Exercise 2: Predict house prices using Machine Learning Linear Regression -- [15 Points]

Use the provided "house_data.csv" dataset Import libraries such as numpy, pands, sklearn, matplotlib, seaborn, etc 15 points breakdown will be on:

- [5 Points] data wrangling: reading the dataset, filetring warnings, describing how many columns and rows, finding correlations between variables, missing values and how to address them, outliers, etc
- [5 Points] data analytics: plotting relationships between house value and proximity to ocean, household income, number of bedrooms, house age, etc - use bar charts, histograms, pairplots, etc
- [5 Points] machine learning linear regression: split dataset 70/30 between Train and Test datasets by importing "train_test_split" from "sklearn" library build your linear regression model by using "LinearRegression" to train your model using the Train dataset model the future house predictions by using the Test dataset check your Mean Square Error (RMSE) score. Is it satisfactory? plot your predictions using matplotlib to show combined line charts for Actual and Predicted, and use seaborn "jointplot" to plot Actual and Predicted over a scatter plot

Data Wrangling and data Analytics

In [122]:
```
#Adding Packages and Modules.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
```

In [123]:
```
import warnings
warnings.filterwarnings('ignore')
#filetring warnings
```

In [124]:
```
housing=pd.read_csv("house_data.csv") # Reading the dataset.
```

In [125]:
```
housing.shape # how many columns and rows
```

Out[125]:
```
(5246, 9)
```

In [126]:
```
housing.columns # Column names
```

Out[126]:
```
Index(['sales_id', 'house_median_age', 'total_nr_rooms', 'total_nr_bedrooms',
       'population', 'households', 'median_income', 'house_median_value',
       'proximity_to_ocean'],
      dtype='object')
```

In [127]:
```
housing.head() # top 5 rows
```

Out[127]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households	median_income	house_median_value	proximity_to_ocea
0	1	49	1655	366.0	754	329	1.3750	104900	Near Ba
1	2	51	2665	574.0	1258	536	2.7303	109700	Near Ba
2	3	49	1215	282.0	570	264	1.4861	97200	Near Ba
3	4	48	1798	432.0	987	374	1.0972	104500	Near Ba
4	5	52	1511	390.0	901	403	1.4103	103900	Near Ba

In [128]:
```
housing.info() #checking for null values.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5246 entries, 0 to 5245
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   sales_id            5246 non-null   int64  
 1   house_median_age    5246 non-null   int64  
 2   total_nr_rooms      5246 non-null   int64  
 3   total_nr_bedrooms   5191 non-null   float64
 4   population          5246 non-null   int64  
 5   households          5246 non-null   int64  
 6   median_income       5246 non-null   float64
 7   house_median_value  5246 non-null   int64  
 8   proximity_to_ocean  5246 non-null   object 
dtypes: float64(2), int64(6), object(1)
memory usage: 369.0+ KB
```

In [129]:
```
pd.isnull(housing).sum()  #number of null values.
```

Out[129]:
```
sales_id               0
house_median_age       0
total_nr_rooms         0
total_nr_bedrooms     55
population             0
households             0
median_income          0
house_median_value     0
proximity_to_ocean     0
dtype: int64
```

In [130]:
```
housing.isnull().any()#Check for missing value.
#there are some missing values in total_nr_bedrooms column.
```

Out[130]:
```
sales_id              False
house_median_age      False
total_nr_rooms        False
total_nr_bedrooms      True
population            False
households            False
median_income         False
house_median_value    False
proximity_to_ocean    False
dtype: bool
```

In [131]:

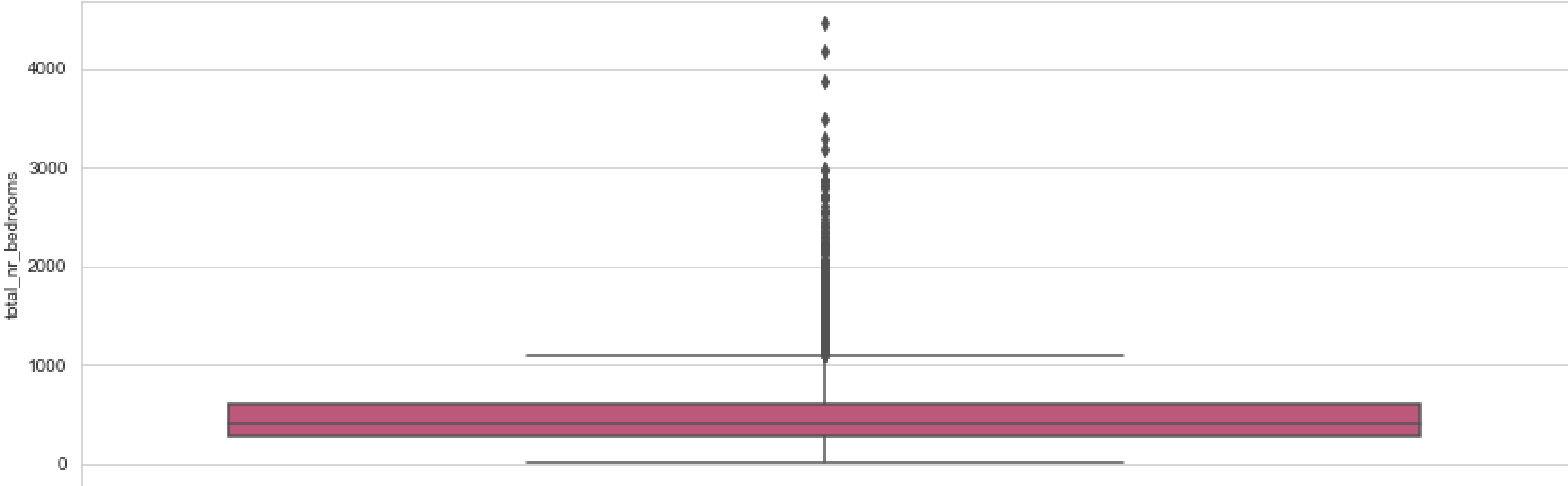
```
housing.describe() #Stats data(Mean,Std,min.....)
```

Out[131]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households	median_income	house_median_value
count	5246.000000	5246.000000	5246.000000	5191.000000	5246.000000	5246.000000	5246.000000	5246.000000
mean	2623.500000	31.309379	2380.551468	508.337893	1331.402402	472.835684	3.495436	181370.856653
std	1514.534087	12.769937	1805.073347	376.150423	941.270206	348.319536	1.972248	114754.900924
min	1.000000	1.000000	2.000000	2.000000	3.000000	2.000000	0.499900	14999.000000
25%	1312.250000	21.000000	1340.000000	287.000000	764.000000	270.000000	2.160075	93800.000000
50%	2623.500000	33.000000	1950.000000	413.000000	1109.000000	386.000000	3.049000	153100.000000
75%	3934.750000	41.000000	2867.750000	612.500000	1637.750000	572.000000	4.268350	229200.000000
max	5246.000000	52.000000	28258.000000	4457.000000	12203.000000	4204.000000	15.000100	500001.000000

In [132]:
```
plt.figure(figsize=(15,5))
sns.boxplot(y='total_nr_bedrooms',data=housing,orient='h',palette='plasma')
plt.plot
#checking for outliers
```

Out[132]: <function matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)>



In [133]:
```
housing['total_nr_bedrooms']=housing['total_nr_bedrooms'].fillna((housing['total_nr_bedrooms'].median()))
# replace null values with median values.
```

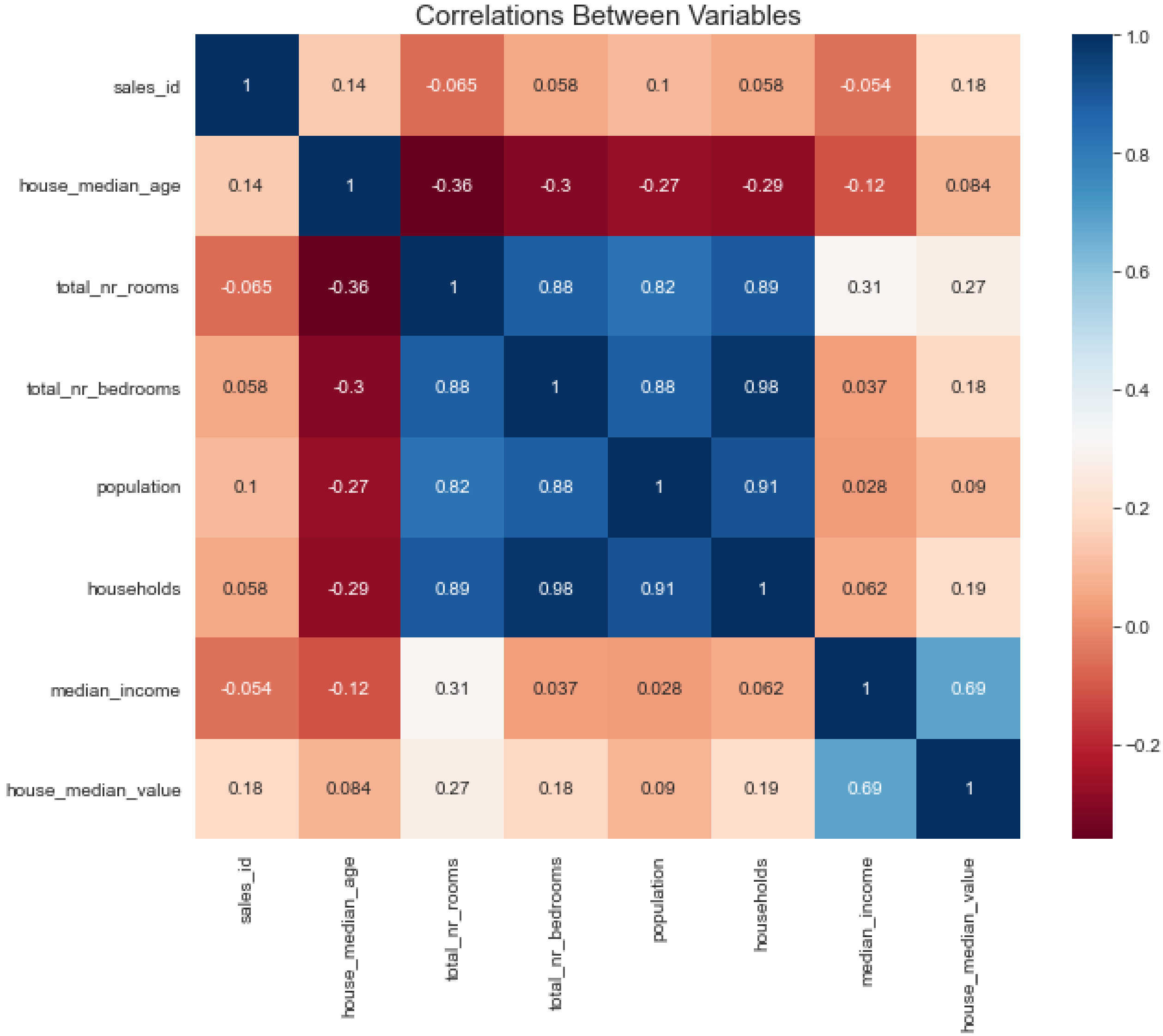
In [134]:
```
housing.describe() #Stats data(Mean,Std,min.....)(The Stats values change as compered to previous values)
```

Out[134]:

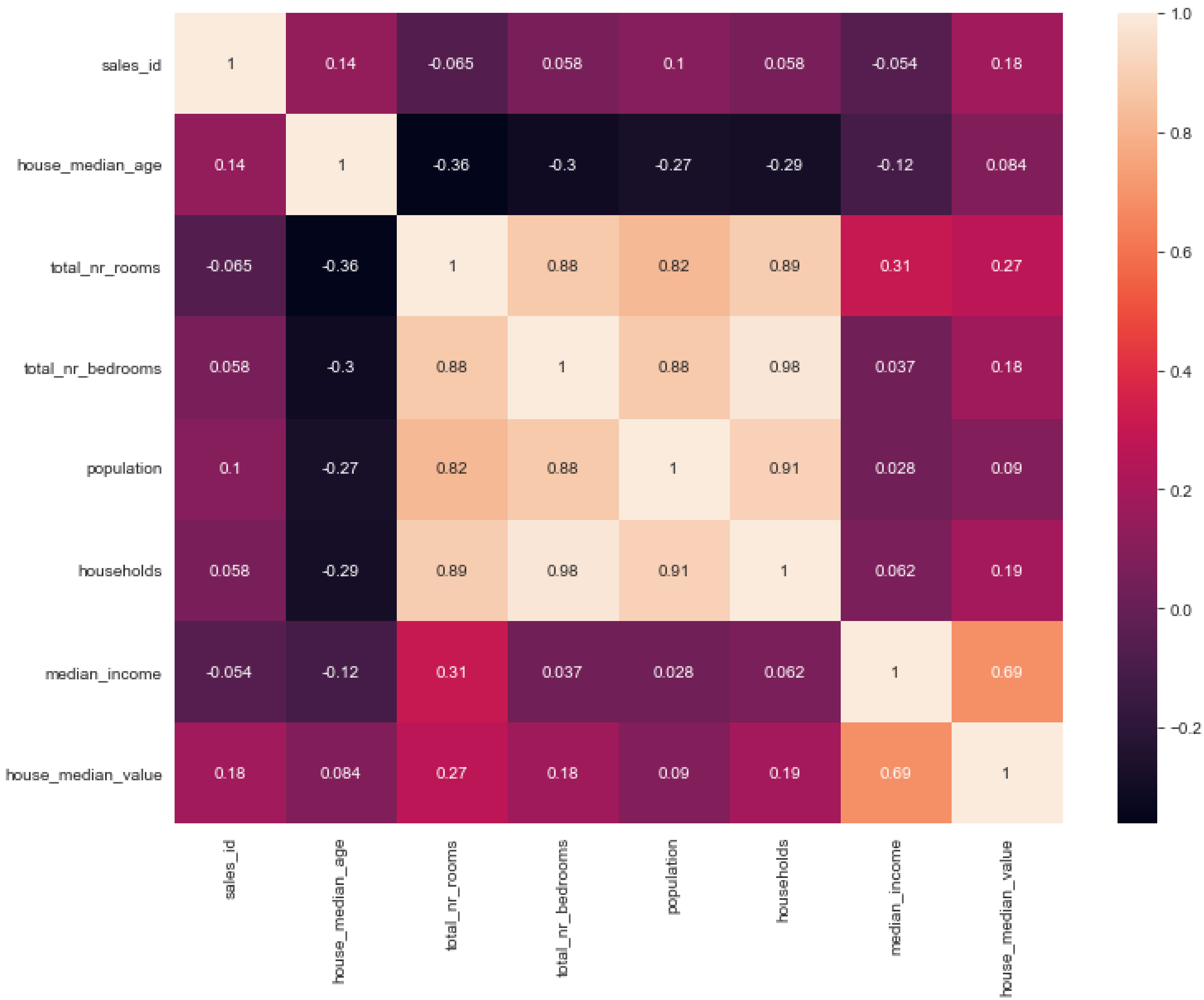
	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households	median_income	house_median_value
count	5246.000000	5246.000000	5246.000000	5246.000000	5246.000000	5246.000000	5246.000000	5246.000000
mean	2623.500000	31.309379	2380.551468	507.338353	1331.402402	472.835684	3.495436	181370.856653
std	1514.534087	12.769937	1805.073347	374.299043	941.270206	348.319536	1.972248	114754.900924
min	1.000000	1.000000	2.000000	2.000000	3.000000	2.000000	0.499900	14999.000000
25%	1312.250000	21.000000	1340.000000	288.000000	764.000000	270.000000	2.160075	93800.000000
50%	2623.500000	33.000000	1950.000000	413.000000	1109.000000	386.000000	3.049000	153100.000000
75%	3934.750000	41.000000	2867.750000	610.000000	1637.750000	572.000000	4.268350	229200.000000
max	5246.000000	52.000000	28258.000000	4457.000000	12203.000000	4204.000000	15.000100	500001.000000

In [135]:

```
plt.figure(figsize=(10,8)) #Correlation matrix all variables.
sns.heatmap(housing.corr(), cmap="RdBu",annot=True)
plt.title("Correlations Between Variables", size=15)
plt.show()
```



In [136]:
```
fig, ax = plt.subplots(figsize=(12,9))
sns.heatmap(housing.corr(), ax=ax,annot=True); #HeatMap
```



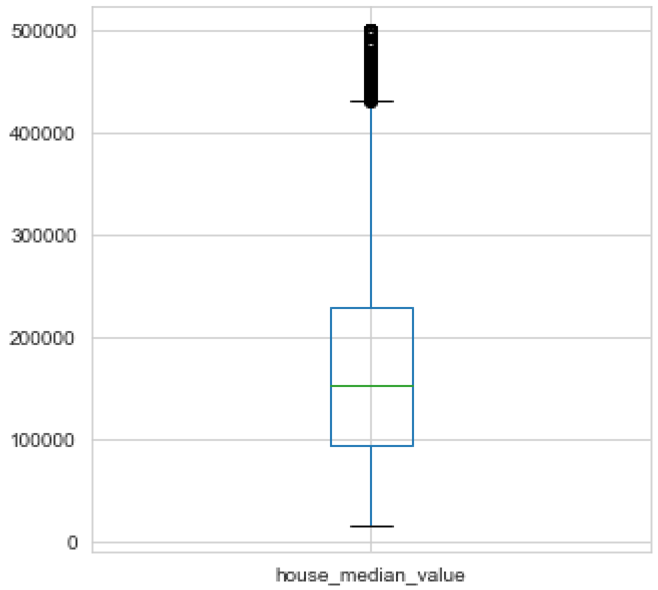
In [137]:
```
housing.corr() #Correaltion of all Columns
```

Out[137]:

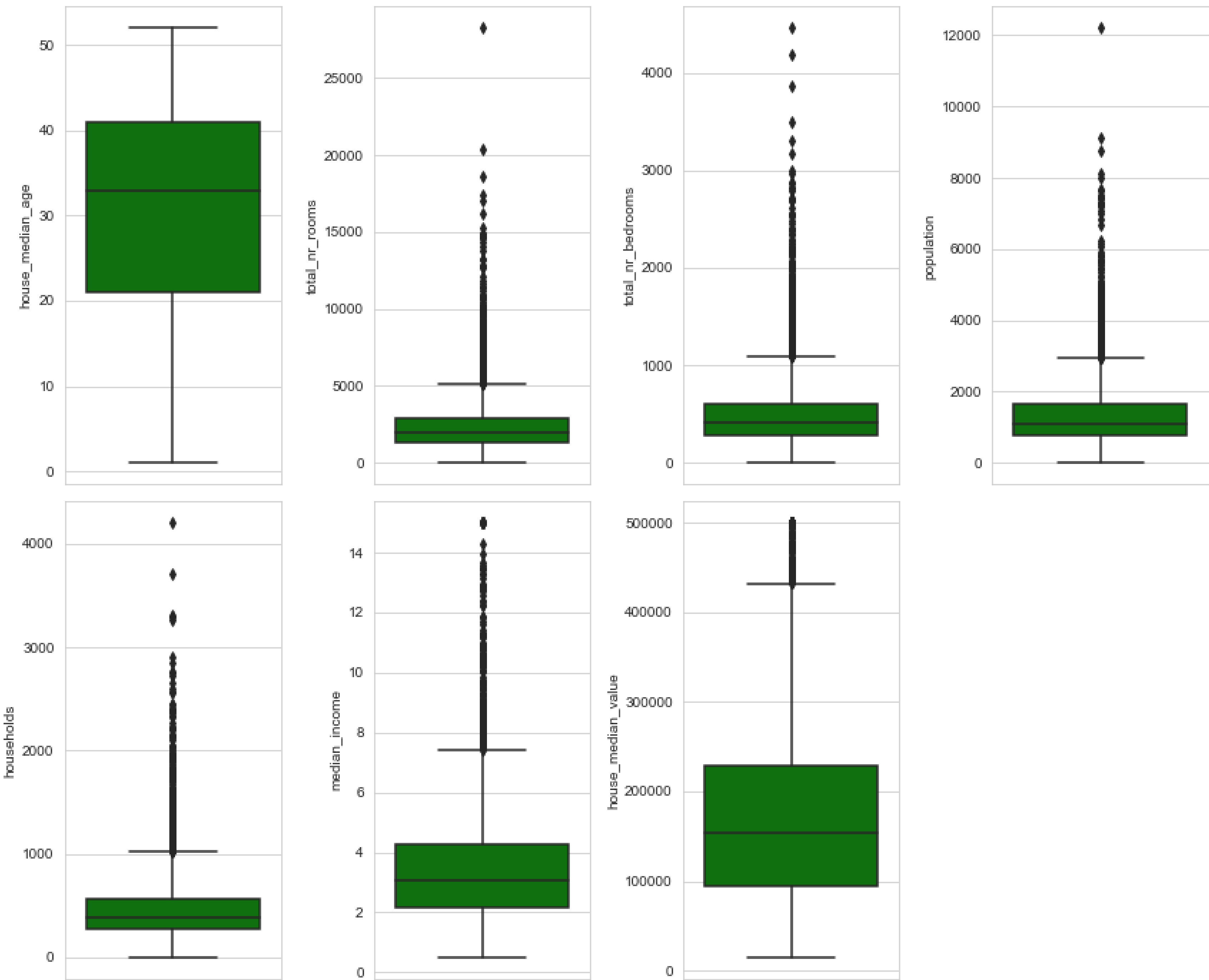
	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households	median_income	house_median_valu
sales_id	1.000000	0.139236	-0.065099	0.057839	0.100802	0.057595	-0.054194	0.18443
house_median_age	0.139236	1.000000	-0.361420	-0.304748	-0.272638	-0.286427	-0.115312	0.08409
total_nr_rooms	-0.065099	-0.361420	1.000000	0.881173	0.819849	0.889199	0.305356	0.26654
total_nr_bedrooms	0.057839	-0.304748	0.881173	1.000000	0.881020	0.981064	0.036796	0.17770
population	0.100802	-0.272638	0.819849	0.881020	1.000000	0.908644	0.028489	0.08976
households	0.057595	-0.286427	0.889199	0.981064	0.908644	1.000000	0.062171	0.19422
median_income	-0.054194	-0.115312	0.305356	0.036796	0.028489	0.062171	1.000000	0.68605
house_median_value	0.184434	0.084097	0.266540	0.177702	0.089762	0.194221	0.686057	1.00000

In [138]:
```
housing.boxplot(["house_median_value"], figsize=(5,5)) #Boxplot.
```

Out[138]: <AxesSubplot:>

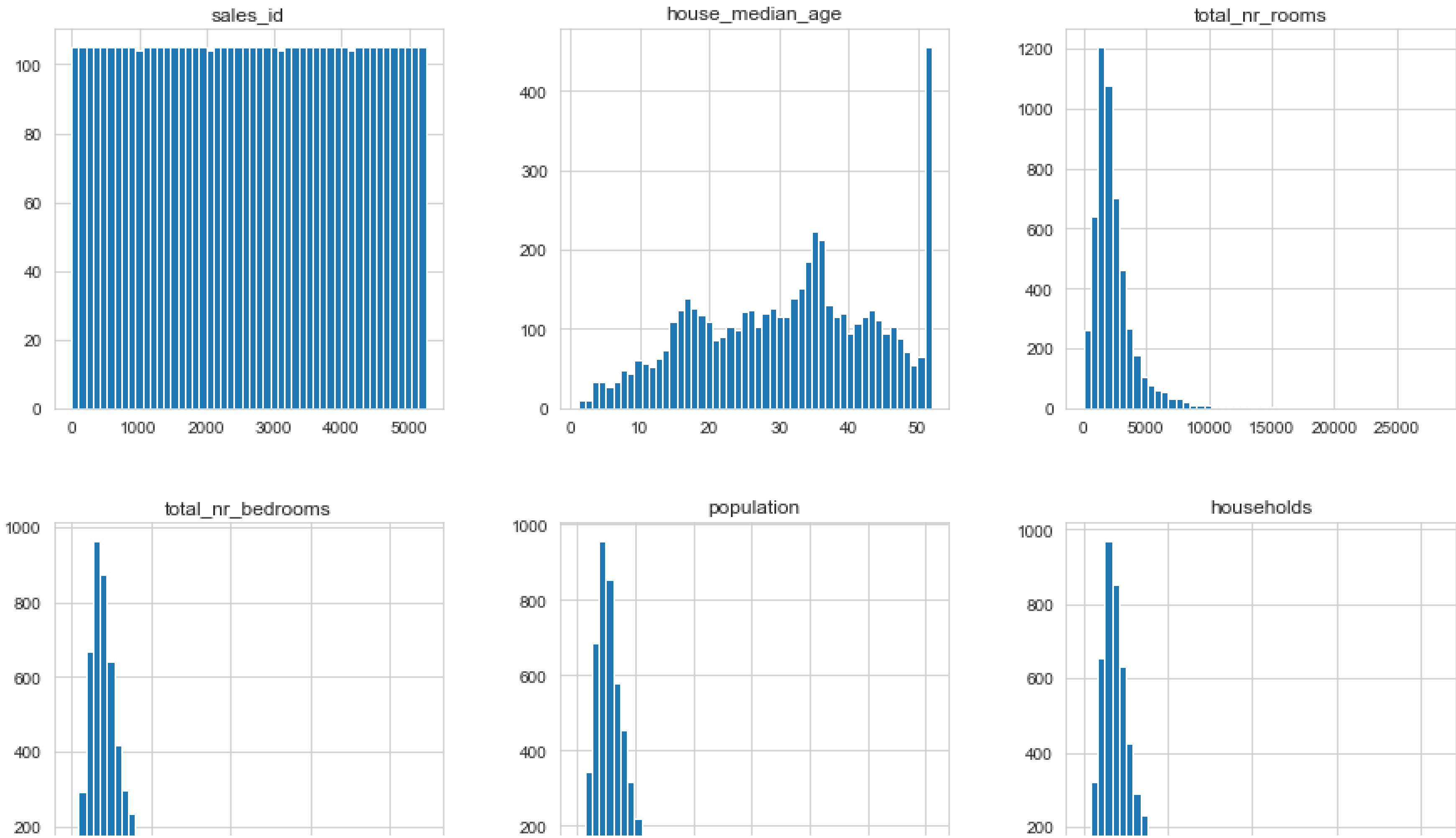


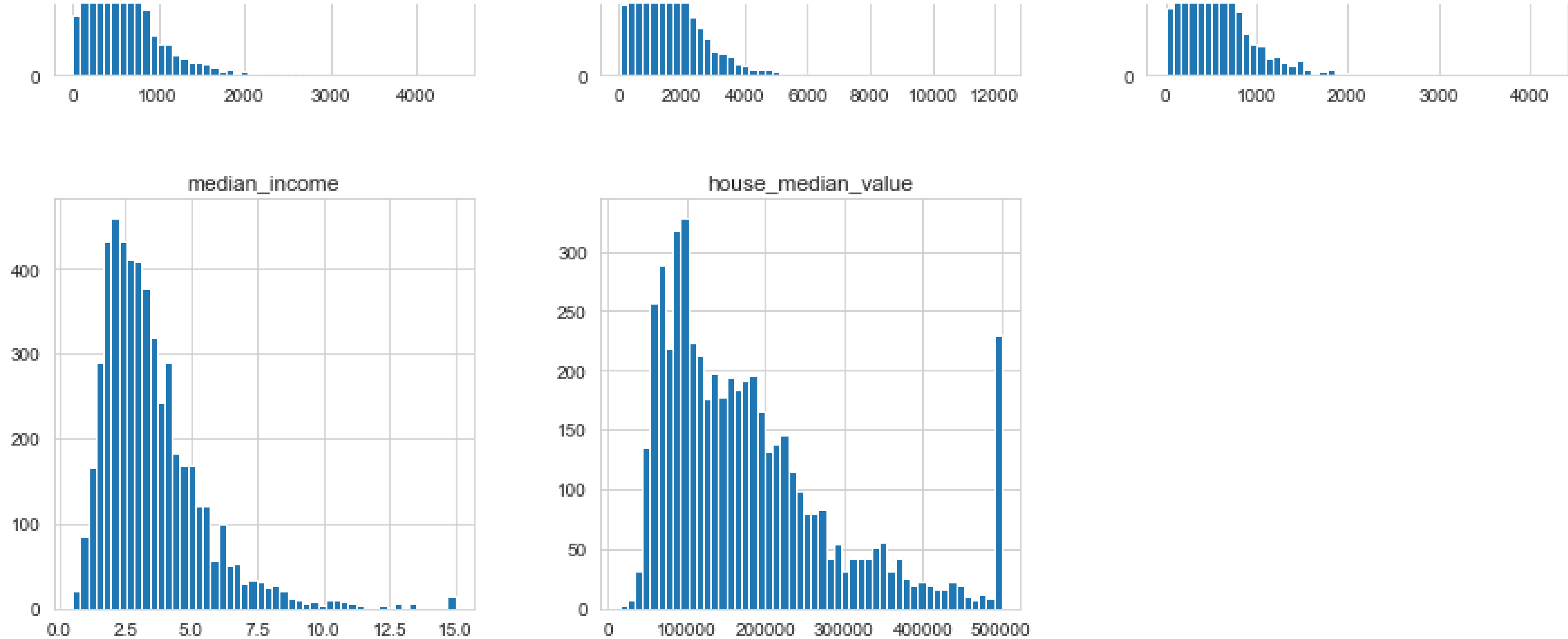
In [139]:
```
columns = ['house_median_age', 'total_nr_rooms', 'total_nr_bedrooms', 'population', 'households','median_income', 'house_
number_of_columns=4
number_of_rows = len(columns)-1/number_of_columns
plt.figure(figsize=(3*number_of_columns,5*number_of_rows))
for i in range(0,len(columns)):
    plt.subplot(number_of_rows + 1,number_of_columns,i+1)
    sns.set_style('whitegrid')
    sns.boxplot(y=columns[i], data=housing,color='green')
    plt.tight_layout()
```


house_median_age
0
10
20
30
40
50
total_nr_rooms
0
5000
10000
15000
20000
25000
total_nr_bedrooms
0
1000
2000
3000
4000
population
0
2000
4000
6000
8000
10000
12000
households
0
1000
2000
3000
4000
median_income
0
2
4
6
8
10
12
14
house_median_value
0
100000
200000
300000
400000
500000

In [140]:
```
housing.hist(bins=50,figsize=(15,15)) #Histogram all of columns.
```

Out[140]:
```
array([[<AxesSubplot:title={'center':'sales_id'}>,
        <AxesSubplot:title={'center':'house_median_age'}>,
        <AxesSubplot:title={'center':'total_nr_rooms'}>],
       [<AxesSubplot:title={'center':'total_nr_bedrooms'}>,
        <AxesSubplot:title={'center':'population'}>,
        <AxesSubplot:title={'center':'households'}>],
       [<AxesSubplot:title={'center':'median_income'}>,
        <AxesSubplot:title={'center':'house_median_value'}>,
        <AxesSubplot:>]], dtype=object)
```



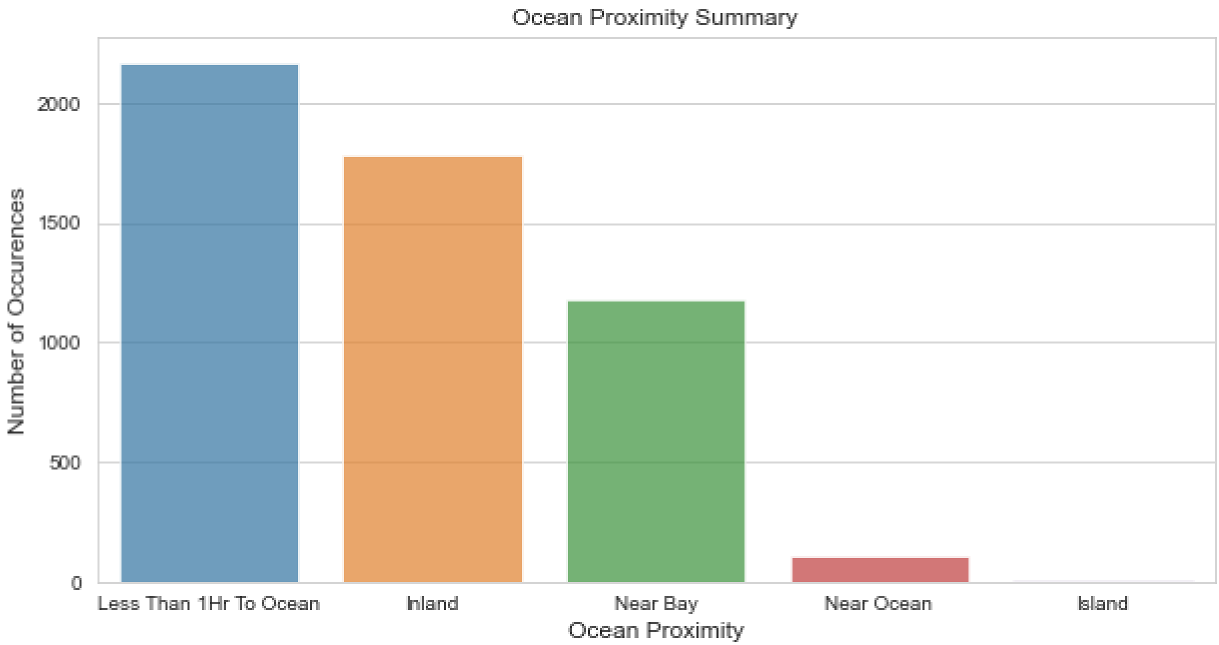


In [141]:
```
housing['proximity_to_ocean'].value_counts() #count value of proximity_to_ocean
```

Out[141]:
```
Less Than 1Hr To Ocean    2167
Inland                    1785
Near Bay                  1182
Near Ocean                 108
Island                       4
Name: proximity_to_ocean, dtype: int64
```

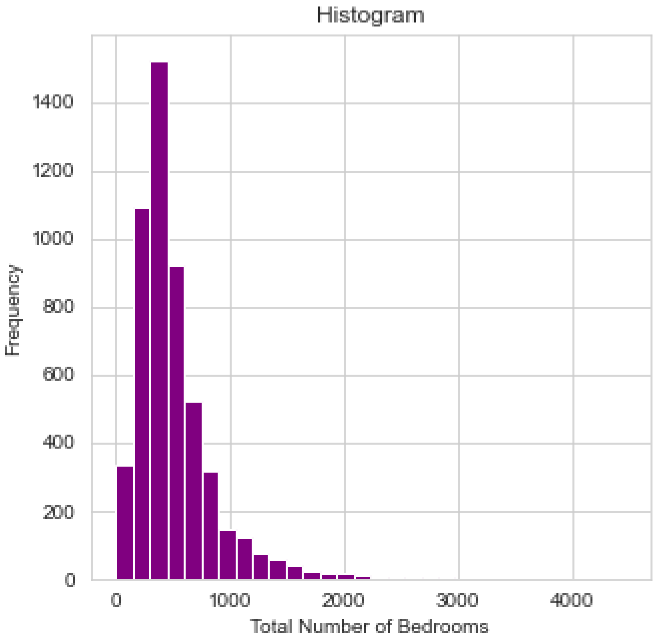
In [142]:
```
op_count=housing['proximity_to_ocean'].value_counts()
plt.figure(figsize=(10,5))
sns.barplot(op_count.index,op_count.values,alpha=0.7)
plt.title('Ocean Proximity Summary')
plt.ylabel("Number of Occurences",fontsize=12)
plt.xlabel("Ocean Proximity",fontsize=12)
plt.show()

#Barchart for proximity to ocean.
```



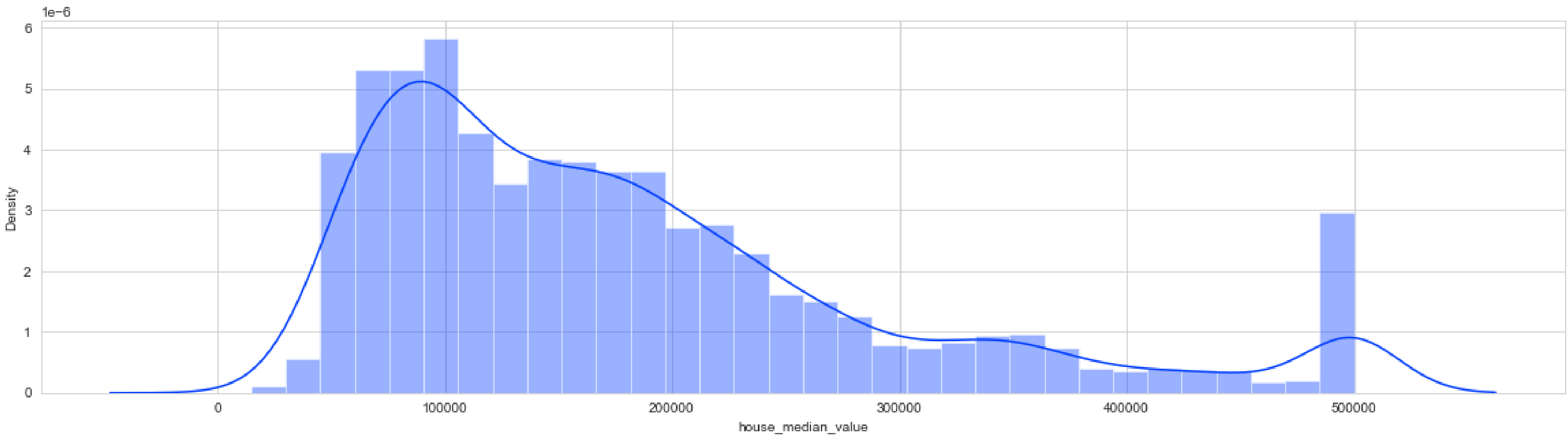
In [143]:
```
plt.figure(figsize=(5,5))
plt.hist(housing[housing['total_nr_bedrooms'].notnull()]['total_nr_bedrooms'],bins=30,color='purple')
plt.title("Histogram")
plt.xlabel("Total Number of Bedrooms")
plt.ylabel("Frequency")
```

Out[143]:
```
Text(0, 0.5, 'Frequency')
```



In [144]:
```
plt.figure(figsize=(20,5))
sns.set_color_codes(palette='bright')
sns.distplot(housing['house_median_value'],color='b')
#distplot to check for outliers.
```

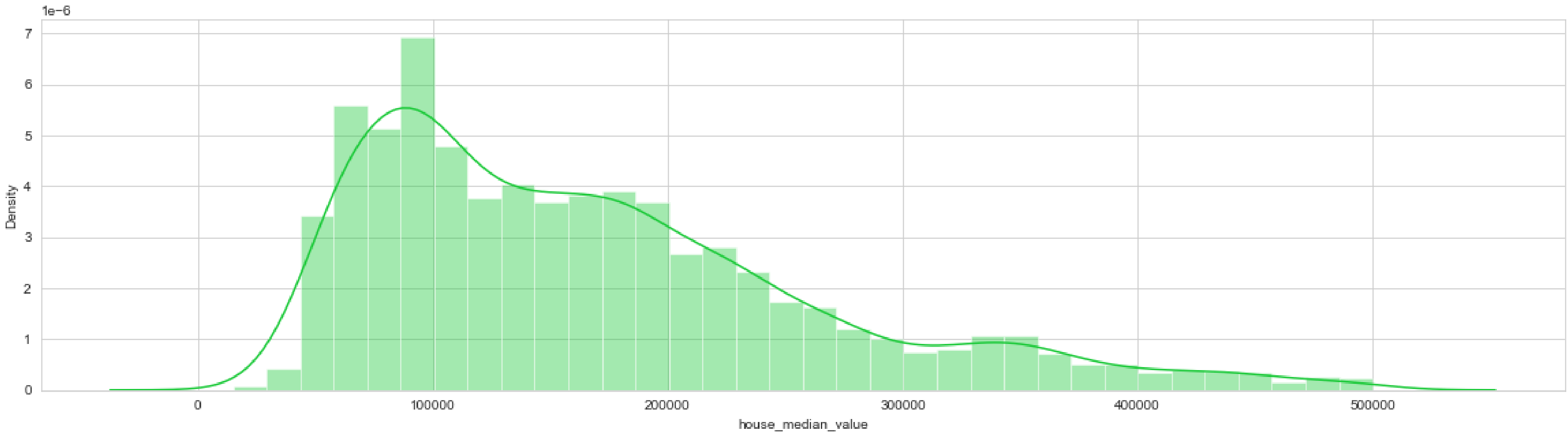
Out[144]: <AxesSubplot:xlabel='house_median_value', ylabel='Density'>



In [145]:
```
housing[housing['house_median_value']>300000]['house_median_value'].value_counts().head(10)
housing1=housing.loc[housing['house_median_value']<500001,:] ##creating new dataset housing1 with better coulmn values
#To better the outliers.
#In the graph we can see that there is outlier value 500000.
```

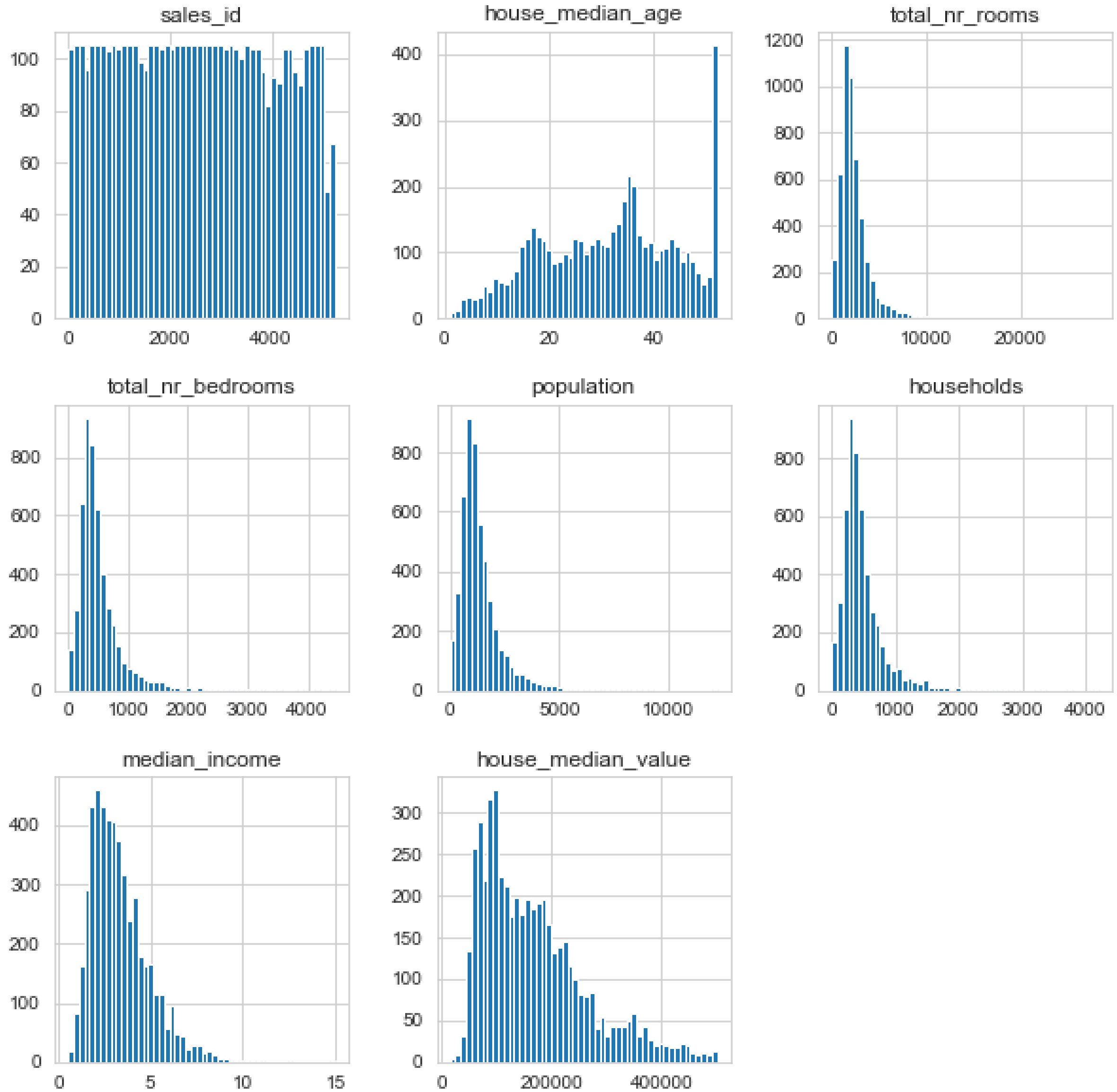
In [146]:
```
plt.figure(figsize=(20,5))
sns.set_color_codes(palette="bright")
sns.distplot(housing1['house_median_value'],color='g')
```

Out[146]: <AxesSubplot:xlabel='house_median_value', ylabel='Density'>



In [147]:
```
housing1.hist(bins=50,figsize=(10,10))
```

Out[147]:
```
array([[<AxesSubplot:title={'center':'sales_id'}>,
        <AxesSubplot:title={'center':'house_median_age'}>,
        <AxesSubplot:title={'center':'total_nr_rooms'}>],
       [<AxesSubplot:title={'center':'total_nr_bedrooms'}>,
        <AxesSubplot:title={'center':'population'}>,
        <AxesSubplot:title={'center':'households'}>],
       [<AxesSubplot:title={'center':'median_income'}>,
        <AxesSubplot:title={'center':'house_median_value'}>,
        <AxesSubplot:>]], dtype=object)
```



In [148]:
```
housing1['total rooms per households']=housing1['total_nr_rooms']/housing1['households']
# creating a new column for better ML Train
```

In [149]:
```
housing1.head()
```

Out[149]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households	median_income	house_median_value	proximity_to_ocea
0	1	49	1655	366.0	754	329	1.3750	104900	Near Ba
1	2	51	2665	574.0	1258	536	2.7303	109700	Near Ba
2	3	49	1215	282.0	570	264	1.4861	97200	Near Ba
3	4	48	1798	432.0	987	374	1.0972	104500	Near Ba
4	5	52	1511	390.0	901	403	1.4103	103900	Near Ba

In [150]:
```
housing1.drop('total_nr_rooms', axis=1, inplace=True) #removing total_nr_rooms
```

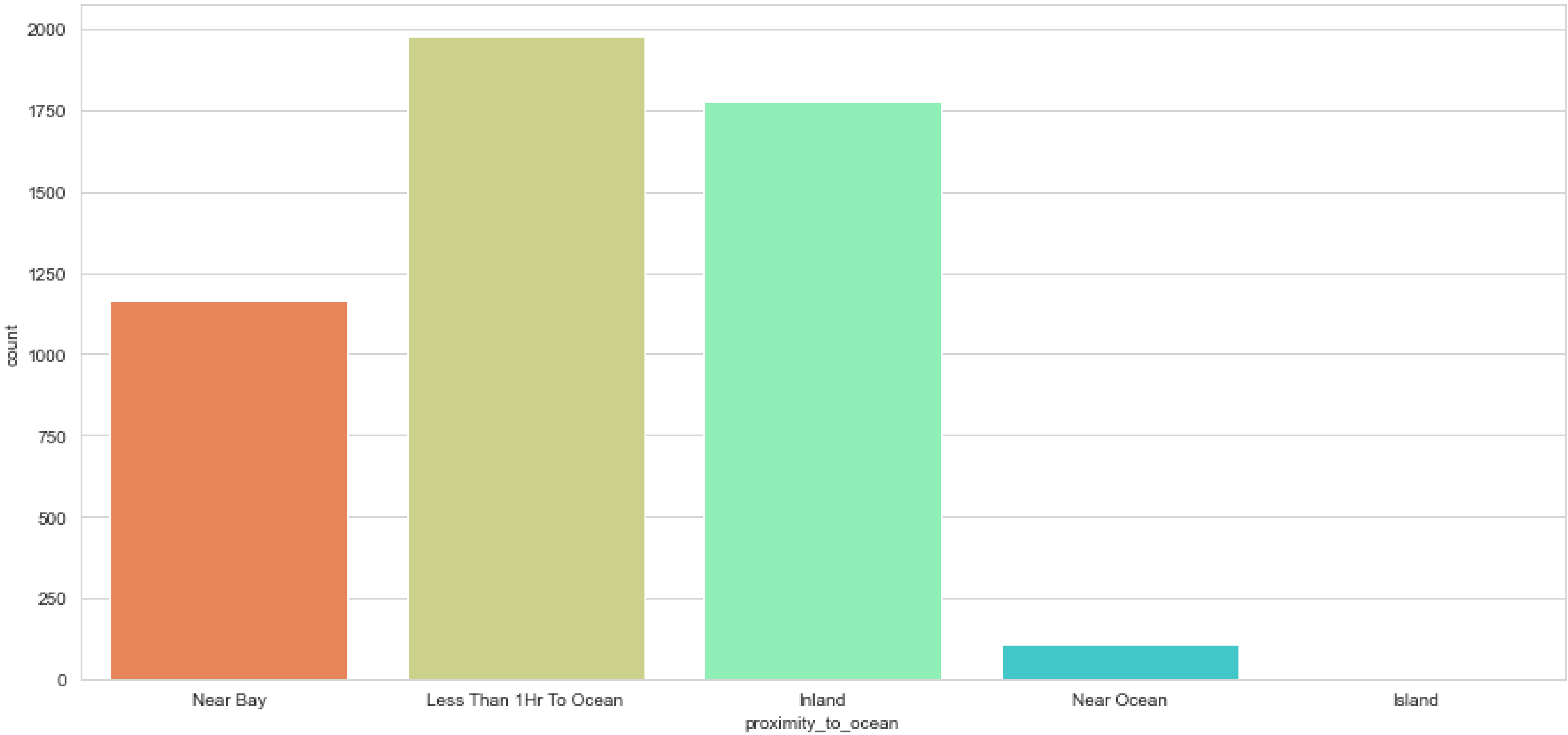
In [151]:
```
housing1.head()
```

Out[151]:

	sales_id	house_median_age	total_nr_bedrooms	population	households	median_income	house_median_value	proximity_to_ocean	total rooms per households
0	1	49	366.0	754	329	1.3750	104900	Near Bay	5.030395
1	2	51	574.0	1258	536	2.7303	109700	Near Bay	4.972015
2	3	49	282.0	570	264	1.4861	97200	Near Bay	4.602273
3	4	48	432.0	987	374	1.0972	104500	Near Bay	4.807487
4	5	52	390.0	901	403	1.4103	103900	Near Bay	3.749380

