IMPORTING LIBRARIES

In [1]:
```
# Importing essential libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries for regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Data visualization settings (optional)
%matplotlib inline
plt.style.use('ggplot')

# Ignore warnings (optional)
import warnings
warnings.filterwarnings('ignore')

# Set random seed for reproducibility (optional)
np.random.seed(0)
```

LOADING FILES AND DATA

In [2]:
```
file_path = 'C:/Users/engin/Downloads/house.csv'
```

In [3]:
```
# Read the CSV file into a DataFrame
house = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify the import
house.head()
```

Out[3]:

	longitude	latitude	age	rooms	population	households	income	houseprice
0	-122.23	37.88	41	880	322	126	8.3252	452600
1	-122.22	37.86	21	7099	2401	1138	8.3014	358500
2	-122.24	37.85	52	1467	496	177	7.2574	352100
3	-122.25	37.85	52	1274	558	219	5.6431	341300
4	-122.25	37.85	52	1627	565	259	3.8462	342200

DATA ANALYSIS

In [4]:
```
house.shape
```

Out[4]:
```
(20640, 8)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   longitude   20640 non-null  float64
 1   latitude    20640 non-null  float64
 2   age         20640 non-null  int64  
 3   rooms       20640 non-null  int64  
 4   population  20640 non-null  int64  
 5   households  20640 non-null  int64  
 6   income      20640 non-null  float64
 7   houseprice  20640 non-null  int64  
dtypes: float64(3), int64(5)
memory usage: 1.3 MB
```

In [6]:
```
house.isnull().sum()
```

Out[6]:
```
longitude     0
latitude      0
age           0
rooms         0
population    0
households    0
income        0
houseprice    0
dtype: int64
```

In [7]:
```
house.isnull().sum()
```

Out[7]:
```
longitude     0
latitude      0
age           0
rooms         0
population    0
households    0
income        0
houseprice    0
dtype: int64
```

In [8]:
```
house.describe()
```

Out[8]:

	longitude	latitude	age	rooms	population	households	
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640
mean	-119.569704	35.631861	28.639486	2635.763081	1425.476744	499.539680	3
std	2.003532	2.135952	12.585558	2181.615252	1132.462122	382.329753	1
min	-124.350000	32.540000	1.000000	2.000000	3.000000	1.000000	0
25%	-121.800000	33.930000	18.000000	1447.750000	787.000000	280.000000	2
50%	-118.490000	34.260000	29.000000	2127.000000	1166.000000	409.000000	3
75%	-118.010000	37.710000	37.000000	3148.000000	1725.000000	605.000000	4
max	-114.310000	41.950000	52.000000	39320.000000	35682.000000	6082.000000	15

In [9]:
```
house['houseprice'].value_counts()
```

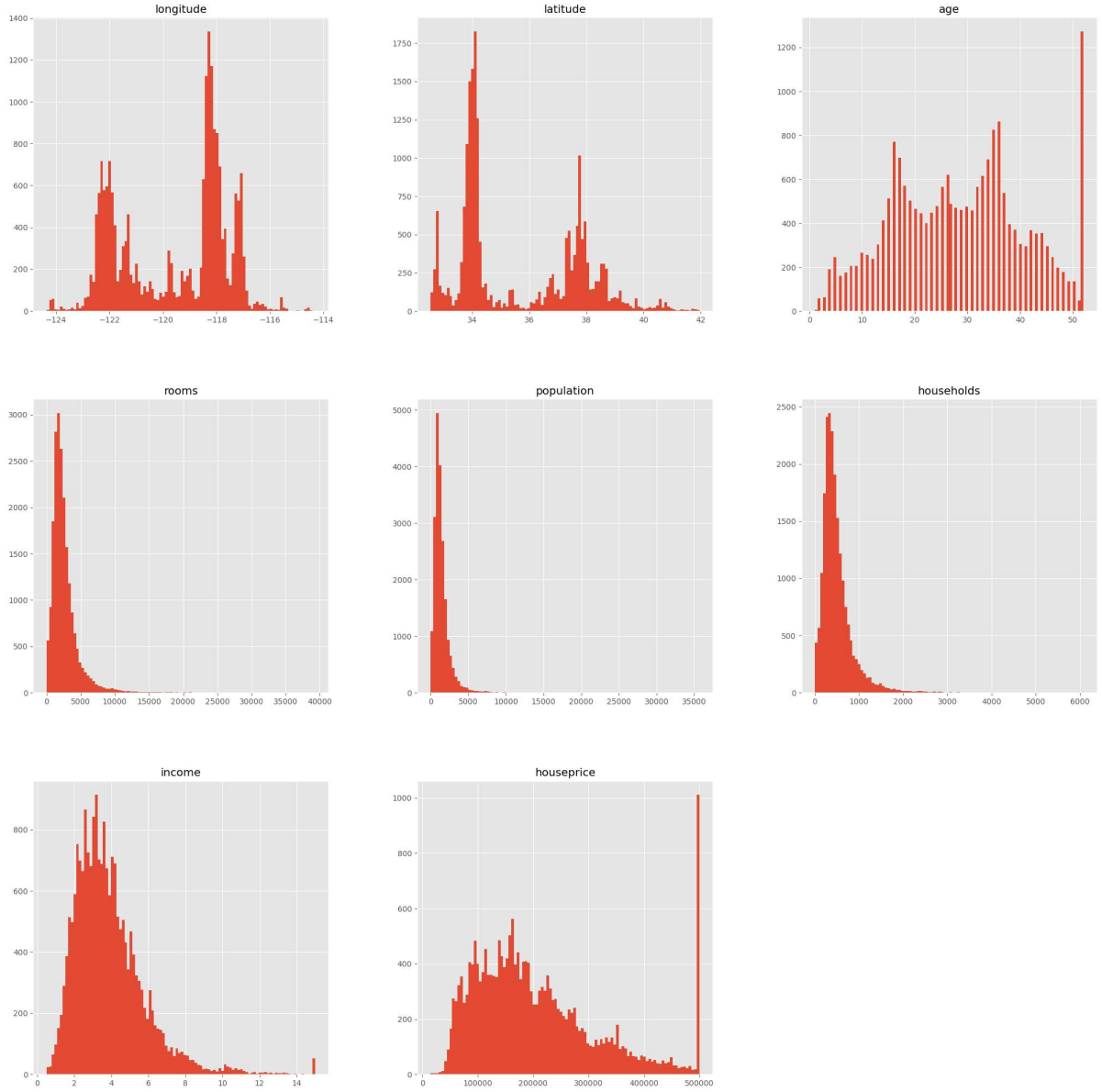
Out[9]:
```
houseprice
500001    965
137500    122
162500    117
112500    103
187500     93
          ...
359200      1
54900       1
377600      1
81200       1
47000       1
Name: count, Length: 3842, dtype: int64
```

