In [29]:
```
#importing libaries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Load the data :

- Read the “housing.csv” file from the folder into the program.
- Print first few rows of this data.
- Extract input (X) and output (Y) data from the dataset.

In [30]:
```
data = pd.read_excel('1553768847_housing.xlsx')
```

In [31]:
```
data.shape
```

Out[31]:
```
(20640, 10)
```

In [32]:
```
data.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  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   ocean_proximity     20640 non-null  object 
 9   median_house_value  20640 non-null  int64  
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [33]:
```
data.head()
```

Out[33]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
0	-122.23	37.88	41	880	129.0	322	[illegible]
1	-122.22	37.86	21	7099	1106.0	2401	1[illegible]
2	-122.24	37.85	52	1467	190.0	496	[illegible]
3	-122.25	37.85	52	1274	235.0	558	[illegible]
4	-122.25	37.85	52	1627	280.0	565	[illegible]

In [34]:
```
data.columns
```

Out[34]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity', 'median_house_value'],
      dtype='object')
```

- y = ['mdeian_house_value']
- X = ['longitude', 'latitude', 'housing_median_age', 'total_rooms','total_bedrooms', 'population', 'households', 'median_income','ocean_proximity']

In [ ]:
```

```

1. Handle missing values :

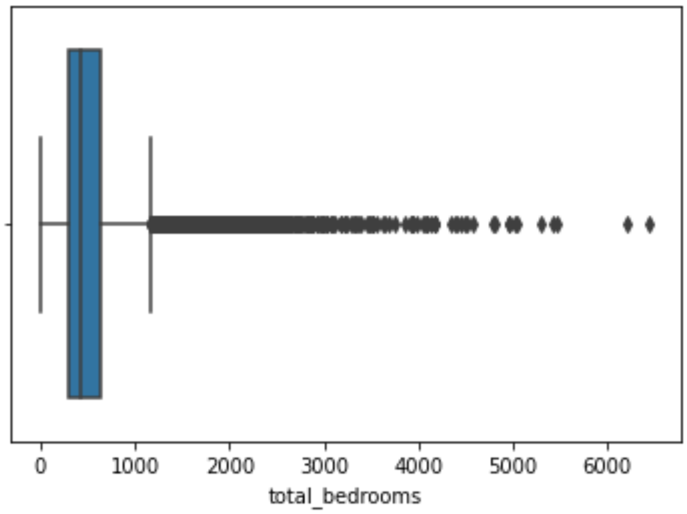
   - Fill the missing values with the mean of the respective column.

In [35]:
```
data.isnull().sum()
```

Out[35]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

In [36]:
```
sns.boxplot(data['total_bedrooms'])
```

Out[36]: <AxesSubplot:xlabel='total_bedrooms'>



In [37]:
```
data['total_bedrooms'].fillna(data['total_bedrooms'].mean(), inplace=True)
```

In [38]:
```
data.isnull().sum()
```

Out[38]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

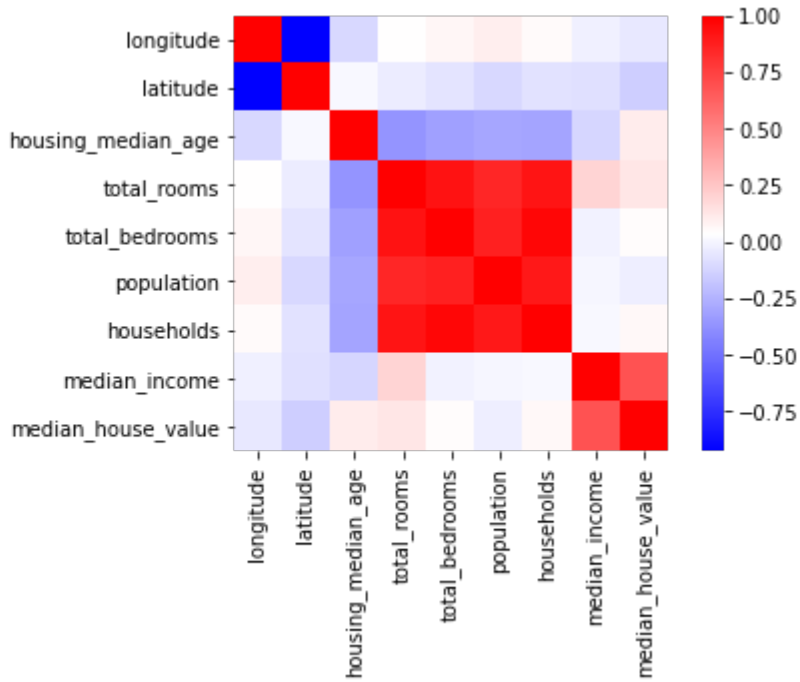
EDA

In [40]:
```
#checking correlation
data.corr()
```

Out[40]:

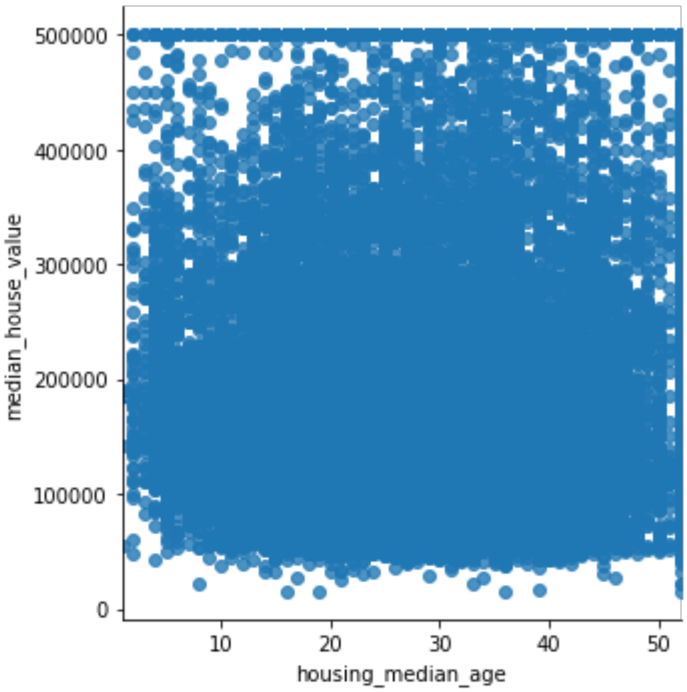
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069260
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066658
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.318998
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.927253
total_bedrooms	0.069260	-0.066658	-0.318998	0.927253	1.000000
population	0.099773	-0.108785	-0.296244	0.857126	0.873910
households	0.055310	-0.071035	-0.302916	0.918484	0.974725
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007682
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049454

In [41]:
```
sns.heatmap(data.corr(),square=True,cmap='bwr')
plt.show()
```



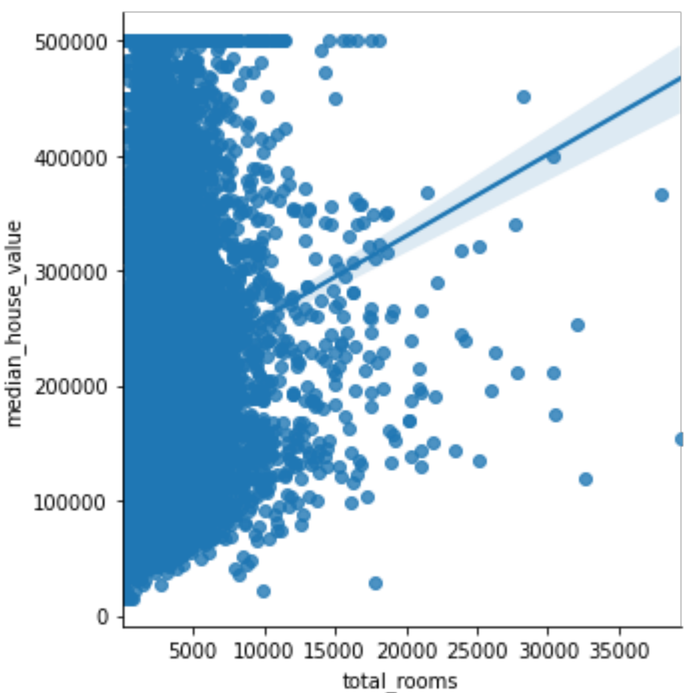
In [42]:
```
sns.lmplot(x ='housing_median_age', y ='median_house_value', data = dat
a)
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x7f39e13c04d0>



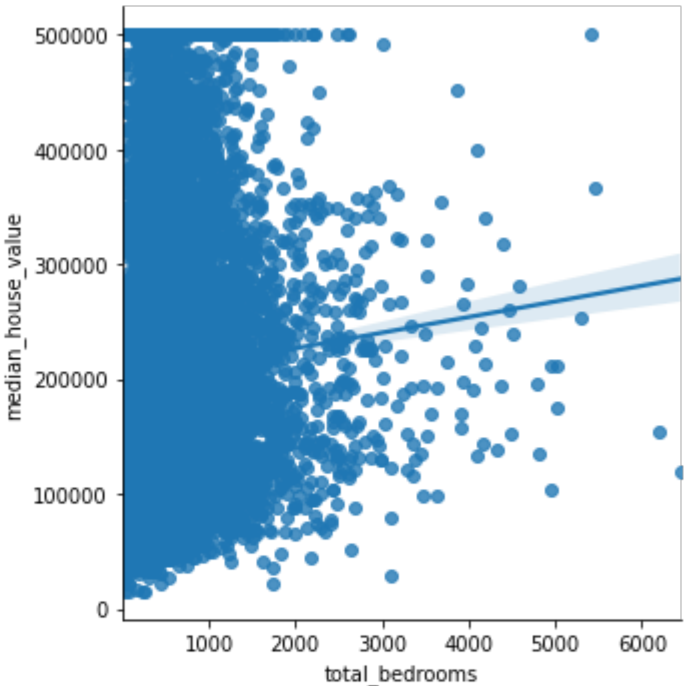
In [43]:
```
sns.lmplot(x ='total_rooms', y ='median_house_value', data = data)
```

Out[43]: <seaborn.axisgrid.FacetGrid at 0x7f39e7d839d0>



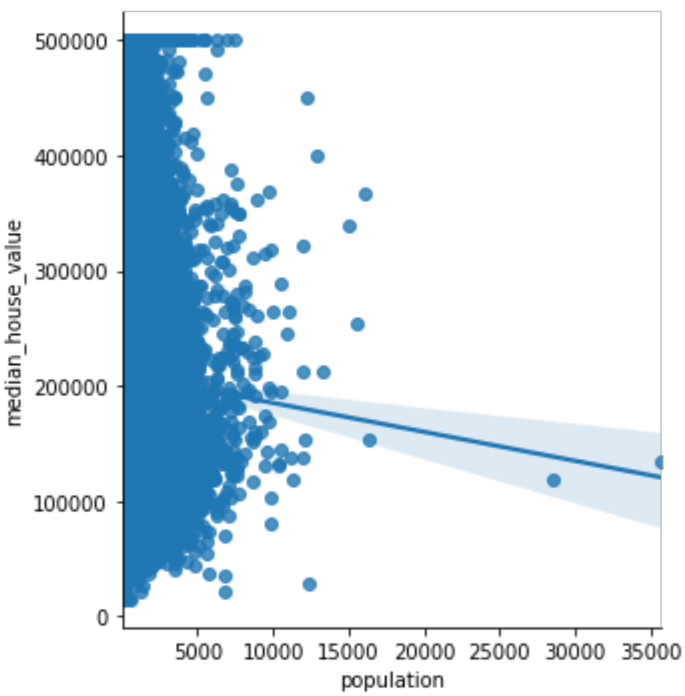
In [44]:
```
sns.lmplot(x ='total_bedrooms', y ='median_house_value', data = data)
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x7f39e132dc10>



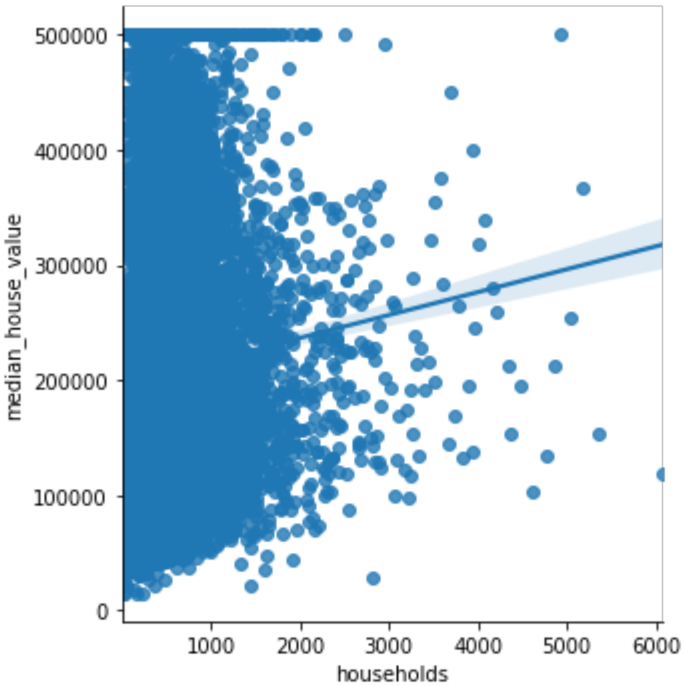
In [45]:
```
sns.lmplot(x ='population', y ='median_house_value', data = data)
```

Out[45]: <seaborn.axisgrid.FacetGrid at 0x7f39e0df5ed0>



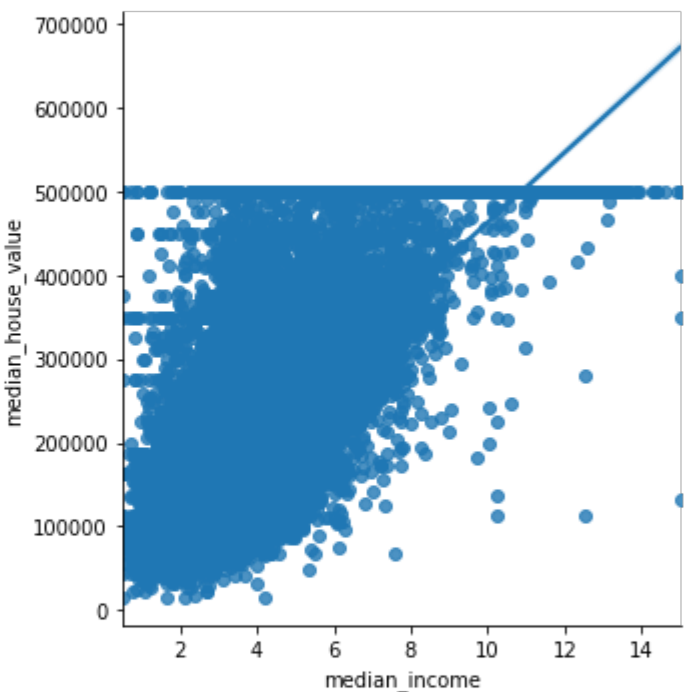
In [46]:
```
sns.lmplot(x ='households', y ='median_house_value', data = data)
```

Out[46]: <seaborn.axisgrid.FacetGrid at 0x7f39e0de6950>



In [47]:
```
sns.lmplot(x ='median_income', y ='median_house_value', data = data)
```

Out[47]: <seaborn.axisgrid.FacetGrid at 0x7f39e12b0850>

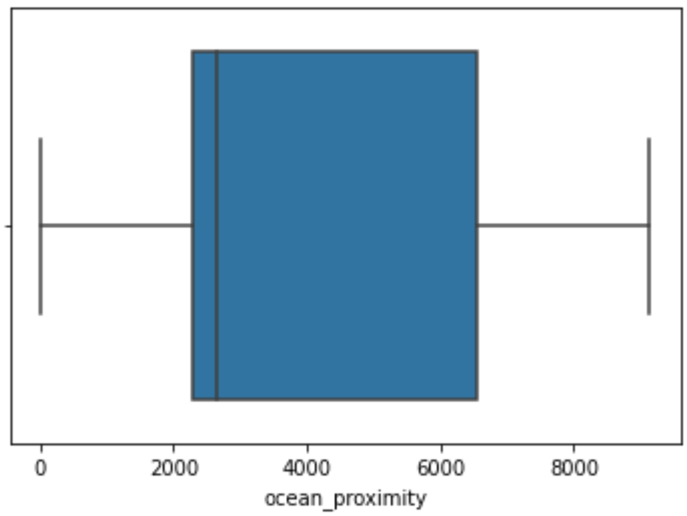


In [48]:
```
# froom the plots, it is observed that only median_income has high post
ive correlation comapred to others
# population shows a negative correlation but does not have considerabl
e signficance
# oter fetaures are some what corelated but not much significant
```

In [49]:
```
#Checking the categorical variable
```

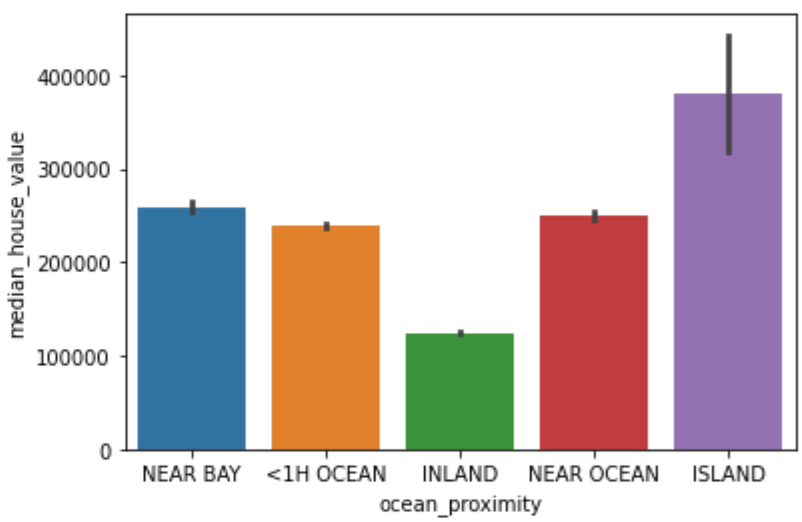
In [50]:
```
sns.boxplot(data['ocean_proximity'].value_counts())
```

Out[50]: <AxesSubplot:xlabel='ocean_proximity'>



In [51]:
```
sns.barplot(data=data, x= data['ocean_proximity'], y=data['median_house
_value'])
```

Out[51]: <AxesSubplot:xlabel='ocean_proximity', ylabel='median_house_value'>

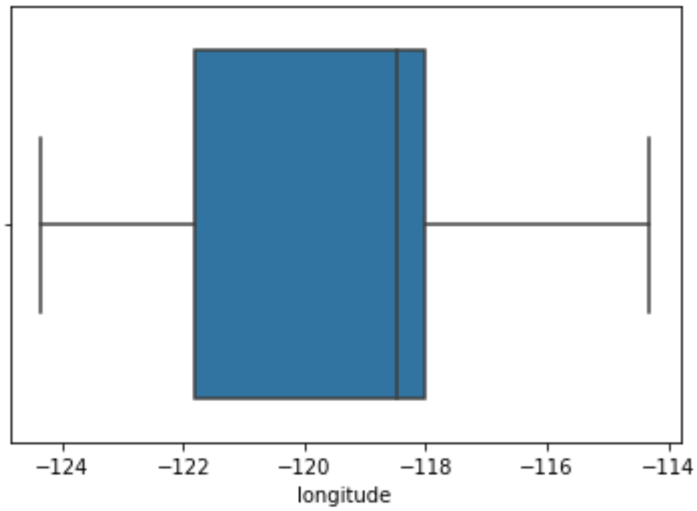


In [52]:
```
# we see that the median_house_vlue in Island apperas to be the most co
mpared Inland.
# median_house_vlue for Near BAY, <1H Ocean and Near Ocean, are mostly
similar
```

In [53]:
```
# checking Outliers
```

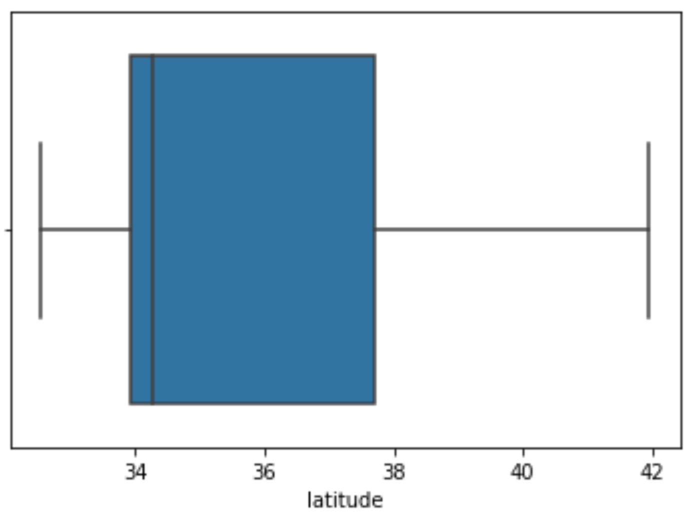
In [54]:
```
sns.boxplot(data['longitude'])
```

Out[54]: <AxesSubplot:xlabel='longitude'>



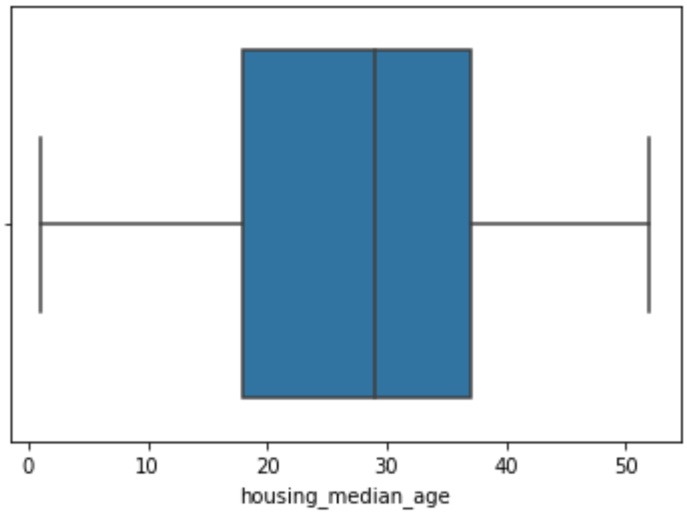
In [55]:
```
sns.boxplot(data['latitude'])
```

Out[55]: <AxesSubplot:xlabel='latitude'>



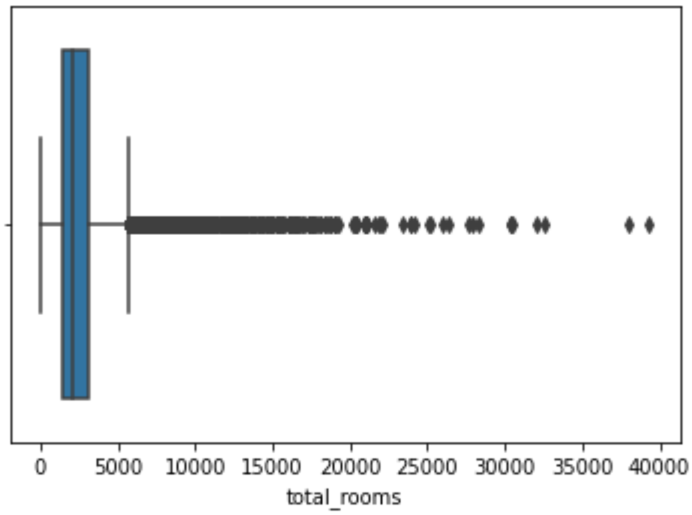
In [56]:
```
sns.boxplot(data['housing_median_age'])
```

Out[56]: <AxesSubplot:xlabel='housing_median_age'>



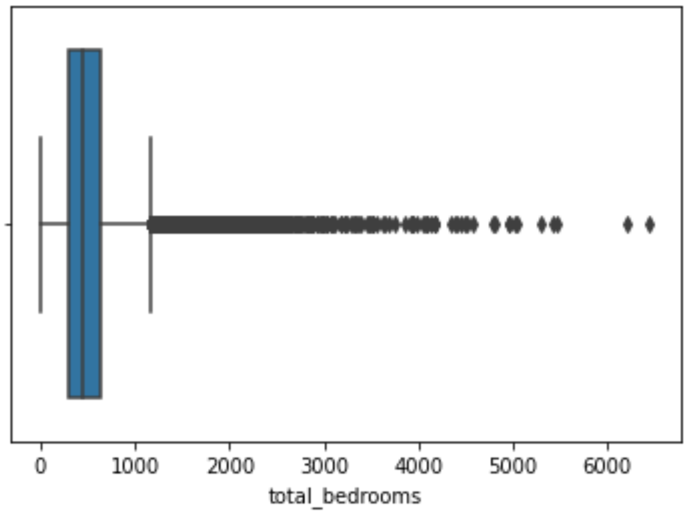
In [57]:
```
sns.boxplot(data['total_rooms'])
```

Out[57]: <AxesSubplot:xlabel='total_rooms'>



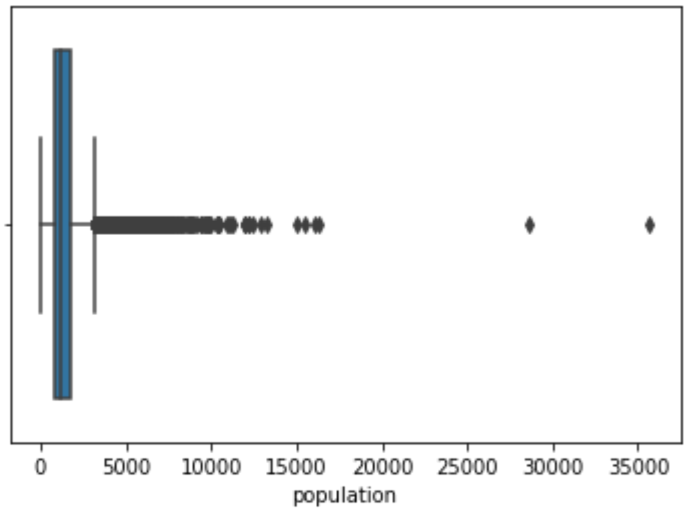
In [58]:
```
sns.boxplot(data['total_bedrooms'])
```

Out[58]: <AxesSubplot:xlabel='total_bedrooms'>



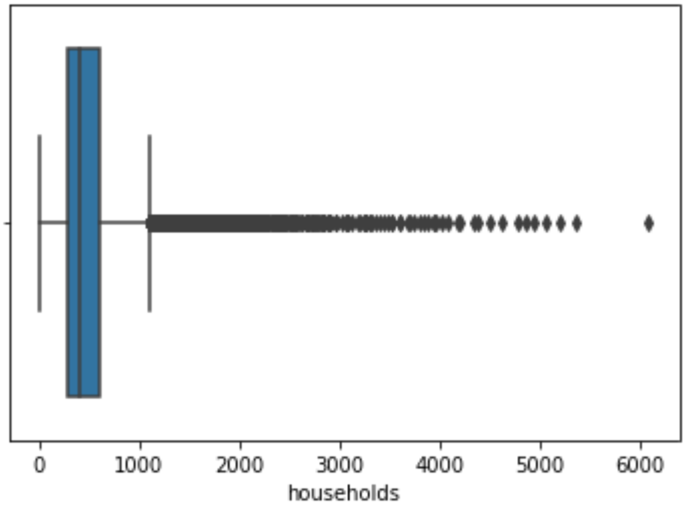
In [59]:
```
sns.boxplot(data['population'])
```

Out[59]: <AxesSubplot:xlabel='population'>



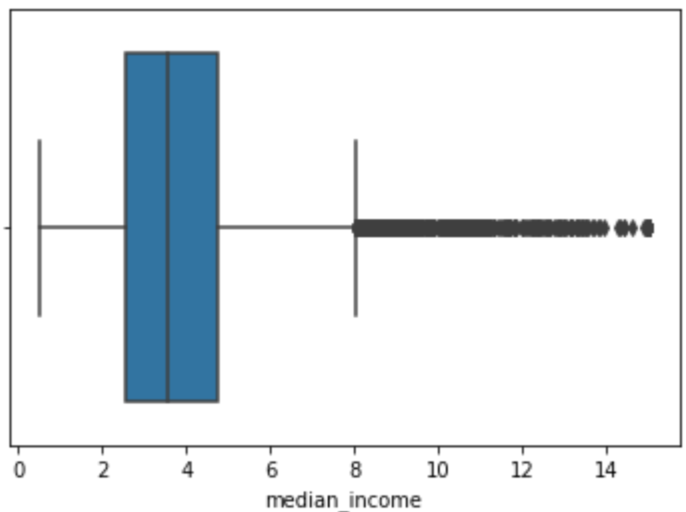
In [60]:
```
sns.boxplot(data['households'])
```

Out[60]: <AxesSubplot:xlabel='households'>



In [61]:
```
sns.boxplot(data['median_income'])
```

Out[61]: <AxesSubplot:xlabel='median_income'>



In [62]:
```
# We see lots of outliers in total_rooms, total_bedrooms, populations,
households, median_income
```

In [ ]:
```
```

1. Encode categorical data :

   *Convert categorical column in the dataset to numerical data.

In [63]:
```python
data['ocean_proximity'].value_counts()
```

Out[63]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [64]:
```python
data =pd.get_dummies(columns=['ocean_proximity'], drop_first=True,data=data)
print(data.shape)
data.head()
```

```
(20640, 13)
```

Out[64]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
0	-122.23	37.88	41	880	129.0	322	·
1	-122.22	37.86	21	7099	1106.0	2401	1·
2	-122.24	37.85	52	1467	190.0	496	·
3	-122.25	37.85	52	1274	235.0	558	:
4	-122.25	37.85	52	1627	280.0	565	:

In [65]:
```
import statsmodels.formula.api as smf
sm_sales_tv_model =smf.ols(formula = 'median_house_value ~ longitude+la
titude+housing_median_age+total_rooms+total_bedrooms+population+househo
lds + median_income',data=data).fit()
sm_sales_tv_model.summary()
```

Out[65]:

OLS Regression Results

Dep. Variable:	median_house_value	**R-squared:**	0.636
Model:	OLS	**Adj. R-squared:**	0.635
Method:	Least Squares	**F-statistic:**	4499.
Date:	Tue, 09 Nov 2021	**Prob (F-statistic):**	0.00
Time:	08:40:47	**Log-Likelihood:**	-2.5945e+05
No. Observations:	20640	**AIC:**	5.189e+05
Df Residuals:	20631	**BIC:**	5.190e+05
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-3.571e+06	6.26e+04	-57.035	0.000	-3.69e+06	-3.45e+06
longitude	-4.262e+04	714.112	-59.685	0.000	-4.4e+04	-4.12e+04
latitude	-4.248e+04	674.532	-62.976	0.000	-4.38e+04	-4.12e+04
housing_median_age	1144.4786	43.236	26.471	0.000	1059.733	1229.224
total_rooms	-6.6806	0.775	-8.621	0.000	-8.199	-5.162
total_bedrooms	82.4636	6.047	13.636	0.000	70.610	94.317
population	-39.8333	1.072	-37.164	0.000	-41.934	-37.732
households	78.1123	6.784	11.514	0.000	64.815	91.410
median_income	3.977e+04	331.942	119.815	0.000	3.91e+04	4.04e+04

Omnibus:	5040.064	**Durbin-Watson:**	0.965
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18972.448
Skew:	1.184	**Prob(JB):**	0.00
Kurtosis:	7.056	**Cond. No.**	5.09e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.09e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

1. Split the dataset :

   *Split the data into 80% training dataset and 20% test dataset.

In [66]:
```
from sklearn.model_selection import train_test_split
```

In [67]:
```
X = data.drop('median_house_value', axis=1)
y = data['median_house_value']
```

In [68]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [69]:
```
X_Scaled = scaler.fit_transform(X)
```

In [70]:
```
print(X_Scaled.shape)
print(y.shape)
```

```
(20640, 12)
(20640,)
```

In [71]:
```
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y , train
_size=0.8, random_state=1)
```

In [72]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(16512, 12)
(16512,)
(4128, 12)
(4128,)
```

In [ ]:

1. Perform Linear Regression :

   Perform Linear Regression on training data. Predict output for test dataset using the fitted model. *Print root mean squared error (RMSE) from Linear Regression.

   ```
   *[ HINT: Import mean_squared_error from sklearn.metrics ]
   ```

In [73]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [74]:
```
lr_model = LinearRegression()
```

In [75]:
```
lr_model.fit(X_train, y_train)
```
Out[75]:
```
LinearRegression()
```

In [76]:
```
lr_y_pred = lr_model.predict(X_test)
```

In [77]:
```
print(r2_score(lr_y_pred, y_test))
print(mean_squared_error(lr_y_pred, y_test))
print(np.sqrt(mean_squared_error(lr_y_pred, y_test)))
```
```
0.47099130550835133
4754050720.172421
68949.62451074278
```

In [78]:
```
lr_y_pred
```
Out[78]:
```
array([243626.05897437,  93442.21732292, 247630.00077113, ...,
       284944.56343132, 266458.51436442, 137361.4394673 ])
```

In [ ]:
```

```

1. Perform Decision Tree Regression :

- Perform Decision Tree Regression on training data.
- Predict output for test dataset using the fitted model.
- Print root mean squared error from Decision Tree Regression.

In [79]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```
Out[79]:
```
DecisionTreeClassifier()
```

In [80]:
```
y_predict = classifier.predict(X_test)
```

In [81]:
```
print(r2_score(y_predict, y_test))
print(mean_squared_error(y_predict, y_test))
print(np.sqrt(mean_squared_error(y_predict, y_test)))
```
```
0.5346428836334868
6100414232.157219
78105.14856369725
```

In [82]:
```
y_predict
```
Out[82]:
```
array([234100,  69400, 173400, ..., 212500, 351100,  82200])
```

In [ ]:
```

```

1. Perform Random Forest Regression :

- Perform Random Forest Regression on training data.
- Predict output for test dataset using the fitted model.
- Print RMSE (root mean squared error) from Random Forest Regression.

