Dividend Discount Model Data Collection and Export

Create an Empty DataFrame

Initialize an empty DataFrame, `all_data`, to serve as a container for the combined data of all stock tickers.

Loop through Each Ticker

For each stock ticker in our list (`stocks`), we'll download historical data for the specified date range.

Download and Format Data

- Use the `yfinance` library to download historical stock data, including prices and dividends.
- Format the date index to display only the date (removing the timestamp) for clarity in the final dataset.

Filter for Dividend Payments

- Filter the data to include only dates where dividends were paid (i.e., rows where the `Dividends` column is greater than zero).
- Select relevant columns, such as `Adj Close`, `Close`, and `Dividends`, to focus on essential data for dividend analysis.

Handle Missing Data

Remove any rows with NaN values to ensure the data is clean and ready for analysis.

Display and Verify

Display the dividend data for each stock ticker as we go to ensure accuracy before merging all data into `all_data`.

Explanation: Here, we import the essential libraries for our data analysis and portfolio optimization. This includes pandas for data handling and numpy for numerical operations.

- **yfinance**: A library that allows easy access to financial data from Yahoo Finance, making it convenient to download historical price data for stocks and other financial assets.
- **pandas**: A powerful data manipulation library in Python that provides data structures like DataFrames for handling large datasets with ease.
- **pandas_datareader**: A library that retrieves data from various online sources, including Yahoo Finance, though `yfinance` is primarily used here.
- **datetime**: A Python module for manipulating dates and times, essential for setting up date ranges for data retrieval.
- **numpy**: A library that provides support for large, multi-dimensional arrays and matrices, as well as a collection of mathematical functions to operate on these arrays.
- **os**: A module that provides a way of using operating system-dependent functionality, such as file and directory management.

In [2]:
```
import yfinance as yf
import pandas as pd
from pandas_datareader import data as pdr
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import os
from IPython.display import display  # For displaying data nicely in Jupyter Notebo
import datetime as dt

# Configure pandas to display full text in cells
pd.set_option('display.max_colwidth', None)  # Show full text in columns without tr
pd.set_option('display.max_columns', None)   # Show all columns
pd.set_option('display.expand_frame_repr', False)  # Disable line wrapping
# pd.set_option('display.max_rows', None)   # Uncomment to show all rows
pd.set_option('display.max_columns', None)
```

In this step, we set up the date range for our analysis. Specifically, we calculate today's date and a date 5 years ago to serve as our start date for data retrieval.

In [3]:
```
endDate = dt.datetime.now().date()
startDate = (dt.datetime.now() - dt.timedelta(days=365*10)).date()

endDate, startDate
```

Out[3]:
```
(datetime.date(2024, 11, 7), datetime.date(2014, 11, 10))
```

Here, we define a list of stock tickers that will make up our portfolio. These represent a mix of asset classes to diversify the portfolio.

Tickers in the Portfolio

Here's a list of 10 U.S. companies known for stable dividend payments:

- **International Business Machines Corp. (IBM)**: A major player in information technology with a strong history of dividend payments.
- **NextEra Energy (NEE)**: A leader in renewable energy, known for reliable dividends.
- **Caterpillar Inc. (CAT)**: A heavy equipment manufacturer with a long track record of increasing dividends.
- **Realty Income Corporation (O)**: A real estate investment trust (REIT) that pays monthly dividends.
- **Albemarle Corporation (ALB)**: A major lithium producer with regular dividend payouts.
- **Essex Property Trust (ESS)**: A REIT specializing in residential properties, offering consistent dividends.
- **Brown & Brown, Inc. (BRO)**: An insurance brokerage firm with steady dividend payments.
- **West Pharmaceutical Services, Inc. (WST)**: A healthcare company known for regular dividends.
- **Ecolab Inc. (ECL)**: A provider of water treatment and hygiene solutions, with stable dividend payments.
- **Chevron Corporation (CVX)**: One of the largest oil and gas companies globally, known for steady dividends.

In [8]:
```
stocks = ['IBM', 'NEE', 'CAT', 'O', 'ALB', 'ESS', 'BRO', 'WST', 'ECL', 'CVX']
```

In [9]:
```
# Path for saving the file
output_folder = r"C:\Users\qwerty\Desktop\Python\JNotebook\start\Dividend Discount
output_path = os.path.join(output_folder, 'Dividend Discount Model.xlsx')

# Create a dictionary to store the latest closing prices
latest_close_prices = {}

# Create an Excel file with separate sheets
with pd.ExcelWriter(output_path) as writer:
    for stock in stocks:
        # Download data for the ticker
        df = yf.download(stock, start=startDate, end=endDate, actions=True)
        df.index = df.index.date

        # Filter for dividends and remove NaN values
        dividend_data = df[df["Dividends"] > 0][["Adj Close", "Close", "Dividends"]

        # Add dividend data to a separate sheet
        dividend_data.to_excel(writer, sheet_name=stock)

        # Save the latest closing price
        latest_close_prices[stock] = df["Close"].iloc[-1]

    # Create a DataFrame with the latest closing prices
    latest_close_df = pd.DataFrame(list(latest_close_prices.items()), columns=["Tic
```

```
    # Write the DataFrame with the latest closing prices to a separate sheet
    latest_close_df.to_excel(writer, sheet_name="Close Price Today", index=False)

print(f"Data successfully saved to {output_path} with a separate sheet for today's
```

```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
Data successfully saved to C:\Users\qwerty\Desktop\Python\JNotebook\start\Dividend D
iscount Model (Python + Excel)\Dividend Discount Model.xlsx with a separate sheet fo
r today's closing prices.
```

The next step is to analyze the assets and calculate the Sharpe ratio in 'optimal_portfolio.xlsx'. Additionally, refer to the README for all necessary instructions.