In [1]: ▶

#### # Import packages

import pandas as pd # Reading csv file

from shapely.geometry import Point # Shapely for converting Latitude/Longtitude to ge
import geopandas as gpd # To create GeodataFrame
import matplotlib.pyplot as plt # for plotting

from pyproj import CRS

In [2]:

```
# import housing data
df_all = pd.read_csv('data/kc_house_data.csv')
df_all.head(10)
```

#### Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	7129300520	10/13/2014	221900	3	1.00	1180	5650	1.0
1	6414100192	12/9/2014	538000	3	2.25	2570	7242	2.0
2	5631500400	2/25/2015	180000	2	1.00	770	10000	1.0
3	2487200875	12/9/2014	604000	4	3.00	1960	5000	1.0
4	1954400510	2/18/2015	510000	3	2.00	1680	8080	1.0
5	7237550310	5/12/2014	1230000	4	4.50	5420	101930	1.0
6	1321400060	6/27/2014	257500	3	2.25	1715	6819	2.0
7	2008000270	1/15/2015	291850	3	1.50	1060	9711	1.0
8	2414600126	4/15/2015	229500	3	1.00	1780	7470	1.0
9	3793500160	3/12/2015	323000	3	2.50	1890	6560	2.0

10 rows × 21 columns

```
In [3]:
```

```
# clean data--convert "NaN" to 0 and replace "?" with 0
df_all = df_all.fillna(0).replace('?',0)
df_all.head(10)
```

### Out[3]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
7129300520	10/13/2014	221900	3	1.00	1180	5650	1.0
6414100192	12/9/2014	538000	3	2.25	2570	7242	2.0
5631500400	2/25/2015	180000	2	1.00	770	10000	1.0
2487200875	12/9/2014	604000	4	3.00	1960	5000	1.0
1954400510	2/18/2015	510000	3	2.00	1680	8080	1.0
7237550310	5/12/2014	1230000	4	4.50	5420	101930	1.0
1321400060	6/27/2014	257500	3	2.25	1715	6819	2.0
2008000270	1/15/2015	291850	3	1.50	1060	9711	1.0
2414600126	4/15/2015	229500	3	1.00	1780	7470	1.0
3793500160	3/12/2015	323000	3	2.50	1890	6560	2.0
	7129300520 6414100192 5631500400 2487200875 1954400510 7237550310 1321400060 2008000270 2414600126	7129300520 10/13/2014 6414100192 12/9/2014 5631500400 2/25/2015 2487200875 12/9/2014 1954400510 2/18/2015 7237550310 5/12/2014 1321400060 6/27/2014 2008000270 1/15/2015 2414600126 4/15/2015	7129300520 10/13/2014 221900 6414100192 12/9/2014 538000 5631500400 2/25/2015 180000 2487200875 12/9/2014 604000 1954400510 2/18/2015 510000 7237550310 5/12/2014 1230000 1321400060 6/27/2014 257500 2008000270 1/15/2015 291850 2414600126 4/15/2015 229500	7129300520       10/13/2014       221900       3         6414100192       12/9/2014       538000       3         5631500400       2/25/2015       180000       2         2487200875       12/9/2014       604000       4         1954400510       2/18/2015       510000       3         7237550310       5/12/2014       1230000       4         1321400060       6/27/2014       257500       3         2008000270       1/15/2015       291850       3         2414600126       4/15/2015       229500       3	7129300520       10/13/2014       221900       3       1.00         6414100192       12/9/2014       538000       3       2.25         5631500400       2/25/2015       180000       2       1.00         2487200875       12/9/2014       604000       4       3.00         1954400510       2/18/2015       510000       3       2.00         7237550310       5/12/2014       1230000       4       4.50         1321400060       6/27/2014       257500       3       2.25         2008000270       1/15/2015       291850       3       1.50         2414600126       4/15/2015       229500       3       1.00	7129300520       10/13/2014       221900       3       1.00       1180         6414100192       12/9/2014       538000       3       2.25       2570         5631500400       2/25/2015       180000       2       1.00       770         2487200875       12/9/2014       604000       4       3.00       1960         1954400510       2/18/2015       510000       3       2.00       1680         7237550310       5/12/2014       1230000       4       4.50       5420         1321400060       6/27/2014       257500       3       2.25       1715         2008000270       1/15/2015       291850       3       1.50       1060         2414600126       4/15/2015       229500       3       1.00       1780	7129300520         10/13/2014         221900         3         1.00         1180         5650           6414100192         12/9/2014         538000         3         2.25         2570         7242           5631500400         2/25/2015         180000         2         1.00         770         10000           2487200875         12/9/2014         604000         4         3.00         1960         5000           1954400510         2/18/2015         510000         3         2.00         1680         8080           7237550310         5/12/2014         1230000         4         4.50         5420         101930           1321400060         6/27/2014         257500         3         2.25         1715         6819           2008000270         1/15/2015         291850         3         1.50         1060         9711           2414600126         4/15/2015         229500         3         1.00         1780         7470

10 rows × 21 columns

# In [4]:

# want to look at houses with two bedrooms or less
df = df\_all[(df\_all['bedrooms'] <= 2)]
df.info</pre>

# Out[4]:

<pre><bound dataframe.info="" method="" or<="" pre=""></bound></pre>						io	t	date	e pr	ice
bedroo		•	t_livi	_	•	2		4 00		770
2	5631500400		/2015	18000		2		1.00		770
11	9212900260	-	/2014	46800		2		1.00		1160
18	16000397		/2014	18900		2		1.00		1200
23	8091400200			25270		2		1.50		1070
31	2426039314	12/1	/2014	28000	0	2		1.50		1190
• • •	• • •			• •		• • •		• • •		• • •
21570	2767604724			50500		2		2.50		1430
21572	2767600688	-		41450		2		1.50		1210
21579	1972201967	-		52000	0	2		2.25		1530
21594	1523300141	6/23	/2014	40210	1	2		0.75		1020
21596	1523300157	10/15	/2014	32500	0	2		0.75		1020
	sqft_lot ·	floors	water	rfront	view	{	grade	sqft_a	above	\
2	10000	1.0		0.0	0.0		6		770	
11	6000	1.0		0.0	0.0		7		860	
18	9850	1.0		0.0	0.0		7		1200	
23	9643	1.0		0.0	0.0		7		1070	
31	1265	3.0		0.0	0.0	• • •	7		1190	
	4204	• • •		• • •	• • •	• • •	• • •			
21570	1201	3.0		0.0	0.0	• • •	8		1430	
21572	1278	2.0		0.0	0.0	• • •	8		1020	
21579	981	3.0		0.0	0.0	• • •	8		1480	
21594	1350	2.0		0.0	0.0	• • •	7		1020	
21596	1076	2.0		0.0	0.0	• • •	7		1020	
	sqft_baseme	nt yr_b	uilt	yr_ren	ovated	zipco	ode	lat	1	ong
\										
2		0	1933		0.0	986	928 4	7.7379	-122.	233
11	30	00	1942		0.0	981	L15 4	7.6900	-122.	292
18		0	1921		0.0	986	902 4	7.3089	-122.	210
23		0	1985		0.0	986	930 4	7.3533	-122.	166
31		0	2005		0.0	981	L33 4	7.7274	-122.	357
• • •	•	• •	• • •		• • •		• •	· · ·		• • •
21570			2009		0.0			7.6707		
21572			2007		0.0			7.6756		
21579	!		2006		0.0			7.6533		
21594			2009		0.0			7.5944		
21596		0	2008		0.0	981	L44 4	7.5941	-122.	299
	sqft_living15 sqft_lot15									

localhost:8890/notebooks/Documents/Flatiron/dsc-phase-2-project/data.ipynb

8062

2720

2

```
11
                  1330
                                6000
                                5095
18
                  1060
23
                  1220
                                8386
31
                  1390
                                1756
. . .
                                 . . .
21570
                  1430
                                1249
21572
                  1210
                                1118
21579
                  1530
                                1282
21594
                  1020
                                2007
21596
                  1020
                                1357
```

```
[2956 rows x 21 columns]>
```

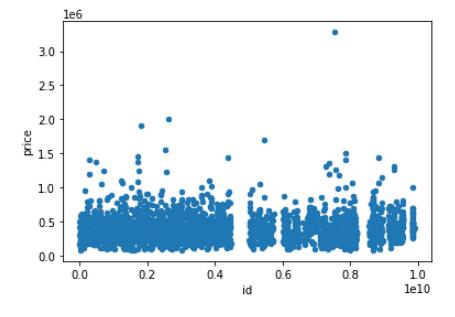
In [5]:

# want to look at price range

```
# want to look at price range
df.plot(kind='scatter', x='id', y='price')
```

### Out[5]:

<AxesSubplot:xlabel='id', ylabel='price'>



# In [6]:

# limit price range to less than \$2 mill to remove obvious outlier
df = df[(df['price'] <= 2000000)]
df.info</pre>

### Out[6]:

<bound< th=""><th>method Data</th><th></th><th></th><th></th><th>id</th><th>dat</th><th>ce price</th></bound<>	method Data				id	dat	ce price
2	5631500400	2/25/2015	_	a	2	1.00	770
11	9212900260	5/27/2014			2	1.00	1160
18	16000397	12/5/2014			2	1.00	1200
23	8091400200	5/16/2014			2	1.50	1070
31	2426039314	12/1/2014			2	1.50	1190
 21570	 2767604724	10/15/2014			2	2.50	 1430
21572	2767600688	11/13/2014			2	1.50	1210
21579	1972201967	10/31/2014			2	2.25	1530
21594	1523300141	6/23/2014			2	0.75	1020
21596	1523300157	10/15/2014			2	0.75	1020
		, ,					
	sqft_lot f	loors wate	erfront	view	gr	ade sqft_	_above \
2	10000	1.0	0.0	0.0	• • •	6	770
11	6000	1.0	0.0	0.0	• • •	7	860
18	9850	1.0	0.0	0.0	• • •	7	1200
23	9643	1.0	0.0	0.0		7	1070
31	1265	3.0	0.0	0.0	• • •	7	1190
• • •	• • •	• • •		• • •	• • •	• • •	• • •
21570	1201	3.0	0.0	0.0		8	1430
21572	1278	2.0	0.0	0.0	• • •	8	1020
21579	981	3.0	0.0	0.0	• • •	8	1480
21594	1350	2.0	0.0	0.0		7	1020
21596	1076	2.0	0.0	0.0	• • •	7	1020
_	sqft_basemen	nt yr_built	yr_ren	ovated	zipcod	e lat	long
\							
2		0 1933		0.0	9802		-122.233
11	30			0.0	9811		-122.292
18		0 1921		0.0	9800		-122.210
23		0 1985		0.0		0 47.3533	
31		0 2005		0.0	9813	3 4/./2/4	-122.357
				• • •			
21570		0 2009		0.0	9810		7 -122.381
21572	19			0.0	9811		5 -122.375
21579		2006		0.0	9810		3 -122.346
21594		0 2009		0.0	9814		-122.299
21596		0 2008		0.0	9814	4 4/.5941	-122.299
	sqft_living	15 saft 1	nt15				
2	. – .	220	2062				

localhost:8890/notebooks/Documents/Flatiron/dsc-phase-2-project/data.ipynb

8062

2720

2

```
11
                  1330
                                6000
18
                  1060
                                5095
23
                  1220
                                8386
31
                  1390
                                1756
. . .
                   . . .
                                  . . .
21570
                  1430
                                1249
21572
                  1210
                                1118
21579
                  1530
                                1282
21594
                                2007
                  1020
21596
                  1020
                                1357
```

[2955 rows x 21 columns]>

```
In [7]:
```

H

```
# function to create GeoDataFrame
def add_geo_col(df):
    # create a geometry column
    geometry = [Point(xy) for xy in zip(df['long'], df['lat'])]

# Coordinate reference system : WGS84 (the GPS model for conversion)
    crs = CRS('epsg:4326')

# Creating a Geographic data frame
    gdf = gpd.GeoDataFrame(df, crs=crs, geometry=geometry).reset_index()
    return gdf
```

In [8]: ▶

```
gdf = add_geo_col(df)
gdf.head()
```

# Out[8]:

	index	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	flc
0	2	5631500400	2/25/2015	180000	2	1.0	770	10000	
1	11	9212900260	5/27/2014	468000	2	1.0	1160	6000	
2	18	16000397	12/5/2014	189000	2	1.0	1200	9850	
3	23	8091400200	5/16/2014	252700	2	1.5	1070	9643	
4	31	2426039314	12/1/2014	280000	2	1.5	1190	1265	

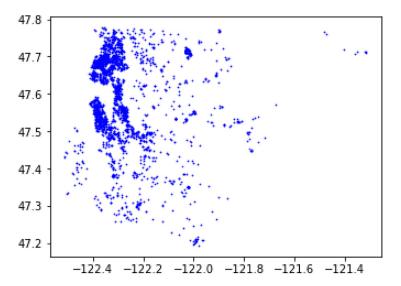
5 rows × 23 columns

In [9]: ▶

```
# Plot all points
gdf.plot(marker='o', color='b', markersize=0.5)
```

# Out[9]:

# <AxesSubplot:>



```
In [10]:
# check what projection is used
gdf.crs
Out[10]:
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
In [11]:
# function to convert meters to miles
def m 2 mi(meters):
    return meters * 0.00062137
In [12]:
# function to find distance between two Points
def dist to point(point1, point2):
    pt1 gdf = gpd.GeoSeries([point1], crs=4326)
    pt2 gdf = gpd.GeoSeries([point2], crs=4326)
    pt1 gdf = pt1 gdf.to crs(3857)
    pt2_gdf = pt2_gdf.to_crs(3857)
    distance = m 2 mi(pt1 gdf.distance(pt2 gdf))
    return round(distance.at[0], 3)
```

```
In [13]:
```

```
# Want to compile list of top 5 employers in Seattle area with a centralized campus
# Source is https://www.huduser.gov/portal/publications/pdf/SeattleWA-CHMA-19.pdf
df_top10_employers = pd.read_csv('data/top_employers.csv')
df_central5 = df_top10_employers[df_top10_employers['centralized_campus']=='y'].reset
df_central5 = add_geo_col(df_central5)
df_central5
```

#### Out[13]:

	level_0	index	rank	employer	no_employees	long	lat	centralized
0	0	0	1	The Boeing Company	64,300	-122.312023	47.532685	
1	1	1	2	Amazon.com, Inc.	45,000	-122.339688	47.615875	
2	2	2	3	Microsoft Corporation	43,031	-122.339688	47.645744	
3	3	3	4	University of Washington	30,200	-122.303644	47.655544	
4	4	7	8	Starbucks Corporation	11,239	-122.336000	47.580700	
4								•

```
In [14]: ▶
```

```
# function to find the average distance of a house to the top 5 employers
def avg_dists(point1, gdf):
    dists = [dist_to_point(point1, point2) for point2 in gdf['geometry']]
    avg = sum(dists)/len(dists)
    return avg
```

```
In [15]:
```

```
downtown = Point(-122.3344, 47.6050) # coordinates for center of downtown Seattle

# Calculate average distance of each property to central downtown
gdf['dist_2_downtown'] = [dist_to_point(point, downtown) for point in gdf['geometry']
```

H

```
In [16]:

# Calculate average distance of each property to the top 5 employers
gdf['avg_dists'] = [avg_dists(point, df_central5) for point in gdf['geometry']]

In [17]:
```

gdf.head()

#### Out[17]:

	index	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	flo
0	2	5631500400	2/25/2015	180000	2	1.0	770	10000	
1	11	9212900260	5/27/2014	468000	2	1.0	1160	6000	
2	18	16000397	12/5/2014	189000	2	1.0	1200	9850	
3	23	8091400200	5/16/2014	252700	2	1.5	1070	9643	
4	31	2426039314	12/1/2014	280000	2	1.5	1190	1265	

5 rows × 25 columns

In [18]:

gdf.to\_pickle('data/geodata.pkl')

H