In [1]:

```
import pandas as pd
import datetime as dt
import mplfinance as mpf
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas_datareader.data as web
from matplotlib import style
from mplfinance.original_flavor import candlestick_ohlc
```

In [2]:

```
style.use("ggplot")
```

In [3]:

```
start = dt.datetime(2010, 2, 15) #Upper bound limit
end = dt.datetime(2020, 9, 27)#Lower bound limit
```

In [4]:

```
df = web.DataReader("AMZN", "yahoo", start, end) # Using yahoo API to pull data from S&P 500 logs
```

In [5]:

```
print(df.head(9)) #Test print the 9 most ancient entries
```

```
                 High         Low        Open       Close   Volume  \
Date
2010-02-16  120.500000  117.180000  120.059998  117.529999  8932700
2010-02-17  117.129997  115.550003  117.070000  116.309998  8944800
2010-02-18  118.510002  114.820000  115.839996  118.080002  9800100
2010-02-19  119.089996  117.000000  117.910004  117.519997  7115600
2010-02-22  118.970001  116.180000  117.370003  118.010002  6807300
2010-02-23  119.250000  116.510002  118.010002  117.239998  7068200
2010-02-24  119.800003  117.150002  117.959999  119.720001  7389900
2010-02-25  118.339996  115.849998  118.169998  118.199997  9533400
2010-02-26  119.430000  117.000000  117.879997  118.400002  5721600

             Adj Close
Date
2010-02-16  117.529999
2010-02-17  116.309998
2010-02-18  118.080002
2010-02-19  117.519997
2010-02-22  118.010002
2010-02-23  117.239998
2010-02-24  119.720001
2010-02-25  118.199997
2010-02-26  118.400002
```

In [6]:

```
print(df.tail(18)) #Test print the 18 most recent entries
```

```
                 High          Low         Open        Close   Volume  \
Date
2020-09-01  3513.870117  3467.000000  3489.580078  3499.120117  3476400
2020-09-02  3552.250000  3486.689941  3547.000000  3531.449951  3931500
2020-09-03  3488.409912  3303.000000  3485.000000  3368.000000  8161100
2020-09-04  3381.500000  3111.129883  3318.000000  3294.620117  8781800
2020-09-08  3250.850098  3130.000000  3144.000000  3149.840088  6094200
2020-09-09  3303.179932  3185.000000  3202.989990  3268.610107  5188700
2020-09-10  3349.889893  3170.550049  3307.219971  3175.110107  5330700
2020-09-11  3217.340088  3083.979980  3208.689941  3116.219971  5094000
2020-09-14  3187.389893  3096.000000  3172.939941  3102.969971  4529600
2020-09-15  3175.020020  3108.919922  3136.159912  3156.129883  4021500
```

```
2020-09-16  3187.239990  3074.149902  3179.989990  3078.100098  4512200
2020-09-17  3029.429932  2972.550049  3009.250000  3008.729980  6449100
2020-09-18  3037.800049  2905.540039  3031.739990  2954.909912  8892600
2020-09-21  2962.000000  2871.000000  2906.500000  2960.469971  6117900
2020-09-22  3133.989990  3000.199951  3033.840088  3128.989990  6948800
2020-09-23  3127.000000  2992.379883  3120.429932  2999.860107  5652700
2020-09-24  3069.300049  2965.000000  2977.790039  3019.790039  5529400
2020-09-25  3101.540039  2999.000000  3054.860107  3095.129883  4615200

              Adj Close
Date
2020-09-01  3499.120117
2020-09-02  3531.449951
2020-09-03  3368.000000
2020-09-04  3294.620117
2020-09-08  3149.840088
2020-09-09  3268.610107
2020-09-10  3175.110107
2020-09-11  3116.219971
2020-09-14  3102.969971
2020-09-15  3156.129883
2020-09-16  3078.100098
2020-09-17  3008.729980
2020-09-18  2954.909912
2020-09-21  2960.469971
2020-09-22  3128.989990
2020-09-23  2999.860107
2020-09-24  3019.790039
2020-09-25  3095.129883
```

In [7]:

```
df.to_csv("amazon.csv") #Converts all the data into comma separated values
```

In [8]:

```
df = pd.read_csv("amazon.csv", parse_dates = True, index_col = 0) #Reads the dataframe previously s
aved as csv, parse the dates, and assign them as the index column {as opposed to the default [0::]
indexing
```

In [9]:

```
print(df.head()) #Test print
```

```
                 High         Low        Open       Close   Volume  \
Date
2010-02-16  120.500000  117.180000  120.059998  117.529999  8932700
2010-02-17  117.129997  115.550003  117.070000  116.309998  8944800
2010-02-18  118.510002  114.820000  115.839996  118.080002  9800100
2010-02-19  119.089996  117.000000  117.910004  117.519997  7115600
2010-02-22  118.970001  116.180000  117.370003  118.010002  6807300

             Adj Close
Date
2010-02-16  117.529999
2010-02-17  116.309998
2010-02-18  118.080002
2010-02-19  117.519997
2010-02-22  118.010002
```

In [10]:

```
df.plot()
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10ee99eb8>
```

In [11]:

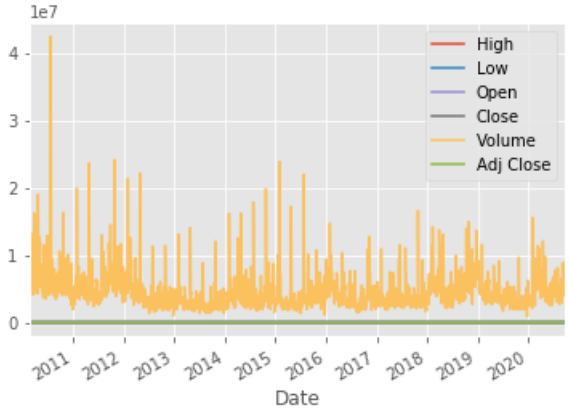
```
print(df["Volume"].head()) #Testing a plot call for simply the Volume column and only asking for t
```

```
he oldest five data points
```

```
Date
2010-02-16    8932700
2010-02-17    8944800
2010-02-18    9800100
2010-02-19    7115600
2010-02-22    6807300
Name: Volume, dtype: int64
```

In [12]:

```
plt.show()
```



In [13]:

```
df["90ma"] = df["Adj Close"].rolling(window=90).mean() #Calculates the moving average over 90 days
```

In [14]:

```
print(df.head()) #Test print which should NaN at the column "90ma" for all of the 5 rows printed {
it should do the same for the 90 first rows}
```

```
                 High         Low        Open       Close   Volume  \
Date
2010-02-16  120.500000  117.180000  120.059998  117.529999  8932700
2010-02-17  117.129997  115.550003  117.070000  116.309998  8944800
2010-02-18  118.510002  114.820000  115.839996  118.080002  9800100
2010-02-19  119.089996  117.000000  117.910004  117.519997  7115600
2010-02-22  118.970001  116.180000  117.370003  118.010002  6807300

             Adj Close  90ma
Date
2010-02-16  117.529999   NaN
2010-02-17  116.309998   NaN
2010-02-18  118.080002   NaN
2010-02-19  117.519997   NaN
2010-02-22  118.010002   NaN
```

In [15]:

```
df.dropna(inplace=True) #Will simply take out of the data frame call, the days where "90ma" is NaN
```

"""To bypass the .dropna funtion, we can simply set the minimum period required to "calculate" "90ma" to zero. This allows for the column "90ma" to be populated by incremented averages of "Adj Close" from the very first day onwards. {as it reaches 90 days, the intended window calculations will take effect}

The code will look like this : df["90ma"] = df["Adj Close"].rolling(window = 90, min_periods = 0).mean() #Feel free to copy and test it
"""

In [16]:

```
print(df.tail()) #Test Print
```

```
                 High          Low         Open        Close    Volume  \
Date
2020-09-21  2962.000000  2871.000000  2906.500000  2960.469971  6117900
2020-09-22  3133.989990  3000.199951  3033.840088  3128.989990  6948800
2020-09-23  3127.000000  2992.379883  3120.429932  2999.860107  5652700
2020-09-24  3069.300049  2965.000000  2977.790039  3019.790039  5529400
2020-09-25  3101.540039  2999.000000  3054.860107  3095.129883  4615200

             Adj Close         90ma
Date
2020-09-21  2960.469971  2936.810004
2020-09-22  3128.989990  2945.033781
2020-09-23  2999.860107  2951.590226
2020-09-24  3019.790039  2958.185004
2020-09-25  3095.129883  2965.360558
```

''' Let us now create two subplots on a 6x1 grid

- their second variables indicate where they start
- the function sharex aligns ax2's x-axis to ax1's '''

In [17]:

```
ax1 = plt.subplot2grid((6, 1), (0, 0), rowspan = 5, colspan = 1)
ax2 = plt.subplot2grid((6, 1), (5, 0), rowspan = 1, colspan = 1, sharex = ax1)
```

In [18]:

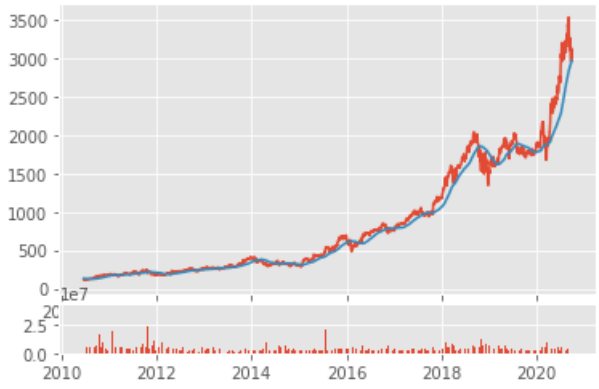
```
ax1.plot(df.index, df["Adj Close"]) #Adj Close on the first axis
ax1.plot(df.index, df["90ma"]) #90ma on the first axis
ax2.bar(df.index, df["Volume"]) #Volume on the second axis
```

Out[18]:

```
<BarContainer object of 2584 artists>
```

In [19]:

```
plt.show()
```



''' The coming part brings us all the way back to when we read and parsed the "amazon.csv" file. We are going to explore resampling {depending of the time span of our liking as to adjust data granularity} using candlestick_ohlc from the python module matplotlib.finance {we are going to set up the import at the top of the script}. We'll use the module on "Adj Close". We'll resample to every 18 days of data.

It is worth noting that appropriate OHLC data is necessary. Splits in company history, for example, can render such an analysis bumby. '''

In [20]:

```
df_ohlc = df["Adj Close"].resample("18D").ohlc() #Defining OHLC data with resampled data points fo
r every 45 minutes
df_volume = df["Volume"].resample("18D").sum() #Resampling the volume just the same but asking for
```

```
a ".sum()" as to reflect the actual {true} volume
```

In [21]:

```
print(df_ohlc.head()) #Test print of data points
```

```
                open        high         low       close
Date
2010-06-23  121.449997  121.449997  108.610001  117.260002
2010-07-11  119.510002  123.650002  117.129997  117.129997
2010-07-29  116.860001  130.000000  116.860001  124.690002
2010-08-16  126.070000  135.210007  123.790001  135.210007
2010-09-03  138.789993  151.300003  137.220001  151.300003
```

In [22]:

```
'''
Reminder that we had already set the dates as indexes; however, with ohlc, we no longer want that.
We'll simply reset the indexes to the default [0::] format.
'''
```

Out[22]:

```
"\nReminder that we had already set the dates as indexes; however, with ohlc, we no longer want th
at. We'll simply reset the indexes to the default [0::] format.\n"
```

In [23]:

```
df_ohlc.reset_index(inplace = True)
```

In [24]:

```
print(df_ohlc.head()) #Verification
```

```
        Date        open        high         low       close
0 2010-06-23  121.449997  121.449997  108.610001  117.260002
1 2010-07-11  119.510002  123.650002  117.129997  117.129997
2 2010-07-29  116.860001  130.000000  116.860001  124.690002
3 2010-08-16  126.070000  135.210007  123.790001  135.210007
4 2010-09-03  138.789993  151.300003  137.220001  151.300003
```

In [25]:

```
df_ohlc["Date"] = df_ohlc["Date"].map(mdates.date2num) #Map out the dates after converting our curr
ent date time objects to matplotlib's mdates
```

In [26]:

```
print(df_ohlc.head()) #Test print
```

```
       Date        open        high         low       close
0  733946.0  121.449997  121.449997  108.610001  117.260002
1  733964.0  119.510002  123.650002  117.129997  117.129997
2  733982.0  116.860001  130.000000  116.860001  124.690002
3  734000.0  126.070000  135.210007  123.790001  135.210007
4  734018.0  138.789993  151.300003  137.220001  151.300003
```

In [27]:

```
ax1.xaxis_date() #Convert m_dates back to beautiful dates
```

In [28]:

```
print(df_ohlc.tail()) #Test print
```

```
         Date         open         high          low        close
```

```
204  737618.0  3104.000000  3196.840088  2961.969971  3033.530029
205  737636.0  3051.879883  3225.000000  3051.879883  3148.020020
206  737654.0  3182.409912  3531.449951  3182.409912  3368.000000
207  737672.0  3294.620117  3294.620117  2954.909912  2960.469971
208  737690.0  3128.989990  3128.989990  2999.860107  3095.129883
```

In [29]:

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

Out[29]:

```
<matplotlib.collections.PolyCollection at 0x113ca6ac8>
```

In [30]:

```
plt.show()
```