Data Cleaning and Exploration

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [42]:
```
data = pd.read_csv("housing.csv")
```

In [43]:
```
data
```

Out[43]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20640 rows × 10 columns

In [44]:
```
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  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 [45]:
```
#drop null value
data.dropna(inplace=True)
```

In [46]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20433 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20433 non-null  float64
 1   latitude            20433 non-null  float64
 2   housing_median_age  20433 non-null  float64
 3   total_rooms         20433 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20433 non-null  float64
 6   households          20433 non-null  float64
 7   median_income       20433 non-null  float64
 8   median_house_value  20433 non-null  float64
 9   ocean_proximity     20433 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.7+ MB
```

In [47]:
```
from sklearn.model_selection import train_test_split

X = data.drop(['median_house_value'], axis =1)
y = data['median_house_value'] #Target variable as median_house_value
```

In [48]:
```
X
```

Out[48]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 9 columns

In [49]:
```
y
```

Out[49]:
```
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: 20433, dtype: float64
```

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2)
```

In [51]:
```
train_data = X_train.join(y_train) # combine training data for x and y
```

In [52]:
```
train_data
```

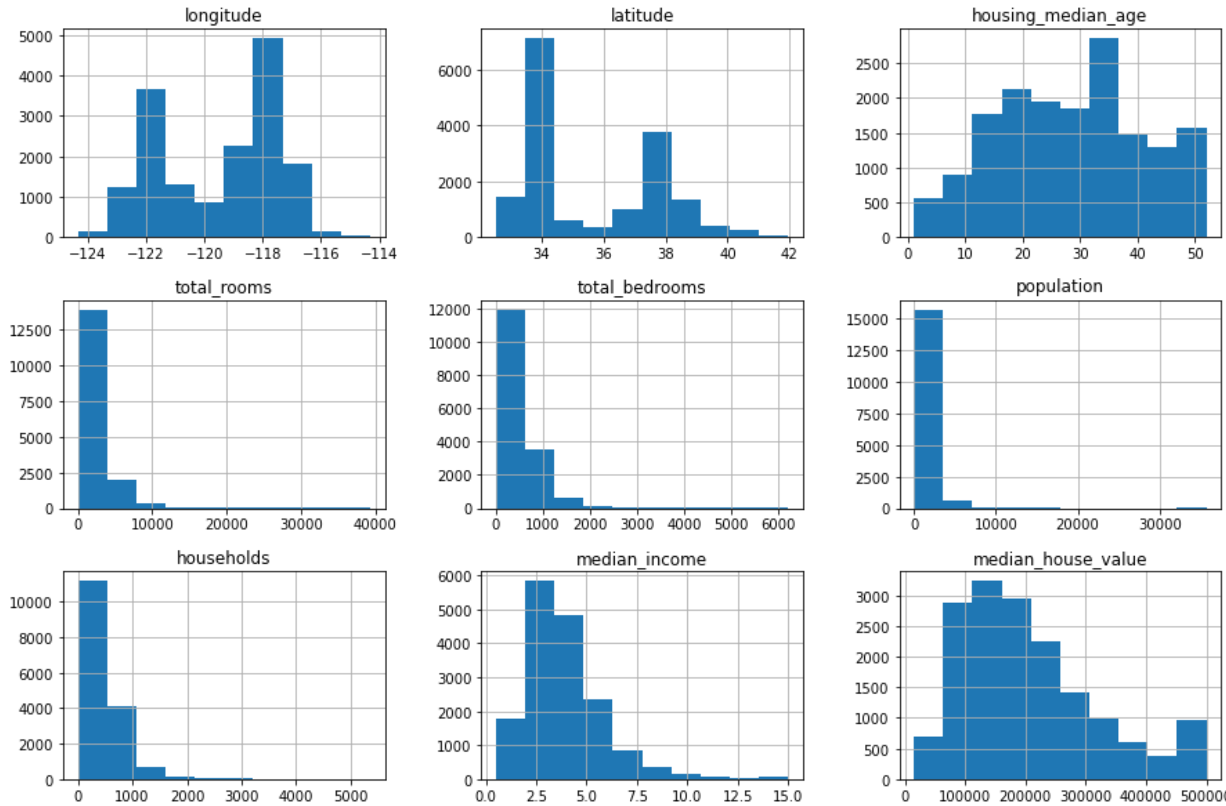
Out[52]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
6189	-117.89	34.10	27.0	3341.0	728.0	1762.0	
7889	-118.05	33.87	18.0	4928.0	773.0	2952.0	
17167	-122.25	37.45	34.0	2999.0	365.0	927.0	
19769	-122.11	39.82	27.0	1065.0	214.0	508.0	
2809	-119.02	35.42	36.0	2044.0	447.0	1021.0	
...	...	...	...	...	...	...	
4704	-118.34	34.05	52.0	2530.0	458.0	1122.0	
9775	-121.24	36.33	13.0	1642.0	418.0	1534.0	
13482	-117.35	34.12	22.0	5640.0	889.0	3157.0	
3957	-118.59	34.21	17.0	2737.0	868.0	2924.0	
4666	-118.29	34.05	30.0	1417.0	589.0	1615.0	

16346 rows × 10 columns

In [53]:
```
#train_data Histogram
train_data.hist(figsize =(15,10))
```

Out[53]:
```
array([[<AxesSubplot:title={'center':'longitude'}>,
        <AxesSubplot:title={'center':'latitude'}>,
        <AxesSubplot:title={'center':'housing_median_age'}>],
       [<AxesSubplot:title={'center':'total_rooms'}>,
        <AxesSubplot:title={'center':'total_bedrooms'}>,
        <AxesSubplot:title={'center':'population'}>],
       [<AxesSubplot:title={'center':'households'}>,
        <AxesSubplot:title={'center':'median_income'}>,
        <AxesSubplot:title={'center':'median_house_value'}>]],
      dtype=object)
```



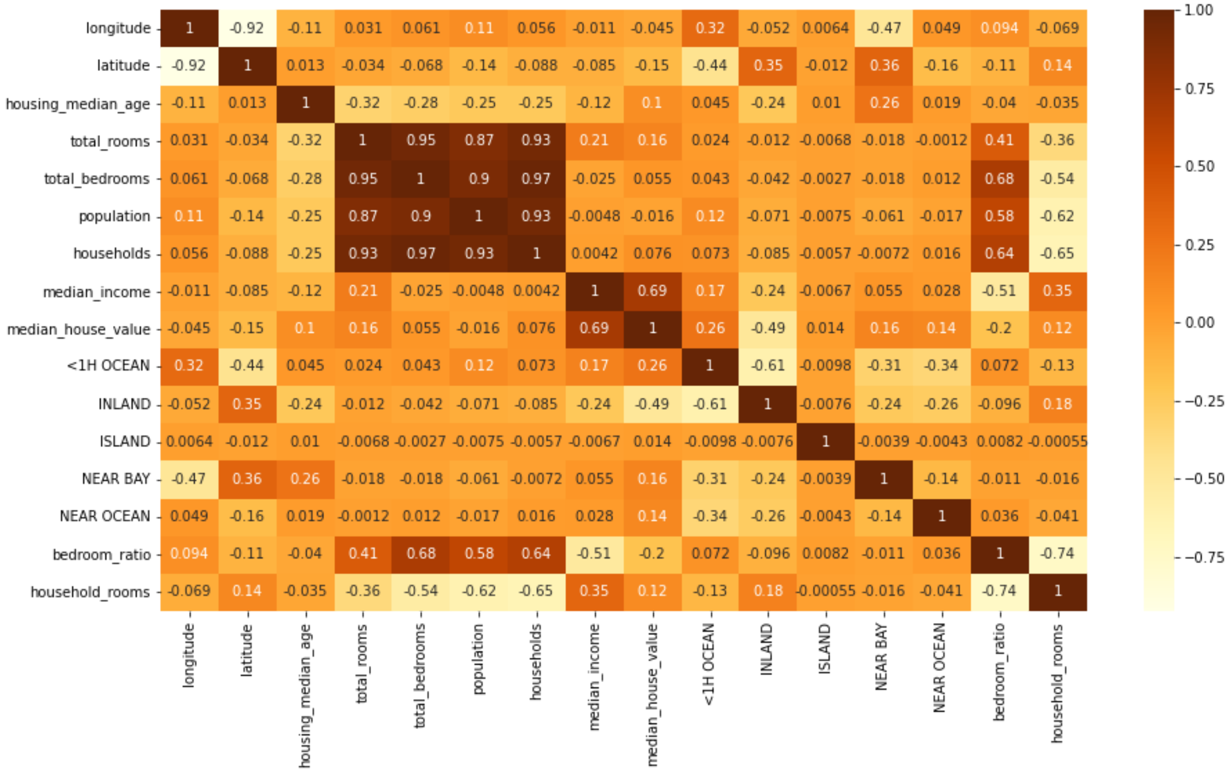
In [54]:
```
train_data.corr()
```

Out[54]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
longitude	1.000000	-0.925058	-0.114327	0.052139	0.075867
latitude	-0.925058	1.000000	0.018055	-0.043112	-0.073318
housing_median_age	-0.114327	0.018055	1.000000	-0.361972	-0.321216
total_rooms	0.052139	-0.043112	-0.361972	1.000000	0.929340
total_bedrooms	0.075867	-0.073318	-0.321216	0.929340	1.000000
population	0.105464	-0.113983	-0.298673	0.856049	0.876083
households	0.061947	-0.077307	-0.304371	0.917310	0.979464
median_income	-0.012824	-0.080792	-0.120067	0.199183	-0.007711
median_house_value	-0.041273	-0.147505	0.105700	0.134888	0.050491

In [223…
```
plt.figure(figsize = (15,8))
sns.heatmap(train_data.corr(),annot= True, cmap="YlOrBr")
```

Out[223]: <AxesSubplot:>



The correlation graph shows the variables' level of dependency

Conclusion

1. Correlation of median_income and median_house_value.
2. Latitude is negatively correlating with the house value.

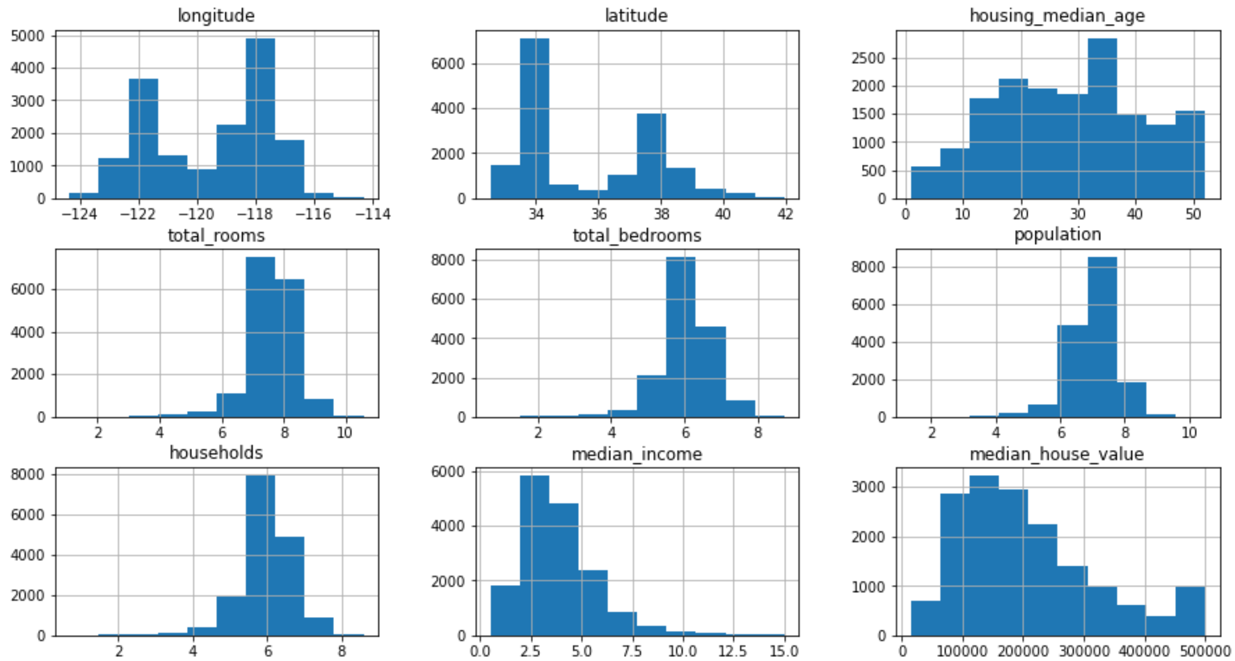
Data Preprocessing

Converting right skewed graphs to a gaussian curve by logarithm value.

In [56]:
```
train_data['total_rooms'] = np.log(train_data['total_rooms']+1) # +1 is to prev
train_data['total_bedrooms'] = np.log(train_data['total_bedrooms']+1)
train_data['population'] = np.log(train_data['population']+1)
train_data['households'] = np.log(train_data['households']+1)
```

In [57]:
```
train_data.hist(figsize = (15,8))
```

Out[57]:
```
array([[<AxesSubplot:title={'center':'longitude'}>,
        <AxesSubplot:title={'center':'latitude'}>,
        <AxesSubplot:title={'center':'housing_median_age'}>],
       [<AxesSubplot:title={'center':'total_rooms'}>,
        <AxesSubplot:title={'center':'total_bedrooms'}>,
        <AxesSubplot:title={'center':'population'}>],
       [<AxesSubplot:title={'center':'households'}>,
        <AxesSubplot:title={'center':'median_income'}>,
        <AxesSubplot:title={'center':'median_house_value'}>]],
      dtype=object)
```



Add ocean_proximity because the closer the house is to the ocean the more price the house would have.

In [58]:
```
train_data.ocean_proximity.value_counts()
```

Out[58]:
```
<1H OCEAN     7209
INLAND        5197
NEAR OCEAN    2116
NEAR BAY      1821
ISLAND           3
Name: ocean_proximity, dtype: int64
```

Conversion of ocean_proximity to one hot encoding with binary variable of 0 Or 1, to improve prediction accuracy

In [59]:
```
pd.get_dummies(train_data.ocean_proximity)
```

Out[59]:

	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
6189	1	0	0	0	0
7889	1	0	0	0	0
17167	0	0	0	0	1
19769	0	1	0	0	0
2809	0	1	0	0	0
...	...	...	...	...	...
4704	1	0	0	0	0
9775	1	0	0	0	0
13482	0	1	0	0	0
3957	1	0	0	0	0
4666	1	0	0	0	0

16346 rows × 5 columns

In [60]:
```
train_data = train_data.join(pd.get_dummies(train_data.ocean_proximity)).drop([
```

In [61]:
```
train_data
```

Out[61]:

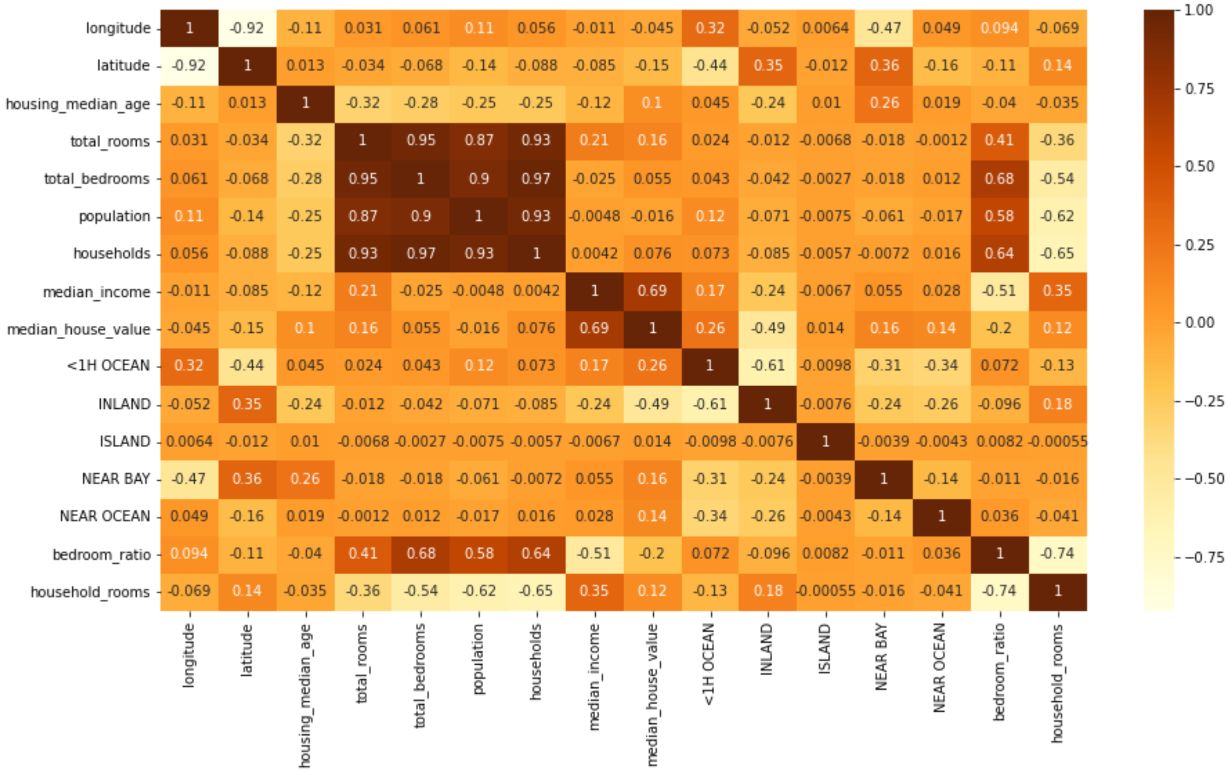
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
6189	-117.89	34.10	27.0	8.114325	6.591674	7.474772	6.
7889	-118.05	33.87	18.0	8.502891	6.651572	7.990577	6
17167	-122.25	37.45	34.0	8.006368	5.902633	6.833032	5
19769	-122.11	39.82	27.0	6.971669	5.370638	6.232448	5.
2809	-119.02	35.42	36.0	7.623153	6.104793	6.929517	5.
...	...	...	...	...	...	...	
4704	-118.34	34.05	52.0	7.836370	6.129050	7.023759	6
9775	-121.24	36.33	13.0	7.404279	6.037871	7.336286	5.
13482	-117.35	34.12	22.0	8.637817	6.791221	8.057694	6
3957	-118.59	34.21	17.0	7.914983	6.767343	7.981050	6.
4666	-118.29	34.05	30.0	7.257003	6.380123	7.387709	6

16346 rows × 14 columns

Checking how the new features correlate with the target variable

In [221…
```
plt.figure(figsize = (15,8))
sns.heatmap(train_data.corr(),annot= True, cmap="YlOrBr")
```

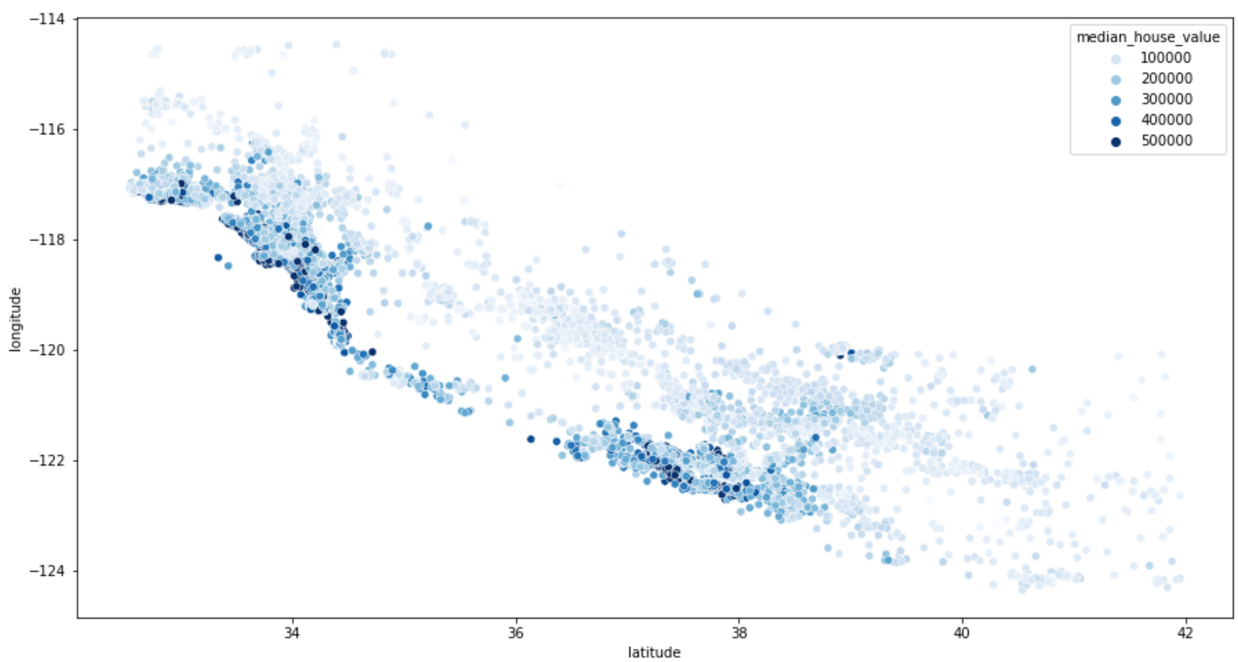
Out[221]: <AxesSubplot:>



In [63]:
```
# 1 hour away from the ocean teh median_house_value is higher than inland
```

In [225…
```
#visualising the coorodinates
plt.figure(figsize = (15,8))
sns.scatterplot(x= 'latitude', y='longitude', data= train_data, hue= 'median_hc
```

Out[225]:
```
<AxesSubplot:xlabel='latitude', ylabel='longitude'>
```



Conclusion

All the houses one hour away from the ocean are likely to be more expensive.

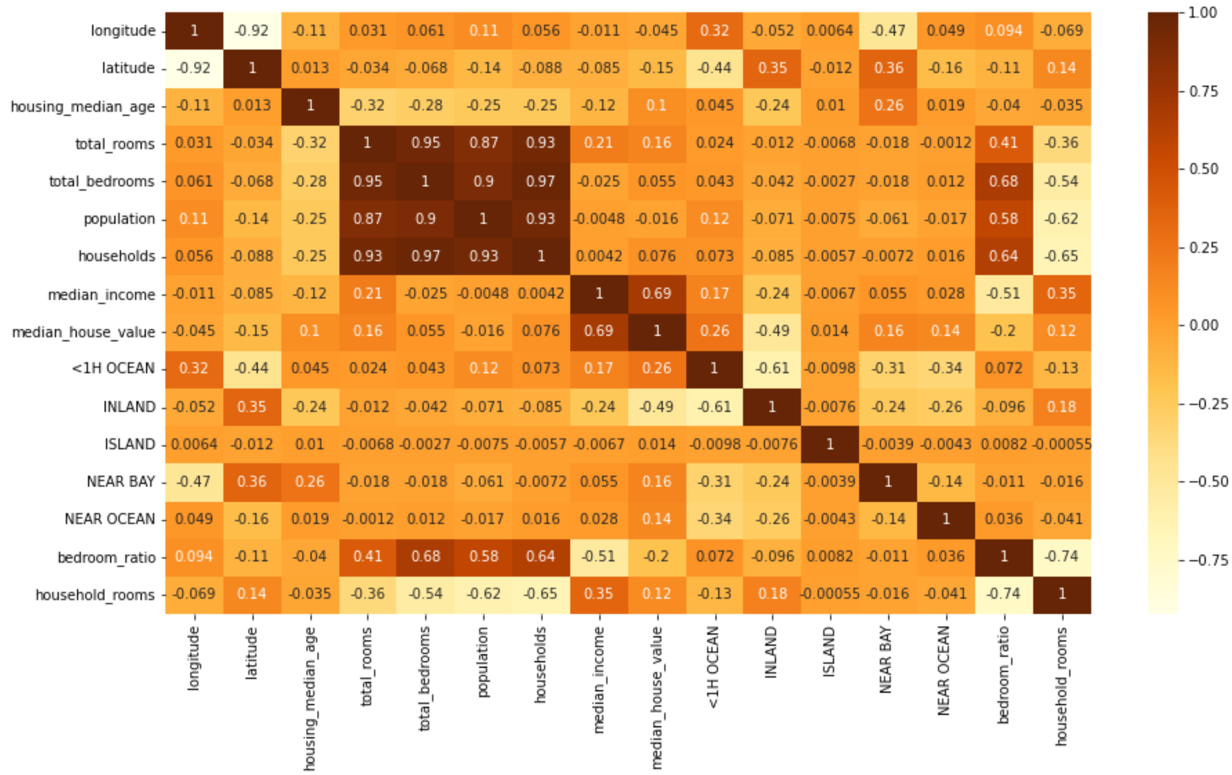
Feature Engineering

Find total bedrooms in the houses and total rooms per household.

In [65]:
```
train_data['bedroom_ratio'] = train_data['total_bedrooms']/ train_data['total_r
train_data['household_rooms'] = train_data["total_rooms"] / train_data['househo
```

In [222…
```
plt.figure(figsize = (15,8))
sns.heatmap(train_data.corr(),annot= True, cmap="YlOrBr")
```

Out[222]: <AxesSubplot:>



Linear Regression Model

In [67]:
```
from sklearn.linear_model import LinearRegression

X_train,y_train = train_data.drop(["median_house_value"], axis =1), train_data[

#creating model
reg = LinearRegression()

#fitting training data
reg.fit(X_train, y_train)
```

Out[67]: LinearRegression()

Following the same procedure as training for the testing data.

In [68]:
```
test_data = X_test.join(y_test) # combined testing data for x and y

test_data['total_rooms'] = np.log(test_data['total_rooms']+1) # +1 is to preven
test_data['total_bedrooms'] = np.log(test_data['total_bedrooms']+1)
test_data['population'] = np.log(test_data['population']+1)
test_data['households'] = np.log(test_data['households']+1)

#join with testing data and dropping ocean_proximity
test_data = test_data.join(pd.get_dummies(test_data.ocean_proximity)).drop(['oc

test_data['bedroom_ratio'] = test_data['total_bedrooms']/ test_data['total_room
test_data['household_rooms'] = test_data["total_rooms"] / test_data['households
```

In [69]:
```
X_test, y_test = test_data.drop(["median_house_value"], axis =1), test_data["me
```

In [70]:
```
reg.score(X_test,y_test) #checking good fit
```

Out[70]:
```
0.6508979007288292
```

The Regression score is less than 0.9, which means that the gap between the actual value and the estimated value is more than what we want. Now, we will use ensemble Random forest model to get a good fit.

Random Forest Model

In [71]:
```
from sklearn.ensemble import RandomForestRegressor

forest= RandomForestRegressor()

forest.fit(X_train,y_train)
```

Out[71]:
```
RandomForestRegressor()
```

In [72]:
```
forest.score(X_test,y_test)
```

Out[72]:
```
0.8102602251684189
```

0.81 value score for random forest gave us a good fit. Trying K-fold cross validatoin and GridSearchCV for hyperparameter tuning to get the best model

In [73]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    "n_estimators":[3,10,30],
    "max_features":[2,4,6,8]
}
grid_search = GridSearchCV(forest, param_grid, cv=5,
                           scoring ="neg_mean_squared_error",
                           return_train_score=True)
grid_search.fit(X_train,y_train)
```

Out[73]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_features': [2, 4, 6, 8],
                         'n_estimators': [3, 10, 30]},
             return_train_score=True, scoring='neg_mean_squared_error')
```

In [74]:
```
grid_search.best_estimator_
```

Out[74]:
```
RandomForestRegressor(max_features=8, n_estimators=30)
```

In [75]:
```
grid_search.best_estimator_.score(X_test,y_test)
```

Out[75]:
```
0.8065677483971704
```

In [ ]: