Python 3

For this bootcamp we'll be using a few data visualization modules to plot data using Python.

In this notebook we will:

1. Import required modules and datasets
2. Manipulate the data using Pandas
3. Visualize the data

In [33]:
```
#Remove warnings from our outputs
import warnings
warnings.filterwarnings("ignore")
```

Matplotlib

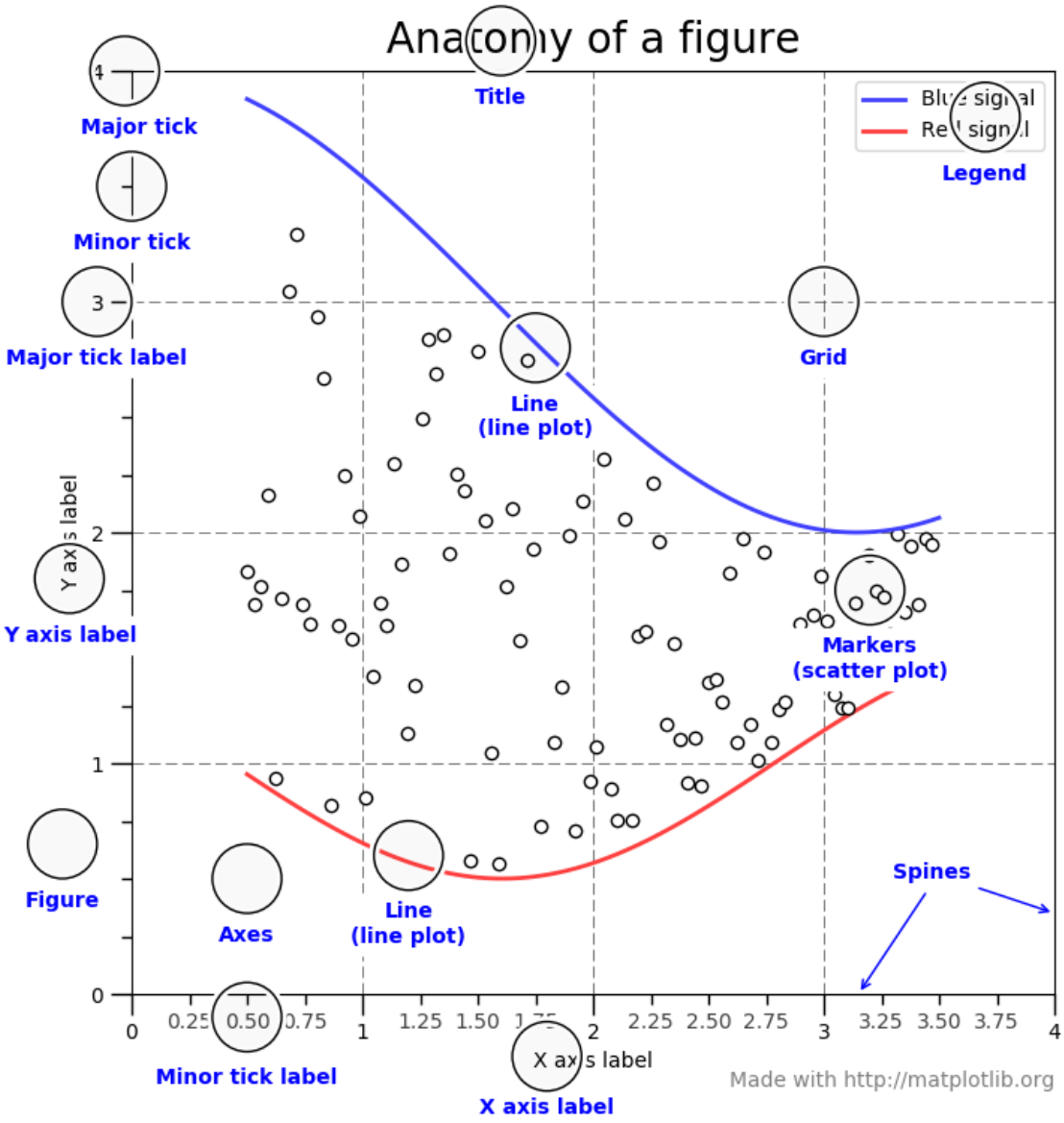
"Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python."

Matplotlib is one of the most popular libraries used to create data vizualizations in Python. It uses an object-oriented API (classes) which we've already worked with when using Pandas

Below is a breakdown of some of the key elements that go into a matplotlib figure

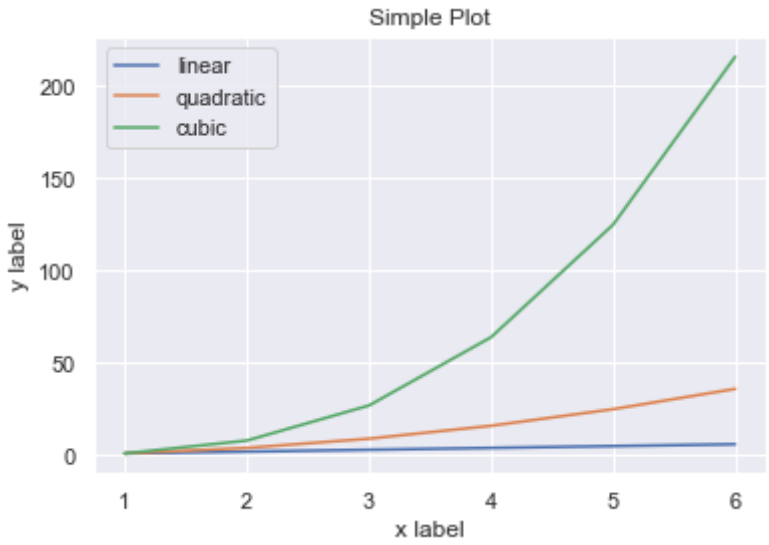
Two main concepts to understand

- A figure is the whole figure and can contain any number of axes (usually at least 1)
- Axes are the "plot" that will contain your title, legend, etc.


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Y axis label
Figure
Axes
Minor tick label
X axis label
Line
(line plot)
Grid
Markers
(scatter plot)
Spines
Legend
Made with http://matplotlib.org

In [34]:
```
import matplotlib.pyplot as plt
import numpy as np

x  = [1,2,3,4,5,6]

data = np.array(x)

# https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutoria
# Create a figure and an axes.
fig, ax = plt.subplots()

# Plot some data on the axes.
ax.plot(data, data, label='linear')

# Plot more data on the axes...
ax.plot(data, data**2, label='quadratic')

# ... and some more.
ax.plot(data, data**3, label='cubic')

# Add an x-label to the axes.
ax.set_xlabel('x label')

# Add a y-label to the axes.
ax.set_ylabel('y label')

# Add a title to the axes.
ax.set_title("Simple Plot")

# Add a legend.
ax.legend()
```

Out[34]:
```
<matplotlib.legend.Legend at 0x2957edeb940>
```



In [35]:
```
#Save our plot as an image
plt.savefig('line_plot.png')
```

```
<Figure size 432x288 with 0 Axes>
```

Pandas Plotting

Pandas offers a easy way to access Matplotlib to plot the data inside of a DataFrame.

We will go over a few ways to plot some stock data.

In [36]:
```
#Import Pandas
import pandas as pd

#A few configurations
pd.plotting.register_matplotlib_converters()
%matplotlib inline

print("Setup Complete")
```

```
Setup Complete
```

In [37]:
```
# Stock data, from 2013 to 2018
stock_df = pd.read_csv('data/all_stocks_5yr.csv', index_col="date", parse_d
```

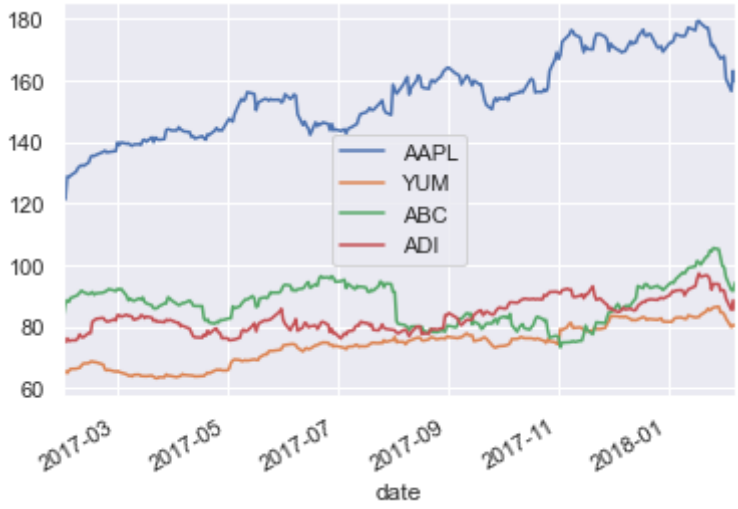
In [38]:
```
#Take a look at the data
stock_df.head()
```

Out[38]:

date	AAPL	YUM	ABC	ADI
2017-01-30	121.63	65.68	83.62	76.28
2017-01-31	121.35	65.53	87.28	74.94
2017-02-01	128.75	64.87	88.61	76.17
2017-02-02	128.53	65.67	88.05	75.23
2017-02-03	129.08	66.23	89.28	75.52

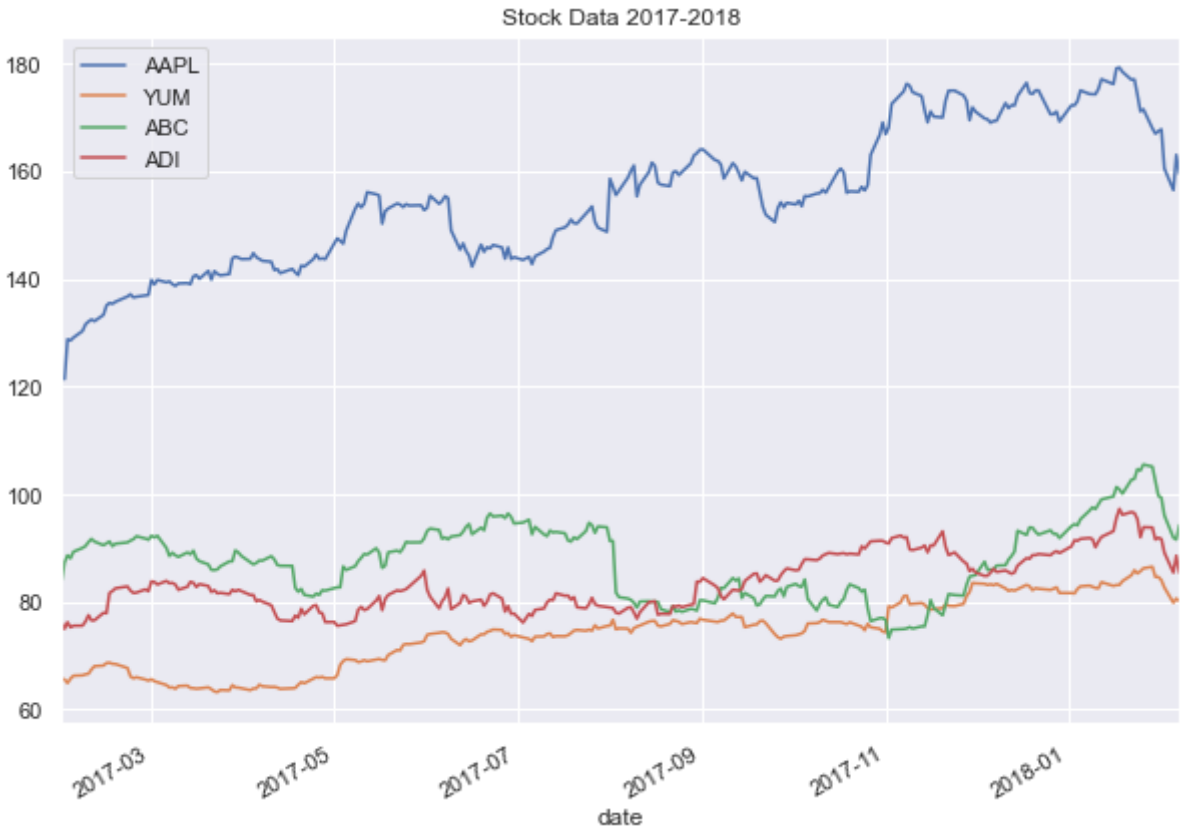
In [39]:
```
#Plotting data as easy as calling the plot() function
stock_df.plot(kind='line')
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x2957ed82eb8>

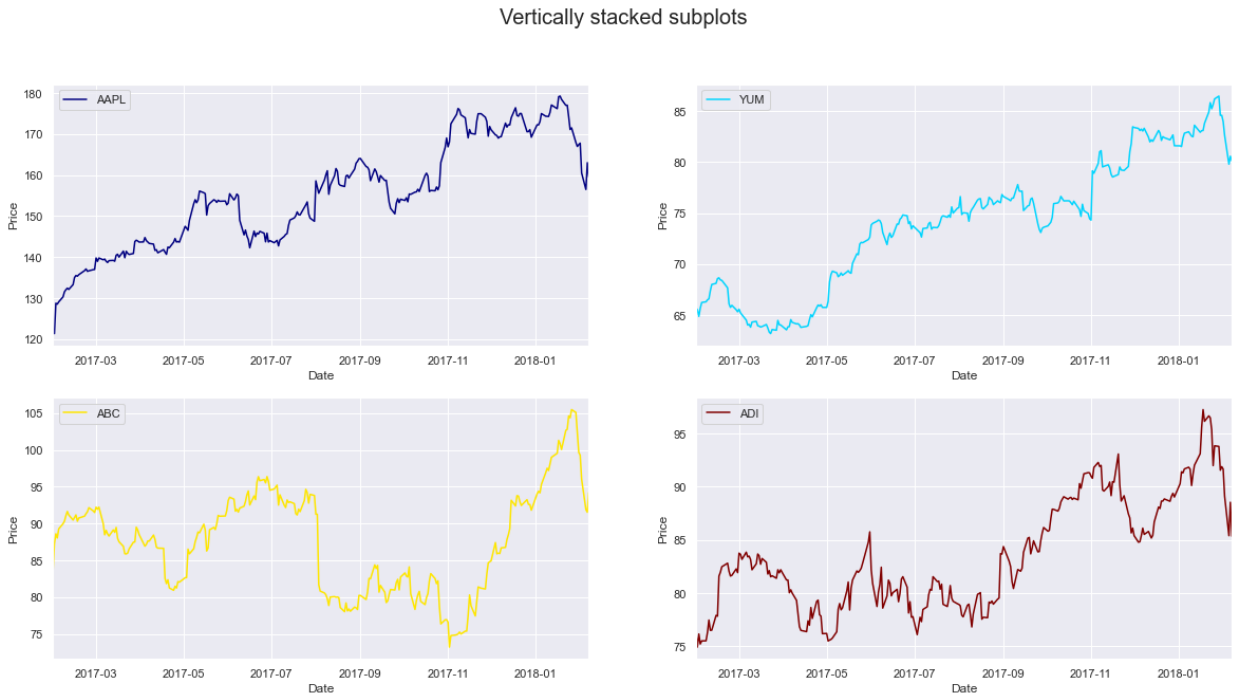


In [40]:
```
#Plotting data as easy as calling the plot() function
stock_df.plot(kind='line', figsize=(10,7), title='Stock Data 2017-2018')
```

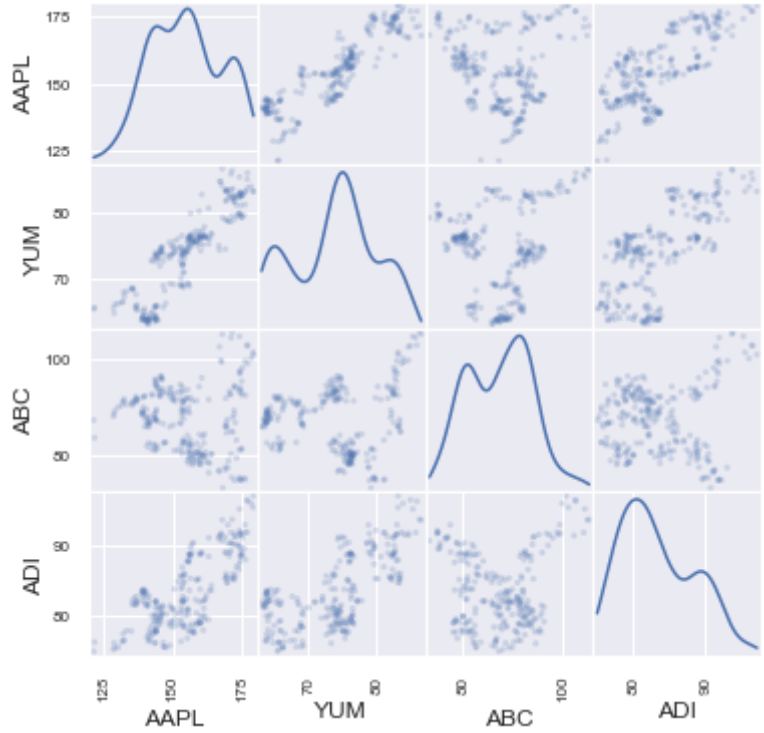
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x29510342a20>



In [41]:
```
#Define a MPL figure and axes to give us more control of our visual
fig, axs = plt.subplots(2, 2, figsize = (20,10))
fig.suptitle('Vertically stacked subplots', fontsize=20)

#Must specify the axes we want to plot onto, and can specify addicitonal st
stock_df.plot(kind='line', subplots=True,  colormap='jet', ax = axs)

#Use a FOR loop to add on the X and Y labels
for ax in axs.flat:
    ax.set(xlabel='Date', ylabel='Price')
```



In [42]:
```
from pandas.plotting import scatter_matrix

scatter_matrix(stock_df, alpha=0.2, figsize=(6, 6), diagonal='kde')
plt.show()
```



Barchart

For the next portion of the bootcamp, we're going to be using Airbnb data.

We'll be going over some of the other kinds of plots we can create directly from a Pandas DataFrame

In [43]:
```
#https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
#Import new dataset from the data/AB_NYC_2019.csv file into variable ab_df
ab_df = pd.read_csv('data/AB_NYC_2019.csv')
```

In [44]:
```
ab_df.head()
```

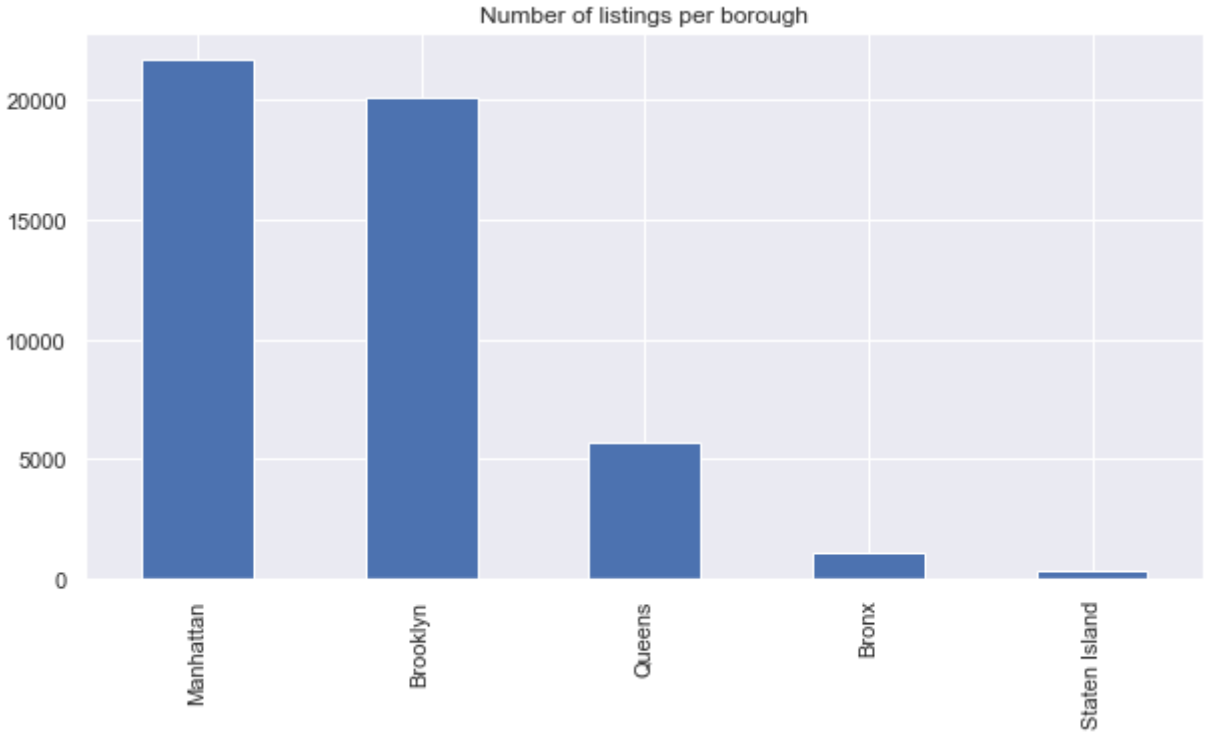
Out[44]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	lon
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73

In [45]:
```
#Calculate the number of listings per borough (neighbourhood group)
freq = ab_df['neighbourhood_group'].value_counts()
```

In [46]:
```
#Plot this data as a bar chart
plt.figure(figsize=(10,5))

freq.plot(kind='bar', title='Number of listings per borough')
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29511398208>
```



In [47]:
```
#Create a pivot so we can visualize the data
ab_pivot = ab_df.pivot_table(values='minimum_nights', index="neighbourhood_
                             columns = 'room_type', aggfunc=np.sum)
```

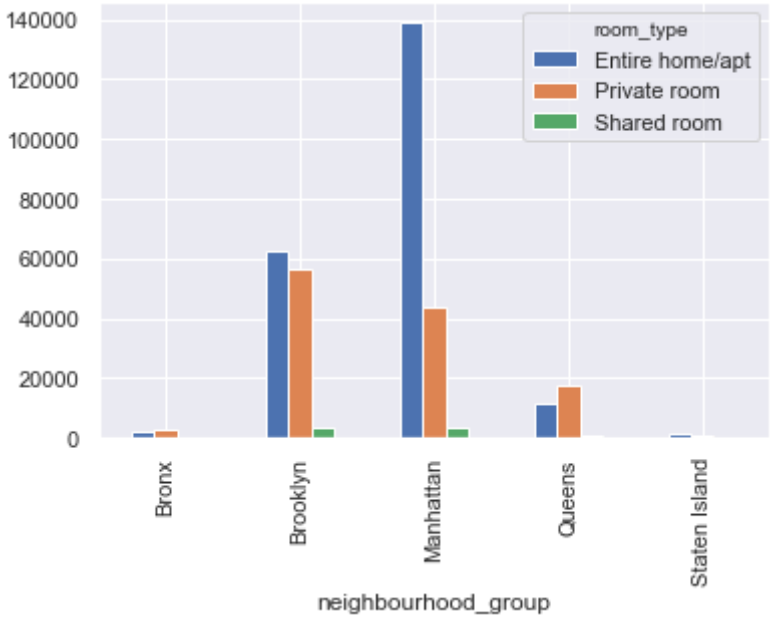
In [48]:
```
ab_pivot.head()
```

Out[48]:

room_type neighbourhood_group	Entire home/apt	Private room	Shared room
Bronx	2258	2516	202
Brooklyn	62433	56126	3202
Manhattan	139108	43477	3248
Queens	11255	17265	838
Staten Island	1098	683	21

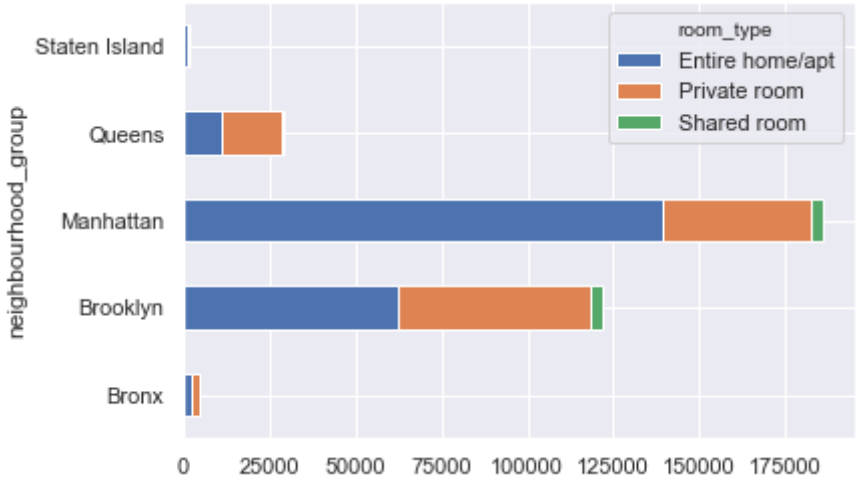
In [49]:
```
#Create a basic bar chart of the pivot table
ab_pivot.plot(kind='bar')
```

Out[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x295113e7c88>
```

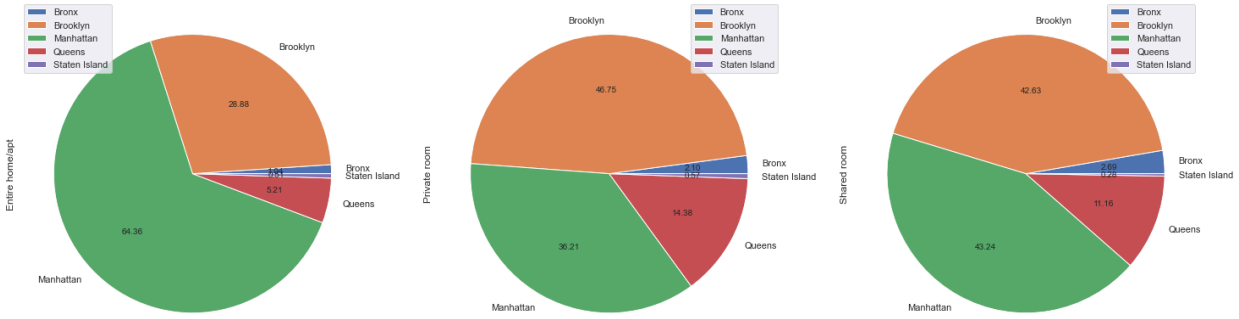


In [50]:
```
#We can stack the bar chart and change the orientation to horizontal
ab_pivot.plot(kind='barh', stacked=True)
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x295113e7b00>



In [51]:
```
#Pie charts and configurations
ab_pivot.plot(kind='pie', subplots=True, figsize=(25, 10),  autopct='%.2f')
plt.show()
```



Seaborn

Seaborn is a Python data visualization library based on matplotlib.

It provides a high-level interface for drawing attractive and informative statistical graphics.

In [52]:
```
import seaborn as sns

#Load tip data and
tips = sns.load_dataset("tips")
```

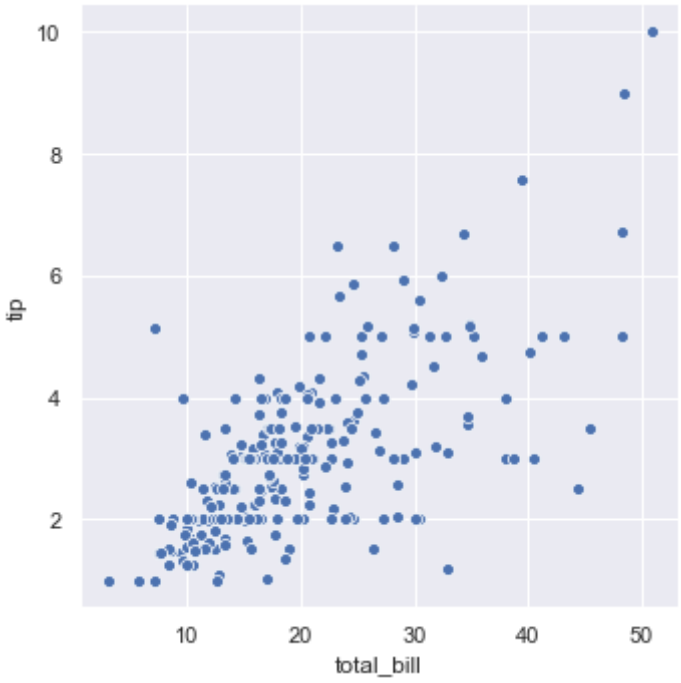
In [53]:
```
tips.head()
```

Out[53]:

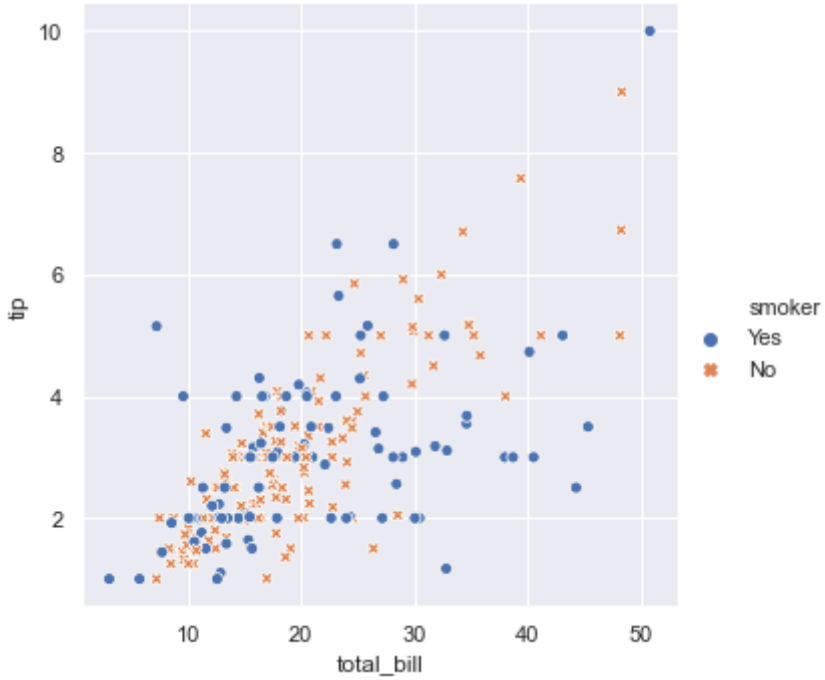
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [54]:
```
#Assign a style
sns.set(style="darkgrid")

# Set the width and height of the figure
plt.figure(figsize=(16,6))

sns.relplot(x='total_bill', y='tip', data=tips)
```

Out[54]:
```
<seaborn.axisgrid.FacetGrid at 0x29512f2dcf8>

<Figure size 1152x432 with 0 Axes>
```

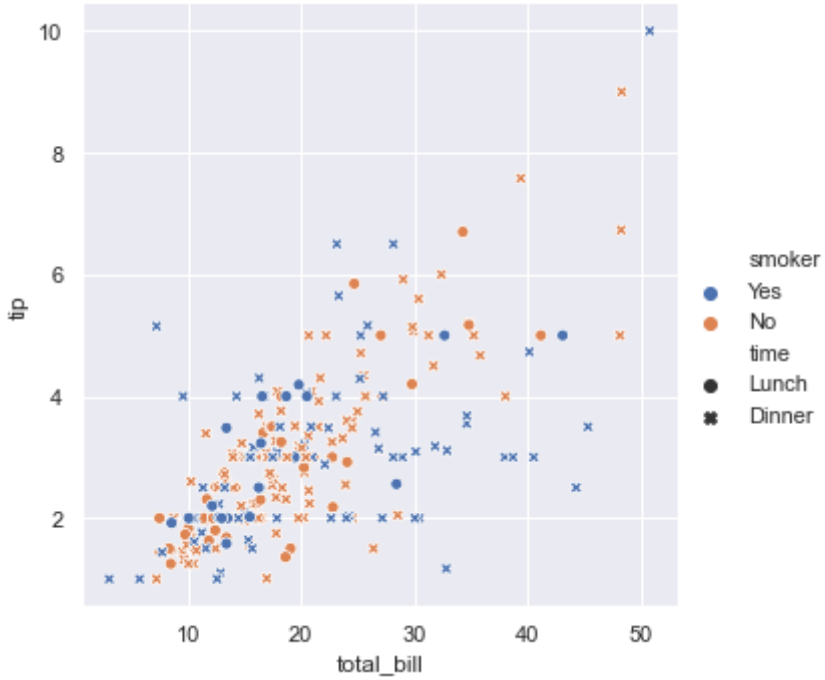


In [55]:
```
# We can add a third dimension with color and style
sns.relplot(x="total_bill", y="tip", hue="smoker", style="smoker",
            data=tips);
```



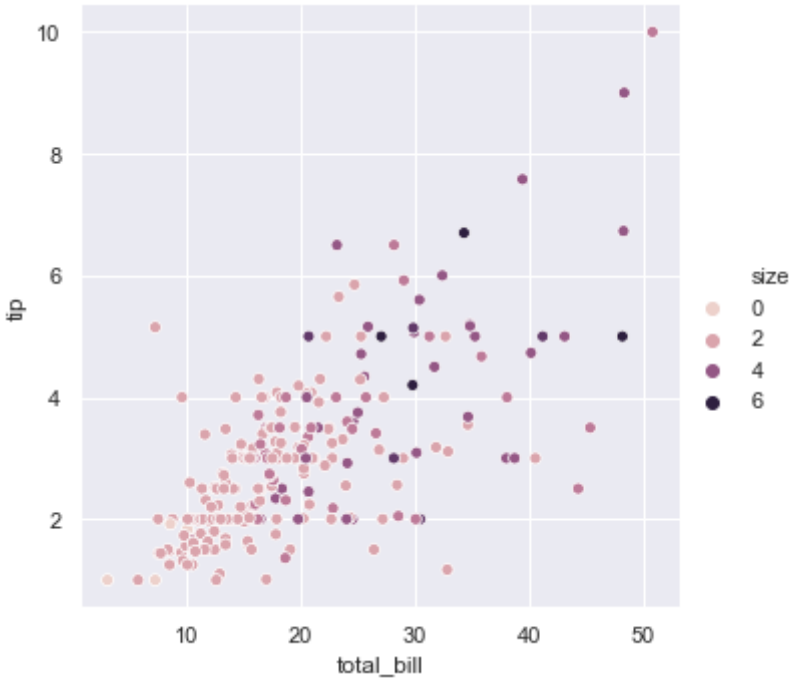
In [56]:
```
#Add a fourth dimension using different variables for hue and style
sns.relplot(x="total_bill", y="tip", hue="smoker", style="time", data=tips)
```

Out[56]: <seaborn.axisgrid.FacetGrid at 0x2957c9863c8>

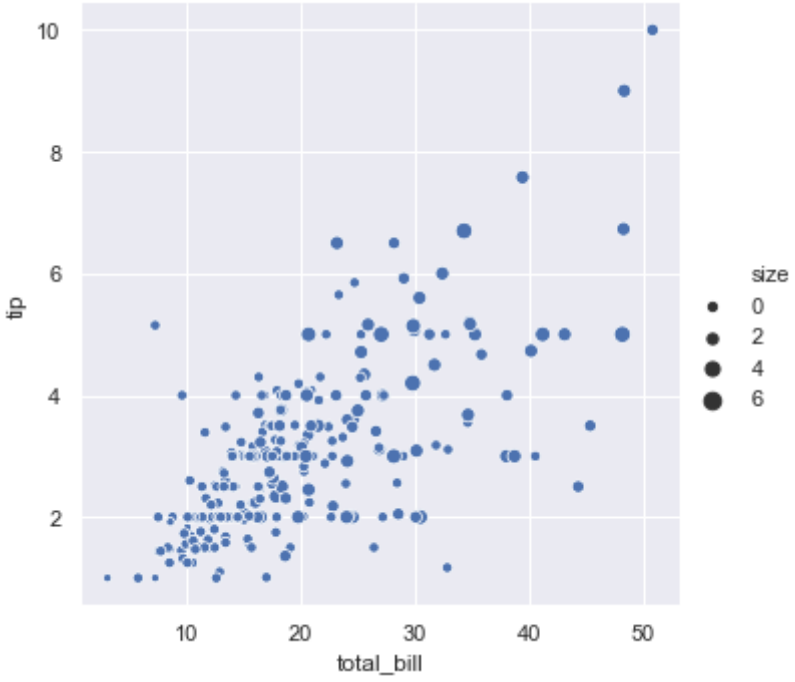


In [57]:
```
#Replot using size variable for hue
sns.relplot(x="total_bill", y="tip", hue="size", data=tips)
```

Out[57]: <seaborn.axisgrid.FacetGrid at 0x29512ae5b00>

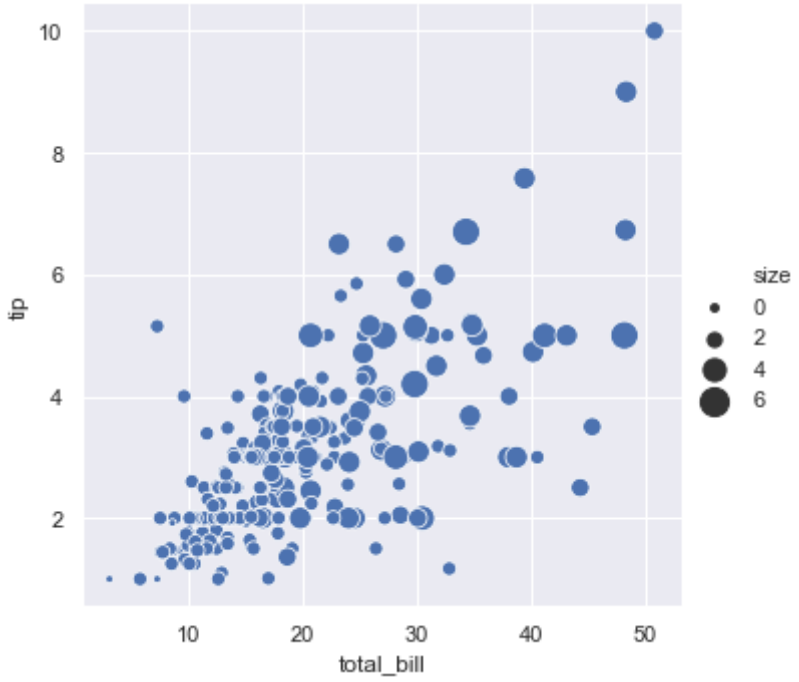


In [58]:
```
#The size parameter allows us to change the size of data points using varia
sns.relplot(x="total_bill", y="tip", size="size", data=tips);
```



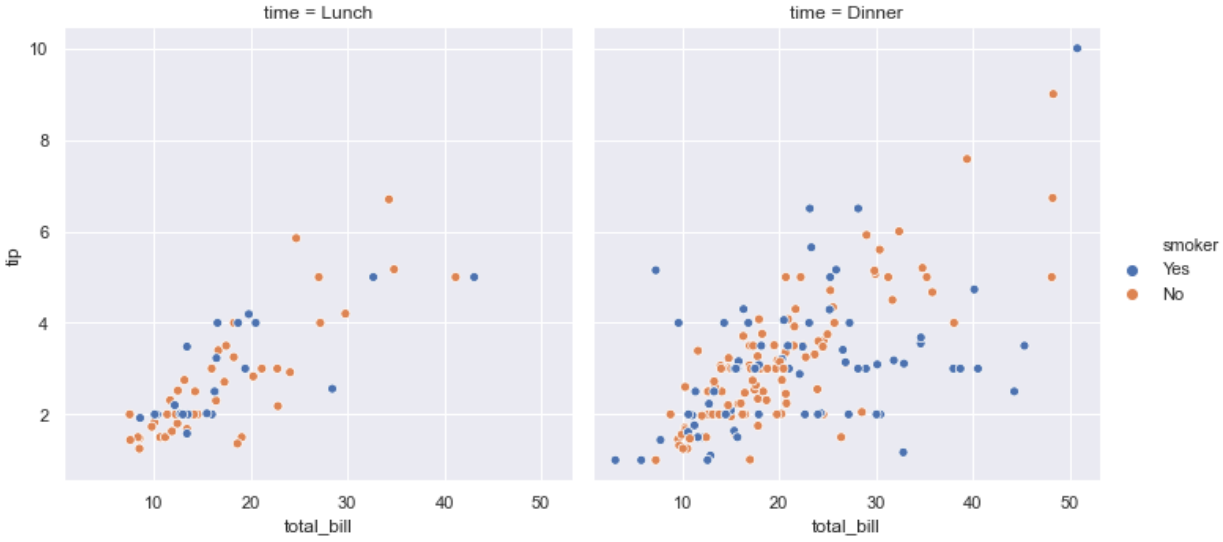
In [59]:
```
#The sizes parameter determines the scale of the data points
