Time series analysis of top US companies

In [1]:
```
import pandas as pd
import fix_yahoo_finance as fyf
from matplotlib import pyplot as plt
from pandas_datareader import data as pdr
fyf.pdr_override()
%matplotlib inline
```

In [2]:
```
CMCSA = pdr.get_data_yahoo('CMCSA', start='2017-01-01')
CMCSA.head()
```

```
[**********100%**********]  1 of 1 downloaded
```

Out[2]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-01-03	34.834999	34.945000	34.115002	34.525002	33.212471	23670400
2017-01-04	34.755001	35.130001	34.590000	34.935001	33.606884	22010800
2017-01-05	34.794998	35.130001	34.700001	35.075001	33.741566	16986000
2017-01-06	35.105000	35.270000	34.910000	35.134998	33.799274	13528000
2017-01-09	35.070000	35.480000	35.025002	35.415001	34.068638	18135400

In [3]:
```
plt.plot(CMCSA['Close'])
plt.show()
```

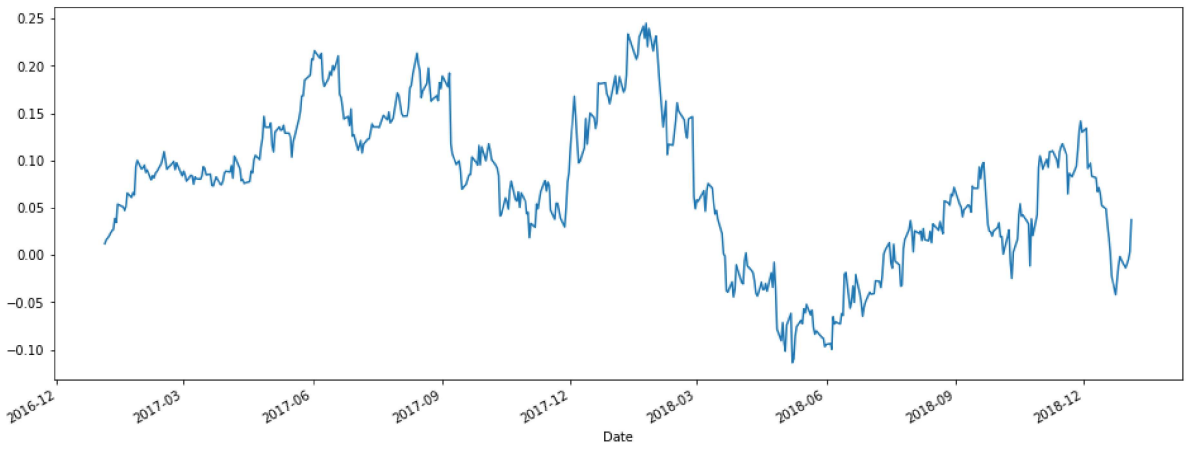
In [4]:
```
CMCSA['dailyReturn'] = CMCSA['Close'].pct_change()
CMCSA['cumReturn'] = CMCSA['Close'].diff().cumsum() / CMCSA['Close'].iloc[0]
CMCSA.head()
```

Out[4]:

	Open	High	Low	Close	Adj Close	Volume	dailyReturn	cu
Date								
2017-01-03	34.834999	34.945000	34.115002	34.525002	33.212471	23670400	NaN	Na
2017-01-04	34.755001	35.130001	34.590000	34.935001	33.606884	22010800	0.011875	0.0
2017-01-05	34.794998	35.130001	34.700001	35.075001	33.741566	16986000	0.004007	0.0
2017-01-06	35.105000	35.270000	34.910000	35.134998	33.799274	13528000	0.001711	0.0
2017-01-09	35.070000	35.480000	35.025002	35.415001	34.068638	18135400	0.007969	0.0

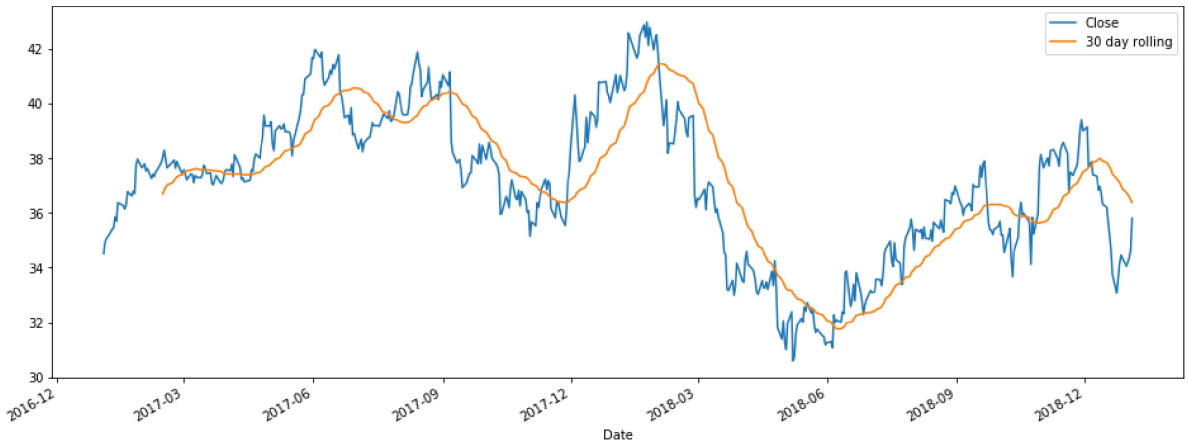
In [5]:
```
CMCSA['cumReturn'].plot(figsize=(16,6))
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1b94835cc88>



In [6]:
```
# create new column for 30 days then plot
CMCSA['30 day rolling'] = CMCSA['Close'].rolling(window=30).mean()
CMCSA[['Close', '30 day rolling']].plot(figsize=(16,6))
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1b9486025f8>



