In [1]:
```python
# Import appropriate Libraries.
```

In [2]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns',None)
```

In [3]:
```python
df_train=pd.read_csv('train.csv')
df_test=pd.read_csv('test.csv')
```

In [4]:
```python
df_train.head()
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.84
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.70
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.79
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.39
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.19

In [5]:
```
df_test.head()
```

Out[5]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	48239	313
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	4210	207
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract	14871	607
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	tract	42633	606
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract	78410	361

In [6]:
```
df_train.shape
```

Out[6]: (27321, 80)

In [7]:
```
df_test.shape
```

Out[7]: (11709, 80)

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

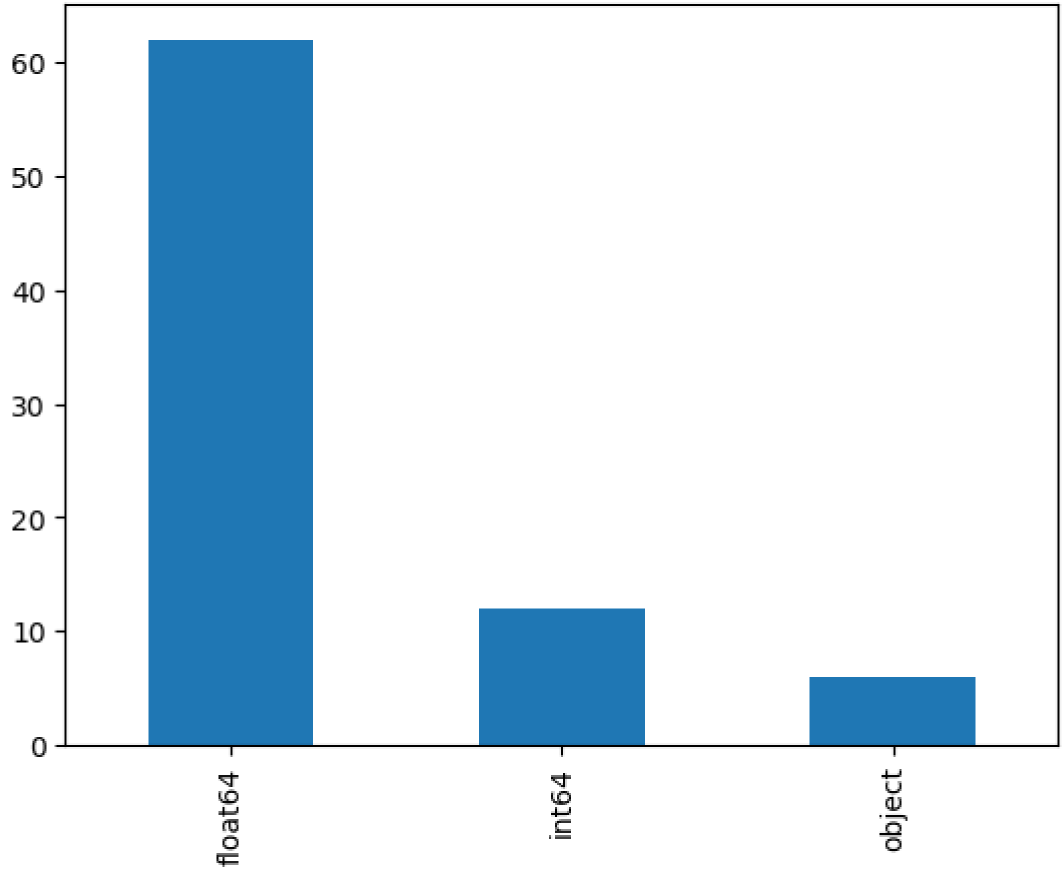
In [8]:
```
len(set(df_train['UID']).intersection(set(df_test['UID'])))
```

Out[8]: 123

In [9]:
```
df_train.dtypes
```

Out[9]:
```
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 [10]:
```
df_train.dtypes.value_counts().plot(kind='bar')
```
Out[10]: <AxesSubplot:>



In [11]:
```
df_train.describe(include='O')
```

Out[11]:

	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 [12]:
```
#This flag will help us split the data back later
```

In [13]:
```
df_train['split']='Train'
df_test['split']='Test'
```

In [14]:
```
df_combined=df_train.append(df_test, ignore_index=True)
df_combined.head()
```

Out[14]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.84
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.70
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.79
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.39
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.19

In [15]:
```
df_combined.tail()
```

Out[15]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_co
39025	238088	NaN	140	105	12	Florida	FL	Lakeland	Crystal Springs	City	tract	33810	8
39026	242811	NaN	140	31	17	Illinois	IL	Chicago	Chicago City	Village	tract	60609	7
39027	250127	NaN	140	9	25	Massachusetts	MA	Lawrence	Methuen Town City	City	tract	1841	9
39028	241096	NaN	140	27	19	Iowa	IA	Carroll	Carroll City	City	tract	51401	7
39029	287763	NaN	140	453	48	Texas	TX	Austin	Sunset Valley City	Town	tract	78745	5

In [16]:
```
df_combined.isna().sum()
```

Out[16]:
```
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 [17]:
```
# Fill rate of the variables -> (1- missing %)
```

In [18]:
```python
1-df_combined.isna().sum()/len(df_combined)
```

Out[18]:
```
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 [19]:
```python
# BLOCKID is completly missing or Null in both train and test data. So we will drop BLOCKID feature.
```

In [20]:
```python
df_combined.drop(columns=['BLOCKID'], axis=1, inplace=True)
```

In [21]:
```python
df_combined.isna().sum()/len(df_combined)*100
```

Out[21]:
```
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 [22]:

```
# Missing value greater than zero
```

In [23]:

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

```
Out[23]: ['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 [24]:

```
#If the feature have less than 8 unique value then I am consdering as categorical else it will be continuous
```

In [25]:
```
for i in null_col:
    print(i)
    if df_combined[i].nunique()>8:
        df_combined[i].fillna(df_combined[i].median(),inplace=True)
```

```
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
```

In [26]:
```
df_combined.isna().sum()/len(df_combined)*100
```

Out[26]:
```
UID              0.0
SUMLEVEL         0.0
COUNTYID         0.0
STATEID          0.0
state            0.0
                ...
married          0.0
married_snp      0.0
separated        0.0
divorced         0.0
split            0.0
Length: 80, dtype: float64
```

In [27]:
```
df_combined.shape
```
Out[27]:
```
(39030, 80)
```
In [28]:
```
# Drop duplicate observations
```
In [29]:
```
df_combined.drop_duplicates(inplace=True)
df_combined.shape
```
Out[29]:
```
(38838, 80)
```
In [30]:
```
# As we have seen above we have 123 unique UID which are common in both train and test data. so duplicate UID remo
```
In [31]:
```
df_combined.drop_duplicates(subset=['UID'],inplace=True)
df_combined.shape
```
Out[31]:
```
(38715, 80)
```
Exploratory Data Analysis (EDA):

In [32]:
```
top_2500_loc=df_train[(df_train['second_mortgage']<0.50) &
 (df_train['pct_own']>0.10) ].sort_values(by='second_mortgage', ascending=False).head(2500)
```

In [33]:

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

Out[33]:

	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 [34]:

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

Out[34]:

	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 (-82.49540 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 33.98320)
25642	Maryland	Baltimore	MD	Lochearn	39.353095	-76.733315	POINT (-76.73331 39.35310)
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 (-121.24290 37.73214)

2500 rows × 7 columns

In [35]:
```
#Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage
```

In [36]:
```
df_combined['bad_debt']=df_combined['second_mortgage'] + df_combined['home_equity'] - df_combined['home_equity_sec
df_combined.head()
```

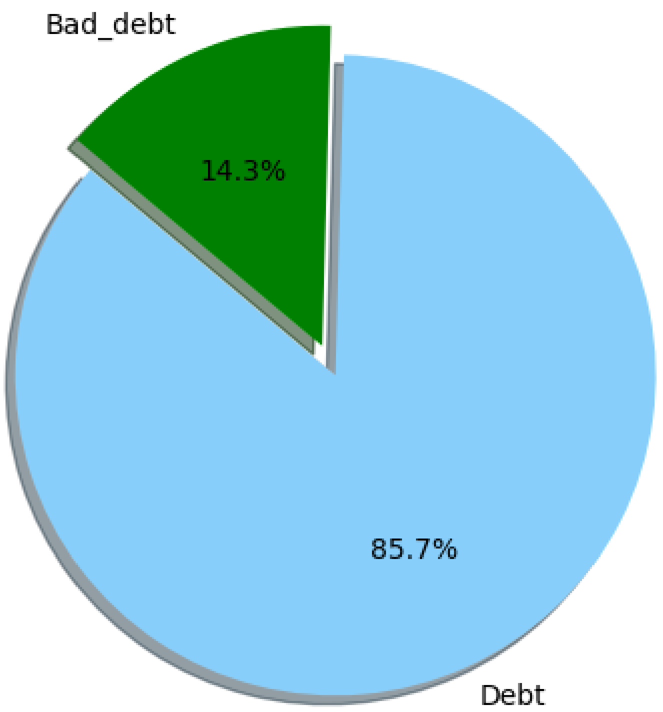
Out[36]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5

In [37]:
```
# Create pie charts to show overall debt and bad debt
```

In [38]:

```
labels='Debt', 'Bad_debt'
sizes=[df_combined['debt'].mean()*100, df_combined['bad_debt'].mean()*100]
colors=['lightskyblue', 'green']
explode=(0.1, 0) # explode 1st slice
# Plot
plt.pie(sizes,explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.axis('equal')
plt.show()
```



In [39]:

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

In [40]:

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

Out[40]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5

In [41]:

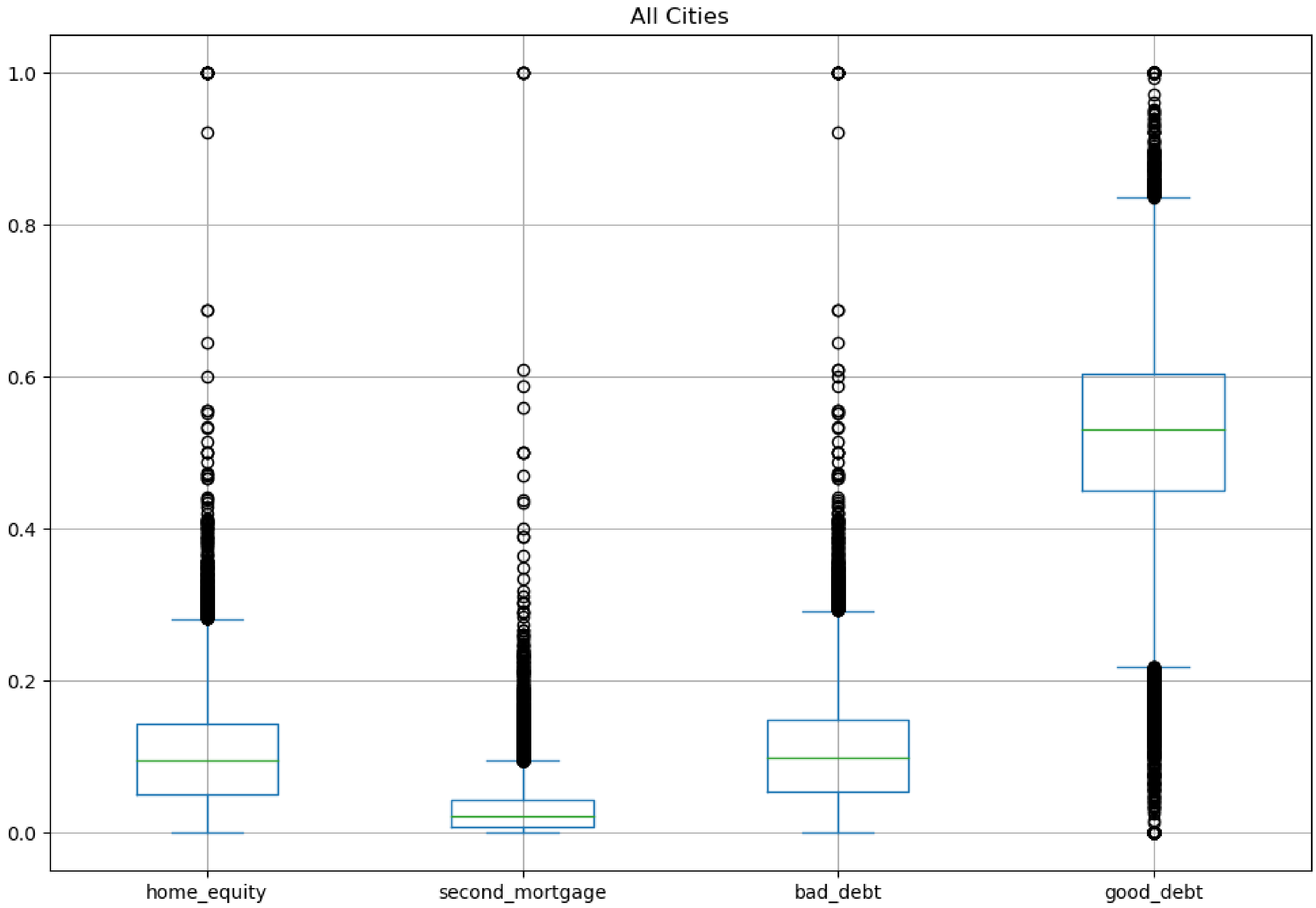
```
df_combined.columns
```

Out[41]:

```
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')
```

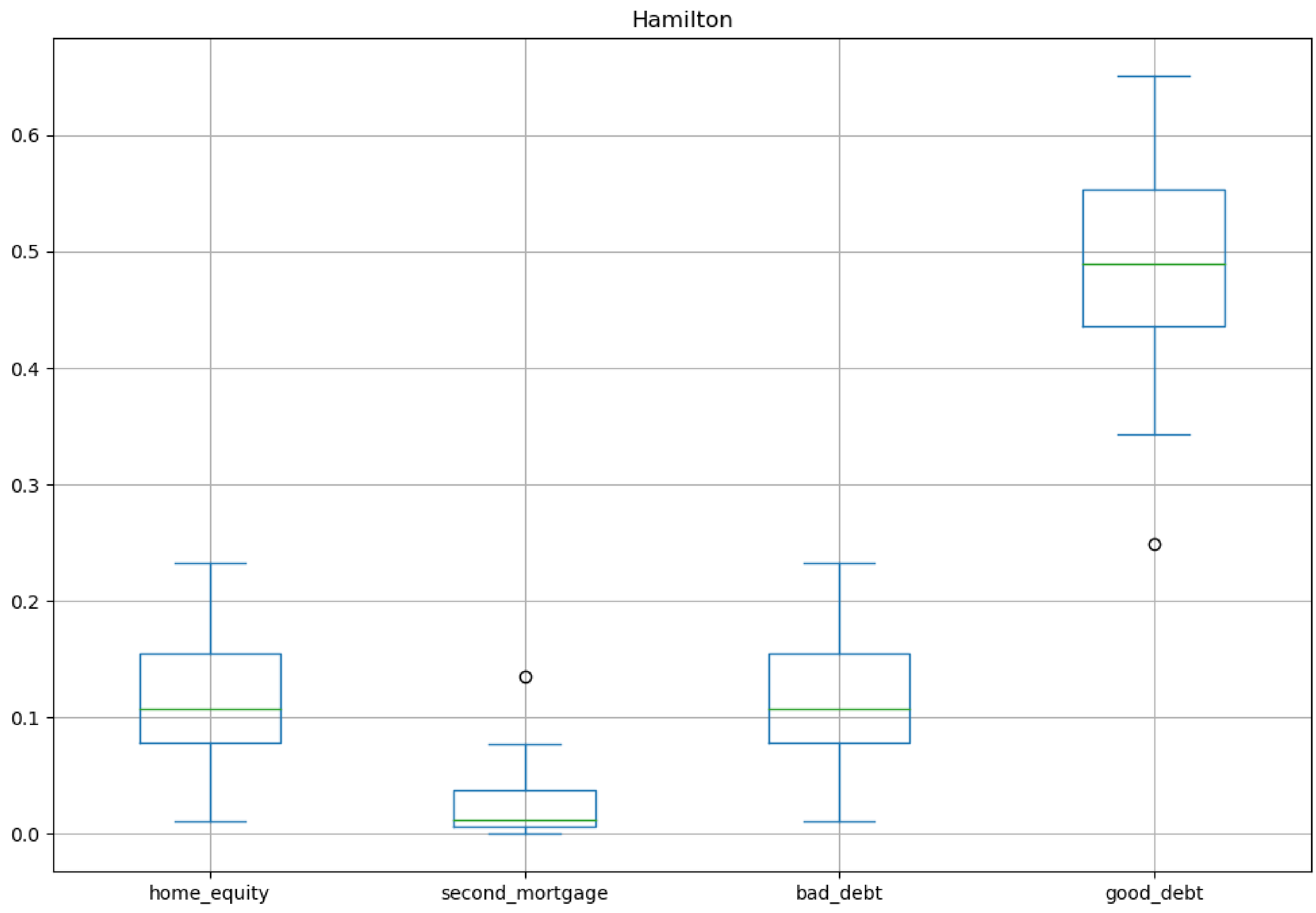
In [42]:

```
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()
```



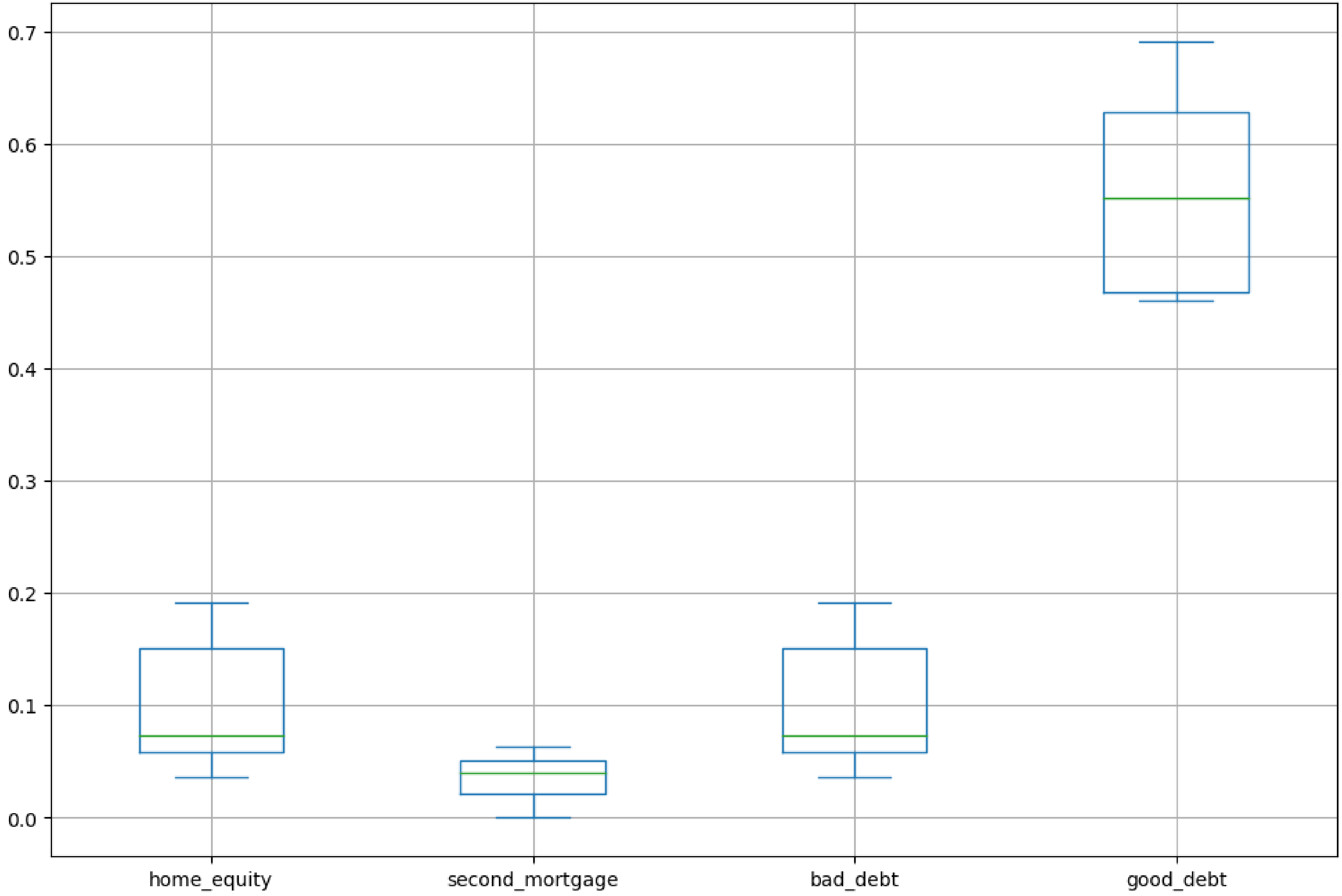
In [43]:

```
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()
```


Hamilton
0.6
0.5
0.4
0.3
0.2
0.1
0.0
home_equity
second_mortgage
bad_debt
good_debt

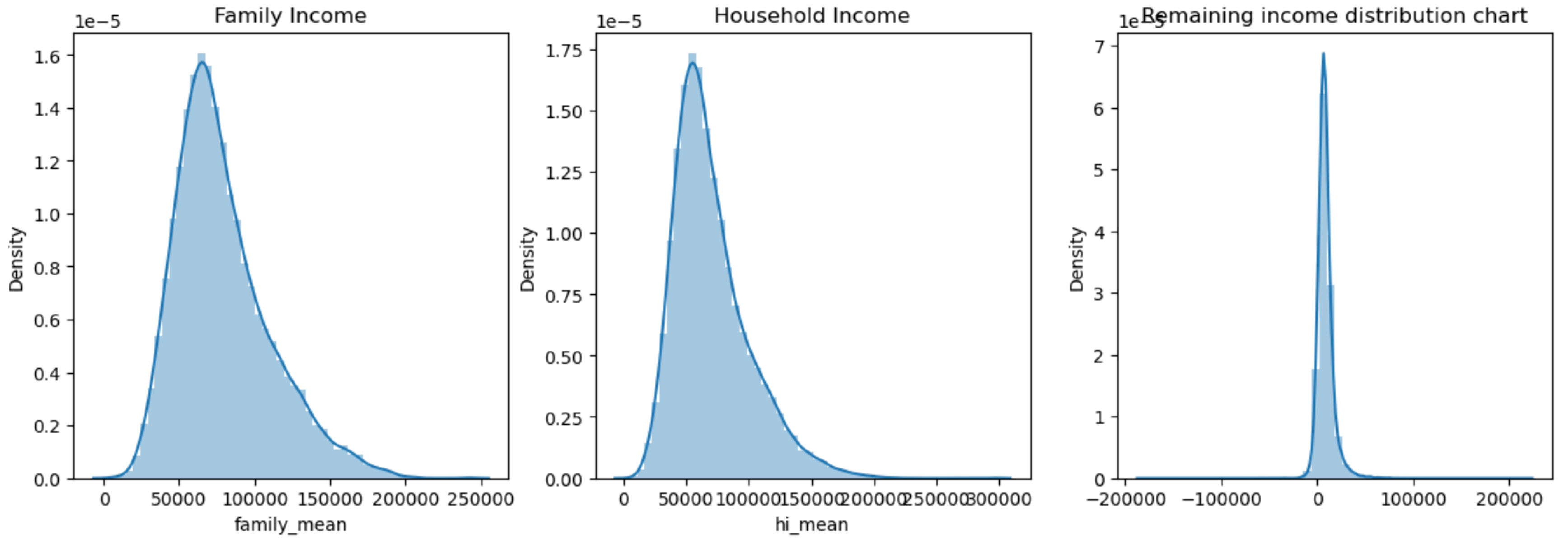
In [44]:

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



In [45]:
```
# Create a collated income distribution chart for family income, house hold income, and remaining income
```

In [46]:
```
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()
```



Project Task Exploratory Data Analysis (EDA):

In [47]:
```
df_combined['population_density']=df_combined['pop']/df_combined['ALand']
```

In [48]:
```
df_combined.head()
```

Out[48]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5[illegible]
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2[illegible]
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5[illegible]
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1[illegible]
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5[illegible]

In [49]:
```
# Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age
```

In [50]:
```

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

In [51]:
```
df_combined['median_age']=((df_combined['male_age_median']*df_combined['male_pop'])+(df_combined['female_age_media
```

In [52]:
```
df_combined.head()
```

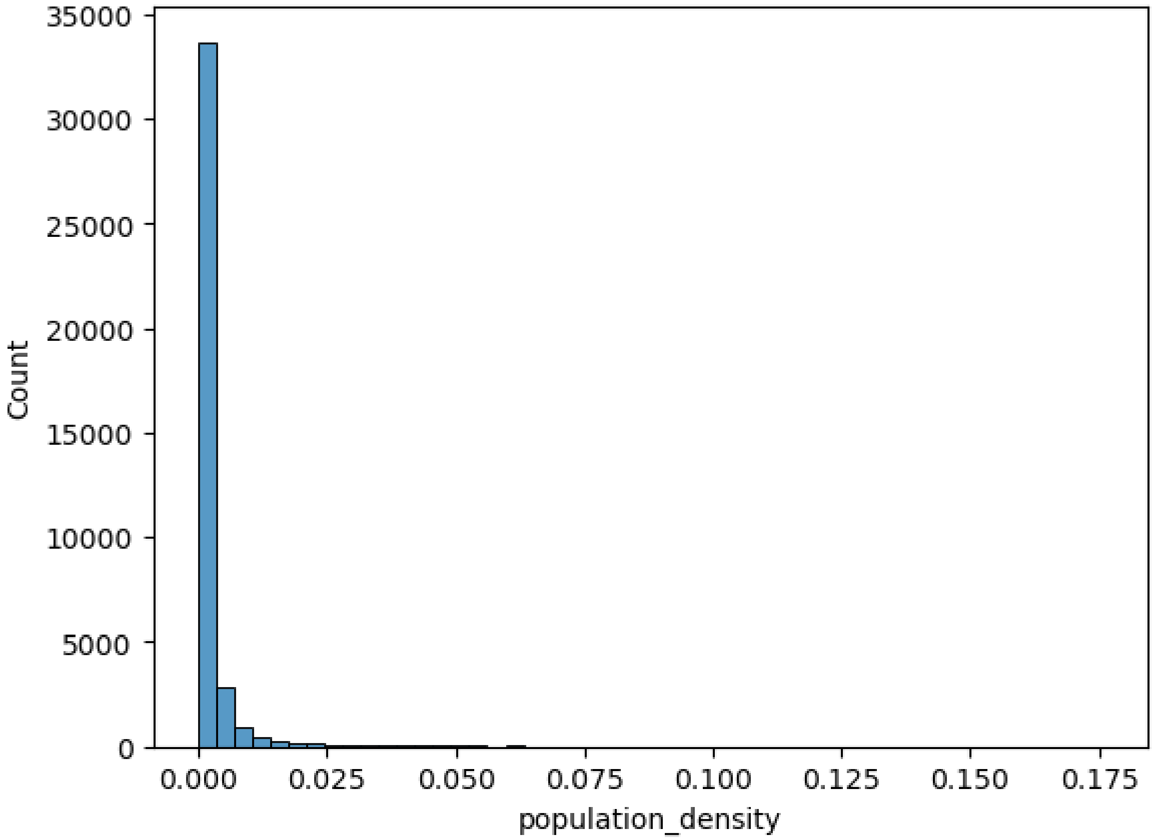
Out[52]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5

In [53]:
```
# Visualize the findings using appropriate chart type
```

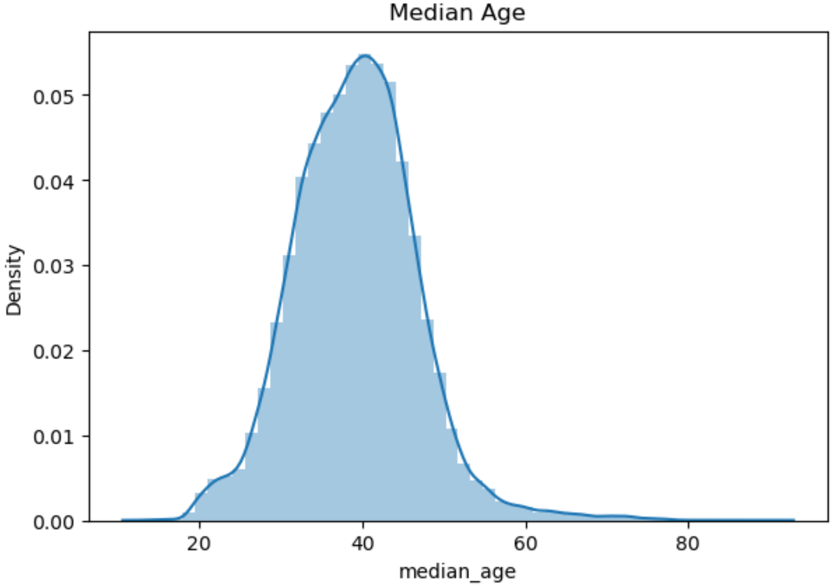
In [54]:
```
sns.histplot(df_combined['population_density'], bins=50)
```

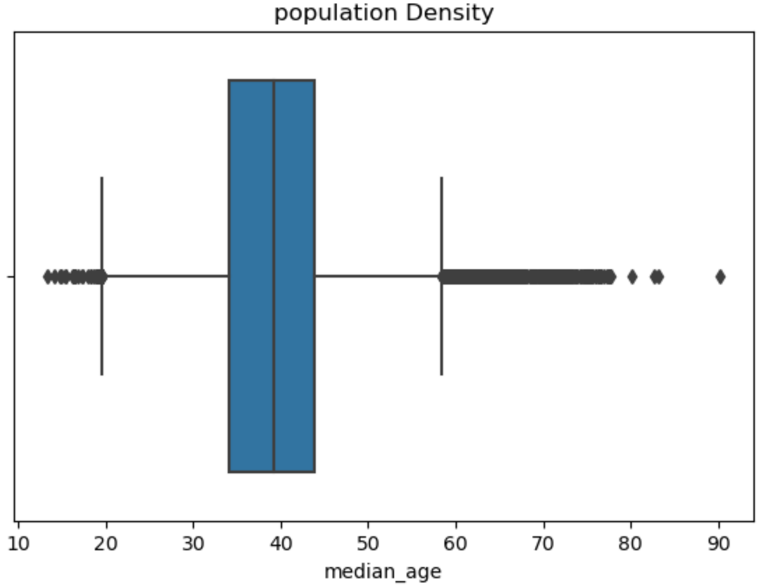
Out[54]: <AxesSubplot:xlabel='population_density', ylabel='Count'>



In [55]:

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





In [56]:

```
# Create bins for population into a new variable by selecting appropriate class interval so that the number of cat
```

In [57]:

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

Out[57]:

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

In [58]:

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

In [59]:

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

Out[59]:

	married	separated	divorced
pop_bins			
very low	38350	38350	38350
low	348	348	348
medium	12	12	12
high	4	4	4
very high	1	1	1

In [60]:
```
df_combined.groupby(by='pop_bins')[['married','separated','divorced']].agg(['mean', 'median'])
```

Out[60]:

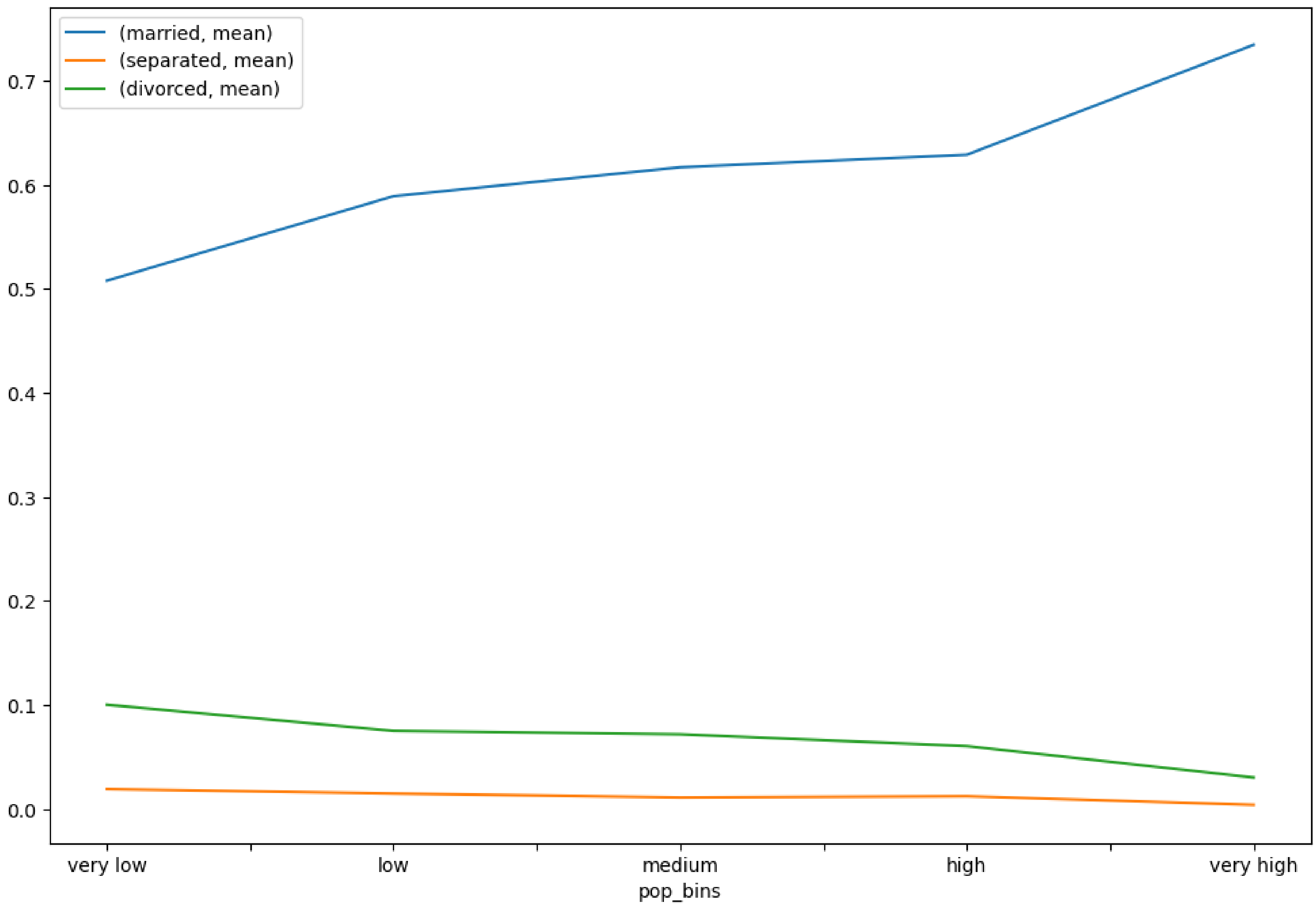
	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.508000	0.526210	0.019127	0.013580	0.100325	0.09510
low	0.589247	0.601815	0.014929	0.010255	0.075192	0.06934
medium	0.617047	0.605765	0.011203	0.007745	0.071870	0.06909
high	0.629132	0.675095	0.012372	0.007340	0.060562	0.05987
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.03036

In [61]:
```
# Visualize using appropriate chart type
```

In [62]:

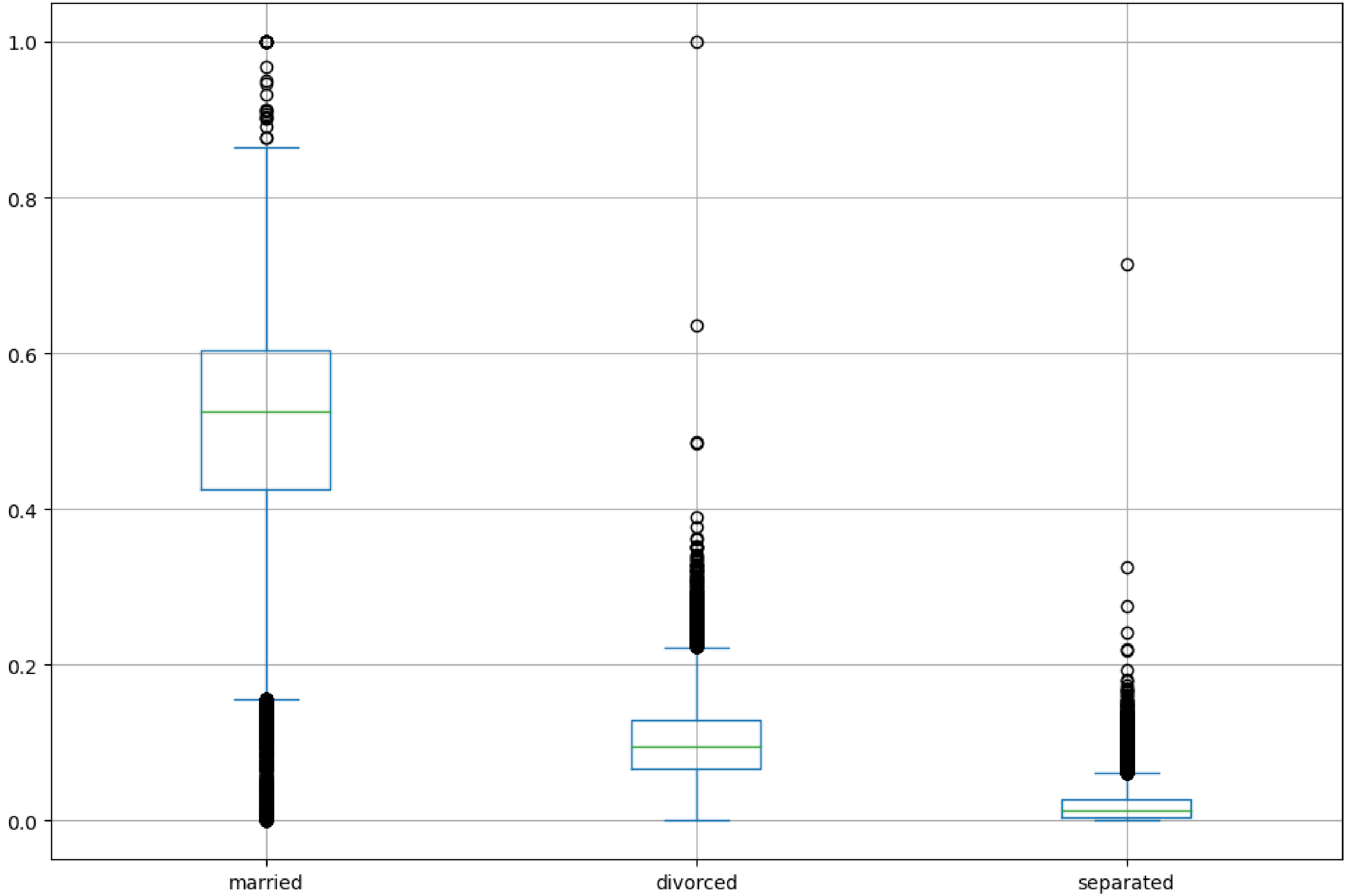
```
plt.figure(figsize=(12,8))
pop_bin_married=df_combined.groupby(by='pop_bins')[['married','separated','divorced']].agg(['mean'])
pop_bin_married.plot(figsize=(12,8))
plt.legend(loc='best')
plt.show()
```

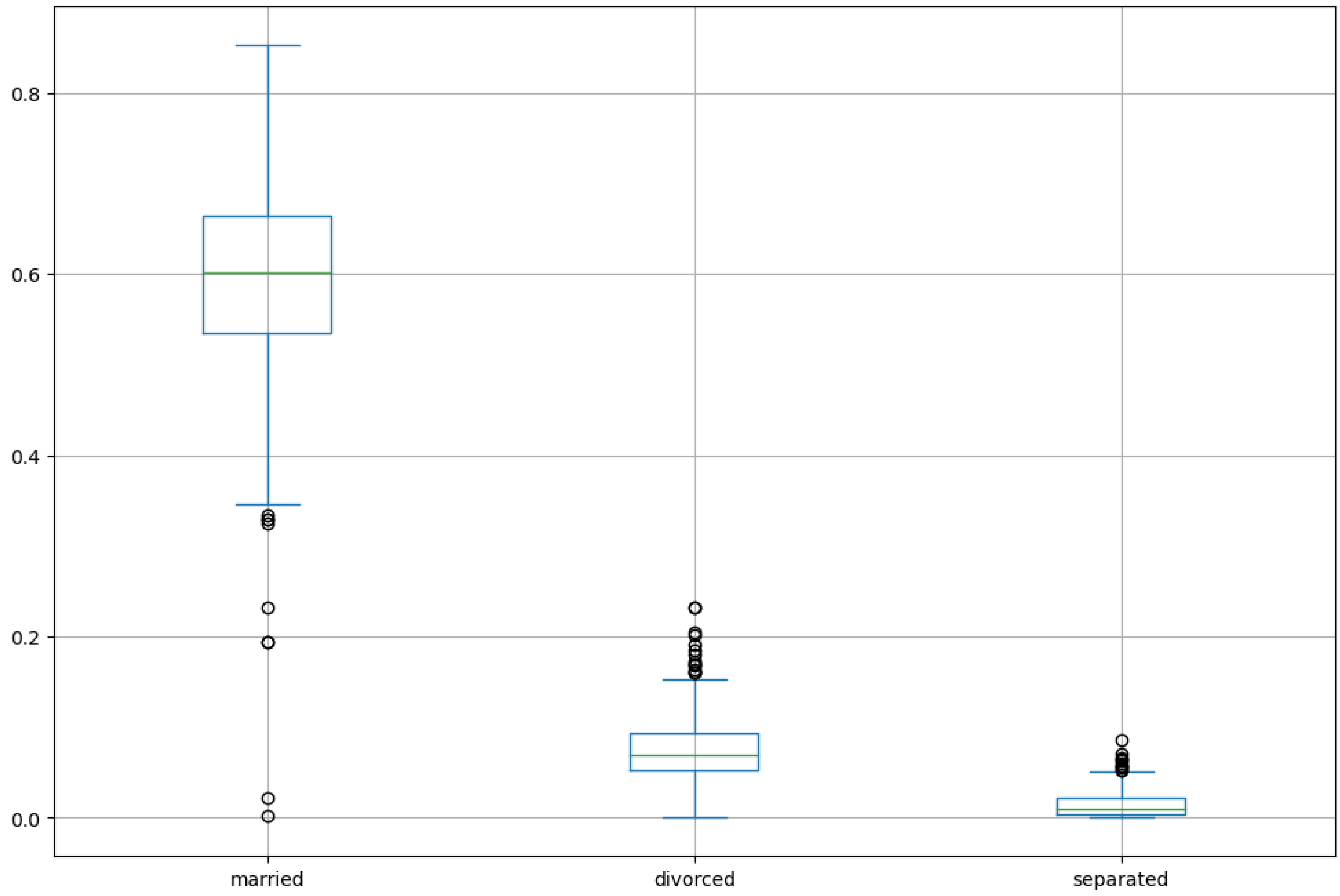
```
<Figure size 1200x800 with 0 Axes>
```

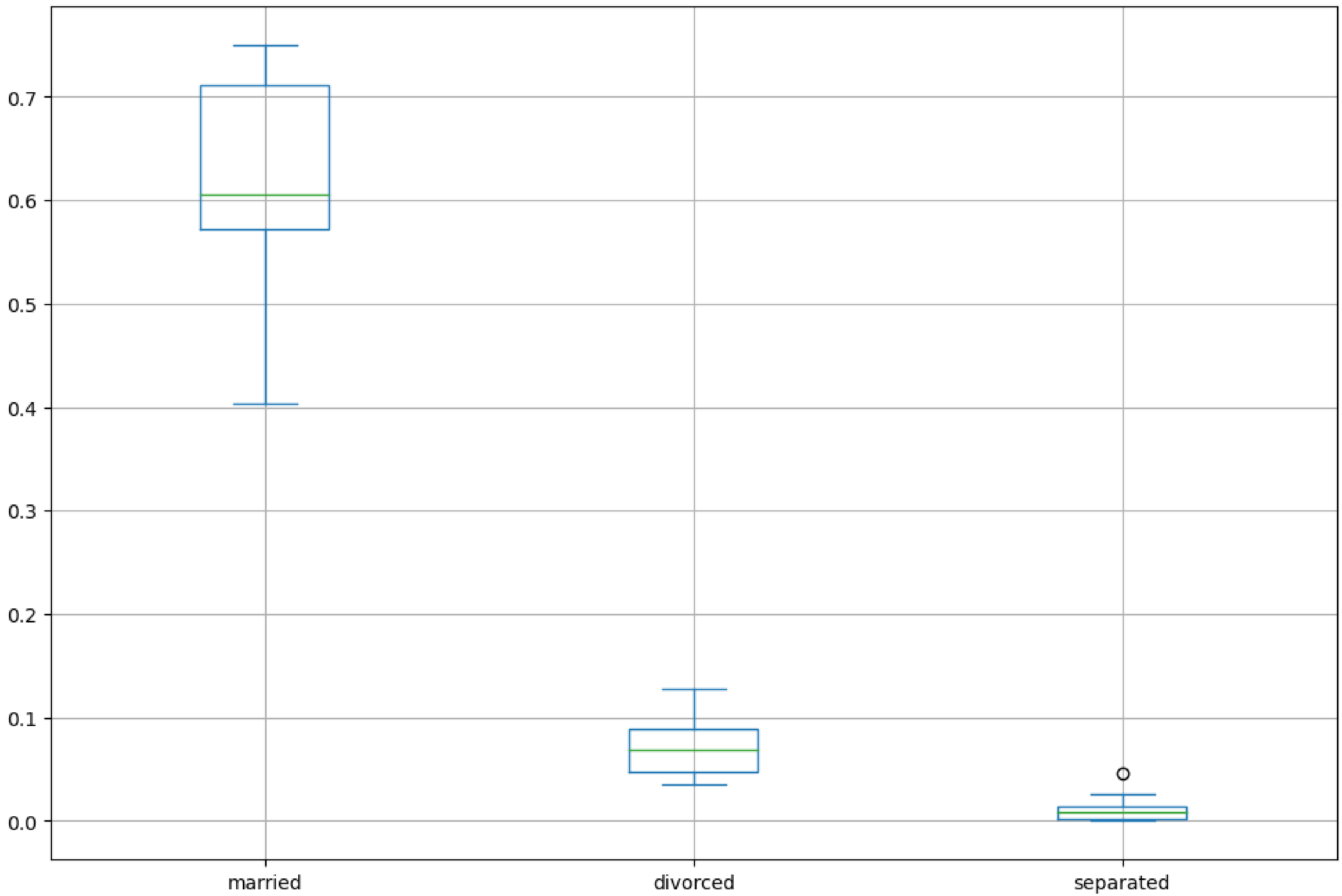

(married, mean)
(separated, mean)
(divorced, mean)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
very low
low
medium
high
very high
pop_bins

In [63]:

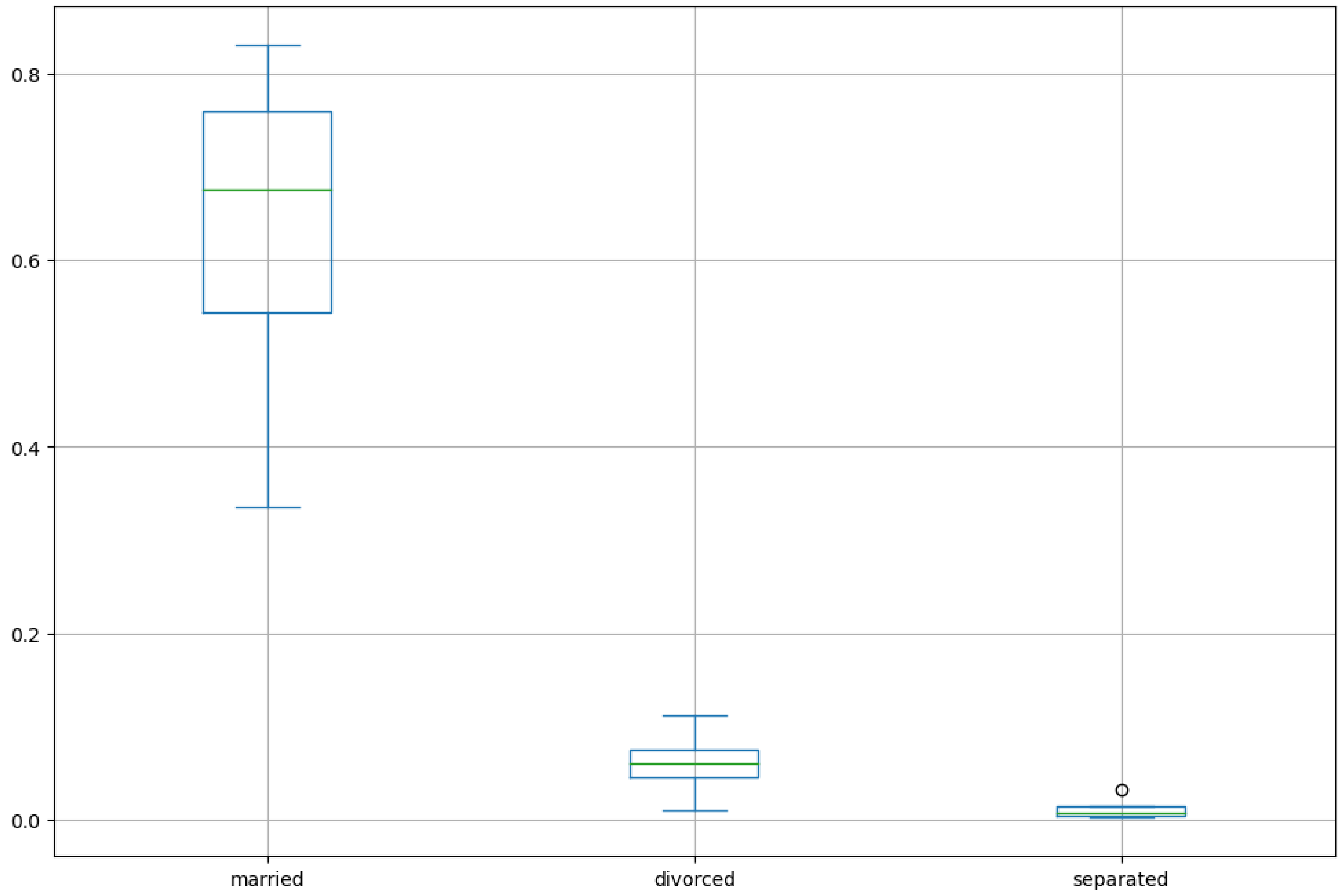
```
df_combined.groupby(by='pop_bins')[['married','divorced','separated']].plot.box(figsize=(12,8),grid='True')
plt.show()
```



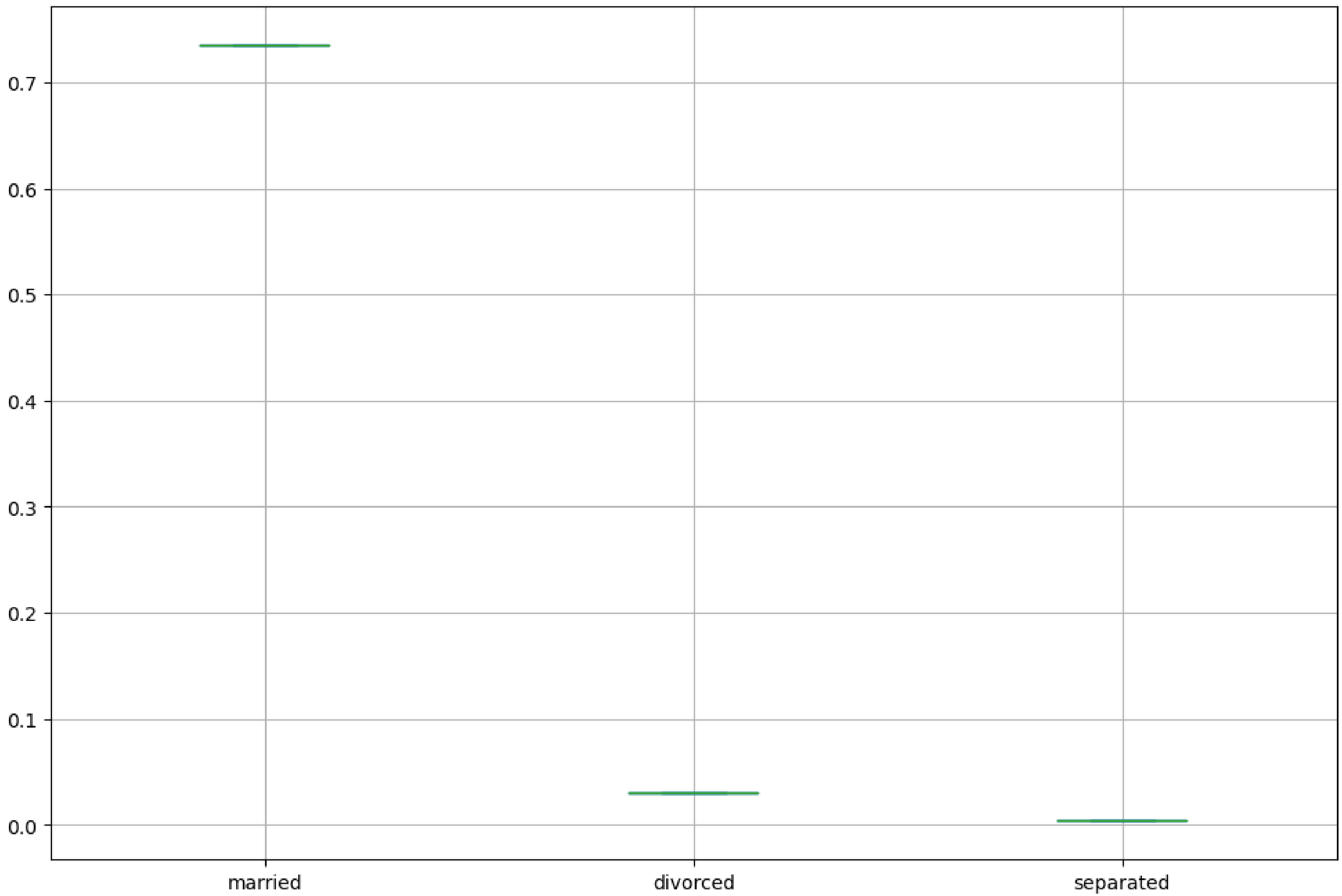

0.8
0.6
0.4
0.2
0.0
married
divorced
separated



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



In [64]:

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

In [65]:

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

Out[65]:

	mean
state	
Alabama	765.872557
Alaska	1190.093590
Arizona	1084.510940
Arkansas	716.544987
California	1466.020465

In [66]:

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

Out[66]:

	mean
state	
Alabama	65311.510962
Alaska	91911.137520
Arizona	73014.068487
Arkansas	64234.705963
California	87711.550734

In [67]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']*100
rent_perc_of_income.head(10)
```

Out[67]:
```
state
Alabama                 1.172646
Alaska                  1.294831
Arizona                 1.485345
Arkansas                1.115511
California              1.671411
Colorado                1.359697
Connecticut             1.272141
Delaware                1.311538
District of Columbia    1.357450
Florida                 1.576101
Name: mean, dtype: float64
```

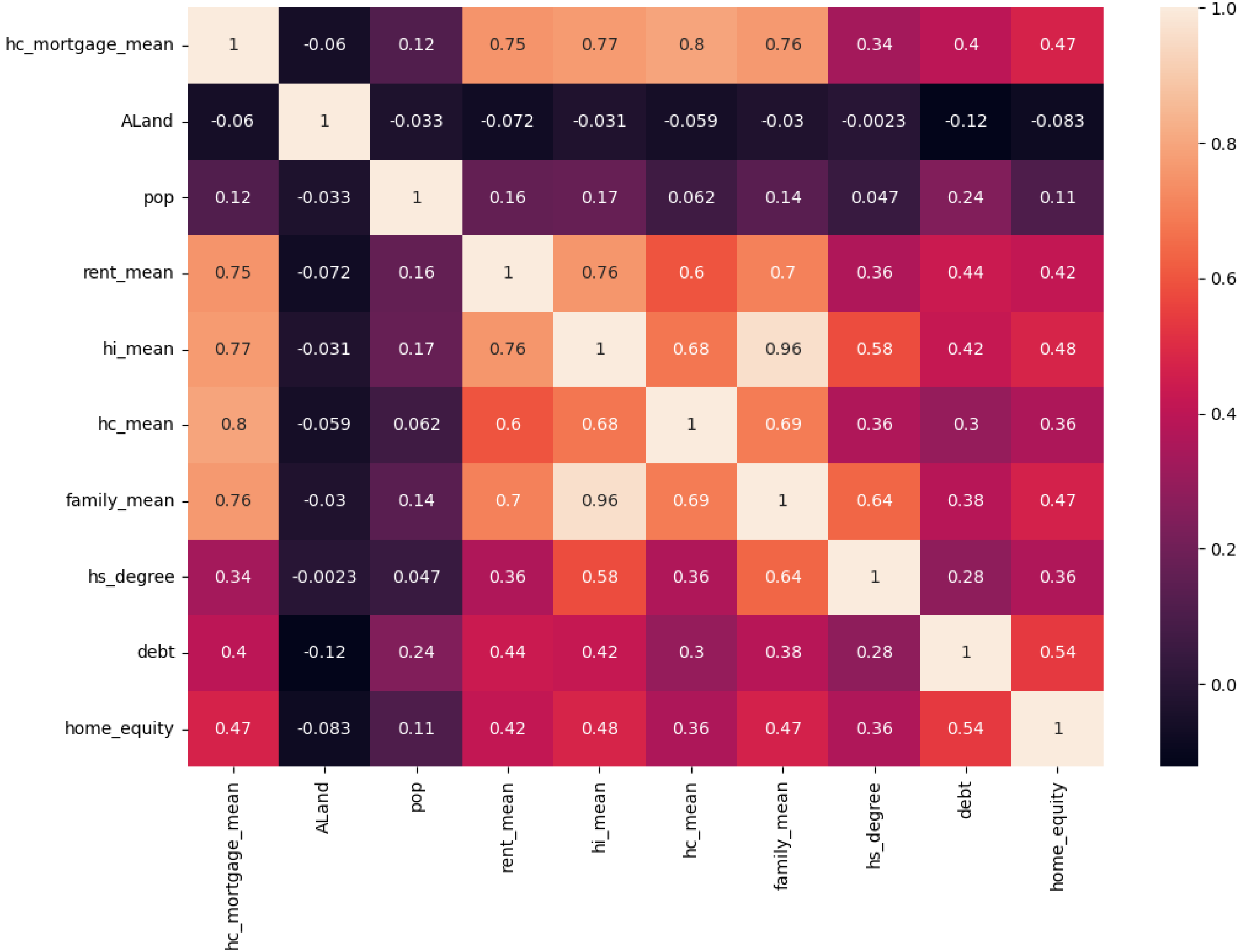
In [68]:
```
sum(df_combined['rent_mean'])/sum(df_combined['family_mean'])
```

Out[68]:
```
0.013351543786573208
```

In [69]:
```
# Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.
```

In [70]:

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


hc_mortgage_mean
ALand
pop
rent_mean
hi_mean
hc_mean
family_mean
hs_degree
debt
home_equity
1 -0.06 0.12 0.75 0.77 0.8 0.76 0.34 0.4 0.47
-0.06 1 -0.033 -0.072 -0.031 -0.059 -0.03 -0.0023 -0.12 -0.083
0.12 -0.033 1 0.16 0.17 0.062 0.14 0.047 0.24 0.11
0.75 -0.072 0.16 1 0.76 0.6 0.7 0.36 0.44 0.42
0.77 -0.031 0.17 0.76 1 0.68 0.96 0.58 0.42 0.48
0.8 -0.059 0.062 0.6 0.68 1 0.69 0.36 0.3 0.36
0.76 -0.03 0.14 0.7 0.96 0.69 1 0.64 0.38 0.47
0.34 -0.0023 0.047 0.36 0.58 0.36 0.64 1 0.28 0.36
0.4 -0.12 0.24 0.44 0.42 0.3 0.38 0.28 1 0.54
0.47 -0.083 0.11 0.42 0.48 0.36 0.47 0.36 0.54 1
1.0
0.8
0.6
0.4
0.2
0.0

rent_mean, hi_mean, hc_mean, family_mean has a good correlation with the target i.e-hc_mortagage_mean

In [71]:
```
train=df_combined[df_combined['split'] == 'Train']
test=df_combined[df_combined['split'] == 'Test']
```

In [72]:
```
train.head()
```

Out[72]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.5
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.2
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317	39.792202	-86.5
3	279653	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787	18.396103	-66.1
4	247218	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785	39.195573	-96.5

In [73]:
```
test.head()
```

Out[73]:

	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
27321	255504	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	48239	313	42.34
27322	252676	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	4210	207	44.10
27323	276314	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract	14871	607	41.94
27324	248614	140	231	21	Kentucky	KY	Monticello	Monticello City	City	tract	42633	606	36.74
27325	286865	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract	78410	361	27.88

Project Task: Week 3

Data Pre-processing:

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

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

In [74]:
```python
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [75]:
```python
df_train.describe().T
```

Out[75]:

	count	mean	std	min	25%	50%	75%	max
UID	27321.0	257331.996303	21343.859725	220342.0	238816.000000	257220.000000	275818.000000	294334.00000
BLOCKID	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SUMLEVEL	27321.0	140.000000	0.000000	140.0	140.000000	140.000000	140.000000	140.00000
COUNTYID	27321.0	85.646426	98.333097	1.0	29.000000	63.000000	109.000000	840.00000
STATEID	27321.0	28.271806	16.392846	1.0	13.000000	28.000000	42.000000	72.00000
...	...	...	...	...	...	...	...	...
pct_own	27053.0	0.640434	0.226640	0.0	0.502780	0.690840	0.817460	1.00000
married	27130.0	0.508300	0.136860	0.0	0.425102	0.526665	0.605760	1.00000
married_snp	27130.0	0.047537	0.037640	0.0	0.020810	0.038840	0.065100	0.71429
separated	27130.0	0.019089	0.020796	0.0	0.004530	0.013460	0.027488	0.71429
divorced	27130.0	0.100248	0.049055	0.0	0.065800	0.095205	0.129000	1.00000

74 rows × 8 columns

In [109]:
```python
fa = FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

...

- Variables should have significant impact on predicting Monthly mortgage and owner costs
  - Utilize all predictor variable to start with initial hypothesis
  - R square of 60 percent and above should be achieved
  - Ensure Multi-collinearity does not exist in dependent variables

In [77]:
```
train.columns
```

Out[77]:
```
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 [78]:
```
train['type'].unique()
```

Out[78]:
```
array(['City', 'Urban', 'Town', 'CDP', 'Village', 'Borough'], dtype=object)
```

In [79]:
```
type_dict={'type':{'City':1, 'Urban':2, 'Town':3, 'CDP':4, 'Village':5, 'Borough':6}}
train.replace(type_dict,inplace=True)
```

In [80]:
```
test.replace(type_dict,inplace=True)
```

In [81]:
```
train['type'].unique()
```

Out[81]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [82]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean','second_mortgage', 'home_equity', 'debt
 'pct_own', 'married','separated', 'divorced']
```

In [83]:
```
xtrain=train[feature_cols]
ytrain=train['hc_mortgage_mean']
```

In [84]:
```
xtest=test[feature_cols]
ytest=test['hc_mortgage_mean']
```

In [85]:
```
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
```

In [86]:
```
xtrain.head()
```

Out[86]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_degree	pct_own	married	separat
0	53	36	13346	1	5230	67994.14790	0.02077	0.08919	0.52963	0.89288	0.79046	0.57851	0.012
1	141	18	46616	1	2633	50670.10337	0.02222	0.04274	0.60855	0.90487	0.52483	0.34886	0.014
2	63	18	46122	1	6881	95262.51431	0.00000	0.09512	0.73484	0.94288	0.85331	0.64745	0.016
3	127	72	927	2	2700	56401.68133	0.01086	0.01086	0.52714	0.91500	0.65037	0.47257	0.020
4	161	20	66502	1	5637	54053.42396	0.05426	0.05426	0.51938	1.00000	0.13046	0.12356	0.000

In [87]:
```
xtest.head()
```

Out[87]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_degree	pct_own	married	se
27321	163	26	48239	4	3417	53802.87122	0.06443	0.07651	0.63624	0.91047	0.70252	0.28217	
27322	1	23	4210	1	3796	85642.22095	0.01175	0.14375	0.64755	0.94290	0.85128	0.64221	
27323	15	42	14871	6	3944	65694.06582	0.01316	0.06497	0.45395	0.89238	0.81897	0.59961	
27324	231	21	42633	1	2508	44156.38709	0.00995	0.01741	0.41915	0.60908	0.84609	0.56953	
27325	355	48	78410	3	6230	123527.02420	0.00000	0.03440	0.63188	0.86297	0.79077	0.57620	

In [88]:
```
sc=StandardScaler()
xtrain_scaled=sc.fit_transform(xtrain)
xtest_scaled=sc.fit_transform(xtest)
```

In [89]:
```
# Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.
```

In [90]:
```
lr =LinearRegression()
lr.fit(xtrain_scaled, ytrain)
```

Out[90]: LinearRegression()

In [91]:
```
 ypred=lr.predict(xtest_scaled)
```

R square of 60 percent and above should be achieved

In [94]:
```
r2_score(ytest,ypred)
```

Out[94]: 0.7381882934134452

In [95]:
```
mean_absolute_error(ytest,ypred)
```
Out[95]:
```
233.86965694140093
```
In [96]:
```
mean_squared_error(ytest,ypred)
```
Out[96]:
```
103818.40486733473
```
In [97]:
```
np.sqrt(mean_squared_error(ytest,ypred))
```
Out[97]:
```
322.20863561880947
```
In [98]:
```
r2_score(ytrain, lr.predict(xtrain_scaled))
```
Out[98]:
```
0.734344756627955
```
In [99]:
```
lr.coef_
```
Out[99]:
```
array([ -28.50842455,  -21.7100607 ,  -22.98370175,  -57.43101333,
         -4.78426374,  558.7402445 ,   -0.55955638,   70.89657588,
         12.81271881, -113.18431746, -176.51983734,    8.10645154,
          5.24214879,  -55.79637445])
```
In [100]:
```
xtrain.columns
```
Out[100]:
```
Index(['COUNTYID', 'STATEID', 'zip_code', 'type', 'pop', 'family_mean',
       'second_mortgage', 'home_equity', 'debt', 'hs_degree', 'pct_own',
       'married', 'separated', 'divorced'],
      dtype='object')
```

In [101]:

```
state=train['STATEID'].unique()
state
```

Out[101]:

```
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 [102]:

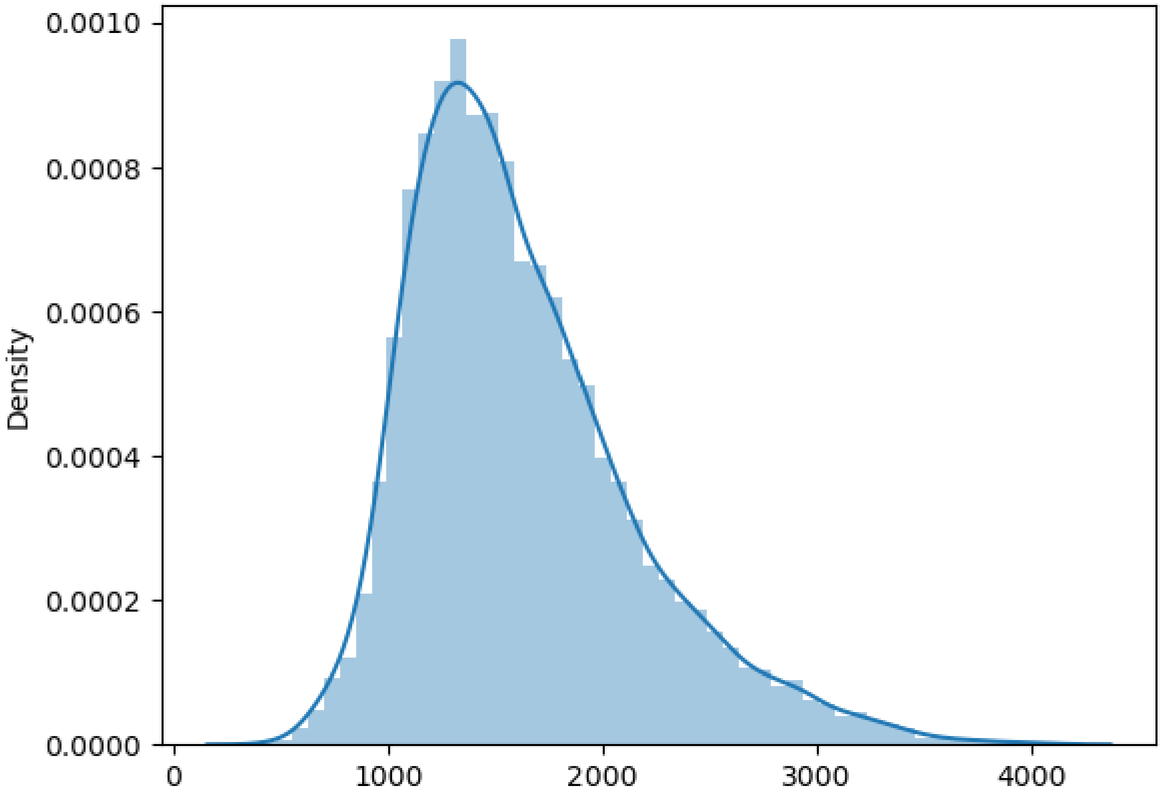
```python
for i in [11,1,29]:
 print("State ID-",i)

 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']

 X_train_scaled_nation = sc.fit_transform(X_train_nation)
 X_test_scaled_nation = sc.fit_transform(X_test_nation)

 lr.fit(X_train_scaled_nation,y_train_nation)
 y_pred_nation = lr.predict(X_test_scaled_nation)

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

```
State ID- 11
Overall R2 score of linear regression model for state, 11 :- 0.7458953509562302
Overall RMSE of linear regression model for state, 11 :- 238.52276788095128


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8086161640279985
Overall RMSE of linear regression model for state, 1 :- 311.53290720356193


State ID- 29
Overall R2 score of linear regression model for state, 29 :- 0.7090032526359475
Overall RMSE of linear regression model for state, 29 :- 270.0684126427754
```

Test if predicted variable is normally distributed

In [104]:

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



Data Reporting :

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.)

In [ ]: