

Contents

[Power Query documentation](#)

[What is Power Query?](#)

[Get data](#)

[Getting data](#)

[Authentication with a data source](#)

[SharePoint and OneDrive for Business files import](#)

[Lack of support connecting to Microsoft Graph](#)

[Connectivity without an existing connector](#)

[Transform data](#)

[Use Power Query to transform data](#)

[Using the Applied Steps list](#)

[Query folding basics](#)

[Query folding examples](#)

[Using the data profiling tools](#)

[Using the Queries pane](#)

[Diagram view](#)

[Using Schema view \(Preview\)](#)

[Share a query](#)

[Using custom functions](#)

[Transform table](#)

[Promote or demote column headers](#)

[Filter by row position](#)

[Filter by values](#)

[Choose or remove columns](#)

[Grouping or summarizing rows](#)

[Unpivot column](#)

[Pivot column](#)

[Transpose table](#)

[Reverse rows](#)

- [Data types](#)
- [Dealing with errors](#)
- [Working with duplicates](#)
- [Transform columns](#)
 - [Fill values in a column](#)
 - [Sort columns](#)
 - [Rename column](#)
 - [Move columns](#)
 - [Replace values](#)
 - [Parse text as JSON or XML](#)
- [Add columns](#)
 - [Add a column from examples](#)
 - [Add an index column](#)
 - [Add a custom column](#)
 - [Add a conditional column](#)
 - [Add a cluster values column](#)
- [Combine data](#)
 - [Append queries](#)
 - [Combine files](#)
 - [Overview](#)
 - [CSV documents](#)
 - [Merge queries](#)
 - [Overview](#)
 - [Left outer join](#)
 - [Right outer join](#)
 - [Full outer join](#)
 - [Inner join](#)
 - [Left anti join](#)
 - [Right anti join](#)
 - [Fuzzy merge](#)
 - [Cross join](#)
 - [Split columns](#)

[By delimiter](#)

[By number of characters](#)

[By positions](#)

[By lowercase to uppercase](#)

[By uppercase to lowercase](#)

[By digit to non-digit](#)

[By non-digit to digit](#)

Dataflows

[Overview of dataflows](#)

[Analytical vs. standard dataflows](#)

[Create and use dataflows](#)

[Create and use dataflows in Teams](#)

[Use dataflows in solutions](#)

[Use incremental refresh](#)

[Connect to data sources](#)

[Dataflow licenses](#)

[Migrate queries to dataflows](#)

[Use an on-premises data gateway](#)

[Analytical dataflows](#)

[Get data from dataflows](#)

[Create computed entities](#)

[Link entities between dataflows](#)

[Connect Azure Data Lake for dataflow storage](#)

[Storage structure](#)

[Common Data Model storage structure](#)

[Configure storage and compute](#)

[Best practices for dataflows](#)

[Scenarios for computed entities](#)

[Developing complex dataflows](#)

[Reusing dataflows](#)

[Dimensional model](#)

[Performance](#)

Get dataflows data from Azure services

Standard dataflows

Relationships and lookups

Field mapping

Security and Roles

Sync Excel data

Add data to a table in Dataverse

Integration

Data Factory dataflows

Power Automate templates for dataflows

Overview of Power Automate templates for dataflows

Send notification when a dataflow refresh completes

Open a ticket when a dataflow refresh fails

Trigger dataflows and Power BI datasets sequentially

Use Dataverse to build a dataflows monitoring report

Use Excel to build a dataflows monitoring report

Use a Power BI dataset to build a dataflows monitoring report

Troubleshooting dataflows

Creating dataflows

Getting data from dataflows

Connecting to the data source

Keyboard shortcuts

Best practices

Feedback and support

Advanced topics

Query folding

Fuzzy matching

Behind the scenes of the Data Privacy Firewall

Query Diagnostics

What is Query Diagnostics for Power Query?

Recording Query Diagnostics

Reading Query Diagnostics

- [Understanding folding with Query Diagnostics](#)
- [Why does my query run multiple times?](#)
- [Step Folding Indicators](#)
- [Query plan \(Preview\)](#)
- [Use Query Parameters](#)
- [Error handling](#)
- [Import data using native database query](#)
- [Create Power Platform dataflows from queries in Microsoft Excel \(Preview\)](#)
- [Optimize Power Query when expanding table columns](#)
- [Using Microsoft Edge for OAuth in Power BI Desktop](#)
- [Connector reference](#)
 - [List of all Power Query connectors](#)
 - [Access database](#)
 - [Adobe Analytics](#)
 - [Amazon Athena](#)
 - [Amazon Redshift](#)
 - [Anaplan](#)
 - [AssembleViews](#)
 - [Automy Data Analytics \(Beta\)](#)
 - [Azure Cosmos DB v2 \(Beta\)](#)
 - [Azure SQL database](#)
 - [Azure Synapse Analytics \(SQL DW\)](#)
 - [Azure Synapse Analytics workspace \(Beta\)](#)
 - [Bloomberg Data and Analytics](#)
 - [BQE Core \(Beta\)](#)
 - [Common Data Service \(Legacy\)](#)
 - [Data Lake Storage](#)
 - [Dataverse](#)
 - [Delta Sharing](#)
 - [EQuIS \(Beta\)](#)
 - [Essbase](#)
 - [Excel](#)

[FHIR](#)

[Overview](#)

[Authentication](#)

[Query folding](#)

[Query folding patterns](#)

[Data relationships](#)

[Folder](#)

[Google Analytics](#)

[Google BigQuery](#)

[Google Sheets \(Beta\)](#)

[Hive LLAP](#)

[IBM Db2 database](#)

[JSON](#)

[Mailchimp \(Deprecated\)](#)

[Microsoft Azure Consumption Insights \(Beta\) \(Deprecated\)](#)

[Microsoft Graph Security \(Deprecated\)](#)

[MySQL database](#)

[OData Feed](#)

[OData Feed connector](#)

[Connecting to Azure DevOps using OData](#)

[ODBC](#)

[Oracle database](#)

[PDF](#)

[PostgreSQL](#)

[QuickBooks Online \(Beta\)](#)

[Salesforce Objects](#)

[Salesforce Reports](#)

[SAP Business Warehouse](#)

[SAP Business Warehouse Application Server](#)

[SAP Business Warehouse Message Server](#)

[SAP BW fundamentals](#)

[Navigate the query objects](#)

- [Transform and filter an SAP BW dataset](#)
- [Implementation details](#)
- [Import vs. DirectQuery for SAP BW](#)
- [Windows authentication and single sign-on](#)
- [Use advanced options](#)
- [SAP BW connector troubleshooting](#)
- [SAP HANA database](#)
 - [Overview](#)
 - [SAP HANA encryption](#)
 - [Configure ODBC for SAP HANA](#)
 - [Troubleshooting](#)
- [SharePoint folder](#)
- [SharePoint list](#)
- [SharePoint Online list](#)
- [SIS-CC SDMX](#)
- [Snowflake](#)
- [SoftOne BI \(Beta\)](#)
- [SQL Server database](#)
- [Stripe \(Deprecated\)](#)
- [SumTotal \(Beta\)](#)
- [Text/CSV](#)
- [TIBCO\(R\) Data Virtualization](#)
- [Usercube](#)
- [Web](#)
 - [Web connector](#)
 - [Web by example](#)
 - [Troubleshooting](#)
- [Workforce Dimensions \(Beta\) \(Deprecated\)](#)
- [XML](#)
- [Zendesk \(Beta\)](#)
- [Support and troubleshooting](#)
- [Power Query online limits](#)

[Common issues](#)

[Review script changes](#)

[Connector support and feedback](#)

[Capture web requests with Fiddler](#)

[Create custom Power Query connectors](#)

[Quick starts](#)

[Install the Power Query SDK](#)

[Start developing custom connectors](#)

[Create your first connector - Hello World](#)

[Walkthroughs](#)

[TripPin walkthrough](#)

[Overview](#)

[1. OData](#)

[2. REST API](#)

[3. Navigation tables](#)

[4. Paths](#)

[5. Paging](#)

[6. Schemas](#)

[7. Advanced schemas](#)

[8. Diagnostics](#)

[9. Test connection](#)

[10. Folding](#)

[OAuth tutorials](#)

[GitHub](#)

[Samples](#)

[Functional samples](#)

[ODBC samples](#)

[TripPin samples](#)

[Advanced concepts](#)

[Additional connector functionality](#)

[Handling authentication](#)

[Handling data access](#)

ODBC development

[Overview](#)

[ODBC extensibility functions](#)

[Parameters for your Data Source function](#)

[Parameters for Odbc.DataSource](#)

[Troubleshooting and testing](#)

[Handling resource path](#)

[Handling paging](#)

[Handling transformations](#)

[Static](#)

[Dynamic](#)

[Handling schemas](#)

[Handling status codes](#)

[Default behavior](#)

[Wait retry pattern](#)

[Handling unit testing](#)

[Helper functions](#)

[Handling errors](#)

[Handling documentation](#)

[Handling navigation tables](#)

[Handling gateway support](#)

[Handling connector signing](#)

[Connector Certification](#)

[Overview](#)

[Submission](#)

[Custom connector documentation](#)

[Resources](#)

[Power BI documentation](#)

[M function reference](#)

[M language document](#)

[M type reference](#)

What is Power Query?

1/15/2022 • 6 minutes to read • [Edit Online](#)

Power Query is a data transformation and data preparation engine. Power Query comes with a graphical interface for getting data from sources and a Power Query Editor for applying transformations. Because the engine is available in many products and services, the destination where the data will be stored depends on where Power Query was used. Using Power Query, you can perform the extract, transform, and load (ETL) processing of data.

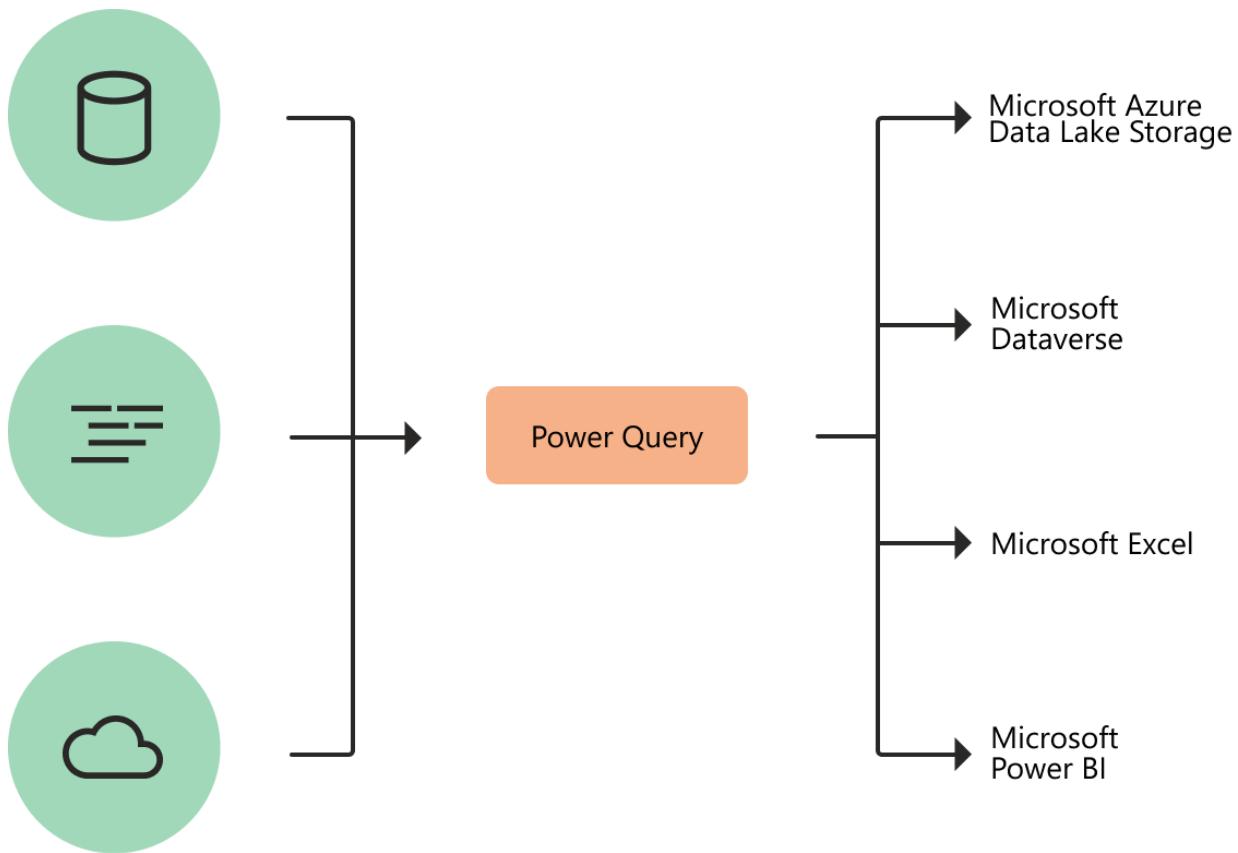


Diagram with symbolized data sources on the right, passing through Power Query for transformation, and then going to various destinations, such as Azure Data Lake Storage, Dataverse, Microsoft Excel, or Power BI.

How Power Query helps with data acquisition

Business users spend up to 80 percent of their time on data preparation, which delays the work of analysis and decision-making. Several challenges contribute to this situation, and Power Query helps address many of them.

EXISTING CHALLENGE	HOW DOES POWER QUERY HELP?
Finding and connecting to data is too difficult	Power Query enables connectivity to a wide range of data sources, including data of all sizes and shapes.
Experiences for data connectivity are too fragmented	Consistency of experience, and parity of query capabilities over all data sources.
Data often needs to be reshaped before consumption	Highly interactive and intuitive experience for rapidly and iteratively building queries over any data source, of any size.

EXISTING CHALLENGE	HOW DOES POWER QUERY HELP?
Any shaping is one-off and not repeatable	<p>When using Power Query to access and transform data, you define a repeatable process (query) that can be easily refreshed in the future to get up-to-date data.</p> <p>In the event that you need to modify the process or query to account for underlying data or schema changes, you can use the same interactive and intuitive experience you used when you initially defined the query.</p>
Volume (data sizes), velocity (rate of change), and variety (breadth of data sources and data shapes)	<p>Power Query offers the ability to work against a subset of the entire dataset to define the required data transformations, allowing you to easily filter down and transform your data to a manageable size.</p> <p>Power Query queries can be refreshed manually or by taking advantage of scheduled refresh capabilities in specific products (such as Power BI) or even programmatically (by using the Excel object model).</p> <p>Because Power Query provides connectivity to hundreds of data sources and over 350 different types of data transformations for each of these sources, you can work with data from any source and in any shape.</p>

Power Query experiences

The Power Query user experience is provided through the Power Query Editor user interface. The goal of this interface is to help you apply the transformations you need simply by interacting with a user-friendly set of ribbons, menus, buttons, and other interactive components.

The Power Query Editor is the primary data preparation experience, where you can connect to a wide range of data sources and apply hundreds of different data transformations by previewing data and selecting transformations from the UI. These data transformation capabilities are common across all data sources, whatever the underlying data source limitations.

When you create a new transformation step by interacting with the components of the Power Query interface, Power Query automatically creates the M code required to do the transformation so you don't need to write any code.

Currently, two Power Query experiences are available:

- **Power Query Online**—Found in integrations such as Power BI dataflows, Microsoft Power Platform dataflows, Azure Data Factory wrangling dataflows, and many more that provide the experience through an online webpage.
- **Power Query for Desktop**—Found in integrations such as Power Query for Excel and Power BI Desktop.

NOTE

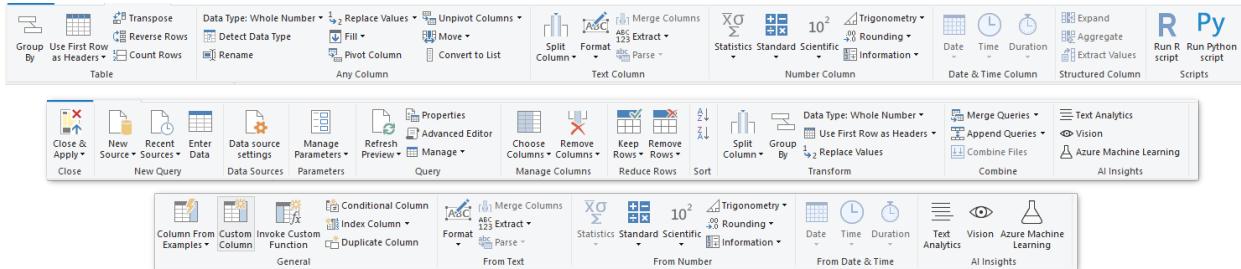
Although two Power Query experiences exist, they both provide almost the same user experience in every scenario.

Transformations

The transformation engine in Power Query includes many prebuilt transformation functions that can be used through the graphical interface of the Power Query Editor. These transformations can be as simple as removing a column or filtering rows, or as common as using the first row as a table header. There are also advanced transformation options such as merge, append, group by, pivot, and unpivot.

All these transformations are made possible by choosing the transformation option in the menu, and then

applying the options required for that transformation. The following illustration shows a few of the transformations available in Power Query Editor.



More information: [Quickstart: Using Power Query in Power BI](#)

Dataflows

Power Query can be used in many products, such as Power BI and Excel. However, using Power Query within a product limits its usage to only that specific product. *Dataflows* are a product-agnostic service version of the Power Query experience that runs in the cloud. Using dataflows, you can get data and transform data in the same way, but instead of sending the output to Power BI or Excel, you can store the output in other storage options such as Dataverse or Azure Data Lake Storage. This way, you can use the output of dataflows in other products and services.

More information: [What are dataflows?](#)

Power Query M formula language

In any data transformation scenario, there are some transformations that can't be done in the best way by using the graphical editor. Some of these transformations might require special configurations and settings that the graphical interface doesn't currently support. The Power Query engine uses a scripting language behind the scenes for all Power Query transformations: the Power Query M formula language, also known as M.

The M language is the data transformation language of Power Query. Anything that happens in the query is ultimately written in M. If you want to do advanced transformations using the Power Query engine, you can use the Advanced Editor to access the script of the query and modify it as you want. If you find that the user interface functions and transformations won't perform the exact changes you need, use the Advanced Editor and the M language to fine-tune your functions and transformations.

```

let
    Source = Exchange.Contents("xyz@contoso.com"),
    Mail1 = Source{[Name="Mail"]}[Data],
    #"Expanded Sender" = Table.ExpandRecordColumn(Mail1, "Sender", {"Name"}, {"Name"}),
    #"Filtered Rows" = Table.SelectRows(#"Expanded Sender", each ([HasAttachments] = true)),
    #"Filtered Rows1" = Table.SelectRows(#"Filtered Rows", each ([Subject] = "sample files for email PQ test") and ([Folder Path] = "\Inbox\")),
    #"Removed Other Columns" = Table.SelectColumns(#"Filtered Rows1", {"Attachments"}),
    #"Expanded Attachments" = Table.ExpandTableColumn(#"Removed Other Columns", "Attachments", {"Name", "AttachmentContent"}, {"Name", "AttachmentContent"}),
    #"Filtered Hidden Files1" = Table.SelectRows(#"Expanded Attachments", each [Attributes]?[Hidden]? <> true),
    #"Invoke Custom Function1" = Table.AddColumn(#"Filtered Hidden Files1", "Transform File from Mail", each
        #"Transform File from Mail"([AttachmentContent])),
    #"Removed Other Columns1" = Table.SelectColumns(#"Invoke Custom Function1", {"Transform File from Mail"}),
    #"Expanded Table Column1" = Table.ExpandTableColumn(#"Removed Other Columns1", "Transform File from Mail", Table.ColumnNames(#"Transform File from Mail"("#Sample File"))),
    #"Changed Type" = Table.TransformColumnTypes(#"Expanded Table Column1", [{"Column1", type text}, {"Column2", type text}, {"Column3", type text}, {"Column4", type text}, {"Column5", type text}, {"Column6", type text}, {"Column7", type text}, {"Column8", type text}, {"Column9", type text}, {"Column10", type text}])
in
    #"Changed Type"

```

More information: [Power Query M formula language](#)

Where can you use Power Query?

The following table lists Microsoft products and services where Power Query can be found.

PRODUCT	M ENGINE ¹	POWER QUERY DESKTOP ²	POWER QUERY ONLINE ³	DATAFLOWS ⁴
Excel for Windows	Yes	Yes	No	No
Excel for Mac	Yes	No	No	No
Power BI	Yes	Yes	Yes	Yes
Power Apps	Yes	No	Yes	Yes
Power Automate	Yes	No	Yes	No
Azure Data Factory	Yes	No	Yes	Yes
SQL Server Integration Services	Yes	No	No	No
SQL Server Analysis Services	Yes	Yes	No	No
Dynamics 365 Customer Insights	Yes	No	Yes	Yes

¹ M engine	The underlying query execution engine that runs queries expressed in the Power Query formula language ("M").
² Power Query Desktop	The Power Query experience found in desktop applications.
³ Power Query Online	The Power Query experience found in web browser applications.
⁴ Dataflows	Power Query as a service that runs in the cloud and is product-agnostic. The stored result can be used in other applications as services.

See also

[Data sources in Power Query](#)

[Getting data](#)

[Power Query quickstart](#)

[Shape and combine data using Power Query](#)

[What are dataflows](#)

Getting data

1/15/2022 • 3 minutes to read • [Edit Online](#)

Power Query can connect to many different data sources so you can work with the data you need. This article walks you through the steps for bringing in data to Power Query.

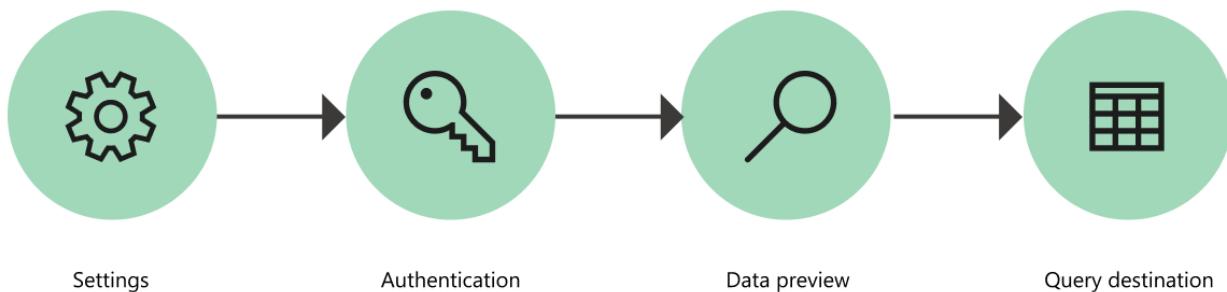
Connecting to a data source with Power Query follows a standard set of stages before landing the data at a destination. This article describes each of these stages.

NOTE

In some cases, a connector might have all of these stages, and in other cases a connector might have just a few of them. For more information about the experience of a specific connector, go to the documentation available for the specific connector.

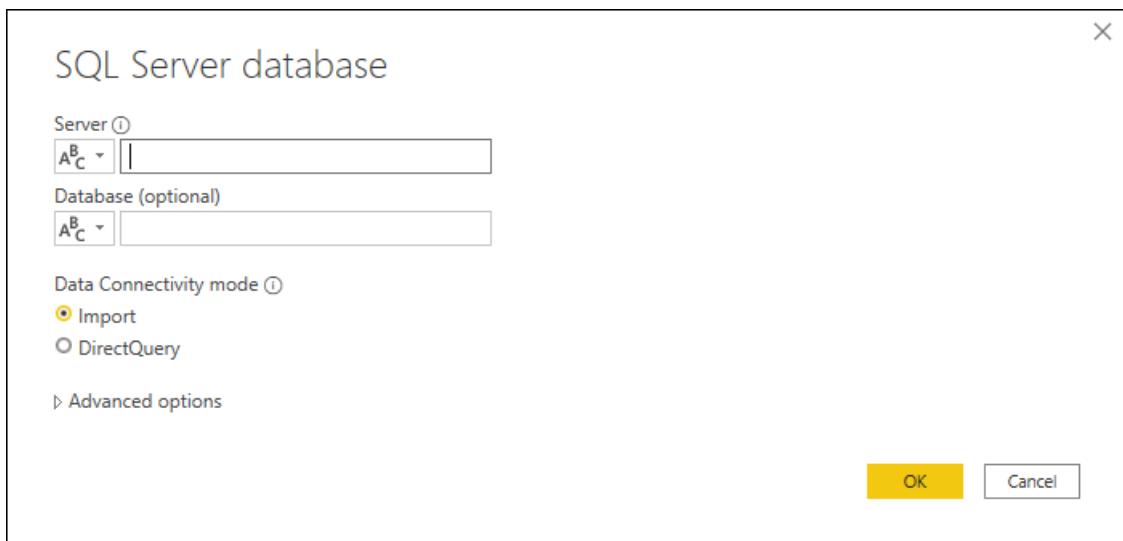
The stages are:

1. Connection settings
2. Authentication
3. Data preview
4. Query destination



1. Connection settings

Most connectors initially require at least one parameter to initialize a connection to the data source. For example, the SQL Server connector requires at least the host name to establish a connection to the SQL Server database.



In comparison, when trying to connect to an Excel file, Power Query requires that you use the file path to find the file you want to connect to.

The connector parameters are commonly used to establish a connection to a data source, and they—in conjunction with the connector used—define what's called a *data source path*.

NOTE

Some connectors don't require you to enter any parameters at all. These are called *singleton connectors* and will only have one data source path available per environment. Some examples are Adobe Analytics, MailChimp, and Google Analytics.

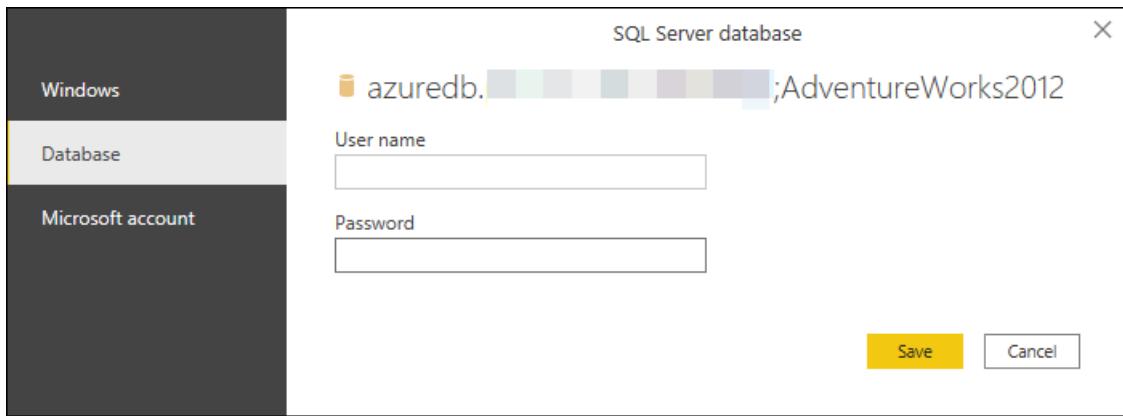
2. Authentication

Every single connection that's made in Power Query has to be authenticated. The authentication methods vary from connector to connector, and some connectors might offer multiple methods of authentication.

The currently available methods of authentication for Power Query are:

- **Anonymous:** Commonly used when connecting to a data source that doesn't require user authentication, such as a webpage or a file available over public HTTP.
- **Basic:** A **username** and **password** sent in base64 encoding are accepted for authentication.
- **API Key:** A single API key is accepted for authentication.
- **Organizational account or Microsoft account:** This method is also known as **OAuth 2.0**.
- **Windows:** Can be implicit or explicit.
- **Database:** This is only available in some database connectors.

For example, the available authentication methods for the SQL Server database connector are Windows, Database, and Microsoft account.



3. Data preview

The goal of the data preview stage is to provide you with a user-friendly way to preview and select your data.

Depending on the connector that you're using, you can preview data by using either:

- Navigator window
- Table preview dialog box

Navigator window (navigation table)

The **Navigator** window consists of two main sections:

- The object selection pane is displayed on the left side of the window. The user can interact with and select these objects.

NOTE

For Power Query in Excel, select the **Select multiple items** option from the upper-left corner of the navigation window to select more than one object at a time in the object selection pane.

NOTE

The list of objects in Power Query Desktop is limited to 10,000 items. This limit does not exist in Power Query Online. For a workaround in Power Query Desktop, see [Object limitation workaround](#).

- The data preview pane on the right side of the window shows a preview of the data from the object you selected.

The screenshot shows the Power Query Desktop Navigator window. On the left, there's a tree view of data sources under 'AdventureWorks2012 [93]'. On the right, a preview pane displays the contents of the 'HumanResources.vEmployee' table.

BusinessEntityID	Title	FirstName	MiddleName
1	null	Ken	J
2	null	Terri	Lee
3	null	Roberto	
4	null	Rob	
5	Ms.	Gail	A
6	Mr.	Jossef	H
7	null	Dylan	A
8	null	Diane	L
9	null	Gigi	N
10	null	Michael	
11	null	Ovidiu	V
12	null	Thierry	B
13	Ms.	Janice	M
14	null	Michael	I
15	null	Sharon	B
16	null	David	M
17	null	Kevin	F
18	null	John	L
19	null	Mary	A
20	null	Wanida	M
21	null	Terry	J
22	null	Sariya	E

Buttons at the bottom include 'Select Related Tables', 'Load', 'Transform Data', and 'Cancel'.

Object limitation workaround

There's a fixed limit of 10,000 objects in the **Navigator** in Power Query Desktop. This limit does not occur in Power Query Online. Eventually, the Power Query Online UI will replace the one in the desktop.

In the interim, you can use the following workaround:

1. Right-click on the root node of the **Navigator**, and then select **Transform Data**.

The screenshot shows the Power Query Editor interface. A context menu is open over a table node in the navigation pane. The 'Transform Data' option is highlighted.

- Transform Data
- Load
- Refresh

2. Power Query Editor then opens with the full navigation table in the table preview area. This view doesn't

have a limit on the number of objects, and you can use filters or any other Power Query transforms to explore the list and find the rows you want (for example, based on the **Name** column).

- Upon finding the item you want, you can get at the contents by selecting the data link (such as the **Table** link in the following image).

	Name	Data	Signature
1	Alphabetical_list_of_products	Table	table
2	Categories	Table	table
3	Category_Sales_for_1997	Table	table
4	Current_Product_Lists	Table	table
5	Customer_and_Suppliers_by_Cities	Table	table
6	CustomerDemographics	Table	table
7	Customers	Table	table
8	Employees	Table	table
9	Invoices	Table	table
10	Order_Details	Table	table

Table preview dialog box

The table preview dialog box consists of only one section for the data preview. An example of a connector that provides this experience and window is the **Folder** connector.

Content	Name	Extension	Date accessed	Date modified	Created	Attributes	Folder Path
[Binary]	.xlsx		null	6/6/2016, 10:31:53 AM	2/9/2016, 1:14:34 PM	[Record]	
[Binary]	.pptx		null	2/16/2016, 2:02:36 PM	2/9/2016, 4:03:46 PM	[Record]	
[Binary]	.pptx		null	2/10/2016, 7:26:03 AM	2/10/2016, 7:26:03 AM	[Record]	
[Binary]	.pptx		null	2/12/2016, 7:59:13 AM	2/11/2016, 8:40:05 AM	[Record]	
[Binary]	.pptx		null	2/12/2016, 12:35:59 PM	2/11/2016, 9:33:01 PM	[Record]	
[Binary]	.onetoc2		null	6/21/2016, 11:07:42 PM	2/18/2016, 12:20:32 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2016, 11:33:26 AM	2/18/2016, 12:22:00 PM	[Record]	
[Binary]	.onetrc2		null	12/5/2016, 4:48:29 PM	2/18/2016, 12:22:03 PM	[Record]	
[Binary]	.one		null	4/4/2018, 10:37:21 AM	2/18/2016, 12:22:04 PM	[Record]	
[Binary]	.one		null	7/29/2016, 10:26:07 PM	2/18/2016, 12:22:05 PM	[Record]	
[Binary]	.onetoc2		null	1/2/2019, 3:35:18 PM	2/18/2016, 12:49:30 PM	[Record]	
[Binary]	.onetoc2		null	3/29/2016, 12:20:58 PM	2/18/2016, 12:49:35 PM	[Record]	
[Binary]	.onetoc2		null	1/7/2019, 3:45:51 PM	2/18/2016, 12:50:07 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2016, 11:33:25 AM	2/18/2016, 12:51:41 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2016, 11:32:48 AM	2/18/2016, 12:53:49 PM	[Record]	
[Binary]	.pptx		null	2/19/2016, 2:33:29 PM	2/18/2016, 2:51:45 PM	[Record]	
[Binary]	.onetoc2		null	6/21/2016, 11:07:40 PM	2/18/2016, 2:54:06 PM	[Record]	
[Binary]	.one		null	4/4/2018, 10:33:35 AM	2/18/2016, 2:54:56 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2018, 11:33:09 AM	2/19/2016, 9:59:42 AM	[Record]	
[Binary]	.one		null	4/1/2018, 10:37:17 AM	2/19/2016, 9:59:43 AM	[Record]	
[Binary]	.pptx		null	2/19/2016, 6:54:36 PM	2/19/2016, 1:18:38 PM	[Record]	
[Binary]	.ca		null	7/25/2016, 8:49:51 AM	2/25/2016, 8:49:51 AM	[Record]	
[Binary]	.dock		null	6/3/2019, 4:25:22 AM	5/28/2019, 12:35:48 AM	[Record]	
[Binary]	.dock		null	7/25/2016, 8:49:50 AM	2/25/2019, 8:49:50 AM	[Record]	
[Binary]	.one		null	9/12/2019, 5:29:28 AM	2/22/2016, 2:59:16 PM	[Record]	
[Binary]	.cs		null	7/25/2016, 8:49:53 AM	2/25/2019, 8:49:53 AM	[Record]	
[Binary]	.cs		null	7/25/2016, 8:49:52 AM	2/25/2019, 8:49:52 AM	[Record]	
[Binary]	.dock		null	7/25/2016, 8:49:48 AM	2/25/2019, 8:49:48 AM	[Record]	
[Binary]	.onetoc2		null	12/10/2018, 2:31:28 AM	2/23/2016, 10:23:34 AM	[Record]	
[Binary]	.one		null	4/4/2018, 10:36:58 AM	2/23/2016, 10:23:36 AM	[Record]	
[Binary]	.one		null	4/1/2018, 10:34:45 AM	2/24/2016, 1:41:16 PM	[Record]	
[Binary]	.one		null	4/1/2018, 10:37:17 AM	2/24/2016, 6:58:14 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2016, 11:33:22 AM	2/24/2016, 7:06:38 PM	[Record]	
[Binary]	.one		null	4/1/2018, 10:36:57 AM	2/25/2016, 10:14:27 AM	[Record]	
[Binary]	.one		null	4/4/2018, 10:34:36 AM	2/25/2016, 10:56:50 AM	[Record]	
[Binary]	.one		null	1/10/2019, 2:51:37 PM	2/25/2016, 7:22:47 PM	[Record]	
[Binary]	.onetoc2		null	10/8/2016, 11:33:15 AM	2/26/2016, 12:16:21 PM	[Record]	
[Binary]	.one		null	4/6/2018, 11:03:41 AM	2/26/2016, 12:39:19 PM	[Record]	

4. Query destination

This is the stage in which you specify where to load the query. The options vary from integration to integration, but the one option that's always available is loading data to the Power Query Editor to further transform and enrich the query.

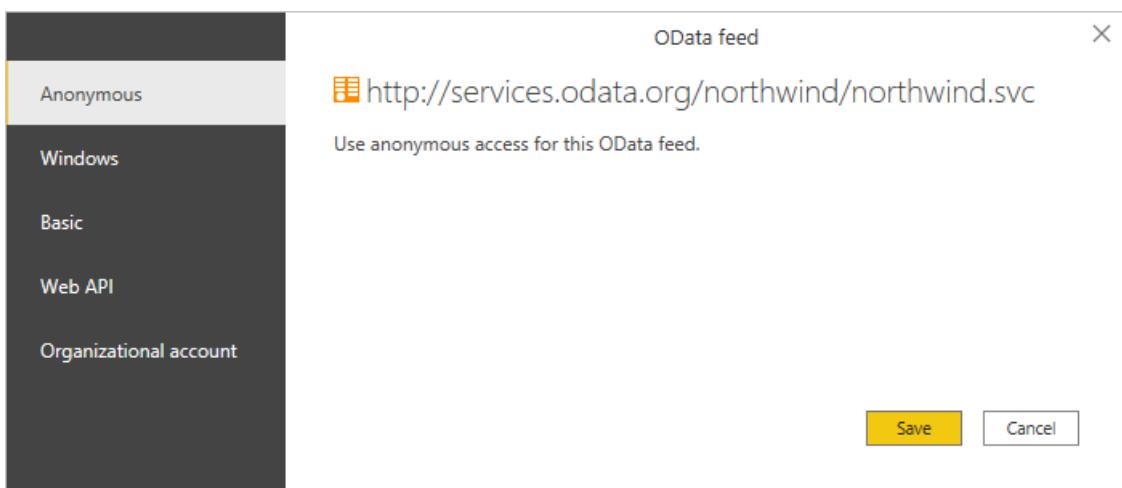
Authentication with a data source

1/15/2022 • 5 minutes to read • [Edit Online](#)

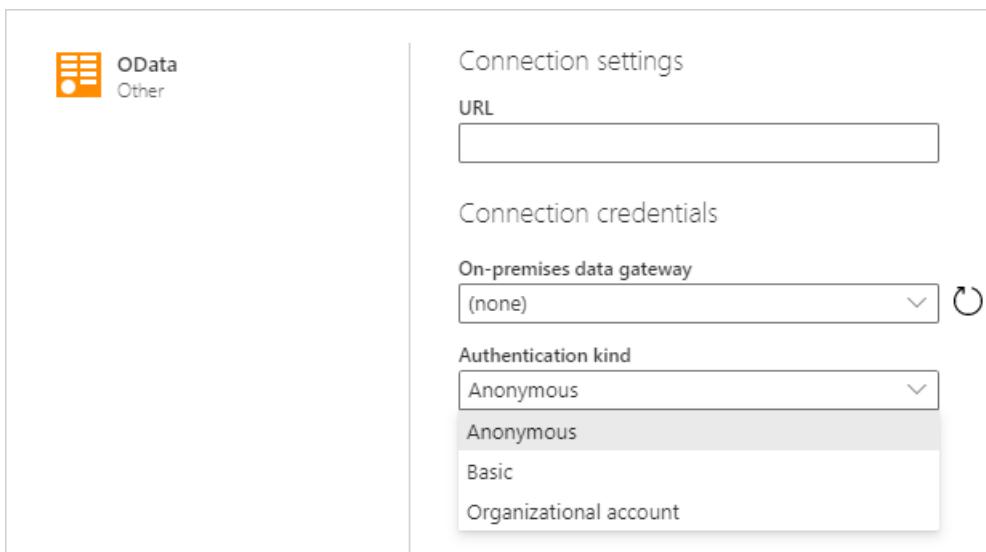
When you attempt to connect to a data source using a new connector for the first time, you might be asked to select the authentication method to use when accessing the data. After you've selected the authentication method, you won't be asked to select an authentication method for the connector using the specified connection parameters. However, if you need to change the authentication method later, you can do so.

Select an authentication method

Different connectors show different authentication methods. For example, the OData Feed connector in Power BI Desktop and Excel displays the following authentication method dialog box.



If you're using a connector from an online app, such as the Power BI service or Power Apps, you'll see an authentication method dialog box for the OData Feed connector that looks something like the following image.

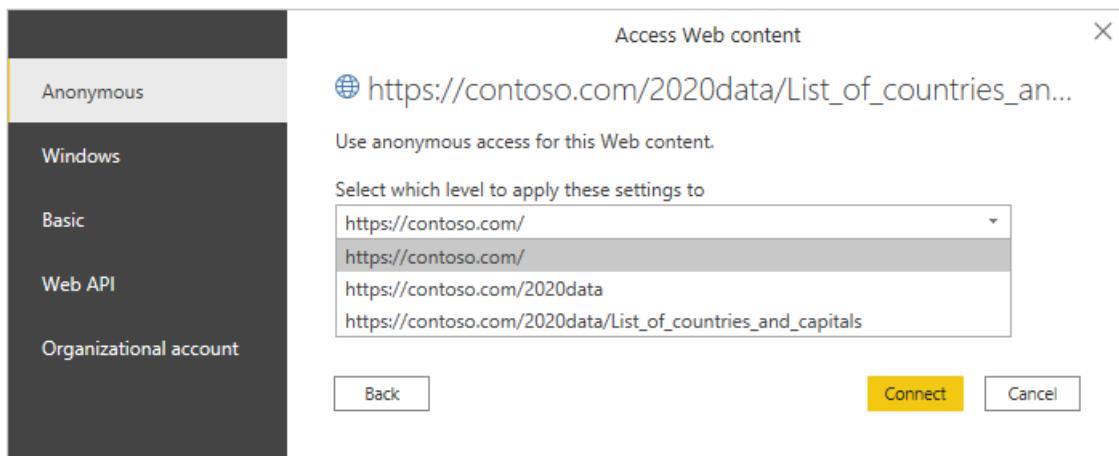


As you can see, a different selection of authentication methods is presented from an online app. Also, some connectors might ask you to enter the name of an on-premises data gateway to be able to connect to your data.

Set the level of the authentication method

In connectors that require you to enter a URL, you'll be asked to select the level to which the authentication

method will be applied. For example, if you select the Web connector with a URL of [https://contoso.com](https://contoso.com/2020data>List_of_countries_and_capitals, the default level setting for your authentication method will be <a href=).



The level you select for the authentication method you chose for this connector determines what part of a URL will have the authentication method applied to it. If you select the top-level web address, the authentication method you select for this connector will be used for that URL address or any subaddress within that address.

However, you might not want to set the top-level address to a specific authentication method because different subaddresses can require different authentication methods. One example might be if you were accessing two separate folders of a single SharePoint site and wanted to use different Microsoft accounts to access each one.

After you've set the authentication method for a connector's specific address, you won't need to select the authentication method for that connector using that URL address or any subaddress again. For example, let's say you select the <https://contoso.com/> address as the level you want the Web connector URL settings to apply to. Whenever you use a [Web connector](#) to access any webpage that begins with this address, you won't be required to select the authentication method again.

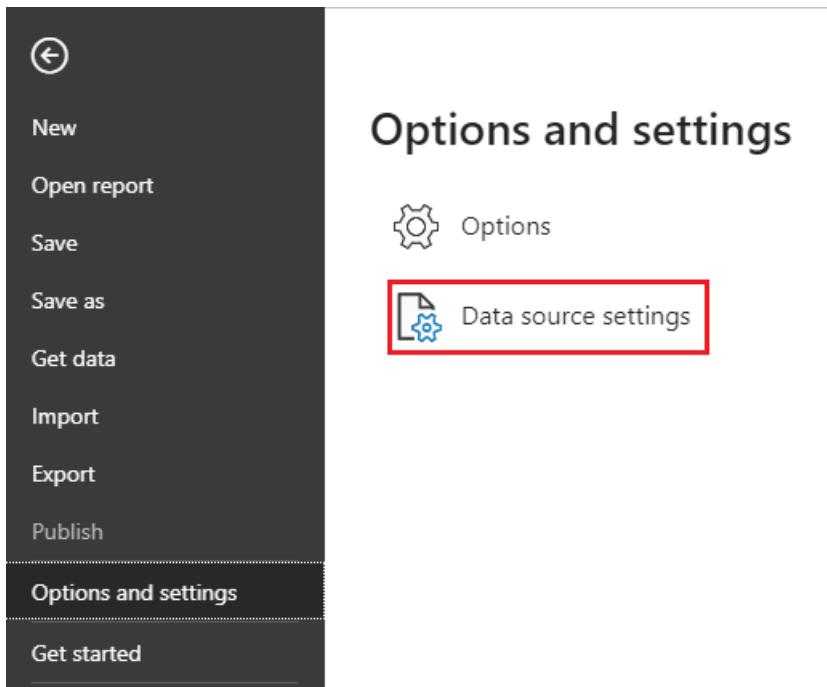
Change the authentication method

In some cases, you might need to change the authentication method you use in a connector to access a specific data source.

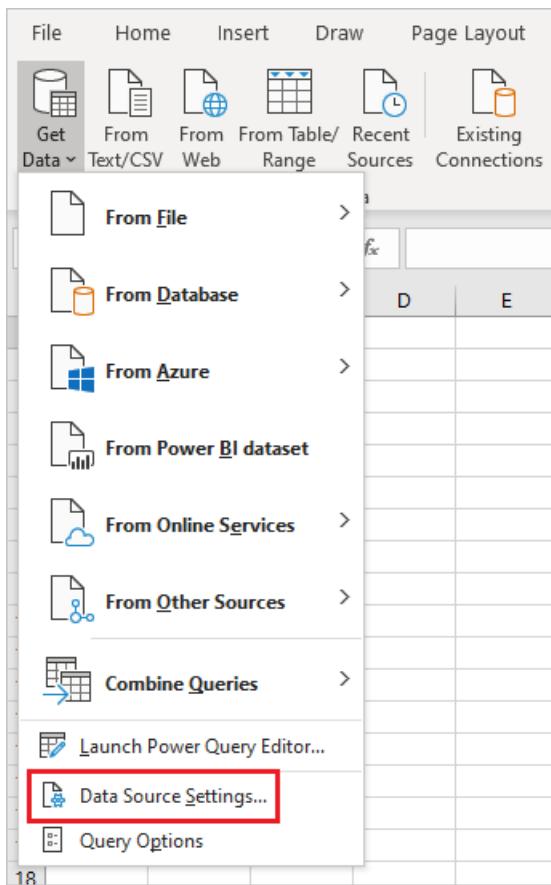
To edit the authentication method in Power BI Desktop or Excel

1. Do one of the following:

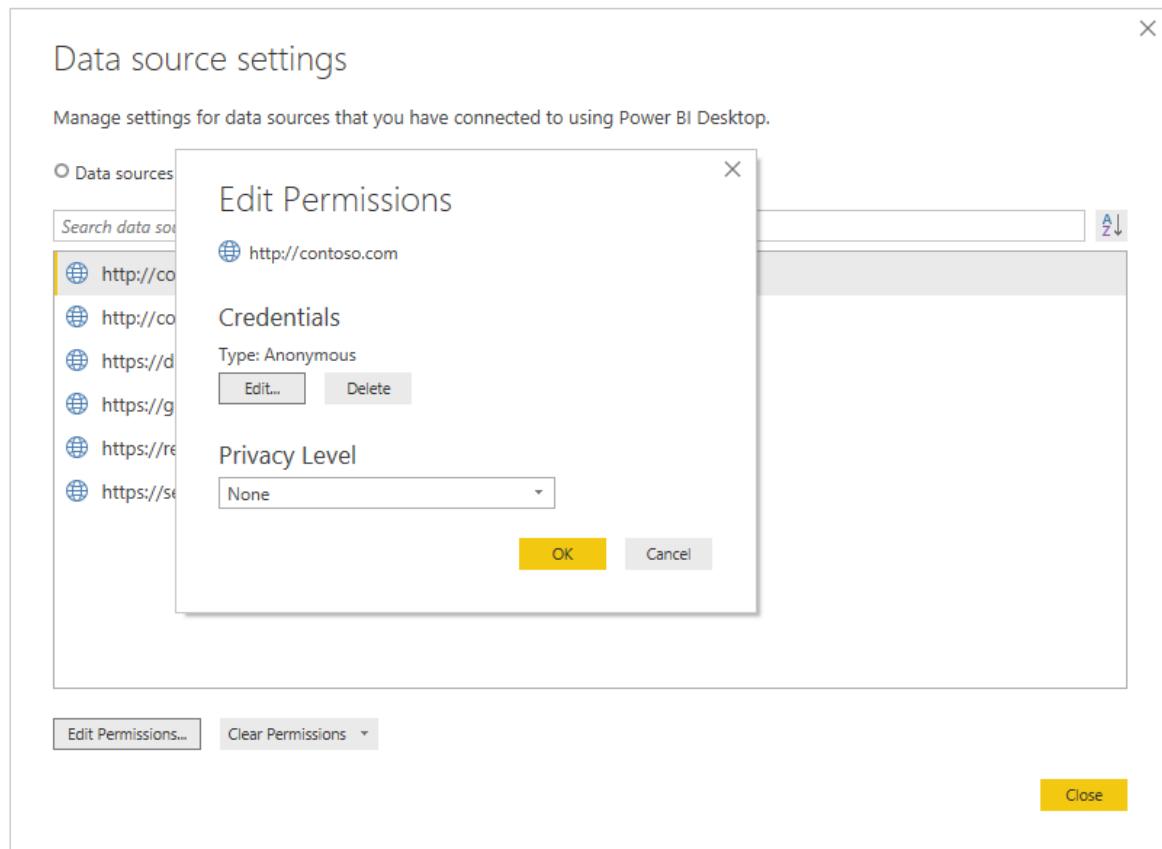
- In Power BI Desktop, on the **File** tab, select **Options and settings > Data source settings**.



- In Excel, on the Data tab, select Get Data > Data Source Settings.



2. In the Data source settings dialog box, select Global permissions, choose the website where you want to change the permission setting, and then select Edit Permissions.
3. In the Edit Permissions dialog box, under Credentials, select Edit.

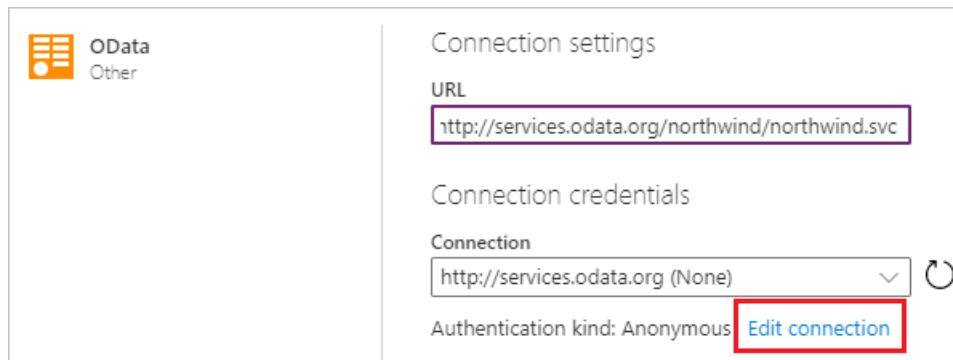


4. Change the credentials to the type required by the website, select **Save**, and then select **OK**.

You can also delete the credentials for a particular website in step 3 by selecting **Clear Permissions** for a selected website, or by selecting **Clear All Permissions** for all of the listed websites.

To edit the authentication method in online services, such as for dataflows in the Power BI service and Microsoft Power Platform

1. Select the connector, and then select **Edit connection**.



2. Make the required changes, and then select **Next**.

Connecting with Azure Active Directory using the Web and OData connectors

When connecting to data sources and services that require authentication through OAuth or Azure Active Directory-based authentication, in certain cases where the service is configured correctly, you can use the built-in [Web](#) or [OData](#) connectors to authenticate and connect to data without requiring a service-specific or custom connector.

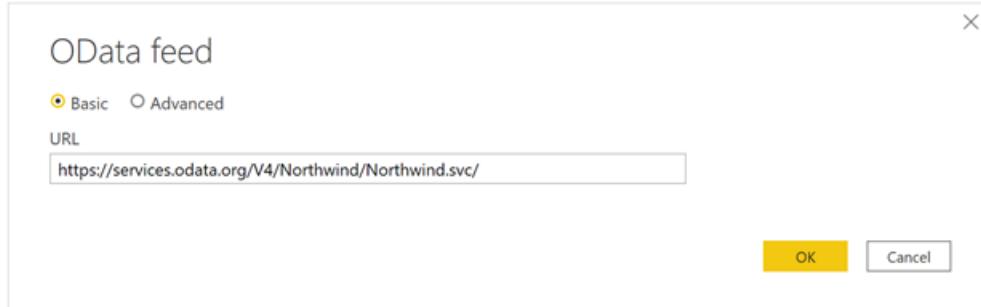
This section outlines connection symptoms when the service isn't configured properly. It also provides information on how Power Query interacts with the service when it's properly configured.

Symptoms when the service isn't configured properly

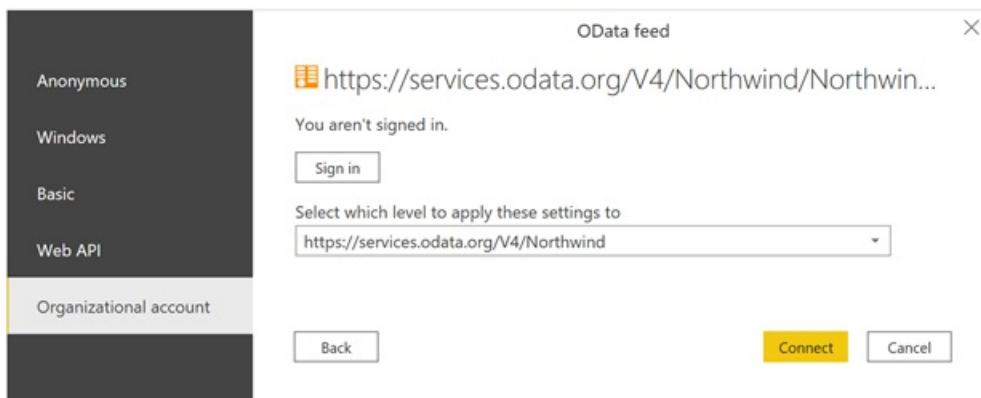
If you run into the error **We were unable to connect because this credential type isn't supported for this resource. Please choose another credential type.**, this error means that your service doesn't support the authentication type.

One example of this is the Northwind OData service.

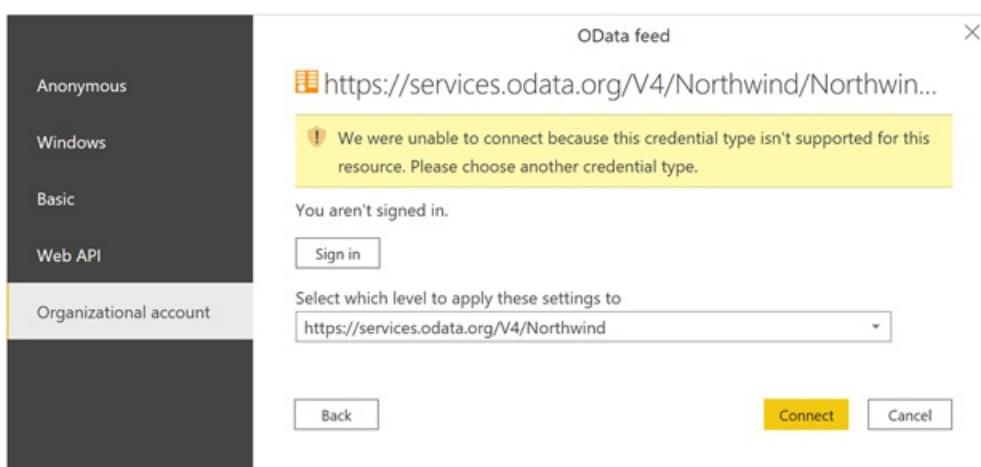
1. Enter the Northwind endpoint in the "Get Data" experience using the OData connector.



2. Select **OK** to enter the authentication experience. Normally, because Northwind isn't an authenticated service, you would just use **Anonymous**. To demonstrate lack of support for Azure Active Directory, choose **Organizational account**, and then select **Sign in**.



3. You'll encounter the error, indicating that OAuth or Azure Active Directory authentication isn't supported in the service.



Supported workflow

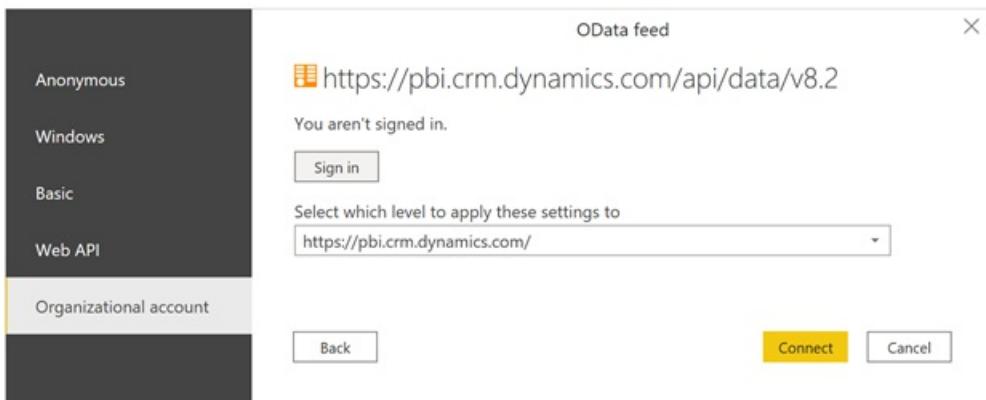
One example of a supported service working properly with OAuth is CRM, for example,

https://*.crm.dynamics.com/api/data/v8.2.

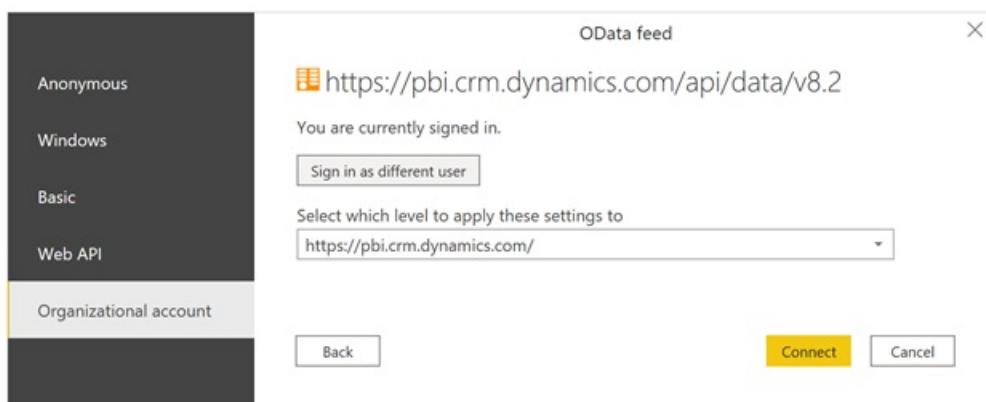
1. Enter the URL in the "Get Data" experience using the OData connector.



2. Select **Organizational Account**, and then select **Sign-in** to proceed to connect using OAuth.



3. The request succeeds and the OAuth flow continues to allow you to authenticate successfully.



When you select **Sign-in** in Step 2 above, Power Query sends a request to the provided URL endpoint with an Authorization header with an empty bearer token.

```
GET https://*.crm.dynamics.com/api/data/v8.2 HTTP/1.1
Authorization: Bearer
User-Agent: Microsoft.Data.Mashup (https://go.microsoft.com/fwlink/?LinkID=304225)
Host: pbi.crm.dynamics.com
Connection: Keep-Alive
```

The service is then expected to respond with a **401** response with a **WWW_Authenticate** header indicating the Azure AD authorization URI to use. This response should include the tenant to sign into, or **/common/** if the resource isn't associated with a specific tenant.

```
HTTP/1.1 401 Unauthorized
Cache-Control: private
Content-Type: text/html
Server:
WWW-Authenticate: Bearer authorization_uri=https://login.microsoftonline.com/3df2eaf6-33d0-4a10-8ce8-7e596000eb7/oauth2/authorize
Date: Wed, 15 Aug 2018 15:02:04 GMT
Content-Length: 49
```

Power Query can then initiate the OAuth flow against the **authorization_uri**. Power Query requests an Azure AD Resource or Audience value equal to the domain of the URL being requested. This value would be the value you use for your Azure Application ID URL value in your API/service registration. For example, if accessing <https://api.myservice.com/path/to/data/api>, Power Query would expect your Application ID URL value to be equal to <https://api.myservice.com>.

The following Azure Active Directory client IDs are used by Power Query. You might need to explicitly allow these client IDs to access your service and API, depending on your overall Azure Active Directory settings.

CLIENT ID	TITLE	DESCRIPTION
a672d62c-fc7b-4e81-a576-e60dc46e951d	Power Query for Excel	Public client, used in Power BI Desktop and Gateway.
b52893c8-bc2e-47fc-918b-77022b299bbc	Power BI Data Refresh	Confidential client, used in Power BI service.

If you need more control over the OAuth flow (for example, if your service must respond with a [302](#) rather than a [401](#)), or if your application's Application ID URL or Azure AD Resource value don't match the URL of your service, then you'd need to use a custom connector. For more information about using our built-in Azure AD flow, go to [Azure Active Directory authentication](#).

SharePoint and OneDrive for Business files import

1/15/2022 • 6 minutes to read • [Edit Online](#)

Power Query offers a series of ways to gain access to files that are hosted on either SharePoint or OneDrive for Business.

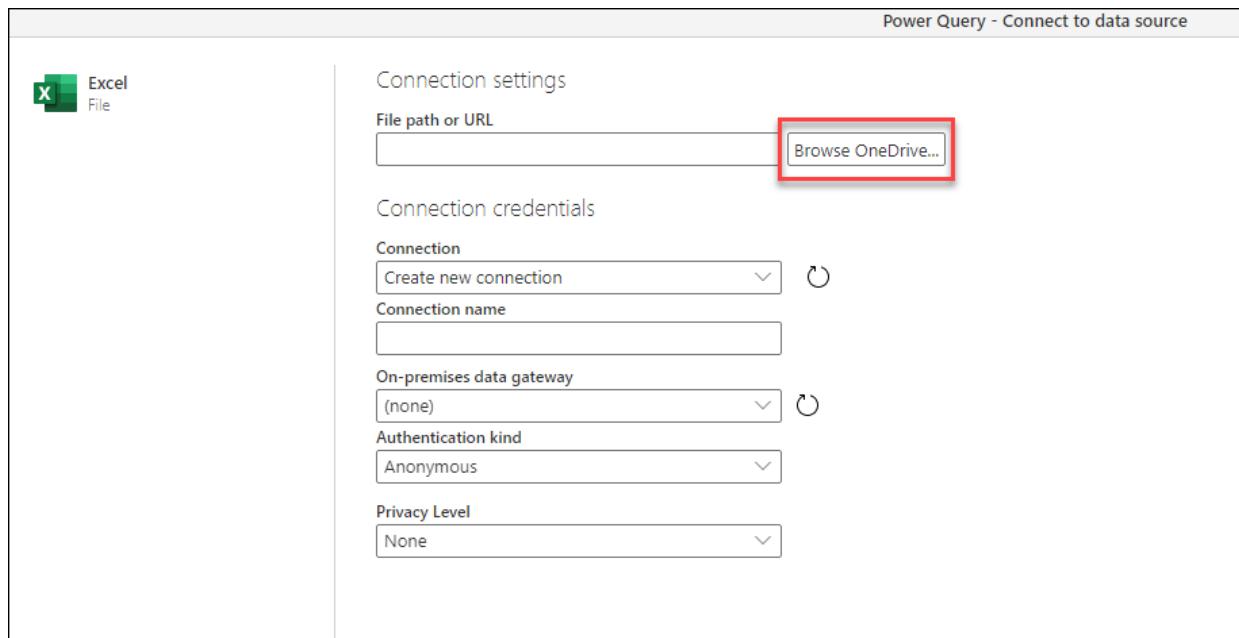
Browse files

NOTE

Currently, you can only browse for OneDrive for Business files of the authenticated user inside of Power Query Online for PowerApps.

Power Query provides a **Browse OneDrive** button next to the **File path or URL** text box when you create a dataflow in PowerApps using any of these connectors:

- [Excel](#)
- [JSON](#)
- [PDF](#)
- [XML](#)
- [TXT/CSV](#)



When you select this button, you'll be prompted to go through the authentication process. After completing this process, a new window appears with all the files inside the OneDrive for Business of the authenticated user.

My files

Name	Modified	Modified By	File size	Sharing
Top M Articles.xlsx	October 23	Miguel Torrez	102 KB	Private

Open Cancel

You can select the file of your choice, and then select the **Open** button. After selecting **Open**, you'll be taken back to the initial connection settings page where you'll see that the **File path or URL** text box now holds the exact URL to the file you've selected from OneDrive for Business.

Power Query - Connect to data source

Excel File

Connection settings

File path or URL
https://contoso1-my.sharepoint.com/personal/... Browse OneDrive...

Connection credentials

Connection
mtorrez@contoso1.com (None) ⚡
Authentication kind: Organizational account Edit connection

You can select the **Next** button at the bottom-right corner of the window to continue the process and get your data.

From the Web connector using a file URL

1. Navigate to your OneDrive for Business location using a browser. Right-click the file you want to use, and select **Open in Excel**.

NOTE

Your browser interface might not look exactly like the following image. There are many ways to select **Open in Excel** for files in your OneDrive for Business browser interface. You can use any option that allows you to open the file in Excel.

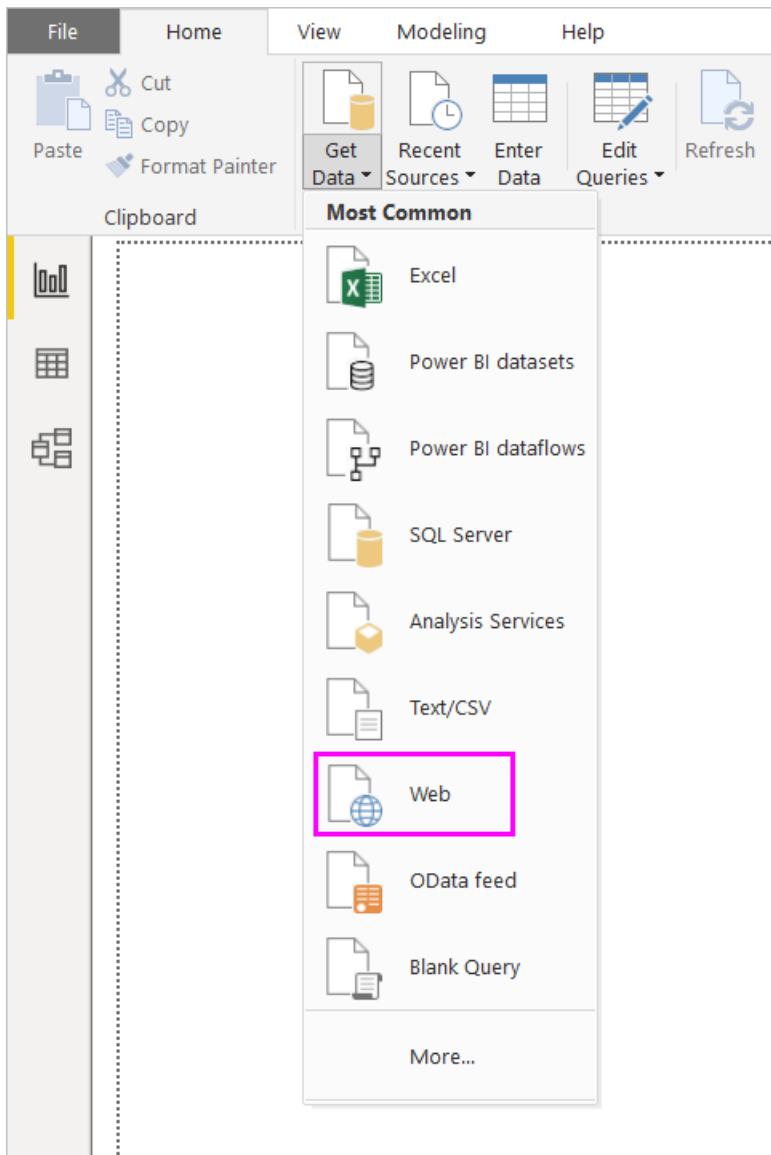
A screenshot of the Microsoft OneDrive web interface. On the left, there's a sidebar with 'Recent', 'Shared with me', 'Discover', and 'Recycle bin' sections, followed by 'Groups' which includes 'DACL', 'New Samples Group', 'C+E BI content tea...', and 'Q3 Product Group'. The main area shows a folder structure: 'Files > Power BI > PBI De...'. A context menu is open over a file named 'PBI_Edu_ELSi_Enrollment.xlsx'. The menu items are: 'Open in Excel Online', 'Open In Excel' (which is highlighted with a mouse cursor), 'Share', 'Get a link', 'Download', 'Delete', 'Move to', 'Copy to', 'Rename', 'Version History', and 'Details'. Below the menu, a list of files is shown with columns for 'Name', 'Modified By', 'File Size', and 'Sharing'. The file 'PBI_Edu_ELSi_Enrollment.xlsx' is selected.

2. In Excel, select **File > Info**, and then select the **Copy path** button.

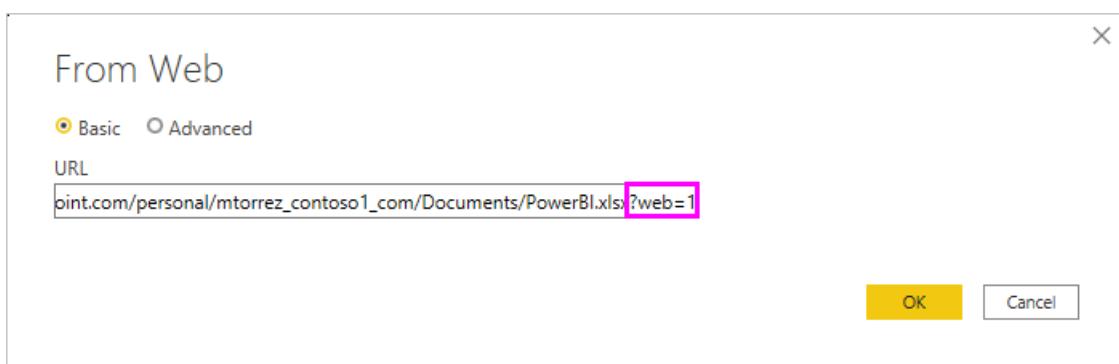
A screenshot of the Microsoft Excel 'Info' dialog box. The title bar says '2019 BI Offsite SWOT Analysis.xlsx - Saving...'. The left sidebar has options: Home, New, Open, Info (which is selected and highlighted with a green box), Save a Copy, Print, Share, and Export. The main area shows the file name '2019 BI Offsite SWOT Analysis' and its location 'OneDrive - Microsoft » Power BI'. At the bottom of the dialog, there are three buttons: 'Share', 'Copy path' (which is highlighted with a pink arrow), and 'Open file location'.

To use the link you just copied in Power Query, take the following steps:

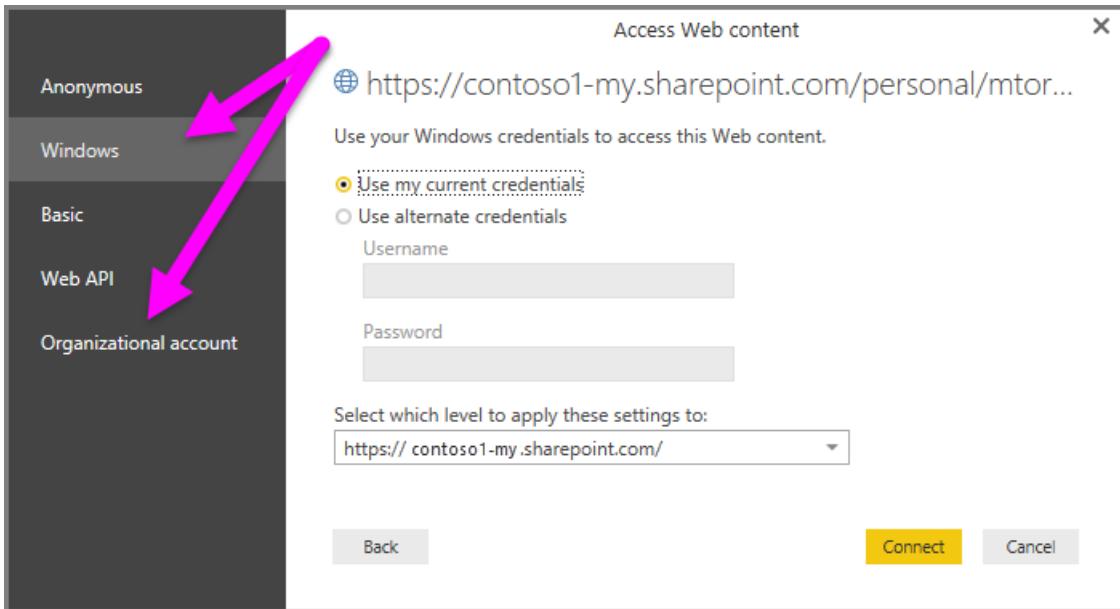
1. Select **Get Data > Web**.



2. In the **From Web** dialog box, select the **Basic** option and paste the link in **URL**.
3. Remove the **?web=1** string at the end of the link so that Power Query can properly navigate to your file, and then select **OK**.



4. If Power Query prompts you for credentials, choose either **Windows** (for on-premises SharePoint sites) or **Organizational Account** (for Microsoft 365 or OneDrive for Business sites). Then select **Connect**.



Caution

When working with files hosted on OneDrive for Home, the file that you want to connect to needs to be publicly available. When setting the authentication method for this connection, select the **Anonymous** option.

When the **Navigator** dialog box appears, you can select from the list of tables, sheets, and ranges found in the Excel workbook. From there, you can use the OneDrive for Business file just like any other Excel file. You can create reports and use it in datasets like you would with any other data source.

NOTE

To use a OneDrive for Business file as a data source in the Power BI service, with **Service Refresh** enabled for that file, make sure you select **OAuth2** as the **Authentication method** when configuring your refresh settings. Otherwise, you may encounter an error (such as, *Failed to update data source credentials*) when you attempt to connect or to refresh. Selecting **OAuth2** as the authentication method remedies that credentials error.

SharePoint folder connector

You can read a detailed step-by-step guide on how to connect to the files hosted on your SharePoint site in the [SharePoint folder](#) article.

After successfully establishing the connection, you'll be prompted with a table preview that shows the files in your SharePoint site. Select the **Transform data** button at the bottom right of the window.

Power Query - Preview folder data

<https://contoso.sharepoint.com>

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
[Binary]	01-January.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	02-February.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	03-March.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	04-April.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	05-May.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	06-June.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
[Binary]	03-March.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	02-Febrero.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	04-Abril.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	05-Mayo.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	06-Junio.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	01-Enero.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
[Binary]	Miembros.csv	.csv	null	4/30/2020, 9:45:00 AM	4/30/2020, 9:45:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Foro/
[Binary]	Evaluaciones.app	.app	null	7/5/2018, 9:34:00 AM	7/5/2018, 9:34:00 AM	[Record]	https://contoso.sharepoint.com/Evaluaciones1/
[Binary]	My New App.app	.app	null	7/5/2018, 8:44:00 AM	7/5/2018, 8:44:00 AM	[Record]	https://contoso.sharepoint.com/My New App1/
[Binary]	My New App.app	.app	null	7/5/2018, 9:35:00 AM	7/5/2018, 9:35:00 AM	[Record]	https://contoso.sharepoint.com/My New App_d4657293/

Back Cancel Combine Transform data

Selecting the **Transform Data** button will take you to a view of the data called the *File system view*. Each of the rows in this table represents a file that was found in your SharePoint site.

Power Query - Edit queries

Home Transform Add column View

Get data Enter data Options Manage parameters Refresh Advanced editor Properties Choose columns Remove columns Keep rows Remove rows Sort Split column Group by Replace values 123 Data type: Binary Use first row as headers Transform Merge queries Append queries Combine files Map to entity CDM AI insights

Queries SharePoint.Files("https://contoso.sharepoint.com", [ApiVersion = 151])

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
1 [Binary]	01-January.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
2 [Binary]	02-February.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
3 [Binary]	03-March.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
4 [Binary]	04-April.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
5 [Binary]	05-May.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
6 [Binary]	06-June.csv	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	https://contoso.sharepoint.com/Shared Documents/Sales Reports/
7 [Binary]	03-March.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
8 [Binary]	02-Febrero.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
9 [Binary]	04-Abril.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
10 [Binary]	05-Mayo.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
11 [Binary]	06-Junio.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
12 [Binary]	01-Enero.csv	.csv	null	9/12/2017, 7:08:00 AM	9/12/2017, 7:08:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Reportes de Vent...
13 [Binary]	Miembros.csv	.csv	null	4/30/2020, 9:45:00 AM	4/30/2020, 9:45:00 A..	[Record]	https://contoso.sharepoint.com/Shared Documents/Foro/
14 [Binary]	Evaluaciones.app	.app	null	7/5/2018, 9:34:00 AM	7/5/2018, 9:34:00 AM	[Record]	https://contoso.sharepoint.com/Evaluaciones1/
15 [Binary]	My New App.app	.app	null	7/5/2018, 8:44:00 AM	7/5/2018, 8:44:00 AM	[Record]	https://contoso.sharepoint.com/My New App1/
16 [Binary]	My New App.app	.app	null	7/5/2018, 9:35:00 AM	7/5/2018, 9:35:00 AM	[Record]	https://contoso.sharepoint.com/My New App_d4657293/

1 warning Columns: 8 Rows: 16

Completed (3.66 s) Cancel Save & close

The table has a column named **Content** that contains your file in a binary format. The values in the **Content** column have a different color than the rest of the values in the other columns of the table, which indicates that they're selectable.

By selecting a **Binary** value in the **Content** column, Power Query will automatically add a series of steps in your query to navigate to the file and interpret its contents where possible.

For example, from the table shown in the previous image, you can select the second row where the **Name** field has a value of **02-February.csv**. Power Query will automatically create a series of steps to navigate and interpret the contents of the file as a CSV file.

Power Query - Edit queries															
Queries	A1..Month														
	A..Month		A..Australia		A..Canada		A..Central		A..France		A..Germany		A..North		A..South
1	All-Purpose Blk..	7,698.75	45,058.28	1,558,631	16,045.76	13,886.16	82,684.53	23,779.48	7,119.44	4,059.11	9,905.9				
2	AWC Logo Cap	24,396.12	10,058.69	7,698.35	5,498.26	5,775.23	50,782.27	5,854.9	7,141.51	5,610.46					
3	Bike Wash - Dis...	32,983.7	13,049.42	11,342.9	8,071.57	7,794.22	83,247	53,721.53	23,608.47	20,554.32	58,108.61				
4	Cable Lock	23,351.31	7,393.56	22,578.3	26,464.49	49,624.65	17,286.76	24,356.99	35,220.42	11,418.46	536.49				
5	Chain	101,41	1,628.77	8,155.52	12,263.53	25,587.89	1,053.04	53,773.43	42,050.59	11,949.83	23,357.15				
6	Classic Vest_L	2,015.2	40,356.0	2,970.04	7,597.42	1,372.5	12,767.26	41,169.13	21,215.52	870.52	4,466.78				
7	Classic Vest_S	14,610.61	10,357.92	8,407.36	8,628.73	4,765.59	2,186.08	19,877.74	18,665.54	36,100.26	30,762.82				
8	Fender Set_...	6,999.52	58,358.89	57,640.67	1,423.55	16,086.61	23,368.18	2,670.8	886.65	8,740.76	3,316.62				
9	Front Brakes	4,777.97	8,673.63	30,399.68	11,741.95	47,602.30	45,634.03	30,021.1	62,001.16	34,457.47	17,639.51				
10	Full-Finger Glov...	36,515.9	45,191.6	6,698.35	47,490.92	56,643.39	21,674.31	10,656.68	18,378.08	15,13.67	959.14				
11	Half-Finger Go...	29,864.51	21,862.71	1,651.86	4,966.25	8,246.24	8,329.41	3,975.2	9,806.14	1,983.31					
12	Half-Finger Goo...	14,456.25	20,490.15	18,302.42	21,763.94	10,476.62	16,738.88	35,211.25	16,293.95	10,636.41					
13	HL Bottom Brac...	39,582.5	84,112.7	65,933.4	3,628.86	386.78	3,118.86	7,752.28	17,946.99	38,277.38	2,124.94				
14	HL Crankset	38,861.22	80,933.8	9,481.62	1,020.95	3,687.09	55,782.84	30,710.14	10,516.15	15,004.89					
15	HL Fork	8,703.0	37,771.23	10,595.88	26,629.37	27,155.82	26,047.1	14,387.49	20,026.97	18,894.26	20,160.75				
16	HL Headset	9,429.12	1,768.11	36,483.83	33,846.1	29,906.3	33,340.19	21,790.74	22,052.73	6,517.99	11,075.22				
17	HL Mountain Fr...	30,233.4	634.76	68,458.67	55,590.59	2,054.33	14,320.52	52,437.88	21,618.33	47,487.44	16,156.05				
18	HL Mountain Fr...	24,726.99	10,740.18	36,507.98	36,507.98	30,073.98	8,405.01	36,214.24	37,394	8,274.12	9,909.02				
19	HL Mountain Fr...	4,494.64	34,856.17	54,181.19	26,996.48	17,822.7	2,247.28	4,624.22	1,486.25	6,027.94	60,896.96				
20	HL Mountain Fr...	28,067.19	34,341.49	6,194.83	54,193.83	46,768.18	954.94	10,070.63	43,146.11	55,068.07	32,313.12				
21	HL Mountain Fr...	2,471.22	43,701.27	51,046.07	10,299.16	2,045.27	11,912.08	956.68	2,570.62	13,130.69	43,629.91				
22	HL Mountain H...	39,427.45	3,193.58	12,483.83	13,119.17	14,455.83	9,885.17	27,652.77	12,438.59	52,795.64	30,826.87				
23	HL Road Frame ...	11,691.27	48,165.82	34,876.48	39,724.64	765.71	379.97	46,533.58	28,255.78	74,917.37	6,351.3				
24	HL Road Frame ...	11,793.4	1,077.07	25,794.72	1,055.87	54,577.42	34,616.68	54,016.17	18,663.94	46,294.35	9,646.64				
25	HL Road Frame ...	2,189.69	4,180.39	88,054.88	3,391.77	7,446.55	4,482.51	6,770.45	19,094.19	4,700.29	46,106.37				
26	HL Road Handl...	28,777.37	35,972.15	2,848.57	70,669.61	49,765.12	51,062.26	40,223.88	15,705.6	15,659.12	18,339.44				
27	HL Road Pedal...	6,657.7	709.4	66,853.69	23,268.44	1,442.01	30,310.36	28,075.15	19,555.77	3,041.91	44,187.27				
28	HL Road Rear ...	17,556.83	7,126.67	2,242.67	11,821.07	38.4	38,258.01	43,554.78	13,750.45	16,733.91	15,193.6				
29	HL Road Tire	14,252.38	2,043.0	32,798.22	7,717.75	25,884.4	2,597.32	20,611.74	22,995.05	7,945.89	33,854.03				
30	HL Touring Fe...	18,201.58	29,730.34	35,198.66	75,044.63	594.16	24,836.8	21,774.41	6,375.61	37,558.89	2,369.02				
31	HL Touring Fee...	6,079.04	77,448.07	8,483.42	8,010.34	11,379.01	39,961.61	36,274.68	33,301.81	73,471.08	13,387.10				

NOTE

You can interact with the table by applying filters, sortings, and other transforms before navigating to the file of your choice. Once you've finished these transforms, select the **Binary** value you want to view.

OneDrive for Business experience

The SharePoint folder connector and its experience also work for files hosted on OneDrive for Business. However, the URL that you need to use is different from the one you would use for a SharePoint site. To locate your unique URL, go to your OneDrive for Business portal and copy the URL from your browser. This URL may look similar to the following example:

https://contoso-my.sharepoint.com/personal/user123_contoso_com/_layouts/15/onedrive.aspx

You don't need the full URL, but only the first few parts. The URL you need to use in Power Query will have the following format:

https://<unique_tenant_name>.sharepoint.com/personal/<user_identifier>

For example:

https://contoso-my.sharepoint/personal/user123_contoso_com

SharePoint.Contents function

While the SharePoint folder connector offers you an experience where you can see all the files available in your SharePoint or OneDrive for Business site at once, you can also opt for a different experience. In this experience, you can navigate through your SharePoint or OneDrive for Business folders and reach the folder or file(s) that you're interested in.

This experience is provided through the `SharePoint.Contents` function. Take the following steps to use this function:

1. Create a Blank Query.
 2. Change the code in the formula bar to be `SharePoint.Contents("url")` where `url` is the same format used for the SharePoint folder connector. For example:

```
SharePoint.Contents("https://contoso.sharepoint.com/marketing/data")
```

NOTE

By default, this function tries to use SharePoint API Version 14 to connect. If you aren't certain of the API version being used by your SharePoint site, you might want to try using the following example code:

```
SharePoint.Contents("https://contoso.sharepoint.com/marketing/data", [ApiVersion="Auto"]).
```

3. Power Query will request that you add an authentication method for your connection. Use the same authentication method that you'd use for the SharePoint files connector.
4. Navigate through the different documents to the specific folder or file(s) that you're interested in.

For example, imagine a SharePoint site with a *Shared Documents* folder. You can select the **Table** value in the **Content** column for that folder and navigate directly to that folder.

	ABC Content	A _C Name	A _C Extension	Date accessed	Date modified	Date created	Attributes	A _C Folder Path
1	[Table]	_catalogs		null	9/15/2013, 5:39:17 PM +00...	9/15/2013, 5:39:17 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
2	[Table]	_private		null	9/15/2013, 5:39:09 PM +00...	9/15/2013, 5:39:09 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
3	[Table]	Sharing Links		null	5/7/2020, 12:56:44 PM +00...	6/14/2019, 2:03:59 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
4	[Table]	images		null	9/15/2013, 5:39:09 PM +00...	9/15/2013, 5:39:09 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
5	[Table]	Evaluaciones1		null	7/5/2018, 4:34:24 PM +00...	7/5/2018, 4:33:34 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
6	[Table]	_cts		null	9/15/2013, 5:39:12 PM +00...	9/15/2013, 5:39:12 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
7	[Table]	SitePages		null	4/19/2015, 4:15:30 AM +00...	9/15/2013, 5:40:20 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
8	[Table]	Shared Documents		null	11/1/2020, 5:32:06 PM +00...	9/15/2013, 5:40:22 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
9	[Table]	My New App_d46572...		null	7/5/2018, 4:35:31 PM +00...	7/5/2018, 4:34:30 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
10	[Table]	m		null	9/15/2013, 5:40:19 PM +00...	9/15/2013, 5:40:19 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
11	[Table]	Style Library		null	9/15/2013, 5:39:47 PM +00...	9/15/2013, 5:39:22 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
12	[Table]	My New App1		null	7/5/2018, 3:45:04 PM +00...	7/5/2018, 3:36:40 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
13	[Table]	ProjectPolicyitemList		null	9/15/2013, 5:39:32 PM +00...	9/15/2013, 5:39:32 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
14	[Table]	_vti_pvt		null	9/15/2013, 5:39:09 PM +00...	9/15/2013, 5:39:09 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
15	[Table]	Access Requests		null	4/5/2020, 11:48:23 PM +00...	5/31/2014, 5:51:12 AM +00...	[Record]	https://contoso.sharepoint.com/Shar...
16	[Table]	SiteAssets		null	8/24/2016, 3:40:15 AM +00...	9/15/2013, 5:40:20 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
17	[Table]	IWConvertedForms		null	9/15/2013, 5:39:54 PM +00...	9/15/2013, 5:39:54 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
18	[Table]	FormServerTemplates		null	9/15/2013, 5:39:54 PM +00...	9/15/2013, 5:39:54 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
19	[Table]	Lists		null	9/15/2013, 5:39:13 PM +00...	9/15/2013, 5:39:13 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...

Inside this *Shared Documents* folder there's a folder where the company stores all the sales reports. This folder is named *Sales Reports*. You can select the **Table** value on the **Content** column for that row.

	ABC Content	A _C Name	A _C Extension	Date accessed	Date modified	Date created	Attributes	A _C Folder Path
1	[Table]	Foro		null	4/30/2020, 4:47:02 PM +00...	4/30/2020, 4:47:01 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
2	[Table]	Sales Reports		null	1/9/2016, 9:52:06 PM +00...	1/9/2016, 9:51:16 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
3	[Table]	Reportes de Ventas		null	9/12/2017, 2:08:09 PM +00...	9/12/2017, 2:08:01 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
4	[Table]	Forms		null	9/15/2013, 5:40:22 PM +00...	9/15/2013, 5:40:22 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...

With all the files inside the *Sales Reports* folder, you could select the **Combine files** button (see [Combine files overview](#)) to combine the data from all the files in this folder to a single table. Or you could navigate directly to a single file of your choice by selecting the **Binary** value from the **Content** column.

	ABC Content	A _C Name	A _C Extension	Date accessed	Date modified	Date created	Attributes	A _C Folder Path
1	[Binary]	01-January.csv	.CSV	null	1/9/2016, 9:52:04 PM +00...	1/9/2016, 9:52:04 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
2	[Binary]	05-May.csv	.CSV	null	1/9/2016, 9:52:05 PM +00...	1/9/2016, 9:52:05 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
3	[Binary]	02-February.csv	.CSV	null	1/9/2016, 9:52:04 PM +00...	1/9/2016, 9:52:04 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
4	[Binary]	03-March.csv	.CSV	null	1/9/2016, 9:52:05 PM +00...	1/9/2016, 9:52:05 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
5	[Binary]	06-June.csv	.CSV	null	1/9/2016, 9:52:06 PM +00...	1/9/2016, 9:52:06 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...
6	[Binary]	04-April.csv	.CSV	null	1/9/2016, 9:52:05 PM +00...	1/9/2016, 9:52:05 PM +00...	[Record]	https://contoso.sharepoint.com/Shar...

NOTE

The experience provided by the `SharePoint.Contents` function is optimal for SharePoint and OneDrive for Business environments with a large number of files.

Lack of Support for Microsoft Graph in Power Query

1/15/2022 • 2 minutes to read • [Edit Online](#)

Connecting to [Microsoft Graph REST APIs](#) from Power Query isn't recommended or supported. Instead, we recommend users explore alternative solutions for retrieving analytics data based on Graph, such as [Microsoft Graph data connect](#).

Users may find they can make certain REST calls to Microsoft Graph API endpoints work through the `Web.Contents` or `OData.Feed` functions, but these approaches are not reliable as long-term solutions.

This article outlines the issues associated with Microsoft Graph connectivity from Power Query and explain why it isn't recommended.

Authentication

The built-in Organizational Account authentication flow for Power Query's `Web.Contents` and `OData.Feed` functions isn't compatible with most Graph endpoints. Specifically, Power Query's Azure Active Directory (Azure AD) client requests the `user_impersonation` scope, which isn't compatible with Graph's security model. Graph uses a rich set of permissions that aren't available through our generic Web and OData connectors.

Implementing your own Azure AD credential retrieval flows directly from your query, or using hardcoded or embedded credentials, also isn't recommended for security reasons.

OData Libraries' incompatibility

Certain Graph endpoints and extensions to Graph may require the use of OData libraries and features that aren't supported by Power Query's built-in `OData.Feed` function because Graph and Power Query may be using two different versions of OData libraries. These issues generally result in errors retrieving the service's \$metadata document. Users may discover common guidance related to passing the `Implementation = "2.0"` option to the `OData.Feed` function call to ensure the latest supported OData libraries are used. While this approach does resolve certain OData incompatibilities, users may still encounter errors over time as Graph and Power Query adopt new versions of the OData libraries at different times.

Performance

The Microsoft Graph API is designed to support many application scenarios but is suboptimal for the large-scale data retrieval required for most analytics scenarios. Users trying to retrieve large amounts of data from Graph APIs may encounter performance issues. Details around scenario applicability can be found in the Graph documentation.

Using a custom connector

Some Power Query users have enabled Graph connectivity through custom connectors, limiting their functionality to certain parts of the Graph API. This approach allows connector developers to resolve general authentication issues by defining their own Azure AD client with Graph specific permissions. Some custom connectors work around OData challenges by using `Web.Contents` and simulating OData support within their connector logic. However, this approach isn't recommended as users frequently hit the performance and scalability issues described above. Developers who take this route should continue with these limitations in

mind.

Alternatives to out-of-box connectivity in Power BI Desktop

1/15/2022 • 2 minutes to read • [Edit Online](#)

While Power BI Desktop offers out-of-box connectivity to over 150 data sources, there may be cases where a user wants to connect to a data source for which no out-of-box connector is available.

Connectivity through generic interfaces

It may be possible to connect to certain data sources without a built-in out-of-box connector by using generic interface connectors.

For example, the **ODBC** connector can connect to services with ODBC interfaces, and the **Web** connector can connect to services with REST API interfaces.

Using available Power BI out-of-box generic interface connectors to connect through interfaces that the end data source supports allows users to connect to many more data sources on the internet than there are specific out-of-box connectors for.

Learn more about connectivity through generic interfaces [here](#).

Connectivity through a custom connector

The [Power Query SDK](#) allows users to create custom connectors to unlock connectivity scenarios to Power BI Desktop. Users can create and distribute custom connectors to end services and data sources they can authenticate to.

Community members and organizations may also share custom connectors that they've created. While Microsoft doesn't offer any support, ownership, or guarantees for these custom connectors, users may be able to use them for their scenarios. The Power BI Partner Program also includes many partners which can build custom connectors. Learn more about the program or find a partner here: [Contact a Power BI Partner](#).

Users that own an end service or data source can create a custom connector and may be eligible to [certify](#) the connector to have it made available publicly out-of-box within Power BI Desktop.

Request the data source owner to build and certify a connector

As only the data source owner or an approved third party can build and certify a custom connector for any service, end users are encouraged to share the demand for a connector directly with the data source owner to encourage investment into creating and certifying one.

Request in Power BI Ideas forum

In addition to directly engaging the data source owner, users should also create or vote on ideas in the [Power BI Ideas Forum](#) to demonstrate the need and demand for a connector. This feedback may also help encourage the data source owner to invest into a certified connector available for users out-of-box in Power BI Desktop.

The Power Query user interface

1/15/2022 • 10 minutes to read • [Edit Online](#)

With Power Query, you can connect to many different data sources and transform the data into the shape you want.

In this article, you'll learn how to create queries with Power Query by discovering:

- How the "Get Data" experience works in Power Query.
- How to use and take advantage of the Power Query user interface.
- How to perform common transformations like grouping and merging data.

If you're new to Power Query, you can [sign up for a free trial of Power BI](#) before you begin. You can use Power BI dataflows to try out the Power Query Online experiences described in this article.

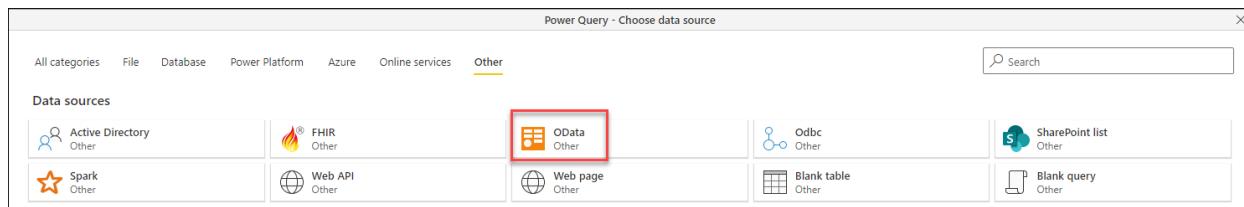
You can also [download Power BI Desktop for free](#).

Examples in this article connect to and use the [Northwind OData feed](#).

```
https://services.odata.org/V4/Northwind/Northwind.svc/
```

Connect to an OData feed

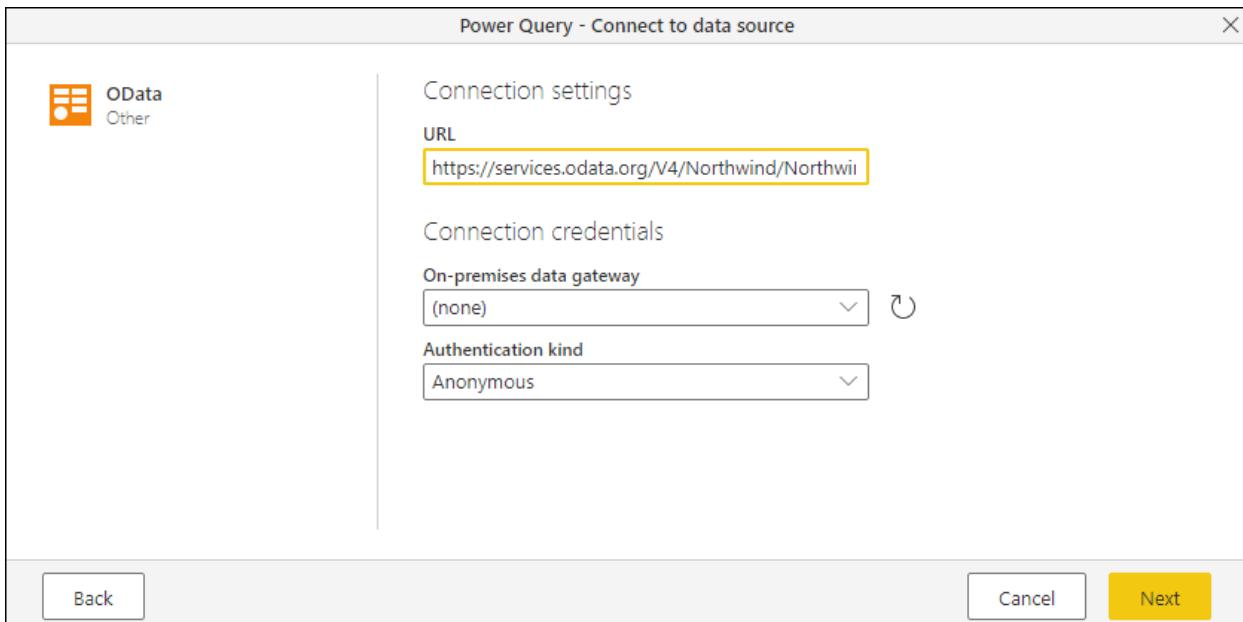
To start, locate the **OData** feed connector from the "Get Data" experience. You can select the **Other** category from the top, or search for **OData** in the search bar in the top-right corner.



Once you select this connector, the screen displays the connection settings and credentials.

- For **URL**, enter the URL to the Northwind OData feed shown in the previous section.
- For **On-premises data gateway**, leave as none.
- For **Authentication kind**, leave as anonymous.

Select the **Next** button.



The **Navigator** now opens, where you select the tables you want to connect to from the data source. Select the **Customers** table to load a preview of the data, and then select **Transform data**.

CustomerID	CompanyName	ContactName	ContactTitle	Address
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 12
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57
BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus
CACTU	Cactus Comidas para llevar	Patricia Simpson	Sales Agent	Cerrito 333
CENTC	Centro comercial Mótezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 23
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkeley Gardens 12
DRACD	Drachenblut Delikatessen	Sven Ottieb	Order Administrator	Walserweg 21
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinquante
EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George
ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6
FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assistant	Rua Orós, 92
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diono Rael	Accounting Manager	C/ Moralzarzal 86

The dialog then loads the data from the **Customers** table into the Power Query editor.

The above experience of connecting to your data, specifying the authentication method, and selecting the specific object or table to connect to is called the **Get data** experience and is documented with further detail in the [Getting data](#) article.

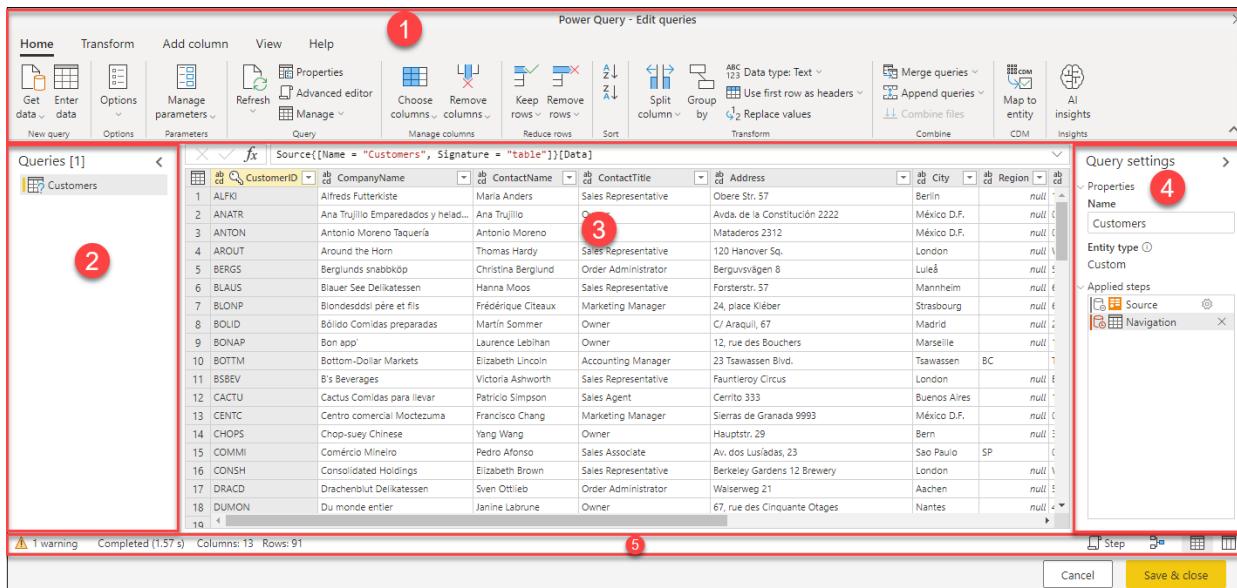
NOTE

To learn more about the OData feed connector, see [OData feed](#).

The Power Query editor user experience

The Power Query editor represents the Power Query user interface, where you can add or modify queries,

manage queries by grouping or adding descriptions to query steps, or visualize your queries and their structure with different views. The Power Query user interface has five distinct components.



- Ribbon**: the ribbon navigation experience, which provides multiple tabs to add transforms, select options for your query, and access different ribbon buttons to complete various tasks.
- Queries pane**: a view of all your available queries.
- Current view**: your main working view, that by default, displays a preview of the data for your query. You can also enable the [diagram view](#) along with the data preview view. You can also switch between the [schema view](#) and the data preview view while maintaining the diagram view.
- Query settings**: a view of the currently selected query with relevant information, such as query name, query steps, and various indicators.
- Status bar**: a bar displaying relevant important information about your query, such as execution time, total columns and rows, and processing status. This bar also contains buttons to change your current view.

NOTE

The schema and diagram view are currently only available in Power Query Online.

Using the Power Query editor

In this section, you'll begin transforming your data using Power Query. But before you start working on transforming the data, we'll discuss some of the UI panes that can be expanded or collapsed depending on their context. Selecting the appropriate panes lets you focus on the view that matters the most to you. We'll also discuss the different views that are available in the Power Query UI.

Expand and collapse panes

You'll notice that throughout the Power Query user interface there are icons that help you collapse or expand certain views or sections. For example, there's an icon on the top right-hand corner of the Queries pane that collapses the queries pane when selected, and expands the pane when selected again.

Queries [1] < fx Source{[Name = "Customers", Signature = "table"]}[Data]

	ab cd CompanyName	ab cd ContactName	ab cd ContactTitle
1 ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative
2 ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
3 ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner
4 AROUT	Around the Horn	Thomas Hardy	Sales Representative

Switch between views

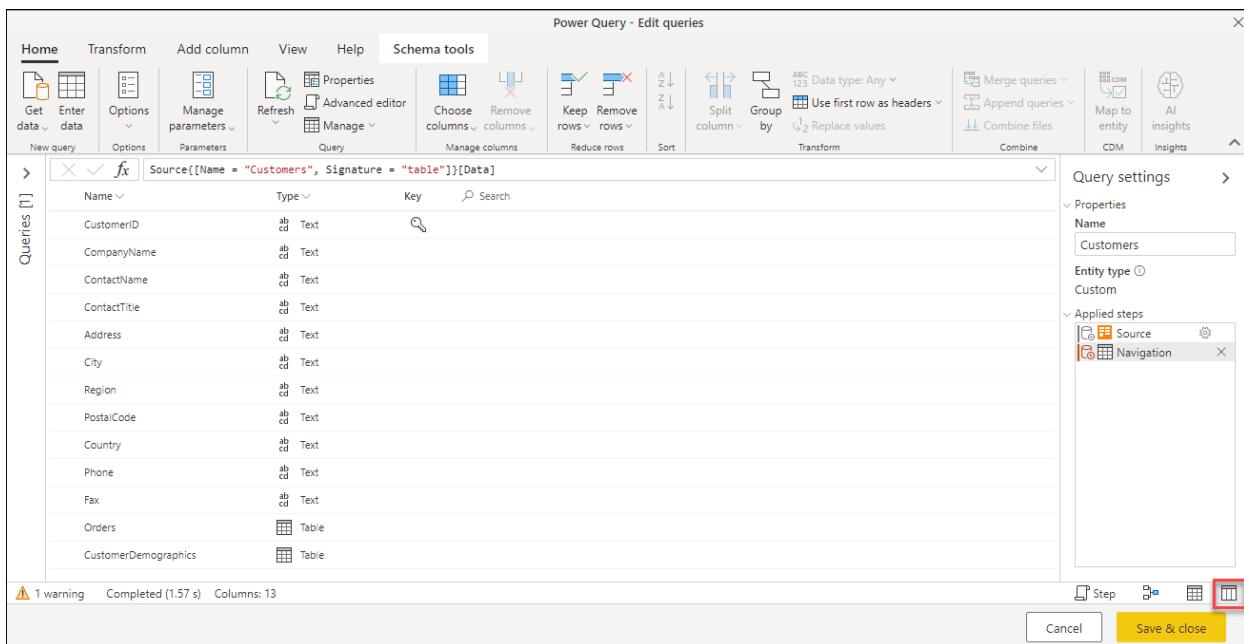
Apart from being able to collapse certain panes and sections in the Power Query user interface, you can also switch what views are displayed. To switch views, go to the **View** tab in the ribbon and you'll find the **Preview** and **Layout** groups, which control how the Power Query user interface will look.

You're encouraged to try all of these options to find the view and layout that you feel most comfortable working with. As an example, select **Schema view** from the ribbon.

Queries [1] > fx Source{[Name = "Customers", Signature = "table"]}[Data]

	ab cd CompanyName	ab cd ContactName	ab cd ContactTitle
1 ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative
2 ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
3 ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner
4 AROUT	Around the Horn	Thomas Hardy	Sales Representative
5 BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator

The right side of the status bar also contains icons for the diagram, data, and schema views. You can use these icons to change between views. You can also use these icons to enable or disable the view of your choice.



What is schema view

The schema view offers you a quick and straightforward way to interact only with the components of the schema for your table, such as the column names and data types. We recommend the schema view when you want to do schema-related actions, such as removing columns, renaming columns, changing column data types, reordering columns, or duplicating columns.

NOTE

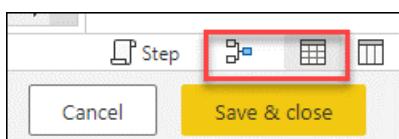
To learn more about schema view, see [Using Schema view](#).

For example, in schema view, select the check mark next to the **Orders** and **CustomerDemographics** columns, and from the ribbon select the **Remove columns** action. This selection applies a transformation to remove these columns from your data.

The screenshot shows the Power Query - Edit query interface. The ribbon at the top has tabs: Home, Transform, Add column, View, Help, and Schema tools. Under the Schema tools tab, there are several icons: Get data, Enter data, Options, Manage parameters, Refresh, Properties, Advanced editor, Manage, Choose columns, Remove columns, Keep rows, Remove rows, and Sort. The 'Remove columns' icon is highlighted with a red box. Below the ribbon is a toolbar with icons for New query, Options, Parameters, and Query. The main area is titled 'Source{[Name = "Customers", Signature = "table"]}[Dat' and shows a list of columns: Name, CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax, Orders, and CustomerDemographics. The 'Orders' and 'CustomerDemographics' columns are selected. At the bottom right of the main area, there is a 'Remove other columns' button.

What is diagram view

You can now switch back to the data preview view and enable diagram view to see a more visual perspective of your data and query.



The diagram view helps you visualize how your query is structured and how it might interact with other queries in your project. Each step in your query has a distinct icon to help you recognize the transform that was used. There are also lines that connect steps to illustrate dependencies. Since both data preview view and diagram view are enabled, the diagram view displays on top of the data preview.

Power Query - Edit queries

Queries [1]

Customer

OData Navigation Remove columns

Table.RemoveColumns(Navigation, {"Orders", "CustomerDemographics"})

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Oberre Str. 57	Berlin	null	12209	Germany	(030) 0074321
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	null	05021	Mexico	(5) 555-4729
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	null	05023	Mexico	(5) 555-3932
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	null	W1A 1DP	UK	(171) 555-7788
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvägen 8	Luleå	null	S-958 22	Sweden	0921-12 34 65
6	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany	0621-08460
7	BLONP	Blondesdöss père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	null	67000	France	88.60.15.31
8	BOLID	Bólido Comidas preparadas	Martin Sommer	Owner	C/ Aragó, 67	Madrid	null	28023	Spain	(91) 555-22 82
9	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	null	13008	France	91.24.45.40
10	BOTTOM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729
11	BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	null	EC2 5NT	UK	(171) 555-1212
12	CACTU	Cactus Comidas para llevar	Patricia Simpson	Sales Agent	Cerrito 333	Buenos Aires	null	1010	Argentina	(1) 135-5555
13										

Columns: 11 Rows: 13

Step Cancel Save & close

NOTE

To learn more about diagram view, see [Diagram view](#).

Begin transforming your data

With diagram view enabled, select the plus sign. You can search for a new transform to add to your query.

Search for **Group by** and select the transform.

Customer

OData Navigation Remove columns

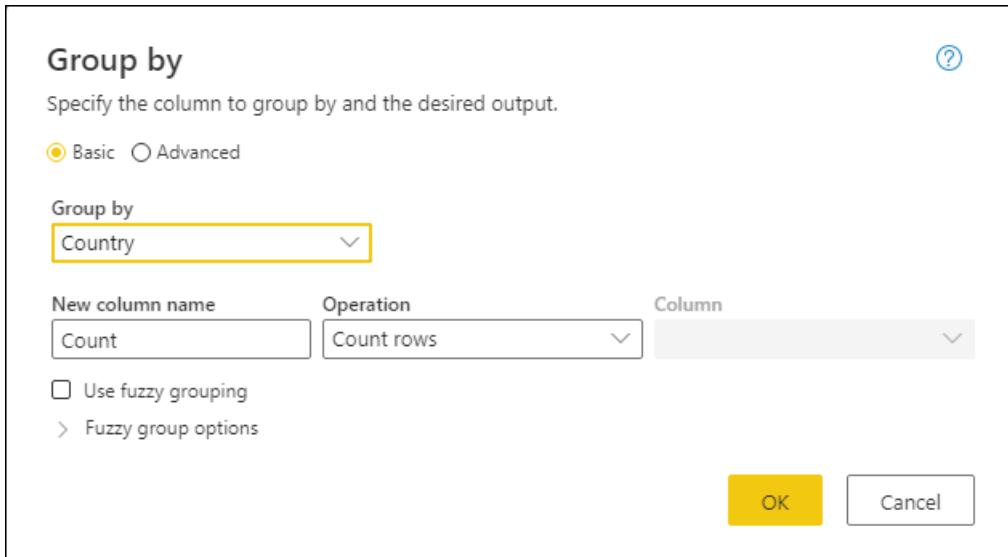
group

Transform table

Group by

The **Group by** dialog then appears. You can set the **Group by** operation to group by the country and count the number of customer rows per country.

1. Keep the **Basic** radio button selected.
2. Select **Country** to group by.
3. Select **Customers** and **Count rows** as the column name and operation respectively.



Select **OK** to perform the operation. Your data preview refreshes to show the total number of customers by country.

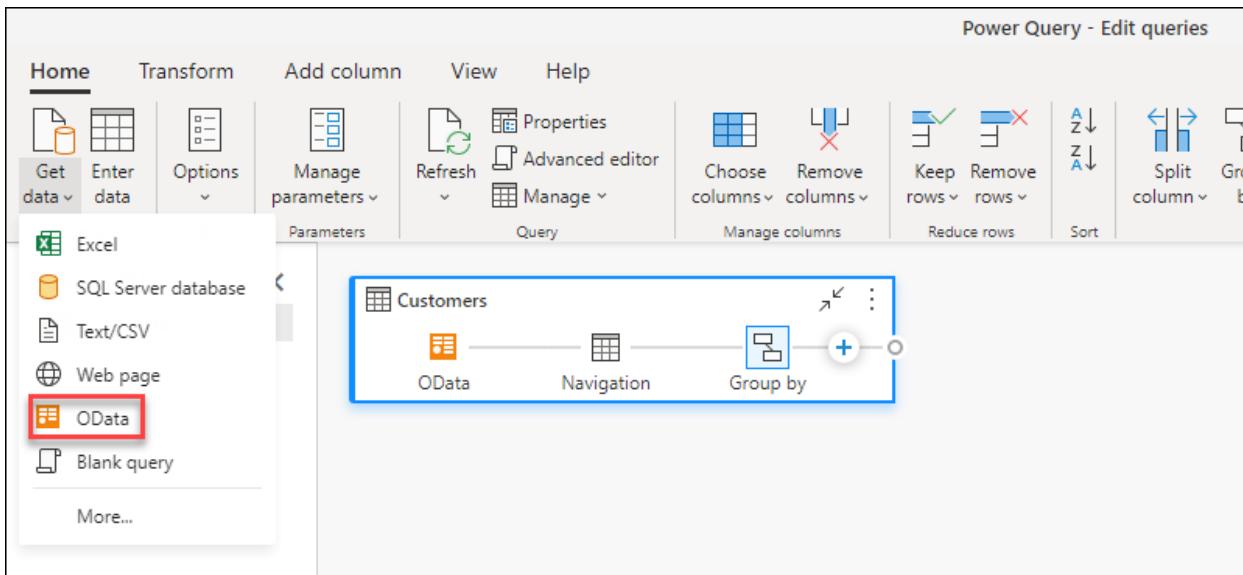
An alternative way to launch the **Group by** dialog would be to use the **Group by** button in the ribbon or by right-clicking the **Country** column.

For convenience, transforms in Power Query can often be accessed from multiple places, so users can opt to use the experience they prefer.

Adding a new query

Now that you have a query that provides the number of customers per country, you can add context to this data by finding the total number of suppliers for each territory.

First, you'll need to add the **Suppliers** data. Select **Get Data** and from the drop-down menu, and then select **OData**.



The OData connection experience reappears. Enter the connection settings as described in [Connect to an OData feed](#) to connect to the Northwind OData feed. In the **Navigator** experience, search for and select the **Suppliers** table.

SupplierID	CompanyName	ContactName	ContactTitle	Address
1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	49 Gilbert St.
2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	P.O. Box 78934
3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	707 Oxford Rd.
4	Tokyo Traders	Yoshi Nagase	Marketing Manager	9-8 Sekimai Musashino-shi
5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saave...	Export Administrator	Calle del Rosal 4
6	Mayumi's	Máyumi Ohno	Marketing Representative	92 Setsuko Chuo-ku
7	Pavlova, Ltd.	Ian Deviling	Marketing Manager	74 Rose St. Moonie Ponds
8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	29 King's Way
9	PB Knäckebroöd AB	Lars Peterson	Sales Agent	Kalaodagatan 13
10	Refrescos Americanas LTDA	Carlos Diaz	Marketing Manager	Av. das Americanas 12.890
11	Heli Süßwaren GmbH & Co. KG	Petra Winkler	Sales Manager	Tiergartenstraße 5
12	Plutzer Lebensmittelgroßmärkte AG	Martin Bein	International Marketing M...	Bogenallee 51
13	Nord-Ost-Fisch Handelsgesellschaft m...	Sven Petersen	Coordinator Foreign Mark...	Frahmredder 112a
14	Formaggi Fortini s.r.l.	Elio Rossi	Sales Representative	Viale Dante, 75
15	Norske Meierier	Beate Vileid	Marketing Manager	Høstvegen 5
16	Bigfoot Breweries	Cheryl Saylor	Regional Account Rep.	3400 - 8th Avenue Suite 210
17	Svensk Sjöföda AB	Michael Björn	Sales Representative	Brovallavägen 231
18	Aux joyeux ecclésiastiques	Guyène Nodier	Sales Manager	203, Rue des Francs-Bourgeois
19	New England Seafood Cannery	Robb Merchant	Wholesale Account Agent	Order Processing Dept. 2100 Paul Re
20	Leka Trading	Chandra Leka	Owner	471 Serangoon Loop, Suite #402
21	Lyngby Silde	Niels Petersen	Sales Manager	Lyngbysild Fiskebakken 10

Select **Create** to add the new query to the Power Query editor. The queries pane should now display both the **Customers** and the **Suppliers** query.

Power Query - Edit queries

Queries [2]

- Customers
- Suppliers

Transform

Applied steps

- Source
- Navigation

Query settings

Name: Suppliers
Entity type: Custom

Data

SupplierID	CompanyName	ContactName	ContactTitle	Address	City
1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	49 Gilbert St.	London
2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	P.O. Box 78934	New Orleans
3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	707 Oxford Rd.	Ann Arbor
4	Tokyo Traders	Yoshi Nagase	Marketing Manager	9-8 Sekimai Musashino-shi	Tokyo
5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	Export Administrator	Calle del Rosario 12	Oviedo
6	Mayumi's	Mayumi Ohno	Marketing Representative	92 Setsuko Chuo-ku	Osaka
7	Pavlova, Ltd.	Ian Devling	Marketing Manager	74 Rose St. Moonie Ponds	Melbourne
8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	29 King's Way	Manchester
9	PB Knäckebrot AB	Lars Peterson	Sales Agent	Kalodagatan 13	Göteborg
10	Refracs Americanas LTDA	Carlos Diaz	Marketing Manager	Av. das Americas 12,890	Sao Paulo
11	Heli Süßwaren GmbH & Co. KG	Petra Winkler	Sales Manager	Tiergartenstraße 5	Berlin
12	Plutzer Lebensmittelgroßmärkte AG	Martin Bein	International Marketing M...	Eogenallee 51	Frankfurt

Step **Cancel** **Save & close**

Open the **Group by** dialog again, this time by selecting the **Group by** button on the ribbon under the **Transform** tab.

Transform

Group by

In the **Group by** dialog, set the **Group by** operation to group by the country and count the number of supplier rows per country.

- Keep the **Basic** radio button selected.
- Select **Country** to group by.
- Select **Suppliers** and **Count rows** as the column name and operation respectively.

Group by

Specify the column to group by and the desired output.

Basic Advanced

Group by
Country

New column name Suppliers **Operation** Count rows **Column**

Use fuzzy grouping
Fuzzy group options

OK **Cancel**

NOTE

To learn more about the **Group by** transform, see [Grouping or summarizing rows](#).

Referencing queries

Now that you have a query for customers and a query for suppliers, your next goal is to combine these queries into one. There are many ways to accomplish this, including using the **Merge** option in the **Customers** table, duplicating a query, or referencing a query. For this example, you'll create a reference by right-clicking the **Customers** table and selecting **Reference**, which effectively creates a new query that references the **Customers** query.

The screenshot shows the Power Query Editor interface with two queries listed in the 'Queries [2]' pane: 'Customers' and 'Suppliers'. The 'Customers' query is selected. A context menu is open over the 'Customers' table, with the 'Reference' option highlighted by a red box. Other options in the menu include 'Copy', 'Paste', 'Delete', 'Rename', 'Enable load' (which is checked), 'Duplicate', 'Move to group', 'Move up', 'Move down', 'Create function...', 'Convert to parameter', 'Advanced editor', and 'Properties...'. Below the menu, the 'Country Analysis' query is shown, which contains a table with columns 'Country' and 'Customers'. The data in the table is as follows:

Country	Customers
Germany	11
Mexico	5
UK	7
Sweden	2
France	11
Spain	5
Canada	3
Argentina	3
Switzerland	2
Brazil	9
Austria	2
Italy	3
Portugal	2

After creating this new query, change the name of the query to **Country Analysis** and disable the load of the **Customers** table by unmarking the **Enable load** option from the **Suppliers** query.

The screenshot shows the Power Query Editor with the 'Transform' ribbon tab selected. In the 'Queries' pane, there are three queries: 'Customers', 'Suppliers', and 'Country Analysis'. The 'Country Analysis' query is currently selected. The 'Suppliers' query has a 'Reference' step added to it, which is highlighted with a blue box. The 'Country Analysis' query also has a 'Reference' step. The 'Country Analysis' table is expanded, showing a list of countries and their corresponding customer counts. The data is as follows:

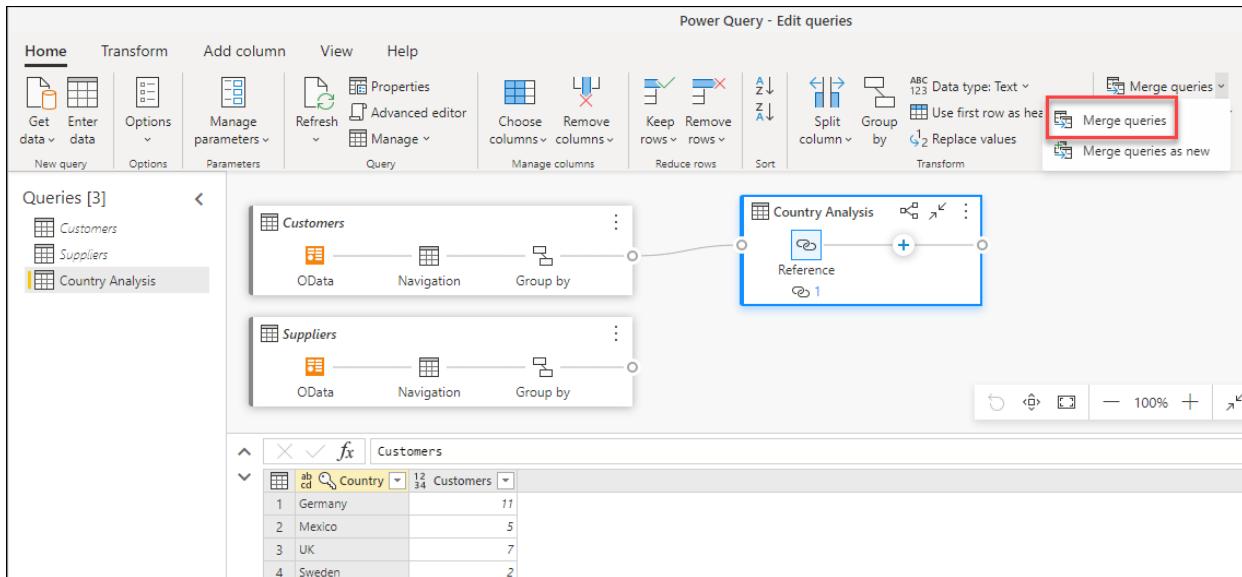
Country	Customers
Germany	11
Mexico	5
UK	7
Sweden	2
France	11
Spain	5
Canada	3
Argentina	3
Switzerland	2
Brazil	9
Austria	2
Italy	3
Portugal	2

The 'Query settings' pane on the right shows the 'Name' field set to 'Country Analysis'. The 'Applied steps' pane at the bottom shows the 'Source' step for the 'Suppliers' query.

Merging queries

A **merge queries** operation joins two existing tables together based on matching values from one or multiple columns. In this example, the goal is to join both the **Customers** and **Suppliers** tables into one table only for the countries that have both **Customers** and **Suppliers**.

Inside the **Country Analysis** query, select the **Merge queries** option from the Home tab in the ribbon.



The screenshot shows the Power Query - Edit queries interface. The ribbon at the top has tabs for Home, Transform, Add column, View, and Help. Under the Home tab, there are buttons for Get data, Options, Manage parameters, Refresh, Advanced editor, Properties, and a Merge queries button. The Merge queries button is highlighted with a red box. Below the ribbon is a list of Queries [3]: Customers, Suppliers, and Country Analysis. The Country Analysis query is selected and shown in a preview pane. The preview shows two tables: Customers (OData, Navigation, Group by) and Suppliers (OData, Navigation, Group by). A merge operation is shown between them, leading to the Country Analysis query, which contains a Reference step and a step with a value of 1. The preview pane also shows a table for the Customers query with columns Country and Customers, containing data for Germany, Mexico, UK, and Sweden.

A new dialog for the **Merge** operation appears. You can then select the query to merge with your current query. Select the **Suppliers** query and select the **Country** field from both queries. Finally, select the **Inner** join kind, as you only want the countries where you have **Customers** and **Suppliers** for this analysis.

Merge

Select a table and matching columns to create a merged table.

Country Analysis

ab cd	Country	1 ² 3 ⁴	Customers
Germany		11	
Mexico		5	
UK		7	
Sweden		2	
France		11	

Right table for merge

ab cd	Country	1 ² 3 ⁴	Suppliers
UK		2	
USA		4	
Japan		2	
Spain		1	
Australia		2	

Join kind

- Left outer
- Right outer
- Full outer
- Inner
- Left anti
- Right anti

Use fuzzy matching to perform the merge

> Fuzzy matching options

✓ The selection matches 12 rows from both the tables

OK **Cancel**

After selecting the **OK** button, a new column is added to your **Country Analysis** query that contains the data from the **Suppliers** query. Select the icon next to the **Suppliers** field, which displays a menu where you can select which fields you want to expand. Select only the **Suppliers** field, and then select the **OK** button.

Table.NestedJoin(Source, {"Country"}, Suppliers, {"Country"}, "Suppliers", JoinKind.Inner)

ab cd	Country	1 ² 3 ⁴	Customers	Suppliers
1	Germany		11	[Table]
2	UK		7	[Table]
3	Sweden		2	[Table]
4	USA		13	[Table]
5	France		11	[Table]
6	Spain		5	[Table]
7	Canada		3	[Table]
8	Brazil		9	[Table]
9	Italy		3	[Table]
10	Norway		1	[Table]
11	Denmark		2	[Table]
12	Finland		2	[Table]

Columns: 3 Rows: 12

(Select all)

Country

Suppliers

Use original column name as prefix

OK **Cancel**

The result of this **expand** operation is a table with only 12 rows. Rename the **Suppliers.Suppliers** field to just **Suppliers** by double-clicking the field name and entering the new name.

fx Table.RenameColumns(#"Expanded Suppliers", {"Suppliers.Suppliers", "Suppliers"})

	ab cd Country	12 34 Customers	12 34 Suppliers
1	Germany	11	3
2	UK	7	2
3	Sweden	2	2
4	USA	13	4
5	France	11	3
6	Spain	5	1
7	Canada	3	2
8	Brazil	9	1
9	Italy	3	2
10	Norway	1	1
11	Denmark	2	1
12	Finland	2	1

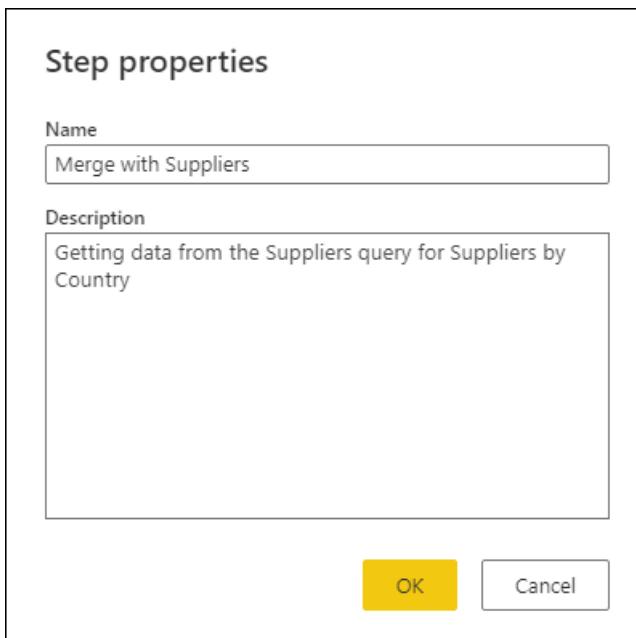
NOTE

To learn more about the **Merge queries** feature, see [Merge queries overview](#).

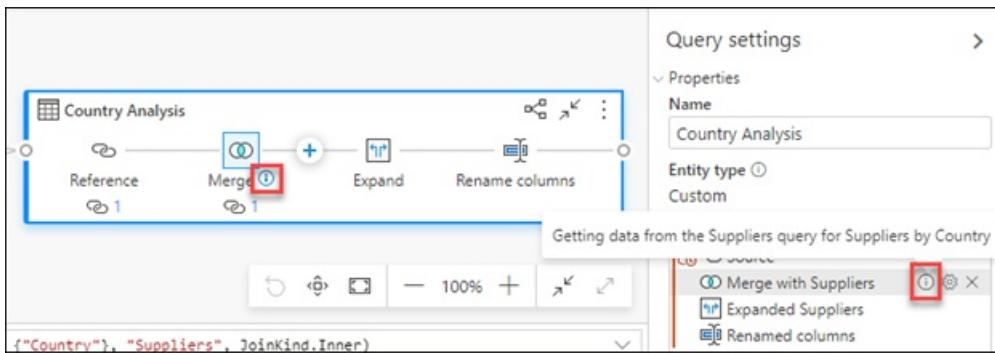
Applied steps

Every transformation that is applied to your query is saved as a step in the **Applied steps** section of the query settings pane. If you ever need to check how your query is transformed from step to step, you can select a step and preview how your query resolves at that specific point.

You can also right-click a query and select the **Properties** option to change the name of the query or add a description for the query. For example, right-click the **Merge queries** step from the **Country Analysis** query and change the name of the query to be **Merge with Suppliers** and the description to be **Getting data from the Suppliers query for Suppliers by Country**.



This change adds a new icon next to your step that you can hover over to read its description.



NOTE

To learn more about **Applied steps**, see [Using the Applied Steps list](#).

Before moving on to the next section, disable the **Diagram view** to only see the **Data preview**.

Adding a new column

With the data for customers and suppliers in a single table, you can now calculate the ratio of customers-to-suppliers for each country. Select the last step of the **Country Analysis** query, and then select both the **Customers** and **Suppliers** columns. In the **Add column** tab in the ribbon and inside the **From number** group, select **Standard**, and then **Divide (Integer)** from the dropdown.

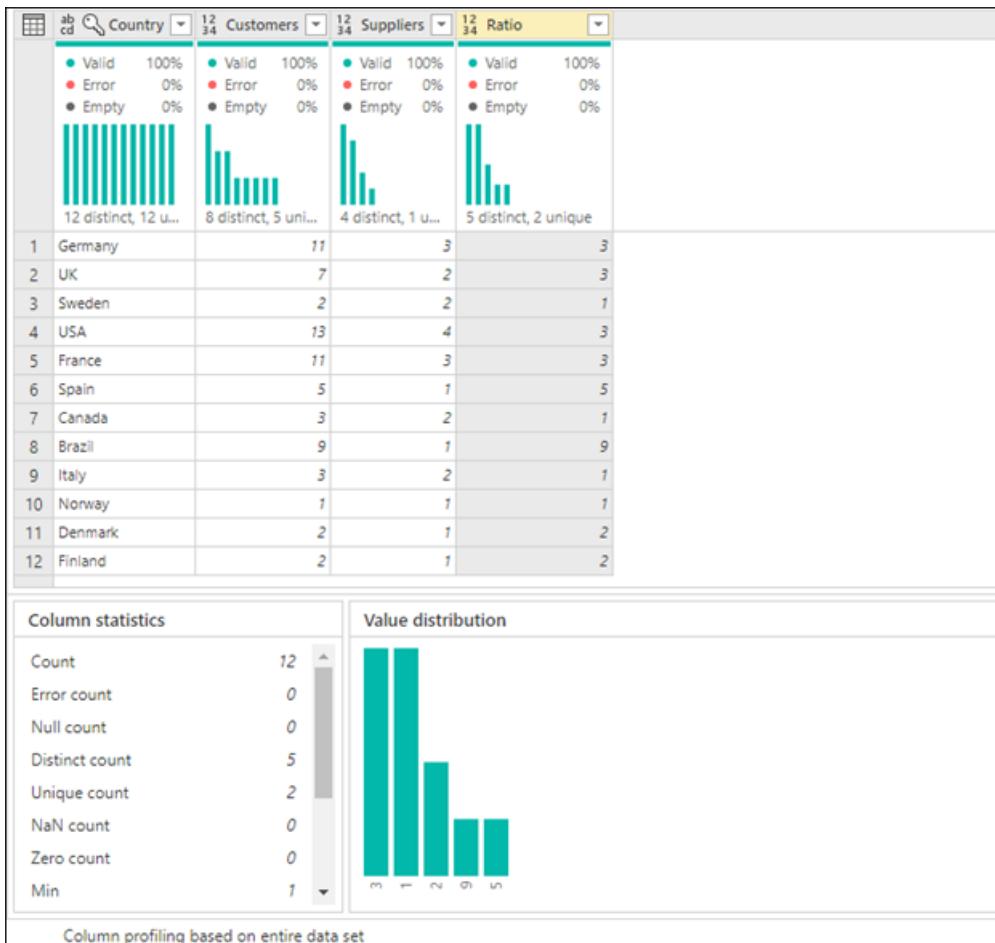
The screenshot shows the 'Power Query - Edit queries' window. The ribbon is visible with tabs Home, Transform, Add column, View, and Help. The 'Add column' tab is selected. On the far right, there's a contextual menu with options like Trigonometry, Rounding, Information, and Date and time column. Below the ribbon is a toolbar with icons for Conditional column, Merge columns, Format, Parse, Cluster values, and statistical functions. The main area shows a table with three columns: 'Country', 'Customers', and 'Suppliers'. A context menu is open over the 'Divide' option in the 'From number' group of the 'Add column' ribbon. The menu items are Add, Multiply, Subtract, Divide, Divide (Integer) (highlighted with a red box), Modulo, Percentage, and Percent of. The data preview shows the table with a new column 'Suppliers' added.

This change creates a new column called **Integer-division** that you can rename to **Ratio**. This change is the final step of your query as you can see the customer-to-supplier ratio for the countries where the data has customers and suppliers.

Data profiling

Another Power Query feature that can help you better understand your data is **Data Profiling**. By enabling the data profiling features, you'll get feedback about the data inside your query fields, such as value distribution, column quality, and more.

We recommended that you use this feature throughout the development of your queries, but you can always enable and disable the feature at your convenience. The following image shows all the data profiling tools enabled for your **Country Analysis** query.



NOTE

To learn more about **Data profiling**, see [Using the data profiling tools](#).

Summary

In this article, you created a series of queries with Power Query that provides a customer-to-supplier ratio analysis at the country level for the Northwind corporation.

You learned the components of the Power Query user interface, how to create new queries inside the query editor, reference queries, merge queries, understand the applied steps section, add new columns, and how to use the data profiling tools to better understand your data.

Power Query is a powerful tool used to connect to many different data sources and transform the data into the shape you want. The scenarios outlined in this article are examples to show how users can use Power Query to transform raw data into important actionable business insights.

Using the Applied Steps list

1/15/2022 • 2 minutes to read • [Edit Online](#)

Any transformations to your data will show in the **Applied Steps** list. For instance, if you change the first column name, it will display in the **Applied Steps** list as **Renamed Columns**.

The screenshot shows the Power Query Editor interface. On the left is a data grid with 10 rows of data. The first column is labeled 'NewColumnName'. The columns are labeled Column2, Column3, Column4, and Column5. The data in Column2 consists of GUIDs. The data in Column3 consists of dates. The data in Column4 and Column5 consists of zeros. To the right of the data grid is a 'Query Settings' pane. In the 'APPLIED STEPS' section, there is a list with one item: 'Renamed Columns'. This item is highlighted with a red box.

Selecting any step will show you the results of that particular step, so you can see exactly how your data changes as you add steps to the query.

Access the Applied Steps list

Select the **View** tab from the ribbon, and then select **Query Settings**.

The screenshot shows the Power BI ribbon. The 'View' tab is highlighted with a red box. Below the ribbon is a 'Layout' pane. In the 'Layout' pane, there is a 'Query Settings' icon (a document with a gear) which is also highlighted with a red box. To the right of the 'Layout' pane are several other settings options: 'Formula Bar' (checked), 'Monospaced' (unchecked), 'Column distribution' (unchecked), 'Show whitespace' (checked), 'Column profile' (unchecked), and 'Column quality' (unchecked). Below the ribbon is a 'Queries [1]' bar. At the bottom of the screen is a formula bar with icons for clear, checkmark, and fx, followed by the formula '= Table.Tran'.

The **Query Settings** menu will open to the right with the **Applied Steps** list.

The screenshot shows the Power Query Editor interface. At the top, there are various settings like 'Formula Bar', 'Monospaced', 'Column distribution', etc. Below that is a 'Layout' ribbon with tabs for 'Data Preview', 'Columns', 'Parameters', 'Advanced', and 'Dependencies'. The main area displays a table with columns labeled 'Column1' through 'Column5'. The 'Applied Steps' pane on the right is highlighted with a red border. It contains a list of steps: 'Source', 'Changed Type', and 'Renamed Col'. A context menu is open over the 'Renamed Col' step, with the 'Rename' option highlighted by a red box.

Rename step

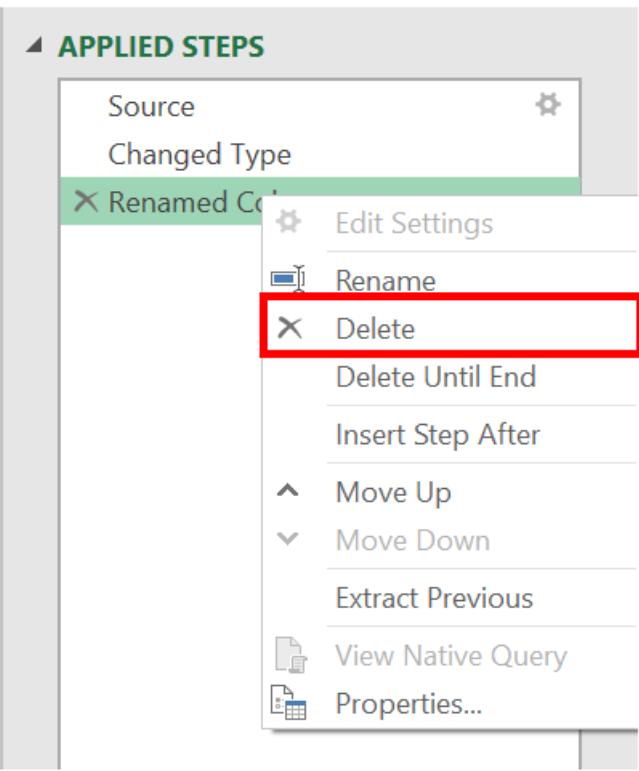
To rename a step, right-click the step and select **Rename**.

This screenshot shows the 'Renamed Col' step in the 'Applied Steps' list. A context menu is open over it, listing options like 'Edit Settings', 'Rename' (which is highlighted with a red box), 'Delete', 'Move Up', 'Move Down', 'Extract Previous', 'View Native Query', and 'Properties...'. The 'Edit Settings' option is also highlighted with a red box.

Enter in the name you want, and then either select **Enter** or click away from the step.

Delete step

To delete a step, right-click the step and select **Delete**.

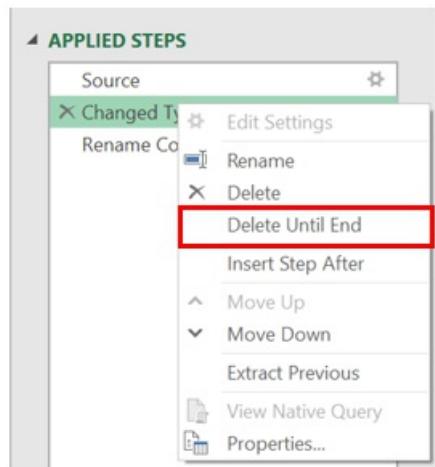


Alternatively, select the x next to the step.

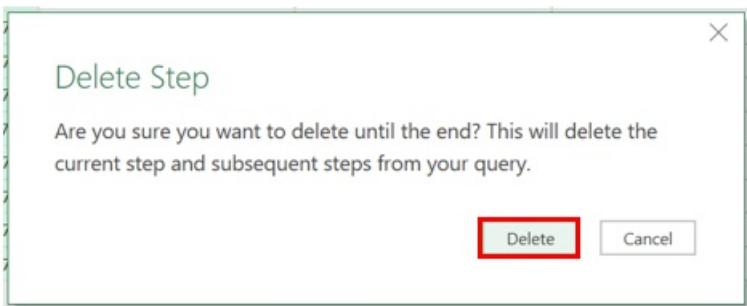


Delete until end

To delete a series of steps, right-click the first step of the series and select **Delete until end**. This action will delete the selected step and all the subsequent steps.



Select **Delete** in the new window.

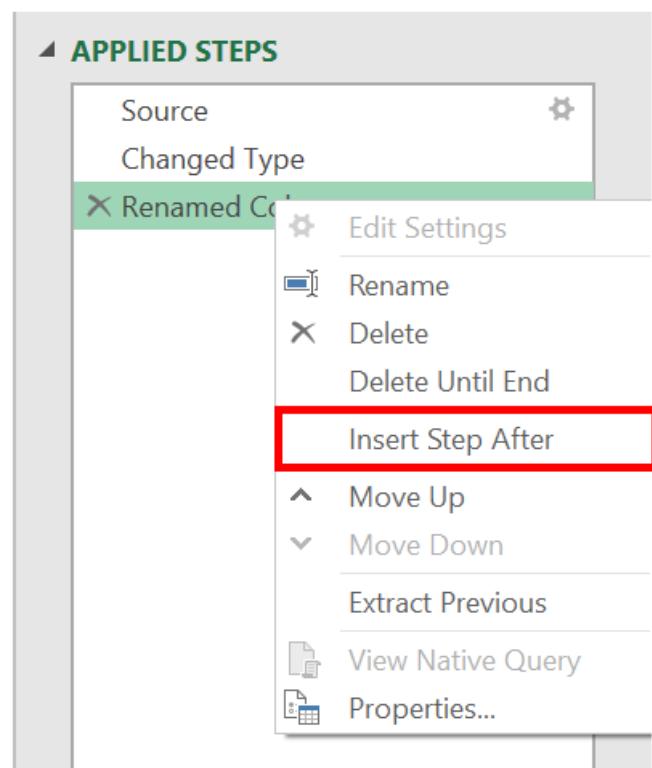


The following image shows the Applied steps list after using the **Delete until end**.

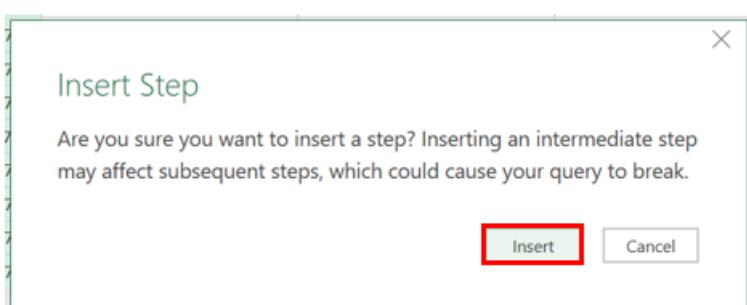


Insert step after

To add a new step, right-click on the last step in the list and select **Insert step after**.



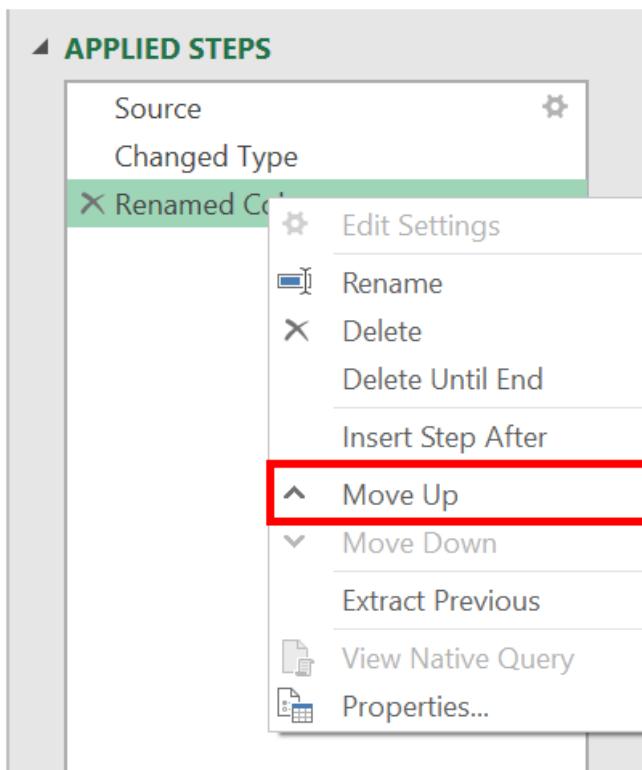
To insert a new intermediate step, right-click on a step and select **Insert step after**. Then select **Insert** on the new window.



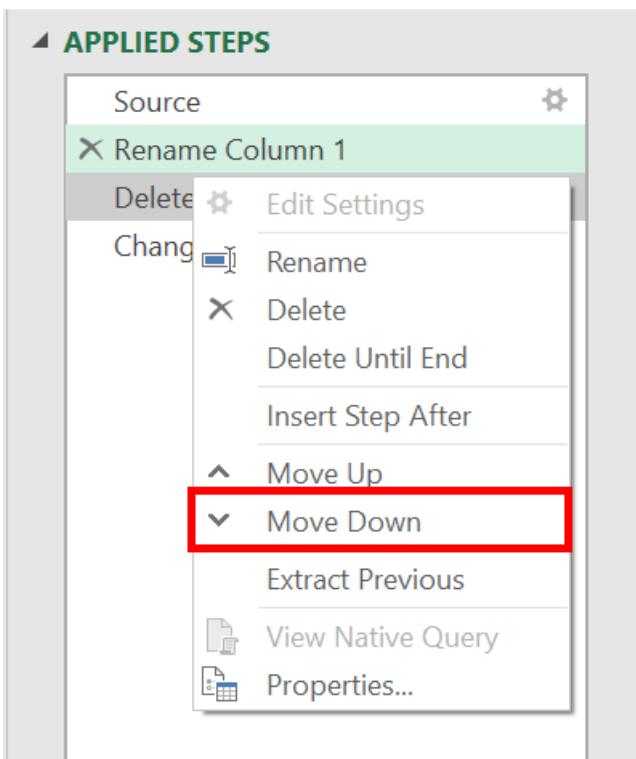
To set a transformation for the new step, select the new step in the list and make the change to the data. It will automatically link the transformation to the selected step.

Move step

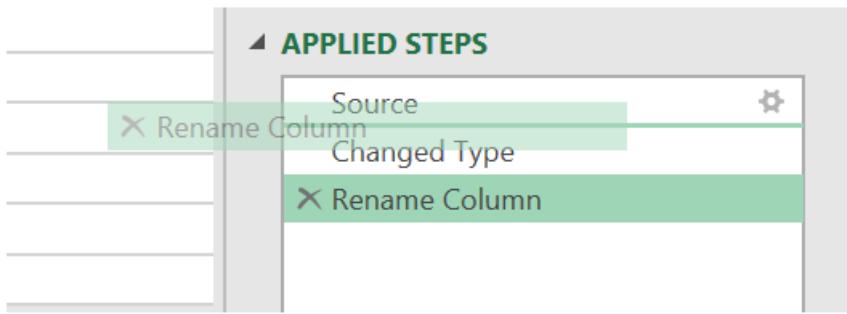
To move a step up one position in the list, right-click the step and select **Move up**.



To move a step down one position in the list, right-click the step and select **Move down**.

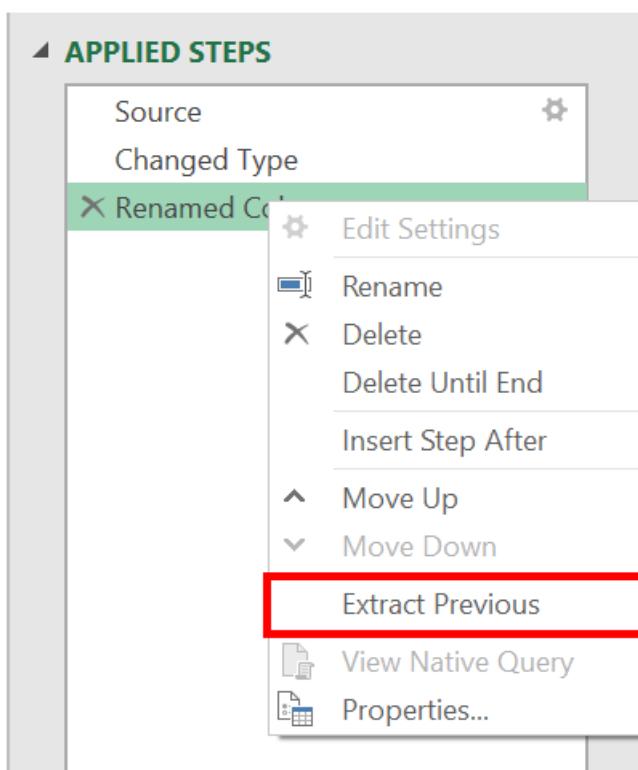


Alternatively, or to move more than a single position, drag and drop the step to the desired location.



Extract the previous steps into query

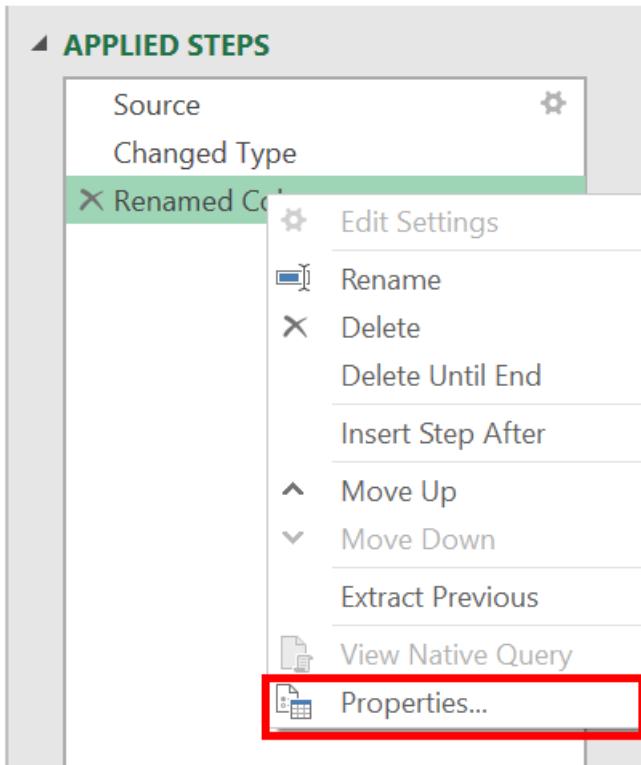
You can also separate a series of transformations into a different query. This allows the query to be referenced for other sources, which can be helpful if you're trying to apply the same transformation to multiple datasets. To extract all the previous steps into a new query, right-click the first step you do *not* want to include in the query and select **Extract Previous**.



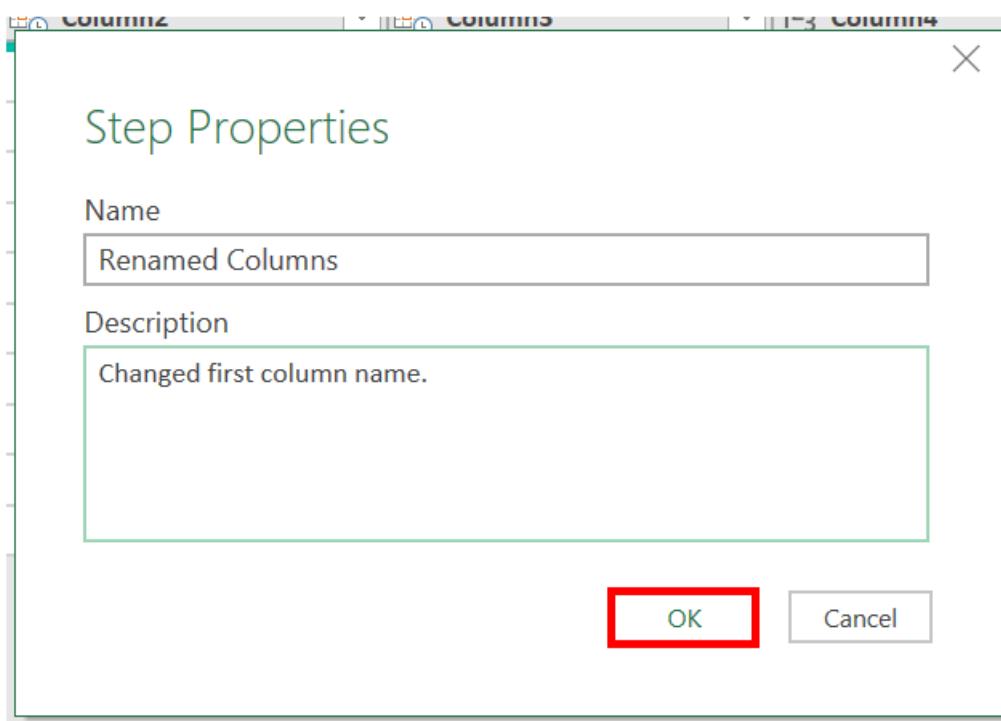
Name the new query and select **OK**. To access the new query, navigate to the **Queries pane** on the left side of the screen.

Edit step names and their descriptions

To edit the step, right-click the step and select **Properties**.

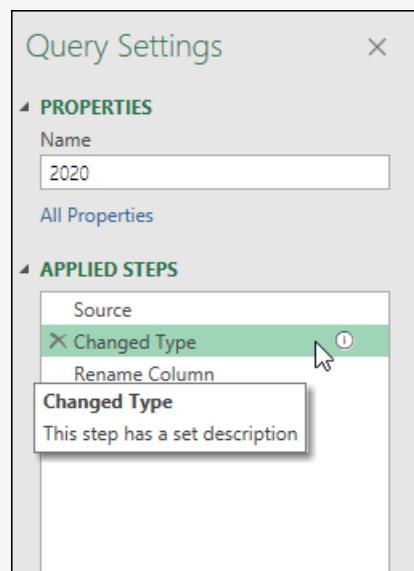


In the window, you can change the step name and description and save the changes by selecting OK.



NOTE

Adding a description to a step will add a small icon next to the step to denote that the step has a description. You can hover over this icon to display the description as a tooltip.



Overview of query evaluation and query folding in Power Query

1/15/2022 • 8 minutes to read • [Edit Online](#)

This article provides a basic overview of how M queries are processed and turned into data source requests.

Power Query M script

Any query, whether created by Power Query, manually written by you in the advanced editor, or entered using a blank document, consists of functions and syntax from the [Power Query M formula language](#). This query gets interpreted and evaluated by the Power Query engine to output its results. The M script serves as the set of instructions needed to evaluate the query.

TIP

You can think of the M script as a recipe that describes how to prepare your data.

The most common way to create an M script is by using the Power Query editor. For example, when you connect to a data source, such as a SQL Server database, you'll notice on the right-hand side of your screen that there's a section called [applied steps](#). This section displays all the steps or transforms used in your query. In this sense, the Power Query editor serves as an interface to help you create the appropriate M script for the transforms that you're after, and ensures that the code you use is valid.

NOTE

The M script is used in the Power Query editor to:

- Display the query as a series of steps and allow the creation or modification of new steps.
- Display a diagram view.

Power Query - Edit queries

Home Transform Add column View Help

Get data Enter data Options Manage parameters Query Manage columns Reduce rows Sort Transform Combine Map to entity CDM AI insights

Queries [4]

Table.FIRSTN(#"Sorted rows", 20)

	SalesOrderID	SalesOrderNumber	AccountNumber
1	75123	SO75123	10-4030-018759
2	75122	SO75122	10-4030-015868
3	75121	SO75121	10-4030-015251
4	75120	SO75120	10-4030-018749
5	75119	SO75119	10-4030-011981
6	75118	SO75118	10-4030-013671
7	75117	SO75117	10-4030-018178
8	75116	SO75116	10-4030-016402
9	75115	SO75115	10-4030-026832
10	75114	SO75114	10-4030-024704
11	75113	SO75113	10-4030-021524
12	75112	SO75112	10-4030-021523
13	75111	SO75111	10-4030-019072
14	75110	SO75110	10-4030-013753

Completed (1.28 s) Columns: 3 Rows: 20

Query settings

Properties Name: Full folding Entity type: Custom

Applied steps

- Source
- Navigation
- Removed other columns
- Sorted rows
- Kept top rows

Step Cancel Save & close

The previous image emphasizes the applied steps section, which contains the following steps:

- Source:** Makes the connection to the data source. In this case, it's a connection to a SQL Server database.
- Navigation:** Navigates to a specific table in the database.
- Removed other columns:** Selects which columns from the table to keep.
- Sorted rows:** Sorts the table using one or more columns.
- Kept top rows:** Filters the table to only keep a certain number of rows from the top of the table.

This set of step names is a friendly way to view the M script that Power Query has created for you. There are several ways to view the full M script. In Power Query, you can select **Advanced Editor** in the **View** tab. You can also select **Advanced Editor** from the **Query** group in the **Home** tab. In some versions of Power Query, you can also change the view of the formula bar to show the query script by going into the **View** tab and from the **Layout** group, select **Script view > Query script**.

Power Query - Edit queries

Home Transform Add column View Help

Get data Enter data Options Manage parameters Refresh Advanced editor Choose columns Remove columns Keep rows Remove rows Filter rows Sort Transform Combine Map to entity CDM AI insights

Queries [5]

```

1 let
2     Source = Sql.Database(ServerName, DatabaseName),
3     Navigation = Source[[Schema = "Sales", Item = "SalesorderHeader"]][Data],
4     #"Removed other columns" = Table.SelectColumns(Navigation, {"SalesOrderID", "SalesOrderNumber", "AccountNumber"}),
5     #"Sorted rows" = Table.Sort(#"Removed other columns", {{"SalesOrderID", Order.Descending}}),
6     #"Kept top rows" = Table.FirstN(#"Sorted rows", 20)
7 in
8     #"Kept top rows"

```

	SalesOrderID	SalesOrderNumber	AccountNumber
1	75123	SO75123	10-4030-018759
2	75122	SO75122	10-4030-015868
3	75121	SO75121	10-4030-015251
4	75120	SO75120	10-4030-018749
5	75119	SO75119	10-4030-011981
6	75118	SO75118	10-4030-013671
7	75117	SO75117	10-4030-018178
8	75116	SO75116	10-4030-016402
9	75115	SO75115	10-4030-026832
10	75114	SO75114	10-4030-024704
11	75113	SO75113	10-4030-021524

Completed (3.51 s) Columns: 3 Rows: 20

Query settings

Properties Name: Full folding Entity type: Custom

Applied steps

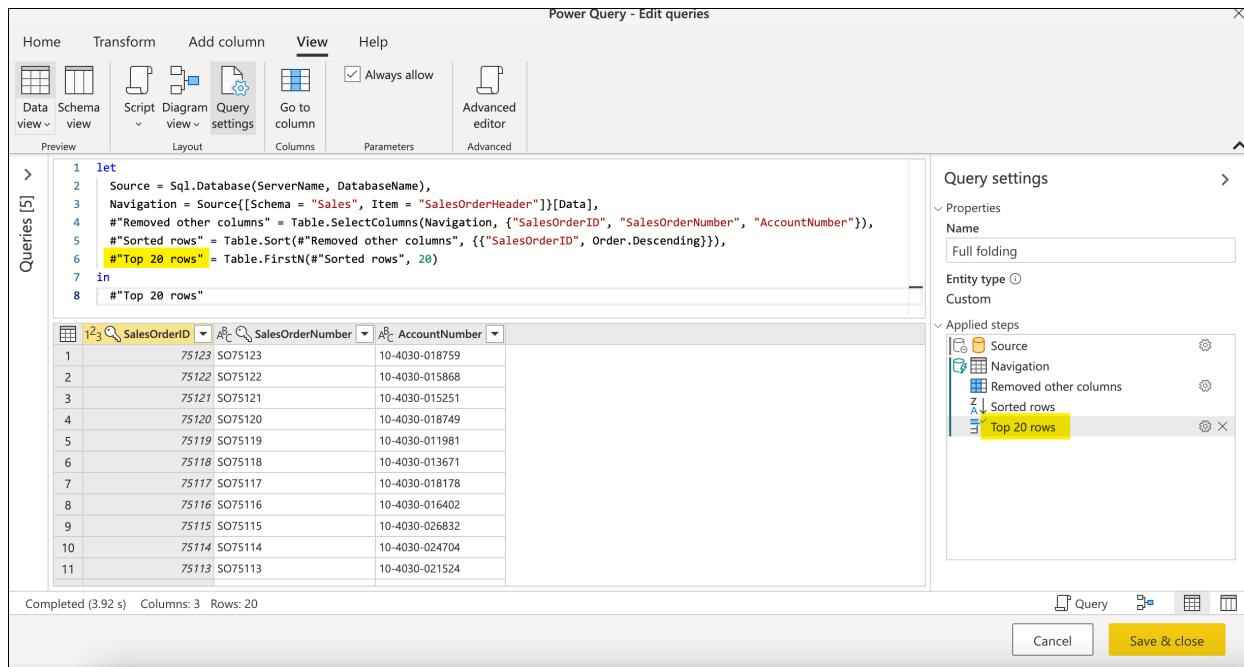
- Source
- Navigation
- Removed other columns
- Sorted rows
- Kept top rows

Query Cancel Save & close

Most of the names found in the Applied steps pane are also being used as is in the M script. Steps of a query are named using something called *identifiers* in the M language. Sometimes extra characters are wrapped around step names in M, but these characters aren't shown in the applied steps. An example is

`#"Kept top rows"`, which is categorized as a *quoted identifier* because of these extra characters. A quoted identifier can be used to allow any sequence of zero or more Unicode characters to be used as an identifier, including keywords, whitespace, comments, operators, and punctuators. To learn more about *identifiers* in the M language, go to [lexical structure](#).

Any changes that you make to your query through the Power Query editor will automatically update the M script for your query. For example, using the previous image as the starting point, if you change the **Kept top rows** step name to be **Top 20 rows**, this change will automatically be updated in the script view.



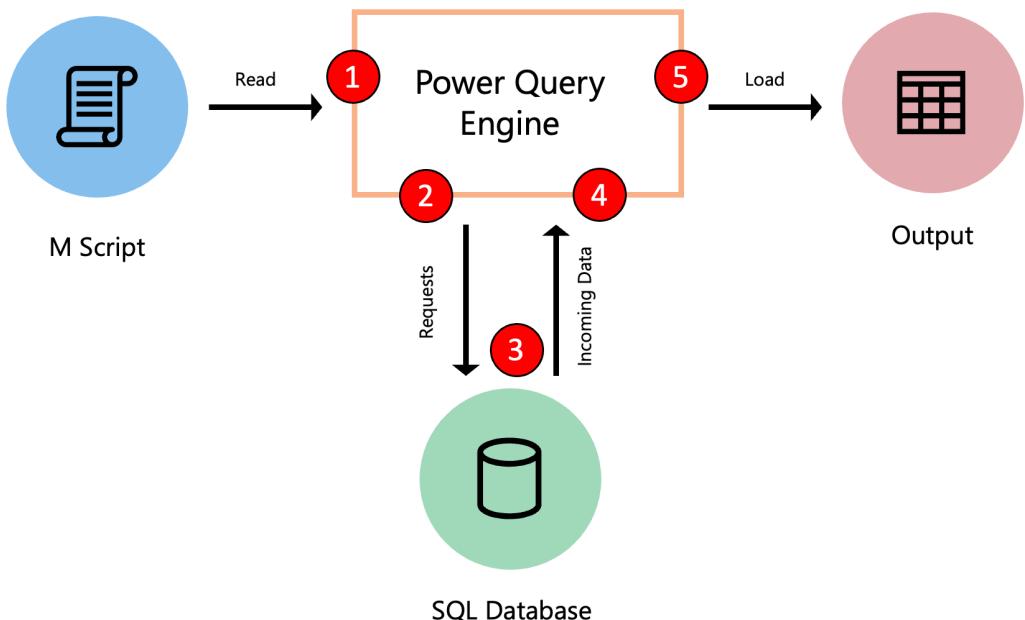
While we recommend that you use the Power Query editor to create all or most of the M script for you, you can manually add or modify pieces of your M script. To learn more about the M language, go to the [official docs site for the M language](#).

NOTE

M script, also referred to as M code, is a term used for any code that uses the M language. In the context of this article, M script also refers to the code found inside a Power Query query and accessible through the advanced editor window or through the script view in the formula bar.

Query evaluation in Power Query

The following diagram explores the process that occurs when a query is evaluated in Power Query.



1. The M script, found inside the advanced editor, is submitted to the Power Query engine. Other important information is also included, such as credentials and data source privacy levels.
2. Power Query determines what data needs to be extracted from the data source and submits a request to the data source.
3. The data source responds to the request from Power Query by transferring the requested data to Power Query.
4. Power Query receives the incoming data from the data source and does any transformations using the Power Query engine if necessary.
5. The results derived from the previous point are loaded to a destination.

NOTE

While this example showcases a query with a SQL Database as a data source, the concept applies to queries with or without a data source.

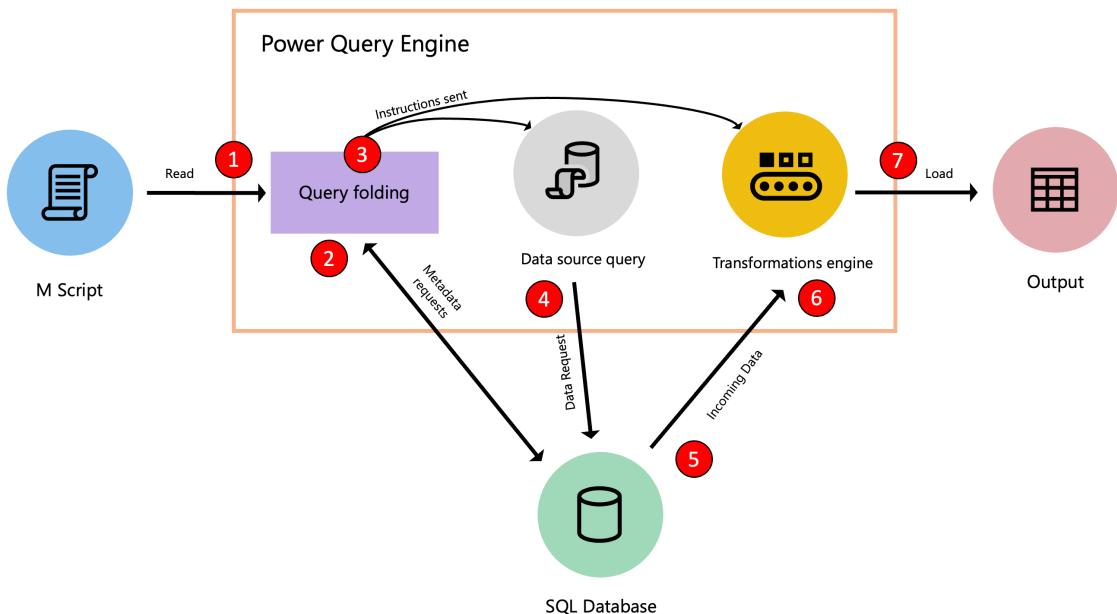
When Power Query reads your M script, it runs the script through an optimization process to more efficiently evaluate your query. In this process, it determines which steps (transforms) from your query can be offloaded to your data source. It also determines which other steps need to be evaluated using the Power Query engine. This optimization process is called *query folding*, where Power Query tries to push as much of the possible execution to the data source to optimize your query's execution.

IMPORTANT

All rules from the [Power Query M formula language](#) (also known as the M language) are followed. Most notably, *lazy evaluation* plays an important role during the optimization process. In this process, Power Query understands what specific transforms from your query need to be evaluated. Power Query also understands what other transforms don't need to be evaluated because they're not needed in the output of your query.

Furthermore, when multiple sources are involved, the data privacy level of each data source is taken into consideration when evaluating the query. More information: [Behind the scenes of the Data Privacy Firewall](#)

The following diagram demonstrates the steps that take place in this optimization process.



1. The M script, found inside the advanced editor, is submitted to the Power Query engine. Other important information is also supplied, such as credentials and data source privacy levels.
2. The Query folding mechanism submits metadata requests to the data source to determine the capabilities of the data source, table schemas, relationships between different entities at the data source, and more.
3. Based on the metadata received, the query folding mechanism determines what information to extract from the data source and what set of transformations need to happen inside the Power Query engine. It sends the instructions to two other components that take care of retrieving the data from the data source and transforming the incoming data in the Power Query engine if necessary.
4. Once the instructions have been received by the internal components of Power Query, Power Query sends a request to the data source using a data source query.
5. The data source receives the request from Power Query and transfers the data to the Power Query engine.
6. Once the data is inside Power Query, the transformation engine inside Power Query (also known as mashup engine) does the transformations that couldn't be folded back or offloaded to the data source.
7. The results derived from the previous point are loaded to a destination.

NOTE

Depending on the transformations and data source used in the M script, Power Query determines if it will stream or buffer the incoming data.

Query folding overview

The goal of query folding is to offload or push as much of the evaluation of a query to a data source that can compute the transformations of your query.

The query folding mechanism accomplishes this goal by translating your M script to a language that can be interpreted and executed by your data source. It then pushes the evaluation to your data source and sends the result of that evaluation to Power Query.

This operation often provides a much faster query execution than extracting all the required data from your data source and running all transforms required in the Power Query engine.

When you use the [get data experience](#), Power Query guides you through the process that ultimately lets you connect to your data source. When doing so, Power Query uses a series of functions in the M language categorized as [accessing data functions](#). These specific functions use mechanisms and protocols to connect to your data source using a language that your data source can understand.

However, the steps that follow in your query are the steps or transforms that the query folding mechanism attempts to optimize. It then checks if they can be offloaded to your data source instead of being processed using the Power Query engine.

IMPORTANT

All data source functions, commonly shown as the **Source** step of a query, queries the data at the data source in its native language. The query folding mechanism is utilized on all transforms applied to your query after your data source function so they can be translated and combined into a single data source query or as many transforms that can be offloaded to the data source.

Depending on how the query is structured, there could be three possible outcomes to the query folding mechanism:

- **Full query folding:** When all of your query transformations get pushed back to the data source and minimal processing occurs at the Power Query engine.
- **Partial query folding:** When only a few transformations in your query, and not all, can be pushed back to the data source. In this case, only a subset of your transformations is done at your data source and the rest of your query transformations occur in the Power Query engine.
- **No query folding:** When the query contains transformations that can't be translated to the native query language of your data source, either because the transformations aren't supported or the connector doesn't support query folding. For this case, Power Query gets the raw data from your data source and uses the Power Query engine to achieve the output you want by processing the required transforms at the Power Query engine level.

NOTE

The query folding mechanism is primarily available in connectors for structured data sources such as, but not limited to, [Microsoft SQL Server](#) and [OData Feed](#). During the optimization phase, the engine might sometimes reorder steps in the query.

Leveraging a data source that has more processing resources and has query folding capabilities can expedite your query loading times as the processing occurs at the data source and not at the Power Query engine.

Next steps

For detailed examples of the three possible outcomes of the query folding mechanism, go to [Query folding examples](#).

Query folding examples

1/15/2022 • 17 minutes to read • [Edit Online](#)

This article provides some example scenarios for each of the three possible outcomes for query folding. It also includes some suggestions on how to get the most out of the query folding mechanism, and the effect that it can have in your queries.

The scenario

Imagine a scenario where, using the [Wide World Importers database for Azure Synapse Analytics SQL database](#), you're tasked with creating a query in Power Query that connects to the `fact_Sale` table and retrieves the last 10 sales with only the following fields:

- Sale Key
- Customer Key
- Invoice Date Key
- Description
- Quantity

NOTE

For demonstration purposes, this article uses the database outlined on the tutorial on loading the Wide World Importers database into Azure Synapse Analytics. The main difference in this article is the `fact_Sale` table only holds data for the year 2000, with a total of 3,644,356 rows.

While the results might not exactly match the results that you get by following the tutorial from the Azure Synapse Analytics documentation, the goal of this article is to showcase the core concepts and impact that query folding can have in your queries.

	1 ² 3 Sale Key	1 ² 3 Customer Key	1 ² 3 Invoice Date Key	A ^B Description	1 ² 3 Quantity
1	3731521	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
2	3731461	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
3	3731401	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
4	3731341	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
5	3731281	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
6	3731221	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
7	3731161	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
8	3731101	191	12/30/2000	Developer joke mug - fun was unexpected at this time (Whi...	3
9	3731041	376	12/30/2000	"The Gu" red shirt XML tag t-shirt (Black) XXS	60
10	3730981	376	12/30/2000	"The Gu" red shirt XML tag t-shirt (Black) XXS	60

This article showcases three ways to achieve the same output with different levels of query folding:

- No query folding
- Partial query folding
- Full query folding

No query folding example

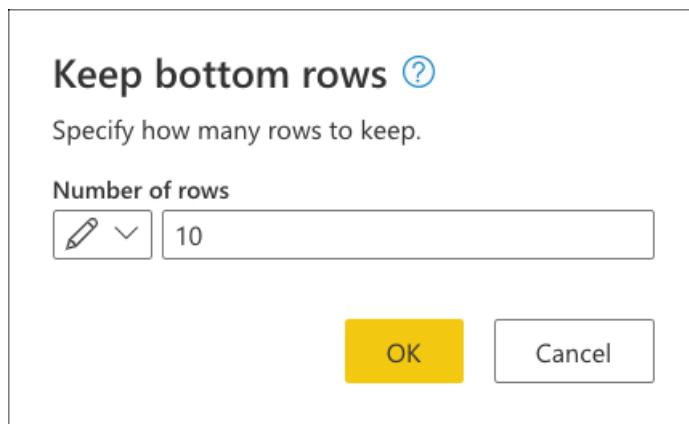
IMPORTANT

Queries that rely solely on unstructured data sources or that don't have a compute engine, such as CSV or Excel files, don't have query folding capabilities. This means that Power Query evaluates all the required data transformations using the Power Query engine.

After connecting to your database and navigating to the `fact_Sale` table, you select the **Keep bottom rows** transform found inside the **Reduce rows** group of the Home tab.

The screenshot shows the Power Query ribbon with the 'Home' tab selected. In the 'Transform' group, the 'Keep bottom rows' icon is highlighted with a red box. A dropdown menu is open, showing several options: 'Keep top rows', 'Keep bottom rows' (which is selected and highlighted with a red box), 'Keep range of rows', 'Keep duplicates', and 'Keep errors'. Below the dropdown, there are two buttons: 'OK' and 'Cancel'.

After selecting this transform, a new dialog appears. In this new dialog, you can enter the number of rows that you'd like to keep. For this case, enter the value 10, and then select OK.



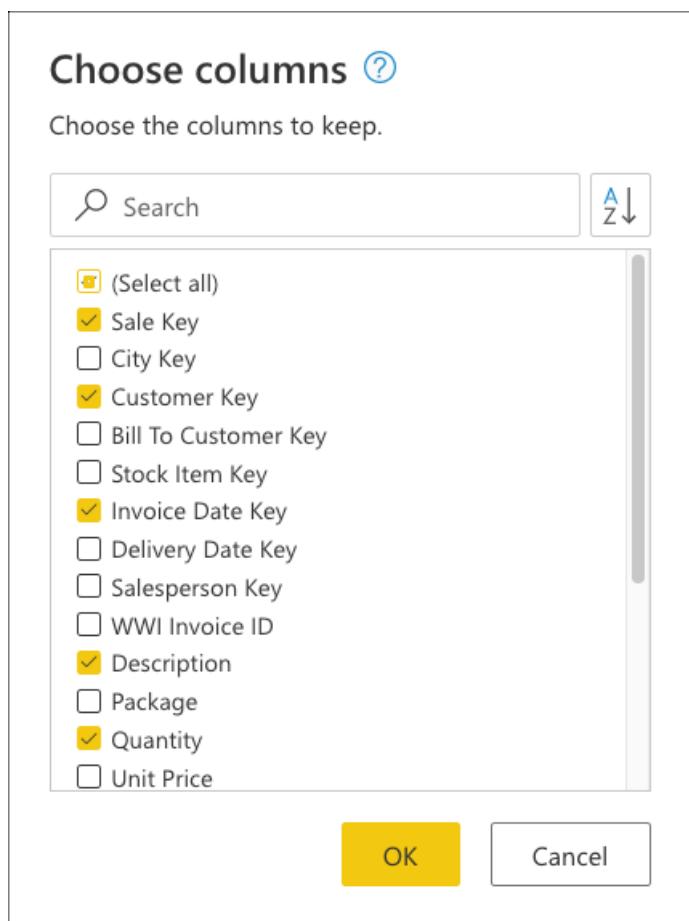
TIP

For this case, performing this operation yields the result of the last ten sales. In most scenarios, we recommend that you provide a more explicit logic that defines which rows are considered last by applying a sort operation on the table.

Next, select the **Choose columns** transform found inside the **Manage columns** group of the Home tab. You can then select the columns you want to keep from your table and remove the rest.

The screenshot shows the Power Query ribbon with the 'Home' tab selected. In the 'Manage columns' group, the 'Choose columns' icon is highlighted with a red box.

Lastly, inside the **Choose columns** dialog, select the `Sale Key`, `Customer Key`, `Invoice Date Key`, `Description`, and `Quantity` columns, and then select **OK**.

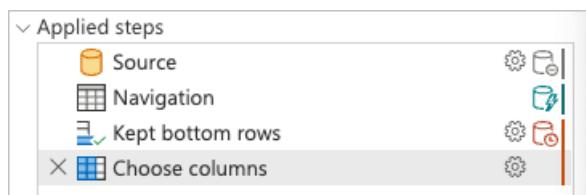


The following code sample is the full M script for the query you created:

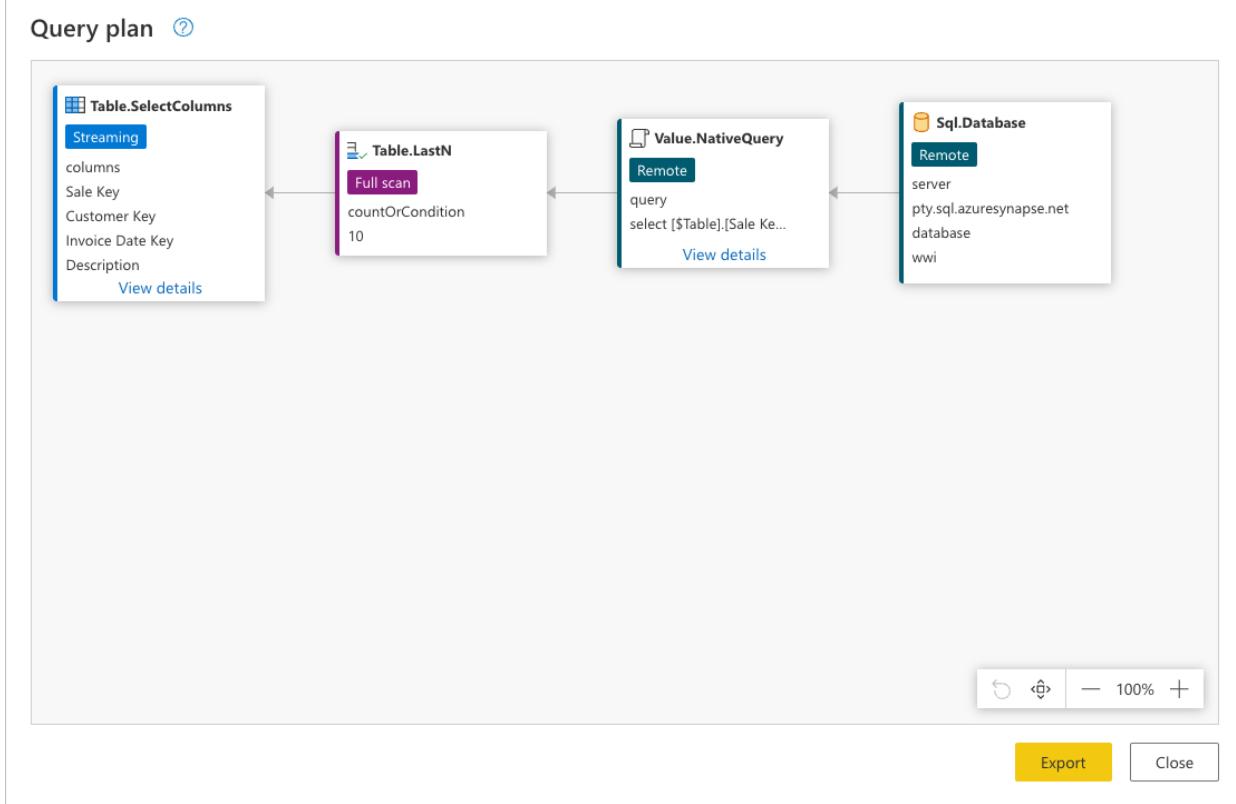
```
let
    Source = Sql.Database(ServerName, DatabaseName),
    Navigation = Source{[Schema = "wwi", Item = "fact_Sale"]}[Data],
    #"Kept bottom rows" = Table.LastN(Navigation, 10),
    #"Choose columns" = Table.SelectColumns(#"Kept bottom rows", {"Sale Key", "Customer Key", "Invoice Date Key", "Description", "Quantity"})
in
    #"Choose columns"
```

No query folding: Understanding the query evaluation

Under **Applied steps** in the Power Query editor, you'll notice that the step folding indicators for **Kept bottom rows** and **Choose columns** are marked as steps that will be evaluated outside the data source or, in other words, by the Power Query engine.

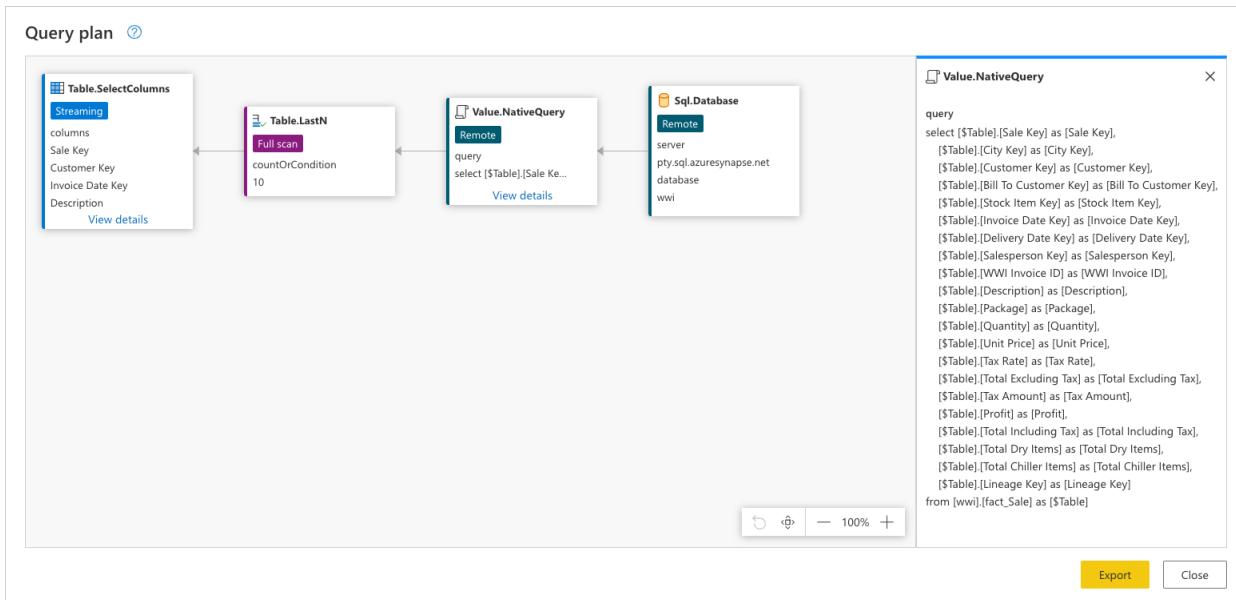


You can right-click the last step of your query, the one named **Choose columns**, and select the option that reads **View Query plan**. The goal of the query plan is to provide you with a detailed view of how your query is run. To learn more about this feature, go to [Query plan](#).



Each box in the previous image is called a *node*. A node represents the operation breakdown to fulfill this query. Nodes that represent data sources, such as SQL Server in the example above and the `Value.NativeQuery` node, represent which part of the query is offloaded to the data source. The rest of the nodes, in this case `Table.LastN` and `Table.SelectColumns` highlighted in the rectangle in the previous image, are evaluated by the Power Query engine. These two nodes represent the two transforms that you added, **Kept bottom rows** and **Choose columns**. The rest of the nodes represent operations that happen at your data source level.

To see the exact request that is sent to your data source, select **View details** in the `Value.NativeQuery` node.



This data source request is in the native language of your data source. For this case, that language is SQL and this statement represents a request for all the rows and fields from the `fact_Sale` table.

Consulting this data source request can help you better understand the story that the query plan tries to convey:

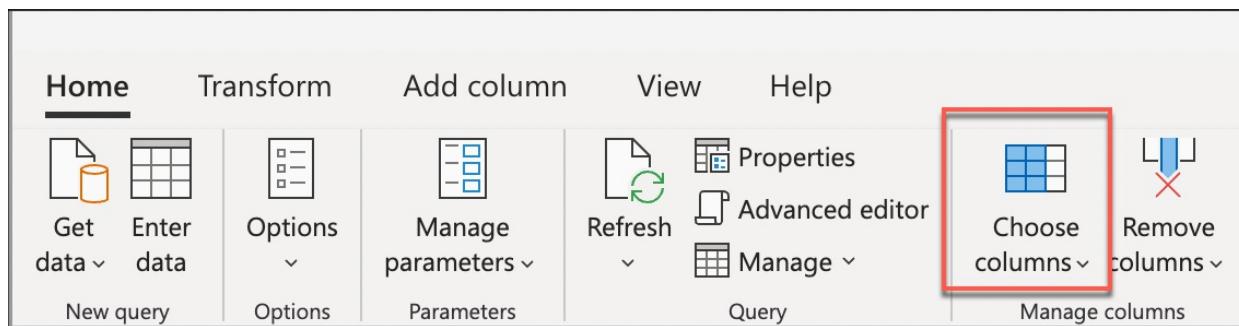
- `Sql.Database`: This node represents the data source access. Connects to the database and sends metadata requests to understand its capabilities.

- `Value.NativeQuery` : Represents the request that was generated by Power Query to fulfill the query. Power Query submits the data requests in a native SQL statement to the data source. In this case, that represents all records and fields (columns) from the `fact_Sale` table. For this scenario, this case is undesirable, as the table contains millions of rows and the interest is only in the last 10.
- `Table.LastN` : Once Power Query receives all records from the `fact_Sale` table, it uses the Power Query engine to filter the table and keep only the last 10 rows.
- `Table.SelectColumns` : Power Query will use the output of the `Table.LastN` node and apply a new transform called `Table.SelectColumns`, which selects the specific columns that you want to keep from a table.

For its evaluation, this query had to download all rows and fields from the `fact_Sale` table. This query took an average of 3 minutes and 4 seconds to be processed in a standard instance of Power BI dataflows (which accounts for the evaluation and loading of data to dataflows).

Partial query folding example

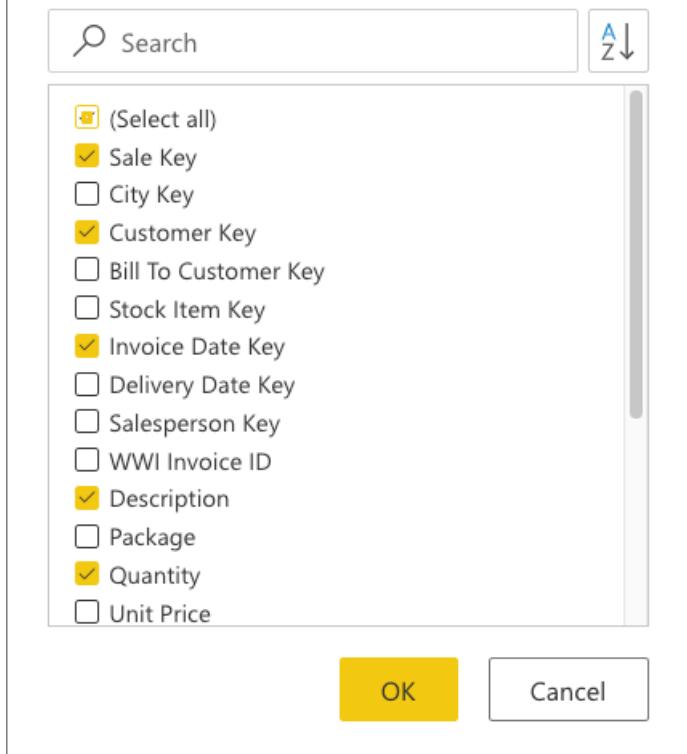
After connecting to the database and navigating to the `fact_Sale` table, you start by selecting the columns that you want to keep from your table. Select the **Choose columns** transform found inside the **Manage columns** group from the **Home** tab. This transform helps you to explicitly select the columns that you want to keep from your table and remove the rest.



Inside the **Choose columns** dialog, select the `Sale Key`, `Customer Key`, `Invoice Date Key`, `Description`, and `Quantity` columns and then select OK.

Choose columns [?](#)

Choose the columns to keep.



You now create logic that will sort the table to have the last sales at the bottom of the table. Select the `Sale Key` column, which is the primary key and incremental sequence or index of the table. Sort the table using only this field in ascending order from the context menu for the column.

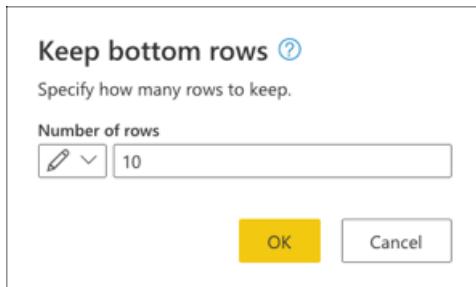
	123 Sale Key	123 Customer Key	Invoice Date Key	A B C Description
1		A Z ↓ Sort ascending		g - inheritance is the OO way
2		Z A ↑ Sort descending		White) 400L
3				e - pizza slice
4				lass with care despatch tape
5				(Gray) S
6				'Pink) M
7				ML tag t-shirt (Black) XXL
8				cket (Blue) S
9				vare: part of the computer th
10				cket (Blue) M
11				g - (hip, hip, array) (White)
12				ML tag t-shirt (White) L
13				metal insert blade (Yellow) 9m
14				blades 18mm
15				blue 5mm nib (Blue) 5mm
16				cket (Blue) S
17				oe 48mmx75m
18				owered slippers (Green) XL
19				ML tag t-shirt (Black) 5XL
20	20		304	1/1/2000 Shipping carton (Brown) 229x229x229mm

The context menu for the 'Sale Key' column is open, showing options: 'Sort ascending' (highlighted with a red box), 'Sort descending', 'Remove empty', and 'Number filters'. Below the table is a search bar and a list of selected values from 4 to 20. At the bottom are 'OK' and 'Cancel' buttons. A note says 'List may be incomplete.' and there is a 'Load more' link.

Next, select the table contextual menu and choose the **Keep bottom rows** transform.

	1 ² 3 Sale Key	1 ² 3 Customer Key	Invoice Date Key	Description
Copy preview data	191	1/1/2000	Developer joke mug - inheritance is t	
Use first row as headers	0	1/1/2000	Void fill 400 L bag (White) 400L	
Add custom column	380	1/1/2000	USB food flash drive - pizza slice	
Add conditional column	140	1/1/2000	Black and orange glass with care des	
Index column	8	1/1/2000	Plush shark slippers (Gray) S	
Choose columns	0	1/1/2000	Furry animal socks (Pink) M	
Keep top rows	264	1/1/2000	Superhero action jacket (Blue) S	
Keep bottom rows	0	1/1/2000	IT joke mug - hardware: part of the c	
Keep range of rows	290	1/1/2000	Superhero action jacket (Blue) M	
Keep duplicates	224	1/1/2000	Developer joke mug - (hip, hip, array)	
Keep errors	158	1/1/2000	"The Gu" red shirt XML tag t-shirt (W	
Remove top rows	389	1/1/2000	Packing knife with metal insert blade	
Remove bottom rows	393	1/1/2000	Large replacement blades 18mm	
Remove alternate rows	68	1/1/2000	Permanent marker blue 5mm nib (Blu	
Remove duplicates	27	1/1/2000	Superhero action jacket (Blue) S	
Remove errors	251	1/1/2000	Clear packaging tape 48mmx75m	
Merge queries	387	1/1/2000	Dinosaur battery-powered slippers (C	
Append queries	39	1/1/2000	"The Gu" red shirt XML tag t-shirt (Bl	
	304	1/1/2000	Shipping carton (Brown) 229x229x22	
	137	1/1/2000	Superhero action jacket (Blue) XL	
	22	22	0	Permanent marker black 5mm nib (Bl

In **Keep bottom rows**, enter the value 10, and then select **OK**.



The following code sample is the full M script for the query you created:

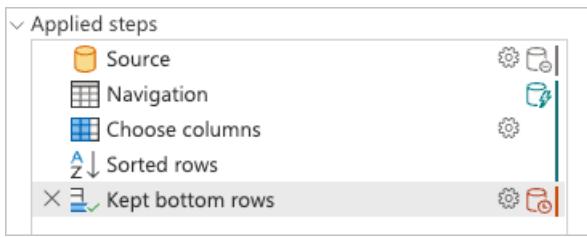
```

let
    Source = Sql.Database(ServerName, DatabaseName),
    Navigation = Source{[Schema = "wwi", Item = "fact_Sale"]}[Data],
    #"Choose columns" = Table.SelectColumns(Navigation, {"Sale Key", "Customer Key", "Invoice Date Key", "Description", "Quantity"}),
    #"Sorted rows" = Table.Sort(#"Choose columns", {"Sale Key", Order.Ascending}),
    #"Kept bottom rows" = Table.LastN(#"Sorted rows", 10)
in
    #"Kept bottom rows"

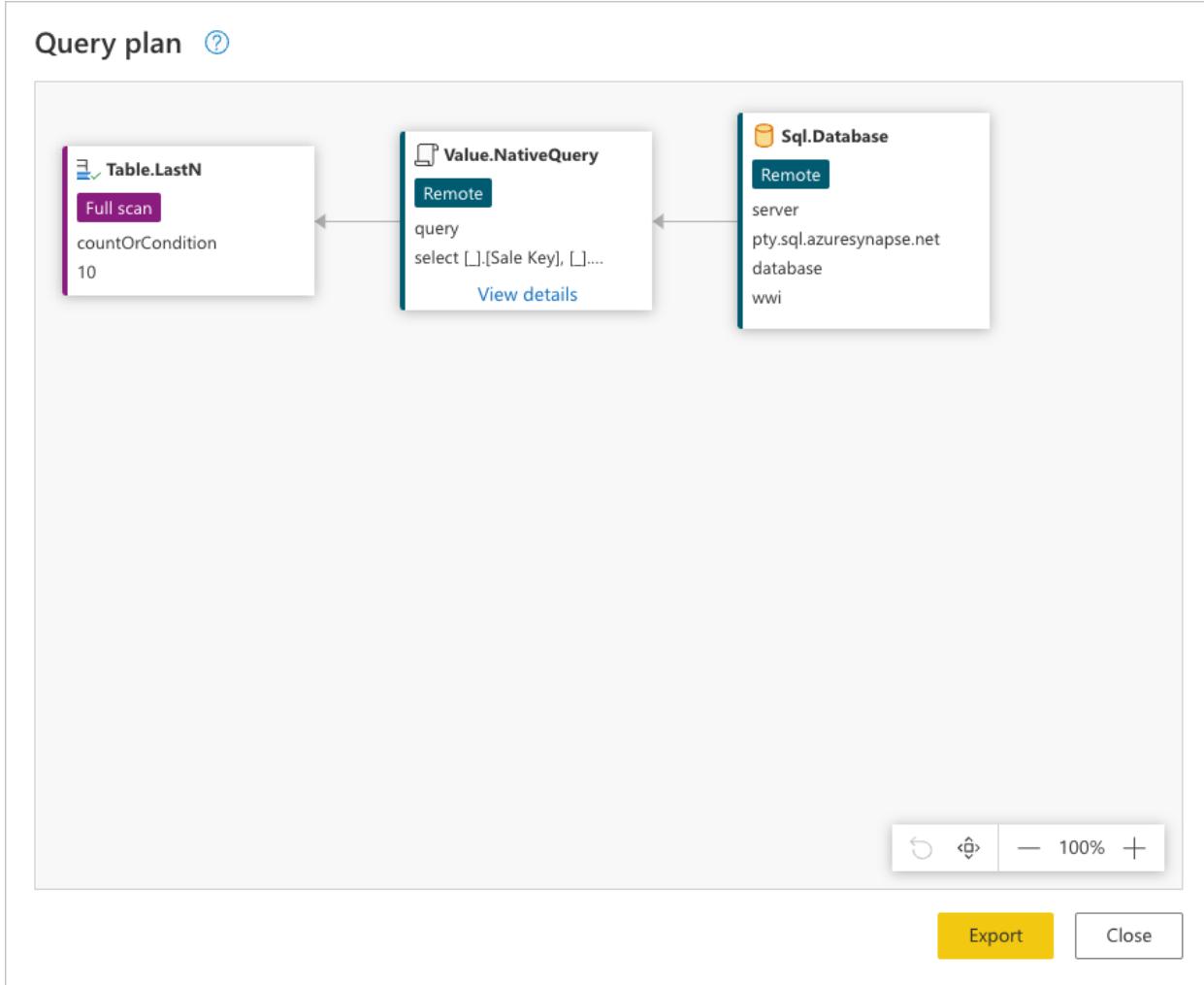
```

Partial query folding example: Understanding the query evaluation

Checking the applied steps pane, you notice that the step folding indicators are showing that the last transform that you added, `Kept bottom rows`, is marked as a step that will be evaluated outside the data source or, in other words, by the Power Query engine.

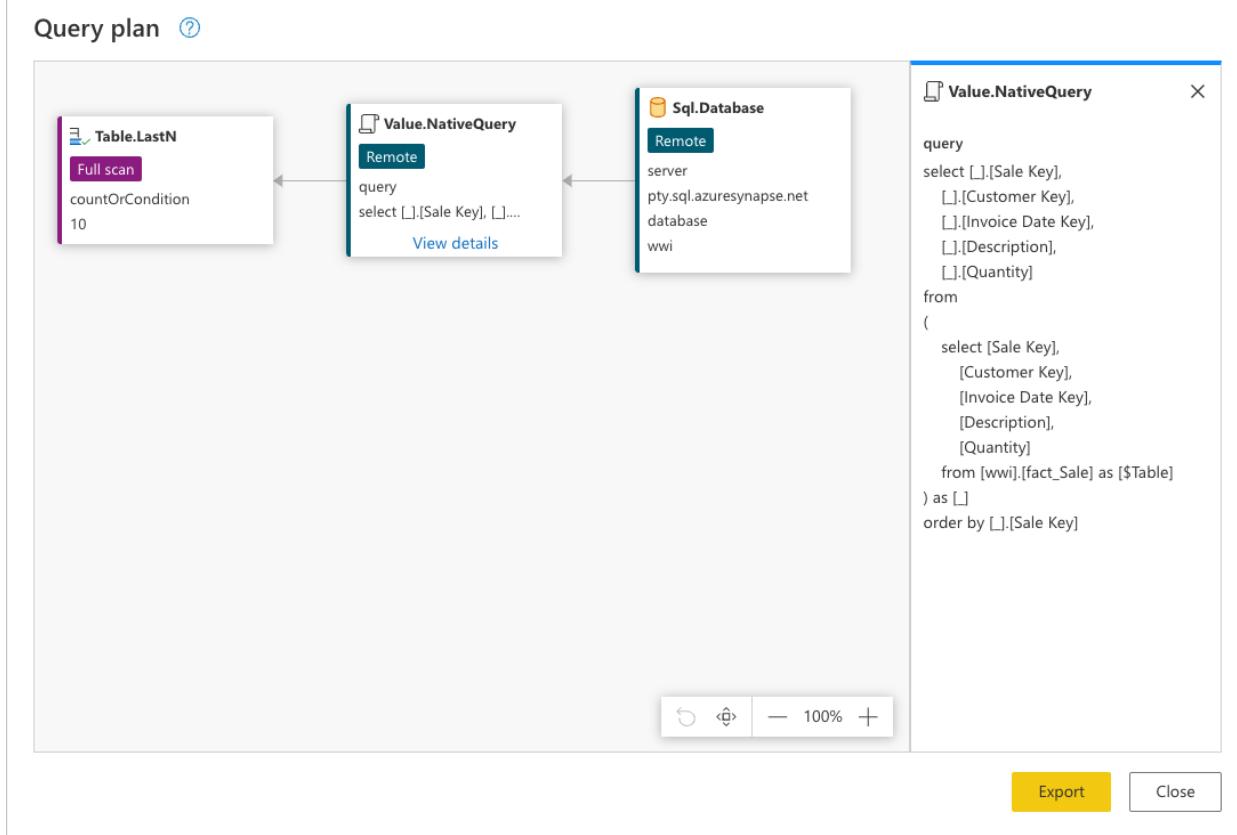


You can right-click the last step of your query, the one named `Kept bottom rows`, and select the **Query plan** option to better understand how your query might be evaluated.



Each box in the previous image is called a *node*. A node represents every process that needs to happen (from left to right) in order for your query to be evaluated. Some of these nodes can be evaluated at your data source while others, like the node for `Table.LastN`, represented by the **Kept bottom rows** step, are evaluated using the Power Query engine.

To see the exact request that is sent to your data source, select **View details** in the `Value.NativeQuery` node.



This request is in the native language of your data source. For this case, that language is SQL and this statement represents a request for all the rows, with only the requested fields from the `fact_Sale` table ordered by the `Sale Key` field.

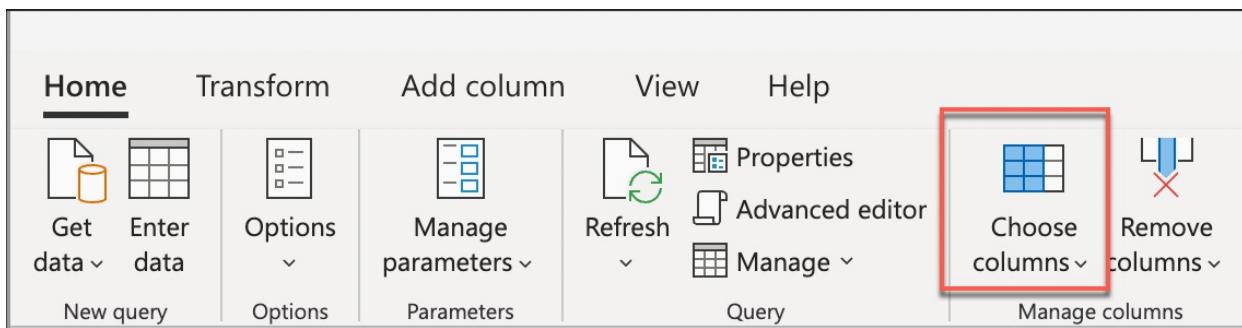
Consulting this data source request can help you better understand the story that the full query plan tries to convey. The order of the nodes is a sequential process that starts by requesting the data from your data source:

- `Sql.Database` : Connects to the database and sends metadata requests to understand its capabilities.
- `Value.NativeQuery` : Represents the request that was generated by Power Query to fulfill the query. Power Query submits the data requests in a native SQL statement to the data source. For this case, that represents all records, with only the requested fields from the `fact_Sale` table in the database sorted in ascending order by the `Sales Key` field.
- `Table.LastN` : Once Power Query receives all records from the `fact_Sale` table, it uses the Power Query engine to filter the table and keep only the last 10 rows.

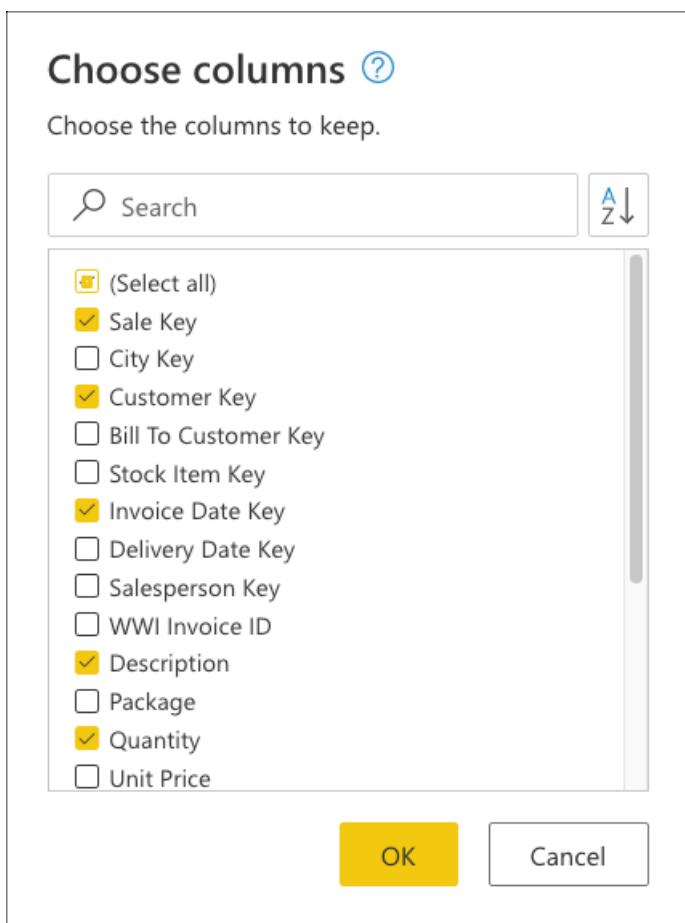
For its evaluation, this query had to download all rows and only the required fields from the `fact_Sale` table. It took an average of 3 minutes and 4 seconds to be processed in a standard instance of Power BI dataflows (which accounts for the evaluation and loading of data to dataflows).

Full query folding example

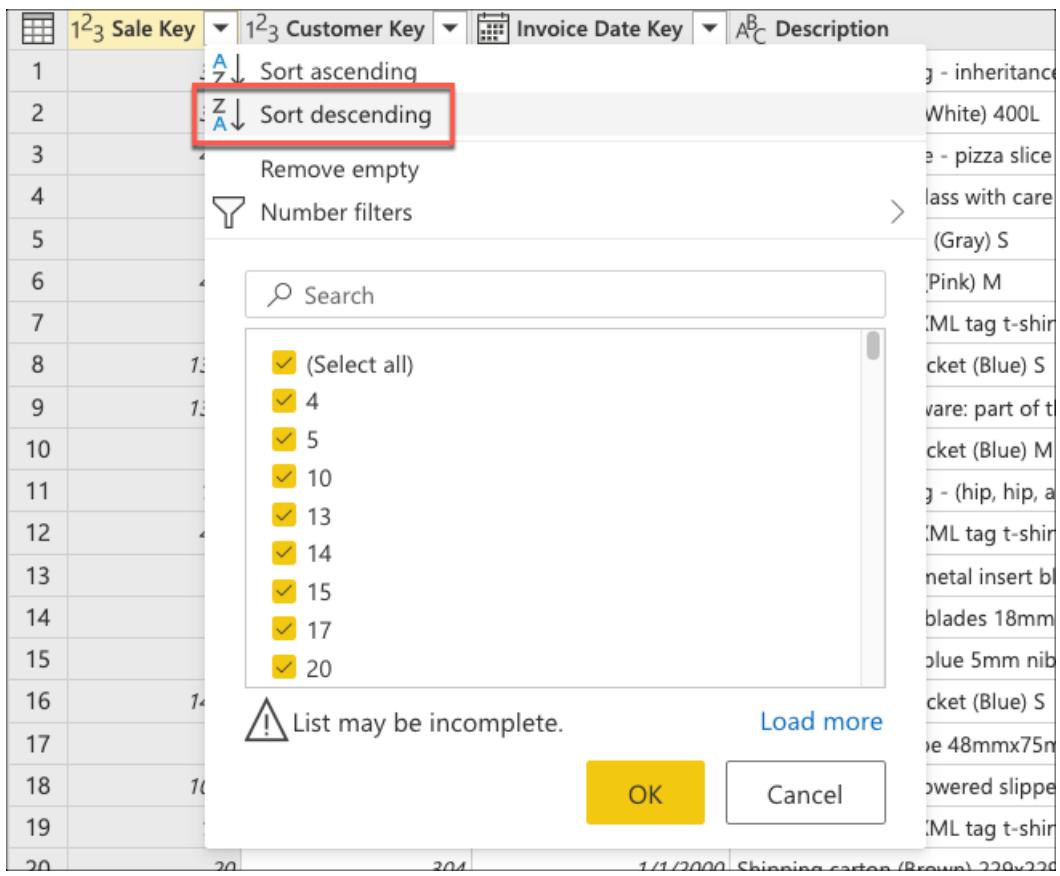
After connecting to the database and navigating to the `fact_Sale` table, start by selecting the columns that you want to keep from your table. Select the **Choose columns** transform found inside the **Manage columns** group from the **Home** tab. This transform helps you to explicitly select the columns that you want to keep from your table and remove the rest.



In Choose columns, select the `Sale Key`, `Customer Key`, `Invoice Date Key`, `Description`, and `Quantity` columns, and then select OK.



You now create logic that will sort the table to have the last sales at the top of the table. Select the `Sale Key` column, which is the primary key and incremental sequence or index of the table. Sort the table only using this field in descending order from the context menu for the column.



Next, select the table contextual menu and choose the **Keep top rows** transform.

	1 ² 3 Sale Key	1 ² 3 Customer Key	Invoice Date Key	Description
	Copy preview data	191	12/30/2000	Developer joke mu
	Use first row as headers	191	12/30/2000	Developer joke mu
	Add custom column	191	12/30/2000	Developer joke mu
	Add conditional column	191	12/30/2000	Developer joke mu
	Index column >	191	12/30/2000	Developer joke mu
	Choose columns	191	12/30/2000	Developer joke mu
	Keep top rows	191	12/30/2000	Developer joke mu
	Keep bottom rows	376	12/30/2000	"The Gu" red shirt X
	Keep range of rows	376	12/30/2000	"The Gu" red shirt X
	Keep duplicates	376	12/30/2000	"The Gu" red shirt X
	Keep errors	376	12/30/2000	"The Gu" red shirt X
	Remove top rows	376	12/30/2000	"The Gu" red shirt X
	Remove bottom rows	376	12/30/2000	"The Gu" red shirt X
	Remove alternate rows	376	12/30/2000	"The Gu" red shirt X
	Remove duplicates	191	12/30/2000	Developer joke mu
	Remove errors	191	12/30/2000	Developer joke mu
	Merge queries	191	12/30/2000	Developer joke mu
	Append queries	191	12/30/2000	Developer joke mu
22	3730261	191	12/30/2000	Developer joke mu

In **Keep top rows**, enter the value 10, and then select **OK**.

Keep top rows [?](#)

Specify how many rows to keep.

Number of rows



10

OK

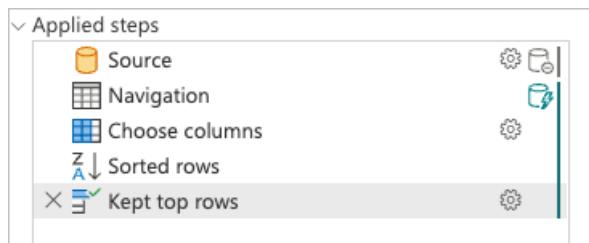
Cancel

The following code sample is the full M script for the query you created:

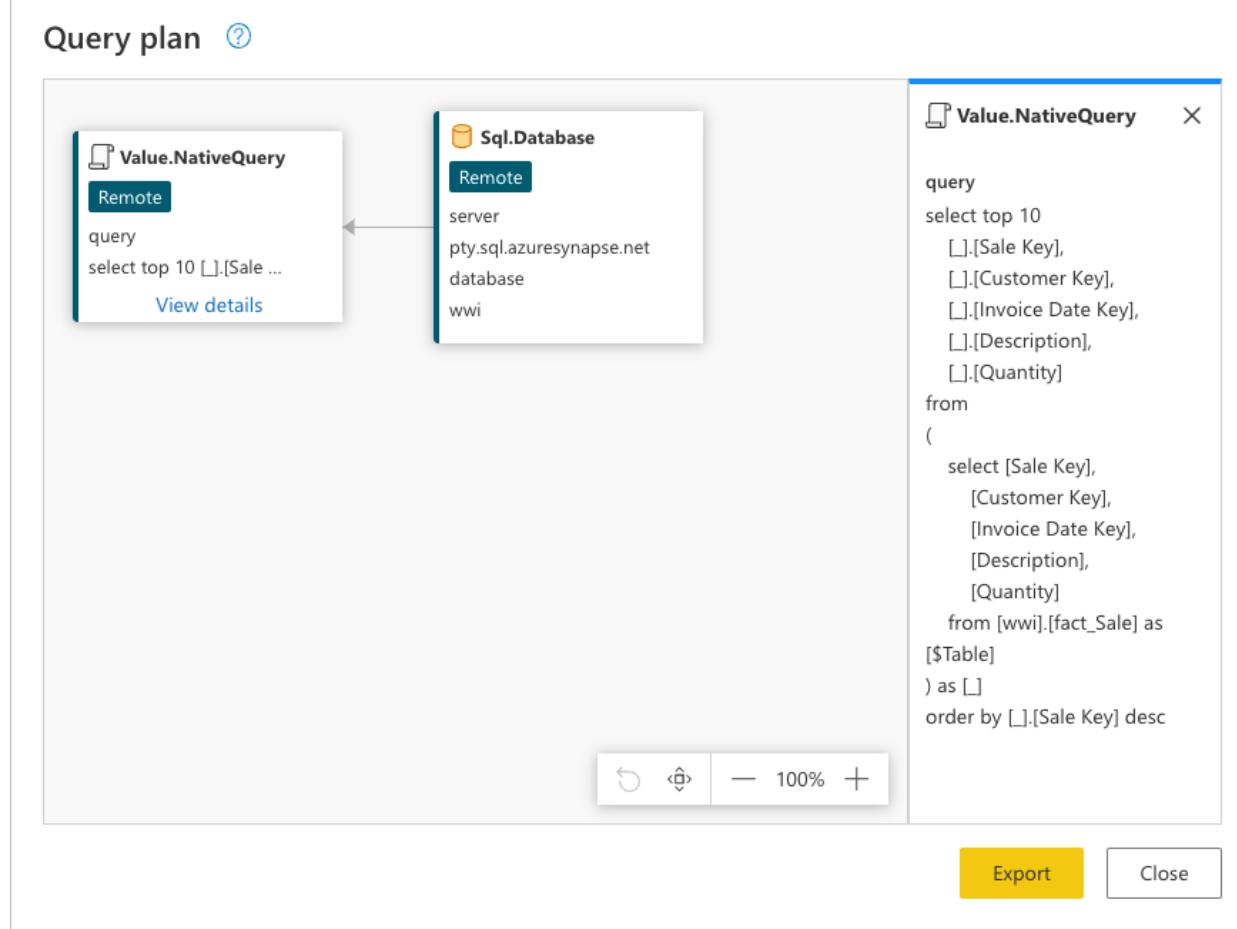
```
let
    Source = Sql.Database(ServerName, DatabaseName),
    Navigation = Source{[Schema = "wwi", Item = "fact_Sale"]}[Data],
    #"Choose columns" = Table.SelectColumns(Navigation, {"Sale Key", "Customer Key", "Invoice Date Key", "Description", "Quantity"}),
    #"Sorted rows" = Table.Sort(#"Choose columns", {{"Sale Key", Order.Descending}}),
    #"Kept top rows" = Table.FirstN(#"Sorted rows", 10)
in
    #"Kept top rows"
```

Full query folding example: Understanding the query evaluation

When checking the applied steps pane, you'll notice that the step folding indicators are showing that the transforms that you added, **Choose columns**, **Sorted rows**, and **Kept top rows**, are marked as steps that will be evaluated at the data source.



You can right-click the last step of your query, the one named **Kept top rows**, and select the option that reads **Query plan**.



This request is in the native language of your data source. For this case, that language is SQL and this statement represents a request for all the rows and fields from the `fact_Sale` table.

Consulting this data source query can help you better understand the story that the full query plan tries to convey:

- `Sql.Database` : Connects to the database and sends metadata requests to understand its capabilities.
- `Value.NativeQuery` : Represents the request that was generated by Power Query to fulfill the query. Power Query submits the data requests in a native SQL statement to the data source. For this case, that represents a request for only the top 10 records of the `fact_Sale` table, with only the required fields after being sorted in descending order using the `Sale Key` field.

NOTE

While there's no clause that can be used to SELECT the bottom rows of a table in the T-SQL language, there's a TOP clause that retrieves the top rows of a table.

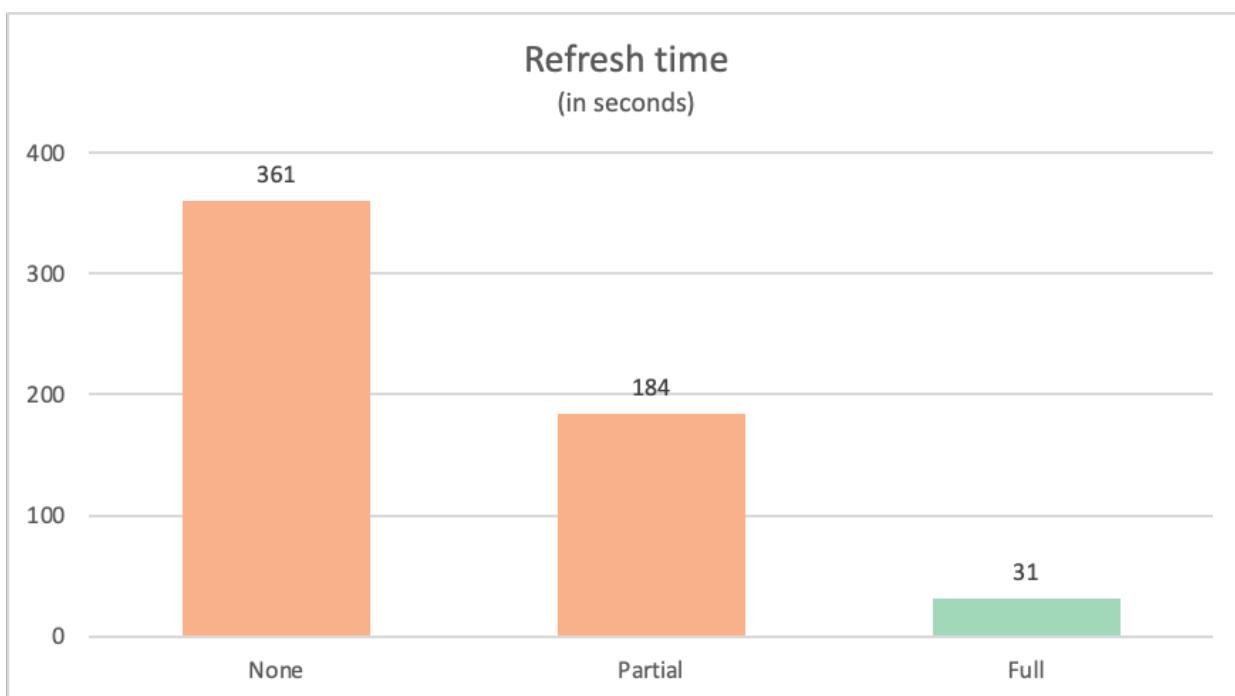
For its evaluation, this query only downloads 10 rows, with only the fields that you requested from the `fact_Sale` table. This query took an average of 31 seconds to be processed in a standard instance of Power BI dataflows (which accounts for the evaluation and loading of data to dataflows).

Performance comparison

To better understand the affect that query folding has in these queries, you can refresh your queries, record the time it takes to fully refresh each query, and compare them. For simplicity, this article provides the average refresh timings captured using the Power BI dataflows refresh mechanic while connecting to a dedicated Azure Synapse Analytics environment with DW2000c as the service level.

The refresh time for each query was as follows:

EXAMPLE	LABEL	TIME IN SECONDS
No query folding	None	361
Partial query folding	Partial	184
Full query folding	Full	31



It's often the case that a query that fully folds back to the data source outperforms similar queries that don't completely fold back to the data source. There could be many reasons why this is the case. These reasons range from the complexity of the transforms that your query performs, to the query optimizations implemented at your data source, such as indexes and dedicated computing, and network resources. Still, there are two specific key processes that query folding tries to use that minimizes the affect that both of these processes have with Power Query:

- Data in transit
- Transforms executed by the Power Query engine

The following sections explain the affect that these two processes have in the previously mentioned queries.

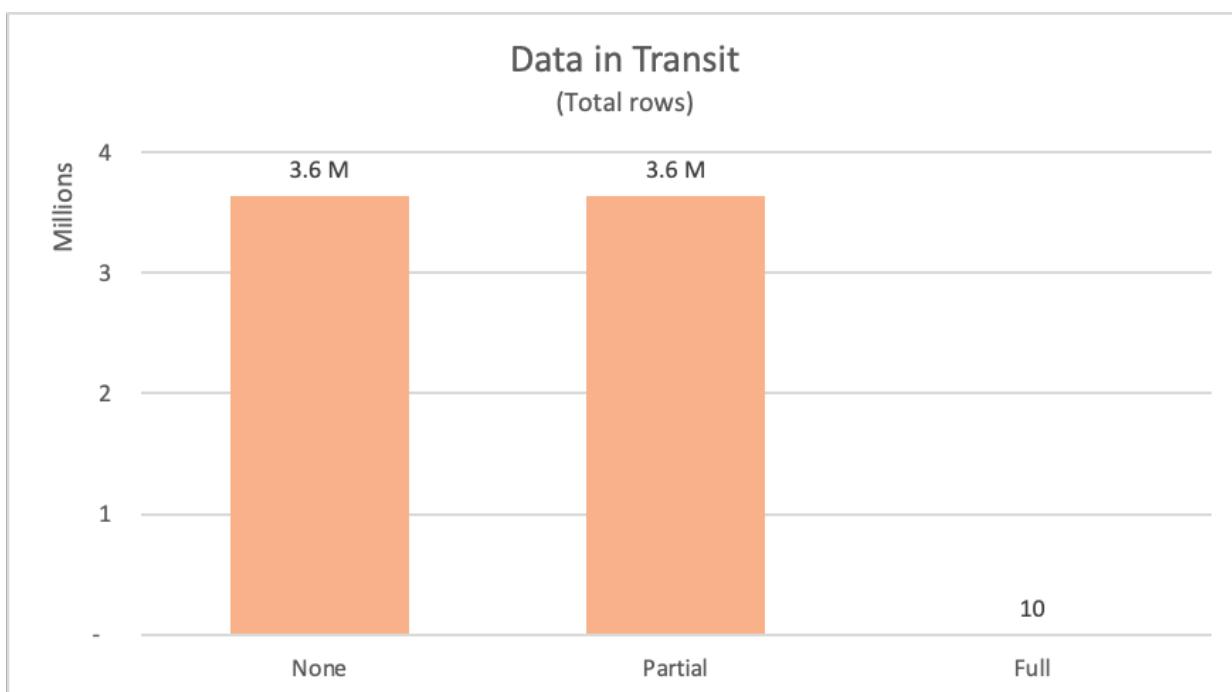
Data in transit

When a query gets executed, it tries to fetch the data from the data source as one of its first steps. What data is fetched from the data source is defined by the query folding mechanism. This mechanism identifies the steps from the query that can be offloaded to the data source.

The following table lists the number of rows requested from the `fact_Sale` table of the database. The table also includes a brief description of the SQL statement sent to request such data from the data source.

EXAMPLE	LABEL	ROWS REQUESTED	DESCRIPTION
No query folding	None	3644356	Request for all fields and all records from the <code>fact_Sale</code> table

EXAMPLE	LABEL	ROWS REQUESTED	DESCRIPTION
Partial query folding	Partial	3644356	Request for all records, but only required fields from the fact_Sale table after it was sorted by the Sale Key field
Full query folding	Full	10	Request for only the required fields and the TOP 10 records of the fact_Sale table after being sorted in descending order by the Sale Key field



When requesting data from a data source, the data source needs to compute the results for the request and then send the data to the requestor. While the computing resources have already been mentioned, the network resources of moving the data from the data source to Power Query, and then have Power Query be able to effectively receive the data and prepare it for the transforms that will happen locally can take some time depending on the size of the data.

For the showcased examples, Power Query had to request over 3.6 million rows from the data source for the no query folding and partial query folding examples. For the full query folding example, it only requested 10 rows. For the fields requested, the no query folding example requested all the available fields from the table. Both the partial query folding and the full query folding examples only submitted a request for exactly the fields that they needed.

Caution

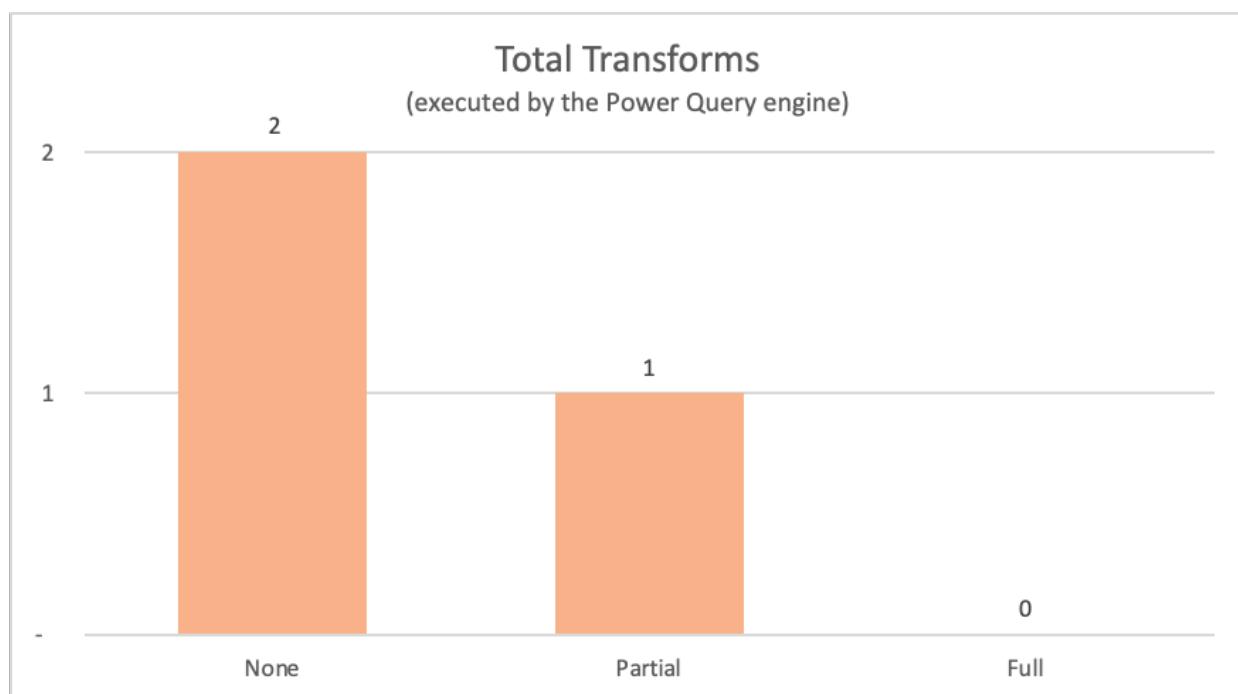
We recommend that you implement incremental refresh solutions that leverage query folding for queries or entities with large amounts of data. Different product integrations of Power Query implement timeouts to terminate long running queries. Some data sources also implement timeouts on long running sessions, trying to execute expensive queries against their servers. More information: [Using incremental refresh with dataflows](#) and [Incremental refresh for datasets](#)

Transforms executed by the Power Query engine

This article showcased how you can use the [Query plan](#) to better understand how your query might be evaluated. Inside the query plan, you can see the exact nodes of the transform operations that will be performed by the Power Query engine.

The following table showcases the nodes from the query plans of the previous queries that would have been evaluated by the Power Query engine.

EXAMPLE	LABEL	POWER QUERY ENGINE TRANSFORM NODES
No query folding	None	<code>Table.LastN</code> , <code>Table.SelectColumns</code>
Partial query folding	Partial	<code>Table.LastN</code>
Full query folding	Full	—



For the examples showcased in this article, the full query folding example doesn't require any transforms to happen inside the Power Query engine as the required output table comes directly from the data source. In contrast, the other two queries required some computation to happen at the Power Query engine. Because of the amount of data that needs to be processed by these two queries, the process for these examples takes more time than the full query folding example.

Transforms can be grouped into the following categories:

TYPE OF OPERATOR	DESCRIPTION
Remote	Operators that are data source nodes. The evaluation of these operators occurs outside of Power Query.

TYPE OF OPERATOR	DESCRIPTION
Streaming	Operators are pass-through operators. For example, <code>Table.SelectRows</code> with a simple filter can usually filter the results as they pass through the operator, and won't need to gather all rows before moving the data. <code>Table.SelectColumns</code> and <code>Table.ReorderColumns</code> are other examples of these sort of operators.
Full scan	Operators that need to gather all the rows before the data can move on to the next operator in the chain. For example, to sort data, Power Query needs to gather all the data. Other examples of full scan operators are <code>Table.Group</code> , <code>Table.NestedJoin</code> , and <code>Table.Pivot</code> .

TIP

While not every transform is the same from a performance standpoint, in most cases, having fewer transforms is usually better.

Considerations and suggestions

- Follow the best practices when creating a new query, as stated in [Best practices in Power Query](#).
- Use the step folding indicators to check which steps are preventing your query from folding. Reorder them if necessary to increase folding.
- Use the query plan to determine which transforms are happening at the Power Query engine for a particular step. Consider modifying your existing query by re-arranging your steps. Then check the query plan of the last step of your query again and see if the query plan looks better than the previous one. For example, the new query plan has less nodes than the previous one, and most of the nodes are "Streaming" nodes and not "full scan". For data sources that support folding, any nodes in the query plan other than `Value.NativeQuery` and data source access nodes represent transforms that didn't fold.
- When available, you can use the **View Native Query** (or **View data source query**) option to ensure that your query can be folded back to the data source. If this option is disabled for your step, and you're using a source that normally enables it, you've created a step that stops query folding. If you're using a source that doesn't support this option, you can rely on the step folding indicators and query plan.
- Use the query diagnostics tools to better understand the requests being sent to your data source when query folding capabilities are available for the connector.
- When combining data sourced from the use of multiple connectors, Power Query tries to push as much work as possible to both of the data sources while complying with the privacy levels defined for each data source.
- Read the article on [privacy levels](#) to protect your queries from running against a Data Privacy Firewall error.
- Use other tools to check query folding from the perspective of the request being received by the data source. Based on the example in this article, you can use the Microsoft SQL Server Profiler to check the requests being sent by Power Query and received by the Microsoft SQL Server.
- If you add a new step to a fully folded query and the new step also folds, Power Query might send a new request to the data source instead of using a cached version of the previous result. In practice, this process can result in seemingly simple operations on a small amount of data taking longer to refresh in the preview than expected. This longer refresh is due to Power Query requerying the data source rather than working off a local copy of the data.

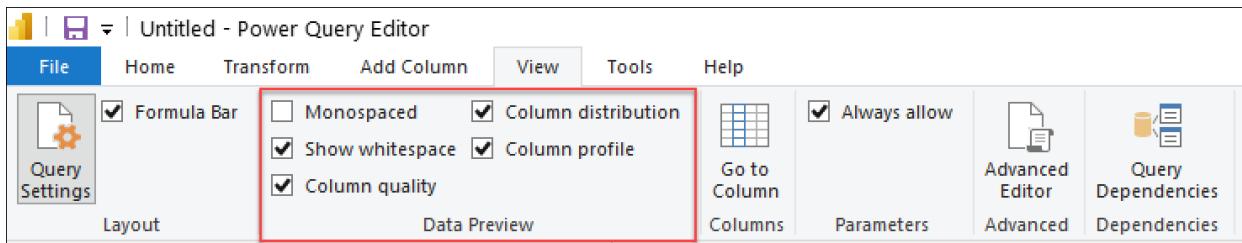
Using the data profiling tools

1/15/2022 • 2 minutes to read • [Edit Online](#)

The data profiling tools provide new and intuitive ways to clean, transform, and understand data in Power Query Editor. They include:

- Column quality
- Column distribution
- Column profile

To enable the data profiling tools, go to the **View** tab on the ribbon. Enable the options you want in the **Data preview** group, as shown in the following image.



After you enable the options, you'll see something like the following image in Power Query Editor.

Column statistics for the previewed data:

Statistic	Value
Count	701
Error	0
Empty	0
Distinct	5
Unique	0
Nan	0
Zero	0
Min	300000
Max	3000000
Average	1584736...

NOTE

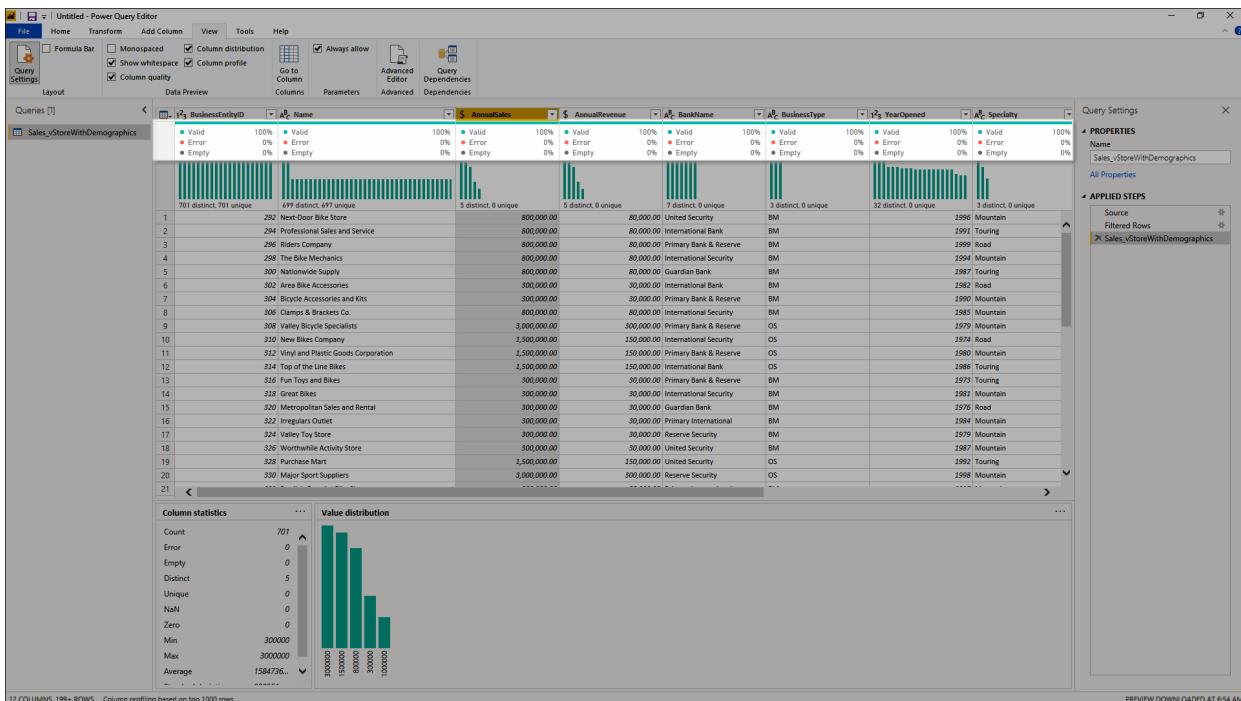
By default, Power Query will perform this data profiling over the first 1,000 rows of your data. To have it operate over the entire dataset, check the lower-left corner of your editor window to change how column profiling is performed.

Column quality

The column quality feature labels values in rows in five categories:

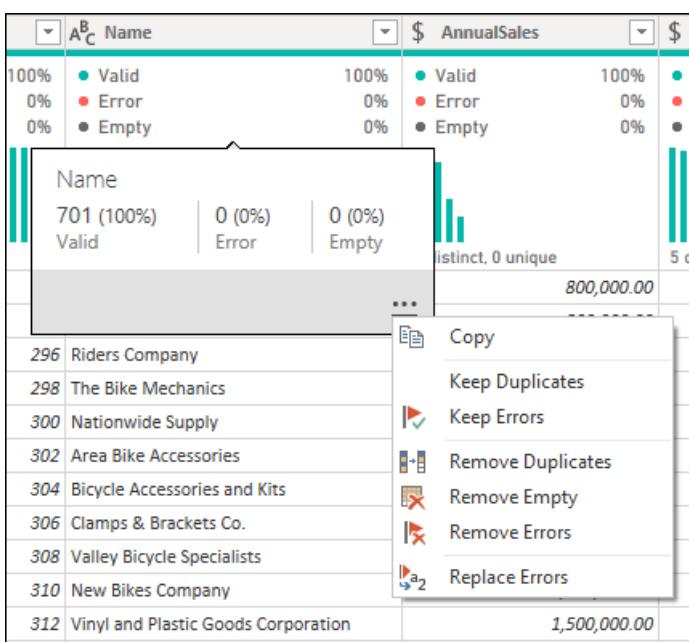
- **Valid**, shown in green.
- **Error**, shown in red.
- **Empty**, shown in dark grey.
- **Unknown**, shown in dashed green. Indicates when there are errors in a column, the quality of the remaining data is unknown.
- **Unexpected error**, shown in dashed red.

These indicators are displayed directly underneath the name of the column as part of a small bar chart, as shown in the following image.



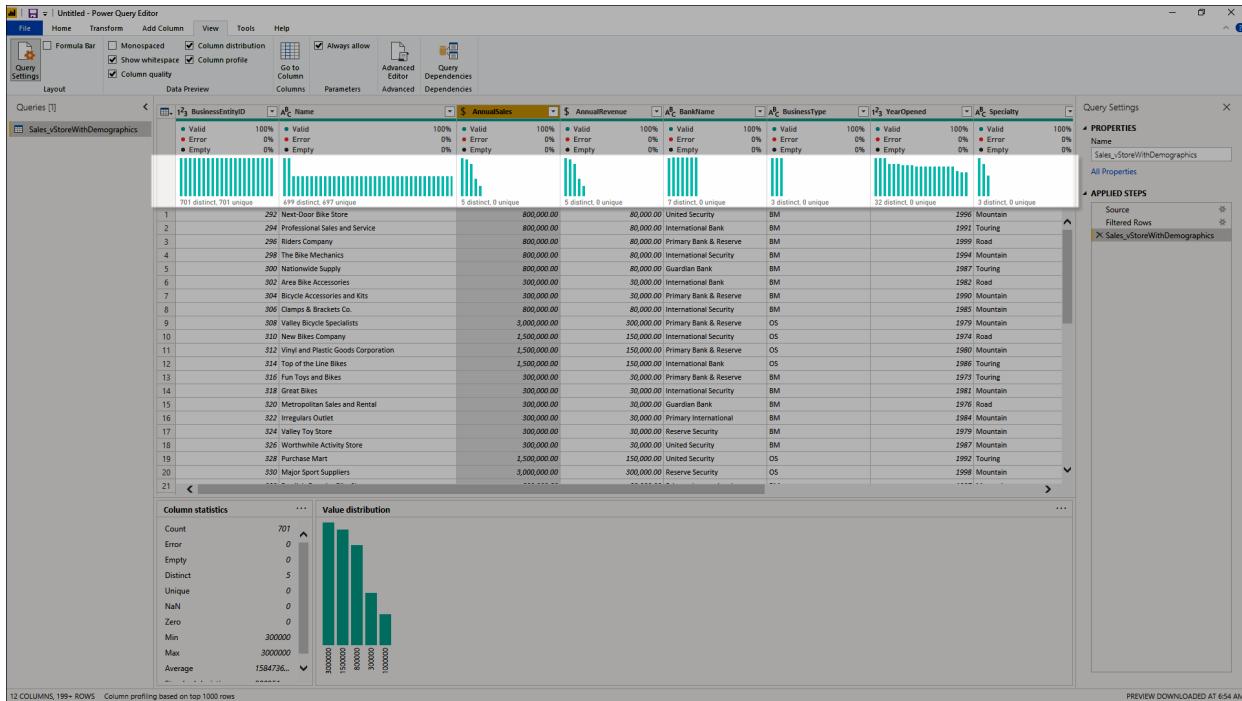
The number of records in each column quality category is also displayed as a percentage.

By hovering over any of the columns, you are presented with the numerical distribution of the quality of values throughout the column. Additionally, selecting the ellipsis button (...) opens some quick action buttons for operations on the values.

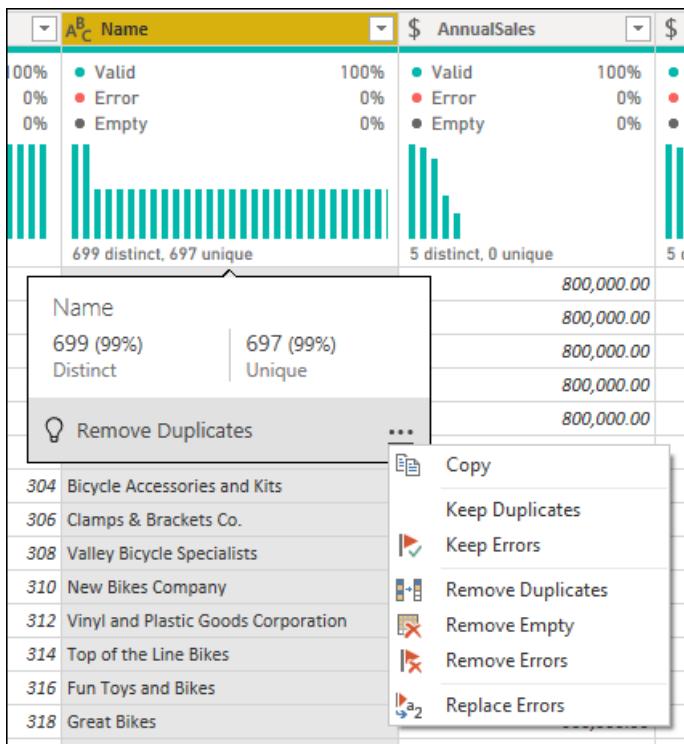


Column distribution

This feature provides a set of visuals underneath the names of the columns that showcase the frequency and distribution of the values in each of the columns. The data in these visualizations is sorted in descending order from the value with the highest frequency.

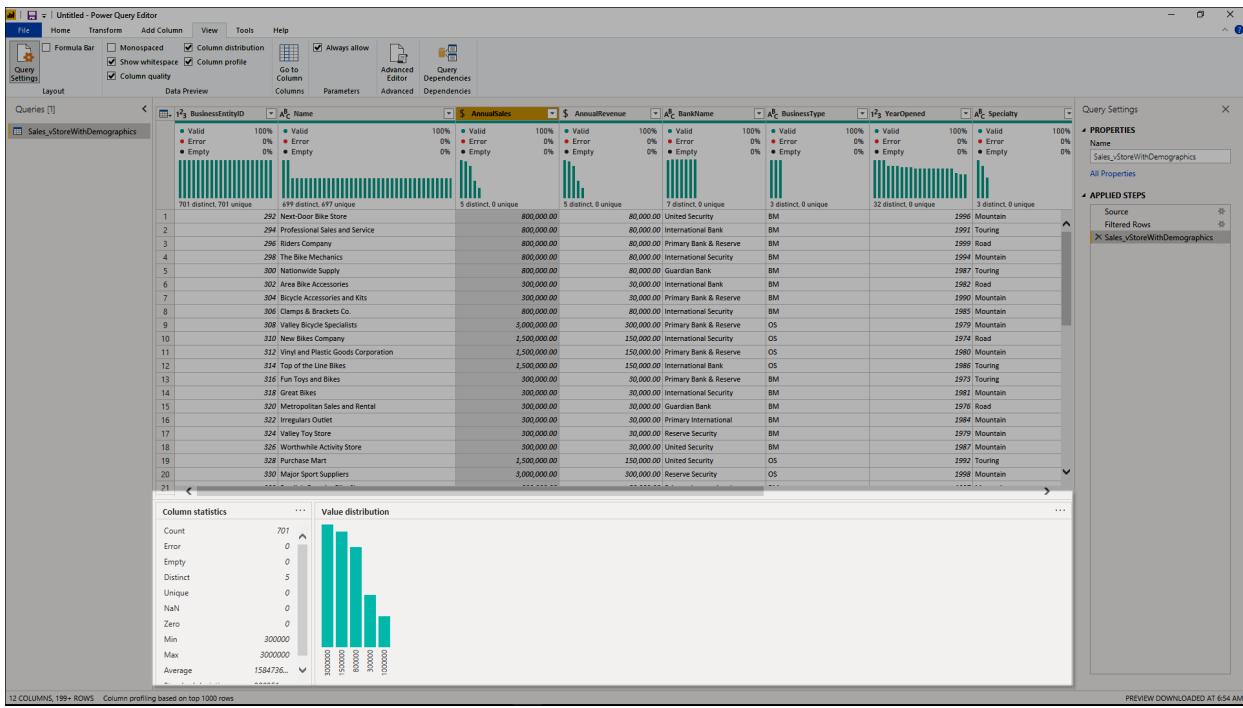


By hovering over the distribution data in any of the columns, you get information about the overall data in the column (with distinct count and unique values). You can also select the ellipsis button and choose from a menu of available operations.



Column profile

This feature provides a more in-depth look at the data in a column. Apart from the column distribution chart, it contains a column statistics chart. This information is displayed underneath the data preview section, as shown in the following image.

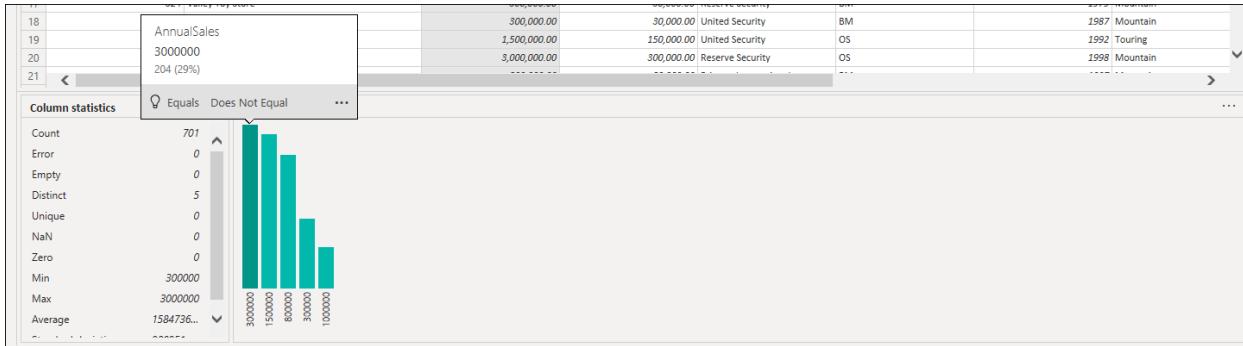


12 COLUMNS, 199+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 6:54 AM

Filter by value

You can interact with the value distribution chart on the right side and select any of the bars by hovering over the parts of the chart.

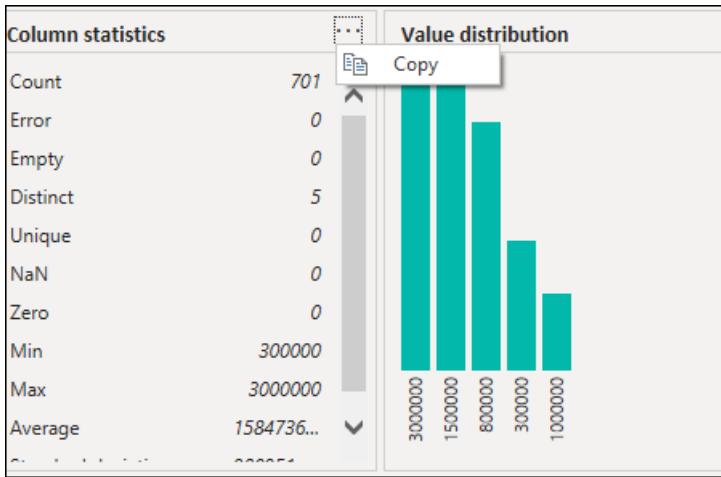


Right-click to display a set of available transformations for that value.



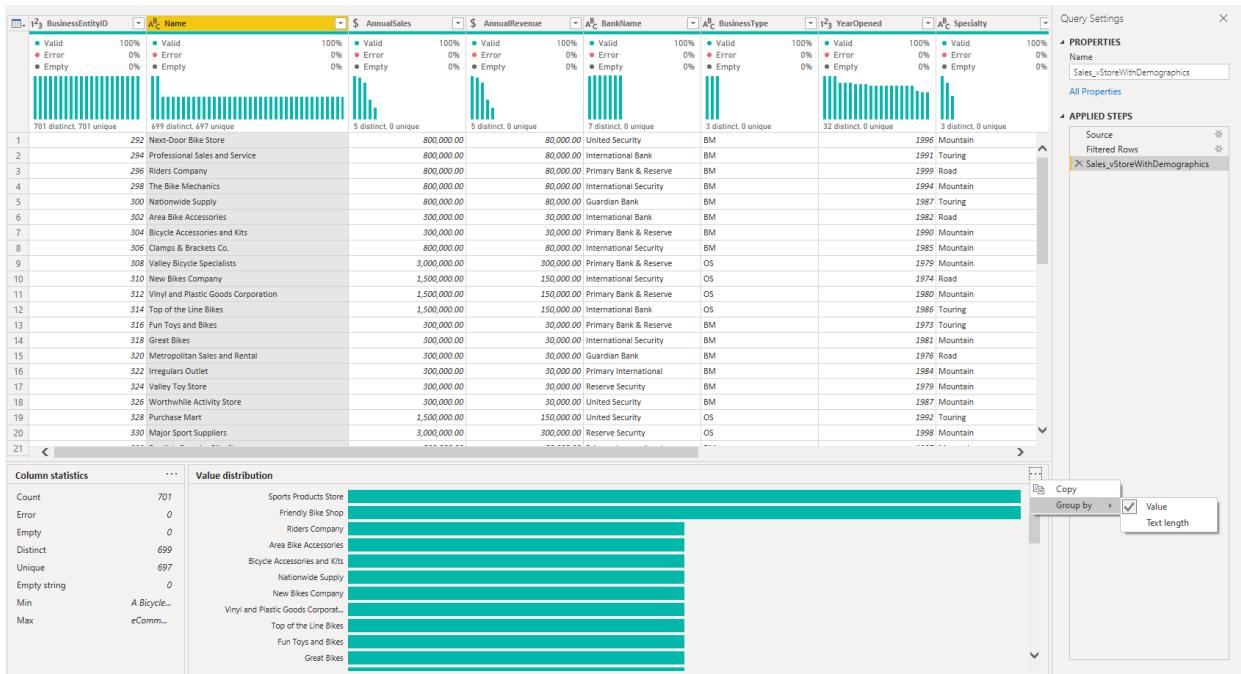
Copy data

In the upper-right corner of both the column statistics and value distribution sections, you can select the ellipsis button (...) to display a **Copy** shortcut menu. Select it to copy the data displayed in either section to the clipboard.

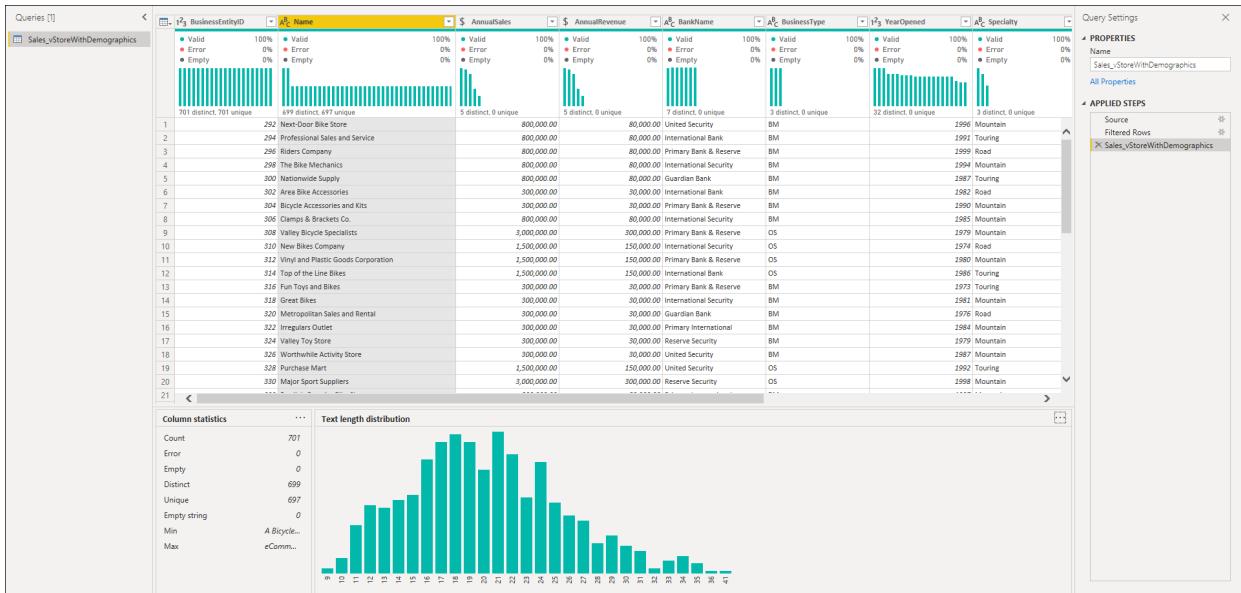


Group by value

When you select the ellipsis button (...) in the upper-right corner of the value distribution chart, in addition to **Copy** you can select **Group by**. This feature groups the values in your chart by a set of available options.



The image below shows a column of product names that have been grouped by text length. After the values have been grouped in the chart, you can interact with individual values in the chart as described in [Filter by value](#).

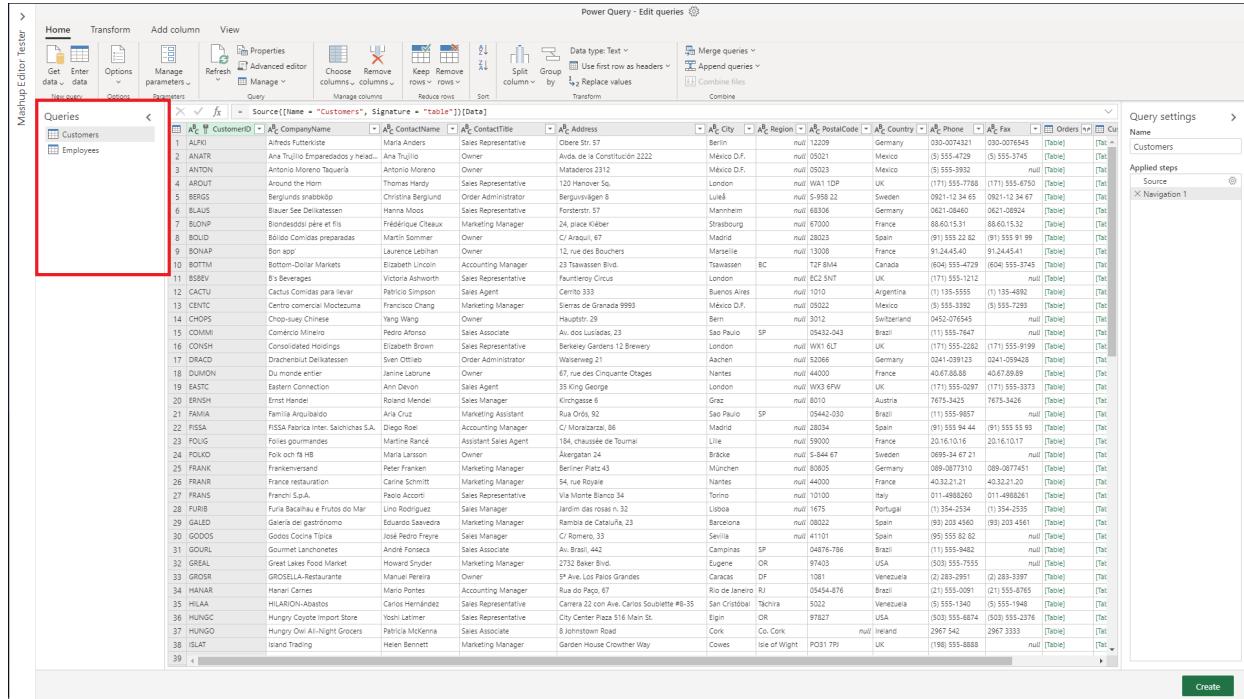


Using the Queries pane

1/15/2022 • 3 minutes to read • [Edit Online](#)

In Power Query, you'll be creating many different queries. Whether it be from getting data from many tables or from duplicating the original query, the number of queries will increase.

You'll be using the **Queries** pane to navigate through the queries.



The screenshot shows the Power Query - Edit queries interface. The top ribbon has tabs for Home, Transform, Add column, View, Get data, Options, Manage parameters, Refresh, Advanced editor, Properties, Choose columns, Remove columns, Split column by, Data type: Text, Use first row as headers, Append queries, Merge queries, Keep rows, Remove rows, Split group by, Replace values, Transform, and Combine files. The main area is titled "Power Query - Edit queries". On the left, there's a "Queries" pane with a tree view showing "Customers" and "Employees". A red box highlights the "Customers" node. To the right is a large table with 39 rows of data, and on the far right, there's a "Query settings" pane with sections for Name, Applied steps, Source, and Navigation 1. At the bottom right is a "Create" button.

Navigating with the Queries pane

The most basic usage of the **Queries** pane is to navigate to all of the queries. The navigation is similar to a file explorer. To switch between the queries, just select the query you want to go to.

NOTE

Some actions in the Power Query Online editor may be different than actions in the Power Query Desktop editor. These differences will be noted in this article.

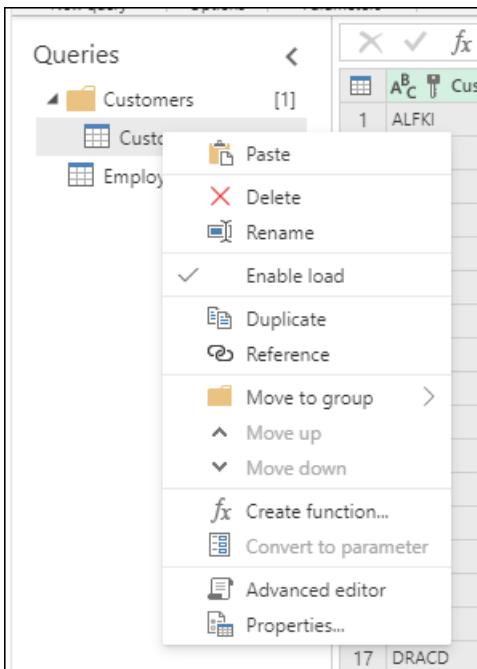
Basic actions in the Query pane

Similar to features throughout the ribbon and the editor, the context menu of a query lets you make transformations directly onto the query.

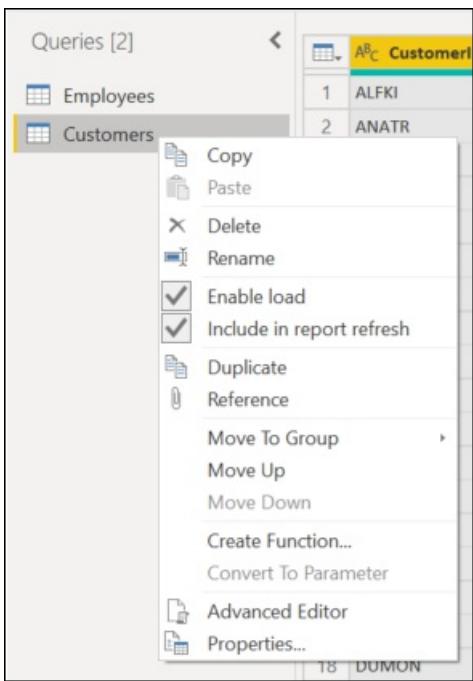
To reach these actions, open the context menu (the right-click menu) in the **Query** pane.

Differences between online and desktop:

- Power Query Online



- Power Query Desktop



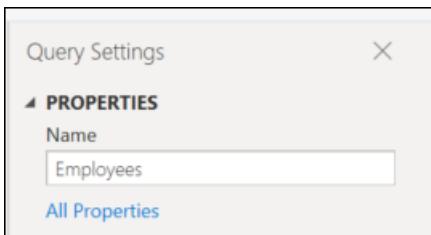
To be more comprehensive, we'll be touching on all of the context menu actions that are relevant for either.

Rename a query

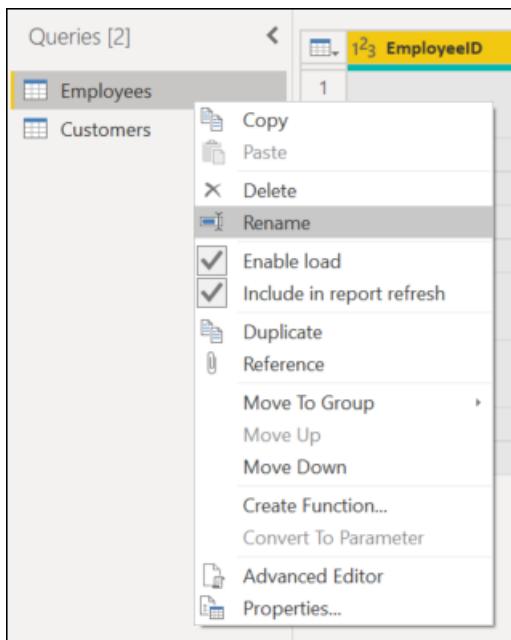
To directly change the name of the query, double-select on the name of the query. This action will allow you to immediately change the name.

Other options to rename the query are:

- Go to the context menu and select **Rename**.

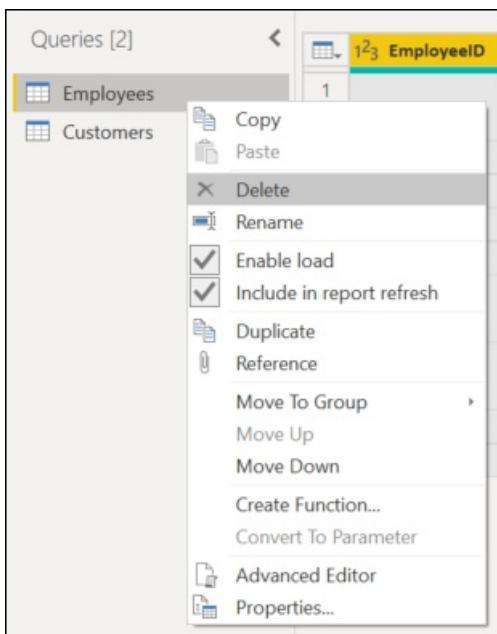


- Go to **Query Settings** and enter in a different name in the **Name** input field.



Delete a query

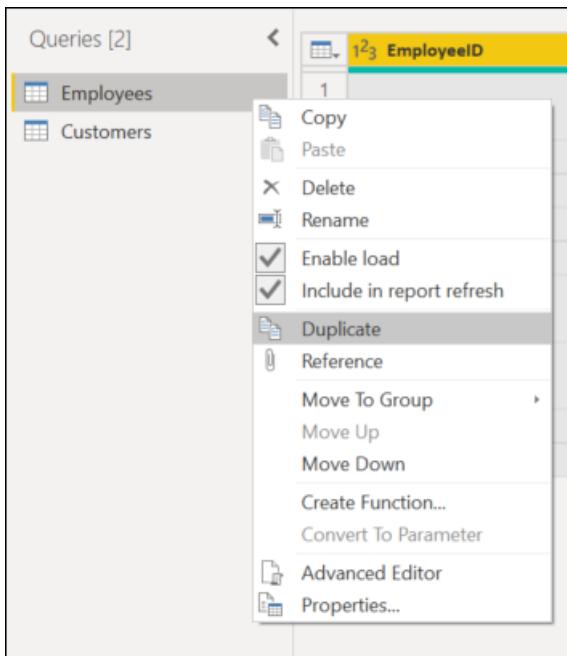
To delete a query, open the context pane on the query and select **Delete**. There will be an additional pop-up confirming the deletion. To complete the deletion, select the **Delete** button.



Duplicating a query

Duplicating a query will create a copy of the query you're selecting.

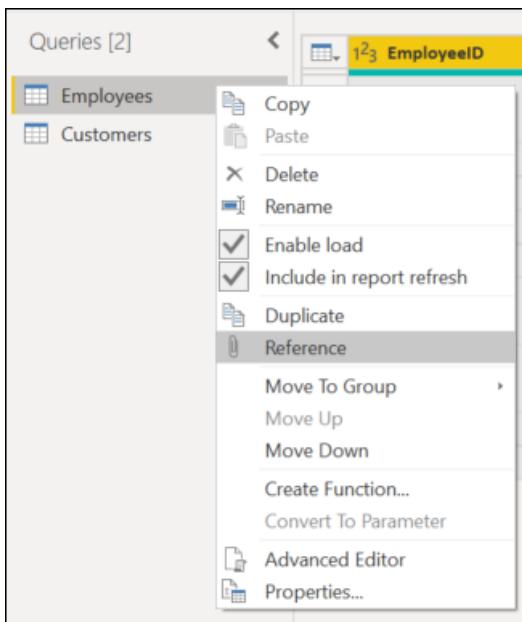
To duplicate your query, open the context pane on the query and select **Duplicate**. A new duplicate query will pop up on the side of the query pane.



Referencing a query

Referencing a query will create a new query. The new query uses the steps of a previous query without having to duplicate the query. Additionally, any changes on the original query will transfer down to the referenced query.

To reference your query, open the context pane on the query and select **Reference**. A new referenced query will pop up on the side of the query pane.



Copy and paste

Copy and paste can be used when you have a copied query to place in the Power Query editor.

NOTE

To learn more about how to copy and paste queries in Power Query, see [Sharing a query](#).

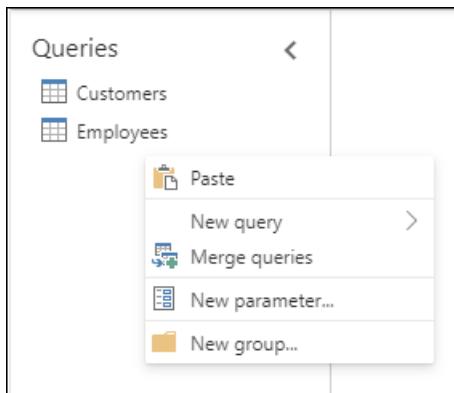
Context pane options in the Queries pane

There are some additional context pane options in the **Queries** pane that you can use. These options are **New query**, **Merge queries**, **New parameter**, and **New group**.

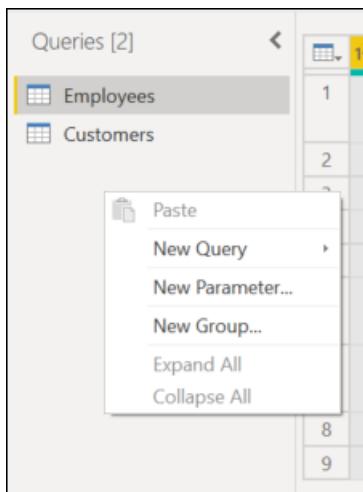
To reach these options, open the context menu (the right-click menu) in the Queries pane.

Differences between online and desktop:

- Power Query Online



- Power Query Desktop

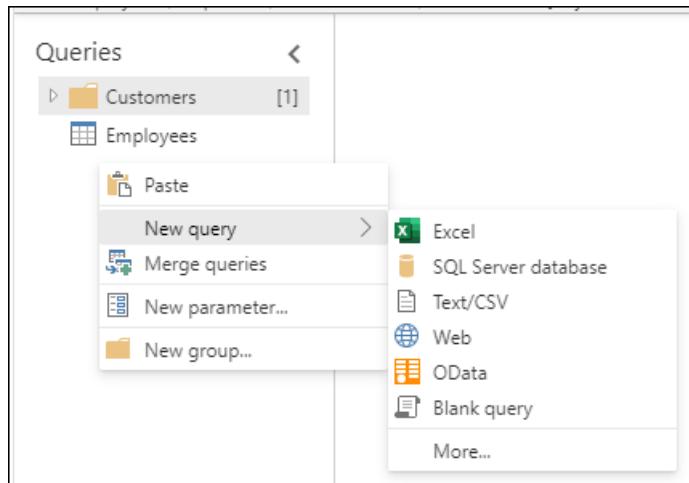


For the sake of being more comprehensive, we'll once again describe all of the context menu actions that are relevant for either.

New query

You can import data into the Power Query editor as an option from the context menu.

This option functions the same as the **Get Data** feature.



NOTE

To learn about how to get data into Power Query, see [Getting data](#)

Merge queries

When you select the **Merge queries** option from the context menu, the **Merge queries** input screen opens.

This option functions the same as the **Merge queries** feature located on the ribbon and in other areas of the editor.

NOTE

To learn more about how to use the **Merge queries** feature, see [Merge queries overview](#).

New parameter

When you select the **New parameter** option from the context menu, the **New parameter** input screen opens.

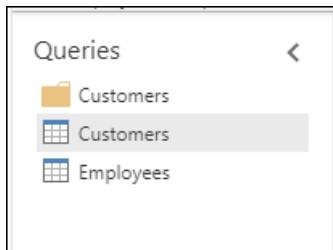
This option functions the same as the **New parameter** feature located on the ribbon.

NOTE

To learn more about **Parameters** in Power Query, see [Using parameters](#).

New group

You can make folders and move the queries into and out of the folders for organizational purposes. These folders are called *groups*.



To move the query into a group, open the context menu on the specific query.

In the menu, select **Move to group**.

Then, select the group you want to put the query in.

The screenshot shows the Power Query Editor interface. On the left, there's a tree view labeled 'Queries' with three items: 'Customers' (represented by a folder icon), 'Customers' (represented by a grid icon), and 'Employees'. The second 'Customers' item is selected and highlighted in grey. A context menu is open over this selected item, listing options: 'Paste', 'Delete', 'Rename', 'Enable load', 'Duplicate', 'Reference', 'Move to group', 'Move up', 'Move down', 'Create function...', 'Convert to parameter', 'Advanced editor', and 'Properties...'. The 'Move to group' option is currently selected. A dropdown menu under 'Move to group' shows a list of existing groups: 'Customers', 'Queries (root)', and 'New group...'. To the right of the menu, the Power Query table view displays data from the 'Customers' table, including columns 'CustomerID' and 'CompanyName', with rows for various companies like 'Alfreds Futterkiste' and 'Ana Trujillo Emparedados'. At the bottom of the table view, there's a status bar with the text '1 / 10400'.

The move will look like the following image. Using the same steps as above, you can also move the query out of the group by selecting **Queries (root)** or another group.

This screenshot shows the Power Query Editor after the move. The tree view on the left now shows a single 'Customers' folder icon containing one item, which is the previously selected 'Customers' query. The 'Employees' query remains as a separate item below it. The rest of the interface, including the table view and status bar, appears identical to the previous screenshot.

In desktop versions of Power Query, you can also drag and drop the queries into the folders.

Diagram view

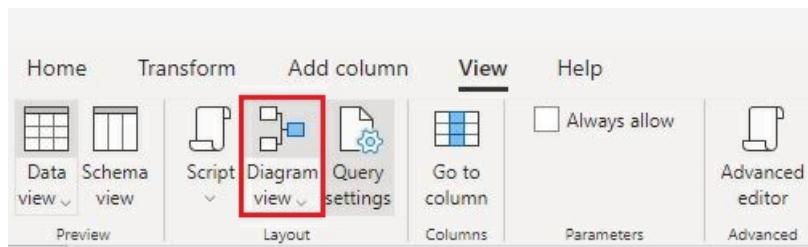
1/15/2022 • 11 minutes to read • [Edit Online](#)

Diagram view offers a visual way to prepare data in the Power Query editor. With this interface, you can easily create queries and visualize the data preparation process. Diagram view simplifies the experience of getting started with data wrangling. It speeds up the data preparation process and helps you quickly understand the dataflow, both the "big picture view" of how queries are related and the "detailed view" of the specific data preparation steps in a query.

This article provides an overview of the capabilities provided by diagram view.

The screenshot shows the Power Query - Edit queries window in Diagram view. The ribbon tabs are Home, Transform, Add column, and View. The View tab is selected. The main area displays a flowchart of data transformation steps. The Customers query is connected to the Orders query, which is then connected to the Top Customers by Orders step. The Products query is shown separately with a note '3 steps'. Below the flowchart is a detailed table of data with columns for CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, and Phone. The table shows 13 rows of data from various US companies like Great Lakes Food Market and Rattlersnake Canyon Grocery.

This feature is enabled by selecting **Diagram view** in the **View** tab on the ribbon. With diagram view enabled, the steps pane and queries pane will be collapsed.



NOTE

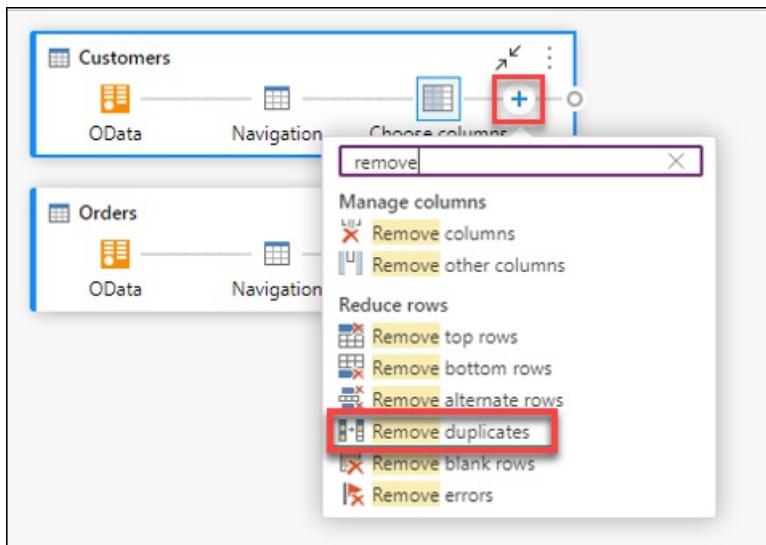
Currently, diagram view is only available in Power Query Online.

Authoring queries using diagram view

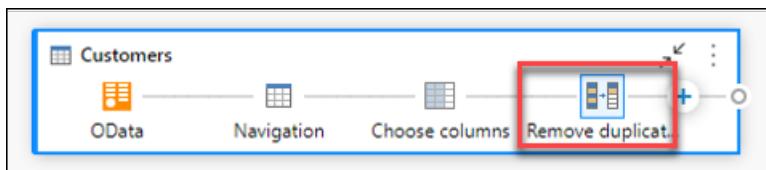
Diagram view provides you with a visual interface to create, view, or modify your queries. In diagram view, you can connect to many different types of data sources using the "[Get Data](#)" experience.

Diagram view is also connected to the Data Preview and the ribbon so that you can select columns in the Data Preview.

You can add a new step within a query, after the currently selected step, by selecting the + button, and then either search for the transform or choose the item from the shortcut menu. These are the same transforms you'll find in the Power Query editor ribbon.



By searching and selecting the transform from the shortcut menu, the step gets added to the query, as shown in the following image.

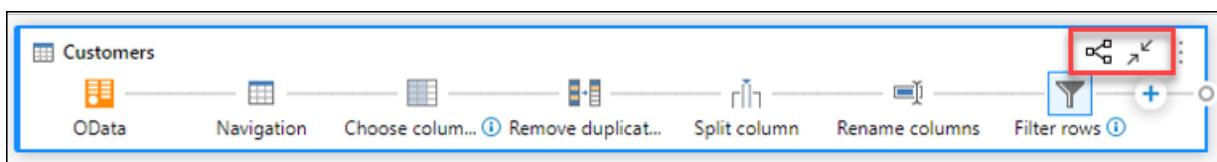


NOTE

To learn more about how to author queries in the Query editor using the Power Query editor ribbon or data preview, go to [Power Query Quickstart](#).

Query level actions

You can perform two quick actions on a query—*expand/collapse* a query and *highlight related queries*. These quick actions show up on an active selected query or when hovering over a query.



You can perform more query level actions such as duplicate, reference, and so on, by selecting the query level context menu (the three vertical dots). You can also right-click in the query and get to the same context menu.

The screenshot shows the Power BI Data Editor interface. At the top, there's a toolbar with buttons for OData, Navigation, Remove duplicates, Choose columns, Expand, and Group by. Below the toolbar is a dropdown menu with options like Collapse, Highlight related queries, Delete, Rename, Enable load, Duplicate, Reference, Move to group, Create function..., Convert to parameter, Advanced editor, Properties..., Append queries, Append queries as new, Merge queries, and Merge queries as new. The 'Orders' query is expanded, showing a table with columns: CustomerID, CompanyName, FirstName, LastName, and ContactTitle. The table contains 13 rows of data from the Northwind database.

Expand or collapse query

To expand or collapse a query, right-click in the query and select **Expand/Collapse** from the query's context menu. You can also double-click in the query to expand or collapse a query.

This screenshot shows the Power BI Data Editor with the 'Employees' query expanded. The context menu is open, and the 'Collapse' option is highlighted with a red box. The 'Employees' query is connected to the 'Order_Details' query via a relationship. The 'Order_Details' query is collapsed, indicated by a blue button labeled '4 steps'. Below the collapsed query, its definition is visible: `Table.Group(#"Expanded Order_Details", {"CustomerID"}, {{...}})`. The main table below shows the count of orders per customer ID.

Highlight related queries

To view all the related queries for a given query, right-click in a query and select **Highlight related queries**. You can also select the highlight related queries button on the top-right of a query.

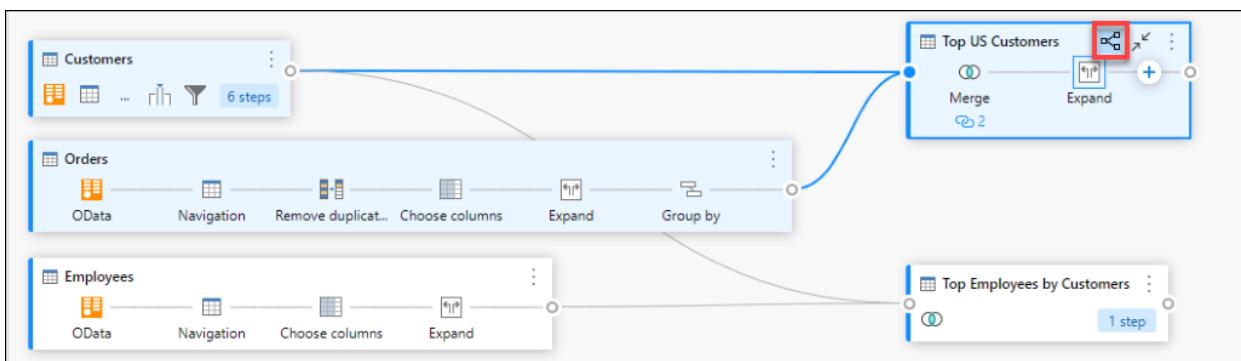
The screenshot shows the Power BI Data Flow interface. A context menu is open over the 'Top US Customers' query, which is highlighted with a blue border. The menu items include:

- Collapse
- Highlight related queries** (highlighted with a red box)
- Delete
- Rename
- Enable load
- Duplicate
- Reference
- Move to group
- Create function...
- Convert to parameter
- Advanced editor
- Properties...
- Append queries
- Append queries as new
- Merge queries
- Merge queries as new

Below the menu, there is a preview of the data with columns: Address, City, and ID. The data rows are:

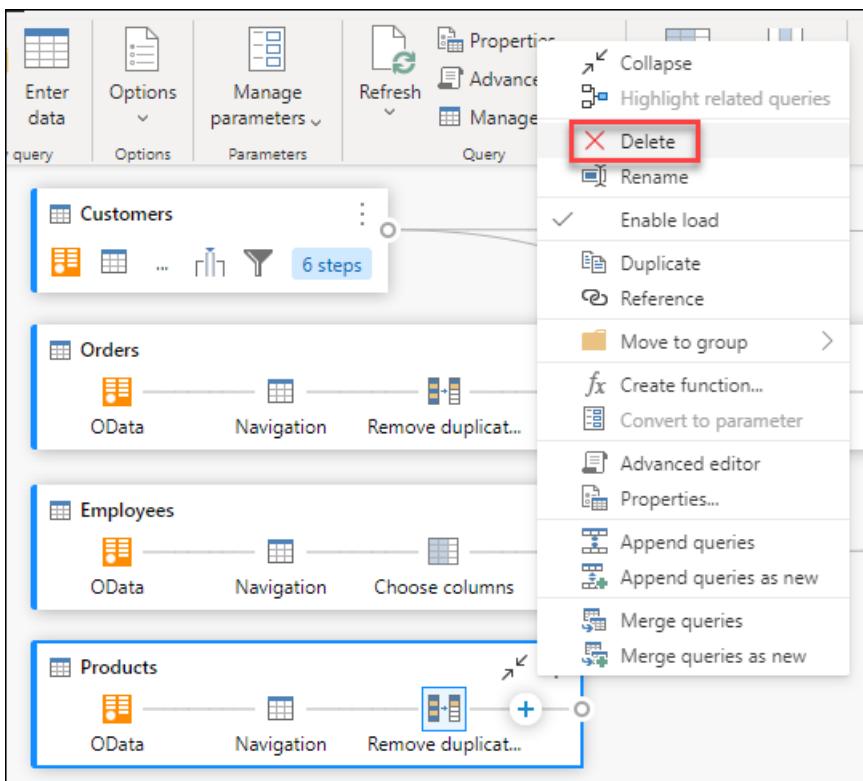
Address	City	ID
2817 Milton Dr.	Albuquerque	N
305 - 14th Ave. S. Suite 3B	Seattle	W
P.O. Box 555	Lander	W
2743 Bering St.	Anchorage	A
89 Chiaroscuro Rd.	Portland	C
89 Jefferson Way Suite 2	Portland	C
187 Suffolk Ln.	Boise	USA

For example, if you select the highlight related queries button in the **Top US Customers** query, you can see that the **Customers** and **Orders** queries are highlighted.



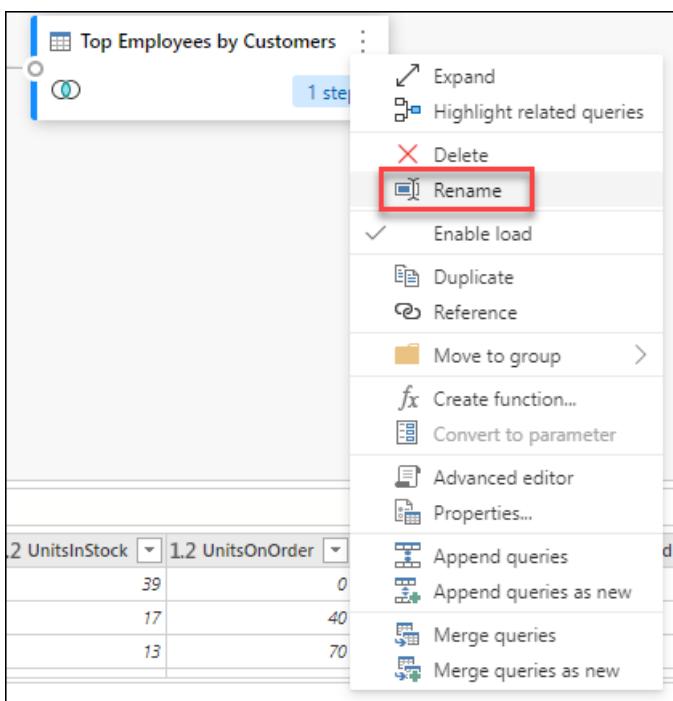
Delete query

To delete a query, right-click in a query and select **Delete** from the context menu. There will be an additional pop-up to confirm the deletion.



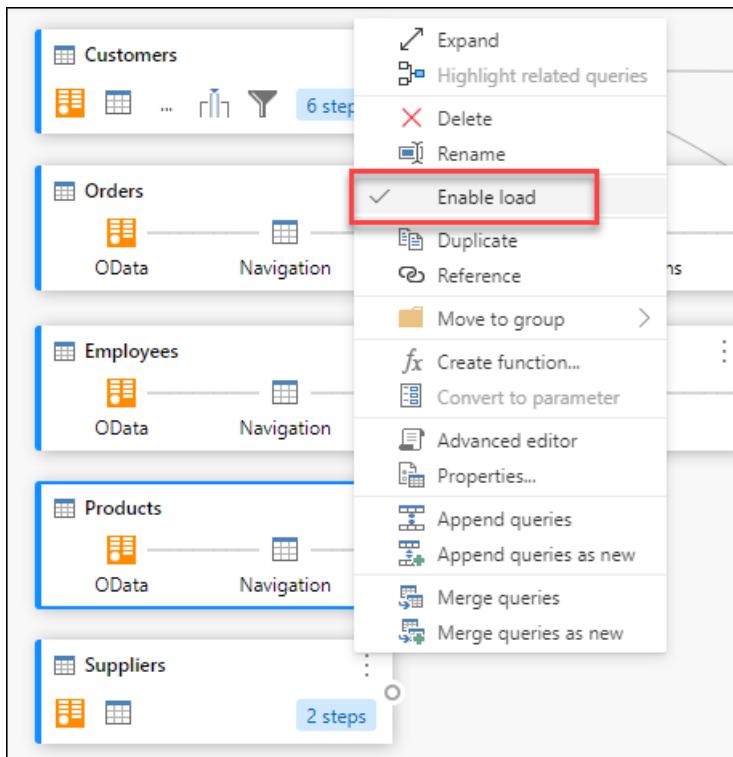
Rename query

To rename a query, right-click in a query and select **Rename** from the context menu.



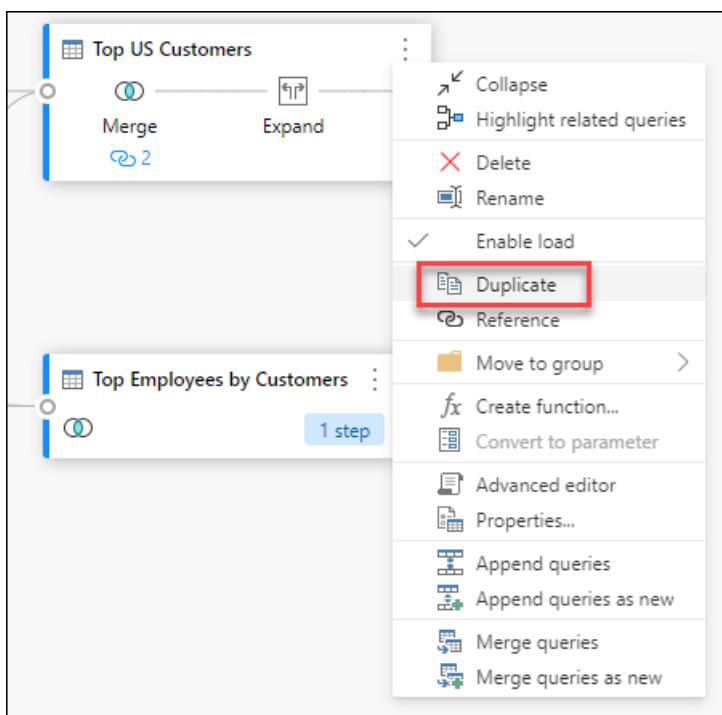
Enable load

To ensure that the results provided by the query are available for downstream use such as report building, by default **Enable load** is set to true. In case you need to disable load for a given query, right-click in a query and select **Enable load**. The queries where **Enable load** is set to false will be displayed with a grey outline.



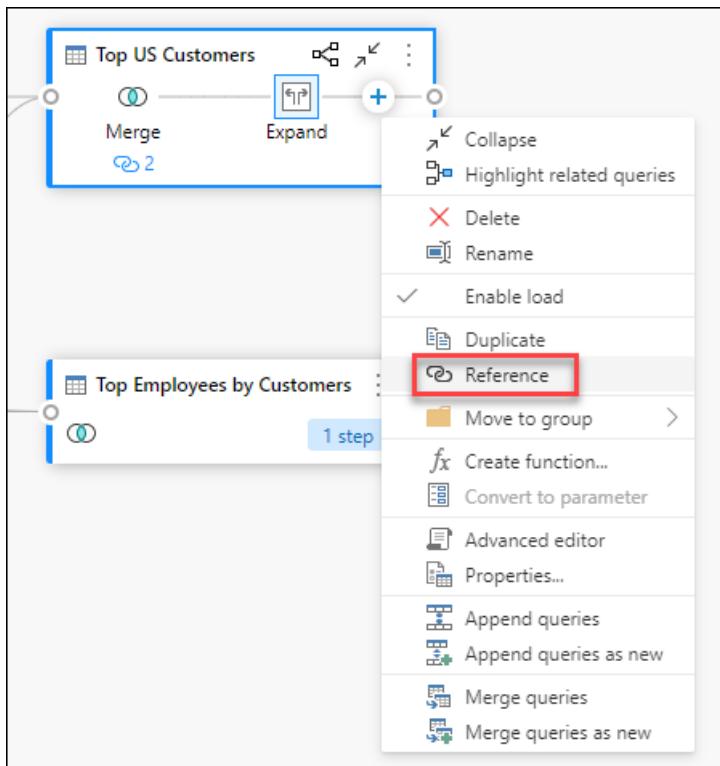
Duplicate

To create a copy of a given query, right-click in the query and select **Duplicate**. A new duplicate query will appear in the diagram view.



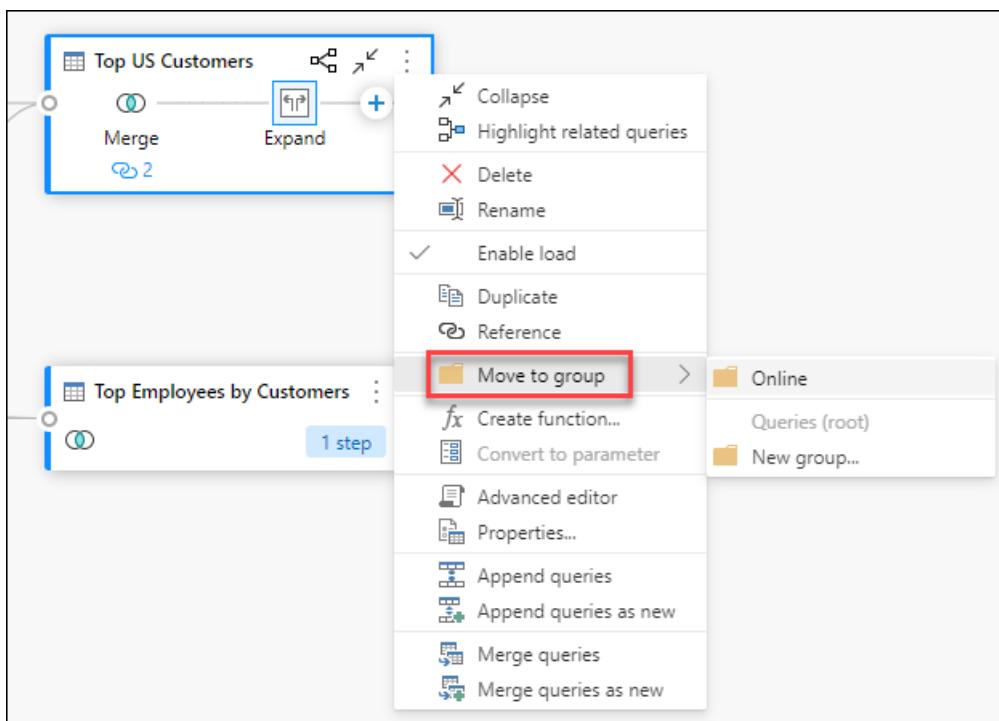
Reference

Referencing a query will create a new query. The new query will use the steps of the previous query without having to duplicate the query. Additionally, any changes on the original query will transfer down to the referenced query. To reference a query, right-click in the query and select **Reference**.

