East Bay Area Apartment Prices: A Web Scraping Project

February 3, 2019

1 Webscrape the East Bay for category: Rooms/Shares

In [3]:
```
#import get to call a get request on the site
from requests import get

#get the first page of the east bay housing prices
response = get('https://sfbay.craigslist.org/search/eby/apa?hasPic=1&availabilityMode=0'

from bs4 import BeautifulSoup
html_soup = BeautifulSoup(response.text, 'html.parser')

#get the macro-container for the housing posts
posts = html_soup.find_all('li', class_= 'result-row')
print(type(posts)) #to double check that I got a ResultSet
print(len(posts)) #to double check I got 120 (elements/page)
```

```
<class 'bs4.element.ResultSet'>
120
```

In [ ]:
```
#grab the first post
post_one = posts[0]
```

In [12]:
```
#grab the price of the first post
post_one_price = post_one.a.text
post_one_price.strip()
```

Out[12]:
```
'$2200'
```

In [13]:
```
#grab the time of the post in datetime format to save on cleaning efforts
post_one_time = post_one.find('time', class_= 'result-date')
post_one_datetime = post_one_time['datetime']
```

In [22]:
```
#title is a and that class, link is grabbing the href attribute of that variable
post_one_title = post_one.find('a', class_='result-title hdrlnk')
post_one_link = post_one_title['href']

#easy to grab the post title by taking the text element of the title variable
post_one_title_text = post_one_title.text
```

In [81]:
```
#grabs the whole segment of housing details. We will need missing value handling in the
#the text can be split, and we can use indexing to grab the elements we want. number of
#sqft is the third element

post_one_num_bedrooms = post_one.find('span', class_ = 'housing').text.split()[0]

post_one_sqft = post_one.find('span', class_ = 'housing').text.split()[2][:-3] #cleans
```

Out[81]:
```
['3br', '-']
```

In [241]:
```
#the neighborhood is grabbed by finding the span class 'result-hood' and pulling the t
post_one_hood = posts[0].find('span', class_='result-hood').text

import re
re.findall(r"[\w]+", testing)[0].title() #this takes only letters, thereby bypassing a


#post_hood = posts[100].find('span', class_= 'result-hood').text
#post_hood = re.findall(r"[\w]+", posts[100].find('span', class_= 'result-hood').text)
#post_hood

hood = posts[28].find('span', class_= 'result-hood').text.strip().split()[0:2]

" ".join(hood).title()
```

