Exercise 1

In [1]:
```
import re #Package name
```

In [16]:
```
str = "This year, the Labor Day was on September 4"
```

In [17]:
```
import re

pattern = r'\b(?:January|February|March|April|May|June|July|August|Septembe

# Test the pattern
matched_string = re.findall(pattern, str)
matched_string
```

Out[17]:
```
['September 4']
```

In [18]:
```
matches = re.finditer(pattern, str)
valid_date_found = False
for match in matches:
    start_index, end_index = match.start(), match.end()
    full_match = str[start_index:end_index]
    month_day_parts = full_match.split()
    month, day = month_day_parts[0], month_day_parts[1]

    print("Match Start Index:", start_index)
    print("Match End Index:", end_index)
    print("Full Match:", full_match)
    print("Month:", month)
    print("Day:", day)

    # Set the flag to indicate that a valid date format was found
    valid_date_found = True

# Check if a valid date format was found or not
if not valid_date_found:
    print("No valid date format found in the input string.")
```

```
Match Start Index: 32
Match End Index: 43
Full Match: September 4
Month: September
Day: 4
```

Exercise 2 Data Wrangling:

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import pandas as pd # pandas
import numpy as np # numpy
```

In [3]:
```
house_data_df = pd.read_csv("house_data.csv")
# Load the dataset.
```

In [4]:
```
house_data_df
```

Out[4]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	household
0	1	49	1655	366.0	754	32
1	2	51	2665	574.0	1258	53
2	3	49	1215	282.0	570	26
3	4	48	1798	432.0	987	37
4	5	52	1511	390.0	901	40
...	...	...	...	...	...	
5241	5242	36	1552	388.0	867	35
5242	5243	50	2458	602.0	1200	43
5243	5244	48	2228	514.0	744	29
5244	5245	48	886	286.0	348	18
5245	5246	32	614	218.0	488	16

5246 rows × 9 columns

In [5]:
```
house_data_df.info()
```

```
<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 [6]:
```
# Check the number of rows and columns
num_rows, num_cols = house_data_df.shape
print(f"Number of rows: {num_rows}, Number of columns: {num_cols}")
```

```
Number of rows: 5246, Number of columns: 9
```

In [7]:
```
house_data_df.columns
```

Out[7]:
```
Index(['sales_id', 'house_median_age', 'total_nr_rooms', 'total_nr_bedroom
s',
       'population', 'households', 'median_income', 'house_median_value',
       'proximity_to_ocean'],
      dtype='object')
```

In [8]:
```
house_data_df.isnull()
```

Out[8]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	household
0	False	False	False	False	False	Fals
1	False	False	False	False	False	Fals
2	False	False	False	False	False	Fals
3	False	False	False	False	False	Fals
4	False	False	False	False	False	Fals
...	...	...	...	...	...	
5241	False	False	False	False	False	Fals
5242	False	False	False	False	False	Fals
5243	False	False	False	False	False	Fals
5244	False	False	False	False	False	Fals
5245	False	False	False	False	False	Fals

5246 rows × 9 columns

In [9]:
```
housedata_df = house_data_df.dropna()
```

In [10]:
```
housedata_df
```

Out[10]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	household
0	1	49	1655	366.0	754	32
1	2	51	2665	574.0	1258	53
2	3	49	1215	282.0	570	26
3	4	48	1798	432.0	987	37
4	5	52	1511	390.0	901	40
...	...	...	...	...	...	
5241	5242	36	1552	388.0	867	35
5242	5243	50	2458	602.0	1200	43
5243	5244	48	2228	514.0	744	29
5244	5245	48	886	286.0	348	18
5245	5246	32	614	218.0	488	16

