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 [2]:
```
#Remove warnings from our outputs
import warnings
warnings.filterwarnings("ignore")
```

Matplotlib

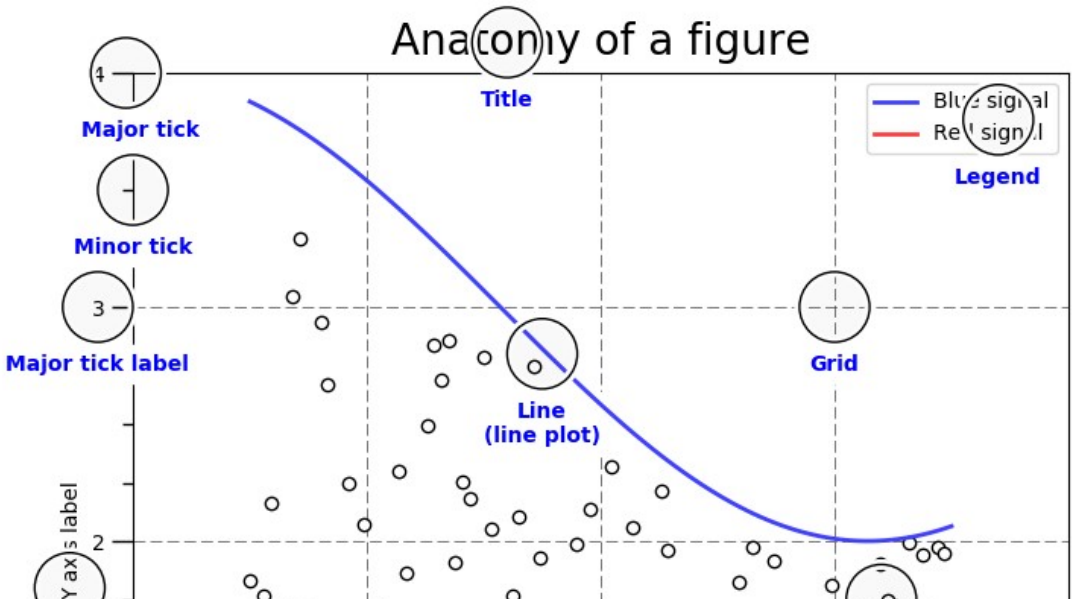
"Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python."

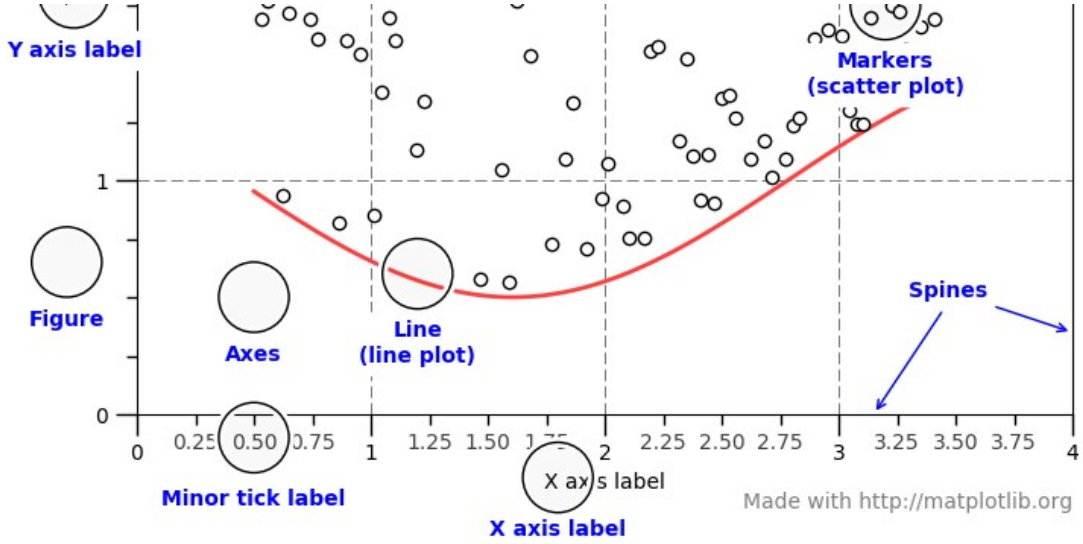
Matplotlib is one of the most popular libraries used to create data vizualizations in Python. It uses an object-oriented API (classes) which we've already worked with when using Pandas

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.





