X23 Group 4

Project with Real Estate Dataset

Christian Rodriguez, Emmanuel Salcedo, Ranjita Summan, Crystian Chavez

In [10]:
```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
from scipy.stats import zscore
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score

# allow output to span multiple output lines in the console
pd.set_option('display.max_columns', 500)
```

Dataset Selection: We have chosen the USA Real Estate Dataset from Kaggle, which contains information about real estate properties in the USA.

https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset?resource=download

Google Drive is mounted to colab and placed into path /content/drive/MyDrive/realtor-data.csv.

In [11]:
```
#from google.colab import drive
#drive.mount('/content/drive')
```

Goal Definition: Goal is to build a system that predicts the cost of a home based on features such as footage, number of beds, and baths, on a per-state basis.

Feature Selection:In this case, the relevant features are "footage," "beds," and "baths." These features will be used to predict the cost of a home.

Preprocessing: Perform preprocessing steps on the dataset to prepare it for analysis. This may include handling missing values, encoding categorical variables, and scaling numerical features. Skipping this for now.

Exploration and Visualization: Here we are exploring the relationship between variables by using plots such as histograms, scatter plots, or box plots.

In [12]:

```
#df = pd.read_csv('/content/drive/MyDrive/realtor-data.csv')
#df.head(100)
df = pd.read_csv(r'C:\Users\Crystian\Documents\GitHub\CST383\realtor-data.csv')
df.head(20)
```

Out[12]:

	status	bed	bath	acre_lot	city	state	zip_code	house_size	prev_sold_date	price
0	for_sale	3.0	2.0	0.12	Adjuntas	Puerto Rico	601.0	920.0	NaN	105000.0
1	for_sale	4.0	2.0	0.08	Adjuntas	Puerto Rico	601.0	1527.0	NaN	80000.0
2	for_sale	2.0	1.0	0.15	Juana Diaz	Puerto Rico	795.0	748.0	NaN	67000.0
3	for_sale	4.0	2.0	0.10	Ponce	Puerto Rico	731.0	1800.0	NaN	145000.0
4	for_sale	6.0	2.0	0.05	Mayaguez	Puerto Rico	680.0	NaN	NaN	65000.0
5	for_sale	4.0	3.0	0.46	San Sebastian	Puerto Rico	612.0	2520.0	NaN	179000.0
6	for_sale	3.0	1.0	0.20	Ciales	Puerto Rico	639.0	2040.0	NaN	50000.0
7	for_sale	3.0	2.0	0.08	Ponce	Puerto Rico	731.0	1050.0	NaN	71600.0
8	for_sale	2.0	1.0	0.09	Ponce	Puerto Rico	730.0	1092.0	NaN	100000.0
9	for_sale	5.0	3.0	7.46	Las Marias	Puerto Rico	670.0	5403.0	NaN	300000.0
10	for_sale	3.0	2.0	13.39	Isabela	Puerto Rico	662.0	1106.0	NaN	89000.0
11	for_sale	3.0	2.0	0.08	Juana Diaz	Puerto Rico	795.0	1045.0	NaN	150000.0
12	for_sale	3.0	2.0	0.10	Lares	Puerto Rico	669.0	4161.0	NaN	155000.0
13	for_sale	5.0	2.0	0.12	Utuado	Puerto Rico	641.0	1620.0	NaN	79000.0
14	for_sale	5.0	5.0	0.74	Ponce	Puerto Rico	731.0	2677.0	NaN	649000.0
15	for_sale	3.0	2.0	0.08	Yauco	Puerto Rico	698.0	1100.0	NaN	120000.0
16	for_sale	4.0	4.0	0.22	Mayaguez	Puerto Rico	680.0	3450.0	NaN	235000.0
17	for_sale	3.0	2.0	0.08	Ponce	Puerto Rico	728.0	1500.0	NaN	105000.0
18	for_sale	3.0	2.0	3.88	San Sebastian	Puerto Rico	685.0	4000.0	NaN	575000.0
19	for_sale	6.0	3.0	0.25	Anasco	Puerto Rico	610.0	1230.0	NaN	140000.0

