Real Estate Project

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.

Project Task: Week 1

A) Data Import and Preparation:

1.Import data.

In [1]:
```
# Import reuired libraries
import pandas as pd
import numpy as np

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

In [2]:
```
# Import reuired datasets
df_train=pd.read_csv("train.csv")

df_test=pd.read_csv("test.csv")

print("train shape : ", df_train.shape)
print("test shape  : ", df_test.shape)
```

```
train shape :  (27321, 80)
test shape  :  (11709, 80)
```

In [3]:
```
#  check top 2 train dataset
df_train.head(2)
```

Out[3]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...

2 rows × 80 columns

In [4]:
```
#  check top test dataset
```

```
df_test.head(2)
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...

2 rows × 80 columns

In [5]:

```
# description of the data in the DataFrame
df_train.describe()
```

Out[5]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	27321.000000	27
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	596.507668	
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	232.497482	
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	201.000000	
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	405.000000	
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	614.000000	
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	801.000000	
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	989.000000	

8 rows × 74 columns

In [6]:

```
# description of the data in the DataFrame
df_test.describe()
```

Out[6]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	11709.000000	11
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	593.598514	
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	232.074263	
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	201.000000	
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	404.000000	
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	612.000000	
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	787.000000	
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	989.000000	

8 rows × 74 columns

In [7]:
```
# summary of the dataframe
df_train.info()
```

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

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

In [8]:

```
# summary of the dataframe
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11709 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   UID                          11709 non-null  int64
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     11709 non-null  int64
 3   COUNTYID                     11709 non-null  int64
 4   STATEID                      11709 non-null  int64
 5   state                        11709 non-null  object
 6   state_ab                     11709 non-null  object
 7   city                         11709 non-null  object
 8   place                        11709 non-null  object
 9   type                         11709 non-null  object
 10  primary                      11709 non-null  object
 11  zip_code                     11709 non-null  int64
 12  area_code                    11709 non-null  int64
 13  lat                          11709 non-null  float64
 14  lng                          11709 non-null  float64
```

```
15  ALand                          11709 non-null  int64
16  AWater                         11709 non-null  int64
17  pop                            11709 non-null  int64
18  male_pop                       11709 non-null  int64
19  female_pop                     11709 non-null  int64
20  rent_mean                      11561 non-null  float64
21  rent_median                    11561 non-null  float64
22  rent_stdev                     11561 non-null  float64
23  rent_sample_weight             11561 non-null  float64
24  rent_samples                   11561 non-null  float64
25  rent_gt_10                     11560 non-null  float64
26  rent_gt_15                     11560 non-null  float64
27  rent_gt_20                     11560 non-null  float64
28  rent_gt_25                     11560 non-null  float64
29  rent_gt_30                     11560 non-null  float64
30  rent_gt_35                     11560 non-null  float64
31  rent_gt_40                     11560 non-null  float64
32  rent_gt_50                     11560 non-null  float64
33  universe_samples               11709 non-null  int64
34  used_samples                   11709 non-null  int64
35  hi_mean                        11587 non-null  float64
36  hi_median                      11587 non-null  float64
37  hi_stdev                       11587 non-null  float64
38  hi_sample_weight               11587 non-null  float64
39  hi_samples                     11587 non-null  float64
40  family_mean                    11573 non-null  float64
41  family_median                  11573 non-null  float64
42  family_stdev                   11573 non-null  float64
43  family_sample_weight           11573 non-null  float64
44  family_samples                 11573 non-null  float64
45  hc_mortgage_mean               11441 non-null  float64
46  hc_mortgage_median             11441 non-null  float64
47  hc_mortgage_stdev              11441 non-null  float64
48  hc_mortgage_sample_weight      11441 non-null  float64
49  hc_mortgage_samples            11441 non-null  float64
50  hc_mean                        11419 non-null  float64
51  hc_median                      11419 non-null  float64
52  hc_stdev                       11419 non-null  float64
53  hc_samples                     11419 non-null  float64
54  hc_sample_weight               11419 non-null  float64
55  home_equity_second_mortgage    11489 non-null  float64
56  second_mortgage                11489 non-null  float64
57  home_equity                    11489 non-null  float64
58  debt                           11489 non-null  float64
59  second_mortgage_cdf            11489 non-null  float64
60  home_equity_cdf                11489 non-null  float64
61  debt_cdf                       11489 non-null  float64
62  hs_degree                      11624 non-null  float64
63  hs_degree_male                 11620 non-null  float64
64  hs_degree_female               11604 non-null  float64
65  male_age_mean                  11625 non-null  float64
66  male_age_median                11625 non-null  float64
67  male_age_stdev                 11625 non-null  float64
68  male_age_sample_weight         11625 non-null  float64
69  male_age_samples               11625 non-null  float64
70  female_age_mean                11613 non-null  float64
71  female_age_median              11613 non-null  float64
72  female_age_stdev               11613 non-null  float64
73  female_age_sample_weight       11613 non-null  float64
74  female_age_samples             11613 non-null  float64
```

```
 75  pct_own                    11587 non-null  float64
 76  married                    11625 non-null  float64
 77  married_snp                11625 non-null  float64
 78  separated                  11625 non-null  float64
 79  divorced                   11625 non-null  float64
dtypes: float64(61), int64(13), object(6)
memory usage: 7.1+ MB
```

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

In [9]:
```
# unique values/non-repeating values can be used as primary key and duplicates can be r

print('Duplicates in training dataset :')
print(df_train.duplicated().value_counts(),'\n')

print('Duplicates in testing dataset :')
print(df_test.duplicated().value_counts(),'\n')
```

```
Duplicates in training dataset :
False    27161
True       160
dtype: int64

Duplicates in testing dataset :
False    11677
True        32
dtype: int64
```

In [10]:
```
# Removing the duplicates from the data set
df_train.drop_duplicates(keep='first', inplace = True)
df_test.drop_duplicates(keep='first', inplace = True)
```

