Historical Data with Python, using Yahoo Finance data

This guide is prepared by information gathered on internet and some Youtube channels. It's just for fun, so advices or improves to add in this docs always been wellcome. Having in mind, some steps taking about the installation in case that the module importation didn't work, or other easy steps needed to some easy problems, I don't aggregate here.

Obviously, it's important remind you that we use Python and its libraries, in particular Pandas library and other called Yfinance.

Initial Setup

We start importing the modules able to import finance data, in which information ups as DataFrame structure, with columns related by the Price and its changes in the day, adjusted by the dividends and the volume of long and short positions.

```
import pandas as pd
import yfinance as yf
from yahoofinancials import YahooFinancials
```

First Method: Using yfinance

The module **yfinance** is now a very popular library that is very python friendly and can be used as a patch to pandas_datareader or a standalone library in itself. It has many potential uses and many people use it to download stock prices and also crypto prices. Without any further delay, let us execute the following code. We import data of Mastercard ('MA') since 2018 to 2021 completly

```
ma_df=yf.download('MA',
                  start='2018-01-01',
                  end='2020-12-31',
                  progress=False,
                 )
```

Maybe you need all the data available of the equity. In this case just avoid using the parameters of dates, and just use the name of the stock you want to get. Remember, in Jupyter if you want to know the other parameters has the function using *shift + tab* to look into it.

In [9]:
```
!pip install yfinance
```

In [1]:
```
import pandas as pd
import yfinance as yf

ma_df=yf.download('MA',
                  start='2018-01-01',
                  end='2020-12-31',
                  progress=False,
                 )
ma_df.head()
```

Out[1]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-01-02	152.009995	153.410004	151.119995	151.910004	148.318100	3168900
2018-01-03	152.289993	153.979996	152.160004	153.820007	150.182907	3732400
2018-01-04	154.539993	157.399994	154.320007	155.809998	152.125824	3247200
2018-01-05	156.199997	159.039993	156.190002	159.039993	155.279465	2747700
2018-01-08	158.580002	160.479996	158.080002	159.270004	155.748886	3143100

Multiple Stocks

Also, you can download multiple equities you need just using a list structure data, in which you adding the stocks names, as the same we did before.

In [2]:
```
multiple_eq=['AMZN','MA','LTMAQ','NSRGY']
df= yf.download(multiple_eq,
                start='2018-01-01',
                end='2020-12-31',
                progress=False,
               )
df.tail()
```

Out[2]:

	Adj Close				Close				High		...	Low				Open			
	AMZN	LTMAQ	MA	NSRGY	AMZN	LTMAQ	MA	NSRGY	AMZN	LTMAQ	...	MA	NSRGY	AMZN	LTMAQ	MA	NSRGY	AMZN	A
Date																			
2020-12-23	3185.270020	1.82	326.740112	108.450607	3185.270020	1.82	329.230011	113.949997	3210.129883	1.89	...	329.029999	113.529999	3205.000000	1.78	334.130005	114.139999	20
2020-12-24	3172.689941	1.87	333.458862	108.041359	3172.689941	1.87	336.000000	113.519997	3202.000000	1.90	...	330.350006	113.050003	3193.899902	1.86	330.649994	113.050003	14
2020-12-28	3283.959961	1.81	341.864838	110.668152	3283.959961	1.81	344.470001	116.279999	3304.000000	1.90	...	336.869995	116.089996	3194.000000	1.90	337.859985	116.089996	56
2020-12-29	3322.000000	1.80	344.038269	112.219490	3322.000000	1.80	346.660004	117.910004	3350.649902	1.82	...	344.760010	117.570000	3309.939941	1.82	347.779999	117.570000	48
2020-12-30	3285.850098	1.74	352.861023	112.847633	3285.850098	1.74	355.549988	118.570000	3342.100098	1.85	...	347.209991	118.000000	3341.000000	1.82	348.679993	118.139999	32

5 rows × 24 columns

In [13]:
```
df_returns = df["Adj Close"]
df_returns.tail()
```

Out[13]:

Date	AMZN	LTMAQ	MA	NSRGY
2020-12-23	3185.270020	1.82	326.740112	108.450607
2020-12-24	3172.689941	1.87	333.458862	108.041359
2020-12-28	3283.959961	1.81	341.864838	110.668152
2020-12-29	3322.000000	1.80	344.038269	112.219490
2020-12-30	3285.850098	1.74	352.861023	112.847633

Also, we can use the argument **auto-adjust = True**, so all the current prices are adjusted for potential corporate actions like splits.

In [14]:
```
multiple_eq=['AMZN','MA','LTMAQ','NSRGY']
df= yf.download(multiple_eq,
                start='2018-01-01',
                end='2020-12-31',
                progress=False,
                auto_adjust=True
               )
df.head()
```

Out[14]:

	Close				High				Low				Open			
	AMZN	LTMAQ	MA	NSRGY	AMZN	LTMAQ	MA	NSRGY	AMZN	LTMAQ	MA	NSRGY	AMZN	LTMAQ	MA	NSRGY
Date																
2018-01-02	1189.010010	13.706138	148.318085	76.915833	1190.000000	13.706138	149.782617	77.481725	1170.510010	13.297708	147.546756	76.664328	1172.000000	13.297708	148.415711	77.293096
2018-01-03	1204.199951	13.554165	150.182892	76.601448	1205.489990	13.687141	150.339097	76.790077	1188.300049	13.440184	148.562139	76.179277	1188.300049	13.658647	148.689055	76.215207
2018-01-04	1209.589966	13.734634	152.125870	76.646370	1215.869995	13.791625	153.678271	76.960751	1204.660034	13.554166	150.671110	76.547564	1205.000000	13.554166	150.885895	76.817030
2018-01-05	1229.140015	13.943597	155.279495	76.996674	1229.140015	13.991088	155.279495	77.059550	1210.000000	13.734633	152.496892	76.646363	1217.510010	13.734633	152.506651	76.888883
2018-01-08	1246.869995	14.105070	155.748871	76.781105	1253.079956	14.209552	156.932112	77.068540	1232.030029	13.886607	154.585177	76.682299	1236.000000	13.924601	155.074123	77.059556

Ticker

Apart from using *yf.download* function, we can also use the *ticker* module, executing to download the last 5year stock prices of some equity

In [34]:
```
ticker = yf.Ticker("AMZN")
amzn_df = ticker.history(period = "5y")
amzn_df["Close"].plot(title = "AMZN's stock price")
```

Out[34]:
```
<AxesSubplot:title={'center':"AMZN's stock price"}, xlabel='Date'>
```

Also, it's important to mention the aditional information we can extract by using *Ticker* module. For mentioned just two of these:

- **info**: Prints a JSON formatter output with a lot of information about the company, starting from their business full name, summary, industry, exchanges listed on with country and time zone. It also comes equipped with the beta coefficient (Remember: Tells us the systematic risk of the equity by changes on market)
- **recommendations**: Contains a historial list of recommendations made by different analysts regarding the stock and whether to buy, sell, hold or give suggestions on it.
- **actions**: Displays actions like splits and dividends
- **major_holders**: Major holders of the share along with other several details.
- **institutional_holders**: Institutional holders of a particular share.
- **calendar**: Incoming events such as the earnings and you can even add this to your google calendar through code. Basically, it shows the important dividend dates for a company.

```
yf.Ticker("AMZN").info
yf.Ticker("AMZN").recommendations
yf.Ticker("AMZN").actions
yf.Ticker("AMZN").major_holders
yf.Ticker("AMZN").institutional_holders
yf.Ticker("AMZN").calendar
```