

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.

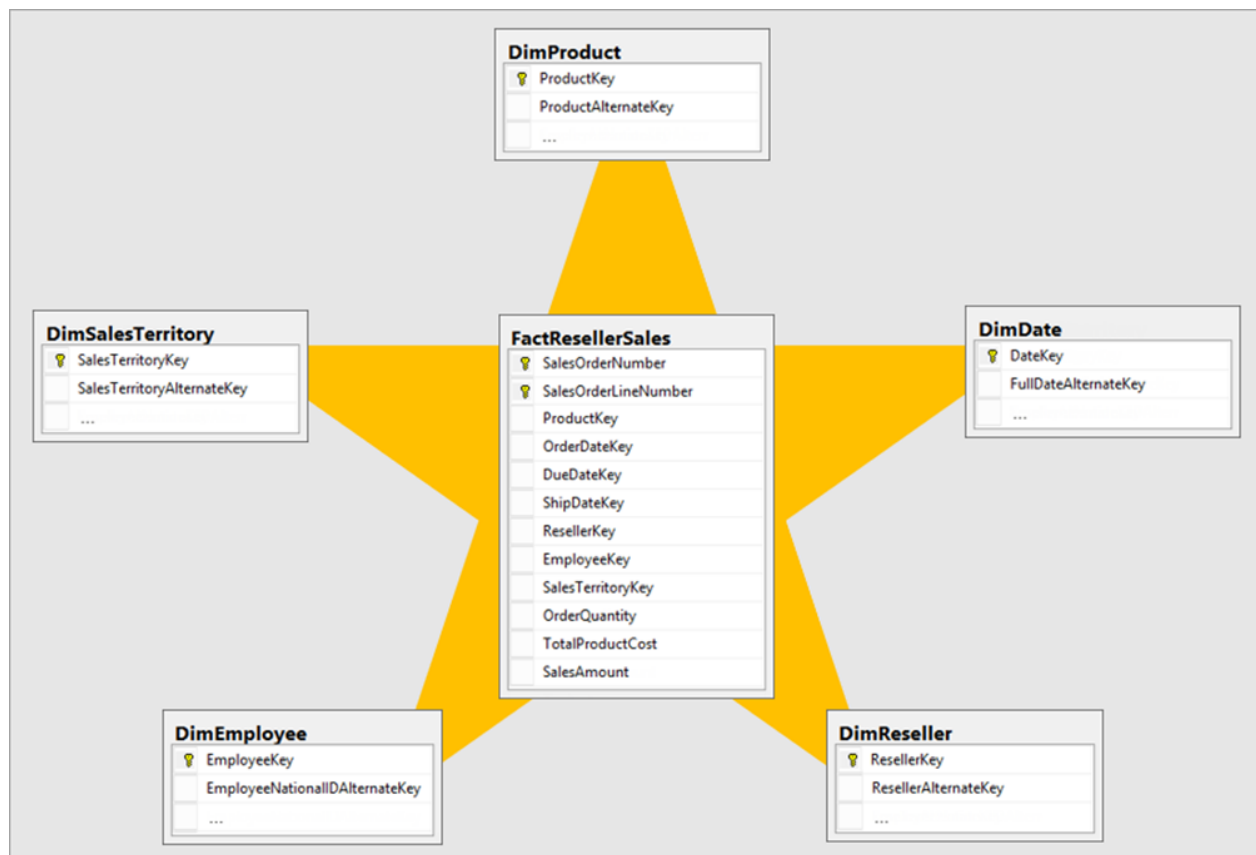
Data warehouse in Microsoft Fabric offers a rich set of features that make it easier to manage and analyze data. It includes advanced query processing capabilities, and supports the full transactional T-SQL capabilities like an enterprise data warehouse.

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
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

CalendarYear	ApproxOrders
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 shows the Power BI interface with the 'Home' menu at the top. In the 'Reporting' section, the 'New SQL query' button is highlighted with a red box and a red arrow labeled '1'. In the 'Explorer' pane on the left, the 'Queries' node is expanded, and the 'My queries' folder is highlighted with a red box and a red arrow labeled '2'. At the bottom of the interface, the 'Query' tab is selected, highlighted with a red box and a red arrow labeled '3'. The main area displays a 'Data preview' table with 1000 rows and 10 columns.

	123	DateID	Date	ABC DateKey	ABC DayOfMonth	ABC DaySuffix	ABC DayName	ABC DayOfWeek	ABC DayOfWeekInMon
1	20150317	2015-03-17 00:00:00.0000000	03/17/2015	17	17th	Tuesday	3	3	
2	20050317	2005-03-17 00:00:00.0000000	03/17/2005	17	17th	Thursday	5	3	
3	20010317	2001-03-17 00:00:00.0000000	03/17/2001	17	17th	Saturday	7	3	
4	20090317	2009-03-17 00:00:00.0000000	03/17/2009	17	17th	Tuesday	3	3	
5	20040317	2004-03-17 00:00:00.0000000	03/17/2004	17	17th	Wednesday	4	3	
6	20000317	2000-03-17 00:00:00.0000000	03/17/2000	17	17th	Friday	6	3	
7	20120317	2012-03-17 00:00:00.0000000	03/17/2012	17	17th	Saturday	7	3	
8	20060317	2006-03-17 00:00:00.0000000	03/17/2006	17	17th	Friday	6	3	
9	20110317	2011-03-17 00:00:00.0000000	03/17/2011	17	17th	Thursday	5	3	
10	20100317	2010-03-17 00:00:00.0000000	03/17/2010	17	17th	Wednesday	4	3	
11	20070317	2007-03-17 00:00:00.0000000	03/17/2007	17	17th	Saturday	7	3	
12	20140214	2014-02-14 00:00:00.0000000	02/14/2014	14	14th	Friday	6	2	
13	20030214	2003-02-14 00:00:00.0000000	02/14/2003	14	14th	Friday	6	2	
14	20080214	2008-02-14 00:00:00.0000000	02/14/2008	14	14th	Thursday	5	2	
15	20110214	2011-02-14 00:00:00.0000000	02/14/2011	14	14th	Monday	2	2	
16	20050214	2005-02-14 00:00:00.0000000	02/14/2005	14	14th	Monday	2	2	
17	20010214	2001-02-14 00:00:00.0000000	02/14/2001	14	14th	Wednesday	4	2	
18	20060214	2006-02-14 00:00:00.0000000	02/14/2006	14	14th	Tuesday	3	2	
19	20020214	2002-02-14 00:00:00.0000000	02/14/2002	14	14th	Thursday	5	2	
20	20120214	2012-02-14 00:00:00.0000000	02/14/2012	14	14th	Tuesday	3	2	
21	20000214	2000-02-14 00:00:00.0000000	02/14/2000	14	14th	Monday	2	2	
22	20100214	2010-02-14 00:00:00.0000000	02/14/2010	14	14th	Sunday	1	2	
23	20070214	2007-02-14 00:00:00.0000000	02/14/2007	14	14th	Wednesday	4	2	
24	20130214	2013-02-14 00:00:00.0000000	02/14/2013	14	14th	Thursday	5	2	
25	20091031	2009-10-31 00:00:00.0000000	10/31/2009	31	31st	Saturday	7	5	
26	20121031	2012-10-31 00:00:00.0000000	10/31/2012	31	31st	Wednesday	4	5	
27	20071031	2007-10-31 00:00:00.0000000	10/31/2007	31	31st	Wednesday	4	5	
28	20021031	2002-10-31 00:00:00.0000000	10/31/2002	31	31st	Thursday	5	5	

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 Azure Data Studio interface. On the left, the 'Explorer' pane shows a tree view of the database structure, including 'sample-dw', 'Schemas', 'dbo', 'Tables', 'Views', 'Functions', 'Stored Procedures', 'guest', 'INFORMATION_SCHEMA', 'queryinsights', 'sys', 'Security', and 'Queries'. The 'Queries' section is expanded, showing a list of queries. The main area displays a SQL query result. The query is a SELECT statement with columns: DateID, Date, ABC, DateKey, ABC, DayOfMonth, ABC, DaySuffix, ABC, DayName, ABC, DayOfWeek, and ABC, DayOfWeekInMonth. The table shows 29 rows of data. A 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	DayOfWeekInMonth
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	20080214	2008-02-14 00:00:00.000000	02/14/2008	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.

ID	DateID	Date	ABC DateKey	ABC DayOfMonth	ABC DaySuffix	ABC DayName	ABC DayOfWeek	ABC DayOfWeekMonth	ABC DayOfWeekYear	A
94	20000101	2000-01-01 00:00:00.000000	01/01/2000	1	1st	Saturday	7	1	1	1
960	20000129	2000-01-29 00:00:00.000000	01/29/2000	29	29th	Saturday	7	5	5	5
138	20000130	2000-01-30 00:00:00.000000	01/30/2000	30	30th	Sunday	1	5	5	5
108	20000131	2000-01-31 00:00:00.000000	01/31/2000	31	31st	Monday	2	5	5	5
283	20000201	2000-02-01 00:00:00.000000	02/01/2000	1	1st	Tuesday	3	1	5	5
381	20000202	2000-02-02 00:00:00.000000	02/02/2000	2	2nd	Wednesday	4	1	5	5
425	20000203	2000-02-03 00:00:00.000000	02/03/2000	3	3rd	Thursday	5	1	5	5
683	20000204	2000-02-04 00:00:00.000000	02/04/2000	4	4th	Friday	6	1	5	5
874	20000205	2000-02-05 00:00:00.000000	02/05/2000	5	5th	Saturday	7	1	6	6
21	20000214	2000-02-14 00:00:00.000000	02/14/2000	14	14th	Monday	2	2	7	7
382	20000301	2000-03-01 00:00:00.000000	03/01/2000	1	1st	Wednesday	4	1	9	9
426	20000302	2000-03-02 00:00:00.000000	03/02/2000	2	2nd	Thursday	5	1	9	9
684	20000303	2000-03-03 00:00:00.000000	03/03/2000	3	3rd	Friday	6	1	9	9
875	20000304	2000-03-04 00:00:00.000000	03/04/2000	4	4th	Saturday	7	1	10	1
6	20000317	2000-03-17 00:00:00.000000	03/17/2000	17	17th	Friday	6	3	11	1
876	20000401	2000-04-01 00:00:00.000000	04/01/2000	1	1st	Saturday	7	1	14	1
967	20000429	2000-04-29 00:00:00.000000	04/29/2000	29	29th	Saturday	7	5	18	5
141	20000430	2000-04-30 00:00:00.000000	04/30/2000	30	30th	Sunday	1	5	18	5
214	20000501	2000-05-01 00:00:00.000000	05/01/2000	1	1st	Monday	2	1	18	5
289	20000502	2000-05-02 00:00:00.000000	05/02/2000	2	2nd	Tuesday	3	1	18	5
389	20000503	2000-05-03 00:00:00.000000	05/03/2000	3	3rd	Wednesday	4	1	18	5
427	20000504	2000-05-04 00:00:00.000000	05/04/2000	4	4th	Thursday	5	1	18	5
685	20000505	2000-05-05 00:00:00.000000	05/05/2000	5	5th	Friday	6	1	18	5
877	20000506	2000-05-06 00:00:00.000000	05/06/2000	6	6th	Saturday	7	1	19	6
993	20000529	2000-05-29 00:00:00.000000	05/29/2000	29	29th	Monday	2	5	22	9

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

<https://learn.microsoft.com/en-us/training/modules/query-data-warehouse-microsoft-fabric/5-use-client-tools>

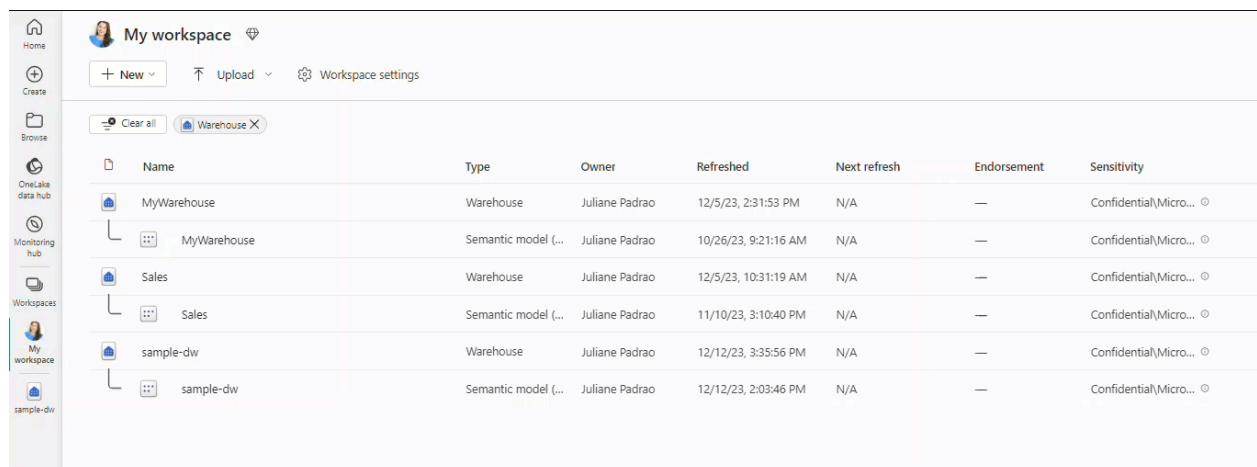
Use client tools to query a warehouse

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.

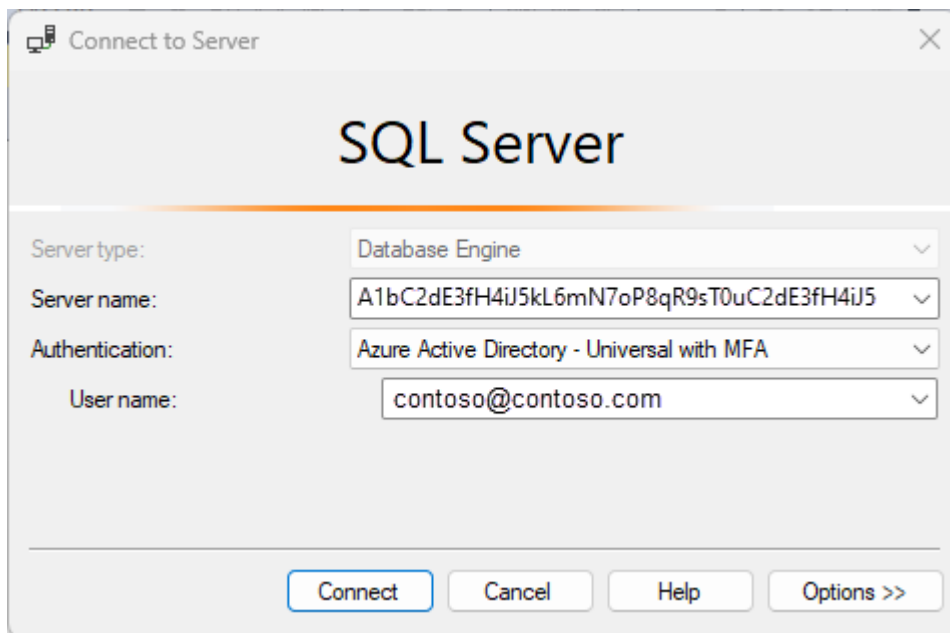


The screenshot shows the 'My workspace' interface in Microsoft Fabric. It features a sidebar with navigation icons for Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, My workspace, and sample-dw. The main area displays a table of assets with columns: Name, Type, Owner, Refreshed,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

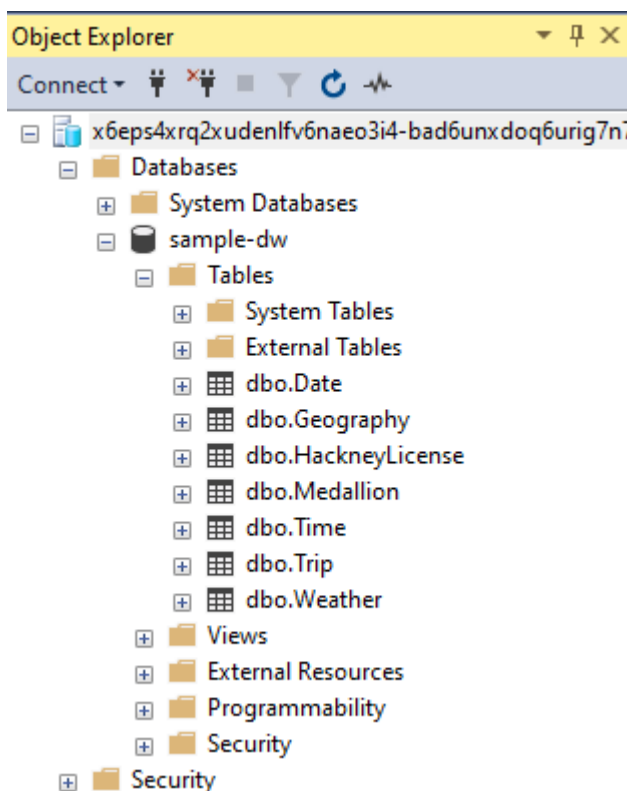
Name	Type	Owner	Refreshed	Next refresh	Endorsement	Sensitivity
MyWarehouse	Warehouse	Juliane Padrao	12/5/23, 2:31:53 PM	N/A	—	Confidential\Micro...
MyWarehouse	Semantic model (...)	Juliane Padrao	10/26/23, 9:21:16 AM	N/A	—	Confidential\Micro...
Sales	Warehouse	Juliane Padrao	12/5/23, 10:31:19 AM	N/A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

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](#)