In [7]:
```
# imports daily closing price and volume
NFLX = pdr.get_data_yahoo('NFLX', start='2014-01-01')[['Close','Volume']]
CMCSA = pdr.get_data_yahoo('CMCSA', start='2014-01-01')[['Close','Volume']]
VZ = pdr.get_data_yahoo('VZ', start='2014-01-01')[['Close','Volume']]
GE = pdr.get_data_yahoo('GE', start='2014-01-01')[['Close','Volume']]
```

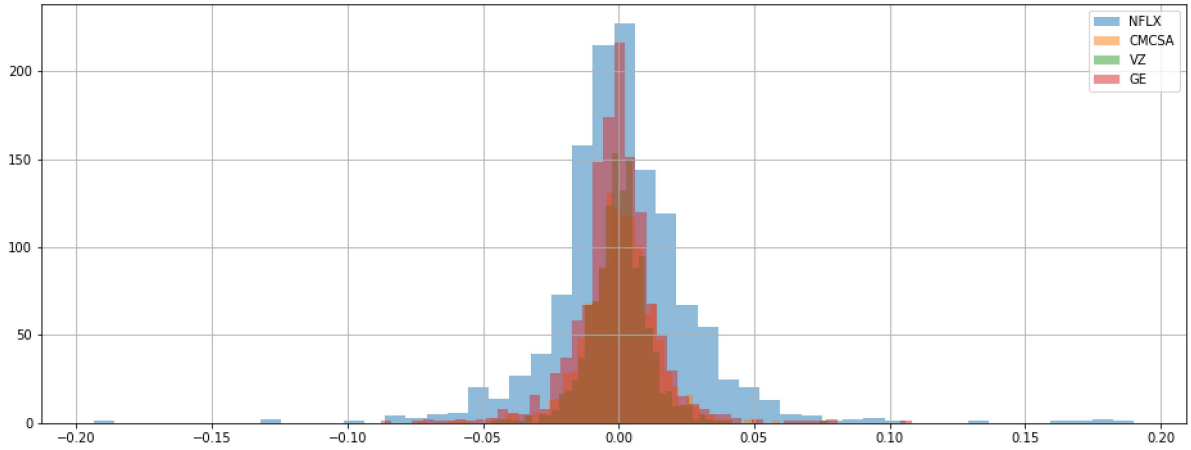
```
[**********************100%**********************]  1 of 1 downloaded
[**********************100%**********************]  1 of 1 downloaded
[**********************100%**********************]  1 of 1 downloaded
[**********************100%**********************]  1 of 1 downloaded
```

In [8]:
```
# creates daily price change column
NFLX['dailyChange'] = NFLX['Close'].pct_change()
CMCSA['dailyChange'] = CMCSA['Close'].pct_change()
VZ['dailyChange'] = VZ['Close'].pct_change()
GE['dailyChange'] = GE['Close'].pct_change()
```

In [9]:
```
# creates cumulative price change percentage since 2014
NFLX['cumulativeReturns'] = (1+ NFLX['dailyChange']).cumprod()
CMCSA['cumulativeReturns'] = (1+ CMCSA['dailyChange']).cumprod()
VZ['cumulativeReturns'] = (1+ VZ['dailyChange']).cumprod()
GE['cumulativeReturns'] = (1+ GE['dailyChange']).cumprod()
```

In [10]:
```
# histogram of daily returns bucketed into bins of 50
NFLX['dailyChange'].hist(bins=50,label='NFLX', figsize=(16,6),alpha=0.5)
CMCSA['dailyChange'].hist(bins=50,label='CMCSA',alpha=0.5)
VZ['dailyChange'].hist(bins=50,label='VZ',alpha=0.5)
GE['dailyChange'].hist(bins=50,label='GE',alpha=0.5)
plt.legend()
```

Out[10]:
```
<matplotlib.legend.Legend at 0x1b9483e3358>
```



In [11]:
```
# combines daily return data of all companies into a single dataframe
box_df = pd.concat([NFLX['dailyChange'], CMCSA['dailyChange'], VZ['dailyChang
e'], GE['dailyChange']], axis=1)
box_df.columns = ['Netflix returns', 'Comcast returns', 'Verizon returns', 'GE
 returns']
```

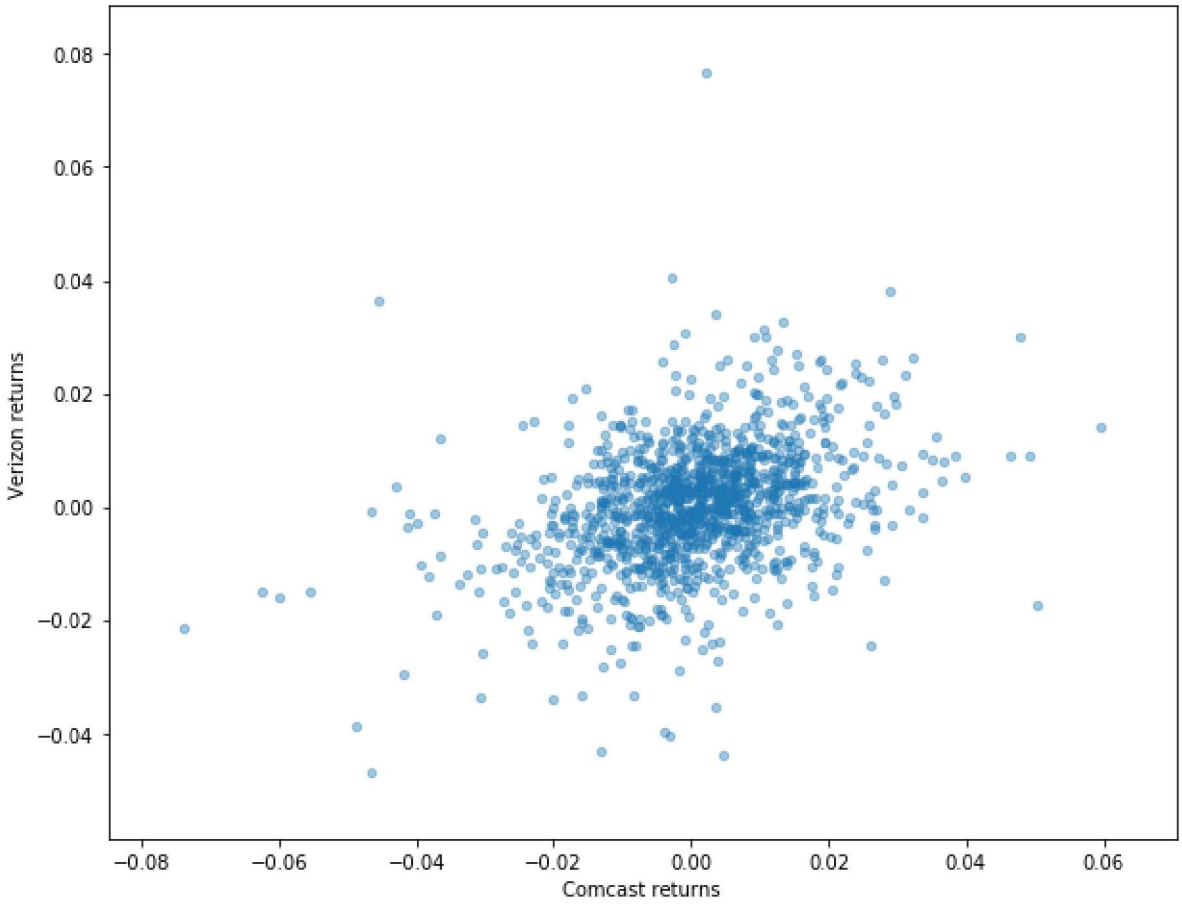
In [12]:
```
box_df.head()
```

Out[12]:

	Netflix returns	Comcast returns	Verizon returns	GE returns
Date				
2014-01-02	NaN	NaN	NaN	NaN
2014-01-03	0.000772	-0.007386	-0.011837	-0.000727
2014-01-06	-0.009722	-0.000979	0.005576	-0.008006
2014-01-07	-0.055817	0.035476	0.012528	0.001101
2014-01-08	0.004389	-0.001514	-0.016227	-0.002932

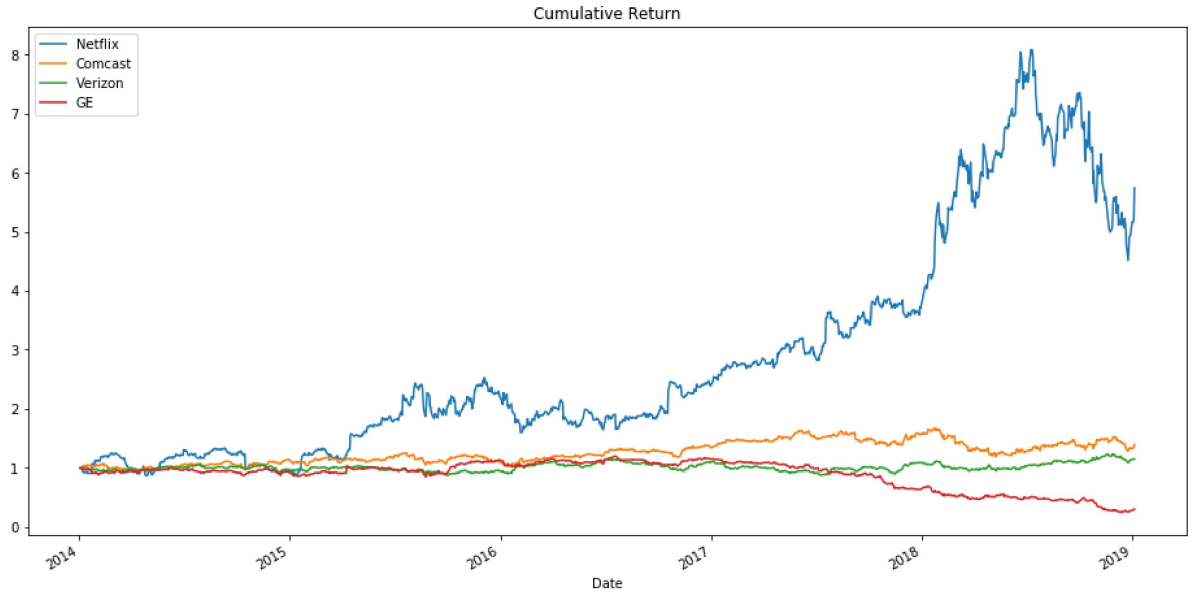
In [13]:
```
# scatter plot of Comcast and Verizion returns
box_df.plot(kind='scatter', x='Comcast returns', y='Verizon returns', alpha=0.
4,figsize=(10,8))
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1b9486ed8d0>



In [14]:
```
# time series chart of cumulative stock returns over the last four years
NFLX['cumulativeReturns'].plot(label='Netflix', figsize=(16,8), title='Cumulat
ive Return')
CMCSA['cumulativeReturns'].plot(label='Comcast')
VZ['cumulativeReturns'].plot(label='Verizon')
GE['cumulativeReturns'].plot(label='GE')
plt.legend()
```

Out[14]: <matplotlib.legend.Legend at 0x1b948a6c9b0>



Comcast stock price analysis using Bollinger Bands

In [2]:
```
import pandas as pd
import fix_yahoo_finance as fyf
from matplotlib import pyplot as plt
from pandas_datareader import data as pdr
fyf.pdr_override()
%matplotlib inline
```

In [3]:
```
# reads in Comcast stock to df
df = pdr.get_data_yahoo('CMCSA', start='2017-01-01')
```

