GKBoginyFreyaBank

February 14, 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.

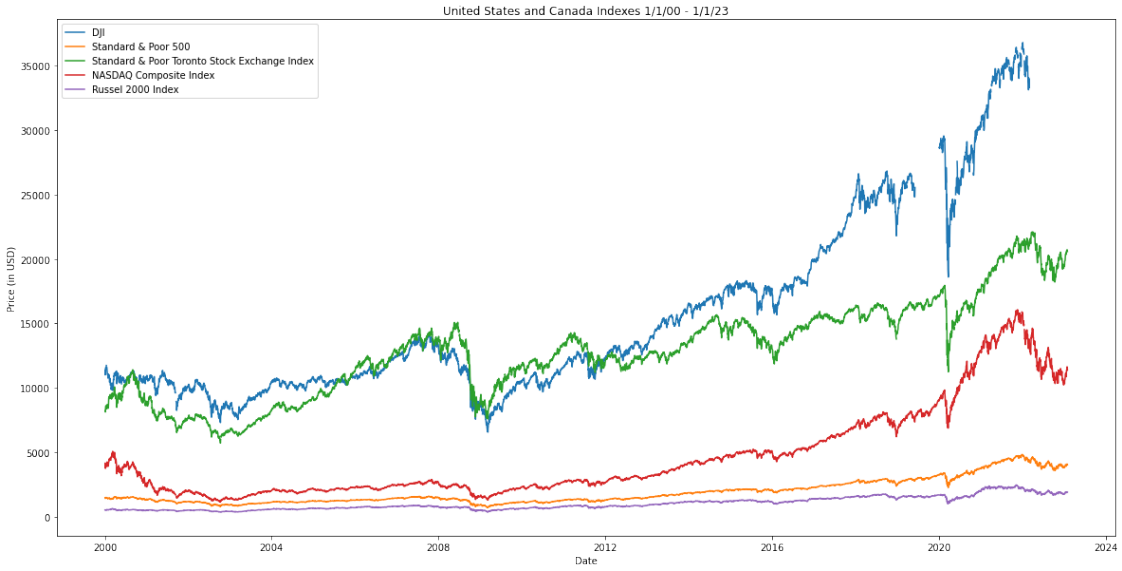
Obtain the Financial Data Data related to stock indices can be retrieved from Yahoo! Finance. In Python, there has been a popular module to retrieve data more easily from Yahoo! Finance. If it has not been installed yet, you can install it in your Jupyter Notebook by typing this chunk of codes, followed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

1.1 US and Canada Indexes

[17]:
```
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)

data.rename(columns={'^GSPC': 'Standard & Poor 500',
                     '^DJI': 'Dow Jones Industrial Average Index ',
                     '^IXIC': 'NASDAQ Composite Index',
```

```
                    '^RUT': 'Russel 2000 Index',
                    '^GSPTSE': 'Standard & Poor Toronto Stock Exchange␣
 ↪Index'}, inplace=True)

#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

[15]:
```
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)

data.rename(columns={'^BVSP': 'IBOVESPA Brazil Index',
                     '^MXX': 'Mexico Stock Exchange Index ',
                     '^IPSA': 'Santiago Composite Index'}, inplace=True)

#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()
```

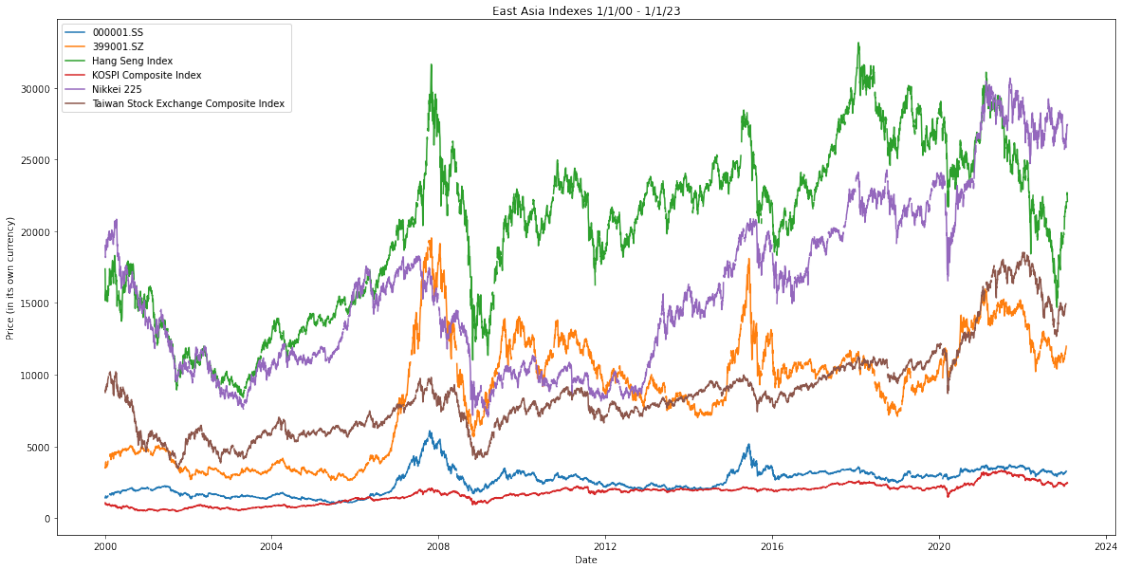
```
[**********************100%**********************]  3 of 3 completed
```

1.3 East Asia Indexes

[16]:
```
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)

data.rename(columns={'^N225': 'Nikkei 225',
                     '^HSI': 'Hang Seng Index ',
                     '^000001.SS': 'Shanghai Stock Exchange Composite Index',
                     '^399001.SZ': 'Shenzhen Index ',
                     '^TWII': 'Taiwan Stock Exchange Composite Index ',
                     '^KS11': 'KOSPI Composite Index '}, inplace=True)

#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("East Asia Indexes 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

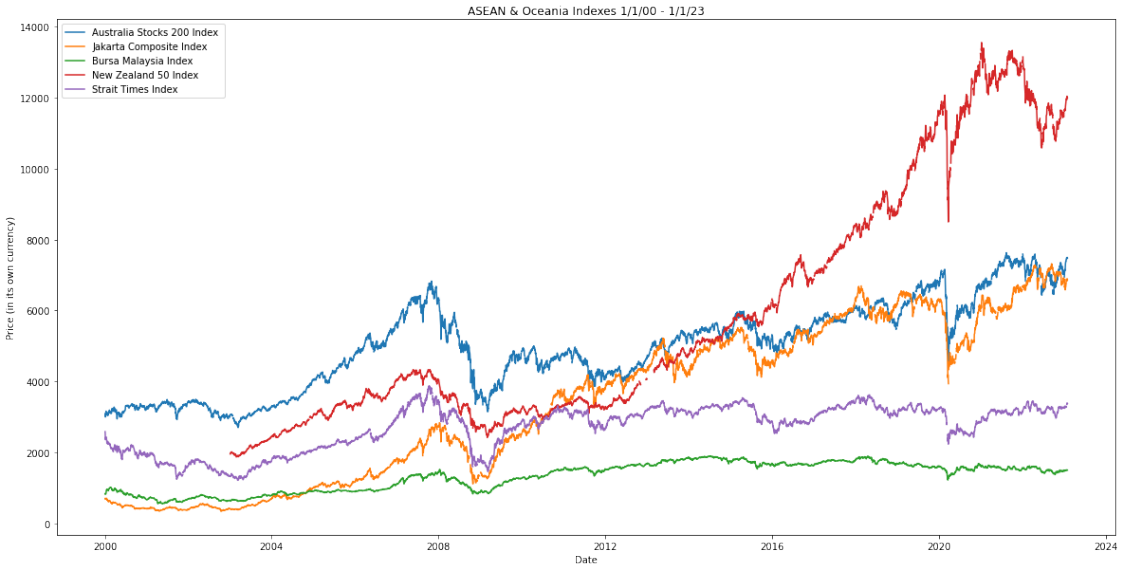
```
[**********************100%**********************]  6 of 6 completed
```



1.4 ASEAN & Oceania Indexes

[12]:
```
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)

data.rename(columns={'^STI': 'Strait Times Index',
                     '^JKSE': 'Jakarta Composite Index ',
                     '^KLSE': 'Bursa Malaysia Index',
                     '^AXJO': 'Australia Stocks 200 Index ',
                     '^NZ50': 'New Zealand 50 Index '}, inplace=True)

#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()
```

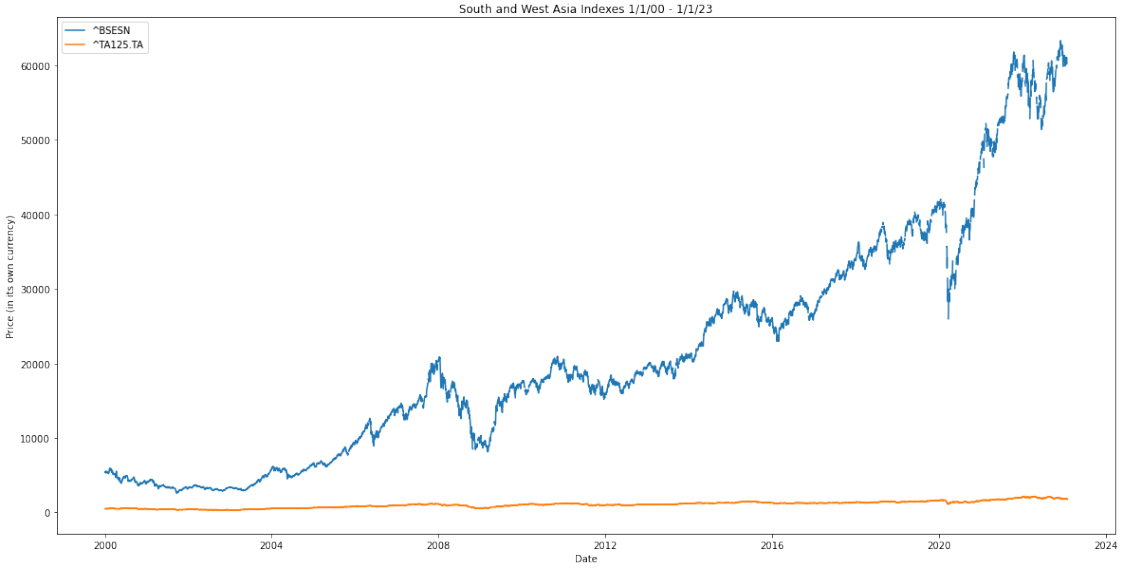
```
[**********************100%**********************]  5 of 5 completed
```



1.5 South and West Asia Indexes

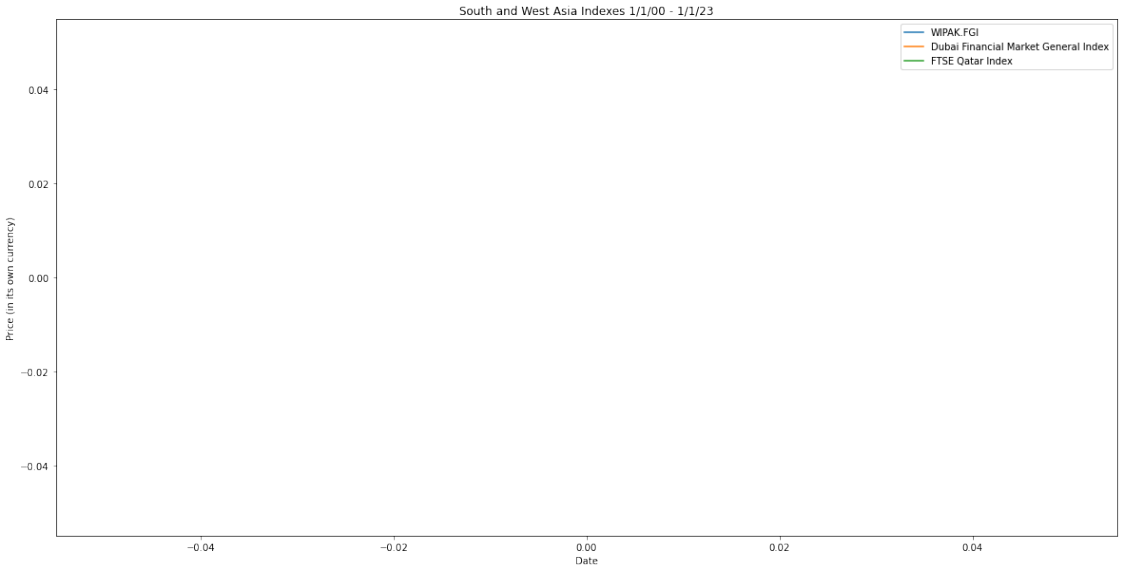
[10]:
```
from pandas_datareader import data as pdr
import yfinance as yf

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
```



[11]:
```
import matplotlib.pyplot as plt
import yfinance as yf

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['^DFMGI.AE', '^WIQAT.FGI','WIPAK.FGI']
from datetime import datetime
startdate = datetime(2010,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'^DFMGI.AE': 'Dubai Financial Market General Index',
                     '^WIQAT.FGI': 'FTSE Qatar Index ',
                     '^WIPAK.FGI': 'FTSE Pakistan Index'}, inplace=True)

#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%**********************]  3 of 3 completed

3 Failed downloads:
- ^WIQAT.FGI: No timezone found, symbol may be delisted
- ^DFMGI.AE: No timezone found, symbol may be delisted
- WIPAK.FGI: Period 'max' is invalid, must be one of ['1d', '5d']
```

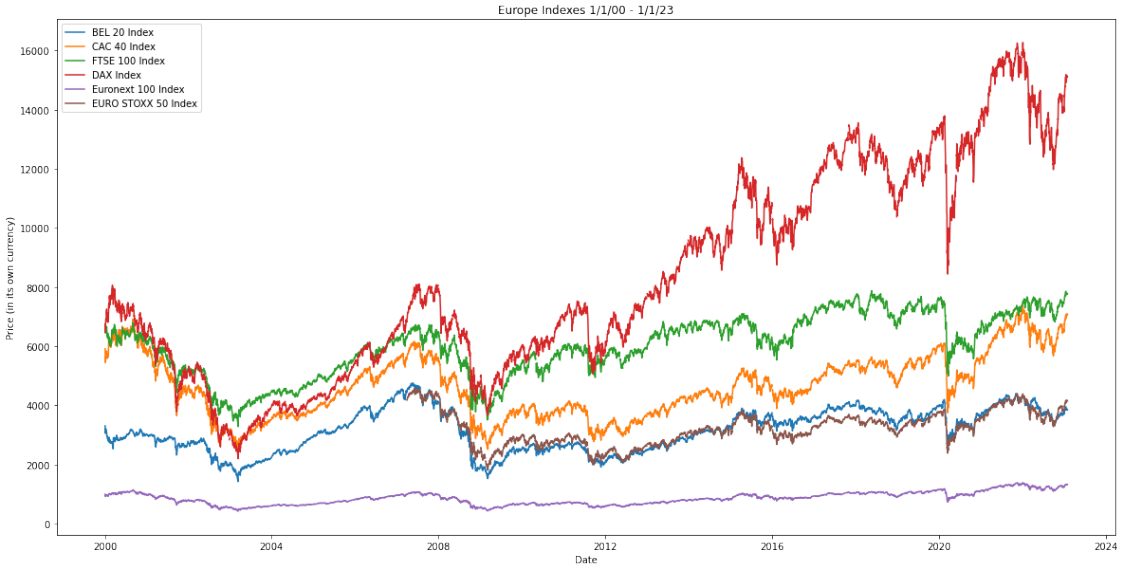


1.6 Europe Indexes

[1]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from forex_python.converter import CurrencyRates
from datetime import datetime

cr = CurrencyRates()

# Get exchange rate between GBP and USD and EUR and USD
gbp_to_usd = cr.get_rate('GBP', 'USD')
eur_to_usd = cr.get_rate('EUR', 'USD')

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

data.rename(columns={'^FTSE': 'FTSE 100 Index',
                     '^GDAXI': 'DAX Index',
                     '^FCHI': 'CAC 40 Index',
                     '^STOXX50E': 'EURO STOXX 50 Index',
                     '^N100': 'Euronext 100 Index',
                     '^BFX': 'BEL 20 Index'}, inplace=True)
```

```
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()
```

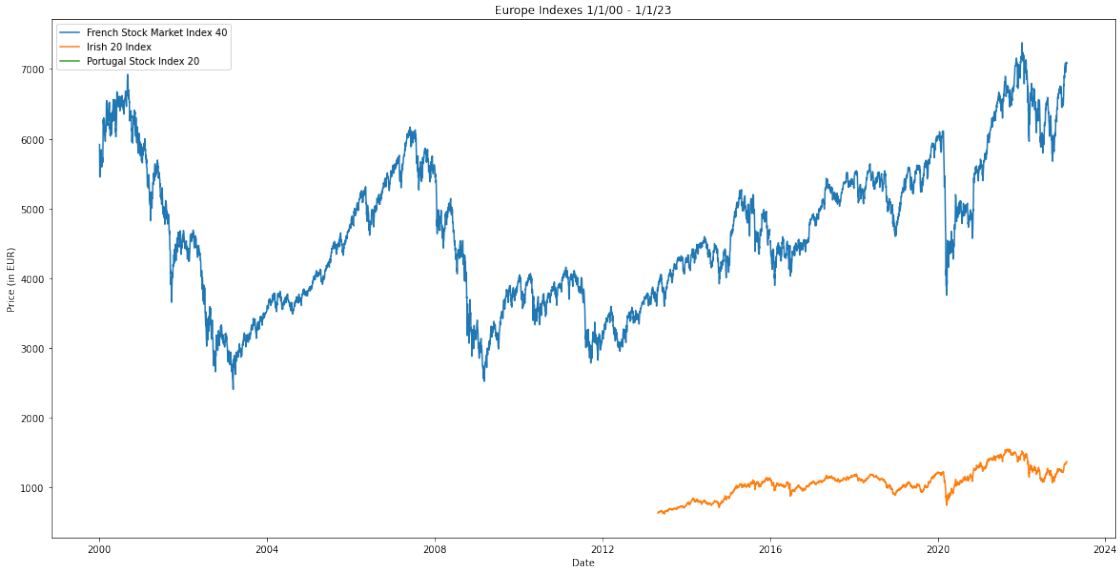
```
[**********100%**********]  6 of 6 completed
```



[18]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from forex_python.converter import CurrencyRates
from datetime import datetime

cr = CurrencyRates()

# Get exchange rate between GBP and USD and EUR and USD
gbp_to_usd = cr.get_rate('GBP', 'USD')
eur_to_usd = cr.get_rate('EUR', 'USD')

yf.pdr_override()
y_symbols = ['^FCHI', '^IETP', '^PSI20.LS']
startdate = datetime(2000,1,1)
```

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

data.rename(columns={'^FCHI': 'French Stock Market Index 40',
                     '^IETP': 'Irish 20 Index',
                     '^PSI20.LS': 'Portugal Stock Index 20'}, inplace=True)

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

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

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

1 Failed download:
- ^PSI20.LS: No timezone found, symbol may be delisted
```



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	

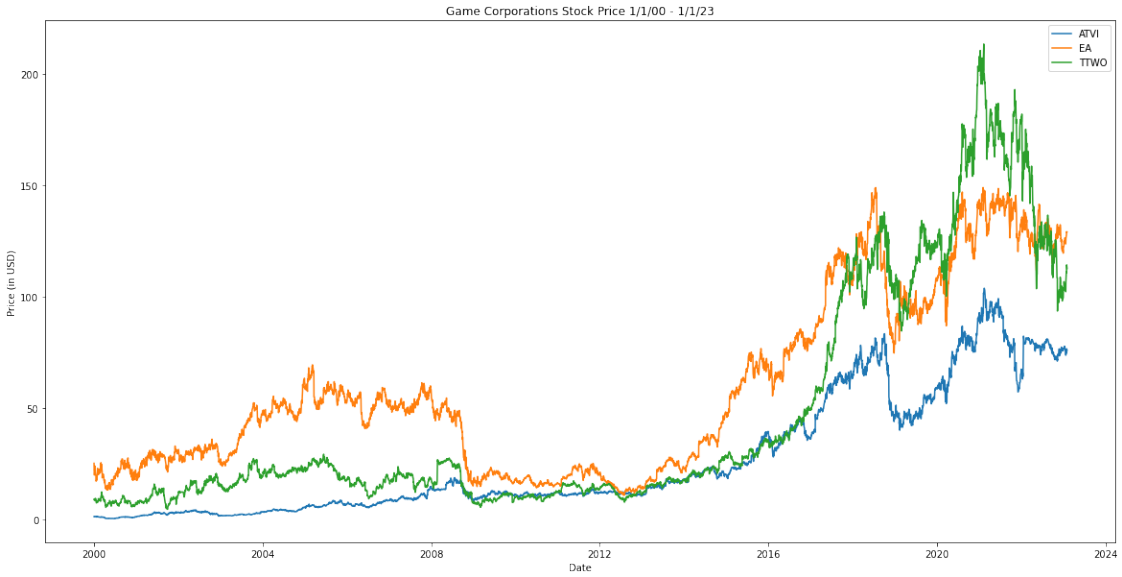
		High			Low	\
	TTWO	ATVI	EA	TTWO	ATVI	
Date						
2000-01-03	9.125000	1.375000	28.765625	10.000000	1.166667	
2000-01-04	8.875000	1.354167	24.500000	9.333333	1.187500	
2000-01-05	8.833333	1.364583	24.375000	8.875000	1.312500	
2000-01-06	8.750000	1.333333	22.625000	9.000000	1.296875	
2000-01-07	9.000000	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
```

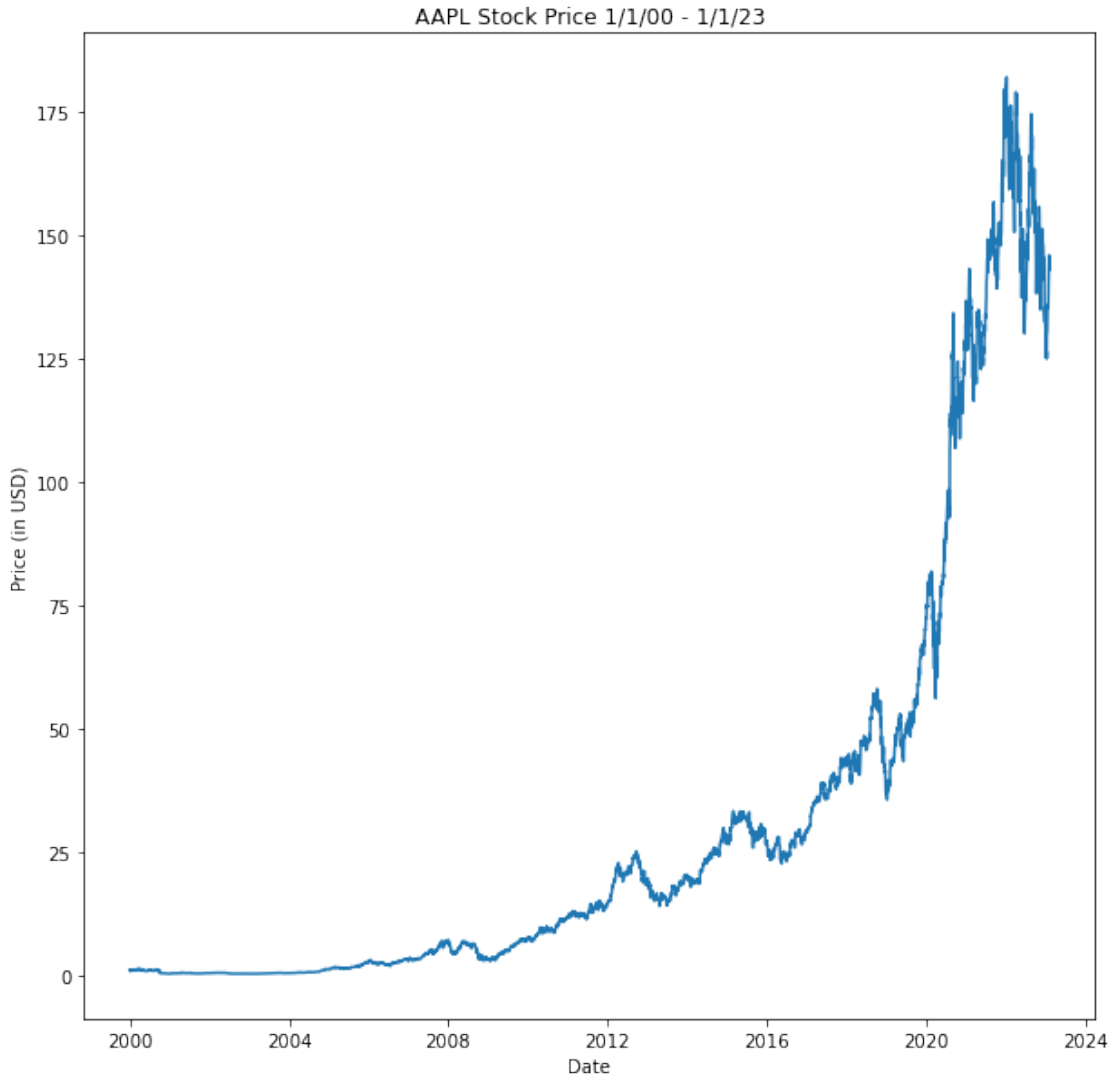
[67]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-17	134.830002	137.289993	134.130005	135.940002	135.732758	63646600
2023-01-18	136.820007	138.610001	135.029999	135.210007	135.003876	69672800
2023-01-19	134.080002	136.250000	133.770004	135.270004	135.063782	58280400
2023-01-20	135.279999	138.020004	134.220001	137.869995	137.659805	79972200
2023-01-23	138.119995	143.320007	137.899994	141.110001	140.894882	81760300
2023-01-24	140.309998	143.160004	140.300003	142.529999	142.312714	66435100
2023-01-25	140.889999	142.429993	138.809998	141.860001	141.643738	65799300
2023-01-26	143.169998	144.250000	141.899994	143.960007	143.740540	54105100
2023-01-27	143.160004	147.229996	143.080002	145.929993	145.707520	70492800
2023-01-30	144.960007	145.550003	142.850006	143.000000	142.781998	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")
```

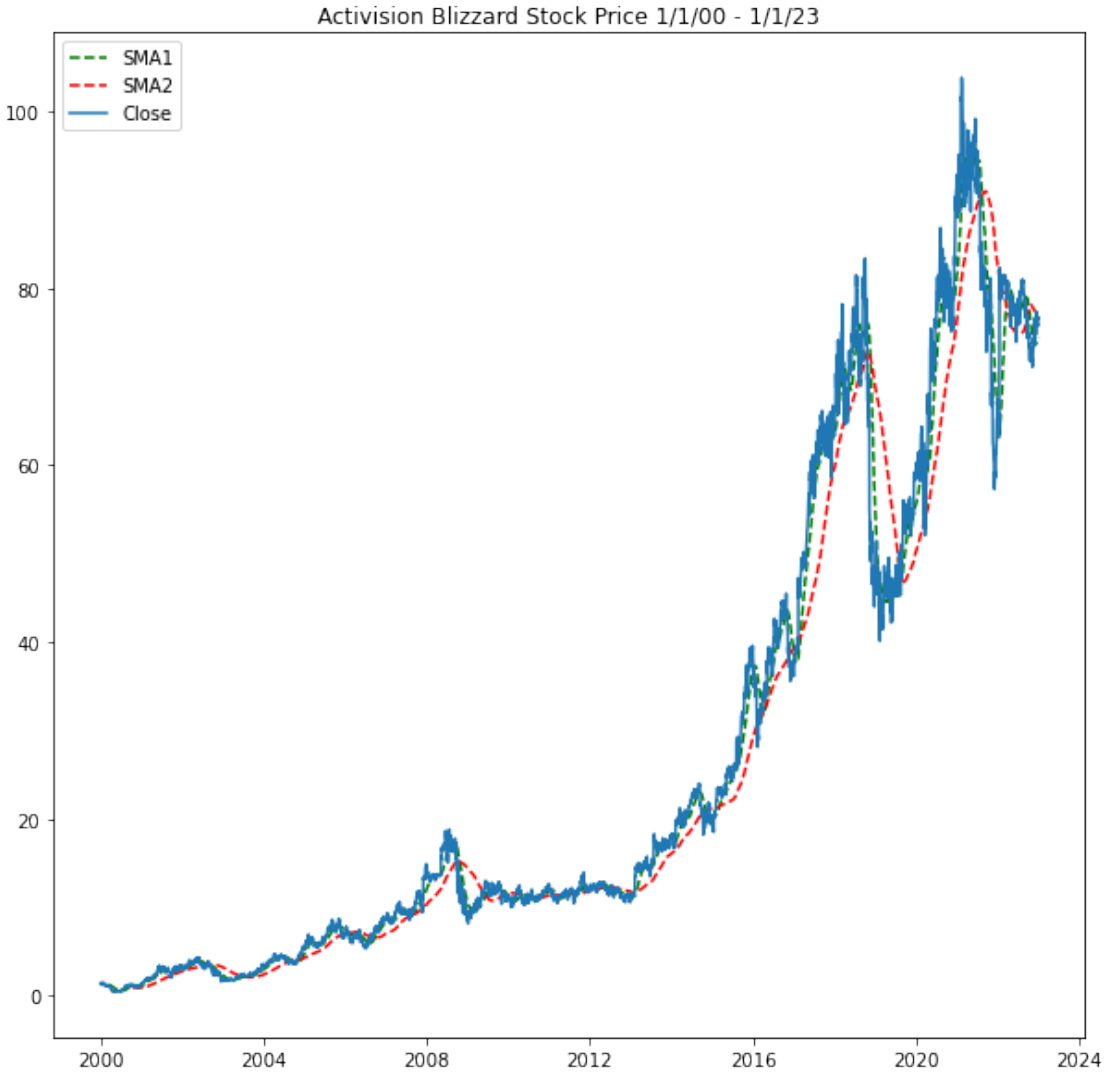
[68]: Text(0.5, 1.0, 'AAPL Stock Price 1/1/00 - 1/1/23')



2.3 Single Plot using Pandas and yfinance with Moving Average

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

company = 'ATVI'

# 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
```

