GKBoginyFreyaBank

February 13, 2023

1 Chapter I: Analyzing World' Stock Indices

Source:

- https://towardsdatascience.com/analyzing-world-stock-indices-performance-in-python-610df6a578f

The stock index is a list of selected companies, where its averaged price (or weighted-average) reflects the stock market. Stock indices also may reflect a certain industry or region that they cover. They are also often referred to as a benchmark to measure the performance of funds. The more important part of the stock index than its current price is the performance or price changes to a definitive previous moment, whether it is the day or three months before.

Why we compare indexes from all around the world? After reading the end of Chapter 7 of The Intelligent Investor, the author said that we need to allocate few money in foreign land / foreign bonds / foreign stocks. Enlarge your knowledge, unlimit your world.

Obtain the Financial Data Data related to stock indices can be retrieved from Yahoo! Finance. In Python, there has been a popular module to retrieve data more easily from Yahoo! Finance. If it has not been installed yet, you can install it in your Jupyter Notebook by typing this chunk of codes, followed by importing the needed libraries/modules for processing.

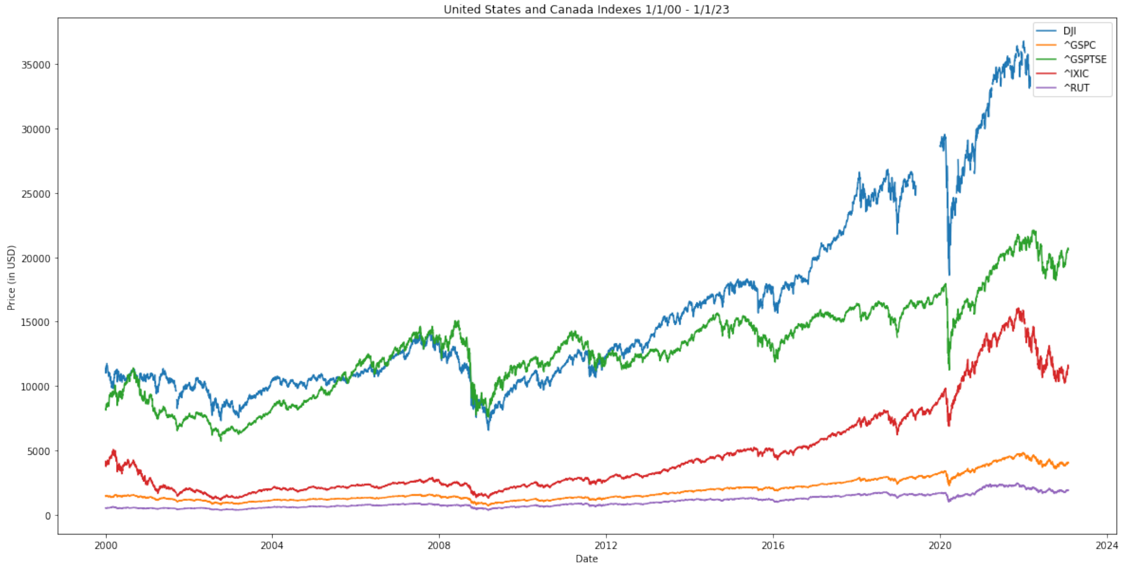
1.1 US and Canada Indexes

[105]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^GSPC', 'DJI', '^IXIC', '^RUT', '^GSPTSE']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()
```

```
plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)

plt.xlabel("Date")
plt.ylabel("Price (in USD)")
plt.title("United States and Canada Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[*********************100%***********************]  5 of 5 completed
```

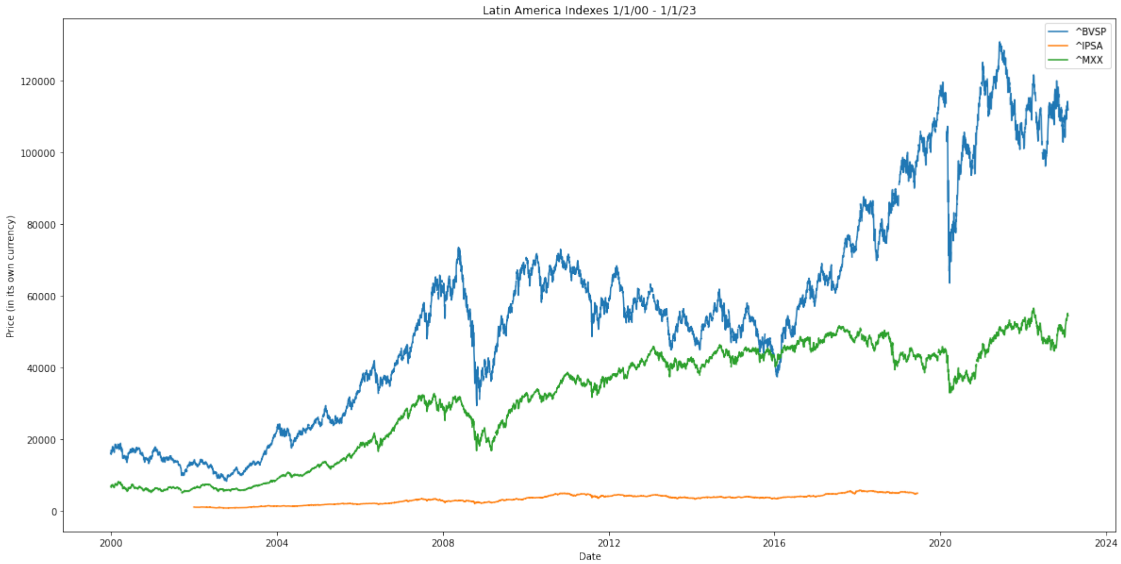


1.2 Latin America Indexes

[108]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^BVSP', '^MXX', '^IPSA']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)

plt.xlabel("Date")
plt.ylabel("Price (in its own currency)")
plt.title("Latin America Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[*********************100%***********************]  3 of 3 completed
```

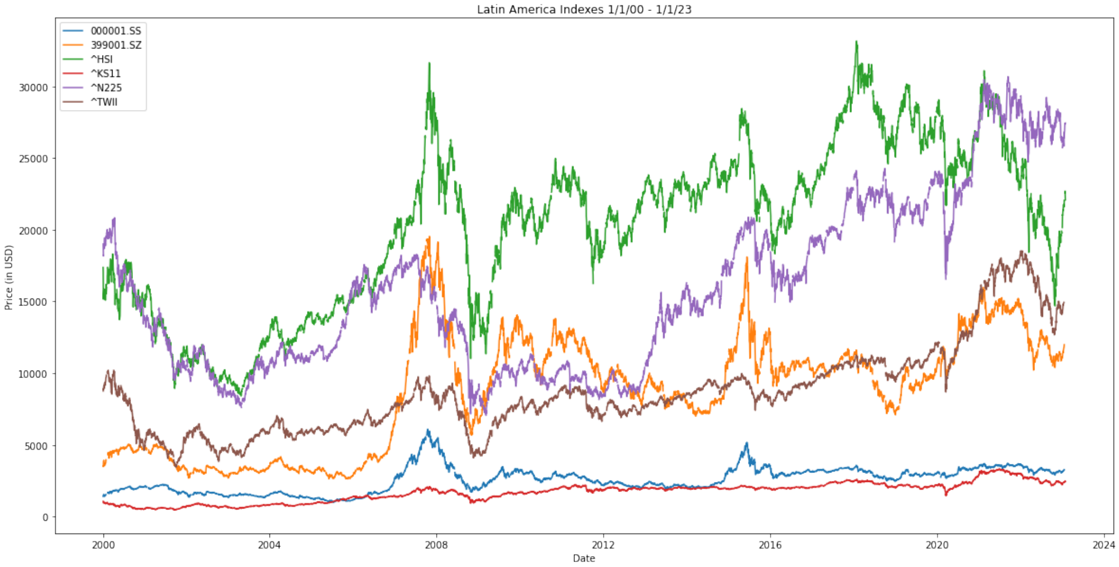


1.3 East Asia Indexes

[107]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^N225', '^HSI', '000001.SS', '399001.SZ', '^TWII', '^KS11']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)
plt.xlabel("Date")
plt.ylabel("Price (in its own currency)")
plt.title("Latin America Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

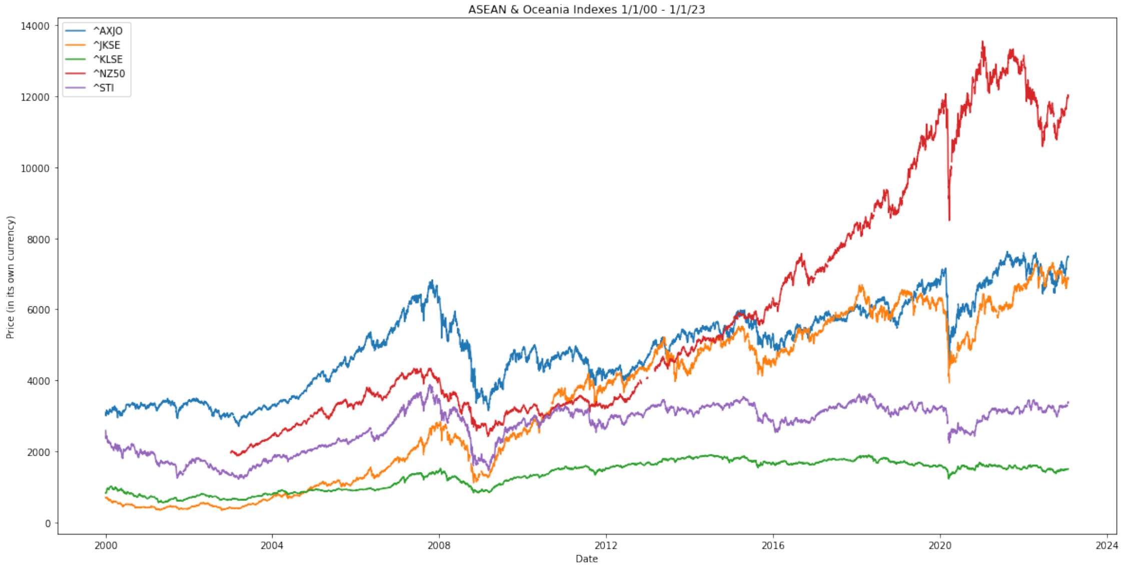
```
[*********************100%***********************]  6 of 6 completed
```



1.4 ASEAN & Oceania Indexes

[109]:
```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^STI', '^JKSE', '^KLSE','^AXJO',  '^NZ50']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)
plt.xlabel("Date")
plt.ylabel("Price (in its own currency)")
plt.title("ASEAN & Oceania Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[*********************100%***********************]  5 of 5 completed
```

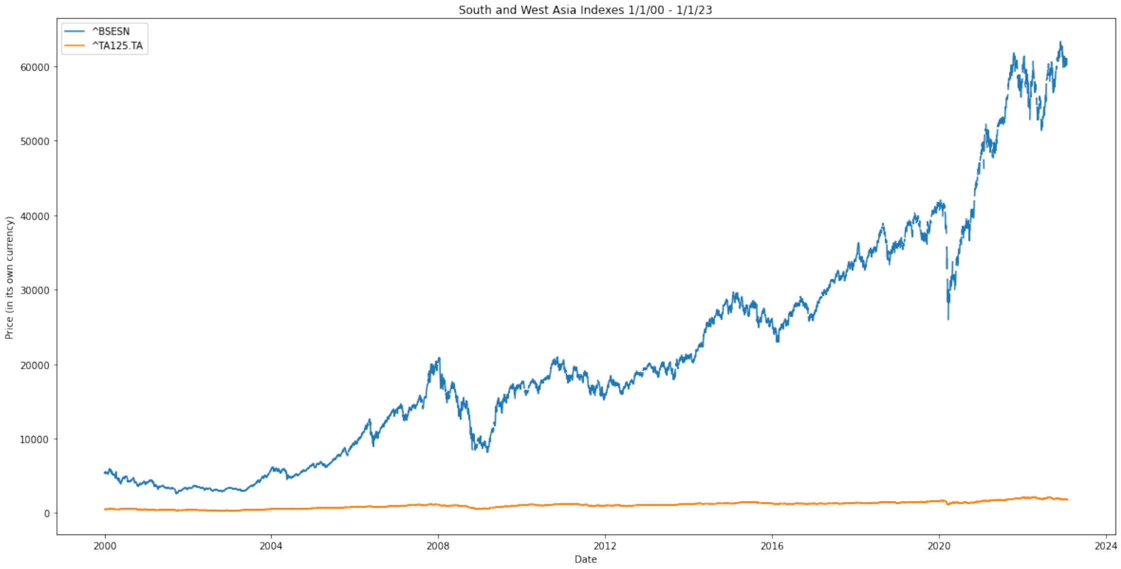


1.5 South and West Asia Indexes

[110]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^BSESN', '^TA125.TA']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)

plt.xlabel("Date")
plt.ylabel("Price (in its own currency)")
plt.title("South and West Asia Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[*********************100%***********************]  2 of 2 completed
```

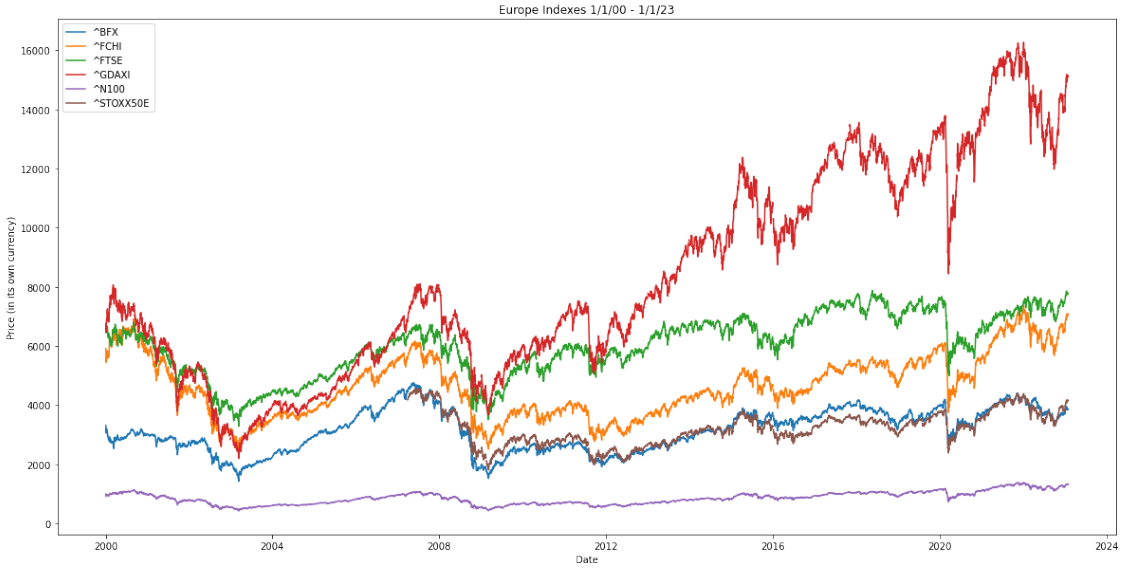


1.6 Europe Indexes

[112]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['^FTSE', '^GDAXI', '^FCHI', '^STOXX50E','^N100', '^BFX']
from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

#print(data)
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)

plt.xlabel("Date")
plt.ylabel("Price (in its own currency)")
plt.title("Europe Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[*********************100%***********************]  6 of 6 completed
```



2 Chapter II: Comparing Multiple Corporation Stocks Price and Using Indicator for Single Stocks Price

Sources: * https://tcoil.info/plot-multiple-stocks-in-python/

- https://medium.com/wealthy-bytes/visualizing-free-stock-data-for-algorithmic-trading-with-python-and-matplotlib-dca1abbd286c
- https://pythoninoffice.com/draw-stock-chart-with-python/

In this section we are going to:

1. Plot multiple corporations stocks price to compare their performance against their competitor within an industry
2. Plot a single stocks price, to put all your eggs in a single basket, after choosing your winning corporation, all in.
3. Plot a single stocks price, with indicator Moving Average, the average price of the stocks in 50 days and 200 days
4. Plot a single stocks price, with indicator of Bollinger Bands. It is quite a strong indicator that trader will love.

5. Plot a single stocks price, with volume indicator. To make sure the corporation' stocks is liquid enough.
6. Plot a single stocks price, with candlestick style.
7. Plot a single stocks price, with candlestick style and 20 Days Moving Average.
8. Plot a single stocks price, with candlestick style and 20 Days Moving Average that hiding non-trading days.

Why comparing? we need to compare stock performance between each other or against the index during specific time interval to choose the one we want to invest in.

Very useful for comparing not only stocks between each other, but also major indexes (or ETFs) in the world, or commodities prices.

2.1 Multiple Plots using Pandas and yfinance

[88]:

```
from pandas_datareader import data as pdr
import yfinance as yf

yf.pdr_override()
y_symbols = ['EA', 'TTWO', 'ATVI']

from datetime import datetime
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)
```

```
[*********************100%***********************]  3 of 3 completed
```

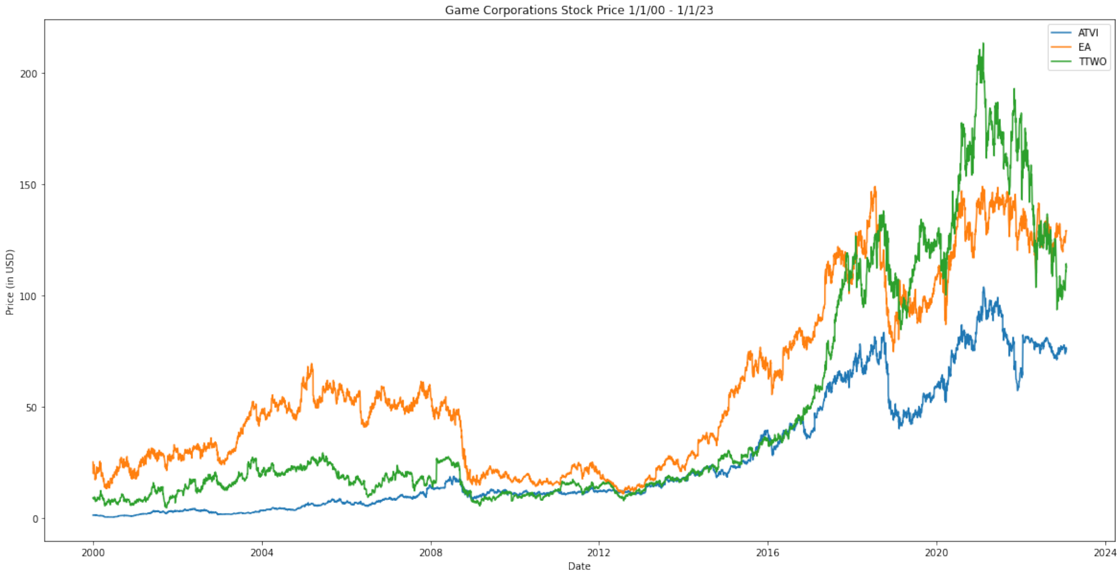
[89]:

```
print(data)
```

```
            Adj Close                                  Close              \
                 ATVI          EA        TTWO       ATVI          EA
Date
2000-01-03   1.214421   24.965612    9.124436   1.369792   25.265625
2000-01-04   1.177480   22.078434    8.874452   1.328125   22.343750
2000-01-05   1.182098   22.062996    8.832788   1.333333   22.328125
2000-01-06   1.159010   19.762514    8.749459   1.307292   20.000000
2000-01-07   1.191333   20.349215    8.999444   1.343750   20.593750
…                 …         …         …        …         …
2023-01-24  75.110001  127.489998  111.269997  75.110001  127.489998
2023-01-25  74.639999  127.559998  110.699997  74.639999  127.559998
2023-01-26  75.599998  129.139999  111.889999  75.599998  129.139999
2023-01-27  76.610001  128.869995  114.279999  76.610001  128.869995
2023-01-30  75.959999  128.990005  112.660004  75.959999  128.990005

                             High                                Low  \
                  TTWO       ATVI          EA        TTWO       ATVI
Date
2000-01-03    9.125000   1.375000   28.765625   10.000000   1.166667
2000-01-04    8.875000   1.354167   24.500000    9.333333   1.187500
2000-01-05    8.833333   1.364583   24.375000    8.875000   1.312500
2000-01-06    8.750000   1.333333   22.625000    9.000000   1.296875
2000-01-07    9.000000   1.354167   21.656250    9.041667   1.291667
…                  …         …           …          …         …
2023-01-24  111.269997  75.430000  128.070007  111.660004  74.500000
2023-01-25  110.699997  75.110001  127.650002  111.389999  74.529999
2023-01-26  111.889999  75.660004  129.449997  112.250000  74.650002
2023-01-27  114.279999  76.760002  130.570007  115.339996  75.220001
2023-01-30  112.660004  77.080002  129.470001  114.540001  75.839996

                                                  Open                      \
                    EA        TTWO       ATVI          EA        TTWO
Date
2000-01-03   20.593750    8.666667   1.312500   21.250000    8.916667
2000-01-04   22.312500    8.666667   1.343750   23.750000    9.083333
2000-01-05   21.515625    8.333333   1.317708   22.000000    8.750000
2000-01-06   19.843750    8.500000   1.322917   22.062500    8.791667
2000-01-07   20.312500    8.375000   1.322917   21.015625    8.666667
…                  …           …         …          …           …
2023-01-24  126.370003  109.050003  75.000000  127.709999  110.470001
2023-01-25  126.269997  109.269997  75.000000  126.589996  109.589996
2023-01-26  128.190002  110.559998  74.790001  128.309998  111.849998
2023-01-27  128.789993  113.360001  75.500000  129.139999  114.099998
2023-01-30  128.110001  112.339996  76.629997  128.919998  113.099998

              Volume
                ATVI       EA     TTWO
Date
2000-01-03   7226400  9040800  1176750
2000-01-04   4262400  6331200   345300
2000-01-05   3390000  5072000   628800
2000-01-06   2430000  6408400   374100
2000-01-07  15549600  5456400   482850
…                …        …       …
2023-01-24   5069600  1301800  1245900
2023-01-25   4004300  1099800  1821200
2023-01-26   3960800  1196100  1252900
2023-01-27   4381700  1786200  1864900
2023-01-30   4247400  2446900  1368800

[5806 rows x 18 columns]
```

[96]:

```
#data['Close'].plot()

plt.figure(figsize=(20,10))
plt.plot(data.index, data['Close'], label=data["Close"].columns)

plt.xlabel("Date")
plt.ylabel("Price (in USD)")
plt.title("Game Corporations Stock Price 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```



2.2 Single Plot using Pandas and yfinance

[67]:

```
import datetime as dt
import yfinance as yf

company = 'AAPL'

# Define a start date and End Date
start = dt.datetime(2000,1,1)
end =  dt.datetime(2023,1,31)

# Read Stock Price Data
data = yf.download(company, start , end)

data.tail(10)
#print(data)
```

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

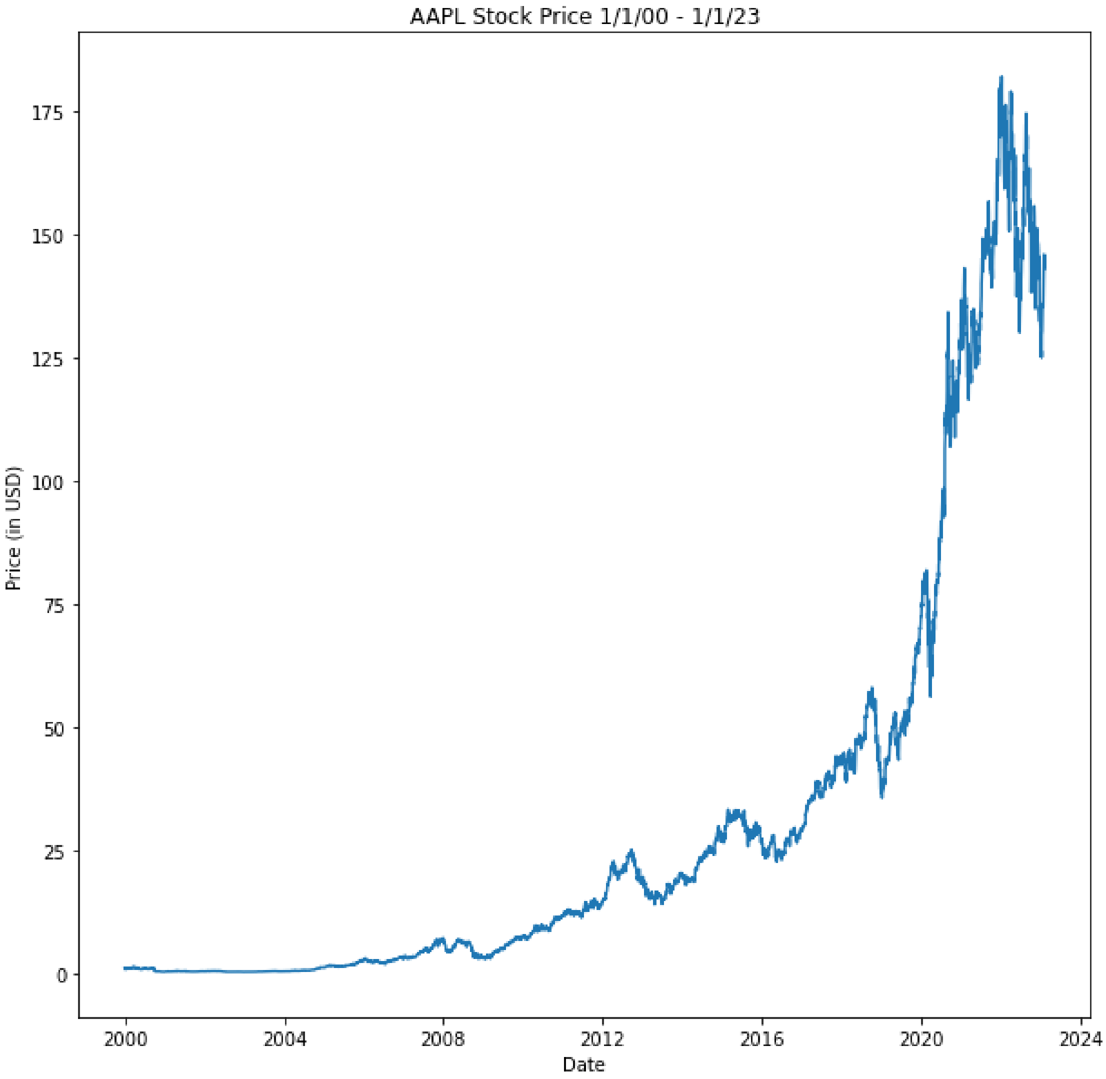
[67]:

```
                  Open        High         Low       Close   Adj Close  \
Date
2023-01-17  134.830002  137.289993  134.130005  135.940002  135.732758
2023-01-18  136.820007  138.610001  135.029999  135.210007  135.003876
2023-01-19  134.080002  136.250000  133.770004  135.270004  135.063782
2023-01-20  135.279999  138.020004  134.220001  137.869995  137.659805
2023-01-23  138.119995  143.320007  137.899994  141.110001  140.894882
2023-01-24  140.309998  143.160004  140.300003  142.529999  142.312714
2023-01-25  140.889999  142.429993  138.809998  141.860001  141.643738
2023-01-26  143.169998  144.250000  141.899994  143.960007  143.740540
2023-01-27  143.160004  147.229996  143.080002  145.929993  145.707520
2023-01-30  144.960007  145.550003  142.850006  143.000000  142.781998

              Volume
Date
2023-01-17  63646600
2023-01-18  69672800
2023-01-19  58280400
2023-01-20  79972200
2023-01-23  81760300
2023-01-24  66435100
2023-01-25  65799300
2023-01-26  54105100
2023-01-27  70492800
2023-01-30  64015300
```

[68]:

```
#data['Close'].plot()

plt.figure(figsize=(10,10))
plt.plot(data.index, data['Close'])
plt.xlabel("Date")
plt.ylabel("Price (in USD)")
plt.title("AAPL Stock Price 1/1/00 - 1/1/23")
```

```
[68]: Text(0.5, 1.0, 'AAPL Stock Price 1/1/00 - 1/1/23')
```

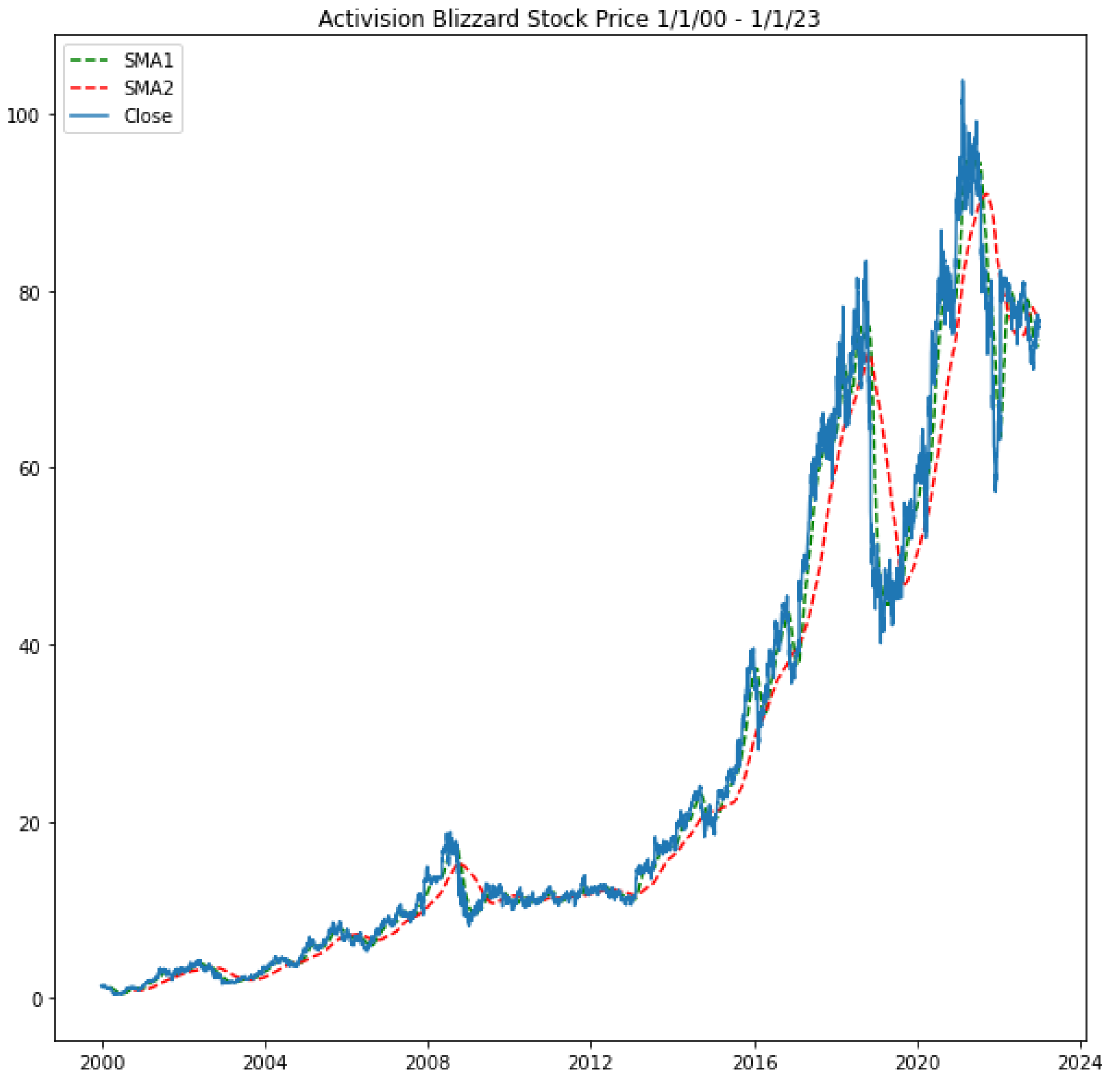


2.3 Single Plot using Pandas and yfinance with Moving Average

```
[84]: import datetime as dt
import yfinance as yf

company = 'ATVI'
```

```python
# Define a start date and End Date
start = dt.datetime(2000,1,1)
end =  dt.datetime(2023,1,1)

# Read Stock Price Data
data = yf.download(company, start , end)

#data.tail(10)
#print(data)

# Creating and Plotting Moving Averages
data["SMA1"] = data['Close'].rolling(window=50).mean()
data["SMA2"] = data['Close'].rolling(window=200).mean()
data['ewma'] = data['Close'].ewm(halflife=0.5, min_periods=20).mean()

plt.figure(figsize=(10,10))
plt.plot(data['SMA1'], 'g--', label="SMA1")
plt.plot(data['SMA2'], 'r--', label="SMA2")
plt.plot(data['Close'], label="Close")
plt.title("Activision Blizzard Stock Price 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

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



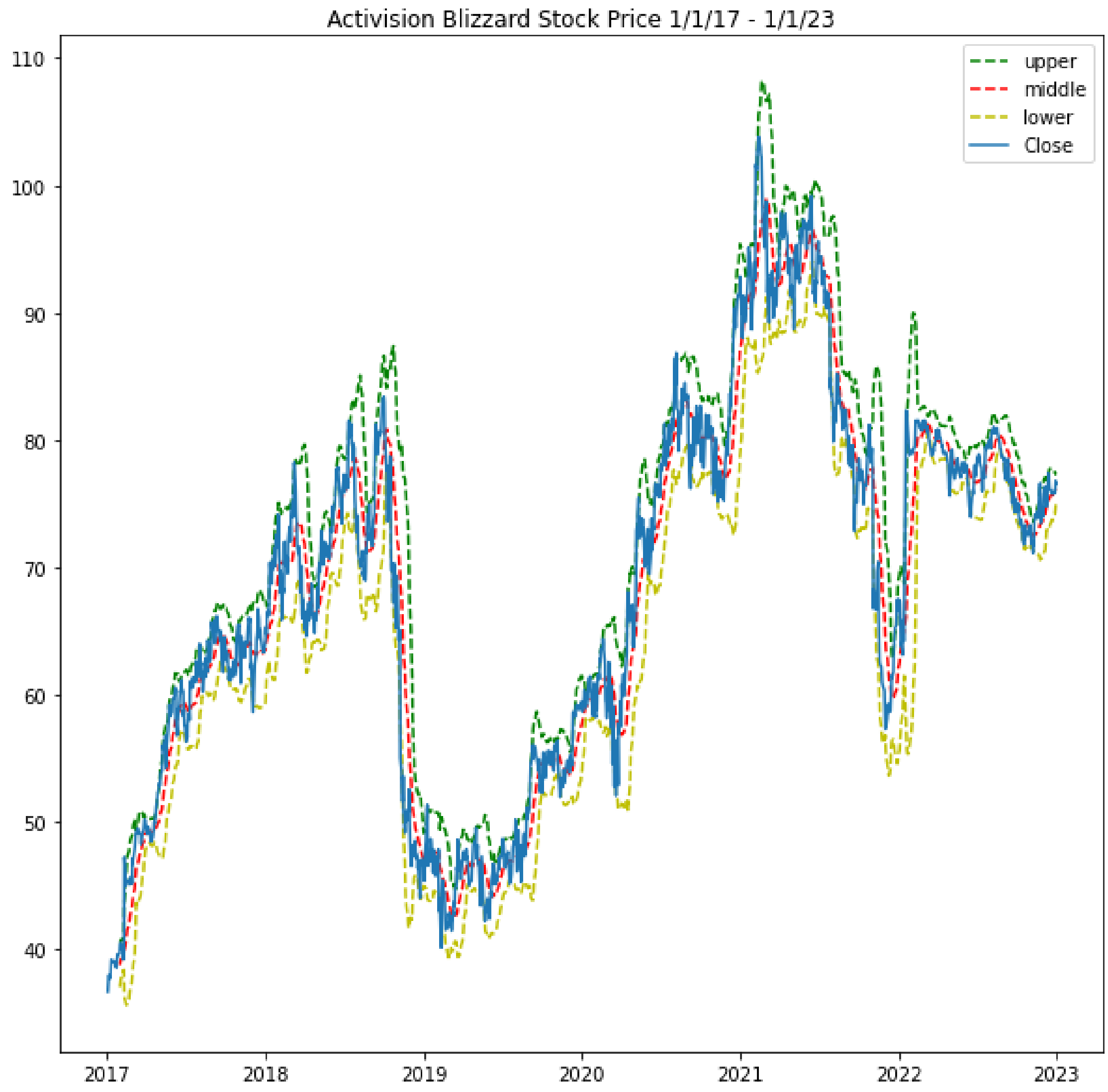
2.4 Single Plot using Pandas and yfinance with Bollinger Bands

[86]:

```
import datetime as dt
import yfinance as yf

company = 'ATVI'

# Define a start date and End Date
start = dt.datetime(2017,1,1)
end =  dt.datetime(2023,1,1)

# Read Stock Price Data
```

```python
data = yf.download(company, start , end)

#data.tail(10)
#print(data)

# Creating and Plotting Bollinger Bands
data['middle_band'] = data['Close'].rolling(window=20).mean()
data['upper_band'] = data['Close'].rolling(window=20).mean() + data['Close'].rolling(window=20).std()*2
data['lower_band'] = data['Close'].rolling(window=20).mean() - data['Close'].rolling(window=20).std()*2

plt.figure(figsize=(10,10))
plt.plot(data['upper_band'], 'g--', label="upper")
plt.plot(data['middle_band'], 'r--', label="middle")
plt.plot(data['lower_band'], 'y--', label="lower")
plt.plot(data['Close'], label="Close")
plt.title("Activision Blizzard Stock Price 1/1/17 - 1/1/23")
plt.legend()
plt.show()
```

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



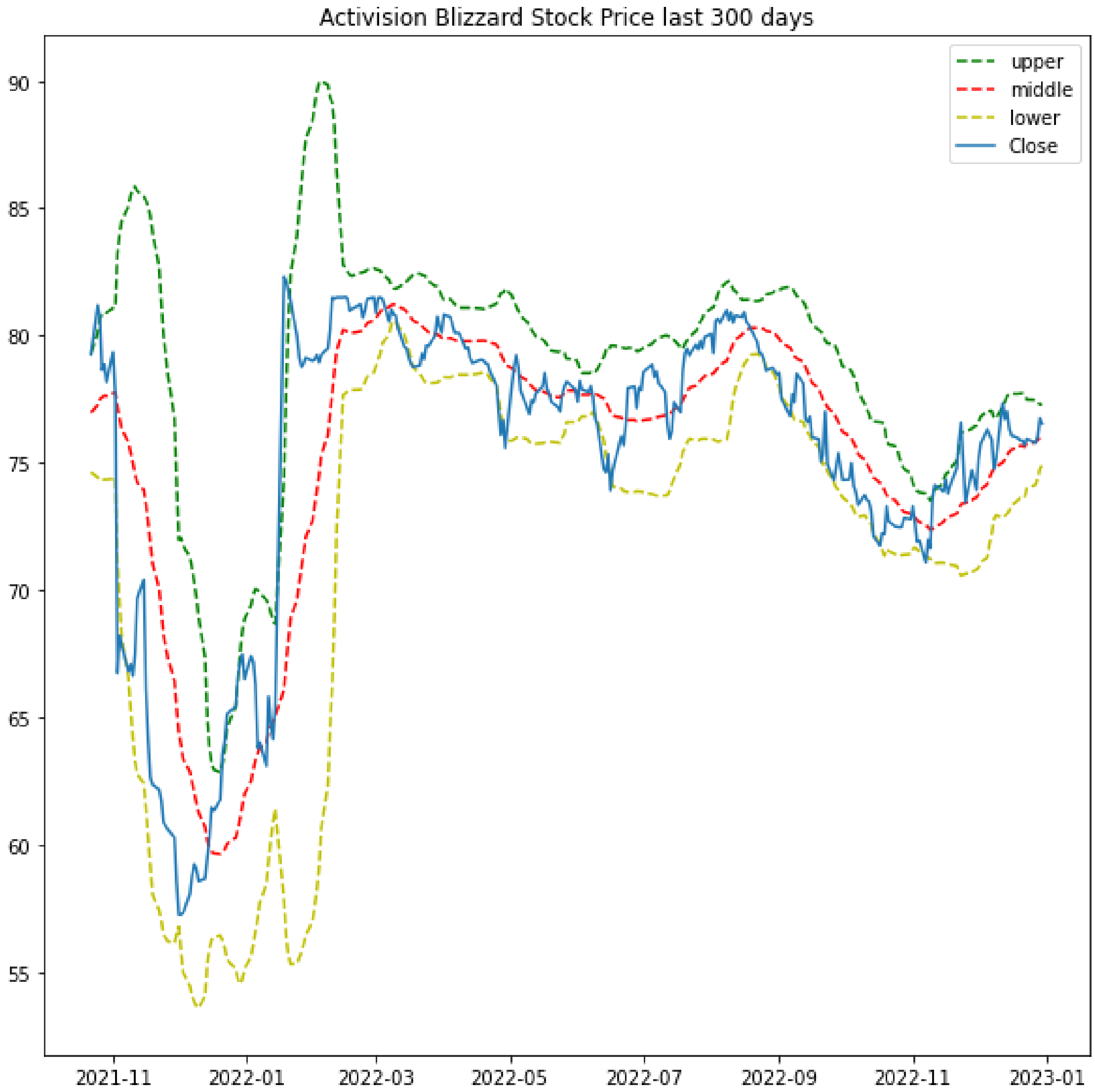
Bollinger Bands that zooms in on the past 300 days of trading

[102]:

```
import datetime as dt
import yfinance as yf

company = 'ATVI'

# Define a start date and End Date
start = dt.datetime(2017,1,1)
end =  dt.datetime(2023,1,1)

# Read Stock Price Data
data = yf.download(company, start , end)
```

```
#data.tail(10)
#print(data)

# Creating and Plotting Bollinger Bands
data['middle_band'] = data['Close'].rolling(window=20).mean()
data['upper_band'] = data['Close'].rolling(window=20).mean() + data['Close'].
 ↪rolling(window=20).std()*2
data['lower_band'] = data['Close'].rolling(window=20).mean() - data['Close'].
 ↪rolling(window=20).std()*2

plt.figure(figsize=(10,10))
plt.plot(data['upper_band'].iloc[-300:], 'g--', label="upper")
plt.plot(data['middle_band'].iloc[-300:], 'r--', label="middle")
plt.plot(data['lower_band'].iloc[-300:], 'y--', label="lower")
plt.plot(data['Close'].iloc[-300:], label="Close")
plt.title("Activision Blizzard Stock Price last 300 days")
plt.legend()
plt.show()
```

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



Codes not working (Check)

[39]:

```
import numpy as np
import pandas as pd
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
import yfinance as yf

stocks = [
    {
        'ticker': 'UU.L',
        'name': 'United Utilities'
    },
    {
        'ticker': 'VOD.L',
        'name': 'Vodafone Group'
    },
    {
        'ticker': 'BP.L',
        'name': 'BP Group'
    }
]

def create_plots(stocks):
    data = pd.DataFrame()
    for stock in stocks:
        data[stock['ticker']] = yf.download(stock['ticker'],␣
 ↪data_source='yahoo', start='2007-1-1')['Adj Close']
    returns = data.apply(lambda x: (x / x[0] * 100))

    plt.figure(figsize=(10,6))

    for stock in stocks:
        plt.plot(returns[stock['ticker']], label=stock['name'])
    plt.legend()
    plt.ylabel('Cumulative Returns %')
    plt.xlabel('Time')

    plt.show
```

2.5 Single Plot using Pandas and yfinance with Volume

[116]:

```
import yfinance
stockvol = yfinance.Ticker('CAT')
hist = stockvol.history(period='3y')
```

[117]:

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(x=hist.index,y=hist['Close'],␣
 ↪mode='lines+markers'))
fig.show()
```

[120]:

```
from plotly.subplots import make_subplots

fig2 = make_subplots(specs=[[{"secondary_y": True}]])
fig2.add_trace(go.Scatter(x=hist.
 ↪index,y=hist['Close'],name='Price'),secondary_y=False)
```

```
fig2.add_trace(go.Bar(x=hist.
 ↪index,y=hist['Volume'],name='Volume'),secondary_y=True)

#fig2.update_yaxes(range=[0,7000000000],secondary_y=True)
#fig2.update_yaxes(visible=False, secondary_y=True)

fig2.show()
```

2.6 Single Candlestick Plot using Pandas and yfinance

[127]:

```
import yfinance
stockvol = yfinance.Ticker('CAT')
hist = stockvol.history(period='3y')

import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(x=hist.index,y=hist['Close'],␣
 ↪mode='lines+markers'))

fig2 = make_subplots(specs=[[{"secondary_y": True}]])
fig2.add_trace(go.Scatter(x=hist.
 ↪index,y=hist['Close'],name='Price'),secondary_y=False)
fig2.add_trace(go.Bar(x=hist.
 ↪index,y=hist['Volume'],name='Volume'),secondary_y=True)

fig2.update_yaxes(range=[0,7000000000],secondary_y=True)
fig2.update_yaxes(visible=False, secondary_y=True)

fig3 = make_subplots(specs=[[{"secondary_y": True}]])
fig3.add_trace(go.Candlestick(x=hist.index,
                              open=hist['Open'],
                              high=hist['High'],
                              low=hist['Low'],
                              close=hist['Close'],
                             ))
fig3.show()

#fig3.add_trace(go.Bar(x=hist.index, y=hist['Volume'],␣
 ↪name='Volume'),secondary_y=True)
#fig3.update_layout(xaxis_rangeslider_visible=False)
```

2.7 Single Candlestick Plot using Pandas and yfinance with 20 Days MA

[128]:

```
import yfinance
stockvol = yfinance.Ticker('CAT')
hist = stockvol.history(period='3y')

import plotly.graph_objects as go

fig3.add_trace(go.Scatter(x=hist.index,y=hist['Close'].rolling(window=20).
 ↪mean(),marker_color='blue',name='20 Day MA'))
fig3.add_trace(go.Bar(x=hist.index, y=hist['Volume'],␣
 ↪name='Volume'),secondary_y=True)
fig3.update_layout(title={'text':'CAT', 'x':0.5})
fig3.update_yaxes(range=[0,1000000000],secondary_y=True)
fig3.update_yaxes(visible=False, secondary_y=True)
fig3.update_layout(xaxis_rangeslider_visible=False)  #hide range slider
fig3.show()

#hist['diff'] = hist['Close'] - hist['Open']
#hist.loc[hist['diff']>=0, 'color'] = 'green'
#hist.loc[hist['diff']<0, 'color'] = 'red'
```

2.8 Single Candlestick Plot using Pandas and yfinance with 20 Days MA + Hide Non-trading Days

[134]:

```
import yfinance
stockvol = yfinance.Ticker('CAT')
hist = stockvol.history(period='1y')

import plotly.graph_objects as go

hist['diff'] = hist['Close'] - hist['Open']
hist.loc[hist['diff']>=0, 'color'] = 'green'
hist.loc[hist['diff']<0, 'color'] = 'red'

fig3 = make_subplots(specs=[[{"secondary_y": True}]])
fig3.add_trace(go.Candlestick(x=hist.index,
                              open=hist['Open'],
                              high=hist['High'],
                              low=hist['Low'],
                              close=hist['Close'],
                              name='Price'))
fig3.add_trace(go.Scatter(x=hist.index,y=hist['Close'].rolling(window=20).
 ↪mean(),marker_color='blue',name='20 Day MA'))
```

```
fig3.add_trace(go.Bar(x=hist.index, y=hist['Volume'], name='Volume',␣
 ↪marker={'color':hist['color']}),secondary_y=True)
fig3.update_yaxes(range=[0,700000000],secondary_y=True)
fig3.update_yaxes(visible=False, secondary_y=True)
fig3.update_layout(xaxis_rangeslider_visible=False)  #hide range slider
fig3.update_layout(title={'text':'CAT', 'x':0.5})
#fig3.show()

fig3.update_xaxes(rangebreaks = [
                       dict(bounds=['sat','mon']), # hide weekends
                       #dict(bounds=[16, 9.5], pattern='hour'), # for hourly␣
 ↪chart, hide non-trading hours (24hr format)
                       dict(values=["2021-12-25","2022-01-01"]) #hide Xmas and␣
 ↪New Year
                                ])
```

Codes not working (Check)

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf

# Retrieving List of World Major Stock Indices from Yahoo! Finance
df_list = pd.read_html('https://finance.yahoo.com/world-indices/')
majorStockIdx = df_list[0]
majorStockIdx.head()
```

[1]:
```
  Symbol                          Name  Last Price  Change % Change    Volume  \
0  ^GSPC                       S&P 500     4090.46    8.96   +0.22%    2.312B
1   ^DJI  Dow Jones Industrial Average    33869.27  169.37   +0.50%  289.725M
2  ^IXIC             NASDAQ Composite     11718.12  -71.48   -0.61%    4.223B
3   ^NYA           NYSE COMPOSITE (DJ)    15910.69   82.09   +0.52%         0
4   ^XAX      NYSE AMEX COMPOSITE INDEX    4344.55  126.93   +3.01%         0

  Intraday High/Low  52 Week Range  Day Chart
0               NaN            NaN        NaN
1               NaN            NaN        NaN
2               NaN            NaN        NaN
3               NaN            NaN        NaN
4               NaN            NaN        NaN
```

[2]:
```
# ^GSPC is the symbol of S&P 500 in Yahoo! Finance
# the data will be retrieved daily ('1d') and started from 1 January 2020 until␣
 ↪30 September 2020.
```

```
tickerData = yf.Ticker('^GSPC')
tickerDf1 = tickerData.history(period='1d', start='2010-1-1', end='2020-10-1')
```

[3]:

```
tickerData.history(period='1d', start='2010-1-1', end='2020-10-1')
```

[3]:

```
                                  Open         High          Low        Close  \
Date
2010-01-04 00:00:00-05:00  1116.560059  1133.869995  1116.560059  1132.989990
2010-01-05 00:00:00-05:00  1132.660034  1136.630005  1129.660034  1136.520020
2010-01-06 00:00:00-05:00  1135.709961  1139.189941  1133.949951  1137.140015
2010-01-07 00:00:00-05:00  1136.270020  1142.459961  1131.319946  1141.689941
2010-01-08 00:00:00-05:00  1140.520020  1145.390015  1136.219971  1144.979980
…                                  …          …          …          …
2020-09-24 00:00:00-04:00  3226.139893  3278.699951  3209.449951  3246.590088
2020-09-25 00:00:00-04:00  3236.659912  3306.879883  3228.439941  3298.459961
2020-09-28 00:00:00-04:00  3333.899902  3360.739990  3332.909912  3351.600098
2020-09-29 00:00:00-04:00  3350.919922  3357.919922  3327.540039  3335.469971
2020-09-30 00:00:00-04:00  3341.209961  3393.560059  3340.469971  3363.000000

                               Volume  Dividends  Stock Splits
Date
2010-01-04 00:00:00-05:00  3991400000        0.0           0.0
2010-01-05 00:00:00-05:00  2491020000        0.0           0.0
2010-01-06 00:00:00-05:00  4972660000        0.0           0.0
2010-01-07 00:00:00-05:00  5270680000        0.0           0.0
2010-01-08 00:00:00-05:00  4389590000        0.0           0.0
…                                 …        …           …
2020-09-24 00:00:00-04:00  4601920000        0.0           0.0
2020-09-25 00:00:00-04:00  3803330000        0.0           0.0
2020-09-28 00:00:00-04:00  3950910000        0.0           0.0
2020-09-29 00:00:00-04:00  3661590000        0.0           0.0
2020-09-30 00:00:00-04:00  4738640000        0.0           0.0

[2705 rows x 7 columns]
```