```
[**********100%**********]  1 of 1 downloaded
```

In [4]:
```
df.tail()
```

Out[4]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-12-28	34.860001	35.360001	34.180000	34.349998	34.160000	16231300
2018-12-31	34.290001	34.599998	33.299999	34.049999	34.049999	21613200
2019-01-02	33.490002	34.450001	33.419998	34.369999	34.369999	16970400
2019-01-03	34.330002	35.330002	34.130001	34.639999	34.639999	28750400
2019-01-04	35.029999	35.840000	34.860001	35.810001	35.810001	22843800

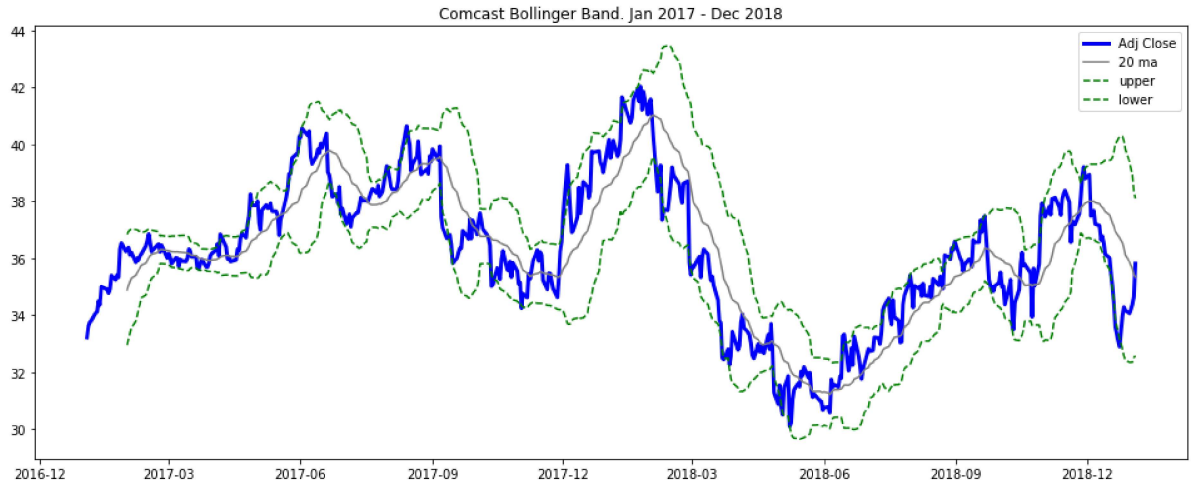
In [5]:
```
# Add columns "20 day moving average", "20 day standard deviation", "STD upper
 band", "STD low band"
df['20 ma'] = df['Adj Close'].rolling(20).mean()
df['20 sd'] = df['Adj Close'].rolling(20).std()
df['upper'] = df['20 ma'] + (df['20 sd'] * 2)
df['lower'] = df['20 ma'] - (df['20 sd'] * 2)
```

In [6]:
```
df.head()
```

Out[6]:

	Open	High	Low	Close	Adj Close	Volume	20 ma	20 sd	upp
Date									
2017-01-03	34.834999	34.945000	34.115002	34.525002	33.212471	23670400	NaN	NaN	NaN
2017-01-04	34.755001	35.130001	34.590000	34.935001	33.606884	22010800	NaN	NaN	NaN
2017-01-05	34.794998	35.130001	34.700001	35.075001	33.741566	16986000	NaN	NaN	NaN
2017-01-06	35.105000	35.270000	34.910000	35.134998	33.799274	13528000	NaN	NaN	NaN
2017-01-09	35.070000	35.480000	35.025002	35.415001	34.068638	18135400	NaN	NaN	NaN

In [7]:
```
fig = plt.figure(figsize=(16,6))
axis = fig.add_axes([0.1,0.1,0.8,0.8])
axis.plot(df['Adj Close'],c='b',lw=3)
axis.plot(df['20 ma'],c='gray')
axis.plot(df['upper'],c='g',ls='--')
axis.plot(df['lower'],c='g',ls='--')
plt.title('Comcast Bollinger Band. Jan 2017 - Dec 2018')
plt.legend();
```



Seaborn libary for visualization

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# loads dummy dataset from seaborn libary
df = sns.load_dataset('tips')
df.head()
```

Out[2]:

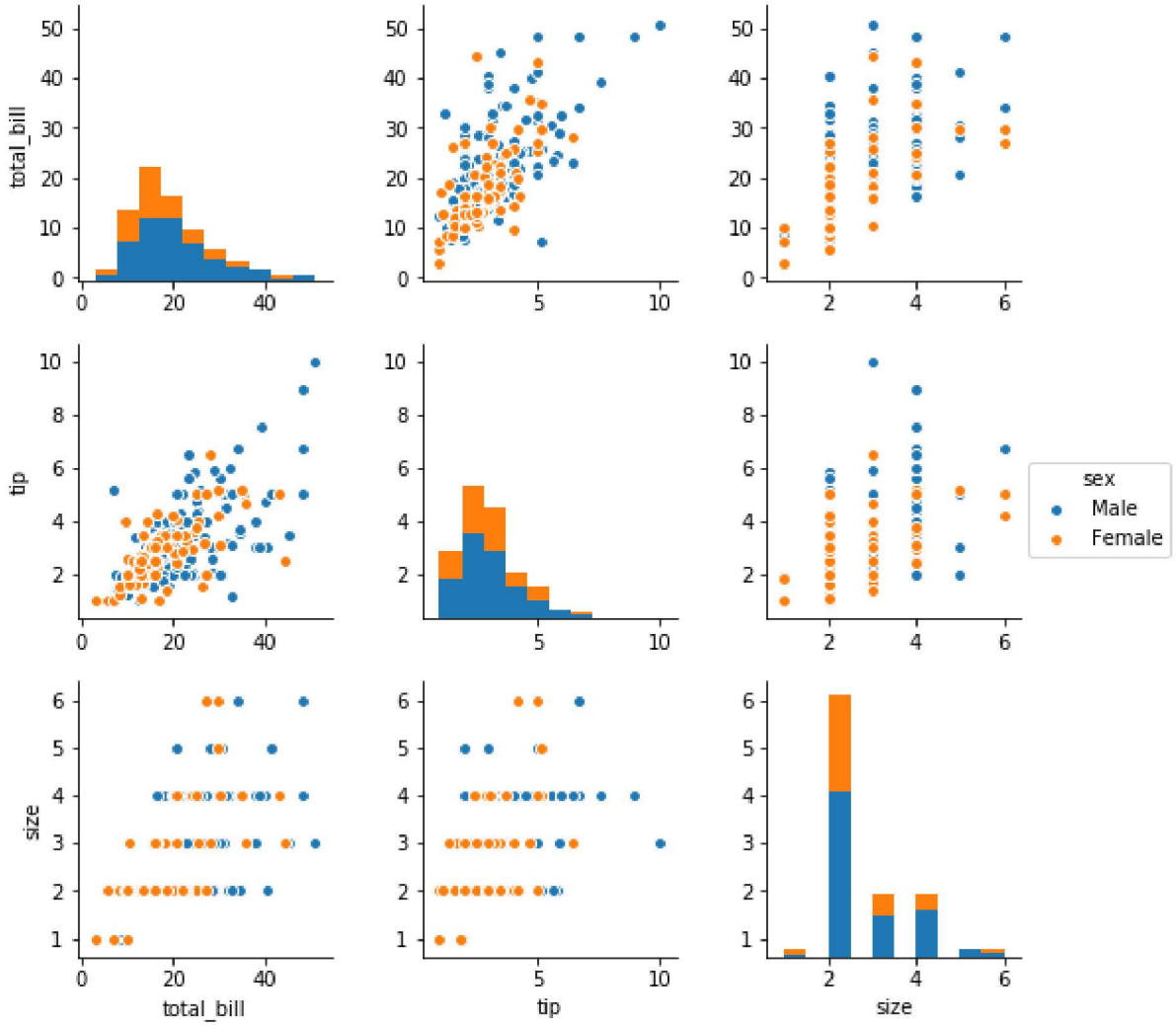
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [3]:
```
df.info()
```

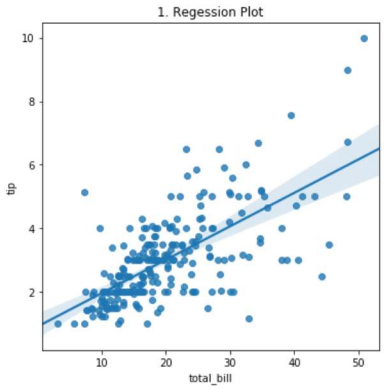
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.2 KB
```

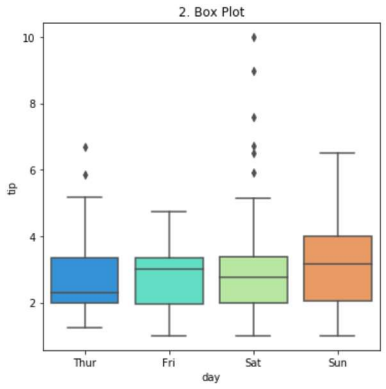
In [4]:
```
# pairplot
sns.pairplot(df,hue='sex')
```

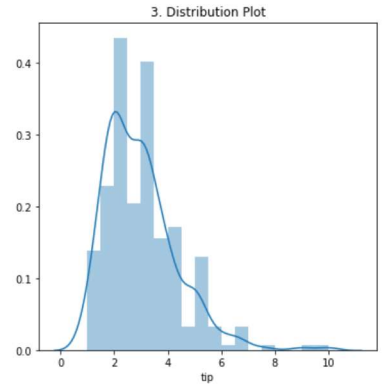
Out[4]: <seaborn.axisgrid.PairGrid at 0x20853a22898>

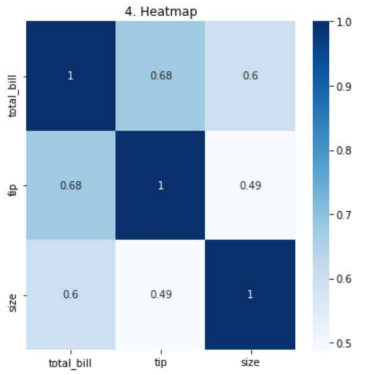


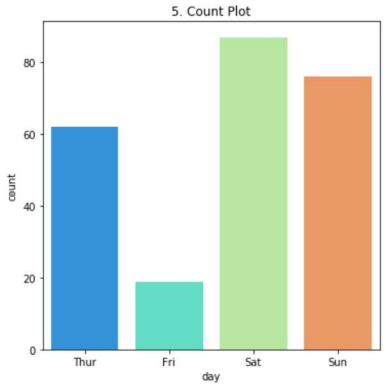
In [5]:
```
# Collection of various charts available in seaborn

fig, axes = plt.subplots(nrows=3, ncols=3,figsize=(20,20))

sns.regplot(x='total_bill',y='tip',data=df, ax=axes[0][0])
axes[0][0].set_title('1. Regession Plot')

sns.boxplot(x='day',y='tip',data=df, ax=axes[0][1],palette='rainbow')
axes[0][1].set_title('2. Box Plot')

sns.distplot(df['tip'],ax=axes[0][2])
axes[0][2].set_title('3. Distribution Plot')

sns.heatmap(df.corr(),annot=True,cmap='Blues',ax=axes[1][0])
axes[1][0].set_title('4. Heatmap')

sns.countplot(x='day',data=df,ax=axes[1][1], palette='rainbow')
axes[1][1].set_title('5. Count Plot')

sns.stripplot(x='day',y='total_bill',data=df,jitter=True,hue='sex', ax=axes[1]
[2])
axes[1][2].set_title('6. Strip Plot')

sns.violinplot(x='time',y='total_bill',data=df,hue='smoker',split=True,palette
='Set1', ax=axes[2][0])
axes[2][0].set_title('7. Violin Plot')

sns.barplot(x='total_bill',y='day',data=df,ax=axes[2][1])
axes[2][1].set_title('8. Bar Plot')

sns.lvplot(x='total_bill',y='day',data=df,ax=axes[2][2])
axes[2][2].set_title('9. Letter Value Plot');
```

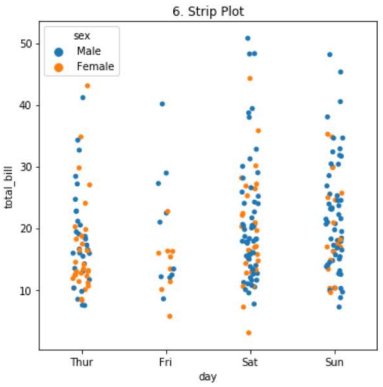

1. Regession Plot
tip
total_bill

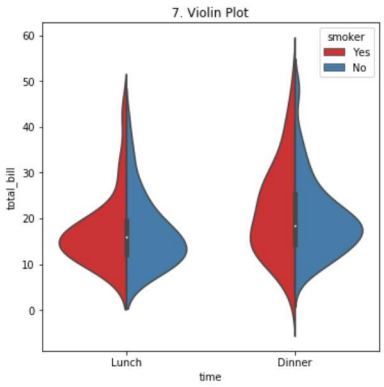

2. Box Plot
tip
day
Thur
Fri
Sat
Sun

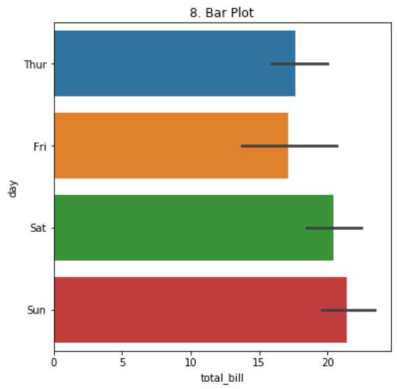

3. Distribution Plot
tip

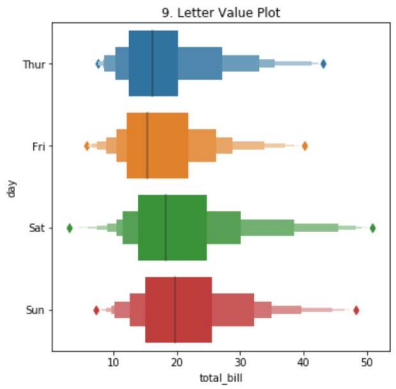

4. Heatmap
total_bill
tip
size
1
0.68
0.6
0.68
1
0.49
0.6
0.49
1


5. Count Plot
count
day
Thur
Fri
Sat
Sun


6. Strip Plot
sex
Male
Female
total_bill
day
Thur
Fri
Sat
Sun


7. Violin Plot
smoker
Yes
No
total_bill
time
Lunch
Dinner


8. Bar Plot
day
total_bill
Thur
Fri
Sat
Sun


9. Letter Value Plot
day
total_bill
Thur
Fri
Sat
Sun

Coding guessing game with a for loop and while loop.

In [1]:
```
#This is a number guessing game using a while loop
import random

#asks for the player's name
print('Hello. What is your name?')
name = input()

print('Well ' + name + ', I am thinking of a number between 1 and 10')
secretNum = random.randint(1,10)
numOfGuesses = 0
allowedGuesses = 6

#player takes the allowed number of guesses
while numOfGuesses < allowedGuesses:
    try: #requires numeric guess
        print('Take a guess.')
        guess = int(input())
        if guess < secretNum:
            print('Your guess is too low')
        elif guess > secretNum:
            print('Your guess is too high')
        else:
            print('You guessed the number!')
            break #breaks out of loop if guess is correct
        numOfGuesses += 1
    except ValueError:
        print('You must enter a numeric number!')

if numOfGuesses == allowedGuesses:
    print('Sorry ' + name + '. You\'ve taken too many guesses. Try again late
r')
else:
    print('Congratulations ' + name + '! The secret number is ' + str(secretNu
m))
```

```
Hello. What is your name?
Kin
Well Kin, I am thinking of a number between 1 and 10
Take a guess.
5
Your guess is too high
Take a guess.
3
Your guess is too high
Take a guess.
1
You guessed the number!
Congratulations Kin! The secret number is 1
```

In [2]:
```
#this is a guess the number game using a for loop
import random

#asks for the player's name
print('Hello. What is your name?')
name = input()

print('Well ' + name + ', I am thinking of a number between 1 and 10')
secretNum = random.randint(1,10)

#player takes 6 guesses
for guessesTaken in range(1, 7):
    try: #requires numeric guess
        print('Take a guess.')
        guess = int(input())

        if guess < secretNum:
            print('Your guess is too low')
        elif guess > secretNum:
            print('Your guess is too high')
        else:
            print('You guessed it!')
            break #breaks out of loop of guess is correct
    except ValueError:
        print('You must enter a numeric number!')

if guessesTaken == 6:
    print('Sorry ' + name + '. You\'ve taken too many guesses. Try again late
r')
else:
    print('Congratuations ' + name + '! The secret number is ' + str(secretNum
))
```

```
Hello. What is your name?
Kin
Well Kin, I am thinking of a number between 1 and 10
Take a guess.
5
Your guess is too high
Take a guess.
3
Your guess is too high
Take a guess.
1
Your guess is too low
Take a guess.
2
You guessed it!
Congratuations Kin! The secret number is 2
```

Comcast stock analysis with OHLC chart (open, high, low, close)

In [3]:
```
from matplotlib import pyplot as plt
from matplotlib import style
from matplotlib.finance import candlestick_ohlc
from pandas_datareader import data as pdr
import matplotlib.dates as mdates
import pandas as pd
import fix_yahoo_finance as fyf
fyf.pdr_override()
style.use('ggplot')
%matplotlib inline
```

In [4]:
```
# reads in Comcast stock to df
df = pdr.get_data_yahoo('CMCSA',start='2016-01-01')
style.use('ggplot')
```

```
[**********100%**********]  1 of 1 downloaded
```

In [5]:
```
df.head()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2016-01-04	27.575001	27.825001	27.424999	27.820000	25.877962	26524600
2016-01-05	27.920000	28.025000	27.740000	27.825001	25.882614	28308800
2016-01-06	27.459999	27.825001	27.375000	27.610001	25.682621	21650800
2016-01-07	27.045000	27.680000	26.885000	27.305000	25.398912	32423400
2016-01-08	27.430000	28.020000	27.285000	27.334999	25.426817	28184600

In [6]:
```
# resampling with a 10 day window
df_ohlc = df['Adj Close'].resample('10D').ohlc()
df_volume = df['Volume'].resample('10D').sum()
```

In [7]:
```
# resets the idnex so the dates is a column
df_ohlc.reset_index(inplace=True)
df_ohlc['Date'] = df_ohlc['Date'].map(mdates.date2num)
```

In [8]:
```
df_ohlc.head()
```

Out[8]:

	Date	open	high	low	close
0	735967.0	25.877962	25.882614	24.905910	24.905910
1	735977.0	25.603554	25.733786	24.933815	25.733786
2	735987.0	25.212872	25.957031	25.166365	25.389610
3	735997.0	26.901169	27.631376	25.957031	26.408169
4	736007.0	26.733742	27.045357	26.659327	27.045357

In [9]:
```
plt.figure(figsize=(16,6))
ax1 = plt.subplot2grid((6,1),(0,0),rowspan=5,colspan=1)
ax2 = plt.subplot2grid((6,1),(5,0),rowspan=1,colspan=1,sharex=ax1)
ax1.xaxis_date()

candlestick_ohlc(ax1, df_ohlc.values, width=2, colorup='g')
ax2.fill_between(df_volume.index.map(mdates.date2num),df_volume.values,0);
```

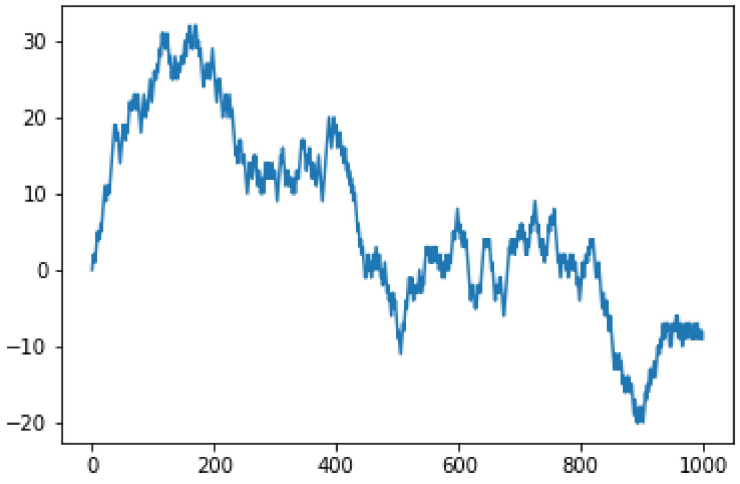
Random walk simulation

In [5]:
```
import matplotlib.pyplot as plt
import random
position = 0
walk = [position]
steps = 1000
for i in range(steps):
    step = 1 if random.randint(0,1) else -1
    position += step
    walk.append(position)
```

In [6]:
```
# current steps from original point
position
```

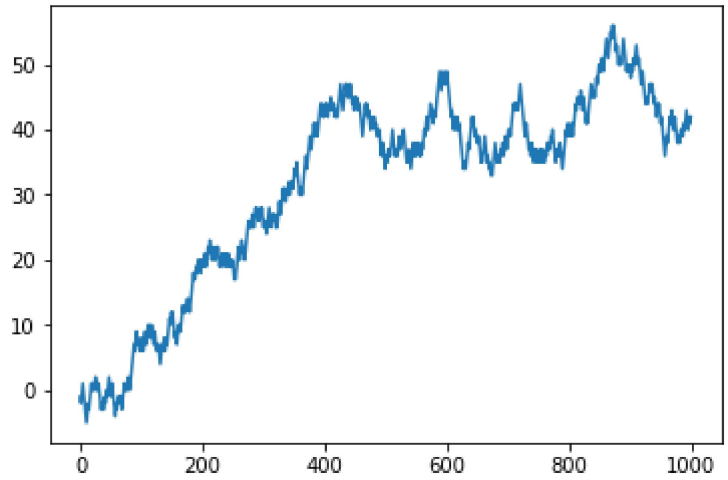
Out[6]: -10

In [8]:
```
# tracks historical positioning
plt.plot(walk[:1000])
plt.show()
```



In [9]:
```
# does the same as above but with the numpy library
import numpy as np
nsteps = 1000
draws = np.random.randint(0,2,size=nsteps)
steps = np.where(draws > 0,1,-1)
walk = steps.cumsum()
```

