In [1]:
```
# Initial Library imports
import os
import requests
import pandas as pd
from dotenv import load_dotenv
import alpaca_trade_api as tradeapi
import numpy as np
import datetime as dt
```

In [2]:
```
# Load .env environment variables
load_dotenv()
```

Out[2]: True

In [3]:
```
# Set adjustable variables
no_tickers = 30
start_date = "2022-05-10" # format yyyy-mm-dd
end_date   = "2022-05-17" # format yyyy-mm-dd *** note with 4Hour timeframe, data wi

#Set timeframe of Tickers
timeframe = "4Hour"
```

In [4]:
```
# Set Alpaca API key and secret
alpaca_api_key = os.getenv("ALPACA_API_KEY")
alpaca_secret_key = os.getenv("ALPACA_SECRET_KEY")
```

In [5]:
```
# Create the Alpaca API object
alpaca = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version="v2")
```

In [6]:
```
# import csv list of all nasdaq tickers and clean them for use

#import csv file of all tickers to a dataframe
df_full_ticker_list = pd.read_csv ('Data/nasdaq_screener_assets.csv')
#drop unwanted columns
df_full_ticker_list = df_full_ticker_list.drop(['Last Sale', 'Net Change', '% Change
       'Country', 'IPO Year', 'Volume', 'Sector', 'Industry'], axis = 'columns')

# Remove Tickers with '^', '/', & spaces as Alpaca does not accept them
df_full_ticker_list = df_full_ticker_list[~df_full_ticker_list.Symbol.str.contains('
                                    & ~df_full_ticker_list.Symbol.str.contains('/',
                                    & ~df_full_ticker_list.Symbol.str.contains(' ',
                                   ]

# convert df to list for use with Alpaca API
alpaca_tickers = list(df_full_ticker_list['Symbol'])
```

In [7]:
```
#set up for Alpaca API call

# Format current date as ISO format
start_date = pd.Timestamp(start_date, tz="America/New_York").isoformat()
end_date = pd.Timestamp(end_date, tz="America/New_York").isoformat()
```

In [8]:

```
# make API call to Alpaca to receive a data frame of all stock data

df_stock_all = alpaca.get_bars(
    alpaca_tickers,
    timeframe,
    start = start_date,
    end = end_date
).df

#check output of df_stock_all
df_stock_all.head(10)
```

Out[8]:

timestamp	open	high	low	close	volume	trade_count	vwap	symbol
2022-05-10 08:00:00+00:00	55.600	55.600	54.60	55.030	34898	288	55.157951	AA
2022-05-10 12:00:00+00:00	54.900	56.880	54.07	54.520	4036281	38858	55.486529	AA
2022-05-10 16:00:00+00:00	54.580	56.910	53.61	55.835	4177742	50011	55.568406	AA
2022-05-10 20:00:00+00:00	55.840	56.100	55.61	55.700	203721	27	55.839540	AA
2022-05-11 08:00:00+00:00	56.890	56.930	56.50	56.930	955	17	56.761602	AA
2022-05-11 12:00:00+00:00	57.000	59.360	55.19	58.370	2748562	30562	58.218161	AA
2022-05-11 16:00:00+00:00	58.405	58.405	56.27	56.620	3124526	32199	57.113865	AA
2022-05-11 20:00:00+00:00	56.640	56.790	56.00	56.660	229478	105	56.648243	AA
2022-05-12 08:00:00+00:00	56.000	56.000	54.31	55.150	4834	50	54.991585	AA
2022-05-12 12:00:00+00:00	54.610	56.360	53.34	55.030	3098872	30907	54.867158	AA

In [9]:

```
# calculate the change in price accross the selected date range to determine a suita

# Set up df_stock_price_change varialbe as data frame to capture all start and end v
df_stock_price_change = pd.DataFrame(columns = ['Ticker', 'Start Price', 'End Price'
# capture all Ticker symbols in df_stock_price_change form df_full_ticker_list
df_stock_price_change['Ticker'] = df_full_ticker_list['Symbol']

# loop through each ticker and capture the first and last pice for each
for index in df_stock_price_change.index:
    ticker = df_stock_price_change['Ticker'][index]
    ticker_data_temp = df_stock_all[df_stock_all.symbol == ticker]

    # ignore ticker if no symbol was found from Alpaca, only process if the size of
    if ticker_data_temp.size != 0:

        df_stock_price_change['Start Price'][index] = ticker_data_temp.at[ticker_dat
        df_stock_price_change['End Price'][index] = ticker_data_temp.at[ticker_data_
```

```
# calculate price change
df_stock_price_change['Price Change'] = df_stock_price_change['End Price'] - df_stoc
df_stock_price_change['Price Change %'] = (df_stock_price_change['Price Change']/df_

# check output of df_stock_price_change
df_stock_price_change.head(10)
```

Out[9]:

	Ticker	Start Price	End Price	Price Change	Price Change %
0	A	114.46	120.3	5.84	5.102219
1	AA	55.03	61.11	6.08	11.048519
2	AAC	9.815	9.79	-0.025	-0.254712
3	AACG	1.09	1.01	-0.08	-7.33945
4	AACI	9.84	9.84	0.0	0.0
5	AACIW	0.2698	0.2678	-0.002	-0.74129
6	AADI	12.67	13.72	1.05	8.287293
7	AAIC	3.0482	3.62	0.5718	18.758612
10	AAIN	23.245	24.19	0.945	4.06539
11	AAL	16.53	16.87	0.34	2.056866

In [10]:

```
# Clean df_stock_price_change, drop NaNs and sort in order of % change
df_stock_price_change.sort_values(by = 'Price Change %',ascending = False, inplace =
df_stock_price_change.dropna(axis = 'index', how = 'any' ,inplace = True)
df_stock_price_change.reset_index(drop = True, inplace = True)

# check output of df_stock_price_change
df_stock_price_change.head(10)
```

Out[10]:

	Ticker	Start Price	End Price	Price Change	Price Change %
0	RMTI	0.2793	1.9	1.6207	580.272109
1	PXS	0.6261	2.63	2.0039	320.060693
2	PT	0.44	1.71	1.27	288.636364
3	PIXY	0.198	0.46	0.262	132.323232
4	EYESW	0.2125	0.45	0.2375	111.764706
5	AGRI	1.3	2.69	1.39	106.923077
6	HHGCW	0.1353	0.2799	0.1446	106.873614
7	NLSPW	0.109	0.22	0.111	101.834862
8	NDRAW	0.0153	0.03	0.0147	96.078431
9	ENTXW	0.0628	0.12	0.0572	91.082803

In [11]:

```
# Select Tickers for analysis distributed through Data Set equally
# this is to ensure that the tickers selected for analysis are a well represented se

#set number of rows
rows_count = df_stock_price_change.shape[0]
```

```
# calculate step size for selection less one to avoid the ends
increment = round(rows_count/(no_tickers))-1
# set indexes of tickers to capture for analysis
list_select_tickers = range(round(no_tickers/2), rows_count, increment)

# capture the ticker symbols to use for anlysis
list_analysis_tickers = list(df_stock_price_change['Ticker'][list_select_tickers].va

# check output of list_analysis_tickers
list_analysis_tickers
```

Out[11]:
```
['PXSAW',
 'SCSC',
 'MNTK',
 'DINO',
 'ADMA',
 'BEPI',
 'ESMT',
 'FTAI',
 'TDF',
 'WD',
 'OOMA',
 'CRL',
 'WEA',
 'FELE',
 'MELI',
 'YTPG',
 'VSSYW',
 'TOAC',
 'SNOW',
 'MCACU',
 'LOGI',
 'AXAC',
 'SMAPW',
 'EVTL',
 'ORCL',
 'PFTAW',
 'GCMGW',
 'BLZE',
 'GOEVW',
 'CYBN',
 'SCOBW']
```

In [12]:
```
# capture stock price data of the selected tickers and clean ready for analysis

# capture the data for only the chosen tickers from df_stock_all
df_stock_price_data = df_stock_all[df_stock_all['symbol'].isin(list_analysis_tickers
# reset index
df_stock_price_data.reset_index(inplace = True)

# Change 'timestamp' values to date only
##df_stock_data.loc[:,'timestamp'] = df_stock_data.loc[:,'timestamp'].dt.date
# df_stock_price_data.timestamp = pd.to_datetime(df_stock_price_data.timestamp)
# df_stock_price_data['timestamp'] = df_stock_price_data['timestamp'].to_pydatetime(

# drop un-needed columns
df_stock_price_data.drop(['open','high','low','volume','trade_count','vwap'], axis='
# rename remaining columns to suitable names
df_stock_price_data.columns = ['Date','Close','Ticker']

# check output of df_stock_price_data
df_stock_price_data.head(10)
```

```
C:\Users\mclew\anaconda3\envs\pyvizenv\lib\site-packages\pandas\core\frame.py:4913:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

Out[12]:

	Date	Close	Ticker
0	2022-05-10 08:00:00+00:00	1.700	ADMA
1	2022-05-10 12:00:00+00:00	1.590	ADMA
2	2022-05-10 16:00:00+00:00	1.605	ADMA
3	2022-05-10 20:00:00+00:00	1.680	ADMA
4	2022-05-11 12:00:00+00:00	1.550	ADMA
5	2022-05-11 16:00:00+00:00	1.430	ADMA
6	2022-05-11 20:00:00+00:00	1.530	ADMA
7	2022-05-12 08:00:00+00:00	1.460	ADMA
8	2022-05-12 12:00:00+00:00	1.665	ADMA
9	2022-05-12 16:00:00+00:00	1.675	ADMA

In [13]:

```
# save CSV of df_stock_price_data as 'Data/StockPriceData.csv'
df_stock_price_data.to_csv('Data/StockPriceData.csv')
```

In [15]:

```
#save df_stock_price_change in .csv for use in other program
#convert list to DF
df_stock_price_change.to_csv('Data/StockPriceChange.csv')
```