In [3]:
```
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-tu
# 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()

#Save our plot as an image
plt.savefig('line_plot.png')
```

Pandas Plotting

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

In [4]:
```
#Import Pandas
import pandas as pd

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

print("Setup Complete")
```

```
Setup Complete
```

In [5]:
```
# Import stock data
stock_df = pd.read_csv('data/stocks.csv', index_col="date", parse_date
```

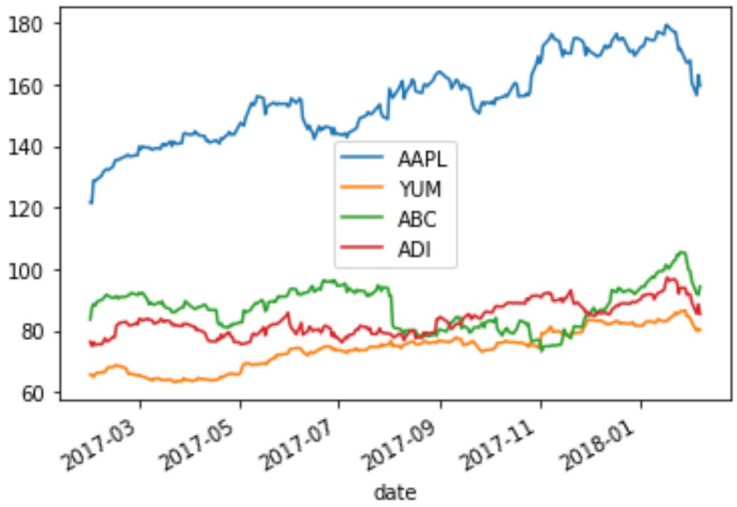
In [6]:
```
#Take a look at the data
stock_df.head()
```

Out[6]:

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

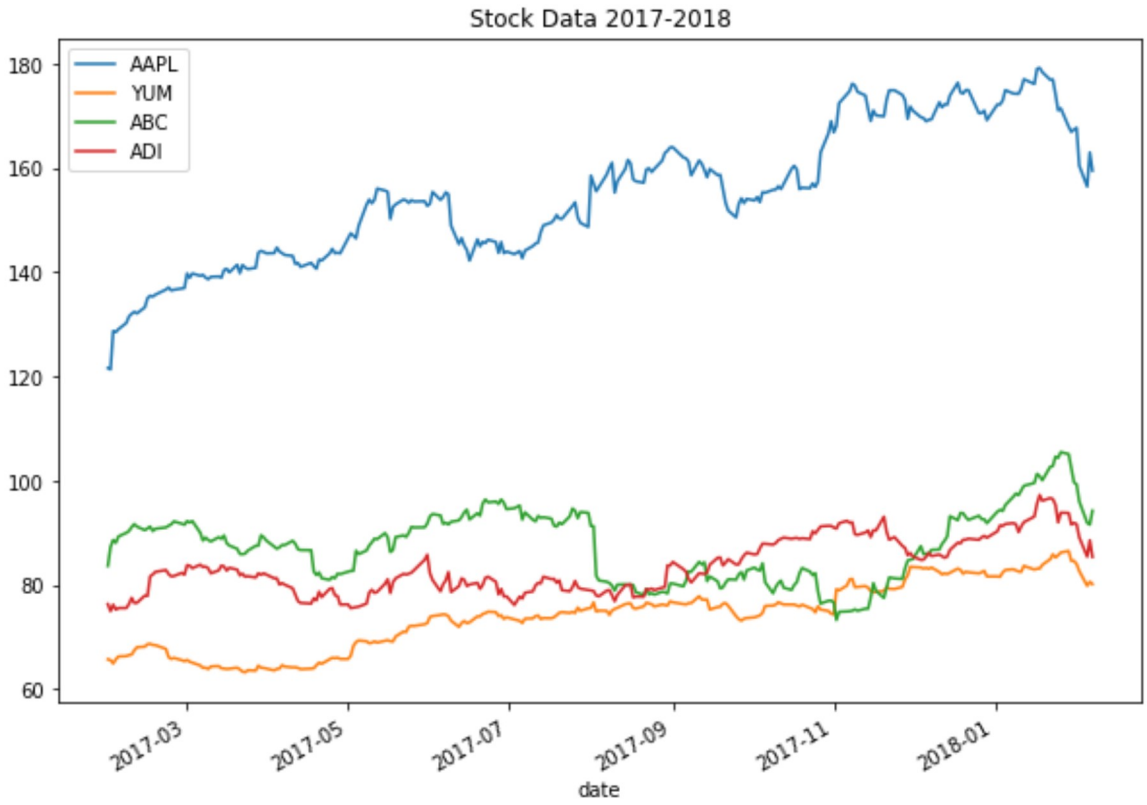
In [7]:
```
#Plotting data as easy as calling the plot() function
stock_df.plot(kind='line')
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2318df63828>
```



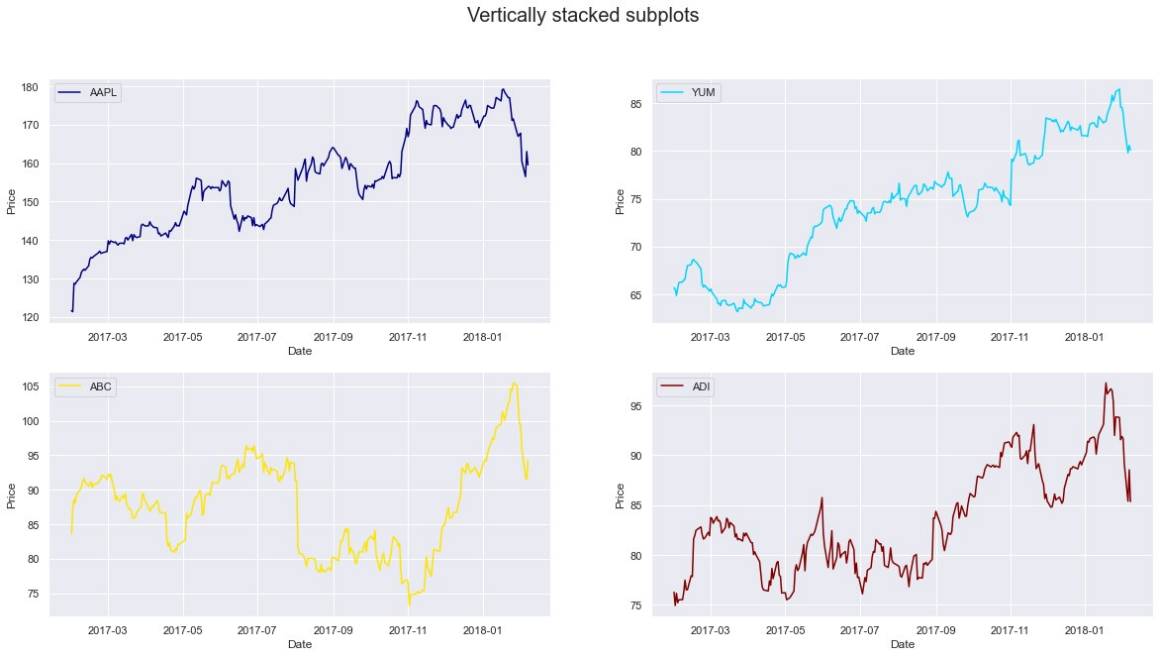
In [8]:
```
#Plotting data as easy as calling the plot() function
stock_df.plot(kind='line', figsize=(10,7), title='Stock Data 2017-2018
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2318e017940>

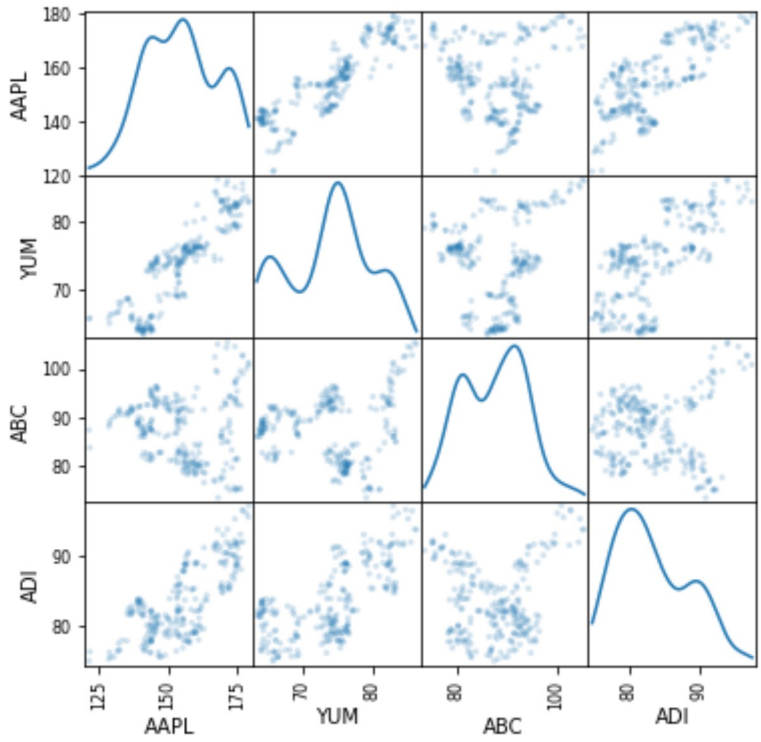


In [38]:

```
#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 addicitor
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 [10]:
```
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 [11]:
```
#https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
#Import new dataset from the data/AB_NYC_2019.csv file into variable a
ab_df = pd.read_csv('data/AB_NYC_2019.csv')
```

In [12]:
```
ab_df.head()
```

Out[12]:

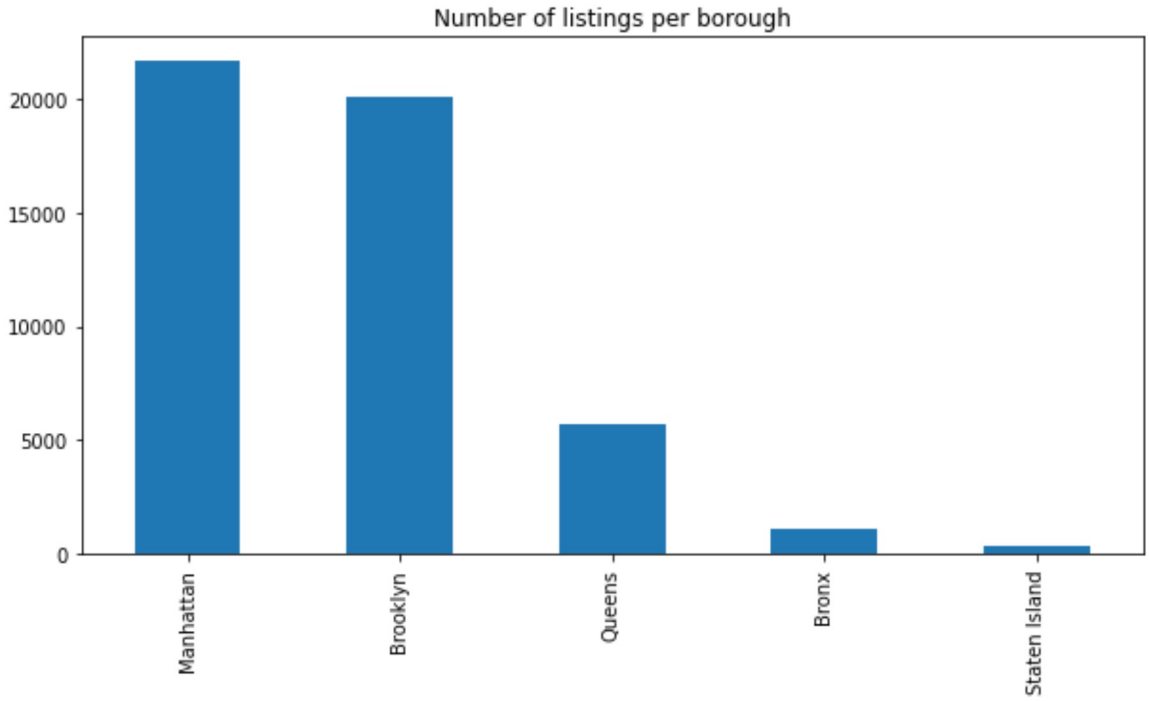
	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitud
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6474
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7536
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8090

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitud
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6851
		Entire Apt: Spacious					

In [13]:
```
#Calculate the number of listings per borough (neighbourhood group)
freq = ab_df['neighbourhood_group'].value_counts()
```

In [14]:
```
#Plot this data as a bar chart
plt.figure(figsize=(10,5))

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

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x231a098c588>



In [15]:
```
#Create a pivot so we can visualize the data
ab_pivot = ab_df.pivot_table(values='minimum_nights', index="neighbour
                             columns = 'room_type', aggfunc=np.sum)
```

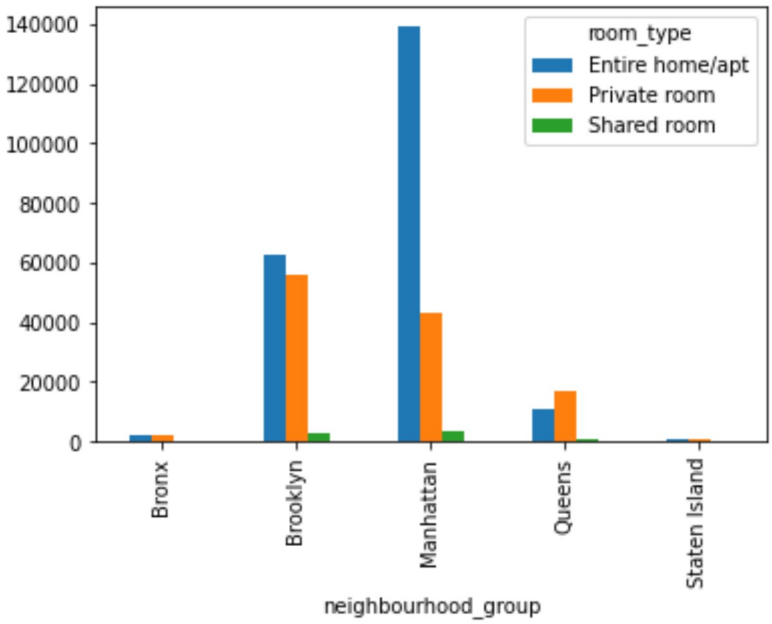
In [16]:
```
ab_pivot.head()
```

Out[16]:

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

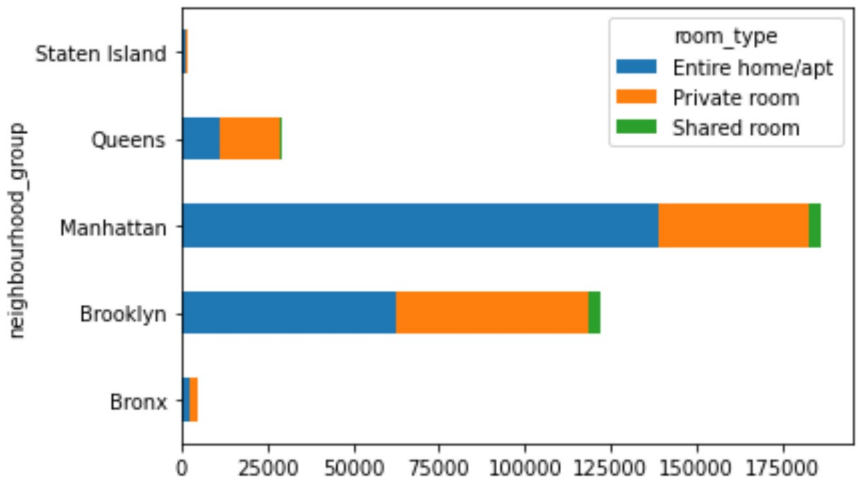
In [17]:
```
#Create a basic bar chart of the pivot table
ab_pivot.plot(kind='bar')
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2318e387080>

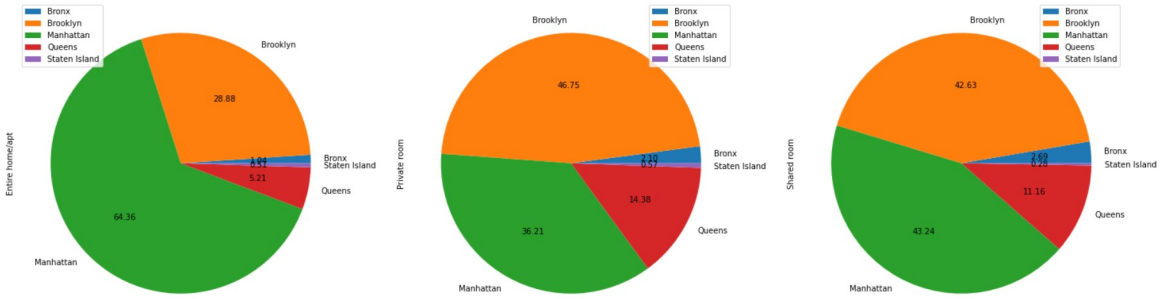


In [18]:
```
#We can stack the bar chart and change the orientation to horizontal
ab_pivot.plot(kind='barh', stacked=True)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x2318e29eac8>



In [19]:
```
#Pie charts and configurations
ab_pivot.plot(kind='pie', subplots=True, figsize=(25, 10),  autopct='%
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 [20]:
```
import seaborn as sns

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

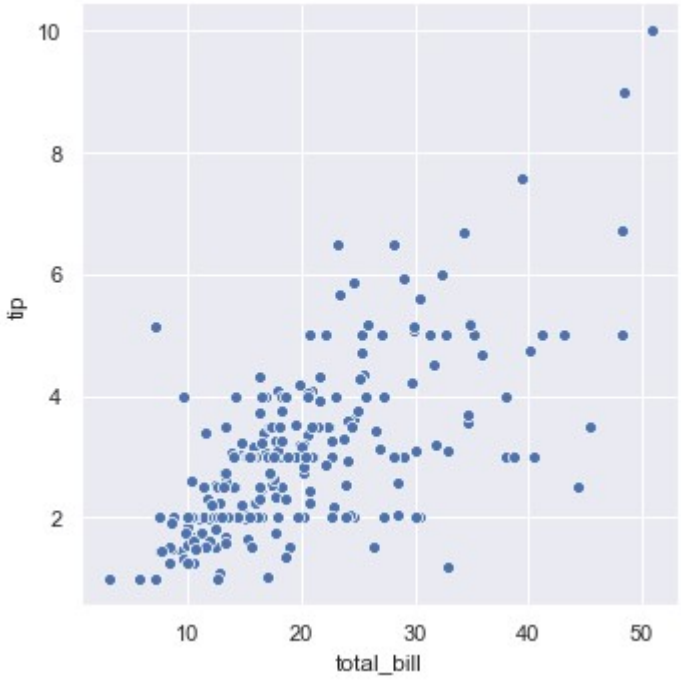
In [21]:
```
tips.head()
```

Out[21]:

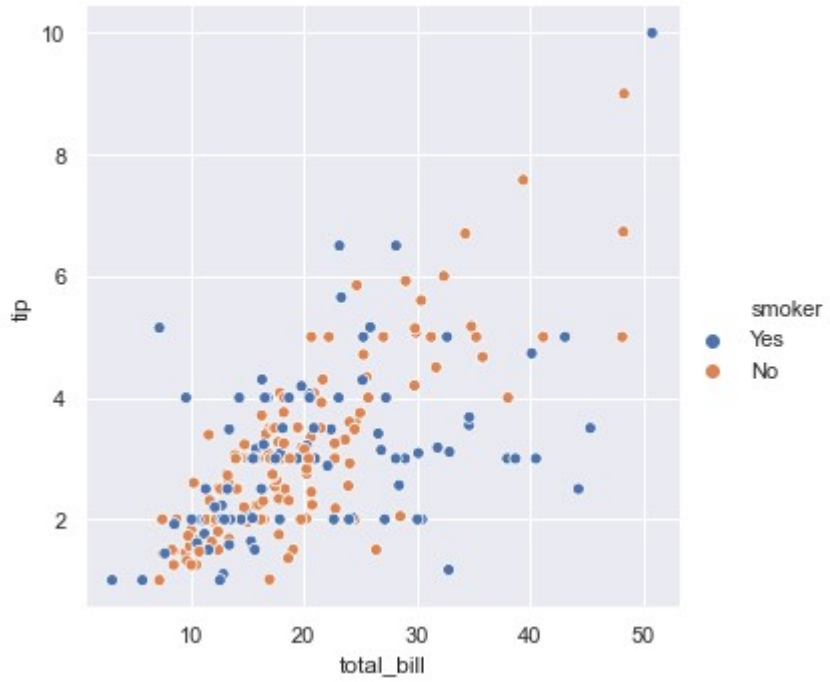
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [22]:
```
#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[22]:
```
<seaborn.axisgrid.FacetGrid at 0x231a0f4d1d0>

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

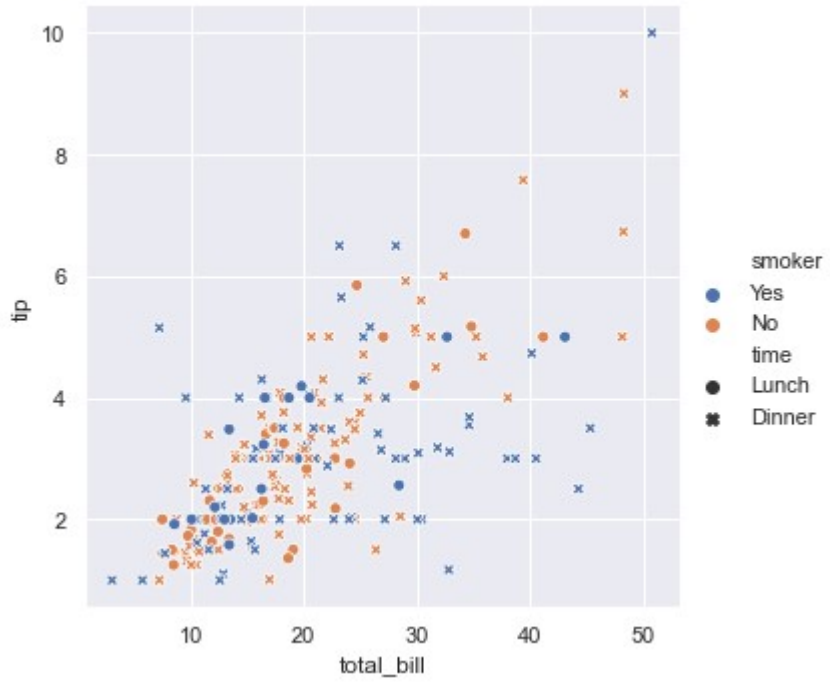


In [45]:
```
# We can add a third dimension with color and style
sns.relplot(x="total_bill", y="tip", hue="smoker", data=tips);
```



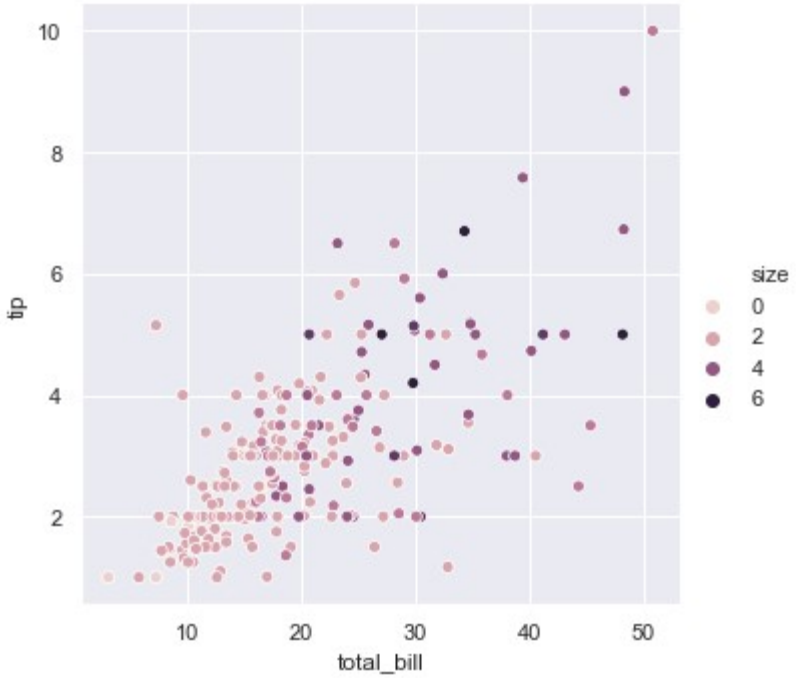
In [24]:
```
#Add a fourth dimension using different variables for hue and style
sns.relplot(x="total_bill", y="tip", hue="smoker", style="time", data=
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x231a1219390>

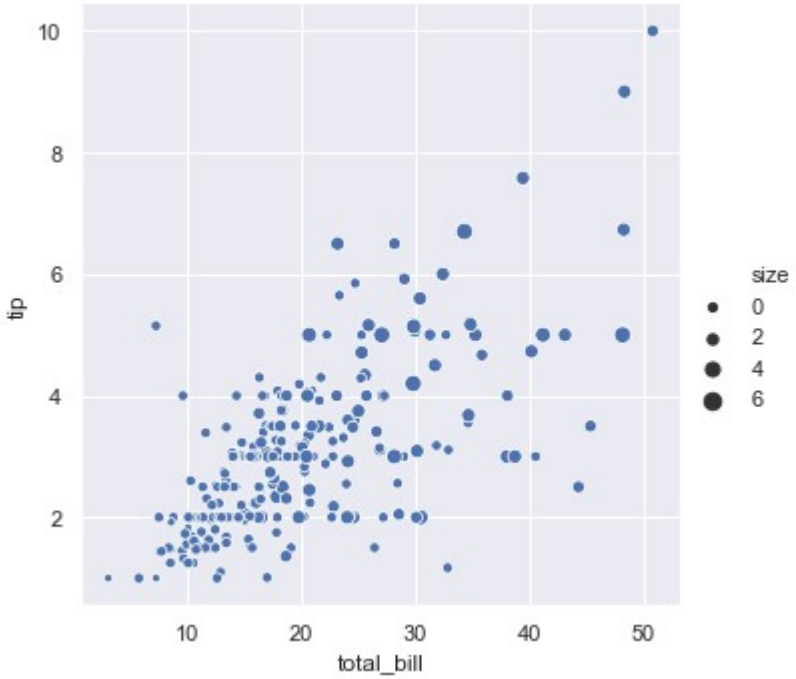


In [25]:
```
#Replot using size variable for hue
sns.relplot(x="total_bill", y="tip", hue="size", data=tips)
```

Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x231a12a6320>
```

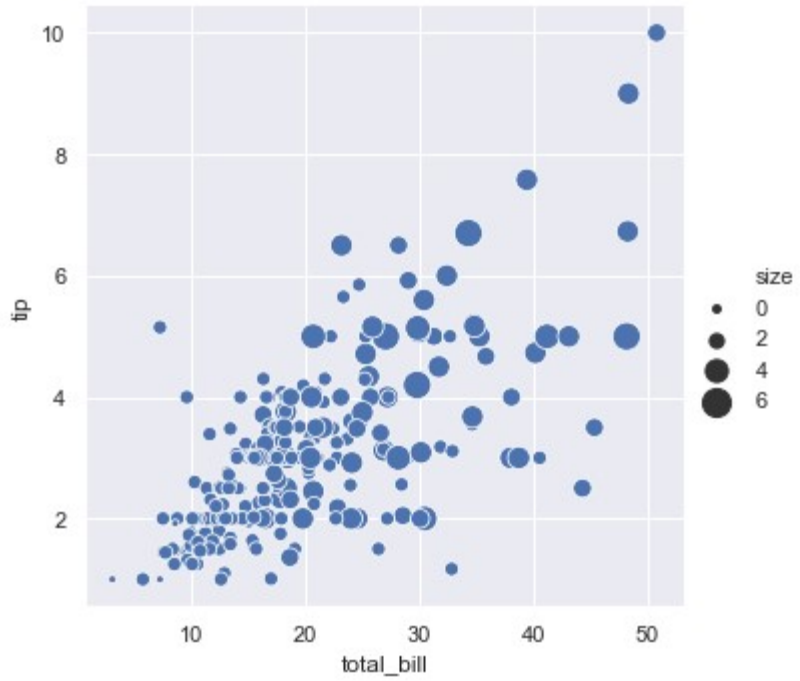


In [46]:
```
#The size parameter allows us to change the size of data points using
sns.relplot(x="total_bill", y="tip", size="size", data=tips);
```



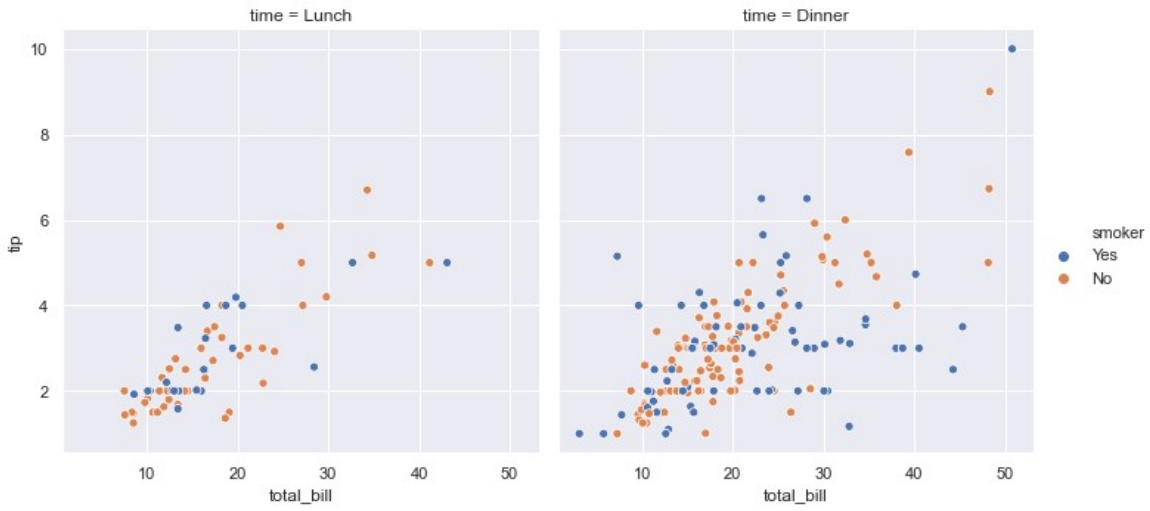
In [27]:
```
#The sizes parameter determines the scale of the data points
sns.relplot(x="total_bill", y="tip", size="size", sizes=(15, 200), dat
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x231a16376a0>



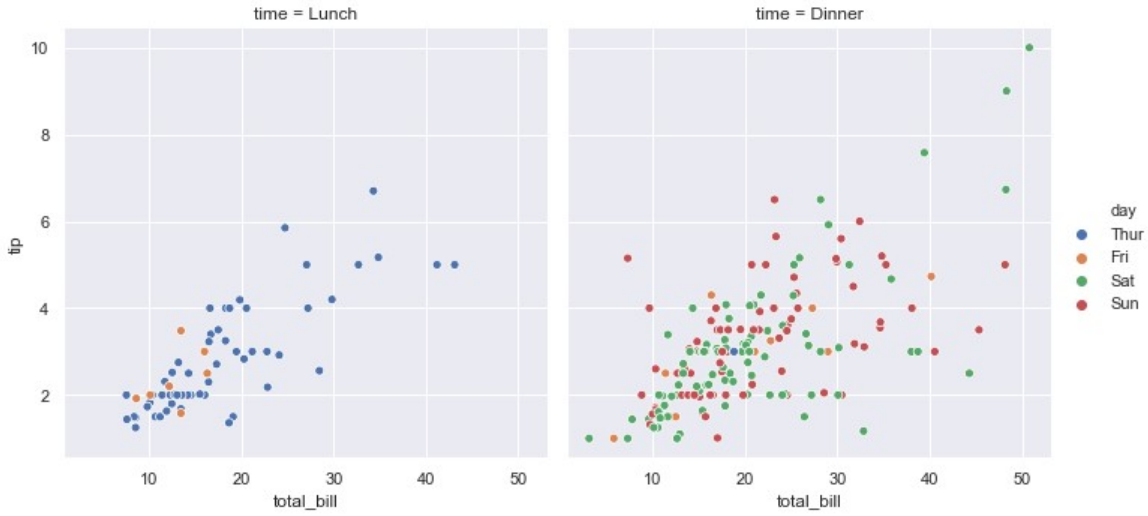
In [28]:
```
#The col parameter creates subplots along the provided variable
sns.relplot(x="total_bill", y="tip", hue="smoker", col="time", data=ti
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x231a1566780>



In [42]:
```
#Try plotting the Day of the week to see if it has an effect on the ti
sns.relplot(x="total_bill", y="tip", hue="day", col="time", data=tips)

#Derive one insight from the graph
#A:
```

Out[42]:
```
<seaborn.axisgrid.FacetGrid at 0x231a2533080>
```



In [29]:

Out[29]:

	AAPL	ABC
date		
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 [30]:
```
#Create a pivot table of the tips data
hm = tips.pivot_table(index='day',columns='size',values='tip')
```

In [31]:
```
hm.head()
```

Out[31]:

size	1	2	3	4	5	6
day						
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 [32]:
```
#An effective way to plot our pivoted data is with a heatmap
sns.heatmap(hm)
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x2318e1df470>