Time Series Analysis

Importing libraries

In [1]:
```
# Financial Financial libraries
import pandas as pd
import numpy as np

# Data's date & time
import yfinance as yf

# Latest matplotlib
import matplotlib
matplotlib.rcParams['figure.figsize'] = (15,6)
```

Retrieving End-of-Day Data - Single Security

Case 1

In [2]:
```
# Fetch data by specifying the number of the period
yf.download('ABBV', period='5d')
```

Out[2]:
```
[*********************100%***********************]  1 of 1 completed
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-09-27	166.210760	167.050074	165.64.202203	167.16.440278	167.16.440278	203600
2023-09-28	167.05.38076	167.06.03301	164.62.289929	165.53.03701	165.53.03701	263600
2023-09-29	165.180619	167.26.25302	165.01.250191	166.26.33291	166.26.33291	243200
2023-09-30	166.22.40200	166.53.38076	164.73.63291	165.26.75003	165.26.75003	227700
2023-10-04	164.30.3076	164.57.83291	163.53.389929	164.06.289929	164.06.289929	0

Case 2

In [3]:
```
# Stock the data from yahoo using start and end dates
yf.download('ABBV', start='2023-10-20', end='2023-09-20', progress=False)
```

Out[3]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-09-20	166.75.03201	2103.03.03201	166.76.849629	169.01.452291	169.01.452291	179100
2023-09-21	166.63.03701	166.66.750203	165.76.849629	167.62.248629	167.62.248629	275300
2023-09-22	167.64.849629	167.66.03201	165.71.03201	165.76.03201	165.76.03201	246400
2023-09-25	165.76.18076	167.54.703201	165.03.250191	166.76.03701	166.76.03701	186400
2023-09-26	166.61.83076	166.03.349629	163.67.449278	164.64.849629	164.64.849629	254600
2023-09-27	163.57.03701	165.53.850629	163.64.203201	167.16.440278	167.16.440278	212600
2023-09-28	167.61.83076	167.66.03201	164.62.289929	165.53.03701	165.53.03701	263600
2023-09-29	165.180619	167.26.250203	165.01.250191	166.26.33291	166.26.33291	243200

Case 3

In [4]:
```
# Fetch data for year to date (YTD)
Nifty = yf.download('NIFTY', period='ytd', progress=False)
```

In [5]:
```
# Display the first five rows of the dataframe to check the results.
Nifty.head()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-01-02	18,131.050195	18,175.150391	18,086.320313	18,107.449219	18,107.449219	258100
2023-01-03	18,102.199219	18,323.449219	18,100.650391	18,251.150391	18,251.150391	258700
2023-01-04	18,224.650391	18,243.300300	18,123.199609	18,242.249219	18,242.249219	235200
2023-01-05	18,213.349219	18,232.350391	17993.199609	17992.150391	17992.150391	208800
2023-01-06	18,026.050781	18,047.400391	17,795.349781	17855.449219	17855.449219	238200

Extracting Data for Multiple Securities

In [6]:
```
# Define the securities
stocks = ['RELIANCE.NS', 'BHARTIARTL.NS', 'TCS.NS', 'HINDUNILVR.NS', 'HDFCBANK.NS']
start_date = '2023-01-01'
end_date = '2023-09-30'

# Fetching data for multiple tickers
df = yf.download(stocks, start=start_date, end=end_date, progress=False)['Adj Close']

# Display the first five rows of the dataframe to check the results.
df.head()
```

Out[6]:

	BHARTIARTL.NS	HDFCBANK.NS	HINDUNILVR.NS	RELIANCE.NS	TCS.NS
Date					
2023-01-03	799.678987	1597.177246	2563.017725	2301.150424	2985.579125
2023-01-04	815.022216	1636.328223	2539.268875	2326.755266	2951.009041
2023-01-05	784.894141	1625.767520	2575.542725	2313.679725	3052.320398
2023-01-06	789.473072	1614.027756	2581.651383	2315.788398	2996.460255
2023-01-10	788.726990	1598.898909	2587.286862	2307.070142	3016.387337

Retrieving Intraday Data

In [7]:
```
# Receive intraday data
df_intraday = yf.download(tickers='ABBV', period='5d', interval='1m', progress=False)

# Display the first five rows of the dataframe to check the results.
df_intraday.head()
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
Datetime						
2023-09-27 09:15:00+05:30	166.210760	166.210000	166.06.403201	166.21.750203	166.21.750203	0
2023-09-27 09:16:00+05:30	166.21.090625	166.24.250781	166.03.250781	166.10.750203	166.10.750203	0
2023-09-27 09:17:00+05:30	166.10.750203	166.26.443278	166.06.403201	166.22.250625	166.22.250625	0
2023-09-27 09:18:00+05:30	166.22.250781	166.23.350625	166.22.250781	166.23.603701	166.23.603701	0
2023-09-27 09:19:00+05:30	166.23.603701	166.24.503203	166.26.503203	166.27.050625	166.27.050625	0

Resampling

In [8]:
```
# Resampling to derive weekly values for daily time series
Nifty_weekly = Nifty.resample('W').last()

# Resampling to derive monthly values from daily time series
Nifty_monthly = Nifty.resample('M').last()

# Display the first five rows of the data frame to check the output
Nifty_monthly.tail()
```

Out[8]:

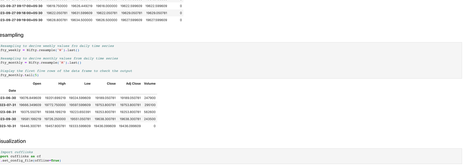
	Open	High	Low	Close	Adj Close	Volume
Date						
2023-06-30	18,179.849609	18,231.050781	18,114.199609	19189.550781	19189.550781	267600
2023-07-31	19666.349609	19772.750203	19587.599609	19753.850781	19753.850781	280300
2023-08-31	19275.150781	19356.199219	19133.650391	19253.650781	19253.650781	302600
2023-09-30	19261.199219	19256.250203	19053.250781	19126.350781	19126.350781	243900
2023-10-31	19446.199219	19453.550781	19233.199609	19418.199609	19418.199609	0

Visualization

In [9]:
```
# Import matplotlib
import matplotlib as mf
mf.set_config_file('offline=True)
```

Visualising a Price Series

In [10]:
```
df['RELIANCE.NS'].iplot(kind='line', title='Reliance Price Series')
```

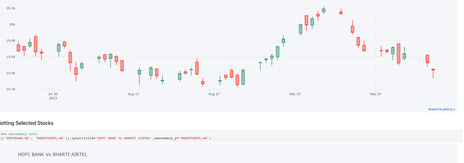


Plotting OHLC Data

In [11]:
```
Nifty[:50].iplot(kind='ohlc',title='Nifty Price')
```

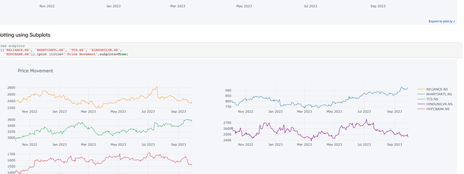
Plotting Candlestick

In [12]:
```
Nifty[:50].iplot(kind='candle',title='Nifty Price')
```



Plotting Selected Stocks

In [13]:
```
# Use secondary axis
df[['HDFCBANK.NS', 'BHARTIARTL.NS']].iplot(title='HDFC BANK Vs BHARTI AIRTEL',secondary_y='BHARTIARTL.NS')
```



Plotting using Subplots

In [14]:
```
# Use subplots
df[['RELIANCE.NS', 'BHARTIARTL.NS', 'TCS.NS', 'HINDUNILVR.NS',
    'HDFCBANK.NS']].iplot.line(subplots=True, theme='ggplot',subplot=True)
```

Normalized Plot

In [15]:
```
df.normalize().iplot(title='Normalized')
```

In [16]:
```
# Calculating Log Normal Returns
# Use the shift function to derive log normal returns
daily_returns = np.log(df).diff().fillna(0)

# Display the last five rows of the data frame to check the output
daily_returns.head(5)
```

Out[16]:

	BHARTIARTL.NS	HDFCBANK.NS	HINDUNILVR.NS	RELIANCE.NS	TCS.NS
Date					
2022-10-03	0.000000	0.000000	0.000000	0.000000	0.000000
2022-10-04	0.006535	0.027774	0.017762	0.016275	0.010680
2022-10-06	-0.031939	-0.010073	-0.024429	-0.001380	0.003488
2022-10-07	0.000950	-0.004324	-0.005465	-0.004202	-0.010016
2022-10-10	-0.000846	-0.01104	0.002387	-0.010975	0.017053

Visualising Return Series

In [17]:
```
daily_returns[['RELIANCE.NS', 'BHARTIARTL.NS', 'TCS.NS', 'HINDUNILVR.NS',
               'HDFCBANK.NS']].iplot(title='Daily Log Returns')
```

Plotting Annual Returns

In [18]:
```
# Book Annual Returns
daily_returns.mean()*252.iplot(kind='bar')
```


Rolling Returns

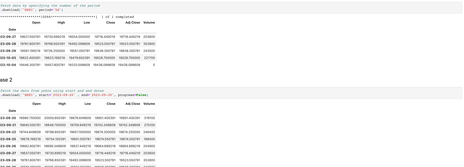
In [19]:
```
# In addition to log rolling returns, simply use daily returns for 21 days as log returns are additive
rolling_return = daily_returns.rolling(21).sum().dropna()

# Display the last five rows of the data frame to check the output
rolling_return.head(5)
```

Out[19]:

	BHARTIARTL.NS	HDFCBANK.NS	HINDUNILVR.NS	RELIANCE.NS	TCS.NS
Date					
2022-11-02	0.006059	-0.060031	-0.032867	0.071737	0.080100
2022-11-03	0.016866	0.064628	-0.022782	0.075432	0.074260
2022-11-04	0.006476	0.030603	-0.016215	0.077196	0.042615
2022-11-07	0.042232	0.048757	-0.024869	0.057041	0.044981
2022-11-08	0.032196	0.052783	-0.018804	0.060918	0.050724

Visualising a Rolling Return Series

In [20]:
```
rolling_return['TCS.NS'].iplot(title='21-Days Rolling Returns of TCS')
```


Normal Distribution

In [21]:
```
# Plot log normal distribution of returns
daily_returns.iplot(kind='histogram', title = 'Histogram of Daily Returns', subplots=True)
```