5191 rows × 9 columns

In [11]:
```
house_data_df.corr()
```

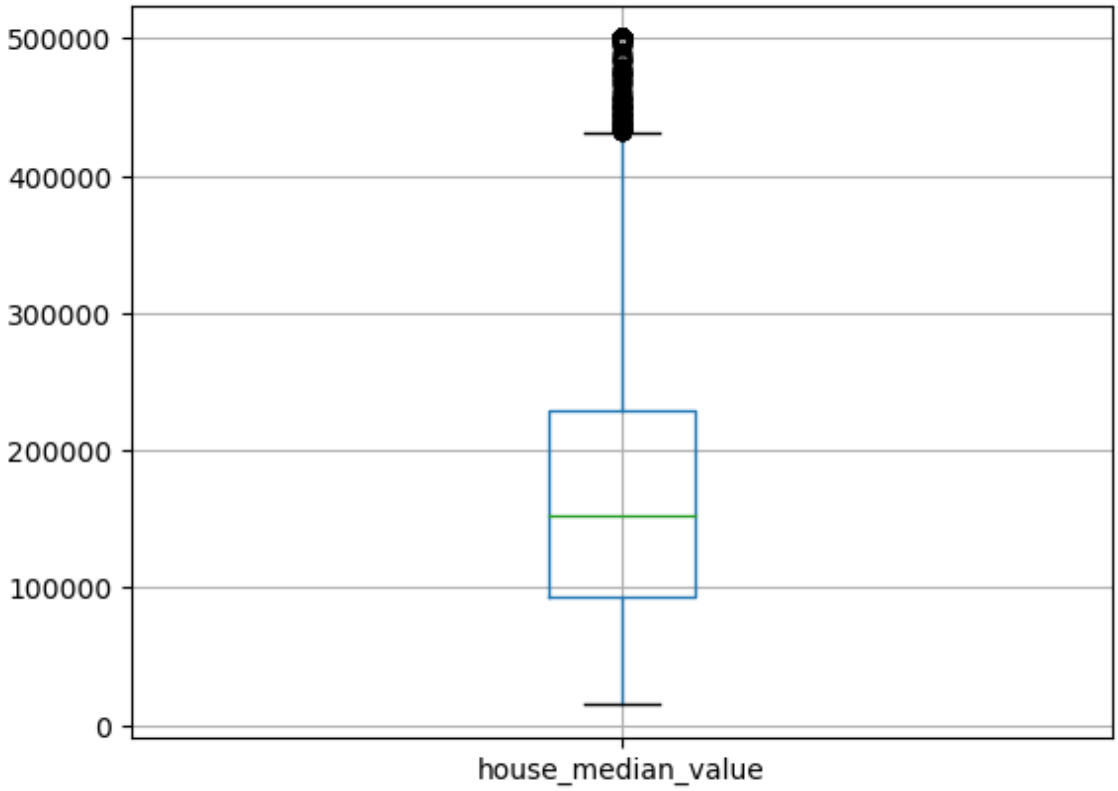
Out[11]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	popu
sales_id	1.000000	0.139236	-0.065099	0.059027	0.1
house_median_age	0.139236	1.000000	-0.361420	-0.306069	-0.2
total_nr_rooms	-0.065099	-0.361420	1.000000	0.886647	0.8
total_nr_bedrooms	0.059027	-0.306069	0.886647	1.000000	0.8
population	0.100802	-0.272638	0.819849	0.887325	1.0
households	0.057595	-0.286427	0.889199	0.987911	0.9
median_income	-0.054194	-0.115312	0.305356	0.037060	0.0
house_median_value	0.184434	0.084097	0.266540	0.178374	0.0

In [12]:
```
import matplotlib.pyplot as plt
```

In [13]:
```
house_data_df.boxplot('house_median_value')
```

Out[13]: <Axes: >



Exercise 2 Data Analytics:

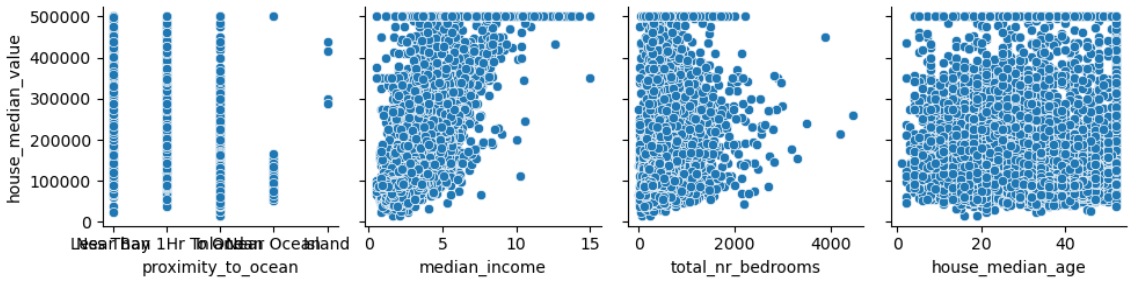
In [14]:
```
#Assginment A_04(A)
#Exercise 2 - Data analytics:  plotting relationships between house value a
```

In [15]:
```
import warnings
warnings.filterwarnings('ignore')
```

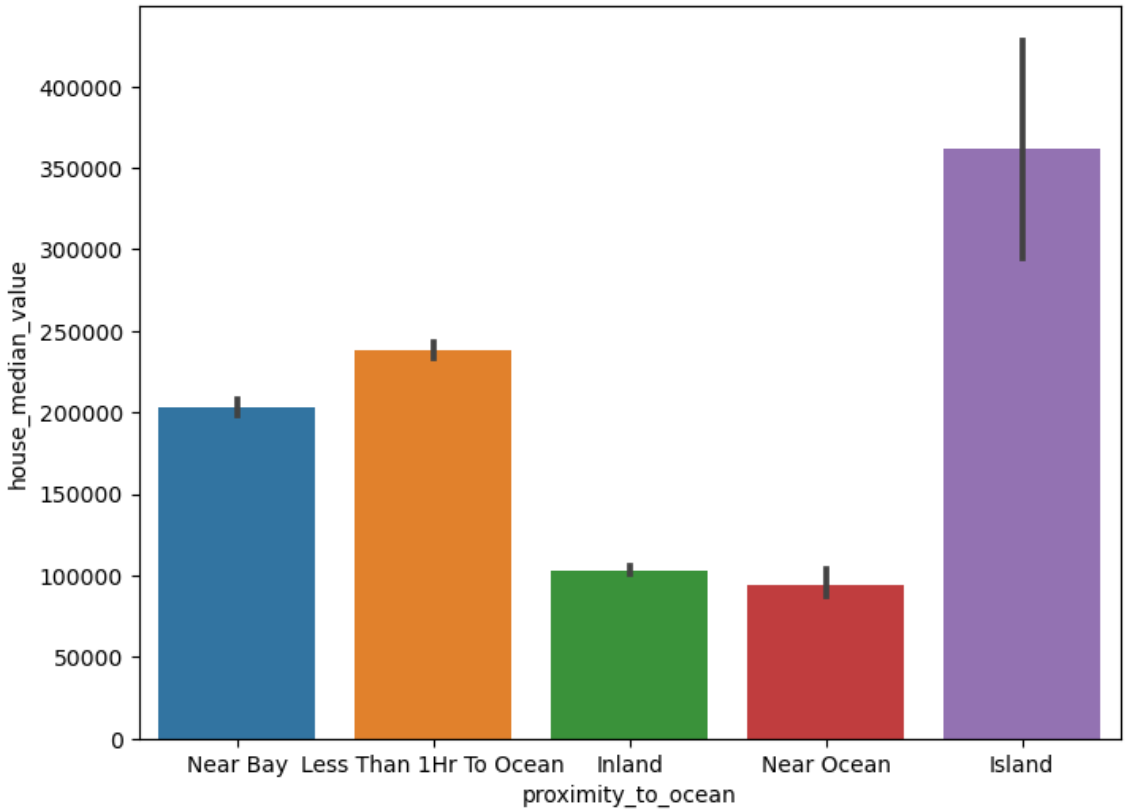
In [16]:
```
import pandas as pd #pandas

df = pd.read_csv('house_data.csv')
```

In [17]:
```
import matplotlib.pyplot as plt
import seaborn as sns

#Plotting the relationships
plt.figure(figsize=(6,6))
sns.pairplot(df, x_vars=['proximity_to_ocean', 'median_income', 'total_nr_b
plt.show()
```

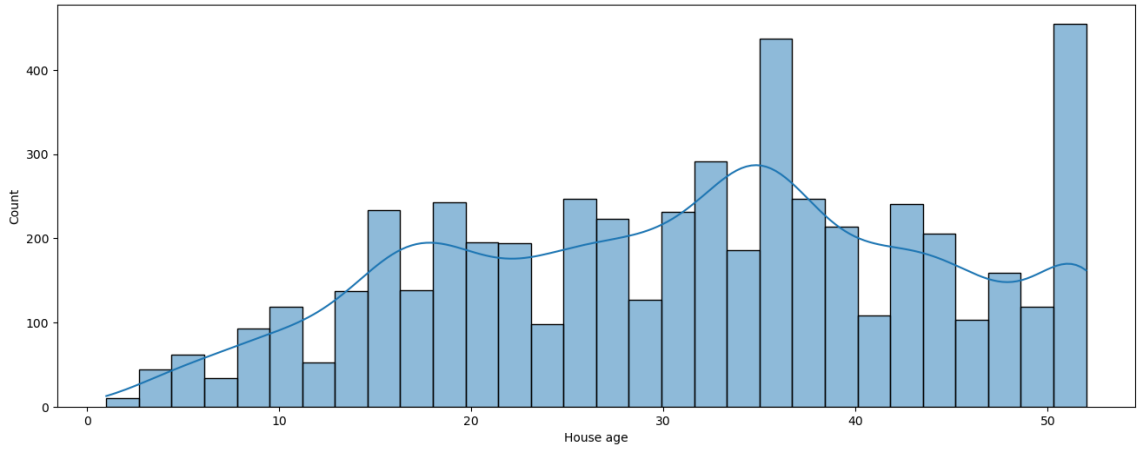
```
<Figure size 600x600 with 0 Axes>
```



In [18]:
```
#Bar chart for proximity to ocean vs house value
plt.figure(figsize=(8, 6))

sns.barplot(x = 'proximity_to_ocean', y = 'house_median_value', data=df)

plt.show()
```



In [19]:
```
#Histogram for house age

plt.figure(figsize=(16, 6))
sns.histplot(df['house_median_age'], bins=30, kde=True)
plt.xlabel('House age')
plt.ylabel('Count')
plt.show()
```



Exercise 2 Linear Regression

In [20]:
```
#Team 4_A_04(G)
#Exercise 2:machine learning linear regression:
```

In [21]:
```
#import libraries
import warnings
warnings.filterwarnings('ignore')
import pandas as pd # pandas
import numpy as np # numpy
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
```

In [22]:
```
housedata1_df = pd.read_csv("house_data.csv")
housedata1_df.head( 10 )
```

Out[22]:

	sales_id	house_median_age	total_nr_rooms	total_nr_bedrooms	population	households
0	1	49	1655	366.0	754	329
1	2	51	2665	574.0	1258	536
2	3	49	1215	282.0	570	264
3	4	48	1798	432.0	987	374
4	5	52	1511	390.0	901	403
5	6	52	1470	330.0	689	309
6	7	52	2432	715.0	1377	696
7	8	52	1665	419.0	946	395
8	9	51	936	311.0	517	249
9	10	49	713	202.0	462	189

In [23]:
```
housedata_df = housedata1_df.dropna()
```

In [24]:
```
X_features = housedata_df.columns
X_features = ['house_median_age','median_income', 'total_nr_bedrooms', 'pro
```

In [25]:
```
#Encoding Categorical Features
housedata_df['proximity_to_ocean'].unique()
```

Out[25]:
```
array(['Near Bay', 'Less Than 1Hr To Ocean', 'Inland', 'Near Ocean',
       'Island'], dtype=object)
```

In [26]:
```
pd.get_dummies(housedata_df['proximity_to_ocean'])[0:5]
```

Out[26]:

	Inland	Island	Less Than 1Hr To Ocean	Near Bay	Near Ocean
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

In [27]:
```
categorical_feature = ['proximity_to_ocean']
```

In [28]:
```
housedata_encoded_df = pd.get_dummies(housedata_df[X_features],
columns = categorical_feature,
drop_first = True)

housedata_encoded_df.columns
```

Out[28]:
```
Index(['house_median_age', 'median_income', 'total_nr_bedrooms',
       'proximity_to_ocean_Island',
       'proximity_to_ocean_Less Than 1Hr To Ocean',
       'proximity_to_ocean_Near Bay', 'proximity_to_ocean_Near Ocean'],
      dtype='object')
```

In [29]:
```
X_features = housedata_encoded_df.columns
```

In [30]:
```
X = sm.add_constant(housedata_encoded_df)
X.head(5)
```

Out[30]:

	const	house_median_age	median_income	total_nr_bedrooms	proximity_to_ocean_Island
0	1.0	49	1.3750	366.0	0
1	1.0	51	2.7303	574.0	0
2	1.0	49	1.4861	282.0	0
3	1.0	48	1.0972	432.0	0
4	1.0	52	1.4103	390.0	0

In [31]:
```
Y = housedata_df['house_median_value']
train_X, test_X, train_y, test_y = train_test_split(X,Y,train_size = 0.7,ra
```

In [32]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(train_X, train_y)
```

Out[32]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
# Predicting the Test set result
Y_Pred = regressor.predict(test_X)
```

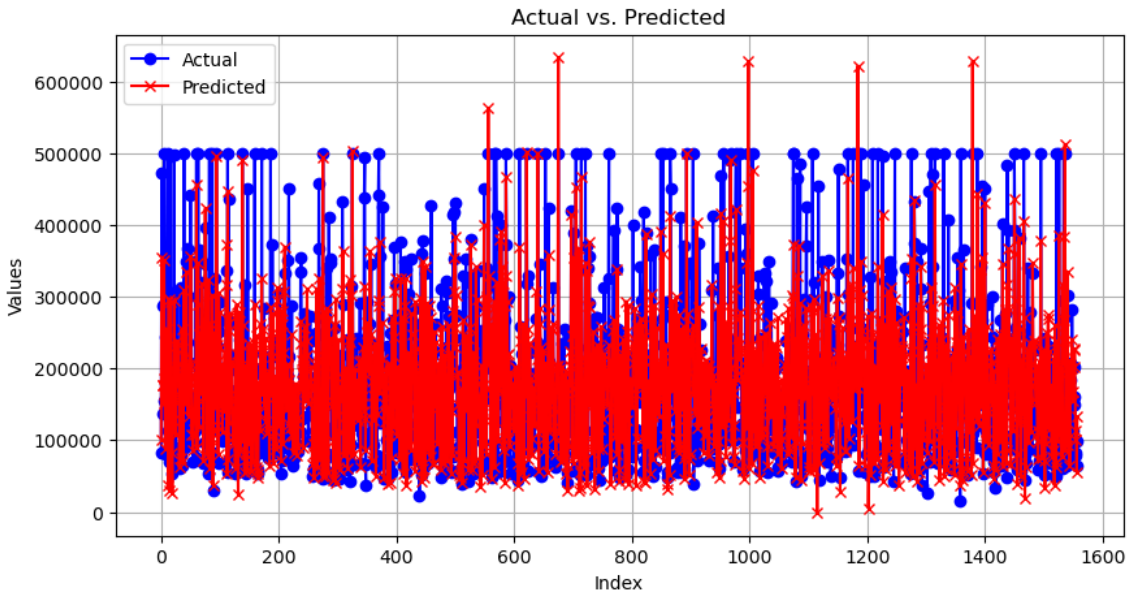
In [34]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(test_y, Y_Pred)
mse
```

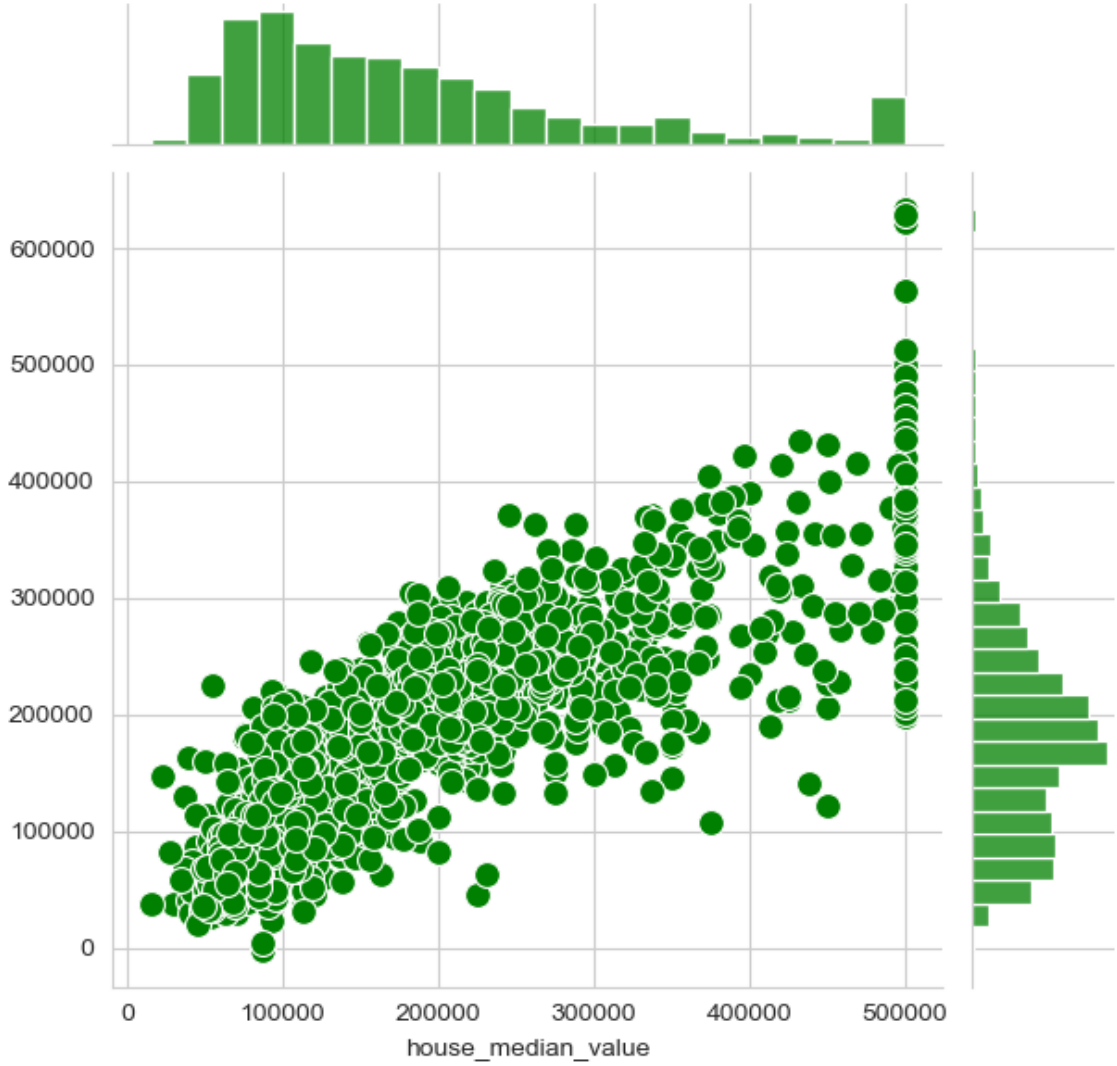
Out[34]:
```
4353709658.229696
```

In [35]:
```
rmse = np.sqrt(mse)
rmse
#yes it is satisfied.
```

Out[35]:
```
65982.64664462692
```

In [36]:
```
#plot predictions
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

actual_values = test_y
predicted_values = Y_Pred

# Create a range of indices for the x-axis
x_indices = np.arange(len(actual_values))

# Create a Matplotlib figure and axes
plt.figure(figsize=(10, 5))
plt.title('Actual vs. Predicted')

# Plot the actual values as a blue line
plt.plot(x_indices, actual_values, label='Actual', color='blue', marker='o'

# Plot the predicted values as a red line
plt.plot(x_indices, predicted_values, label='Predicted', color='red', marke

# Add labels, legend, and gridlines
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend()
plt.grid()

# Show the combined line chart using Matplotlib
plt.show()

# Create a Seaborn jointplot for a scatter plot with marginal histograms
sns.set_style("whitegrid")
sns.jointplot(x=actual_values, y=predicted_values, kind='scatter', s=100, c

# Show the Seaborn jointplot
plt.show()
```




600000
500000
400000
300000
200000
100000
0
0
100000
200000
300000
400000
500000
house_median_value

In [ ]: