

Domain 2: Implement a data warehouse 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

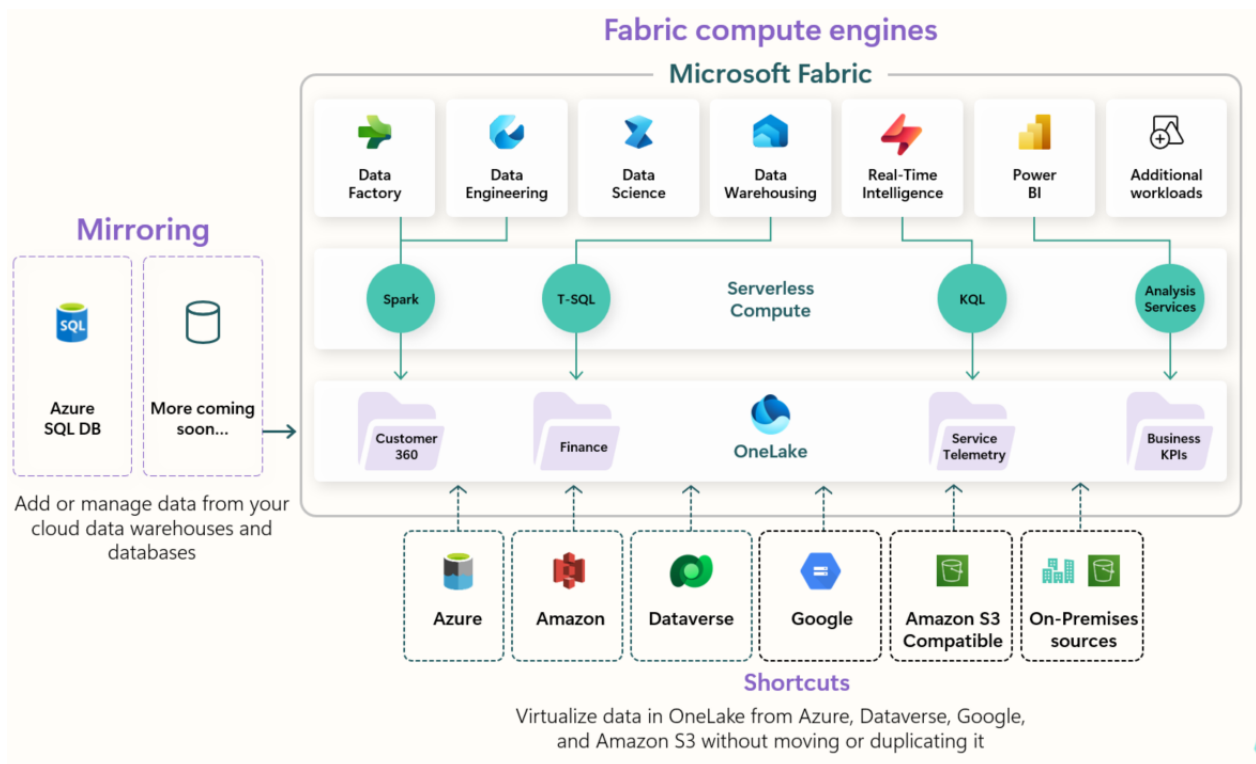
Scalable analytics can be complex, fragmented, and expensive. Microsoft Fabric simplifies analytics solutions by providing a single, easy-to-use product that integrates various tools and services into one platform.

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.

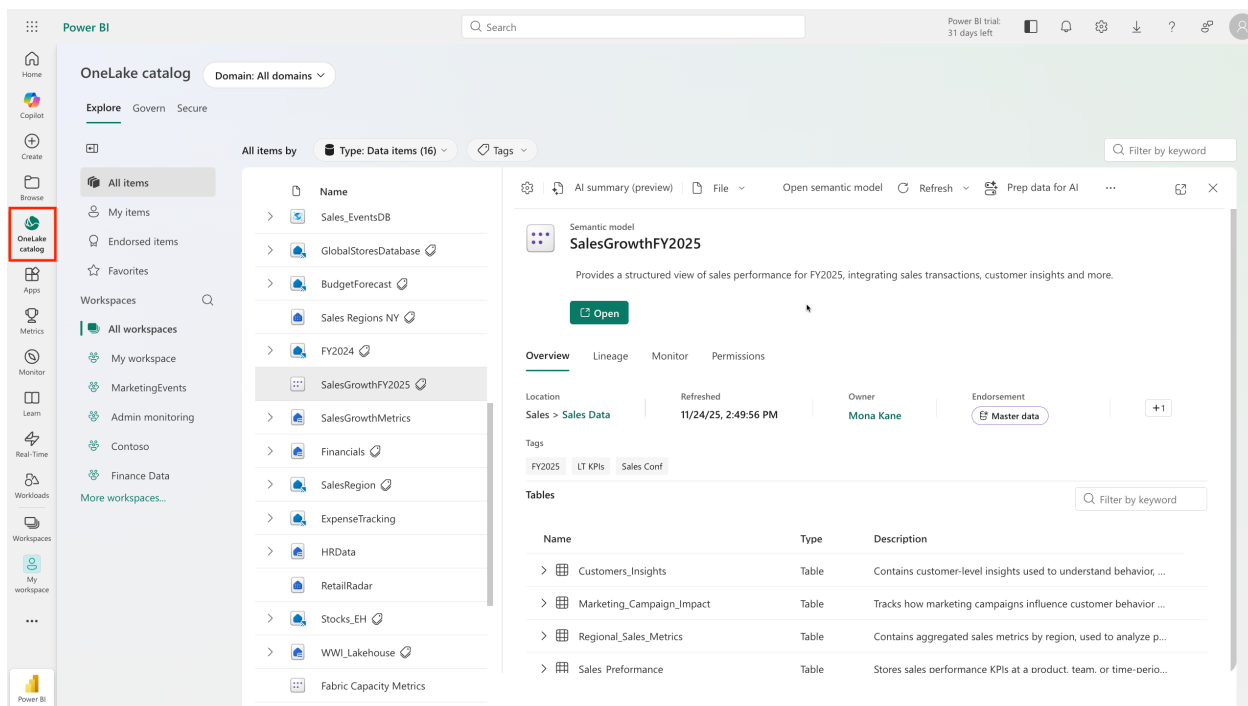
Note

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 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

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.

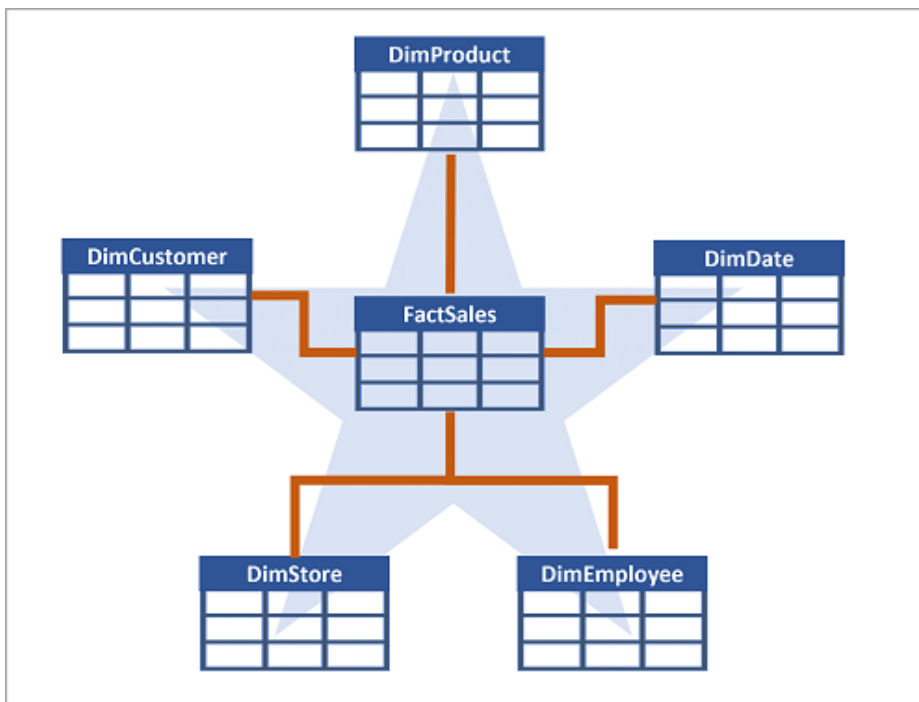
Time dimensions provide information about the time period in which an event occurred. This table enables data analysts to aggregate data over temporal intervals. For example, a time dimension might include columns for the year, quarter, month, and day of a sales order.

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:

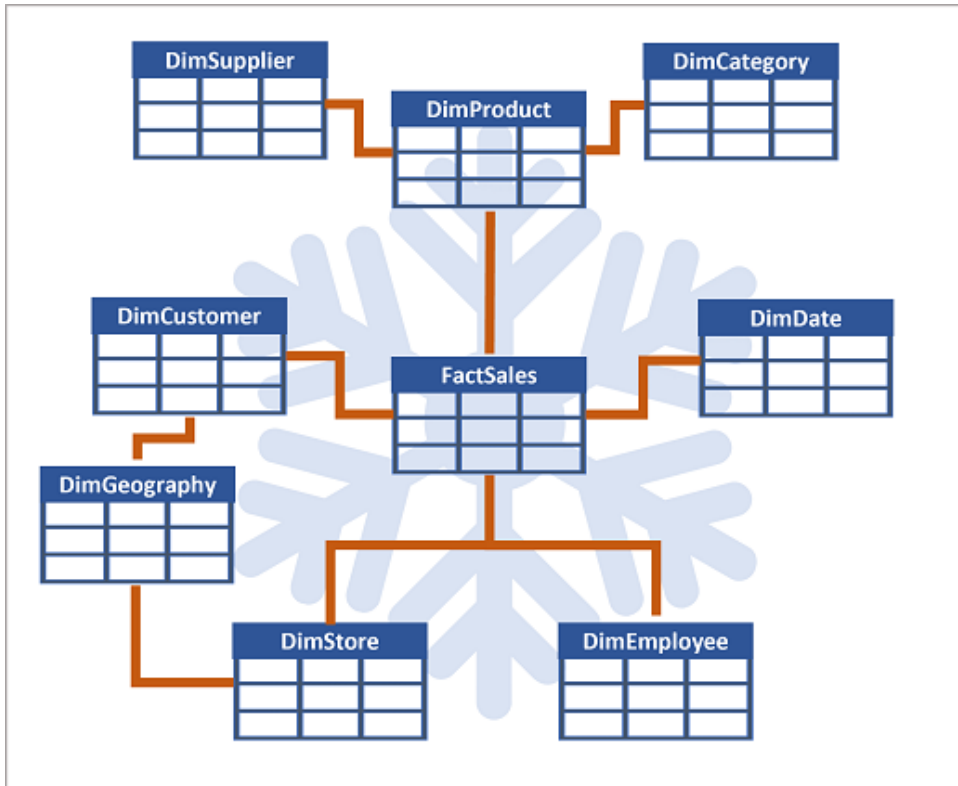


You can use dimension attributes to group fact table numbers at different levels. For example, you could find the total sales revenue for a whole region or just for one customer. You can store the information for each level in the same dimension table.

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

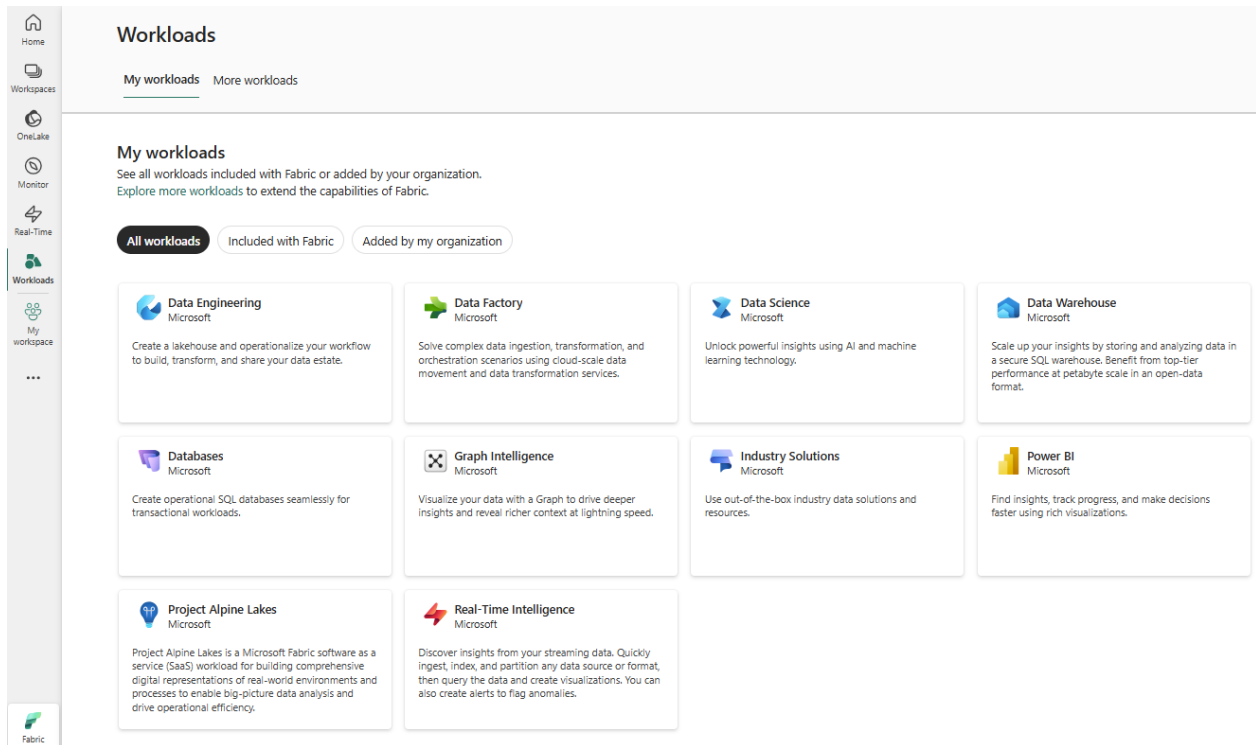
Fabric workspaces can contain two types of SQL-based items that serve different purposes.

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.

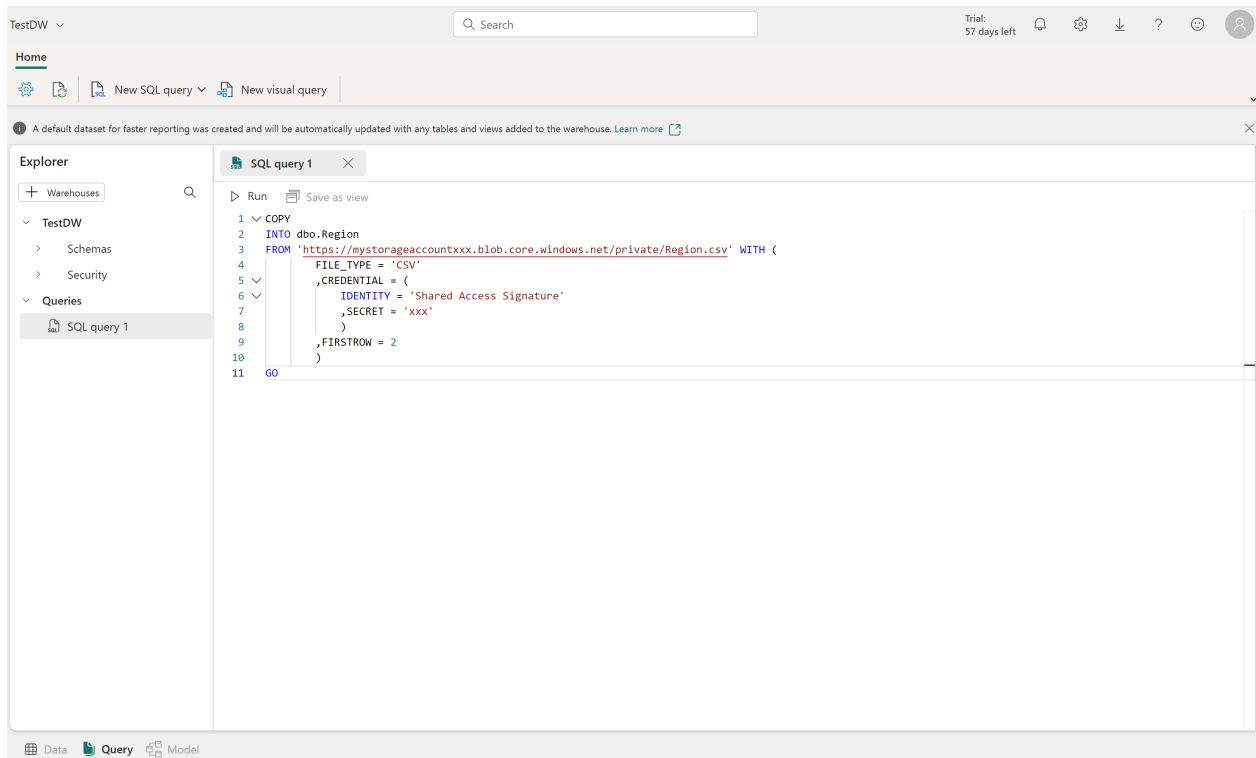
Ingest data into a warehouse

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

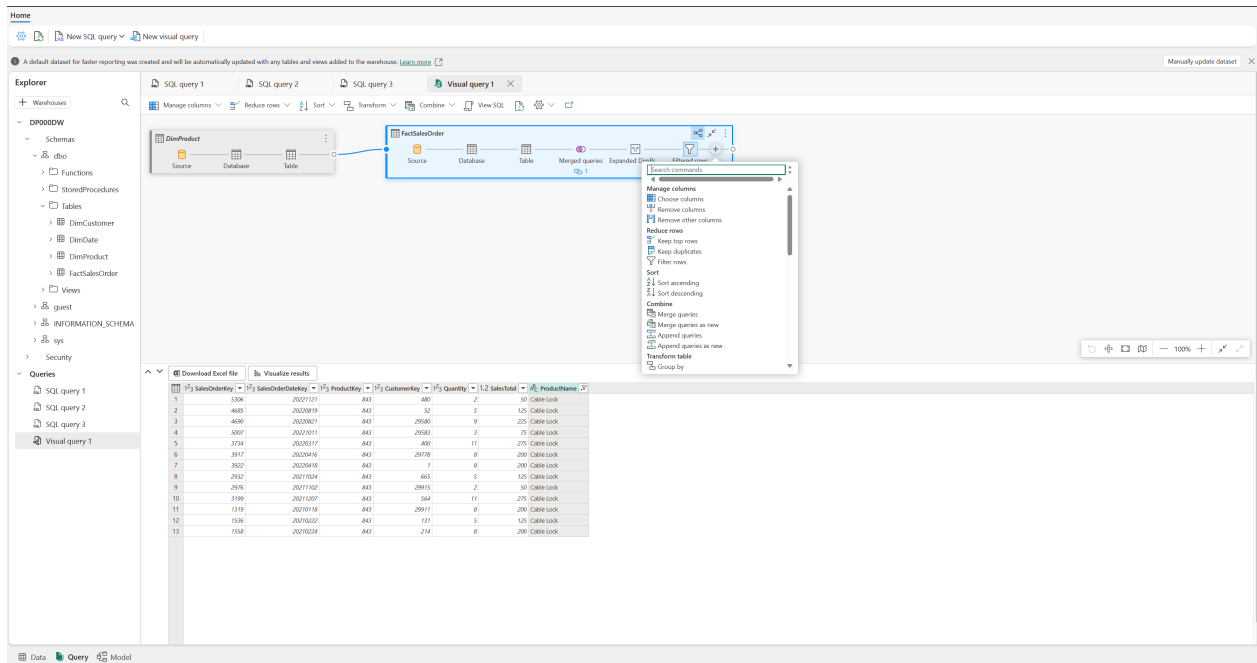
The **SQL query editor** provides a query experience that includes IntelliSense, code completion, syntax highlighting, client-side parsing, and validation. This will feel familiar if you have experience writing T-SQL in SQL Server Management Studio (SSMS) or Azure Data Studio (ADS).

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.

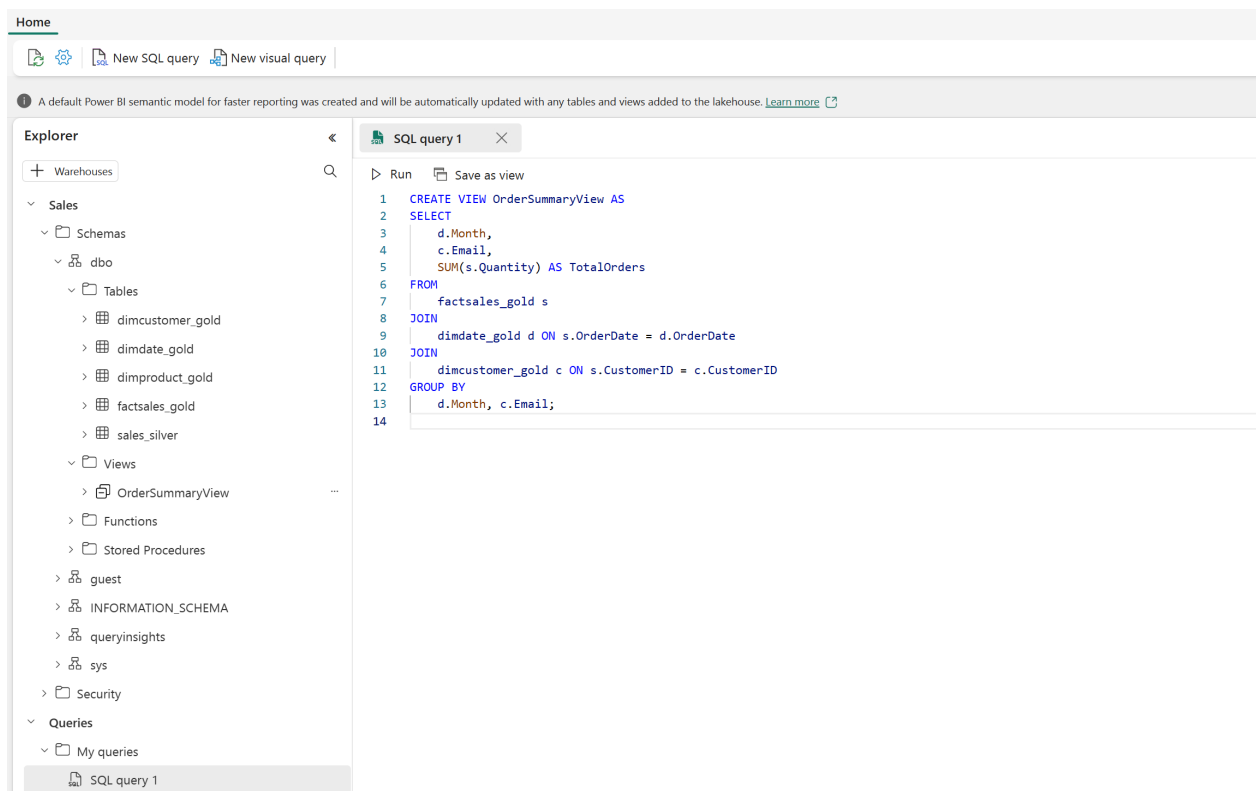
Views define a saved query that you can reference like a table. Use views to standardize how analysts access data, such as combining fact and dimension tables into a reporting-friendly format or filtering rows to a specific business context. For example:

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.

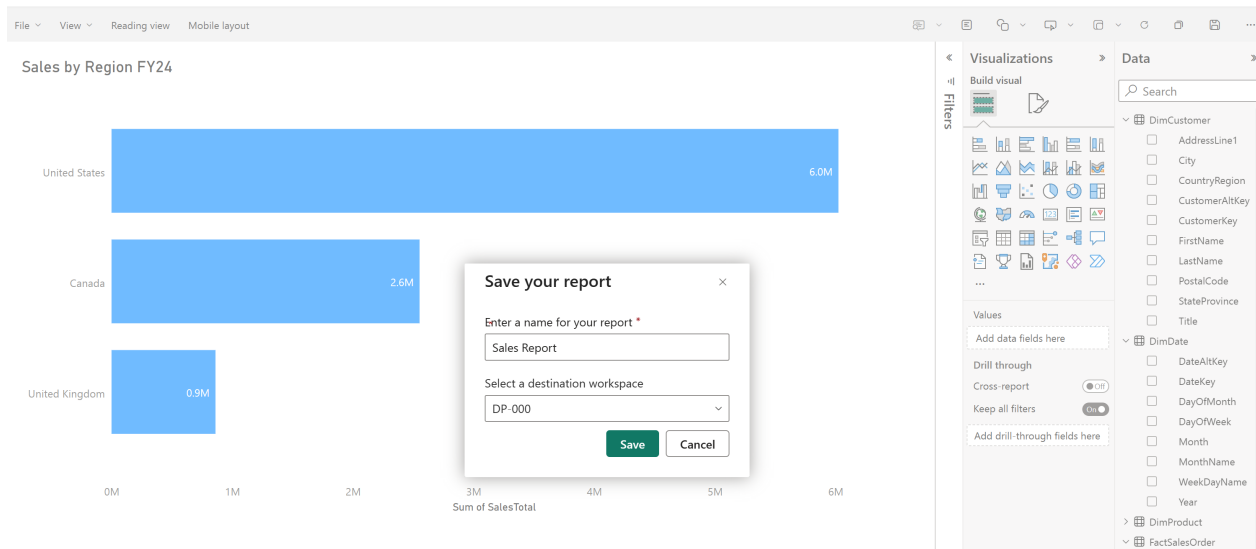
Tip

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

<https://learn.microsoft.com/en-us/training/modules/get-started-data-warehouse/8-knowledge-check>

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](#)

Load data into a Microsoft Fabric data warehouse

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/1-introduction>

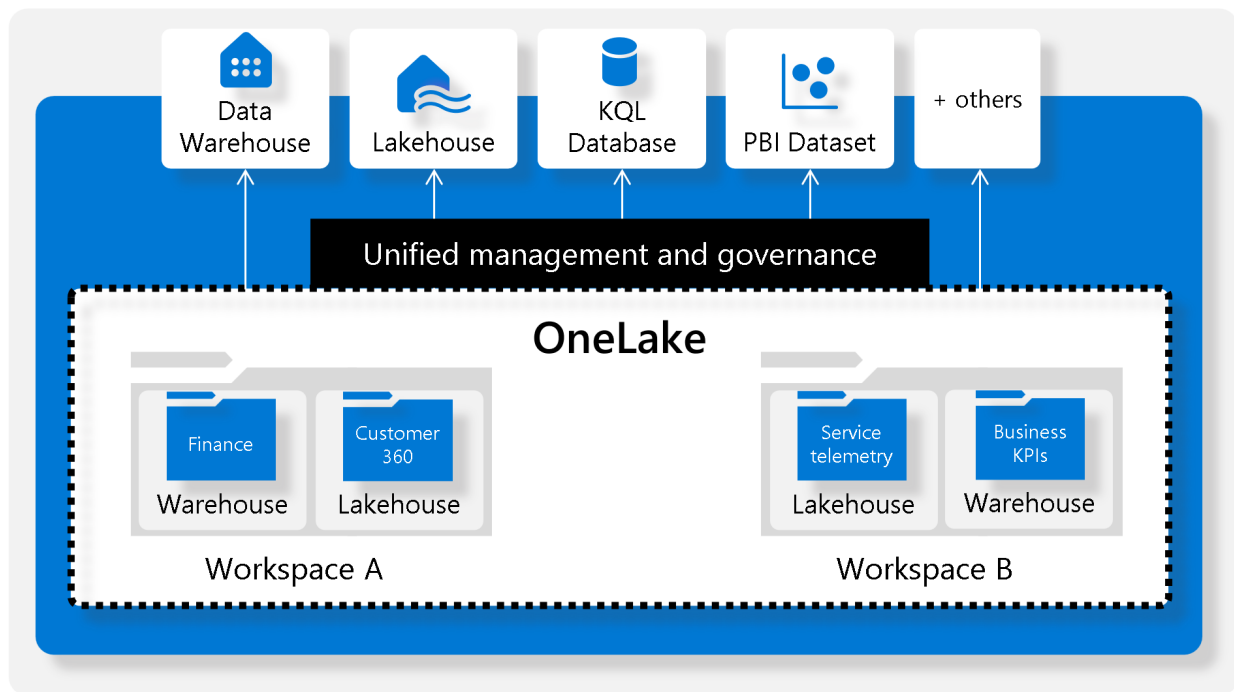
Introduction

Completed

[Microsoft Fabric Data Warehouse](#) is a complete platform for data, analytics, and AI (Artificial Intelligence). It refers to the process of storing, organizing, and managing large volumes of structured and semi-structured data.

Data warehouse in Microsoft Fabric offers a rich set of features that make it easier to manage and analyze data. It includes advanced query processing capabilities, and supports the full transactional T-SQL capabilities like an enterprise data warehouse.

Unlike a dedicated SQL pool in Synapse Analytics, a warehouse in Microsoft Fabric is centered around a single data lake. The data in the Microsoft Fabric warehouse is stored in the Parquet file format. This setup allows users to focus on tasks such as data preparation, analysis, and reporting. It takes advantage of the SQL engine's extensive capabilities, where a unique copy of their data is stored in [Microsoft OneLake](#).



Understand the ETL (Extract, Transform, and Load) process

ETL provides the foundation for data analytics and data warehouse workstreams. Let's review some aspects of data manipulation in an ETL process.

	Description
Data extraction	It involves connecting to the source system and collecting necessary data for analytical processing.
Data transformation	It involves a series of steps performed on the extracted data to convert it into a standard format. Combining data from different tables, cleaning data, deduplicating data, and performing data validations.
Data loading	The extracted and transformed data are loaded into the fact and dimension tables. For an incremental load, this involves periodically applying ongoing changes as per requirement. This process often involves reformatting the data to ensure its quality and compatibility with the data warehouse schema.
Post-load optimizations	Once the data is loaded, certain optimizations can be performed to enhance the performance of the data warehouse.

All these steps in the ETL process can run in parallel depending on the scenario. As soon as some data is ready, it's loaded without waiting for the previous steps to be completed.

In the next units, we'll explore various ways of loading data in a warehouse, and how they can facilitate the tasks of building a data warehouse workload.

2. Explore data load strategies

Explore data load strategies

Completed

In Microsoft Fabric, there are many ways you can choose to load data in a warehouse. This step is fundamental as it ensures that high-quality, transformed, or processed data is integrated into a single repository.

Also, the efficiency of data loading directly impacts the timeliness and accuracy of analytics, making it vital for real-time decision-making processes. Investing time and resources in designing and implementing a robust data loading strategy is essential for the success of the data warehouse project.

Understand data ingestion and data load operations

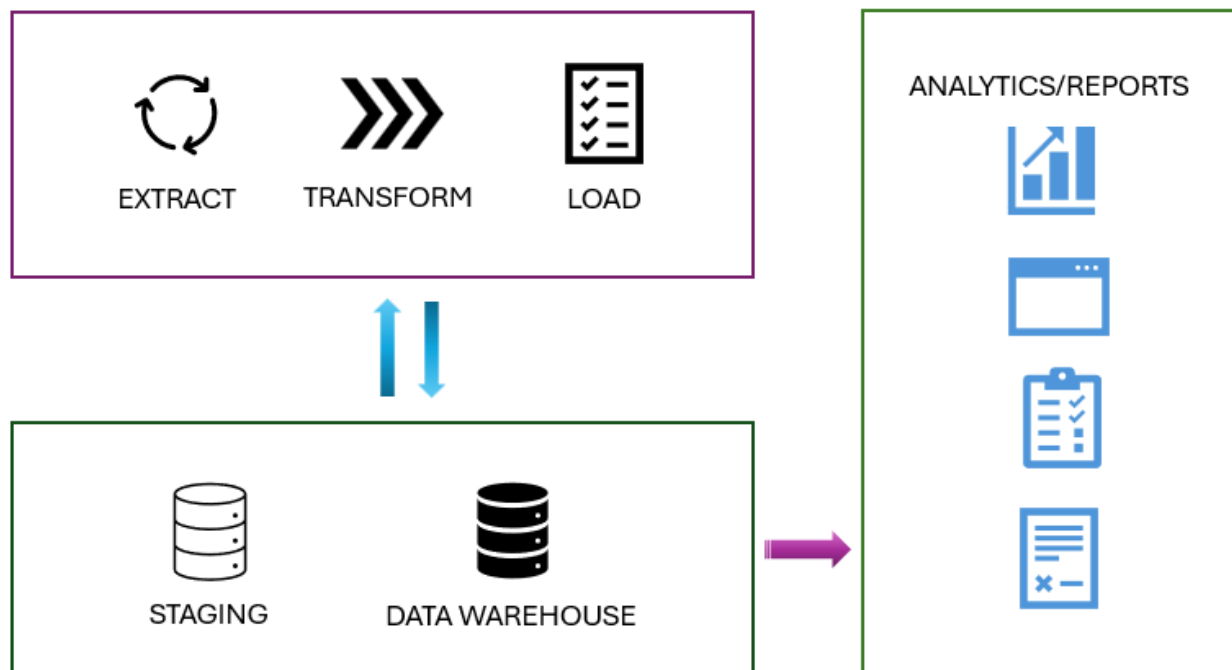
While both processes are part of the ETL (Extract, Transform, Load) pipeline in a data warehouse scenario, they usually serve different purposes. **Data ingestion/extract** is about moving raw data from various sources into a central repository. On the other hand, **data loading** involves taking the transformed or processed data and loading it into the final storage destination for analysis and reporting.

Fabric data warehouses and lakehouses automatically store their data in OneLake using the Delta Parquet format.

Stage your data

You might have to build and work with auxiliary objects involved in a load operation, such as tables, stored procedures, and functions. These auxiliary objects are commonly known as **staging**. Staging objects act as temporary storage and transformation areas. They can either share resources with a data warehouse or reside in their own storage area.

Staging serves as an abstraction layer, simplifying and facilitating the load operation to the final tables in the data warehouse.



Also, staging area provides a buffer that can help to minimize the impact of the load operation on the performance of the data warehouse. This is important in environments where the data warehouse needs to remain operational and responsive during the data loading process.

Review type of data loads

There are two types of data loads to consider when loading a data warehouse.

Load Type	Description	Operation	Duration	Complexity	Best used
Full (initial) load	The process of populating the data warehouse for the first time.	All the tables are truncated and reloaded, and the old data is lost	It may take longer to complete due to the amount of data being handled	Easier to implement as there's no history preserved	This method is typically used when setting up a new data warehouse, or when a complete refresh of the data is required
Incremental load	The process of updating the data warehouse with the changes since the last update	The history is preserved, and tables are updated with new information	Takes less time than the initial load	Implementation is more complex than the initial load	This method is commonly used for regular updates to the data warehouse, such as daily or hourly updates. It requires mechanisms to track changes in the source data since the last load.

An ETL (Extract, Transform, Load) process for a data warehouse doesn't always need both the full load and the incremental load. In some cases, a combination of both methods might be used. The choice between a full load and an incremental load depends on many factors such as the amount of data, the characteristics of the data, and the requirements of the data warehouse.

To learn more about how to perform an incremental load, see [Incremental load](#).

Understand business key and surrogate key

In a data warehouse, both surrogate keys and business keys are essential for effective data warehousing and data integration, but they serve different purposes.

- **Surrogate key:** A surrogate key is a system-generated identifier that is used to uniquely identify a record in a table within the data warehouse. It has no business meaning and is typically an integer or a unique identifier. Surrogate keys are used to maintain consistency and accuracy in the data warehouse, especially when integrating data from multiple sources. They help to avoid issues that can arise from changes in the source systems, such as reusing or changing business keys.
- **Business Key:** A business key, also known as a natural key, is an identifier that comes from the source system and has business meaning. It's used to uniquely identify a record in the source system. Examples of business keys include product codes, customer IDs, and employee numbers. Business keys are important for maintaining traceability between the data warehouse and the source systems. They help to ensure that data in the warehouse can be accurately matched to the corresponding records in the source systems.

Load a dimension table

Think of a dimension table as the "*who, what, where, when, why*" of your data warehouse. It's like the descriptive backdrop that gives context to the raw numbers found in the fact tables.

For example, if you're running an online store, your fact table might contain the raw sales data - how many units of each product were sold. But without a dimension table, you wouldn't know who bought those products, when they were bought, or where the buyer is located.

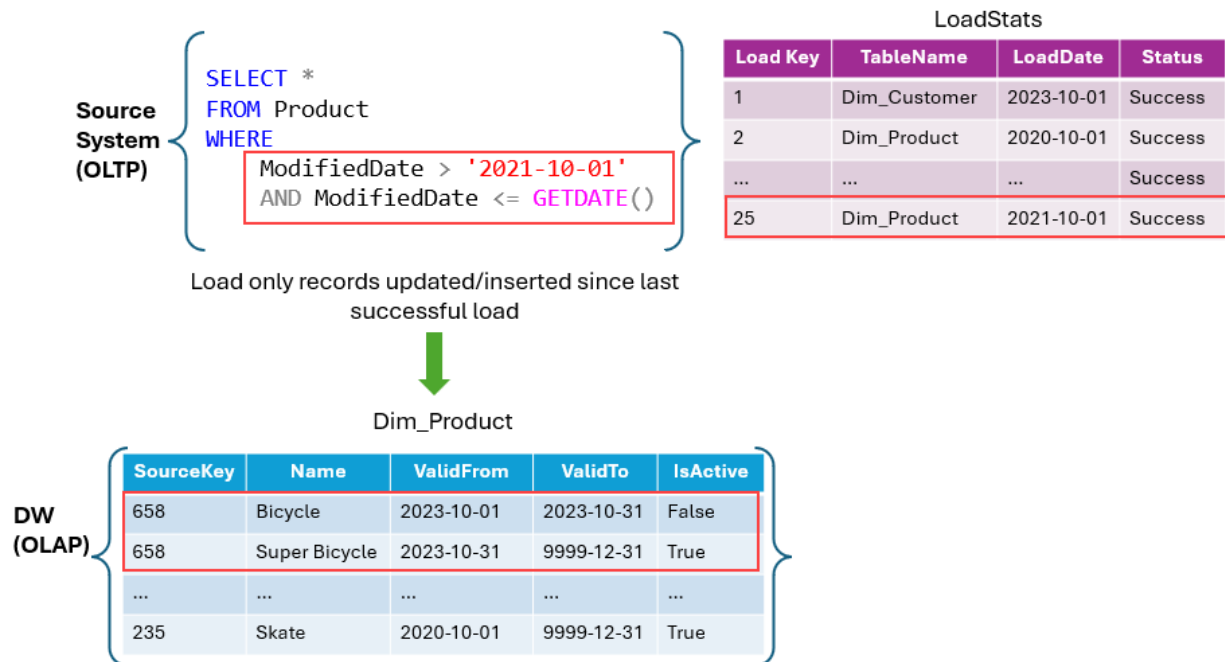
Slowly changing dimensions (SCD)

Slowly Changing Dimensions evolve over time, but at a slow pace and unpredictably. Take, for instance, a customer's address in a retail business. When a customer relocates, their address changes. If you overwrite the old address with the new one, you lose the historical data. However, if you need to analyze historical sales data, it's crucial to know where the customer lived at the time of each sale. This is where SCDs become essential.

There are several types of slowly changing dimensions in a data warehouse, with type 1 and type 2 being the most frequently used.

- **Type 0 SCD:** The dimension attributes never change.
- **Type 1 SCD:** Overwrites existing data, doesn't keep history.
- **Type 2 SCD:** Adds new records for changes, keeps full history for a given natural key.
- **Type 3 SCD:** History is added as a new column.
- **Type 4 SCD:** A new dimension is added.
- **Type 5 SCD:** When certain attributes of a large dimension change over time, but using type 2 isn't feasible due to the dimension's large size.
- **Type 6 SCD:** Combination of type 2 and type 3.

In type 2 SCD, when a new version of the same element is brought to the data warehouse, the old version is considered expired and the new one becomes active.



The following code shows a simple example on how to handle the business key in a type 2 SCD for the *Dim_Products* table using T-SQL.

```
IF EXISTS (SELECT 1 FROM Dim_Products WHERE SourceKey = @ProductID AND IsActive = 'True')
BEGIN
    -- Existing product record
    UPDATE Dim_Products
    SET ValidTo = GETDATE(), IsActive = 'False'
    WHERE SourceKey = @ProductID
    AND IsActive = 'True';
END
ELSE
BEGIN
    -- New product record
    INSERT INTO Dim_Products (SourceKey, ProductName, StartDate, EndDate, IsActive)
    VALUES (@ProductID, @ProductName, GETDATE(), '9999-12-31', 'True');
END
```

The mechanism for detecting changes in source systems is crucial for determining when records are inserted, updated, or deleted. [Change Data Capture \(CDC\)](#), [change tracking](#), and [triggers](#) are all features available for managing data tracking in source systems such as SQL Server.

Load a fact table

Typically, a standard data warehouse load operation involves loading fact tables after dimension tables. This ensures that the dimensions, which the facts reference, are already present in the data warehouse.

The staged fact data usually includes business keys for the related dimensions, so your loading logic must look up the corresponding surrogate keys. When dealing with slowly changing dimensions in the data

warehouse, it's crucial to identify the appropriate version of the dimension record to ensure the correct surrogate key is used. This matches the event recorded in the fact table with the state of the dimension at the time the fact occurred.

In many cases, you can retrieve the latest "current" version of the dimension. However, sometimes you might need to find the correct dimension record based on DateTime columns that indicate the period of validity for each version of the dimension.

The following example assumes that dimension records have incrementing surrogate keys and that the most recently added version of a specific dimension instance, which will have the highest key value, might be used.

```
-- Lookup keys in dimension tables
INSERT INTO dbo.FactSales
SELECT (SELECT MAX(DateKey)
        FROM dbo.DimDate
        WHERE FullDateAlternateKey = stg.OrderDate) AS OrderDateKey,
       (SELECT MAX(CustomerKey)
        FROM dbo.DimCustomer
        WHERE CustomerAlternateKey = stg.CustNo) AS CustomerKey,
       (SELECT MAX(ProductKey)
        FROM dbo.DimProduct
        WHERE ProductAlternateKey = stg.ProductID) AS ProductKey,
       (SELECT MAX(StoreKey)
        FROM dbo.DimStore
        WHERE StoreAlternateKey = stg.StoreID) AS StoreKey,
       OrderNumber,
       OrderLineItem,
       OrderQuantity,
       UnitPrice,
       Discount,
       Tax,
       SalesAmount
FROM dbo.StageSales AS stg
```

3. Use data pipelines to load a warehouse

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/3-load-data-using-data-pipeline>

Use data pipelines to load a warehouse

Completed

Microsoft Fabric's Warehouse provides integrated data ingestion tools, enabling users to load and ingest data into warehouses on a large scale through either coding or noncoding experiences.

Data pipeline is the cloud-based service for data integration, which enables the creation of workflows for data movement and data transformation at scale. You can create and schedule data pipelines that can ingest

and load data from disparate data stores. You can build complex ETL, or ELT processes that transform data visually with data flows.

Most of the functionality of data pipelines in Microsoft Fabric comes from Azure Data Factory, allowing for seamless integration and utilization of its features within the Microsoft Fabric ecosystem.

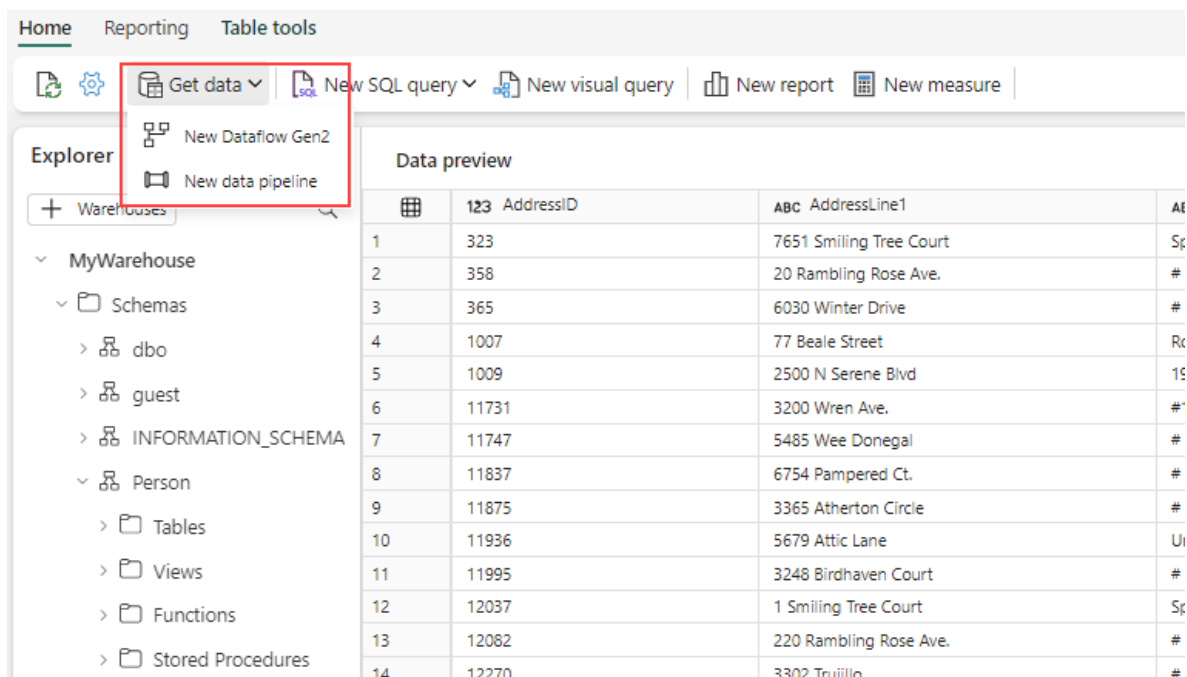
Note

All data in a Warehouse is automatically stored in the Delta Parquet format in OneLake.

Create a data pipeline

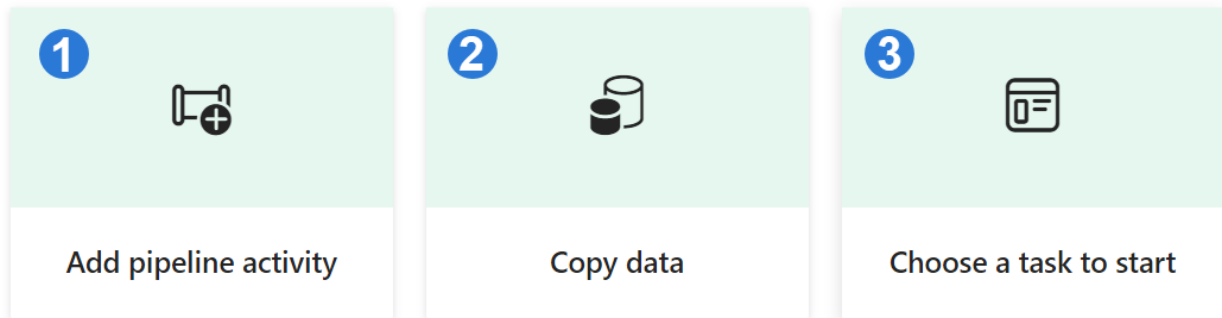
There are a few ways to launch the data pipeline editor.

- **From the workspace:** Select **+ New**, then select **Data pipeline**. If it's not visible in the list, select **More options**, then find **Data pipeline** under the **Get data** section.
- **From the warehouse asset** - Select **Get Data**, and then **New data pipeline**.



There are three options available when creating a pipeline.

Start building your data pipeline



Option	Description
1. Add pipeline activity	Launches the pipeline editor where you can create your own pipeline.
2. Copy data	Launches an assistant to copy data from various data sources to a data destination. A new pipeline activity is generated at the end with a preconfigured Copy Data task.
3. Choose a task to start	You can choose from a collection of predefined templates to assist you in initiating pipelines based on many scenarios.

Configure the copy data assistant

The copy data assistant provides a step-by-step interface that facilitates the configuration of a **Copy Data** task.

The screenshot shows the 'Copy data' assistant interface. On the left is a vertical sidebar with steps: 'Choose data source' (active), 'Connect to data source', 'Choose data destination', 'Connect to data destination', 'Settings', and 'Review + save'. The main area has a title 'Copy data' and a close button. Below the title is a 'Sample data' section with five cards: 'COVID-19 Data Lake' (various formats), 'NYC Taxi - Green' (2 GB Parquet), 'Diabetes' (14 KB Parquet), 'Public Holidays' (500 KB Parquet), and 'Retail Data Model from Wide World Importers' (352MB Parquet). Below this is a 'Data sources' section with tabs for 'All categories', 'Workspace', 'Azure' (selected), 'Database', 'File', 'Generic protocol', 'NoSQL', and 'Services and apps'. Under the 'Azure' tab are three cards: 'Azure Blob Storage', 'Azure Cosmos DB for NoSQL', and 'Azure Data Explorer'. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

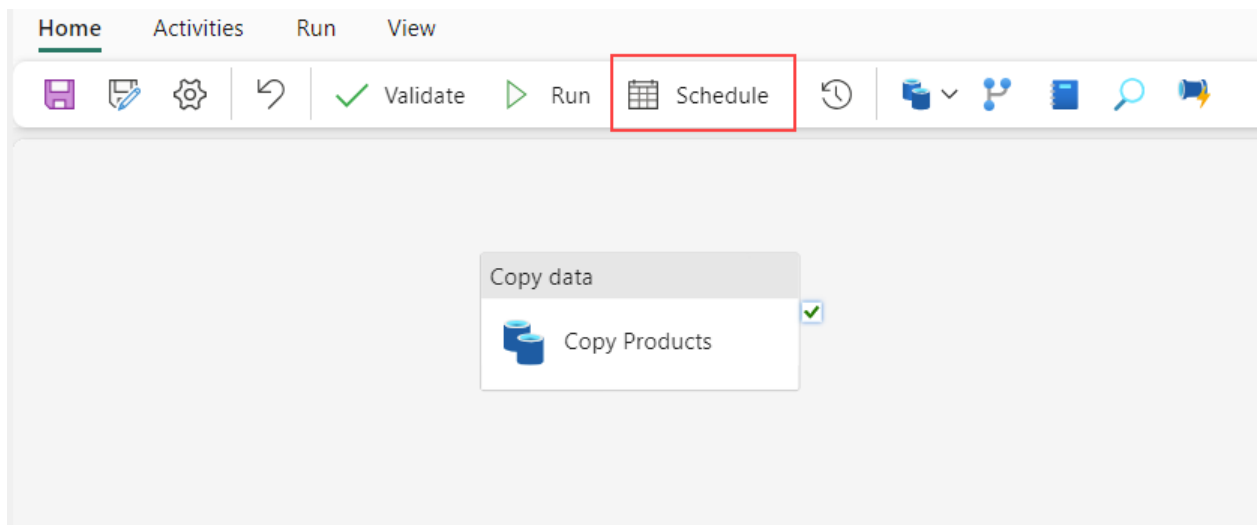
- **Choose data source:** Select a connector, and provide the connection information.
- **Connect to a data source:** Select, preview, and choose the data. This can be done from tables or views, or you can customize your selection by providing your own query.
- **Choose data destination:** Select the data store as the destination.

- **Connect to data destination:** Select and map columns from source to destination. You can load to a new or existing table.
- **Settings:** Configure other settings like staging, and default values.

After you copy the data, you can use other tasks to further transform and analyze it. You can also use the **Copy Data** task to publish transformation and analysis results for business intelligence (BI) and application consumption.

Schedule a data pipeline

You can schedule your data pipeline by selecting **Schedule** from the data pipeline editor.



You can also configure the schedule by selecting **Settings** in the **Home** menu in the data pipeline editor.

🕒 Schedule

Scheduled run

☒ On
 ☐ Off

Repeat

Weekly ▾

Every

☐ Su
 ☒ Mo
 ☐ Tu
 ☒ We
 ☐ Th
 ☒ Fr
 ☐ Sa

Time

Enter time 🕒 🗑️

+ Add a time

Start date and time

End date and time

Enter start date 📅

Enter end date 📅

Time zone

(UTC-06:00) Central Time (US and Canada) ▾

Apply

Discard

We recommend data pipelines for a code-free or low-code experience due to the graphical user interface. They're ideal for data workflows that run at a schedule, or that connects to different data sources.

To learn more about data pipelines, see [Ingest data into your Warehouse using data pipelines](#).

4. Load data using T-SQL

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/4-load-data-using-tsql>

Load data using T-SQL

Completed

SQL developers or citizen developers, who are often well-versed in the SQL engine and adept at using T-SQL, will find the Warehouse in Microsoft Fabric favorable.

This is because the Warehouse is powered by the same SQL engine they're familiar with, enabling them to perform complex queries and data manipulations. These operations include filtering, sorting, aggregating, and joining data from different tables. The SQL engine's wide range of functions and operators further allows for sophisticated data analysis and transformations at the database level.

Use COPY statement

The [COPY statement](#) serves as the main method for importing data into the Warehouse. It facilitates efficient data ingestion from an external Azure storage account.

It offers flexibility, allowing you to specify the format of the source file, designate a location for storing rows that are rejected during the import process, skip header rows, among other configurable options.

The option to store rejected rows separately is useful for data cleaning and quality control. It allows you to easily identify and investigate any issues with the data that weren't successfully imported.

To connect to an Azure storage account, you need to use either Shared Access Signature (SAS) or Storage Account Key (SAK).

Handle error

The option to use a different storage account for the *ERRORFILE* location (`REJECTED_ROW_LOCATION`) allows for better error handling and debugging. It makes it easier to isolate and investigate any issues that occur during the data loading process. *ERRORFILE* only applies to CSV.

Load multiple files

The ability to specify wildcards and multiple files in the storage location path allows the COPY statement to handle bulk data loading efficiently. This is useful when dealing with large datasets distributed across multiple files.

Multiple file locations can only be specified from the same storage account and container via a comma-separated list.

```
COPY INTO my_table
FROM 'https://myaccount.blob.core.windows.net/myblobcontainer/folder0/*.csv,
     https://myaccount.blob.core.windows.net/myblobcontainer/folder1/'
WITH (
    FILE_TYPE = 'CSV',
    CREDENTIAL=(IDENTITY= 'Shared Access Signature', SECRET='<Your_SAS_Token>')
    FIELDTERMINATOR = '|'
)
```

The following example shows how to load a PARQUET file.

```
COPY INTO test_parquet
FROM 'https://myaccount.blob.core.windows.net/myblobcontainer/folder1/*.parquet'
WITH (
```

```
CREDENTIAL=(IDENTITY= 'Shared Access Signature', SECRET='<Your_SAS-Token>')  
)
```

Ensure that all the files have the same structure (that is, same columns in the same order) and that this structure matches the structure of the target table.

Load table from other warehouses and lakehouses

You can load data from various data assets in a workspace, such as other warehouses and lakehouses.

To reference the data asset, ensure that you use [three-part naming](#) to combine data from tables on these workspace assets. You can then use `CREATE TABLE AS SELECT` (CTAS) and `INSERT...SELECT` to load the data into the warehouse.

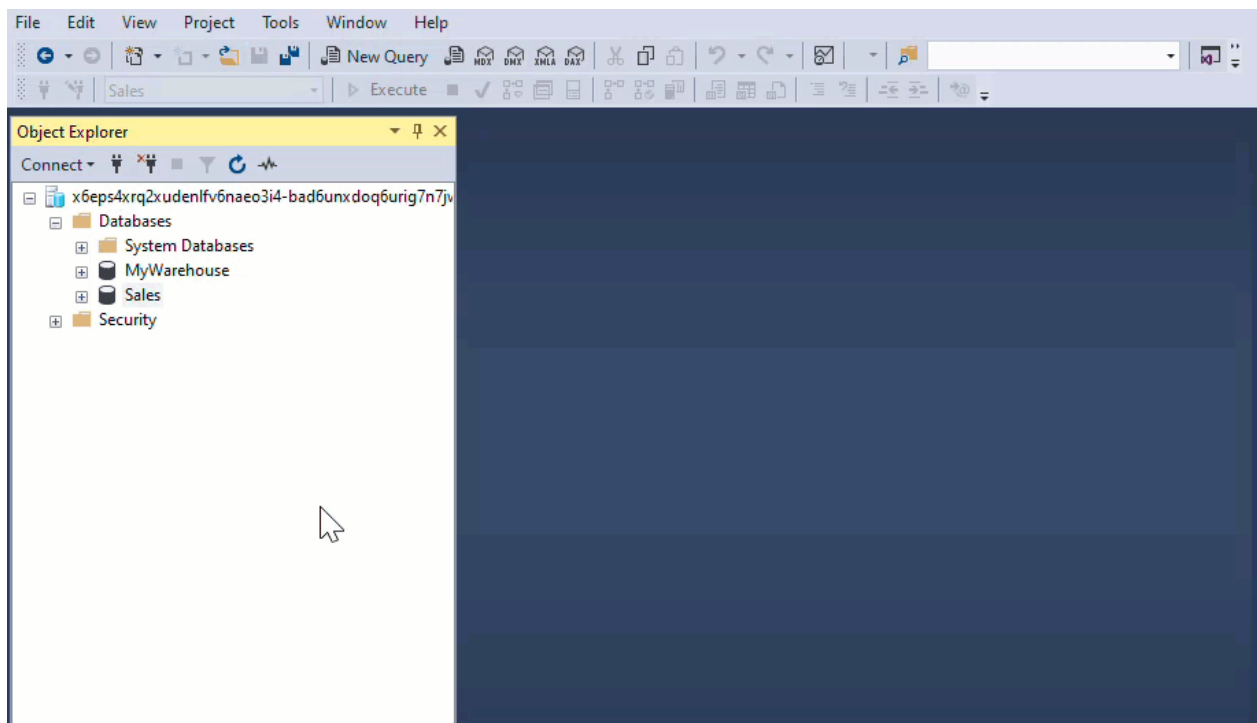
SQL Statement	Description
<code>CREATE TABLE AS SELECT</code>	Allows you to create a new table based on the output of a <code>SELECT</code> statement. This operation is often used for creating a copy of a table or for transforming and loading the results of complex queries.
<code>INSERT...SELECT</code>	Allows you to insert data from one table into another. It's useful when you want to copy data from one table to another without creating a new table.

In a scenario where an analyst needs data from both a warehouse and a lakehouse, they can use this feature to combine the data. They can then load this combined data into the warehouse for analysis. This feature is useful when data is distributed across many assets in a workspace.

The following query creates a new table in the `analysis_warehouse` that combines data from the `sales_warehouse` and the `social_lakehouse` using the `product_id` as the common key. The new table can then be used for further analysis.

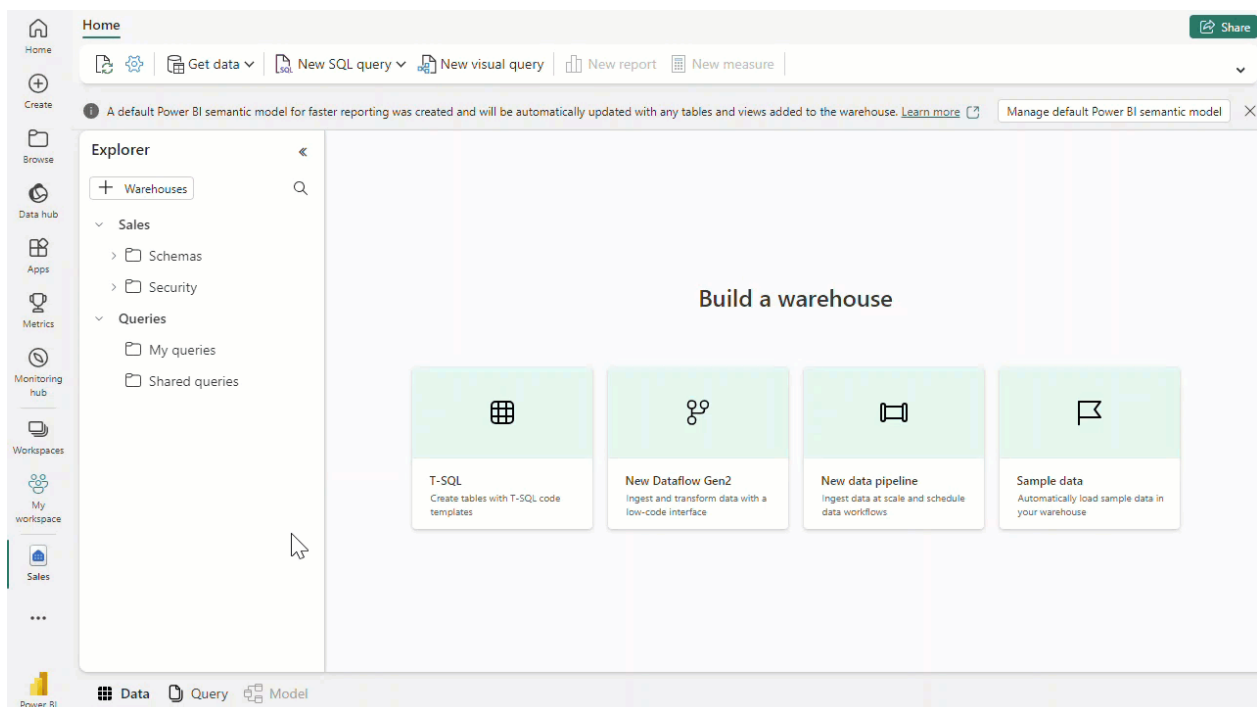
```
CREATE TABLE [analysis_warehouse].[dbo].[combined_data]  
AS  
SELECT *  
FROM [sales_warehouse].[dbo].[sales_data] sales  
INNER JOIN [social_lakehouse].[dbo].[social_data] social  
ON sales.[product_id] = social.[product_id];
```

All the Warehouses that share the same workspace are integrated into the same logical SQL server. If you use SQL client tools such as [SQL Server Management Studio](#), you can easily perform a cross-database query like in any SQL Server instance.



MyWarehouse and *Sales* are both warehouse assets that share the same workspace.

If you're using the object Explorer from the workspace to query your Warehouses, you need to add them explicitly. The warehouses added will also be visible from the Visual query editor.



Data can be efficiently loaded into a warehouse in Microsoft Fabric through the COPY statement, or from other warehouses and lakehouses within the same workspace, allowing for seamless data management and analysis.

5. Load and transform data with Dataflow Gen2

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/5-load-data-using-dataflow>

Load and transform data with Dataflow Gen2

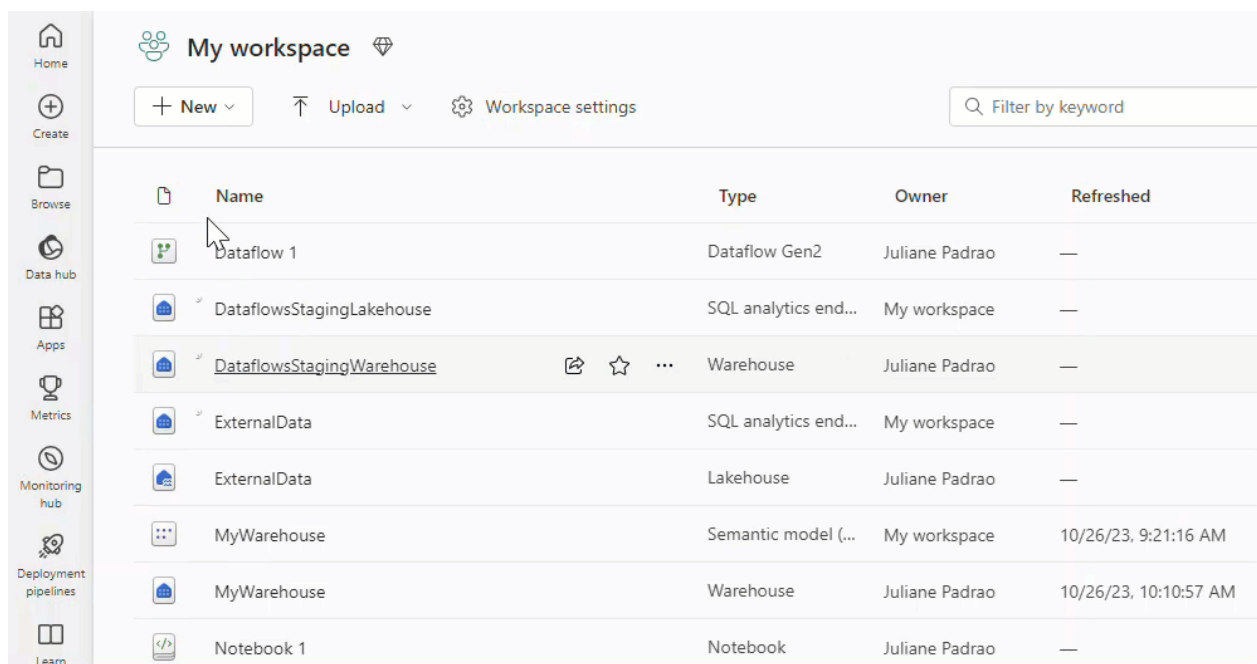
Completed

Dataflow Gen2 is the new generation of dataflows. It provides a comprehensive Power Query experience, guiding you through each step of importing data into your dataflow. The process of creating dataflows has been simplified, reducing the number of steps involved.

You can use dataflows in data pipelines to ingest data into a lakehouse or warehouse, or to define a dataset for a Power BI report.

Create a dataflow

To create a new dataflow, navigate to your workspace, then select **+ New**. If **Dataflow Gen2** isn't visible in the list, select **More options**, then find **Dataflow Gen2** under the **Data Factory** section.

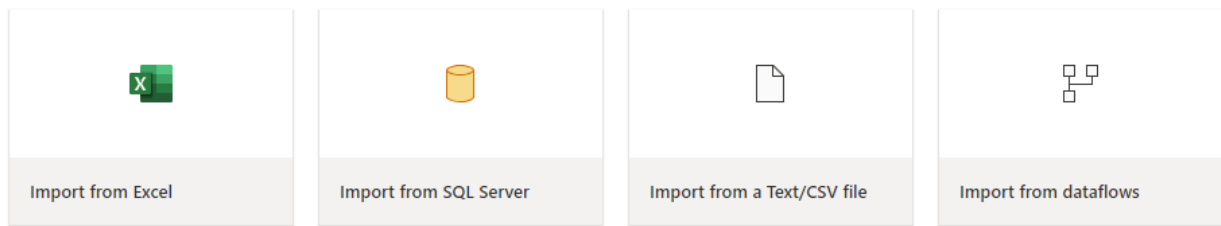


The screenshot shows the Microsoft Fabric workspace interface. On the left is a navigation sidebar with icons for Home, Create, Browse, Data hub, Apps, Metrics, Monitoring hub, Deployment pipelines, and Learn. The main area is titled 'My workspace' and contains a table of data assets. The table has columns for Name, Type, Owner, and Refreshed. A mouse cursor is hovering over the 'Dataflow 1' entry. The table lists various data assets including Dataflows, Staging Lakehouses, Staging Warehouses, External Data, and Warehouses, along with their types and owners.

Name	Type	Owner	Refreshed
Dataflow 1	Dataflow Gen2	Juliane Padrao	—
DataflowsStagingLakehouse	SQL analytics end...	My workspace	—
DataflowsStagingWarehouse	Warehouse	Juliane Padrao	—
ExternalData	SQL analytics end...	My workspace	—
ExternalData	Lakehouse	Juliane Padrao	—
MyWarehouse	Semantic model (...)	My workspace	10/26/23, 9:21:16 AM
MyWarehouse	Warehouse	Juliane Padrao	10/26/23, 10:10:57 AM
Notebook 1	Notebook	Juliane Padrao	—

Import data

Once the Dataflow Gen2 launches, there are many options to load your data available.



[Get data from another source →](#)

[Import from a Power Query template](#)

You can load different file types with just a few steps. For example, to load a text or CSV file from your local computer.

Get data

Connect to data source



Text/CSV
File
[Learn more](#)

Connection settings

☐ Link to file ☒ Upload file (Preview) ⓘ

 customer-churn 1.csv
Upload successful (481.6 KB)

Connection credentials

Connection
 ↻

Authentication kind: Organizational account [Edit connection](#)

Back

Cancel

Next

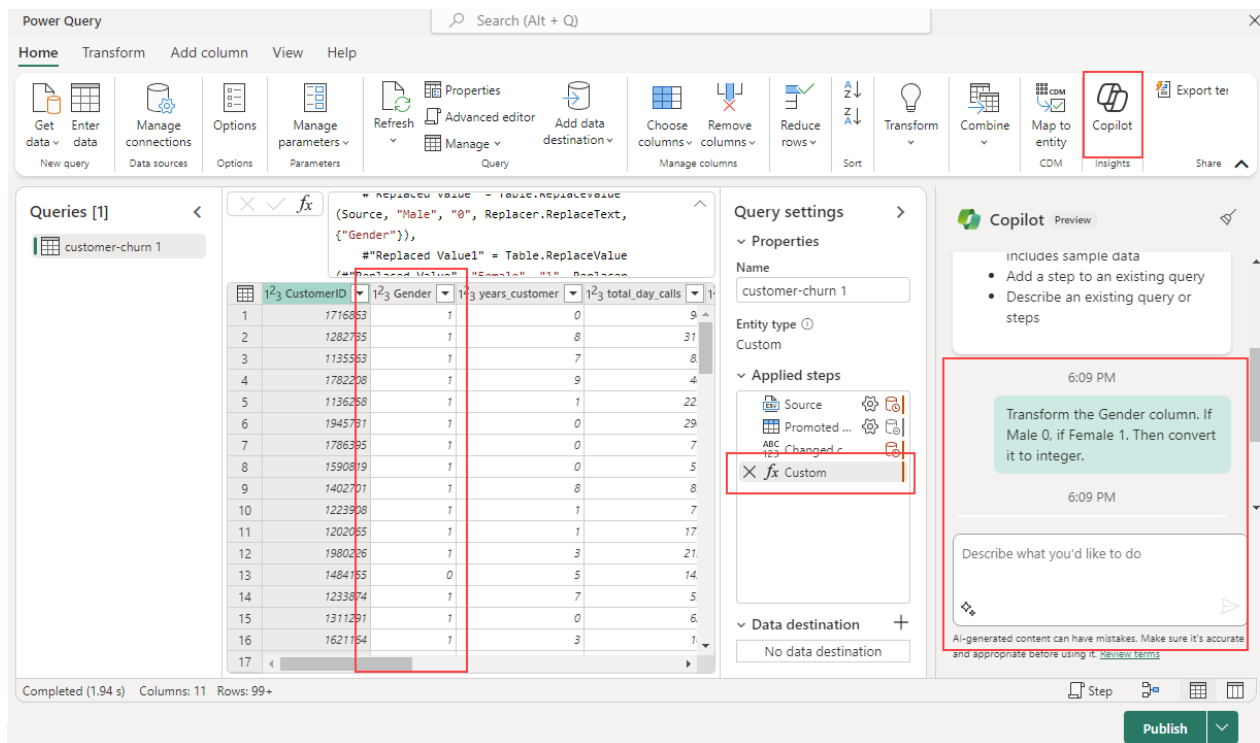
Once the data is imported you can start authoring your dataflow, you might decide to clean your data, reshape, remove columns, and create new ones. All the steps you perform are saved.

Transform data with Copilot

Copilot can be a valuable tool for assisting with dataflow transformations. Let's say we have a *Gender* column that contains 'Male' and 'Female' and we want to transform it.

The first step is to activate Copilot within your dataflow. Once that's done, you can then provide specific instructions on the transformation you want to perform.

For instance, you might input the following command: *"Transform the Gender column. If Male 0, if Female 1. Then convert it to integer."*



Copilot adds a new step automatically, and you can always revert it if you want, or continue to build on it for further transformations.

Add a data destination

With the **Add data destination** feature, you can separate your ETL logic and destination storage. This separation can lead to cleaner, more maintainable code and can make it easier to modify either the ETL process or the storage configuration without affecting the other.

Once the data is transformed, the next step is to add a destination step. On the **Query settings** tab, select **+** to add a destination step in your dataflow.

Query settings

▼ Properties

Name

customer-churn 1

Entity type ⓘ

Custom

▼ Applied steps

Source

Promoted headers

Changed column type

Custom

▼ Data destination

No data destination

The following destination options are available.

- Azure SQL Database
- Lakehouse
- Azure Data Explorer (Kusto)
- Azure Synapse Analytics (SQL DW)
- Warehouse

Data that's loaded into a destination like a warehouse can be easily accessed and analyzed using various tools. This improves the accessibility of your data and allows for more flexible and comprehensive data analysis.

When you select a warehouse as a destination, you can choose the following update methods.



- **Append:** Add new rows to an existing table.
- **Replace:** Replace the entire content of a table with a new set of data.

Publish a dataflow

After you choose your update method, the final step is to publish your dataflow.

Publishing makes your transformations and data loading operations live, allowing the dataflow to be executed either manually or on a schedule. This process encapsulates your ETL operations into a single and reusable unit, streamlining your data management workflow.

Any changes made in the dataflow take effect when it's published. So, always ensure to publish your dataflow after making any relevant modifications.

6. Exercise: Load data into a warehouse in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/6-exercise-load-data-into-warehouse>

Exercise: Load data into a warehouse in Microsoft Fabric

Completed

Now it's your chance to load data into a warehouse in Microsoft Fabric.

In this exercise, you'll learn how to load data into a warehouse using T-SQL.

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](#)

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/load-data-into-microsoft-fabric-data-warehouse/8-summary>

Summary

Completed

There's no one-size-fits-all solution for loading your data. The best approach depends on the specifics of your business requirement and the question you're trying to answer.

When it comes to loading data in a data warehouse, there are several considerations to keep in mind.

	Description
Load volume & frequency	Assess data volume and load frequency to optimize performance.
Governance	Any data that lands in OneLake is governed by default.
Data mapping	Manage mapping from source to staging to warehouse.
Dependencies	Understand dependencies in the data model for loading dimensions.
Script design	Design efficient import scripts considering column names, filtering rules, value mapping, and database indexing.

For additional reading, you can refer to the following URLs:

- [Create a Warehouse in Microsoft Fabric](#)
- [Ingest data into the Warehouse](#)
- [Compare the Warehouse and the SQL analytics endpoint of the Lakehouse](#)

Query a data warehouse in Microsoft Fabric

1. Introduction

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

Introduction

Completed

Microsoft Fabric Data Warehouse is a complete platform for data, analytics, and AI (Artificial Intelligence). It refers to the process of storing, organizing, and managing large volumes of structured and semi-structured data.

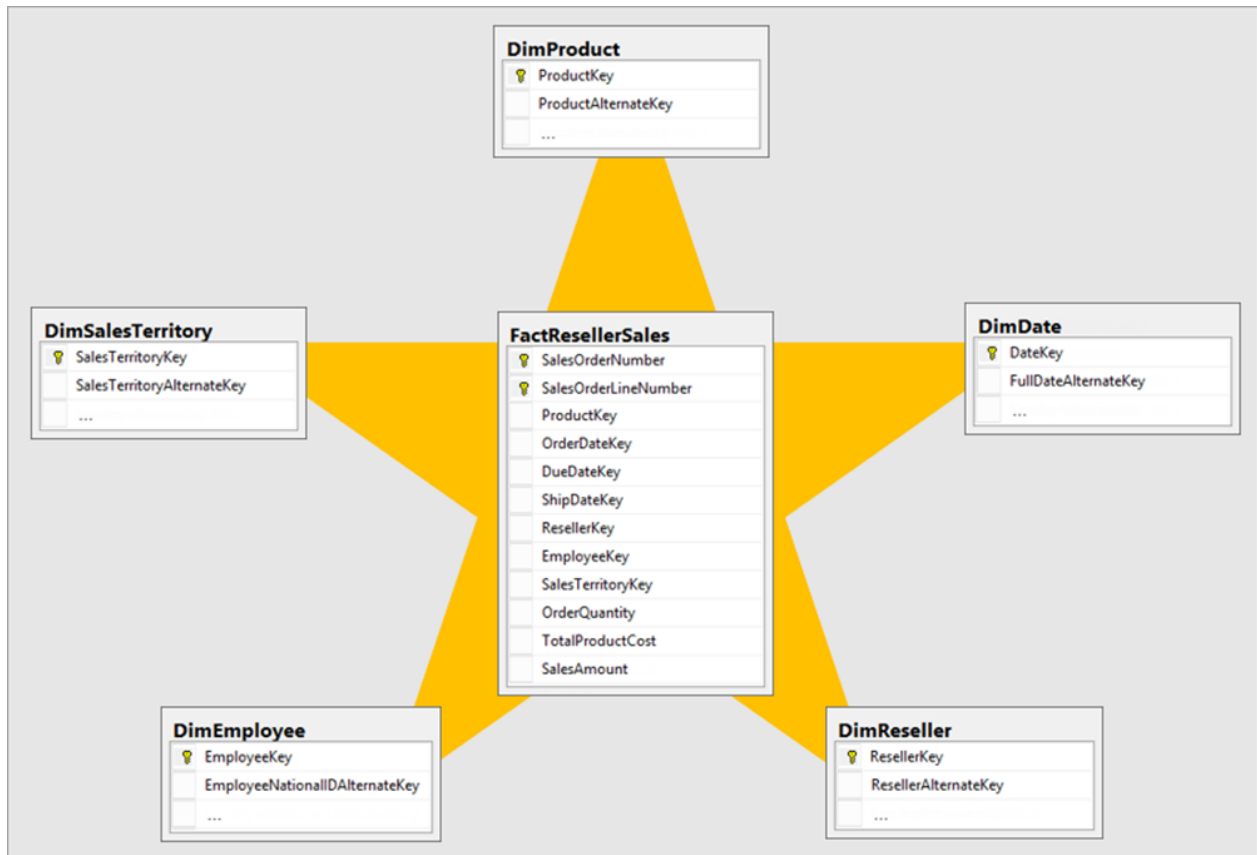
Data warehouse in Microsoft Fabric offers a rich set of features that make it easier to manage and analyze data. It includes advanced query processing capabilities, and supports the full transactional T-SQL capabilities like an enterprise data warehouse.

The process of querying a data warehouse is a key component of business intelligence. It involves the extraction and manipulation of the data stored in a data warehouse, allowing users to extract valuable insights from large volumes of data.

Star schema design

In a typical data warehouse, the data is organized using a schema, often a [star schema](#) or a [snowflake schema](#). The star schema and snowflake schema are mature modeling approaches widely adopted by relational data warehouses. It requires you to classify tables as either dimension or fact.

Fact tables store the measurable, quantitative data about a business, while dimension tables contain descriptive attributes related to fact data.



Think of a dimension table as the "*who, what, where, when, why*" of your data warehouse. It's like the descriptive backdrop that gives context to the raw numbers found in the fact tables.

For example, if you're running an online store, your fact table might contain the raw sales data - how many units of each product were sold. But without a dimension table, you wouldn't know who bought those products, when they were bought, or where the buyer is located.

We'll explore different ways to connect and query a data warehouse, and how they can facilitate the tasks of effectively extracting information.

For more information, see [Understand star schema and the importance for Power BI](#).

2. Query data

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/2-query-data>

Query data

Completed

Once the dimension and fact tables in a data warehouse are populated with data, you can use T-SQL to query these tables and perform data analysis. The Transact-SQL (T-SQL) syntax used for querying tables in a

warehouse in Fabric closely resembles the SQL syntax used in SQL Server or Azure SQL Database. This familiarity allows for an easy transition for those already used to working with these platforms.

Aggregate measures by dimension attributes

In most data analytics scenarios involving a data warehouse, the process typically revolves around aggregating numeric measures from fact tables based on attributes in dimension tables. Due to the structure of a star or snowflake schema, such aggregation queries depend on `JOIN` clauses to link fact tables with dimension tables. Also, they use aggregate functions and `GROUP BY` clauses to define the aggregation hierarchies.

The following SQL query aggregates sales amounts by year and quarter from the **FactSales** and **DimDate** tables in a hypothetical data warehouse:

```
SELECT  dates.CalendarYear,
        dates.CalendarQuarter,
        SUM(sales.SalesAmount) AS TotalSales
FROM    dbo.FactSales AS sales
JOIN    dbo.DimDate AS dates ON sales.OrderDateKey = dates.DateKey
GROUP BY dates.CalendarYear, dates.CalendarQuarter
ORDER BY dates.CalendarYear, dates.CalendarQuarter;
```

The results from this query would look similar to the following table.

CalendarYear	CalendarQuarter	TotalSales
2020	1	25980.16
2020	2	27453.87
2020	3	28527.15
2020	4	31083.45
2021	1	34562.96
2021	2	36162.27
...	...	...

You can join as many dimension tables as needed to calculate the aggregations you need. For example, the following code extends the previous example to break down the quarterly sales totals by city based on the customer's address details in the **DimCustomer** table.

```
SELECT  dates.CalendarYear,
        dates.CalendarQuarter,
        custs.City,
        SUM(sales.SalesAmount) AS TotalSales
FROM    dbo.FactSales AS sales
JOIN    dbo.DimDate AS dates ON sales.OrderDateKey = dates.DateKey
JOIN    dbo.DimCustomer AS custs ON sales.CustomerKey = custs.CustomerKey
GROUP BY dates.CalendarYear, dates.CalendarQuarter, custs.City
ORDER BY dates.CalendarYear, dates.CalendarQuarter, custs.City;
```

This time, the results include a quarterly sales total for each city.

CalendarYear	CalendarQuarter	City	TotalSales
2020	1	Amsterdam	5982.53
2020	1	Berlin	2826.98
2020	1	Chicago	5372.72
...	...	...	..
2020	2	Amsterdam	7163.93
2020	2	Berlin	8191.12
2020	2	Chicago	2428.72
...	...	...	..
2020	3	Amsterdam	7261.92
2020	3	Berlin	4202.65
2020	3	Chicago	2287.87
...	...	...	..
2020	4	Amsterdam	8262.73
2020	4	Berlin	5373.61
2020	4	Chicago	7726.23
...	...	...	..
2021	1	Amsterdam	7261.28
2021	1	Berlin	3648.28
2021	1	Chicago	1027.27
...	...	...	..

Joins in a snowflake schema

When using a snowflake schema, dimensions may be partially normalized; requiring multiple joins to relate fact tables to snowflake dimensions. For example, suppose your data warehouse includes a **DimProduct** dimension table from which the product categories have been normalized into a separate **DimCategory** table. A query to aggregate items sold by product category might look similar to the following example:

```
SELECT  cat.ProductCategory,
        SUM(sales.OrderQuantity) AS ItemsSold
FROM    dbo.FactSales AS sales
JOIN    dbo.DimProduct AS prod ON sales.ProductKey = prod.ProductKey
JOIN    dbo.DimCategory AS cat ON prod.CategoryKey = cat.CategoryKey
GROUP BY cat.ProductCategory
ORDER BY cat.ProductCategory;
```

The results from this query include the number of items sold for each product category:

ProductCategory	ItemsSold
Accessories	28271
Bits and pieces	5368
...	...

Note

JOIN clauses for **FactSales** and **DimProduct** and for **DimProduct** and **DimCategory** are both required, even though no fields from **DimProduct** are returned by the query.

Using ranking functions

Another common kind of analytical query is to partition the results based on a dimension attribute and *rank* the results within each partition. For example, you might want to rank stores each year by their sales revenue. To accomplish this goal, you can use Transact-SQL *ranking* functions such as **ROW_NUMBER**, **RANK**, **DENSE_RANK**, and **NTILE**. These functions enable you to partition the data over categories, each returning a specific value that indicates the relative position of each row within the partition:

- **ROW_NUMBER** returns the ordinal position of the row within the partition. For example, the first row is numbered 1, the second 2, and so on.
- **RANK** returns the ranked position of each row in the ordered results. For example, in a partition of stores ordered by sales volume, the store with the highest sales volume is ranked 1. If multiple stores have the same sales volumes, they'll be ranked the same, and the rank assigned to subsequent stores reflects the number of stores that have higher sales volumes - including ties.
- **DENSE_RANK** ranks rows in a partition the same way as **RANK**, but when multiple rows have the same rank, subsequent rows are ranking positions ignore ties.
- **NTILE** returns the specified percentile in which the row falls. For example, in a partition of stores ordered by sales volume, **NTILE(4)** returns the quartile in which a store's sales volume places it.

For example, consider the following query:

```
SELECT ProductCategory,
       ProductName,
       ListPrice,
       ROW_NUMBER() OVER
         (PARTITION BY ProductCategory ORDER BY ListPrice DESC) AS RowNumber,
       RANK() OVER
         (PARTITION BY ProductCategory ORDER BY ListPrice DESC) AS Rank,
       DENSE_RANK() OVER
         (PARTITION BY ProductCategory ORDER BY ListPrice DESC) AS DenseRank,
       NTILE(4) OVER
         (PARTITION BY ProductCategory ORDER BY ListPrice DESC) AS Quartile
FROM dbo.DimProduct
ORDER BY ProductCategory;
```

The query partitions products into groupings based on their categories, and within each category partition, the relative position of each product is determined based on its list price. The results from this query might look similar to the following table:

ProductCategory	ProductName	ListPrice	RowNumber	Rank	DenseRank	Quartile
Accessories	Widget	8.99	1	1	1	1
Accessories	Knicknak	8.49	2	2	2	1
Accessories	Sprocket	5.99	3	3	3	2
Accessories	Doodah	5.99	4	3	3	2
Accessories	Spangle	2.99	5	5	4	3
Accessories	Badabing	0.25	6	6	5	4
Bits and pieces	Flimflam	7.49	1	1	1	1
Bits and pieces	Snickity wotsit	6.99	2	2	2	1
Bits and pieces	Flange	4.25	3	3	3	2
...	...	...	...	...	...	...

Note

The sample results demonstrate the difference between `RANK` and `DENSE_RANK`. Note that in the *Accessories* category, the *Sprocket* and *Doodah* products have the same list price; and are both ranked as the 3rd highest priced product. The next highest priced product has a *RANK* of 5 (there are four products more expensive than it) and a *DENSE_RANK* of 4 (there are three higher prices).

To learn more about ranking functions, see the [Use built-in functions and GROUP BY in Transact-SQL](#) module.

Retrieving an approximate count

While the purpose of a data warehouse is primarily to support analytical data models and reports for the enterprise; data analysts and data scientists often need to perform some initial data exploration, just to determine the basic scale and distribution of the data.

For example, the following query uses the `COUNT` function to retrieve the number of sales for each year in a hypothetical data warehouse:

```
SELECT dates.CalendarYear AS CalendarYear,
       COUNT(DISTINCT sales.OrderNumber) AS Orders
FROM FactSales AS sales
JOIN DimDate AS dates ON sales.OrderDateKey = dates.DateKey
GROUP BY dates.CalendarYear
ORDER BY CalendarYear;
```

The results of this query might look similar to the following table:

CalendarYear	Orders
2019	239870
2020	284741

CalendarYear	Orders
2021	309272
...	...

The volume of data in a data warehouse can mean that even simple queries to count the number of records that meet specified criteria can take a considerable time to run. In many cases, a precise count isn't required - an approximate estimate will suffice. In such cases, you can use the `APPROX_COUNT_DISTINCT` function as shown in the following example:

```
SELECT dates.CalendarYear AS CalendarYear,
       APPROX_COUNT_DISTINCT(sales.OrderNumber) AS ApproxOrders
FROM FactSales AS sales
JOIN DimDate AS dates ON sales.OrderDateKey = dates.DateKey
GROUP BY dates.CalendarYear
ORDER BY CalendarYear;
```

The `APPROX_COUNT_DISTINCT` function uses a *HyperLogLog* algorithm to retrieve an approximate count. The result is guaranteed to have a maximum error rate of 2% with 97% probability, so the results of this query with the same hypothetical data as before might look similar to the following table:

CalendarYear	ApproxOrders
2019	235552
2020	290436
2021	304633
...	...

The counts are less accurate, but still sufficient for an approximate comparison of yearly sales. With a large volume of data, the query using the `APPROX_COUNT_DISTINCT` function completes more quickly, and the reduced accuracy may be an acceptable trade-off during basic data exploration.

Note

See the [APPROX_COUNT_DISTINCT](#) function documentation for more details.

3. Use the SQL query editor

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/3-use-sql-query-editor>

Use the SQL query editor

Completed

The SQL query editor in Microsoft Fabric is a versatile tool that supports Transact-SQL (T-SQL), allowing you to create and run scripts to query your data warehouse. It also provides features such as IntelliSense and debugging to aid in the development process.

T-SQL allows users, analysts and developers to manipulate the data stored in a warehouse.

Launch a SQL query editor

The SQL query editor isn't just about querying; it's about managing your data effectively. Whether you're creating a new table, inserting rows into a table, granting objects permission or executing complex queries, the SQL query editor is designed to make these tasks seamless.

To launch the SQL query editor, you need to select your warehouse asset from **My workspace**. This step connects to your data warehouse automatically, without you having to prompt any connection information.

From the warehouse **Explorer**, there are a few ways to create a SQL query editor.

The screenshot displays the Microsoft Fabric interface. At the top, the 'Home' menu is visible, with a red box highlighting the 'New SQL query' option. A red arrow points from this option to the 'Queries' node in the 'Explorer' sidebar, which is also highlighted with a red box. Another red arrow points from the 'Queries' node to the 'Query' tab at the bottom of the interface. The main area shows a 'Data preview' table with columns like 'DateID', 'Date', 'DateKey', 'DayOfMonth', 'DaySuffix', 'DayName', 'DayOfWeek', and 'DayOfWeekInMon'. The table contains 1,000 rows of data. The status bar at the bottom indicates 'Succeeded (28 sec 562 ms)' and 'Columns: 32 Rows: 1,000'.

1. **Home** menu: Select **New SQL query** or select any of the templates available.
2. **Queries** node: Select ... in **My queries**, then select **New SQL Query**.
3. **Query** tab: This option opens a new query editor.

No matter how you initiate a new query editor, a new query item is automatically created in **My queries** folder in the **Explorer**.

Any new query item updated is automatically saved.

Run queries and view results

To run a query, type it into a new query editor and select **Run** at the top of the editor.

The screenshot shows the Microsoft Fabric Reporting interface. On the left is the Explorer pane with a tree view of the 'sample-dw' warehouse, including schemas like 'dbo', 'guest', 'INFORMATION_SCHEMA', and 'sys'. The main area is the 'New SQL query' editor, which is currently blank. A dropdown menu is open, showing options like 'Blank', 'New SQL query', 'Schema', 'Table', 'View', and 'Stored procedure'. The table of results is displayed below the editor, showing 1,000 rows of data. The table has columns: DateID, Date, ABC DateKey, ABC DayOfMonth, ABC DaySuffix, ABC DayName, ABC DayOfWeek, and ABC DayOfWeekinMon. The status bar at the bottom indicates 'Succeeded (4 sec 326 ms)' and 'Columns: 32 Rows: 1,000'.

DateID	Date	ABC DateKey	ABC DayOfMonth	ABC DaySuffix	ABC DayName	ABC DayOfWeek	ABC DayOfWeekinMon	
150317	2015-03-17 00:00:00.000000	03/17/2015	17	17th	Tuesday	3	3	
150317	2005-03-17 00:00:00.000000	03/17/2005	17	17th	Thursday	5	3	
110317	2001-03-17 00:00:00.000000	03/17/2001	17	17th	Saturday	7	3	
190317	2009-03-17 00:00:00.000000	03/17/2009	17	17th	Tuesday	3	3	
140317	2004-03-17 00:00:00.000000	03/17/2004	17	17th	Wednesday	4	3	
100317	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3	
120317	2012-03-17 00:00:00.000000	03/17/2012	17	17th	Saturday	7	3	
8	20060317	2006-03-17 00:00:00.000000	03/17/2006	17	17th	Friday	6	3
9	20110317	2011-03-17 00:00:00.000000	03/17/2011	17	17th	Thursday	5	3
10	20100317	2010-03-17 00:00:00.000000	03/17/2010	17	17th	Wednesday	4	3
11	20070317	2007-03-17 00:00:00.000000	03/17/2007	17	17th	Saturday	7	3
12	20140214	2014-02-14 00:00:00.000000	02/14/2014	14	14th	Friday	6	2
13	20030214	2003-02-14 00:00:00.000000	02/14/2003	14	14th	Friday	6	2
14	20080214	2008-02-14 00:00:00.000000	02/14/2008	14	14th	Thursday	5	2
15	20110214	2011-02-14 00:00:00.000000	02/14/2011	14	14th	Monday	2	2
16	20050214	2005-02-14 00:00:00.000000	02/14/2005	14	14th	Monday	2	2
17	20010214	2001-02-14 00:00:00.000000	02/14/2001	14	14th	Wednesday	4	2
18	20060214	2006-02-14 00:00:00.000000	02/14/2006	14	14th	Tuesday	3	2
19	20020214	2002-02-14 00:00:00.000000	02/14/2002	14	14th	Thursday	5	2
20	20120214	2012-02-14 00:00:00.000000	02/14/2012	14	14th	Tuesday	3	2
21	20000214	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2
22	20100214	2010-02-14 00:00:00.000000	02/14/2010	14	14th	Sunday	1	2
23	20070214	2007-02-14 00:00:00.000000	02/14/2007	14	14th	Wednesday	4	2
24	20130214	2013-02-14 00:00:00.000000	02/14/2013	14	14th	Thursday	5	2
25	20091031	2009-10-31 00:00:00.000000	10/31/2009	31	31st	Saturday	7	5
26	20121031	2012-10-31 00:00:00.000000	10/31/2012	31	31st	Wednesday	4	5
27	20071031	2007-10-31 00:00:00.000000	10/31/2007	31	31st	Wednesday	4	5
28	20021031	2002-10-31 00:00:00.000000	10/31/2002	31	31st	Thursday	5	5
29	20151031	2015-10-31 00:00:00.000000	10/31/2015	31	31st	Saturday	7	5

The **Results** section shows a preview, limited to 10,000 rows if exceeded.

Export results

Your query results can be exported as an Excel file. To export it, select **Download Excel file**.

You need to select the text of one `SELECT` statement in your query to export results to Excel.

Similarly, the query editor offers the ability to save your highlighted query as either a view or a table, each with distinct features:

- **Save as view:** Allows you to create a view in the warehouse using the `SELECT` statement highlighted in the query editor.
- **Save as table:** Creates a new table with your query results.

For both options, you must provide the schema and the name before confirming the creation.

To learn more about SQL query editor, see [Query using the SQL query editor](#).

4. Explore the visual query editor

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/4-explore-visual-query-editor>

Explore the visual query editor

Completed

The visual query editor in Microsoft Fabric data warehouse is a tool that provides an intuitive interface for building SQL queries.

It simplifies the process of writing and managing queries, especially for those who might not be comfortable with SQL syntax.

- **Graphical interface:** The editor provides a graphical interface where you can drag and drop tables, and visually design your queries. This makes it easier to understand the structure of your query and the relationships between tables.
- **Automatic query generation:** As you design your query using the visual interface, the corresponding SQL query is automatically generated. This allows you to focus on the logic of your query, rather than the syntax. All workspace users can save their queries in **My queries** folder.

Build your query visually

The visual query editor enhances intuitive understanding of data relationships by allowing users to visually organize tables and fields.

Also, the visual query editor is user-friendly, even for those with little to no SQL experience. Nontechnical team members can use it to extract valuable insights from your data, democratizing data access within your organization and speeding up the decision-making process.

123 DateID	Date	ABC DateKey	ABC DayOfMonth	ABC DaySuffix	ABC DayName	ABC DayOfWeek	ABC DayOfWeekInMonth	ABC DayOfWeekInYear
94	2000-01-01 00:00:00.000000	01/01/2000	1	1st	Saturday	7	1	1
960	2000-01-29 00:00:00.000000	01/29/2000	29	29th	Saturday	7	5	5
138	2000-01-30 00:00:00.000000	01/30/2000	30	30th	Sunday	1	5	5
108	2000-01-31 00:00:00.000000	01/31/2000	31	31st	Monday	2	5	5
283	2000-02-01 00:00:00.000000	02/01/2000	1	1st	Tuesday	3	1	5
361	2000-02-02 00:00:00.000000	02/02/2000	2	2nd	Wednesday	4	1	5
425	2000-02-03 00:00:00.000000	02/03/2000	3	3rd	Thursday	5	1	5
683	2000-02-04 00:00:00.000000	02/04/2000	4	4th	Friday	6	1	5
874	2000-02-05 00:00:00.000000	02/05/2000	5	5th	Saturday	7	1	6
21	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2	7
382	2000-03-01 00:00:00.000000	03/01/2000	1	1st	Wednesday	4	1	9
426	2000-03-02 00:00:00.000000	03/02/2000	2	2nd	Thursday	5	1	9
684	2000-03-03 00:00:00.000000	03/03/2000	3	3rd	Friday	6	1	9
875	2000-03-04 00:00:00.000000	03/04/2000	4	4th	Saturday	7	1	10
6	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3	11
876	2000-04-01 00:00:00.000000	04/01/2000	1	1st	Saturday	7	1	14
967	2000-04-29 00:00:00.000000	04/29/2000	29	29th	Saturday	7	5	18
141	2000-04-30 00:00:00.000000	04/30/2000	30	30th	Sunday	1	5	18
214	2000-05-01 00:00:00.000000	05/01/2000	1	1st	Monday	2	1	18
289	2000-05-02 00:00:00.000000	05/02/2000	2	2nd	Tuesday	3	1	18
389	2000-05-03 00:00:00.000000	05/03/2000	3	3rd	Wednesday	4	1	18
427	2000-05-04 00:00:00.000000	05/04/2000	4	4th	Thursday	5	1	18
685	2000-05-05 00:00:00.000000	05/05/2000	5	5th	Friday	6	1	18
877	2000-05-06 00:00:00.000000	05/06/2000	6	6th	Saturday	7	1	19
993	2000-05-29 00:00:00.000000	05/29/2000	29	29th	Monday	2	5	22

Similarly to the SQL query editor, the **Save as table** and **Save as view** options are also available. These features can be useful for reusing your queries or for creating more complex queries based on the results of previous queries.

To learn more about SQL query editor, see [Query using the visual query editor](#).

5. Use client tools to query a warehouse

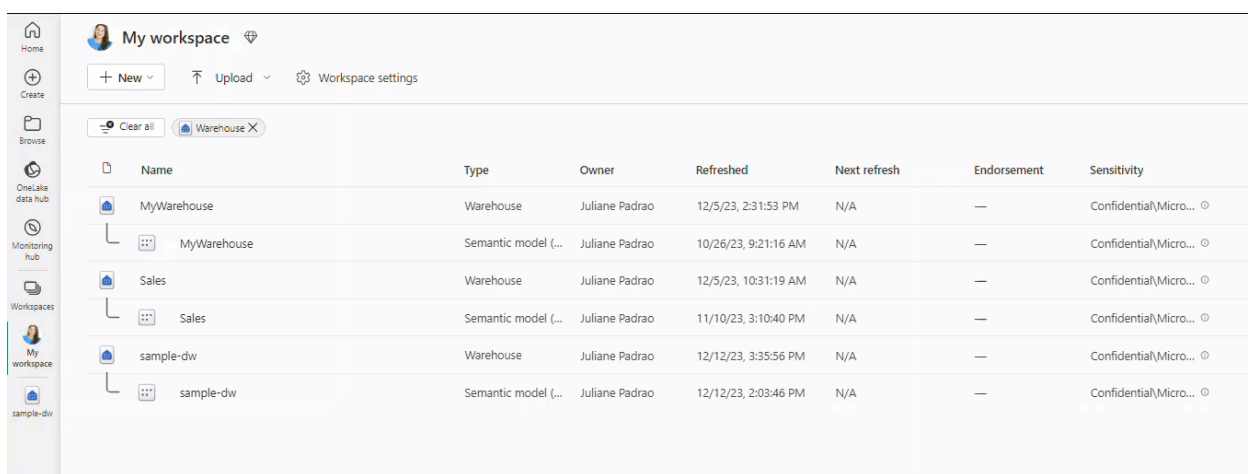
Use client tools to query a warehouse

Completed

Using SQL Server Management Studio (SSMS) to connect to a data warehouse in Fabric can facilitate your workflow, especially if you're already familiar with the tool.

Connect to your data warehouse

SQL Server Management Studio provides a familiar interface for those who regularly work with SQL Server.

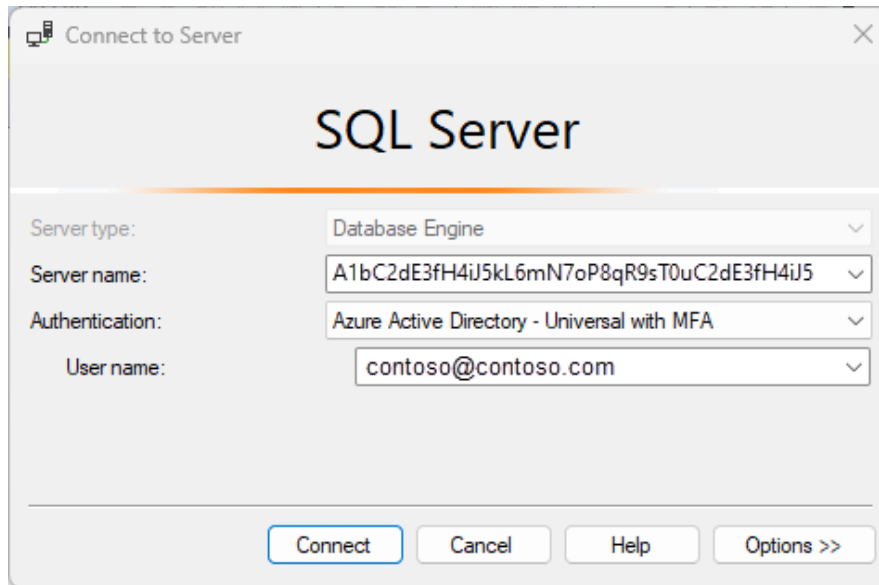


The screenshot shows the 'My workspace' interface in Microsoft Fabric. It features a sidebar with navigation icons for Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, and My workspace. The main area displays a table of data assets. The table has columns for Name, Type, Owner, Refreshed, Next refresh, Endorsement, and Sensitivity. The assets listed are MyWarehouse (Warehouse), MyWarehouse (Semantic model), Sales (Warehouse), Sales (Semantic model), sample-dw (Warehouse), and sample-dw (Semantic model). All assets are owned by Juliane Padrao.

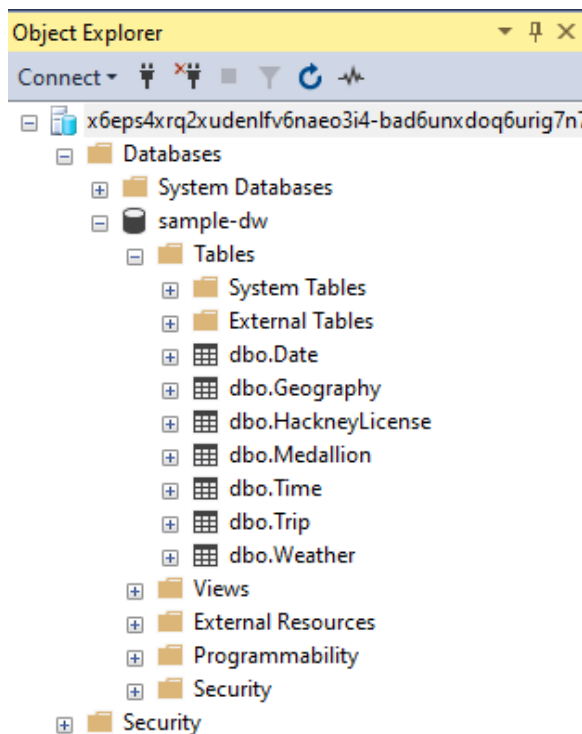
Name	Type	Owner	Refreshed	Next refresh	Endorsement	Sensitivity
MyWarehouse	Warehouse	Juliane Padrao	12/5/23, 2:31:53 PM	N/A	—	Confidential\Micro...
MyWarehouse	Semantic model (...)	Juliane Padrao	10/26/23, 9:21:16 AM	N/A	—	Confidential\Micro...
Sales	Warehouse	Juliane Padrao	12/5/23, 10:31:19 AM	N/A	—	Confidential\Micro...
Sales	Semantic model (...)	Juliane Padrao	11/10/23, 3:10:40 PM	N/A	—	Confidential\Micro...
sample-dw	Warehouse	Juliane Padrao	12/12/23, 3:35:56 PM	N/A	—	Confidential\Micro...
sample-dw	Semantic model (...)	Juliane Padrao	12/12/23, 2:03:46 PM	N/A	—	Confidential\Micro...

Follow these steps to connect to data warehouse in Fabric from SSMS:

1. Navigate to your Microsoft Fabric workspace.
2. On your warehouse asset, select more options, then select **Copy SQL connection string**.
3. Launch SQL Server Management Studio, and paste the SQL connection string copied into the **Server** name box, and provide the appropriate credentials for authentication.



4. After establishing a connection, SSMS shows the connected warehouse, along with its corresponding tables and views, all ready for querying.



Authentication options

In Microsoft Fabric, two types of authenticated users are supported through the SQL connection string:

- Microsoft Entra ID (formerly Azure Active Directory) user principals, or user identities
- Microsoft Entra ID (formerly Azure Active Directory) service principals

Note

SQL authentication is not supported.

Other tools

Any third-party tool can use the SQL connection string via ODBC or OLE DB drivers to connect to a Microsoft Fabric Warehouse or SQL analytics endpoint, using Microsoft Entra ID (formerly Azure Active Directory) authentication.

6. Exercise: Query a data warehouse in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/6-exercise-query-data-warehouse-microsoft-fabric>

Exercise: Query a data warehouse in Microsoft Fabric

Completed

Now it's your chance to query data in a data warehouse in Microsoft Fabric.

In this exercise, you'll learn how to query data using the SQL editor 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](#)

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/8-summary>

Summary

Completed

Learning how to query a data warehouse empowers individuals to extract, analyze, and interpret large volumes of stored data, transforming raw data into meaningful insights. These insights can drive strategic decision-making, optimize operations, and uncover hidden patterns and trends.

There's no one-size-fits-all solution for querying your data. The best approach depends on the specifics of your data warehouse and the question you're trying to answer.

For additional reading, you can refer to the following URLs:

- [Connectivity to data warehousing in Microsoft Fabric](#)
- [Query the SQL analytics endpoint or Warehouse in Microsoft Fabric](#)
- [View data in the Data preview in Microsoft Fabric](#)

Monitor a Microsoft Fabric data warehouse

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/01-introduction>

Introduction

Completed

A data warehouse is often central to enterprise analysis and reporting, and as such is a critical business asset. It's important to monitor a data warehouse to track and manage costs, identify and resolve query performance issues, and to gain insights into how your data is being used.

In this module, we'll explore some tools and techniques you can use to monitor your Microsoft Fabric data warehouse.

2. Monitor capacity metrics

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/02-capacity-metrics>

Monitor capacity metrics

Completed

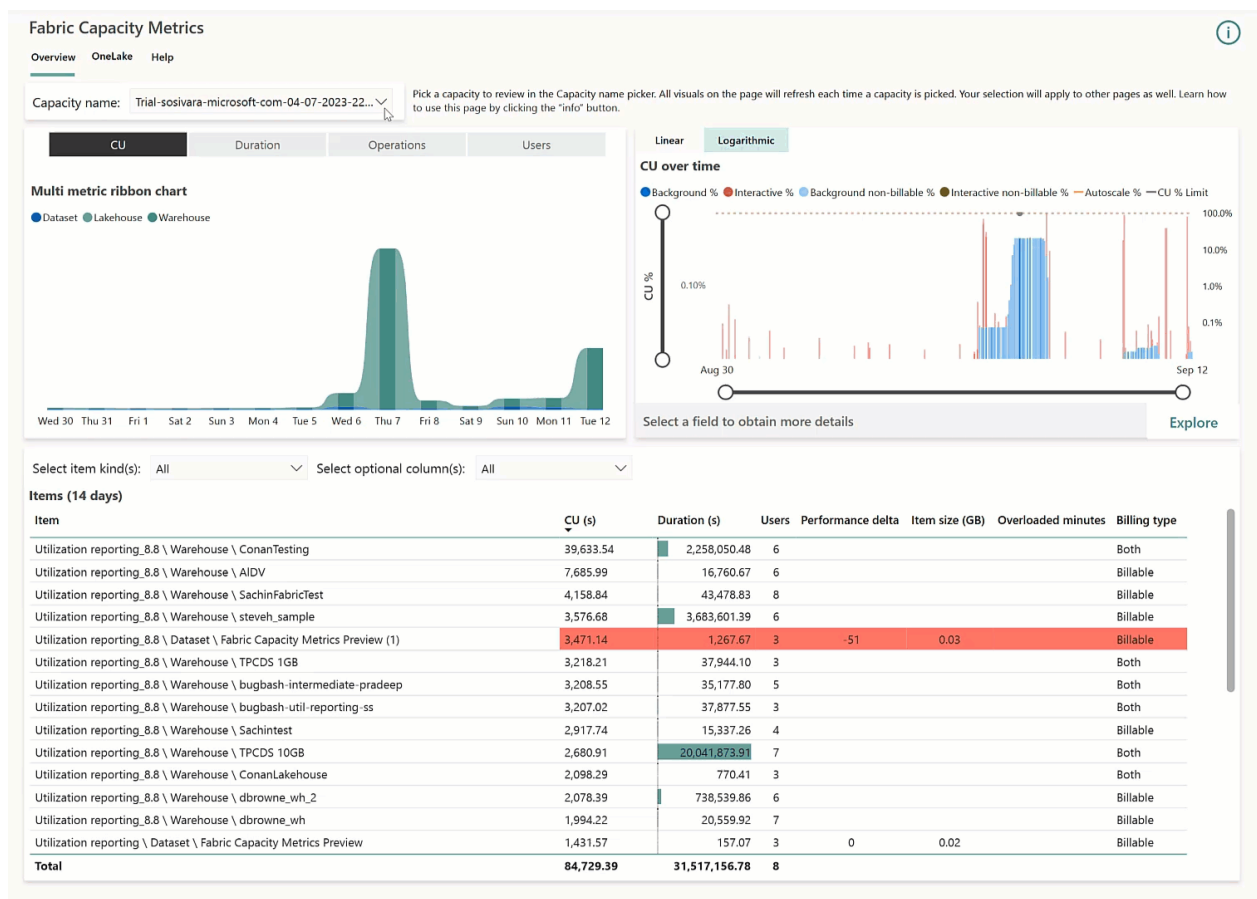
When your organization uses Microsoft Fabric, the license used to purchase the service determines the *capacity* available. A capacity is a pool of resources that you can use to implement Fabric capabilities.

The cost of using Fabric is based on *capacity units* (CUs). Each action you perform in a Fabric resource can consume CUs, for which your organization is billed. It's therefore important to be able to monitor capacity usage to plan and manage costs. In data warehouse workloads, CUs are consumed by data read and write activities, so queries in your data warehouse and the underlying file operations to OneLake storage are a significant factor in the cost of your Fabric analytics solution.

Using the Microsoft Fabric Capacity Metrics app

The Microsoft Fabric Capacity Metrics app is an app that an administrator can install in a Fabric environment and use to monitor capacity utilization. To monitor capacity utilization related to data warehousing, you can

filter the interface to show only **warehouse** activity, like this:



By using the Fabric Capacity Metrics app, you can observe capacity utilization trends to determine what processes are consuming CUs in your Fabric environment and whether any *throttling* is occurring (which indicates that your processes require more capacity than is available within the constraints of your purchased capacity license). With this information, you can optimize your capacity license for your needs.

Tip

For more information about Microsoft Fabric Capacity Metrics app, refer to [Billing and utilization reporting in Data Warehouse](#) in the Microsoft Fabric documentation.

3. Monitor current activity

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/03-dynamic-management-views>

Monitor current activity

Completed

You can use dynamic management views (DMVs) to retrieve information about the current state of the data warehouse. Specifically, Microsoft Fabric data warehouses include the following DMVs:

- **sys.dm_exec_connections:** Returns information about data warehouse connections.
- **sys.dm_exec_sessions:** Returns information about authenticated sessions.
- **sys.dm_exec_requests:** Returns information about active requests.

The schema of these tables is shown here:

sys.dm_exec_sessions	
session_id	
login_time	
host_name	
program_name	
host_process_id	
client_version	
client_interface_name	
security_id	
login_name	
nt_domain	
nt_user_name	
status	
context_info	
cpu_time	
memory_usage	
total_scheduled_time	
total_elapsed_time	
endpoint_id	
last_request_start_time	
last_request_end_time	
reads	
writes	
logical_reads	
is_user_process	
text_size	
language	
date_format	
date_first	
quoted_identifier	
arithabort	
ansi_null_dflt_on	
ansi_defaults	
ansi_warnings	
ansi_padding	
ansi_nulls	
concat_null_yields_null	
transaction_isolation_level	
lock_timeout	
deadlock_priority	
row_count	
prev_error	
original_security_id	
original_login_name	
last_successful_logon	
last_unsuccessful_logon	
unsuccessful_logons	
group_id	
database_id	
authenticating_database_id	
open_transaction_count	
is_filtered	
page_server_reads	
contained_availability_group_id	

sys.dm_exec_connections	
session_id	
most_recent_session_id	
connect_time	
net_transport	
protocol_type	
protocol_version	
endpoint_id	
encrypt_option	
auth_scheme	
node_affinity	
num_reads	
num_writes	
last_read	
last_write	
net_packet_size	
client_net_address	
client_tcp_port	
local_net_address	
local_tcp_port	
connection_id	
parent_connection_id	
most_recent_sql_handle	

sys.dm_exec_requests	
session_id	
request_id	
start_time	
status	
command	
sql_handle	
statement_start_offset	
statement_end_offset	
plan_handle	
database_id	
user_id	
connection_id	
blocking_session_id	
wait_type	
wait_time	
last_wait_type	
wait_resource	
open_transaction_count	
open_resultset_count	
transaction_id	
context_info	
percent_complete	
estimated_completion_time	
cpu_time	
total_elapsed_time	
scheduler_id	
task_address	
reads	
writes	
logical_reads	
text_size	
language	
date_format	
date_first	
quoted_identifier	
arithabort	
ansi_null_dflt_on	
ansi_defaults	
ansi_warnings	
ansi_padding	
ansi_nulls	
concat_null_yields_null	
transaction_isolation_level	
lock_timeout	
deadlock_priority	
row_count	
prev_error	
nest_level	
granted_query_memory	
executing_managed_code	
group_id	
query_hash	
query_plan_hash	
statement_sql_handle	
statement_context_id	
dop	
parallel_worker_count	
external_script_request_id	
is_resumable	
page_resource	
page_server_reads	
dist_statement_id	

Querying DMVs

You can retrieve detailed information about current activities in the data warehouse by querying the `dm_exec-*` DMVs. For example, consider the following query:

```
SELECT sessions.session_id, sessions.login_name,
       connections.client_net_address,
       requests.command, requests.start_time, requests.total_elapsed_time
FROM sys.dm_exec_connections AS connections
INNER JOIN sys.dm_exec_sessions AS sessions
    ON connections.session_id=sessions.session_id
INNER JOIN sys.dm_exec_requests AS requests
    ON requests.session_id = sessions.session_id
WHERE requests.status = 'running'
    AND requests.database_id = DB_ID()
ORDER BY requests.total_elapsed_time DESC
```

This query returns details about the active requests in the current database, ordered by the duration for which they have been executing; which may be useful to identify long-running queries that could benefit from optimization. An example result set from the query is shown here:

session_id	login_name	client_net_address	command	start_time	total_elapsed_time
60	fred@contoso.com	10.23.139.162	SELECT	2023-12-07T14:56:41.3530000	57266
126	nandita@contoso.com	10.23.137.98	SELECT	2023-12-07T14:57:22.7800000	15840
137	zoe@contoso.com	10.23.119.171	SELECT	2023-12-07T14:57:38.6070000	4

Tip

For more information about using DMVs, refer to [Monitor connections, sessions, and requests using DMVs](#) in the Microsoft Fabric documentation.

4. Monitor queries

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/04-query-insights>

Monitor queries

Completed

Microsoft Fabric data warehouses include *query insights* feature that provides historical, aggregated information about the queries that have been run; enabling you to identify frequently used or long-running queries, and helping you analyze and tune query performance.

Query insights are provided through the following views:

- **queryinsights.exec_requests_history**: Details of each completed SQL query.
- **queryinsights.long_running_queries**: Details of query execution time.
- **queryinsights.frequently_run_queries**: Details of frequently run queries.

The schema for these tables is shown here:

queryinsights.exec_requests_history	queryinsights.long_running_queries	queryinsights.frequently_run_queries
batch_id	last_dist_statement_id	avg_total_elapsed_time_ms
command	last_run_command	last_dist_statement_id
connection_id	last_run_session_id	last_run_command
distributed_statement_id	last_run_start_time	last_run_start_time
end_time	last_run_total_elapsed_time_ms	last_run_total_elapsed_time_ms
login_name	median_total_elapsed_time_ms	max_run_total_elapsed_time_ms
program_name	number_of_runs	min_run_total_elapsed_time_ms
query_hash		number_of_canceled_runs
root_batch_id		number_of_failed_runs
row_count		number_of_runs
session_id		number_of_successful_runs
start_time		
status		
total_elapsed_time_ms		

Retrieving query insights

The query insights views are a useful source of information about the queries that are being run in your data warehouse.

For example, consider the following query:

```
SELECT start_time, login_name, command
FROM queryinsights.exec_requests_history
WHERE start_time >= DATEADD(MINUTE, -60, GETUTCDATE())
```

This query uses the **queryinsights.exec_requests_history** view to identify queries that were run in the previous hour.

Note

Depending on the concurrent workloads, queries may take up to 15 minutes to be reflected in the query insights views.

You can get details of long-running queries from the **queryinsights.long_running_queries** view like this:

```
SELECT last_run_command, number_of_runs, median_total_elapsed_time_ms, last_run_start_time
FROM queryinsights.long_running_queries
WHERE number_of_runs > 1
ORDER BY median_total_elapsed_time_ms DESC;
```

This query identifies long-running SQL commands that have been used more than once and returns them in descending order of their median time to complete.

Note

To enable the views to provide aggregated metrics, queries with predicates are parameterized and considered the same query if the parameterized statements are an exact match. For example, the following queries would be considered to be the same command:

```
SELECT * FROM sales WHERE orderdate > '01/01/2023'
```

```
SELECT * FROM sales WHERE orderdate > '12/31/2021'
```

To find commonly used queries, you can use the **queryinsights.frequently_run_queries** view, like this:

```
SELECT last_run_command, number_of_runs, number_of_successful_runs, number_of_failed_runs  
FROM queryinsights.frequently_run_queries  
ORDER BY number_of_runs DESC;
```

This query returns details of successful and failed runs for frequently run commands.

Tip

For more information about using query insights refer to [Query insights in Fabric data warehousing](#) in the Microsoft Fabric documentation.

5. Exercise - Monitor a data warehouse in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/05-exercise-monitor>

Exercise - Monitor a data warehouse in Microsoft Fabric

Completed

Now it's time to try monitoring a Microsoft Fabric data warehouse for yourself.

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/monitor-fabric-data-warehouse/06-knowledge-check>

Module assessment

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/monitor-fabric-data-warehouse/07-summary>

Summary

Completed

Microsoft Fabric data warehouses are an important asset in any organization, and it's important to monitor them to track and manage costs, identify and resolve query performance issues, and to gain insights into how your data is being used.

This module explored the following ways in which you can monitor a data warehouse:

- The Microsoft Fabric Capacity Metrics app
- Dynamic Management Views
- Query Insights views

Secure a Microsoft Fabric data warehouse

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/1-introduction>

Introduction

Completed

Microsoft Fabric Data Warehouse is a complete platform for data, analytics, and AI (Artificial Intelligence). It refers to the process of storing, organizing, and managing large volumes of structured and semi-structured data.

In a warehouse, administrators have access to a suite of technologies 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 Warehouse and SQL analytics endpoints. This ensures a smooth and secure user experience, with no need for alterations to the existing applications.

Understand security features

Data engineers who are often well-versed in the SQL engine and adept at using T-SQL, will find warehouses in Microsoft Fabric easy to use.

This is because warehouses are 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 warehouse 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 data warehousing operations within an organization.
- **Item permissions** – Individual warehouses can have item permissions assigned directly to them. The main intent behind assigning such permissions is to facilitate the sharing of the Warehouse for downstream use.
- **Data protection security** – For more precise control, you can use T-SQL to grant specific permissions to users. Warehouse supports a range of data protection features that enable administrators to shield sensitive data from unauthorized access. This 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. These features ensure data

protection across Warehouses and SQL analytics endpoints without necessitating changes to applications.

In the next units, we'll explore various ways of enabling security in a warehouse, and how they can facilitate the tasks of keeping your data warehouse workload protected.

2. Explore dynamic data masking

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/2-explore-dynamic-data-masking>

Explore dynamic data masking

Completed

Dynamic Data Masking (DDM) is a security feature that limits data exposure to nonprivileged users by obscuring sensitive information.

Dynamic data masking offers several key benefits that enhance the security and manageability of your data. One of the primary advantages is its real-time masking feature. When querying sensitive data, DDM applies dynamic masking to it in real time. This process means that the actual data is never exposed to unauthorized users, thus enhancing the security of your data. Furthermore, DDM is straightforward to implement. It doesn't require complex coding, making it accessible for users of all skill levels.

Another benefit of DDM is that the data in the database isn't changed when DDM is applied. Instead, the masking rules are applied to the query results. This benefit means that the actual data remains intact and secure, while nonprivileged users only see a masked version of the data.

Define masking rule

Dynamic data masking, which is set up at the column level, offers a suite of features including comprehensive and partial masking capabilities, along with a random masking function designed for numeric data.

Masking Type	Description	Use Case	Limitations	Masking Rule
Default	Full masking according to the data types of the designated fields.	Useful when you want to completely hide the actual data.	Completely mask the data. No information is visible.	<code>default()</code>
Email	Exposes the first letter of an email address and the constant suffix ".com"	Useful when you want to show that the data field contains an email without revealing the actual email.	Only applicable to email fields.	<code>email()</code>

Masking Type	Description	Use Case	Limitations	Masking Rule
Custom Text	Exposes the first and last 'n' characters and adds a custom padding string in the middle.	Useful when you want to partially hide the actual data.	Not suitable for numeric, date, and time data types.	<code>partial(prefix_padding, padding_string, suffix_padding)</code>
Random	Replaces any numeric or binary value with a random number within a specified range.	Useful when you want to hide the actual numeric or binary data.	Only applicable to numeric and binary data types.	<code>random(low, high)</code>

The `prefix_padding` and `suffix_padding` parameters in the `partial()` function specify the number of characters to expose at the beginning and end of the string, and the `padding_string` parameter specifies the string to use for masking the remaining characters.

The `low` and `high` parameters in the `random()` function specify the range of random numbers to generate.

These masking types help prevent unauthorized viewing of sensitive data by enabling administrators to specify how much sensitive data to reveal, with minimal effect on the application layer. They're applied to query results, so the data in the database isn't changed. This approach allows many applications to mask sensitive data without modifying existing queries.

Configure data masking

Let's consider an example of a warehouse that stores customer information. The warehouse contains a `Customer` table with fields such as `CustomerName`, `Email`, `PhoneNumber`, and `CreditCardNumber`.

To apply data masking on the `CustomerName`, `Email`, `PhoneNumber`, and `CreditCardNumber` columns, run the following command:

```
-- For Email
ALTER TABLE Customers
ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()');

-- For PhoneNumber
ALTER TABLE Customers
ALTER COLUMN PhoneNumber ADD MASKED WITH (FUNCTION = 'partial(0,"XXX-XXX-",4)');

-- For CreditCardNumber
ALTER TABLE Customers
ALTER COLUMN CreditCardNumber ADD MASKED WITH (FUNCTION = 'partial(0,"XXXX-XXXX-XXXX-",4)');
```

View masked results

Without Dynamic Data Masking, if a nonprivileged user runs a query to fetch customer details, they might see something like this:

```
CustomerName: John Doe
Email: johndoe@contoso.com
PhoneNumber: 123-456-7890
CreditCardNumber: 1234-5678-9012-3456
```

However, with DDM applied to the `Email`, `PhoneNumber`, and `CreditCardNumber` fields, the same query would return:

```
CustomerName: John Doe
Email: j*****@contoso.com
PhoneNumber: XXX-XXX-7890
CreditCardNumber: XXXX-XXXX-XXXX-3456
```

As you can see, the sensitive data is hidden from the nonprivileged user, enhancing the security of your data. This scenario is a basic example of how Dynamic Data Masking works. It helps to ensure that sensitive data isn't exposed to users who don't have the necessary privileges to view it.

Note

Unprivileged users with query permissions can infer the actual data since the data isn't physically obfuscated.

DDM should be used as part of a comprehensive data security strategy that includes proper management of object-level security with SQL granular permissions and adherence to the principle of minimal required permissions.

3. Implement row-level security

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/3-implement-row-level-security>

Implement row-level security

Completed

Row-Level Security (RLS) is a feature that provides granular control over access to rows in a table based on group membership or execution context.

For example, in an e-commerce platform, you can ensure that sellers only have access to order rows that are related to their own products. This way, each seller can manage their orders independently, while maintaining the privacy of other sellers' order information.

If you have experience with SQL Server, you find that row-level security shares similar characteristics and features.

Protect your data

Row-Level Security (RLS) works by associating a function, known as a security predicate, with a table. This function is defined to return *true* or *false* based on certain conditions, typically involving the values of one or more columns in the table. When a user attempts to access data in the table, the security predicate function is invoked. If the function returns *true*, the row is accessible to the user; if it returns *false*, the row doesn't show up in the query results.

Depending on the business requirements, an RLS predicate can be as simple as `WHERE CustomerId = 29` or as complex as required.

This process is transparent to the user and is enforced automatically by SQL Server, ensuring consistent application of security rules.

Row-level security is implemented in two main steps:

- **Filter predicates** - It's an inline table-valued function that filters the results based on the predicate defined.

Access	Definition
SELECT	Can't view rows that are filtered.
UPDATE	Can't update rows that are filtered.
DELETE	Can't delete rows that are filtered.
INSERT	Not applicable.

- **Security policy** - It's a security policy that invokes an inline table-valued function to protect access to the rows in a table.

Because access control is configured and applied at the warehouse level, application changes are minimal - if any. Also, users can directly have access to the tables and can query their own data.

Configure row-level security

The T-SQL commands below demonstrate how to use RLS in a scenario where user access is segregated by tenant:

```
-- Create supporting objects for this example
CREATE TABLE [Sales] (SalesID INT,
    ProductID INT,
    TenantName NVARCHAR(50),
    OrderQty INT,
    UnitPrice MONEY)
GO

INSERT INTO [Sales] VALUES (1, 3, 'tenant1@contoso.com', 5, 10.00);
INSERT INTO [Sales] VALUES (2, 4, 'tenant2@contoso.com', 2, 57.00);
INSERT INTO [Sales] VALUES (3, 7, 'tenant3@contoso.com', 4, 23.00);
INSERT INTO [Sales] VALUES (4, 2, 'tenant4@contoso.com', 2, 91.00);
INSERT INTO [Sales] VALUES (5, 9, 'tenant5@contoso.com', 5, 80.00);
```

```
-- View all the rows in the table
SELECT * FROM Sales;
```

Next, we create a new schema, an inline table-valued function, and grant user access to the new function. The `WHERE @TenantName = USER_NAME() OR USER_NAME() = 'TenantAdmin'` predicate evaluates if the user name executing the query matches the *TenantName* column values.

```
--Create a schema
CREATE SCHEMA [Sec];
GO

--Create the filter predicate
CREATE FUNCTION sec.tvf_SecurityPredicatebyTenant(@TenantName AS NVARCHAR(10))
RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN      SELECT 1 AS result
                  WHERE @TenantName = USER_NAME() OR USER_NAME() = 'tenantAdmin@contoso'
GO

--Create security policy and add the filter predicate
CREATE SECURITY POLICY sec.SalesPolicy
ADD FILTER PREDICATE sec.tvf_SecurityPredicatebyTenant(TenantName) ON [dbo].[Sales]
WITH (STATE = ON);
GO
```

The *tenantAdmin@contoso.com* user should see all the rows. The *tenant1@contoso.com* to *tenant5@contoso.com* users should only see their own rows.

If you alter the security policy with `WITH (STATE = OFF);`, you notice that users see all the rows.

Note

There is a risk of information leakage if an attacker writes a query with a specially crafted `WHERE` clause and, for example, a divide-by-zero error, to force an exception if the `WHERE` condition is true. This is known as a *side-channel attack*.

Explore use cases

Row-level security is ideal for many scenarios, including:

- When you need to isolate departmental access at the row level.
- When you need to restrict customers' data access to only the data relevant to their company.
- When you need to restrict access for compliance purposes.

Apply best practices

Here are a few best practices to consider when implementing RLS:

- It's recommended to create a separate schema for predicate functions, and security policies.
- Whenever possible, avoid type conversions in predicate functions.

- To maximize performance, avoid using excessive table joins and recursion in predicate functions.

4. Implement column-level security

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/4-implement-column-level-security>

Implement column-level security

Completed

Column-level security (CLS) allows you to restrict column access in order to protect sensitive data. It provides granular control over who can access specific pieces of data, enhancing the overall security of your data warehouse.

Secure sensitive data

Let's consider a practical example of column-level security (CLS) in the healthcare industry. Suppose we have a table named `Patients` with the following columns: `PatientID`, `Name`, `Address`, `DateOfBirth`, and `MedicalHistory`.

The `MedicalHistory` column contains sensitive health information about the patients. As per the healthcare regulations and privacy laws, this information should only be accessible to authorized medical personnel such as doctors or nurses.

Here's how you might implement column-level security in this scenario:

1. **Identify the sensitive columns:** In this case, the `MedicalHistory` column is identified as containing sensitive data.
2. **Define access roles:** Define roles such as `Doctor` and `Nurse` who are allowed to access the `MedicalHistory` column. Other roles such as `Receptionist` or `Patient` might be restricted from accessing this column.
3. **Assign roles to users:** Assign the appropriate roles to each user in the warehouse. For example, user `DrSmith` might be assigned the `Doctor` role, while user `JohnDoe` might be assigned the `Patient` role.
4. **Implement access control:** Restrict access to the `MedicalHistory` column based on the user's role.

Column-level security can help ensure that sensitive health information is only accessible to those who are authorized to see it, though protecting patient privacy and complying with healthcare regulations.

Configure column-level security

In the scenario we've recently explored, the syntax to use for implementing column-level security might look as follows:

```
-- Create roles
CREATE ROLE Doctor AUTHORIZATION dbo;
CREATE ROLE Nurse AUTHORIZATION dbo;
CREATE ROLE Receptionist AUTHORIZATION dbo;
CREATE ROLE Patient AUTHORIZATION dbo;
GO

-- Grant SELECT on all columns to all roles
GRANT SELECT ON dbo.Patients TO Doctor;
GRANT SELECT ON dbo.Patients TO Nurse;
GRANT SELECT ON dbo.Patients TO Receptionist;
GRANT SELECT ON dbo.Patients TO Patient;
GO

-- Deny SELECT on the MedicalHistory column to the Receptionist and Patient roles
DENY SELECT ON dbo.Patients (MedicalHistory) TO Receptionist;
DENY SELECT ON dbo.Patients (MedicalHistory) TO Patient;
GO
```

In this example, we first create the roles `Doctor`, `Nurse`, `Receptionist`, and `Patient`. We then grant `SELECT` permissions on all columns in the `Patients` table to all roles. Finally, we deny `SELECT` permissions on the `MedicalHistory` column to the `Receptionist` and `Patient` roles. This ensures that only users with the `Doctor` or `Nurse` role can access the `MedicalHistory` column.

Understand the benefits

In the realm of warehouse security, two commonly used techniques are column-level security and views. Both methods serve to restrict access to sensitive data, but they do so in different ways and offer different advantages. The following table provides a comparative analysis of these two techniques across various aspects such as granularity of access control, maintenance, performance, transparency, and flexibility.

This comparison can help you understand the strengths and weaknesses of each method and guide you in choosing the most suitable approach for your specific application requirements.

Aspect	Column-Level Security	Views
Granularity of Access Control	Allows control at a more granular level. Can specify different access rights for different users or roles on different columns within the same table.	Would need to create different views for different sets of permissions.
Maintenance	Permissions are tied to the columns themselves, so they automatically adapt to changes in table structure.	If the underlying table structure changes, views may need to be updated to reflect these changes.
Performance	Generally more efficient because it operates directly on the table data.	Can introduce performance overhead, especially if they are complex or if the underlying tables are large.
Transparency	The restrictions are transparent to the user. The user queries the table as usual, and the database engine	The user needs to query a different object (the view instead of the table).

Aspect	Column-Level Security	Views
	takes care of applying the security rules.	
Flexibility	Less flexible than views.	Very flexible and can provide row-level security (by including a <code>WHERE</code> clause in the view definition) in addition to column-level security. They can also transform the data (for example, by calculating derived columns) which is not possible with column-level security.

The choice between using column-level security or views will depend on the specific requirements of your application. Always make sure to test any security changes in a safe environment before applying them to a production warehouse.

5. Configure SQL granular permissions using T-SQL

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/5-configure-sql-granular-permissions>

Configure SQL granular permissions using T-SQL

Completed

If you're familiar with relational databases and enterprise warehouses, it's common knowledge that there are four fundamental permissions governing [Data Manipulation Language \(DML\)](#) operations. These permissions, namely `SELECT`, `INSERT`, `UPDATE`, and `DELETE`, are universally applicable across all database platforms.

All of these permissions can be granted, revoked or denied on tables and views. If a permission is granted using the `GRANT` statement, then the permission is given to the user or role referenced in the `GRANT` statement. Users can also be denied permissions using the `DENY` command. If a user is granted a permission and denied the same permission, the `DENY` will always supersede the grant, and the user will be denied access to the specific object.

Table and view permissions

Tables and views represent the objects on which permissions can be granted within a warehouse. Within those tables and views, you can additionally restrict the columns that are accessible to a given security principal.

Permission	Definition
<code>SELECT</code>	Allows the user to view the data within the object (table or view). When denied, the user will be prevented from viewing the data within the object.
<code>INSERT</code>	Allows the user to insert data into the object. When denied, the user will be prevented from inserting data into the object.

Permission	Definition
UPDATE	Allows the user the update data within the object. When denied, the user will be prevented from updating data in the object.
DELETE	Allows the user to delete data within the object. When denied, the user will be prevented from deleting data from the object.

Function and stored procedure permissions

Like tables and views, functions and stored procedures have several permissions, which can be granted or denied.

Permission	Definition
ALTER	Grants the user the ability to change the definition of the object.
CONTROL	Grants the user all rights to the object.

Principle of least privilege

The basic idea of the principle of least privilege is that users and applications should only be given the permissions needed in order for them to complete the task. Applications should only have permissions that they need to do in order to complete the task at hand.

As an example, if an application accesses all data through stored procedures, then the application should only have the permission to execute the stored procedures, with no access to the tables.

Dynamic SQL

Dynamic SQL is a concept where a query is built programmatically. Dynamic SQL allows T-SQL statements to be generated within a stored procedure or a query itself. A simple example is shown below.

```
CREATE PROCEDURE sp_TopTenRows @tableName NVARCHAR(128)
AS
BEGIN
    DECLARE @query NVARCHAR(MAX);
    SET @query = N'SELECT TOP 10 * FROM ' + QUOTENAME(@tableName);
    EXEC sp_executesql @query;
END;
```

In this example, `@tableName` is the parameter that you can replace with the name of the table you want to inspect. The `QUOTENAME` function is used to safely quote the table name, preventing SQL injection attacks. The `sp_executesql` stored procedure is then used to execute the dynamically built query.

Please note that this is a simple example and real-world data warehouse tasks might require more complex dynamic SQL queries. Always be cautious when using dynamic SQL due to the risk of SQL injection attacks. Always use parameterization methods like `sp_executesql` or `QUOTENAME` to sanitize inputs.

6. Exercise: Secure a warehouse in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/secure-data-warehouse-in-microsoft-fabric/6-exercise-secure-data-warehouse>

Exercise: Secure a warehouse in Microsoft Fabric

Completed

Now it's your chance to secure a warehouse in Microsoft Fabric.

In this exercise, you'll learn how to secure a warehouse using the concepts explored in this module.

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.
- This exercise includes optional steps that require a second user account to view the applied security measures. If you're unable to create a second account, you can still complete the exercise using your own account. In that case, please refer to the screenshots provided at the end of each section to see the expected results.

How to create a second user for this exercise

If you're part of an organization that has an Entra or Microsoft 365 tenant:

Work with your identity administrator to create the second user either in [Entra](#) or the [Microsoft 365 admin center](#).

If you're *not* a member of an organization with an Entra or Microsoft 365 tenant:

1. You're unable to sign up for a Fabric trial with your personal email address. You can sign up for a [Microsoft 365 trial](#) and a new organizational tenant will be created and you'll become the User and Billing administrator of the tenant and can then create users.
2. Create the second user in the [Microsoft 365 admin center](#) See: [Add users](#)
3. Enable the Fabric trial using [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/secure-data-warehouse-in-microsoft-fabric/8-summary>

Summary

Completed

In this module, you learned about Dynamic Data Masking (DDM), Row-Level Security (RLS), Column-level security (CLS), and granular SQL permissions in Fabric warehouses.

The main takeaways from this module include understanding how DDM, RLS, and CLS work and their use cases. DDM operates at the column level, offering full, email, custom text, and random masking types. RLS works by associating a security predicate function with a table. CLS can be implemented by identifying sensitive columns, defining access roles, assigning roles to users, and implementing access control. In addition, you learned about the principle of least privilege, which suggests that users and applications should only be given the permissions necessary to complete their tasks.

For additional reading, you can refer to the following URLs:

- [Create a Warehouse in Microsoft Fabric](#)
- [Security for data warehousing in Microsoft Fabric](#)
- [Share your warehouse and manage permissions](#)