In [83]:
```
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y , train
_size=0.5, random_state=1)
```

In [84]:
```
from sklearn.ensemble import RandomForestClassifier

RF_classifier_scaled = RandomForestClassifier(30)
RF_classifier_scaled.fit(X_train, y_train)
```

Out[84]:
```
RandomForestClassifier(n_estimators=30)
```

In [85]:
```
rf_y_predict = RF_classifier_scaled.predict(X_test)
```

In [86]:
```
print(r2_score(rf_y_predict, y_test))
print(mean_squared_error(rf_y_predict, y_test))
print(np.sqrt(mean_squared_error(rf_y_predict, y_test)))
```

```
0.6421264811573941
5276055553.713857
72636.46159962541
```

In [87]:
```
# Random Forest Regressor gave comparitively better result compared to
Linear regression or Decision tree regression
```

1. Bonus exercise: Perform Linear Regression with one independent variable :

- Extract just the median_income column from the independent variables (from X_train and X_test).
- Perform Linear Regression to predict housing values based on median_income.
- Predict output for test dataset using the fitted model.
- Plot the fitted model for training data as well as for test data to check if the fitted model satisfies the test data.

In [88]:
```
data.head()
```

Out[88]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
0	-122.23	37.88	41	880	129.0	322	·
1	-122.22	37.86	21	7099	1106.0	2401	1·
2	-122.24	37.85	52	1467	190.0	496	·
3	-122.25	37.85	52	1274	235.0	558	:
4	-122.25	37.85	52	1627	280.0	565	:

In [89]:
```
X = data[['median_income']]
X.head()
```

Out[89]:

	median_income
0	8.3252
1	8.3014
2	7.2574
3	5.6431
4	3.8462

In [90]:
```
y = data[['median_house_value']]
y.head()
```

Out[90]:

	median_house_value
0	452600
1	358500
2	352100
3	341300
4	342200

In [91]:
```
print(X.shape)
print(y.shape)
```

```
(20640, 1)
(20640, 1)
```

In [92]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,
random_state=1)
```

In [93]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Out[93]:
```
LinearRegression()
```

In [94]:
```
y_lr_pred = lr.predict(X_test)
```

In [95]:
```
y_lr_pred[0:5]
```

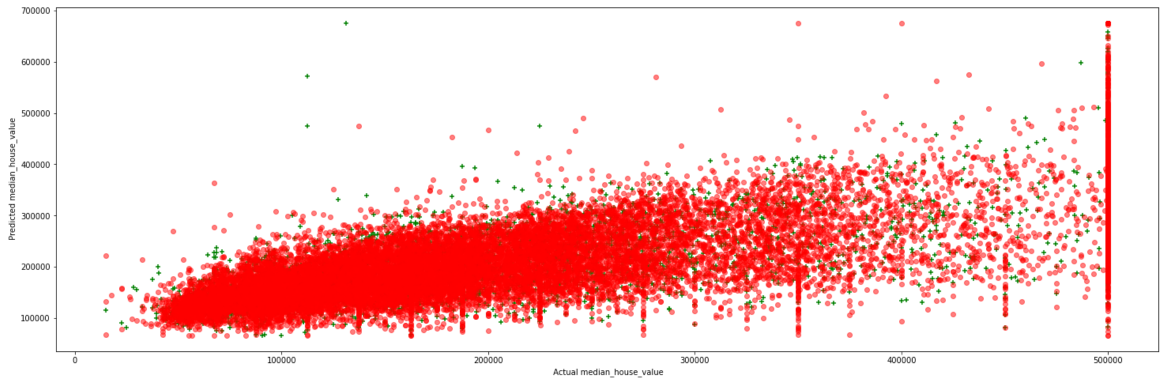
Out[95]:
```
array([[181402.07011843],
       [127924.35050918],
       [213498.79519375],
       [108679.77321036],
       [262548.07514048]])
```

In [96]:
```
print(r2_score(y_lr_pred, y_test))
print(mean_squared_error(y_lr_pred, y_test))
print(np.sqrt(mean_squared_error(y_lr_pred, y_test)))
```

```
-0.048968356140921765
6926929696.091081
83228.17849797675
```

In [97]:
```
y_lr_pred_train = lr.predict(X_train)
```

In [98]:
```
fig = plt.figure(figsize=(25,8))
plt.scatter(y_test,y_lr_pred, color='green', marker='+')
plt.scatter(y_train,y_lr_pred_train, color='red', alpha=0.5, marker='o
')
plt.xlabel(" Actual median_house_value")
plt.ylabel(" Predicted median_house_value")
plt.show()
```



We see lots of outliers in total_rooms, total_bedrooms, populations,households, median_income and due to which the data is highly imbalance

In [ ]:

```

```