Microsoft Fabric

#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026

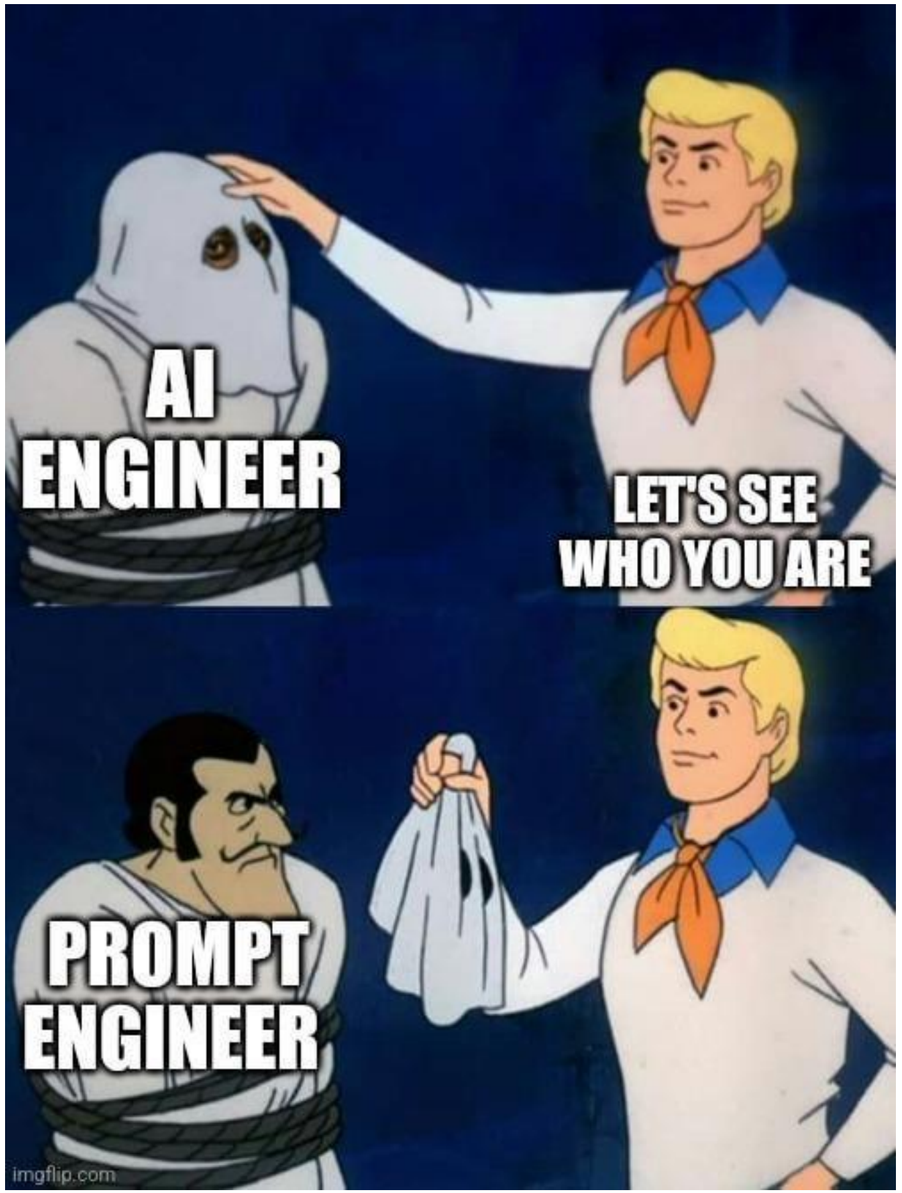
Forget Prompts: Building Declarative AI Pipelines in Fabric

Sandeep Pawar

Principal Program Manager, Fabric CAT: Data Science and AI

Agenda

- The Problem with Prompt Engineering at Scale
- What Are AI Pipelines? (vs. Data Pipelines)
- Fabric AI Functions: Built-in AI at Scale
- DSPy: Programming Over Prompting
- Use Case Demos, Best Practices, and Pitfalls
- Q&A



[Prompt Engineering Guide | Prompt Engineering Guide]

Prompt Engineering Guide Courses About Search... CTRL K

- Prompt Engineering
- Introduction
- Prompting Techniques
- AI Agents
- Guides
- Applications
- Prompt Hub
- Models
  - ChatGPT
  - Claude 3
  - Code Llama
  - Flan
  - Gemini
  - Gemini Advanced
  - Gemini 1.5 Pro
  - Gemma
  - GPT-4
  - Grok-1
  - Kimi K2.5
  - LLaMA
  - Llama 3
  - Mistral 7B
  - Mistral Large
  - Mixtral
  - Mixtral 8x22B
  - OLMo
  - Phi-2

Prompt Engineering

Copy page

Prompt Engineering Guide

Prompt engineering is a relatively new discipline for developing and optimizing prompts to efficiently use language models (LMs) for a wide variety of applications and research topics. Prompt engineering skills help to better understand the capabilities and limitations of large language models (LLMs).

Researchers use prompt engineering to improve the capacity of LLMs on a wide range of common and complex tasks such as question answering and arithmetic reasoning. Developers use prompt engineering to design robust and effective prompting techniques that interface with LLMs and other tools.

Prompt engineering is not just about designing and developing prompts. It encompasses a wide range of skills and techniques that are useful for interacting and developing with LLMs. It's an important skill to interface, build with, and understand capabilities of LLMs. You can use prompt engineering to improve safety of LLMs and build new capabilities like augmenting LLMs with domain knowledge and external tools.

Motivated by the high interest in developing with LLMs, we have created this new prompt engineering guide that contains all the latest papers, advanced prompting techniques, learning guides, model-specific prompting guides, lectures, references, new LLM capabilities, and tools related to prompt engineering.

Related Learning

COURSE

Prompt Engineering for LLMs

Learn essential prompt engineering techniques to get the most out of large language models. From basic prompting to advanced strategies.

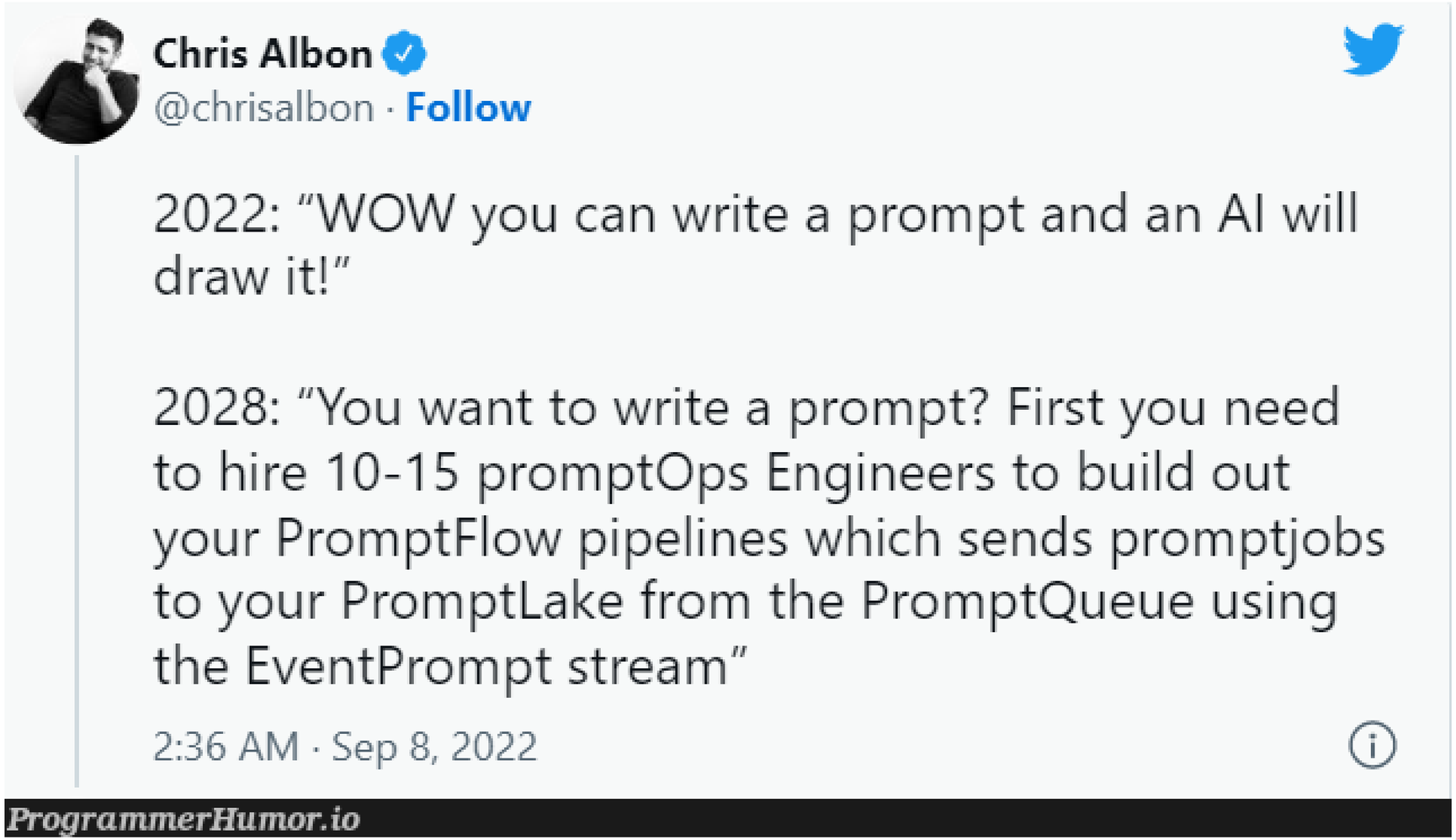
Beginner 2 hours

COURSE

Building Effective AI Agents

Learn to build effective AI agents. Covers function calling, tool integration, and debugging agentic systems.

Intermediate 5 hours


Chris Albon
@chrisalbon · Follow
2022: “WOW you can write a prompt and an AI will draw it!”
2028: “You want to write a prompt? First you need to hire 10-15 promptOps Engineers to build out your PromptFlow pipelines which sends promptjobs to your PromptLake from the PromptQueue using the EventPrompt stream”
2:36 AM · Sep 8, 2022
ProgrammerHumor.io

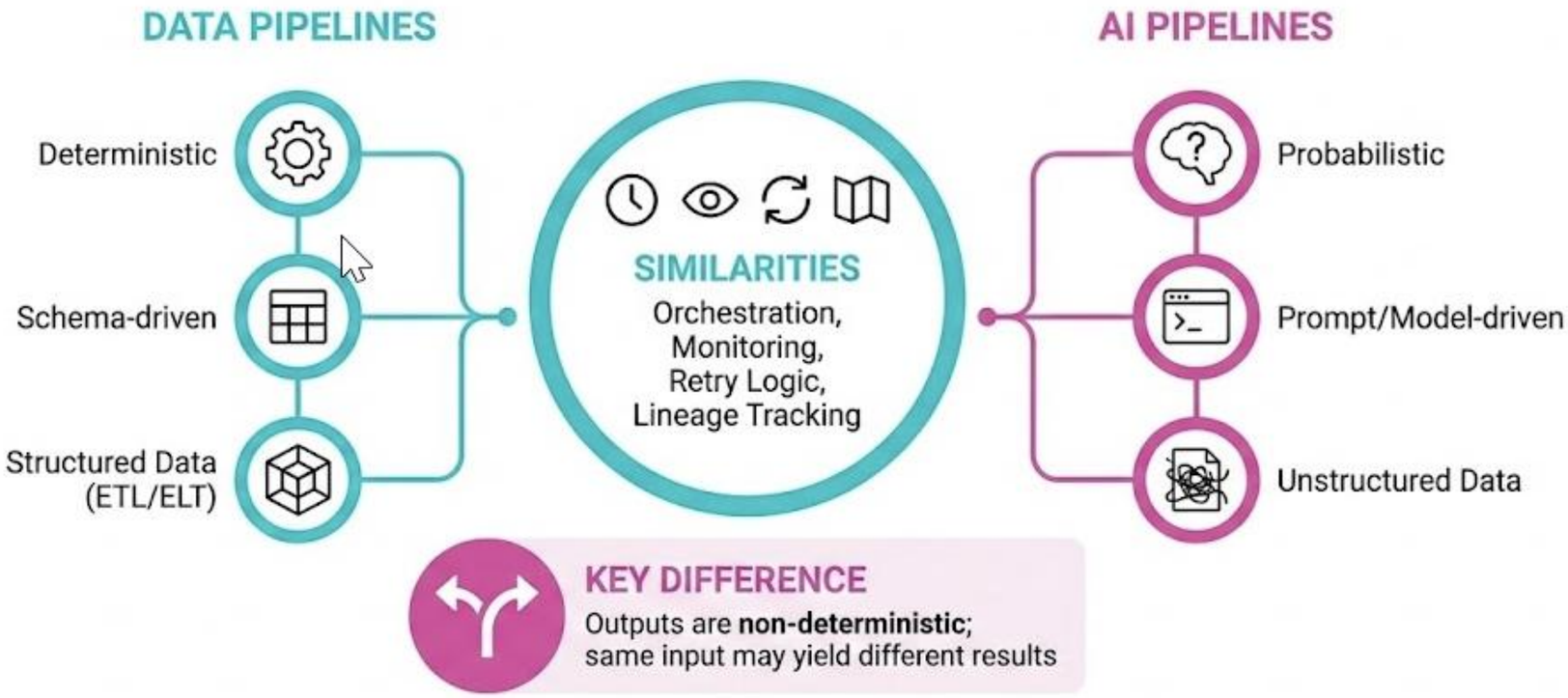
Prompt engineering is **powerful** but **brittle**. Hard to scale, harder to maintain.

What if you could declare what you want instead of how to get it?

AI Pipelines are structured workflows that chain AI operations on unstructured data.

Input → Transform → Enrich → Output. Think ETL, but for text, images, audio, and more. Non-deterministic by nature.

AI Pipelines vs. Data Pipelines



Why Fabric for AI Pipelines?

- **Built-in Azure OpenAI Endpoints**
  - GPT-5, GPT-4.1, GPT-4.1-mini available via Fabric auth (Entra ID)
  - No Azure subscription, no API keys, no resource deployment. Billed in CU seconds.

- **Notebooks + Spark Runtime = Execution Layer**
  - PySpark (distributed) or Python (lightweight) notebooks with
  - SynapseML pre-installed for distributed AI/ML

- **OneLake = Unified Storage**
  - All data stored in one storage layer. Lakehouses for structured (Tables/) and unstructured (Files/) data.
  - OneLake Shortcuts for virtual references to ADLS, S3, Sharepoint without data duplication

- **Consumption**
  - Final consumption layer using Power BI, Real Tiem Dashboards, Data Agents

Imperative: "Craft this exact prompt, parse JSON, retry on failure, handle edge cases..."

Declarative: "Extract feedback with sentiment and topics from this transcript."
Let the framework optimize the how.

Declare what you want. Let the framework figure out how.

Intent driven; engineering focused framework

Fabric AI Functions

- Built-in AI functions in Fabric notebooks and Spark runtime
  - ai.analyze_sentiment()
  - ai.extract(),
  - ai.classify()
  - ai.translate()
  - ai.summarize()
  - ai.generate_response()…
- SynapseML integration for batch AI processing
- Works on DataFrames natively: apply AI to columns at scale
- Powered by Azure OpenAI under the hood (no key management needed)

Fabric AI Functions: Patterns in Action

- **Pattern: Batch Classification**
  - `df["classification"] = df["input"].ai.classify("category1", "category2", "category3")`
  - Apply classification across millions of rows in a single Spark operation

- **Pattern: Entity Extraction at Scale**
  - `df_entities = df["text"].ai.extract("entity1", "entity2", "entity3")`
  - Extract structured fields (names, dates, amounts) from free-text columns

- **Pattern: Multilingual Summarization**
  - `df["summaries"] = df["text"].ai.summarize()`
  - Chain translate + summarize for multilingual document processing

- **Key Advantage: No API keys, no endpoints, no infrastructure. Just Python + Spark + AI.**

REAL WORLD EXAMPLE

CALL CENTER OPERATIONS

CALL ID	AGENT	DURATION	PLAN	CALL TYPE	RESOLUTION
CALL000847	Sarah M. · A014	18 min 24 sec	Standard	Account Management	Resolved

Agent Thank you for calling, this is Sarah. How can I help you today?

Customer Hi Sarah. I've been a customer for four years and honestly I've been getting really frustrated lately. I don't feel like I'm getting good value.

Agent I'm sorry to hear that. Can you tell me what's been going on?

Customer I got a mailer from T-Mobile last week. They're offering unlimited data and a free iPhone 15 if I switch. I'm paying $95 a month here and not getting nearly as much for it.

Agent I completely understand. Let me pull up your account and see what we can do.

Customer I've already decided I'm switching unless something significant changes today. I just don't see why I should stay.

Agent You've been with us since 2019 — you qualify for our loyalty rate. I can bring you to $72 a month with unlimited data, effective today.

Customer ...okay, that's a real difference. If you can do that right now, I'll stay.

☐ Competitor + offer ☐ Churn risk signal ☐ Switching reason ☐ Customer sentiment ☐ Retention action

`ai.extract`
Competitor: **T-Mobile**
Offer: free iPhone 15, unlimited data

`ai.classify`
Churn risk: **high**
Switching reason: price

`ai.analyze_sentiment`
Frustrated → open to staying after offer

`ai.generate_response`
Agent introduced loyalty offer at the right moment — customer retained

DSPy: Programming Over Prompting

- Define signatures, not prompts: declare inputs and outputs
- Optimizers tune prompts automatically based on metrics
- Composable modules: ChainOfThought, ReAct, Predict
- Built-in evaluation, assertions, and metrics
- Ideal for complex extraction pipelines in Fabric notebooks
- Works with Azure OpenAI, open-source models, and more

DSPy: The Paradigm Shift

- **Traditional Prompt Engineering**
  - Hand-craft prompts with trial and error
  - Prompts are tightly coupled to models and break on model changes
  - No systematic way to improve: manual A/B testing at best
  - Impossible to maintain across 10+ tasks in production

- **The DSPy Way: Signatures + Modules + Optimizers**
  - Signatures: Declare I/O (e.g., "document -> summary, sentiment, entities")
  - Modules: Composable building blocks (Predict, ChainOfThought, ReAct)
  - Optimizers: Automatically tune prompts using metrics on your data
  - Result: GPT-3.5 + DSPy outperforms hand-tuned GPT-4 prompts by 25%+

DSPy: Building Blocks in Practice

- **Step 1: Define a Signature (What, not How)**
  - class ExtractFeedback(dspy.Signature): "Extract feedback from transcript"
  - transcript: str = dspy.InputField() | feedback: list[dict] = dspy.OutputField()

- **Step 2: Choose a Module (Reasoning Strategy)**
  - extractor = dspy.ChainOfThought(ExtractFeedback)
  - Options: Predict (simple), ChainOfThought (reasoning), ReAct (tool use)

- **Step 3: Optimize with Data (Let DSPy Find the Best Prompt)**
  - optimizer = dspy.MIPROv2(metric=feedback_accuracy, auto="light")
  - optimized = optimizer.compile(extractor, trainset=examples)

- **Step 4: Deploy and Iterate**
  - Save, version, and track with MLflow. Swap models without rewriting prompts.

Hands-on DEMO

How to operationalize in Fabric

Pixeltable: Declarative Multimodal AI Tables

- **Works out of the box in Fabric with installation, no keys required**

- **The Core Idea: Computed Columns = AI Pipeline Steps**
  - t.add_computed_column(summary = openai.chat_completions(messages, model))
  - t.add_computed_column(entities = huggingface.ner(t.text))
  - Define once. Runs automatically on all existing and future rows.

- **Pattern: Declarative RAG in 5 Lines**
  - Create table > Add documents > Create chunks view > Add embedding index > Query
  - No separate vector DB, no ETL, no orchestration code

- **Pattern: Video Analysis Pipeline**
  - frames = pxt.create_view(t, iterator=FrameIterator(t.video, fps=1))
  - frames.add_computed_column(desc = openai.vision("Describe", image=frames.frame))

- **Key Differentiator: Full versioning, incremental updates, and multimodal-native**

Use Case Demos

- **Customer Feedback Extraction**
  - DSPy signatures to extract structured feedback from interview transcripts
- **Workload Classification Pipeline**
  - Declarative classification with DSPy optimizers and Fabric AI functions
- **PII Redaction at Scale**
  - Fabric AI functions for batch PII detection and redaction
- **Semantic Search with Embeddings**
  - Similarity search pipelines with embedding generation and vector indexing
- **Document Summarization at Scale**
  - Daft distributed dataframes for processing large document collections

DEMO

How to operationalize in Fabric

Best Practices

- Start declarative, not imperative: define what, not how -> intent driven
- Build **evaluation frameworks first**: metrics before optimization
- Use structured outputs (JSON schemas, Pydantic models)
- Version your pipelines and signatures, not just prompts
- Monitor for semantic drift: outputs can degrade silently
- Test with adversarial and edge-case inputs

- **Common Pitfalls**
  - Over-engineering prompts instead of defining clear signatures
  - Skipping evaluation: optimizing without measuring
  - Ignoring cost: batch and parallelize to manage API spend

Resources

- DSPy Documentation - dspy.ai
- Daft Documentation - getdaft.io
- Pixeltable Documentation - pixeltable.github.io
- Fabric AI Functions - Microsoft Fabric | Microsoft Learn
- SynapseML for AI in Fabric - Microsoft Fabric | Microsoft Learn
- Sandeep Pawar – fabric.guru


Q&A


MICROSOFT FABRIC
THANK YOU

Daft: Distributed DataFrames for AI

- Distributed DataFrame engine purpose-built for multimodal AI
- Native column types: Image, Tensor, Embedding, FixedSizeList
- SQL-like API with lazy evaluation and automatic query optimization
- Python-native UDFs without serialization overhead (unlike PySpark)
- Scales from laptop to cluster with Ray integration
- Reads Parquet, CSV, JSON, Delta Lake, images, and cloud storage

- **Think of it as: Pandas API + Spark distribution + first-class AI data types**

Daft: Distributed AI on Unstructured Data

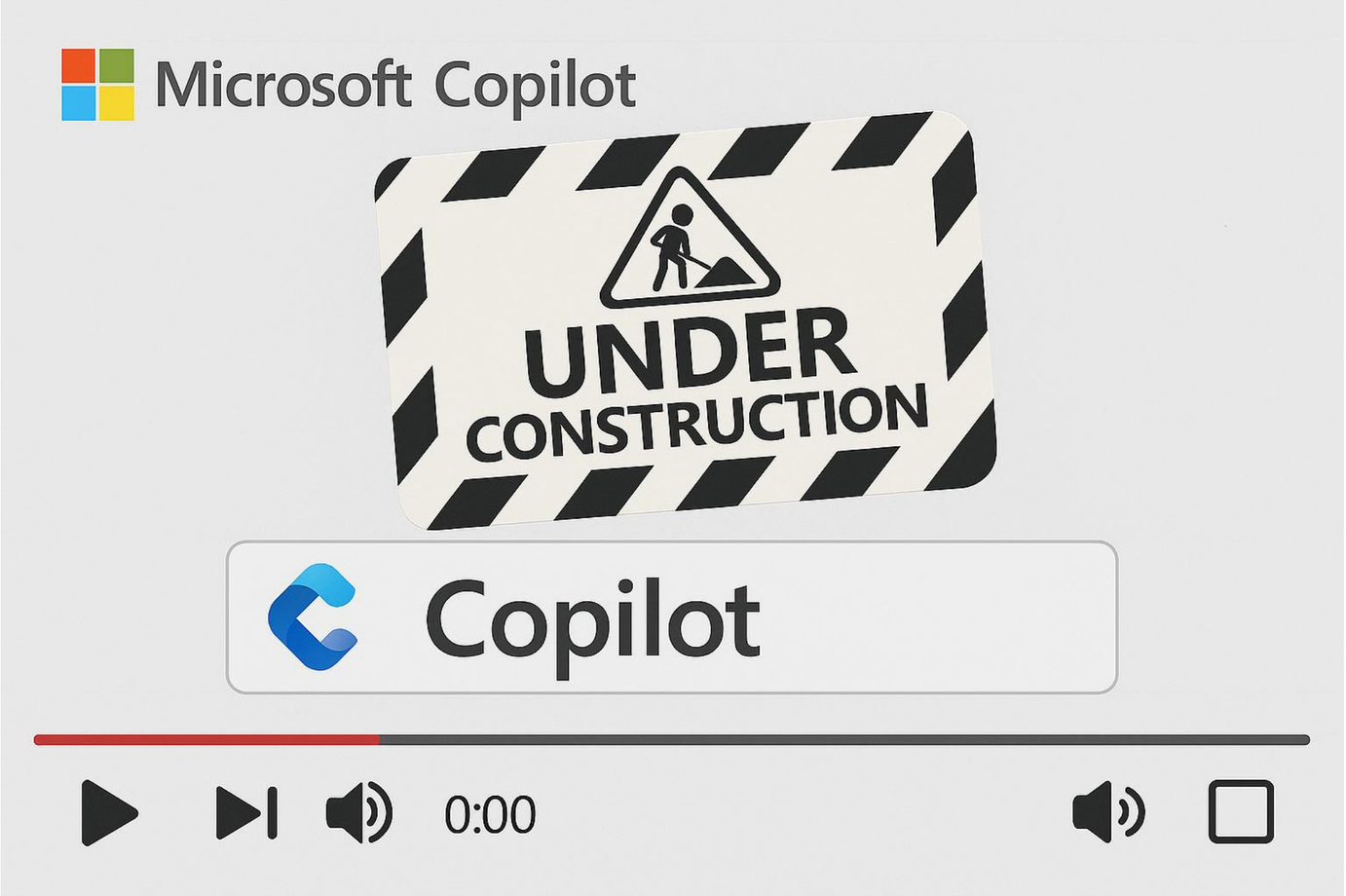
- **Pattern: Image Classification Pipeline**
  - df = daft.read_parquet("s3://images/").with_column("label", classify_udf(col("image")))
  - Process millions of images with lazy evaluation and distributed execution

- **Pattern: Embedding Generation at Scale**
  - df = df.with_column("embedding", embed_udf(col("text")))
  - Generate embeddings across billions of text records with Ray integration

- **Why Daft over Spark for AI Workloads?**
  - Native Python UDFs without serialization overhead
  - First-class support for images, tensors, and embeddings as column types
  - Lazy execution with automatic query optimization
  - Reads Parquet, CSV, JSON, images, and Delta Lake natively

DEMO

How to operationalize in Fabric



Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

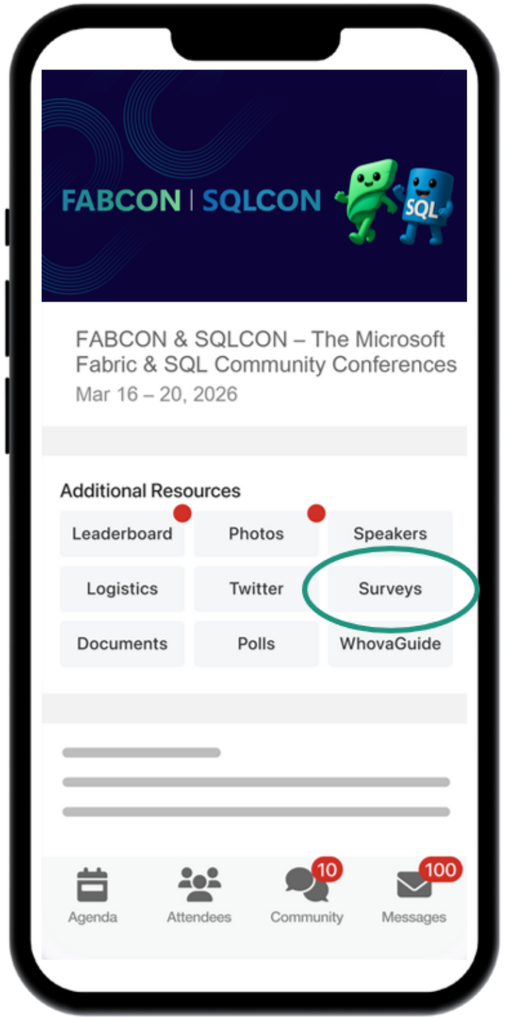
Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!



Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

Request your voucher by March 23, 2026.

https://aka.ms/fabcon/cert100