Finding the right balance between “no hallucinations” and “useful output” often means loosening some of the strictest constraints. Here are a few concrete adjustments you can make so the model still produces meaningful analysis without inventing data:

1. **Relax the “output null if you can’t calculate exactly” rule**Instead of forcing nulls for every field the model can’t compute exactly, allow it to make reasonable estimates (with a caveat). For example, rather than:  
   “If you cannot compute a field exactly as defined, output null”  
   you might say:  
   “If you can’t compute this to two‐decimal precision, provide your best estimate and append \"note\": \"estimate based on available data\".”  
   That gives the model permission to “fill in” missing pieces, while still signaling when it’s guessing.
2. **Remove overly detailed indicator definitions**For instance, telling the model exactly how to compute RSI or support levels can be replaced with:  
   “Use common technical‐analysis conventions (20-period SMA, 14-period RSI, local minima/maxima for support and resistance). If you don’t have enough five-minute data for a precise 14-period RSI, estimate based on the last available interval and note that it’s an estimate.”  
   By referencing “common conventions” rather than spelling out every formula, you let the model draw on its internal knowledge, which often leads to more coherent output.
3. **Allow a small nonzero temperature (0.1–0.2)**Zero temperature makes the LLM overly deterministic and brittle. Setting "temperature": 0.1 still keeps it “locked down” against creative fabrication but gives it enough flexibility to produce realistic‐sounding support/resistance levels or confidence ranges.

**Provide a simpler JSON schema without forcing every key**Rather than insisting on an exhaustive template, focus on the fields you really need:  
{

"predictions": {

"1d": { "point": number, "low": number, "high": number, "confidence": integer },

"1w": { … },

"1m": { … }

},

"technical": {

"trend": string,

"support\_levels": [number, …],

"resistance\_levels": [number, …]

},

"recommendation": string

}

1. If the model cannot compute, it can omit a low/high or return null only for those specific sub-fields—rather than nulling out entire sections.
2. **Move heavy numeric calculations out of the LLM completely**If you can pre-compute basic indicators (moving averages, regression coefficients, RSI, etc.) in your own code, then pass those values into the prompt as “pre‐calculated indicators.” The LLM’s job becomes: “Given these SMAs, RSI, and recent price trends, interpret and produce JSON.” That completely eliminates the risk of the LLM inventing a wrong SMA or regression.

### **Example of a more balanced prompt JSON**

{

"model": "gpt-4o",

"temperature": 0.1,

"messages": [

{

"role": "system",

"content": "You are a professional financial analyst. Use only the data and indicators provided or computed. If you must estimate a value (e.g., support level), include a \"note\":\"estimated\" for that field. Always respond in valid JSON format."

},

{

"role": "user",

"content": "Here are 78 five‐minute data points for AAPL (time, price, volume).

Additionally, I have precomputed:

• sma\_20 = 201.23

• rsi\_14 = 58.4

• regression\_slope = 0.02 (R² = 0.65)

Please provide exactly this output:

{

\"predictions\": {

\"1d\": {\"point\": number, \"low\": number, \"high\": number, \"confidence\": integer},

\"1w\": {\"point\": number, \"low\": number, \"high\": number, \"confidence\": integer},

\"1m\": {\"point\": number, \"low\": number, \"high\": number, \"confidence\": integer}

},

\"technical\": {

\"trend\": \"up\" | \"down\" | \"sideways\",

\"support\_levels\": [number, …],

\"resistance\_levels\": [number, …]

},

\"recommendation\": \"buy\" | \"sell\" | \"hold\"

}

Use common conventions for support (local minima) and resistance (local maxima). If you can’t compute a field precisely, estimate and add \"note\":\"estimated\" to that entry."

}

],

"response\_format": { "type": "json\_object" }

}

**Why this works better:**

* You’ve offloaded the heavy math (SMA, RSI, regression) to your code.
* The model no longer has to invent moving‐average calculations. It only interprets the indicators you provide.
* You still get a controlled JSON schema, but you’ve removed the “force‐null-if-uncertain” trap, so the model can give a reasonable prediction instead of defaulting to nonsensical nulls.
* A small temperature (0.1) keeps the model from freely fabricating but lets it express realistic support/resistance levels.

By loosening the “must-null” rules, trimming overly prescriptive formulas, and (ideally) precomputing the core indicators yourself, you’ll see far more coherent, actionable JSON from the model—without a high risk of made-up numbers.