In [ ]:
```
```

DATA VISUALIZATION

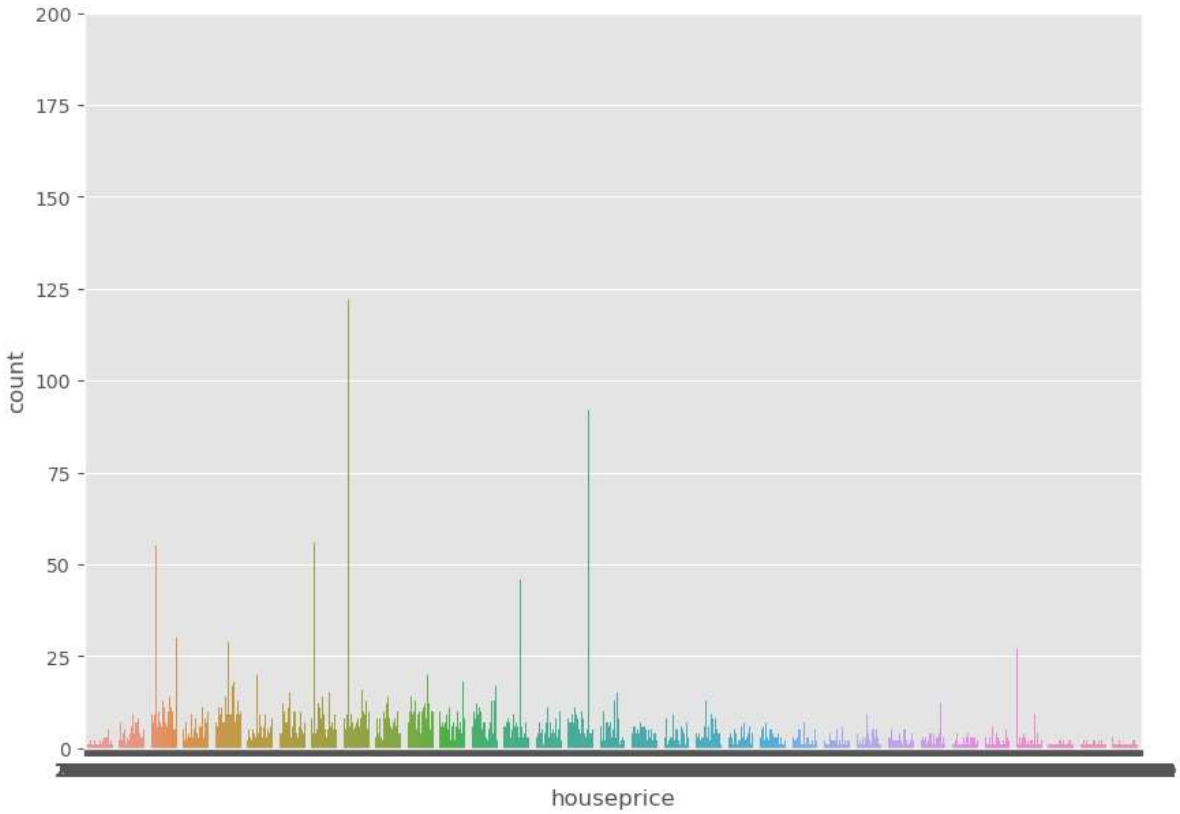
PAIRWISE HISTOGRAM

In [10]:
```
house.hist(bins=100, figsize=(25,25))
plt.show()
```



COUNT PLOT

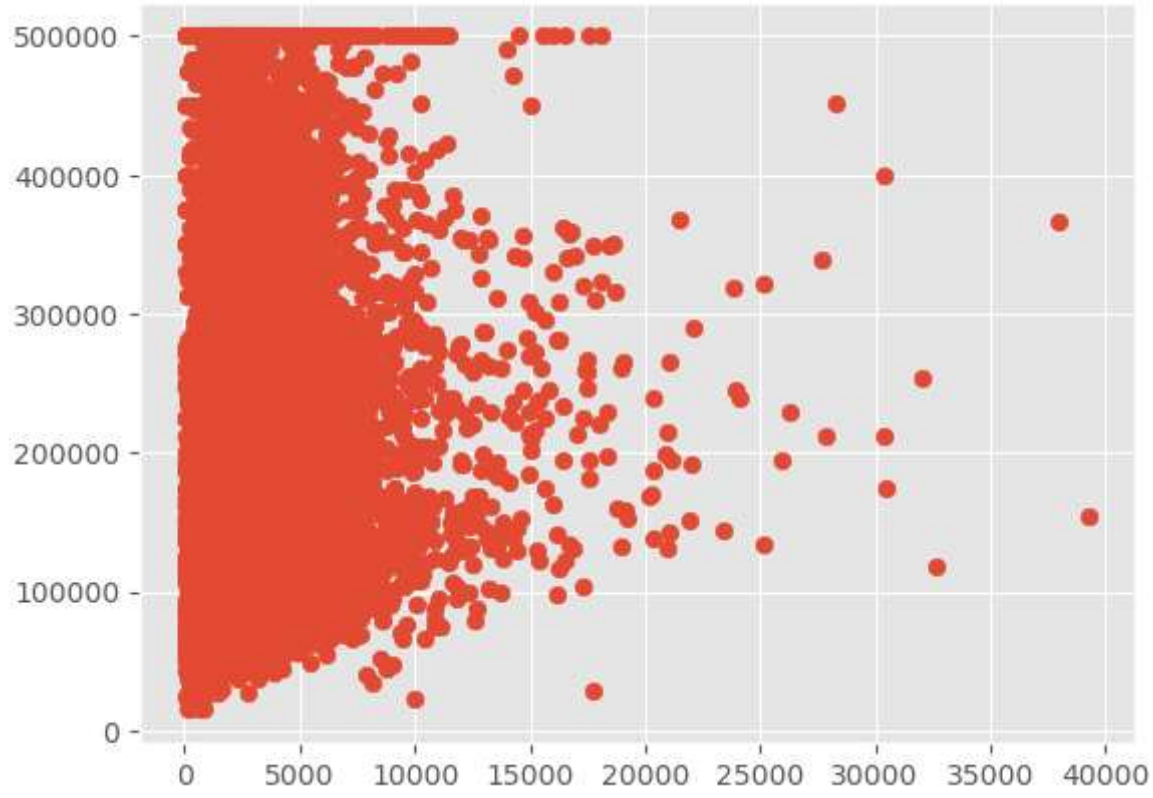
In [11]:
```
plt.figure(figsize=(10, 7))

# Create the countplot
ax = sns.countplot(data=house, x='houseprice')

# Set the y-axis limits to a maximum of 200
ax.set(ylim=(0, 200))

# Show the plot
plt.show()
```



