In [3]:
```python
import pandas as pd
import numpy as np
```

In [4]:
```python
df = pd.read_csv("housing.csv")
df
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND

20640 rows × 10 columns

In [5]:
```python
df.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 [7]:
```python
df["ocean_proximity"].value_counts()
```

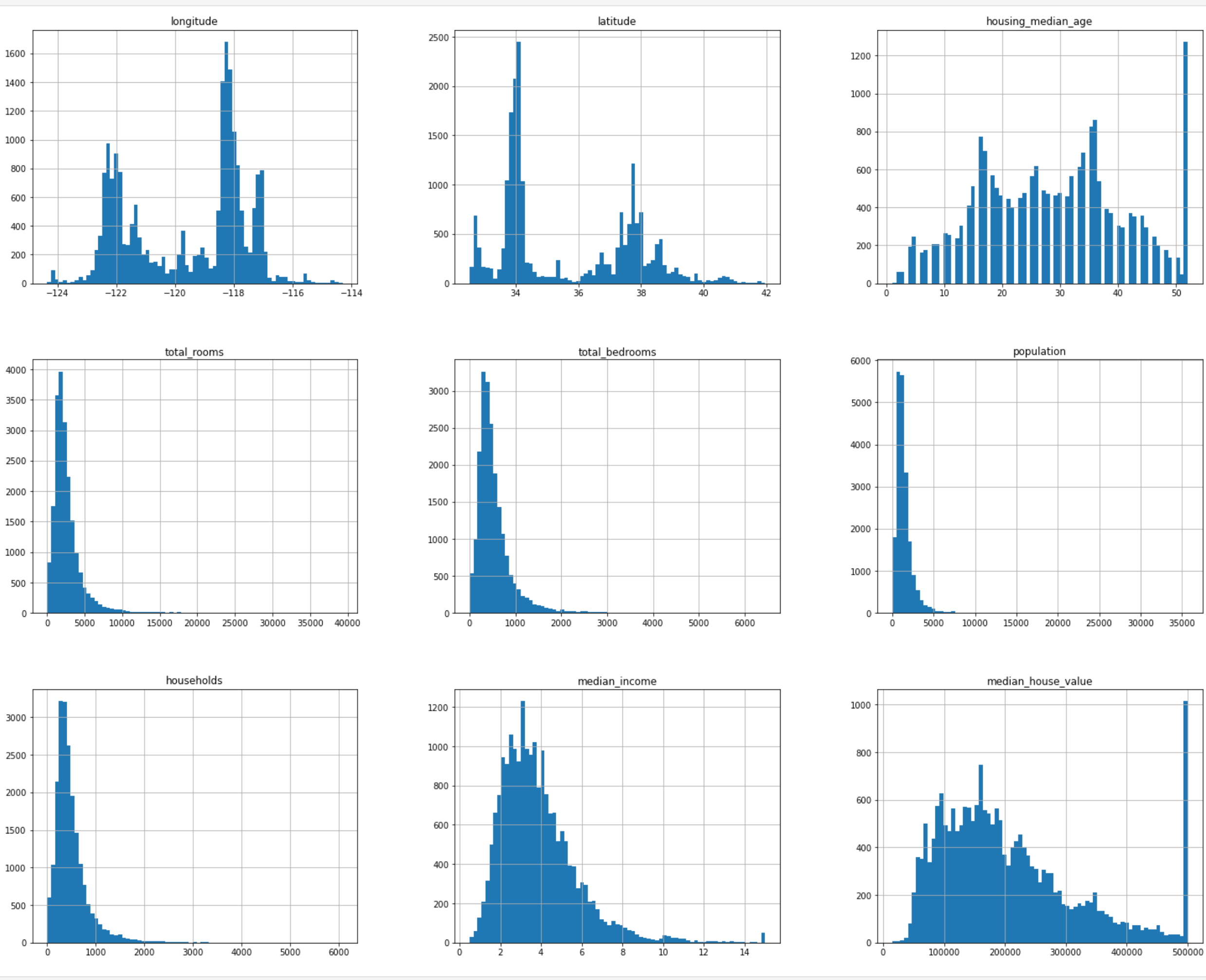
Out[7]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [8]:
```python
df.describe()
```