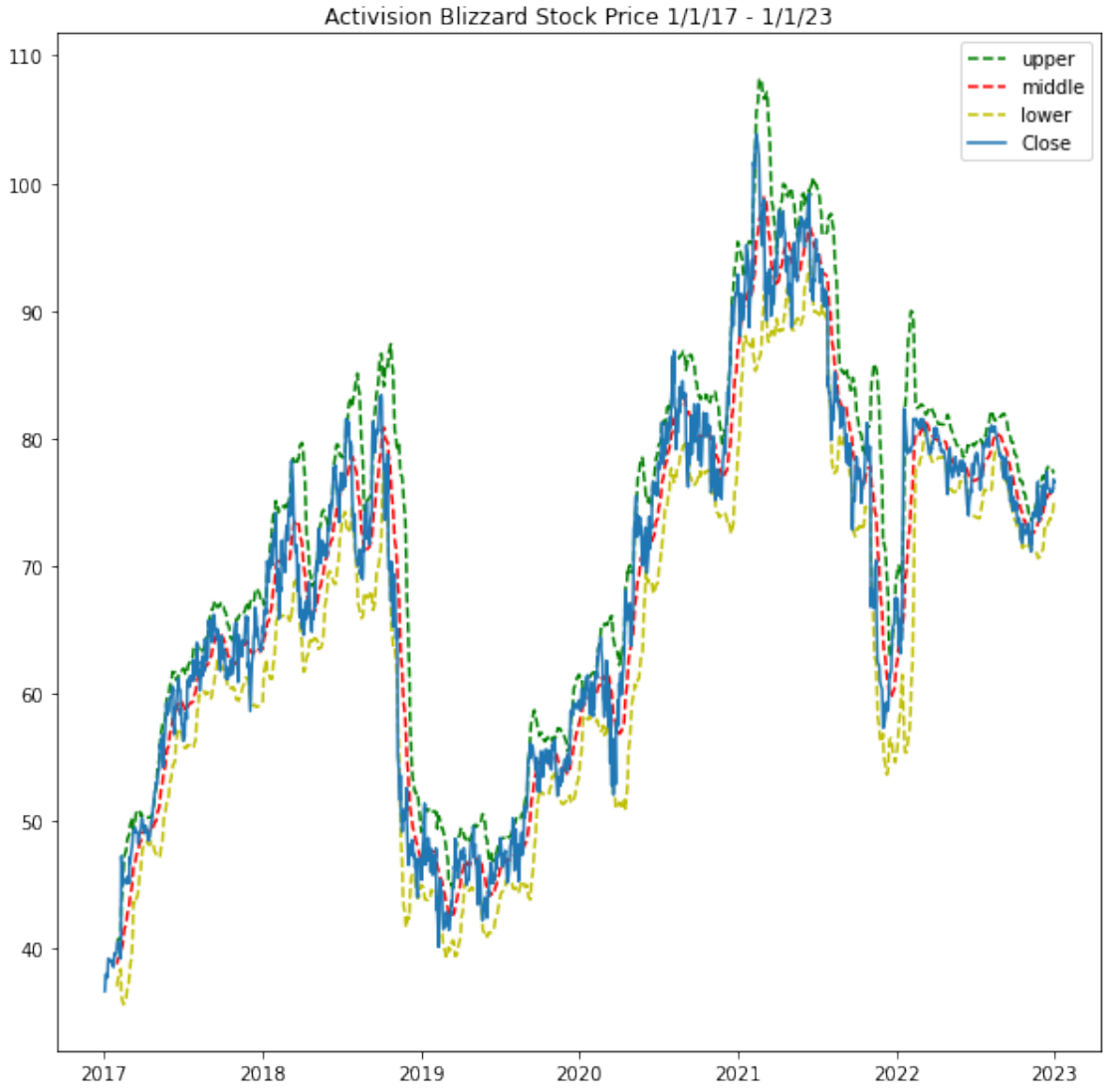


2.4 Single Plot using Pandas and yfinance with Bollinger Bands

[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
```

```
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
```

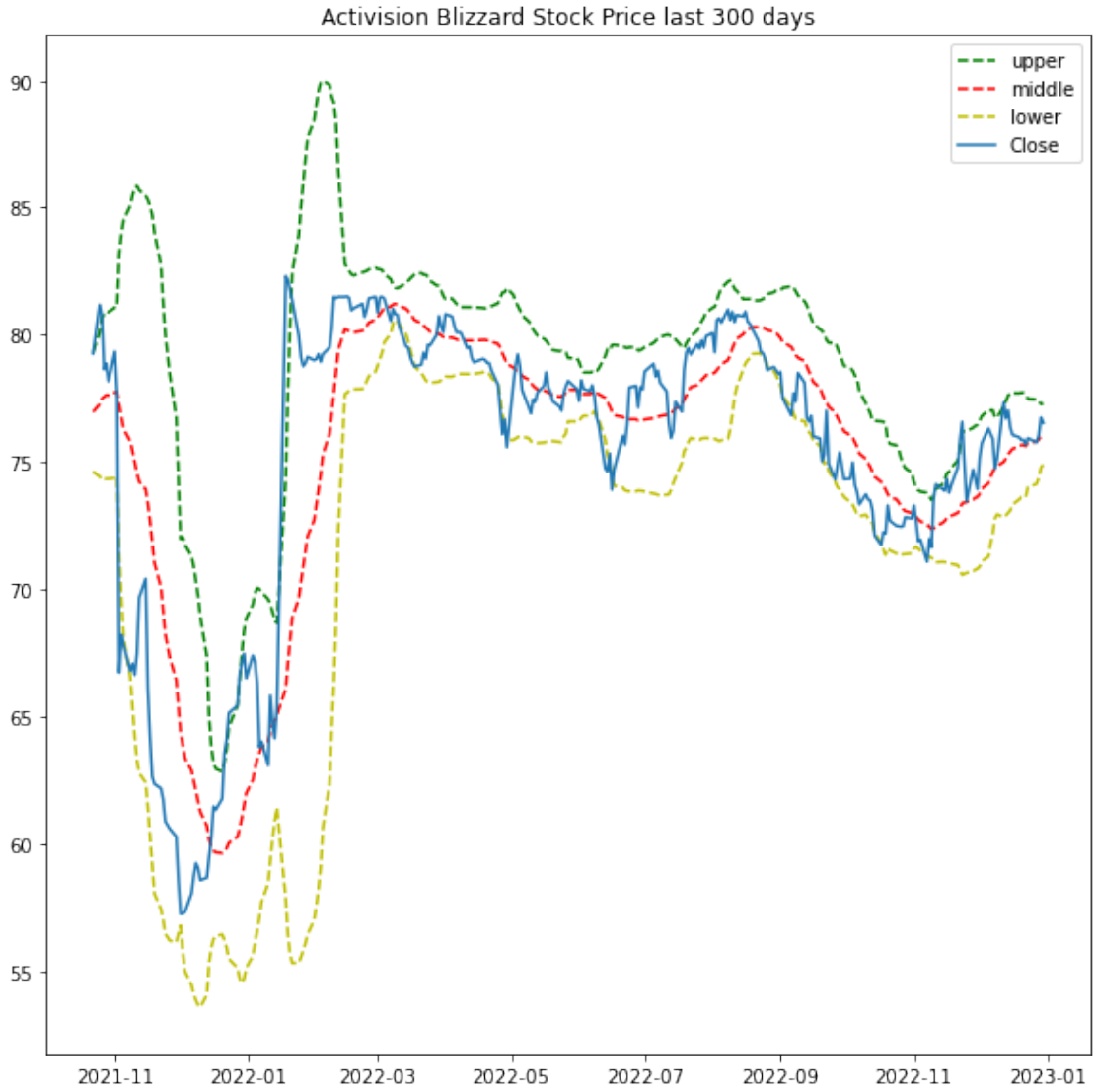


Bollinger Bands that zooms in on the past 300 days of trading

[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
                                    ])
```

3 Chapter III: Commodities

Sources:

- https://www.icdx.co.id/product/gold?category=goldud
- https://finance.yahoo.com/commodities?.tsrc=fin-srch

We are going to plot the prices of several commodities:

1. Gold
2. Silver
3. Crude Oil
4. Aluminum
5. Natural Gas
6. Copper
7. Cocoa
8. Ethanol
9. Palladium
10. Soybean
11. Sugar
12. Cotton

Commodities are traded on Futures Exchange. Futures Exchange is the Exchange for commodities that will deliver the products from the seller of the contracts to the buyer at certain date in the future for fixed price paid today.If you look at the price of commodities it will contain the month and

year of delivery as well, e.g `Aluminum Futures,Apr-2023 (ALI=F)`. It means that the Aluminum contract you will buy at a price X will be delivered on April 2023. If you buy it today on February 14th, 2023, at a price of USD 2425 per metric ton to fill the base material of your packaging factory that will need new aluminum on April 2023, you will be happy if the Aluminum price on April 2023 (at spot rate / spot market) is USD 2788 (above your future price that you have bought on February 14th, 2023)

Why we need to learn this? Simple.. you can hedge and minimize your losses, mazimize your profit, another example if you want to start a business in garment / textile you might need to enter a contract with seller of Cotton in Futures Market / Exchange to deliver it at date which you predict your storage will be running out of cotton. You might have a lot of contracts ordering your garments if it is loved and have high quality and always stylist. It is better than buying cotton in the spot market, the futures market might be a good hedging that can save you a lot of money. Bad weather in the future can make the cotton price at spot market hikes beyond anticipation.

3.1 Gold

Gold is considered a category of precious metal because of its relatively high rarity. Due to its valuable nature and widespread use throughout the world, gold is valued universally almost equally between places or countries.

In the past, gold was used as bartering currency for daily necessities. However, its shape and weight made gold difficult to carry in large quantities. To that end, central banks then based their currency printing on the gold deposits in their vaults.

Other than its past function and characteristics, gold is also considered a safe haven asset because its value will remain the same over time. Regarding the characteristics of gold as a safe haven asset, gold prices move relatively in line with changes in global risk sentiment.

3.2 Silver

Silver is among the most traded commodities in the world. It is a precious metal which has been used by humanity for millennia now. Silver is intrinsic to industrial demand in sectors such as electronics, medicine etc. in fact, more than half of all silver consumption is for industrial purposes.

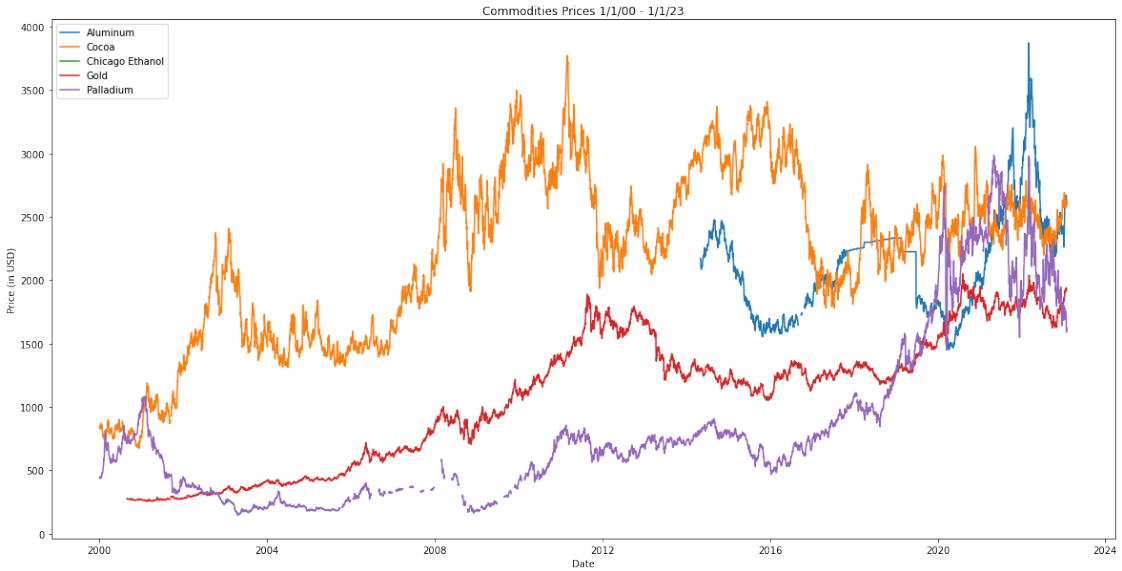
Futures contracts are the main way to trade silver. A futures contract is an agreement to buy or sell silver for a set price on a future date. While futures contracts can be used to take possession of the physical commodity, you don't necessarily have to – futures contracts can be settled in cash

3.3 Crude Oil

Crude oil is still the world's main energy source, contributing 40% of the world's total energy needs. The world consumes about 76 million barrels of crude oil per day. With more than 160 different types of benchmarks for oil, crude oil prices are pegged to three main benchmarks: London Brent, West Texas Intermediate (WTI) and Dubai / Oman crude.

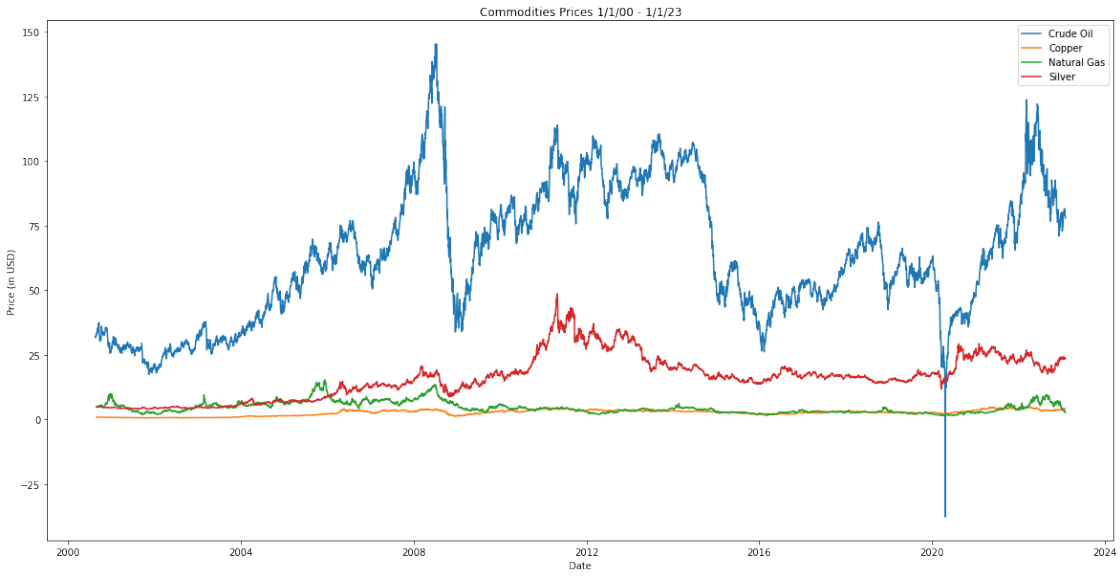
[34]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['GC=F', 'ALI=F', 'CU=F', 'CC=F','PA=F']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'GC=F': 'Gold',
                     'ALI=F': 'Aluminum',
                     'CU=F': 'Chicago Ethanol',
                     'PA=F': 'Palladium',
                     'CC=F': 'Cocoa'}, inplace=True)

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("Commodities Prices 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

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

1 Failed download:
- CU=F: Period 'max' is invalid, must be one of ['1d', '5d']
```



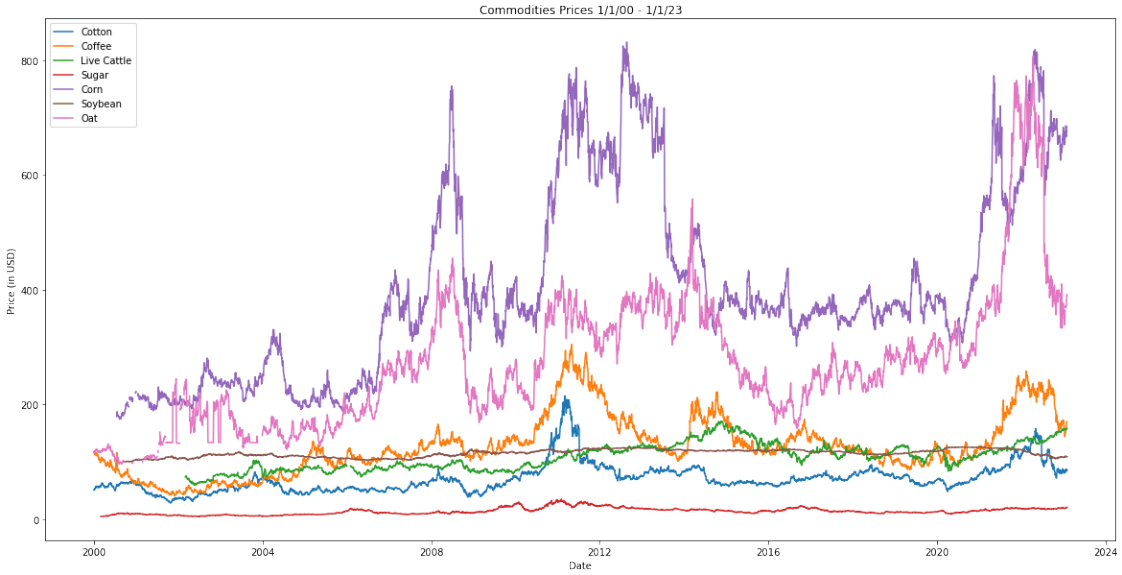
[28]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['SI=F', 'CL=F', 'NG=F','HG=F']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'SI=F': 'Silver',
                     'CL=F': 'Crude Oil',
                     'NG=F': 'Natural Gas',
                     'HG=F': 'Copper'}, inplace=True)

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("Commodities Prices 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[**********************100%**********************]  4 of 4 completed
```



[32]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['ZC=F', 'ZO=F', 'LE=F','KC=F', 'SB=F', 'CT=F', 'ZF=F']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'ZC=F': 'Corn',
                     'ZO=F': 'Oat',
                     'LE=F': 'Live Cattle',
                     'KC=F': 'Coffee',
                     'SB=F': 'Sugar',
                     'CT=F': 'Cotton',
                     'ZF=F': 'Soybean'}, inplace=True)

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("Commodities Prices 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[**********************100%**********************]  7 of 7 completed
```



4 Chapter IV: Foreign Exchange

After paying with gold in the past, we are now stuck for a very long time with cash money: USD, GBP, CNY, JPY. The exchange rate between a pair / two currencies are traded in the futures exchange, tourists are the targets to increase the value of one currency, you can see that country with lots of tourist will have high value of its own currency compared to country with less tourist. Besides tourist, trade is also the main factor, if a country is main exporter of almost all commodities that needed by other countries then its currencies will also have high value.

[37]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['EURUSD=X', 'GBPUSD=X']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'EURUSD=X': 'EUR/USD',
                     'GBPUSD=X': 'GBP/USD'}, inplace=True)

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

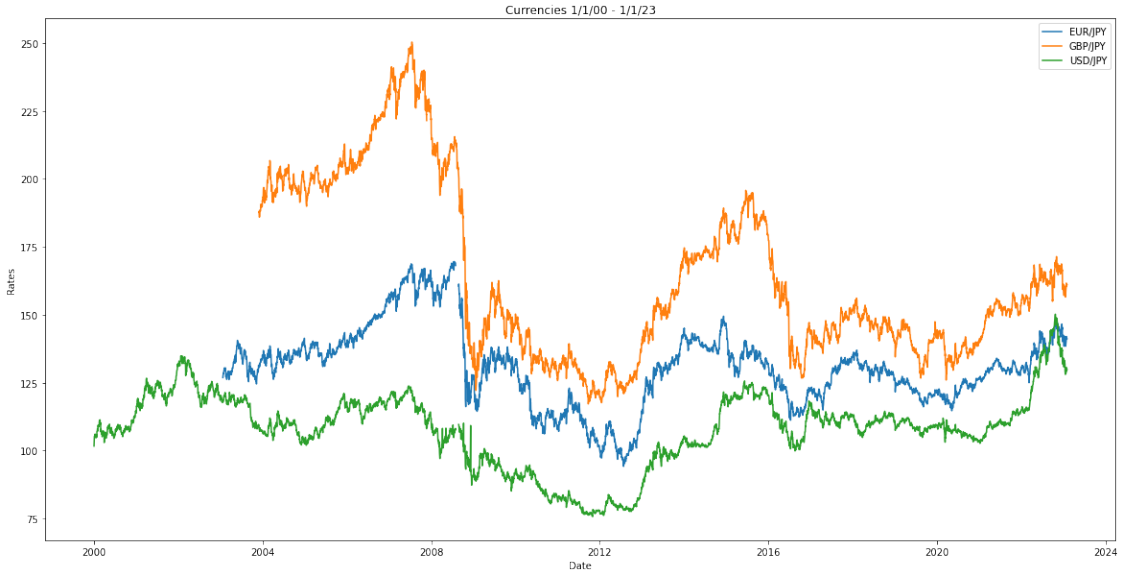
```
plt.xlabel("Date")
plt.ylabel("Rates")
plt.title("Currencies 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

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

[38]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['USDJPY=X', 'GBPJPY=X', 'EURJPY=X']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'USDJPY=X': 'USD/JPY',
                     'GBPJPY=X': 'GBP/JPY',
                     'EURJPY=X': 'EUR/JPY'}, inplace=True)

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

```
plt.xlabel("Date")
plt.ylabel("Rates")
plt.title("Currencies 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

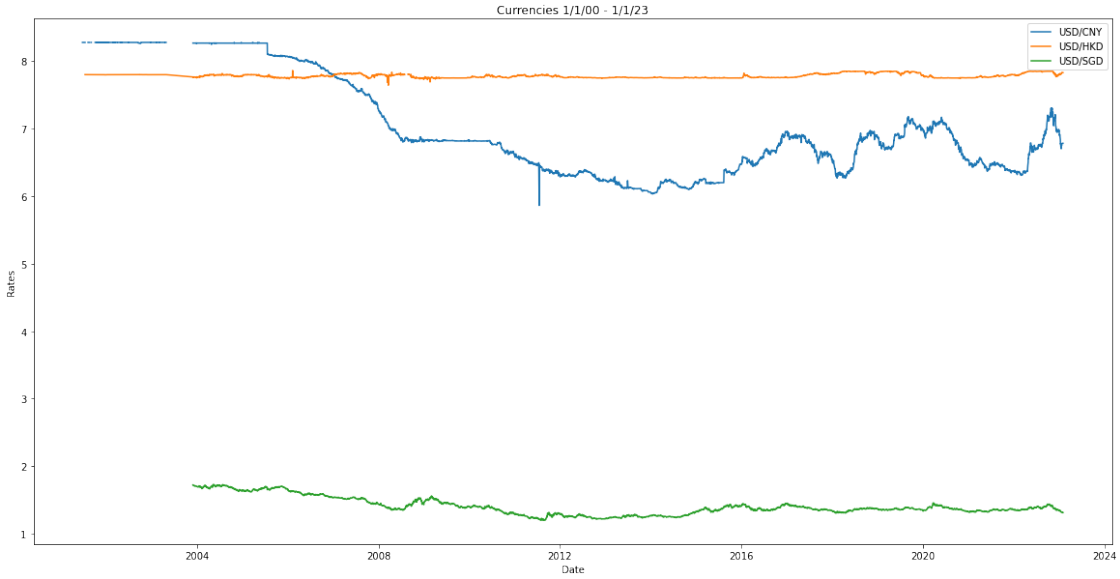
```
[**********100%**********]  3 of 3 completed
```



[39]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['USDCNY=X', 'USDHKD=X', 'USDSGD=X']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'USDCNY=X': 'USD/CNY',
                     'USDHKD=X': 'USD/HKD',
                     'USDSGD=X': 'USD/SGD'}, inplace=True)

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

plt.xlabel("Date")
```

```
plt.ylabel("Rates")
plt.title("Currencies 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

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



5 Chapter V: Bonds

Bonds are the safest instrument to invest in, either corporate or government, bondholders receive the right to be paid first before shareholders. It is based on the U.S. laws that is applied now.

Bonds are paid in coupon ever period (every 3 months, every 6 months or every year) before its maturity date, and it will pay the maturity price. There are lots of variations in bonds, bonds with lower face value, below its par value, or premium bonds.

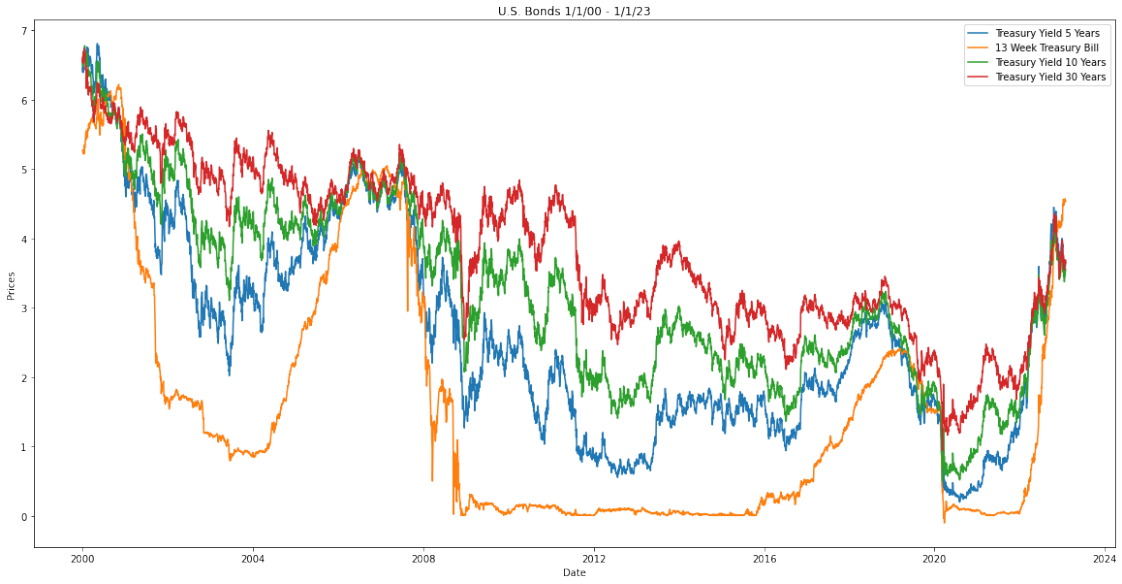
Yahoo Finance provides the bonds price for the U.S. government only.

5.1 Foreign Bonds: Indonesia Government Bond

In Indonesia there is SBR012, a government bond with maturity of 2 years(SBR012-T2) and 4 years(SBR012-T4), offered if you have an account in BNI Sekuritas around January 2023 and February 2023. The coupon for SBR012-T2 is 6.15 %, and the coupon for SBR012-T4 is 6.35 %. The rates / coupon might be higher than the U.S. government bonds with same maturity, but we know that the currency of Indonesia is very weak, with USD/IDR around 15,500 in February 14th, 2023. But if you are confident in 2 years USD/IDR could become 5,000 then why not take the chance?

[41]:
```
import yfinance as yf
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
from datetime import datetime

yf.pdr_override()
y_symbols = ['^IRX', '^FVX', '^TNX', '^TYX']
startdate = datetime(2000,1,1)
enddate = datetime(2023,1,31)
data = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

data.rename(columns={'^IRX': '13 Week Treasury Bill',
                     '^FVX': 'Treasury Yield 5 Years',
                     '^TNX': 'Treasury Yield 10 Years',
                     '^TYX': 'Treasury Yield 30 Years'}, inplace=True)

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

plt.xlabel("Date")
plt.ylabel("Prices")
plt.title("U.S. Bonds 1/1/00 - 1/1/23")
plt.legend()
plt.show()
```

```
[**********************100%**********************]  4 of 4 completed
```



6 Chapter VI: 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]:

Date	Open	High	Low	Close
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]:
```python
#  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()
```

7 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
```

[ ]:
```

```