Autocomplete TAB

In [1]:
```
%config Completer.use_jedi = False
```

Basic Stocks Data Analysis and Visualization

1. Import Dataset & Libraries
2. Perform EDA (Exploratory Data Analysis) and Basic Visualization
3. Perform Interactive Data Visualization
4. Calculate Stocks Daily Return
5. Calculate Correlation between Stocks Daily Return
6. Plot Histogram for Stocks Daily Return

Import Dataset & Libraies

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from copy import copy
from scipy import stats
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go
```

In [3]:
```
df = pd.read_csv('All Bank1.csv')
```

In [4]:
```
stocks_df = df[809:].copy()
```

In [5]:
```
stocks_df = stocks_df.sort_values(by=['Date'])
```

In [6]:
```
stocks_df
```

Out[6]:

	Date	BCA	BNI	BRI	MANDIRI
809	2013-04-17	11000.0	5250.0	1700.0	5300.0
810	2013-04-18	11000.0	5250.0	1730.0	5375.0
811	2013-04-19	11000.0	5300.0	1750.0	5300.0
812	2013-04-22	10900.0	5350.0	1740.0	5350.0
813	2013-04-23	10750.0	5300.0	1730.0	5275.0
...	...	...	...	...	...
2724	2020-12-17	34675.0	6675.0	4330.0	6875.0
2725	2020-12-18	34000.0	6650.0	4280.0	6700.0
2726	2020-12-21	34150.0	6600.0	4210.0	6700.0
2727	2020-12-22	33575.0	6300.0	4130.0	6400.0

	Date	BCA	BNI	BRI	MANDIRI
2728	2020-12-23	33625.0	6250.0	4160.0	6350.0

1920 rows × 5 columns

In [7]:
```
my_stocks_df= stocks_df.set_index('Date', drop=True)
```

In [8]:
```
my_stocks_df.to_csv('My Stocks Data.csv')
```

PERFORM EXPLORATORY DATA ANALYSIS AND BASIC VISUALIZATION

In [9]:
```
# Check if data contains any null values
stocks_df.isnull().sum()
```

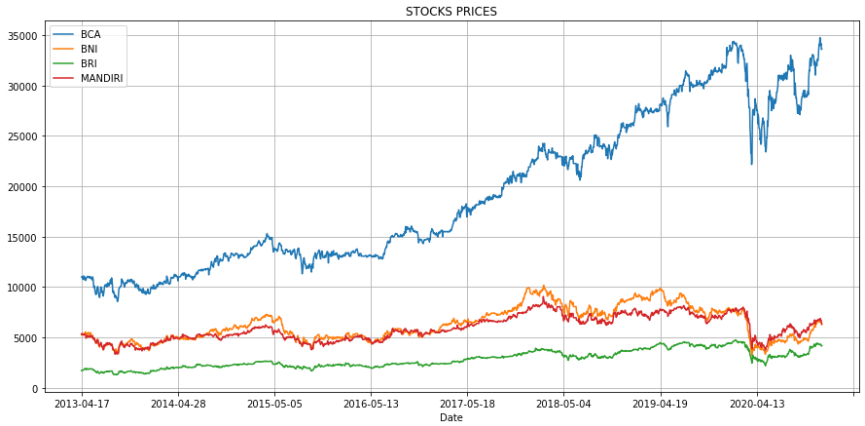
Out[9]:
```
Date       0
BCA        0
BNI        0
BRI        0
MANDIRI    0
dtype: int64
```

In [10]:
```
# Getting dataframe info
stocks_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1920 entries, 809 to 2728
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Date     1920 non-null   object 
 1   BCA      1920 non-null   float64
 2   BNI      1920 non-null   float64
 3   BRI      1920 non-null   float64
 4   MANDIRI  1920 non-null   float64
dtypes: float64(4), object(1)
memory usage: 90.0+ KB
```

In [11]:
```
# Define a function to plot the entire dataframe
def show_plot(df, fig_title):
    df.plot(x='Date', figsize=(15, 7), title=fig_title)
    plt.grid()
    plt.show()
```

In [12]:
```
# Plot the data
show_plot(stocks_df, 'STOCKS PRICES')
```



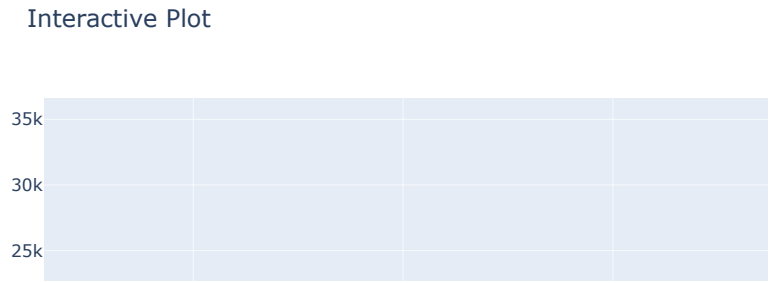
PERFORM INTERACTIVE DATA VISUALIZATION

In [13]:
```
# Function to perform an interactive data plotting using plotly express
def interactive_plot (df, fig_title):
    fig = px.line(title=fig_title)

    # Loop through each stock (while ignoring time columns with index 0)
    for i in df.columns[1:]:
        fig.add_scatter(x=df['Date'], y=df[i], name=i)

    fig.show()
```

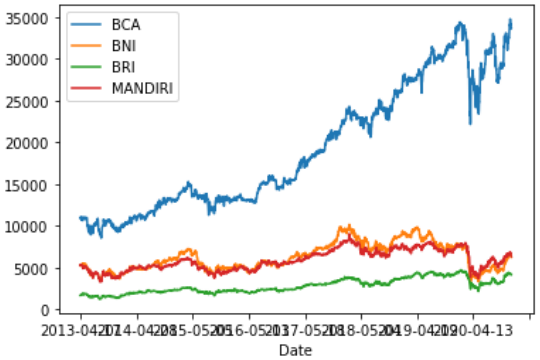
In [14]:
```
fig = px.line(title="Interactive Plot")
for z in stocks_df.columns[1:]:
    fig.add_scatter(x=stocks_df['Date'], y=stocks_df[z], name=z)
fig.show()
```



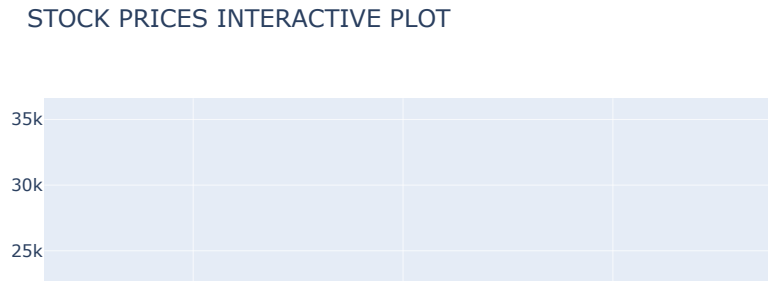
In [15]:
```
chart_data = pd.DataFrame(stocks_df[0:], columns=['Date','BCA','BNI','BRI', 'MANDIRI
```

In [16]:
```
chart_data.plot(x='Date', y=['BCA','BNI','BRI', 'MANDIRI'])
```

Out[16]: <AxesSubplot:xlabel='Date'>



In [17]:
```
# Plot interactive chart
interactive_plot(stocks_df, 'STOCK PRICES INTERACTIVE PLOT')
```



CALCULATE INDIVIDUAL STOCKS DAILY RETURNS

In [18]:
```
new_df = stocks_df.copy()
```

In [19]:
```
new_df = new_df.reset_index(drop=True)
```

In [ ]:
```

```

In [20]:
```
# Let's define a function to calculate stocks daily returns (for all stocks)
def daily_return(df):
    df_new = df.copy()

    # Loop through each stock (while ignoring time columns with index 0)
    for x in df.columns[1:]:

        # Loop through each row belonging to the stock
        for y in range(1, len(df)):

            # Calculate the percentage of change from the previous day
            df_new[x][y] = ((df[x][y] - df[x][y-1])/df[x][y-1])*100

        # set the value of first row to zero since the previous value is not availab
        df_new[x][0] = 0
    return df_new
```

In [21]:
```
stocks_daily_return = daily_return(new_df)
```

In [22]:
```
my_stocks_daily_return = stocks_daily_return.set_index('Date')
```

In [23]:
```
my_stocks_daily_return.to_csv('My Stocks Daily Return.csv')
```

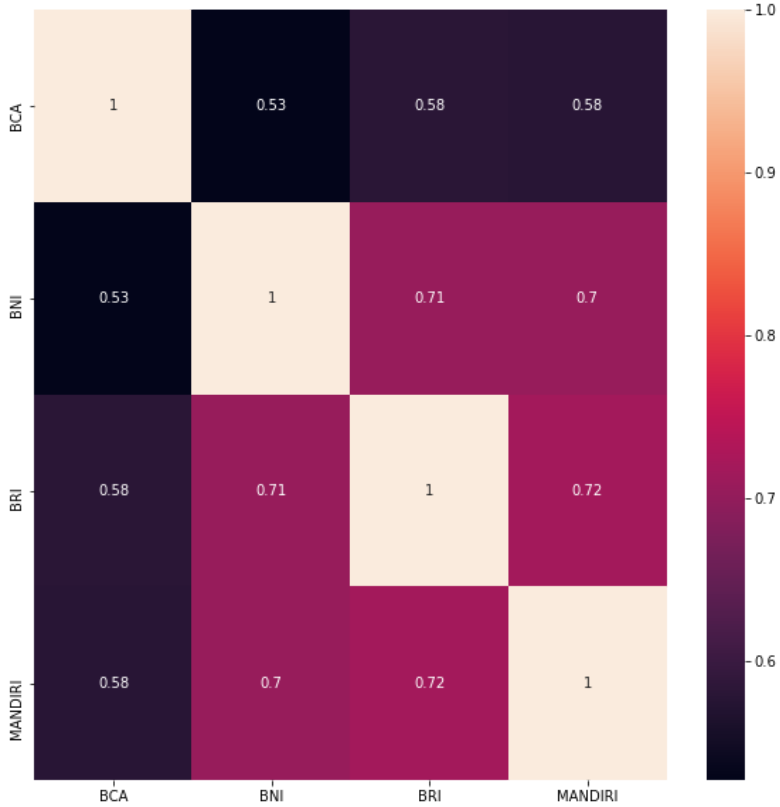
In [ ]:
```

```

CALCULATE THE CORRELATIONS BETWEEN DAILY RETURNS

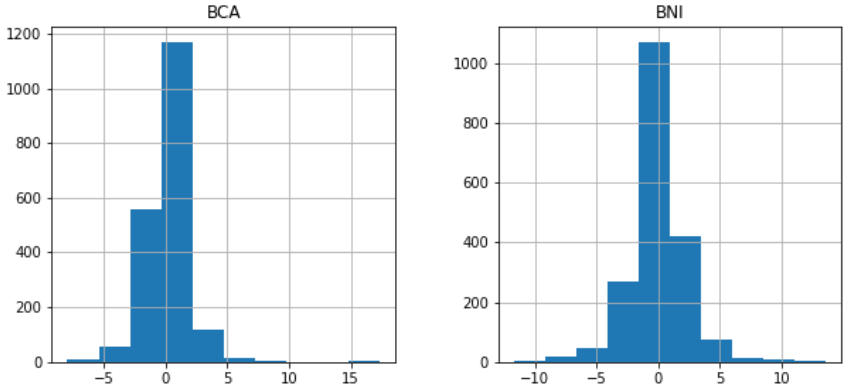
In [24]:
```
cm = stocks_daily_return.drop(columns=['Date']).corr()
```

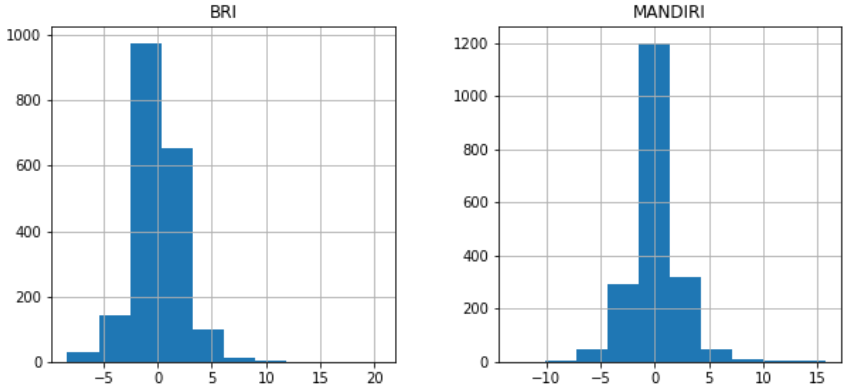
In [25]:
```
plt.figure(figsize=(10, 10))
ax = plt.subplot()
sns.heatmap(cm, annot=True, ax=ax);
```

PLOT THE HISTOGRAM FOR DAILY RETURNS

In [26]:
```
# Histogram of daily returns
# Stock returns are normally distributed with zero mean
# Notice how Tesla Standard deviation is high indicating a more volatile stock
stocks_daily_return.hist(figsize=(10,10), bins=10);
```





In [27]:
```
df_hist = stocks_daily_return.copy()

df_hist = df_hist.drop(columns=['Date'])

data = []

for i in df_hist.columns:
    data.append(stocks_daily_return[i].values)
data
```

Out[27]:
```
[array([ 0.        ,  0.        ,  0.        , ...,  0.44117647,
        -1.68374817,  0.14892033]),
 array([ 0.        ,  0.        ,  0.95238095, ..., -0.7518797 ,
        -4.54545455, -0.79365079]),
 array([ 0.        ,  1.76470588,  1.15606936, ..., -1.63551402,
        -1.90023753,  0.72639225]),
 array([ 0.        ,  1.41509434, -1.39534884, ...,  0.        ,
        -4.47761194, -0.78125   ])]
```

In [28]:
```
fig = ff.create_distplot(data, df_hist.columns)
fig.show()
```