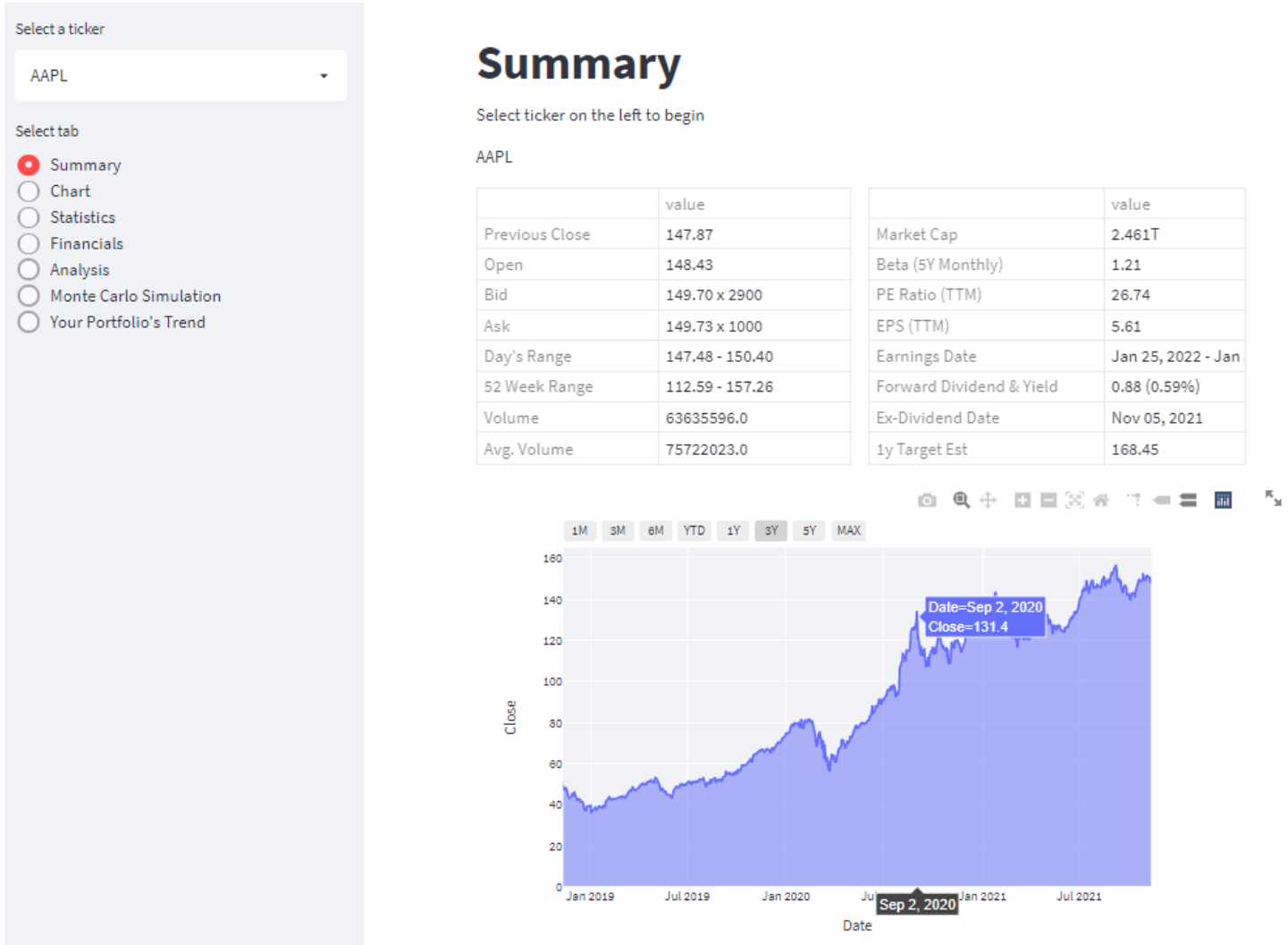
FinDash

Product _ “FinDash

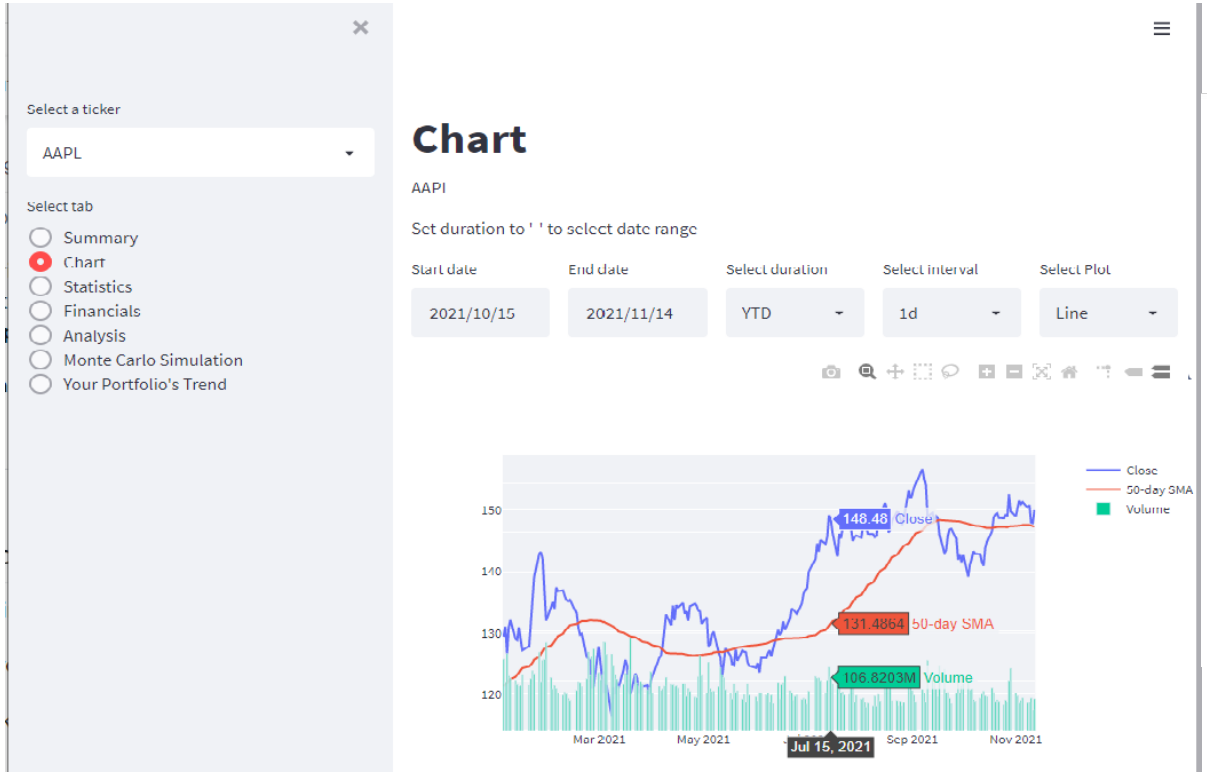
Product Goals & Challenges

- To build a **window** into the world of S&P 500.
- To build a **one stop** solution for stock market enthusiasts. All important data is present in the dashboard.
- To present stock market data in an **interactive** dashboard.
- To help investors view the **trends** in their portfolio.
- To help investors see **predictions** and VAR of their stock.



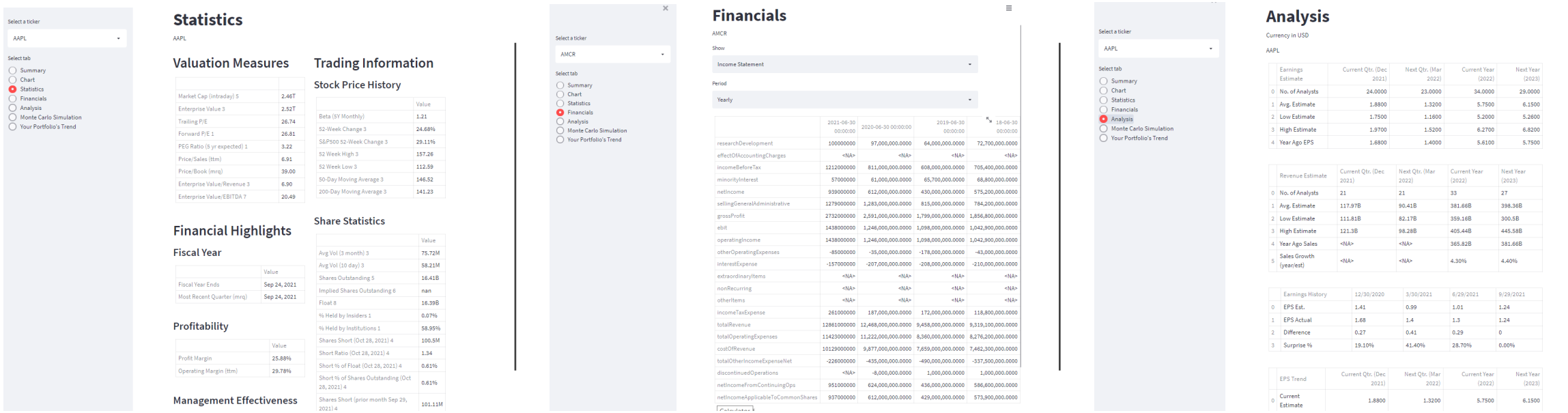
Tab 1

- View important statistics
- View trend of close price for selected period



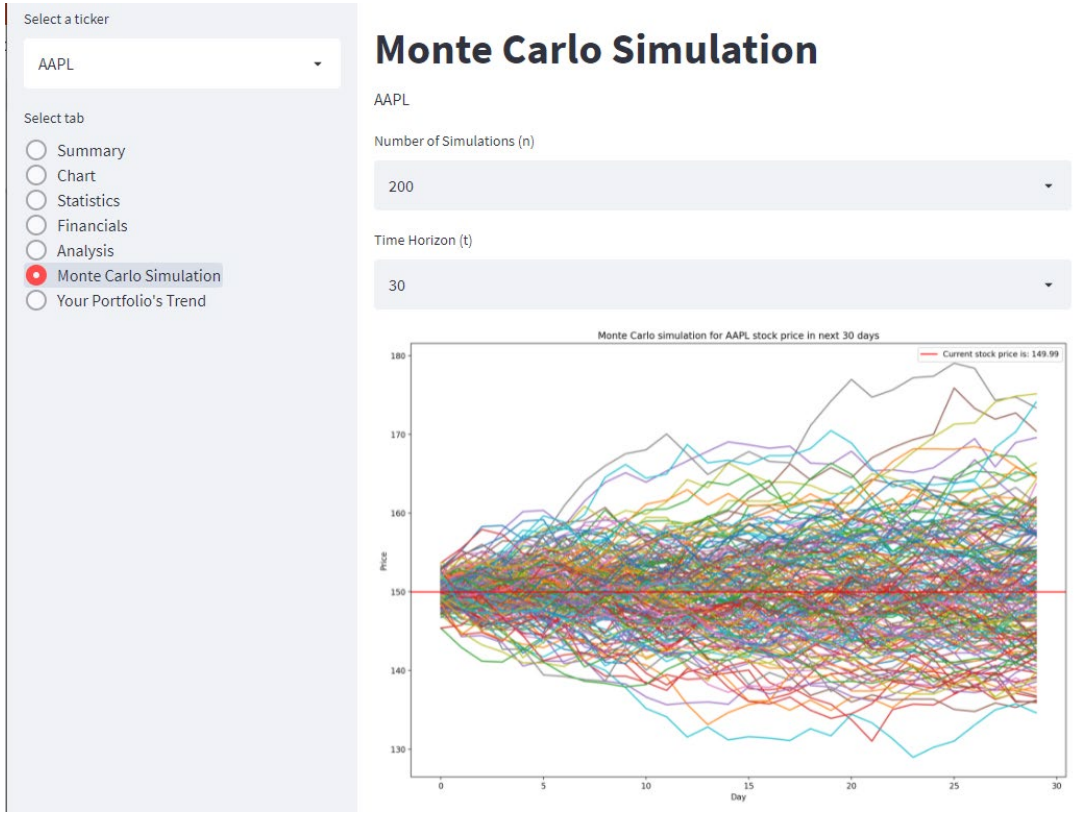
Tab 2

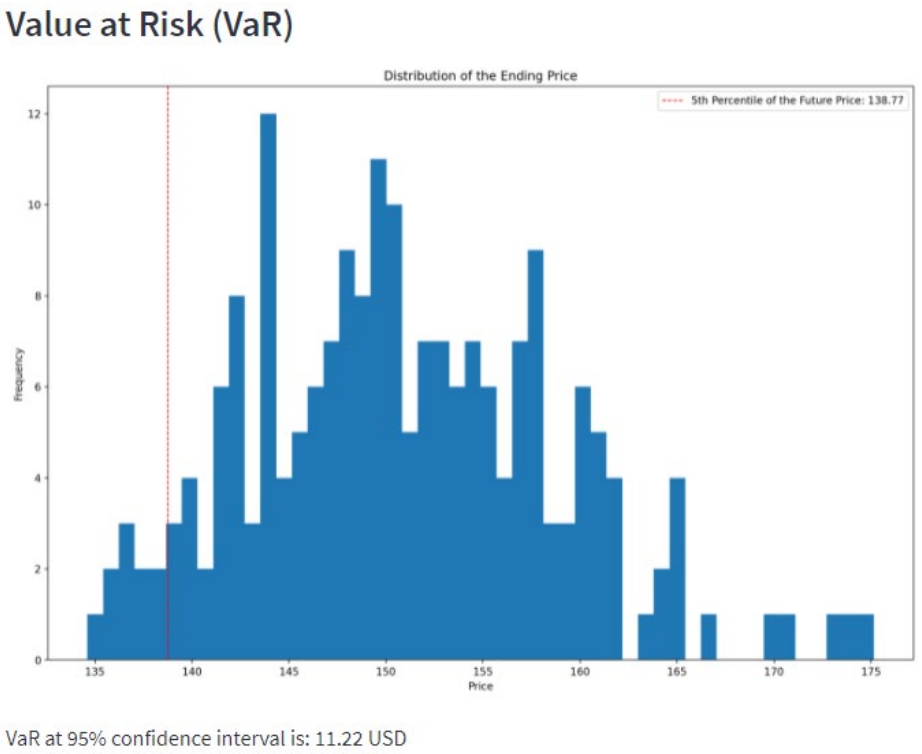
- View three plots in one figure
- Choose your desired parameters
- See datapoints as you hover



Tab 3, 4 & 5

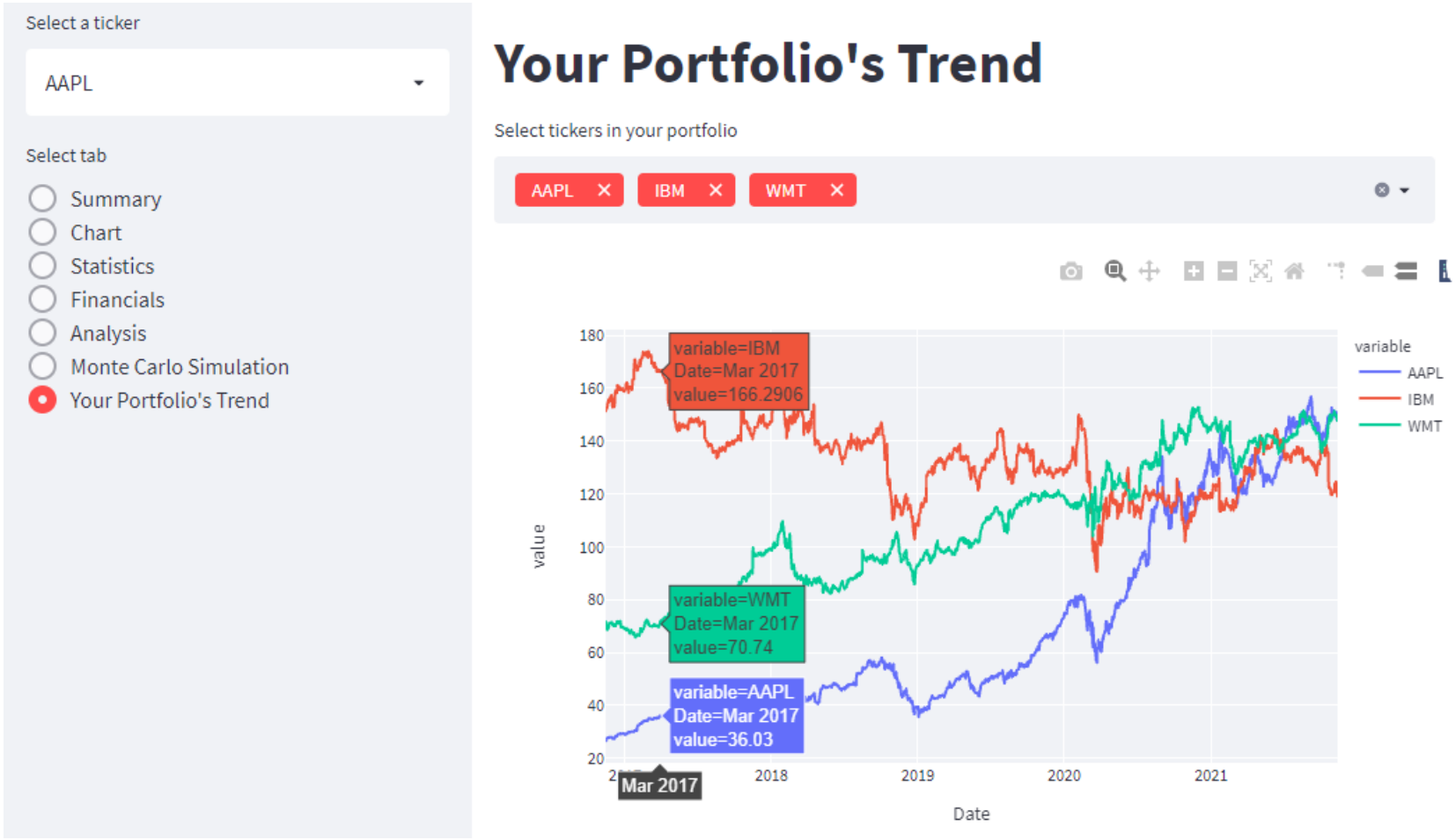
- View clearly labelled statistics
- View all financial statements with the option for period selection
- View all analysis information





Tab 6

- View stock price prediction
- Select between time horizon and number of intervals
- Value at risk calculated for you



Tab 7

- View price trend for all the stocks in your portfolio

How was the product built?

- Data was scraped from Yahoo Finance website using Python libraries **yahoo_fin.stock_info** and **Yfinance.**
- Dashboard was built using **Spyder** & **Streamlit.**
- Data was processed using Python libraries **Pandas**, **Numpy**, and **Datetime.**
- Data visualization was done using Python libraries **Matplotlib** and **Plotly.**

```python
def tab1():

    st.title("Summary")
    st.write("Select ticker on the left to begin")
    st.write(ticker)

    #The code below gets the quota table from Yahoo Finance. The streamlit page
    #is divided into 2 columns and selected columns are displayed on each side of the page.

    def getsummary(ticker):
            table = si.get_quote_table(ticker, dict_result = False)
            return table

    c1, c2 = st.columns((1,1))
    with c1:
        if ticker != '-':
            summary = getsummary(ticker)
            summary['value'] = summary['value'].astype(str)
            showsummary = summary.iloc[[14, 12, 5, 2, 6, 1, 16, 3],]
            showsummary.set_index('attribute', inplace=True)
            st.dataframe(showsummary)


    with c2:
        if ticker != '-':
            summary = getsummary(ticker)
            summary['value'] = summary['value'].astype(str)
            showsummary = summary.iloc[[11, 4, 13, 7, 8, 10, 9, 0],]
            showsummary.set_index('attribute', inplace=True)
            st.dataframe(showsummary)
```

How to scrape and display a table

```python
#The code below uses the yahoofinance package to get all the available stock
#price data. Plotly is then used to visualize the data.  An interesting feature
#from plotly called range selector is also used. A list of dictionaries
#is added to range selector to make buttons and identify the periods.
#References:
#https://plotly.com/python/range-slider/


@st.cache
def getstockdata(ticker):
    stockdata = yf.download(ticker, period = 'MAX')
    return stockdata

if ticker != '-':
        chartdata = getstockdata(ticker)

        fig = px.area(chartdata, chartdata.index, chartdata['Close'])

        |

        fig.update_xaxes(
            rangeselector=dict(
                buttons=list([
                    dict(count=1, label="1M", step="month", stepmode="backward"),
                    dict(count=3, label="3M", step="month", stepmode="backward"),
                    dict(count=6, label="6M", step="month", stepmode="backward"),
                    dict(count=1, label="YTD", step="year", stepmode="todate"),
                    dict(count=1, label="1Y", step="year", stepmode="backward"),
                    dict(count=3, label="3Y", step="year", stepmode="backward"),
                    dict(count=5, label="5Y", step="year", stepmode="backward"),
                    dict(label = "MAX", step="all")
                ])
            )
        )
        st.plotly_chart(fig)
```

Chart with buttons for selecting period

Sample Codes

```python
#The code below first obtains all the data using the download option from yahoo finance.
#It then creates a column for the simple moving average, makes the date index into a column
#and then subsets the dataframe to get just the date and and SMA column.
#Then if a duration is selected from the dropdown, data for that duration is downloaded
# and the SMA column is merged to the dataframe. If a duration is not selected then
#automatically the specified date range is used to get the data and that is also merged
#with the SMA column
#References:
#https://towardsdatascience.com/data-science-in-finance-56a4d99279f7



    @st.cache
    def getchartdata(ticker):
        SMA = yf.download(ticker, period = 'MAX')
        SMA['SMA'] = SMA['Close'].rolling(50).mean()
        SMA = SMA.reset_index()
        SMA = SMA[['Date', 'SMA']]

        if duration != '-':
            chartdata1 = yf.download(ticker, period = duration, interval = inter)
            chartdata1 = chartdata1.reset_index()
            chartdata1 = chartdata1.merge(SMA, on='Date', how='left')
            return chartdata1
        else:
            chartdata2 = yf.download(ticker, start_date, end_date, interval = inter)
            chartdata2 = chartdata2.reset_index()
            chartdata2 = chartdata2.merge(SMA, on='Date', how='left')
            return chartdata2
```

Data Manipulation

```python
#The code below uses plotly to visualize the data. Subplots from plotly is used to make 2 y axis.
#First y axis shows the stock close price and SMA and the second is used to show volume.
#Plotly graph objects are used to add graphs to the axes.The range for the y axis for
#volume is manipulated so that the bars appear small.
#References:
#https://plotly.com/python/multiple-axes/

    if ticker != '-':
            chartdata = getchartdata(ticker)


            fig = make_subplots(specs=[[{"secondary_y": True}]])

            if plot == 'Line':
                fig.add_trace(go.Scatter(x=chartdata['Date'], y=chartdata['Close'], mode='lines',
                                         name = 'Close'), secondary_y = False)
            else:
                fig.add_trace(go.Candlestick(x = chartdata['Date'], open = chartdata['Open'],
                                             high = chartdata['High'], low = chartdata['Low'], close = chartdata['Close'], name = 'Candle'))


            fig.add_trace(go.Scatter(x=chartdata['Date'], y=chartdata['SMA'], mode='lines', name = '50-day SMA'), secondary_y = False)

            fig.add_trace(go.Bar(x = chartdata['Date'], y = chartdata['Volume'], name = 'Volume'), secondary_y = True)

            fig.update_yaxes(range=[0, chartdata['Volume'].max()*3], showticklabels=False, secondary_y=True)


            st.plotly_chart(fig)
```

Multiple graphs in same figure

Sample Codes