Stocks Price EDA of Apple, Google, Microsoft & Amazon with IEX(The Investor Exchange)

- (By Lakshmi Date-5th April 2023)

Table of Contents

1. Introduction

Stock Market Data Analysis

Stock Market Analysis and Prediction is the project related to Exploratory data analysis(EDA), Data visualization and Predictive analysis using data, provided by The Investors Exchange (IEX).

Basic Analysis of Stock Information

In this section we'll go over how to handle requesting stock information with pandas, and how to analyze basic attributes of a stock.

2. Importing Packages

In [1]:
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in c:\users\pankaj singh\appdata\local\programs
\python\python310\lib\site-packages (23.0.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import numpy as np
import pandas as pd
```

Importing packages for Visualization

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
%matplotlib inline
```

For reading stock data from The Investors Exchange (IEX)

In [4]:
```
pip install pandas_datareader
```

```
Requirement already satisfied: pandas_datareader in c:\users\pankaj singh\appdata
\local\programs\python\python310\lib\site-packages (0.10.0)
Requirement already satisfied: pandas>=0.23 in c:\users\pankaj singh\appdata\local
\programs\python\python310\lib\site-packages (from pandas_datareader) (1.4.1)
Requirement already satisfied: lxml in c:\users\pankaj singh\appdata\local\program
s\python\python310\lib\site-packages (from pandas_datareader) (4.9.2)
Requirement already satisfied: requests>=2.19.0 in c:\users\pankaj singh\appdata\l
ocal\programs\python\python310\lib\site-packages (from pandas_datareader) (2.27.1)
Requirement already satisfied: numpy>=1.21.0 in c:\users\pankaj singh\appdata\loca
l\programs\python\python310\lib\site-packages (from pandas>=0.23->pandas_datareade
r) (1.22.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\pankaj singh\app
data\local\programs\python\python310\lib\site-packages (from pandas>=0.23->pandas_
datareader) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pankaj singh\appdata\local
\programs\python\python310\lib\site-packages (from pandas>=0.23->pandas_datareade
r) (2022.7.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\pankaj singh\appdata\local
\programs\python\python310\lib\site-packages (from requests>=2.19.0->pandas_datare
ader) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\pankaj singh
\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0-
>pandas_datareader) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pankaj singh\appdata
\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->pandas_
datareader) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\pankaj singh\appd
ata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->pand
as_datareader) (1.26.8)
Requirement already satisfied: six>=1.5 in c:\users\pankaj singh\appdata\local\pro
grams\python\python310\lib\site-packages (from python-dateutil>=2.8.1->pandas>=0.2
3->pandas_datareader) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

2.1 Data Loading

- **(From IEX website into the python)**

In [5]:
```
import os
import pandas_datareader as pdr
from datetime import datetime
```

In [6]:
```
os.environ["IEX_API_KEY"] = "sk_ab4a9c57e15b49f0961a6a7279f60dc0"
```

Let's use iexfinanceThe Investors Exchange (IEX) and pandas to grab some data for some Tech company stocks.

In [7]:
```
#The tech stocks we'll use for analysis:-
tech_stocks = ['AAPL','GOOG','MSFT','AMZN']
```

In [8]:
```
# Setup END and START Time for Data Load:-

end = datetime.now()
end
```

Out[8]:
```
datetime.datetime(2023, 4, 5, 22, 20, 57, 733323)
```

Bring the Data from last year till now

In [9]:
```
start = datetime(end.year -1, end.month, end.day)
start
```

Out[9]:
```
datetime.datetime(2022, 4, 5, 0, 0)
```

In [10]:
```
#For loop for garbing iex finance data and setting as a dataframe

for stock in tech_stocks:
      globals()[stock] = pdr.DataReader(stock,'iex',start,end, api_key=os.getenv('
```

2.2 Introduction of Data Imported

- **(From IEX website into the python)**

APPLE

In [11]:
```
AAPL.head()
```

Out[11]:

date	open	high	low	close	volume
2022-04-05	177.50	178.30	174.415	175.06	73401786
2022-04-06	172.36	173.63	170.130	171.83	89058782
2022-04-07	171.16	173.36	169.850	172.14	77594650
2022-04-08	171.78	171.78	169.200	170.09	76575508
2022-04-11	168.71	169.03	165.500	165.75	72246706

In [12]:
```
AAPL.count()
```

Out[12]:
```
open      251
high      251
low       251
close     251
volume    251
dtype: int64
```

In [13]:
```
AAPL.describe()
```

Out[13]:

	open	high	low	close	volume
count	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	149.519076	151.574225	147.627531	149.660080	8.115180e+07
std	11.086980	10.957394	11.114192	11.019197	2.397366e+07
min	126.010000	127.770000	124.170000	125.020000	3.519586e+07
25%	142.110000	143.855000	139.950000	142.465000	6.605844e+07
50%	148.870000	150.920000	147.240000	149.350000	7.598192e+07
75%	156.275000	158.155000	154.165000	156.780000	8.908621e+07
max	177.500000	178.300000	174.415000	175.060000	1.826020e+08

In [14]:
```
AAPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 251 entries, 2022-04-05 to 2023-04-04
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    251 non-null    float64
 1   high    251 non-null    float64
 2   low     251 non-null    float64
 3   close   251 non-null    float64
 4   volume  251 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 11.8+ KB
```

Google

In [15]:
```
GOOG.head()
```

Out[15]:

date	open	high	low	close	volume
2022-04-05	143.3995	143.59000	140.943500	141.0630	19255560
2022-04-06	139.1615	139.84850	136.418110	137.1760	23574600
2022-04-07	136.6180	137.70150	134.857250	136.4650	19448540
2022-04-08	136.2500	136.25000	133.752505	134.0105	16434480
2022-04-11	132.9000	132.93918	129.617500	129.7965	24187340

In [16]:
```
GOOG.count()
```

Out[16]:
```
open      251
high      251
low       251
close     251
volume    251
dtype: int64
```

In [17]:
```
GOOG.describe()
```

Out[17]:

	open	high	low	close	volume
count	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	105.085451	106.665789	103.701534	105.166618	2.870115e+07
std	11.600865	11.666535	11.359628	11.462983	1.123245e+07
min	85.510000	86.550000	83.450000	83.490000	8.567819e+06
25%	95.760000	97.350000	94.470000	95.840000	2.185202e+07
50%	102.880000	104.220000	101.860000	103.630000	2.603392e+07
75%	113.983250	116.341000	112.562000	114.574250	3.235121e+07
max	143.399500	143.590000	140.943500	141.063000	9.779857e+07

In [18]:
```
GOOG.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 251 entries, 2022-04-05 to 2023-04-04
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    251 non-null    float64
 1   high    251 non-null    float64
 2   low     251 non-null    float64
 3   close   251 non-null    float64
 4   volume  251 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 11.8+ KB
```

MICROSOFT

In [19]:
```
MSFT.head()
```

Out[19]:

date	open	high	low	close	volume
2022-04-05	313.27	314.865	309.87	310.88	23156719
2022-04-06	305.19	307.000	296.71	299.50	40110372
2022-04-07	296.66	303.650	296.35	301.37	31411155
2022-04-08	300.44	301.120	296.28	296.97	24361917
2022-04-11	291.79	292.610	285.00	285.26	34569264

In [20]:
```
MSFT.count()
```

Out[20]:
```
open      251
high      251
low       251
close     251
volume    251
dtype: int64
```

In [21]:
```
MSFT.describe()
```

Out[21]:

	open	high	low	close	volume
count	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	257.361745	260.623093	254.106635	257.411235	2.982603e+07
std	18.672160	18.543884	18.404496	18.486580	1.024924e+07
min	217.550000	220.410000	213.431000	214.250000	9.200772e+06
25%	243.160000	245.305000	240.265000	242.355000	2.284654e+07
50%	256.390000	259.720000	252.770000	255.140000	2.785291e+07
75%	271.885000	274.710000	267.715000	271.595000	3.363343e+07
max	313.270000	314.865000	309.870000	310.880000	8.610199e+07

In [22]:
```
MSFT.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 251 entries, 2022-04-05 to 2023-04-04
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    251 non-null    float64
 1   high    251 non-null    float64
 2   low     251 non-null    float64
 3   close   251 non-null    float64
 4   volume  251 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 11.8+ KB
```

AMAZON

In [23]:
```
AMZN.head()
```

Out[23]:

date	open	high	low	close	volume
2022-04-05	167.742	168.111	163.266	164.055	53728560.0
2022-04-06	161.651	162.200	157.255	158.756	79055760.0
2022-04-07	158.400	160.079	154.512	157.785	68136780.0
2022-04-08	156.750	157.369	154.231	154.461	46001660.0
2022-04-11	152.713	154.137	150.535	151.122	52112340.0

In [24]:
```
AMZN.count()
```

Out[24]:
```
open      251
high      251
low       251
close     251
volume    251
dtype: int64
```

In [25]:
```
AMZN.describe()
```

Out[25]:

	open	high	low	close	volume
count	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	111.367845	113.284893	109.309333	111.295876	7.054671e+07
std	18.731238	18.906272	18.384658	18.547055	3.268914e+07
min	82.800000	83.480000	81.430000	81.820000	3.520700e+04
25%	96.095000	97.537000	94.195000	96.260000	5.286433e+07
50%	108.200000	112.130000	106.324000	108.859000	6.396932e+07
75%	122.550000	124.248000	120.333500	122.385000	8.296061e+07
max	167.742000	168.111000	163.266000	164.055000	2.726617e+08

In [26]:
```
AMZN.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 251 entries, 2022-04-05 to 2023-04-04
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    251 non-null    float64
 1   high    251 non-null    float64
 2   low     251 non-null    float64
 3   close   251 non-null    float64
 4   volume  251 non-null    float64
dtypes: float64(5)
memory usage: 11.8+ KB
```

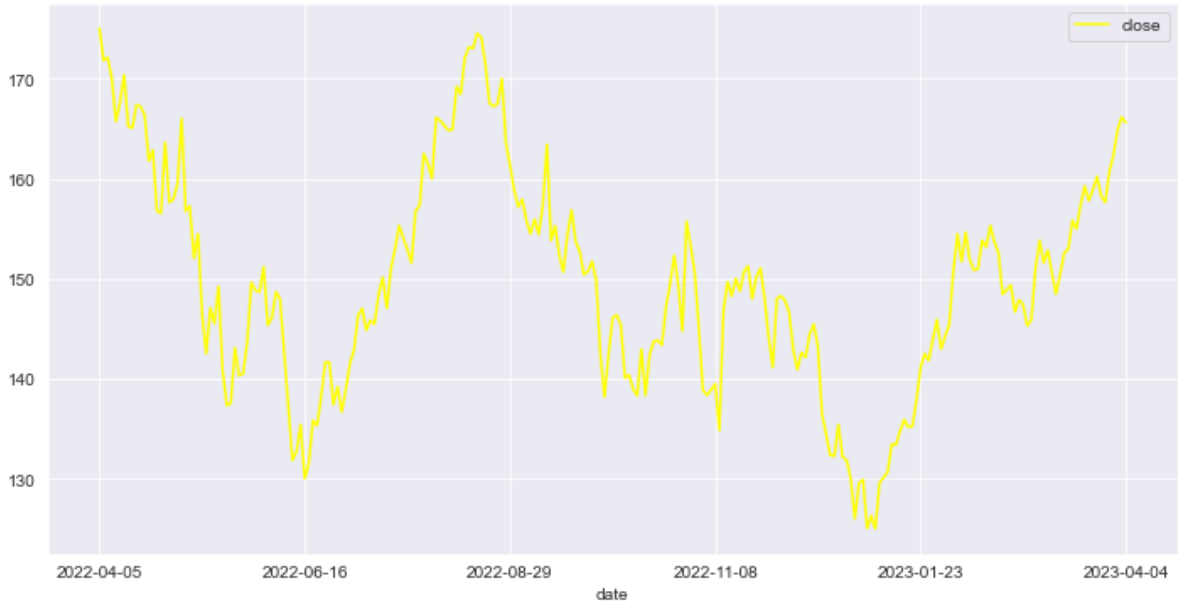
Now that we've seen the DataFrame, let's go ahead and plot out the volume and closing price of the stocks

3.1 Let's see a historical view of the closing price over past 1 year

APPLE

In [27]:
```
AAPL['close'].plot(legend = True, figsize = (12,6), color='yellow')
```

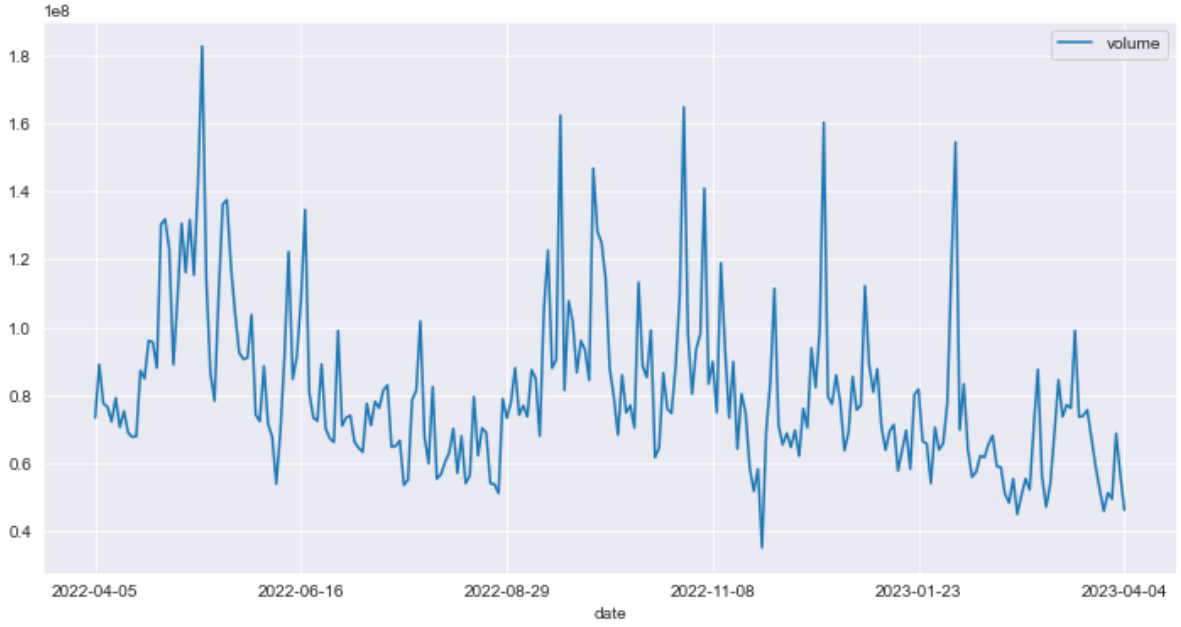
Out[27]:
```
<AxesSubplot:xlabel='date'>
```



There is a Deflation the Apple stock price from **177 to 167(i.e, around 10)**

Now let's plot the total volume of stock being traded each day over the past 1 years

In [28]:
```
AAPL['volume'].plot(legend = True, figsize = (12,6))
plt.show()
```

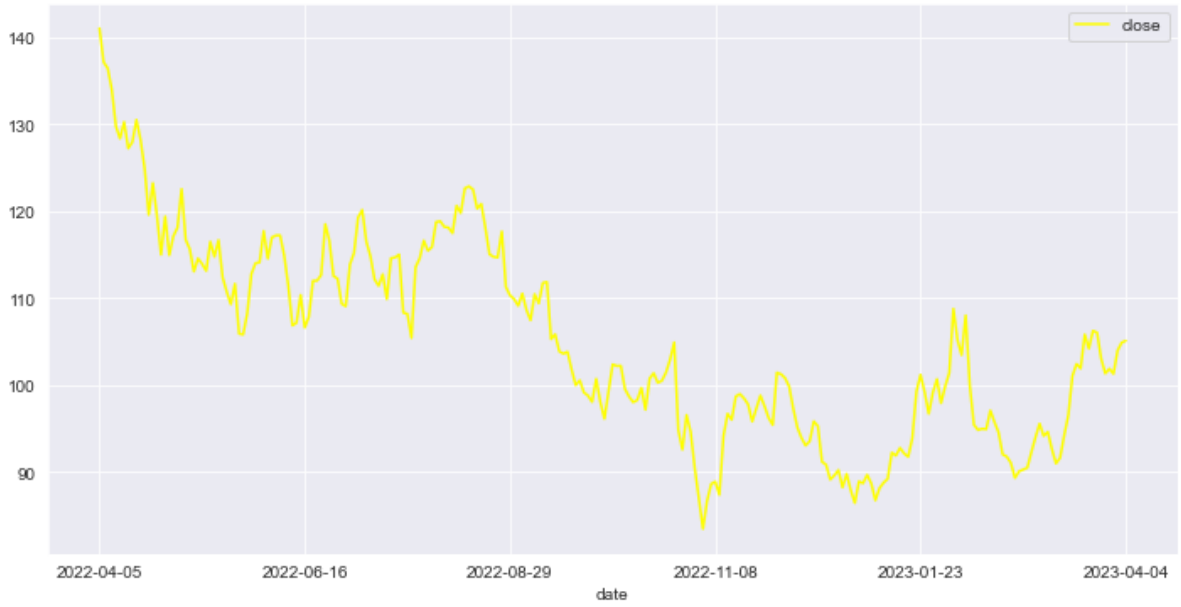


There is a Deflation the Apple stock trade volume from **7 to 4.8(i.e, around 2.2)** till now.
Maximum Trade of stock happend at **May 2022**.

Google

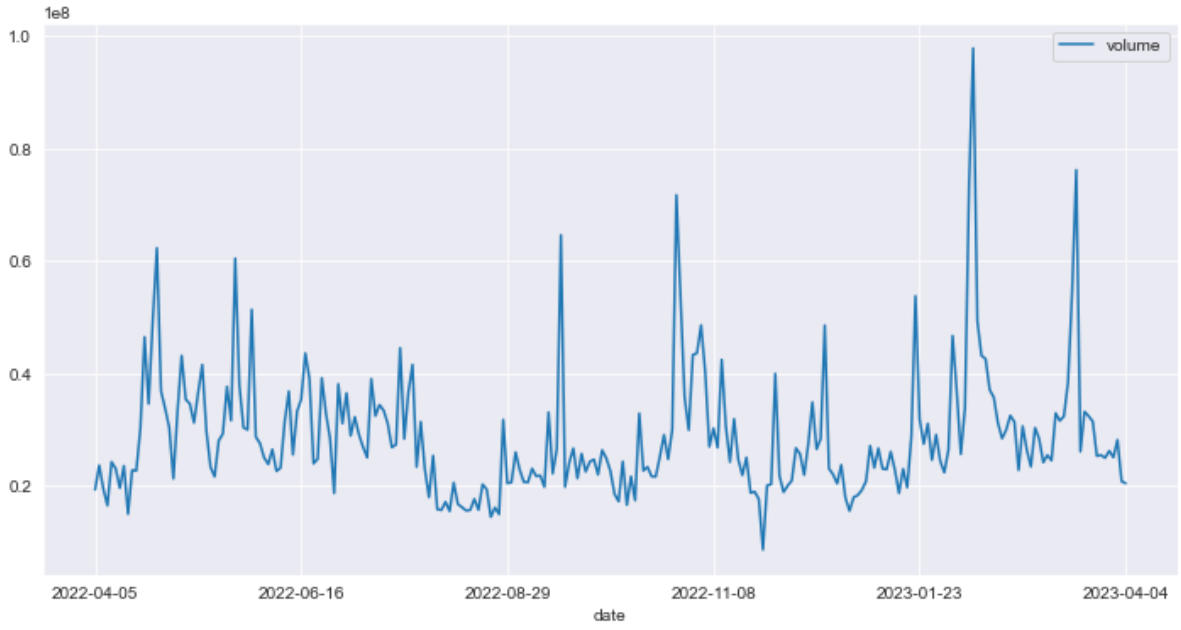
In [29]:
```
GOOG['close'].plot(legend = True, figsize = (12,6), color='yellow')
```

Out[29]:
```
<AxesSubplot:xlabel='date'>
```



There is a Deflation the Google stock price from **142 to 105(i.e, around 37)**

In [30]:
```
GOOG['volume'].plot(legend = True, figsize = (12,6))
plt.show()
```

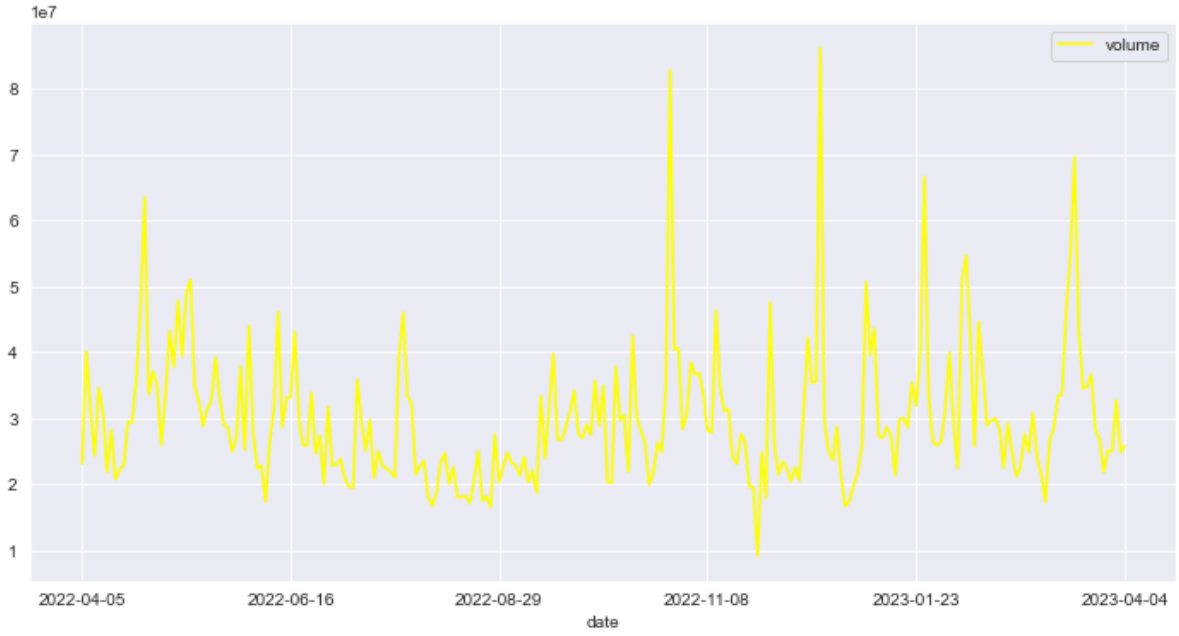


There is an Inflation in the Google stock trade volume from **1.9 to 2.1(i.e, around 0.2)**.
Maximum Trade of stock happend at **Last of January 2023/Start of Feburay 2023**.

MICROSOFT

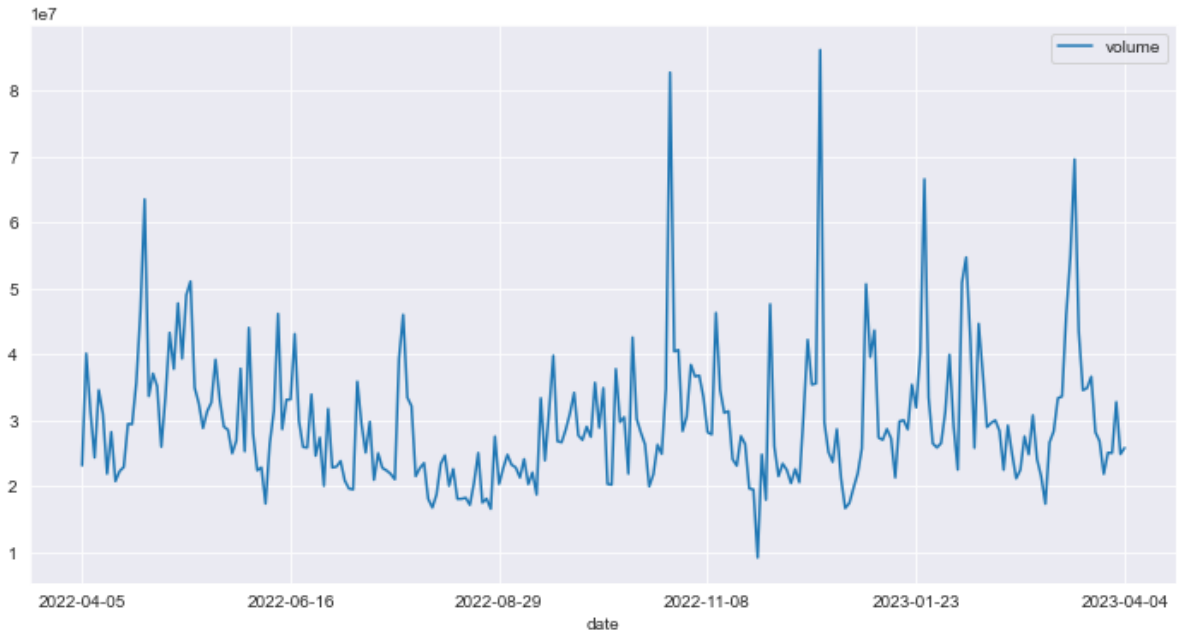
In [31]:
```
MSFT['volume'].plot(legend = True, figsize = (12,6), color='yellow')
```

Out[31]:
```
<AxesSubplot:xlabel='date'>
```



There is an Inflation the Microsoft stock price from **2.4 to 2.6(i.e, around 0.2)**

In [32]:
```
MSFT['volume'].plot(legend = True, figsize = (12,6))
plt.show()
```

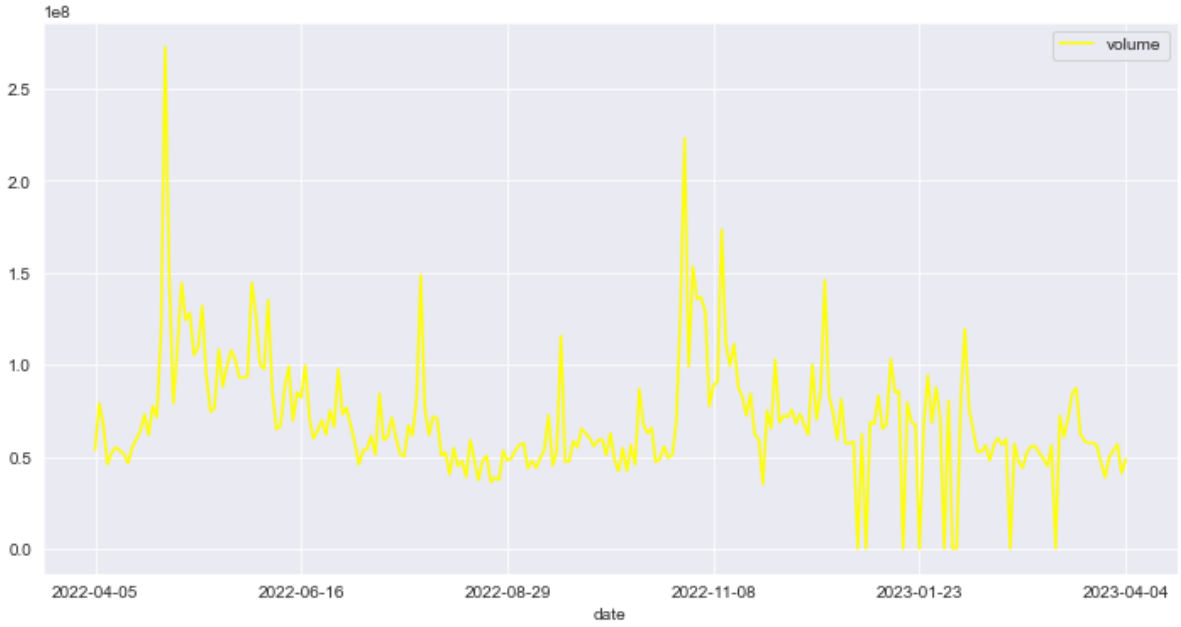


There is an Inflation the Apple stock price from **2.4 to 2.6(i.e, around 0.2)** Maximum Trade of stocks happened at **December 2022**

AMAZON

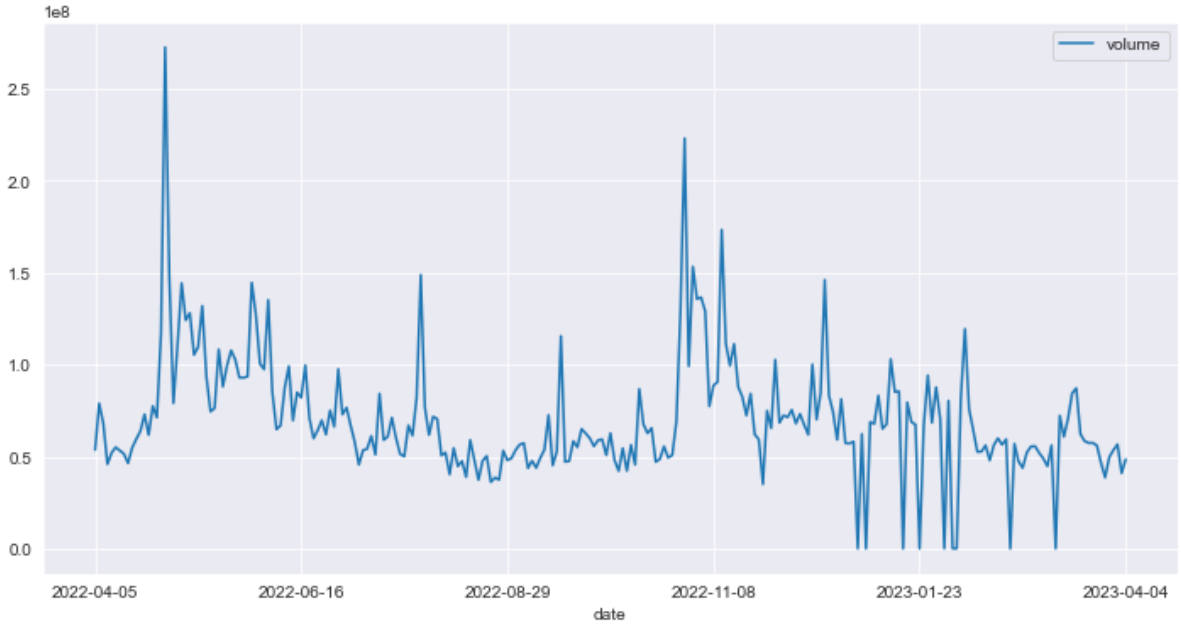
In [33]:
```
AMZN['volume'].plot(legend = True, figsize = (12,6), color='yellow')
```

Out[33]:
```
<AxesSubplot:xlabel='date'>
```



There is a Deflation the Amazon stock price from **0.51 to 0.5(i.e, around 0.01)**

In [34]:
```
AMZN['volume'].plot(legend = True, figsize = (12,6))
plt.show()
```



There is a Deflation the Apple stock trade from **0.51 to 0.5(i.e, around 0.01)** Maximum Trade of stock happened at **last of April 2022**

Now that we've seen the visualisation for closing price & the volume traded each day, let's go ahead and calculate the moving aerage for the stock

3.2 'Moving Average- MA'

A moving average (MA) is a widely used indicator intechnical analysis that helps smooth out price action by filtering out the "noise" from random price fluctuation. it is a trend-following, or lagging, indicator because it is based on the past prices.

APPLE

In [35]:
```
#Lets go ahead and plot out several moving averages

ma_day = [10,20,50]

for ma in ma_day:
    col_name = "MA for %s days" %(str(ma))
    AAPL[col_name] = pd.DataFrame.rolling(AAPL['close'],ma).mean()
```

In [36]:
```
AAPL.head(50)
```

Out[36]:

date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-04-05	177.500	178.300	174.415	175.06	73401786	NaN	NaN	NaN
2022-04-06	172.360	173.630	170.130	171.83	89058782	NaN	NaN	NaN
2022-04-07	171.160	173.360	169.850	172.14	77594650	NaN	NaN	NaN
2022-04-08	171.780	171.780	169.200	170.09	76575508	NaN	NaN	NaN
2022-04-11	168.710	169.030	165.500	165.75	72246706	NaN	NaN	NaN
2022-04-12	168.020	169.870	166.640	167.66	79265181	NaN	NaN	NaN
2022-04-13	167.390	171.040	166.770	170.40	70618925	NaN	NaN	NaN
2022-04-14	170.620	171.270	165.040	165.29	75329376	NaN	NaN	NaN
2022-04-18	163.920	166.598	163.570	165.07	69023941	NaN	NaN	NaN
2022-04-19	165.020	167.820	163.910	167.40	67723833	169.069	NaN	NaN
2022-04-20	168.760	168.880	166.100	167.23	67929814	168.286	NaN	NaN
2022-04-21	168.910	171.530	165.910	166.42	87227768	167.745	NaN	NaN
2022-04-22	166.460	167.870	161.500	161.79	84882424	166.710	NaN	NaN
2022-04-25	161.120	163.170	158.460	162.88	96046376	165.989	NaN	NaN
2022-04-26	162.250	162.340	156.720	156.80	95623240	165.094	NaN	NaN
2022-04-27	155.910	159.790	155.380	156.57	88063191	163.985	NaN	NaN
2022-04-28	159.250	164.515	158.930	163.64	130216792	163.309	NaN	NaN
2022-04-29	161.840	166.200	157.250	157.65	131747571	162.545	NaN	NaN
2022-05-02	156.710	158.230	153.270	157.96	123055265	161.834	NaN	NaN
2022-05-03	158.150	160.710	156.320	159.48	88966526	161.042	165.0555	NaN
2022-05-04	159.670	166.480	159.260	166.02	108256503	160.921	164.6035	NaN
2022-05-05	163.850	164.080	154.950	156.77	130525275	159.956	163.8505	NaN

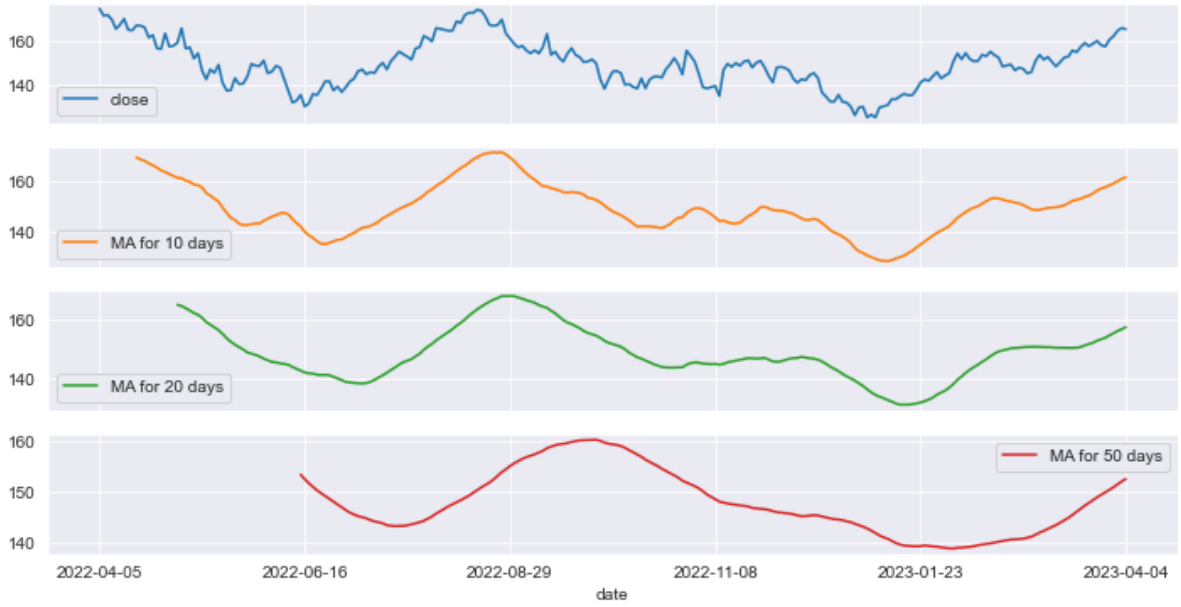
date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-05-06	156.010	159.440	154.180	157.28	116124647	159.505	163.1075	NaN
2022-05-09	154.925	155.830	151.490	152.06	131577921	158.423	162.2060	NaN
2022-05-10	155.520	156.740	152.930	154.51	115366736	158.194	161.6440	NaN
2022-05-11	153.500	155.450	145.810	146.50	142689825	157.187	160.5860	NaN
2022-05-12	142.770	146.200	138.800	142.56	182602041	155.079	159.1940	NaN
2022-05-13	144.590	148.105	143.110	147.11	113990852	154.025	158.2850	NaN
2022-05-16	145.550	147.520	144.180	145.54	86643781	152.783	157.3085	NaN
2022-05-17	148.860	149.770	146.680	149.24	78336254	151.759	156.4005	NaN
2022-05-18	146.850	147.360	139.900	140.82	109742890	149.239	155.0800	NaN
2022-05-19	139.880	141.660	136.600	137.35	136095640	147.297	153.6265	NaN
2022-05-20	139.090	140.700	132.610	137.59	137426125	145.328	152.4165	NaN
2022-05-23	137.790	143.260	137.650	143.11	117726265	144.433	151.4280	NaN
2022-05-24	140.805	141.970	137.330	140.36	104132746	143.018	150.6060	NaN
2022-05-25	138.430	141.785	138.340	140.52	92482696	142.420	149.8035	NaN
2022-05-26	137.390	144.340	137.140	143.78	90601548	142.542	148.8105	NaN
2022-05-27	145.390	149.680	145.260	149.64	90978503	142.795	148.4100	NaN
2022-05-31	149.070	150.660	146.840	148.84	103718416	143.125	147.9540	NaN
2022-06-01	149.900	151.740	147.680	148.71	74286635	143.072	147.4155	NaN
2022-06-02	147.830	151.270	146.860	151.21	72348055	144.111	146.6750	NaN
2022-06-03	146.900	147.970	144.460	145.38	88570289	144.914	146.1055	NaN
2022-06-06	147.030	148.569	144.900	146.14	71598380	145.769	145.5485	NaN
2022-06-07	144.345	149.000	144.100	148.71	67808150	146.329	145.3810	NaN

date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-06-08	148.580	149.870	147.460	147.96	53950201	147.089	145.0535	NaN
2022-06-09	147.080	147.950	142.530	142.64	69472976	147.301	144.8605	NaN
2022-06-10	140.280	140.760	137.060	137.13	91566637	146.636	144.5890	NaN
2022-06-13	132.870	135.200	131.440	131.88	122207099	144.860	143.8275	NaN
2022-06-14	133.130	133.890	131.480	132.76	84784326	143.252	143.1885	NaN
2022-06-15	134.290	137.340	132.160	135.43	91532972	141.924	142.4980	153.3732

In [37]:
```
AAPL[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```

There is a Deflation the Apple stock price & moving average of 10,20,50 days

In [38]:
```
AAPL[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```



Apple closing Price - **From 179 to 167** MA for 10 days - **From 168 to 161** MA for 20 days - **From 162 to 158** Ma for 50 days - **From 154 to 152**

Google

In [39]:
```
ma_day = [10,20,50]

for ma1 in ma_day:
    col_name = "MA for %s days" %(str(ma1))
    GOOG[col_name] = pd.DataFrame.rolling(GOOG['close'],ma1).mean()
```

In [40]:
```
GOOG.head(50)
```

Out[40]:

date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-04-05	143.39950	143.590000	140.943500	141.0630	19255560	NaN	NaN	NaN
2022-04-06	139.16150	139.848500	136.418110	137.1760	23574600	NaN	NaN	NaN
2022-04-07	136.61800	137.701500	134.857250	136.4650	19448540	NaN	NaN	NaN
2022-04-08	136.25000	136.250000	133.752505	134.0105	16434480	NaN	NaN	NaN
2022-04-11	132.90000	132.939180	129.617500	129.7965	24187340	NaN	NaN	NaN
2022-04-12	132.42350	132.423500	127.576000	128.3745	23003220	NaN	NaN	NaN
2022-04-13	128.62650	130.655750	128.438595	130.2860	19542960	NaN	NaN	NaN
2022-04-14	130.64950	130.710250	127.111500	127.2530	23483360	NaN	NaN	NaN
2022-04-18	127.41000	128.712000	126.578465	127.9610	14917200	NaN	NaN	NaN
2022-04-19	128.07700	130.903750	127.451500	130.5310	22719300	132.29165	NaN	NaN
2022-04-20	131.28400	131.923500	127.894055	128.2455	22609380	131.00990	NaN	NaN
2022-04-21	129.35000	130.307500	124.650000	124.9375	30157540	129.78605	NaN	NaN
2022-04-22	125.00000	125.452000	119.140500	119.6140	46410300	128.10095	NaN	NaN
2022-04-25	119.42950	123.278000	118.769250	123.2500	34521800	127.02490	NaN	NaN
2022-04-26	122.75000	122.750000	119.161850	119.5060	49393040	125.99585	NaN	NaN
2022-04-27	114.37300	117.500000	113.124250	115.0205	62238120	124.66045	NaN	NaN
2022-04-28	117.11500	120.438500	115.143890	119.4115	36790940	123.57300	NaN	NaN
2022-04-29	117.57800	118.960000	114.694000	114.9665	33693100	122.34435	NaN	NaN
2022-05-02	113.90650	117.339500	113.399500	117.1570	30279640	121.26395	NaN	NaN
2022-05-03	116.76500	119.300000	116.627000	118.1295	21215740	120.02380	126.157725	NaN
2022-05-04	118.00350	123.143000	115.738500	122.5750	33231460	119.45675	125.233325	NaN
2022-05-05	120.22050	121.233250	115.182500	116.7465	43089040	118.63765	124.211850	NaN

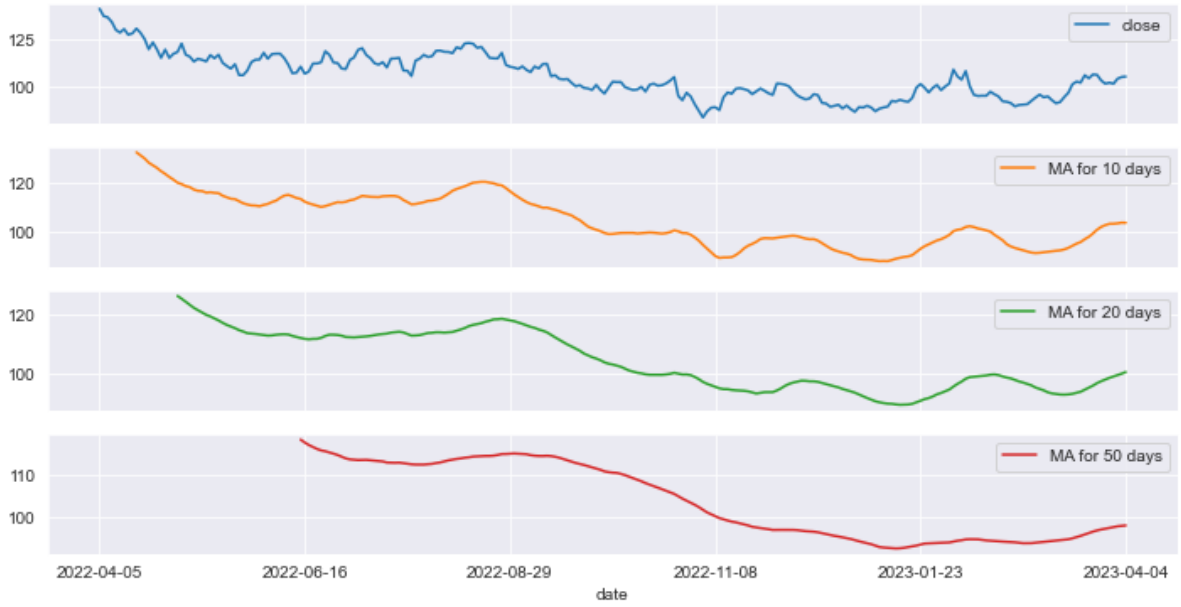
date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-05-06	115.51900	117.498500	114.143000	115.6600	35309480	118.24225	123.171600	NaN
2022-05-09	113.30350	115.562905	112.551500	113.0840	34520960	117.22565	122.125275	NaN
2022-05-10	116.04050	116.691000	113.383285	114.5845	31157780	116.73350	121.364675	NaN
2022-05-11	113.71050	116.671000	113.650000	113.9610	36501640	116.62755	120.644000	NaN
2022-05-12	111.93800	114.856500	110.113500	113.1610	41464880	116.00250	119.787750	NaN
2022-05-13	114.84550	118.085000	114.000000	116.5155	29737560	116.15740	119.250875	NaN
2022-05-16	115.38400	116.607500	114.335000	114.7925	23282380	115.92095	118.592450	NaN
2022-05-17	117.22750	117.227500	115.337500	116.7015	21576080	115.77815	117.900975	NaN
2022-05-18	115.23750	115.695670	112.142000	112.4010	27982760	114.76075	117.108750	NaN
2022-05-19	111.84100	113.587500	110.468000	110.7455	29191740	114.16065	116.399150	NaN
2022-05-20	112.08550	112.550000	106.373000	109.3130	37586020	113.52595	115.884100	NaN
2022-05-23	110.10400	112.005500	109.154250	111.6665	31558220	113.38420	115.304925	NaN
2022-05-24	106.37750	106.395000	102.208000	105.9260	60386380	112.51835	114.625925	NaN
2022-05-25	105.14200	106.544690	104.211250	105.8395	37899340	111.70620	114.166875	NaN
2022-05-26	106.05050	108.955250	105.488000	108.2960	30287480	111.21970	113.611100	NaN
2022-05-27	109.78850	112.868000	109.550000	112.7990	29924420	110.84805	113.502725	NaN
2022-05-31	113.07900	116.433500	112.572500	114.0390	51301920	110.77270	113.346825	NaN
2022-06-01	114.93150	117.399000	113.550500	114.1370	28629280	110.51625	113.147200	NaN
2022-06-02	114.18800	117.898000	113.308000	117.7460	27495400	111.05075	112.905750	NaN
2022-06-03	115.99250	116.364500	113.668000	114.5640	24952080	111.43260	112.796625	NaN
2022-06-06	116.74250	119.398480	116.528290	117.0105	23786720	112.20235	112.864150	NaN
2022-06-07	115.64800	117.748625	115.125505	117.2295	26413540	112.75865	113.071425	NaN

date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days
2022-06-08	116.87650	118.646000	116.696750	117.2380	22544260	113.88985	113.204100	NaN
2022-06-09	116.34150	118.350000	114.867000	114.9180	23141600	114.79770	113.251950	NaN
2022-06-10	112.78125	113.497000	110.861000	111.4275	31349740	115.11085	113.165275	NaN
2022-06-13	107.44600	109.218500	106.588055	106.8765	36756200	114.51860	112.683325	NaN
2022-06-14	106.89000	108.457500	106.352000	107.1940	25480940	113.83410	112.303400	NaN
2022-06-15	108.89950	112.063000	108.118750	110.3905	33192020	113.45945	111.987850	118.41386

In [41]:
```
GOOG[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```

There is a Deflation the Google stock price from **141 to 105(i.e, around 36)**

In [42]:
```
GOOG[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```



Google closing Price - **From 141 to 105** MA for 10 days - **From 125 to 102** MA for 20 days - **From 123 to 100.1** Ma for 50 days - **From 115 to 98**

MICROSOFT

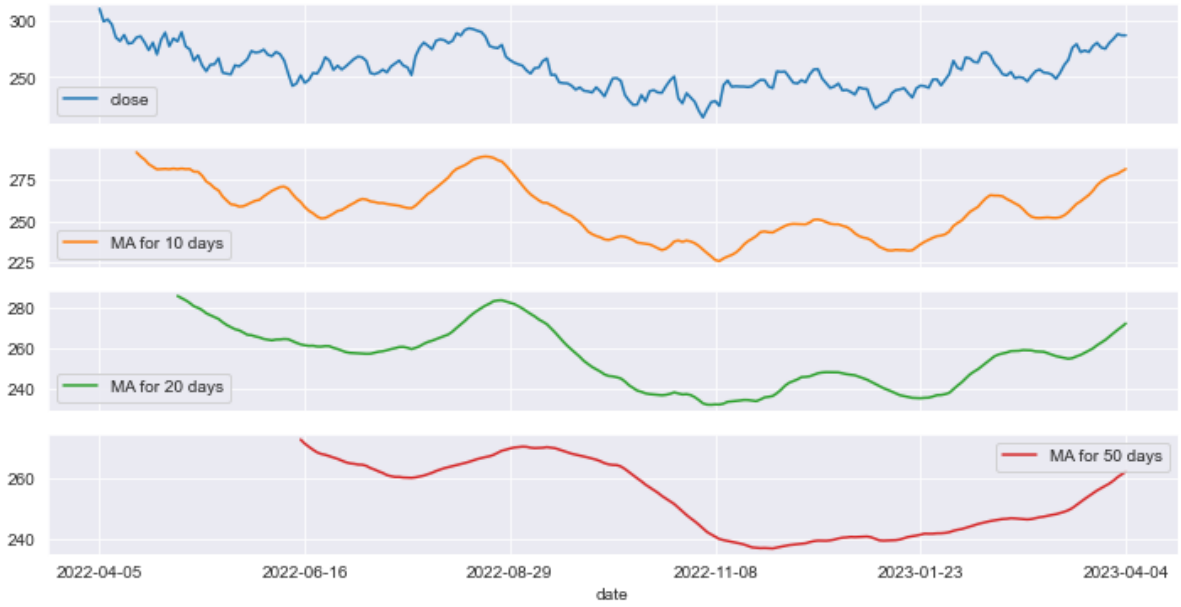
In [43]:
```
ma_day = [10,20,50]

for ma2 in ma_day:
    col_name = "MA for %s days" %(str(ma2))
    MSFT[col_name] = pd.DataFrame.rolling(MSFT['close'],ma2).mean()
```

In [44]:
```
MSFT[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```

There is a deflation the Microsoft stock price from **314 to 288(i.e, around 26)**

In [45]:
```
MSFT[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```



Apple closing Price - **From 314 to 288** MA for 10 days - **From 280 to 280.5** MA for 20 days - **From 282 to 274** Ma for 50 days - **From 272 to 261**

AMAZON

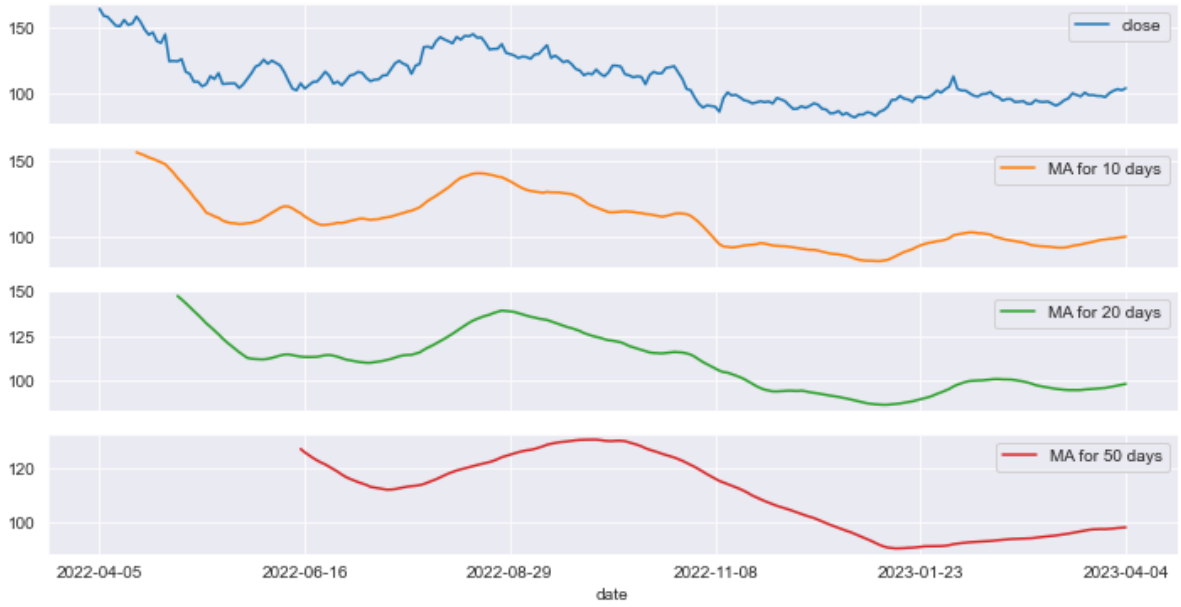
In [46]:
```
ma_day = [10,20,50]

for ma3 in ma_day:
    col_name = "MA for %s days" %(str(ma3))
    AMZN[col_name] = pd.DataFrame.rolling(AMZN['close'],ma3).mean()
```

In [47]:
```
AMZN[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```

There is a Deflation the Apple stock price from **164 to 106(i.e, around 58)**

In [48]:
```
AMZN[['close','MA for 10 days','MA for 20 days','MA for 50 days']].plot(subplots =
plt.show()
```



Apple closing Price - **From 164 to 106** MA for 10 days - **From 151 to 100.1** MA for 20 days - **From 149 to 99** Ma for 50 days - **From 124 to 98**

3.3 Daily Return Analysis

Now that we've done some baseline analysis, let's go ahead and dive a little deeper. We're now going to analyze the risk of the stock. In order to do so we'll need to take a closer look at the daily changes of the stock, and not just its absolute value. Let's go ahead and use pandas to retrieve the daily returns for the Apple stock.

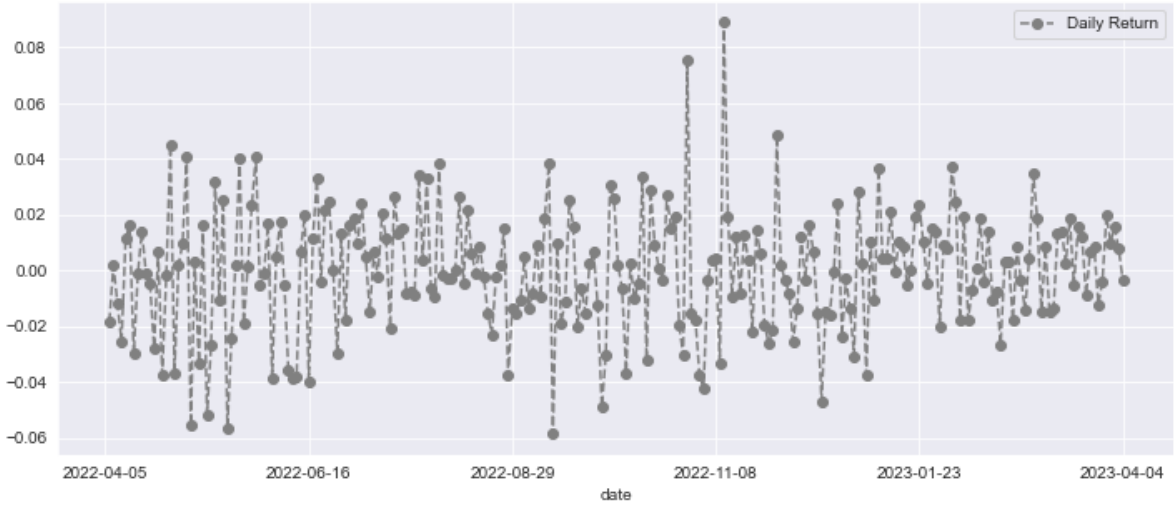
In [49]:
```
# use pct_change to find the percent change for each day
AAPL['Daily Return'] = AAPL['close'].pct_change()
```

In [50]:
```
AAPL.head()
```

Out[50]:

date	open	high	low	close	volume	MA for 10 days	MA for 20 days	MA for 50 days	Daily Return
2022-04-05	177.50	178.30	174.415	175.06	73401786	NaN	NaN	NaN	NaN
2022-04-06	172.36	173.63	170.130	171.83	89058782	NaN	NaN	NaN	-0.018451
2022-04-07	171.16	173.36	169.850	172.14	77594650	NaN	NaN	NaN	0.001804
2022-04-08	171.78	171.78	169.200	170.09	76575508	NaN	NaN	NaN	-0.011909
2022-04-11	168.71	169.03	165.500	165.75	72246706	NaN	NaN	NaN	-0.025516

In [51]:
```
# plot the daily return percentage
AAPL['Daily Return'].plot(figsize=(12,5),legend=True,linestyle='--',marker='o',col
plt.show()
```



There is Inflation in Daily Returns of Apple stock shifted from **-0.019 to -0.009**

Now let's get an overall look at the average daily return using a histogram. We'll use seaborn to create both a histogram and kde plot on the same figure.

In [52]:
```
# Note the use of dropna() here, otherwise the NaN values can't be read by seaborn
sns.distplot(AAPL['Daily Return'].dropna(),bins=100,color='grey')
plt.show()
```

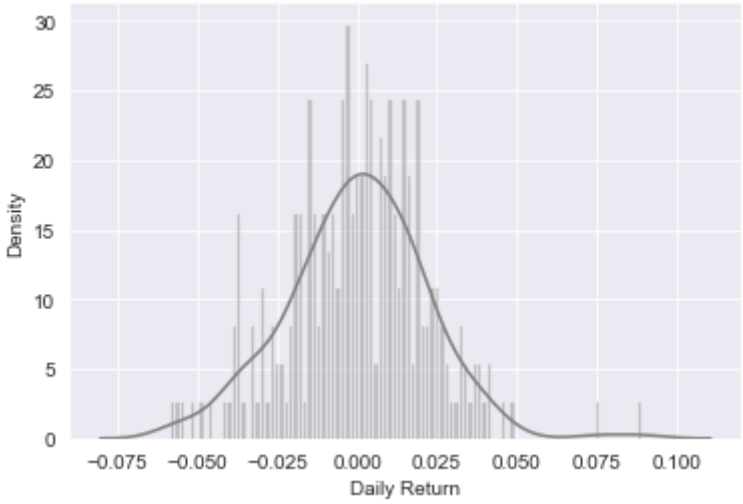
```
C:\Users\Pankaj Singh\AppData\Local\Temp\ipykernel_11660\72655809.py:2: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(AAPL['Daily Return'].dropna(),bins=100,color='grey')
```



Now want to analyze the returns of all the stocks in our list. Let's go ahead and build a DataFrame with all the ['close'] columns for each of the stocks dataframes.

In [53]:
```
closing_df1 = pd.DataFrame(AAPL['close'])
close1  = closing_df1.rename(columns={"close": "AAPL_close"})
```

```
closing_df2 = pd.DataFrame(GOOG['close'])
close2  = closing_df2.rename(columns={"close": "GOOG_close"})

closing_df3 = pd.DataFrame(MSFT['close'])
close3  = closing_df3.rename(columns={"close": "MSFT_close"})

closing_df4 = pd.DataFrame(AMZN['close'])
close4  = closing_df4.rename(columns={"close": "AMZN_close"})

closing_df = pd.concat([close1, close2, close3, close4], axis=1)
closing_df.head()
```

Out[53]:

date	AAPL_close	GOOG_close	MSFT_close	AMZN_close
2022-04-05	175.06	141.0630	310.88	164.055
2022-04-06	171.83	137.1760	299.50	158.756
2022-04-07	172.14	136.4650	301.37	157.785
2022-04-08	170.09	134.0105	296.97	154.461
2022-04-11	165.75	129.7965	285.26	151.122

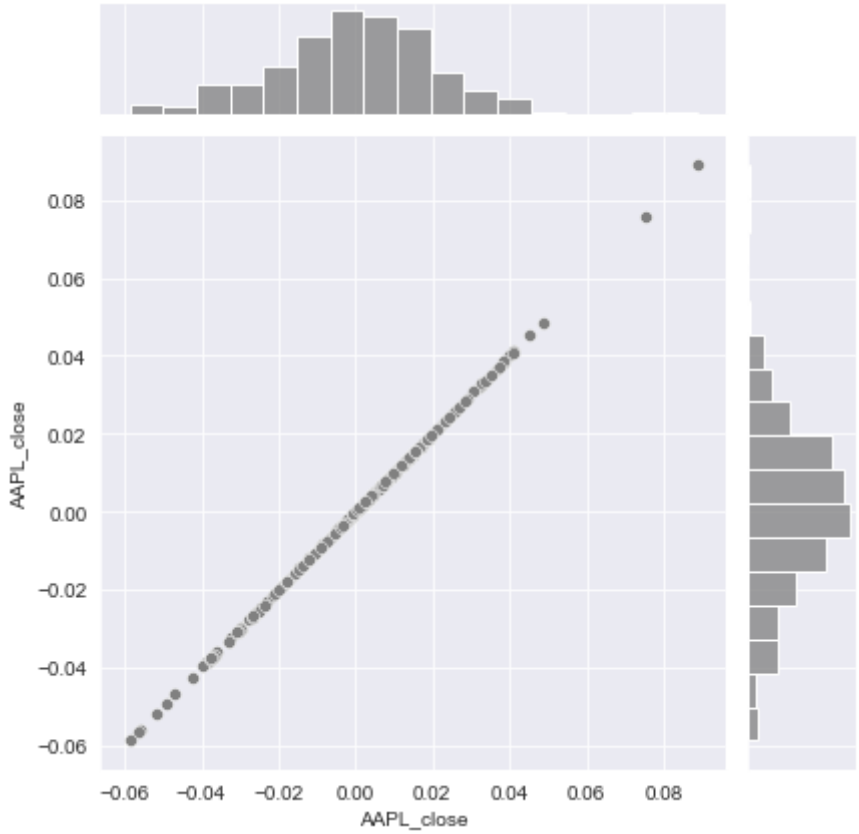
Now that we have all the closing prices, let's go ahead and get the daily return for all the stocks.

In [54]:
```
# Make a new tech returns DataFrame
tech_returns = closing_df.pct_change()
```

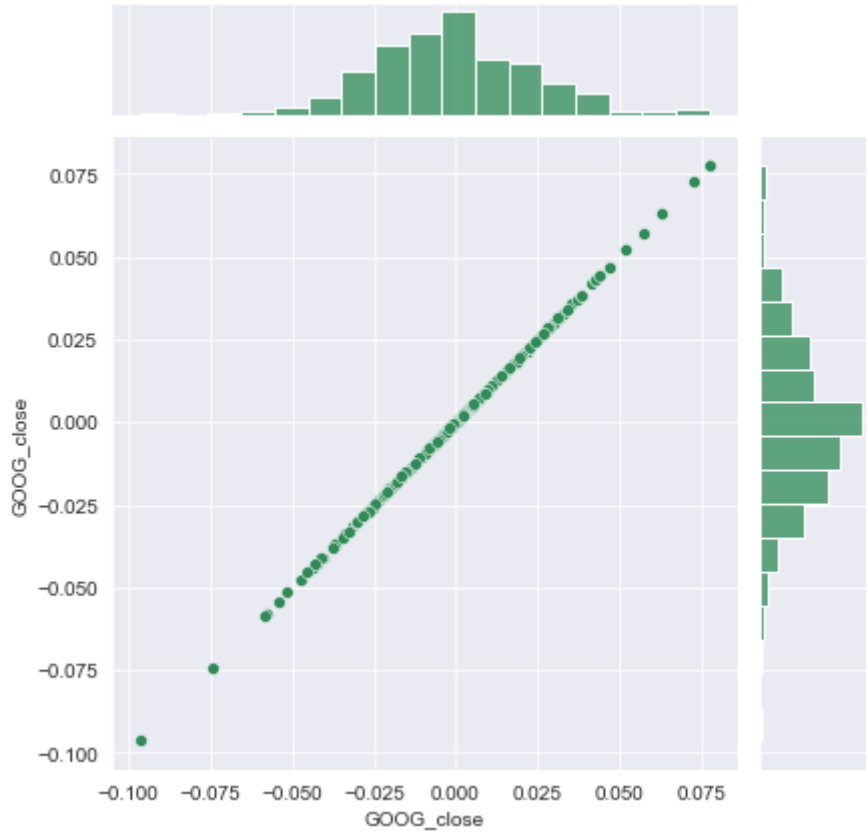
Now we can compare the daily percentage return of two stocks to check how correlated.

First let's see a sotck compared to itself.

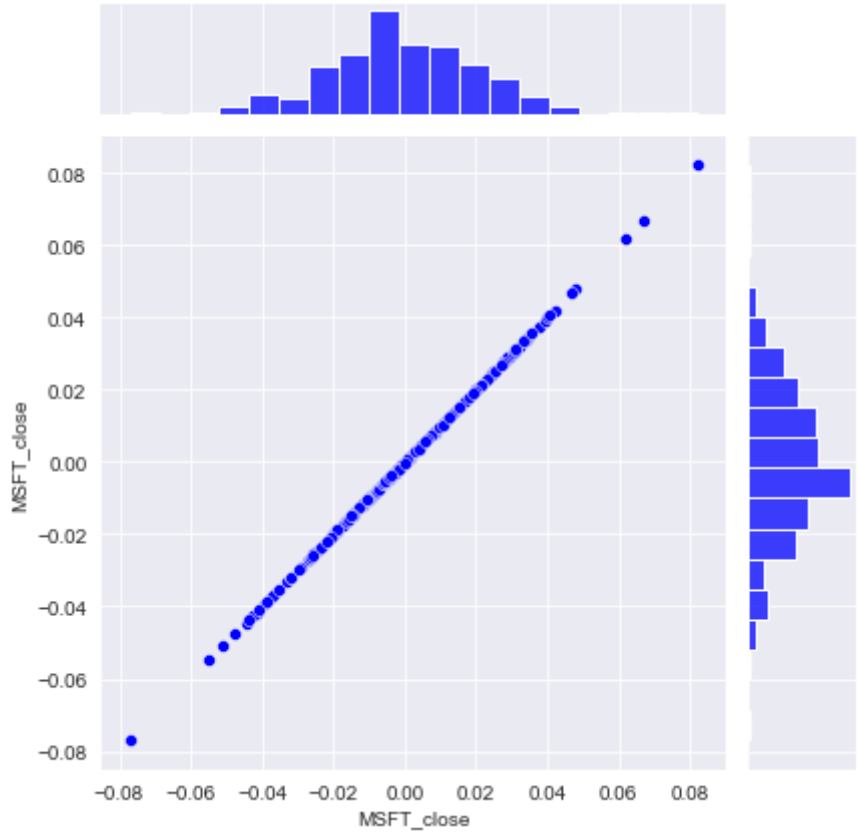
In [55]:
```
# Comparing Apple to itself should show a perfectly linear relationship
sns.jointplot(x='AAPL_close',y='AAPL_close',data=tech_returns,kind="scatter",color=
plt.show()
```



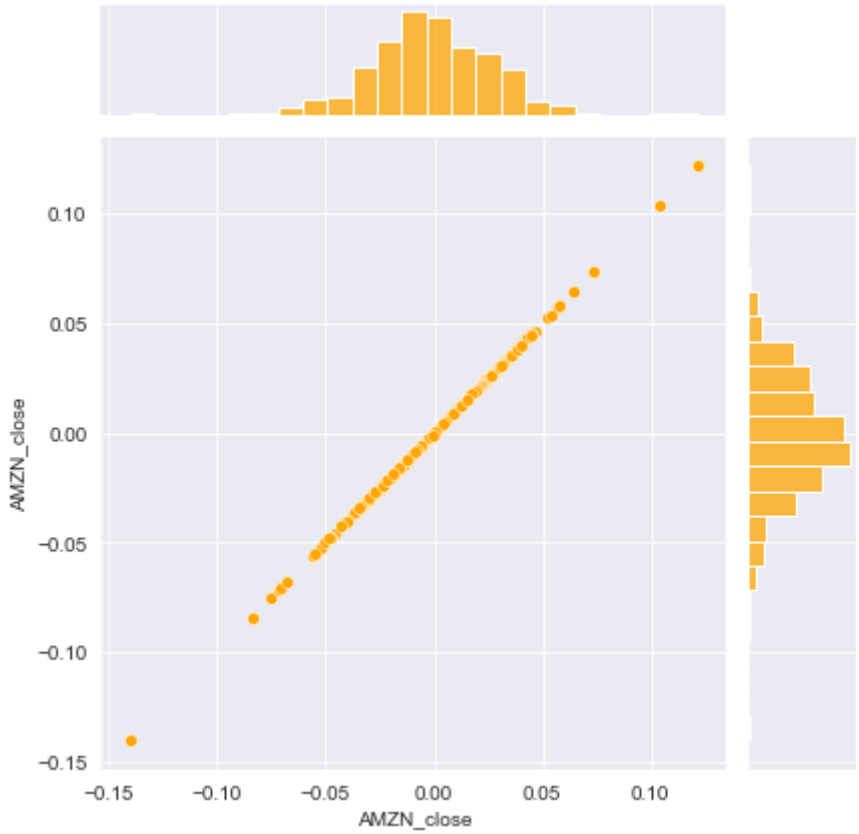
In [56]:
```
# Comparing Google to itself should show a perfectly linear relationship
sns.jointplot(x='GOOG_close',y='GOOG_close',data=tech_returns,kind="scatter",color
plt.show()
```



In [57]:
```
# Comparing Microsoft to itself should show a perfectly linear relationship
sns.jointplot(x='MSFT_close',y='MSFT_close',data=tech_returns,kind="scatter",color
plt.show()
```



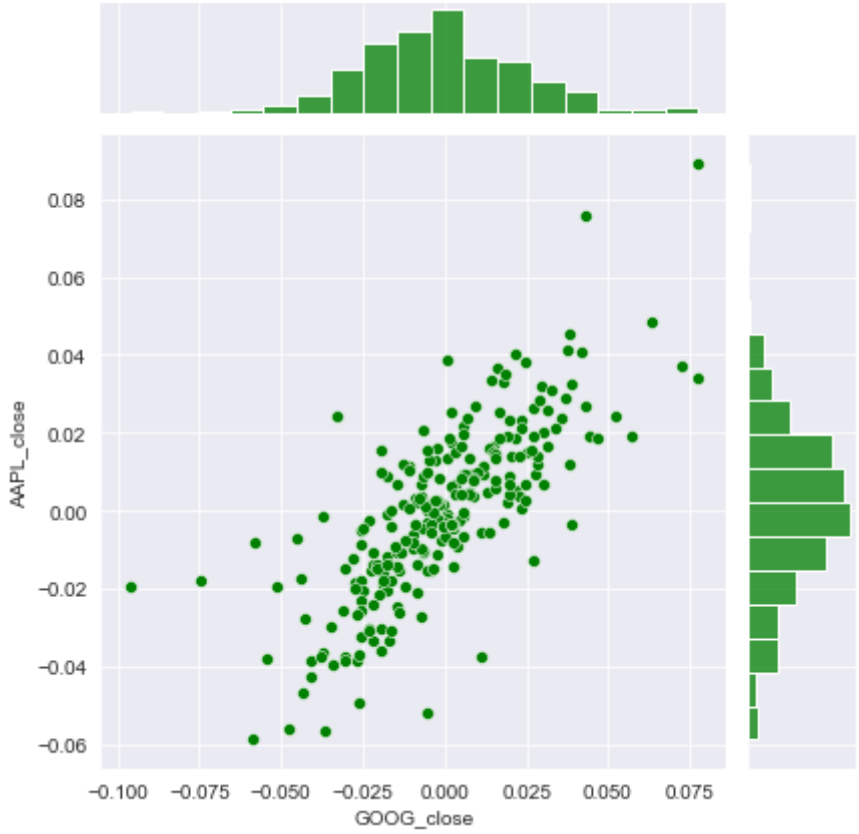
In [58]:
```
# Comparing Amazon to itself should show a perfectly linear relationship
sns.jointplot(x='AMZN_close',y='AMZN_close',data=tech_returns,kind="scatter",color=
plt.show()
```



So now we can see that if two stocks are perfectly (and positivley) correlated with each other a linear relationship bewteen its daily return values should occur. So let's go ahead and compare Google and Microsoft the same way.

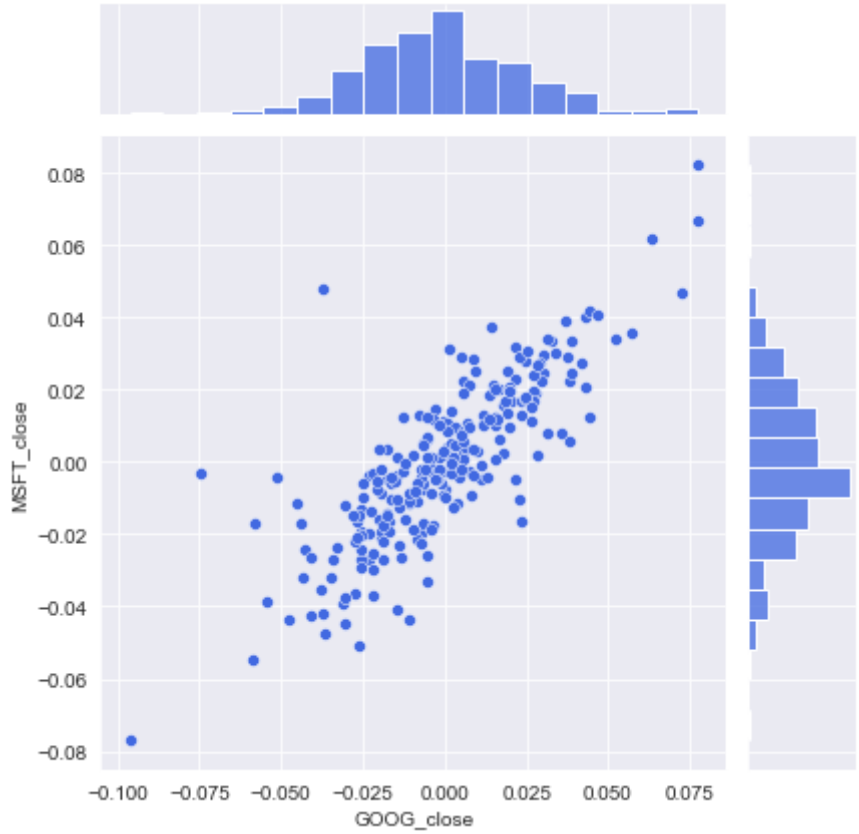
In [59]:
```
# We'll use joinplot to compare the daily returns of Google and Apple

sns.jointplot(x='GOOG_close',y='AAPL_close',data=tech_returns,kind="scatter",color=
plt.show()
```



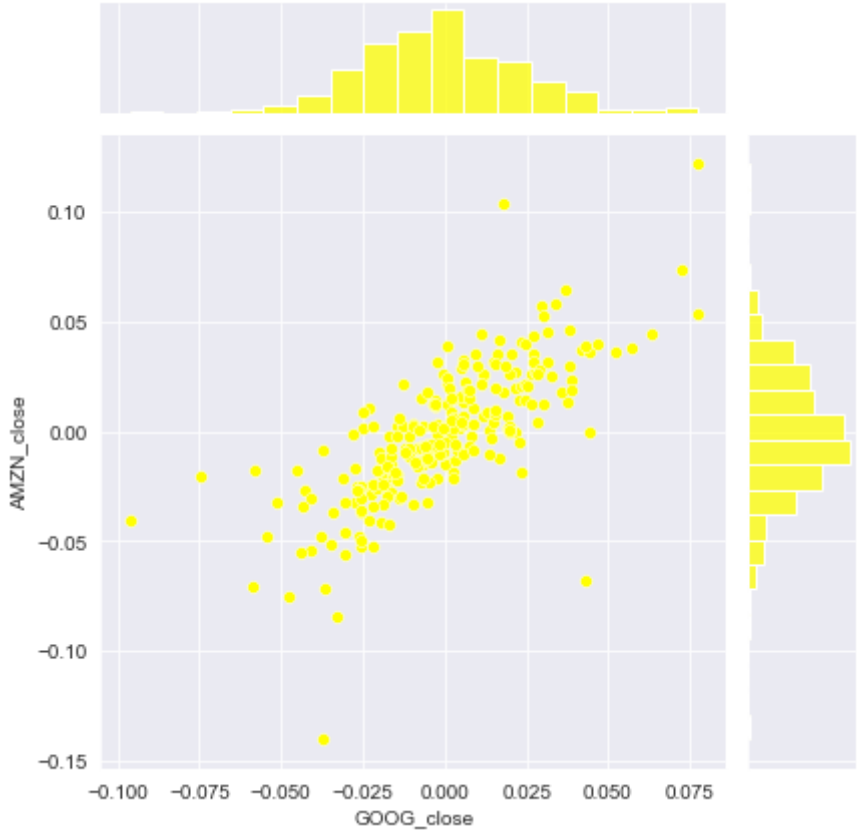
In [60]:
```
# We'll use joinplot to compare the daily returns of Google and Microsoft

sns.jointplot(x='GOOG_close',y='MSFT_close',data=tech_returns,kind="scatter",color=
plt.show()
```



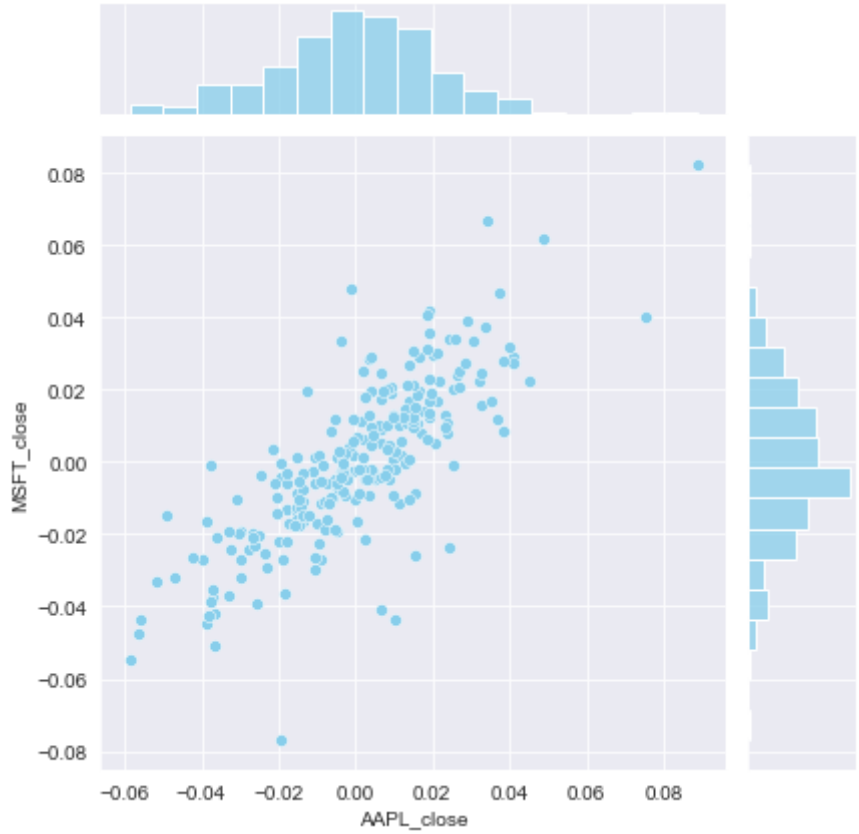
In [61]:
```
# We'll use joinplot to compare the daily returns of Google and Amazon

sns.jointplot(x='GOOG_close',y='AMZN_close',data=tech_returns,kind="scatter",color=
plt.show()
```



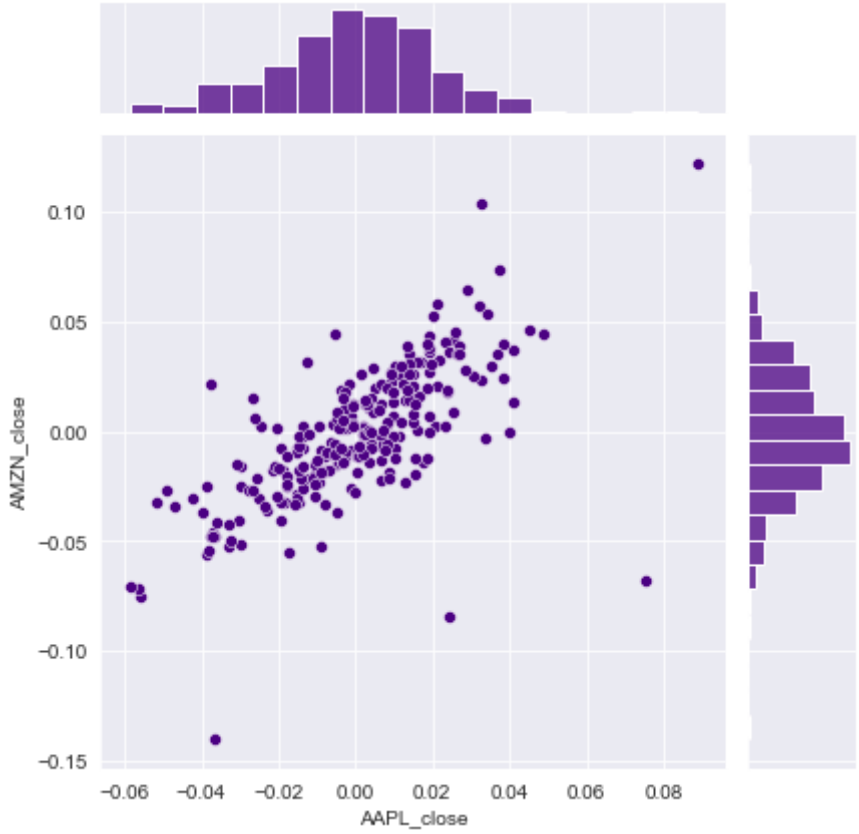
In [62]:
```
# We'll use joinplot to compare the daily returns of Apple and Microsoft

sns.jointplot(x='AAPL_close',y='MSFT_close',data=tech_returns,kind="scatter",color=
plt.show()
```



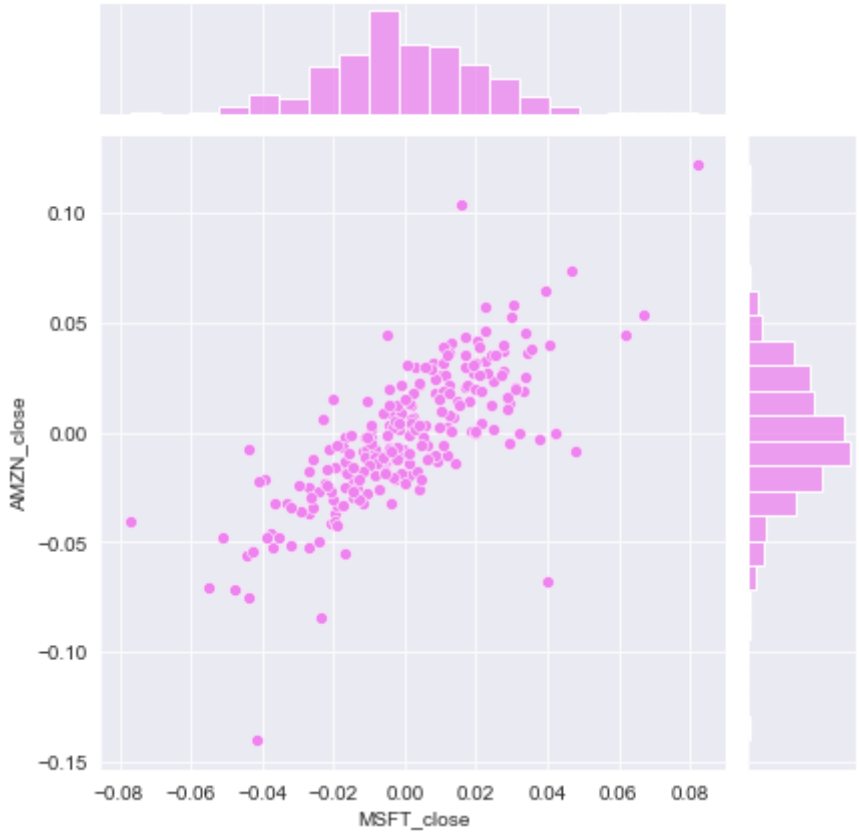
In [63]:
```
# We'll use joinplot to compare the daily returns of Apple and Amazon

sns.jointplot(x='AAPL_close',y='AMZN_close',data=tech_returns,kind="scatter",color
plt.show()
```



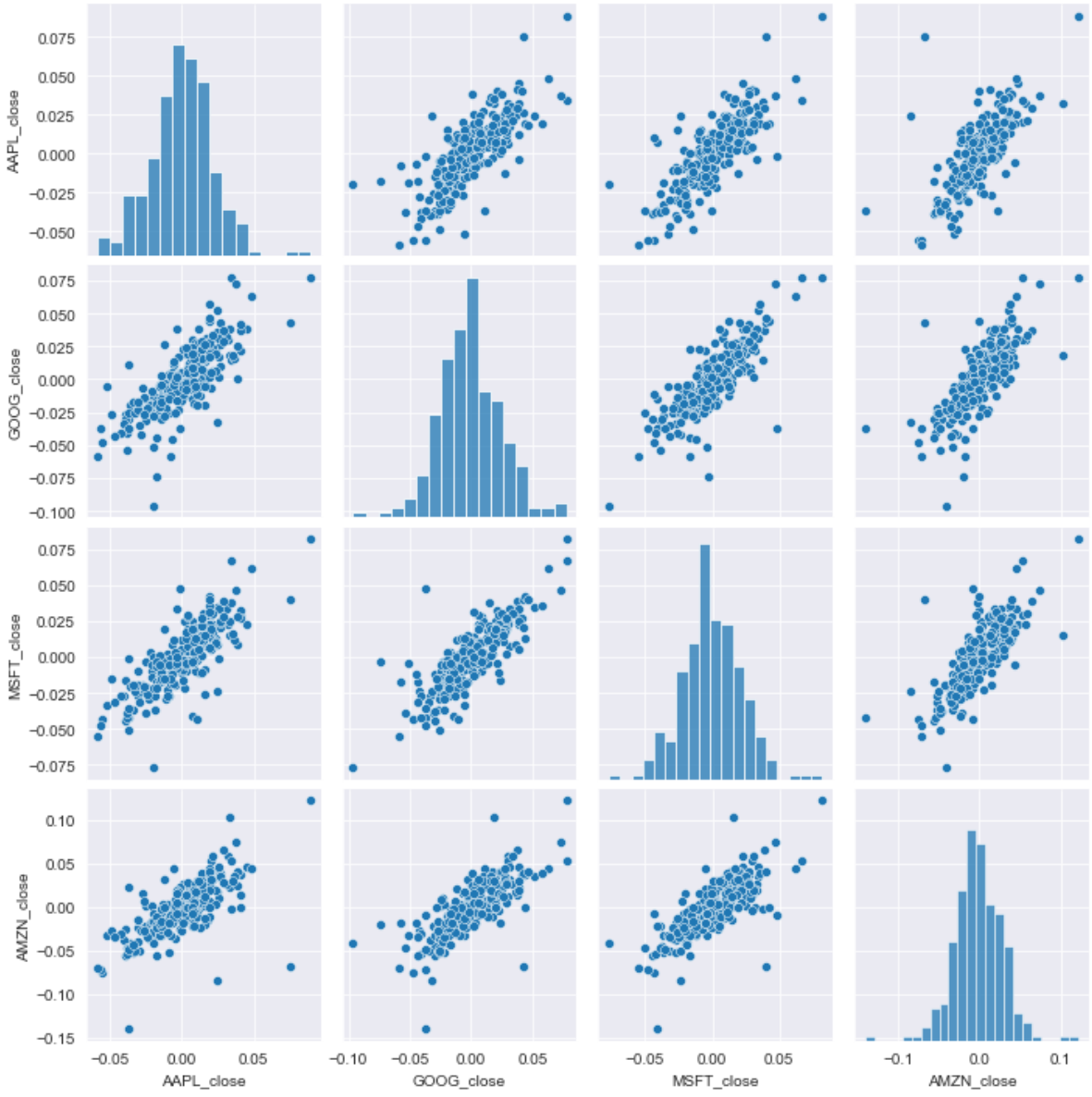
In [64]:
```
# We'll use joinplot to compare the daily returns of Microsoft and Amazon

sns.jointplot(x='MSFT_close',y='AMZN_close',data=tech_returns,kind="scatter",color
plt.show()
```



Python's Seaborn and pandas packages make it very easy to repeat this comparison analysis for every possible combination of stocks in our technology stock ticker list. We can use sns.pairplot() to automatically create this plot.

In [65]:
```
# We can simply call pairplot on our DataFrame for an automatic visual analysis of
sns.pairplot(tech_returns.dropna())
plt.show()
```



Above we can see all the relationships on daily returns between all the stocks.

We could have also analyzed the correlation of the closing prices using this exact same technique. Here it is shown, the code repeated from above with the exception of the DataFrame called.

3.4 Correlation between different Stocks' closing prices

A correlation plot, to get actual numerical values for the correlation between the stocks' closing price values.

In [66]:
```
# use seabron for a quick correlation plot for the closing price of all the stocks

# Compute the correlation matrix
corr = closing_df.dropna().corr()

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr,annot = True, cmap='viridis')
plt.show()
```

- Apple's Stock Closing price is highly correlated with Microsoft's Stock Closing price.
- Google's Stock Closing price is highly correlated with Amazon's Stock Closing price.
- Microsoft's Stock Closing price is highly correlated with both Apple's & Google's Stock Closing prices.
- Amazon's Stock Closing price is highly correlated with Apple's Stock Closing price.

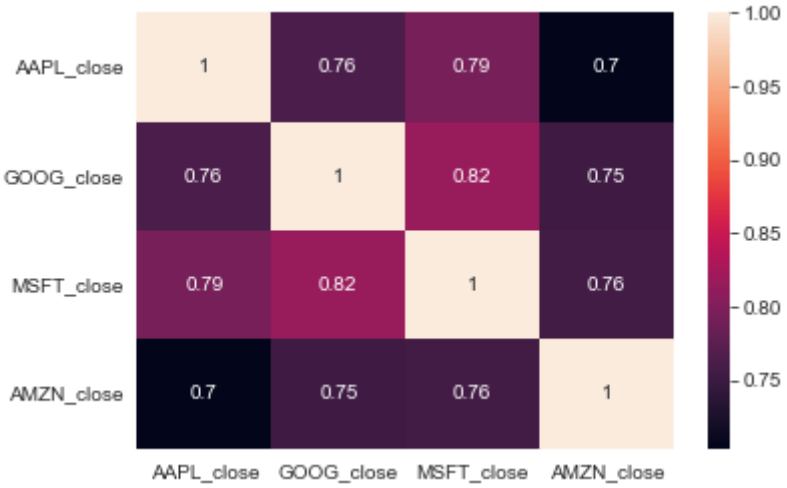
3.5 Correlation between different Stocks' daily returns

Finally, we could also do a correlation plot, to get actual numerical values for the correlation between the stocks' daily return values.

In [67]:
```
# use seabron for a quick correlation plot for the daily returns of all the stocks

# Compute the correlation matrix
corr = tech_returns.dropna().corr()

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr,annot = True, color='#FF0000')
plt.show()
```



- Apple's Stock Daliy Returns is highly correlated with Microsoft's Stock Daliy Returns.
- Google's Stock Daliy Returns is highly correlated with Microsoft's Stock Daliy Returns.
- Microsoft's Stock Daliy Returns is highly correlated with both Google's & Google's Stock Daliy Returns.
- Amazon's Stock Daliy Returns is highly correlated with Microsoft's Stock Daliy Returns.