# Analysis of a Company

## Financial Parameter Analysis

When a pension house evaluates a prospective company, it typically conducts an in-depth analysis of various financial parameters to assess the company’s financial health, stability, and growth potential. These parameters often include:

### Profitability Ratios

#### Net Profit Margin, Return on Equity (ROE) and Return on Assets (ROA)

1. The **Net Profit Margin** measures how much of each dollar in revenue the company retains as profit after all expenses. It is calculated using the following formula:

where,

* **Net Income**: The company's total earnings (profit) after taxes and all expenses.
* **Revenue**: The total income generated from sales of goods or services before any expenses are subtracted.

1. **Return on Equity (ROE)**: ROE measures a company’s profitability in relation to shareholders' equity, indicating how effectively the company uses equity to generate profit. It is calculated as:

where,

* **Net Income:** The company's total earnings after taxes and expenses.
* **Shareholders' Equity:** The residual interest in the assets of the company after deducting liabilities (essentially, the company's net worth).

1. **Return on Assets (ROA)**: ROA indicates how efficiently a company is using its assets to generate profit. It measures the profitability relative to the company's total assets. The formula for calculating ROA is:

Where,​

* **Net Income:** The company's total earnings after all expenses, taxes, and costs.
* **Total Assets:** The total value of everything the company owns (current assets, fixed assets, etc.).

**Interpretation**

* **Net Profit Margin**:
  + A **high net profit margin** indicates efficient management and good cost control, allowing a company to retain a significant portion of its revenue as profit.
  + A **low net profit margin** suggests that the company may be struggling with high costs or inefficient operations.
* **Return on Equity (ROE)**:
  + A **high ROE** indicates that the company is effectively using shareholders' equity to generate profit.
  + A **low ROE** might suggest poor management performance or inefficient use of equity.
* **Return on Assets (ROA):**
  + **High ROA:** indicates that the company is efficiently utilizing its assets to generate earnings. It is generally considered a positive sign of management effectiveness.
  + **Low ROA:** suggests that the company is not effectively using its assets to generate profit. This might indicate inefficiencies or a need for asset optimization.

ROA is particularly useful for comparing the performance of companies within the same industry, as asset structures can vary significantly between different industries.

**Python Code**

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data  
msft = yf.Ticker("MSFT")  
  
# Fetch annual financials (income statement and balance sheet)  
financials = msft.financials  
income\_statement = msft.financials  
balance\_sheet = msft.balance\_sheet  
  
# Extract net income, total revenue and total assets for past 5 years  
net\_income = income\_statement.loc["Net Income"]  
revenue = financials.loc["Total Revenue"]  
total\_assets = balance\_sheet.loc["Total Assets"]  
shareholders\_equity = balance\_sheet.loc["Stockholders Equity"]  
  
# Calculate Net Profit Margin  
net\_profit\_margin = (net\_income / revenue) \* 100  
  
# Calculate ROE (Return on Equity)  
roe = (net\_income / shareholders\_equity) \* 100  
  
# Calculate ROA for each year  
roa = net\_income / total\_assets  
  
# Prepare a DataFrame with the relevant data  
profitability\_ratios = pd.DataFrame(  
 {  
 "Year": net\_income.index.year,  
 "Net Income": net\_income.values,  
 "Revenue": revenue.values,  
 "Shareholders Equity": shareholders\_equity.values,  
 "Net Profit Margin (%)": net\_profit\_margin.values,  
 "ROE (%)": roe.values,  
 "Total Assets": total\_assets.values,  
 "ROA": roa.values,  
 }  
)  
profitability\_ratios

### Liquidity Ratios

#### Current Ratio and Quick Ratio

1. **Current Ratio**: This ratio measures a company’s ability to pay its short-term obligations with its short-term assets. It is calculated using the following formula:
   * **Current Assets**: Assets that are expected to be converted into cash or used up within one year (e.g., cash, inventory, accounts receivable).
   * **Current Liabilities**: Obligations that are due within one year (e.g., accounts payable, short-term debt).
2. **Quick Ratio**: Also known as the Acid-Test Ratio, it measures a company’s ability to meet its short-term obligations using its most liquid assets (excluding inventory). It is calculated as:

*​*

* + This ratio excludes inventory from current assets because inventory is not as readily convertible to cash as other current assets.

**Interpretation**

* **Current Ratio**:
  + A **current ratio above 1** indicates that the company has more current assets than current liabilities, suggesting good short-term financial health.
  + A **current ratio below 1** could suggest potential liquidity issues, indicating that the company might struggle to meet its short-term obligations.
* **Quick Ratio**:
  + A **quick ratio above 1** suggests that the company can meet its short-term liabilities with its most liquid assets, even without relying on the sale of inventory.
  + A **quick ratio below 1** could indicate potential liquidity issues, implying that the company may struggle to cover its short-term liabilities with its most liquid assets.

**Python Code**

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data and balance sheet data for the past 5 years  
msft = yf.Ticker("MSFT")  
balance\_sheet = msft.balance\_sheet  
  
# Extract relevant data from the balance sheet  
current\_assets = balance\_sheet.loc['Current Assets']  
current\_liabilities = balance\_sheet.loc['Current Liabilities']  
inventory = (balance\_sheet.loc['Inventory']  
  
# Handle cases where inventory is zero or not listed  
# inventory = (balance\_sheet.loc['Inventory']   
# if 'Inventory' in balance\_sheet.index   
# else pd.Series([0] \* len(current\_assets), index=current\_assets.index)   
#)  
  
# Calculate Current Ratio  
current\_ratio = current\_assets / current\_liabilities  
  
# Calculate Quick Ratio  
quick\_ratio = (current\_assets - inventory) / current\_liabilities  
  
# Prepare a DataFrame with the relevant data  
liquidity\_metrics = pd.DataFrame({  
 'Year': current\_assets.index.year,  
 'Current Assets': current\_assets.values,  
 'Current Liabilities': current\_liabilities.values,  
 'Inventory': inventory.values,  
 'Current Ratio': current\_ratio.values,  
 'Quick Ratio': quick\_ratio.values,  
})  
  
# Display the liquidity metrics  
liquidity\_metrics

### Leverage Ratios

#### Debt-to-Equity Ratio and Interest Coverage Ratio

1. **Debt-to-Equity Ratio**: This ratio measures a company’s financial leverage by comparing its total liabilities to its shareholders' equity. It indicates how much debt a company is using to finance its assets relative to the value of shareholders' equity.
   * **Total Liabilities**: The sum of all the company's debts and obligations, including both current and long-term liabilities.
   * **Shareholders' Equity**: The residual interest in the assets of the company after deducting liabilities, essentially the company’s net worth.
2. **Interest Coverage Ratio**: This ratio measures a company’s ability to pay interest on its outstanding debt. It shows how easily a company can cover its interest obligations with its operating income.
   * **EBIT (Earnings Before Interest and Taxes)**: A measure of a company's profitability that excludes interest and income tax expenses.
   * **Interest Expense**: The cost incurred by an entity for borrowed funds.

**Interpretation**

* **Debt-to-Equity Ratio**:
  + A **high debt-to-equity ratio** suggests that the company is heavily financed by debt relative to equity, which could indicate higher financial risk if the company faces difficulties meeting its debt obligations.
  + A **low debt-to-equity ratio** indicates that the company is using less debt to finance its operations, which could imply lower financial risk.
* **Interest Coverage Ratio**:
  + A **high interest coverage ratio** indicates that the company has ample earnings to cover its interest payments, suggesting strong financial health.
  + A **low interest coverage ratio** (below 1.5) could indicate that the company may struggle to meet its interest obligations, posing a risk of financial distress.

**Python Code**

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data and financial statements for the past 5 years  
msft = yf.Ticker("MSFT")  
balance\_sheet = msft.balance\_sheet  
financials = msft.financials  
  
# Extract relevant data from the balance sheet and income statement  
total\_liabilities = balance\_sheet.loc['Total Liabilities']  
shareholders\_equity = balance\_sheet.loc['Total Stockholder Equity']  
ebit = financials.loc['Ebit'] # Earnings Before Interest and Taxes  
interest\_expense = financials.loc['Interest Expense'] if 'Interest Expense' in financials.index else pd.Series([0] \* len(ebit), index=ebit.index) # Handle cases where interest expense is zero or not listedExpense' in financials.index else pd.Series([0] \* len(ebit), index=ebit.index) # Handle cases where interest expense is zero or not listed  
  
# Calculate Debt-to-Equity Ratio  
debt\_to\_equity\_ratio = total\_liabilities / shareholders\_equity  
  
# Calculate Interest Coverage Ratio  
interest\_coverage\_ratio = ebit / interest\_expense.replace(0, float('nan')) # Avoid division by zero by replacing 0 with NaN  
  
# Prepare a DataFrame with the relevant data  
leverage\_ratios = pd.DataFrame({  
 'Year': total\_liabilities.index.year,  
 'Total Liabilities': total\_liabilities.values,  
 'Shareholders Equity': shareholders\_equity.values,  
 'EBIT': ebit.values,  
 'Interest Expense': interest\_expense.values,  
 'Debt-to-Equity Ratio': debt\_to\_equity\_ratio.values,  
 'Interest Coverage Ratio': interest\_coverage\_ratio.values,  
})  
  
# Display the financial ratios  
leverage\_ratios

### Efficiency Ratios

#### Asset Turnover Ratio and Inventory Turnover Ratio

1. **Asset Turnover Ratio**: This ratio measures how efficiently a company uses its assets to generate sales. It indicates how many dollars of sales are generated for each dollar of assets owned by the company.

​

* + **Revenue**: The total income generated from the sale of goods or services.
  + **Average Total Assets**: The average of total assets at the beginning and end of the period. It is calculated as .

1. **Inventory Turnover Ratio**: This ratio measures how efficiently a company manages its inventory by comparing the cost of goods sold (COGS) to the average inventory. It shows how many times a company’s inventory is sold and replaced over a period.

​

* + **Cost of Goods Sold (COGS)**: The direct costs attributable to the production of goods sold by the company.
  + **Average Inventory**: The average of inventory at the beginning and end of the period. It is calculated as .

**Interpretation**

* **Asset Turnover Ratio**:
  + A **high asset turnover ratio** indicates that the company is using its assets efficiently to generate sales.
  + A **low asset turnover ratio** suggests that the company may not be using its assets effectively, which could indicate operational inefficiencies.
* **Inventory Turnover Ratio**:
  + A **high inventory turnover ratio** indicates efficient inventory management, as the company is able to sell its inventory quickly.
  + A **low inventory turnover ratio** suggests that the company might be overstocking or facing difficulties in selling its inventory, leading to potential liquidity issues.

**Python Code**

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data and financial statements for the past 5 years  
msft = yf.Ticker("MSFT")  
financials = msft.financials  
balance\_sheet = msft.balance\_sheet  
  
# Extract relevant data from the financial statements  
revenue = financials.loc['Total Revenue']  
cogs = financials.loc['Cost Of Revenue']  
total\_assets = balance\_sheet.loc['Total Assets']  
inventory = balance\_sheet.loc['Inventory'] if 'Inventory' in balance\_sheet.index else pd.Series([0] \* len(total\_assets), index=total\_assets.index) # Handle cases where inventory is zero or not listed  
  
# Calculate Average Total Assets for Asset Turnover Ratio  
average\_total\_assets = (total\_assets + total\_assets.shift(1)) / 2  
  
# Calculate Asset Turnover Ratio  
asset\_turnover\_ratio = revenue / average\_total\_assets  
  
# Calculate Average Inventory for Inventory Turnover Ratio  
average\_inventory = (inventory + inventory.shift(1)) / 2  
  
# Calculate Inventory Turnover Ratio  
inventory\_turnover\_ratio = cogs / average\_inventory.replace(0, float('nan')) # Avoid division by zero by replacing 0 with NaN  
  
# Prepare a DataFrame with the relevant data  
turnover\_ratios = pd.DataFrame({  
 'Year': revenue.index.year,  
 'Revenue': revenue.values,  
 'COGS': cogs.values,  
 'Total Assets': total\_assets.values,  
 'Inventory': inventory.values,  
 'Asset Turnover Ratio': asset\_turnover\_ratio.values,  
 'Inventory Turnover Ratio': inventory\_turnover\_ratio.values,  
})  
  
# Display the turnover ratios  
turnover\_ratios

### Valuation Metrics

#### Price-to-Earnings (P/E) Ratio and Enterprise Value to EBITDA (EV/EBITDA)

1. **Price-to-Earnings (P/E) Ratio**: This ratio measures a company’s current share price relative to its per-share earnings, indicating how much investors are willing to pay per dollar of earnings.

P/E Ratio=Market Price per ShareEarnings per Share (EPS)\text{P/E Ratio} = \frac{\text{Market Price per Share}}{\text{Earnings per Share (EPS)}}P/E Ratio=Earnings per Share (EPS)Market Price per Share​

* + **Market Price per Share**: The current market price of one share of the company’s stock.
  + **Earnings per Share (EPS)**: The company's net income divided by the number of outstanding shares.

1. **Enterprise Value to EBITDA (EV/EBITDA) Ratio**: This ratio measures a company’s enterprise value (EV) relative to its earnings before interest, taxes, depreciation, and amortization (EBITDA). It is often used to assess the valuation of companies, especially for comparing companies within the same industry.

EV/EBITDA=Enterprise Value (EV)EBITDA\text{EV/EBITDA} = \frac{\text{Enterprise Value (EV)}}{\text{EBITDA}}EV/EBITDA=EBITDAEnterprise Value (EV)​

* + **Enterprise Value (EV)**: A measure of a company’s total value, calculated as market capitalization plus total debt minus cash and cash equivalents.
  + **EBITDA**: A measure of a company's overall financial performance that excludes interest, taxes, depreciation, and amortization.

**Interpretation**

* **P/E Ratio**:
  + A **high P/E ratio** might indicate that the stock is overvalued, or investors are expecting high growth rates in the future.
  + A **low P/E ratio** might suggest that the stock is undervalued or that the company is facing challenges.
* **EV/EBITDA Ratio**:
  + A **low EV/EBITDA ratio** might indicate that the company is undervalued or generating strong earnings relative to its enterprise value.
  + A **high EV/EBITDA ratio** might suggest that the company is overvalued or not generating sufficient earnings compared to its enterprise value.

Python Code

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data and financial statements for the past 5 years  
msft = yf.Ticker("MSFT")  
  
# Fetch historical market price and calculate EPS  
prices = msft.history(period="5y")  
financials = msft.financials  
balance\_sheet = msft.balance\_sheet  
  
# Extract relevant data  
net\_income = financials.loc['Net Income']  
shares\_outstanding = msft.info['sharesOutstanding']  
eps = net\_income / shares\_outstanding  
market\_price\_per\_share = prices['Close'].resample('Y').last() # Use the last closing price of each year  
  
# Calculate P/E Ratio  
pe\_ratio = market\_price\_per\_share / eps  
  
# Calculate EV/EBITDA Ratio  
ebitda = financials.loc['Ebitda'] # Use EBITDA from financials  
total\_debt = balance\_sheet.loc['Long Term Debt'] + balance\_sheet.loc['Short Long Term Debt'] if 'Short Long Term Debt' in balance\_sheet.index else balance\_sheet.loc['Long Term Debt']  
cash = balance\_sheet.loc['Cash'] if 'Cash' in balance\_sheet.index else pd.Series([0] \* len(total\_debt), index=total\_debt.index) # Handle cases where cash is zero or not listed  
  
market\_cap = market\_price\_per\_share \* shares\_outstanding  
enterprise\_value = market\_cap + total\_debt - cash  
ev\_ebitda\_ratio = enterprise\_value / ebitda  
  
# Prepare a DataFrame with the relevant data  
valuation\_ratios = pd.DataFrame({  
 'Year': eps.index.year,  
 'Market Price per Share': market\_price\_per\_share.values,  
 'EPS': eps.values,  
 'P/E Ratio': pe\_ratio.values,  
 'EBITDA': ebitda.values,  
 'Total Debt': total\_debt.values,  
 'Cash': cash.values,  
 'Enterprise Value': enterprise\_value.values,  
 'EV/EBITDA Ratio': ev\_ebitda\_ratio.values,  
})  
  
# Display the valuation ratios  
valuation\_ratios

### Cash Flow Analysis

#### Operating Cash Flow and Free Cash Flow

1. **Operating Cash Flow (OCF)**: Operating Cash Flow is a measure of the cash generated by a company's normal business operations. It indicates whether a company can generate sufficient positive cash flow to maintain and grow its operations.
   * **Net Income**: The company’s total earnings after taxes and all expenses.
   * **Non-Cash Expenses**: Includes depreciation, amortization, and other non-cash items.
   * **Changes in Working Capital**: The difference between current assets and current liabilities, excluding cash.
2. **Free Cash Flow (FCF)**: Free Cash Flow represents the cash a company generates after accounting for cash outflows to support operations and maintain its capital assets. It indicates the cash available to the company for expansion, paying dividends, or reducing debt.
   * **Operating Cash Flow (OCF)**: Cash generated from operations.
   * **Capital Expenditures (CapEx)**: Funds used by a company to acquire, upgrade, and maintain physical assets such as property, plants, or equipment.

**Interpretation**

* **Operating Cash Flow (OCF)**:
  + A **positive OCF** indicates that the company’s core business operations are generating sufficient cash to maintain and grow the business.
  + A **negative OCF** might suggest operational difficulties or that the company is not generating enough cash from its core business operations.
* **Free Cash Flow (FCF)**:
  + A **positive FCF** indicates that the company has sufficient cash to expand, pay dividends, reduce debt, or invest in other projects.
  + A **negative FCF** might indicate that the company needs to raise additional capital or is investing heavily in its growth, which might not be immediately profitable.

**Python Code**

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data and financial statements for the past 5 years  
msft = yf.Ticker("MSFT")  
cashflow = msft.cashflow  
  
# Extract relevant data from the cash flow statement  
net\_income = cashflow.loc['Net Income']  
depreciation = cashflow.loc['Depreciation']  
changes\_in\_working\_capital = cashflow.loc['Change In Working Capital']  
capex = cashflow.loc['Capital Expenditures']  
  
# Calculate Operating Cash Flow  
operating\_cash\_flow = net\_income + depreciation + changes\_in\_working\_capital  
  
# Calculate Free Cash Flow  
free\_cash\_flow = operating\_cash\_flow - capex  
  
# Prepare a DataFrame with the relevant data  
cash\_flow\_metrics = pd.DataFrame({  
 'Year': operating\_cash\_flow.index.year,  
 'Net Income': net\_income.values,  
 'Depreciation': depreciation.values,  
 'Changes in Working Capital': changes\_in\_working\_capital.values,  
 'Operating Cash Flow (OCF)': operating\_cash\_flow.values,  
 'Capital Expenditures (CapEx)': capex.values,  
 'Free Cash Flow (FCF)': free\_cash\_flow.values,  
})  
  
# Display the cash flow metrics  
cash\_flow\_metrics

### Dividend Analysis

#### Dividend Yield and Dividend Payout Ratio

1. **Dividend Yield**: This ratio measures how much a company pays out in dividends each year relative to its stock price. It reflects the return on investment from dividends alone.
   * **Annual Dividends per Share**: The total dividends paid per share over a year.
   * **Market Price per Share**: The current market price of one share of the company's stock.
2. **Dividend Payout Ratio**: This ratio shows the percentage of earnings distributed to shareholders in the form of dividends. It reflects how much of the net income is paid out as dividends.
   * **Total Dividends Paid**: The sum of all dividends paid to shareholders during the period.
   * **Net Income**: The company’s total earnings after taxes and expenses.

**Interpretation**

* **Dividend Yield**:
  + A **high dividend yield** might indicate a strong income stream from dividends, but it could also signal potential issues, such as a declining stock price.
  + A **low dividend yield** might suggest that the company is reinvesting more into growth or that its stock price is relatively high compared to dividends paid.
* **Dividend Payout Ratio**:
  + A **high payout ratio** means the company is paying out a large portion of its earnings as dividends, which could be a concern if it limits growth or if earnings decline.
  + A **low payout ratio** suggests the company retains more of its earnings for growth, debt reduction, or other investments.

Python Code

import yfinance as yf  
import pandas as pd  
  
# Fetch Microsoft stock data for the past 5 years  
msft = yf.Ticker("MSFT")  
  
# Fetch historical stock prices and dividends data  
prices = msft.history(period="5y")  
dividends = msft.dividends  
  
# Fetch annual financials: Net Income and calculate Total Dividends Paid  
annual\_financials = msft.financials  
net\_income = annual\_financials.loc['Net Income']  
shares\_outstanding = msft.info['sharesOutstanding']  
  
# Calculate Annual Dividends per Share by resampling dividends to yearly and summing  
annual\_dividends\_per\_share = dividends.resample('Y').sum()  
  
# Calculate Market Price per Share (use the last closing price of each year)  
market\_price\_per\_share = prices['Close'].resample('Y').last()  
  
# Calculate Total Dividends Paid (Annual Dividends per Share \* Shares Outstanding)  
total\_dividends\_paid = annual\_dividends\_per\_share \* shares\_outstanding  
  
# Calculate Dividend Yield  
dividend\_yield = (annual\_dividends\_per\_share / market\_price\_per\_share) \* 100  
  
# Calculate Dividend Payout Ratio  
dividend\_payout\_ratio = (total\_dividends\_paid / net\_income) \* 100  
  
# Prepare a DataFrame with the relevant data  
dividend\_metrics = pd.DataFrame({  
 'Year': annual\_dividends\_per\_share.index.year,  
 'Net Income': net\_income.values,  
 'Total Dividends Paid': total\_dividends\_paid.values,  
 'Annual Dividends per Share': annual\_dividends\_per\_share.values,  
 'Market Price per Share': market\_price\_per\_share.values,  
 'Dividend Yield (%)': dividend\_yield.values,  
 'Dividend Payout Ratio (%)': dividend\_payout\_ratio.values,  
})  
  
# Display the dividend metrics  
dividend\_metrics

Growth Analysis

Revenue Growth and Earnings Growth

### P/E Ratio

The **Price-to-Earnings (P/E) Ratio** is a financial metric that measures a company's current share price relative to its per-share earnings. The formula for calculating the P/E Ratio is:

where,

* **Market Price per Share**: The current market price of one share of the company's stock.
* **Earnings per Share (EPS)**: The company's earnings divided by the number of outstanding shares. EPS can be found in financial statements or calculated using net income and outstanding shares.

**Interpretation**

* **High P/E Ratio**: A high P/E ratio might indicate that the stock is overvalued, or investors are expecting high growth rates in the future.
* **Low P/E Ratio**: A low P/E ratio might suggest that the stock is undervalued, or the company is experiencing difficulties. It could also indicate a mature company with stable earnings and slower growth prospects.

#### Python Code

import yfinance as yf  
import pandas as pd  
from datetime import datetime  
  
*# Fetch Microsoft stock data for the past 5 years*  
msft = yf.Ticker("MSFT")  
prices = msft.history(period="5y")  
  
*# Fetch the quarterly earnings (net income) data from SEC filings or Yahoo Finance*  
earnings = msft.earnings  
  
*# Calculate annual EPS (Earnings per Share)*  
*# Yahoo Finance gives annual earnings, we divide by number of shares to get EPS*  
annual\_earnings = msft.financials.loc['Net Income']  
shares\_outstanding = msft.info['sharesOutstanding']  
eps = annual\_earnings / shares\_outstanding  
  
*# Prepare a DataFrame with the relevant data*  
pe\_ratios = pd.DataFrame({  
 'Year': annual\_earnings.index,  
 'Net Income': annual\_earnings.values,  
 'Shares Outstanding': shares\_outstanding,  
 'EPS': eps.values,  
 'Price per Share': prices['Close'].resample('Y').last().values, *# Get the last closing price of each year*  
})  
  
*# Calculate the P/E Ratio*  
pe\_ratios['P/E Ratio'] = pe\_ratios['Price per Share'] / pe\_ratios['EPS']  
  
*# Display the P/E Ratios*  
import ace\_tools as tools; tools.display\_dataframe\_to\_user(name="Microsoft P/E Ratios Over 5 Years", dataframe=pe\_ratios)

**Explanation**

1. **Stock Price Data**: The code uses yfinance to get historical price data for Microsoft for the last 5 years.
2. **Earnings Data**: It fetches annual net income data and divides by the number of shares outstanding to calculate EPS.
3. **P/E Calculation**: It calculates the P/E Ratio by dividing the closing stock price by the EPS for each year.
4. **Output**: The results are displayed in a DataFrame with columns for Year, Net Income, Shares Outstanding, EPS, Price per Share, and the P/E Ratio.

To execute this code, ensure you have the yfinance package installed and access to relevant financial data for accuracy.

### Dividend Yield and Dividend Payout Ratio

1. The **Dividend Yield** measures how much a company pays out in dividends each year relative to its stock price. It is calculated using the following formula:

where,

* + **Annual Dividends per Share:** The total dividends paid per share over a year.
  + **Market Price per Share:** The current market price of one share of the company's stock.

1. The **Dividend Payout Ratio**: This ratio shows the percentage of earnings distributed to shareholders in the form of dividends. It is calculated as:
2. where
3. **Total Dividends Paid**: The sum of all dividends paid to shareholders during the period.
4. **Net Income**: The company's total earnings (after taxes and expenses).

**Interpretation**

* **Dividend Yield**:
  + A **high dividend yield** might indicate a good income stream but could also suggest that the stock price is depressed or the dividend is unsustainable.
  + A **low dividend yield** might suggest a focus on reinvestment into the company rather than paying dividends.
* **Dividend Payout Ratio**:
  + A **high payout ratio** could indicate that the company is returning most of its earnings to shareholders, which may be unsustainable if earnings decline.
  + A **low payout ratio** suggests that the company is retaining more earnings for growth or debt reduction.

#### Python Code

import yfinance as yf  
import pandas as pd  
  
*# Fetch Microsoft stock data and dividends data for the past 5 years*  
msft = yf.Ticker("MSFT")  
prices = msft.history(period="5y")  
dividends = msft.dividends  
  
*# Fetch earnings (net income) data and shares outstanding*  
annual\_earnings = msft.financials.loc['Net Income']  
shares\_outstanding = msft.info['sharesOutstanding']  
  
*# Calculate Annual Dividends per Share*  
*# Resample dividends to yearly and sum to get total annual dividends per share*  
annual\_dividends\_per\_share = dividends.resample('Y').sum()  
  
*# Calculate Dividend Yield*  
*# Use the closing price of the last day of the year for the market price*  
prices\_yearly = prices['Close'].resample('Y').last()  
dividend\_yield = annual\_dividends\_per\_share / prices\_yearly  
  
*# Calculate Total Dividends Paid and Dividend Payout Ratio*  
total\_dividends\_paid = annual\_dividends\_per\_share \* shares\_outstanding  
dividend\_payout\_ratio = total\_dividends\_paid / annual\_earnings  
  
*# Prepare a DataFrame with the relevant data*  
dividend\_data = pd.DataFrame({  
 'Year': annual\_earnings.index.year,  
 'Net Income': annual\_earnings.values,  
 'Total Dividends Paid': total\_dividends\_paid.values,  
 'Annual Dividends per Share': annual\_dividends\_per\_share.values,  
 'Market Price per Share': prices\_yearly.values,  
 'Dividend Yield': dividend\_yield.values,  
 'Dividend Payout Ratio': dividend\_payout\_ratio.values,  
})  
  
*# Display the Dividend Yield and Dividend Payout Ratios*  
dividend\_data