Where oh where are the home buyers?

Business Understanding

Noznas Inc has noticed a rapid decline in their house sales lately and would like to know where to focus their marketing efforts. With so much competition out there, they want to be on top of who the home buyers are and what they are looking for. They want to essentially speak right to them through their advertising making the potential buyers feel as if Noznas knows exactly what they need. Their sales in King County, Washington did well and they supplied the data from there as a reference.

Data Understanding

The data used for this project was obtained from a the kc_house_data.csv file which contains data for 2021-2022 home sales in King County, Washington.

Data Preparation

Loading the Data

First I loaded all the libraries I felt I needed.

In [1]:
```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns
import statsmodels.api as sm
```

Next was the file.

In [2]:
```
dfkc = pd.read_csv('data/kc_house_data.csv')
```

Data Exploration

I looked through the file to familiarize myself with the information provided and to see if I can find what what buyers are looking for in a house.

In [3]:

```
dfkc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30155 entries, 0 to 30154
Data columns (total 25 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             30155 non-null  int64  
 1   date           30155 non-null  object 
 2   price          30155 non-null  float64
 3   bedrooms       30155 non-null  int64  
 4   bathrooms      30155 non-null  float64
 5   sqft_living    30155 non-null  int64  
 6   sqft_lot       30155 non-null  int64  
 7   floors         30155 non-null  float64
 8   waterfront     30155 non-null  object 
 9   greenbelt      30155 non-null  object 
 10  nuisance       30155 non-null  object 
 11  view           30155 non-null  object 
 12  condition      30155 non-null  object 
 13  grade          30155 non-null  object 
 14  heat_source    30123 non-null  object 
 15  sewer_system   30141 non-null  object 
 16  sqft_above     30155 non-null  int64  
 17  sqft_basement  30155 non-null  int64  
 18  sqft_garage    30155 non-null  int64  
 19  sqft_patio     30155 non-null  int64  
 20  yr_built       30155 non-null  int64  
 21  yr_renovated   30155 non-null  int64  
 22  address        30155 non-null  object 
 23  lat            30155 non-null  float64
 24  long           30155 non-null  float64
dtypes: float64(5), int64(10), object(10)
memory usage: 5.8+ MB
```

In [4]:
```
dfkc.head()
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
0	7399300360	5/24/2022	675000.0	4	1.0	1180	7140	1.0	
1	8910500230	12/13/2021	920000.0	5	2.5	2770	6703	1.0	
2	1180000275	9/29/2021	311000.0	6	2.0	2880	6156	1.0	
3	1604601802	12/14/2021	775000.0	3	3.0	2160	1400	2.0	
4	8562780790	8/24/2021	592500.0	2	2.0	1120	758	2.0	

5 rows × 25 columns

In [5]:
```
dfkc.columns
```

Out[5]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'greenbelt', 'nuisance', 'view',
       'condition', 'grade', 'heat_source', 'sewer_system', 'sqft_above',
       'sqft_basement', 'sqft_garage', 'sqft_patio', 'yr_built',
       'yr_renovated', 'address', 'lat', 'long'],
      dtype='object')
```

In [6]:
```
dfkc.describe()
```

Out[6]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	3.015500e+04	3.015500e+04	30155.000000	30155.000000	30155.000000	3.015500e+04
mean	4.538104e+09	1.108536e+06	3.413530	2.334737	2112.424739	1.672360e+04
std	2.882587e+09	8.963857e+05	0.981612	0.889556	974.044318	6.038260e+04
min	1.000055e+06	2.736000e+04	0.000000	0.000000	3.000000	4.020000e+02
25%	2.064175e+09	6.480000e+05	3.000000	2.000000	1420.000000	4.850000e+03
50%	3.874011e+09	8.600000e+05	3.000000	2.500000	1920.000000	7.480000e+03
75%	7.287100e+09	1.300000e+06	4.000000	3.000000	2619.500000	1.057900e+04
max	9.904000e+09	3.075000e+07	13.000000	10.500000	15360.000000	3.253932e+06

Data Cleaning

First and foremost, I made a copy of the file just so any changes I make will not affect the actual file.

In [7]:
```
dfkc_copy=dfkc
```

Checking to make sure the copy worked.

In [8]:
```
dfkc_copy.head()
```

2	1180000275	9/29/2021	311000.0	6	2.0	2880	6156	1.0
3	1604601802	12/14/2021	775000.0	3	3.0	2160	1400	2.0
4	8562780790	8/24/2021	592500.0	2	2.0	1120	758	2.0

In [9]:
```
dfkc_copy.columns
```

Out[9]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'greenbelt', 'nuisance', 'view',
       'condition', 'grade', 'heat_source', 'sewer_system', 'sqft_above',
       'sqft_basement', 'sqft_garage', 'sqft_patio', 'yr_built',
       'yr_renovated', 'address', 'lat', 'long'],
      dtype='object')
```

Took out the columns I didn't feel were necessary.

In [10]:
```
dfkc_copy1=dfkc_copy.drop(labels='nuisance',axis=1)
```

In [11]:
```
dfkc_copy2=dfkc_copy1.drop(labels='id',axis=1)
```

In [12]:
```
dfkc_copy3=dfkc_copy2.drop(labels='floors',axis=1)
```

In [13]:
```
dfkc_copy4=dfkc_copy3.drop(labels='sqft_basement',axis=1)
```

In [14]:
```
dfkc_copy5=dfkc_copy4.drop(labels='sqft_above',axis=1)
```

In [15]:
```
dfkc_copy5.columns
```

Out[15]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'waterfront', 'greenbelt', 'view', 'condition', 'grade', 'heat_sourc
e',
       'sewer_system', 'sqft_garage', 'sqft_patio', 'yr_built', 'yr_renovat
ed',
       'address', 'lat', 'long'],
      dtype='object')
```

Dropped the rows with null values. There was just a small number of them and I felt dropping them completely wouldn't affect the overall numbers much.

In [16]:
```
dfkc_copy5.dropna(subset=['heat_source','sewer_system'], axis=0, inplace=True
dfkc_copy5.head()
```

Out[16]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	waterfront	greenbelt
0	5/24/2022	675000.0	4	1.0	1180	7140	NO	NO
1	12/13/2021	920000.0	5	2.5	2770	6703	NO	NO
2	9/29/2021	311000.0	6	2.0	2880	6156	NO	NO

In [17]:
```
dfkc_copy5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30111 entries, 0 to 30154
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           30111 non-null  object 
 1   price          30111 non-null  float64
 2   bedrooms       30111 non-null  int64  
 3   bathrooms      30111 non-null  float64
 4   sqft_living    30111 non-null  int64  
 5   sqft_lot       30111 non-null  int64  
 6   waterfront     30111 non-null  object 
 7   greenbelt      30111 non-null  object 
 8   view           30111 non-null  object 
 9   condition      30111 non-null  object 
 10  grade          30111 non-null  object 
 11  heat_source    30111 non-null  object 
 12  sewer_system   30111 non-null  object 
 13  sqft_garage    30111 non-null  int64  
 14  sqft_patio     30111 non-null  int64  
 15  yr_built       30111 non-null  int64  
 16  yr_renovated   30111 non-null  int64  
 17  address        30111 non-null  object 
 18  lat            30111 non-null  float64
 19  long           30111 non-null  float64
dtypes: float64(4), int64(7), object(9)
memory usage: 4.8+ MB
```

Remamed the date column to date_sold. I originally was going to leave it but for some reason couldn't grasp the concept of date. :)

In [18]:
```
rnmdate = {'date':'date_sold'}
dfkc_copy5.rename(columns=rnmdate,inplace=True)
dfkc_copy5.head(5)
```

Out[18]:

	date_sold	price	bedrooms	bathrooms	sqft_living	sqft_lot	waterfront	greenbelt	
0	5/24/2022	675000.0	4	1.0	1180	7140	NO	NO	
1	12/13/2021	920000.0	5	2.5	2770	6703	NO	NO	AV
2	9/29/2021	311000.0	6	2.0	2880	6156	NO	NO	AV
3	12/14/2021	775000.0	3	3.0	2160	1400	NO	NO	AV
4	8/24/2021	592500.0	2	2.0	1120	758	NO	NO	

Changed date_sold from a string to date/time to calculate age of homes, then added age column to the table.

In [19]:
```
#change date from str to date/time
dfkc_copy5.date_sold = dfkc_copy5.date_sold.apply(lambda x: pd.to_datetime(x,
```

In [20]:
```
#add age column
dfkc_copy5['age'] = np.where(dfkc_copy5['yr_renovated'] != 0, dfkc_copy5.date
                          dfkc_copy5['date_sold'].apply(lambda x: x.year) - d
```

In [21]:
```
#check to see if column was added
dfkc_copy5.head()
```

Out[21]:

	date_sold	price	bedrooms	bathrooms	sqft_living	sqft_lot	waterfront	greenbelt	
0	2022-05-24	675000.0	4	1.0	1180	7140	NO	NO	
1	2021-12-13	920000.0	5	2.5	2770	6703	NO	NO	AVE
2	2021-09-29	311000.0	6	2.0	2880	6156	NO	NO	AVE
3	2021-12-14	775000.0	3	3.0	2160	1400	NO	NO	AVE
4	2021-08-24	592500.0	2	2.0	1120	758	NO	NO	

5 rows × 21 columns

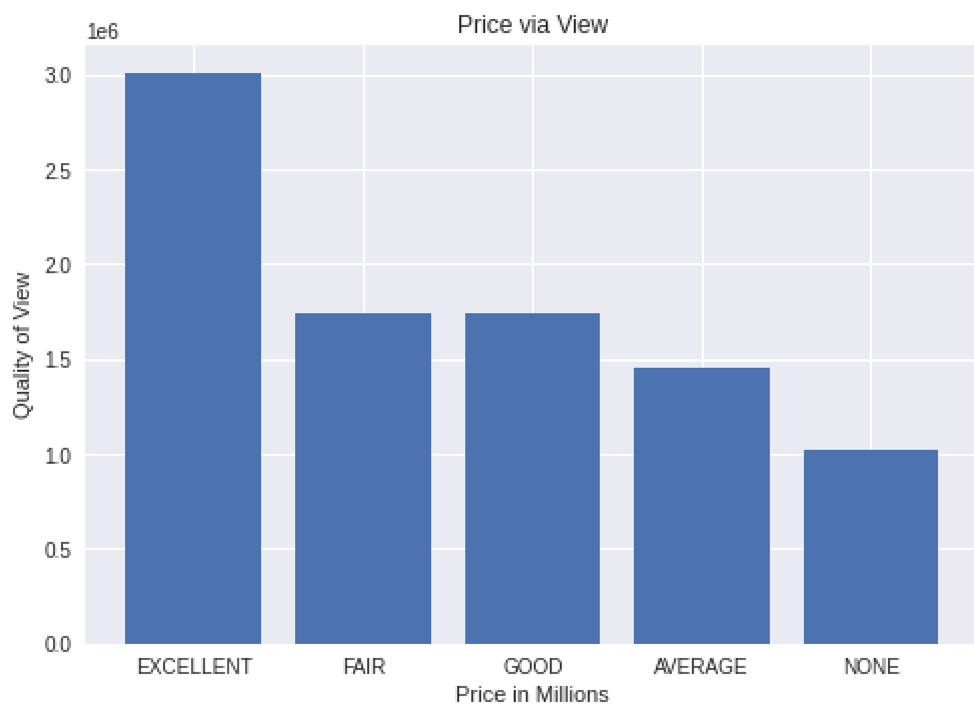
I researched the house features home buyers are looking for. While all the features on this table are considered, I found the most important and calculated the average selling price for each feature to find the best condition and made a visualization for each.

In [22]:
```
avg_view=dfkc_copy5.groupby('view').mean()['price'].reset_index().sort_values
avg_view
```

Out[22]:

	view	price
1	EXCELLENT	3.007975e+06
2	FAIR	1.742069e+06
3	GOOD	1.738145e+06
0	AVERAGE	1.454727e+06
4	NONE	1.018818e+06

In [23]:
```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.bar(avg_view['view'],avg_view['price'])
plt.title('Price via View')
plt.ylabel('Quality of View')
plt.xlabel('Price in Millions')
plt.show()
```

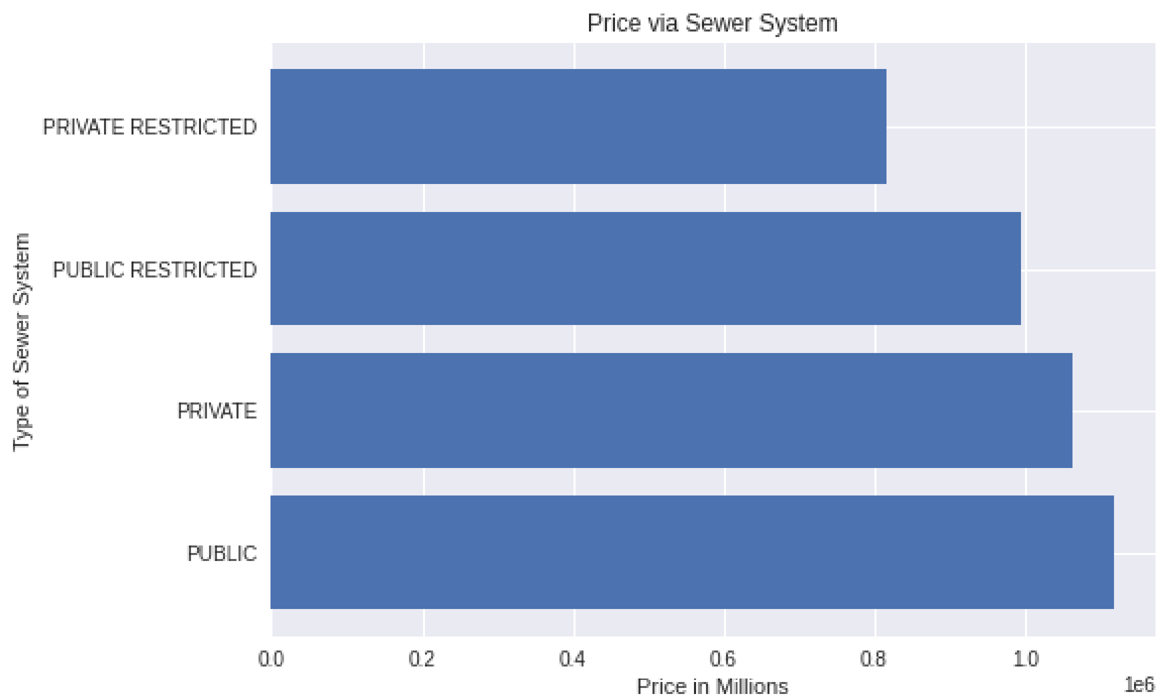


In [24]:
```
avg_ss=dfkc_copy5.groupby('sewer_system').mean()['price'].reset_index().sort_
avg_ss
```

Out[24]:

	sewer_system	price
2	PUBLIC	1.116769e+06
0	PRIVATE	1.063043e+06
3	PUBLIC RESTRICTED	9.951490e+05
1	PRIVATE RESTRICTED	8.164000e+05

In [25]:
```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.barh(avg_ss['sewer_system'],avg_ss['price'])
plt.title('Price via Sewer System')
plt.ylabel('Type of Sewer System')
plt.xlabel('Price in Millions')
plt.show()
```



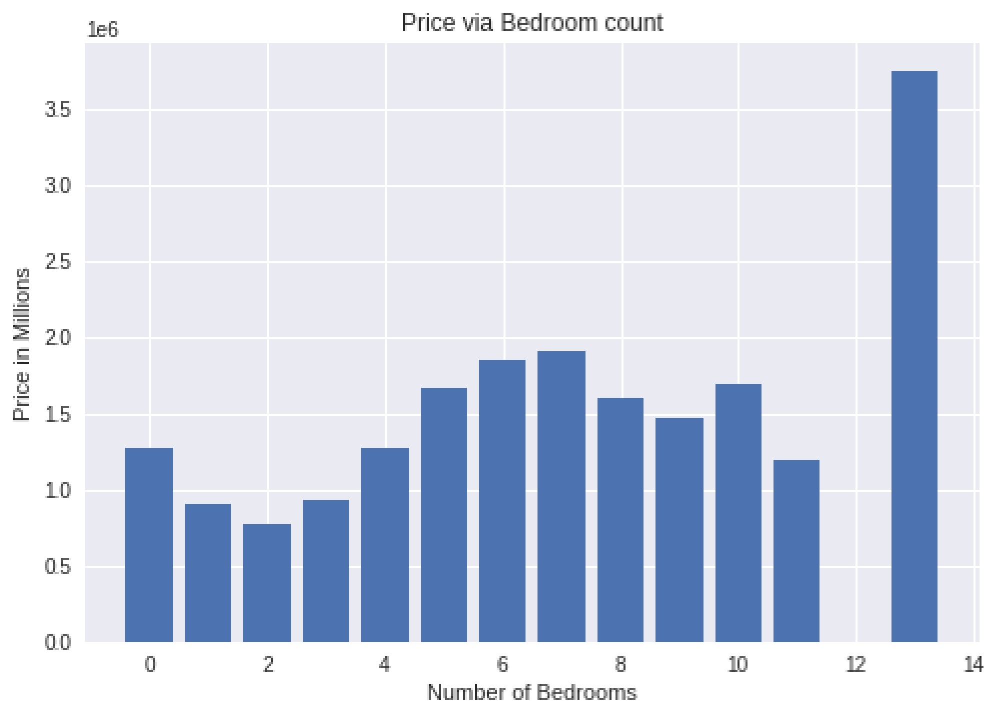
In [26]:
```
avg_bdrm=dfkc_copy5.groupby('bedrooms').mean()['price'].reset_index().sort_va
avg_bdrm
```

Out[26]:

	bedrooms	price
12	13	3.750000e+06
7	7	1.903534e+06
6	6	1.848920e+06
10	10	1.700000e+06
5	5	1.673307e+06
8	8	1.605601e+06
9	9	1.474579e+06
4	4	1.269938e+06
0	0	1.268990e+06
11	11	1.200000e+06
3	3	9.345138e+05
1	1	9.110776e+05
2	2	7.807247e+05

In [27]:

```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.bar(avg_bdrm['bedrooms'],avg_bdrm['price'])
plt.title('Price via Bedroom count')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Price in Millions')
plt.show()
```



In [28]:

```
Bedrooms=avg_bdrm.mean().reset_index()
Bedrooms
```

Out[28]:

	index	0
0	bedrooms	6.076923e+00
1	price	1.563168e+06

In [29]:
```
avg_lot=dfkc_copy5.groupby('sqft_lot').mean()['price'].reset_index().sort_val
avg_lot
```

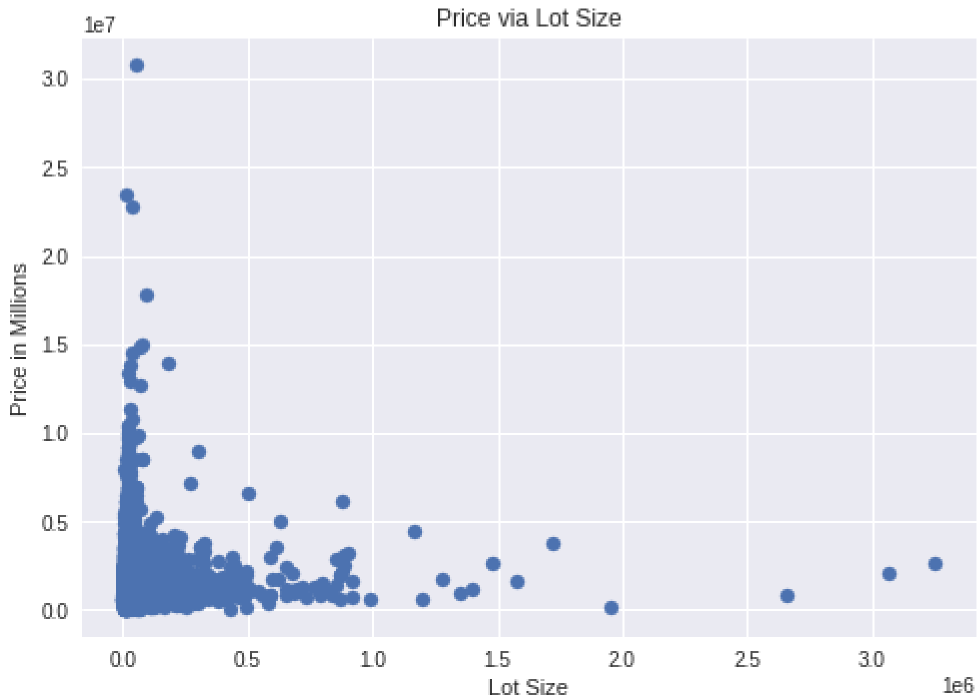
Out[29]:

	sqft_lot	price
11265	50705	30750000.0
8865	15494	23500000.0
10503	32920	22750000.0
11615	92345	17800000.0
11522	77594	15000001.0
...	...	...
11295	52101	40000.0
12140	429071	30108.0
11387	60229	29941.0
1992	3755	28559.0
11354	56809	28307.0

12223 rows × 2 columns

In [30]:

```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(avg_lot['sqft_lot'],avg_lot['price'])
plt.title('Price via Lot Size')
plt.xlabel('Lot Size')
plt.ylabel('Price in Millions')
plt.show()
```



In [31]:

```
Lot=avg_lot.mean().reset_index()
Lot
```

Out[31]:

	index	0
0	sqft_lot	2.634945e+04
1	price	1.214927e+06

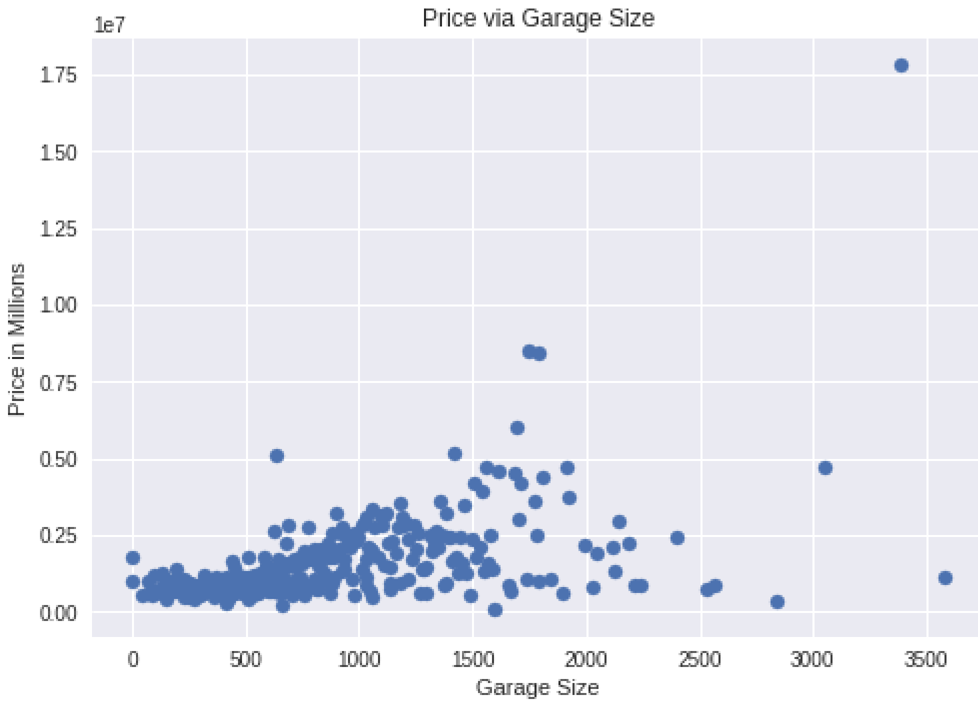
In [32]:
```
avg_garage=dfkc_copy5.groupby('sqft_garage').mean()['price'].reset_index().sc
avg_garage
```

Out[32]:

	sqft_garage	price
409	3390	17800000.0
385	1750	8500000.0
389	1790	8400000.0
381	1690	6000000.0
358	1420	5160000.0
...	...	...
147	511	390000.0
407	2840	328000.0
89	416	304625.0
221	664	204500.0
376	1600	67500.0

411 rows × 2 columns

In [33]:
```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(avg_garage['sqft_garage'],avg_garage['price'])
plt.title('Price via Garage Size')
plt.xlabel('Garage Size')
plt.ylabel('Price in Millions')
plt.show()
```



In [34]:
```
Garage=avg_garage.mean().reset_index()
Garage
```

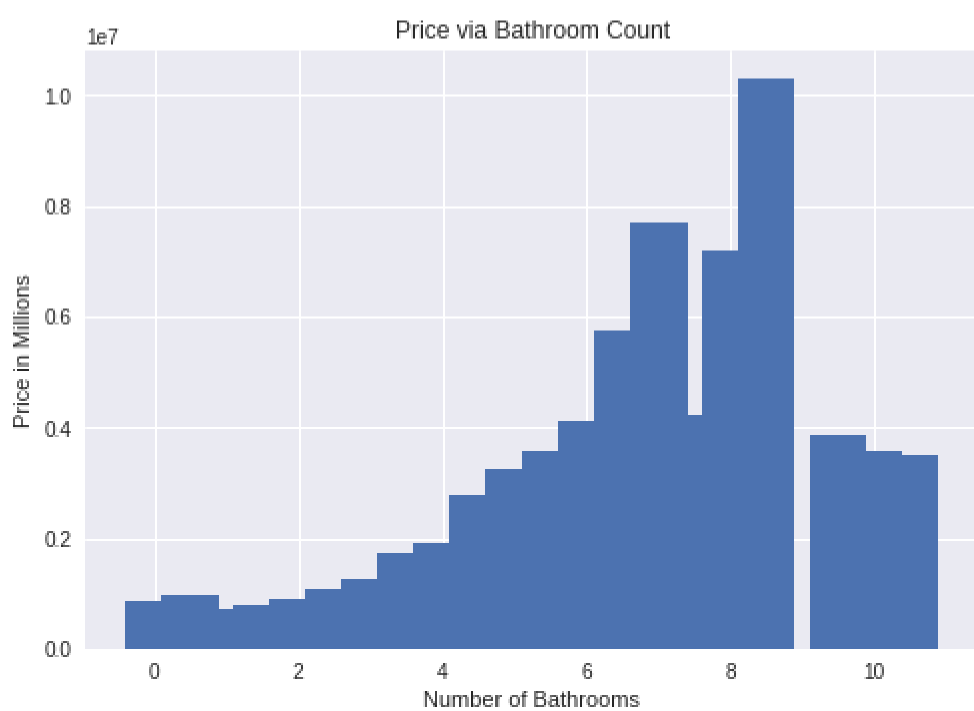
Out[34]:

	index	0
0	sqft_garage	7.949051e+02
1	price	1.425485e+06

In [35]:
```
avg_bathrooms=dfkc_copy5.groupby('bathrooms').mean()['price'].reset_index().s
avg_bathrooms
```

11	5.5	3.571516e+06
20	10.5	3.495000e+06
10	5.0	3.230708e+06
9	4.5	2.764199e+06
8	4.0	1.903924e+06
7	3.5	1.724917e+06
6	3.0	1.244328e+06
5	2.5	1.072449e+06
1	0.5	9.773900e+05
4	2.0	8.936014e+05
0	0.0	8.541956e+05
3	1.5	7.835607e+05
2	1.0	7.105680e+05

In [36]:
```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.bar(avg_bathrooms['bathrooms'],avg_bathrooms['price'])
plt.title('Price via Bathroom Count')
plt.xlabel('Number of Bathrooms')
plt.ylabel('Price in Millions')
plt.show()
```



In [37]:

```
Bathrooms=avg_bathrooms.mean().reset_index()
Bathrooms
```

Out[37]:

	index	0
0	bathrooms	5.071429e+00
1	price	3.330347e+06

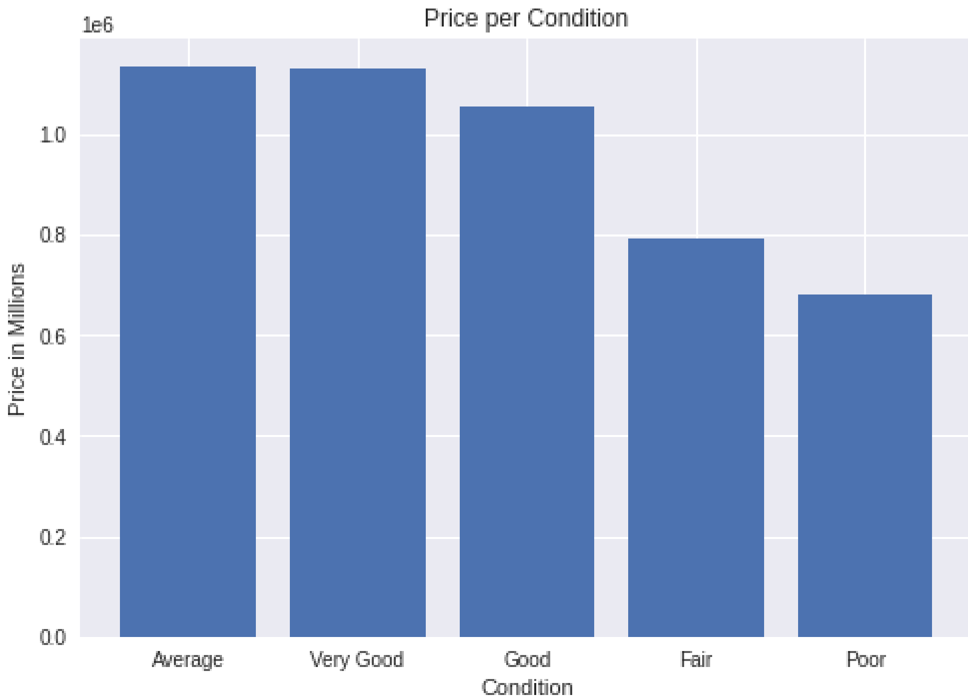
In [38]:

```
#keep
avg_condition=dfkc_copy5.groupby('condition').mean()['price'].reset_index().s
avg_condition
```

Out[38]:

	condition	price
0	Average	1.134565e+06
4	Very Good	1.130900e+06
2	Good	1.053324e+06
1	Fair	7.924678e+05
3	Poor	6.821867e+05

In [39]:
```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.bar(avg_condition['condition'],avg_condition['price'])
plt.title('Price per Condition')
plt.xlabel('Condition')
plt.ylabel('Price in Millions')
plt.show()
```



In [40]:
```
avg_living_sqft=dfkc_copy5.groupby('sqft_living').mean()['price'].reset_index
avg_living_sqft
```

Out[40]:

	sqft_living	price
1293	8160	22750000.0
1283	7610	20000000.0
1310	12470	17800000.0
1255	6780	15000001.0
1305	10250	14500000.0
...	...	...
43	684	188622.0
227	1532	153124.0
485	2244	119250.0
321	1806	64559.0
1187	5780	40000.0

1314 rows × 2 columns

In [41]:

```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(avg_living_sqft['sqft_living'],avg_living_sqft['price'])
plt.title('Price and Living Space distribution')
plt.xlabel('Living Space')
plt.ylabel('Price in Millions')
plt.show()
```

In [42]:

```
Living_Sqft=avg_living_sqft.mean().reset_index()
Living_Sqft
```

Out[42]:

	index	0
0	sqft_living	3.109803e+03
1	price	1.718650e+06

In [43]:

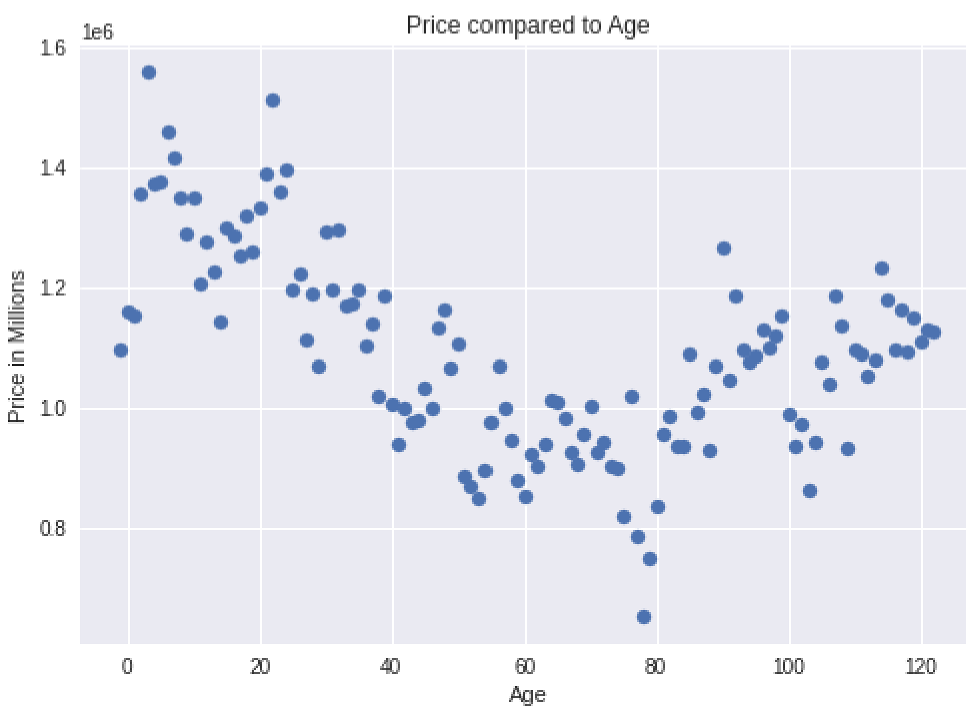
```
avg_age=dfkc_copy5.groupby('age').mean()['price'].reset_index().sort_values("
avg_age
```

Out[43]:

	age	price
4	3	1.557398e+06
23	22	1.512969e+06
7	6	1.459599e+06
8	7	1.416208e+06
25	24	1.396900e+06
...	...	...
81	80	8.362164e+05
76	75	8.180666e+05
78	77	7.852669e+05
80	79	7.483135e+05
79	78	6.510239e+05

In [44]:

```
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(avg_age['age'],avg_age['price'])
plt.title('Price compared to Age')
plt.xlabel('Age')
plt.ylabel('Price in Millions')
plt.show()
```



In [45]:

```
Age=avg_age.mean().reset_index()
Age
```

Out[45]:

	index	0
0	age	6.050000e+01
1	price	1.090556e+06

To create my regression model, I first created a table to only include the most important features of a house to home buyers.

In [46]:

```
#create table to only include columns of importance
clist=['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'view','condition','sewer_system', 'sqft_garage', 'sqft_patio','age']
imp_feats=dfkc_copy5[clist]
imp_feats.head()
```

Out[46]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	view	condition	sewer_system
0	675000.0	4	1.0	1180	7140	NONE	Good	PUBLIC
1	920000.0	5	2.5	2770	6703	AVERAGE	Average	PUBLIC
2	311000.0	6	2.0	2880	6156	AVERAGE	Average	PUBLIC
3	775000.0	3	3.0	2160	1400	AVERAGE	Average	PUBLIC
4	592500.0	2	2.0	1120	758	NONE	Average	PUBLIC

In [47]:

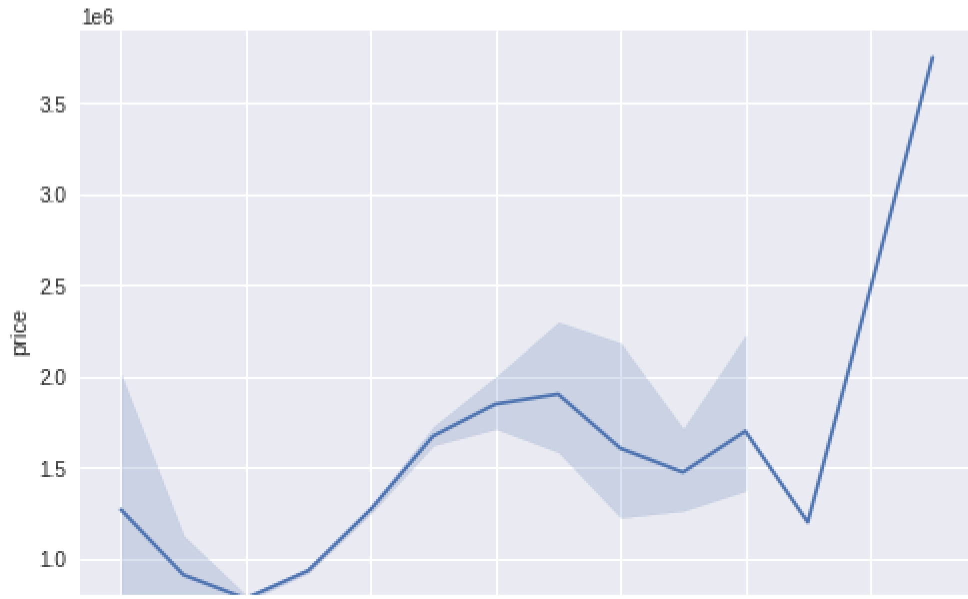
```
imp_feats.describe()
```

Out[47]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_garage	
count	3.011100e+04	30111.000000	30111.000000	30111.000000	3.011100e+04	30111.000000	3(
mean	1.108971e+06	3.415197	2.335708	2113.342798	1.664880e+04	330.475308	
std	8.965158e+05	0.979755	0.888293	973.453260	5.993303e+04	285.725020	
min	2.736000e+04	0.000000	0.000000	3.000000	4.020000e+02	0.000000	
25%	6.492360e+05	3.000000	2.000000	1420.000000	4.850000e+03	0.000000	
50%	8.600000e+05	3.000000	2.500000	1920.000000	7.477000e+03	400.000000	
75%	1.300000e+06	4.000000	3.000000	2620.000000	1.056800e+04	510.000000	
max	3.075000e+07	13.000000	10.500000	15360.000000	3.253932e+06	3580.000000	4

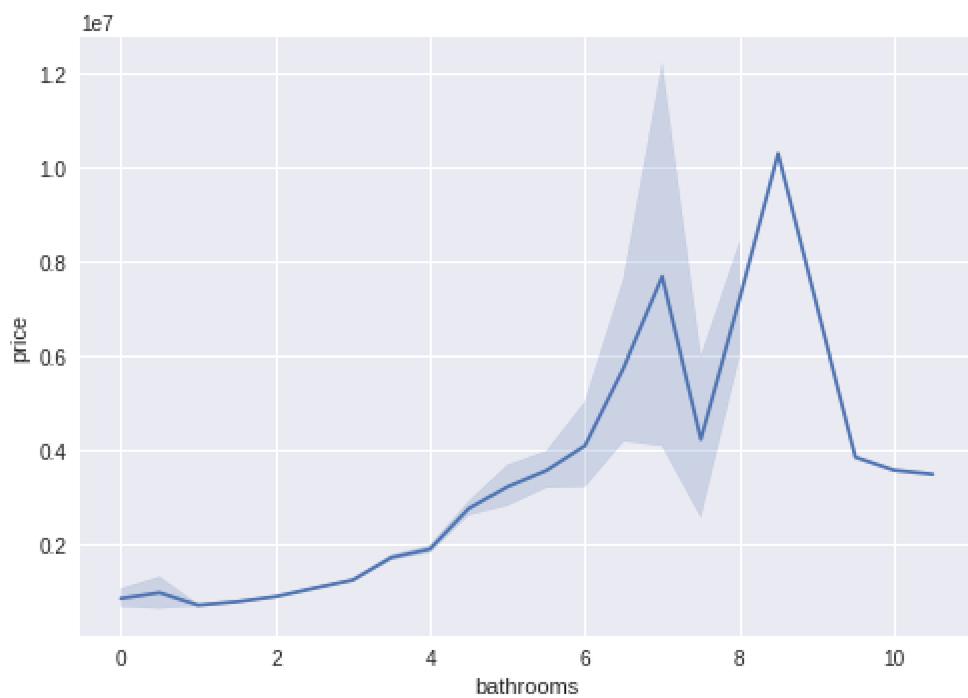
In [48]:
```
sns.lineplot(data=imp_feats, x='bedrooms', y="price")
```

Out[48]: <AxesSubplot:xlabel='bedrooms', ylabel='price'>

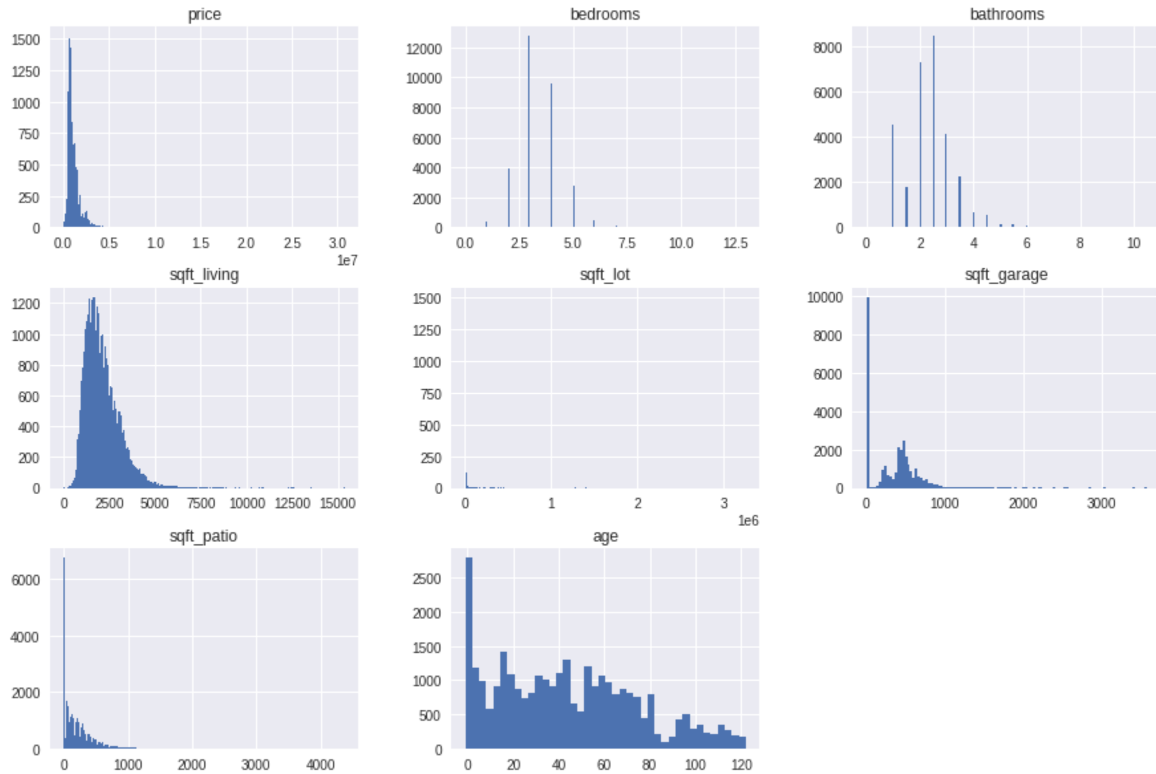


In [49]:
```
sns.lineplot(data=imp_feats, x="bathrooms", y="price")
```

Out[49]: <AxesSubplot:xlabel='bathrooms', ylabel='price'>



In [50]:
```
imp_feats.hist(figsize=(15,10), bins="auto");
```



Modeling

Baseline Model

In [51]:
```
imp_feats.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30111 entries, 0 to 30154
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   price         30111 non-null  float64
 1   bedrooms      30111 non-null  int64  
 2   bathrooms     30111 non-null  float64
 3   sqft_living   30111 non-null  int64  
 4   sqft_lot      30111 non-null  int64  
 5   view          30111 non-null  object 
 6   condition     30111 non-null  object 
 7   sewer_system  30111 non-null  object 
 8   sqft_garage   30111 non-null  int64  
 9   sqft_patio    30111 non-null  int64  
 10  age           30111 non-null  int64  
dtypes: float64(2), int64(6), object(3)
memory usage: 2.8+ MB
```

In [52]:
```
imp_feats_num = [x for x in imp_feats.columns if x not in ['date_sold','price
imp_feats_num
```

Out[52]:
```
['bedrooms',
 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'sqft_garage',
 'sqft_patio',
 'age']
```

In [53]:
```
preds = imp_feats[imp_feats_num]
target = imp_feats.price
```

In [54]:
```
y= target
X= preds
model = sm.OLS(y, sm.add_constant(X))
results = model.fit()
results.summary()
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142: F
utureWarning: In a future version of pandas all arguments of concat excep
t for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[54]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.404
Model:	OLS	**Adj. R-squared:**	0.404
Method:	Least Squares	**F-statistic:**	2916.
Date:	Mon, 03 Oct 2022	**Prob (F-statistic):**	0.00
Time:	02:32:43	**Log-Likelihood:**	-4.4764e+05
No. Observations:	30111	**AIC:**	8.953e+05
Df Residuals:	30103	**BIC:**	8.954e+05
Df Model:	7		

The model overall explains about 40% of the variance in sale price.

Bedrooms: For each additional bedroom, the price decreases by about $171K

Bathrooms: For each additional bathroom, the price increases by about $133K

Sqft_living: For each additional square foot of living space, the price increases by about $600

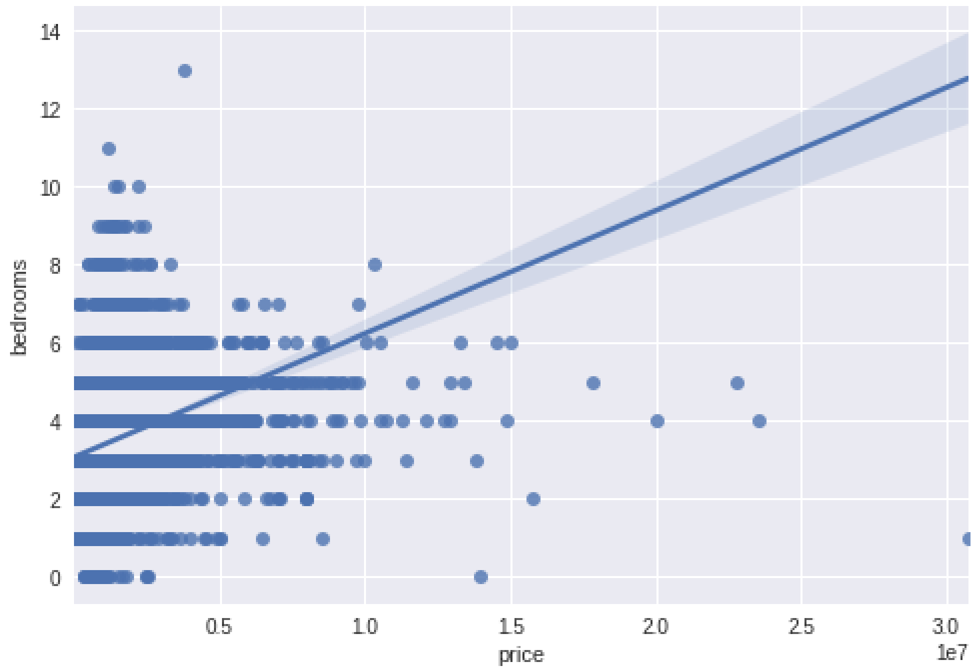
Sqft_lot: For each additional square foot of lot, the price decreases by about $.06

Sqft_garage: For each additional square foot of garage area, the price decreases by about $135

Sqft_patio: For each additional sqare foot of patio area, the price increases by about $265

age: For each additional year the house ages, the price increases by about $2957

In [55]:
```
sns.regplot(x="price", y="bedrooms", data=imp_feats);
```



In [56]:
```
X.dtypes
```

Out[56]:
```
bedrooms          int64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
sqft_garage       int64
sqft_patio        int64
age               int64
dtype: object
```

In [57]:
```
#dummifying categorial columns
cat_columns =['view','condition','sewer_system']
dummy_imp_feats= pd.get_dummies(data=imp_feats, columns=cat_columns, drop_fir
dummy_imp_feats.columns
```

Out[57]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'sqft_garage', 'sqft_patio', 'age', 'view_EXCELLENT', 'view_FAIR',
       'view_GOOD', 'view_NONE', 'condition_Fair', 'condition_Good',
       'condition_Poor', 'condition_Very Good',
       'sewer_system_PRIVATE RESTRICTED', 'sewer_system_PUBLIC',
       'sewer_system_PUBLIC RESTRICTED'],
      dtype='object')
```

In [58]:
```
dummy_imp_feats_cat = imp_feats[cat_columns].copy()
dummy_imp_feats_cat = pd.get_dummies(dummy_imp_feats_cat, columns=cat_columns
dummy_imp_feats_cat
```

Out[58]:

	view_EXCELLENT	view_FAIR	view_GOOD	view_NONE	condition_Fair	condition_Good
0	0	0	0	1	0	1
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	1	0	0
...	...	...	...	...	...	...
30150	0	0	0	1	0	1
30151	0	1	0	0	0	0
30152	0	0	0	1	0	0
30153	0	0	0	1	0	0
30154	0	0	0	1	0	0

30111 rows × 11 columns

In [59]:

```
X=dummy_imp_feats.drop(labels =['price'], axis=1)
y=dummy_imp_feats.price

first_dummy_model = sm.OLS(y,sm.add_constant(X))
results = first_dummy_model.fit()
results.summary()
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142: Fut
ureWarning: In a future version of pandas all arguments of concat except fo
r the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[59]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.443
Model:	OLS	**Adj. R-squared:**	0.443
Method:	Least Squares	**F-statistic:**	1330.
Date:	Mon, 03 Oct 2022	**Prob (F-statistic):**	0.00
Time:	02:32:45	**Log-Likelihood:**	-4.4662e+05
No. Observations:	30111	**AIC:**	8.933e+05
Df Residuals:	30092	**BIC:**	8.934e+05
Df Model:	18		
Covariance Type:	nonrobust		

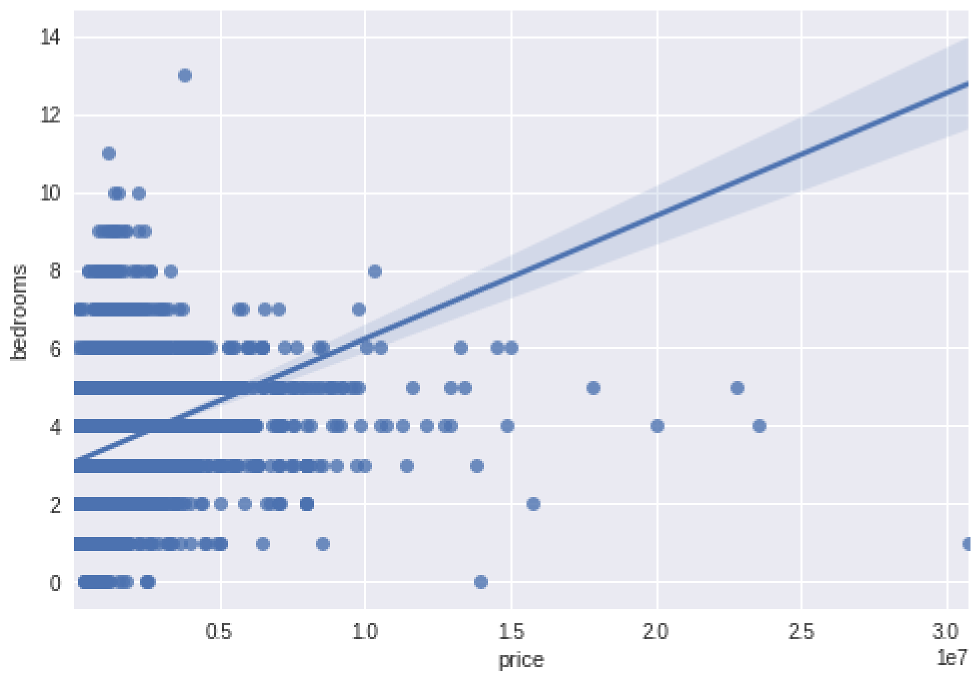
	coef	std err	t	P>|t|	[0.025	0.975]
const	-7.305e+04	2.67e+04	-2.735	0.006	-1.25e+05	-2.07e+04
bedrooms	-1.484e+05	5369.404	-27.643	0.000	-1.59e+05	-1.38e+05
bathrooms	1.097e+05	7730.355	14.186	0.000	9.45e+04	1.25e+05
sqft_living	558.0284	7.438	75.022	0.000	543.449	572.608
sqft_lot	0.3400	0.070	4.863	0.000	0.203	0.477
sqft_garage	-54.1629	16.761	-3.231	0.001	-87.016	-21.310
sqft_patio	215.4326	17.822	12.088	0.000	180.502	250.364
age	2534.6824	164.716	15.388	0.000	2211.831	2857.534
view_EXCELLENT	1.117e+06	3.27e+04	34.123	0.000	1.05e+06	1.18e+06
view_FAIR	2.072e+05	4.77e+04	4.345	0.000	1.14e+05	3.01e+05
view_GOOD	6.359e+04	2.74e+04	2.320	0.020	9868.052	1.17e+05
view_NONE	-1.125e+05	1.62e+04	-6.953	0.000	-1.44e+05	-8.08e+04
condition_Fair	-6.387e+04	4.54e+04	-1.407	0.160	-1.53e+05	2.51e+04
condition_Good	-1.22e+04	9712.258	-1.257	0.209	-3.12e+04	6832.920
condition_Poor	-6.589e+04	8.63e+04	-0.763	0.445	-2.35e+05	1.03e+05
condition_Very Good	1.951e+04	1.37e+04	1.429	0.153	-7256.323	4.63e+04
sewer_system_PRIVATE RESTRICTED	-3.697e+05	3e+05	-1.231	0.218	-9.58e+05	2.19e+05

sewer_system_PUBLIC	2.172e+05	1.22e+04	17.873	0.000	1.93e+05	2.41e+05
sewer_system_PUBLIC RESTRICTED	1.4e+05	3.87e+05	0.362	0.717	-6.18e+05	8.98e+05

Omnibus:	41771.818	**Durbin-Watson:**	1.842
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	42988029.275
Skew:	7.589	**Prob(JB):**	0.00
Kurtosis:	187.481	**Cond. No.**	6.24e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.24e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [60]:
```
sns.regplot(x="price", y="bedrooms", data=dummy_imp_feats);
```



This dummy model explains about 44% of the variance in sale price. According to this model, the following is found:

Each additional bedroom, the price decreases by about $148K.

Each additional bathroom, the price increases by about $109K.

Each additional square foot of living space, the price increases by about $558.

Each additional square foot of lot, the price decreases by about $.34.

Each additional square foot of garage area, the price decreases by about $54.

Each additional sqare foot of patio area, the price increases by about $215.

Each additional year the house ages, the price increases by about $2535.

Now I will explore multicollinearity problems.

In [61]:
```
imp_feats.corr()
```

Out[61]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_garage	sqft_patio
price	1.000000	0.288954	0.480337	0.608616	0.086550	0.263674	0.313789
bedrooms	0.288954	1.000000	0.588035	0.637048	0.006215	0.318110	0.183660
bathrooms	0.480337	0.588035	1.000000	0.772226	0.038028	0.456264	0.327982
sqft_living	0.608616	0.637048	0.772226	1.000000	0.122271	0.510967	0.396530
sqft_lot	0.086550	0.006215	0.038028	0.122271	1.000000	0.089318	0.154575
sqft_garage	0.263674	0.318110	0.456264	0.510967	0.089318	1.000000	0.216512
sqft_patio	0.313789	0.183660	0.327982	0.396530	0.154575	0.216512	1.000000
age	-0.126909	-0.156650	-0.471854	-0.312269	-0.003427	-0.409075	-0.157426

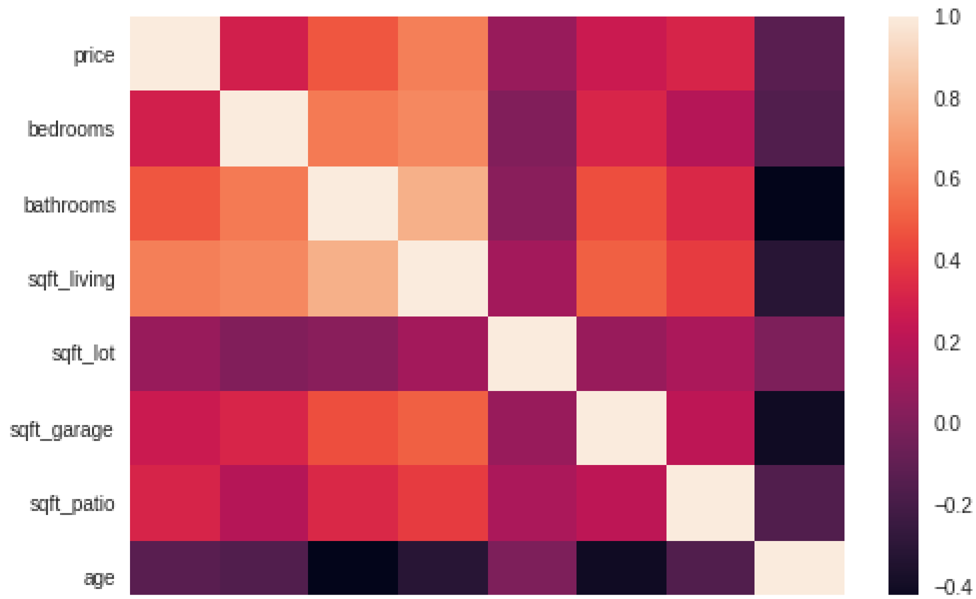
In [62]:
```
abs(imp_feats.corr()) > 0.75
```

Out[62]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_garage	sqft_patio	age
price	True	False	False	False	False	False	False	False
bedrooms	False	True	False	False	False	False	False	False
bathrooms	False	False	True	True	False	False	False	False
sqft_living	False	False	True	True	False	False	False	False
sqft_lot	False	False	False	False	True	False	False	False
sqft_garage	False	False	False	False	False	True	False	False
sqft_patio	False	False	False	False	False	False	True	False
age	False	False	False	False	False	False	False	True

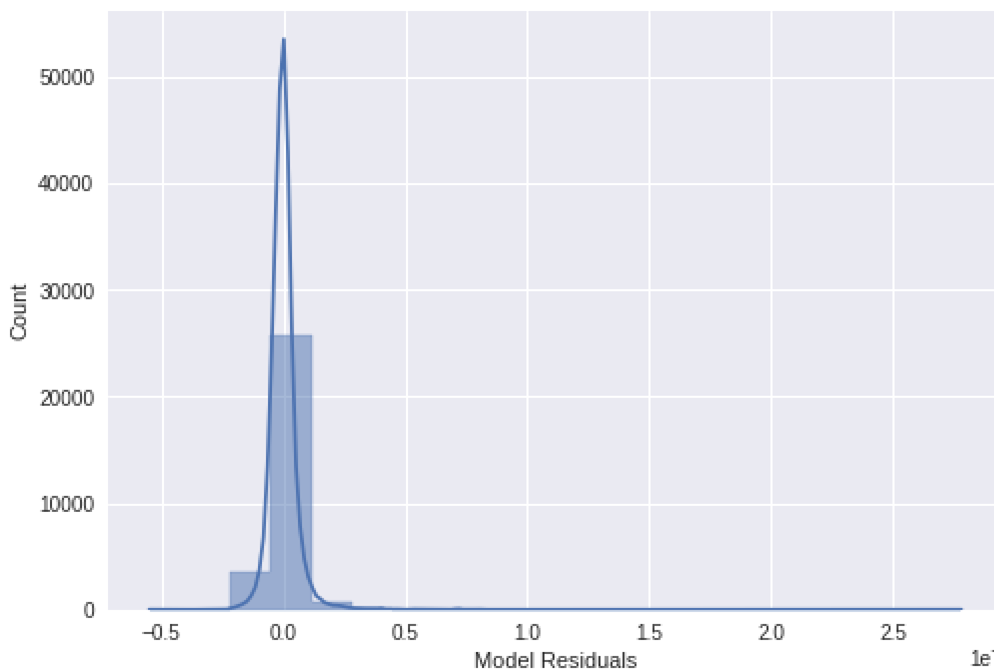
In [63]:
```
sns.heatmap(imp_feats.corr())
```

Out[63]: <AxesSubplot:>



Using a heatmap, it appears that living space has the greatest impact on price, followed by number of bathrooms and number of bedrooms. The least important feature is the age of the house.

In [64]:
```
fig, ax = plt.subplots()
sns.histplot(results.resid, bins=20, element="step", kde=True, ax=ax)
ax.set_xlabel("Model Residuals");
```



The residuals from the actual model are skewed and not quite normal.

In [65]:

```
#get t test p-values
results.pvalues
```

Out[65]:

```
const                              6.248366e-03
bedrooms                          4.011338e-166
bathrooms                          1.577300e-45
sqft_living                        0.000000e+00
sqft_lot                           1.159479e-06
sqft_garage                        1.233103e-03
sqft_patio                         1.456048e-33
age                                3.136302e-53
view_EXCELLENT                    1.985477e-250
view_FAIR                          1.398813e-05
view_GOOD                          2.034182e-02
view_NONE                          3.650523e-12
condition_Fair                     1.595540e-01
condition_Good                     2.089419e-01
condition_Poor                     4.452540e-01
condition_Very Good                1.531130e-01
sewer_system_PRIVATE RESTRICTED    2.182960e-01
sewer_system_PUBLIC                4.476682e-71
sewer_system_PUBLIC RESTRICTED     7.172644e-01
dtype: float64
```

In [66]:

```
#get the 95% confidence interval for the coefficients
print(results.conf_int())
```

```
                                            0             1
const                           -1.254051e+05 -2.069133e+04
bedrooms                        -1.589524e+05 -1.379039e+05
bathrooms                        9.450807e+04  1.248117e+05
sqft_living                      5.434493e+02  5.726075e+02
sqft_lot                         2.029998e-01  4.770850e-01
sqft_garage                     -8.701588e+01 -2.130995e+01
sqft_patio                       1.805017e+02  2.503636e+02
age                              2.211831e+03  2.857534e+03
view_EXCELLENT                   1.052504e+06  1.180784e+06
view_FAIR                        1.137026e+05  3.006107e+05
view_GOOD                        9.868052e+03  1.173038e+05
view_NONE                       -1.442618e+05 -8.081293e+04
condition_Fair                  -1.528658e+05  2.512921e+04
condition_Good                  -3.123996e+04  6.832920e+03
condition_Poor                  -2.350794e+05  1.032954e+05
condition_Very Good             -7.256323e+03  4.627436e+04
sewer_system_PRIVATE RESTRICTED -9.582338e+05  2.188873e+05
sewer_system_PUBLIC              1.933736e+05  2.410107e+05
sewer_system_PUBLIC RESTRICTED  -6.178649e+05  8.979057e+05
```

Analysis

My analysis shows that the following averages of houses sold had: 3 bedrooms, 2.5 bathrooms, 2100 sqft living space, with 1.5 car garage, a patio and an age of around 45 years old. In my findings, it was noticable that there was a threshold where additional space began to lose value.

Conclusion

The perfect house that home buyers are looking for have an excellent view, a public sewer system, 6 bedrooms, 5 bathrooms, is in average condition with living space of around 3100 sqft and no older than 10 years. My analysis leads to the following recommendations for Noznas Inc. to successfully increase their housing sales. They should build houses with similar specifications stated above, calculating their asking price around the square footage of living space, number of bedrooms and the number of bathrooms. Directing their advertising towards middle class families will result in more attention and traffic to their sales department because these families are who need the above features.

In [ ]:

```
```