**Detailed Breakdown of the "Financial Metrics Dashboard with Heatmap" Code**

This Pine Script code creates a TradingView indicator called "Financial Metrics Dashboard with Heatmap." The script fetches financial and technical data for multiple stocks, processes the information, and displays it in a formatted table with a heatmap to visually highlight the data. Here’s a breakdown of each part of the code, explaining what it does and why it’s written this way.

**1. Indicator Declaration**

pinescript

Copy code

//@version=5

//@Yaseen Khalil

indicator("Financial Metrics Dashboard with Heatmap", overlay=false)

* **//@version=5**: Specifies the use of Pine Script version 5, the latest version with enhanced features.
* **indicator("Financial Metrics Dashboard with Heatmap", overlay=false)**: Defines the indicator's name and sets overlay=false so that the indicator appears in its own pane rather than over the price chart.

**2. Custom Type Definition**

pinescript

Copy code

type FinancialData

float marketCap

float price

float eps

float peRatio

float psRatio

float pbRatio

float profitMargin

float debtEquity

float currentRatio

float ma1

float ma2

float volume

float avgVolume

string currency

* **Purpose**: Defines a custom type called FinancialData to store multiple related financial metrics for each stock. This approach makes the data organized and accessible as a single object, simplifying the process of storing and displaying financial information for each symbol.

**3. User Inputs**

pinescript

Copy code

primarySymbol = input.symbol('NASDAQ:AAPL', 'Primary Symbol')

secondarySymbol = input.symbol('NASDAQ:GOOGL', 'Secondary Symbol')

tertiarySymbol = input.symbol('NASDAQ:AMZN', 'Tertiary Symbol')

quaternarySymbol = input.symbol('NASDAQ:TSLA', 'Quaternary Symbol')

quinarySymbol = input.symbol('NASDAQ:MSFT', 'Quinary Symbol')

* **Purpose**: Allows users to select up to five symbols for analysis. The default symbols are popular U.S. tech stocks like Apple, Google, Amazon, Tesla, and Microsoft.

pinescript

Copy code

tableLocationY = input.string('middle', title='Table Vertical Position', options=['top', 'middle', 'bottom'])

tableLocationX = input.string('center', title='Table Horizontal Position', options=['left', 'center', 'right'])

fontSize = input.string(size.small, title='Font Size', options=[size.tiny, size.small, size.normal, size.large, size.huge])

avgVolumePeriod = input.int(30, title='Average Volume Period', minval=5, maxval=100)

maType1 = input.string('EMA', title='MA Type 1', options=['SMA', 'EMA'], inline='MA1')

maLength1 = input.int(20, title='', minval=5, inline='MA1')

maType2 = input.string('EMA', title='MA Type 2', options=['SMA', 'EMA'], inline='MA2')

maLength2 = input.int(50, title='', minval=5, inline='MA2')

* **Purpose**: These inputs control the table's position, font size, and settings for technical indicators like moving averages (MA). The average volume period setting defines how many days are considered when calculating average volume.

**4. Utility Functions**

**Moving Average Function**

pinescript

Copy code

moving\_average(source, length, ma\_type) =>

ma\_type == "SMA" ? ta.sma(source, length) : ta.ema(source, length)

* **Purpose**: This function calculates the moving average of a data series (source) based on the user-selected type (SMA or EMA) and length. SMA (Simple Moving Average) takes the average over the specified period, while EMA (Exponential Moving Average) gives more weight to recent data.

**Format Large Number Function**

pinescript

Copy code

format\_large\_number(value) =>

abs\_value = math.abs(value)

if abs\_value >= 1e9

str.tostring(value / 1e9, "#.##B")

else if abs\_value >= 1e6

str.tostring(value / 1e6, "#.##M")

else if abs\_value >= 1e3

str.tostring(value / 1e3, "#.##K")

else

str.tostring(value, "#.##")

* **Purpose**: Converts large numbers into readable formats with suffixes (B for billions, M for millions, K for thousands). This helps present financial data like market cap and volume more concisely.

**5. Fetching Financial Data**

**Fetch Financial Data Function**

pinescript

Copy code

fetch\_financial\_data(symbol) =>

[price, ma1, ma2, vol, avgVol, currency] = request.security(symbol, timeframe.period, [close,moving\_average(close, maLength1, maType1),moving\_average(close, maLength2, maType2),volume,ta.sma(volume, avgVolumePeriod),syminfo.currency])

eps = request.financial(symbol, "EARNINGS\_PER\_SHARE", "TTM")

peRatio = price / eps

marketCap = request.financial(symbol, "TOTAL\_SHARES\_OUTSTANDING", "FQ") \* price

psRatio = marketCap / request.financial(symbol, "TOTAL\_REVENUE", "TTM")

pbRatio = price / request.financial(symbol, "BOOK\_VALUE\_PER\_SHARE", "FQ")

profitMargin = 100 \* request.financial(symbol, "NET\_INCOME", "TTM") / request.financial(symbol, "TOTAL\_REVENUE", "TTM")

debtEquity = request.financial(symbol, "DEBT\_TO\_EQUITY", "FQ")

currentRatio = request.financial(symbol, "CURRENT\_RATIO", "FQ")

FinancialData.new(marketCap, price, eps, peRatio, psRatio, pbRatio, profitMargin, debtEquity, currentRatio, ma1, ma2, vol, avgVol, currency)

* **Purpose**: This function fetches various technical and financial data points for a given stock symbol. It uses the request.security function to pull data like closing price, moving averages, volume, and currency.
* **Calculations**:
  + **peRatio**: Price divided by earnings per share (EPS), showing how much investors are willing to pay for each dollar of earnings.
  + **marketCap**: Total market value of a company's outstanding shares.
  + **psRatio**: Price-to-sales ratio, calculated by dividing market cap by total revenue. A lower ratio often indicates a better value.
  + **pbRatio**: Price-to-book ratio, which compares the price of a stock to its book value. A ratio below 1 can suggest the stock is undervalued.
  + **profitMargin**: Percentage of revenue that turns into profit, indicating how efficiently a company operates.
  + **debtEquity**: Measures a company’s financial leverage by comparing its total liabilities to its shareholder equity.
  + **currentRatio**: A liquidity ratio that measures a company's ability to pay short-term obligations.

**6. Initialize Data Arrays and Table**

pinescript

Copy code

financialData = array.new<FinancialData>(5)

array.set(financialData, 0, fetch\_financial\_data(primarySymbol))

array.set(financialData, 1, fetch\_financial\_data(secondarySymbol))

array.set(financialData, 2, fetch\_financial\_data(tertiarySymbol))

array.set(financialData, 3, fetch\_financial\_data(quaternarySymbol))

array.set(financialData, 4, fetch\_financial\_data(quinarySymbol))

* **Purpose**: Creates an array of FinancialData objects to store the fetched data for each of the five stocks selected by the user.

pinescript

Copy code

var dashboardTable = table.new(position = tableLocationY + '\_' + tableLocationX, columns = 15, rows = 6, frame\_width = 1, frame\_color = color.teal, border\_width = 1, border\_color = color.blue)

* **Purpose**: Initializes a table with 15 columns and 6 rows to display the financial metrics. The table position and style are defined by the user inputs.

**7. Display Data in the Table**

**Headers and Data Rows**

pinescript

Copy code

if barstate.islast

headers = array.new\_string(15)

array.set(headers, 0, "#")

array.set(headers, 1, "Stock")

array.set(headers, 2, "Price")

array.set(headers, 3, "Mkt Cap")

array.set(headers, 4, "EPS")

array.set(headers, 5, "P/E")

array.set(headers, 6, "P/S")

array.set(headers, 7, "P/B")

array.set(headers, 8, "Prof Mgn")

array.set(headers, 9, "Debt/Equity")

array.set(headers, 10, "Current Ratio")

array.set(headers, 11, maType1 + str.tostring(maLength1))

array.set(headers, 12, maType2 + str.tostring(maLength2))

array.set(headers, 13, "Volume")

array.set(headers, 14, "Avg Vol")

// Set header row

for col = 0 to 14

table.cell(dashboardTable, col, 0, text = array.get(headers, col), text\_color = color.white, bgcolor = #1a5d93, text\_size = fontSize)

// Populate Table Rows

for i = 0 to 4

data = array.get(financialData, i)

symbolText = array.get(symbolArray, i)

table.cell(dashboardTable, 0, i + 1, text = str.tostring(i + 1), text\_color = color.black, bgcolor = color.rgb(180, 180, 255), text\_size = fontSize)

table.cell(dashboardTable, 1, i + 1, text = symbolText, text\_color = color.black, bgcolor = color.rgb(180, 180, 255), text\_size = fontSize)

...

* **Purpose**: Adds headers to the table and populates each row with the respective data from the financialData array. The table's cells are color-coded using heatmap logic to provide a visual representation of the data's performance, like higher values being shown in green and lower or less favorable values shown in red.

**8. Heatmap Function**

pinescript

Copy code

heatmap\_color(value, condition) =>

condition ? color.green : color.red

* **Purpose**: Determines the cell color based on a condition. This is used for visual cues; for example, a green cell might indicate a positive metric like a high profit margin or low debt-to-equity ratio, while a red cell indicates a less favorable metric.