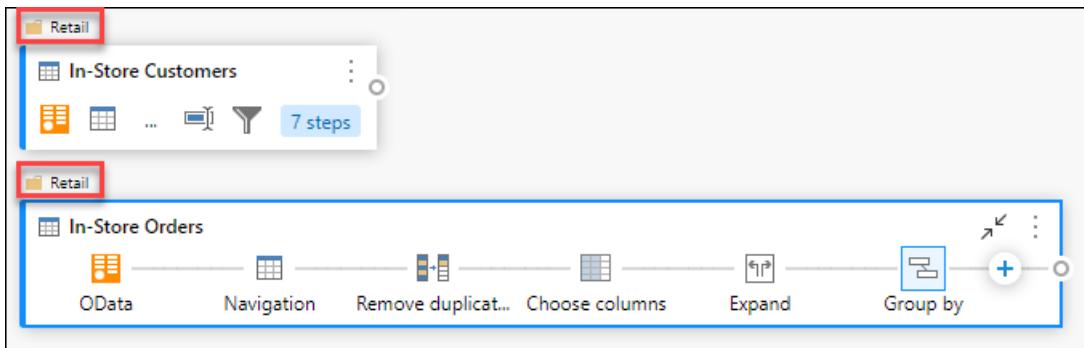


Move to group

You can make folders and move the queries into these folders for organizational purposes. These folders are called *groups*. To move a given query to a Query group, right-click in a query and select **Move to group**. You can choose to move the queries to an existing group or create a new query group.



You can view the query groups above the query box in the diagram view.

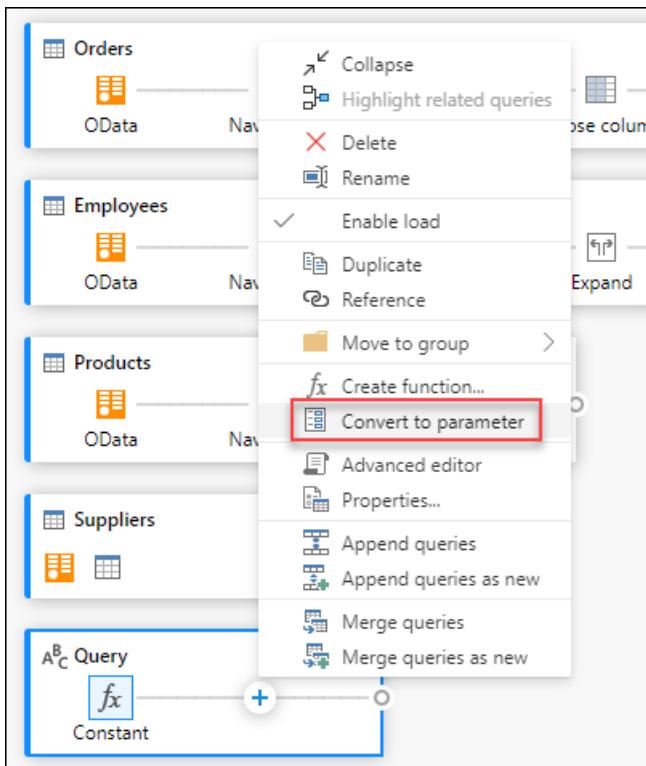


Create function

When you need to apply the same set of transformations in different queries or values, creating custom Power Query *functions* can be valuable. To learn more about custom functions, go to [Using custom functions](#). To convert a query into a reusable function, right-click in a given query and select **Create function**.

Convert to parameter

A parameter provides the flexibility to dynamically change the output of your queries depending on their value and promotes reusability. To convert a non-structured value such as date, text, number, and so on, right-click in the query and select **Convert to Parameter**.

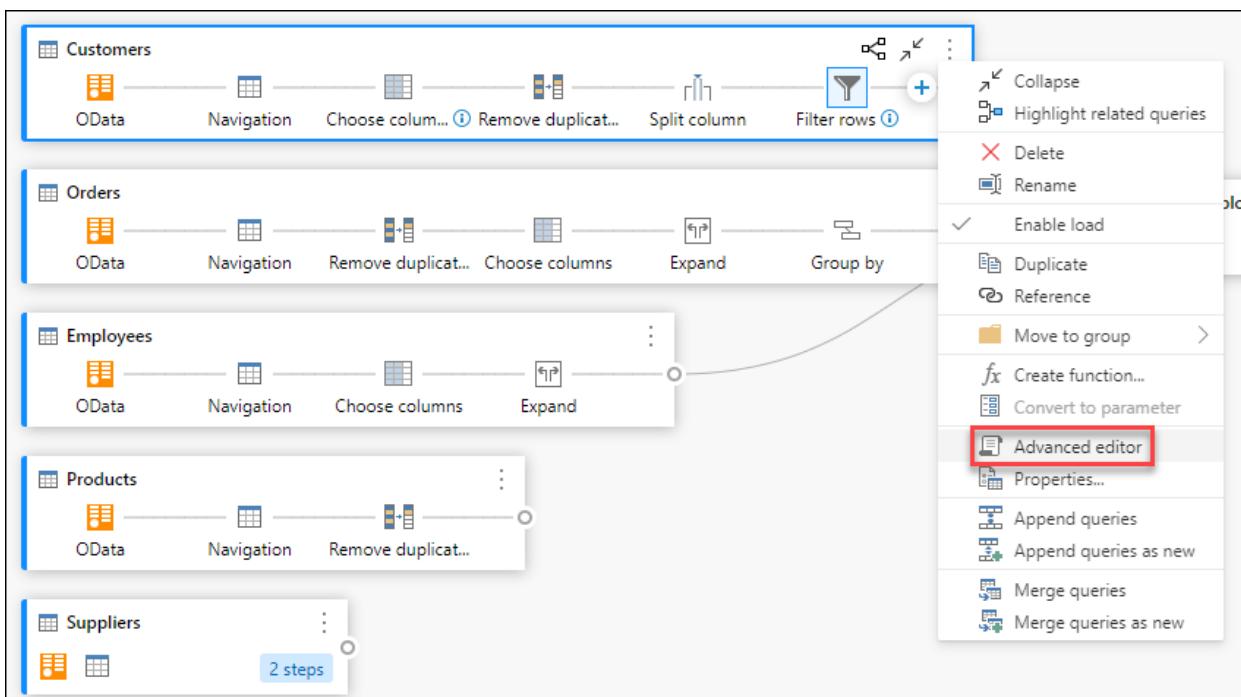


NOTE

To learn more about parameters, go to [Power Query parameters](#).

Advanced editor

With the advanced editor, you can see the code that Power Query editor is creating with each step. To view the code for a given query, right-click in the query and select **Advanced editor**.

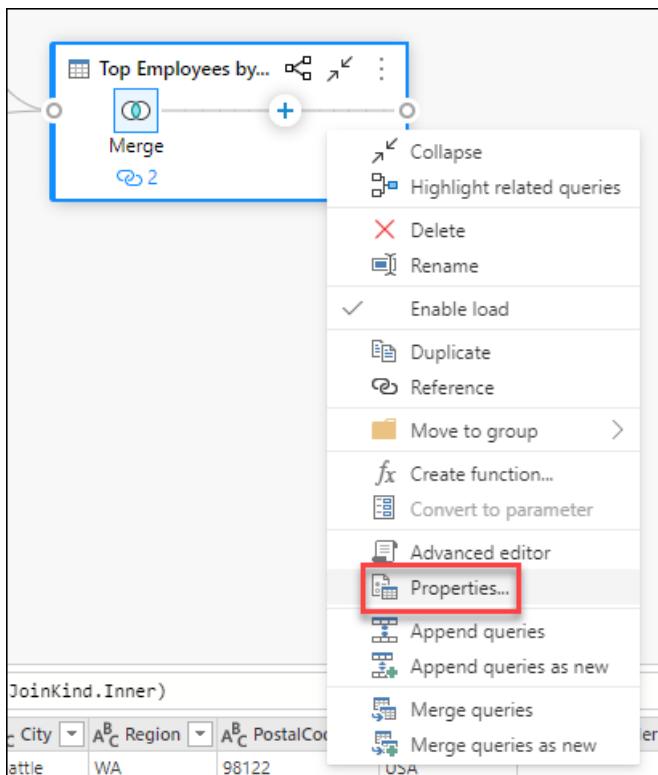


NOTE

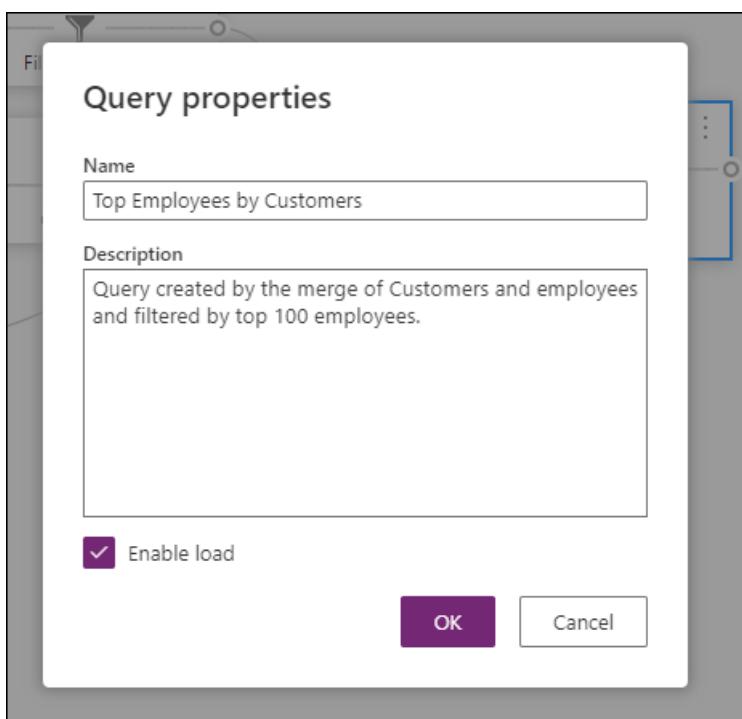
To learn more about the code used in the advanced editor, go to [Power Query M language specification](#).

Edit query name and description

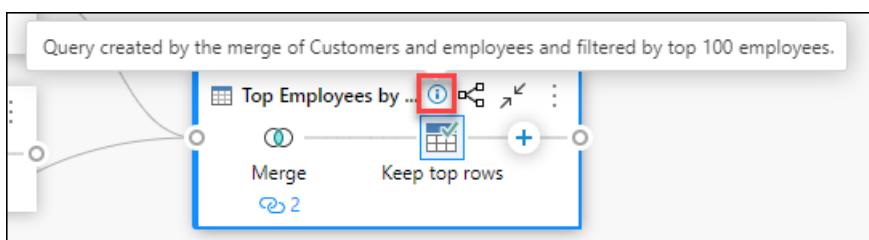
To edit the name of a query or add a description, right-click in a query and select **Properties**.



This action will open a dialog box where you can edit the name of the query or add to or modify the query description.

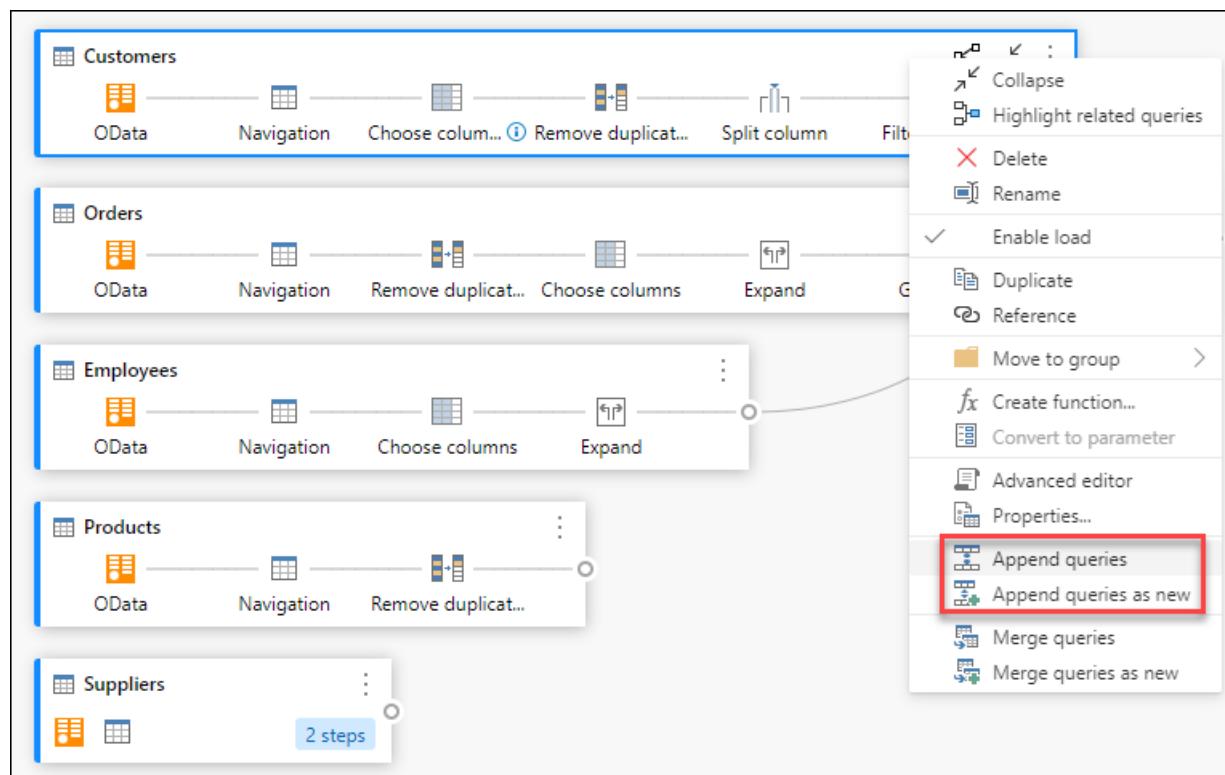


Queries with query description will have an affordance (*i* icon). You can view the query description by hovering near the query name.



Append queries/Append queries as new

To append or perform a UNION of queries, right-click in a query and select **Append queries**. This action will display the **Append** dialog box where you can add more tables to the current query. **Append queries as new** will also display the **Append** dialog box, but will allow you to append multiple tables into a new query.

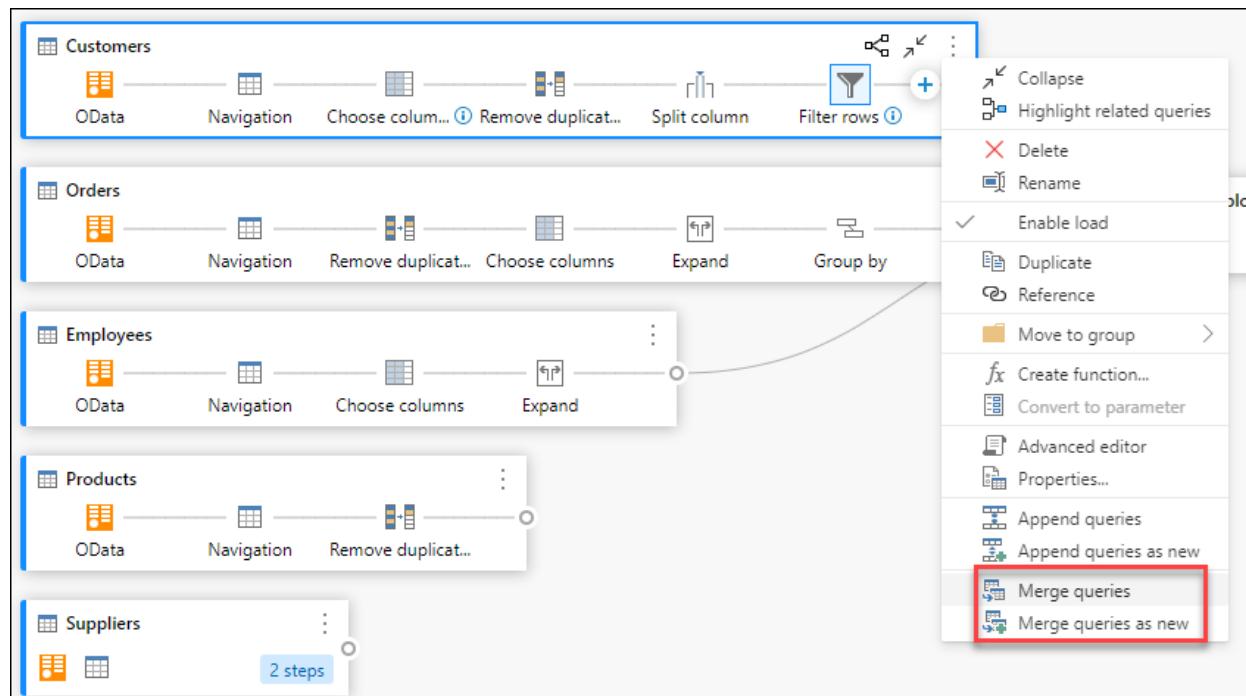


NOTE

To learn more about how to append queries in Power Query, go to [Append queries](#).

Merge queries/Merge queries as new

To merge or JOIN queries, right-click in a query and select **Merge queries**. This action will display the **Merge** dialog box, with the selected query as the left table of the merge operation. **Merge queries as new** will also display the **Merge** dialog box but will allow you to merge two tables into a new query.

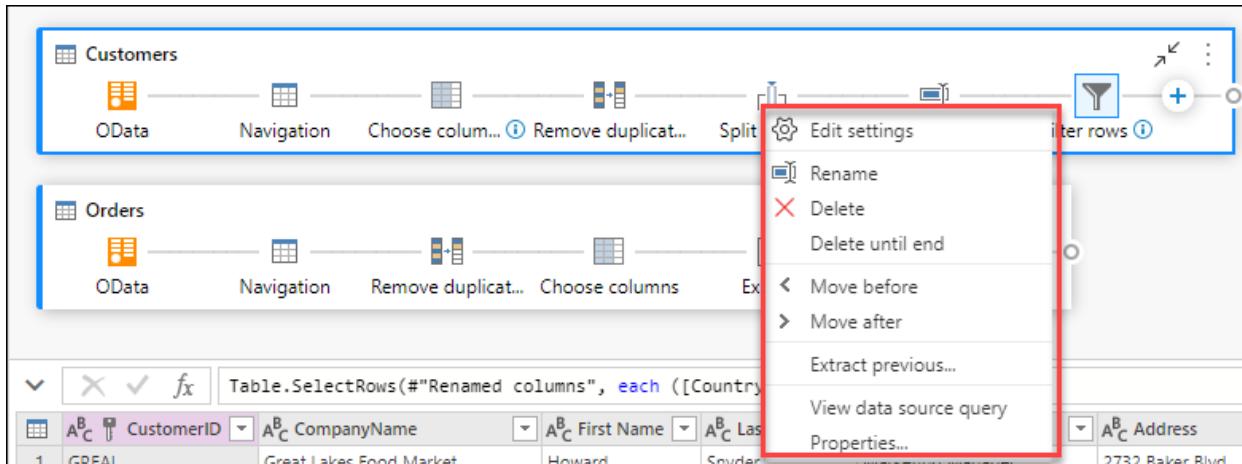


NOTE

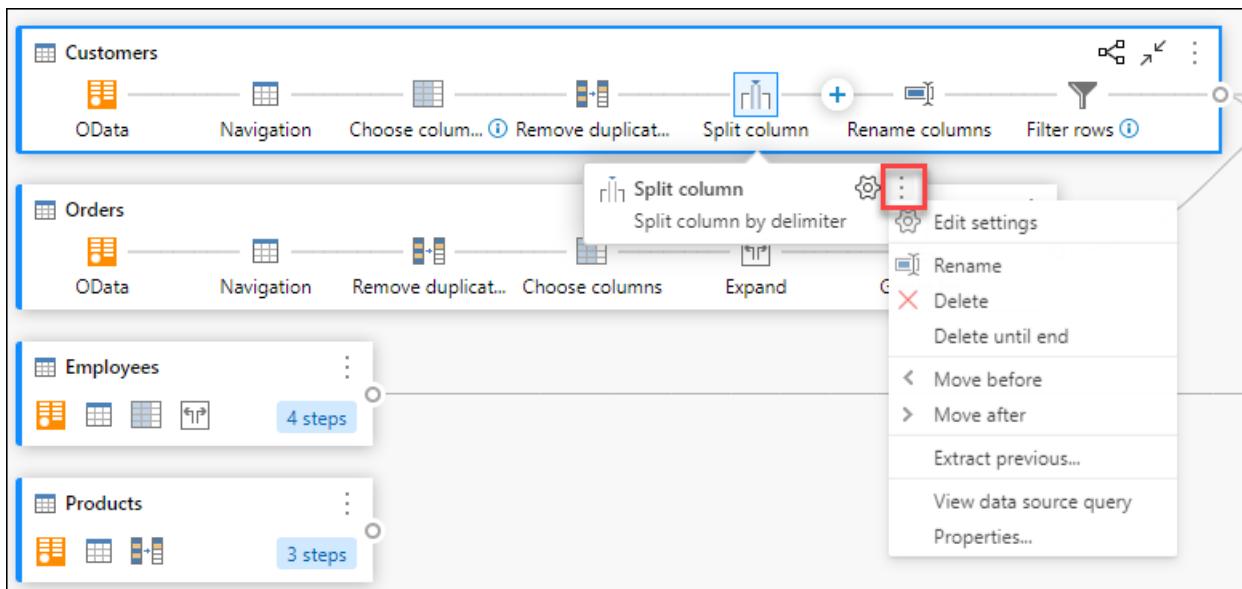
To learn more about how to merge queries in Power Query, go to [Merge queries overview](#).

Step level actions

By right-clicking a step, you can perform step level actions such as *Edit settings*, *Rename*, and so on.

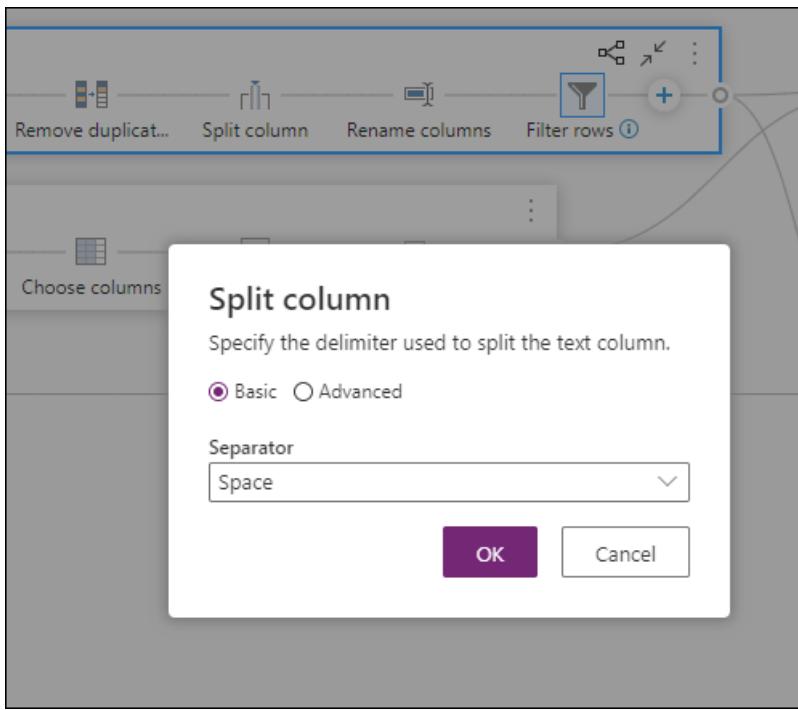


You can also perform step level actions by hovering over the step and selecting the ellipsis (three vertical dots).



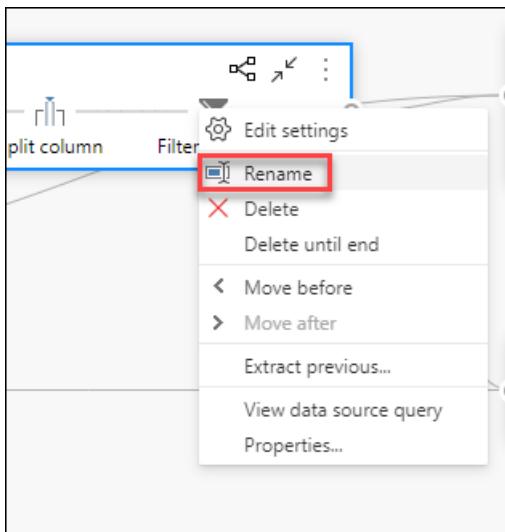
Edit settings

To edit the step level settings, right-click the step and choose **Edit settings**. Instead, you can double-click the step (that has step settings) and directly get to the settings dialog box. In the settings dialog box, you can view or change the step level settings. For example, the following image shows the settings dialog box for the **Split column** step.



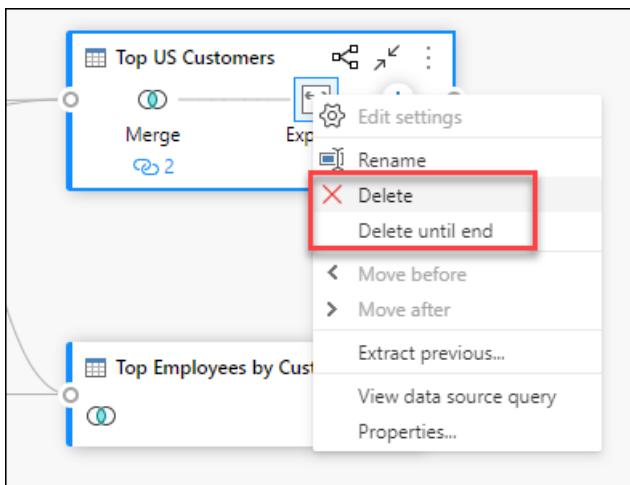
Rename step

To rename a step, right-click the step and select **Rename**. This action opens the **Step properties** dialog. Enter the name you want, and then select **OK**.



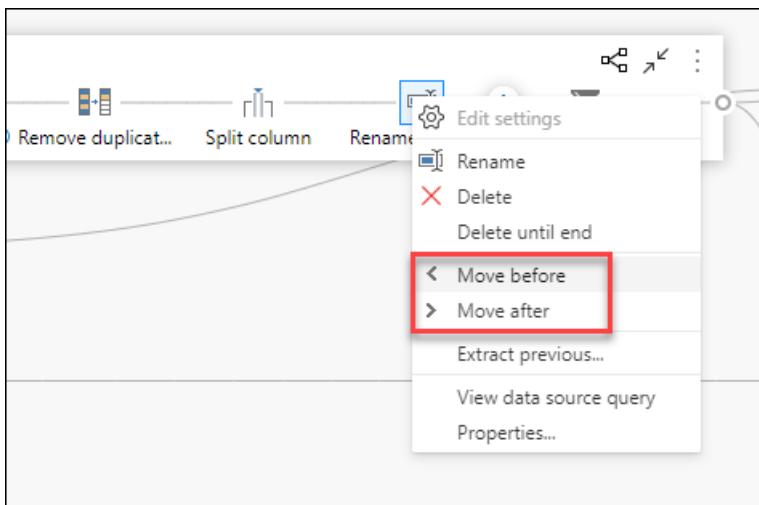
Delete step

To delete a step, right-click the step and select **Delete**. To delete a series of steps until the end, right-click the step and select **Delete until end**.



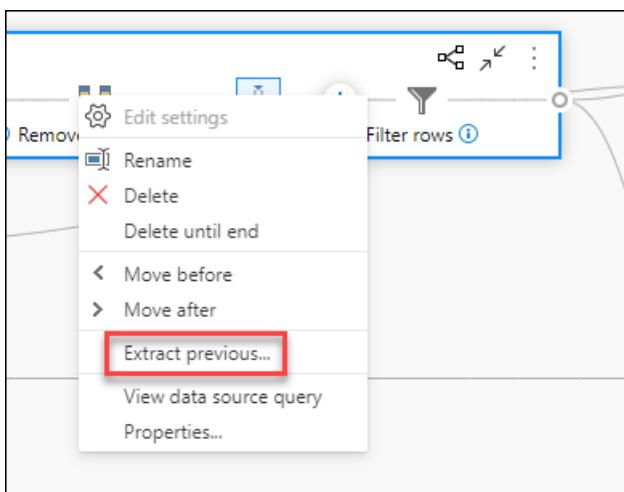
Move before/Move after

To move a step one position before, right-click a step and select **Move before**. To move a step one position after, right-click a step and select **Move after**.



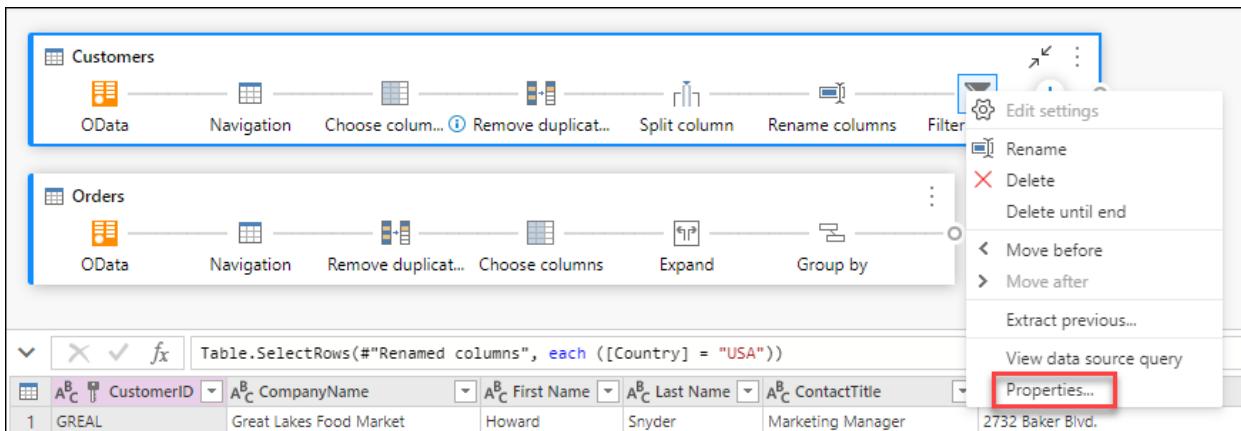
Extract previous

To extract all previous steps into a new query, right-click the first step that you do *not* want to include in the query and then select **Extract previous**.

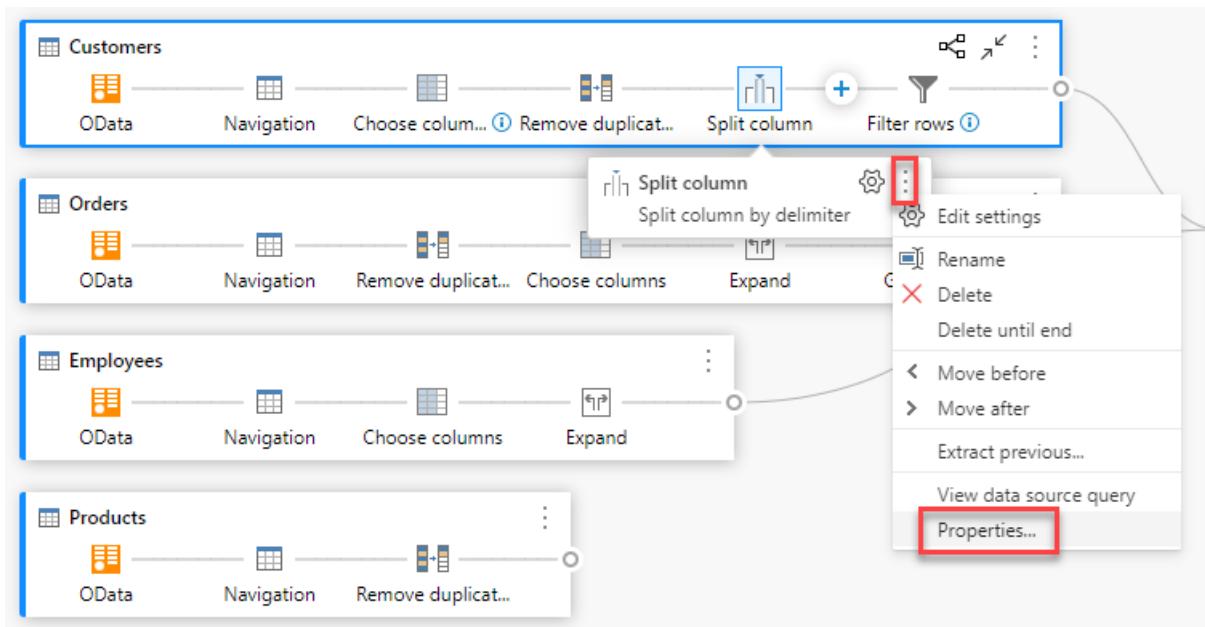


Edit step name and description

To add step descriptions, right-click a step in a query and then choose **Properties**.



You can also get to the step level context menu by hovering over the step and selecting the ellipsis (three vertical dots).



This action will open a dialog box where you can add the step description. This step description will come handy when you come back to the same query after a few days or when you share your queries or dataflows with other users.

The screenshot shows the Power BI Data Flow interface with the 'Step properties' dialog box open. The dialog has two main sections: 'Name' and 'Description'. The 'Name' field contains the value 'Filtered rows'. The 'Description' field contains the text 'Filter by US'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons. In the background, the Power BI Data Flow interface is visible, showing a data pipeline with various steps and a preview of the 'Customers' data table.

CustomerID	CompanyName	First Name	Last Name	Region
1	Great Lakes Food Market	Howard	Snyder	Mar
2	Hungry Coyote Import Store	Yoshi	Latimer	Sale
3	Lazy K Kountry Store	John	Steel	Mar
4	Let's Stop N Shop	Jaime	Yorres	Own
5	Lonesome Pine Restaurant	Fran	Wilson	Sale
6	Old World Delicatessen	Rene	Phillips	Sale
7	Rattlesnake Canyon Grocery	Paula	Wilson	Ass
8	Save-a-lot Markets	Jose	Pavarotti	Sale
9	Split Rail Beer & Ale	Art	Braunschweiger	Sale
10	The Big Cheese	Liz	Nixon	Marketing Manager

By hovering over each step, you can view a call out that shows the step label, step name, and step descriptions (that were added).

Power Query - Edit queries

Queries

Customers

Orders

Table.SelectRows(#"Renamed columns", each ([Country] = "USA"))

A _C #	A _C CustomerID	A _C CompanyName	A _C First Name	A _C Last Name	A _C ContactTitle	A _C Address	A _C City	A _C Region	A _C PostalCode	A _C Country	A _C Phone
1	GREAL	Great Lakes Food Market	Howard	Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR	97403	USA	(503) 555-7555
2	HUNGC	Hungry Coyote Import Store	Yoshi	Latimer	Sales Representative	City Center Plaza 516 Main St.	Eugine	OR	97827	USA	(503) 555-6874
3	LAZYK	Lazy K Kountry Store	John	Steel	Marketing Manager	12 Orchestra Terrace	Walla Walla	WA	99362	USA	(509) 555-7989
4	LETSS	Let's Stop N Shop	Jaime	Yorres	Owner	87 Pox St. Suite 5	San Francisco	CA	94117	USA	(415) 555-5938
5	LONEP	Lonesome Pine Restaurant	Fran	Wilson	Sales Manager	89 Chiaroscuro Rd.	Portland	OR	97219	USA	(503) 555-9573
6	OLDWO	Old World Delicatessen	Rene	Phillips	Sales Representative	2743 Bering St.	Anchorage	AK	99508	USA	(907) 555-7584
7	RATTG	Rattlesnake Canyon Grocery	Paula	Wilson	Assistant Sales Representative	2817 Milton Dr.	Albuquerque	NM	87110	USA	(505) 555-5939
8	SAVEA	Save-a-lot Markets	Jose	Pavarotti	Sales Representative	187 Suffolk Ln.	Boise	ID	83720	USA	(208) 555-8097
9	SPLIR	Split Rail Beer & Ale	Art	Braunschweiger	Sales Manager	P.O. Box 555	Lander	WY	82520	USA	(307) 555-4680
10	THEBI	The Big Cheese	Liz	Nixon	Marketing Manager	89 Jefferson Way Suite 2	Portland	OR	97201	USA	(503) 555-3612
11	THECR	The Cracker Box	Liu	Wong	Marketing Assistant	55 Grizzly Peak Rd.	Butte	MT	59801	USA	(406) 555-5834
12	TRAIH	Trail's Head Gourmet Provisioners	Helvetius	Nagy	Sales Associate	722 DaVinci Blvd.	Kirkland	WA	98034	USA	(206) 555-8257
13	WHITC	White Clover Markets	Karl	Jablonski	Owner	305 - 14th Ave. S. Suite 3B	Seattle	WA	98128	USA	(206) 555-4112

Columns: 11 Rows: 13

By selecting each step, you can see the corresponding data preview for that step.

Expand and collapse queries

To ensure that you can view your queries in the diagram view, you can collapse the ones that you aren't actively working on and expand the ones that you care about. Expand or collapse queries by selecting the **Expand/Collapse** button on the top-right of a query. Alternatively, double-clicking an expanded query will collapse the query and vice-versa.

Customers

OData Navigation Choose colum... Remove duplicat... Split column Rename columns Filter rows

You can also expand or collapse a query by selecting the query level actions from the query's context menu.

Orders

OData Navigation Remove duplicat... Choose columns Expand Group by

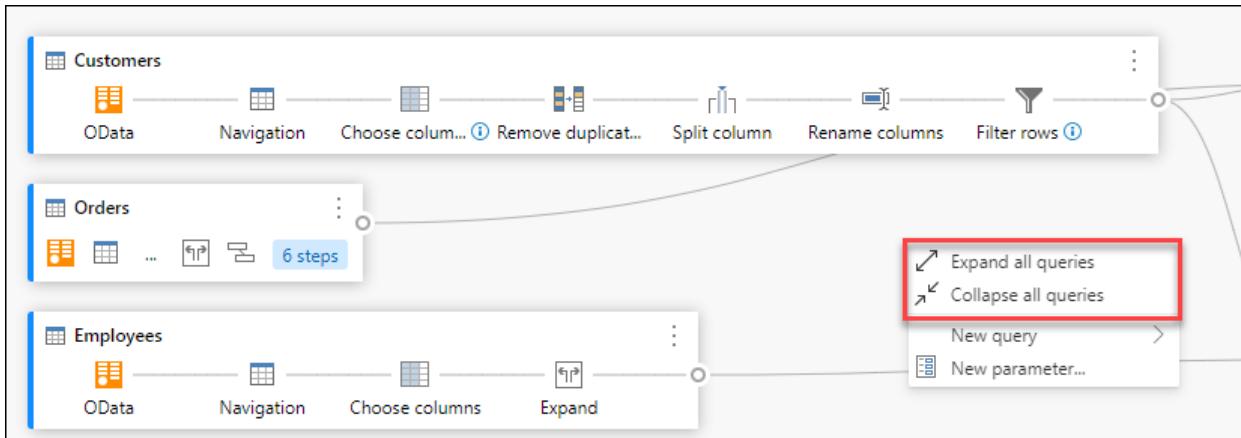
Table.ExpandTableColumn(Source, "Orders", {"Count"}, {"Orders.Count"})

A _C #	A _C CustomerID	A _C CompanyName	A _C First Name	A _C Last Name	A _C ContactTitle
1	RATTG	Rattlesnake Canyon Grocery	Paula	Wilson	Assistant Sales Representative
2	WHITC	White Clover Markets	Karl	Jablonski	Owner
3	SPLIR	Split Rail Beer & Ale	Art	Braunschweiger	Sales Manager
4	OLDWO	Old World Delicatessen	Rene	Phillips	Sales Representative
5	LONEP	Lonesome Pine Restaurant	Fran	Wilson	Sales Manager
6	THEBI	The Big Cheese	Liz	Nixon	Marketing Manager
7	SAVEA	Save-a-lot Markets	Jose	Pavarotti	Sales Representative
8	HUNGC	Hungry Coyote Import Store	Yoshi	Latimer	Sales Representative
9	LAZYK	Lazy K Kountry Store	John	Steel	Marketing Manager
10	GREAL	Great Lakes Food Market	Howard	Snyder	Marketing Manager
11	TRAIH	Trail's Head Gourmet Provisioners	Helvetius	Nagy	Sales Associate
12	LETSS	Let's Stop N Shop	Jaime	Yorres	Owner
13	THECR	The Cracker Box	Liu	Wong	Marketing Assistant

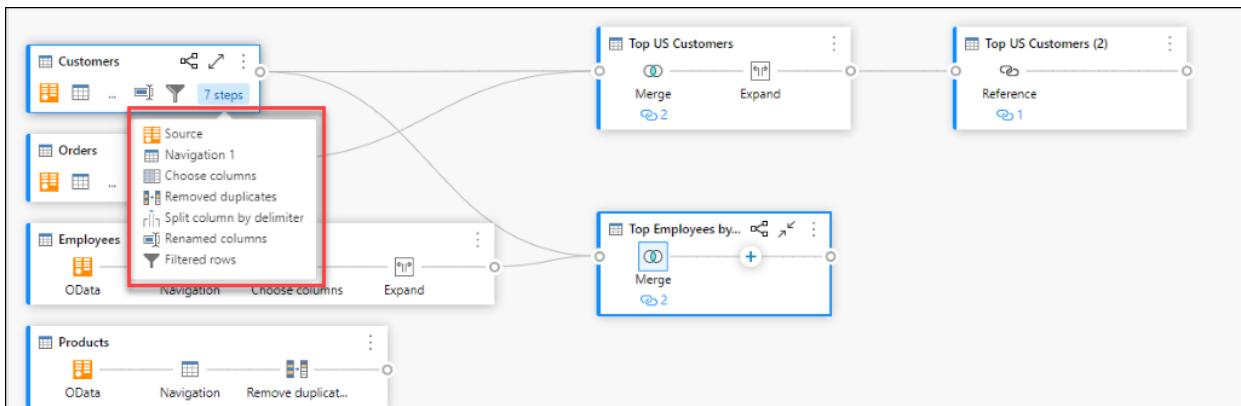
To expand all or collapse all queries, select the **Expand all/Collapse all** button next to the layout options in the diagram view pane.



You can also right-click any empty space in the diagram view pane and see a context menu to expand all or collapse all queries.



In the collapsed mode, you can quickly look at the steps in the query by hovering over the number of steps in the query. You can select these steps to navigate to that specific step within the query.

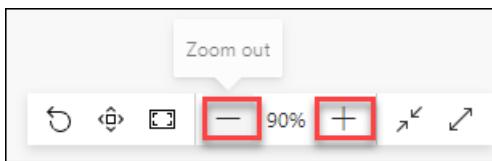


Layout Options

There are four layout options available in the diagram view: zoom out, zoom in, full screen, fit to view, and reset.

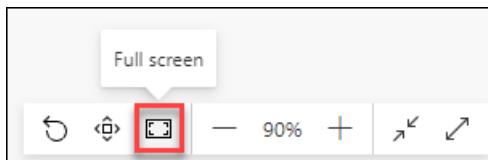
Zoom out/zoom in

With this option, you can adjust the zoom level and zoom out or zoom in to view all the queries in the diagram view.



Full screen

With this option, you can view all the queries and their relationships through the *Full screen* mode. The diagram view pane expands to full screen and the data preview pane, queries pane, and steps pane remain collapsed.



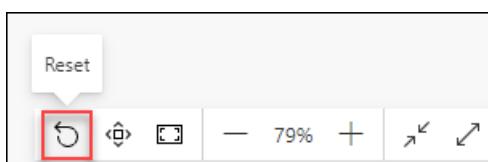
Fit to view

With this option, you can adjust the zoom level so that all the queries and their relationships can be fully viewed in the diagram view.



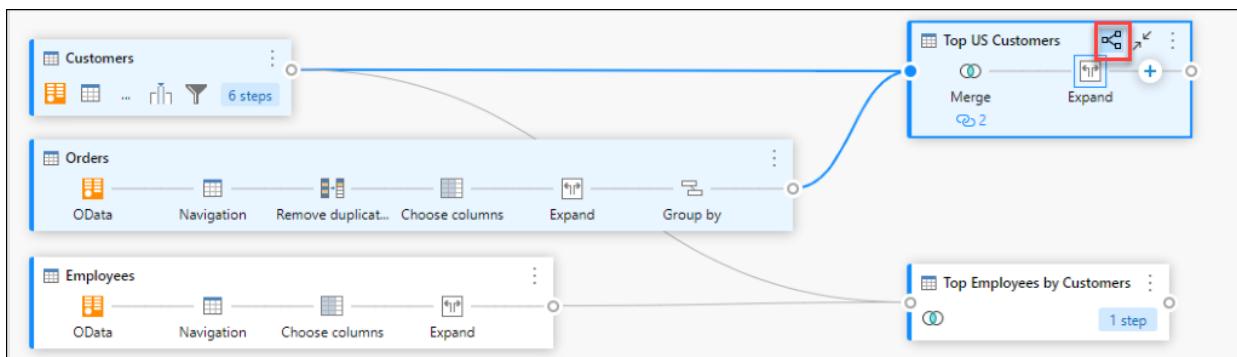
Reset

With this option, you can reset the zoom level back to 100% and also reset the pane to the top-left corner.

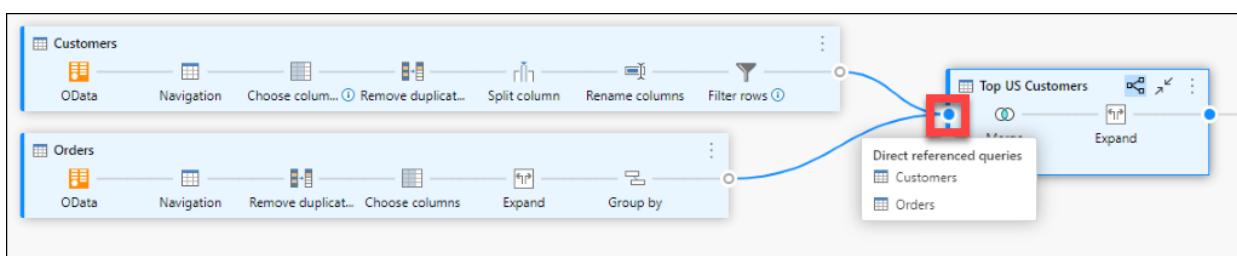


View query relationships

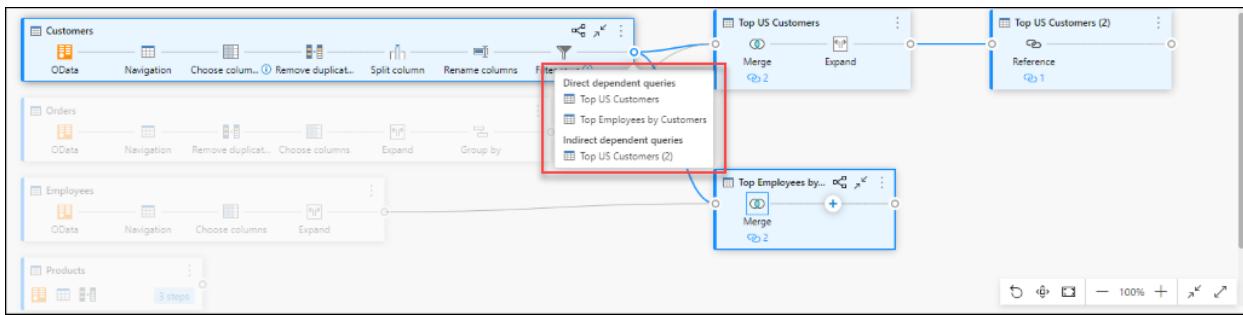
To view all the related queries for a given query, select the **Highlight related queries** button. For instance, by selecting the highlight related queries button in the **Top US Customers** query, the **Customers** and **Orders** queries are highlighted, as shown in the following image.



You can also select the dongle on the left of a given query to see the direct and indirect referenced queries.



Similarly, you can select the right dongle to view direct and indirect dependent queries.



You can also hover on the link icon below a step to view a callout that shows the query relationships.

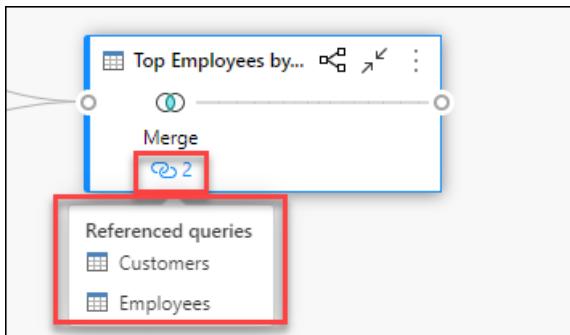
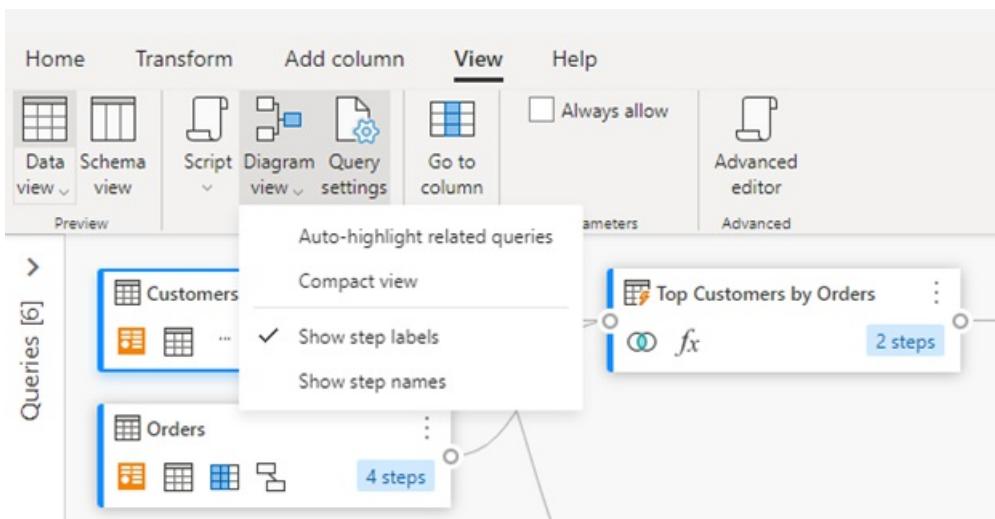


Diagram view settings

To modify diagram view settings, select the lower half of the **Diagram View** button inside the **View** tab in the ribbon.

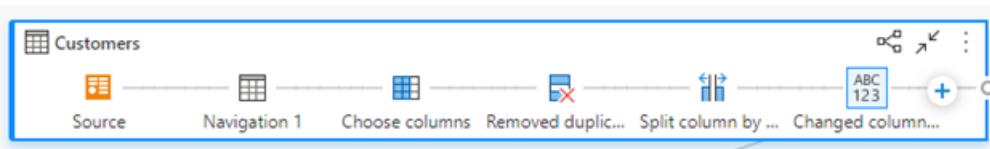


Step labels and step names

We show **step labels** by default within the diagram view.

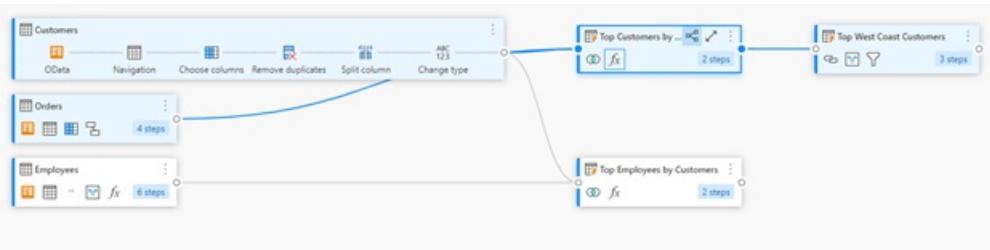


You can change diagram view settings to show **step names** to match the **applied steps** within the **query settings** pane.



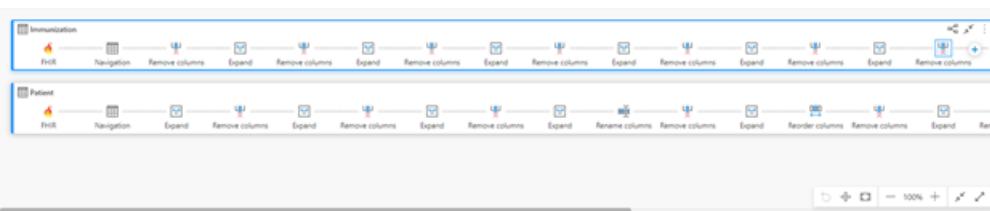
Auto-highlight related queries

By selecting **Auto-highlight related queries** within diagram view settings, related queries are always highlighted so that you can visually see the query dependencies better

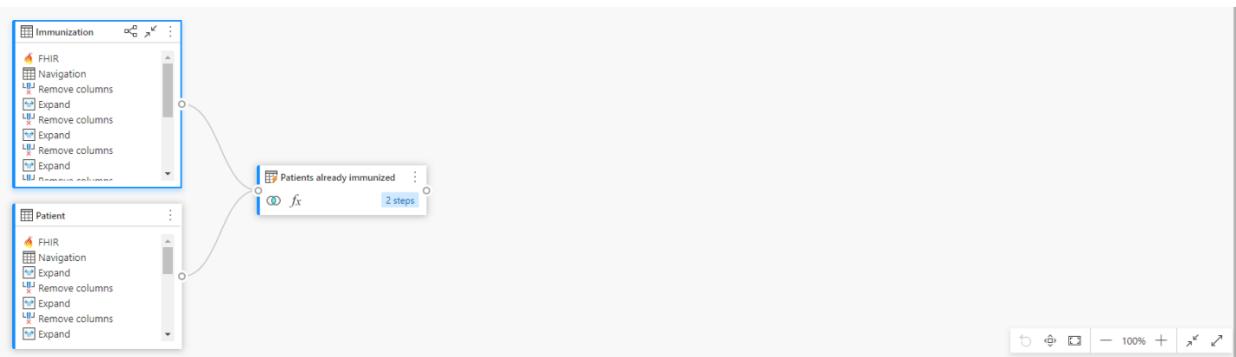


Compact view

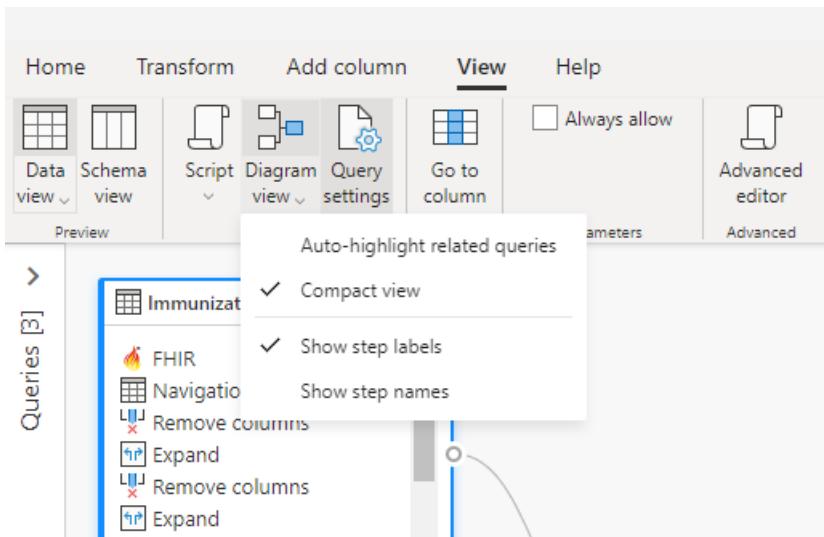
When you have queries with multiple steps, it can be challenging to scroll horizontally to view all your steps within the viewport.



To address this, diagram view offers **Compact view**, which compresses the steps from top to bottom instead of left to right. This view can be especially useful when you have queries with multiple steps, so that you can see as many queries as possible within the viewport.



To enable this view, navigate to diagram view settings and select **Compact view** inside the **View** tab in the ribbon.



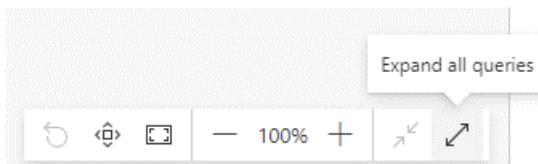
Maximize data preview

You may want to see more data within the data preview to understand and analyze the data. To do so, expand the data preview so that you can see as much data as before within the data preview without leaving diagram View.

	CustomerID	CompanyName	ContactName.1	ContactName.2	ContactName.3	ContactTitle	Address	City	Region	PostalCode	Country	Phone
1	ALFKI	Alfreds Futterkiste	Maria	Anders		Sales Representative	Obere Str. 57	Berlin	null	12209	Germany	030-0074321
2	ANATR	Ana Trujillo Emparedados y helados	Ana	Trujillo		Owner	Avenida de la Constitución 22...	México D.F.	null	05021	Mexico	(5) 555-4729
3	ANTON	Antonio Moreno Taquería	Antonio	Moreno		Owner	Mataderos 2312	México D.F.	null	05023	Mexico	(5) 555-3932
4	AROUT	Around the Horn	Thomas	Hardy		Sales Representative	120 Hanover Sq.	London	null	WA1 1DP	UK	(171) 555-77...
5	BERGS	Berglunds snabbköp	Christina	Berglund		Order Administrator	Berguvägen 8	Luleå	null	958 22	Sweden	092-12 34 65
6	BLAUS	Blauer See DelikatesSEN	Hanna	Moos		Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany	0621-06460
7	BLONP	Blondedel'ss pére et fils	Frédérique	Côteaux		Marketing Manager	24, place Kléber	Strasbourg	null	67000	France	88.60.15.31
8	BOUD	Bólido Comidas preparadas	Martin	Sommer		Owner	C/ Arquill, 67	Madrid	null	28023	Spain	(91) 555 22 82
9	BONAP	Bon app'	Laurence	Lebihan		Owner	12, rue des Bouchers	Marseille	null	13008	France	9124.45.40
10	BOTTM	Bottom-Dollar Markets	Elizabeth	Lincoln		Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-47...

Expand or collapse all queries

By default, the queries within diagram view are collapsed. There are options to expand or collapse each query in a single click.

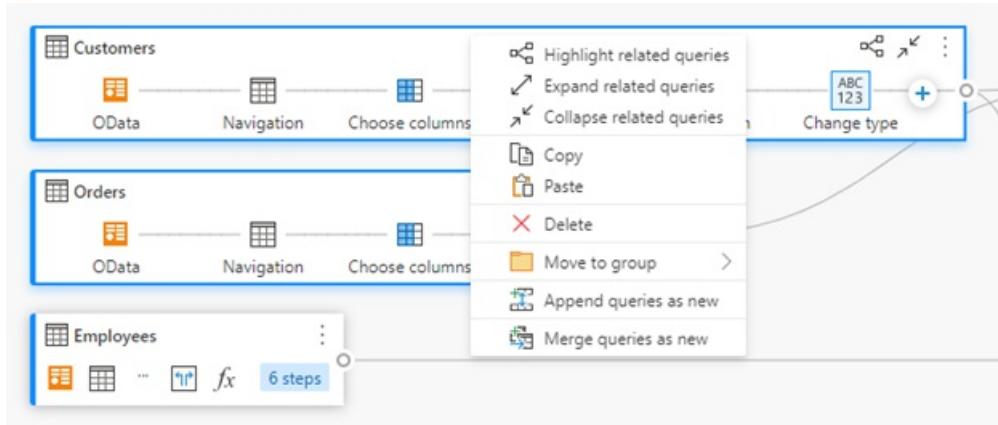


You can also expand or collapse related queries from the query level context menu.

	CustomerID	CompanyName	ContactName.1	ContactName.2	ContactName.3	ContactTitle	Address	City	Region	PostalCode	Country	Phone
1	ALFKI	Alfreds Futterkiste	Maria	Anders		Sales Representative	Obere Str. 57	Berlin	null	12209	Germany	030-0074321
2	ANATR	Ana Trujillo Emparedados y helados	Ana	Trujillo		Owner	Avenida de la Constitución 22...	México D.F.	null	05021	Mexico	(5) 555-4729
3	ANTON	Antonio Moreno Taquería	Antonio	Moreno		Owner	Mataderos 2312	México D.F.	null	05023	Mexico	(5) 555-3932
4	AROUT	Around the Horn	Thomas	Hardy		Sales Representative	120 Hanover Sq.	London	null	WA1 1DP	UK	(171) 555-77...
5	BERGS	Berglunds snabbköp	Christina	Berglund		Order Administrator	Berguvägen 8	Luleå	null	958 22	Sweden	092-12 34 65
6	BLAUS	Blauer See DelikatesSEN	Hanna	Moos		Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany	0621-06460
7	BLONP	Blondedel'ss pére et fils	Frédérique	Côteaux		Marketing Manager	24, place Kléber	Strasbourg	null	67000	France	88.60.15.31
8	BOUD	Bólido Comidas preparadas	Martin	Sommer		Owner	C/ Arquill, 67	Madrid	null	28023	Spain	(91) 555 22 82
9	BONAP	Bon app'	Laurence	Lebihan		Owner	12, rue des Bouchers	Marseille	null	13008	France	9124.45.40
10	BOTTM	Bottom-Dollar Markets	Elizabeth	Lincoln		Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-47...

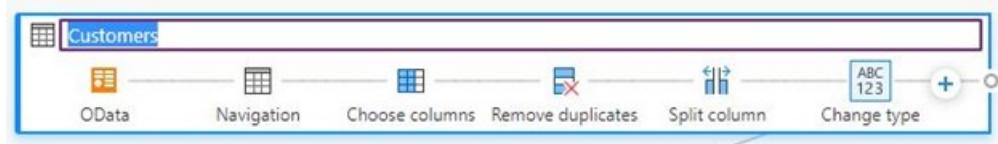
Multi-select queries

You select multiple queries within the diagram view by holding down the Ctrl key and clicking queries. Once you multi-select, right-clicking will show a context menu that allows performing operations such as merge, append, move to group, expand/collapse and more.

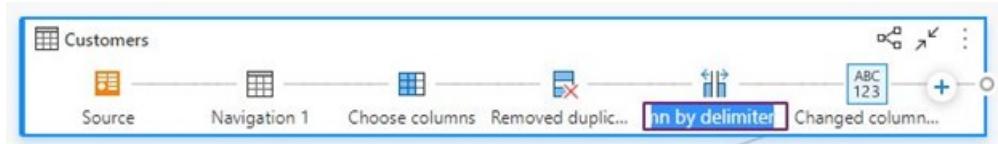


Inline rename

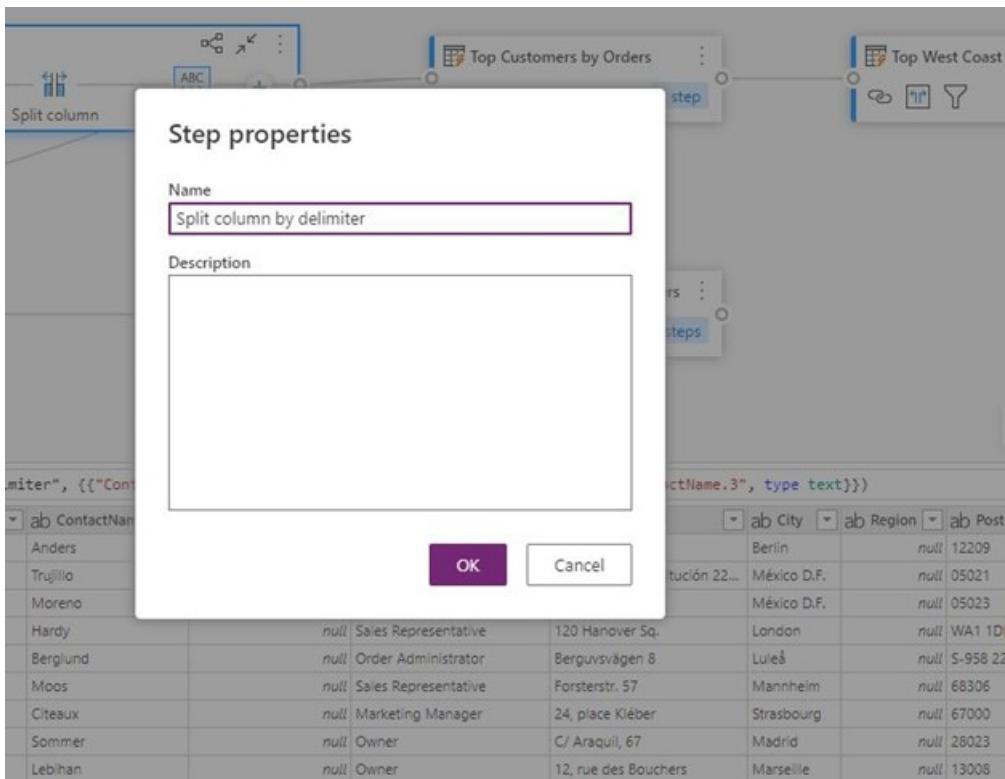
You can double-click the query name to rename the query.



Double-clicking the step name allows you to rename the step, provided the diagram view setting is showing step names.



When step labels are displayed in diagram view, double-clicking the step label shows the dialog box to rename the step name and provide a description.



Accessibility

Diagram view supports accessibility features such as keyboard navigation, high-contrast mode, and screen reader support. The following table describes the keyboard shortcuts that are available within diagram view. To learn more about keyboard shortcuts available within Power Query Online, see [keyboard shortcuts in Power Query](#).

ACTION	KEYBOARD SHORTCUT
Expand selected query	Ctrl+Right arrow key
Collapse selected query	Ctrl+Left arrow key
Move focus from query level to step level	Alt+Down arrow key
Move focus from step level to query level	Esc
Expand all queries	Ctrl+Shift+Right arrow key
Collapse all queries	Ctrl+Shift+Left arrow key
Insert new step using + button (after selected step)	Ctrl+Alt+N
Highlight related queries	Ctrl+Alt+R
Select all queries	Ctrl+A
Copy queries	Ctrl+C
Paste queries	Ctrl+V

Using Schema view (Preview)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Schema view is designed to optimize your flow when working on schema level operations by putting your query's column information front and center. Schema view provides contextual interactions to shape your data structure, and lower latency operations as it only requires the column metadata to be computed and not the complete data results.

This article walks you through schema view and the capabilities it offers.

A screenshot of the Microsoft Power Query Editor interface. The top navigation bar includes Home, Transform, Add column, View, and Schema tools, with Schema tools being the active tab. Below the tabs is a toolbar with various icons for refresh, properties, advanced editor, manage columns, detect data type, and transform. The main workspace shows a list of columns from a query named 'Employees'. The columns listed are: EmployeeID, LastName, FirstName, Title, TitleOfCourtesy, BirthDate, HireDate, Address, City, Region, PostalCode, Country, HomePhone, Extension, Photo, Notes, and ReportsTo. Each column entry includes a name, type (e.g., Whole number, Text, Date/Time, Binary), and a key indicator. To the left of the main workspace is a sidebar titled 'Queries' containing a list of available tables: Categories, Customers, Employees (which is selected and highlighted in grey), Invoices, Orders, Products, Regions, and Suppliers. On the right side, there is a panel titled 'Query settings' with 'Name' set to 'Employees', and another panel titled 'Applied steps' showing a step named 'Source' and 'Navigation 1'.

NOTE

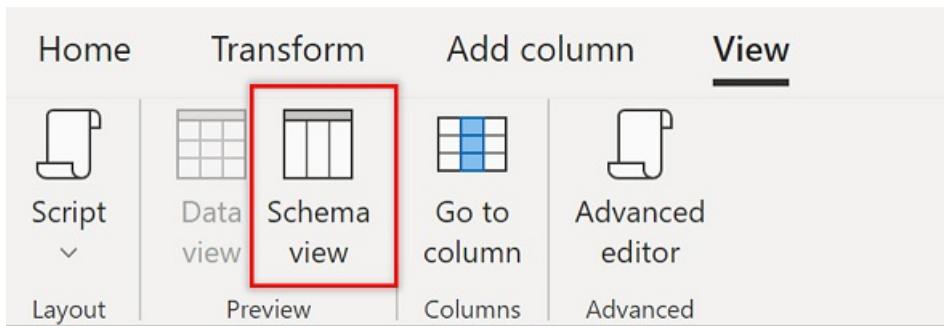
The Schema view feature is available only for Power Query Online.

Overview

When working on data sets with many columns, simple tasks can become incredibly cumbersome because even finding the right column by horizontally scrolling and parsing through all the data is inefficient. Schema view displays your column information in a list that's easy to parse and interact with, making it easier than ever to work on your schema.

In addition to an optimized column management experience, another key benefit of schema view is that transforms tend to yield results faster. These results are faster because this view only requires the columns information to be computed instead of a preview of the data. So even working with long running queries with a few columns will benefit from using schema view.

You can turn on schema view by selecting **Schema view** in the **View** tab. When you're ready to work on your data again, you can select **Data view** to go back.



Reordering columns

One common task when working on your schema is reordering columns. In Schema View this can easily be done by dragging columns in the list and dropping in the right location until you achieve the desired column order.

The screenshot shows the Schema view interface. It displays a list of columns with their names, data types, and keys. A green horizontal bar highlights a range of columns from 'CategoryName' to 'CategoryName'. The columns listed are: CategoryID (Whole number, Key), CategoryName (Text, checked), Description (Text), Picture (Binary), and Products (Table).

Name	Type	Key
CategoryID	Whole number	
CategoryName	Text	✓
Description	Text	
Picture	Binary	✓
Products	Table	

Applying transforms

For more advanced changes to your schema, you can find the most used column-level transforms right at your fingertips directly in the list and in the Schema tools tab. Plus, you can also use transforms available in other tabs on the ribbon.

The screenshot shows the Schema tools tab selected in the ribbon. The tab includes options for Refresh, Properties, Advanced editor, Manage, Query, Remove columns, Remove other columns, Manage columns, Data type (set to Whole number), Detect data type, Mark as key, Transform (Rename, Duplicate, Move column), Close Schema view, and Close.

Share a query

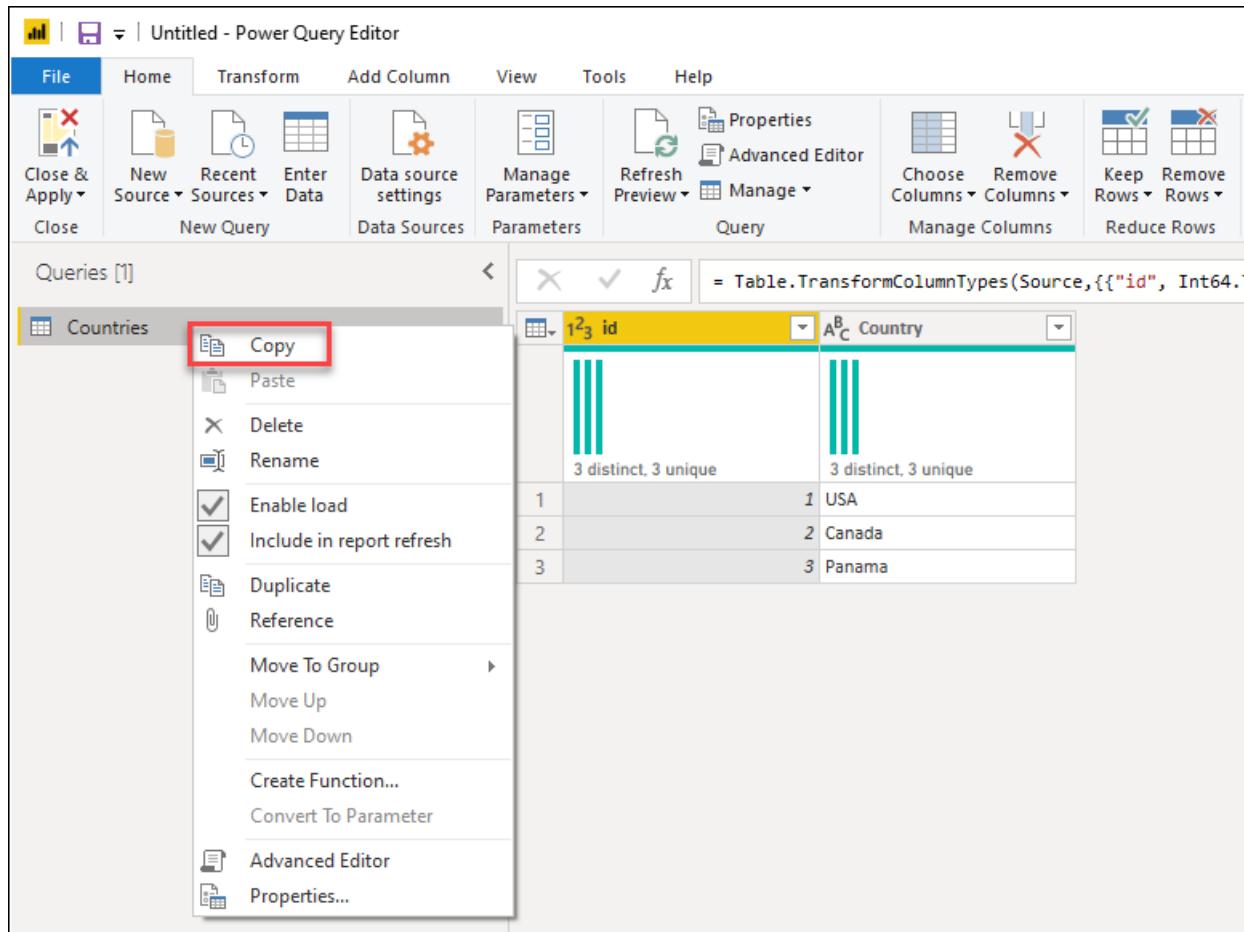
1/15/2022 • 2 minutes to read • [Edit Online](#)

You can use Power Query to extract and transform data from external data sources. These extraction and transformations steps are represented as queries. Queries created with Power Query are expressed using the M language and executed through the M Engine.

