Projet BIG DATA

POIRON Alex 23492 et THIL Tom 23034

1. Exploration : extract informations

2. Questions on the data

3. Moving average

4. Correlation between stocks

5. Return rate

6. More insights

```
! pip install pyspark
```

```
Requirement already satisfied: pyspark in
/home/alex/.local/lib/python3.8/site-packages (3.2.1)
Requirement already satisfied: py4j==0.10.9.3 in
/home/alex/.local/lib/python3.8/site-packages (from pyspark)
(0.10.9.3)
```

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *

#Manipulation to not confuse max() function from Python and max()
function from Pyspark
to_exclude = ['max', 'sum']
from pyspark.sql.functions import *
for name in to_exclude:
    del globals()[name]

from pyspark.sql.functions import max as max_
from pyspark.sql.functions import sum as sum_

#Correlation imports
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler

#Visualization
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
from matplotlib import rcParams
```

```
import itertools
import requests
```

```
spark_application_name = "Projet"
spark =
(SparkSession.builder.appName(spark_application_name).getOrCreate())
```

```
22/05/21 18:18:06 WARN Utils: Your hostname, alex-ASUS resolves to a
loopback address: 127.0.1.1; using 192.168.1.26 instead (on interface
wlp2s0)
22/05/21 18:18:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind
to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/opt/spark/jars/spark-unsafe_2.12-3.2.1.jar) to constructor
java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of
org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further
illegal reflective access operations
WARNING: All illegal access operations will be denied in a future
release
Using Spark's default log4j profile: org/apache/spark/log4j-
defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/05/21 18:18:07 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where
applicable
```

Exploration : exctract informations

Lire les donnees et changement du schema

Nous allons créer des dataframe avec chaque dataset. Si on décide de le faire à la main, on le peut faire de cette manière :

```
amazon = spark.read.csv("./stocks_data/AMAZON.csv")
amazon.printSchema()
amazon.select('*').limit(10).show()
print("Number of rows =", amazon.count())
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
```

```
 |-- _c7: string (nullable = true)

+----------+-----------------+-----------------+-----------------
+-----------------+-------+-----------------+------------+
|       _c0|              _c1|              _c2|              _c3|
_c4|    _c5|              _c6|         _c7|
+----------+-----------------+-----------------+-----------------
+-----------------+-------+-----------------+------------+
|      Date|             High|              Low|             Open|
Close|  Volume|        Adj Close|company_name|
|2017-01-03| 758.760009765625|747.7000122070312|757.9199829101562|
753.6699829101562|3521100|753.6699829101562|      AMAZON|
|2017-01-04|759.6799926757812|754.2000122070312|758.3900146484375|
757.1799926757812|2510500|757.1799926757812|      AMAZON|
|2017-01-05|782.4000244140625| 760.260009765625|761.5499877929688|
780.4500122070312|5830100|780.4500122070312|      AMAZON|
|2017-01-06|799.4400024414062|  778.47998046875|782.3599853515625|
795.989990234375|5986200| 795.989990234375|      AMAZON|
|2017-01-09|  801.77001953125|  791.77001953125|            798.0|
796.9199829101562|3446100|796.9199829101562|      AMAZON|
|2017-01-10|            798.0|789.5399780273438|796.5999755859375|
795.9000244140625|2558400|795.9000244140625|      AMAZON|
|2017-01-11|            799.5| 789.510009765625|793.6599731445312|
799.02001953125|2992800|  799.02001953125|      AMAZON|
|2017-01-12|814.1300048828125|            799.5|800.3099975585938|
813.6400146484375|4873900|813.6400146484375|      AMAZON|
|2017-01-13|821.6500244140625|811.4000244140625|814.3200073242188|
817.1400146484375|3791900|817.1400146484375|      AMAZON|
+----------+-----------------+-----------------+-----------------
+-----------------+-------+-----------------+------------+

Number of rows = 988
```

Il nous faut changer le schema egalement de ces dataframe pour que nous puissions mieux nous servir des donnees.

```
stocks_columns = [StructField("Date",TimestampType()),
StructField("High",FloatType()), StructField("Low",FloatType()),
StructField("Open",FloatType()),StructField("Close",FloatType()),
StructField("Volume",FloatType()), StructField("Adj
Close",FloatType()), StructField("company_name",StringType())]
new_schema = StructType(stocks_columns)
```

On remarque que l'on peut faire une fonction générique qui peut être appelée à chaque fois. Cela serait beucoup plus rapide si à chaque fois en plus de load le dataset, on souhaite avoir quelques informations dessus. C'est ce que nous allons faire maintenant.

```
def read_infos(path):
    """
        :param path : related path to the file
```

```
        :return the new DataFrame created
    """
    df_with_null = spark.read.csv(path, new_schema) #Creating the df
with the new schema, we have a first null row
    df = spark.createDataFrame(df_with_null.tail(df_with_null.count()-
1), df_with_null.schema) #With this line we delete it
    df.printSchema()
    df.show(40)
    print("Number of rows =", df.count())
    return df

df_amazon = read_infos('./stocks_data/AMAZON.csv')
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)
```

```
+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
+------------+
|2017-01-03 00:00:00|758.76| 747.7|757.92|753.67|3521100.0|   753.67|
AMAZON|
|2017-01-04 00:00:00|759.68| 754.2|758.39|757.18|2510500.0|   757.18|
AMAZON|
|2017-01-05 00:00:00| 782.4|760.26|761.55|780.45|5830100.0|   780.45|
AMAZON|
|2017-01-06 00:00:00|799.44|778.48|782.36|795.99|5986200.0|   795.99|
AMAZON|
|2017-01-09 00:00:00|801.77|791.77| 798.0|796.92|3446100.0|   796.92|
AMAZON|
|2017-01-10 00:00:00| 798.0|789.54| 796.6| 795.9|2558400.0|    795.9|
AMAZON|
|2017-01-11 00:00:00| 799.5|789.51|793.66|799.02|2992800.0|   799.02|
AMAZON|
|2017-01-12 00:00:00|814.13| 799.5|800.31|813.64|4873900.0|   813.64|
AMAZON|
|2017-01-13 00:00:00|821.65| 811.4|814.32|817.14|3791900.0|   817.14|
AMAZON|
|2017-01-17 00:00:00| 816.0|803.44| 815.7|809.72|3670500.0|   809.72|
```

```
AMAZON|
|2017-01-18 00:00:00|811.73|804.27| 809.5|807.48|2354200.0|   807.48|
AMAZON|
|2017-01-19 00:00:00|813.51|807.32| 810.0|809.04|2540800.0|   809.04|
AMAZON|
|2017-01-20 00:00:00|816.02|806.26|815.28|808.33|3376200.0|   808.33|
AMAZON|
|2017-01-23 00:00:00| 818.5|805.08| 806.8|817.88|2797500.0|   817.88|
AMAZON|
|2017-01-24 00:00:00|823.99| 814.5| 822.0|822.44|2971700.0|   822.44|
AMAZON|
|2017-01-25 00:00:00|837.42|825.29|825.79|836.52|3922600.0|   836.52|
AMAZON|
|2017-01-26 00:00:00|843.84| 833.0|835.53|839.15|3586300.0|   839.15|
AMAZON|
|2017-01-27 00:00:00| 839.7|829.44| 839.0|835.77|2998700.0|   835.77|
AMAZON|
|2017-01-30 00:00:00| 833.5|816.38| 833.0|830.38|3747300.0|   830.38|
AMAZON|
|2017-01-31 00:00:00|826.99|819.56|823.75|823.48|3137200.0|   823.48|
AMAZON|
|2017-02-01 00:00:00|833.78|824.94|829.21|832.35|3850200.0|   832.35|
AMAZON|
|2017-02-02 00:00:00|842.49|828.26|836.59|839.95|7350500.0|   839.95|
AMAZON|
|2017-02-03 00:00:00| 818.3| 804.0|806.72| 810.2|1.08688E7|    810.2|
AMAZON|
|2017-02-06 00:00:00|810.72| 803.0| 809.8|807.64|3897300.0|   807.64|
AMAZON|
|2017-02-07 00:00:00|816.16| 807.5|809.31| 812.5|3466100.0|    812.5|
AMAZON|
|2017-02-08 00:00:00|821.48| 812.5|812.69|819.71|2858000.0|   819.71|
AMAZON|
|2017-02-09 00:00:00| 825.0|819.71| 821.6|821.36|2484900.0|   821.36|
AMAZON|
|2017-02-10 00:00:00| 828.0|822.85|823.82|827.46|2429600.0|   827.46|
AMAZON|
|2017-02-13 00:00:00| 843.0|828.55|831.62|836.53|4172600.0|   836.53|
AMAZON|
|2017-02-14 00:00:00|838.31|831.45| 837.0|836.39|2792400.0|   836.39|
AMAZON|
|2017-02-15 00:00:00|842.81|832.82| 834.0| 842.7|2968900.0|    842.7|
AMAZON|
|2017-02-16 00:00:00| 845.0|839.38|841.84|844.14|2714700.0|   844.14|
AMAZON|
|2017-02-17 00:00:00|847.27|840.73| 842.0|845.07|3112300.0|   845.07|
AMAZON|
|2017-02-21 00:00:00|857.98|847.25|848.84|856.44|3507700.0|   856.44|
AMAZON|
|2017-02-22 00:00:00|858.43|852.18|856.95|855.61|2617000.0|   855.61|
```

```
AMAZON|
|2017-02-23 00:00:00|860.86| 848.0|857.57|852.19|3462000.0|   852.19|
AMAZON|
|2017-02-24 00:00:00|845.81|837.75|844.69|845.24|3688000.0|   845.24|
AMAZON|
|2017-02-27 00:00:00| 852.5|839.67|842.38|848.64|2713600.0|   848.64|
AMAZON|
|2017-02-28 00:00:00|854.09|842.05|851.45|845.04|2793700.0|   845.04|
AMAZON|
|2017-03-01 00:00:00|854.83|849.01|853.05|853.08|2760100.0|   853.08|
AMAZON|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
```

```
df_apple = read_infos('./stocks_data/APPLE.csv')
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+-------+-------+-------+----------
+---------+------------+
|               Date|   High|    Low|   Open|  Close|    Volume|Adj
Close|company_name|
+-------------------+-------+-------+-------+-------+----------
+---------+------------+
|2017-01-03 00:00:00|29.0825|  28.69|  28.95|29.0375|1.151276E8|
27.27764|       APPLE|
|2017-01-04 00:00:00|29.1275|28.9375|28.9625| 29.005| 8.44724E7|
27.247108|       APPLE|
|2017-01-05 00:00:00| 29.215|28.9525|  28.98|29.1525| 8.87744E7|
27.385668|       APPLE|
|2017-01-06 00:00:00|  29.54|29.1175| 29.195|29.4775|1.270076E8|
27.690971|       APPLE|
|2017-01-09 00:00:00|29.8575| 29.485|29.4875|29.7475|1.342476E8|
27.944603|       APPLE|
|2017-01-10 00:00:00| 29.845| 29.575|29.6925|29.7775| 9.78484E7|
27.972786|       APPLE|
|2017-01-11 00:00:00|29.9825|  29.65| 29.685|29.9375|1.103544E8|
28.123089|       APPLE|
|2017-01-12 00:00:00| 29.825|29.5525| 29.725|29.8125|1.083448E8|
```

28.005665| APPLE|
|2017-01-13 00:00:00| 29.905|29.7025|29.7775| 29.76|1.044476E8|
27.95635| APPLE|
|2017-01-17 00:00:00| 30.06| 29.555| 29.585| 30.0|1.377592E8|
28.1818| APPLE|
|2017-01-18 00:00:00| 30.125|29.9275| 30.0|29.9975| 9.4852E7|
28.179457| APPLE|
|2017-01-19 00:00:00|30.0225|29.8425| 29.85| 29.945|1.023892E8|
28.130133| APPLE|
|2017-01-20 00:00:00|30.1125|29.9325|30.1125| 30.0|1.303916E8|
28.1818| APPLE|
|2017-01-23 00:00:00|30.2025|29.9425| 30.0| 30.02| 8.82008E7|
28.20059| APPLE|
|2017-01-24 00:00:00| 30.025| 29.875|29.8875|29.9925| 9.2844E7|
28.174757| APPLE|
|2017-01-25 00:00:00| 30.525| 30.07| 30.105| 30.47|1.295104E8|
28.623314| APPLE|
|2017-01-26 00:00:00| 30.61| 30.4|30.4175| 30.485|1.053504E8|
28.637407| APPLE|
|2017-01-27 00:00:00|30.5875| 30.4| 30.535|30.4875| 8.22516E7|
28.639755| APPLE|
|2017-01-30 00:00:00|30.4075| 30.165|30.2325|30.4075| 1.2151E8|
28.564606| APPLE|
|2017-01-31 00:00:00|30.3475| 30.155|30.2875|30.3375| 1.96804E8|
28.498844| APPLE|
|2017-02-01 00:00:00|32.6225|31.7525|31.7575|32.1875| 4.4794E8|
30.23672| APPLE|
|2017-02-02 00:00:00|32.3475| 31.945| 31.995|32.1325|1.348416E8|
30.18506| APPLE|
|2017-02-03 00:00:00|32.2975| 32.04|32.0775| 32.27| 9.80292E7|
30.31423| APPLE|
|2017-02-06 00:00:00| 32.625| 32.225|32.2825|32.5725|1.073836E8|
30.598392| APPLE|
|2017-02-07 00:00:00|33.0225|32.6125| 32.635|32.8825|1.527352E8|
30.8896| APPLE|
|2017-02-08 00:00:00| 33.055| 32.805|32.8375| 33.01| 9.20164E7|
31.009373| APPLE|
|2017-02-09 00:00:00|33.1125| 32.78|32.9125| 33.105|1.133996E8|
31.233446| APPLE|
|2017-02-10 00:00:00| 33.235|33.0125| 33.115| 33.03| 8.0262E7|
31.16269| APPLE|
|2017-02-13 00:00:00| 33.455|33.1875| 33.27|33.3225| 9.21416E7|
31.438652| APPLE|
|2017-02-14 00:00:00|33.7725|33.3125|33.3675| 33.755|1.329048E8|
31.846703| APPLE|
|2017-02-15 00:00:00|34.0675| 33.655| 33.88|33.8775|1.424924E8|
31.962276| APPLE|
|2017-02-16 00:00:00| 33.975| 33.71|33.9175|33.8375| 9.03384E7|
31.924547| APPLE|
|2017-02-17 00:00:00|33.9575| 33.775| 33.775| 33.93| 8.87928E7|

```
32.01181|       APPLE|
|2017-02-21 00:00:00|34.1875| 33.995|34.0575| 34.175| 9.80288E7|
32.242958|       APPLE|
|2017-02-22 00:00:00|  34.28|34.0275|34.1075|34.2775| 8.33476E7|
32.339664|       APPLE|
|2017-02-23 00:00:00|  34.37| 34.075| 34.345|34.1325| 8.31528E7|
32.20287|       APPLE|
|2017-02-24 00:00:00| 34.165|  33.82|33.9775| 34.165| 8.71064E7|
32.233524|       APPLE|
|2017-02-27 00:00:00|  34.36|  34.07| 34.285|34.2325| 8.10296E7|
32.29721|       APPLE|
|2017-02-28 00:00:00|  34.36| 34.175|  34.27|34.2475| 9.39316E7|
32.31136|       APPLE|
|2017-03-01 00:00:00|35.0375|   34.4|34.4725|34.9475|1.456584E8|
32.97178|       APPLE|
+-------------------+-------+-------+-------+-------+----------
+---------+------------+
only showing top 40 rows

Number of rows = 987
```

```
df_fb = read_infos("./stocks_data/FACEBOOK.csv")
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
+------------+
|2017-01-03 00:00:00|117.84|115.51|116.03|116.86|2.06639E7|   116.86|
FACEBOOK|
|2017-01-04 00:00:00|119.66|117.29|117.55|118.69|1.96309E7|   118.69|
FACEBOOK|
|2017-01-05 00:00:00|120.95|118.32|118.86|120.67|1.94922E7|   120.67|
FACEBOOK|
|2017-01-06 00:00:00|123.88|120.03|120.98|123.41|2.85453E7|   123.41|
FACEBOOK|
|2017-01-09 00:00:00|125.43|123.04|123.55| 124.9|2.28804E7|    124.9|
FACEBOOK|
|2017-01-10 00:00:00| 125.5|124.28|124.82|124.35|1.73246E7|   124.35|
```

```
FACEBOOK|
|2017-01-11 00:00:00|126.12|124.06|124.35|126.09|1.83565E7|   126.09|
FACEBOOK|
|2017-01-12 00:00:00|126.73| 124.8|125.61|126.62|1.86539E7|   126.62|
FACEBOOK|
|2017-01-13 00:00:00|129.27|127.37|127.49|128.34|2.48843E7|   128.34|
FACEBOOK|
|2017-01-17 00:00:00|128.34| 127.4|128.04|127.87|1.52945E7|   127.87|
FACEBOOK|
|2017-01-18 00:00:00|128.43|126.84|128.41|127.92|1.31459E7|   127.92|
FACEBOOK|
|2017-01-19 00:00:00|128.35|127.45|128.23|127.55|1.21955E7|   127.55|
FACEBOOK|
|2017-01-20 00:00:00|128.48|126.78| 128.1|127.04|1.90972E7|   127.04|
FACEBOOK|
|2017-01-23 00:00:00|129.25|126.95|127.31|128.93|1.65936E7|   128.93|
FACEBOOK|
|2017-01-24 00:00:00| 129.9|128.38|129.38|129.37|1.51627E7|   129.37|
FACEBOOK|
|2017-01-25 00:00:00|131.74|129.77| 130.0|131.48|1.87313E7|   131.48|
FACEBOOK|
|2017-01-26 00:00:00|133.14|131.44|131.63|132.78|2.00201E7|   132.78|
FACEBOOK|
|2017-01-27 00:00:00|132.95|131.08|132.68|132.18|1.95395E7|   132.18|
FACEBOOK|
|2017-01-30 00:00:00|131.58| 129.6|131.58|130.98|1.89561E7|   130.98|
FACEBOOK|
|2017-01-31 00:00:00|130.66|129.52|130.17|130.32|1.97905E7|   130.32|
FACEBOOK|
|2017-02-01 00:00:00|133.49|130.68|132.25|133.23|5.01398E7|   133.23|
FACEBOOK|
|2017-02-02 00:00:00|135.49| 130.4|133.22|130.84|5.43664E7|   130.84|
FACEBOOK|
|2017-02-03 00:00:00|132.85|130.76|131.24|130.98|2.48049E7|   130.98|
FACEBOOK|
|2017-02-06 00:00:00|132.06| 130.3|130.98|132.06|1.70585E7|   132.06|
FACEBOOK|
|2017-02-07 00:00:00| 133.0|131.66|132.24|131.84|1.45964E7|   131.84|
FACEBOOK|
|2017-02-08 00:00:00|134.44|132.44| 132.6| 134.2|2.23906E7|    134.2|
FACEBOOK|
|2017-02-09 00:00:00| 134.5|133.31|134.49|134.14|1.64706E7|   134.14|
FACEBOOK|
|2017-02-10 00:00:00|134.94|133.68| 134.1|134.19|1.50619E7|   134.19|
FACEBOOK|
|2017-02-13 00:00:00| 134.7| 133.7| 134.7|134.05|1.35262E7|   134.05|
FACEBOOK|
|2017-02-14 00:00:00|134.23|132.55| 134.1|133.85|1.43649E7|   133.85|
FACEBOOK|
|2017-02-15 00:00:00| 133.7|132.66|133.45|133.44|1.32265E7|   133.44|
```

```
FACEBOOK|
|2017-02-16 00:00:00|133.87|133.02|133.07|133.84|1.28311E7|   133.84|
FACEBOOK|
|2017-02-17 00:00:00|134.09|133.17| 133.5|133.53|1.22765E7|   133.53|
FACEBOOK|
|2017-02-21 00:00:00|133.91| 132.9| 133.5|133.72|1.47591E7|   133.72|
FACEBOOK|
|2017-02-22 00:00:00|136.79|133.46| 133.6|136.12|2.73601E7|   136.12|
FACEBOOK|
|2017-02-23 00:00:00|136.12|134.33|135.89|135.36|1.84225E7|   135.36|
FACEBOOK|
|2017-02-24 00:00:00|135.62|134.16|134.16|135.44|1.26257E7|   135.44|
FACEBOOK|
|2017-02-27 00:00:00|137.18|135.02|135.26|136.41|1.43067E7|   136.41|
FACEBOOK|
|2017-02-28 00:00:00|136.81|134.75|136.79|135.54|1.61121E7|   135.54|
FACEBOOK|
|2017-03-01 00:00:00|137.48| 136.3|136.47|137.42| 1.6257E7|   137.42|
FACEBOOK|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
```

```
df_google = read_infos("./stocks_data/GOOGLE.csv")
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+-------+------+-------+---------
+---------+------------+
|               Date|   High|    Low|  Open|  Close|   Volume|Adj
Close|company_name|
+-------------------+-------+-------+------+-------+---------
+---------+------------+
|2017-01-03 00:00:00| 789.63|  775.8|778.81| 786.14|1657300.0|
786.14|      GOOGLE|
|2017-01-04 00:00:00| 791.34| 783.16|788.36|  786.9|1073000.0|
786.9|      GOOGLE|
|2017-01-05 00:00:00| 794.48| 785.02|786.08| 794.02|1335200.0|
794.02|      GOOGLE|
|2017-01-06 00:00:00|  807.9|792.204|795.26| 806.15|1640200.0|
```

806.15| GOOGLE|
|2017-01-09 00:00:00|809.966| 802.83| 806.4| 806.65|1274600.0|
806.65| GOOGLE|
|2017-01-10 00:00:00| 809.13| 803.51|807.86| 804.79|1176800.0|
804.79| GOOGLE|
|2017-01-11 00:00:00| 808.15| 801.37| 805.0| 807.91|1065900.0|
807.91| GOOGLE|
|2017-01-12 00:00:00| 807.39| 799.17|807.14| 806.36|1353100.0|
806.36| GOOGLE|
|2017-01-13 00:00:00|811.224| 806.69|807.48| 807.88|1099200.0|
807.88| GOOGLE|
|2017-01-17 00:00:00| 807.14| 800.37|807.08| 804.61|1362100.0|
804.61| GOOGLE|
|2017-01-18 00:00:00|806.205| 800.99|805.81| 806.07|1294400.0|
806.07| GOOGLE|
|2017-01-19 00:00:00| 809.48| 801.8|805.12|802.175| 919300.0|
802.175| GOOGLE|
|2017-01-20 00:00:00| 806.91| 801.69|806.91| 805.02|1670000.0|
805.02| GOOGLE|
|2017-01-23 00:00:00| 820.87| 803.74|807.25| 819.31|1963600.0|
819.31| GOOGLE|
|2017-01-24 00:00:00| 825.9|817.821| 822.3| 823.87|1474000.0|
823.87| GOOGLE|
|2017-01-25 00:00:00| 835.77| 825.06|829.62| 835.67|1494500.0|
835.67| GOOGLE|
|2017-01-26 00:00:00| 838.0| 827.01|837.81| 832.15|2973900.0|
832.15| GOOGLE|
|2017-01-27 00:00:00| 841.95| 820.44|834.71| 823.31|2965800.0|
823.31| GOOGLE|
|2017-01-30 00:00:00| 815.84| 799.8|814.66| 802.32|3246600.0|
802.32| GOOGLE|
|2017-01-31 00:00:00| 801.25| 790.52|796.86| 796.79|2160600.0|
796.79| GOOGLE|
|2017-02-01 00:00:00| 801.19| 791.19|799.68|795.695|2029700.0|
795.695| GOOGLE|
|2017-02-02 00:00:00| 802.7| 792.0| 793.8| 798.53|1532100.0|
798.53| GOOGLE|
|2017-02-03 00:00:00| 806.0| 800.37|802.99| 801.49|1463400.0|
801.49| GOOGLE|
|2017-02-06 00:00:00| 801.67| 795.25| 799.7| 801.34|1184500.0|
801.34| GOOGLE|
|2017-02-07 00:00:00| 810.5| 801.78|803.99| 806.97|1241200.0|
806.97| GOOGLE|
|2017-02-08 00:00:00| 811.84| 803.19| 807.0| 808.38|1155300.0|
808.38| GOOGLE|
|2017-02-09 00:00:00| 810.66| 804.54|809.51| 809.56| 989700.0|
809.56| GOOGLE|
|2017-02-10 00:00:00| 815.25| 809.78| 811.7| 813.67|1135000.0|
813.67| GOOGLE|
|2017-02-13 00:00:00|820.959| 815.49| 816.0| 819.24|1213300.0|

```
819.24|      GOOGLE|
|2017-02-14 00:00:00|  823.0|  816.0| 819.0| 820.45|1054700.0|
820.45|      GOOGLE|
|2017-02-15 00:00:00|  823.0| 818.47|819.36| 818.98|1313600.0|
818.98|      GOOGLE|
|2017-02-16 00:00:00|  824.4| 818.98|819.93| 824.16|1287600.0|
824.16|      GOOGLE|
|2017-02-17 00:00:00| 828.07|821.655|823.02| 828.07|1611000.0|
828.07|      GOOGLE|
|2017-02-21 00:00:00| 833.45| 828.35|828.66| 831.66|1262300.0|
831.66|      GOOGLE|
|2017-02-22 00:00:00| 833.25| 828.64|828.66| 830.76| 982900.0|
830.76|      GOOGLE|
|2017-02-23 00:00:00| 832.46| 822.88|830.12| 831.33|1472800.0|
831.33|      GOOGLE|
|2017-02-24 00:00:00|  829.0|  824.2|827.73| 828.64|1392200.0|
828.64|      GOOGLE|
|2017-02-27 00:00:00|  830.5|  824.0|824.55| 829.28|1101500.0|
829.28|      GOOGLE|
|2017-02-28 00:00:00| 828.54|  820.2|825.61| 823.21|2260800.0|
823.21|      GOOGLE|
|2017-03-01 00:00:00|836.255| 827.26|828.85| 835.24|1496500.0|
835.24|      GOOGLE|
+-------------------+-------+-------+------+-------+---------
+---------+------------+
only showing top 40 rows

Number of rows = 987

df_microsoft = read_infos("./stocks_data/MICROSOFT.csv")

root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-----+-----+-----+-----+---------+---------
+------------+
|               Date| High|  Low| Open|Close|   Volume|Adj Close|
company_name|
+-------------------+-----+-----+-----+-----+---------+---------
+------------+
|2017-01-03 00:00:00|62.84|62.13|62.79|62.58|2.06941E7|58.673244|
MICROSOFT|
|2017-01-04 00:00:00|62.75|62.12|62.48| 62.3|  2.134E7|58.410725|
```

MICROSOFT|
|2017-01-05 00:00:00|62.66|62.03|62.19| 62.3| 2.4876E7|58.410725|
MICROSOFT|
|2017-01-06 00:00:00|63.15|62.04| 62.3|62.84|1.99229E7|58.917015|
MICROSOFT|
|2017-01-09 00:00:00|63.08|62.54|62.76|62.64|2.03827E7|58.729496|
MICROSOFT|
|2017-01-10 00:00:00|63.07|62.28|62.73|62.62| 1.8593E7|58.710747|
MICROSOFT|
|2017-01-11 00:00:00|63.23|62.43|62.61|63.19|2.15173E7| 59.24516|
MICROSOFT|
|2017-01-12 00:00:00| 63.4|61.95|63.06|62.61|2.09682E7| 58.70137|
MICROSOFT|
|2017-01-13 00:00:00|62.87|62.35|62.62| 62.7|1.94223E7|58.785755|
MICROSOFT|
|2017-01-17 00:00:00| 62.7|62.03|62.68|62.53| 2.0664E7| 58.62637|
MICROSOFT|
|2017-01-18 00:00:00| 62.7|62.12|62.67| 62.5|1.96701E7|58.598248|
MICROSOFT|
|2017-01-19 00:00:00|62.98| 62.2|62.24| 62.3|1.84517E7|58.410725|
MICROSOFT|
|2017-01-20 00:00:00|62.82|62.37|62.67|62.74|3.02135E7| 58.82326|
MICROSOFT|
|2017-01-23 00:00:00|63.12|62.57| 62.7|62.96|2.30976E7|59.029526|
MICROSOFT|
|2017-01-24 00:00:00|63.74|62.94| 63.2|63.52|2.46729E7| 59.55457|
MICROSOFT|
|2017-01-25 00:00:00| 64.1|63.45|63.95|63.68|2.36727E7|59.704575|
MICROSOFT|
|2017-01-26 00:00:00|64.54|63.55|64.12|64.27|4.35546E7|60.257736|
MICROSOFT|
|2017-01-27 00:00:00|65.91|64.89|65.39|65.78| 4.4818E7| 61.67348|
MICROSOFT|
|2017-01-30 00:00:00|65.79| 64.8|65.69|65.13|3.16514E7|61.064045|
MICROSOFT|
|2017-01-31 00:00:00|65.15|64.26|64.86|64.65|2.52705E7| 60.61402|
MICROSOFT|
|2017-02-01 00:00:00|64.62|63.47|64.36|63.58|3.96715E7|59.610825|
MICROSOFT|
|2017-02-02 00:00:00|63.41|62.75|63.25|63.17| 4.5827E7|59.226406|
MICROSOFT|
|2017-02-03 00:00:00| 63.7|63.07| 63.5|63.68|3.03018E7|59.704575|
MICROSOFT|
|2017-02-06 00:00:00|63.65|63.14| 63.5|63.64|1.97964E7| 59.66707|
MICROSOFT|
|2017-02-07 00:00:00|63.78|63.23|63.74|63.43|2.02772E7| 59.47018|
MICROSOFT|
|2017-02-08 00:00:00|63.81|63.22|63.57|63.34|1.80964E7| 59.3858|
MICROSOFT|
|2017-02-09 00:00:00|64.44|63.32|63.52|64.06|2.26444E7| 60.06085|

```
MICROSOFT|
|2017-02-10 00:00:00| 64.3|63.98|64.25| 64.0|1.81707E7|60.004604|
MICROSOFT|
|2017-02-13 00:00:00|64.86|64.13|64.24|64.72|2.29201E7|60.679653|
MICROSOFT|
|2017-02-14 00:00:00|64.72|64.02|64.41|64.57|2.31084E7|60.906044|
MICROSOFT|
|2017-02-15 00:00:00|64.57|64.16| 64.5|64.53|1.70052E7|60.868298|
MICROSOFT|
|2017-02-16 00:00:00|65.24|64.44|64.74|64.52|2.05463E7| 60.85886|
MICROSOFT|
|2017-02-17 00:00:00|64.69| 64.3|64.47|64.62|2.12488E7|60.953197|
MICROSOFT|
|2017-02-21 00:00:00|64.95|64.45|64.61|64.49|2.06559E7| 60.83057|
MICROSOFT|
|2017-02-22 00:00:00|64.39|64.05|64.33|64.36|1.92927E7|60.707947|
MICROSOFT|
|2017-02-23 00:00:00|64.73|64.19|64.42|64.62|2.02731E7|60.953197|
MICROSOFT|
|2017-02-24 00:00:00| 64.8|64.14|64.53|64.62|2.17968E7|60.953197|
MICROSOFT|
|2017-02-27 00:00:00|64.54|64.05|64.54|64.23|1.58715E7| 60.58533|
MICROSOFT|
|2017-02-28 00:00:00| 64.2|63.76|64.08|63.98|2.32398E7|60.349503|
MICROSOFT|
|2017-03-01 00:00:00|64.99|64.02|64.13|64.94|2.69375E7| 61.25505|
MICROSOFT|
+-------------------+-----+-----+-----+-----+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
```

```
df_tesla = read_infos('./stocks_data/TESLA.csv')
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
```

```
+------------+
|2017-01-03 00:00:00|44.066|42.192|42.972|43.398|2.96165E7|   43.398|
TESLA|
|2017-01-04 00:00:00|  45.6|42.862| 42.95|45.398|5.60675E7|   45.398|
TESLA|
|2017-01-05 00:00:00|45.496| 44.39|45.284| 45.35|2.95585E7|    45.35|
TESLA|
|2017-01-06 00:00:00|46.062| 45.09|45.386|45.802|2.76395E7|   45.802|
TESLA|
|2017-01-09 00:00:00|46.384|  45.6|45.794|46.256|1.98975E7|   46.256|
TESLA|
|2017-01-10 00:00:00|  46.4|45.378|  46.4|45.974|   1.83E7|   45.974|
TESLA|
|2017-01-11 00:00:00|45.996|45.336|45.814|45.946| 1.8254E7|   45.946|
TESLA|
|2017-01-12 00:00:00| 46.14|45.116|45.812|45.918| 1.8951E7|   45.918|
TESLA|
|2017-01-13 00:00:00| 47.57|45.918|  46.0| 47.55| 3.0465E7|    47.55|
TESLA|
|2017-01-17 00:00:00|47.992|46.874| 47.34|47.116|2.30875E7|   47.116|
TESLA|
|2017-01-18 00:00:00|47.942|47.116| 47.33|47.672| 1.8845E7|   47.672|
TESLA|
|2017-01-19 00:00:00|49.736| 48.15| 49.45|48.752|3.86615E7|   48.752|
TESLA|
|2017-01-20 00:00:00|  49.2|48.602|49.092|48.946|2.10215E7|   48.946|
TESLA|
|2017-01-23 00:00:00|50.178|  49.1| 49.17|49.784|3.13145E7|   49.784|
TESLA|
|2017-01-24 00:00:00| 50.96| 49.93|  50.0|50.922|2.48275E7|   50.922|
TESLA|
|2017-01-25 00:00:00|51.692| 50.36|51.462|50.894| 2.5713E7|   50.894|
TESLA|
|2017-01-26 00:00:00|51.148| 50.15|50.858|50.502|1.57605E7|   50.502|
TESLA|
|2017-01-27 00:00:00|  50.6|49.704|50.276| 50.59|1.58315E7|    50.59|
TESLA|
|2017-01-30 00:00:00|51.058| 49.42|50.506|50.126|1.90055E7|   50.126|
TESLA|
|2017-01-31 00:00:00|51.178| 49.54|49.848|50.386|2.05805E7|   50.386|
TESLA|
|2017-02-01 00:00:00| 50.64| 49.81| 50.61|49.848| 1.9794E7|   49.848|
TESLA|
|2017-02-02 00:00:00|50.484|49.542|49.668| 50.31| 1.2499E7|    50.31|
TESLA|
|2017-02-03 00:00:00|50.436|49.936|50.382|50.266|1.09335E7|   50.266|
TESLA|
|2017-02-06 00:00:00|51.564|50.126|  50.2|51.554|1.78125E7|   51.554|
TESLA|
|2017-02-07 00:00:00|  52.0|51.284|51.638|51.496| 2.1224E7|   51.496|
```

```
TESLA|
|2017-02-08 00:00:00|52.672| 51.24| 51.47|52.416| 1.9665E7|   52.416|
TESLA|
|2017-02-09 00:00:00|54.236| 53.23| 53.25| 53.84| 3.9101E7|    53.84|
TESLA|
|2017-02-10 00:00:00| 54.19|53.222|53.958|53.846|1.80985E7|   53.846|
TESLA|
|2017-02-13 00:00:00|56.158|54.102|54.148| 56.12| 3.5148E7|    56.12|
TESLA|
|2017-02-14 00:00:00|57.478|55.722|55.806|56.196| 3.6726E7|   56.196|
TESLA|
|2017-02-15 00:00:00|56.448|55.288|  56.0|55.952|2.47395E7|   55.952|
TESLA|
|2017-02-16 00:00:00|  56.0|  53.7| 55.52| 53.79|3.53865E7|    53.79|
TESLA|
|2017-02-17 00:00:00|54.578| 52.83| 53.16|54.446|3.12855E7|   54.446|
TESLA|
|2017-02-21 00:00:00| 56.28|54.802| 55.09|55.478|2.83835E7|   55.478|
TESLA|
|2017-02-22 00:00:00| 56.69| 54.52|56.062|54.702| 4.3775E7|   54.702|
TESLA|
|2017-02-23 00:00:00|52.932|51.112|  52.8|51.198| 7.4576E7|   51.198|
TESLA|
|2017-02-24 00:00:00| 51.65| 50.04|50.532|  51.4| 4.0858E7|     51.4|
TESLA|
|2017-02-27 00:00:00|49.672|48.402|49.634|49.246| 5.7304E7|   49.246|
TESLA|
|2017-02-28 00:00:00|  50.2| 48.78|48.838|49.998|3.03905E7|   49.998|
TESLA|
|2017-03-01 00:00:00| 50.97|49.822|50.836|50.004|2.40475E7|   50.004|
TESLA|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
```

```
df_zoom = read_infos('./stocks_data/ZOOM.csv')
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+------+------+------+---------+---------
```

```
+------------+
|               Date|   High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+-------+------+------+------+---------+---------
+------------+
|2019-04-18 00:00:00|   66.0|60.321|  65.0|  62.0|2.57647E7|     62.0|
ZOOM|
|2019-04-22 00:00:00|   68.9| 59.94|  61.0|  65.7|9949700.0|     65.7|
ZOOM|
|2019-04-23 00:00:00| 74.169| 65.55| 66.87|  69.0|6786500.0|     69.0|
ZOOM|
|2019-04-24 00:00:00|   71.5| 63.16|  71.4|  63.2|4973500.0|     63.2|
ZOOM|
|2019-04-25 00:00:00|  66.85|  62.6| 64.74|  65.0|3863300.0|     65.0|
ZOOM|
|2019-04-26 00:00:00|  66.99|  63.6| 66.12| 66.22|1527400.0|    66.22|
ZOOM|
|2019-04-29 00:00:00|   68.5| 64.75| 66.53| 68.17|1822300.0|    68.17|
ZOOM|
|2019-04-30 00:00:00|  72.52| 66.67|  68.4| 72.47|4113100.0|    72.47|
ZOOM|
|2019-05-01 00:00:00|  76.95|70.816| 72.72| 72.76|3301900.0|    72.76|
ZOOM|
|2019-05-02 00:00:00|  75.89|69.691| 72.75|  75.5|2525300.0|     75.5|
ZOOM|
|2019-05-03 00:00:00|  80.25|  75.0|  75.0| 79.18|2590300.0|    79.18|
ZOOM|
|2019-05-06 00:00:00|  80.79|  74.5| 75.01| 78.24|2051800.0|    78.24|
ZOOM|
|2019-05-07 00:00:00|  78.05| 73.25| 77.85| 73.33|1975200.0|    73.33|
ZOOM|
|2019-05-08 00:00:00|   78.5| 74.03| 74.61| 77.68|2265500.0|    77.68|
ZOOM|
|2019-05-09 00:00:00|  76.99|  74.0| 76.85| 75.21|1348200.0|    75.21|
ZOOM|
|2019-05-10 00:00:00|  79.74| 74.77| 75.79| 79.63|1555100.0|    79.63|
ZOOM|
|2019-05-13 00:00:00|  77.39|  70.6| 77.39| 72.54|2873200.0|    72.54|
ZOOM|
|2019-05-14 00:00:00| 76.885| 73.11| 74.12| 73.14|1950400.0|    73.14|
ZOOM|
|2019-05-15 00:00:00|   80.0| 72.21|  73.4| 79.76|2426500.0|    79.76|
ZOOM|
|2019-05-16 00:00:00|  87.55| 79.25| 80.12|  83.4|4580700.0|     83.4|
ZOOM|
|2019-05-17 00:00:00|  90.28| 81.88| 82.25| 89.98|3442500.0|    89.98|
ZOOM|
|2019-05-20 00:00:00|  91.46| 83.27|  90.1| 84.67|3666800.0|    84.67|
ZOOM|
|2019-05-21 00:00:00|   89.7|  84.5| 86.63| 85.44|2576000.0|    85.44|
```

```
ZOOM|
|2019-05-22 00:00:00|   85.7|  82.0| 84.63| 82.43|1596400.0|    82.43|
ZOOM|
|2019-05-23 00:00:00|   81.5| 77.26| 81.25| 78.76|2856000.0|    78.76|
ZOOM|
|2019-05-24 00:00:00|  81.25|  74.2| 80.48| 76.25|2946800.0|    76.25|
ZOOM|
|2019-05-28 00:00:00|  79.39|  76.8|  76.8| 77.77|1641300.0|    77.77|
ZOOM|
|2019-05-29 00:00:00|  77.93|73.583|  77.0| 75.77|1512100.0|    75.77|
ZOOM|
|2019-05-30 00:00:00|  80.97|  76.6| 76.68| 80.42|1996000.0|    80.42|
ZOOM|
|2019-05-31 00:00:00|  83.17| 77.78| 78.77| 79.73|1594300.0|    79.73|
ZOOM|
|2019-06-03 00:00:00|  81.94| 75.65|  80.0|  75.9|1570500.0|     75.9|
ZOOM|
|2019-06-04 00:00:00|  78.88| 76.62|  78.2| 78.74|1134900.0|    78.74|
ZOOM|
|2019-06-05 00:00:00|   80.6| 76.24| 80.14| 78.04|1295800.0|    78.04|
ZOOM|
|2019-06-06 00:00:00|  79.75| 77.03|  77.4| 79.43|3024000.0|    79.43|
ZOOM|
|2019-06-07 00:00:00|  98.89|  92.5| 93.66| 94.05|9487800.0|    94.05|
ZOOM|
|2019-06-10 00:00:00|105.985|  96.0| 98.51| 102.0|4852800.0|    102.0|
ZOOM|
|2019-06-11 00:00:00|  101.2| 91.57| 101.0| 94.87|4372400.0|    94.87|
ZOOM|
|2019-06-12 00:00:00|104.185|  94.0|  94.6|102.77|3151700.0|   102.77|
ZOOM|
|2019-06-13 00:00:00|105.172| 98.55| 105.1|100.95|3189100.0|   100.95|
ZOOM|
|2019-06-14 00:00:00| 104.57| 99.25|100.47|100.29|1889300.0|   100.29|
ZOOM|
+-------------------+-------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 411
```

On créé une variable globale qui va contenir tous nos DatFrame dans une liste

```
LIST_DF = [df_amazon, df_apple, df_fb, df_google, df_microsoft,
df_tesla, df_zoom]
```

Date period

Nous allons maintenant nous intéresser à la période qui s'écoule entre chaque observation de chaque DataFrame. Pour ce faire, nous allons faire une fonction qui en prenant un

DataFrame en paramètre, retournera le type de période éecartant chaque observation. (jour le jour, quelques jours, semaines, ...)

```
def date_period(df):
    """
        :param df the DataFrame
        :return a string that correspond to the period in the
DataFrame
    """
    diff = []
    name = df.first()["company_name"]
    dates = df.select('Date').rdd.flatMap(lambda x: x).collect()
    #Day case
    for i in range(0, len(dates)-1):
        diff.append(dates[i+1].day - dates[i].day)

    period = max(diff, key=diff.count)
    if period == 1 :
        return "For " + name + ", it's a day period"

    #Month and year case
    diff = []
    for i in range(0, len(dates)-1):
        diff.append(dates[i+1].month - dates[i].month)
    period = max(diff, key=diff.count)

    if period == 1 :
        return "For " + name + ", it's a month period"
    else:
        return "For " + name + ", its a year period"

for df in LIST_DF:
    print(date_period(df))
```

```
For AMAZON, it's a day period
For APPLE, it's a day period
For FACEBOOK, it's a day period
For GOOGLE, it's a day period
For MICROSOFT, it's a day period
For TESLA, it's a day period
For ZOOM, it's a day period
```

Descriptive Statistics

Fonction permettant d'avoir certaines statistiques sur chaque colonne d'un dataframe (min, max, moyenne, variance)

```
def stats(df):
    """
        :param df : DataFrame
        :return Print mean, min, max, sd for each column of the
```

```
DataFrame
    """
    columns = ["High", "Low", "Open", "Close", "Volume", "Adj Close"]

    for colname in columns:
        print("Stats for :", colname)
        df.agg(mean(df[colname]), min(df[colname]), max_(df[colname]),
stddev(df[colname])).show()

stats(df_amazon)

Stats for : High
+------------------+---------+---------+-----------------+
|         avg(High)|min(High)|max(High)|stddev_samp(High)|
+------------------+---------+---------+-----------------+
|1762.0071216958152|   758.76|  3552.25| 667.238531575268|
+------------------+---------+---------+-----------------+

Stats for : Low
+------------------+--------+--------+----------------+
|          avg(Low)|min(Low)|max(Low)|stddev_samp(Low)|
+------------------+--------+--------+----------------+
|1722.1011452099956|   747.7| 3486.69|644.798809338276|
+------------------+--------+--------+----------------+

Stats for : Open
+-----------------+---------+---------+-----------------+
|        avg(Open)|min(Open)|max(Open)|stddev_samp(Open)|
+-----------------+---------+---------+-----------------+
|1743.433881363487|   757.92|   3547.0|657.1153070927141|
+-----------------+---------+---------+-----------------+

Stats for : Close
+------------------+----------+----------+------------------+
|        avg(Close)|min(Close)|max(Close)|stddev_samp(Close)|
+------------------+----------+----------+------------------+
|1742.9566644206718|    753.67|   3531.45| 655.9576061129327|
+------------------+----------+----------+------------------+

Stats for : Volume
+----------------+-----------+-----------+-------------------+
|     avg(Volume)|min(Volume)|max(Volume)|stddev_samp(Volume)|
+----------------+-----------+-----------+-------------------+
|4509728.05775076|   881300.0|   1.6565E7| 2179817.6286312877|
+----------------+-----------+-----------+-------------------+

Stats for : Adj Close
+------------------+--------------+-------------
+----------------------+
|    avg(Adj Close)|min(Adj Close)|max(Adj Close)|stddev_samp(Adj
```

```
Close)|
+------------------+-------------+-------------
+----------------------+
|1742.9566644206718|        753.67|       3531.45|
655.9576061129327|
+------------------+-------------+-------------
+----------------------+
```

Check missing values

Fonction qui permet de verifier s'il y a des valeurs nulles dans chaque colonnes du DataFrame donnee en parametre.

```
def check_missing(df):
    """
        :param df : DataFrame
        :return : Show NULL values
    """
    df.select(*[
    (
        count(when((isnan(c) | col(c).isNull()), c)) if t not in
"timestamp"
        else count(when(col(c).isNull(), c))
    ).alias(c)
    for c, t in df.dtypes if c in df.columns
]).show()
```

Nous verifions pour chaque dataframe

```
for df in LIST_DF:
    print("Missing values for each column of", df.first()
["company_name"], "DataFrame")
    check_missing(df)
```

```
Missing values for each column of AMAZON DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of APPLE DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of FACEBOOK DataFrame
+----+----+---+----+-----+------+---------+------------+
```

```
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of GOOGLE DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of MICROSOFT DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of TESLA DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+

Missing values for each column of ZOOM DataFrame
+----+----+---+----+-----+------+---------+------------+
|Date|High|Low|Open|Close|Volume|Adj Close|company_name|
+----+----+---+----+-----+------+---------+------------+
|   0|   0|  0|   0|    0|     0|        0|           0|
+----+----+---+----+-----+------+---------+------------+
```

Correlation

Pour chaque DataFrame, nous allons étudier les corrélations entre les données. Pour ce faire, nous allons changer le type des donées en **Vector** puis appliquer dessus une fonction du package **ML** de pyspark pour avoir la matrice de corrélation. Nous crééons donc une fonction générique qui prend en paramètre un DataFrame et qui retourne sa matrice de corrélation associée.

```python
def get_corr_matrix(df):
    """
        :param df : DataFrame
        :return the correlation matrix of the DataFrame
    """
    vect_col = "corr_data"
    df = df.drop('Date','company_name') #Only numeric values
    assembler = VectorAssembler(inputCols=df.columns,
```

```
outputCol=vect_col)
    df_vector = assembler.transform(df).select(vect_col)
    print(df.columns)

    #Transform dataframe matrix in list by getting its values
    matrix = Correlation.corr(df_vector, vect_col).collect()[0]
[0].toArray().tolist()
    return matrix
```

Maintenant que la fonction est faite, essayons de l'appliquer au dataset de Amazon. Pour faciliter la visualisation, on transforme la matrice en dataframe que l'on show par la suite.

```
matrix_amazon = get_corr_matrix(df_amazon)
amazon_num = df_amazon.drop('Date','company_name')
df_matrix = spark.createDataFrame(matrix_amazon,amazon_num.columns)
df_matrix.show()
```

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```

```
22/05/21 18:18:28 WARN InstanceBuilder$NativeBLAS: Failed to load
implementation from:dev.ludovic.netlib.blas.JNIBLAS
22/05/21 18:18:28 WARN InstanceBuilder$NativeBLAS: Failed to load
implementation from:dev.ludovic.netlib.blas.ForeignLinkerBLAS
/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/context.py:1
25: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(

+------------------+------------------+------------------
+------------------+------------------+------------------+
|              High|               Low|              Open|
Close|            Volume|         Adj Close|
+------------------+------------------+------------------
+------------------+------------------+------------------+
|               1.0| 0.9991960804346885| 0.9995248053674247|
0.9994330757017126|0.16053643475450852|0.9994330757017126|
| 0.9991960804346885|               1.0|  0.999344391894671|
0.9994769384519298|0.13293677767054252|0.9994769384519298|
| 0.9995248053674247|  0.999344391894671|               1.0|
0.9988525402840831|0.14904748966416423|0.9988525402840831|
| 0.9994330757017126| 0.9994769384519298| 0.9988525402840831|
1.0| 0.1461794933995774|0.9999999999999999|
|0.16053643475450852|0.13293677767054252|0.14904748966416423|
0.1461794933995774|                1.0|0.1461794933995774|
| 0.9994330757017126| 0.9994769384519298| 0.9988525402840831|
0.9999999999999999| 0.1461794933995774|              1.0|
+------------------+------------------+------------------
+------------------+------------------+------------------+
```

Visualisation

On se rend compte que même comme cela, c'est assez difficile de se visualiser nos résultats. Nous allons donc faire une fonction qui permet de visualiser nos matrices de corrélation.

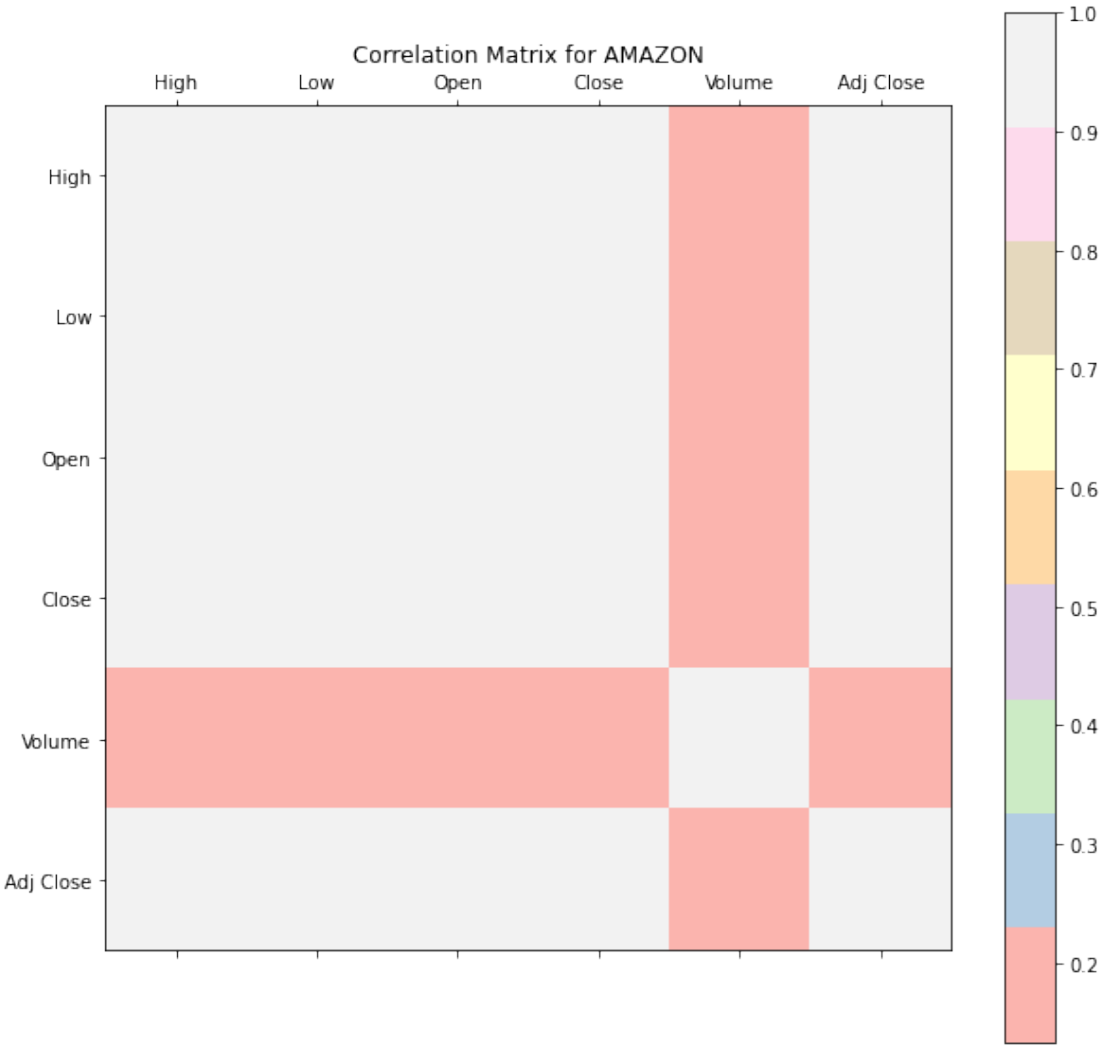
```
def visualization_corr_matrix(corr_matrix,columns, name):
    """
        :param
            - corr_matrix : correlation matrix
            - columns : numeric columns of the DataFrame
            - name : name of the DataFrame we are working with

        :return : Plotting the correlation matrix
    """
    fig=plt.figure(figsize=(10,10))
    ax=fig.add_subplot(111)

    ax.set_title("Correlation Matrix for " + name)
    ax.set_xticklabels(['']+columns)
    ax.set_yticklabels(['']+columns)
    cax=ax.matshow(corr_matrix,cmap=cm.Pastel1)
    fig.colorbar(cax)

    plt.show()
```

Pour chaque DataFrame, on regarde donc sa matrice de corrélation

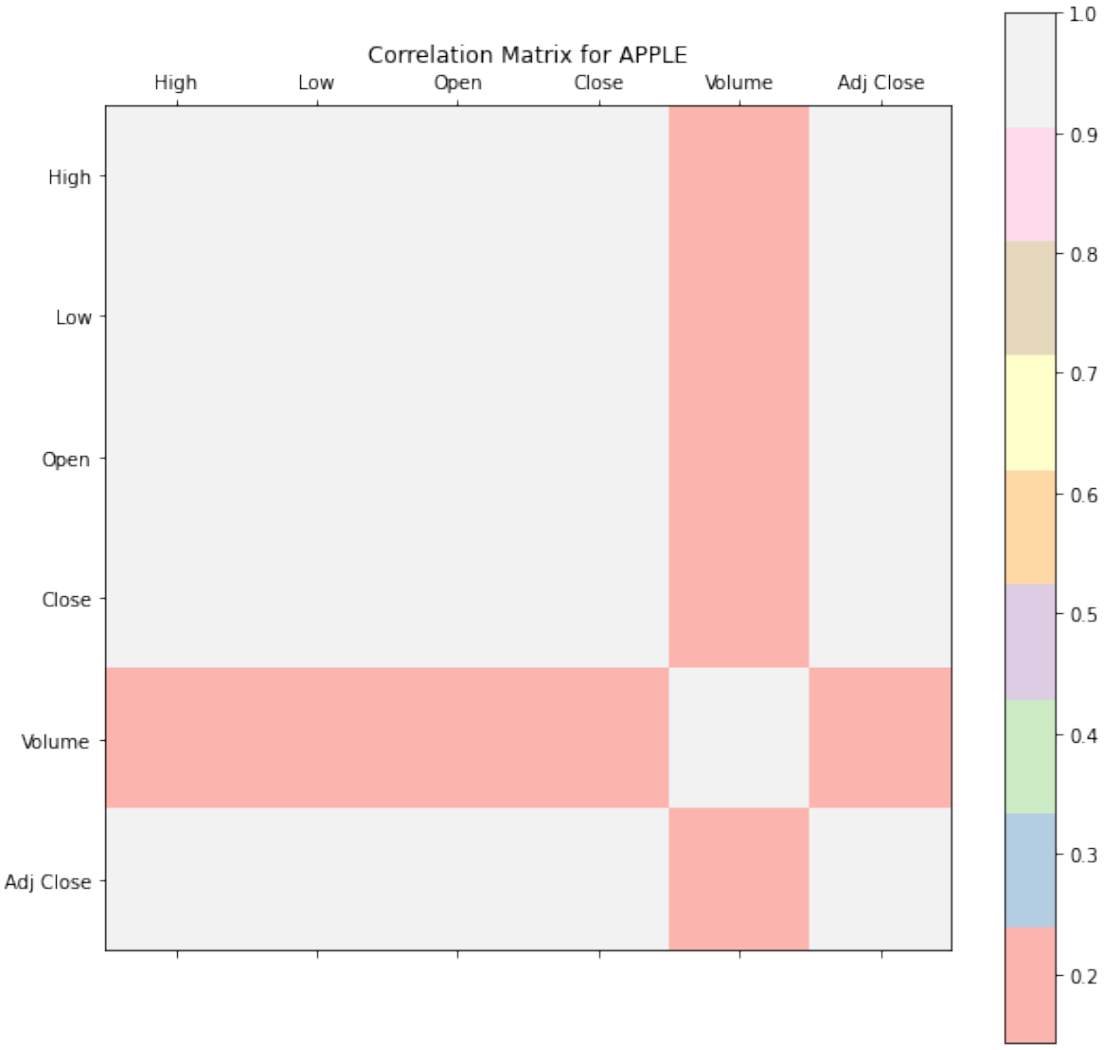
```
for df in LIST_DF:
    matrix = get_corr_matrix(df)
    name = df.first()['company_name']
    copy_num = df.drop('Date','company_name') # copy dataframe and
drop non-numeric columns
    visualization_corr_matrix(matrix, copy_num.columns, name)
```

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```

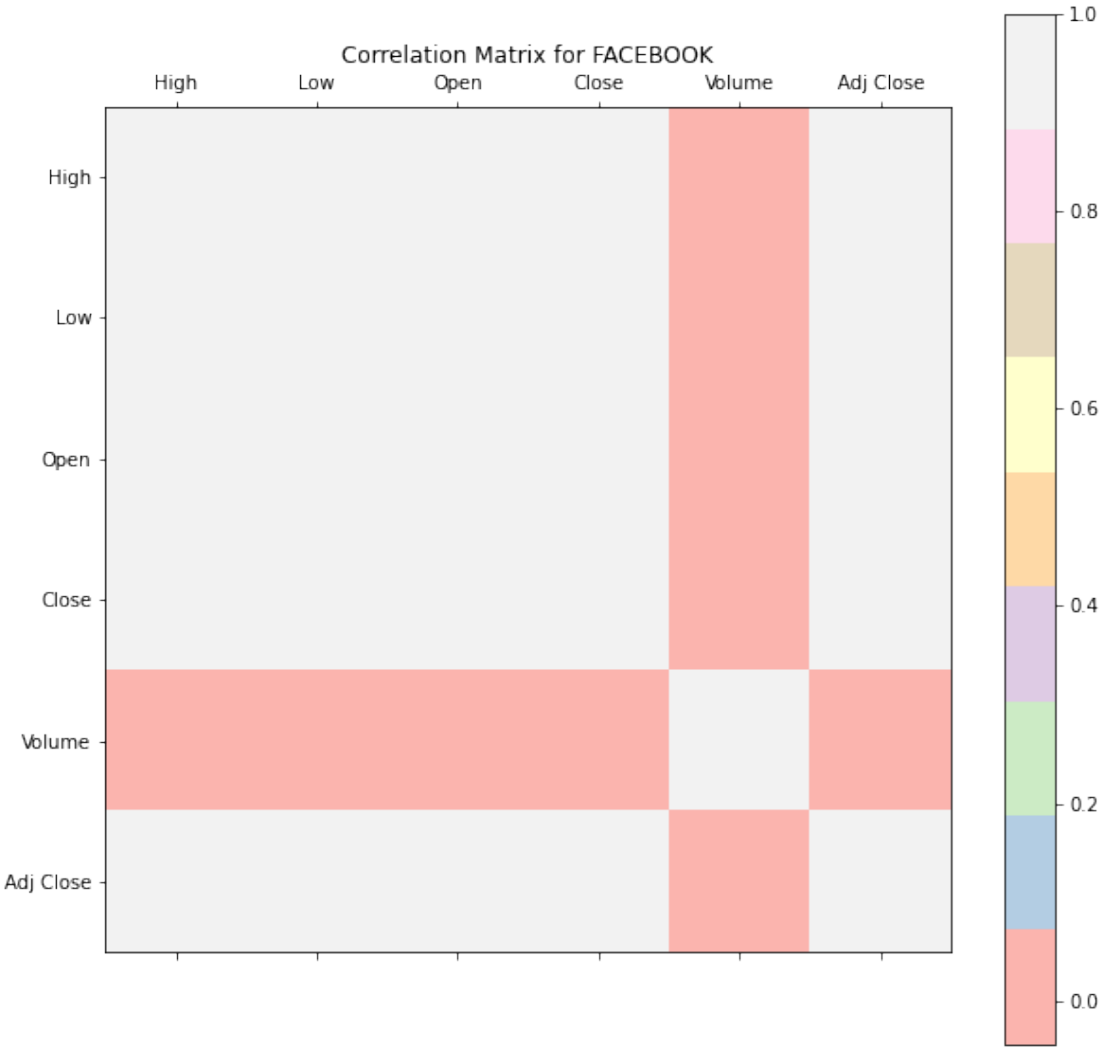
```
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```



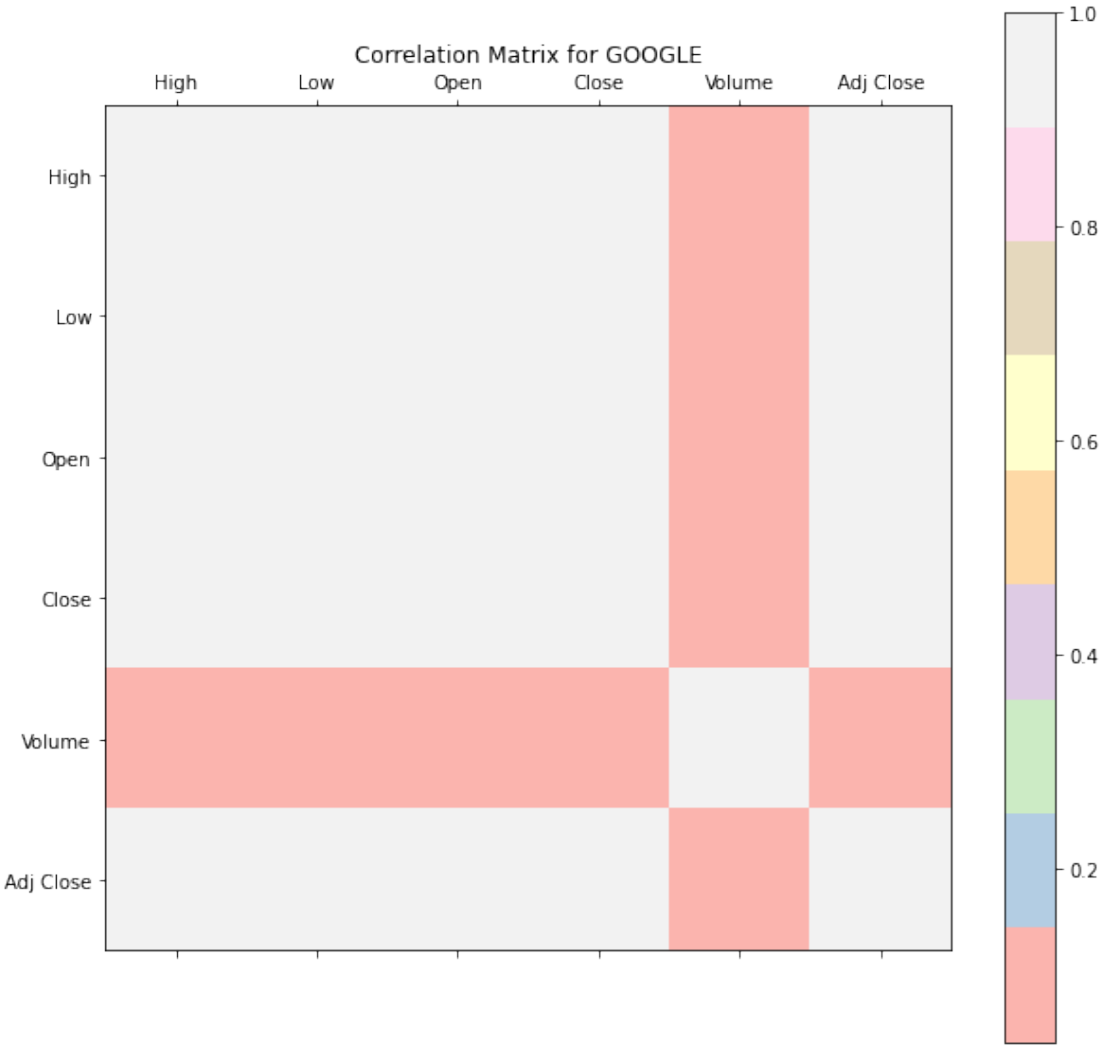

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```



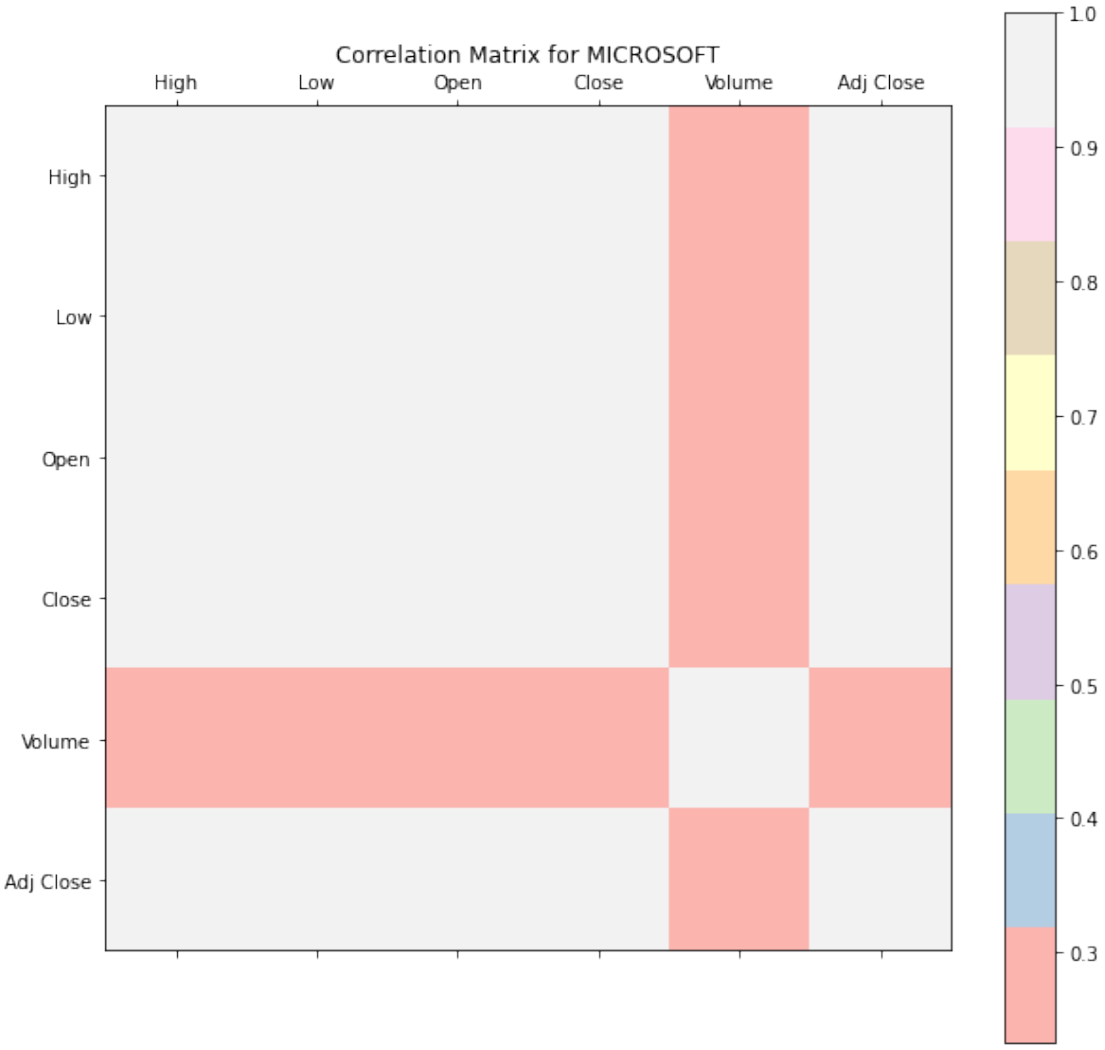

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```



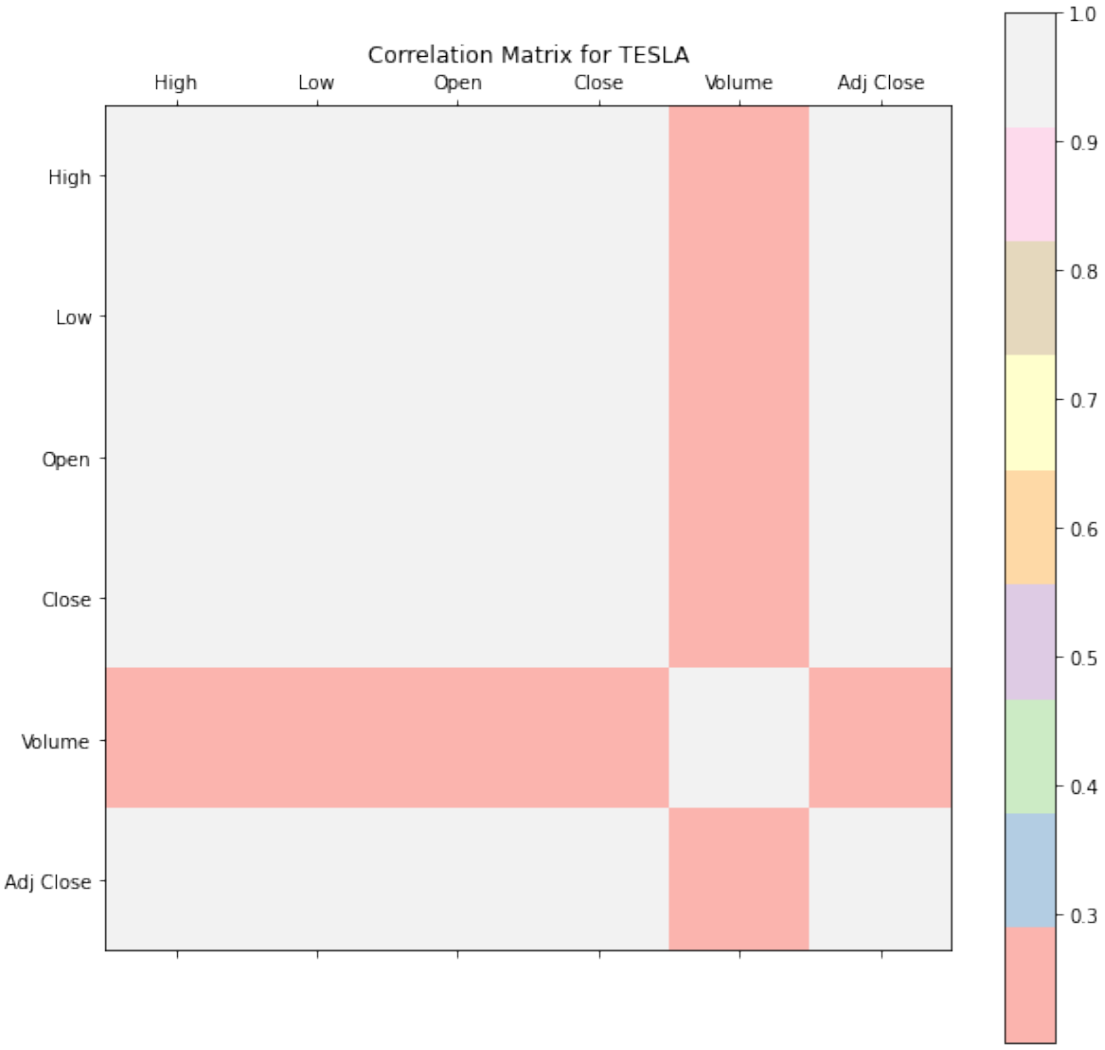

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```



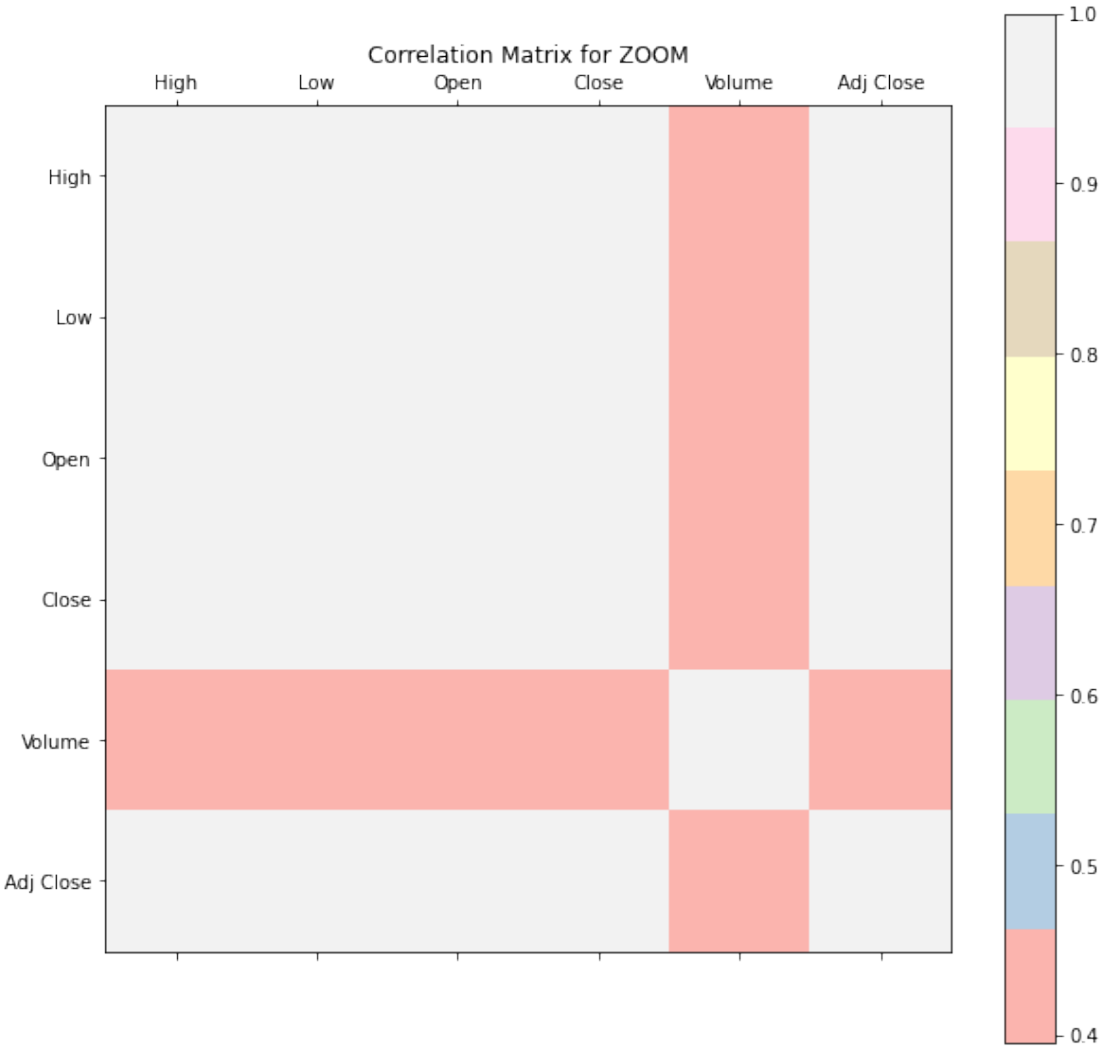

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```




```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```




```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']
```



2. Questions on the data

- What is the average of the opening and closing prices for each stock price and for different time periods (week, month, year)

On créé une fonction qui calcule la moyenne des prix selon une période précise. Pour nous aider, on créée également une fonction auxiliaire qui permet de construire une liste correspondant à la bonne temporalité demandée.

```
#Construct the list containing values foolowing the correct period
def get_list_period(df, w=False, m=False, y=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise
            - y : Boolean True if we want a year period, False
```

```
otherwise

        :return : List containing date values corresponding to the
period.
            - [2017,2018,...] for years
            - [1, 2, 3, 4, ..., 12] for months
            - [1, 2, ..., 52] for weeks
    """
    list = []
    if w:
        dates = df.select(weekofyear(df['Date'])).collect()
        for i in range(len(dates)):
            if dates[i][0] not in list:
                list.append(dates[i][0])
    else:
        dates = df.select('Date').collect() #convert column to list
        for date in dates:
            if y:
                if date[0].year not in list:
                    list.append(date[0].year)
            else:
                if date[0].month not in list:
                    list.append(date[0].month)
    list.sort()
    return list

def average_price(df, w=False, m=False, y=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise
            - y : Boolean True if we want a year period, False
otherwise

        :return : Show average for opening and closing price
calculated following the period choose
            - if year -> [2017, 2018, ...] -> average open and close
price for 2017, 2018, ...
            - if month -> [1, 2, ...] -> average open and clos eprice
for 1, 2, 3, ...
            - if week -> [1, 2, ..., 52] -> average open and clos
eprice for 1, 2, 3, ..., 52
    """
    list_period = get_list_period(df, w, m, y)
    for i in list_period:
        #Filter the Dataframe following the period choose
        if y: df_filtered = df.filter(year(df['Date']) == i)
```

```
        elif m: df_filtered = df.filter(month(df['Date']) == i)
        else: df_filtered = df.filter(weekofyear(df['Date']) == i)

        print("Average open price and close price in", i)
        df_filtered.agg(mean(df_filtered['Open']),
mean(df_filtered['Close'])).show()

for df in LIST_DF:
    print("Average price for each year of", df.first()
['company_name'])
    average_price(df, y=True)
```

```
Average price for each year of AMAZON
Average open price and close price in 2017
+----------------+-----------------+
|       avg(Open)|       avg(Close)|
+----------------+-----------------+
|968.275618959708|968.1670116409363|
+----------------+-----------------+

Average open price and close price in 2018
+------------------+------------------+
|         avg(Open)|        avg(Close)|
+------------------+------------------+
|1644.0727091633466|1641.7261758629545|
+------------------+------------------+

Average open price and close price in 2019
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|1788.7461896623884|1789.189206077939|
+------------------+-----------------+

Average open price and close price in 2020
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|2636.5054538710433|2636.649604240712|
+------------------+-----------------+

Average price for each year of APPLE
Average open price and close price in 2017
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|37.61122511297583|37.637768870805836|
+-----------------+------------------+

Average open price and close price in 2018
```

```
+------------------+------------------+
|         avg(Open)|        avg(Close)|
+------------------+------------------+
|47.277858642942874|47.263356698936676|
+------------------+------------------+

Average open price and close price in 2019
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|51.96727168370807|52.063988049825035|
+-----------------+------------------+

Average open price and close price in 2020
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|92.48257523237892|92.56351605198414|
+-----------------+-----------------+

Average price for each year of FACEBOOK
Average open price and close price in 2017
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|156.4810755756272|156.57617537053932|
+-----------------+------------------+

Average open price and close price in 2018
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|171.4729481427318|171.5109556889629|
+-----------------+-----------------+

Average open price and close price in 2019
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|181.56654727269733|181.6374996124752|
+------------------+-----------------+

Average open price and close price in 2020
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|230.78562233069425|231.0295712057613|
+------------------+-----------------+
```

```
Average price for each year of GOOGLE
Average open price and close price in 2017
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|921.1211927193569|921.7808373439834|
+-----------------+-----------------+

Average open price and close price in 2018
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|1113.554100735729|1113.225134131443|
+-----------------+-----------------+

Average open price and close price in 2019
+-----------------+-----------------+
|         avg(Open)|       avg(Close)|
+-----------------+-----------------+
|1187.0098210894873|1188.393057444739|
+-----------------+-----------------+

Average open price and close price in 2020
+-----------------+-----------------+
|         avg(Open)|        avg(Close)|
+-----------------+-----------------+
|1454.6135625880163|1456.6964127045333|
+-----------------+-----------------+

Average price for each year of MICROSOFT
Average open price and close price in 2017
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|71.95430287516925|71.98402421502954|
+-----------------+-----------------+

Average open price and close price in 2018
+-----------------+-----------------+
|         avg(Open)|        avg(Close)|
+-----------------+-----------------+
|101.12235092831799|101.03398411967365|
+-----------------+-----------------+

Average open price and close price in 2019
+-----------------+-----------------+
|         avg(Open)|        avg(Close)|
+-----------------+-----------------+
|130.33904787093874|130.38202400813026|
+-----------------+-----------------+
```

```
Average open price and close price in 2020
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|190.76480678836674|190.8616180419922|
+------------------+-----------------+

Average price for each year of TESLA
Average open price and close price in 2017
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|62.85924295980142|62.863258969736286|
+-----------------+------------------+

Average open price and close price in 2018
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|63.43669347269127|63.46198397328654|
+-----------------+-----------------+

Average open price and close price in 2019
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|54.60562690855965|54.706039686051625|
+-----------------+------------------+

Average open price and close price in 2020
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|259.2505243980833|259.50230853035725|
+-----------------+------------------+

Average price for each year of ZOOM
Average open price and close price in 2019
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|80.39499985769893|80.23938206876262|
+-----------------+-----------------+

Average open price and close price in 2020
+------------------+------------------+
|         avg(Open)|        avg(Close)|
+------------------+------------------+
```

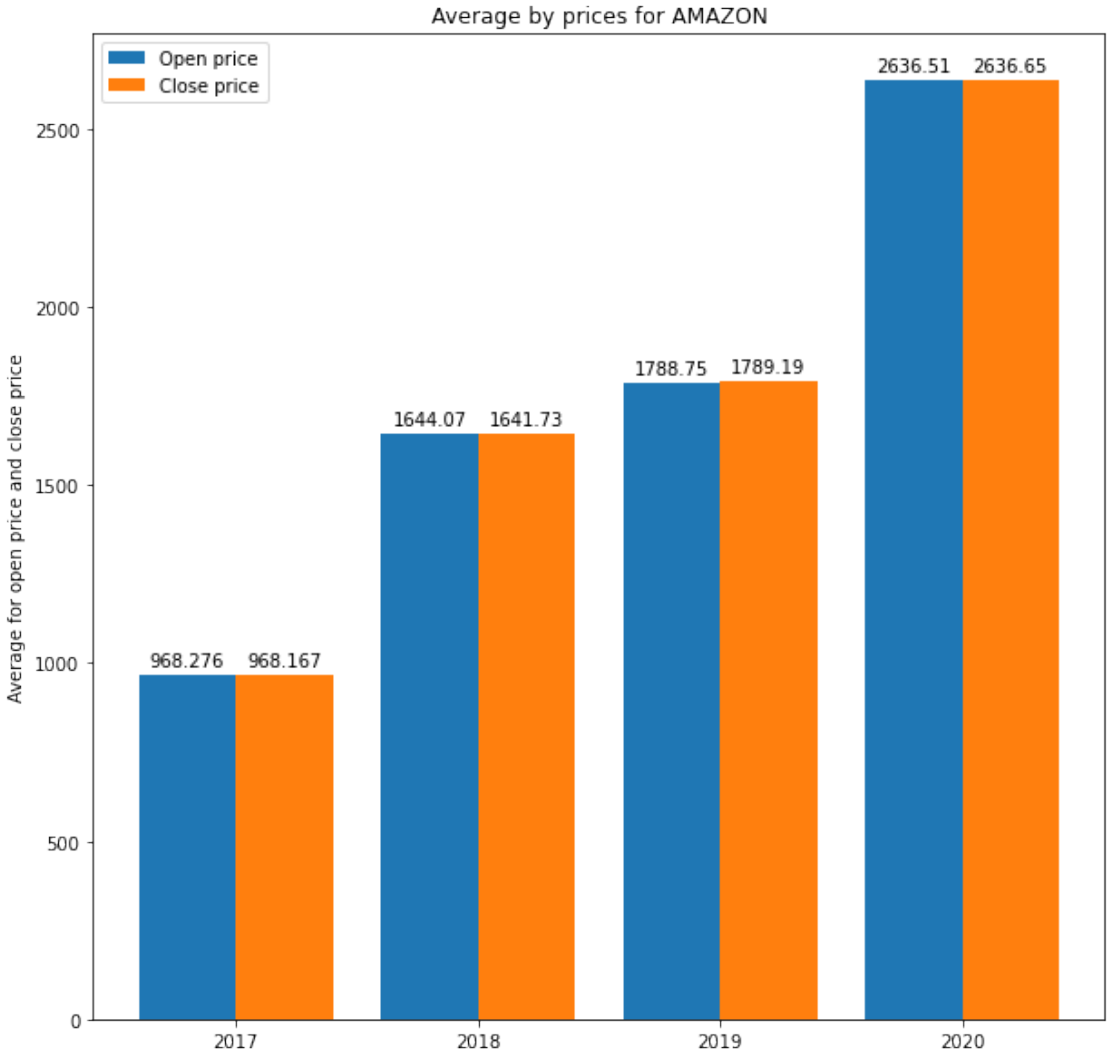
```
|251.14446795549514|251.74615088859852|
+------------------+------------------+
```

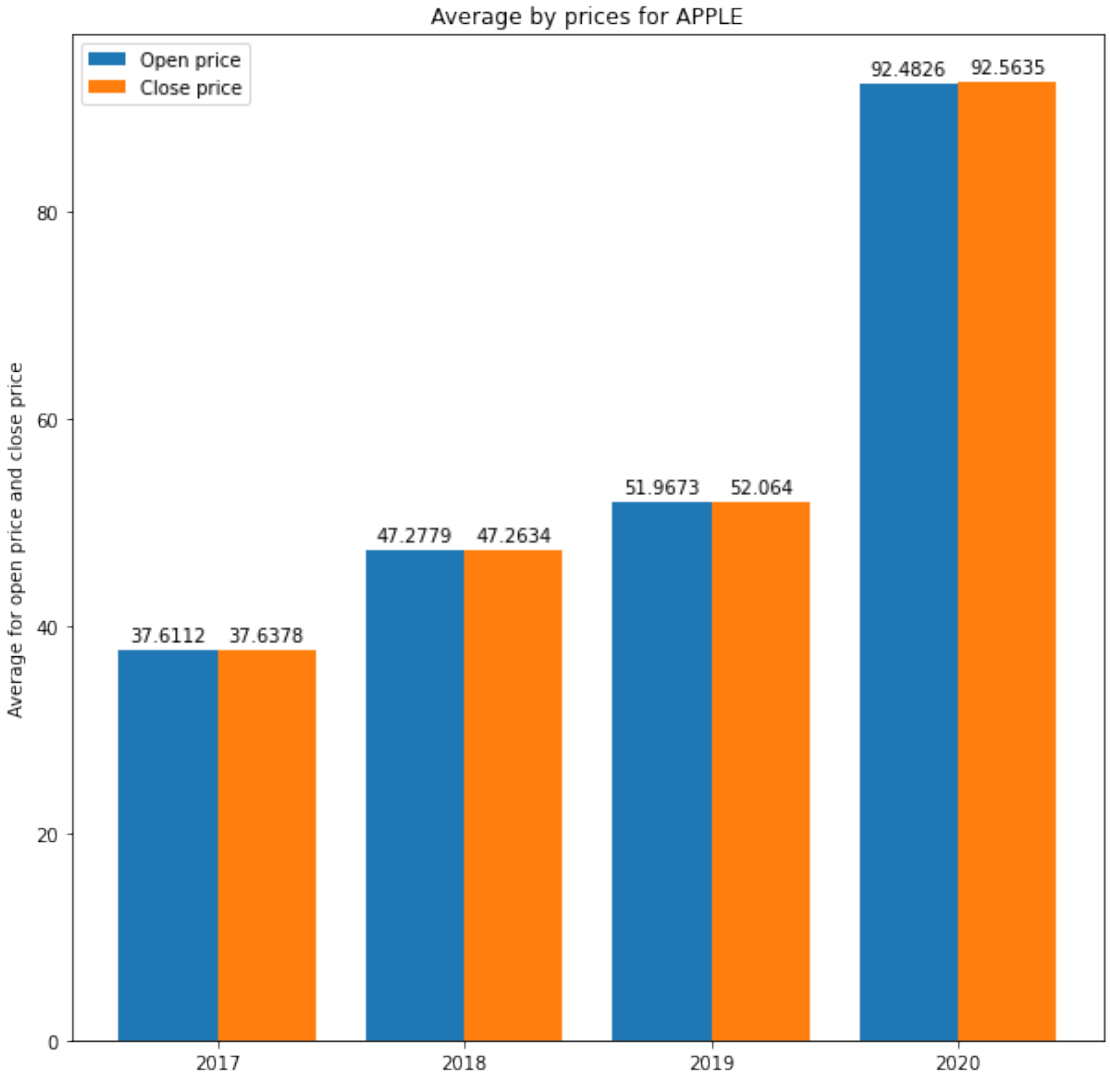
Afin de faciliter la visibilite de ces informations, nous faisons une fonction de visualisation pour **l'average_price** sur la periode d'annee, de mois ou de semaine.

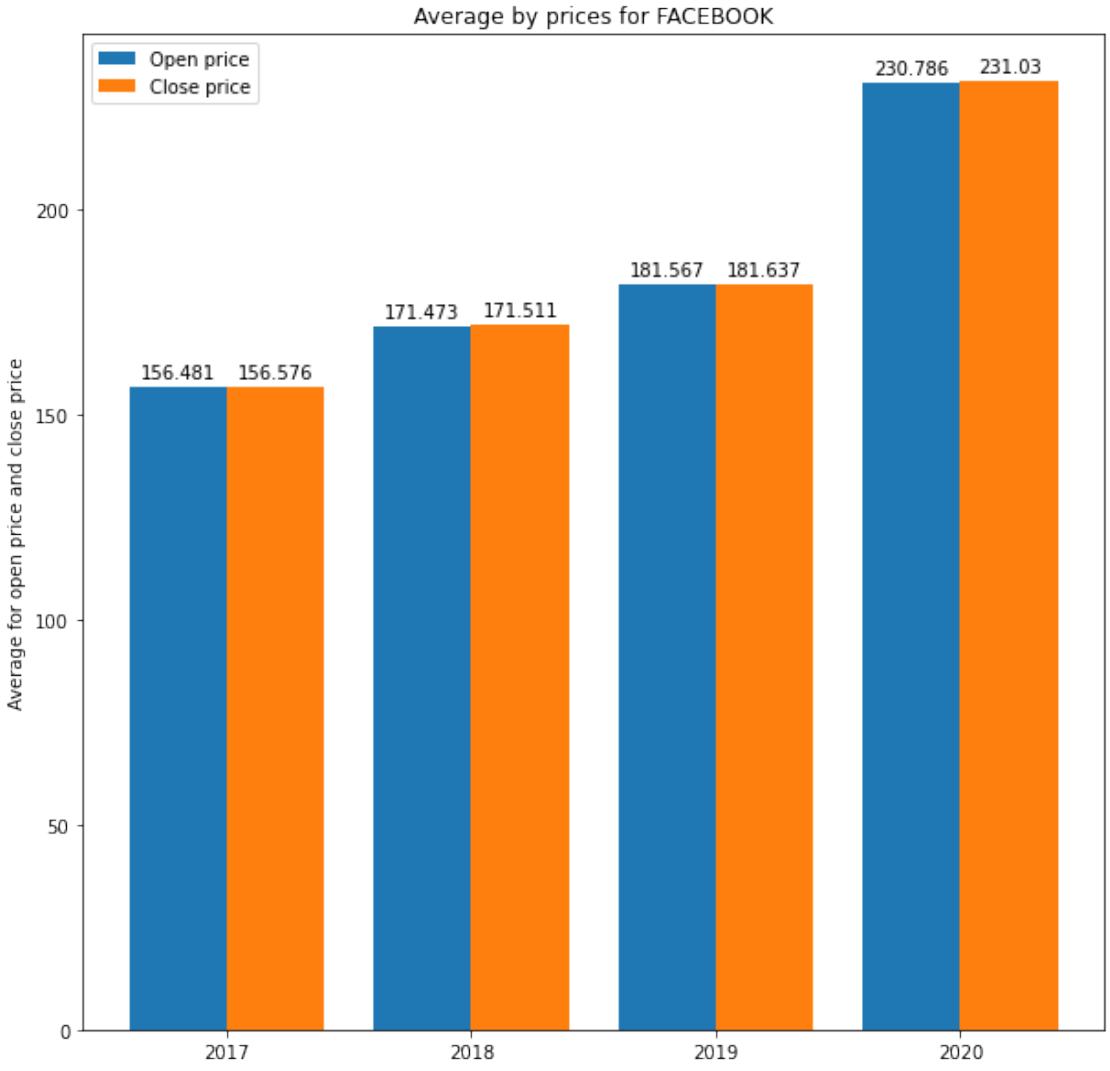
```
def plot_avg_price(df, w=False, m=False, y=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise
            - y : Boolean True if we want a year period, False
otherwise

        :return : Plotting the results obtained by the function
avg_price for each DataFrame
    """


    #Construct list of y axis that is period scale (year, week or
month)
    list_period = get_list_period(df, w, m, y)
    avg_open = []
    avg_close = []
    #Construct lists for average open and close prices
    for i in list_period:
        if y: df_filtered = df.filter(year(df['Date']) == i)
        elif m: df_filtered = df.filter(month(df['Date']) == i)
        else: df_filtered = df.filter(weekofyear(df['Date']) == i)


avg_open.append(df_filtered.agg(mean(df_filtered['Open'])).collect()
[0][0])

avg_close.append(df_filtered.agg(mean(df_filtered['Close'])).collect()
[0][0])

    #Create bars
    x = np.arange(len(list_period))
    fig, ax = plt.subplots(figsize=(10,10))
    width = 0.4
    open_bar = ax.bar(x - width/2, avg_open, width, label='Open
price')
    close_bar = ax.bar(x + width/2, avg_close, width, label='Close
price')
```

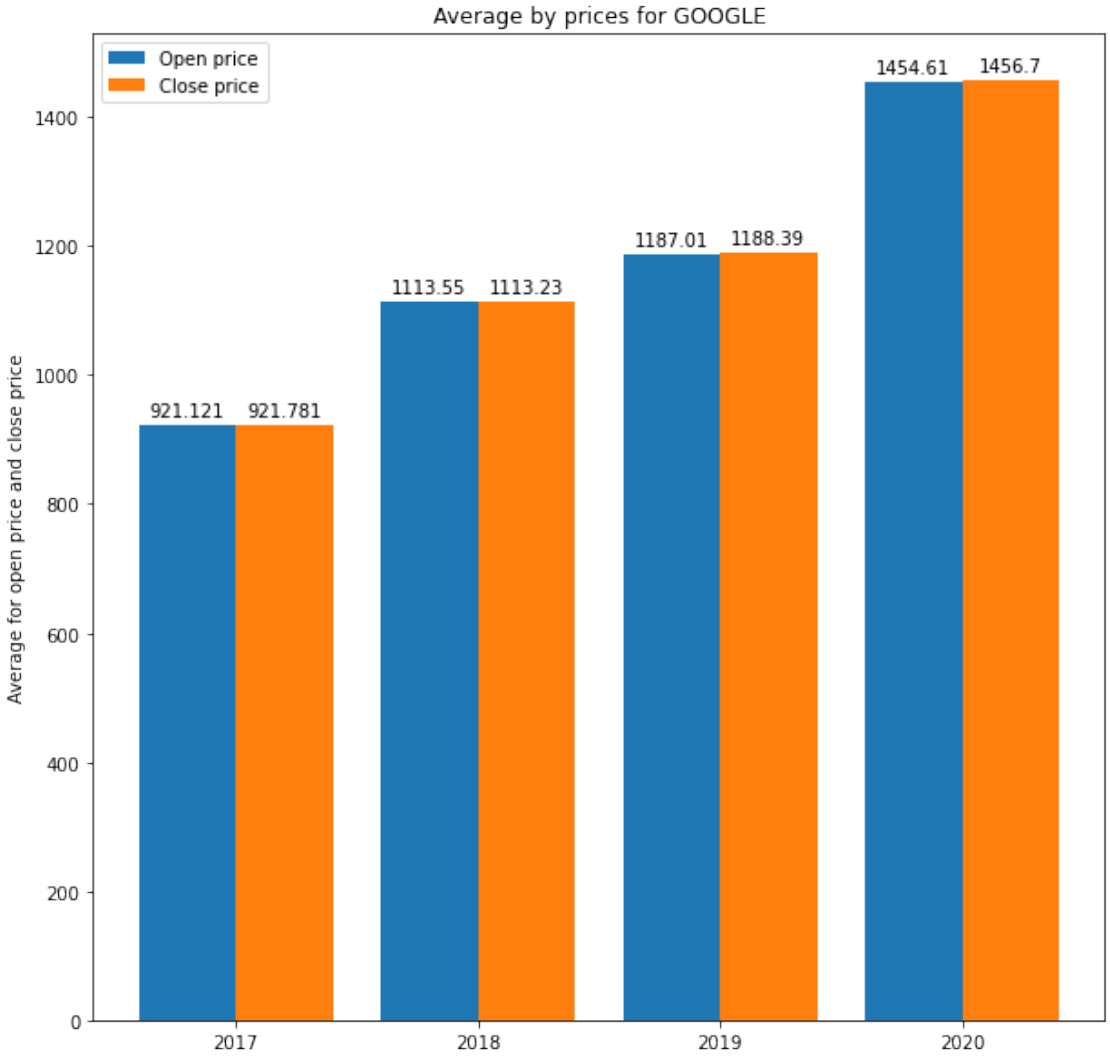
```
    #Legend
    name = df.first()['company_name']
    ax.set_ylabel('Average for open price and close price')
    ax.set_title('Average by prices for ' + name)
    ax.set_xticks(x, list_period)
    ax.legend()

    ax.bar_label(open_bar, padding=3)
    ax.bar_label(close_bar, padding=3)

    plt.show()
```

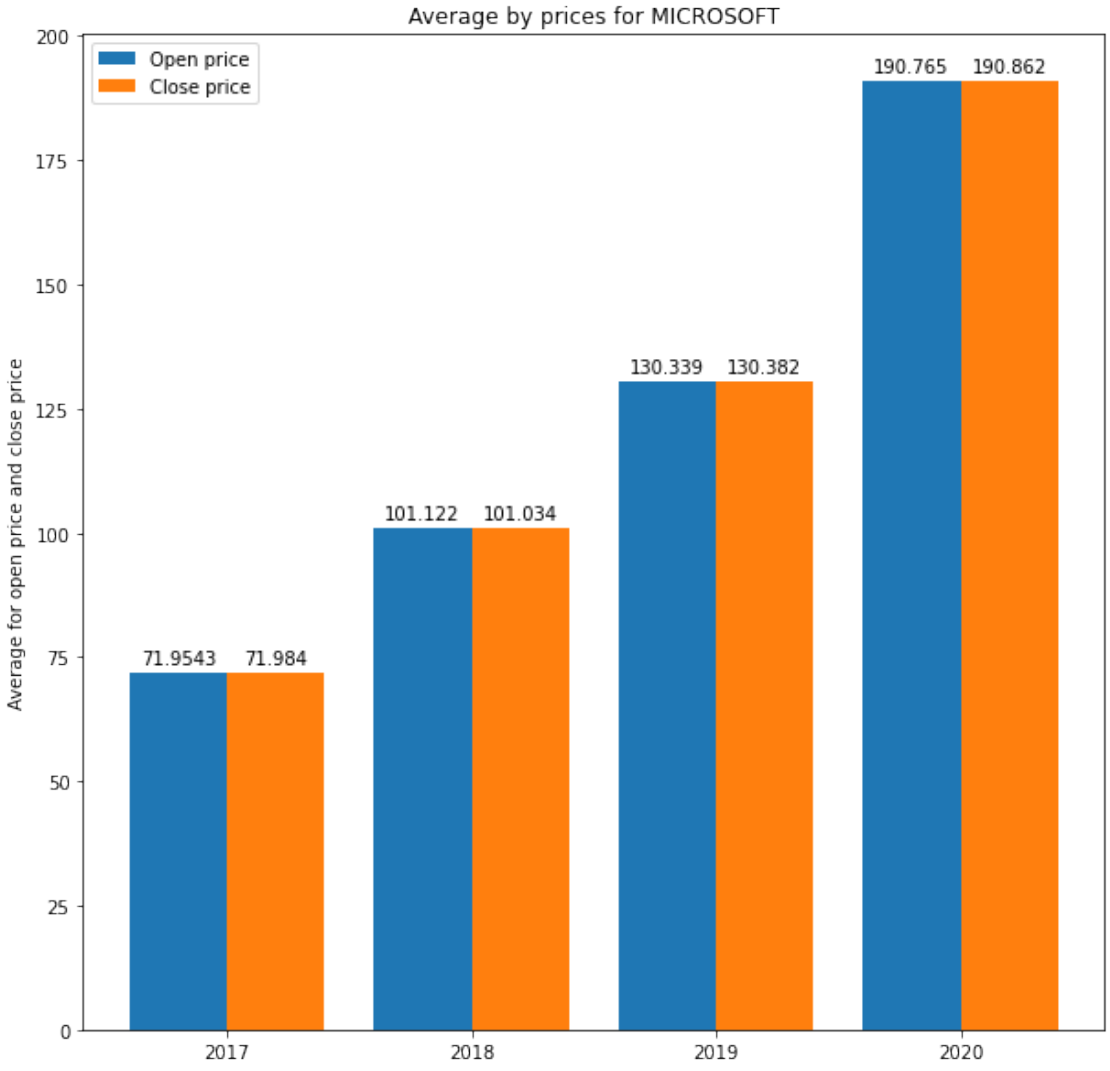
```
for df in LIST_DF:
    plot_avg_price(df,y=True)
```

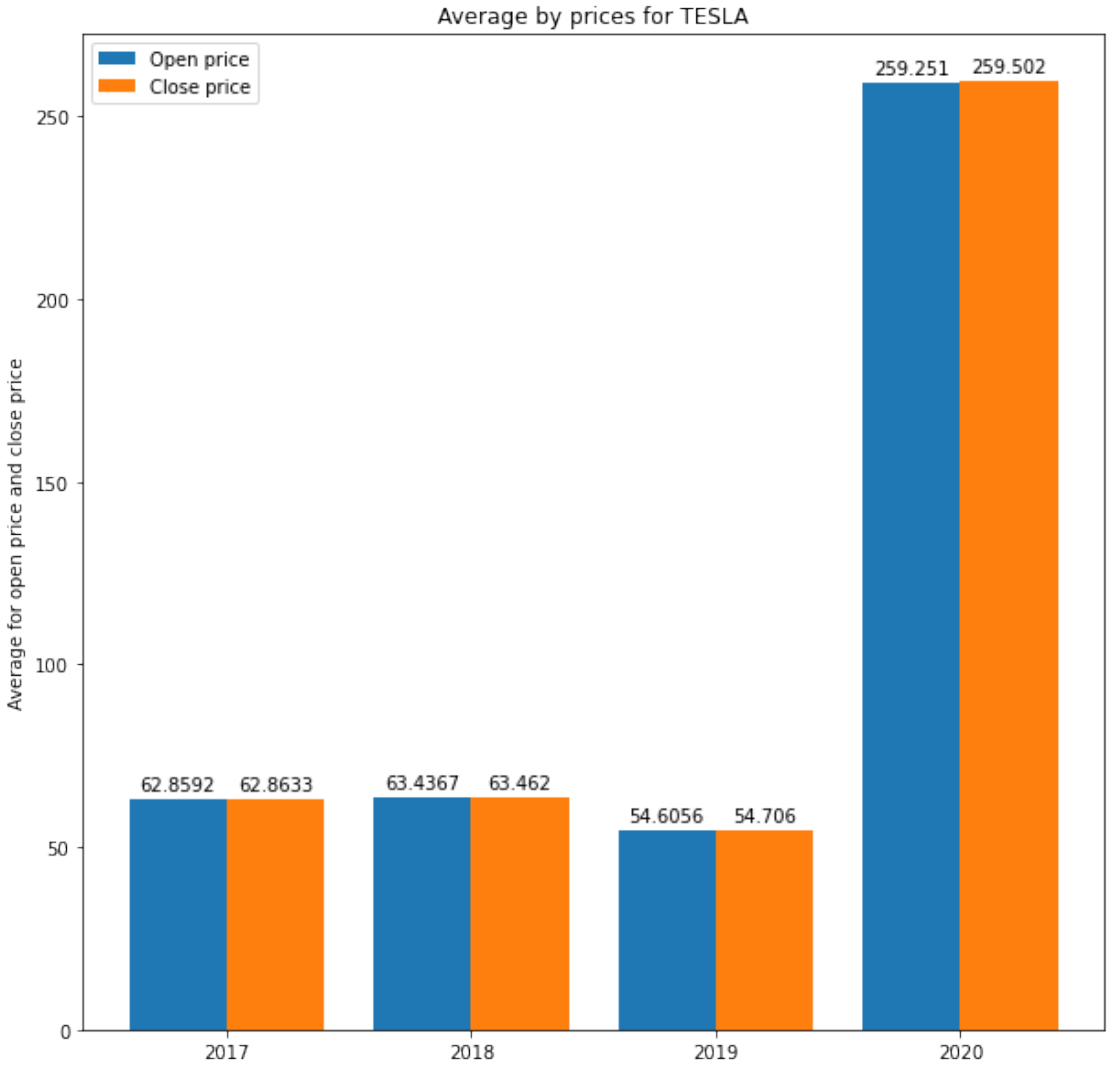


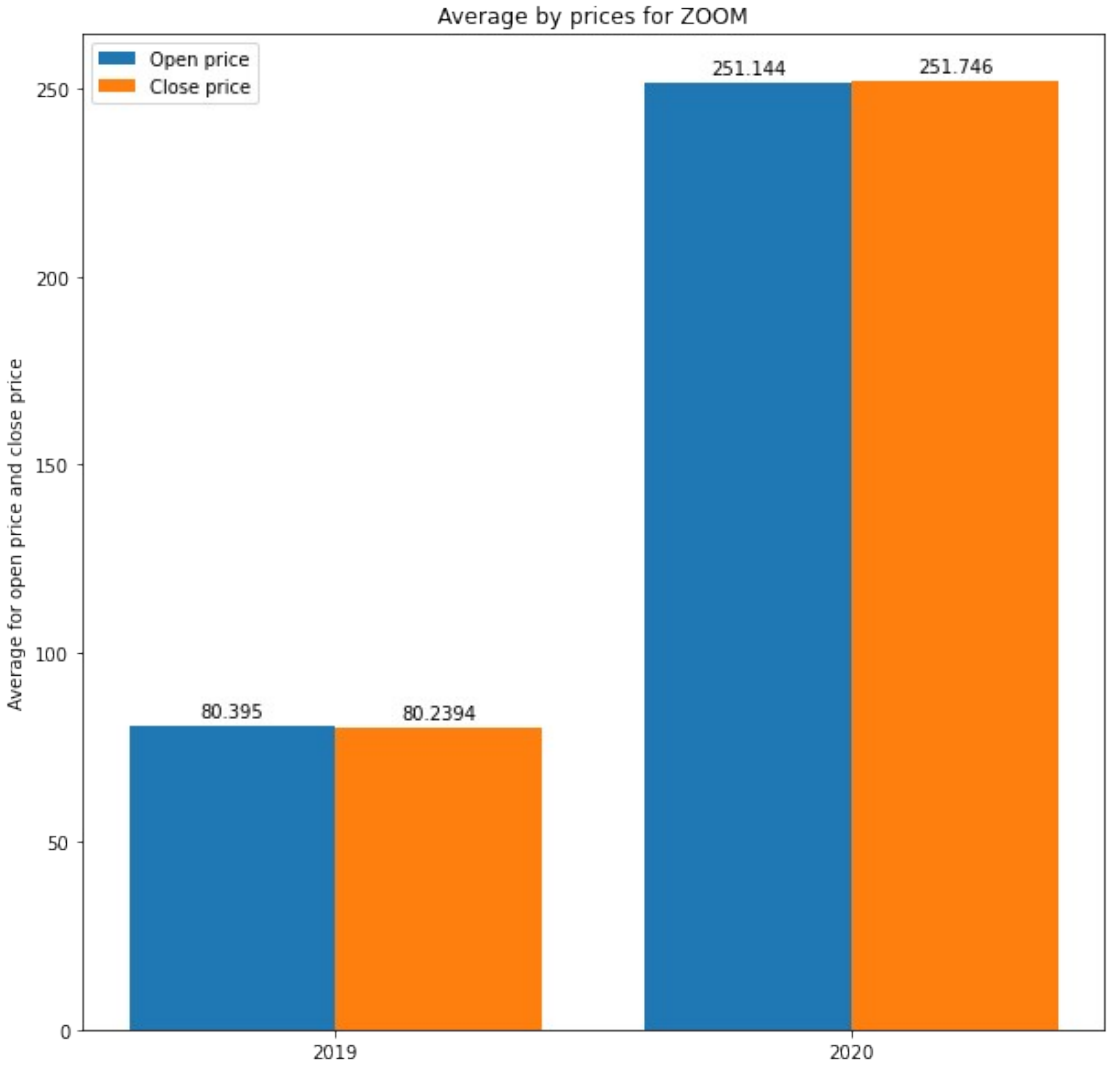

Average by prices for APPLE
Open price
Close price
Average for open price and close price
0
20
40
60
80
37.6112
37.6378
47.2779
47.2634
51.9673
52.064
92.4826
92.5635
2017
2018
2019
2020


Average by prices for FACEBOOK
Open price
Close price
Average for open price and close price
0
50
100
150
200
156.481
156.576
171.473
171.511
181.567
181.637
230.786
231.03
2017
2018
2019
2020


Average by prices for GOOGLE
Open price
Close price
Average for open price and close price
0
200
400
600
800
1000
1200
1400
921.121
921.781
1113.55
1113.23
1187.01
1188.39
1454.61
1456.7
2017
2018
2019
2020


Average by prices for MICROSOFT
Open price
Close price
Average for open price and close price
200
175
150
125
100
75
50
25
0
71.9543
71.984
101.122
101.034
130.339
130.382
190.765
190.862
2017
2018
2019
2020


Average by prices for TESLA
Open price
Close price
Average for open price and close price
0
50
100
150
200
250
62.8592
62.8633
63.4367
63.462
54.6056
54.706
259.251
259.502
2017
2018
2019
2020



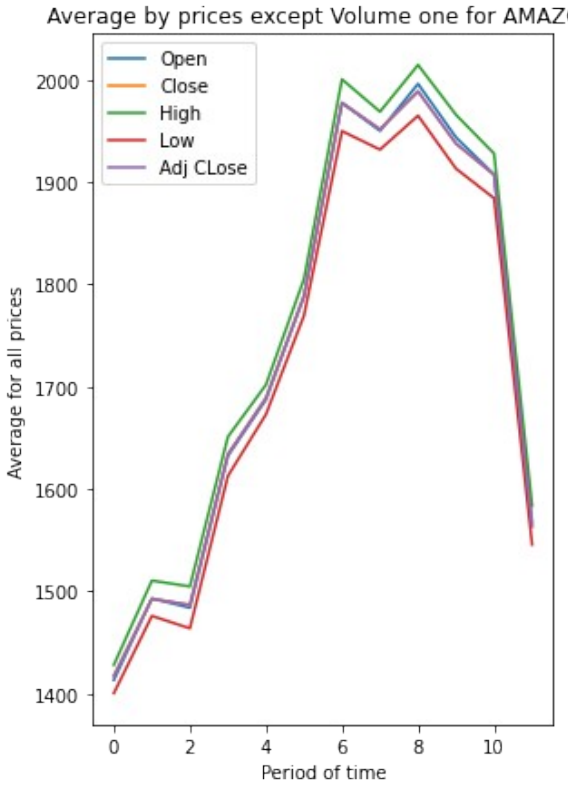
- How do the stock prices change day to day and month to month (may be you can create new columns to save those calculations)

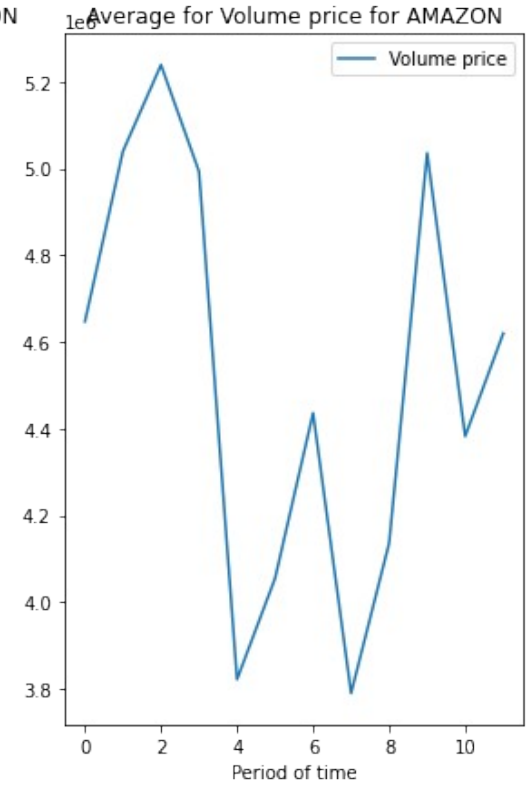
Fonction qui permet de visualiser les différents prix d'une certaine action soit au jours le jour ou chaque mois

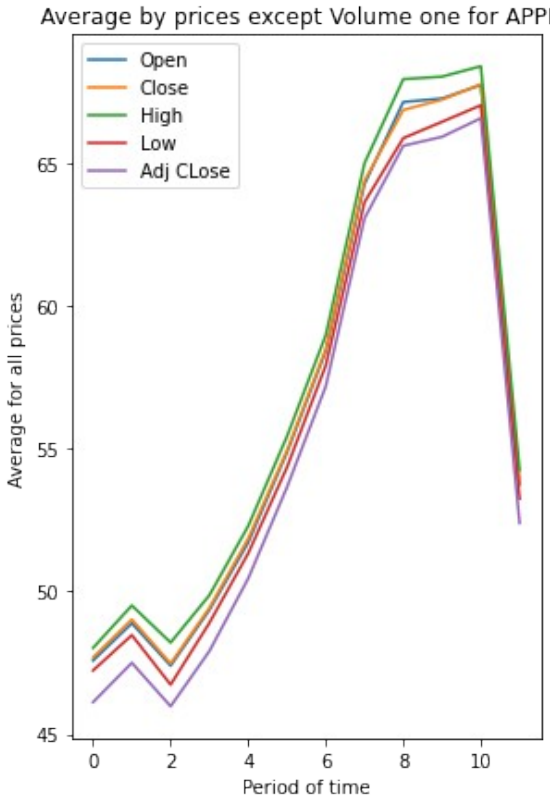
```
def plot_evolution_stock_prices(df, d=False,m=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise

        :return Plotting evolution for each price for each DataFrame
following a
                specific period(each month or day to day).
```

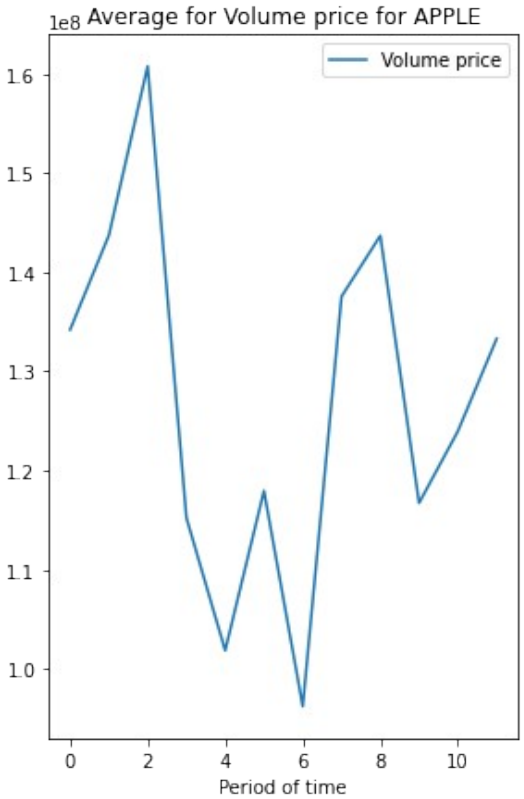
```
                For each DataFrame, we have a subplot with all prices
except Volume price and another
                subplot with Volume price. This separation is caused
by the values of Volume price
                that are higher than other prices
    """
    #Construct list of y axis that is period scale (year, week or
month)
    period = []
    if d:
        dates = df.select(dayofyear(df['Date'])).collect()
        for i in range(len(dates)):
                period.append(dates[i][0])
    else:
        dates = df.select('Date').collect() #convert column to list
        for date in dates:
            if date[0].month not in period:
                period.append(date[0].month)

    if m: period.sort() #sort for months not for days

    #Lists for all prices except volume
    avg_open = []
    avg_close = []
    avg_high = []
    avg_low = []
    avg_adj_close = []
    prices = [
        (avg_open, "Open"),
        (avg_close, "Close"),
        (avg_high, "High"),
        (avg_low, "Low"),
        (avg_adj_close, "Adj CLose")
        ]

    avg_volume = []

    #Construct lists
    for i in period:
        if m: df_filtered = df.filter(month(df['Date']) == i)
        else: df_filtered = df.filter(dayofyear(df['Date']) == i)

        for (price, name_price) in prices:

price.append(df_filtered.agg(mean(df_filtered[name_price])).collect()
[0][0])


avg_volume.append(df_filtered.agg(mean(df_filtered['Volume'])).collect
()[0][0])
```

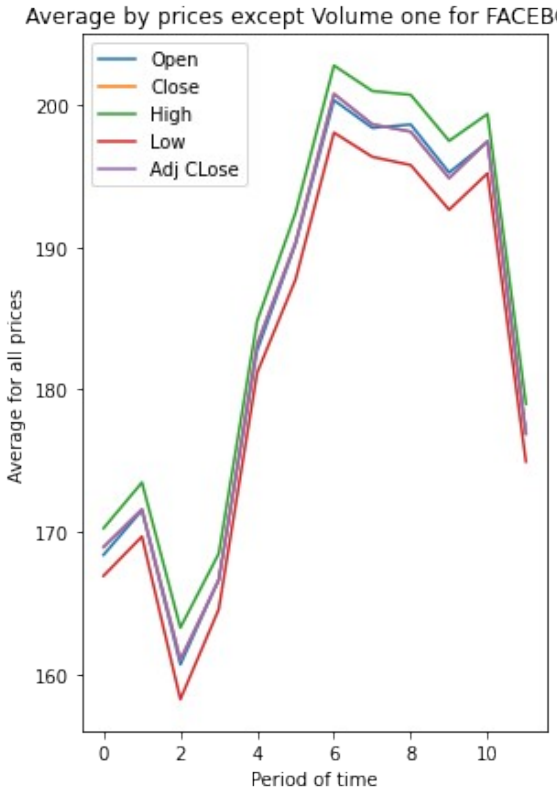
```
    #PLotting
    name = df.first()['company_name']
    x = np.arange(len(period))
    figure, axis = plt.subplots(1, 2, figsize=(10,7))

    #First plot for al prices except volume due to the diffence in
values
    for (price, name_price) in prices:
        axis[0].plot(x, price, label=name_price)

    axis[0].set_title('Average by prices except Volume one for ' +
name)
    axis[0].set_xlabel("Period of time")
    axis[0].set_ylabel('Average for all prices')
    axis[0].legend()

    #2d plot for volume prices
    axis[1].plot(x, avg_volume, label="Volume price")
    axis[1].set_title('Average for Volume price for ' + name)
    axis[1].legend()
    axis[1].set_xlabel('Period of time')


    plt.show()

for df in LIST_DF:
    plot_evolution_stock_prices(df,m=True)
```

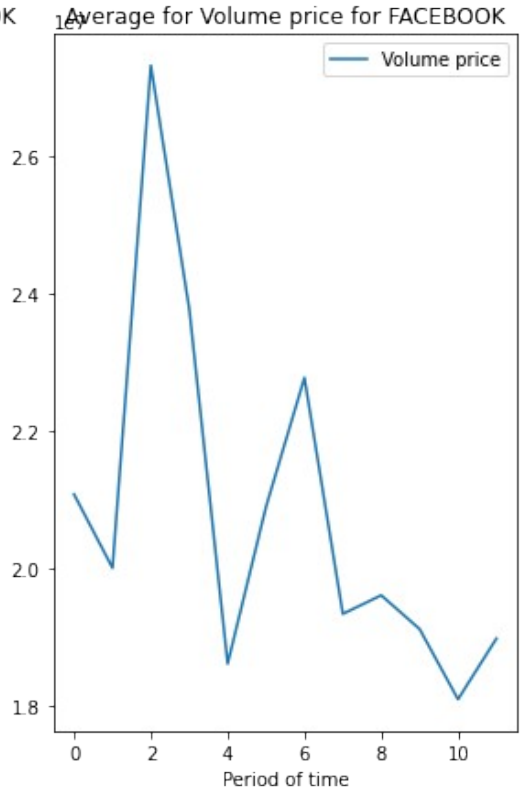

Average by prices except Volume one for AMAZON
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time

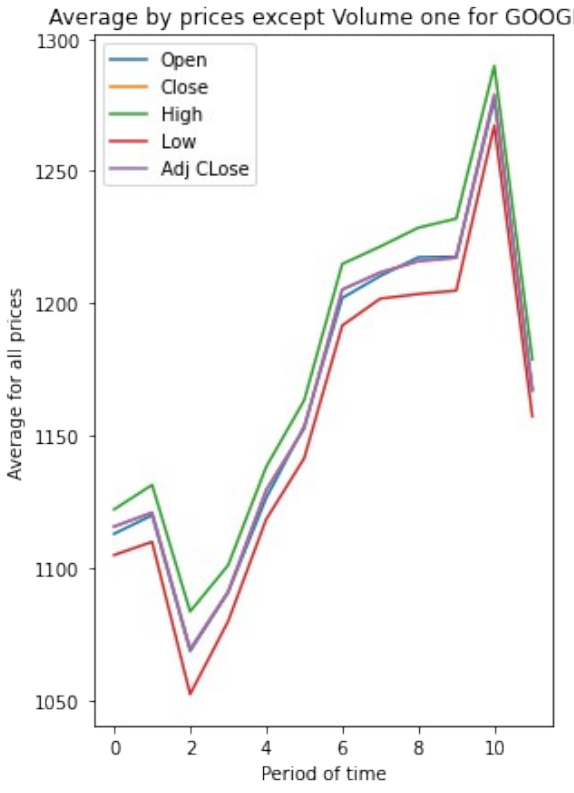

Average for Volume price for AMAZON
1e6
Volume price
Period of time

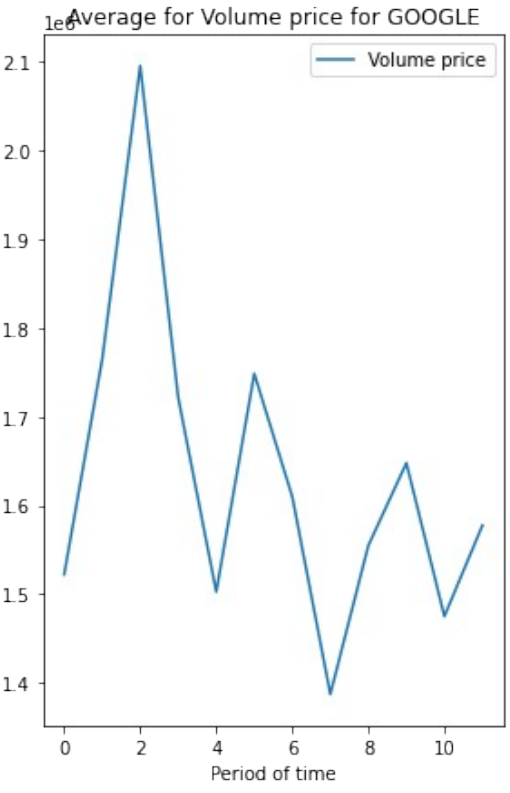

Average by prices except Volume one for APPLE
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time

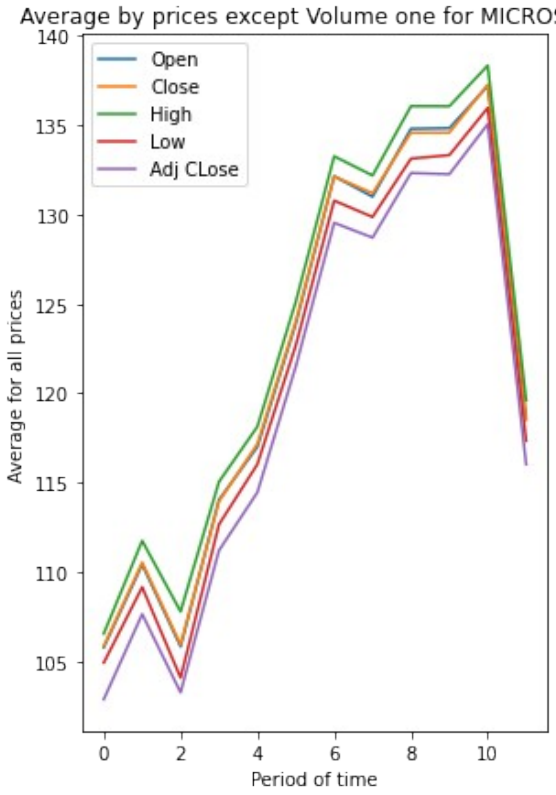

Average for Volume price for APPLE
1e8
Volume price
Period of time

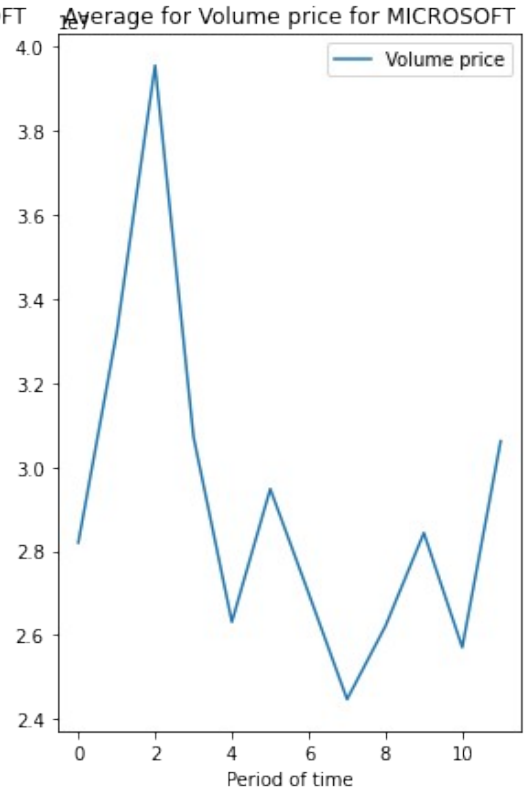

Average by prices except Volume one for FACEBOOK
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time

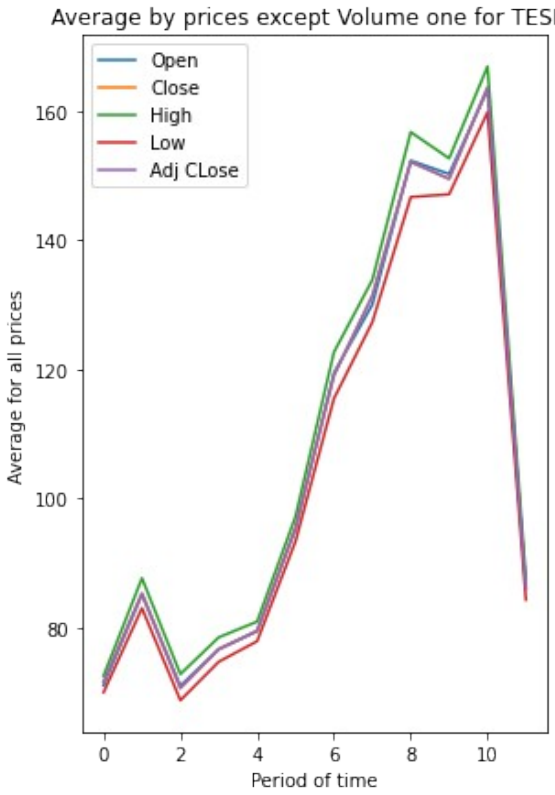

Average for Volume price for FACEBOOK
1e7
Volume price
Period of time

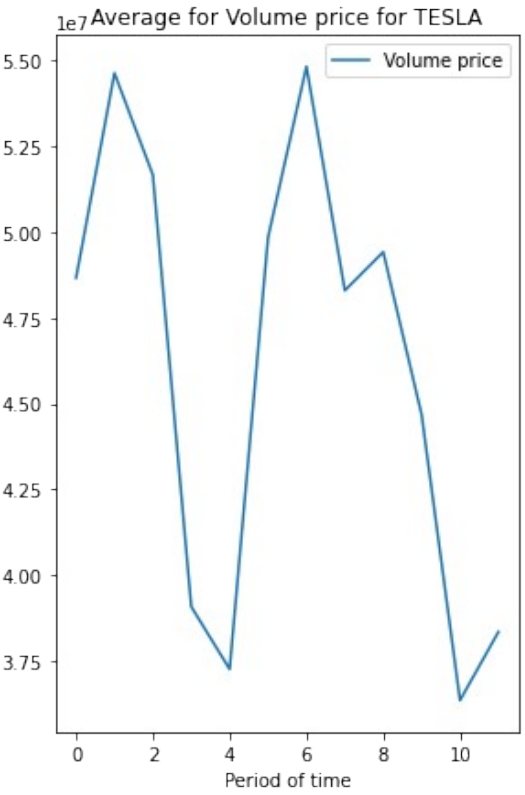

Average by prices except Volume one for GOOGLE
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time

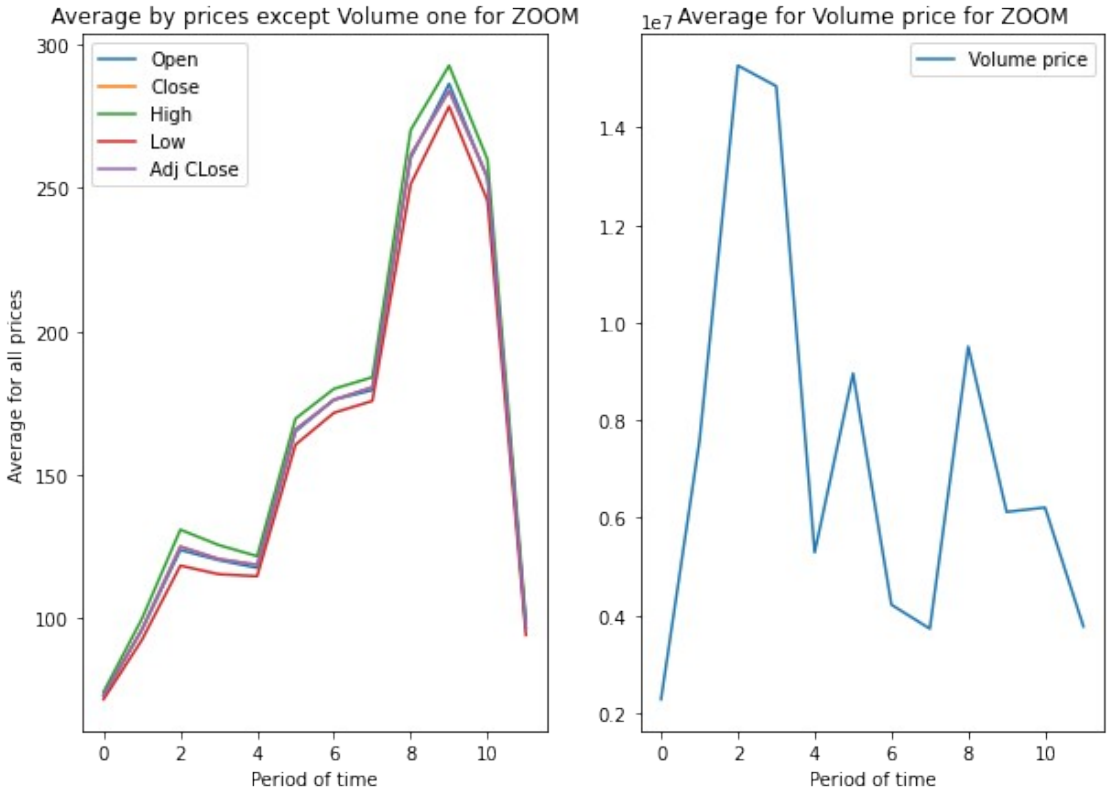

Average for Volume price for GOOGLE
1e6
Volume price
Period of time


Average by prices except Volume one for MICROSOFT
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time


Average for Volume price for MICROSOFT
1e7
Volume price
Period of time


Average by prices except Volume one for TESLA
Open
Close
High
Low
Adj CLose
Average for all prices
Period of time


Average for Volume price for TESLA
1e7
Volume price
Period of time



- Based on the opening and closing price, calculate the daily return of each stock

```
def daily_return(df):
    """
        :param df : Dataframe
        :return add in a column the daily return calculated day to day
    """
    df.withColumn("evolution_price", df['Close'] - df['Open']).show()

for df in LIST_DF:
    print(df.first()['company_name'])
    daily_return(df)
```

```
AMAZON
+-------------------+------+------+------+------+---------+---------
+------------+---------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|evolution_price|
+-------------------+------+------+------+------+---------+---------
+------------+---------------+
|2017-01-03 00:00:00|758.76| 747.7|757.92|753.67|3521100.0|   753.67|
AMAZON|          -4.25|
|2017-01-04 00:00:00|759.68| 754.2|758.39|757.18|2510500.0|   757.18|
AMAZON|      -1.210022|
|2017-01-05 00:00:00| 782.4|760.26|761.55|780.45|5830100.0|   780.45|
AMAZON|      18.900024|
|2017-01-06 00:00:00|799.44|778.48|782.36|795.99|5986200.0|   795.99|
```

```
AMAZON|      13.630005|
|2017-01-09 00:00:00|801.77|791.77| 798.0|796.92|3446100.0|   796.92|
AMAZON|     -1.0800171|
|2017-01-10 00:00:00| 798.0|789.54| 796.6| 795.9|2558400.0|    795.9|
AMAZON|     -0.6999512|
|2017-01-11 00:00:00| 799.5|789.51|793.66|799.02|2992800.0|   799.02|
AMAZON|      5.3600464|
|2017-01-12 00:00:00|814.13| 799.5|800.31|813.64|4873900.0|   813.64|
AMAZON|      13.330017|
|2017-01-13 00:00:00|821.65| 811.4|814.32|817.14|3791900.0|   817.14|
AMAZON|      2.8200073|
|2017-01-17 00:00:00| 816.0|803.44| 815.7|809.72|3670500.0|   809.72|
AMAZON|     -5.9800415|
|2017-01-18 00:00:00|811.73|804.27| 809.5|807.48|2354200.0|   807.48|
AMAZON|     -2.0200195|
|2017-01-19 00:00:00|813.51|807.32| 810.0|809.04|2540800.0|   809.04|
AMAZON|      -0.960022|
|2017-01-20 00:00:00|816.02|806.26|815.28|808.33|3376200.0|   808.33|
AMAZON|      -6.950012|
|2017-01-23 00:00:00| 818.5|805.08| 806.8|817.88|2797500.0|   817.88|
AMAZON|      11.080017|
|2017-01-24 00:00:00|823.99| 814.5| 822.0|822.44|2971700.0|   822.44|
AMAZON|     0.44000244|
|2017-01-25 00:00:00|837.42|825.29|825.79|836.52|3922600.0|   836.52|
AMAZON|     10.7300415|
|2017-01-26 00:00:00|843.84| 833.0|835.53|839.15|3586300.0|   839.15|
AMAZON|       3.619995|
|2017-01-27 00:00:00| 839.7|829.44| 839.0|835.77|2998700.0|   835.77|
AMAZON|     -3.2299805|
|2017-01-30 00:00:00| 833.5|816.38| 833.0|830.38|3747300.0|   830.38|
AMAZON|      -2.619995|
|2017-01-31 00:00:00|826.99|819.56|823.75|823.48|3137200.0|   823.48|
AMAZON|    -0.27001953|
+-------------------+------+------+------+------+---------+---------
+------------+--------------+
only showing top 20 rows

APPLE
+-------------------+-------+-------+-------+-------+----------
+---------+------------+--------------+
|               Date|   High|    Low|   Open|  Close|    Volume|Adj
Close|company_name|evolution_price|
+-------------------+-------+-------+-------+-------+----------
+---------+------------+--------------+
|2017-01-03 00:00:00|29.0825|  28.69|  28.95|29.0375|1.151276E8|
27.27764|       APPLE|     0.08749962|
|2017-01-04 00:00:00|29.1275|28.9375|28.9625| 29.005| 8.44724E7|
27.247108|       APPLE|    0.042499542|
|2017-01-05 00:00:00| 29.215|28.9525|  28.98|29.1525| 8.87744E7|
27.385668|       APPLE|     0.17250061|
```

```
|2017-01-06 00:00:00|  29.54|29.1175| 29.195|29.4775|1.270076E8|
27.690971|       APPLE|     0.28250122|
|2017-01-09 00:00:00|29.8575| 29.485|29.4875|29.7475|1.342476E8|
27.944603|       APPLE|     0.26000023|
|2017-01-10 00:00:00| 29.845| 29.575|29.6925|29.7775| 9.78484E7|
27.972786|       APPLE|     0.08500099|
|2017-01-11 00:00:00|29.9825|  29.65| 29.685|29.9375|1.103544E8|
28.123089|       APPLE|     0.25250053|
|2017-01-12 00:00:00| 29.825|29.5525| 29.725|29.8125|1.083448E8|
28.005665|       APPLE|     0.08749962|
|2017-01-13 00:00:00| 29.905|29.7025|29.7775|  29.76|1.044476E8|
27.95635|       APPLE|   -0.017499924|
|2017-01-17 00:00:00|  30.06| 29.555| 29.585|   30.0|1.377592E8|
28.1818|       APPLE|     0.41500092|
|2017-01-18 00:00:00| 30.125|29.9275|   30.0|29.9975|  9.4852E7|
28.179457|       APPLE|   -0.002500534|
|2017-01-19 00:00:00|30.0225|29.8425|  29.85| 29.945|1.023892E8|
28.130133|       APPLE|     0.09499931|
|2017-01-20 00:00:00|30.1125|29.9325|30.1125|   30.0|1.303916E8|
28.1818|       APPLE|    -0.11249924|
|2017-01-23 00:00:00|30.2025|29.9425|   30.0|  30.02| 8.82008E7|
28.20059|       APPLE|    0.020000458|
|2017-01-24 00:00:00| 30.025| 29.875|29.8875|29.9925|  9.2844E7|
28.174757|       APPLE|     0.10499954|
|2017-01-25 00:00:00| 30.525|  30.07| 30.105|  30.47|1.295104E8|
28.623314|       APPLE|     0.36499977|
|2017-01-26 00:00:00|  30.61|   30.4|30.4175| 30.485|1.053504E8|
28.637407|       APPLE|     0.06750107|
|2017-01-27 00:00:00|30.5875|   30.4| 30.535|30.4875| 8.22516E7|
28.639755|       APPLE|    -0.04750061|
|2017-01-30 00:00:00|30.4075| 30.165|30.2325|30.4075|  1.2151E8|
28.564606|       APPLE|     0.17499924|
|2017-01-31 00:00:00|30.3475| 30.155|30.2875|30.3375| 1.96804E8|
28.498844|       APPLE|    0.049999237|
+------------------+------+------+------+------+----------
+---------+------------+---------------+
only showing top 20 rows

FACEBOOK
+------------------+------+------+------+------+---------+---------
+------------+---------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|evolution_price|
+------------------+------+------+------+------+---------+---------
+------------+---------------+
|2017-01-03 00:00:00|117.84|115.51|116.03|116.86|2.06639E7|   116.86|
FACEBOOK|     0.83000183|
|2017-01-04 00:00:00|119.66|117.29|117.55|118.69|1.96309E7|   118.69|
FACEBOOK|      1.1399994|
|2017-01-05 00:00:00|120.95|118.32|118.86|120.67|1.94922E7|   120.67|
```

```
FACEBOOK|      1.8099976|
|2017-01-06 00:00:00|123.88|120.03|120.98|123.41|2.85453E7|   123.41|
FACEBOOK|      2.4300003|
|2017-01-09 00:00:00|125.43|123.04|123.55| 124.9|2.28804E7|    124.9|
FACEBOOK|      1.3499985|
|2017-01-10 00:00:00| 125.5|124.28|124.82|124.35|1.73246E7|   124.35|
FACEBOOK|    -0.47000122|
|2017-01-11 00:00:00|126.12|124.06|124.35|126.09|1.83565E7|   126.09|
FACEBOOK|      1.7399979|
|2017-01-12 00:00:00|126.73| 124.8|125.61|126.62|1.86539E7|   126.62|
FACEBOOK|      1.0100021|
|2017-01-13 00:00:00|129.27|127.37|127.49|128.34|2.48843E7|   128.34|
FACEBOOK|      0.8499985|
|2017-01-17 00:00:00|128.34| 127.4|128.04|127.87|1.52945E7|   127.87|
FACEBOOK|    -0.16999054|
|2017-01-18 00:00:00|128.43|126.84|128.41|127.92|1.31459E7|   127.92|
FACEBOOK|     -0.4900055|
|2017-01-19 00:00:00|128.35|127.45|128.23|127.55|1.21955E7|   127.55|
FACEBOOK|     -0.6799927|
|2017-01-20 00:00:00|128.48|126.78| 128.1|127.04|1.90972E7|   127.04|
FACEBOOK|     -1.0600052|
|2017-01-23 00:00:00|129.25|126.95|127.31|128.93|1.65936E7|   128.93|
FACEBOOK|      1.6199951|
|2017-01-24 00:00:00| 129.9|128.38|129.38|129.37|1.51627E7|   129.37|
FACEBOOK|   -0.010009766|
|2017-01-25 00:00:00|131.74|129.77| 130.0|131.48|1.87313E7|   131.48|
FACEBOOK|      1.4799957|
|2017-01-26 00:00:00|133.14|131.44|131.63|132.78|2.00201E7|   132.78|
FACEBOOK|      1.1499939|
|2017-01-27 00:00:00|132.95|131.08|132.68|132.18|1.95395E7|   132.18|
FACEBOOK|           -0.5|
|2017-01-30 00:00:00|131.58| 129.6|131.58|130.98|1.89561E7|   130.98|
FACEBOOK|     -0.6000061|
|2017-01-31 00:00:00|130.66|129.52|130.17|130.32|1.97905E7|   130.32|
FACEBOOK|     0.15000916|
+-------------------+------+------+------+------+---------+---------
+------------+--------------+
only showing top 20 rows

GOOGLE
+-------------------+-------+-------+------+-------+---------
+---------+------------+--------------+
|               Date|   High|    Low|  Open|  Close|   Volume|Adj
Close|company_name|evolution_price|
+-------------------+-------+-------+------+-------+---------
+---------+------------+--------------+
|2017-01-03 00:00:00| 789.63|  775.8|778.81| 786.14|1657300.0|
786.14|      GOOGLE|       7.330017|
|2017-01-04 00:00:00| 791.34| 783.16|788.36|  786.9|1073000.0|
786.9|      GOOGLE|     -1.4599609|
```

```
|2017-01-05 00:00:00| 794.48| 785.02|786.08| 794.02|1335200.0|
794.02|      GOOGLE|      7.9400024|
|2017-01-06 00:00:00|  807.9|792.204|795.26| 806.15|1640200.0|
806.15|      GOOGLE|      10.890015|
|2017-01-09 00:00:00|809.966| 802.83| 806.4| 806.65|1274600.0|
806.65|      GOOGLE|           0.25|
|2017-01-10 00:00:00| 809.13| 803.51|807.86| 804.79|1176800.0|
804.79|      GOOGLE|     -3.0700073|
|2017-01-11 00:00:00| 808.15| 801.37| 805.0| 807.91|1065900.0|
807.91|      GOOGLE|      2.9099731|
|2017-01-12 00:00:00| 807.39| 799.17|807.14| 806.36|1353100.0|
806.36|      GOOGLE|     -0.7800293|
|2017-01-13 00:00:00|811.224| 806.69|807.48| 807.88|1099200.0|
807.88|      GOOGLE|      0.4000244|
|2017-01-17 00:00:00| 807.14| 800.37|807.08| 804.61|1362100.0|
804.61|      GOOGLE|     -2.4700317|
|2017-01-18 00:00:00|806.205| 800.99|805.81| 806.07|1294400.0|
806.07|      GOOGLE|     0.26000977|
|2017-01-19 00:00:00| 809.48|  801.8|805.12|802.175| 919300.0|
802.175|      GOOGLE|     -2.9450073|
|2017-01-20 00:00:00| 806.91| 801.69|806.91| 805.02|1670000.0|
805.02|      GOOGLE|     -1.8899536|
|2017-01-23 00:00:00| 820.87| 803.74|807.25| 819.31|1963600.0|
819.31|      GOOGLE|      12.059998|
|2017-01-24 00:00:00|  825.9|817.821| 822.3| 823.87|1474000.0|
823.87|      GOOGLE|      1.5700073|
|2017-01-25 00:00:00| 835.77| 825.06|829.62| 835.67|1494500.0|
835.67|      GOOGLE|       6.049988|
|2017-01-26 00:00:00|  838.0| 827.01|837.81| 832.15|2973900.0|
832.15|      GOOGLE|      -5.659973|
|2017-01-27 00:00:00| 841.95| 820.44|834.71| 823.31|2965800.0|
823.31|      GOOGLE|     -11.400024|
|2017-01-30 00:00:00| 815.84|  799.8|814.66| 802.32|3246600.0|
802.32|      GOOGLE|     -12.339966|
|2017-01-31 00:00:00| 801.25| 790.52|796.86| 796.79|2160600.0|
796.79|      GOOGLE|   -0.070007324|
+-------------------+-------+-------+------+-------+---------
+---------+------------+--------------+
only showing top 20 rows

MICROSOFT
+-------------------+-----+-----+-----+-----+---------+---------
+------------+--------------+
|               Date| High|  Low| Open|Close|   Volume|Adj Close|
company_name|evolution_price|
+-------------------+-----+-----+-----+-----+---------+---------
+------------+--------------+
|2017-01-03 00:00:00|62.84|62.13|62.79|62.58|2.06941E7|58.673244|
MICROSOFT|    -0.20999908|
|2017-01-04 00:00:00|62.75|62.12|62.48| 62.3|  2.134E7|58.410725|
```

```
MICROSOFT|     -0.1800003|
|2017-01-05 00:00:00|62.66|62.03|62.19| 62.3| 2.4876E7|58.410725|
MICROSOFT|     0.11000061|
|2017-01-06 00:00:00|63.15|62.04| 62.3|62.84|1.99229E7|58.917015|
MICROSOFT|      0.5400009|
|2017-01-09 00:00:00|63.08|62.54|62.76|62.64|2.03827E7|58.729496|
MICROSOFT|    -0.11999893|
|2017-01-10 00:00:00|63.07|62.28|62.73|62.62| 1.8593E7|58.710747|
MICROSOFT|    -0.11000061|
|2017-01-11 00:00:00|63.23|62.43|62.61|63.19|2.15173E7| 59.24516|
MICROSOFT|       0.579998|
|2017-01-12 00:00:00| 63.4|61.95|63.06|62.61|2.09682E7| 58.70137|
MICROSOFT|    -0.45000076|
|2017-01-13 00:00:00|62.87|62.35|62.62| 62.7|1.94223E7|58.785755|
MICROSOFT|     0.08000183|
|2017-01-17 00:00:00| 62.7|62.03|62.68|62.53| 2.0664E7| 58.62637|
MICROSOFT|    -0.15000153|
|2017-01-18 00:00:00| 62.7|62.12|62.67| 62.5|1.96701E7|58.598248|
MICROSOFT|    -0.16999817|
|2017-01-19 00:00:00|62.98| 62.2|62.24| 62.3|1.84517E7|58.410725|
MICROSOFT|     0.05999756|
|2017-01-20 00:00:00|62.82|62.37|62.67|62.74|3.02135E7| 58.82326|
MICROSOFT|     0.07000351|
|2017-01-23 00:00:00|63.12|62.57| 62.7|62.96|2.30976E7|59.029526|
MICROSOFT|     0.25999832|
|2017-01-24 00:00:00|63.74|62.94| 63.2|63.52|2.46729E7| 59.55457|
MICROSOFT|      0.3199997|
|2017-01-25 00:00:00| 64.1|63.45|63.95|63.68|2.36727E7|59.704575|
MICROSOFT|    -0.27000046|
|2017-01-26 00:00:00|64.54|63.55|64.12|64.27|4.35546E7|60.257736|
MICROSOFT|      0.1499939|
|2017-01-27 00:00:00|65.91|64.89|65.39|65.78| 4.4818E7| 61.67348|
MICROSOFT|      0.3899994|
|2017-01-30 00:00:00|65.79| 64.8|65.69|65.13|3.16514E7|61.064045|
MICROSOFT|     -0.5600052|
|2017-01-31 00:00:00|65.15|64.26|64.86|64.65|2.52705E7| 60.61402|
MICROSOFT|    -0.20999908|
+------------------+-----+-----+-----+-----+---------+---------
+------------+---------------+
only showing top 20 rows

TESLA
+------------------+------+------+------+------+---------+---------
+------------+---------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|evolution_price|
+------------------+------+------+------+------+---------+---------
+------------+---------------+
|2017-01-03 00:00:00|44.066|42.192|42.972|43.398|2.96165E7|   43.398|
TESLA|      0.4259987|
```

```
|2017-01-04 00:00:00|  45.6|42.862| 42.95|45.398|5.60675E7|   45.398|
TESLA|       2.447998|
|2017-01-05 00:00:00|45.496| 44.39|45.284| 45.35|2.95585E7|    45.35|
TESLA|     0.06599808|
|2017-01-06 00:00:00|46.062| 45.09|45.386|45.802|2.76395E7|   45.802|
TESLA|     0.41599655|
|2017-01-09 00:00:00|46.384|  45.6|45.794|46.256|1.98975E7|   46.256|
TESLA|      0.4620018|
|2017-01-10 00:00:00|  46.4|45.378|  46.4|45.974|   1.83E7|   45.974|
TESLA|     -0.4260025|
|2017-01-11 00:00:00|45.996|45.336|45.814|45.946| 1.8254E7|   45.946|
TESLA|     0.13199997|
|2017-01-12 00:00:00| 46.14|45.116|45.812|45.918| 1.8951E7|   45.918|
TESLA|     0.10599899|
|2017-01-13 00:00:00| 47.57|45.918|  46.0| 47.55| 3.0465E7|    47.55|
TESLA|      1.5499992|
|2017-01-17 00:00:00|47.992|46.874| 47.34|47.116|2.30875E7|   47.116|
TESLA|    -0.22399902|
|2017-01-18 00:00:00|47.942|47.116| 47.33|47.672| 1.8845E7|   47.672|
TESLA|     0.34199905|
|2017-01-19 00:00:00|49.736| 48.15| 49.45|48.752|3.86615E7|   48.752|
TESLA|    -0.69800186|
|2017-01-20 00:00:00|  49.2|48.602|49.092|48.946|2.10215E7|   48.946|
TESLA|    -0.14599991|
|2017-01-23 00:00:00|50.178|  49.1| 49.17|49.784|3.13145E7|   49.784|
TESLA|      0.6140022|
|2017-01-24 00:00:00| 50.96| 49.93|  50.0|50.922|2.48275E7|   50.922|
TESLA|      0.9220009|
|2017-01-25 00:00:00|51.692| 50.36|51.462|50.894| 2.5713E7|   50.894|
TESLA|     -0.5680008|
|2017-01-26 00:00:00|51.148| 50.15|50.858|50.502|1.57605E7|   50.502|
TESLA|     -0.3560028|
|2017-01-27 00:00:00|  50.6|49.704|50.276| 50.59|1.58315E7|    50.59|
TESLA|     0.31399918|
|2017-01-30 00:00:00|51.058| 49.42|50.506|50.126|1.90055E7|   50.126|
TESLA|    -0.38000107|
|2017-01-31 00:00:00|51.178| 49.54|49.848|50.386|2.05805E7|   50.386|
TESLA|       0.538002|
+-------------------+------+------+------+------+---------+---------
+------------+---------------+
only showing top 20 rows

ZOOM

+-------------------+------+------+-----+-----+---------+---------
+------------+---------------+
|               Date|  High|   Low| Open|Close|   Volume|Adj Close|
company_name|evolution_price|
+-------------------+------+------+-----+-----+---------+---------
+------------+---------------+
```

```
|2019-04-18 00:00:00|  66.0|60.321| 65.0| 62.0|2.57647E7|     62.0|
ZOOM|           -3.0|
|2019-04-22 00:00:00|  68.9| 59.94| 61.0| 65.7|9949700.0|     65.7|
ZOOM|       4.699997|
|2019-04-23 00:00:00|74.169| 65.55|66.87| 69.0|6786500.0|     69.0|
ZOOM|      2.1299973|
|2019-04-24 00:00:00|  71.5| 63.16| 71.4| 63.2|4973500.0|     63.2|
ZOOM|      -8.200001|
|2019-04-25 00:00:00| 66.85|  62.6|64.74| 65.0|3863300.0|     65.0|
ZOOM|     0.26000214|
|2019-04-26 00:00:00| 66.99|  63.6|66.12|66.22|1527400.0|    66.22|
ZOOM|    0.099998474|
|2019-04-29 00:00:00|  68.5| 64.75|66.53|68.17|1822300.0|    68.17|
ZOOM|      1.6399994|
|2019-04-30 00:00:00| 72.52| 66.67| 68.4|72.47|4113100.0|    72.47|
ZOOM|      4.0699997|
|2019-05-01 00:00:00| 76.95|70.816|72.72|72.76|3301900.0|    72.76|
ZOOM|    0.040000916|
|2019-05-02 00:00:00| 75.89|69.691|72.75| 75.5|2525300.0|     75.5|
ZOOM|           2.75|
|2019-05-03 00:00:00| 80.25|  75.0| 75.0|79.18|2590300.0|    79.18|
ZOOM|      4.1800003|
|2019-05-06 00:00:00| 80.79|  74.5|75.01|78.24|2051800.0|    78.24|
ZOOM|      3.2299957|
|2019-05-07 00:00:00| 78.05| 73.25|77.85|73.33|1975200.0|    73.33|
ZOOM|     -4.5199966|
|2019-05-08 00:00:00|  78.5| 74.03|74.61|77.68|2265500.0|    77.68|
ZOOM|      3.0699997|
|2019-05-09 00:00:00| 76.99|  74.0|76.85|75.21|1348200.0|    75.21|
ZOOM|     -1.6399994|
|2019-05-10 00:00:00| 79.74| 74.77|75.79|79.63|1555100.0|    79.63|
ZOOM|      3.8399963|
|2019-05-13 00:00:00| 77.39|  70.6|77.39|72.54|2873200.0|    72.54|
ZOOM|     -4.8499985|
|2019-05-14 00:00:00|76.885| 73.11|74.12|73.14|1950400.0|    73.14|
ZOOM|    -0.98000336|
|2019-05-15 00:00:00|  80.0| 72.21| 73.4|79.76|2426500.0|    79.76|
ZOOM|      6.3600006|
|2019-05-16 00:00:00| 87.55| 79.25|80.12| 83.4|4580700.0|     83.4|
ZOOM|      3.2799988|
+-------------------+------+------+-----+-----+---------+---------
+------------+---------------+
only showing top 20 rows
```

- What are the stocks with the highest daily return

```
def highest_daily_return():
    """
        :return : the highest daily return
    """
```

```
    maxi = 0
    for df in LIST_DF:
        tmp = df.agg(max_(df['Close'] - df['Open'])).collect()[0][0]
        name_tmp = df.first()['company_name']
        if (tmp > maxi):
            maxi = tmp
            name = name_tmp

    print("Highest daily return is :",maxi ,"found in", name)

highest_daily_return()
```

```
Highest daily return is : 196.64013671875 found in AMAZON
```

- Calculate the average daily return for different periods (week, month, and year)

```
def avg_daily_return(df, w=False, m=False, y=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise
            - y : Boolean True if we want a year period, False
otherwise

        :return Average daily return calculated following the perdio
choose
    """
    list = []
    if w:
        dates = df.select(weekofyear(df['Date'])).collect()
        for i in range(len(dates)):
            if dates[i][0] not in list:
                list.append(dates[i][0])
    else:
        dates = df.select('Date').collect() #convert column to list
        for date in dates:
            if y:
                if date[0].year not in list:
                    list.append(date[0].year)
            else:
                if date[0].month not in list:
                    list.append(date[0].month)

    list.sort()

    for i in list:
        if y: df_filtered = df.filter(year(df['Date']) == i)
        elif m: df_filtered = df.filter(month(df['Date']) == i)
```

```
        else: df_filtered = df.filter(weekofyear(df['Date']) == i)
        print("Average daily return in", i)
        df_filtered.agg(mean(df_filtered['Close']-
df_filtered['Open'])).show()

for df in LIST_DF:
    print("Average daily return for each year of", df.first()
['company_name'])
    avg_daily_return(df_amazon, y=True)
```

```
Average daily return for each year of AMAZON
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+--------------------+
|avg((Close - Open))|
+--------------------+
|-2.3465333003921813|
+--------------------+

Average daily return in 2019
+--------------------+
|avg((Close - Open))|
+--------------------+
|0.44301641555059523|
+--------------------+

Average daily return in 2020
+--------------------+
|avg((Close - Open))|
+--------------------+
| 0.1441503696687232|
+--------------------+

Average daily return for each year of APPLE
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+--------------------+
|avg((Close - Open))|
```

```
+-------------------+
|-2.3465333003921813|
+-------------------+

Average daily return in 2019
+-------------------+
|avg((Close - Open))|
+-------------------+
|0.44301641555059523|
+-------------------+

Average daily return in 2020
+-------------------+
|avg((Close - Open))|
+-------------------+
| 0.1441503696687232|
+-------------------+

Average daily return for each year of FACEBOOK
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+-------------------+
|avg((Close - Open))|
+-------------------+
|-2.3465333003921813|
+-------------------+

Average daily return in 2019
+-------------------+
|avg((Close - Open))|
+-------------------+
|0.44301641555059523|
+-------------------+

Average daily return in 2020
+-------------------+
|avg((Close - Open))|
+-------------------+
| 0.1441503696687232|
+-------------------+

Average daily return for each year of GOOGLE
Average daily return in 2017
```

```
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+--------------------+
|avg((Close - Open))|
+--------------------+
|-2.3465333003921813|
+--------------------+

Average daily return in 2019
+--------------------+
|avg((Close - Open))|
+--------------------+
|0.44301641555059523|
+--------------------+

Average daily return in 2020
+--------------------+
|avg((Close - Open))|
+--------------------+
| 0.1441503696687232|
+--------------------+

Average daily return for each year of MICROSOFT
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+--------------------+
|avg((Close - Open))|
+--------------------+
|-2.3465333003921813|
+--------------------+

Average daily return in 2019
+--------------------+
|avg((Close - Open))|
+--------------------+
|0.44301641555059523|
+--------------------+
```

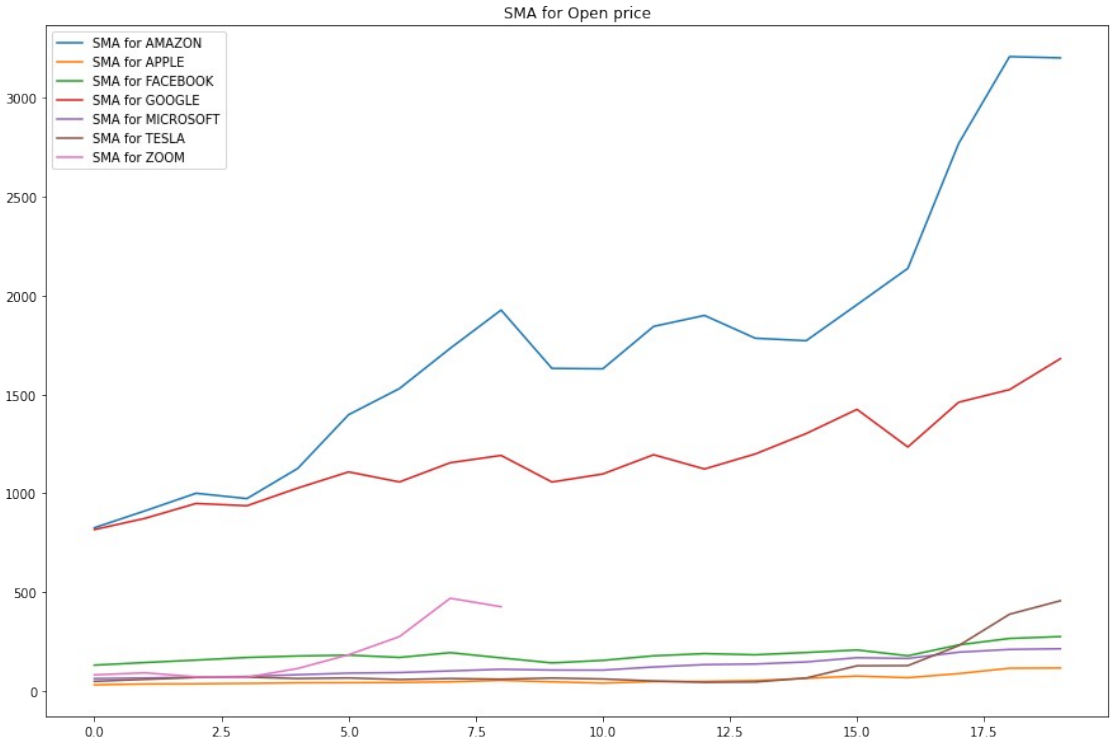
```
Average daily return in 2020
+-------------------+
|avg((Close - Open))|
+-------------------+
| 0.1441503696687232|
+-------------------+

Average daily return for each year of TESLA
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+-------------------+
|avg((Close - Open))|
+-------------------+
|-2.3465333003921813|
+-------------------+

Average daily return in 2019
+-------------------+
|avg((Close - Open))|
+-------------------+
|0.44301641555059523|
+-------------------+

Average daily return in 2020
+-------------------+
|avg((Close - Open))|
+-------------------+
| 0.1441503696687232|
+-------------------+

Average daily return for each year of ZOOM
Average daily return in 2017
+--------------------+
| avg((Close - Open))|
+--------------------+
|-0.10860731877178785|
+--------------------+

Average daily return in 2018
+-------------------+
|avg((Close - Open))|
+-------------------+
|-2.3465333003921813|
+-------------------+
```

```
Average daily return in 2019
+-------------------+
|avg((Close - Open))|
+-------------------+
|0.44301641555059523|
+-------------------+

Average daily return in 2020
+-------------------+
|avg((Close - Open))|
+-------------------+
| 0.1441503696687232|
+-------------------+
```

Moving Average

Fonction qui calcule le moving average à selon une certaine periode

```
#date_begin format : yyyy-mm-dd
def moving_average(df, column_name, period):
    """
        :param
            - df : the DataFrame
            - column_name : specific price
            - period : Number of days for each period
    """
    list_SMA = []
    for i in range (0, df.count(), period):
        df_filtered = spark.createDataFrame(df.collect()[i:
(i+period)])
        sma =
df_filtered.agg(mean(df_filtered[column_name])).collect()[0][0]
        list_SMA.append(sma)
    return list_SMA

def visualization_moving_average(column_name, period):
    figure, axes = plt.subplots(1, figsize=(15,10))
    for dfs in LIST_DF:
        sma = moving_average(dfs, column_name, period)
        label = "SMA for "+ dfs.first()['company_name']
        axes.plot(sma, label=label)
        title = "SMA for "+ column_name + " price"
        axes.set_title(title)
        axes.legend()
    plt.show()

visualization_moving_average('Open', 50)
```



Correlation between stocks

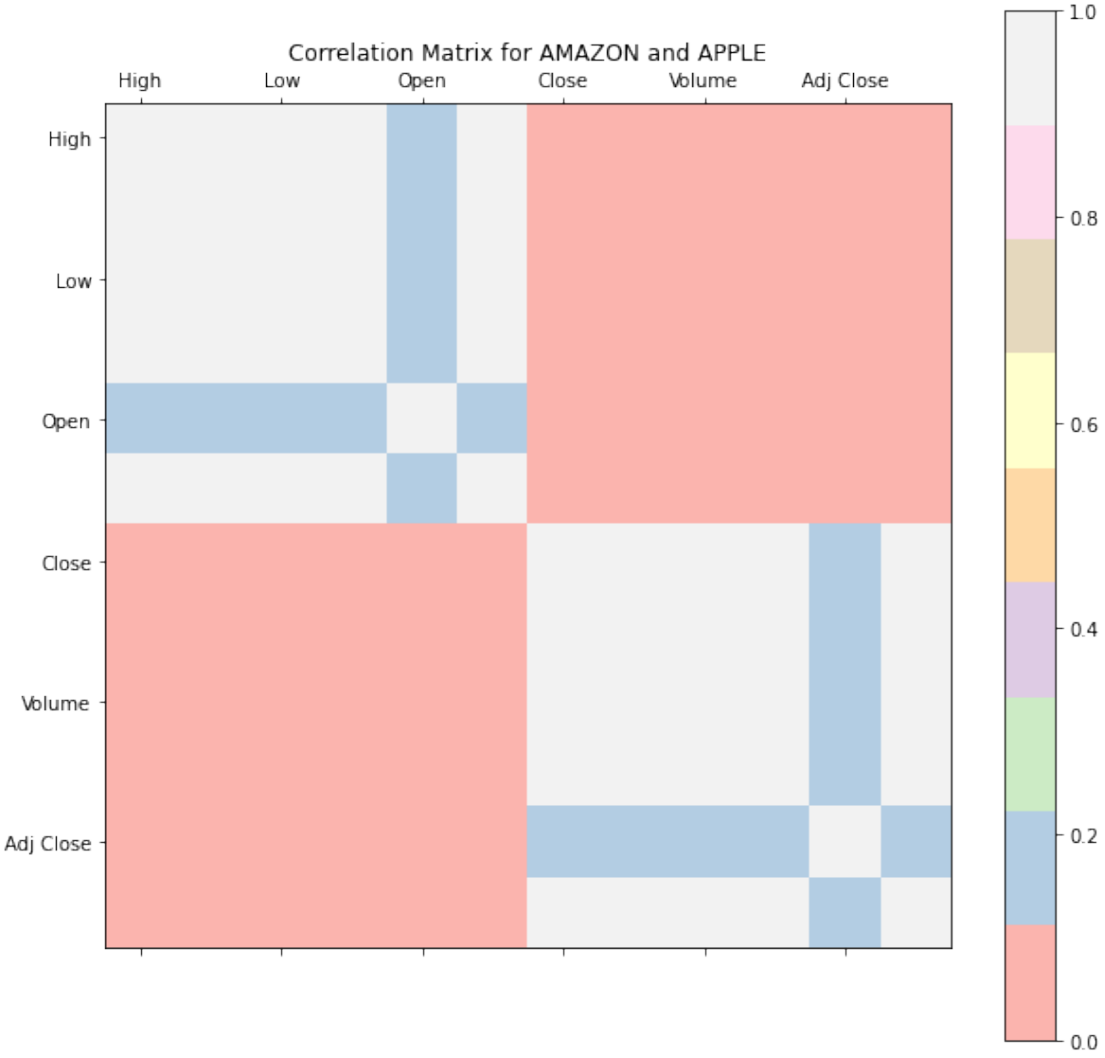
On créé une première fonction qui va merge 2 dataframes pour avoir les colonnes numériques de 2 stocks. Par la suite on créé la matrice de corrélation de ce nouveau dataframe en utilisant la fonction faite précédemment

```
def merge(df1, df2):
    """
        :param :
            - df1 : first dataframe
            - df2 : second dataframe

        :return : join the two dataframe with only numerics values
    """
    stocks1 = df1.drop('Date', 'company_name')
    stocks2 = df2.drop('Date', 'company_name')
    stocks2_rename = stocks2.toDF(*(c+"_2" for c in stocks2.columns))
    return stocks1.join(stocks2_rename)
```

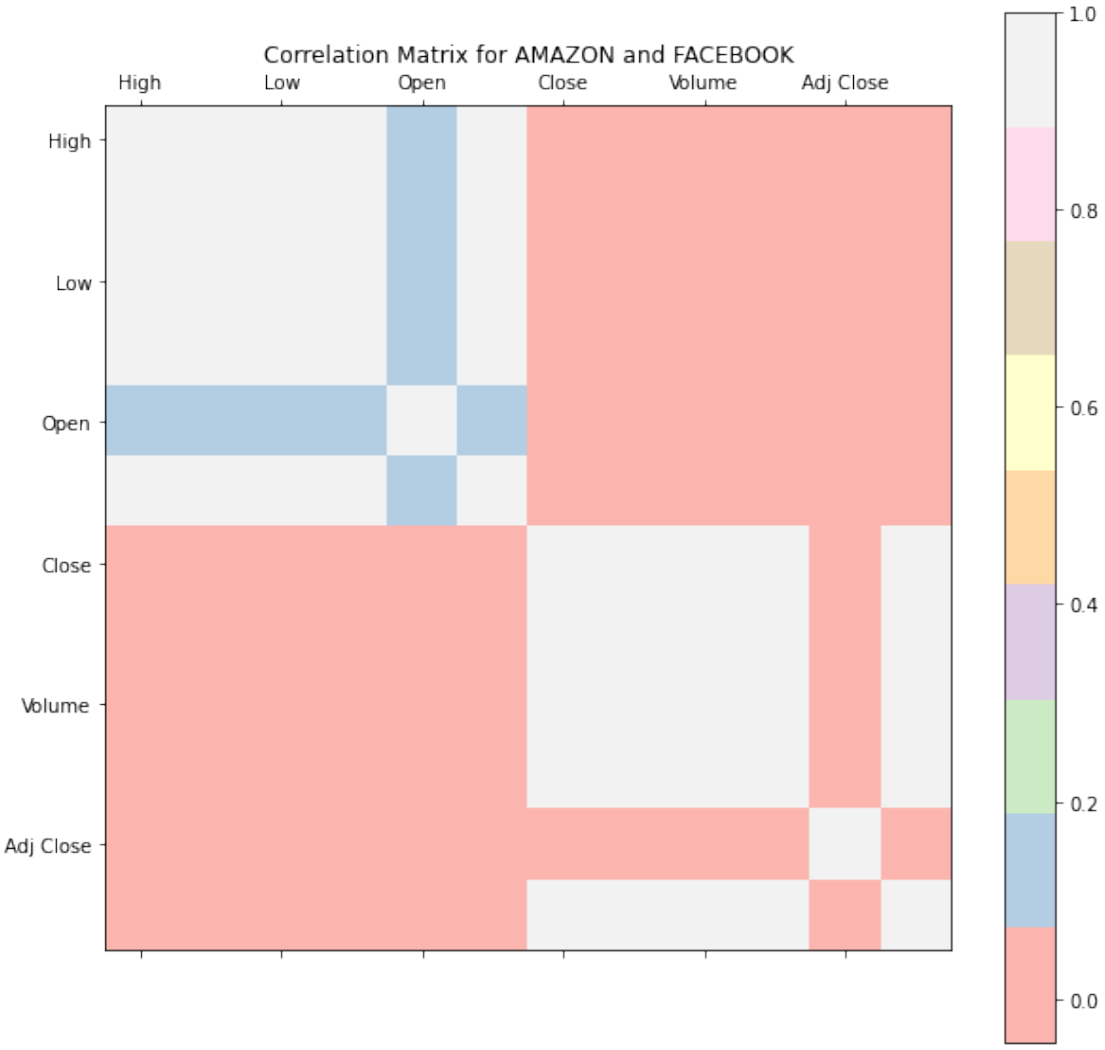
Nous allons tester cette fonction sur toutes les combinaisons possibles de nos différents dataframes.

```
for (df1,df2) in itertools.combinations(LIST_DF, 2):
    df_merged = merge(df1,df2)
    mat = get_corr_matrix(df_merged)
    final_name = df1.first()['company_name'] + " and " + df2.first()
['company_name']
    visualization_corr_matrix(mat,df_merged.columns, final_name)
```

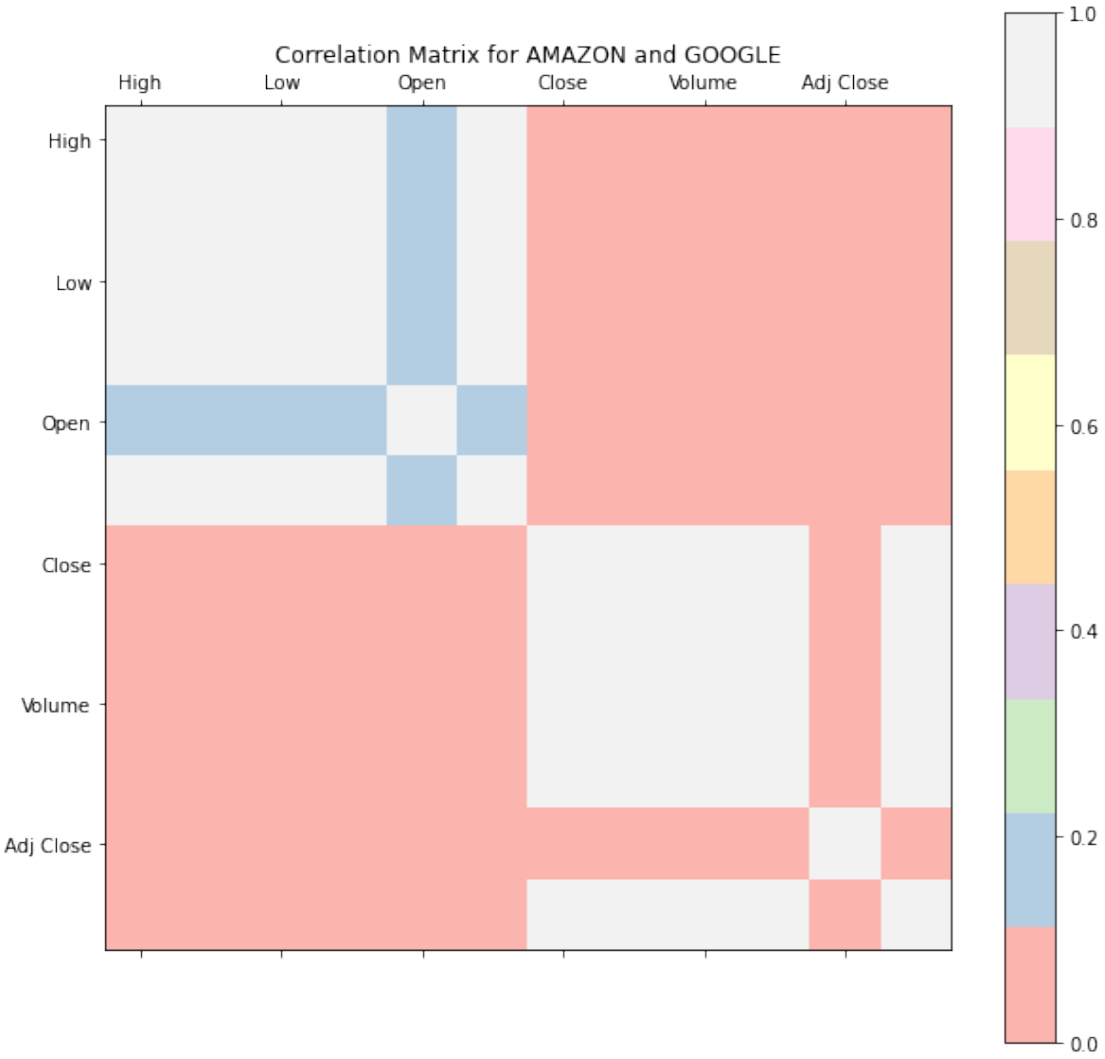
```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```



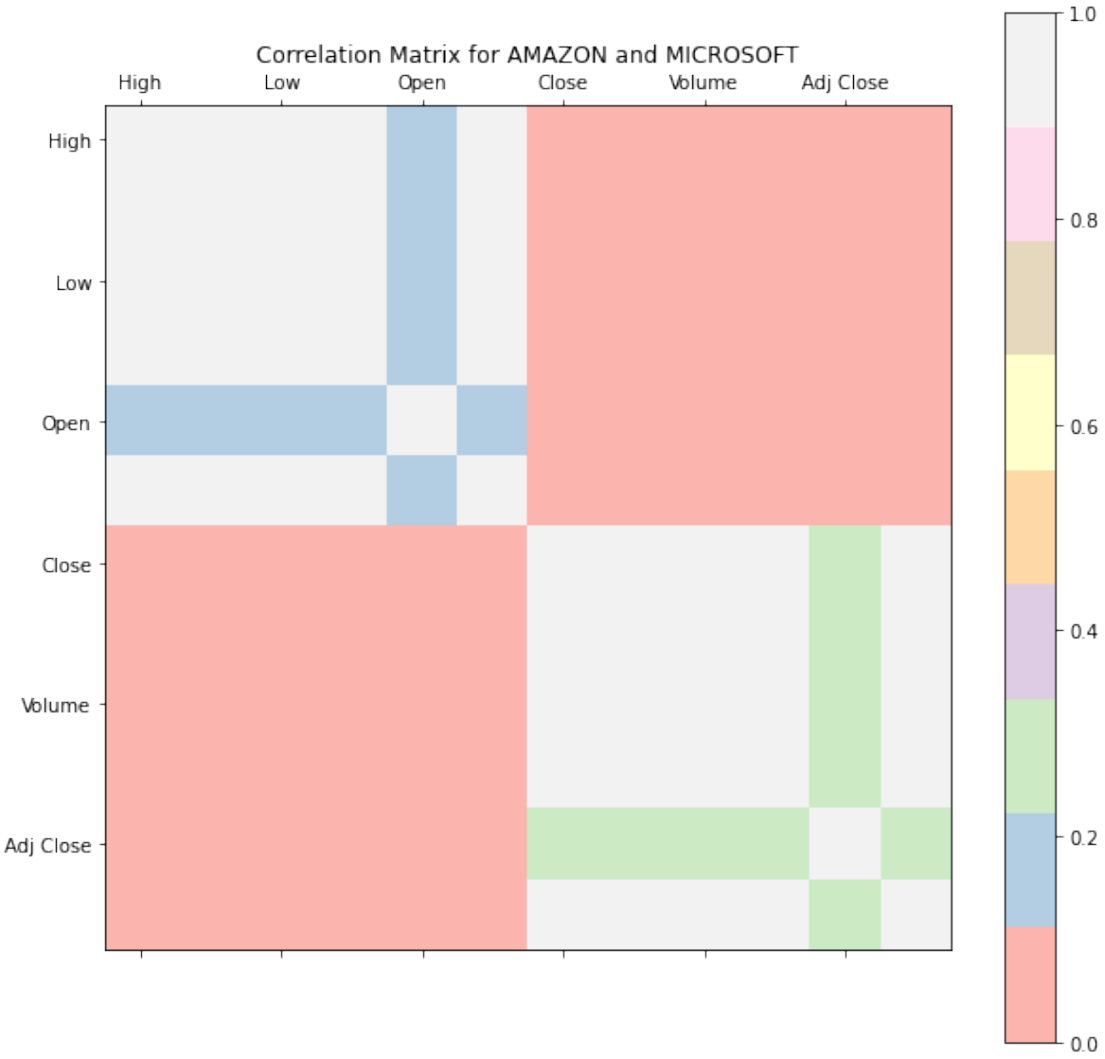

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



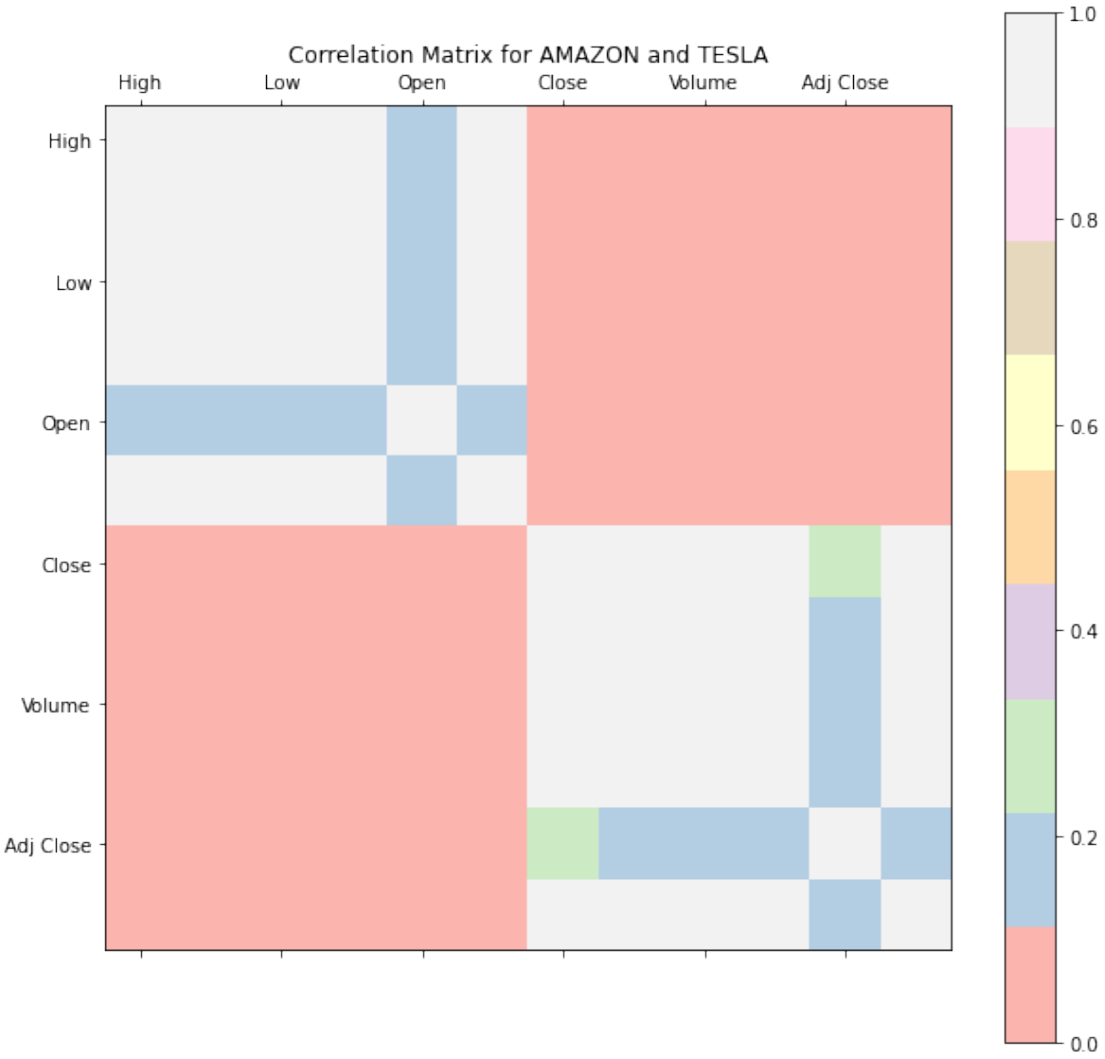

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



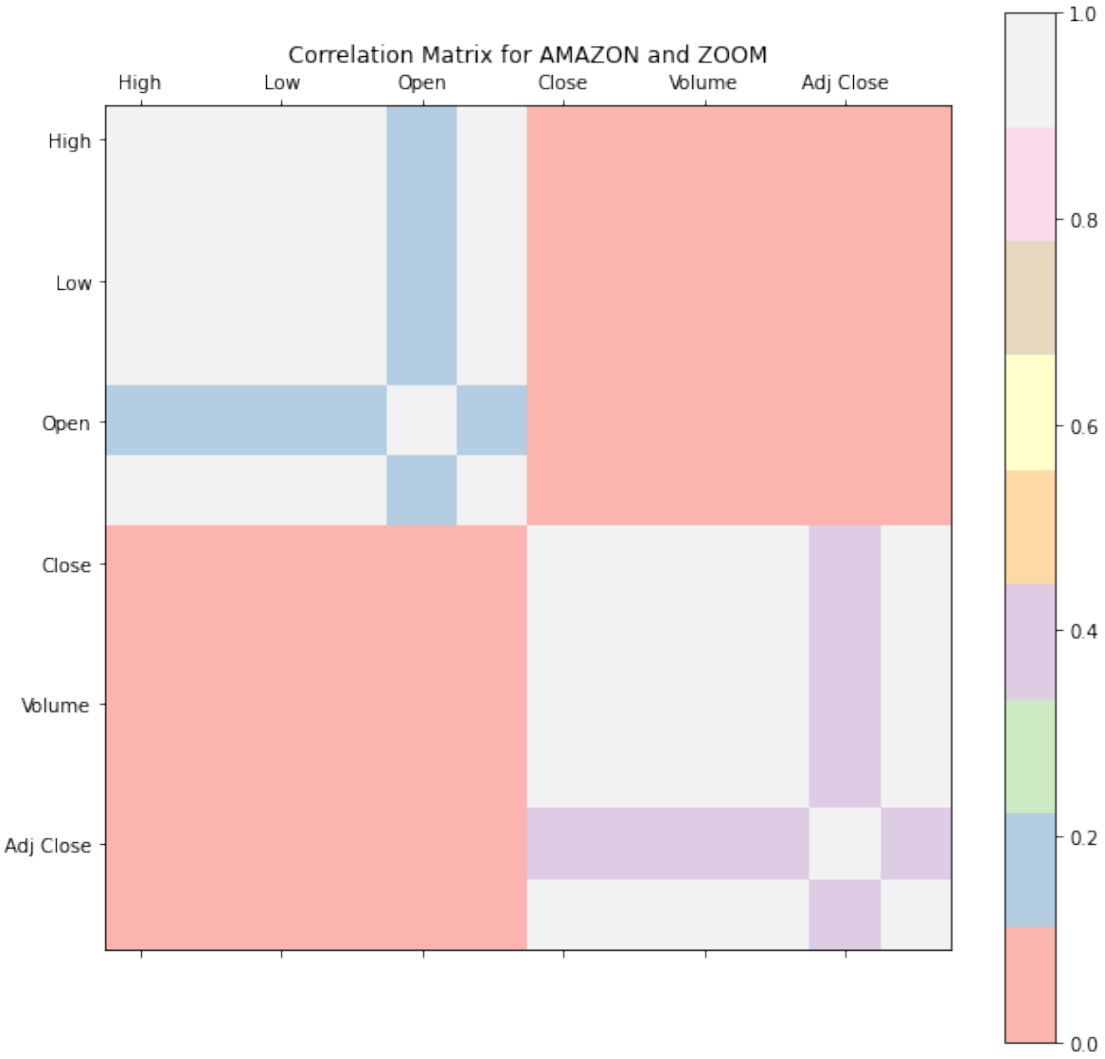

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



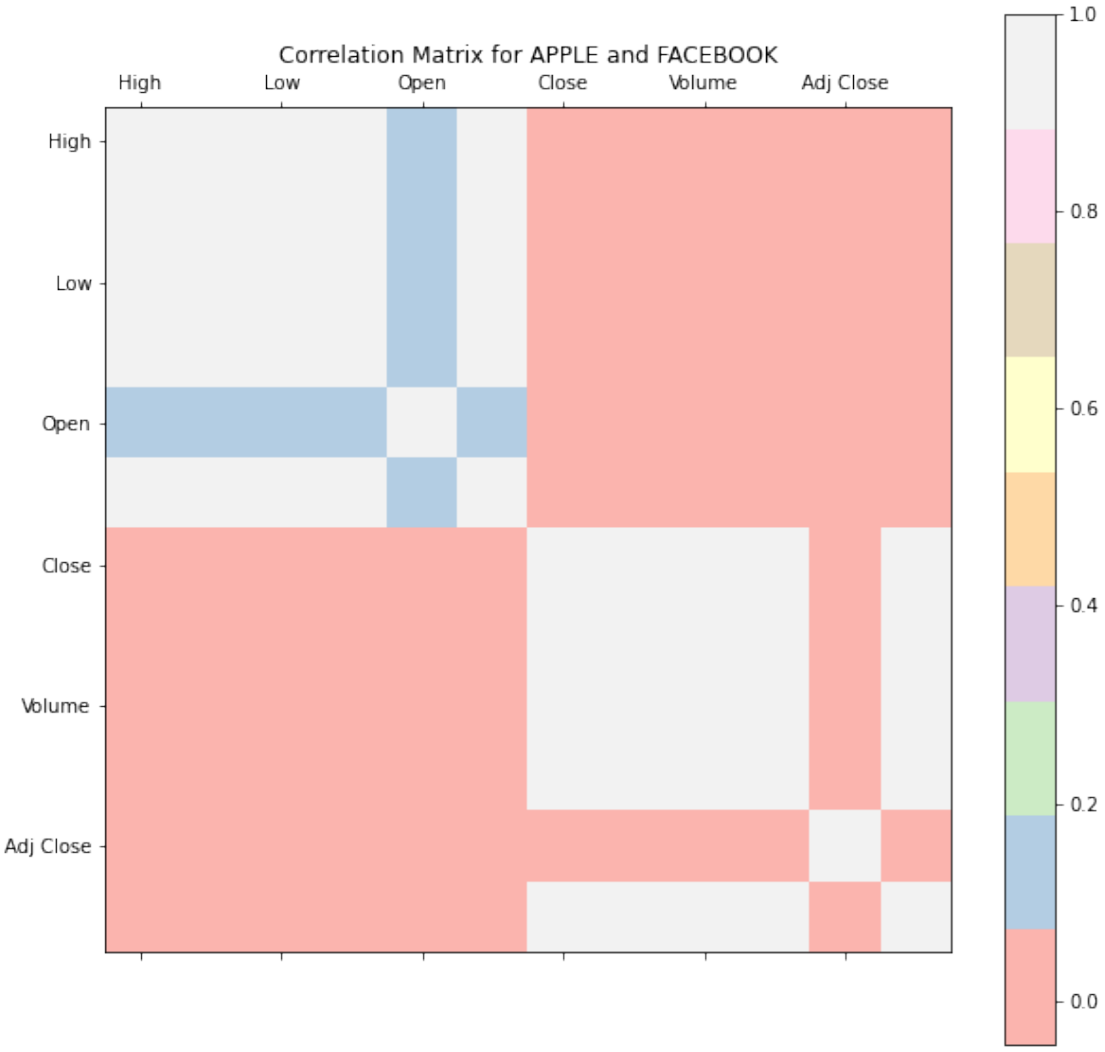

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



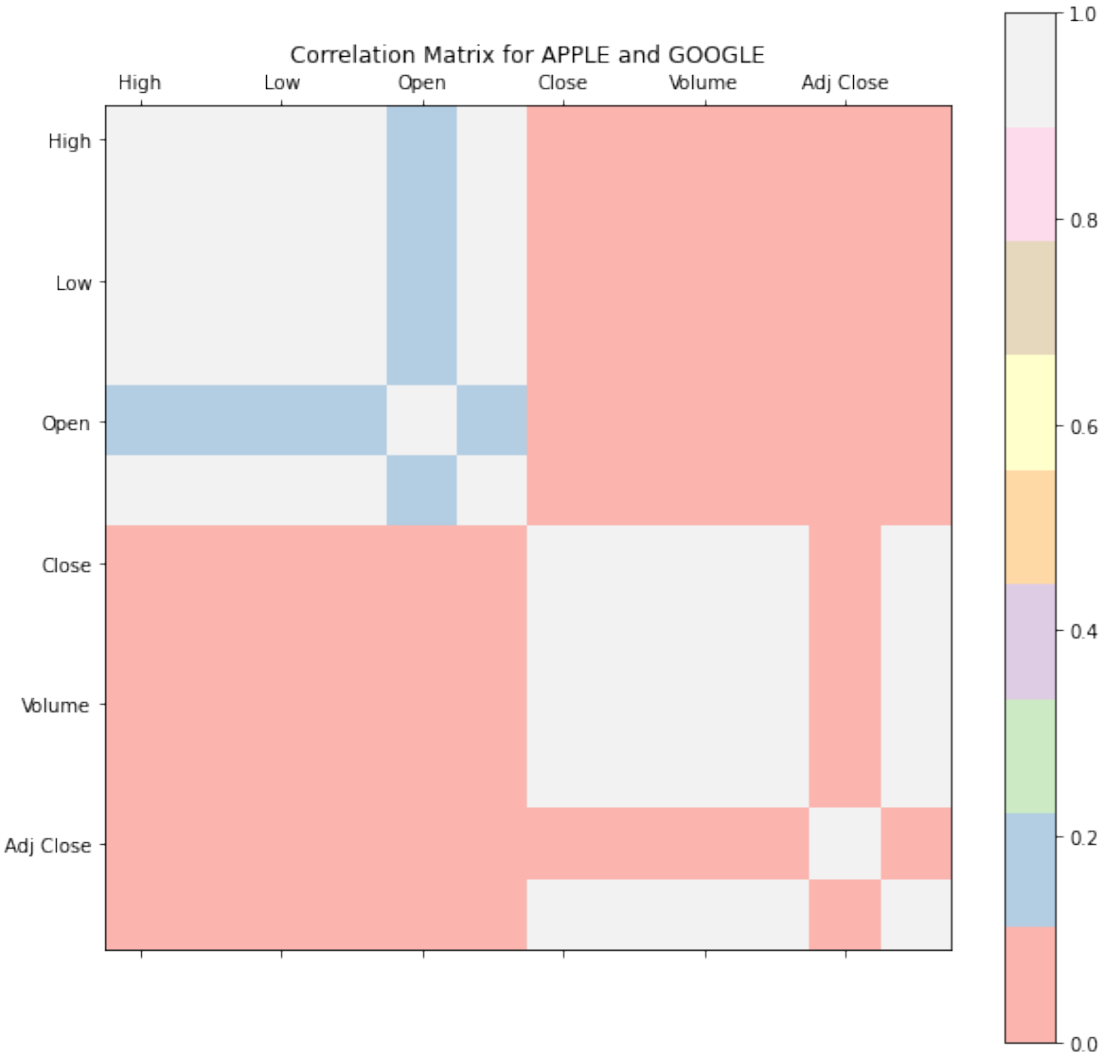

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



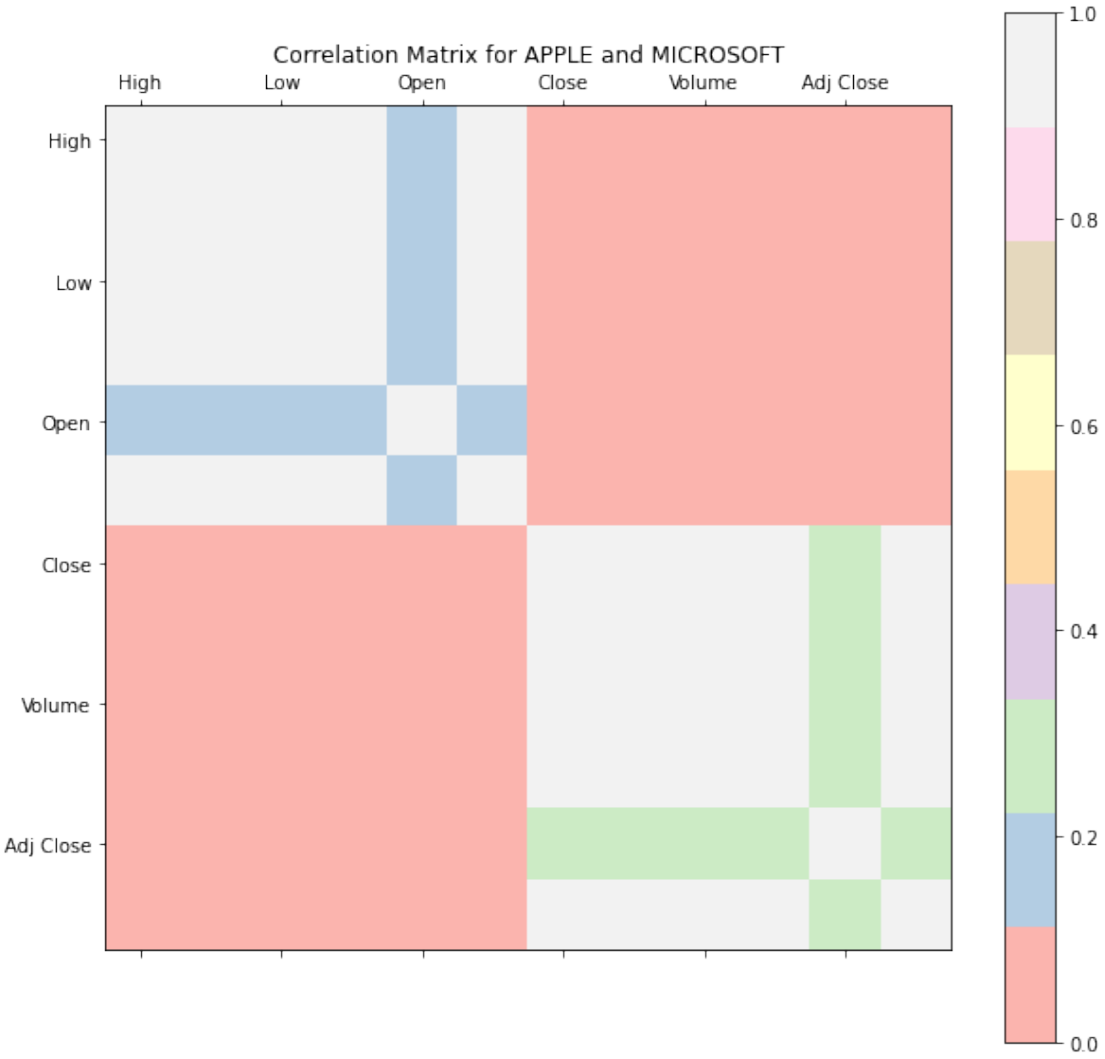

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



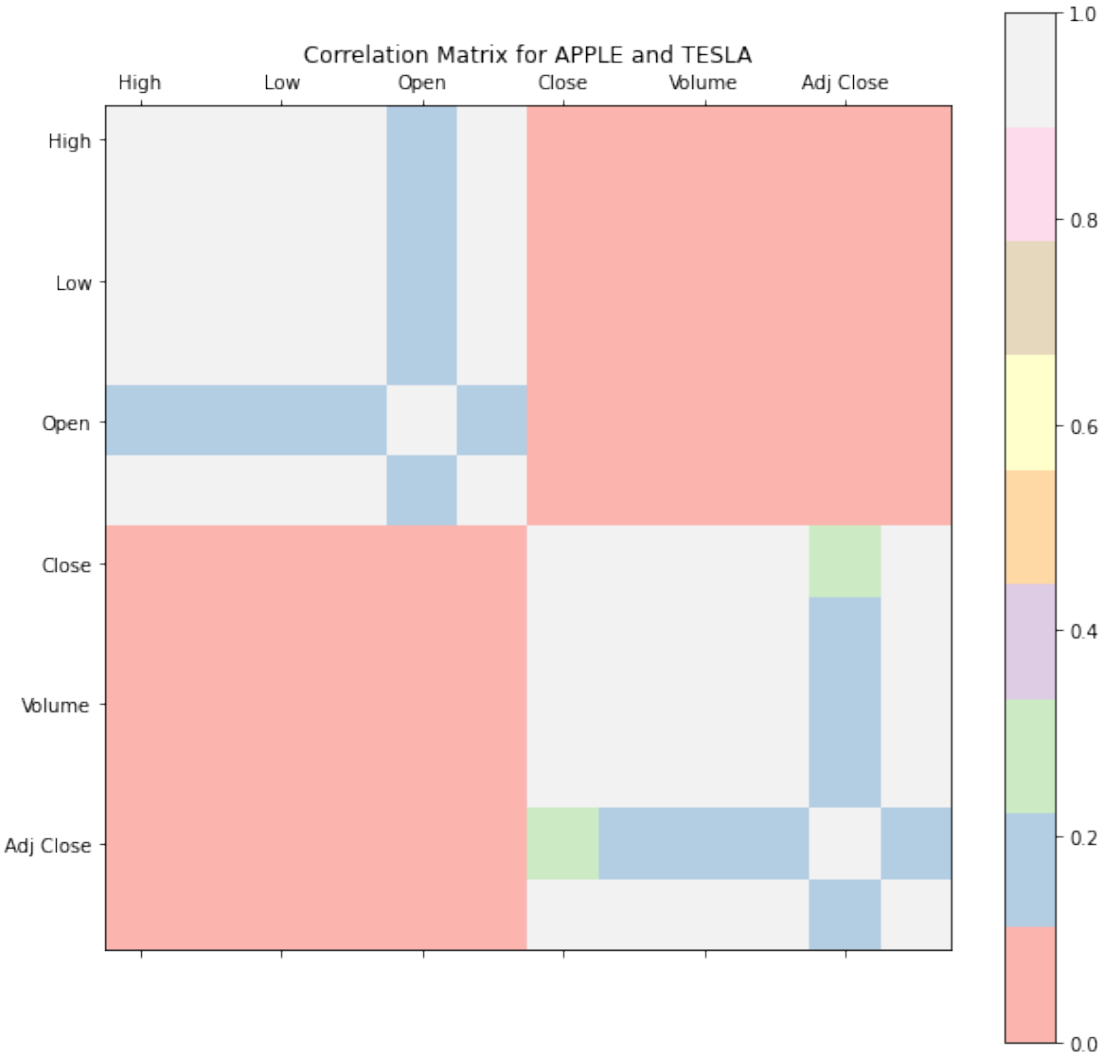

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



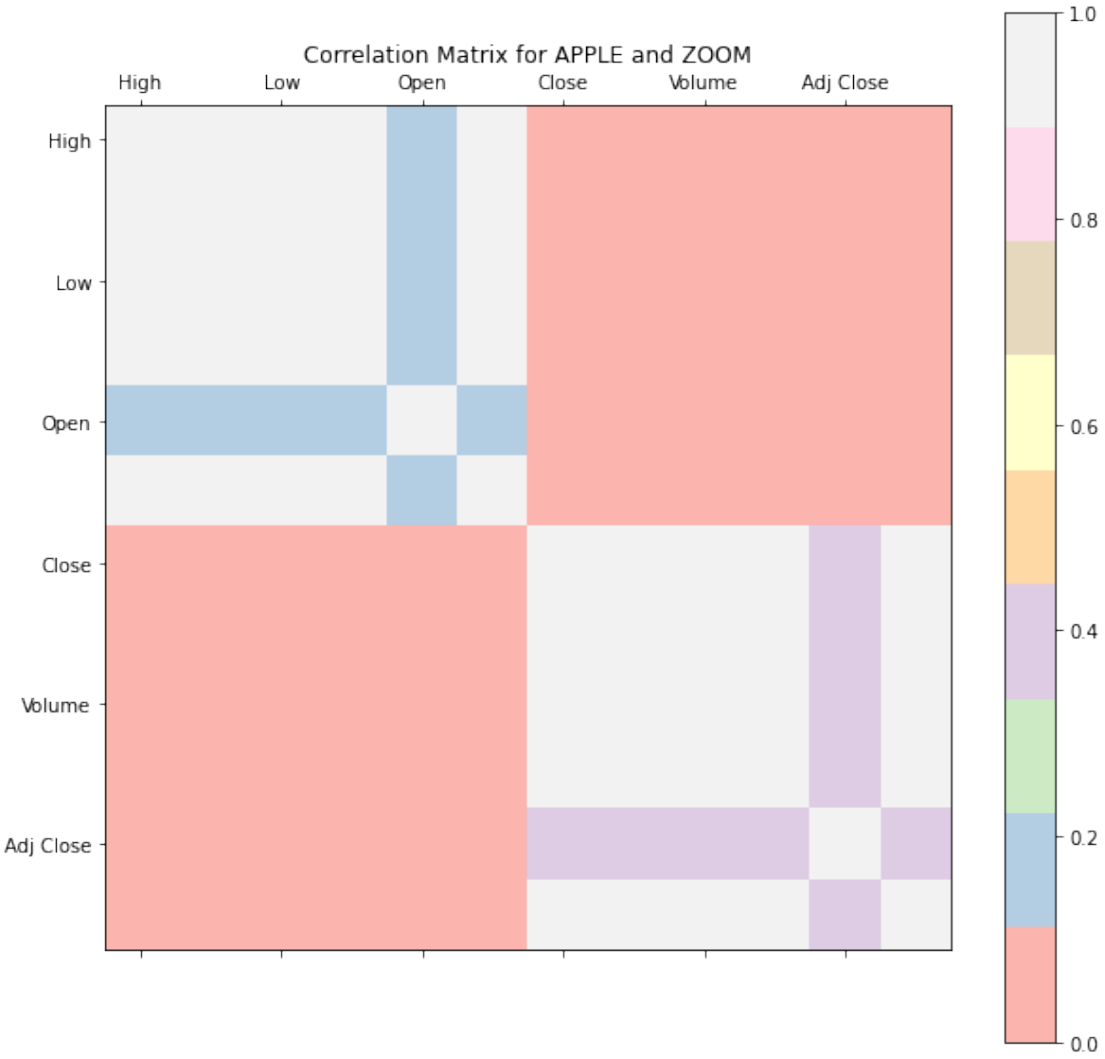

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



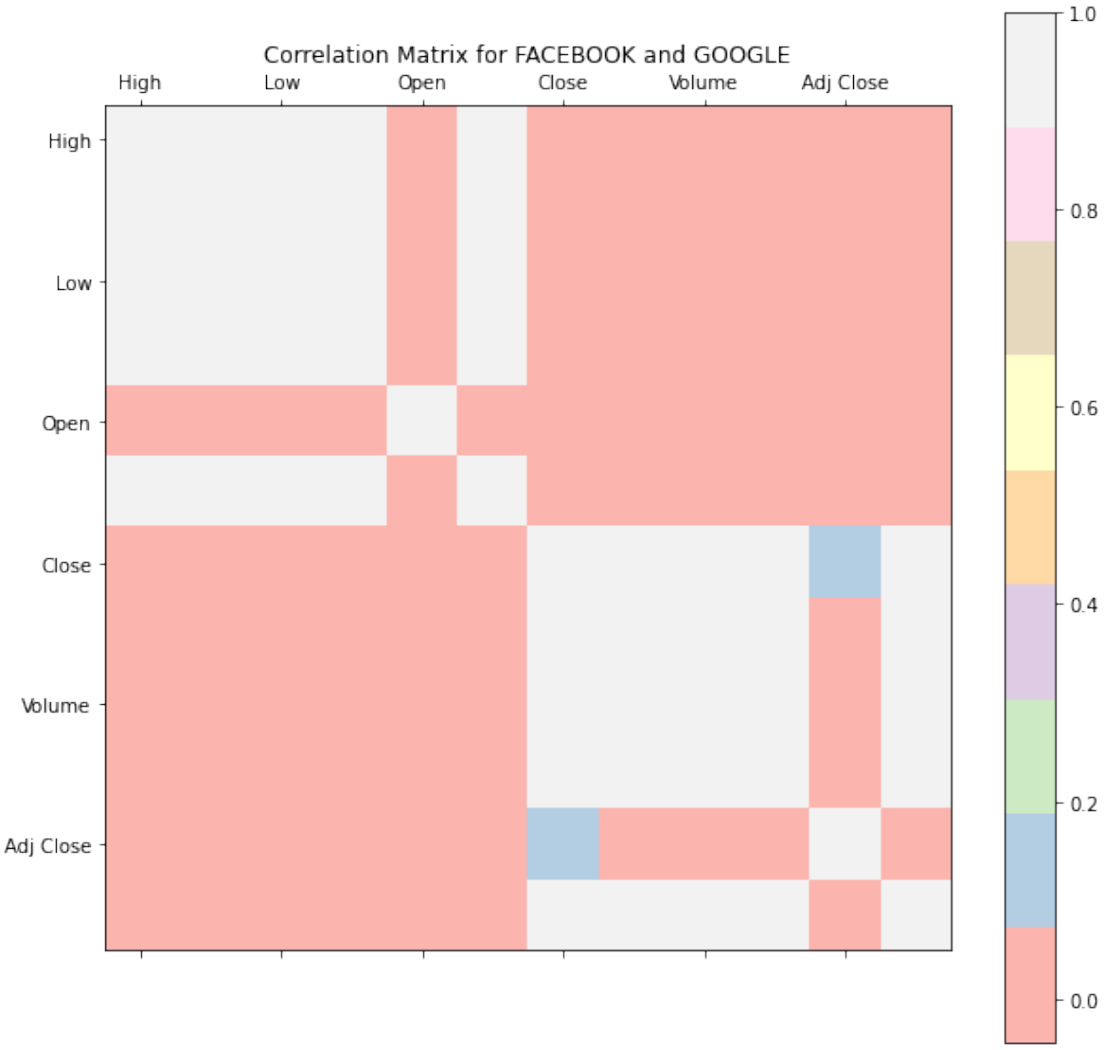

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



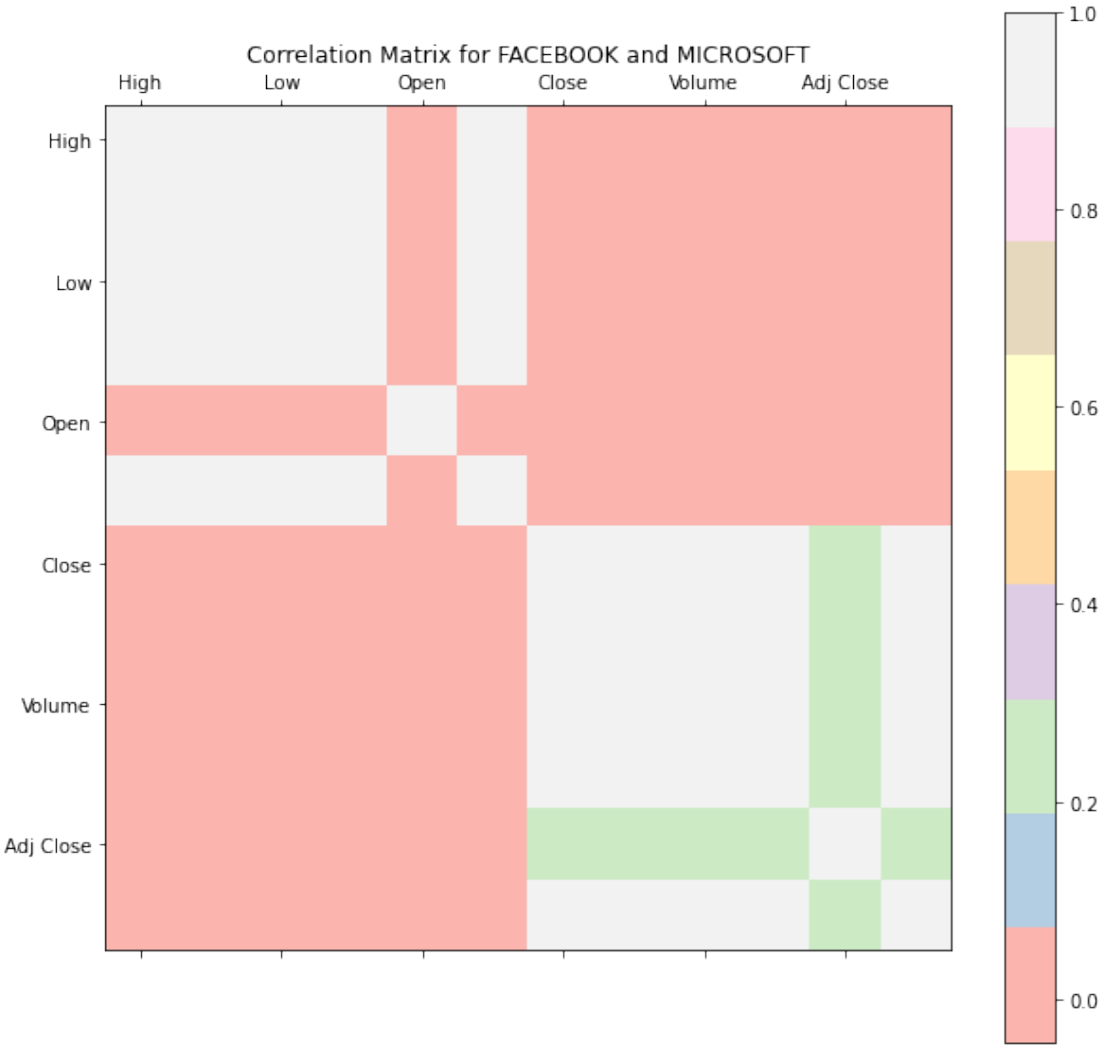

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



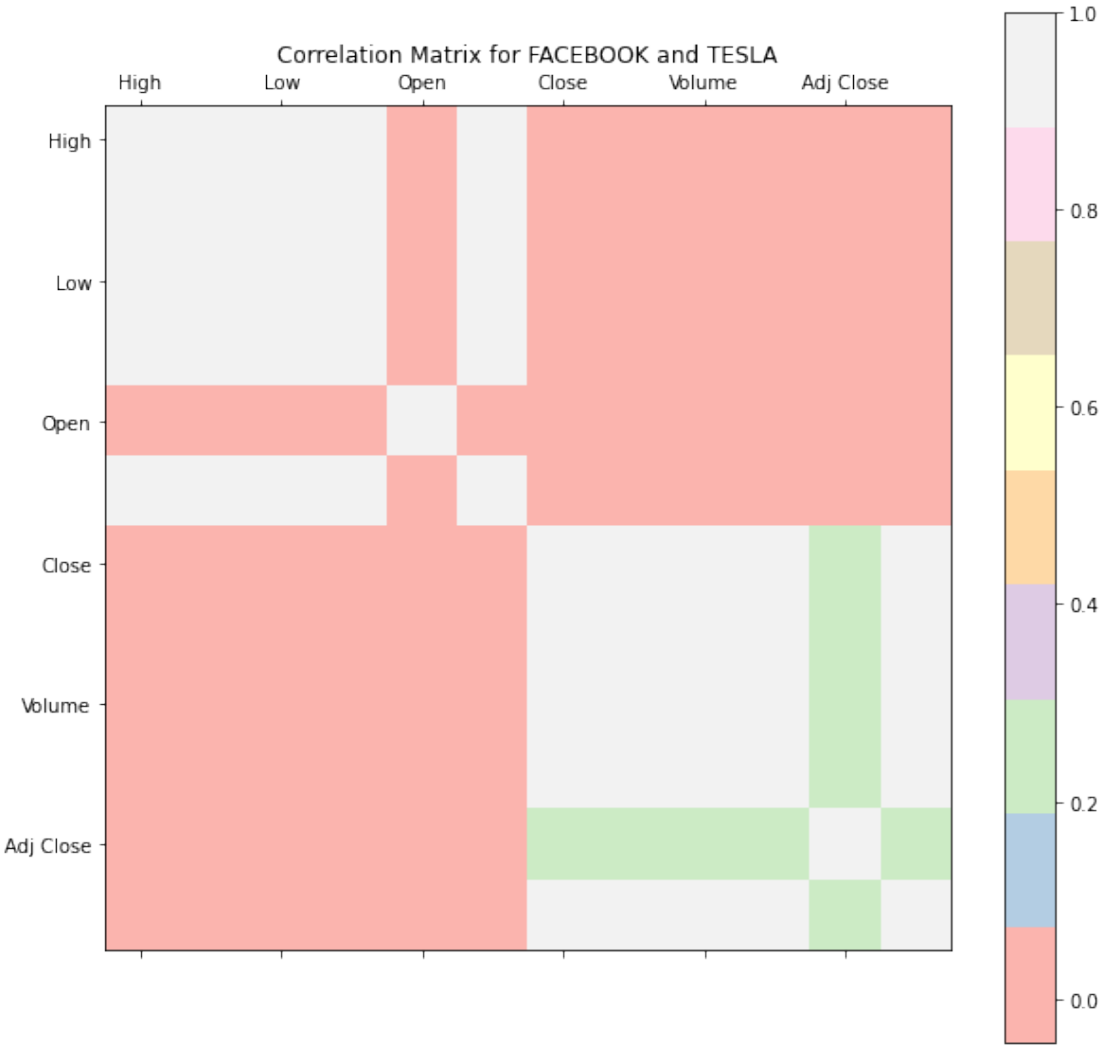

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



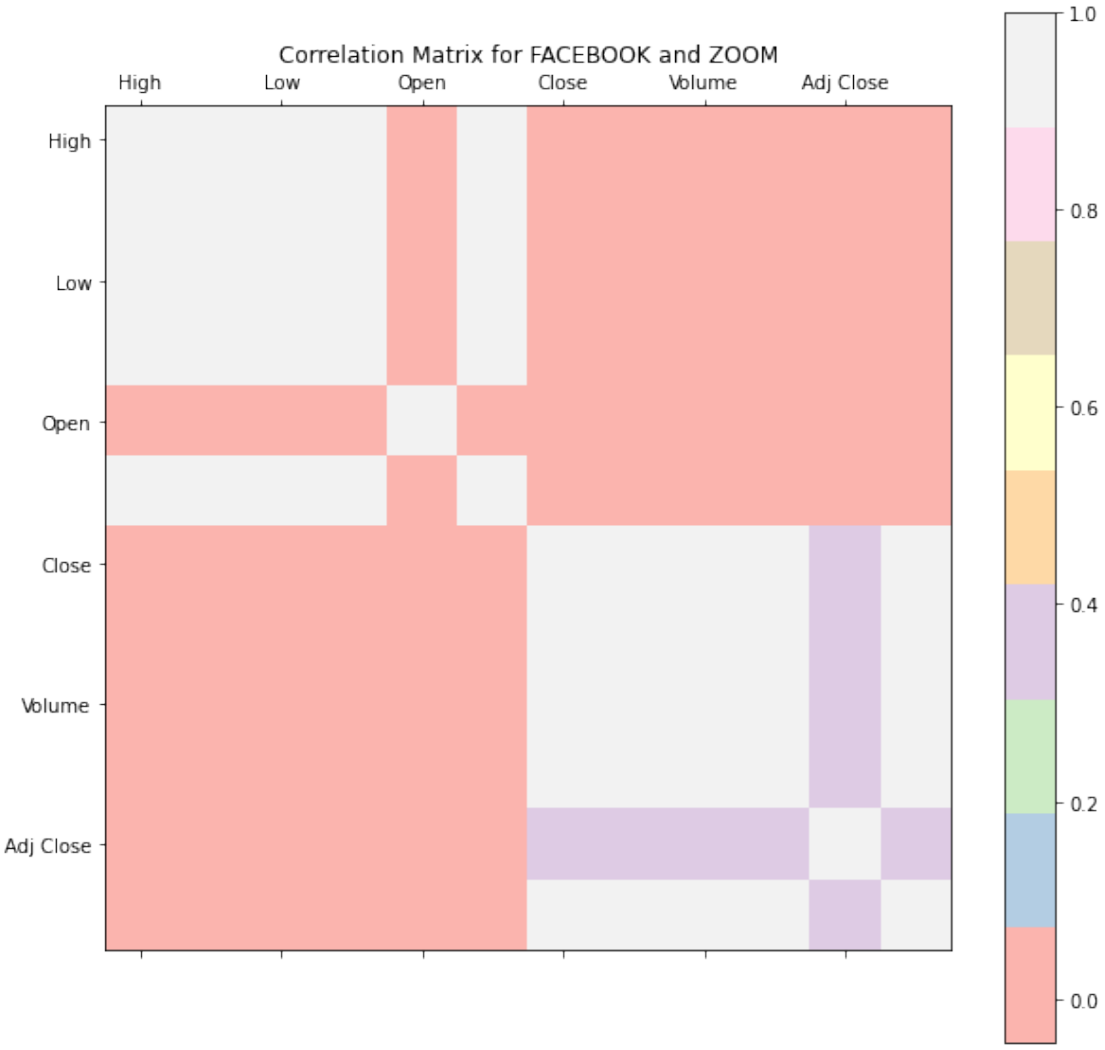

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



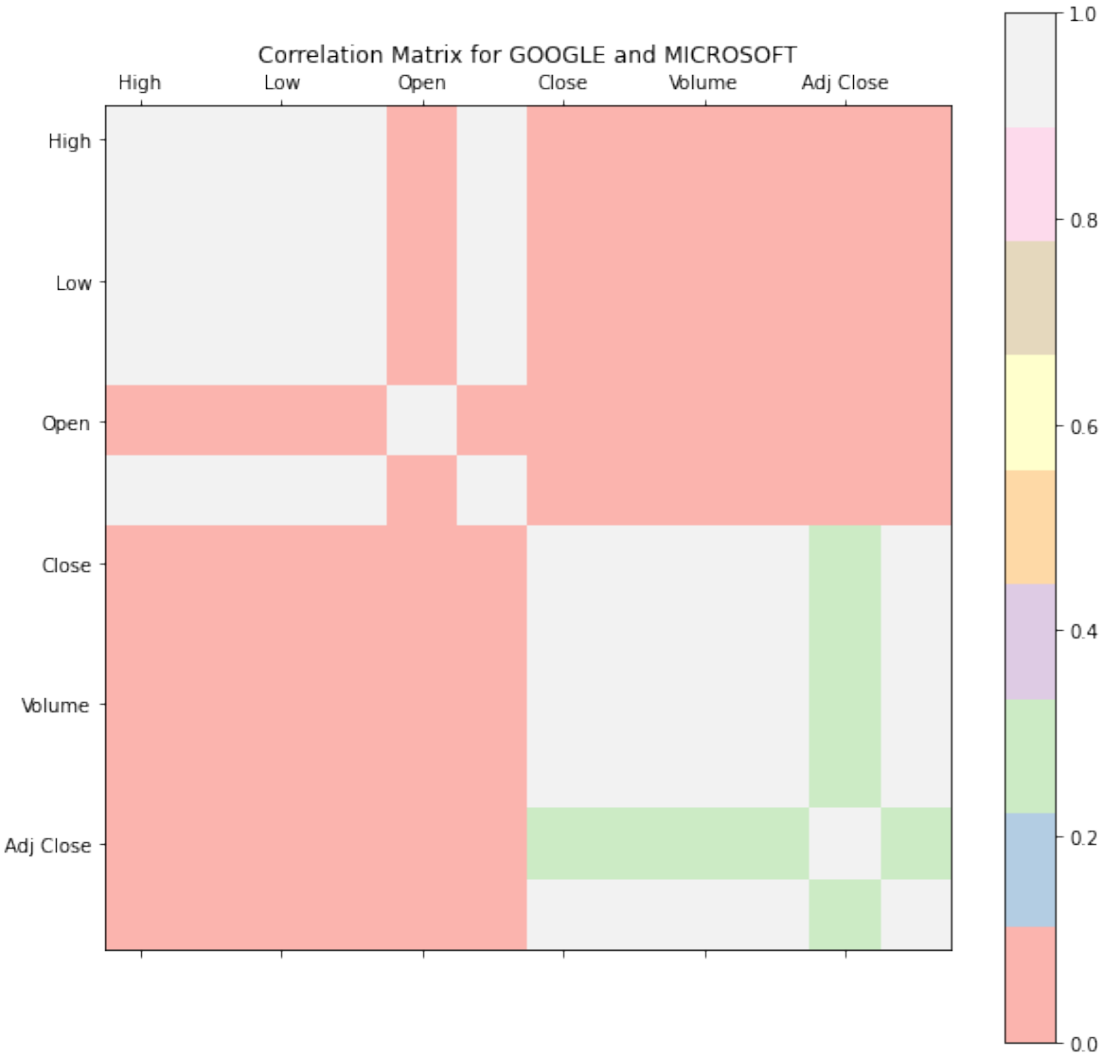

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



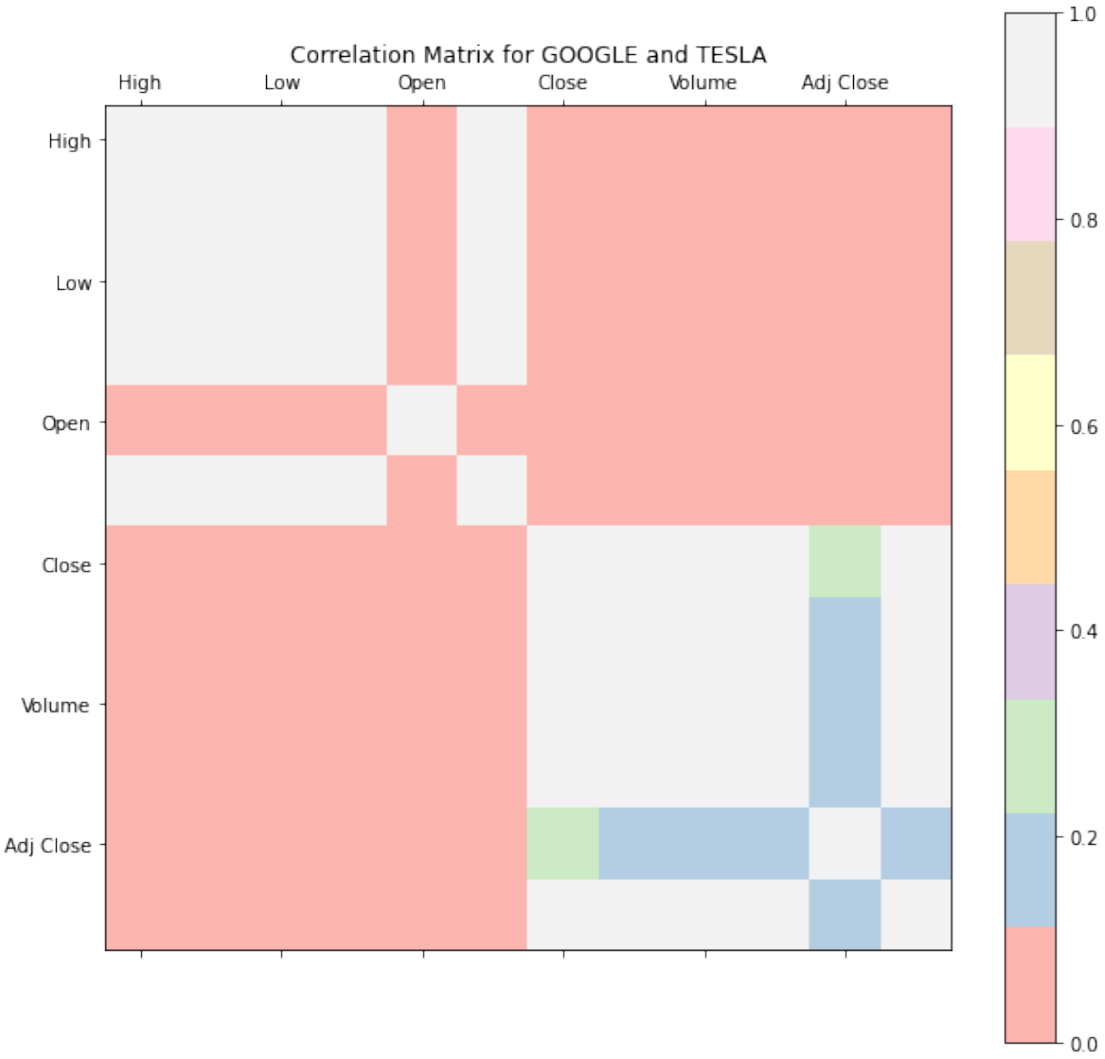

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



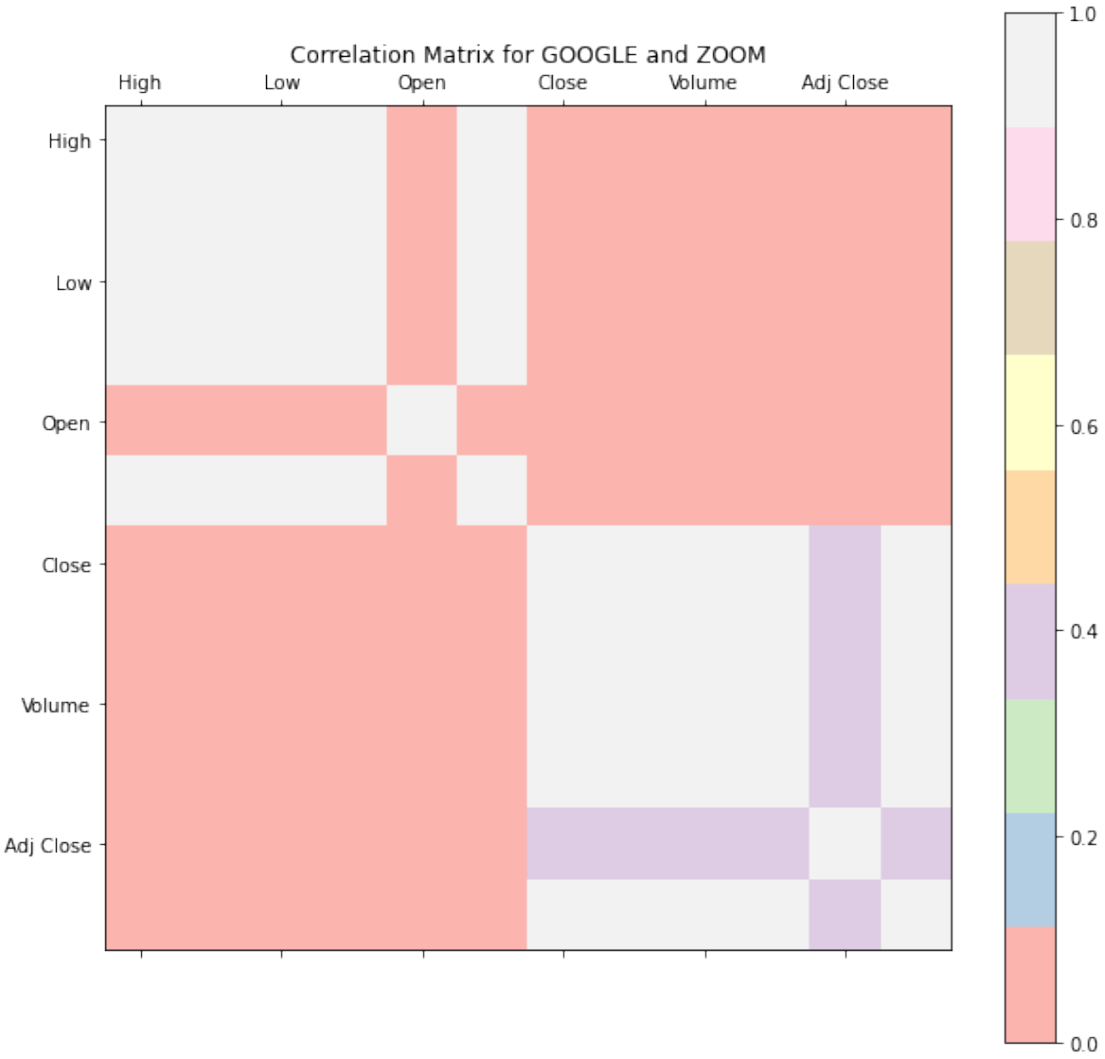

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



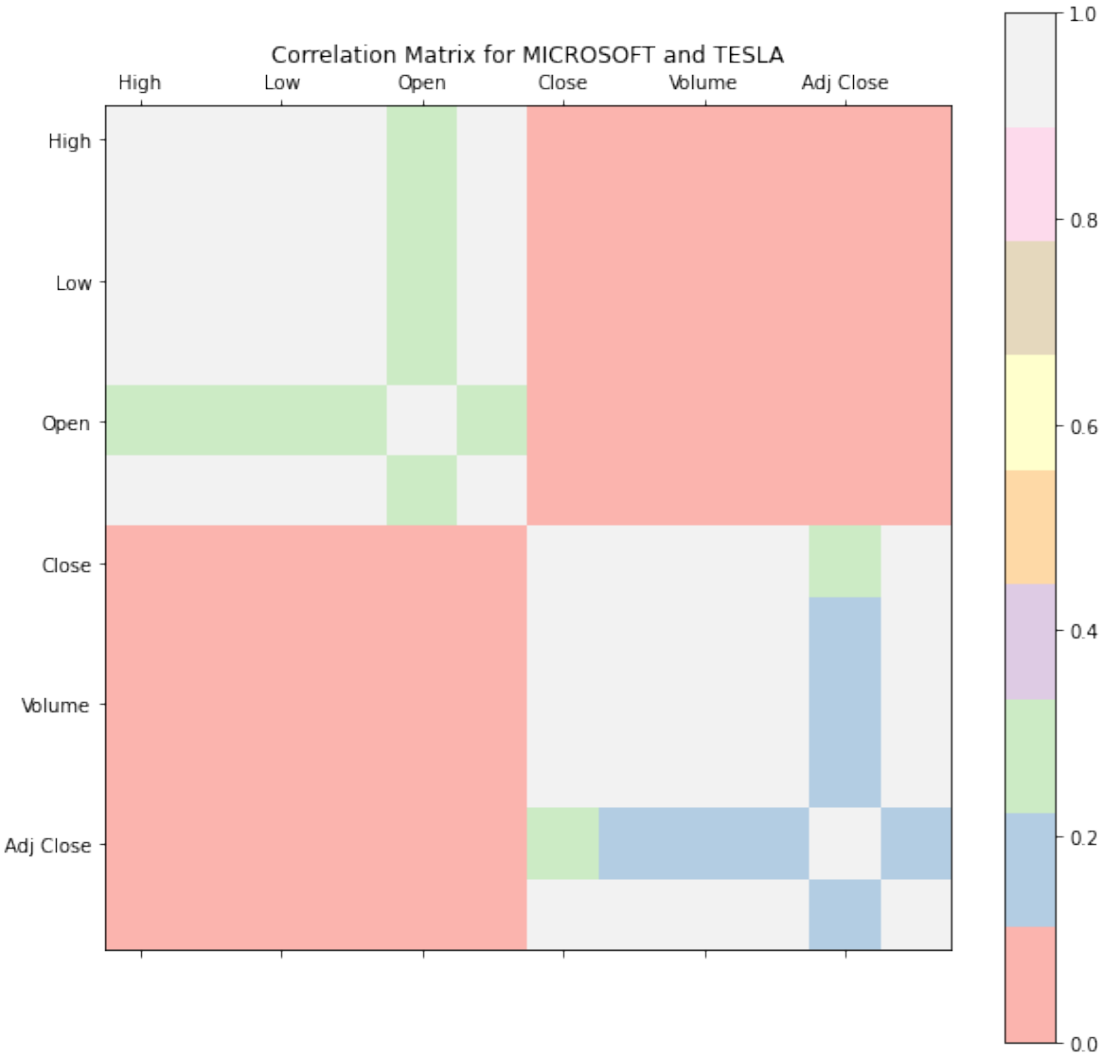

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



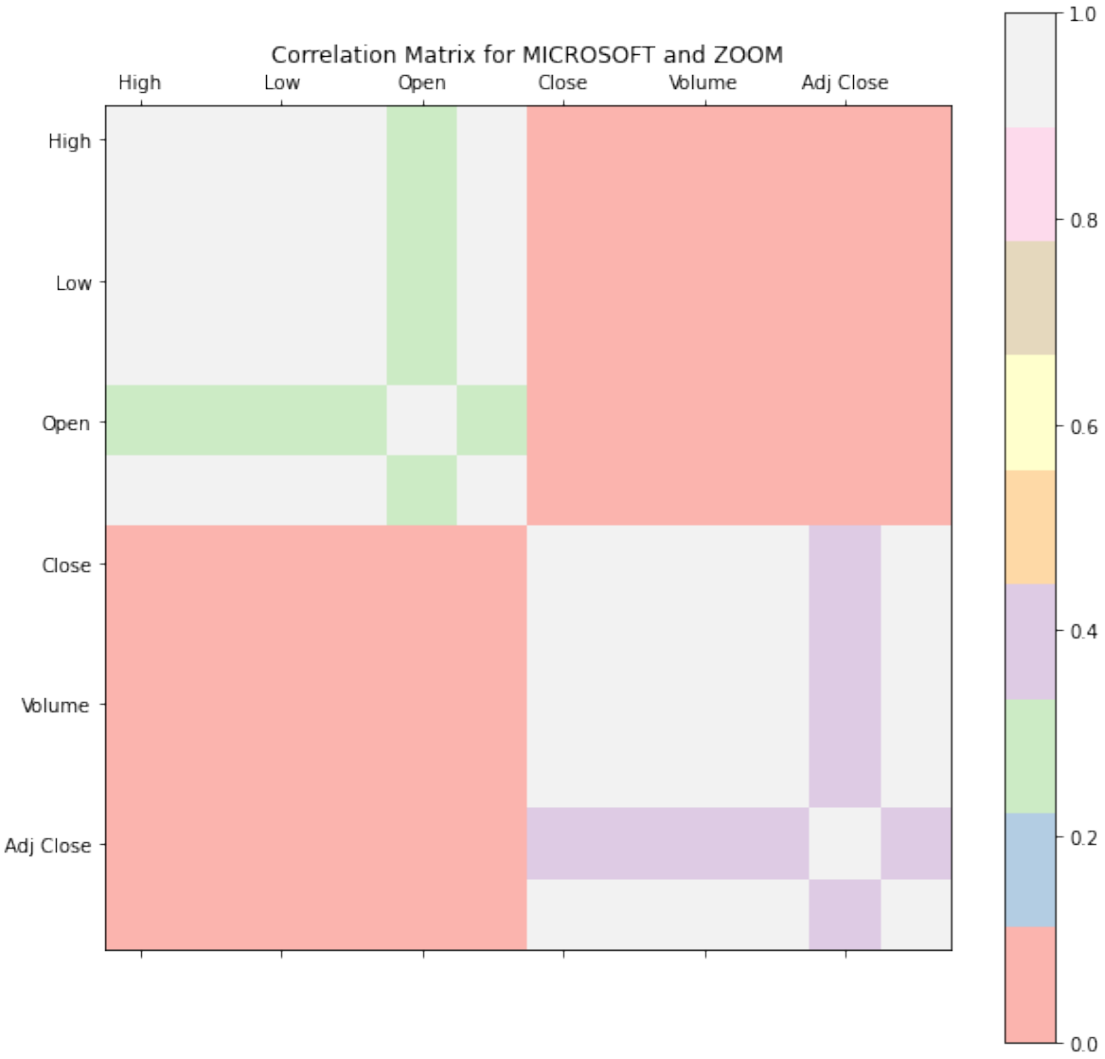

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



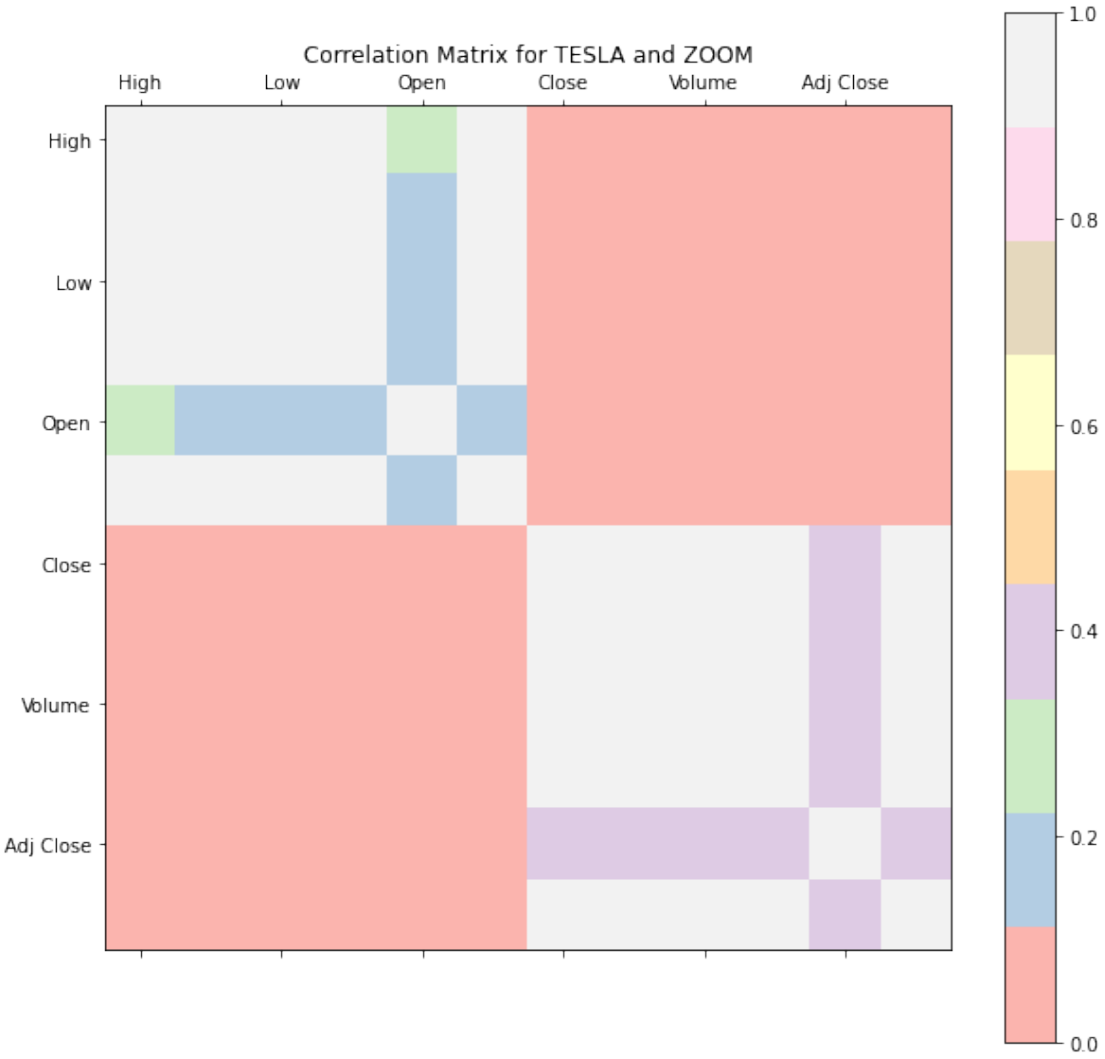

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```




```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```




```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



Return Rate

Fonction qui permet d'avoir le return rate en fonction d'une période. Prenons un exemple :

- Si l'on souhaite calculer le return rate sur 1 mois, alors on calcule la différence entre le close price du fin du mois et l'open price du premier jour du mois. A cette différence, on la soustrait par la valeur initiale du début du mois. On pultiplie le tout par 100 pour avoir un pourcentage.
- Formule : (end_value - init_value)/ init_value * 100

```
def return_rate(df, w=False, m=False, y=False):
    """
        :param
            - df : DataFrame
            - w : Boolean True if we want a week period, False
otherwise
            - m : Boolean True if we want a month period, False
otherwise
```

```
            - y : Boolean True if we want a year period, False
otherwise

        :return the return_rate following a specific period
    """
    list_period = get_list_period(df, w, m, y)

    for i in list_period:
        if y: df_filtered = df.filter(year(df['Date']) == i)
        elif m: df_filtered = df.filter(month(df['Date']) == i)
        else: df_filtered = df.filter(weekofyear(df['Date']) == i)

        print("Average return rate in", i)
        init_value = df_filtered.collect()[0]['Open']
        end_value = df_filtered.collect()[-1]['Close']
        return_rate = ((end_value - init_value) / init_value) * 100
        print("Return rate in ", i, "is " + str(return_rate)+"%")
```

Nous testons notre fonction pour le return rate de chaque dataframe sur chaque année

```
for df in LIST_DF:
    print("The return rate for each year for", df.first()
['company_name'])
    return_rate(df, y=True)
    print("-------------------------")
```

```
The return rate for each year for AMAZON
Average return rate in 2017
Return rate in  2017 is 54.29992572735133%
Average return rate in 2018
Return rate in  2018 is 28.154434360334896%
Average return rate in 2019
Return rate in  2019 is 26.115207985258248%
Average return rate in 2020
Return rate in  2020 is 71.38338541666667%
-------------------------
The return rate for each year for APPLE
Average return rate in 2017
Return rate in  2017 is 46.139888832213806%
Average return rate in 2018
Return rate in  2018 is -7.299011460770763%
Average return rate in 2019
Return rate in  2019 is 89.58615472504773%
Average return rate in 2020
Return rate in  2020 is 64.94640262601172%
-------------------------
The return rate for each year for FACEBOOK
Average return rate in 2017
Return rate in  2017 is 52.08136565571764%
Average return rate in 2018
Return rate in  2018 is -26.221295733000723%
```

```
Average return rate in 2019
Return rate in  2019 is 59.12085530601625%
Average return rate in 2020
Return rate in  2020 is 38.162028486812275%
-------------------------
The return rate for each year for GOOGLE
Average return rate in 2017
Return rate in  2017 is 34.358833052260174%
Average return rate in 2018
Return rate in  2018 is -1.2142988804961714%
Average return rate in 2019
Return rate in  2019 is 31.522670342253058%
Average return rate in 2020
Return rate in  2020 is 34.70388202778932%
-------------------------
The return rate for each year for MICROSOFT
Average return rate in 2017
Return rate in  2017 is 36.231883529681795%
Average return rate in 2018
Return rate in  2018 is 17.92639374639424%
Average return rate in 2019
Return rate in  2019 is 58.41284993858932%
Average return rate in 2020
Return rate in  2020 is 34.85766503815274%
-------------------------
The return rate for each year for TESLA
Average return rate in 2017
Return rate in  2017 is 44.908313043083034%
Average return rate in 2018
Return rate in  2018 is 6.6666601458168%
Average return rate in 2019
Return rate in  2019 is 36.66448660232486%
Average return rate in 2020
Return rate in  2020 is 564.346312320088%
-------------------------
The return rate for each year for ZOOM
Average return rate in 2019
Return rate in  2019 is 4.6769244854266825%
Average return rate in 2020
Return rate in  2020 is 509.9246885643587%
-------------------------
```

More Insights

MACD (Moving Average Convergence and Divergence)

On va créer une fonction qui nous calcul la MACD. Il nout faut 3 valeurs :

- MACD = EMA12 - EMA26
- SIGNAL = EMA9(MACD)

- Difference = MACD - SIGNAL

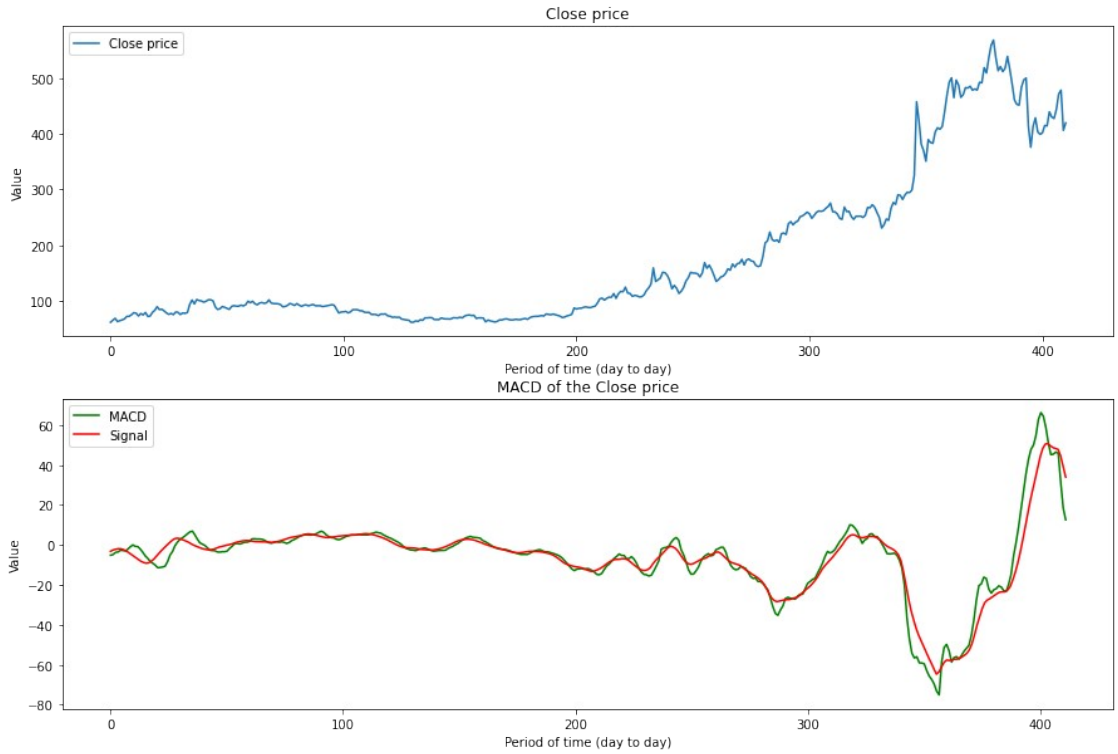
Pour chaquie MACD, il nous faut calculer l'EMA et non la SMA, ce qui nous oblige à faire une nouvelle fonction. Une complète analyse représente la visualisation de ces 3 courbes en plus de celle du prix que l'on souhaite observer.

Ici, on a les valeurs 12, 26 et 9 qui sont des valeurs basiques qui ont été mises en référence dans ces fonctions. Bien sur, il est évident que l'on peut prendre des périodes différentes selon le besoin.

```
def exponential_moving_average(df, column_name, period, smooth=2):
    """
        :param
            - df : dataframe
            - column_name : specific price
            - period :  period time selected
            - smooth : smoothness use in the formula (usually 2)

        :return the exponential moving average for a specific price on
a specific period of time
    """
    prices = df.select(column_name).rdd.flatMap(lambda x: x).collect()
    ema = [sum(prices[:period]) / period]

    for price in prices[period:]:
        ema.append((price * (smooth / (1 + period))) + ema[-1] * (1 -
(smooth / (1 + period))))

    return ema

def get_MACD(df, column_name, period1=12, period2=26, smooth=2):
    """
        :param
            - df : the DataFrame
            - column_name : specific price
            - period1 : used to calculate first EMA (usually 12)
            - period2 : used to calculate second EMA (usually 26)
            - smooth : (usually 2) -> used in formula to get the EMA

        :return the MACD calculated with the formula : MACD =
EMA(period1) - EMA(period2)
    """
    sma_26 = exponential_moving_average(df, column_name, period2,
smooth)
    sma_12 = exponential_moving_average(df, column_name, period1,
smooth)
    diff = len(sma_12) - len(sma_26)
    sma_12 = sma_12[:-diff]
    sma_12 = np.array(sma_12)
    sma_26 = np.array(sma_26)
```

```
    return list(np.subtract(sma_12, sma_26))


def get_signal(macd, signal_period=9, smooth=2):
    """
        :param
            - macd : MACD calculated previously
            - signal_period : period used to calculate the EMA of the
signal(usually 9)
            - smooth : (usually 2) -> used in formula to get the EMA

        :return the EMA of the MACD calculated before -> called the
signal
    """
    df_macd = spark.createDataFrame([float(x) for x in macd],
FloatType())
    signal = exponential_moving_average(df_macd, "value",
signal_period, smooth)
    return signal

def visualization_MACD(df, column_name, period1=12, period2=26,
signal_period=9, smooth=2):
    """
        :param
            - df : the DataFrame
            - column_name : specific price
            - period1 : used to calculate first EMA (usually 12)
            - period2 : used to calculate second EMA (usually 26)
            - signal_period : period used to calculate the EMA of the
signal(usually 9)
            - smooth : (usually 2) -> used in formula to get the EMA

        :return : Visualization of a specific price in constrast with
both MACD and signal curves calculated.

    """
    #Get MACD and SIGNAL
    macd = get_MACD(df, column_name, period1, period2, smooth)
    signal = get_signal(macd,signal_period, smooth)

    figure, axes = plt.subplots(2,1, figsize=(15,10))

    #Create linspace for MACD and SIGNAL to facilitate the
vizualisation
    x_macd = np.linspace(0,
len(df.select(column_name).rdd.flatMap(lambda x: x).collect()),
num=len(macd))
    x_signal = np.linspace(0,
len(df.select(column_name).rdd.flatMap(lambda x: x).collect()),
```

```
num=len(signal))

    #Plotting MACD
    axes[1].plot(x_macd, macd, label='MACD', c='g')
    axes[1].plot(x_signal, signal, label='Signal', c='r')
    axes[1].set_xlabel("Period of time (day to day)")
    axes[1].set_ylabel("Value")
    title = "MACD of the " + column_name + " price"
    axes[1].set_title(title)
    axes[1].legend()

    #Plotting normal price
    label = column_name + " price"
    axes[0].plot(df.select(column_name).rdd.flatMap(lambda x:
x).collect(), label=label)
    title = column_name + " price"
    axes[0].set_title(title)
    axes[0].set_xlabel("Period of time (day to day)")
    axes[0].set_ylabel("Value")
    axes[0].legend()


visualization_MACD(df_zoom, "Close")
```



Aroon Oscillator

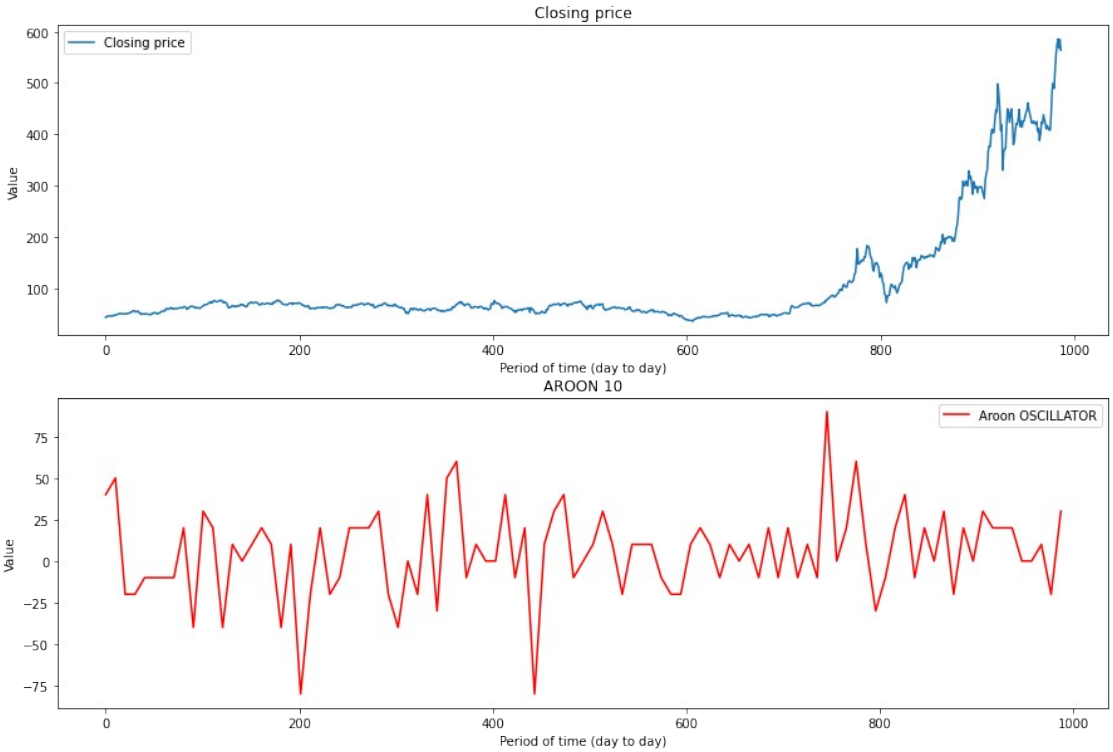
Cette fois-ci, nous allons voir ce qu'est le Aroon Oscillator. Le Aroon oscillator est la différence entre deux indicateurs qui sont le Aroon up et le Aroon down. Pour calculer toutes ces valeurs, on utilise les formules suivantes :

- Arron up = [25 - nb_period avec nouveau High] / 25 * 100
- Arron down = [25 - nb_period avec nouveau Low] / 25 * 100

Ensuite, nous n'avons plus au'a **plot** cette courbe.

```
def count_days(high, low):
    """
        :param
            - high : list of high price
            - low : list of low price

        :return (count_high, count_low) : time with a new high price
reached and a new low price reached
    """
    count_high = 1
    count_low = 1
    for i in range(len(high)-1):
        if high[i+1] > high[i]: #New High reached
            break
        else:
            count_high += 1

    for i in range(len(low)-1):
        if low[i+1] < low[i]: #New Low reached
            break
        else:
            count_low += 1

    return count_high, count_low

def get_aroon_indicators(df, period=25):
    """
        :param
            - df : the Dataframe
            - period : number used for the formula (usually 25)

        :return (aroon_up, aroon_down) : two indicators for aroon
oscillator
    """
    aroon_up = []
    aroon_down = []
    for i in range (0, df.count(), period):
        df_filtered = spark.createDataFrame(df.collect()[i:
(i+period)])
```

```
        high = df_filtered.select("High").rdd.flatMap(lambda x :
x).collect()
        low = df_filtered.select("Low").rdd.flatMap(lambda x :
x).collect()

        count_high, count_low = count_days(high, low)
        aroon_up.append((period - count_high) / period * 100)
        aroon_down.append((period - count_low) / period * 100)

    return aroon_up, aroon_down

def plot_aroon_oscillator(df, period=25):
    """
        :param
            - df : the DataFrame
            - period : number used for the formula to calculate the
Aroon indicators (usually 25)

        :return : Plot the Aroon Oscillator calculated in
comparisation with usually the closing price
    """
    #Get Aroon indicators and create Aroon Oscillator
    aroon_up, aroon_down = np.array(get_aroon_indicators(df, period))
    aroon_oscillator = list(np.subtract(aroon_up, aroon_down))

    figure, axes = plt.subplots(2,1, figsize=(15,10))
    x_aroon = np.linspace(0, len(df.select("High").rdd.flatMap(lambda
x: x).collect()), num=len(aroon_up))

    #Plotting Aroon up and Aroon down
    axes[1].plot(x_aroon, aroon_oscillator, label='Aroon OSCILLATOR',
c='r')
    axes[1].set_xlabel("Period of time (day to day)")
    axes[1].set_ylabel("Value")
    title = "AROON " + str(period)
    axes[1].set_title(title)
    axes[1].legend()

    #Plotting normal price
    axes[0].plot(df.select("Close").rdd.flatMap(lambda x:
x).collect(), label="Closing price")
    axes[0].set_title("Closing price")
    axes[0].set_xlabel("Period of time (day to day)")
    axes[0].set_ylabel("Value")
    axes[0].legend()

plot_aroon_oscillator(df_tesla, 10)
```



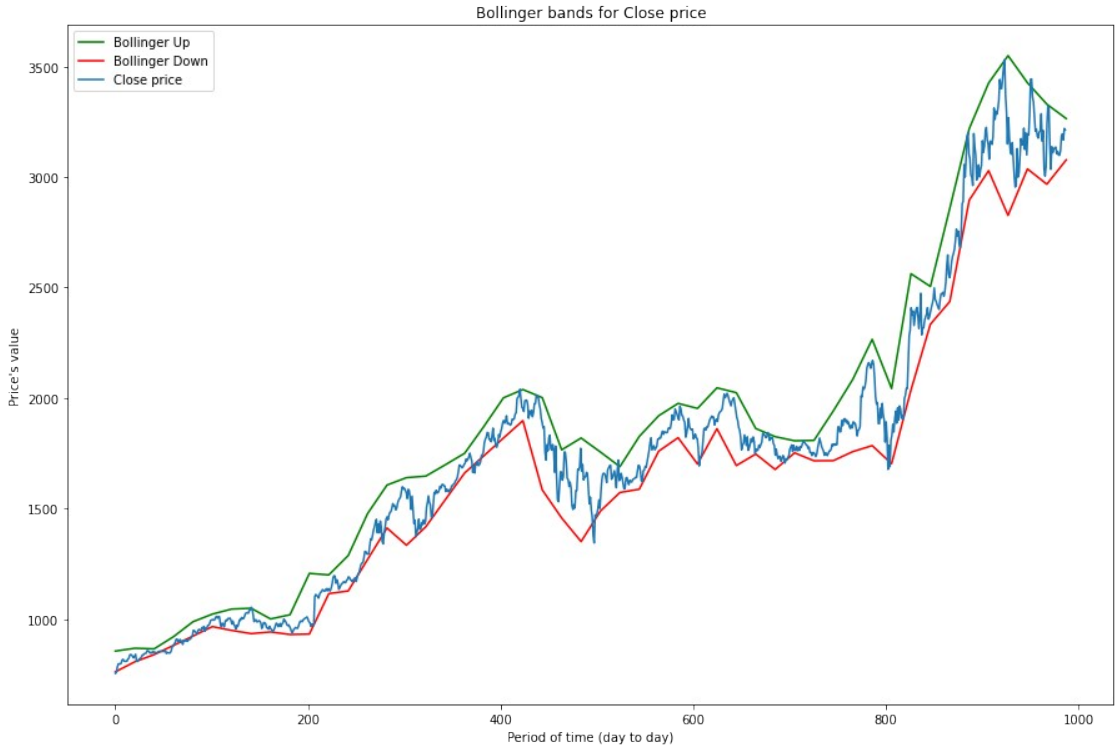
Bollinger Bands

Pour faire les bandes de Bollinger, nous devons dans un premier temps calculer la SMA (Simple Moving Average) ainsi que la std selon une periode precise. Avec ces deux valeurs, on créé donc les deux bandes de Bollinger que l'on compare avec les valeurs normales du prix.

```
def get_SMA(df, column_name, period):
    """
        :param
            - df : the DataFrame
            - column_name :
            - period : period used to calculate the SMA

        :return list_SMA : the SMA calculated
    """
    list_SMA = []
    for i in range (0,df.count(), period):
        df_filtered = spark.createDataFrame(df.collect()[i:
(i+period)])
        sma =
df_filtered.agg(mean(df_filtered[column_name])).collect()[0][0]
        list_SMA.append(sma)

    return list_SMA

def get_std(df, column_name, period):
```

```
    """
        :param
            - df : the DataFrame
            - column_name :
            - period : period used to calculate the SMA

        :return list_std : the std calculated
    """
    list_std = []
    for i in range (0,df.count(), period):
        df_filtered = spark.createDataFrame(df.collect()[i:
(i+period)])
        std =
df_filtered.agg(stddev(df_filtered[column_name])).collect()[0][0]
        list_std.append(std)

    return list_std

def get_bollinger_bands(df, column_name, period):
    """
        :param
            - df : the DataFrame
            - column_name :
            - period : period used to calculate the SMA

        :return (bollinger_up, bollinger_down) : 2 curves that form
the bollinger bands
    """
    sma = np.array(get_SMA(df,column_name, period))
    std = np.array(get_std(df,column_name,period))

    bollinger_up = sma + std * 2
    bollinger_down = np.subtract(sma, std*2)

    return bollinger_up, bollinger_down

def plot_bollinger_bands(df, column_name, period):
    """
        :param
            - df : the DataFrame
            - column_name :
            - period : period used to calculate the SMA

        :return : Plot the Bolligner bands with the specific price we
want
    """
    #Get Bollinger Bands
    bollinger_up, bollinger_down = get_bollinger_bands(df,
column_name, period)
```

```
    #Create linspace for both bands fot the vizualisation
    x_bollinger_up = np.linspace(0,
len(df.select(column_name).rdd.flatMap(lambda x: x).collect()),
num=len(bollinger_up))
    x_bollinger_down = np.linspace(0,
len(df.select(column_name).rdd.flatMap(lambda x: x).collect()),
num=len(bollinger_down))

    figure, axes = plt.subplots(1, figsize=(15,10))

    #Plotting Bollinger Bands
    axes.plot(x_bollinger_up,bollinger_up, label='Bollinger Up',
c='g')
    axes.plot(x_bollinger_down,bollinger_down, label='Bollinger Down',
c='r')

    #Plotting the specific price with its name
    label = column_name + " price"
    axes.plot(df.select(column_name).rdd.flatMap(lambda x:
x).collect(), label=label)

    #Legend + title
    axes.set_xlabel("Period of time (day to day)")
    axes.set_ylabel("Price's value")
    title = "Bollinger bands for " + column_name + " price"
    axes.set_title(title)
    axes.legend()

plot_bollinger_bands(df_amazon, 'Close', 20)
```



Portfolio Diversification

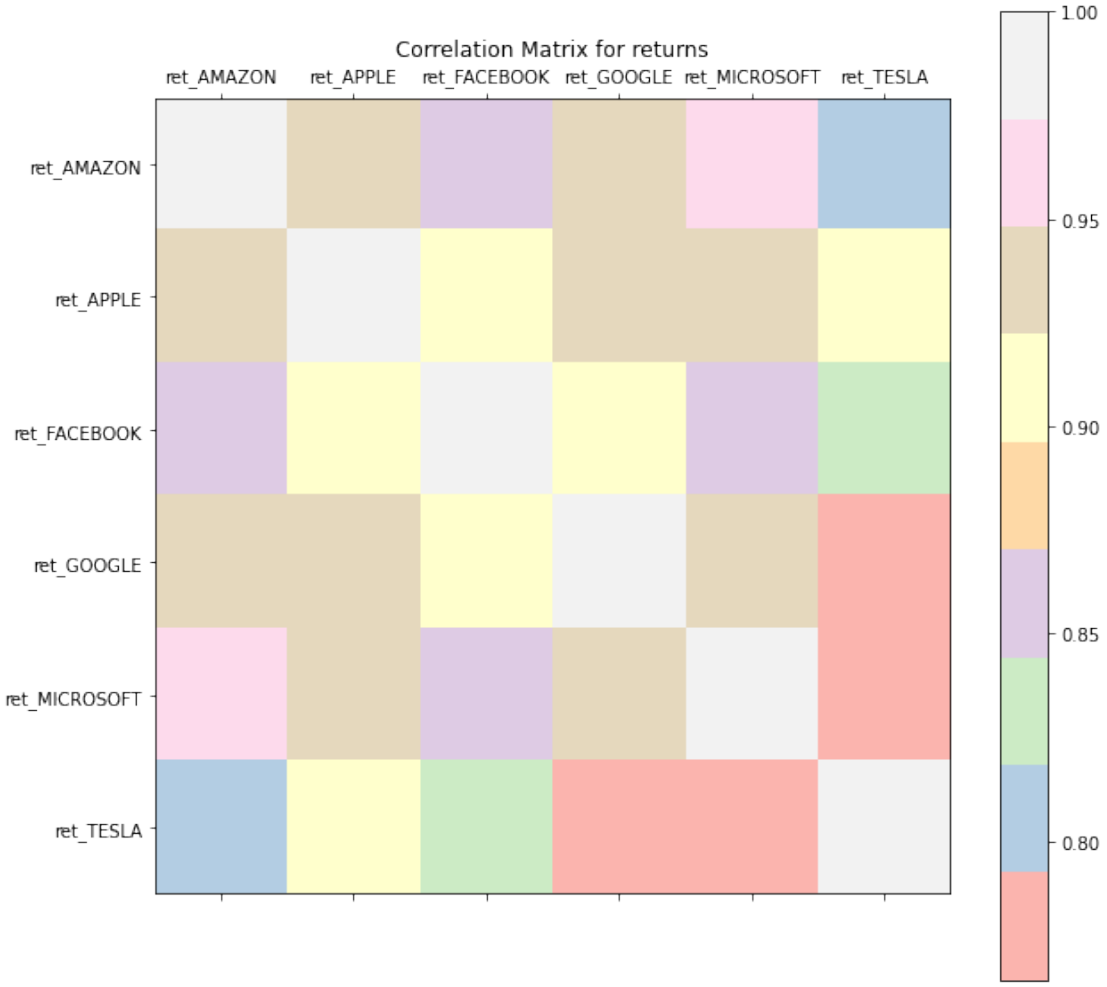
Ce concept consiste à avoir un nombre diversifié de stocks dans un seul portfolio qui sont correlés. Cela permet de réduire les risques et aussi d'accroitre des revenus. Pour se faire, on calcule les **returns** pour chaque stocks et on les compare au sein d'une matrice de correlation

```
def create_df_returns():
    """
        :return : new DataFrame containing the returns for each
DataFrame.
    """

    #Calcul returns for each dataframe
    returns = []
    cols = []
    for df in LIST_DF:
        name = df.first()['company_name']
        if name != "ZOOM":
            df_returns = df.withColumn("returns", lit(df['Close'] /
df.count()))
            list_ret = df_returns.select("returns").rdd.flatMap(lambda
x: x).collect()

            returns.append(list_ret)
            cols.append("ret_"+name)
```

```
    #Construct data to fit in a DataFrame
    data = []
    lg = len(returns[0])
    for i in range(lg):
        data.append([item[i] for item in returns])

    #Create Dataframe with returns for each stocks and plot its
correlation matrix
    df_returns = spark.createDataFrame(data, cols)
    return df_returns

df_returns = create_df_returns()
matrix = get_corr_matrix(df_returns)
visualization_corr_matrix(matrix, df_returns.columns, "returns")
```

```
['ret_AMAZON', 'ret_APPLE', 'ret_FACEBOOK', 'ret_GOOGLE',
'ret_MICROSOFT', 'ret_TESLA']
```

```
/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```



API

Find Company

On crée une fonction qui permet de trouver, avec un nom, les symboles correnspondants à des entreprises qui sont cotées en bourses. Il suffit ensuite de prendre un de ces symboles pour récupérer les informations de ses stocks dans la prochaine fonction.

```
def find_company(company_name):
    url = 'https://www.alphavantage.co/query?
function=SYMBOL_SEARCH&keywords='+ company_name
+'&apikey=BG5RZ7YIV7QX2UYN'
    r = requests.get(url)
    data = r.json()
    for e in data['bestMatches']:
        print('Company Name : ' + e['2. name'] + '\n'+ 'Symbol : ' +
e['1. symbol'] + '\n----------------------\n')
find_company('Apple')
```

```
Company Name : Apple Hospitality REIT Inc
Symbol : APLE
```

```
---------------------

Company Name : Apple Inc
Symbol : AAPL
---------------------

Company Name : Apple Inc
Symbol : AAPL34.SAO
---------------------

Company Name : Apple Inc
Symbol : APC.DEX
---------------------

Company Name : Apple Inc
Symbol : APC.FRK
---------------------

Company Name : Apple Green Holding Inc
Symbol : AGPL
---------------------

Company Name : Apple Inc.
Symbol : 0R2V.LON
---------------------

Company Name : Apple Rush Company Inc
Symbol : APRU
---------------------

Company Name : Apple Finance Limited
Symbol : 500014.BSE
---------------------

Company Name : Apple Flavor Fragrance Group Company Ltd
Symbol : 603020.SHH
---------------------
```

Get api stock

On crée une fonction qui permet de trouver, avec le symbole d'une entreprise en bourse, les stocks correnspondants à cette entreprise par mois pendant 20 ans.

```
def get_api_stock(company_name='AMZN'):
    url_api = 'https://www.alphavantage.co/query?
function=TIME_SERIES_MONTHLY_ADJUSTED&symbol='+ company_name +
'&apikey=BG5RZ7YIV7QX2UYN'
    request_api = requests.get(url_api)
    data_api = request_api.json()
```

```
    schema = StructType([
    StructField('Date', TimestampType(), True),
    StructField('High', FloatType(), True),
    StructField('Low', FloatType(), True),
    StructField('Open', FloatType(), True),
    StructField('Close', FloatType(), True),
    StructField('Volume', FloatType(), True),
    StructField('Adj Close', FloatType(), True),
    StructField('company_name', StringType(), True),
    ])


    df_api = spark.sparkContext.emptyRDD()
    df_api = spark.createDataFrame(df_api,schema)
    columns = ['Date','High','Low','Open','Close','Volume','Adj
Close','company_name']
    for e in data_api['Monthly Adjusted Time Series']:
        d = data_api['Monthly Adjusted Time Series']
        newrow_api = spark.createDataFrame([(e,d[e]['2. high'],d[e]
['3. low'],d[e]['1. open'],d[e]['4. close'],d[e]['6. volume'],d[e]['5.
adjusted close'],company_name)], columns)
        df_api = df_api.union(newrow_api)
    return df_api

get_api_stock('APLE').show(truncate=False)

+----------+-------+-------+-------+-------+--------+---------
+------------+
|Date      |High   |Low    |Open   |Close  |Volume  |Adj Close|
company_name|
+----------+-------+-------+-------+-------+--------+---------
+------------+
|2022-05-20|17.8300|15.2300|17.7000|15.5200|34790676|15.5200  |APLE
|
|2022-04-29|18.6900|16.2300|18.1300|17.6900|44791739|17.6396  |APLE
|
|2022-03-31|18.6500|16.6700|17.7400|17.9700|53778038|17.8694  |APLE
|
|2022-02-28|17.9150|15.8100|16.1600|17.6900|38442512|17.5420  |APLE
|
|2022-01-31|17.0500|14.9550|16.2100|16.1300|32915754|15.9951  |APLE
|
|2021-12-31|16.4100|14.5550|15.3000|16.1500|46934884|16.0149  |APLE
|
|2021-11-30|17.4400|14.3600|15.7200|15.0200|50394080|14.8851  |APLE
|
|2021-10-29|16.4550|15.2934|15.9000|15.7100|30080164|15.5689  |APLE
|
|2021-09-30|16.5900|14.4800|15.1000|15.7300|42681030|15.5888  |APLE
```

```
|
|2021-08-31|15.4200|14.1100|15.0300|14.7800|33272540|14.6382  |APLE
|
|2021-07-30|15.5500|13.8300|15.4300|14.9500|34030773|14.8065  |APLE
|
|2021-06-30|16.5700|14.8250|16.1300|15.2600|65912767|15.1136  |APLE
|
|2021-05-28|16.1400|14.7150|15.9400|15.8700|38296838|15.7072  |APLE
|
|2021-04-30|15.9300|14.5500|14.7600|15.8600|39071438|15.6974  |APLE
|
|2021-03-31|16.0600|13.4700|14.6500|14.5700|47474572|14.4206  |APLE
|
|2021-02-26|15.1500|12.3600|12.5400|14.2500|39101362|14.0944  |APLE
|
|2021-01-29|13.5400|12.2900|12.9300|12.4800|36247691|12.3438  |APLE
|
|2020-12-31|13.8100|12.4200|13.4900|12.9100|44650849|12.7691  |APLE
|
|2020-11-30|13.5400|9.7150 |10.0200|13.2600|59036169|13.1152  |APLE
|
|2020-10-30|11.0600|9.5550 |9.6600 |9.9000 |36419758|9.7919   |APLE
|
+----------+-------+-------+-------+-------+--------+---------
+------------+
only showing top 20 rows
```

Main

Dernière patie, nous créeons une fonction **main** servant à appeler toutes les foncions utiles à l'analyse d'un DataFrame contenant des informations sur les acions d'une certaine entreprise.

Cette fonction **main** appelle cherche tous les fichiers correpondant à des fichiers **CSV** et les traite un par un

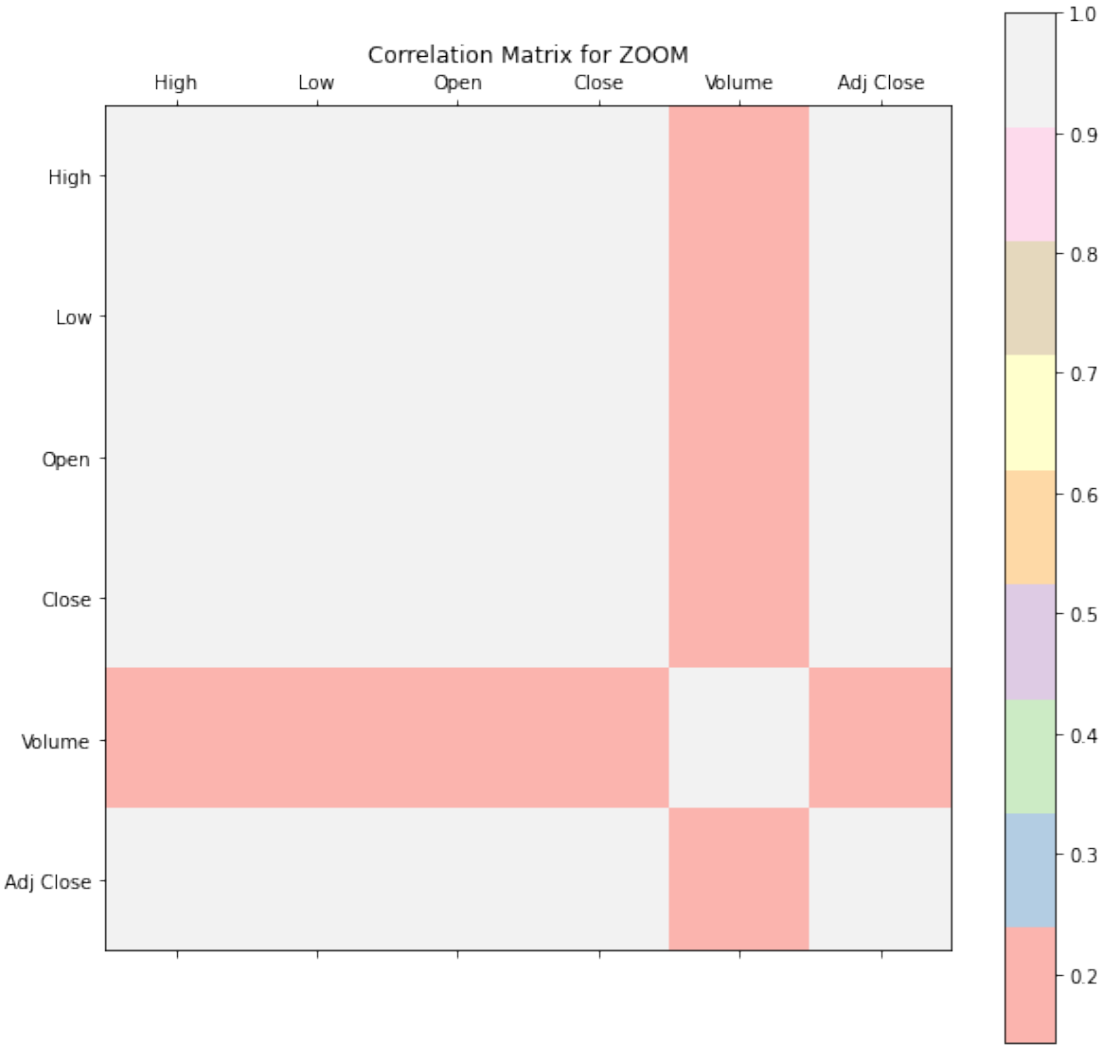
```python
def main_visualization():
    all_df = []
    files = os.listdir('./stocks_data')
    for file in files :
        print("Visualization and plots for", file[:-4])
        df_main = read_infos("./stocks_data/" + file) #Infos
        all_df.append(df_main)

        #Visualization
        matrix = get_corr_matrix(df_main)
        df_num = df_main.drop("Date", "company_name")
        visualization_corr_matrix(matrix, df_num.columns, name)
        average_price(df_main, y=True)
```

```
        plot_avg_price(df_main, y=True)
        plot_evolution_stock_prices(df_main,m=True)

        #Insights
        visualization_MACD(df_main, "Close")
        plot_aroon_oscillator(df_main, 10)
        plot_bollinger_bands(df_main, 'Close', 20)

        print("================")

    print("Moving Average for all prices")
    cols = ["High", "Low", "Open", "Close", "Volume", "Adj Close"]
    for c in cols:
        visualization_moving_average(c, 50)

    print("We want to see all correlations between the differents
DataFrame created")
    for (df1,df2) in itertools.combinations(all_df, 2):
        df_merged = merge(df1,df2)
        mat = get_corr_matrix(df_merged)
        final_name = df1.first()['company_name'] + " and " +
df2.first()['company_name']
        visualization_corr_matrix(mat,df_merged.columns, final_name)

    #Insight for all DataFrames (correlation matrix for returns)
    df_returns = create_df_returns()
    matrix = get_corr_matrix(df_returns)
    visualization_corr_matrix(matrix, df_returns.columns, "returns")

main_visualization()
```

```
Visualization and plots for APPLE
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+-------+-------+-------+----------
+---------+------------+
|               Date|   High|    Low|   Open|  Close|    Volume|Adj
Close|company_name|
+-------------------+-------+-------+-------+-------+----------
+---------+------------+
|2017-01-03 00:00:00|29.0825|  28.69|  28.95|29.0375|1.151276E8|
27.27764|       APPLE|
```

|2017-01-04 00:00:00|29.1275|28.9375|28.9625| 29.005| 8.44724E7|
27.247108| APPLE|
|2017-01-05 00:00:00| 29.215|28.9525| 28.98|29.1525| 8.87744E7|
27.385668| APPLE|
|2017-01-06 00:00:00| 29.54|29.1175| 29.195|29.4775|1.270076E8|
27.690971| APPLE|
|2017-01-09 00:00:00|29.8575| 29.485|29.4875|29.7475|1.342476E8|
27.944603| APPLE|
|2017-01-10 00:00:00| 29.845| 29.575|29.6925|29.7775| 9.78484E7|
27.972786| APPLE|
|2017-01-11 00:00:00|29.9825| 29.65| 29.685|29.9375|1.103544E8|
28.123089| APPLE|
|2017-01-12 00:00:00| 29.825|29.5525| 29.725|29.8125|1.083448E8|
28.005665| APPLE|
|2017-01-13 00:00:00| 29.905|29.7025|29.7775| 29.76|1.044476E8|
27.95635| APPLE|
|2017-01-17 00:00:00| 30.06| 29.555| 29.585| 30.0|1.377592E8|
28.1818| APPLE|
|2017-01-18 00:00:00| 30.125|29.9275| 30.0|29.9975| 9.4852E7|
28.179457| APPLE|
|2017-01-19 00:00:00|30.0225|29.8425| 29.85| 29.945|1.023892E8|
28.130133| APPLE|
|2017-01-20 00:00:00|30.1125|29.9325|30.1125| 30.0|1.303916E8|
28.1818| APPLE|
|2017-01-23 00:00:00|30.2025|29.9425| 30.0| 30.02| 8.82008E7|
28.20059| APPLE|
|2017-01-24 00:00:00| 30.025| 29.875|29.8875|29.9925| 9.2844E7|
28.174757| APPLE|
|2017-01-25 00:00:00| 30.525| 30.07| 30.105| 30.47|1.295104E8|
28.623314| APPLE|
|2017-01-26 00:00:00| 30.61| 30.4|30.4175| 30.485|1.053504E8|
28.637407| APPLE|
|2017-01-27 00:00:00|30.5875| 30.4| 30.535|30.4875| 8.22516E7|
28.639755| APPLE|
|2017-01-30 00:00:00|30.4075| 30.165|30.2325|30.4075| 1.2151E8|
28.564606| APPLE|
|2017-01-31 00:00:00|30.3475| 30.155|30.2875|30.3375| 1.96804E8|
28.498844| APPLE|
|2017-02-01 00:00:00|32.6225|31.7525|31.7575|32.1875| 4.4794E8|
30.23672| APPLE|
|2017-02-02 00:00:00|32.3475| 31.945| 31.995|32.1325|1.348416E8|
30.18506| APPLE|
|2017-02-03 00:00:00|32.2975| 32.04|32.0775| 32.27| 9.80292E7|
30.31423| APPLE|
|2017-02-06 00:00:00| 32.625| 32.225|32.2825|32.5725|1.073836E8|
30.598392| APPLE|
|2017-02-07 00:00:00|33.0225|32.6125| 32.635|32.8825|1.527352E8|
30.8896| APPLE|
|2017-02-08 00:00:00| 33.055| 32.805|32.8375| 33.01| 9.20164E7|
31.009373| APPLE|

```
|2017-02-09 00:00:00|33.1125|  32.78|32.9125| 33.105|1.133996E8|
31.233446|       APPLE|
|2017-02-10 00:00:00| 33.235|33.0125| 33.115|  33.03|  8.0262E7|
31.16269|       APPLE|
|2017-02-13 00:00:00| 33.455|33.1875|  33.27|33.3225| 9.21416E7|
31.438652|       APPLE|
|2017-02-14 00:00:00|33.7725|33.3125|33.3675| 33.755|1.329048E8|
31.846703|       APPLE|
|2017-02-15 00:00:00|34.0675| 33.655|  33.88|33.8775|1.424924E8|
31.962276|       APPLE|
|2017-02-16 00:00:00| 33.975|  33.71|33.9175|33.8375| 9.03384E7|
31.924547|       APPLE|
|2017-02-17 00:00:00|33.9575| 33.775| 33.775|  33.93| 8.87928E7|
32.01181|       APPLE|
|2017-02-21 00:00:00|34.1875| 33.995|34.0575| 34.175| 9.80288E7|
32.242958|       APPLE|
|2017-02-22 00:00:00|  34.28|34.0275|34.1075|34.2775| 8.33476E7|
32.339664|       APPLE|
|2017-02-23 00:00:00|  34.37| 34.075| 34.345|34.1325| 8.31528E7|
32.20287|       APPLE|
|2017-02-24 00:00:00| 34.165|  33.82|33.9775| 34.165| 8.71064E7|
32.233524|       APPLE|
|2017-02-27 00:00:00|  34.36|  34.07| 34.285|34.2325| 8.10296E7|
32.29721|       APPLE|
|2017-02-28 00:00:00|  34.36| 34.175|  34.27|34.2475| 9.39316E7|
32.31136|       APPLE|
|2017-03-01 00:00:00|35.0375|   34.4|34.4725|34.9475|1.456584E8|
32.97178|       APPLE|
+-------------------+-------+-------+-------+-------+----------
+---------+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels([''']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels([''']+columns)
```



Average open price and close price in 2017

avg(Open)	avg(Close)
37.61122511297583	37.637768870805836

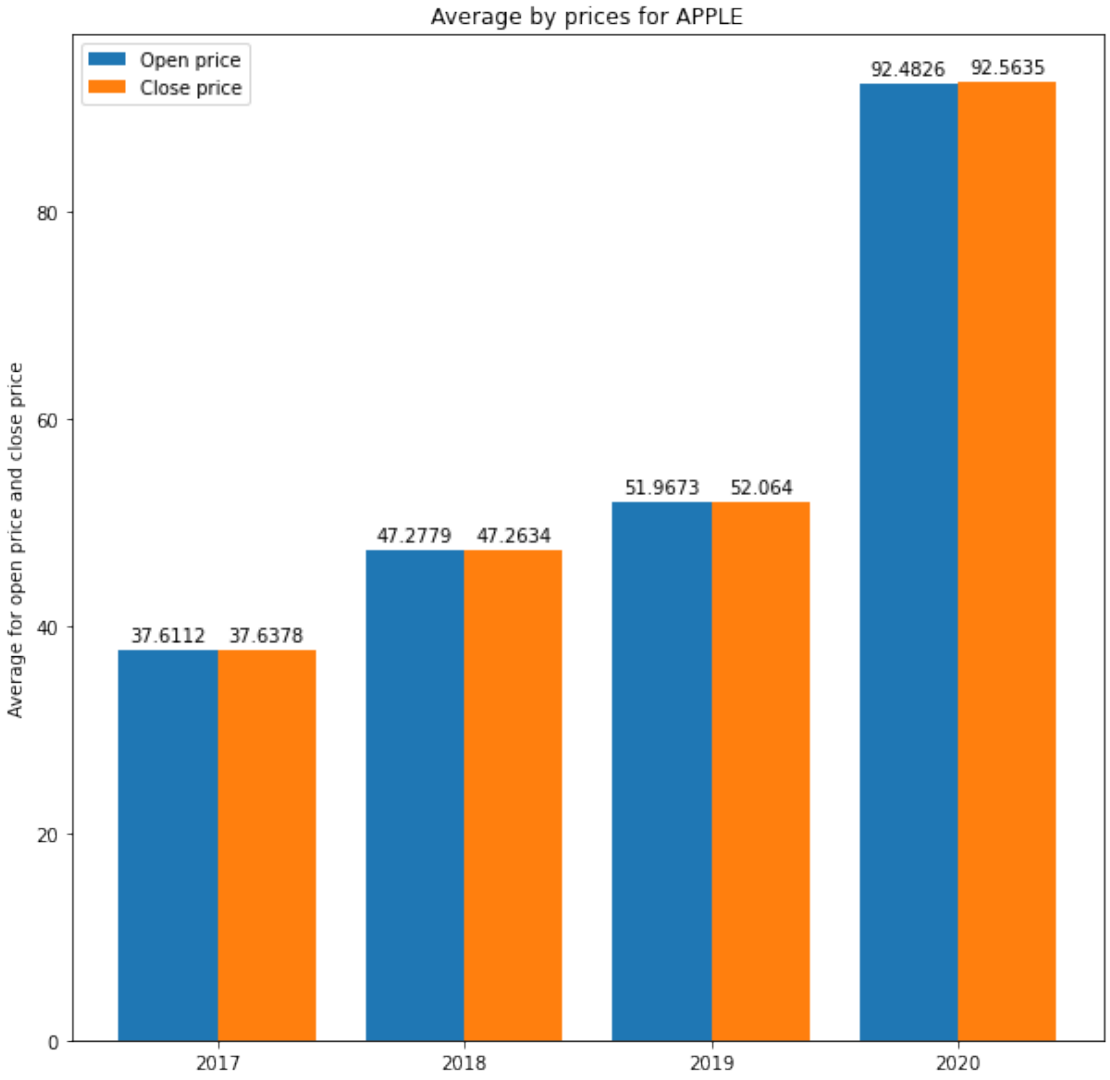
Average open price and close price in 2018

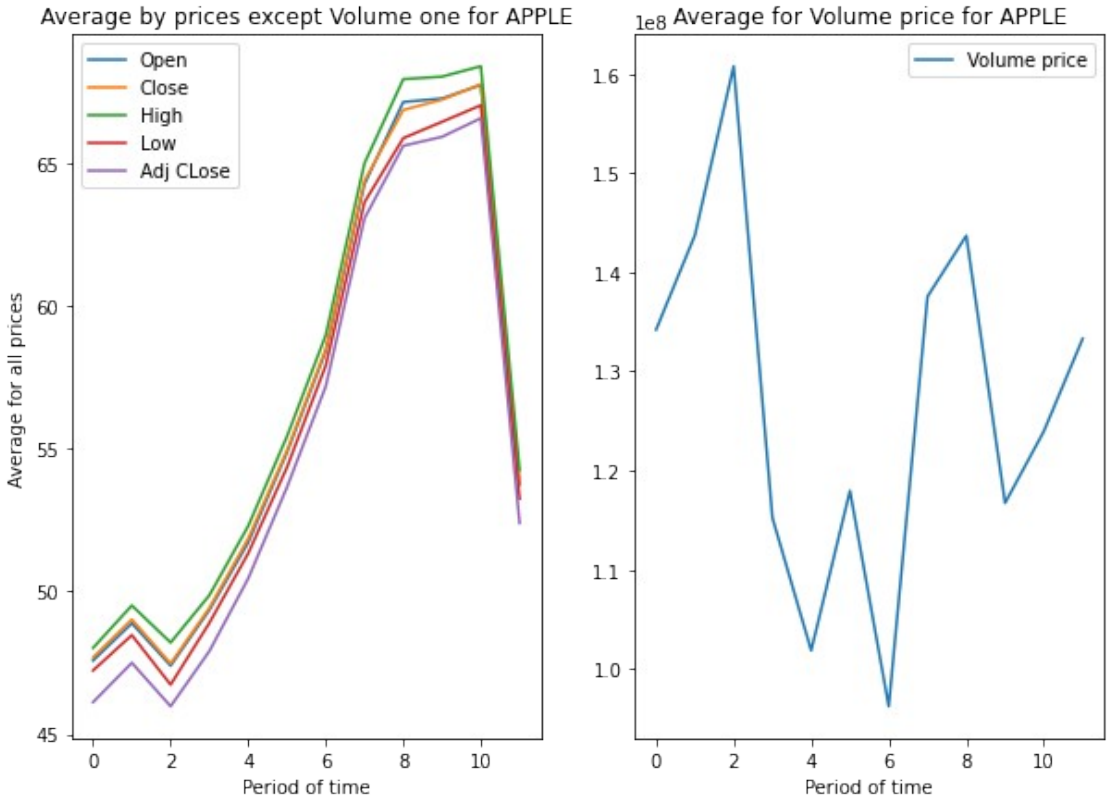
avg(Open)	avg(Close)
47.277858642942874	47.263356698936676

Average open price and close price in 2019

avg(Open)	avg(Close)

```
|51.96727168370807|52.063988049825035|
+-----------------+------------------+

Average open price and close price in 2020
+-----------------+-----------------+
|        avg(Open)|       avg(Close)|
+-----------------+-----------------+
|92.48257523237892|92.56351605198414|
+-----------------+-----------------+


Exception in thread "serve-DataFrame" java.net.SocketTimeoutException:
Accept timed out
      at java.base/java.net.PlainSocketImpl.socketAccept(Native Method)
      at
java.base/java.net.AbstractPlainSocketImpl.accept(AbstractPlainSocketI
mpl.java:474)
      at
java.base/java.net.ServerSocket.implAccept(ServerSocket.java:565)
      at java.base/java.net.ServerSocket.accept(ServerSocket.java:533)
      at org.apache.spark.security.SocketAuthServer$
$anon$1.run(SocketAuthServer.scala:64)
```

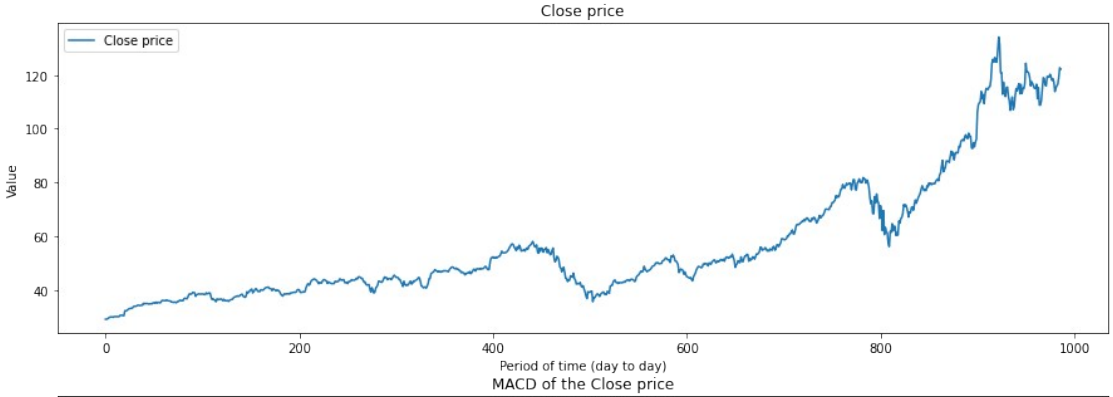

Average by prices for APPLE
Open price
Close price
Average for open price and close price
0
20
40
60
80
37.6112
37.6378
47.2779
47.2634
51.9673
52.064
92.4826
92.5635
2017
2018
2019
2020

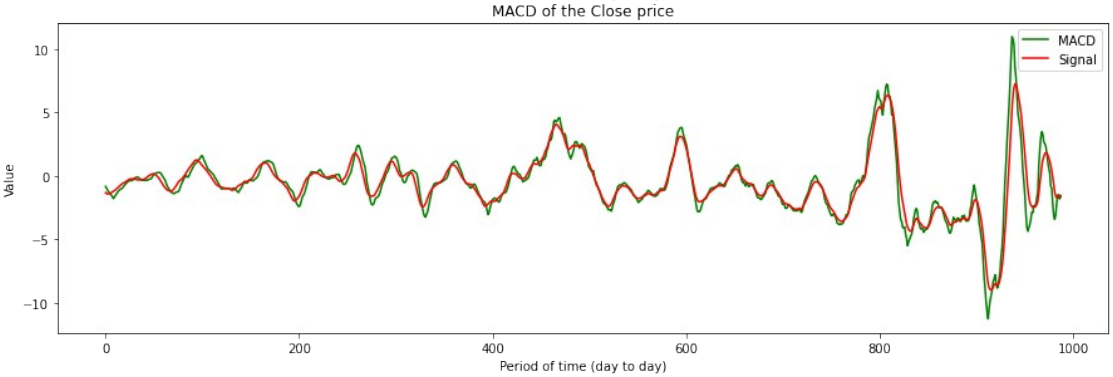


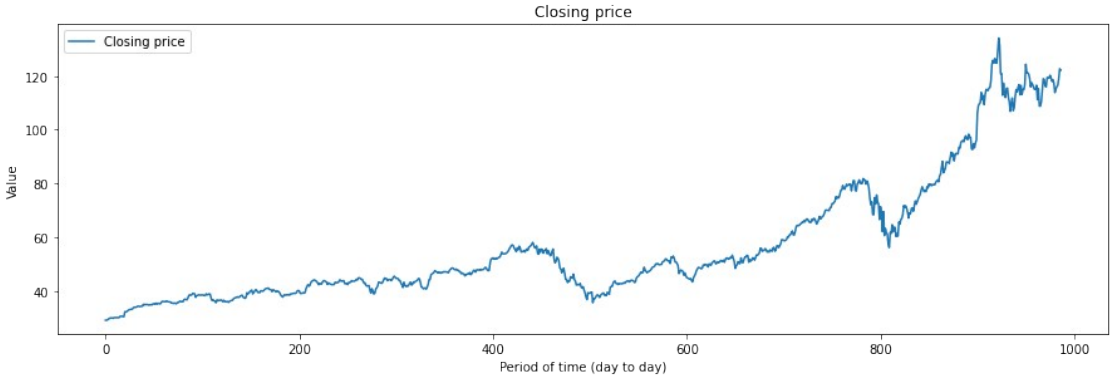

```
================
Visualization and plots for ZOOM
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+------+------+------+---------+---------
+------------+
|               Date|   High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+-------+------+------+------+---------+---------
+------------+
|2019-04-18 00:00:00|   66.0|60.321|  65.0|  62.0|2.57647E7|     62.0|
ZOOM|
|2019-04-22 00:00:00|   68.9| 59.94|  61.0|  65.7|9949700.0|     65.7|
ZOOM|
|2019-04-23 00:00:00| 74.169| 65.55| 66.87|  69.0|6786500.0|     69.0|
ZOOM|
|2019-04-24 00:00:00|   71.5| 63.16|  71.4|  63.2|4973500.0|     63.2|
```

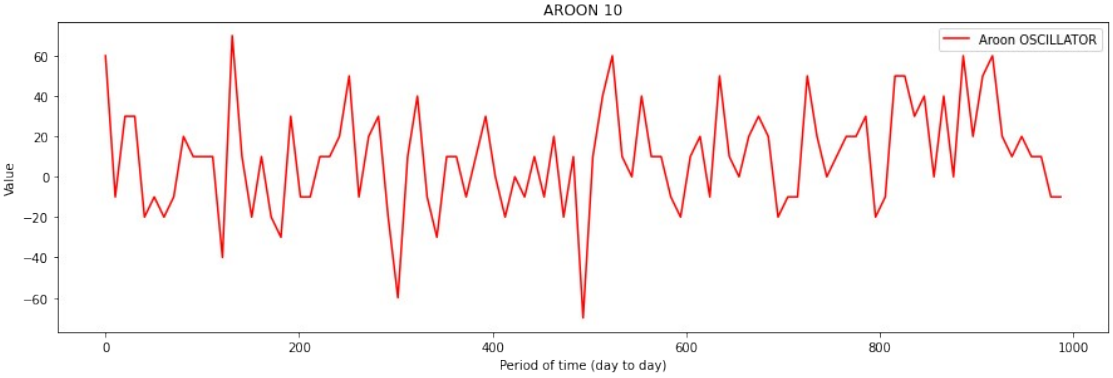
```
ZOOM|
|2019-04-25 00:00:00|  66.85|  62.6| 64.74|  65.0|3863300.0|     65.0|
ZOOM|
|2019-04-26 00:00:00|  66.99|  63.6| 66.12| 66.22|1527400.0|    66.22|
ZOOM|
|2019-04-29 00:00:00|   68.5| 64.75| 66.53| 68.17|1822300.0|    68.17|
ZOOM|
|2019-04-30 00:00:00|  72.52| 66.67|  68.4| 72.47|4113100.0|    72.47|
ZOOM|
|2019-05-01 00:00:00|  76.95|70.816| 72.72| 72.76|3301900.0|    72.76|
ZOOM|
|2019-05-02 00:00:00|  75.89|69.691| 72.75|  75.5|2525300.0|     75.5|
ZOOM|
|2019-05-03 00:00:00|  80.25|  75.0|  75.0| 79.18|2590300.0|    79.18|
ZOOM|
|2019-05-06 00:00:00|  80.79|  74.5| 75.01| 78.24|2051800.0|    78.24|
ZOOM|
|2019-05-07 00:00:00|  78.05| 73.25| 77.85| 73.33|1975200.0|    73.33|
ZOOM|
|2019-05-08 00:00:00|   78.5| 74.03| 74.61| 77.68|2265500.0|    77.68|
ZOOM|
|2019-05-09 00:00:00|  76.99|  74.0| 76.85| 75.21|1348200.0|    75.21|
ZOOM|
|2019-05-10 00:00:00|  79.74| 74.77| 75.79| 79.63|1555100.0|    79.63|
ZOOM|
|2019-05-13 00:00:00|  77.39|  70.6| 77.39| 72.54|2873200.0|    72.54|
ZOOM|
|2019-05-14 00:00:00| 76.885| 73.11| 74.12| 73.14|1950400.0|    73.14|
ZOOM|
|2019-05-15 00:00:00|   80.0| 72.21|  73.4| 79.76|2426500.0|    79.76|
ZOOM|
|2019-05-16 00:00:00|  87.55| 79.25| 80.12|  83.4|4580700.0|     83.4|
ZOOM|
|2019-05-17 00:00:00|  90.28| 81.88| 82.25| 89.98|3442500.0|    89.98|
ZOOM|
|2019-05-20 00:00:00|  91.46| 83.27|  90.1| 84.67|3666800.0|    84.67|
ZOOM|
|2019-05-21 00:00:00|   89.7|  84.5| 86.63| 85.44|2576000.0|    85.44|
ZOOM|
|2019-05-22 00:00:00|   85.7|  82.0| 84.63| 82.43|1596400.0|    82.43|
ZOOM|
|2019-05-23 00:00:00|   81.5| 77.26| 81.25| 78.76|2856000.0|    78.76|
ZOOM|
|2019-05-24 00:00:00|  81.25|  74.2| 80.48| 76.25|2946800.0|    76.25|
ZOOM|
|2019-05-28 00:00:00|  79.39|  76.8|  76.8| 77.77|1641300.0|    77.77|
ZOOM|
|2019-05-29 00:00:00|  77.93|73.583|  77.0| 75.77|1512100.0|    75.77|
ZOOM|
|2019-05-30 00:00:00|  80.97|  76.6| 76.68| 80.42|1996000.0|    80.42|
```

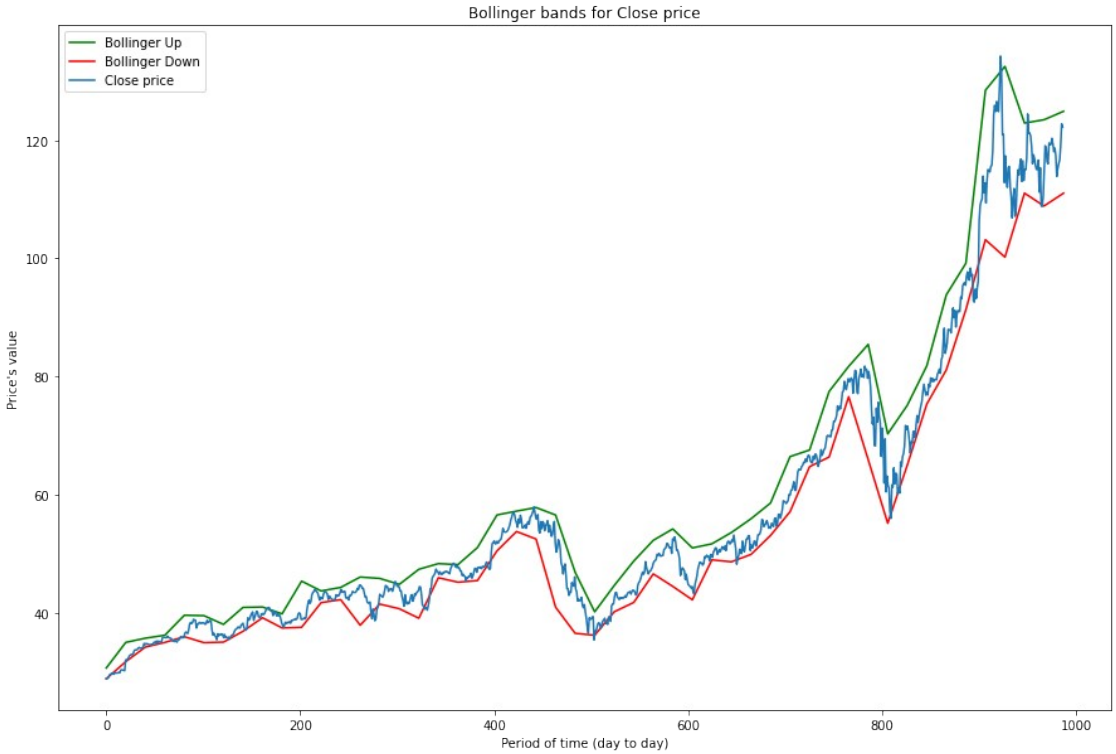
```
ZOOM|
|2019-05-31 00:00:00|  83.17| 77.78| 78.77| 79.73|1594300.0|    79.73|
ZOOM|
|2019-06-03 00:00:00|  81.94| 75.65|  80.0|  75.9|1570500.0|     75.9|
ZOOM|
|2019-06-04 00:00:00|  78.88| 76.62|  78.2| 78.74|1134900.0|    78.74|
ZOOM|
|2019-06-05 00:00:00|   80.6| 76.24| 80.14| 78.04|1295800.0|    78.04|
ZOOM|
|2019-06-06 00:00:00|  79.75| 77.03|  77.4| 79.43|3024000.0|    79.43|
ZOOM|
|2019-06-07 00:00:00|  98.89|  92.5| 93.66| 94.05|9487800.0|    94.05|
ZOOM|
|2019-06-10 00:00:00|105.985|  96.0| 98.51| 102.0|4852800.0|    102.0|
ZOOM|
|2019-06-11 00:00:00|  101.2| 91.57| 101.0| 94.87|4372400.0|    94.87|
ZOOM|
|2019-06-12 00:00:00|104.185|  94.0|  94.6|102.77|3151700.0|   102.77|
ZOOM|
|2019-06-13 00:00:00|105.172| 98.55| 105.1|100.95|3189100.0|   100.95|
ZOOM|
|2019-06-14 00:00:00| 104.57| 99.25|100.47|100.29|1889300.0|   100.29|
ZOOM|
+-------------------+-------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 411
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```

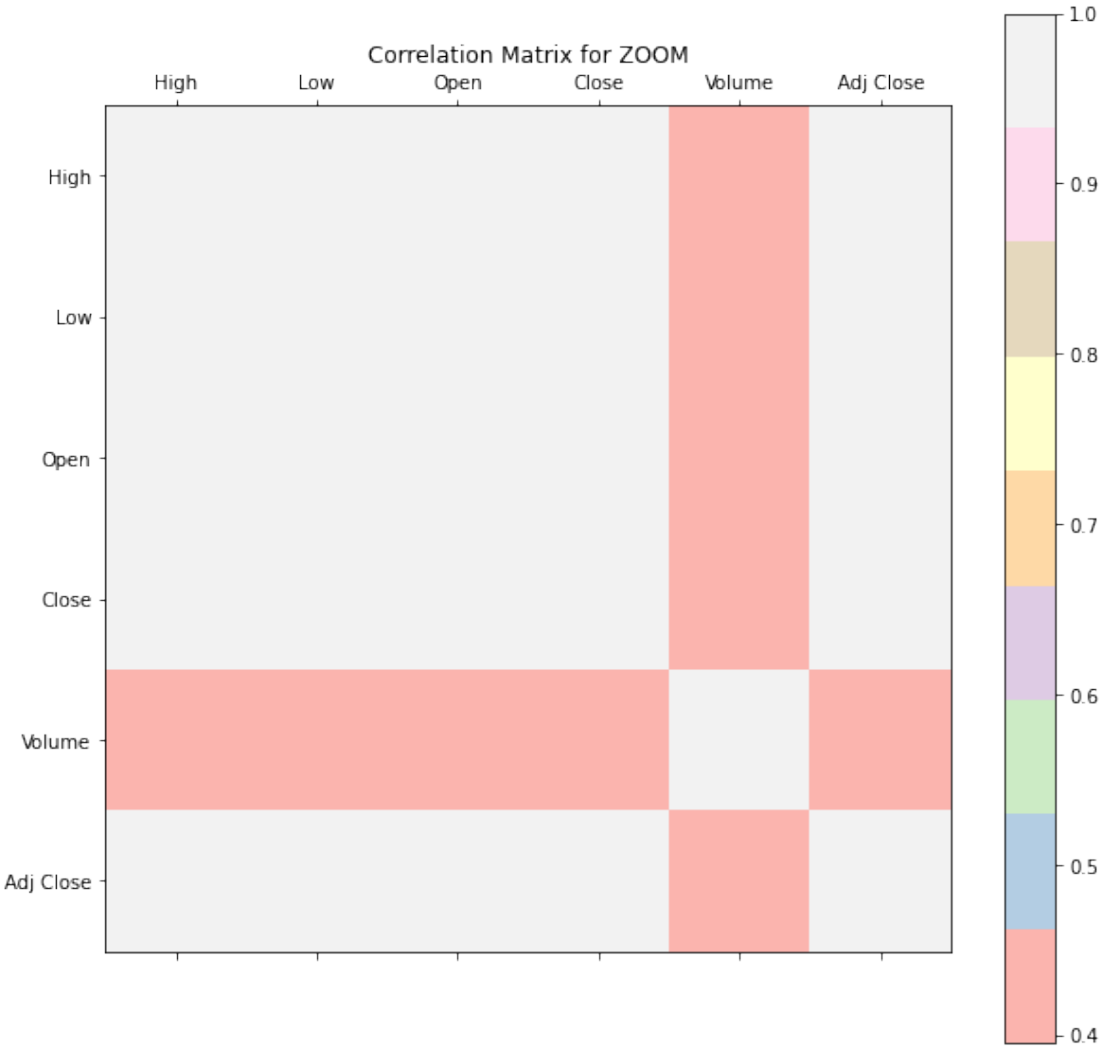

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
120
100
80
60
40
0
200
400
600
800
1000

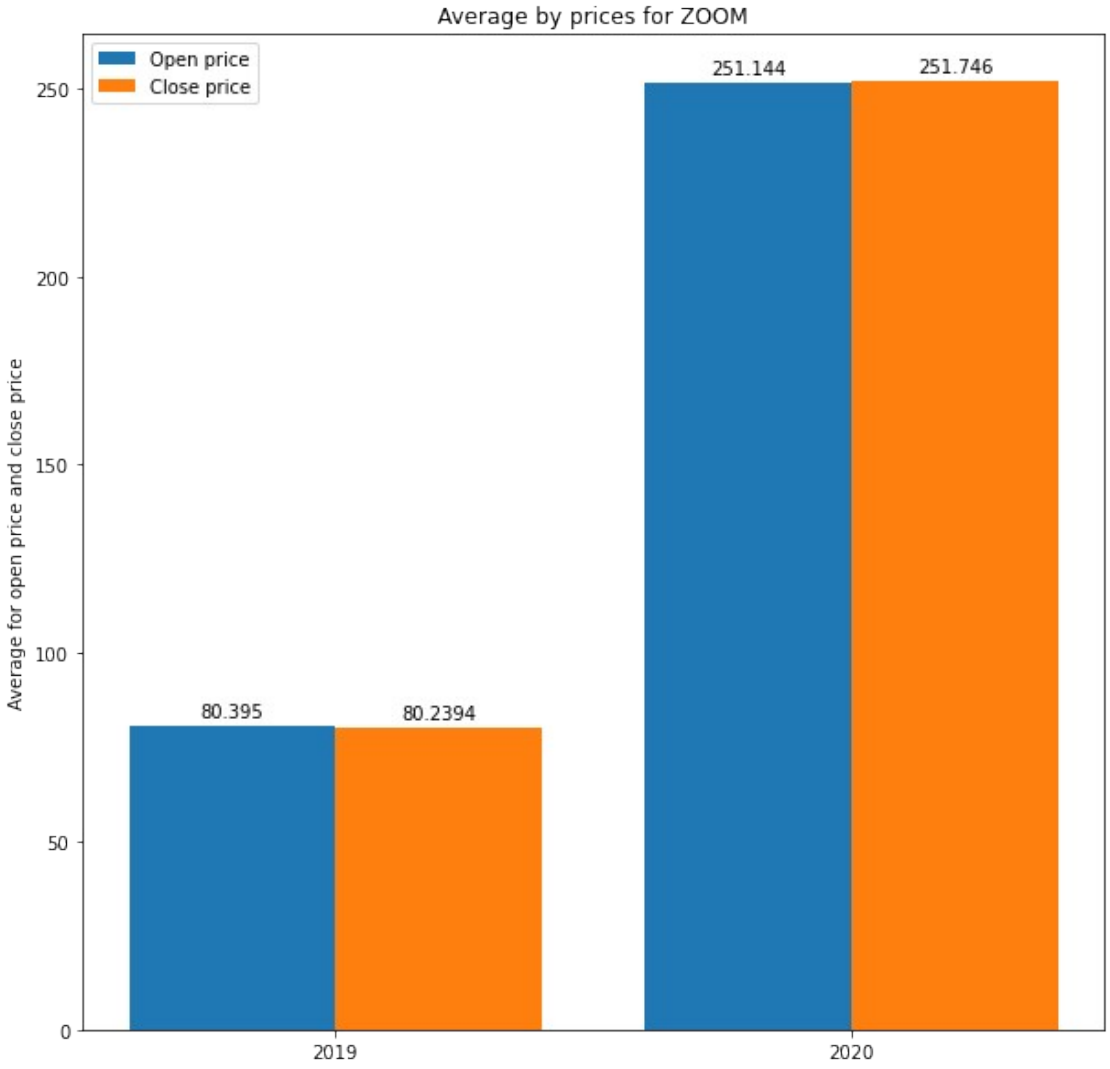


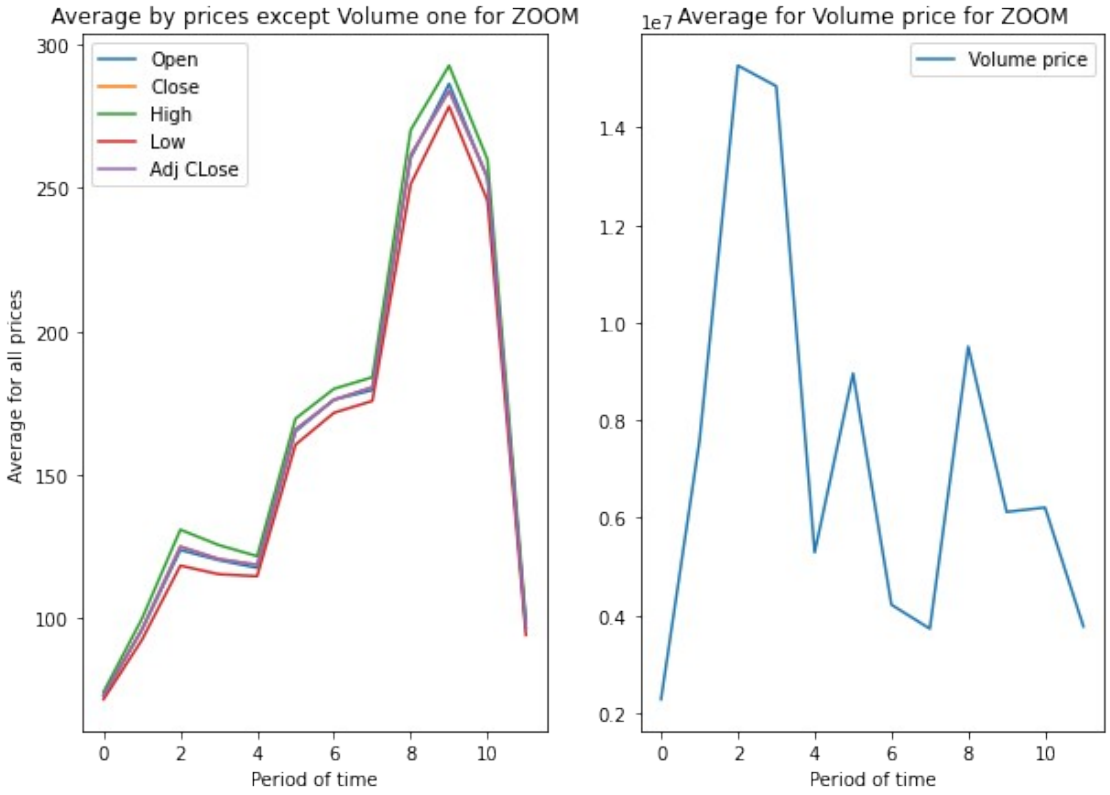
Average open price and close price in 2019

avg(Open)	avg(Close)
80.39499985769893	80.23938206876262

Average open price and close price in 2020

avg(Open)	avg(Close)
251.14446795549514	251.74615088859852

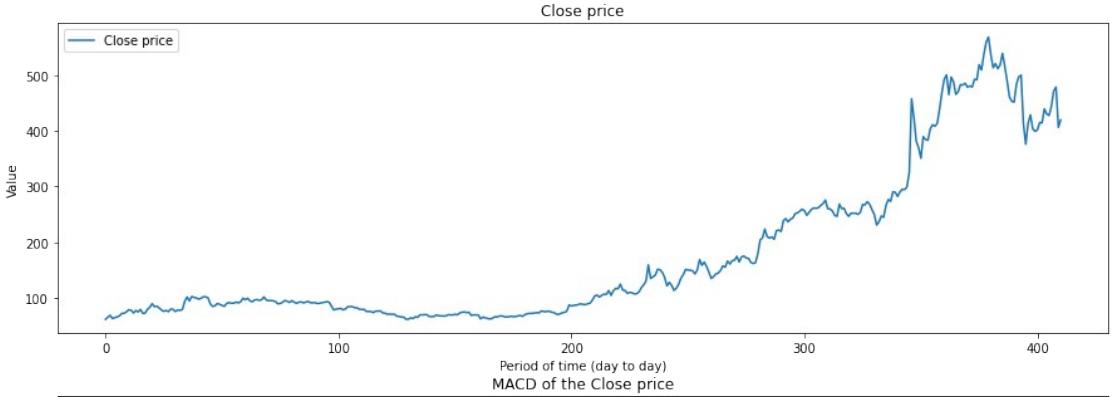

Average by prices for ZOOM
Open price
Close price
Average for open price and close price
0
50
100
150
200
250
80.395
80.2394
251.144
251.746
2019
2020

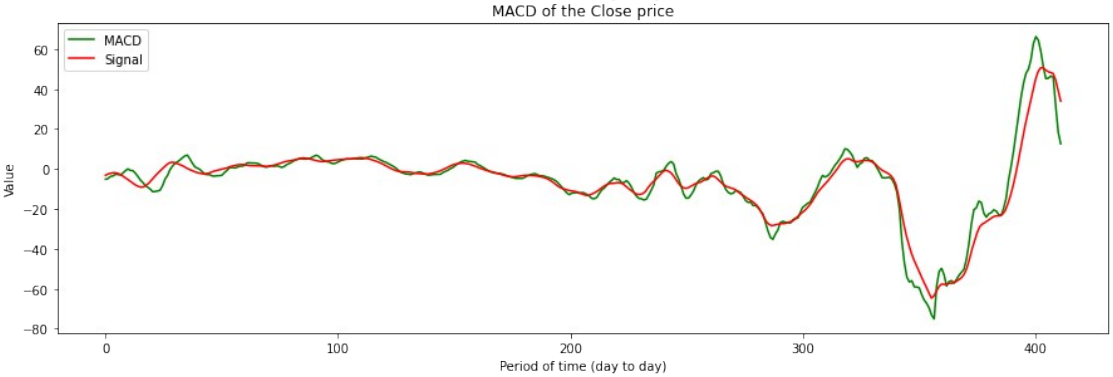


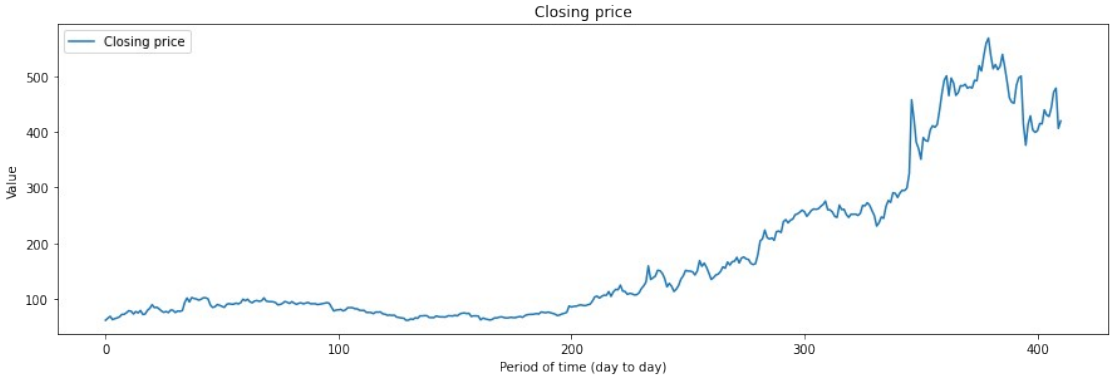

```
================
Visualization and plots for MICROSOFT
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-----+-----+-----+-----+---------+---------
+------------+
|               Date| High|  Low| Open|Close|   Volume|Adj Close|
company_name|
+-------------------+-----+-----+-----+-----+---------+---------
+------------+
|2017-01-03 00:00:00|62.84|62.13|62.79|62.58|2.06941E7|58.673244|
MICROSOFT|
|2017-01-04 00:00:00|62.75|62.12|62.48| 62.3|  2.134E7|58.410725|
MICROSOFT|
|2017-01-05 00:00:00|62.66|62.03|62.19| 62.3| 2.4876E7|58.410725|
MICROSOFT|
|2017-01-06 00:00:00|63.15|62.04| 62.3|62.84|1.99229E7|58.917015|
MICROSOFT|
```

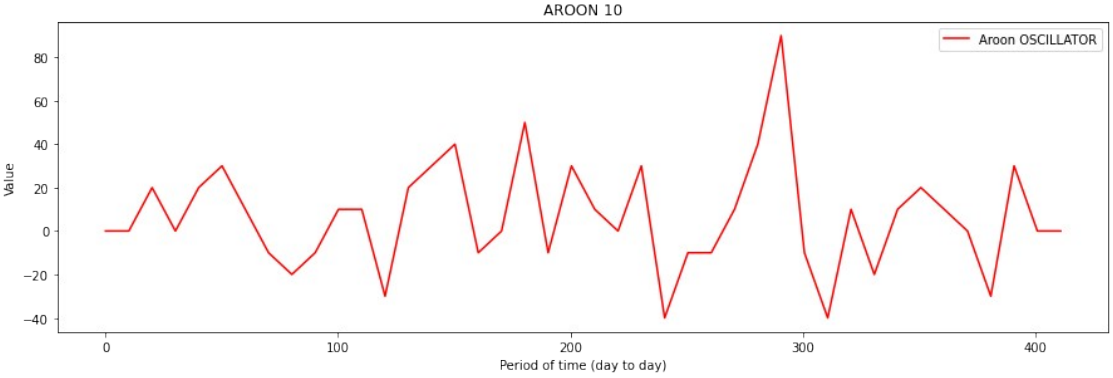
|2017-01-09 00:00:00|63.08|62.54|62.76|62.64|2.03827E7|58.729496|
MICROSOFT|
|2017-01-10 00:00:00|63.07|62.28|62.73|62.62| 1.8593E7|58.710747|
MICROSOFT|
|2017-01-11 00:00:00|63.23|62.43|62.61|63.19|2.15173E7| 59.24516|
MICROSOFT|
|2017-01-12 00:00:00| 63.4|61.95|63.06|62.61|2.09682E7| 58.70137|
MICROSOFT|
|2017-01-13 00:00:00|62.87|62.35|62.62| 62.7|1.94223E7|58.785755|
MICROSOFT|
|2017-01-17 00:00:00| 62.7|62.03|62.68|62.53| 2.0664E7| 58.62637|
MICROSOFT|
|2017-01-18 00:00:00| 62.7|62.12|62.67| 62.5|1.96701E7|58.598248|
MICROSOFT|
|2017-01-19 00:00:00|62.98| 62.2|62.24| 62.3|1.84517E7|58.410725|
MICROSOFT|
|2017-01-20 00:00:00|62.82|62.37|62.67|62.74|3.02135E7| 58.82326|
MICROSOFT|
|2017-01-23 00:00:00|63.12|62.57| 62.7|62.96|2.30976E7|59.029526|
MICROSOFT|
|2017-01-24 00:00:00|63.74|62.94| 63.2|63.52|2.46729E7| 59.55457|
MICROSOFT|
|2017-01-25 00:00:00| 64.1|63.45|63.95|63.68|2.36727E7|59.704575|
MICROSOFT|
|2017-01-26 00:00:00|64.54|63.55|64.12|64.27|4.35546E7|60.257736|
MICROSOFT|
|2017-01-27 00:00:00|65.91|64.89|65.39|65.78| 4.4818E7| 61.67348|
MICROSOFT|
|2017-01-30 00:00:00|65.79| 64.8|65.69|65.13|3.16514E7|61.064045|
MICROSOFT|
|2017-01-31 00:00:00|65.15|64.26|64.86|64.65|2.52705E7| 60.61402|
MICROSOFT|
|2017-02-01 00:00:00|64.62|63.47|64.36|63.58|3.96715E7|59.610825|
MICROSOFT|
|2017-02-02 00:00:00|63.41|62.75|63.25|63.17| 4.5827E7|59.226406|
MICROSOFT|
|2017-02-03 00:00:00| 63.7|63.07| 63.5|63.68|3.03018E7|59.704575|
MICROSOFT|
|2017-02-06 00:00:00|63.65|63.14| 63.5|63.64|1.97964E7| 59.66707|
MICROSOFT|
|2017-02-07 00:00:00|63.78|63.23|63.74|63.43|2.02772E7| 59.47018|
MICROSOFT|
|2017-02-08 00:00:00|63.81|63.22|63.57|63.34|1.80964E7| 59.3858|
MICROSOFT|
|2017-02-09 00:00:00|64.44|63.32|63.52|64.06|2.26444E7| 60.06085|
MICROSOFT|
|2017-02-10 00:00:00| 64.3|63.98|64.25| 64.0|1.81707E7|60.004604|
MICROSOFT|
|2017-02-13 00:00:00|64.86|64.13|64.24|64.72|2.29201E7|60.679653|
MICROSOFT|

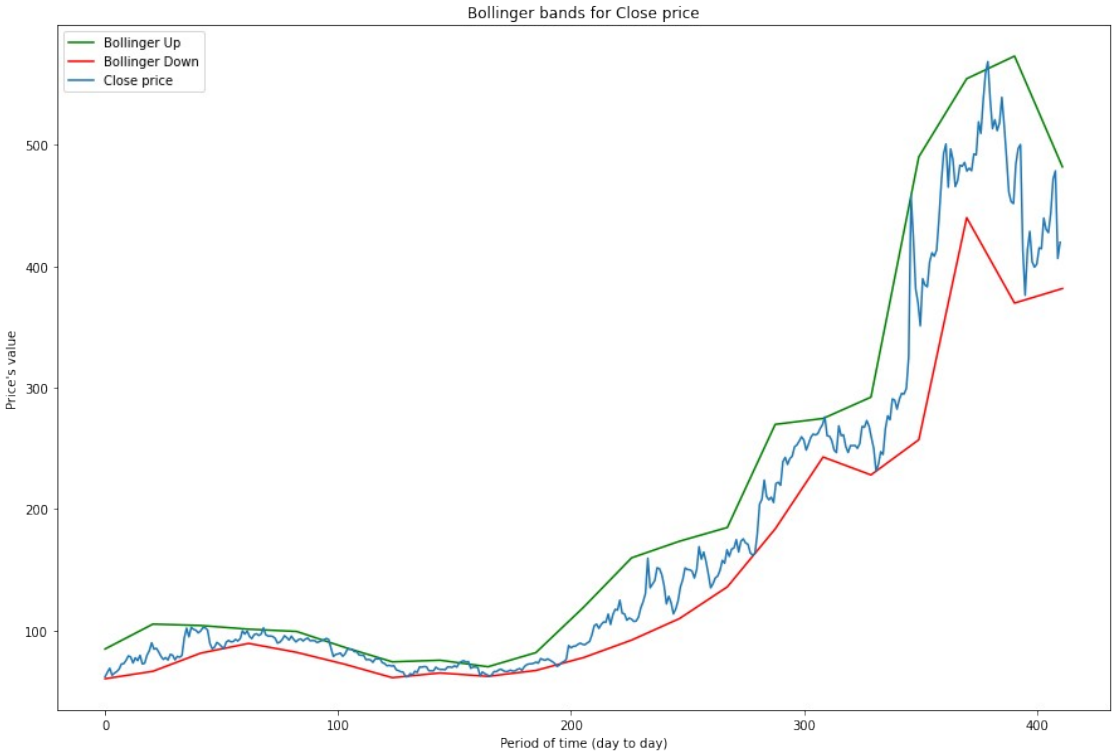
```
|2017-02-14 00:00:00|64.72|64.02|64.41|64.57|2.31084E7|60.906044|
MICROSOFT|
|2017-02-15 00:00:00|64.57|64.16| 64.5|64.53|1.70052E7|60.868298|
MICROSOFT|
|2017-02-16 00:00:00|65.24|64.44|64.74|64.52|2.05463E7| 60.85886|
MICROSOFT|
|2017-02-17 00:00:00|64.69| 64.3|64.47|64.62|2.12488E7|60.953197|
MICROSOFT|
|2017-02-21 00:00:00|64.95|64.45|64.61|64.49|2.06559E7| 60.83057|
MICROSOFT|
|2017-02-22 00:00:00|64.39|64.05|64.33|64.36|1.92927E7|60.707947|
MICROSOFT|
|2017-02-23 00:00:00|64.73|64.19|64.42|64.62|2.02731E7|60.953197|
MICROSOFT|
|2017-02-24 00:00:00| 64.8|64.14|64.53|64.62|2.17968E7|60.953197|
MICROSOFT|
|2017-02-27 00:00:00|64.54|64.05|64.54|64.23|1.58715E7| 60.58533|
MICROSOFT|
|2017-02-28 00:00:00| 64.2|63.76|64.08|63.98|2.32398E7|60.349503|
MICROSOFT|
|2017-03-01 00:00:00|64.99|64.02|64.13|64.94|2.69375E7| 61.25505|
MICROSOFT|
+-------------------+-----+-----+-----+-----+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```

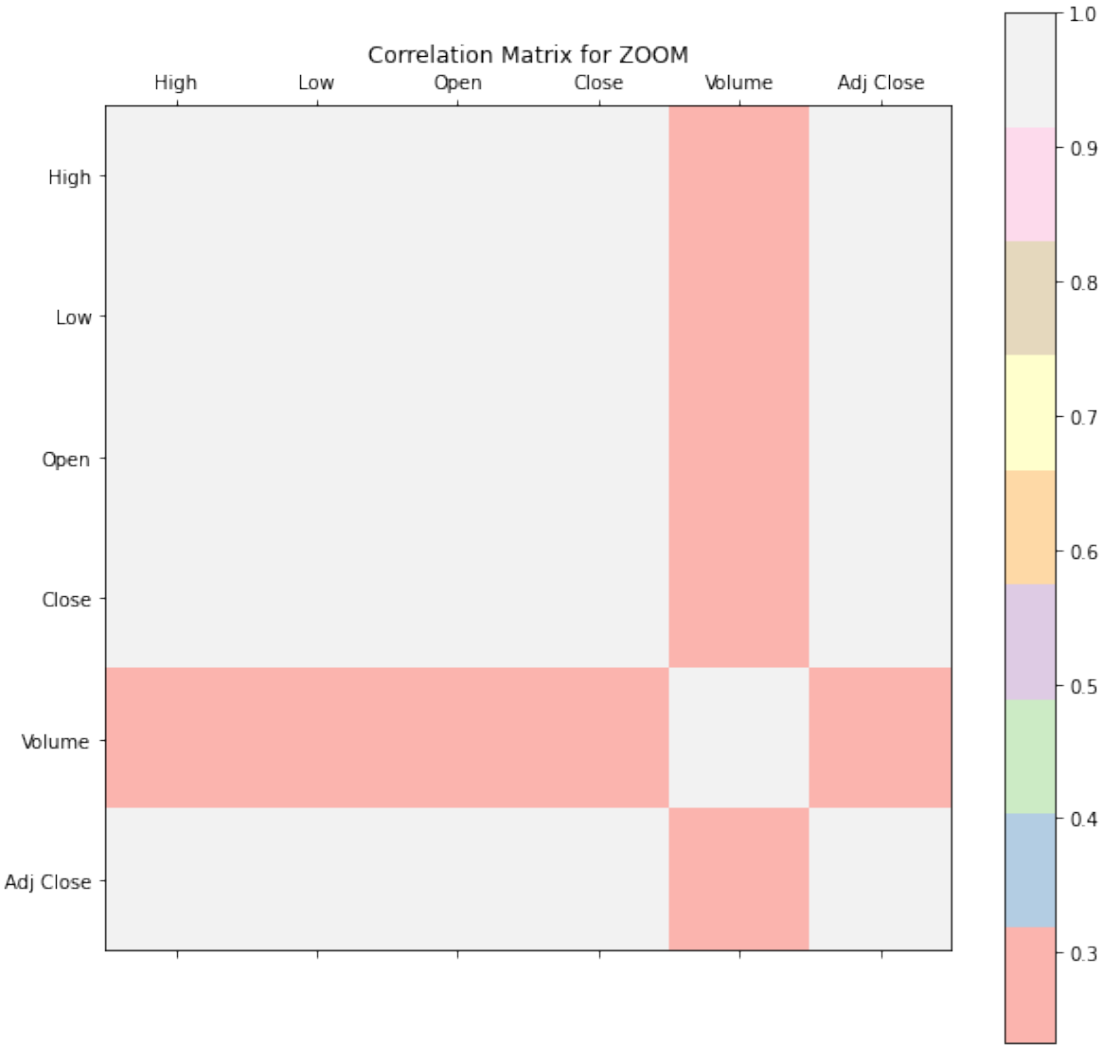

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
100
200
300
400
500
0
100
200
300
400



Average open price and close price in 2017

avg(Open)	avg(Close)
71.95430287516925	71.98402421502954

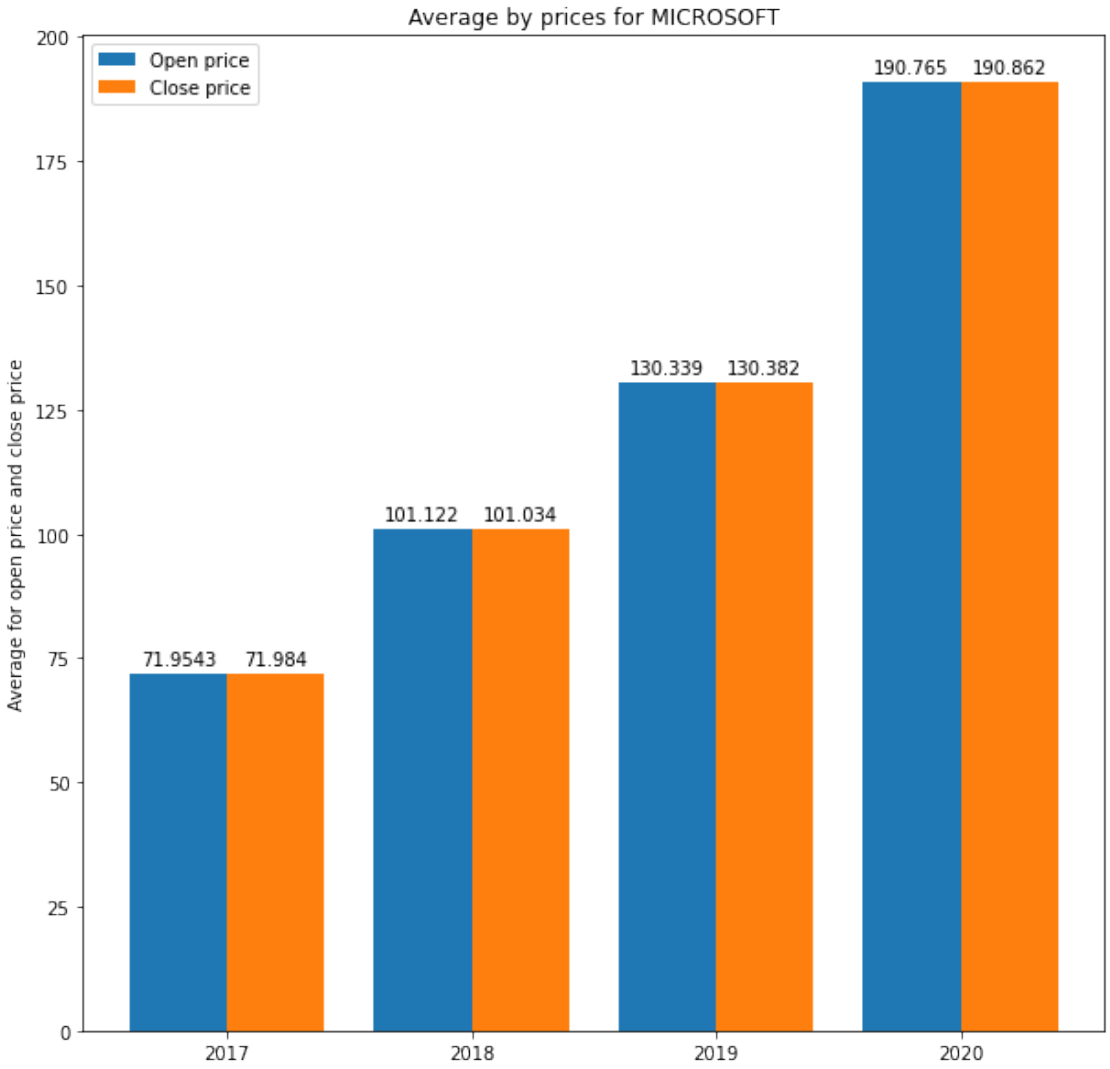
Average open price and close price in 2018

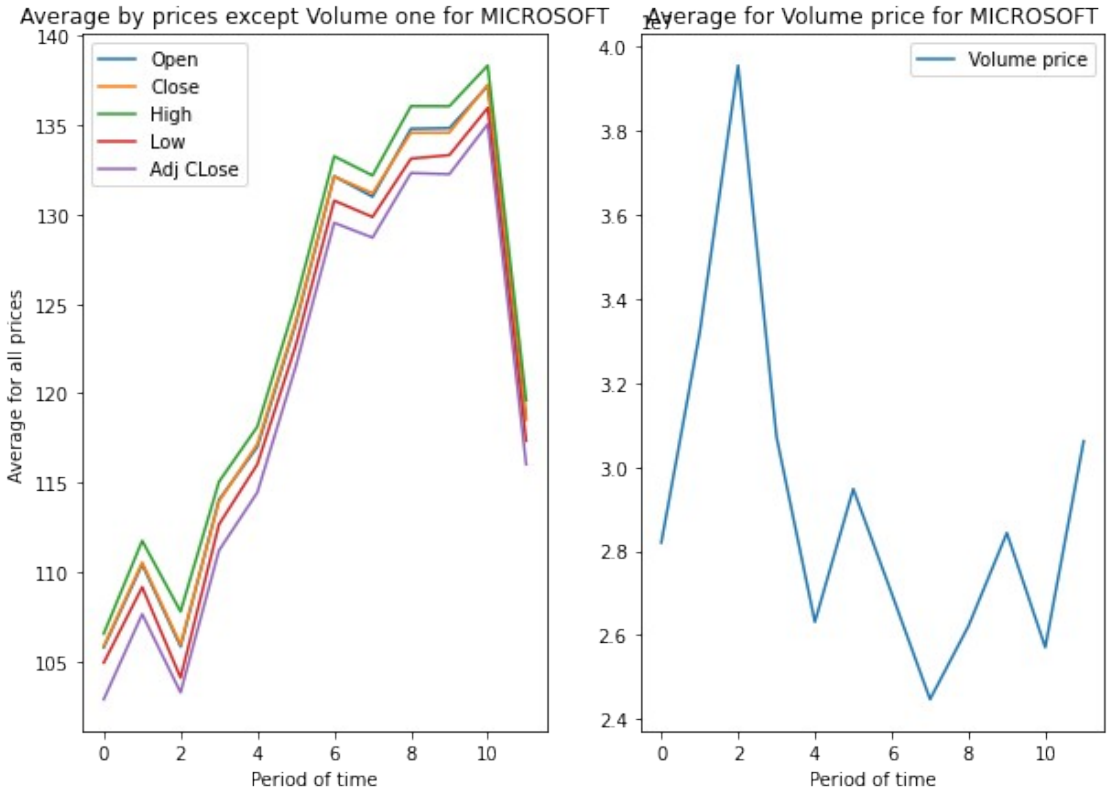
avg(Open)	avg(Close)
101.12235092831799	101.03398411967365

Average open price and close price in 2019

avg(Open)	avg(Close)

```
|130.33904787093874|130.38202400813026|
+------------------+------------------+

Average open price and close price in 2020
+------------------+------------------+
|         avg(Open)|       avg(Close)|
+------------------+------------------+
|190.76480678836674|190.8616180419922|
+------------------+------------------+
```

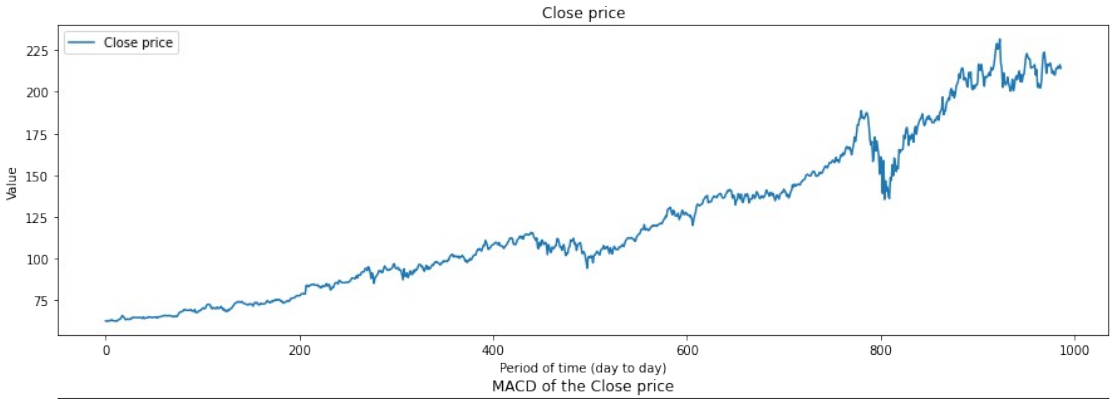


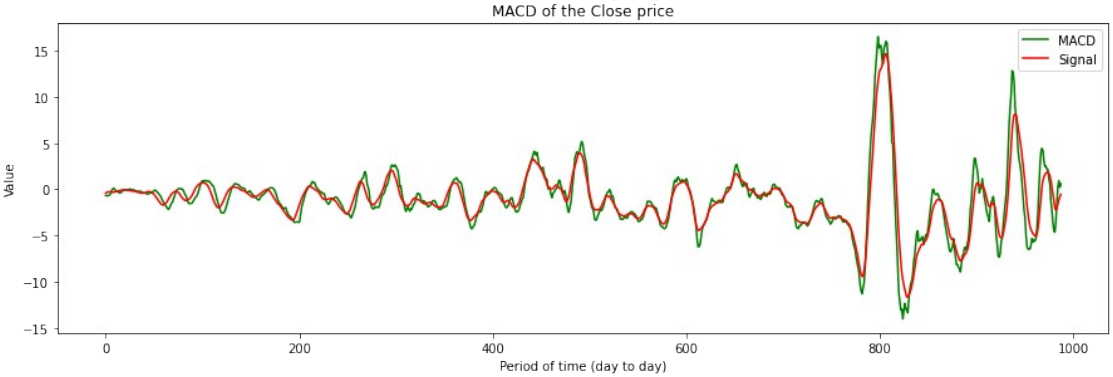


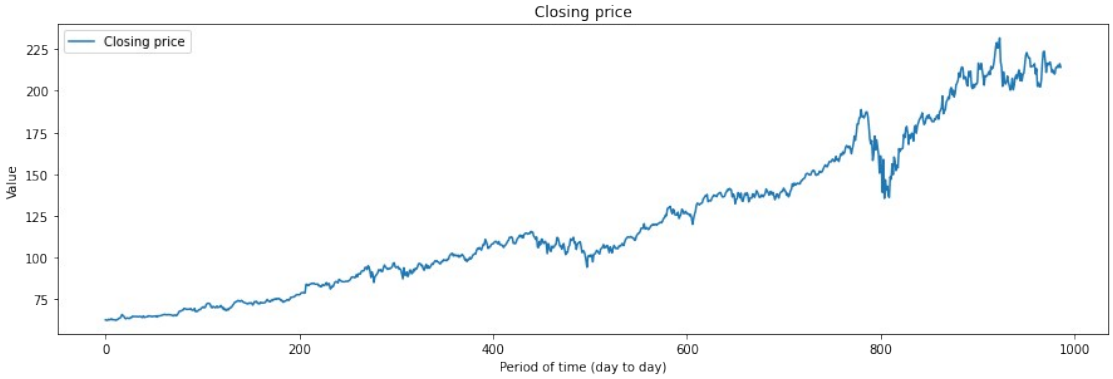

```
================
Visualization and plots for FACEBOOK
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
+------------+
|2017-01-03 00:00:00|117.84|115.51|116.03|116.86|2.06639E7|   116.86|
FACEBOOK|
|2017-01-04 00:00:00|119.66|117.29|117.55|118.69|1.96309E7|   118.69|
FACEBOOK|
|2017-01-05 00:00:00|120.95|118.32|118.86|120.67|1.94922E7|   120.67|
FACEBOOK|
|2017-01-06 00:00:00|123.88|120.03|120.98|123.41|2.85453E7|   123.41|
FACEBOOK|
```

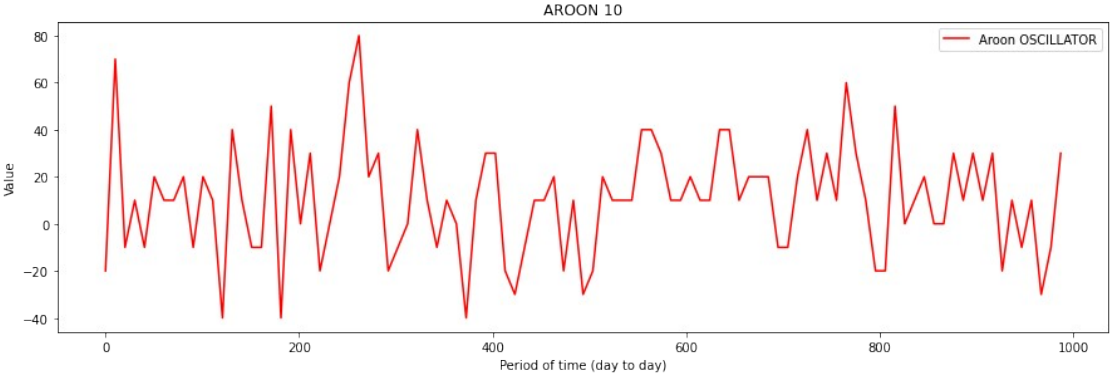
|2017-01-09 00:00:00|125.43|123.04|123.55| 124.9|2.28804E7| 124.9|
FACEBOOK|
|2017-01-10 00:00:00| 125.5|124.28|124.82|124.35|1.73246E7| 124.35|
FACEBOOK|
|2017-01-11 00:00:00|126.12|124.06|124.35|126.09|1.83565E7| 126.09|
FACEBOOK|
|2017-01-12 00:00:00|126.73| 124.8|125.61|126.62|1.86539E7| 126.62|
FACEBOOK|
|2017-01-13 00:00:00|129.27|127.37|127.49|128.34|2.48843E7| 128.34|
FACEBOOK|
|2017-01-17 00:00:00|128.34| 127.4|128.04|127.87|1.52945E7| 127.87|
FACEBOOK|
|2017-01-18 00:00:00|128.43|126.84|128.41|127.92|1.31459E7| 127.92|
FACEBOOK|
|2017-01-19 00:00:00|128.35|127.45|128.23|127.55|1.21955E7| 127.55|
FACEBOOK|
|2017-01-20 00:00:00|128.48|126.78| 128.1|127.04|1.90972E7| 127.04|
FACEBOOK|
|2017-01-23 00:00:00|129.25|126.95|127.31|128.93|1.65936E7| 128.93|
FACEBOOK|
|2017-01-24 00:00:00| 129.9|128.38|129.38|129.37|1.51627E7| 129.37|
FACEBOOK|
|2017-01-25 00:00:00|131.74|129.77| 130.0|131.48|1.87313E7| 131.48|
FACEBOOK|
|2017-01-26 00:00:00|133.14|131.44|131.63|132.78|2.00201E7| 132.78|
FACEBOOK|
|2017-01-27 00:00:00|132.95|131.08|132.68|132.18|1.95395E7| 132.18|
FACEBOOK|
|2017-01-30 00:00:00|131.58| 129.6|131.58|130.98|1.89561E7| 130.98|
FACEBOOK|
|2017-01-31 00:00:00|130.66|129.52|130.17|130.32|1.97905E7| 130.32|
FACEBOOK|
|2017-02-01 00:00:00|133.49|130.68|132.25|133.23|5.01398E7| 133.23|
FACEBOOK|
|2017-02-02 00:00:00|135.49| 130.4|133.22|130.84|5.43664E7| 130.84|
FACEBOOK|
|2017-02-03 00:00:00|132.85|130.76|131.24|130.98|2.48049E7| 130.98|
FACEBOOK|
|2017-02-06 00:00:00|132.06| 130.3|130.98|132.06|1.70585E7| 132.06|
FACEBOOK|
|2017-02-07 00:00:00| 133.0|131.66|132.24|131.84|1.45964E7| 131.84|
FACEBOOK|
|2017-02-08 00:00:00|134.44|132.44| 132.6| 134.2|2.23906E7| 134.2|
FACEBOOK|
|2017-02-09 00:00:00| 134.5|133.31|134.49|134.14|1.64706E7| 134.14|
FACEBOOK|
|2017-02-10 00:00:00|134.94|133.68| 134.1|134.19|1.50619E7| 134.19|
FACEBOOK|
|2017-02-13 00:00:00| 134.7| 133.7| 134.7|134.05|1.35262E7| 134.05|
FACEBOOK|

```
|2017-02-14 00:00:00|134.23|132.55| 134.1|133.85|1.43649E7|   133.85|
FACEBOOK|
|2017-02-15 00:00:00| 133.7|132.66|133.45|133.44|1.32265E7|   133.44|
FACEBOOK|
|2017-02-16 00:00:00|133.87|133.02|133.07|133.84|1.28311E7|   133.84|
FACEBOOK|
|2017-02-17 00:00:00|134.09|133.17| 133.5|133.53|1.22765E7|   133.53|
FACEBOOK|
|2017-02-21 00:00:00|133.91| 132.9| 133.5|133.72|1.47591E7|   133.72|
FACEBOOK|
|2017-02-22 00:00:00|136.79|133.46| 133.6|136.12|2.73601E7|   136.12|
FACEBOOK|
|2017-02-23 00:00:00|136.12|134.33|135.89|135.36|1.84225E7|   135.36|
FACEBOOK|
|2017-02-24 00:00:00|135.62|134.16|134.16|135.44|1.26257E7|   135.44|
FACEBOOK|
|2017-02-27 00:00:00|137.18|135.02|135.26|136.41|1.43067E7|   136.41|
FACEBOOK|
|2017-02-28 00:00:00|136.81|134.75|136.79|135.54|1.61121E7|   135.54|
FACEBOOK|
|2017-03-01 00:00:00|137.48| 136.3|136.47|137.42| 1.6257E7|   137.42|
FACEBOOK|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```

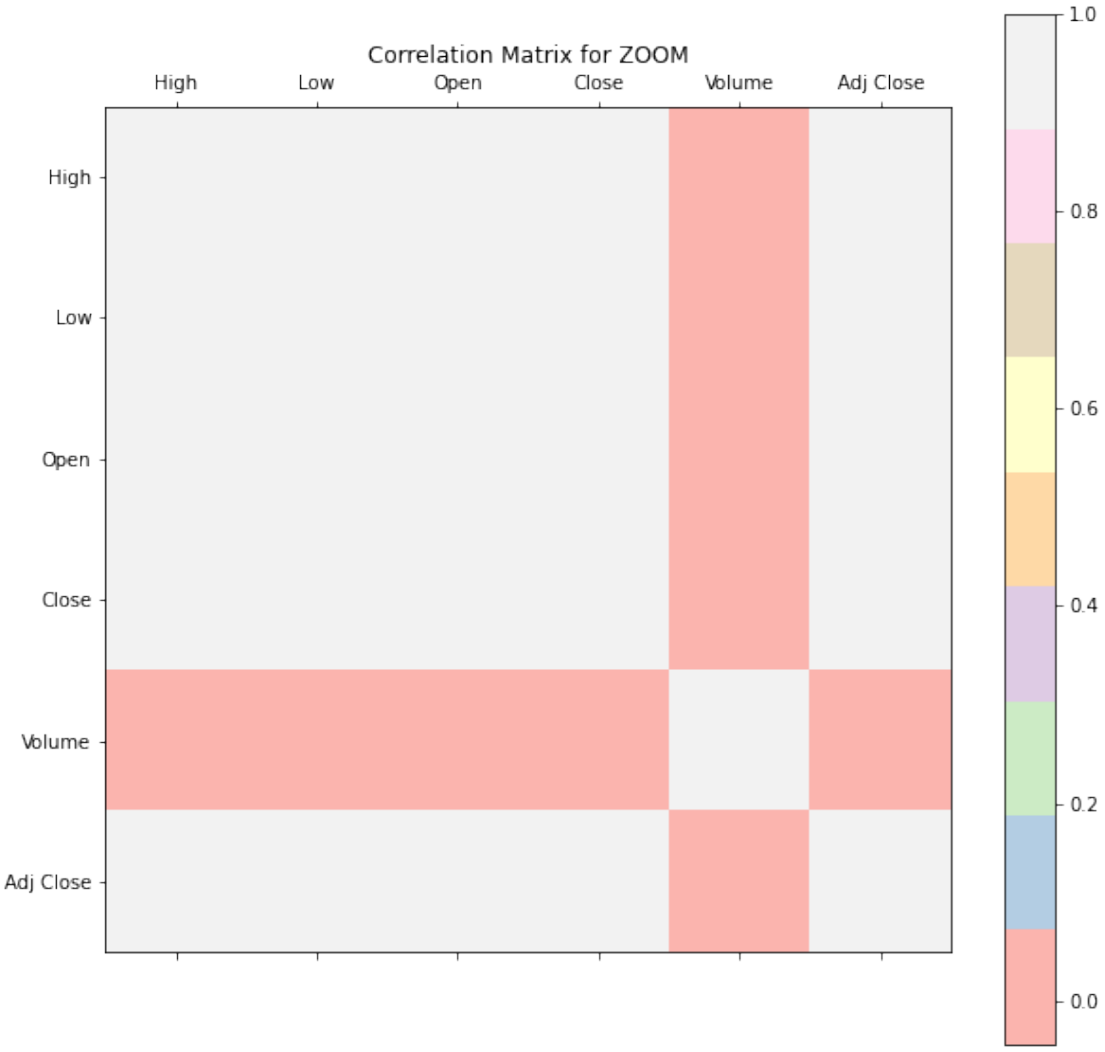

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
225
200
175
150
125
100
75
0
200
400
600
800
1000



Average open price and close price in 2017

avg(Open)	avg(Close)
156.4810755756272	156.57617537053932

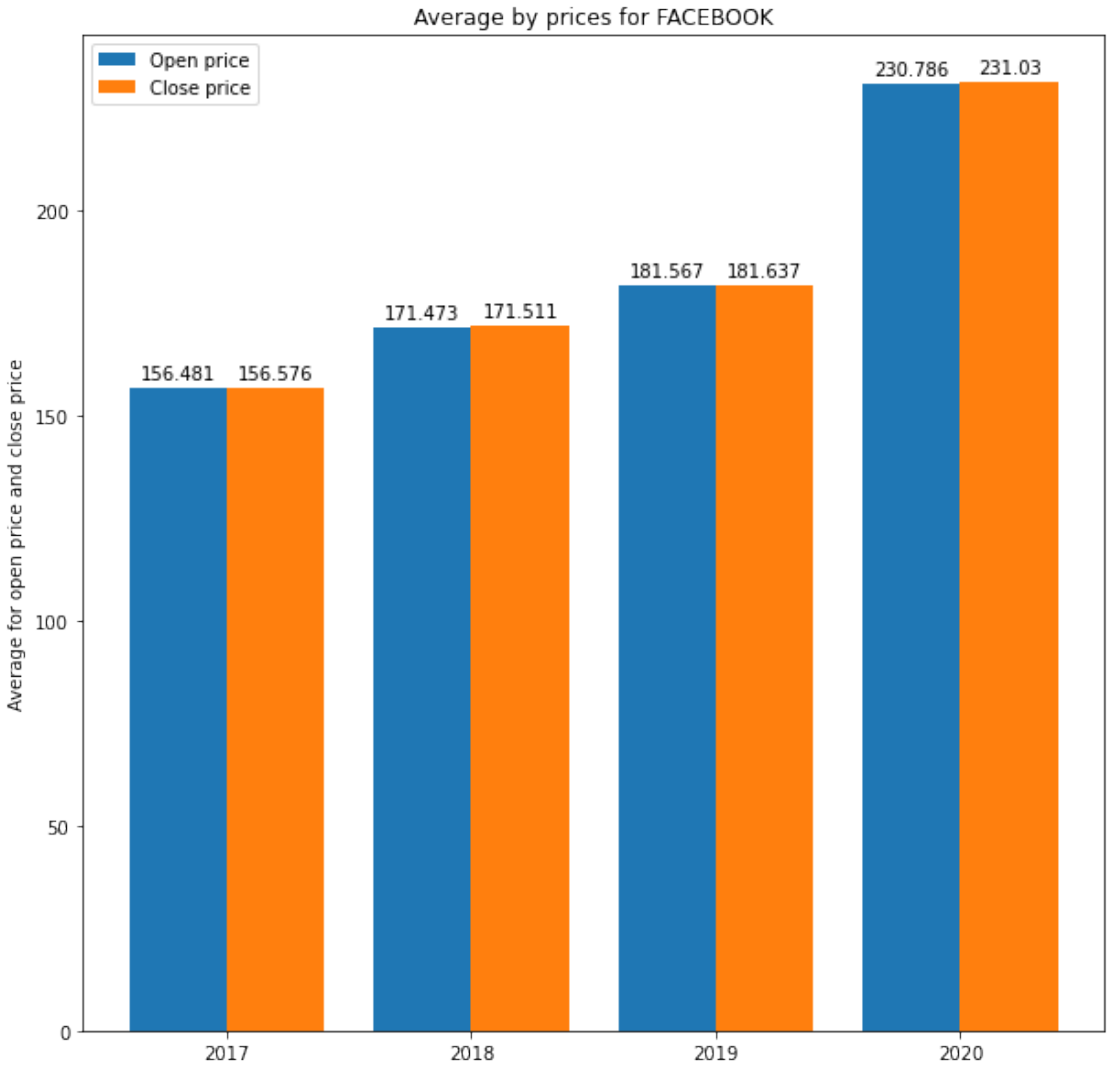
Average open price and close price in 2018

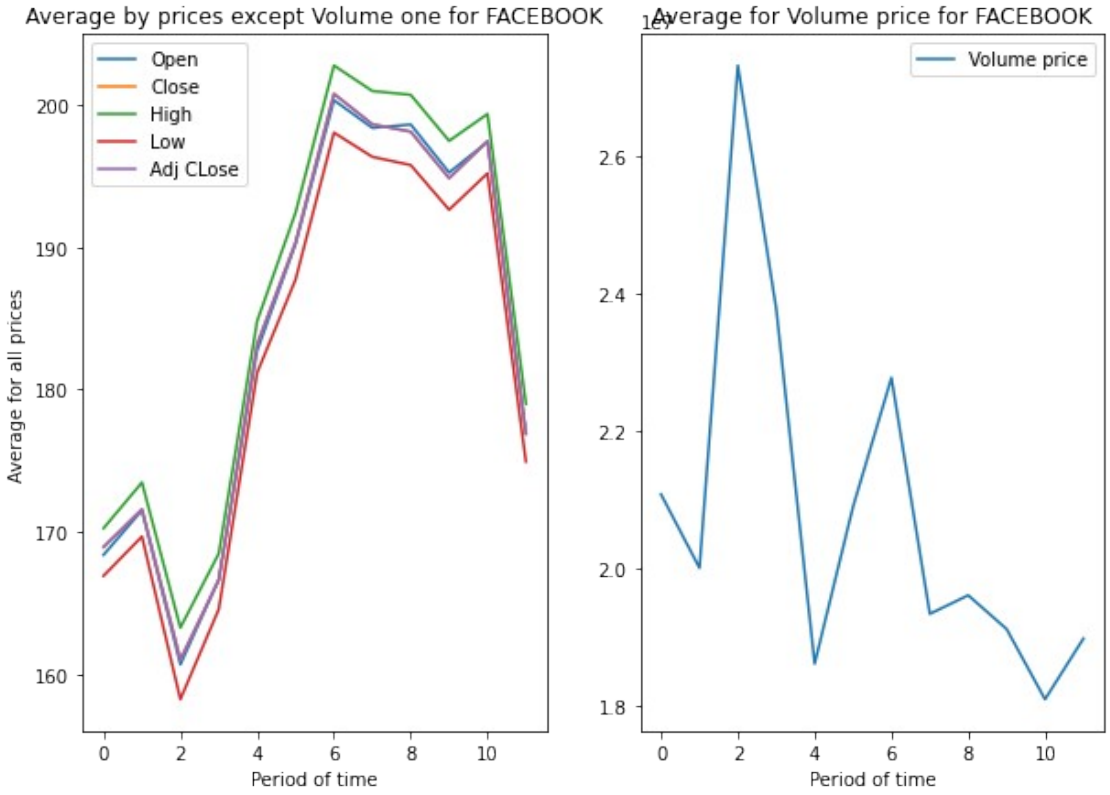
avg(Open)	avg(Close)
171.4729481427318	171.5109556889629

Average open price and close price in 2019

avg(Open)	avg(Close)

```
|181.56654727269733|181.6374996124752|
+------------------+-----------------+

Average open price and close price in 2020
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|230.78562233069425|231.0295712057613|
+------------------+-----------------+
```

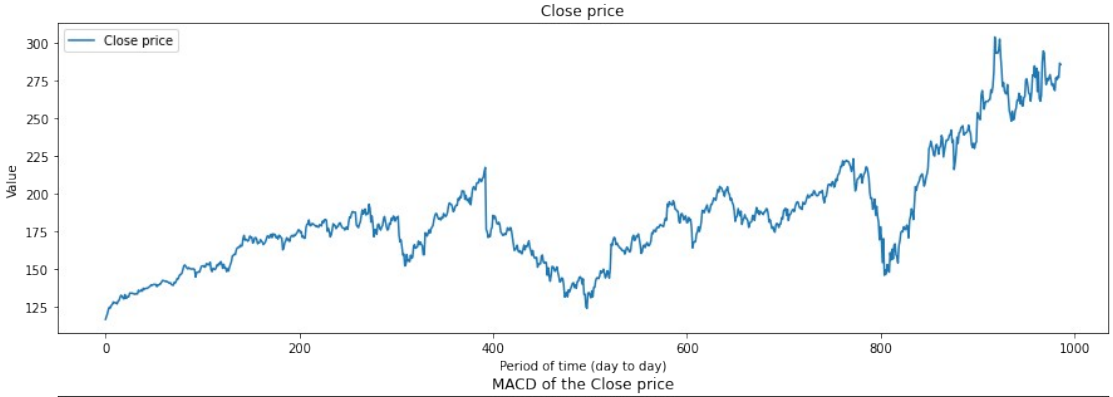


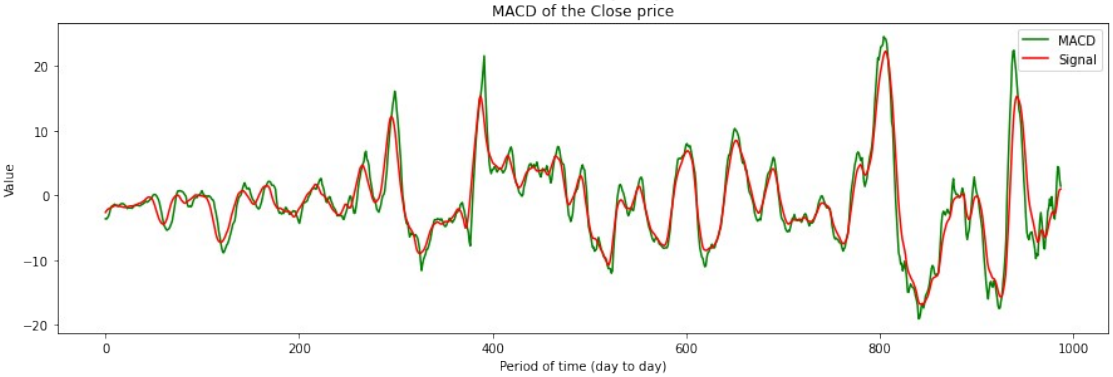



```
================
Visualization and plots for AMAZON
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
+------------+
|2017-01-03 00:00:00|758.76| 747.7|757.92|753.67|3521100.0|   753.67|
AMAZON|
|2017-01-04 00:00:00|759.68| 754.2|758.39|757.18|2510500.0|   757.18|
AMAZON|
|2017-01-05 00:00:00| 782.4|760.26|761.55|780.45|5830100.0|   780.45|
AMAZON|
|2017-01-06 00:00:00|799.44|778.48|782.36|795.99|5986200.0|   795.99|
AMAZON|
```

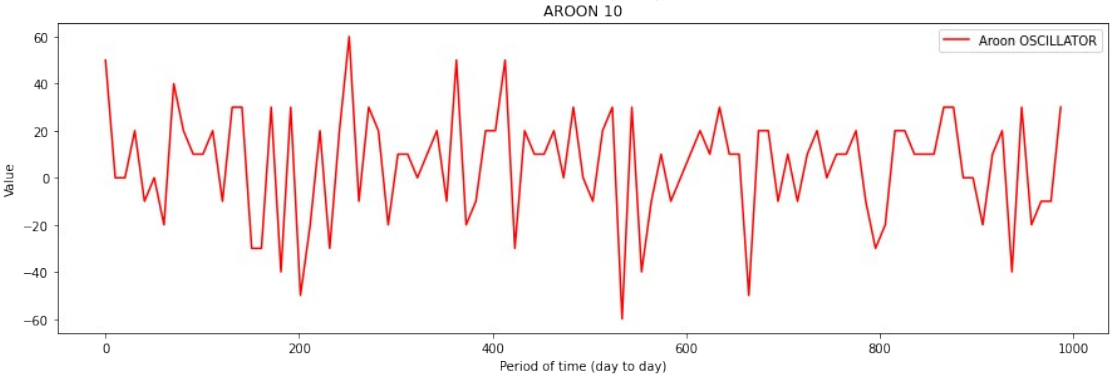
|2017-01-09 00:00:00|801.77|791.77| 798.0|796.92|3446100.0| 796.92|
AMAZON|
|2017-01-10 00:00:00| 798.0|789.54| 796.6| 795.9|2558400.0| 795.9|
AMAZON|
|2017-01-11 00:00:00| 799.5|789.51|793.66|799.02|2992800.0| 799.02|
AMAZON|
|2017-01-12 00:00:00|814.13| 799.5|800.31|813.64|4873900.0| 813.64|
AMAZON|
|2017-01-13 00:00:00|821.65| 811.4|814.32|817.14|3791900.0| 817.14|
AMAZON|
|2017-01-17 00:00:00| 816.0|803.44| 815.7|809.72|3670500.0| 809.72|
AMAZON|
|2017-01-18 00:00:00|811.73|804.27| 809.5|807.48|2354200.0| 807.48|
AMAZON|
|2017-01-19 00:00:00|813.51|807.32| 810.0|809.04|2540800.0| 809.04|
AMAZON|
|2017-01-20 00:00:00|816.02|806.26|815.28|808.33|3376200.0| 808.33|
AMAZON|
|2017-01-23 00:00:00| 818.5|805.08| 806.8|817.88|2797500.0| 817.88|
AMAZON|
|2017-01-24 00:00:00|823.99| 814.5| 822.0|822.44|2971700.0| 822.44|
AMAZON|
|2017-01-25 00:00:00|837.42|825.29|825.79|836.52|3922600.0| 836.52|
AMAZON|
|2017-01-26 00:00:00|843.84| 833.0|835.53|839.15|3586300.0| 839.15|
AMAZON|
|2017-01-27 00:00:00| 839.7|829.44| 839.0|835.77|2998700.0| 835.77|
AMAZON|
|2017-01-30 00:00:00| 833.5|816.38| 833.0|830.38|3747300.0| 830.38|
AMAZON|
|2017-01-31 00:00:00|826.99|819.56|823.75|823.48|3137200.0| 823.48|
AMAZON|
|2017-02-01 00:00:00|833.78|824.94|829.21|832.35|3850200.0| 832.35|
AMAZON|
|2017-02-02 00:00:00|842.49|828.26|836.59|839.95|7350500.0| 839.95|
AMAZON|
|2017-02-03 00:00:00| 818.3| 804.0|806.72| 810.2|1.08688E7| 810.2|
AMAZON|
|2017-02-06 00:00:00|810.72| 803.0| 809.8|807.64|3897300.0| 807.64|
AMAZON|
|2017-02-07 00:00:00|816.16| 807.5|809.31| 812.5|3466100.0| 812.5|
AMAZON|
|2017-02-08 00:00:00|821.48| 812.5|812.69|819.71|2858000.0| 819.71|
AMAZON|
|2017-02-09 00:00:00| 825.0|819.71| 821.6|821.36|2484900.0| 821.36|
AMAZON|
|2017-02-10 00:00:00| 828.0|822.85|823.82|827.46|2429600.0| 827.46|
AMAZON|
|2017-02-13 00:00:00| 843.0|828.55|831.62|836.53|4172600.0| 836.53|
AMAZON|

```
|2017-02-14 00:00:00|838.31|831.45| 837.0|836.39|2792400.0|   836.39|
AMAZON|
|2017-02-15 00:00:00|842.81|832.82| 834.0| 842.7|2968900.0|    842.7|
AMAZON|
|2017-02-16 00:00:00| 845.0|839.38|841.84|844.14|2714700.0|   844.14|
AMAZON|
|2017-02-17 00:00:00|847.27|840.73| 842.0|845.07|3112300.0|   845.07|
AMAZON|
|2017-02-21 00:00:00|857.98|847.25|848.84|856.44|3507700.0|   856.44|
AMAZON|
|2017-02-22 00:00:00|858.43|852.18|856.95|855.61|2617000.0|   855.61|
AMAZON|
|2017-02-23 00:00:00|860.86| 848.0|857.57|852.19|3462000.0|   852.19|
AMAZON|
|2017-02-24 00:00:00|845.81|837.75|844.69|845.24|3688000.0|   845.24|
AMAZON|
|2017-02-27 00:00:00| 852.5|839.67|842.38|848.64|2713600.0|   848.64|
AMAZON|
|2017-02-28 00:00:00|854.09|842.05|851.45|845.04|2793700.0|   845.04|
AMAZON|
|2017-03-01 00:00:00|854.83|849.01|853.05|853.08|2760100.0|   853.08|
AMAZON|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```

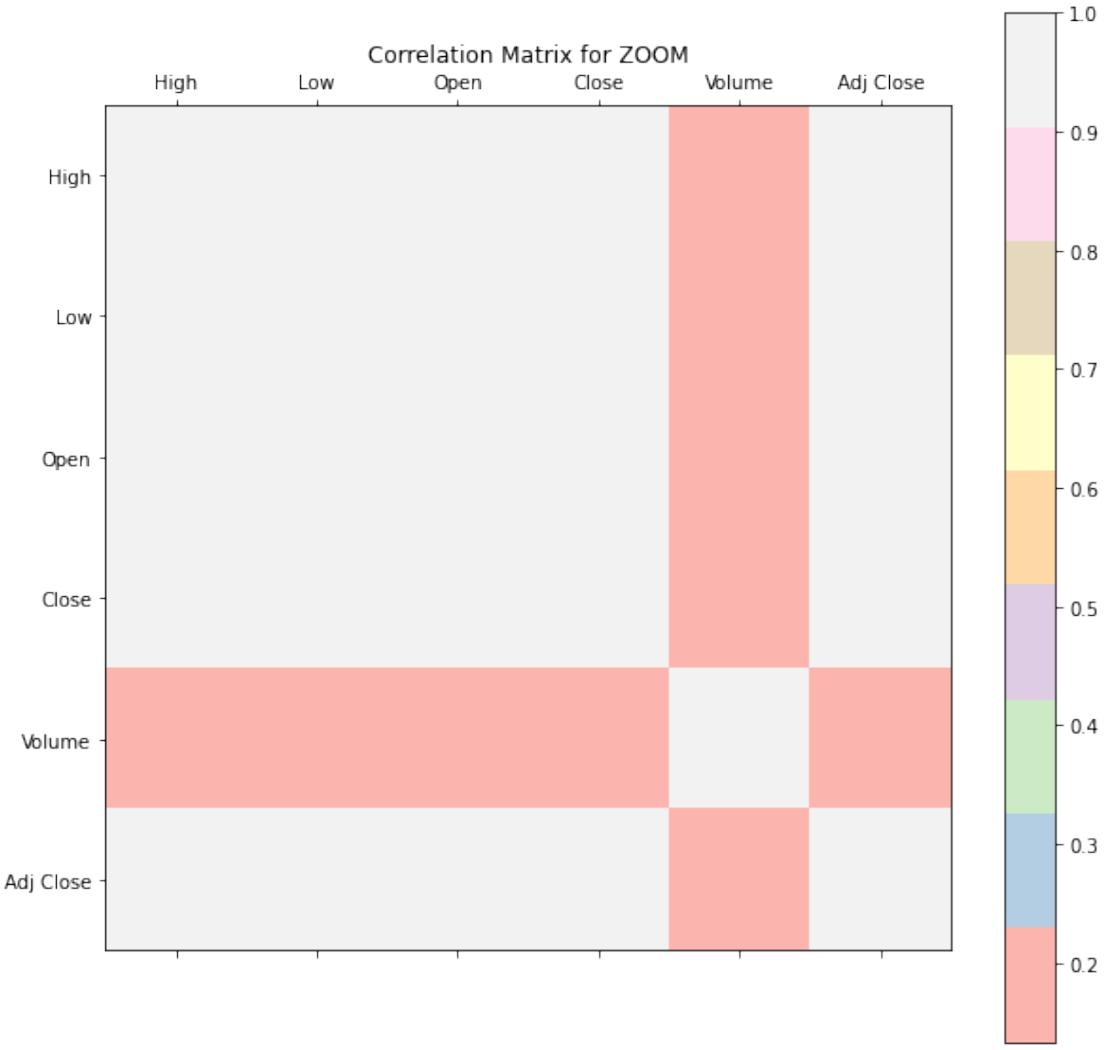

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
300
275
250
225
200
175
150
125
0
200
400
600
800
1000



Average open price and close price in 2017

avg(Open)	avg(Close)
968.275618959708	968.1670116409363

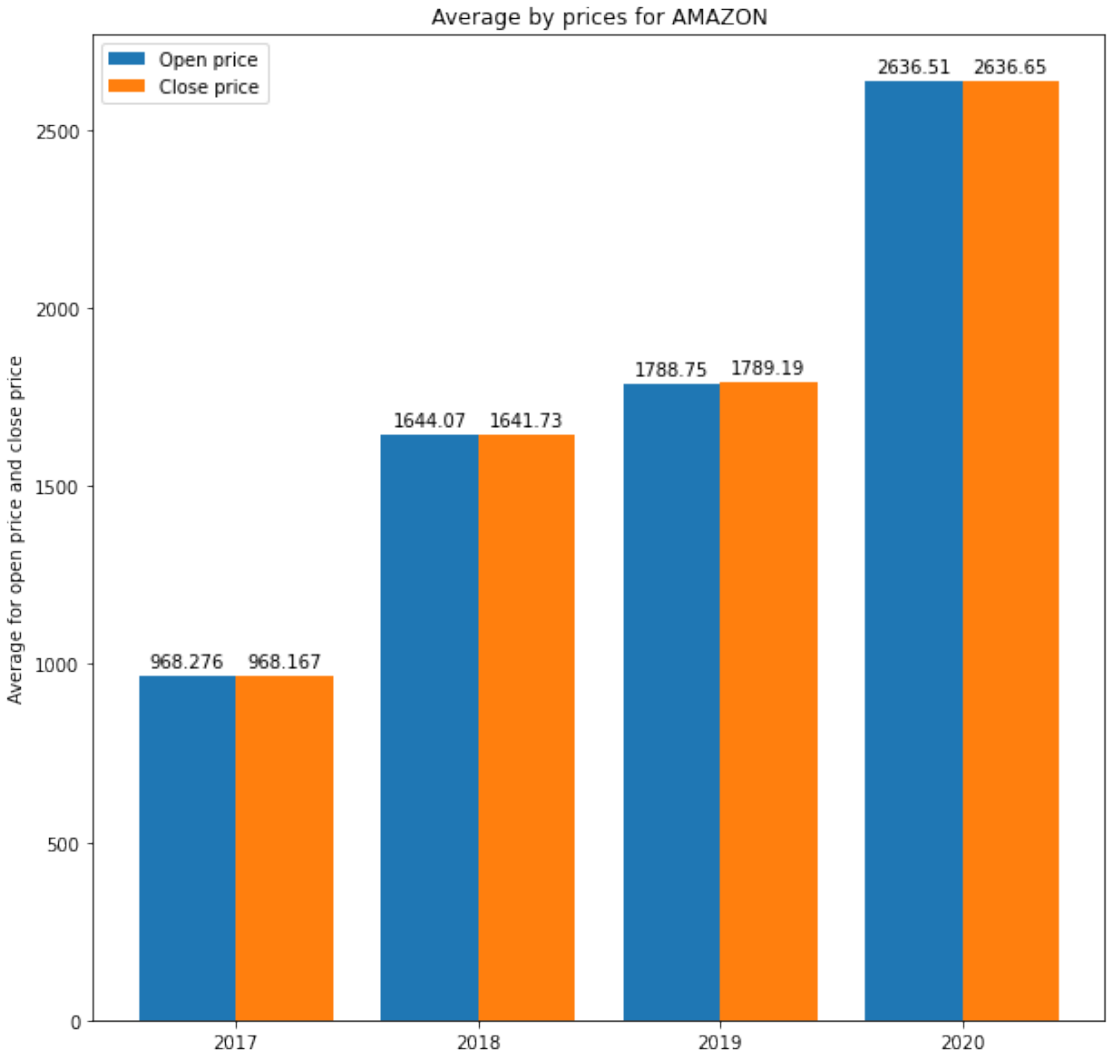
Average open price and close price in 2018

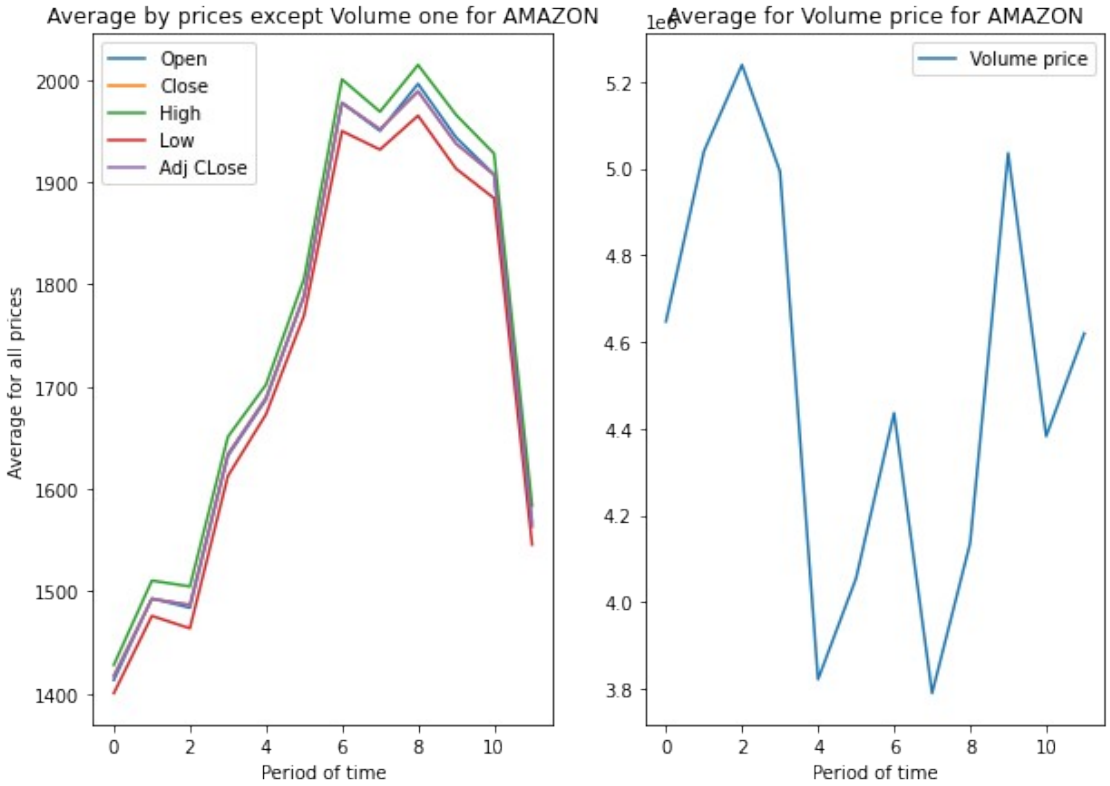
avg(Open)	avg(Close)
1644.0727091633466	1641.7261758629545

Average open price and close price in 2019

avg(Open)	avg(Close)

```
|1788.7461896623884|1789.189206077939|
+------------------+-----------------+

Average open price and close price in 2020
+------------------+-----------------+
|         avg(Open)|       avg(Close)|
+------------------+-----------------+
|2636.5054538710433|2636.649604240712|
+------------------+-----------------+
```

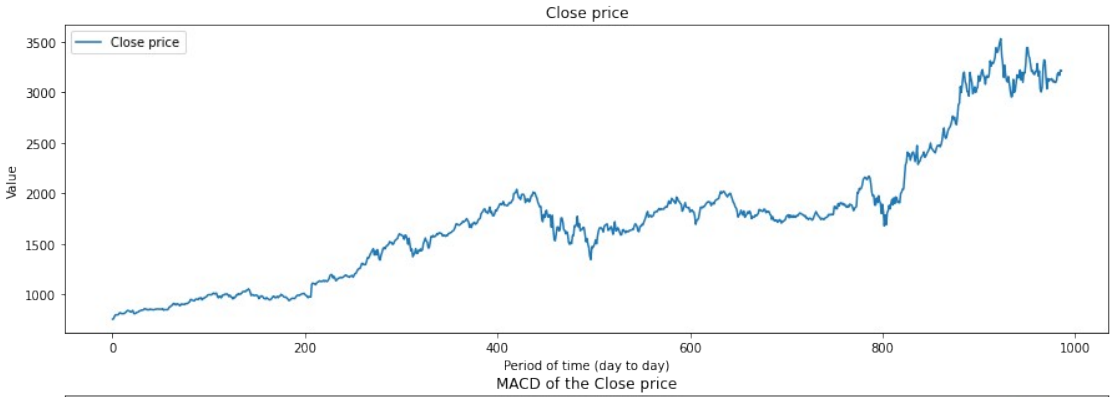


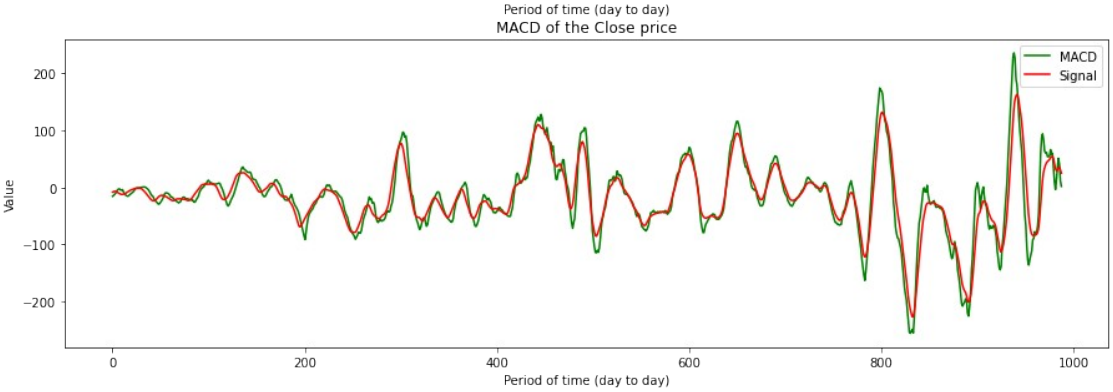


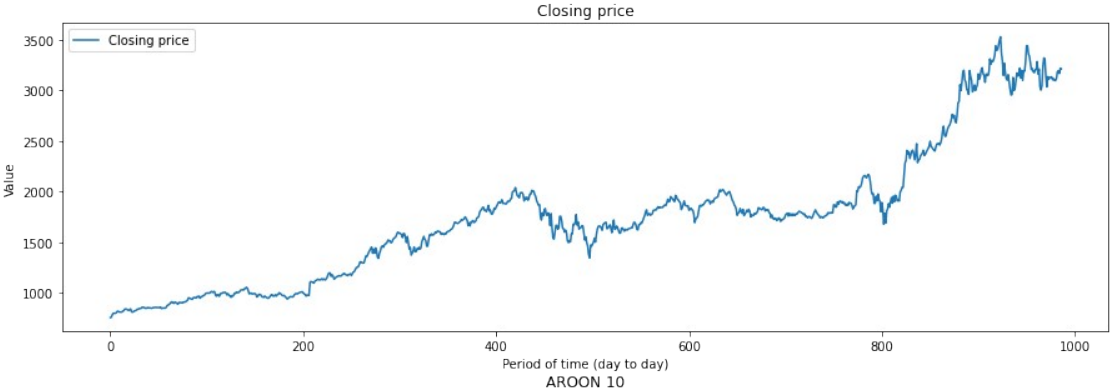

```
================
Visualization and plots for GOOGLE
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+-------+-------+------+-------+---------
+---------+------------+
|               Date|   High|    Low|  Open|  Close|   Volume|Adj
Close|company_name|
+-------------------+-------+-------+------+-------+---------
+---------+------------+
|2017-01-03 00:00:00| 789.63|  775.8|778.81| 786.14|1657300.0|
786.14|      GOOGLE|
|2017-01-04 00:00:00| 791.34| 783.16|788.36|  786.9|1073000.0|
786.9|      GOOGLE|
|2017-01-05 00:00:00| 794.48| 785.02|786.08| 794.02|1335200.0|
794.02|      GOOGLE|
|2017-01-06 00:00:00|  807.9|792.204|795.26| 806.15|1640200.0|
806.15|      GOOGLE|
```

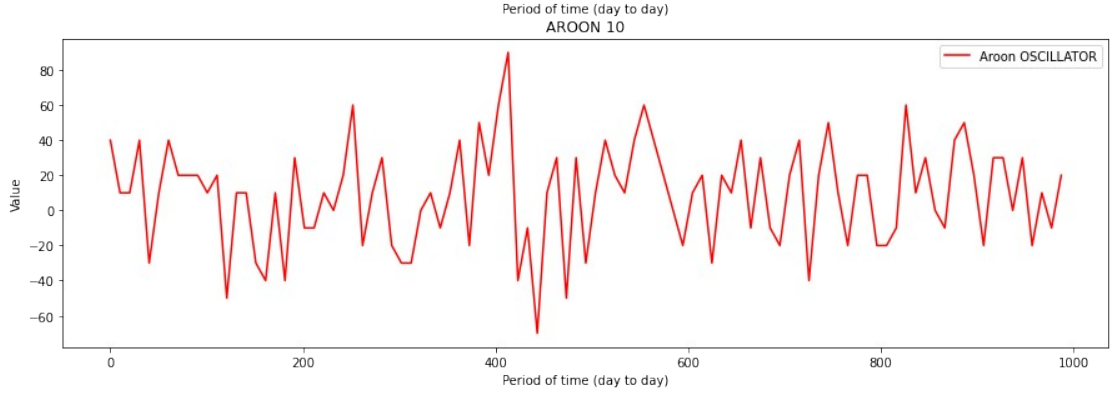
|2017-01-09 00:00:00|809.966| 802.83| 806.4| 806.65|1274600.0|
806.65| GOOGLE|
|2017-01-10 00:00:00| 809.13| 803.51|807.86| 804.79|1176800.0|
804.79| GOOGLE|
|2017-01-11 00:00:00| 808.15| 801.37| 805.0| 807.91|1065900.0|
807.91| GOOGLE|
|2017-01-12 00:00:00| 807.39| 799.17|807.14| 806.36|1353100.0|
806.36| GOOGLE|
|2017-01-13 00:00:00|811.224| 806.69|807.48| 807.88|1099200.0|
807.88| GOOGLE|
|2017-01-17 00:00:00| 807.14| 800.37|807.08| 804.61|1362100.0|
804.61| GOOGLE|
|2017-01-18 00:00:00|806.205| 800.99|805.81| 806.07|1294400.0|
806.07| GOOGLE|
|2017-01-19 00:00:00| 809.48| 801.8|805.12|802.175| 919300.0|
802.175| GOOGLE|
|2017-01-20 00:00:00| 806.91| 801.69|806.91| 805.02|1670000.0|
805.02| GOOGLE|
|2017-01-23 00:00:00| 820.87| 803.74|807.25| 819.31|1963600.0|
819.31| GOOGLE|
|2017-01-24 00:00:00| 825.9|817.821| 822.3| 823.87|1474000.0|
823.87| GOOGLE|
|2017-01-25 00:00:00| 835.77| 825.06|829.62| 835.67|1494500.0|
835.67| GOOGLE|
|2017-01-26 00:00:00| 838.0| 827.01|837.81| 832.15|2973900.0|
832.15| GOOGLE|
|2017-01-27 00:00:00| 841.95| 820.44|834.71| 823.31|2965800.0|
823.31| GOOGLE|
|2017-01-30 00:00:00| 815.84| 799.8|814.66| 802.32|3246600.0|
802.32| GOOGLE|
|2017-01-31 00:00:00| 801.25| 790.52|796.86| 796.79|2160600.0|
796.79| GOOGLE|
|2017-02-01 00:00:00| 801.19| 791.19|799.68|795.695|2029700.0|
795.695| GOOGLE|
|2017-02-02 00:00:00| 802.7| 792.0| 793.8| 798.53|1532100.0|
798.53| GOOGLE|
|2017-02-03 00:00:00| 806.0| 800.37|802.99| 801.49|1463400.0|
801.49| GOOGLE|
|2017-02-06 00:00:00| 801.67| 795.25| 799.7| 801.34|1184500.0|
801.34| GOOGLE|
|2017-02-07 00:00:00| 810.5| 801.78|803.99| 806.97|1241200.0|
806.97| GOOGLE|
|2017-02-08 00:00:00| 811.84| 803.19| 807.0| 808.38|1155300.0|
808.38| GOOGLE|
|2017-02-09 00:00:00| 810.66| 804.54|809.51| 809.56| 989700.0|
809.56| GOOGLE|
|2017-02-10 00:00:00| 815.25| 809.78| 811.7| 813.67|1135000.0|
813.67| GOOGLE|
|2017-02-13 00:00:00|820.959| 815.49| 816.0| 819.24|1213300.0|
819.24| GOOGLE|

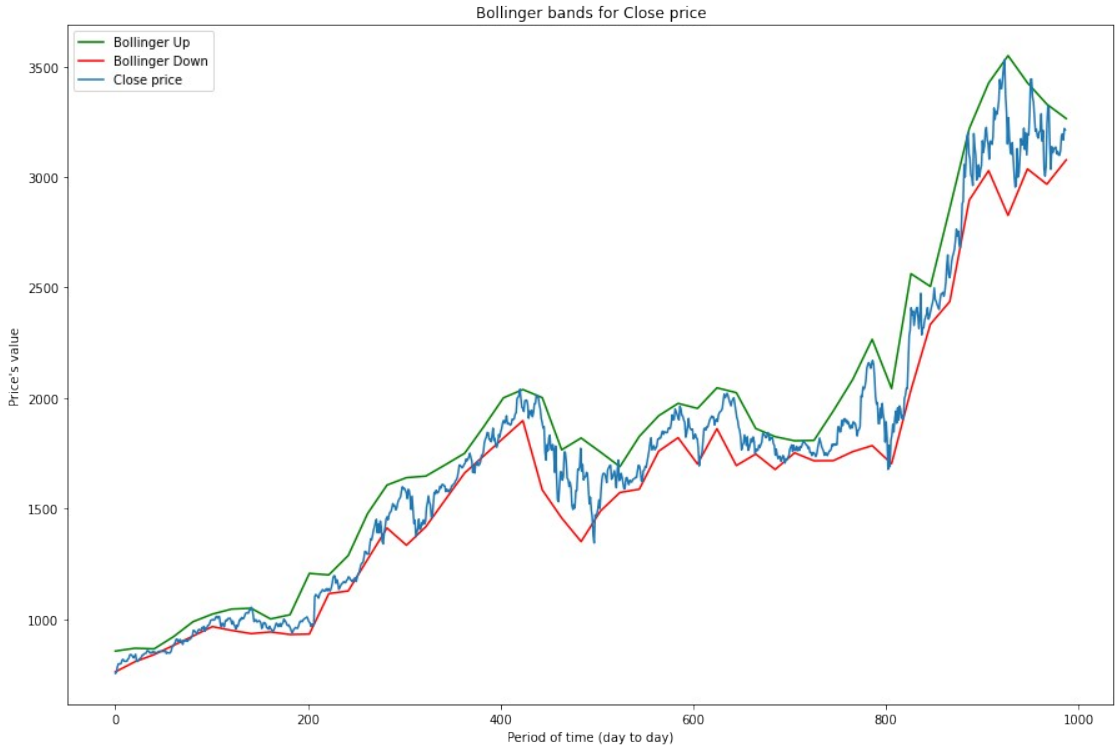
```
|2017-02-14 00:00:00|  823.0|  816.0| 819.0| 820.45|1054700.0|
820.45|      GOOGLE|
|2017-02-15 00:00:00|  823.0| 818.47|819.36| 818.98|1313600.0|
818.98|      GOOGLE|
|2017-02-16 00:00:00|  824.4| 818.98|819.93| 824.16|1287600.0|
824.16|      GOOGLE|
|2017-02-17 00:00:00| 828.07|821.655|823.02| 828.07|1611000.0|
828.07|      GOOGLE|
|2017-02-21 00:00:00| 833.45| 828.35|828.66| 831.66|1262300.0|
831.66|      GOOGLE|
|2017-02-22 00:00:00| 833.25| 828.64|828.66| 830.76| 982900.0|
830.76|      GOOGLE|
|2017-02-23 00:00:00| 832.46| 822.88|830.12| 831.33|1472800.0|
831.33|      GOOGLE|
|2017-02-24 00:00:00|  829.0|  824.2|827.73| 828.64|1392200.0|
828.64|      GOOGLE|
|2017-02-27 00:00:00|  830.5|  824.0|824.55| 829.28|1101500.0|
829.28|      GOOGLE|
|2017-02-28 00:00:00| 828.54|  820.2|825.61| 823.21|2260800.0|
823.21|      GOOGLE|
|2017-03-01 00:00:00|836.255| 827.26|828.85| 835.24|1496500.0|
835.24|      GOOGLE|
+-------------------+-------+-------+------+-------+---------
+---------+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels([''']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels([''']+columns)
```

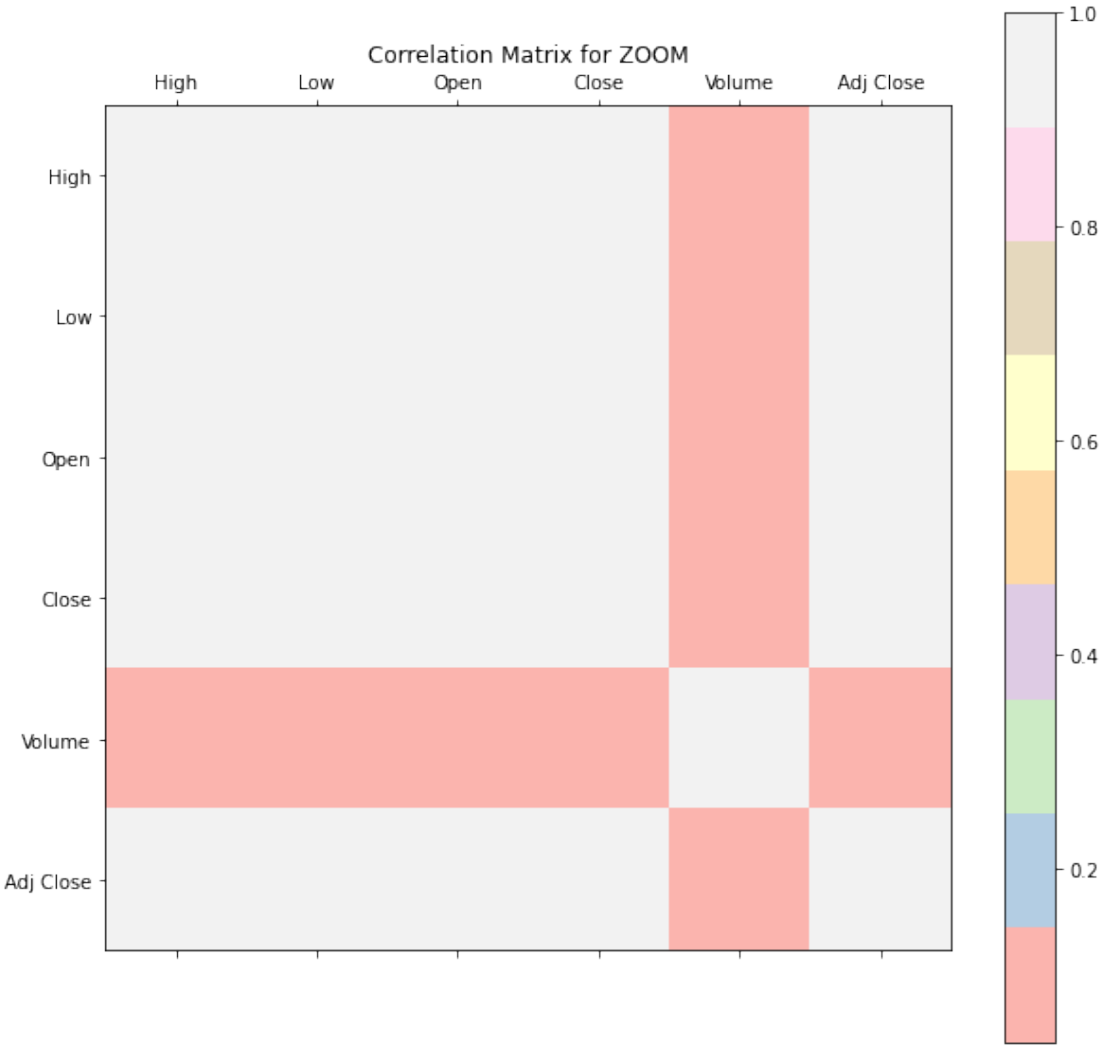

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
3500
3000
2500
2000
1500
1000
0
200
400
600
800
1000



Average open price and close price in 2017

avg(Open)	avg(Close)
921.1211927193569	921.7808373439834

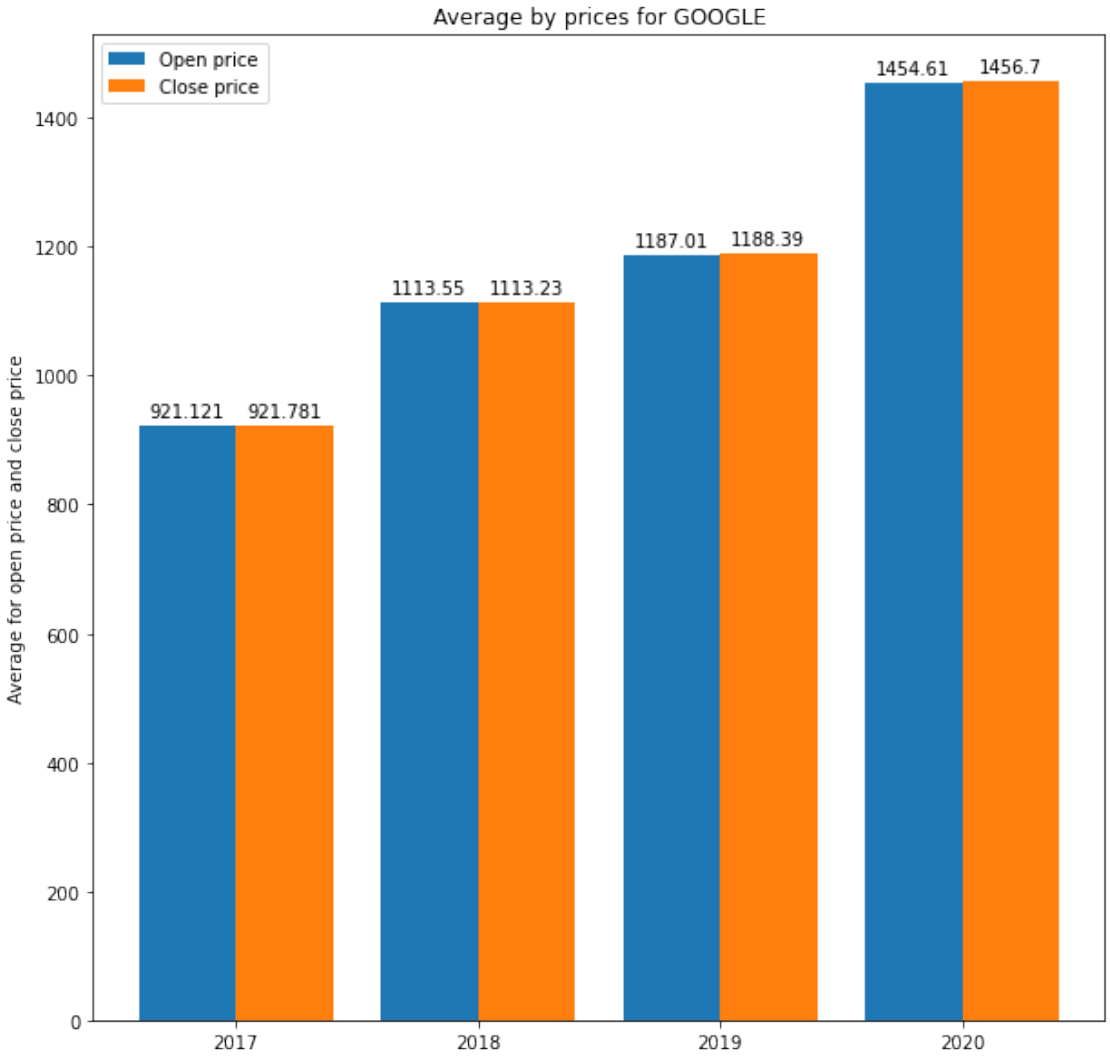
Average open price and close price in 2018

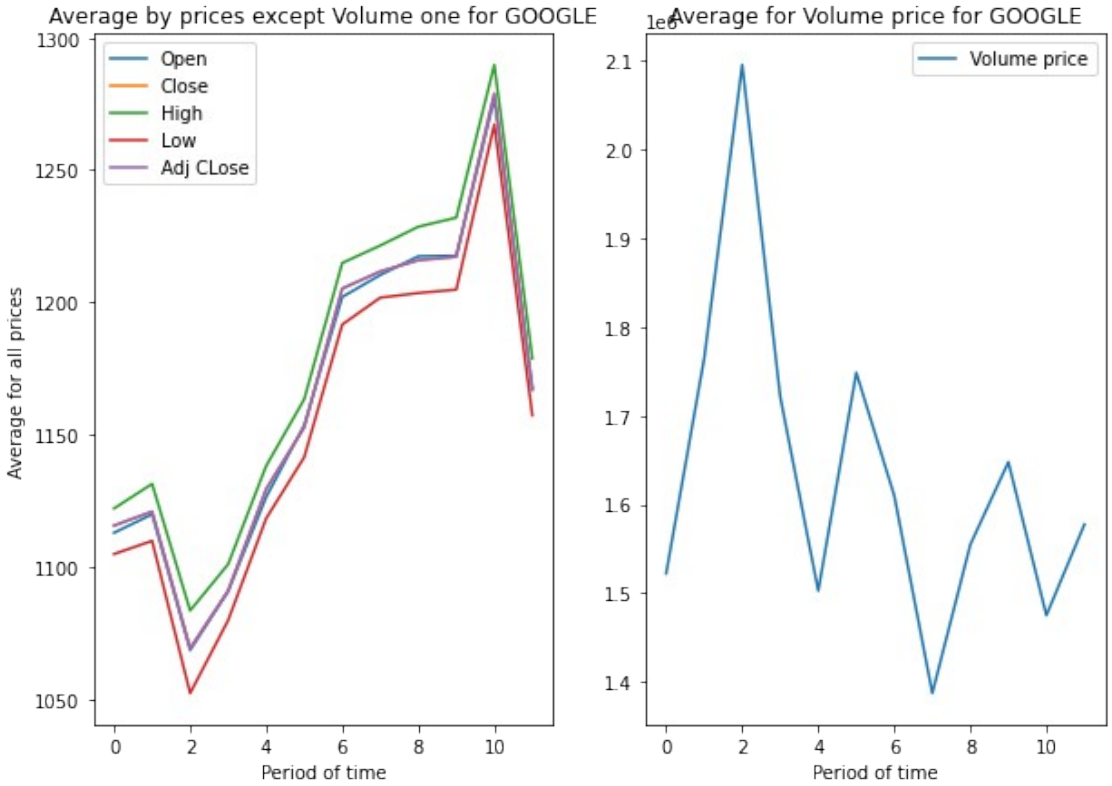
avg(Open)	avg(Close)
1113.554100735729	1113.225134131443

Average open price and close price in 2019

avg(Open)	avg(Close)

```
|1187.0098210894873|1188.393057444739|
+------------------+-----------------+

Average open price and close price in 2020
+------------------+------------------+
|         avg(Open)|        avg(Close)|
+------------------+------------------+
|1454.6135625880163|1456.6964127045333|
+------------------+------------------+
```

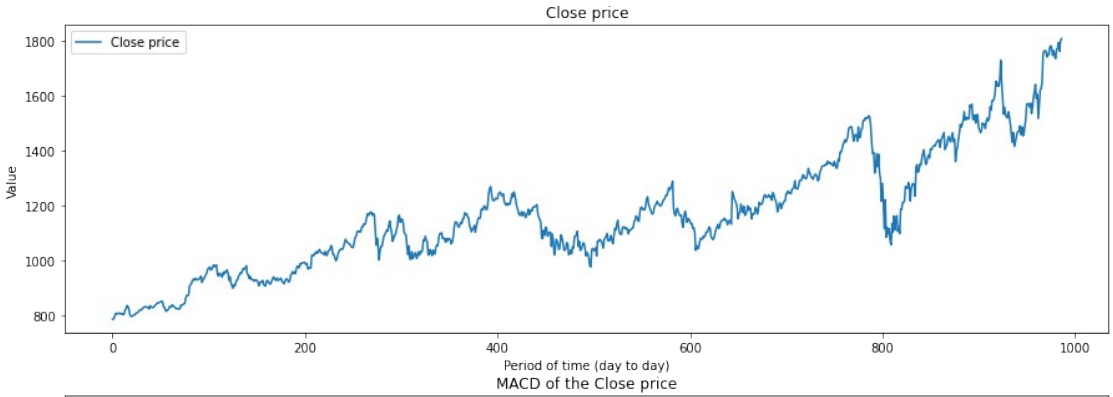


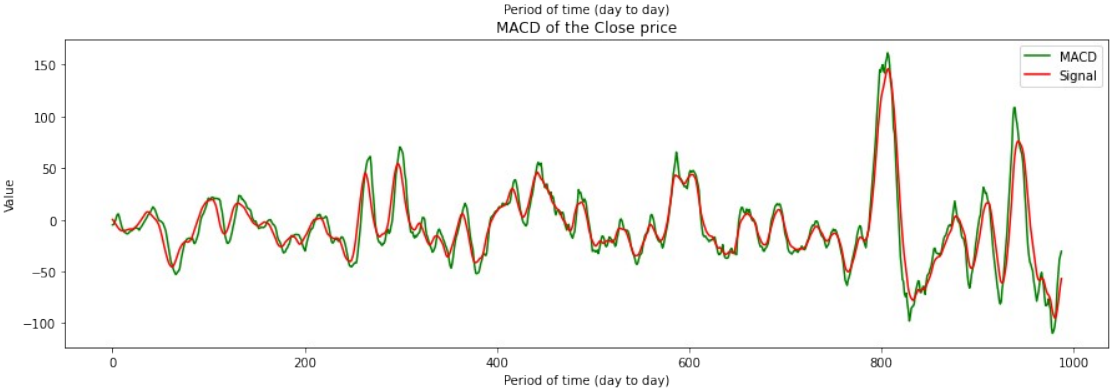



```
================
Visualization and plots for TESLA
root
 |-- Date: timestamp (nullable = true)
 |-- High: float (nullable = true)
 |-- Low: float (nullable = true)
 |-- Open: float (nullable = true)
 |-- Close: float (nullable = true)
 |-- Volume: float (nullable = true)
 |-- Adj Close: float (nullable = true)
 |-- company_name: string (nullable = true)

+-------------------+------+------+------+------+---------+---------
+------------+
|               Date|  High|   Low|  Open| Close|   Volume|Adj Close|
company_name|
+-------------------+------+------+------+------+---------+---------
+------------+
|2017-01-03 00:00:00|44.066|42.192|42.972|43.398|2.96165E7|   43.398|
TESLA|
|2017-01-04 00:00:00|  45.6|42.862| 42.95|45.398|5.60675E7|   45.398|
TESLA|
|2017-01-05 00:00:00|45.496| 44.39|45.284| 45.35|2.95585E7|    45.35|
TESLA|
|2017-01-06 00:00:00|46.062| 45.09|45.386|45.802|2.76395E7|   45.802|
TESLA|
```

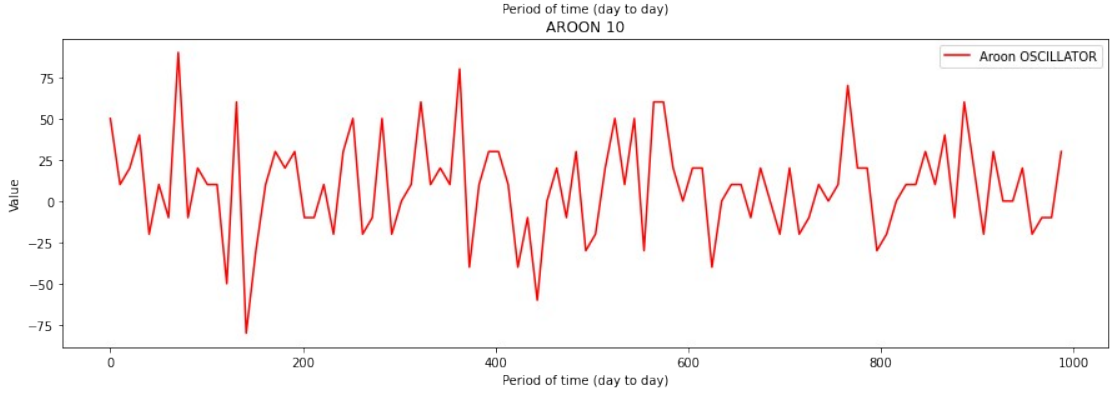
|2017-01-09 00:00:00|46.384| 45.6|45.794|46.256|1.98975E7| 46.256|
TESLA|
|2017-01-10 00:00:00| 46.4|45.378| 46.4|45.974| 1.83E7| 45.974|
TESLA|
|2017-01-11 00:00:00|45.996|45.336|45.814|45.946| 1.8254E7| 45.946|
TESLA|
|2017-01-12 00:00:00| 46.14|45.116|45.812|45.918| 1.8951E7| 45.918|
TESLA|
|2017-01-13 00:00:00| 47.57|45.918| 46.0| 47.55| 3.0465E7| 47.55|
TESLA|
|2017-01-17 00:00:00|47.992|46.874| 47.34|47.116|2.30875E7| 47.116|
TESLA|
|2017-01-18 00:00:00|47.942|47.116| 47.33|47.672| 1.8845E7| 47.672|
TESLA|
|2017-01-19 00:00:00|49.736| 48.15| 49.45|48.752|3.86615E7| 48.752|
TESLA|
|2017-01-20 00:00:00| 49.2|48.602|49.092|48.946|2.10215E7| 48.946|
TESLA|
|2017-01-23 00:00:00|50.178| 49.1| 49.17|49.784|3.13145E7| 49.784|
TESLA|
|2017-01-24 00:00:00| 50.96| 49.93| 50.0|50.922|2.48275E7| 50.922|
TESLA|
|2017-01-25 00:00:00|51.692| 50.36|51.462|50.894| 2.5713E7| 50.894|
TESLA|
|2017-01-26 00:00:00|51.148| 50.15|50.858|50.502|1.57605E7| 50.502|
TESLA|
|2017-01-27 00:00:00| 50.6|49.704|50.276| 50.59|1.58315E7| 50.59|
TESLA|
|2017-01-30 00:00:00|51.058| 49.42|50.506|50.126|1.90055E7| 50.126|
TESLA|
|2017-01-31 00:00:00|51.178| 49.54|49.848|50.386|2.05805E7| 50.386|
TESLA|
|2017-02-01 00:00:00| 50.64| 49.81| 50.61|49.848| 1.9794E7| 49.848|
TESLA|
|2017-02-02 00:00:00|50.484|49.542|49.668| 50.31| 1.2499E7| 50.31|
TESLA|
|2017-02-03 00:00:00|50.436|49.936|50.382|50.266|1.09335E7| 50.266|
TESLA|
|2017-02-06 00:00:00|51.564|50.126| 50.2|51.554|1.78125E7| 51.554|
TESLA|
|2017-02-07 00:00:00| 52.0|51.284|51.638|51.496| 2.1224E7| 51.496|
TESLA|
|2017-02-08 00:00:00|52.672| 51.24| 51.47|52.416| 1.9665E7| 52.416|
TESLA|
|2017-02-09 00:00:00|54.236| 53.23| 53.25| 53.84| 3.9101E7| 53.84|
TESLA|
|2017-02-10 00:00:00| 54.19|53.222|53.958|53.846|1.80985E7| 53.846|
TESLA|
|2017-02-13 00:00:00|56.158|54.102|54.148| 56.12| 3.5148E7| 56.12|
TESLA|

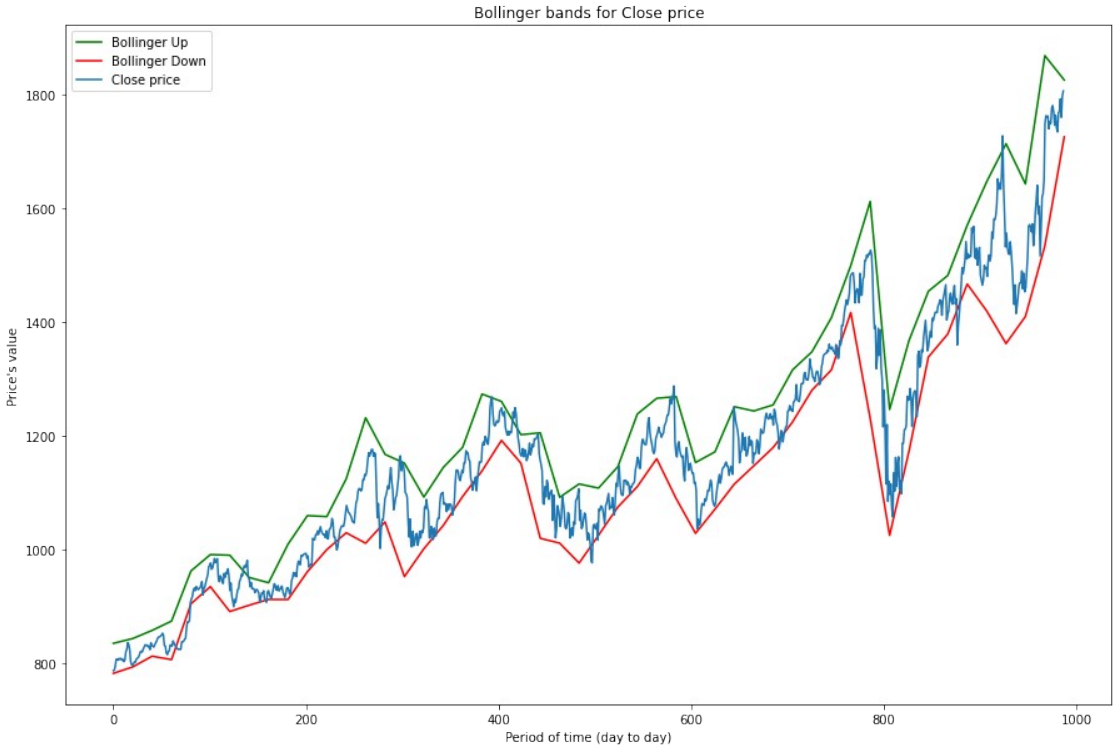
```
|2017-02-14 00:00:00|57.478|55.722|55.806|56.196| 3.6726E7|   56.196|
TESLA|
|2017-02-15 00:00:00|56.448|55.288|  56.0|55.952|2.47395E7|   55.952|
TESLA|
|2017-02-16 00:00:00|  56.0|  53.7| 55.52| 53.79|3.53865E7|    53.79|
TESLA|
|2017-02-17 00:00:00|54.578| 52.83| 53.16|54.446|3.12855E7|   54.446|
TESLA|
|2017-02-21 00:00:00| 56.28|54.802| 55.09|55.478|2.83835E7|   55.478|
TESLA|
|2017-02-22 00:00:00| 56.69| 54.52|56.062|54.702| 4.3775E7|   54.702|
TESLA|
|2017-02-23 00:00:00|52.932|51.112|  52.8|51.198| 7.4576E7|   51.198|
TESLA|
|2017-02-24 00:00:00| 51.65| 50.04|50.532|  51.4| 4.0858E7|     51.4|
TESLA|
|2017-02-27 00:00:00|49.672|48.402|49.634|49.246| 5.7304E7|   49.246|
TESLA|
|2017-02-28 00:00:00|  50.2| 48.78|48.838|49.998|3.03905E7|   49.998|
TESLA|
|2017-03-01 00:00:00| 50.97|49.822|50.836|50.004|2.40475E7|   50.004|
TESLA|
+-------------------+------+------+------+------+---------+---------
+------------+
only showing top 40 rows

Number of rows = 987
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']

/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```

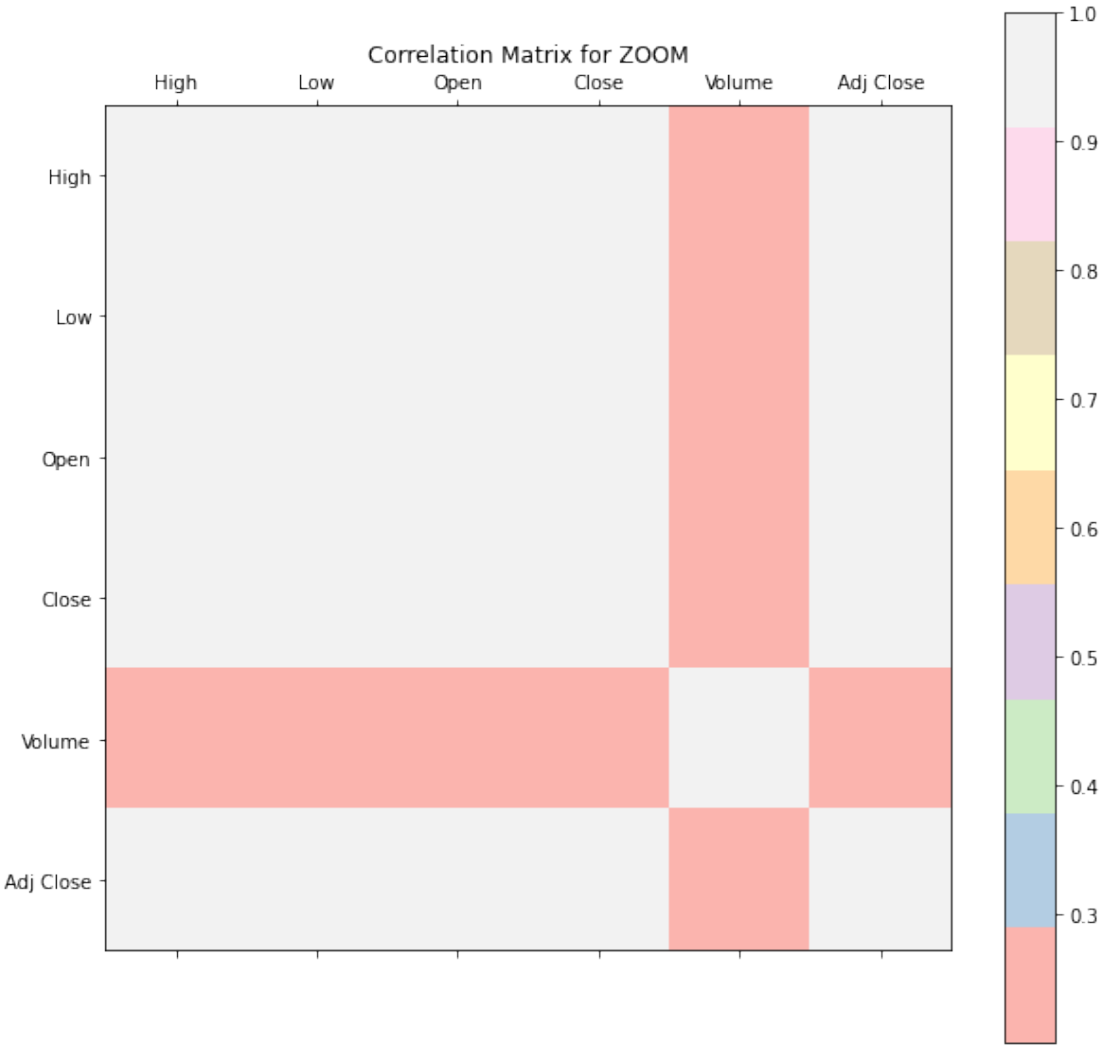

Close price
Close price
Value
Period of time (day to day)


MACD of the Close price
MACD
Signal
Value
Period of time (day to day)


Closing price
Closing price
Value
Period of time (day to day)


AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)


Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)
800
1000
1200
1400
1600
1800
0
200
400
600
800
1000



Average open price and close price in 2017

avg(Open)	avg(Close)
62.85924295980142	62.863258969736286

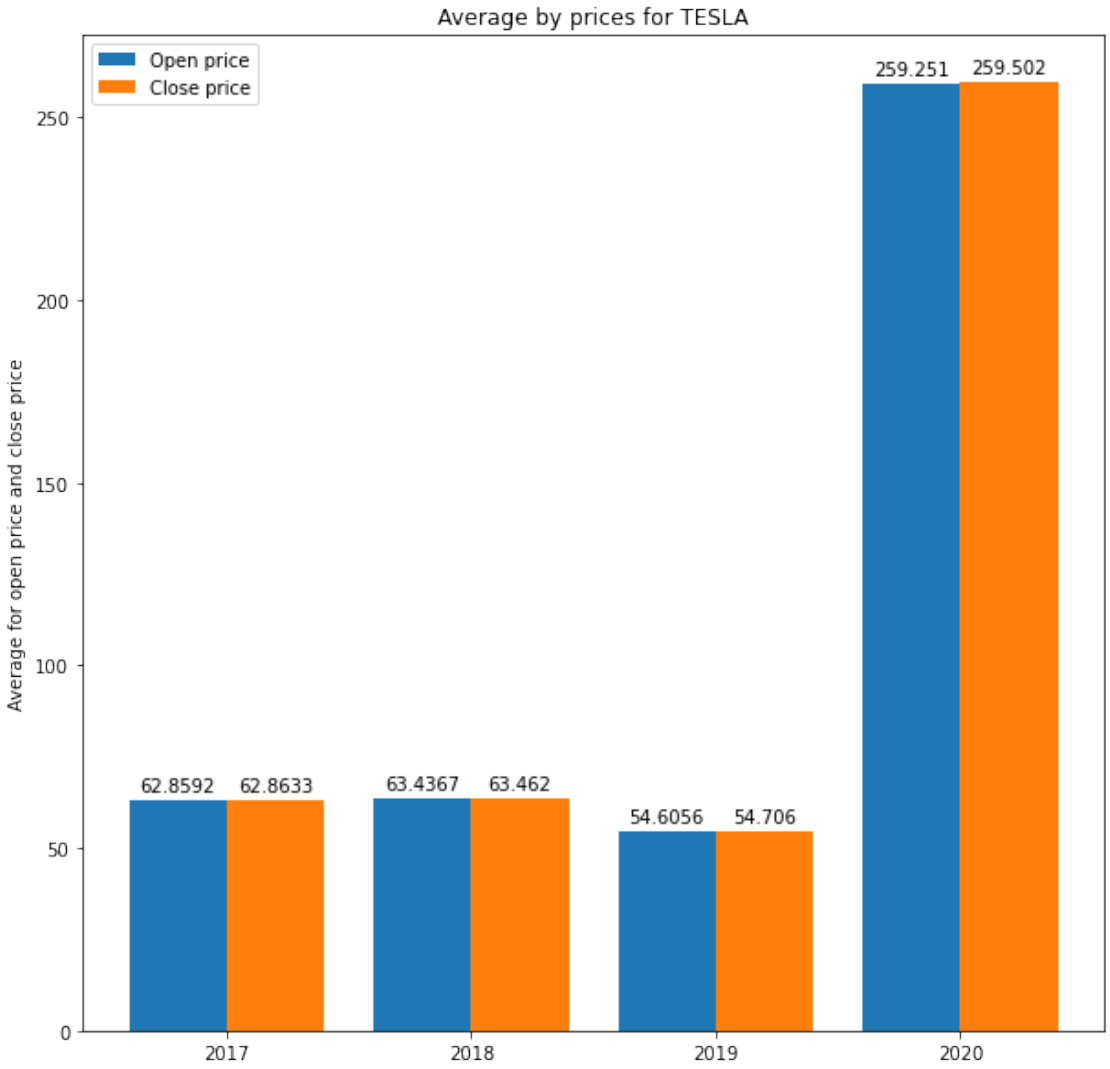
Average open price and close price in 2018

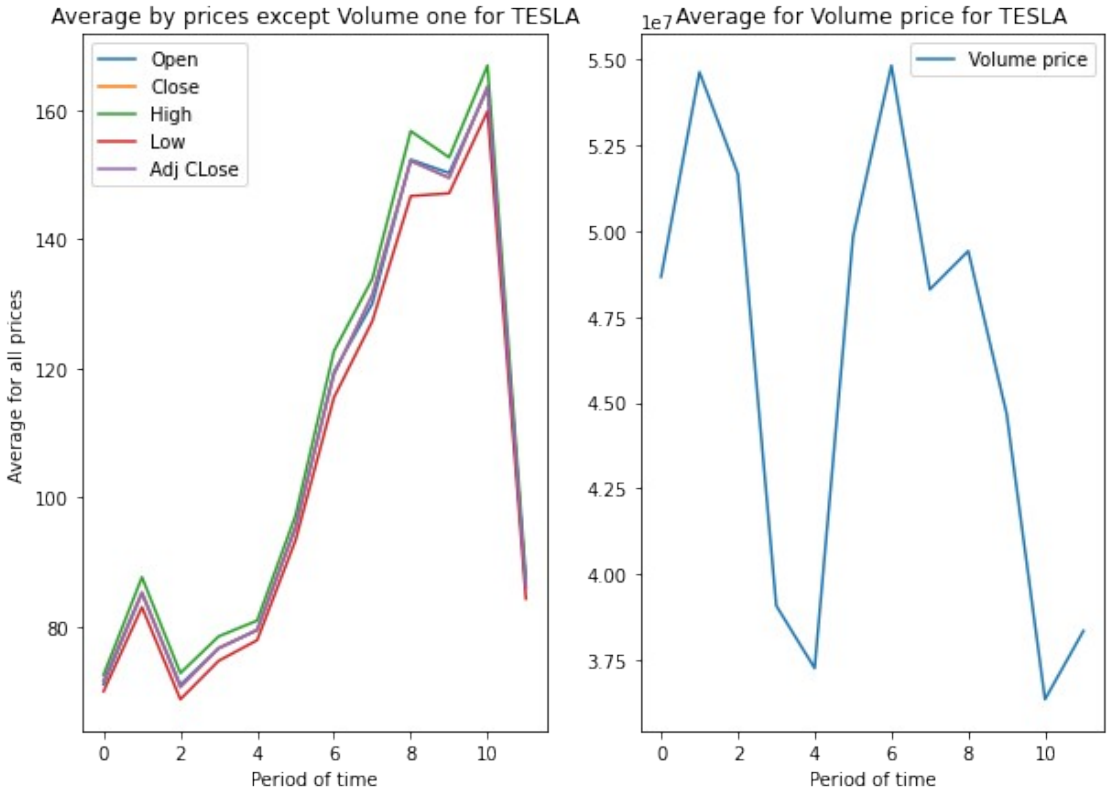
avg(Open)	avg(Close)
63.43669347269127	63.46198397328654

Average open price and close price in 2019

avg(Open)	avg(Close)

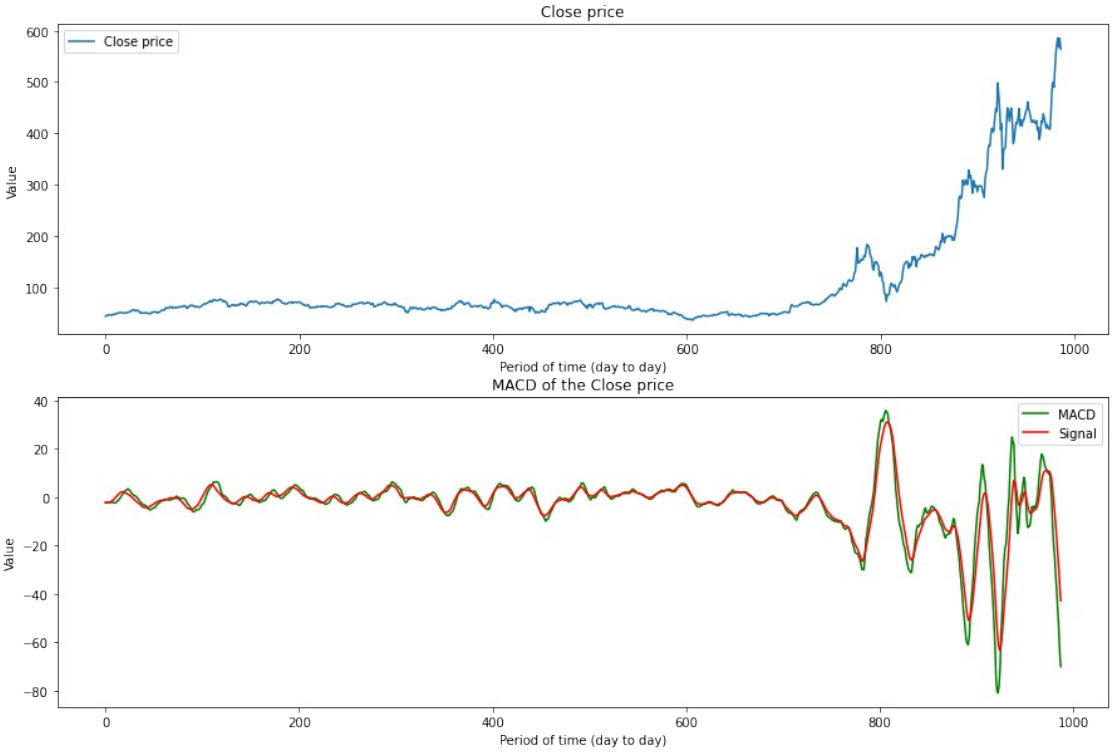
```
|54.60562690855965|54.706039686051625|
+-----------------+------------------+

Average open price and close price in 2020
+-----------------+------------------+
|        avg(Open)|        avg(Close)|
+-----------------+------------------+
|259.2505243980833|259.50230853035725|
+-----------------+------------------+
```

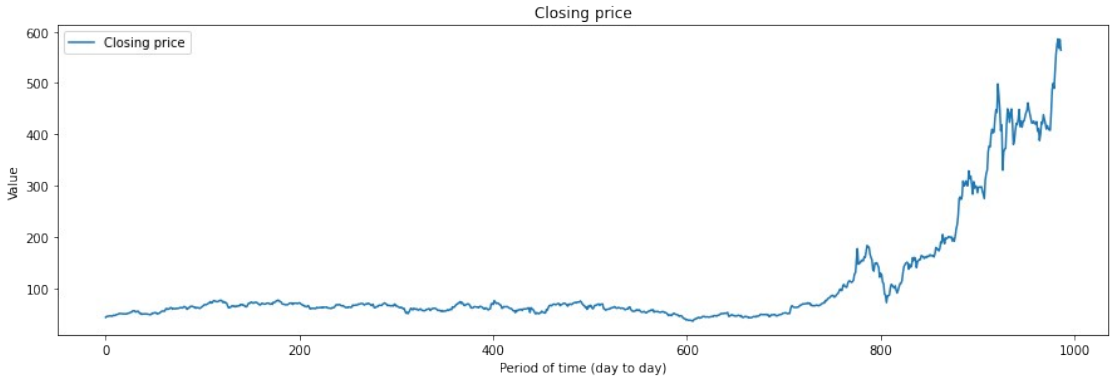


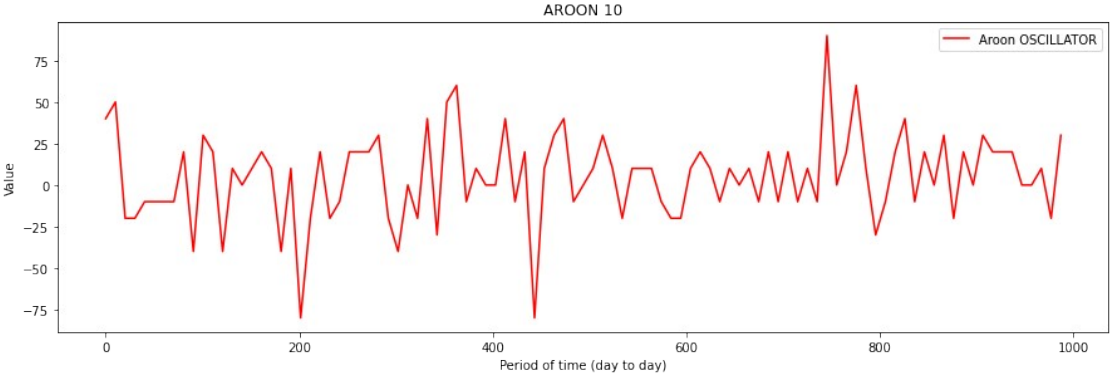


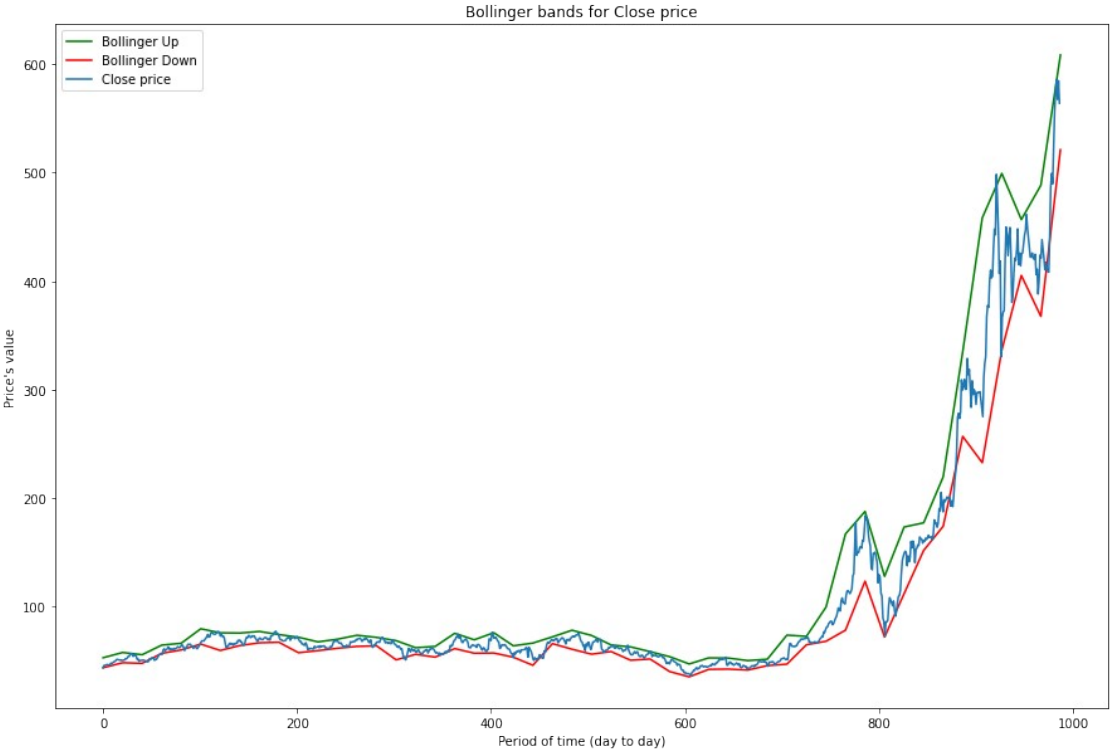
==========

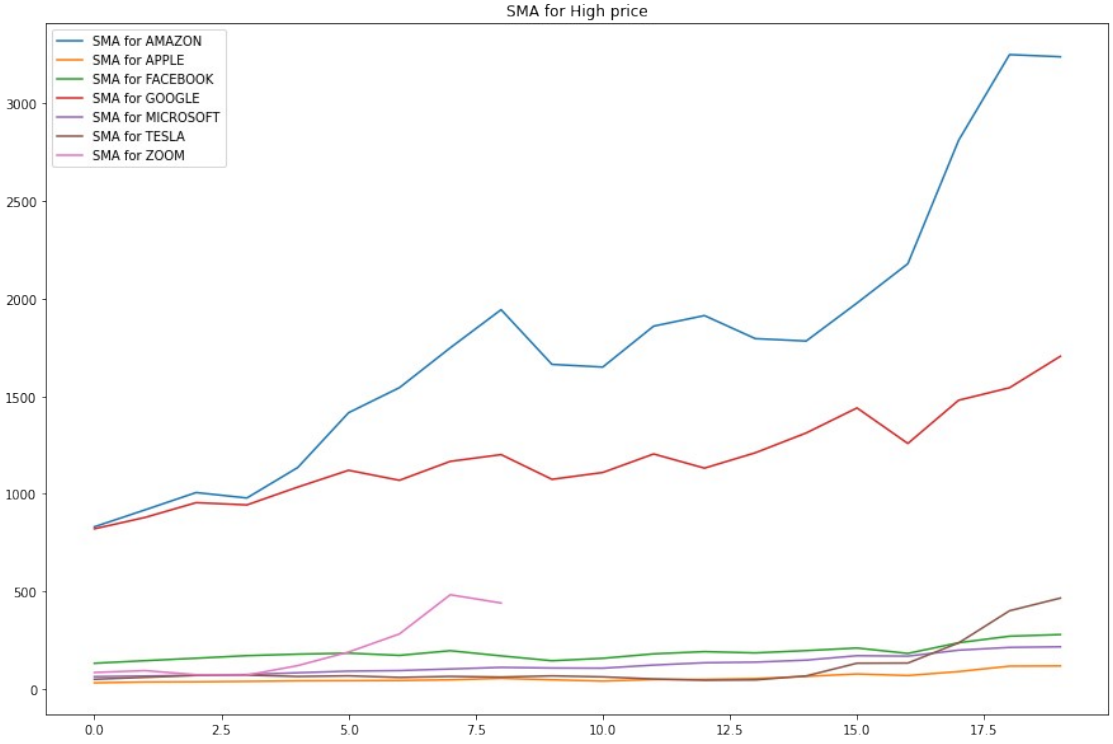
Moving Average for all prices

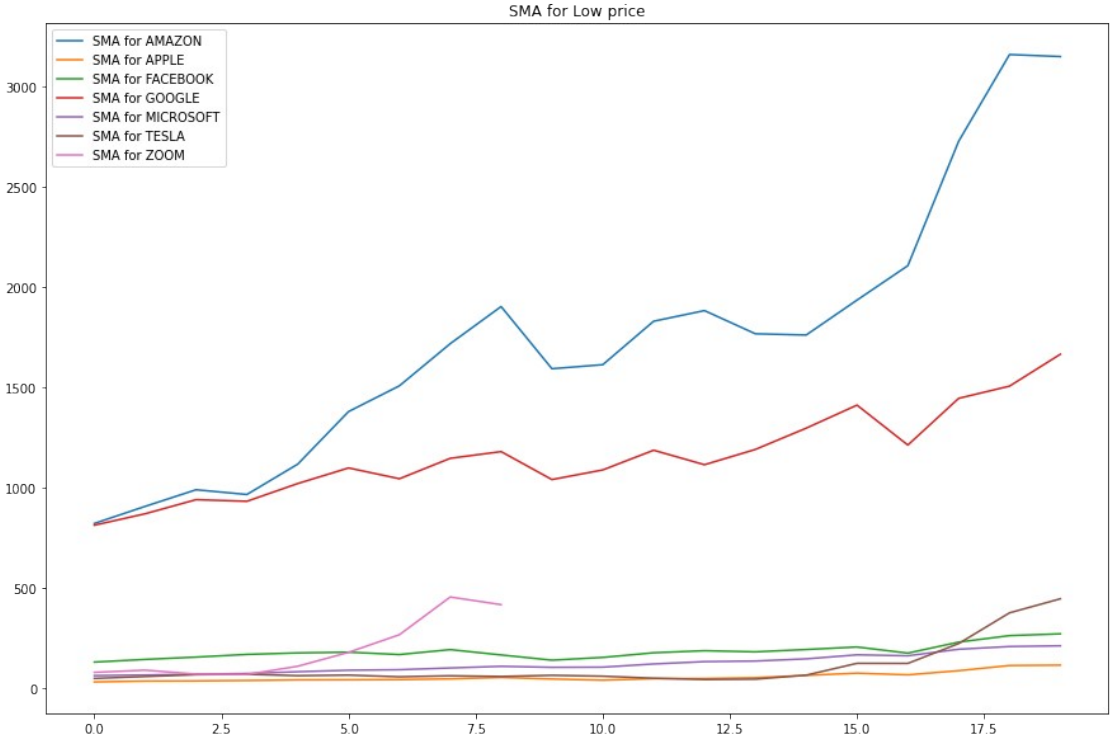


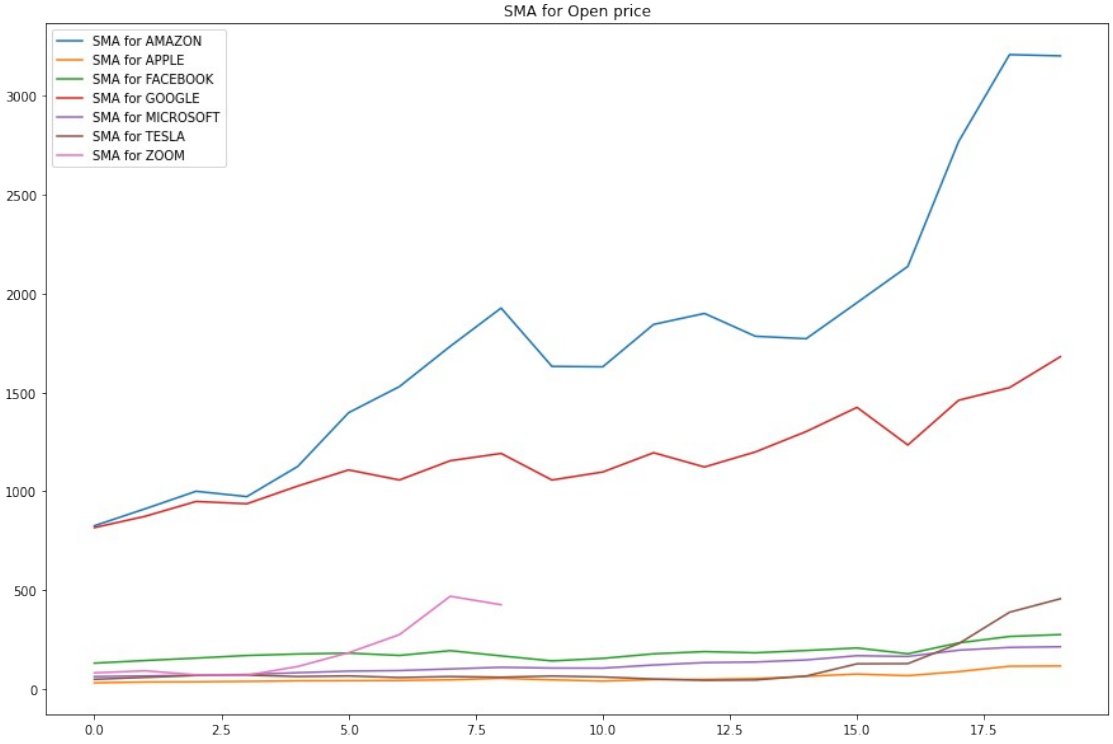

Closing price
Closing price
Value
Period of time (day to day)

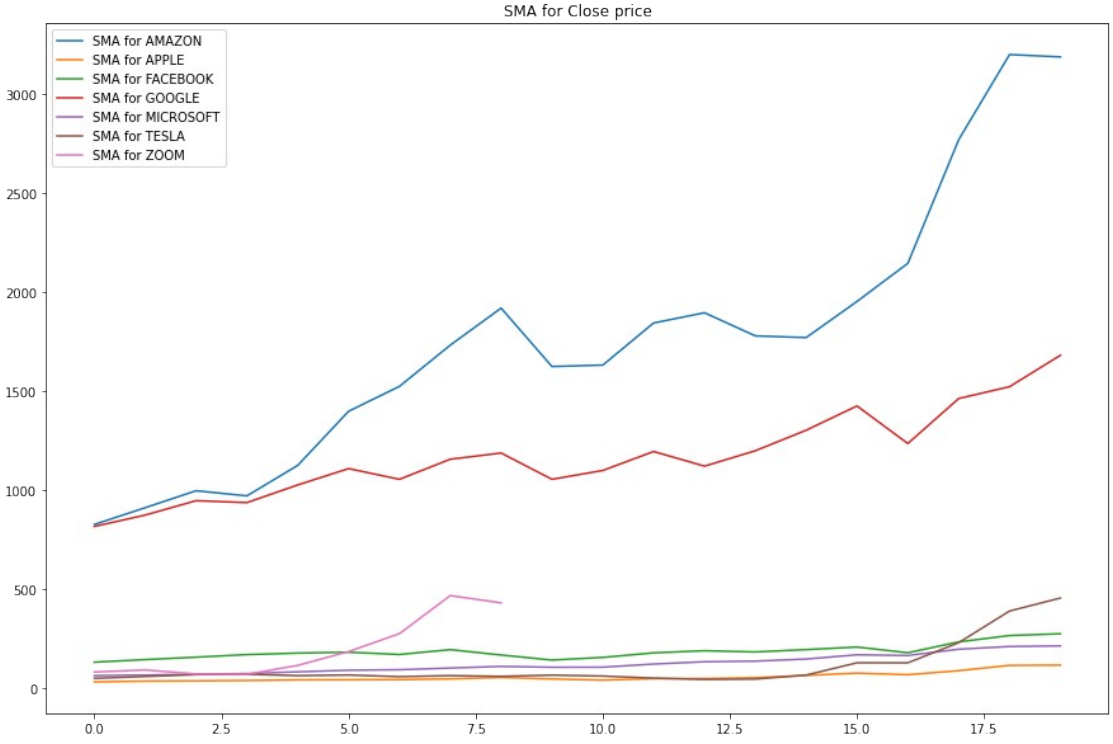

AROON 10
Aroon OSCILLATOR
Value
Period of time (day to day)

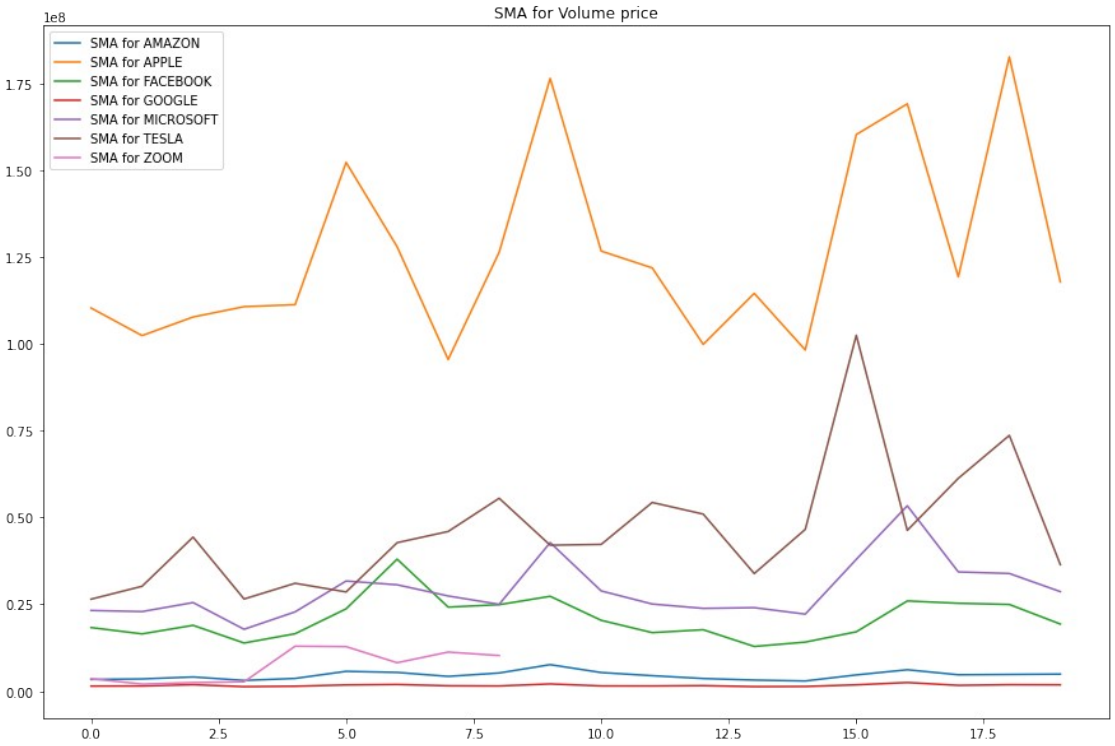

Bollinger bands for Close price
Bollinger Up
Bollinger Down
Close price
Price's value
Period of time (day to day)

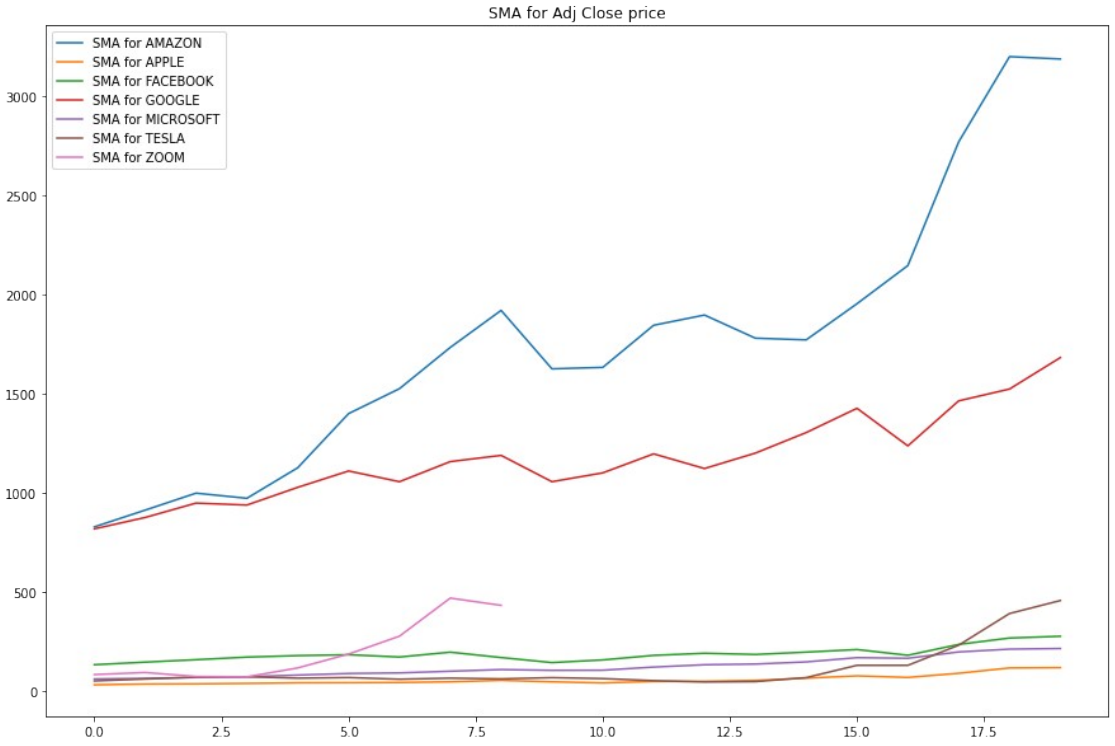

SMA for High price
SMA for AMAZON
SMA for APPLE
SMA for FACEBOOK
SMA for GOOGLE
SMA for MICROSOFT
SMA for TESLA
SMA for ZOOM
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


SMA for Low price
SMA for AMAZON
SMA for APPLE
SMA for FACEBOOK
SMA for GOOGLE
SMA for MICROSOFT
SMA for TESLA
SMA for ZOOM
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


SMA for Open price
SMA for AMAZON
SMA for APPLE
SMA for FACEBOOK
SMA for GOOGLE
SMA for MICROSOFT
SMA for TESLA
SMA for ZOOM
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

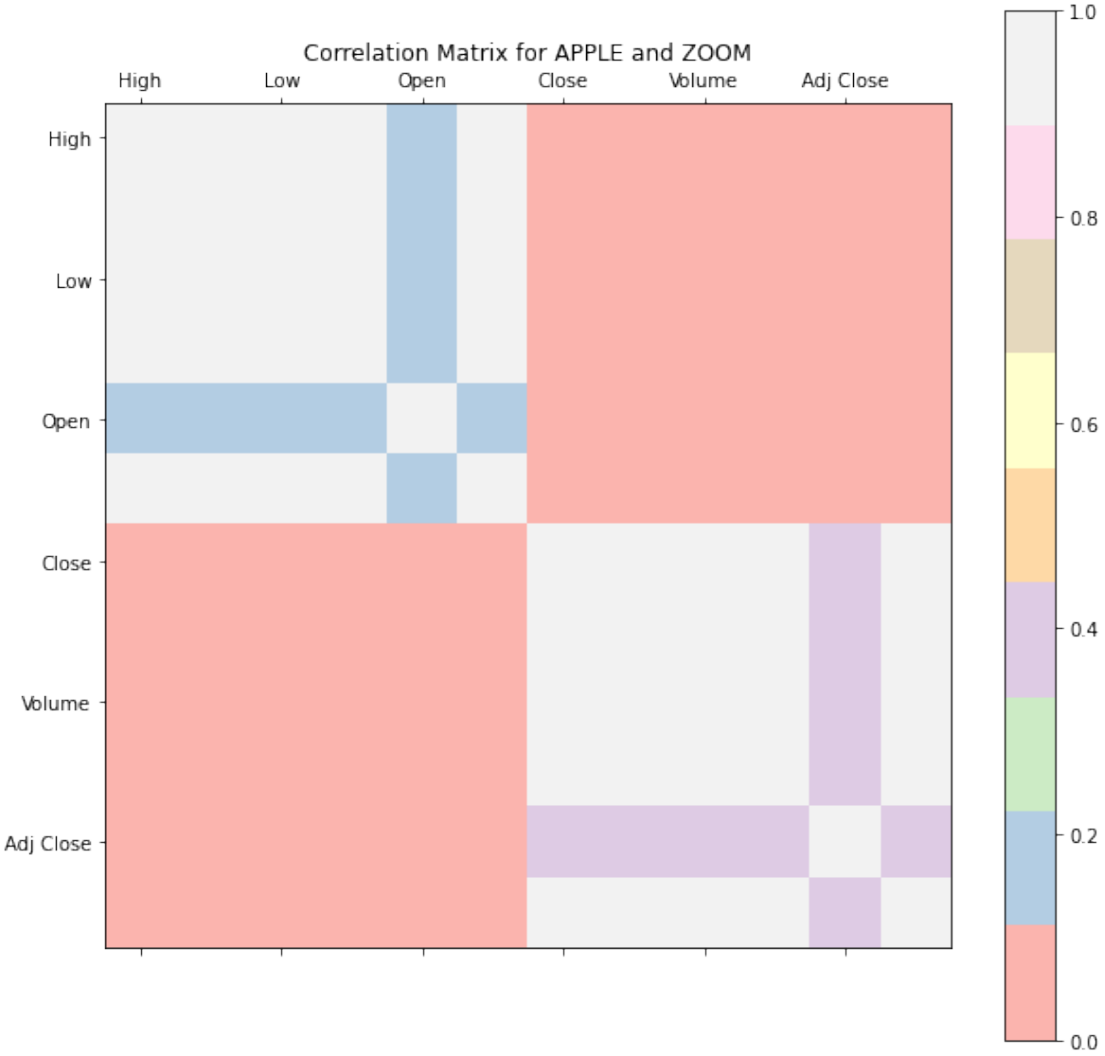

SMA for Close price
SMA for AMAZON
SMA for APPLE
SMA for FACEBOOK
SMA for GOOGLE
SMA for MICROSOFT
SMA for TESLA
SMA for ZOOM
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



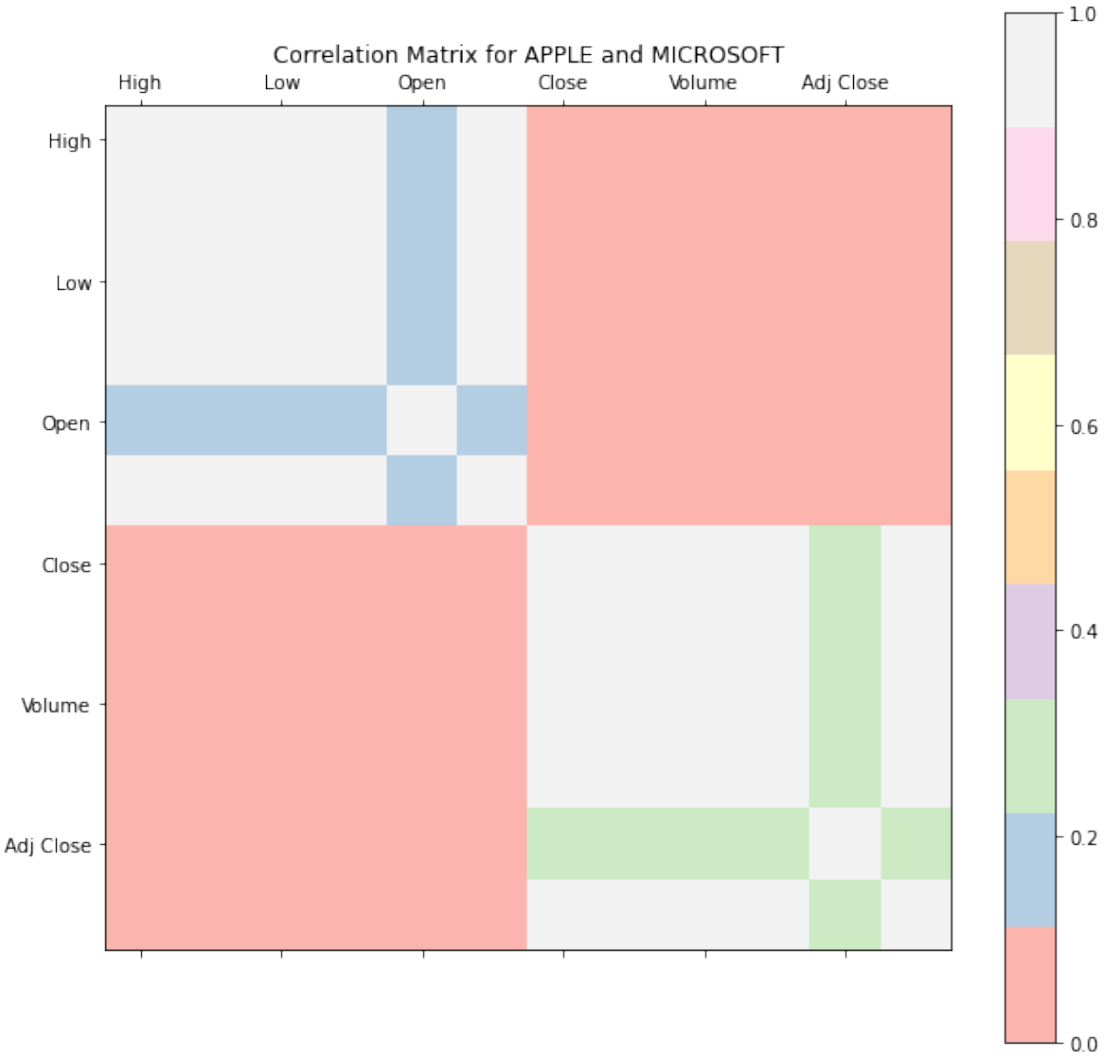


We want to see all correlations between the differents DataFrame created
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2', 'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']

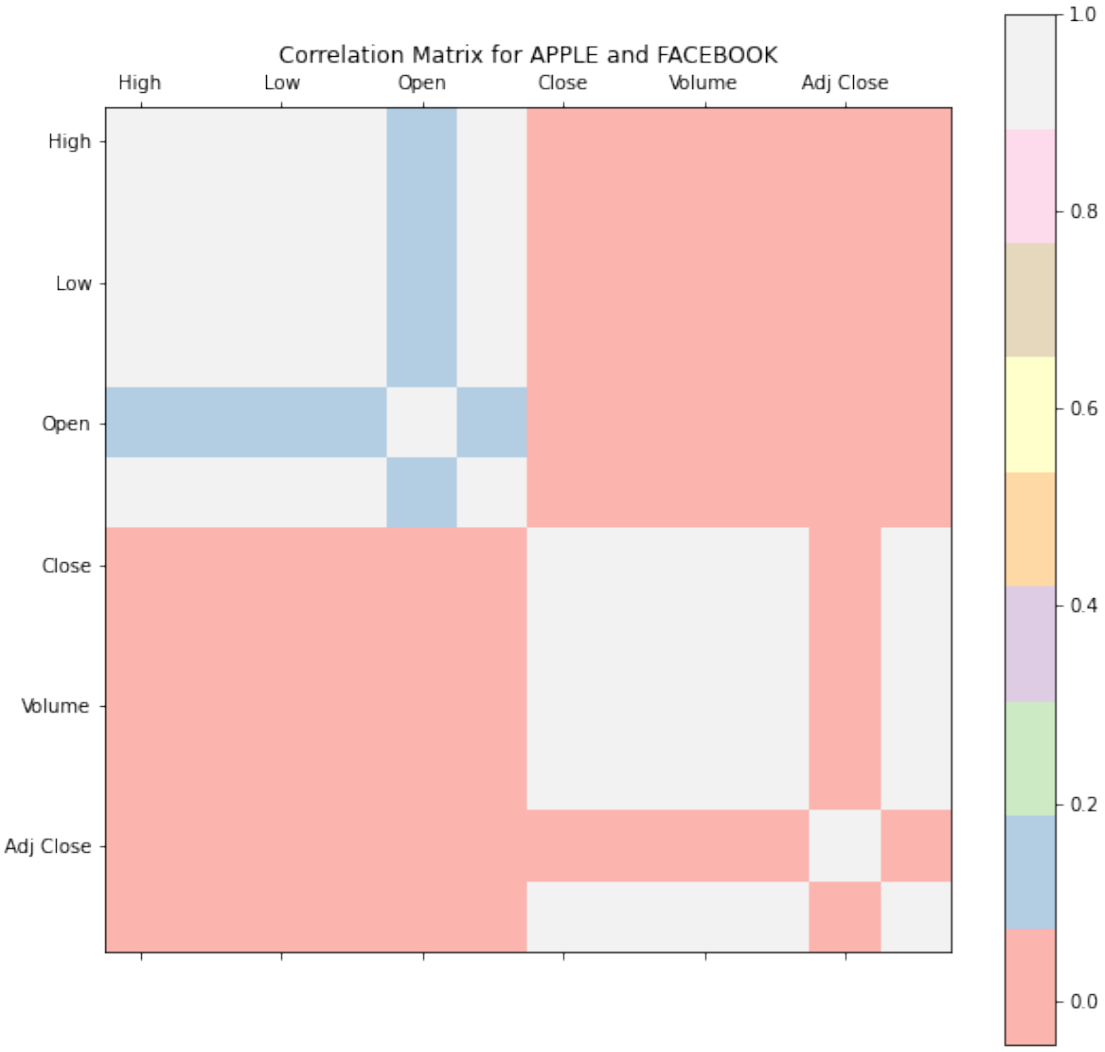
```
/home/alex/.local/lib/python3.8/site-packages/pyspark/sql/
context.py:125: FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
  warnings.warn(
/tmp/ipykernel_6301/617499620.py:14: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_xticklabels(['']+columns)
/tmp/ipykernel_6301/617499620.py:15: UserWarning: FixedFormatter
should only be used together with FixedLocator
  ax.set_yticklabels(['']+columns)
```



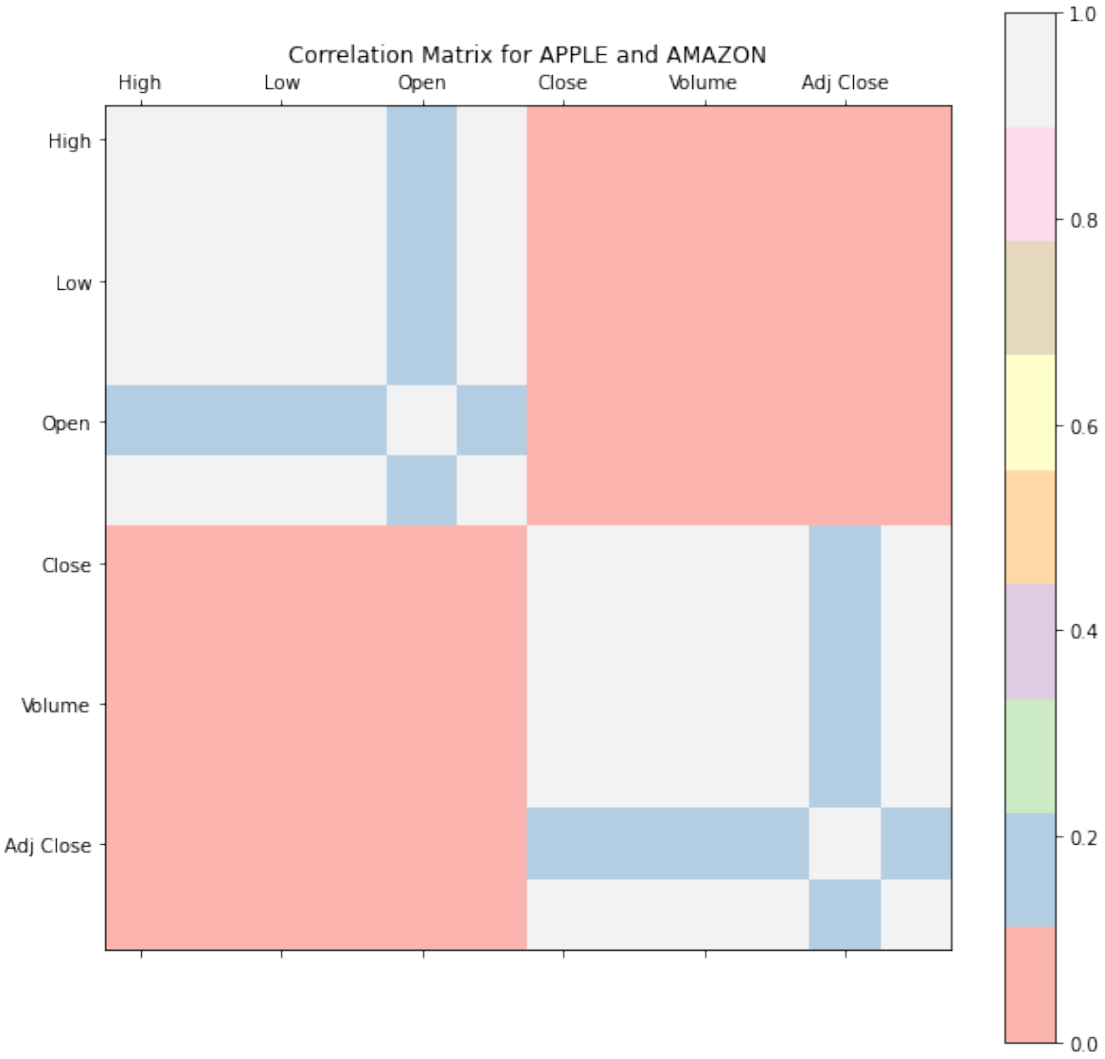

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



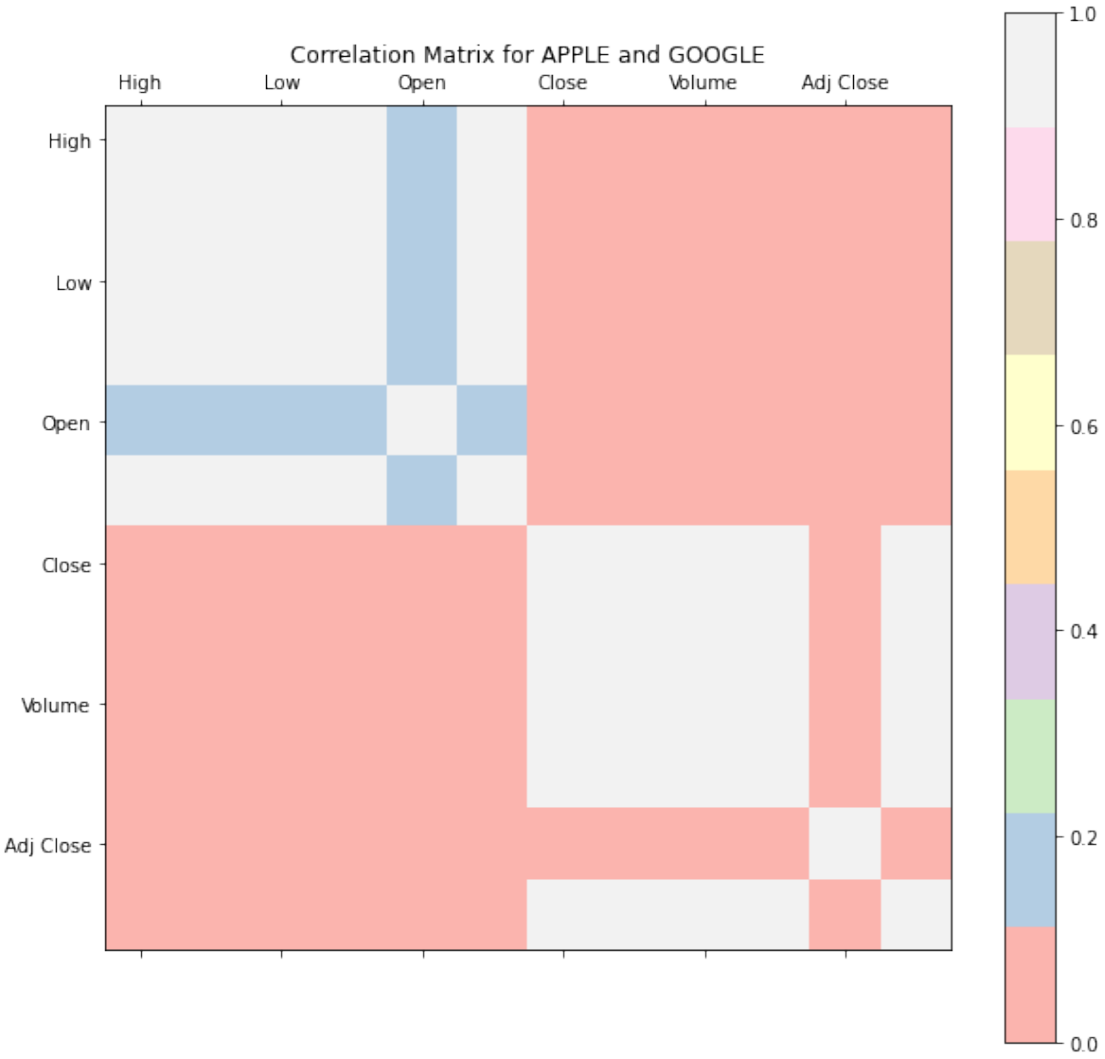

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



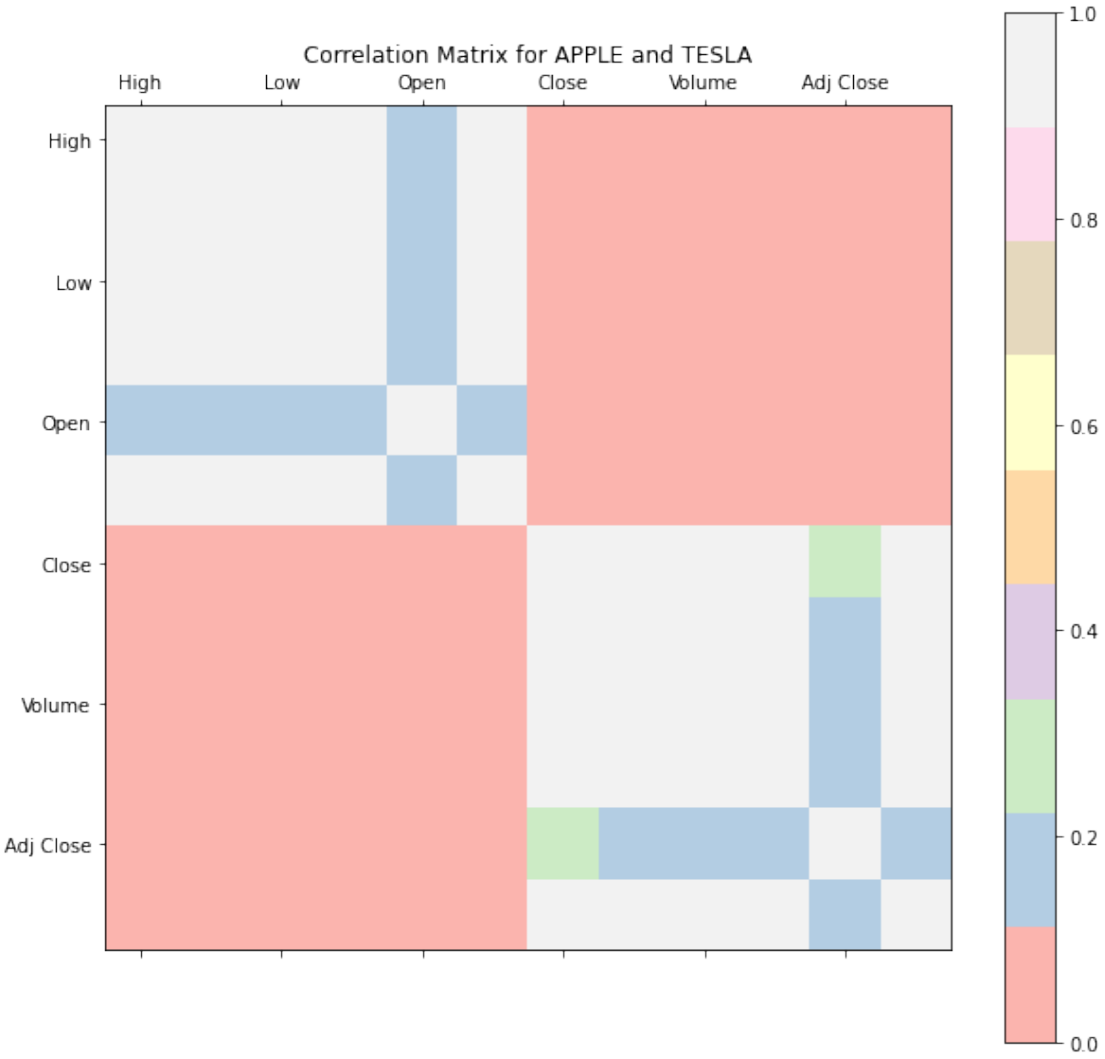

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



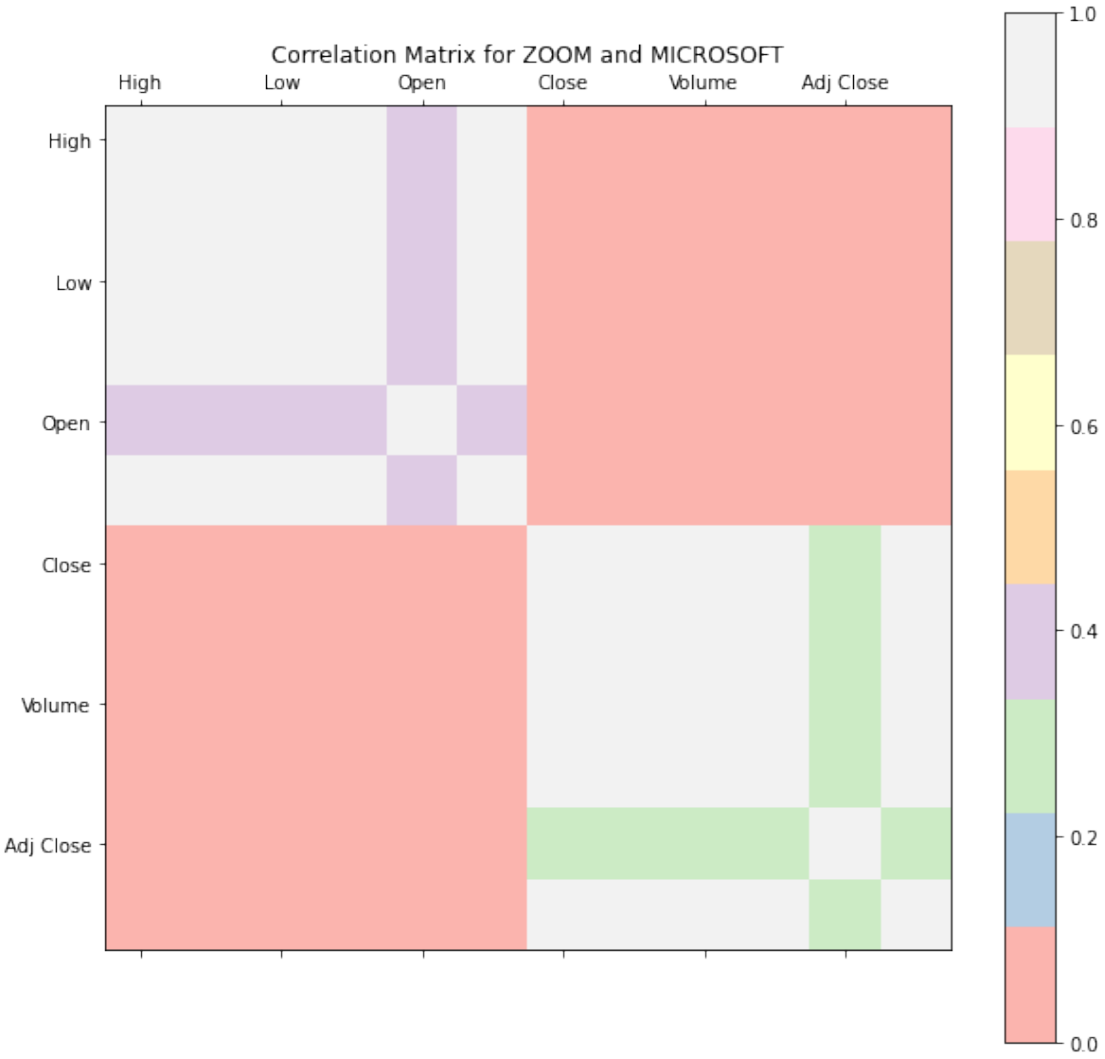

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



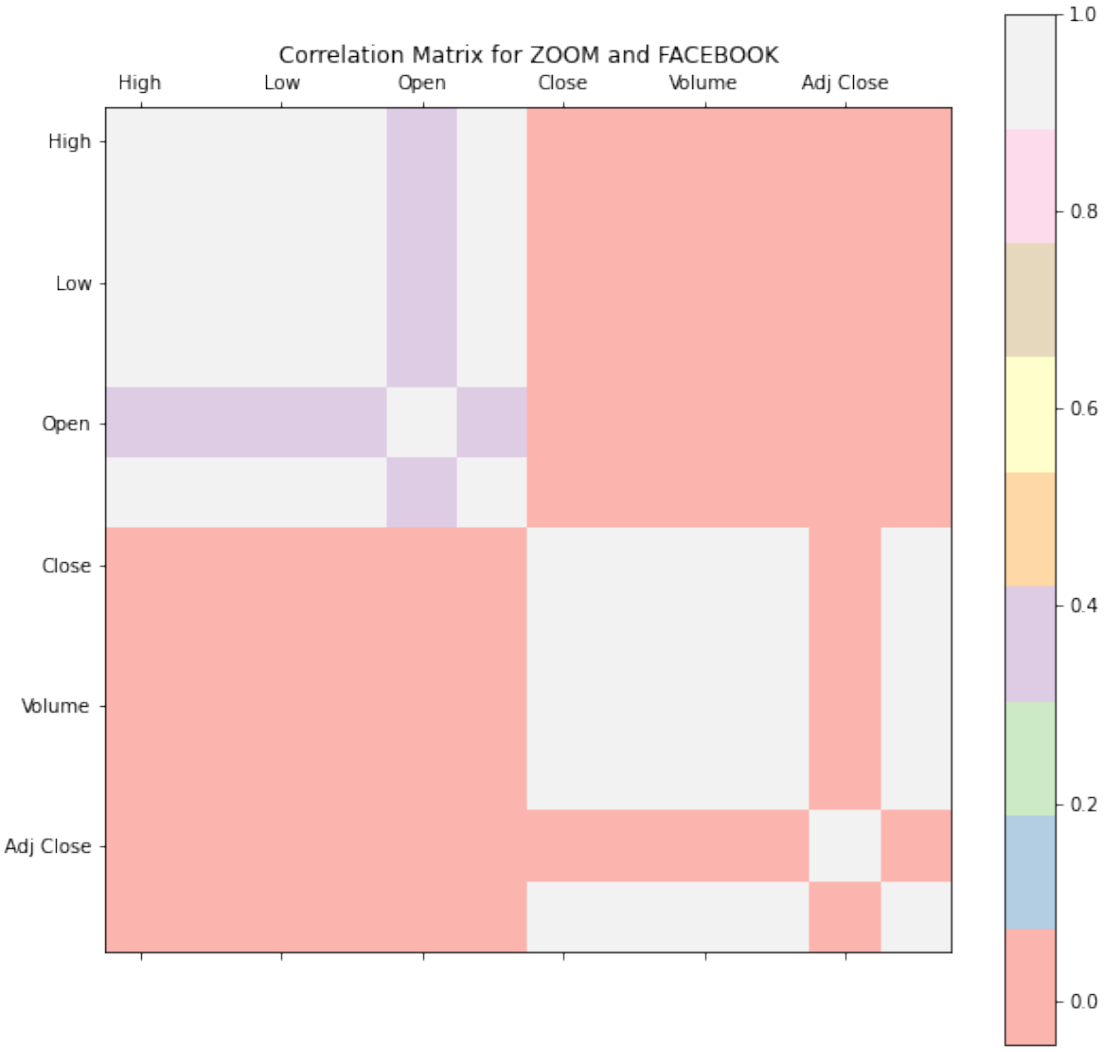

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



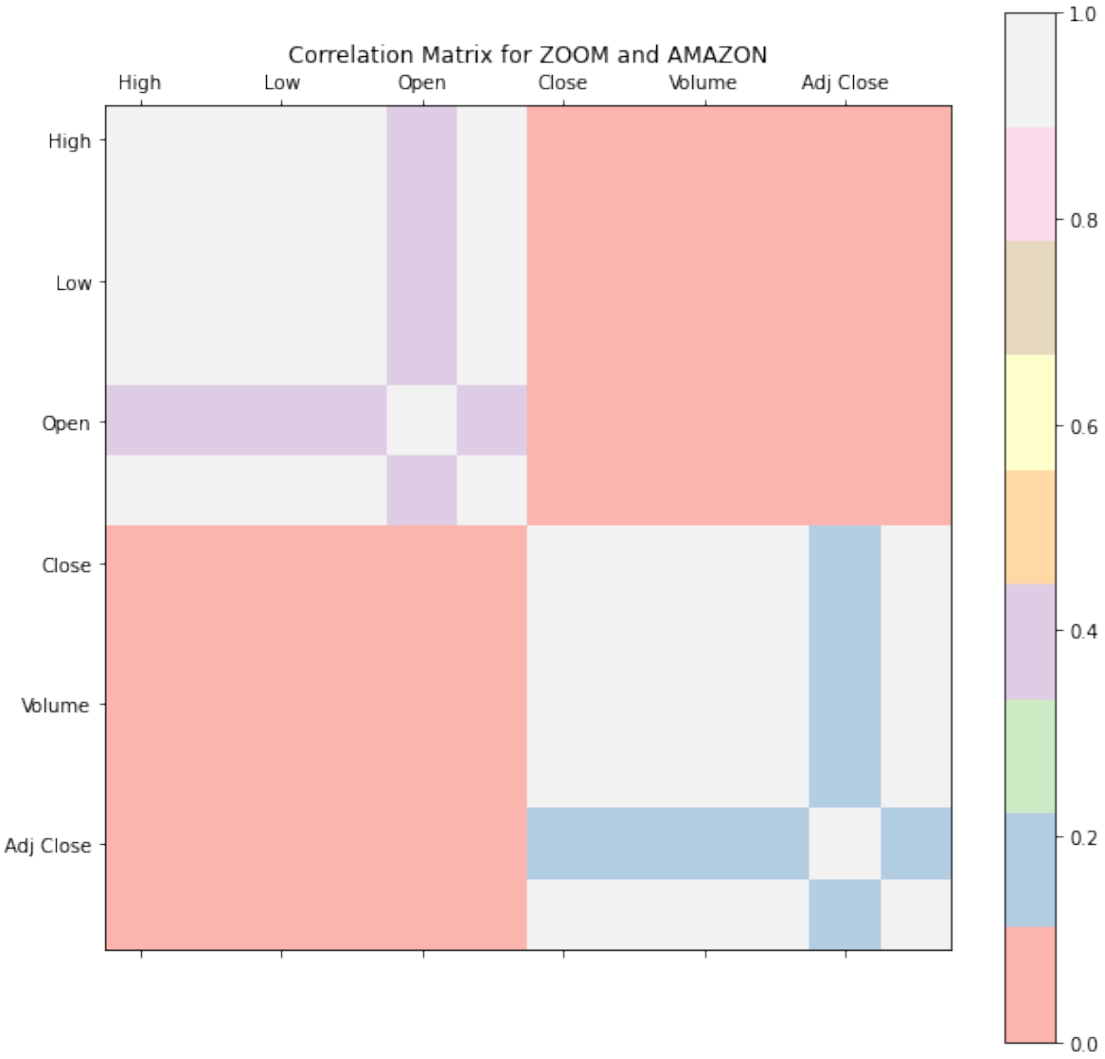

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



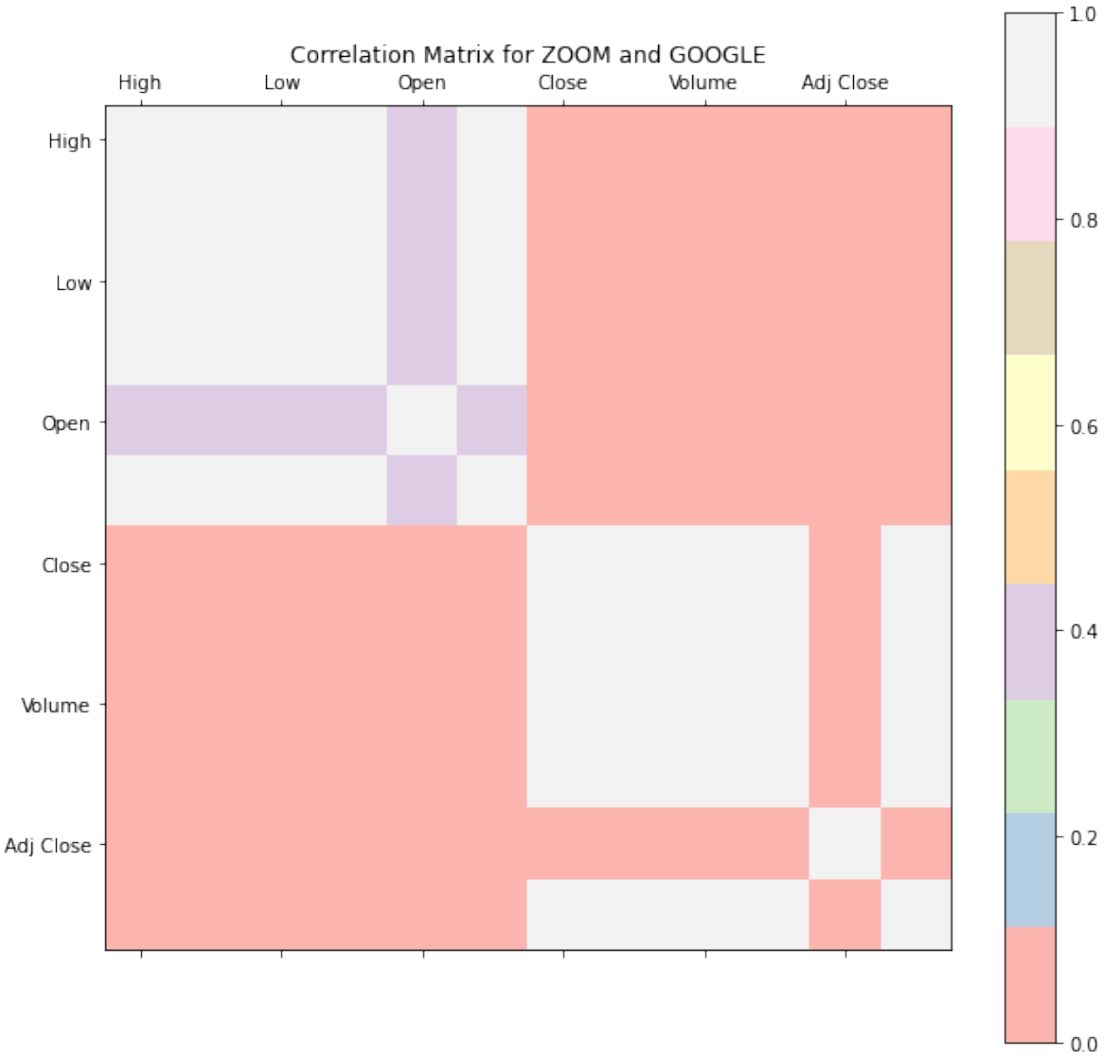

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



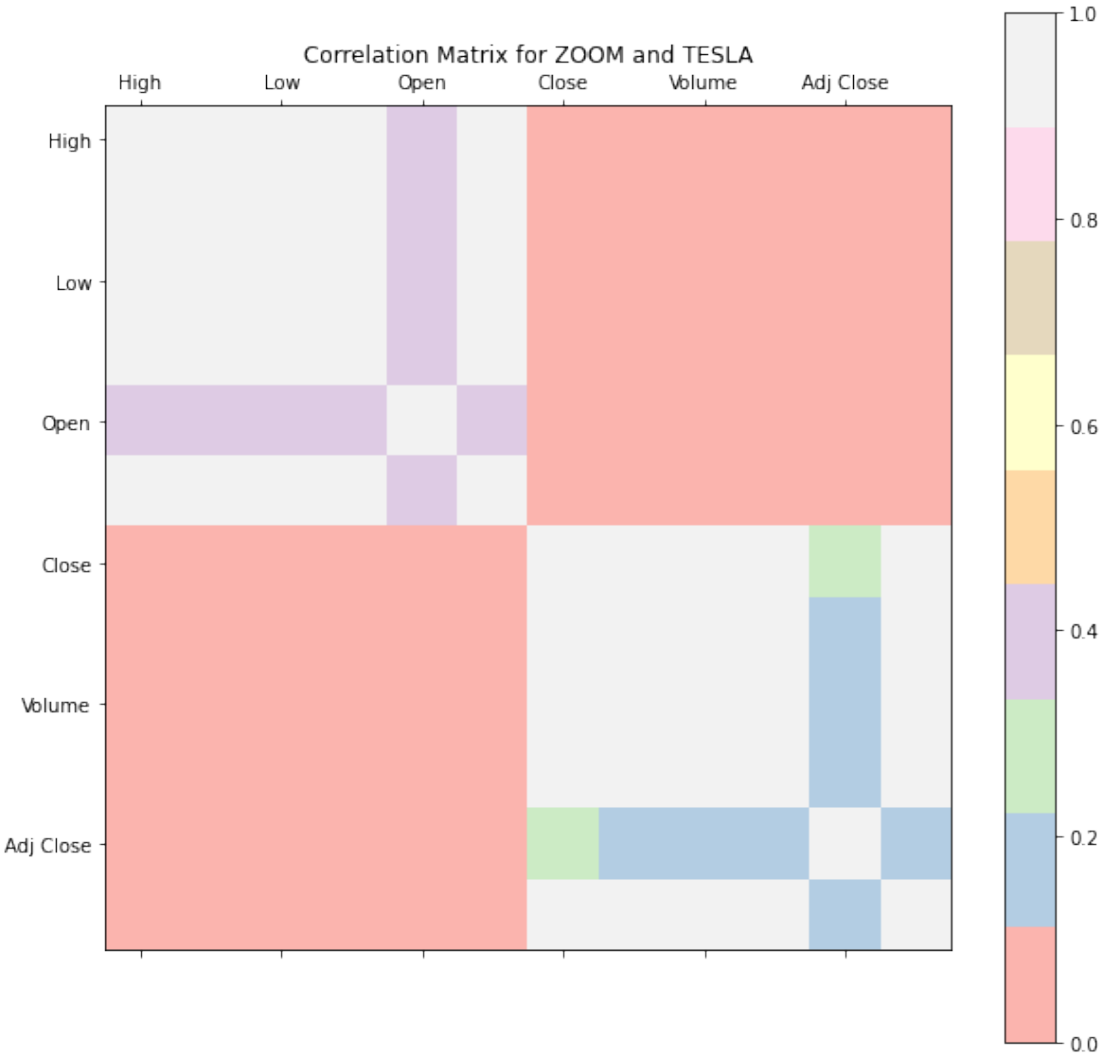

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



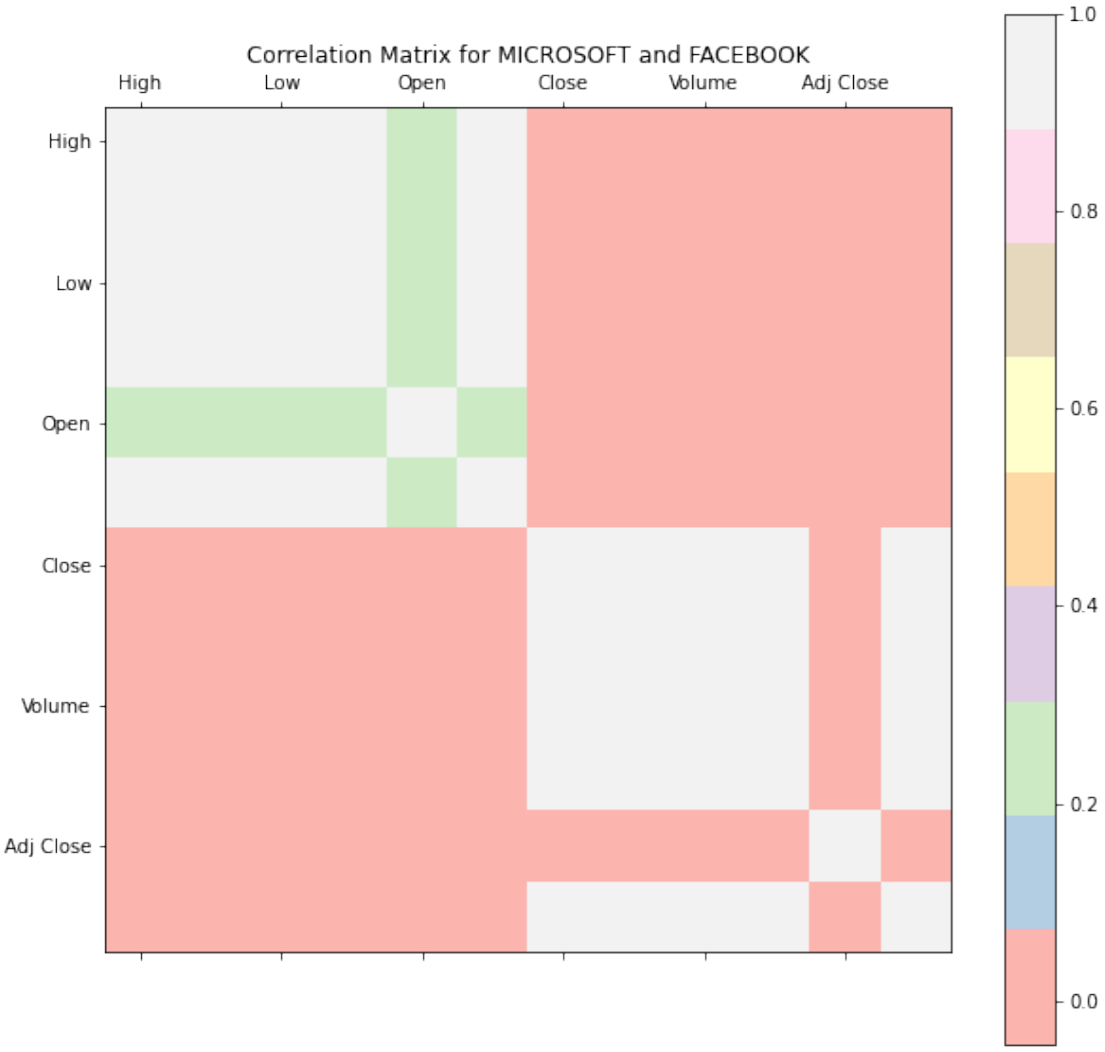

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



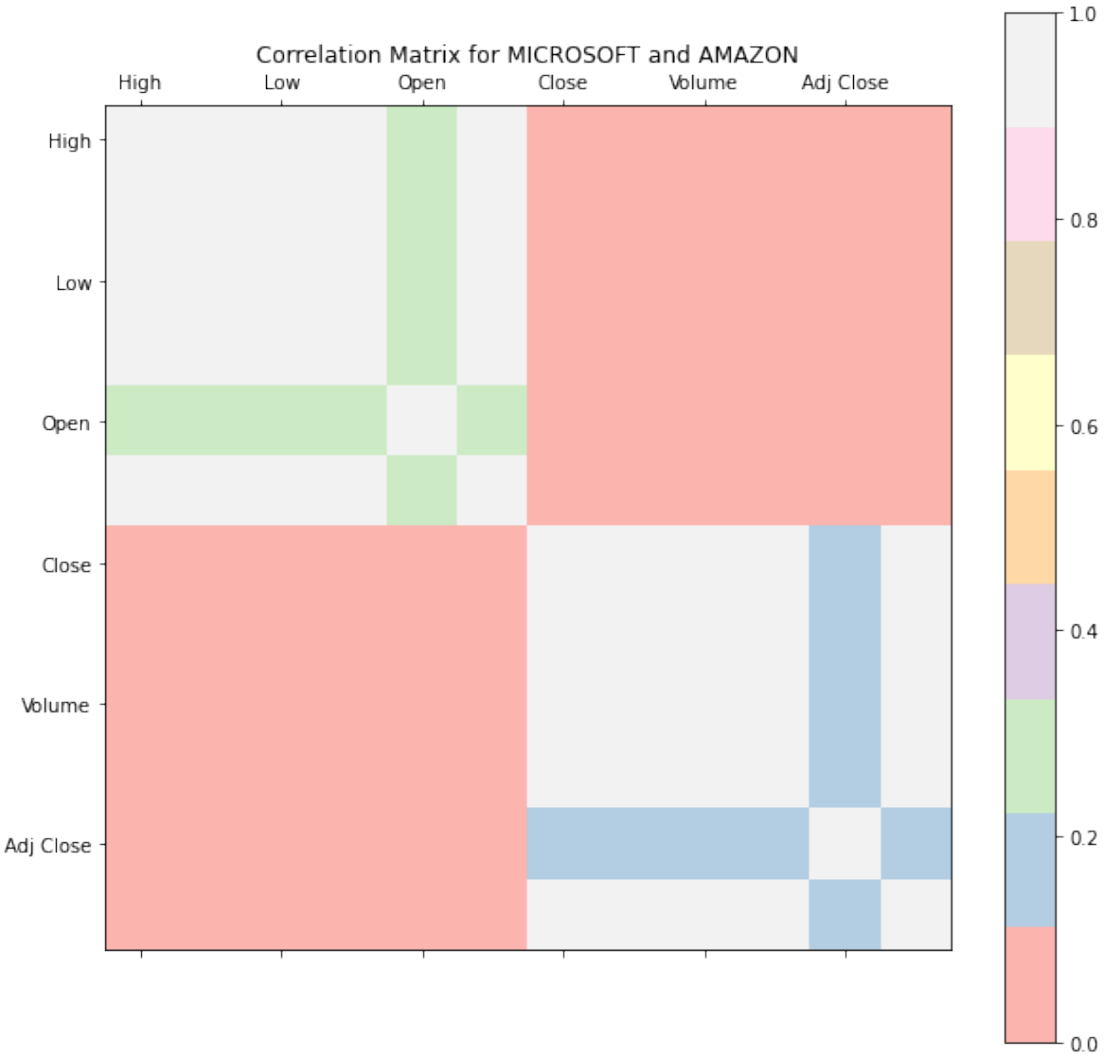

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



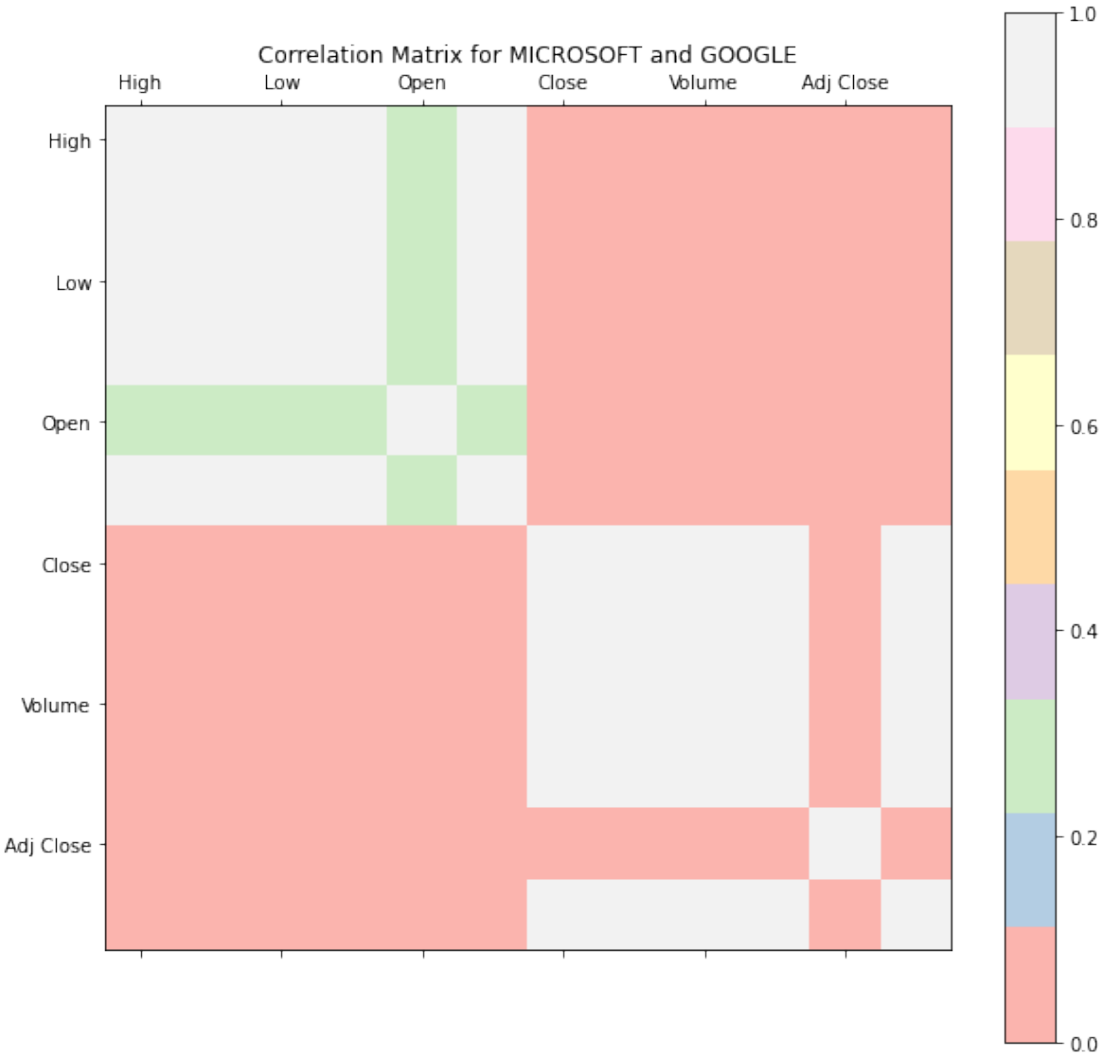

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



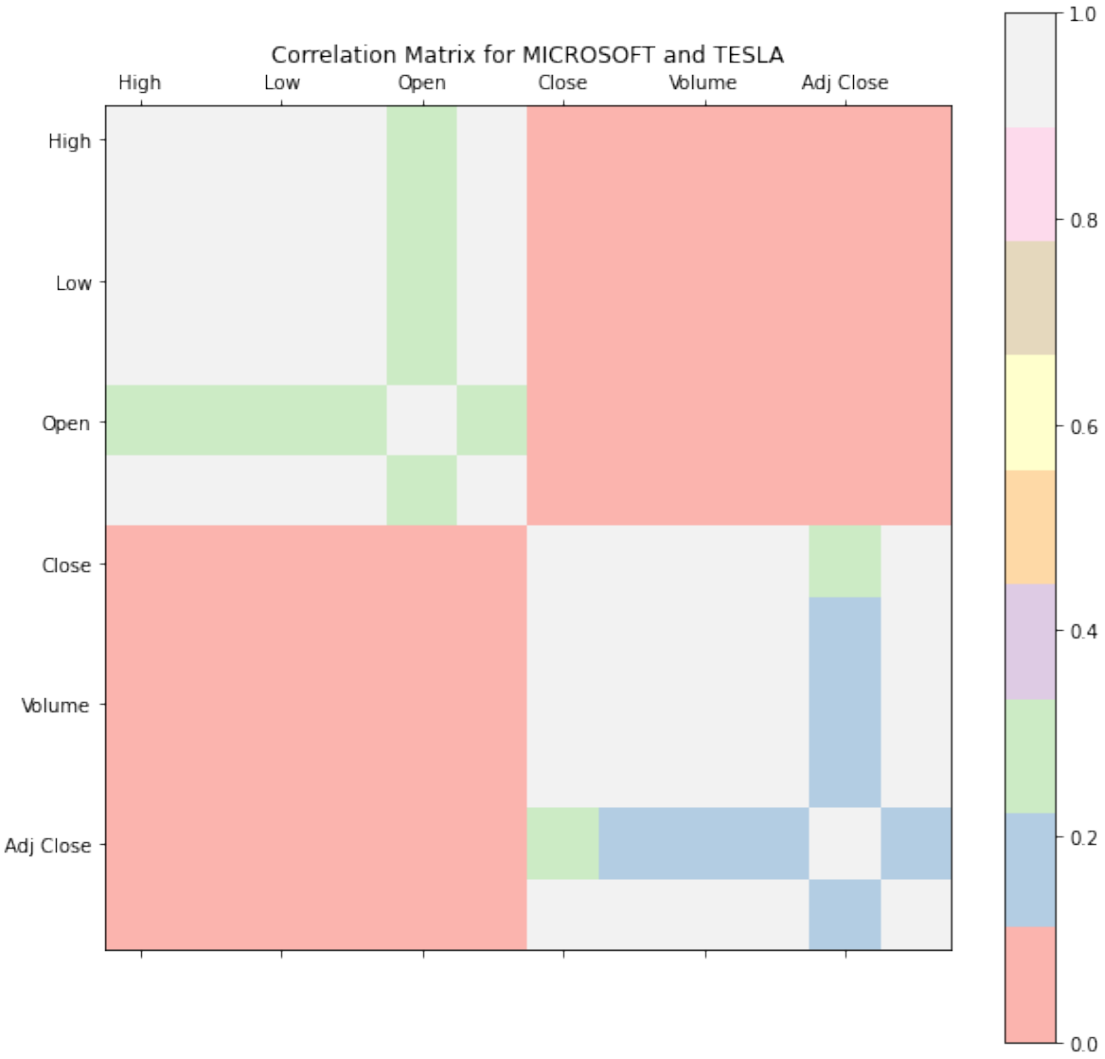

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



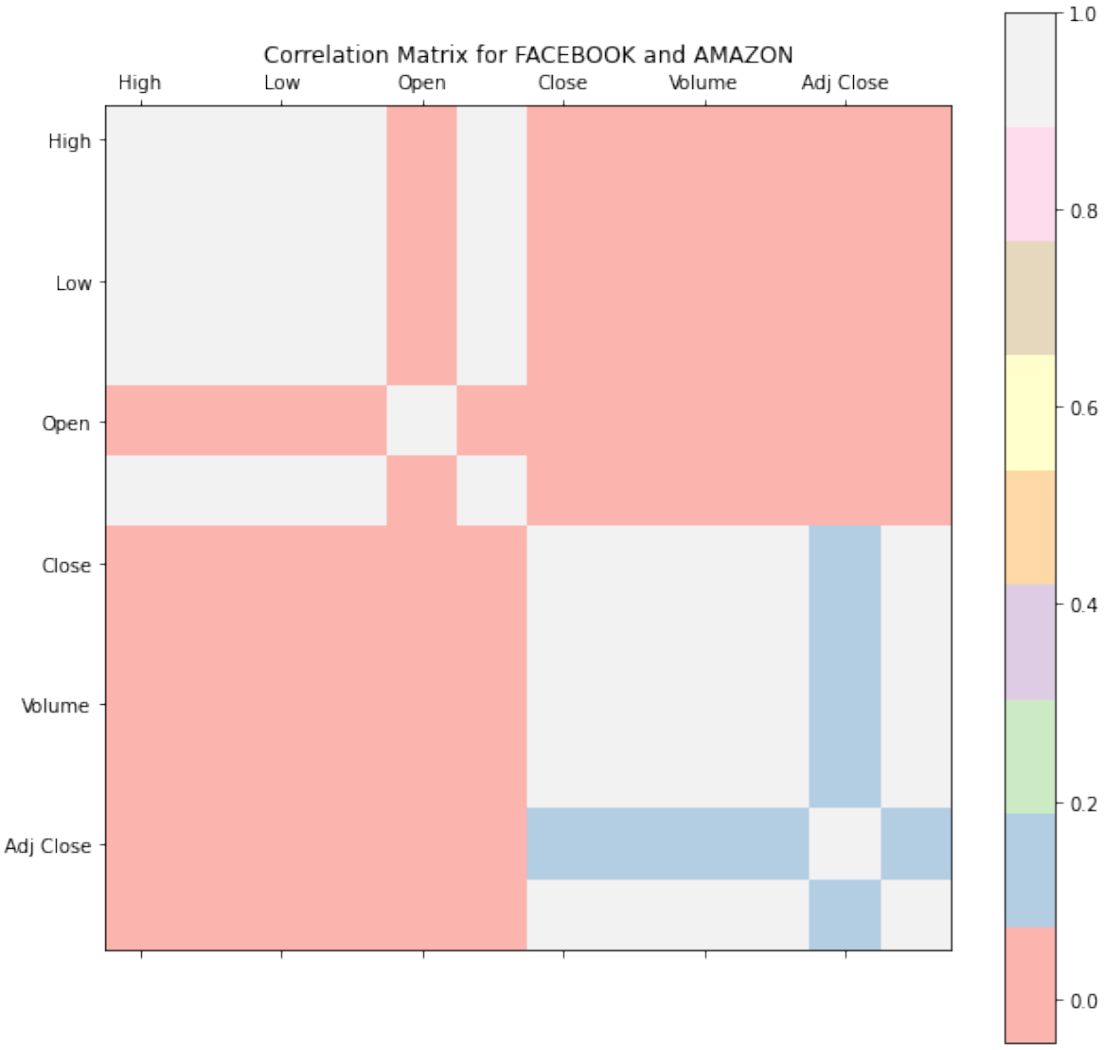

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



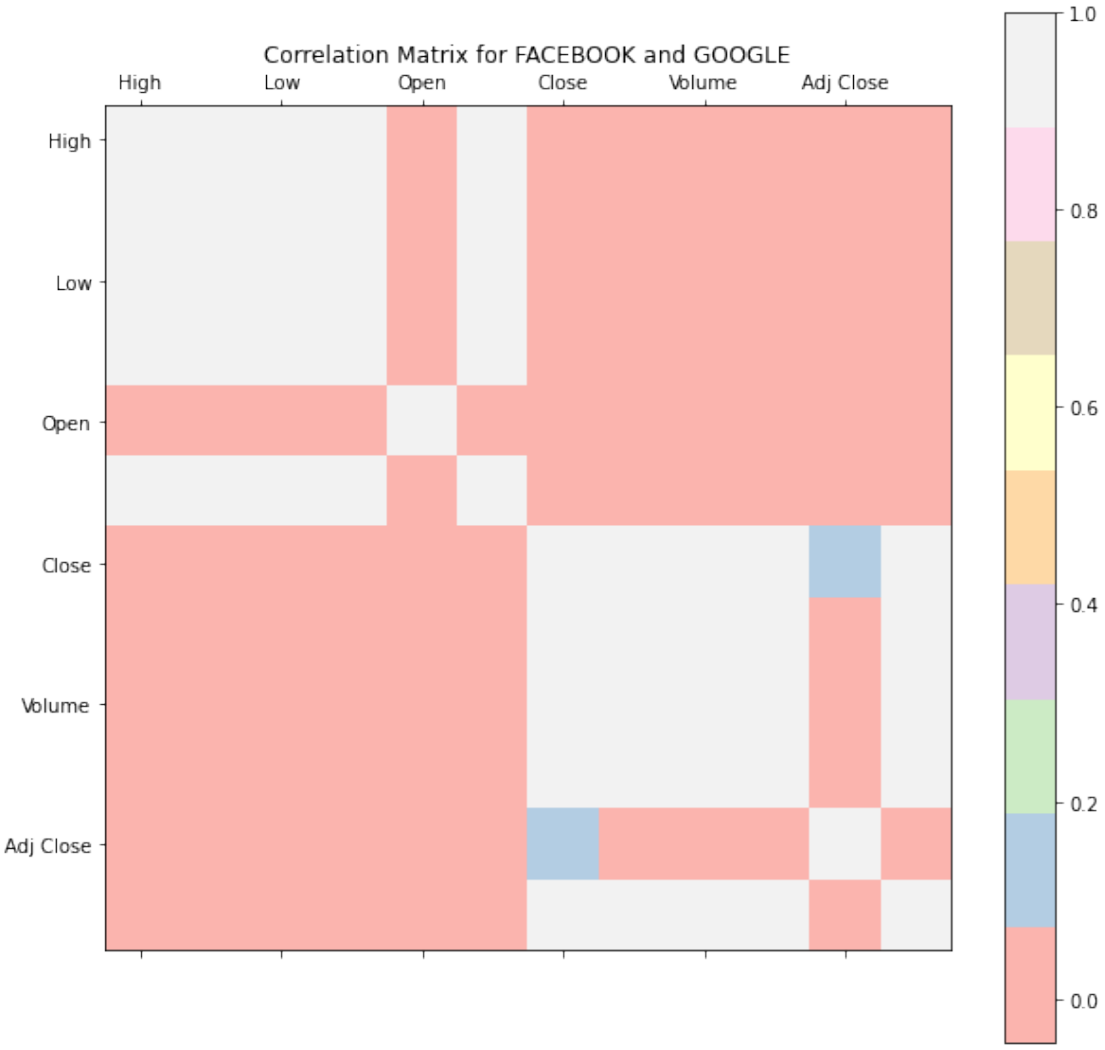

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



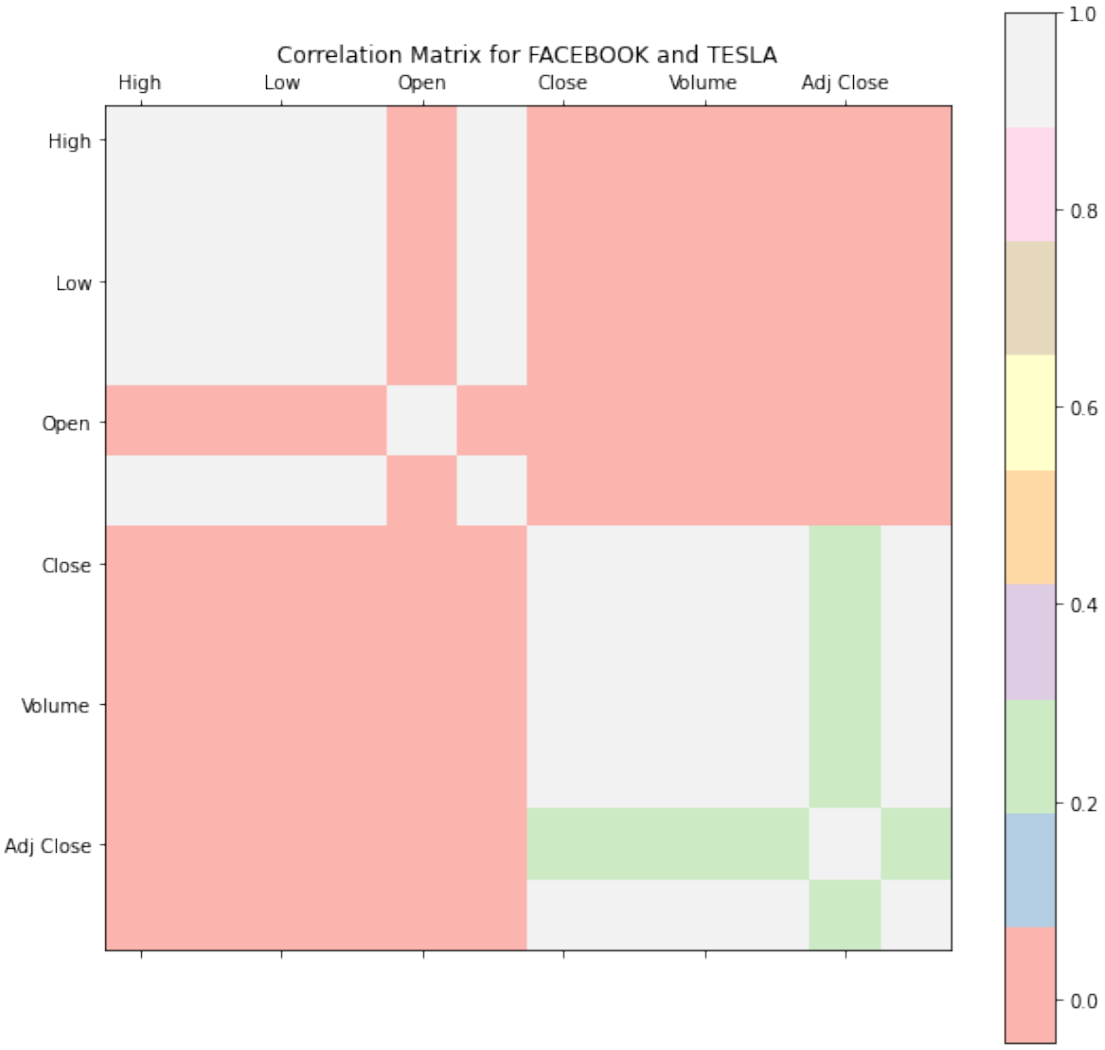

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



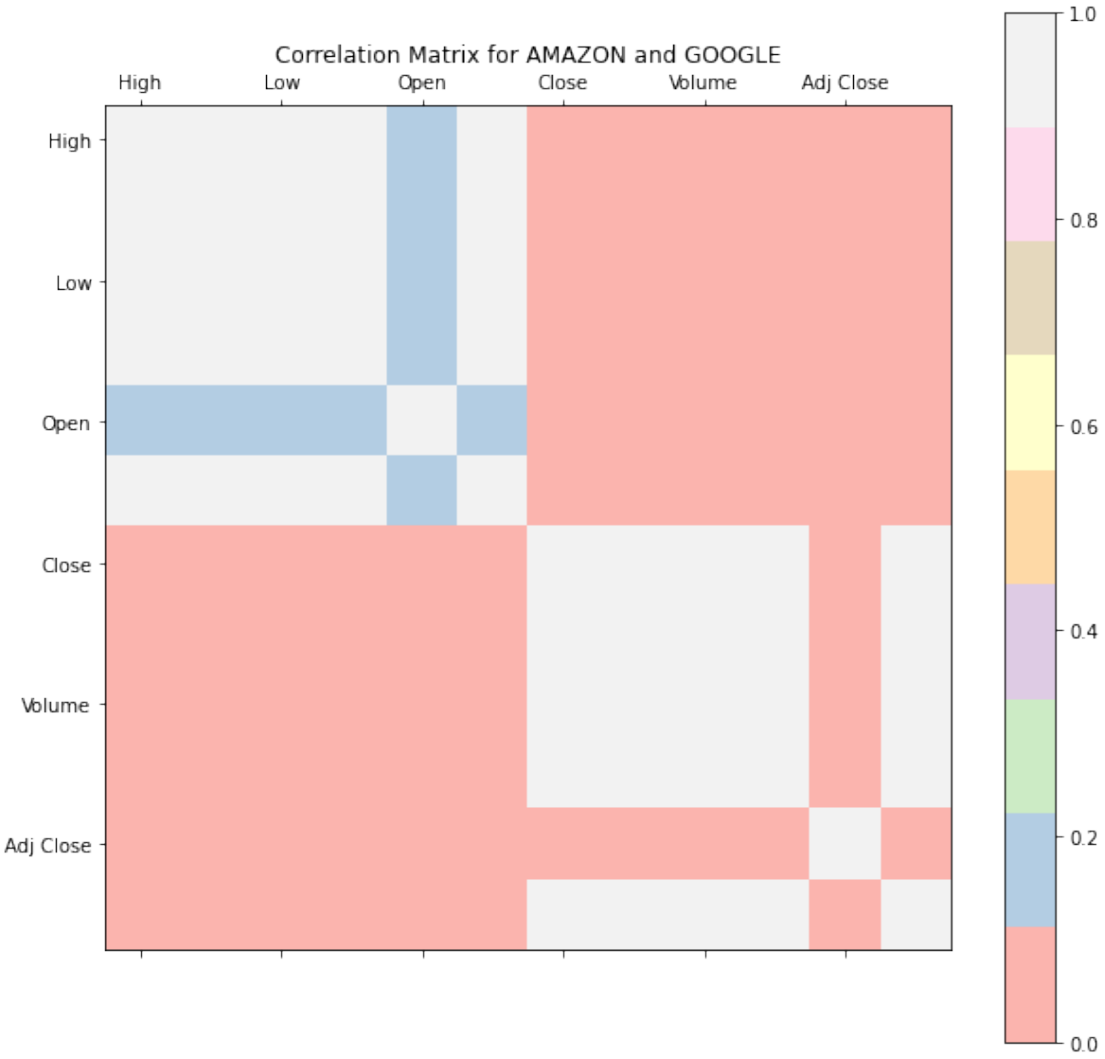

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



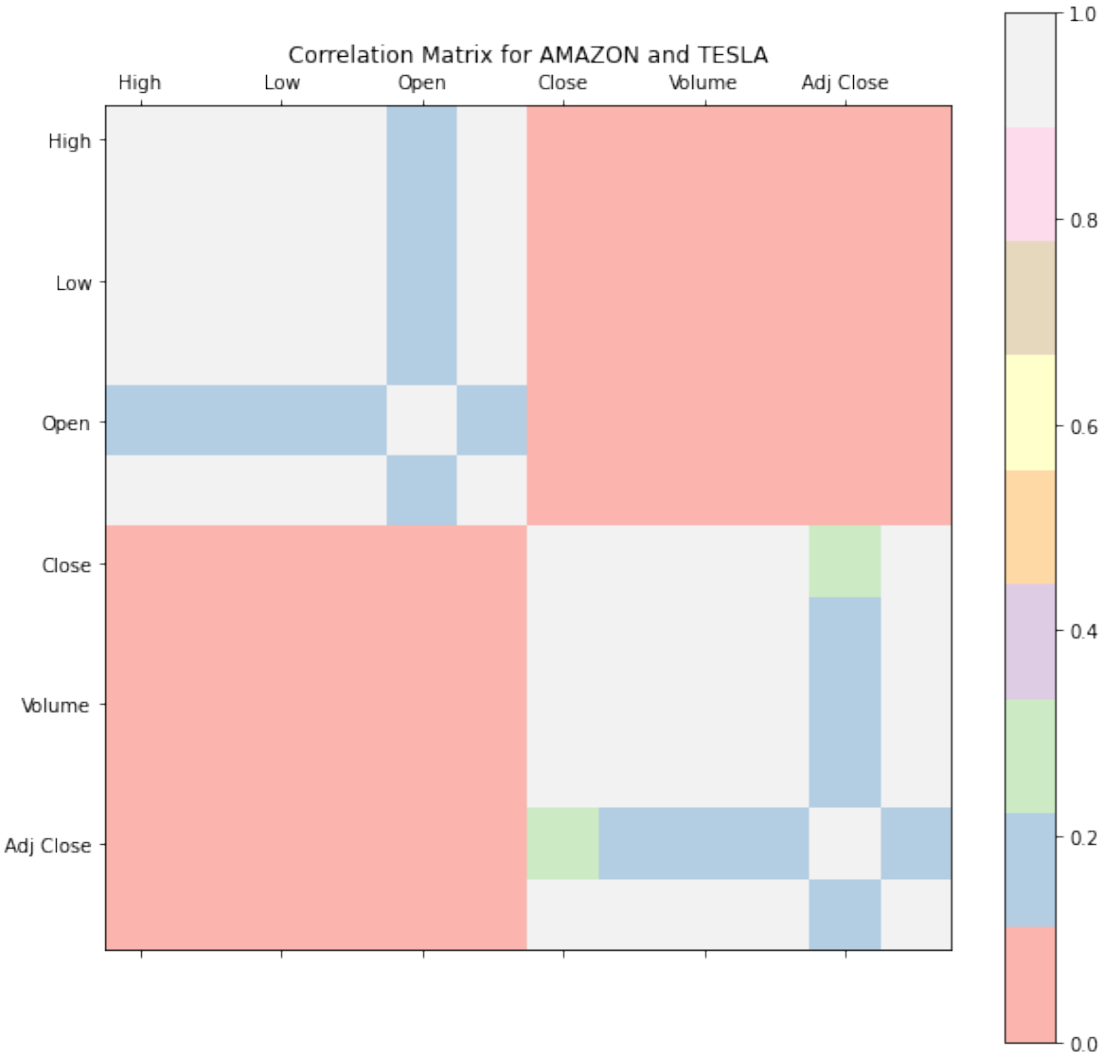

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



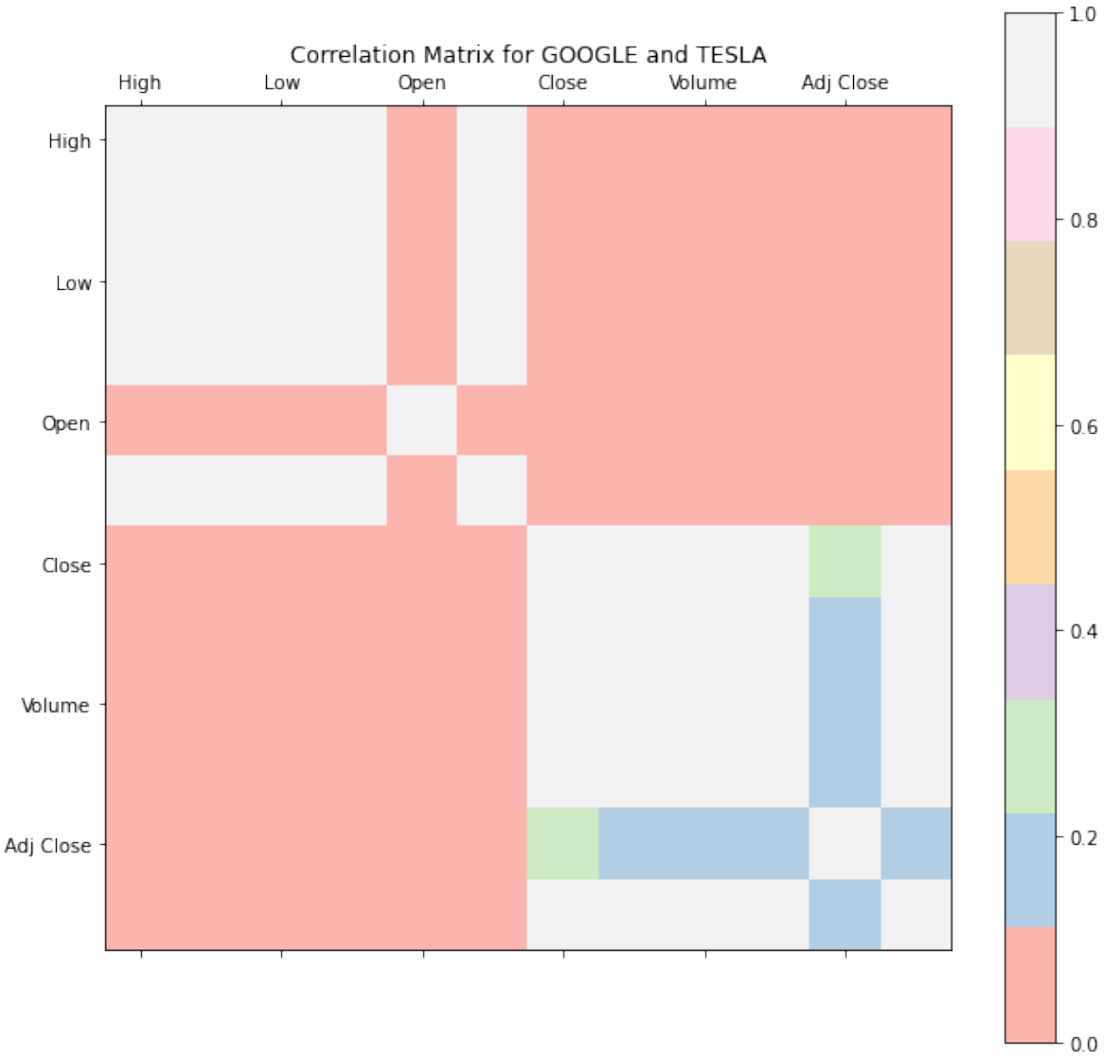

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```



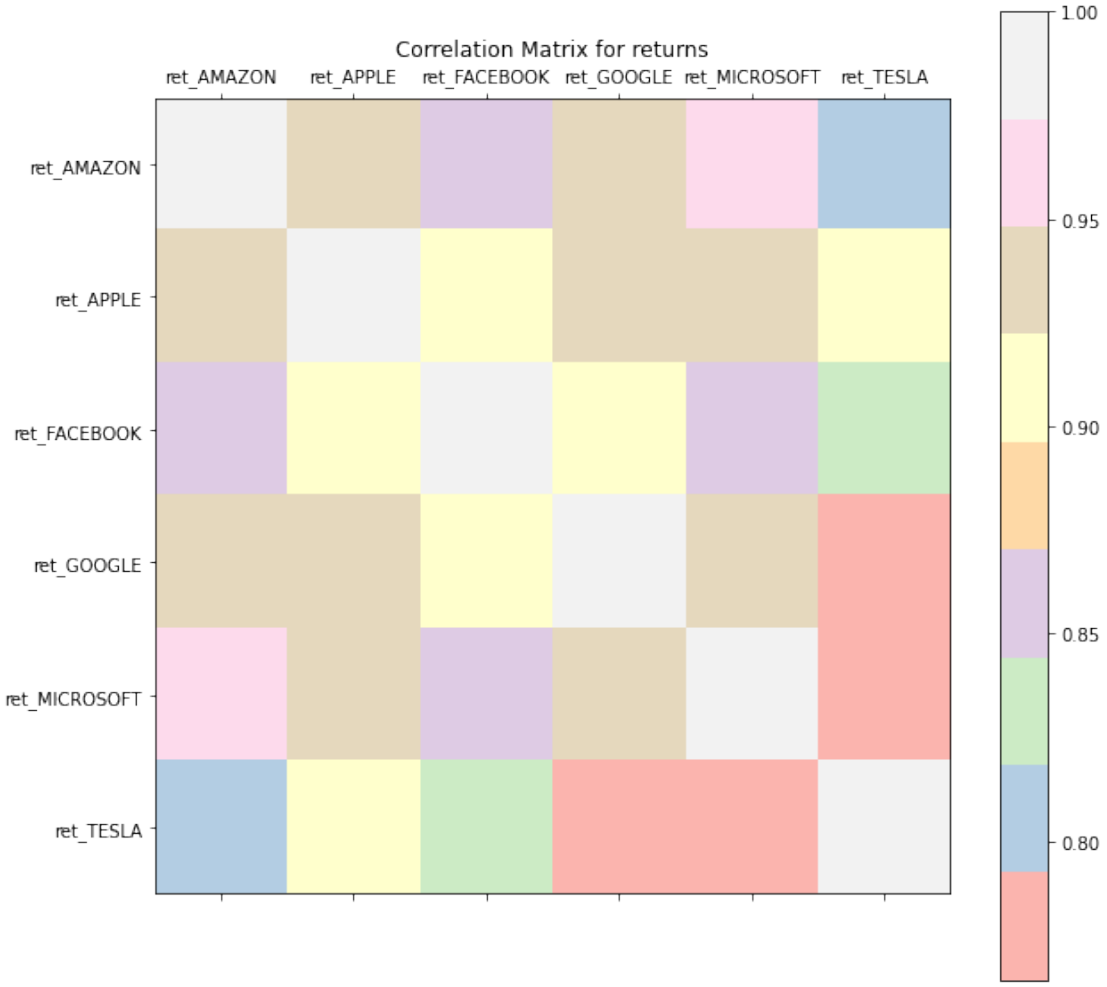

```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```




```
['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close', 'High_2',
'Low_2', 'Open_2', 'Close_2', 'Volume_2', 'Adj Close_2']
```




```
['ret_AMAZON', 'ret_APPLE', 'ret_FACEBOOK', 'ret_GOOGLE',
'ret_MICROSOFT', 'ret_TESLA']
```


Correlation Matrix for returns
ret_AMAZON
ret_APPLE
ret_FACEBOOK
ret_GOOGLE
ret_MICROSOFT
ret_TESLA
1.00
0.95
0.90
0.85
0.80