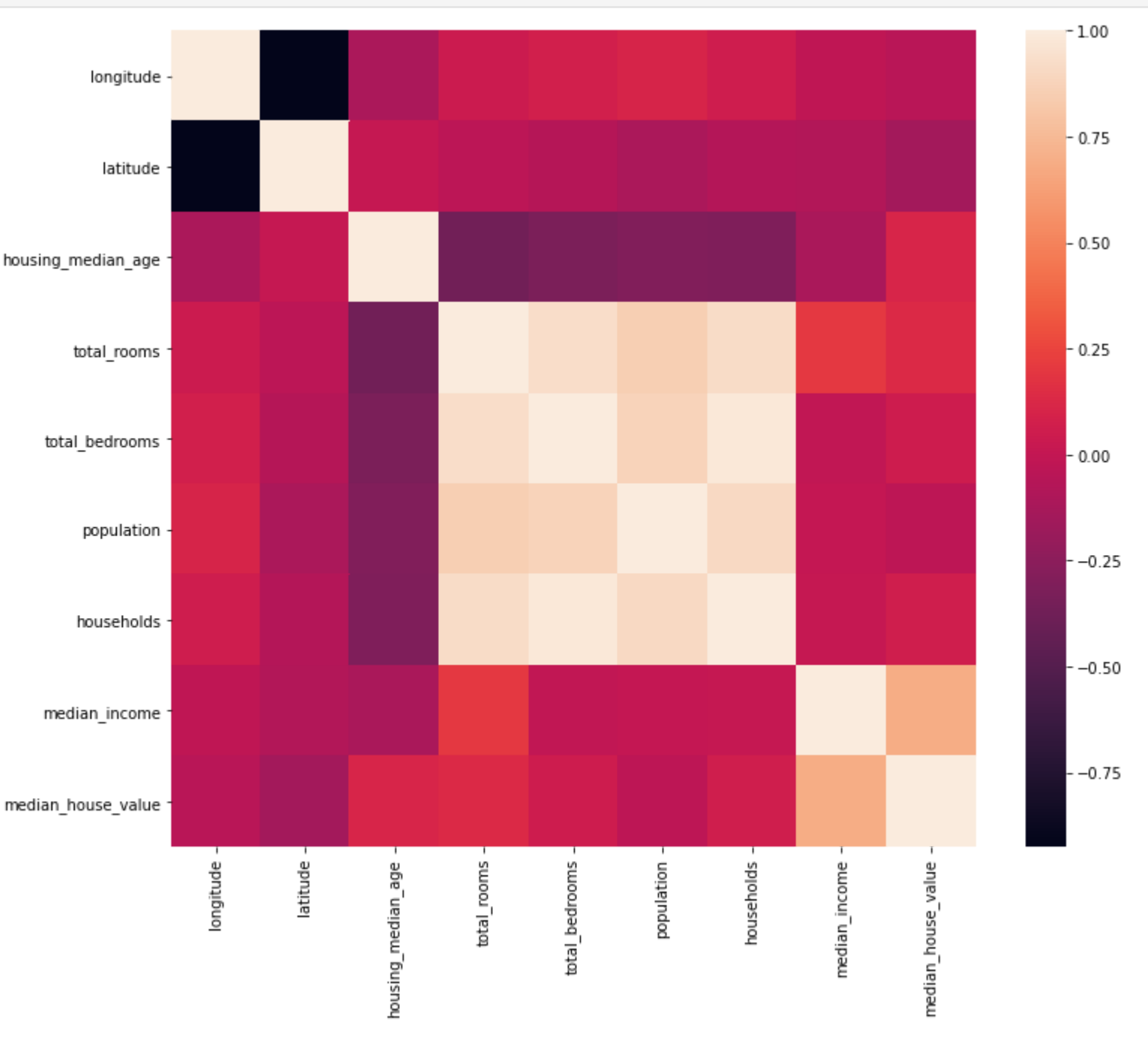
Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

In [9]:
```python
import matplotlib.pyplot  as plt
df.hist(bins=75,figsize=(25,20))
plt.show()
```

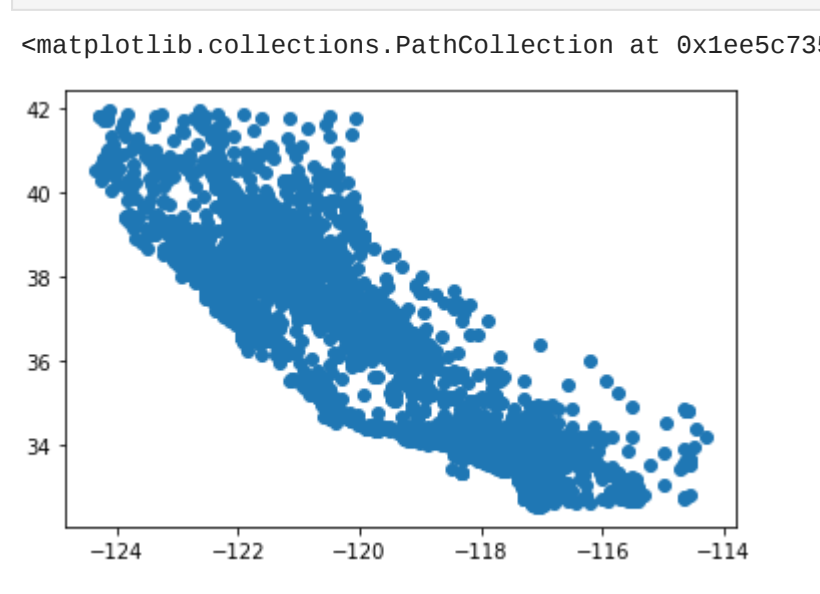


In [12]:
```python
import seaborn as sns
plt.figure(figsize=(12,10))
df_corr = df.corr()
sns.heatmap(df_corr)
plt.show()
```



In [14]:
```python
plt.scatter(df["longitude"],df["latitude"])
```

Out[14]: <matplotlib.collections.PathCollection at 0x1ee5c7356d0>



In [15]:
```python
df_corr["median_house_value"].sort_values(ascending=False)
```

Out[15]:
```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

In [19]:
```python
df.isnull().sum()
```

Out[19]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

In [22]:
```python
df["total_bedrooms"].fillna(df["total_bedrooms"].median(),inplace=True)
```

In [23]:
```python
df.isnull().sum()
```

Out[23]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

In [24]:
```python
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
housing_category = df["ocean_proximity"]
housing_category_encoded = encoder.fit_transform(housing_category)
housing_category_encoded
```

Out[24]: array([3, 3, 3, ..., 1, 1, 1])

In [25]:
```python
encoder.classes_
```

Out[25]:
```
array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
      dtype=object)
```

In [ ]:

In [91]:
```python
from sklearn.preprocessing import OneHotEncoder
encoder_1 = OneHotEncoder()
encoded_1 = encoder_1.fit_transform(housing_category_encoded.reshape(-1,1))
encoded_1
```

Out[91]:
```
<20640x5 sparse matrix of type '<class 'numpy.float64'>'
	with 20640 stored elements in Compressed Sparse Row format>
```

In [41]:
```python
encoded_1_Arr = encoded_1.toarray()
encoded_1_Arr
```

Out[41]:
```
array([[0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.]])
```

In [28]:
```python
rissh = df.drop("ocean_proximity",axis=1)
rissh
```

Out[28]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0
...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0

20640 rows × 9 columns

In [30]:
```python
rissh_x = rissh.drop("median_house_value",axis=1)
rissh_y = rissh.iloc[:,-1]
rissh_y
```

Out[30]:
```
0        452600.0
1        358500.0
2        352100.0
3        341300.0
4        342200.0
           ...
20635     78100.0
20636     77100.0
20637     92300.0
20638     84700.0
20639     89400.0
Name: median_house_value, Length: 20640, dtype: float64
```

In [34]:
```python
from sklearn.preprocessing import StandardScaler
scalar =StandardScaler()
rissh_scaled = scalar.fit_transform(rissh_x)
rissh_scaled = pd.DataFrame(rissh_scaled)
rissh_scaled
```

Out[34]:

	0	1	2	3	4	5	6	7
0	-1.327835	1.052548	0.982143	-0.804819	-0.972476	-0.974429	-0.977033	2.344766
1	-1.322844	1.043185	-0.607019	2.045890	1.357143	0.861439	1.669961	2.332238
2	-1.332827	1.038503	1.856182	-0.535746	-0.827024	-0.820777	-0.843637	1.782699
3	-1.337818	1.038503	1.856182	-0.624215	-0.719723	-0.766028	-0.733781	0.932968
4	-1.337818	1.038503	1.856182	-0.462404	-0.612423	-0.759847	-0.629157	-0.012881
...	...	...	...	...	...	...	...	...
20635	-0.758826	1.801647	-0.289187	-0.444985	-0.388283	-0.512592	-0.443449	-1.216128
20636	-0.818722	1.806329	-0.845393	-0.888704	-0.922403	-0.944405	-1.008420	-0.691593
20637	-0.823713	1.778237	-0.924851	-0.174995	-0.123608	-0.369537	-0.174042	-1.142593
20638	-0.873626	1.778237	-0.845393	-0.355600	-0.304827	-0.604429	-0.393753	-1.054583
20639	-0.833696	1.750146	-1.004309	0.068408	0.188757	-0.033977	0.079672	-0.780129

20640 rows × 8 columns

In [ ]:

In [49]:
```python
encoded_1_Arr_df = pd.DataFrame(encoded_1_Arr,columns=[["House_Cat_1","House_Cat_2","House_Cat_3","House_Cat_4","House_Cat_5"]])
encoded_1_Arr_df
```

Out[49]:

	House_Cat_1	House_Cat_2	House_Cat_3	House_Cat_4	House_Cat_5
0	0.0	0.0	0.0	1.0	0.0
1	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...
20635	0.0	1.0	0.0	0.0	0.0
20636	0.0	1.0	0.0	0.0	0.0
20637	0.0	1.0	0.0	0.0	0.0
20638	0.0	1.0	0.0	0.0	0.0
20639	0.0	1.0	0.0	0.0	0.0

20640 rows × 5 columns

In [57]:
```python
rissh_X_Prepared = pd.concat([rissh_scaled,encoded_1_Arr_df],axis=1)
rissh_X_Prepared
```

Out[57]:

	0	1	2	3	4	5	6	7	House_cat_1	House_cat_2	House_cat_3	House_cat_4	(House_Cat_1,)	(House_Cat_2,)	(House_Cat_3,)	(H
0	-1.327835	1.052548	0.982143	-0.804819	-0.972476	-0.974429	-0.977033	2.344766	NaN	NaN	NaN	NaN	0.0	0.0	0.0	
1	-1.322844	1.043185	-0.607019	2.045890	1.357143	0.861439	1.669961	2.332238	NaN	NaN	NaN	NaN	0.0	0.0	0.0	
2	-1.332827	1.038503	1.856182	-0.535746	-0.827024	-0.820777	-0.843637	1.782699	NaN	NaN	NaN	NaN	0.0	0.0	0.0	
3	-1.337818	1.038503	1.856182	-0.624215	-0.719723	-0.766028	-0.733781	0.932968	NaN	NaN	NaN	NaN	0.0	0.0	0.0	
4	-1.337818	1.038503	1.856182	-0.462404	-0.612423	-0.759847	-0.629157	-0.012881	NaN	NaN	NaN	NaN	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20635	-0.758826	1.801647	-0.289187	-0.444985	-0.388283	-0.512592	-0.443449	-1.216128	NaN	NaN	NaN	NaN	0.0	1.0	0.0	
20636	-0.818722	1.806329	-0.845393	-0.888704	-0.922403	-0.944405	-1.008420	-0.691593	NaN	NaN	NaN	NaN	0.0	1.0	0.0	
20637	-0.823713	1.778237	-0.924851	-0.174995	-0.123608	-0.369537	-0.174042	-1.142593	NaN	NaN	NaN	NaN	0.0	1.0	0.0	
20638	-0.873626	1.778237	-0.845393	-0.355600	-0.304827	-0.604429	-0.393753	-1.054583	NaN	NaN	NaN	NaN	0.0	1.0	0.0	
20639	-0.833696	1.750146	-1.004309	0.068408	0.188757	-0.033977	0.079672	-0.780129	NaN	NaN	NaN	NaN	0.0	1.0	0.0	

20640 rows × 17 columns

In [58]:
```python
rissh_X_Prepare = rissh_X_Prepared.dropna(axis=1)
rissh_X_Prepare
```

Out[58]:

	0	1	2	3	4	5	6	7	(House_Cat_1,)	(House_Cat_2,)	(House_Cat_3,)	(House_Cat_4,)	(House_Cat_5,)
0	-1.327835	1.052548	0.982143	-0.804819	-0.972476	-0.974429	-0.977033	2.344766	0.0	0.0	0.0	1.0	0.0
1	-1.322844	1.043185	-0.607019	2.045890	1.357143	0.861439	1.669961	2.332238	0.0	0.0	0.0	1.0	0.0
2	-1.332827	1.038503	1.856182	-0.535746	-0.827024	-0.820777	-0.843637	1.782699	0.0	0.0	0.0	1.0	0.0
3	-1.337818	1.038503	1.856182	-0.624215	-0.719723	-0.766028	-0.733781	0.932968	0.0	0.0	0.0	1.0	0.0
4	-1.337818	1.038503	1.856182	-0.462404	-0.612423	-0.759847	-0.629157	-0.012881	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
20635	-0.758826	1.801647	-0.289187	-0.444985	-0.388283	-0.512592	-0.443449	-1.216128	0.0	1.0	0.0	0.0	0.0
20636	-0.818722	1.806329	-0.845393	-0.888704	-0.922403	-0.944405	-1.008420	-0.691593	0.0	1.0	0.0	0.0	0.0
20637	-0.823713	1.778237	-0.924851	-0.174995	-0.123608	-0.369537	-0.174042	-1.142593	0.0	1.0	0.0	0.0	0.0
20638	-0.873626	1.778237	-0.845393	-0.355600	-0.304827	-0.604429	-0.393753	-1.054583	0.0	1.0	0.0	0.0	0.0
20639	-0.833696	1.750146	-1.004309	0.068408	0.188757	-0.033977	0.079672	-0.780129	0.0	1.0	0.0	0.0	0.0

20640 rows × 13 columns

In [59]:
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(rissh_X_Prepare,rissh_y,test_size=0.25)
print(x_train.shape,y_test.shape)
```

```
(15480, 13) (5160,)
```