You can easily share and reuse your queries across projects, and also across Power Query product integrations. This article covers the general mechanisms to share a query in Power Query.

Copy / Paste

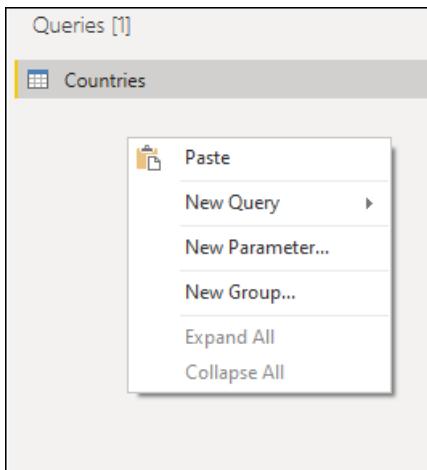
In the queries pane, right-click the query you want to copy. From the dropdown menu, select the **Copy** option. The query and its definition will be added to your clipboard.



NOTE

The copy feature is currently not available in Power Query Online instances.

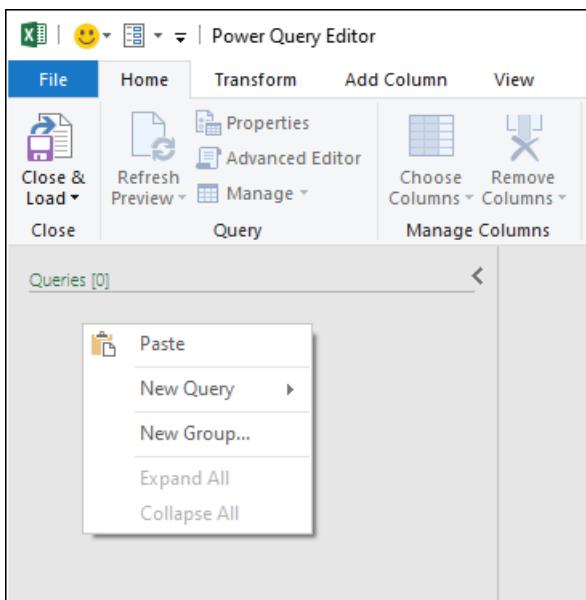
To paste the query from your clipboard, go to the queries pane and right-click on any empty space in it. From the menu, select **Paste**.



When pasting this query on an instance that already has the same query name, the pasted query will have a suffix added with the format `(#)`, where the pound sign is replaced with a number to distinguish the pasted queries.



You can also paste queries between multiple instances and product integrations. For example, you can copy the query from Power BI Desktop, as shown in the previous images, and paste it in Power Query for Excel as shown in the following image.

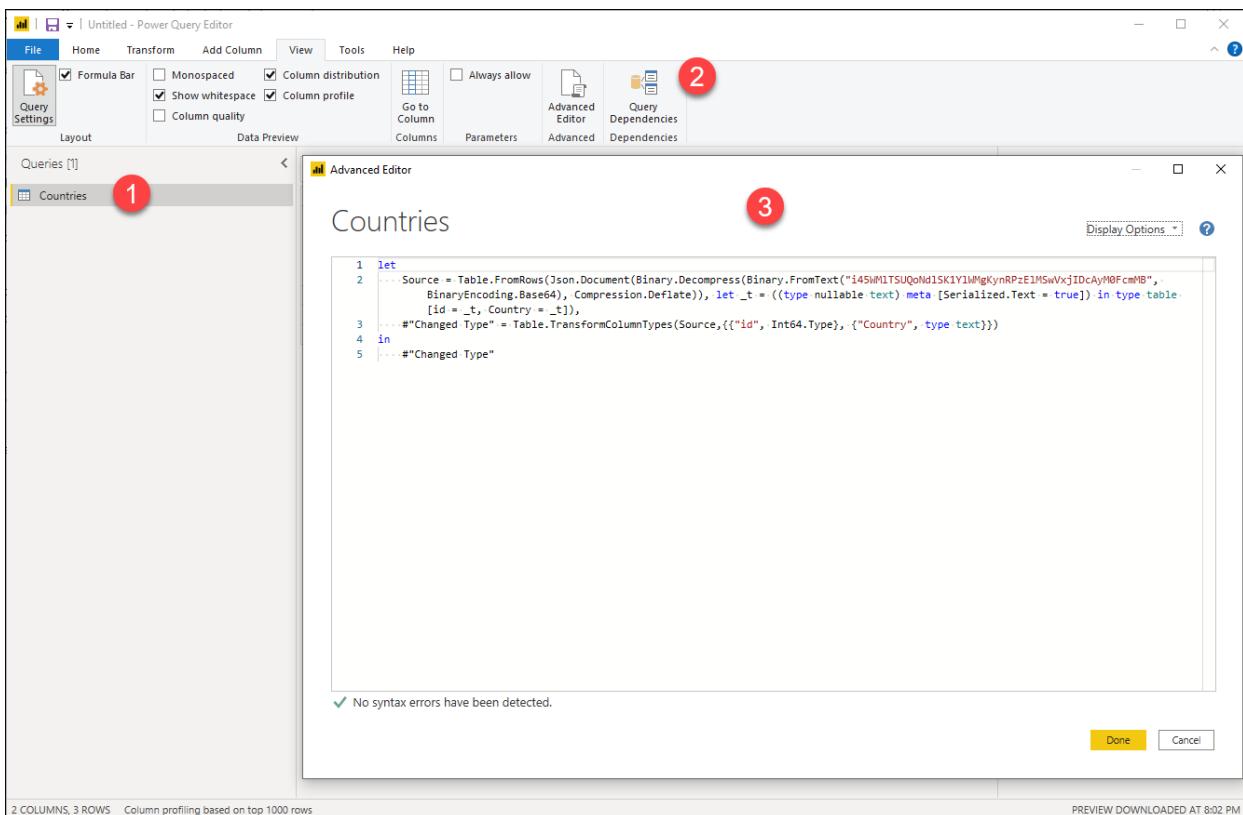


WARNING

Copying and pasting queries between product integrations doesn't guarantee that all functions and functionality found in the pasted query will work on the destination. Some functionality might only be available in the origin product integration.

Copy the M code

You can also copy the full code of your query.



1. Select the query that you want to share.
2. In the ribbon, select the **View** tab and then select **Advanced Editor**.
3. In the **Advanced Editor** window, select all the code and copy it.

With the code of your query in your clipboard, you can share this query through the means of your choice. The recipient of this code needs to create a blank query and follow the same steps as described above. But instead of copying the code, the recipient will replace the code found in their blank query with the code that you provided.

NOTE

To create a blank query, go to the **Get Data** window and select **Blank query** from the options.

Product-specific query sharing capabilities

Some Power Query product integrations might offer more ways to share queries such as but not limited to:

- **In Microsoft Excel**—Creating an Office Data Connection (.odc) to share with other users.
- **In Power BI Desktop**—Creating a Power BI Template (.pbix) to share with other users.

We recommend that you read the documentation of the product integration that you're interested in to learn more about the query sharing capabilities found in those products.

Using custom functions

1/15/2022 • 9 minutes to read • [Edit Online](#)

If you find yourself in a situation where you need to apply the same set of transformations to different queries or values, creating a Power Query custom function that can be reused as many times as you need could be beneficial. A Power Query custom function is a mapping from a set of input values to a single output value, and is created from native M functions and operators.

While you can manually create your own Power Query custom function using code as shown in [Understanding Power Query M functions](#), the Power Query user interface offers you features to speed up, simplify, and enhance the process of creating and managing a custom function. This article focuses on this experience provided only through the Power Query user interface and how to get the most out of it.

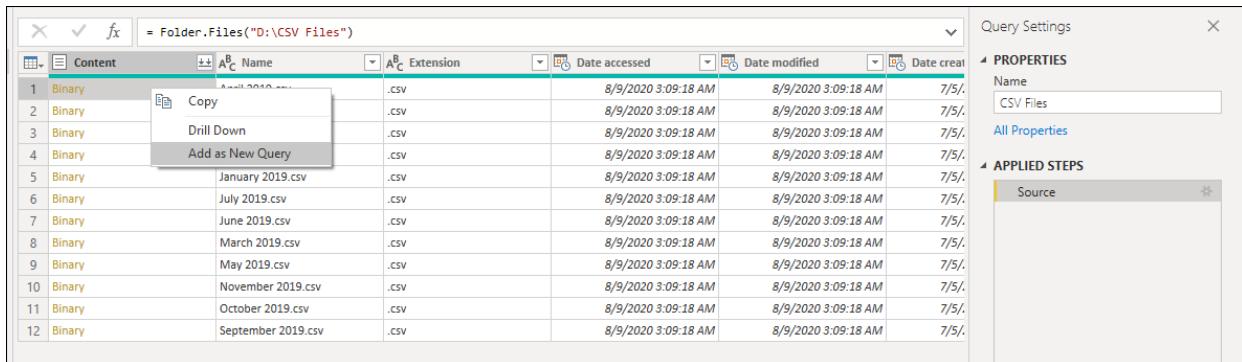
IMPORTANT

This article outlines how to create a custom function with Power Query using common transforms accessible in the Power Query user interface. It focuses on the core concepts to create custom functions, and links to additional articles in Power Query documentation for more information on specific transforms that are referenced in this article.

Create a custom function from a table reference

You can follow along with this example by downloading the sample files used in this article from the following [download link](#). For simplicity, this article will be using the Folder connector. To learn more about the Folder connector, see [Folder](#). The goal of this example is to create a custom function that can be applied to all the files in that folder before combining all of the data from all files into a single table.

Start by using the Folder connector experience to navigate to the folder where your files are located and select **Transform Data** or **Edit**. This will take you to the Power Query experience. Right-click on the **Binary** value of your choice from the **Content** field and select the **Add as New Query** option. For this example, you'll see that the selection was made for the first file from the list, which happens to be the file *April 2019.csv*.



This option will effectively create a new query with a navigation step directly to that file as a Binary, and the name of this new query will be the file path of the selected file. Rename this query to be **Sample File**.

The screenshot shows the Power BI Query Editor interface. On the left, under 'Queries [2]', there is a tree view with 'CSV Files' expanded, and 'Sample File' selected. The main area displays the query definition: `= Source{[#Folder Path"]="D:\CSV Files\CSV Files\",`. Below this, it shows a preview of the file 'April 2019.csv' with a size of 9361 bytes. The 'Query Settings' pane on the right contains 'PROPERTIES' (Name: Sample File) and 'APPLIED STEPS' (Source, Navigation).

Create a new parameter with the name **File Parameter**. Use the **Sample File** query as the **Current Value**, as shown in the following image.

The screenshot shows the 'Manage Parameters' dialog box. A new parameter named 'File Parameter' is being created. The parameters are set as follows:

- Name:** File Parameter
- Description:** (empty)
- Required:** checked
- Type:** Binary
- Suggested Values:** Binary
- Default Value:** Sample File
- Current Value:** Sample File

At the bottom, there are 'OK' and 'Cancel' buttons.

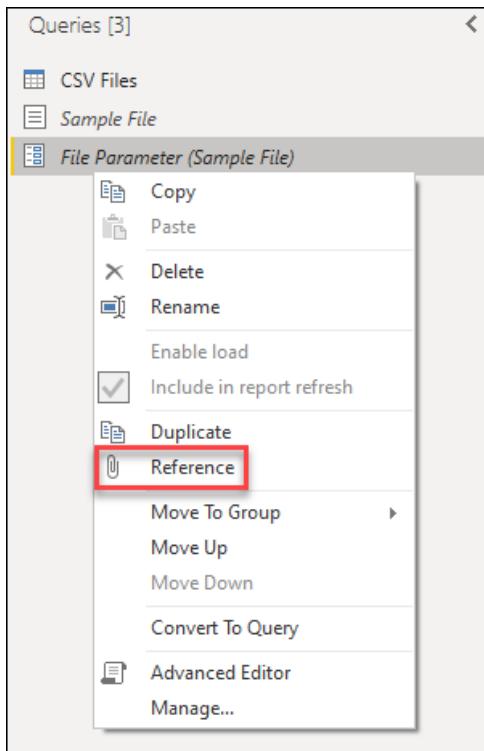
NOTE

We recommend that you read the article on [Parameters](#) to better understand how to create and manage parameters in Power Query.

Custom functions can be created using any parameters type. There's no requirement for any custom function to have a binary as a parameter.

It's possible to create a custom function without a parameter. This is commonly seen in scenarios where an input can be inferred from the environment where the function is being invoked. For example, a function that takes the environment's current date and time, and creates a specific text string from those values.

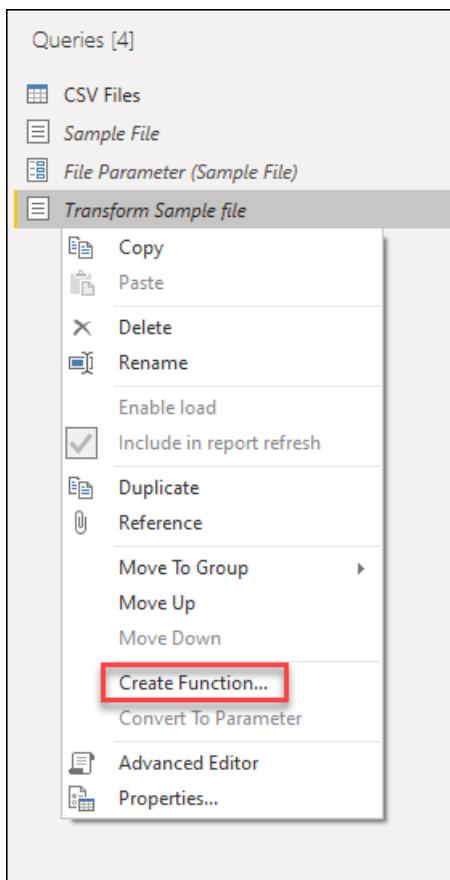
Right-click **File Parameter** from the **Queries** pane. Select the **Reference** option.



Rename the newly created query from **File Parameter (2)** to **Transform Sample file**.



Right-click this new **Transform Sample file** query and select the **Create Function** option.

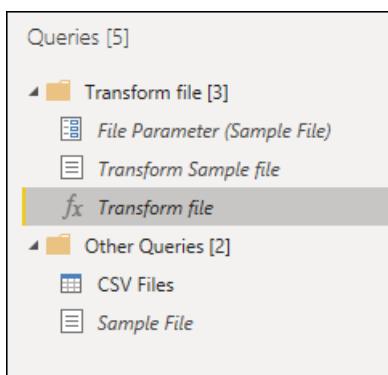


This operation will effectively create a new function that will be linked with the **Transform Sample file** query. Any changes that you make to the **Transform Sample file** query will be automatically replicated to your custom function. During the creation of this new function, use **Transform file** as the **Function name**.



After creating the function, you'll notice that a new group will be created for you with the name of your function. This new group will contain:

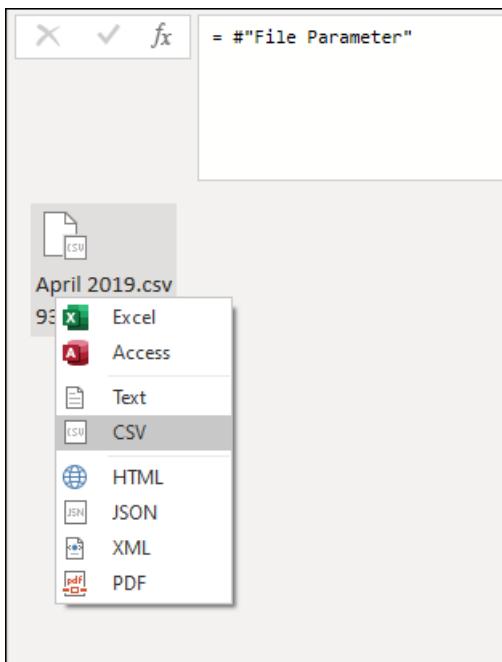
- All parameters that were referenced in your **Transform Sample file** query.
- Your **Transform Sample file** query, commonly known as the *sample query*.
- Your newly created function, in this case **Transform file**.



Applying transformations to a sample query

With your new function created, select the query with the name **Transform Sample file**. This query is now linked with the **Transform file** function, so any changes made to this query will be reflected in the function. This is what is known as the concept of a sample query linked to a function.

The first transformation that needs to happen to this query is one that will interpret the binary. You can right-click the binary from the preview pane and select the **CSV** option to interpret the binary as a CSV file.

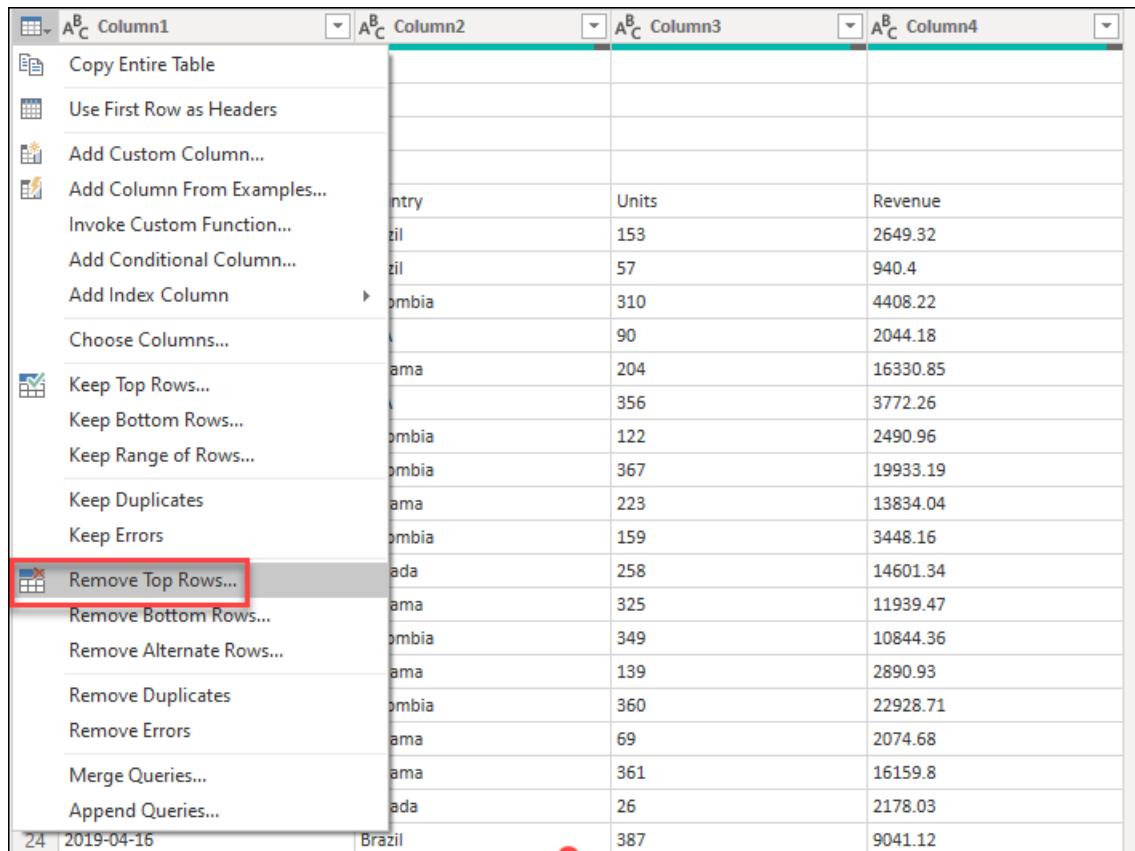


The format of all the CSV files in the folder is the same. They all have a header that spans the first top four rows. The column headers are located in row five and the data starts from row six downwards, as shown in the next image.

	Column1	Column2	Column3	Column4
1	Report generated on 01-01-2020			
2	Created by: user9284			
3	Company XYZ			
4				
5	Date	Country	Units	Revenue
6	2019-04-22	Brazil	153	2649.32
7	2019-04-14	Brazil	57	940.4
8	2019-04-26	Colombia	310	4408.22
9	2019-04-25	USA	90	2044.18
10	2019-04-23	Panama	204	16330.85
11	2019-04-07	USA	356	3772.26
12	2019-04-11	Colombia	122	2490.96
13	2019-04-27	Colombia	367	19933.19
14	2019-04-24	Panama	223	13834.04
15	2019-04-16	Colombia	159	3448.16
16	2019-04-08	Canada	258	14601.34
17	2019-04-14	Panama	325	11939.47
18	2019-04-01	Colombia	349	10844.36
19	2019-04-07	Panama	139	2890.93

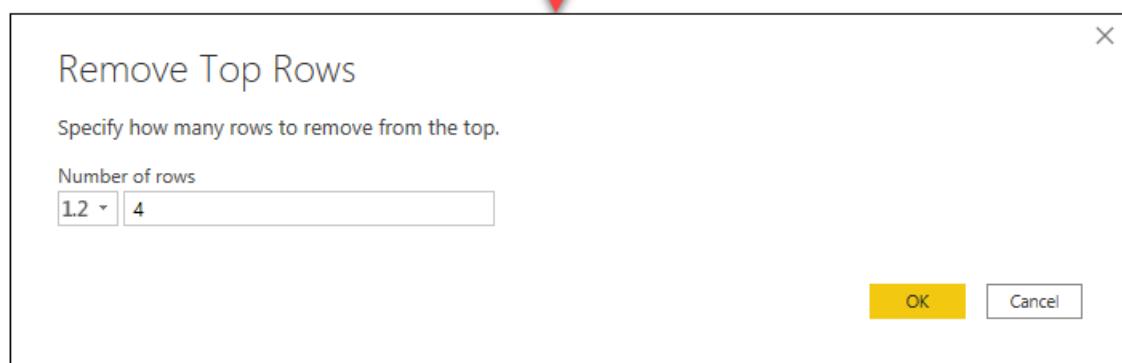
The next set of transformation steps that need to be applied to the **Transform Sample file** are:

1. **Remove the top four rows**—This action will get rid of the rows that are considered part of the header section of the file.



A screenshot of a table context menu in Power BI. The menu includes options like 'Copy Entire Table', 'Use First Row as Headers', 'Add Custom Column...', 'Add Column From Examples...', 'Invoke Custom Function...', 'Add Conditional Column...', 'Add Index Column', 'Choose Columns...', 'Keep Top Rows...', 'Keep Bottom Rows...', 'Keep Range of Rows...', 'Keep Duplicates', 'Keep Errors', 'Remove Top Rows...', 'Remove Bottom Rows...', 'Remove Alternate Rows...', 'Remove Duplicates', 'Remove Errors', 'Merge Queries...', and 'Append Queries...'. The 'Remove Top Rows...' option is highlighted with a red box.

Country	Units	Revenue
Brazil	153	2649.32
Brazil	57	940.4
Colombia	310	4408.22
Colombia	90	2044.18
Colombia	204	16330.85
Colombia	356	3772.26
Colombia	122	2490.96
Colombia	367	19933.19
Colombia	223	13834.04
Colombia	159	3448.16
Colombia	258	14601.34
Colombia	325	11939.47
Colombia	349	10844.36
Colombia	139	2890.93
Colombia	360	22928.71
Colombia	69	2074.68
Colombia	361	16159.8
Colombia	26	2178.03
Brazil	387	9041.12



NOTE

To learn more about how to remove rows or filter a table by row position, see [Filter by row position](#).

2. **Promote headers**—The headers for your final table are now in the first row of the table. You can promote them as shown in the next image.

Column1	Column2	Column3	Column4
24	2019-04-29	Country	Units
		Brazil	153
		Brazil	57
		Colombia	310
		USA	90
		Panama	204
		Colombia	356
		Colombia	122
		Colombia	367
		Panama	223
		Colombia	159
		Colombia	258
		Panama	325
		Colombia	349
		Panama	139
		Colombia	360
		Panama	69
		Panama	361
		Panama	26
		Brazil	387
		Panama	222
		Brazil	23
		Canada	398
		Canada	358

Power Query by default will automatically add a new **Changed Type** step after promoting your column headers that will automatically detect the data types for each column. Your **Transform Sample file** query will look like the next image.

NOTE

To learn more about how to promote and demote headers, see [Promote or demote column headers](#).

The screenshot shows the Power Query Editor interface with the following details:

- Queries [5]**: Shows the current queries: Transform file [3] (selected), File Parameter (Sample File), Transform Sample file (highlighted), Transform file, and Other Queries [2] (CSV Files, Sample File).
- Transform Sample file** (selected):
 - Code View** (fx): Shows the M code: `= Table.TransformColumnTypes(#"Promoted Headers",{{"Date", type date}, {"Country", type text}, {"Units", Int64.Type}, {"Revenue", type number}})`
 - Preview View**: Displays the transformed data with columns: Date, Country, Units, and Revenue.
 - Query Settings** pane on the right:
 - PROPERTIES**: Name is set to "Transform Sample file".
 - APPLIED STEPS** list:
 - Source
 - Changed Type
 - Removed Top Rows
 - Promoted Headers
 - Changed Type1 (highlighted)
- Other Queries [2]**: CSV Files, Sample File.

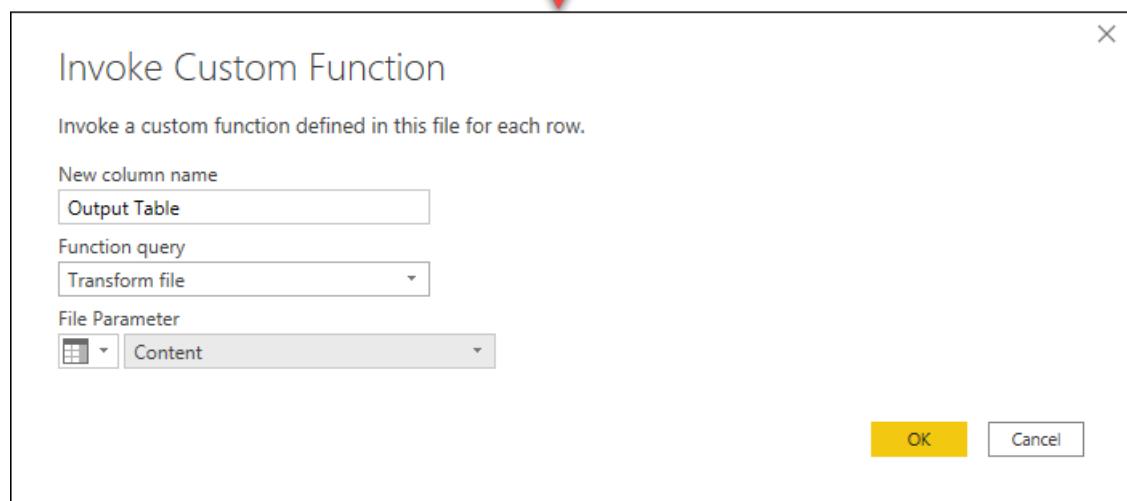
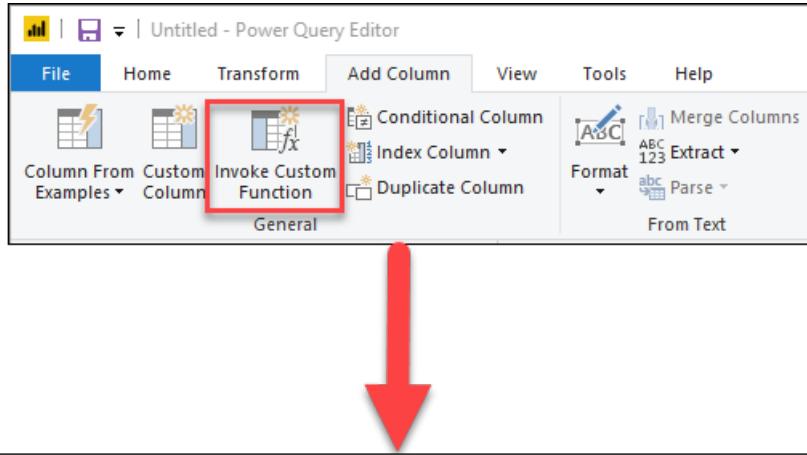
Caution

Your **Transform file** function relies on the steps performed in the **Transform Sample file** query. However, if you try to manually modify the code for the **Transform file** function, you'll be greeted with a warning that reads

The definition of the function 'Transform file' is updated whenever query 'Transform Sample file' is updated. However, updates will stop if you directly modify function 'Transform file'.

Invoke a custom function as a new column

With the custom function now created and all the transformation steps incorporated, you can go back to the original query where you have the list of files from the folder. Inside the **Add Column** tab in the ribbon, select **Invoke Custom Function** from the **General** group. Inside the **Invoke Custom Function** window, enter **Output Table** as the **New column name**. Select the name of your function, **Transform file**, from the **Function query** dropdown. After selecting the function from the dropdown menu, the parameter for the function will be displayed and you can select which column from the table to use as the argument for this function. Select the **Content** column as the value / argument to be passed for the **File Parameter**.



After you select **OK**, a new column with the name **Output Table** will be created. This column has **Table** values in its cells, as shown in the next image. For simplicity, remove all columns from this table except **Name** and **Output Table**.

Queries [5]

- Transform file [3]
 - File Parameter (Sample File)
 - Transform Sample file
 - fx Transform file**
- Other Queries [2]
 - CSV Files
 - Sample File

fx Transform file

= Table.SelectColumns(#"Invoked Custom")

Name	Output Table
1 April 2019.csv	Table
2 August 2019.csv	Table
3 December 2019.csv	Table
4 February 2019.csv	Table
5 January 2019.csv	Table
6 July 2019.csv	Table
7 June 2019.csv	Table
8 March 2019.csv	Table
9 May 2019.csv	Table
10 November 2019.csv	Table
11 October 2019.csv	Table
12 September 2019.csv	Table

Query Settings

PROPERTIES

Name: CSV Files

APPLIED STEPS

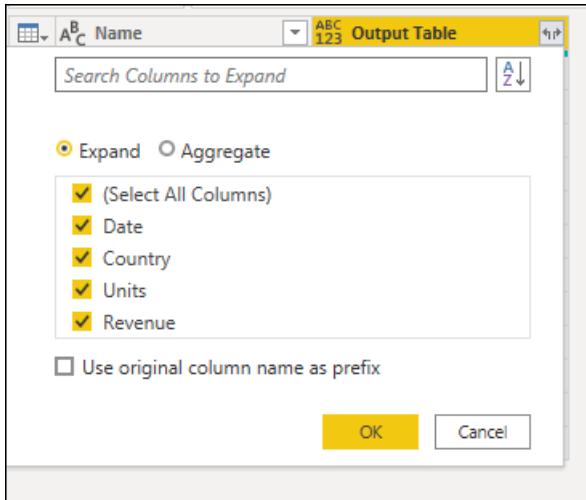
- Source
- Invoked Custom Function
- Removed Other Columns

2 COLUMNS, 12 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 4:18 AM

NOTE

To learn more about how to choose or remove columns from a table, see [Choose or remove columns](#).

Your function was applied to every single row from the table using the values from the **Content** column as the argument for your function. Now that the data has been transformed into the shape that you're looking for, you can expand the **Output Table** column, as shown in the image below, without using any prefix for the expanded columns.



You can verify that you have data from all files in the folder by checking the values in the **Name** or **Date** column. For this case, you can check the values from the **Date** column, as each file only contains data for a single month from a given year. If you see more than one, it means that you've successfully combined data from multiple files into a single table.

The screenshot shows the Power Query Editor interface. On the left, the 'Queries [5]' pane lists 'Transform file [3]', 'File Parameter (Sample File)', 'Transform Sample file', 'fx Transform file', and 'Other Queries [2]'. Under 'Other Queries [2]', 'CSV Files' is selected, and its properties are shown in the 'PROPERTIES' pane: Name is 'CSV Files' and 'All Properties' is expanded. The main area displays a table with four columns: 'Date' (sorted by ascending date), 'Country', 'Units', and 'Revenue'. A date filter dialog is open over the table, showing a list of dates from 10/21/2019 to 11/6/2019, with all dates checked. The 'OK' button is highlighted. The 'APPLIED STEPS' pane on the right shows the steps: 'Source', 'Invoked Custom Function', and 'Removed Other Columns' (which is highlighted).

NOTE

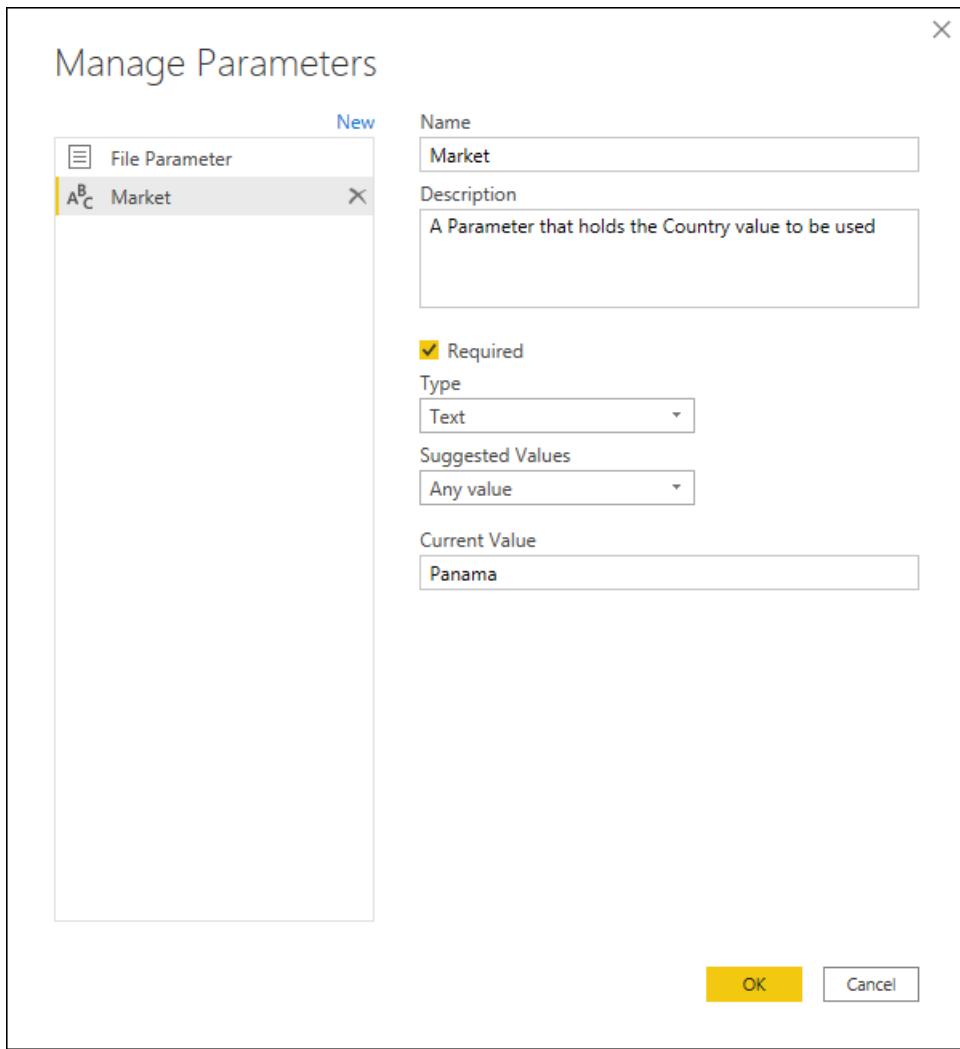
What you've read so far is fundamentally the same process that happens during the **Combine files** experience, but done manually.

We recommend that you also read the article on [Combine files overview](#) and [Combine CSV files](#) to further understand how the combine files experience works in Power Query and the role that custom functions play.

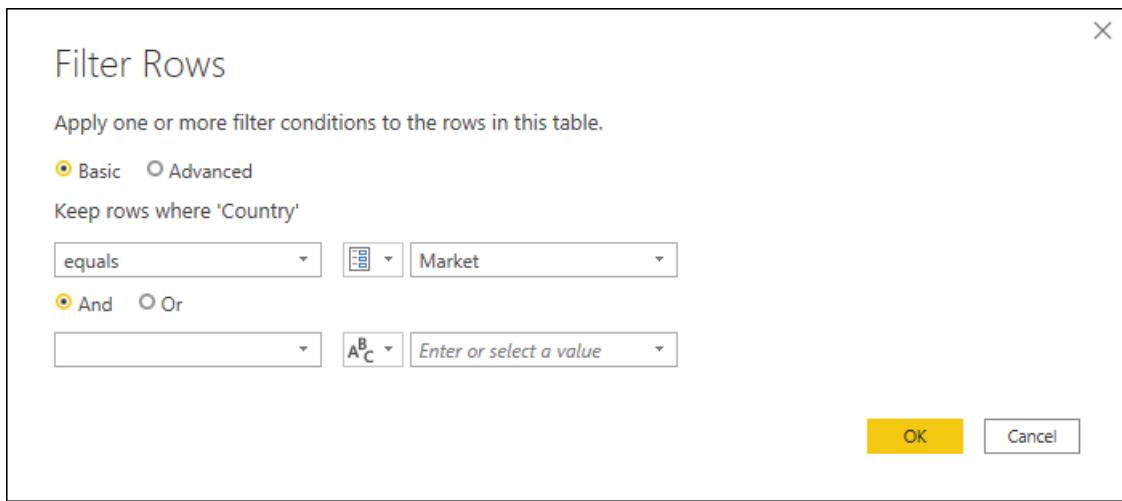
Add new parameter to existing custom function

Imagine that there's a new requirement on top of what you've built. The new requirement requires that before you combine the files, you filter the data inside them to only get the rows where the Country is equals to *Panama*.

To make this requirement happen, create a new parameter called **Market** with the text data type. For the **Current Value**, enter the value **Panama**.



With this new parameter, select the **Transform Sample file** query and filter the **Country** field using the value from the **Market** parameter.



NOTE

To learn more about how to filter columns by values, see [Filter values](#).

Applying this new step to your query will automatically update the **Transform file** function, which will now require two parameters based on the two parameters that your **Transform Sample file** uses.

The screenshot shows the Power Query Editor interface. On the left, the 'Queries [6]' pane lists several queries: 'Transform file [3]' (containing 'File Parameter (Sample File)', 'Transform Sample file', and 'fx Transform file'), 'Other Queries [3]' (containing 'CSV Files', 'Sample File', and 'Market (Panama)'). The 'fx Transform file' query is selected. In the center, the formula bar shows a function definition: `= (#"File Parameter" as binary, Market as text) => let`. Below it, the 'Enter Parameters' section has two fields: 'File Parameter' (empty) and 'Market' (with placeholder 'Example: abc'). At the bottom, the formula is displayed again: `function (File Parameter as binary, Market as text) as any`. On the right, the 'Query Settings' pane shows the 'PROPERTIES' section with 'Name' set to 'Transform file' and the 'APPLIED STEPS' section showing 'Source'.

But the **CSV files** query has a warning sign next to it. Now that your function has been updated, it requires two parameters. So the step where you invoke the function results in error values, since only one of the arguments was passed to the **Transform file** function during the **Invoked Custom Function** step.

The screenshot shows the Power Query Editor. The 'Queries [6]' pane is similar to the previous one, with 'fx Transform file' selected. The 'fx CSV Files' query has a yellow warning bar above it: 'Expression.Error: 1 arguments were passed to a function which expects 2.' Below the bar, 'Details:' show 'Pattern=' and 'Arguments=[List]'. The 'Query Settings' pane on the right shows 'Name' set to 'CSV Files' and the 'APPLIED STEPS' section listing 'Source', 'Invoked Custom Function', 'Removed Other Columns', and 'Expanded Output Table'.

To fix the errors, double-click **Invoked Custom Function** in the Applied Steps to open the **Invoke Custom Function** window. In the **Market** parameter, manually enter the value **Panama**.

The screenshot shows the 'Invoke Custom Function' dialog box. It contains the following fields:

- New column name:** Output Table
- Function query:** Transform file
- File Parameter:** Content
- Market:** Panama

At the bottom right are 'OK' and 'Cancel' buttons.

You can now check your query to validate that only rows where **Country** is equal to **Panama** show up in the final result set of the **CSV Files** query.

5 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

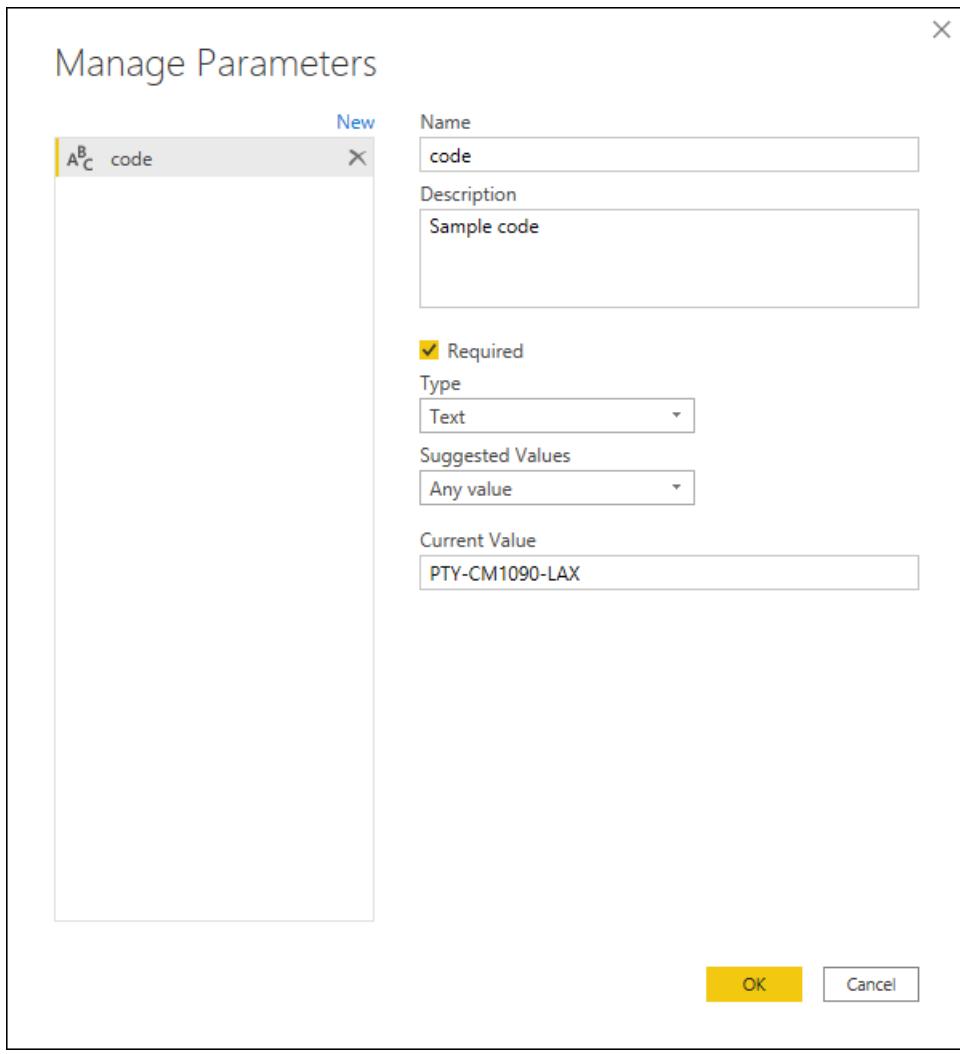
Create a custom function from a reusable piece of logic

If you have multiple queries or values that require the same set of transformations, you could create a custom function that acts as a reusable piece of logic. Later, this custom function can be invoked against the queries or values of your choice. This custom function could save you time and help you in managing your set of transformations in a central location, which you can modify at any moment.

For example, imagine a query that has several codes as a text string and you want to create a function that will decode those values.

	code
5 distinct, 5 unique	
1	PTY-CM1090-LAX
2	LAX-CM701-PTY
3	PTY-CM4441-MIA
4	MIA-UA1257-LAX
5	LAX-XY2842-MIA

You start by having a parameter that has a value that serves as an example. For this case, it will be the value PTY-CM1090-LAX.



From that parameter, you create a new query where you apply the transformations that you need. For this case, you want to split the code *PTY-CM1090-LAX* into multiple components:

- **Origin** = PTY
- **Destination** = LAX
- **Airline** = CM
- **FlightID** = 1090

Origin	Destination	Airline	FlightID
1 distinct, 1 unique 1 PTY	1 distinct, 1 unique 1 LAX	1 distinct, 1 unique 1 CM	1 distinct, 1 unique 1 1090

The M code for that set of transformations is shown below.

```

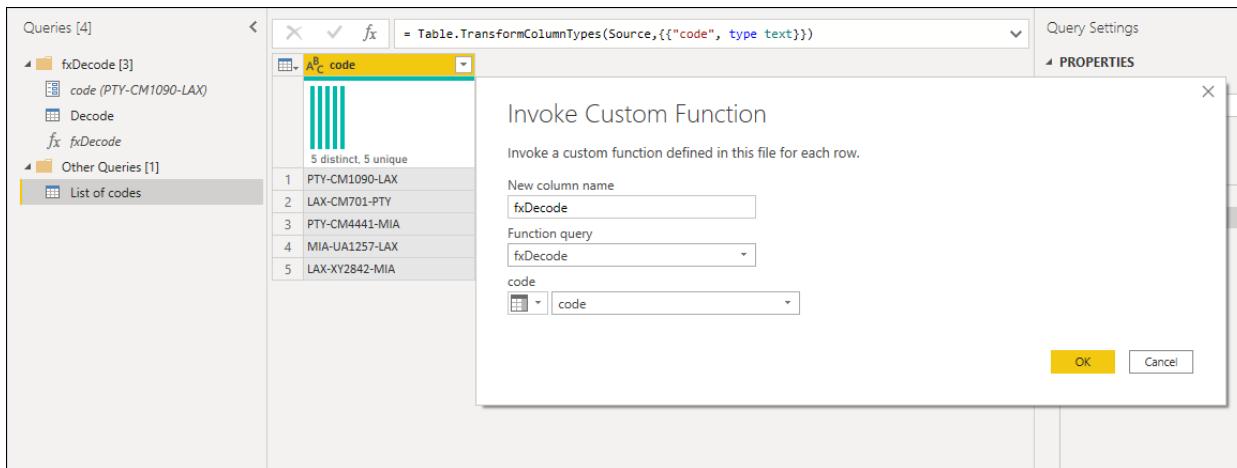
let
    Source = code,
    SplitValues = Text.Split( Source, "-" ),
    CreateRow = [Origin= SplitValues{0}, Destination= SplitValues{2}, Airline=Text.Start( SplitValues{1},2),
    FlightID= Text.End( SplitValues{1}, Text.Length( SplitValues{1} ) - 2 ) ],
    RowToTable = Table.FromRecords( { CreateRow } ),
    #"Changed Type" = Table.TransformColumnTypes(RowToTable,{{"Origin", type text}, {"Destination", type text},
    {"Airline", type text}, {"FlightID", type text}})
in
    #"Changed Type"

```

NOTE

To learn more about the Power Query M formula language, see [Power Query M formula language](#)

You can then transform that query into a function by doing a right-click on the query and selecting **Create Function**. Finally, you can invoke your custom function into any of your queries or values, as shown in the next image.



After a few more transformations, you can see that you've reached your desired output and leveraged the logic for such a transformation from a custom function.

code	Origin	Destination	Airline	FlightID
PTY-CM1090-LAX	PTY	LAX	CM	1090
LAX-CM701-PTY	LAX	PTY	CM	701
PTY-CM4441-MIA	PTY	MIA	CM	4441
MIA-UA1257-LAX	MIA	LAX	UA	1257
LAX-XY2842-MIA	LAX	MIA	XY	2842

Promote or demote column headers

1/15/2022 • 2 minutes to read • [Edit Online](#)

When creating a new query from unstructured data sources such as text files, Power Query analyzes the contents of the file. If Power Query identifies a different pattern for the first row, it will try to promote the first row of data to be the column headings for your table. However, Power Query might not identify the pattern correctly 100 percent of the time, so this article explains how you can manually promote or demote column headers from rows.

To promote rows to column headers

In the following example, Power Query wasn't able to determine the column headers for the table because the table contains a set of header rows for the first three rows. The actual column headers for the table are contained in row 5.

	A ^B Column1	A ^B Column2	A ^B Column3	A ^B Column4
1	SALES REPORT			
2	Created on: 30-Jun-2020			
3	Created by user: PTY			
4				
5	Date	Country	Total Units	Total Revenue
6	6/8/2020	Panama	556	1917.35
7	6/4/2020	USA	926	4208.45
8	6/12/2020	Canada	157	72.25
9	6/22/2020	Panama	334	933.44
10	6/3/2020	USA	434	1142.87
11	6/8/2020	Canada	407	5851.3
12	6/8/2020	Mexico	806	4614.86

Table with the columns (Column1, Column2, Column3 and column 4) all set to the Text data type, with four rows containing a header at the top, a column header in row 5, and seven data rows at the bottom.

Before you can promote the headers, you need to remove the first four rows of the table. To make that happen, select the table menu in the upper-left corner of the preview window, and then select **Remove top rows**.

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	A ^B _C Column4
<input type="checkbox"/>	Use first row as headers			
<input type="checkbox"/>	Add custom column			
<input type="checkbox"/>	Add conditional column			
<input type="checkbox"/>	Index column	>		
<input type="checkbox"/>	Choose columns			
<input checked="" type="checkbox"/>	Keep top rows	Panama	Country	Total Units
<input checked="" type="checkbox"/>	Keep bottom rows	USA	926	1917.35
<input checked="" type="checkbox"/>	Keep range of rows	Canada	157	4208.45
<input checked="" type="checkbox"/>	Keep errors	Panama	334	72.25
<input checked="" type="checkbox"/>	Remove top rows	USA	434	933.44
<input checked="" type="checkbox"/>	Remove bottom rows	Canada	407	1142.87
<input checked="" type="checkbox"/>	Remove alternate rows	Mexico	806	5851.3
<input checked="" type="checkbox"/>	Remove duplicates			
<input checked="" type="checkbox"/>	Remove errors			
<input type="checkbox"/>	Merge queries			
<input type="checkbox"/>	Append queries			

In the **Remove top rows** window, enter 4 in the **Number of rows** box.

Remove top rows

Specify how many rows to remove from the top.

Number of rows

OK
Cancel

NOTE

To learn more about **Remove top rows** and other table operations, go to [Filter by row position](#).

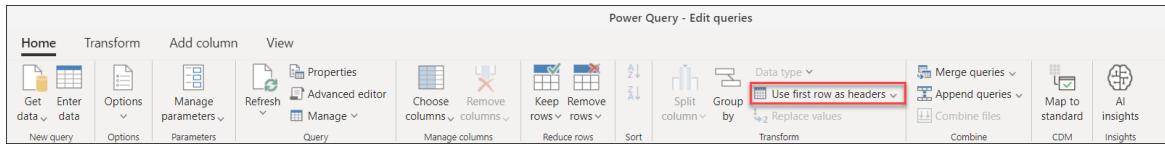
The result of that operation will leave the headers as the first row of your table.

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	A ^B _C Column4
1	Date	Country	Total Units	Total Revenue
2	6/8/2020	Panama	556	1917.35
3	6/4/2020	USA	926	4208.45
4	6/12/2020	Canada	157	72.25
5	6/22/2020	Panama	334	933.44
6	6/3/2020	USA	434	1142.87
7	6/8/2020	Canada	407	5851.3
8	6/8/2020	Mexico	806	4614.86

Locations of the promote headers operation

From here, you have a number of places where you can select the promote headers operation:

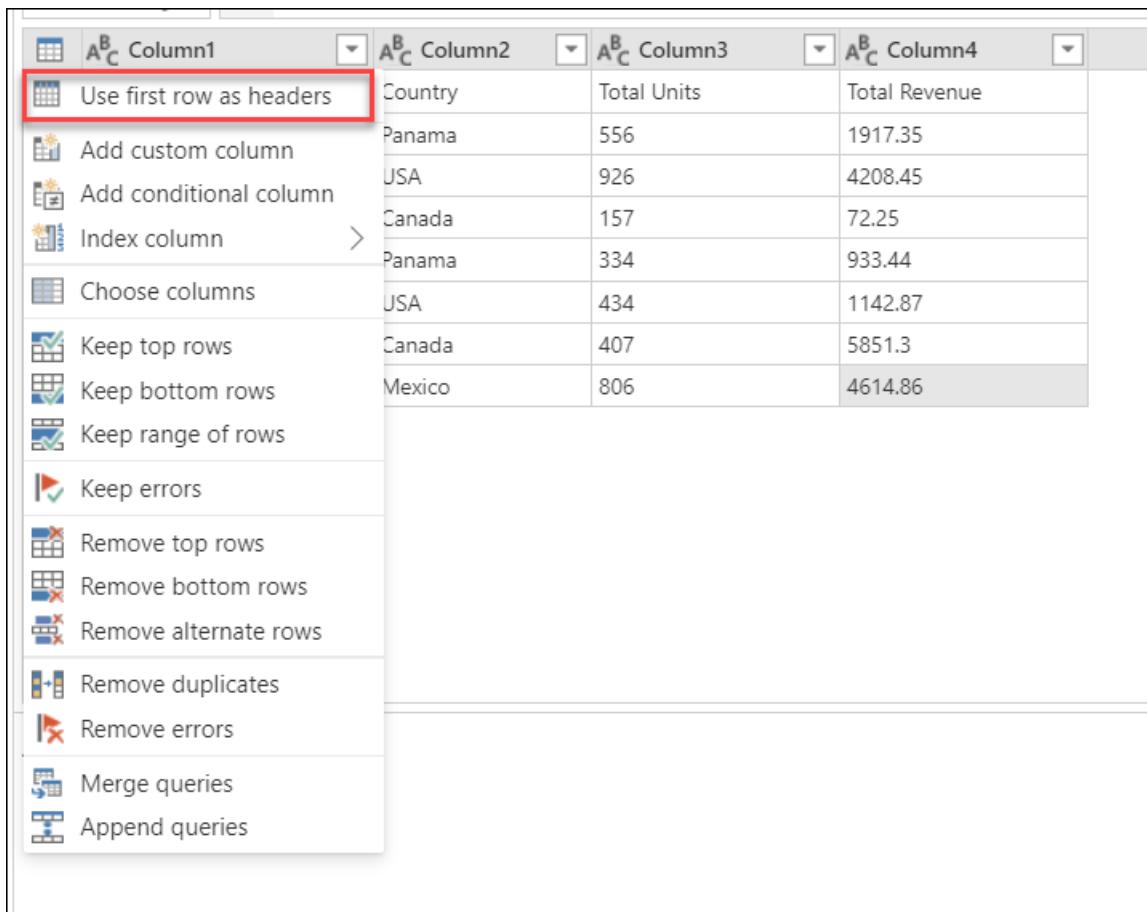
- On the Home tab, in the Transform group.



- On the Transform tab, in the Table group.



- On the table menu.



After you do the promote headers operation, your table will look like the following image.

	Date	Country	Total Units	Total Revenue
1	6/8/2020	Panama	556	1917.35
2	6/4/2020	USA	926	4208.45
3	6/12/2020	Canada	157	72.25
4	6/22/2020	Panama	334	933.44
5	6/3/2020	USA	434	1142.87
6	6/8/2020	Canada	407	5851.3
7	6/8/2020	Mexico	806	4614.86

Table with Date, Country, Total Units, and Total Revenue column headers, and seven rows of data. The Date column header has a Date data type, the Country column header has a Text data type, the Total Units column header has a Whole number data type, and the Total Revenue column header has a Decimal number data type.

NOTE

Table column names must be unique. If the row you want to promote to a header row contains multiple instances of the same text string, Power Query will disambiguate the column headings by adding a numeric suffix preceded by a dot to every text string that isn't unique.

To demote column headers to rows

In the following example, the column headers are incorrect: they're actually part of the table's data. You need to demote the headers to be part of the rows of the table.

	6/8/2020	A ^B _C Panama	1 ² ₃ 556	1.2 1917.35
1	6/4/2020	USA	926	4208.45
2	6/12/2020	Canada	157	72.25
3	6/22/2020	Panama	334	933.44
4	6/3/2020	USA	434	1142.87
5	6/8/2020	Canada	407	5851.3
6	6/8/2020	Mexico	806	4614.86

Locations of the demote headers operation

You have a number of places where you can select the demote headers operation:

- On the Home tab, in the Transform group.

The screenshot shows the Power Query interface with the 'Transform' tab selected. In the ribbon, under the 'Transform' tab, there is a 'Group' button followed by a dropdown menu labeled 'Use first row as headers'. Below the ribbon, in the main area, there is a table with six rows of data. At the top of the table, the columns are labeled with the values from the first row: '6/8/2020', 'A^B_C Panama', '1²₃ 556', and '1.2 1917.35'. A red box highlights the 'Use headers as first row' checkbox located in the 'Group' dropdown menu.

- On the Transform tab, in the Table group.

The screenshot shows the Power Query interface with the 'Transform' tab selected. In the ribbon, under the 'Transform' tab, there is a 'Group' button followed by a dropdown menu labeled 'Use first row as headers'. Below the ribbon, in the main area, there is a table with six rows of data. A red box highlights the 'Use headers as first row' checkbox located in the 'Group' dropdown menu.

After you do this operation, your table will look like the following image.

	Column1	Column2	Column3	Column4
1	6/8/2020	Panama	556	1917.35
2	6/4/2020	USA	926	4208.45
3	6/12/2020	Canada	157	72.25
4	6/22/2020	Panama	334	933.44
5	6/3/2020	USA	434	1142.87
6	6/8/2020	Canada	407	5851.3
7	6/8/2020	Mexico	806	4614.86

As a last step, select each column and type a new name for it. The end result will resemble the following image.

The screenshot shows the Power Query - Edit queries window. The ribbon at the top has tabs for Home, Transform, Add column, and View. The Home tab is selected. The ribbon contains various icons for data operations like Get data, Enter data, Options, Manage parameters, Refresh, Advanced editor, Properties, Choose columns, Remove columns, Keep rows, Remove rows, Sort, Split column, Group by, Replace values, Merge queries, Append queries, Combine files, Map to standard, CDM, and Insights. The main area displays a table titled "Table.RenameColumns(#"Changed column type", {{"Column1", "Date"}, {"Column2", "Country"}, {"Column3", "Total Units"}, {"Column4", "Total Revenue"}})". The table has four columns: Date, Country, Total Units, and Total Revenue. The data rows are identical to the initial table. To the right of the table, the "Query settings" pane is open, showing the "Name" field set to "Query". Under "Applied steps", there is a list: Source, Changed Type, Demoted headers, Changed column t..., and Renamed columns. The M code for the query is visible in the formula bar at the top:

```
Table.RenameColumns(#"Changed column type", {{"Column1", "Date"}, {"Column2", "Country"}, {"Column3", "Total Units"}, {"Column4", "Total Revenue"}})
```

Final table after renaming column headers to Date, Country, Total Units, and Total Revenue, with Renamed columns emphasized in the Query settings pane and the M code shown in the formula bar.

See also

[Filter by row position](#)

Filter a table by row position

1/15/2022 • 6 minutes to read • [Edit Online](#)

Power Query has multiple options to filter a table based on the positions of its rows, either by keeping or removing those rows. This article covers all the available methods.

Keep rows

The keep rows set of functions will select a set of rows from the table and remove any other rows that don't meet the criteria.

There are two places where you can find the **Keep rows** buttons:

- On the Home tab, in the Reduce Rows group.

A screenshot of the Microsoft Power Query ribbon interface. The 'Home' tab is selected. In the 'Reduce Rows' group, there are three buttons: 'Keep top rows', 'Keep bottom rows', and 'Keep range of rows'. A red box highlights these three buttons. Below them are other options: 'Keep duplicates' and 'Keep errors'.

Period	Country	Units
Present	USA	51
P-1	USA	71
P-2	USA	89
P-3	USA	39

- On the table menu.

A screenshot of the Power Query table context menu. The 'Keep top rows', 'Keep bottom rows', and 'Keep range of rows' options are highlighted with a red box. Other options in the menu include 'Use first row as headers', 'Add custom column', 'Add conditional column', 'Index column', 'Choose columns', 'Keep errors', 'Remove top rows', 'Remove bottom rows', 'Remove alternate rows', 'Remove duplicates', 'Remove errors', 'Merge queries', and 'Append queries'.

NOTE

In the data preview section in the middle of the Power Query window, you can see the position of your rows on the left side of the table. Each row position is represented by a number. The top row starts with position 1.

Keep top rows

Imagine the following table that comes out of a system with a fixed layout.

	A ^B C Period	A ^B C Country	A ^B C Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88
8			
9	Lorem ipsum dolor sit amet, consectetur...		
10			
11	Vestibulum morbi blandit cursus risu...		
12			
13	Ut porttitor leo a diam sollicitudin te...		
14			
15	Egestas diam in arcu cursus euismod...		
16			
17	Libero id faucibus nisl tincidunt eget...		

This report always contains seven rows of data, and below the data it has a section for comments with an unknown number of rows. In this example, you only want to keep the first seven rows of data. To do that, select **Keep top rows** from the table menu. In the **Keep top rows** dialog box, enter 7 in the **Number of rows** box.

Keep top rows

Specify how many rows to keep.

Number of rows

7

OK

Cancel

The result of that change will give you the output table you're looking for. After you set the data types for your columns, your table will look like the following image.

	A ^B _C Period	A ^B _C Country	1 ² ₃ Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Keep bottom rows

Imagine the following table that comes out of a system with a fixed layout.

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3
1	The BEST Company		
2			
3	Supert Important Report		
4	Created on:		Jun-20
5			
6	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempo...		
7			
8	Vestibulum morbi blandit cursus risus at ultrices mi tempus imperdiet. A erat na...		
9			
10	Ut porttitor leo a diam sollicitudin tempor id eu. At urna condimentum mattis p...		
11			
12	Egestas diam in arcu cursus euismod quis viverra nibh. Enim sit amet venenatis ...		
13			
14	Libero id faucibus nisl tincidunt eget nullam non. Dui id ornare arcu odio ut. Lec...		
15			
16	Period	Country	Units
17	Present	USA	51
18	P-1	USA	71
19	P-2	USA	89
20	P-3	USA	39
21	P-4	USA	33
22	P-5	USA	22
23	P-6	USA	88

Initial sample table with Column1, Column2, and Column3 as the column headers, all set to the Text data type, and the bottom seven rows containing data, and above that a column headers row and an unknown number of comments.

This report always contains seven rows of data at the end of the report page. Above the data, the report has a section for comments with an unknown number of rows. In this example, you only want to keep those last seven rows of data and the header row.

To do that, select **Keep bottom rows** from the table menu. In the **Keep bottom rows** dialog box, enter **8** in the **Number of rows** box.

Keep bottom rows

Specify how many rows to keep.

Number of rows

8

OK

Cancel

The result of that operation will give you eight rows, but now your header row is part of the table.

	A ^B Column1	A ^B Column2	A ^B Column3
1	Period	Country	Units
2	Present	USA	51
3	P-1	USA	71
4	P-2	USA	89
5	P-3	USA	39
6	P-4	USA	33
7	P-5	USA	22
8	P-6	USA	88

You need to promote the column headers from the first row of your table. To do this, select **Use first row as headers** from the table menu. After you define data types for your columns, you'll create a table that looks like the following image.

	A ^B Period	A ^B Country	1 ² 3 Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Final sample table for Keep bottom rows after promoting the first row to column headers and retaining seven rows of data, and then setting the Units to the Number data type.

More information: [Promote or demote column headers](#)

Keep a range of rows

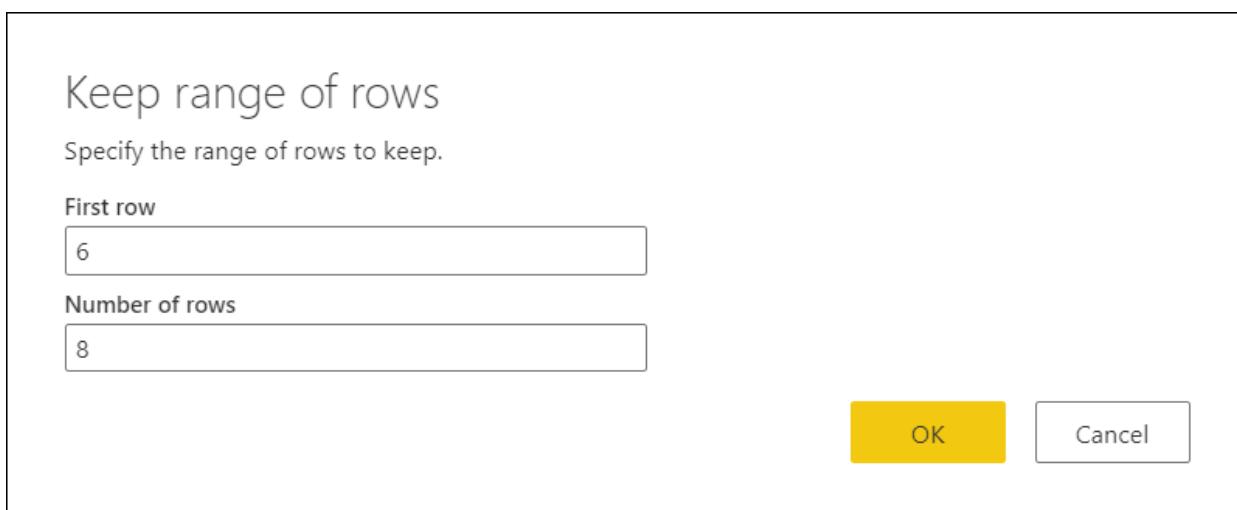
Imagine the following table that comes out of a system with a fixed layout.

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3
1	The BEST Company		
2			
3	Supert Important Report		
4	Created on:		Jun-20
5			
6	Period	Country	Units
7	Present	USA	51
8	P-1	USA	71
9	P-2	USA	89
10	P-3	USA	39
11	P-4	USA	33
12	P-5	USA	22
13	P-6	USA	88
14			
15	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempo...		
16			
17	Vestibulum morbi blandit cursus risus at ultrices mi tempus imperdiet. A erat na...		
18			
19	Ut porttitor leo a diam sollicitudin tempor id eu. At urna condimentum mattis p...		
20			
21	Egestas diam in arcu cursus euismod quis viverra nibh. Enim sit amet venenatis ...		
22			
23	Libero id faucibus nisl tincidunt eget nullam non. Dui id ornare arcu odio ut. Lec...		

Initial sample table with the columns (Column1, Column2, and Column3) all set to the Text data type, and containing the column headers and seven rows of data in the middle of the table.

This report always contains five rows for the header, one row of column headers below the header, seven rows of data below the column headers, and then an unknown number of rows for its comments section. In this example, you want to get the eight rows after the header section of the report, and only those eight rows.

To do that, select **Keep range of rows** from the table menu. In the **Keep range of rows** dialog box, enter 6 in the **First row** box and 8 in the **Number of rows** box.



Similar to the previous example for keeping bottom rows, the result of this operation gives you eight rows with your column headers as part of the table. Any rows above the **First row** that you defined (row 6) are removed.

	Column1	Column2	Column3
1	Period	Country	Units
2	Present	USA	51
3	P-1	USA	71
4	P-2	USA	89
5	P-3	USA	39
6	P-4	USA	33
7	P-5	USA	22
8	P-6	USA	88

You can perform the same operation as described in [Keep bottom rows](#) to promote the column headers from the first row of your table. After you set data types for your columns, your table will look like the following image.

	Period	Country	Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Final sample table for Keep range of rows after promoting first row to column headers, setting the Units column to the Number data type, and keeping seven rows of data.

Remove rows

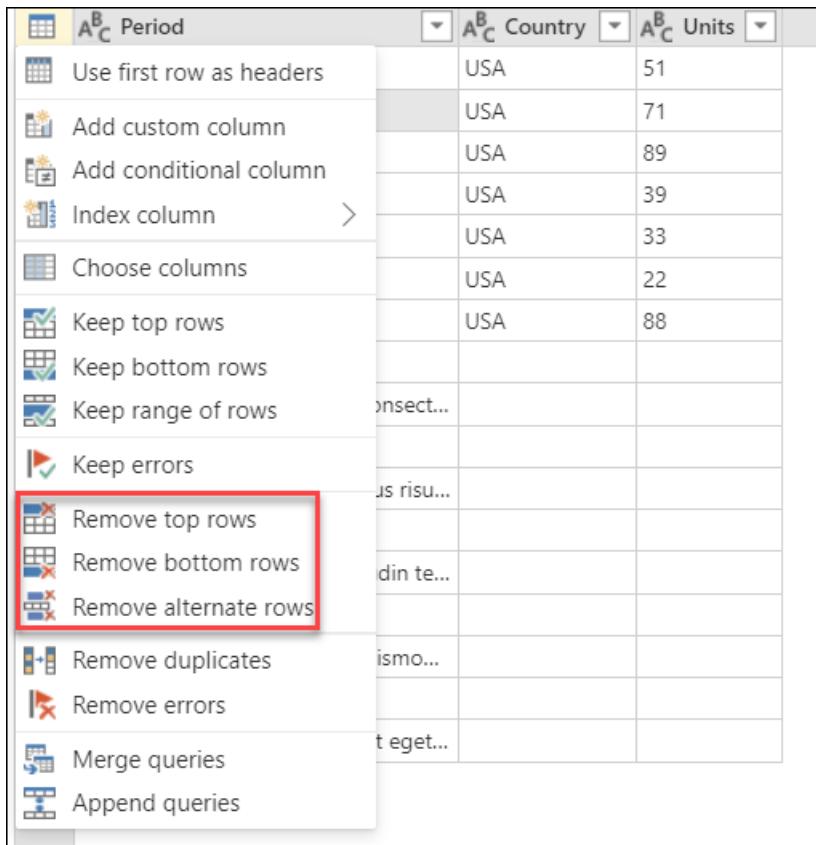
This set of functions will select a set of rows from the table, remove them, and keep the rest of the rows in the table.

There are two places where you can find the **Remove rows** buttons:

- On the Home tab, in the Reduce Rows group.

The screenshot shows the Power BI desktop interface with the Home tab selected. In the ribbon, under the Home tab, there is a 'Reduce Rows' group containing several icons: 'Keep rows', 'Remove rows', and others. Below the ribbon, the 'Queries' pane is open, showing a query named 'Keep Top Rows'. To the right of the pane, a context menu is displayed, listing options for removing rows: 'Remove top rows', 'Remove bottom rows', and 'Remove alternate rows'. These three options are highlighted with a red box. Other options listed in the menu include 'Remove duplicates', 'Remove blank rows', and 'Remove errors'.

- On the table menu.



Remove top rows

Imagine the following table that comes out of a system with a fixed layout.

	Column1	Column2	Column3
1	The BEST Company		
2			
3	Supert Important Rep...		
4	Created on:		Jun-20
5			
6	Period	Country	Units
7	Present	USA	51
8	P-1	USA	71
9	P-2	USA	89
10	P-3	USA	39
11	P-4	USA	33
12	P-5	USA	22
13	P-6	USA	88

Initial sample table for Remove top rows with the columns (Column1, Column2, and Column3) all set to the Text data type, a header at the top and a column header row and seven data rows at the bottom.

This report always contains a fixed header from row 1 to row 5 of the table. In this example, you want to remove these first five rows and keep the rest of the data.

To do that, select **Remove top rows** from the table menu. In the **Remove top rows** dialog box, enter 5 in the **Number of rows** box.

Remove top rows

Specify how many rows to remove from the top.

Number of rows

5

OK

Cancel

In the same way as the previous examples for "Keep bottom rows" and "Keep a range of rows," the result of this operation gives you eight rows with your column headers as part of the table.

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3
1	Period	Country	Units
2	Present	USA	51
3	P-1	USA	71
4	P-2	USA	89
5	P-3	USA	39
6	P-4	USA	33
7	P-5	USA	22
8	P-6	USA	88

You can perform the same operation as described in previous examples to promote the column headers from the first row of your table. After you set data types for your columns, your table will look like the following image.

	A ^B _C Period	A ^B _C Country	1 ² ₃ Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Final sample table for Remove top rows after promoting first row to column headers and setting the Units column to the Number data type, and retaining seven rows of data.

Remove bottom rows

Imagine the following table that comes out of a system with a fixed layout.