In [152]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=housing1,x='proximity_to_ocean',palette='rainbow_r')
```

Out[152]: <AxesSubplot:xlabel='proximity_to_ocean', ylabel='count'>



In [153]:
```
#Rearranging the data for splitting it conviently
housing1=housing1[['house_median_age','total_nr_bedrooms','population','households','median_income',
                   'proximity_to_ocean','total rooms per households','house_median_value']]
housing1
```

Out[153]:

	house_median_age	total_nr_bedrooms	population	households	median_income	proximity_to_ocean	total rooms per households	house_median_value
0	49	366.0	754	329	1.3750	Near Bay	5.030395	104900
1	51	574.0	1258	536	2.7303	Near Bay	4.972015	109700
2	49	282.0	570	264	1.4861	Near Bay	4.602273	97200
3	48	432.0	987	374	1.0972	Near Bay	4.807487	104500
4	52	390.0	901	403	1.4103	Near Bay	3.749380	103900
...	...	...	...	...	...	...	...	...
5241	36	388.0	867	352	3.6467	Less Than 1Hr To Ocean	4.409091	346700
5242	50	602.0	1200	433	2.8333	Island	5.676674	416700
5243	48	514.0	744	298	3.3906	Island	7.476510	300200
5244	48	286.0	348	180	2.6042	Island	4.922222	440000
5245	32	218.0	488	160	2.7361	Island	3.837500	288600

5030 rows × 8 columns

machine learning linear regression

In [154]:
```
## Splitting the Data
x=housing1.iloc[:,0:7]
#('house_median_age','total_nr_bedrooms','population','households','median_income','proximity_to_ocean','total rooms per
y=housing1.iloc[:,7]
#'house_median_value
```

In [155]:
```
x.count()
```

Out[155]:
```
house_median_age            5030
total_nr_bedrooms           5030
population                  5030
households                  5030
median_income               5030
proximity_to_ocean          5030
total rooms per households    5030
dtype: int64
```

In [156]:
```
y.count()
```

Out[156]:
```
5030
```

In [157]:
```
x=pd.get_dummies(x)
```

In [158]:
```
x.head()
```

Out[158]:

	house_median_age	total_nr_bedrooms	population	households	median_income	total rooms per households	proximity_to_ocean_Inland	proximity_to_ocean_Islanc
0	49	366.0	754	329	1.3750	5.030395	0	(
1	51	574.0	1258	536	2.7303	4.972015	0	(
2	49	282.0	570	264	1.4861	4.602273	0	(
3	48	432.0	987	374	1.0972	4.807487	0	(
4	52	390.0	901	403	1.4103	3.749380	0	(

Spliting datasets.

In [159]:
```
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=0)
```

In [160]:
```
##Feature Scaling/ Normalization
from sklearn.preprocessing import StandardScaler
scale  = StandardScaler()
x_train=scale.fit_transform(x_train)
x_test=scale.fit_transform(x_test)
```

In [161]:
```
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(x_train,y_train)
```

Out[161]:
```
LinearRegression()
```

In [162]:
```
## model for future prediciton
y_pred=lin_reg.predict(x_test)
```

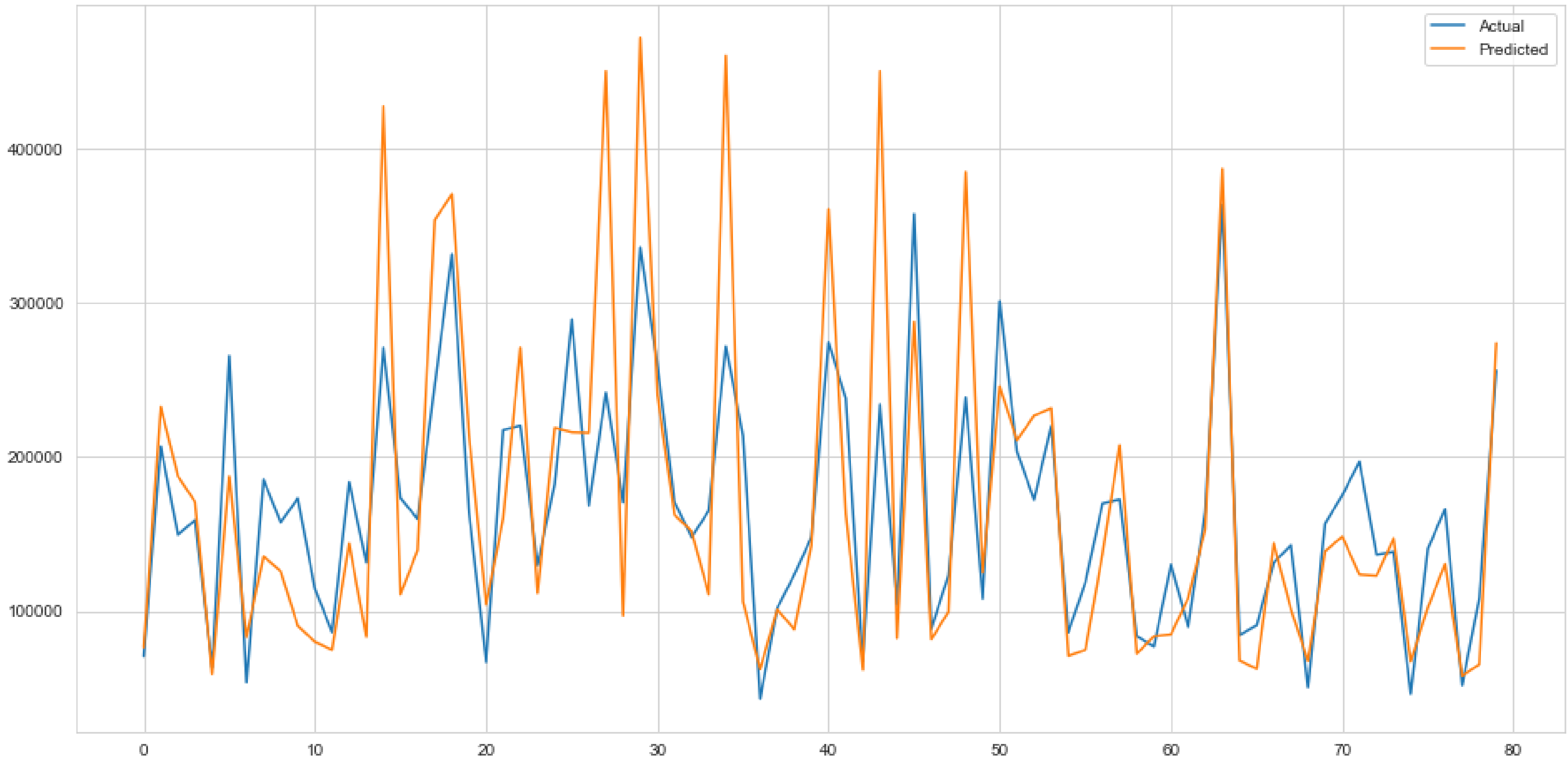
In [163]:
```
from sklearn.metrics import mean_squared_error
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
```

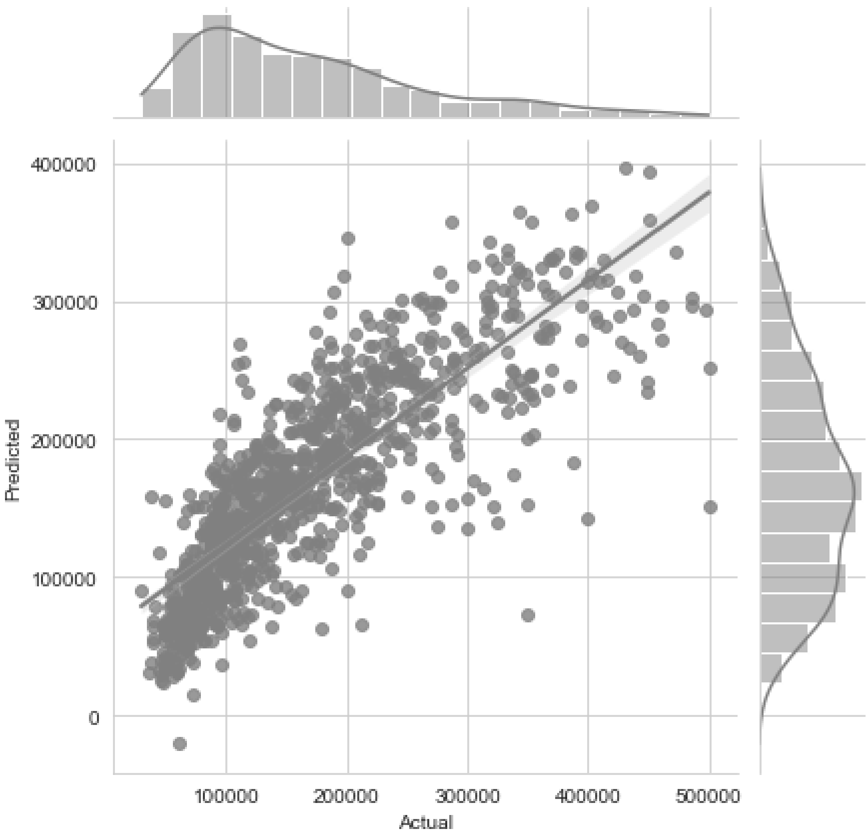
In [164]:
```
score = lin_reg.score(x_test, y_test)
score
```

Out[164]:
```
0.6695042021606388
```

In [165]:
```
test = pd.DataFrame({'Predicted':y_pred,'Actual':y_test})
fig= plt.figure(figsize=(16,8))
test = test.reset_index()
test = test.drop(['index'],axis=1)
plt.plot(test[:80])
plt.legend(['Actual','Predicted'])
sns.jointplot(x='Actual',y='Predicted',data=test,kind='reg',color="grey")
#Graph Between Actual and Predicted.
```

Out[165]: <seaborn.axisgrid.JointGrid at 0x7f7ca5901610>




400000
300000
200000
100000
0
Predicted
100000
200000
300000
400000
500000
Actual