Real Estate

Problem Statement

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

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline

import seaborn as sns
sns.set(style="white", color_codes=True)
sns.set(font_scale=1.5)

pd.options.display.max_columns = 999
pd.options.display.max_rows = 999
```

Project Task: Week 1

Data Import and Preparation:

Data Import and Preparation

1. Import data.
2. Figure out the primary key and look for the requirement of indexing.
3. Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.

Exploratory Data Analysis (EDA)

4. Perform debt analysis. You may take the following steps:
    1. 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
    2. Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage
    3. Create pie charts to show overall debt and bad debt
    4. Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities
    5. Create a collated income distribution chart for family income, house hold income, and remaining income

Importing Dataset

Dataset Description

Following are the themes the fields fall under Home Owner Costs: Sum of utilities, property taxes.

1. Second Mortgage: Households with a second mortgage statistics.
2. Home Equity Loan: Households with a Home equity Loan statistics.
3. Debt: Households with any type of debt statistics.
4. Mortgage Costs: Statistics regarding mortgage payments, home equity loans, utilities and property taxes
5. Home Owner Costs: Sum of utilities, property taxes statistics
6. Gross Rent: Contract rent plus the estimated average monthly cost of utility features

7. Gross Rent as Percent of Income Gross rent as the percent of income very interesting
8. High school Graduation: High school graduation statistics.
9. Population Demographics: Population demographic statistics.
10. Age Demographics: Age demographic statistics.
11. Household Income: Total income of people residing in the household.
12. Family Income: Total income of people related to the householder.

```
train_DF=pd.read_csv("train.csv")
test_DF=pd.read_csv("test.csv")
```

Properties Of Dataset

```
train_DF.shape
```

```
(27321, 80)
```

```
train_DF.columns
```

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

```
train_DF.info()
#float64(68), int64(6), object(6)
```

```
 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
```

```
#Null Count
```

```
null_count = pd.DataFrame(data = train_DF.isna().sum()).reset_index()
null_count.columns = ['Columns', 'Count']
null_count
```

	Columns	Count
0	UID	0
1	BLOCKID	27321
2	SUMLEVEL	0
3	COUNTYID	0
4	STATEID	0
5	state	0
6	state_ab	0
7	city	0
8	place	0
9	type	0
10	primary	0
11	zip_code	0
12	area_code	0
13	lat	0
14	lng	0
15	ALand	0
16	AWater	0
17	pop	0
18	male_pop	0
19	female_pop	0
20	rent_mean	314
21	rent_median	314
22	rent_stdev	314
23	rent_sample_weight	314
24	rent_samples	314
25	rent_gt_10	314
26	rent_gt_15	314
27	rent_gt_20	314
28	rent_gt_25	314
29	rent_gt_30	314

30	rent_gt_35	314
31	rent_gt_40	314
32	rent_gt_50	314
33	universe_samples	0
34	used_samples	0
35	hi_mean	268
36	hi_median	268
37	hi_stdev	268
38	hi_sample_weight	268
39	hi_samples	268
40	family_mean	298
41	family_median	298
42	family_stdev	298
43	family_sample_weight	298
44	family_samples	298
45	hc_mortgage_mean	573
46	hc_mortgage_median	573
47	hc_mortgage_stdev	573
48	hc_mortgage_sample_weight	573
49	hc_mortgage_samples	573
50	hc_mean	600

```
train_DF.describe()
```

```
                    UID  BLOCKID  SUMLEVEL      COUNTYID       STATEID      zip_code     are
train_DF.head()
```

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

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

[ ] ↳ *2 cells hidden*

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.

```
#percantage of missing values in train set
missing_list_train=train_DF.isnull().sum() *100/len(train_DF)
missing_values_train_DF=pd.DataFrame(missing_list_train,columns=['Percantage of missing value
missing_values_train_DF.sort_values(by=['Percantage of missing values'],inplace=True,ascendin
missing_values_train_DF[missing_values_train_DF['Percantage of missing values'] >0][:10]
```

	Percantage of missing values
BLOCKID	100.000000
hc_samples	2.196113

```
#percantage of missing values in test set
missing_list_test=test_DF.isnull().sum() *100/len(train_DF)
missing_values_test_DF=pd.DataFrame(missing_list_test,columns=['Percantage of missing values'
missing_values_test_DF.sort_values(by=['Percantage of missing values'],inplace=True,ascending
missing_values_test_DF[missing_values_test_DF['Percantage of missing values'] >0][:10]
```

	Percantage of missing values
BLOCKID	42.857143
hc_samples	1.061455
hc_mean	1.061455
hc_median	1.061455
hc_stdev	1.061455
hc_sample_weight	1.061455
hc_mortgage_mean	0.980930
hc_mortgage_stdev	0.980930
hc_mortgage_sample_weight	0.980930
hc_mortgage_samples	0.980930

```
train_Variance = pd.DataFrame(train_DF.var(),columns=['Variance'])
train_Variance_sort = train_Variance.sort_values(by=['Variance'],ascending=True)
train_Variance_sort
```

	Variance
SUMLEVEL	0.000000e+00
separated	4.324823e-04
home_equity_second_mortgage	9.816144e-04
second_mortgage	1.165163e-03
married_snp	1.416734e-03
divorced	2.406412e-03
rent_gt_10	3.992479e-03
home_equity	4.803112e-03
rent_gt_15	1.202414e-02
hs_degree_female	1.260518e-02
hs_degree	1.263819e-02
hs_degree_male	1.457972e-02
married	1.873070e-02
rent_gt_50	1.897297e-02
rent_gt_20	2.067820e-02
rent_gt_40	2.347543e-02
debt	2.441949e-02
rent_gt_35	2.566433e-02
rent_gt_25	2.569779e-02
rent_gt_30	2.689811e-02
pct_own	5.136550e-02
home_equity_cdf	6.559987e-02
debt_cdf	6.976878e-02
second_mortgage_cdf	8.699930e-02
female_age_stdev	6.452908e+00
male_age_stdev	6.454528e+00
lat	3.122874e+01
male_age_mean	3.138880e+01
female_age_mean	3.464873e+01
male_age_median	6.201013e+01

female_age_median	6.463493e+01
lng	2.671203e+02
STATEID	2.687254e+02
hc_stdev	8.364293e+03
COUNTYID	9.669398e+03
rent_stdev	3.504021e+04
hc_sample_weight	3.606685e+04
hc_mortgage_sample_weight	3.815782e+04
hc_mean	4.899137e+04
hc_median	5.354243e+04
area_code	5.405508e+04
hc_mortgage_stdev	5.667666e+04
hc_samples	6.286450e+04
rent_sample_weight	7.409473e+04
female_age_sample_weight	8.039884e+04
family_sample_weight	8.445016e+04
male_age_sample_weight	9.792059e+04
rent_mean	1.913455e+05
rent_median	1.969565e+05
used_samples	2.030608e+05
hi_sample_weight	2.052613e+05
rent_samples	2.130261e+05
hc_mortgage_samples	2.156778e+05
universe_samples	2.171653e+05
family_samples	3.145786e+05
hc_mortgage_mean	3.883859e+05
hc_mortgage_median	4.259121e+05
hi_samples	5.641452e+05
female_age_samples	1.186612e+06
female_pop	1.214173e+06
male_age_samples	1.220127e+06
male_pop	1.243111e+06

pop	4.705542e+06

```
test_Variance = pd.DataFrame(test_DF.var(),columns=['Variance'])
test_Variance_sort = test_Variance.sort_values(by=['Variance'],ascending=True)
test_Variance_sort
```

	Variance
SUMLEVEL	0.000000e+00
separated	4.591488e-04
home_equity_second_mortgage	9.310234e-04
second_mortgage	1.131915e-03
married_snp	1.497178e-03
divorced	2.354662e-03
rent_gt_10	4.045380e-03
home_equity	4.957866e-03
rent_gt_15	1.161853e-02
hs_degree_female	1.281528e-02
hs_degree	1.309286e-02
hs_degree_male	1.503193e-02
rent_gt_50	1.894976e-02
married	1.953678e-02
rent_gt_20	2.031020e-02
rent_gt_40	2.371054e-02
debt	2.484846e-02
rent_gt_35	2.602161e-02
rent_gt_25	2.610039e-02
rent_gt_30	2.747616e-02
pct_own	5.393173e-02
home_equity_cdf	6.612493e-02
debt_cdf	7.020468e-02
second_mortgage_cdf	8.815262e-02
female_age_stdev	6.527548e+00
male_age_stdev	6.669518e+00
male_age_mean	3.113336e+01
lat	3.165080e+01
female_age_mean	3.423645e+01
male_age_median	6.077617e+01

female_age_median	6.355320e+01
lng	2.692165e+02
STATEID	2.758012e+02
hc_stdev	8.669225e+03
COUNTYID	9.861351e+03
rent_stdev	3.579432e+04
hc_sample_weight	3.620181e+04
hc_mortgage_sample_weight	3.887804e+04
hc_mean	5.121523e+04
area_code	5.385846e+04
hc_median	5.641313e+04
hc_mortgage_stdev	5.799220e+04
hc_samples	6.232246e+04
female_age_sample_weight	7.895680e+04
rent_sample_weight	7.913797e+04
family_sample_weight	8.353823e+04

```
train_DF.drop(columns=['BLOCKID','SUMLEVEL'],inplace=True) #SUMLEVEL doest not have any predi
```

```
test_DF.drop(columns=['BLOCKID','SUMLEVEL'],inplace=True) #SUMLEVEL doest not have any predic
```

```
# Imputing  missing values with mean
missing_train_cols=[]
for col in train_DF.columns:
    if train_DF[col].isna().sum() !=0:
        missing_train_cols.append(col)
print(missing_train_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_g
```

```
# Missing cols are all numerical variables
for col in train_DF.columns:
    if col in (missing_train_cols):
        train_DF[col].replace(np.nan, train_DF[col].mean(),inplace=True)
```

```
# Imputing  missing values with mean
missing_test_cols=[]
for col in test_DF.columns:
    if test_DF[col].isna().sum() !=0:
```

```
        missing_test_cols.append(col)
print(missing_test_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_g
```

```
# Missing cols are all numerical variables
for col in test_DF.columns:
    if col in (missing_test_cols):
        test_DF[col].replace(np.nan, test_DF[col].mean(),inplace=True)
```

```
family_mean             1.022658e+09
```

```
train_DF.isna().sum().sum()
```

```
0
```

```
ALand                   5.813971e+17
```

```
test_DF.isna().sum().sum()
```

```
0
```

Exploratory Data Analysis (EDA):

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

1.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

```
!pip install pandasql
from pandasql import sqldf
q1 = "select place,pct_own,second_mortgage,lat,lng from train_DF where pct_own >0.10 and seco
pysqldf = lambda q: sqldf(q, globals())#The main function used in pandasql is sqldf. sqldf ac
                                       #1. a sql query string = q1
                                       #2. an set of session/environment variables (locals()
train_DF_location_mort_pct=pysqldf(q1)
```

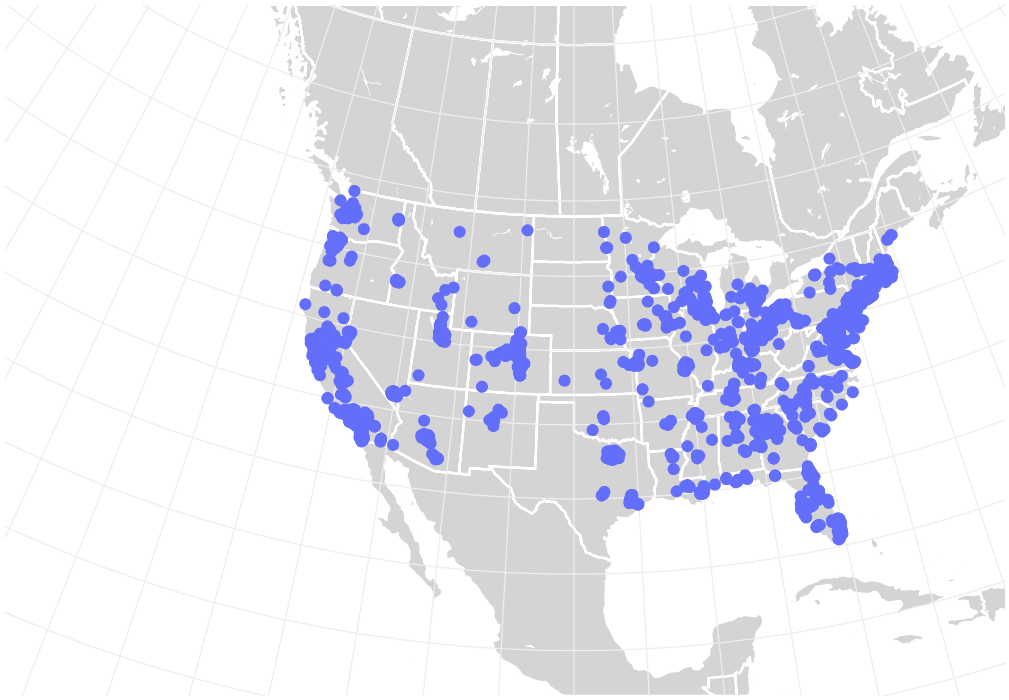
```
Requirement already satisfied: pandasql in /usr/local/lib/python3.6/dist-packages (0.7.3
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from pa
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from par
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from
```

```
train_DF_location_mort_pct.head()
```

	place	pct_own	second_mortgage	lat	lng
0	Worcester City	0.20247	0.43363	42.254262	-71.800347
1	Harbor Hills	0.15618	0.31818	40.751809	-73.853582
2	Glen Burnie	0.22380	0.30212	39.127273	-76.635265
3	Egypt Lake-leto	0.11618	0.28972	28.029063	-82.495395
4	Lincolnwood	0.14228	0.28899	41.967289	-87.652434

```
#!pip install plotly
import plotly.express as px
import plotly.graph_objects as go


fig = go.Figure(data=go.Scattergeo(
    lat = train_DF_location_mort_pct['lat'],
    lon = train_DF_location_mort_pct['lng']),
    )
fig.update_layout(
    geo=dict(
        scope = 'north america',
        showland = True,
        landcolor = "rgb(212, 212, 212)",
        subunitcolor = "rgb(255, 255, 255)",
        countrycolor = "rgb(255, 255, 255)",
        showlakes = True,
        lakecolor = "rgb(255, 255, 255)",
        showsubunits = True,
        showcountries = True,
        resolution = 50,
        projection = dict(
            type = 'conic conformal',
            rotation_lon = -100
        ),
        lonaxis = dict(
            showgrid = True,
            gridwidth = 0.5,
            range= [ -140.0, -55.0 ],
            dtick = 5
        ),
        lataxis = dict (
            showgrid = True,
            gridwidth = 0.5,
            range= [ 20.0, 60.0 ],
            dtick = 5
        )
    ),
```

```
    title='Top 2,500 locations with second mortgage is the highest and percent ownership is a
fig.show()
```

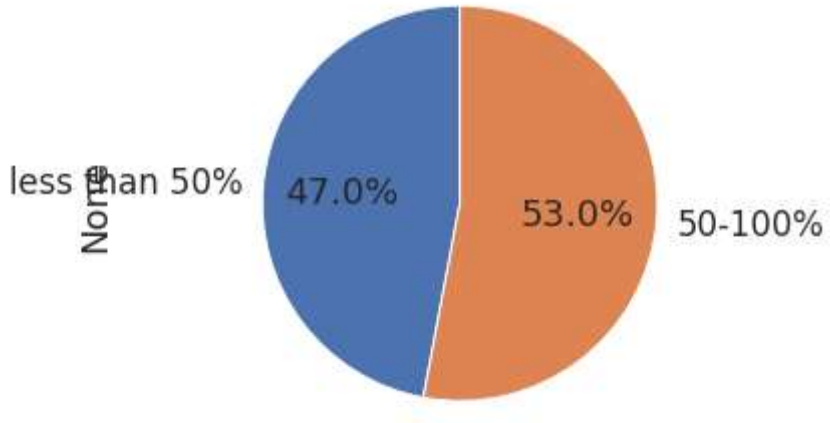


2.Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage

```
train_DF['bad_debt']=train_DF['second_mortgage']+train_DF['home_equity']-train_DF['home_equit
```

3.Create pie charts to show overall debt and bad debt

```
train_DF['bins'] = pd.cut(train_DF['bad_debt'],bins=[0,0.10,1], labels=["less than 50%","50-1
train_DF.groupby(['bins']).size().plot(kind='pie',subplots=True,startangle=90, autopct='%1.1f
plt.axis('equal')

plt.show()
#df.plot.pie(subplots=True,figsize=(8, 3))
```



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

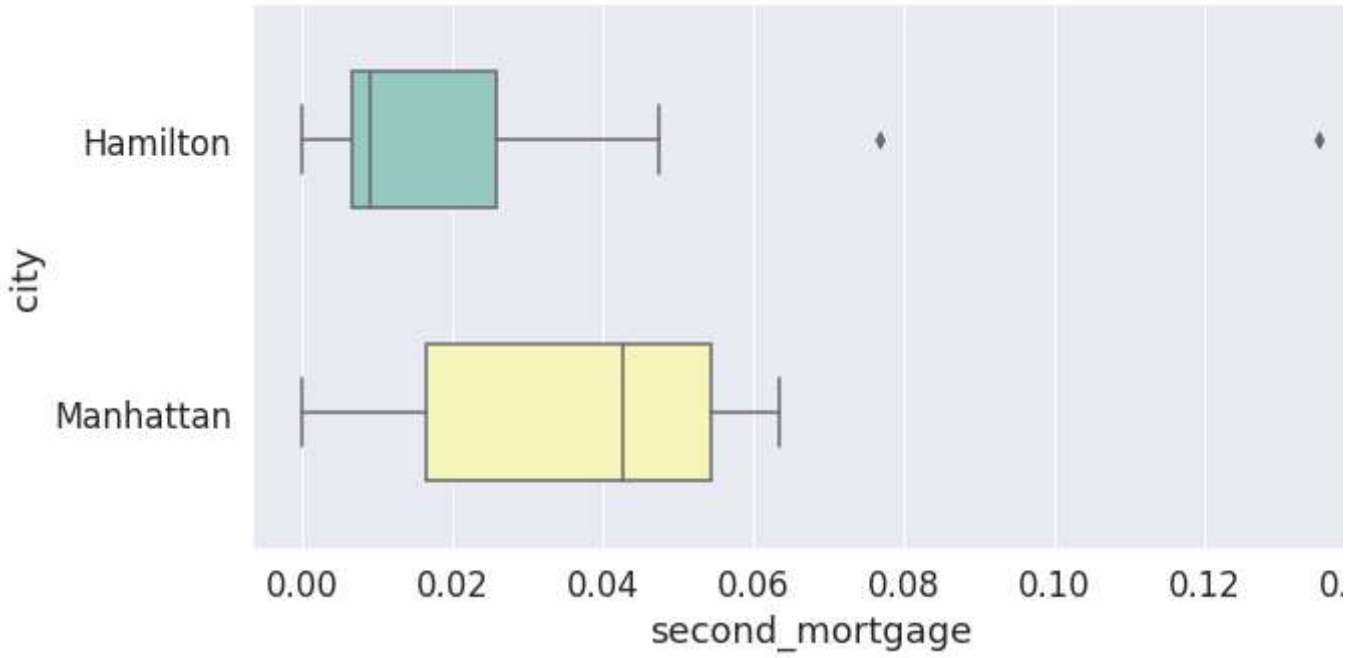
```
cols=[]
train_DF.columns
```

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

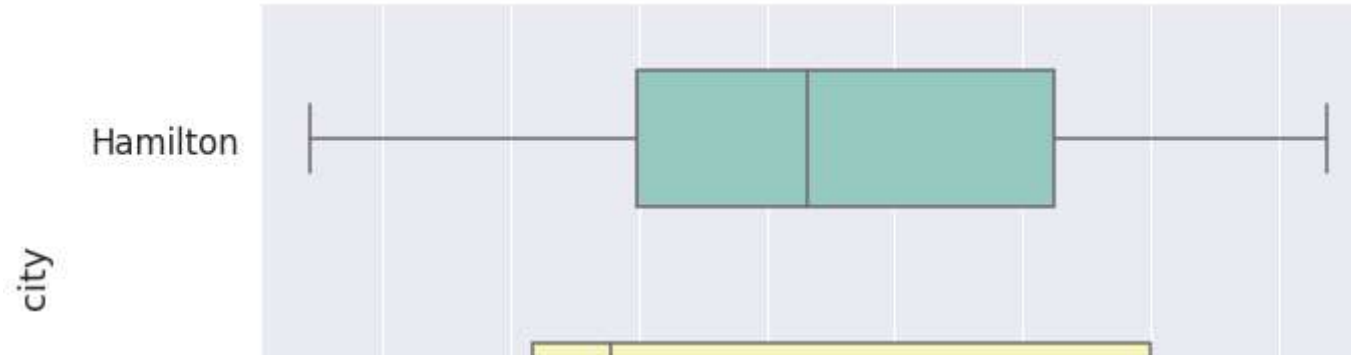
```
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
df_box_hamilton=train_DF.loc[train_DF['city'] == 'Hamilton']
df_box_manhattan=train_DF.loc[train_DF['city'] == 'Manhattan']
df_box_city=pd.concat([df_box_hamilton,df_box_manhattan])
df_box_city.head(4)
```

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_c
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	8
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	45
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	39

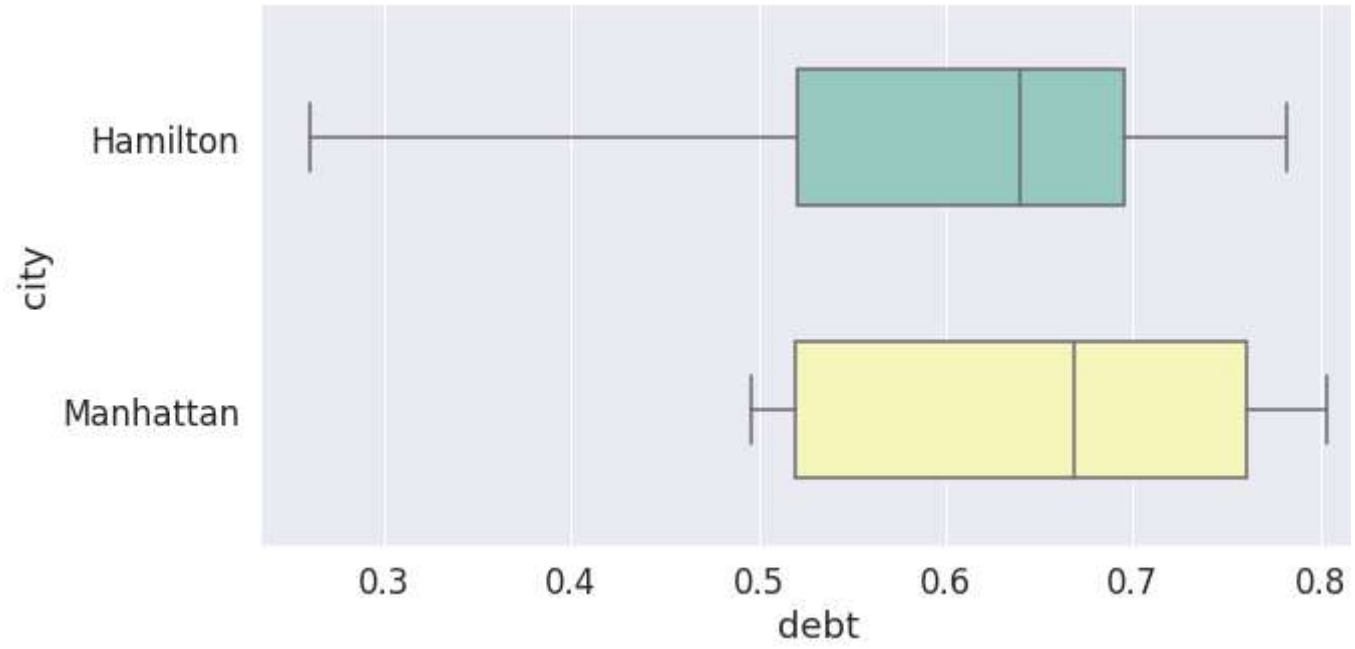
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.5,palette="Set3")
plt.show()
```



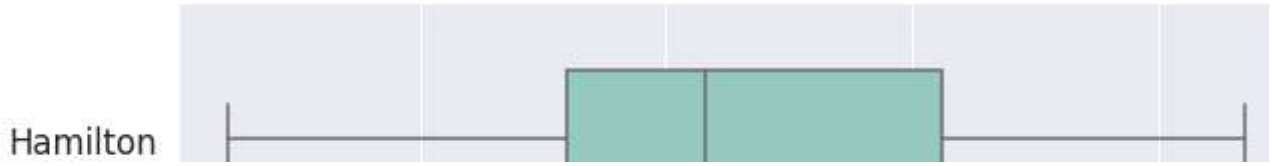

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



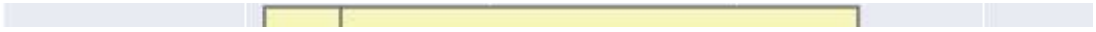

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```




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

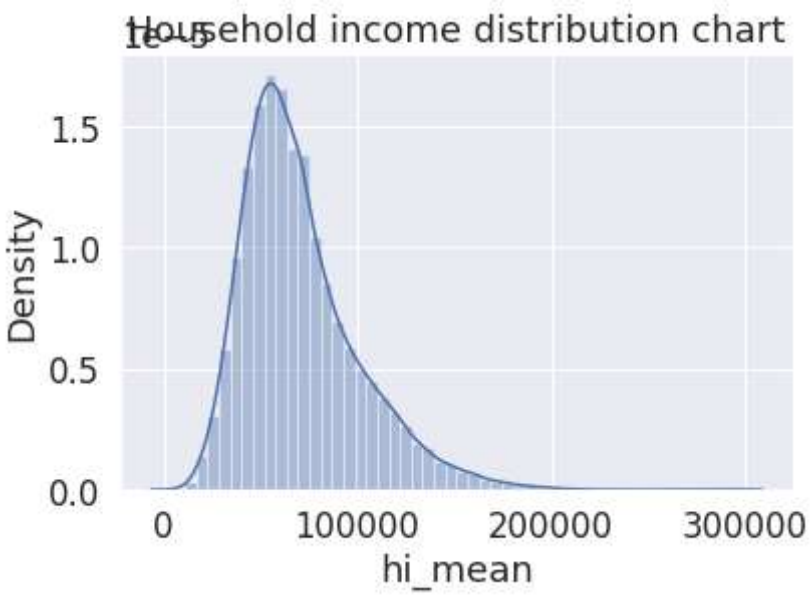


5.Create a collated income distribution chart for family income, house hold income, and remaining income



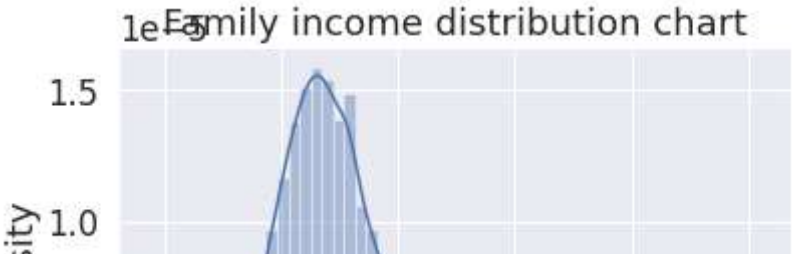
```
sns.distplot(train_DF['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```



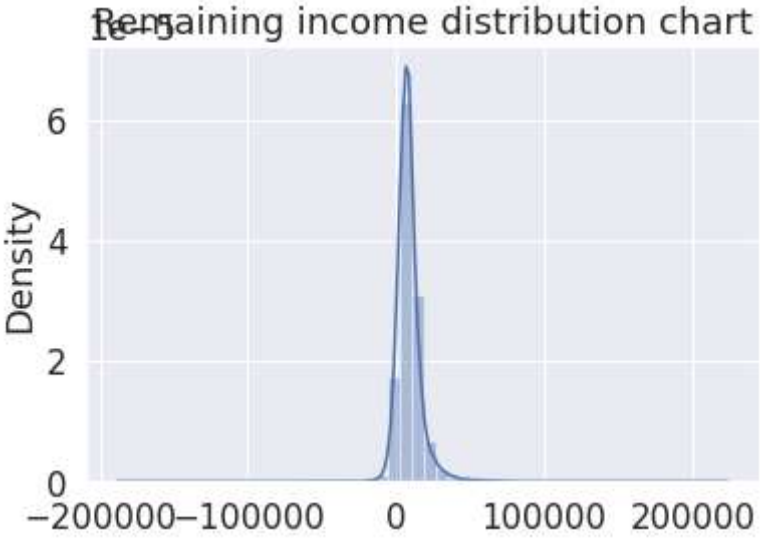

```
sns.distplot(train_DF['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```




```
sns.distplot(train_DF['family_mean']-train_DF['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```



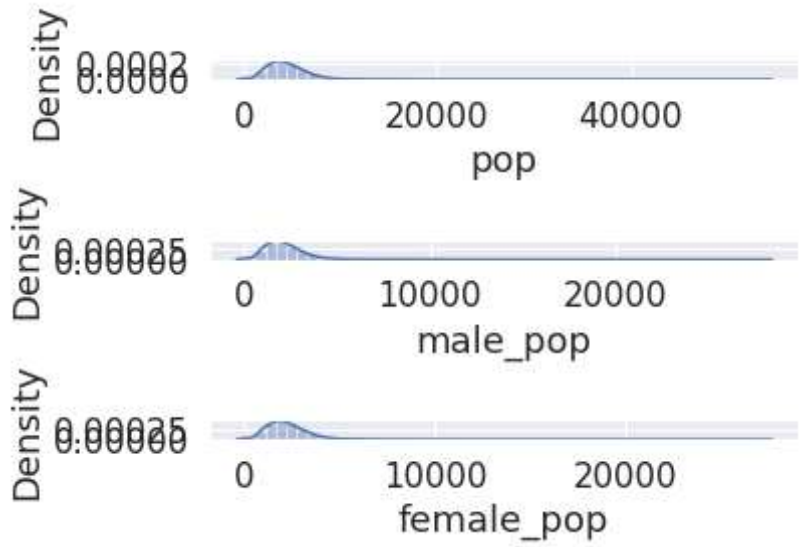
Project Task: Week 2

Exploratory Data Analysis (EDA):

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

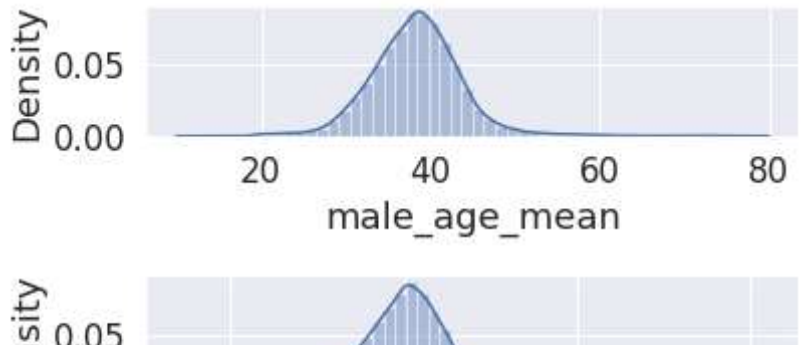
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2,ax3)=plt.subplots(3,1)
sns.distplot(train_DF['pop'],ax=ax1)
sns.distplot(train_DF['male_pop'],ax=ax2)
sns.distplot(train_DF['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap

/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap

/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```




```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2)=plt.subplots(2,1)
sns.distplot(train_DF['male_age_mean'],ax=ax1)
sns.distplot(train_DF['female_age_mean'],ax=ax2)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

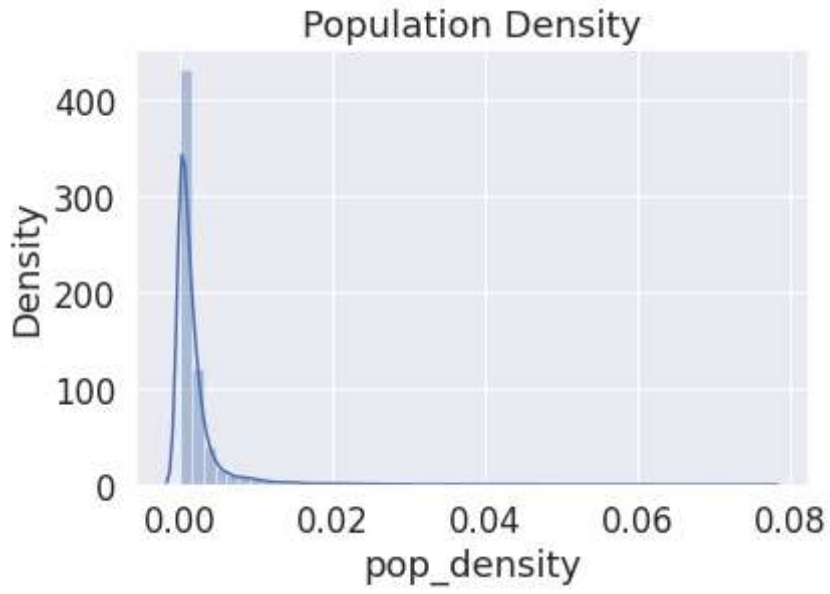
```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap

/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```



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

```
20          40          60          80
train_DF['pop_density']=train_DF['pop']/train_DF['ALand']


test_DF['pop_density']=test_DF['pop']/test_DF['ALand']


sns.distplot(train_DF['pop_density'])
plt.title('Population Density')
plt.show() # Very less density is noticed
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```



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

```
train_DF['age_median']=(train_DF['male_age_median']+train_DF['female_age_median'])/2
test_DF['age_median']=(test_DF['male_age_median']+test_DF['female_age_median'])/2
```

```
train_DF[['male_age_median','female_age_median','male_pop','female_pop','age_median']].head()
```

UID	male_age_median	female_age_median	male_pop	female_pop	age_median
267822	44.00000	45.33333	2612	2618	44.666665
246444	32.00000	37.58333	1349	1284	34.791665
245683	40.83333	42.83333	3643	3238	41.833330
279653	48.91667	50.58333	1141	1559	49.750000
247218	22.41667	21.58333	2586	3051	22.000000

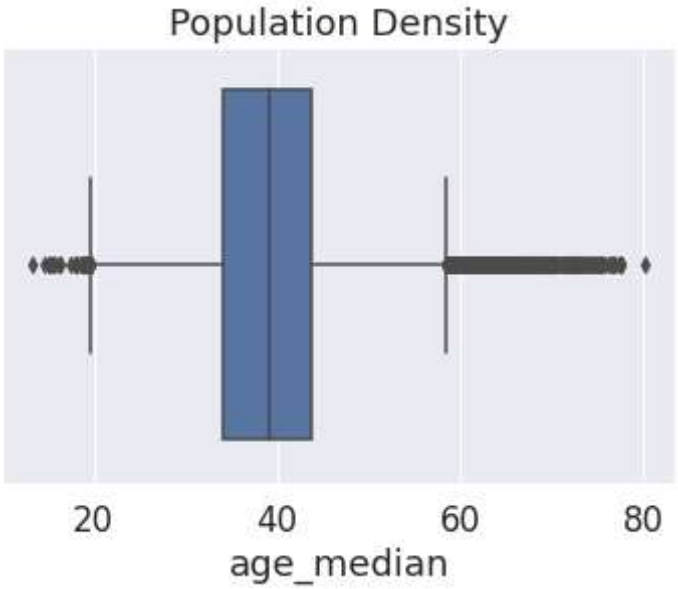
c) Visualize the findings using appropriate chart type

```
sns.distplot(train_DF['age_median'])
plt.title('Median Age')
plt.show()
# Age of population is mostly between 20 and 60
# Majority are of age around 40
# Median age distribution has a gaussian distribution
# Some right skewness is noticed
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
```


```
sns.boxplot(train_DF['age_median'])
plt.title('Population Density')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
```



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.

```
train_DF['pop'].describe()
```

```
count    27321.000000
mean      4316.032685
std       2169.226173
min          0.000000
25%       2885.000000
50%       4042.000000
75%       5430.000000
max      53812.000000
Name: pop, dtype: float64
```

```
train_DF['pop_bins']=pd.cut(train_DF['pop'],bins=5,labels=['very low','low','medium','high','
```

```
train_DF[['pop','pop_bins']]
```

UID	pop	pop_bins
267822	5230	very low
246444	2633	very low
245683	6881	very low
279653	2700	very low
247218	5637	very low
...	...	...
279212	1847	very low
277856	4155	very low
233000	2829	very low
287425	11542	low
265371	3726	very low

27321 rows × 2 columns

```
train_DF['pop_bins'].value_counts()
```

```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

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

```
train_DF.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

```
                married  separated  divorced
```

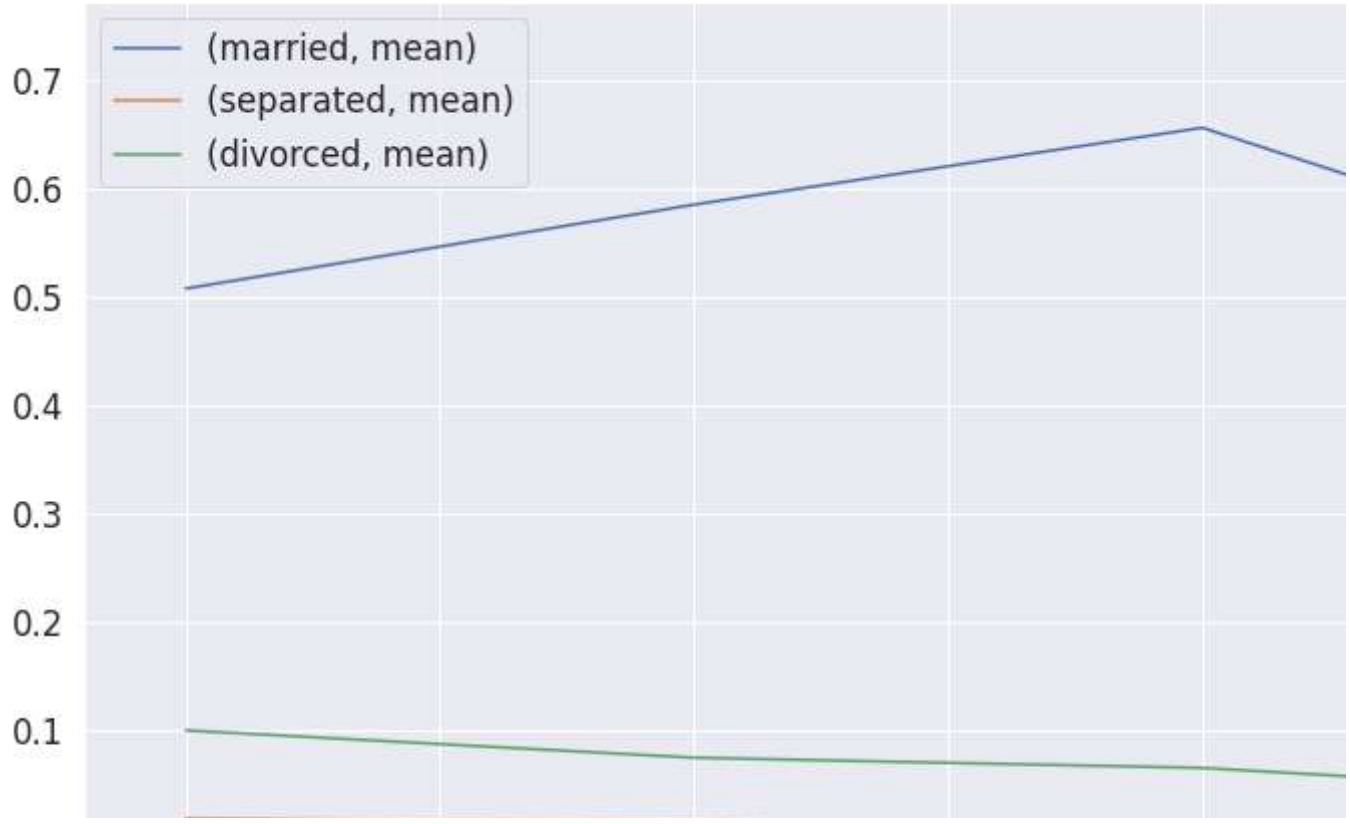
```
train_DF.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"])
```

	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.507548	0.524680	0.019126	0.013650	0.100504	0.096020
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
high	0.503359	0.335660	0.008141	0.002500	0.039030	0.010320
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.030360

b) Visualize using appropriate chart type

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

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



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

```
rent_state_mean=train_DF.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

state	mean
Alabama	774.004927
Alaska	1185.763570
Arizona	1097.753511
Arkansas	720.918575
California	1471.133857

```
income_state_mean=train_DF.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

state	mean
Alabama	67030.064213
Alaska	92136.545109
Arizona	73328.238798
Arkansas	64765.377850
California	87655.470820

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

```
state
Alabama                 0.011547
Alaska                  0.012870
Arizona                 0.014970
Arkansas                0.011131
California              0.016783
Colorado                0.013529
Connecticut             0.012637
Delaware                0.012929
District of Columbia    0.013198
```

```
Florida                 0.015772
Name: mean, dtype: float64
```

```
#overall level rent as a percentage of income
sum(train_DF['rent_mean'])/sum(train_DF['family_mean'])
```

```
0.013358170721473864
```

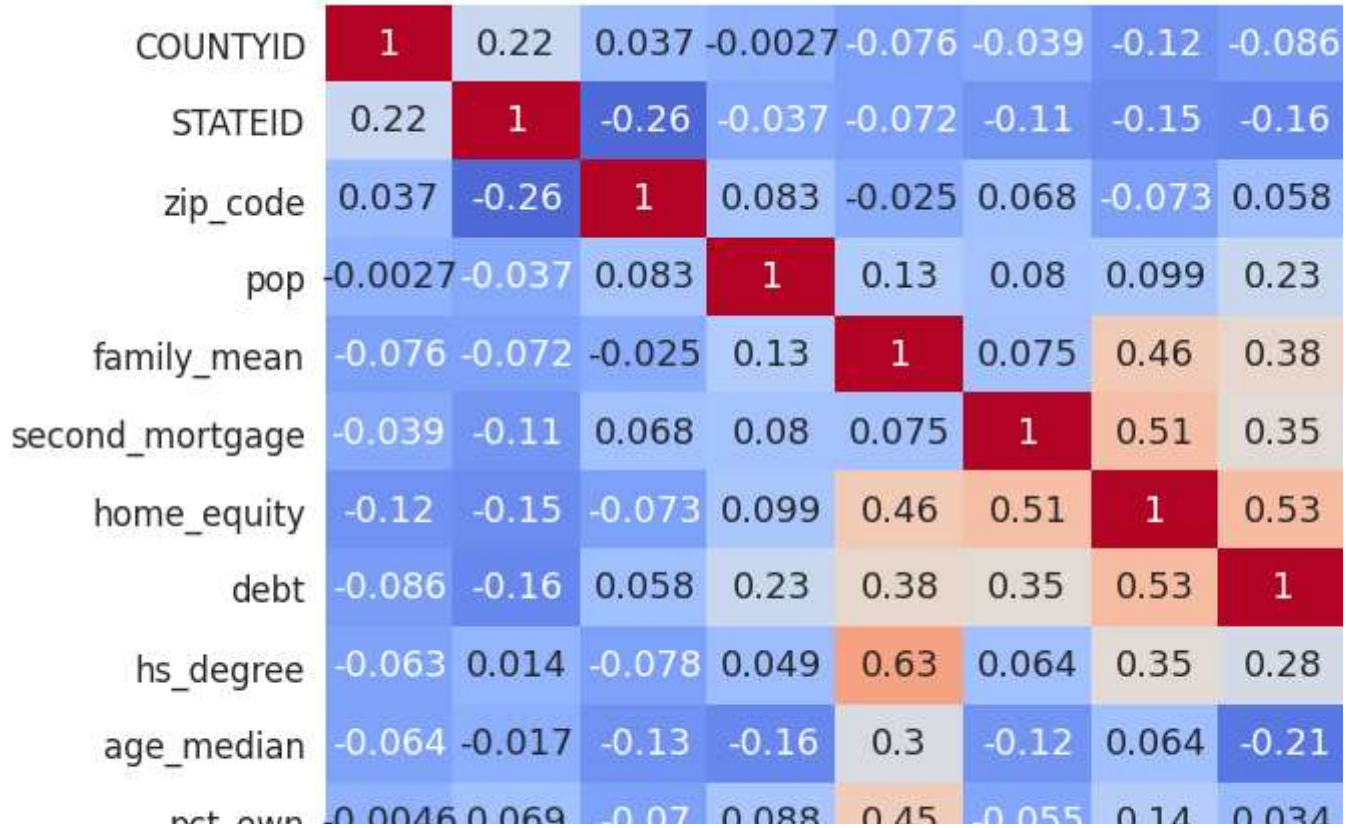
4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

```
train_DF.columns
```

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

```
cor=train_DF[['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
          'age_median','pct_own', 'married','separated', 'divorced']].corr()
```

```
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```



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

```
!pip install factor_analyzer
```

```
Collecting factor_analyzer
  Downloading https://files.pythonhosted.org/packages/44/b5/cbd83484ca6dd4c6562c6d66a6a3
     |████████████████████████████████| 40kB 3.4MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from fa
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from fac
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from fac
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from
Building wheels for collected packages: factor-analyzer
  Building wheel for factor-analyzer (setup.py) ... done
  Created wheel for factor-analyzer: filename=factor_analyzer-0.3.2-cp36-none-any.whl si
  Stored in directory: /root/.cache/pip/wheels/4a/d0/57/f1330cb9c80e82d8d05391c74c94ed61
Successfully built factor-analyzer
Installing collected packages: factor-analyzer
Successfully installed factor-analyzer-0.3.2
```

```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

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

```
            4.24181150e-03,  2.98043464e-01],
          [-1.23469884e-01,  6.07438127e-01,  6.33039232e-01,
           -2.14798787e-02,  2.47973924e-01],
          [-3.42866888e-01,  5.59526275e-01,  5.88213011e-01,
           -2.51533494e-02,  2.18419892e-01],
          [-1.60867194e-01, -1.53062586e-02, -1.57026593e-01,
            1.09243745e-01, -6.61660788e-01],
          [-1.37306737e-01, -2.17250612e-02, -1.58408940e-01,
            1.25156180e-01, -6.71630758e-01],
          [ 2.45096193e-01, -2.54584579e-02, -2.66691522e-02,
            9.53148400e-02, -6.42510821e-01],
          [ 2.03988661e-01,  7.85172846e-02, -3.01656220e-01,
            2.28379465e-02, -6.29223343e-01],
          [ 1.08926071e-01, -6.34332394e-02, -3.36565132e-02,
           -9.49480404e-02,  6.81473818e-01],
          [-2.63787634e-01, -6.43281189e-03, -3.58792082e-02,
           -9.37962365e-02,  6.47816982e-01],
          [-2.15717035e-01, -7.36588940e-02,  3.50113245e-01,
           -1.95201632e-02,  6.36783801e-01],
          [ 3.94306145e-01,  6.09565679e-02,  2.55337867e-01,
           -2.20362097e-01, -1.84248087e-01],
          [ 4.07877889e-01,  6.27256517e-02,  2.23926914e-01,
           -2.10028738e-01, -1.71989233e-01],
          [ 3.53156876e-01,  5.36715654e-02,  2.69603576e-01,
           -2.16933222e-01, -1.80072079e-01],
```

```
 -2.10933222e-01, -1.80072079e-01],
[ 2.33537268e-01, -4.91732979e-02,  8.14450768e-01,
  9.36688754e-02,  3.27131947e-01],
[ 2.40298221e-01, -3.38140106e-02,  8.31496989e-01,
  7.52417576e-02,  2.46323636e-01],
[-6.71839407e-02,  6.58504509e-02,  5.86207671e-01,
  8.72955133e-02,  9.12541511e-02],
[ 5.59835495e-02,  8.17918705e-01, -1.78458347e-01,
 -1.55949400e-02, -3.34299864e-02],
[ 7.16426359e-02,  9.23428542e-01, -1.07142692e-01,
 -2.78635349e-02, -4.35991223e-02],
[ 1.92496958e-01, -4.75870404e-02,  8.03173184e-01,
  1.43492698e-01,  3.33862180e-01],
[ 1.87644443e-01, -3.29941026e-02,  8.58024490e-01,
  1.31329948e-01,  2.55679750e-01],
[-1.02263652e-01,  6.03984287e-02,  4.72982275e-01,
  7.36848428e-02,  1.12273932e-01],
[ 6.14776606e-02,  8.77962754e-01, -1.50410285e-01,
  2.20991065e-02, -4.17158282e-02],
[ 7.83728193e-02,  9.54508790e-01, -5.91095884e-02,
  1.64800946e-02, -4.32591064e-02],
[-3.24381953e-02,  1.11167158e-01,  7.84467397e-01,
 -4.37718484e-02, -2.80931236e-01],
[ 1.76682381e-01,  1.90494237e-01,  5.61405504e-01,
 -1.20746159e-01, -1.32570798e-01],
[-6.37386604e-02, -7.03047894e-02, -2.68934067e-01,
  1.28589786e-01,  1.88507862e-01],
[-1.56051265e-01, -7.08033934e-02, -1.45964508e-01,
  1.24253731e-01,  1.46293131e-01],
[-3.56716286e-01, -5.29910739e-02,  1.47771603e-01,
  2.87196129e-02,  1.13159600e-01],
[ 2.42173838e-01, -2.86199102e-02, -3.25958459e-02,
  1.05027806e-01, -6.55406049e-01],
[ 3.50196756e-01, -1.05016373e-02, -3.95274126e-01,
  5.92876705e-02,  2.91651799e-01],
[ 2.25671558e-01, -3.42672757e-02,  8.92876637e-01,
```

Project Task: Week 4

Data Modeling:

> Keep below considerations while building a linear regression model.

Data Modeling :

1. Variables should have significant impact on predicting Monthly mortgage and owner costs
2. Utilize all predictor variable to start with initial hypothesis
3. R square of 60 percent and above should be achieved

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

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.

```
train_DF.columns
```

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

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

```
train_DF['type'].unique()
```

```
array([1, 2, 3, 4, 5, 6])
```

```
test_DF.replace(type_dict,inplace=True)
```

```
test_DF['type'].unique()
```

```
array([4, 1, 6, 3, 5, 2])
```

```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']
```

```
x_train=train_DF[feature_cols]
y_train=train_DF['hc_mortgage_mean']
```

```
x_test=test_DF[feature_cols]
y_test=test_DF['hc_mortgage_mean']
```

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

```
x_train.head()
```

UID	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equ
267822	53	36	13346	1	5230	67994.14790	0.02077	0.08
246444	141	18	46616	1	2633	50670.10337	0.02222	0.04
245683	63	18	46122	1	6881	95262.51431	0.00000	0.09
279653	127	72	927	2	2700	56401.68133	0.01086	0.01
247218	161	20	66502	1	5637	54053.42396	0.05426	0.05

```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

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

```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```

```
      LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)


y_pred=linereg.predict(x_test_scaled)


print("Overall R2 score of linear regression model", r2_score(y_test,y_pred))
print("Overall RMSE of linear regression model", np.sqrt(mean_squared_error(y_test,y_pred)))

      Overall R2 score of linear regression model 0.7348210754610929
      Overall RMSE of linear regression model 323.1018894984635


#The Accuracy and R2 score are good, but still will investigate the model performance at stat
```

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

```
state=train_DF['STATEID'].unique()
state
#Picking a few iDs 20,1,45,6

      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])


for i in [20,1,45]:
    print("State ID-",i)

    x_train_nation=train_DF[train_DF['COUNTYID']==i][feature_cols]
    y_train_nation=train_DF[train_DF['COUNTYID']==i]['hc_mortgage_mean']

    x_test_nation=test_DF[test_DF['COUNTYID']==i][feature_cols]
    y_test_nation=test_DF[test_DF['COUNTYID']==i]['hc_mortgage_mean']

    x_train_scaled_nation=sc.fit_transform(x_train_nation)
    x_test_scaled_nation=sc.fit_transform(x_test_nation)

    linereg.fit(x_train_scaled_nation,y_train_nation)
    y_pred_nation=linereg.predict(x_test_scaled_nation)

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

     State ID- 20
     Overall R2 score of linear regression model for state, 20 :- 0.6046603766461813
     Overall RMSE of linear regression model for state, 20 :- 307.97188999314704
```

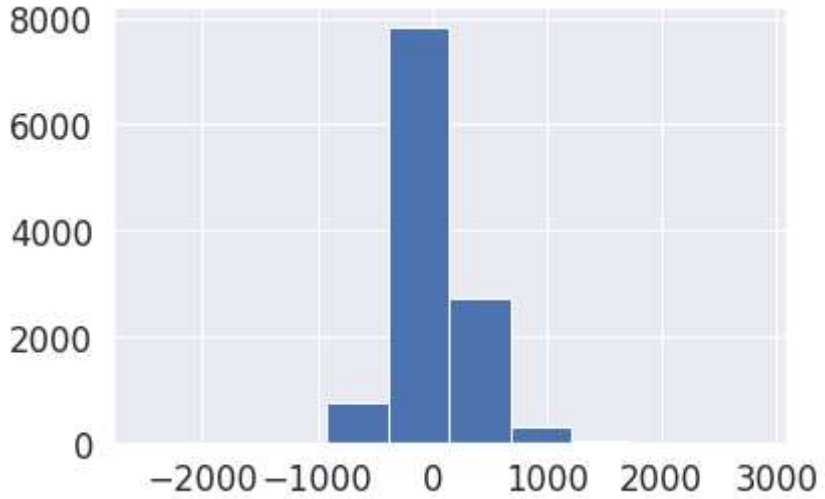
```
State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8104382475484617
Overall RMSE of linear regression model for state, 1 :- 307.8275861848434


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.7887446497855253
Overall RMSE of linear regression model for state, 45 :- 225.69615420724125
```

```
# To check the residuals
residuals=y_test-y_pred
residuals
```

```
UID
255504    281.969088
252676    -69.935775
276314    190.761969
248614   -157.290627
286865     -9.887017
             ...
238088    -67.541646
242811    -41.578757
250127   -127.427569
241096   -330.820475
287763    217.760642
Name: hc_mortgage_mean, Length: 11709, dtype: float64
```

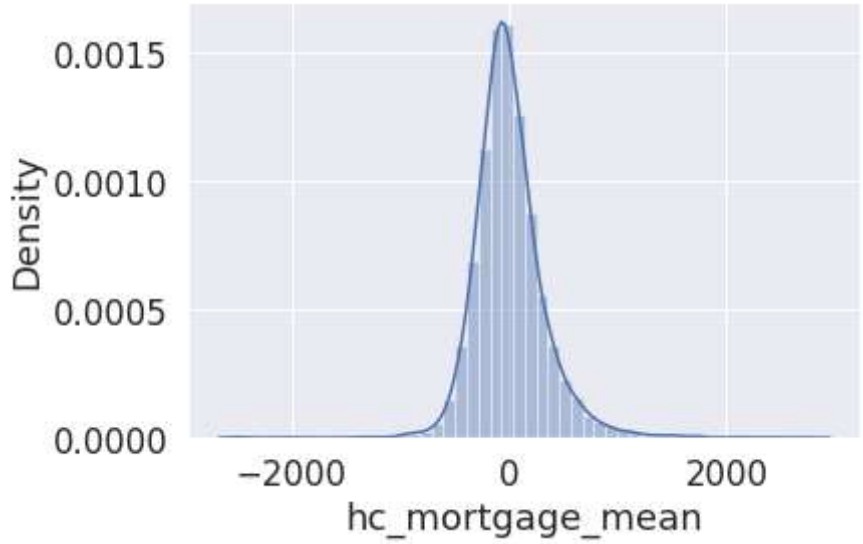
```
plt.hist(residuals) # Normal distribution of residuals
```

```
(array([6.000e+00, 3.000e+00, 2.900e+01, 7.670e+02, 7.823e+03, 2.716e+03,
        3.010e+02, 4.900e+01, 1.200e+01, 3.000e+00]),
 array([-2515.04284233, -1982.92661329, -1450.81038425,  -918.69415521,
         -386.57792617,   145.53830287,   677.65453191,  1209.77076095,
         1741.88698999,  2274.00321903,  2806.11944807]),
 <a list of 10 Patch objects>)
```



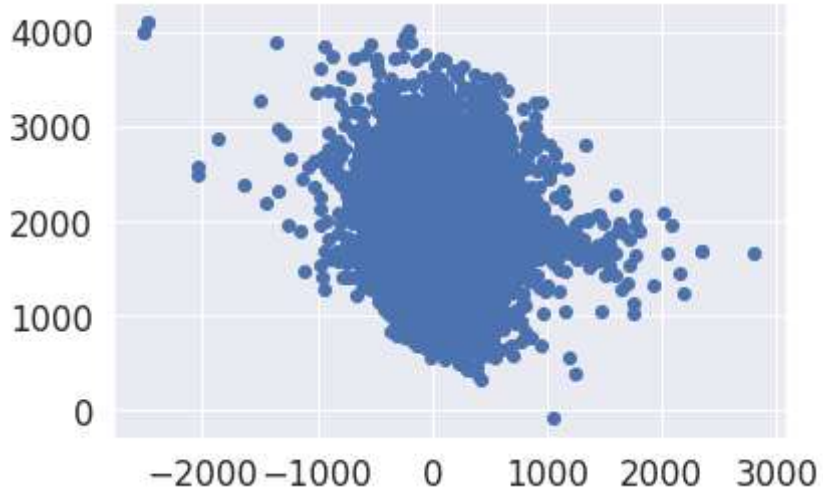

```
sns.distplot(residuals)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap

<matplotlib.axes._subplots.AxesSubplot at 0x7f4704be3860>
```




```
plt.scatter(residuals,y_pred)
# Same variance and residuals does not have correlation with predictor
# Independance of residuals
```

```
<matplotlib.collections.PathCollection at 0x7f470232c710>
```



Data Reporting:

2. 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.)

Click Here To See DashBoard