project_house_price

April 22, 2025

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

sns.set(style="whitegrid")
```

1 King County House Price Analysis

Dataset: House Sales in King County, USA
Source: https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

2 2. Load Dataset

[8]:
```python
df = pd.read_csv("kc_house_data.csv")
```

3 Quick glance at the data

[11]:
```python
df.head()
```

[11]:
```
           id             date     price  bedrooms  bathrooms  sqft_living  \
0  7129300520  20141013T000000  221900.0         3       1.00         1180
1  6414100192  20141209T000000  538000.0         3       2.25         2570
2  5631500400  20150225T000000  180000.0         2       1.00          770
3  2487200875  20141209T000000  604000.0         4       3.00         1960
4  1954400510  20150218T000000  510000.0         3       2.00         1680

   sqft_lot  floors  waterfront  view  …  grade  sqft_above  sqft_basement  \
0      5650     1.0           0     0  …      7        1180              0
1      7242     2.0           0     0  …      7        2170            400
2     10000     1.0           0     0  …      6         770              0
3      5000     1.0           0     0  …      7        1050            910
4      8080     1.0           0     0  …      8        1680              0

   yr_built  yr_renovated  zipcode      lat     long  sqft_living15  \
0      1955             0    98178  47.5112 -122.257           1340
```

```
1      1951          1991    98125  47.7210 -122.319           1690
2      1933             0    98028  47.7379 -122.233           2720
3      1965             0    98136  47.5208 -122.393           1360
4      1987             0    98074  47.6168 -122.045           1800

   sqft_lot15
0        5650
1        7639
2        8062
3        5000
4        7503

[5 rows x 21 columns]
```

3.1 3. Data Overview

4 Data structure

[15]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

5 Summary statistics

[19]:
```
df.describe()
```

[19]:

	id	price	bedrooms	bathrooms	sqft_living
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000

	sqft_lot	floors	waterfront	view	condition
count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000
mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430
std	4.142051e+04	0.539989	0.086517	0.766318	0.650743
min	5.200000e+02	1.000000	0.000000	0.000000	1.000000
25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000
50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000
75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000
max	1.651359e+06	3.500000	1.000000	4.000000	5.000000

	grade	sqft_above	sqft_basement	yr_built	yr_renovated
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	7.656873	1788.390691	291.509045	1971.005136	84.402258
std	1.175459	828.090978	442.575043	29.373411	401.679240
min	1.000000	290.000000	0.000000	1900.000000	0.000000
25%	7.000000	1190.000000	0.000000	1951.000000	0.000000
50%	7.000000	1560.000000	0.000000	1975.000000	0.000000
75%	8.000000	2210.000000	560.000000	1997.000000	0.000000
max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000

	zipcode	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	98077.939805	47.560053	-122.213896	1986.552492	12768.455652
std	53.505026	0.138564	0.140828	685.391304	27304.179631
min	98001.000000	47.155900	-122.519000	399.000000	651.000000
25%	98033.000000	47.471000	-122.328000	1490.000000	5100.000000
50%	98065.000000	47.571800	-122.230000	1840.000000	7620.000000
75%	98118.000000	47.678000	-122.125000	2360.000000	10083.000000
max	98199.000000	47.777600	-121.315000	6210.000000	871200.000000

5.1 4. Data Cleaning

5.1.1 Check for missing values

[23]:
```
print(df.isnull().sum())
```

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

6 Drop duplicates

[26]:
```
df.drop_duplicates(inplace=True)
```

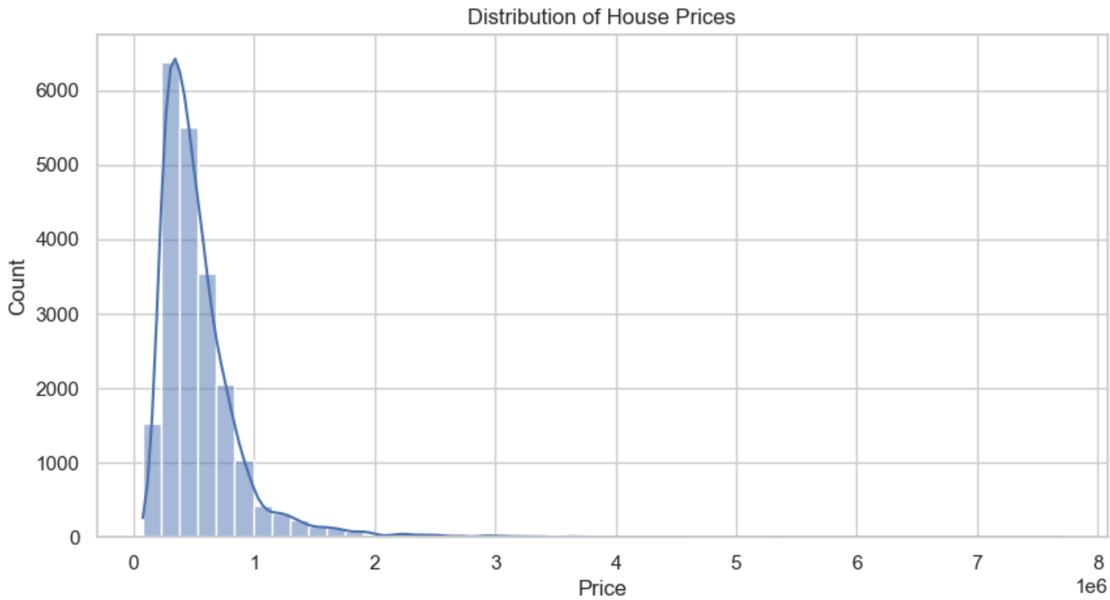
7 Convert date to datetime

[31]:
```
df['date'] = pd.to_datetime(df['date'])
```

7.1 5. Exploratory Data Analysis (EDA)

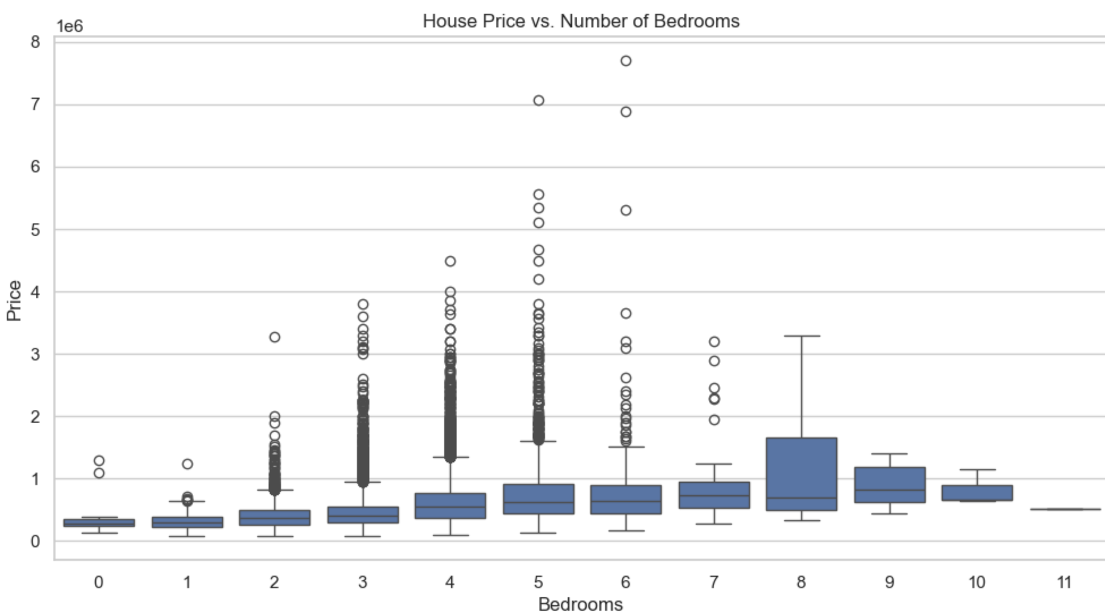
7.1.1 5.1 House Price Distribution

[35]:
```
plt.figure(figsize=(10,5))
sns.histplot(df['price'], bins=50, kde=True)
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()
```



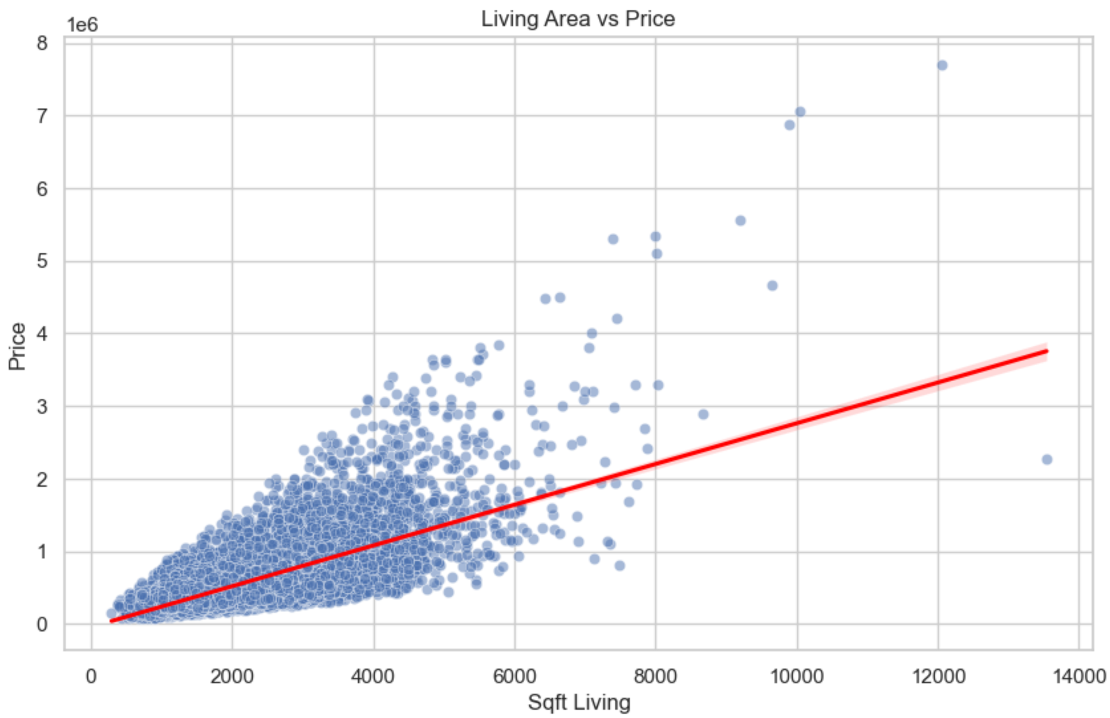
7.1.2 5.2 Bedrooms vs Price

[38]:
```
plt.figure(figsize=(12,6))
sns.boxplot(x='bedrooms', y='price', data=df[df['bedrooms'] < 12])
plt.title('House Price vs. Number of Bedrooms')
plt.xlabel('Bedrooms')
plt.ylabel('Price')
plt.show()
```



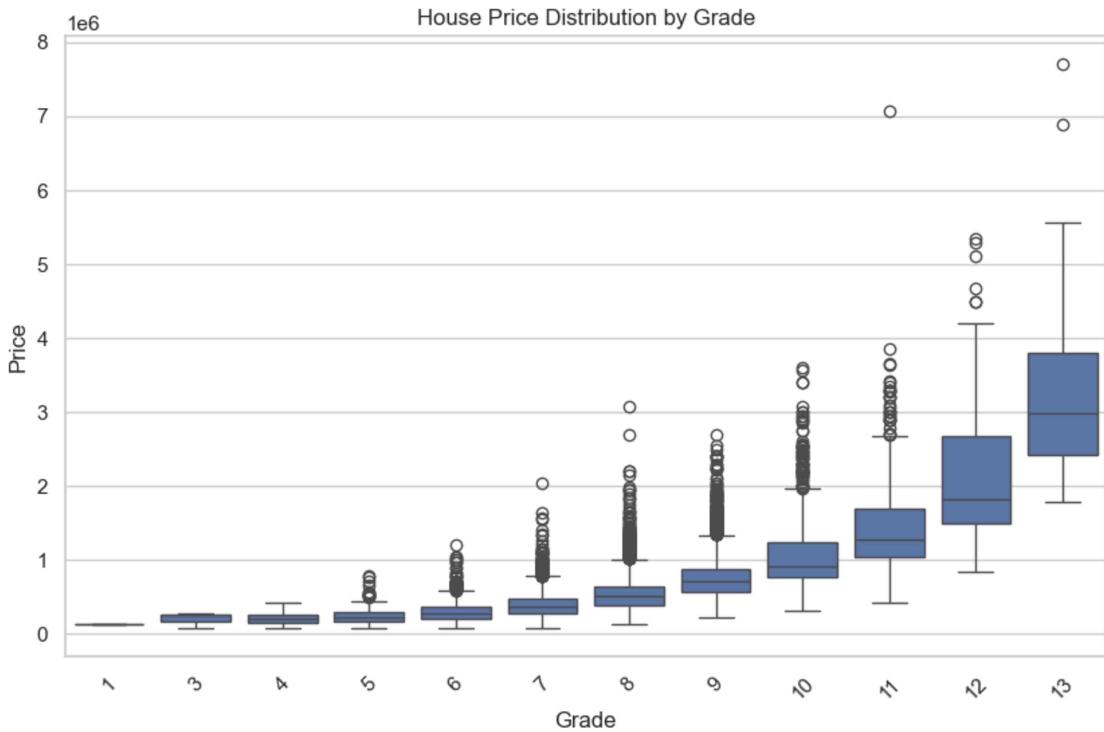
7.1.3 5.3 Living Area vs Price

[41]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(x='sqft_living', y='price', data=df, alpha=0.5)
sns.regplot(x='sqft_living', y='price', data=df, scatter=False, color='red')
plt.title('Living Area vs Price')
plt.xlabel('Sqft Living')
plt.ylabel('Price')
plt.show()
```



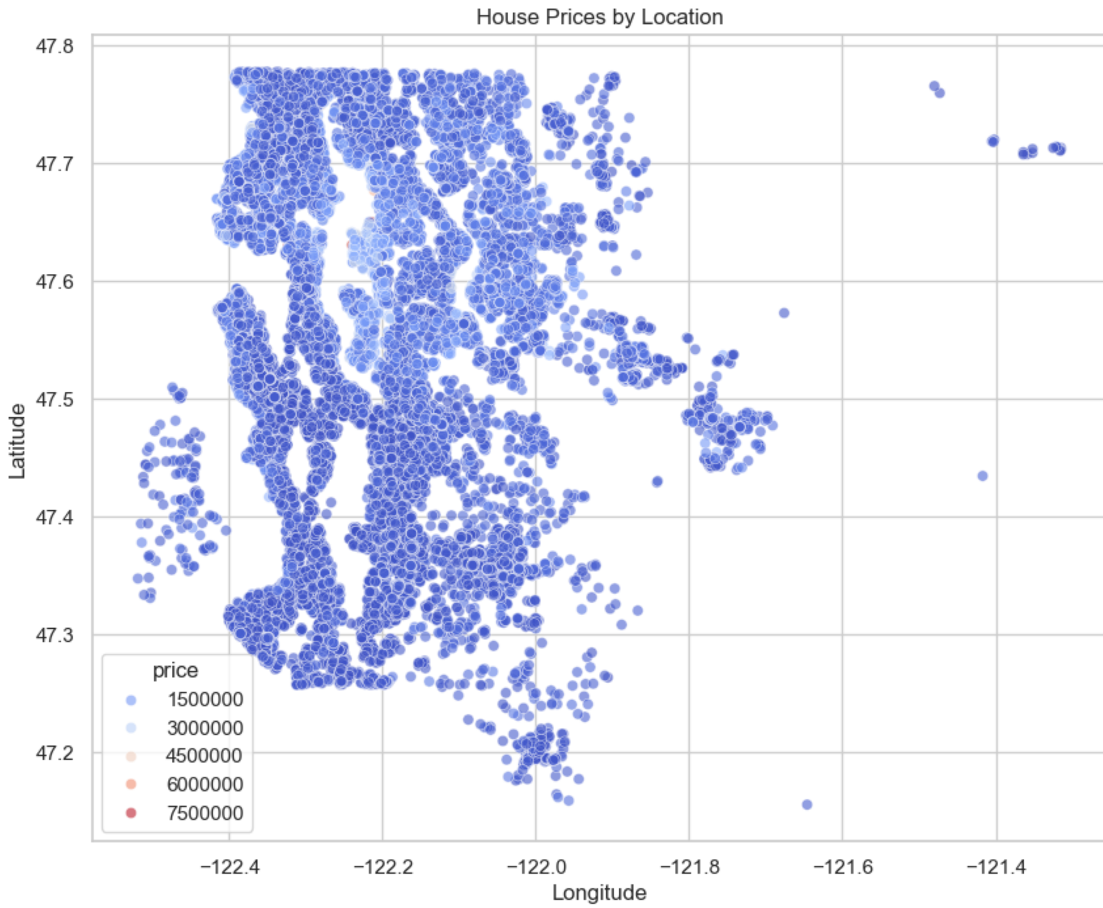
7.1.4 5.4 Grade vs Price

[44]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='grade', y='price', data=df)
plt.title('House Price Distribution by Grade')
plt.xlabel('Grade')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```



7.1.5 5.5 Price by Location (Latitude vs Longitude)

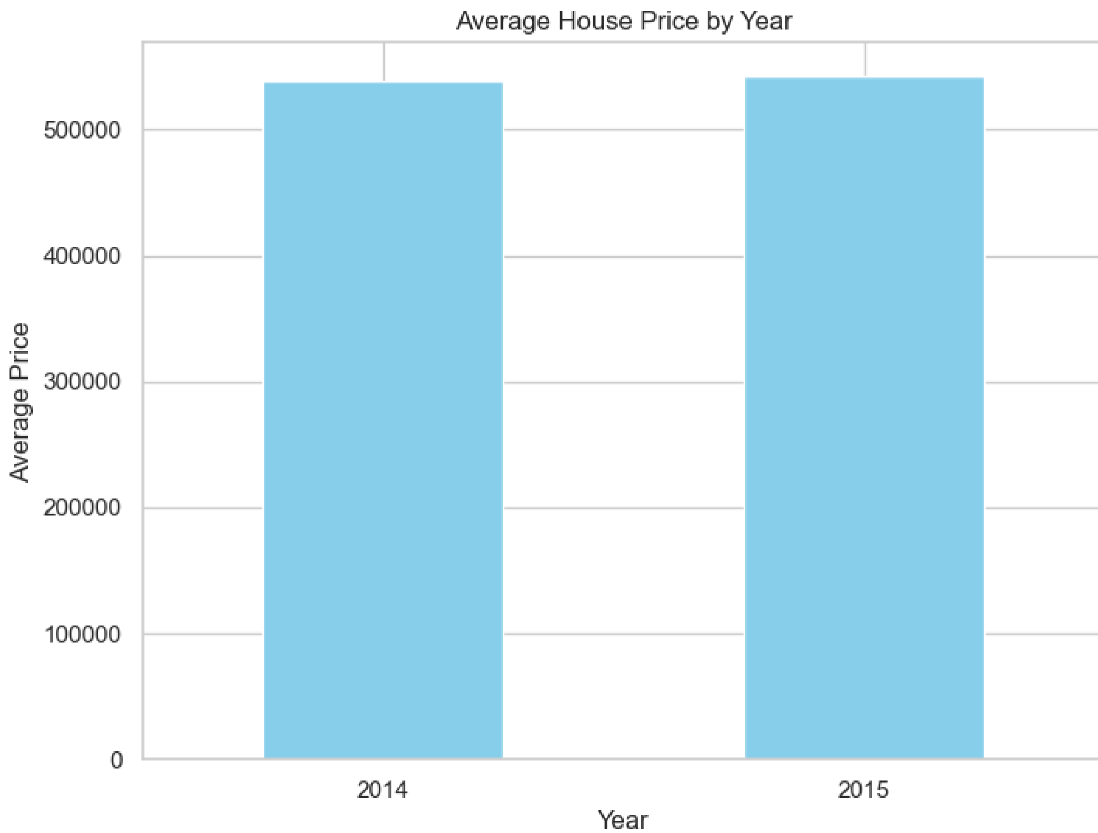
[47]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(x='long', y='lat', hue='price', data=df, palette='coolwarm',␣
    ↪alpha=0.6)
plt.title('House Prices by Location')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



7.1.6 5.6 Price Trend Over Time

[50]:
```
df['year'] = df['date'].dt.year
avg_price_by_year = df.groupby('year')['price'].mean()

plt.figure(figsize=(8,6))
avg_price_by_year.plot(kind='bar', color='skyblue')
plt.title('Average House Price by Year')
plt.ylabel('Average Price')
plt.xlabel('Year')
plt.xticks(rotation=0)
plt.show()
```



7.2 6. Outlier Treatment

[55]:
```
df = df[df['bedrooms'] < 10]
df = df[df['sqft_living'] < 10000]
```

7.3 7. Feature Engineering

7.3.1 Price per square foot

[59]:
```
df['price_per_sqft'] = df['price'] / df['sqft_living']
```

7.3.2 House age

[62]:
```
df['house_age'] = 2025 - df['yr_built']
```

7.3.3 7.1 Bathrooms vs Price

[65]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(x='bathrooms', y='price', data=df, alpha=0.6)
plt.title('Scatterplot of Bathrooms vs Price')
plt.xlabel('Number of Bathrooms')
```

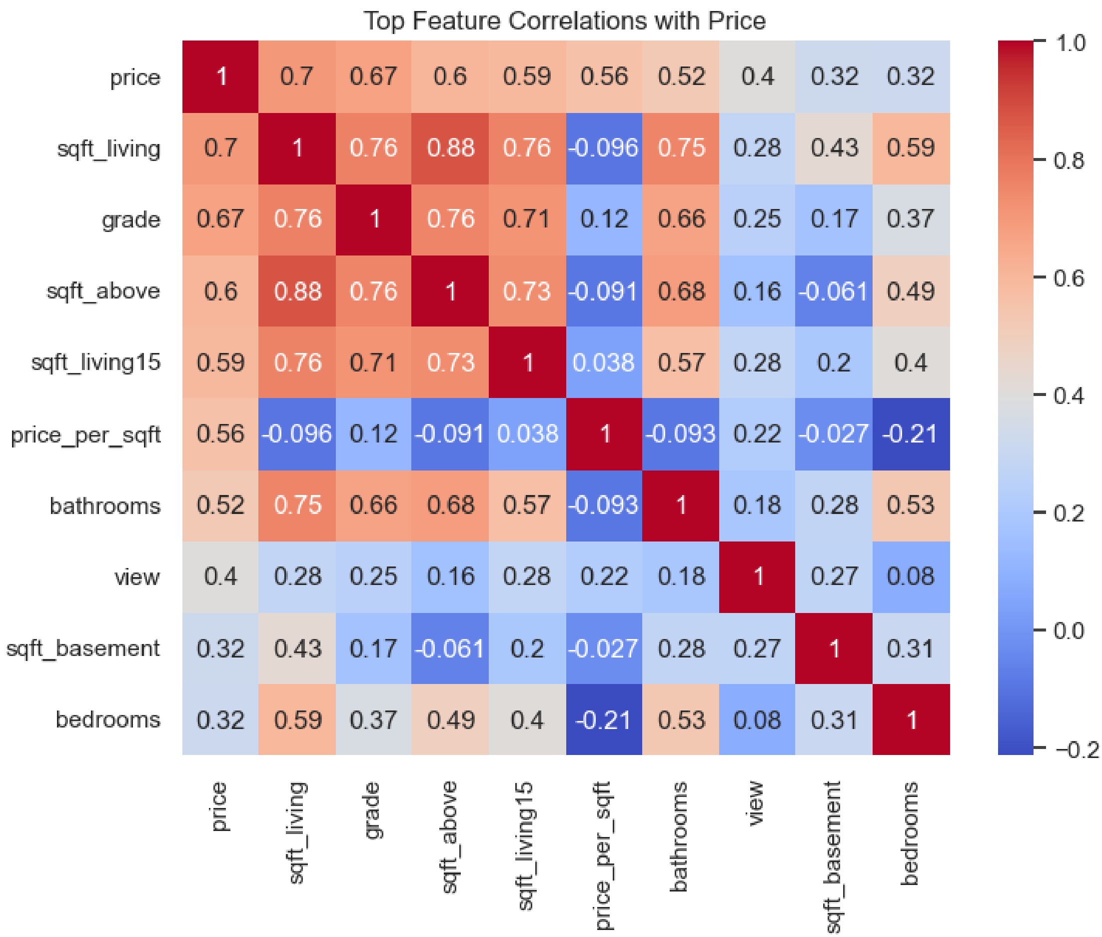
```
plt.ylabel('Price')
plt.show()
```

7.4 8. Correlation Heatmap

[68]:
```
corr = df.corr(numeric_only=True)


top_corr = corr['price'].abs().sort_values(ascending=False).head(10).index

plt.figure(figsize=(8,6))
sns.heatmap(df[top_corr].corr(), annot=True, cmap='coolwarm')
plt.title('Top Feature Correlations with Price')
plt.show()
```



7.5 9. Key Insights

- **Larger homes (sqft_living)** and **higher grade ratings** strongly correlate with higher prices.
- **Location matters** — Central and coastal areas have pricier homes.
- Most houses are under $1M, with a few luxurious outliers.
- Price per square foot and age are helpful derived metrics.
- Slight upward price trend from 2014 to 2015.