In [11]:
```
# Shape
print("train shape : ", df_train.shape)
print("test shape  : ", df_test.shape)
```

```
train shape :  (27161, 80)
test shape  :  (11677, 80)
```

In [12]:
```
# check unique values-non-repeating values can be used as primary key
df_train.nunique()  == df_train.shape[0]
```

Out[12]:
```
UID             True
BLOCKID        False
SUMLEVEL       False
COUNTYID       False
STATEID        False
               ...
pct_own        False
married        False
married_snp    False
separated      False
divorced       False
Length: 80, dtype: bool
```

In [13]:
```

```

```
df_test.nunique() == df_test.shape[0]
```

Out[13]:
```
UID             True
BLOCKID        False
SUMLEVEL       False
COUNTYID       False
STATEID        False
               ...
pct_own        False
married        False
married_snp    False
separated      False
divorced       False
Length: 80, dtype: bool
```

Insights from above there is unique values in UID variable hence UID column is consider as Primary key in dataset.

3.Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.

In [14]:
```
df_train.isnull().sum()
```

Out[14]:
```
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 [15]:
```
df_train.nunique()
```

Out[15]:
```
UID          27161
BLOCKID          0
SUMLEVEL         1
COUNTYID       296
STATEID         52
               ...
pct_own      22302
married      20282
married_snp  10350
separated     6190
divorced     13688
Length: 80, dtype: int64
```

In [16]:
```
df_test.nunique()
```

Out[16]:
```
UID          11677
BLOCKID          0
SUMLEVEL         1
COUNTYID       246
```

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

In [17]:
```
# Block ID column has all missing values, and SUMLEVEL and primary each have single val
# Hence, will remove these 3 features from both the data sets.

df_test.drop(columns=['BLOCKID', 'SUMLEVEL','primary'], axis = 1, inplace=True)
df_train.drop(columns=['BLOCKID', 'SUMLEVEL','primary'], axis = 1, inplace=True)

df_test.shape[1], df_test.shape[1]
```

Out[17]:
```
(77, 77)
```

In [18]:
```
# missing value treatment

def Check_Null_Col(data):
    Null_Variables=[]
    for i in data.columns:
         if data[i].isna().any()==True:
            Null_Variables.append(i)
    return Null_Variables
```

In [19]:
```
from sklearn.impute import SimpleImputer
imp=SimpleImputer(strategy='mean')
```

In [20]:
```
Null_val_train=Check_Null_Col(df_train)
Null_val_test=Check_Null_Col(df_test)
```

In [21]:
```
def Mean_Imputer(data, nullset):
    for i in nullset:
        data[i]=imp.fit_transform(data[[i]])
```

In [22]:
```
Mean_Imputer(df_train,Null_val_train)
Mean_Imputer(df_test,Null_val_test)
```

In [23]:
```
# Shape
print("train shape : ", df_train.shape)
print("test shape  : ", df_test.shape)
```

```
train shape :  (27161, 77)
test shape  :  (11677, 77)
```

In [24]:
```
df_train.to_csv('clean_train.csv')
df_test.to_csv('clean_test.csv')
```

In [25]:
```
df_train.isna().sum().sum()
```

Out[25]:
```
0
```

In [26]:
```
df_test.isna().sum().sum()
```

Out[26]:
```
0
```

B) Exploratory Data Analysis (EDA):

4.Perform debt analysis. You may take the following steps:

Step1:- Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map. You may keep the upper limit for the percent of households with a second mortgage to 50 percent.

In [27]:
```
top_2500_loc = df_train[(df_train['second_mortgage']>0.065) &
                        (df_train["home_equity"]>.10)].sort_values (by = 'second_mortga
```

In [28]:
```
top_2500_loc=top_2500_loc[['state','city','state_ab','place','lat','lng']]
top_2500_loc.head()
```

Out[28]:

	state	city	state_ab	place	lat	lng
17289	Louisiana	Fort Polk	LA	Fort Polk North	31.099608	-93.202088
7413	Virginia	Alexandria	VA	Bailey's Crossroads	38.828707	-77.121251
1211	Kentucky	Fort Knox	KY	Fort Knox	37.879246	-85.961748
16834	Kentucky	Fort Knox	KY	Fort Knox	37.901474	-85.986474
20807	California	Los Angeles	CA	West Hollywood City	34.067232	-118.292902

In [29]:
```
conda install geopandas
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.

