



# Context Engineering for the Data Engineer

How to fix the AI-on-data failure mode, translate schemas for LLMs, and build the foundation that empowers your analysts.

A blueprint based on the DT + DX + CTX framework.

# Before We Start



Slides, recording, and transcription will be shared this week.



Put your questions in the chat window; we will answer at the end.



50-minute presentation followed by a Q&A session.



**Christopher Bergh**

CEO, Head Chef  
*DataKitchen*

# Context Engineering Caveats



Context Engineering Is **A New Discipline**



DataKitchen Has Done Context Engineering In Only Two Projects



Ideas And Technology Are Changing Rapidly



**Christopher Bergh**

CEO, Head Chef  
*DataKitchen*

# Your AI chatbot is lying to you. Not maliciously. Confidently.

⚠ It queries your data warehouse and returns an answer in the exact same tone regardless of data correctness. Every 'hallucination', every wrong answer, every time an analyst says "I don't trust what the chatbot told me" — that is a data infrastructure failure dressed up as an AI failure.

- The model is fine.
- The data underneath it is the disaster.

# The MIT BEAVER Benchmark Proves the Point

**85%+**

GPT-4 accuracy on standard public SQL benchmarks

**~0%**

End-to-end execution accuracy when tested against real, undocumented enterprise data warehouses

When researchers simply handed the models the correct table and column context, performance recovered sharply. The intelligence was there the whole time. The context disappeared.

# OpenAI's Internal Agent Required Context to Scale

**600+ PB**

of data across  
70,000+ datasets  
processed by their  
internal data agent.

**3,500+**

internal users served  
reliably across all  
organizational  
functions.

**22 min → 90 sec**

query time reduction  
when long-memory  
context was injected.

Raw data alone wasn't enough. The same metric means different things depending on business context. Combining data with context turned an ambiguous reporting tool into a reasoning engine.

# The Empirical Proof: Models Fail Without Context

**MIT BEAVER |**  
Sept 2024

**85% to ~0%**  
accuracy drop on real data

GPT-4 scored 85%+ on public benchmarks. On real enterprise warehouse data: nearly 0% end-to-end accuracy. When given the correct tables and column context, performance recovered sharply.

**OpenAI In-House Data Agent |** Jan 2026

**4,000+**  
employees served  
after adding context

OpenAI built a custom context layer -- metadata, human annotations, code-derived enrichment -- before their agent could reliably answer questions across 70,000 datasets and 600+ petabytes.

**arXiv Text-to-SQL Survey |** 2024-2025

**#1 Factor**  
schema context  
drives accuracy

Across dozens of studies, schema linking and domain context are the single biggest drivers of SQL generation accuracy. Without them, even the best models guess wrong tables and wrong joins.

**ACM Computing Surveys |** 2025

**Significant**  
degradation without  
domain context

Performance degrades significantly without extra information on SQL knowledge and external commonsense. Domain-specific jargon and hidden business concepts cannot be resolved without explicit context.

# LLMs are like infinitely energetic interns

## The Problem: LLMs lack business reality

Fast. Helpful. Totally capable of saying something wildly wrong with confidence.

### Without context, an LLM cannot tell:

- Which tables matter
- What “good” looks like in your business
- When a plausible answer is actually nonsense



**Context engineering turns a “smart 20 year old” into grounded decision support.**

**Infinite intern energy. Zero business judgment. Context is the adult in the room.**

# The Enterprise Data Hurdle



## 01. Schemas Too Complex

Enterprise databases have hundreds of tables built up over years. LLMs cannot navigate them without knowing which tables matter, how they join, or what columns mean.



## 02. Business Questions Too Hard

Real queries require multi-table joins, nested aggregations, and domain-specific logic. Annotators writing benchmark questions never faced this complexity.



## 03. Models Never Saw Your Data

Public LLMs train on public data. Enterprise data warehouses are private. The model has absolutely no exposure to your private schemas, your terminology, or your business rules.

# The Three-Part Infrastructure Failure

## 1 Your team doesn't trust the data

A team that has been burned by bad data will second-guess every AI answer. Without tested, monitored pipelines, the AI just amplifies bad answers at speed.

## 2 The schema is a graveyard

150 tables, half of them wrong for the question. Staging tables, deprecated views, experimental datasets. The AI guesses wrong because nobody deletes anything.

## 3 The AI has no idea what your business means

"Equalized prescriptions." "Territory alignment." These concepts aren't in the public training data. Without an engineered semantic layer, the AI is always guessing at domain language.

# The Framework for AI Readiness

$$\text{DT} + \text{DX} + \text{CTX} = 10x$$

## ✅ DT (Data Trust)

Tested, monitored,  
fresh data delivered on  
time.

The absolute  
prerequisite.

## 📊 DX (Data Experience)

Curated schemas.  
10 tables that matter  
instead of hundreds of  
tables that confuse.

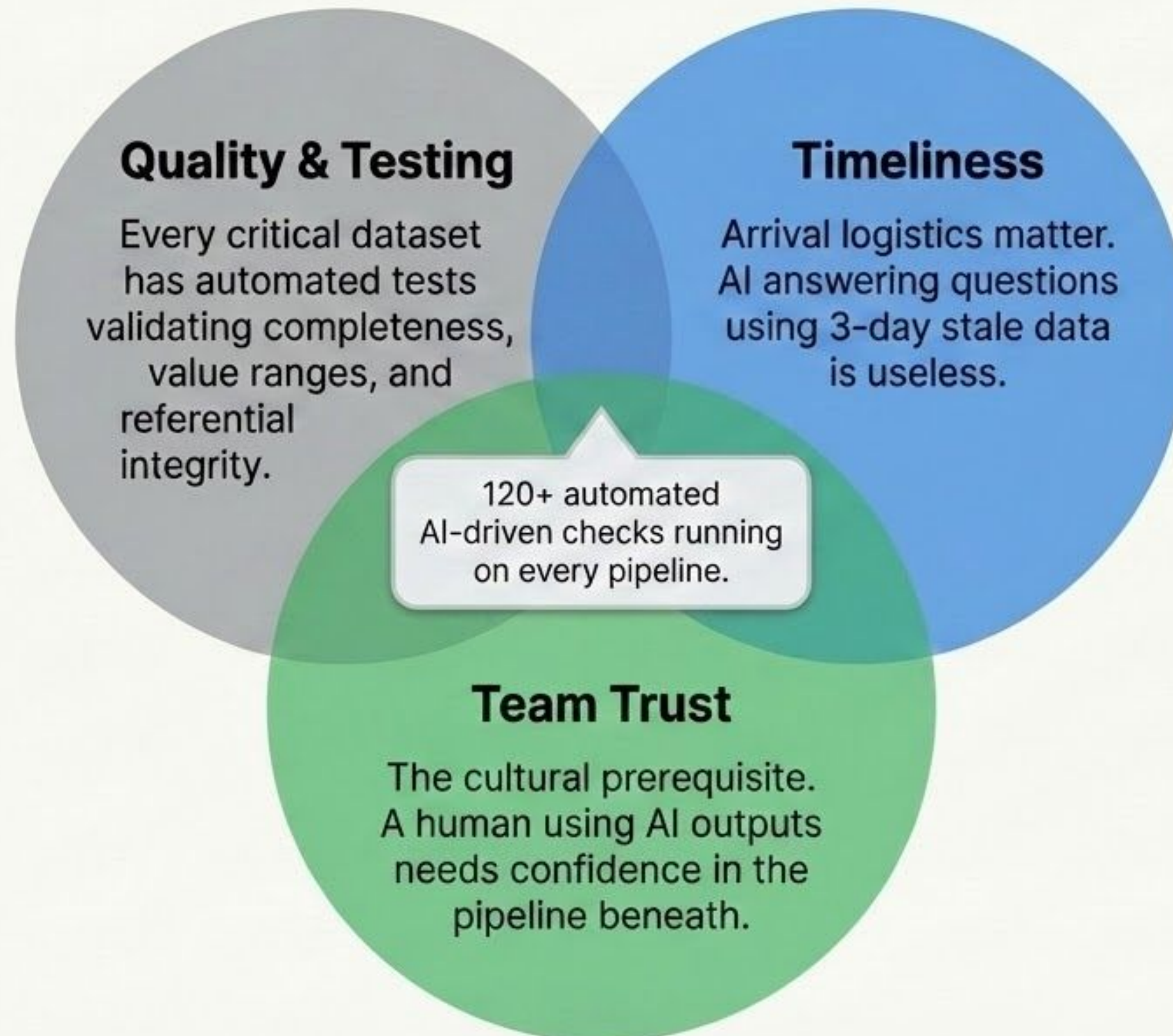
Consistent  
representation.  
The legible foundation.

## 🧠 CTX (Context)

Business definitions,  
example queries, and  
operational history.

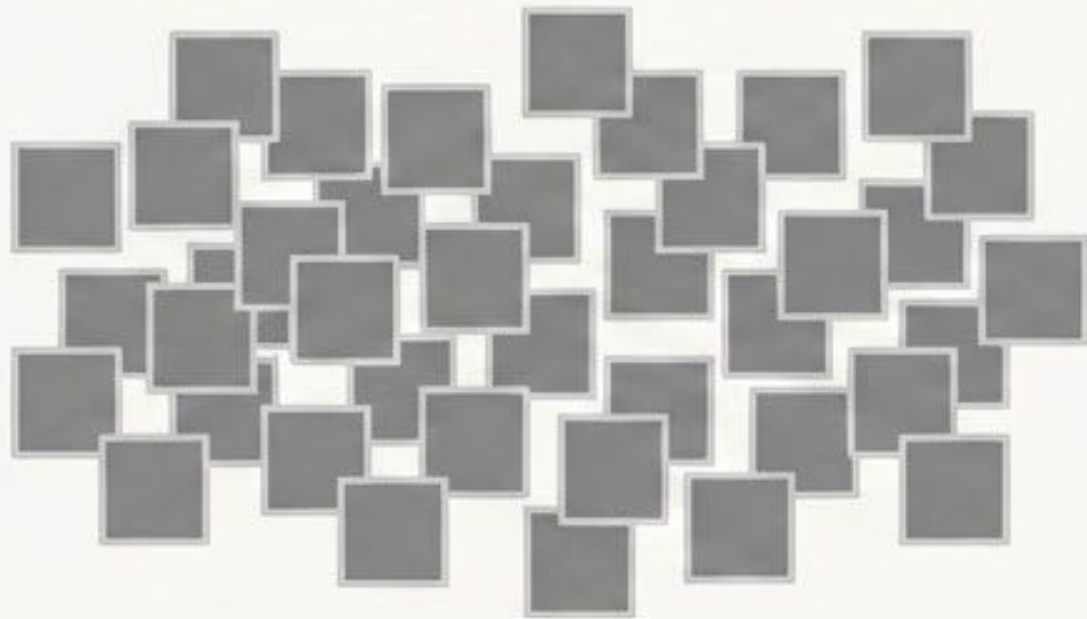
The multiplier that turns  
a generic LLM into a  
domain expert.

# DT (Data Trust) is Not Perfect Data. It is Monitored Data.



# DX (Data eXperience) Solves the "Hundred-Table Graveyard"

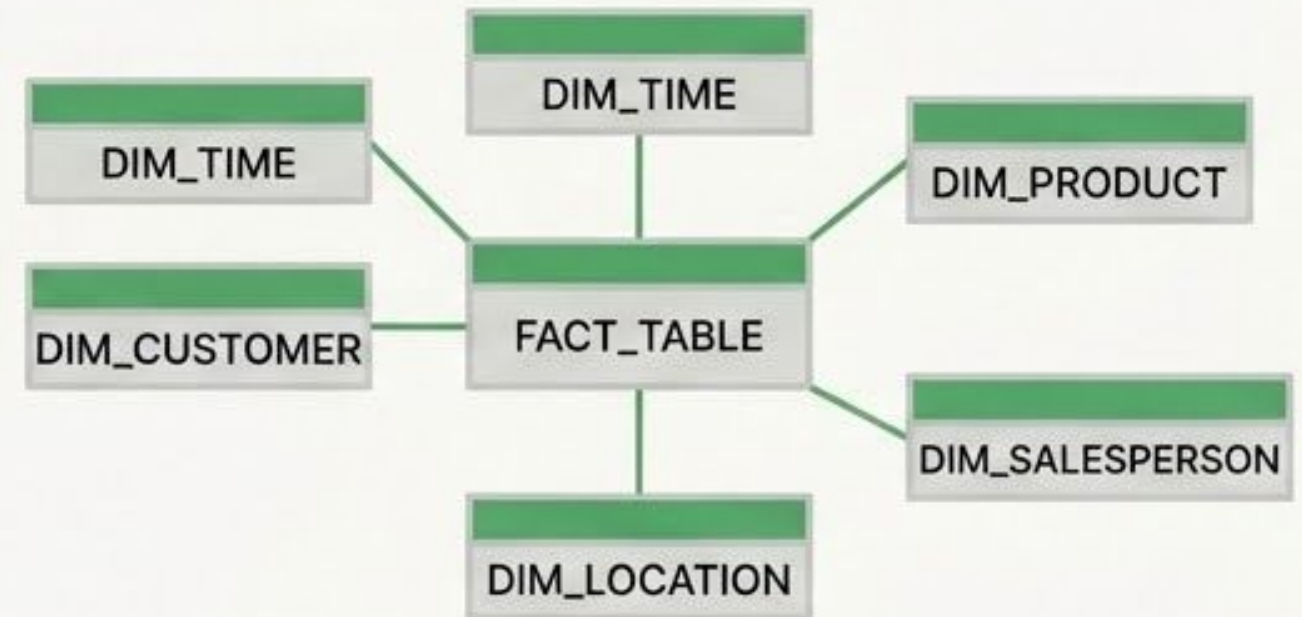
## The Warehouse Reality



150+ tables (staging, deprecated, raw).

The AI guesses. It joins technically related but semantically wrong tables, queries deprecated tables (FACT\_TRX\_PROD\_BACKUP), and doubles rows with bad joins.

## Curated Schema

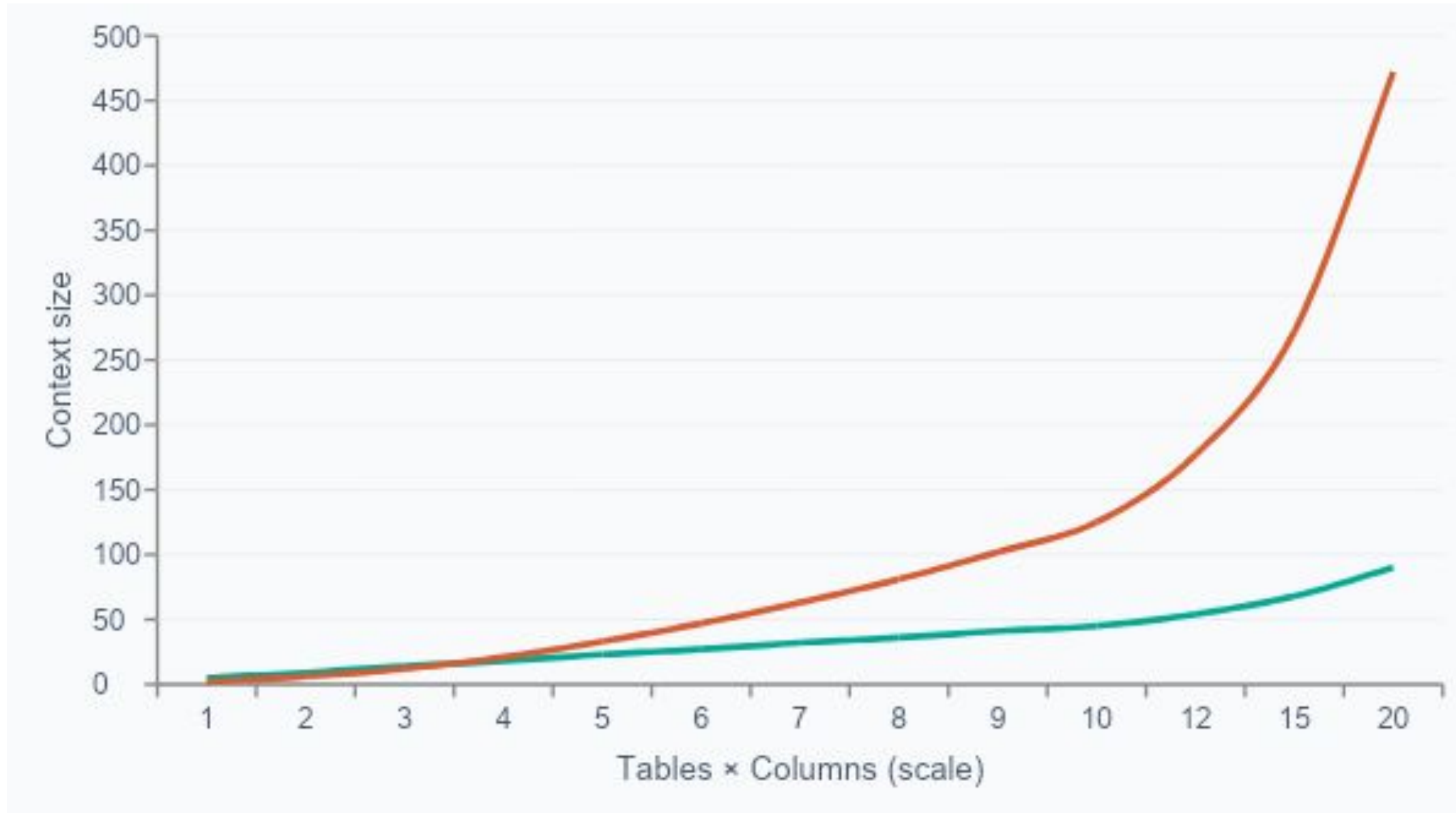


10 tables that matter.

The AI stops guessing and starts reasoning. A curated front door gives the AI a focused, strictly bounded space to operate.

# Context Size Grows Non-Linearly With Number Of Tables

*More tables × more columns = exponentially more context to write and maintain*



Linear (what people expect)

Actual context size

## Why the curve bends

$$\text{Join pairs} = T \times (T-1) \div 2$$

5 tables → 10 join pairs

10 tables → 45 join pairs

20 tables → 190 join pairs

*Each pair needs a plain-language explanation for the AI*

$$\text{Column defs} = T \times C$$

10 tables × 20 cols = 200

20 tables × 30 cols = 600

*Each needs business meaning, caveats, and usage guidance*

This is why DX matters — curating from hundreds of tables down to 10 collapses the context configuration burden dramatically.

# Curating the Front Door

✗ DIM\_HCP\_XREF\_V2



✓ PRESCRIBER\_MASTER

(One row per prescriber. Includes NPI, specialty, territory).

✗ FACT\_TRX\_PROD\_V3



✓ WEEKLY\_PRESCRIPTIONS

(One row per prescriber × product × week).

✗ TERRITORY\_MAP\_Q3\_FINAL



✓ TERRITORY\_ALIGNMENT

(Current alignment only. History in separate table).

✗ HCP\_MASTER\_FINAL\_v3



✓ PRESCRIBER\_TARGETS

(Annual targets by territory and product group).

This isn't hiding data. It's creating a front door—and it is strictly data engineering work, not AI work.

# CTX is the Multiplier

**Without Context**, the AI sees:

```
PRODGRP_NO VARCHAR(9)
HCP_ID NUMBER
EQ_TRX NUMBER(38,6)
```

*Column names with no business meaning. The AI makes educated guesses dressed up as answers.*

**With Context**, the AI knows:

```
PRODGRP_NO: Product Group
Number. Join through NDC
instead for accurate product
mapping.
HCP_ID: Business Partner ID.
This is the prescriber; join to
D_PRESCRIBER for names.
EQ_TRX: Equalized Total
Prescriptions. Normalized for
days supply; primary volume
metric.
```

*Now the AI knows what it's looking at, how to join it, and what the values mean.*

# The Full Taxonomy: Six Categories of Context

## Foundational Layer

### 1. Schema & Structure

What the data is, physically.

### 2. Metric & Business Logic

What the data means, semantically.

### 3. Example Queries

How your business asks questions.

### 4. Data Quality & Freshness

What the AI can rely on right now.

### 5. Org & Operational Context

What was happening when data was created.

### 6. Policy & Entitlements

What the analyst is allowed to see.

## Institutional Layer

# 1 Context Category 1: Schema & Structure

## Table Purpose Statements

Not just the name, but a sentence defining its use. (e.g., PRESCRIBER\_MASTER: one row per HCP, used for territory lookups.)

## Grain Declaration

Explicitly state what one row represents. If the AI doesn't know the grain is HCP × product group × week, it will aggregate at the wrong level. Also include data profiling info

## Join Path Guidance

When multiple paths exist, explicitly state which is correct for which question type, including cardinality fan-out warnings.

## Deprecated Field Registry

Tables that exist but shouldn't be queried. The highest-leverage DX investment for AI.

## 2 Context Category 2: Metric & Business Logic

### **Canonical KPI Definitions.**

Every metric needs its filters, time windows, and edge cases defined (e.g., NRR excludes churned accounts).



### **Allowed Aggregation Grains.**

State rules explicitly: “Equalized prescriptions can be summed by territory, but cannot be averaged by rep without adjusting for panel size.”

### **Banned Metrics.**

Name metrics that look computable but are deprecated, and point the AI to the replacement standard.



### **Default Filter Logic.**

Tell the AI to automatically exclude internal accounts, test users, and pilot markets unless explicitly overridden by the prompt.

### 3 Context Category 3: Example Queries & Patterns

The most underrated category — and the highest immediate ROI per hour invested.

Q: 'Which territories exceeded prescription targets last month?'

A: JOIN WEEKLY\_PRESCRIPTIONS to PRESCRIBER\_TARGETS on TERR\_CD + PRODGRP\_NO.  
SUM EQ\_TRX WHERE WEEK\_DT IN last fiscal month. Do NOT use raw TRX.

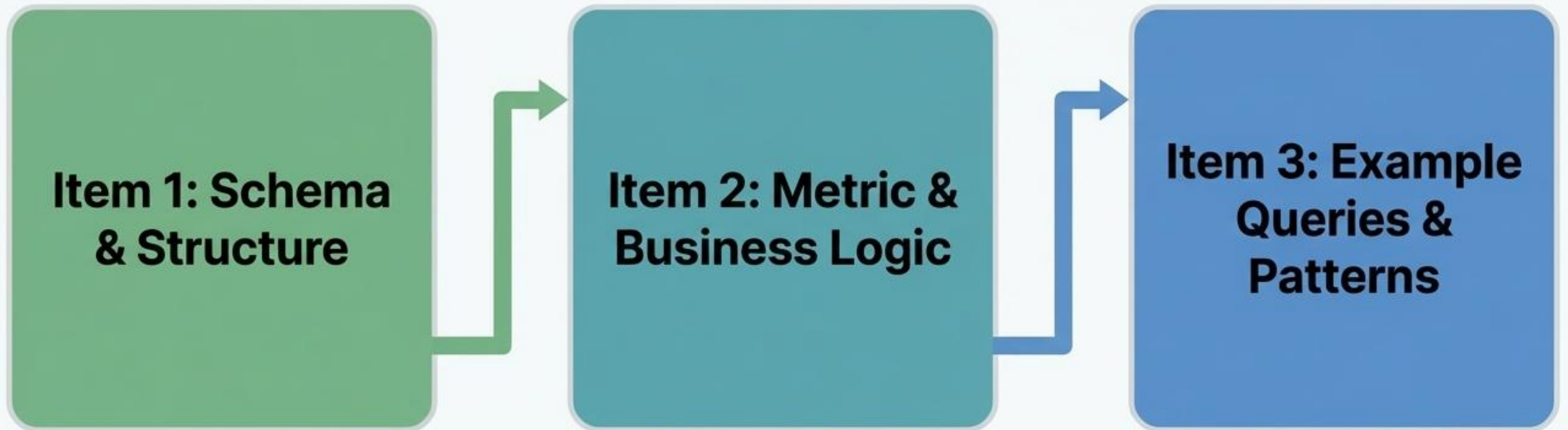
#### Common Templates

One confirmed template for territory vs target, or period-over-period trend, covers 80% of analyst queries.

#### Counterexamples

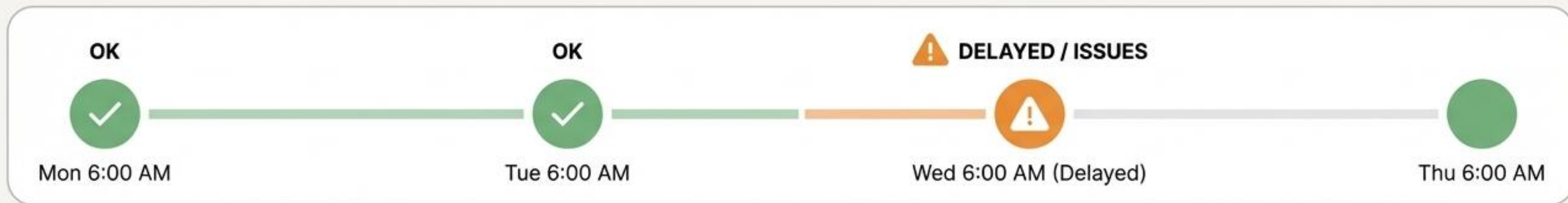
Feed the AI queries that look right but aren't, explaining why the join doubles rows.

# Categories 1-3 Form the Foundation Layer



Together, these three categories resolve the vast majority of wrong-table, wrong-join, and wrong-definition failures. If you build nothing else, build these.

## 4 Context Category 4: Data Quality & Freshness



### Pipeline Freshness Timestamps:

If the AI is answering “what happened yesterday” with 72-hour-old data, every answer is wrong.

### Known Data Quality Issues:

The AI needs to know if “TERR\_CD was unreliable for Pacific region in weeks 12-14” before it reasons over those values.

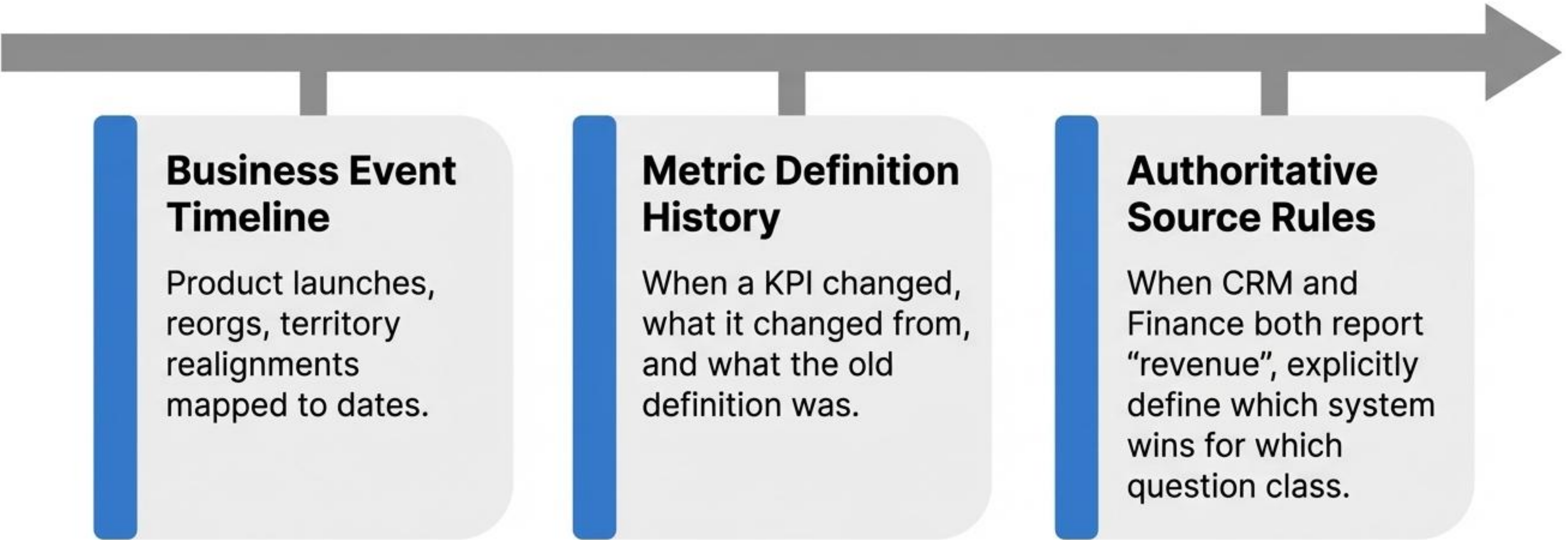
### SLA Status & Delays:

If weekly data arrives Tuesday, Monday morning answers look current but aren't.

**Every other AI system assumes trustworthy data.  
Category 4 ensures the AI actually knows whether to trust it.**

## 5 Context Category 5: Org & Operational Context

A dip in volume during a territory realignment is not the same as a dip caused by competitive pressure. Without this layer, the AI cannot tell the difference.



### Business Event Timeline

Product launches, reorgs, territory realignments mapped to dates.

### Metric Definition History

When a KPI changed, what it changed from, and what the old definition was.

### Authoritative Source Rules

When CRM and Finance both report “revenue”, explicitly define which system wins for which question class.

## 6 Context Category 6: Policy & Entitlements

### Row-Level Access & PII

- Define which columns contain personally identifiable information at the context layer, not post-hoc.
- Ensure an AI that can see all rows does not return individual-level data to a rep who only has territory-level access.

### Regulatory Constraints & Caveats

- In regulated industries, certain disclosures must accompany answers programmatically (e.g., 'preliminary data', 'not yet audited').
- Hard constraints: "Do not return prescriber-level data for controlled substances without auth."

# Categories 4–6 Form the Institutional Layer



Reporter

Item 4: Data  
Quality & Freshness

Item 5: Org &  
Operational Context

Item 6: Policy &  
Entitlements



Reasoner

While the foundation layer fixes basic schema errors, the institutional layer turns the AI from a simple SQL reporter into a true business reasoner. It separates “technically correct” from “business accurate.”

# The Artifacts of Context Engineering



## Business Definitions

Plain-language meanings for critical columns, caveats, and calculation rules.



## Example Queries

Validated SQL templates that teach the AI business querying patterns.



## Ontologies

Structured maps connecting domain rules and industry standards that don't live in the database.



## Quality Metadata

Automated profiling, lineage, and SLA statuses surfaced to the AI layer.

# Context Maintenance: Not All Layers Age the Same Way

**Slow / Tech-Driven**

**Categories:** Schema, Business Logic, Example Queries.

**Update Cadence:** Update on schema migrations or KPI changes. Low ongoing cost once seeded.

**Medium /  
Policy-Driven**

**Categories:** Entitlements.

**Update Cadence:** Follows existing RBAC and data governance policy patterns.

**Fast /  
Event-Driven**

**Categories:** Quality, Operational History.

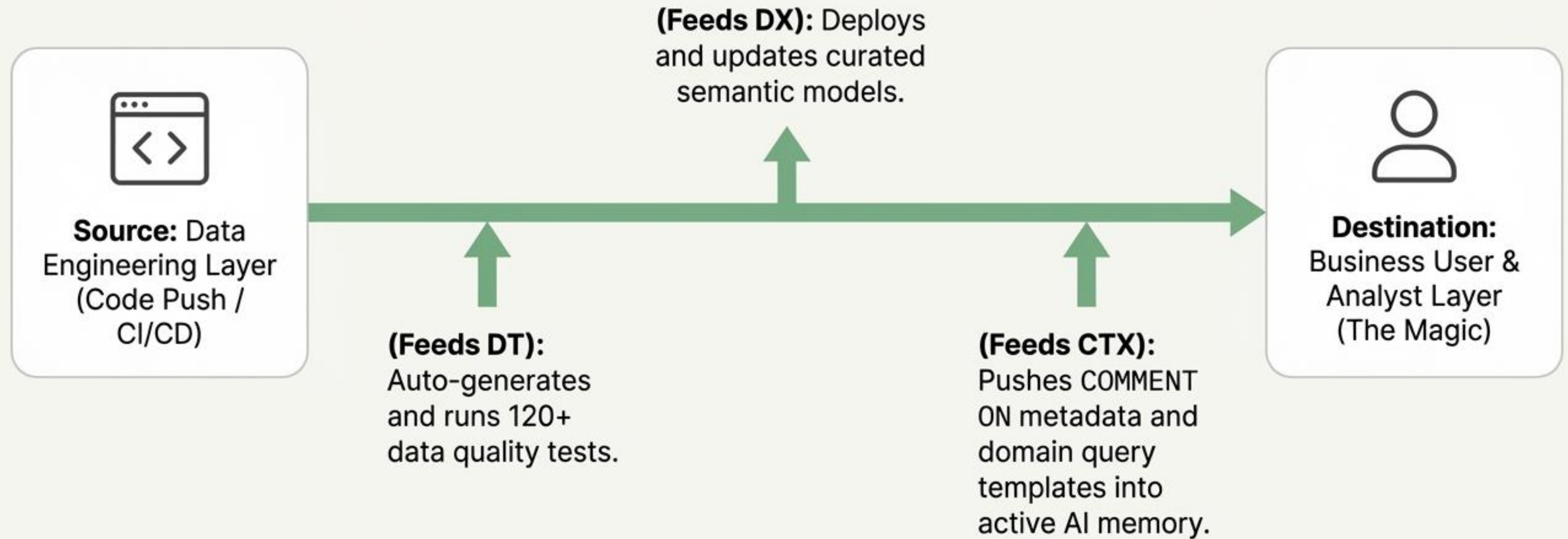
**Update Cadence:** Must be live. Quality is automated via pipeline observability. Operational history requires a cultural habit to capture business events in real-time.

# Context Engineering is Platform-Portable

The content is permanent—the platform is just where it runs. Everything maps 1:1.

|                         | Snowflake            | Databricks                  |
|-------------------------|----------------------|-----------------------------|
| Natural language agents | Cortex Agents        | AI/BI Genie                 |
| Semantic definitions    | Semantic Views       | Unity Catalog Metric Views  |
| Verified queries        | Verified Queries     | Trusted Assets              |
| AI coding assistant     | Cortex Code          | Databricks Assistant        |
| Domain skills           | Skills (auto-loaded) | Custom Skills (auto-loaded) |

# The Translation Contract



# Example Context Markdown Files

## Updated With Data

## Terminology

## Identifiers

## Audiences

## Example Queries

## Skill Description

## Domain Overview

## Metric Descriptions

# The 10x Equation Requires Progress, Not Perfection

## Start with DT

If your team doubts the numbers, run automated quality tests on critical tables first to build trust.

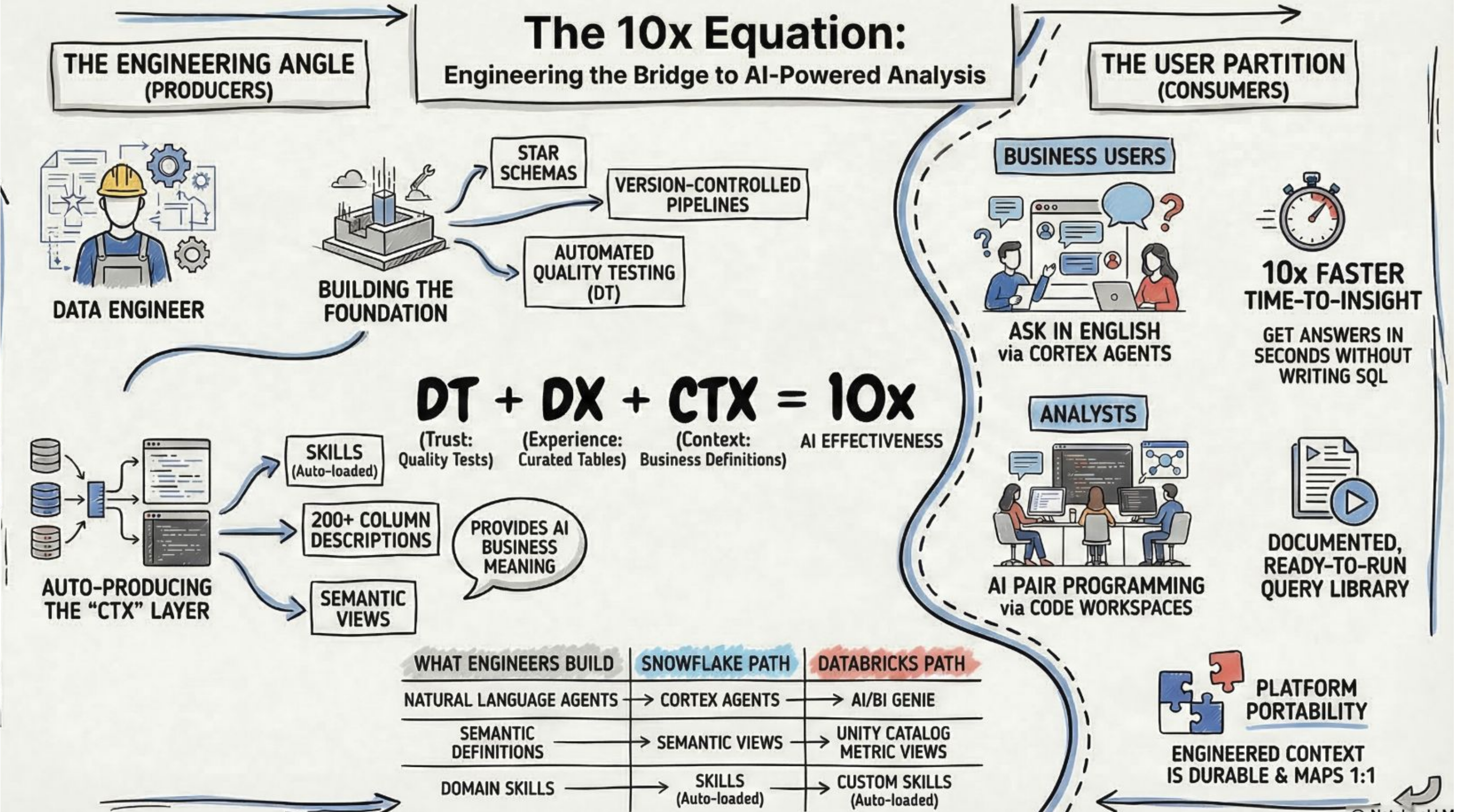
## Start with DX

If analysts are lost, identify the 10 tables that answer 80% of questions and curate them into a clean semantic layer.

## Start with CTX

If AI gives wrong answers, write business definitions and 5 example queries for your most important data points.

# THE BIG PICTURE: Overall Summary



# Team Benefits

## Business Users

- Ask questions in plain English.
- Get validated answers in seconds, self-serve.

## Analysts & Data Teams

- AI pair programmers that actually know the custom schema.
- Fewer “can you run this for me?” ad-hoc interruptions.

## Leadership

- Institutional knowledge permanently captured in the platform.
- AI adoption that actually meets users where they are, without massive shadow IT risk.

# The Bottom Line

**The teams that will win with AI are not the ones with the most sophisticated models. They are the ones who do the foundational data work that makes any model effective.**



## **DT: Data Trust**

Ensuring data is tested, validated, and delivered on time to build user confidence.

+



Hundreds of confusing tables

## **DX: Data Experience**

Curating a "front door" of 10-20 legible tables instead of 100+ confusing ones.

+



## **CTX: Context**

Adding business definitions and example queries to turn generic AI into a domain expert.

=



# DataKitchen Data (and Context) Engineering Services

## Install Open Source TestGen

Automated data quality testing for your modern data stack.

[info.datakitchen.io/testgen](https://info.datakitchen.io/testgen)

## DataOps Observability

Install open source observability for your data operations.

[docs.datakitchen.io/install-guide](https://docs.datakitchen.io/install-guide)

## The DataOps Manifesto

Join thousands of data professionals in signing the manifesto.

[dataopsmanifesto.org](https://dataopsmanifesto.org)

## DataOps Cookbook

The definitive guide to implementing DataOps in your organization.

[datakitchen.io/cookbook](https://datakitchen.io/cookbook)

## DataOps Certification

Get certified in the fundamentals of DataOps for free.

[info.datakitchen.io/certification](https://info.datakitchen.io/certification)

## Quality & Observability

Master the techniques of data quality and observability testing.

[info.datakitchen.io/data-quality-cert](https://info.datakitchen.io/data-quality-cert)

# DataKitchen Data Engineering & Context Engineering Services

## Decades of Expertise

We build systems that deliver extraordinary productivity, then hand them off fully operational.

**Build • Operate • Transfer**

*We build it with you, run it until it works, then transfer it to your team.*

## Accelerated by AI

Using tools like Claude Code to automate schema ID, testing, and pipeline profiling.

**Weeks of work reduced to hours. Speed passed directly to you.**

## The Platform & Methodology

### Dual-Focus Foundation

A single methodology built for both data engineers and analysts, not a tool one merely tolerates.

### Process Centralization

Testing, monitoring, and semantic layer curation—all connected to eliminate productivity drains.

### Trust & Insight

Engineers ship faster with fewer errors. Analysts trust what they query and spend time on value.

## The Best Foundation for AI

Good data infrastructure IS context engineering. It's exactly what LLMs need to reason accurately.

If you want AI to work on your data, build the data engineering foundation that makes it possible. **That is what we do.**

# The teams that win with AI are not the ones with the most sophisticated models.

They are the ones who do the foundational data work that makes any model effective.

**DT + DX + CTX = 10x Faster AI Data Analysis**

- Run TestGen on your most critical tables: profile, test, trust
- Identify the 10 tables that answer 80% of analyst questions
- Write business definitions for your top 20 columns
- Add 5 validated example queries to your AI context layer

## DataKitchen

DT

**TestGen**  
Open source data quality testing for any table, any platform

DX

**Semantic Layer**  
Curated schemas your AI and your analysts can both navigate

CTX

**Context Engineering**  
Business definitions, example queries, org knowledge

10x

**Services**  
Build, Operate, Transfer with Claude Code acceleration

# Opportunity to use AI to 10x the productivity of:

The Integration Ecosystem

Data Engineers who are integrating the data

**DataOps + FITT + Data Testing = 10x**

The Insight Ecosystem

Data Users directly using the data to give business insight

**DT + DX + CTX = 10x**

The architectural foundation is what makes the acceleration safe across the entire enterprise.



# 10x Your Data Engineering Productivity

## Best Practice AI With Claude Code

**DataKitchen** Webinar – Register Now

---

**April 30, 2026; 12 pm EST / 4 pm GMT**

**Register:** <https://info.datakitchen.io/webinar-registration-10x-your-data-engineering-with-ai-patterns-that-actually-work>

**Blog:** <https://datakitchen.io/dataops-fitt-data-testing-10x-data-engineering-productivity-with-ai/>