SCATTER PLOT

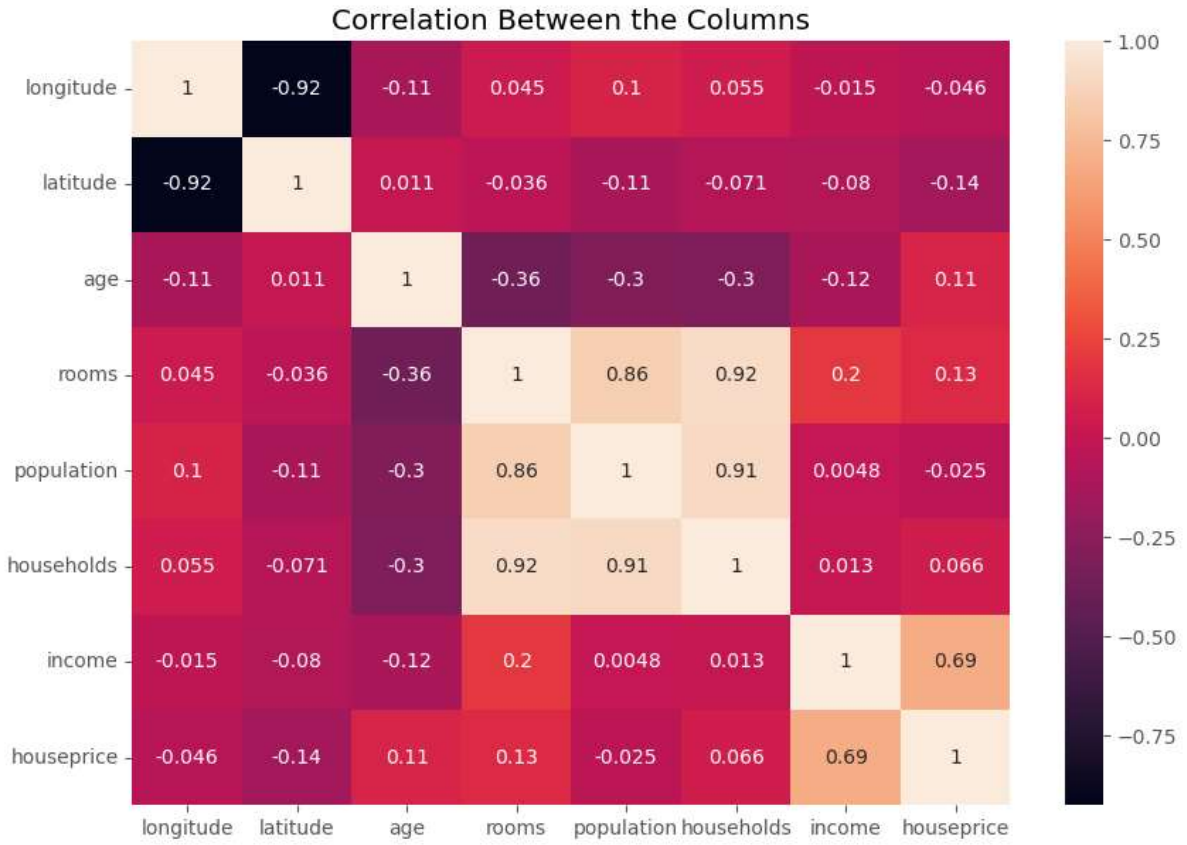
In [12]:
```
import matplotlib.pyplot as plt

plt.scatter(house['rooms'], house['houseprice'])
```

