SMALL WEB, LLC · TECHNICAL WHITE PAPER · 2026-11-26

The Trust Layer for AI Data Analysis

TinyPivot MCP (@smallwebco/tinypivot-mcp) · https://tiny-pivot.com/whitepaper · Zero Heavy Tooling · < 2,500 Free Schema Tokens

Abstract — Large language models write fluent SQL and Python, yet routinely fail silently on production data: multiplying totals 2.25× on many-to-many joins, dropping 40% of NULL rows on `!=`, shipping broken A/B splits, or burning dozens of turns wrestling with multi-sheet Excel workbooks. We evaluate **TinyPivot MCP** — a 100% local, zero-bloat DuckDB verification and analytics server — across two strictly separated tracks: (A) an **Internal Controlled Suite** of 18 unhinted BI, Relational, and Hard Data Science tasks across 5 local and frontier models (270 runs), and (B) **External Public Benchmarks** comprising a controlled same-model A/B on **MIT KramaBench (ICLR 2026)** and three live 450-task blind leaderboard submissions on **Adyen / Hugging Face `adyen/DABstep`**. TinyPivot lifts internal accuracy monotonically from **91% → 93% Free → 100% Pro** (2.1× faster Hard Data Science; Thinking Machines `Inkling 2 Mini` +22 pts & −50% output tokens) and delivers consistent same-model lifts across all 3 models on MIT KramaBench (**72.5% → 75.2%** across 372 runs; Haiku +6.1 pts, Biomedical `.xlsx` +22.2 pts & 2.14× faster; local DeepSeek V4.1 +1.1 pts & +10.7 pts above #2 TalkTask on the same model family).

91% → 100%
Correct answers across 5 AI models on our 18 trap-laden analysis tasks (90/90 with TinyPivot Pro)

2.1× faster
On the hardest statistics questions, averaged across all 5 models — while also getting them right

72.5% → 75.2%
MIT KramaBench, an independent benchmark: all 3 models improved, Claude Haiku the most (+6.1 pts)

97.2% Easy
#1 on the Easy tier of Adyen's blind-scored DABstep leaderboard (Claude Sonnet 5.5; 68.8% on Hard)

1. Problem Statement & Honest Evaluation Process

Syntax errors are harmless — an agent fixes them in one retry. The real barrier to trusting AI analytics in executive reviews is **silent analytical failure**: queries that execute cleanly in 12 ms while returning confident, plausible, wrong numbers. To prove TinyPivot adds genuine reliability rather than hiding behind a strong frontier model, we separate our benchmarks into two tracks:

TRACK A · INTERNAL CONTROLLED SUITE

18 Unhinted BI, Relational & Hard DS Tasks

Zero hints in any prompt across 5 models × 3 arms (270 runs): Thinking Machines `Inkling 2 Mini`, DGX Spark `DeepSeek V4.1`, `Claude Haiku 5.5`, `Sonnet 5.5`, and `Opus 5.5`.

TRACK B · EXTERNAL PUBLIC BENCHMARKS

MIT KramaBench (ICLR 2026) & Adyen DABstep

Independent third-party datasets we did not author: controlled 3-model same-model A/B on 62 multi-file KramaBench tasks (372 runs across Sonnet 5.5, Haiku 5.5 & local DeepSeek V4.1) plus 3 live 450-task blind Adyen DABstep leaderboard submissions.

2. Track A — Internal Controlled Suite (270 Runs)

STANDOUT LOCAL BOOST · THINKING MACHINES INKLING 2 MINI (MAC LOCAL, $0 CLOUD SPEND): TinyPivot Pro lifted Mac-local **Thinking Machines Inkling 2 Mini** by **+22.2 pts** (78% → 83% Free → 100% Pro, matching Opus 5.5), cut latency **−44%** (46.1s → 26.0s; 3.8× faster on Hard DS), and halved output tokens (**−50%**), while local Recursive Data Memory cut repeat schema inquiry tokens a further **−47%** (6 → 3 turns).

MODEL & HARDWARE	BASELINE	TINYPIVOT FREE	TINYPIVOT PRO	BASELINE → PRO IMPACT
Thinking Machines Inkling 2 Mini Small local (Mac) · vs Bare DuckDB SQL	**14/18 (78%)** 46.1s/task · 5.11 turns · $0.027 API†	**15/18 (83%)** 37.3s/task · 5.11 turns · $0.059 API†	**18/18 (100%)** 26.0s/task · 4.22 turns · $0.046 API†	+22.2 pts (78% → 100%) · 44% faster (3.8× Hard DS) −50% out · −17% turns · $0 local · $0.046 API†
DeepSeek V4.1 Flash 1017 Local cluster (DGX Spark) · vs Bare DuckDB SQL	**17/18 (94%)** 25.7s/task · 4.94 turns · $0.059 API†	**17/18 (94%)** 27.7s/task · 5.28 turns · $0.090 API†	**18/18 (100%)** 15.6s/task · 3.44 turns · $0.055 API†	+5.6 pts (94% → 100%) · 39% faster (3.4× Hard DS) −48% out · −30% turns · −7% API† ($0.055) · $0 local
Claude Haiku 5.5 Fast cloud · vs Python / Bash	**15/18 (83%)** 13.1s/task · 3.94 turns · $0.074 API*	**16/18 (89%)** 13.9s/task · 4.11 turns · $0.083 API*	**18/18 (100%)** 12.5s/task · 5.61 turns · $0.083 API*	+16.7 pts (83% → 100%) · 5% faster (23% Hard DS) −12% out on Hard DS · +0.05¢/task vs Python*
Claude Sonnet 5.5 Balanced frontier · vs Python / Bash	**18/18 (100%)** 20.0s/task · 3.89 turns · $1.22 API*	**18/18 (100%)** 22.1s/task · 4.67 turns · $0.88 API*	**18/18 (100%)** 18.8s/task · 5.11 turns · $1.01 API*	100% Free & Pro + lineage · 6% faster (21% Hard DS) 6/6 Hard DS in 25.9s · −17% Pro · −28% Free API*
Claude Opus 5.5 Flagship frontier · vs Python / Bash	**18/18 (100%)** 21.5s/task · 3.61 turns · $2.32 API*	**18/18 (100%)** 21.9s/task · 4.11 turns · $2.65 API*	**18/18 (100%)** 22.0s/task · 4.28 turns · $1.41 API*	100% parity (Free & Pro) · 19% faster Hard DS −17% out on Hard DS · −39% warm API* ($1.41)
Combined (90 model-tasks/arm)	**82/90 (91%)**	**84/90 (93%)**	**90/90 (100%)**	**+9 pts overall · −25% faster avg (25.3s → 19s) · 2.1× faster Hard DS**

*Cloud API-equiv: Claude 5.5 arms ran via local Claude Code subscription ($0 extra spend) with costs calculated at published Anthropic API token rates (warm-session caching on Sonnet/Opus). †Imputed local API pricing: local models ran at $0 marginal cost; API-equiv USD imputed at 2026 DeepSeek-Flash Peak ($0.30/$0.006/$1.20) for DeepSeek V4.1 and GPT-4o-mini tier ($0.15/$0.075/$0.60) for Inkling 2 Mini.

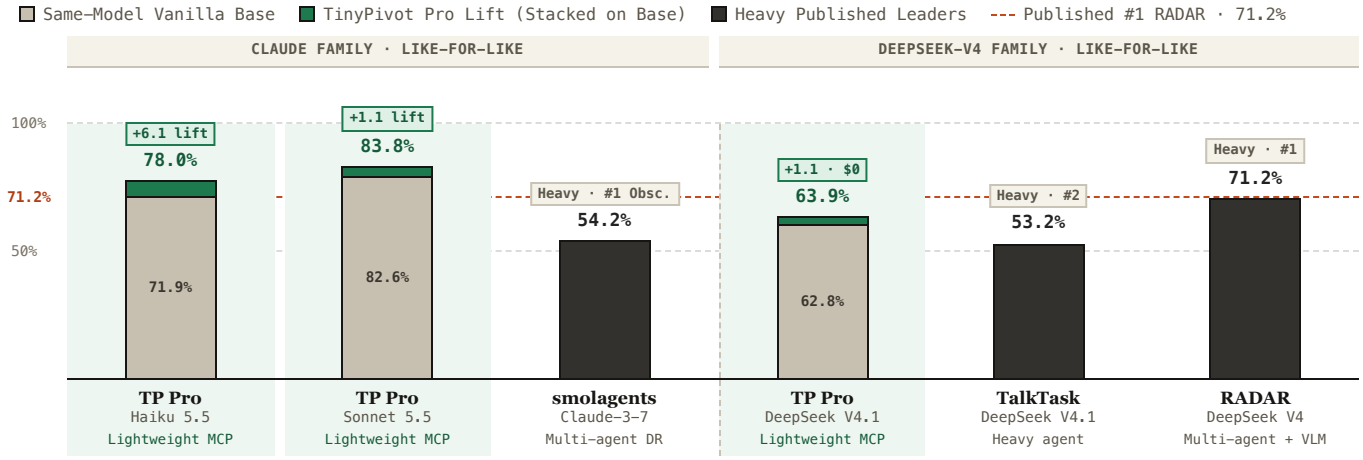
3. Track B — External Public Benchmarks & Leaderboard Rankings

To separate base-model strength from TinyPivot's direct lift, we plot our TinyPivot Pro runs (green), same-model Vanilla baselines (sand), and published leaderboard leaders (charcoal) side-by-side.

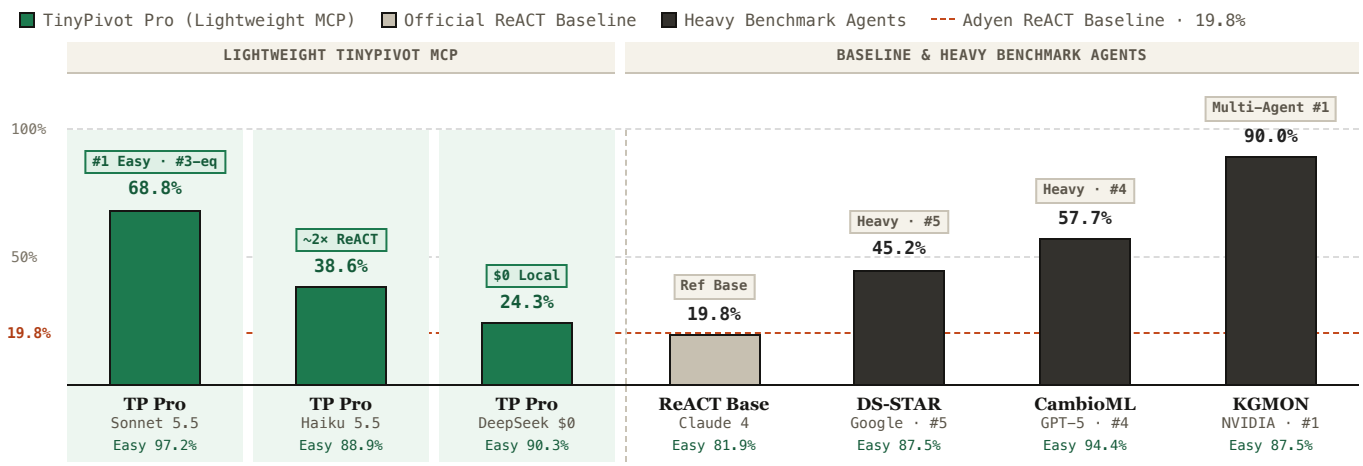
TINYPIVOT MCP (v0.8.5) · LIGHTWEIGHT DROP-IN TOOL
Single local MCP (< 2,500 Free schema tokens): DuckDB SQL + smart `.xlsx` + 1-call profiling. Zero Jupyter, zero VLM judges, zero multi-agent loops, zero benchmark prompts.

LEADERBOARD SYSTEMS · HEAVY RESEARCH INSTRUMENTATION
RADAR, TalkTask, smolagents DR, NVIDIA KGMON & Google DS-STAR use multi-agent planner/verifier loops, persistent Jupyter sandboxes, VLM checks, or benchmark-tuned workflows.

EXTERNAL 01 · MIT KRAMABENCH (ICLR 2026) — STACKED SAME-MODEL LIFT VS. HEAVY LEADERS · https://kramabench.org/



EXTERNAL 02 · OFFICIAL 450-TASK BLIND ADYEN DABSTEP (HARD ACCURACY %) · huggingface.co/spaces/adyen/DABstep



Honest attribution: On `adyen/DABstep`, Sonnet 5.5's 97.2% Easy / 68.8% Hard score is strongly driven by base-model reasoning paired with TinyPivot's DuckDB workflow (our 10-task Dev A/B showed 90% parity on Sonnet, 80% parity on Opus, and +10 pts on Haiku, 60% → 70%). Our controlled same-model Internal Suite (91% → 100%) and 3-model MIT KramaBench A/B (72.5% → 75.2%, Haiku +6.1 pts, local DeepSeek +1.1 pts & +10.7 pts above #2 TalkTask) isolate TinyPivot's direct lift.

4. Track C — Direct Warehouse Connections (192 Runs Across 4 Models)

8 questions over fake Acme tables in BigQuery (2.7 million orders), each hiding a trap, run 2 times per setup on Claude Haiku and Claude Sonnet (96 Claude Code runs, $6.33 cold-cache API spend) and on small local models Inkling 2 Mini and DeepSeek V4.1 Flash (96 local runs, 8-turn cap), on 2026-11-27.

Cloud Models (Claude Code)

MODEL	BQ COMMAND LINE	EXPORT, THEN TINYPIVOT	TINYPIVOT DIRECT
Claude Haiku	16/16 correct · 37.4s · $0.11	16/16 correct · 89.7s · $0.16	16/16 correct · 17.0s · $0.08
Claude Sonnet	16/16 correct · 44.4s · $2.05	16/16 correct · 63.0s · $2.30	16/16 correct · 26.6s · $1.63

Small Local Models (8-Turn Cap)

MODEL	BARE BIGQUERY SQL	LOCAL PARQUET CEILING	TINYPIVOT PRO DIRECT
Inkling 2 Mini (Mac local)	14/16 correct · 56.6s · 7.8t · est. $0.0416	16/16 correct · 64.2s · 6.6t · est. $0.0755	16/16 correct · 58.7s · 3.6t · est. $0.0451
DeepSeek V4.1 Flash (DGX Spark)	14/16 correct · 35.5s · 6.9t · est. $0.0868	16/16 correct · 30.3s · 4.6t · est. $0.0943	16/16 correct · 21.8s · 3.5t · est. $0.0525

Time is the average per question; Claude cost is cold-cache API spend for all 16 runs in a cell; local models run at $0 with API-equivalent token estimates shown. **bq command line:** The agent with a shell and Google's bq command line, and no TinyPivot. It can write any SQL it likes. **Export, then TinyPivot:** Shell plus TinyPivot Pro, told to export what it needs with bq and analyse the exported file locally. **TinyPivot direct:** TinyPivot Pro's 1-call warehouse profile and query_bigquery (with batched queries[], session-cached trap audits and dialect repair hints): no shell, no files.

- On Claude Haiku and Sonnet, every setup answered all 96 runs correctly — and with 1-call warehouse profiling, batched queries[], session-cached audits and dialect repair hints, TinyPivot direct was 46% faster than the bq command line (17.0s vs 37.4s on Haiku; 26.6s vs 44.4s on Sonnet) and 71% faster than exporting first (89.7s / 63.0s).
- It was also cheaper than the command line on both Claude models (cold-cache): 25% less API spend on Haiku ($0.0832 vs $0.1110, 4.1 vs 5.8 turns) and 20% less on Sonnet ($1.63 vs $2.05). Earlier direct runs without 1-call warehouse profiling took 6.7–7.1 turns and cost 4–15% more than bq.
- On small local models (Inkling 2 Mini and DeepSeek V4.1 Flash, 8-turn cap), TinyPivot Pro direct lifted accuracy from 88% (28/32 on bare BigQuery SQL) to 100% (32/32), matching the pre-exported local Parquet ceiling.
- It cut turns roughly in half on both local models (7.8 → 3.6 on Inkling, 6.9 → 3.5 on DeepSeek) and cut SQL errors from 22 combined on the v0.9.2 baseline to 2 total (0 on DeepSeek, 2 on Inkling). On DeepSeek, direct BigQuery beat both bare SQL (-39% latency, -40% est. cost) and local Parquet files (-28% latency, -44% est. cost).
- What drove the step change: a 1-call warehouse profile() brief (0-scan metadata + 1 single-pass aggregate probe per table surfacing schemas, distinct filter values, date ranges, join fan-out, NULL shares and mixed currencies), session-cached probes, batched queries[], and deterministic dialect/column repair hints.
- Limits: eight questions and two runs per setup is a focused benchmark on BigQuery; multi-warehouse live tests cover Snowflake, Databricks, ClickHouse, MySQL and Postgres.

5. Lightweight Neuro-Symbolic Architecture (< 2,500 Free Schema Tokens)

TinyPivot stays fast, local, and dependency-light by pairing LLM reasoning with deterministic DuckDB & TypeScript verification — zero Jupyter servers, zero VLM judges, and raw rows never leave your machine.

01

Checks every query

Runs lightweight DuckDB probes (~5–15 ms) after each query. Completely silent when SQL is clean; emits actionable [AUDIT] blocks (unweighted AVG of rates/positions, fan-out multipliers, partial-period daily run-rates, excluded NULL totals, zero-row filter suggestions, duplicate event IDs, mixed currencies, extreme outliers, and 1-call analyze routing hints) only when a real trap triggers.

02

Reads whole folders and Excel files in one step

profile({ dirPath }) collapses partitioned file series (e.g. 22 yearly CSVs) into DuckDB glob hints and inspects multi-sheet .xlsx workbooks (auto-detecting headers below preamble title/note rows!) without pasting raw rows into context. Named queries persist as DuckDB views across turns, while TinyPivot Free stays strictly under 2,500 schema tokens.

03

Tested statistics, plus memory

Five deterministic statistical engines (explain_change, ab_test with χ^2 SRM + Holm-Bonferroni + whale trim, controlled OLS regression, right-censored cohorts, seasonal anomalies) run in 10–40 ms. Local Recursive Data Memory (~/.local/share/tinypivot/memory.json) learns traps and Studio edits to cut repeat schema tokens by 47%.

04

Shows its work

Every chart and pivot links back to editable SQL (with lossless revision history), clickable Underlying Source Rows, order-independent SHA-256 Data Hashes, claim-checked prose, and zero-CDN Offline HTML Notebooks — or pass --no-studio (TINYPIVOT_NO_STUDIO=1) for zero-popup headless CI and benchmarks.

6. Unlimited Local Rows on Free & 10-Second Setup

TINYPIVOT MCP FREE · $0 (NO SIGNUP)

Unlimited Local Rows + Studio Lineage

Unlimited local rows (CSV, TSV, JSON, Parquet, smart `.xlsx` header/notes reading), 1-call folder profiling with partitioned file-series collapsing, DuckDB SQL, Studio IDE + Proof-of-Work Lineage, headless `--no-studio` mode, offline HTML export, basic SQL guardrails, and `analyze(explain_change, ab_test)`.

TINYPIVOT MCP PRO · $149/YR OR $399 LIFETIME

Full Enterprise Trust & Data Science Engine

Deep relational & statistical audits (join fan-out, chasm traps, partial trailing periods, silent NULL drops, extreme outliers), full `analyze` shelf (SRM + Holm-Bonferroni, controlled OLS, right-censored cohorts, seasonal change-points), cross-file join keys, Recursive Data Memory (−47% tokens), 10 themes, white-label HTML/`.xlsx` export, and read-only PostgreSQL.

1-COMMAND MULTI-HARNESS INSTALL (CLAUDE CODE, CLAUDE DESKTOP, CURSOR, ANTIGRAVITY, CODEX, OPENCODE, PI):

`npx @smallwebco/tinypivot-mcp@latest init` · https://tiny-pivot.com/docs · https://tiny-pivot.com/pricing

Appendix A — All 18 Internal Controlled Benchmark Tasks (Exact Prompts)

01 · Executive YoY Revenue Summary (Routine BI)
"Using orders_saas.csv in this folder, summarize 2025 vs 2026 total revenue, YoY growth %, and the leading region in 2026."
Trap: Clean baseline executive KPI check — ensures guardrails stay 100% silent with zero false positives.
Guard: Silent-when-clean DuckDB SQL execution + exact-table return.

02 · 6-Day Partial Trailing Month (Routine BI)
"Using monthly_sales.csv in this folder, summarize our monthly revenue trend from Jan to Apr 2026 and tell me how April is performing vs March."
Trap: April only covers Apr 1–6 (6 days), making raw monthly revenue look like a −79% crash when daily run-rate actually grew.
Guard: Pro partial-period audit detects incomplete trailing month and computes exact daily run-rate.

03 · 40% NULL Anti-Join Exclusion (Routine BI)
"Using promo_orders.csv in this folder, what is our total revenue for orders without the 'PROMO20' discount code?"
Trap: SQL WHERE discount_code != 'PROMO20' silently drops all 40% NULL discount_code rows.
Guard: Pro NULL-exclusion audit counts dropped NULL rows and flags IS DISTINCT FROM / IS NULL.

04 · Whale Orders Inverting Regional Leader (Routine BI)
"Using regional_orders.csv in this folder, which region led revenue — East or West — and by how much?"
Trap: Two $250k anomalous whale orders in East invert the underlying East vs. West regional ranking.
Guard: Pro extreme-outlier audit surfaces the 2 whale order IDs alongside clean vs. raw totals.

05 · 1.35× Duplicate Event Ingestion (Routine BI)
"Using checkout_events.csv in this folder, what was our total checkout revenue and total order count?"
Trap: 35% of event_id rows were double-ingested, inflating naive SUM(amount) and COUNT(*).
Guard: Pro duplicate-ID audit detects non-unique primary event keys and prompts deduplication.

06 · Mixed USD/EUR/GBP/JPY Currencies (Routine BI)
"Using global_tx.csv (and fx_rates.csv if needed) in this folder, what is our total revenue across all regions?"
Trap: Naive SUM(amount) adds USD, EUR, GBP, and JPY face values together without FX conversion.
Guard: Pro mixed-currency audit detects multi-currency amounts and flags FX normalization.

07 · 2.25× Dimension Join Fan-Out (Relational Analytics)
"Using orders.csv and account_reps.csv in this folder, report total revenue by region and overall total."
Trap: Joining orders to multi-row account_reps silently multiplies total revenue by 2.25×.
Guard: Pro join-fanout audit probes key uniqueness on both sides and alerts on row-multiplication.

08 · 3-Table Multi-Fact Chasm Trap (Relational Analytics)
"Using accounts.csv, orders.csv, and support_tickets.csv in this folder, report total revenue and total support tickets by tier plus grand totals."
Trap: Joining two fact tables (orders + support_tickets) via accounts creates a Cartesian explosion (6×–8× totals).
Guard: Pro join-fanout audit catches multi-fact row explosion and suggests pre-aggregating each fact in CTEs.

09 · Unweighted Conversion-Rate Average (Relational Analytics)
"Using channel_daily.csv in this folder, what is our overall conversion rate (%) and which channel converts best?"
Trap: AVG(conversion_rate) weights tiny-traffic days equally with high-volume days and flips the top channel.
Guard: Built-in non-additive rate guard flags AVG(rate) and requires SUM(conversions) / SUM(sessions).

10 · Kitagawa Mix-vs-Rate Simpson's Paradox (Relational Analytics)
"Using q4_funnel_events.csv in this folder, why did our blended conversion rate drop from 2026-Q3 to 2026-Q4? Quantify the drivers in percentage points."
Trap: Every individual channel improved conversion rate, yet blended conversion dropped due to traffic mix shift.
Guard: 1-call analyze(explain_change) computes exact Kitagawa rate vs. mix percentage-point decomposition.

11 · Search Console Impression-Weighted CTR & Position (Relational Analytics)
"Using gsc_queries.csv in this folder, what is our overall CTR (%) and average search position, and what is our biggest page-2 (position 10-20) opportunity query?"
Trap: Unweighted AVG(ctr) and AVG(position) distort SEO performance vs. impression-weighted averages.
Guard: Non-additive rate/position guard flags unweighted AVG(ctr) / AVG(position).

12 · Survivorship-Biased Customer LTV (Relational Analytics)
"Using the customer files in this folder (all_customers.csv, active_customers.csv), what is our average customer lifetime spend?"
Trap: Querying only active_customers.csv excludes churned cohorts and severely overstates true LTV.
Guard: Multi-file profile(dirPath) + peer dataset inspection surfaces both tables and survivorship gap.

13 · 12-Segment 54/46 SRM + P-Hacking (Hard Data Science)
"Using experiment_results.csv in this folder, did any segment win our A/B test and should we ship?"
Trap: Broken 54/46 traffic split (SRM) plus 12 segments where raw p < 0.05 winners disappear under Holm-Bonferroni.
Guard: 1-call analyze(ab_test) runs automatic χ^2 SRM + Holm-Bonferroni correction on Turn 1.

14 · Subtle 52.5/47.5 SRM Across 20 Geos (Hard Data Science)
"Using ab_20seg.csv in this folder, did any geo win our A/B test and should we ship?"
Trap: Subtle 52.5/47.5 imbalance escapes eyeballing while 1 of 20 geos shows raw p = 0.015 noise.
Guard: 1-call analyze(ab_test) catches subtle χ^2 SRM (p < 1e-4) and rejects false-positive geo under Holm.

15 · Confounded Discount Causal Reversal (Hard Data Science)
"Using account_renewals.csv in this folder, our VP says giving discounts hurts renewals. What do the numbers show? Quantify the effect."
Trap: Discounts target at-risk accounts: naive group comparison shows −19.1 pp drop while controlled OLS lift is +8.0 pp.
Guard: 1-call analyze(regression) fits controlled multivariate OLS and flags CONFOUNDING / SIGN REVERSAL.

16 · Right-Censored Cohort Retention (Hard Data Science)
"Using user_activity.csv in this folder, what is our M3 cohort retention rate and how is it trending across recent cohorts?"
Trap: The 3 newest monthly cohorts are not yet 3 months old — averaging them as 0% fabricates a retention cliff.
Guard: Pro SQL audit routes to analyze(cohort_retention) which masks immature cells as null (~40% flat).

17 · Seasonal Structural Break (Weekend Masking) (Hard Data Science)
"Using daily_orders.csv in this folder, how are daily orders trending — did anything shift, and if so when and by how much?"
Trap: 62% weekend order drops hide an exact −16.5% structural break starting 2026-02-18.
Guard: 1-call analyze(anomalies) removes day-of-week seasonality and pinpoints 2026-02-18 (−16.5%).

18 · Whale-Skewed Revenue A/B Test (Hard Data Science)
"Using ab_revenue.csv in this folder, did the treatment lift revenue per user enough to ship?"
Trap: 3 extreme whale orders ($12,500+) create a fake +$12k revenue lift that vanishes when top 1% is trimmed.
Guard: 1-call analyze(ab_test) runs automatic top-1% trimmed Welch t-test and warns not to ship.

Appendix B — Representative External Benchmark Questions

MIT KramaBench (ICLR 2026) — Sample Multi-File Data-Lake Questions (github.com/mitdbg/Kramabench)

environment-easy-1 · Environment · 36 CSVs
"What percentage (to 3 decimal places) of water samples collected from Massachusetts beaches during the 2013 bathing season exceeded bacterial standards?"
Expected: 4.796

environment-hard-20 · Environment · Multi-file join
"In 2015, what are the three most polluted beaches of the city that had the least rainfall in the summer (June, July, August)?"
Expected: ["Bucks Creek", "Pleasant Street", "Forest Beach"]

biomedical-hard-1 · Biomedical · 7 .xlsx workbooks
"What is the Spearman correlation between the abundance of the protein PLK1 and the abundance of CHEK2-S163 in tumor samples? Exclude samples not in the study and with missing values."
Expected: 0.4765

biomedical-easy-9 · Biomedical · Multi-sheet .xlsx
"What is the difference between the average false discovery rate (FDR) in CBX3 genes and the average FDR for the rest of the genes?"
Expected: −0.008003975

archeology-hard-1 · Archaeology · Preamble-row .xlsx
"What is the average Potassium in ppm from the first and last time the study recorded people in the Maltese area? Assume Potassium is linearly interpolated between samples."
Expected: 8577.5298

archeology-hard-7 · Archaeology · CSV + .xlsx
"How many radiocarbon samples belong to the Neolithic period in the regional radiocarbon database?"
Expected: 274

wildfire-easy-1 · Wildfire · 17 CSVs + JSON
"Find the 3-month period with the highest total acres burned since Jan 2000, according to NOAA. What was the total acres burned in that period?"
Expected: 7805421

wildfire-hard-21 · Wildfire · Multi-table ranking
"Based on NOAA data, what are the top 3 states that lost the most residential property in value between 2005 and 2010 (inclusive)?"
Expected: ["California", "Washington", "Idaho"]

Adyen / Hugging Face `adyen/DABstep` — All 10 Official Labeled Dev Questions

task-5 · Easy · Ledger aggregation
"Which issuing country has the highest number of transactions?"
Expected: NL

task-49 · Easy · Fraud rate
"What is the top country (ip_country) for fraud? A. NL, B. BE, C. ES, D. FR"
Expected: B. BE

task-70 · Easy · Manual rule check
"Is Martinis_Fine_Steakhouse in danger of getting a high-fraud rate fine?"
Expected: Not Applicable

task-1273 · Hard · Nested JSON fee rules
"For credit transactions, what would be the average fee that the card scheme GlobalCard would charge for a transaction value of 10 EUR?"
Expected: 0.120132

task-1305 · Hard · MCC + Fee rule join
"For account type H and the MCC description: Eating Places and Restaurants, what would be the average fee that GlobalCard would charge for a 10 EUR transaction?"
Expected: 0.123217

task-1464 · Hard · Wildcard rule matching
"What is the fee ID or IDs that apply to account_type = R and aci = B?"
Expected: 391 matching fee IDs (comma-separated)

task-1681 · Hard · Merchant monthly volume + fraud
"For the 10th day of the year 2023, what are the Fee IDs applicable to Belles_cookbook_store?"
Expected: 741, 709, 454, 813, 381, 536, 473, 572, 477, 286

task-1753 · Hard · Monthly rule set (34 IDs)
"What are the applicable fee IDs for Belles_cookbook_store in March 2023?"
Expected: 34 matching fee IDs

task-1871 · Hard · 14-decimal fee counterfactual
"In January 2023 what delta would Belles_cookbook_store pay if the relative fee of the fee with ID=384 changed to 1?"
Expected: −0.94810300000017

task-2697 · Hard · Fraud ACI fee optimization
"For Belles_cookbook_store in January, if we moved fraudulent transactions to a different ACI, which ACI minimizes total fees and what is the cost?"
Expected: E:13.57

Appendix C — Direct Warehouse Questions and Their Traps

median_order · A median over 606k orders

“What was the median order amount for completed orders placed by North America customers in 2026, and what share of those orders were over $500?”

Trap: Needs all 605,766 rows. A 100-row command-line preview or a 5,000-row sample gives a believable but wrong share.

ship_fanout · A three-table join that double-counts

“For completed orders placed in Q3 2026 by North America customers that were shipped (at least in part) by Bluejay Freight, what was total order revenue by customer segment, and the grand total?”

Trap: Orders, customers and shipments. An order can have up to three shipments, so a plain join to shipments overstates revenue by 19%.

partial_month · A 12-day month

“Show monthly completed order revenue for North America customers from January 2026 through the latest month in the data, and explain how the latest month compares with the month before.”

Trap: The data stops on November 12. November looks 61% down on October; per day it is flat.

null_promo · Missing values a filter drops

“How many completed 2026 orders from North America customers did not use the SAVE10 promo code, and what was their total revenue?”

Trap: Most orders have no promo code (NULL), and promo_code != 'SAVE10' silently drops them: 84,565 orders instead of 520,653.

weighted_rate · An average of daily rates

“Using campaign_daily for Q3 2026: what was the conversion rate (conversions ÷ sessions) and cost per conversion for each channel, what was the conversion rate across all channels combined, and which channel converts best?”

Trap: Tiny affiliate campaigns convert 12–20% a day. Averaging the stored daily rate puts Affiliate first at 14.6%; weighted by sessions it is 2.3%, and Search leads.

fx_revenue · Three currencies in one column

“What was Acme’s total completed order revenue in October 2026, in US dollars, and how much came from each region?”

Trap: EMEA amounts are in euros and pounds. Converting needs the monthly fx_rates table; a plain sum is 4% low.

cohort · A first-order cohort over 2.7M orders

“Of the customers whose first-ever order was placed in Q1 2026 (January–March), what percentage placed at least one more order in Q3 2026 (July–September)? Count orders of any status.”

Trap: Needs each customer’s first order across the whole history. Counting everyone who ordered in Q1 gives 31.7% instead of 39.6%.

top_share · The top 1% of 119k customers

“Among North America customers who placed completed orders in 2026, what share of their 2026 completed revenue came from the top 1% of those customers by 2026 revenue?”

Trap: Needs a revenue total for each of 119,478 customers, far more rows than an agent can read.