In [60]:
```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
y_pred =  model.predict(x_test)
y_pred
```

Out[60]:
```
array([105807.54013013, 227495.59068338, 187248.90677909, ...,
       232393.94674771, 148383.28576013, 199826.88664005])
```

In [75]:
```python
from sklearn.metrics import mean_squared_error
msedt = mean_squared_error(y_test,y_pred)
msedt
```

Out[75]: 5001567802.565698

In [62]:
```python
model.score(x_train,y_train)*100
```

Out[62]: 64.4144766565902

In [77]:
```python
from sklearn.tree import DecisionTreeRegressor
tr = DecisionTreeRegressor()
tr.fit(x_train,y_train)
y_pred_dt = tr.predict(x_test)
y_pred_dt
```

Out[77]: array([153600., 340500., 148200., ..., 279900., 159200., 200000.])

In [78]:
```python
from sklearn.metrics import mean_squared_error
msedt = mean_squared_error(y_test,y_pred_dt)
msedt
```

Out[78]: 4901206908.532558

In [80]:
```python
from sklearn.ensemble import RandomForestRegressor
rf_r = RandomForestRegressor()
rf_r.fit(x_train,y_train)
y_pred_rf = rf_r.predict(x_test)
y_pred_rf
```

Out[80]:
```
array([112448.  , 369548.16, 173863.  , ..., 165725.  , 156817.  ,
       221180.01])
```

In [81]:
```python
rf_sq = mean_squared_error(y_test,y_pred_rf)
rf_sq
```

Out[81]: 2460718460.0699463

In [83]:
```python
rf_r.score(x_train,y_train)*100
```

Out[83]: 97.52418916162283

In [ ]: