Project Target :

Designing a ML for predicting properties prices in each area of California,

Criteria:

- **Population of the region,**
- **Average Earnings,**
- **Number of bedrooms,**

The concept of the "Reagin" in the project refers to the smallets geograpgic unit with a population of 600 and 3000 people,

Date of information: 1990

Technolologies and libraries Used:

- **Pandas,**
- **Numpy,**
- **Sklearn,**
- **Matplotlib,**

In [40]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from pandas.plotting import scatter_matrix
from sklearn.impute import SimpleImputer

# Custom transformer for creating new attributes by combining existing attributes
from sklearn.base import BaseEstimator, TransformerMixin

#Standardization Feature scaling
from sklearn.preprocessing import StandardScaler
```

In [8]:

```
#Download the "Housing Price.csv" file
property_price_report= pd.read_csv('property_price_report.csv')
```

In [9]:

```
#"head" is a function which shows the first 5 rows of our dataset
property_price_report.head()
```

Out[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	1
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	11
2	-122.24	37.85	52.0	1467.0	190.0	496.0	1
3	-122.25	37.85	52.0	1274.0	235.0	558.0	2
4	-122.25	37.85	52.0	1627.0	280.0	565.0	2

In [10]:

```
property_price_report.shape
```

Out[10]:

```
(20640, 10)
```

In [37]:

```
property_price_report.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [11]:

```
property_price_report.columns
```

Out[11]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [74]:

```
property_price_report['ocean_proximity']
```

Out[74]:

```
0        NEAR BAY
1        NEAR BAY
2        NEAR BAY
3        NEAR BAY
4        NEAR BAY
           ...
20635      INLAND
20636      INLAND
20637      INLAND
20638      INLAND
20639      INLAND
Name: ocean_proximity, Length: 20640, dtype: object
```

In [12]:

```
property_price_report['ocean_proximity'].unique()
```

Out[12]:

```
array(['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND'],
      dtype=object)
```

In [40]:

```
property_price_report['ocean_proximity'].value_counts()
```

Out[40]:

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [13]:

```
property_price_report[property_price_report['ocean_proximity']=='ISLAND']
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
8314	-118.32	33.35	27.0	1675.0	521.0	744.0	
8315	-118.33	33.34	52.0	2359.0	591.0	1100.0	
8316	-118.32	33.33	52.0	2127.0	512.0	733.0	
8317	-118.32	33.34	52.0	996.0	264.0	341.0	
8318	-118.48	33.43	29.0	716.0	214.0	422.0	

In [42]:

```
property_price_report['population'][property_price_report['ocean_proximity']=='ISLAND']
```

Out[42]:

```
8314     744.0
8315    1100.0
8316     733.0
8317     341.0
8318     422.0
Name: population, dtype: float64
```

In [14]:

```
property_price_report[['population','ocean_proximity']][property_price_report['ocean_pro
```

Out[14]:

	population	ocean_proximity
8314	744.0	ISLAND
8315	1100.0	ISLAND
8316	733.0	ISLAND
8317	341.0	ISLAND
8318	422.0	ISLAND

In [15]:

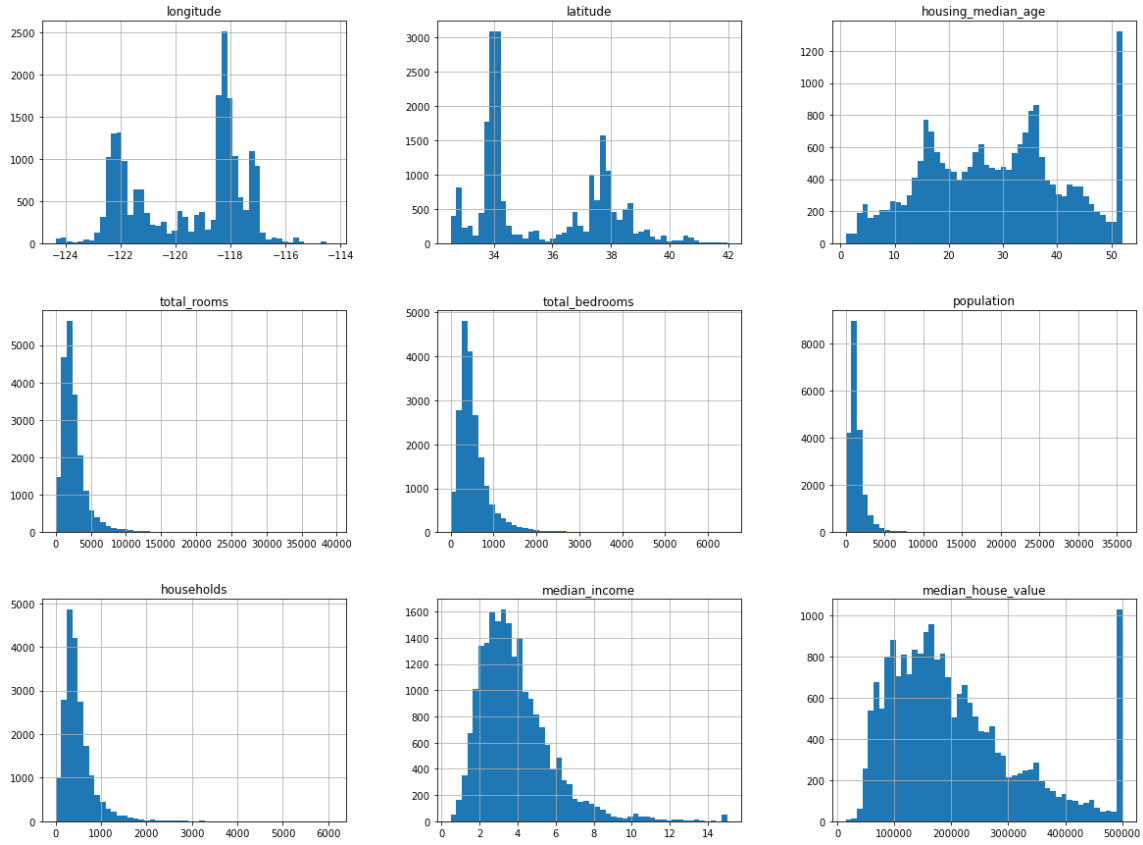
```
property_price_report.describe()
```

Out[15]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	2064
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	142
std	2.003532	2.135952	12.585558	2181.615252	421.385070	113
min	-124.350000	32.540000	1.000000	2.000000	1.000000	
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	78
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	116
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	172
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	3568

In [16]:

```
property_price_report.hist(bins=50,figsize=(20,15))
plt.show()
```



In [17]:

```
train_set, test_set = train_test_split(property_price_report, test_size = 0.2, random_st
```

In [51]:

```
train_set.shape
```

Out[51]:

```
(16512, 10)
```

In [18]:

```
data = train_set.copy()
```

In [19]:

```
data.shape
```

Out[19]:

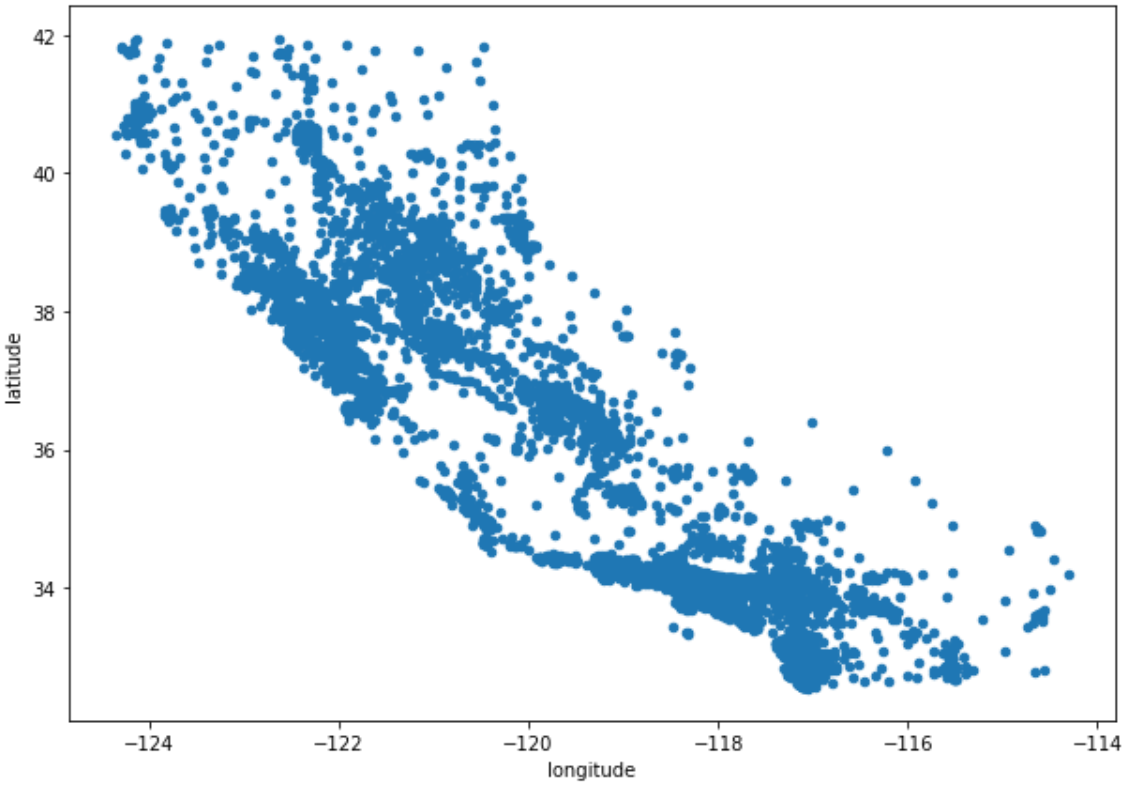
```
(16512, 10)
```

In [20]:

```
data.plot(kind='scatter', x="longitude", y="latitude", figsize=(10,7))
```

Out[20]:

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

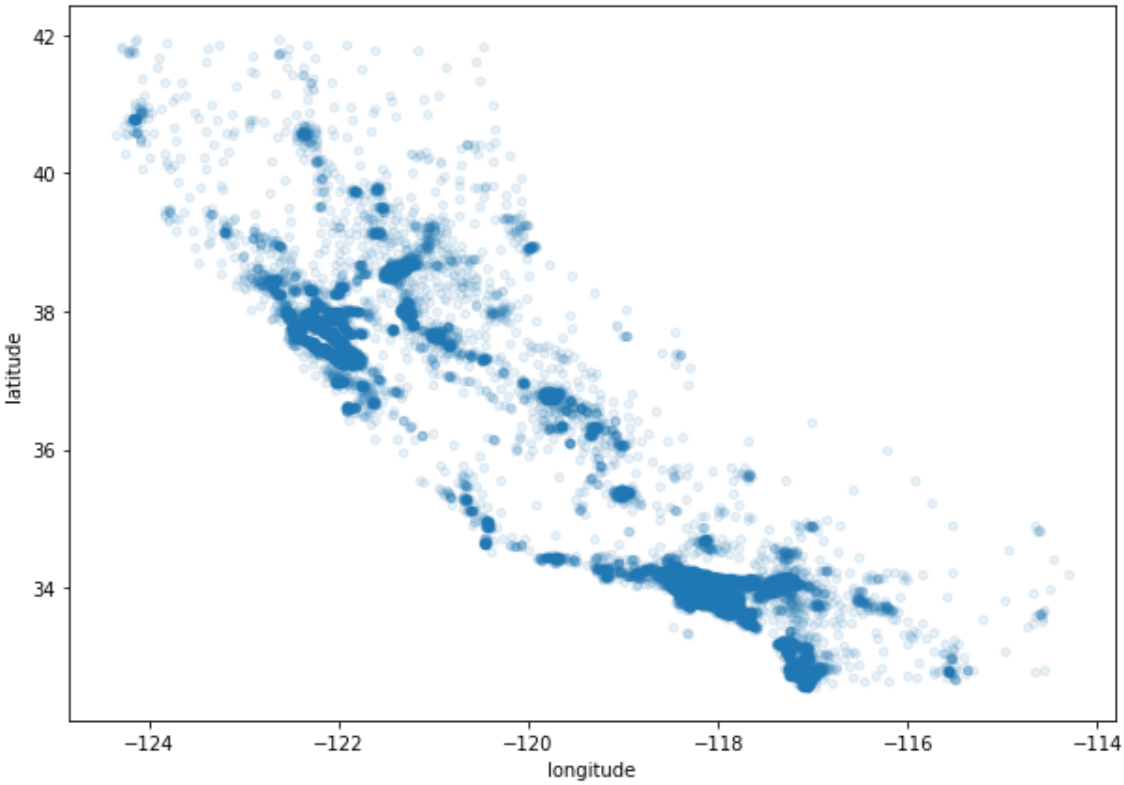


In [45]:

```
data.plot(kind='scatter', x="longitude", y="latitude", figsize=(10,7), alpha=0.1)
```

Out[45]:

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

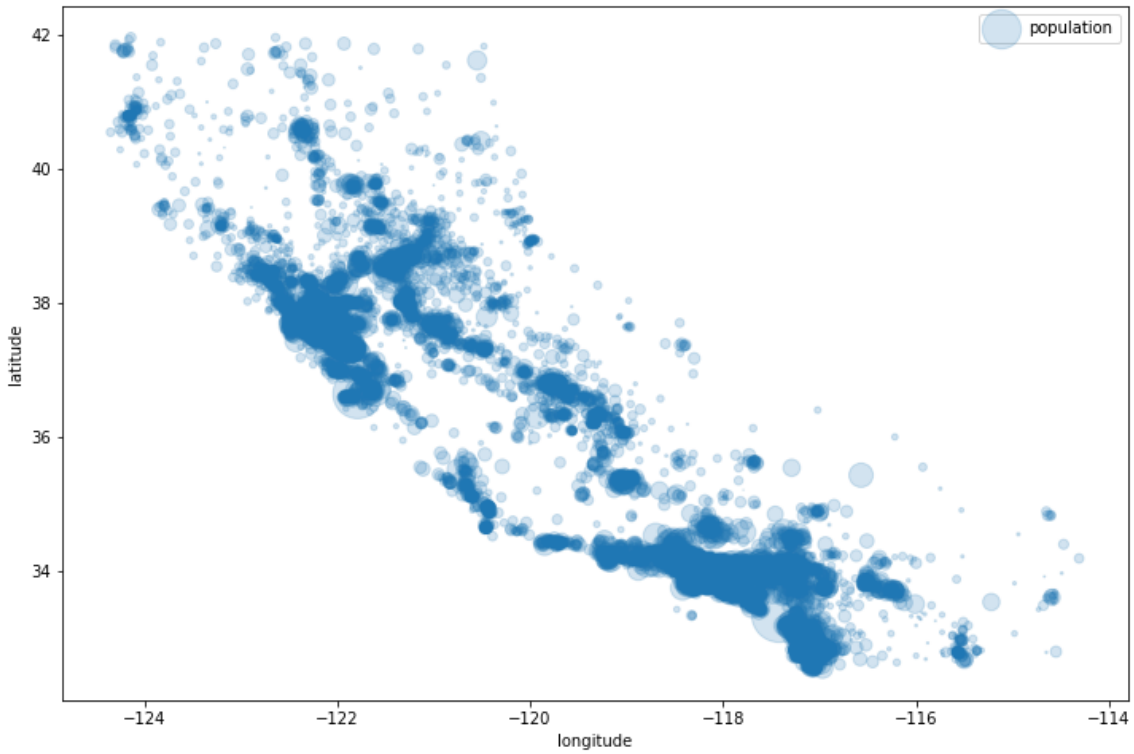


In [22]:

