In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
df=pd.read_csv("data.csv")

df.head()
```

Out[7]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	viev
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	

In [8]:
```
df.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 [9]:
```
df.shape
```

Out[9]: (4600, 18)

In [10]:
```
df.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 [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
df.nunique()
```

Out[12]:
```
date              70
price           1741
bedrooms          10
bathrooms         26
sqft_living      566
sqft_lot        3113
floors             6
waterfront         2
view               5
condition          5
sqft_above       511
sqft_basement    207
yr_built         115
yr_renovated      60
street          4525
city              44
statezip          77
country            1
dtype: int64
```

In [13]:
```
df.describe()
```

Out[13]:

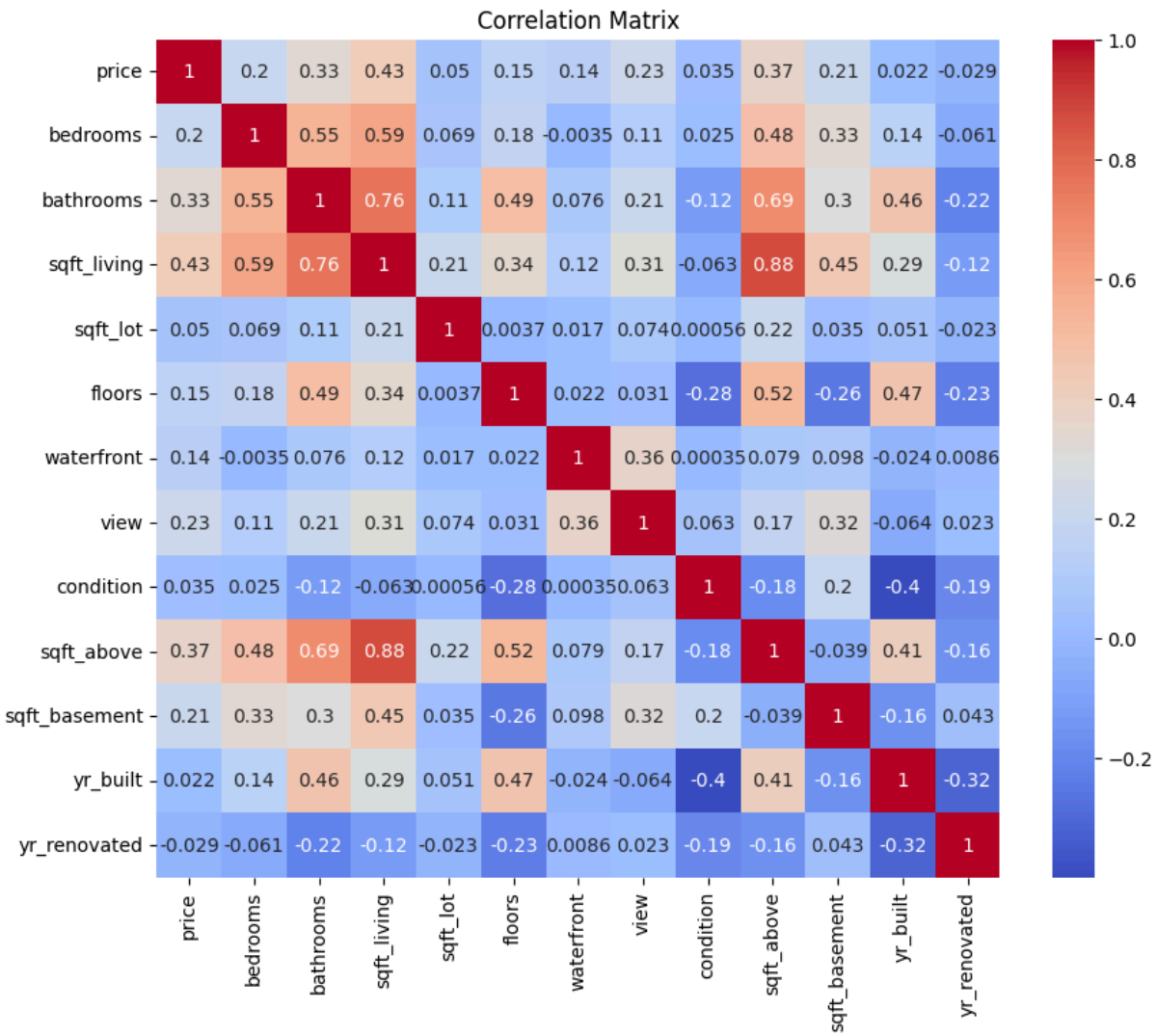
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000

In [15]:
```
print(df.dtypes)
```

```
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

In [16]:
```
df_numeric = df.select_dtypes(include=[float, int])
df_coor = df_numeric.corr()
```

In [18]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df_coor, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [19]:
```
df.columns
```

Out[19]:
```
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 [21]:
```
columns_to_remove = ['date', 'yr_renova', 'street', 'statsize', 'country']
df = df.drop(columns=[col for col in columns_to_remove if col in df.columns])
```

In [22]:
```
df.head()
```

Out[22]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditi
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	

In [40]:
```
df_numeric = df.select_dtypes(include=[float, int])
z_scores = stats.zscore(df_numeric)
```

In [39]:
```
df = df_numeric
```

In [38]:
```
import scipy.stats as stats
z_scores = stats.zscore(df)
threshold=3
print ("size before removing outlines:", df.shape)
outliers_df =df[(z_scores>threshold).any(axis=1)]
df = df[(z_scores<=threshold).all(axis=1)]
print("size after removing outlines:",df.shape)
```

```
size before removing outlines: (0, 13)
size after removing outlines: (0, 13)
```

In [37]:
```
outliers_df.head()
```

Out[37]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condi
8	452500.0	3.0	2.5	2430	88426	1.0	0	0	
11	1400000.0	4.0	2.5	2920	4000	1.5	0	0	
28	675000.0	5.0	2.5	2820	67518	2.0	0	0	
35	604000.0	3.0	2.5	3240	33151	2.0	0	2	
38	403000.0	3.0	2.0	1960	13100	1.0	0	2	

In [34]:
```
print("DataFrame shape:", df.shape)
```

```
DataFrame shape: (0, 13)
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
scaler.fit(df)  # Ensure df is not empty here
df_scaled = pd.DataFrame(scaler.transform(df), columns=df.columns)
```

In [42]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)

df_scaled = pd.DataFrame(scaler.transform(df),columns=df.columns)
```

In [43]:
```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [44]:
```
x= df.drop('price',axis=1)
y= df['price']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
```

In [47]:
```
print("x_train shape:", x_train.shape)
print("x_test shape:", x_test.shape)
```

```
x_train shape: (3450, 12)
x_test shape: (1150, 12)
```

In [57]:
```
model =[
    ('Random Forest', RandomForestRegressor()),
    ('Linear Regression', LinearRegression()),
    ('Decision Tree', DecisionTreeRegressor()),
    ('KNR',KNeighborsRegressor())
]
```

In [58]:
```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
```

In [59]:
```
for name,model in model:
    print(name)
    print()
    model.fit(x_train,y_train)
    y_pred= model.predict(x_test)
    print("mean squared error:",mean_squared_error(y_test,y_pred))
    print('\n')
    print("mean absolute error:",mean_absolute_error(y_test,y_pred))
    print('\n')
    print("R-squared (R2):", r2_score(y_test,y_pred))
    print('\n')
```

```
Random Forest

mean squared error: 806174777990.1157


mean absolute error: 199454.83271332414


R-squared (R2): 0.04290677554992928


Linear Regression

mean squared error: 804480138625.4857


mean absolute error: 203136.06911390202


R-squared (R2): 0.044918656717704564


Decision Tree

mean squared error: 859165295737.1176


mean absolute error: 258198.45520358084


R-squared (R2): -0.020003733288116443


KNR

mean squared error: 816841719514.4725


mean absolute error: 219575.70791233462


R-squared (R2): 0.030242949122588403
```