sns.relplot(x="total_bill", y="tip", size="size", sizes=(15, 200), data=tip
```

Out[59]: <seaborn.axisgrid.FacetGrid at 0x2957c8782e8>



In [60]:
```
#The col parameter creates subplots along the provided variable
sns.relplot(x="total_bill", y="tip", hue="smoker",
            col="time", data=tips)
```

Out[60]: <seaborn.axisgrid.FacetGrid at 0x2957c8dd320>



In [61]:
```
stock_df[['AAPL', 'ABC']].head()
```

Out[61]:

date	AAPL	ABC
2017-01-30	121.63	83.62
2017-01-31	121.35	87.28
2017-02-01	128.75	88.61
2017-02-02	128.53	88.05
2017-02-03	129.08	89.28

In [62]:
```
#Create a pivot table of the tips data
hm = tips.pivot_table(index='day',columns='size',values='tip')
```

In [63]:
```
hm.head()
```

Out[63]:

day	size 1	2	3	4	5	6
Thur	1.83	2.442500	2.692500	4.218000	5.000000	5.3
Fri	1.92	2.644375	3.000000	4.730000	NaN	NaN
Sat	1.00	2.517547	3.797778	4.123846	3.000000	NaN
Sun	NaN	2.816923	3.120667	4.087778	4.046667	5.0

In [64]:
```
#An effective way to plot our pivoted data is with a heatmap
sns.heatmap(hm)
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x2957c8dd0b8>