```
// This Pine Script® code is subject to the terms of the Mozilla Public License 2.0 at https://mozilla.org/MPL/2.0/ MPL-2.0

//@version=6

indicator("MFB - VRS 5-Day HVN Zones", overlay = true, max_bars_back = 5000, max_boxes_count = 300, max_lines_count = 300, max_labels_count = 300)


// --- Constants ---

const int PROFILE_DAYS = 5

const string DEFAULT_SESSION = "0930-1600"

const string DEFAULT_TIMEZONE = "America/New_York"

const color COLOR_UP = #089981

const color COLOR_DOWN = #f23645

const color COLOR_POC = #b24cff

const color COLOR_VA = #7e57c2

const color COLOR_MID = #f2c744


// --- Inputs ---

string sessionInput = input.session(DEFAULT_SESSION, "Structural Session", tooltip = "Session used to build each completed daily volume profile.")

string timezoneInput = input.string(DEFAULT_TIMEZONE, "Timezone", tooltip = "Timezone used for session detection.")

int rowCountInput = input.int(50, "Profile Rows", minval = 10, maxval = 100, tooltip = "Price bins used to estimate each completed session profile.")
```

```
float nodeThresholdInput = input.float(65.0, "Daily HVN Threshold (% of POC row)", minval = 20.0, maxval = 95.0, step = 5.0, tooltip = "Threshold used when calculating each daily session's local volume nodes.")

float compositePeakThresholdInput = input.float(25.0, "5-Day Peak Threshold (% of POC row)", minval = 5.0, maxval = 90.0, step = 5.0, group = "Composite HVN", tooltip = "Only local peaks at or above this share of the five-day composite POC row are eligible; the three strongest distinct peaks are drawn.")

float valueAreaPercentInput = input.float(70.0, "Value Area Percentage", minval = 50.0, maxval = 90.0, step = 1.0, tooltip = "Volume percentage used to calculate each session's VA midpoint.") * 0.01

bool showLabelsInput = input.bool(true, "Show Composite Labels", tooltip = "Label the five-day composite VAH, VAL, POC, and selected HVN peak lines.")

bool showCompositeLevelsInput = input.bool(true, "Show 5-Day VAH / VAL / POC", tooltip = "Draw only the pooled five-session composite VAH, VAL, and POC reference lines.")

color compositePocColorInput = input.color(#b0b0b0, "Composite POC Color", group = "Style", tooltip = "Neutral gray color for the pooled five-session point of control.")

color compositeVaColorInput = input.color(#b0b0b0, "Composite VAH / VAL Color", group = "Style", tooltip = "Neutral gray color for the pooled five-session value-area boundaries.")

int compositePocWidthInput = input.int(3, "Composite POC Line Thickness", minval = 1, maxval = 4, group = "Style", tooltip = "Line thickness for the five-day composite point of control.")

int compositeVaWidthInput = input.int(2, "Composite VAH / VAL Line Thickness", minval = 1, maxval = 4, group = "Style", tooltip = "Line thickness for the five-day composite value-area boundaries.")
```

```
int compositeHvnWidthInput = input.int(2, "Composite HVN Peak Line Thickness", minval = 1, maxval = 4, group = "Style", tooltip = "Line thickness for the selected five-day composite HVN peak lines.")

bool showDirectionalTableInput = input.bool(true, "Show 5-Day Direction Table", tooltip = "Keep the five-session summary of POC, VA-mid, and up/down volume migration.")

int fillTransparencyInput = input.int(86, "Zone Fill Transparency", minval = 0, maxval = 100, group = "Style", tooltip = "Base transparency for the newest session's HVN fill; older sessions fade further.")

int borderTransparencyInput = input.int(25, "Zone Border Transparency", minval = 0, maxval = 100, group = "Style", tooltip = "Base transparency for the newest session's HVN border; older sessions fade further.")


// --- Data Type ---

type SessionProfile

    int startBar

    int endBar

    float node1Top

    float node1Bottom

    float node1Volume

    float node2Top

    float node2Bottom

    float node2Volume

    float pocPrice

    float vaMidPrice

    float upVolume
```

```
    float downVolume
    bool valid
    float vahPrice = na
    float valPrice = na
    float sessionHigh = na
    float sessionLow = na
    float[] sourcePrices = na
    float[] sourceVolumes = na

// --- Profile Builder ---
f_buildProfile(float[] prices, float[] volumes, int startBar, int endBar, float sessionHigh, float sessionLow, int rows, float nodeThreshold, float valueAreaPercent, float upVolume, float downVolume) =>
    float priceRange = sessionHigh - sessionLow
    float node1Top = na
    float node1Bottom = na
    float node1Volume = 0.0
    float node2Top = na
    float node2Bottom = na
    float node2Volume = 0.0
    float pocPrice = na
    float vaMidPrice = na
    float vahPrice = na
    float valPrice = na
```

```
    bool profileValid = false

    if priceRange > 0 and array.size(prices) > 0
        float rowStep = priceRange / rows
        float[] rowVolumes = array.new_float(rows, 0.0)
        float totalVolume = 0.0
        for i = 0 to array.size(prices) - 1
            float price = array.get(prices, i)
            float volumeValue = nz(array.get(volumes, i), 0.0)
            int rowIndex = int(math.floor((price - sessionLow) / rowStep))
            rowIndex := math.max(0, math.min(rows - 1, rowIndex))
            array.set(rowVolumes, rowIndex, array.get(rowVolumes, rowIndex) +
volumeValue)
            totalVolume += volumeValue

        float maxRowVolume = array.get(rowVolumes, 0)
        int pocIndex = 0
        for rowSearch = 1 to rows - 1
            float candidateVolume = array.get(rowVolumes, rowSearch)
            if candidateVolume > maxRowVolume
                maxRowVolume := candidateVolume
                pocIndex := rowSearch
```

```
    if maxRowVolume > 0 and totalVolume > 0
        int lowerVaIndex = pocIndex
        int upperVaIndex = pocIndex
        float accumulatedVolume = maxRowVolume
        float targetVolume = totalVolume * valueAreaPercent
        int expansionSteps = 0
        while accumulatedVolume < targetVolume and (upperVaIndex < rows - 1 or lowerVaIndex > 0) and expansionSteps < rows
            expansionSteps += 1
            bool canExpandUp = upperVaIndex < rows - 1
            bool canExpandDown = lowerVaIndex > 0
            float upperVolume = canExpandUp ? array.get(rowVolumes, upperVaIndex + 1) : 0.0
            float lowerVolume = canExpandDown ? array.get(rowVolumes, lowerVaIndex - 1) : 0.0
            if canExpandUp and (not canExpandDown or upperVolume >= lowerVolume)
                upperVaIndex += 1
                accumulatedVolume += upperVolume
            else if canExpandDown
                lowerVaIndex -= 1
                accumulatedVolume += lowerVolume

        valPrice := sessionLow + lowerVaIndex * rowStep
```

```
    vahPrice := sessionLow + (upperVaIndex + 1) * rowStep
    pocPrice := sessionLow + (pocIndex + 0.5) * rowStep
    vaMidPrice := (valPrice + vahPrice) * 0.5


    float minimumNodeVolume = maxRowVolume * nodeThreshold
    int minimumPeakSpacing = math.max(2, int(math.round(rows * 0.04)))
    int[] localPeakIndices = array.new_int()
    float[] localPeakVolumes = array.new_float()
    for peakIndex = 0 to rows - 1
      float peakVolume = array.get(rowVolumes, peakIndex)
      float previousPeakNeighbor = peakIndex > 0 ? array.get(rowVolumes,
peakIndex - 1) : -1.0
      float nextPeakNeighbor = peakIndex < rows - 1 ?
array.get(rowVolumes, peakIndex + 1) : -1.0
      bool isLocalPeak = peakVolume >= minimumNodeVolume and
peakVolume >= previousPeakNeighbor and peakVolume >= nextPeakNeighbor
and (peakVolume > previousPeakNeighbor or peakVolume >
nextPeakNeighbor)
      if isLocalPeak
        array.push(localPeakIndices, peakIndex)
        array.push(localPeakVolumes, peakVolume)


    int strongestPeakIndex = na
    float strongestPeakVolume = 0.0
    int secondPeakIndex = na
```

```
    float secondPeakVolume = 0.0
    if array.size(localPeakIndices) > 0
      for candidateIndex = 0 to array.size(localPeakIndices) - 1
        float candidateVolume = array.get(localPeakVolumes,
candidateIndex)
        if candidateVolume > strongestPeakVolume
          strongestPeakVolume := candidateVolume
          strongestPeakIndex := array.get(localPeakIndices, candidateIndex)
      for candidateIndex = 0 to array.size(localPeakIndices) - 1
        int candidateRow = array.get(localPeakIndices, candidateIndex)
        float candidateVolume = array.get(localPeakVolumes,
candidateIndex)
        if math.abs(candidateRow - strongestPeakIndex) >=
minimumPeakSpacing and candidateVolume > secondPeakVolume
          secondPeakVolume := candidateVolume
          secondPeakIndex := candidateRow

    if not na(strongestPeakIndex)
      node1Bottom := sessionLow + strongestPeakIndex * rowStep
      node1Top := sessionLow + (strongestPeakIndex + 1) * rowStep
      node1Volume := strongestPeakVolume
    if not na(secondPeakIndex)
      node2Bottom := sessionLow + secondPeakIndex * rowStep
      node2Top := sessionLow + (secondPeakIndex + 1) * rowStep
```

```
            node2Volume := secondPeakVolume
        profileValid := node1Volume > 0

    SessionProfile.new(startBar, endBar, node1Top, node1Bottom,
node1Volume, node2Top, node2Bottom, node2Volume, pocPrice, vaMidPrice,
upVolume, downVolume, profileValid, vahPrice, valPrice, sessionHigh,
sessionLow, array.copy(prices), array.copy(volumes))

f_clearDrawings(box[] boxes, label[] labels) =>
    for drawingBox in boxes
        box.delete(drawingBox)
    array.clear(boxes)
    for drawingLabel in labels
        label.delete(drawingLabel)
    array.clear(labels)

f_dayColor(int dayIndex) =>
    dayIndex == 0 ? COLOR_DOWN : dayIndex == 1 ? COLOR_MID : dayIndex ==
2 ? #5b9cf6 : dayIndex == 3 ? COLOR_UP : COLOR_POC

f_priceTouched(float priceLevel) =>
    not na(priceLevel) and low <= priceLevel and high >= priceLevel

// --- Session Collection ---
```

```
int sessionTime = time(timeframe.period, sessionInput + ":23456",
timezoneInput)

bool inSession = not na(sessionTime)

bool isNewSession = inSession and not inSession[1]

bool isSessionEnd = not inSession and inSession[1]

var float[] currentPrices = array.new_float()

var float[] currentVolumes = array.new_float()

var int currentStartBar = na

var float currentHigh = na

var float currentLow = na

var float currentUpVolume = 0.0

var float currentDownVolume = 0.0

var SessionProfile[] completedProfiles = array.new<SessionProfile>()

var float compositePocPrice = na

var float compositeVahPrice = na

var float compositeValPrice = na

var float compositePeak1Price = na

var float compositePeak2Price = na

var float compositePeak3Price = na

var bool compositeValid = false


if isNewSession

   array.clear(currentPrices)
```

```
    array.clear(currentVolumes)
    currentStartBar := bar_index
    currentHigh := high
    currentLow := low
    currentUpVolume := 0.0
    currentDownVolume := 0.0

if inSession
    currentHigh := math.max(nz(currentHigh, high), high)
    currentLow := math.min(nz(currentLow, low), low)
    array.push(currentPrices, hlc3)
    array.push(currentVolumes, nz(volume, 0.0))
    if close >= open
        currentUpVolume += nz(volume, 0.0)
    else
        currentDownVolume += nz(volume, 0.0)

if isSessionEnd and not na(currentStartBar) and not na(currentHigh) and not
na(currentLow)
    SessionProfile finishedProfile = f_buildProfile(currentPrices,
currentVolumes, currentStartBar, bar_index - 1, currentHigh, currentLow,
rowCountInput, nodeThresholdInput * 0.01, valueAreaPercentInput,
currentUpVolume, currentDownVolume)
    if finishedProfile.valid
```

```
    array.unshift(completedProfiles, finishedProfile)
    if array.size(completedProfiles) > PROFILE_DAYS
        array.pop(completedProfiles)

    compositeValid := false
    compositePocPrice := na
    compositeVahPrice := na
    compositeValPrice := na
    compositePeak1Price := na
    compositePeak2Price := na
    compositePeak3Price := na
    if array.size(completedProfiles) == PROFILE_DAYS
        float compositeHigh = na
        float compositeLow = na
        for rangeIndex = 0 to PROFILE_DAYS - 1
            SessionProfile rangeProfile = array.get(completedProfiles, rangeIndex)
            compositeHigh := na(compositeHigh) ? rangeProfile.sessionHigh :
math.max(compositeHigh, rangeProfile.sessionHigh)
            compositeLow := na(compositeLow) ? rangeProfile.sessionLow :
math.min(compositeLow, rangeProfile.sessionLow)

        float compositeRange = compositeHigh - compositeLow
        if compositeRange > 0
            float compositeStep = compositeRange / rowCountInput
```

```
    float[] compositeRows = array.new_float(rowCountInput, 0.0)
    float compositeTotalVolume = 0.0
    for profileIndex = 0 to PROFILE_DAYS - 1
      SessionProfile compositeProfile = array.get(completedProfiles, profileIndex)
      for sourceIndex = 0 to array.size(compositeProfile.sourcePrices) - 1
        float sourcePrice = array.get(compositeProfile.sourcePrices, sourceIndex)
        float sourceVolume = nz(array.get(compositeProfile.sourceVolumes, sourceIndex), 0.0)
        int compositeRow = int(math.floor((sourcePrice - compositeLow) / compositeStep))
        compositeRow := math.max(0, math.min(rowCountInput - 1, compositeRow))
        array.set(compositeRows, compositeRow, array.get(compositeRows, compositeRow) + sourceVolume)
        compositeTotalVolume += sourceVolume

    float compositeMaxVolume = array.get(compositeRows, 0)
    int compositePocIndex = 0
    for rowSearch = 1 to rowCountInput - 1
      float rowVolume = array.get(compositeRows, rowSearch)
      if rowVolume > compositeMaxVolume
        compositeMaxVolume := rowVolume
        compositePocIndex := rowSearch
```

```
        if compositeMaxVolume > 0 and compositeTotalVolume > 0
            int compositeLowerIndex = compositePocIndex
            int compositeUpperIndex = compositePocIndex
            float compositeValueVolume = compositeMaxVolume
            float compositeTargetVolume = compositeTotalVolume *
valueAreaPercentInput
            int compositeExpansionSteps = 0
            while compositeValueVolume < compositeTargetVolume and
(compositeUpperIndex < rowCountInput - 1 or compositeLowerIndex > 0) and
compositeExpansionSteps < rowCountInput
                compositeExpansionSteps += 1
                bool canExpandCompositeUp = compositeUpperIndex <
rowCountInput - 1
                bool canExpandCompositeDown = compositeLowerIndex > 0
                float compositeUpperVolume = canExpandCompositeUp ?
array.get(compositeRows, compositeUpperIndex + 1) : 0.0
                float compositeLowerVolume = canExpandCompositeDown ?
array.get(compositeRows, compositeLowerIndex - 1) : 0.0
                if canExpandCompositeUp and (not canExpandCompositeDown
or compositeUpperVolume >= compositeLowerVolume)
                    compositeUpperIndex += 1
                    compositeValueVolume += compositeUpperVolume
                else if canExpandCompositeDown
                    compositeLowerIndex -= 1
```

```
compositeValueVolume += compositeLowerVolume


        compositePocPrice := compositeLow + (compositePocIndex + 0.5) *
compositeStep

        compositeValPrice := compositeLow + compositeLowerIndex *
compositeStep

        compositeVahPrice := compositeLow + (compositeUpperIndex + 1) *
compositeStep


        float minimumCompositePeakVolume = compositeMaxVolume *
compositePeakThresholdInput * 0.01

        int minimumCompositePeakSpacing = math.max(2,
int(math.round(rowCountInput * 0.06)))

        int strongestPeakRow = -1

        float strongestPeakVolume = 0.0

        int secondPeakRow = -1

        float secondPeakVolume = 0.0

        int thirdPeakRow = -1

        float thirdPeakVolume = 0.0


        for peakCandidate1 = 0 to rowCountInput - 1

          float candidateVolume1 = array.get(compositeRows,
peakCandidate1)

          float candidateLeft1 = peakCandidate1 > 0 ?
array.get(compositeRows, peakCandidate1 - 1) : -1.0
```

```
            float candidateRight1 = peakCandidate1 < rowCountInput - 1 ?
array.get(compositeRows, peakCandidate1 + 1) : -1.0

            bool isLocalPeak1 = candidateVolume1 >=
minimumCompositePeakVolume and candidateVolume1 >= candidateLeft1
and candidateVolume1 >= candidateRight1 and (candidateVolume1 >
candidateLeft1 or candidateVolume1 > candidateRight1)

            if isLocalPeak1 and candidateVolume1 > strongestPeakVolume

              strongestPeakVolume := candidateVolume1

              strongestPeakRow := peakCandidate1


          for peakCandidate2 = 0 to rowCountInput - 1

            float candidateVolume2 = array.get(compositeRows,
peakCandidate2)

            float candidateLeft2 = peakCandidate2 > 0 ?
array.get(compositeRows, peakCandidate2 - 1) : -1.0

            float candidateRight2 = peakCandidate2 < rowCountInput - 1 ?
array.get(compositeRows, peakCandidate2 + 1) : -1.0

            bool isLocalPeak2 = candidateVolume2 >=
minimumCompositePeakVolume and candidateVolume2 >= candidateLeft2
and candidateVolume2 >= candidateRight2 and (candidateVolume2 >
candidateLeft2 or candidateVolume2 > candidateRight2)

            bool spacedFromFirst = strongestPeakRow < 0 or
math.abs(peakCandidate2 - strongestPeakRow) >=
minimumCompositePeakSpacing

            if isLocalPeak2 and spacedFromFirst and candidateVolume2 >
secondPeakVolume

              secondPeakVolume := candidateVolume2
```

```
            secondPeakRow := peakCandidate2


        for peakCandidate3 = 0 to rowCountInput - 1
            float candidateVolume3 = array.get(compositeRows,
peakCandidate3)
            float candidateLeft3 = peakCandidate3 > 0 ?
array.get(compositeRows, peakCandidate3 - 1) : -1.0
            float candidateRight3 = peakCandidate3 < rowCountInput - 1 ?
array.get(compositeRows, peakCandidate3 + 1) : -1.0
            bool isLocalPeak3 = candidateVolume3 >=
minimumCompositePeakVolume and candidateVolume3 >= candidateLeft3
and candidateVolume3 >= candidateRight3 and (candidateVolume3 >
candidateLeft3 or candidateVolume3 > candidateRight3)
            bool spacedFromFirst3 = strongestPeakRow < 0 or
math.abs(peakCandidate3 - strongestPeakRow) >=
minimumCompositePeakSpacing
            bool spacedFromSecond3 = secondPeakRow < 0 or
math.abs(peakCandidate3 - secondPeakRow) >=
minimumCompositePeakSpacing
            if isLocalPeak3 and spacedFromFirst3 and spacedFromSecond3
and candidateVolume3 > thirdPeakVolume
                thirdPeakVolume := candidateVolume3
                thirdPeakRow := peakCandidate3


        if strongestPeakRow >= 0
            compositePeak1Price := compositeLow + (strongestPeakRow +
0.5) * compositeStep
```

```
            if secondPeakRow >= 0
                compositePeak2Price := compositeLow + (secondPeakRow + 0.5)
* compositeStep
            if thirdPeakRow >= 0
                compositePeak3Price := compositeLow + (thirdPeakRow + 0.5) *
compositeStep
            compositeValid := true
    currentStartBar := na
    currentHigh := na
    currentLow := na

// --- HVN Rendering ---
var box[] nodeBoxes = array.new_box()
var label[] nodeLabels = array.new_label()
var line[] levelLines = array.new_line()

if barstate.islast
    f_clearDrawings(nodeBoxes, nodeLabels)
    for levelLine in levelLines
        line.delete(levelLine)
    array.clear(levelLines)
    if showCompositeLevelsInput and compositeValid and
array.size(completedProfiles) == PROFILE_DAYS
        SessionProfile newestCompleted = array.get(completedProfiles, 0)
```

```
    int compositeStartBar = newestCompleted.endBar

    line compositePocLine = line.new(compositeStartBar,
compositePocPrice, bar_index + 1, compositePocPrice, xloc = xloc.bar_index,
extend = extend.right, color = compositePocColorInput, width =
compositePocWidthInput)

    line compositeVahLine = line.new(compositeStartBar,
compositeVahPrice, bar_index + 1, compositeVahPrice, xloc = xloc.bar_index,
extend = extend.right, color = compositeVaColorInput, style =
line.style_dashed, width = compositeVaWidthInput)

    line compositeValLine = line.new(compositeStartBar, compositeValPrice,
bar_index + 1, compositeValPrice, xloc = xloc.bar_index, extend = extend.right,
color = compositeVaColorInput, style = line.style_dashed, width =
compositeVaWidthInput)

    array.push(levelLines, compositePocLine)

    array.push(levelLines, compositeVahLine)

    array.push(levelLines, compositeValLine)

    color compositePeakColor = color.new(#b0b0b0, 15)

    if not na(compositePeak1Price)

        line compositePeak1Line = line.new(compositeStartBar,
compositePeak1Price, bar_index + 1, compositePeak1Price, xloc =
xloc.bar_index, extend = extend.right, color = compositePeakColor, width =
compositeHvnWidthInput)

        array.push(levelLines, compositePeak1Line)

    if not na(compositePeak2Price)

        line compositePeak2Line = line.new(compositeStartBar,
compositePeak2Price, bar_index + 1, compositePeak2Price, xloc =
xloc.bar_index, extend = extend.right, color = compositePeakColor, width =
compositeHvnWidthInput)
```

```
        array.push(levelLines, compositePeak2Line)
    if not na(compositePeak3Price)
        line compositePeak3Line = line.new(compositeStartBar,
compositePeak3Price, bar_index + 1, compositePeak3Price, xloc =
xloc.bar_index, extend = extend.right, color = compositePeakColor, width =
compositeHvnWidthInput)
        array.push(levelLines, compositePeak3Line)
    if showLabelsInput
        int compositeLabelX = bar_index + 2
        label compositePocLabel = label.new(compositeLabelX,
compositePocPrice, "5D COMP POC", xloc = xloc.bar_index, yloc = yloc.price,
color = #00000000, textcolor = compositePocColorInput, style =
label.style_label_left, size = size.small)
        label compositeVahLabel = label.new(compositeLabelX,
compositeVahPrice, "5D COMP VAH", xloc = xloc.bar_index, yloc = yloc.price,
color = #00000000, textcolor = compositeVaColorInput, style =
label.style_label_left, size = size.small)
        label compositeValLabel = label.new(compositeLabelX,
compositeValPrice, "5D COMP VAL", xloc = xloc.bar_index, yloc = yloc.price,
color = #00000000, textcolor = compositeVaColorInput, style =
label.style_label_left, size = size.small)
        array.push(nodeLabels, compositePocLabel)
        array.push(nodeLabels, compositeVahLabel)
        array.push(nodeLabels, compositeValLabel)
        if not na(compositePeak1Price)
          label compositePeak1Label = label.new(compositeLabelX,
compositePeak1Price, "5D HVN Peak 1", xloc = xloc.bar_index, yloc =
```

```
yloc.price, color = #00000000, textcolor = compositePeakColor, style =
label.style_label_left, size = size.tiny)

        array.push(nodeLabels, compositePeak1Label)

    if not na(compositePeak2Price)

        label compositePeak2Label = label.new(compositeLabelX,
compositePeak2Price, "5D HVN Peak 2", xloc = xloc.bar_index, yloc =
yloc.price, color = #00000000, textcolor = compositePeakColor, style =
label.style_label_left, size = size.tiny)

        array.push(nodeLabels, compositePeak2Label)

    if not na(compositePeak3Price)

        label compositePeak3Label = label.new(compositeLabelX,
compositePeak3Price, "5D HVN Peak 3", xloc = xloc.bar_index, yloc =
yloc.price, color = #00000000, textcolor = compositePeakColor, style =
label.style_label_left, size = size.tiny)

        array.push(nodeLabels, compositePeak3Label)


// --- Five-Day Direction Table ---

var table directionTable = table.new(position.bottom_right, 1, 7, bgcolor =
color.new(chart.bg_color, 5), frame_color = COLOR_MID, frame_width = 2,
border_color = color.new(chart.fg_color, 70), border_width = 1)


if barstate.islast

  if showDirectionalTableInput

    int profileCount = array.size(completedProfiles)

    string statusText = "BUILDING DATA"
```

```
    string scoreText = "Completed profiles: " + str.tostring(profileCount) + "/" +
str.tostring(PROFILE_DAYS)
    string pocText = "POC migration: waiting"
    string vaText = "VA Mid migration: waiting"
    string volumeText = "Histogram volume: waiting"
    color statusColor = COLOR_MID

    if profileCount >= PROFILE_DAYS
      int bullishScore = 0
      int bearishScore = 0
      SessionProfile newestProfile = array.get(completedProfiles, 0)
      SessionProfile oldestProfile = array.get(completedProfiles,
PROFILE_DAYS - 1)
      if newestProfile.pocPrice > oldestProfile.pocPrice
        bullishScore += 2
      else if newestProfile.pocPrice < oldestProfile.pocPrice
        bearishScore += 2
      if newestProfile.vaMidPrice > oldestProfile.vaMidPrice
        bullishScore += 2
      else if newestProfile.vaMidPrice < oldestProfile.vaMidPrice
        bearishScore += 2

      for transitionIndex = 0 to PROFILE_DAYS - 2
```

```
        SessionProfile newerProfile = array.get(completedProfiles,
transitionIndex)
        SessionProfile olderProfile = array.get(completedProfiles,
transitionIndex + 1)
        if newerProfile.pocPrice > olderProfile.pocPrice
          bullishScore += 1
        else if newerProfile.pocPrice < olderProfile.pocPrice
          bearishScore += 1
        if newerProfile.vaMidPrice > olderProfile.vaMidPrice
          bullishScore += 1
        else if newerProfile.vaMidPrice < olderProfile.vaMidPrice
          bearishScore += 1


    float aggregateUpVolume = 0.0
    float aggregateDownVolume = 0.0
    for profileIndex = 0 to PROFILE_DAYS - 1
      SessionProfile profileForScore = array.get(completedProfiles,
profileIndex)
      aggregateUpVolume += profileForScore.upVolume
      aggregateDownVolume += profileForScore.downVolume
    if aggregateUpVolume > aggregateDownVolume
      bullishScore += 2
    else if aggregateDownVolume > aggregateUpVolume
      bearishScore += 2
```

```
int netScore = bullishScore - bearishScore

statusText := netScore >= 3 ? "BULLISH PROFILE TREND" : netScore <= -3 ? "BEARISH PROFILE TREND" : "MIXED / BALANCED PROFILE"

statusColor := netScore >= 3 ? COLOR_UP : netScore <= -3 ? COLOR_DOWN : COLOR_MID

scoreText := "Score: " + str.tostring(bullishScore) + " bullish / " + str.tostring(bearishScore) + " bearish"

pocText := newestProfile.pocPrice > oldestProfile.pocPrice ? "POC migration: rising" : newestProfile.pocPrice < oldestProfile.pocPrice ? "POC migration: falling" : "POC migration: flat"

vaText := newestProfile.vaMidPrice > oldestProfile.vaMidPrice ? "VA Mid migration: rising" : newestProfile.vaMidPrice < oldestProfile.vaMidPrice ? "VA Mid migration: falling" : "VA Mid migration: flat"

volumeText := aggregateUpVolume > aggregateDownVolume ? "Histogram volume: bullish" : aggregateDownVolume > aggregateUpVolume ? "Histogram volume: bearish" : "Histogram volume: balanced"


color cellBackground = color.new(chart.bg_color, 0)

table.cell(directionTable, 0, 0, "MFB - VRS - 5 DAYS", text_color = COLOR_MID, text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)

table.cell(directionTable, 0, 1, statusText, text_color = statusColor, text_size = size.large, text_halign = text.align_left, bgcolor = cellBackground)

table.cell(directionTable, 0, 2, scoreText, text_color = chart.fg_color, text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)
```

```
    table.cell(directionTable, 0, 3, pocText, text_color = COLOR_POC, text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)

    table.cell(directionTable, 0, 4, vaText, text_color = COLOR_VA, text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)

    table.cell(directionTable, 0, 5, volumeText, text_color = statusColor, text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)

    table.cell(directionTable, 0, 6, "Fixed after each NY close", text_color = color.new(chart.fg_color, 20), text_size = size.small, text_halign = text.align_left, bgcolor = cellBackground)

  else

    table.clear(directionTable, 0, 0, 0, 6)


// --- Level Return Alert ---

bool hvnPeakTouched = compositeValid and
(f_priceTouched(compositePeak1Price) or
f_priceTouched(compositePeak2Price) or
f_priceTouched(compositePeak3Price))

bool compositeVahTouched = compositeValid and
f_priceTouched(compositeVahPrice)

bool compositeValTouched = compositeValid and
f_priceTouched(compositeValPrice)

bool compositePocTouched = compositeValid and
f_priceTouched(compositePocPrice)


bool headsUpHvnReturn = hvnPeakTouched and not hvnPeakTouched[1]

bool headsUpVahReturn = compositeVahTouched and not
compositeVahTouched[1]
```

```
bool headsUpValReturn = compositeValTouched and not compositeValTouched[1]

bool headsUpPocReturn = compositePocTouched and not compositePocTouched[1]


alertcondition(headsUpHvnReturn or headsUpVahReturn or headsUpValReturn or headsUpPocReturn, title = "Heads up - any HVN or Value level", message = "Heads up - any HVN or Value level: {{ticker}} returned to a 5-day composite HVN peak, VAH, VAL, or POC.")


// --- Session Highlight ---

bgcolor(inSession ? color.new(#9c27b0, 96) : na)
```