Springboard #4

Summary

#1 Importing the info

```
:  # Let's import the pandas, numpy libr
   import pandas as pd
   import numpy as np

   # Load the pyplot collection of functi
   import matplotlib.pyplot as plt
```

First import the necessary tools to work w/

#2 Cleaning

	City of London	Barking & Dagenham	Barnet	Bexley	Brent
	E09000001	E09000002	E09000003	E09000004	E09000005
Jan-95	91,449	50,460	93,285	64,958	71,307
Feb-95	82,203	51,086	93,199	64,788	72,022
Mar-95	79,121	51,269	92,248	64,367	72,016
Apr-95	77,101	53,134	90,763	64,278	72,966
May-95	84,409	53,042	90,258	63,997	73,704
Jun-95	94,901	53,700	90,107	64,252	74,310
Jul-95	110,128	52,113	91,441	63,723	74,127
Aug-95	112,329	52,232	92,361	64,433	73,547
Sep-95	104,473	51,472	93,273	64,510	73,790
Oct-95	108,038	51,514	92,567	64,530	73,264
Nov-95	117,636	50,849	90,883	63,846	72,782
Dec-95	127,232	50,945	91,134	63,817	72,524
Jan-96	108,999	50,828	91,111	63,996	72,806
Feb-96	93,357	51,441	92,430	64,504	73,084
Mar-96	93,707	51,907	91,410	64,788	72,780
Apr-96	120,543	51,724	92,394	65,286	72,370
May-96	112,050	51,736	91,059	65,081	72,995

The information came from the web.
It needed some work

To begin the Borough's needed to become the index
While the dates the columns

#2 Cleaning

	index	0	1	2	3	4	
0	Unnamed: 0	NaT	1995-01-01 00:00:00	1995-02-01 00:00:00	1995-03-01 00:00:00	1995-04-01 00:00:00	00
1	City of London	E09000001	91449	82202.8	79120.7	77101.2	8
2	Barking & Dagenham	E09000002	50460.2	51085.8	51269	53133.5	5
3	Barnet	E09000003	93284.5	93190.2	92247.5	90762.9	
4	Bexley	E09000004	64958.1	64787.9	64367.5	64277.7	6

5 rows × 211 columns

When moved, the first Row became what you wanted the columns to be.

Thus, the below was required to bring it up top & clear it for the Borough's.

```
properties_adjusted = properties.T
properties_adjusted = properties_adjusted.reset_index()
properties_adjusted.columns = properties_adjusted.iloc[0]
properties_adjusted = properties_adjusted.drop(0)
```

#2 Cleaning

	Unnamed: 0	NaN	1995-01-01 00:00:00	1995-02-01 00:00:00	1995-03-01 00:00:00	1995-04-01 00:00:00	1995-05-01 00:00:00	1995-06-01 00:00:00	1
1	City of London	E09000001	91449	82202.8	79120.7	77101.2	84409.1	94900.5	
2	Barking & Dagenham	E09000002	50460.2	51085					
3	Barnet	E09000003	93284.5	93190.2	92247.5	90762.9	90258	90107.2	
4	Bexley	E09000004	64958.1	64787.9	64367.5	64277.7	63997.1	64252.3	
5	Brent	E09000005	71306.6	72022.3	72015.8	72965.6	73704	74310.5	

Once the infrastructure is set up, need to clean up some names…

```
properties_adjusted = properties_adjusted.rename(columns = { 'Unnamed: 0':'London_Borough', pd.NaT : 'ID' })
properties_adjusted.columns
properties_adjusted.head()
```

	London_Borough	ID	1995-01-01 00:00:00	1995-02-01 00:00:00	1
1	City of London	E09000001	91449	82202.8	
2	Barking & Dagenham	E09000002	50460.2	51085.8	
3	Barnet	E09000003	93284.5	93190.2	
4	Bexley	E09000004	64958.1	64787.9	
5	Brent	E09000005	71306.6	72022.3	

#2 Cleaning

```
cleaned = pd.melt(properties_adjusted, id_vars = [ 'London_Borough', 'ID' ])
cleaned = cleaned.rename( columns = { 0 : 'Month', 'value' : 'Average_price' } )
cleaned.head()
```

	London_Borough	ID	Month	Average_price
0	City of London	E09000001	1995-01-01	91449
1	Barking & Dagenham	E09000002	1995-01-01	50460.2
2	Barnet	E09000003	1995-01-01	93284.5
3	Bexley	E09000004	1995-01-01	64958.1
4	Brent	E09000005	1995-01-01	71306.6

You then ‘melt’ all the dates into one column. This is similar to what you must do in Tableau Prep.

Basically, each month is arranged & shows all the Boroughs before moving to the next month

#2 Cleaning

```
cleaned['Average_price'] = pd.to_numeric(cleaned['Average_price'])
cleaned.dtypes
```

```
London_Borough            object
ID                        object
Month             datetime64[ns]
Average_price            float64
dtype: object
```

```
n1 = cleaned[cleaned['Average_price'].isna()]
n2 = cleaned.dropna()
nonBoroughs = ['Inner London', 'Outer London',
               'NORTH EAST', 'NORTH WEST', 'YORKS & THE HUMBER',
               'EAST MIDLANDS', 'WEST MIDLANDS',
              'EAST OF ENGLAND', 'LONDON', 'SOUTH EAST',
              'SOUTH WEST', 'England']
n2[n2.London_Borough.isin(nonBoroughs)]
n2[~n2.London_Borough.isin(nonBoroughs)]
n2 = n2[~n2.London_Borough.isin(nonBoroughs)]
df = n2
df.head()
```

	London_Borough	ID	Month	Average_price
0	City of London	E09000001	1995-01-01	91448.98487
1	Barking & Dagenham	E09000002	1995-01-01	50460.22660
2	Barnet	E09000003	1995-01-01	93284.51832
3	Bexley	E09000004	1995-01-01	64958.09036
4	Brent	E09000005	1995-01-01	71306.56698

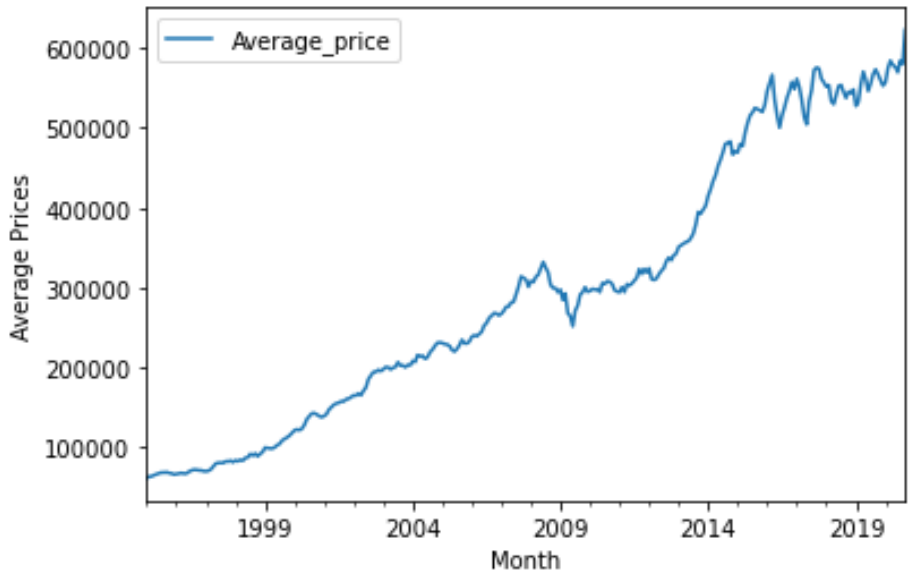
You then notice 2 things:

1. **The average price column is NOT a # so you need to change it to a #**
2. **There appears to be a noticeable differential between the # of rows for each column alluding to the idea that there are null values**
3. **We decided to just remove the null values.**

#2 Cleaning

```
hackney_prices = df[df['London_Borough'] == 'Hackney']
hack = hackney_prices.plot(kind='line', x='Month', y='Average_price')
hack.set_ylabel('Average Prices')
```

```
Text(0, 0.5, 'Average Prices')
```



To double check to see if it's okay to move to the next round, one of the borough's was chosen & shown.

#2 Cleaning

```
df['Year'] = df['Month'].apply(lambda t: t.year)
df.tail()
```

	London_Borough	ID	Month	Average_price	Year
14812	Sutton	E09000029	2020-09-01	386214.0222	2020
14813	Tower Hamlets	E09000030	2020-09-01	437574.0254	2020
14814	Waltham Forest	E09000031	2020-09-01	461726.9731	2020
14815	Wandsworth	E09000032	2020-09-01	615802.8152	2020
14816	Westminster	E09000033	2020-09-01	925003.9220	2020

The previous worked & thus for future modeling, Annual Data was added on top of the Monthly data.

#2 Cleaning

```
dfg = df.groupby(by=['London_Borough', 'Year']).mean()
dfg.sample(10)
```

:

		Average_price
London_Borough	Year	
Merton	2010	299999.358992
Brent	2015	440951.665383
Havering	2008	231473.737725
Hounslow	2000	136624.857775
Redbridge	1999	102940.450633
Croydon	2009	203706.368500
Redbridge	2014	304162.470942
Haringey	2014	435807.737383
Southwark	2009	261151.042333
Greenwich	2007	228861.968033

```
dfg = dfg.reset_index()
dfg.head()
```

:

	London_Borough	Year	Average_price
0	Barking & Dagenham	1995	51817.969390
1	Barking & Dagenham	1996	51718.192690
2	Barking & Dagenham	1997	55974.262309
3	Barking & Dagenham	1998	60285.821083
4	Barking & Dagenham	1999	65320.934441

Since we want to look at average prices per year, the Borough's were grouped by their year(s) & indexed accordingly.

#3 Modeling

```
def price_ratio(d):
    y1998 = float(d['Average_price'][d['Year']==1998])
    y2018 = float(d['Average_price'][d['Year']==2018])
    ratio = [ y2018 / y1998 ]
    return ratio
```

```
final = {}
for bor in dfg['London_Borough'].unique():
    borough = dfg[dfg['London_Borough'] == bor]
    final[bor] = price_ratio(borough)
```

Since we want to see the price appreciation between 1998 & 2018, we built a function (above) that a Loop (below) could be sent through.

We also built an empty dictionary (Final) to receive what the Loop produces.

#3 Modeling

```
df_ratios = pd.DataFrame(final)
```

```
df_ratios_T = df_ratios.T
df_ratios = df_ratios_T.reset_index()
```

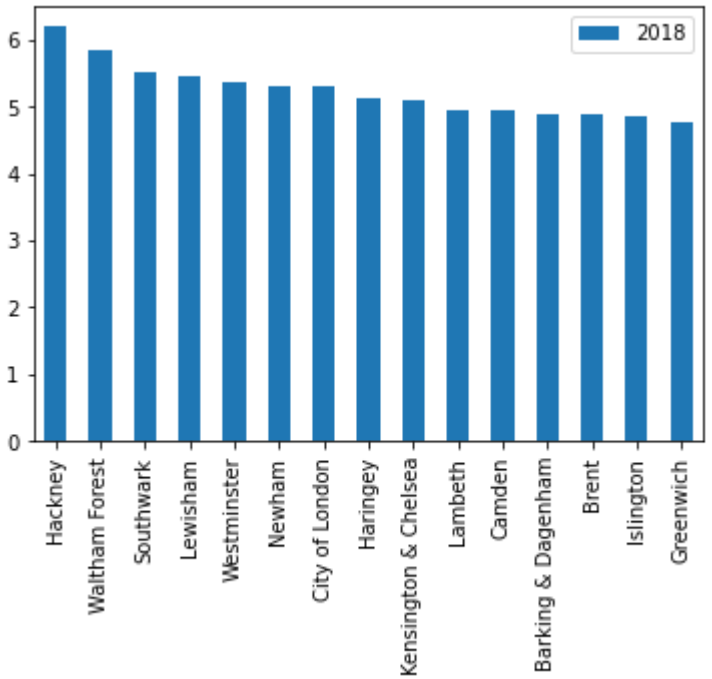
```
df_ratios.rename(columns={'index':'Borough', 0:'2018'}, inplace=True)
df_ratios.head()
```

	Borough	2018
0	Barking & Dagenham	4.896619
1	Barnet	4.358196
2	Bexley	4.248977
3	Brent	4.894554
4	Bromley	4.094785

That information through the Loop was sent to a new variable (df_ratios) which was then pivoted (similar to what was previously done on Slide 4) to a clean set that we can work with.

#3 Modeling

```
top20 = df_ratios.sort_values(by='2018', ascending=False).head(20)
top20_chart = top20[['Borough','2018']].plot(kind='bar')
top20_chart.set_xticklabels(top20.Borough)
```



Now that we can work with it, we establish the infrastructure to see what we originally wanted to see (Top 20 Boroughs by price appreciation).

We made sure to tell the variable we want to see the sorted rows in descending order in a Box Plot.

#4 Conclusion

Property prices in the top 20 boroughs in London witnessed a ~500% increase in price between 1998 & 2018 representing appreciation of ~8.46% per annum. The greatest appreciation was seen in Hackney & Watham Forest. I have never been to London & thus I am, at this time, not in a position to comment as to why these saw the greatest appreciation.