```
data.plot(kind='scatter', x="longitude", y="latitude",
          figsize=(12,8), alpha=0.2,
          s=data["population"]/30,
        label="population")
```

Out[22]:

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

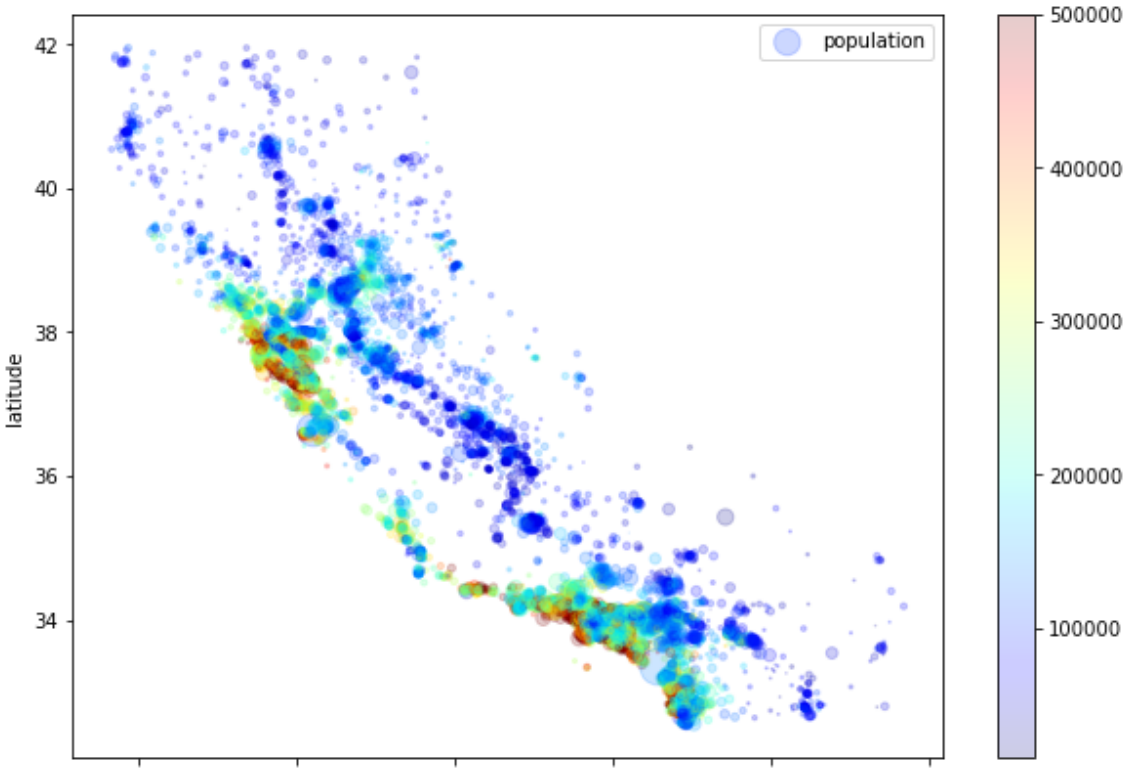


In [63]:

```
data.plot(kind='scatter', x="longitude", y="latitude",
          figsize=(10,7), alpha=0.2,
          s=data["population"]/100,
        label="population",
         c= data["median_house_value"], cmap= plt.get_cmap("jet"))
```

Out[63]:

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



In [24]:

```
# standard correlation coefficient
corr_matrix = data.corr()
corr_matrix
```

Out[24]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
longitude	1.000000	-0.924485	-0.101818	0.038676	0.063064
latitude	-0.924485	1.000000	0.005296	-0.029224	-0.059998
housing_median_age	-0.101818	0.005296	1.000000	-0.360922	-0.320624
total_rooms	0.038676	-0.029224	-0.360922	1.000000	0.930489
total_bedrooms	0.063064	-0.059998	-0.320624	0.930489	1.000000
population	0.094276	-0.102499	-0.292283	0.857936	0.878932
households	0.049306	-0.064061	-0.302796	0.920482	0.980255
median_income	-0.017040	-0.076571	-0.121711	0.198268	-0.009141
median_house_value	-0.046349	-0.142983	0.103706	0.133989	0.047980

In [25]:

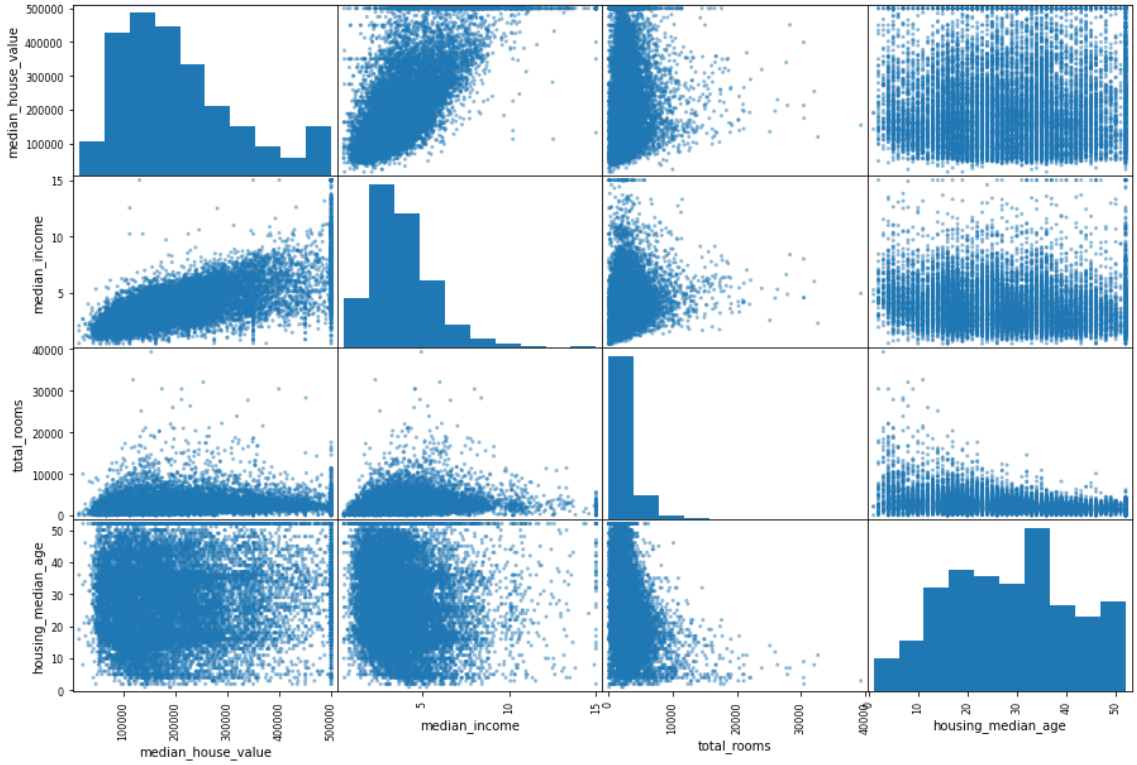
```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

Out[25]:

```
median_house_value    1.000000
median_income         0.690647
total_rooms           0.133989
housing_median_age    0.103706
households            0.063714
total_bedrooms        0.047980
population           -0.026032
longitude            -0.046349
latitude             -0.142983
Name: median_house_value, dtype: float64
```

In [26]:

```
features = ["median_house_value","median_income","total_rooms","housing_median_age"]
scatter_matrix(data[features], figsize = (15,10))
plt.show()
```

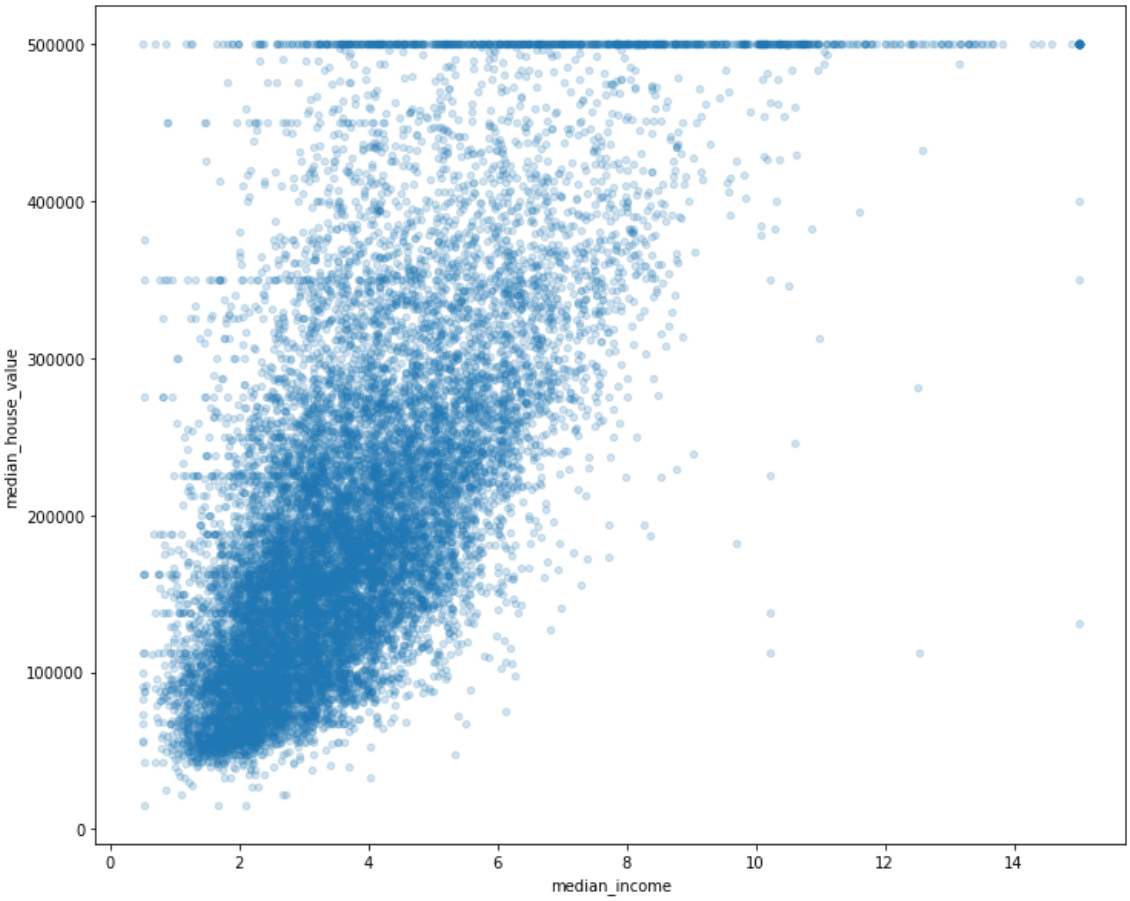


In [46]:

```
data.plot(kind="scatter", x= "median_income", y="median_house_value", alpha= 0.2, figsiz
```

Out[46]:

```
<AxesSubplot:xlabel='median_income', ylabel='median_house_value'>
```



In [28]:

```
data.head()
```

Out[28]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	ho
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	

In [29]:

```
data["total_bedrooms_per_total_rooms"]=data["total_bedrooms"]/data["total_rooms"]
data["population_per_households"]=data["population"]/data["households"]
data["total_rooms_per_households"]=data["total_rooms"]/data["households"]
data.head()
```

Out[29]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	ho
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	

In [30]:

```
corr_matrix = data.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

Out[30]:

```
median_house_value                1.000000
median_income                     0.690647
total_rooms_per_households        0.158485
total_rooms                       0.133989
housing_median_age                0.103706
households                        0.063714
total_bedrooms                    0.047980
population_per_households        -0.022030
population                       -0.026032
longitude                        -0.046349
latitude                         -0.142983
total_bedrooms_per_total_rooms   -0.257419
Name: median_house_value, dtype: float64
```

In [83]:

```
#Preparing the Data
```

In [31]:

```
df = train_set.copy()
```

In [32]:

```
df_label = df["median_house_value"].copy
df = df.drop(["median_house_value"], axis=1)
```

In [33]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16512 entries, 14196 to 15795
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           16512 non-null  float64
 1   latitude            16512 non-null  float64
 2   housing_median_age  16512 non-null  float64
 3   total_rooms         16512 non-null  float64
 4   total_bedrooms      16512 non-null  float64
 5   population          16512 non-null  float64
 6   households          16512 non-null  float64
 7   median_income       16512 non-null  float64
 8   ocean_proximity     16512 non-null  object 
dtypes: float64(8), object(1)
memory usage: 1.3+ MB
```

In [34]:

```
df_num = df.drop("ocean_proximity", axis = 1)
df_num
```

Out[34]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	ho
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	
...	...	...	...	...	...	...	
11284	-117.96	33.78	35.0	1330.0	201.0	658.0	
11964	-117.43	34.02	33.0	3084.0	570.0	1753.0	
5390	-118.38	34.03	36.0	2101.0	569.0	1756.0	
860	-121.96	37.58	15.0	3575.0	597.0	1777.0	
15795	-122.42	37.77	52.0	4226.0	1315.0	2619.0	

16512 rows × 8 columns

In [35]:

```
imputer = SimpleImputer(missing_values = np.nan, strategy = 'median')
imputer.fit(df_num)
x = imputer.transform(df_num)
df_num_impute_tr = pd.DataFrame(x, columns= df_num.columns)
df_num_impute_tr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16512 entries, 0 to 16511
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           16512 non-null  float64
 1   latitude            16512 non-null  float64
 2   housing_median_age  16512 non-null  float64
 3   total_rooms         16512 non-null  float64
 4   total_bedrooms      16512 non-null  float64
 5   population          16512 non-null  float64
 6   households          16512 non-null  float64
 7   median_income       16512 non-null  float64
dtypes: float64(8)
memory usage: 1.0 MB
```

In [36]:

```
# Custom transformer for creating new three attributes by combining existing attributes

rooms_ix, bedrooms_ix, population_ix, household_ix = 3, 4, 5, 6

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):   # Nothing to do in fit in this project
          return self

    def trafnsform(self, X, y=None):
        rooms_per_household      = X[:, rooms_ix] / X[:, household_ix]
        population_per_household = X[:, population_ix ] / X[:, household_ix]
        bedrooms_per_rooms       = X[:, bedrooms_ix] / X[:, rooms_ix]
        return np.c_[X, rooms_per_household, population_per_household, bedrooms_per_room
```

In [37]:

```
custom = CombinedAttributesAdder()
data_custom_tr_temp = custom.trafnsform(df_num_impute_tr.values)
data_cutom_tr = pd.DataFrame(data_custom_tr_temp)
```

In [38]:

```
columns = list(df_num_impute_tr.columns)
columns.append("rooms_per_household")
columns.append("population_per_household")
columns.append("bedrooms_per_rooms")
data_cutom_tr.columns = columns
data_cutom_tr.head(10)
```

Out[38]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-117.03	32.71	33.0	3126.0	627.0	2300.0	6
1	-118.16	33.77	49.0	3382.0	787.0	1314.0	7
2	-120.48	34.66	4.0	1897.0	331.0	915.0	3
3	-117.11	32.69	36.0	1421.0	367.0	1418.0	3
4	-119.80	36.78	43.0	2382.0	431.0	874.0	3
5	-121.86	37.42	20.0	5032.0	808.0	2695.0	8
6	-117.97	34.04	28.0	1686.0	417.0	1355.0	3
7	-122.53	37.91	37.0	2524.0	398.0	999.0	4
8	-117.90	34.13	5.0	1126.0	316.0	819.0	3
9	-117.79	34.02	5.0	18690.0	2862.0	9427.0	27

In [39]:

```
data_cutom_tr.describe()
```

Out[39]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	16512.000000	16512.000000	16512.000000	16512.000000	16512.000000	165
mean	-119.582290	35.643149	28.608285	2642.004784	538.496851	142
std	2.005654	2.136665	12.602499	2174.646744	419.007096	113
min	-124.350000	32.550000	1.000000	2.000000	1.000000	
25%	-121.810000	33.930000	18.000000	1454.000000	296.750000	78
50%	-118.510000	34.260000	29.000000	2129.000000	437.000000	116
75%	-118.010000	37.720000	37.000000	3160.000000	647.000000	172
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	3568

In [42]:

```
# Feature Scaling (Standardization Mathod)
feature_scale = StandardScaler()
data_num_scaled_tr = pd.DataFrame(feature_scale.fit_transform(data_cutom_tr.values), col
data_num_scaled_tr.head(10)
```

Out[42]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
0	1.272587	-1.372811	0.348490	0.222569	0.211228	0.768276	0.3
1	0.709162	-0.876696	1.618118	0.340293	0.593094	-0.098901	0.6
2	-0.447603	-0.460146	-1.952710	-0.342597	-0.495226	-0.449818	-0.4
3	1.232698	-1.382172	0.586545	-0.561490	-0.409306	-0.007434	-0.3
4	-0.108551	0.532084	1.142008	-0.119565	-0.256559	-0.485877	-0.3
5	-1.135679	0.831625	-0.683082	1.099060	0.643214	1.115675	0.7
6	0.803897	-0.750327	-0.048268	-0.439627	-0.289972	-0.062842	-0.2
7	-1.469745	1.060961	0.665897	-0.054266	-0.335319	-0.375941	-0.2
8	0.838800	-0.708204	-1.873359	-0.697148	-0.531026	-0.534249	-0.4
9	0.893646	-0.759688	-1.873359	7.379811	5.545428	7.036405	5.9

In [ ]:

```

```