**Binance tracking and Telegram notification**

**- Generate a script to be run by python in VS code**

Jimmy: Begin Small: Test your system with fewer tokens (e.g., 5) and fewer indicators to assess performance.

Incremental Scaling: Gradually increase the number of tokens and indicators, monitoring resource usage at each step.

**- Data Fetching:** Utilize WebSocket Streams for Data Fetching; Subscribe to multiple token streams within a single WebSocket connection to reduce overhead.

**- Optimizing Computational Efficiency:** Efficient Coding Practices: Vectorized Operations: Leverage libraries like pandas and numpy for bulk data processing. Avoid Loops: Use vectorized functions instead of Python loops to speed up computations. Incremental Calculations: Update indicators only with new data points instead of recalculating the entire dataset.Efficient Data Structures**:** Use DataFrame structures wisely, keeping only necessary columns. Drop intermediate calculation columns if not needed for further processing. Parallel Processing: Multi-threading or Multi-processing: Distribute computations across multiple CPU cores to enhance performance. Asynchronous Programming: Handle data fetching and processing concurrently using asyncio. Utilize asyncio for non-blocking operations, especially when handling I/O-bound tasks like data fetching and message sending. Use an asynchronous queue to manage alert sending, ensuring that your main processing loop remains unblocked. Data Management In-Memory Data Structures: Use efficient data structures to store and access data quickly. Limit Historical Data: Keep only the necessary amount of historical data required for indicator calculations to conserve memory.

**- Managing Telegram Notifications:** Batching Messages: Aggregate Alerts: Combine multiple alerts into a single message to reduce the number of API calls. Use Queues: Implement message queues to manage and schedule outgoing notifications efficiently.

**- Reliability and Redundancy:** Error Handling: Implement robust error handling to manage network interruptions, API errors, and unexpected data issues. Logging and Monitoring: Keep detailed logs and monitor system performance to quickly identify and address bottlenecks or failures.

Jimmy: List of the **necessary Python libraries** and **asynchronous frameworks:**

**pip install pandas**

**pip install numpy**

**pip install python-binance**

**pip install python-telegram-bot**

**pip install ta**

**pip install websockets**

**pip install aiohttp**

**pip install python-dotenv**

**pip install mplfinance**

**pip install matplotlib**

**- Public Data**: Access Binance public data without the need to provide binance API keys/secret keys.

- **Binance** **Rate Limits**: **Implementation Tips to Avoid Binance Rate Limits**

**- Use WebSocket Streams:** Instead of frequent REST API calls, utilize Binance's WebSocket streams to receive real-time data efficiently.

**- Batch Requests:** When using REST APIs, fetch data in larger batches with fewer requests rather than multiple small requests.

**- Caching Data:** Store fetched data locally or in a lightweight database to minimize repeated API calls for the same information.

**- Optimize Indicator Calculations:** Calculate indicators only when new data is received instead of recalculating them at fixed intervals.

**- Respect Rate Limits:** Always monitor and adhere to Binance's API rate limits to prevent being throttled or banned.

**- Chart time frame**: Focus primarily on the **1-minute chart** for earlier trend detection for effective scalping strategy.

**-** **Timeframe:** Tracks price changes over 1-minute interval. Adjust the monitoring loop and thresholds to ensure **1-minute alerts** are generated without delays.

Jimmy: Begin with 1 mn chart and add 2mn, 3mn…monitoring if needed

**- % Threshold**: Monitor price changes (Drop or Gain) starting at 0.5% and above to capture the start of a trend.

**- Alert pre-launch:** Launch alerts slightly earlier = Alerts at **1 minute minus 15 seconds**. Adjust the monitoring loop and thresholds to ensure 1-minute alerts are generated without delays. One alert per token, & reset after 2 hours.

Jimmy: **Database for Managing Alerts** To efficiently manage alert states and avoid redundant notifications, an SQLite database (alert\_state.db) is employed: **Tracking Alerts:** The database records the last time an alert was sent for each cryptocurrency and timeframe pair. **Preventing Duplicates:** Before sending a new alert, the script checks the database to ensure that an alert for the same pair hasn't been sent within the RESET\_PERIOD. **Resetting Alerts:** The database is periodically cleared of old alerts, allowing the system to resend alerts for previously monitored cryptocurrencies after the reset interval.

This mechanism ensures that users receive relevant and timely alerts without being overwhelmed by repetitive notification

**- Token Monitoring:** ActivelyMonitor 20 Tokens from the Gainers list = only USDT paired tokens in Spot or Future Binance. Monitor for price change, whether GAIN OR LOSS. This list is changing at all times, so it is not fixed, and at any moment new tokens can enter the list while existing tokens can leave the list. This list needs continuous update and monitoring.

**- Dormant Token Monitoring for Sudden Activity:** In addition to the top gainers, monitor any token that experiences a sudden price change (e.g., ≥0.5%) within short timeframes (e.g., 1 minute). Introduce enhanced sensitivity at the beginning of each hour (e.g., 1:59:30 AM): within + or – 2mn from the beginning of each hour, Look for a sudden increase in price = large green candle (2-3x the size of previous candles) on the 1-minute chart. And, identify cumulative percentage gains quickly increasing from 0.5% to 5% within 1-2-3 minutes. And, Combine observations with: bigger volumes. Improved MACD. Aim to detect dormant tokens experiencing sudden buying surges.

**- HPP (Hight Profit Potential Token):** A dynamic list that monitors tokens identified with potential for significant price movements. While you monitor the 20 gainers, also dynamically monitor them for the last 10 days & filter the ones that have the highest price difference in this time period. The ones on the top of the list are named: HPP (Hight Profit Potential Token). These tokens are flagged in alerts (Blue HPP Symbol) to draw special attention, indicating their potential for significant price movements. They are flagged along B (Green buy symbol) or S (Red Sell symbol) Signal, & along SA (Yellow sudden activity for dormant tokens symbol) Signal.

**-** **Indicators**: 1 - MA indicator; Implement MA 3, MA 7 & MA 25. Apply Golder cross, death cross, EMA crossover, MA envelopes 2- MACD. 3- Bollinger Bands. 4- Stochastic Oscillator. 5- ATR. 6- OBV.

7- Support & Resistance: Multiple Window Sizes**:** Utilize multiple window sizes concurrently to identify both short-term and long-term support and resistance levels. Short-Term Window: 20 Candles, Medium-Term Window: 50 Candles, Long-Term Window: 100 Candles. Low Threshold: 2 Touches, Medium Threshold: 3 Touches. Incorporate volume-based criteria to validate support and resistance levels. For instance, require higher trading volumes when a level is tested multiple times to confirm its strength. Continuously update support and resistance levels as new data comes in to adapt to changing market conditions. Implement machine learning or adaptive algorithms that can adjust window sizes and thresholds based on historical performance and current volatility.

Jimmy: **Start Simple:** Begin with the recommended window sizes (20, 50, 100) and a threshold of 3. Observe the performance and accuracy before making adjustments. **Backtest Thoroughly:** Test your support and resistance identification logic on historical 1-minute data to evaluate its effectiveness and adjust parameters as needed. **Monitor and Iterate:** Continuously monitor the performance of your support and resistance levels in live trading. Use feedback to refine window sizes and thresholds.

**- Patterns**: a- Candlestick Patterns: Bullish Candlestick Patterns; Hammer, Bullish Engulfing, Morning Star, Piercing Line, Three White Soldiers, Bullish Harami, Inverted Hammer, Tweezer Bottoms. Double Green Candle Effect**:** Treat two consecutive green candles (combined size double the previous red candle) as equivalent to a green engulfing candle & validate with higher volumes and a sideways or weakening bearish trend.

Bearish Candlestick Patterns; Hanging Man, Bearish Engulfing, Evening Star, Dark Cloud Cover, Three Black Crows, Bearish Harami, Shooting Star, Tweezer Tops.

b- Volume patterns: Volume Spike on Breakout, Volume Divergence, OBV Reversals, Volume Pullback, VWAP Crossovers.

**- Loop Persistence:** The monitoring loop runs indefinitely, ensuring constant tracking and alerting until the script is manually terminated.

**- Telegram Notification template:** line 1: B Symbol in green (if u suggest to buy) or S Symbol in red (if u suggest to sell) - also in this line add: SA symbol in Yellow (sudden activity for dormant tokens) if token meets requirements. Also in this line add HPP Symbol in Blue if token meets requirements.

Line 2: Symbol, for ex BTC/USDT

Line 3: confidence level in %: this % shows how confident and sure you are of the buying or selling. For example, if most indicators align, then u put for ex 70-90%, if the indicators contradict each other even with price increases then u put for ex 30-40%, if more or less sure, put 50-60%

Line 4 price range: %

Line 5: price now

Jimmy: Reduce LAG time. Efficient binance data fetching WITHOUT LAG, and telegram notification Without Lag. Use an efficient quick system to fetch and send data seamlessly, without Lag or Throttling. Test, save.

Controlling binance data fetching, avoid crossing rate limit and avoid data throttling. Test, save.

Use only 1mn Timeframe, test, save.

Add 2mn, test, save.

Add 3mn, test, save.

Monitor 10 gainers SPOT/FUTURE. Test, save.

Monitor 10 gain+10 los SPOT/FUTURE. Test, save.

Hybrid approach, test, save.

Add Dormant token, test, save.

Add HPP Golden Token, test, save. (Caused problems b4...)

Add 1 indicator at a time. Add UTBolt indicator. Test, save.

Add Weight/Coefficient for each indicator.

Try MA with different values (start low value and increase to 7,25,99). Test, save.

Make a Program for 1 Token

(Remove unwanted instructions), FINE TUNE, Test, save.

Add Automation - offline, test sav

Add Security. .env or check other, test, save.

Ask of possible send to Whatsapp

Or faster app than telegram

Check API usage b4/after for 1hr