Project Task: Week 1

Data Import and Preparation:

1. Import data.

In [1]:
```
import pandas as pd
#pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
```

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

# Drop the 'BLOCKID' column
df_train = df_train.drop(['BLOCKID'], axis=1)
df_test = df_test.drop(['BLOCKID'], axis=1)
```

In [3]:
```
df_train.head()
```

Out[3]:

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

In [4]:
```
df_test.head()
```

Out[4]:

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

In [5]:
```
df_train.shape
```

Out[5]:
```
(27321, 79)
```

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

Out[6]:
```
(11709, 79)
```

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

In [7]:
```
len(set(df_train['UID']).intersection(set(df_test['UID'])))
```

Out[7]:
```
123
```

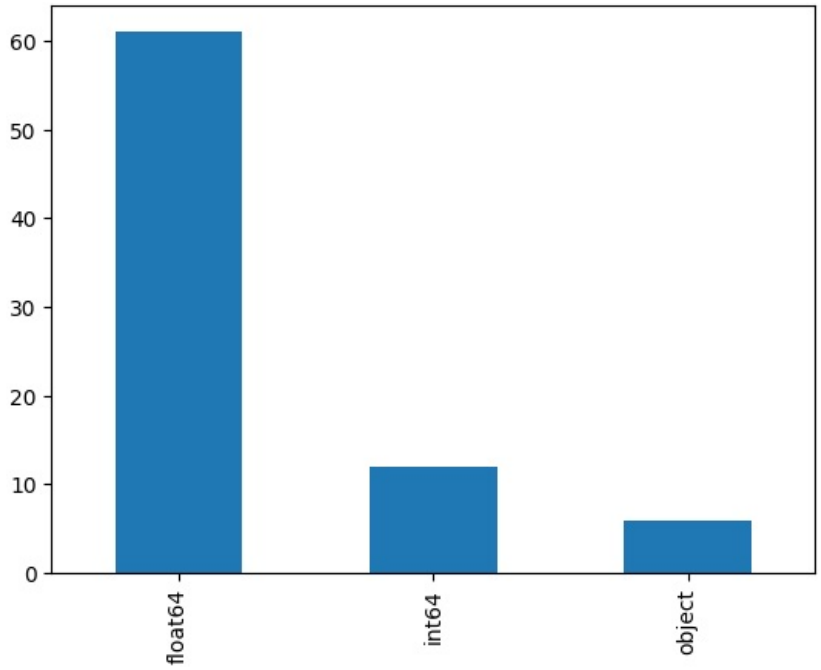
So here 123 common UID in train and test data.

In [8]:
```
df_train.dtypes
```

Out[8]:
```
UID              int64
SUMLEVEL         int64
COUNTYID         int64
STATEID          int64
state           object
                 ...
pct_own        float64
married        float64
married_snp    float64
separated      float64
divorced       float64
Length: 79, dtype: object
```

In [9]:
```
df_train.dtypes.value_counts().plot(kind='bar')
```

Out[9]:
```
<Axes: >
```



In [10]:
```
df_train.describe(include='O')
```

Out[10]:

	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

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

In [11]:
```
#This flag will help us split the data back later
df_train['split']= 'Train'
df_test['split']= 'Test'
```

In [12]:
```
df_combined = pd.concat([df_train, df_test], ignore_index=True)
df_combined.head()
```

Out[12]:

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

In [13]:
```
df_combined.tail()
```

Out[13]:

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

In [14]:
```
df_combined.shape
```

Out[14]:
```
(39030, 80)
```

In [15]:
```
df_combined.isna().sum()
```

Out[15]:
```
UID             0
SUMLEVEL        0
COUNTYID        0
STATEID         0
state           0
              ...
married       275
married_snp   275
separated     275
divorced      275
split           0
Length: 80, dtype: int64
```

In [16]:
```
# Fill rate of the variables -> (1- missing %)
1-df_combined.isna().sum()/len(df_combined)
```

Out[16]:
```
UID           1.000000
SUMLEVEL      1.000000
COUNTYID      1.000000
STATEID       1.000000
state         1.000000
                ...
married       0.992954
married_snp   0.992954
separated     0.992954
divorced      0.992954
split         1.000000
Length: 80, dtype: float64
```

In [18]:
```
df_combined.isna().sum()/len(df_combined)*100
```

Out[18]:
```
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 [19]:
```
# Missing value greater than zero
col_check=df_combined.isna().sum().to_frame().reset_index()
null_col=col_check[col_check[0]>0]['index'].tolist()
null_col
```

Out[19]:
```
['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 [20]:
```
#If the feature have less than 8 unique value then I am consdering as categorical else it will be continuous
for i in null_col:
    print(i)
    if df_combined[i].nunique()>8:      #Continuous data
        df_combined[i].fillna(df_combined[i].median(),inplace=True)    #Bcz median is not impacted by outlier
    else:df_combined[i].fillna(df_combined[i].mode()[0],inplace=True)  #Categorical data
```

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

Out[21]:
```
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 [22]:
```
df_combined.shape
```

Out[22]:
```
(39030, 80)
```

In [23]:
```
# Drop duplicate observations
df_combined.drop_duplicates(inplace=True)
df_combined.shape
```

Out[23]:
```
(38838, 80)
```

In [24]:
```
# As we have seen above we have 123 unique UID which are common in both train and test data. so duplicate UID r
df_combined.drop_duplicates(subset=['UID'],inplace=True)
df_combined.shape
```

Out[24]: (38715, 80)

Exploratory Data Analysis (EDA):

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

In [25]:
```
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 [26]:
```
top_2500_loc=top_2500_loc[['state','city','state_ab','place','lat','lng']]
top_2500_loc.head()
```

Out[26]:

	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 [27]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [28]:
```
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[28]:

	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

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

In [29]:
```
#Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage
df_combined['bad_debt'] = df_combined['second_mortgage'] + df_combined['home_equity'] - df_combined['home_equit
df_combined.head()
```

Out[29]:

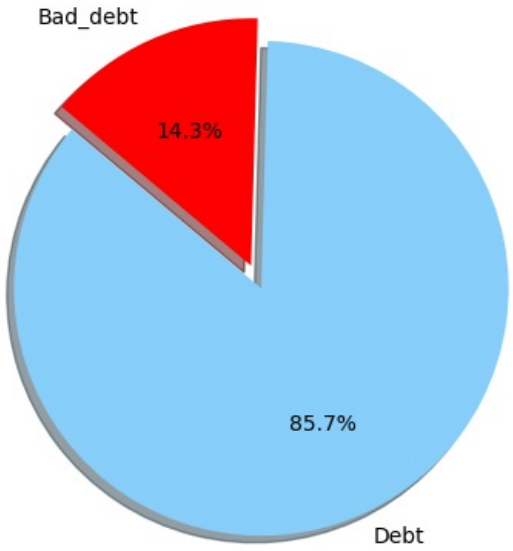
	UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	
0	267822	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.84
1	246444	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.70
2	245683	140	63	18	Indiana	IN	Danville	Danville	City	tract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

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

In [30]:
```
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)  # 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()
```



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

In [31]:
```
df_combined['good_debt']=df_combined['debt']-df_combined['bad_debt']
df_combined.head()
```

