



BENCHMARK REPORT

Same data, same model. The data layer decides the answer.

How you deliver your business data to an AI changes how often it answers correctly - and how often it is confidently wrong - even when the data and the model are identical.

43 questions

3 real datasets

759 graded runs

3+ repetitions each

ground truth written **before** any run

Data

Google Search Console, HubSpot deals,
Google Ads - 43 questions, 1,255 runs

Models

Claude Sonnet 5 (1M context),
Claude Opus 5, a low-cost open model

Published

September 2026

Dataddo



RESULTS AT A GLANCE

We changed only the data layer. The answers got measurably better.

78%

of questions answered correctly - the best of every delivery method tested

2x

fewer answers that were wrong and stated as fact

39%

cheaper per question than maintaining documented exports

- **On the questions that decide something** - a rate that must be weighted, two properties that overlap, a month only half-reported - the baselines answered wrongly and confidently about half the time. Dataddo Data to AI, about a quarter.
- **Across every question:** 78.5% answered correctly, the best of the four methods - and wrong-and-confident answers roughly halved, 14.0% against 24.0% and 26.3% ($p = 0.026$, $p = 0.004$).
- **Most database connectors give a model three tools:** list the tables, describe one, run SQL. That is exactly the condition that was confidently wrong one time in four.
- **On messy CRM data, better on 8 questions and worse on none** - 2.8x more answered correctly ($p = 0.008$).

The most convincing benchmark is your own. **Try Dataddo Data to AI on your data - start free at dataddo.com.**

Executive summary

Organizations increasingly connect large language models to their own business data and expect dependable answers. In practice, the quality of those answers depends not only on the model but on how the data is delivered to it. A model handed raw exports must first work out what the data means and how to compute over it before it can respond; a model given a governed, well-described layer can spend its effort on the question itself. This report measures the size of that difference.

We asked 43 pre-registered questions of three real business datasets - Google Search Console, HubSpot deals and Google Ads - and delivered each dataset to the same model in four ways: as plain CSV files, as CSV files with a written data dictionary and a SQL engine, through a generic SQL database connector, and through Dataddo's Data to AI layer over the open Model Context Protocol (MCP). Every question was asked at least three times in every condition - six times where a condition was run twice - and no run shared context with any other. The rows, the model, the prompts and the runtime were identical across conditions.

Dataddo Data to AI answered **78.5%** of the questions correctly - more than any other way of connecting the data, against 71.5% for a generic SQL connector and 69.0% for documented CSV files with a SQL engine. But the more important measure is not how often a model is right. It is how often it is **wrong while sounding right**: an answer stated as fact that does not match the data. Those are the answers that get pasted into a deck before anyone re-checks the arithmetic. Dataddo Data to AI produced one in **14.0%** of runs, against 24.0% and 26.3% for the two baselines - roughly half as often. The gap widens on the questions that decide things, and widens again on the messiest of the three datasets.

HEADLINE RESULTS - SAME DATA, SAME MODEL

78.5%

of questions answered correctly - the best of the four delivery methods tested

2x

fewer confidently wrong answers - 14.0% of runs against 24.0% and 26.3% ($p = 0.026$, $p = 0.004$)

2.8x

more correct answers on messy CRM data - better on 8 questions, worse on none ($p = 0.008$)

A word on where this comes from. We are Dataddo, and we ran this benchmark on our own product. That is exactly why the questions, the acceptance rules and the ground-truth queries were written and frozen *before* any run, why eight of the 43 questions were written by someone who had never seen our semantic model, why seven of them cannot be answered from the data at all, and why the baselines were built to be strong rather than to fail. Section 7 states plainly where the layer made no difference, and how to check all of it on your own data.

1 Why data delivery matters

Even a clean export places the burden of interpretation on the model. Faced with a large, unlabelled dataset, an AI must infer what each column means, decide how to filter and aggregate, and then write and run its own code to produce anything beyond a simple lookup. Much of the time this works. The risk concentrates in questions that require careful handling: a specific time window, an anomaly threshold defined relative to history, or a period-over-period calculation. There, a small methodological slip does not produce an obvious error; it produces a confident, plausible, and wrong number.

That failure mode is what erodes trust in AI over business data. A dashboard that is an hour stale is a minor inconvenience; an agent or analyst assistant that quietly miscalculates a growth rate can send a decision in the wrong direction before anyone re-checks the arithmetic. The question this report examines is whether the way data is delivered to the model measurably reduces that risk.

2 Design of the experiment

The experiment was constructed so that a skeptical reader could reproduce it. Three datasets were served to one model in four conditions that differ in exactly one respect: how the data reaches the model. Everything else was held constant.

2.1 The four conditions

All four used the same runtime (Claude Code), the same model, the same prompts and the same underlying rows - every comparison in Section 3 holds the model constant. Code execution was available in three of them, so no condition was denied the ability to compute; the fourth is the generic SQL connector, where computation happens in the database.

Condition	What the model gets	Left to infer
CSV files The plain export	The full export as correctly named CSV files, plus general-purpose shell and file tools to compute with.	What every column means, which file to use, and how to aggregate it.
CSV files + documentation The strong baseline	The same files, plus a written data dictionary describing every field, plus a SQL engine to query them with.	How to aggregate, and which definitions to apply.
Generic SQL connector What most database MCP servers do	List the tables, describe one, run any read-only SQL. No descriptions, no definitions, no guardrails.	What the tables mean, and how to aggregate them.
Dataddo Data to AI The governed layer	Defined entities, fields and metrics over MCP, each with a plain-language description and stated scope.	Little - meaning and structure travel with the data.

Table 1. The four delivery conditions. Each is a real way a business connects an AI to its data today.

A low-cost-model comparison was also run and is not reported; Section 7.2 explains why.

The second condition is the one to watch. A well-documented export with a SQL engine is a strong baseline, not a straw man, and on one of the three datasets it performs as well as the

governed layer. We report that in Section 7 rather than leaving you to find it.

2.2 The three datasets

Datasets were chosen to span the range of what businesses actually ask an AI to reason over: one flat, well-behaved analytics export, one advertising export, and one genuinely messy operational dataset with duplicate records and overlapping entities.

Dataset	Shape	Rows	Questions	Runs
Google Search Console Flat, self-describing columns	One wide table, five properties	129,209	19	513
HubSpot deals Multi-entity, duplicated records	Deals, owners, stages, pipelines	22,696	14	433
Google Ads Large, regular	Campaign performance by day	2,329,157	10	310
Total		2,481,062	43	1,255

Table 2. The three datasets, all real production data from Dataddo's own account.

Ten delivery variants were tried across the study's 1,255 graded runs. The four conditions above account for 759 of them; the remainder were earlier versions of our own server and a low-cost-model comparison that is not reported here, for the reason given in Section 7.2.

2.3 Questions and scoring

Every question, every acceptance rule and every ground-truth query was written and frozen before a single run took place, and the ground truth was computed directly against a frozen snapshot of the data - never through the layer under test. The question set deliberately includes three kinds that a benchmark can be built to avoid:

- **Hazard questions (18).** A real property of the data decides the answer - overlapping properties that double-count if summed, a rate that must be weighted rather than averaged, a partial month at the end of the window. The plausible method gives the wrong number.
- **Unanswerable questions (7).** The data cannot answer them. The correct response is to decline. An answer, however confident, is scored wrong.

- **Blind questions (8).** Written by someone who had never seen the semantic model, so the layer could not have been shaped to fit them.

Each question was asked independently at least three times in each condition, and six times in the conditions that were run twice, with no context shared between runs. Scoring reads only the model's final structured answer against the frozen rule - never its prose, and never a human judgement of whether it "seemed right". Two grader bugs were found and fixed after runs began; both are logged with their reason, one of them favoured a baseline over our own product, and every run was re-graded afterwards.

NOTE ON THE COMPARISON

The underlying rows are identical in every condition. The treatment does not give the model a better prompt or a more capable model - only a better-described data layer. What it does add is human modelling work: somebody defined the entities, fields and metrics in advance. That is the product, and it is what is being measured.

3 Results

3.1 Correct answers

Across all 43 questions and all three datasets, the governed layer answered more of them correctly than any other way of delivering the same rows.

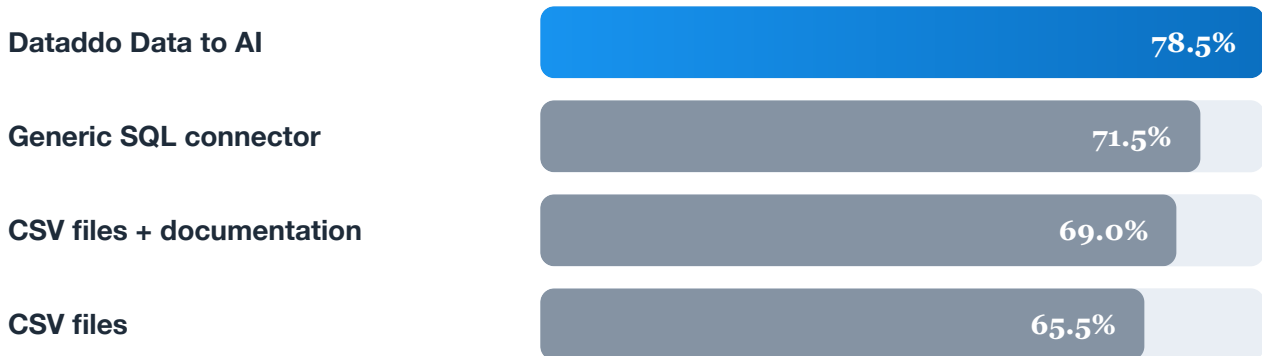


Figure 1. Questions answered correctly, all three datasets. A question the data cannot answer counts as correct only if the model declined it.

3.2 Answers that were wrong and sounded right

The more consequential measure is the share of runs in which the model gave an answer, stated it as fact, and was wrong. Nobody catches those without redoing the work.

Note what the second baseline is. **Most database connectors give a model three tools: list the tables, describe one, run SQL.** That is exactly what the generic SQL connector condition does, and it is what a large share of businesses have already connected their AI to. It answered confidently and wrongly one time in four.

Condition	Runs	Confidently wrong	vs. Dataddo Data to AI	p
Dataddo Data to AI	186	14.0%	-	-
CSV files + documentation	129	24.0%	1.7x more	0.026
Generic SQL connector	186	26.3%	1.9x more	0.004
CSV files	258	33.3%	2.4x more	<0.001

Table 3. Confidently wrong answers, all three datasets. Comparison is against Dataddo Data to AI, Fisher exact test.

3.3 The questions that do not announce they are hard

Eighteen of the 43 questions look ordinary and are not. A real property of the data decides the answer, and it is nowhere in the rows. **Two site properties that overlap, so summing them double-counts. A rate that must be weighted by volume rather than averaged. A final month that is only half-reported.** Every analyst recognises these. A model reading raw columns does not, and the plausible method returns a wrong number with no sign that anything went amiss.

This is where delivery matters most, and where the gap is widest.

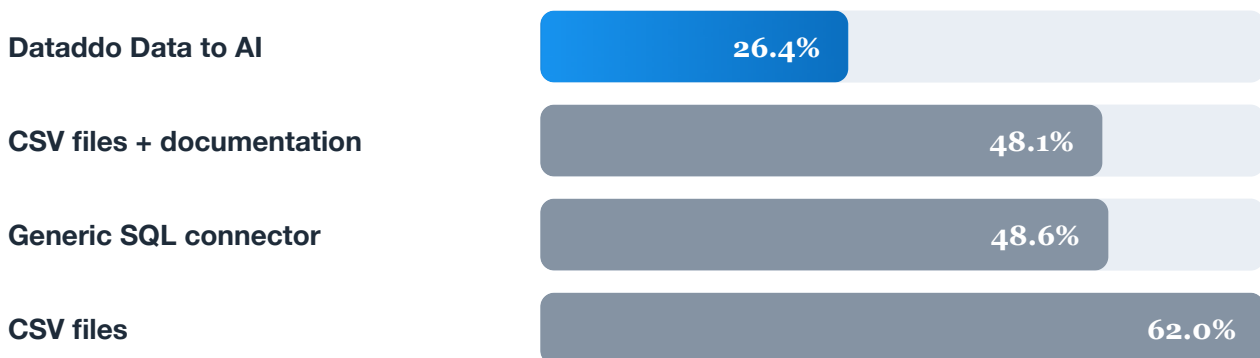


Figure 2. Confidently wrong answers on the 18 hazard questions. Lower is better.

On the questions a business would actually act on, both baselines were confidently wrong **about half the time**. The governed layer was wrong about a quarter of the time. Same rows, same model, same questions.

“Two different baselines for the same country and the same day - in a single reply.”

What an improvised method looks like when you happen to catch it. Section 4 has the transcript.

3.4 Messy data

On HubSpot deals - the dataset with duplicate records, multiple related entities and no self-describing column names - the governed layer answered more questions correctly than any other condition, and the advantage holds question by question rather than resting on one lucky result.

Condition	Correct	Confidently wrong	Questions better / worse vs. Dataddo Data to AI	p
Dataddo Data to AI	17 / 42	52%	-	-
CSV files + documentation	6 / 42	74%	0 better, 8 worse	0.008
Generic SQL connector	6 / 42	71%	0 better, 7 worse	0.016
CSV files	15 / 84	75%	1 better, 7 worse	0.070

Table 4. HubSpot deals, 42 graded runs per condition. Paired comparison is by question, against Dataddo Data to AI.

Against the documented-file baseline, the governed layer was better on eight questions and worse on none. Note the absolute numbers as well as the comparison: on this dataset every condition is wrong more often than it is right. Deals data is hard, and a governed layer improves it substantially without making it easy. We would rather you read that here than discover it in a pilot.

3.5 What it costs

Cost was measured, not estimated - the figures below are the billed cost of the runs above, per question, averaged over every run in each condition.

Condition	Cost per question	Tool calls
Generic SQL connector	\$0.025	3.9
Dataddo Data to AI	\$0.047	4.0
CSV files + documentation	\$0.076	4.7

Table 5. Measured cost per question, all datasets.

Two things follow, and we state both. Dataddo Data to AI answered for **39% less than the documented-file baseline**, and for **1.9x more than a bare SQL connector**. A governed layer is not free; what the premium buys is the wrong-answer rate in Section 3.2.

One caveat cuts in our favour, and we flag it rather than bank it. Every condition here ran with shell and code-execution tools available - that is what lets the file and SQL baselines compute over data of this size at all. Many real deployments give the model no shell: a chat web client, for instance. There, the file and export baselines cannot do the arithmetic, while Dataddo Data to AI still answers, because the computation happens in the layer. Measured against clients like those rather than against a fully-tooled baseline, the comparison would be considerably more favourable to Dataddo Data to AI than the figures above.

There is also a further implication on cost, which we have not validated. Because Dataddo Data to AI raises reliability, a smaller and cheaper model should be able to clear the same bar a frontier model clears on raw files - which would cut the per-question bill well below every figure above. We ran that comparison and are not yet reporting a number (Section 7.2 explains why), so treat it as a reasoned expectation, not a measured result.

4 What a confidently wrong answer looks like

The rates in Section 3 say how often each delivery method produces an answer that is wrong and sounds right. They do not say what one looks like. Below are two, from a separate side-by-side run on a larger Search Console export - 1.1 million rows across five levels of aggregation - in which five analytical questions were put to Claude Opus 5 through CSV files and through Dataddo Data to AI.

That run is an illustration, not a measurement: one pass per condition, too small to carry a rate. Its value is that both failures were captured in full, with the transcripts showing the method that produced them. Neither announced itself.

4.1 A growth figure for the wrong period

The question asked specifically for growth between September and October 2025. Served through Dataddo Data to AI, the model computed month-over-month growth for exactly that window across the top landing pages, and additionally recognised that a redirect between two pricing URLs had split a single page's traffic across two paths, consolidating them before reporting the trend. Served as CSV files, the model instead reported a different period - the most recent month-over-month movement in the data, not the window requested - and asserted that a page had "disappeared completely" when the underlying rows show it was still growing. It never produced the figure the question asked for.

Nothing in that answer looks wrong. It contains a page, a percentage and a direction, and the period it silently substituted is a plausible one. A reader who did not re-run the query would have no reason to doubt it.

4.2 Two different baselines for the same day

This question requires each country's historical daily average, a three-times threshold applied per country per day, and a consistent ranking. Served through Dataddo Data to AI, the model identified the single largest anomaly correctly and explicitly noted that "largest" is ambiguous - a spike can be largest by absolute volume or by ratio to a country's own baseline - and distinguished the two. Served as CSV files, the model produced inconsistent figures for the same country and day drawn from the identical data, and led with a negligible single-click outlier ranked purely by ratio.

The inconsistency needs no answer key to detect. Two baselines for the same event in one reply mean the aggregation was improvised at each step rather than defined once - the mechanism behind the rates in Section 3.2. A method re-invented per question is occasionally re-invented differently.

5 How Dataddo Data to AI works, and why it answers differently

Dataddo Data to AI was one of the four delivery methods tested. Its results are consistent with how it is built, so it is worth setting out the architecture before explaining the mechanism.

5.1 The architecture

Business systems connect through Dataddo's managed connectors and are landed in a stable, deduplicated store (SmartCache), so the model queries a consistent copy rather than live systems. Over that store sits a semantic layer: defined entities, fields and metrics, each carrying a plain-language description and a stated scope. The model reaches this layer over the open MCP standard - the same protocol a web client or agent would use - and never touches the raw tables. In effect, the modelling an analyst would otherwise redo for every question is done once, in advance, and travels with the data.

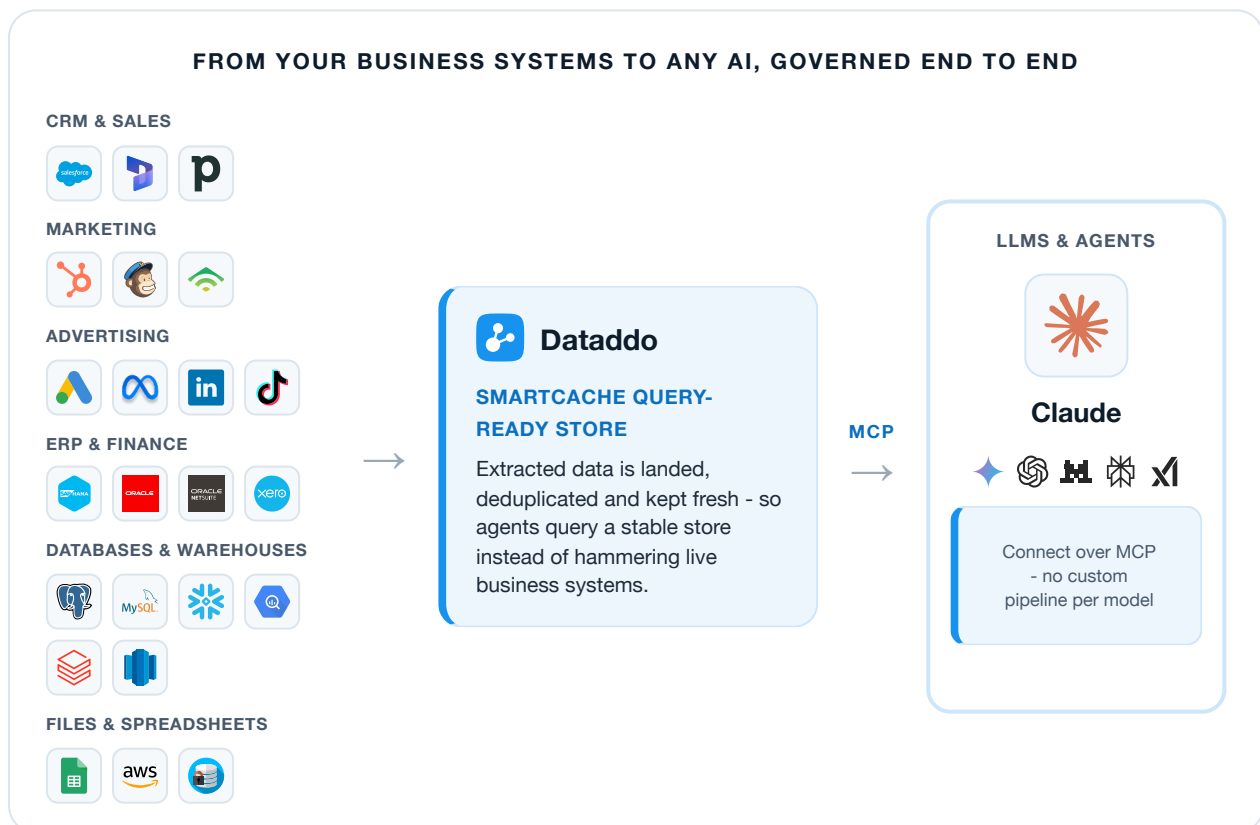


Figure 3. How Dataddo connects business systems to a model: extraction into a query-ready store, a defined semantic layer, and delivery to Claude and other models over the open MCP standard. Shown for orientation - the benchmark exercised the semantic layer and MCP delivery only.

5.2 Why that changes the answers

Four properties of that layer map onto the failure modes in Section 4. This is an interpretation of the mechanism, reasoned from the observed errors; it is not a component-by-component measurement.

- **Every field arrives with a description.** The model does not have to infer what a column means or which level of detail to use. This is the difference between reading a column called amount and being told which currency it is in, whether it is net of discount, and which rows are excluded.
- **Definitions are applied once, not re-derived per query.** Click-through rate, a calendar month boundary, what counts as an open deal - defined in the layer rather than improvised per question. This is what removes the inconsistency in Section 4.2.
- **Scope travels with the data.** Where a dataset is filtered, partial or covers a particular window, the layer states it in the answer rather than leaving the model to assume it has everything. Most of the hazard questions in Section 2.3 turn on exactly this.
- **Some questions are refused rather than answered.** Where a question would require an aggregation the data cannot support - double-counting across a one-to-many relationship, summing an identifier, averaging a rate that must be weighted - the layer declines and says why. A refusal costs a follow-up question; a confident wrong answer costs a decision.

ON THE EVIDENCE

The four properties above are the ones exercised by this benchmark. Dataddo's wider platform includes capabilities - lineage, quality signals, PII controls - that this test did not measure and that are not claimed here as contributors to the result.

6 Interpretation

The finding is specific: across 759 graded runs on three real datasets, how the data reached the model changed how often it answered wrongly while sounding right - roughly halving it against the delivery method most businesses have today, a generic SQL connector, and against documented exports. The advantage concentrated exactly where it matters most: on the questions whose correct answer depends on a property of the data that is invisible in the rows, where both baselines were wrong about half the time and the governed layer about a quarter.

The cost consequence is measured rather than inferred. A governed layer costs more per question than a bare SQL connector and 39% less than maintaining documented exports - and it is the documented exports it replaces, since the SQL connector is the thing it outperforms. For a team running a large volume of questions, the larger number on the invoice is not the price of the layer but the cost of the answers nobody can trust.

What this does not establish is a rate for your data. Three datasets from one company and one modelling author is not a general law about governed layers. It is a measurement precise enough to be worth reproducing, which is the next section.

7 Where it did not help, and how to check

7.1 Where the layer made no difference

We publish the dataset where we do not win, because a layer that only works on data you already understand is not worth buying.

The results above are the whole study. Split by dataset, one of the three goes the other way. On Google Search Console alone, a documented CSV export queried with a SQL engine was **as accurate as the governed layer, and never once confidently wrong across 57 runs**. Paired question by question, Dataddo Data to AI was better on two and worse on five. There is no advantage to report there, and we are not going to report one.

That result is worth more than the ones in our favour, because it says what the layer is for. Search Console data is a single flat table whose columns are named `clicks`, `impressions`, `ctr` and `position`. Anyone can describe it in a paragraph, and once described, a capable model handles it. The governed layer earns its place where that stops being true: many related entities, duplicated records, coded values, filtered extracts, and business definitions that are nowhere in the column names. That is HubSpot deals in this study, and it is most operational data in practice.

Two further limits. Google Ads was answered almost perfectly in every condition, so it separates nothing - which means the whole-study advantage in Section 3 is carried by Search Console and, mostly, by HubSpot deals. And on deals every condition, ours included, was wrong more often than right. A governed layer improves hard data substantially without making it easy.

7.2 What we tried and are not reporting

We tested whether a governed layer lets a small, inexpensive model do a frontier model's job. On HubSpot deals and Google Ads the low-cost model came close to the frontier ones, at roughly a twentieth of the cost per question. We are not reporting that as a result, because the same comparison on Search Console is not yet valid: that flow is a rolling window, it advanced by three days between the frozen export the file conditions read and the run, so the two conditions were answering questions about different data. A two-dataset result is not a finding, and we would rather say so than publish the half that looks good.

It is a fair question and we intend to answer it properly. What we can say from the datasets that are comparable is that the layer's advantage does not depend on model size: it appeared on both.

7.3 What a skeptic should check

- **We wrote the questions and we built the product.** Mitigated by freezing the question set, the acceptance rules and the ground-truth queries before any run, by grading only the structured answer against those rules, and by including eight questions written blind. Mitigated, not eliminated.
- **The ground truth is ours.** It was computed directly against a frozen snapshot, never through the layer under test, so the layer cannot define its own correctness. Where a question admits more than one defensible convention, the rule accepts both.
- **Three datasets, one company, one modelling author.** The rates here measure this setup. They are not a constant.
- **Two grader bugs were fixed mid-study.** Both logged with their reason and date, one of them correcting a rule that had been unfair to a baseline; every run was re-graded afterwards.

7.4 See it on your own data

Point Dataddo Data to AI at your own business data. Ask the questions your teams actually ask - especially the ones whose answer depends on something only your team knows - run them on the model you already use, and repeat each one a few times rather than once. Then hand the same data to the model as files and compare. Count the answers that were wrong and sounded right. That is the number that decides whether an AI can be trusted in front of your business.

Connecting is quick: Dataddo speaks to Claude and other models over the open MCP standard, with no custom pipeline per model, and you can start free.

8 Conclusion

Across 759 graded runs on identical data and an identical model, changing only how the data was delivered produced the most correct answers of any method tested - 78.5% - and roughly halved the answers that were wrong and stated as fact. On the questions whose answer turns on something you cannot see in the rows, the baselines were confidently wrong about half the time and the governed layer about a quarter. On messy, multi-entity business data it answered nearly three times as many questions correctly and was never worse on any single one. It did so for 39% less than the cost of maintaining documented exports.

It did not help on a single flat, well-labelled export, and we have said so. For teams putting AI in front of business data, how that data is described and delivered is not an implementation detail; it is a determinant of whether the answer can be relied upon. That is why we built Dataddo Data to AI - and why we would rather you prove it on your own data than take our word for it.

Send us five questions your team actually asks

We will connect your data, answer them through Dataddo Data to AI, and show you the working - including any we decline, and why. Then ask the same five of your files and compare. That is the whole argument, on your numbers.

[Test Dataddo Data to AI](#)[See how it connects](#)

This benchmark was produced by Dataddo on its own product and data. Measured study: 43 pre-registered questions across Google Search Console (129,209 rows), HubSpot deals (22,696 rows) and Google Ads (2,329,157 rows); ten delivery variants tried in total, of which the four reported here account for 759 of the 1,255 graded runs; at least three independent repetitions of every question in every condition. Model: claude-sonnet-5 via Claude Code; all reported conditions use that model. Ground truth was computed directly against frozen snapshots, never through the layer under test. Significance is Fisher's exact test for rates and a two-sided sign test for paired question-level comparisons. The worked example in Section 4 is a separate single-run comparison on a 1.1 million-row Search Console export using claude-opus-5, and carries no rate. Question set, acceptance rules, ground-truth queries, grader changelog and transcripts available on request.