	A ^B C Period	A ^B C Country	A ^B C Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88
8			
9	The BEST Company		
10			
11	Supert Important Report		
12	Created on:		Jun-20

Initial sample table for Remove bottom rows, with the header columns all set to the Text data type, seven rows of data, then a footer of fixed length at the bottom.

This report always contains a fixed section or footer that occupies the last five rows of the table. In this example, you want to remove those last five rows and keep the rest of the data.

To do that, select **Remove bottom rows** from the table menu. In the **Remove top rows** dialog box, enter 5 in the **Number of rows** box.



The result of that change will give you the output table that you're looking for. After you set data types for your columns, your table will look like the following image.

	A ^B C Period	A ^B C Country	1 ² 3 Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Remove alternate rows

Imagine the following table that comes out of a system with a dynamic layout.

	A ^B _C Period	A ^B _C Country	A ^B _C Units
1	Present	USA	51
2	Not the greatest results. We...		
3	P-1	USA	71
4	Still on track!		
5	P-2	USA	89
6	Amazing Results!		
7	P-3	USA	39
8	15 stores still under refit.		
9	P-4	USA	33
10	40 stores under maintenanc...		
11	P-5	USA	22
12	20 stores were closed due t...		
13	P-6	USA	88
14	Amazing Results!		

Initial sample table with the column headers present and all set to the Text data type, and every other data row containing comments about the data row above it.

The way this report is structured is that you have elements in pairs of rows. Every odd row (1, 3, 5...) contains the data that you need. Every even row, directly underneath each odd row, contains comments about each of those records. You don't need the comments, and you want to remove all of them.

To do that, select **Remove alternate rows** from the table menu. In the **Remove alternate rows** dialog box, enter the following values:

- In the **First row to remove** box, enter 2.
You want to start counting from the second row. Any rows above this **First row to remove** will be kept.
- In the **Number of rows to remove** box, enter 1.
Here you start defining the pattern for removing rows. After you find the second row, you only want to remove that specific row, so you specify that you only need to remove one row.
- In the **Number of rows to keep** box, enter 1.
After you remove one row, you keep the next row. The process starts again for the next row.

Remove alternate rows

Specify the pattern of rows to remove and keep.

First row to remove

Number of rows to remove

Number of rows to keep

The result of that selection will give you the output table that you're looking for. After you set the data types to your columns, your table will look like the following image.

	A ^B _C Period	A ^B _C Country	1 ² ₃ Units
1	Present	USA	51
2	P-1	USA	71
3	P-2	USA	89
4	P-3	USA	39
5	P-4	USA	33
6	P-5	USA	22
7	P-6	USA	88

Filter by values in a column

1/15/2022 • 4 minutes to read • [Edit Online](#)

In Power Query, you can include or exclude rows according to a specific value in a column. You can choose from three methods to filter the values in your column:

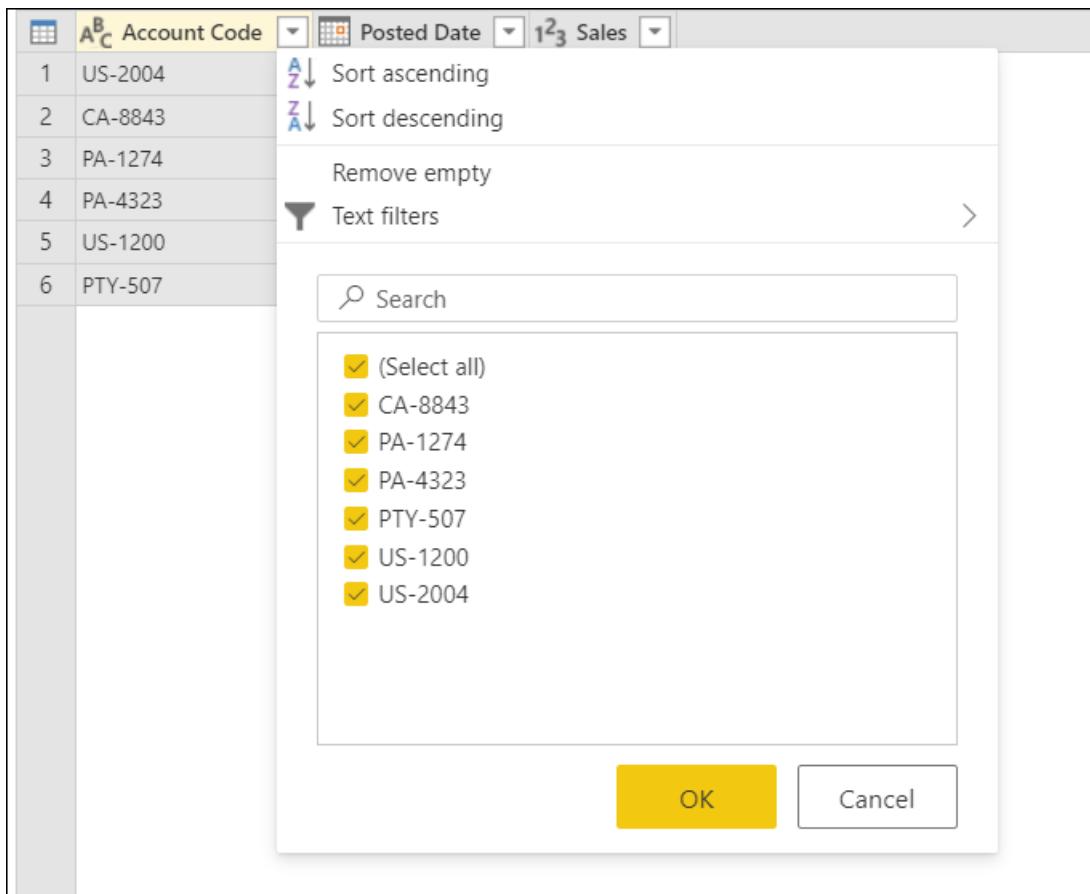
- [Sort and filter menu](#)
- [Cell shortcut menu](#)
- [Type-specific filter](#)

After you apply a filter to a column, a small filter icon appears in the column heading, as shown in the following illustration.



Sort and filter menu

In the column header, you'll see an icon with an inverse triangle. When you select this icon, the sort and filter menu is displayed. With this menu, you can apply or remove any filters to or from your column.



NOTE

In this article, we'll focus on aspects related to filtering data. To learn more about the sort options and how to sort columns in Power Query, go to [Sort columns](#).

Remove empty

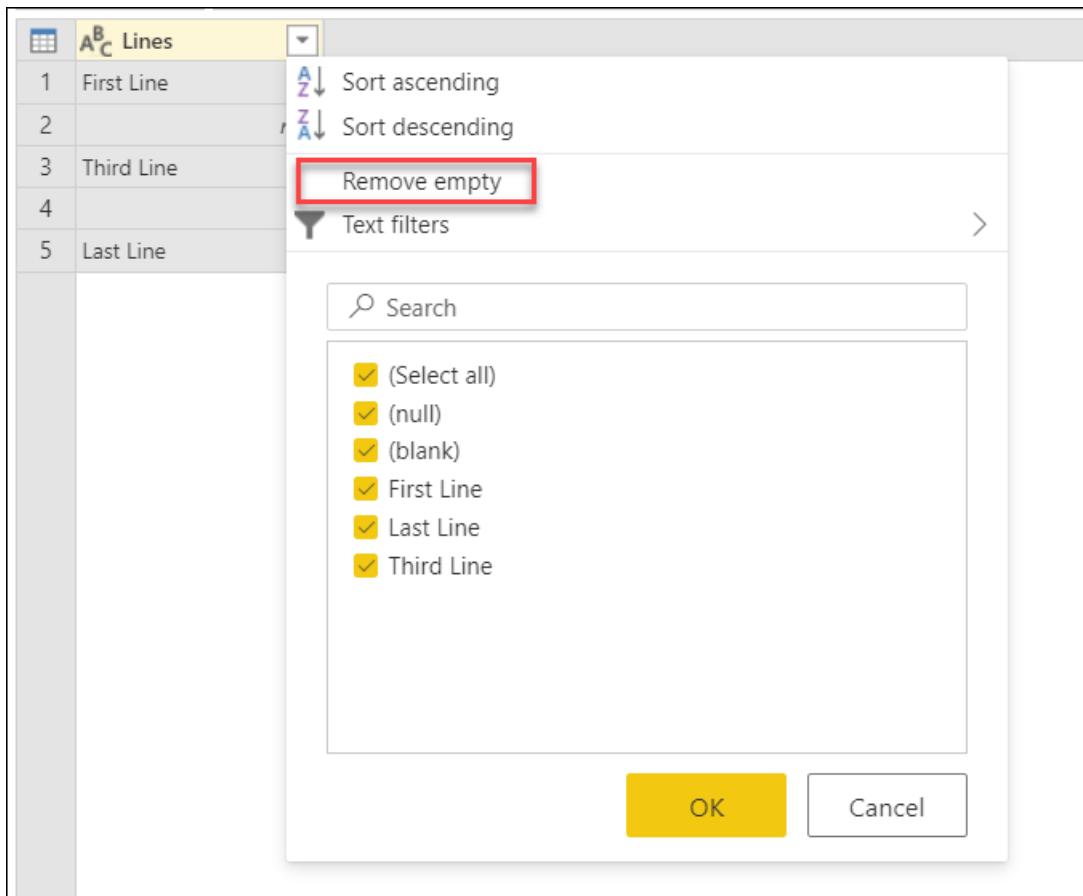
The **Remove empty** command applies two filter rules to your column. The first rule gets rid of any null values. The second rule gets rid of any blank values. For example, imagine a table with just one text column with five rows, where you have one null value and one blank cell.

	A ^B _C Lines	
1	First Line	
2		null
3	Third Line	
4		
5	Last Line	

NOTE

A null value is a specific value in the Power Query language that represents no value.

You then select **Remove empty** from the sort and filter menu, as shown in the following image.



You can also select this option from the **Home** tab in the **Reduce Rows** group in the **Remove Rows** drop-down options, as shown in the next image.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Manage columns' section, a context menu is open over a table named 'Table.TransformColumnTypes(#"Renamed column")'. The menu includes options like 'Remove top rows', 'Remove bottom rows', 'Remove alternate rows', 'Remove duplicates', 'Remove blank rows' (which is highlighted with a red box), and 'Remove errors'.

The result of the Remove empty operation gives you the same table without the empty values.

	A ^B Lines	
1	First Line	
2	Third Line	
3	Last Line	

Clear filter

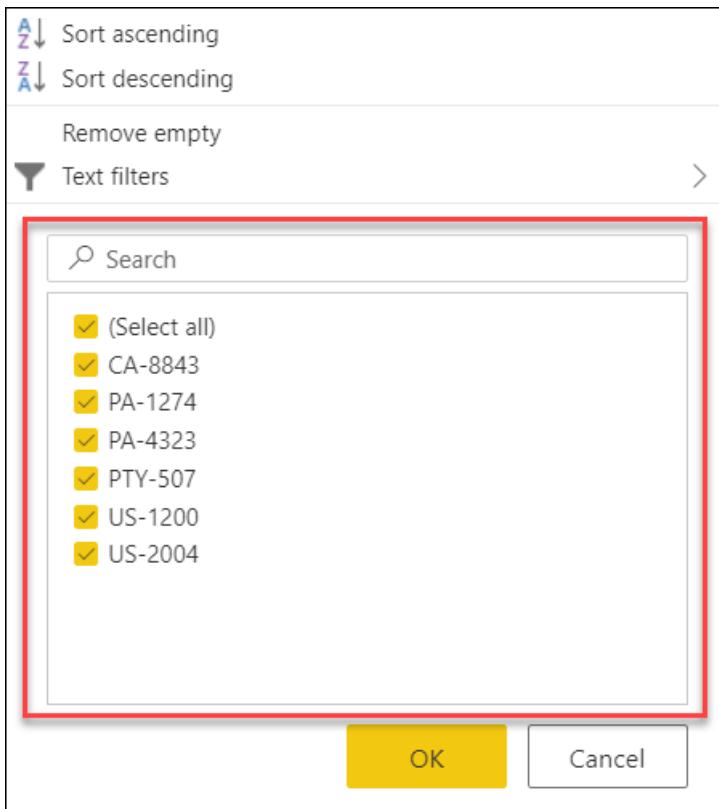
When a filter is applied to a column, the Clear filter command appears on the sort and filter menu.

The screenshot shows the Power BI sort and filter menu for the 'Account Code' column. The menu includes 'Sort ascending', 'Sort descending', 'Clear filter' (which is highlighted with a red box), 'Remove empty', and 'Text filters'. Below the menu is an 'Auto filter' list containing unique values: '(Select all)', 'CA-8843', 'PA-1274', 'PA-4323', 'PTY-507', 'US-1200', and 'US-2004'. The 'PA-1274' value is checked. At the bottom are 'OK' and 'Cancel' buttons.

Auto filter

The list in the sort and filter menu is called the *auto filter* list, which shows the unique values in your column. You can manually select or deselect which values to include in the list. Any selected values will be taken into consideration by the filter; any values that aren't selected will be ignored.

This auto filter section also has a search bar to help you find any values from your list.



NOTE

When you load the auto filter list, only the top 1,000 distinct values in the column are loaded. If there are more than 1,000 distinct values in the column in the that you're filtering, a message will appear indicating that the list of values in the filter list might be incomplete, and the **Load more** link appears. Select the **Load more** link to load another 1,000 distinct values.

- If exactly 1,000 distinct values are found again, the list is displayed with a message stating that the list might still be incomplete.
- If fewer than 1,000 distinct values are found, the full list of values is shown.

Cell shortcut menu

You can right-click a particular cell in a column to open the shortcut menu for that value. Point to the small filter icon, and then select the filter option you want to use.

A screenshot of the Power Query interface showing a table with four columns: Account Code, Posted Date, and Sales. The 'Account Code' column has a context menu open, displaying various filter options. The 'Text filters' section is visible, listing nine different comparison operators for filtering text values.

	Account Code	Posted Date	Sales	
1	US-2004	1/20/2020	580	
2	CA-8843	7/18/2019	280	
3	PA-1274	1/12/2019	90	
4	PA-4323	4/14/2020	187	
5	US-1200	Text filters >	Equals	
6	PTY-507		Does not equal	
		Drill down	Begins with	
		Add as new query	Does not begin with	
			Ends with	
			Does not end with	
			Contains	
			Does not contain	

NOTE

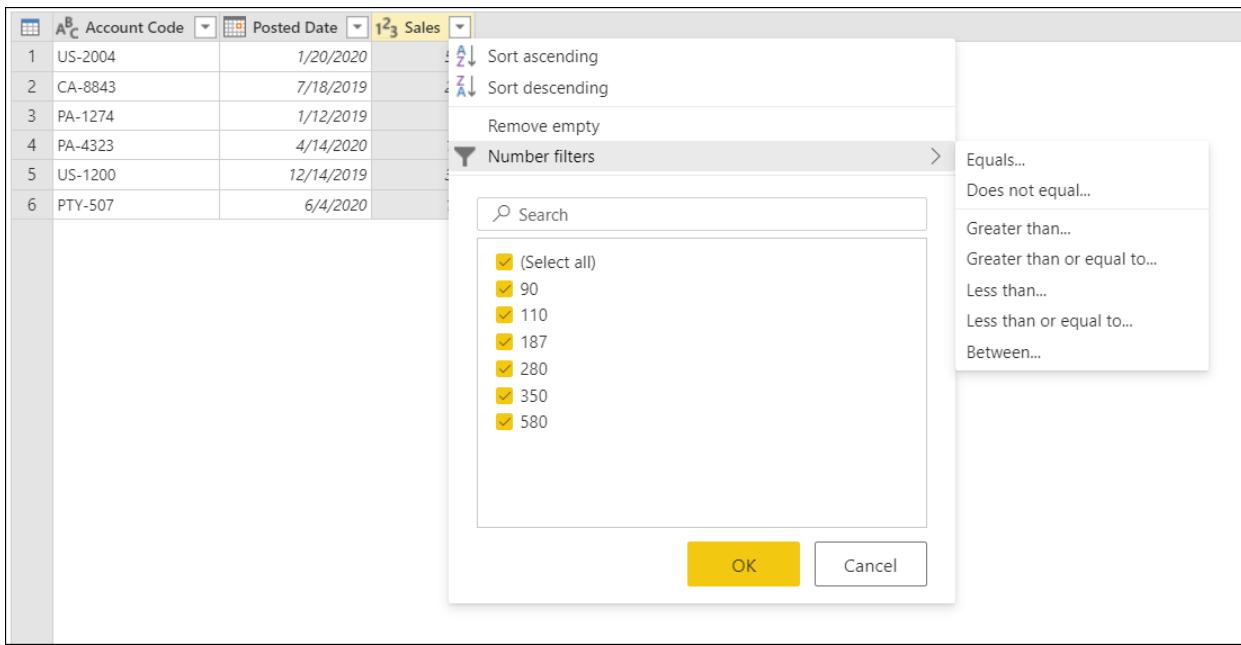
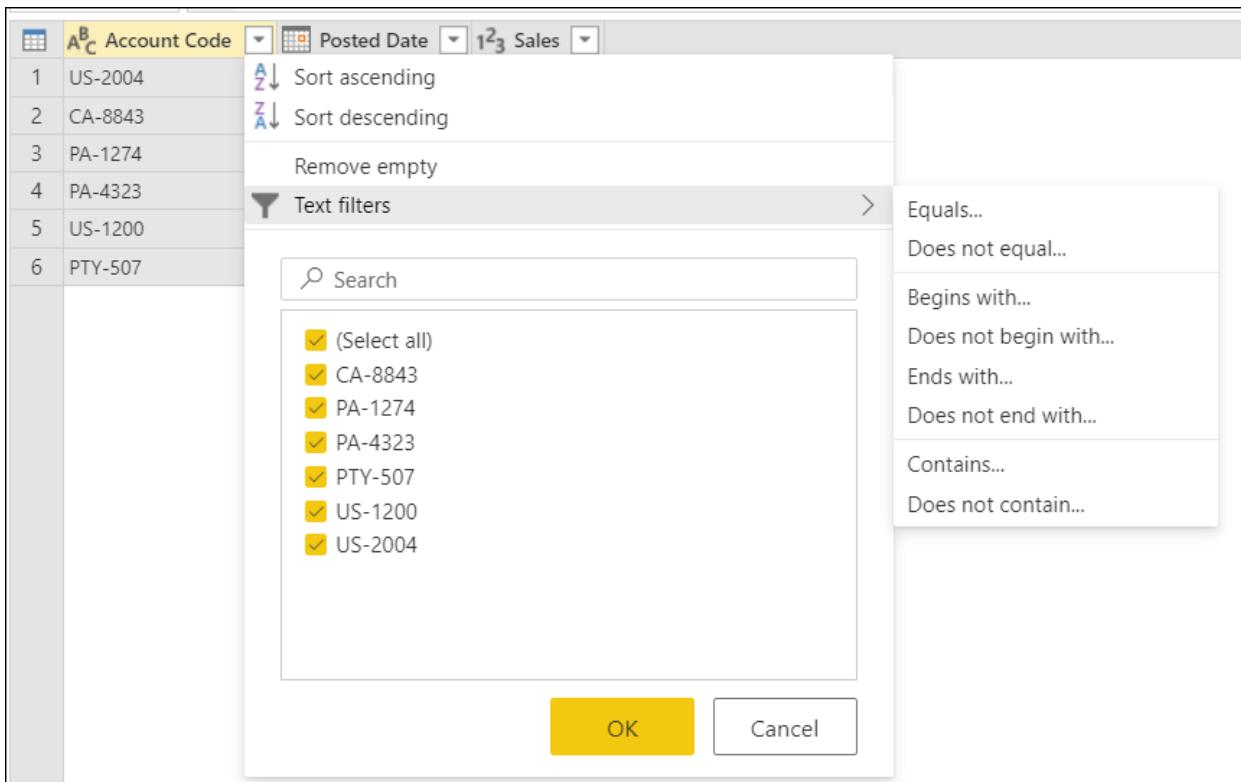
Power Query displays a type-specific filter based on the data type of the column.

Type-specific filters

Depending on the data type of your column, you'll see different commands in the sort and filter menu. The following images show examples for date, text, and numeric columns.

A screenshot of the Power Query interface showing the 'Date Filters' dialog for the 'Posted Date' column. The dialog includes a search bar, a list of specific dates, and a detailed hierarchy for filtering by year, quarter, month, week, day, hour, minute, and second. The 'In the Previous...' option is highlighted with a red box.

LUMNS, 6 ROWS Column profiling based on top 1000 rows



Filter rows

When selecting any of the type-specific filters, you'll use the **Filter rows** dialog box to specify filter rules for the column. This dialog box is shown in the following image.

Filter rows

Apply one or more filter conditions to the rows in this table.

Basic Advanced

Keep rows where "Account Code"

begins with

Enter a value

and or

Enter a value

OK

Cancel

The **Filter rows** dialog box has two modes: **Basic** and **Advanced**.

Basic

With basic mode, you can implement up to two filter rules based on type-specific filters. In the preceding image, notice that the name of the selected column is displayed after the label **Keep rows where**, to let you know which column these filter rules are being implemented on.

For example, imagine that in the following table, you want to filter the **Account Code** by all values that start with either **PA** or **PTY**.

	A B C Account Code	Posted Date	1 2 3 Sales
1	US-2004	1/20/2020	580
2	CA-8843	7/18/2019	280
3	PA-1274	1/12/2019	90
4	PA-4323	4/14/2020	187
5	US-1200	12/14/2019	350
6	PTY-507	6/4/2020	110

To do that, you can go to the **Filter rows** dialog box for the **Account Code** column and specify the set of filter rules you want.

In this example, first select the **Basic** button. Then under **Keep rows where "Account Code"**, select **begins with**, and then enter **PA**. Then select the **or** button. Under the **or** button, select **begins with**, and then enter **PTY**. Then select **OK**.

Filter rows

Apply one or more filter conditions to the rows in this table.

Basic Advanced

Keep rows where "Account Code"

begins with

PA

and or

begins with

PTY

OK

Cancel

The result of that operation will give you the set of rows that you're looking for.

	Account Code	Posted Date	Sales
1	PA-1274	1/12/2019	90
2	PA-4323	4/14/2020	187
3	PTY-507	6/4/2020	110

Advanced

With advanced mode, you can implement as many type-specific filters as necessary from all the columns in the table.

For example, imagine that instead of applying the previous filter in basic mode, you wanted to implement a filter to **Account Code** to show all values that end with 4. Also, you want to show values over \$100 in the **Sales** column.

In this example, first select the **Advanced** button. In the first row, select **Account Code** under **Column name**, **ends with** under **Operator**, and select **4** for the **Value**. In the second row, select **and**, and then select **Sales** under **Column Name**, **is greater than** under **Operator**, and **100** under **Value**. Then select **OK**.

Filter rows

Apply one or more filter conditions to the rows in this table.

Basic Advanced

Keep rows where

Column name

Account Code

Operator

ends with

Value

4

...

and

Sales

is greater than

100

Add clause

OK

Cancel

The result of that operation will give you just one row that meets both criteria.

	A ^B C Account Code	Posted Date	1 ² ₃ Sales
1	US-2004	1/20/2020	580

NOTE

You can add as many clauses as you'd like by selecting **Add clause**. All clauses act at the same level, so you might want to consider creating multiple filter steps if you need to implement filters that rely on other filters.

Choose or remove columns

1/15/2022 • 2 minutes to read • [Edit Online](#)

Choose columns and **Remove columns** are operations that help you define what columns your table needs to keep and which ones it needs to remove. This article will showcase how to use the **Choose columns** and **Remove columns** commands by using the following sample table for both operations.

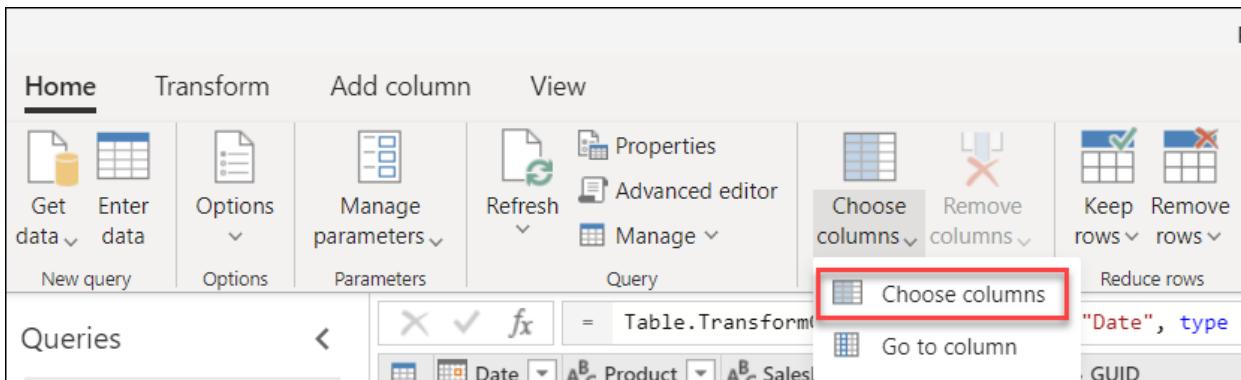
	Date	A ^B C Product	A ^B C SalesPerson	1 ² 3 Units	A ^B C GUID	A ^B C Report created by
1	1/1/2020	Shirt	Jason	20	988f98f6-5016-429c-aeb2-a433d879f723	Miguel
2	1/2/2020	Jeans	Mike	40	7879e75z6-1245-448c-aeb2-a433d879f7...	Miguel
3	1/3/2020	Leggings	Bill	50	1234e56z7-8888-235c-aeb2-a433d879f7...	Miguel

The goal is to create a table that looks like the following image.

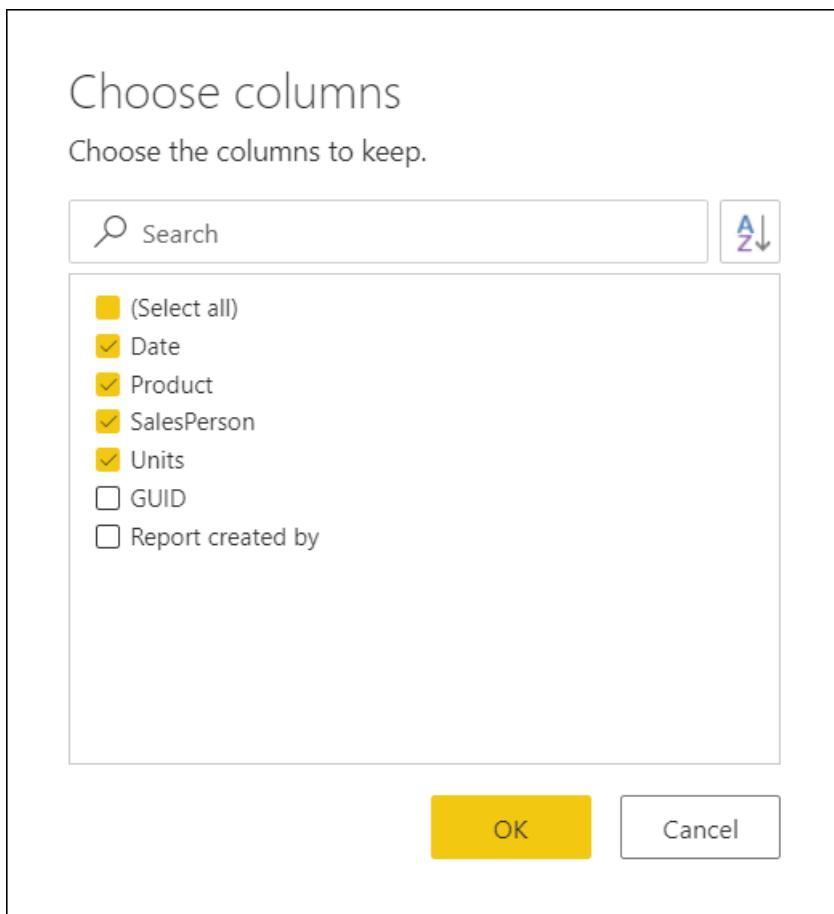
	Date	A ^B C Product	A ^B C SalesPerson	1 ² 3 Units
1	1/1/2020	Shirt	Jason	20
2	1/2/2020	Jeans	Mike	40
3	1/3/2020	Leggings	Bill	50

Choose columns

On the **Home** tab, in the **Manage columns** group, select **Choose columns**.



The **Choose columns** dialog box appears, containing all the available columns in your table. You can select all the fields that you want to keep and remove specific fields by clearing their associated check box. For this example, you want to remove the **GUID** and **Report created by** columns, so you clear the check boxes for those fields.



After selecting **OK**, you'll create a table that only contains the **Date**, **Product**, **SalesPerson**, and **Units** columns.

	Date	Product	SalesPerson	Units
1	1/1/2020	Shirt	Jason	20
2	1/2/2020	Jeans	Mike	40
3	1/3/2020	Leggings	Bill	50

Remove columns

When you select **Remove columns** from the **Home** tab, you have two options:

- **Remove columns**: Removes the selected columns.
- **Remove other columns**: Removes all columns from the table *except* the selected ones.

Remove selected columns

Starting from the sample table, select the **GUID** and the **Report created** columns. Right-click to select any of the column headings. A new shortcut menu appears, where you can select the **Remove columns** command.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Transform' section, the 'Remove columns' button is highlighted with a red box. A dropdown menu is open, listing various options: Remove columns, Remove other columns, Remove duplicates, Remove errors, Replace values, Replace errors, Merge columns, Change type, Transform columns, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, and Move.

Date	Product	SalesPerson	Units	GUID	Report created by
1/1/2020	Shirt	Jason	20	988f98f6-5016-429c-aeb2-a433d879f723	Miguel
1/2/2020	Jeans	Mike	40	7879e75z6-1245-448c-aeb2-a433d879f7...	Miguel
1/3/2020	Leggings	Bill	50	1234e56z7-8888-235c-aeb2-a433d879f7...	Miguel

After selecting **Remove columns**, you'll create a table that only contains the **Date**, **Product**, **SalesPerson**, and **Units** columns.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Transform' section, the 'Remove other columns' button is highlighted with a red box. A dropdown menu is open, listing various options: Remove columns, Remove other columns, Remove duplicates, Remove errors, Replace values, Replace errors, Merge columns, Change type, Transform columns, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, and Move.

Date	Product	SalesPerson	Units
1/1/2020	Shirt	Jason	20
1/2/2020	Jeans	Mike	40
1/3/2020	Leggings	Bill	50

Remove other columns

Starting from the sample table, select all the columns from the table except **GUID** and **Report created**. On the **Home** tab, select **Remove columns** > **Remove other columns**.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Transform' section, the 'Remove other columns' button is highlighted with a red box. A dropdown menu is open, listing various options: Remove columns, Remove other columns, Remove duplicates, Remove errors, Replace values, Replace errors, Merge columns, Change type, Transform columns, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, and Move.

Date	Product	SalesPerson	Units	GUID	Report created by
1/1/2020	Shirt	Jason	20	988f98f6-5016-429c-aeb2-a433d879f723	Miguel
1/2/2020	Jeans	Mike	40	7879e75z6-1245-448c-aeb2-a433d879f7...	Miguel
1/3/2020	Leggings	Bill	50	1234e56z7-8888-235c-aeb2-a433d879f7...	Miguel

After selecting **Remove other columns**, you'll create a table that only contains the **Date**, **Product**, **SalesPerson**, and **Units** columns.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Transform' section, the 'Remove other columns' button is highlighted with a red box. A dropdown menu is open, listing various options: Remove columns, Remove other columns, Remove duplicates, Remove errors, Replace values, Replace errors, Merge columns, Change type, Transform columns, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, and Move.

Date	Product	SalesPerson	Units
1/1/2020	Shirt	Jason	20
1/2/2020	Jeans	Mike	40
1/3/2020	Leggings	Bill	50

Grouping or summarizing rows

1/15/2022 • 5 minutes to read • [Edit Online](#)

In Power Query, you can group values in various rows into a single value by grouping the rows according to the values in one or more columns. You can choose from two types of grouping operations:

- Aggregate a column by using an aggregate function.
- Perform a row operation.

For this tutorial, you'll be using the sample table shown in the following image.

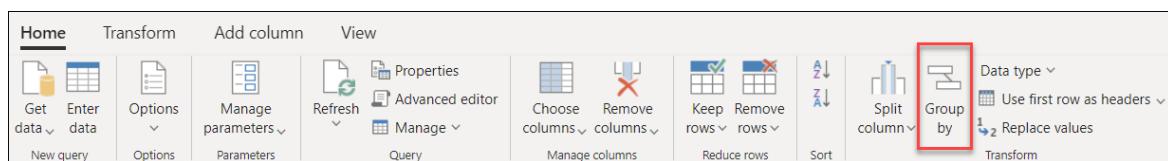
	A ^B C Year	A ^B C Country	A ^B C Product	A ^B C Sales Channel	A ^B C Units
1	2020	USA	Shirt	Online	5000
2	2020	USA	Shorts	Online	4000
3	2020	USA	Shirt	Reseller	7500
4	2020	USA	Shorts	Reseller	4500
5	2020	Panama	Shirt	Online	55
6	2020	Panama	Shorts	Online	70
7	2020	Panama	Shirt	Reseller	200
8	2020	Panama	Shorts	Reseller	150
9	2020	Canada	Shirt	Online	1200
10	2020	Canada	Shorts	Online	1450
11	2020	Canada	Shirt	Reseller	700
12	2020	Canada	Shorts	Reseller	800

Table with columns showing Year (2020), Country (USA, Panama, or Canada), Product (Shirt or Shorts), Sales channel (Online or Reseller), and Units (various values from 55 to 7500)

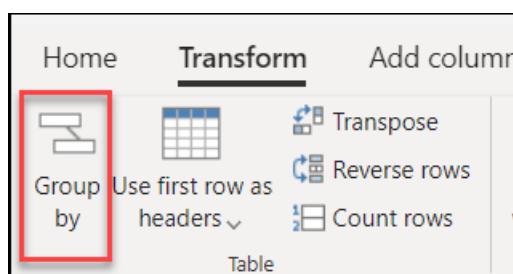
Where to find the Group by button

You can find the **Group by** button in three places:

- On the **Home** tab, in the **Transform** group.



- On the **Transform** tab, in the **Table** group.



- On the shortcut menu when you right-click to select columns.

The screenshot shows a Microsoft Power BI Data Editor interface. A context menu is open over a table with 12 rows of data. The columns are labeled: Year, Country, Product, Sales Channel, and Units. The 'Group by' option in the menu is highlighted with a red box. Other options visible in the menu include Remove columns, Remove other columns, Remove duplicates, Remove errors, Replace values, Replace errors, Merge columns, Change type, Transform columns, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, and Move.

	Year	Country	Product	Sales Channel	Units
1	2020	USA	Shirt	Online	
2	2020	USA	Shorts	Online	
3	2020	USA	Shirt	Reseller	
4	2020	USA	Shorts	Reseller	
5	2020	Panama	Shirt	Online	
6	2020	Panama	Shorts	Online	
7	2020	Panama	Shirt	Reseller	
8	2020	Panama	Shorts	Reseller	
9	2020	Canada	Shirt	Online	
10	2020	Canada	Shorts	Online	
11	2020	Canada	Shirt	Reseller	
12	2020	Canada	Shorts	Reseller	

Use an aggregate function to group by one or more columns

In this example, your goal is to summarize the total units sold at the country and sales channel level. You'll use the **Country** and **Sales Channel** columns to perform the group by operation.

1. Select **Group by** on the **Home** tab.
2. Select the **Advanced** option, so you can select multiple columns to group by.
3. Select the **Country** and **Sales Channel** columns.
4. In the New columns section, create a new column where the name is **Total units**, the aggregate operation is **Sum**, and the column used is **Units**.
5. Hit OK

Group by

Specify the column to group by and the desired output.

Basic Advanced

Group by

Country

Sales Channel

Add grouping

New column name

Total units

Operation

Sum

Column

Units

Add aggregation

Use fuzzy grouping

> Fuzzy group options

OK

Cancel

This operation gives you the table that you're looking for.

The screenshot shows the Power Query - Edit queries window. The ribbon at the top has 'Transform' selected. The main area displays a table with three columns: 'Country' (containing USA, USA, Panama, Panama, Canada, Canada), 'Sales Channel' (containing Online, Reseller, Online, Reseller, Online, Reseller), and 'Total units' (containing 9000, 12000, 125, 350, 2650, 1500). Above the table, the formula bar shows: Table.Group(#"Changed Type", {"Country", "Sales Channel"}, {"Total units", each List.Sum([Units]), type number}). To the right of the table, there are 'Query settings' (Name: Query, Entity type: Custom) and an 'Applied steps' pane (Source, Changed Type, Grouped Rows).

Perform a row operation to group by one or more columns

In this example, you want total units sold and—in addition—you want two other columns that give you the name and units sold for the top-performing product, summarized at the country and sales channel level.

Your goal is to reach a table that looks like the following image from your original sample table.

	Country	Sales Channel	1. Total units	2. Top performer product.Product	3. Top performer product.Units
1	USA	Online	9000	Shirt	5000
2	USA	Reseller	12000	Shirt	7500
3	Panama	Online	125	Shorts	70
4	Panama	Reseller	350	Shirt	200
5	Canada	Online	2650	Shorts	1450
6	Canada	Reseller	1500	Shorts	800

1. Use the following columns as **Group by** columns:

- Country
- Sales Channel

2. Create two new columns by doing the following:

- Aggregate the **Units** column by using the **Sum** operation. Name this column **Total units**.
- Add a new **Products** column by using the **All rows** operation.

Group by

Specify the column to group by and the desired output.

Basic Advanced

Group by

Country	▼
Sales Channel	▼

Add grouping

New column name	Operation	Column
Total units	Sum	Units
Products	All rows	▼

Add aggregation

Use fuzzy grouping
 Fuzzy group options

OK
Cancel

After that operation is complete, notice how the **Products** column has [Table] values inside each cell. Each [Table] value contains all the rows that were grouped by the **Country** and **Sales Channel** columns from your original table. You can select the white space inside the cell to see a preview of the contents of the table at the bottom of the dialog box.

The screenshot shows a Power BI data model with two tables:

- Products** table (top):

	Country	Sales Channel	Total units	Products
1	USA	Online	9000	[Table]
2	USA	Reseller	12000	[Table]
3	Panama	Online	125	[Table]
4	Panama	Reseller	350	[Table]
5	Canada	Online	2650	[Table]
6	Canada	Reseller	1500	[Table]
- Details Preview** (bottom):

Year	Country	Product	Sales Channel	Units
2020	USA	Shirt	Online	5000
2020	USA	Shorts	Online	4000

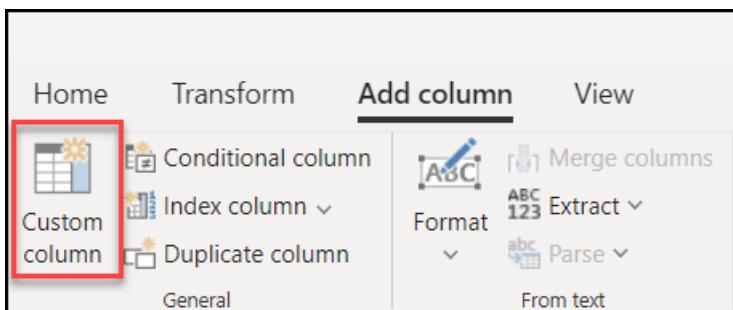
NOTE

The details preview pane might not show all the rows that were used for the group-by operation. You can select the [Table] value to see all rows pertaining to the corresponding group-by operation.

Next, you need to extract the row that has the highest value in the **Units** column of the tables inside the new **Products** column, and call that new column **Top performer product**.

Extract the top performer product information

With the new **Products** column with [Table] values, you create a new custom column by going to the **Add Column** tab on the ribbon and selecting **Custom column** from the **General** group.



Name your new column **Top performer product**. Enter the formula `Table.Max([Products], "Units")` under **Custom column formula**.

Custom column

Add a column that is computed from other columns or values.

New column name

Top performer product

Custom column formula ⓘ

```
= Table.Max([Products], "Units")
```

Available column(s)

Country
Sales Channel
Total units
Products

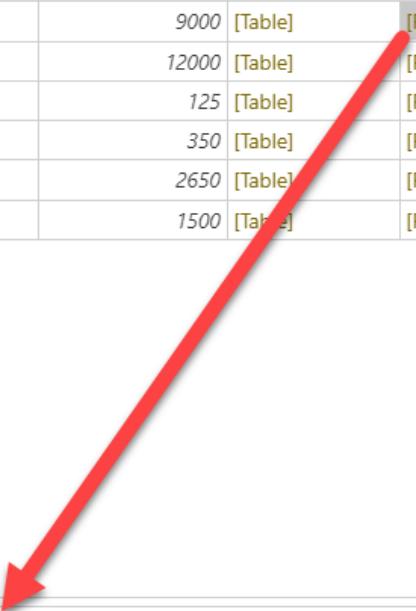
Insert column

[Learn more about Power Query formulas](#)

OK

Cancel

The result of that formula creates a new column with [Record] values. These record values are essentially a table with just one row. These records contain the row with the maximum value for the **Units** column of each [Table] value in the **Products** column.



	Country	Sales Channel	Total units	Products	Top performer product
1	USA	Online	9000	[Table]	[Record]
2	USA	Reseller	12000	[Table]	[Record]
3	Panama	Online	125	[Table]	[Record]
4	Panama	Reseller	350	[Table]	[Record]
5	Canada	Online	2650	[Table]	[Record]
6	Canada	Reseller	1500	[Table]	[Record]

Year	2020
Country	USA
Product	Shirt
Sales Channel	Online
Units	5000

With this new **Top performer product** column that contains [Record] values, you can select the ⚡ expand icon,

select the **Product** and **Units** fields, and then select **OK**.

The screenshot shows the Power Query ribbon with the 'Products' column highlighted. A context menu is open, displaying options for 'Select all', 'Year', 'Country', 'Product', 'Sales Channel', and 'Units'. The 'Product' and 'Units' checkboxes are checked. A checkbox for 'Use original column name as prefix' is also present. At the bottom are 'OK' and 'Cancel' buttons.

After removing your **Products** column and setting the data type for both newly expanded columns, your result will resemble the following image.

	Country	Sales Channel	Total units	Top performer product	Top performer product.Units
1	USA	Online	9000	Shirt	5000
2	USA	Reseller	12000	Shirt	7500
3	Panama	Online	125	Shorts	70
4	Panama	Reseller	350	Shirt	200
5	Canada	Online	2650	Shorts	1450
6	Canada	Reseller	1500	Shorts	800

Fuzzy grouping

NOTE

The following feature is only available in Power Query Online.

To demonstrate how to do "fuzzy grouping," consider the sample table shown in the following image.

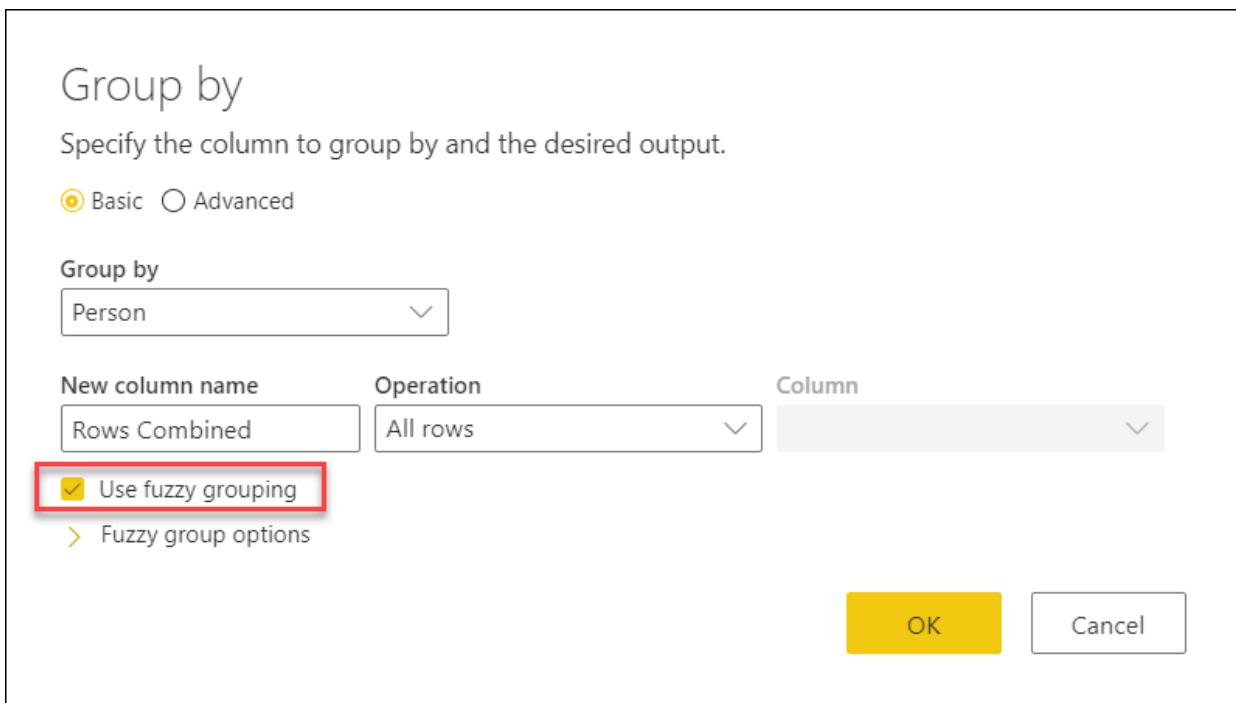
	A ^B C id	A ^B C Person
1	1	miguel
2	2	Miguel
3	3	migueel
4	4	mike
5	5	Mike
6	6	William
7	7	Bill
8	8	billy
9	9	Miguel

The goal of fuzzy grouping is to do a group-by operation that uses an approximate match algorithm for text strings. Power Query uses the Jaccard similarity algorithm to measure the similarity between pairs of instances. Then it applies agglomerative hierarchical clustering to group instances together. The following image shows the

output that you expect, where the table will be grouped by the **Person** column.

	A C Person	1 2 3 Frequency
1	Miguel	3
2	mike	2
3	Bill	3
4	William	1

To do the fuzzy grouping, you perform the same steps previously described in this article. The only difference is that this time, in the **Group by** dialog box, you select the **Use fuzzy grouping** check box.



For each group of rows, Power Query will pick the most frequent instance as the "canonical" instance. If multiple instances occur with the same frequency, Power Query will pick the first one. After you select **OK** in the **Group by** dialog box, you'll get the result that you were expecting.

	A C Person	1 2 3 Frequency
1	Miguel	3
2	mike	2
3	Bill	3
4	William	1

However, you have more control over the fuzzy grouping operation by expanding **Fuzzy group options**.

Group by

Specify the column to group by and the desired output.

Basic Advanced

Group by

Person

New column name

Frequency

Operation

Count rows

Column

▼

Use fuzzy grouping

Fuzzy group options

Similarity threshold (optional) ⓘ

0.8

Ignore case

Group by combining text parts ⓘ

Transformation table (optional) ⓘ

▼

OK

Cancel

The following options are available for fuzzy grouping:

- Similarity threshold (optional):** This option indicates how similar two values must be to be grouped together. The minimum setting of 0 will cause all values to be grouped together. The maximum setting of 1 will only allow values that match exactly to be grouped together. The default is 0.8.
- Ignore case:** When comparing text strings, case will be ignored. This option is enabled by default.
- Group by combining text parts:** The algorithm will try to combine text parts (such as combining Micro and soft into Microsoft) to group values.
- Transformation table (optional):** You can select a transformation table that will map values (such as mapping MSFT to Microsoft) to group them together.

For this example, a transformation table will be used to demonstrate how values can be mapped. The transformation table has two columns:

- From:** The text string to look for in your table.
- To:** The text string to use to replace the text string in the **From** column.

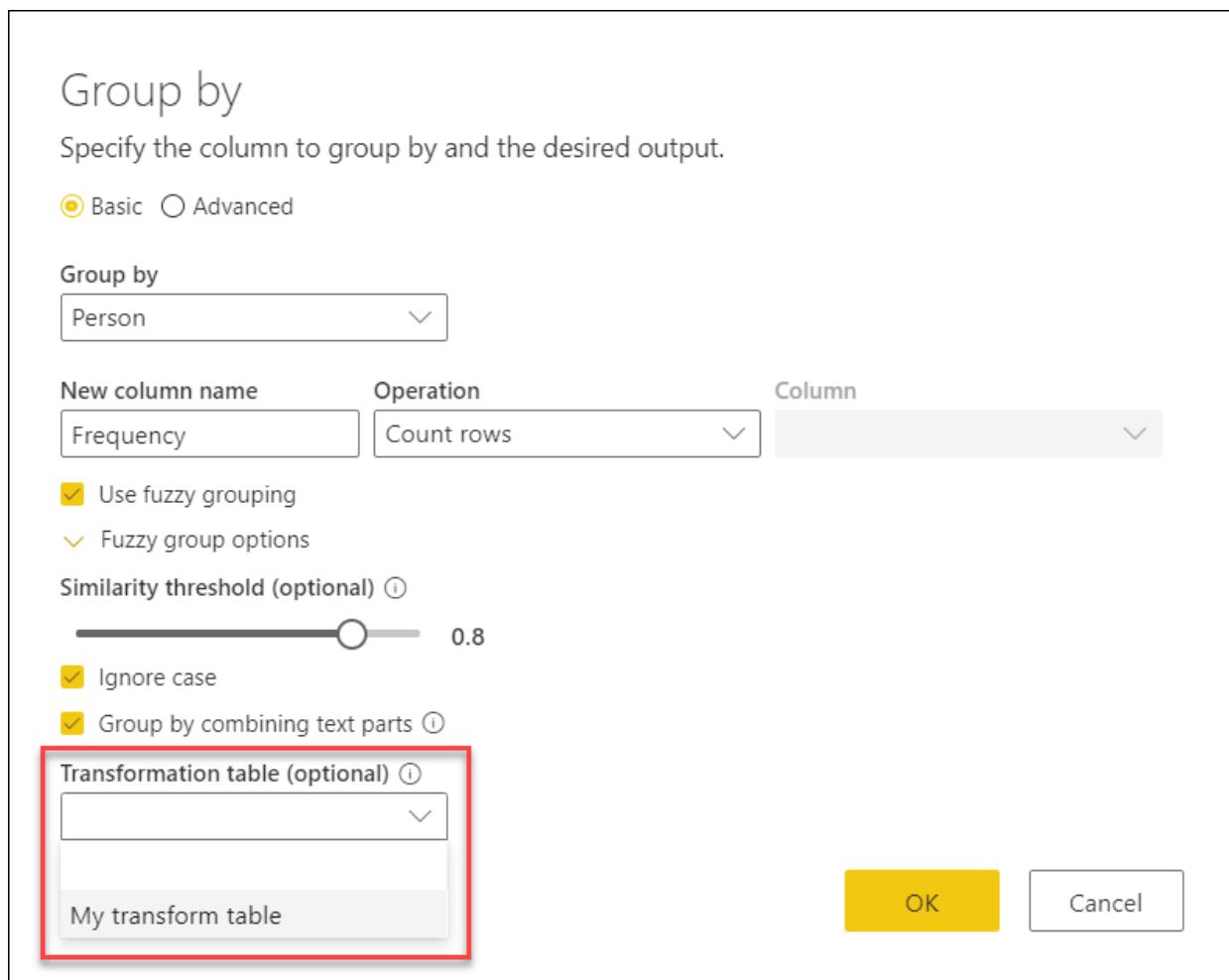
The following image shows the transformation table used in this example.

	A ^B C From	A ^B C To
1	mike	Miguel
2	William	Bill

IMPORTANT

It's important that the transformation table has the same columns and column names as shown above (they have to be "From" and "To"), otherwise Power Query will not recognize these.

Return to the **Group by** dialog box, expand **Fuzzy group options**, and then select the **Transformation table** drop-down menu.



After selecting your transformation table, select **OK**. The result of that operation will give you the result shown in the following image.

	A	B	C	D	E
1	Miguel		Frequency	6	
2	William			3	

In this example, the **Ignore case** option was enabled, so the values in the **From** column of the **Transformation table** will be used to look for the text string without considering the case of the string. This transformation operation occurs first, and then the fuzzy grouping operation is performed.

NOTE

When grouping by multiple columns, the transformation table will perform the replace operation in all columns if replacing the value increases the similarity score.

See also

- [Add a custom column](#)
- [Remove duplicates](#)

Unpivot columns

1/15/2022 • 7 minutes to read • [Edit Online](#)

In Power Query, you can transform columns into attribute-value pairs, where columns become rows.

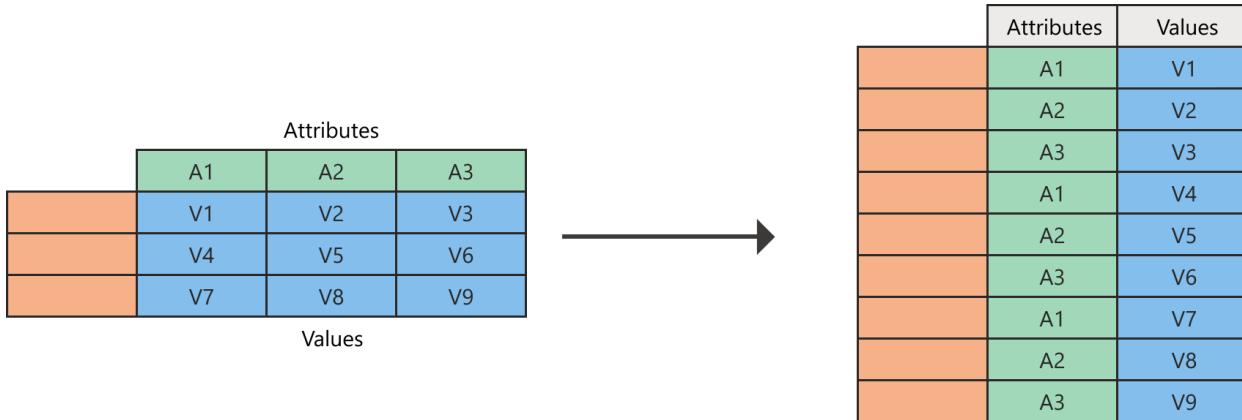


Diagram showing a table on the left with a blank column and rows, and the Attributes values A1, A2, and A3 as column headers. The A1 column contains the values V1, V4, and V7, the A2 column contains the values V2, V5, and V8, and the A3 column contains the values V3, V6, and V9. With the columns unpivoted, a table on the right of the diagram contains a blank column and rows, an Attributes column with nine rows with A1, A2, and A3 repeated three times, and a Values column with values V1 through V9.

For example, given a table like the following, where country rows and date columns create a matrix of values, it's difficult to analyze the data in a scalable way.

	Country	6/1/2020	7/1/2020	8/1/2020
1	USA	785	450	567
2	Canada	357	421	254
3	Panama	20	40	80

Table containing a Country column set in the Text data type, and 6/1/2020, 7/1/2020, and 8/1/2020 columns set as the Whole number data type. The Country column contains USA in row 1, Canada in row 2, and Panama in row 3.

Instead, you can transform the table into a table with unpivoted columns, as shown in the following image. In the transformed table, it's easier to use the date as an attribute to filter on.

	Country	Attribute	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

Table containing a Country column set as the Text data type, an Attribute column set as the Text data type, and a

Value column set as the Whole number data type. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Attribute column contains 6/1/2020 in the first, forth, and seventh rows, 7/1/2020 in the second, fifth, and eighth rows, and 8/1/2020 in the third, sixth, and ninth rows.

The key in this transformation is that you have a set of dates in the table that should all be part of a single column. The respective value for each date and country should be in a different column, effectively creating an attribute-value pair.

Power Query will always create the attribute-value pair by using two columns:

- **Attribute:** The name of the column headings that were unpivoted.
- **Value:** The values that were underneath each of the unpivoted column headings.

There are multiple places in the user interface where you can find **Unpivot columns**. You can right-click the columns that you want to unpivot, or you can select the command from the **Transform** tab in the ribbon.

A screenshot of the Microsoft Power Query ribbon. The 'Transform' tab is highlighted in yellow, indicating it is the active tab. A context menu is open over a table, with the 'Unpivot columns' option highlighted in grey, indicating it is the selected command. The table has three columns: 'Country' (containing USA, Canada, Panama), 'Attribute' (containing 6/1/2020, 7/1/2020, 8/1/2020), and 'Value' (containing 450, 567, 421, 254, 40, 80).

The screenshot shows the Power BI interface with the 'Transform' ribbon tab selected. In the center, there is a table with three rows and four columns. The first column contains numbers 1, 2, and 3. The second column contains country names: USA, Canada, and Panama. The third column contains dates: 6/1/2020, 7/1/2020, and 8/1/2020. The fourth column contains numerical values: 785, 357, 20; 450, 421, 40; and 254, 80, 80 respectively. A context menu is open over the third column, listing three options: 'Unpivot columns', 'Unpivot other columns', and 'Unpivot only selected columns'. The 'Unpivot columns' option is highlighted.

There are three ways that you can unpivot columns from a table:

- **Unpivot columns**
- **Unpivot other columns**
- **Unpivot only selected columns**

Unpivot columns

For the scenario described above, you first need to select the columns you want to unpivot. You can select **Ctrl** as you select as many columns as you need. For this scenario, you want to select all the columns except the one named **Country**. After selecting the columns, right-click any of the selected columns, and then select **Unpivot columns**.

The screenshot shows the Power BI interface with the 'Transform' ribbon tab selected. In the center, there is a table with three rows and four columns. The first column contains numbers 1, 2, and 3. The second column contains country names: USA, Canada, and Panama. The third column contains dates: 6/1/2020, 7/1/2020, and 8/1/2020. The fourth column contains numerical values: 785, 357, 20; 450, 421, 40; and 254, 80, 80 respectively. A context menu is open over the third column, listing various data transformation options. The 'Unpivot columns' option is highlighted.

The result of that operation will yield the result shown in the following image.

The screenshot shows the Power Query - Edit queries window. The ribbon at the top has tabs for Home, Transform, Add column, and View. The Home tab is selected. The ribbon contains various icons for data operations like Get data, Enter data, Options, Manage parameters, Refresh, Advanced editor, Properties, Manage, Choose columns, Remove columns, Keep rows, Remove rows, Sort, Reduce rows, Data type, Split column, Group by, Use first row as headers, Merge queries, Append queries, Combine files, Combine, Map to standard, CDM, and AI insights. Below the ribbon is a 'Queries' pane on the left containing a 'Query' item. The main area displays a table titled 'Table.UnpivotOtherColumns(#"Changed Type", {"Country"}, "Attribute", "Value")'. The table has three columns: 'Country' (Text), 'Attribute' (Text), and 'Value' (Whole number). The data is as follows:

	Country	Attribute	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

On the right side, there is a 'Query settings' pane with fields for 'Name' (Query) and 'Entity type' (Custom). Below that is a 'Applied steps' section with a list: 'Source' (Changed Type) and 'Unpivoted columns'.

Table containing a Country column set as the Text data type, an Attribute column set as the Text data type, and a Value column set as the Whole number data type. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Attribute column contains 6/1/2020 in the first, forth, and seventh rows, 7/1/2020 in the second, fifth, and eighth rows, and 8/1/2020 in the third, sixth, and ninth rows. In addition, the Unpivot columns entry is emphasized in the Query settings pane and the M language code is shown in the formula bar.

Special considerations

After creating your query from the steps above, imagine that your initial table gets updated to look like the following screenshot.

	Country	6/1/2020	7/1/2020	8/1/2020	9/1/2020
1	USA	785	450	567	645
2	Canada	357	421	254	330
3	Panama	20	40	80	50
4	UK	543	435	400	700
5	Mexico	150	180	204	170

Table with the same original Country, 6/1/2020, 7/1/2020, and 8/1/2020 columns, with the addition of a 9/1/2020 column. The Country column still contains the USA, Canada, and Panama values, but also has UK added to the fourth row and Mexico added to the fifth row.

Notice that you've added a new column for the date 9/1/2020 (September 1, 2020), and two new rows for the countries UK and Mexico.

If you refresh your query, you'll notice that the operation will be done on the updated column, but won't affect the column that wasn't originally selected (**Country**, in this example). This means that any new column that's added to the source table will be unpivoted as well.

The following image shows what your query will look like after the refresh with the new updated source table.

	A ^B _C Country	A ^B _C Attribute	1 ² ₃ Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	USA	9/1/2020	645
5	Canada	6/1/2020	357
6	Canada	7/1/2020	421
7	Canada	8/1/2020	254
8	Canada	9/1/2020	330
9	Panama	6/1/2020	20
10	Panama	7/1/2020	40
11	Panama	8/1/2020	80
12	Panama	9/1/2020	50
13	UK	6/1/2020	543
14	UK	7/1/2020	435
15	UK	8/1/2020	400
16	UK	9/1/2020	700
17	Mexico	6/1/2020	150
18	Mexico	7/1/2020	180
19	Mexico	8/1/2020	204
20	Mexico	9/1/2020	170

Table with Country, Attribute, and Value columns. The first four rows of the Country column contains USA, the second four rows contains Canada, the third four rows contains Panama, the fourth four rows contains UK, and the fifth four rows contains Mexico. The Attribute column contains 6/1/2020, 7/1/2020, 8/1/2020, and 9/1/2020 in the first four rows, which are repeated for each country.

Unpivot other columns

You can also select the columns that you don't want to unpivot and unpivot the rest of the columns in the table. This operation is where **Unpivot other columns** comes into play.

The screenshot shows a Power BI Data Editor window with a context menu open over a table. The table has three columns: Country, Attribute, and Value. The Country column contains USA, Canada, and Panama. The Attribute column contains 6/1/2020, 7/1/2020, and 8/1/2020. The Value column contains 785, 450, 567, 357, 421, 254, 20, 40, and 80. The context menu is expanded, showing various data transformation options.

The result of that operation will yield exactly the same result as the one you got from **Unpivot columns**.

	Country	Attribute	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

Table containing a Country column set as the Text data type, an Attribute column set as the Text data type, and a Value column set as the Whole number data type. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Attribute column contains 6/1/2020 in the first, forth, and seventh rows, 7/1/2020 in the second, fifth, and eighth rows, and 8/1/2020 in the third, sixth, and ninth rows.

NOTE

This transformation is crucial for queries that have an unknown number of columns. The operation will unpivot all columns from your table except the ones that you've selected. This is an ideal solution if the data source of your scenario got new date columns in a refresh, because those will get picked up and unpivoted.

Special considerations

Similar to the **Unpivot columns** operation, if your query is refreshed and more data is picked up from the data source, all the columns will be unpivoted except the ones that were previously selected.

To illustrate this, say that you have a new table like the one in the following image.

	A ^B Country	A ^B 6/1/2020	A ^B 7/1/2020	A ^B 8/1/2020	A ^B 9/1/2020
1	USA	785	450	567	645
2	Canada	357	421	254	330
3	Panama	20	40	80	50
4	UK	543	435	400	700
5	Mexico	150	180	204	170

Table with Country, 6/1/2020, 7/1/2020, 8/1/2020, and 9/1/2020 columns, with all columns set to the Text data type. The Country column contains, from top to bottom, USA, Canada, Panama, UK, and Mexico.

You can select the **Country** column, and then select **Unpivot other column**, which will yield the following result.

	A ^B Country	A ^B Attribute	1 ² 3 Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	USA	9/1/2020	645
5	Canada	6/1/2020	357
6	Canada	7/1/2020	421
7	Canada	8/1/2020	254
8	Canada	9/1/2020	330
9	Panama	6/1/2020	20
10	Panama	7/1/2020	40
11	Panama	8/1/2020	80
12	Panama	9/1/2020	50
13	UK	6/1/2020	543
14	UK	7/1/2020	435
15	UK	8/1/2020	400
16	UK	9/1/2020	700
17	Mexico	6/1/2020	150
18	Mexico	7/1/2020	180
19	Mexico	8/1/2020	204
20	Mexico	9/1/2020	170

Table with Country, Attribute, and Value columns. The Country and Attribute columns are set to the Text data type. The Value column is set to the Whole value data type. The first four rows of the Country column contain USA, the second four rows contains Canada, the third four rows contains Panama, the fourth four rows contains UK, and the fifth four rows contains Mexico. The Attribute column contains 6/1/2020, 7/1/2020, 8/1/2020, and 9/1/2020 in the first four rows, which are repeated for each country.

Unpivot only selected columns

The purpose of this last option is to only unpivot specific columns from your table. This is important for scenarios where you're dealing with an unknown number of columns from your data source and you only want

to unpivot the selected columns.

To perform this operation, select the columns to unpivot, which in this example is all the columns except the **Country** column. Then right-click any of the columns you selected, and then select **Unpivot only selected columns**.

The screenshot shows the Power BI Data Editor interface. A context menu is open over the third column, labeled "Attribute". The menu items are:

- Remove columns
- Remove other columns
- Remove duplicates
- Remove errors
- Replace values
- Replace errors
- Merge columns
- Change type >
- Transform columns >
- Group by
- Fill >
- Unpivot columns
- Unpivot other columns
- Unpivot only selected columns** (highlighted)
- Move >

Notice how this operation will yield the same output as the previous examples.

	Country	Attribute	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

Table containing a Country column set as the Text data type, an Attribute column set as the Text data type, and a Value column set as the Whole number data type. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Attribute column contains 6/1/2020 in the first, forth, and seventh rows, 7/1/2020 in the second, fifth, and eighth rows, and 8/1/2020 in the third, sixth, and ninth rows.

Special considerations

After doing a refresh, if our source table changes to have a new **9/1/2020** column and new rows for UK and Mexico, the output of the query will be different from the previous examples. Say that our source table, after a refresh, changes to the table in the following image.

	A ^B _C Country	A ^B _C 6/1/2020	A ^B _C 7/1/2020	A ^B _C 8/1/2020	A ^B _C 9/1/2020
1	USA	785	450	567	645
2	Canada	357	421	254	330
3	Panama	20	40	80	50
4	UK	543	435	400	700
5	Mexico	150	180	204	170

The output of our query will look like the following image.

	A ^B _C Country	1 ² ₃ 9/1/2020	A ^B _C Attribute	1 ² ₃ Value
1	USA		6/1/2020	785
2	USA		7/1/2020	450
3	USA		8/1/2020	567
4	Canada		6/1/2020	357
5	Canada		7/1/2020	421
6	Canada		8/1/2020	254
7	Panama		6/1/2020	20
8	Panama		7/1/2020	40
9	Panama		8/1/2020	80
10	UK		6/1/2020	543
11	UK		7/1/2020	435
12	UK		8/1/2020	400
13	Mexico		6/1/2020	150
14	Mexico		7/1/2020	180
15	Mexico		8/1/2020	204

It looks like this because the unpivot operation was applied only on the **6/1/2020**, **7/1/2020**, and **8/1/2020** columns, so the column with the header **9/1/2020** remains unchanged.

Pivot columns

1/15/2022 • 4 minutes to read • [Edit Online](#)

In Power Query, you can create a table that contains an aggregate value for each unique value in a column. Power Query groups each unique value, does an aggregate calculation for each value, and pivots the column into a new table.

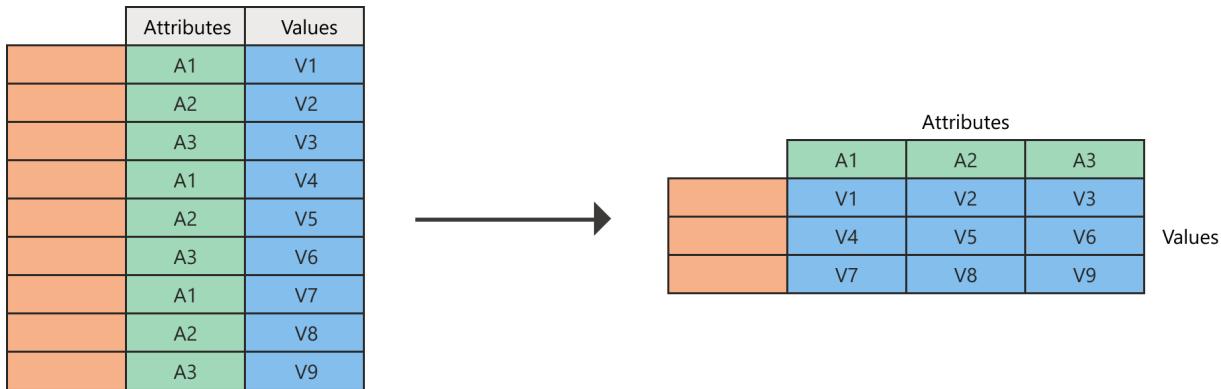


Diagram showing a table on the left with a blank column and rows. An Attributes column contains nine rows with A1, A2, and A3 repeated three times. A Values column contains, from top to bottom, values V1 through V9. With the columns pivoted, a table on the right contains a blank column and rows, the Attributes values A1, A2, and A3 as column headers, with the A1 column containing the values V1, V4, and V7, the A2 column containing the values V2, V5, and V8, and the A3 column containing the values V3, V6, and V9.

Imagine a table like the one in the following image.

	Country	Date	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

Table containing a Country column set as the Text data type, a Date column set as the Data data type, and a Value column set as the Whole number data type. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Date column contains 6/1/2020 in the first, forth, and seventh rows, 7/1/2020 in the second, fifth, and eighth rows, and 8/1/2020 in the third, sixth, and ninth rows.

This table contains values by country and date in a simple table. In this example, you want to transform this table into the one where the date column is pivoted, as shown in the following image.

	A ^B _C Country	1 ² ₃ 6/1/2020	1 ² ₃ 7/1/2020	1 ² ₃ 8/1/2020
1	Canada	357	421	254
2	Panama	20	40	80
3	USA	785	450	567

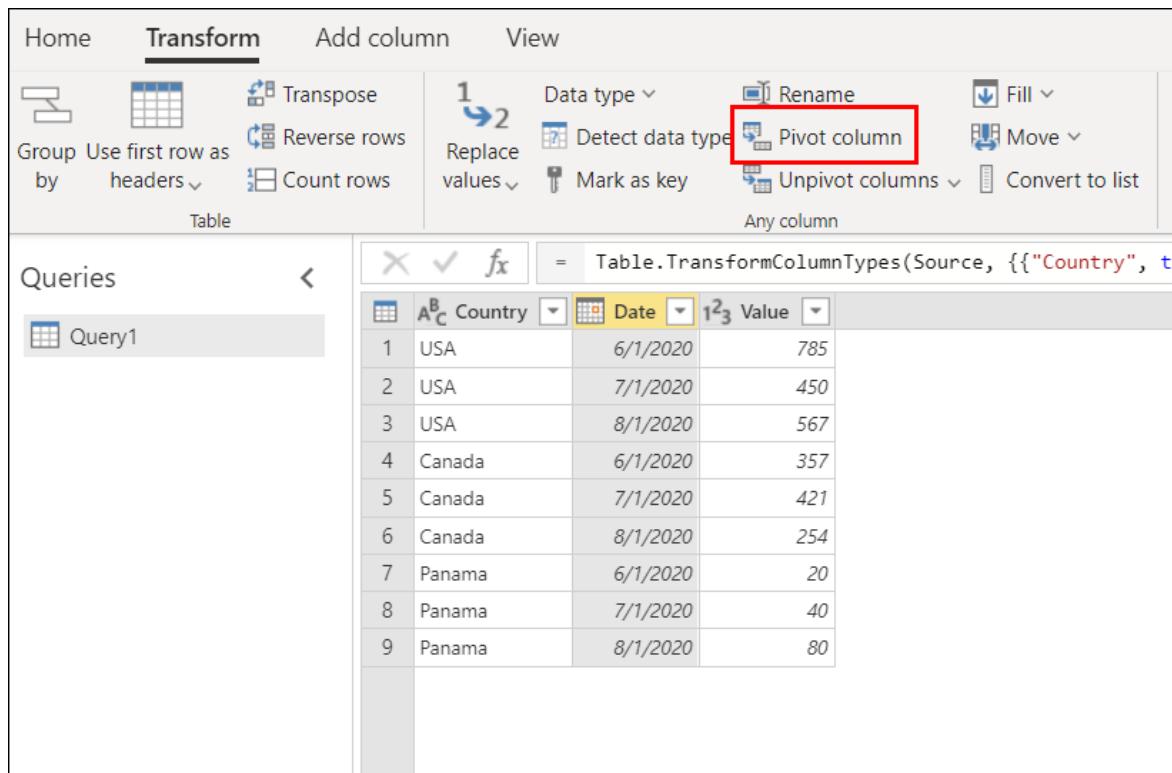
Table containing a Country column set in the Text data type, and 6/1/2020, 7/1/2020, and 8/1/2020 columns set as the Whole number data type. The Country column contains Canada in row 1, Panama in row 2, and USA in row 3.

NOTE

During the pivot columns operation, Power Query will sort the table based on the values found on the first column—at the left side of the table—in ascending order.

