EUROPEAN MICROSOFT FABRIC Community Conference

STOCKHOLM 24-27 SEPTEMBER 2024

JOIN THE CONVERSATION

#FABCONEUROPE

FABRIC IN A DAY FROM ZERO TO HERO

WILL CRAYGER

OWNER & LEAD ARCHITECT, LUCID BI, UNITED STATES

Will Crayger

- Be better than you were yesterday
- Data Architecture
- Data Engineering
- Lucid Control Framework

Career

- **Founder** Data & Analytics
- **Solution Principal** Data & Analytics
- **Director, Data & Analytics** Data & Analytics
- **Senior Consultant** Data & Analytics

My Toolbelt

Azure · Data Lake · Synapse · Power BI · Fabric

My Last 2 Years

September 2022
- Invited to Project Fabric Kickoff
- Enterprise Voice Program (EVP)

October 2022
- Project Fabric Private Preview release

December 2022
- Created the first metadata-driven orchestration system for Fabric

May 2023
- MSFT Build: Fabric released for Public Preview
- Launched first client Fabric POC

July 2023
- Invited to work directly with the Microsoft Product Group (PG) to iterate through Fabric development

August 2023
- Designed and launched Fabric in a Day training series

January 2024
- Invited to co-present session related to Fabric DirectLake connectivity at first annual Fabric Conference in Las Vegas

March 2024
- Founded Lucid BI, an independent consultancy

FABRIC IN A DAY: MORNING AGENDA

9:00 – 9:30 AM – Fabric Overview

- Intro to Fabric
- What is OneLake
- Fabric Capacity Overview
- Capacity and Licensing Cost

9:30 – 10:15 AM – Working with the Lakehouse

- Lakehouse Architecture and Design
- Overview of SQL Endpoints
- Creating your first Lakehouse
- Load data to your Lakehouse
- Complete Lab 1

10:30 – 11:00 AM – Morning Break

11:00 – 11:50 AM – Transforming Data with Dataflows gen2

- Understanding Dataflows
- Navigating Power Query
- Complete Lab 2

11:50 – 12:45 PM – Data Orchestration with Pipelines

- Understanding pipelines
- Intro to parameterization
- Using the Copy Data Tool
- Complete Lab 3

12:45 – 2:00 pm – Lunch

FABRIC IN A DAY: AFTERNOON AGENDA

2:00 – 2:50 PM – Intro to Spark and Notebooks

- Understanding Spark and how Notebooks are used
- Overview of Spark configuration in Fabric
- Import and execute a Notebook in Fabric
- Complete Lab 4

2:50 – 3:15 PM – Working with the Warehouse

- Overview of Fabric warehouse
- Warehouse loading patterns

3:15 – 3:45 PM – Afternoon Break

3:45 – 4:15 PM – Working with the Warehouse Continued

- Complete Lab 5

4:15 – 4:45 PM – Analyzing and Visualizing Data

- Building your first semantic model and reports
- Complete Lab 6

4:45 – 5:00 PM – Wrap up and final questions

END-TO-END ANALYTICS WITH FABRIC

TODAY'S DATA READINESS

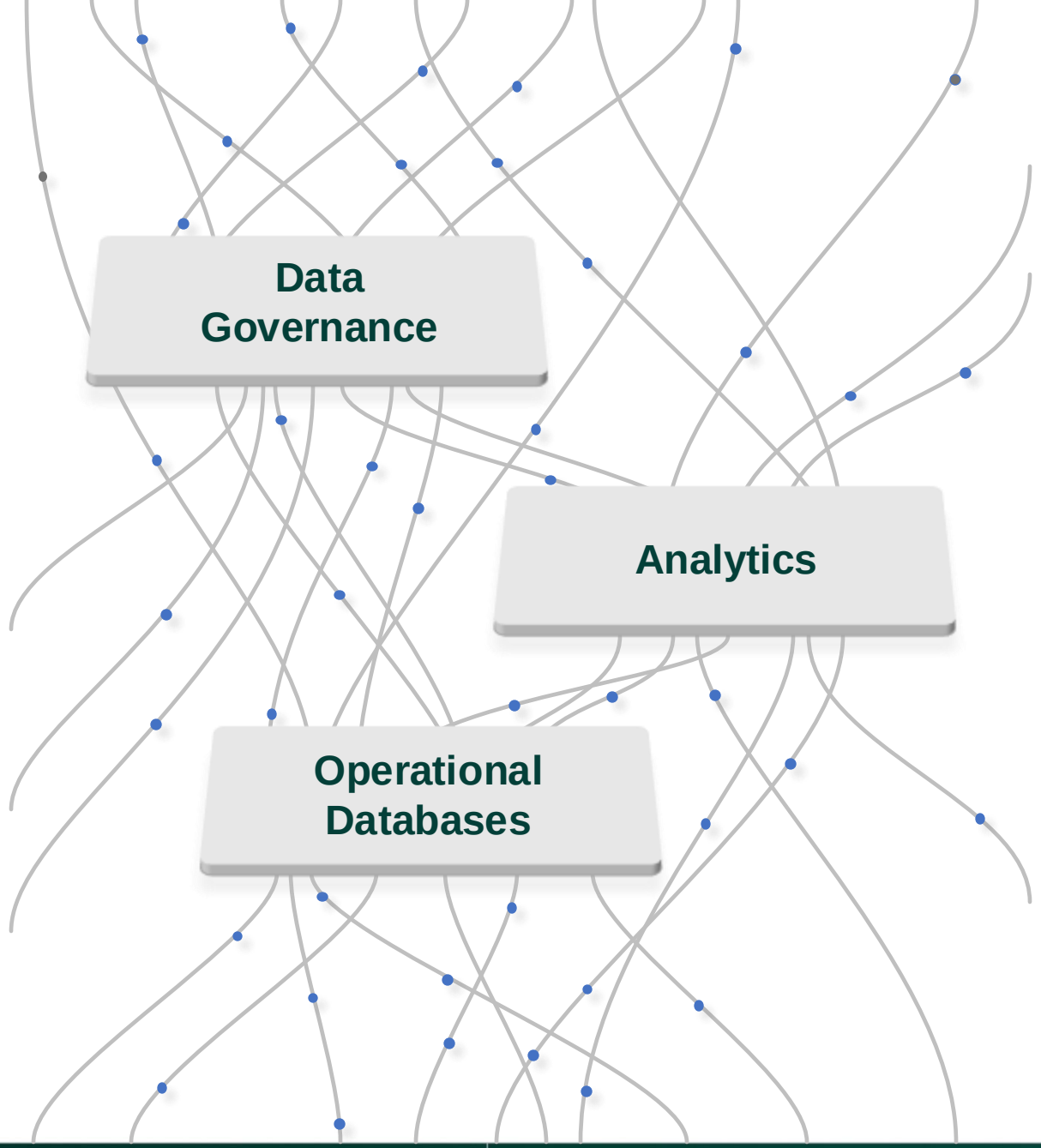
What **data** do I have?

Is it **trustworthy?**

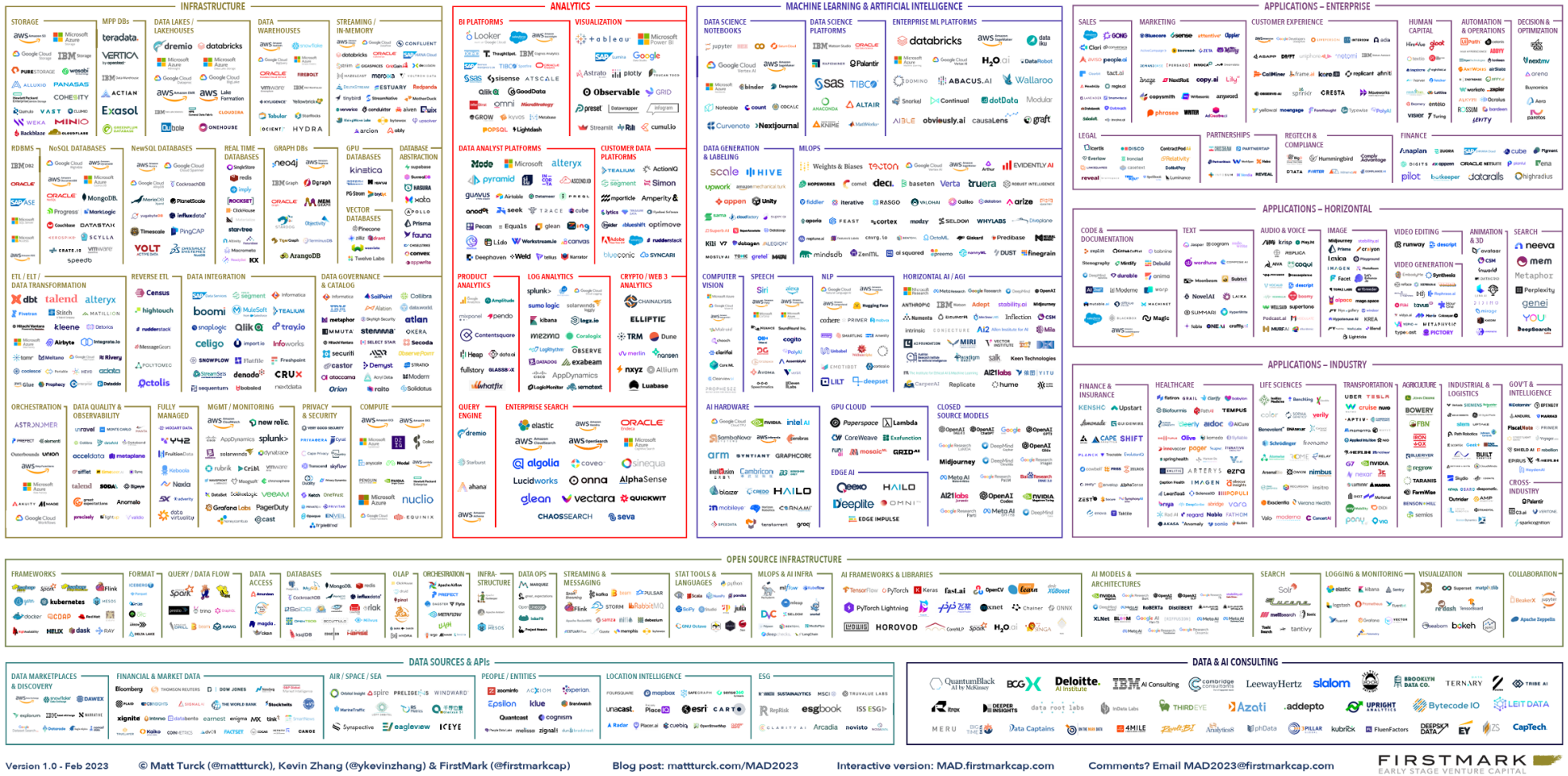
Can people **access the data needed** to make the right decisions?

How can I enable faster **business insights?**

What's my **compliance** exposure?



FRAGMENTATION OF THE MODERN DATA STACK



Version 1.0 - Feb 2023 © Matt Turck (@mattturck), Kevin Zhang (@ykevinzhang) & FirstMark (@firstmarkcap) Blog post: mattturck.com/MAD2023 Interactive version: MAD.firstmarkcap.com Comments? Email MAD2023@firstmarkcap.com

FIRSTMARK EARLY STAGE VENTURE CAPITAL

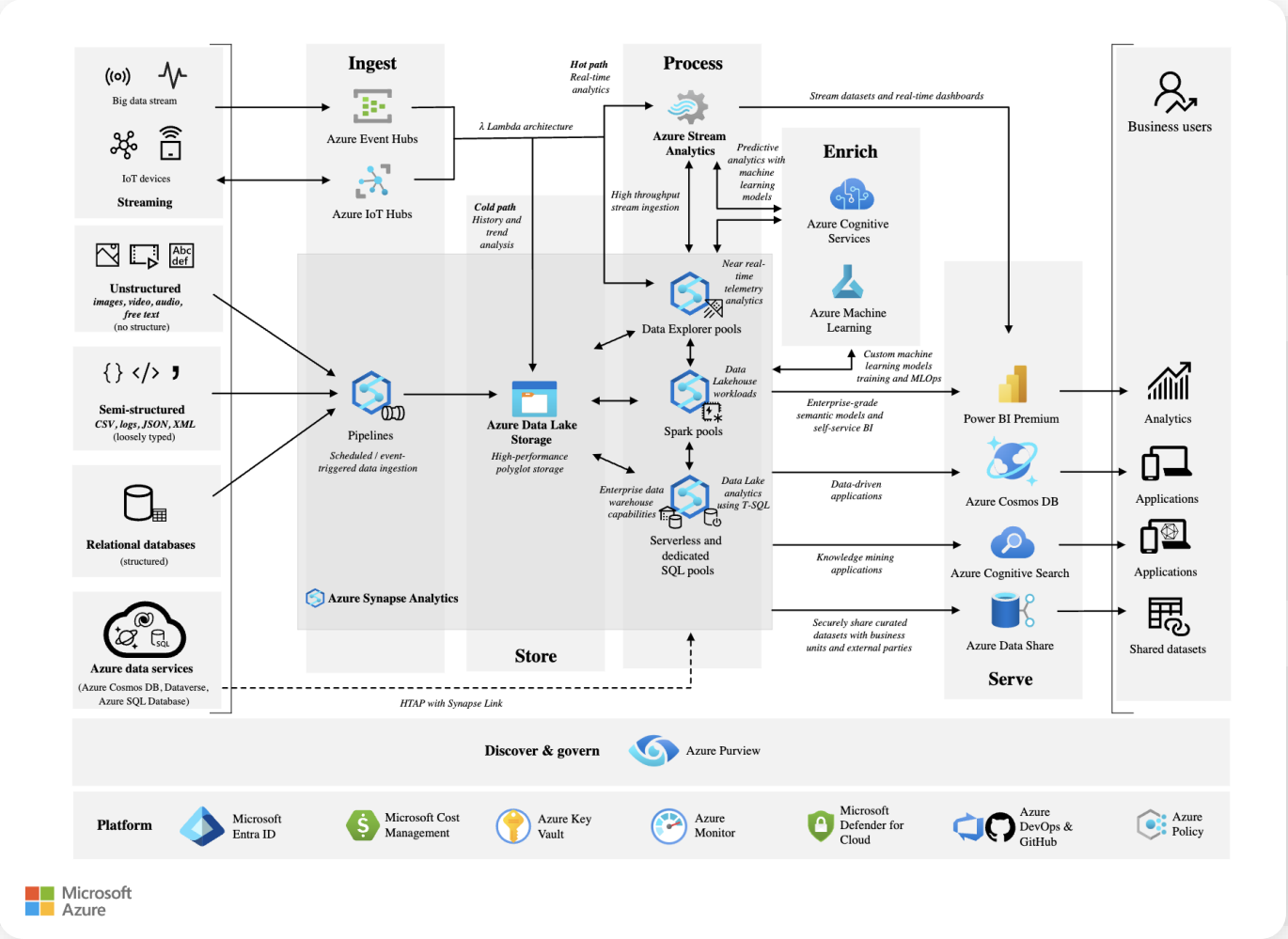
2023 ML, AI, and Data Landscape

SCALABLE ANALYTICS ARE COMPLEX & FRAGMENTED

Every analytics project has many subsystems

Products often come from multiple vendors

Integration at scale across products is complex, fragile and expensive



EXISTING MICROSOFT ANALYTICS SUITE

- Many Products
- Different Experiences
- Proprietary and Open
- Dedicated and Serverless
- PaaS and SaaS
- Different Business Models
- Steep Learning Curves
- Deep Expertise Needed
- High Integration Effort

Purview

Azure AI

Power BI

Synapse DW

Kusto

Synapse Spark

Data Factory

THE EVOLUTION OF MICROSOFT ANALYTICS SUITE: MICROSOFT FABRIC

Data Integration | Data Lake | Data Engineering | Data Warehouse | Real Time Analytics | Data Science | Business Intelligence | Governance

Unified analytics fabric

End-to-end analytics platform

INTRODUCTION TO ONELAKE AND FABRIC CAPACITY

INTRODUCTION TO ONELAKE

Lake-centric architecture that offers a unified environment for data professionals and business leaders to work together

Merges storage from various regions and cloud platforms into one logical lake, minimizing data duplication or movement

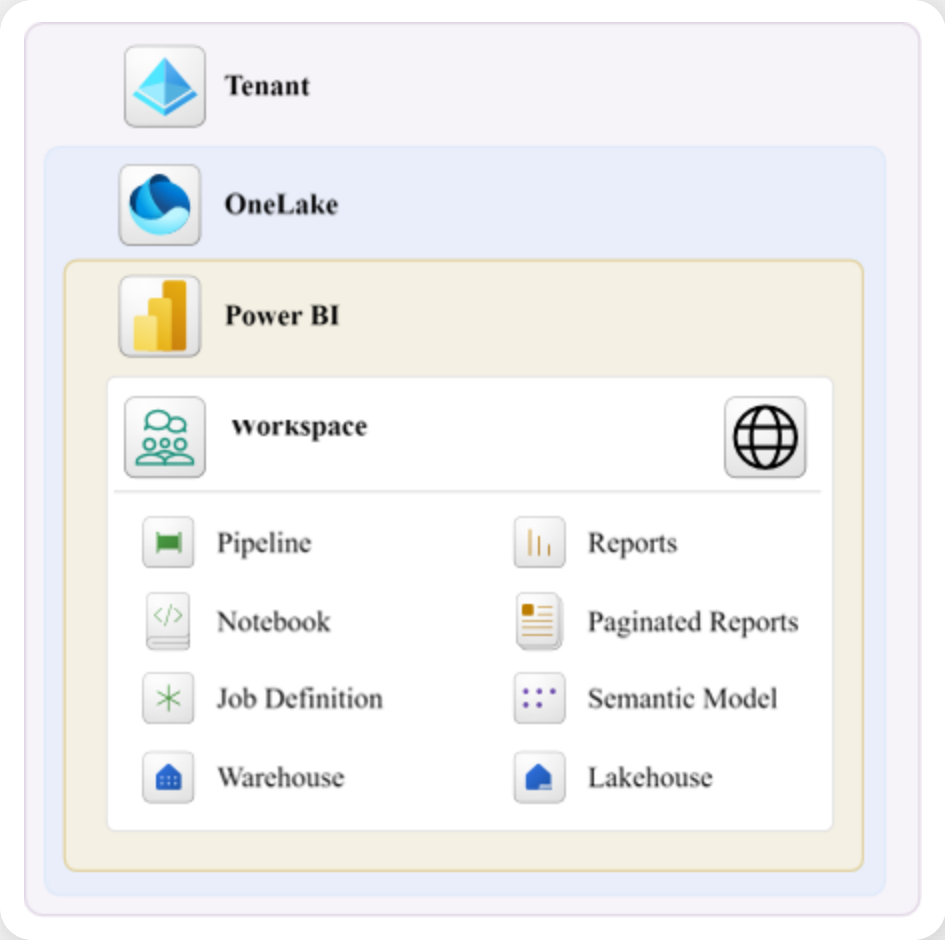
Just as Office apps are naturally connected to OneDrive, Fabric's compute engines function with OneLake to read and write data

ORGANIZATIONAL STRUCTURE OF ONELAKE

OneLake is tenant-wide, meaning there is only a single OneLake per tenant

Within OneLake, data connectivity across workspaces within the tenant is automatically enabled allowing for cross-workspace workloads

Workspaces can be thought of as folders within OneLake



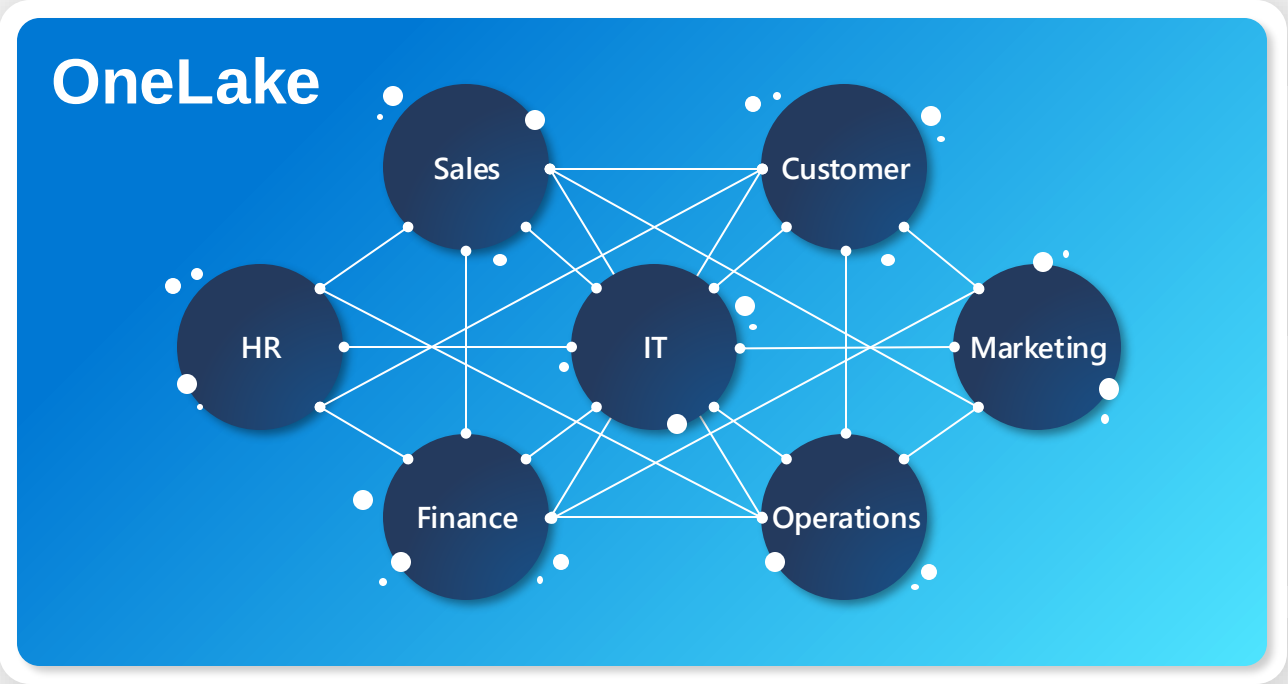
ONELAKE FOR ALL DOMAINS

Workspaces and artifacts for different data domains, contribute to building the same data lake

Without data movement, data from different domains can be analyzed, blended and transformed together

Data is secured and governed in one place while remaining easily discoverable and accessible to all who should have access across the organization

Data can be certified by domain experts to enabling trust for data which is discovered

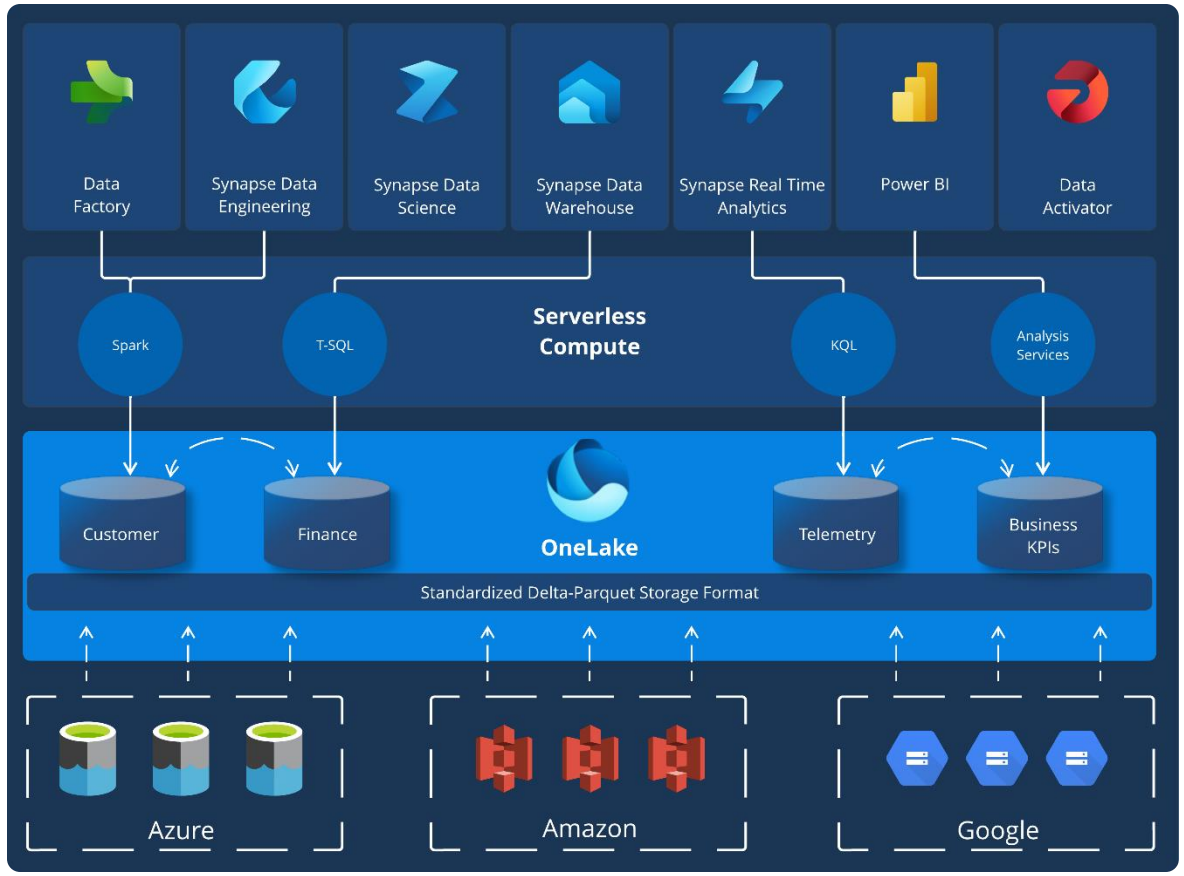


ORGANIZATIONAL STRUCTURE OF ONELAKE

OneLake is constructed on Azure Data Lake Storage (ADLS)

Supports structured and unstructured data formats such as Delta, Parquet, CSV, and JSON

For structured data, Fabric's analytics engines save data in the delta-parquet format, ensuring smooth interaction across all engines



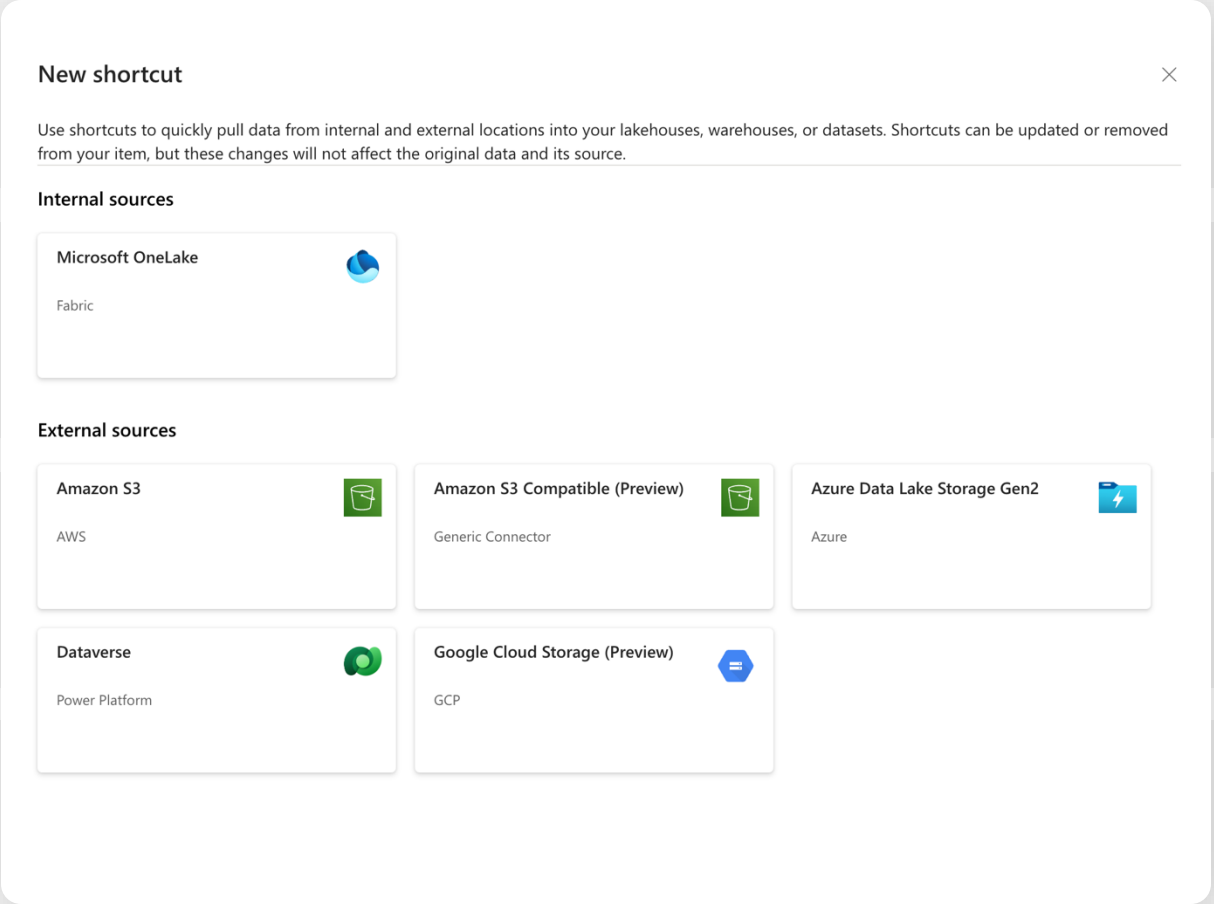
CREATING ONE COPY WITH SHORTCUTS & MIRRORING

With **shortcuts**, data throughout OneLake can be composed together without requiring data movement

Shortcuts are references (pointers) directing to other files or storage points for data visualization

Shortcuts also support data already in Azure, and other clouds, without creating a copying, making **OneLake a multi-cloud data lake**

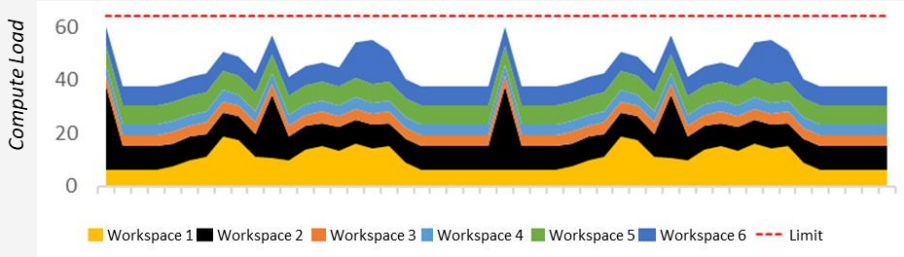
Mirroring provides a low-latency replication option for data from sources such as Azure SQL, Azure Cosmos, and Snowflake database environments



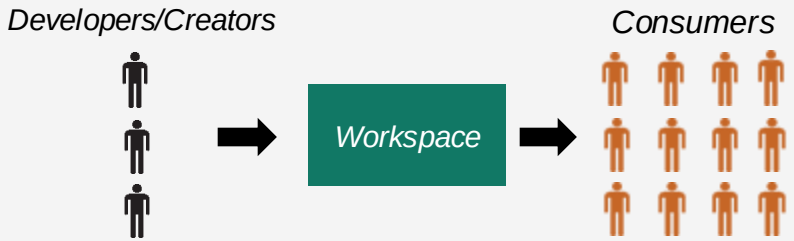
ORGANIZATIONAL STRUCTURE OF ONELAKE

A single capacity is providing the compute power for all the Fabric's workloads

A single capacity typically supports dozens of separate projects simultaneously, each managed in its own workspace

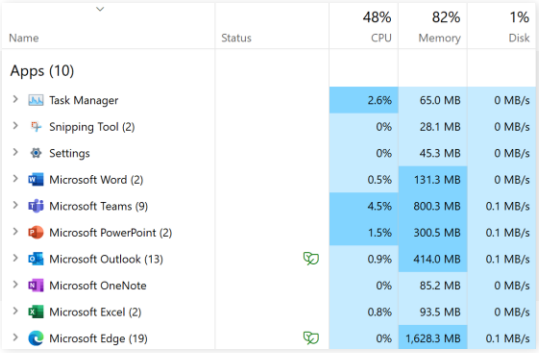


Developers will share a workspace where collaborative development and consumption at scale is managed



UNDERSTANDING FABRIC COMPUTE CAPACITIES

Personal Computing	When you purchase a PC, you choose the number of CPU cores. The more CPU cores the more load the PC can handle.	The CPU cores are dynamically shared across all applications with no need to pre-allocate by app.	The total consumption of the CPU across all apps cannot exceed the number of cores. CPU overload causes throttling.



Fabric Capacities	Fabric uses Capacities with “capacity units”. More CU’s means heavier compute loads are possible. Capacities can be scaled up/down as needed.	The capacity units are dynamically shared across all the Fabric workloads, with no pre-allocation necessary.	The total consumption of the capacity across all the workloads cannot exceed the capacity units provisioned, or throttling will occur.

CAPACITIES ARE A SAAS RESOURCE

Always On Auto-pilot

Capacities are a self-managed system that alleviates day-to-day or hour-to-hour resource management by admins.

Providing complete transparency into the usage of the capacities at any grain: projects/workspaces, artifacts, queries, users and jobs

Eliminate Workload Management

Large jobs can be scheduled at any time without impacting any other jobs running

Smoothing automatically prevents spikey loads from creating temporal overload of the system – thereby you are purchasing for average load vs. max load.

Always Peak Performance

Bursting allows all jobs and queries to always run at peak performance regardless of any other compute activity happening on the capacity

Auto scale can automatically adjust the compute units provisioned to the capacity

WORKING WITH THE FABRIC LAKEHOUSE

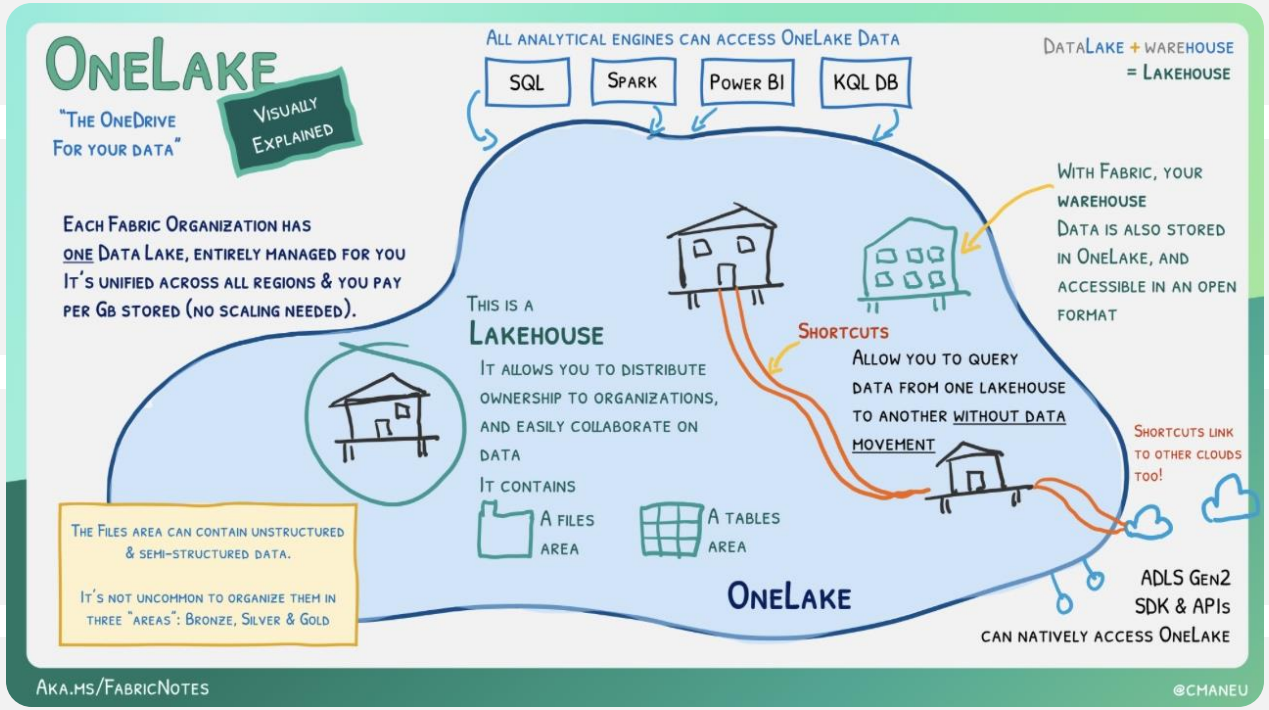
OVERVIEW OF A LAKEHOUSE

Data platform designed to store, manage, and analyze both structured and unstructured data

Appears as a database built on a data lake using Delta tables

Merges SQL-based analytics of a data warehouse with a data lake's scalability and flexibility

Stores all data formats and can be used with various analytics tools as well as programming languages

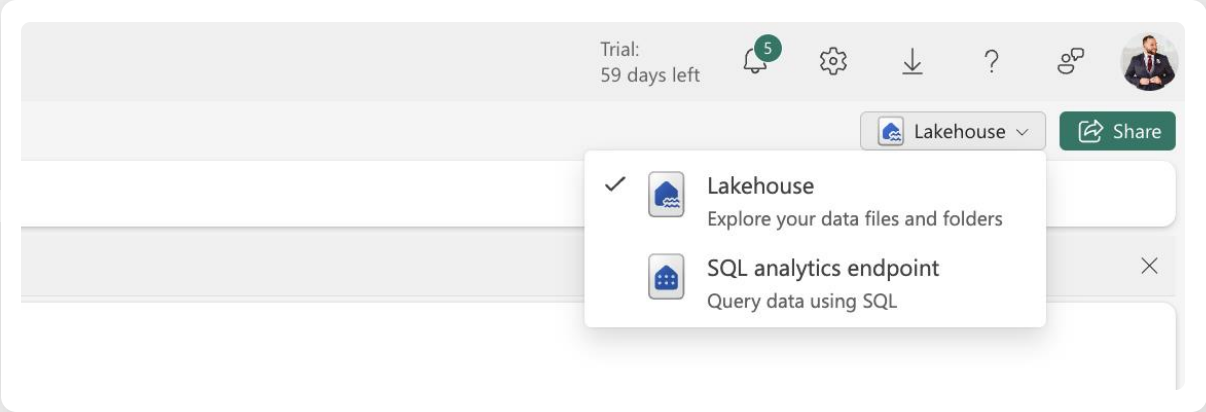


SETTING UP AND NAVIGATING THE LAKEHOUSE

Each Lakehouse in a Fabric-enabled workspace consists of three main elements:

Lakehouse: Storage and metadata section used to engage files, folders, and tables

Semantic Model (default): Auto-generated data model derived from the tables in the Lakehouse



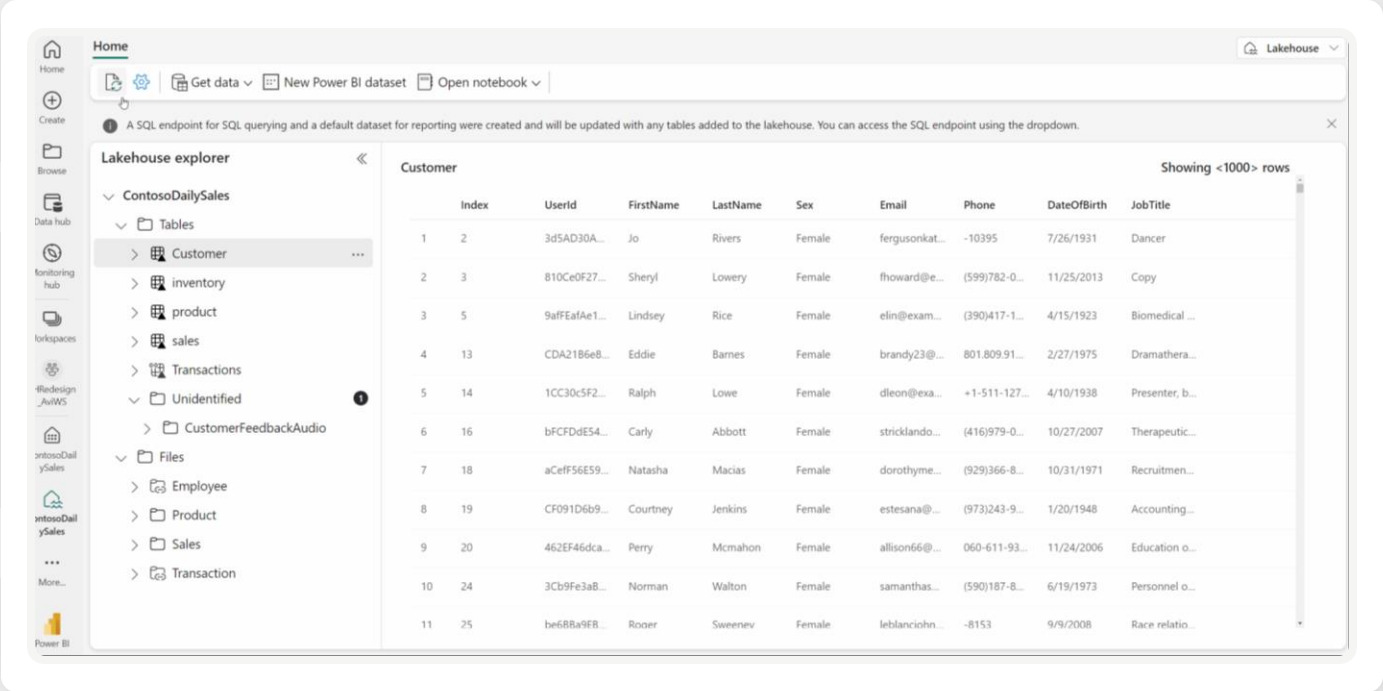
SQL Analytics Endpoint: Provides a read-only SQL interface, allowing users to connect and query data using T-SQL

USING THE LAKEHOUSE EXPLORER

Allows for loading, browsing, and previewing data elements within the Lakehouse

Lakehouse Explorer: Provides a preview of tables and files in the lakehouse

Ribbon: Enables creation of notebooks and semantic models as well as the capture new data using shortcuts, dataflow gen2, pipelines, and event streams



TRANSFORMING AND ANALYZING DATA IN FABRIC LAKEHOUSE

LOADING PATTERNS WITH LAKEHOUSE

1

Directly upload local files or folders to the Lakehouse

2

Utilize Power Query Online with Dataflows (Gen2) to import and transform data from various sources, providing a visual transformation representation

3

Use Data Factory Pipelines to copy data and coordinate data processing tasks such as orchestrating Spark, Dataflows, and other control flow logic

4

Establish connections to external storage solutions, such as OneLake, Azure Data Lake Store Gen2, Amazon S3 and S3 compatible sources for seamless integration into your Lakehouse

TRANSFORMATION PATTERNS WITH LAKEHOUSE

Every Fabric lakehouse is equipped to utilize Spark pools for data processing in files and tables using Scala, PySpark, or Spark SQL

Developers can use interactive Notebooks to directly read from, transform, and write data to the lakehouse in the form of tables or files

On-demand or scheduled Spark Job Definitions can also employ the Spark engine for data processing within the lakehouse

Read-only SQL analytics endpoints allow you to execute T-SQL commands to query, filter, aggregate, and explore data stored in the lakehouse tables

GUIDED LEARNING LABS 1

Provisioning a Lakehouse

Upload CSV data file, create a managed table

Exploring Lakehouse, SQL Endpoint, and Dataset

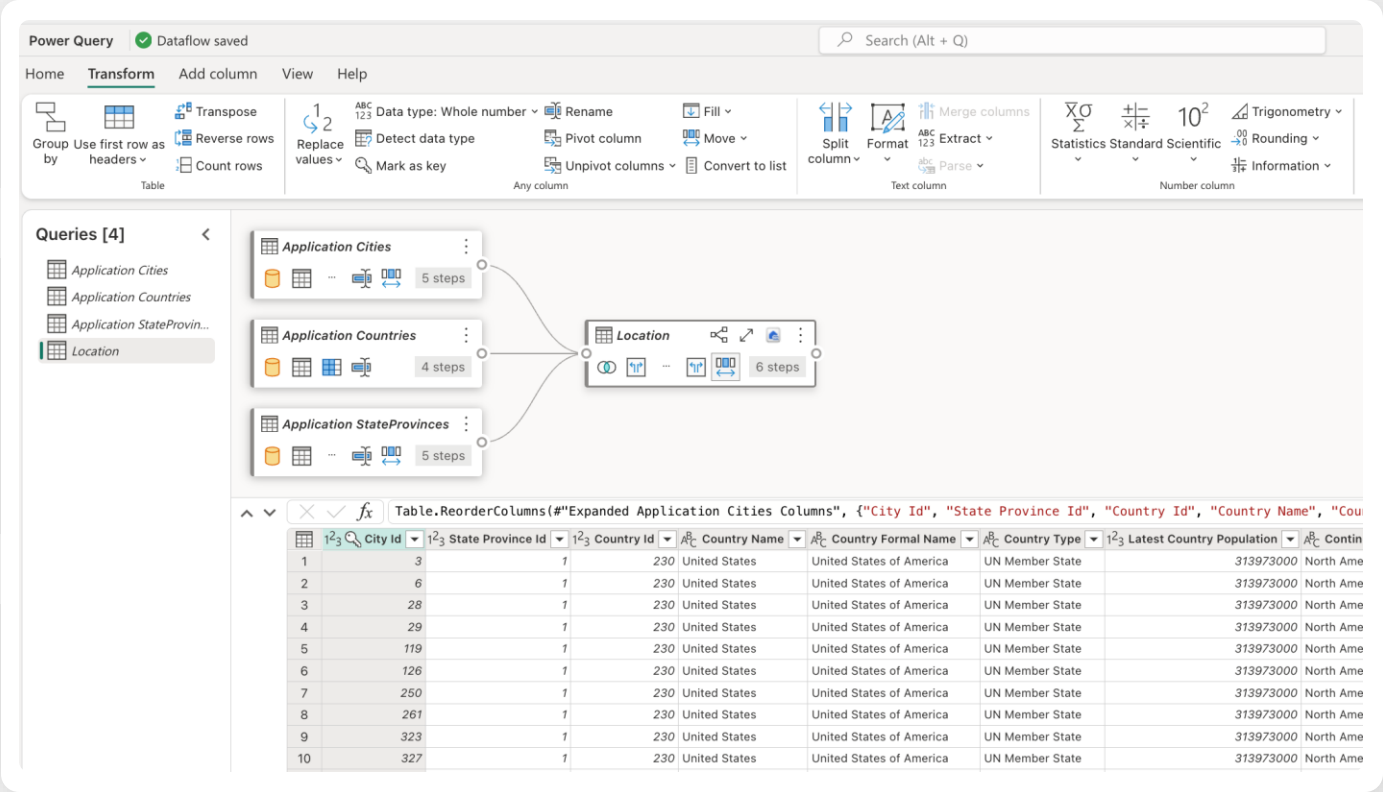
PREPARING DATA WITH DATAFLOWS GEN2

FABRIC DATAFLOWS GEN2

Cloud-based, Power Query, data prep tool for extracting, transforming, and loading

Currently has native support for over 145 source connectors, significantly more than pipelines

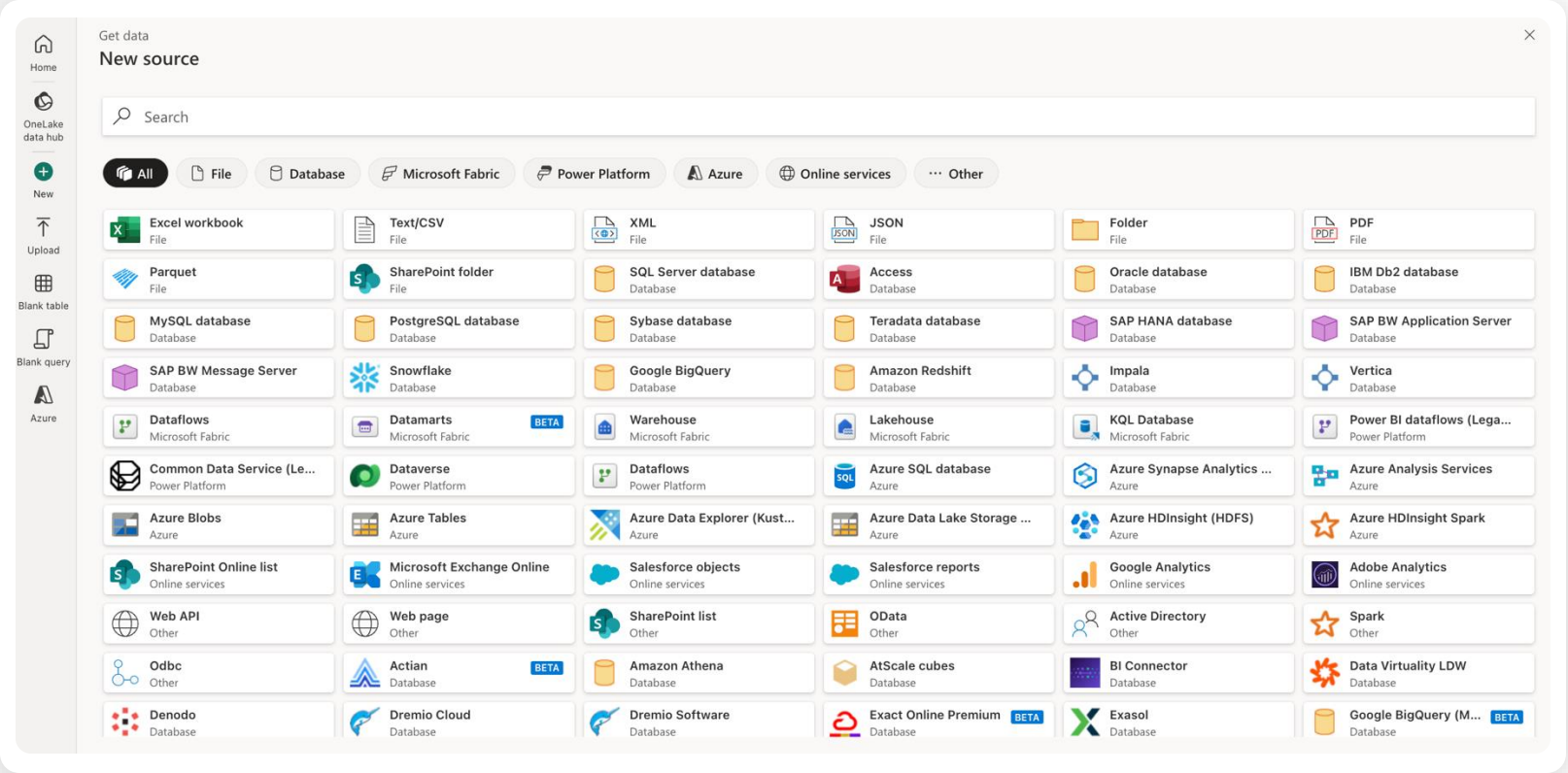
Includes all standard Power Query transformations as well as support for destinations providing similar functionality to pipelines



	City Id	State Province Id	Country Id	Country Name	Country Formal Name	Country Type	Latest Country Population	Contin
1	3	1	230	United States	United States of America	UN Member State	313973000	North Ame
2	6	1	230	United States	United States of America	UN Member State	313973000	North Ame
3	28	1	230	United States	United States of America	UN Member State	313973000	North Ame
4	29	1	230	United States	United States of America	UN Member State	313973000	North Ame
5	119	1	230	United States	United States of America	UN Member State	313973000	North Ame
6	126	1	230	United States	United States of America	UN Member State	313973000	North Ame
7	250	1	230	United States	United States of America	UN Member State	313973000	North Ame
8	261	1	230	United States	United States of America	UN Member State	313973000	North Ame
9	323	1	230	United States	United States of America	UN Member State	313973000	North Ame
10	327	1	230	United States	United States of America	UN Member State	313973000	North Ame

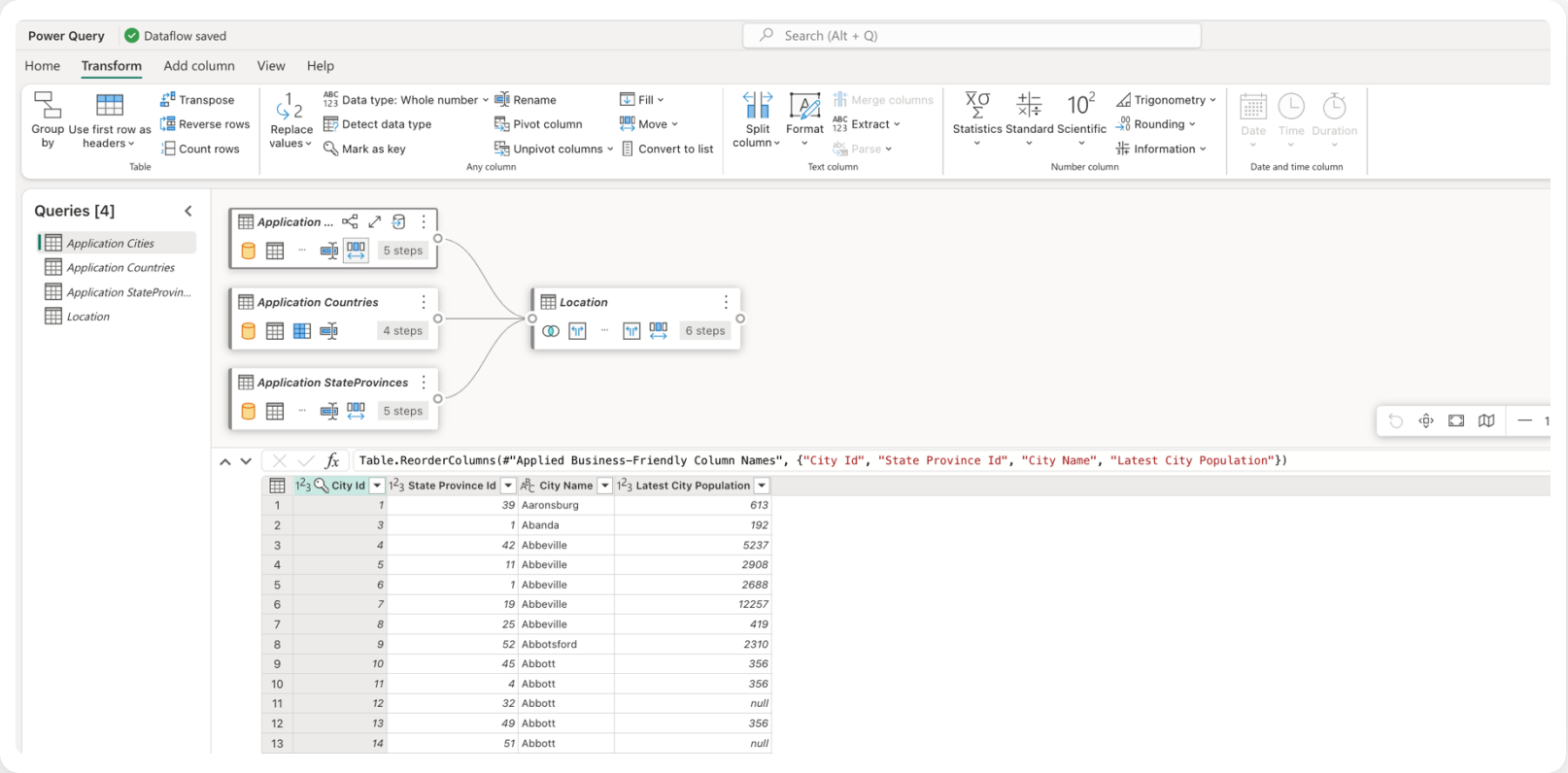
COMMON SOURCE & DESTINATION CONNECTORS

- SQL Server
- Oracle
- SAP BW / SAP HANA
- Azure Synapse
- Azure Databricks
- Data Lake Store gen2 and Azure Blobs
- SharePoint
- Salesforce
- Google Analytics
- Profisee
- Fabric Lakehouse
- Fabric Warehouse
- Fabric KQL database



COMMON TRANSFORMATION TASKS

- Filtering and sorting rows
- Pivoting and unpivoting
- Merging and appending queries
- Splitting and conditional splits
- Replacing values and removing duplicates
- Managing columns (adding, renaming, reordering, deleting)
- Ranking
- Write to destination



QUERIES PANE, DIAGRAM VIEW, DATA PREVIEW PANE

Queries Pane:

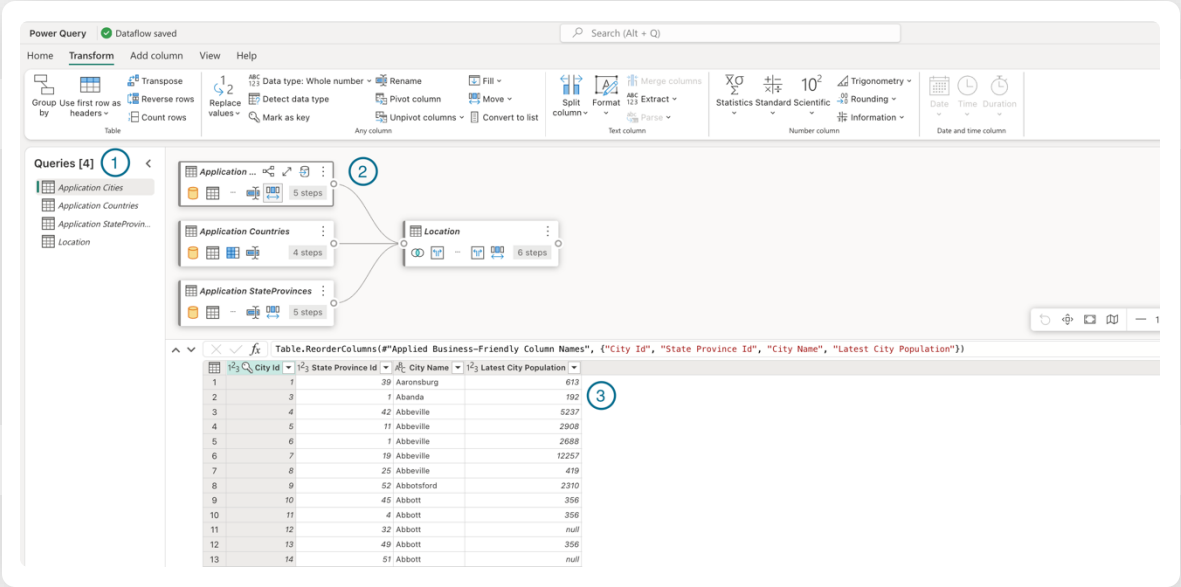
- Displays different data sources, referred to as queries
- Offers options like renaming, duplicating, referencing, and enabling load

Diagram View

- Provides a visual representation of how data sources relate, and the transformations applied

Data Preview Pane

- Showcases a subset of data, allowing users to identify necessary transformations and their impacts
- Allows for drag-and-drop interaction



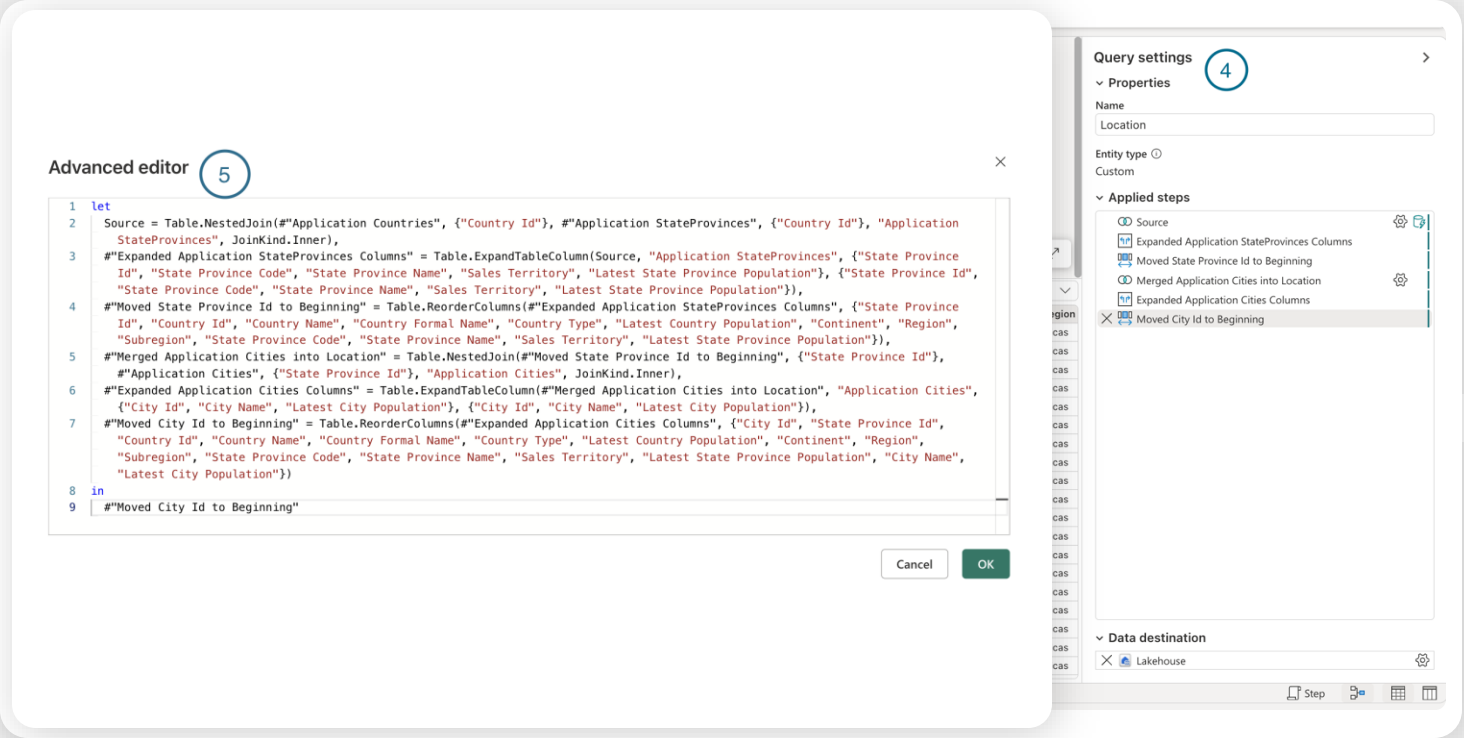
QUERY SETTINGS PANE

Queries Settings:

- Each transformation corresponds to a step
- Numerous steps might be associated with each query
- Optional Data Destination tab, unique to dataflows gen2

Advanced Editor:

- Displays the M code being produced from each applied step
- Interface where M code can be directly written/applied



DATAFLOW GEN2 BENEFITS AND CONSIDERATIONS

Benefits:

- Allows data to be extracted from source once and can be reused
- Simplifies data integration with a low-code interface
- Ensures data consistency and quality by enabling pre-load data cleansing and transformation
- Has a Fast Copy mode to address performance limitations during data extraction
- Can be integrated into Fabric pipeline workloads

Considerations:

- Performance of the Power Query engine can degrade as dataset size increases causing a bottleneck in workloads
- Fast Copy mode has several limitations including source type, file format, transformations being applied, and whether a gateway is required

GUIDED LEARNING LAB 2

Create Dataflow (Gen2)

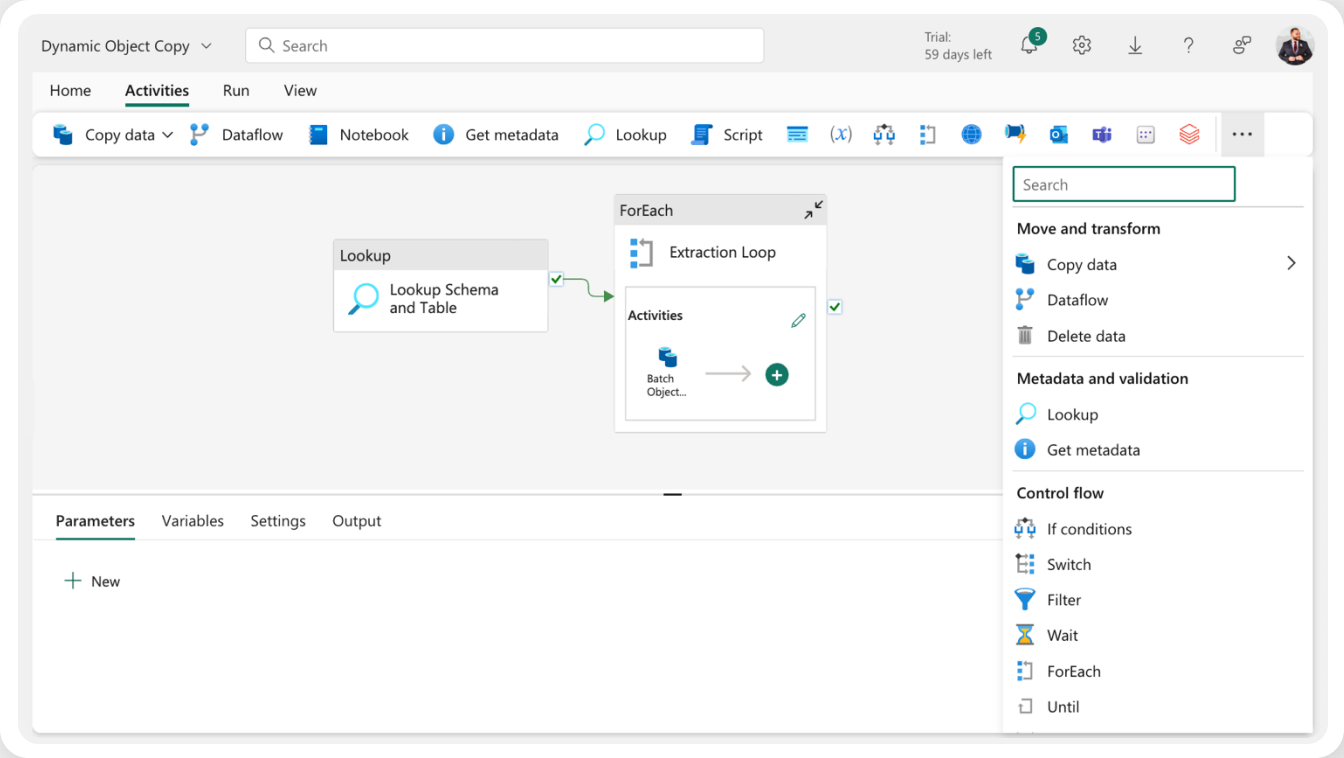
Build flow in PowerQuery to pull data from source and move to Lakehouse

DATA ORCHESTRATION WITH DATA PIPELINES

UNDERSTANDING FABRIC PIPELINES

Data pipelines orchestrate and automate ETL/ELT processes using pipeline activities

- Extracting data from sources
- Triggering transformation activities
- Writing data to a destination



Leverages the same architecture as Azure Data Factory and Synapse integration pipelines

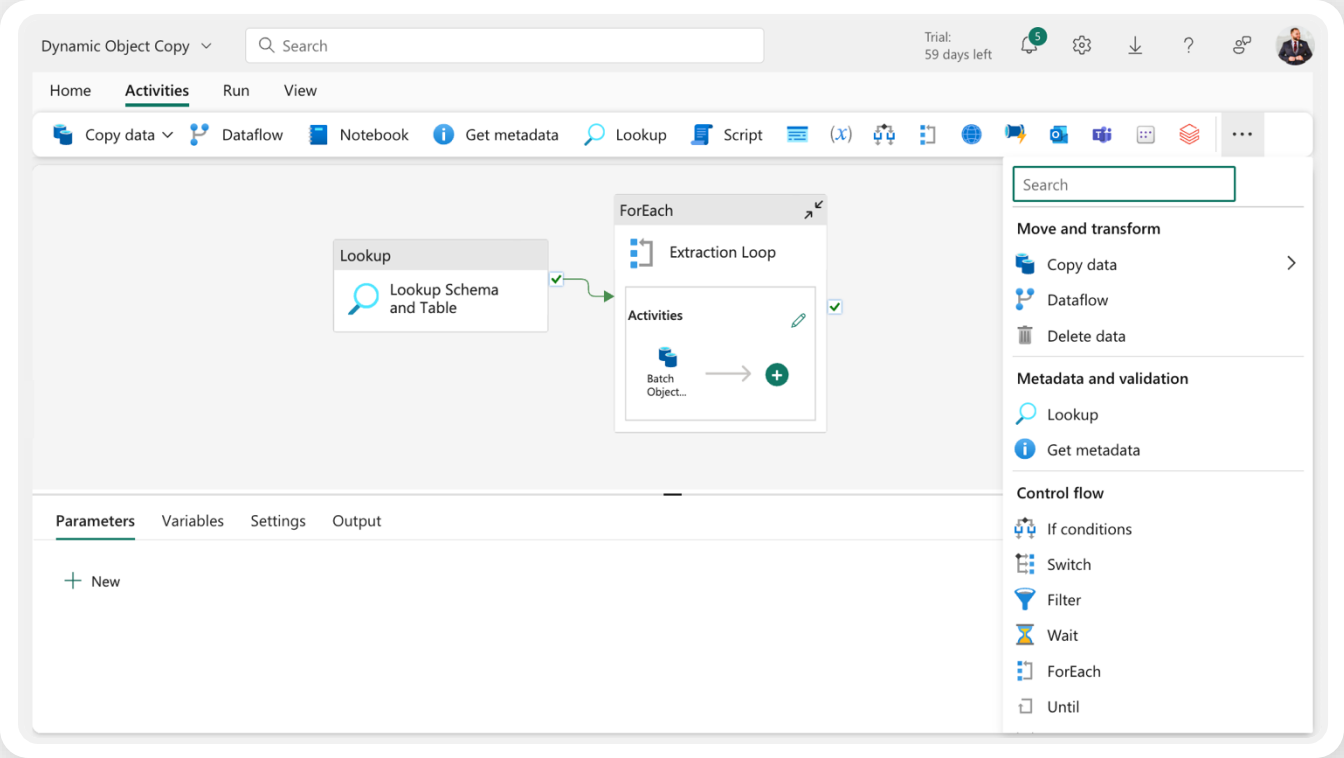
- Uses Connections instead of a Linked Service
- Currently has only a subset of source connectors
- Optimal for working with large datasets

PIPELINE ORGANIZATION AND STRUCTURE

Pipeline canvas serves as the primary element for building workflows of single or multiple activities

Orchestration and processing is completed using pipeline activities, which can be combined to define activity dependencies

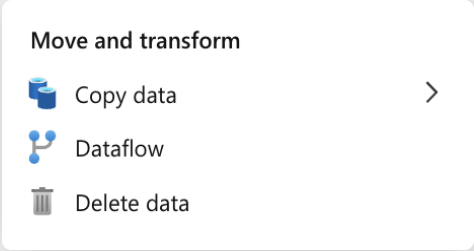
Parameters and variables can be used to create dynamic orchestration patterns



PRIMARY ACTIVITY TYPES

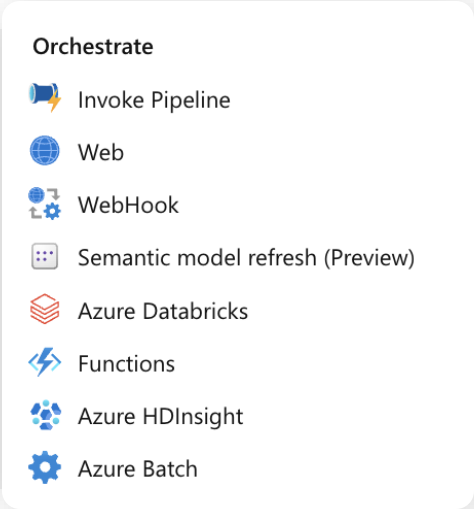
Move and Transform:

Pipelines are considered orchestration and movement items rather than transformation items



Orchestrate:

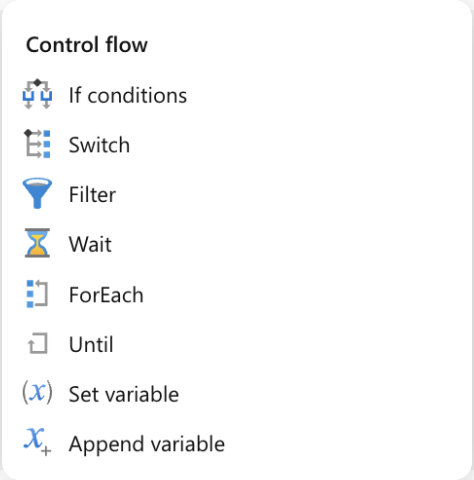
Often the primary choice for conducting multiple tasks in a particular order such as extracting data with a copy task, using a Databricks activity to transform, and refreshing the semantic model for consumption



PRIMARY ACTIVITY TYPES

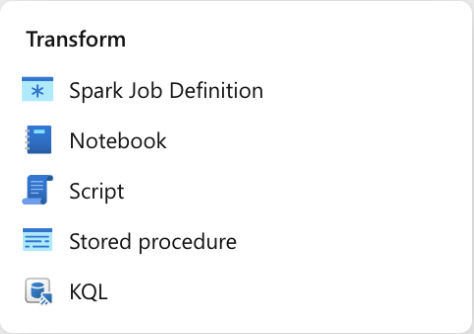
Control Flow:

Dynamic patterns leverage iterators like ForEach and conditionals like If and Switch



Transform:

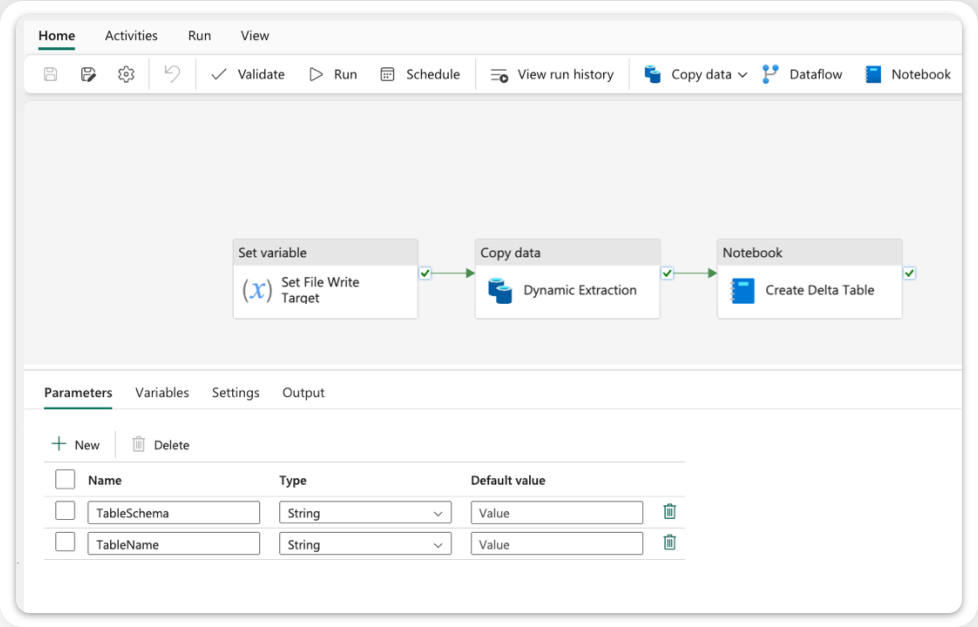
References to transforming data in the context of a pipeline often refer to orchestration engaging a notebook, stored procedure, or equivalent script – the pipeline does not perform the transformation natively

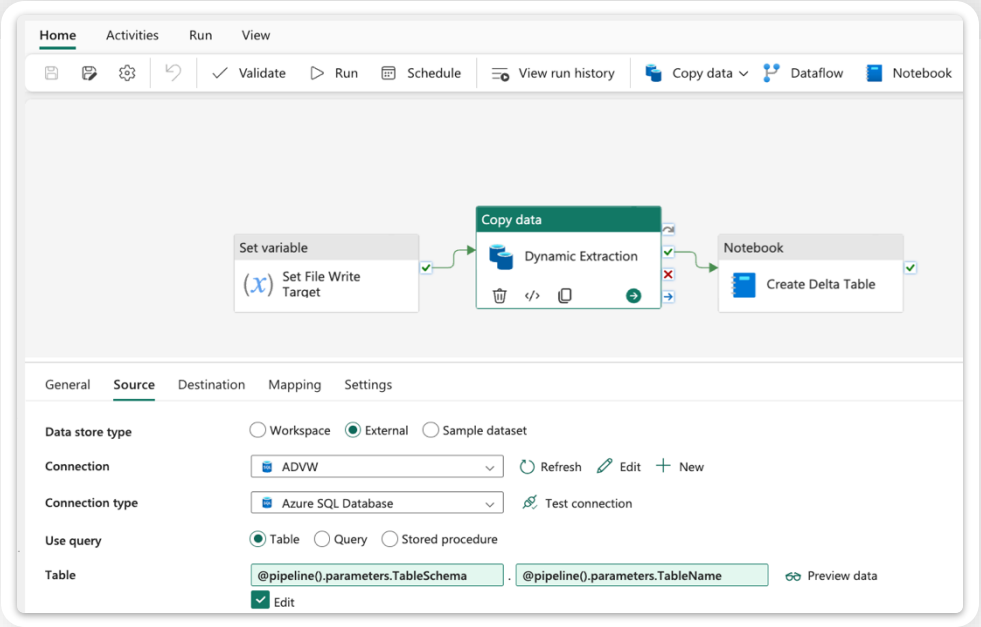


PIPELINE PARAMETERIZATION

Parameters allow developers to dynamically provide specific values for a pipeline run

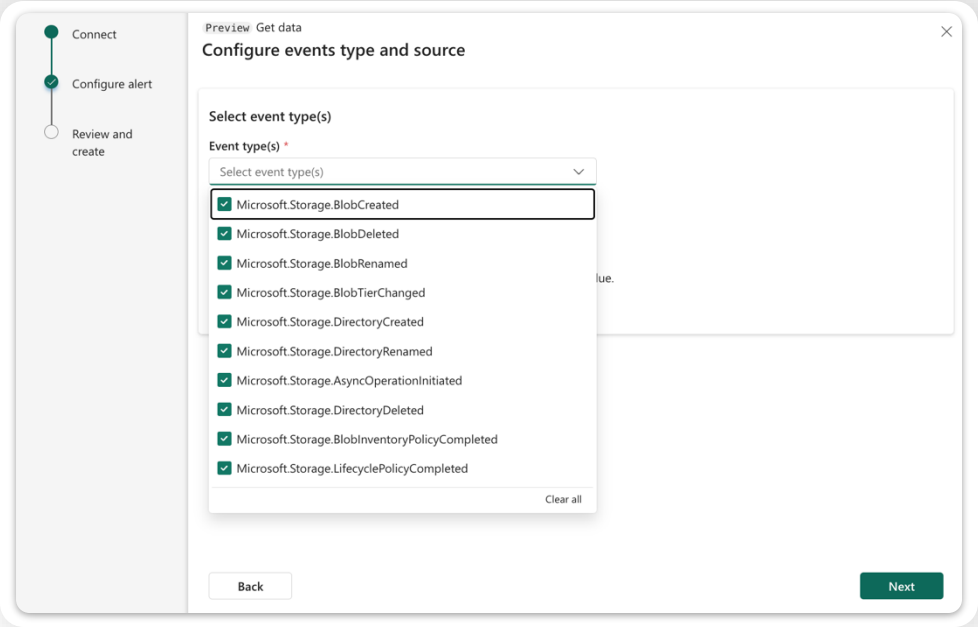
- Source and destination objects, folders, and files
- Can be combined with other activities, like ForEach loops, to optimize dynamic ingestion patterns
- Enhances pipeline reusability

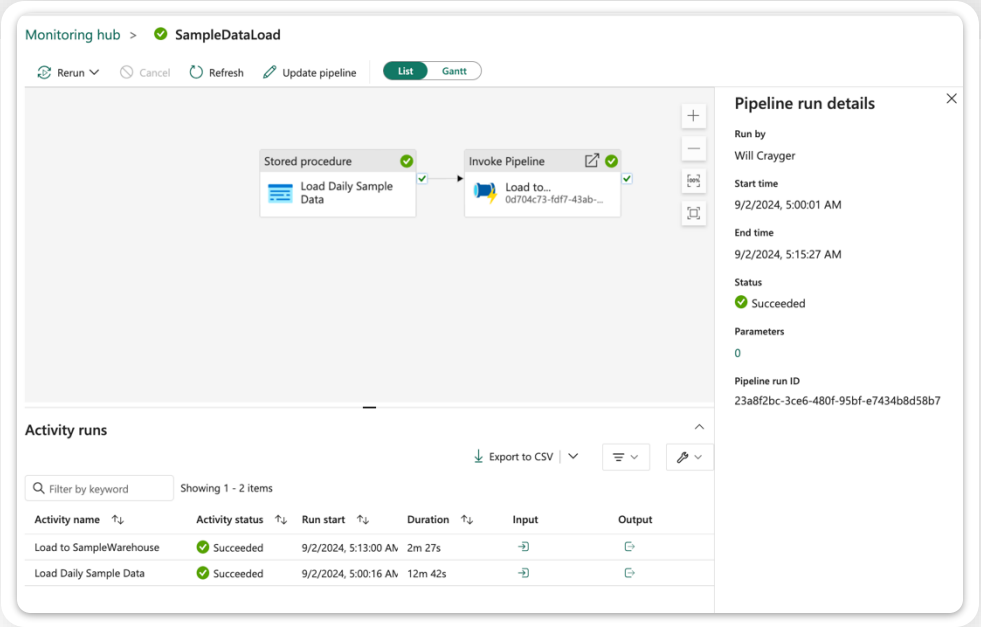




PIPELINE EXECUTION AND MONITORING

- Pipelines can be executed on-demand, scheduled, or triggered based on a storage blob event
- Pipeline run history can be viewed in the monitoring hub or the Run History pipeline tab
- Run id’s captures every pipeline execution and are useful for reviewing run details about an execution
- Execution history can be viewed as an activity list or a Gantt chart



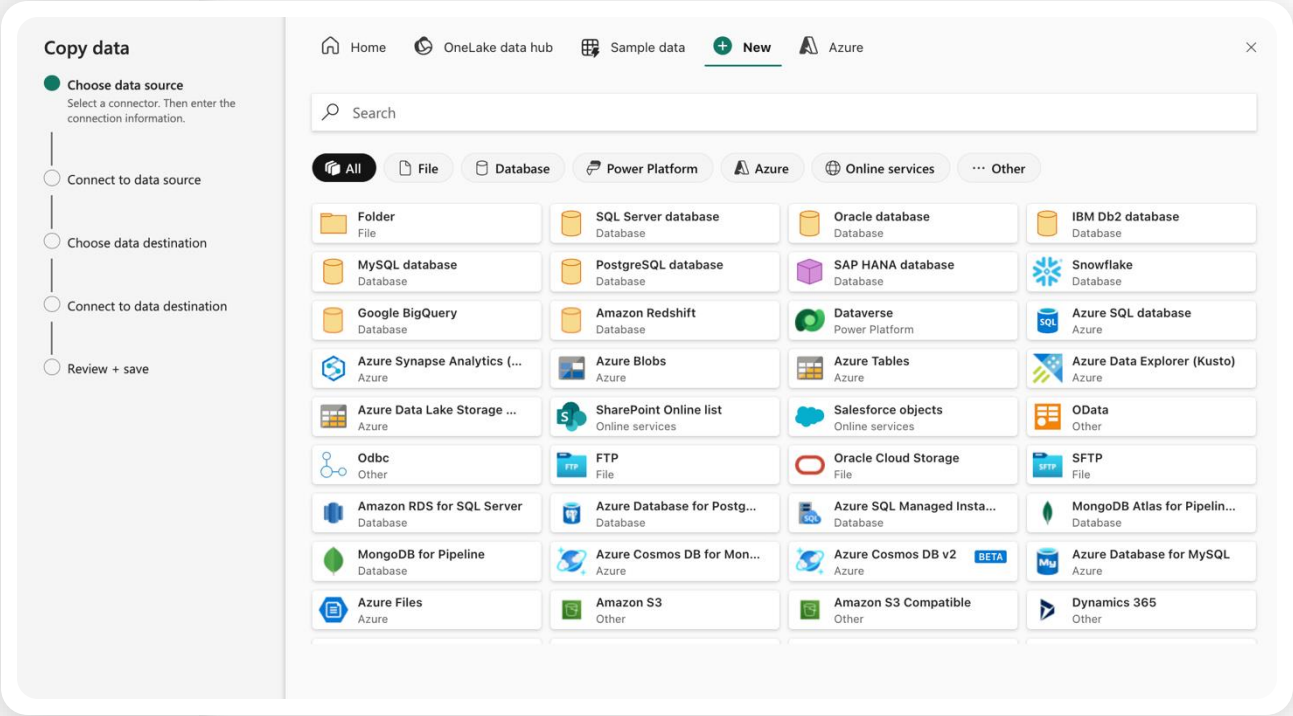


COPY DATA ASSISTANT

A graphical tool that guides users through configuring the data source and destination

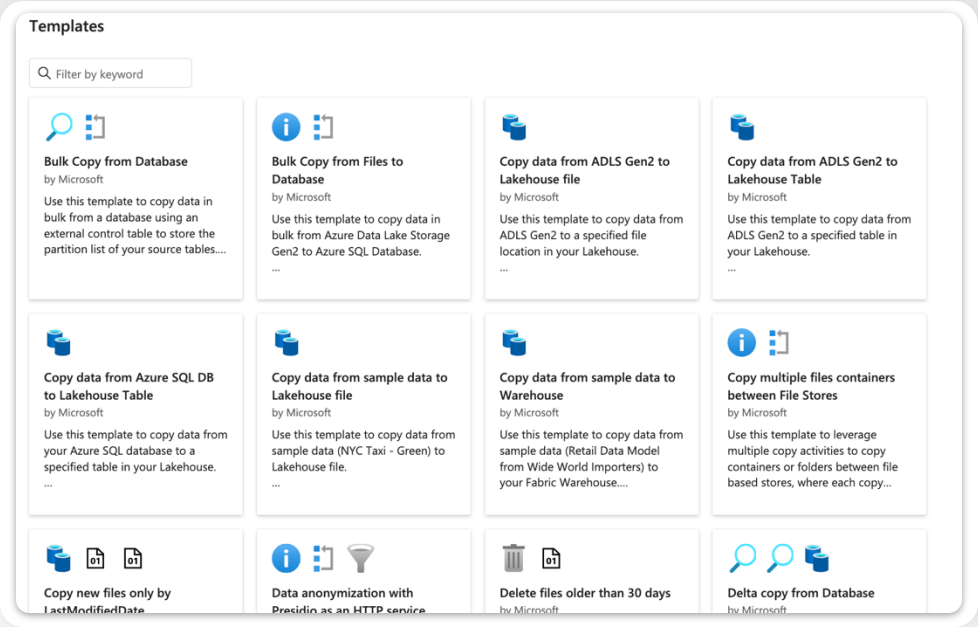
Can pick from existing source in **OneLake** Data Hub or select from a broad range of source connections

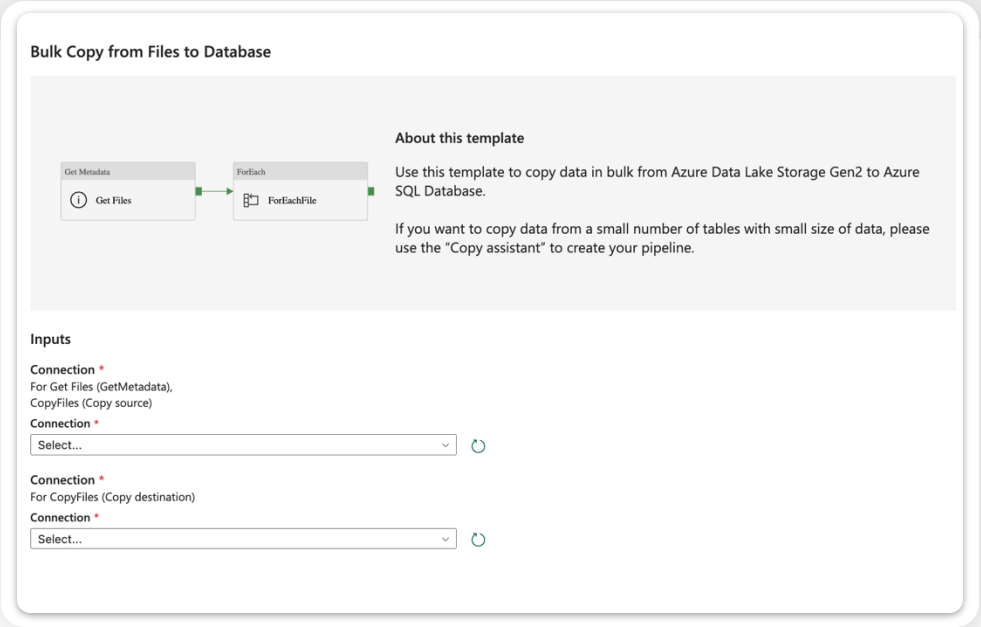
Additional connectors are released each month



USING PIPELINE TEMPLATES

- Fabric offers predefined pipeline templates for common scenarios
- Templates can be used as a starting point and customized as needed
- Navigating creation using a template constructs the pipeline canvas for you to simplify development



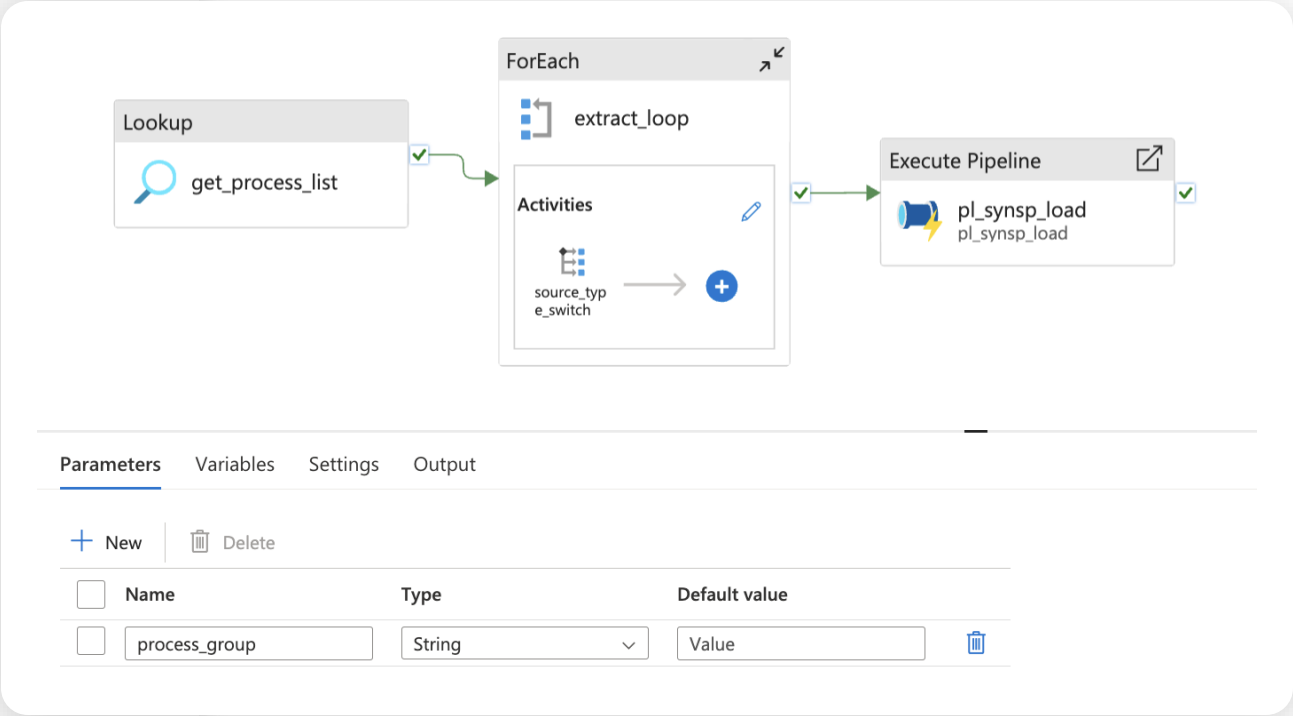


CUSTOM COPY PATTERNS

Copy task can be combined with other activities to create repeatable and flexible data ingestion patterns

Example:

- Lookup activity executes a query against configuration tables using a pipeline parameter to return a list of tables to process
- List of tables is passed to a ForEach loop containing a switch statement for conditional execution of copy activities
- Execute Pipeline activity passes the parameter to a pipeline containing another Lookup activity and a Notebook activity for transforming and loading data



GUIDED LEARNING LAB 3

Create a pipeline

Add a simple copy activity

Move data from source to OneLake

Scheduling

INTRO TO APACHE SPARK IN MICROSOFT FABRIC

INTRO TO SPARK

Apache Spark is an open-source framework for large-scale data processing.

It's popular for "big data" processing and is available in multiple platforms:

- Azure HDInsight
- Azure Synapse Analytics
- Azure Databricks
- Microsoft Fabric

Spark in Fabric is used for data ingestion, processing, and analysis with lakehouse at the core.

BASICS OF APACHE SPARK

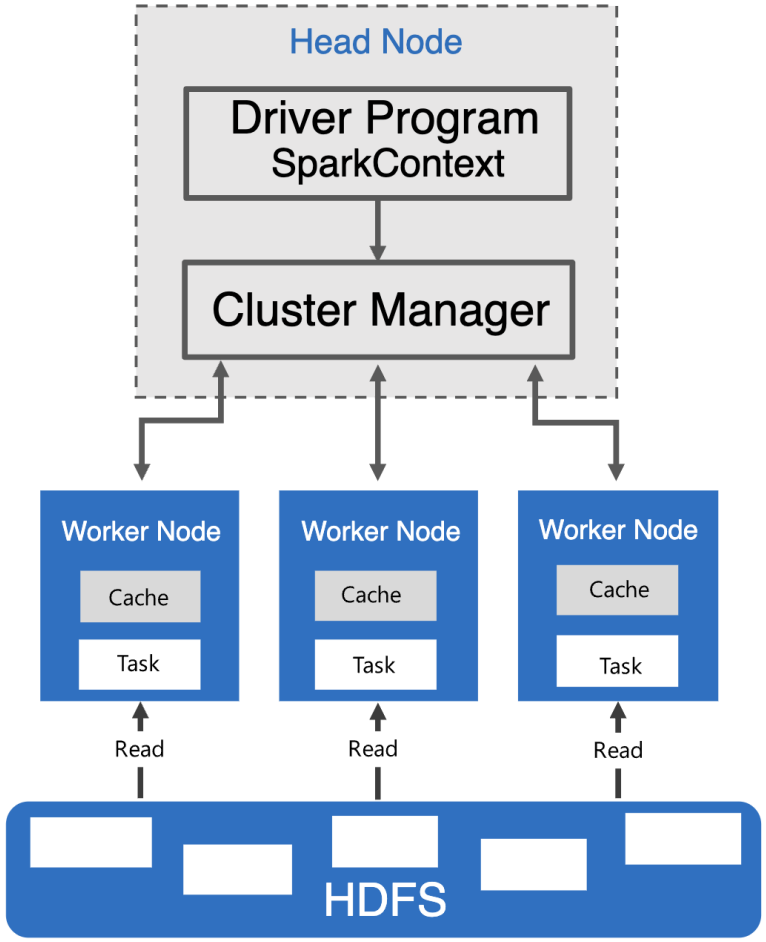
Distributed data processing framework that facilitates large-scale data analytics by coordinating tasks across multiple processing nodes

Uses a "divide and conquer" strategy by spreading the workload across multiple machines

Distribution of tasks and results aggregation is managed by Spark

Distribution process:

- User submits a data processing job using code
- This code initiates a driver program
- Driver program uses SparkContext to oversee the distribution of tasks within the Spark cluster



SPARK LAZY EVALUATION

Deferred Execution:

- Spark operations are not immediately executed and are instead are added to a DAG (Directed Acyclic Graph) of transformations
- The actual execution only happens when specific actions are triggered (e.g., collect(), count(), write())

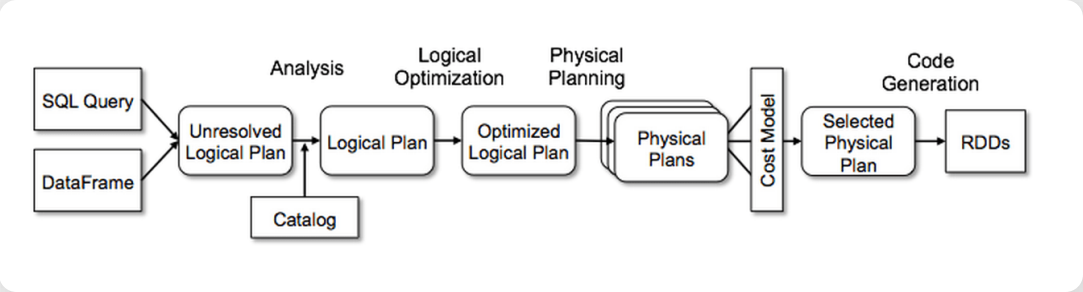
Efficiency Benefits:

- Allows Spark to optimize the execution plan based on the entire pipeline of transformations rather than executing each transformation step by step
- Minimizes redundant computations and reduces I/O operations by pipelining transformations.

SPARK SQL CATALYST OPTIMIZER

Analysis:

- Uses the Catalog to build a logical plan by looking up table and column name references
- Applies rules to map attributes to their input, assign unique IDs to attributes, and propagate or coerce data types



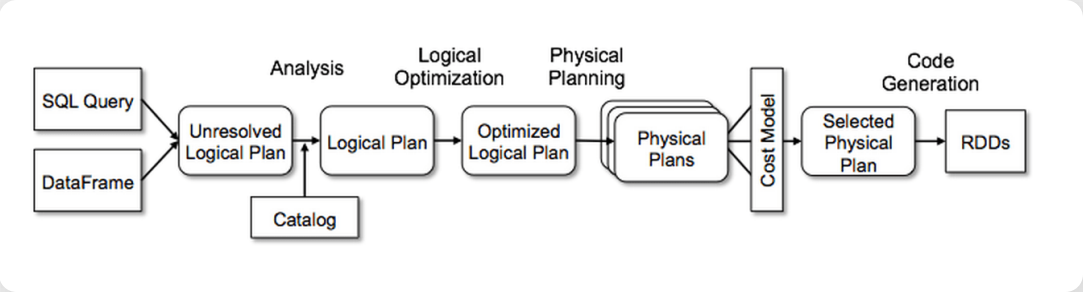
Logical Optimization:

- Applies rule-based optimizations such as constant folding, predicate pushdown, projection pruning, null propagation, and Boolean expression simplification
- Performs cost-based optimization by generating and evaluating multiple plans.

SPARK SQL CATALYST OPTIMIZER

Physical Optimization:

- Generates physical plans and selects the best using a cost-based model by choosing the optimal join algorithm
- Applies rule-based optimizations, like pipelining projections and filters into a single operation, and supports pushdown to data sources.



Code Generation:

- Uses quasiquotes in Scala to generate and compile Java bytecode for execution
- Transforms SQL expressions into abstract syntax trees (ASTs) for faster, CPU-bound in-memory processing

SPARK NOTEBOOKS

Ideal for big data workloads and analysis using Spark

Allows for a mix of text, images, and code in multiple languages

Consists of markdown cells for documenting and executable code cells

Code can be executed interactively with immediate results

Build Stage DataFrame

This code builds a staging DataFrame named `df_stage` using a SQL query. The query performs the following operations:

- **Selects and renames columns**:
  - `OrderLineID` is renamed to `order_line_id`.
  - `Quantity` is renamed to `order_quantity`.
  - `UnitPrice` is renamed to `unit_price`.
  - `TaxRate` is renamed to `tax_rate`.
  - The keys for salesperson, package type, stock item, and customer are selected from their respective tables with any missing values replaced by `-1`.
- **Joins tables**:
  - The `orderlines` table is joined with the `orders` table on `orderid`.
  - A left join is performed with dimension tables (`dim_salesperson`, `dim_package_type`, `dim_stock_item`, `dim_customer`) to include additional information, such as salesperson, package type, stock item, and customer keys.

The resulting DataFrame `df_stage` combines data from various sources in the `bronze` and `gold` layers of the data lake.

```python
# Build stage dataframe# Build stage
df_stage = spark.sql("""
    SELECT
        OrderLineID     order_line_id
        ,COALESCE(s.salesperson_key, -1)    salesperson_key
        ,COALESCE(p.package_type_key, -1)   package_type_key
        ,COALESCE(st.stock_item_key, -1)    stock_item_key
        ,COALESCE(c.customer_key, -1)       customer_key
        ,Quantity       order_quantity
        ,UnitPrice      unit_price
        ,TaxRate        tax_rate
    FROM bronze.orderlines ol
    LEFT JOIN bronze.orders o ON o.orderid = ol.orderid
    LEFT JOIN gold.dim_salesperson s ON o.salespersonpersonid = s.salesperson_id
    LEFT JOIN gold.dim_package_type p ON ol.packagetypeid = p.package_type_id
    LEFT JOIN gold.dim_stock_item st ON ol.stockitemid = st.stock_item_id
    LEFT JOIN gold.dim_customer c ON o.customerid = c.customer_id
""")
```

✓ - Command executed in 3 sec 495 ms by Will Crayger on 7:44:31 PM, 6/09/24 PySpark (Python)

SPARK DATA STRUCTURES

Resilient Distributed Dataset (RDD):

- Fault-tolerant and parallelizable collection of elements providing low-level operations and control over data processing
- Generally used with unstructured data

DataFrame:

- Provides a schema-based view of data, similar to tables in relational databases
- Leverage Sparks Catalyst optimizer which automatically optimizes queries based on structure and operations performed

```
1   display(df_stage)
```

16 sec - Command executed in 16 sec 645 ms by Will Crayger on 9:40:12 PM, 9/03/24 PySpark (Python)

Log

Table Chart Download Showing rows 1 - 1000 Inspect Search

	order_line_id	salesperson_key	package_type_key	stock_item_key	customer_key	order_quantity	unit_price	tax_rate
1	922	5	10	57	-1	84	5.00	15.000
2	3157	8	10	57	-1	84	5.00	15.000
3	5647	4	10	57	-1	84	5.00	15.000
4	5979	2	10	57	-1	84	5.00	15.000
5	6428	1	10	57	-1	84	5.00	15.000
6	8516	4	10	57	-1	84	5.00	15.000
7	12870	8	10	57	-1	84	5.00	15.000
8	22195	7	10	57	-1	84	5.00	15.000
9	23972	2	10	57	-1	84	5.00	15.000
10	27887	10	10	57	-1	84	5.00	15.000
11	31337	1	10	57	-1	84	5.00	15.000
12	33564	8	10	57	-1	84	5.00	15.000
13	33794	5	10	57	-1	84	5.00	15.000
14	34211	7	10	57	-1	84	5.00	15.000
15	37537	9	10	57	-1	84	5.00	15.000
16	38715	6	10	57	-1	84	5.00	15.000
17	38990	1	10	57	-1	84	5.00	15.000

SCHEMA DEFINITION

Inferred:

- Automatic Detection: Spark automatically determines data types based on the data
- Simplified by Headers: Easier when the first row contains column names

```python
# Set file path
file_path = "abfss://6c54a0c2-fdfe-4265-88f1-b75bf57286b3@onelake.dfs.fabric.microsoft.com

# Load data from a file and apply the schema
df_package_type_inferred = spark.read.format("csv") \
    .option("header", "true") \
    .load(file_path)

display(df_package_type_inferred)
```

PySpark (Python)

Explicit:

- User-Defined: Required when column names or types aren't clear in the data
- Performance Boost: Enhances performance by reducing the need for Spark to infer data types

```python
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, TimestampType

# Set file path
file_path = "abfss://6c54a0c2-fdfe-4265-88f1-b75bf57286b3@onelake.dfs.fabric.microsoft.com/848ae3

# Define the schema
schema = StructType([
    StructField("PackageTypeID", IntegerType(), True),
    StructField("PackageTypeName", StringType(), True),
    StructField("LastEditedBy", IntegerType(), True),
    StructField("ValidFrom", TimestampType(), True),
    StructField("ValidTo", TimestampType(), True)
])

# Load data from a file and apply the schema
df_package_type_explicit = spark.read.format("csv") \
    .option("header", "true") \
    .schema(schema) \
    .load(file_path)

display(df_package_type_explicit)
```

PySpark (Python)

SPARK CATALOG OBJECTS - TEMPORARY VIEW

Used to make DataFrame data available for querying with SQL

```python
# Create DataFrame with SQL query
df_stage = spark.sql("SELECT ... FROM bronze.orderlines ol ... LEFT JOIN ...")

# Create temporary view
df_stage.createOrReplaceTempView("temp_view_stage")
```

PySpark (Python)

Created within a session and deleted when the session ends

Data is not persisted; no additional storage used

```sql
%%sql

-- Create temporary view with Spark SQL
CREATE OR REPLACE TEMP VIEW temp_view_stage AS
SELECT ... FROM bronze.orderlines ol ... LEFT JOIN ...
```

Spark SQL

Suitable for quick, ad-hoc analysis and temporary processing

SPARK CATALOG OBJECTS - TABLE

Persistent storage of data in a defined location

Metadata stored in the catalog, describing data structure

Can be queried using Spark SQL across sessions

Delta tables support:

- ACID transactions for data integrity
- Data versioning and time travel for historical queries
- Streaming support for real-time processing

Table | Chart | Download | Showing rows 1 - 7

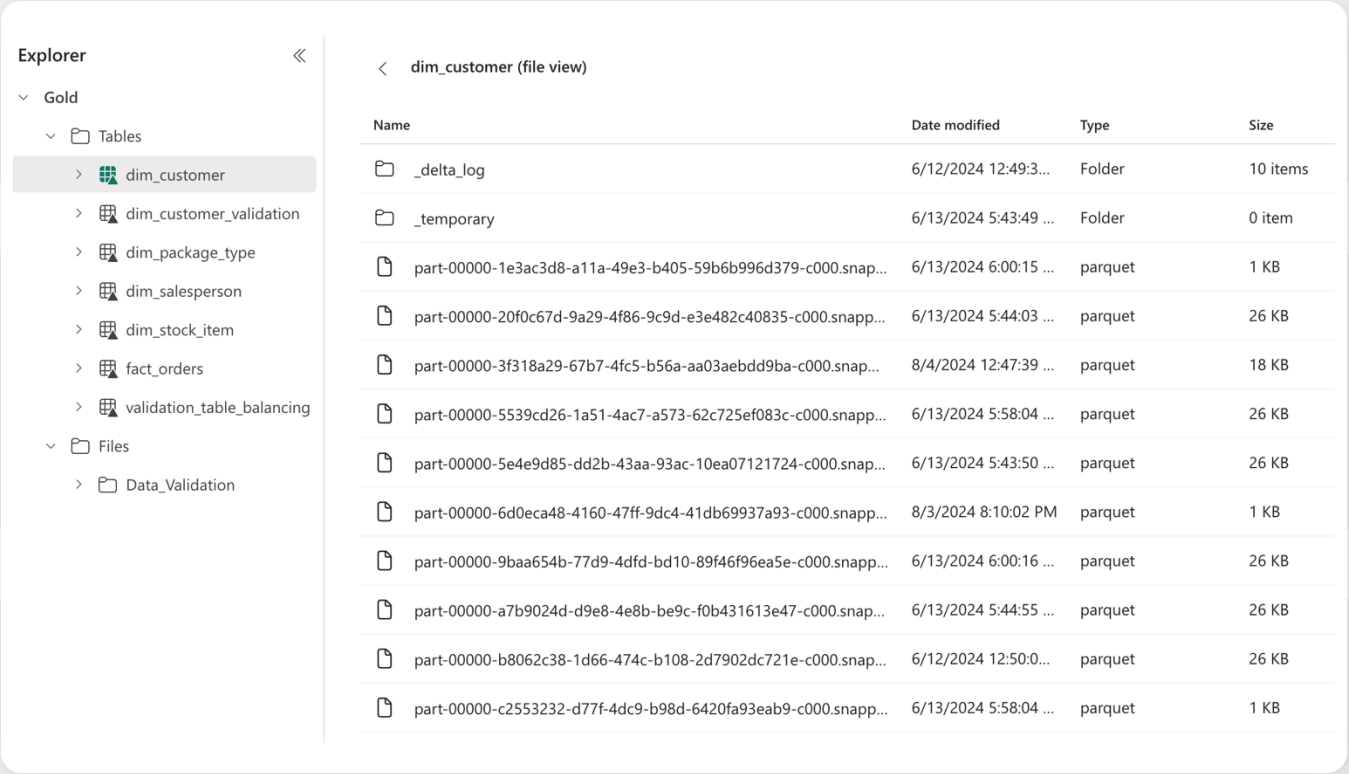
	format	name	numFiles	sizeInBytes	minReaderVersion	minwriterVersion
1	delta	spark_catalog.gold.dim_customer	1	18741	1	2
2	delta	spark_catalog.gold.validation_table_balancing	23	88223	1	2
3	delta	spark_catalog.gold.dim_customer_validation	1	5977	1	2
4	delta	spark_catalog.gold.dim_stock_item	2	22706	1	2
5	delta	spark_catalog.gold.dim_salesperson	1	5429	1	2
6	delta	spark_catalog.gold.dim_package_type	2	5927	1	2
7	delta	spark_catalog.gold.fact_orders	1	5526846	1	2

DELTA LAKE TABLES IN FABRIC

Denoted by a triangular Delta (▲) icon

Tables are schema abstractions over data files stored in the Delta format

Each table consists of a series of parquet data files and a _delta_Log folder logging transaction details



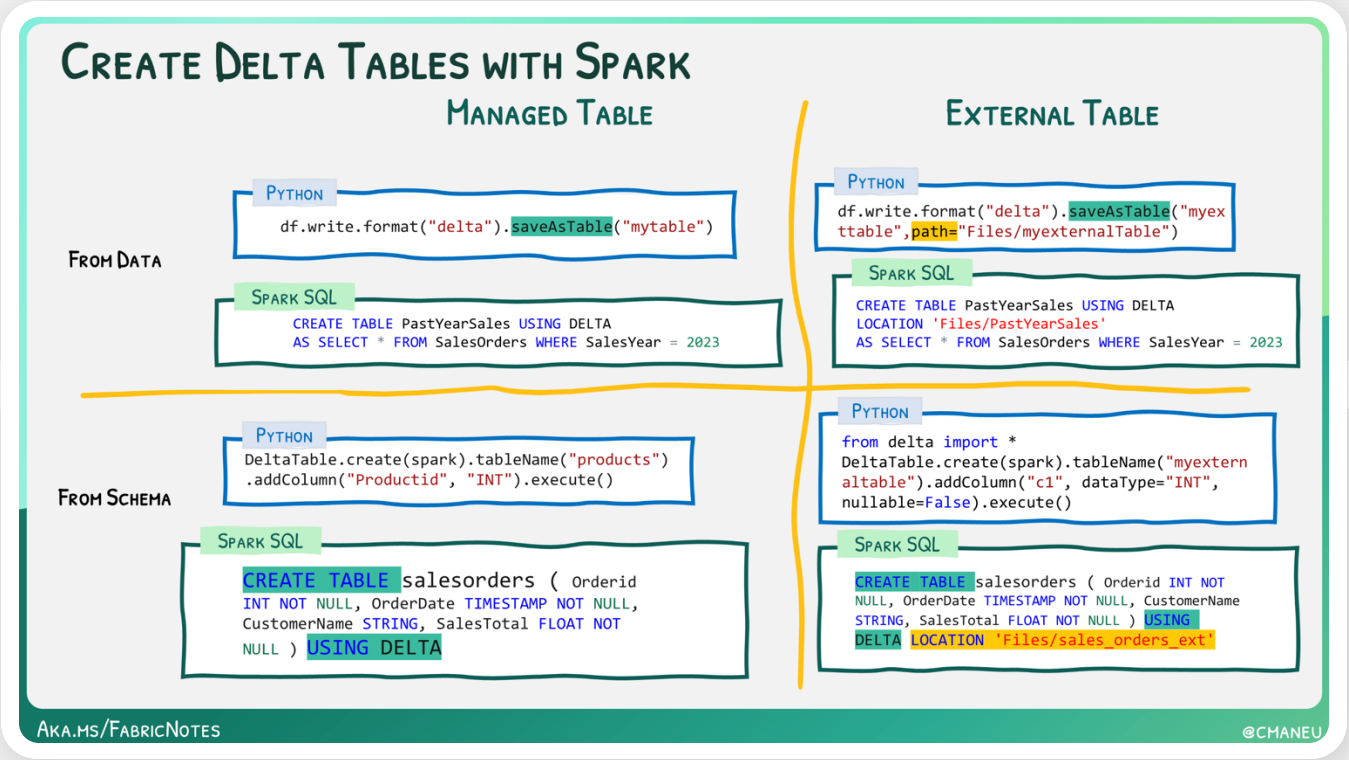
MANAGED OR EXTERNAL TABLES

Managed Tables:

- Both the table definition in the metastore and the underlying data files are managed by the Spark runtime
- Underlying files from the Tables storage location will be deleted with the table

External Tables:

- Relational table definition in the metastore is mapped to an alternate file storage location
- Deleting an external table from the lakehouse metastore does not delete the associated data files



GUIDED LEARNING LAB 4

Import Notebook using the Data Engineering Experience

Build “silver” Lakehouse

Using views to transform data

Work through transformations notebook

WORKING WITH FABRIC WAREHOUSE

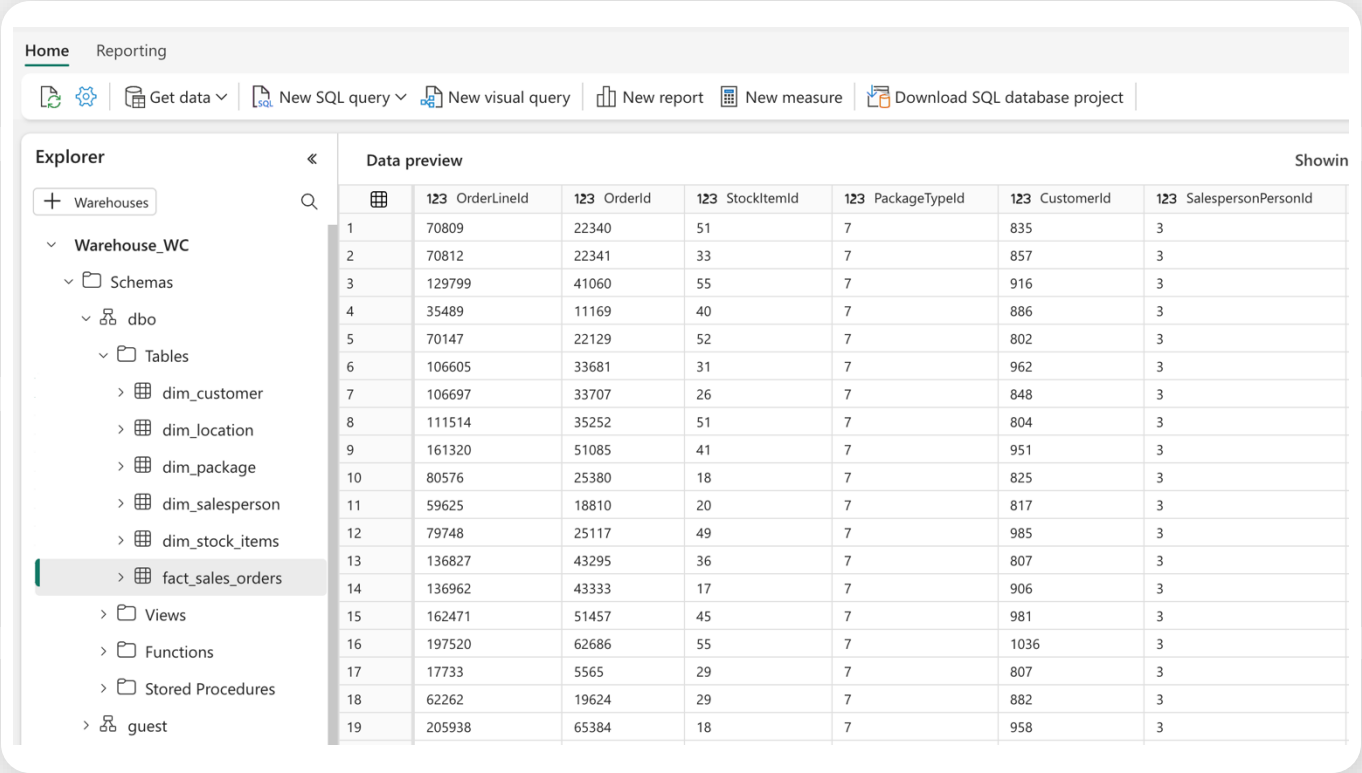
WORKING WITH FABRIC WAREHOUSE

Fabric Warehouse, like the Lakehouse, uses a lake-centric architecture leveraging delta storage format

Unlike the Lakehouse, Warehouse includes full SQL semantics allowing for insert, update, and delete operations

Accessible from common tools like SSMS and Azure Data Studio

Ideal for working with structured data workloads



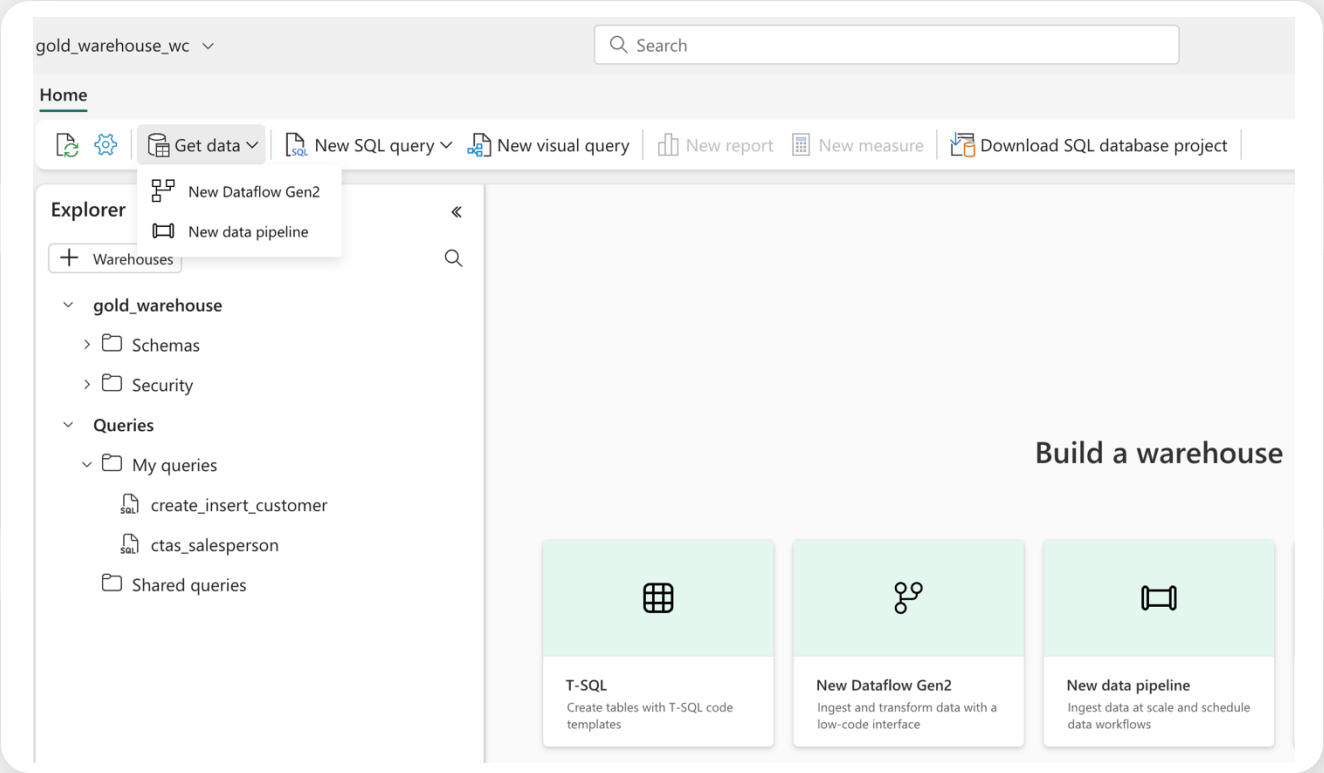
	OrderLineId	OrderId	StockItemId	PackageTypeId	CustomerId	SalespersonPersonId
1	70809	22340	51	7	835	3
2	70812	22341	33	7	857	3
3	129799	41060	55	7	916	3
4	35489	11169	40	7	886	3
5	70147	22129	52	7	802	3
6	106605	33681	31	7	962	3
7	106697	33707	26	7	848	3
8	111514	35252	51	7	804	3
9	161320	51085	41	7	951	3
10	80576	25380	18	7	825	3
11	59625	18810	20	7	817	3
12	79748	25117	49	7	985	3
13	136827	43295	36	7	807	3
14	136962	43333	17	7	906	3
15	162471	51457	45	7	981	3
16	197520	62686	55	7	1036	3
17	17733	5565	29	7	807	3
18	62262	19624	29	7	882	3
19	205938	65384	18	7	958	3

WAREHOUSE FEATURES AND INTEGRATION

Both data pipeline and dataflow gen2 can write directly to Warehouse tables

Warehouse database project support for managing source code

Multi-table transactional support

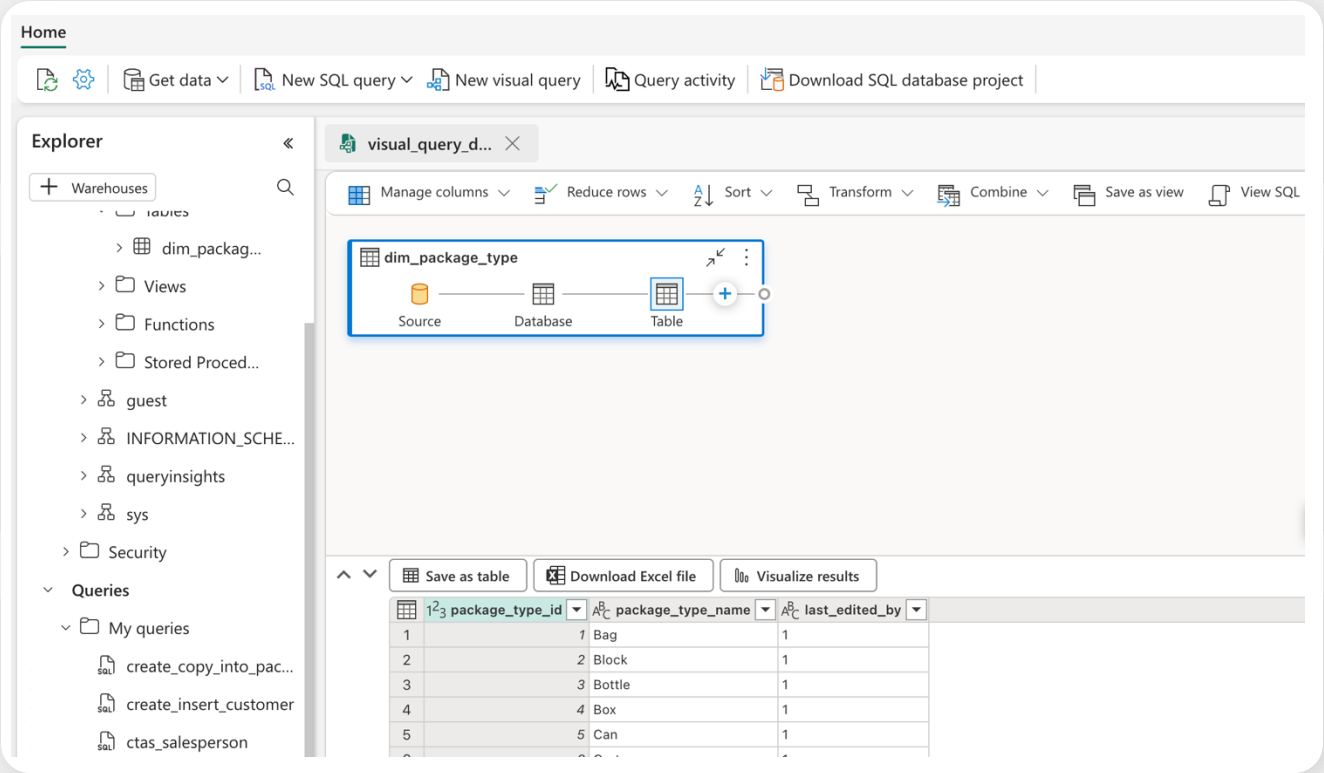


WAREHOUSE FEATURES AND INTEGRATION

Allows for the creation of tables, views, and stored procedures

Provides features like intellisense, code completion, and syntax highlighting

Supports query creation using the new visual query editor



SETTING UP AND NAVIGATING THE WAREHOUSE

Each Warehouse in a Fabric-enabled workspace consists of two main elements:

Warehouse: Storage and metadata section used to engage tables

Semantic Model (default): Auto-generated data model derived from the tables in the Lakehouse

TABLE CREATION AND LOADING

CREATE TABLE and INSERT as independent operations

```sql
-- Create table
CREATE TABLE [dim_package_type](
    [package_type_id] [int] NOT NULL,
    [package_type_name] [varchar](100) NOT NULL,
    [last_edited_by] [varchar](20) NOT NULL
);

-- Load table using COPY INTO
COPY INTO [dbo].[dim_package_type]
FROM 'https://adlsfabconeu.blob.core.windows.net/fabcon-eu/package_types.csv'
WITH (
    FILE_TYPE = 'CSV',
    FIRSTROW = 2
);

-- Load table using INSERT INTO
INSERT INTO [dim_customer]
SELECT customer_id
      ,customer_name
      ,phone_number
FROM [silver_lakehouse_wc].[dbo].[sales_customers]
```

CREATE TABLE and COPY INTO as independent operations

CREATE TABLE AS SELECT to combine the table creation and loading to a single operation

```sql
-- Create and load table using CREATE TABLE AS SELECT (CTAS)
CREATE TABLE dim_salesperson AS
SELECT person_id
      ,person_name
      ,preferred_name
      ,logon_name
FROM [silver_lakehouse_wc].[dbo].[salesperson]
```

GUIDED LEARNING LAB 5

Create Fabric Warehouse

Review Warehouse Loading Patterns

Build Dimensional Model

ANALYZING DATA IN FABRIC WITH POWER BI

UNDERSTANDING SEMANTIC MODELS

A default semantic model is automatically created whenever a Data Warehouse or Lakehouse is provisioned

Default semantic models are automatically synchronized with the Warehouse or Lakehouse

Users can create custom datasets tailored to their requirements

Name	Type
bronze_lakehouse_wc	Lakehouse
bronze_lakehouse_wc	Semantic model (default)
bronze_lakehouse_wc	SQL analytics endpoint
gold_warehouse_wc	Warehouse
gold_warehouse_wc	Semantic model (default)
sales_model_wc	Semantic model

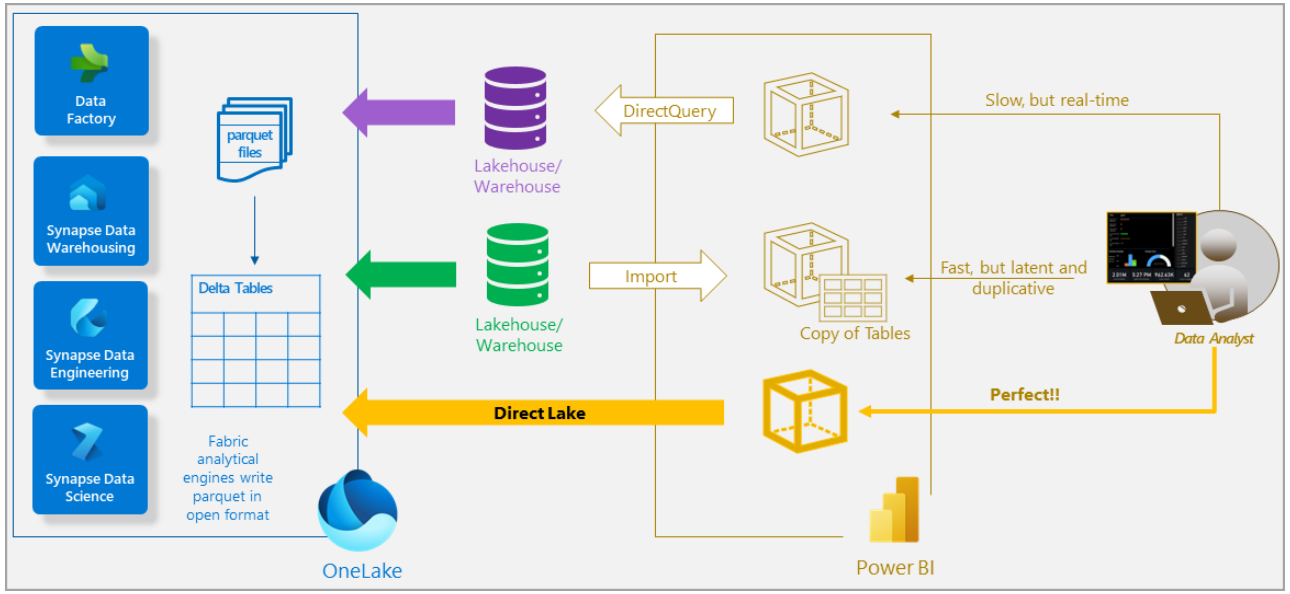
SEMANTIC MODEL CONNECTIVITY

There are three primary connectivity options when creating a new semantic model:

Import: Data is cached in the semantic model meaning there's no need to translate queries from DAX/MDX to the source

Direct Query: Data is queried from the source at report consumption. Performance is often slow, but data is considered "live"

Direct Lake: Data is read directly from OneLake delta tables without requiring an import or the need to query from the source directly



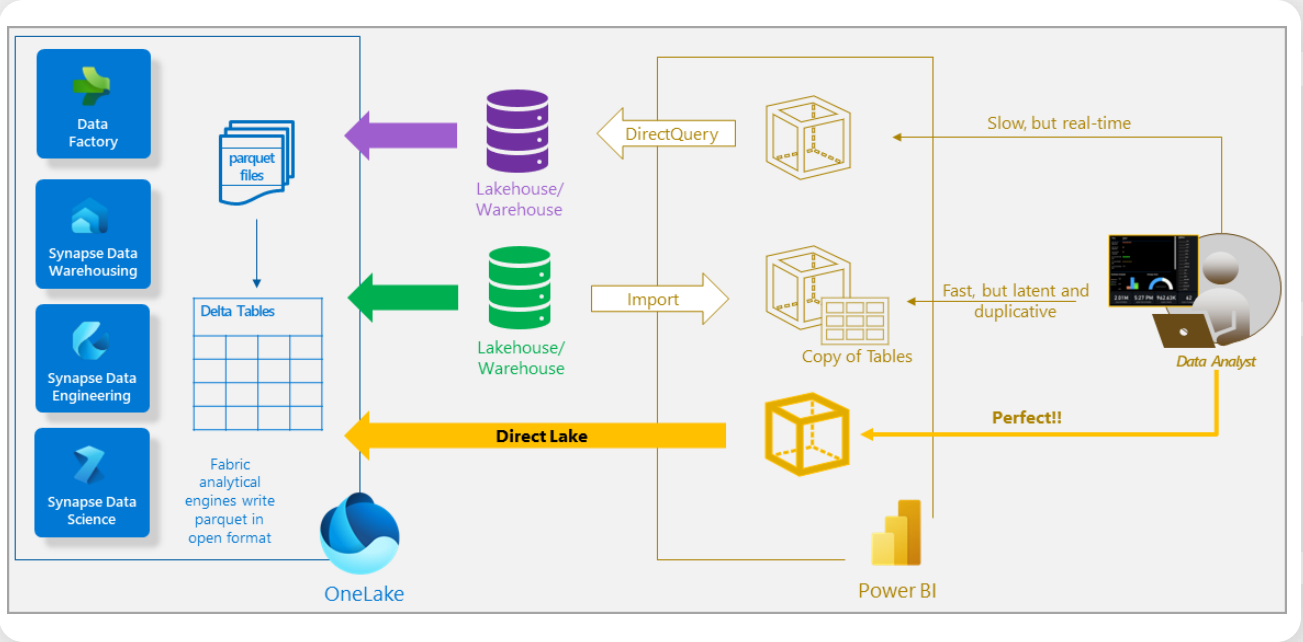
DIRECT LAKE CONSIDERATIONS

Semantic Models must be created directly from the Fabric UI, desktop development is not supported

Direct Lake connections may “Fallback” to Direct Query connections

Only semantic model tables originating from a Fabric Lakehouse or Warehouse are supported

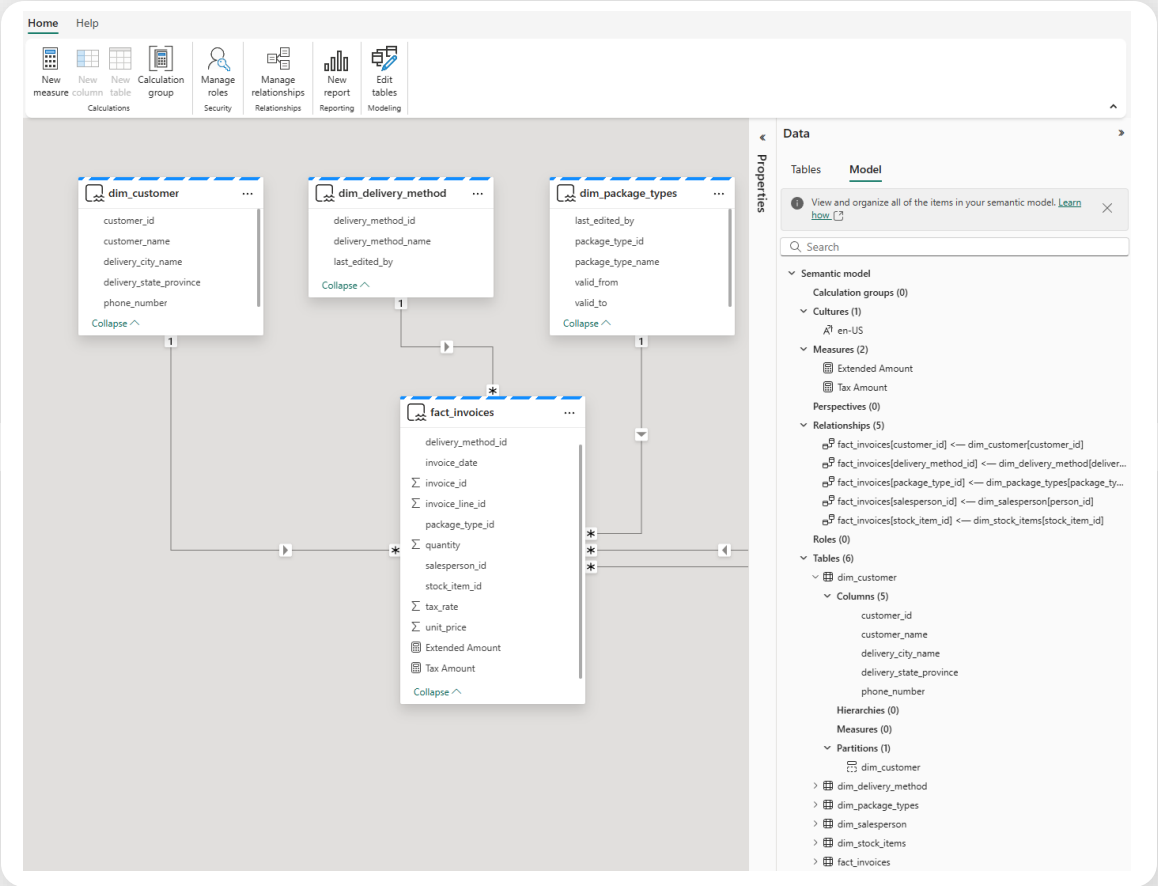
Calculated tables and columns aren’t supported



PREPARING DATA FOR ANALYSIS AND REPORTING

Semantic models allow you to define:

- Relationships between tables
- Rules for data aggregation and summarization
- Calculations or measures



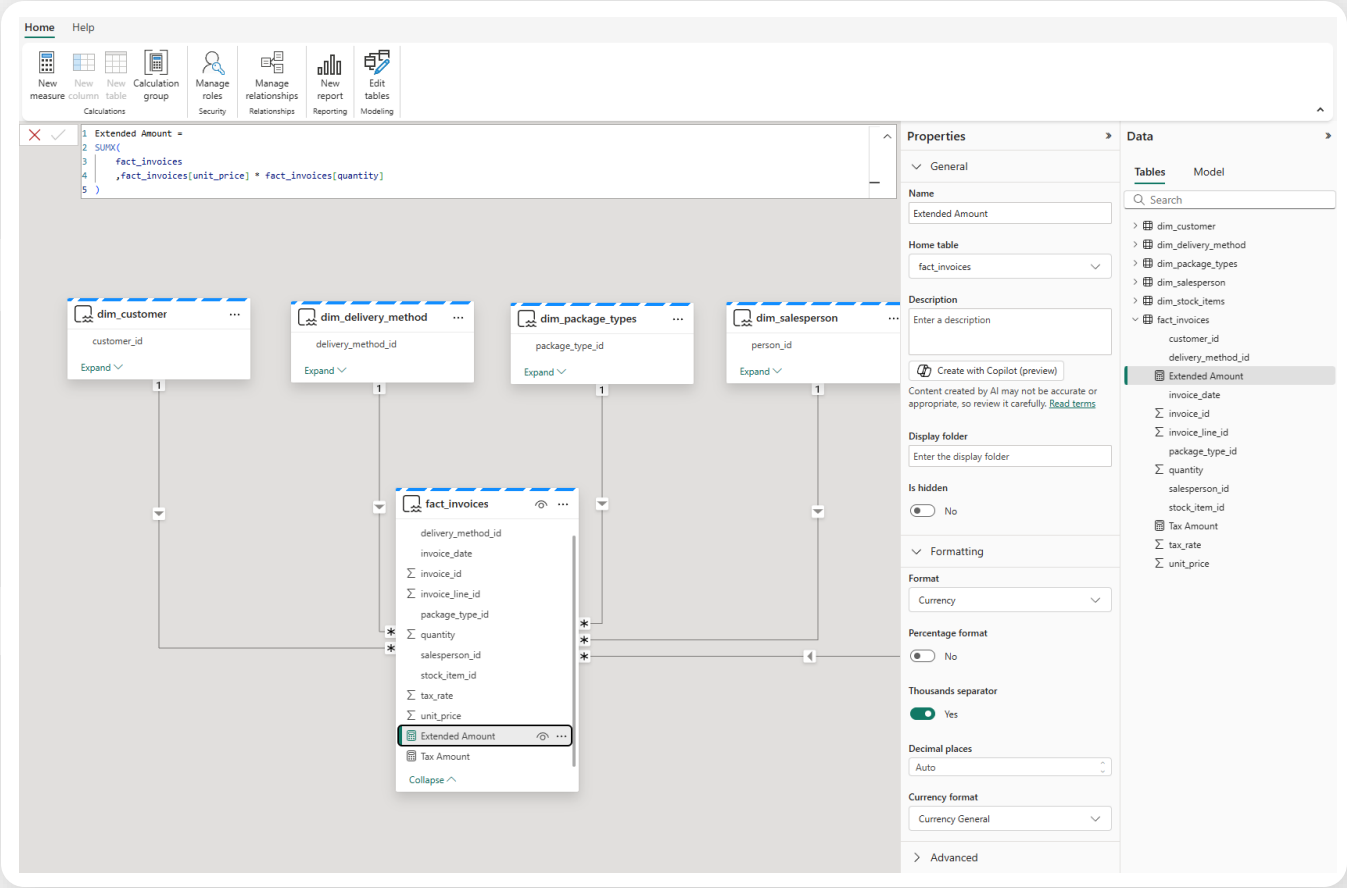
Serve as the foundation for Power BI report generation

DATA AGGREGATION USING MEASURES

Calculated measures represent the metrics to be analyzed

Measures are defined using the Data Analysis Expressions (DAX) formula language

Can be created from the Fabric UI or using other tools connected to the XMLA endpoint such as Tabular Editor



GUIDED LEARNING LAB 6

Create custom semantic model

Add a measure for explicit aggregation

Create a custom Power BI report using DirectLake

Create an auto-generated report

Microsoft

Get Involved in the Fabric Community

aka.ms/FabricCommunity
Connect with community members, ask questions, and learn more about Fabric

aka.ms/FabricUserGroups
Find a user group that matches your interests in your area or online

aka.ms/SuperUsers
Spread your Fabric knowledge, insights, and best practices with others

aka.ms/MVP
Technology experts that share their knowledge and passion with the community

FREE Microsoft Fabric

Certification Exam

Are you ready to get Fabric certified by the end of October?

Claim your **100% discount voucher** for Exam DP-600: Fabric Analytics Engineer.

Come find us in the Community Lounge under the Get Certified banner to learn more!

aka.ms/FabCon24/IAmReady

Please rate this session on the app

cvent

GET IT ON Google Play

Download on the App Store