In [11]:
```
# plots first 100 positions out of 10000
plt.plot(walk[:1000])
plt.show()
```



In [12]:
```
# shows the index in which the walker reached 10 paces away from zero
(np.abs(walk) >= 10).argmax()
```

Out[12]: 111

In [13]:
```
# simulating 10000 walks of 100 steps.
position= 0
nsteps = 100
nwalks = 10000

draws = np.random.randint(0, 2, size=(nwalks,nsteps)) #array of 0 and 1
steps = np.where(draws > 0, 1, -1)
walks = steps.cumsum(1)
```

In [14]:
```
walks
```

Out[14]:
```
array([[  1,   2,   1, ..., -14, -15, -14],
       [  1,   0,   1, ...,  -2,  -1,  -2],
       [ -1,  -2,  -1, ...,  12,  11,  12],
       ...,
       [  1,   0,  -1, ...,  28,  29,  30],
       [  1,   0,   1, ...,   0,  -1,   0],
       [ -1,   0,  -1, ...,  -2,  -1,  -2]], dtype=int32)
```

In [20]:
```
print('Most steps taken backwards across 10,000 walks are ' + str(walks.min
()))
print('Most steps taken forward across 10,000 walks are ' + str(walks.max()))
```

```
Most steps taken backwards across 10,000 walks are -39
Most steps taken forward across 10,000 walks are 46
```

Regex to scrape phone numbers and emails from your clipboard

In [ ]:
```
# Code will scrape the phone and email from https://www.phila.gov/departments/
department-of-revenue/about/contact-us/
import re, pyperclip

# Create a regex for phone numbers
phoneRegex = re.compile(r'''
# types of numbers 415-555-5555, (610) 555-5555, 555,0000, 555-0000 ext. 12345
 x12345

(((\d\d\d)|(\(\d\d\d\)))?     # area code (optional)
(\s|-)                       # first separator
\d\d\d                       # first 3 digits
-                            # separator
\d\d\d\d                     # last 4 digits
(((ext(\.)?\s)|x)             # extension word-part (optional)
(\d{2,5}))?                 # extension number-part (optional)
)
''', re.VERBOSE)

# Create a regex for email addresses
emailRegex = re.compile(r'''

# some.+_thing@(\d{2,5}))?.com

[a-zA-Z0-9_.+]+        # name part
@                      # @ symbol
[a-zA-Z0-9_.+]+        # domain name part

''', re.VERBOSE)
```

In [7]:
```
# Get the text off the clipboard
text = pyperclip.paste()

# Extract the email/phone from this text
extractedPhone = phoneRegex.findall(text)
extractedEmail = emailRegex.findall(text)

allPhoneNumbers = []
for phoneNumber in extractedPhone:
    allPhoneNumbers.append(phoneNumber[0])

allEmail = []
for email in extractedEmail:
    allEmail.append(email)

print(extractedPhone)
print(extractedEmail)
```

```
[('(215) 686-6600', '(215)', '', '(215)', ' ', '', '', '', '', ''), ('(215) 6
86-6831', '(215)', '', '(215)', ' ', '', '', '', '', ''), ('(877) 309-3710',
'(877)', '', '(877)', ' ', '', '', '', '', ''), ('(215) 686-6442', '(215)',
'', '(215)', ' ', '', '', '', '', ''), ('(215) 686-4343', '(215)', '', '(21
5)', ' ', '', '', '', '', ''), ('(215) 686-4336', '(215)', '', '(215)', ' ',
'', '', '', '', '', ''), ('(877) 309-3709', '(877)', '', '(877)', ' ', '', '',
'', '', ''), ('(215) 685-6300', '(215)', '', '(215)', ' ', '', '', '', '',
''), ('(215) 685-6300', '(215)', '', '(215)', ' ', '', '', '', '', ''), ('(21
5) 686-6565', '(215)', '', '(215)', ' ', '', '', '', '', ''), ('(215) 686-657
4', '(215)', '', '(215)', ' ', '', '', '', '', ''), ('(215) 686-6575', '(21
5)', '', '(215)', ' ', '', '', '', '', ''), ('(215) 686-6578', '(215)', '',
'(215)', ' ', '', '', '', '', ''), ('(215) 686-2670', '(215)', '', '(215)', '
', '', '', '', '', ''), ('(215) 686-6648', '(215)', '', '(215)', ' ', '', '',
'', '', '')]
['revenue@phila.gov', 'vicki.riley@phila.gov', 'revenue@phila.gov', 'wrbhelpd
esk@phila.gov', 'tax.clearance@phila.gov', 'refundunit@phila.gov', 'agency.re
ceivables@phila.gov', 'egovservices@phila.gov', 'amountdue@phila.gov', 'water
amountdue@phila.gov']
```

In [12]:
```
# shows first 10 phone numbers
allPhoneNumbers[:10]
```

Out[12]:
```
['(215) 686-6600',
 '(215) 686-6831',
 '(877) 309-3710',
 '(215) 686-6442',
 '(215) 686-4343',
 '(215) 686-4336',
 '(877) 309-3709',
 '(215) 685-6300',
 '(215) 685-6300',
 '(215) 686-6565']
```

In [13]:
```
# shows first 10 email addresses
allEmail[:10]
```

Out[13]:
```
['revenue@phila.gov',
 'vicki.riley@phila.gov',
 'revenue@phila.gov',
 'wrbhelpdesk@phila.gov',
 'tax.clearance@phila.gov',
 'refundunit@phila.gov',
 'agency.receivables@phila.gov',
 'egovservices@phila.gov',
 'amountdue@phila.gov',
 'wateramountdue@phila.gov']
```

In [34]:
```
# copies results onto your computer's clipboard
results = '\n'.join(allPhoneNumbers) + '\n' + '\n'.join(extractedEmail)
pyperclip.copy(results)
```