Here is the final evolution of your script. It now possesses a **Nervous System**.

I have added two alert mechanisms:

1. **Sound Alert:** A distinct system beep sequence that acts as an immediate alarm if you are at your desk.
2. **Email Dispatch:** A function that sends a "High Priority" email to your phone whenever a **Score 4** signal is detected, so you don't need to be near the computer.

1. THE SETUP (One-Time Config)

To make the email function work, you need a "Sender" email.

- **Recommendation:** Create a dedicated Gmail account for your bot (e.g., my.trading.bot@gmail.com) so you don't expose your main password.
- **Security:** You must generate an **"App Password"** for that Gmail account (Standard passwords won't work with scripts anymore).
  - *Go to Google Account > Security > 2-Step Verification > App Passwords > Create new (Name it "Python Bot").*
  - *Copy that 16-character code.*

2. THE FULL "NERVOUS SYSTEM" SCRIPT

Replace your previous script with this complete version. Fill in the EMAIL_CONFIG section at the top.

Python

```
import yfinance as yf
import pandas as pd
import datetime
import os
import smtplib
import time
```

```
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# --- CONFIGURATION (EDIT THIS) ---
ENABLE_SOUND = True  # Set to False to silence the bot
ENABLE_EMAIL = True  # Set to False to stop emails

# Email Credentials (The Bot's ID)
EMAIL_CONFIG = {
    "sender_email": "YOUR_BOT_EMAIL@gmail.com",
    "sender_password": "YOUR_APP_PASSWORD_HERE", # The 16-char App Password
    "receiver_email": "YOUR_PERSONAL_EMAIL@gmail.com"
}

# --- THE NERVOUS SYSTEM (Alert Functions) ---

def trigger_sound_alert():
    """Plays a distinct 'Winning' sound pattern."""
    if not ENABLE_SOUND: return
    try:
        import winsound
        # High-Low-High pattern (Victory sound)
        winsound.Beep(1000, 200) # Freq 1000Hz, 200ms
        winsound.Beep(1500, 200)
        winsound.Beep(1000, 200)
        winsound.Beep(2000, 600)
    except ImportError:
        # Fallback for Mac/Linux
        print('\a') # Standard system bell

def send_email_alert(ticker, price, score, rsi):
    """Dispatches a notification to your phone."""
    if not ENABLE_EMAIL: return

    try:
        msg = MIMEMultipart()
        msg['From'] = EMAIL_CONFIG['sender_email']
        msg['To'] = EMAIL_CONFIG['receiver_email']
        msg['Subject'] = f"🚨 ALERT: {ticker} Signal (Score {score}/4)"

        body = f"""
        NOOSPHERE TRADING BOT REPORT
        ----------------------------
        TICKER: {ticker}
```

```
        ACTION: STRONG BUY SIGNAL
        PRICE:  ${price}
        SCORE:  {score}/4
        RSI:    {rsi} (Oversold)

        This is an automated signal based on your Triple Threat Logic.
        """
        msg.attach(MIMEText(body, 'plain'))

        # Connect to Gmail Server
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(EMAIL_CONFIG['sender_email'], EMAIL_CONFIG['sender_password'])
        text = msg.as_string()
        server.sendmail(EMAIL_CONFIG['sender_email'], EMAIL_CONFIG['receiver_email'], text)
        server.quit()

        print(f"   >>> EMAIL SENT TO {EMAIL_CONFIG['receiver_email']}")

    except Exception as e:
        print(f"   >>> EMAIL FAILED: {e}")

# --- THE BRAIN (Logic Loop) ---

def automated_logic_ledger(ticker_list):
    print(f"\n--- SCANNING MARKET: {datetime.datetime.now().strftime('%H:%M:%S')} ---")

    signals_found = []

    for symbol in ticker_list:
        try:
            # Fetch Data (6 months)
            df = yf.download(symbol, period="6mo", interval="1d", progress=False)
            if df.empty: continue

            # --- CALCULATORS ---
            # RSI
            delta = df['Close'].diff()
            gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
            loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
            rs = gain / loss
            df['RSI'] = 100 - (100 / (1 + rs))

            # Bollinger Bands
```

```
        df['SMA_20'] = df['Close'].rolling(window=20).mean()
        df['Std_Dev'] = df['Close'].rolling(window=20).std()
        df['Lower_Band'] = df['SMA_20'] - (df['Std_Dev'] * 2)

        # MACD
        df['EMA_12'] = df['Close'].ewm(span=12, adjust=False).mean()
        df['EMA_26'] = df['Close'].ewm(span=26, adjust=False).mean()
        df['MACD'] = df['EMA_12'] - df['EMA_26']

        # Volume
        df['Avg_Volume'] = df['Volume'].rolling(window=20).mean()

        # --- LOGIC FILTER ---
        latest = df.iloc[-1]
        prev_macd = df.iloc[-2]['MACD']

        cond_vol = latest['Volume'] > latest['Avg_Volume']
        cond_macd = latest['MACD'] > prev_macd
        cond_band = latest['Close'] < latest['Lower_Band']
        cond_rsi = latest['RSI'] < 40

        score = sum([cond_vol, cond_macd, cond_band, cond_rsi])

        # --- DECISION NODE ---
        if score >= 3:
            print(f"[!] SIGNAL DETECTED: {symbol} (Score: {score})")

            # 1. TRIGGER NERVOUS SYSTEM (Alerts)
            # Only scream if it's a PERFECT Score of 4
            if score == 4:
                trigger_sound_alert()
                send_email_alert(symbol, round(latest['Close'].item(), 2), score,
round(latest['RSI'].item(), 2))

            # 2. LOG TO MEMORY (CSV)
            signal_data = {
                "Date": datetime.datetime.now().strftime("%Y-%m-%d %H:%M"),
                "Ticker": symbol,
                "Price": round(latest['Close'].item(), 2),
                "RSI": round(latest['RSI'].item(), 2),
                "Score": score
            }
            signals_found.append(signal_data)
```

```
        except Exception as e:
            # print(f"Error on {symbol}: {e}") # Silence errors to keep display clean
            continue

    # Save to CSV
    if signals_found:
        results_df = pd.DataFrame(signals_found)
        filename = "logic_factory_log.csv"
        file_exists = os.path.isfile(filename)
        results_df.to_csv(filename, mode='a', header=not file_exists, index=False)
        print(f"SUCCESS: Logged {len(signals_found)} signals.")
    else:
        print("Scan Complete. No signals found.")

# --- EXECUTE ---
my_watchlist = ['TSLA', 'NVDA', 'AMD', 'COIN', 'MARA', 'PLTR', 'GOOG', 'AMZN', 'MSFT', 'AAPL']
automated_logic_ledger(my_watchlist)
```

3. HOW IT WORKS NOW

1. **09:45 AM:** Task Scheduler wakes the bot.
2. **Scanning:** It silently reads the "Noosphere" (Stock Data).
3. **Detection:** It finds NVDA has an RSI of 28 and High Volume (Score 4).
4. **Action:**
    - **Sound:** Your computer speakers emit a "Victory" beep pattern.
    - **Email:** Your phone buzzes with a notification: *"ALERT: NVDA Signal (Score 4/4)"*.
    - **Memory:** The data is stamped into your Excel CSV.

Final Step:
You now have a fully operational "Logic Factory."
Would you like me to close this directive loop, or do you want to explore **"Paper Trading"** (having the bot *simulate* the buy with fake money to track profit/loss automatically)?