# Creation of an Equity Dashboard using Python Dash Library

Environment: JupyterLab in Anaconda, Created by: Jen Bianca Tan

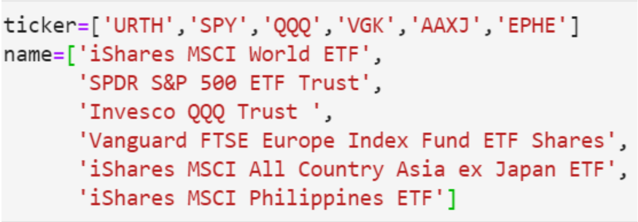
## Desired Output

The objective is to have a summary of key equity indices without the need for a Bloomberg Terminal.

## Data

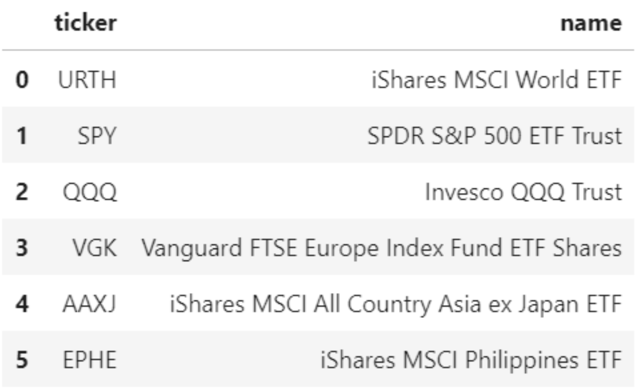
### Tickers, ***One-Time Input***

We indicate tickers we want from the yahoo finance website.



```
ticker=['URTH','SPY','QQQ','VGK','AAXJ','EPHE']
name=['iShares MSCI World ETF',
      'SPDR S&P 500 ETF Trust',
      'Invesco QQQ Trust ',
      'Vanguard FTSE Europe Index Fund ETF Shares',
      'iShares MSCI All Country Asia ex Japan ETF',
      'iShares MSCI Philippines ETF']
```

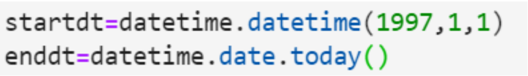
Below shows the tickers and index names arranged in a data frame.



| | ticker | name |
|---|---|---|
| **0** | URTH | iShares MSCI World ETF |
| **1** | SPY | SPDR S&P 500 ETF Trust |
| **2** | QQQ | Invesco QQQ Trust |
| **3** | VGK | Vanguard FTSE Europe Index Fund ETF Shares |
| **4** | AAXJ | iShares MSCI All Country Asia ex Japan ETF |
| **5** | EPHE | iShares MSCI Philippines ETF |

### Dates, ***One-Time Input***

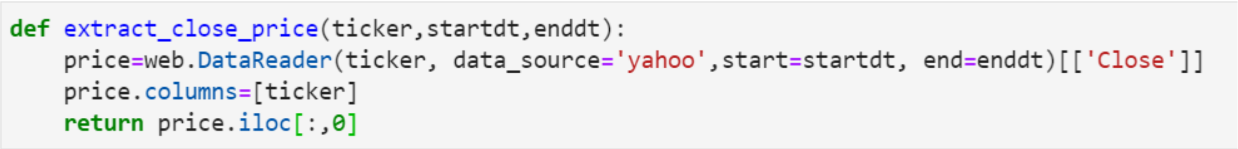
Input the dates depending on how long a historical data set we want to see.



```
startdt=datetime.datetime(1997,1,1)
enddt=datetime.date.today()
```

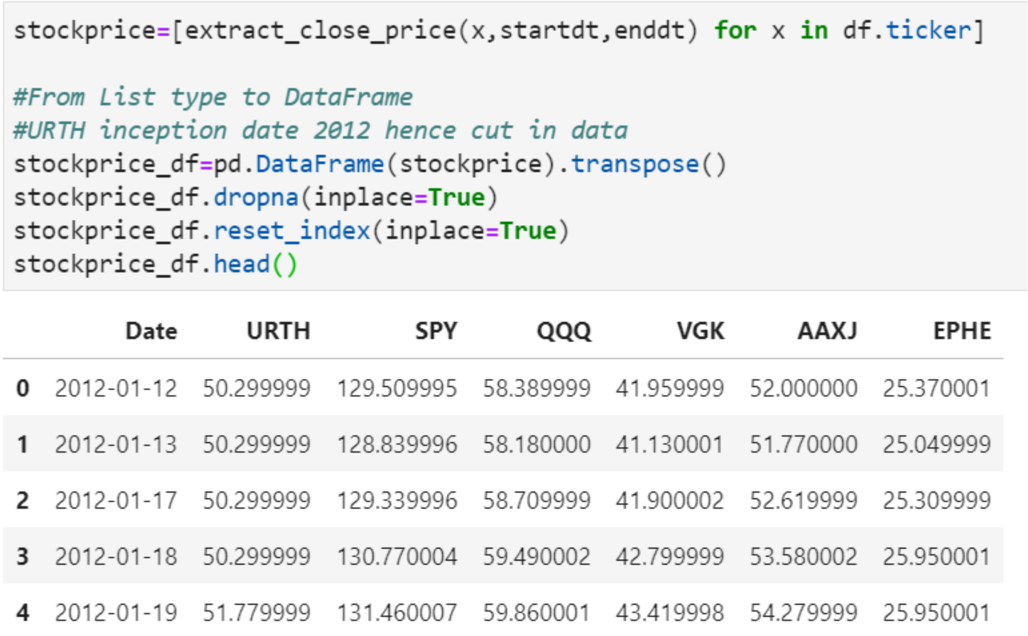
### Price Data

To extract the historical prices of the indices, I create function for ease that extracts only the closing price for the specified dates.



```
def extract_close_price(ticker,startdt,enddt):
    price=web.DataReader(ticker, data_source='yahoo',start=startdt, end=enddt)[['Close']]
    price.columns=[ticker]
    return price.iloc[:,0]
```

We use this function in a loop of the ticker list and arrange it in a data frame for ease in viewing and analysis:

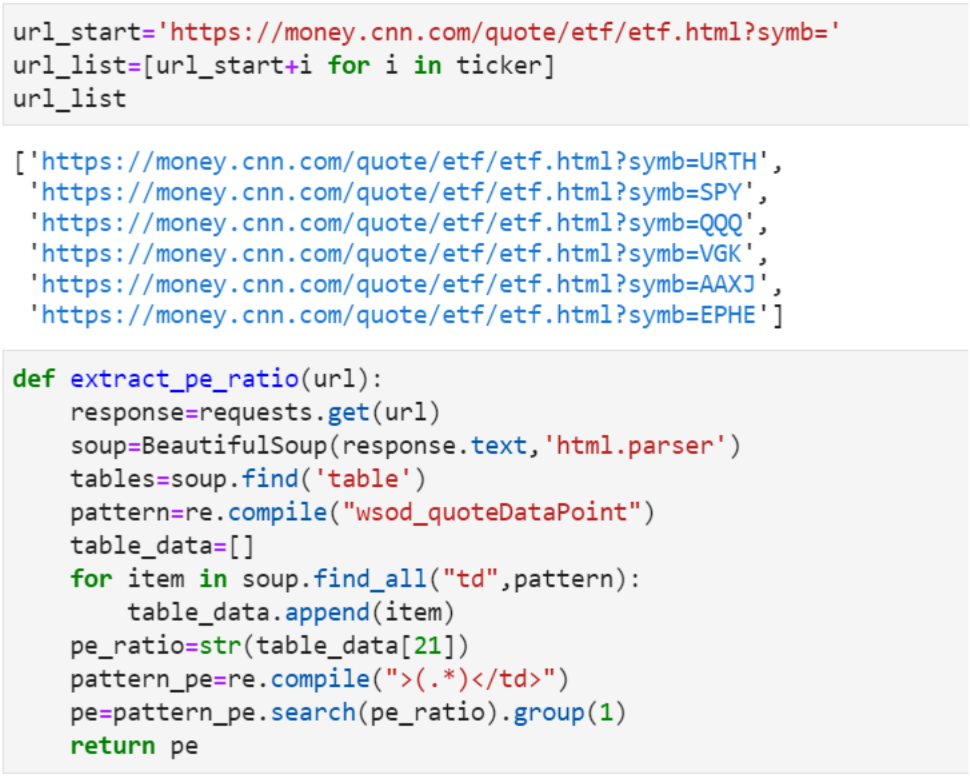


```
stockprice=[extract_close_price(x,startdt,enddt) for x in df.ticker]

#From List type to DataFrame
#URTH inception date 2012 hence cut in data
stockprice_df=pd.DataFrame(stockprice).transpose()
stockprice_df.dropna(inplace=True)
stockprice_df.reset_index(inplace=True)
stockprice_df.head()
```

|  | Date | URTH | SPY | QQQ | VGK | AAXJ | EPHE |
|---|---|---|---|---|---|---|---|
| **0** | 2012-01-12 | 50.299999 | 129.509995 | 58.389999 | 41.959999 | 52.000000 | 25.370001 |
| **1** | 2012-01-13 | 50.299999 | 128.839996 | 58.180000 | 41.130001 | 51.770000 | 25.049999 |
| **2** | 2012-01-17 | 50.299999 | 129.339996 | 58.709999 | 41.900002 | 52.619999 | 25.309999 |
| **3** | 2012-01-18 | 50.299999 | 130.770004 | 59.490002 | 42.799999 | 53.580002 | 25.950001 |
| **4** | 2012-01-19 | 51.779999 | 131.460007 | 59.860001 | 43.419998 | 54.279999 | 25.950001 |

### Price-to-Earnings Ratio

We extract the P/E ratios from www.money.cnn.com. First, we create the url links for the tickers, then create a function that extract this based on its position in the web page using BeautifulSoup.



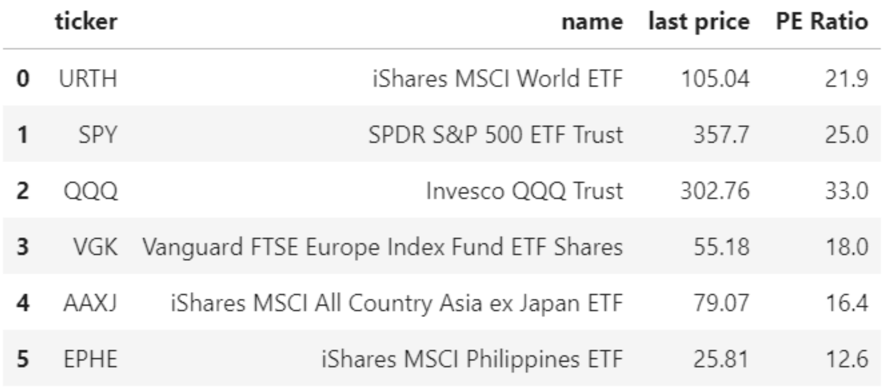
```
url_start='https://money.cnn.com/quote/etf/etf.html?symb='
url_list=[url_start+i for i in ticker]
url_list
```

```
['https://money.cnn.com/quote/etf/etf.html?symb=URTH',
 'https://money.cnn.com/quote/etf/etf.html?symb=SPY',
 'https://money.cnn.com/quote/etf/etf.html?symb=QQQ',
 'https://money.cnn.com/quote/etf/etf.html?symb=VGK',
 'https://money.cnn.com/quote/etf/etf.html?symb=AAXJ',
 'https://money.cnn.com/quote/etf/etf.html?symb=EPHE']
```

```
def extract_pe_ratio(url):
    response=requests.get(url)
    soup=BeautifulSoup(response.text,'html.parser')
    tables=soup.find('table')
    pattern=re.compile("wsod_quoteDataPoint")
    table_data=[]
    for item in soup.find_all("td",pattern):
        table_data.append(item)
    pe_ratio=str(table_data[21])
    pattern_pe=re.compile(">(.*)</td>")
    pe=pattern_pe.search(pe_ratio).group(1)
    return pe
```

Then we loop this for all the tickers.

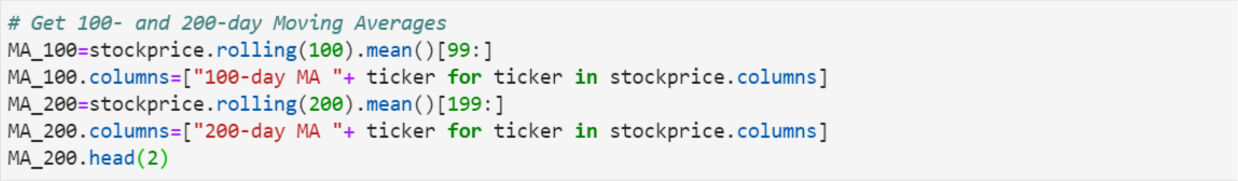
Finally, we have the first section of our dashboard which contains the latest price and the latest P/E.



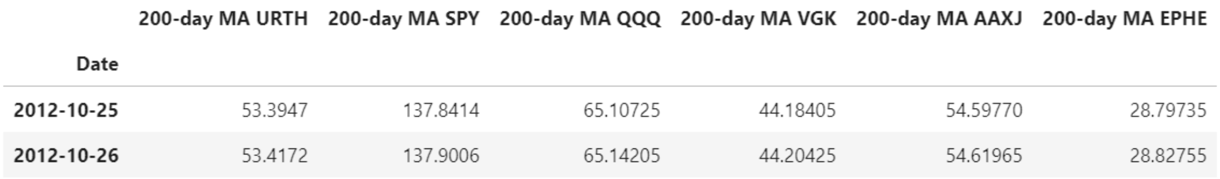
|  | ticker | name | last price | PE Ratio |
|---|---|---|---|---|
| 0 | URTH | iShares MSCI World ETF | 105.04 | 21.9 |
| 1 | SPY | SPDR S&P 500 ETF Trust | 357.7 | 25.0 |
| 2 | QQQ | Invesco QQQ Trust | 302.76 | 33.0 |
| 3 | VGK | Vanguard FTSE Europe Index Fund ETF Shares | 55.18 | 18.0 |
| 4 | AAXJ | iShares MSCI All Country Asia ex Japan ETF | 79.07 | 16.4 |
| 5 | EPHE | iShares MSCI Philippines ETF | 25.81 | 12.6 |

## Technical Indicators

We compute historical 100-day and 200-day moving averages.

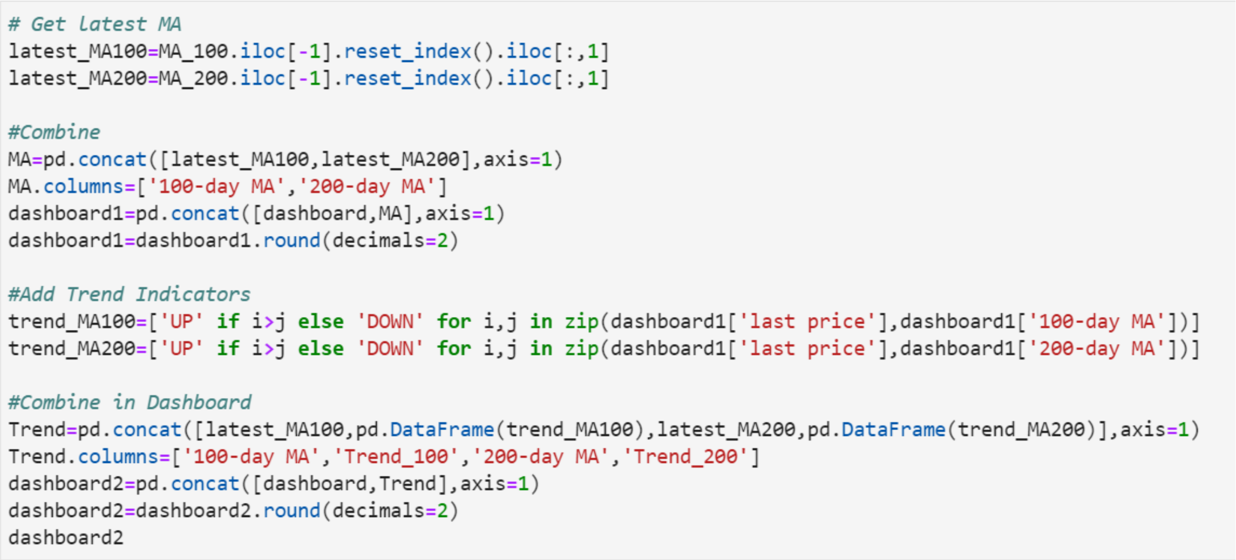


```
# Get 100- and 200-day Moving Averages
MA_100=stockprice.rolling(100).mean()[99:]
MA_100.columns=["100-day MA "+ ticker for ticker in stockprice.columns]
MA_200=stockprice.rolling(200).mean()[199:]
MA_200.columns=["200-day MA "+ ticker for ticker in stockprice.columns]
MA_200.head(2)
```



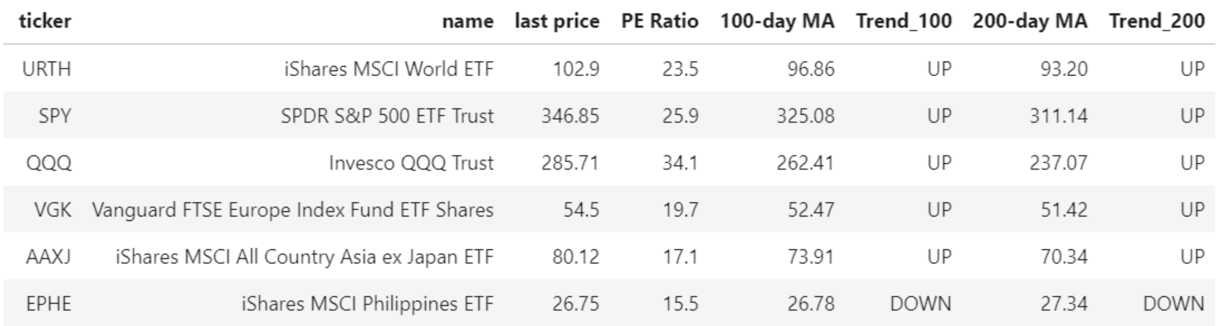
|  | 200-day MA URTH | 200-day MA SPY | 200-day MA QQQ | 200-day MA VGK | 200-day MA AAXJ | 200-day MA EPHE |
|---|---|---|---|---|---|---|
| **Date** |  |  |  |  |  |  |
| **2012-10-25** | 53.3947 | 137.8414 | 65.10725 | 44.18405 | 54.59770 | 28.79735 |
| **2012-10-26** | 53.4172 | 137.9006 | 65.14205 | 44.20425 | 54.61965 | 28.82755 |

However, we just get the latest one. We also indicate the signal as “UP” if the last price is higher than the moving average or “DOWN” if its lower than the moving average.



```
# Get Latest MA
latest_MA100=MA_100.iloc[-1].reset_index().iloc[:,1]
latest_MA200=MA_200.iloc[-1].reset_index().iloc[:,1]

#Combine
MA=pd.concat([latest_MA100,latest_MA200],axis=1)
MA.columns=['100-day MA','200-day MA']
dashboard1=pd.concat([dashboard,MA],axis=1)
dashboard1=dashboard1.round(decimals=2)

#Add Trend Indicators
trend_MA100=['UP' if i>j else 'DOWN' for i,j in zip(dashboard1['last price'],dashboard1['100-day MA'])]
trend_MA200=['UP' if i>j else 'DOWN' for i,j in zip(dashboard1['last price'],dashboard1['200-day MA'])]

#Combine in Dashboard
Trend=pd.concat([latest_MA100,pd.DataFrame(trend_MA100),latest_MA200,pd.DataFrame(trend_MA200)],axis=1)
Trend.columns=['100-day MA','Trend_100','200-day MA','Trend_200']
dashboard2=pd.concat([dashboard,Trend],axis=1)
dashboard2=dashboard2.round(decimals=2)
dashboard2
```

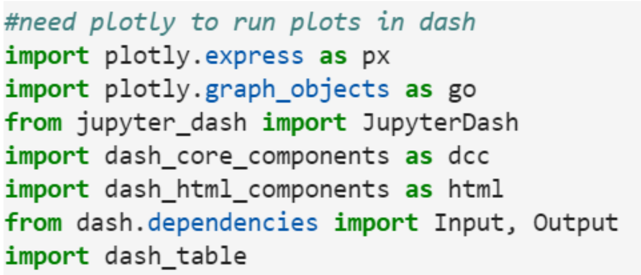
For instance, we might say we expect the QQQ index to trend upward given the current price is greater than both the 100-day and 200-day moving averages.



| ticker | name | last price | PE Ratio | 100-day MA | Trend_100 | 200-day MA | Trend_200 |
|---|---|---|---|---|---|---|---|
| URTH | iShares MSCI World ETF | 102.9 | 23.5 | 96.86 | UP | 93.20 | UP |
| SPY | SPDR S&P 500 ETF Trust | 346.85 | 25.9 | 325.08 | UP | 311.14 | UP |
| QQQ | Invesco QQQ Trust | 285.71 | 34.1 | 262.41 | UP | 237.07 | UP |
| VGK | Vanguard FTSE Europe Index Fund ETF Shares | 54.5 | 19.7 | 52.47 | UP | 51.42 | UP |
| AAXJ | iShares MSCI All Country Asia ex Japan ETF | 80.12 | 17.1 | 73.91 | UP | 70.34 | UP |
| EPHE | iShares MSCI Philippines ETF | 26.75 | 15.5 | 26.78 | DOWN | 27.34 | DOWN |

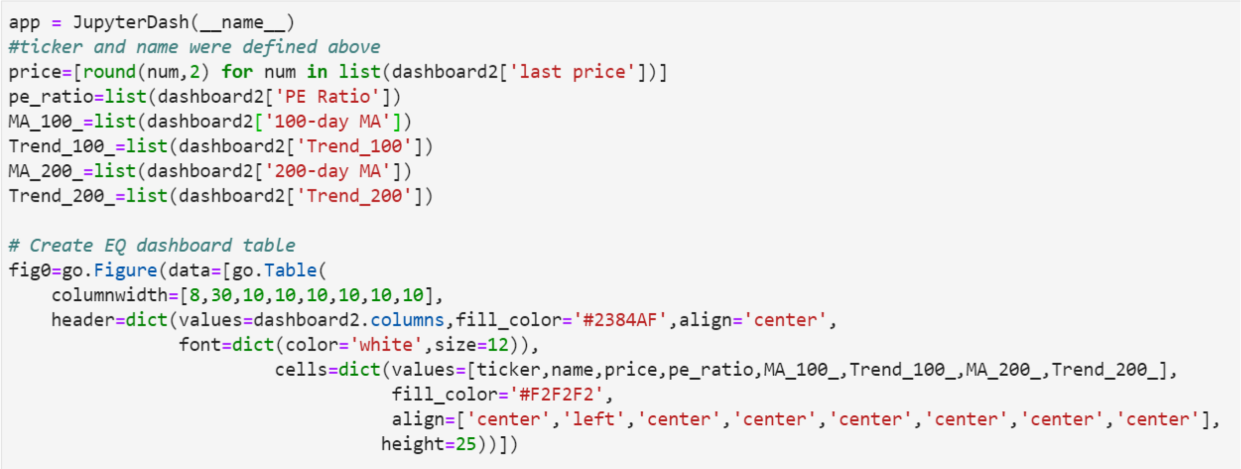
Dashboard using Dash Library

- Load the necessary libraries for plotting



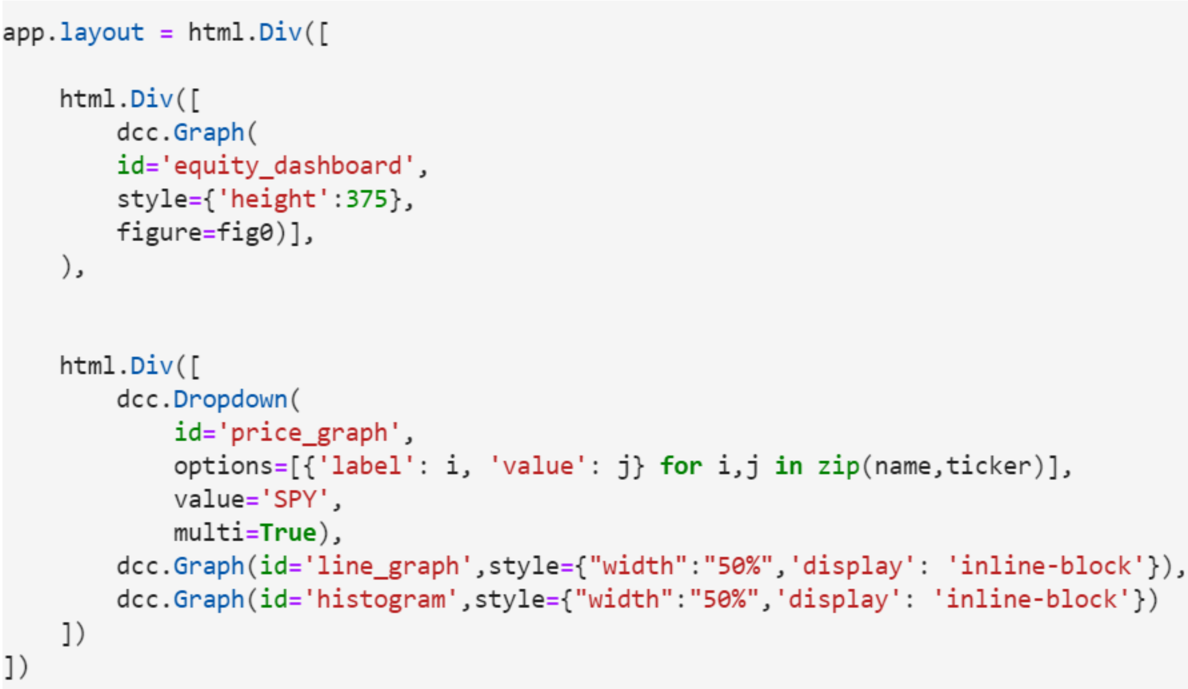
```
#need plotly to run plots in dash
import plotly.express as px
import plotly.graph_objects as go
from jupyter_dash import JupyterDash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import dash_table
```

- Create a list for all the values and use these to create a table with plotly.graph_objects.



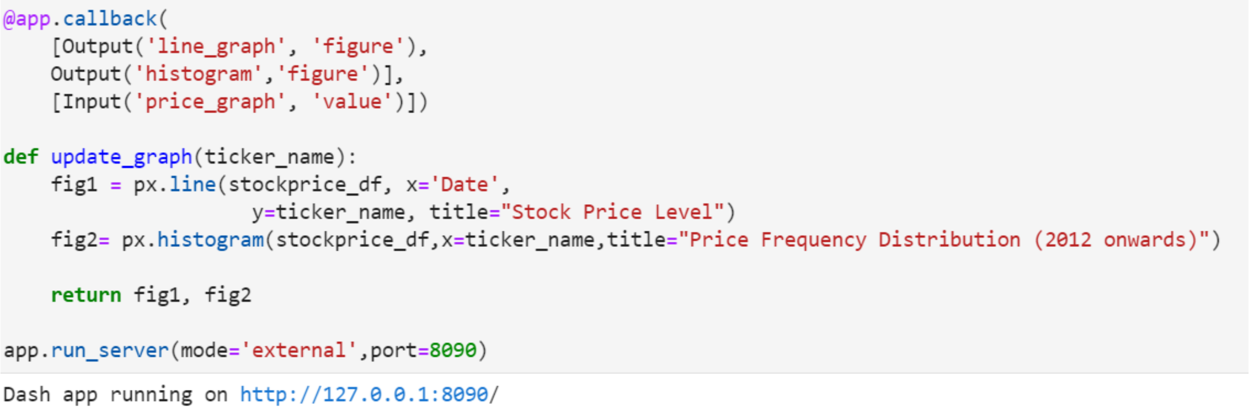
```
app = JupyterDash(__name__)
#ticker and name were defined above
price=[round(num,2) for num in list(dashboard2['last price'])]
pe_ratio=list(dashboard2['PE Ratio'])
MA_100_=list(dashboard2['100-day MA'])
Trend_100_=list(dashboard2['Trend_100'])
MA_200_=list(dashboard2['200-day MA'])
Trend_200_=list(dashboard2['Trend_200'])

# Create EQ dashboard table
fig0=go.Figure(data=[go.Table(
    columnwidth=[8,30,10,10,10,10,10,10],
    header=dict(values=dashboard2.columns,fill_color='#2384AF',align='center',
                font=dict(color='white',size=12)),
                         cells=dict(values=[ticker,name,price,pe_ratio,MA_100_,Trend_100_,MA_200_,Trend_200_],
                                    fill_color='#F2F2F2',
                                    align=['center','left','center','center','center','center','center','center'],
                                   height=25))])
```

- The first image in the first line is the table
- The images in the second line are the line chart and histogram each occupying 50% of the window. Plotly express library is used to create the line charts and histogram
- Tickers are stored as a list for our dcc.dropdown



```
app.layout = html.Div([

    html.Div([
        dcc.Graph(
        id='equity_dashboard',
        style={'height':375},
        figure=fig0)],
    ),


    html.Div([
        dcc.Dropdown(
            id='price_graph',
            options=[{'label': i, 'value': j} for i,j in zip(name,ticker)],
            value='SPY',
            multi=True),
        dcc.Graph(id='line_graph',style={"width":"50%",'display': 'inline-block'}),
        dcc.Graph(id='histogram',style={"width":"50%",'display': 'inline-block'})
    ])
])
```

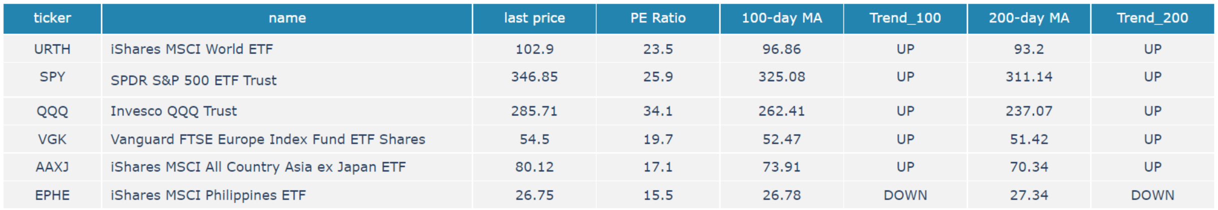
After some formatting and naming:



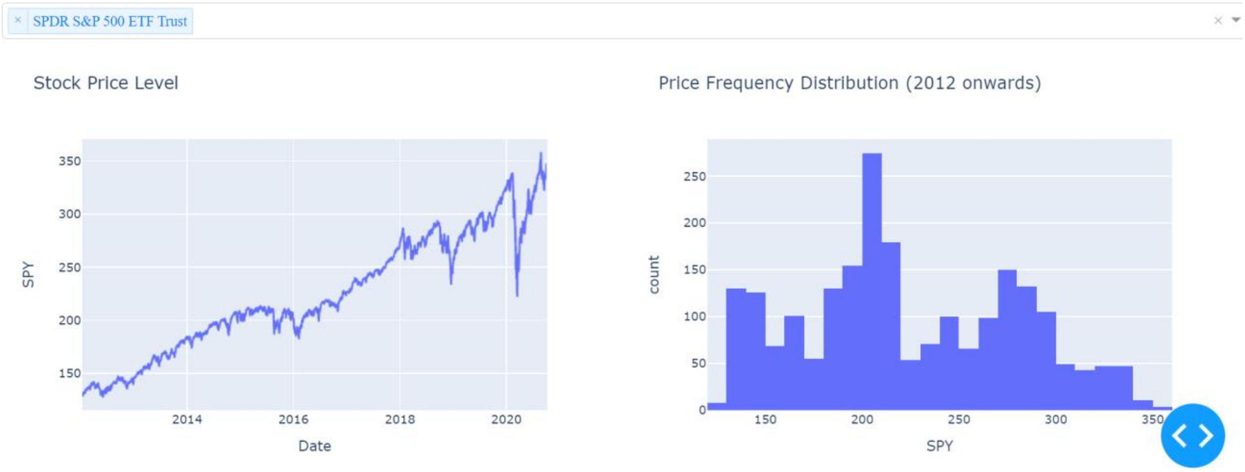
```
@app.callback(
    [Output('line_graph', 'figure'),
    Output('histogram','figure')],
    [Input('price_graph', 'value')])

def update_graph(ticker_name):
    fig1 = px.line(stockprice_df, x='Date',
                  y=ticker_name, title="Stock Price Level")
    fig2= px.histogram(stockprice_df,x=ticker_name,title="Price Frequency Distribution (2012 onwards)")

    return fig1, fig2

app.run_server(mode='external',port=8090)
```

```
Dash app running on http://127.0.0.1:8090/
```

The final dash app appears as below:



| ticker | name | last price | PE Ratio | 100-day MA | Trend_100 | 200-day MA | Trend_200 |
|---|---|---|---|---|---|---|---|
| URTH | iShares MSCI World ETF | 102.9 | 23.5 | 96.86 | UP | 93.2 | UP |
| SPY | SPDR S&P 500 ETF Trust | 346.85 | 25.9 | 325.08 | UP | 311.14 | UP |
| QQQ | Invesco QQQ Trust | 285.71 | 34.1 | 262.41 | UP | 237.07 | UP |
| VGK | Vanguard FTSE Europe Index Fund ETF Shares | 54.5 | 19.7 | 52.47 | UP | 51.42 | UP |
| AAXJ | iShares MSCI All Country Asia ex Japan ETF | 80.12 | 17.1 | 73.91 | UP | 70.34 | UP |
| EPHE | iShares MSCI Philippines ETF | 26.75 | 15.5 | 26.78 | DOWN | 27.34 | DOWN |

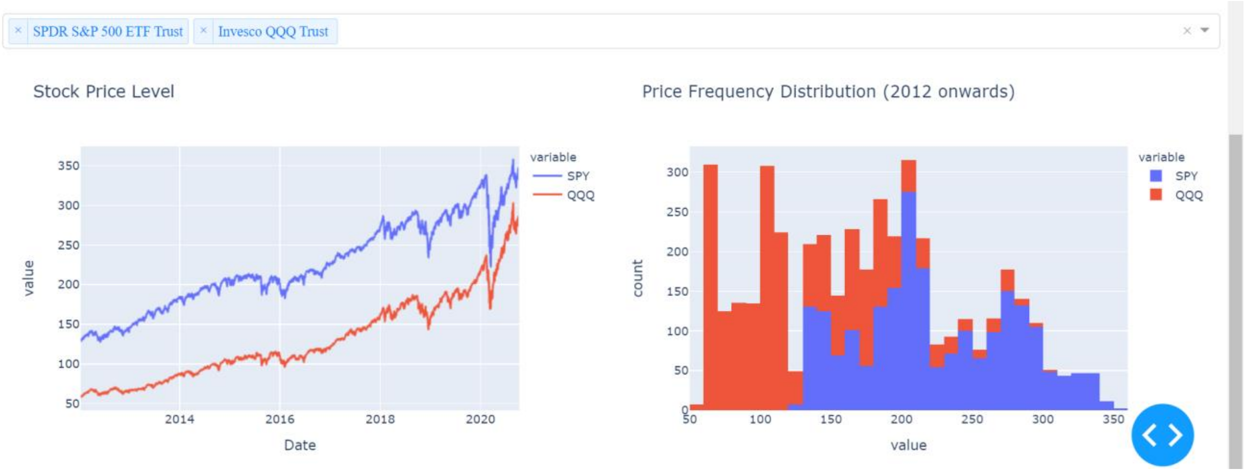


× SPDR S&P 500 ETF Trust

Stock Price Level

Price Frequency Distribution (2012 onwards)

We can also select two (or multiple) to overlay:



× SPDR S&P 500 ETF Trust × Invesco QQQ Trust

Stock Price Level

Price Frequency Distribution (2012 onwards)