Out[12]:
```
<matplotlib.collections.PathCollection at 0x15d2f9b2e10>
```

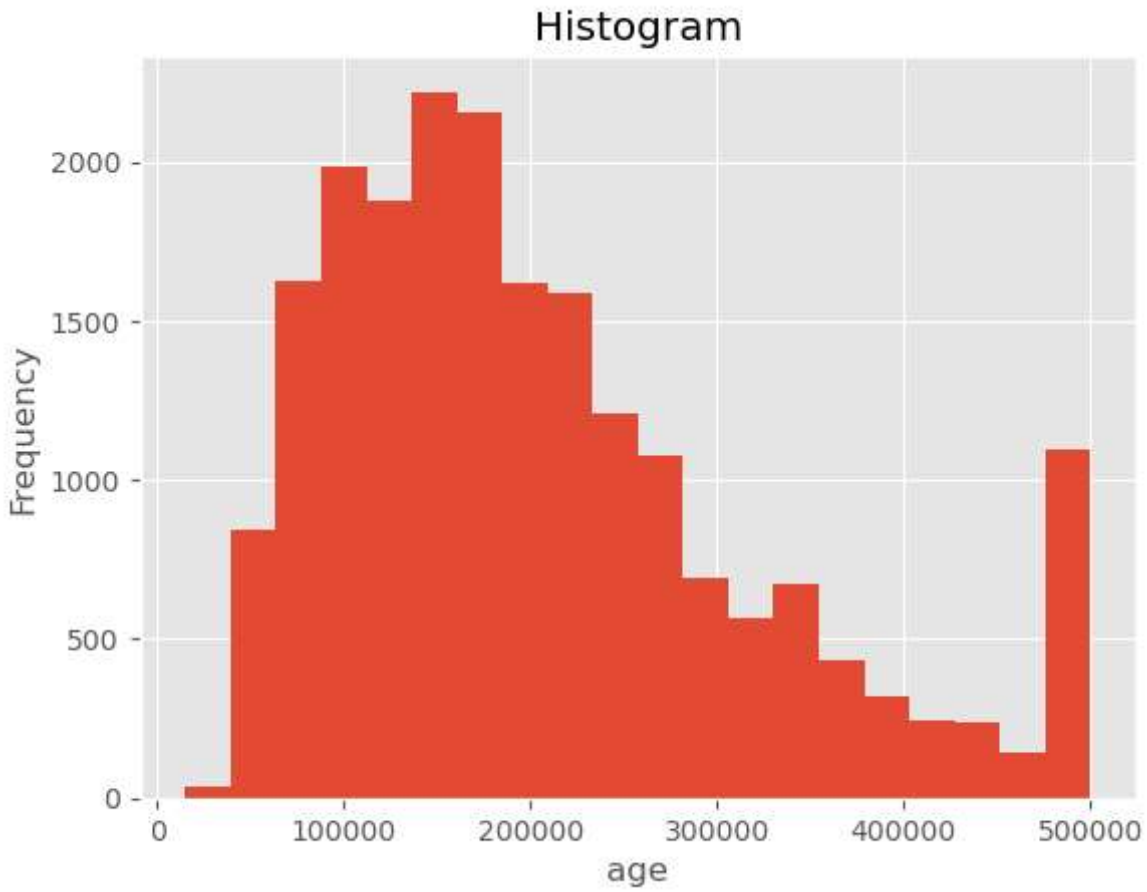


Heatmap of Correlation

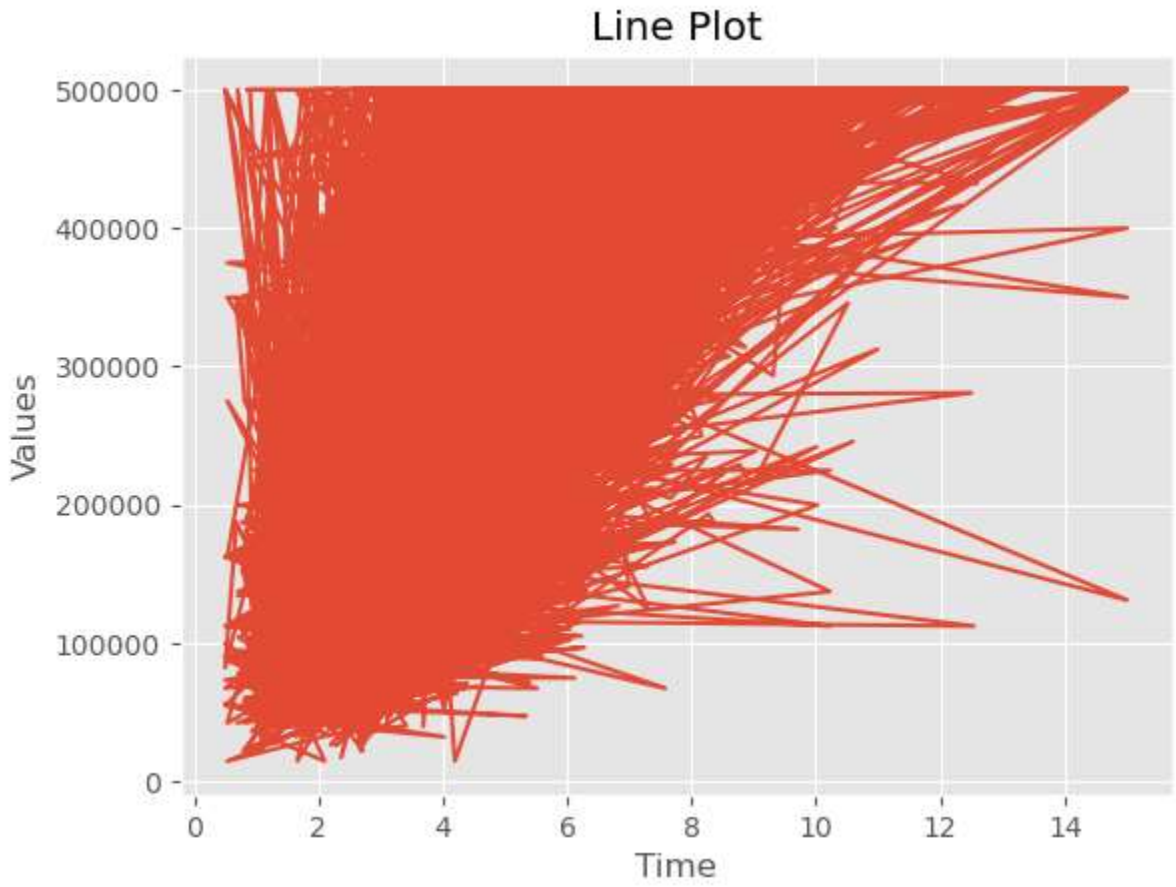
In [13]:
```
plt.figure(figsize=(10, 7))
sns.heatmap(house.corr(), annot=True)
plt.title('Correlation Between the Columns')
plt.show()
```



In [14]:
```
import matplotlib.pyplot as plt

# Replace 'variable' with the actual column name from your 'house' DataFrame
plt.hist(house['houseprice'], bins=20)
plt.xlabel('age')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.show()
```



In [15]:
```
import matplotlib.pyplot as plt

plt.plot(house['income'], house['houseprice'])
plt.xlabel('Time')
plt.ylabel('Values')
plt.title('Line Plot')
plt.show()
```