Retrieving notices: ...working... done

Note: you may need to restart the kernel to use updated packages.
```

In [30]:
```
import geopandas as gpd
gdf = gpd.GeoDataFrame(top_2500_loc, geometry=gpd.points_from_xy(x=top_2500_loc.lng, y=
gdf
```

Out[30]:

	state	city	state_ab	place	lat	lng	geometry

	state	city	state_ab	place	lat	lng	geometry
17289	Louisiana	Fort Polk	LA	Fort Polk North	31.099608	-93.202088	POINT (-93.20209 31.09961)
7413	Virginia	Alexandria	VA	Bailey's Crossroads	38.828707	-77.121251	POINT (-77.12125 38.82871)
1211	Kentucky	Fort Knox	KY	Fort Knox	37.879246	-85.961748	POINT (-85.96175 37.87925)
16834	Kentucky	Fort Knox	KY	Fort Knox	37.901474	-85.986474	POINT (-85.98647 37.90147)
20807	California	Los Angeles	CA	West Hollywood City	34.067232	-118.292902	POINT (-118.29290 34.06723)
...	...	...	...	...	...	...	...
19068	Kansas	Augusta	KS	Augusta City	37.613920	-96.926250	POINT (-96.92625 37.61392)
27001	Michigan	Davisburg	MI	Village Of Clarkston City	42.728586	-83.539028	POINT (-83.53903 42.72859)
13509	Colorado	Thornton	CO	Northglenn City	39.948761	-104.974382	POINT (-104.97438 39.94876)
8157	Washington	Bremerton	WA	Navy Yard City	47.564968	-122.657581	POINT (-122.65758 47.56497)
3001	California	Carlsbad	CA	Carlsbad City	33.099173	-117.246899	POINT (-117.24690 33.09917)

2500 rows × 7 columns

Step2:- Use the following bad debt equation:

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

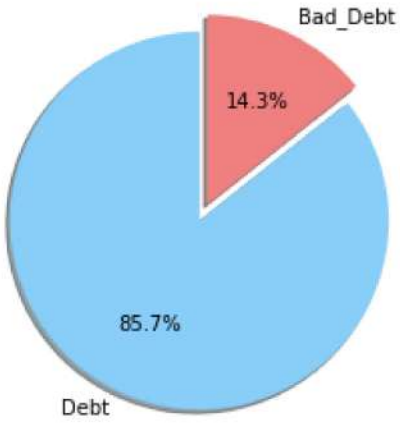
In [31]:
```
import numpy as np
bad_debt_train= (df_train['second_mortgage']+ df_train['home_equity'])-df_train['home_e
bad_debt_test= (df_test['second_mortgage']+ df_test['home_equity'])-df_test['home_equit
```

In [32]:
```
df_train['bad_debt']=bad_debt_train
df_test['bad_debt']=bad_debt_test
```

In [33]:
```
df_train['good_debt']=df_train['debt']-df_train['bad_debt']
df_test['good_debt']=df_test['debt']-df_test['bad_debt']
```

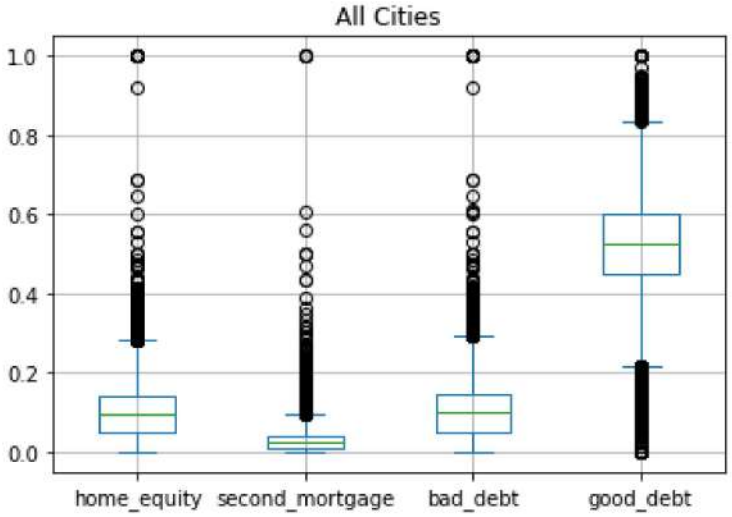
Step3:- Create pie charts to show overall debt and bad debt

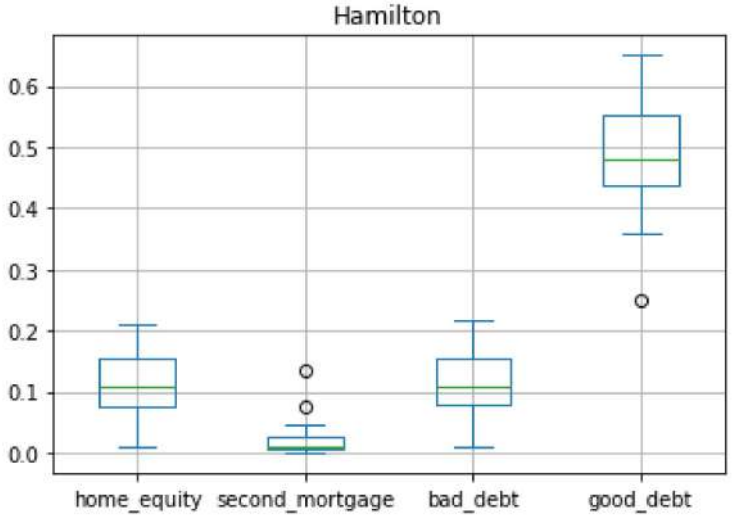
In [34]:
```
import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline

labels = 'Debt', 'Bad_Debt'
sizes = [df_train['debt'].mean()*100, df_train['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=90)

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

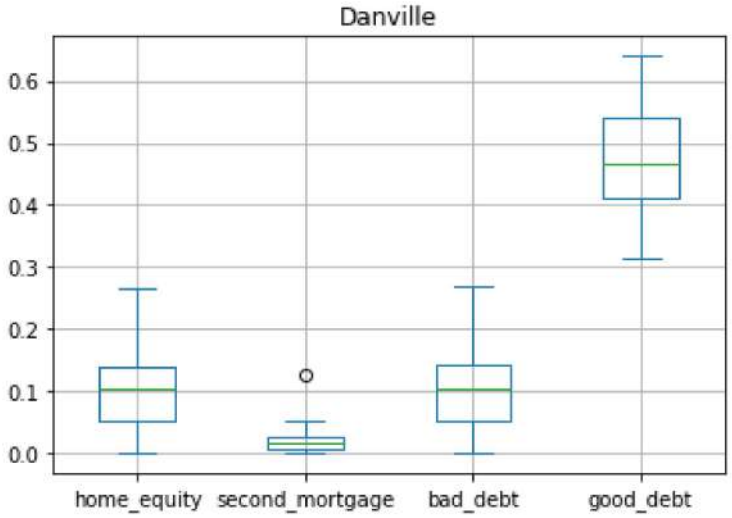


Step4:-Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities

In [35]:
```
import matplotlib.pyplot as plt

all_cities=df_train[['home_equity','second_mortgage','bad_debt', 'good_debt']]
all_cities.plot.box(grid=True)
plt.title('All Cities')
plt.show()

hamilton=df_train[df_train['city']=='Hamilton']
hamilton=hamilton[['home_equity','second_mortgage','bad_debt', 'good_debt']]
hamilton.plot.box(grid=True)
plt.title('Hamilton')
plt.show()

danville=df_train[df_train['city']=='Danville']
danville=danville[['home_equity','second_mortgage','bad_debt', 'good_debt']]
danville.plot.box(grid=True)
plt.title('Danville')
plt.show()
```



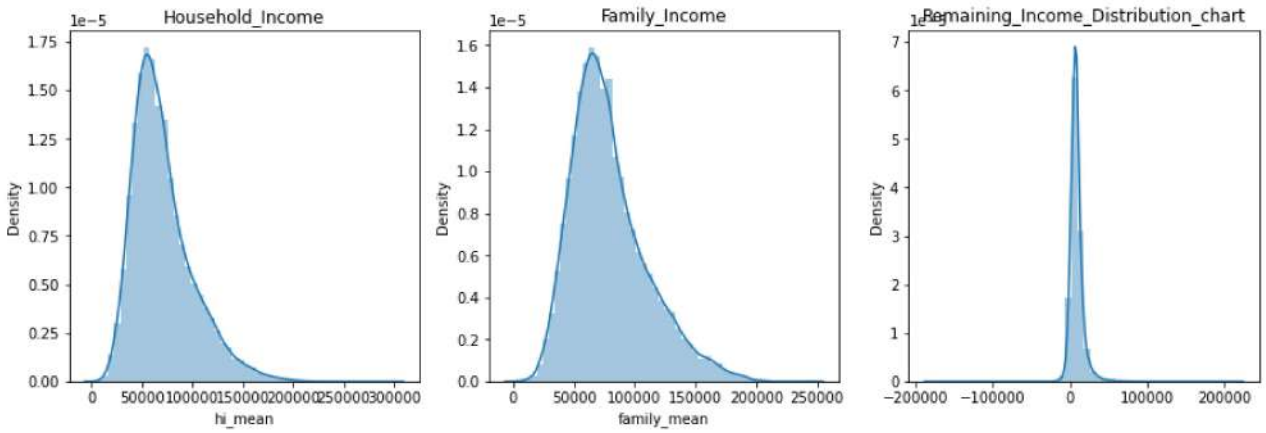




Step5:-Create a collated income distribution chart for family income, house hold income, and remaining income

In [36]:
```
import seaborn as sns

import seaborn as sns
plt.figure(figsize=(15,10))
```

```
plt.subplot(2,3,1)
sns.distplot(df_train['hi_mean'])
plt.title('Household_Income')

plt.subplot(2,3,2)
sns.distplot(df_train['family_mean'])
plt.title('Family_Income')

plt.subplot(2,3,3)
sns.distplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining_Income_Distribution_chart')
plt.show()
```

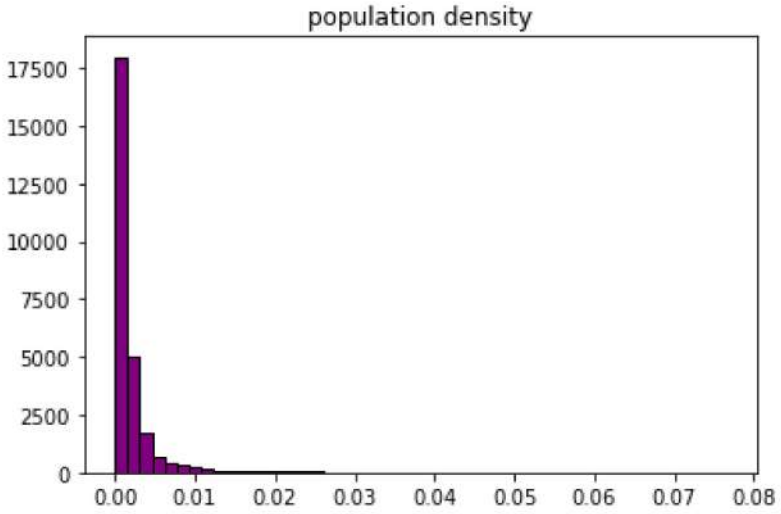


5.Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):

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

In [37]:
```
Pop_Density_train = df_train['pop'] / df_train['ALand']
Pop_Density_test  = df_test['pop']  /  df_test['ALand']
df_train['pop_density'] = Pop_Density_train
df_test['pop_density'] = Pop_Density_test
```

In [38]:
```
plt.hist(df_train['pop_density'], bins=50, color='purple', edgecolor='black')
plt.title('population density')
plt.show()
```



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

In [39]:
```
df_train["age_median"] = (df_train["male_age_median"]) + df_train["female_age_median"]
df_test["age_median"]  = (df_test["male_age_median"])  + df_test ["female_age_median"]
```

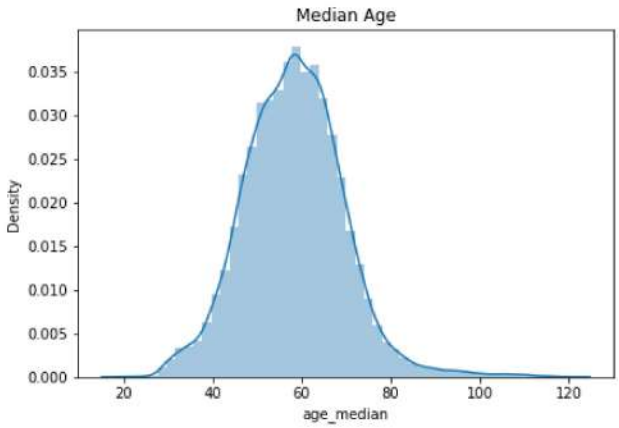
In [40]:
```
df_train[["male_age_median", "female_age_median", "male_pop", "female_pop", "age_median
```

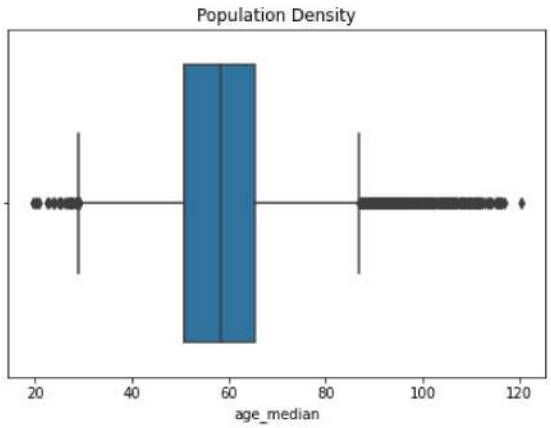
Out[40]:

	male_age_median	female_age_median	male_pop	female_pop	age_median
0	44.00000	45.33333	2612	2618	66.666665
1	32.00000	37.58333	1349	1284	50.791665
2	40.83333	42.83333	3643	3238	62.249995
3	48.91667	50.58333	1141	1559	74.208335
4	22.41667	21.58333	2586	3051	33.208335

c. Visualize the findings using appropriate chart type

In [41]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.subplot(2,2,2)
sns.boxplot(df_train['age_median'])
plt.title('Population Density')
plt.show()
```





6.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 [42]:
```
df_train['pop_bins'] = pd.cut(df_train['pop'], bins=5, labels=['very low','low','medium
df_train['pop_bins'].value_counts()
```

Out[42]:
```
very low     26901
low            246
medium           9
high             4
very high        1
Name: pop_bins, dtype: int64
```

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

In [43]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

Out[43]:

pop_bins	married	separated	divorced
very low	26901	26901	26901
low	246	246	246
medium	9	9	9
high	4	4	4
very high	1	1	1

In [44]:
```
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "media
```

Out[44]:

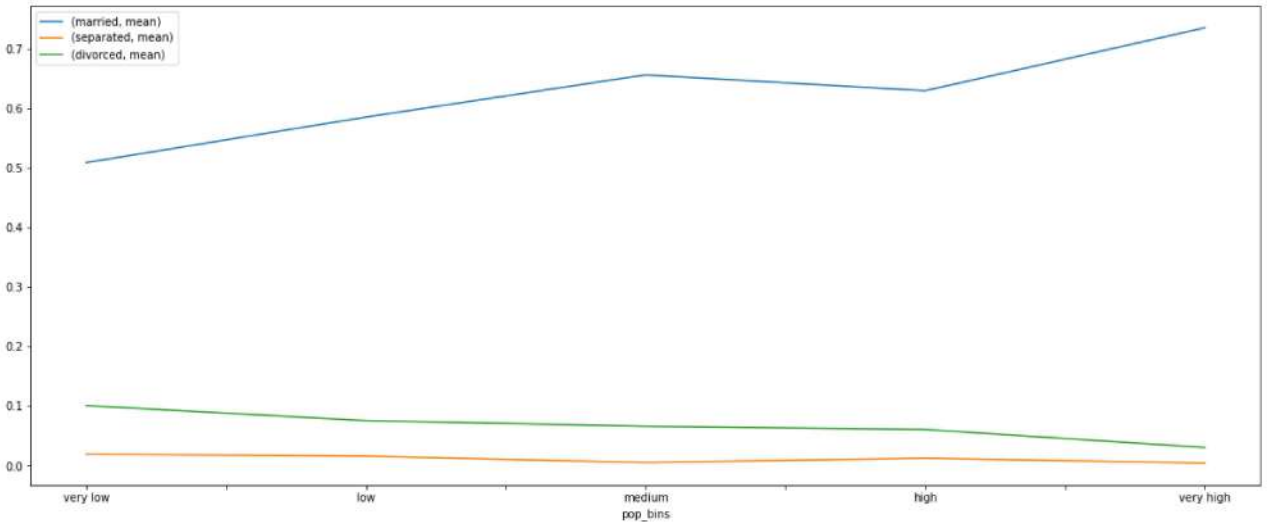
	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.508546	0.525390	0.019109	0.013640	0.100634	0.096080
low	0.584894	0.593135	0.015833	0.011195	0.075348	0.070045
medium	0.655737	0.618710	0.005003	0.004120	0.065927	0.064890

	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
high	0.629132	0.675095	0.012372	0.007340	0.060562	0.059870
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.030360

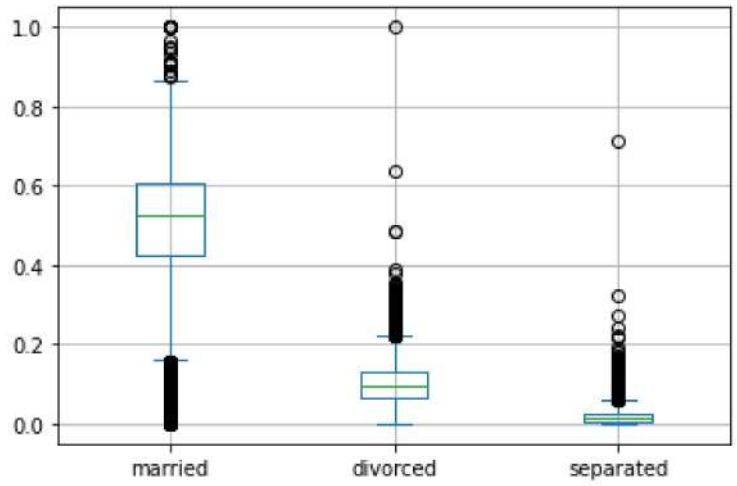
b) Visualize using appropriate chart type

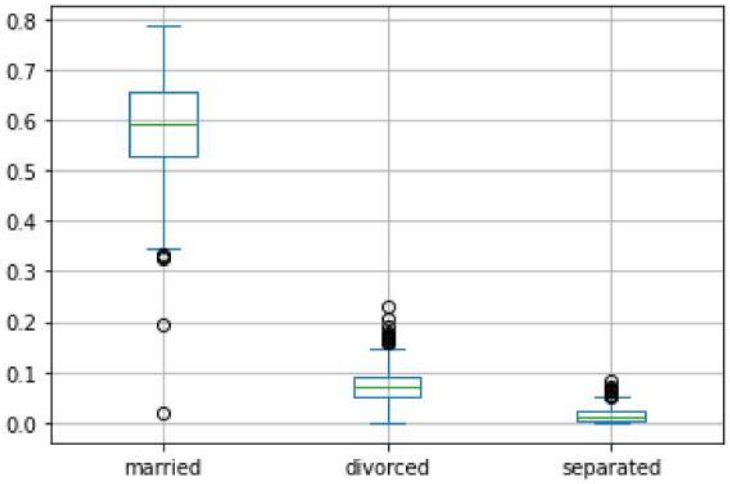
In [45]:
```
plt.figure(figsize=(10,5))
pop_bin_married=df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg
pop_bin_married.plot(figsize=(20,8))
plt.legend(loc='best')
plt.show()
```

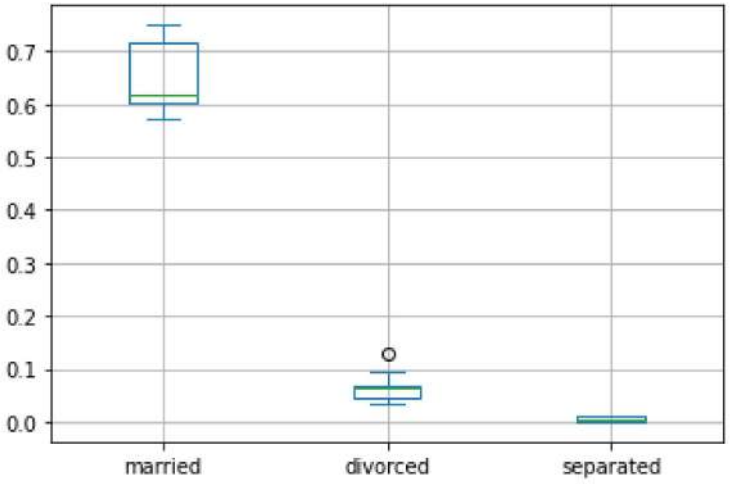
```
<Figure size 720x360 with 0 Axes>
```

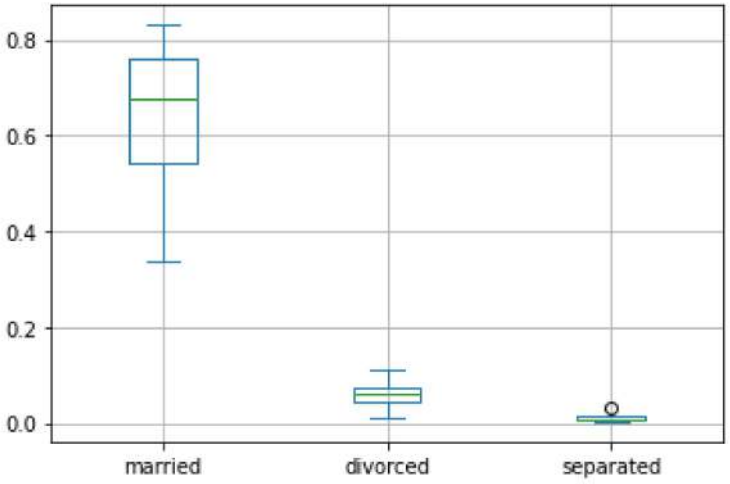


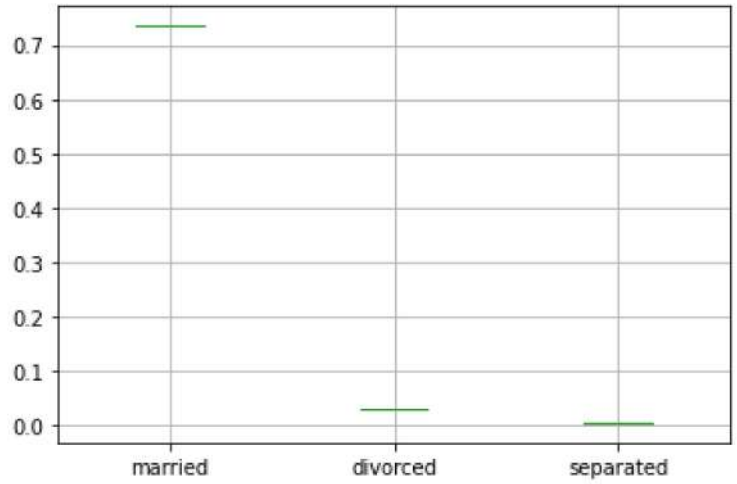
In [46]:
```
df_train.groupby(by='pop_bins')[['married','divorced', 'separated']].plot.box(grid='Tru
plt.show()
```




0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
married
divorced
separated


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
married
divorced
separated


0.8
0.6
0.4
0.2
0.0
married
divorced
separated



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

In [47]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

Out[47]:

state	mean
Alabama	774.006498
Alaska	1185.763570
Arizona	1098.689988
Arkansas	718.425272
California	1473.493903

In [48]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

Out[48]:

state	mean
Alabama	67030.487478
Alaska	92136.545109
Arizona	73390.637993
Arkansas	64409.153057
California	87831.121491

In [49]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

Out[49]:
```
state
```

```
Alabama                 0.011547
Alaska                  0.012870
Arizona                 0.014970
Arkansas                0.011154
California              0.016776
Colorado                0.013554
Connecticut             0.012612
Delaware                0.012929
District of Columbia    0.013196
Florida                 0.015789
Name: mean, dtype: float64
```

In [50]:
```
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

Out[50]:
```
0.013351076774049805
```

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

In [51]:
```
plt.figure(figsize=(17,7))
sns.heatmap(data=df_train[['hc_mortgage_mean','ALand','pop','rent_mean','hi_mean','hc_m
                           'hs_degree','debt','home_equity']].corr(),annot=True)
plt.show()
```

Project Task: Week 2

C) 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 [52]:
```
pip install factor_analyzer
```

```
Requirement already satisfied: factor_analyzer in c:\users\sonal\anaconda3\lib\site-pack
ages (0.4.1)
Requirement already satisfied: scikit-learn in c:\users\sonal\anaconda3\lib\site-package
s (from factor_analyzer) (1.1.1)
Requirement already satisfied: scipy in c:\users\sonal\anaconda3\lib\site-packages (from
factor_analyzer) (1.7.1)
Requirement already satisfied: pre-commit in c:\users\sonal\anaconda3\lib\site-packages
(from factor_analyzer) (2.20.0)
Requirement already satisfied: pandas in c:\users\sonal\anaconda3\lib\site-packages (fro
m factor_analyzer) (1.3.4)
Requirement already satisfied: numpy in c:\users\sonal\anaconda3\lib\site-packages (from
factor_analyzer) (1.20.3)
Requirement already satisfied: pytz>=2017.3 in c:\users\sonal\anaconda3\lib\site-package
s (from pandas->factor_analyzer) (2021.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\sonal\anaconda3\lib\si
te-packages (from pandas->factor_analyzer) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\sonal\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas->factor_analyzer) (1.16.0)
Requirement already satisfied: nodeenv>=0.11.1 in c:\users\sonal\anaconda3\lib\site-pack
ages (from pre-commit->factor_analyzer) (1.7.0)
Requirement already satisfied: cfgv>=2.0.0 in c:\users\sonal\anaconda3\lib\site-packages
(from pre-commit->factor_analyzer) (3.3.1)
Requirement already satisfied: virtualenv>=20.0.8 in c:\users\sonal\anaconda3\lib\site-p
ackages (from pre-commit->factor_analyzer) (20.16.5)
Requirement already satisfied: pyyaml>=5.1 in c:\users\sonal\anaconda3\lib\site-packages
(from pre-commit->factor_analyzer) (6.0)
Requirement already satisfied: toml in c:\users\sonal\anaconda3\lib\site-packages (from
pre-commit->factor_analyzer) (0.10.2)
Requirement already satisfied: identify>=1.0.0 in c:\users\sonal\anaconda3\lib\site-pack
ages (from pre-commit->factor_analyzer) (2.5.6)
Requirement already satisfied: setuptools in c:\users\sonal\anaconda3\lib\site-packages
(from nodeenv>=0.11.1->pre-commit->factor_analyzer) (58.0.4)
Requirement already satisfied: platformdirs<3,>=2.4 in c:\users\sonal\anaconda3\lib\site
-packages (from virtualenv>=20.0.8->pre-commit->factor_analyzer) (2.5.2)
Requirement already satisfied: distlib<1,>=0.3.5 in c:\users\sonal\anaconda3\lib\site-pa
ckages (from virtualenv>=20.0.8->pre-commit->factor_analyzer) (0.3.6)
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\sonal\anaconda3\lib\site-p
ackages (from virtualenv>=20.0.8->pre-commit->factor_analyzer) (3.8.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\sonal\anaconda3\lib\site
-packages (from scikit-learn->factor_analyzer) (2.2.0)
Requirement already satisfied: joblib>=1.0.0 in c:\users\sonal\anaconda3\lib\site-packag
es (from scikit-learn->factor_analyzer) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [53]:
```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [54]:
```
df_train.isna().sum().sum()
```

Out[54]: 0

In [55]:
```
df_train.describe().T
```

Out[55]:

	count	mean	std	min	25%	50%	
UID	27161.0	257328.592209	21342.667653	220342.00000	238826.000000	257212.000000	275810.
COUNTYID	27161.0	85.660322	98.373195	1.00000	29.000000	63.000000	109.
STATEID	27161.0	28.267185	16.385918	1.00000	13.000000	28.000000	42.
zip_code	27161.0	50105.133243	29546.218041	602.00000	26847.000000	47804.000000	77093.
area_code	27161.0	596.589816	232.424069	201.00000	405.000000	614.000000	801.
...	...	...	...	...	...	...	
divorced	27161.0	0.100385	0.048673	0.00000	0.066210	0.095660	0.
bad_debt	27161.0	0.105161	0.070420	0.00000	0.052840	0.100190	0.
good_debt	27161.0	0.524461	0.129606	0.00000	0.449290	0.525880	0.
pop_density	27161.0	0.002044	0.004509	0.00000	0.000121	0.000851	0.
age_median	27161.0	58.312437	11.350315	19.58333	50.750005	58.249995	65.

76 rows × 8 columns

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

D) Data Modeling :

3.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:

- Variables should have significant impact on predicting Monthly mortgage and owner costs
- Variables should have significant impact on predicting Monthly mortgage and owner costs
- R square of 60 percent and above should be achieved

- Ensure Multi-collinearity does not exist in dependent variables
- Test if predicted variable is normally distributed

In [56]:
```
df_train.columns
```

Out[56]:
```
Index(['UID', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place',
       'type', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater', 'pop',
       'male_pop', 'female_pop', 'rent_mean', 'rent_median', 'rent_stdev',
       'rent_sample_weight', 'rent_samples', 'rent_gt_10', 'rent_gt_15',
       'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35', 'rent_gt_40',
       'rent_gt_50', 'universe_samples', 'used_samples', 'hi_mean',
       'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'bad_debt', 'good_debt', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

In [57]:
```
df_train['type'].unique()
```

Out[57]:
```
array(['City', 'Urban', 'Town', 'CDP', 'Village', 'Borough'], dtype=object)
```

In [58]:
```
type_dict={'type':{'City':1, 'Urban':2, 'Town':3, 'CDP':4, 'Village':5, 'Borough':6}}
df_train.replace(type_dict,inplace=True)
```

In [59]:
```
df_test.replace(type_dict,inplace=True)
```

In [60]:
```
df_train['type'].unique()
```

Out[60]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [61]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean','second_mortg
              'age_median','pct_own', 'married','separated', 'divorced']
```

