**Comprehensive Financial Analysis for Pension Investment Decisions**

A **pension house** focuses on **long-term stability, financial strength, and consistent returns** when evaluating a company's stock. Below is a structured approach with **key financial ratios**, their formulas, interpretation, and **Python code** to automate this analysis.

**📌 Key Ratios Used in Pension House Investment Analysis**

These ratios are divided into **Liquidity, Profitability, Efficiency, Valuation, and Leverage** categories.

**1. Liquidity Ratios (Short-Term Stability)**

| **Ratio** | **Formula** | **Interpretation** |
| --- | --- | --- |
| **Current Ratio** | Current Assets / Current Liabilities | **> 1.5 is good** (indicates ability to cover short-term liabilities). **< 1 is risky**. |
| **Quick Ratio** | (Current Assets - Inventory) / Current Liabilities | **> 1 is good**, measures liquidity **excluding inventory** (more conservative). |
| **Cash Ratio** | Cash & Equivalents / Current Liabilities | Higher is better; indicates the company’s **pure cash position**. |

**2. Profitability Ratios (Earnings & Returns)**

| **Ratio** | **Formula** | **Interpretation** |
| --- | --- | --- |
| **Return on Equity (ROE)** | Net Income / Shareholders’ Equity | **> 15% is good** (higher return on shareholder investment). |
| **Return on Assets (ROA)** | Net Income / Total Assets | **> 5% is healthy**, shows how efficiently assets generate profit. |
| **Return on Capital Employed (ROCE)** | EBIT / (Total Assets - Current Liabilities) | **> 12% preferred**, measures efficiency in capital utilization. |
| **Net Profit Margin** | Net Profit / Revenue | **> 10% is strong**, measures final profitability per dollar of revenue. |

**3. Efficiency Ratios (Operational Effectiveness)**

| **Ratio** | **Formula** | **Interpretation** |
| --- | --- | --- |
| **Asset Turnover Ratio** | Revenue / Total Assets | **Higher is better**, measures how effectively assets generate sales. |
| **Inventory Turnover** | COGS / Inventory | **> 5 means fast-moving stock**, lower values indicate excess inventory. |

**4. Valuation Ratios (Stock Price Fairness)**

| **Ratio** | **Formula** | **Interpretation** |
| --- | --- | --- |
| **P/E Ratio (Price-to-Earnings)** | Stock Price / EPS | **< 20 is reasonable**, lower means undervaluation, higher suggests overvaluation. |
| **P/B Ratio (Price-to-Book)** | Stock Price / (Total Equity / Shares Outstanding) | **< 3 is good**, shows market price vs. book value. |
| **EV/EBITDA** | (Market Cap + Debt - Cash) / EBITDA | **< 12 is preferred**, a lower ratio means a cheaper valuation. |

**5. Leverage Ratios (Debt Risk)**

| **Ratio** | **Formula** | **Interpretation** |
| --- | --- | --- |
| **Debt-to-Equity Ratio** | Total Debt / Shareholders’ Equity | **< 1 is good**, higher values mean higher debt risk. |
| **Interest Coverage Ratio** | EBIT / Interest Expense | **> 3 is safe**, lower values mean high risk of default. |

**📌 Python Code for Automated Pension Investment Analysis**

The script includes **three functions**:

1. fetch\_financial\_data() → Downloads stock data, balance sheet, income statement, and cash flow.
2. calculate\_ratios() → Computes all ratios and returns a **Pandas DataFrame** sorted by financial year-end.
3. analyze\_with\_llm() → Passes the computed data to **GPT-4** to generate a **buy/hold/sell recommendation**.

**🔧 Install Required Libraries**

pip install yfinance openai pandas numpy

**📜 End-to-End Python Script**

import yfinance as yf  
import pandas as pd  
import numpy as np  
import openai  
  
*# Set OpenAI API Key*  
openai.api\_key = "your\_openai\_api\_key"  
  
def fetch\_financial\_data(ticker):  
 """Fetch stock financials from Yahoo Finance."""  
 stock = yf.Ticker(ticker)  
  
 *# Fetch Financial Statements*  
 income\_stmt = stock.financials.T *# Income Statement*  
 balance\_sheet = stock.balance\_sheet.T *# Balance Sheet*  
 cash\_flow = stock.cashflow.T *# Cash Flow Statement*  
  
 *# Ensure all dates are in ascending order*  
 income\_stmt.sort\_index(inplace=True)  
 balance\_sheet.sort\_index(inplace=True)  
 cash\_flow.sort\_index(inplace=True)  
  
 return income\_stmt, balance\_sheet, cash\_flow, stock  
  
  
def calculate\_ratios(income\_stmt, balance\_sheet, cash\_flow, stock):  
 """Calculate financial ratios and return DataFrame sorted by year-end."""  
 ratios\_df = pd.DataFrame(index=income\_stmt.index)  
 ratios\_df.index.name = "Year-End"  
  
 *# Liquidity Ratios*  
 ratios\_df["Current Ratio"] = balance\_sheet["Total Current Assets"] / balance\_sheet["Total Current Liabilities"]  
 ratios\_df["Quick Ratio"] = (balance\_sheet["Total Current Assets"] - balance\_sheet.get("Inventory", np.nan)) / balance\_sheet["Total Current Liabilities"]  
  
 *# Profitability Ratios*  
 ratios\_df["ROE (%)"] = (income\_stmt["Net Income"] / balance\_sheet["Total Stockholder Equity"]) \* 100  
 ratios\_df["ROA (%)"] = (income\_stmt["Net Income"] / balance\_sheet["Total Assets"]) \* 100  
 ratios\_df["Net Profit Margin (%)"] = (income\_stmt["Net Income"] / income\_stmt["Total Revenue"]) \* 100  
  
 *# Efficiency Ratios*  
 ratios\_df["Asset Turnover Ratio"] = income\_stmt["Total Revenue"] / balance\_sheet["Total Assets"]  
  
 *# Valuation Ratios*  
 market\_price = stock.history(period="5y")["Close"].resample("Y").last()  
 shares\_outstanding = balance\_sheet["Ordinary Shares Number"]  
   
 ratios\_df["EPS"] = income\_stmt["Net Income"] / shares\_outstanding  
 ratios\_df["P/E Ratio"] = market\_price / ratios\_df["EPS"]  
 ratios\_df["P/B Ratio"] = market\_price / (balance\_sheet["Total Stockholder Equity"] / shares\_outstanding)  
  
 *# Leverage Ratios*  
 ratios\_df["Debt-to-Equity"] = balance\_sheet["Total Debt"] / balance\_sheet["Total Stockholder Equity"]  
  
 *# Sort by Year-End (Ascending)*  
 ratios\_df.sort\_index(inplace=True)  
 return ratios\_df  
  
def analyze\_with\_llm(ratios\_df, ticker):  
 """Pass the financial ratios DataFrame to OpenAI for stock recommendation."""  
 prompt = f"""  
 You are a pension fund investment expert. Analyze the financial ratios for {ticker} and provide a recommendation (BUY, HOLD, SELL) along with reasons.  
  
 Data:  
 {ratios\_df.to\_string()}  
  
 Consider profitability, leverage, valuation, liquidity, and efficiency ratios.  
 Provide a professional, long-term-focused analysis for institutional investors.  
 """  
  
 response = openai.ChatCompletion.create(  
 model="gpt-4",  
 messages=[{"role": "system", "content": "You are an expert pension fund advisor."},  
 {"role": "user", "content": prompt}]  
 )  
  
 return response["choices"][0]["message"]["content"]  
  
*# Example Usage*  
ticker = "RELIANCE.NS" *# Replace with any stock symbol*  
income\_stmt, balance\_sheet, cash\_flow, stock = fetch\_financial\_data(ticker)  
ratios\_df = calculate\_ratios(income\_stmt, balance\_sheet, cash\_flow, stock)  
  
import ace\_tools as tools  
tools.display\_dataframe\_to\_user(name="Pension Fund Stock Analysis", dataframe=ratios\_df)  
  
*# Get LLM recommendation*  
llm\_recommendation = analyze\_with\_llm(ratios\_df, ticker)  
print(llm\_recommendation)