In [ ]:
```
#I used a dataset from the Kaggle website to analyze real estate listings in Canada:
https://www.kaggle.com/datasets/smmmmmmmmmmmm/canada-real-estate
```

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

# Load the dataset
file_path = 'Real Estate in Canada.xlsx'
df = pd.read_excel(file_path)
df
```

Out[1]:

	Price	Bedrooms	Bathrooms	SqFt	City	Province	Year_Built	Type	Garage	Lot_Area
0	873630	5	2	1010	Montreal	BC	1960	Condo	1	7919
1	377869	2	2	3591	Toronto	ON	1958	House	1	7304
2	128030	4	1	3823	Montreal	ON	2002	House	0	4548
3	117730	3	2	2848	Montreal	QC	1975	Apartment	1	3374
4	292476	4	1	3659	Vancouver	QC	2018	Condo	1	1281
...	...	...	...	...	...	...	...	...	...	...
4995	156065	2	3	1454	Ottawa	ON	1979	Apartment	0	7076
4996	606176	5	2	849	Montreal	ON	1957	Apartment	0	2832
4997	655316	3	3	1711	Ottawa	AB	1993	House	1	3585
4998	258542	4	3	1661	Ottawa	ON	2017	Condo	0	9360
4999	998572	1	2	3180	Vancouver	AB	1980	Condo	1	5740

5000 rows × 10 columns

In [2]:
```
# Display the first few rows of the dataset
print(df.head())
```

```
    Price  Bedrooms  Bathrooms  SqFt       City Province  Year_Built  \
0  873630         5          2  1010   Montreal       BC        1960
1  377869         2          2  3591    Toronto       ON        1958
2  128030         4          1  3823   Montreal       ON        2002
3  117730         3          2  2848   Montreal       QC        1975
4  292476         4          1  3659  Vancouver       QC        2018

        Type  Garage  Lot_Area
0      Condo       1      7919
1      House       1      7304
2      House       0      4548
3  Apartment       1      3374
4      Condo       1      1281
```

In [3]:

```
# Data Cleaning
# Check for missing values
print(df.isnull().sum())
```

```
Price         0
Bedrooms      0
Bathrooms     0
SqFt          0
City          0
Province      0
Year_Built    0
Type          0
Garage        0
Lot_Area      0
dtype: int64
```

In [4]:

```
# Fill or drop missing values (example: fill with mean for numerical columns)
df['Price'] = df['Price'].fillna(df['Price'].mean())
df['Bedrooms'] = df['Bedrooms'].fillna(df['Bedrooms'].mode()[0])
df
```

Out[4]:

	Price	Bedrooms	Bathrooms	SqFt	City	Province	Year_Built	Type	Garage	Lot_Area
0	873630	5	2	1010	Montreal	BC	1960	Condo	1	7919
1	377869	2	2	3591	Toronto	ON	1958	House	1	7304
2	128030	4	1	3823	Montreal	ON	2002	House	0	4548
3	117730	3	2	2848	Montreal	QC	1975	Apartment	1	3374
4	292476	4	1	3659	Vancouver	QC	2018	Condo	1	1281
...	...	...	...	...	...	...	...	...	...	...
4995	156065	2	3	1454	Ottawa	ON	1979	Apartment	0	7076
4996	606176	5	2	849	Montreal	ON	1957	Apartment	0	2832
4997	655316	3	3	1711	Ottawa	AB	1993	House	1	3585
4998	258542	4	3	1661	Ottawa	ON	2017	Condo	0	9360
4999	998572	1	2	3180	Vancouver	AB	1980	Condo	1	5740

5000 rows × 10 columns

In [5]:

```
# Drop any duplicates
df.drop_duplicates(inplace=True)
df
```

Out[5]:

	Price	Bedrooms	Bathrooms	SqFt	City	Province	Year_Built	Type	Garage	Lot_Area
0	873630	5	2	1010	Montreal	BC	1960	Condo	1	7919
1	377869	2	2	3591	Toronto	ON	1958	House	1	7304
2	128030	4	1	3823	Montreal	ON	2002	House	0	4548
3	117730	3	2	2848	Montreal	QC	1975	Apartment	1	3374
4	292476	4	1	3659	Vancouver	QC	2018	Condo	1	1281
...	...	...	...	...	...	...	...	...	...	...
4995	156065	2	3	1454	Ottawa	ON	1979	Apartment	0	7076
4996	606176	5	2	849	Montreal	ON	1957	Apartment	0	2832
4997	655316	3	3	1711	Ottawa	AB	1993	House	1	3585
4998	258542	4	3	1661	Ottawa	ON	2017	Condo	0	9360
4999	998572	1	2	3180	Vancouver	AB	1980	Condo	1	5740

5000 rows × 10 columns

In [6]:

```
# Convert data types if necessary
df['Price'] = df['Price'].astype(float)
df
```

Out[6]:

	Price	Bedrooms	Bathrooms	SqFt	City	Province	Year_Built	Type	Garage	Lot_Area
0	873630.0	5	2	1010	Montreal	BC	1960	Condo	1	7919
1	377869.0	2	2	3591	Toronto	ON	1958	House	1	7304
2	128030.0	4	1	3823	Montreal	ON	2002	House	0	4548
3	117730.0	3	2	2848	Montreal	QC	1975	Apartment	1	3374
4	292476.0	4	1	3659	Vancouver	QC	2018	Condo	1	1281
...	...	...	...	...	...	...	...	...	...	...
4995	156065.0	2	3	1454	Ottawa	ON	1979	Apartment	0	7076
4996	606176.0	5	2	849	Montreal	ON	1957	Apartment	0	2832
4997	655316.0	3	3	1711	Ottawa	AB	1993	House	1	3585
4998	258542.0	4	3	1661	Ottawa	ON	2017	Condo	0	9360
4999	998572.0	1	2	3180	Vancouver	AB	1980	Condo	1	5740

5000 rows × 10 columns

In [7]:

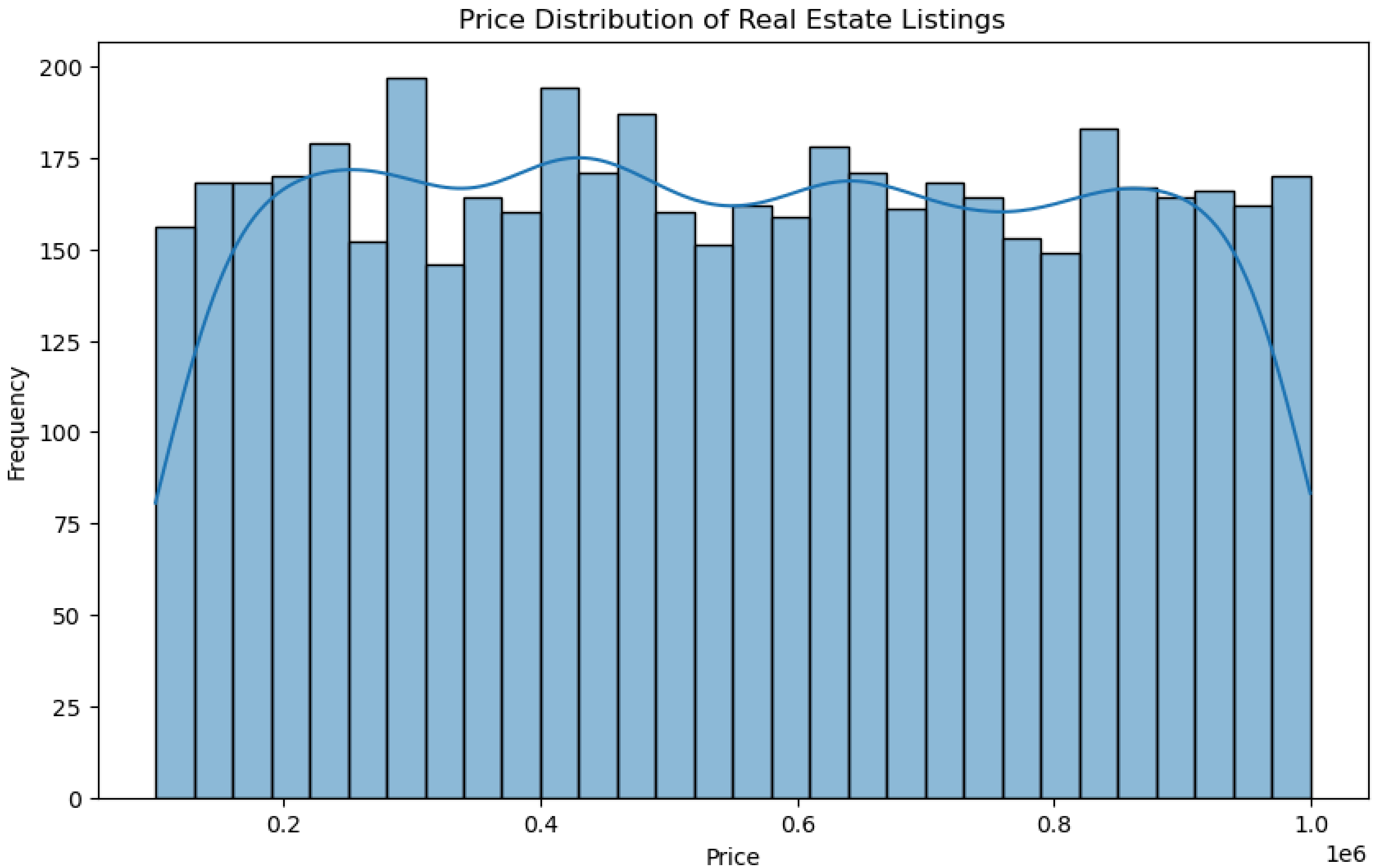
```
# Exploratory Data Analysis (EDA)
# Descriptive statistics
print(df.describe())
```

```
               Price     Bedrooms    Bathrooms         SqFt   Year_Built  \
count    5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean   548136.640000     3.025200     2.003400  2383.556800  1985.685200
std    259134.352521     1.418932     0.822875   925.070872    20.954893
min    100268.000000     1.000000     1.000000   800.000000  1950.000000
25%    321943.000000     2.000000     1.000000  1572.500000  1968.000000
50%    544006.000000     3.000000     2.000000  2381.000000  1985.000000
75%    773302.250000     4.000000     3.000000  3187.000000  2004.000000
max    999361.000000     5.000000     3.000000  3999.000000  2022.000000

            Garage    Lot_Area
count  5000.000000  5000.00000
mean      0.510400  5508.07240
std       0.499942  2585.34411
min       0.000000  1000.00000
25%       0.000000  3259.00000
50%       1.000000  5530.00000
75%       1.000000  7755.50000
max       1.000000  9999.00000
```

In [8]:

```python
# Visualizations
# Price distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], bins=30, kde=True)
plt.title('Price Distribution of Real Estate Listings')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



In [9]:

```
# Print the column names to check for any non-numeric columns
print("Columns in the DataFrame:", df.columns)

# Check the first few rows to understand the data types
print(df.head())

# Identify non-numeric columns
non_numeric_cols = df.select_dtypes(exclude=['number']).columns
print("Non-numeric columns:", non_numeric_cols)
```

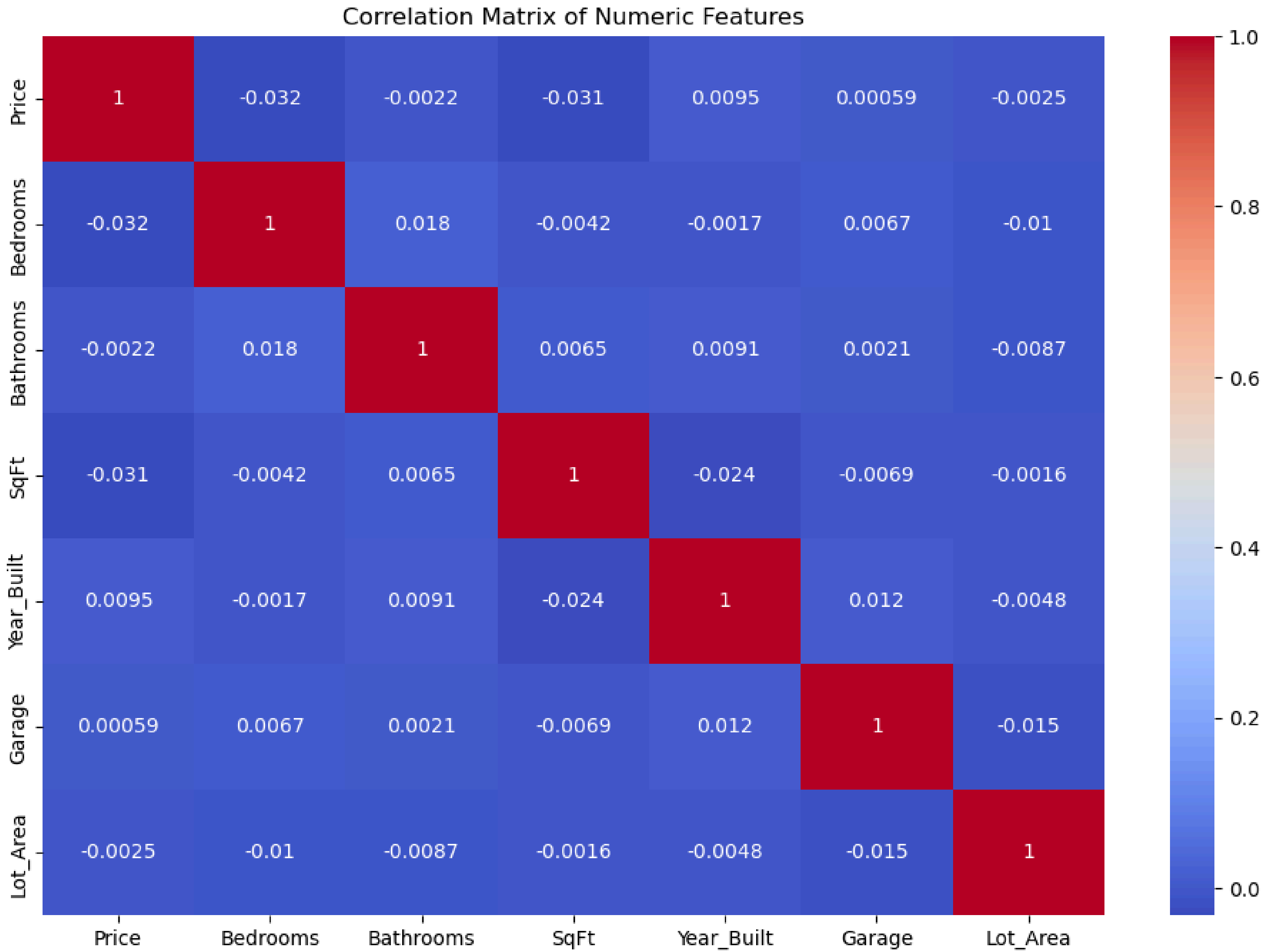
```python
# Option 1: Drop non-numeric columns for correlation matrix
df_numeric = df.select_dtypes(include=['number'])

# Option 2: Convert categorical columns to numeric if needed
# Calculate the correlation matrix
correlation_matrix = df_numeric.corr()

# Plot the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Numeric Features')
plt.show()
```

```
Columns in the DataFrame: Index(['Price', 'Bedrooms', 'Bathrooms', 'SqFt', 'City', 'Province',
       'Year_Built', 'Type', 'Garage', 'Lot_Area'],
      dtype='object')
      Price  Bedrooms  Bathrooms  SqFt       City Province  Year_Built  \
0  873630.0         5          2  1010   Montreal       BC        1960
1  377869.0         2          2  3591    Toronto       ON        1958
2  128030.0         4          1  3823   Montreal       ON        2002
3  117730.0         3          2  2848   Montreal       QC        1975
4  292476.0         4          1  3659  Vancouver       QC        2018

        Type  Garage  Lot_Area
0      Condo       1      7919
1      House       1      7304
2      House       0      4548
3  Apartment       1      3374
4      Condo       1      1281
Non-numeric columns: Index(['City', 'Province', 'Type'], dtype='object')
```



In [10]:

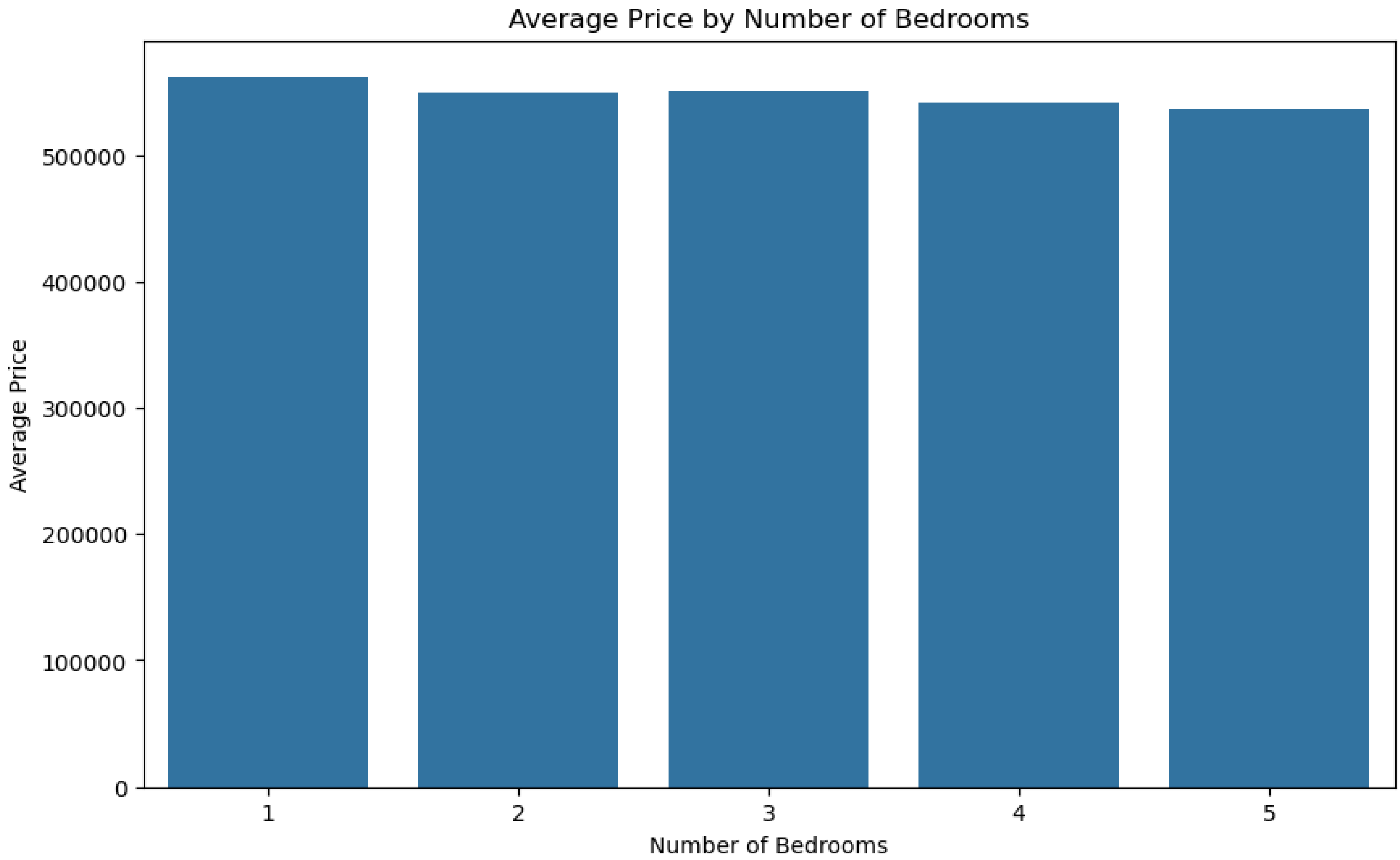
```
# Analysis: Finding the average price by number of bedrooms
average_price_by_bedrooms = df.groupby('Bedrooms')['Price'].mean().reset_index()
```

```
print(average_price_by_bedrooms)
```

```
   Bedrooms          Price
0         1  562423.896866
1         2  549350.161885
2         3  551145.143438
3         4  541438.417969
4         5  537034.622568
```

In [11]:

```
# Visualization of average price by number of bedrooms
plt.figure(figsize=(10, 6))
sns.barplot(x='Bedrooms', y='Price', data=average_price_by_bedrooms)
plt.title('Average Price by Number of Bedrooms')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Average Price')
plt.show()
```



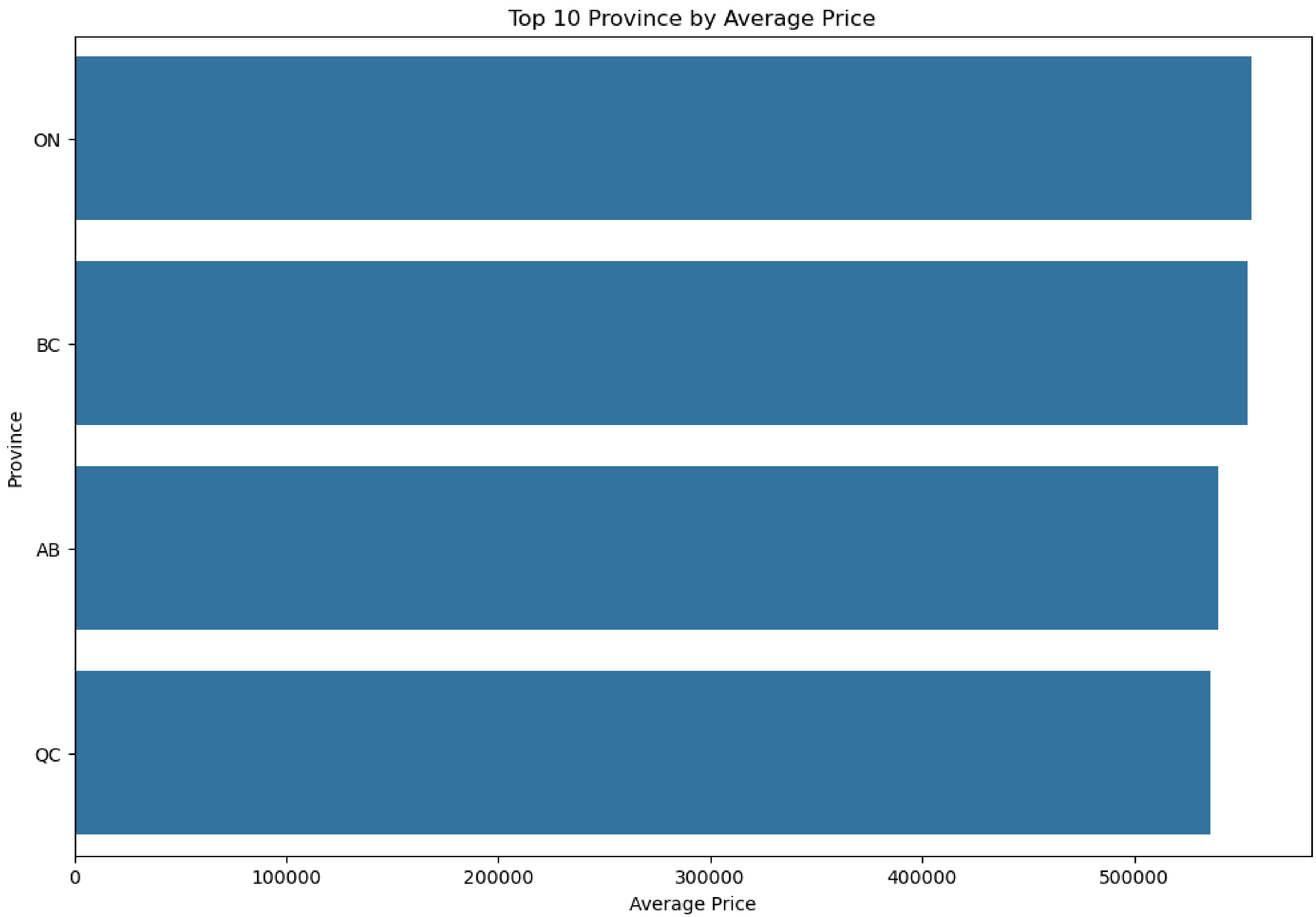
In [12]:

```
# Additional analysis (e.g., average price by province, city, year built, etc.)
average_price_by_province = df.groupby('Province')['Price'].mean().reset_index()
print(average_price_by_province.sort_values(by='Price', ascending=False))
average_price_by_city = df.groupby('City')['Price'].mean().reset_index()
print(average_price_by_city.sort_values(by='Price', ascending=False))
```

```
  Province          Price
2       ON  555588.997950
1       BC  553761.959770
0       AB  539891.212876
3       QC  536167.313820
        City          Price
4  Vancouver  552841.888247
1   Montreal  550167.140102
3    Toronto  547786.426667
0    Calgary  546874.241011
2     Ottawa  542747.509764
```

In [13]:

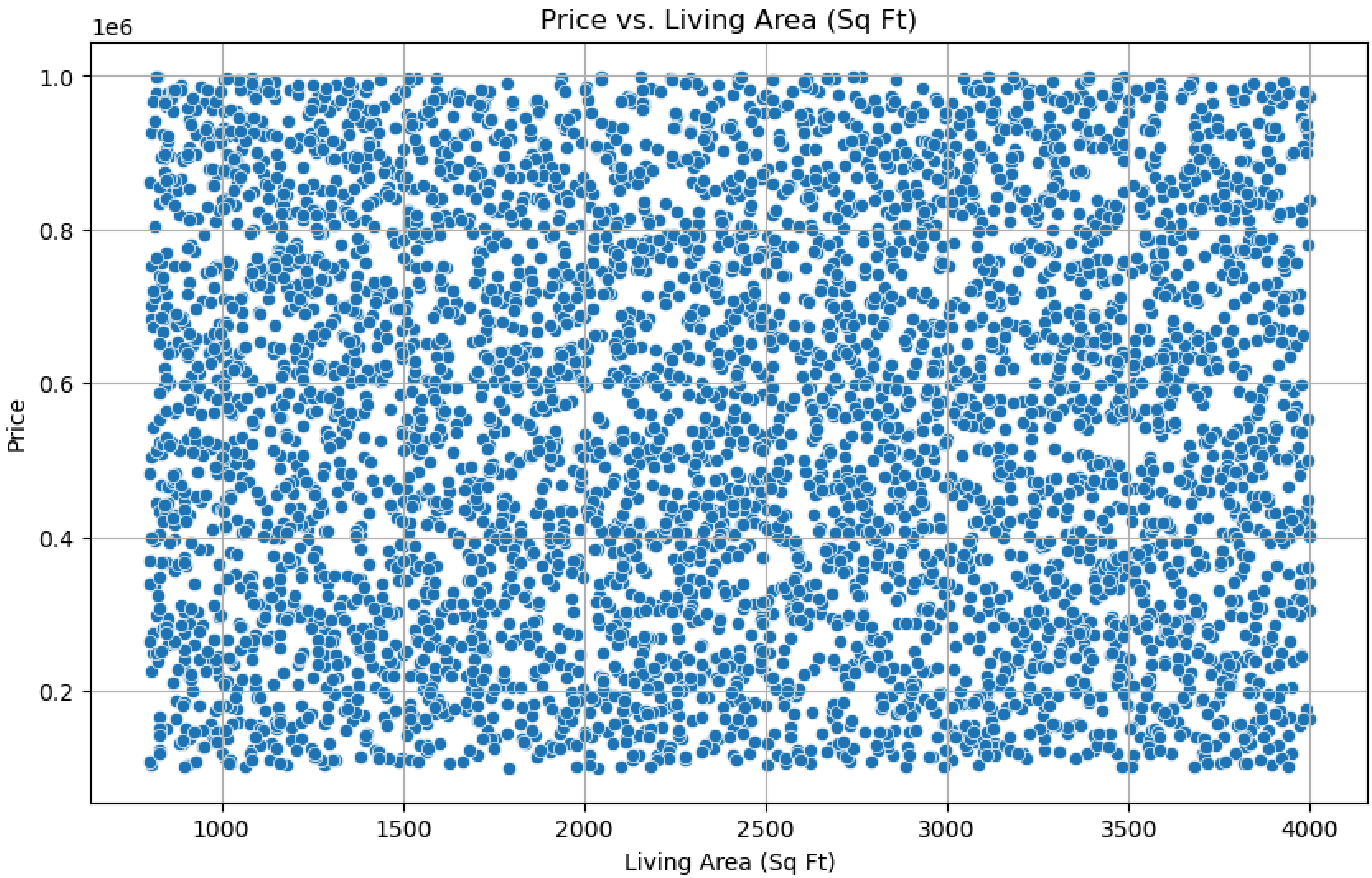
```python
# Visualization of average price by province
plt.figure(figsize=(12, 8))
sns.barplot(x='Price', y='Province', data=average_price_by_province.sort_values(by='Price', ascending=False).head(10)
plt.title('Top 10 Province by Average Price')
plt.xlabel('Average Price')
plt.ylabel('Province')
plt.show()
```



In [21]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Scatter plot for Price vs. SqFt
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='SqFt', y='Price')
```

```
plt.title('Price vs. Living Area (Sq Ft)')
plt.xlabel('Living Area (Sq Ft)')
plt.ylabel('Price')
plt.grid()
plt.show()
```



In [23]:
```
# Check the data types of each column
print(df.dtypes)
```

```
Price          float64
Bedrooms         int64
Bathrooms        int64
SqFt             int64
City            object
Province        object
Year_Built       int64
Type            object
Garage           int64
Lot_Area         int64
dtype: object
```

In [26]:

```python
# One-hot encode categorical variables
df_encoded = pd.get_dummies(df, drop_first=True)

# Now define features and target again
X = df_encoded.drop('Price', axis=1)  # All columns except target
y = df_encoded['Price']  # Target variable

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

```
Mean Squared Error: 69228408196.02744
R^2 Score: -0.00492490646235666
```

In [30]:

```
print(df.columns)
```

```
Index(['Price', 'Bedrooms', 'Bathrooms', 'SqFt', 'City', 'Province',
       'Year_Built', 'Type', 'Garage', 'Lot_Area'],
      dtype='object')
```

In [37]:

```
# Example: Recommend houses below a certain predicted price threshold
predicted_prices = model.predict(X)  # Predict prices for the whole dataset
df['Price'] = predicted_prices  # Add predicted prices to the DataFrame

# Recommend houses below a certain price threshold (e.g., 300,000)
recommended_houses = df[df['Price'] < 500000]  # Adjust the price limit as needed

# Print the recommended houses with the correct column names
print(recommended_houses[['Price', 'Bedrooms', 'Bathrooms', 'SqFt', 'City', 'Province',
       'Year_Built', 'Type', 'Garage', 'Lot_Area']])
```

	Price	Bedrooms	Bathrooms	SqFt	City	Province	Year_Built	\
20	498607.048446	4	3	3154	Ottawa	QC	2016	
87	499774.757926	5	1	2523	Montreal	QC	1959	
118	490352.457248	5	2	3295	Ottawa	QC	1983	
123	495997.992884	5	3	3540	Toronto	QC	2019	
183	499496.383001	5	2	3316	Calgary	QC	1982	
366	499049.320248	5	3	2881	Montreal	QC	2002	
549	498815.947380	5	2	3658	Calgary	QC	2013	
763	489275.404680	5	3	2774	Montreal	QC	1966	
837	485116.222206	4	3	3884	Ottawa	QC	1976	
941	495213.357722	3	3	3660	Montreal	QC	1953	
961	493845.130008	5	3	2404	Toronto	QC	1953	
1196	489405.081867	4	3	3617	Montreal	QC	1965	
1329	498551.972084	5	2	3336	Calgary	QC	1965	
1377	489884.403444	4	2	3481	Ottawa	QC	1957	
1563	492905.209446	5	2	3180	Toronto	QC	1957	
1783	494857.185017	5	3	3399	Ottawa	QC	2021	
1823	490042.865507	5	2	3844	Calgary	QC	1959	
1884	495592.779176	5	1	3738	Toronto	QC	1981	
2216	494890.389888	5	1	3948	Montreal	QC	1995	
2246	495642.849427	4	3	3746	Montreal	QC	1975	
2284	497495.722994	5	3	2835	Montreal	QC	2006	
2369	499757.125543	5	3	3582	Calgary	QC	1994	
2398	494742.308661	4	1	3652	Montreal	QC	1952	
2409	497472.413763	5	2	3274	Montreal	QC	2001	
2425	495551.303464	5	3	3967	Calgary	QC	2008	
2487	499022.501019	5	2	2060	Montreal	QC	1957	
2608	487324.902617	5	3	3739	Montreal	QC	1995	
2632	491559.412678	3	2	3974	Ottawa	QC	1958	
2707	492842.534893	4	2	3532	Toronto	QC	1955	
2731	493111.330121	5	2	3840	Toronto	QC	1993	
3031	491085.845709	4	2	3886	Toronto	QC	1960	
3129	491341.940366	4	3	3450	Montreal	QC	1965	
3142	497338.687353	3	1	3928	Ottawa	QC	1961	
3566	496976.953812	5	2	3375	Toronto	QC	1986	
3963	497071.158376	5	1	3431	Montreal	QC	1974	

	Type	Garage	Lot_Area
20	House	1	9419
87	House	1	6676
118	House	1	7040
123	House	0	5607

183	House	0	4205
366	House	0	2260
549	House	0	8230
763	House	0	9552
837	House	0	8214
941	House	1	7372
961	House	1	9950
1196	House	0	7695
1329	House	1	3641
1377	House	1	8519
1563	House	0	4430
1783	House	0	1330
1823	House	1	8337
1884	House	0	3431
2216	House	0	1924
2246	House	1	1514
2284	House	0	6109
2369	House	1	1209
2398	House	1	7389
2409	House	0	3741
2425	House	1	5862
2487	House	1	8833
2608	House	1	9354
2632	House	1	9326
2707	House	0	8906
2731	House	1	6605
3031	House	0	8174
3129	House	0	6958
3142	House	1	6462
3566	House	0	2806
3963	House	1	2642

In [39]:

```
def recommend_houses(df, max_price, min_bedrooms, min_bathrooms):
    recommendations = df[
        (df['Price'] <= max_price) &
        (df['Bedrooms'] >= min_bedrooms) &
        (df['Bathrooms'] >= min_bathrooms)
    ]

    return recommendations

# Example usage
```

```python
recommended_houses = recommend_houses(df, max_price=500000, min_bedrooms=2, min_bathrooms=1)
print(recommended_houses[['Price', 'Bedrooms', 'Bathrooms', 'City', 'Province']])
```

```
               Price  Bedrooms  Bathrooms      City Province
20     498607.048446         4          3    Ottawa       QC
87     499774.757926         5          1  Montreal       QC
118    490352.457248         5          2    Ottawa       QC
123    495997.992884         5          3   Toronto       QC
183    499496.383001         5          2   Calgary       QC
366    499049.320248         5          3  Montreal       QC
549    498815.947380         5          2   Calgary       QC
763    489275.404680         5          3  Montreal       QC
837    485116.222206         4          3    Ottawa       QC
941    495213.357722         3          3  Montreal       QC
961    493845.130008         5          3   Toronto       QC
1196   489405.081867         4          3  Montreal       QC
1329   498551.972084         5          2   Calgary       QC
1377   489884.403444         4          2    Ottawa       QC
1563   492905.209446         5          2   Toronto       QC
1783   494857.185017         5          3    Ottawa       QC
1823   490042.865507         5          2   Calgary       QC
1884   495592.779176         5          1   Toronto       QC
2216   494890.389888         5          1  Montreal       QC
2246   495642.849427         4          3  Montreal       QC
2284   497495.722994         5          3  Montreal       QC
2369   499757.125543         5          3   Calgary       QC
2398   494742.308661         4          1  Montreal       QC
2409   497472.413763         5          2  Montreal       QC
2425   495551.303464         5          3   Calgary       QC
2487   499022.501019         5          2  Montreal       QC
2608   487324.902617         5          3  Montreal       QC
2632   491559.412678         3          2    Ottawa       QC
2707   492842.534893         4          2   Toronto       QC
2731   493111.330121         5          2   Toronto       QC
3031   491085.845709         4          2   Toronto       QC
3129   491341.940366         4          3  Montreal       QC
3142   497338.687353         3          1    Ottawa       QC
3566   496976.953812         5          2   Toronto       QC
3963   497071.158376         5          1  Montreal       QC
```

In [41]:

```python
# Recommendations based on analysis
recommendations = []
```

```python
# Example recommendations based on findings
if average_price_by_bedrooms['Price'].max() > 500000:
    recommendations.append("Consider targeting listings with 2-3 bedrooms as they yield higher average prices.")

if average_price_by_province['Price'].max() > 600000:
    recommendations.append("Focus marketing efforts in high-demand areas with average prices exceeding $600,000.")

if average_price_by_city['Price'].max() > 600000:
    recommendations.append("Focus marketing efforts in high-demand areas with average prices exceeding $600,000.")

print("Recommendations:")
for recommendation in recommendations:
    print(f"- {recommendation}")
```

```
Recommendations:
- Consider targeting listings with 2-3 bedrooms as they yield higher average prices.
```

Recommendations for House Listings in Canada

. Focus on 2-3 Bedroom Listings:

Action: We should prioritize marketing and promoting listings that feature 2-3 bedrooms, as these properties tend to yield higher average prices.

This segment appeals to a broad audience, including young families, first-time homebuyers, and downsizers looking for a balance of space and affordability.

. Highlight Properties with Attractive Features:

_ Action: We need to ensure that our listings emphasize desirable features such as updated kitchens, outdoor spaces, and proximity to amenities like schools and parks.

Rationale: Features that enhance living quality can justify higher price points and attract buyers willing to pay a premium for convenience and comfort.

. Market Homes in High-Demand Areas:

_ Action: Let's identify neighborhoods or cities where 2-3 bedroom homes are particularly sought after and concentrate our marketing efforts there. We should highlight local amenities, schools, and community features in our listings.

_ Rationale: Focusing on high-demand areas can increase visibility and attract more potential buyers, leading to quicker sales.

. Offer Virtual Tours and Open Houses:

_ Action: We should implement virtual tours and host open houses for our 2-3 bedroom listings to reach a wider audience and provide an immersive viewing experience.

_ Rationale: This approach can attract remote buyers and create a sense of urgency, encouraging potential buyers to act quickly.

_ Consider Price Competitiveness:

_ Action: We must regularly analyze the pricing of 2-3 bedroom homes in the market to ensure our listings are competitively priced. We can also consider offering incentives or flexible financing options to attract buyers.

_ Rationale: Competitive pricing can make our listings more attractive and can lead to increased interest and faster sales.

. Leverage Social Media and Online Marketing:

_ Action: We should utilize social media platforms and online real estate marketing to target specific demographics likely to be interested in 2–3-bedroom homes. Using targeted ads, we can reach potential buyers based on age, income, and location.

_ Rationale: Online marketing can help us reach a wider audience and generate more leads by targeting those specifically searching for homes in this category.

. Gather and Use Customer Feedback:

_ Action: We need to collect feedback from potential buyers about their preferences in home features and locations and adjust our marketing strategies accordingly.

_ Rationale: Understanding customer preferences can help us tailor our offerings and ensure we meet the needs of our target market effectively.

. General Overview

The recommended houses generally fall within a price range of approximately **$485,000 to $499,800**, offering a variety of bedrooms and bathrooms. The majority of listings are concentrated in major urban centers like **Montreal**, **Ottawa**, **Toronto**, and **Calgary**.

Top Recommendations

1. **Ottawa, QC**
    - **Price:** $498,607
        - **Bedrooms:** 4
        - **Bathrooms:** 3
        - Ideal for families seeking spacious homes in a capital city with access to amenities and public services.
    - **Price:** $489,884
        - **Bedrooms:** 4
        - **Bathrooms:** 2

        - A more budget-friendly option, still offering significant space for families.
2. **Montreal, QC**
    - **Price:** $499,775
        - **Bedrooms:** 5
        - **Bathrooms:** 1
        - Great for larger families or those wanting extra space for home offices or hobbies.
    - **Price:** $495,213
        - **Bedrooms:** 3
        - **Bathrooms:** 3
        - This property combines a moderate price with ample bedrooms and bathrooms, making it suitable for families or shared living situations.
3. **Toronto, QC**
    - **Price:** $495,998
        - **Bedrooms:** 5
        - **Bathrooms:** 3
        - An excellent option for buyers wanting to live in one of Canada's most vibrant cities while having sufficient space for a family.
    - **Price:** $492,843
        - **Bedrooms:** 4
        - **Bathrooms:** 2
        - This listing offers a blend of comfort and convenience, ideal for professionals or families looking to settle in Toronto.

4. **Calgary, QC**
    - **Price:** $499,496
        - **Bedrooms:** 5
        - **Bathrooms:** 2
        - Perfect for those seeking a balance between urban living and proximity to nature.
    - **Price:** $495,642
        - **Bedrooms:** 4
        - **Bathrooms:** 3
        - A spacious family home with modern amenities, suitable for families seeking community and outdoor activities.

Key Insights

- **Room Preferences:** Most of the recommended properties feature **4 or more bedrooms**, making them ideal for families or those needing extra space.
- **Bathroom Counts:** Many listings also provide **2 or more bathrooms**, enhancing comfort for larger households.
- **Location Advantages:** Major cities like **Montreal**, **Ottawa**, **Toronto**, and **Calgary** offer diverse cultural experiences, excellent education options, and strong job markets, making these locations highly desirable for potential buyers.