Out[241]:
```
'(North Oakland)'
```

In [252]:
```
#build out the loop
from time import sleep
import re
from random import randint #avoid throttling by not sending too many requests one afte
from warnings import warn
from time import time
from IPython.core.display import clear_output
import numpy as np

#find the total number of posts to find the limit of the pagination
results_num = html_soup.find('div', class_= 'search-legend')
results_total = int(results_num.find('span', class_='totalcount').text) #pulled the to

#each page has 119 posts so each new page is defined as follows: s=120, s=240, s=360,
pages = np.arange(0, results_total+1, 120)

iterations = 0

post_timing = []
post_hoods = []
post_title_texts = []
bedroom_counts = []
```

```
sqfts = []
post_links = []
post_prices = []

for page in pages:

    #get request
    response = get("https://sfbay.craigslist.org/search/eby/apt?"
                   + "s=" #the parameter for defining the page number
                   + str(page) #the page number in the pages array from earlier
                   + "&hasPic=1"
                   + "&availabilityMode=0")

    sleep(randint(1,5))

    #throw warning for status codes that are not 200
    if response.status_code != 200:
        warn('Request: {}; Status code: {}'.format(requests, response.status_code))

    #define the html text
    page_html = BeautifulSoup(response.text, 'html.parser')

    #define the posts
    posts = html_soup.find_all('li', class_= 'result-row')

    #extract data item-wise
    for post in posts:

        if post.find('span', class_ = 'result-hood') is not None:

            #posting date
            #grab the datetime element 0 for date and 1 for time
            post_datetime = post.find('time', class_= 'result-date')['datetime']
            post_timing.append(post_datetime)

            #neighborhoods
            post_hood = post.find('span', class_= 'result-hood').text
            post_hoods.append(post_hood)

            #title text
            post_title = post.find('a', class_='result-title hdrlnk')
            post_title_text = post_title.text
            post_title_texts.append(post_title_text)

            #post link
            post_link = post_title['href']
            post_links.append(post_link)
```

```
            #removes the \n whitespace from each side, removes the currency symbol, an
            post_price = int(post.a.text.strip().replace("$", ""))
            post_prices.append(post_price)

            if post.find('span', class_ = 'housing') is not None:

                #if the first element is accidentally square footage
                if 'ft2' in post.find('span', class_ = 'housing').text.split()[0]:

                    #make bedroom nan
                    bedroom_count = np.nan
                    bedroom_counts.append(bedroom_count)

                    #make sqft the first element
                    sqft = int(post.find('span', class_ = 'housing').text.split()[0][:
                    sqfts.append(sqft)

                #if the length of the housing details element is more than 2
                elif len(post.find('span', class_ = 'housing').text.split()) > 2:

                    #therefore element 0 will be bedroom count
                    bedroom_count = post.find('span', class_ = 'housing').text.replace
                    bedroom_counts.append(bedroom_count)

                    #and sqft will be number 3, so set these here and append
                    sqft = int(post.find('span', class_ = 'housing').text.split()[2][:
                    sqfts.append(sqft)

                #if there is num bedrooms but no sqft
                elif len(post.find('span', class_ = 'housing').text.split()) == 2:

                    #therefore element 0 will be bedroom count
                    bedroom_count = post.find('span', class_ = 'housing').text.replace
                    bedroom_counts.append(bedroom_count)

                    #and sqft will be number 3, so set these here and append
                    sqft = np.nan
                    sqfts.append(sqft)

                else:
                    bedroom_count = np.nan
                    bedroom_counts.append(bedroom_count)

                    sqft = np.nan
                    sqfts.append(sqft)

            #if none of those conditions catch, make bedroom nan, this won't be needed
            else:
```

```
                        bedroom_count = np.nan
                        bedroom_counts.append(bedroom_count)

                        sqft = np.nan
                        sqfts.append(sqft)
                    #    bedroom_counts.append(bedroom_count)

                    #    sqft = np.nan
                    #    sqfts.append(sqft)

            iterations += 1
            print("Page " + str(iterations) + " scraped successfully!")

        print("\n")

        print("Scrape complete!")
```

```
Page 1 scraped successfully!
Page 2 scraped successfully!
Page 3 scraped successfully!
Page 4 scraped successfully!
Page 5 scraped successfully!
Page 6 scraped successfully!
Page 7 scraped successfully!
Page 8 scraped successfully!
Page 9 scraped successfully!
Page 10 scraped successfully!
Page 11 scraped successfully!
Page 12 scraped successfully!
Page 13 scraped successfully!
Page 14 scraped successfully!
Page 15 scraped successfully!
Page 16 scraped successfully!
Page 17 scraped successfully!
Page 18 scraped successfully!
Page 19 scraped successfully!
Page 20 scraped successfully!
Page 21 scraped successfully!
Page 22 scraped successfully!
Page 23 scraped successfully!
Page 24 scraped successfully!
Page 25 scraped successfully!
Page 26 scraped successfully!


Scrape complete!
```

```
In [294]: import pandas as pd
```

```
eb_apts = pd.DataFrame({'posted': post_timing,
                        'neighborhood': post_hoods,
                        'post title': post_title_texts,
                        'number bedrooms': bedroom_counts,
                         'sqft': sqfts,
                         'URL': post_links,
                        'price': post_prices})
print(eb_apts.info())
eb_apts.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3120 entries, 0 to 3119
Data columns (total 7 columns):
posted             3120 non-null object
neighborhood       3120 non-null object
post title         3120 non-null object
number bedrooms    2704 non-null object
sqft               2418 non-null float64
URL                3120 non-null object
price              3120 non-null int64
dtypes: float64(1), int64(1), object(5)
memory usage: 170.7+ KB
None
```

Out[294]:

	posted	neighborhood	post title	number bedrooms	sqft
0	2019-02-03 15:55	(dublin / pleasanton / livermore)	Park like Setting! Top floor location. Tour an...	3	996.0
1	2019-02-03 15:54	(North Oakland)	"FULLY REMODELED" Large Studio, 1 Bath, 5plex	NaN	NaN
2	2019-02-03 15:53	(hayward / castro valley)	Single family, modern 4BR/4BA	4	2000.0
3	2019-02-03 15:51	(dublin / pleasanton / livermore)	Large One bedroom with a garage! 24 hour gym! ...	1	705.0
4	2019-02-03 15:51	(dublin / pleasanton / livermore)	Looking To Move Today? 1 Month Free! Come And...	1	754.0
5	2019-02-03 15:49	(Downtown Oakland)	"FULLY REMODELED" 2 Bdrm, 1 Bath + Living Room...	2	NaN
6	2019-02-03 15:48	(North Oakland)	"FULLY REMODELED" 4 Bdrm, 2 Bath + Living Room...	4	NaN
7	2019-02-03 15:48	(fremont / union city / newark)	Immidate Move in Lg 1b1b W/ in unit W/D @sofif...	1	710.0
8	2019-02-03 15:48	(hayward / castro valley)	Elegant & Pet Friendly 2x2, with Stainless App...	2	1012.0
9	2019-02-03 15:48	(North Oakland)			

```
9          "FULLY REMODELED" 2 Bdrm, 1 Bath, Duplex               2     NaN

                                                   URL  price
0  https://sfbay.craigslist.org/eby/apa/d/livermo...   2865
1  https://sfbay.craigslist.org/eby/apa/d/emeryvi...   1695
2  https://sfbay.craigslist.org/eby/apa/d/hayward...   4250
3  https://sfbay.craigslist.org/eby/apa/d/dublin-...   2459
4  https://sfbay.craigslist.org/eby/apa/d/pleasan...   2500
5  https://sfbay.craigslist.org/eby/apa/d/oakland...   2195
6  https://sfbay.craigslist.org/eby/apa/d/emeryvi...   3595
7  https://sfbay.craigslist.org/eby/apa/d/fremont...   2292
8  https://sfbay.craigslist.org/eby/apa/d/hayward...   2698
9  https://sfbay.craigslist.org/eby/apa/d/emeryvi...   2395
```

2 Data cleaning

In [304]:
```
#first things first, drop duplicate URLs because people are spammy on Craigslist.
#Let's see how many uniqe posts we really have.
eb_apts = eb_apts.drop_duplicates(subset='URL')
len(eb_apts.drop_duplicates(subset='URL'))

#make the number bedrooms to a float (since np.nan is a float too)
eb_apts['number bedrooms'] = eb_apts['number bedrooms'].apply(lambda x: float(x))

#convert datetime string into datetime object to be able to work with it
from datetime import datetime

eb_apts['posted'] = pd.to_datetime(eb_apts['posted'])

#Looking at what neighborhoods there are with eb_apts['neighborhood'].unique() allowed
#I needed to deal with in terms of cleaning those.

#remove the parenthesis from the left and right of the neighborhoods
eb_apts['neighborhood'] = eb_apts['neighborhood'].map(lambda x: x.lstrip('(').rstrip('

#titlecase them
eb_apts['neighborhood'] = eb_apts['neighborhood'].str.title()

#just take the first name of the neighborhood list, splitting on the '/' delimiter
eb_apts['neighborhood'] = eb_apts['neighborhood'].apply(lambda x: x.split('/')[0])

#fix one-offs that
eb_apts['neighborhood'].replace('Belmont, Ca', 'Belmont', inplace=True)
eb_apts['neighborhood'].replace('Hercules, Pinole, San Pablo, El Sob', 'Hercules', inp

#remove whitespaces
eb_apts['neighborhood'] = eb_apts['neighborhood'].apply(lambda x: x.strip())
```

```
#save the clean data
eb_apts.to_csv("eb_apts_1642_Jan_2_19_clean.csv", index=False)
```

3 Exploratory Data Analysis

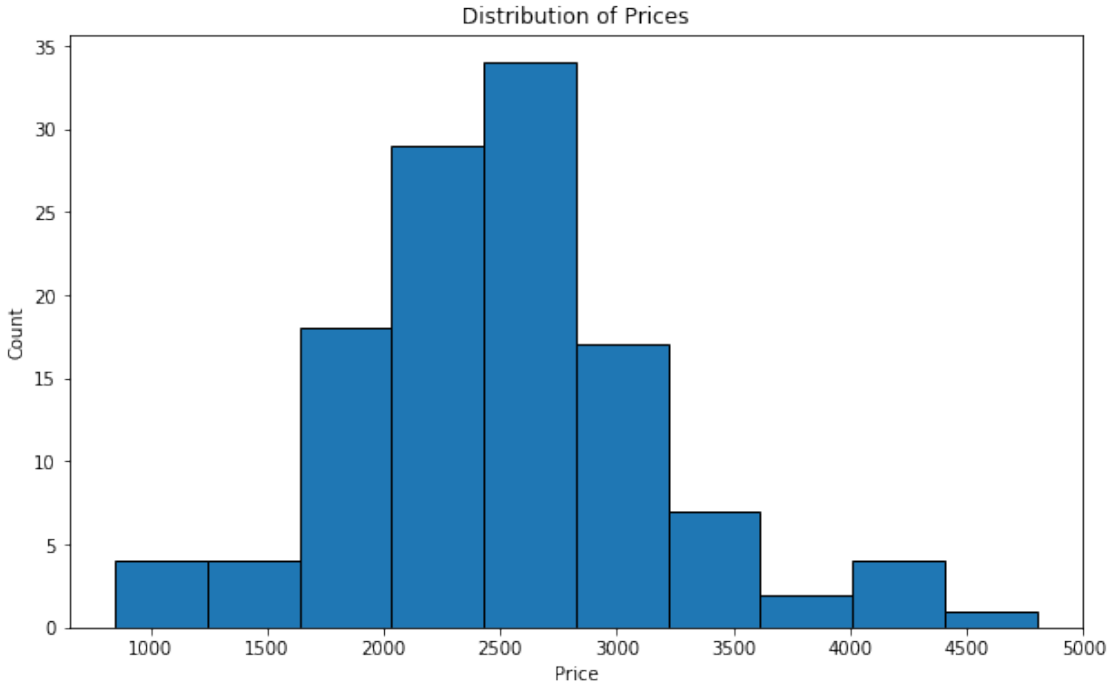
```
In [329]: eb_apts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 120 entries, 0 to 119
Data columns (total 7 columns):
posted             120 non-null datetime64[ns]
neighborhood       120 non-null object
post title         120 non-null object
number bedrooms    104 non-null float64
sqft               93 non-null float64
URL                120 non-null object
price              120 non-null int64
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 7.5+ KB
```

```
In [7]: import pandas as pd, numpy as np
        #I think Craigslist blocked me. Reload dataframe via file for now
        eb_apts = pd.read_csv("eb_apts_1642_Jan_2_19_clean.csv")
```

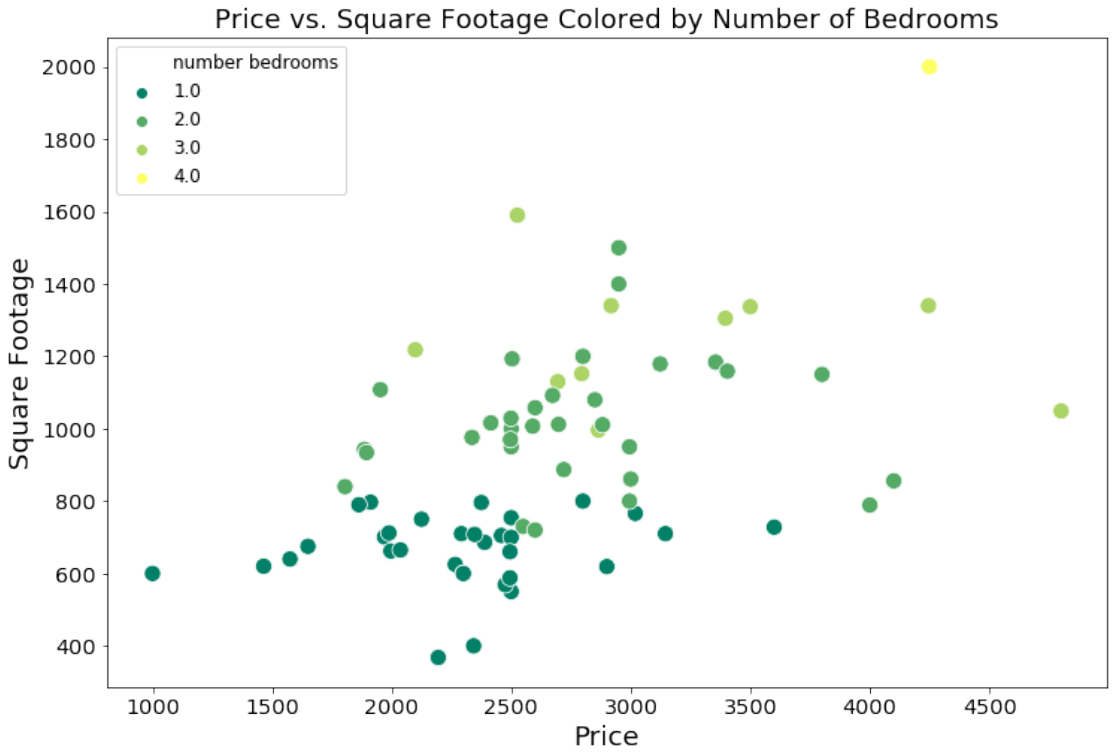
```
In [8]: #start to look at the distributions
        from matplotlib import figure
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        plt.figure(figsize=(10, 6))
        plt.hist(eb_apts['price'], edgecolor='black');
        plt.xlabel("Price")
        plt.ylabel('Count')
        plt.title("Distribution of Prices");
```



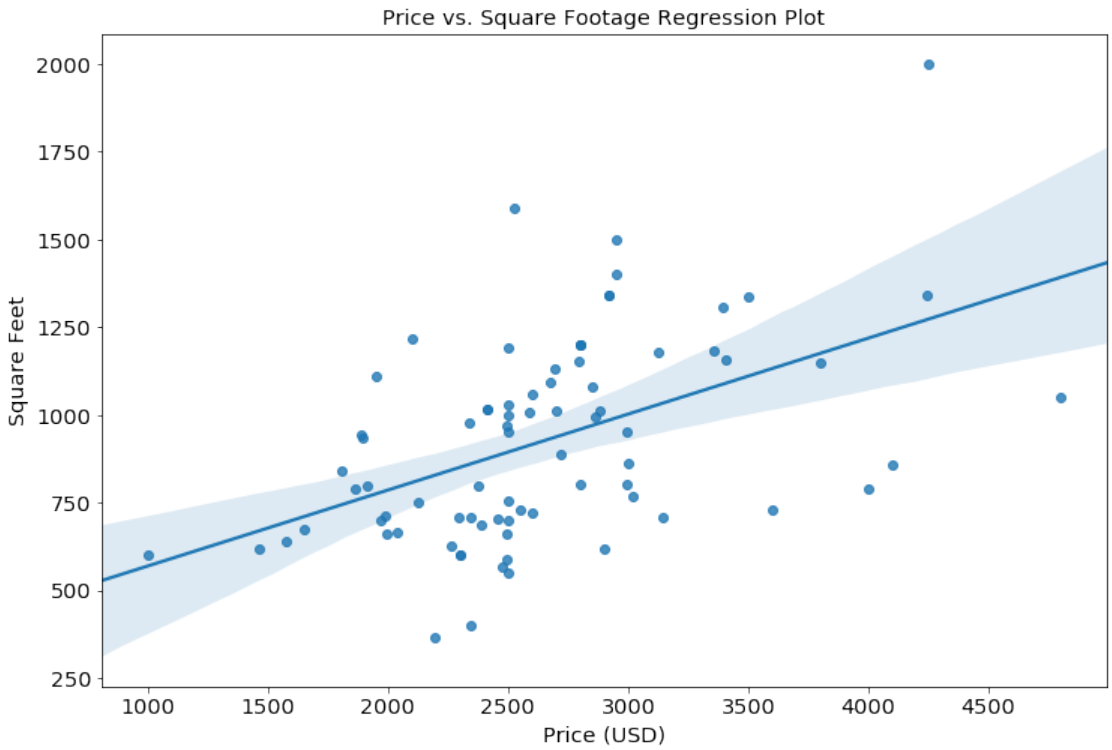

```
In [9]: import matplotlib.pylab as pylab
        params = {'legend.fontsize': 'x-large',
                  'figure.figsize': (15, 5),
                 'axes.labelsize': 'x-large',
                 'axes.titlesize':'x-large',
                 'xtick.labelsize':'x-large',
                 'ytick.labelsize':'x-large'}
        pylab.rcParams.update(params)

        plt.figure(figsize=(12, 8))
        sns.scatterplot(x='price', y='sqft', hue='number bedrooms', palette='summer', x_jitter=T
        plt.legend(fontsize=12)
        plt.xlabel("Price", fontsize=18)
        plt.ylabel("Square Footage", fontsize=18);
        plt.title("Price vs. Square Footage Colored by Number of Bedrooms", fontsize=18);
```




```
In [10]: plt.figure(figsize=(12, 8))
         sns.regplot(x='price', y='sqft', data=eb_apts.dropna());
         plt.title('Price vs. Square Footage Regression Plot');
         plt.xlabel("Price (USD)");
         plt.ylabel("Square Feet");
```

```
C:\Users\riley\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [11]: eb_apts.corr()
```

Out[11]:

	number bedrooms	sqft	price
number bedrooms	1.000000	0.849161	0.552040
sqft	0.849161	1.000000	0.472154
price	0.552040	0.472154	1.000000

```
In [12]: pd.options.display.max_colwidth = 100 #display full URL
         cheap_berkeley = eb_apts[(eb_apts['price'] < 3000) & (eb_apts['neighborhood'] == 'berke
                                  (eb_apts['neighborhood'] == 'oakland piedmont  montclair') |
                                  (eb_apts['neighborhood'] == 'berkeley north hills')]
         cheap_berkeley.sort_values(by='price', ascending=True)
```

```
Out[12]: Empty DataFrame
         Columns: [posted, neighborhood, post title, number bedrooms, sqft, URL, price]
         Index: []
```

```
In [13]: #group by neighborhood
         eb_apts.groupby('neighborhood').mean()
```

Out[13]:

neighborhood	number bedrooms	sqft	price
Alameda	2.333333	970.000000	2811.250000
Belmont	NaN	1830.000000	1000.000000

```
Berkeley                    1.857143   690.428571  2943.222222
Berkeley North              1.000000   800.000000  2800.000000
Concord                     2.166667   914.000000  2320.571429
Danville                    1.333333   798.333333  2285.000000
Downtown Oakland            1.666667          NaN  1746.666667
Dublin                      1.733333   947.466667  2642.200000
East Oakland                2.000000          NaN  2195.000000
Emeryville                  2.333333  1100.500000  3248.333333
Fairfield                   2.000000  1019.000000  2078.000000
Fremont                     1.500000   964.333333  2824.000000
Hayward                     2.200000  1123.125000  2641.266667
Hercules                    1.500000  1108.000000  1651.500000
North Oakland               3.000000          NaN  2561.666667
Oakland Downtown            1.333333   600.666667  2313.333333
Oakland Hills                    NaN   550.000000  2300.000000
Oakland Lake Merritt        1.428571   728.142857  2659.285714
Oakland North               1.500000   685.333333  2691.666667
Oakland Rockridge           1.500000   850.000000  3150.000000
Pittsburg                        NaN          NaN   950.000000
Richmond                    2.000000   970.000000  2497.000000
San Leandro                 2.666667          NaN  2733.333333
Vallejo                     1.666667   819.333333  1991.333333
Walnut Creek                1.000000   705.333333  2481.333333
West Oakland                3.333333          NaN  2795.000000
```

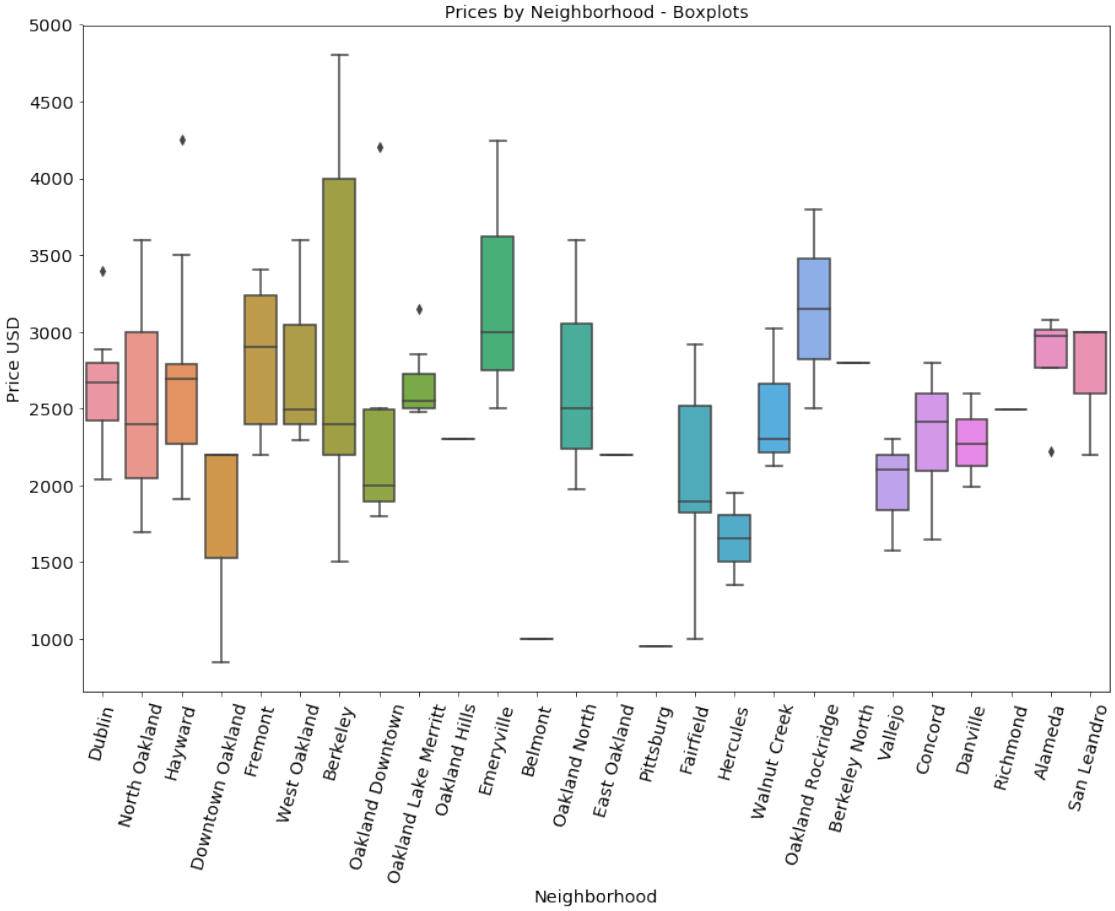
In [14]:
```
#sort price to find cheapest
eb_apts.groupby('neighborhood').mean()['price'].sort_values()
```

Out[14]:
```
neighborhood
Pittsburg                 950.000000
Belmont                  1000.000000
Hercules                 1651.500000
Downtown Oakland         1746.666667
Vallejo                  1991.333333
Fairfield                2078.000000
East Oakland             2195.000000
Danville                 2285.000000
Oakland Hills            2300.000000
Oakland Downtown         2313.333333
Concord                  2320.571429
Walnut Creek             2481.333333
Richmond                 2497.000000
North Oakland            2561.666667
Hayward                  2641.266667
Dublin                   2642.200000
Oakland Lake Merritt     2659.285714
Oakland North            2691.666667
San Leandro              2733.333333
```

```
West Oakland            2795.000000
Berkeley North          2800.000000
Alameda                 2811.250000
Fremont                 2824.000000
Berkeley                2943.222222
Oakland Rockridge       3150.000000
Emeryville              3248.333333
Name: price, dtype: float64
```

In [28]:
```
plt.figure(figsize=(15,10))
params = {'legend.fontsize': 'x-large',
          'figure.figsize': (15, 5),
         'axes.labelsize': 'x-large',
         'axes.titlesize':'x-large',
         'xtick.labelsize':'x-large',
         'ytick.labelsize':'x-large'}
pylab.rcParams.update(params)

sns.boxplot(x='neighborhood', y='price', data=eb_apts)
plt.xlabel("Neighborhood");
plt.xticks(rotation=75)
plt.ylabel("Price USD");
plt.title("Prices by Neighborhood - Boxplots");
```


Prices by Neighborhood - Boxplots
Price USD
5000
4500
4000
3500
3000
2500
2000
1500
1000
Dublin
North Oakland
Hayward
Downtown Oakland
Fremont
West Oakland
Berkeley
Oakland Downtown
Oakland Lake Merritt
Oakland Hills
Emeryville
Belmont
Oakland North
East Oakland
Pittsburg
Fairfield
Hercules
Walnut Creek
Oakland Rockridge
Berkeley North
Vallejo
Concord
Danville
Richmond
Alameda
San Leandro
Neighborhood