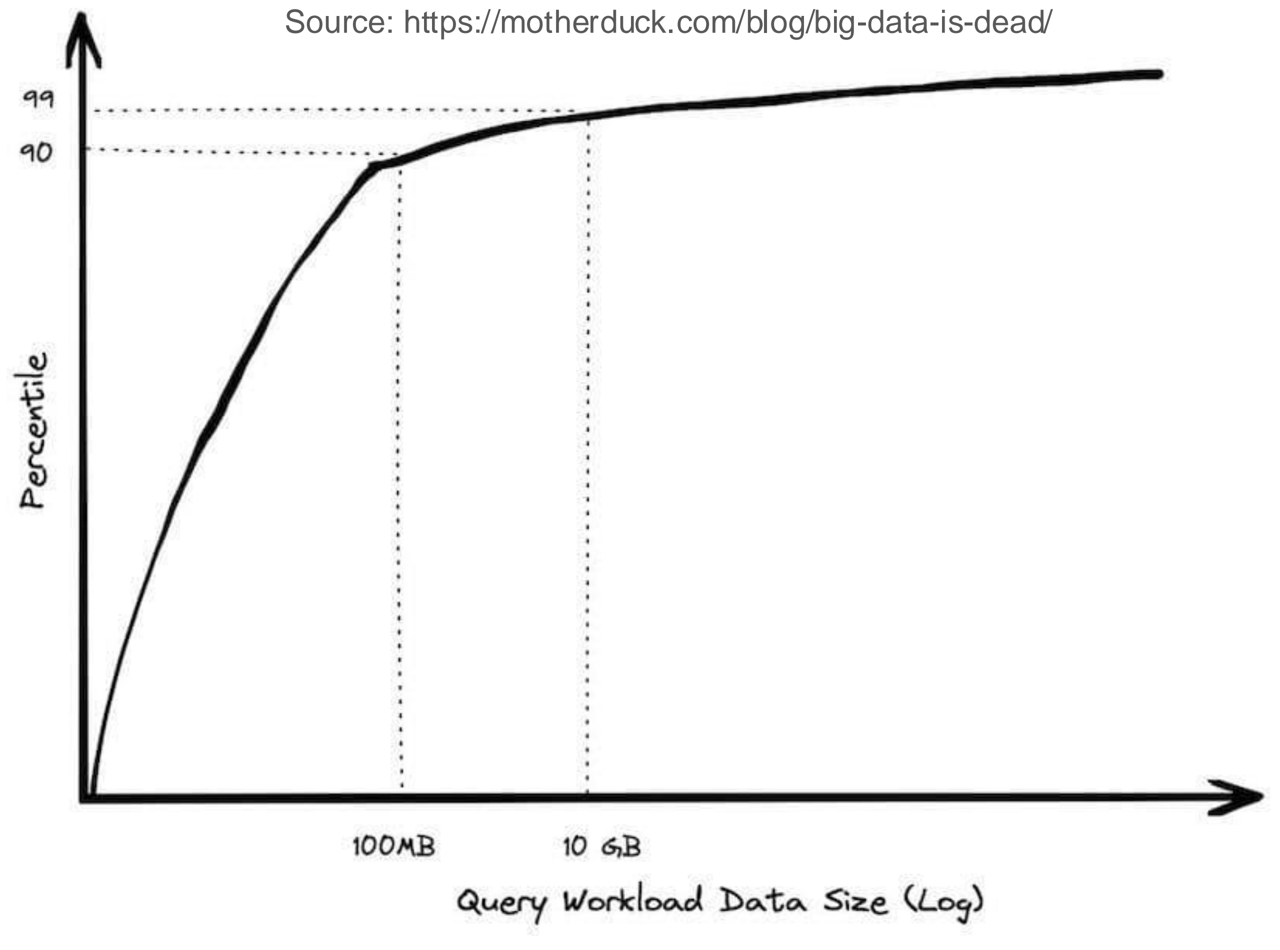
Ducklake

A journey to integrate DuckDB with Unity Catalog

Frank Mbonu, Diederik Greveling

Big data is dead?



in-process analytical database designed for fast query execution, especially suited for analytics workloads.

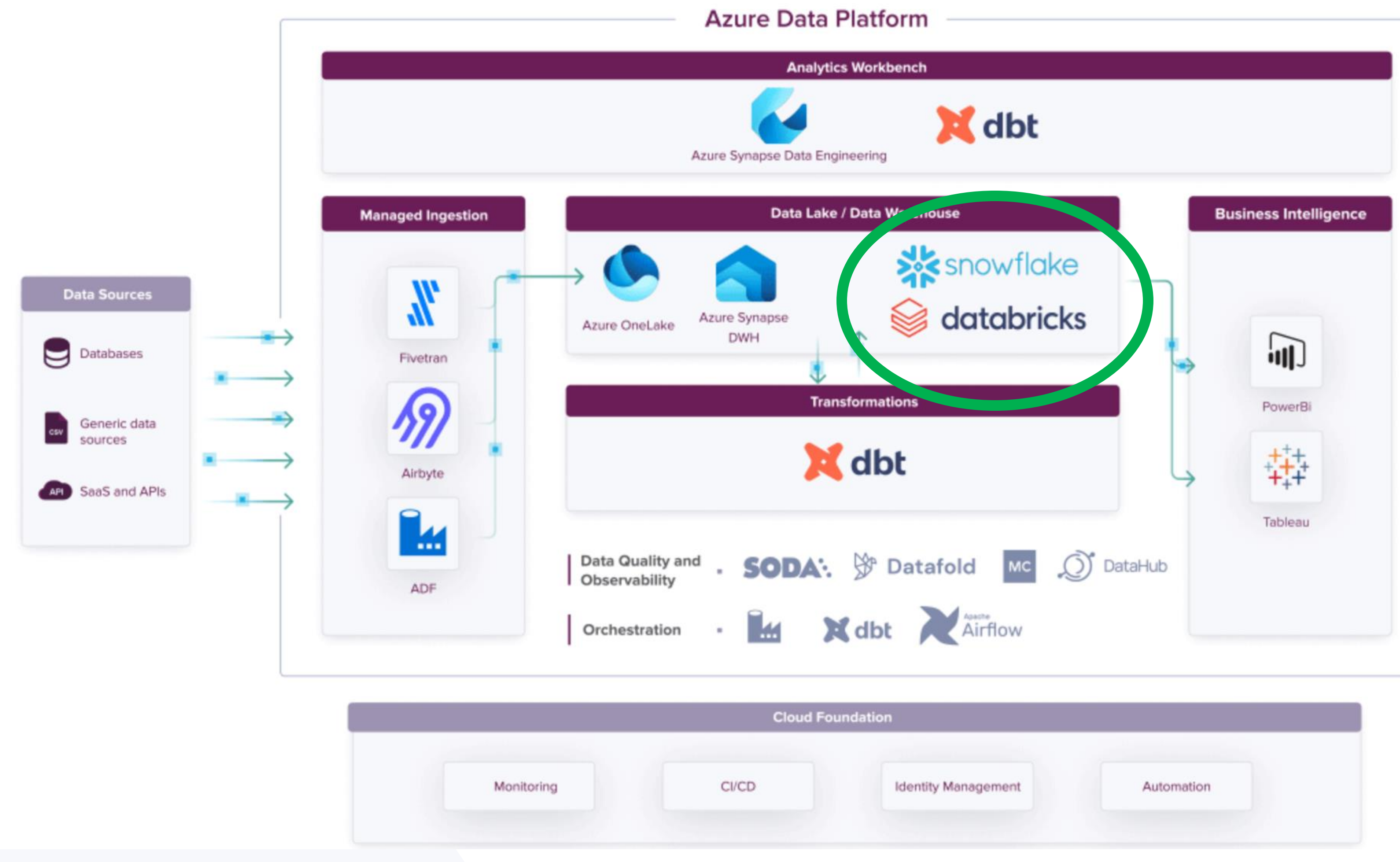
Big data is dead?

Source: https://motherduck.com/blog/big-data-is-dead/

Key takeaways

- Most Organizations Don’t Have "Big Data”
  - “The majority of companies, even large enterprises, typically have data warehouses smaller than a terabyte”
- Workloads Are Smaller Than Data Sizes
  - “Most analytical queries process only a small fraction of the total data stored. For example, 90% of BigQuery queries process less than 100 MB of data”
- The Big Data Frontier is Shrinking
  - “As hardware improves, fewer workloads require distributed systems. A single machine today can handle what required thousands of nodes a decade ago”

For whom is this interesting?



Distributed compute (e.g. Spark) makes a lot of sense for large datasets.

But for smaller ones (< 1Tb) it might not be the best fit

Introducing Ducklake

in-process analytical database designed for fast query execution, especially suited for analytics workloads.

centralized governance, features like access controls and data lineage. Open source since the summer of 2024.

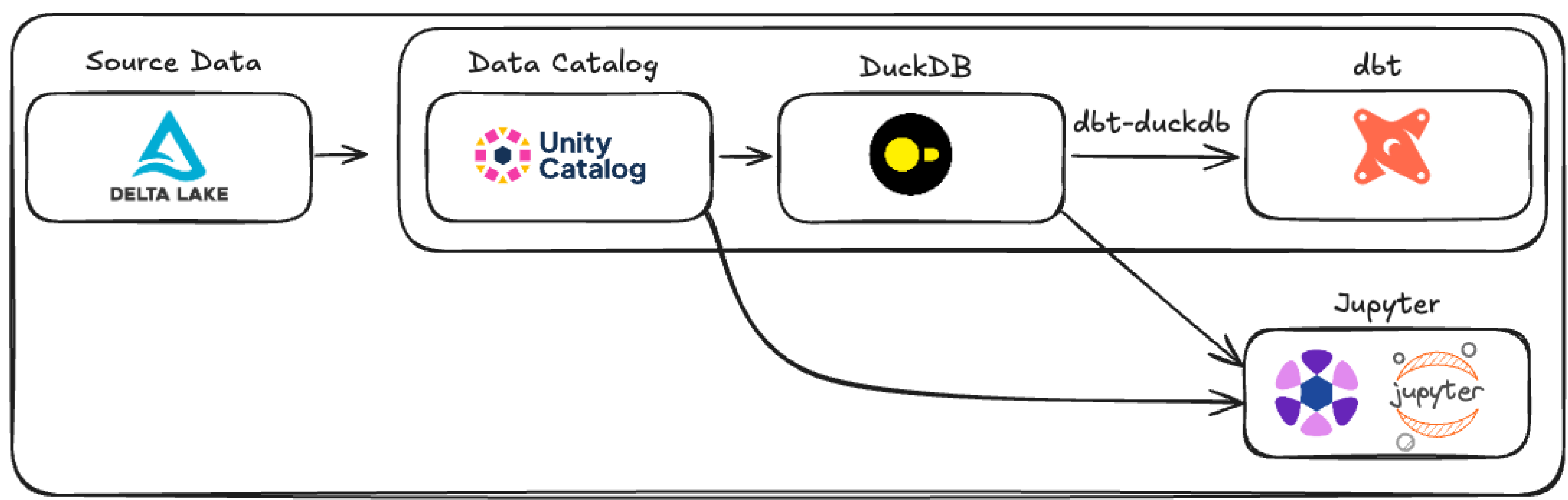
=

Ducklake: Let’s combine DuckDB with the Unity Catalog through the DuckDB Unity Catalog extension

Ducklake high-level design

Requirements:

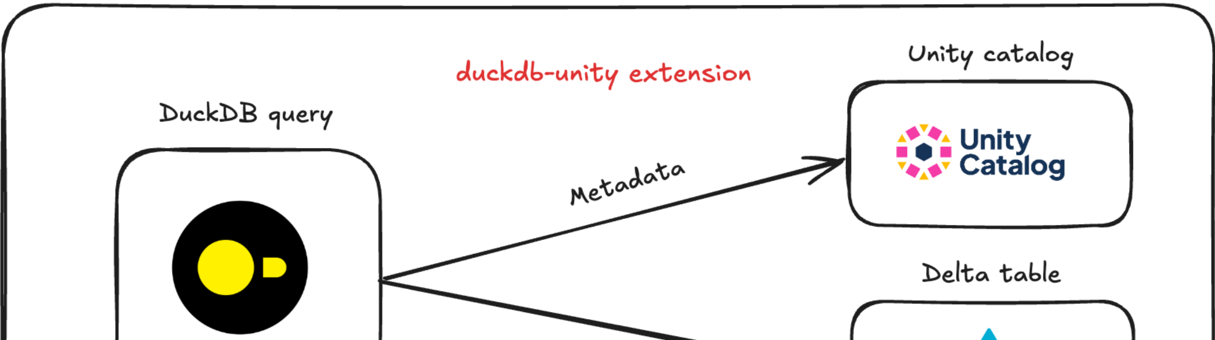
- Access Control
- Support for ACID transactions
  - To ensure data is complete, correct, conflict-free
- Read-write integration with an open data format
- Notebook interface for interactive development
- Storage decoupled from compute
- Full dbt integration



Challenges encountered

Lack of full native compatibility

- The DuckDB delta extension depends on the **delta-kernel-rs***, which currently only supports reads (and blind appends)
- The DuckDB unity extension currently does not support all CRUD operations (e.g. table create, schema create)



*The **Delta Kernel (Java or Rust (C and C++ bindings))** is a set of libraries that provides high-level APIs for interacting with Delta Lake tables designed to make Delta Lake integration easier and more efficient.

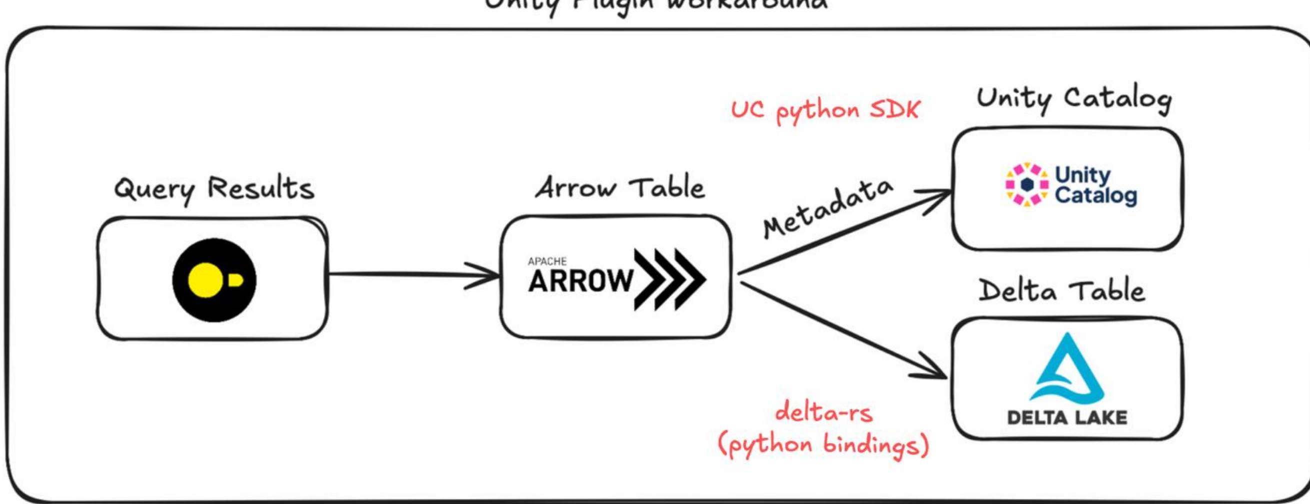
*DuckDB is written in C++ hence the delta-kernel-rs is used for building the delta connector

The workaround

Creating our own dbt-duckdb* plugin

Delta-rs
- Writing the result in-memory arrow table to a delta table

UC SDK
- Create schemas
- Create tables
- Get temporary storage credentials (AWS)



**dbt-duckdb is a dbt adapter for DuckDB*

A `dbt build` will

1. **Compile** our dbt models into executable SQL
2. **Execute** the compiled code against DuckDB
3. **Convert** the query results to a PyArrow table
4. **Create** the unity schema if it doesn't exist
5. **Create** the unity table if it doesn't exist
6. **Write** the PyArrow table to a Delta table

Ducklake in action

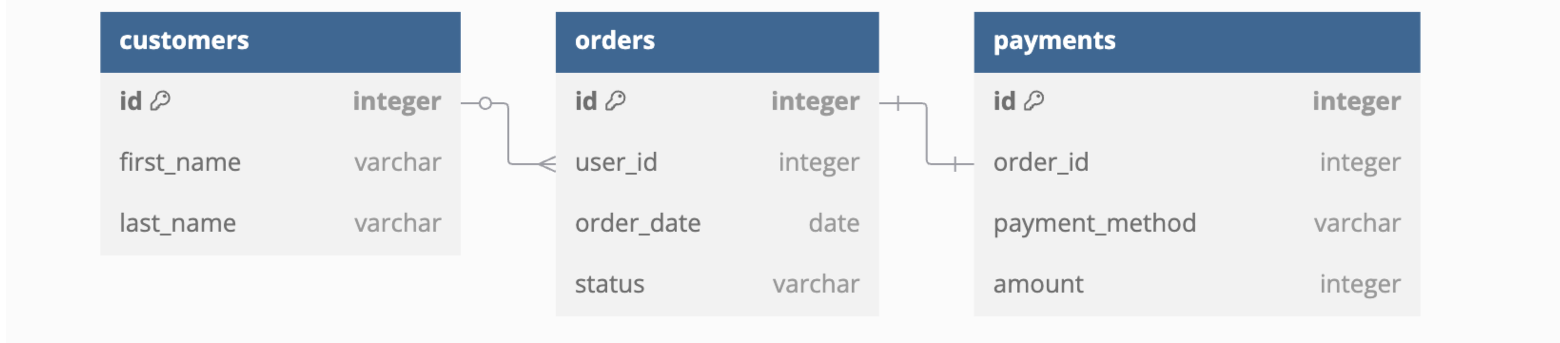
Now that we have created an integrated setup with DuckDB and Unity Catalog

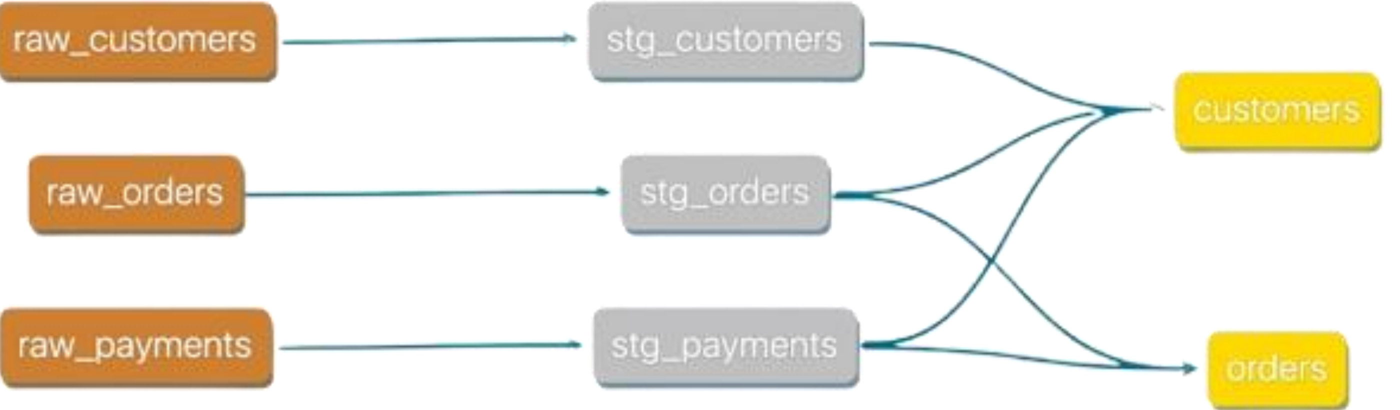
Let's put our Ducklake to test...

First, we need some data to work with. Since we're using dbt in our setup, the jaffle shop seems appropriate to use.

The Jaffle shop is a fictional e-commerce store often used for dbt demos. It transforms raw csv data into `customer` and `order` models.

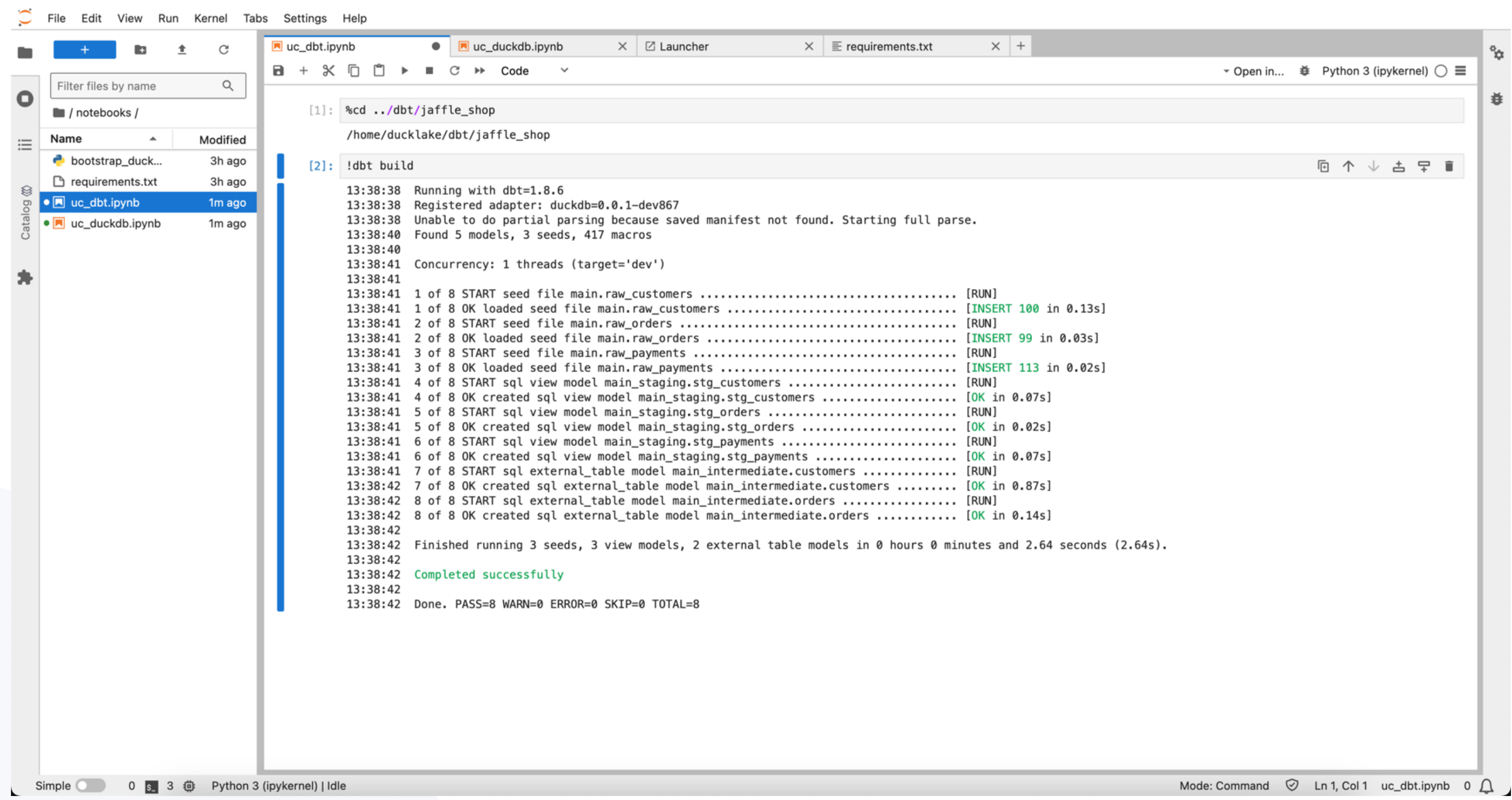
In our example the gold models are **materialized as tables** in Unity Catalog





Ducklake in action

First, let's populate our unity catalog with the jaffle shop materialized tables using `dbt build`

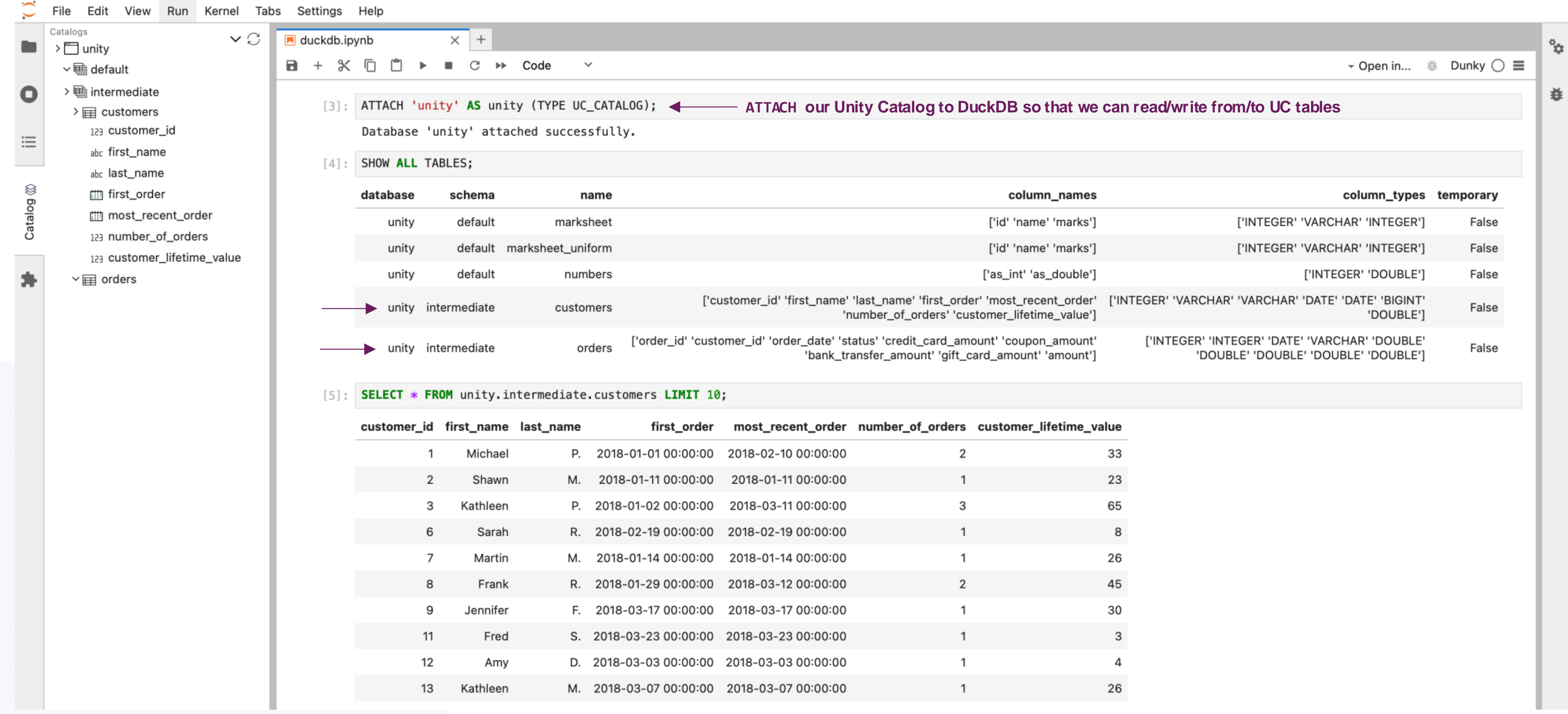


This will

1. **Compile** our dbt models into executable SQL
2. **Execute** the compiled code against DuckDB
3. **Convert** the query results to a PyArrow table
4. **Create** the unity schema if it doesn't exist
5. **Create** the unity table if it doesn't exist
6. **Write** the PyArrow table to a Delta table

Ducklake in action

Now that our Unity Catalog is populated, let's try to query our materialized tables



DuckLake works—no quacks, just results 🦆

Next steps

- 🔐 Implement RBAC (UC v0.2.0)
- ➡️ Support schema evolution
- ☁️ Deploy Ducklake in the cloud (AWS)
- ✨ Improve UI
- 🔌 Make Ducklake pluggable
  - Choose your engine
  - Choose your catalog
- ✍🏼 Write more blogs about our journey

Read our blog post if you haven't already!