Vamshi Jandhyala

An answer that runs is not an answer you can trust

✠

June 2026

An answer that executes is not the same as an answer you can trust.

A quant research analyst at a hedge fund has an idea. Maybe a pricing anomaly that should mean-revert, maybe a signal hiding in some alternative dataset. Before it can become a backtest, let alone a position, it has to survive contact with data: is the effect real, is it large enough, does it hold out of sample. So the analyst opens one of the new data assistants, types the question in plain English, and gets back a table, a number, a chart.[1] The query ran. The chart looks plausible.

The question that decides everything is the one these tools make easiest to skip: is that number something you can act on?

An answer that executes is not the same as an answer you can trust. A generated SQL query that returns ten thousand rows has proven only that it is syntactically valid and that the columns it named exist. It has not proven that it answered the question the analyst actually asked. And the gap between those two things, between a query that runs and a query that is right, widens precisely as the data grows past what a human can check by eye. On a thousand rows you can scan the output and catch the error. On two billion rows spread across a compute cluster you cannot. The whole problem of building a data-discovery agent for someone whose job depends on being right comes down to this: the analyst is the judge, and at scale they lose the ability to judge.

So the harness, the scaffolding around the model that turns "answer my question" into a result, has one overriding job. Its job is not to write SQL; the model does that. Its job is to manufacture trust in the places where the analyst can no longer manufacture it for themselves. Walk the work stage by stage and you can see where those places are. (What the analyst and the agent are allowed to touch in the first place, the entitlements and information barriers that matter intensely at a hedge fund, is a question of authority rather than trust, and it sits in a different part of the harness.)

[1] The category includes OpenAI's ChatGPT Advanced Data Analysis (originally Code Interpreter, released 2023) and Anthropic's Claude analysis tool (announced 24 October 2024), both of which run model-generated code in a sandbox to compute over uploaded data. Anthropic folded the analysis tool into broader code execution in November 2025; the tools turn over, the requirements do not. As of June 2026.

Knowing what the data is

Before a question can be answered it has to be aimed at the right data, and in a real institution that is rarely obvious. There are a dozen tables with "price" in the name. One is adjusted for splits and dividends and one is not. One is point-in-time, recording what was actually known on each date; another has been quietly restated since, so using it in a backtest leaks the future into the past. That is look-ahead, the error that makes a mediocre strategy look brilliant and loses real money when it does not repeat. A human analyst learns these distinctions slowly and painfully. An agent knows none of them unless the harness tells it.

This is why a data-discovery harness lives or dies on its semantic layer: a machine-readable account of what each table and column means, where it came from, whether it is point-in-time, and what its known traps are. Snowflake's Cortex Analyst, for one, does not turn the model loose on raw schemas. It answers against a curated semantic model that defines the metrics and their meaning, because the people building these systems found that pointing a language model at a bare warehouse produces fluent, confident, wrong SQL.[2] The semantic layer is the difference between an agent that finds the right table and one that queries the wrong one. It is also, not by accident, exactly what makes a body of data agent-ready in the first place.

Turning a question into runnable code

Say the analyst asks for "the average daily return of the top decile by book-to-market, rebalanced monthly." The agent writes SQL. A number comes back. This is where trust quietly drains away, because the English was ambiguous and the SQL resolved the ambiguity in silence. Top decile by which date's book-to-market, the formation date or today's? Rebalanced at the close or the open? Are delisted names handled, or has survivorship crept in because the universe table only holds companies that still exist? Each of those is a choice the model made on the analyst's behalf, and any one of them can turn a real result into an artefact.

The uncomfortable part is how often this goes wrong even on tidy problems. Text-to-SQL is not a solved task. On BIRD, a benchmark built from messy, real-world databases, human data engineers reach about 93% execution accuracy. When the benchmark appeared in 2023 the best models managed around 40%, and after two years of concentrated effort the strongest systems have climbed only into the low 70s to low 80s.[3] That is on questions designed to be answerable. The figure that should give a hedge fund pause comes from Spider 2.0, a benchmark drawn from genuinely enterprise databases, the kind with a thousand columns and real SQL dialects: when it launched, an agent built on a frontier reasoning model solved 21% of the tasks, against 91% on the older, toy-sized Spider.[4] Leaderboard scores have climbed since, but part

[2] Snowflake Cortex Analyst answers natural-language questions by generating SQL against a curated semantic model rather than raw schemas, and Snowflake's own evaluation work treats that semantic model as central to accuracy. Cortex Analyst documentation; "Evaluating Text-to-SQL Accuracy for Real-World BI," Snowflake engineering blog, 29 August 2024. As of June 2026.

[3] BIRD, a text-to-SQL benchmark built from large, messy real-world databases (Li et al., 2023, arXiv:2305.03111). Human data engineers reach 92.96% execution accuracy; the introducing paper reported ChatGPT at 40.08%. By 2025 the strongest leaderboard systems sit in the low 70s to low 80s (for example Arctic-Text2SQL-R1-32B at 71.83%, May 2025). As of June 2026.

[4] Spider 2.0, built from enterprise databases with over 1,000 columns across dialects such as BigQuery

of the gold answer set has been public since late 2024 and, as the audit below shows, a majority of the audited answers are wrong, so how much of that climb is real is precisely the question.

And SQL is the friendly case. Increasingly the agent does not write SQL at all but Python, calling pandas or the firm's own data SDKs to pull and reshape the data, because that is how a quant already works. Generated code has far more room to be quietly wrong than a query does: a merge that drops unmatched rows without a word, two series joined on misaligned indices, a resample that shifts every timestamp by half a day, a fillna that invents data where there was none. None of it raises an error, and code is harder to read at a glance than a SELECT. The benchmarks above measure the SQL floor; the Python ceiling is higher and far less mapped. Whichever target the agent generates, the harness requirement does not change.

So the harness cannot treat the generated query or code as an implementation detail to tuck away. It has to surface the working in terms the analyst can actually check: the assumptions it made, the universe it used, the point-in-time logic it applied, so the analyst is verifying the question rather than squinting at joins or dataframe internals. Generating SQL is the cheap part now. The valuable skill is making the distance between what was asked and what was run visible enough that a busy analyst catches it before it reaches a backtest. An agent that hides the query to feel like magic is optimising for the demo and against the user.

But surfacing is necessary, not sufficient. A query shown is not a query read, and an analyst against a deadline will wave through anything that looks roughly right. That is the same automation bias that makes a human a weaker check the more they are asked to rubber-stamp, the irony at the centre of automation. So the harness cannot merely make the working available; it has to make verifying cheaper than skipping it, and force a real check at the few moments where being wrong is expensive.

Checking it where checking is cheap

There is one stage where trust is easy, and a good harness leans on it hard. On a small, recent sample, a few thousand rows the analyst can pull up and read, errors are visible to the naked eye. The decile membership looks wrong. The returns are implausibly smooth. A ticker that should be there is missing. Eyeballing is a real verification method, and on a sample it is nearly free.

The design move is to make the sample first-class, and to make it honest. Run the analysis on the sample first, let the analyst confirm the logic is what they meant, and only then spend real money and time on the full history. The trap is the unrepresentative sample: the last quiet month behaves nothing like 2008 or March 2020, and a query that looks right on calm data can break on the tails where it matters most. So the sample is where the analyst earns their confidence. Everything downstream is the problem of not quietly losing that confidence once the same logic meets data nobody can read.

---

and Snowflake (Lei et al., 2024, arXiv:2411.07763). At launch (November 2024), a code agent on o1-preview solved 21.3% of tasks, against 91.2% on Spider 1.0 and 73.0% on BIRD. By June 2026 the public leaderboard lists vendor-submitted agents up to 96.7 on Spider 2.0-Snow and 73.1 on Spider 2.0-Lite, but part of the benchmark's gold answer set has been public for self-evaluation since December 2024 (the maintainers warn against training on it), and the audit below finds most of those gold answers incorrect, which makes the top scores hard to interpret. As of June 2026.

And the analyst does not ask once. Discovery is a conversation: filter to financials, now drop the least liquid names, now use the prior close rather than the official one. Each turn builds on the last, so the harness has to hold the evolving definition steady, the universe and the adjustments chosen at the third step still in force at the thirtieth, rather than silently resetting or compounding them. An interactive assistant that loses the thread of its own analysis is one the analyst can only restart, never build on.

When the data outgrows the eye

Now the analyst is satisfied with the logic and wants it run across twenty years and the full cross-section, billions of rows. This is where most of the trust built on the sample can evaporate without a sound, for a reason that is easy to miss: the code that runs at scale is often not the code that ran on the sample. The harness may have validated the logic in a notebook against a small extract, then generated fresh distributed SQL to run against the warehouse, and the two are only as equivalent as the translation between them. A subtly different join, a default that treats nulls another way, a window function that breaks ties differently, and the production number drifts from the sample the analyst signed off on. Nothing errors. The query runs.

A harness that takes trust seriously treats the jump from sample to scale as a place where equivalence has to be preserved and shown, not assumed. In practice that means one logical definition driving both runs, so the scaled query is demonstrably the sample query applied to more rows rather than a fresh guess at the same intent. It means every result carries its provenance, which tables, which point-in-time snapshot, which version of the code, so a number can be traced back and reproduced instead of re-conjured. And because scale costs real money while the analyst waits, it means a cost the analyst can see before they commit to it. Discovery is iterative; a harness that makes each turn of the loop slow or expensive is one the analyst stops reaching for, which is a failure of its own.

Verifying what you cannot see

Here is the hard centre of the whole thing. On the sample the analyst verified by looking. On the full dataset there is nothing to look at: the result is one number, or a single time series, distilled from more data than anyone can inspect. How do you trust a figure you cannot check?

You cannot check it directly, so the harness has to hand you proxies for checking. The most useful is reconciliation: the scaled result should agree with the trusted sample wherever they overlap, and a harness that compares the two automatically and flags any divergence catches precisely the silent sample-to-scale errors above. Then there are invariants, the sanity conditions a careful analyst would test by reflex if they had the time. Does each decile hold a tenth of the universe. Do the portfolio weights sum to one. Are there rows dated before the vendor's history even begins. Has the row count moved overnight for a table that should be static. No one of these proves the number is right. Together they rule out the common ways it is wrong, which is most of what verification can ever offer.

The reason this stage is genuinely hard, and not a checklist to tick, is that even defining the right answer is hard, and the experts get it wrong. A 2026 analysis of two widely used text-to-SQL benchmarks found that the supposedly gold reference

answers were themselves wrong in a majority of the audited cases: 52.8% of the BIRD problems examined and 62.8% of the Spider 2.0 ones in the current version of the work (the CIDR version reported 66.1% on an earlier release of the benchmark), mostly traceable to annotators' limited understanding of the data and the schema. Correcting those answers moved the models' measured accuracy by up to about a third and reshuffled the leaderboard.[5] If the people who build these benchmarks cannot reliably say which query is correct, a harness that hands an analyst one confident number is making a promise it cannot keep. The honest design surfaces its own uncertainty. It shows the assumptions it made and the checks it ran and the ones it could not run, and it leaves the analyst something to weigh, not a number to take on faith.

From a chat to something you can stand behind

The last gap is between an answer and an analysis. A figure produced in a conversation is a one-off: not reproducible next quarter, not auditable when the risk committee asks how it was computed, not protected from the slow drift where a dataset is restated and last week's result no longer replicates. A result you cannot reproduce is a result you cannot defend, and a strategy resting on an analysis no one can re-run is a liability waiting for its day. So the quiet final requirement is that an exploration can be frozen: the provenance the scaled run already carried, promoted into a durable, versioned artefact that regenerates the number exactly and survives the underlying data being restated beneath it. It is the same observability the rest of agent design keeps rediscovering, arriving here at the point where it bites a quant hardest.

The deliverable is trust

Put the stages together and the shape is plain. The analyst is the judge from the first question to the last, and the harness exists to keep them able to judge after the data has grown past anything a person can hold in their head. The answer was never the deliverable. Trust was.

[5] "Text-to-SQL Benchmarks are Broken: An In-Depth Analysis of Annotation Errors," Jin, Choi, Zhu and Kang (University of Illinois), CIDR 2026. The authors find gold-answer (annotation) error rates of 52.8% on BIRD Mini-Dev and 66.1% on the Spider 2.0-Snow problems they examined; correcting a sample of the errors shifts agents' execution accuracy by up to 31% in relative terms and moves rankings by up to three positions among the five agents re-evaluated. The extended version of the work (Jin et al., arXiv:2601.08778, VLDB 2026) re-evaluates sixteen agents, finds rank shifts of up to nine places, and revises the Spider 2.0-Snow error rate to 62.8%. As of June 2026.