New Section

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 407890 entries, 0 to 407889
Data columns (total 10 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   status          407890 non-null  object 
 1   bed             320108 non-null  float64
 2   bath            321618 non-null  float64
 3   acre_lot        331873 non-null  float64
 4   city            407838 non-null  object 
 5   state           407890 non-null  object 
 6   zip_code        407693 non-null  float64
 7   house_size      324365 non-null  float64
 8   prev_sold_date  140950 non-null  object 
 9   price           407890 non-null  float64
dtypes: float64(6), object(4)
memory usage: 31.1+ MB
```

In [14]:
```
df.describe()
```

Out[14]:

	bed	bath	acre_lot	zip_code	house_size	price
count	320108.000000	321618.000000	331873.000000	407693.000000	3.243650e+05	4.078900e+05
mean	3.500200	2.566545	17.418487	3299.396838	2.222783e+03	6.758307e+05
std	2.320135	2.391618	931.723094	2222.641467	3.333098e+03	1.178266e+06
min	1.000000	1.000000	0.000000	601.000000	1.000000e+02	1.000000e+00
25%	2.000000	2.000000	0.200000	1890.000000	1.206000e+03	1.999000e+05
50%	3.000000	2.000000	0.560000	2822.000000	1.767000e+03	3.979000e+05
75%	4.000000	3.000000	2.200000	4630.000000	2.640000e+03	7.090000e+05
max	99.000000	198.000000	100000.000000	99999.000000	1.450112e+06	6.000000e+07

Running checks to see what columns have data missing and calculating the percentage in relation to total.

In [15]:
```
missing_total = df.isna().sum()
print("# of items missing from columns")
print(missing_total)
print(" ")
missing_total_per = df.isna().sum()*100/len(df)
print("Percentage of items missing from columns")
print(missing_total_per)
```

```
# of items missing from columns
status                0
bed               87782
bath              86272
acre_lot          76017
city                 52
state                 0
zip_code            197
house_size        83525
prev_sold_date   266940
price                 0
dtype: int64

Percentage of items missing from columns
status             0.000000
bed               21.520998
bath              21.150800
acre_lot          18.636642
city               0.012749
state              0.000000
zip_code           0.048297
house_size        20.477335
prev_sold_date    65.444115
price              0.000000
dtype: float64
```

We will drop rows that have items missing from bed and bath, and house size to trim data to have homes with a price, bed and bath numbers.

In [16]:
```
df = df.dropna(subset=['bed', 'bath', 'house_size', 'zip_code'])

# Resetting the indices using df.reset_index()
df = df.reset_index(drop=True)
```

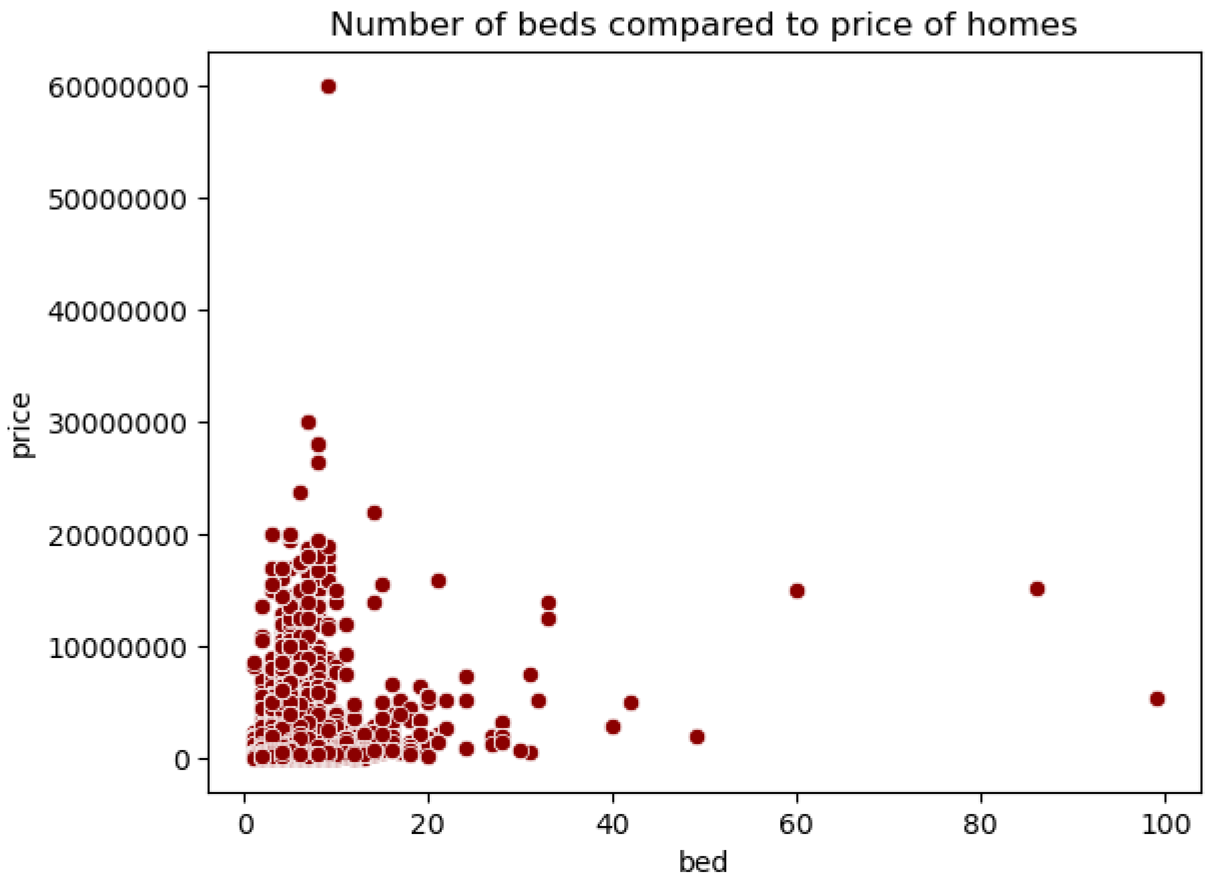
In [17]:
```
missing_total2 = df.isna().sum()
print("# of items missing from columns")
print(missing_total2)
```

