MUDITEK

● Technical Guide

```
$ claude-agent --mode data --source dbt --output slack
```

How to Build an AI Data Agent

The exact architecture, stack, and implementation guide behind replacing 4 analyst hires with one AI agent. From the engineer who built Notion's data stack.

5x	4→1	3wks	$0
TEAM BANDWIDTH INCREASE	ANALYST HIRES REDUCED	END-TO-END BUILD TIME	SEMANTIC LAYER MAINTENANCE

LLM: Claude Opus 4.6 **Data:** dbt + Snowflake **Vector:** pgvector **Embed:** text-embedding-3-small **Interface:** Slack + Notion

Curated by Ghiles Moussaoui · ghiles@muditek.com

Source: Jamie Quint (@jamiequint) · Built Notion's data stack, 2020

THE PARADIGM SHIFT

From Analyst Queues to Instant Answers

The modern data stack made data accessible to analysts. The AI data stack makes data accessible to everyone — by replacing the human translation layer with agents that go from question to answer automatically.

THE OLD WAY		THE NEW WAY
1. Make data accessible (Fivetran, Airbyte)		**1.** Same infra (Fivetran, dbt, warehouse)
2. Organize it (dbt)		**2.** AI agent = the translation layer
3. Store it (Snowflake, BigQuery)	→	**3.** Question asked in Slack
4. Hire analysts to write SQL, build dashboards		**4.** Sub-agent investigates data landscape
5. Wait for the analyst queue		**5.** Agent writes SQL, runs it, returns answer
6. Get the answer next sprint		**6.** Self-heals from corrections
Bottleneck: the human translation layer		**Bottleneck: eliminated**

Key Concept

The semantic layer is dead. Context management replaces it. Instead of a static, hand-maintained mapping between business concepts and SQL, the agent reads your dbt models on-the-fly and builds its own brief per question.

Why This Matters

The semantic layer was the "20% of the data stack that was 80% of the pain." It broke every time the data model changed. Analysts spent more time maintaining the map than answering questions. With context management, the agent traces ref() dependencies through the DAG, reads transformation logic, checks business rules, and builds a structured brief — dynamically, at query time.

Why Most AI Data Projects Fail

They bolt AI onto broken workflows. They build a static semantic layer that breaks on every schema change. They skip diagnosis and wonder why the agent hallucinates. Diagnosis first — figure out which functions are automatable BEFORE building.

THE BUILD

7-Step Architecture Guide

The entire system took ~3 weeks end-to-end, including Slack/Notion integrations and the admin dashboard. Here's the exact build sequence.

1 **Set Up the Agentic Loop**

Use Claude Agent SDK. Opus 4.6 crushes Codex 5.3 xhigh in data analysis despite losing on pure coding tasks. Pick the right model for the job.

2 **Connect Your dbt Repo + Codebase**

Make your dbt repo and application codebase(s) accessible from the agentic loop. The agent needs to read raw transformation logic directly — not a pre-built summary.

3 **Annotate Base Models with Metadata**

Add links from dbt base models (stg_*) to application code + high-level descriptions. Mostly automatable. This is what lets the agent trace business logic back to the source.

4 **Build the Context Sub-Agent**

Runs BEFORE SQL generation. Reads model files, traces upstream dependencies, checks application code, returns structured brief: tables, columns, join paths, filters, dedup rules, caveats. This is the critical step most teams skip.

5 **Build the Knowledge Store ("Quirks")**

When a user corrects the agent ("that metric needs this filter"), the correction gets stored as durable, reusable knowledge. Semantic layer rebuilds itself from usage. Hybrid retrieval: vector similarity (pgvector) + BM25 keyword search (pg_textsearch).

6 **Add Core KPI Definitions**

Human-authored metric definitions for core KPIs go into the same knowledge store. "The 20% of the semantic layer that was actually useful" — everything else is auto-generated from context.

7 **Wire Up Slack + Admin Dashboard**

Multi-threading in Slack for agent conversations. Admin UI for monitoring queries, editing the knowledge store, reviewing/approving learned corrections. This is your control plane.

UNDER THE HOOD

Tech Stack & Self-Healing

Claude Opus 4.6 PRIMARY AGENT LLM	**Claude Agent SDK** AGENTIC LOOP	**dbt** DATA TRANSFORMATION
Postgres + pgvector VECTOR KNOWLEDGE STORE	**OAI Embed 3-small** EMBEDDINGS	**pg_textsearch** BM25 KEYWORD SEARCH
Claude Haiku SEMANTIC SCORING	**Slack** USER INTERFACE	**Notion** ANSWER STORAGE

SELF-SCORING & RECOVERY

Every Query Gets Scored

The agent doesn't generate SQL and hope for the best. Every query runs through a scoring and recovery loop that catches errors before the user sees them.

DIMENSION	WHAT IT CHECKS	METHOD
Structural	Does the SQL parse? Correct table/column names?	Deterministic validation
Execution	Does it run without errors? Reasonable results?	Test execution + sanity checks
Context	Does the answer match what the user asked?	Haiku semantic check + fallback

RECOVERY FLOW

Step 1: Build a context-gap brief from scored queries, breaking the question into sub-questions.

Step 2: If unresolved gaps remain → retry with targeted prompt.

Step 3: If schema/context mismatch → rerun context enrichment first.

Step 4: Human corrections get stored as "quirks" → never repeats the same mistake.

RETRIEVAL TUNING

Knowledge Store Config

Four tuning mechanisms keep retrieval quality high:

Collection weights — balance between metric definitions vs. learned quirks.

Reviewed-item multiplier — human-approved knowledge ranks higher than auto-learned.

Reciprocal-rank fusion — blends vector similarity and BM25 keyword candidates for best-of-both retrieval.

Fixed context budget — prevents stuffing the entire knowledge store into the prompt. Only the most relevant items make it in.

THE OUTCOME

Results in Production

Before → After

Hiring plan: 4-5 data analysts → reduced to 1.

Bandwidth: 5x increase across the data function.

Self-serve teams: Sales, CS, Product, Growth — all answering data questions directly in Slack.

Build time: ~3 weeks end-to-end including integrations.

Who Uses It and How

Sales team → asks questions in Slack, gets answers with SQL and charts in seconds. No tickets.

Customer Success → self-serves account health data without filing a request.

Product & Growth → gets answers instantly, saves analysis to Notion for the team.

The remaining analyst → focuses on complex, judgment-heavy analysis. The work that actually needs a human.

KEY TAKEAWAYS

01 **Kill the semantic layer.** Replace it with context management — agents that read your dbt models on-the-fly, per question. Dynamic > static.

02 **Context sub-agent is critical.** It investigates BEFORE writing SQL. Most teams skip this step and wonder why the agent hallucinates.

03 **Self-healing from corrections.** The semantic layer should rebuild itself from usage, not from analyst maintenance. Store every correction as reusable knowledge.

04 **Score every query on 3 dimensions.** Structural, execution, context alignment. Self-healing beats prompt-and-pray.

05 **Pick the right model.** Opus 4.6 > Codex 5.3 for data analysis. Don't default to the "best coder" — default to the best analyst.

06 **Diagnosis first.** Figure out which analyst functions are automatable BEFORE building anything. Not all tasks can or should be replaced.

Want to know which ops roles AI can replace in your org?

I diagnose which functions are bleeding money, then build the AI system to plug them.
No guessing. No "let's try AI and see what happens."

ghiles@muditek.com

Source: "How to Build a Data Agent in 2026" by Jamie Quint (@jamiequint).