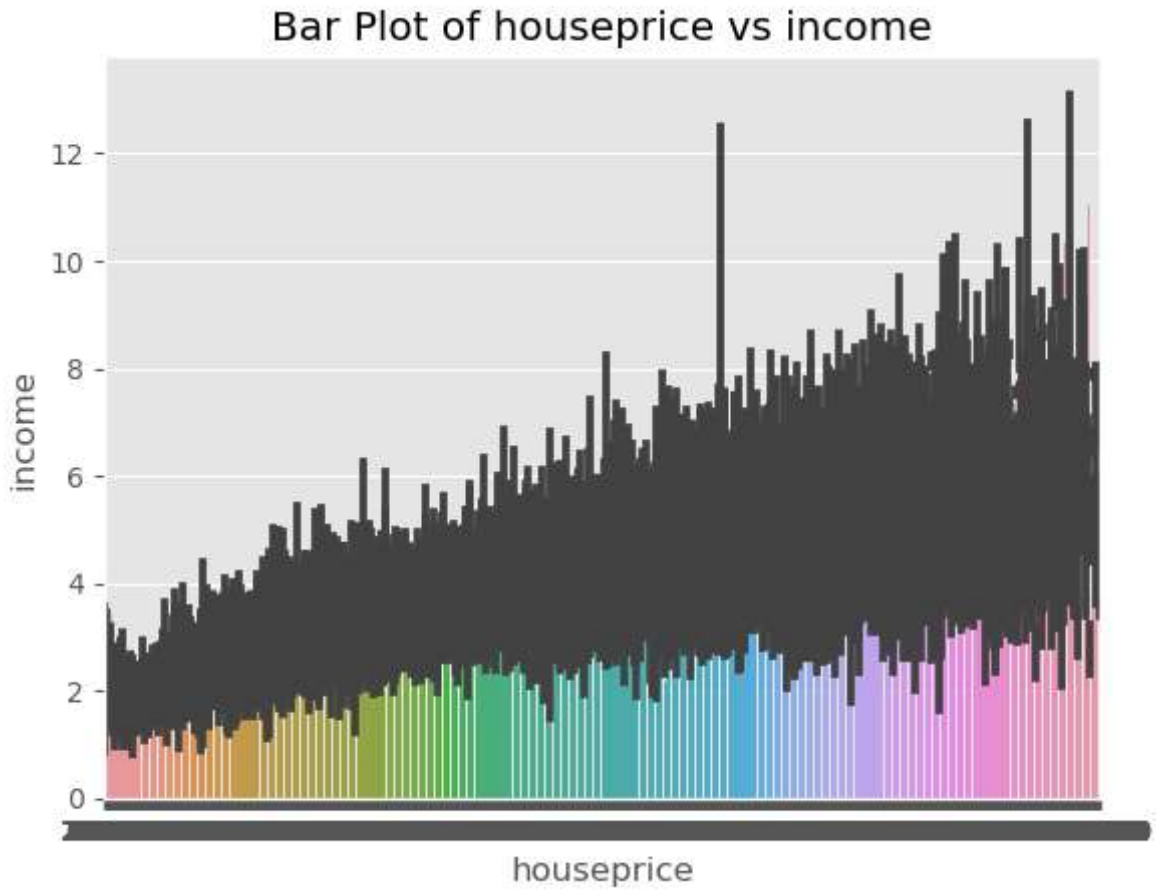


In [16]:
```
house.corr()['houseprice'].sort_values()
```

Out[16]:
```
latitude     -0.144160
longitude    -0.045967
population   -0.024650
households    0.065843
age           0.105623
rooms         0.134153
income        0.688075
houseprice    1.000000
Name: houseprice, dtype: float64
```

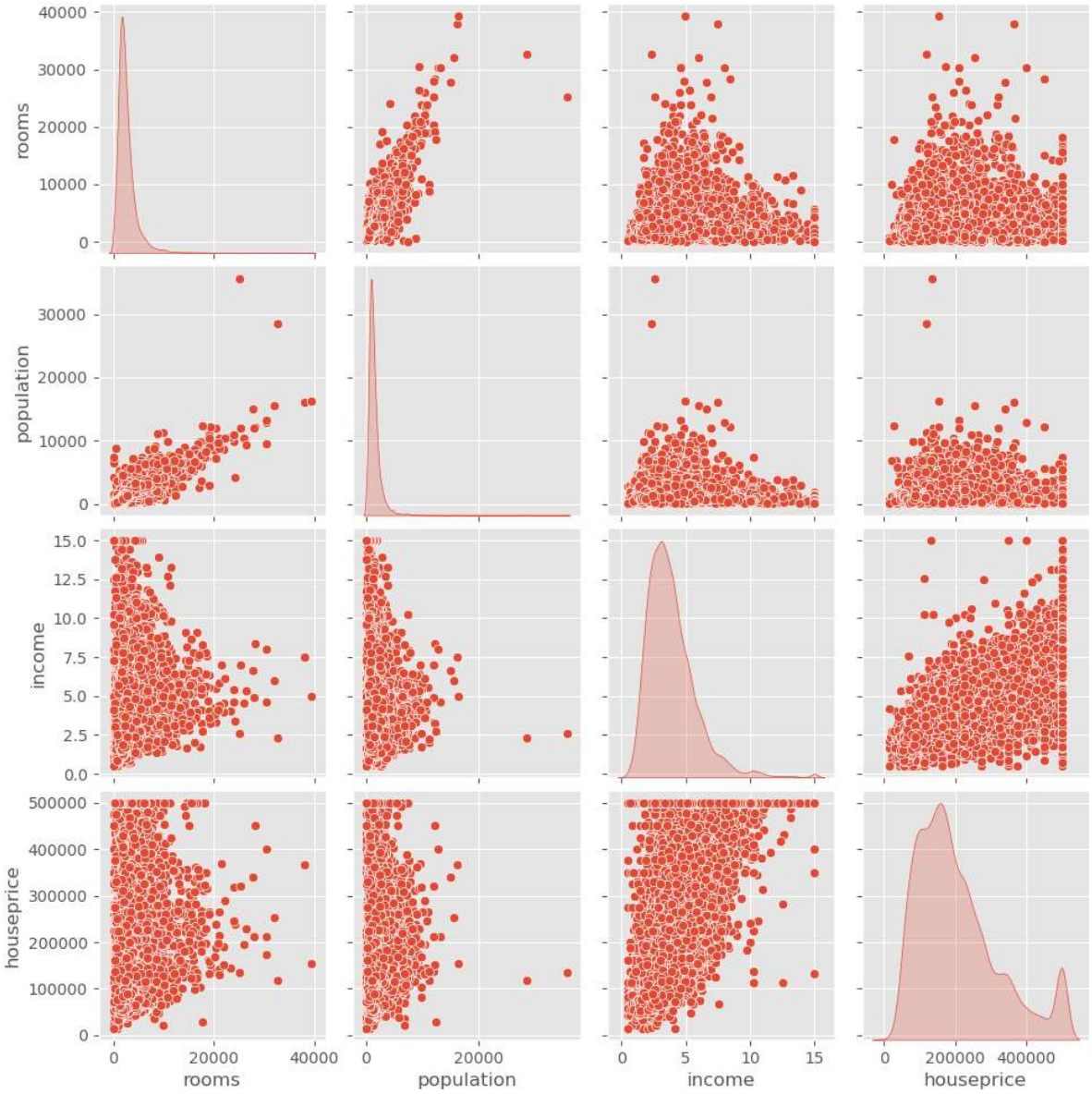
BARPLOT

In [17]:
```
sns.barplot(x=house['houseprice'], y=house['income'])
plt.title('Bar Plot of houseprice vs income')
plt.show()
```



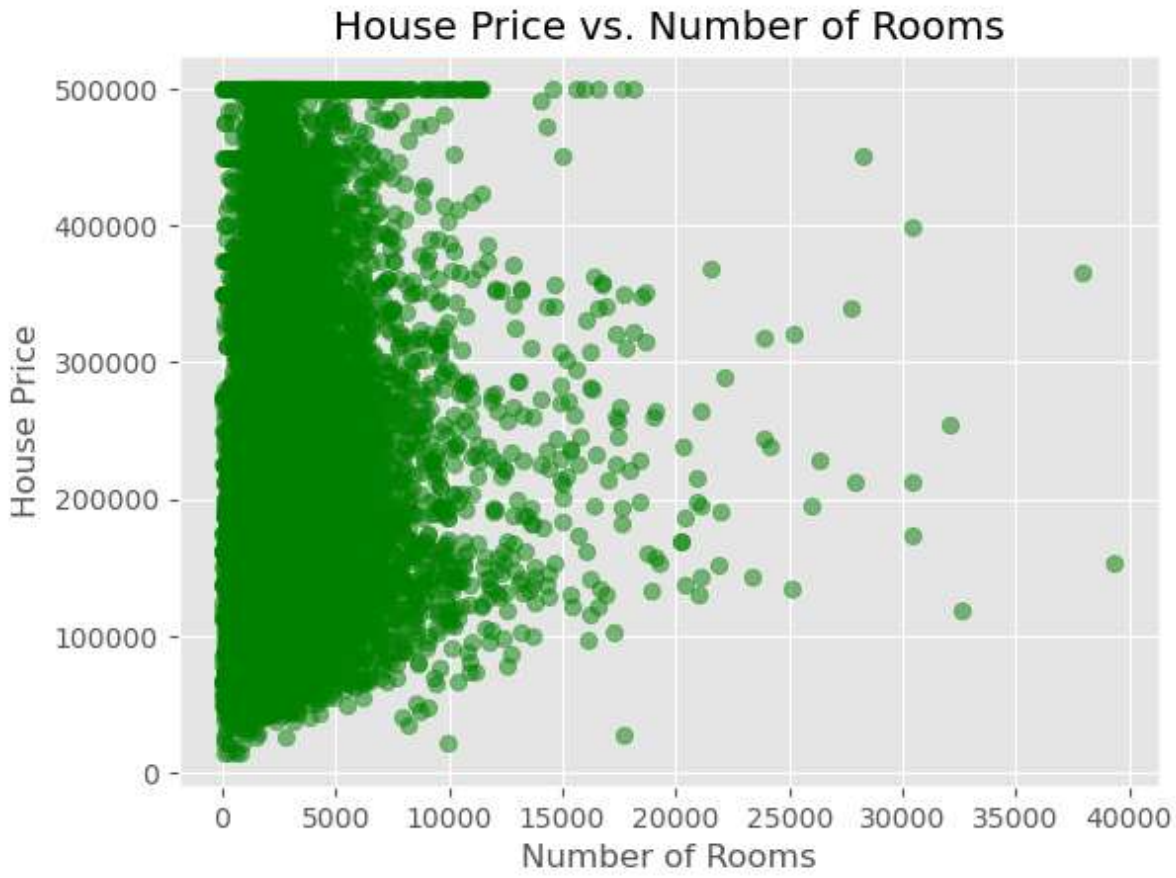
Paiwise Plot of House Price vs. Number of Rooms

In [18]:
```
sns.pairplot(house, vars=['rooms', 'population', 'income', 'houseprice'], diag_kinc
plt.show()
```



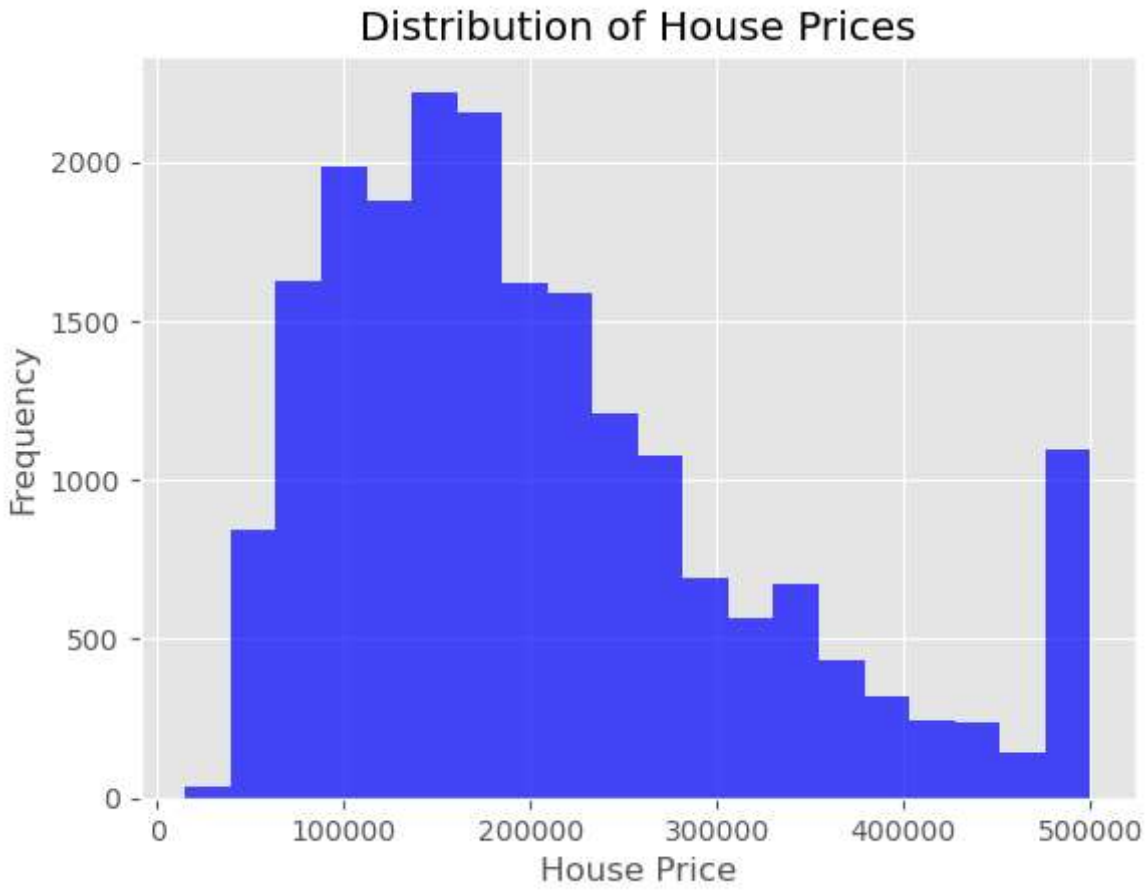
Scatter Plot of House Price vs. Number of Rooms

In [19]:
```
plt.scatter(house ['rooms'], house ['houseprice'], alpha=0.5, color='green')
plt.title('House Price vs. Number of Rooms')
plt.xlabel('Number of Rooms')
plt.ylabel('House Price')
plt.show()
```

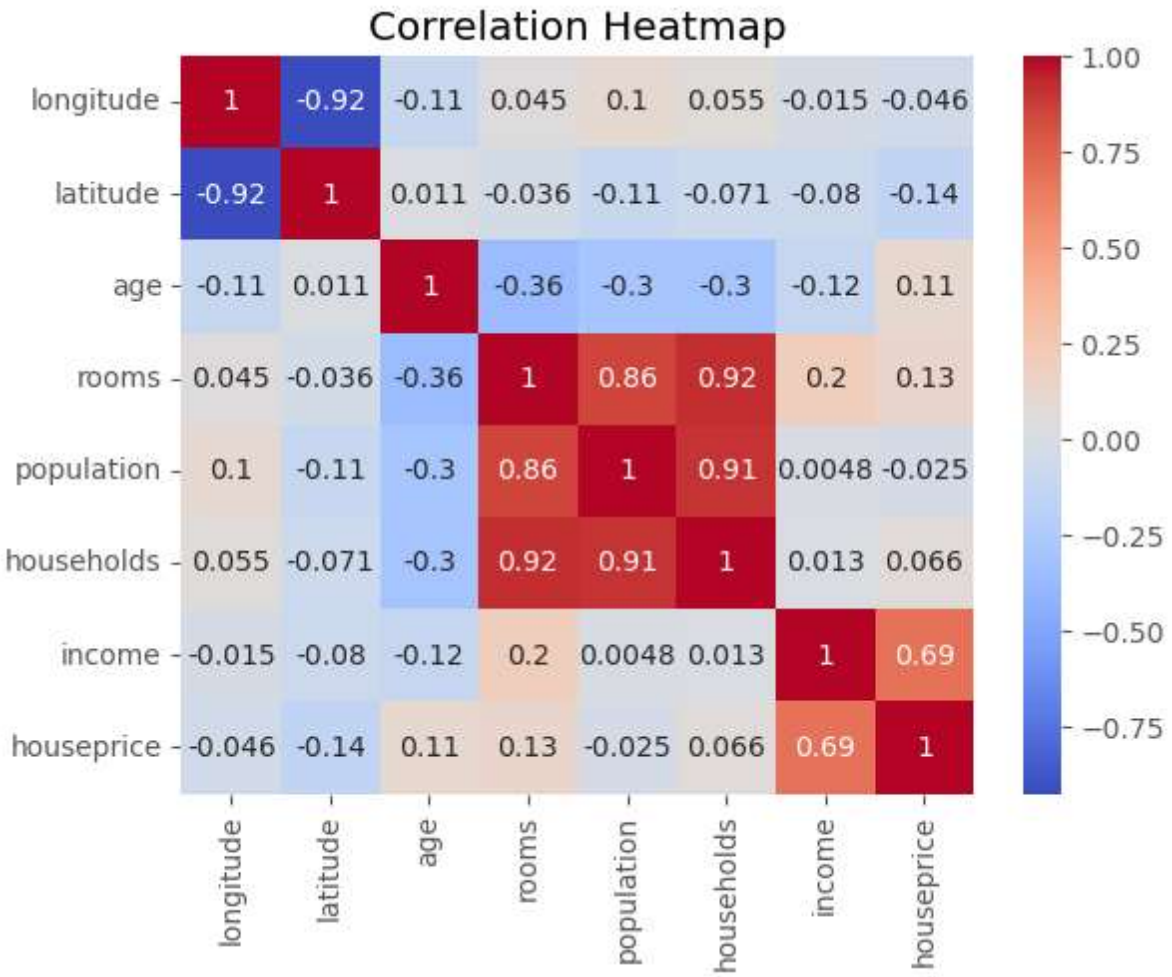


HISTOGRAM

In [20]:
```
plt.hist(house ['houseprice'], bins=20, color='blue', alpha=0.7)
plt.title('Distribution of House Prices')
plt.xlabel('House Price')
plt.ylabel('Frequency')
plt.show()
```

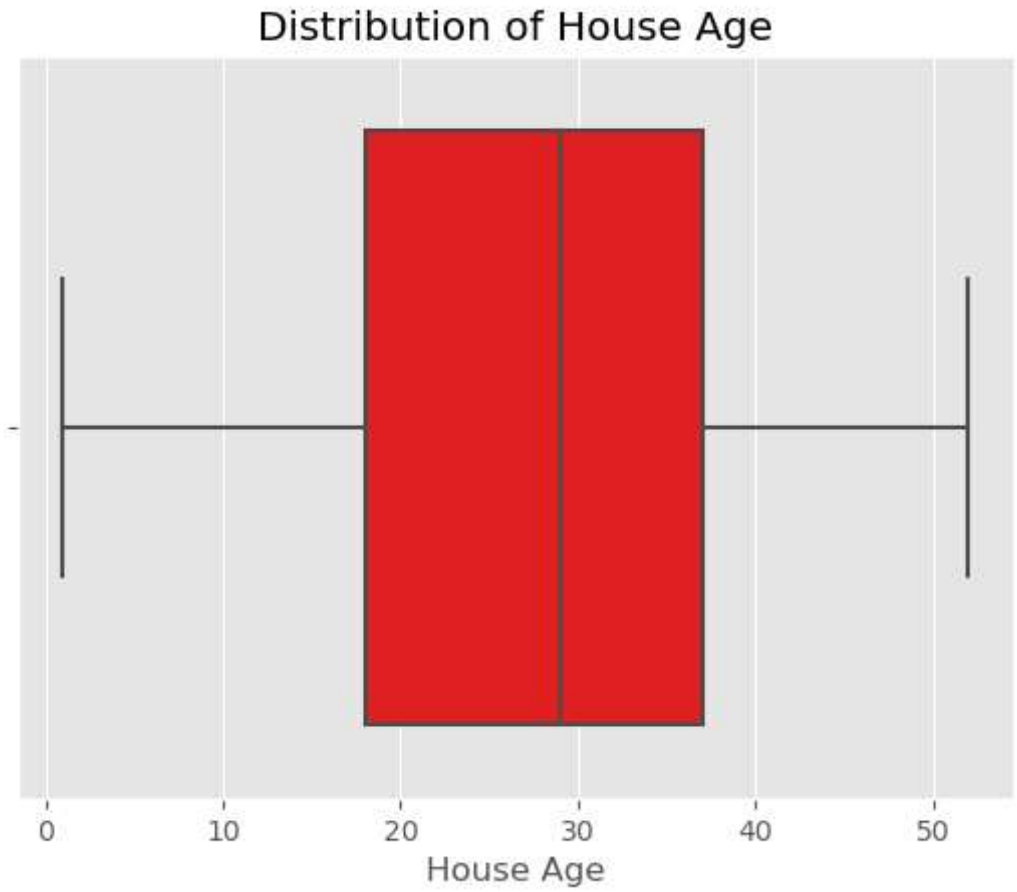


In [21]:
```
correlation_matrix = house.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



BOXPLOT

In [22]:
```
import seaborn as sns

sns.boxplot(x='age', data=house, color='red')
plt.title('Distribution of House Age')
plt.xlabel('House Age')
plt.show()
```



In [23]:
```
count_values = house['houseprice'].value_counts()
print(count_values)
```

```
houseprice
500001    965
137500    122
162500    117
112500    103
187500     93
         ...
359200      1
54900       1
377600      1
81200       1
47000       1
Name: count, Length: 3842, dtype: int64
```

REGRESSION MODEL DEVELOPMENT

Split the data into features (X) and the target (y)

In [35]:
```
X = house.drop('houseprice', axis=1)  # Features (exclude the target column)
y = house['houseprice']  # Target variable
```

Split the data into training and testing sets

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Create a Linear Regression model and fit it to the training data

In [37]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

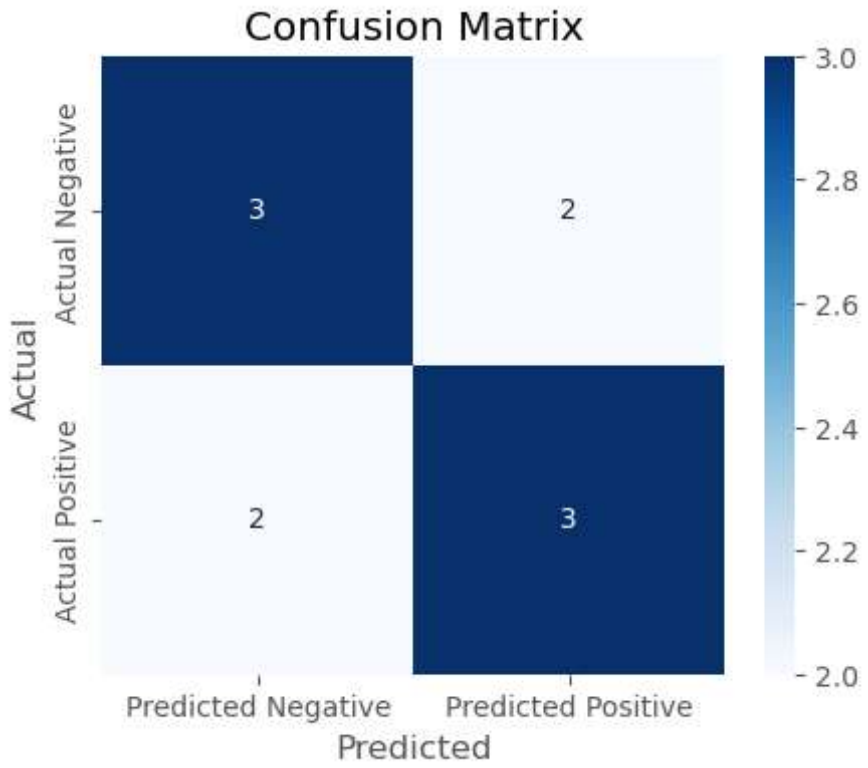
Out[37]:
```
LinearRegression
LinearRegression()
```

Make predictions on the test set

In [38]:
```
y_pred = model.predict(X_test)
```

confusion matrix

In [28]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Actual labels (ground truth)
y_true = [1, 0, 1, 1, 0, 1, 0, 0, 1, 0]

# Predicted labels (model's predictions)
y_pred = [1, 0, 1, 0, 1, 1, 0, 1, 0, 0]

# Compute the confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Create a heatmap for visualization
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted Negative', 'Predicted Positive'],
            yticklabels=['Actual Negative', 'Actual Positive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Evaluate the model's performance

In [39]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 4972764891.781958
R-squared: 0.6205180997418512
```

make predictions on new data by providing the features of a house. For example

In [32]:
```
new_house_features = [-122.26, 37.84, 30, 1400, 600, 220, 4.0]
predicted_price = model.predict([new_house_features])
print(f"Predicted Price: {predicted_price[0]}")
```

```
Predicted Price: 228641.1809624657
```

In [ ]: