In [61]:
```python
import pandas as pd
pd.set_option ('display.max_columns', None)
```

In [62]:
```python
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

In [63]:
```python
df_train.head()
```

Out[63]:

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

In [64]:
```python
df_test.head()
```

Out[64]:

	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	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town

In [65]:
```python
df_train.shape
```

Out[65]:
```
(27321, 80)
```

In [66]:
```python
df_test.shape
```

Out[66]:
```
(11709, 80)
```

In [67]:
```python
#Figure out the primary key and look for the requirement of indexing
```

In [68]:
```python
len(set(df_train['UID']).intersection(set(df_test['UID'])))
```

Out[68]:
```
123
```

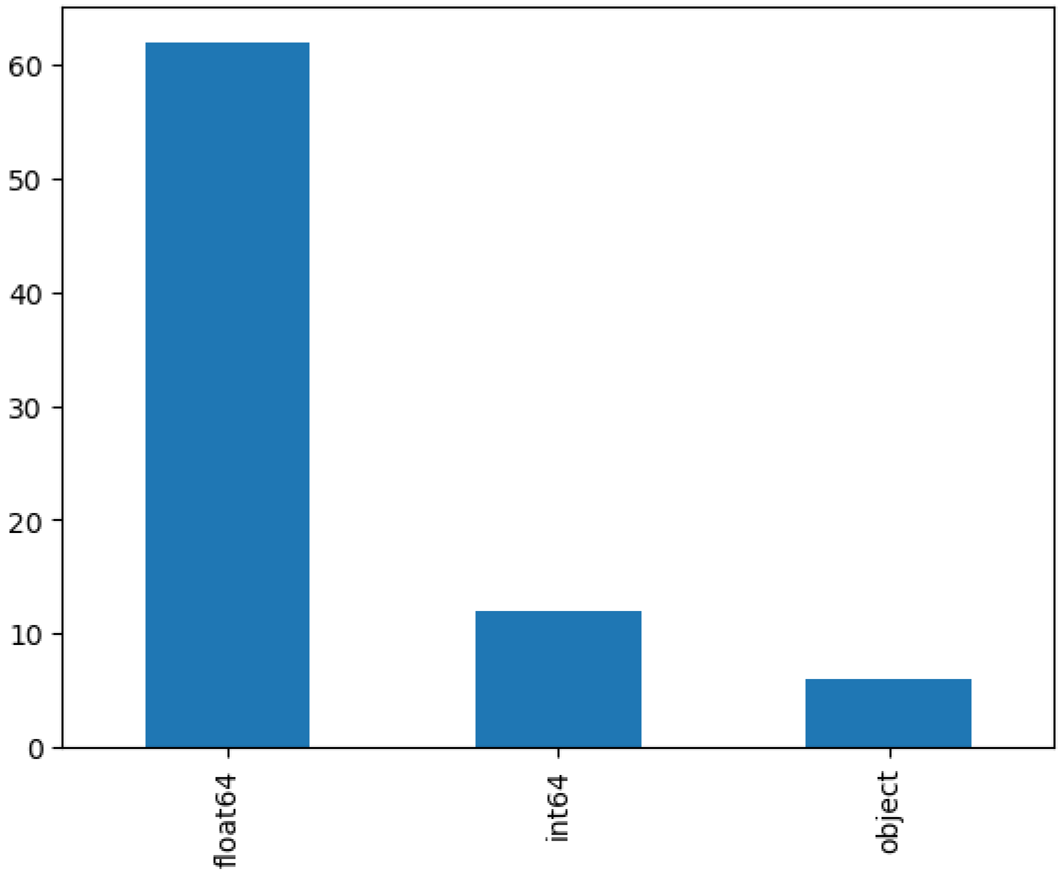
In [69]:
```python
df_train.dtypes
```

Out[69]:
```
UID              int64
BLOCKID        float64
SUMLEVEL         int64
COUNTYID         int64
STATEID          int64
```

```
                ...
pct_own         float64
married         float64
married_snp     float64
separated       float64
divorced        float64
Length: 80, dtype: object
```

In [70]:
```python
df_train.dtypes.value_counts().plot(kind='bar')
```

Out[70]:
```
<Axes: >
```



In [71]:
```python
df_train.describe(include='O')
```

Out[71]:

	state	state_ab	city	place	type	primary
count	27321	27321	27321	27321	27321	27321
unique	52	52	6916	9912	6	1
top	California	CA	Chicago	New York City	City	tract
freq	2926	2926	294	490	15237	27321

In [72]:
```python
#Gauge the fill rate of the variables and devise plans for missing value treatment.
#Please explain explicitly the reason for the treatment chosen for each variable
```

In [73]:
```python
df_train['split']='Train'
df_test['split']='Test'
```

In [74]:
```python
df_combined = pd.concat([df_train, df_test], ignore_index=True)
df_combined.head()
```

Out[74]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primar
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tra

1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tra
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tra
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tra
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tra

In [75]:

```
df_combined.tail()
```

Out[75]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	typ
39025	238088	NaN	140	105	12	Florida	FL	Lakeland	Crystal Springs	Cit
39026	242811	NaN	140	31	17	Illinois	IL	Chicago	Chicago City	Villag
39027	250127	NaN	140	9	25	Massachusetts	MA	Lawrence	Methuen Town City	Cit
39028	241096	NaN	140	27	19	Iowa	IA	Carroll	Carroll City	Cit
39029	287763	NaN	140	453	48	Texas	TX	Austin	Sunset Valley City	Tow

In [76]:

```
df_combined.shape
```

Out[76]:

```
(39030, 81)
```

In [77]:

```
df_combined.isna().sum()
```

Out[77]:

```
UID                0
BLOCKID        39030
SUMLEVEL           0
COUNTYID           0
STATEID            0
               ...
married          275
married_snp      275
separated        275
divorced         275
split              0
Length: 81, dtype: int64
```

In [78]:

```
1-df_combined.isna().sum()/len(df_combined)
```

Out[78]:

```
UID            1.000000
BLOCKID        0.000000
SUMLEVEL       1.000000
COUNTYID       1.000000
STATEID        1.000000
                 ...
married        0.992954
married_snp    0.992954
separated      0.992954
divorced       0.992954
split          1.000000
Length: 81, dtype: float64
```

In [79]:

```
df_combined.drop(columns = ['BLOCKID'], axis=1, inplace=True)
```

In [80]:
```
df_combined.isna().sum()/len(df_combined)*100
```

Out[80]:
```
UID              0.000000
SUMLEVEL         0.000000
COUNTYID         0.000000
STATEID          0.000000
state            0.000000
                   ...
married          0.704586
married_snp      0.704586
separated        0.704586
divorced         0.704586
split            0.000000
Length: 80, dtype: float64
```

In [81]:
```
col_check=df_combined.isna().sum().to_frame().reset_index()
null_col=col_check[col_check[0]>0]['index'].tolist()
null_col
```

Out[81]:
```
['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',
 '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']
```

In [82]:

```
for i in null_col:
    print(i)
    if df_combined[i].nunique()>8:
        df_combined[i].fillna(df_combined[i].median())
    else:df_combined[i].fillna(df_combined[i].mode()[0])
```

```
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
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
```

In [83]:
```
df_combined.isna().sum()/len(df_combined)*100
```

Out[83]:
```
UID             0.000000
SUMLEVEL        0.000000
COUNTYID        0.000000
STATEID         0.000000
state           0.000000
                  ...
married         0.704586
married_snp     0.704586
separated       0.704586
divorced        0.704586
split           0.000000
Length: 80, dtype: float64
```

In [84]:
```
df_combined.shape
```

Out[84]:
```
(39030, 80)
```

In [85]:
```
df_combined.drop_duplicates(subset=['UID'],inplace=True)
df_combined.shape
```

Out[85]:
```
(38715, 80)
```

In [86]:
```
#Explore the top 2,500 locations where the percentage of households with a second mortga
#Visualize using geo-map. You may keep the upper limit for the percent of households wit
top_2500_loc=df_train[(df_train['second_mortgage']<0.50) &
                      (df_train['pct_own']>0.10)].sort_values(by='second_mortgage', asce
```

In [87]:
```
top_2500_loc=top_2500_loc[['state','city','state_ab','place','lat','lng']]
top_2500_loc.head()
```

Out[87]:

	state	city	state_ab	place	lat	lng
11980	Massachusetts	Worcester	MA	Worcester City	42.254262	-71.800347
26018	New York	Corona	NY	Harbor Hills	40.751809	-73.853582
7829	Maryland	Glen Burnie	MD	Glen Burnie	39.127273	-76.635265
2077	Florida	Tampa	FL	Egypt Lake-leto	28.029063	-82.495395
1701	Illinois	Chicago	IL	Lincolnwood	41.967289	-87.652434

In [88]:
```
!pip install geopandas
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: geopandas in c:\users\windows\anaconda3\lib\site-packages
(1.0.1)
Requirement already satisfied: numpy>=1.22 in c:\users\windows\anaconda3\lib\site-packag
es (from geopandas) (1.26.4)
Requirement already satisfied: pyogrio>=0.7.2 in c:\users\windows\anaconda3\lib\site-pac
```

```
kages (from geopandas) (0.9.0)
Requirement already satisfied: packaging in c:\users\windows\anaconda3\lib\site-packages
(from geopandas) (23.2)
Requirement already satisfied: pandas>=1.4.0 in c:\users\windows\anaconda3\lib\site-pack
ages (from geopandas) (2.2.2)
Requirement already satisfied: pyproj>=3.3.0 in c:\users\windows\anaconda3\lib\site-pack
ages (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=2.0.0 in c:\users\windows\anaconda3\lib\site-pac
kages (from geopandas) (2.0.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\windows\anaconda3\lib
\site-packages (from pandas>=1.4.0->geopandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\windows\anaconda3\lib\site-packa
ges (from pandas>=1.4.0->geopandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\windows\anaconda3\lib\site-pac
kages (from pandas>=1.4.0->geopandas) (2023.3)
Requirement already satisfied: certifi in c:\users\windows\anaconda3\lib\site-packages
(from pyogrio>=0.7.2->geopandas) (2024.7.4)
Requirement already satisfied: six>=1.5 in c:\users\windows\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas>=1.4.0->geopandas) (1.16.0)
```

In [89]:

```python
import geopandas as gpd
gdf = gpd.GeoDataFrame(top_2500_loc, geometry=gpd.points_from_xy(x=top_2500_loc.lng, y=t
gdf
```

Out[89]:

	state	city	state_ab	place	lat	lng	geometry
11980	Massachusetts	Worcester	MA	Worcester City	42.254262	-71.800347	POINT (-71.80035 42.25426)
26018	New York	Corona	NY	Harbor Hills	40.751809	-73.853582	POINT (-73.85358 40.75181)
7829	Maryland	Glen Burnie	MD	Glen Burnie	39.127273	-76.635265	POINT (-76.63526 39.12727)
2077	Florida	Tampa	FL	Egypt Lake-leto	28.029063	-82.495395	POINT (-82.4954 28.02906)
1701	Illinois	Chicago	IL	Lincolnwood	41.967289	-87.652434	POINT (-87.65243 41.96729)
...	...	...	...	...	...	...	...
17914	North Carolina	Raleigh	NC	Raleigh City	35.757135	-78.704288	POINT (-78.70429 35.75713)
5478	California	Marina Del Rey	CA	Marina Del Rey	33.983204	-118.466139	POINT (-118.46614 33.9832)
25642	Maryland	Baltimore	MD	Lochearn	39.353095	-76.733315	POINT (-76.73331 39.3531)
26671	Pennsylvania	Philadelphia	PA	Philadelphia City	40.039070	-75.125135	POINT (-75.12514 40.03907)
24443	California	Manteca	CA	Manteca City	37.732143	-121.242902	POINT (-121.2429 37.73214)

2500 rows × 7 columns

In [90]:

```python
#Bad Debt Equation:
#Bad Debt = P (Second Mortgage ∩ Home Equity Loan)
#Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage
df_combined['bad_debt']=df_combined['second_mortgage']+df_combined['home_equity']-df_com
df_combined.head()
```

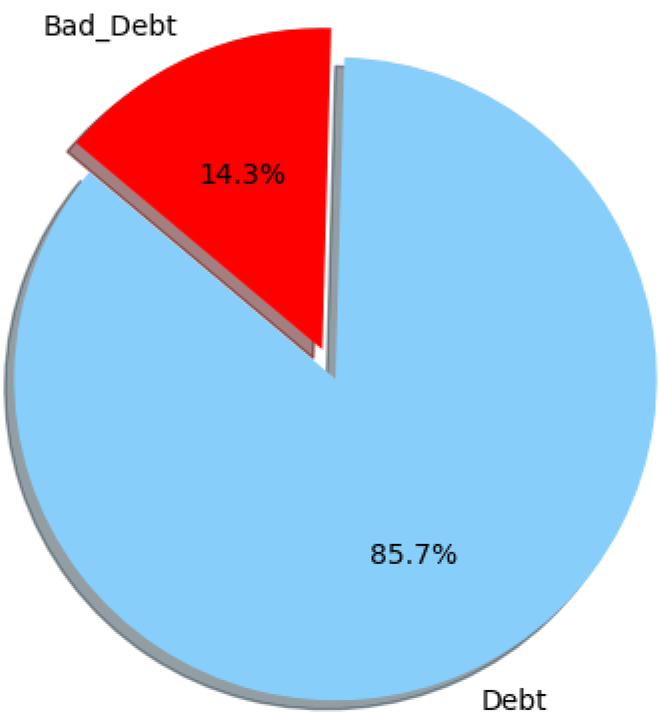
Out[90]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_cod
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	1334

1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	4661
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	4612
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	92
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	6650

In [91]:

```
#Create pie charts to show overall debt and bad debt
import matplotlib.pyplot as plt
labels = 'Debt', 'Bad_Debt'
sizes = [df_combined['debt'].mean()*100, df_combined['bad_debt'].mean()*100]
colors = ['lightskyblue', 'red']
explode = (0.1,0)

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

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



In [92]:

```
#Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity,

df_combined['good_debt']=df_combined['debt']-df_combined['bad_debt']
df_combined.head()
```

Out[92]:

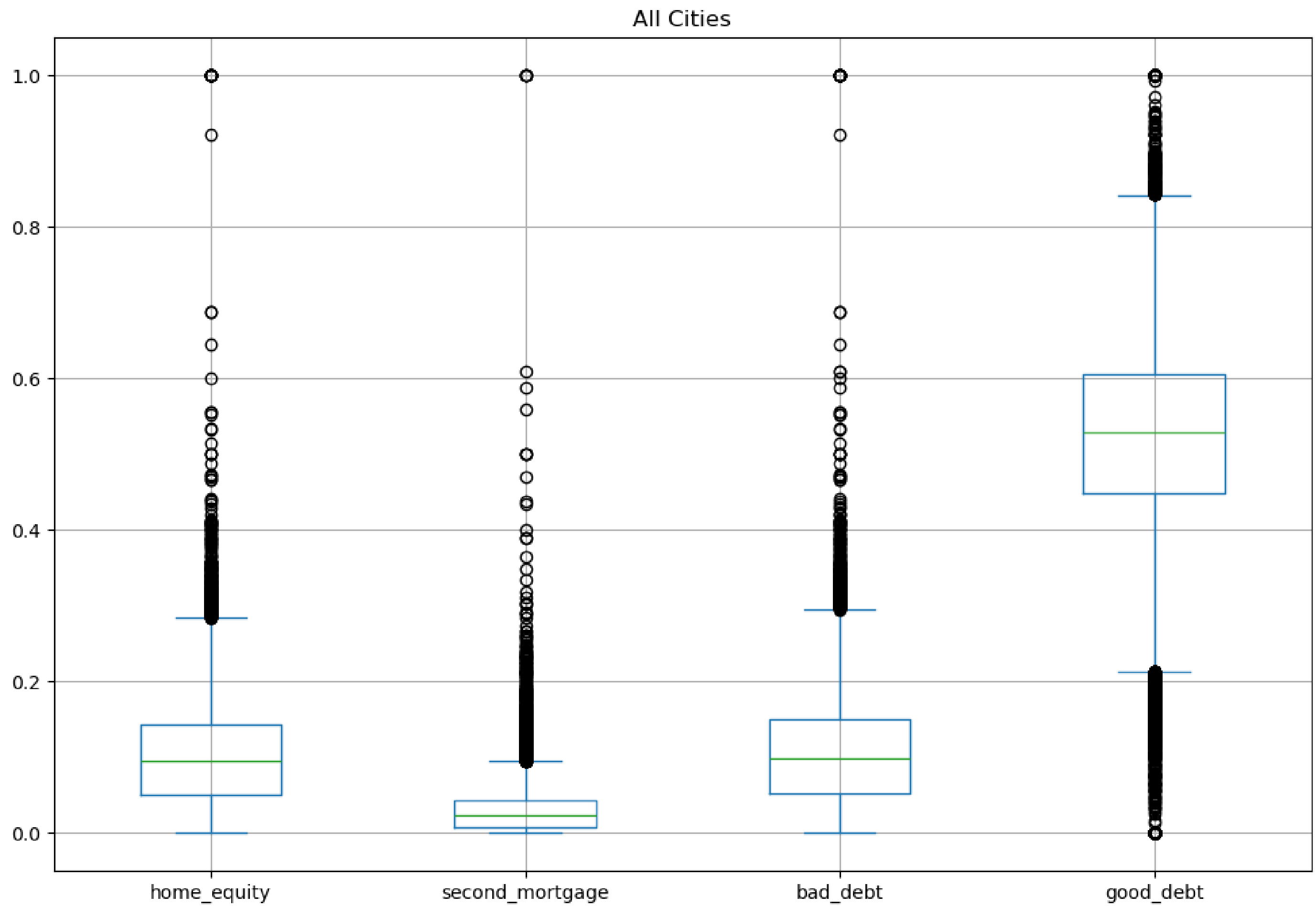
	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_cod
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	1334
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	4661
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	4612
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	92
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan	City	tract	6650

City

In [93]:
```
df_combined.columns
```

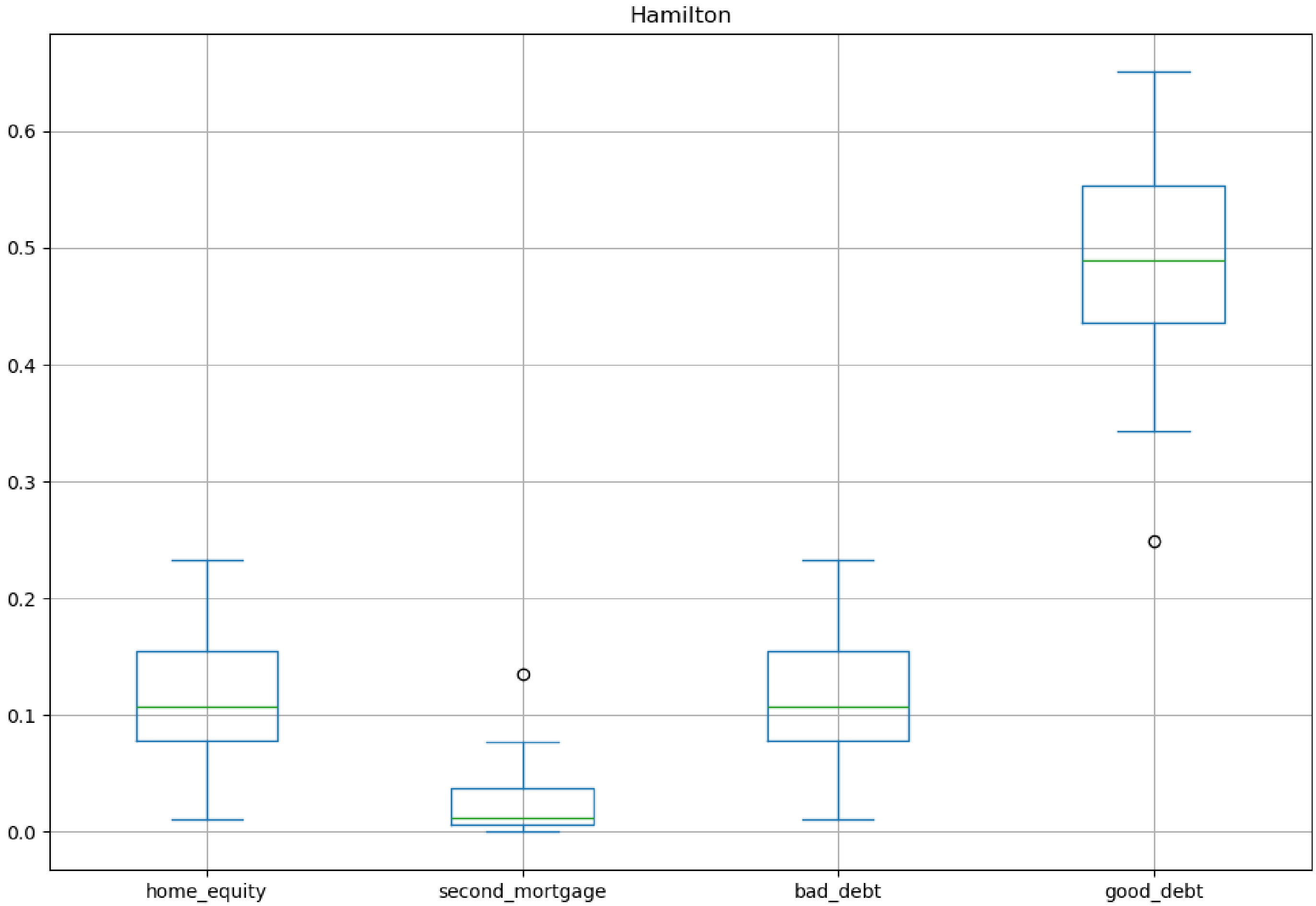
Out[93]:
```
Index(['UID', 'SUMLEVEL', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city',
       'place', 'type', 'primary', 'zip_code', 'area_code', 'lat', 'lng',
       'ALand', 'AWater', 'pop', 'male_pop', 'female_pop', 'rent_mean',
       'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples',
       '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', 'split', 'bad_debt', 'good_debt'],
      dtype='object')
```

In [94]:
```
all_cities = df_combined[['home_equity','second_mortgage','bad_debt','good_debt']]
all_cities.plot.box(figsize=(12,8),grid=True)
plt.title('All Cities')
plt.show()
```

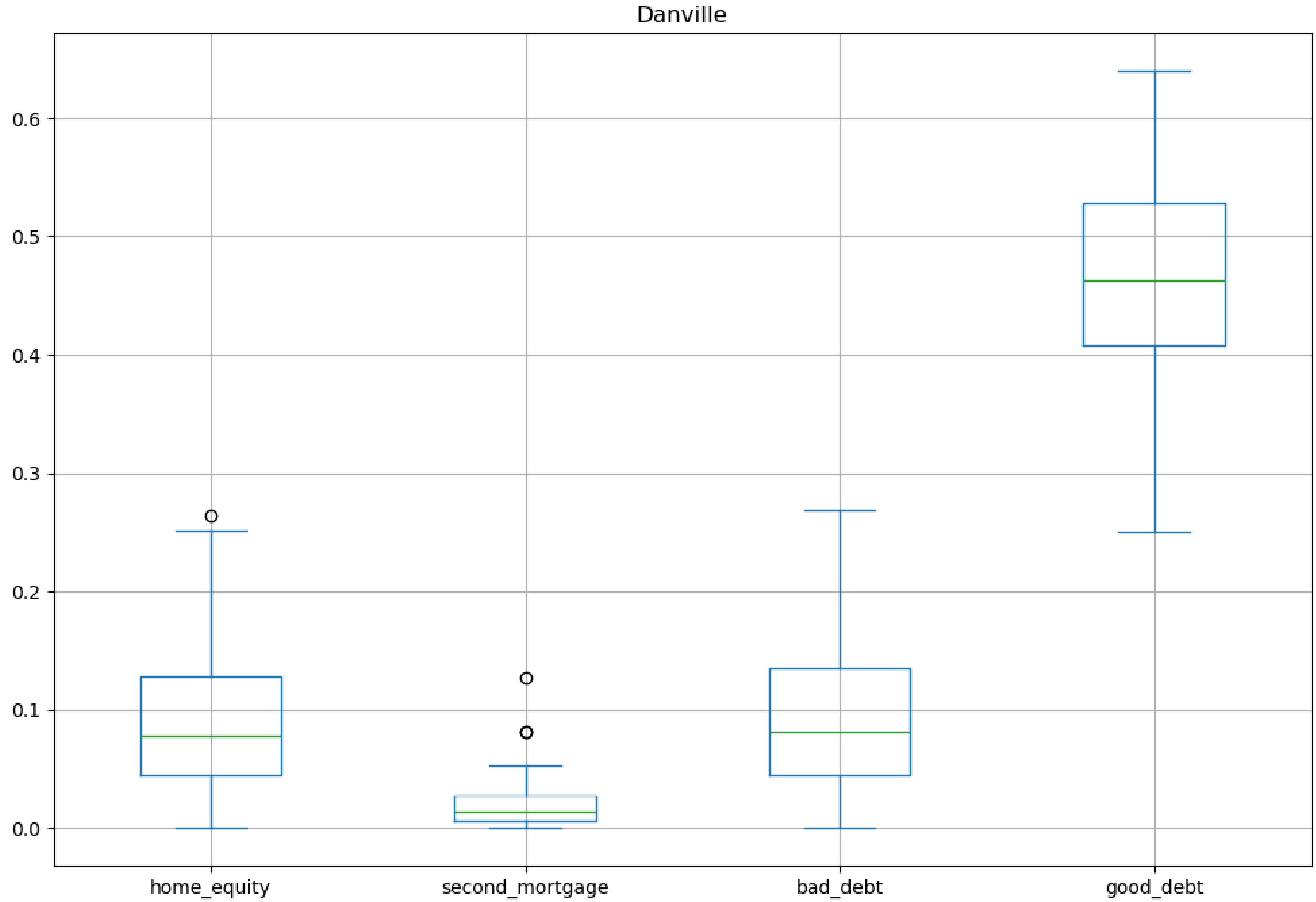


In [95]:
```
hamilton = df_combined[df_combined['city']=='Hamilton']
hamilton = hamilton[['home_equity','second_mortgage','bad_debt','good_debt']]
hamilton.plot.box(figsize=(12,8),grid=True)
```

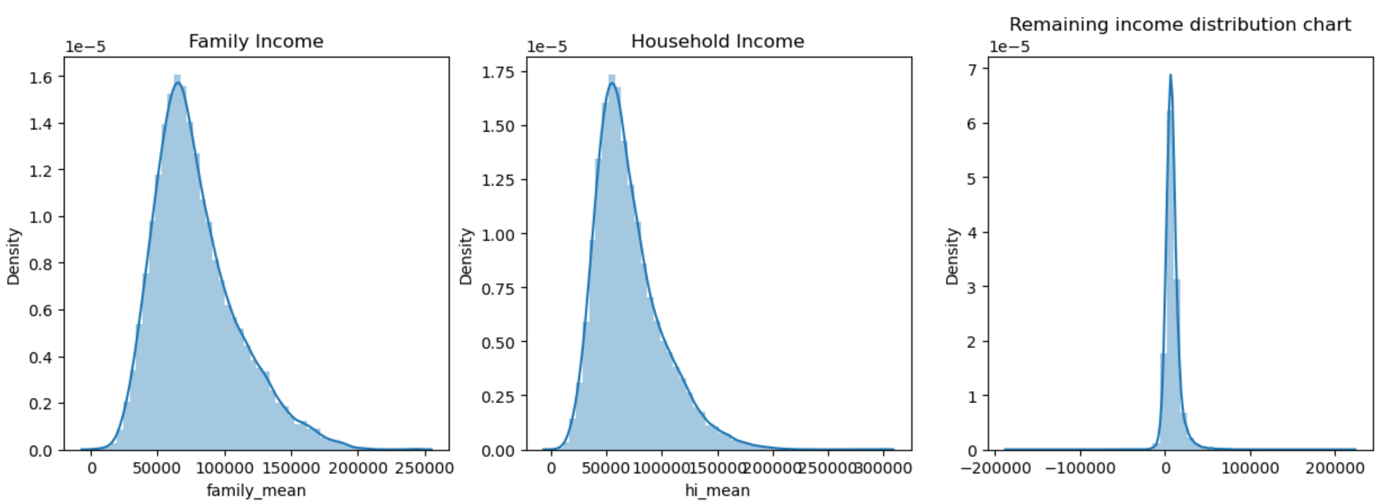
```
plt.title('Hamilton')
plt.show()
```



In [96]:
```
Danville = df_combined[df_combined['city']=='Danville']
Danville = Danville[['home_equity','second_mortgage','bad_debt','good_debt']]
Danville.plot.box(figsize=(12,8),grid=True)
plt.title('Danville')
plt.show()
```



In [97]:
```
#Create a collated income distribution chart for family income, house hold income and re

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

plt.subplot(2,3,1)
sns.distplot(df_train['family_mean'])
plt.title('Family Income')
plt.subplot(2,3,2)
sns.distplot(df_train['hi_mean'])
plt.title('Household Income')
plt.subplot(2,3,3)
sns.distplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
```



In [98]:
```
#Population density and median age
```

```
df_combined['population_density']=df_combined['pop']/df_combined['ALand']
df_combined.head()
```

Out[98]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	1334
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	4661
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	4612
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	92
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	6650

In [99]:

```
# Weighted average
# median_age=((male_age_median * male_pop)+(female_age_median*female_pop))/(male_pop+fem
#            =((40*10)+(50*30))/40
#            =(400+1500)/40
#            =190/4
#            =47.5

df_combined['median_age']=((df_combined['male_age_median']*df_combined['male_pop'])+(df_
df_combined.head()
```

Out[99]:

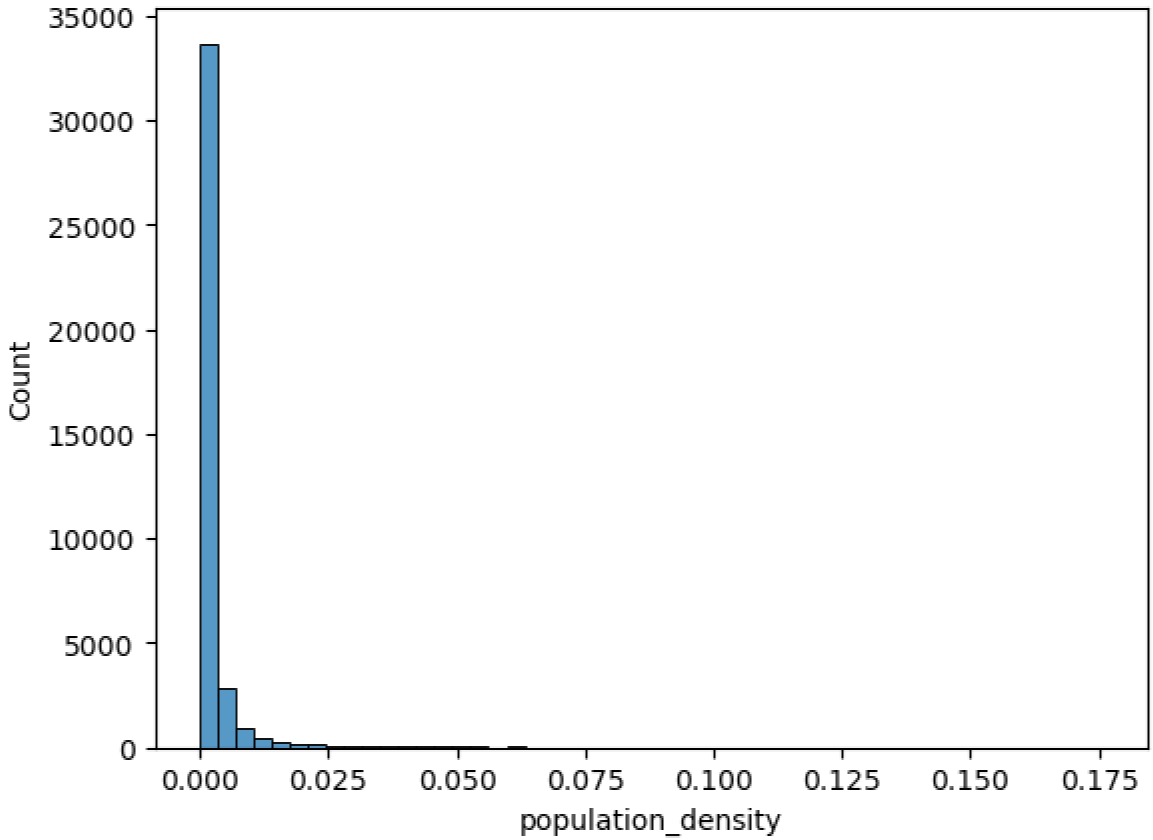
	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	1334
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	4661
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	4612
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	92
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	6650

In [100…

```
#Visualize the findings using appropriate chart type

sns.histplot(df_combined['population_density'], bins=50)
```

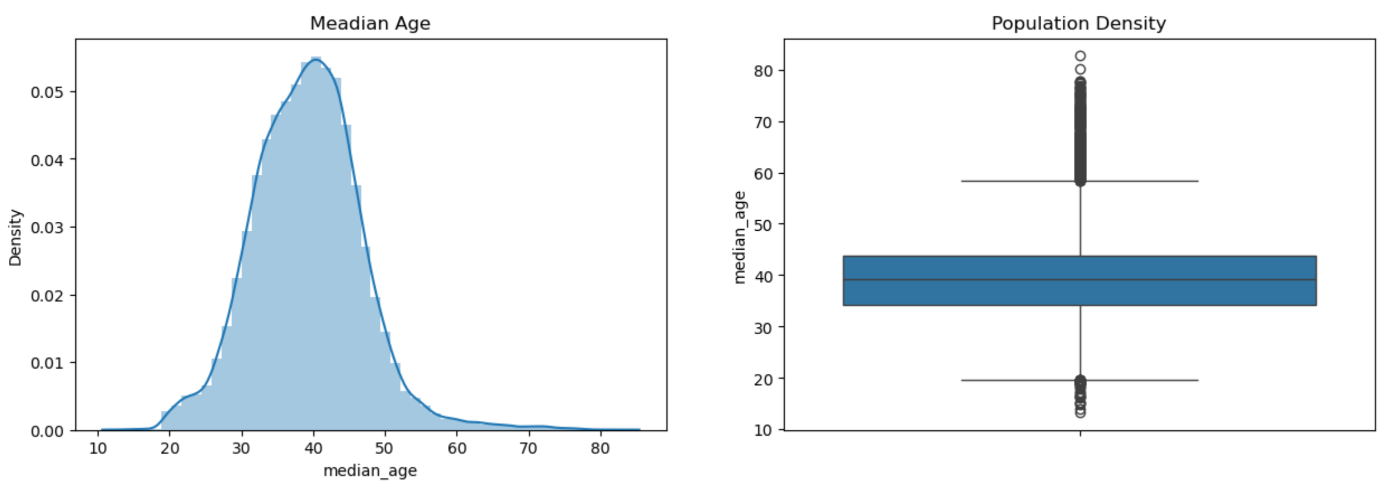
Out[100]:

```
<Axes: xlabel='population_density', ylabel='Count'>
```



In [101…

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.distplot(df_combined['median_age'])
plt.title('Meadian Age')
plt.subplot(2,2,2)
sns.boxplot(df_combined['median_age'])
plt.title('Population Density')
plt.show()
```



In [102…

```
#Create bins for population into a new variable by selecting appropriate class interval

df_combined['pop_bins']=pd.cut(df_combined['pop'],bins=5,labels=['very low','low','mediu
df_combined['pop_bins'].value_counts()
```

Out[102]:

```
pop_bins
very low     38350
low            348
medium          12
high             4
very high        1
Name: count, dtype: int64
```

In [103…

```
#Analyze the married, separated and divorced population for these population brackets

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

Out[103]:

pop_bins	married	separated	divorced
very low	38154	38154	38154
low	348	348	348
medium	12	12	12
high	4	4	4
very high	1	1	1

In [104…

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

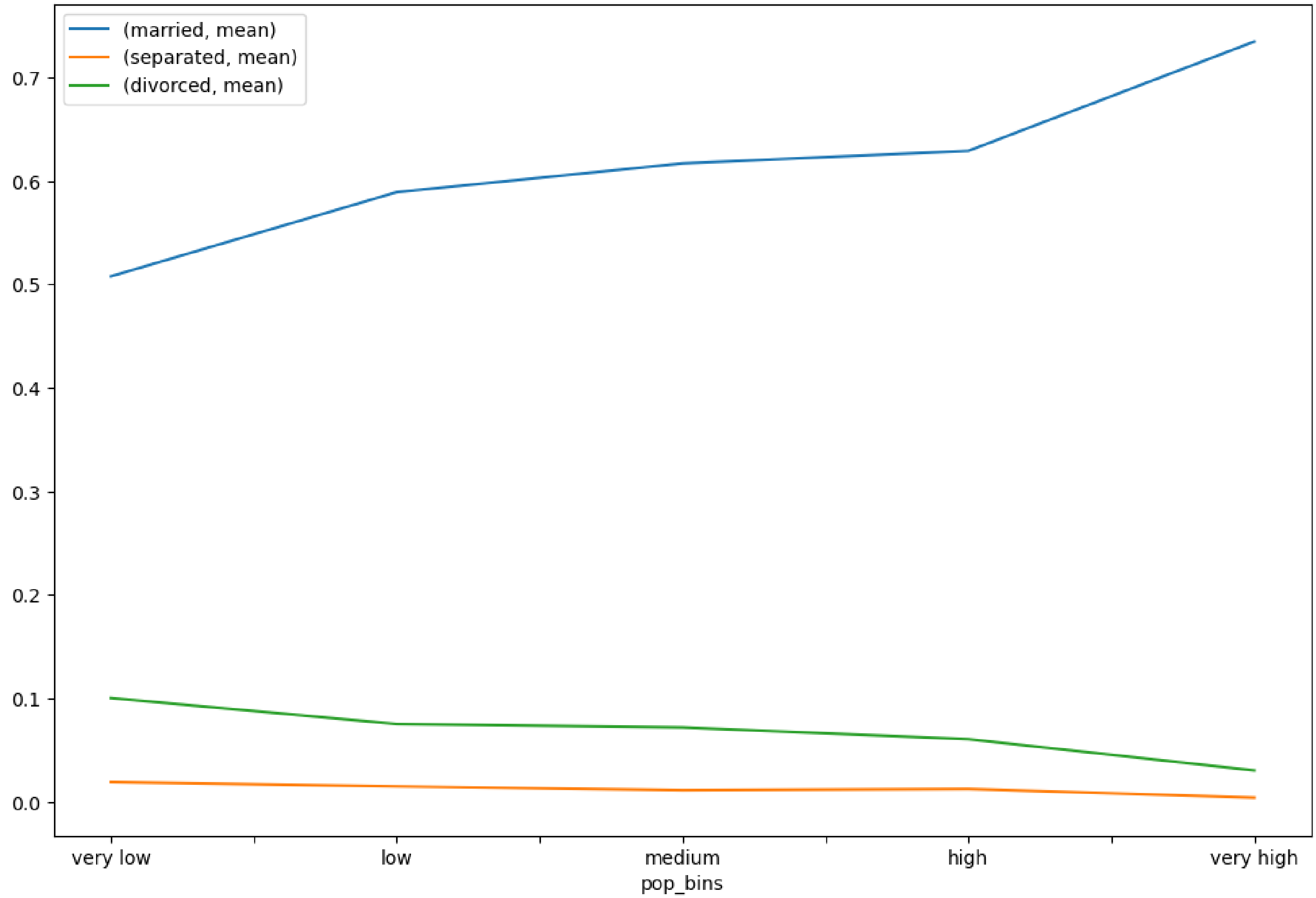
Out[104]:

	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.507906	0.526015	0.019156	0.013620	0.100353	0.09539
low	0.589247	0.601815	0.014929	0.010255	0.075192	0.06934
medium	0.617047	0.605765	0.011203	0.007745	0.071870	0.06909
high	0.629132	0.675095	0.012372	0.007340	0.060562	0.05987
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.03036

In [105…

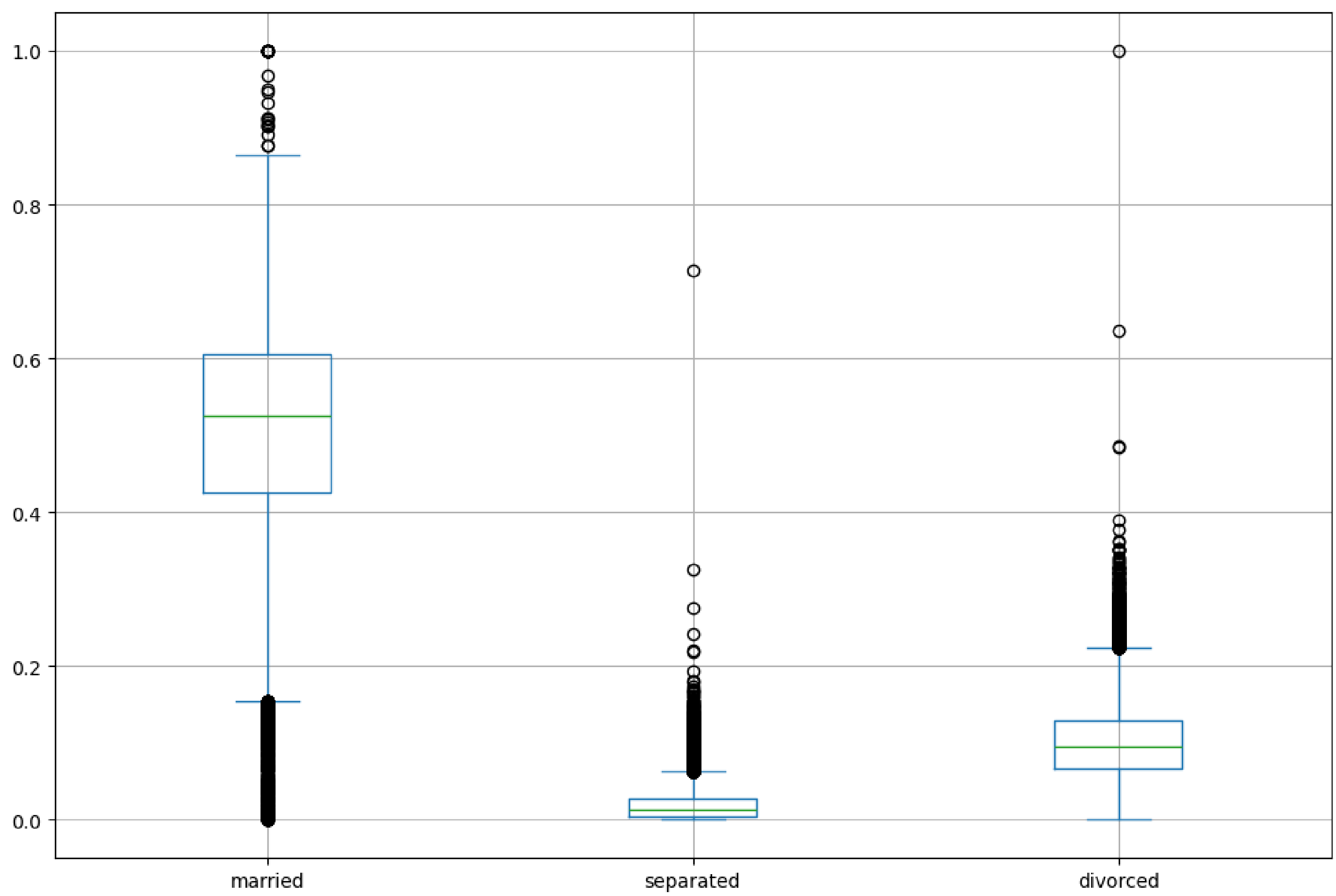
```
#Visualize using appropriate chart type

plt.figure(figsize=(12,8))
pop_bin_married=df_combined.groupby(by='pop_bins')[['married','separated','divorced']].a
pop_bin_married.plot(figsize=(12,8))
plt.legend(loc='best')
plt.show()
```

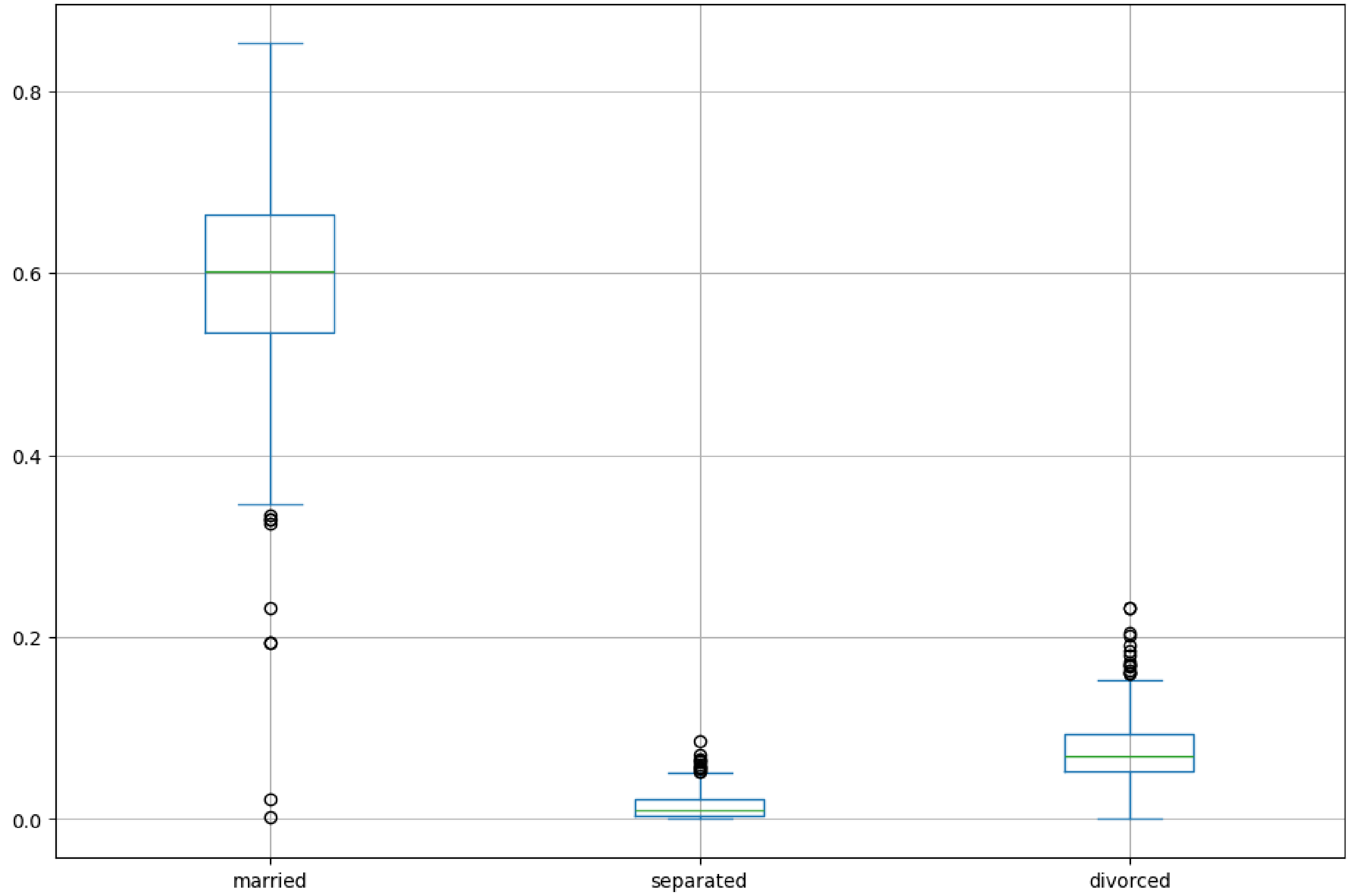
```
<Figure size 1200x800 with 0 Axes>
```

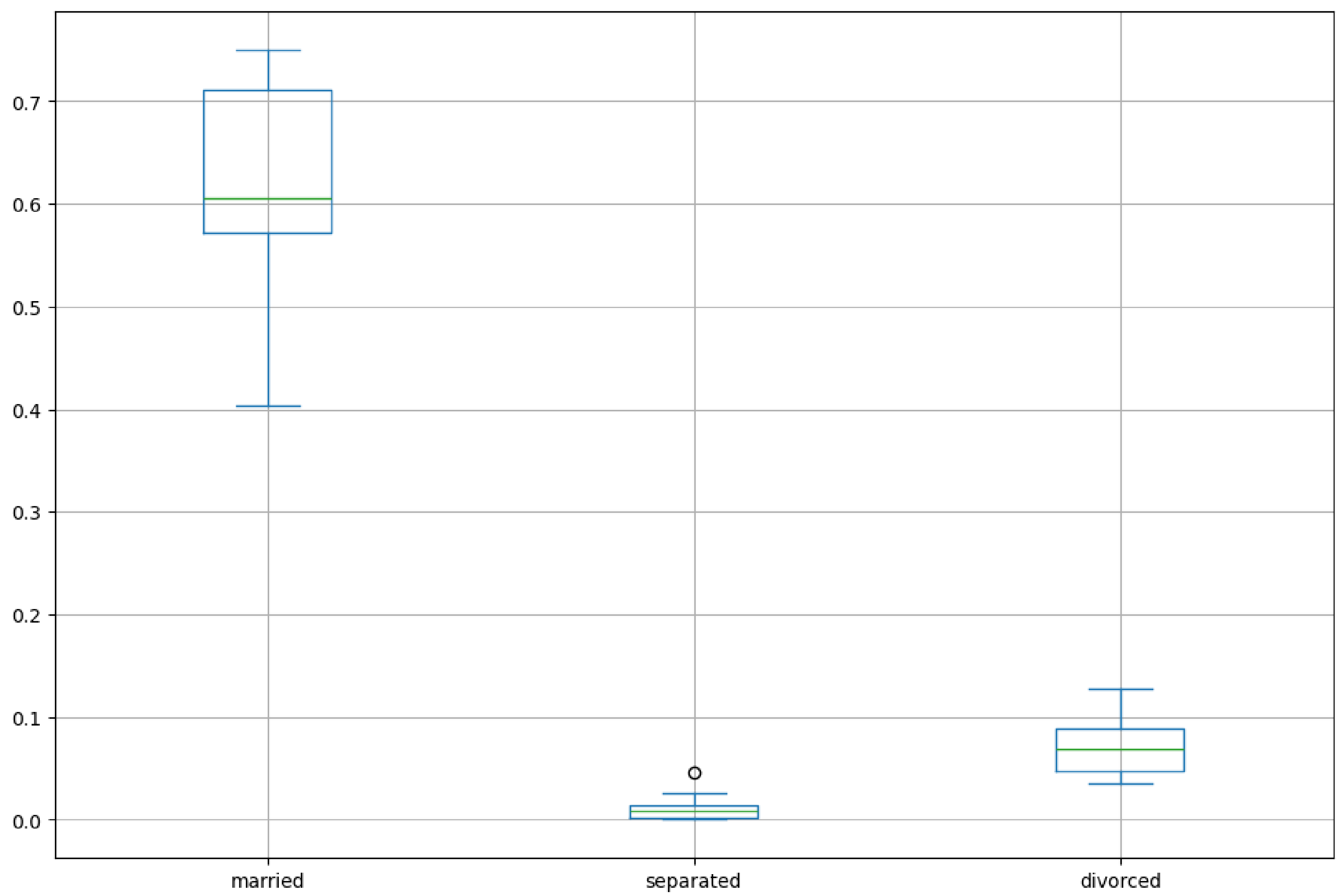


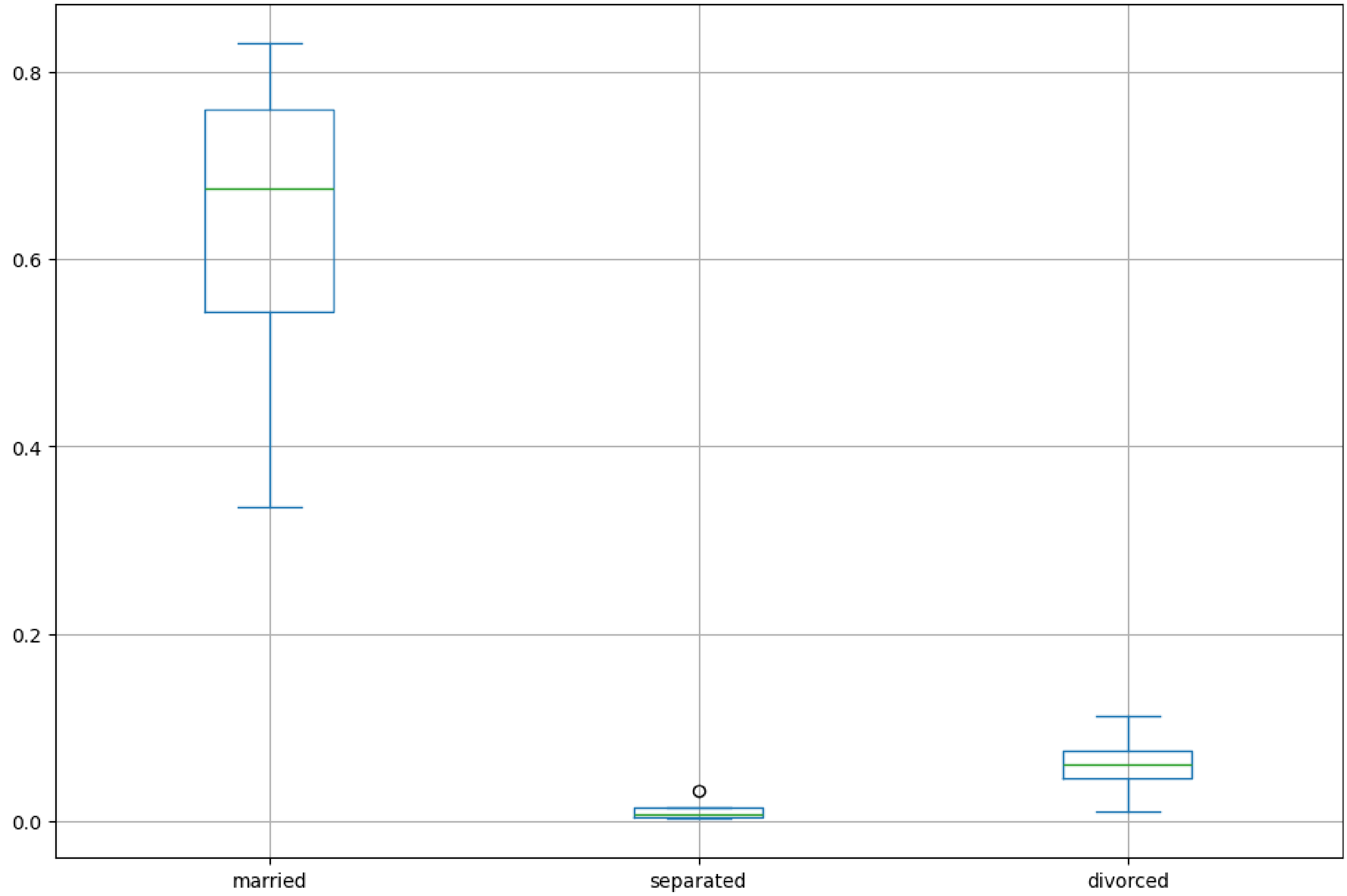
In [106…

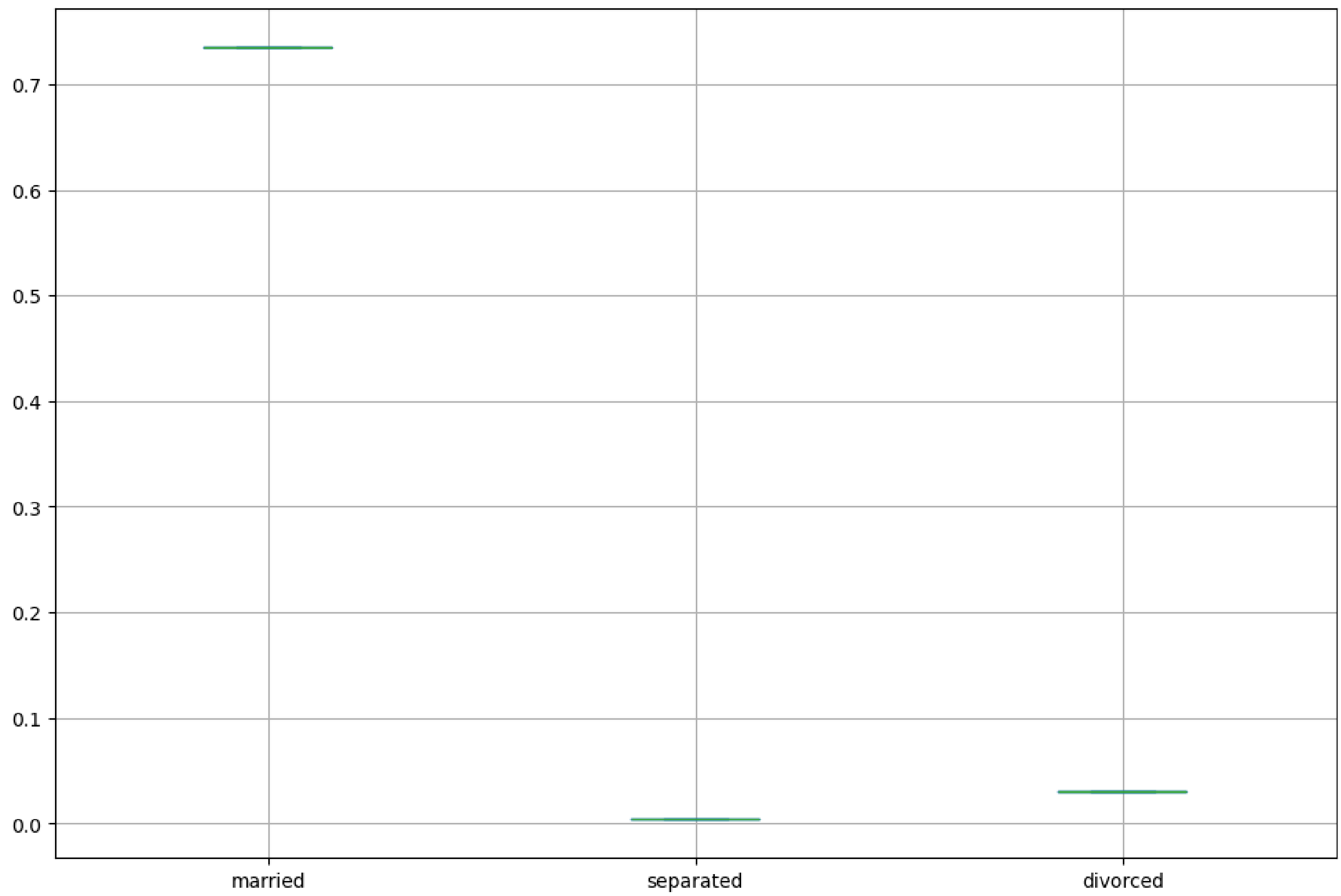
```
df_combined.groupby(by='pop_bins')[['married','separated','divorced']].plot.box(figsize=
plt.show()
```




0.8
0.6
0.4
0.2
0.0
married
separated
divorced


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
married
separated
divorced





In [107…

```
#detail your observations for rent as a percentage of income at an overall level and for

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

Out[107]:

mean

state	
Alabama	764.950875
Alaska	1190.093590
Arizona	1086.197310
Arkansas	715.227833
California	1469.756239

In [108…

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

Out[108]:

	mean
state	
Alabama	65274.351137
Alaska	91911.137520
Arizona	73015.490954
Arkansas	64210.760825
California	87816.098481

In [109…

```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']*100
rent_perc_of_income.head(10)
```

Out[109]:

```
state
Alabama                 1.171901
Alaska                  1.294831
Arizona                 1.487626
Arkansas                1.113875
California              1.673675
Colorado                1.360643
Connecticut             1.271173
Delaware                1.311566
District of Columbia    1.357802
Florida                 1.579205
Name: mean, dtype: float64
```

In [110…

```
sum(df_combined['rent_mean'])/sum(df_combined['family_mean'])
```

Out[110]:

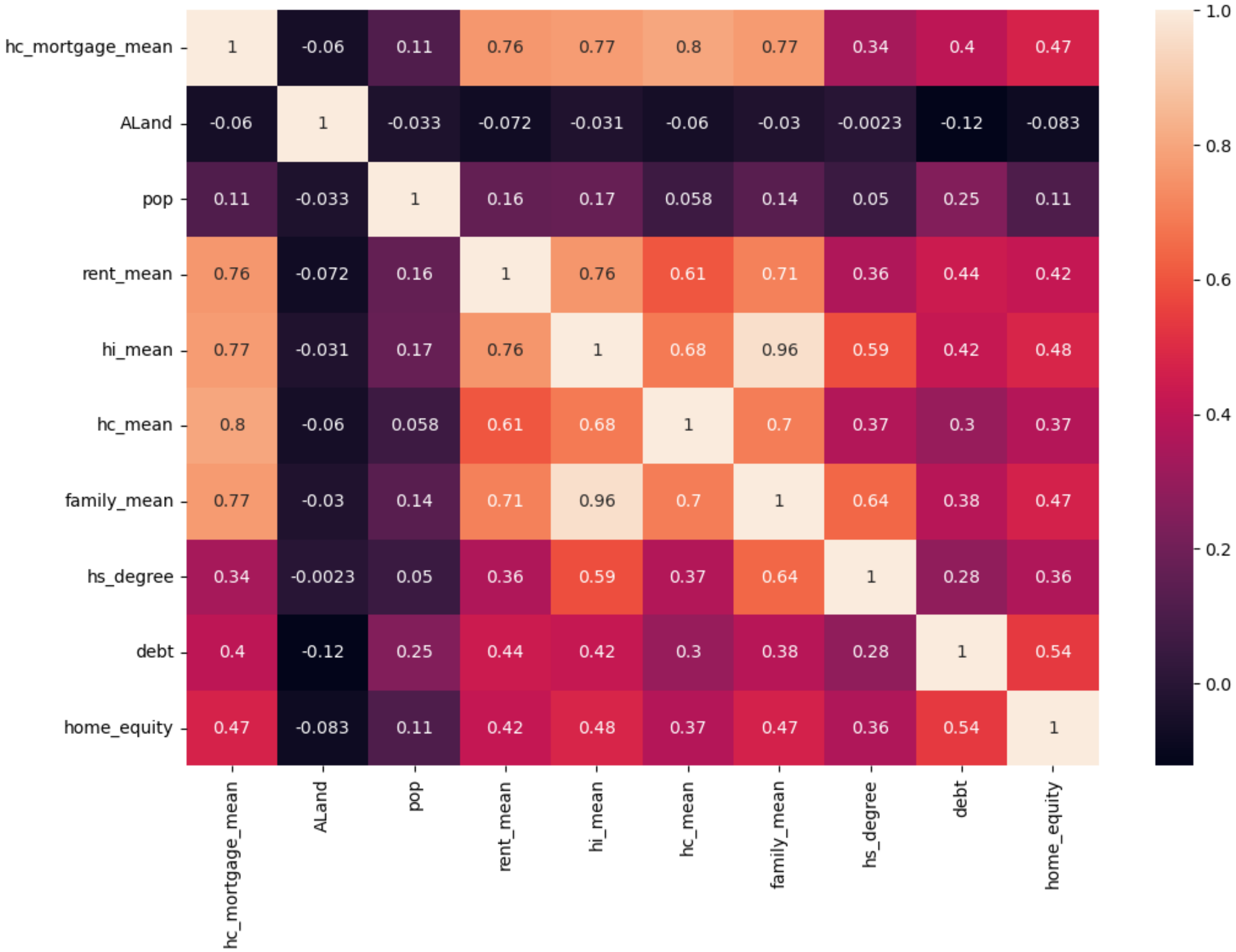
```
nan
```

In [111…

```
#Perform correlation analysis for all the relevant variables by creating a heatmap

plt.figure(figsize=(12,8))
sns.heatmap(data=df_combined[['hc_mortgage_mean','ALand','pop','rent_mean','hi_mean','hc
                             'hs_degree','debt','home_equity']].corr(),annot=True)
```

Out[111]:

```
<Axes: >
```



In [112…

```
train= df_combined[df_combined['split']=='Train']
test= df_combined[df_combined['split']=='Test']
train.head()
```

Out[112]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_co
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	133
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	466
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	461
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	9
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	665

In [113…

```
test.head()
```

Out[113]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	prima
27321	255504	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tr
27322	252676	140	1	23	Maine	ME	Auburn	Auburn City	City	tr
27323	276314	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tr
27324	248614	140	231	21	Kentucky	KY	Monticello	Monticello	City	tr

								City		
27325	286865	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tr

In [114…

```
!pip install factor_analyzer
```

```
Requirement already satisfied: factor_analyzer in c:\users\windows\anaconda3\lib\site-pa
ckages (0.5.1)
Requirement already satisfied: pandas in c:\users\windows\anaconda3\lib\site-packages (f
rom factor_analyzer) (2.2.2)
Requirement already satisfied: scipy in c:\users\windows\anaconda3\lib\site-packages (fr
om factor_analyzer) (1.13.1)
Requirement already satisfied: numpy in c:\users\windows\anaconda3\lib\site-packages (fr
om factor_analyzer) (1.26.4)
Requirement already satisfied: scikit-learn in c:\users\windows\anaconda3\lib\site-packa
ges (from factor_analyzer) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\windows\anaconda3\lib
\site-packages (from pandas->factor_analyzer) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\windows\anaconda3\lib\site-packa
ges (from pandas->factor_analyzer) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\windows\anaconda3\lib\site-pac
kages (from pandas->factor_analyzer) (2023.3)
Requirement already satisfied: joblib>=1.2.0 in c:\users\windows\anaconda3\lib\site-pack
ages (from scikit-learn->factor_analyzer) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\windows\anaconda3\lib\si
te-packages (from scikit-learn->factor_analyzer) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\windows\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas->factor_analyzer) (1.16.0)
```

In [115…

```
import numpy as np
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [116…

```
df_train.describe().T
```

Out[116]:

	count	mean	std	min	25%	50%	75%
UID	27321.0	257331.996303	21343.859725	220342.0	238816.000000	257220.000000	275818.000000
BLOCKID	0.0	NaN	NaN	NaN	NaN	NaN	NaN
SUMLEVEL	27321.0	140.000000	0.000000	140.0	140.000000	140.000000	140.000000
COUNTYID	27321.0	85.646426	98.333097	1.0	29.000000	63.000000	109.000000
STATEID	27321.0	28.271806	16.392846	1.0	13.000000	28.000000	42.000000
...	...	...	...	...	...	...	...
pct_own	27053.0	0.640434	0.226640	0.0	0.502780	0.690840	0.817460
married	27130.0	0.508300	0.136860	0.0	0.425102	0.526665	0.605760
married_snp	27130.0	0.047537	0.037640	0.0	0.020810	0.038840	0.065100
separated	27130.0	0.019089	0.020796	0.0	0.004530	0.013460	0.027488
divorced	27130.0	0.100248	0.049055	0.0	0.065800	0.095205	0.129000

74 rows × 8 columns

In [ ]:

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

In [118…

```
train.columns
```

Out[118]:
```
Index(['UID', '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', 'split', 'bad_debt', 'good_debt', 'population_density',
       'median_age', 'pop_bins'],
      dtype='object')
```

In [120…
```
train['type'].unique()
```

Out[120]:
```
array(['City', 'Urban', 'Town', 'CDP', 'Village', 'Borough'], dtype=object)
```

In [121…
```
type_dict={'type':{'City':1,'Urban':2,'Town':3,'CDP':4,'Village':5,'Borough':6}}
train.replace(type_dict,inplace=True)
test.replace(type_dict,inplace=True)
```

In [122…
```
train['type'].unique()
```

Out[122]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [123…
```
test['type'].unique()
```

Out[123]:
```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

In [124…
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean','second_mortga
              'pct_own', 'married','separated', 'divorced']
X_train = train[feature_cols]
y_train = train['hc_mortgage_mean']

X_test = test[feature_cols]
y_test = test['hc_mortgage_mean']
```

In [142…
```
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error,accuracy_sc
```

In [154…
```
X_train.head()
```

Out[154]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_deg
0	53	36	13346	1	5230	67994.14790	0.02077	0.08919	0.52963	0.89
1	141	18	46616	1	2633	50670.10337	0.02222	0.04274	0.60855	0.90
2	63	18	46122	1	6881	95262.51431	0.00000	0.09512	0.73484	0.94
3	127	72	927	2	2700	56401.68133	0.01086	0.01086	0.52714	0.91
4	161	20	66502	1	5637	54053.42396	0.05426	0.05426	0.51938	1.00

In [156…
```
X_test.head()
```

Out[156]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs
27321	163	26	48239	4	3417	53802.87122	0.06443	0.07651	0.63624	
27322	1	23	4210	1	3796	85642.22095	0.01175	0.14375	0.64755	
27323	15	42	14871	6	3944	65694.06582	0.01316	0.06497	0.45395	
27324	231	21	42633	1	2508	44156.38709	0.00995	0.01741	0.41915	
27325	355	48	78410	3	6230	123527.02420	0.00000	0.03440	0.63188	

In [174…
```
# Imputing NaN values in X_train and X_test
imputer = SimpleImputer(strategy='mean')  # You can use 'median', 'most_frequent', etc.
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Handling NaN values in y_train and y_test
y_imputer = SimpleImputer(strategy='mean')
y_train_imputed = y_imputer.fit_transform(y_train.values.reshape(-1, 1)).ravel()
y_test_imputed = y_imputer.transform(y_test.values.reshape(-1, 1)).ravel()

# Scaling the data using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_test_scaled = scaler.transform(X_test_imputed)

# Fitting the Linear Regression model
lr = LinearRegression()
lr.fit(X_train_scaled, y_train_imputed)

# Predicting on the test set
y_pred = lr.predict(X_test_scaled)
r2 = r2_score(y_test_imputed, y_pred)
mae = mean_absolute_error(y_test_imputed, y_pred)
mse = mean_squared_error(y_test_imputed, y_pred)

print("R^2 Score:", r2)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
```

```
R^2 Score: 0.7390152562048986
Mean Absolute Error: 233.6095295950887
Mean Squared Error: 103387.38513110559
```

In [180…
```
lr.coef_
```

Out[180]:
```
array([ -28.41247847,  -21.8073659 ,  -22.89642756,  -57.25227147,
         -4.61708318,  558.88060628,   -1.36770055,   71.31440596,
         10.29092764, -111.83331007, -181.3218058 ,    8.38647526,
          4.58161286,  -57.3712014 ])
```

In [182…
```
X_train.columns
```

Out[182]:
```
Index(['COUNTYID', 'STATEID', 'zip_code', 'type', 'pop', 'family_mean',
       'second_mortgage', 'home_equity', 'debt', 'hs_degree', 'pct_own',
       'married', 'separated', 'divorced'],
      dtype='object')
```

In [184…
```
#Run another model at State level. There are 52 states in USA
state = train['STATEID'].unique()
state
```

Out[184]:
```
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)
```

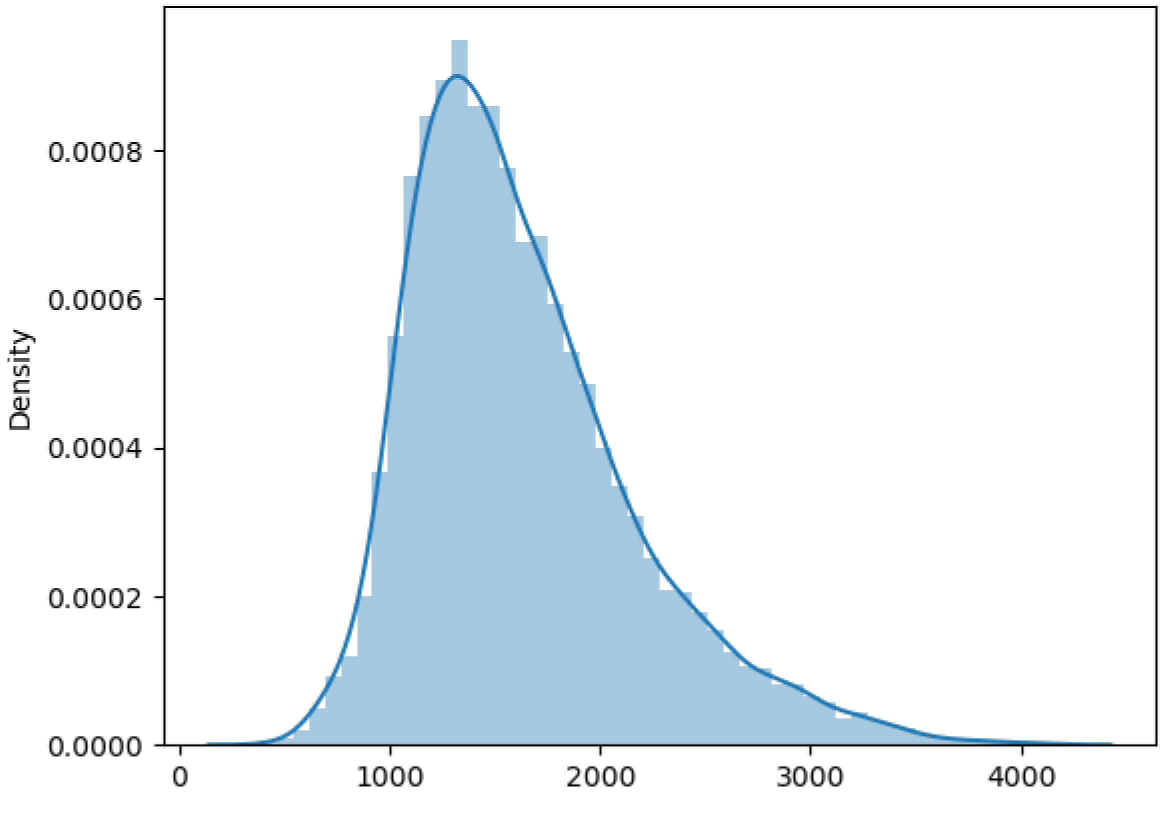
In [190…

```
for i in [11, 1, 29]:
    print("State ID-", i)

    X_train_nation = train[train['COUNTYID'] == i][feature_cols]
    y_train_nation = train[train['COUNTYID'] == i]['hc_mortgage_mean']

    X_test_nation = test[test['COUNTYID'] == i][feature_cols]
    y_test_nation = test[test['COUNTYID'] == i]['hc_mortgage_mean']

    imputer = SimpleImputer(strategy='mean')
    X_train_nation_imputed = imputer.fit_transform(X_train_nation)
    X_test_nation_imputed = imputer.transform(X_test_nation)

    y_imputer = SimpleImputer(strategy='mean')
    y_train_nation_imputed = y_imputer.fit_transform(y_train_nation.values.reshape(-1, 1
    y_test_nation_imputed = y_imputer.fit_transform(y_test_nation.values.reshape(-1, 1))

    scaler = StandardScaler()
    X_train_scaled_nation = scaler.fit_transform(X_train_nation_imputed)
    X_test_scaled_nation = scaler.transform(X_test_nation_imputed)

    lr = LinearRegression()
    lr.fit(X_train_scaled_nation, y_train_nation_imputed)

    y_pred_nation = lr.predict(X_test_scaled_nation)

    r2 = r2_score(y_test_nation_imputed, y_pred_nation)
    rmse = np.sqrt(mean_squared_error(y_test_nation_imputed, y_pred_nation))

    print("Overall R2 score of linear regression model for state,", i, ":-", r2)
    print("Overall RMSE of linear regression model for state,", i, ":-", rmse)
    print("\n")
```

```
State ID- 11
Overall R2 score of linear regression model for state, 11 :- 0.7457099573738879
Overall RMSE of linear regression model for state, 11 :- 238.46609035537793


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8102104003038398
Overall RMSE of linear regression model for state, 1 :- 309.7857339221851


State ID- 29
Overall R2 score of linear regression model for state, 29 :- 0.7407509113850169
Overall RMSE of linear regression model for state, 29 :- 254.90527128453192
```

In [192…

```
sns.distplot(y_pred)
plt.show()
```



In [ ]: