

Domain 1: Get started with Microsoft Fabric

Introduction to end-to-end analytics using Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/introduction-end-analytics-use-microsoft-fabric/1-introduction>

Introduction

Completed

Organizations need to ingest, prepare, govern, and analyze data at scale, often across disconnected tools and teams. Increasingly, that same data also needs to be ready for AI workloads like machine learning models, Copilots, and intelligent agents. Managing these tasks across separate systems creates complexity, governance gaps, and duplicated effort.

Microsoft Fabric is an end-to-end analytics platform that provides a single, integrated environment where data professionals and the business collaborate on data projects. Built on a unified data lake called OneLake, Fabric brings together the tools you need across that entire lifecycle.

Because all data is ingested, prepared, and governed within Fabric, the same data that powers your reports and dashboards is also available to AI capabilities like Copilot, data agents, and Fabric IQ. This means the work you do to organize and govern your data directly supports your organization's AI initiatives.

This module introduces the Fabric platform, discusses who Fabric is for, explores Fabric workloads, and examines how Fabric supports both analytics and AI.

2. Explore end-to-end analytics with Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/introduction-end-analytics-use-microsoft-fabric/2-explore-analytics-fabric>

Explore end-to-end analytics with Microsoft Fabric

Completed

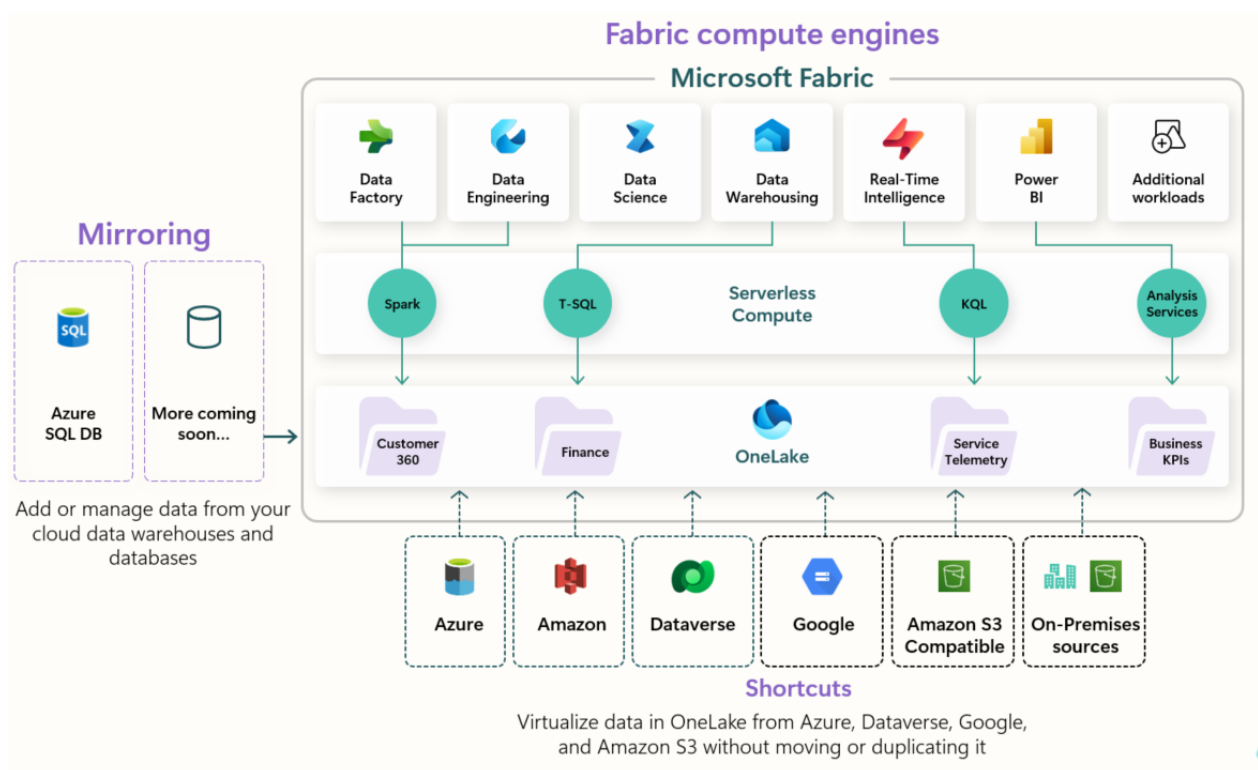
Scalable analytics can be complex, fragmented, and expensive. Microsoft Fabric simplifies analytics solutions by providing a single, easy-to-use product that integrates various tools and services into one platform.

Fabric is a unified *software-as-a-service* (SaaS) platform where all data is stored in a single open format in OneLake. All analytics engines in the platform can access OneLake, ensuring scalability, cost-effectiveness, and accessibility from anywhere with an internet connection.

OneLake

OneLake is Fabric's centralized data storage architecture that enables collaboration by eliminating the need to move or copy data between systems. OneLake unifies your data across regions and clouds into a single logical lake without moving or duplicating data.

OneLake is built on **Azure Data Lake Storage Gen2** (ADLS Gen2) and supports various formats, including Delta, Parquet, CSV, and JSON. All compute engines in Fabric automatically store their data in OneLake, making it directly accessible without the need for movement or duplication. For tabular data, the analytical engines in Fabric write data in delta-parquet format and all engines interact with the format seamlessly.



Shortcuts are references to files or storage locations within OneLake or external data sources, such as Azure Data Lake Storage, Amazon S3, or Database. Shortcuts allow you to access existing data without copying it, ensuring data consistency and enabling Fabric to stay in sync with the source.

Because all Fabric workloads store data in OneLake using an open format, AI capabilities like Copilot and data agents can access the same governed data as your reports and dashboards without separate data preparation pipelines. The work you do to ingest, prepare, and govern data in Fabric is what makes that data available for AI workloads.

Workspaces

In Microsoft Fabric, workspaces serve as logical containers that help you organize and manage your data, reports, and other assets. They provide a clear separation of resources, making it easier to control access and maintain security.

Each workspace has its own set of permissions, ensuring that only authorized users can view or modify its contents. This structure supports team collaboration while maintaining strict access control for both business and IT users.

Workspaces allow you to manage compute resources and integrate with Git for version control. You can optimize performance and cost by configuring compute settings, while Git integration helps track changes, collaborate on code, and maintain a history of your work.

Administration and governance

Fabric's OneLake is centrally governed and open for collaboration. Data is secured and governed in one place, which allows users to easily find and access the data they need. Fabric administration is centralized in the **Admin portal**.

In the admin portal you can manage groups and permissions, configure data sources and gateways, and monitor usage and performance. You can also access the Fabric admin APIs and SDKs in the admin portal, which can automate common tasks and integrate Fabric with other systems.

The **OneLake catalog** helps you analyze, monitor, and maintain data governance. It provides guidance on sensitivity labels, item metadata, and data refresh status, offering insights into the governance status and actions for improvement.

Note

Review the [Microsoft Fabric administration](#) documentation for more information.

3. Explore data teams and Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/introduction-end-analytics-use-microsoft-fabric/3-data-team>

Explore data teams and Microsoft Fabric

Completed

Microsoft Fabric's unified data analytics platform makes it easier for data professionals to collaborate on projects. Fabric increases collaboration between data professionals by removing data silos and the need for multiple systems.

Traditional roles and challenges

In a traditional analytics development process, data teams often face several challenges due to the division of data tasks and workflows.

Data engineers process and curate data for analysts, who then use it to create business reports. This process requires extensive coordination, often leading to delays and misinterpretations.

Data analysts often need to perform downstream data transformations before creating Power BI reports. This process is time-consuming and can lack the necessary context, making it harder for analysts to connect directly with the data.

Data scientists face difficulties integrating native data science techniques with existing systems, which are often complex, and makes it challenging to efficiently provide data-driven insights.

Evolution of collaborative workflows

Microsoft Fabric simplifies the analytics development process by unifying tools into a SaaS platform. Fabric allows different roles to collaborate effectively without duplicating efforts.

- **Data engineers** can ingest, transform, and load data directly into OneLake using Pipelines, which automate workflows and support scheduling. They can store data in lakehouses, using the Delta-Parquet format for efficient storage and versioning. Notebooks provide advanced scripting capabilities for complex transformations.
- **Analytics engineers** bridge the gap between data engineering and analysis by curating data assets in lakehouses, ensuring data quality, and enabling self-service analytics. They can create semantic models in Power BI to organize and present data effectively.
- **Data analysts** can transform data upstream using dataflows and connect directly to OneLake with Direct Lake mode, reducing the need for downstream transformations. They can create interactive reports more efficiently using Power BI.
- **Data scientists** can use integrated notebooks with support for Python and Spark to build and test machine learning models. They can store and access data in lakehouses and integrate with Azure Machine Learning to operationalize and deploy models. The predictions they generate can also serve as grounding data for Copilot and AI agents.
- **Low-to-no-code users** and **citizen developers** can discover curated datasets through the OneLake catalog and use Power BI templates to quickly create reports and dashboards. They can also use dataflows to perform simple ETL tasks without relying on data engineers, or ask questions of their data in natural language using Copilot.

Every role in the data team contributes to the organization's ability to use AI effectively. Data engineers who maintain clean, well-governed data in OneLake build the foundation that Copilot and AI agents rely on. Analytics engineers who create consistent semantic models give AI tools the business context needed to generate accurate, meaningful answers.

4. Enable and use Microsoft Fabric

Enable and use Microsoft Fabric

Completed

Before you can explore the end-to-end capabilities of Microsoft Fabric, it must be enabled for your organization. You might need to work with your IT department to enable Fabric for your organization, including one of the following roles:

- *Fabric administrator*: Manages Fabric settings and configurations.
- *Power Platform administrator*: Oversees Power Platform services, including Fabric.
- *Global administrator*: Has implicit Fabric admin rights through organization-wide permissions.

Enable Microsoft Fabric

Admins can enable Fabric in the **Admin portal > Tenant settings** in the Power BI service. Fabric can be enabled for the entire organization or for specific Microsoft 365 or Microsoft Entra security groups. Admins can also delegate this ability to other users at the capacity level.

Note

If your organization isn't using Fabric or Power BI today, you can sign up for a [free Fabric trial](#) to explore its features.

Create workspaces

Workspaces are collaborative environments where you can create and manage items like lakehouses, warehouses, and reports. All data is stored in OneLake and accessed through workspaces. Workspaces also support data lineage view, providing a visual view of data flow and dependencies to enhance transparency and decision-making.

In *Workspace settings*, you can configure:

- License type to use Fabric features.
- OneDrive access for the workspace.
- Azure Data Lake Gen2 Storage connection.
- Git integration for version control.
- Spark workload settings for performance optimization.

You can manage workspace access through four roles: *admin*, *contributor*, *member*, and *viewer*. These roles apply to all items in a workspace and should be reserved for collaboration. For more granular access control, use item-level permissions based on business needs.

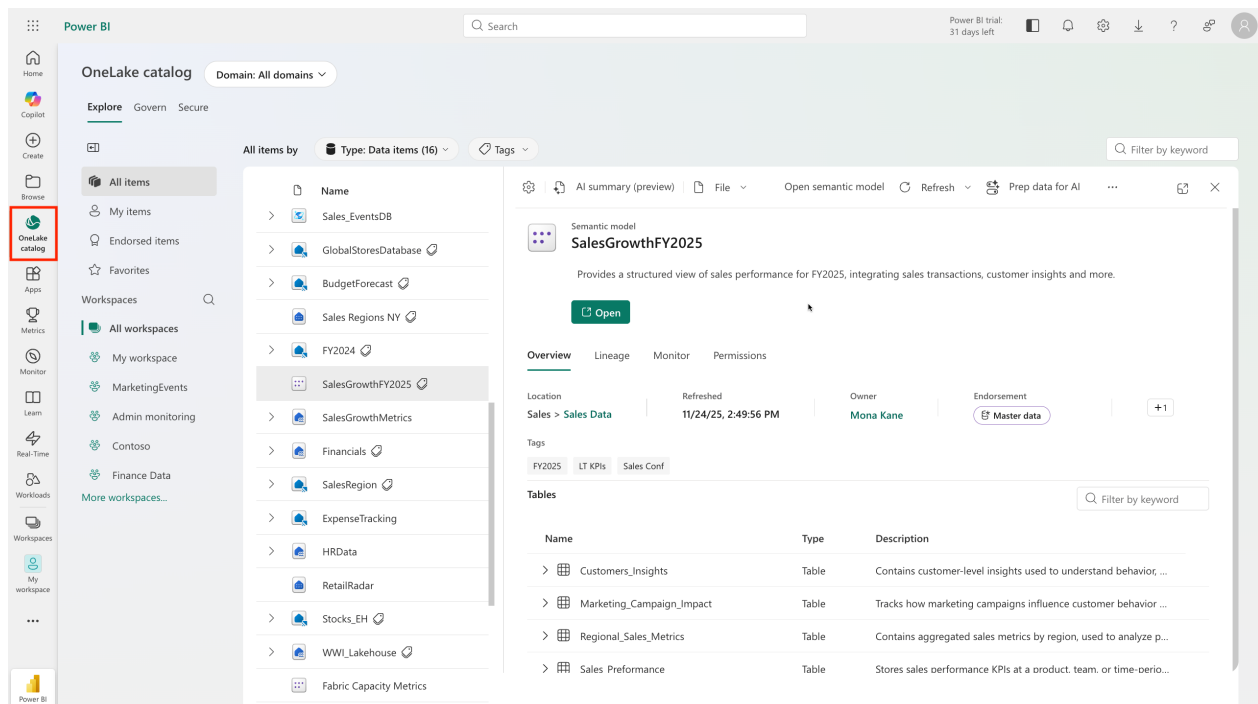
Note

Learn more about workspaces in the [Fabric documentation](#).

Discover data with OneLake catalog

The OneLake catalog in Microsoft Fabric helps you find and access data sources within your organization. You can explore and connect to data sources, ensuring you have the right data for your needs. You only see items that have been shared with you. Here are some considerations when using OneLake catalog:

- Narrow results by workspaces or domains (if implemented).
- Explore default categories to quickly locate relevant data.
- Filter by keyword or item type.



Create items with Fabric workloads

After you create your Fabric enabled workspace, you can start creating items in Fabric. Each workload in Fabric offers different item types for storing, processing, and analyzing data. Fabric workloads include:

- **Data Engineering:** Create lakehouses and operationalize workflows to build, transform, and share your data estate.
- **Data Factory:** Ingest, transform, and orchestrate data.
- **Data Warehouse:** Combine multiple sources in a traditional warehouse for analytics.
- **Real-Time Intelligence:** Process, monitor, and analyze streaming data.
- **Industry Solutions:** Use out-of-the-box industry data solutions.
- **Data Science:** Detect trends, identify outliers, and predict values using machine learning.
- **Databases:** Create and manage databases with tools to insert, query, and extract data.
- **IQ (preview):** Unify data across OneLake and organize it according to the language of your business using ontologies, graphs, and semantic models.
- **Power BI:** Create reports and dashboards to make data-driven decisions.

Fabric integrates capabilities from existing Microsoft tools like Power BI, Azure Synapse Analytics, and Azure Data Factory into a unified platform. Fabric also supports a data mesh architecture, allowing decentralized data ownership while maintaining centralized governance. This design eliminates the need for direct Azure resource access, simplifying data workflows.

AI capabilities in Microsoft Fabric

Fabric includes features that support AI development as well as AI-powered productivity across workloads.

Fabric IQ (preview) is a Fabric workload for unifying data across OneLake and organizing it according to the language of your business. Its core item is the **ontology**, which defines your business concepts, relationships, and rules so that AI agents can reason across domains using consistent business language rather than raw table schemas.

Fabric IQ is one of three IQ workloads that Microsoft provides to give agents access to different aspects of your organization:

- **Fabric IQ** models business data (ontologies, semantic models, and graphs) so agents can reason over analytics in OneLake and Power BI.
- **Foundry IQ** connects structured and unstructured data across Azure, SharePoint, OneLake, and the web so agents can access permission-aware enterprise knowledge.
- **Work IQ** captures collaboration signals from documents, meetings, chats, and workflows, providing agents with insight into how your organization operates.

Each IQ workload is standalone, but you can use them together to provide comprehensive organizational context for agents.

Fabric data agents let you build conversational interfaces where users ask questions about organizational data in natural language. Agents translate those questions into structured queries across your lakehouses, warehouses, and semantic models.

In the Fabric IQ workload, data agents can connect to your ontology as a source, enabling them to understand and use your business concepts when answering questions.

Copilot across workloads

Microsoft Copilot in Fabric is a generative AI assistant available across all Fabric workloads. Copilot helps data professionals and business users complete common tasks more efficiently. Key capabilities include:

- **Code completion and generation:** Copilot provides intelligent code suggestions in notebooks, generates SQL queries from natural language descriptions, and translates questions into Kusto Query Language (KQL) for real-time analysis.
- **Data transformation guidance:** In Data Factory, Copilot supports both citizen and professional data wranglers with code generation for data transformation and plain-language explanations of complex logic.
- **Report and insight generation:** In Power BI, Copilot generates reports automatically, creates page summaries, and lets business users ask questions about their data in natural language.

Note

Copilot in Microsoft Fabric is enabled by default. Administrators can disable Copilot from the **Admin portal > Tenant settings** or control access for specific security groups or at the capacity level.

5. Module assessment

<https://learn.microsoft.com/en-us/training/modules/introduction-end-analytics-use-microsoft-fabric/5-knowledge-check>

Module assessment

Completed

6. Summary

<https://learn.microsoft.com/en-us/training/modules/introduction-end-analytics-use-microsoft-fabric/6-summary>

Summary

Completed

Organizations need to ingest, prepare, govern, and analyze data at scale. They also need that data to be ready for AI workloads like copilots, agents, and machine learning models.

Microsoft Fabric provides a unified foundation for both. In this module, you explored Fabric's OneLake storage architecture, the workloads that serve different analytics needs, and how data teams collaborate within the platform.

You also learned how AI capabilities build on top of well-governed data. Copilot provides intelligent assistance across every workload, data agents let users query organizational data in natural language, and Fabric IQ is a workload that helps agents reason across domains using consistent business language. Together with Foundry IQ and Work IQ, these capabilities position Fabric as both a data platform and an intelligence platform.

Learn more

- [What's new in Microsoft Fabric?](#)
- [Migrate to Microsoft Fabric](#)

Get started with lakehouses in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/1-introduction>

Introduction

Completed

A **lakehouse** in Microsoft Fabric combines the flexible storage of a data lake with the analytical capabilities of a data warehouse. A lakehouse uses Apache Spark and SQL compute engines to process and analyze data at scale, and is built on the **OneLake** storage layer.

A lakehouse is a unified platform that combines:

- The flexible and scalable storage of a data **lake**
- The ability to query and analyze data of a data ware**house**

Imagine your organization relies on a traditional data warehouse for business analytics. The warehouse handles structured transactional data well, but struggles with the growing volume of semi-structured and unstructured data from sources like application logs, IoT devices, and external feeds. Storing and processing these diverse data types requires separate systems, creating data silos and complex integration efforts. Your organization needs a unified solution that handles both structured and unstructured data while maintaining strong analytical capabilities.

In this module, you learn how lakehouses address these challenges. You explore how to create a lakehouse, ingest and transform data, and query lakehouse data using SQL and Spark. You also learn how well-structured lakehouse data supports downstream analytics and AI-powered experiences across the Microsoft Fabric platform.

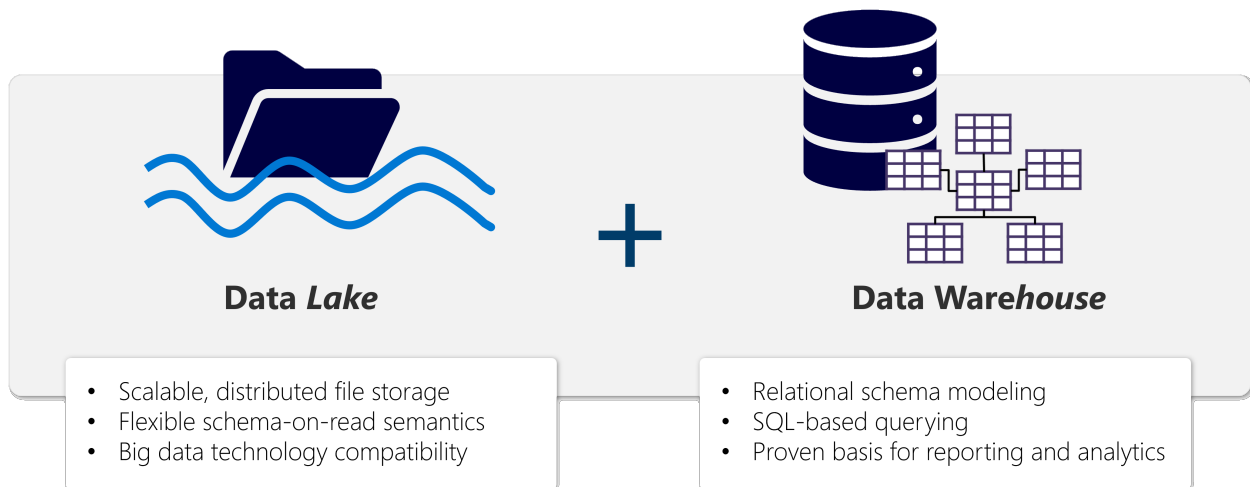
2. Describe lakehouse features and capabilities

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/2-fabric-lakehouse>

Describe lakehouse features and capabilities

Completed

Traditional analytics architectures often force you to choose between two approaches. Data lakes offer flexibility and scalability but lack the structure and performance needed for business analytics. Data warehouses provide strong analytical capabilities but struggle with diverse data formats and can be costly to scale. **Lakehouses** bridge this gap by bringing database-like capabilities directly to your data lake, eliminating the need to maintain separate systems for different workloads.



Understand lakehouse design

A lakehouse organizes data into two main areas: **Tables** and **Files**. Understanding this separation helps you design effective data workflows.

Tables folder: This folder contains Delta Lake tables that provide structured, queryable data. Tables in this folder:

- Support SQL queries through the SQL analytics endpoint
- Enforce schemas and support ACID transactions
- Can be accessed in Power BI for reporting
- Benefit from automatic optimization and maintenance

Files folder: This folder stores raw or semi-structured data files in their native format. Files in this folder:

- Support any file format (CSV, JSON, Parquet, images, documents)
- Provide flexibility for data exploration and processing
- Can be staged before transformation into tables
- Don't enforce schema or support direct SQL queries

This separation lets you maintain both raw data (for compliance or reprocessing) and structured tables (for analytics) within the same lakehouse. You can process files using Spark notebooks or Dataflows Gen2, then load the results into tables for querying and reporting.

Understand Delta Lake tables

At the heart of a lakehouse are **Delta Lake tables**. Delta Lake is an open-source storage layer that brings reliability to data lakes. When you create a table in a lakehouse, the data is stored in Delta format in the underlying OneLake storage.

Delta Lake tables provide several key advantages:

- **ACID transactions:** Delta Lake ensures data consistency even when multiple users read and write data simultaneously.
- **Schema enforcement:** Delta Lake validates that the data you write matches the table schema, preventing corrupt data.
- **Time travel:** Delta Lake maintains a transaction log that lets you query previous versions of your data or roll back changes.
- **Efficient updates and deletes:** Unlike traditional data lake files, Delta tables support efficient update and delete operations.

Each Delta table consists of Parquet data files plus a transaction log that tracks all changes. This architecture enables both batch and streaming workloads to work reliably with the same data.

Manage lakehouse access

When you centralize data in your lakehouse, protecting that data becomes critical. Fabric provides layered access controls to secure lakehouse data at multiple levels.

Use **workspace roles** for collaborators who need access to all items in the workspace. Use **item-level sharing** to grant read-only access for specific needs, such as analytics or Power BI report development.

For granular control, the SQL analytics endpoint supports **row-level** and **column-level security**, so you can restrict what specific users see when they query through SQL. If you organize tables into schemas, you can also apply **schema-level permissions** to control access by business domain.

Fabric lakehouses also support data governance features, including sensitivity labels, and can be extended by using Microsoft Purview with your Fabric tenant.

Note

For more information, see the [Security in Microsoft Fabric](#) documentation.

Build a foundation for intelligent analytics

The data you structure in a lakehouse doesn't just serve traditional reports and dashboards. Well-organized lakehouse data becomes the foundation that intelligent experiences across Microsoft Fabric depend on.

When you create tables with clear schemas, consistent naming conventions, and descriptive column names, you make that data accessible to both human analysts and AI-powered tools. Fabric IQ data agents can query your lakehouse tables through the SQL analytics endpoint, translating natural language questions into SQL queries that return accurate answers. The quality of those answers depends directly on how well you structure and document your data.

Copilot capabilities in Fabric also benefit from well-structured lakehouse data. Copilot in Power BI can generate reports and answer business questions when it can reason over clearly defined tables and relationships. The same lakehouse data can feed semantic models that support natural language exploration across Microsoft 365 experiences.

This means the investment you make in organizing, naming, and structuring lakehouse data pays dividends beyond your immediate analytics needs. Good data engineering practices in the lakehouse create a reusable foundation for intelligent experiences across the platform.

3. Ingest and transform data in a lakehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/3-work-lakehouse>

Ingest and transform data in a lakehouse

Completed

Your lakehouse needs data before it can deliver insights. Whether you're loading files from local storage, connecting to data across clouds, or building transformation pipelines, understanding ingestion and transformation techniques is essential. Start by creating a lakehouse and exploring the tools available to get your data in and ready for analysis.

Create and explore a lakehouse

You create lakehouses within a Fabric-enabled workspace. A lakehouse has two main storage areas and a SQL analytics endpoint.

- **Tables:** Stores Delta Lake tables that provide structured, queryable data. These tables support SQL queries, enforce schemas, and provide ACID transaction guarantees. You can query them through the SQL analytics endpoint and analyze them with Power BI.
- **Files:** Stores raw or semi-structured data files in their native format (CSV, JSON, Parquet, and others). These files don't enforce a schema and provide flexibility for data exploration and processing before transformation into tables.

When you create a lakehouse, **schemas are enabled by default**. A schema named **dbo** is created automatically. Schemas let you organize tables into logical groups based on business domains or functions, such as `sales`, `marketing`, or `hr`. You can create other schemas to keep tables organized as your lakehouse grows. Schema-enabled lakehouses also support schema-level permissions and cross-workspace queries using the four-part namespace (`workspace.lakehouse.schema.table`).

Tip

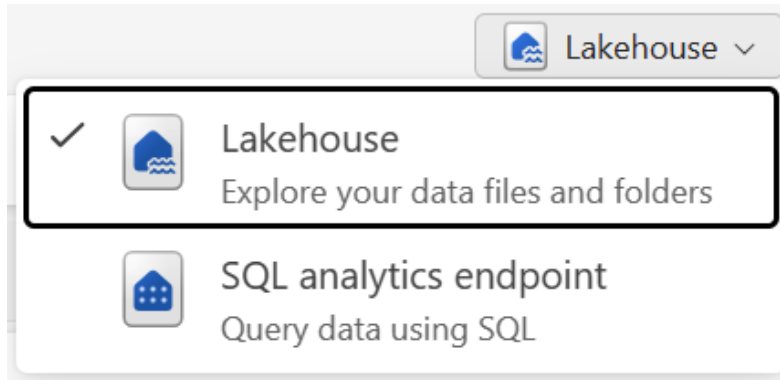
Clear schema organization improves discoverability for everyone working with the lakehouse, including Fabric IQ data agents that translate natural language questions into SQL queries against your tables.

You can work with your lakehouse in two modes:

- **Lakehouse explorer:** Add and interact with tables, files, and folders. This mode allows you to manage data, upload files, create tables, and make changes to your lakehouse. You can also add reference

lakehouses to the explorer pane, so you can browse and manage tables across multiple lakehouses side by side.

- **SQL analytics endpoint:** Query Delta tables using T-SQL in *read-only* mode. You can create views, functions, and apply SQL security, but you can't modify the underlying data.



Ingest data into a lakehouse

Ingesting data into your lakehouse is the first step in your ETL (extract, transform, load) process. Use any of the following methods to bring data into your lakehouse.

- **Upload:** Upload local files or folders directly through the lakehouse explorer.
- **Load to Table:** Select a file or folder in the lakehouse explorer and choose **Load to Table** to create a Delta table without writing code. This no-code option supports Parquet and CSV files, and lets you append or overwrite data in new or existing tables.
- **Dataflows Gen2:** Import and transform data using Power Query.
- **Notebooks:** Use Apache Spark to ingest, transform, and load data programmatically.
- **Data Factory pipelines:** Use the Copy data activity to move data from external sources.

Consider your data loading pattern when ingesting data. You can load raw data as files for staging and later processing, or load directly into tables when the data is already in a supported format.

Note

For more information on ingestion options, see the [Options to get data into the Fabric Lakehouse](#) documentation.

Access data using shortcuts

Shortcuts let you integrate data into your lakehouse without copying it. A shortcut references data in external storage and makes it appear as a folder in your lakehouse.

Shortcuts are valuable because they reduce the amount of times you have to copy data. You can create a shortcut to a different storage account, another cloud provider, or other items in Fabric. OneLake manages source data permissions and credentials. When you access data through a shortcut to another OneLake location, OneLake uses your identity to authorize access to the target data. You must have permissions in the target location to read the data.

You can also create **schema shortcuts** that map an entire schema to a folder of Delta tables in another lakehouse or in Azure Data Lake Storage Gen2. All referenced tables appear as local tables within the schema.

Note

For more information on how to use shortcuts, see the [OneLake shortcuts documentation](#).

Transform data in a lakehouse

Most data requires transformation before loading into tables. You might ingest raw data directly into the Files area, then transform and load it into tables. Regardless of your ETL design, you can transform and load data using the same tools available for ingestion.

- **Notebooks** are favored by data engineers familiar with programming languages including PySpark, SQL, and Scala. Copilot in notebooks can generate transformation code from natural language descriptions and explain existing Spark code.
- **Dataflows Gen2** are suited for users familiar with Power BI or Excel, since they use the Power Query interface.
- **Pipelines** provide a visual interface to orchestrate ETL processes. Pipelines can include multiple activities that run in sequence or parallel.

4. Query and analyze lakehouse data

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/4-explore-data-lakehouse>

Query and analyze lakehouse data

Completed

After you ingest and transform data, it's ready for analysis. A lakehouse in Microsoft Fabric provides multiple ways to query and analyze data, so you can choose the right tool for each task. Whether you prefer SQL for familiar querying patterns or Spark for complex analysis, the lakehouse accommodates your workflow.

Query data using the SQL analytics endpoint

The SQL analytics endpoint provides read-only access to lakehouse tables using T-SQL queries. This endpoint is automatically created with every lakehouse. It allows you to query tables without affecting the underlying data files. You can write queries to explore data, filter rows, aggregate values, and join tables, just like you would with a traditional SQL database.

Common use cases for the SQL analytics endpoint include:

- **Ad-hoc queries:** Quickly investigate data to answer business questions.
- **Business intelligence connections:** Connect tools like Power BI, Excel, or Azure Data Studio to retrieve data for reports.

- **Data validation:** Verify transformation results after loading or processing data.

You can also create SQL views to store reusable query logic. Views are useful when you need to apply business rules, simplify complex joins, or provide curated data for downstream consumers. For example, you might create a view that joins sales and product tables and filters for active products only, making it easier for report authors to consume the data.

The SQL analytics endpoint also supports **row-level security** and **column-level security**, so you can control which users see which data when they query through SQL.

Tip

Copilot for SQL queries can help you write T-SQL queries from natural language descriptions. You can describe what you want to analyze, and Copilot suggests query code to accomplish your goal. This approach accelerates query authoring and helps you learn T-SQL patterns.

Query data using Spark notebooks

Notebooks provide a flexible, code-based environment for querying and analyzing lakehouse data. You can use Spark SQL for SQL-like queries or PySpark for programmatic data manipulation. Notebooks are valuable when you need to perform exploratory data analysis, apply statistical methods, or prepare data for machine learning models.

Spark SQL vs PySpark:

- **Spark SQL:** Use SQL syntax within a notebook cell to query lakehouse tables. Write queries like `SELECT * FROM schema.table` to retrieve and analyze data.
- **PySpark:** Use Python code to manipulate data. You can write SQL queries using `spark.sql()` or use the DataFrame API with Python methods like `df.select()` and `df.filter()`.

Choose the approach that fits your preference and the complexity of your analysis. Spark SQL works well for familiar SQL patterns. PySpark provides greater flexibility for complex transformations and integration with Python libraries.

Common use cases for Spark notebooks include:

- **Exploratory data analysis:** Investigate patterns, outliers, and relationships in data.
- **Complex transformations:** Apply business logic that's easier to express in code than SQL.
- **Cross-workspace queries:** Use the four-part namespace (`workspace.lakehouse.schema.table`) to join data across multiple lakehouses in a single query.

Tip

Copilot for notebooks can generate PySpark or Spark SQL code from natural language prompts and explain existing code. This AI-powered assistance accelerates development and helps you learn Spark syntax as you work.

Analyze and visualize with Power BI

Power BI is the business intelligence and reporting layer in Fabric. It serves as the consumption layer where business users access data through interactive reports and dashboards.

Power BI can connect to lakehouse data in two ways:

- **Query the SQL analytics endpoint:** Analysts can connect directly to the SQL analytics endpoint using Power BI or other tools like Excel to run ad-hoc queries and explore data before building reports.
- **Create a semantic model:** You can create a semantic model that references specific lakehouse tables. This model defines relationships, measures, and business logic that Power BI reports consume.

When you build reports on a lakehouse semantic model, Power BI uses **Direct Lake** mode by default. Direct Lake reads data directly from Delta Lake Parquet files without importing or copying data. This approach provides fast query performance while ensuring reports always reflect the current lakehouse data.

Semantic models also support downstream intelligent experiences. When you define clear relationships and business measures in a semantic model, Copilot in Power BI can generate visualizations and answer business questions by reasoning over your lakehouse data.

5. Exercise - Create a Microsoft Fabric lakehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/5-exercise-lakehouse>

Exercise - Create a Microsoft Fabric lakehouse

Completed

A lakehouse in Microsoft Fabric uses OneLake, a highly scalable file store built on Azure Data Lake Storage Gen2. The lakehouse includes a metastore for relational objects like tables and views. These objects use the open-source Delta Lake table format. With Delta Lake, you define table schemas and query them using SQL.

In this exercise, you perform the following tasks:

- Create a lakehouse
- Upload files
- Load file data into tables
- Query lakehouse tables using SQL
- Create a visual query

Note

You need a Microsoft Fabric trial license with the Fabric preview enabled in your tenant. See [Getting started with Fabric](#) to enable your Fabric trial license.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/6-knowledge-check>

Module assessment

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/7-summary>

Summary

Completed

In this module, you learned how to create a lakehouse, bring data in through multiple ingestion methods, and query that data using SQL and Spark. You also explored how organizing lakehouse data with clear schemas and naming conventions creates a foundation that supports not only traditional reporting through Power BI, but also intelligent experiences powered by Fabric IQ and Copilot.

Without a lakehouse, organizations often maintain separate systems for flexible file storage and structured analytics, leading to data silos, duplication, and complex integration. A lakehouse in Microsoft Fabric eliminates that divide by combining both capabilities in a single platform, built on OneLake.

For more information, see the [Data Engineering in Microsoft Fabric](#) documentation.

Get started with data warehouses in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/1-introduction>

Introduction

Completed

Relational data warehouses are at the center of most enterprise business intelligence (BI) solutions. They provide a structured, SQL-based environment where organizations store, query, and analyze business data at scale.

Microsoft Fabric provides a fully managed data warehouse with full transactional T-SQL capabilities, including the ability to create tables and insert, update, and delete data. Because warehouse data is stored in Delta format on OneLake, it integrates seamlessly with other Fabric workloads. This makes the data warehouse a core component of your end-to-end analytics solution and part of the intelligent data foundation that supports Copilot experiences and AI-driven insights across the platform.

Suppose you work at a retail organization that stores structured business data across multiple systems. Your team needs to centralize this data for analytics and reporting using familiar SQL tools. You need to understand when a Fabric data warehouse is the right choice, how to create one, and how to query and transform data using T-SQL.

In this module, you explore the fundamentals of dimensional modeling with fact and dimension tables, then discover what makes Fabric's data warehouse unique, including full T-SQL support with MERGE capabilities, Copilot-assisted query authoring, and seamless integration with OneLake. You practice querying data, structuring tables, and explore the security and monitoring features that keep your warehouse governed and performant. By the end of this module, you'll know how to build a warehouse that serves both human analysts and AI-powered experiences.

2. Understand data warehouses

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/2-understand-data-warehouse>

Understand data warehouses

Completed

A data warehouse is a centralized, structured store designed for analytical queries and reporting. Unlike operational databases that handle day-to-day business transactions, a data warehouse consolidates data from multiple sources into a format optimized for analysis.

Building a modern data warehouse typically involves:

- **Data ingestion** - Moving data from source systems into the warehouse.
- **Data storage** - Storing the data in a format optimized for analytics.
- **Data processing** - Transforming the data into a format ready for consumption by analytical tools.
- **Data analysis and delivery** - Analyzing the data to gain insights and delivering them to the business.

Design a data warehouse

Data warehouses contain tables organized in a schema optimized for multidimensional modeling. In this approach, you group numerical data related to events by different attributes. For instance, you can analyze the total amount paid for sales orders that occurred on a specific date or at a particular store.

Tables in a data warehouse

You organize data warehouse tables to support efficient analysis of large amounts of data. This organization, known as dimensional modeling, involves structuring tables into fact tables and dimension tables.

Fact tables contain the numerical data that you want to analyze. Fact tables typically have a large number of rows and are the primary source of data for analysis. For example, a fact table might contain the total amount paid for sales orders that occurred on a specific date or at a particular store.

Dimension tables contain descriptive information about the data in the fact tables. Dimension tables typically have a few rows and provide context for the data in the fact tables. For example, a dimension table might contain information about the customers who placed sales orders.

In addition to attribute columns, a dimension table contains a unique key column that uniquely identifies each row in the table. In fact, it's common for a dimension table to include two key columns:

- A *surrogate key* is a unique identifier for each row in the dimension table. It's often an integer value that the database management system generates automatically when you insert a new row.
- An *alternate key* is often a natural or business key that identifies a specific instance of an entity in the transactional source system - such as a product code or a customer ID.

You need both surrogate and alternate keys in a data warehouse, because they serve different purposes. Surrogate keys are specific to the data warehouse and help maintain consistency and accuracy. Alternate keys are specific to the source system and help maintain traceability between the data warehouse and the source system.

Special types of dimension tables

Special types of dimensions provide additional context and enable more comprehensive data analysis.

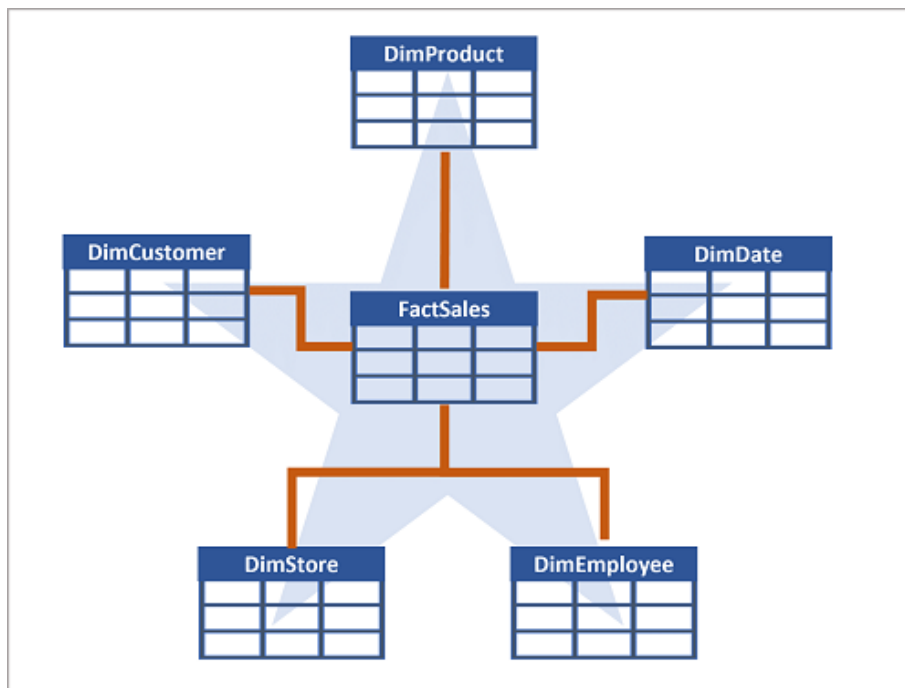
Time dimensions provide information about the time period in which an event occurred. This table enables data analysts to aggregate data over temporal intervals. For example, a time dimension might include columns for the year, quarter, month, and day of a sales order.

Slowly changing dimensions track changes to dimension attributes over time, like changes to a customer's address or a product's price. They're significant in a data warehouse because they enable you to analyze and understand changes to data over time. Slowly changing dimensions ensure that data stays up-to-date and accurate, which is important for making good business decisions.

Data warehouse schema designs

In most transactional databases used in business applications, the data is *normalized* to reduce duplication. In a data warehouse however, the dimension data is denormalized* to reduce the number of joins required to query the data.

Often, a data warehouse uses a *star schema*, in which a fact table relates directly to the dimension tables, as shown in this example:

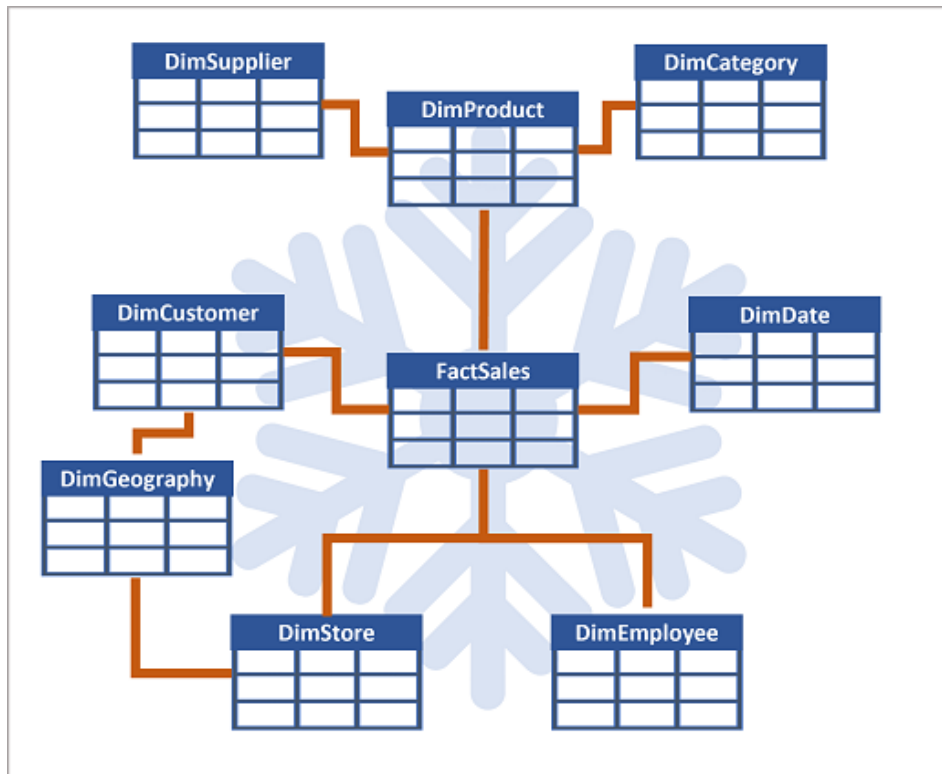


You can use dimension attributes to group fact table numbers at different levels. For example, you could find the total sales revenue for a whole region or just for one customer. You can store the information for each level in the same dimension table.

Tip

See [What is a star schema?](#) for more information on designing star schemas for Fabric.

If there are lots of levels or attributes shared by different things, it might make sense to use a *snowflake schema* instead. Here's an example:



In this case, the **DimProduct** table splits (normalizes) into separate dimension tables for product categories and suppliers.

- Each row in the **DimProduct** table contains key values for the corresponding rows in the **DimCategory** and **DimSupplier** tables.

A **DimGeography** table contains information on where customers and stores are located.

- Each row in the **DimCustomer** and **DimStore** tables contains a key value for the corresponding row in the **DimGeography** table.

3. Understand data warehouses in Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/3-understand-data-warehouse-fabric>

Understand data warehouses in Fabric

Completed

Now that you understand data warehouse fundamentals, let's explore what Microsoft Fabric provides for data warehousing.

Describe a Fabric data warehouse

A Fabric data warehouse is a fully managed, enterprise-scale relational database built on OneLake. It provides full transactional T-SQL capabilities, including DDL statements (CREATE, ALTER, DROP) and DML statements (INSERT, UPDATE, DELETE, MERGE), with full ACID compliance for data consistency.

Data is stored in open Delta format on OneLake, which means other Fabric workloads can access the same data without duplication. You use T-SQL to create tables, load data, build views and stored procedures, and perform transformations, all within a familiar SQL experience.

Key capabilities include:

- **Full T-SQL support** - Write DDL and DML statements, including MERGE for upsert scenarios, using familiar SQL Server syntax.
- **Fully managed** - No infrastructure to configure. Compute scales automatically and independently from storage.
- **OneLake integration** - Warehouse data is stored in Delta format and accessible by other Fabric workloads without duplication.
- **Cross-database querying** - Query data across warehouses and lakehouses without copying data. Use three-part naming (database.schema.table) to join warehouse tables with lakehouse tables in a single query.
- **Familiar tooling** - Connect with SQL Server Management Studio (SSMS), Azure Data Studio, or any SQL client through standard TDS connections.
- **Copilot assistance** - Copilot for Data Warehouse generates SQL queries from natural language, provides code completion as you type, and can explain or fix existing queries in the SQL editor.

Warehouse vs SQL analytics endpoint

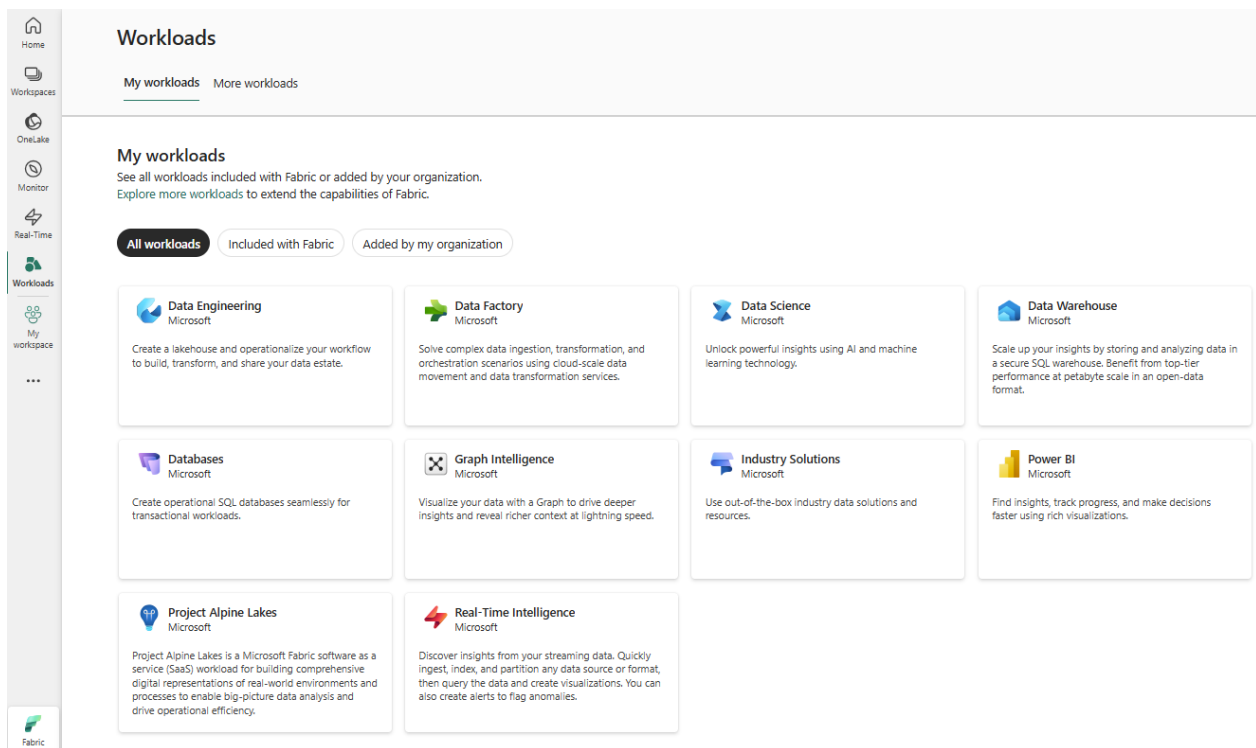
Fabric workspaces can contain two types of SQL-based items that serve different purposes.

Capability	Warehouse	SQL analytics endpoint
Read data	Yes	Yes
Write data (INSERT, UPDATE, DELETE, MERGE)	Yes	No
Create tables (DDL)	Yes	No
Create views and stored procedures	Yes	Yes
Data source	Native warehouse tables	Lakehouse Delta tables

Use a warehouse when you need full read/write T-SQL capabilities. Use the SQL analytics endpoint when you need read-only SQL access to lakehouse data.

Create a data warehouse

You can create a data warehouse in Fabric from the **create hub** or within a **workspace**. After creating an empty warehouse, you can add tables, views, and other objects.



Once your warehouse is created, you can start creating tables and loading data using the SQL query editor in the Fabric portal.

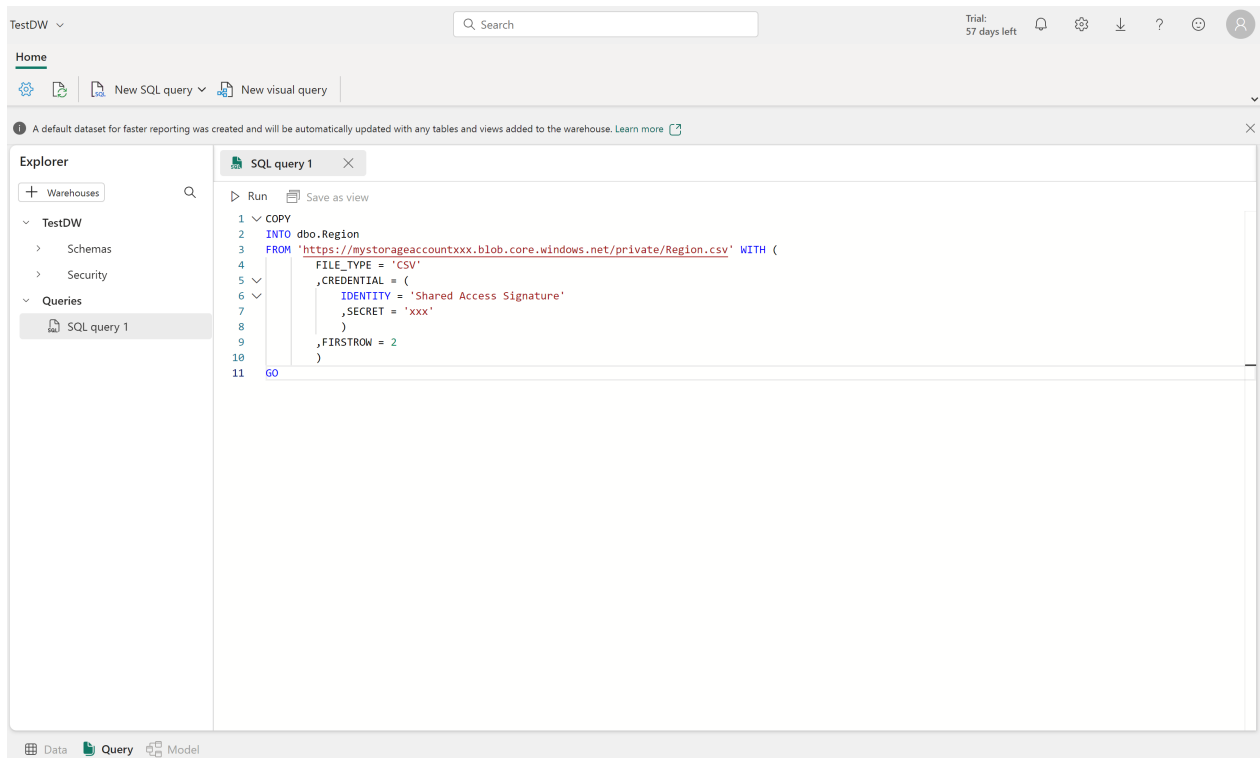
Ingest data into a warehouse

There are several ways to load data into a Fabric data warehouse:

- **COPY INTO** - Bulk load data from external files (CSV, Parquet) in Azure storage into warehouse tables.
- **OPENROWSET** - Query files directly from external storage or OneLake locations for ad hoc analysis or ingestion, without creating tables first.
- **Pipelines and Dataflows** - Use Data Factory pipelines or Dataflows Gen2 for orchestrated data movement and transformation.
- **Cross-database queries** - Query lakehouse tables directly from the warehouse using three-part naming, without copying data.

You can use the `COPY INTO` T-SQL command to bulk load data from files. For example, the following statement loads data from a CSV file into a table:

```
COPY INTO dbo.Region
FROM 'https://mystorageaccount.blob.core.windows.net/data/Region.csv'
WITH (
    FILE_TYPE = 'CSV',
    CREDENTIAL = (
        IDENTITY = 'Shared Access Signature',
        SECRET = 'xxx'
    ),
    FIRSTROW = 2
)
GO
```



Tip

If you have tables in a lakehouse that you want to query from your warehouse without making changes, use cross-database querying instead. You don't need to copy the data.

Create tables and load data

After creating a warehouse and choosing an ingestion method, the next step is to define your tables and load data into them.

You create tables using T-SQL `CREATE TABLE` statements. Define columns with appropriate data types for analytics workloads.

```
CREATE TABLE dbo.DimCustomer
(
    CustomerKey INT NOT NULL,
    CustomerAltKey NVARCHAR(10) NOT NULL,
    CustomerName NVARCHAR(100) NOT NULL,
    Region NVARCHAR(50) NULL
);
GO

CREATE TABLE dbo.FactSales
(
    SalesKey INT NOT NULL,
    CustomerKey INT NOT NULL,
    ProductKey INT NOT NULL,
    DateKey INT NOT NULL,
    SalesAmount DECIMAL(10,2) NOT NULL,
    Quantity INT NOT NULL
)
```

```
);  
GO
```

Choose data types that balance precision with storage efficiency. Use `INT` for key columns, `NVARCHAR` for text that may include special characters, and `DECIMAL` for financial values that require precision.

Use staging tables for data loading

A common pattern in data warehousing is to land raw data in staging tables before transforming and loading it into final dimension and fact tables. Staging tables mirror the structure of your source data and act as a temporary holding area.

After loading data into staging tables using `COPY INTO` or pipelines, you transform and insert it into your dimensional model:

```
INSERT INTO dbo.FactSales (SalesKey, CustomerKey, ProductKey, DateKey, SalesAmount, Quantity)  
SELECT  
    s.OrderID,  
    c.CustomerKey,  
    p.ProductKey,  
    d.DateKey,  
    s.Amount,  
    s.Qty  
FROM dbo.StgSales AS s  
INNER JOIN dbo.DimCustomer AS c ON s.CustomerID = c.CustomerAltKey  
INNER JOIN dbo.DimProduct AS p ON s.ProductID = p.ProductAltKey  
INNER JOIN dbo.DimDate AS d ON s.OrderDate = d.DateValue;  
GO
```

This pattern keeps your source data intact while you apply business rules and key lookups during the load process.

Understand table clones

You can create zero-copy table clones in a Fabric data warehouse. Clones copy table metadata while still referencing the same underlying data files in OneLake. The data itself isn't duplicated, which keeps storage costs low.

The following code example shows you how to create a clone with T-SQL:

```
--Clone creation within the same schema  
CREATE TABLE dbo.Employee AS CLONE OF dbo.EmployeeUSA;
```

Table clones are useful for development and testing, data recovery after a failed release, and preserving data at specific points in time for historical reporting.

Tip

For more information, see the [Clone table in Microsoft Fabric](#) documentation.

Now that you understand Fabric's warehouse capabilities, let's explore how to query and transform your data.

4. Query and transform data

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/4-query-transform-data>

Query and transform data

Completed

Now that you know how to create a warehouse and ingest data, you can start exploring and shaping that data for analytics.

Raw data rarely arrives in the exact format you need for analysis. You might need to join tables, filter rows, aggregate values, or restructure data before it's useful for reporting. A Fabric data warehouse gives you two tools for this work: the SQL query editor for T-SQL and the Visual query editor for a no-code approach.

Query data with the SQL query editor

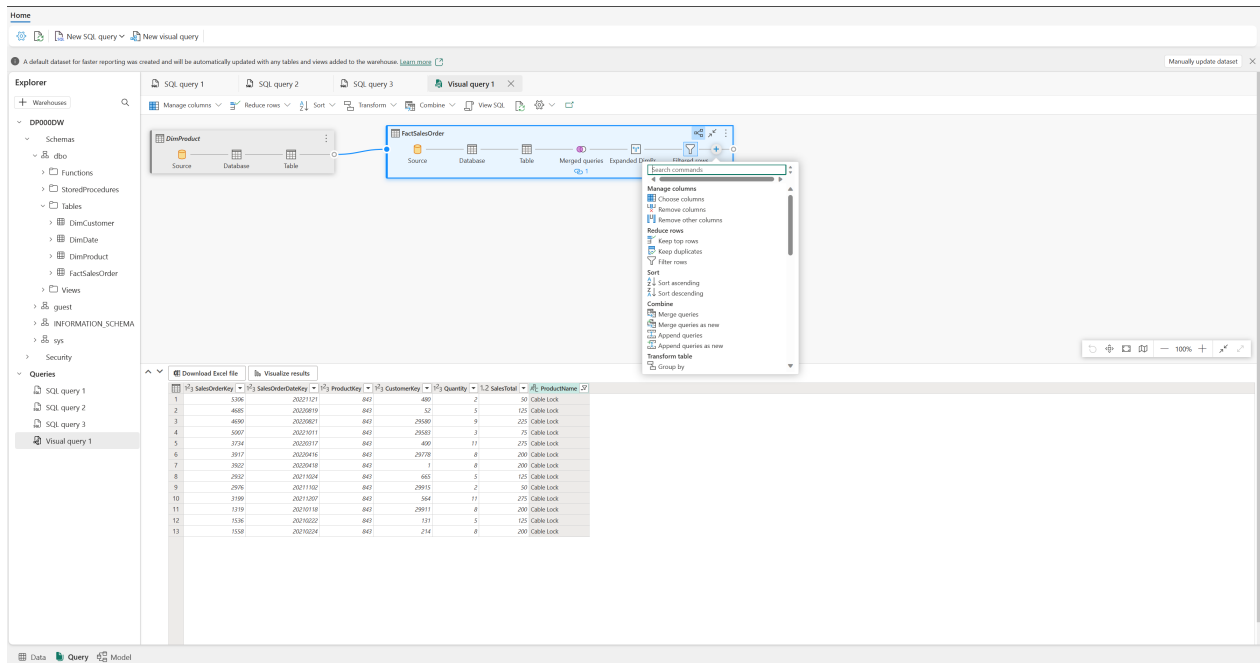
The **SQL query editor** provides a query experience that includes IntelliSense, code completion, syntax highlighting, client-side parsing, and validation. This will feel familiar if you have experience writing T-SQL in SQL Server Management Studio (SSMS) or Azure Data Studio (ADS).

To create a new query, use the **New SQL query** button in the menu. Copilot for Data Warehouse is available in the editor to help generate queries from natural language, complete code as you type, and explain or fix existing queries.

Query data with the Visual query editor

The *Visual query editor* provides an experience similar to the [Power Query online diagram view](#). Use the **New visual query** button to create a new query.

Drag a table from your data warehouse onto the canvas to get started. You can then use the **Transform** menu at the top of the screen to add columns, filters, and other transformations. You can also use the (+) button on the visual itself to perform similar actions.



Transform data with views and stored procedures

Beyond ad hoc queries, you can save transformation logic as reusable objects in the warehouse.

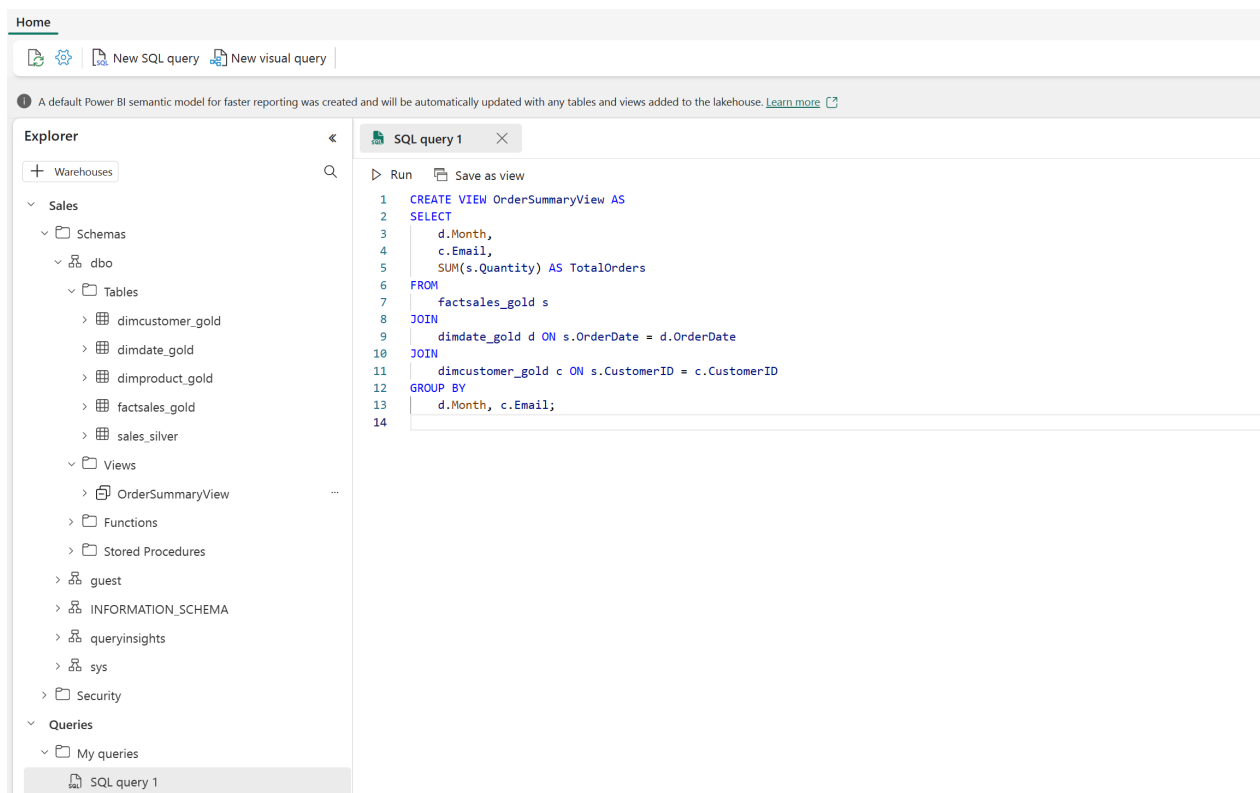
Views define a saved query that you can reference like a table. Use views to standardize how analysts access data, such as combining fact and dimension tables into a reporting-friendly format or filtering rows to a specific business context. For example:

Stored procedures contain T-SQL logic that you can execute on demand. Use stored procedures for repeatable transformation tasks, like loading staging data into final tables or applying business rules.

Views and stored procedures also help make your data more accessible to AI-powered tools. Copilot and Fabric IQ data agents can query views just like tables, so standardizing data access through well-named views improves the accuracy of natural language queries.

The following code shows a sample view and the screenshot shows how to use the code in the warehouse SQL query editor:

```
CREATE VIEW dbo.vw_SalesByRegion
AS
SELECT
    c.Region,
    SUM(f.SalesAmount) AS TotalSales,
    COUNT(f.OrderID) AS OrderCount
FROM dbo.FactSales AS f
INNER JOIN dbo.DimCustomer AS c
    ON f.CustomerKey = c.CustomerKey
GROUP BY c.Region;
```



5. Model data in a warehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/5-model-data>

Model data in a warehouse

Completed

Without data modeling, every consumer has to figure out which tables relate to each other, write their own aggregation logic, and guess at column meanings. Data modeling solves this problem by embedding structure, business logic, and documentation directly into the warehouse. In a Microsoft Fabric warehouse, you prepare data for clarity, define relationships between tables, standardize access through views and measures, and publish semantic models for reporting. These modeling choices affect every downstream experience, including T-SQL queries, Power BI reports, and AI-driven natural language analytics.

Prepare data for consumption

Before you define relationships or add calculations, you need to clean up what consumers see. Raw warehouse tables often contain staging tables, surrogate key columns, and internal flags that are meant for ETL processing, not for analysis. These objects create noise when consumers browse the data. Preparing the warehouse for consumption means surfacing only what's relevant and making it understandable.

In the model view, you can take several steps to improve the consumer experience:

- **Hide internal objects** like staging tables, surrogate key columns, and ETL artifacts that clutter the field list.
- **Rename columns** to use business-friendly names where the warehouse column names are technical or abbreviated. For example, rename `CustRgn` to `Customer Region`.
- **Add descriptions** to tables and columns so that consumers understand what the data represents without referring to external documentation.

These steps matter beyond just tidiness. Copilot in Power BI and Fabric IQ data agents rely on table names, column names, and descriptions to interpret natural language questions and generate accurate SQL or DAX. A column named `Customer Region` with a description like "Geographic region of the customer's primary address" produces better natural language results than `CustRgn` with no description.

With clean, well-named tables in place, you're ready to define how those tables connect to each other.

Understand relationships between tables

A relationship is a logical connection between two tables that enables filtering, grouping, and aggregation across those tables. In a star schema, relationships connect fact tables to dimension tables through shared key columns.

For example, a `CustomerKey` column that exists in both `FactSales` and `DimCustomer` establishes the link that enables analysis of sales by customer attributes like region, segment, or account type.

Each relationship has two important properties.

- **Cardinality** describes how rows in the two tables correspond. In a star schema, fact-to-dimension relationships are typically many-to-one, meaning many fact rows map to a single dimension row.
- **Cross-filter direction** determines which way filters propagate between the tables. Single direction, where the dimension filters the fact table, is the standard setting for most star schema designs because it keeps filter behavior predictable and performant.

Without defined relationships, every consumer who wants to combine data across tables needs to write explicit JOIN logic. Relationships eliminate that repetition by encoding the connection once. When you create a semantic model from the warehouse, these relationships inform how Power BI, Copilot, and Fabric IQ data agents interpret the data. Data agents, for example, use relationships to generate accurate joins when translating natural language questions into SQL.

Note

Most data warehouses use dimensional modeling. Relationships can be created to shape a **star schema**, which is an ideal model for analytics. For more information, see the [Design dimensional models in Microsoft Fabric](#) module.

Standardize data access with views and measures

Now that your tables are clean and connected, the next step is to give consumers reliable, consistent ways to query and calculate against that data. Without standardization, each team writes its own join logic, applies its own filters, and defines its own formulas, which leads to conflicting results.

Views provide this consistency for T-SQL consumers. A view encapsulates join logic, filters, and column selections into a reusable query that consumers reference like a table. For example, a view that joins fact and dimension tables, filters to completed orders, and surfaces only the columns analysts need gives every T-SQL consumer a reliable starting point. Views also serve as stable data sources for reports. Rather than building reports directly against base tables that might change, you can point reports at views that present a consistent shape.

Measures provide the same consistency for DAX calculations. A measure is a reusable DAX expression that defines a calculation like a total, average, ratio, or count. You create measures directly in the warehouse model view by selecting a table and adding a new measure. For example, a `Total Sales` measure that sums the `SalesAmount` column ensures every consumer uses the same calculation.

Because the measure definition lives with the data, it becomes the single source of truth for that metric. When the business changes how it calculates revenue, you update the measure in one place rather than tracking down every report that contains its own formula.

Together, views and measures cover both sides of consumption: views standardize how T-SQL consumers access and query data, while measures standardize how business calculations appear in reports and dashboards.

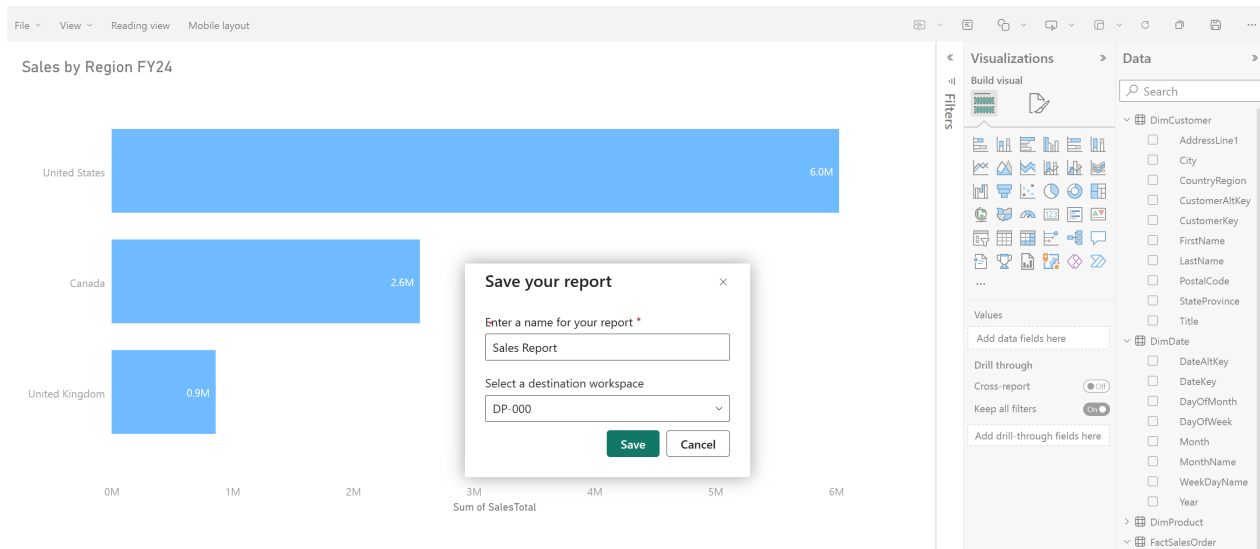
Tip

DAX formulas and advanced measure design are covered in depth in later modules. For views and stored procedures, see the previous unit on querying and transforming data.

Create a semantic model for Power BI reporting

With prepared tables, defined relationships, and standardized views and measures in place, the warehouse is ready for downstream reporting. Teams that query the warehouse directly by using T-SQL or connect through third-party tools can work with the warehouse model as-is. However, when you want to build interactive Power BI reports and dashboards, creating a semantic model is the next step.

Semantic models created from a Fabric warehouse use Direct Lake mode. Unlike traditional import mode, which copies data into Power BI memory, Direct Lake reads data directly from OneLake Parquet files. This means reports reflect the latest warehouse data without requiring scheduled refreshes. It also means you avoid the storage and processing overhead of maintaining a separate copy of the data.



Tip

Semantic model design and scalability patterns are covered in greater depth in [Design scalable semantic models](#). This unit focuses on modeling data in the warehouse itself.

6. Secure and monitor a warehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/6-security-monitor>

Secure and monitor a warehouse

Completed

Security and monitoring are critical aspects of managing your data warehouse. Fabric provides multiple layers of protection and visibility tools to help you control access and understand query performance.

Security

Fabric data warehouse security operates at multiple levels, from workspace access down to individual rows and columns. This design allows you to support the distinct needs of your organization by still allowing the democratization of data, but with governance.

Workspace roles

Data in Fabric is organized into *workspaces*, and workspace roles are the first layer of access control. Assign users to appropriate roles based on the level of access they need. For example, Admins have full control, while Viewers can view items but can't make changes.

Tip

For more information, see [Workspaces in Power BI](#).

Item permissions

In addition to workspace roles, you can grant **item permissions** to share individual warehouses without granting access to the entire workspace. This granularity is useful when you need to share a warehouse for downstream consumption with specific users.

Grant the following permissions as needed:

- **Read** - Allows the user to connect using the SQL analytics endpoint.
- **ReadData** - Allows the user to read data from any table or view in the warehouse.
- **ReadAll** - Allows the user to read raw parquet files in OneLake.

Note

A user connection to the SQL analytics endpoint fails without Read permission at a minimum.

Granular SQL security

For more precise access control, Fabric data warehouse supports granular security using T-SQL. These features let you restrict data visibility without changing the underlying tables:

- **Object-level security** - Control access to specific tables, views, or procedures.
- **Row-level security (RLS)** - Restrict which rows a user can see using WHERE clause predicates.
- **Column-level security (CLS)** - Restrict which columns are visible to specific users.
- **Dynamic data masking** - Mask sensitive data (such as email addresses or account numbers) from non-privileged users.

Securing your warehouse data is important for both regulatory compliance and for ensuring that AI-powered tools like Copilot and data agents operate within governed boundaries. Security policies you define in T-SQL are enforced regardless of how the data is accessed.

Tip

Row-level security, column-level security, and dynamic data masking are covered in depth in [Secure a Microsoft Fabric data warehouse](#).

Monitoring

Monitoring warehouse activity helps you identify performance issues, optimize queries, and understand usage patterns.

Query insights

Query insights provides a central location for historical query data and actionable performance information. It retains data for 30 days and helps you identify long-running queries, track performance changes over time, and understand which queries consume the most resources.

Query insights uses system views that you can query directly:

- `queryinsights.exec_requests_history` - Returns information about each completed SQL request.
- `queryinsights.long_running_queries` - Returns queries ranked by execution time.
- `queryinsights.exec_sessions_history` - Returns information about completed sessions.

Dynamic management views

You can also use *dynamic management views* (DMVs) to monitor active connections, sessions, and requests in real time. For example, use `sys.dm_exec_requests` to identify currently running queries:

```
SELECT request_id, session_id, start_time, total_elapsed_time
FROM sys.dm_exec_requests
WHERE status = 'running'
ORDER BY total_elapsed_time DESC;
```

Important

You must be a workspace Admin to run the `KILL` command to terminate long-running sessions. Members, Contributors, and Viewers can see their own results but can't see other users' queries.

7. Exercise - Create and query a warehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/7-exercise>

Exercise - Create and query a warehouse

Completed

Now it's your turn to create a data warehouse in Fabric and analyze your data.

Note

You need a Microsoft Fabric trial license with the Fabric preview enabled in your tenant. See [Getting started with Fabric](#) to enable your Fabric trial license.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

8. Module assessment

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/9-summary>

Summary

Completed

In this module, you learned how data warehouses use dimensional modeling to organize data into fact and dimension tables, and what makes a Fabric data warehouse unique. You explored querying and transforming data with T-SQL and the visual query editor, structured tables into a star schema, and applied security features like row-level security and dynamic data masking to protect your data.

Without a platform like Microsoft Fabric, building this kind of warehouse environment would require provisioning and managing dedicated SQL infrastructure, configuring separate storage and compute, and manually integrating data across siloed systems. A Fabric data warehouse eliminates that complexity by combining full T-SQL capabilities with OneLake integration in a single, governed platform that supports both traditional analytics and AI-powered experiences.

To learn advanced T-SQL transformation patterns like staging workflows, incremental loads, and MERGE-based upserts, continue to the [Transform data using T-SQL](#) module.

Learn more

- [What is Fabric Data Warehouse?](#)
- [Query the warehouse](#)

Get started with Real-Time Intelligence in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/1-introduction>

Introduction

Completed

Imagine you're a data analyst for a delivery company, responsible for monitoring package delivery performance across your network of distribution centers, delivery vehicles, and customer routes. Your operations team needs to know immediately when delivery delays occur, which routes are experiencing issues, and how customer satisfaction is trending in real time. Currently, your delivery reports are generated overnight, meaning by the time you identify a problem—such as a vehicle breakdown or weather-related delays—hundreds of packages might already be behind schedule and customers are left waiting without updates. You need continuous monitoring of delivery trucks, customer feedback systems, and warehouse scanners to track package movements and delivery performance as events happen, not hours later.

In this scenario, Real-Time Intelligence in Microsoft Fabric can be used to help you work with streaming data from GPS trackers on delivery vehicles, package scanning systems, and customer notification platforms as events occur. Unlike traditional batch processing that shows historical snapshots of delivery status, Real-Time Intelligence could help you analyze package movements and delivery performance as it happens. This approach could help you spot route delays quickly, estimate delivery windows based on current conditions, and set up automated customer notifications while creating workflows for route optimization.

By the end of this module, you'll understand how Microsoft Fabric Real-Time Intelligence components work together to create real-time analytics solutions, how to ingest, process, store and query real-time data, and how to visualize data in motion and automate responses to changing conditions.

2. What is real-time data analytics?

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/2-define-real-time-analytics>

What is real-time data analytics?

Completed

Real-time analytics is the practice of processing, analyzing, and acting on data as it's generated, typically within seconds to minutes of when events occur. Unlike traditional analytics that works with static snapshots of historical data stored in databases, real-time analytics operates on data that's actively flowing through your systems, enabling immediate insights and rapid responses to changing conditions. This approach is also known as *near* real-time analytics, since there's always some degree of processing and network latency involved.

Understand events and streams

Events are records of things that happen in a system. They capture moments when something occurs, changes, or is completed. Examples include website clicks, stock price changes, customer purchases, patient vital sign changes, or equipment sensor readings. Think of them as digital records or log entries that document activity across your systems.

A **stream** is essentially a sequence of events, typically ordered by the time an event occurred. Each event in the stream represents something that happened at a specific moment. Events flow through streams continuously as they occur. For example, a stream of equipment temperature sensor readings contains temperature readings over many points of time. This continuous flow of event information allows you to detect patterns over time, identify opportunities or risks, and take action immediately after something happens, or in *real-time*.

Streams are the delivery mechanism that carries events from where they happen to where they need to be processed, analyzed, or acted upon.

Components of real-time analytics solutions

To build real-time analytics solutions, you need several integrated capabilities working together:

Real-time data ingestion: Collect data from multiple sources simultaneously, as information is generated. For example: database changes from change data capture, sensors, applications, system logs, and APIs.

Stream processing: Transform and analyze data while it flows from sources to destinations. This includes filtering, aggregating, joining with other data sources, and detecting patterns with minimal latency.

Low-latency storage: Use specialized databases and storage systems designed to handle high-velocity data writes and provide fast query responses.

Interactive dashboards: Create visualizations that update automatically as new data arrives, show current state and trends in real-time.

Automated decision making: Set up event-driven rules and triggers that can initiate actions, send alerts, or start workflows based on real-time conditions.

Use real-time analytics

To use real-time data effectively, information has to be ingested, processed, stored, analyzed, and presented to be actionable. Real-time analytics enables you to:

- **Respond immediately** to opportunities or problems as they emerge
- **Optimize operations** by adjusting resources and configurations based on current conditions
- **Enhance customer experiences** through personalized, contextual interactions

- **Prevent issues** by detecting anomalies before they become critical problems

Real-Time Intelligence in Microsoft Fabric brings all these capabilities together in a single platform. Through components like Eventstreams for data ingestion and transformation, Eventhouses for analytics-optimized storage, the Real-Time hub for data discovery, Real-Time Dashboards for visualization, and Activator for automated alerts and actions, Real-Time Intelligence enables you to monitor critical events, trigger automated responses, track business processes, and analyze patterns in real-time, turning what happens in your systems into actionable insights.

3. Ingest and transform real-time data

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/3a-define-real-time-hub>

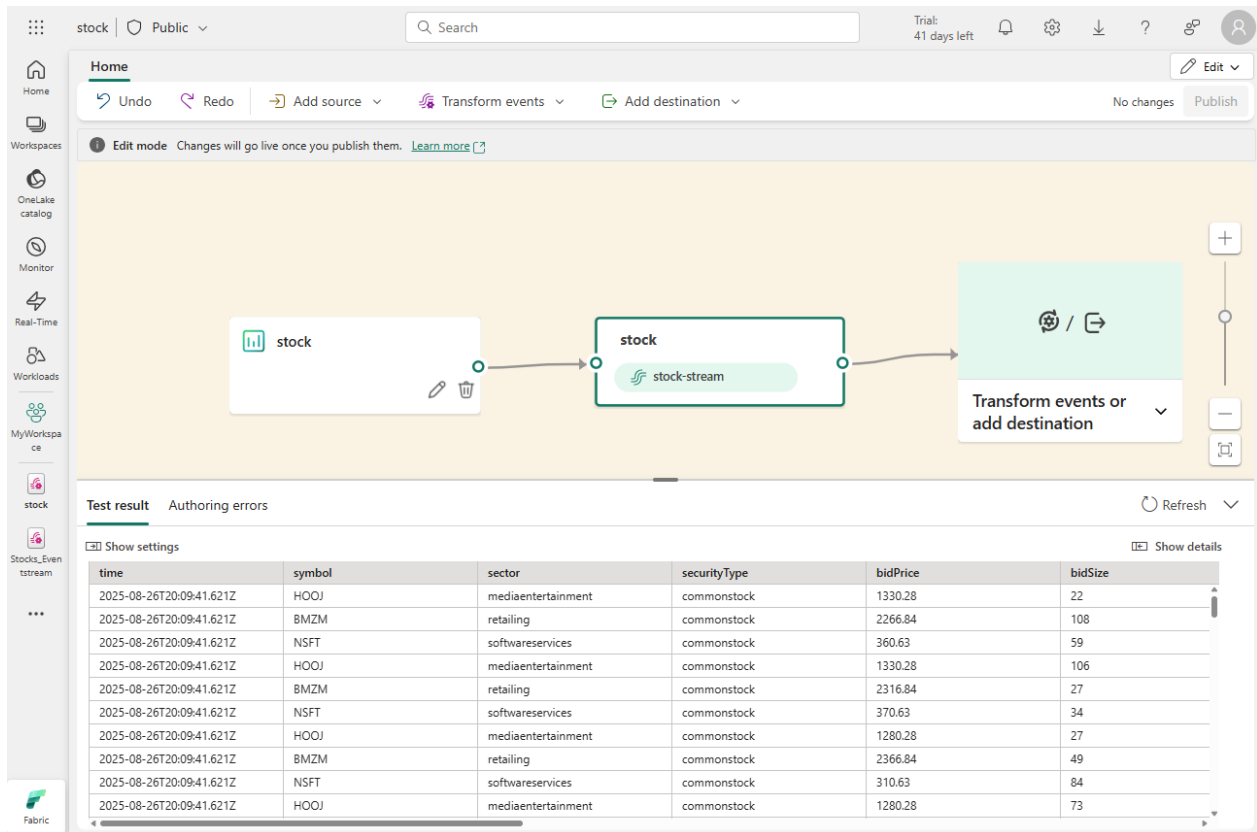
Ingest and transform real-time data

Completed

Real-Time Intelligence in Microsoft Fabric provides two primary approaches for ingesting streaming data: using eventstreams or directly ingesting data into a KQL database in an Eventhouse.

Eventstreams for data ingestion and transformation

Eventstreams are a way to bring real-time events into Fabric, to transform them, and then route data to a destination.



The image shows the three main components of an Eventstream: **sources** where data originates, **transformations** optional processing applied to the data, and **destinations** where the processed data is sent.

Think of the Eventstream components like a water pipe system. The source is your faucet, transformations are filters along the way and you need a destination like a sink or bucket to collect and use the water.

Next, let's review each component of an Eventstream.

Data sources for eventstreams

Once you create an eventstream in Fabric, you can connect it to a wide range of data sources. You can stream data from Microsoft sources and also ingest data from non-Microsoft platforms including:

- **Microsoft sources**, like Azure Event Hubs, Azure IoT Hubs, Azure Service Bus, Change Data Capture (CDC) feeds in database services, and others.
- **Azure events**, like Azure Blob Storage events.
- **Fabric events**, such as changes to items in a Fabric workspace, data changes in OneLake data stores, and events associated with Fabric jobs.
- **External sources**, such as Apache Kafka, Google Cloud Pub/Sub, and MQTT (Message Queuing Telemetry Transport) (in preview)

Tip

To see all supported sources, see [Supported sources](#).

Event transformations in eventstreams

Raw data from a source system is rarely in the exact format you need for analysis or storage. Transformations are what make your data useful and actionable. You can transform the data as it flows in an eventstream, enabling you to filter, summarize, and reshape it before storing it. Examples of available transformations include: SQL code, filter, manage fields, aggregate, group by, expand and join.

Tip

For more information about supported transformations, see [Process event data with event processor editor](#) and [Process events using SQL code editor](#).

Data destinations in eventstreams

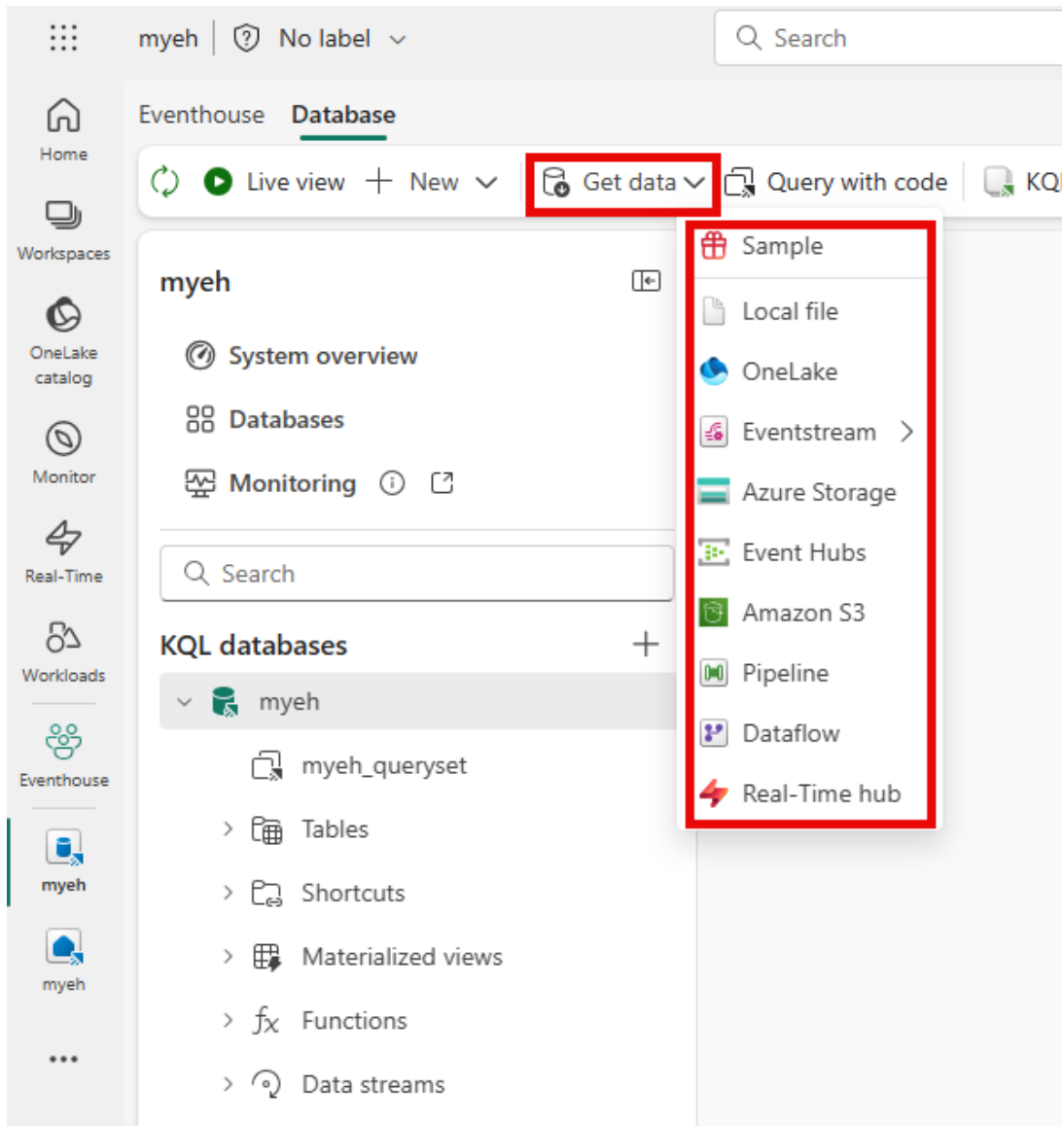
Streaming data flows continuously and is temporary by nature. It requires immediate processing and storage to retain its value. The destination in an eventstream is what makes your real-time data processing actionable. It's where your processed data becomes available for queries, reports, dashboards, alerts, actions, or integration with other systems. You can load the data from your stream into the following destinations: a KQL database in an Eventhouse, Lakehouse, a derived stream, Fabric Activator, or a custom endpoint.

Tip

For more information about supported destinations, see [Add and manage a destination in an eventstream](#).

Direct ingestion to a KQL database in an Eventhouse

Data can also be directly ingested into a KQL (Kusto Query Language) database in an Eventhouse. Some examples of data ingestion sources include: local files, Azure storage, Amazon S3, Azure Event Hubs, OneLake, and more. Data ingestion can be configured using **connectors** or through the **Get data** option in a KQL database as shown in this image.



Tip

For more information about supported ingestion sources for KQL databases in Eventhouses, see [Data sources](#) and [Data connectors overview](#).

Data transformation in a KQL database in Eventhouse with update policies

When directly ingesting data into a KQL database, data first lands in the database, then can be transformed using **update policies**. This is different from eventstream transformations that occur *during* stream processing, before routing data to a destination.

Update policies are automation mechanisms triggered when new data is written to a table. They run a query to transform ingested data and save the result to a destination table.

4. Real-Time Intelligence in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/3-describe-kusto-databases-tables>

Real-Time Intelligence in Microsoft Fabric

Completed

As organizations generate increasing volumes of event-driven data, the ability to process, analyze, and act on data in motion becomes essential for competitive advantage. Real-Time Intelligence provides comprehensive capabilities for working with streaming data with minimal latency.

Explore Real-Time Intelligence use cases

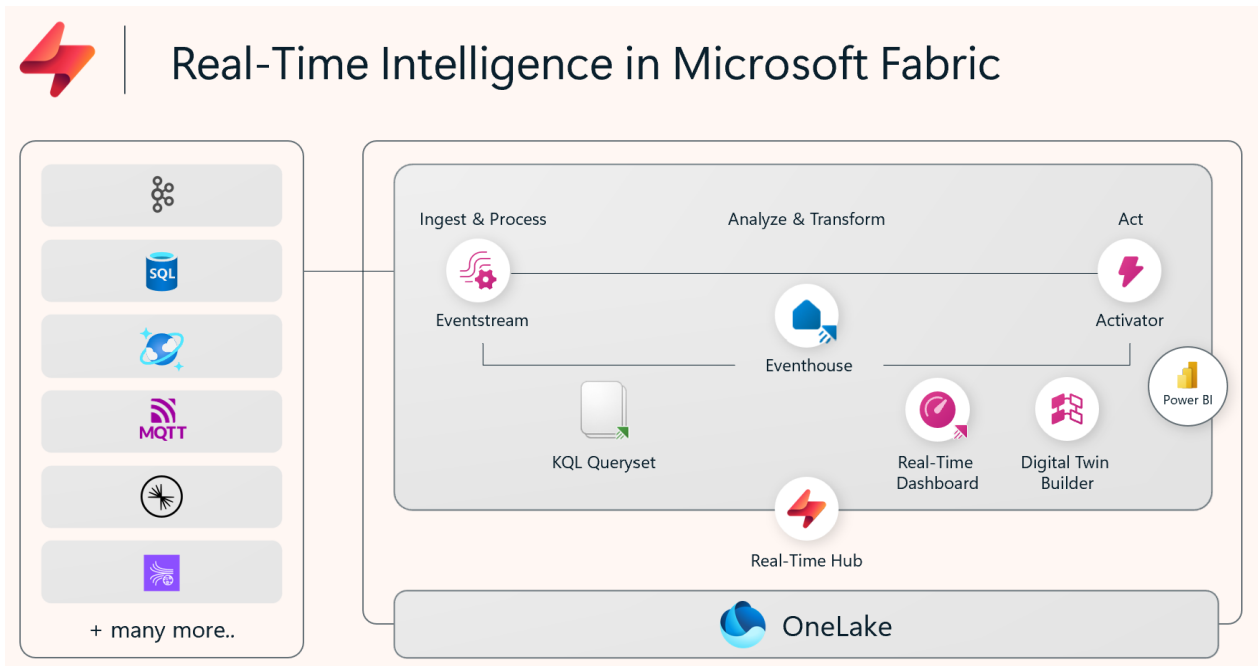
Unlike batch systems that process data on scheduled intervals, Real-Time Intelligence helps you respond to events as they happen, delivering near real-time insights.

Here are some common types of event data and examples of how Real-Time Intelligence can support downstream actions and business responsiveness:

- **Delivery tracking:** Monitor vehicle locations to alert customers when packages are delayed
- **Equipment monitoring:** Track machine temperature to prevent costly breakdowns
- **Fraud detection:** Analyze purchase patterns to block suspicious transactions immediately
- **Website performance:** Monitor page load times to improve user experience
- **System health:** Track application errors to maintain service reliability

Real-Time Intelligence components

Microsoft Fabric's Real-Time Intelligence is an integrated set of components that work together to handle streaming data from capture through automated response.



The diagram shows how Real-Time Intelligence components work together for end-to-end processing. Each component handles a specific stage of the real-time analytics process:

Ingest and process data in motion with Eventstreams

Data ingestion and processing can happen through **Eventstreams**, which capture streaming data from various sources and apply real-time transformations as data flows through the system. Eventstreams can filter, enrich, and transform your data and route it to different destinations.

Store real-time data in an Eventhouse

Real-Time Intelligence stores data in KQL (Kusto Query Language) databases in **Eventhouses**. These databases are designed for time-series data and fast ingestion of streaming data. The storage integrates with OneLake, making your data available to other Fabric tools.

Analyze data with KQL Queryset

KQL Queryset provides a workspace for running and managing queries against KQL databases. The KQL Queryset allows you to save queries for future use, organize multiple query tabs, and share queries with others for collaboration. The KQL Queryset also supports T-SQL queries, allowing you to use familiar SQL syntax alongside KQL for data analysis.

Visualize insights with Real-Time Dashboard

Real-Time Dashboards connect directly to KQL databases and refresh automatically as new data arrives. These dashboards let you explore data interactively and monitor both current conditions and historical trends.

Act on data with Activator

Automated actions can be configured with **Activator**, which continuously monitors streaming data against user-defined rules and thresholds. When conditions are met, Activator can send notifications, trigger workflows in Power Automate, execute Fabric data pipelines or notebooks, creating event-driven automation that responds to real-time conditions.

Discover streaming data with the Real-Time hub

The Fabric **Real-Time hub** is a central location where you can discover and manage all of the data-in-motion that you have access to. It gives you a way to ingest streaming data from Azure and from external sources and it lets you subscribe to Azure and Fabric events.

Think of the Real-Time hub as your streaming data catalog where you can see what's happening in near real-time across your organization. There are connectors you can use to ingest data into Microsoft Fabric from various sources. For example, you might connect to IoT sensor streams through Azure Event Hubs, subscribe to Azure Blob Storage events, use Change Data Capture (CDC) to stream database changes, or monitor Fabric workspace events.

Once you have configured a connection to data source or event source, these items become the foundation for event driven decision making and a wide range of real-time analytics solutions, from building dashboards and setting up alerts to triggering automated workflows and analyzing trends in your data.

The screenshot displays the Microsoft Fabric Real-Time hub interface. The left sidebar contains navigation icons for Home, Workspaces, OneLake catalog, Monitor, and Real-Time (which is highlighted with a red box). The main content area is titled "Discover, analyze, and act on data-in-motion" and features three interactive cards: "Subscribe to OneLake events", "Act on Job events", and "Visualize data". Below these cards is a "Recent streaming data" section with a filter bar and a table of data streams.

Data	Source item	Item owner	Workspace	Endorsement
fabric_events_eventstream-stream	fabric_events_eventstream	System Administrator	My workspace	—
MyTutorialEventstream-stream	MyTutorialEventstream	System Administrator	MYRTIWS	—
MyRawData	MyTutorial	System Administrator	MYRTIWS	—
TransformedData	MyTutorial	System Administrator	MYRTIWS	—
TutorialEventstream3-stream	TutorialEventstream3	System Administrator	RTI_WS	—
new_event_stream-stream	new_event_stream	System Administrator	RTI_WS	—
RTISample-stream	RTISample	System Administrator	My workspace	—
Bikestream	RTISample	System Administrator	My workspace	—

To access the real-time hub, select the **Real-Time** icon in the main Fabric menu bar.

The real-time hub organizes data-in-motion into several main categories:

- **Data sources:** Browse and connect to available streaming data sources, such as Microsoft sources, database change data capture feeds, and external sources from other cloud providers
- **Azure sources:** Discover and configure Azure streaming data sources such as Azure IoT Hub, Azure Service Bus, Azure Data Explorer DB, and more

- **Fabric events:** Subscribe to system-generated events in Fabric that you can access, like job status changes, events produced by action on files or folders in OneLake, and Fabric workspace item changes
- **Azure events:** Subscribe to system events from Azure services that can be used to trigger automated responses such as actions on files or folders in Azure blob storage

In the Real-Time hub, you can preview and explore your streaming data by navigating directly to eventstreams or KQL databases in eventhouses for deeper analysis and querying. You can also build automated responses using *Activator* rules that trigger actions like notifications, workflows, or data processing when specific patterns are detected.

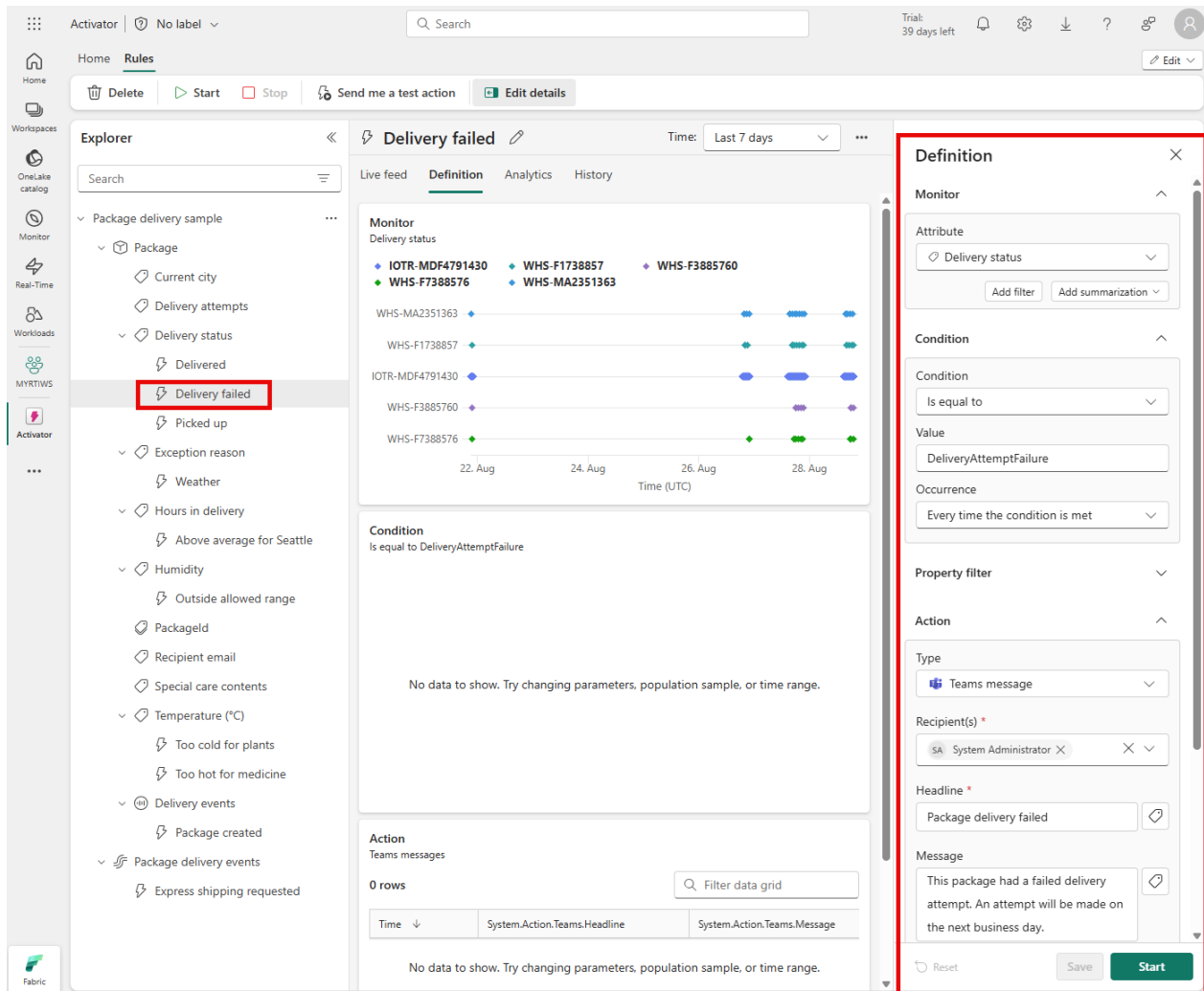
5. Automate actions

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/4c-activator>

Automate actions

Completed

Activator is a technology in Microsoft Fabric that enables automated processing of events that trigger actions. For example, you can use Activator to notify you by email when a value in an Eventstream deviates from a specific range or to run a notebook to perform some Spark-based data processing logic when a Real-Time Dashboard is updated.



This image shows a rule configured to alert when package delivery failures happen, demonstrating how you can automate responses to specific business events.

Understand Activator key concepts

Activator operates based on four core concepts: *Events*, *Objects*, *Properties*, and *Rules*.

- **Events** - Each record in a stream of data represents an *event* that has occurred at a specific point in time.
- **Objects** - The data in an event record can be used to represent an *object*, such as a sales order, a sensor, or some other business entity.
- **Properties** - The fields in the event data can be mapped to *properties* of the business object, representing some aspect of its state. For example, a *total_amount* field might represent a sales order total, or a *temperature* field might represent the temperature measured by an environmental sensor.
- **Rules** - The key to using Activator to automate actions based on events is to define *rules* that set conditions under which an action is triggered based on the property values of objects referenced in events. For example, you might define a rule that sends an email to a maintenance manager if the temperature measured by a sensor exceeds a specific threshold.

Use cases for Activator

Activator can help you in various scenarios, such as dynamic inventory management, real-time customer engagement, and effective resource allocation in cloud environments. It's a powerful tool for any circumstance that requires real-time data analysis and automated actions.

Use Activator to:

- Initiate marketing actions when product sales drop.
- Send notifications when temperature changes could affect perishable goods.
- Flag real-time issues affecting the user experience on apps and websites.
- Trigger alerts when a shipment hasn't been updated within an expected time frame.
- Send alerts when a customer's account balance crosses a certain threshold.
- Respond to anomalies or failures in data processing workflows immediately.
- Run ads when same-store sales decline.
- Alert store managers to move food from failing grocery store freezers before it spoils.

Tip

For more information about Activator, see [What is Activator?](#).

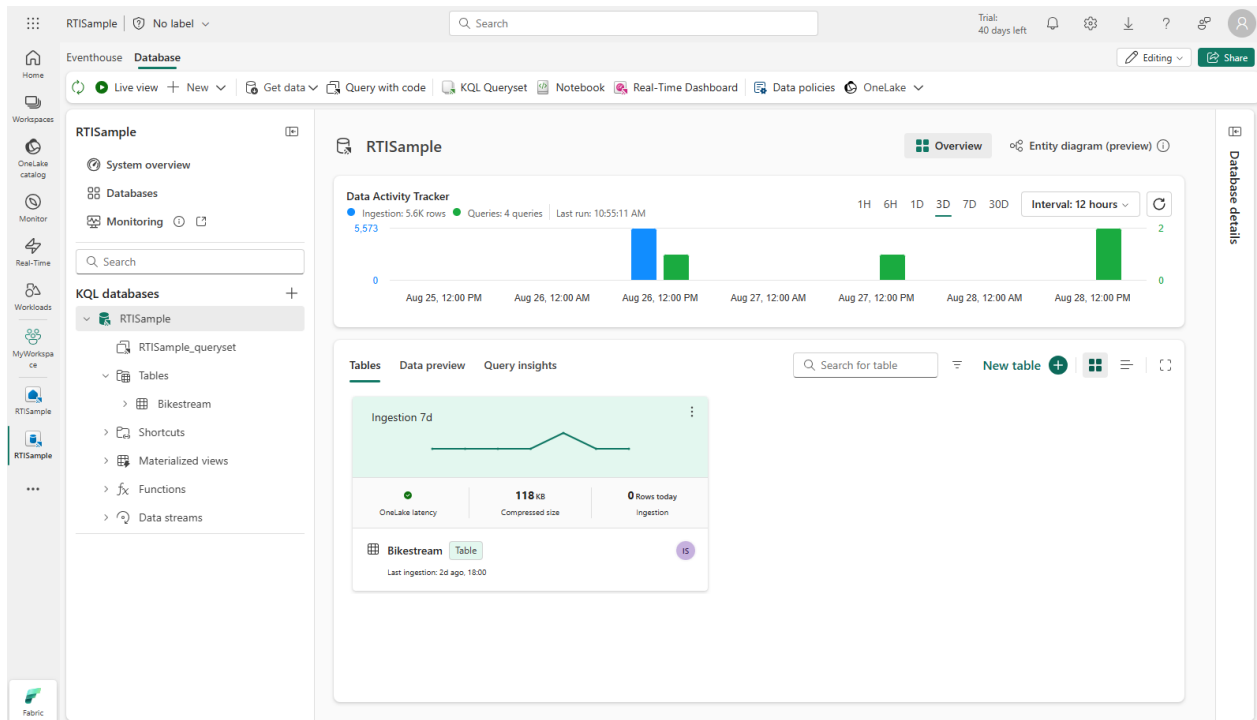
6. Store and query real-time data

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/4-write-queries-kusto-query-language>

Store and query real-time data

Completed

KQL databases in an Eventhouse are where you store and query real-time data that flows from Eventstreams and other streaming sources. Once data is loaded into tables, you can use the Kusto Query Language (KQL) or T-SQL to query your data.



Within an eventhouse, you can create:

- **KQL databases:** Real-time optimized data stores that host a collection of tables, stored functions, materialized views, shortcuts and data streams.
- **KQL querysets:** Collections of KQL queries that you can use to work with data in KQL database tables. A KQL queryset supports queries written using Kusto Query Language (KQL) or a subset of the Transact-SQL language.

Understand the power of Kusto Query Language (KQL)

To query data in a table in a KQL database, you can use the **KQL**. KQL is specifically designed for analyzing large volumes of structured, semi-structured, and unstructured data with exceptional performance. KQL databases are optimized for time-series data and index incoming data by ingestion time and partition it for optimal query performance. KQL is the same language used in Azure Data Explorer, Azure Monitor Log Analytics, Microsoft Sentinel, and in Microsoft Fabric.

Get familiar with KQL syntax

KQL queries are made of one or more query statements. A query statement consists of a table name followed by operators that `take`, `filter`, `transform`, `aggregate`, or `join` data. For example, to print any 10 rows in the `stock` table, execute:

```
stock
| take 10
```

A more complex example might aggregate data to find average stock prices over the last 5 minutes:

```
stock
| where ["time"] > ago(5m)
```

```
| summarize avgPrice = avg(todouble(bidPrice)) by symbol  
| project symbol, avgPrice
```

Tip

To learn more about KQL, see [Kusto Query Language \(KQL\) overview](#).

Automate data processing with management commands

Beyond basic querying, you can automate data processing through **management commands** including:

- **Update policies:** Automatically transform incoming data and save it to different tables as it arrives.
- **Materialized views:** Precalculate and store summary results for faster queries.
- **Stored functions:** Save frequently used query logic that you can reuse across multiple queries.

Tip

For more information about working with KQL databases, including detailed examples of update policies, materialized views, and stored functions, see [Work with real-time data in a Microsoft Fabric Eventhouse](#).

Other query options

Using SQL

KQL databases in Eventhouses also support a subset of common T-SQL expressions for data professionals already familiar with T-SQL syntax. For example:

```
SELECT TOP 10 * FROM stock;
```

Use Copilot to help with queries

Microsoft Fabric includes Copilot for Real-Time Intelligence, which can help you write queries to extract insights from your Eventhouse data. Copilot uses AI to understand what you're looking for and can generate the required query code.

Tip

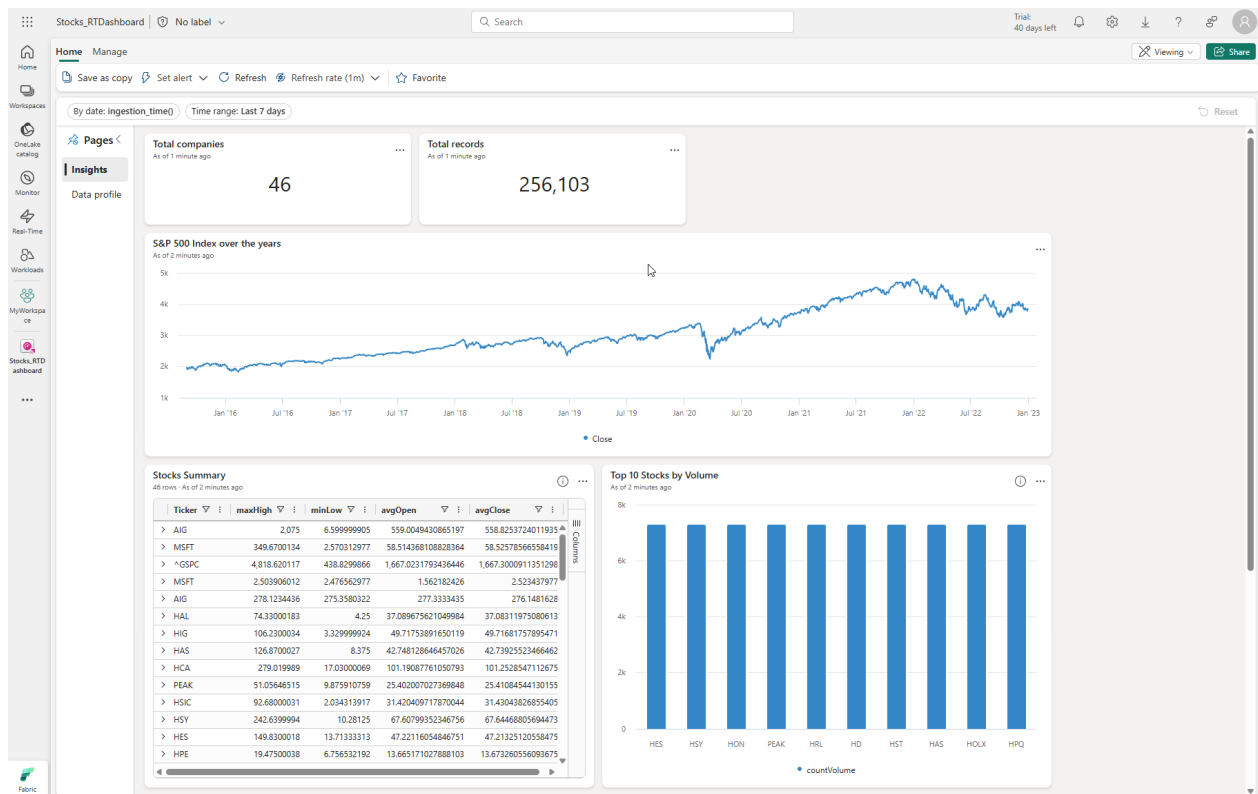
To learn more about Copilot for Real-Time Intelligence, see [Copilot for Real-Time Intelligence](#).

7. Visualize real-time data

Visualize real-time data

Completed

Real-Time Dashboards provide a way to pin data visualizations to a single visual interface, enabling you to surface real-time insights at a glance. Each *tile* in a dashboard shows you different information based on a KQL query that extracts real-time data from tables in an eventhouse.



Create a Real-Time Dashboard

You can create a Real-Time Dashboard in a workspace and then configure its source, or you can create one directly from a KQL queryset in an eventhouse.

Dashboards are composed of one or more *tiles*, each containing a visualization based on a KQL query expression. By default, the visualization shows the results of the query as a table; but you can edit the tile to customize how the data is displayed.

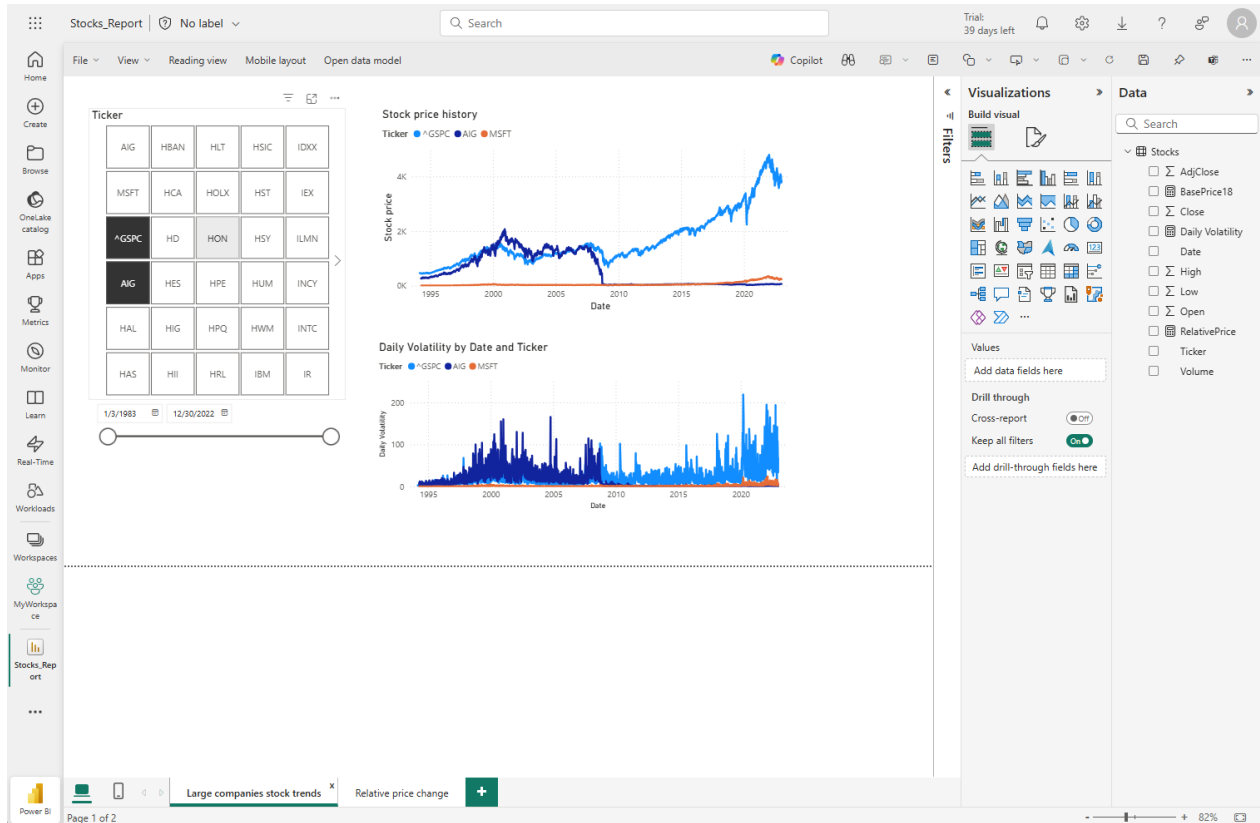
When published, tiles let you explore the data they contain interactively by drilling into the data and using a visual interface to filter and aggregate the data, and change the visualization type.

Tip

To learn more about Real-Time Dashboards, see [Create a Real-Time Dashboard](#).

Visualize real-time data with Power BI

You also can create Power BI reports from your KQL database data.



Tip

To learn more about using Power BI with real-time data in Fabric, see [Visualize data in a Power BI report](#).

8. Exercise - Get started with Real-Time Intelligence in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/5-exercise-use-kusto-query-data-onelake>

Exercise - Get started with Real-Time Intelligence in Microsoft Fabric

Completed

Now it's time to try out Real-Time Intelligence yourself.

In this exercise, you'll ingest real-time data into Microsoft Fabric. You'll then query and visualize the data before defining an alert to automate an action based on a threshold value in the real-time data stream.

Note

You need a Microsoft Fabric tenant to complete this exercise. For more information about accessing Microsoft Fabric, see [Getting started with Fabric](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

9. Module assessment

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/6-knowledge-check>

Module assessment

Completed

10. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-kusto-fabric/7-summary>

Summary

Completed

In this module, you learned how Microsoft Fabric Real-Time Intelligence enables you to ingest, transform, store, visualize, and act on data in motion. You explored the Real-Time Hub for discovering streams, Eventstreams for ingesting and processing event data, and KQL databases in Eventhouses for fast, time-based analytics using KQL. You also learned how to visualize streaming insights with Real-Time dashboards and Power BI, and how to automate responses using Activator.

Tip

For more information about Real-Time Intelligence in Microsoft Fabric, see [Real-Time Intelligence documentation in Microsoft Fabric](#).

Get started with data science in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/1-introduction>

Introduction

Completed

Imagine you're working for a supermarket and you want to know how much bread you need to have in stock every week to meet customer demands while also avoiding food waste.

Or maybe you want to analyze your customers to understand how to best target them with personalized offers.

Whenever you want to make informed decisions within an organization, you can use data science to get insights from the data you have. **Data science** is a combination of mathematics, statistics, and computer engineering.

When you perform data science, you can analyze your data and identify *complicated patterns* that can provide you with meaningful insights for your organization. You can use data science to create **artificial intelligence (AI)** models that encompass the complicated patterns you find in your data. A common approach is to use data science to train **machine learning** models using libraries like `scikit-learn` in Python to achieve AI.

Taking a data science project from beginning to end can be a challenge. Microsoft Fabric offers one workspace to manage an end-to-end data science project.

In this module, you learn about a typical data science project. Additionally, you explore which Microsoft Fabric features you can use for each part of the data science process.

2. Understand the data science process

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/2-data-science>

Understand the data science process

Completed

A common way to extract insights from data is to visualize the data. Whenever you have complex datasets, you may want to dive deeper and try to find intricate patterns in the data.

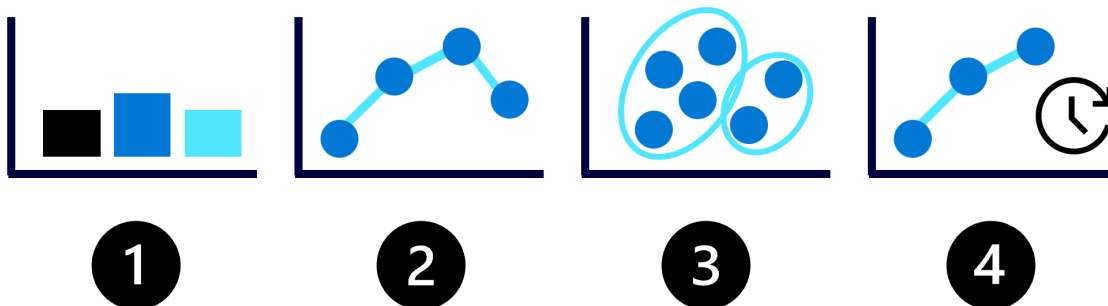
As a data scientist, you can train machine learning models to find patterns in your data. You can use those patterns to generate new insights, or predictions. For example, you can predict the expected number of products you expect to sell in the coming week.

Though training the model is important, it's not the only task in a data science project. Before exploring a typical data science process, let's explore common machine learning models you can train.

Explore common machine learning models

The purpose of machine learning is to train models that can identify patterns in large amounts of data. You can then use the patterns to make predictions that provide you with new insights on which you can take actions.

The possibilities with machine learning may appear endless, so let's begin by understanding the four common types of machine learning models:



1. **Classification**: Predict a categorical value like whether a customer may churn.
2. **Regression**: Predict a numerical value like the price of a product.
3. **Clustering**: Group similar data points into clusters or groups.
4. **Forecasting**: Predict future numerical values based on time-series data like the expected sales for the coming month.

To decide which type of machine learning model you need to train, you first need to understand the business problem and the data available to you.

Understand the data science process

To train a machine learning model, the process commonly involves the following steps:



1. **Define the problem**: Together with business users and analysts, decide on what the model should predict and when it's successful.
2. **Get the data**: Find data sources and get access by storing your data in a Lakehouse.
3. **Prepare the data**: Explore the data by reading it from a Lakehouse into a notebook. Clean and transform the data based on the model's requirements.

4. **Train the model:** Choose an algorithm and hyperparameter values based on trial and error by tracking your experiments with MLflow.
5. **Generate insights:** Use model batch scoring to generate the requested predictions.

As a data scientist, most of your time is spent on preparing the data and training the model. How you prepare the data and which algorithm you choose to train a model can influence your model's success.

You can prepare and train a model by using open-source libraries available for the language of your choice. For example, if you work with Python, you can prepare the data with Pandas and Numpy, and train a model with libraries like [Scikit-Learn](#), [PyTorch](#), or [SynapseML](#).

When experimenting, you want to keep an overview of all the different models you've trained. You want to understand how your choices influence the model's success. By tracking your experiments with MLflow in Microsoft Fabric, you're able to easily manage and deploy the models you've trained.

3. Explore and process data with Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/3-explore-prepare-data>

Explore and process data with Microsoft Fabric

Completed

Data is the cornerstone of data science, particularly when aiming to train a machine learning model for achieving artificial intelligence. Typically, models exhibit enhanced performance as the training dataset size increases. In addition to the quantity of data, the quality of the data is equally crucial.

To guarantee both the quality and quantity of your data, using Microsoft Fabric's robust data ingestion and processing engines is worthwhile. You have the flexibility to opt for either a low-code or code-first approach when establishing the essential data ingestion, exploration, and transformation pipelines.

Ingest your data into Microsoft Fabric

To work with data in Microsoft Fabric, you first need to ingest data. You can ingest data from multiple sources, both local and cloud data sources. For example, you can ingest data from a CSV file stored on your local machine or in an Azure Data Lake Storage (Gen2).

Tip

Learn more about how to [ingest and orchestrate data from various sources with Microsoft Fabric](#).

After connecting to a data source, you can save the data into a Microsoft Fabric **lakehouse**. You can use the lakehouse as a central location to store any structured, semi-structured, and unstructured files. You can then easily connect to the lakehouse whenever you want to access your data for exploration or transformation.

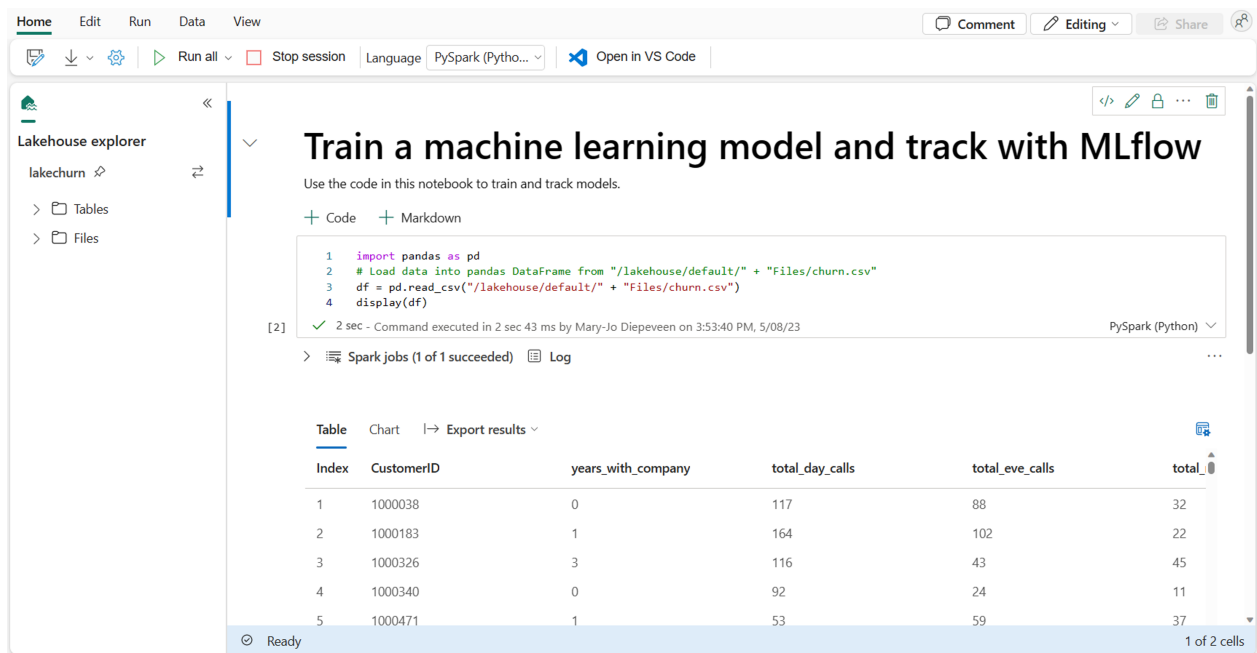
Explore and transform your data

As a data scientist, you may be most familiar with writing and executing code in **notebooks**. Microsoft Fabric offers a familiar notebook experience, powered by Spark compute.

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

Notebooks are automatically attached to Spark compute. When you run a cell in a notebook for the first time, a new Spark session starts. The session persists when you run subsequent cells. The Spark session will automatically stop after some time of inactivity to save costs. You can also manually stop the session.

When you're working in a notebook, you can choose the language you want to use. For data science workloads, you're likely to work with PySpark (Python) or SparkR (R).



The screenshot shows a Microsoft Fabric notebook titled "Train a machine learning model and track with MLflow". The notebook is in PySpark (Python) mode. The code cell contains the following Python code:

```
1 import pandas as pd
2 # Load data into pandas DataFrame from "/lakehouse/default/" + "Files/churn.csv"
3 df = pd.read_csv("/lakehouse/default/" + "Files/churn.csv")
4 display(df)
```

The output of the code cell is a table with 5 rows and 6 columns. The table is titled "Table" and has a "Log" button next to it. The table data is as follows:

Index	CustomerID	years_with_company	total_day_calls	total_eve_calls	total
1	1000038	0	117	88	32
2	1000183	1	164	102	22
3	1000326	3	116	43	45
4	1000340	0	92	24	11
5	1000471	1	53	59	37

The notebook interface also shows a "Lakehouse explorer" on the left side with a tree view containing "lakechurn", "Tables", and "Files". The bottom status bar indicates "Ready" and "1 of 2 cells".

Within the notebook, you can explore your data using your preferred library, or with any of the built-in visualization options. If necessary, you can transform your data and save the processed data by writing it back to the lakehouse.

Prepare your data with the Data Wrangler

To help you explore and transform your data more quickly, Microsoft Fabric offers the easy-to-use **Data Wrangler**.

After launching the Data Wrangler, you'll get a descriptive overview of the data you're working with. You can view the summary statistics of your data to find any issues like missing values.

To clean your data, you can choose any of the built-in data-cleaning operations. When you select an operation, a preview of the result and the associated code is automatically generated for you. When you have selected all necessary operations, you can export the transformations to code and execute it on your data.

4. Train and score models with Microsoft Fabric

Train and score models with Microsoft Fabric

Completed

When you've ingested, explored, and preprocessed your data, you can use the data to train a model. Training a model is an iterative process, and you want to be able to track your work.

Microsoft Fabric integrates with MLflow to easily track and log your work, enabling you to review your work at any time to decide what the best approach is to train the final model. When you track your work, your results are easily reproducible.

Any work you want to track, can be tracked as **experiments**.

Understand experiments

Whenever you train a model in a notebook that you want to track, you create an experiment in Microsoft Fabric.

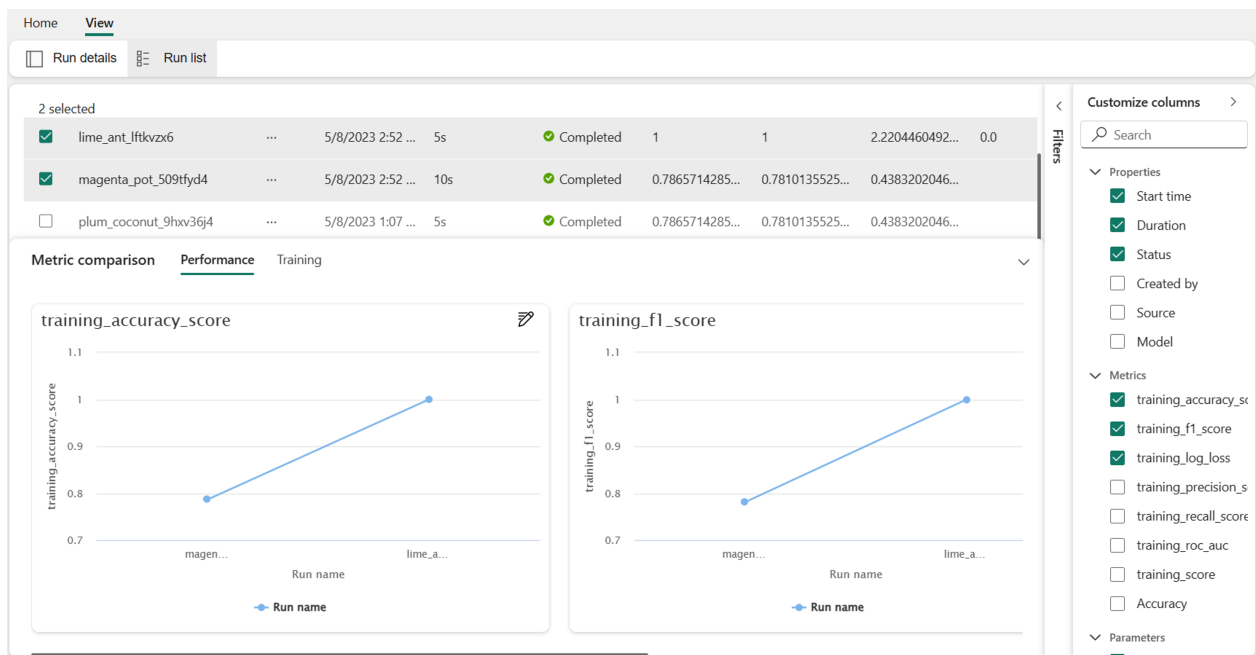
An experiment can consist of multiple runs. Each **run** represents a task you executed in a notebook, like training a machine learning model.

For example, to train a machine learning model for sales forecasting, you can try different training datasets with the same algorithm. Each time you train a model with a different dataset, you create a new experiment run. Then, you can compare the experiment runs to determine the best performing model.

Start tracking metrics

To compare experiment runs, you can track parameters, metrics, and artifacts for each run.

All parameters, metrics, and artifacts you track in an experiment run are shown in the experiments overview. You can view experiment runs individually in the **Run details** tab, or compare across runs with the **Run list**:

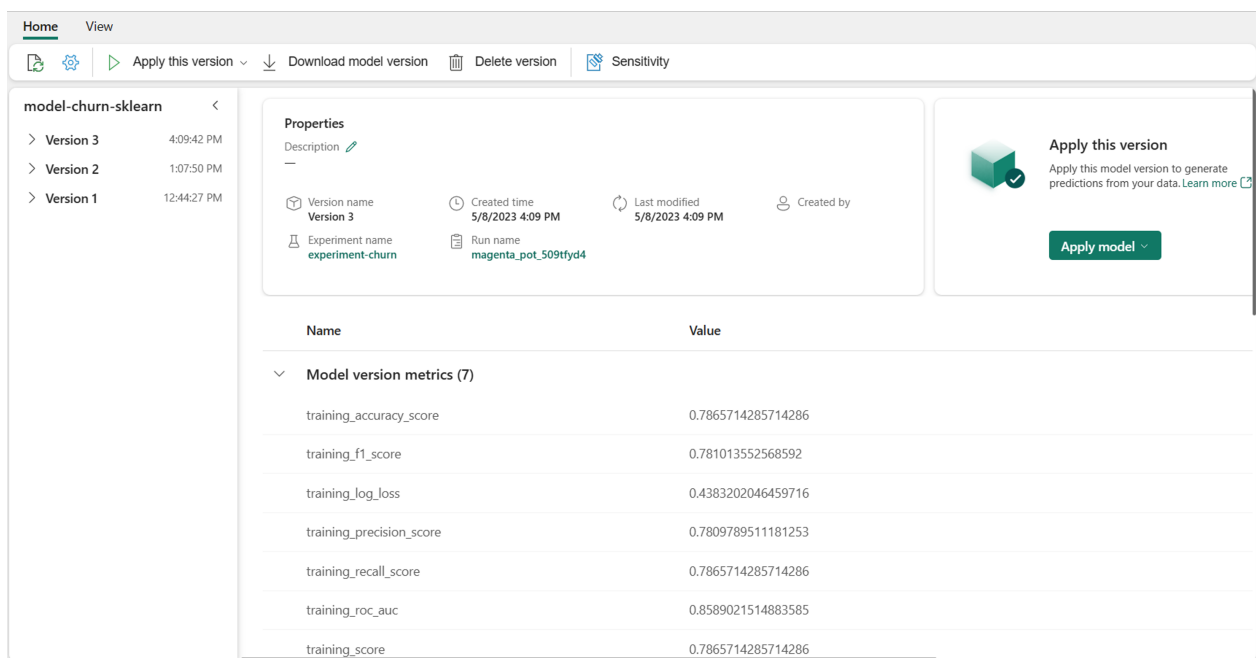


By tracking your work with MLflow, you can compare model training iterations and decide which configuration resulted in the best model for your use case.

Understand models

After you train a model, you want to use it for scoring. With scoring, you use the model on new data to generate predictions or insights. When you train and track a model with MLflow, artifacts are stored within the experiment run to represent your model and its metadata. You can save these artifacts in Microsoft Fabric as a **model**.

By saving your model artifacts as a registered model in Microsoft Fabric, you can easily manage your models. Anytime you train a new model and save it under the same name, you add a new version to the model.



Use a model to generate insights

To use a model for generating predictions, you can use the PREDICT function in Microsoft Fabric. The PREDICT function is built to easily integrate with MLflow models and allows you to use the model for generating batch predictions.

For example, every week you receive sales data from several stores. Based on the historical data, you've trained a model that can predict the sales for the next week, based on the sales of the last few weeks. You tracked the model with MLflow and saved it in Microsoft Fabric. Whenever the new weekly sales data comes in, you use the PREDICT function to let the model generate the forecast for the next week. The forecasted sales data is stored as a table in a lakehouse, which is visualized in a Power BI report for business users to consume.

5. Exercise - Explore data science in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/5-exercise-use-notebook>

Exercise - Explore data science in Microsoft Fabric

Completed

Now it's your chance to explore the data science features in Microsoft Fabric.

Note

You need a Microsoft Fabric trial license with the Fabric preview enabled in your tenant. See [Getting started with Fabric](#) to enable your Fabric trial license.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/6-knowledge-check>

Module assessment

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-data-science-fabric/7-summary>

Summary

Completed

Microsoft Fabric offers one central workspace to perform data science from beginning to end.

To perform data science, you need to first define the problem. Then, you can identify the data you need and ingest it into Microsoft Fabric. When you've ingested your data, you can explore and prepare your data, using notebooks or the Data Wrangler.

To train machine learning models as part of your data science project, you can track your work with experiments. To use a model to generate insights, you can use the built-in PREDICT function.

Get started with SQL Database in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/1-introduction>

Introduction

Completed

SQL Database in Microsoft Fabric is a powerful and versatile solution designed to integrate seamlessly with the broader Microsoft ecosystem. It allows users to use the capabilities of SQL databases within the Microsoft Fabric environment, providing a unified platform for managing and analyzing data. This integration enables organizations to streamline their data workflows, enhance collaboration, and drive more informed decision-making processes.

Moreover, SQL Database in Microsoft Fabric supports a wide range of use cases, from operational and transactional workloads to advanced analytics. This flexibility makes it an ideal choice for organizations looking to consolidate their data infrastructure and use the full potential of their data assets. With features like seamless integration with other Microsoft services, enhanced security, and scalability, SQL Database in Microsoft Fabric empowers users to build and deploy data-driven applications with ease.

The **Databases** workload in Fabric is designed to address many aspects of operational SQL databases, enabling data engineers, analysts, and data scientists to work together to create and query data that is optimized for their specific needs.

In this module, you'll learn about SQL database in Fabric, and how you can manage and query your data efficiently. You'll explain how SQL database in Fabric works, understand security options available, explain how to use Copilot to enhance troubleshooting, and explore mirroring and data virtualization capabilities.

2. Work with SQL databases

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/2-work-with-sql-database>

Work with SQL databases

Completed

SQL Database in Microsoft Fabric is a versatile and developer-friendly transactional database built on the foundation of Azure SQL Database. It allows for the creation and management of operational databases within the Fabric environment.

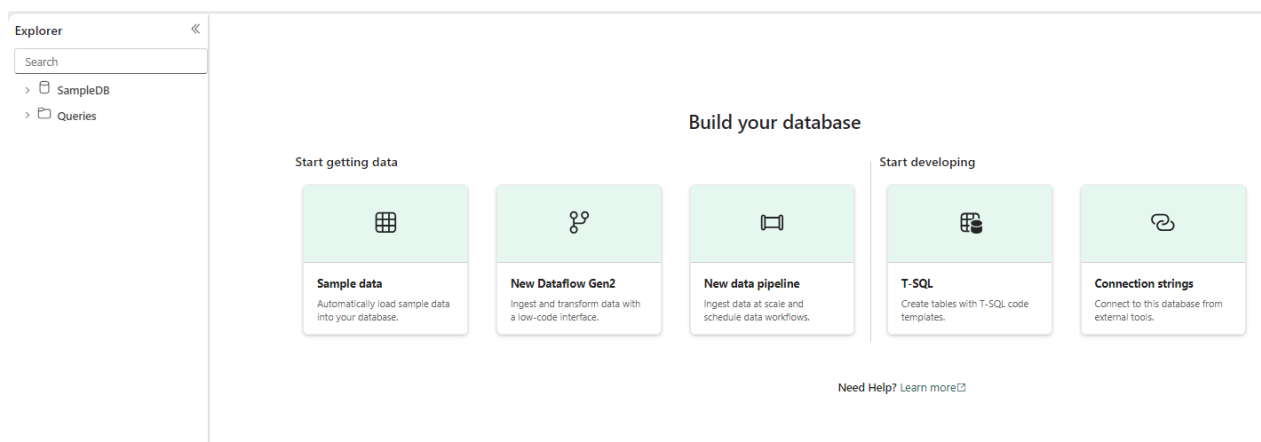
Differently than Azure SQL Database, which is a Platform as a Service (PaaS), SQL Database in Microsoft Fabric is a Software as a Service (SaaS). This means that users can enjoy a low-maintenance solution, allowing them to focus even more on their core business activities.

One of its capabilities is the automatic replication of data into OneLake and conversion to Parquet in near real-time, which facilitates analytics without the need for complex ETL processes. This integration ensures that data is always up-to-date and accessible for various services within Fabric, such as Spark for analytics, notebooks for data engineering, and Power BI for visualization.

Create a SQL Database

To create a new SQL database in Fabric, you need a new or existing workspace. Start by navigating to the Fabric portal and selecting **Databases**. Under the **New** section, select the SQL database tile. Enter a name for your new database and select **Create**.

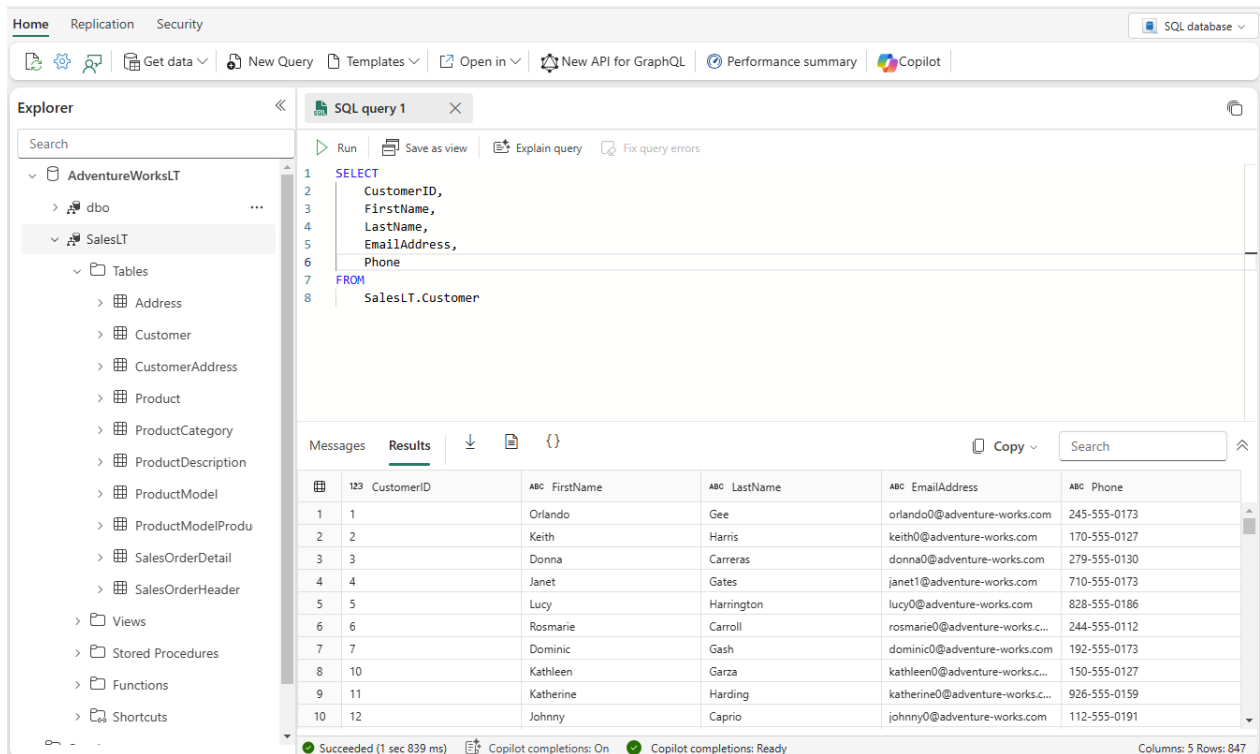
Once the database is provisioned, you see the **Explorer** pane on the **Home** page displaying the database objects.



To help you get started, there are three useful tiles under **Build your database**. The **Sample data** option allows you to import the *AdventureWorksLT* sample data into your empty database. The **T-SQL** option provides a web-editor for writing T-SQL to create database objects like schemas, tables, and views. The **Connection strings** option displays the SQL database connection string needed for connecting with SQL Server Management Studio or other external tools.

Query a SQL Database

You can query a SQL database in Fabric using similar tools available for Azure SQL Database, with the added convenience of a web-based editor in the Fabric portal. This provides an end-to-end, integrated product that simplifies analytics and fosters collaboration.



The **Open in** option allows you to launch Visual Studio Code and SQL Server Management Studio (SSMS) with the connection properties prefilled, making it easier to connect and start working immediately.

Source control

Source control is an essential aspect of managing SQL databases in Microsoft Fabric. It allows you to track changes, collaborate with team members, and maintain a history of modifications. When you integrate your SQL database with a source control system, you ensure that all changes are documented and can be reverted if needed. This practice enhances the reliability and consistency of your database development process.

If you're familiar with source control, you notice that there's no major difference when committing changes to a SQL database in Microsoft Fabric compared to other code repositories.

- **Commit to source control:** You can commit database objects to source control, converting the live database into code. This process reads object definitions from the database and writes them to the repository.
- **Update from source control:** You can update database objects from the contents of source control. The code is validated before applying a differential change to the database.
- **History tracking:** Users can view the history of database objects in the source control system, providing a clear record of changes and facilitating collaboration.

Explore performance capabilities

SQL Database in Fabric offers intelligent performance capabilities like monitoring, and automatic index creation and tuning.

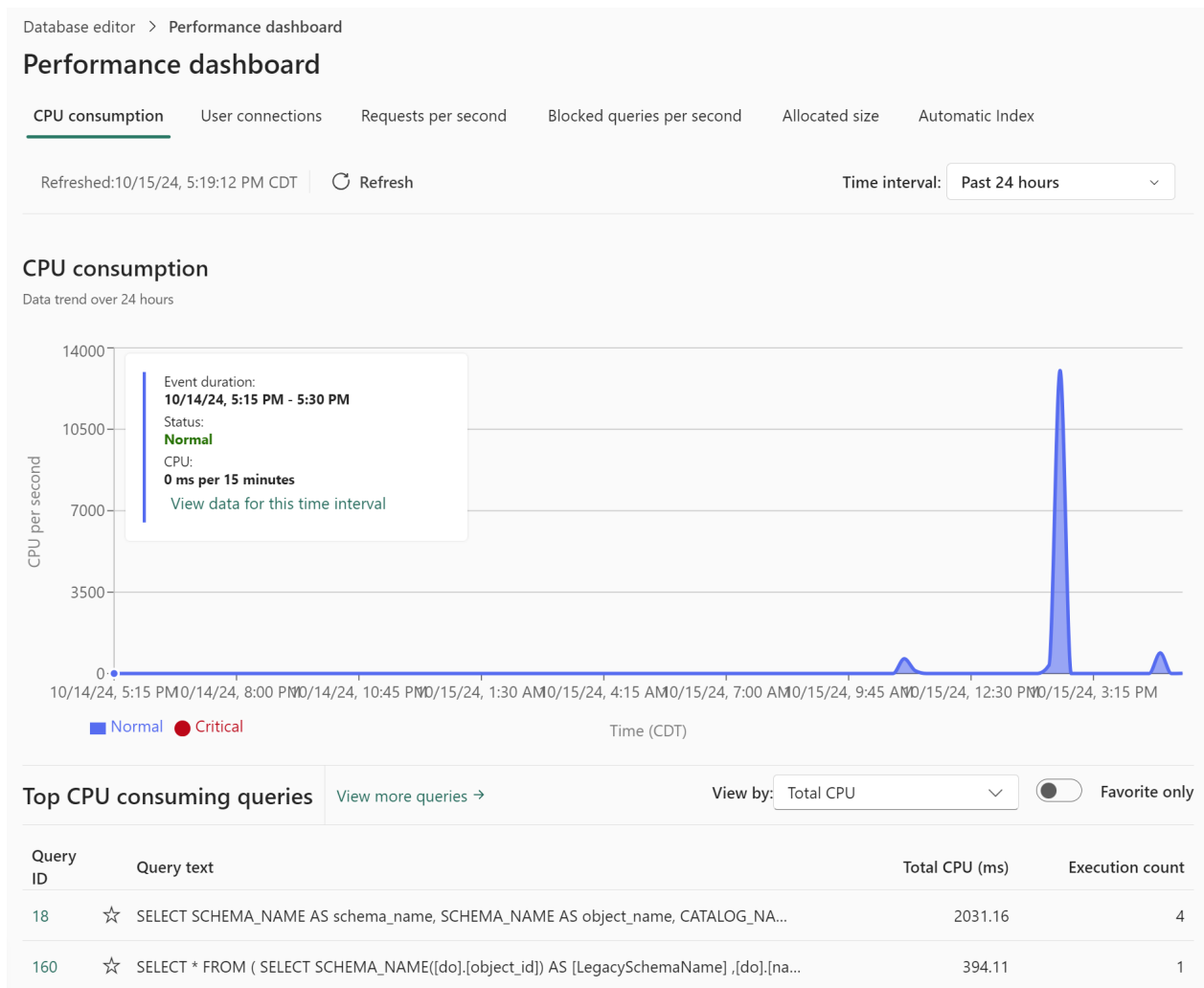
Monitor with performance dashboard

The Performance Dashboard in Fabric SQL Database simplifies the user experience by removing the complexities of monitoring and operation. It allows users to fully use the SQL database engine's capabilities, catering to various workloads in Fabric.

The dashboard offers different levels of metrics visibility to accommodate users with varying SQL expertise. Beginners can access basic query performance metrics, while intermediate and advanced users can view more detailed information.

You can access the performance dashboard by right-clicking on the context button (the three dots) in the item view, then select **Open performance summary**.

Alternatively, you can access the home toolbar in the **Query Editor** window, and select **Performance summary**.



The performance dashboard helps users understand their database performance and receive alerts for any issues. It's designed to assist application developers in detecting and resolving performance bottlenecks early, ensuring an intuitive and efficient user experience.

Explore automatic tuning

Automatic tuning is a built-in capability that applies machine learning to optimize your query performance. It automatically identifies tuning opportunities and implements them to enhance your database's efficiency.

In SQL database in Fabric, indexes are managed dynamically, with a graph showing the count of created, dropped, and reverted indexes over time, and a table listing the indexes created by the tool, including details like schema name, table name, index name, status, key columns, included columns, and creation and drop dates.

You can monitor automatic indexing on the **Automatic index** tab in the performance dashboard.

3. Secure databases in Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/3-secure-database>

Secure databases in Fabric

Completed

In a database, users have access to several capabilities aimed at safeguarding sensitive information. These security measures are capable of securing or masking data from users or roles without proper authorization, ensuring data protection across both the database and the SQL analytics endpoints. This ensures a smooth and secure user experience, with no need for changes to the existing applications.

Understand security capabilities

Users who are often well-versed in the SQL engine and adept at using T-SQL, find databases in Microsoft Fabric easy to use.

This is because SQL Database in Microsoft Fabric is powered by the same SQL engine they're familiar with, enabling them to perform complex queries and data manipulations. The SQL engine's wide range of security features further allows for sophisticated security mechanism at the database level.

Workspaces roles

Designed to provide different levels of access and control within the workspace. You can assign users to the various workspace roles such as **Admin**, **Member**, **Contributor**, and **Viewer**. These roles are crucial for maintaining the security and efficiency of SQL database operations within an organization.

Item permissions

Individual databases can have item permissions assigned directly to them. The main intent behind assigning such permissions is to facilitate the sharing of the SQL database for downstream use. You can review permissions for a SQL database, its SQL analytics endpoint, or its default semantic model by navigating to the item in the workspace and selecting the **Manage permissions** quick action.

Note

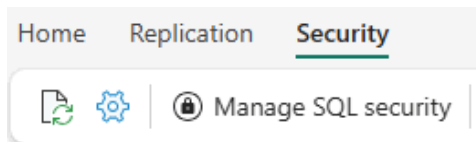
Granting item permissions has no impact on the security metadata inside the database.

Data protection

You can use T-SQL to grant specific permissions to users for more precise control. SQL database supports a range of data protection features that enable administrators to shield sensitive data from unauthorized access.

It includes object-level security for database objects, column-level security for table columns, row-level security for table rows using `WHERE` clause filters, and dynamic data masking to obscure sensitive data like email addresses.

You can manage database-level roles from Fabric portal by selecting **Security** and then choosing **Manage SQL security**.



4. Explore Copilot for SQL Database

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/4-facilitate-database-tasks-with-copilot>

Explore Copilot for SQL Database

Completed

In today's fast-paced tech world, understanding AI is essential for enhancing applications and staying competitive. SQL Database in Fabric is key to this transformation, offering a robust platform for integrating AI. With capabilities like Natural Language to SQL, code completion, quick actions, and intelligent insights, it empowers users to harness AI, and boost application performance.

These tools enable the creation of intelligent, responsive, and efficient applications that meet modern user demands.

Use Copilot for SQL database in Fabric

Microsoft Copilot is integrated with SQL database in Fabric, enhancing SQL management and troubleshooting. It boosts productivity by offering natural language to SQL conversion and self-help for users.

Copilot can automatically correct T-SQL code errors as they occur. By sharing context with the active query tab, Copilot offers helpful suggestions to fix SQL query errors seamlessly. You can also generate T-SQL queries by asking questions in natural language.

With the **natural language to SQL** capability, which translates natural language queries into SQL within the query editor in Fabric portal, database interactions become more intuitive. This integration allows Copilot to answer questions based on your database context, as Copilot is schema-aware.

Which employees have completed more than two projects this quarter?
List all products that were out of stock last month.
Provide the rank of each department by employee satisfaction score and show the department
Create a primary key constraint named PK_Product_ProductID to the column ProductID in the t

Copilot in Fabric SQL database includes several key features:

- **Chat pane:** Use natural language to ask questions and generate T-SQL queries
- **Code completion:** Automatic suggestions while typing T-SQL code
- **Quick actions:** Explain and Fix buttons in the query editor toolbar to help understand and correct queries

To access Copilot features in the Fabric portal, open your SQL database and select the **Copilot** button in the ribbon to open the chat pane, or use the **Explain** and **Fix** quick action buttons in the query editor toolbar.

Note

Copilot for SQL database does not use data in tables to generate T-SQL suggestions.

5. Simplify data management with mirroring and virtualization

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/5-simplify-data-management>

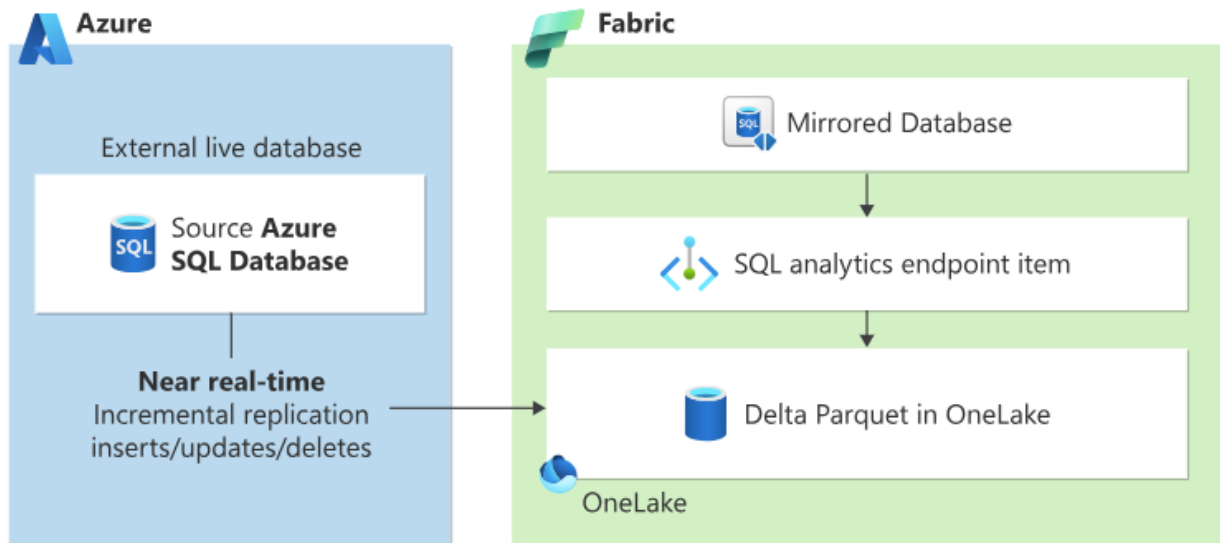
Simplify data management with mirroring and virtualization

Completed

Organizations often face challenges in managing and analyzing their data due to the complexity of integrating various data sources, ensuring data consistency, and maintaining real-time data availability. SQL Database in Microsoft Fabric addresses these challenges by providing a unified platform that simplifies data integration, enhances data consistency, and ensures near real-time data availability.

Integrate with mirroring

One of the key features of SQL Database in Microsoft Fabric is its ability to mirror databases from Azure SQL Database directly into Fabric's OneLake. Additionally, SQL Database in Fabric is automatically mirrored for analytics purposes, with data continuously replicated into OneLake in near real-time.



This mirroring process ensures that data is continuously replicated in near real-time, eliminating the need for complex Extract, Transform, Load (ETL) processes. By doing so, it reduces the total cost of ownership and accelerates the time-to-insight, allowing businesses to unlock business intelligence, artificial intelligence, data engineering, data science, and data sharing scenarios.

After you initiate a mirroring process, you can monitor the replication status by selecting the **Monitor replication** option from the **Replication** tab. If there are no updates in the source tables, the engine will back off and resume regular polling after detecting updated data.

Name	Status	Rows replicated	Last completed
[dbo].[BuildVersion]	Running	0	--
[SalesLT].[ProductDescription]	Running	0	--
[SalesLT].[ProductModelProdu...	Running	0	--
[SalesLT].[Product]	Running	0	--
[SalesLT].[Address]	Running	0	--

To learn more about how to configure mirrored databases, see [Tutorial: Configure Microsoft Fabric mirrored databases from Azure SQL Database](#).

Explore data virtualization

Data virtualization in SQL Database in Fabric is a capability that allows you to access and manipulate data from various sources without the need to physically move or copy the data. This approach provides a unified view of data, enabling seamless integration and analysis across different platforms.

These features enable scenarios such as querying Parquet, CSV, and Delta tables available in a lakehouse.

Capability	Definition	Query Example
Database scoped credential	Allows you to create credentials that can be used to access external data sources securely.	<pre>CREATE DATABASE SCOPED CREDENTIAL MyCredential WITH IDENTITY = 'USER IDENTITY';</pre>
External data source	This enables you to define external data sources, such as files stored in OneLake.	<pre>'abfss://aaaaaaa-0000-1111-2222-bbbbbbbbbbbb@<onelake_account_name>.dfs.fabric.microsoft.com/bbbbbbbb-1111-2222-3333-cccccccccc/Files/parquet/data1.parquet';</pre>
External file format	This capability lets you specify the format of external files, such as Parquet, CSV, and Delta files.	<pre>CREATE EXTERNAL FILE FORMAT MyFileFormat WITH (FORMAT_TYPE = DELIMITEDTEXT, FORMAT_OPTIONS (FIELD_TERMINATOR = ',', STRING_DELIMITER = '"'));</pre>
External table	This allows you to create tables that reference data stored outside the SQL database.	<pre>CREATE EXTERNAL TABLE MyExternalTable (Column1 INT, Column2 NVARCHAR(50)) WITH (LOCATION = 'myfolder/myfile.csv', DATA_SOURCE = MyExternalDataSource, FILE_FORMAT = MyFileFormat);</pre>

6. Exercise: Work with SQL Database in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/6-exercise-query-data-sql-database>

Exercise: Work with SQL Database in Microsoft Fabric

Completed

Now it's your chance to work with SQL queries, manage data, and analyze patterns in a SQL database in Microsoft Fabric.

In this exercise, you create a database in Fabric and load it with sample data. You then query the database and combine data from external data sources. Lastly, you create a database role and secure your data.

Note

To complete this exercise, you need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-sql-database-microsoft-fabric/8-summary>

Summary

Completed

In this module, you have learned about SQL Database in Microsoft Fabric, a Software as a Service (SaaS) transactional database built on Azure SQL Database. The module covered the creation and management of operational databases within the Fabric environment, automatic data replication into OneLake, and conversion to Parquet in near real-time. You also learned about the importance of source control in managing SQL databases in Microsoft Fabric, the platform's intelligent performance capabilities, and the various security measures provided to protect sensitive information. The module also touched on the role of workspace roles and item permissions in controlling access and facilitating sharing.

The main takeaways from this module include understanding how to create a new SQL database in Fabric, query a SQL database using similar tools available for Azure SQL Database, and manage features like object-level, column-level, and row-level security. You also learned about the integration of AI in SQL Database in Fabric, with features like Natural Language to SQL, code completion, quick actions, and intelligent insights. The module also highlighted the role of Microsoft Copilot in enhancing SQL management and troubleshooting. Lastly, you learned about data management simplification through mirroring and data virtualization, and how to secure your data by creating a database role.

Additional reading

- [Microsoft Fabric Documentation](#)
- [SQL database in Microsoft Fabric](#)
- [Overview of Copilot in Fabric](#)
- [What is the admin portal?](#)
- [Share your SQL database and manage permissions](#)
- [What is Mirroring in Fabric?](#)

Design semantic models for scale in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/1-introduction>

Introduction

Completed

Semantic models are the foundation of analytics in Microsoft Fabric. They define how data is structured, calculated, and consumed across reports, dashboards, and AI experiences. A model that works for a small team in Power BI Desktop doesn't automatically serve hundreds of users across multiple data stores. When data volumes grow, teams expand, and consumption patterns shift, the design decisions behind the model need to change.

Suppose an organization is scaling its analytics platform in Microsoft Fabric. Their data lives across lakehouses and warehouses, and their existing semantic models were built in Power BI Desktop for small teams. Now those models need to handle larger datasets, more concurrent users, and broader consumption patterns. The models work at their current size, but they weren't designed for scale.

In this module, you make the design decisions that prepare a semantic model for scale. You start by choosing the right storage mode for how data flows into the model. Then you design star schema relationships for clarity and performance. Next, you design calculations that stay performant and maintainable as data volumes and team size grow. Finally, you configure settings that control how the model handles large datasets, concurrent queries, and external tool access.

By the end of this module, you're able to design semantic models that use the right storage mode, follow star schema best practices, include scalable calculation patterns, and are configured for growing data volumes and consumption demands. Models designed for scale also benefit AI consumption, because AI demands the same things from a model: current data, clear relationships, descriptive structures, and the capacity to handle additional query load.

2. Choose a storage mode

Choose a storage mode

Completed

The first design decision for any semantic model in Microsoft Fabric is how data flows into the model. The storage mode you choose affects query performance, data freshness, and which Fabric features are available. In Fabric, Direct Lake is the default, and for most workloads it's the right choice.

Direct Lake mode

Direct Lake is the default storage mode for semantic models created in Microsoft Fabric. Unlike Import mode, Direct Lake doesn't copy data into the model. Unlike DirectQuery, it doesn't translate queries into source SQL. Instead, Direct Lake reads Delta tables directly from OneLake into memory, which combines the speed of Import with the freshness of DirectQuery.

When a user opens a report backed by a Direct Lake semantic model, the engine loads column data from Delta Parquet files on demand. You don't need to schedule a refresh, like you do with Import mode. When the underlying Delta tables update, the model reflects those changes.

Direct Lake models automatically enable the large semantic model storage format. This setting removes the 10-GB model size limit and is a prerequisite for both query scaleout and XMLA endpoint read/write access. You don't need to enable it manually for Direct Lake models.

Direct Lake connection options

Direct Lake models can connect to data through two paths:

- **OneLake tables:** The model connects directly to Delta tables in a lakehouse or warehouse. This is the simplest path and works well when your data is in a single Fabric data store.
- **SQL analytics endpoint:** The model connects through the SQL endpoint of a lakehouse or warehouse. This path enables access to views, cross-database queries, and security features defined at the SQL layer.

Choose OneLake tables when your data is straightforward and lives in one place. Choose the SQL analytics endpoint when you need views, cross-source joins, or row-level security defined in SQL.

Fallback behavior

Some operations can cause a Direct Lake model to fall back to DirectQuery mode. Complex DAX calculations, queries that exceed available memory, or certain unsupported operations trigger this fallback. When fallback occurs, the query runs against the SQL analytics endpoint rather than reading Delta files directly.

You can configure fallback behavior in the semantic model settings:

- **Allow fallback:** Queries that can't run in Direct Lake mode fall back to DirectQuery automatically. The user gets results, but performance might decrease.
- **Disallow fallback:** Queries that can't run in Direct Lake mode return an error. This option enforces consistent performance but requires that all queries stay within Direct Lake capabilities.

For most production workloads, start with fallback allowed and monitor which queries trigger it. Then optimize those queries or data structures to reduce fallback frequency over time.

Import mode

Import mode copies data into the semantic model and stores it in a compressed, in-memory format. Queries run against the local copy, which makes Import the fastest storage mode for query performance. However, the data is only as current as the last refresh.

Import mode is the right choice when:

- Your data source is outside Fabric (on-premises databases, third-party APIs, flat files).
- Query performance is the top priority and near-real-time freshness isn't required.
- You need features that aren't yet supported in Direct Lake.

Tip

When using Import mode, connect to views instead of raw tables, include only necessary columns, and use appropriate data types to reduce model size. Learn more about [techniques to reduce data loaded into Import models](#).

DirectQuery mode

DirectQuery sends queries directly to the data source at query time. No data is stored in the model, which makes DirectQuery suitable for real-time data scenarios and very large datasets that can't be imported.

The tradeoff is performance. Every report interaction generates a query against the source system. DirectQuery works best when:

- Real-time data is required and even short refresh delays aren't acceptable.
- Source data volumes are too large to import, and the data source is outside Fabric.
- Governance requirements mandate that data stays at the source.

Tip

For more information, see [DirectQuery model guidance](#).

Composite mode

Composite mode combines storage modes within a single model. Some tables use Import, while others use DirectQuery or Direct Lake. This provides flexibility for scenarios where different tables have different performance and freshness needs.

For example, a large fact table might stay in Direct Lake while a small reference table from an external source uses Import. Composite mode also enables many-to-many relationships between tables from different data sources.

Use composite mode when:

- You need data from both Fabric and non-Fabric sources in the same model.
- Some tables require real-time data while others benefit from cached performance.
- You need to combine Direct Lake tables with Import tables for cross-source analysis.

Choose the right storage mode

The following table summarizes when to choose each mode:

Mode	Data location	Query speed	Data freshness	Best for
Direct Lake	OneLake (Delta tables)	Fast	Near real-time	Fabric-native workloads (default)
Import	In-model cache	Fastest	Refresh-dependent	Non-Fabric sources, max performance
DirectQuery	Source system	Depends on source system	Near real-time	Real-time requirements, very large external data
Composite	Mixed	Varies	Mixed	Cross-source scenarios, hybrid requirements

Storage mode also affects AI consumption. When Copilot or data agents query a semantic model, they return answers based on whatever data the model currently reflects. Direct Lake's near-real-time freshness means AI queries return current results without waiting for a scheduled refresh. For models that serve both human users and AI, storage mode choice directly affects the quality of both experiences.

In Fabric, start with Direct Lake. Move to another mode only when your specific scenario requires it.

3. Design star schema for semantic models

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/3-implement-star-schema>

Design star schema for semantic models

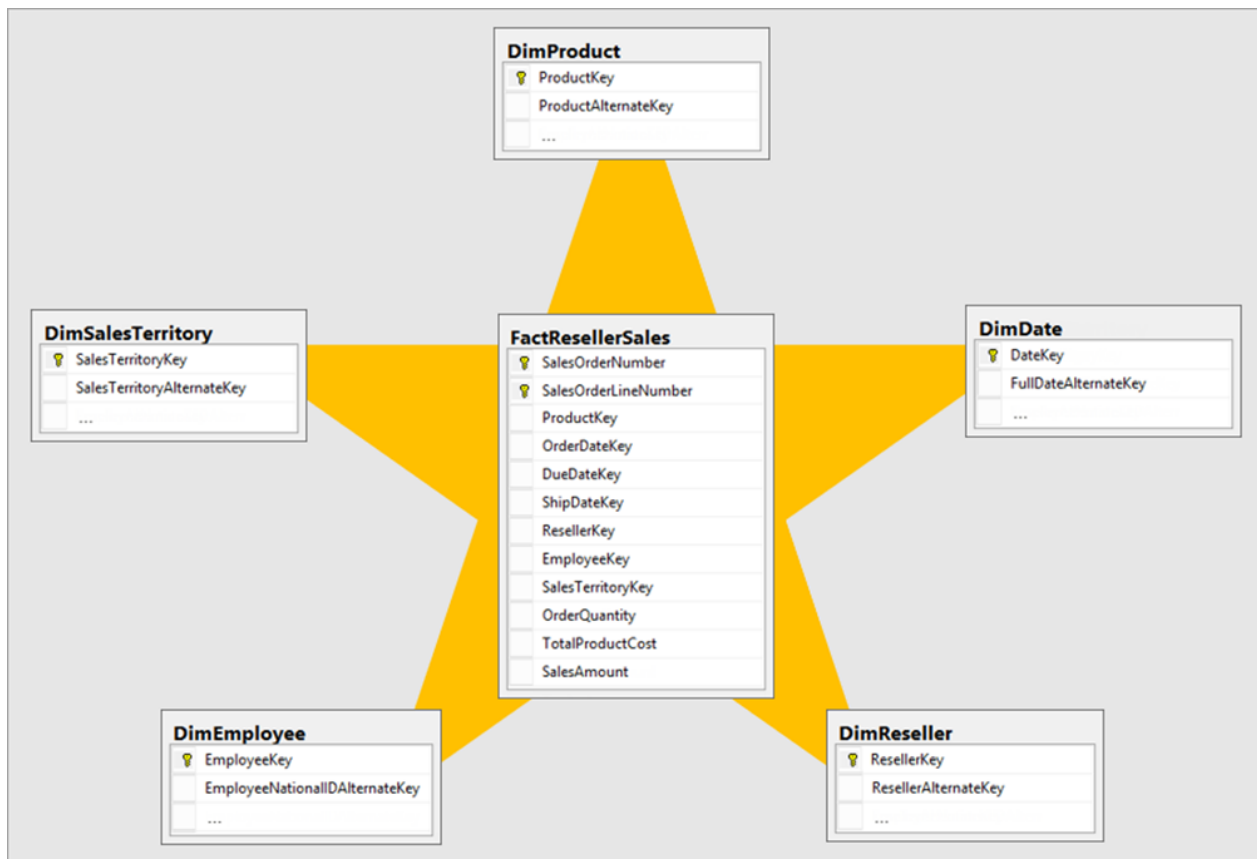
Completed

You chose how data flows into your semantic model. Now design the star schema that organizes it for clear, performant queries. A star schema connects fact tables to dimension tables through relationships, creating the filter paths that reports and AI consumption depend on. If you're familiar with building star schema in Power BI Desktop, this unit focuses on the relationship design decisions that matter as models grow in complexity and scale.

Star schema in a semantic model

In a star schema, fact tables store measurable business events (such as sales transactions, order lines, and web visits), and dimension tables provide the descriptive context (such as product details, customer information,

and date attributes). Dimension tables filter fact tables through relationships, which allows users to slice metrics by any descriptive attribute.



In a Fabric semantic model, this pattern provides clean filter propagation for both reports and AI consumption. When Copilot or a data agent generates a natural language query, a well-organized star schema gives the AI clear paths to the right data. Ambiguous or circular relationships confuse both report consumers and AI tools.

How storage mode affects relationships

Relationships in a semantic model behave differently depending on the storage mode. Understanding these differences is essential for designing star schema that performs well across different scenarios.

Direct Lake relationships

In Direct Lake mode, the engine reads relationships directly from the Delta table metadata. Relationships perform best when:

- Dimension key columns have low cardinality relative to fact table rows.
- Referential integrity is maintained in the source data. When referential integrity holds, the engine uses INNER joins instead of LEFT OUTER joins, which improves query performance.
- Columns used in relationships are indexed in the underlying Delta tables.

Note

If a query involves a relationship that causes the model to exceed memory limits or use unsupported operations, Direct Lake falls back to DirectQuery, and the relationship behavior changes to match DirectQuery semantics.

Cross-source relationships

Fabric semantic models can connect tables from different data stores. A fact table from a lakehouse can have a relationship to a dimension table from a warehouse, or to a table accessed through a SQL analytics endpoint. These cross-source connections use composite model capabilities.

When tables come from different sources, the storage mode for each table determines how the relationship works at query time. The engine resolves each side independently and joins the results.

Relationship types

One-to-many relationships

One-to-many is the most common relationship type in a star schema. One unique value in a dimension table relates to many rows in a fact table. For example, one product row in the Product dimension matches thousands of order rows in the Sales fact table.

Configure one-to-many relationships with the filter direction flowing from the dimension (the "one" side) to the fact table (the "many" side). This is the standard star schema filter pattern.

Many-to-many relationships

Many-to-many relationships are required when neither table has unique values for the relationship column. Use a bridge table to resolve these relationships. A bridge table sits between two tables and holds unique combinations of the keys from each side.

For example, if a customer can have multiple accounts and an account can belong to multiple customers, a Customer-Account bridge table resolves the relationship. The bridge table has one-to-many relationships to both the Customer and Account tables.

Filter direction

In most star schema implementations, use single-direction filtering from dimension to fact. This provides predictable filter propagation and avoids ambiguity in query results.

Bi-directional filtering is sometimes necessary for many-to-many relationships or when dimension tables need to be filtered by values in the fact table. Use bi-directional filters sparingly because they can degrade query performance and create unexpected filter behavior in reports.

Referential integrity

The **Assume referential integrity** setting tells the engine to use INNER joins rather than LEFT OUTER joins when querying across a relationship. In Direct Lake and DirectQuery modes, this setting can significantly improve performance because it reduces the number of rows the engine processes.

Enable this setting when you're confident that every foreign key value in the fact table has a matching value in the dimension table. If referential integrity is violated, rows with unmatched keys silently disappear from query results.

Inactive relationships and USERELATIONSHIP

Only one active relationship can exist between two tables at a time. When you need multiple relationship paths (such as an order date and a ship date both relating to the same Date dimension), make one relationship active and the others inactive.

Use the `USERELATIONSHIP` function in DAX to activate an inactive relationship within a calculation:

```
Shipped Amount =  
CALCULATE(  
    SUM(Sales[Amount]),  
    USERELATIONSHIP(Sales[ShipDate], 'Date'[Date])  
)
```

This pattern keeps the model clean while supporting multiple analytical perspectives on the same data.

Handle snowflake schema in semantic models

Source data often arrives in a normalized snowflake schema, where dimension tables are split into multiple related tables. For example, a Product dimension might be separated into Product, Subcategory, and Category tables, each linked through foreign keys.

In a semantic model, you have two options: flatten the snowflake into a star schema, or preserve the normalized structure.

Flatten into star schema

Flattening means combining the normalized dimension tables into a single denormalized dimension table. The Product table would include Subcategory and Category columns directly, eliminating the extra tables and relationships.

Flatten when:

- The combined dimension table is still small relative to the fact table (which is almost always the case for dimensions).
- You want simpler filter paths from dimension to fact. Each filter travels through one relationship instead of a chain.
- AI consumption is a priority. Fewer tables and simpler relationships give Copilot and data agents clearer paths to the right data.

Flatten dimension tables during data preparation in lakehouses or dataflows, before the data reaches the semantic model. Use Power Query merges, SQL joins, or notebook transformations to combine the normalized tables into a single dimension.

Preserve the snowflake structure

In some cases, keeping the normalized structure makes sense:

- The dimension hierarchy has multiple levels, and flattening would create dozens of redundant columns.
- Multiple fact tables share subdimension tables (such as a shared Category table used by both Sales and Inventory facts), and denormalization would create inconsistent copies.
- Row-level security needs to be applied at a specific level in the hierarchy.

When you preserve a snowflake structure, configure the relationships carefully. Each relationship in the chain must use single-direction filtering from the outermost table toward the fact table so that filters propagate correctly. A filter on Category needs to flow through Subcategory, then through Product, and into the fact table.

Note

In most semantic model scenarios, flattening dimensions into a star schema is the better choice. Fewer tables mean fewer relationships, simpler DAX, faster queries, and better AI consumption. Preserve the snowflake structure only when there's a strong reason to keep it.

When to use composite models for cross-source scenarios

Use composite models when your star schema spans multiple Fabric data stores or includes external sources. Common scenarios include:

- Fact tables in a lakehouse with dimension tables maintained in a warehouse.
- Real-time streaming data from an eventhouse combined with historical data in a lakehouse.
- Reference data from an external source (Import) combined with Fabric-native fact tables (Direct Lake).

In these scenarios, configure the storage mode for each table independently and verify that cross-source relationships perform acceptably at your expected data volumes.

4. Design scalable calculations

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/4-calculation-patterns>

Design scalable calculations

Completed

Your model is structured. Now design the calculations that keep it performant and maintainable as your data and team grow. At small scale, a model with duplicated measures and inconsistent naming still works, even if it's not ideal. At scale, it breaks. A model with hundreds of measures needs structural design decisions that prevent duplicated logic, reduce query time on large datasets, and make it possible for new team members to understand and extend the model without introducing errors.

This unit covers three patterns: calculation groups for reducing measure proliferation, DAX readability discipline for team maintainability, and aggregations for query performance on large fact tables.

Calculation groups

Calculation groups are model objects that apply the same calculation pattern across multiple measures. Instead of creating separate measures for each variation, you define the pattern once and apply it dynamically.

The problem calculation groups solve

Consider an organization with 50 base measures (such as Total Sales, Total Cost, Profit, and Units Sold). Each measure needs Year-to-Date, Quarter-to-Date, and Month-to-Date calculations. Without calculation groups, that's $50 \times 3 = 150$ extra measures. Add prior year comparisons and you're looking at 250+ measures to maintain.

With calculation groups, you create one group with calculation items for each time intelligence pattern. Those items apply to any measure in the model automatically.

How calculation groups work

A calculation group contains calculation items, each defining a DAX expression that modifies the current measure using `SELECTEDMEASURE()`. Here's a time intelligence calculation group:

```
// Year-to-Date
CALCULATE(
    SELECTEDMEASURE(),
    DATESYTD('Date'[Date])
)
```

```
// Quarter-to-Date
CALCULATE(
    SELECTEDMEASURE(),
    DATESQTD('Date'[Date])
)
```

```
// Month-to-Date
CALCULATE(
    SELECTEDMEASURE(),
    DATESMTD('Date'[Date])
)
```

When a user adds the calculation group to a visual, they can switch between YTD, QTD, and MTD for any measure (such as Total Sales, Profit, or Units Sold) without separate measures for each combination.

Dynamic format strings

Dynamic format strings change the display format based on the calculation item context. For example, a percentage calculation should display as a percentage, while currency calculations should display as currency, even when applied to the same base measure.

```
// In the format string expression for a YoY % calculation item:
"0.0%"
```

Dynamic format strings reduce the need for separate formatted measures and keep formatting consistent across the model.

Tip

Learn more about how to [create calculation groups in Power BI](#).

When to use calculation groups

Use calculation groups when you have three or more measures that need the same calculation pattern applied. Common use cases include time intelligence (YTD, QTD, MTD), currency conversion, and variance calculations (actual vs. budget).

DAX readability discipline

At scale with a team maintaining 200+ measures, readability is a design decision, not a personal preference. Consistent, readable DAX reduces maintenance errors and makes it easier for new team members to understand the model.

Variables

Variables store intermediate results, improve readability, and prevent the engine from evaluating the same expression multiple times:

```
Profit Margin =  
VAR TotalRevenue = SUM(Sales[Revenue])  
VAR TotalCost = SUM(Sales[Cost])  
VAR ProfitAmount = TotalRevenue - TotalCost  
RETURN  
    DIVIDE(ProfitAmount, TotalRevenue)
```

Without variables, the same `SUM(Sales[Revenue])` expression might appear three times in a complex measure. Variables evaluate the expression once and reuse the result.

Tip

Learn more about [using variables to improve DAX formulas](#).

Naming conventions

Consistent naming is critical when your model has hundreds of measures maintained by multiple people. Establish conventions for:

- **Measure names:** Use clear, descriptive names such as "Total Sales" or "YoY Revenue Growth." Avoid abbreviations that only the original author understands.
- **Variable names:** Use descriptive names that explain the intermediate value (such as `TotalRevenue` rather than `x` or `temp`).
- **Calculation group items:** Name items by what they do, not how they work (such as "Year-to-Date" rather than "DATESYTD Wrapper").

Descriptive naming also matters for AI consumption. When Copilot or a data agent queries your model, it uses measure names and descriptions to determine which calculations to include. A measure named "YoY Revenue

Growth" produces better AI results than "Calc7_v2."

Tip

Copilot in Power BI can help write and explain DAX formulas. When you're working on complex measures, use Copilot to suggest improvements or explain existing logic.

Iterators vs. aggregation functions

Iterator functions (`SUMX` , `AVERAGEX` , `MAXX`) evaluate a row-by-row expression over a table. Aggregation functions (`SUM` , `AVERAGE` , `MAX`) operate on a single column. At large data volumes, the choice matters:

- Use aggregation functions when you're summarizing a single column. They're faster because the engine can use prebuilt data structures.
- Use iterators when the calculation requires a row-level expression (such as `Quantity × UnitPrice` per row).

Note

Iterators process every row, which can affect performance on large fact tables.

Information functions for defensive patterns

Information functions like `ISBLANK` , `HASONEVALUE` , and `ISINSCOPE` create defensive patterns for measures consumed by multiple reports with different filter contexts:

```
Sales per Customer =  
IF(  
    HASONEVALUE(Customer[CustomerID]),  
    DIVIDE(SUM(Sales[Amount]), 1),  
    DIVIDE(SUM(Sales[Amount]), DISTINCTCOUNT(Sales[CustomerID]))  
)
```

These patterns prevent unexpected results when measures are used in contexts the original author didn't anticipate.

Aggregations

Aggregations are summary tables that store precalculated totals at a higher grain than the detail data. Queries hit these tables first, which improves performance on large fact tables. When a query matches an aggregation, the engine returns results from the smaller summary table rather than scanning millions of detail rows.

Aggregations as a design decision

Deciding *when* to add aggregations and *at what granularity* is a design decision. Performance monitoring and tuning are separate operational concerns, but you make the structural choice during model design.

Consider aggregations when:

- Fact tables exceed millions of rows and commonly used queries summarize data at a higher grain (such as monthly totals by region).
- Users experience slow query response times on summary-level visuals.
- Most report interactions don't need row-level detail.

How aggregation behavior differs by storage mode

In Import mode, aggregations are stored as separate hidden tables. The engine automatically routes matching queries to the aggregation table.

In Direct Lake mode, the Delta tables themselves can serve as aggregation sources. Because Direct Lake reads columnar Parquet files, the engine can handle larger data volumes without aggregations in many scenarios. Add aggregations only when query patterns confirm the need.

Tip

Learn more about [user-defined aggregations in Power BI](#).

5. Configure settings for scale

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/5-enterprise-settings>

Configure settings for scale

Completed

The model is designed. Now configure the settings that control how it handles large datasets, concurrent queries, and external tool access. These settings determine whether the model can keep up as data volumes grow and more users and tools consume it.

Large semantic model storage format

The large semantic model storage format changes how the model stores and compresses data. By default, Power BI limits model uploads to 10 GB. With this setting enabled, models can grow beyond that limit on refresh. The maximum size equals the Fabric capacity size or the limit set by the capacity administrator.

Direct Lake models automatically enable this setting, so you don't need to configure it manually for those models. For Import mode models, you need to enable it explicitly.

This setting is also a prerequisite for both XMLA endpoint read/write access and query scaleout. You need to enable it first when using Import or DirectQuery storage modes.

Enable large semantic model storage format when:

- Your data volumes require models that grow beyond the 10-GB upload limit.
- You need XMLA endpoint access for external tools.

- You plan to use query scaleout for high concurrency.
- You plan to use incremental refresh with partitioned tables.

XMLA endpoint read/write access

The XMLA endpoint lets external tools connect to your semantic model. Tools like Tabular Editor, DAX Studio, and ALM Toolkit use this endpoint for development, debugging, and deployment operations that aren't available in the Fabric service interface.

XMLA endpoint read/write access requires the large semantic model storage format as a prerequisite. Once both are enabled, you can:

- Use Tabular Editor for model development and source control integration.
- Use DAX Studio for query analysis and performance tuning.
- Deploy models through CI/CD pipelines using the Analysis Services client libraries.

At scale, these external tools become essential. Manual edits through the service interface don't support the level of model development that large, team-maintained models require.

Tip

Learn more about [XMLA endpoint connectivity](#).

Query scaleout

Query scaleout distributes read queries across read-only replicas of your semantic model. When hundreds of users access the same model simultaneously, a single instance can become a bottleneck. Query scaleout adds replicas that share the query load.

When you enable query scaleout, the read replicas use a separate copy of the model. This copy synchronizes after each refresh. There can be a brief delay between the primary model finishing a refresh and the replicas reflecting the updated data.

Query scaleout requires large semantic model storage format as a prerequisite.

Enable query scaleout when:

- The model serves hundreds of concurrent users.
- Query performance degrades during peak usage periods.
- The model is backed by a Fabric capacity that supports replicas.

Tip

Learn more about [query scaleout for semantic models](#).

Direct Lake fallback configuration

Direct Lake reads Delta tables directly from OneLake into memory. Some queries can cause the model to fall back to DirectQuery mode, which changes performance characteristics. The fallback setting controls how the

model handles these situations:

- **Allow fallback** (default): Queries that can't run in Direct Lake mode fall back to DirectQuery automatically. Users get results, but performance might decrease.
- **Disallow fallback**: Queries that can't run in Direct Lake mode return an error. This enforces consistent performance but requires that all queries stay within Direct Lake capabilities.

For models at scale, start with fallback allowed. Monitor which queries trigger it, then optimize those queries or data structures to reduce fallback frequency. Disallow fallback only when all query patterns stay within Direct Lake limits and you need guaranteed performance consistency.

OneLake integration

OneLake integration makes your semantic model data accessible as Delta tables in OneLake. When enabled, downstream Fabric items like notebooks, pipelines, and other services can read data directly from the semantic model without rebuilding it from source.

This extends the model's reach beyond reports. A semantic model with well-structured star schema and calculation logic becomes a curated data source for the broader analytics platform.

Enable OneLake integration when:

- Data engineers or data scientists need to consume semantic model data in notebooks or other Fabric items.
- You want to use the semantic model as a shared data source across Fabric.
- Downstream consumers need access to curated, business-logic-enriched data without rebuilding it from raw sources.

Note

OneLake integration currently exports Import mode tables only. Direct Lake tables, DirectQuery tables, measures, and calculation group tables can't be exported. If your model uses Direct Lake exclusively, the underlying Delta tables in OneLake are already accessible to other Fabric items directly.

Settings decision framework

The following table summarizes the key settings decisions for scale:

Setting	Default	Enable when
Large semantic model storage format	Off	Data volumes exceed 10 GB, or you need XMLA endpoint access or query scaleout
XMLA endpoint read/write	Read-only	External tools need to modify the model for development or deployment
Query scaleout	Off	High concurrency degrades query performance (requires large semantic model storage format)
Direct Lake fallback	Allowed	Change to disallowed only when all queries stay within Direct Lake limits

Setting	Default	Enable when
OneLake integration	Off	Downstream Fabric items need to consume semantic model data

Tip

These settings address scale and consumption. Other settings such as endorsement, Copilot approval, and data preparation for AI consumption are covered in separate modules. For a complete reference, see [semantic model settings in the Fabric service](#).

6. Exercise: Design a semantic model for scale in Fabric

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/6-exercise>

Exercise: Design a semantic model for scale in Fabric

Completed

In this exercise, you design a semantic model for scale in Microsoft Fabric by completing the following tasks:

- Connect to Fabric data sources using Direct Lake.
- Build a star schema with fact and dimension tables.
- Configure relationships with appropriate filter direction.
- Create a calculation group for time intelligence across multiple measures.
- Configure settings for scale: enable large semantic model storage format, configure XMLA endpoint access, and enable OneLake integration.
- Verify model behavior by confirming Direct Lake mode and checking fallback configuration.

This lab takes approximately **30** minutes to complete.

Note

You need access to a Fabric-enabled workspace to complete this exercise. For information about a trial license, see [Getting started with Fabric](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

7. Module assessment

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/design-semantic-models-scale/8-summary>

Summary

Completed

In this module, you explored what changes when semantic models need to handle larger datasets, more concurrent users, and broader consumption patterns in Microsoft Fabric. The challenge was clear: models built for small teams in Power BI Desktop don't automatically handle what comes with scale.

You learned to make four critical design decisions. First, you chose Direct Lake as the default storage mode and understood when Import, DirectQuery, or composite models are the better choice. Then you designed star schema relationships for clarity and performance, including referential integrity, inactive relationships, and cross-source connections. Next, you designed scalable calculations using calculation groups to reduce measure proliferation, variables and naming conventions to support team maintainability, and aggregations to handle large data volumes. Finally, you configured settings that control how the model handles large datasets, concurrent queries, and external tool access.

Together, these decisions prepare a semantic model for scale. They also prepare it for AI consumption, because AI demands the same things from a model that scale does: current data, clear relationships, descriptive structures, and capacity.

Learn more

- [Direct Lake overview](#)
- [Create calculation groups in Power BI](#)
- [Large semantic model storage format](#)

Understand Microsoft Fabric IQ fundamentals

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/1-introduction>

Introduction

Completed

Imagine you're a data analyst at Lamna Healthcare, responsible for helping clinical operations teams understand patient care patterns across your facility.

Patient records sit in lakehouse tables while vital signs stream continuously from ICU monitoring equipment into an eventhouse. When hospital administrators ask questions like "Which patients in the ICU have elevated vital signs?" or "How many beds are occupied on the surgical floor?", you need to manually join lakehouse tables with eventhouse streams, translate business terms into technical column names, and write complex queries.

Business users can't explore the data themselves—they depend on you to write queries each time they have a question. By the time you deliver answers, clinical conditions may have already changed.

Fabric IQ solves this challenge by letting you define business vocabulary in an ontology, then bind those concepts to your data sources in OneLake. You define concepts such as Patient, Department, and Room with their properties and relationships, creating a semantic layer. Business users can then ask questions in natural language through data agents or visually explore relationships through Graph in Microsoft Fabric—exploring the data themselves without needing you to write queries.

In this module, you'll discover what Fabric IQ is and how it works. You'll explore the components that work together—ontology items, data agents, Graph in Microsoft Fabric, and Power BI semantic models—and learn when to use each one. You'll also see how ontology modeling shifts your approach from use-case-driven thinking to concept-driven thinking, fundamentally changing how teams collaborate around data.

Important

Fabric IQ is currently in [preview](#).

2. Get started with Fabric IQ

Get started with Fabric IQ

Completed

Fabric IQ is a workload in Microsoft Fabric for creating ontologies that define your business vocabulary. It sits alongside other Fabric workloads like Data Engineering, Data Factory, Data Science, Data Warehouse, Real-Time Intelligence, and Power BI. Within the IQ workload, you create **ontology items**—Fabric artifacts that contain your ontology definitions and data bindings.

An ontology is a shared vocabulary of your business. It's made up of the things in your environment (represented as entity types), their facts (represented as properties of entity types), and the ways they connect (represented as relationships).

You can also think of an ontology like a business context layer, containing:

- A catalog of concepts (like Hospital, Patient, Department) with their properties and relationships
- Data bindings to your lakehouse tables and eventhouse streams
- A graphical representation that links related concepts for navigation and analysis
- A query surface that lets you ask questions about concepts (not just tables), supporting federated queries across sources

Instead of requiring data experts to translate business questions into SQL queries, you model data using business concepts that everyone understands.

The ontology provides a single definition of each concept that can be used by data agents and Graph in Microsoft Fabric for querying and visualization.

Where Fabric IQ fits in the data platform

Fabric IQ builds on Microsoft Fabric's unified data platform. Here's how it connects to other capabilities:

Ingest and store: Fabric IQ works with data you already have in lakehouse tables and eventhouse streams. It doesn't move or duplicate your data—it creates a semantic layer that references your existing data sources.

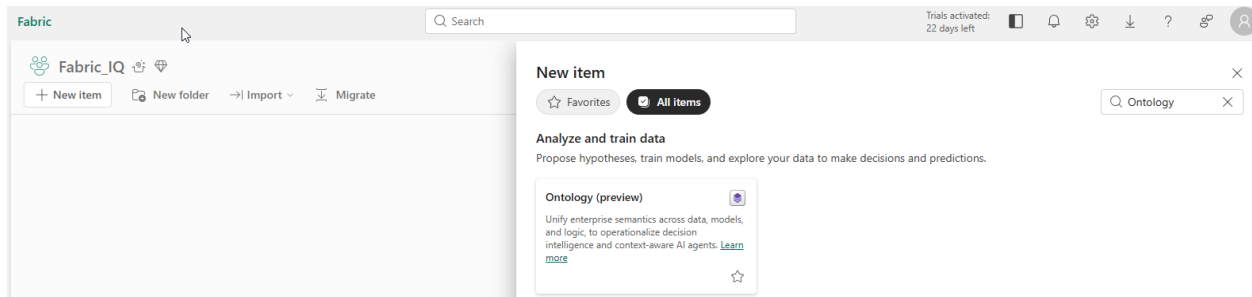
Model and represent semantics: The ontology item offers modeling capabilities by defining entity types, properties, and relationship types. You can generate an ontology structure from existing Power BI semantic models, or create your own from scratch. Then bind these definitions to your data and explore them in a navigable graph that builds automatically.

Analyze and visualize: The ontology item integrates with Graph in Microsoft Fabric to provide a visual graph and query experience based on your business concepts. You can use the ontology to ground data agents with business context.

Access Fabric IQ in your workspace

You create ontology items the same way you create other Fabric items:

1. Navigate to your Fabric workspace
2. Select **+ New item**
3. Search for and select **Ontology (preview)**
4. Enter a name for your ontology (use numbers, letters, and underscores—no spaces or dashes)
5. Select **Create**



The ontology opens when it's ready. You'll see two main areas: the configuration canvas where you define entity types and relationships, and the preview experience where you explore your data.

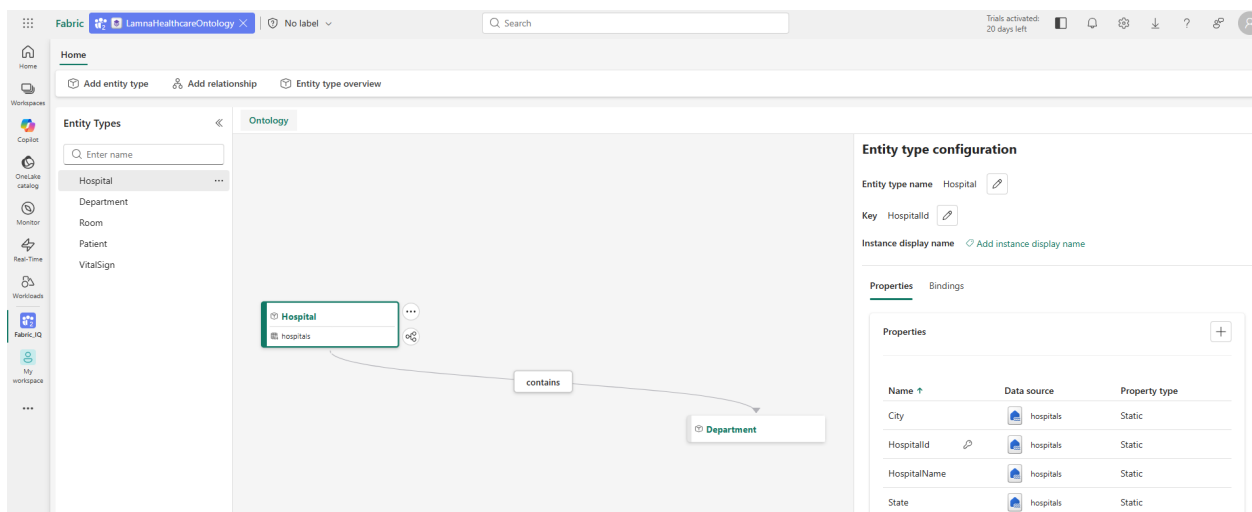
Important

Your Fabric administrator needs to enable certain tenant settings before you can create ontology items. For more information, see [Ontology \(preview\) required tenant settings](#).

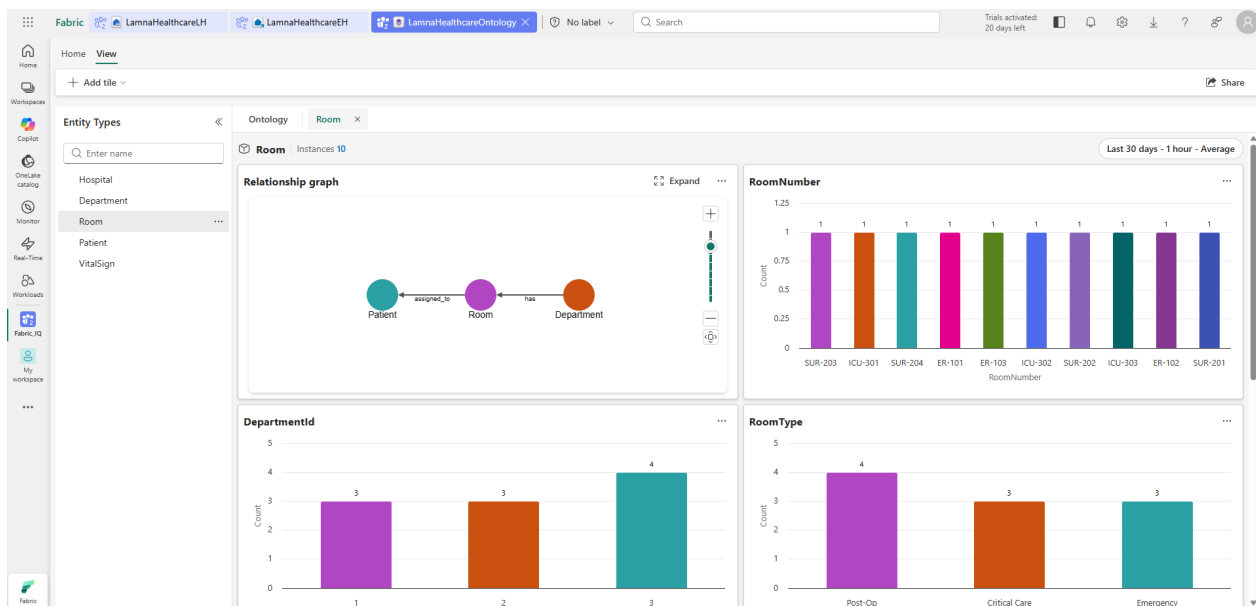
Explore the ontology interface

The ontology item has two primary views:

Configuration canvas: This is where you build your ontology. You create entity types (like Hospital, Department, Room, Patient), define properties on those entities (like FirstName, DateOfBirth, AdmissionDate), and establish relationship types between entities (like "contains" or "assigned to").



Preview experience: This view shows your instantiated ontology. You can see entity instances (specific rooms, departments, and patients), explore relationships in a graph visualization, and query your data using business language instead of SQL. The preview experience integrates with Graph in Microsoft Fabric to provide rich visual exploration.



Understand the build-bind-query workflow

Creating an ontology in Fabric IQ follows three main steps:

Build: Define your business vocabulary by creating entity types, properties, and relationship types. You're modeling the concepts that matter to your business. For example, in a healthcare scenario, this means defining what a Patient is, what properties describe a patient (like name, date of birth, admission date), and how patients relate to other entities (like rooms and departments).

Bind: Connect your ontology definitions to data sources. This includes lakehouse tables for static data (like patient records and room assignments) and eventhouse streams for time-series data (like vital signs monitoring). Data binding maps your source data columns to ontology properties.

Query: Once your ontology is bound to data, you can query it using business concepts instead of database tables. Use Graph in Microsoft Fabric to visualize relationships and traverse connections. Use Query Builder to filter and explore entity instances without writing SQL. Or connect AI agents that can answer natural language questions using your business vocabulary.

This workflow separates business meaning from physical data structures.

How Fabric IQ connects to OneLake

Fabric IQ doesn't move or duplicate your data. Instead, it creates a semantic layer that references existing data sources:

- **Lakehouse tables** contain static data (patient records, hospital information, room assignments)
- **Eventhouse streams** contain time-series data (continuous vital signs from medical monitoring equipment)

When you query your ontology, Fabric IQ automatically sends your queries to the most efficient system to get results quickly. For graph traversals, it uses GQL for Graph in Microsoft Fabric. For time-series queries, it uses KQL for Eventhouse. This federated query capability means you can ask business-level questions that span multiple data sources without knowing the technical details of where data lives.

Two paths to create an ontology

Fabric IQ offers two approaches for creating ontologies:

Generate from Power BI semantic model: If you already have a well-structured Power BI semantic model, you can automatically generate an initial ontology structure from it. Fabric IQ creates entity types matching your tables, properties matching your columns, and relationship types following your model relationships. You then refine this generated ontology by renaming entity types, verifying keys and bindings, and enhancing it with additional data sources like time-series eventhouse streams.

Build from OneLake data: If you don't have a semantic model, or want full control over ontology design, you can build directly from lakehouse and eventhouse data. You manually create entity types, define properties, and establish relationships. This approach gives you complete control over how you model your business vocabulary.

Both paths lead to the same result: a complete ontology that defines your business concepts.

3. Explore Microsoft Fabric IQ components

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/3-explore-fabric-iq-components>

Explore Microsoft Fabric IQ components

Completed

Microsoft Fabric IQ brings together several components that work as an integrated ecosystem. Each component serves a specific role in how you define, query, analyze, and visualize your business data. Understanding these components helps you choose the right tool for each task and leverage their combined strengths.

Fabric IQ includes four core components:

- **Ontology items** define your business vocabulary and bind concepts to data sources
- **Data agents** answer natural language questions across multiple data sources
- **Graph** stores and queries connected data using relationships
- **Semantic models** provide a starting point for generating ontologies from existing Power BI models

Ontology items: Define your business vocabulary

An **ontology item** is where you build your shared business vocabulary. You define the business concepts that matter to your organization, then bind them to actual data sources in OneLake. If you were working with healthcare data, for example, you might bind a Patient concept to a lakehouse table containing patient records, and a VitalSign concept to an eventhouse stream capturing real-time monitoring data.

Consider a scenario where a healthcare data team defines Hospital, Department, Room, Patient, and VitalSign as entity types with properties like hospital ID, department name, and room number. Relationships like

"Department contains Room" and "Patient occupies Room" connect these concepts into a coherent business vocabulary.

The ontology defines your business concepts that can be used by Graph in Microsoft Fabric for visualization and traversal, and by data agents for answering natural language questions.

Data agents: Query data with natural language

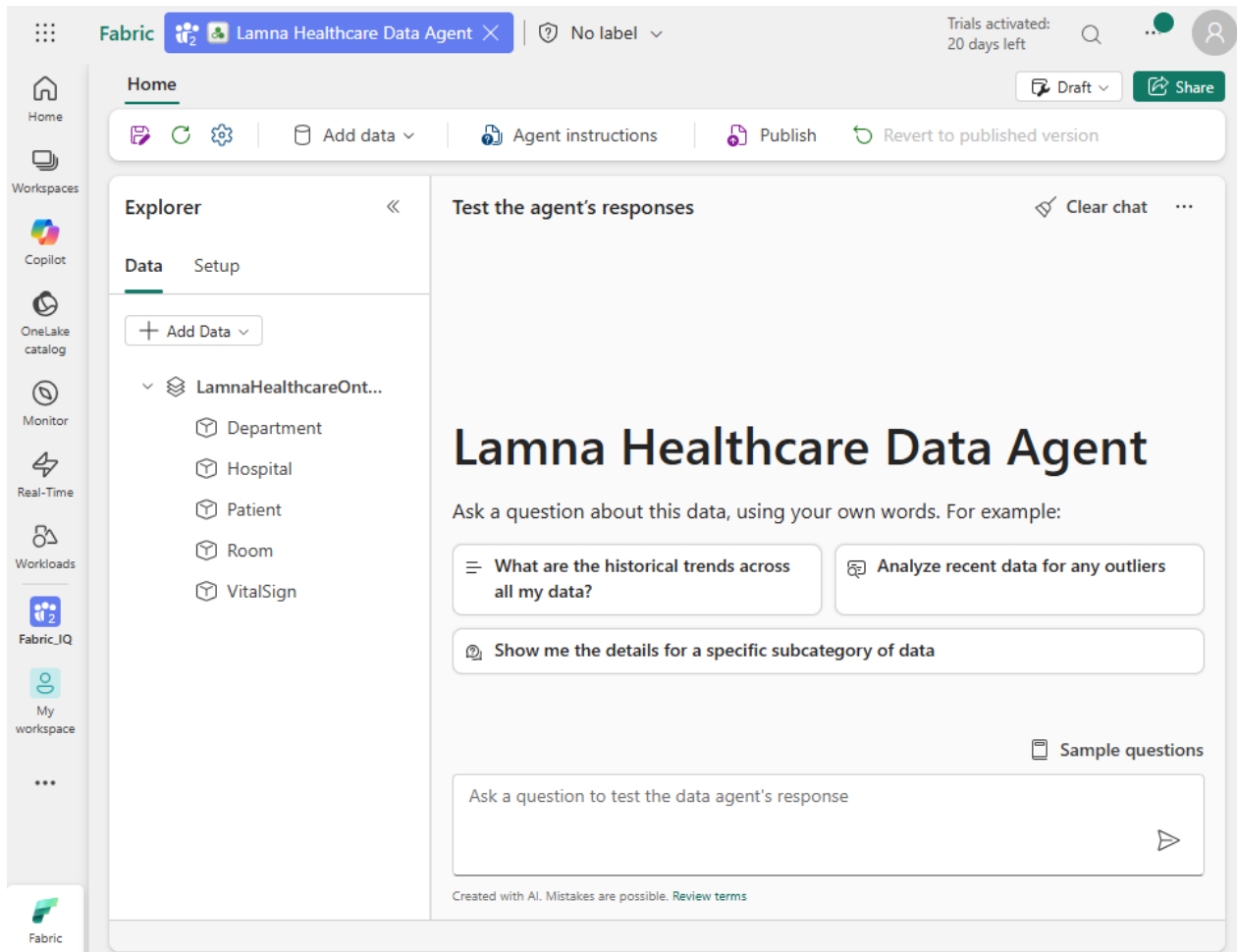
A **Fabric data agent** is a conversational Q&A system powered by generative AI. You configure it with up to five data sources in any combination (lakehouses, warehouses, KQL databases, Power BI semantic models, or ontologies), and it uses Azure OpenAI Assistant APIs to process natural language questions.

When you ask a question, the data agent parses it, identifies the most relevant data source, and generates the appropriate query. For lakehouses and warehouses, it generates SQL. For Power BI semantic models, it generates DAX. For KQL databases, it generates KQL. For ontologies, it uses the defined business vocabulary to understand context.

Consider a scenario where a nurse asks "Which patients in cardiology have elevated heart rates right now?" The data agent understands "cardiology" from the ontology, queries the lakehouse for patient assignments, and queries the eventhouse for current vital sign readings—all without the nurse writing SQL or KQL.

You enhance data agent accuracy by providing **data agent instructions** (guidance on which data source to use for specific question types) and **example queries** (sample question-query pairs that illustrate expected responses). This customization aligns the data agent with your organization's specific terminology and data access patterns.

Data agents enforce read-only access and apply security protocols to ensure users only see data they have permission to access. You can publish data agents to Microsoft 365 Copilot or integrate them with Microsoft Copilot Studio to extend their reach beyond Fabric.



Graph in Microsoft Fabric: Visualize and traverse relationships

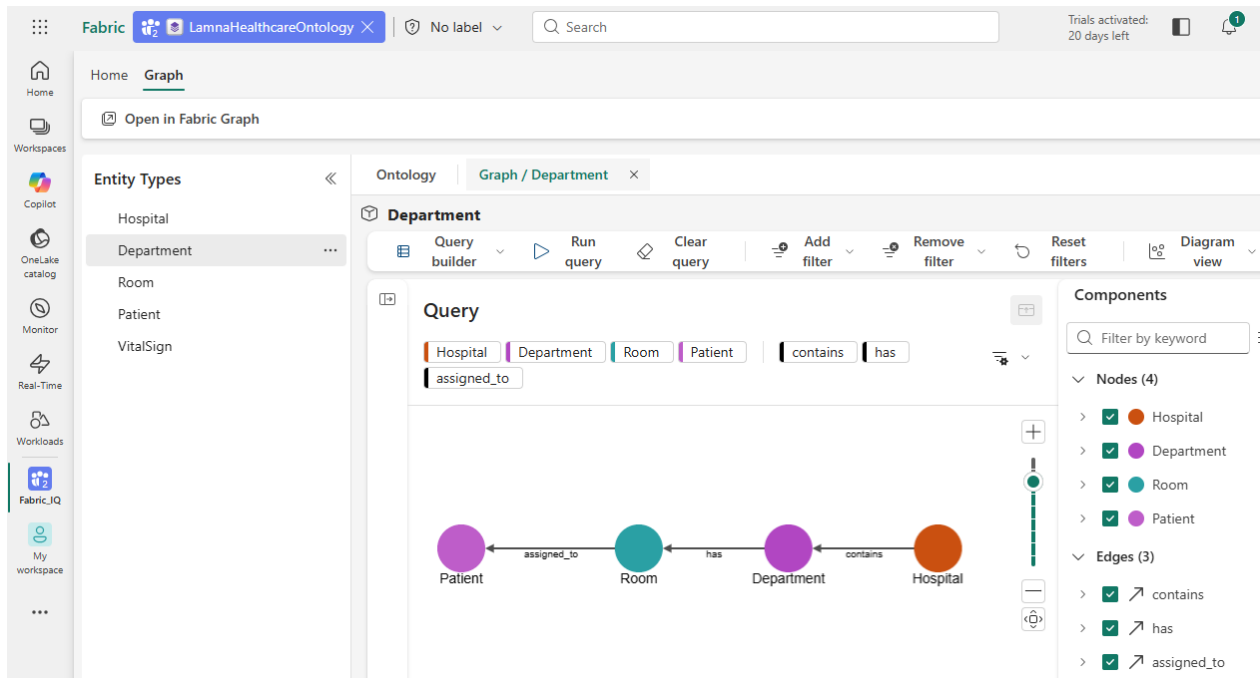
Graph in Microsoft Fabric offers native graph storage and compute for connected data. Unlike relational databases that require complex joins to navigate relationships, it uses a **labeled property graph model** where nodes (entities) and edges (relationships) carry labels and properties that make connections explicit and easy to traverse.

When you create an ontology item, a managed graph is automatically created from your ontology's entity types and relationships. You query it using **GQL (Graph Query Language)**, an international standard for graph queries. Graph excels at answering relationship-heavy questions like "Which patients are assigned to surgical floor rooms?" or "Show me the department hierarchy for a specific hospital."

Consider a scenario where the ontology automatically creates a graph with nodes for hospitals, departments, rooms, and patients, connected by edges representing their relationships. To investigate patient flow patterns, you can traverse from a hospital → its departments → rooms → currently assigned patients.

Graph in Microsoft Fabric operates directly on OneLake without requiring data duplication or ETL. Its scale-out architecture can handle large-scale graph structures with many relationships. You can explore graph data visually through the graph interface or write GQL queries for more complex analysis.

Ontology and Graph in Microsoft Fabric work together seamlessly. The ontology declares your business concepts and relationships, then automatically creates a graph structure. Graph in Microsoft Fabric stores and computes the traversals, enabling visual exploration and advanced queries over your connected data.



Semantic models: Generate ontologies from existing data models

A **Power BI semantic model** provides a structured representation of your data with tables, columns, relationships, and business logic already defined. In the context of Fabric IQ, semantic models serve as an excellent starting point for creating ontologies.

When you generate an ontology from an existing semantic model, Fabric IQ automatically creates:

- **Entity types** matching your semantic model tables (like Hospital, Department, Room, Patient)
- **Properties** matching your table columns (like HospitalId, DepartmentName, Floor)
- **Relationship types** based on your semantic model relationships (like Hospital contains Department, Room assigned to Patient)
- **Keys** identifying unique instances of each entity type

This approach is significantly faster than building an ontology from scratch. Instead of manually creating each entity type and defining every property and relationship, you start with a complete structure that reflects your existing data model.

Consider a scenario where a data team generates an ontology from their existing semantic model with tables for hospitals, departments, rooms, and patients. This automatically creates the entity types and relationships, which they could then enhance by adding time-series vital sign data from eventhouse.

After generating an ontology, you refine it by:

- Verifying that entity type keys are correctly identified
- Confirming that data bindings map to the right source columns
- Adding additional entity types from other data sources (like eventhouse streams)
- Enhancing relationships with binding information that links to actual data

This workflow lets you leverage existing modeling work while extending it with Fabric IQ's cross-domain reasoning and graph capabilities.

4. Understand the ontology modeling paradigm

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/4-understand-ontology-modeling-paradigm>

Understand the ontology modeling paradigm

Completed

Ontology modeling in Fabric IQ defines business concepts independent of specific analytical use cases. This unit explains how the ontology approach differs from traditional analytical data modeling.

Recognize business concepts over table schemas

Traditional analytical data modeling designs tables optimized for specific reporting and analytics needs. You start with questions like "What reports do we need to support? Which dimensions and facts are required?" and design star schemas or dimensional models accordingly.

Ontology modeling inverts this. You start by asking: "What are the core concepts in our business? How do they relate? What facts matter about each concept?" Analytical considerations come after you've captured business meaning.

Consider a scenario where you're working with healthcare data. A traditional approach might create PatientDim, RoomDim, and DepartmentDim tables in a star schema with abbreviated column names like "pt_id," "rm_num," and "dept_id." When different teams build separate data marts independently, they may develop overlapping but inconsistent definitions of "patient" or "department."

With ontology modeling, you define Patient, Room, and Department as entity types using business language: PatientName, DateOfBirth, RoomNumber, DepartmentName. You establish named relationships: Department has Room, Room assigned_to Patient. These definitions become the foundation for AI agents and graph queries that need consistent business context.



This ontology model shows Hospital, Department, Room, and Patient entities connected by named relationships ("contains", "has", "assigned_to") in Fabric IQ's graph interface. Unlike foreign keys that require JOIN statements, these relationships have explicit names and can be queried directly.

Model entity types as reusable concepts

Entity types are conceptual definitions that you create before binding them to data, unlike database tables that combine schema definition with data storage.

An entity type standardizes the name, description, identifiers, and properties for a business concept. By defining what "patient" means at the entity type level—including which properties exist and what the identifier is—you create a definition that can then be bound to a data source. This separates the conceptual model from the underlying table structure.

Define properties with semantic meaning

Properties provide a way to standardize names and data types at the conceptual level, which you then map to actual columns during data binding.

Traditional database columns often vary: "temp," "temperature," "temp_reading," or "body_temp" for the same concept across different tables. Each table defines its own column names independently.

In ontology modeling, you define a standard property name like "Temperature" at the entity type level. When you bind to data, you map this property to whatever column name exists in your table - whether it's "temp," "temperature," or "body_temp." Tools querying the ontology always see the standardized "Temperature" property, regardless of the underlying column name.

Entity Types

«

Hospital

Department

Room

Patient

VitalSign

Ontology

Entity type configuration

Entity type name VitalSign

Key ReadingId

Instance display name Add instance display name

Properties

Bindings

Properties

Name ↑	Data source	Property type
HeartRate	VitalSignsReadings	Timeseries
OxygenSaturation	VitalSignsReadings	Timeseries
PatientId	VitalSignsReadings	Static
ReadingId	VitalSignsReadings	Static
RespiratoryRate	VitalSignsReadings	Timeseries
RoomId	VitalSignsReadings	Static

This VitalSign entity type shows standardized property names like HeartRate, OxygenSaturation, and RespiratoryRate. When bound to data, these properties map to the actual column names in the source table.

Properties can be marked as identifiers to signal which values uniquely identify entity instances. Marking PatientID as an identifier establishes that this value uniquely represents each patient, ensuring consistent entity resolution across data sources.

Establish relationships between concepts

Relationships in ontology modeling are named, directional connections between entity types. Unlike foreign keys that are implicit until you write JOIN statements, these relationships are explicit concepts that tools can query and visualize.

Consider a scenario where a "Department has Room" relationship links hospital departments to their physical spaces. A "Room assigned to Patient" relationship tracks current patient assignments. You can traverse these relationships in the graph interface or query them using GQL to perform dependency analysis—when you need to find all patients in a specific department, you can traverse "Department has Room" followed by "Room assigned to Patient."

Bind concepts to data without duplication

Data binding connects your entity types to actual data sources without copying or moving data. The data stays in place—in lakehouse tables or eventhouse streams in OneLake. The ontology creates a semantic layer that references this data.

Consider a scenario where a Patient entity type binds to a lakehouse table containing demographic information, while the VitalSign entity type binds to an eventhouse stream providing real-time vital signs. Each entity type binds to its appropriate data source using entity-specific keys. When you query across these concepts, the ontology federates queries across the lakehouse and eventhouse, returning integrated results without having moved any data.

✓ Choose data source

● Configure data binding

Configure data binding

Preview data

Binding type

☐ Static
 ☒ Timeseries

Source data timestamp column * ⓘ

Select the column that records when each measurement or event happened. This column must contain date/time values.

Timestamp

Bind your properties

Static * ⓘ

Source column	Property name	
aa ReadingId	ReadingId	
# PatientId	PatientId	
# RoomId	RoomId	
+ Add static property		

Timeseries * ⓘ

Source column	Property name	
# HeartRate	HeartRate	
# OxygenSaturation	OxygenSaturation	
# RespiratoryRate	RespiratoryRate	
+ Add timeseries property		

This data binding configuration shows timeseries properties from an eventhouse mapped to entity type properties. Source columns (HeartRate, OxygenSaturation, RespiratoryRate) bind to standardized property names without copying data—the ontology references the eventhouse data directly.

Data binding creates a semantic layer over your existing data sources without duplication, enabling AI agents to query using business terminology instead of table-specific column names.

5. Module assessment

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/5-knowledge-check>

Module assessment

Completed

6. Summary

Summary

Completed

Microsoft Fabric IQ lets you define business vocabulary once in ontologies, enabling natural language queries and graph visualization of your data. In this module, you learned what Fabric IQ is, how to access it, and how it fits within Microsoft Fabric's data platform.

You explored the build-bind-query workflow for creating ontologies: defining entity types and relationships, binding them to lakehouse tables and eventhouse streams, and querying them through Graph or data agents. You also saw how to generate ontologies from existing Power BI semantic models or build them from scratch using OneLake data.

You examined four components that work together: ontology items define your business vocabulary, data agents answer natural language questions across multiple data sources, Graph in Microsoft Fabric visualizes and analyzes relationships, and Power BI semantic models can generate initial ontology structures.

Finally, you learned how ontology modeling differs from traditional analytical modeling. Instead of starting with specific reporting needs and designing tables optimized for queries, ontology modeling starts with core business concepts and their relationships. This concept-driven approach creates reusable definitions that both data agents and Graph can use, enabling business users to explore data using familiar terminology.