In [62]:
```
xtrain=df_train[feature_cols]
ytrain=df_train['hc_mortgage_mean']
```

In [63]:
```
xtest=df_test[feature_cols]
ytest=df_test['hc_mortgage_mean']
```

In [64]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error,accuracy_s
```

In [65]:
```
xtest.head()
```

Out[65]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt
0	163	26	48239	4	3417	53802.87122	0.06443	0.07651	0.63624
1	1	23	4210	1	3796	85642.22095	0.01175	0.14375	0.64755
2	15	42	14871	6	3944	65694.06582	0.01316	0.06497	0.45395
3	231	21	42633	1	2508	44156.38709	0.00995	0.01741	0.41915
4	355	48	78410	3	6230	123527.02420	0.00000	0.03440	0.63188

In [66]:
```
sc=StandardScaler()
xtrain_scaled=sc.fit_transform(xtrain)
xtest_scaled=sc.fit_transform(xtest)
```

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

In [67]:
```
lr=LinearRegression()
lr.fit(xtrain_scaled,ytrain)
```

Out[67]:
```
LinearRegression()
```

In [68]:
```
ypred= lr.predict(xtest_scaled)
```

R square of 60 percent and above should be achieved

In [69]:
```
print("Overall R2 score of linear regression model", r2_score(ytest,ypred))
print("Overall RMSE of linear regression model", np.sqrt(mean_squared_error(ytest,ypred
```

```
Overall R2 score of linear regression model 0.7393650505184304
Overall RMSE of linear regression model 320.4641119170825
```

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

In [70]:
```
state=df_train['STATEID'].unique()
state
```

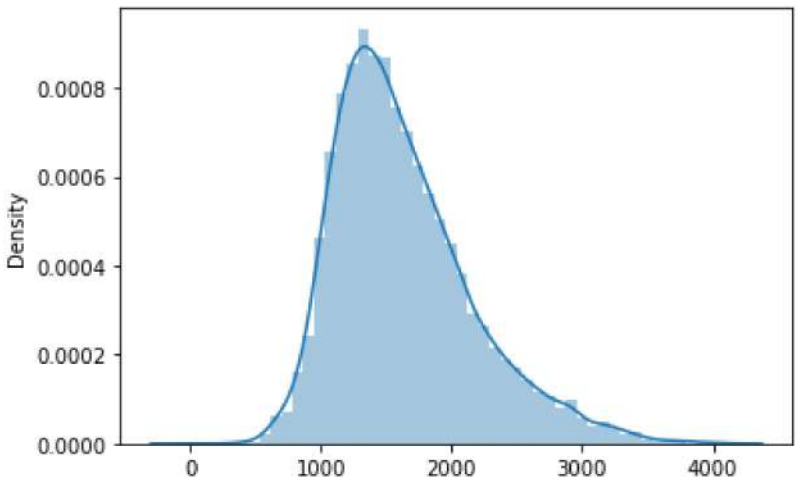
Out[70]:
```
array([36, 18, 72, 20,  1, 48, 45,  6,  5, 24, 17, 19, 47, 32, 22,  8, 44,
       28, 34, 41,  4, 12, 55, 42, 37, 51, 26, 39, 40, 13, 16, 46, 27, 29,
       53, 56,  9, 54, 21, 25, 11, 15, 30,  2, 33, 49, 50, 31, 38, 35, 23,
       10], dtype=int64)
```

In [71]:
```
for i in [11,33,35]:
    print("State ID-",i)

    xtrain_nation=df_train[df_train['COUNTYID']==i][feature_cols]
    ytrain_nation=df_train[df_train['COUNTYID']==i]['hc_mortgage_mean']

    xtest_nation=df_test[df_test['COUNTYID']==i][feature_cols]
    ytest_nation=df_test[df_test['COUNTYID']==i]['hc_mortgage_mean']

    xtrain_scaled_nation=sc.fit_transform(xtrain_nation)
    xtest_scaled_nation=sc.fit_transform(xtest_nation)

    lr.fit(xtrain_scaled_nation,ytrain_nation)
    ypred_nation=lr.predict(xtest_scaled_nation)

    print("Overall R2 score of linear regression model for state,",i,":-" ,r2_score(yte
    print("Overall RMSE of linear regression model for state,",i,":-" ,np.sqrt(mean_squ
    print("\n")
```

```
State ID- 11
Overall R2 score of linear regression model for state, 11 :- 0.7466965040298006
Overall RMSE of linear regression model for state, 11 :- 238.00669942760925


State ID- 33
Overall R2 score of linear regression model for state, 33 :- 0.8615110128116417
Overall RMSE of linear regression model for state, 33 :- 211.17226657126835


State ID- 35
Overall R2 score of linear regression model for state, 35 :- 0.719056112896151
Overall RMSE of linear regression model for state, 35 :- 258.27254147313795
```

Test if predicted variable is normally distributed

In [72]:
```
sns.distplot(ypred)
plt.show()
```



D) Data Reporting :

4.Create a dashboard in tableau by choosing appropriate chart types and metrics useful for

the business. The dashboard must entail the following:

- Box plot of distribution of average rent by type of place (village, urban, town, etc.).
- Pie charts to show overall debt and bad debt.
- 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.
- Heat map for correlation matrix.
- Pie chart to show the population distribution across different types of places (village, urban, town etc.).

Dashboard Link

https://public.tableau.com/app/profile/sonal.ranpise/viz/Capstone-RealEstateProject/RealEstateDashboard?publish=yes