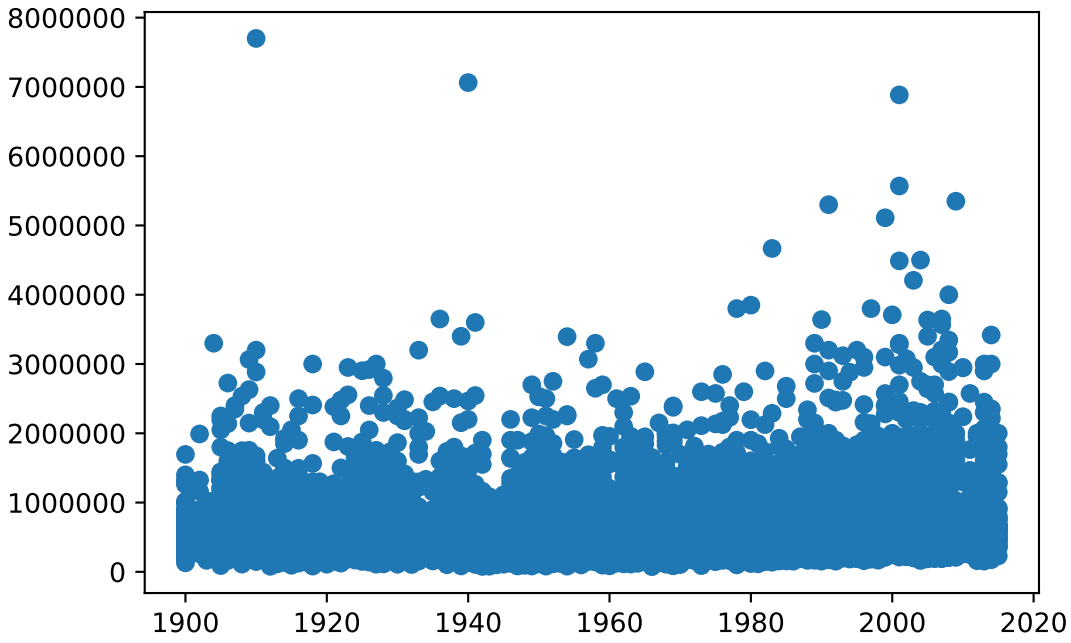
Data Visualization

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

data = pd.read_csv("../housing_data.csv")
```

Scatter plots allow you to visualize a large number of data points on one plot:

In [ ]:
```
fig, ax = plt.subplots()

ax.scatter(data["yr_built"], data["price"])
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7fde1c824a90>
```



Next we're going to create a line graph that allows us to visualize values over time very well. But, since we have several houses built each year all at a different price, we'll find the average price of a house built in a given year (using the `.groupby()` function discussed earlier), and graph that over time.

In [ ]:
```
price_by_yr = data[["yr_built","price"]].groupby("yr_built").mean()
price_by_yr
```

Out[ ]:

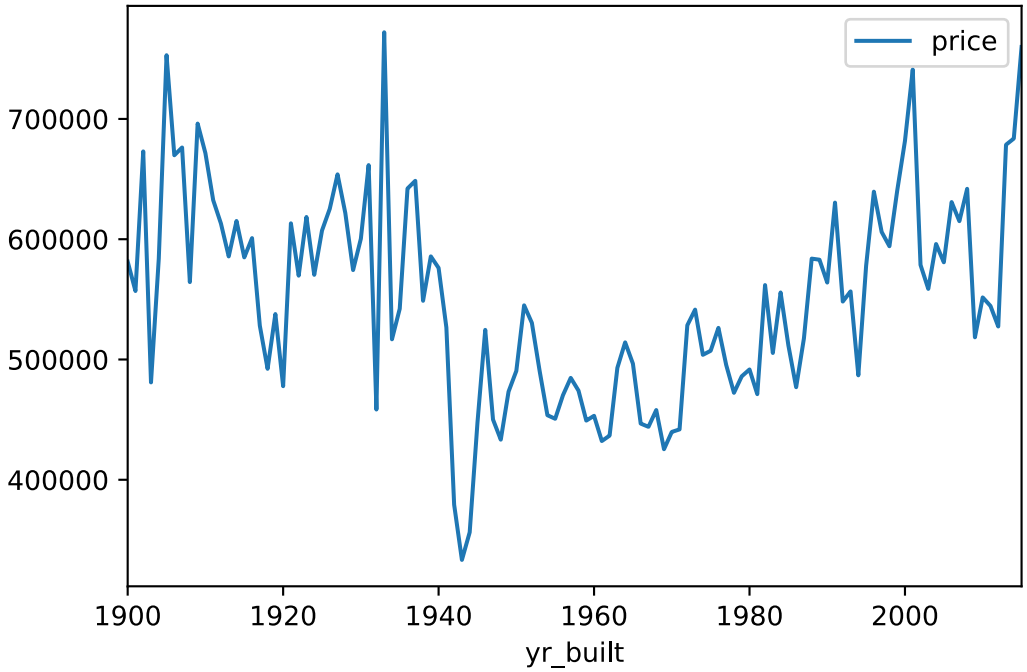
yr_built	price
1900	581387.206897
1901	556935.931034
1902	673007.407407
1903	480958.195652
1904	583756.644444
1905	752977.986486
1906	669799.402174
1907	676257.246154
1908	564348.686047
1909	696135.159574
1910	671536.313433
1911	632488.356164
1912	612990.696203
1913	585683.220339
1914	615153.481481
1915	584896.296875
1916	600915.037975
1917	528108.928571
1918	492246.875000
1919	537779.602273
1920	477804.397959
1921	613204.473684
1922	569688.884211
1923	618475.202381
1924	570391.151079
1925	607219.393939
1926	625315.600000
1927	653921.600000
1928	621713.849206
1929	574335.438596
...	...
1986	476953.488372
1987	517553.955782
1988	583860.029630

yr_built	price
1989	582961.679310
1990	563966.009375
1991	630440.915179
1992	548169.308081
1993	556611.940594
1994	486833.518072
1995	577771.035503
1996	639534.246154
1997	606058.067797
1998	594158.807531
1999	640290.233962
2000	681789.169725
2001	741030.796721
2002	578638.752252
2003	558694.258294
2004	595998.745958
2005	580811.291111
2006	630880.072687
2007	615004.563549
2008	641903.792916
2009	518442.621739
2010	551628.664336
2011	544522.000000
2012	527447.335294
2013	678545.452736
2014	683681.754919
2015	759785.157895

116 rows × 1 columns

In [ ]:
```
price_by_yr.plot.line()
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1bf80c88>
```

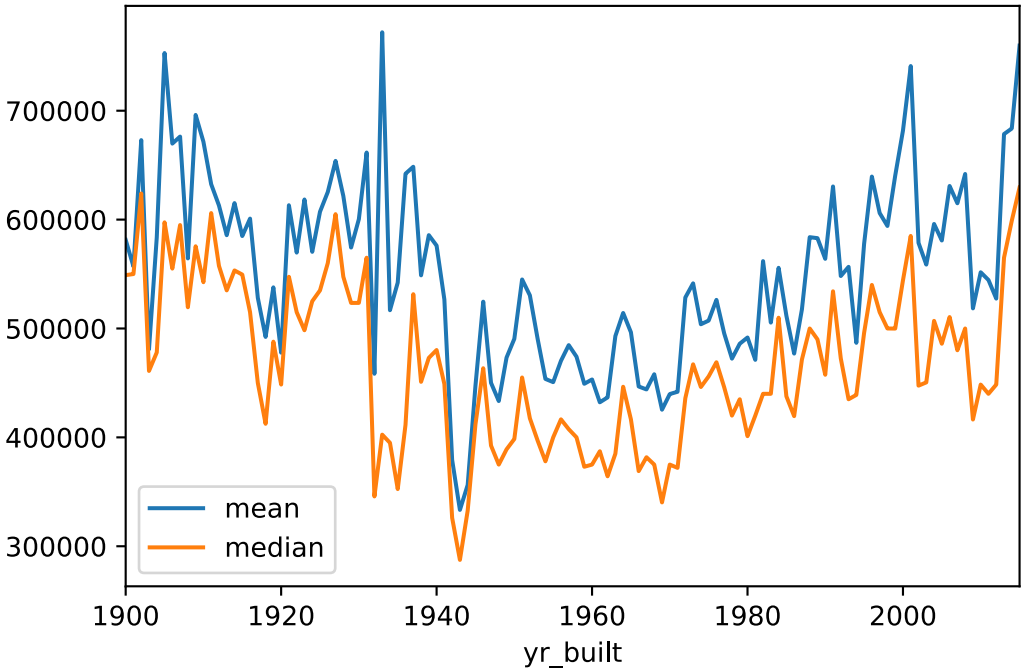


Like we talked about in the last lesson, sometimes the mean isn't a comepltely accurate measure of the center, and median is more useful. We can actual find the median house price for each year and graph that on the same plot as well.

Find the median prices in the same way we found the mean prices, and then combine the two mean and median DataFrames into one DataFrame which we will turn into one line plot by using `.concat()`

In [ ]:
```
price_by_yr_med = data[["yr_built","price"]].groupby("yr_built").median()
price_by_yr = price_by_yr.rename(columns={'price':'mean'})
price_by_yr_med = price_by_yr_med.rename(columns={'price':'median'})
plot_data = pd.concat([price_by_yr, price_by_yr_med], axis=1)
plot_data.plot.line()
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1c287898>
```

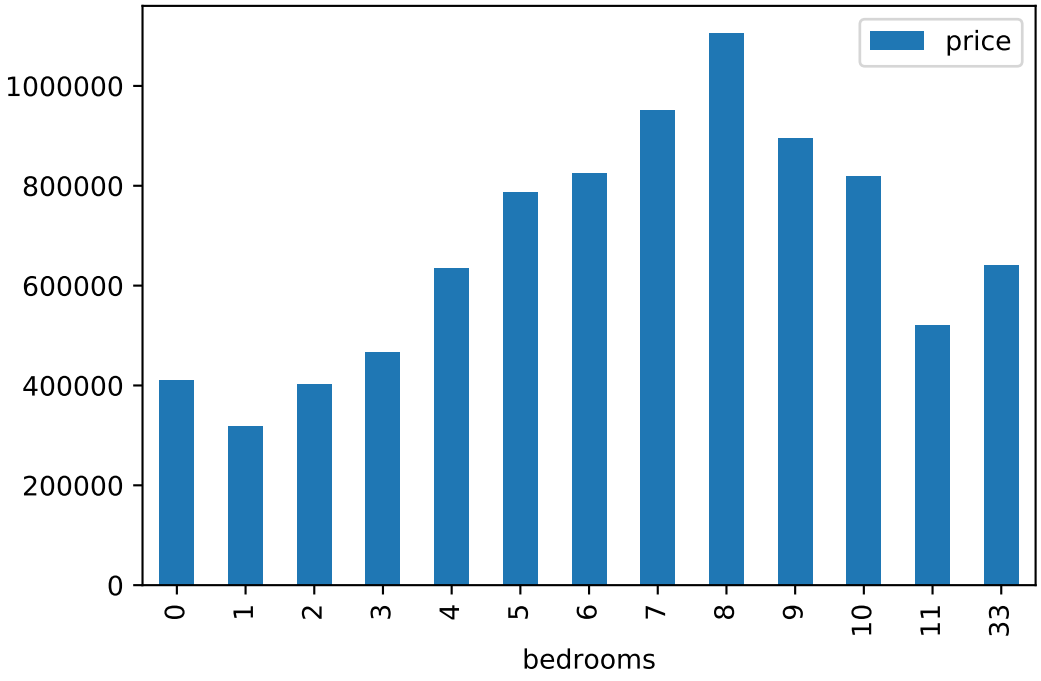


To visualize categorical data (such as how many houses fall into some given category), a great way to go is often a bar chart. Here we can visualize the average price of houses with a given number of bedrooms, by divided the data into categories where each number of bedrooms is its own category:

In [ ]:
```
price_by_num_rooms = data[["bedrooms","price"]].groupby("bedrooms").mean()
```

In [ ]:
```
price_by_num_rooms.plot.bar()
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1bf8b320>
```

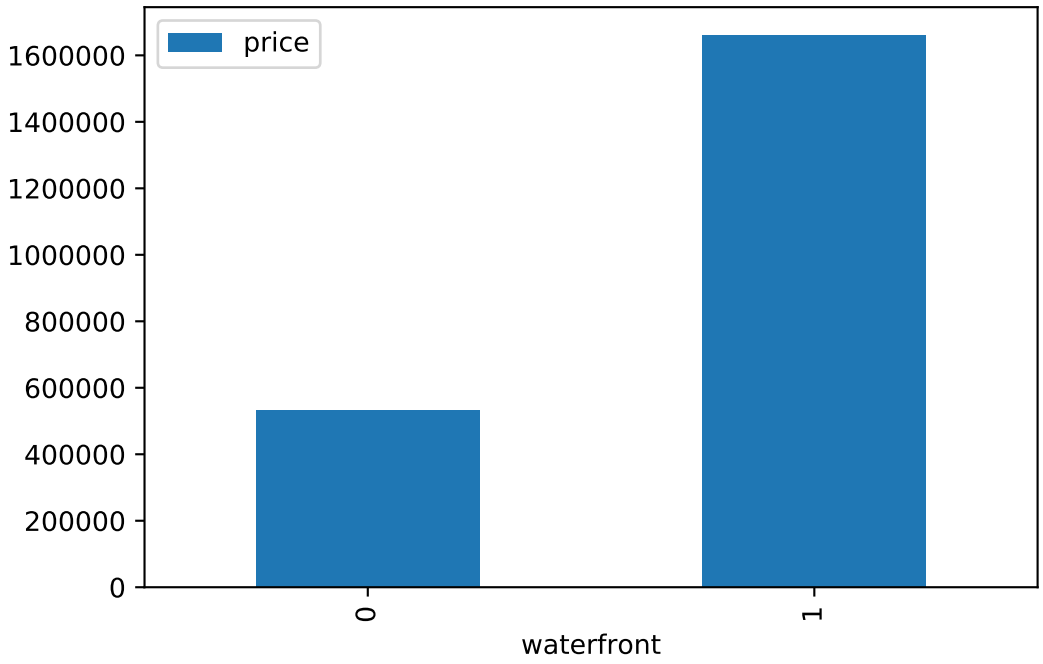


Another very useful application of bar charts is comparing a value of one variable when you

change another variable, for example comparing the average price of a house when you change whether or not it's on the waterfront. This lets us see whether or not there is a big difference in the first variable caused by the second variable.

In [ ]:
```
price_by_waterfront = data[["waterfront","price"]].groupby("waterfront").mean()
price_by_waterfront.plot.bar()
```

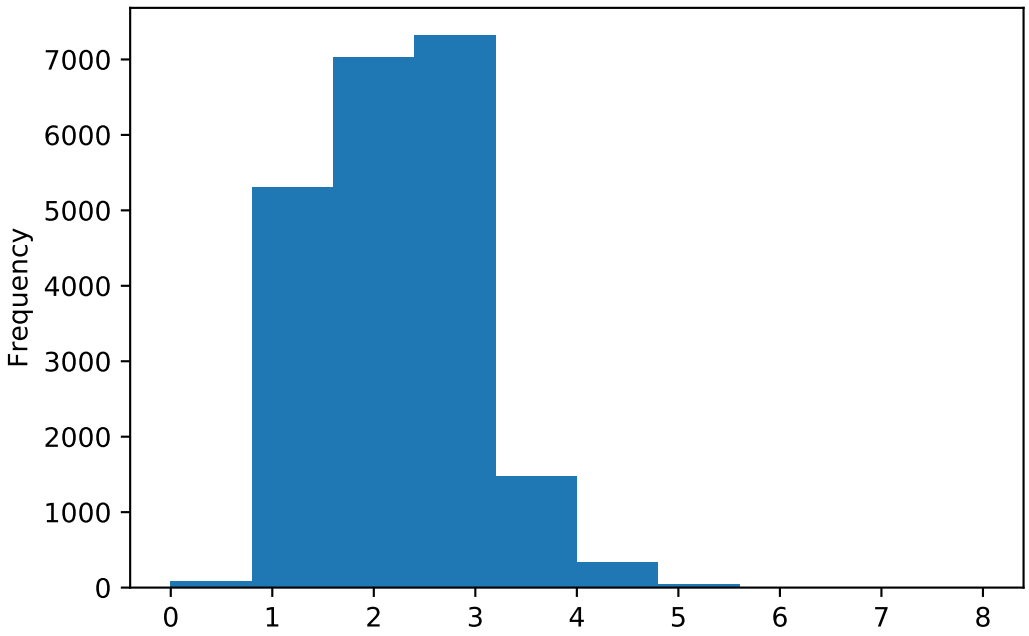
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1c57d668>
```



Finally, histograms can often help us understand the distribution of data by showing how often (the frequency) a given value occurs in a dataset. So, here, we can use a histogram to show us how many houses in our data set have a given number of bathrooms:

In [ ]:
```
data["bathrooms"].plot.hist(bins=10)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1c634668>
```


Frequency
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
8