House Price Prediction

Importing the required Libraries

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the Dataset

In [6]:
```
data=pd.read_csv("data.csv")
```

Exploratory data analysis on the Dataset

In [41]:
```
data.head()
```

Out[41]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renova
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1

In [8]:
```
data.tail()
```

Out[8]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	2009	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	1974	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	4	1490	0	1990	

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [10]:
```
data.describe()
```

Out[10]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_b
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	460
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240652	3.451739	1827.265435	31
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778405	0.677230	862.168977	46
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	370.000000	
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000	1190.000000	
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000000	3.000000	1590.000000	
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000	2300.000000	61
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	9410.000000	482

In [12]:
```
data.isnull().sum()
```

Out[12]:
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [13]:
```
data.columns
```

Out[13]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

In [15]:
```
data.shape
```

Out[15]:
```
(4600, 18)
```

Linear Regression Algorithm

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Prepare the Data

In [7]:
```
X = data[['sqft_lot', 'floors']]
y = data['price']
```

Split the Data

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Create and Train the Linear Regression Model

In [10]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[10]:
```
▾ LinearRegression
LinearRegression()
```

Make Predictions

In [26]:
```
y_pred = model.predict(X_test)
```

Evaluate the Model

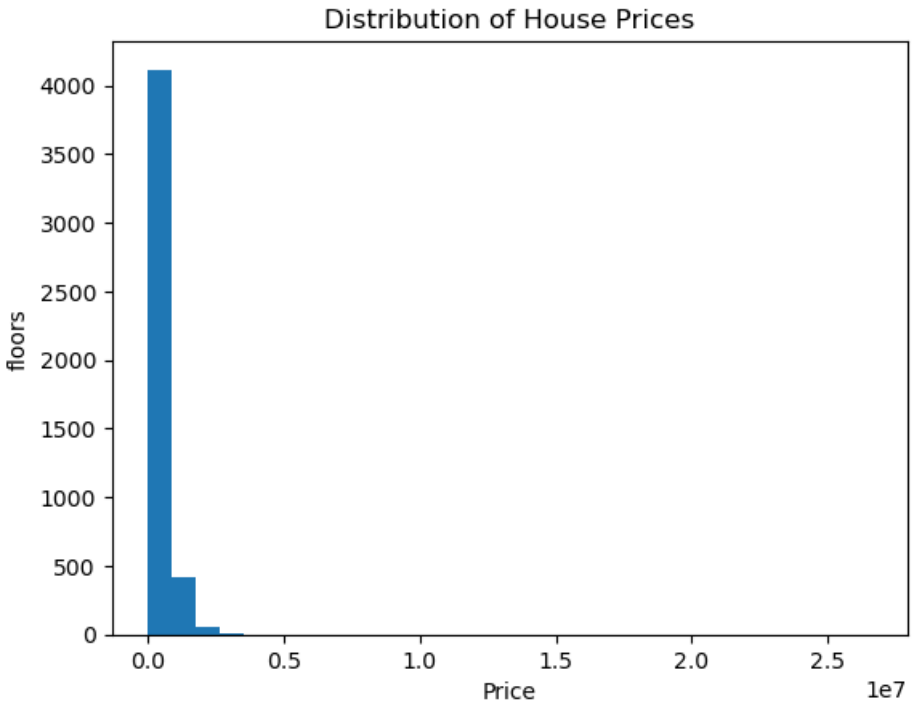
In [27]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

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

```
Mean Squared Error: 1021553025827.8557
Root Mean Squared Error: 1010719.0637500887
R-squared: -0.0016734734183148081
```

Data Visualization

Histogram of House Prices

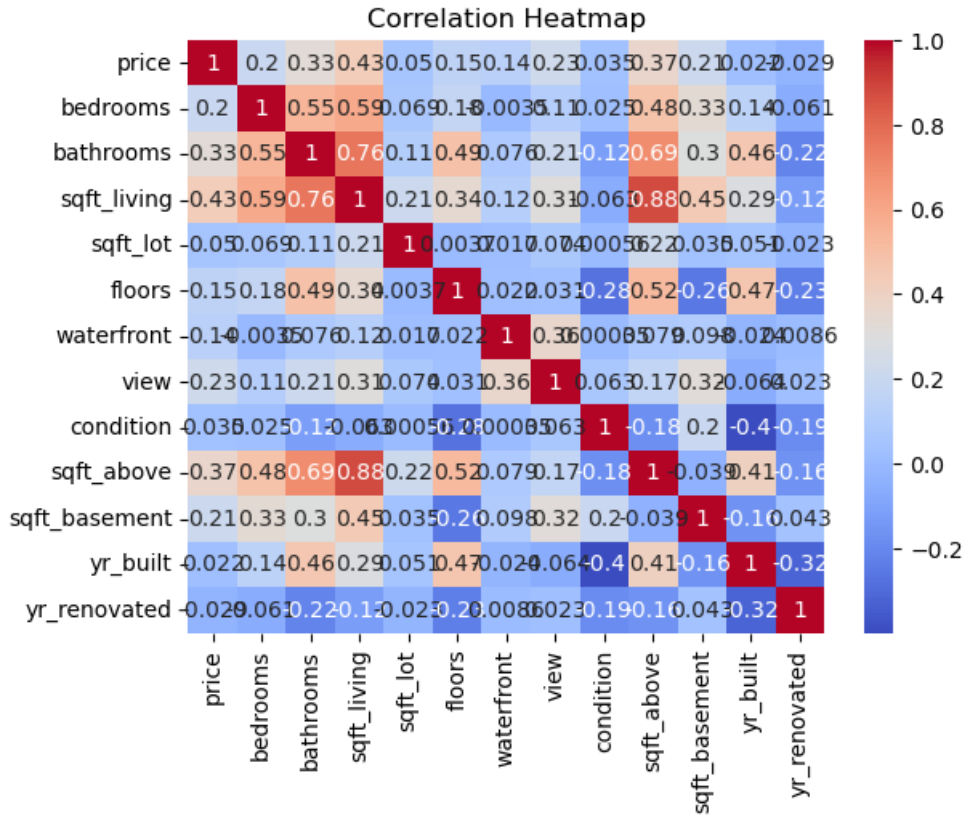
In [36]:
```
plt.hist(data['price'], bins=30)
plt.xlabel('Price')
plt.ylabel('floors')
plt.title('Distribution of House Prices')
plt.show()
```



Correlation Heatmap

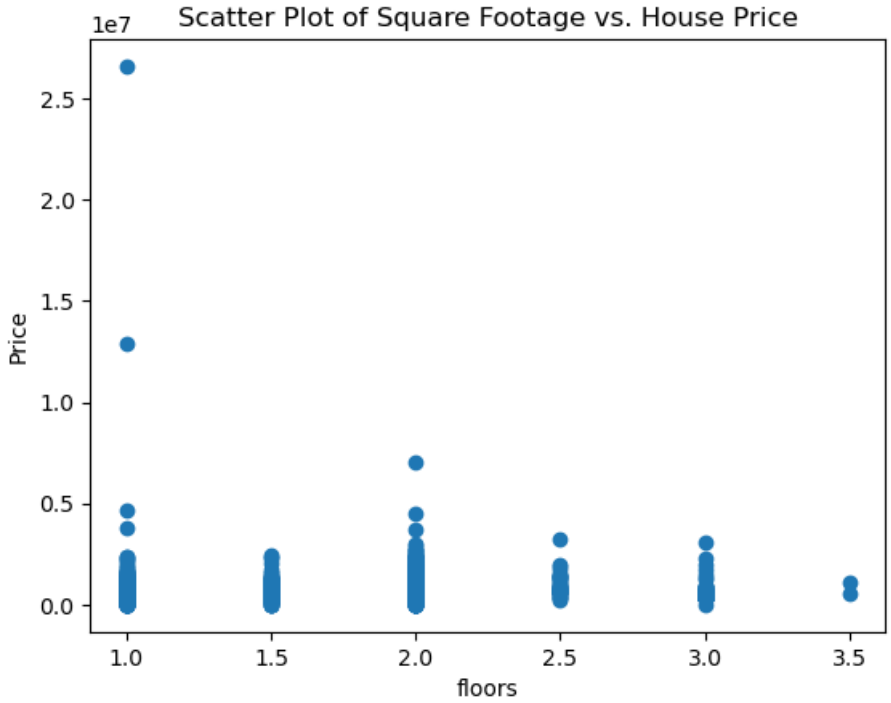
In [29]:
```
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\A.Surya Tejaswini\AppData\Local\Temp\ipykernel_6768\3963569686.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



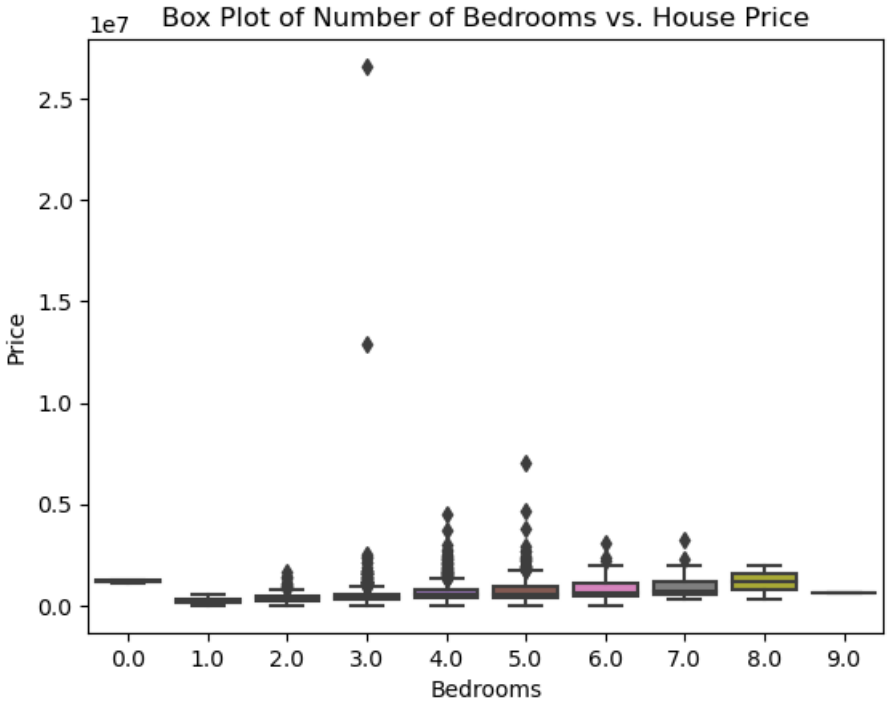
Scatter Plots

In [32]:
```
plt.scatter(data['floors'], data['price'])
plt.xlabel('floors')
plt.ylabel('Price')
plt.title('Scatter Plot of floors vs. Price')
plt.show()
```



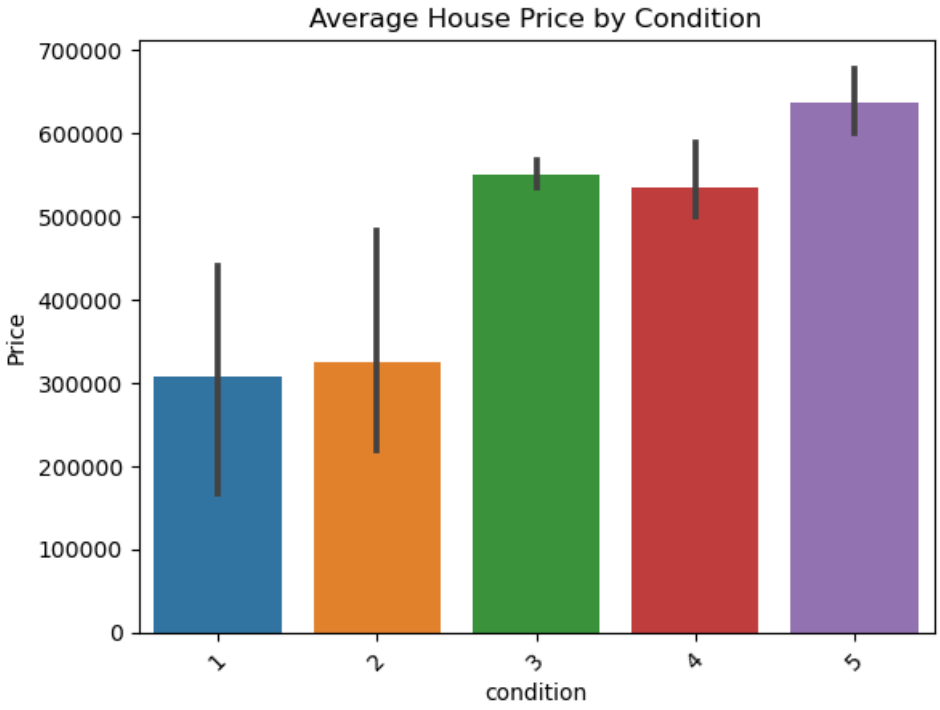
Box Plots

In [34]:
```
sns.boxplot(x='bedrooms', y='price', data=data)
plt.xlabel('Bedrooms')
plt.ylabel('Price')
plt.title('Box Plot of Number of Bedrooms vs. Price')
plt.show()
```



Categorical Plots

In [35]:
```
sns.barplot(x='condition', y='price', data=data)
plt.xlabel('condition')
plt.ylabel('Price')
plt.title('Average House Price by Condition')
plt.xticks(rotation=45)
plt.show()
```



Pair Plots

In [38]:
```
sns.pairplot(data[['sqft_lot', 'bedrooms', 'bathrooms', 'price']])
plt.title('Pair Plot of Numerical Features')
plt.show()
```