AI-powered Data Engineering with Microsoft Fabric

Wolfgang Strasser, June 2026

Microsoft® Most Valuable Professional

ACP CUBIDO Digital Solutions

Data Platform
Next Step

Premium sponsors

VENZO_

INSPARI

valantic companies

itm8®

Eddytor™

dataon

Power your data

Standard sponsors

Backstage

Columbus

TheVIT

DATA - ANALYTICS - AI

twoday

Raffle Prizes

EUROPEAN MICROSOFT FABRIC Community Conference









Servus! Hi!
Welcome!



Wolfgang Strasser

Data Consultant @ ACP CUBIDO Digital Solutions

workingondata.wordpress.com

https://www.youtube.com/@WolfgangStrasser

Microsoft® MVP Most Valuable Professional

Data Community Austria

Practice of turning raw, distributed data
into trusted, reliable, and usable data products
for analytics, AI, and business operations.

Data Engineering

80% Data Preparation vs 20% "Data Fun"

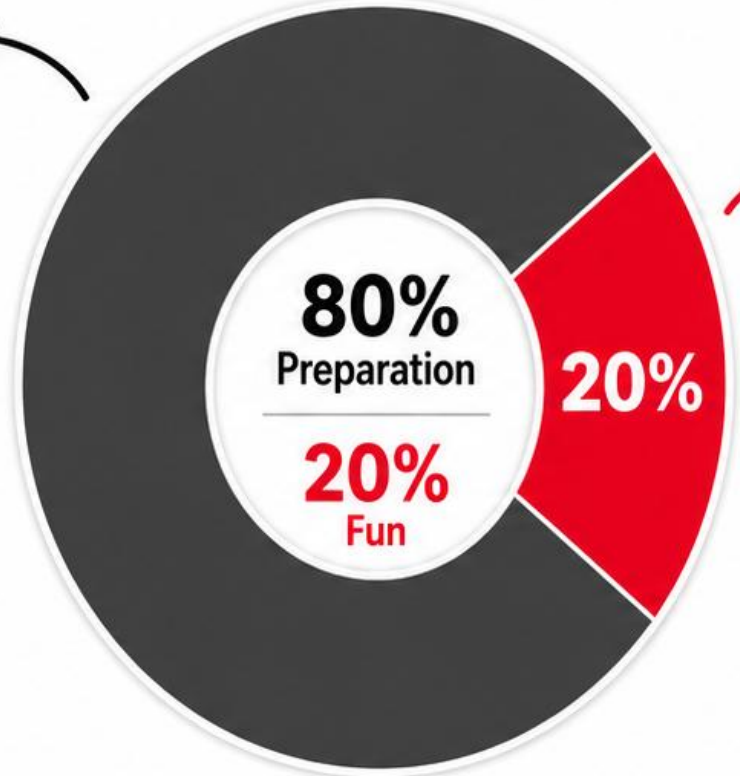
• The data engineer dilemma •

80% Data Preparation

Messy. Tedious. Time-consuming.

- Cleaning
- Joining Tables
- Wrangling
- Missing Values
- Transforming
- Broken Pipelines
- Documentation
- Repetitive Tasks
- Validation
- Fixing





20% Data Fun

Interesting. Impactful. Rewarding.

- Analyzing
- Visualizing
- Insights
- Experimenting
- Driving Impact
- Business Value
- Storytelling



Most of the effort goes into preparation before the interesting work begins.

Microsoft Fabric Data Engineering

Data Engineering in Microsoft Fabric



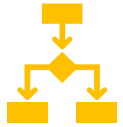
Ingest

Copy Job, Copy Activity, Dataflow Gen2, Shortcuts, Eventstream



Transform

Dataflow Gen2, Notebook. dbt Job



Orchestrate

Data Pipeline, Notebook



Stream

Evenstream



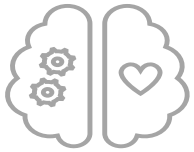
Operationalize

Pipeline scheduling, Spark Job Definitions, dbt Jobs, deployment pipelines



Storage

Lakehouse, Warehouse, Eventhouse



Knowledge

Low-Code, Pro-Code



Technology

Spark, M, SQL, KQL, …

But it’s the year 2026..
..we need to talk about AI

80% Data Preparation vs 20% "Da

• The data engineer dilemma •

80% Data Preparation

Messy. Tedious. Time-consuming.

- Cleaning
- Joining Tables
- Wrangling
- Missing Values
- Transforming
- Documenta

AI can generate a pipeline faster than you can understand the business problem.

- Visualizing
- Insights
- Experimenting
- Driving Impact
- Business Value
- Storytelling





Most of the effort goes into preparation before the interesting work begins.

Where and how does AI fit into Fabric DE ?

https://learn.microsoft.com/en-us/fabric/

Microsoft Fabric documentation

Microsoft Fabric is a unified platform that can meet your organization's data and analytics needs. Discover the capabilities Fabric has to offer, understand how it works, and how to use it.

OVERVIEW
What is Microsoft Fabric?

GET STARTED
Try Fabric for free

TUTORIAL
Develop witl

DEPLOY
Migrate to Fabric

CONCEPT
Fabric decision guides

WHAT'S NE
Fabric roadn

se types

Explore Microsoft Fabric

Learn about Fabric, evaluate its features, the experiences it offers, analyze your data and run reports.

Get started in Fabric

Start with an end-to-end view of how to get data, store it, transform it, analyze it, and share insights in Microsoft Fabric.

Copilot in Fabric

Leverage AI-powered assistance to enhance your data analysis and decision-making processes.

Power BI

Find insights, track progress
visualizations.

tion, and orchestration scenarios.

OneLake

Store all your organization's data in OneLake, a single, unified, logical data lake.

Databases

Create operational SQL databases seamlessly for transactional workloads.

Real-Time Intelligence

Discover insights from your
visualizations, and take acti

dels, and systems to power
d in a live, holistic view of the

Data Engineering

Build, transform, and share your data estate using lakehouses, notebooks, and Spark environments.

Data Science

Unlock powerful insights using AI and machine learning technology.

Data Warehouse

Analyze data in a secure SQ
in an open data format.

n Fabric for CI/CD, Git integration,
and deployment pipelines.

- Use Copilot and AI
  - Understand Copilot in Fabric
  - Learn how Copilot works in Fabric
  - Review Copilot glossary and terms
  - Enable Copilot in Fabric
  - Integrate Copilot in Power BI
  - Use Copilot with semantic models
  - Use Copilot with Power BI reports
  - Get answers to Copilot in Fabric FAQs
  - Check status of Copilot and AI features
  - Apply privacy, security, and responsible use
  - Use Copilot across experiences
    - Use Copilot for Data Factory
    - Use Copilot for Data Warehouse
    - Use Copilot for Real-Time Intelligence
    - Use Copilot for Data Engineering and Data Science
      - Use the Copilot chat panel (preview)
      - Use Copilot for Data Engineering and Data Science overview
  - Monitor Copilot consumption

AI-Assisted DE in / outside of Fabric

Inside Microsoft Fabric

- Copilot in Data Factory, Warehouse, RTI, DE / DS, SQL Endpoints, …
- AI Functions

Outside Microsoft Fabric

- AI assisted coding
  - Claude Code
  - Github Copilot (VS Code)
    - Fabric NotebookAgent
  - ….
- Automation / REST APIs
- MCP server
  - Fabric MCP
- VS Code – SQL Server projects
- SSMS - Copilot

How many copilots are there in Microsoft Fabric?

I'll verify against current Microsoft Fabric documentation, because the naming and availability of Copilot experiences in Fabric has changed over time.

Thought for 5s ›

As of Microsoft's current Fabric documentation, there are **6 named Copilot-in-Fabric workload experiences**:

1. **Copilot for Data Engineering and Data Science**
2. **Copilot for Data Factory**
3. **Copilot for Data Warehouse**
4. **Copilot for SQL database**
5. **Copilot for Power BI**
6. **Copilot for Real-Time Intelligence**

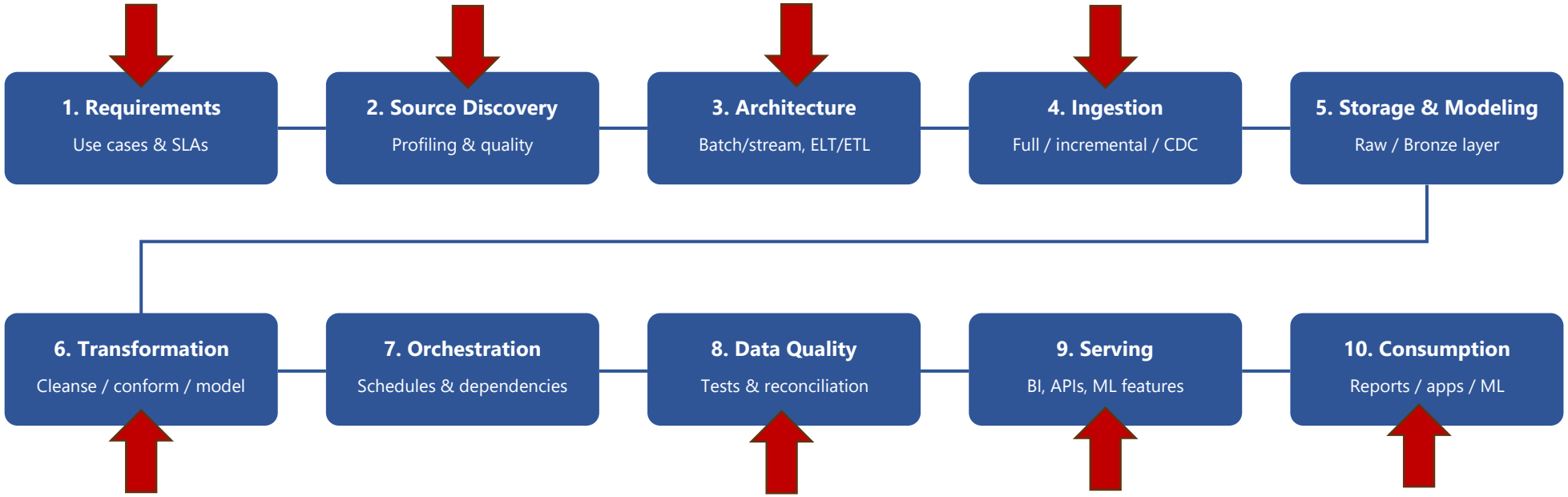
Microsoft also lists related **AI experiences** such as **Fabric Data Agent**, **AI Functions**, and **AI Services**, but those are not all branded as "Copilot for …" workload copilots.

Microsoft Learn

A useful nuance: Microsoft says "there are different Copilots in each of the Fabric workloads," and the release-status table tracks multiple Copilot/AI experiences by workload and item. Some are GA, others remain preview. Microsoft Learn +1

Typical Data Engineering Process

Linear core flow with continuous cross-cutting concerns



Cross-cutting concerns (apply throughout the lifecycle)



Requirements

Where should we begin?

Generate a data pipeline that reads from a local SAP system, re-name the tables using a meaningful naming convention, look for data errors and correct them if necessary.
Store historical changes using SCD2 and add surrogate keys.

In addition, I want you to think about a data model for financial reporting. Use the dimension-/fact based modeling approach by Kimball. I want you to help me with the measures.

And now.. Good luck & Start 😀

+ AI-Assisted DataEngine... Extended

AI makes vague thinking visible

Where should we begin?



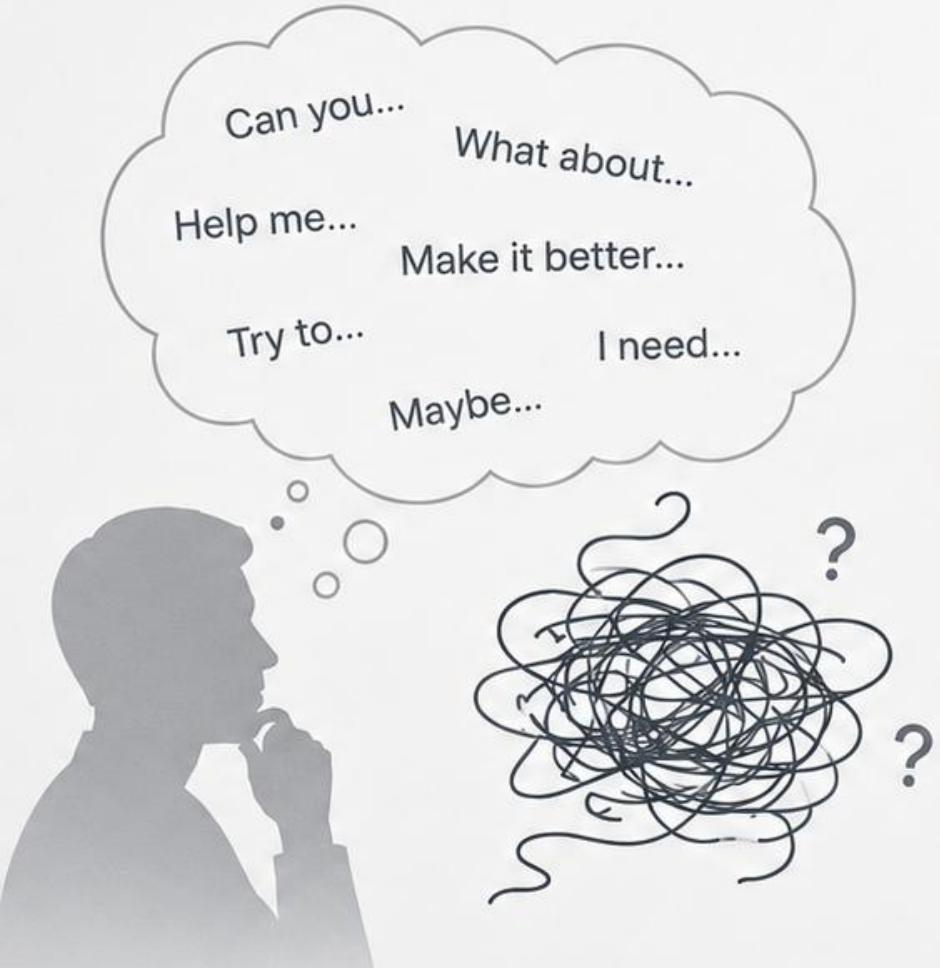
Generate a data pipeline that reads from a local SAP system, re-name the tables using a meani
naming convention, look for data errors and correct them if necessary.
Store historical changes using SCD2 and add surrogate keys.

In addition, I want you to think about a data model for financial reporting. Use the dimens
based modeling approach by Kimball. I want you to help me with the measures.

And now.. Good luck & Start 😀

+ AI-Assisted DataEngine... Extended

From “Prompting” to specifying



- Clear objective
- Defined inputs
- Constraints & rules
- Structured context
- Expected outputs



Work on my specification

prompt.txt

```
I want you to write a specification for a Microsoft Fabr

--------------
Metadata should be stored in a sql database for fabric.
Write the specs for the metadata tables:
* source information (Source system, type, server setti
* Objects to Load (tables)
    - load type (full, incremental)
    - filtering

Logging should happen in the SQL database too.
---
workspace structure
* separate data and logic workspaces
* I want a central, platform workspace
* for Power BI workspaces - model and report workspaces
* specification for permission concept (least required a
-------------

Data load architecture
* medallion architecture
* choose the implementation technology (lakehouse, wareh
* Preference for Warehouse for Gold layer
* semantic models based on gold layer
* Landingzone in /Files section in Bronze Lakehouse
* ingestion into system preferred with Copy activity / j
* data transformation logic - based and implemented in pyspark notebooks.
* data cleansing (configurable in metadata) - Bronze to Silver layer
* historization (configurable in metadata) in Silver Layer

---------------
supported data sources:
* Azure SQL
* SQL Server
* Mirrored Database for Azure SQL

---------------------

Your tasks:
* Write the specification that can be used as a foundation for the implementation.
* specs for workspace structure
* definition for data engineering framework
* spec for metadata database
* spec for logging database
* specification for pipelines (orchestration)
* specification for tranformation logic (notebooks)
* write spec to spec\specification_fabric_framework.md
```

#file:prompt.txt - work on it

prompt.txt | prompt.txt:1-49

Optimized tool selection

I'll turn the prompt into a concrete implementation-oriented specification and write it into the requested markdown file. First I'm going to inspect the workspace so I place it correctly and don't overwrite anything unexpected.

Done. I wrote the specification to specification_fabric_framework.md.

It includes the workspace structure, least-privilege permission concept, medallion architecture decisions, metadata database table specs, logging database table specs, orchestration pipeline design, notebook transformation contracts, supported sources, operational monitoring, deployment lifecycle, open decisions, and acceptance criteria.

prompt.txt | *specification_fabric_framework.md*

spec > specification_fabric_framework.md

Microsoft Fabric Metadata Based Data Engineering Framework Specification

1. Purpose

This document specifies a metadata driven Microsoft Fabric framework for standardized ingestion, transformation, historization, logging, and publication of analytical data products.

The specification is intended as a foundation for implementation. It defines the target workspace structure, data architecture, metadata model, logging model, orchestration pipelines, notebook transformation patterns, and permission concept.

2. Scope

The framework covers the following capabilities:

- Central metadata driven ingestion configuration.
- Supported source systems: Azure SQL Database, SQL Server, and Mirrored Database for Azure SQL.
- Ingestion into a Bronze Lakehouse landing zone.
- Medallion architecture with Bronze, Silver, and Gold layers.
- Configurable full and incremental loads.
- Configurable source filtering.
- Configurable Bronze to Silver data cleansing.
- Configurable historization in the Silver layer.
- Gold layer modeled in a Fabric Warehouse by preference.
- Semantic models based on the Gold layer.
- Central SQL based logging and operational monitoring.
- Least privilege permission concept across platform, data, logic, model, and report workspaces.

3. Design Principles

- Metadata first: operational behavior must be controlled by metadata, not hardcoded per source table.
- Separation of concerns: data storage, orchestration logic, semantic models, and reports must be separated into dedicated workspaces.
- Reusability: ingestion and transformation notebooks must be generic and parameterized.
- Observability: every framework run must write start, end, status, row counts, durations, warnings, and errors to SQL logging tables.
- Least privilege: users, service principals, and managed identities receive only the minimum required access.
- Idempotency: re-running a failed object load must not create duplicate target records when the configured key and load pattern support deterministic processing.
- Recoverability: failed pipeline and object runs must be restartable at batch, source, object, or activity level.
- Auditability: metadata changes and run history must be traceable.

4. High Level Architecture



Architecture / Best Practices

Architecture Discussion (Learn MCP)

I am not sure, how to deal with <topic A> or answer <questionB>

I could google it…

I could use Microsoft Learn..

Or ..

@microsoft-learn What are best practices for workspace separation?

I'll ground this in current Microsoft Learn guidance and then translate it into practical workspace separation patterns for Fabric, including where model/report split and platform/data/logic separation fit.

Reviewed SKILL.md and searched for tool_search

I found the Learn search connector. I'm going to query for Fabric workspace governance, roles, and lifecycle guidance, then fold the key practices into a concise recommendation.

Ran Microsoft Docs Search – microsoft-learn (MCP Server)

https://github.com/microsoftdocs/mcp

README | Code of conduct | CC-BY-4.0 license | MIT license | Security

Microsoft Learn MCP Server

VS Code Install Microsoft Learn MCP | VS Code Insiders Install Microsoft Learn MCP

> **Stop AI Hallucinations.** Give your AI assistant (Claude, Cursor, Copilot, Codex, ...) direct access to the latest official Microsoft documentation.
>
> ✨ **Free. One-click install. No key needed.**

🎯 Why install this?

Stop relying on outdated training data or risky web searches. Learn MCP server provides secure, direct access to Microsoft official docs.

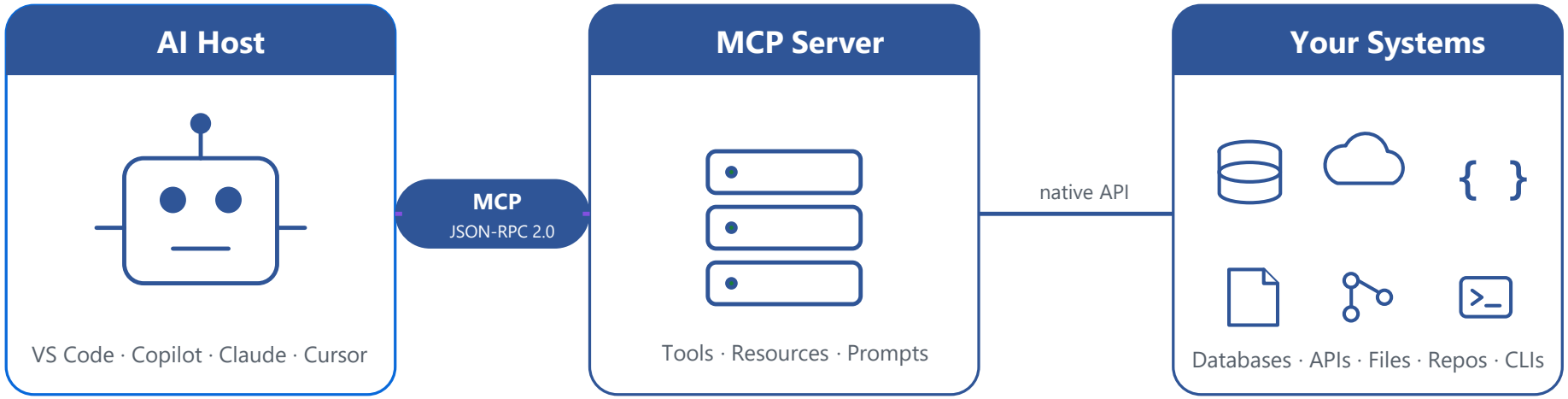
- 🧠 **Eliminate Hallucinations.** Stop your AI from inventing non-existent Azure SDK methods or hallucinating library packages. Get code that actually compiles.
- 🔌 **Plug & Play (No Auth).** No API keys, no logins, no sign-ups required. Just one-click install and start coding immediately.
- 🛡️ **100% Trusted & Safe.** Protect your supply chain. Unlike generic web searches that may scrape insecure blogs or malicious sites, this tool **only** accesses official 1st-party Microsoft documentation.
- 💸 **Completely Free.** High search capacity tailored for seamless, heavy coding sessions.

✨ Example Prompts

Your Agent (AI) and your systems

Model Context Protocol

One open protocol — many tools, many data sources, any AI app



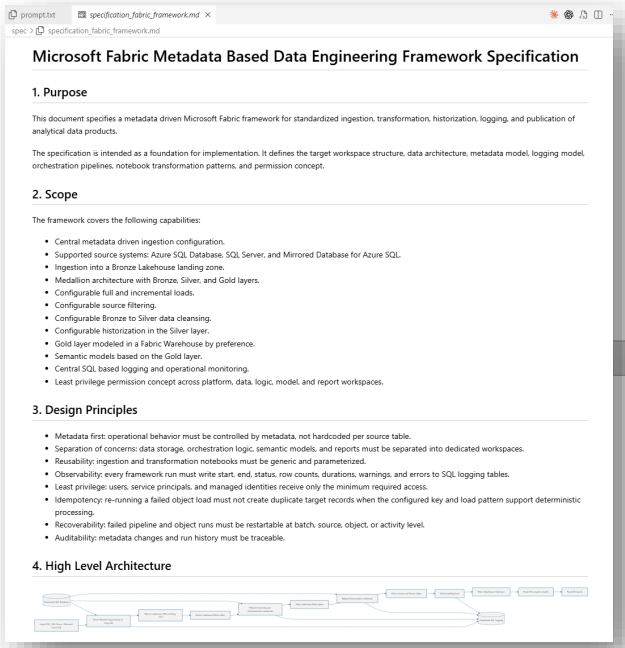
Tools
model-invoked actions

Resources
contextual data

Prompts
reusable templates

Like USB-C for AI: one connector between any host and any capability

Rate the spec (v1)



Microsoft Fabric Metadata Based Data Engineering Framework Specification

1. Purpose

This document specifies a metadata driven Microsoft Fabric framework for standardized ingestion, transformation, historization, logging, and publication of analytical data products.

The specification is intended as a foundation for implementation. It defines the target workspace structure, data architecture, metadata model, logging model, orchestration pipelines, notebook transformation patterns, and permission concept.

2. Scope

The framework covers the following capabilities:

- Central metadata driven ingestion configuration.
- Supported source systems: Azure SQL Database, SQL Server, and Mirrored Database for Azure SQL.
- Ingestion into a Bronze Lakehouse landing zone.
- Medallion architecture with Bronze, Silver, and Gold layers.
- Configurable full and incremental loads.
- Configurable source filtering.
- Configurable Bronze to Silver data cleansing.
- Configurable historization in the Silver layer.
- Gold layer modeled in a Fabric Warehouse by preference.
- Semantic models based on the Gold layer.
- Central SQL based logging and operational monitoring.
- Least privilege permission concept across platform, data, logic, model, and report workspaces.

3. Design Principles

- Metadata first: operational behavior must be controlled by metadata, not hardcoded per source table.
- Separation of concerns: data storage, orchestration logic, semantic models, and reports must be separated into dedicated workspaces.
- Reusability: ingestion and transformation notebooks must be generic and parameterized.
- Observability: every framework run must write start, end, status, row counts, durations, warnings, and errors to SQL logging tables.
- Least privilege: users, service principals, and managed identities receive only the minimum required access.
- Idempotency: re-running a failed object load must not create duplicate target records when the configured key and load pattern support deterministic processing.
- Recoverability: failed pipeline and object runs must be restartable at batch, source, object, or activity level.
- Auditability: metadata changes and run history must be traceable.

4. High Level Architecture

specification_fabric_framework_v1.md

#file:specification_fabric_framework_v1.md contains a specification for Microsoft Fabric framework.

Analyze and review this specification.
List all your questions and unclear points in the specification.
Which chapters need a refinement?

Claude Sonnet 4.6 · Hi

⚠️ New Chat-Session!

⚠️ Iterate

⚠️ Change models in between

Reviewed 5 sections of specification_fabric_framework_v1.md

Here is the full review of the specification.

Review: Questions, Unclear Points, and Chapters Needing Refinement

Chapter 3 — Design Principles

Questions / Unclear points:

- **Idempotency** is qualified with "when the configured key and load pattern support deterministic processing." Which combinations are *not* idempotent? What is the expected behavior when idempotency cannot be guaranteed (e.g., append-only Bronze with no key)?
- **Recoverability** promises "activity level" restart, but Fabric pipelines do not support native partial activity reruns. How is this achieved in practice?

Chapter 6 — Workspace Structure ⚠️ Needs refinement

Questions / Unclear points:

- `env_framework_default` is placed in the **Platform** workspace, but notebooks reside in the **Logic** workspace. Fabric does not natively support cross-workspace environment assignment for notebooks. How is this shared environment referenced from the Logic workspace?
- `lib_framework` is described as "shared Python package or notebook resources" — this is ambiguous. Is it a Fabric environment library (`.whl`), a mounted notebook, an `%run` include, or a standard Python package? This is a critical implementation choice that needs to be resolved here.
- The Logic workspace is potentially shared across domains, but Data workspaces are domain-specific. The specification does not explain how notebooks resolve the correct Bronze/Silver Lakehouse for a given domain at runtime.
- `nb_ingestion_control` is listed as optional ("if SQL stored procedures are insufficient") with no criteria for when it is invoked. This ambiguity will cause implementation inconsistency.

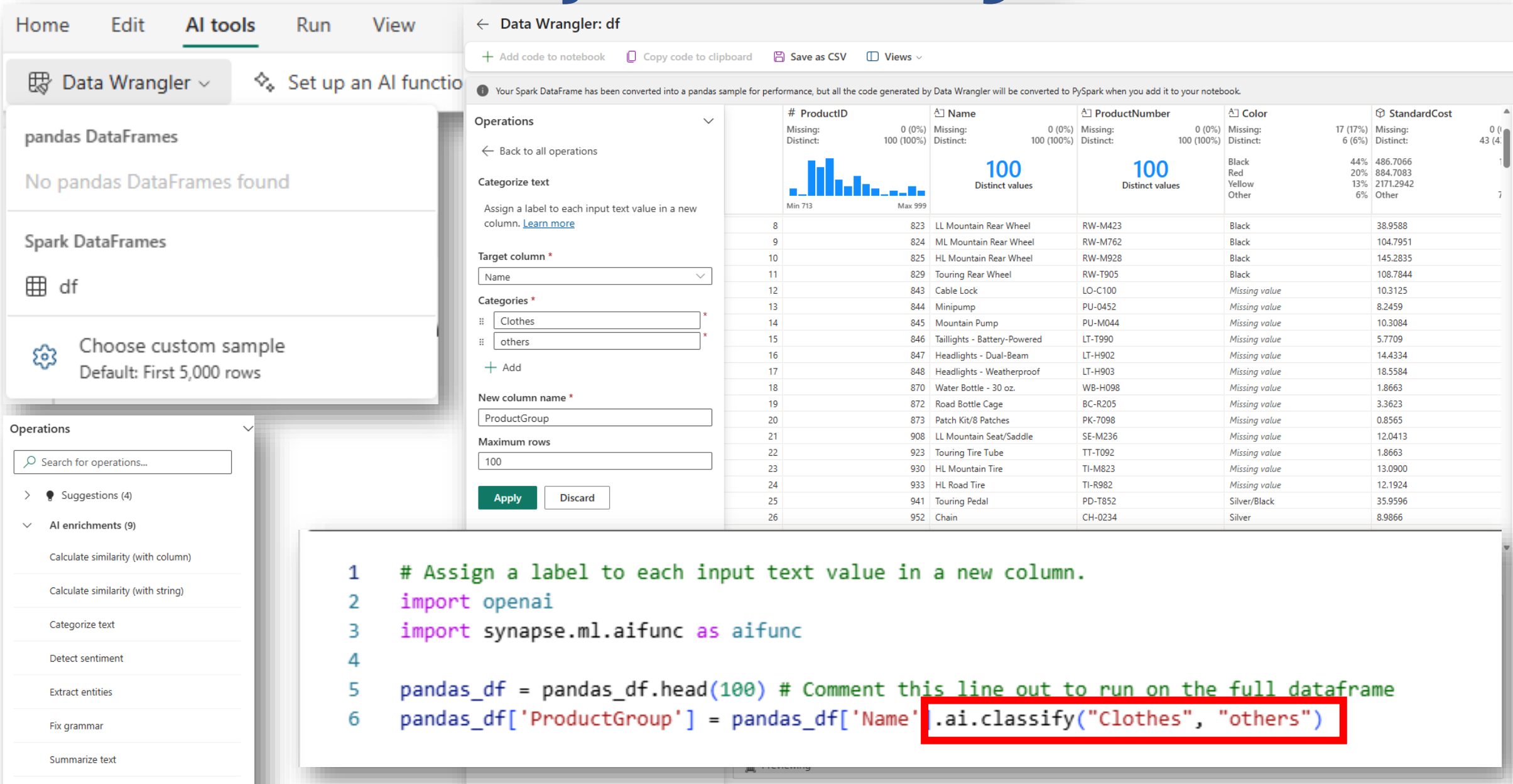
Chapter 7 — Permission Concept ⚠️ Needs refinement

Summary: Chapters Priority for Refinement

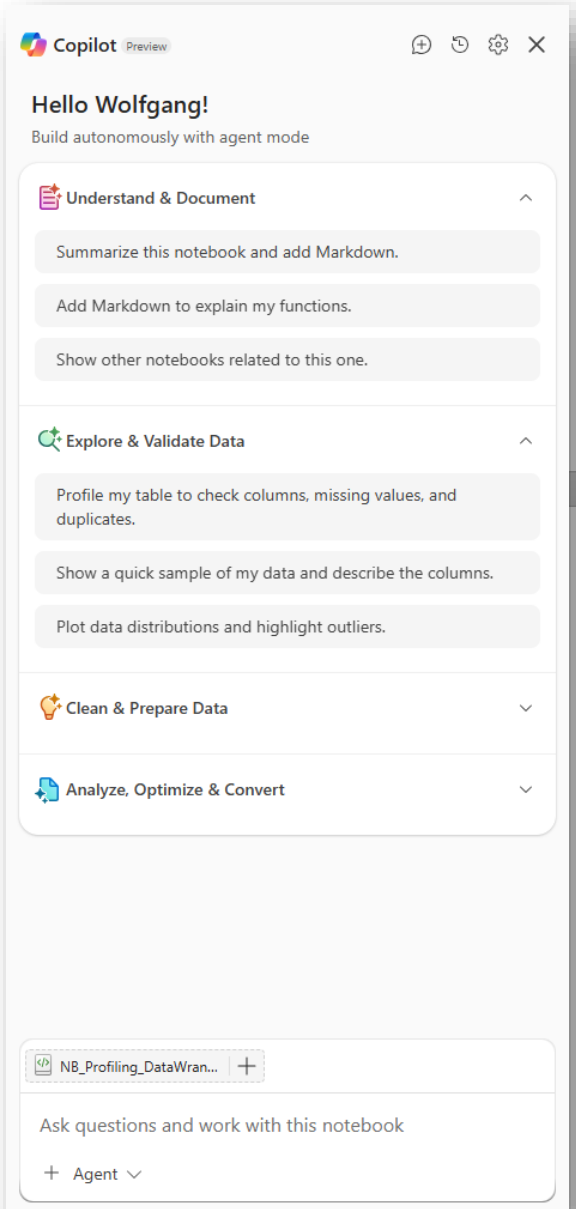
Priority	Chapter	Reason
High	10 — Metadata Database	Incomplete constraints, missing FK, concurrent watermark safety, SQL injection risk
High	12 — Pipeline Orchestration	Undefined rerun modes, missing dependency algorithm, stage execution logic gaps
High	13 — Notebook Transformation	SQL connectivity undefined, replay mode undefined, SCD2 atomicity unspecified
High	6 — Workspace Structure	`lib_framework` packaging undefined, cross-workspace environment unresolved
Medium	11 — Logging Database	Stage-per-object_run ambiguity, missing rerun linkage, no retention/index spec
Medium	7 — Permission Concept	Workspace identities missing, metadata access pattern undefined
Medium	8 — Data Architecture	Landing path edge cases, `_is_current` duplication, surrogate key generation
Medium	14 — Semantic Model	Too thin for production use, no RLS, no refresh strategy
Medium	20 — Open Decisions	No recommendations, several are implementation blockers
Lower	16 — Operational Monitoring	Missing SQL backing for report metrics
Lower	17 — Deployment	Missing CI/CD tooling and item ID update procedure

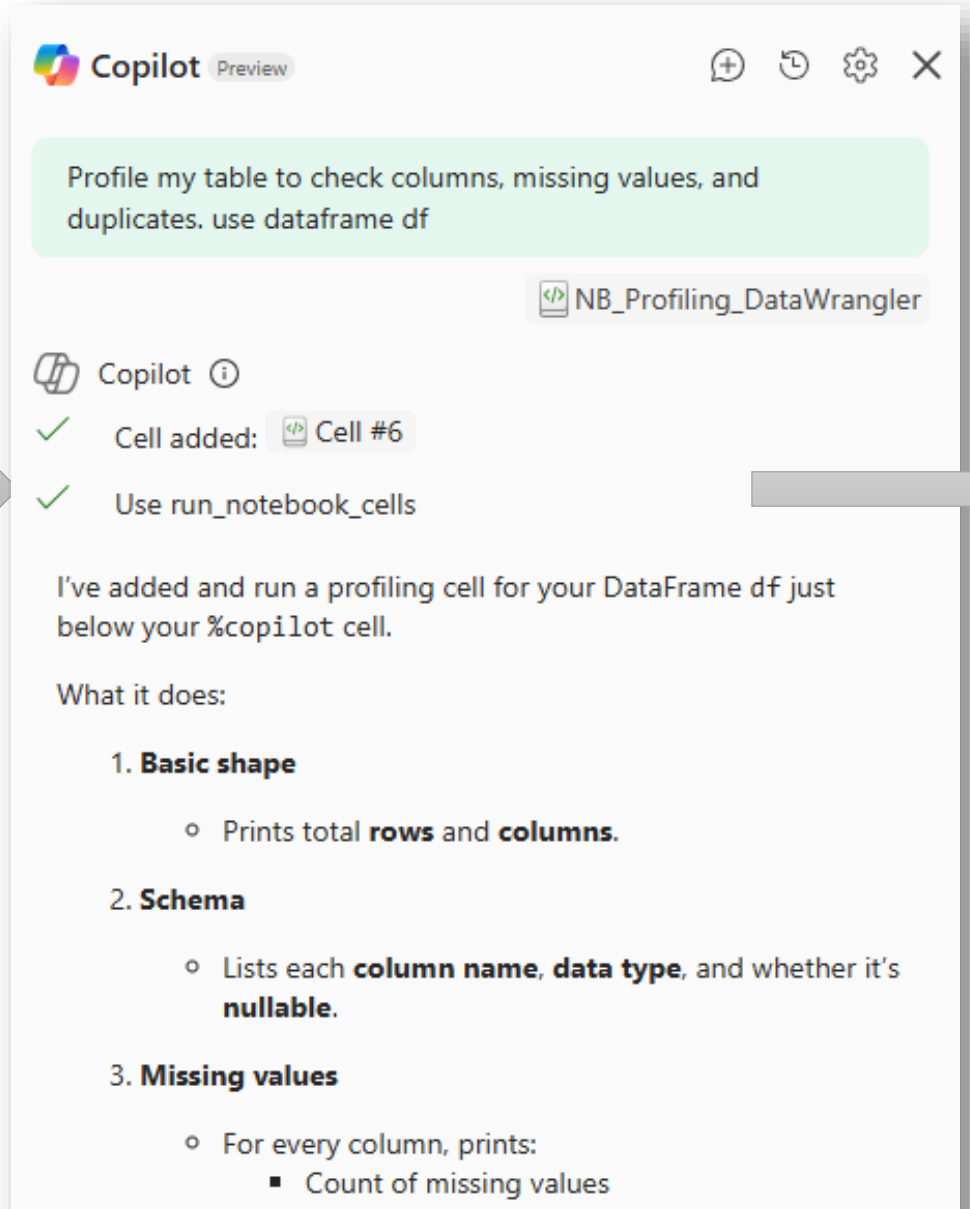
Source Discovery

Source Discovery (Data Wrangler)



Source Discovery (Copilot)






```
Rows: 295, Columns: 16

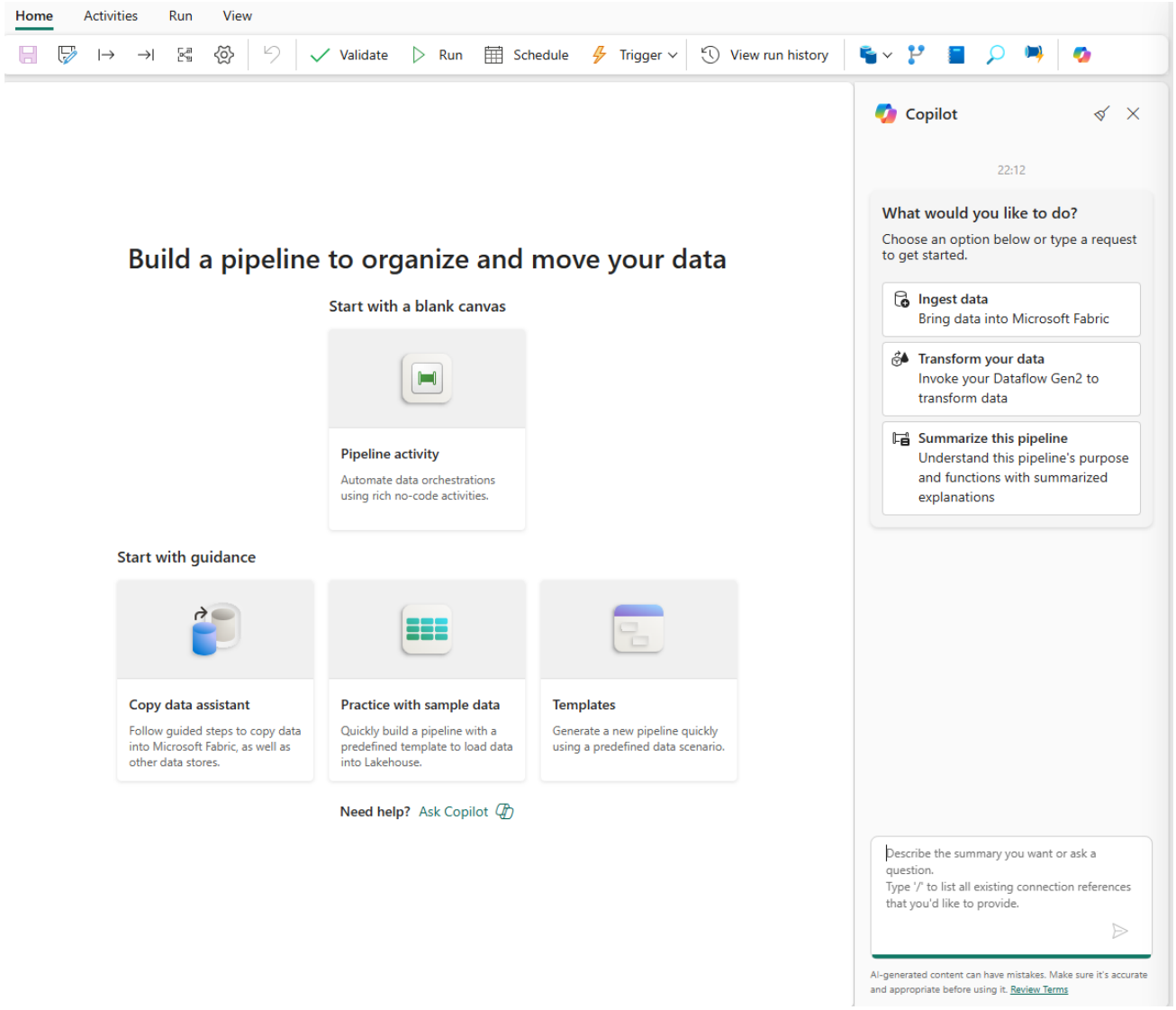
Schema (column name : type):
- ProductID: int (nullable=True)
- Name: string (nullable=True)
- ProductNumber: string (nullable=True)
- Color: string (nullable=True)
- StandardCost: decimal(19,4) (nullable=True)
- ListPrice: decimal(19,4) (nullable=True)
- Size: string (nullable=True)
- Weight: decimal(34,6) (nullable=True)
- ProductCategoryID: int (nullable=True)
- ProductModelID: int (nullable=True)
- SellStartDate: timestamp (nullable=True)
- SellEndDate: timestamp (nullable=True)
- DiscontinuedDate: timestamp (nullable=True)
- ThumbnailPhotoFileName: string (nullable=True)
- rowguid: string (nullable=True)
- ModifiedDate: timestamp (nullable=True)

Missing values per column:
- ProductID: 0 missing (0.00% )
- Name: 0 missing (0.00% )
- ProductNumber: 0 missing (0.00% )
- Color: 50 missing (16.95% )
- StandardCost: 0 missing (0.00% )
- ListPrice: 0 missing (0.00% )
- Size: 84 missing (28.47% )
- Weight: 97 missing (32.88% )
- ProductCategoryID: 0 missing (0.00% )
- ProductModelID: 0 missing (0.00% )
- SellStartDate: 0 missing (0.00% )
- SellEndDate: 197 missing (66.78% )
- DiscontinuedDate: 295 missing (100.00% )
- ThumbnailPhotoFileName: 0 missing (0.00% )
- rowguid: 0 missing (0.00% )
- ModifiedDate: 0 missing (0.00% )

Duplicate rows (all columns):
```

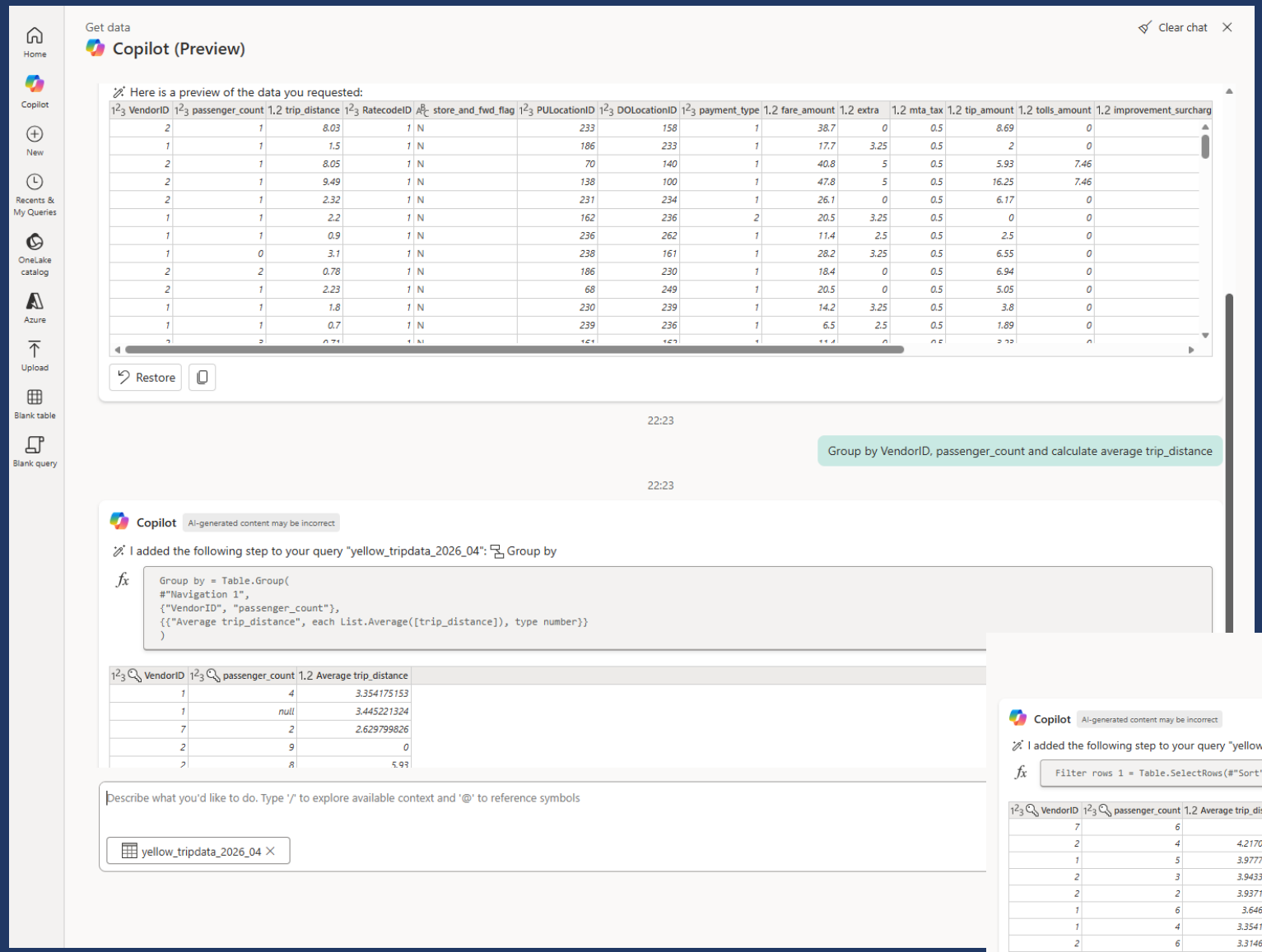
Ingestion

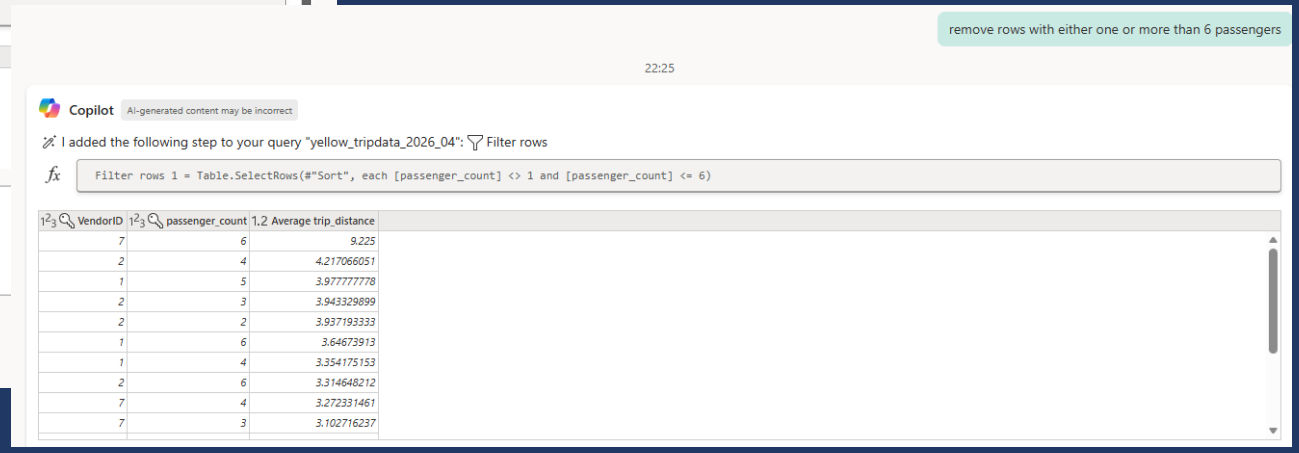
Ingestion (help with Copilot)



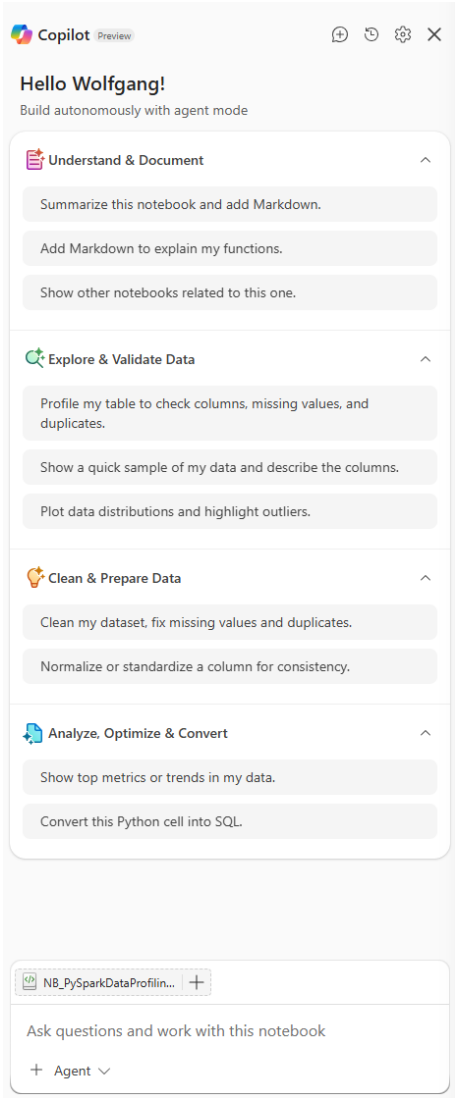
Transformation

Transformation – DataFlow Copilot

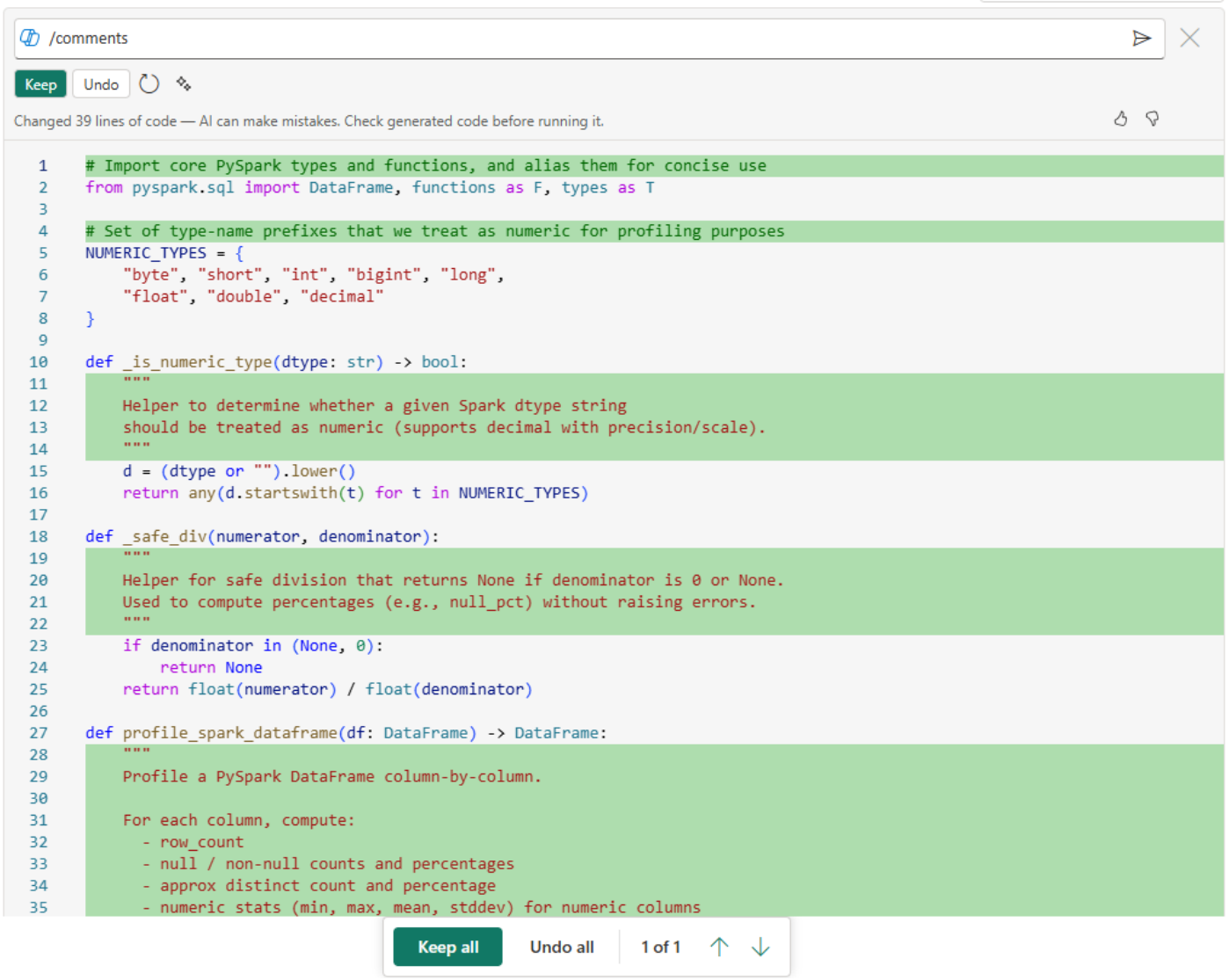




Transformation (Notebooks)



Copilot Chat



In Cell Mode

Implementation

VIBE CODING VS. AGENTING ENGINEERING

VIBE CODING

You prompt. AI generates.
You iterate.

Prompt-driven
You tell AI what you want.

Focus on speed
Quick results, MVP mindset.

Hands-on execution
You build with AI step by step.

You drive the work
Direct control, one task at a time.

More rework risk
Context is manual, easy to lose.

VS.

AGENTING ENGINEERING

You define goals. Agents deliver.
You orchestrate.

Goal-driven
You define the outcome.

Focus on outcomes
Quality, reliability, and impact.

Orchestrate agents
Agents plan, code, test, and iterate.

You architect & guide
Set direction, constraints, and guardrails.

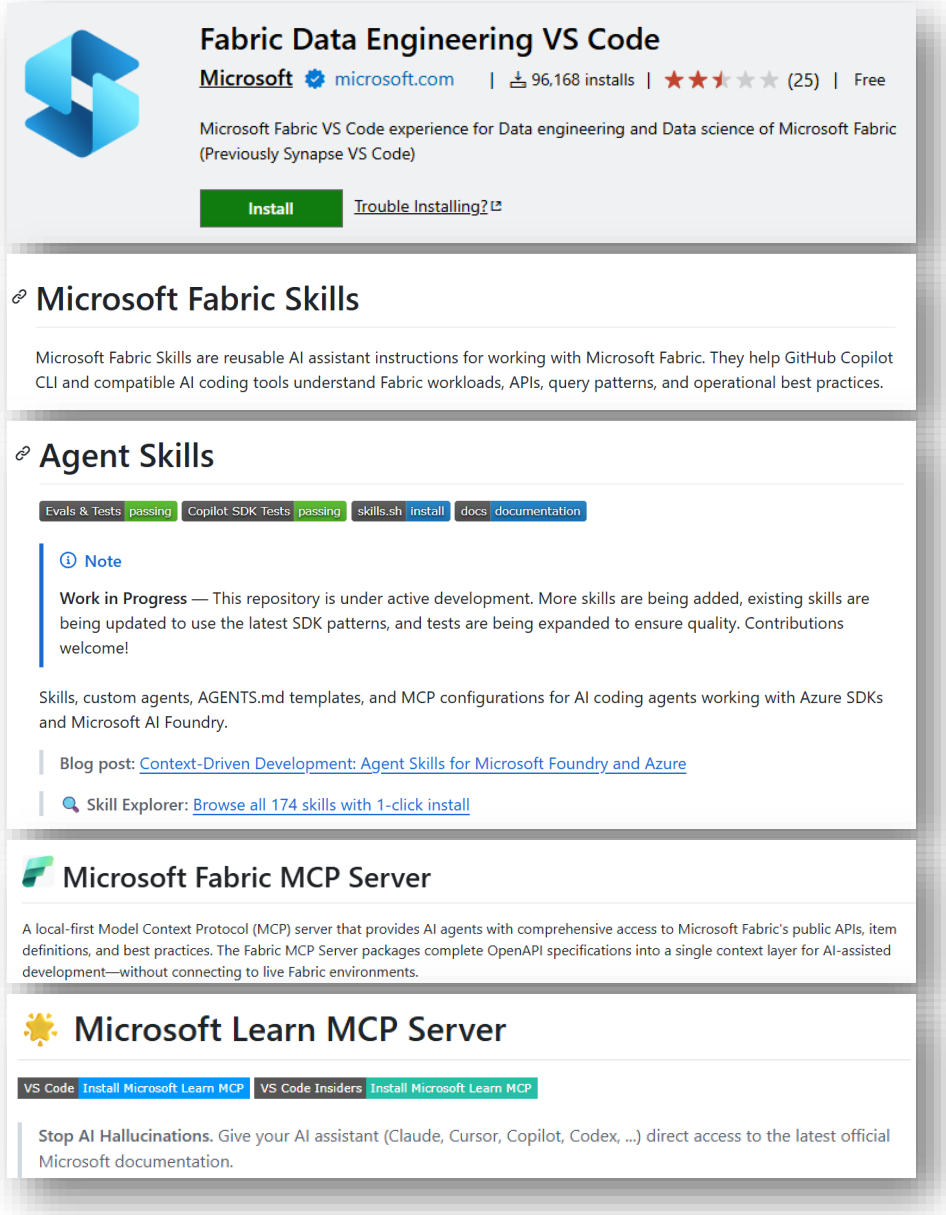
More reliable delivery
Persistent context, testing, and feedback loops.

VIBE CODING WRITES CODE.
AGENTING ENGINEERING DELIVERS SYSTEMS.

A Toolbox for Fabric Agentic Engineering

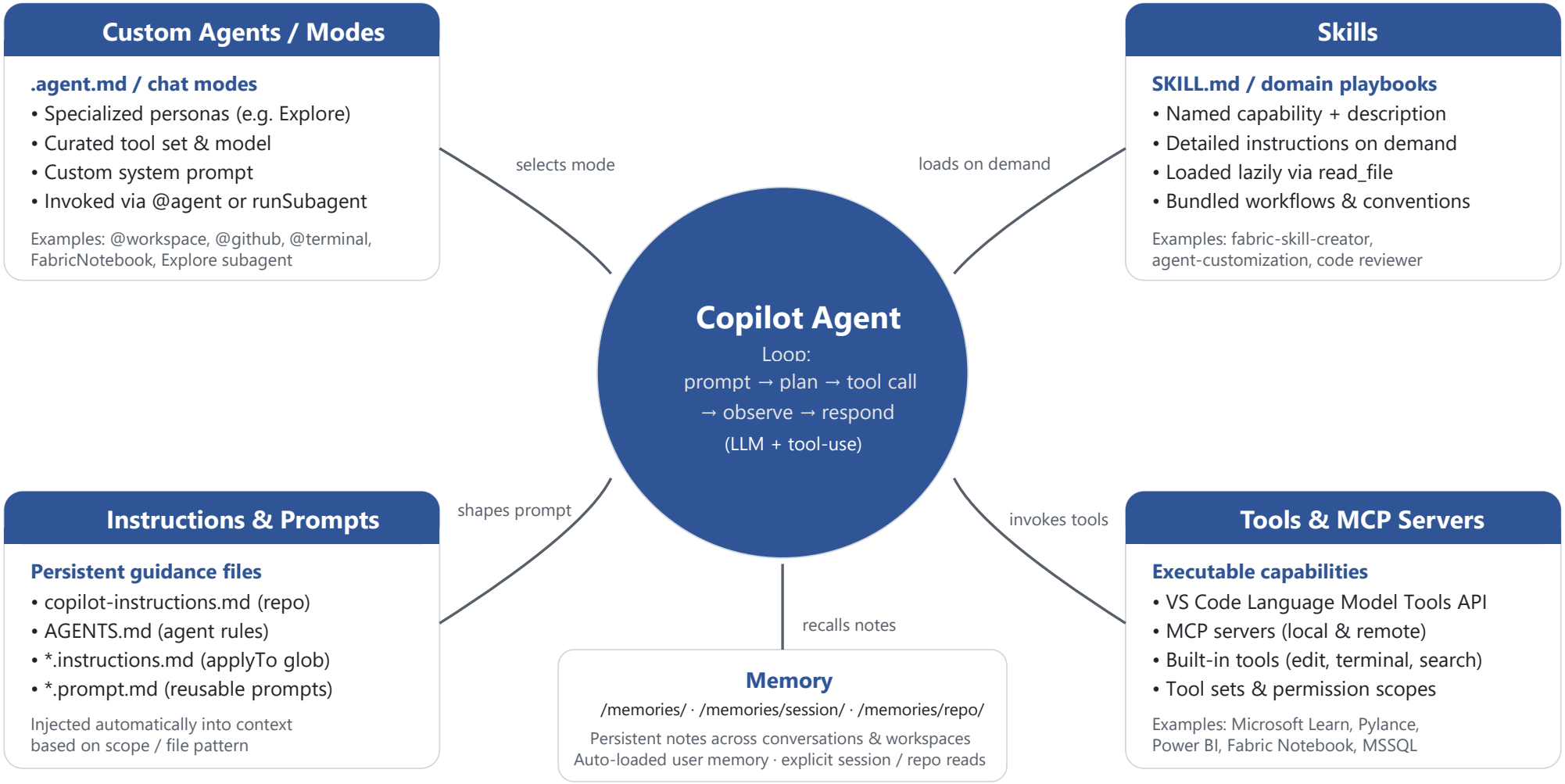
GitHub Copilot

Claude



GitHub Copilot – Extensibility

How custom agents, skills, instructions, and tools plug into the Copilot agent loop



But there is more... SQL for example!?

GitHub Copilot for MSSQL extension for Visual Studio Code

Summarize this article for me

The GitHub Copilot integration with the MSSQL extension for Visual Studio Code provides AI-assisted SQL development. You can use it to write and optimize database code, generate and modify schemas, understand existing logic, and work with code-first and data-first database development patterns.

This integration is designed for developers, with a focus on the following personas:

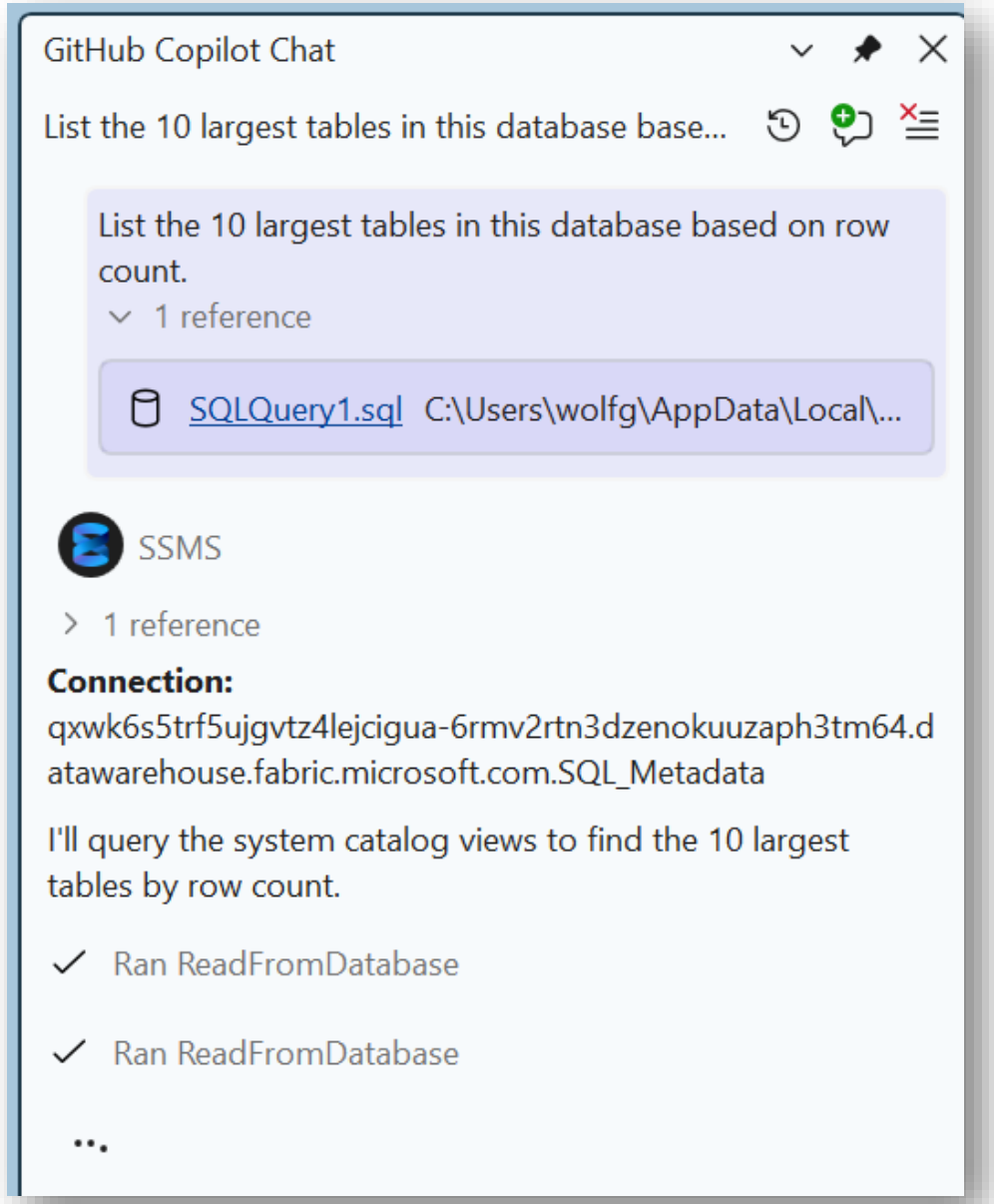
- **Modern application developer**: Builds feature-rich, scalable applications with frameworks like Node.js, Python, .NET, and Go.
- **AI / cloud-native developer**: Specializes in containerized, serverless, and microservices-based applications, often integrating AI-powered capabilities.
- **Solutions architect**: Designs end-to-end systems that align database solutions with broader architectural goals.
- **Database developer**: Focuses on Transact-SQL (T-SQL), database-specific concepts, and optimizing database workflows.

Get started with GitHub Copilot in SQL Server Management Studio

Summarize this article for me

GitHub Copilot is an AI assistant in SQL Server Management Studio (SSMS).

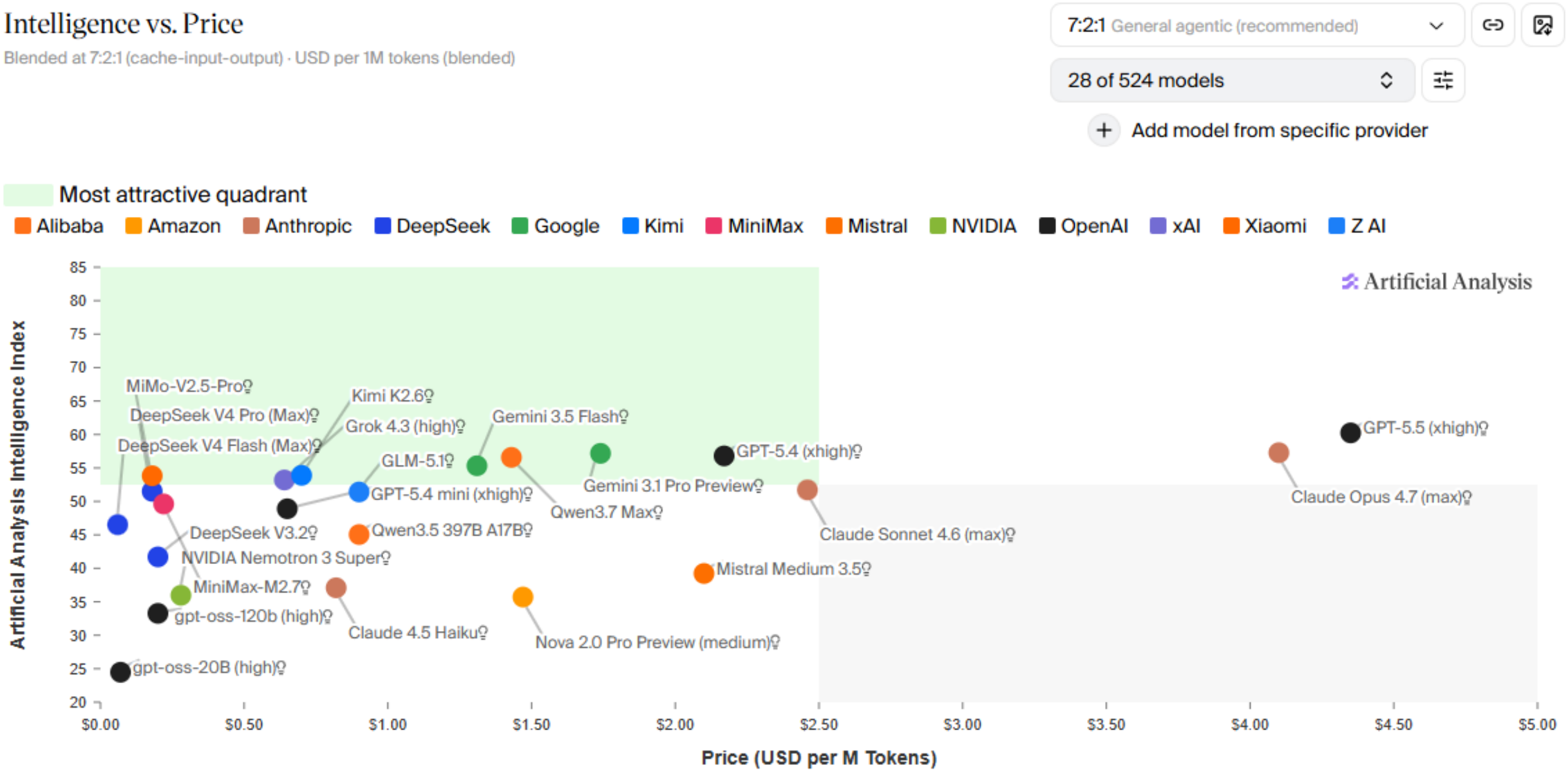
GitHub Copilot helps you write Transact-SQL (T-SQL) faster and more accurately by suggesting queries, edits, answers, and administrator tasks directly in your SSMS environment.



Some Findings

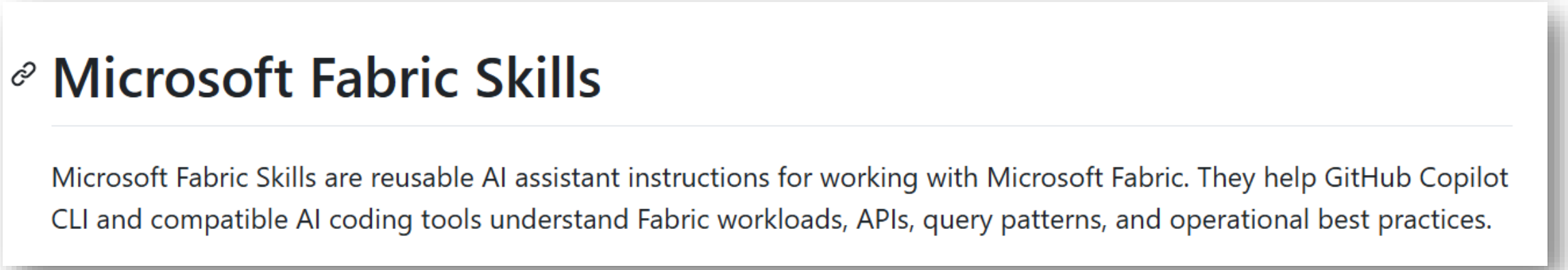
- Plan before implementation (Plan mode)
  - Spend more time in planning and spec -> faster and less errors
- Think about the model
  - Spec & plan = better model
  - Implementation = less capable model
- Think about testing

There is no free lunch



Skills + Agents for Microsoft Fabric Development

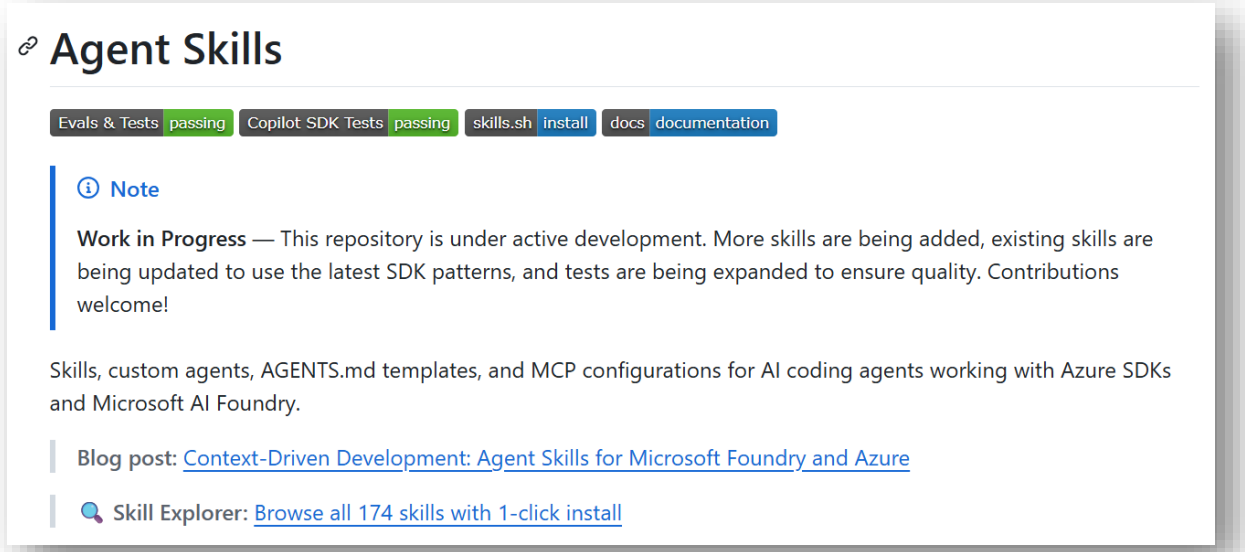
https://github.com/microsoft/skills-for-fabric



Microsoft Fabric Skills

Microsoft Fabric Skills are reusable AI assistant instructions for working with Microsoft Fabric. They help GitHub Copilot CLI and compatible AI coding tools understand Fabric workloads, APIs, query patterns, and operational best practices.

https://github.com/microsoft/skills



Agent Skills

Evals & Tests passing | Copilot SDK Tests passing | skills.sh install | docs documentation

> **Note**
>
> **Work in Progress** — This repository is under active development. More skills are being added, existing skills are being updated to use the latest SDK patterns, and tests are being expanded to ensure quality. Contributions welcome!

Skills, custom agents, AGENTS.md templates, and MCP configurations for AI coding agents working with Azure SDKs and Microsoft AI Foundry.

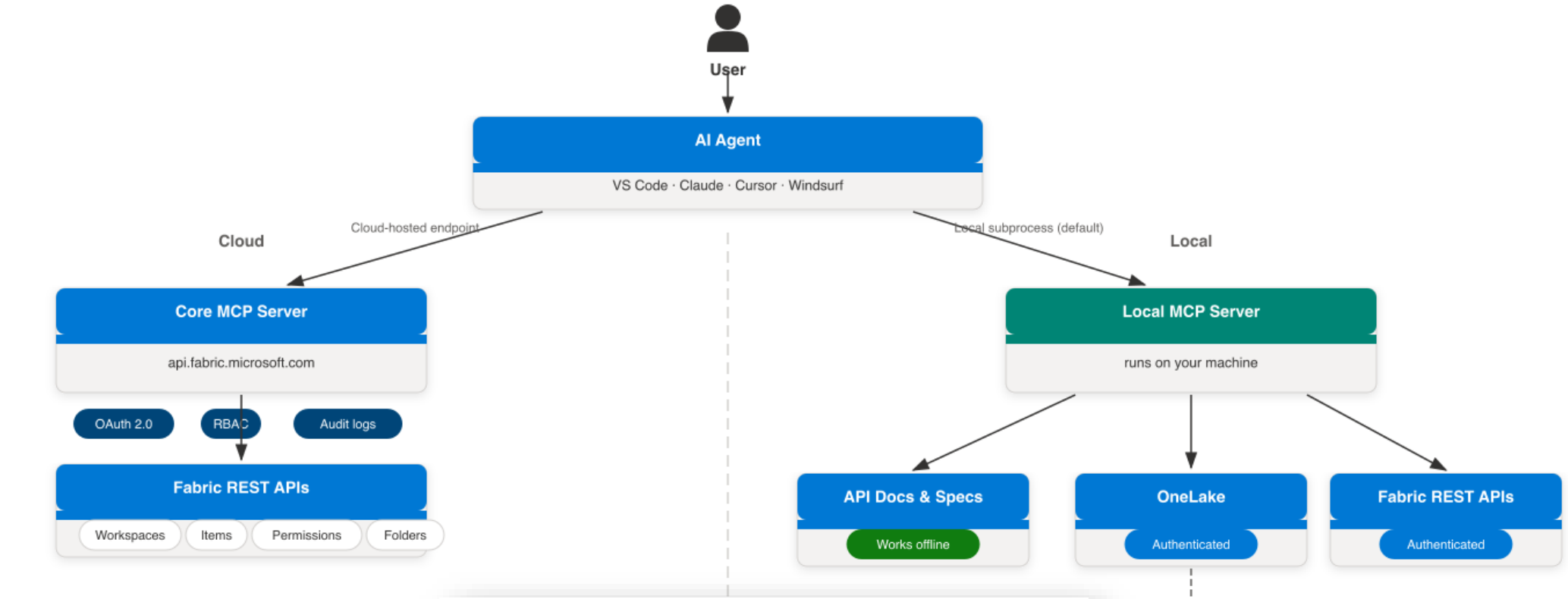
> Blog post: Context-Driven Development: Agent Skills for Microsoft Foundry and Azure

> 🔍 Skill Explorer: Browse all 174 skills with 1-click install

Governance

https://learn.microsoft.com/en-us/rest/api/fabric/articles/mcp-servers/what-is-fabric-mcp-server

Fabric MCP server(s)



Feature	Core (remote)	Local
Workspace management	Yes	—
Item CRUD	Yes	Create only
Permission management	Yes	—
Folder management	Yes	—
OneLake file operations	—	Yes
API docs and specs (offline)	—	Yes
Installation required	No	Yes
Primary install method	N/A (remote endpoint)	VS Code extension
Authentication	OAuth 2.0 (Entra ID)	Local credentials
Audit logging	Yes	—
Open source	No	Yes

Use both servers together — Core for cloud management, Local for development workflows.

Governance (with Fabric MCP Server)

Analyze the content of the workspace "Fabric Data Quality"

Analyzed content of workspace "Fabric Data Quality"

Ran List Items – fabric (MCP Server)

Reviewed 2 files

Now let me also get the folder structure:

Ran List Folders – fabric (MCP Server)

Here's a full analysis of the **Fabric Data Quality** workspace:

Workspace Info

- **Capacity:** North Europe (`ff8ba772...`)
- **Domain:** Fabric Data Quality (`1eba668d...`)
- **Total items:** 35

Items by Type

Type	Count
Notebook	19
SemanticModel	4
SQLEndpoint	3
Lakehouse	3
Report	2
Dashboard	1
Dataflow	1
Warehouse	1
Environment	1
CopyJob	1

List me all the notebooks that do not follow the naming convention to start with the prefix NB_

Identified notebooks lacking "NB_" prefix in names

Based on the workspace items already retrieved, here are the **7 notebooks** that don't follow the `NB_` naming convention:

#	Name	Folder	Issue
1	`Notebook 1`	root	Generic default name
2	`Notebook 2`	root	Generic default name
3	`Notebook 3`	900 System Overview	Generic default name
4	`data_cleaning_functional_dependencies_tutorial`	root	No prefix at all
5	`ZZ_NB_DQChecks_GreatExpectation_sempy`	400 Data Quality Checks	Prefixed with `ZZ_` instead of `NB_`
6	`ZZ_Data Quality Framework`	400 Data Quality Checks	Prefixed with `ZZ_`, spaces in name
7	`NB_Profiling_ydata Copy`	root	Has `Copy` suffix — likely a forgotten draft

Items 1–4 have no prefix at all. Items 5–6 use a `ZZ_` prefix (possibly a deliberate "archive/deprecated" convention, but inconsistent with `NB_`). Item 7 technically starts with `NB_` but the `Copy` suffix suggests it's a stray duplicate that should be cleaned up or renamed.

Governance pt 2

explain me the content / logic in the notebook - data_cleaning_functional_dependencies_tutorial

Searched for content in data_cleaning_functional_dependencies_tutorial

Ran Get Item Definition – fabric (MCP Server)

Searched for tool usage in the project

Ran Get Operation State – fabric (MCP Server)

Ran Get Operation Result – fabric (MCP Server)

Reviewed content.json and ran command in terminal

Ran `$json = Get-Content "c:\Users\wolfg\AppData\Roamin...`

Reviewed 2 files

Here's a full explanation of the notebook's content and logic:

`data_cleaning_functional_dependencies_tutorial`

This is a **Microsoft official tutorial notebook** (from the Fabric samples) demonstrating how to use **functional dependencies** for data quality analysis and cleaning. It uses the **SemPy** (Semantic Link Python) library.

Concept: Functional Dependencies

A functional dependency means one column *determines* another. Example: a `ZIP` code should always map to the same `CITY`. If it doesn't, that's a data quality problem.

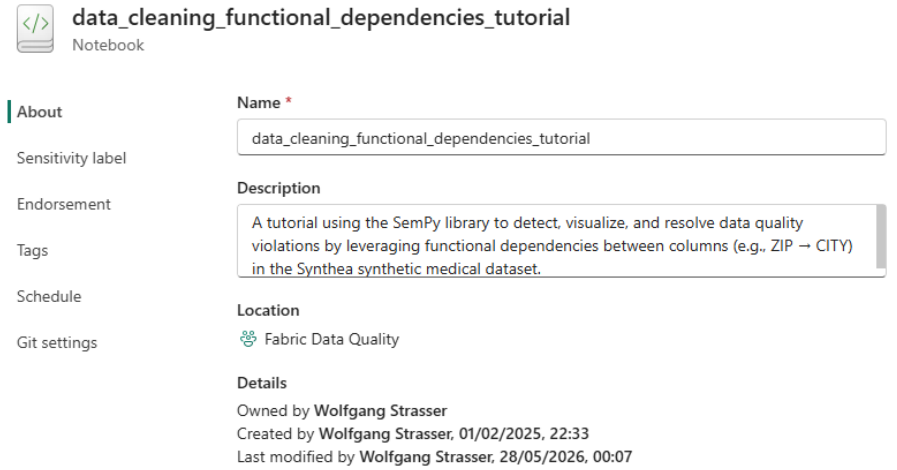
summarize this logic in one sentence and update the description of the notebook with this summary

Updated description of data_cleaning_functional_dependencies_tutorial

Summary: *A tutorial using the SemPy library to detect, visualize, and resolve data quality violations by leveraging functional dependencies between columns (e.g., ZIP → CITY) in the Synthea synthetic medical dataset.*

Ran Update Item – fabric (MCP Server)

Done. The notebook description has been updated successfully.



Your code = Knowledge base

What happens if I change <ObjectX>?

If the structure of the lakehousetable sales_transactions_import changes, which items will break?

Searched for sales_transactions_import

I found more items than expected — there's a `Reports/` folder too. Let me get the full picture:

Reviewed 4 files and searched for multiple terms

Now let me check what specific columns from the table are referenced in the reports to understand exactly what structural changes would break:

Reviewed sales_transactions_import.tmdl

Here is the complete impact analysis:

Items that break if `sales_transactions_import` structure changes

```
LH_LandingZone.sales_transactions_import
        ↓
DF_ImportTransactionData  (writes here)
        ↓ (also reads)
Semantic Model Sales      (DirectLake)
        ↓
Sales Report              (Power BI Report)
Report for my Sales People (Power BI Report)
```

---

1. `DF_ImportTransactionData` — Dataflow Gen2 *(breaks on: column removed/renamed)*

The mashup.pq navigates directly to `sales_transactions_import` as a data destination and maps 27 named columns from the CSV. Any column rename/removal in the lakehouse table will cause the destination query to fail.

2. `Semantic Model Sales` — SemanticModel *(breaks on: column removed/renamed/retyped)*

sales_transactions_import.tmdl maps every column 1:1 by name via `sourceColumn:` and specifies data types. Changes to:

- **Column name** → `sourceColumn` reference breaks
- **Column removed** → referenced field missing → semantic model fails to process
- **Data type change** → type mismatch (e.g., `billing_postal_code` expects `int64`)
- **Table renamed** → `entityName: sales_transactions_import` in the partition definition breaks

3. `Sales Report` — Power BI Report *(breaks on: `currency` or `total` renamed/removed)*

If I rename the currency column in sales_transactions_import. Which objects break? Which reports actually use this column?

Searched for currency in workspaces

Based on the search results, here's the precise impact:

Objects that break if `currency` is renamed

1. `Semantic Model Sales` — SemanticModel *(breaks)*

sales_transactions_import.tmdl:178 maps the column directly:

```
column currency
    sourceColumn: currency
```

The `sourceColumn` reference to the lakehouse column would no longer resolve.

2. `DF_ImportTransactionData` — Dataflow Gen2 *(breaks)*

mashup.pq:7 explicitly types the column by name `{"currency", type text}` in `Table.TransformColumnTypes`, and writes it to the destination table. If the destination column name changes, the load will fail or create a mismatched column.

3. `Sales Report` — Power BI Report *(breaks)*

Sales Report/report.json has a table visual with `sales_transactions_import.currency` as a field reference — it would show a broken/missing field error.

Report that does NOT use `currency`

`Report for my Sales People` — no reference to `currency` anywhere in its report.json. It would be unaffected.

Summary:

Item	Breaks?	Reason
`Semantic Model Sales`	✅ Yes	`sourceColumn: currency` won't resolve
`DF_ImportTransactionData`	✅ Yes	Column name in type mapping and destination won't match
`Sales Report`	✅ Yes	Visual directly references `currency` column
`Report for my Sales People`	❌ No	Does not use `currency`

Documentation

Vibe Documentation

Generate me a documentation of the data flow in this solution (/workspaces subfolders).

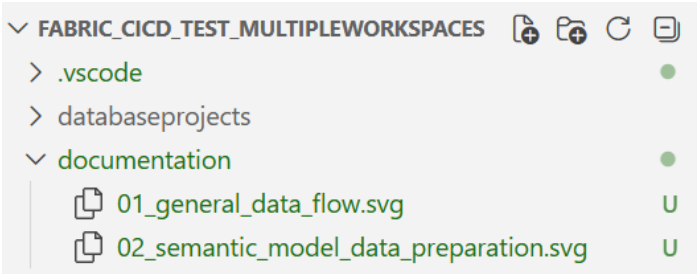
I want you to generate graphical documentation (svg - check syntax - especially escaping).

- General data flow
- detailed data preparation logic (per semantic model)

store the output files in the /documentation folder.

Explored workspaces directory for graphical documentation generation

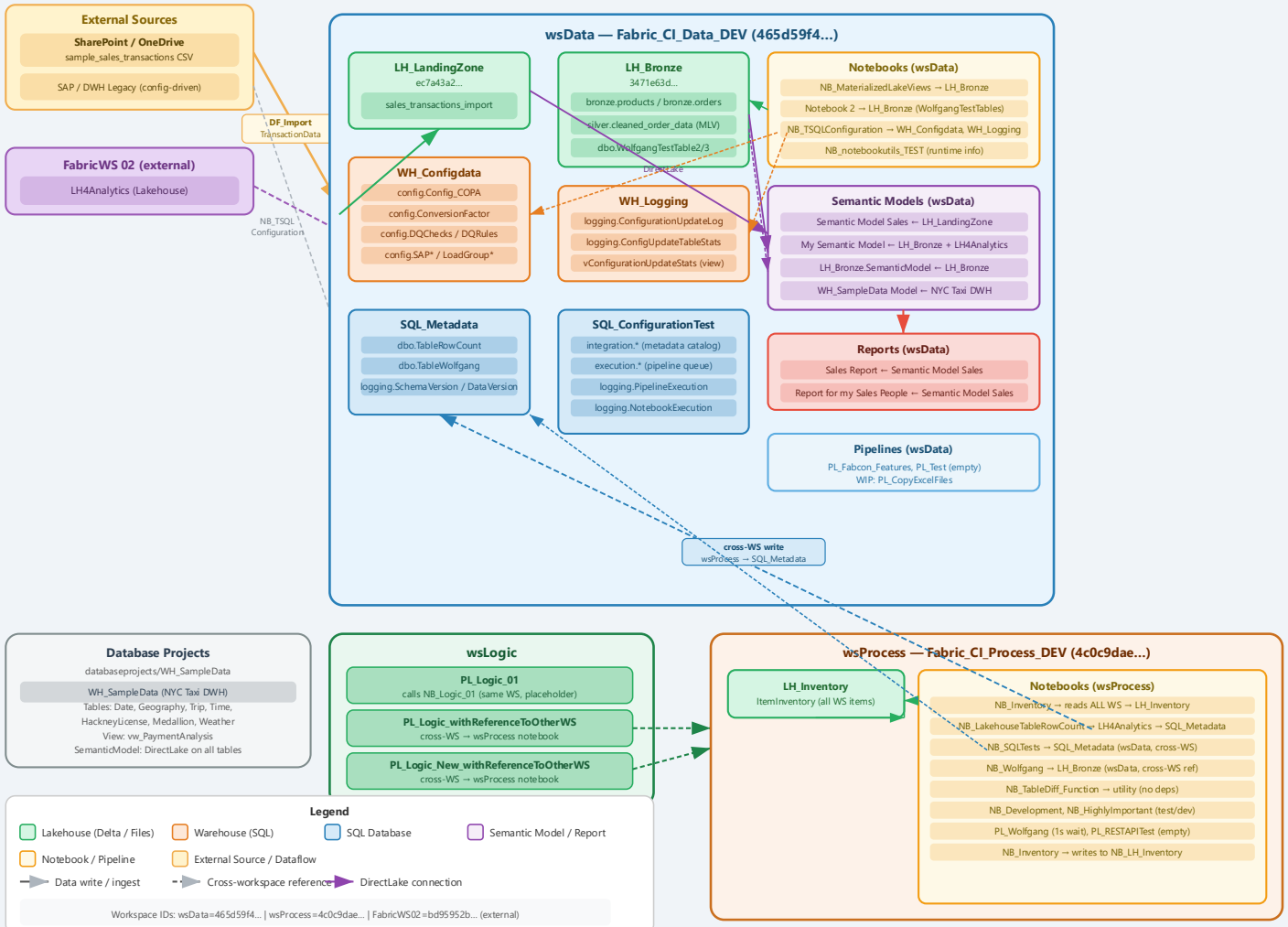
Subagent: Explore workspace structure for data flow documentation — Reading Sales Report.Report



Generated Docu

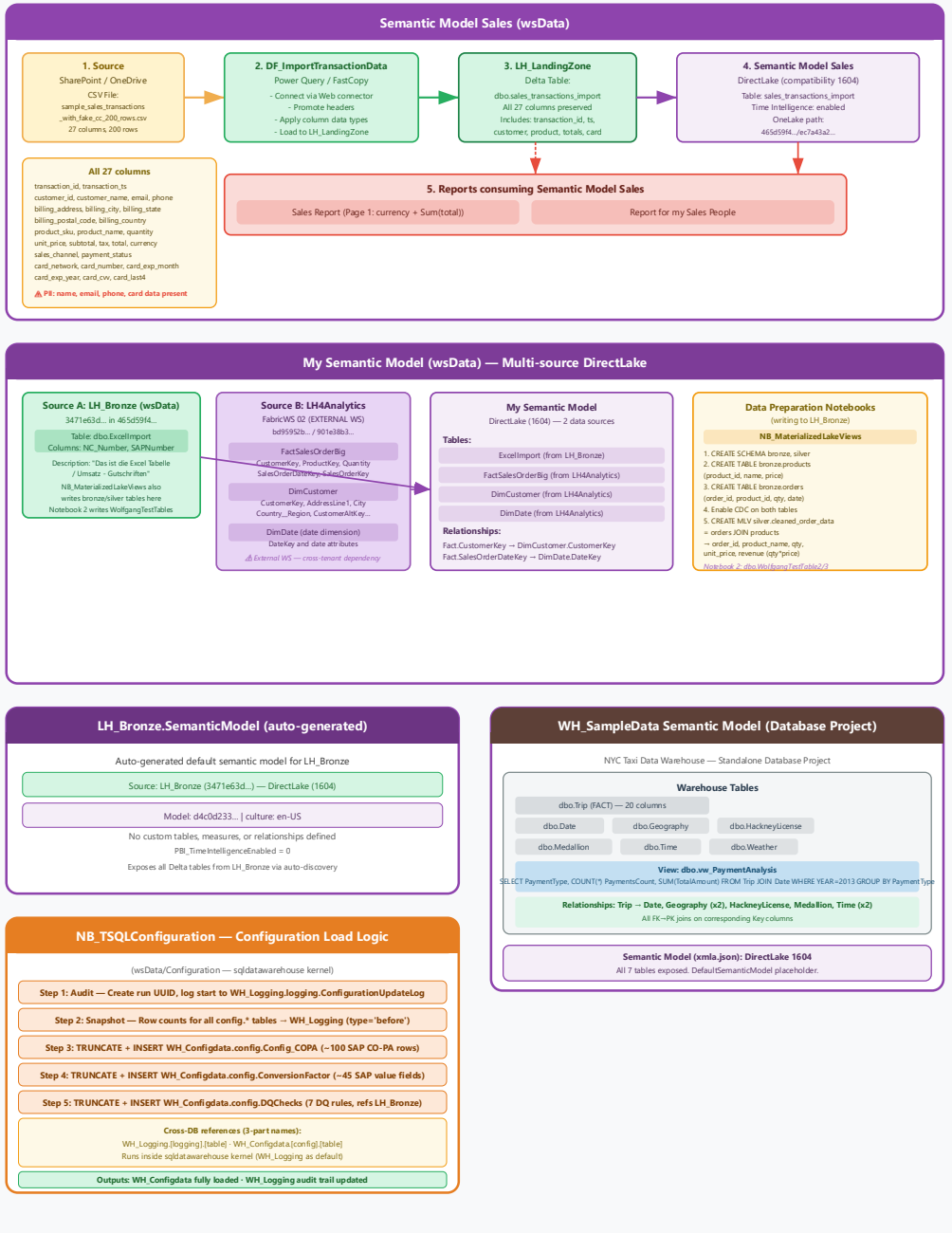
Fabric CICD Multi-Workspace — General Data Flow

Generated from workspace analysis · 2026-05-28



Semantic Model Data Preparation — Detailed Logic

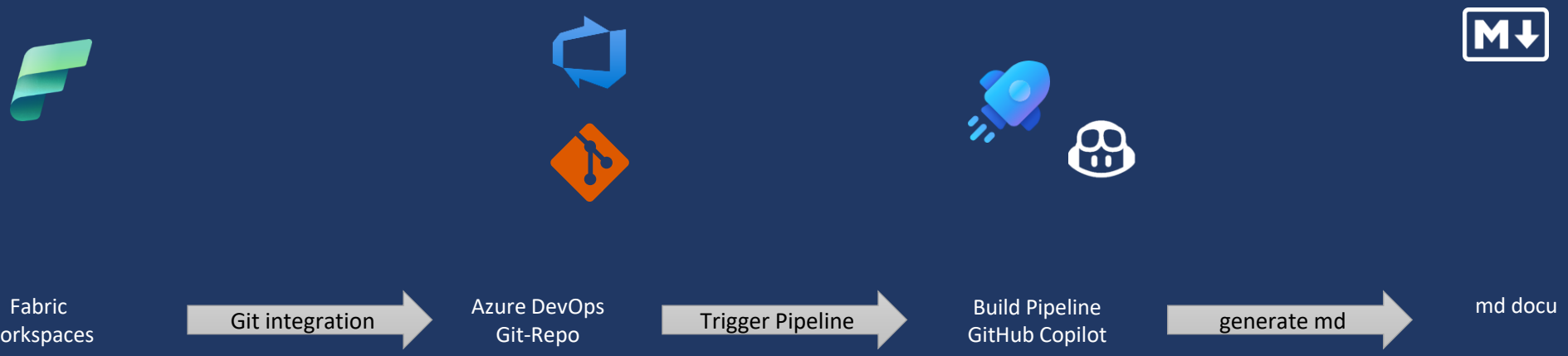
Per-model ingestion and transformation chains · 2026-05-28



Documentation (in sync & automated) ⚙️

What if....

- I do not have to trigger documentation generation manually?
- I have a documentation that _really_ represents my up-to-date source?
- I just automate the documentation process?



Jobs in run #20260528.6

Fabric_Documentation_with_Github_Co

Jobs

Job	12m 17s
Initialize job	<1s
Checkout Fabric_CICD_T...	4s
Generate docume...	9m 47s
validate architecture	2m 21s
Publish artifacts	3s
Post-job: Checkout Fa...	<1s
Finalize Job	<1s
Report build status	<1s

Job

```
1   Pool: Fabric
2   Agent: vm-devopsagent-
3   Started: Today at 15:29
4   Duration: 12m 17s
5
6 ▶ Job preparation parameters
31  1 artifact produced
```

Artifacts published

Name

- Documentation
  - fabric_docu_business_20260528_1329.md
  - fabric_docu_technical_20260528_1329.md
  - naming_convention_checker_20260528_1339.md

fabric_docu_business_20260528_1329.md

C: > Users > wolfg > Downloads > fabric_docu_business_20260528_1329.md

Microsoft Fabric Solution — Business Documentation

Document Header

Field	Value
Generated	2026-05-28 13:29 UTC
Generated by	GitHub Copilot CLI — automated documentation
Source path	`_work\1\s\workspaces`
Git last commit	`60ff5abeddf9573bc3f56962db50b54b4e84363e`
Commit author	Wolfgang Strasser <wolfgang@powerofbi.at>
Commit date	Thu May 28 13:29:03 2026 +0000
Commit message	Update FabricPipeline.yml for Azure Pipelines

Table of Contents

1. What This Solution Does
2. High-Level Architecture
3. Data Sources
4. How Data Flows Through the Platform
5. What Business Users Can Access
6. Workspace Organisation
7. Data Quality & Governance
8. Operational Monitoring

1. What This Solution Does

This Microsoft Fabric platform **collects data from multiple sources**, transforms
structured, layered data architecture that separates raw ingested data from clea

Key business capabilities delivered:

- **Sales transaction reporting** — Sales data loaded from SharePoint/OneD
- **Customer and sales order analysis** — A sales data model combines cus
- **Configuration-driven data integration** — The platform can onboard ne
- **Data quality management** — Rules are defined centrally and applied to
- **Operational monitoring** — Automated tracking of pipeline runs, notebo
- **Workspace inventory** — Automatic cataloguing of all Fabric items acros

Fabric Naming Convention Check Report

Generated: 2026-05-28 13:39
Repository path: `_work\1\s\workspaces`
Rules checked:

- Notebooks (`.Notebook`) → must start with `NB_`
- Pipelines (`.DataPipeline`) → must start with `PL_`

Summary Table

Workspace	Total Notebooks	Invalid Notebooks	Total Pipelines	Invalid Pipelines	Total Items	Total Invalid
wsData	7	2	3	0	10	2
wsLogic	1	0	3	0	4	0
wsProcess	13	0	2	0	15	0
TOTAL	**21**	**2**	**8**	**0**	**29**	**2**

> Items in workspace subfolders (e.g. `Configuration`, `WorkInProgress`, `ZZ_InDevelopment`) are attributed to their parent workspace.

Workspace Details

wsData

Notebooks (7 total | 2 invalid)

Item Name	Subfolder	Status
NB_MaterializedLakeViews_Test_Wolfgang	(root)	✅ Valid
NB_notebookutils_TEST	(root)	✅ Valid
Notebook 1	(root)	❌ Invalid — missing `NB_` prefix
Notebook 2	(root)	❌ Invalid — missing `NB_` prefix
NB_TSQLConfiguration	Configuration	✅ Valid
NB_AnalyzeSemanticModel	WorkInProgress	✅ Valid
NB_AuditLog	WorkInProgress	✅ Valid

Microsoft Fabric Solution — Technical Documentation

Document Header

Field	Value
	29 UTC
	CLI — automated documentation
	rkspaces
	573bc3f56962db50b54b4e84363e
	ser <wolfgang@powerofbi.at>
	29:03 2026 +0000
	ipeline.yml for Azure Pipelines

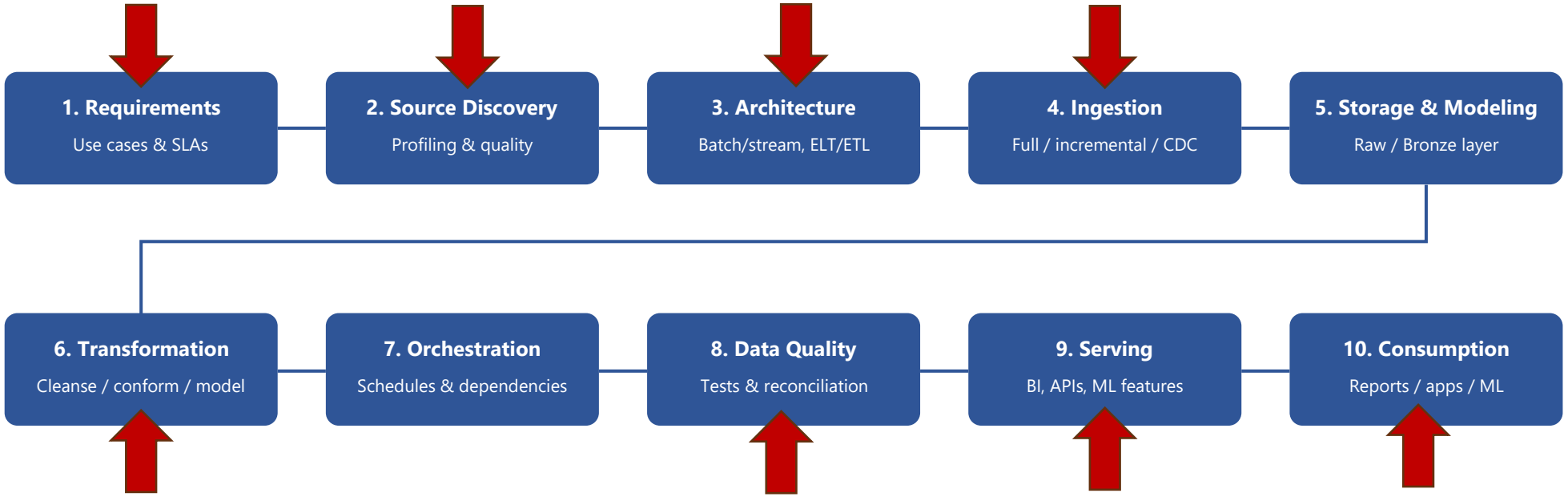
(WH_Configdata)
_Logging)

pace
orkspace

itecture

Typical Data Engineering Process

Linear core flow with continuous cross-cutting concerns



Cross-cutting concerns (apply throughout the lifecycle)



AI changes the workflow, But not the Responsibility

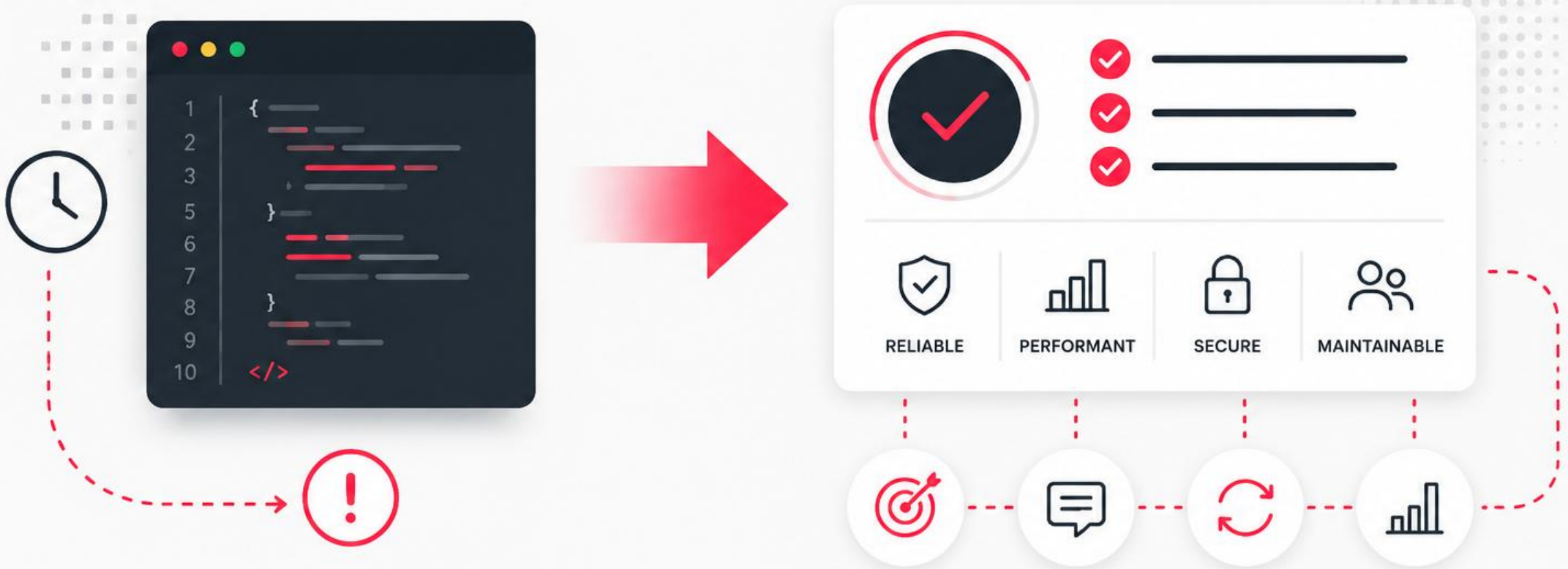
Traditional Workflow

1. **Request** — Understand the need
2. **Prepare** — Collect & process data
3. **Build** — Write code & create logic
4. **Validate** — Test & verify results
5. **Deliver** — Share insights & reports

AI-Augmented Workflow



From Code-First to Quality-First



Future Data Engineers don’t just create code.

They create trust.

No trust – no adoption.



AI-Powered Data Engineering with Microsoft Fabric – From Code Gen to Column-Level Impact Analysis

AI-powered Data Engineering with Microsoft Fabric

Wolfgang Strasser, June 2026

Microsoft® Most Valuable Professional

ACP CUBIDO Digital Solutions



Data Platform
Next Step