_copy_1['place'])
df_copy_1['type']=lb.fit_transform(df_copy_1['type'])
```

In [146...
```
df_copy_1.head()
```

Out[146...

	COUNTYID	STATEID	state	city	place	type	zip_code	area_code	lat	lng	ALand	AWat
0	53	36	32	2933	4296	2	13346	315	42.840812	-75.501524	202183361.0	169912
1	141	18	14	6773	9032	2	46616	574	41.701441	-86.266614	1560828.0	10036
2	63	18	14	1700	2457	2	46122	317	39.792202	-86.515246	69561595.0	28419
3	127	72	39	6378	4227	4	927	787	18.396103	-66.104169	1105793.0	
4	161	20	16	4235	6241	2	66502	785	39.195573	-96.569366	2554403.0	

In [147...
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [148...
```
df_copy_2=scaler.fit_transform(df_copy_1)
df_copy_3= pd.DataFrame(df_copy_2, columns = df_copy_1.columns)
df_copy_3.head()
```

Out[148...

	COUNTYID	STATEID	state	city	place	type	zip_code	area_code	lat	lng	
0	-0.332895	0.467449	0.505052	-0.444883	-0.488829	-0.240498	-1.246069	-1.206771	0.954424	0.970719	0
1	0.561421	-0.627508	-0.685565	1.194102	0.909241	-0.240498	-0.120031	-0.092088	0.750493	0.309416	-0
2	-0.231268	-0.627508	-0.685565	-0.971151	-1.031703	-0.240498	-0.136750	-1.198164	0.408767	0.294143	-0
3	0.419144	2.657362	0.968069	1.025508	-0.509198	1.404447	-1.666396	0.824621	-3.420820	1.548002	-0
4	0.764675	-0.505846	-0.553275	0.110835	0.085336	-0.240498	0.553020	0.816013	0.301979	-0.323485	-0

In [149...
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error
```

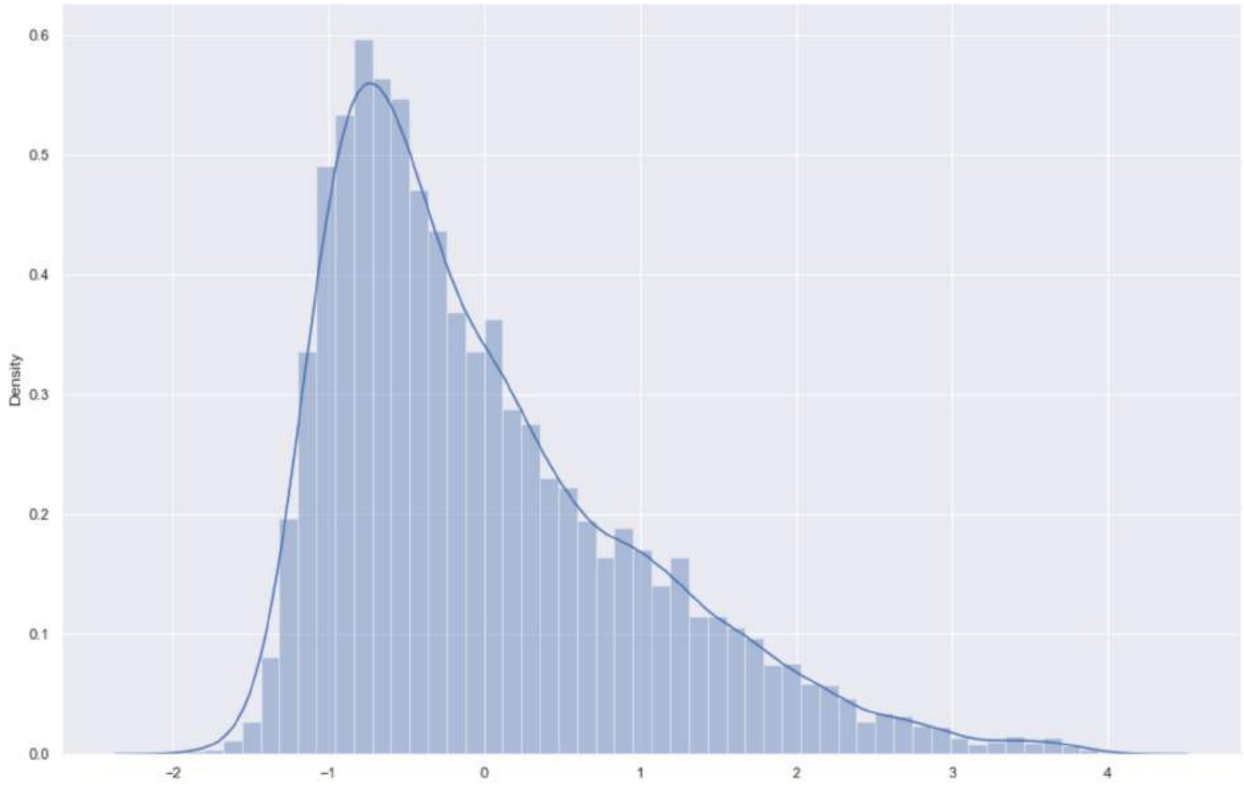
In [150...
```
X=df_copy_3.drop(['hc_mortgage_mean'],axis=1)
y=df_copy_3['hc_mortgage_mean']
```

In [151...
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=4
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
ypredict=lr.predict(X_test)

print('r-squared value: ',r2_score(y_pred=ypredict, y_true=y_test))
print('mean suared error: ',mean_squared_error(y_pred=ypredict, y_true=y_test))
```

```
r-squared value:  0.9886917340693451
mean suared error:  0.011234966670360955
```

In [152...
```
sns.distplot(ypredict)
plt.show()
```



Observations:

- doing PAC on the set of latente variables (Highschool graduation rates,Median population age,Second mortgage statistics,Percent own,Bad debt expense),applying standard scaling and label encoding:state,city, place, type, we see that the R squared increased to almost 99%

In [ ]:

Data Reporting:

1. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

a) Box plot of distribution of average rent by type of place (village, urban, town, etc.).

b) Pie charts to show overall debt and bad debt.

c) 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.

d) Heat map for correlation matrix.

e) Pie chart to show the population distribution across different types of places (village, urban, town etc.)

https://public.tableau.com/shared/6XC5SKDNW?:display_count=n&:origin=viz_share_link

In [ ]: