# Week 1 – Domain Knowledge

## Day 1

First day of my internship with AmBank group. Introduced with the department members as well as getting to know the domain knowledge and task briefing. Was assigned a new project that involves dividend investment strategies, where I will be assigned to work on a system that simplifies the process. First day focuses on learning more about the fundamentals on dividend investment.

## Day 2

Working from home. Received a call from my supervisor where I asked about their dividend investment system, such as the framework used and the API. Continued my research to gain domain knowledge of stock investment and dividends.

## Day 3

Working from home. Involved myself by looking up financial records of various companies to study the important fields and assess share performance. Gained additional domain understanding before working on the system. Tried out web scraping technique to gather financial records from Yahoo Finance to output real time data into a Flask web application.

## Day 4

Working from home. Created logs and slides that outlines the system requirements such as explaining what calculations are required to assess the dividend performance. Researched on other data collection techniques such as Alpha Vantage API and Pandas DataReader API. Tested out the APIs available functions with small prototypes to assess if it fits system requirements.

## Day 5

Working in office. Reported supervisor of my progress and what I have learnt so far. Received additional details on the domain knowledge as well as the stock environment. Researched on the differences between utilizing web scraping and API for data collection and their application. Researched on the most important metrics and ratios to assess company stock performance.

# Week 2 – API Research & Testing

## Day 1

Working from home. Created more slides and logs on the differences between utilizing web scraping and API for data collection and their suited application. Tested with a small prototype on how Pandas DataReader works. Ran into some issues as Yahoo Finance has made some changes which has made DataReader unable to access, rendering it unusable. The issue is researched, addressed and will attempt to solve starting tomorrow.

import pandas\_datareader.data as pdr

def get\_stock\_data(symbol, date):

        #date\_format = "%Y-%m-%d"

        #date\_obj = dt.datetime.strptime(date, date\_format)

        start = dt.datetime(2010, 1, 1)

        end = dt.datetime (2010, 1, 2)

        stock\_data = pdr.DataReader(name = "BTC-USD", data\_source = "yahoo", start = start, end = end)

        print(stock\_data)

        print("hi")

        if len(stock\_data) == 0:

            print(f"No data available for {symbol} on {date}.")

            return None

        else:

            print('No data found for the given symbol.')

            return None

# Your Flask route remains mostly the same

@views.route('/', methods=['GET', 'POST'])

@login\_required

def home():

    if request.method == 'POST':

        company\_symbol = request.form.get('Companysymbol')

        start\_date = request.form.get('StartDate')

        if not company\_symbol:

            flash('Please enter a company symbol', category='error')

        else:

            stock\_data = get\_stock\_data(company\_symbol, start\_date)

            if stock\_data is None:

                flash('Company symbol not found or data retrieval error!', category='error')

            else:

                new\_company\_info = CompanyInfo(

                    companyname=company\_symbol,

                    price=stock\_data['price'],

                    value\_diff=stock\_data['value\_diff'],

                    perc\_diff=stock\_data['perc\_diff'],

                    user\_id=current\_user.id

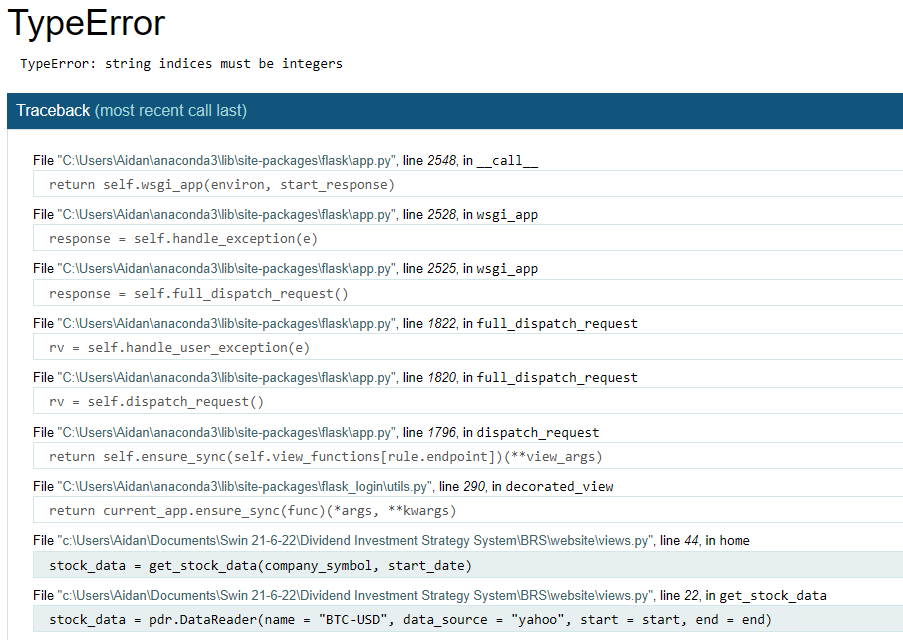
                )

                db.session.add(new\_company\_info)

                db.session.commit()

                flash('Company information added!', category='success')

    return render\_template("home.html", user=current\_user)



Yahoo made changes to their API that broke compatibility with previous pandas datareader versions.

## Day 2

Working from home. Finally addressed the Pandas DataReader issue and concluded that Yahoo has made some internal changes which has made the API unusable. To solve this, I switched to another API library to yfinance which serves the same function. I have successfully requested for company’s financial data with the new library in Pandas Dataframe. Started designing Entity Relationship Diagram (ERD) to structure how data is stored in the proposed system.

def get\_stock\_data(symbol, start\_date):

    try:

        # Convert the start\_date string to a datetime object

        startdate = dt.datetime.strptime(start\_date, '%Y-%m-%d')

        # Create a Ticker object for the specified symbol

        tic ker = yf.Ticker(symbol)

        # Get historical data (adjust parameters as needed)

        historical\_data = ticker.history(period="1d", start=startdate)

        if historical\_data.empty:

            print(f"No data available for {symbol} on {startdate}.")

            return None

        # Extract the latest data (usually the last row in the DataFrame)

        latest\_data = historical\_data.iloc[-1]

        stock = {

            'open': latest\_data['Open'],

            'close': latest\_data['Close'],

            'dividends': latest\_data['Dividends'],

        }

        return stock

    except Exception as e:

        print(f"An error occurred: {str(e)}")

        return None

@views.route('/', methods=['GET', 'POST'])

@login\_required

def home():

    if request.method == 'POST':

        company\_symbol = request.form.get('Companysymbol')

        start\_date = request.form.get('StartDate')

        if not company\_symbol:

            flash('Please enter a company symbol', category='error')

        else:

            stock\_data = get\_stock\_data(company\_symbol, start\_date)

            if stock\_data is None:

                flash('Company symbol not found or data retrieval error!', category='error')

            else:

                new\_company\_info = CompanyInfo(

                    companyname=company\_symbol,

                    price=stock\_data['open'],

                    value\_diff=stock\_data['close'],

                    perc\_diff=stock\_data['dividends'],

                    user\_id=current\_user.id

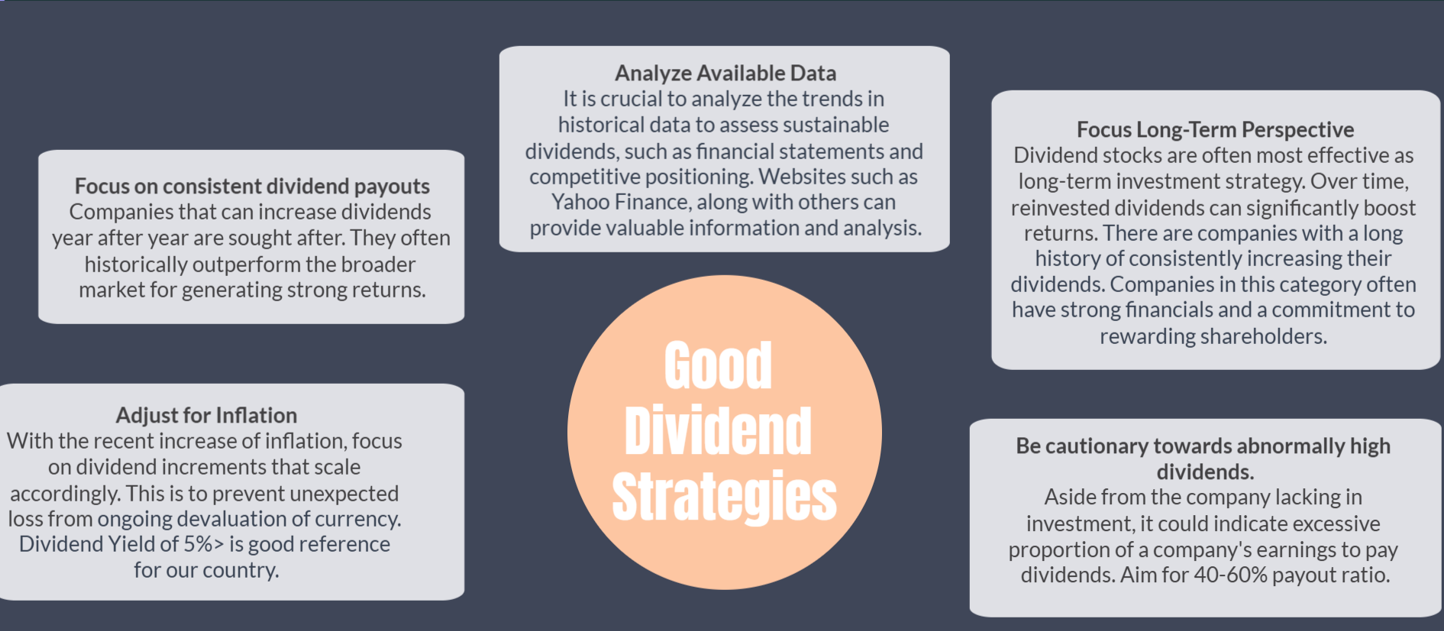
                )

                db.session.add(new\_company\_info)

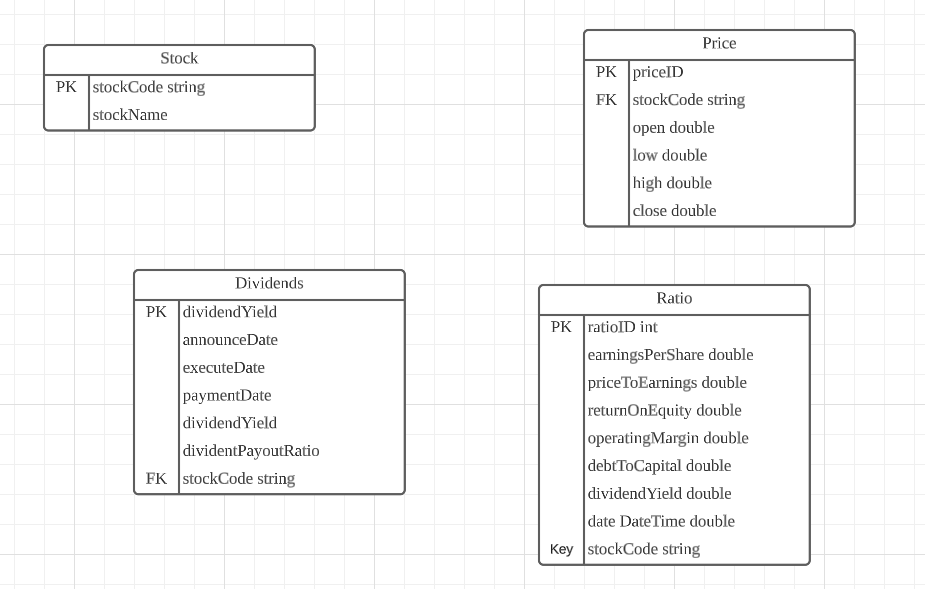
                db.session.commit()

                flash('Company information added!', category='success')

Changed API library from Pandas DataReader to yfinance. Successfully retrieved financial data by passing stock symbol as the parameters and stored in the SQL database prototype.



Designed Slide for Good Dividend Strategies



Started with the initial design of Entity Relationship Diagram (ERD) to to structure how data is stored in the proposed system.

## Day 3

Working from home. Researched and tested more functions of the yfinance library. Created a new Jupyter Notebook Project as a platform to easily test the library. Crosschecked with Yahoo Finance website and researched more on the financial metrics to choose the most important one to scrape for the proposed system.

## Day 4

Working from home. Primarily researched on the most important financial ratios and documented their definitions and ideal values. Discovered and surveyed new financial stock information website, which are Bursa Malaysia and KLSE screener to analyze later for web scraping.

## 1. Payout Ratio

Retrieved by Earnings / Dividends Paid. Can be retrieved with Ticker.info

The dividend payout ratio measures the proportion of a company's earnings that are paid out as dividends. A lower ratio suggests that a company is retaining more earnings for reinvestment or future dividends,

## 2. Dividend Yield

Dividends per share / Share price. Can be retrieved with Ticker.info or Nasdaq

It represents the income generated from an investment in the form of dividends.

## 4. Earnings Per Share (EPS) (chatgpt)

Net Income / Outstanding Shares. Can be retrieved with Ticker.info (trailingeps key)

The portion of a company's profit attributable to each outstanding share of common stock. A higher EPS can support higher dividend payments which increases likelihood of good dividend returns.

## 5. Debt to Equity Ratio (chatgpt)

Liabilities or Debt / Shareholders’ Equity. Can be retrieved with Ticker.info

A high debt-to-equity ratio can signal financial leverage, which may affect a company's ability to maintain dividend payments.

## 6. Price to Earnings Ratio

Share Price / Earnings per Share. Can be retrieved with Ticker.info

## 7. Operating Margin

(Operating Income / Total Revenue) \* 100%. Can be retrieved with Ticker.info

## 8. Return on Equity (ROE)

(Earnings – Dividends) / Shareholders Equity. Can be retrieved with Ticker.info

## 9. Return on Assets (ROA)

Net Income / Average Total Assets. Can be retrieved with Ticker.info

## 10. Profit Margin

Profit / Revenue. Can be retrieved with Ticker.info

## 11. Quick Ratio (gocardless)

(Current Assets – Inventory) / Current Liabilities. Can be retrieved by Ticker.info

## 12. Current Ratio or Working Capital Ratio (gocardless)

Current Assets / Debts or Liabilities. Can be retrieved with Ticker.info

## 13. Dividend Growth Rate (chatgpt)

(Current Dividend - Previous Dividend) / Previous Dividend. Can be retrieved by calculating Ticker.dividends

Measures the rate at which a company's dividends have been increasing over time. Consistent and sustainable dividend growth is often seen as a positive sign.

# Week 3 – Created Flask Website Template & Learnt Web Scraping

## Day 1

Working from home. Mainly focused on completing the first iteration of the Entity Relationship Diagram. Discussed with supervisor where he provided recommendations on what fields and tables to include as well as providing normalization tips. New database design proposal has been accepted as well as open for future changes.

A computer screen shot of a computer

Description automatically generated

## Day 2

Working from home. Can start proceeding with bulk web scraping now that I have successfully learnt the fundamentals of the required libraries and the process itself. Downloaded a list of stock symbols from the official Bursa Malaysia where it can be scraped via PDF tabula library. Retrieved the list of stocks to be put in CSV and inserted in the database schema.

## Day 3

Working from home. Moved on to start scraping historical prices from Yahoo Finance of all the stock symbols that was retrieved. Created an intelligent looping process with Python where it iterates through the list of stocks and passing them each into YFinance to retrieve all daily historical data from 2022-2023. 1 million total records were received in CSV and inserted into the database schema.

## Day 4

Working from home. Created another variation of the previous intelligent loop that now supports scraping of latest data as a way for the system to be up to date with the latest stock prices. By checking the date column, the loop keeps track of the latest entry of each stock symbol where YFinance is called again to gather any new records from that latest date to the present date. Any new records found will be inserted into the database schema.

## Day 5

Working in the office. Reported to supervisor on my progress of scraping respective records from Bursa Malaysia and Yahoo Finance. Received additional tips on how to analyze these retrieved data and how to gain additional useful insights from it. Showed the important financial metrics that can be possibly scraped from symbols in KLSE Screener. Received feedback on overall progress and to continue at the current pace.

# Week 4 – Continue Scraping Stock Values

## Day 1

Working from home. Continued the Python script where it now scrapes Historical Dividend data in KLSE Screener website. The code iterates through the stock symbol in the list to browse in the website before searching for a table element containing the targeted fields. It will then retrieve all the important values in the table into a Dataframe where it will be cleaned and exported to CSV.

## Day 2

Working from home. Continued the Python script where it now scrapes the Quarterly & Annual Report data in the KLSE Screener website. It iterates through the list to browse accordingly before searching for the table element containing the targeted fields. The data is retrieved into a Dataframe where it will be cleaned and exported to CSV.

## Day 3

Working from home. Created a new Python Script where it scrapes from a new website source, which is Bursa Marketplace. A new process is created where the search engine in the website is used to get the link of the symbol details by automatically filling in with the stock symbol. The link is then browsed before searching for the required field which are Price Earnings and Price to Book ratios.

## Day 4

Working form home. Continued the previous Python Script to browse to a different section in the symbol details pages. The link is browsed automatically before searching for the required fields which are Dividend Growth Rate and Price to Cash Flow.

## Day 5

Working in office. Showed and demonstrated to the supervisor on my progress as well as my learning on web scraping and data retrieval. Presented all the fields and values that was retrieved and received feedback on what are the important fields to include for my proposed system to assess. Ended up with less than half of scraped data to be useful.