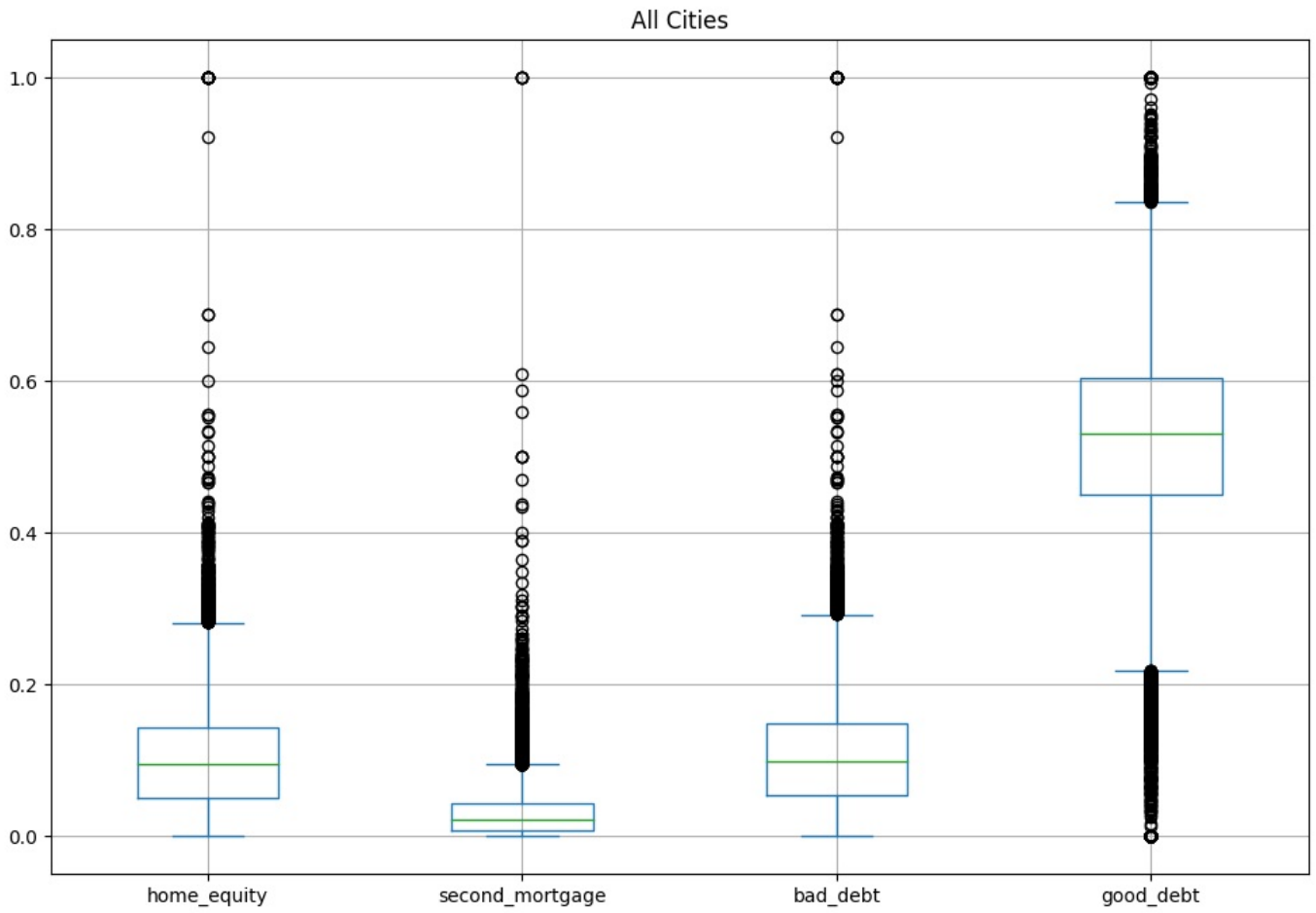
Out[31]:

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

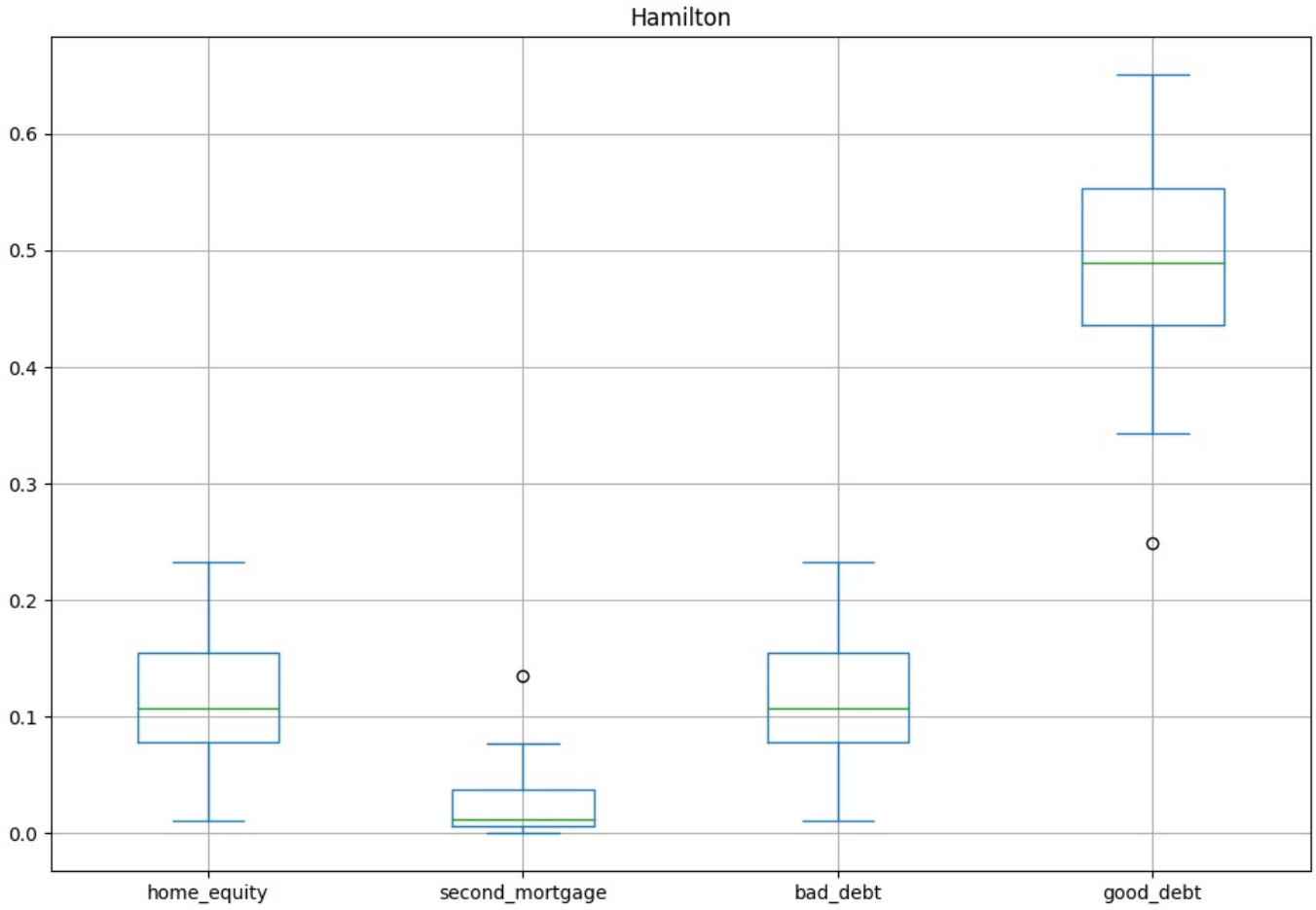
In [32]:
```
df_combined.columns
```

Out[32]:
```
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')
```

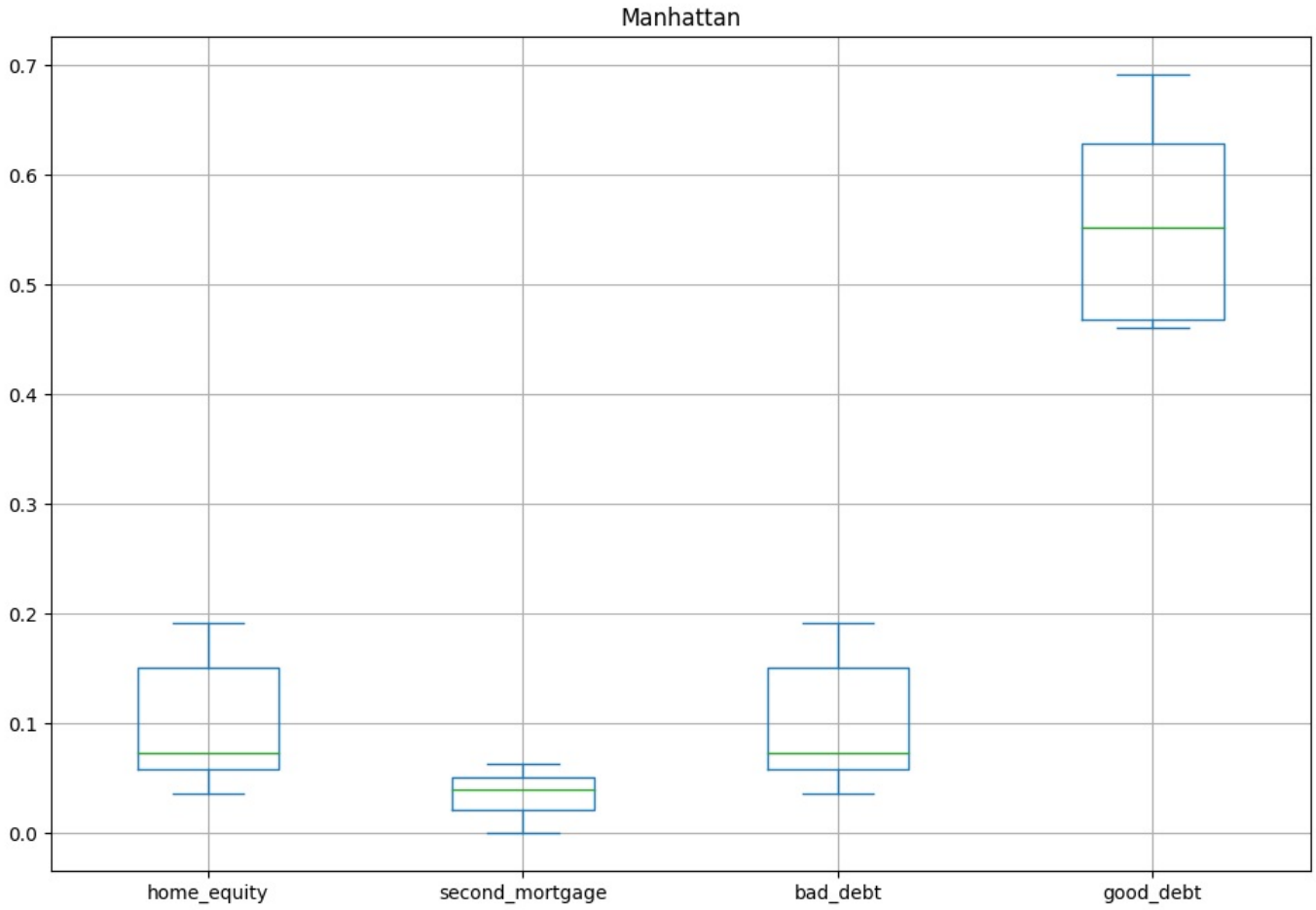
In [33]:
```
all_cities = df_combined[['home_equity','second_mortgage','bad_debt', 'good_debt']]
all_cities.plot.box(figsize=(12,8),grid=True)
plt.title('All Cities')
plt.show()
```



In [34]:
```
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()
```



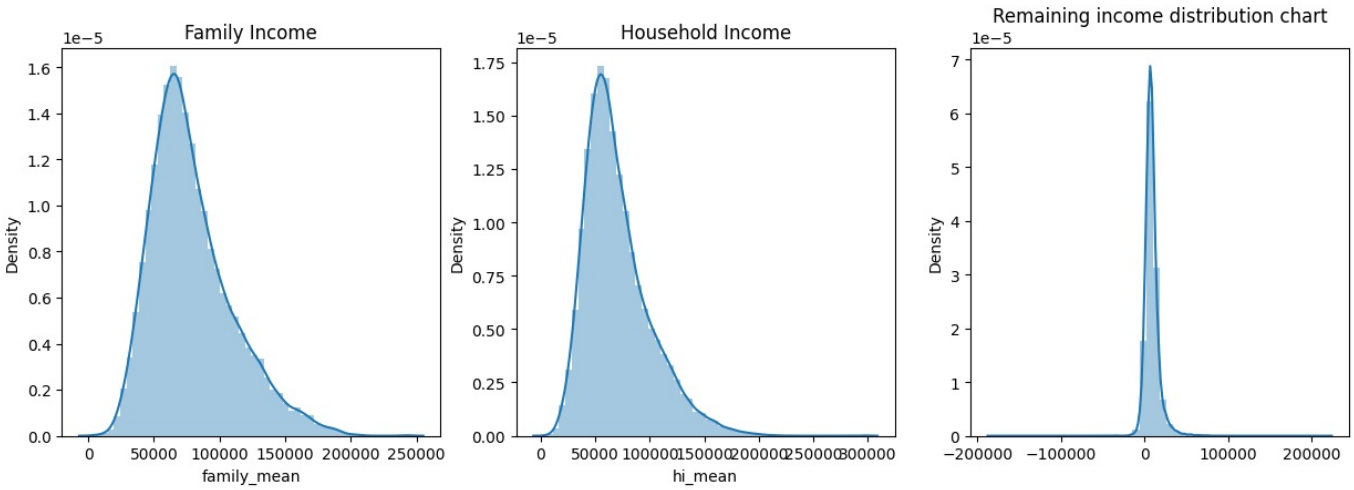
In [35]:
```
Manhattan = df_combined[df_combined['city']=='Manhattan']
Manhattan = Manhattan[['home_equity','second_mortgage','bad_debt', 'good_debt']]
Manhattan.plot.box(figsize=(12,8),grid=True)
plt.title('Manhattan')
plt.show()
```



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

In [36]:
```
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()
```



Project Task: Week 2 Exploratory Data Analysis (EDA):

1. Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):
   a. Use pop and ALand variables to create a new field called population density

In [37]:
```
df_combined['population_density'] = df_combined['pop']/df_combined['ALand']
```

In [38]:
```
df_combined.head()
```

Out[38]:

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

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

In [39]:
```
# 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
df_combined['median_age']=((df_combined['male_age_median'] * df_combined['male_pop'])+(df_combined['female_age_
```

In [40]:
```
df_combined.head()
```

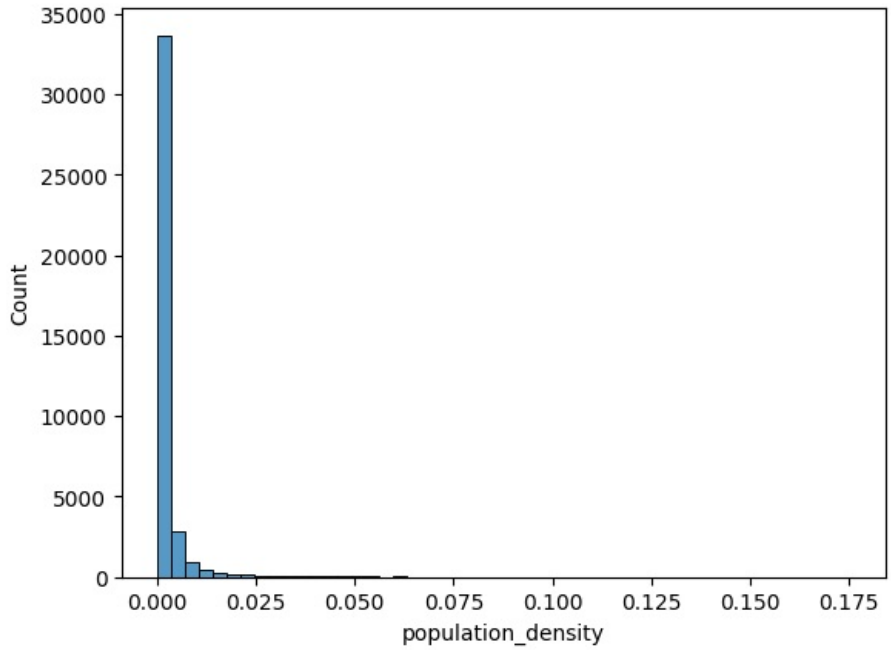
Out[40]:

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

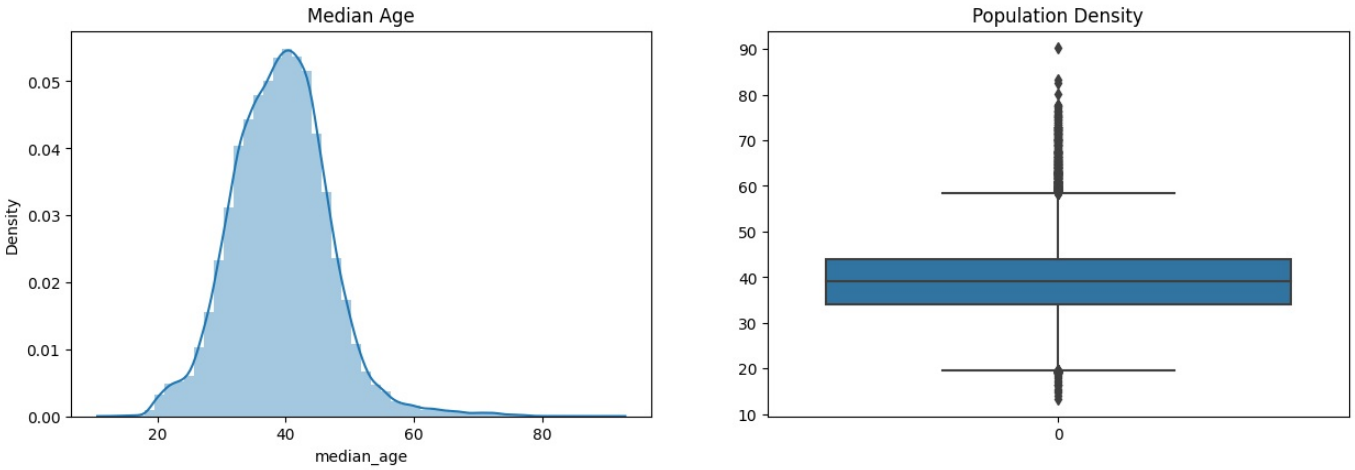
c. Visualize the findings using appropriate chart type

In [41]:
```
sns.histplot(df_combined['population_density'], bins=50)
```

Out[41]:
```
<Axes: xlabel='population_density', ylabel='Count'>
```



In [42]:
```
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()
```



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.

In [43]:
```
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[43]:
```
pop_bins
very low     38350
low            348
medium          12
high             4
very high        1
Name: count, dtype: int64
```

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

In [44]:
```
df_combined.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

Out[44]:

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

In [45]:
```
df_combined.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"])
```

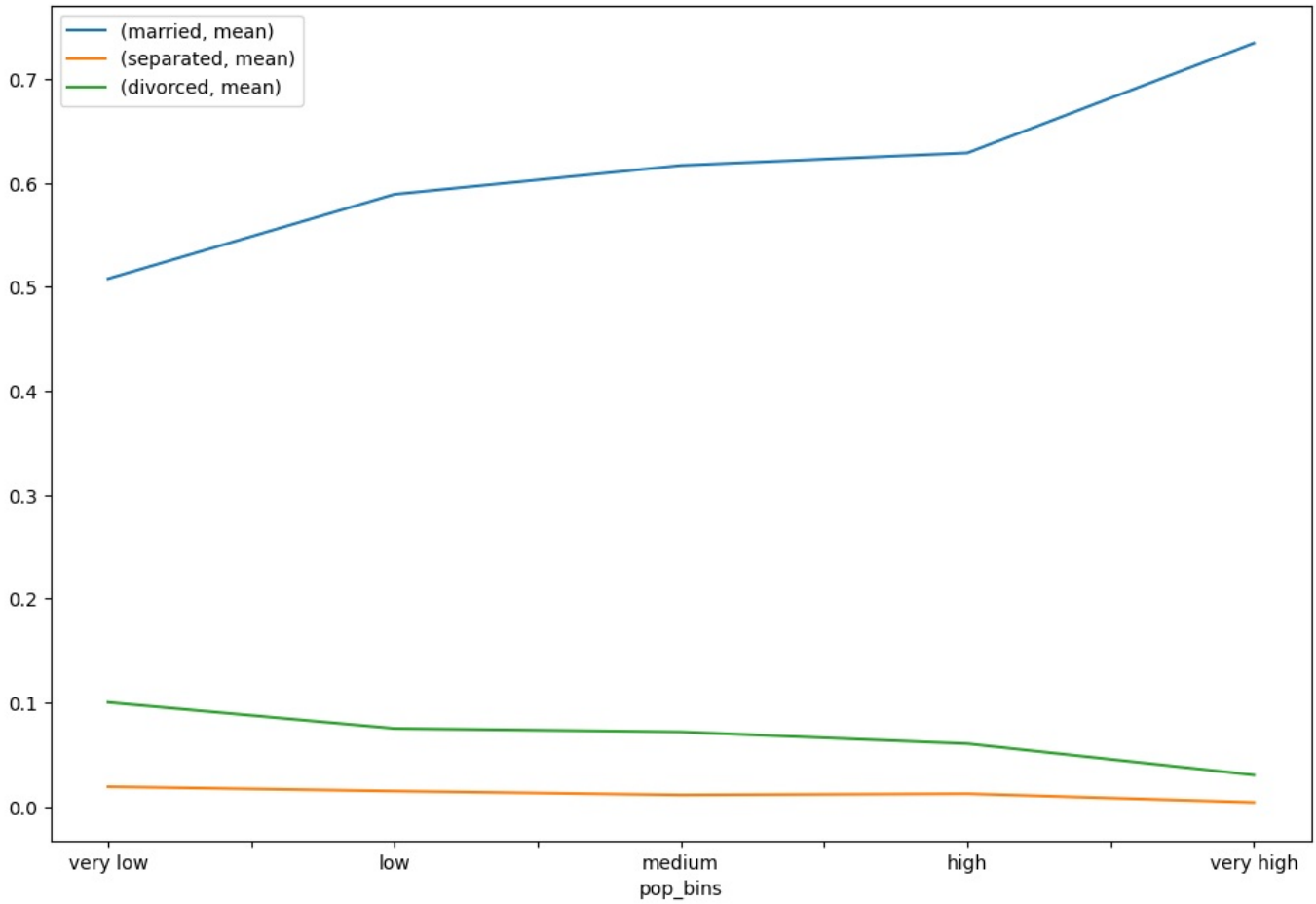
Out[45]:

	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

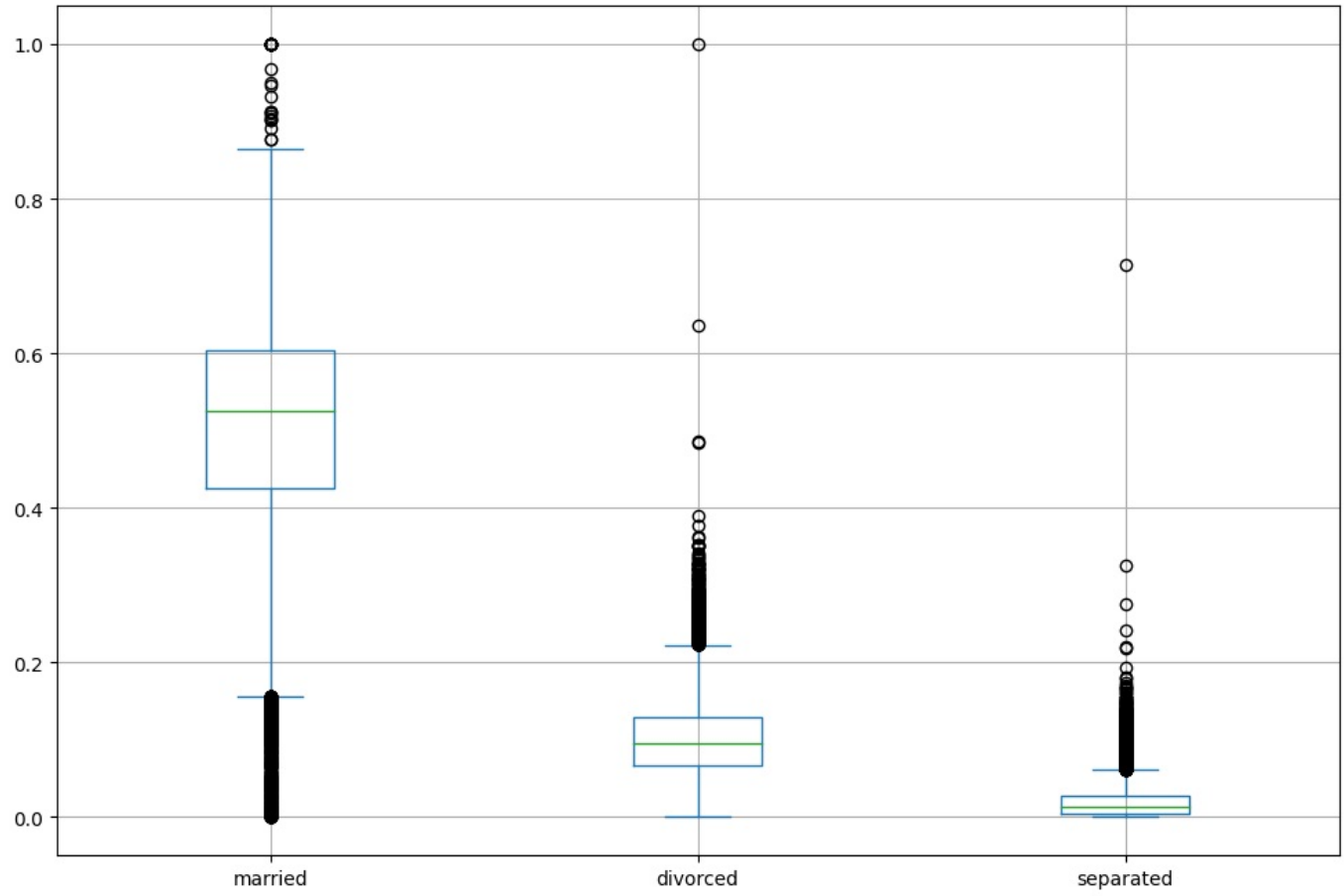
b. Visualize using appropriate chart type

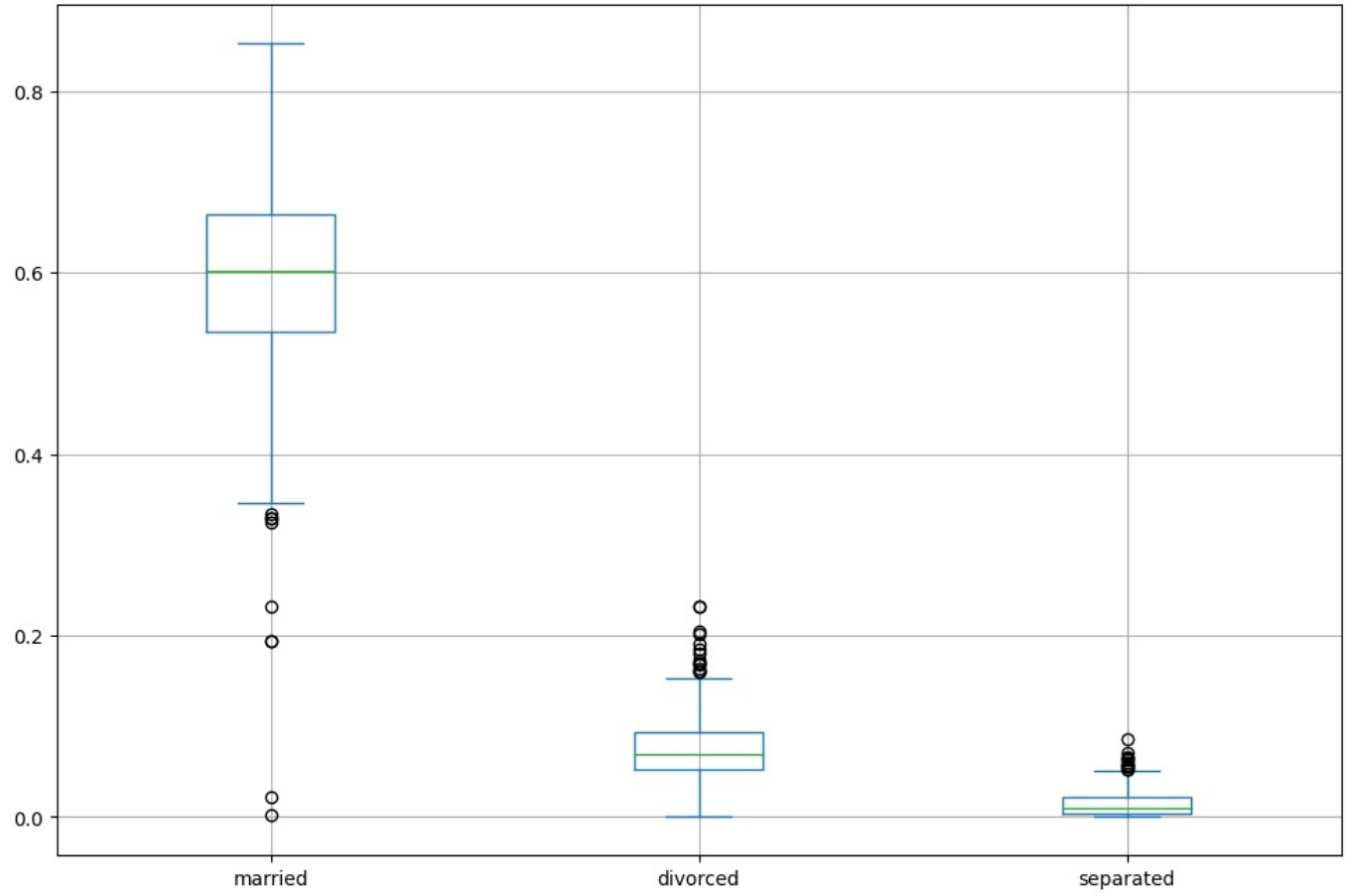
In [46]:
```
plt.figure(figsize=(12,8))
pop_bin_married=df_combined.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean"])
pop_bin_married.plot(figsize=(12,8))
plt.legend(loc='best')
plt.show()
```

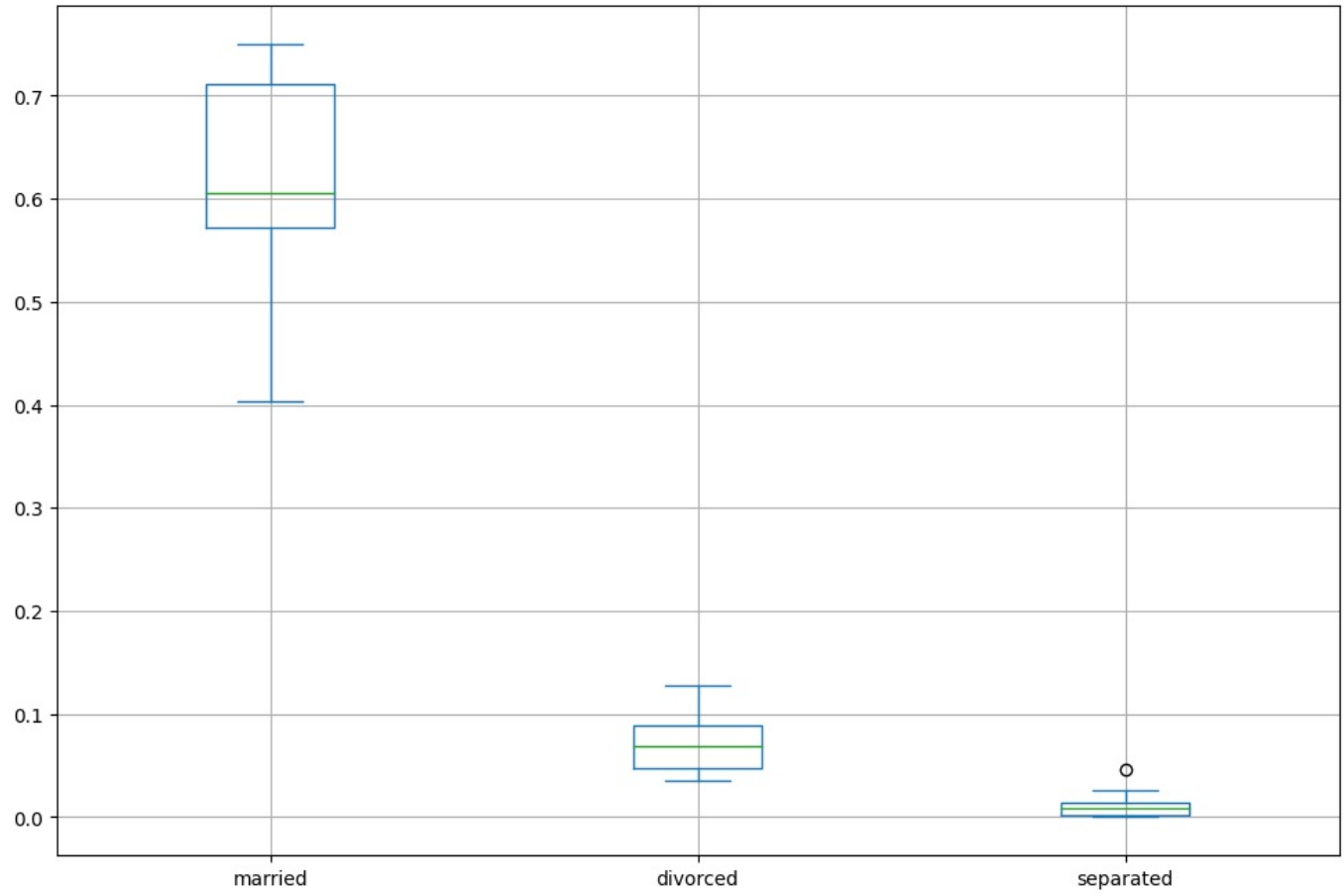
```
<Figure size 1200x800 with 0 Axes>
```

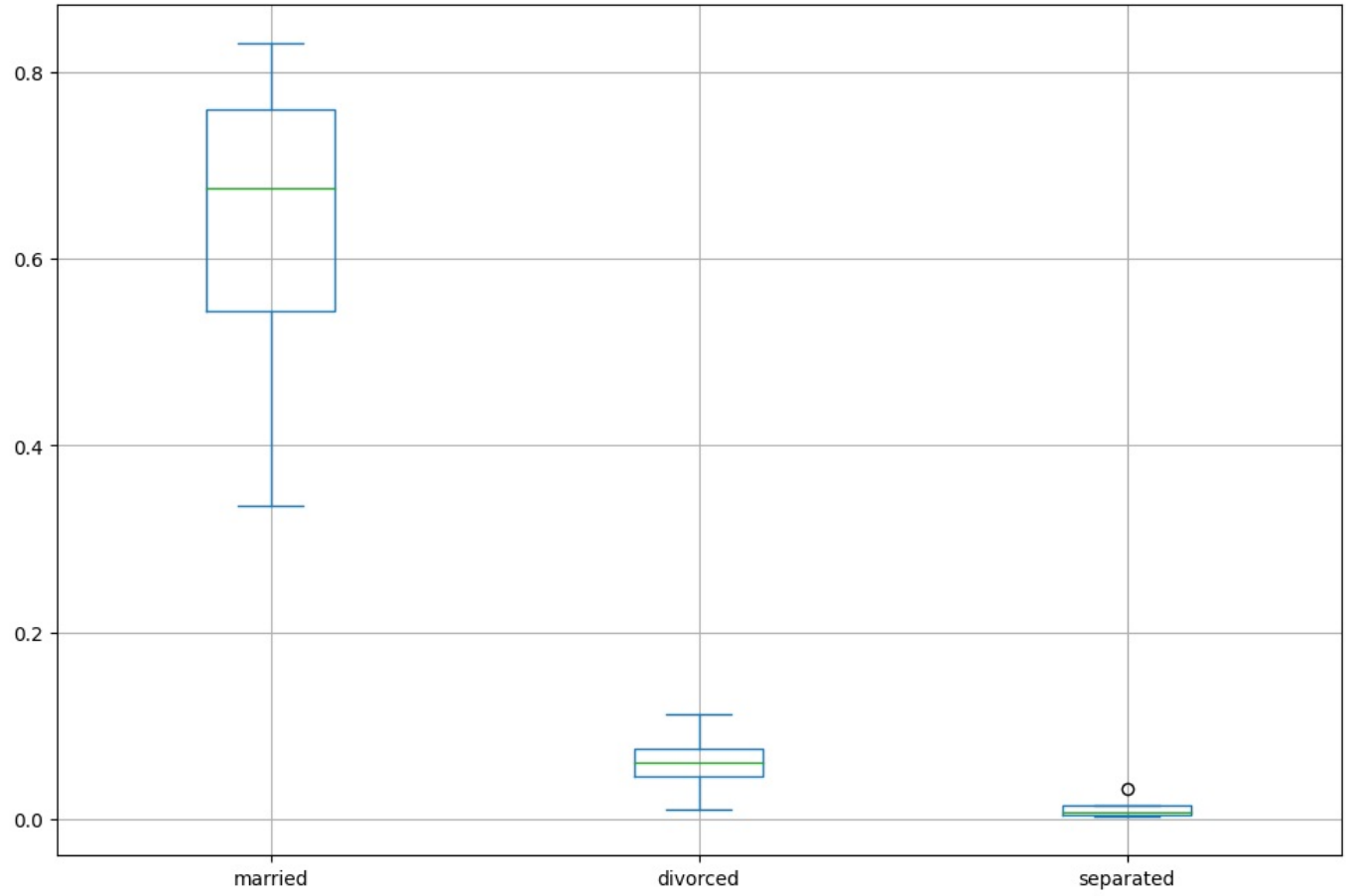


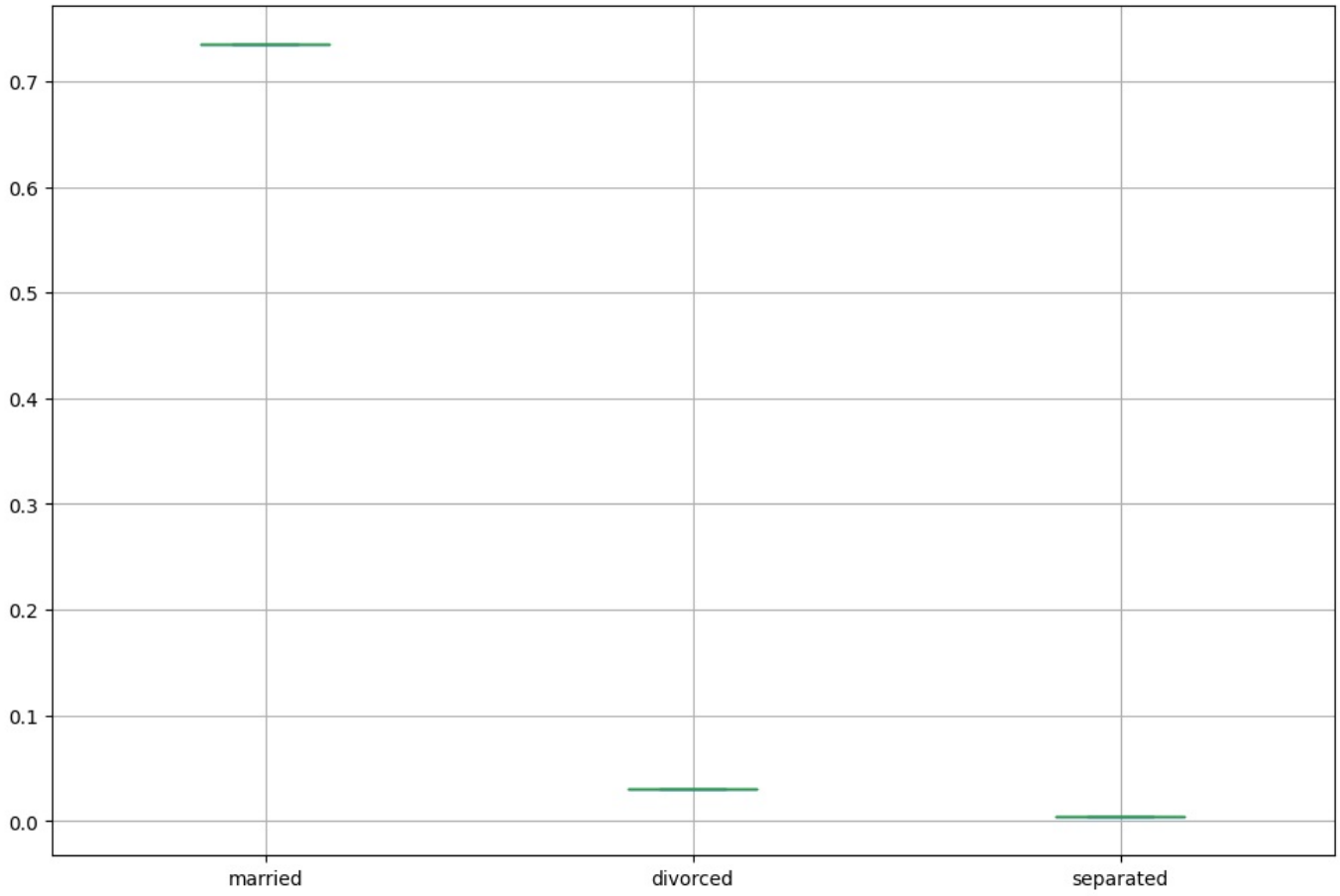
In [47]:
```
df_combined.groupby(by='pop_bins')[['married','divorced', 'separated']].plot.box(figsize=(12,8),grid='True')
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
married
divorced
separated


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



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

In [48]:
```
rent_state_mean = df_combined.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

Out[48]:

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

In [49]:
```
income_state_mean=df_combined.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

Out[49]:

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

In [50]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']*100
rent_perc_of_income.head(10)
```

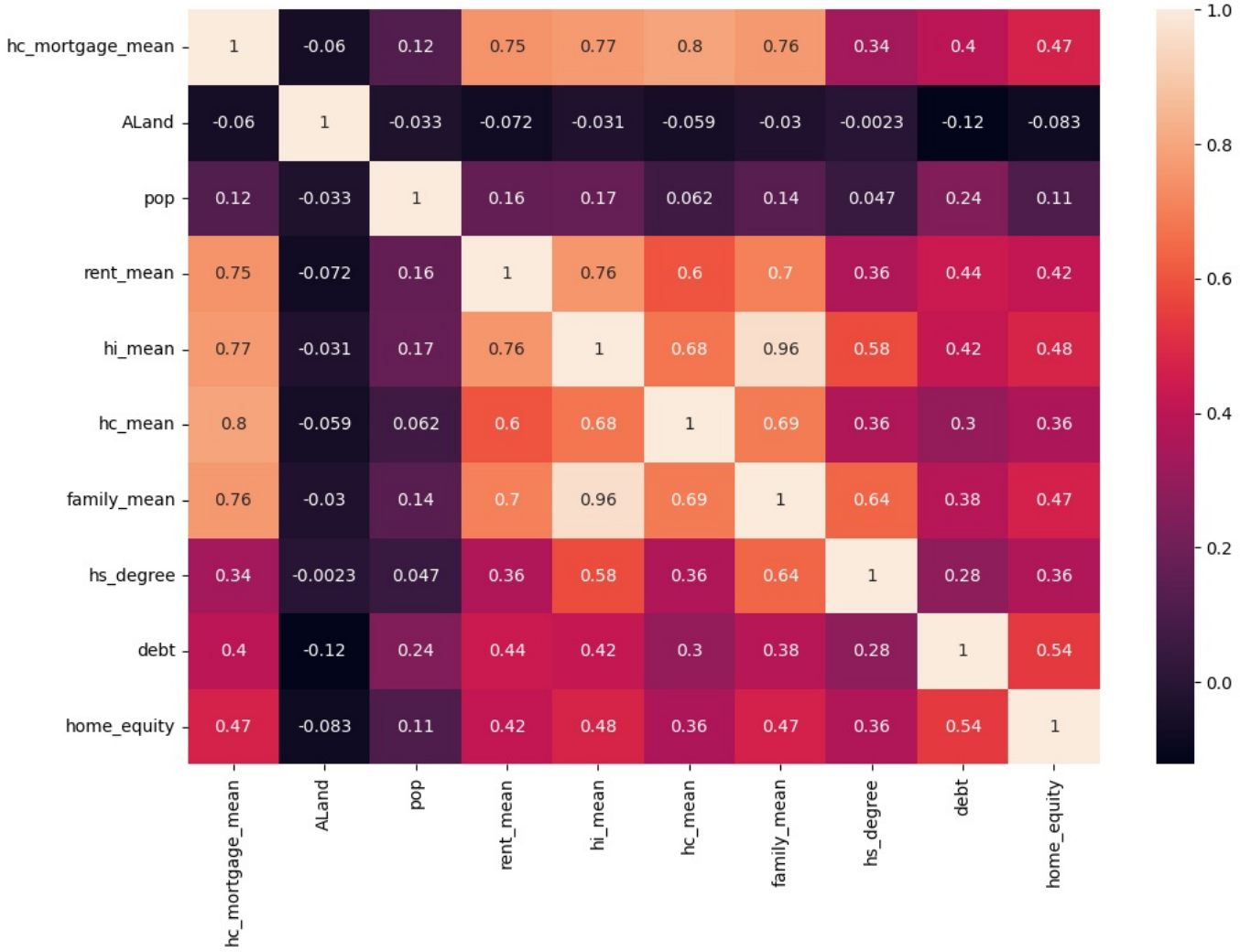
Out[50]:
```
state
Alabama                 1.172646
Alaska                  1.294831
Arizona                 1.485345
Arkansas                1.115511
California              1.671411
Colorado                1.359697
Connecticut             1.272141
Delaware                1.311538
District of Columbia    1.357450
Florida                 1.576101
Name: mean, dtype: float64
```

In [51]:
```
sum(df_combined['rent_mean'])/sum(df_combined['family_mean'])
```

Out[51]:
```
0.013351543786573208
```

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

In [52]:
```
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()
```



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

In [53]:
```
train = df_combined[df_combined['split'] == 'Train']
test = df_combined[df_combined['split'] == 'Test']
```

In [54]:
```
train.head()
```

Out[54]:

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

In [55]:
```
test.head()
```

Out[55]:

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

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 [56]:
```
import numpy as np
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

In [57]:
```
df_train.describe().T
```

Out[57]:

	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
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
zip_code	27321.0	50081.999524	29558.115660	602.0	26554.000000	47715.000000	77093.000000	99925.00000
...	...	...	...	...	...	...	...	...
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

73 rows × 8 columns

In [58]:
```
print(df_train.isnull().any())
numeric_cols = df_train.select_dtypes(include=[np.number])
print(np.isinf(numeric_cols).any())
```

```
UID            False
SUMLEVEL       False
COUNTYID       False
STATEID        False
state          False
               ...
married         True
married_snp     True
separated       True
divorced        True
split          False
Length: 80, dtype: bool
UID            False
SUMLEVEL       False
COUNTYID       False
STATEID        False
zip_code       False
               ...
pct_own        False
married        False
married_snp    False
separated      False
divorced       False
Length: 73, dtype: bool
```

In [59]:
```
# Replace infinity with NaN
df_train.replace([np.inf, -np.inf], np.nan, inplace=True)

# Replace NaNs with the mean of the column only for numerical columns
numerical_cols = df_train.select_dtypes(exclude=('object','category')).columns
df_train[numerical_cols] = df_train[numerical_cols].fillna(df_train[numerical_cols].mean())
```

In [64]:
```
print(df_train.isnull().any().sum())
numeric_cols = df_train.select_dtypes(include=[np.number])
print(np.isinf(numeric_cols).any().sum())
```

```
0
0
```

In [61]:
```
print(df_train.shape)
print(df_train.select_dtypes(exclude=('object','category')).columns)
```

```
(27321, 80)
Index(['UID', 'SUMLEVEL', 'COUNTYID', 'STATEID', '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')
```

In [ ]:
```
fa = FactorAnalyzer(n_factors=2)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

Project Task: Week 4

Data Modeling :

1. Build a linear Regression model to predict the total monthly expenditure for home mortgages loan. Please refer 'deplotment_RE.xlsx'. Column hc_mortgage_mean is predicted variable. This is the mean monthly mortgage and owner costs of specified geographical location. Note: Exclude loans from prediction model which have NaN (Not a Number) values for hc_mortgage_mean.
   a. Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step. b. Run another model at State level. There are 52 states in USA. c. Keep below considerations while building a linear regression model. Data Modeling :

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

In [65]:
```
train.columns
```

Out[65]:
```
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 [66]:
```
train['type'].unique()
```

Out[66]:
```
array(['City', 'Urban', 'Town', 'CDP', 'Village', 'Borough'], dtype=object)
```

In [67]:
```
type_dict={'type':{'City':1, 'Urban':2, 'Town':3, 'CDP':4, 'Village':5, 'Borough':6}}
train.replace(type_dict,inplace=True)
```

In [68]:
```
test.replace(type_dict,inplace=True)
```

In [69]:
```
train['type'].unique()
```

Out[69]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [70]:
```
test['type'].unique()
```

Out[70]:
```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

In [71]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean','second_mortgage', 'home_equity', 'del
              'pct_own', 'married','separated', 'divorced']
```

In [72]:
```
X_train = train[feature_cols]
y_train = train['hc_mortgage_mean']
```

In [73]:
```
X_test = test[feature_cols]
y_test = test['hc_mortgage_mean']
```

In [74]:
```
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 [75]:
```
X_train.head()
```

Out[75]:

	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_degree	pct_own	marrie
0	53	36	13346	1	5230	67994.14790	0.02077	0.08919	0.52963	0.89288	0.79046	0.578
1	141	18	46616	1	2633	50670.10337	0.02222	0.04274	0.60855	0.90487	0.52483	0.348
2	63	18	46122	1	6881	95262.51431	0.00000	0.09512	0.73484	0.94288	0.85331	0.647
3	127	72	927	2	2700	56401.68133	0.01086	0.01086	0.52714	0.91500	0.65037	0.472
4	161	20	66502	1	5637	54053.42396	0.05426	0.05426	0.51938	1.00000	0.13046	0.123

In [76]:
```
X_test.head()
```

Out[76]:

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

In [77]:
```
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.

In [78]:
```
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
```

Out[78]:
```
▾ LinearRegression
LinearRegression()
```

In [79]:
```
y_pred= lr.predict(X_test_scaled)
```

R square of 60 percent and above should be achieved

In [80]:
```
r2_score(y_test,y_pred)
```

Out[80]:
```
0.7381882934134452
```

In [81]:
```
mean_absolute_error(y_test, y_pred)
```

Out[81]:
```
233.86965694140085
```

In [82]:
```
mean_squared_error(y_test, y_pred)
```

Out[82]:
```
103818.40486733473
```

In [83]:
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[83]:
```
322.20863561880947
```

In [84]:
```
r2_score(y_train, lr.predict(X_train_scaled))
```

Out[84]:
```
0.734344756627955
```

In [85]:
```
lr.coef_
```

Out[85]:
```
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 [86]:
```
X_train.columns
```

Out[86]:
```
Index(['COUNTYID', 'STATEID', 'zip_code', 'type', 'pop', 'family_mean',
       'second_mortgage', 'home_equity', 'debt', 'hs_degree', 'pct_own',
       'married', 'separated', 'divorced'],
      dtype='object')
```

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

In [87]:
```
state = train['STATEID'].unique()
state
```

Out[87]:
```
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 [88]:
```
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
    print("\n")
```

```
State ID- 11
Overall R2 score of linear regression model for state, 11 :- 0.7458953509562304
Overall RMSE of linear regression model for state, 11 :- 238.5227678809512


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8086161640279984
Overall RMSE of linear regression model for state, 1 :- 311.532907203562


State ID- 29
Overall R2 score of linear regression model for state, 29 :- 0.7090032526359473
Overall RMSE of linear regression model for state, 29 :- 270.06841264277546
```

Test if predicted variable is normally distributed

In [89]:
```
sns.distplot(y_pred)
plt.show()
```