```
# of items missing from columns
status                0
bed                   0
bath                  0
acre_lot          73083
city                  1
state                 0
zip_code              0
house_size            0
prev_sold_date   183947
price                 0
dtype: int64
```

Creating a simple scatter plot to view how the number of beds relates to the cost of a home. (Price of home in scientific notation)

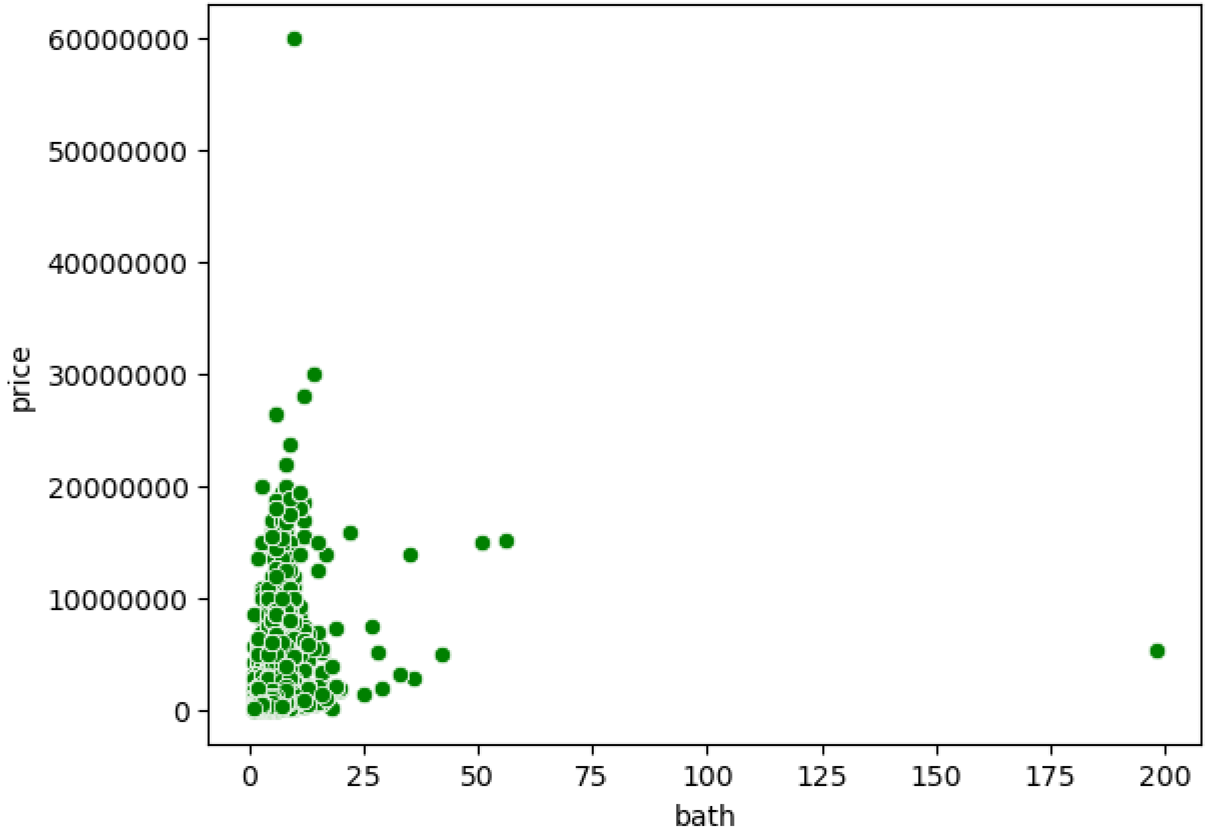
In [18]:
```
sns.scatterplot(data=df, x='bed', y='price', color='darkred')
plt.ticklabel_format(style='plain')
plt.title("Number of beds compared to price of homes")
```

Out[18]: Text(0.5, 1.0, 'Number of beds compared to price of homes')



Another scatterplot of number of baths related to price of home. (scientific notation)

In [19]:
```
sns.scatterplot(data=df, x='bath', y='price', color='green')
plt.ticklabel_format(style='plain')
```



Looking at the scatter plots we can see that the numbers are skewed quite a bit on the number of beds and bath, we will continue to clean up the dataset and remove homes that have more than 8 bedrooms and 10 bath so that we can get a better representation of the average home on the market.

In [20]:

```
indexBed = df[(df['bed'] > 10)].index
df.drop(indexBed , inplace=True)

indexBath = df[(df['bath'] > 8)].index
df.drop(indexBath , inplace=True)

indexPrice = df[(df['price'] > 999999)].index
df.drop(indexPrice , inplace=True)


df.head(15)
```

Out[20]:

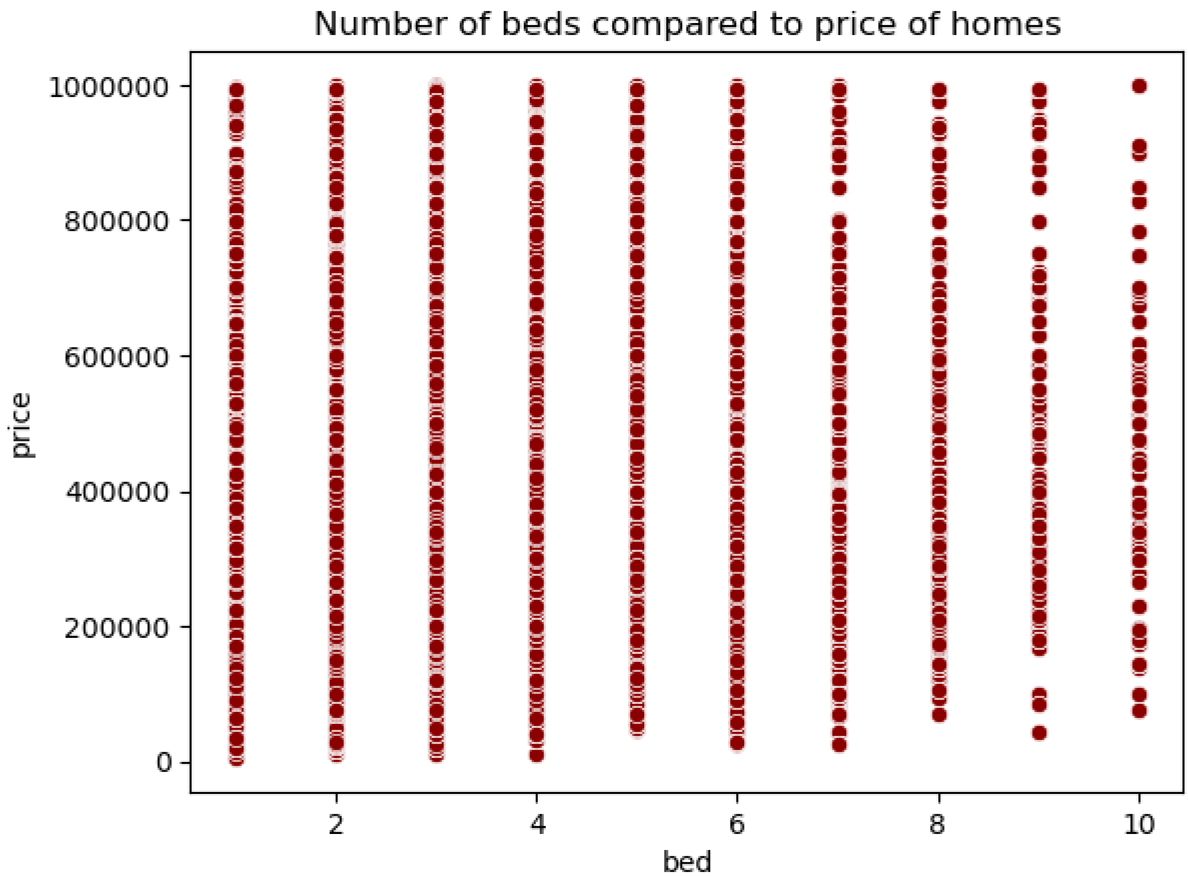
	status	bed	bath	acre_lot	city	state	zip_code	house_size	prev_sold_date	price
0	for_sale	3.0	2.0	0.12	Adjuntas	Puerto Rico	601.0	920.0	NaN	105000.0
1	for_sale	4.0	2.0	0.08	Adjuntas	Puerto Rico	601.0	1527.0	NaN	80000.0
2	for_sale	2.0	1.0	0.15	Juana Diaz	Puerto Rico	795.0	748.0	NaN	67000.0
3	for_sale	4.0	2.0	0.10	Ponce	Puerto Rico	731.0	1800.0	NaN	145000.0
4	for_sale	4.0	3.0	0.46	San Sebastian	Puerto Rico	612.0	2520.0	NaN	179000.0
5	for_sale	3.0	1.0	0.20	Ciales	Puerto Rico	639.0	2040.0	NaN	50000.0
6	for_sale	3.0	2.0	0.08	Ponce	Puerto Rico	731.0	1050.0	NaN	71600.0
7	for_sale	2.0	1.0	0.09	Ponce	Puerto Rico	730.0	1092.0	NaN	100000.0
8	for_sale	5.0	3.0	7.46	Las Marias	Puerto Rico	670.0	5403.0	NaN	300000.0
9	for_sale	3.0	2.0	13.39	Isabela	Puerto Rico	662.0	1106.0	NaN	89000.0
10	for_sale	3.0	2.0	0.08	Juana Diaz	Puerto Rico	795.0	1045.0	NaN	150000.0
11	for_sale	3.0	2.0	0.10	Lares	Puerto Rico	669.0	4161.0	NaN	155000.0
12	for_sale	5.0	2.0	0.12	Utuado	Puerto Rico	641.0	1620.0	NaN	79000.0
13	for_sale	5.0	5.0	0.74	Ponce	Puerto Rico	731.0	2677.0	NaN	649000.0
14	for_sale	3.0	2.0	0.08	Yauco	Puerto Rico	698.0	1100.0	NaN	120000.0

In [21]:

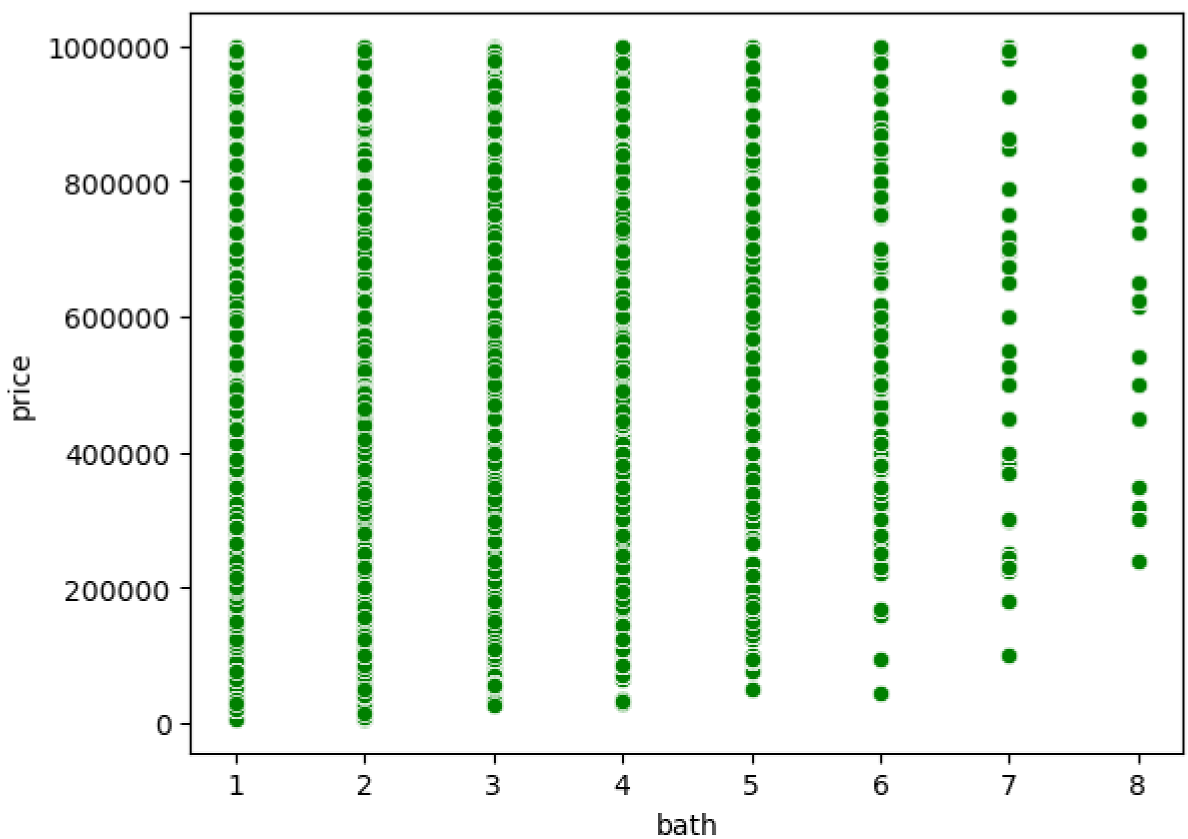
```
sns.scatterplot(data=df, x='bed', y='price', color='darkred')
plt.ticklabel_format(style='plain')
plt.title("Number of beds compared to price of homes")
```

Out[21]:

```
Text(0.5, 1.0, 'Number of beds compared to price of homes')
```



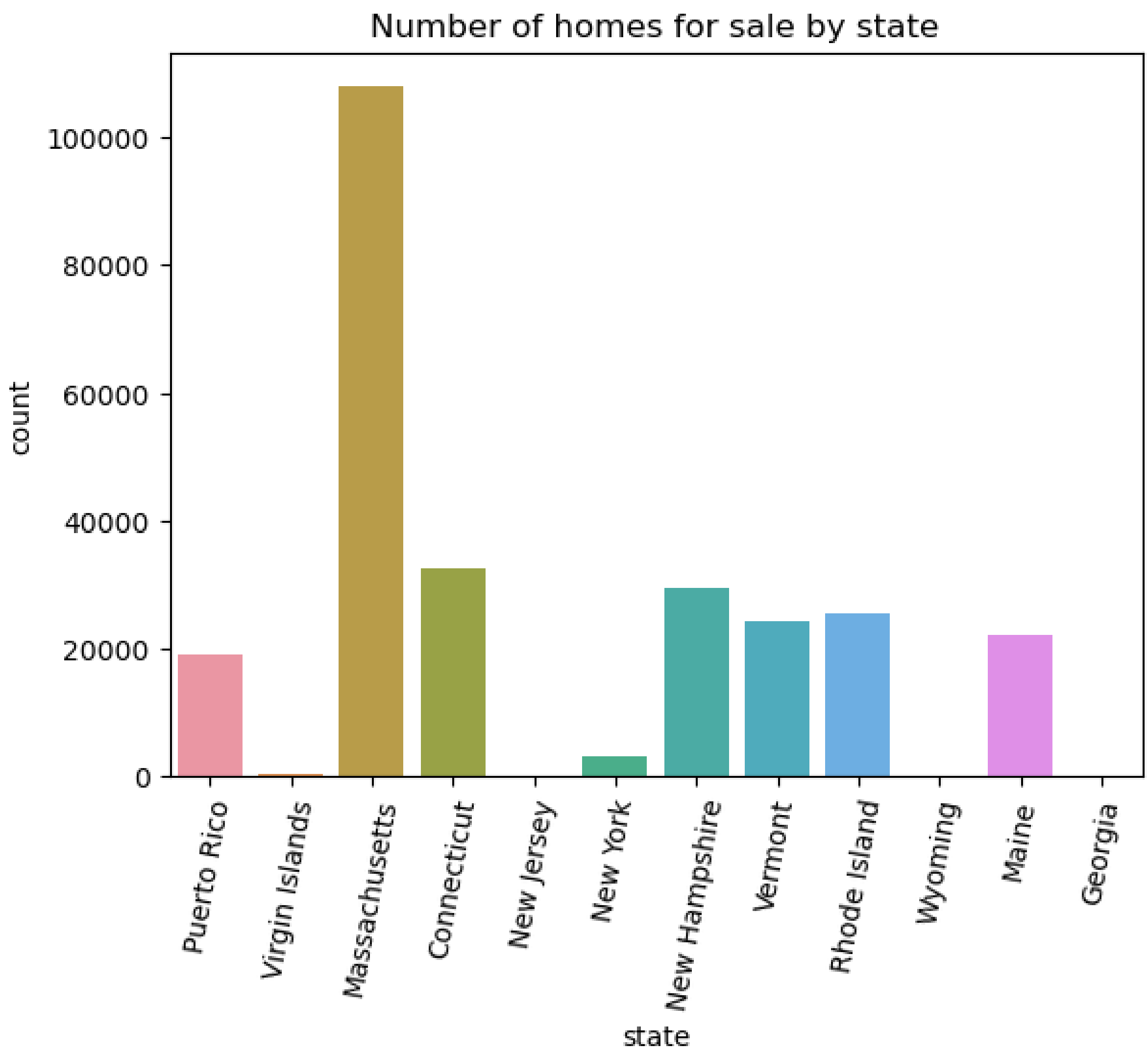
In [22]:
```
sns.scatterplot(data=df, x='bath', y='price', color='green')
plt.ticklabel_format(style='plain')
```



Using seaborn count plot to see the number of homes for sale per state. As we found, some states do not have homes for sale but just land only.

In [23]:
```
sns.countplot(data=df, x='state')
plt.xticks(rotation=80)
plt.title("Number of homes for sale by state")
```

Out[23]:
```
Text(0.5, 1.0, 'Number of homes for sale by state')
```

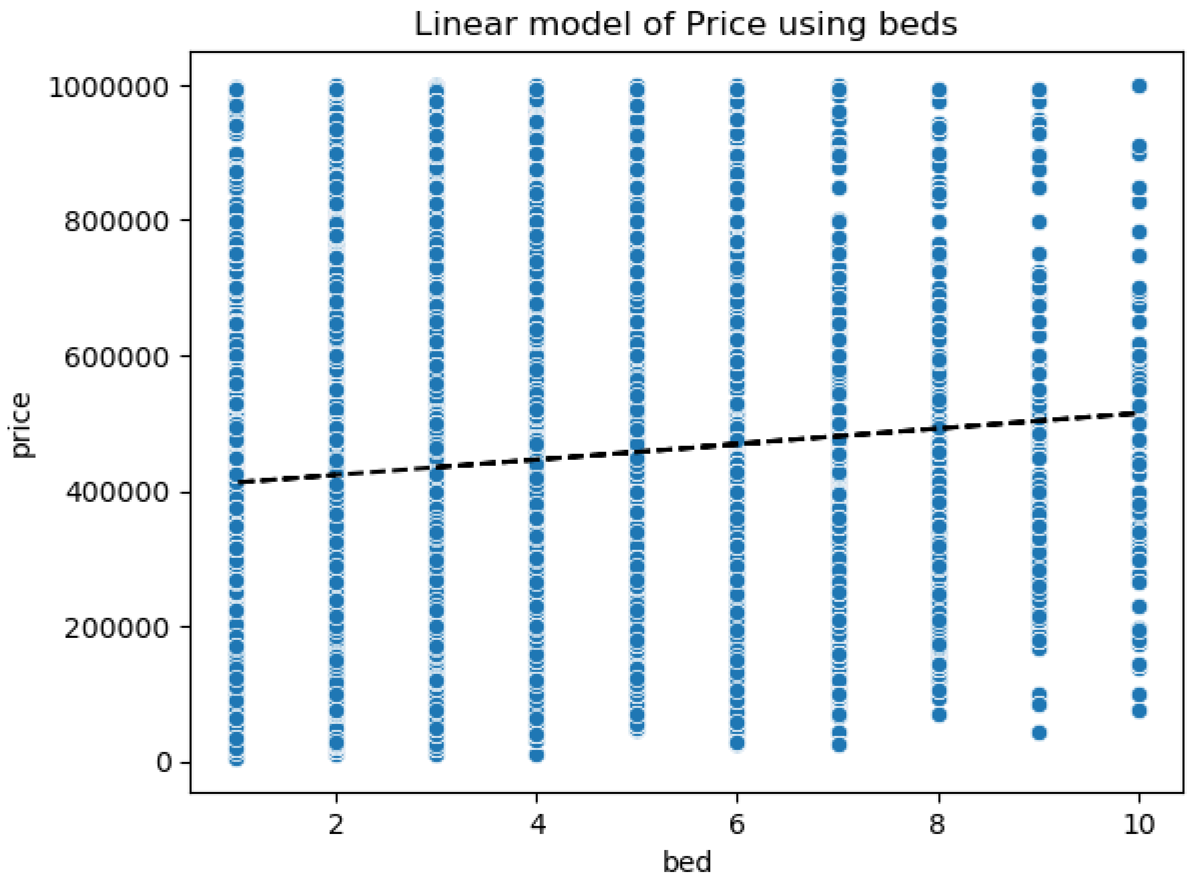


In [24]:
```
X = df[['bed']].values
y = df[['price']].values

reg = LinearRegression()
reg.fit(X,y)

sns.scatterplot(data=df, x='bed', y='price')
plt.title("Linear model of Price using beds")

plt.plot(X, reg.intercept_ + reg.coef_*X, linestyle='dashed', color='black')
plt.ticklabel_format(style='plain')
```



In [25]:
```
print(f'intercept: {reg.intercept_[0]:.2f}')
print(f'coefficient for price: {reg.coef_[0]}')
print(f'r-sqaured value: {reg.score(X,y):.2f}')
```

```
intercept: 401245.26
coefficient for price: [11307.83415314]
r-sqaured value: 0.00
```

In [26]:
```
predictors = ['bed', 'bath']

X = df[predictors]
y = df['price']

reg2 = LinearRegression()
reg2.fit(X,y)

print(f'intercept: {reg2.intercept_:.2f}')
print('coefficients:')
for i, coef in enumerate(reg2.coef_):
    print(f' {predictors[i]}: {coef:.2f}')
```

```
intercept: 290364.02
coefficients:
 bed: -31057.80
 bath: 112282.24
```

In [27]:
```
matrix = pd.DataFrame({'bed': [3], 'bath': [3]})

results = reg2.predict(matrix)[0]
print(f'{results:.2f}')
```

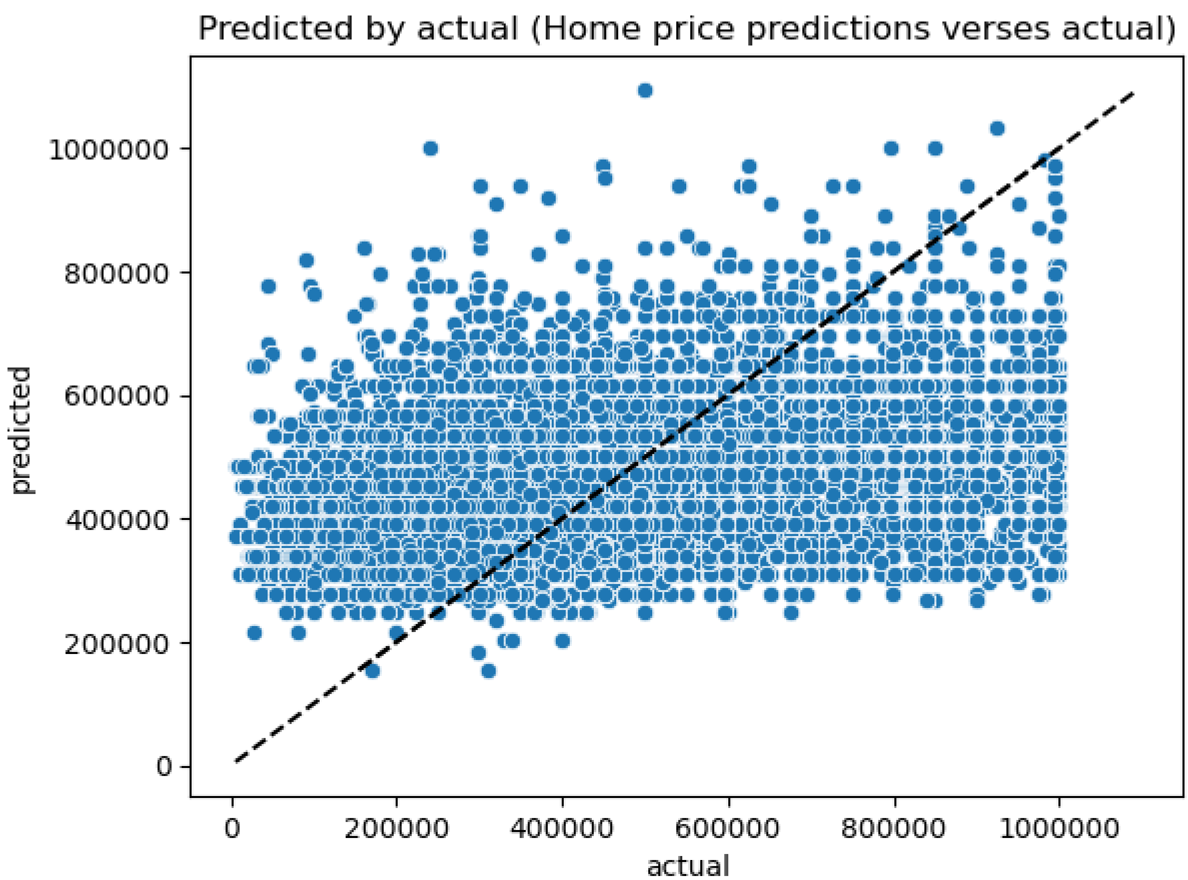
```
534037.32
```

In [28]:
```python
def plot_actual_predicted(actual, predicted, title):
    sns.scatterplot(x=actual, y=predicted)
    a = (np.min(actual),np.min(predicted))
    b = (np.max(actual),np.max(predicted))

    plt.plot((a,b), (a,b), linestyle='dashed', color='black')

    plt.title(title)
    plt.xlabel('actual')
    plt.ylabel('predicted')
    plt.ticklabel_format(style='plain')
```

In [29]:
```python
plot_actual_predicted(y, reg2.predict(X), 'Predicted by actual (Home price predictions
```



In [30]:
```python
predictors = ['bed', 'bath', 'zip_code']

X = df[predictors]
y = df['price']

reg3 = LinearRegression()
reg3.fit(X,y)

print(f'intercept: {reg3.intercept_:.2f}')
print('coefficients:')
for i, coef in enumerate(reg3.coef_):
    print(f' {predictors[i]}: {coef:.2f}')
```

```
intercept: 347289.66
coefficients:
 bed: -30941.97
 bath: 114300.17
 zip_code: -18.78
```