[4]:

```
stock_list = []
for s in majorStockIdx.Symbol: # iterate for every stock indices
    # Retrieve data from Yahoo! Finance
    tickerData = yf.Ticker(s)
    tickerDf1 = tickerData.history(period='1d', start='2010-1-1', end='2020-9-30')
    # Save historical data
    tickerDf1['ticker'] = s # don't forget to specify the index
    stock_list.append(tickerDf1)
# Concatenate all data
msi = pd.concat(stock_list, axis = 0)
```

```
^CASE30: No data found for this date range, symbol may be delisted
```

[5]:
```
pd.concat(stock_list, axis = 0)
```

[5]:
```
                                  Open         High          Low        Close  \
Date
2010-01-04 00:00:00-05:00  1116.560059  1133.869995  1116.560059  1132.989990
2010-01-05 00:00:00-05:00  1132.660034  1136.630005  1129.660034  1136.520020
2010-01-06 00:00:00-05:00  1135.709961  1139.189941  1133.949951  1137.140015
2010-01-07 00:00:00-05:00  1136.270020  1142.459961  1131.319946  1141.689941
2010-01-08 00:00:00-05:00  1140.520020  1145.390015  1136.219971  1144.979980
…                                  …          …          …          …
2020-09-22 00:00:00+02:00  3134.750000  3198.909912  3111.439941  3155.520020
2020-09-23 00:00:00+02:00  3138.000000  3205.540039  3117.360107  3162.409912
2020-09-25 00:00:00+02:00  3193.300049  3193.870117  3066.290039  3105.010010
2020-09-28 00:00:00+02:00  3123.300049  3203.389893  3121.090088  3170.429932
2020-09-29 00:00:00+02:00  3183.360107  3211.629883  3141.090088  3186.250000

                                 Volume  Dividends  Stock Splits    ticker  \
Date
2010-01-04 00:00:00-05:00  3.991400e+09        0.0           0.0     ^GSPC
2010-01-05 00:00:00-05:00  2.491020e+09        0.0           0.0     ^GSPC
2010-01-06 00:00:00-05:00  4.972660e+09        0.0           0.0     ^GSPC
2010-01-07 00:00:00-05:00  5.270680e+09        0.0           0.0     ^GSPC
2010-01-08 00:00:00-05:00  4.389590e+09        0.0           0.0     ^GSPC
…                                   …        …           …       …
2020-09-22 00:00:00+02:00  0.000000e+00        0.0           0.0  ^JNOU.JO
2020-09-23 00:00:00+02:00  0.000000e+00        0.0           0.0  ^JNOU.JO
2020-09-25 00:00:00+02:00  0.000000e+00        0.0           0.0  ^JNOU.JO
2020-09-28 00:00:00+02:00  0.000000e+00        0.0           0.0  ^JNOU.JO
2020-09-29 00:00:00+02:00  0.000000e+00        0.0           0.0  ^JNOU.JO

                           Adj Close
Date
2010-01-04 00:00:00-05:00        NaN
2010-01-05 00:00:00-05:00        NaN
2010-01-06 00:00:00-05:00        NaN
2010-01-07 00:00:00-05:00        NaN
2010-01-08 00:00:00-05:00        NaN
…                               …
2020-09-22 00:00:00+02:00        NaN
2020-09-23 00:00:00+02:00        NaN
2020-09-25 00:00:00+02:00        NaN
2020-09-28 00:00:00+02:00        NaN
2020-09-29 00:00:00+02:00        NaN

[90016 rows x 9 columns]
```

[6]:

```
#  categorize each index by the region
region_idx={ 'US & Canada' : ['^GSPC', '^DJI', '^IXIC', '^RUT','^GSPTSE'],
  'Latin America' : ['^BVSP', '^MXX', '^IPSA'],
  'East Asia' : ['^N225', '^HSI', '000001.SS', '399001.SZ', '^TWII', '^KS11'],
  'ASEAN & Oceania' : ['^STI', '^JKSE', '^KLSE','^AXJO',  '^NZ50'],
  'South & West Asia' : ['^BSESN', '^TA125.TA'],
  'Europe' : ['^FTSE', '^GDAXI', '^FCHI', '^STOXX50E','^N100', '^BFX']
}

# make a new column for the region.

def getRegion(ticker):
    for k in region_idx.keys():
        if ticker in region_idx[k]:
            return k
msi['region']= msi.ticker.apply(lambda x: getRegion(x))
```

[ ]:

```
# Get the data for 4 Jan 2010

begRef = msi.loc[msi.index == '2010-01-04']
def retBegin(ticker, val):
    start_val = begRef.loc[begRef.ticker == ticker, 'Close'].values[0]
    return (val/start_val - 1) * 100

msi['chBegin'] = msi.apply(lambda x: retBegin(x.ticker, x.Close), axis = 1)

# Transform the data to be ticker column-wise
chBegin = msi.groupby(['Date', 'ticker'])['chBegin'].first().unstack()
# Fill null values with the values on the row before
chBegin = chBegin.fillna(method='bfill')
```

[ ]:

```
fig, axes = plt.subplots(3,2, figsize=(12, 8),sharex=True)
pagoda = ["#965757", "#D67469", "#4E5A44", "#A1B482", '#EFE482', "#99BFCF"] #␣
 ↪for coloring
for i, k in enumerate(region_idx.keys()):
# Iterate for each region
    ax = axes[int(i/2), int(i%2)]
    for j,t in enumerate(region_idx[k]):
        # Iterate and plot for each stock index in this region
        ax.plot(chBegin.index, chBegin[t], marker='', linewidth=1, color =␣
 ↪pagoda[j])
        ax.legend([ticker[t] for t in region_idx[k]], loc='upper left',␣
 ↪fontsize=7)
        ax.set_title(k, fontweight='bold')
fig.text(0.5,0, "Year", ha="center", va="center", fontweight ="bold")
fig.text(0,0.5, "Price Change/Return (%)", ha="center", va="center",␣
 ↪rotation=90, fontweight ="bold")
```

```
fig.suptitle("Price Change/Return for Major Stock Indices based on 2010",␣
 ↪fontweight ="bold",y=1.05, fontsize=14)
fig.tight_layout()
```

3 Appendix

[ ]:

```
# To activate project designated for GKBoginyFreyaBank
# Create an empty folder named GKBoginyFreyaBank that is in one folder with␣
 ↪this notebook
import Pkg
Pkg.activate("GKBoginyFreyaBank")
```

[ ]:

```
pip list
```

[ ]:

```
pip install yfinance # install yfinance
```

[ ]:

```
pip install pandas_datareader
```

[ ]:

```
pip install seaborn
```

[ ]:

```
```