To pivot a column

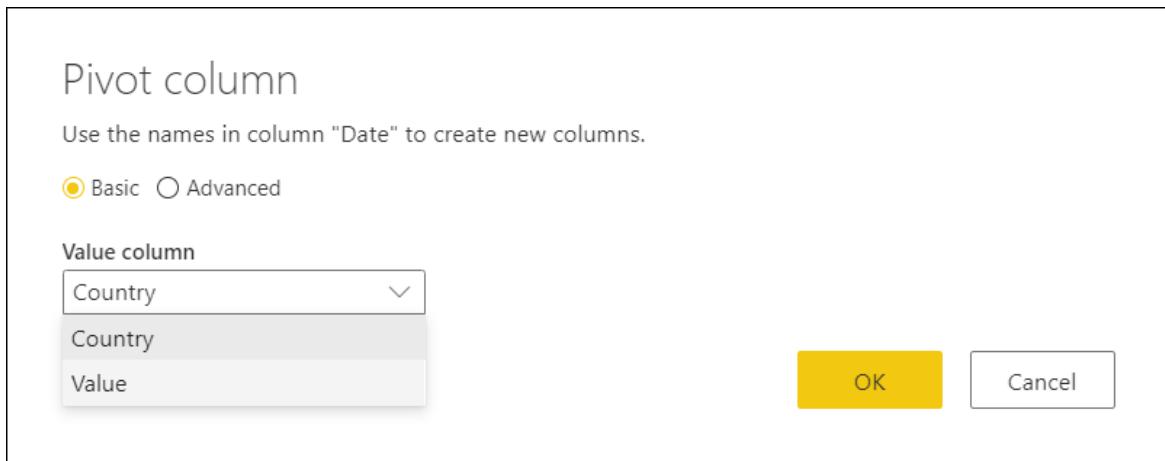
1. Select the column that you want to pivot.
2. On the **Transform** tab in the **Any column** group, select **Pivot column**.



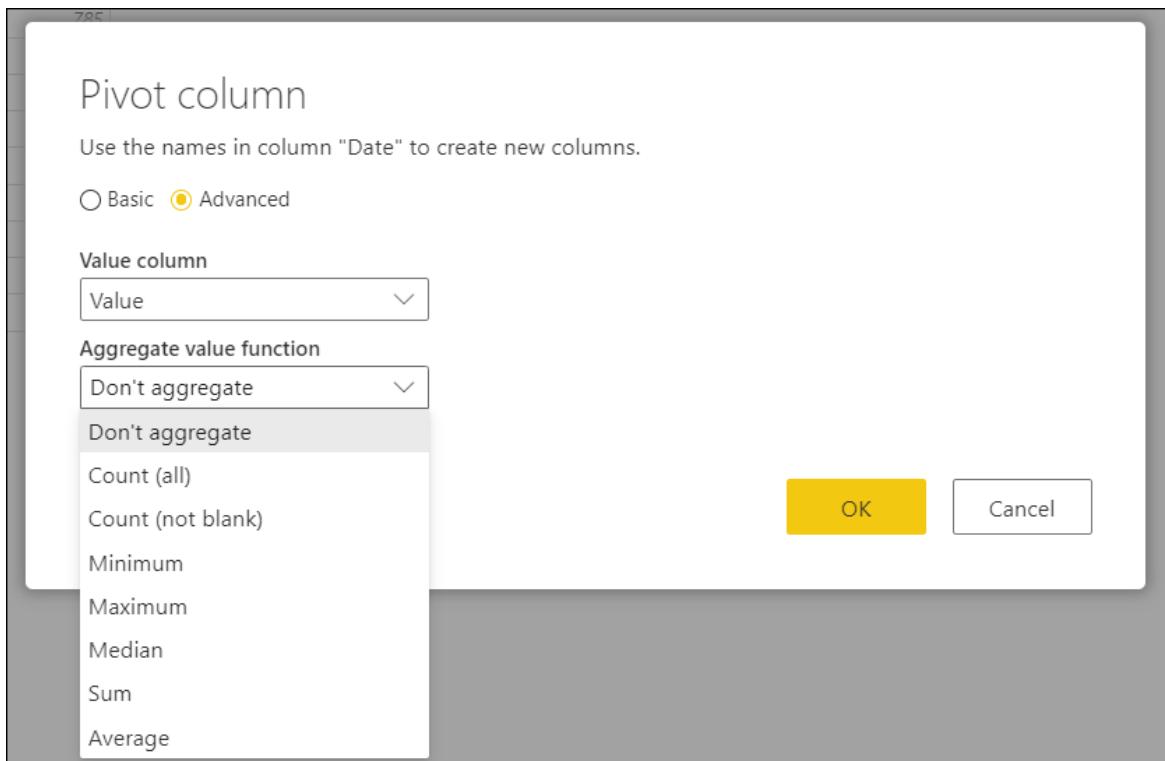
The screenshot shows the Microsoft Power Query ribbon with the 'Transform' tab selected. In the 'Any column' group, the 'Pivot column' button is highlighted with a red box. Below the ribbon, the 'Queries' pane shows 'Query1'. The main area displays a table with three columns: 'Country', 'Date', and 'Value'. The data is as follows:

	A ^B _C Country	Date	Value
1	USA	6/1/2020	785
2	USA	7/1/2020	450
3	USA	8/1/2020	567
4	Canada	6/1/2020	357
5	Canada	7/1/2020	421
6	Canada	8/1/2020	254
7	Panama	6/1/2020	20
8	Panama	7/1/2020	40
9	Panama	8/1/2020	80

3. In the **Pivot column** dialog box, in the **Value column** list, select **Value**.



By default, Power Query will try to do a sum as the aggregation, but you can select the **Advanced** option to see other available aggregations.



The available options are:

- Don't aggregate
- Count (all)
- Count (not blank)
- Minimum
- Maximum
- Median
- Sum
- Average

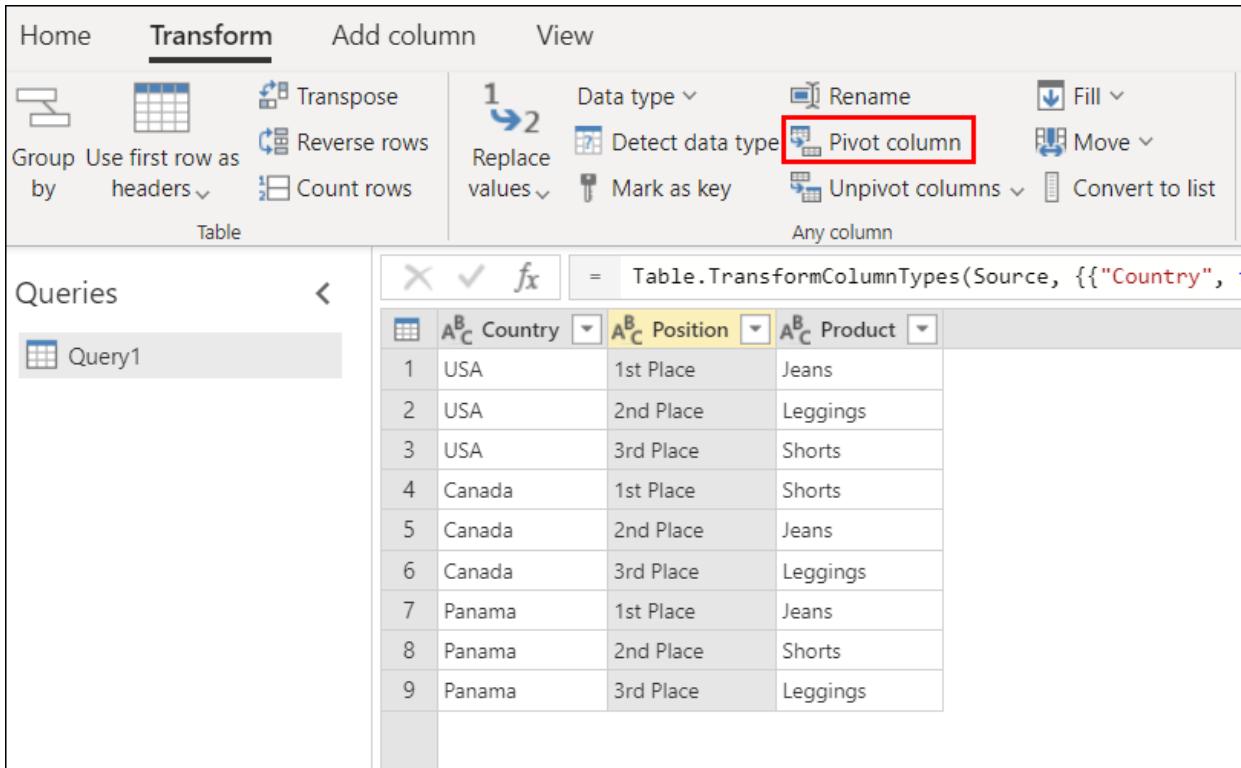
Pivoting columns that can't be aggregated

You can pivot columns without aggregating when you're working with columns that can't be aggregated, or aggregation isn't required for what you're trying to do. For example, imagine a table like the following image, that has **Country**, **Position**, and **Product** as fields.

	A ^B _C Country	A ^B _C Position	A ^B _C Product
1	USA	1st Place	Jeans
2	USA	2nd Place	Leggings
3	USA	3rd Place	Shorts
4	Canada	1st Place	Shorts
5	Canada	2nd Place	Jeans
6	Canada	3rd Place	Leggings
7	Panama	1st Place	Jeans
8	Panama	2nd Place	Shorts
9	Panama	3rd Place	Leggings

Table with Country column containing USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Position column contains First Place in the first, fourth, and seventh rows, Second Place in the second, fifth, and eighth rows, and third Place in the third, sixth, and ninth rows.

Let's say you want to pivot the **Position** column in this table so you can have its values as new columns. For the values of these new columns, you'll use the values from the **Product** column. Select the **Position** column, and then select **Pivot column** to pivot that column.



The screenshot shows the Power BI Transform ribbon with the "Transform" tab selected. In the ribbon, there is a "Pivot column" button, which is highlighted with a red box. Below the ribbon, the Power Query Editor interface is visible, showing a table with the same data as the one above. The table has three columns: "A^B_C Country", "A^B_C Position", and "A^B_C Product". The "Position" column is currently selected. The formula bar at the top of the editor shows the formula: `= Table.TransformColumnTypes(Source, {"Country", "Position", "Product"}))`.

In the **Pivot column** dialog box, select the **Product** column as the value column. Select the **Advanced** option button in the **Pivot columns** dialog box, and then select **Don't aggregate**.

Pivot column

Use the names in column "Position" to create new columns.

Basic Advanced

Value column

Product 

Aggregate value function

Don't aggregate 

[Learn more about Pivot column](#)

OK

Cancel

The result of this operation will yield the result shown in the following image.

	A ^B Country	A ^B 1st Place	A ^B 2nd Place	A ^B 3rd Place
1	Canada	Shorts	Jeans	Leggings
2	Panama	Jeans	Shorts	Leggings
3	USA	Jeans	Leggings	Shorts

Table containing Country, First Place, second Place, and Third Place columns, with the Country column containing Canada in row 1, Panama in row 2, and USA in row 3.

Errors when using the Don't aggregate option

The way the **Don't aggregate** option works is that it grabs a single value for the pivot operation to be placed as the value for the intersection of the column and row pair. For example, let's say you have a table like the one in the following image.

	A ^B Country	Date	1 ² 3 Value
1	USA	6/1/2020	785
2	USA	6/1/2020	450
3	USA	6/1/2020	567
4	Canada	6/1/2020	357
5	Canada	6/1/2020	421
6	Canada	6/1/2020	254
7	Panama	6/1/2020	20
8	Panama	6/1/2020	40
9	Panama	6/1/2020	80

Table with a Country, Date, and Value columns. The Country column contains USA in the first three rows, Canada in the next three rows, and Panama in the last three rows. The Date column contains a date of 6/1/2020 in all rows. The value column contains various whole numbers between 20 and 785.

You want to pivot that table by using the Date column, and you want to use the values from the Value column. Because this pivot would make your table have just the Country values on rows and the Dates as columns, you'd get an error for every single cell value because there are multiple rows for every combination of Country and Date. The outcome of the **Pivot column** operation will yield the results shown in the following image.

	A	B	C	Country	T	2	3	6/1/2020	▼
1	Canada			[Error]					
2	Panama			[Error]					
3	USA			[Error]					

Power Query Editor pane showing a table with Country and 6/1/2020 columns. The Country column contains Canada in the first row, Panama in the second row, and USA in the third row. All of the rows under the 6/1/2020 column contain Errors. Under the table is another pane that shows the expression error with the "There are too many elements in the enumeration to complete the operation" message.

Notice the error message "Expression.Error: There were too many elements in the enumeration to complete the operation." This error occurs because the **Don't aggregate** operation only expects a single value for the country and date combination.

Transpose a table

1/15/2022 • 2 minutes to read • [Edit Online](#)

The transpose table operation in Power Query rotates your table 90 degrees, turning your rows into columns and your columns into rows.

Imagine a table like the one in the following image, with three rows and four columns.

	A ^B Column1	A ^B Column2	A ^B Column3	A ^B Column4
1	Events	Event 1	Event 2	Event 2
2	Participants	150	450	1250
3	Funds	4000	10000	15000

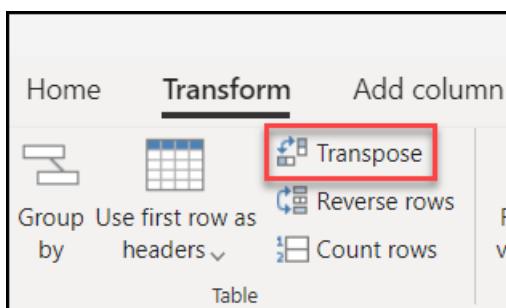
Table with four columns named Column1 through Column4, with all columns set to the Text data type. Column1 contains Events in row 1, Participants in row 2, and Funds in row 3. Column2 contains Event 1 in row 1, 150 in row 2, and 4000 in row 3. Column3 contains Event 2 in row 1, 450 in row 2, and 10000 in row 3. Column4 contains Event 2 in row 1, 1250 in row 2, and 15000 in row 3.

The goal of this example is to transpose that table so you end up with four rows and three columns.

	A ^B Events	1 ² 3 Participants	1 ² 3 Funds
1	Event 1	150	4000
2	Event 2	450	10000
3	Event 2	1250	15000

Table with three columns named Events with a Text data type, Participants with a Whole number data type, and Funds with a whole number data type. The Events column contains, from top to bottom, Event 1, Event 2, and Event 3. The Participants column contains, from top to bottom, 150, 450, and 1250. The Funds column contains, from top to bottom, 4000, 10000, and 15000.

On the **Transform** tab in the ribbon, select **Transpose**.



The result of that operation will look like the following image.

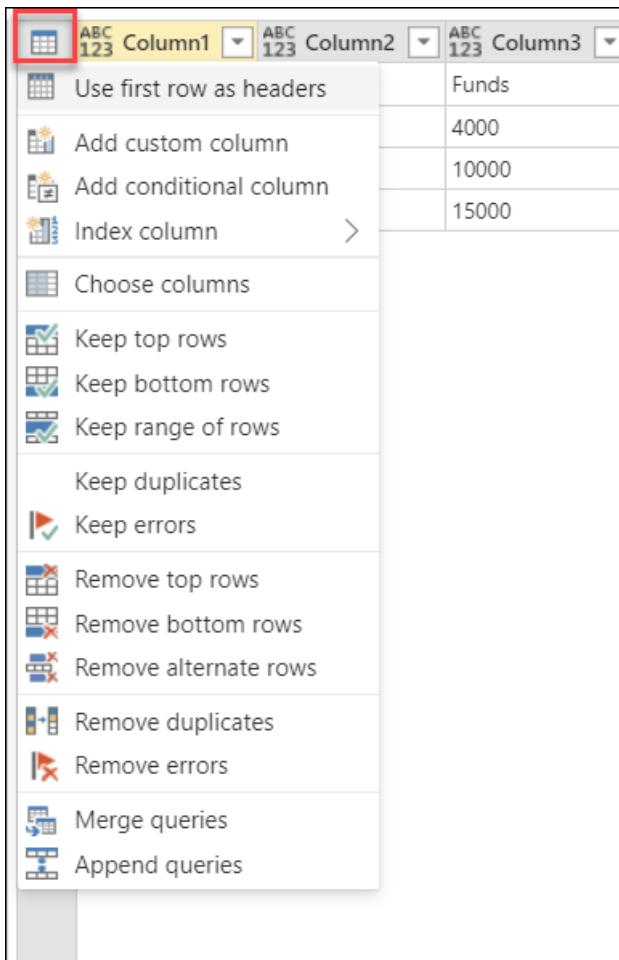
	A ^B Column1	A ^B Column2	A ^B Column3
1	Events	Participants	Funds
2	Event 1	150	4000
3	Event 2	450	10000
4	Event 2	1250	15000

Table with three columns named Column1, Column2, and Column 3, with all columns set to the Any data type. Column1 contains, from top to bottom, Events, Event 1, Event 2, and Event 3. Column2 contains, from top to bottom, Participants, 150, 450, and 1250. Column 3 contains, from top to bottom, Funds, 4000, 10000, and 15000.

NOTE

Only the contents of the table will be transposed during the transpose operation; the column headers of the initial table will be lost. The new columns will have the name **Column** followed by a sequential number.

The headers you need in this example are in the first row of the table. To promote the first row to headers, select the table icon in the upper-left corner of the data preview, and then select **Use first row as headers**.



The result of that operation will give you the output that you're looking for.

	ABC 123 Events	ABC 123 Participants	ABC 123 Funds
1	Event 1	150	4000
2	Event 2	450	10000
3	Event 3	1250	15000

Final table with three columns named Events with a Text data type, Participants with a Whole number data type, and Funds with a whole number data type. The Events column contains, from top to bottom, Event 1, Event 2, and Event 3. The Participants column contains, from top to bottom, 150, 450, and 1250. The Funds column contains, from top to bottom, 4000, 10000, and 15000.

NOTE

To learn more about the promote headers operation, also known as **Use first row as headers**, go to [Promote or demote column headers](#).

Reverse rows

1/15/2022 • 2 minutes to read • [Edit Online](#)

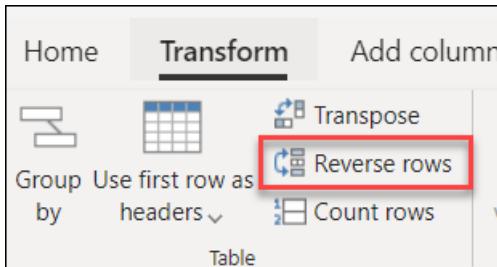
With Power Query, it's possible to reverse the order of rows in a table.

Imagine a table with two columns, **ID** and **Country**, as shown in the following image.

	1 2 3	ID	A B C	Country
1		1		USA
2		2		Canada
3		3		Mexico
4		4		China
5		5		Spain
6		6		Panama
7		7		Colombia

Initial table with ID and Country columns. The ID rows contain, from top to bottom, values of 1 through 7. The Country rows contain, from top to bottom, USA, Canada, Mexico, China, Spain, Panama, and Columbia.

On the **Transform** tab, select **Reverse rows**.



The result of this operation will look as follows.

	1 2 3	ID	A B C	Country
1		7		Colombia
2		6		Panama
3		5		Spain
4		4		China
5		3		Mexico
6		2		Canada
7		1		USA

Output table with the rows reversed. The ID rows now contain, from top to bottom, values of 7 down to 1. The Country rows contain, from top to bottom, Colombia, Panama, Spain, China, Mexico, Canada, and USA.

Data types in Power Query

1/15/2022 • 9 minutes to read • [Edit Online](#)

Data types in Power Query are used to classify values to have a more structured dataset. Data types are defined at the field level—values inside a field are set to *conform* to the data type of the field.

The data type of a column is displayed on the left side of the column heading with an icon that symbolizes the data type.



NOTE

Power Query provides a set of contextual transformations and options based on the data type of the column. For example, when you select a column with a data type of Date, you get transformations and options that apply to that specific data type. These transformations and options occur throughout the Power Query interface, such as on the **Transform** and **Add column** tabs and the smart filter options.

The most common data types used in Power Query are listed in the following table. Although beyond the scope of this article, you can find the complete list of data types in the Power Query M formula language [Types article](#).

DATA TYPE	ICON	DESCRIPTION
Text	A ^B C	A Unicode character data string. Can be strings, numbers, or dates represented in a text format. Maximum string length is 268,435,456 Unicode characters (where each Unicode character is two bytes) or 536,870,912 bytes.
True/False	✗✓	A Boolean value of either True or False.

DATA TYPE	ICON	DESCRIPTION
Decimal number	1.2	<p>Represents a 64-bit (eight-byte) floating-point number. It's the most common number type, and corresponds to numbers as you usually think of them. Although designed to handle numbers with fractional values, it also handles whole numbers. The Decimal Number type can handle negative values from –1.79E +308 through –2.23E –308, 0, and positive values from 2.23E –308 through 1.79E + 308. For example, numbers like 34, 34.01, and 34.000367063 are valid decimal numbers. The largest precision that can be represented in a Decimal Number type is 15 digits long. The decimal separator can occur anywhere in the number. The Decimal Number type corresponds to how Excel stores its numbers. Note that a binary floating-point number can't represent all numbers within its supported range with 100% accuracy. Thus, minor differences in precision might occur when representing certain decimal numbers.</p>
Fixed decimal number	\$	<p>Also known as the Currency type, this data type has a fixed location for the decimal separator. The decimal separator always has four digits to its right and allows for 19 digits of significance. The largest value it can represent is 922,337,203,685,477.5807 (positive or negative). Unlike Decimal Number, the Fixed Decimal Number type is always precise and is thus useful in cases where the imprecision of floating-point notation might introduce errors.</p>
Whole number	1²₃	<p>Represents a 64-bit (eight-byte) integer value. Because it's an integer, it has no digits to the right of the decimal place. It allows for 19 digits; positive or negative whole numbers between –9,223,372,036,854,775,807 ($-2^{63}+1$) and 9,223,372,036,854,775,806 ($2^{63}-2$). It can represent the largest possible precision of the various numeric data types. As with the Fixed Decimal Number type, the Whole Number type can be useful in cases where you need to control rounding.</p>

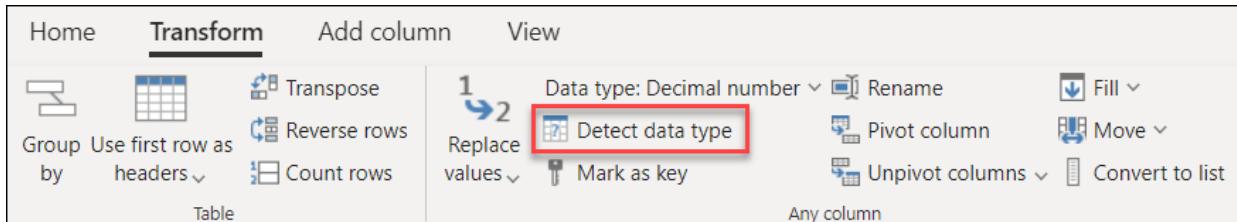
DATA TYPE	ICON	DESCRIPTION
Percentage	%	Fundamentally the same as a Decimal Number type, but it has a mask to format the values in the column as a percentage in the Power Query Editor window.
Date/Time		Represents both a date and time value. Underneath the covers, the Date/Time value is stored as a Decimal Number type, so you can actually convert between the two. The time portion of a date is stored as a fraction to whole multiples of 1/300 seconds (3.33 ms). Dates between the years 1900 and 9999 are supported.
Date		Represents just a date (no time portion). When converted into the model, a Date is the same as a Date/Time value with zero for the fractional value.
Time		Represents just time (no date portion). When converted into the model, a Time value is the same as a Date/Time value with no digits to the left of the decimal place.
Date/Time/Timezone		Represents a UTC Date/Time with a time-zone offset. It's converted into Date/Time when loaded into the model.
Duration		Represents a length of time, which is converted into a Decimal Number type when loaded into the model. As a Decimal Number type, it can be added or subtracted from a Date/Time field with correct results. Because it's a Decimal Number type, you can easily use it in visualizations that show magnitude.
Binary		The Binary data type can be used to represent any other data with a binary format.
Any		The Any data type is the status given to a column that doesn't have an explicit data type definition. Any is the data type that classifies all values. We recommend that you always explicitly define the column data types for your queries from unstructured sources, and avoid having any columns with the Any data type as the output of your query.

Data type detection

Data type detection occurs automatically when connecting to:

- **Structured data sources such as databases**, Power Query reads the table schema from the data source and automatically displays the data by using the correct data type for each column.
- **Unstructured sources such as Excel, CSV, and text files**, Power Query automatically detects data types by inspecting the values in the table. By default, automatic data type detection is enabled in Power Query for unstructured sources.

You can also use the **Detect data type** command in the **Any column** group on the **Transform** tab to automatically detect the data types of the columns in your table.



How to define a column data type

You can define or change the data type of a column in any of four places:

- On the **Home** tab, in the **Transform** group, on the **Data type** drop-down menu.

A screenshot of the 'Power Query - Edit queries' window. The 'Home' tab is selected. In the 'Transform' group, the 'Data type' dropdown menu is open, showing a list of data types including Decimal number, Currency, Whole number, Percentage, Date/Time, Date, Time, Duration, Text, True/False, Binary, and Using locale... A table below shows 14 rows of data with the first column labeled 'ABC' and the second column labeled 'Date'.

- On the **Transform** tab, in the **Any column** group, on the **Data type** drop-down menu.

Home **Transform** Add column View

Group by Use first row as headers Table

Transpose Reverse rows Count rows

1 2 Replace values

Data type ▾

- Rename
- Fill ▾
- Pivot column
- Move ▾
- Unpivot columns ▾
- Convert to list
- Any column

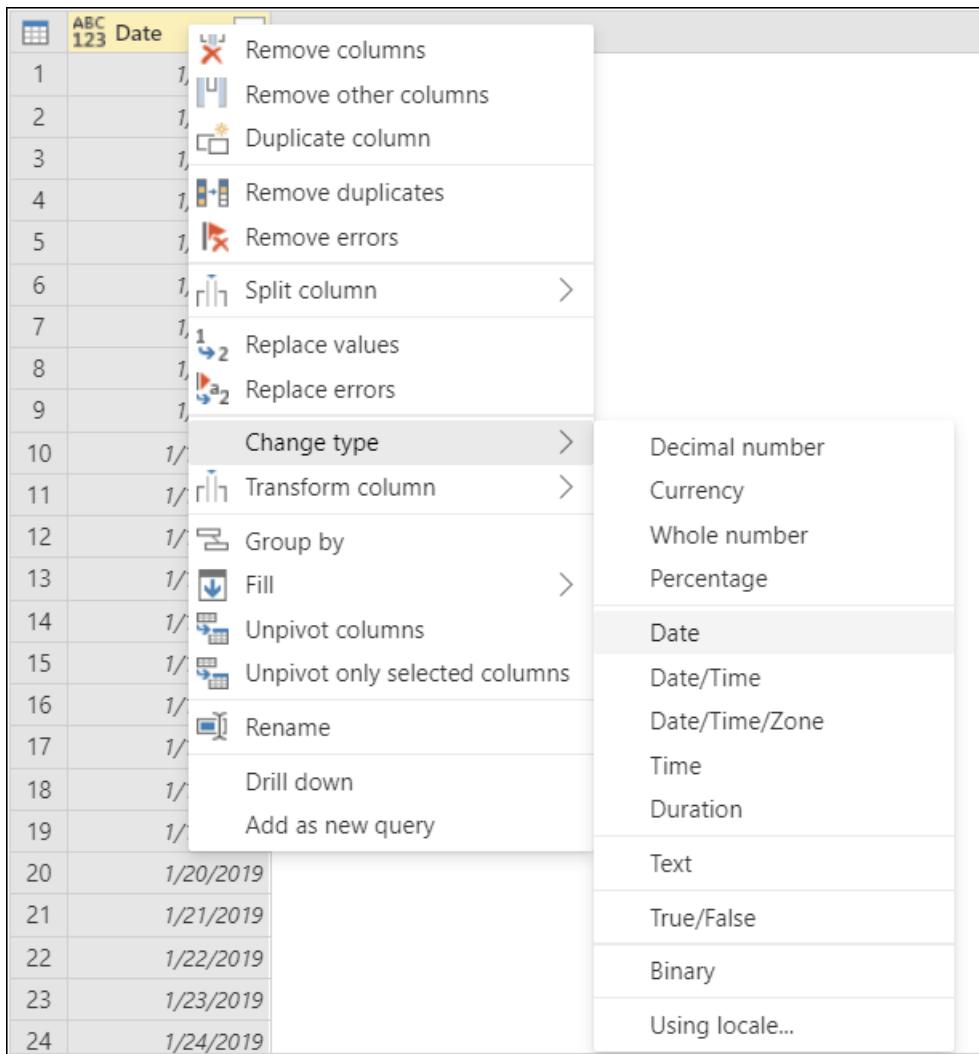
```
table", {{"Column1", "Date"})
```

	ABC 123 Date
1	1/1/2019
2	1/2/2019
3	1/3/2019
4	1/4/2019
5	1/5/2019
6	1/6/2019
7	1/7/2019
8	1/8/2019
9	1/9/2019
10	1/10/2019
11	1/11/2019
12	1/12/2019
13	1/13/2019
14	1/14/2019
15	1/15/2019

- By selecting the icon on the left side of the column heading.

	ABC 123 Date
1	1.2 Decimal number
2	\$ Currency
3	1 ² 3 Whole number
4	% Percentage
5	<input checked="" type="checkbox"/> Date/Time
6	<input checked="" type="checkbox"/> Date
7	<input checked="" type="checkbox"/> Time
8	<input checked="" type="checkbox"/> Date/Time/Zone
9	<input checked="" type="checkbox"/> Duration
10	A ^B C Text
11	<input checked="" type="checkbox"/> True/False
12	<input checked="" type="checkbox"/> Binary
13	Using locale...
14	
15	
16	1/16/2019

- On the column shortcut menu, under **Change Type**.



Automatic detection of column data type and headers

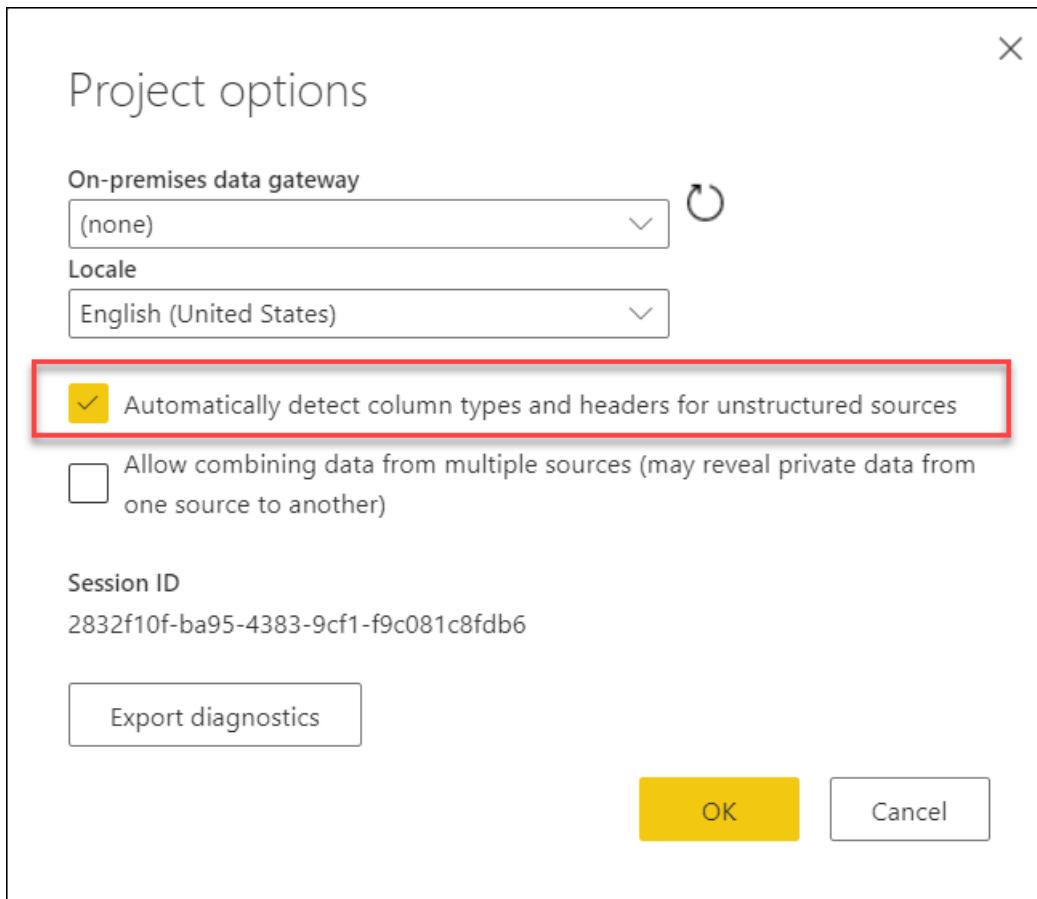
This setting is specifically for unstructured sources. It helps you by automatically inspecting and detecting column types and headers based on the first 200 rows of your table. When this setting is enabled, Power Query automatically adds two steps to your query:

- **Promote column headers:** Promotes the first row of the table to be the column header.
- **Changed type:** Converts the values from the Any data type to a data type based on the inspection of the values from each column.

By default, this setting is enabled. To disable or enable this setting, follow the steps that apply to your Power Query experience.

To configure automatic data type detection in Power Query Online

On the Home tab, select Options, and then select Project options. In the Project options window, select the Automatically detect column types and headers for unstructured sources check box.



To configure automatic data type detection in Power Query for Desktop

You can define this behavior both at the global and per-file level in the **Options** window (in the Power Query Editor, on the **File** tab, select **Options and settings > Options**).

- **Global:** On the left pane under **Global**, select **Data load**. On the right pane under **Type detection**, you can select any of three type detection configurations that will be applied to every new file created in your application:
 - Always detect column types and headers for unstructured sources
 - Detect column types and headers for unstructured sources according to each file's setting
 - Never detect column types and headers for unstructured sources

Options

GLOBAL

- [Data Load](#)
 - [Power Query Editor](#)
 - [DirectQuery](#)
 - [R scripting](#)
 - [Python scripting](#)
 - [Security](#)
 - [Privacy](#)
 - [Regional Settings](#)
 - [Updates](#)
 - [Usage Data](#)
 - [Diagnostics](#)
 - [Preview features](#)
 - [Auto recovery](#)
 - [Report settings](#)
- ### CURRENT FILE
- [Data Load](#)
 - [Regional Settings](#)
 - [Privacy](#)
 - [Auto recovery](#)

Type Detection

- Always detect column types and headers for unstructured sources
- Detect column types and headers for unstructured sources according to each file's setting
- Never detect column types and headers for unstructured sources

Time intelligence

Auto date/time for new files ⓘ [Learn more](#)

Data Cache Management Options ⓘ

Currently used: 688 MB

[Clear Cache](#)

Maximum allowed (MB): ⓘ

[Restore Defaults](#)

Q&A Cache Options ⓘ

Currently used: 384 MB

[Clear Cache](#)

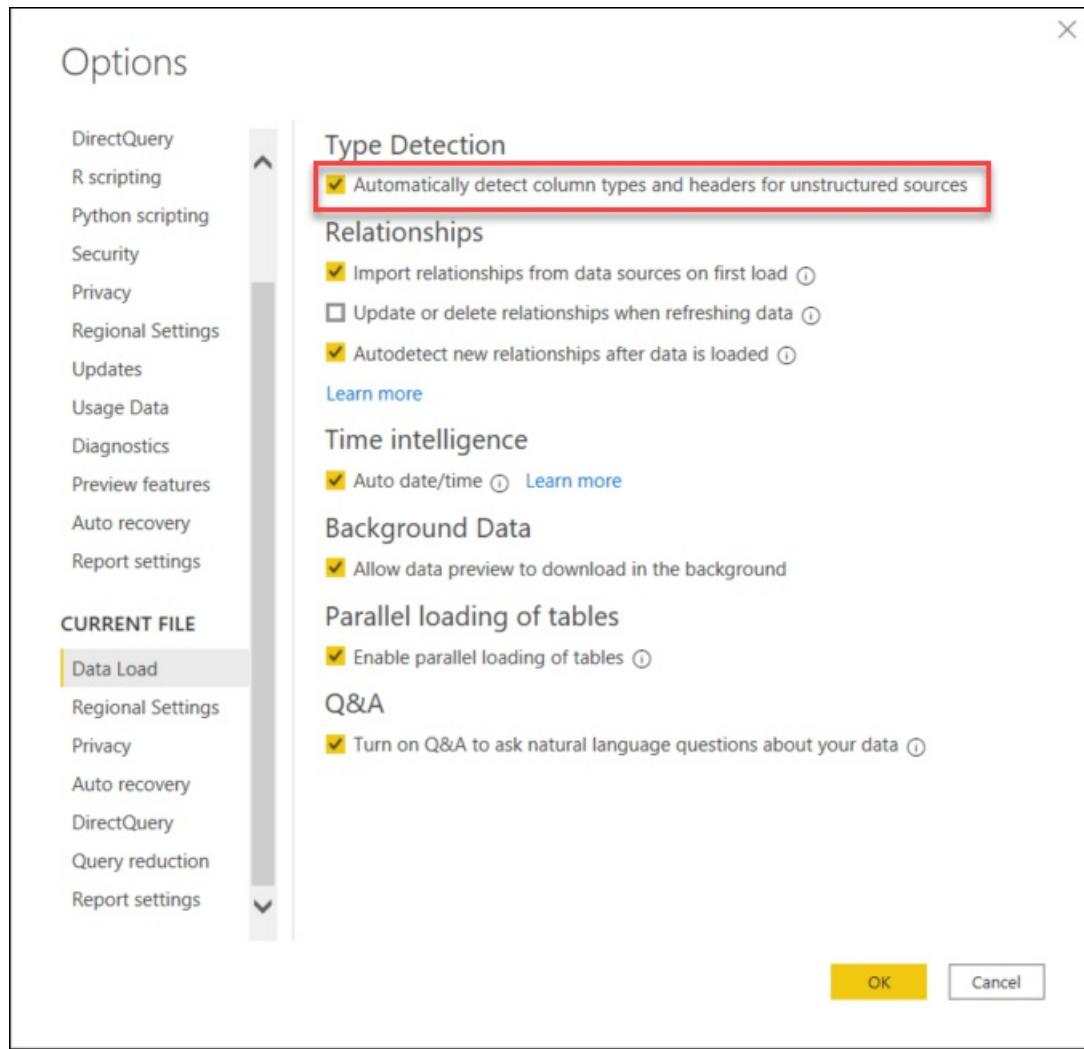
Maximum allowed (MB): ⓘ

[Restore Defaults](#)

[OK](#)

[Cancel](#)

- **Current file:** On the left pane under **Current file**, select **Data load**. On the right pane under **Type detection**, select whether you want to enable or disable type detection for the current file.



Document or project locale

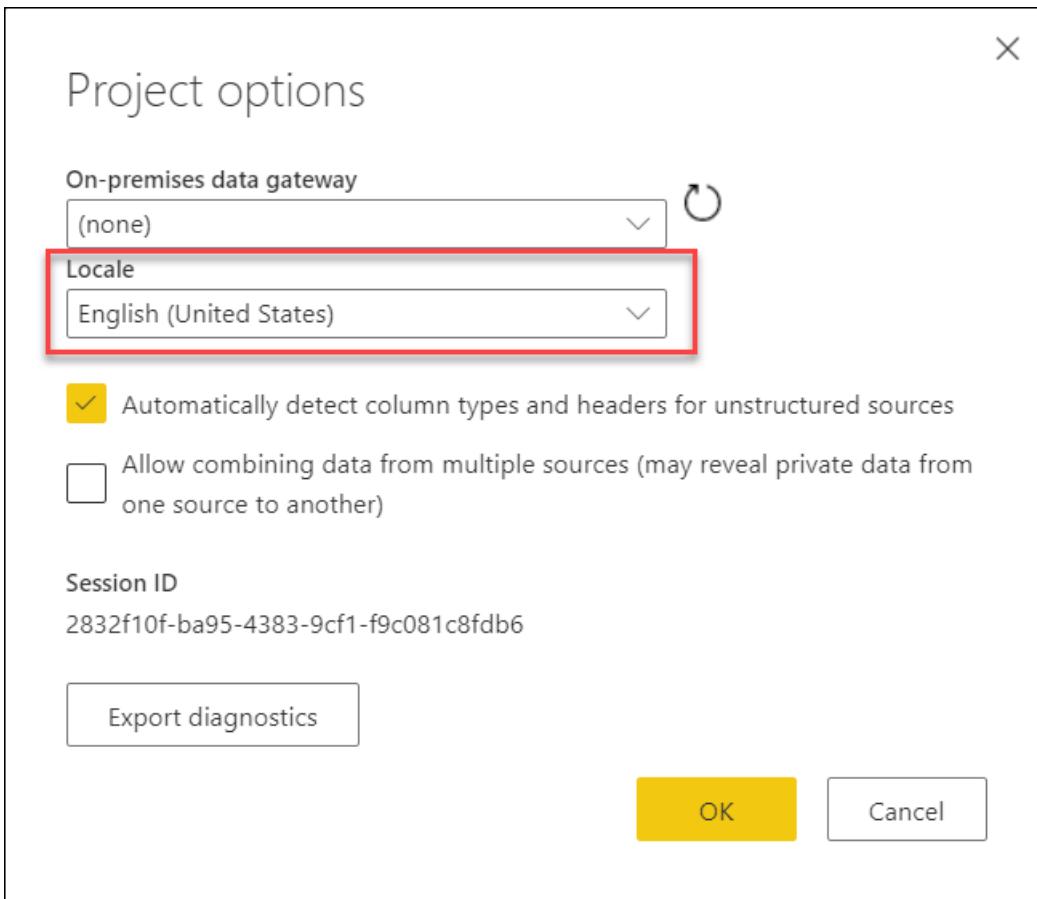
Power Query handles two distinct components that manage the way that things look and are interpreted:

- Localization: the component that tells Power Query in what language it should be displayed.
- Globalization: the component that handles the formatting of the values, in addition to the interpretation of text values.

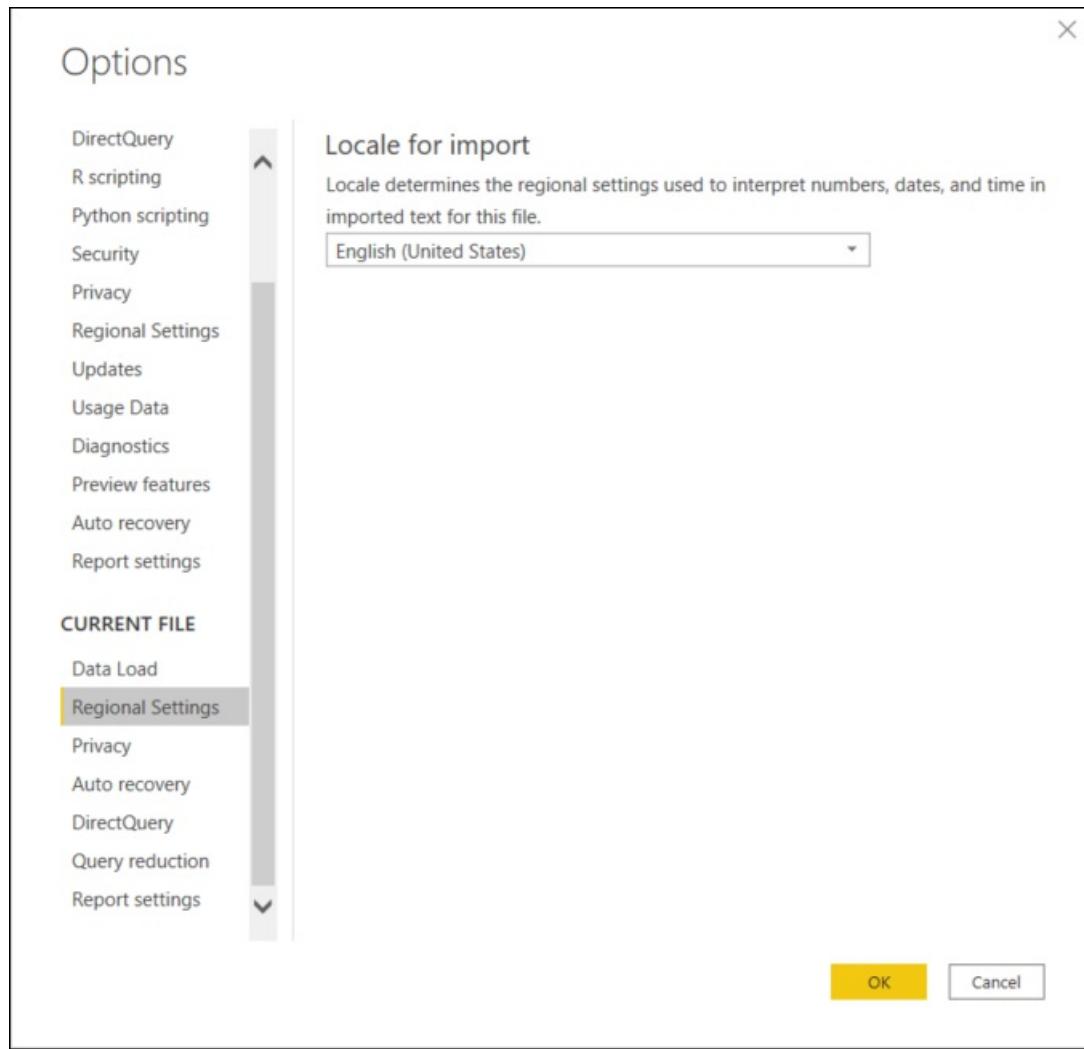
Locale is a single value that holds both the localization and globalization components. Locale is used to interpret text values and convert them into other data types. For example, the locale **English (United States)** means that the *localization* is in United States English and the *globalization*, or format of the value, is based on the standards used in the United States.

When Power Query defines a column data type or converts from one data type to another, it has to interpret the values to be converted before it can transform them to a different data type.

- In Power Query Online, this interpretation is defined in **Project options**, under **Locale**.



- In Power Query for Desktop, Power Query automatically recognizes your operating system regional format and uses that to interpret the values for data type conversion. To override this locale configuration, open the query **Options** window, and in the left pane under **Current file**, select **Regional settings**. From here, you can change the locale to the setting you want.



This locale setting is important for interpreting text values into a specific data type. For example, imagine that you have your locale set as **English (United States)**, but a column in one of your CSV files has dates formatted in the United Kingdom format of day/month/year.

	ABC 123 Date	ABC 123 Units
1	22/01/2020	400
2	23/01/2020	350
3	24/01/2020	375
4	25/01/2020	385

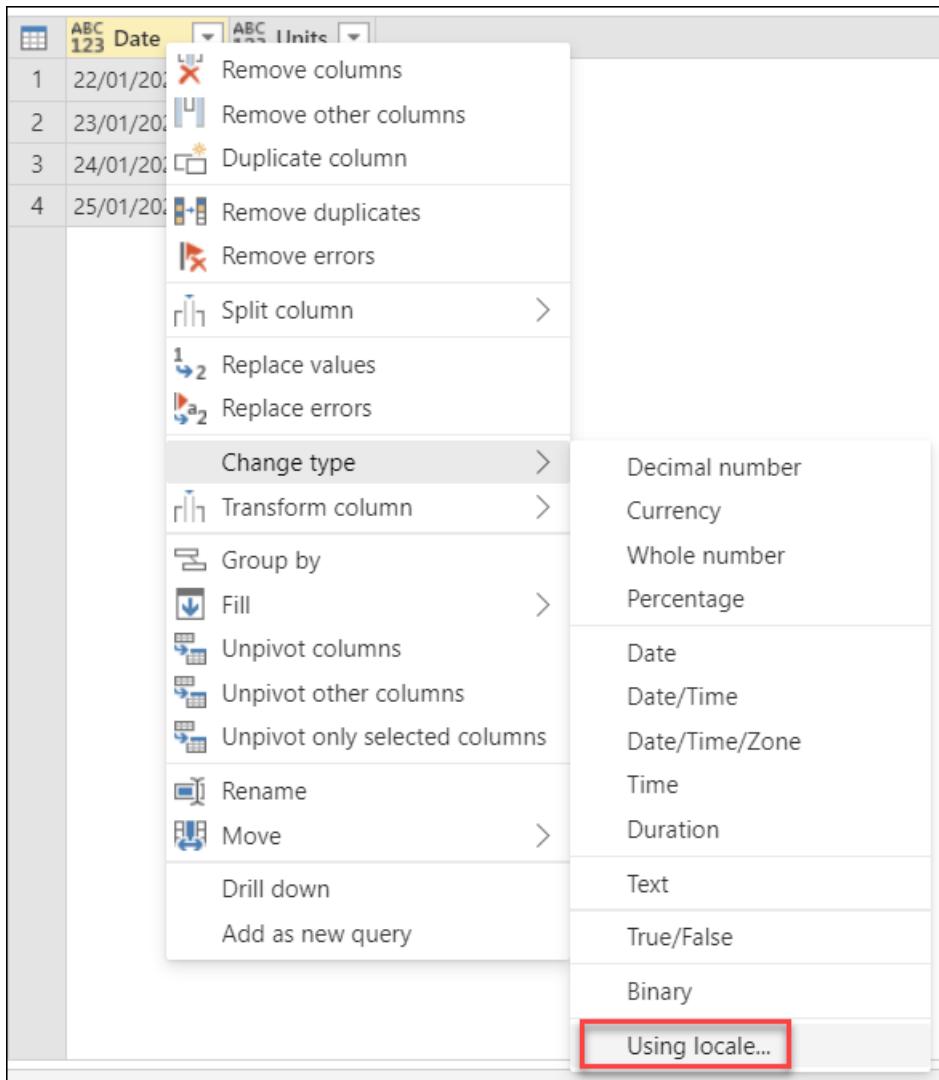
When you try setting the data type of the **Date** column to be **Date**, you get error values.

The screenshot shows the Power BI Data view with four rows of data. The first row has a date value of 22/01/2020. The second row has a date value of 23/01/2020. The third row has a date value of 24/01/2020. The fourth row has a date value of 25/01/2020. A tooltip at the bottom left provides details about the error:

ⓘ DataFormat.Error: We couldn't parse the input provided as a Date value.
Details
22/01/2020
Show details

These errors occur because the locale being used is trying to interpret the date in the English (United States) format, which is month/day/year. Because there's no month 22 in the calendar, it causes an error.

Instead of trying to just select the Date data type, you can right-click the column heading, select **Change type**, and then select **Using locale**.



In the **Change column type with locale** dialog box, you select the data type that you want to set, but you also select which locale to use, which in this case needs to be **English (United Kingdom)**.

Change column type with locale

Change the data type and select the locale of origin.

Data type

Date

Locale

English (United Kingdom)

Sample input values:

29/03/2020

29 March 2020

29 March

March 2020

OK

Cancel

Using this locale, Power Query will be able to interpret values correctly and convert those values to the right data type.

	Date	Units
1	1/22/2020	400
2	1/23/2020	350
3	1/24/2020	375
4	1/25/2020	385

To verify final date values

The formatting of the values is driven by the globalization value. If you have any doubts about the value displayed by Power Query, you can verify the conversion of date values by adding new columns for the day, month, and year from the value. To do this, select the **Date** column and go to the **Add column** tab on the ribbon. In the **Date and time column** group, you'll see the options for a date column.

Power Query - Edit queries

Home Transform Add column View

Conditional column Index column Duplicate column

Custom column Format Parse

Merge columns Extract

Statistics Standard Scientific

Trigonometry Rounding

Date Time Duration

Information

From text From number

Queries < Query

	Date	Units
1	1/22/2020	400
2	1/23/2020	350
3	1/24/2020	375
4	1/25/2020	385

Age Date only Parse

Year Month Quarter Week Day

Subtract days Combine date and time

Earliest Latest

From here, you can extract parts of the date value, such as the year number, the month number, the day number, or even more columns extracted from the **Date** column.

Power Query - Edit queries

Home Transform Add column View

Conditional column Index column Duplicate column

Custom column Format Parse

Merge columns Extract

Statistics Standard Scientific

Trigonometry Rounding

Date Time Duration

Information

From text From number

Queries > < Query

	Date	Units	1.2 Year	1.2 Month	1.2 Day
1	1/22/2020	400	2020	1	22
2	1/23/2020	350	2020	1	23
3	1/24/2020	375	2020	1	24
4	1/25/2020	385	2020	1	25

Query settings

Name: Query

Entity type: Custom

Applied steps:

- Source
- Changed column t...
- Inserted year
- Inserted month
- Inserted day

By using these columns, you can verify that your date value has been converted correctly.

Data type conversion matrix

The following matrix is designed to give you a quick look at the feasibility of data type conversion of a value from one data type to another.

DATA TYPES	1.2	\$	1 ² ₃	%						A ^B _C	
1.2 Decimal number	—										
\$ Currency		—									
1 ² ₃ Whole number			—								
% Percentage				—							
					—						
Date/ Time						—					
						—					
							—				
								—			
Duration									—		
A ^B _C Text									—	—	
										—	—

ICON	DESCRIPTION
	Possible

ICON	DESCRIPTION
	Not possible
	Possible, but it adds values to the original value
	Possible, but it truncates the original value

Dealing with errors in Power Query

1/15/2022 • 6 minutes to read • [Edit Online](#)

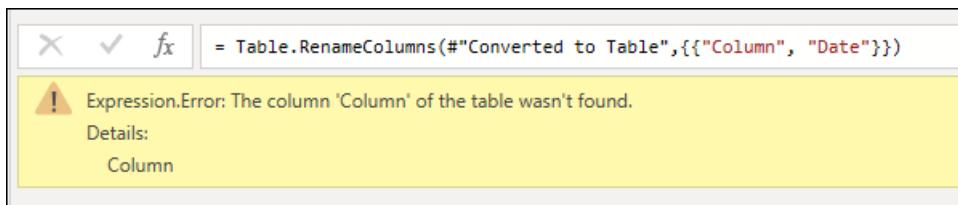
In Power Query, you can encounter two types of errors:

- Step-level errors
- Cell-level errors

This article provides suggestions for how to fix the most common errors you might find at each level, and describes the error reason, error message, and error detail for each.

Step-level error

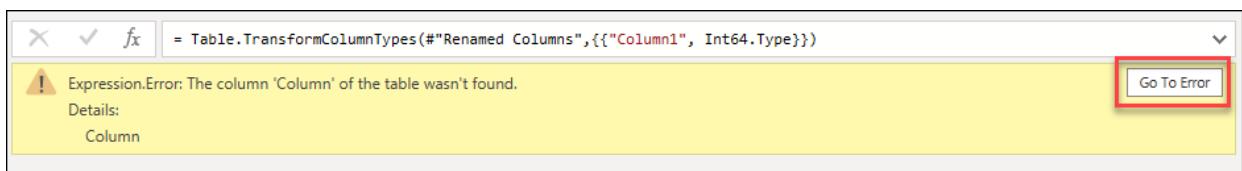
A step-level error prevents the query from loading and displays the error components in a yellow pane.



- **Error reason:** The first section before the colon. In the example above, the error reason is **Expression.Error**.
- **Error message:** The section directly after the reason. In the example above, the error message is **The column 'Column' of the table wasn't found**.
- **Error detail:** The section directly after the **Details:** string. In the example above, the error detail is **Column**.

Common step-level errors

In all cases, we recommend that you take a close look at the error reason, error message, and error detail to understand what's causing the error. You can select the **Go to error** button, if available, to view the first step where the error occurred.



Can't find the source - **DataSource.Error**

This error commonly occurs when the data source is inaccessible by the user, the user doesn't have the correct credentials to access the data source, or the source has been moved to a different place.

Example: You have a query from a text tile that was located in drive D and created by user A. User A shares the query with user B, who doesn't have access to drive D. When this person tries to execute the query, they get a **DataSource.Error** because there's no drive D in their environment.



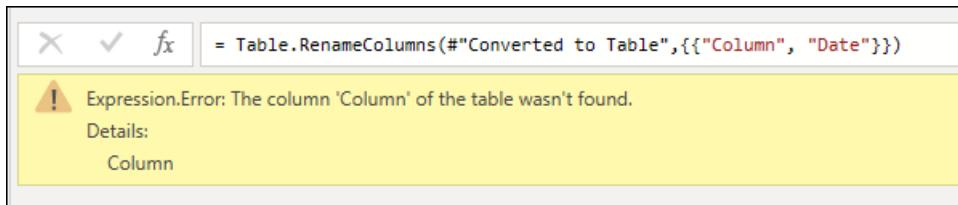
Possible solutions: You can change the file path of the text file to a path that both users have access to. As user B, you can change the file path to be a local copy of the same text file. If the **Edit settings** button is available in

the error pane, you can select it and change the file path.

The column of the table wasn't found

This error is commonly triggered when a step makes a direct reference to a column name that doesn't exist in the query.

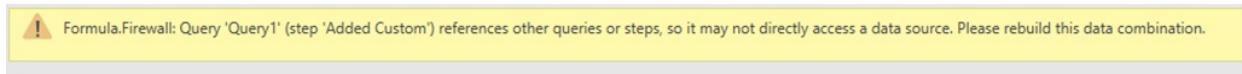
Example: You have a query from a text file where one of the column names was **Column**. In your query, you have a step that renames that column to **Date**. But there was a change in the original text file, and it no longer has a column heading with the name **Column** because it was manually changed to **Date**. Power Query is unable to find a column heading named **Column**, so it can't rename any columns. It displays the error shown in the following image.



Possible solutions: There are multiple solutions for this case, but they all depend on what you'd like to do. For this example, because the correct **Date** column header already comes from your text file, you can just remove the step that renames the column. This will allow your query to run without this error.

Other common step-level errors

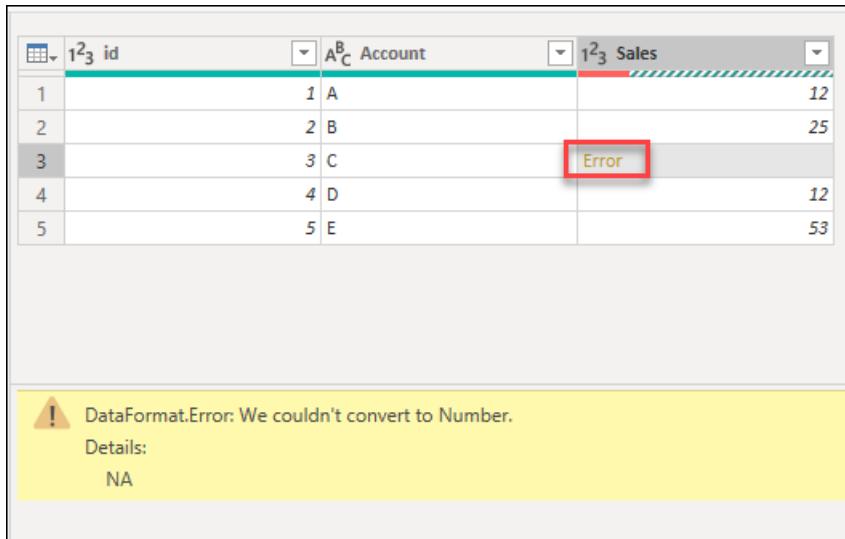
When combining or merging data between multiple data sources, you might get a **Formula.Firewall** error such as the one shown in the following image.



This error can be caused by a number of reasons, such as the data privacy levels between data sources or the way that these data sources are being combined or merged. For more information about how to diagnose this issue, go to [Data privacy firewall](#).

Cell-level error

A cell-level error won't prevent the query from loading, but displays error values as **Error** in the cell. Selecting the white space in the cell displays the error pane underneath the data preview.



NOTE

The data profiling tools can help you more easily identify cell-level errors with the column quality feature. More information: [Data profiling tools](#)

Handling errors at the cell level

When encountering any cell-level errors, Power Query provides a set of functions to handle them either by removing, replacing, or keeping the errors.

For the next sections, the provided examples will be using the same sample query as the start point. In this query, you have a **Sales** column that has one cell with an error caused by a conversion error. The value inside that cell was **NA**, but when you transformed that column to a whole number Power Query couldn't convert **NA** to a number, so it displays the following error.

id	Account	Sales
1	A	12
2	B	25
3	C	Error
4	D	12
5	E	53

Remove errors

To remove rows with errors in Power Query, first select the column that contains errors. On the **Home** tab, in the **Reduce rows** group, select **Remove rows**. From the drop-down menu, select **Remove errors**.

The screenshot shows the Power Query Editor interface with the 'Reduce rows' dropdown menu open. The 'Remove Errors' option is highlighted with a red box. The table below shows the original data with an error in the Sales column for row 3. After applying the 'Remove Errors' operation, the error row is removed, resulting in 4 rows of valid data.

id	Account	Sales
1	A	12
2	B	25
3	C	Error
4	D	12
5	E	53

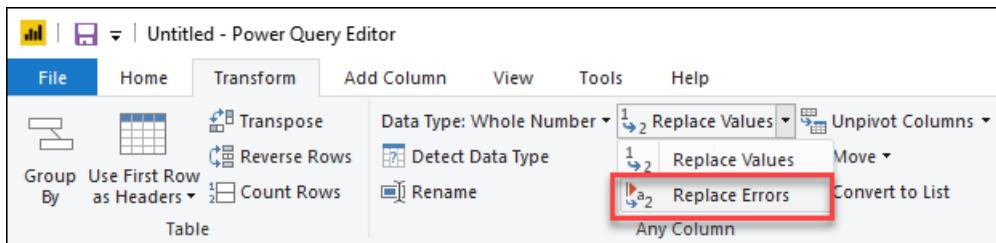
The result of that operation will give you the table that you're looking for.

id	Account	Sales
1	A	12
2	B	25
4	D	12
5	E	53

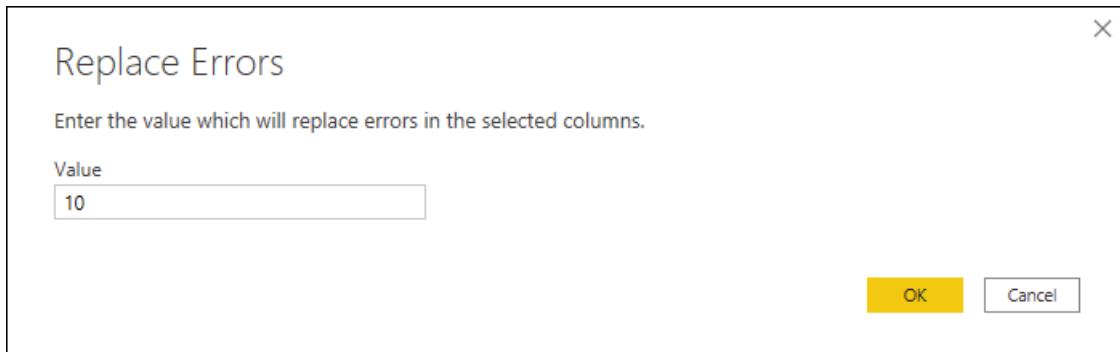
Replace errors

If instead of removing rows with errors, you want to replace the errors with a fixed value, you can do so as well.

To replace rows that have errors, first select the column that contains errors. On the **Transform** tab, in the **Any column** group, select **Replace values**. From the drop-down menu, select **Replace errors**.



In the **Replace errors** dialog box, enter the value **10** because you want to replace all errors with the value 10.

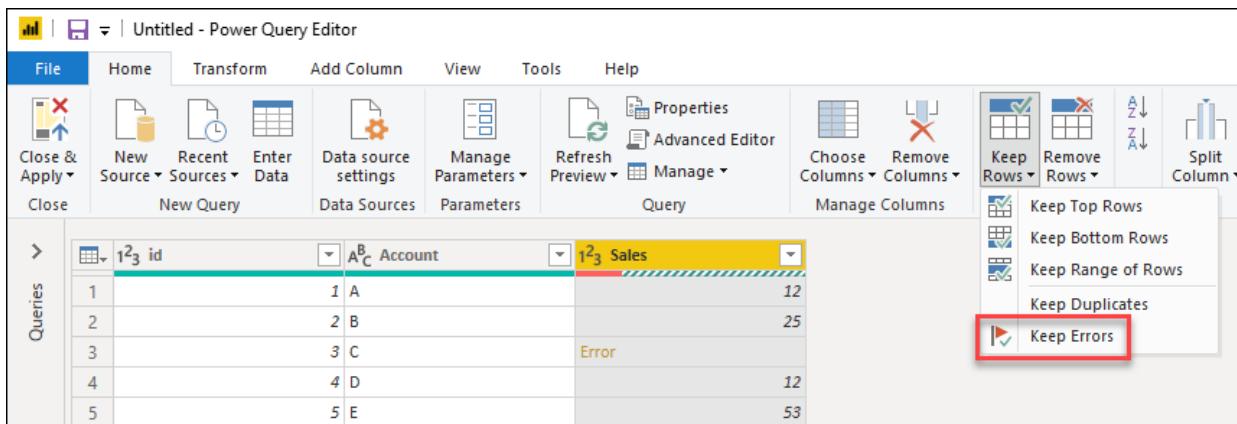


The result of that operation will give you the table that you're looking for.

	id	Account	Sales
1	1	A	12
2	2	B	25
3	3	C	10
4	4	D	12
5	5	E	53

Keep errors

Power Query can serve as a good auditing tool to identify any rows with errors even if you don't fix the errors. This is where **Keep errors** can be helpful. To keep rows that have errors, first select the column that contains errors. On the **Home** tab, in the **Reduce rows** group, select **Keep rows**. From the drop-down menu, select **Keep errors**.



The result of that operation will give you the table that you're looking for.

	id	Account	Sales
1	1	A	12
2	2	B	25
3	3	C	Error
4	4	D	12
5	5	E	53

Common cell-level errors

As with any step-level error, we recommend that you take a close look at the error reasons, error messages, and

error details provided at the cell level to understand what's causing the errors. The following sections discuss some of the most frequent cell-level errors in Power Query.

Data type conversion errors

Commonly triggered when changing the data type of a column in a table. Some values found in the column could not be converted to the desired data type.

Example: You have a query that includes a column named **Sales**. One cell in that column has **NA** as a cell value, while the rest have whole numbers as values. You decide to convert the data type of the column from text to whole number, but the cell with the **NA** value causes an error.

1 ²³ id	A ^B Account	1 ²³ Sales
1	1 A	12
2	2 B	25
3	3 C	Error
4	4 D	12
5	5 E	53

Possible solutions: After identifying the row with the error, you can either modify the data source to reflect the correct value rather than **NA**, or you can apply a **Replace error** operation to provide a value for any **NA** values that cause an error.

Operation errors

When trying to apply an operation that isn't supported, such as multiplying a text value by a numeric value, an error occurs.

Example: You want to create a custom column for your query by creating a text string that contains the phrase "Total Sales: " concatenated with the value from the **Sales** column. An error occurs because the concatenation operation only supports text columns and not numeric ones.

1 ²³ id	A ^B Account	1 ²³ Sales	ABC 123 New Label
1	1 A	12	Error
2	2 B	25	Error
3	4 D	12	Error
4	5 E	53	Error

Possible solutions: Before creating this custom column, change the data type of the **Sales** column to be text.

123 id	A ^B _C Account	A ^B _C Sales	ABC 123 New Label
1	1 A	12	Total Sales is: 12
2	2 B	25	Total Sales is: 25
3	4 D	12	Total Sales is: 12
4	5 E	53	Total Sales is: 53

Working with duplicate values

1/15/2022 • 2 minutes to read • [Edit Online](#)

You can work with duplicate sets of values through transformations that can remove duplicates from your data or filter your data to show duplicates only, so you can focus on them.

WARNING

Power Query is case-sensitive. When working with duplicate values, Power Query considers the case of the text, which might lead to undesired results. As a workaround, users can apply an uppercase or lowercase transform prior to removing duplicates.

For this article, the examples use the following table with **id**, **Category**, and **Total** columns.

	1 ² ₃	id	A ^B _C	Category	1 ² ₃	Total
1		1	A			20
2		2	B			12
3		3	C			15
4		1	A			20
5		5	A			25
6		6	B			32
7		7	C			40
8		7	C			40

Remove duplicates

One of the operations that you can perform is to remove duplicate values from your table.

1. Select the columns that contain duplicate values.
2. Go to the **Home** tab.
3. In the **Reduce rows** group, select **Remove rows**.
4. From the drop-down menu, select **Remove duplicates**.

The screenshot shows the Power Query ribbon with the 'Home' tab selected. In the 'Reduce rows' group, the 'Remove rows' button is highlighted. A dropdown menu is open, showing several options: 'Remove top rows', 'Remove bottom rows', 'Remove alternate rows', 'Remove duplicates' (which is highlighted with a red box), 'Remove blank rows', and 'Remove errors'.

	1 ² ₃	id	A ^B _C	Category	1 ² ₃	Total
1		1	A			20
2		2	B			12
3		3	C			15
4		1	A			20
5		5	A			25
6		6	B			32
7		7	C			40
8		7	C			40

WARNING

There's no guarantee that the first instance in a set of duplicates will be chosen when duplicates are removed.

Remove duplicates from multiple columns

In this example, you want to identify and remove the duplicates by using all of the columns from your table.

	123 id	A B C Category	123 Total
1	1	A	20
2	2	B	12
3	3	C	15
4	1	A	20
5	5	A	25
6	6	B	32
7	7	C	40
8	7	C	40

You have four rows that are duplicates. Your goal is to remove those duplicate rows so there are only unique rows in your table. Select all columns from your table, and then select **Remove duplicates**.

The result of that operation will give you the table that you're looking for.

	123 id	A B C Category	123 Total
1	1	A	20
2	2	B	12
3	3	C	15
4	5	A	25
5	6	B	32
6	7	C	40

NOTE

This operation can also be performed with a subset of columns.

Remove duplicates from a single column

In this example, you want to identify and remove the duplicates by using only the **Category** column from your table.

	123 id	A B C Category	123 Total
1	1	A	20
2	2	B	12
3	3	C	15
4	1	A	20
5	5	A	25
6	6	B	32
7	7	C	40
8	7	C	40

You want to remove those duplicates and only keep unique values. To remove duplicates from the **Category** column, select it, and then select **Remove duplicates**.

The result of that operation will give you the table that you're looking for.

	1 ² ₃ id	A ^B _C	Category	1 ² ₃ Total	
1		1	A	20	
2		2	B	12	
3		3	C	15	

Keep duplicates

Another operation you can perform with duplicates is to keep only the duplicates found in your table.

1. Select the columns that contain duplicate values.
2. Go to the **Home** tab.
3. In the **Reduce rows** group, select **Keep rows**.
4. From the drop-down menu, select **Keep duplicates**.

The screenshot shows the Power BI desktop interface. The ribbon is at the top with the 'Home' tab selected. In the 'Reduce rows' section of the ribbon, a dropdown menu is open, showing several options: 'Keep top rows', 'Keep bottom rows', 'Keep range of rows', 'Keep duplicates' (which is highlighted with a red box), and 'Keep errors'. Below the ribbon is a 'Queries' pane containing a single query named 'Query'. The main area shows a table with the following data:

	1 ² ₃ id	A ^B _C	Category	1 ² ₃ Total	
1		1	A	20	
2		2	B	12	
3		3	C	15	
4		1	A	20	
5		5	A	25	
6		6	B	32	
7		7	C	40	
8		7	C	40	

Keep duplicates from multiple columns

In this example, you want to identify and keep the duplicates by using all of the columns from your table.

	1 ² ₃ id	A ^B _C	Category	1 ² ₃ Total	
1		1	A	20	
2		2	B	12	
3		3	C	15	
4		1	A	20	
5		5	A	25	
6		6	B	32	
7		7	C	40	
8		7	C	40	

You have four rows that are duplicates. Your goal in this example is to keep only the rows that are duplicated in your table. Select all the columns in your table, and then select **Keep duplicates**.

The result of that operation will give you the table that you're looking for.

	1 ² ₃ id	A ^B _C Category	1 ² ₃ Total
1	1	A	20
2	1	A	20
3	7	C	40
4	7	C	40

Keep duplicates from a single column

In this example, you want to identify and keep the duplicates by using only the **id** column from your table.

	1 ² ₃ id	A ^B _C Category	1 ² ₃ Total
1	1	A	20
2	2	B	12
3	3	C	15
4	1	A	20
5	5	A	25
6	6	B	32
7	7	C	40
8	7	C	40

In this example, you have multiple duplicates and you want to keep only those duplicates from your table. To keep duplicates from the **id** column, select the **id** column, and then select **Keep duplicates**.

The result of that operation will give you the table that you're looking for.

	1 ² ₃ id	A ^B _C Category	1 ² ₃ Total
1	1	A	20
2	1	A	20
3	7	C	40
4	7	C	40

See also

[Data profiling tools](#)

Fill values in a column

1/15/2022 • 2 minutes to read • [Edit Online](#)

You can use fill up and fill down to replace null values with the last non-empty value in a column. For example, imagine the following table where you'd like to fill down in the **Date** column and fill up in the **Comments** column.

	Date	Item	Units	Unit Price	Total	Comments
1	null				null	
2	1/1/2020	Brakes	15	\$4.25	63.75	
3	null	Pedals	24	\$8.99	215.76	
4	null	Wheels	20	\$24.52	490.4	
5	null				null	
6	null		null	Subtotal	769.91	Sales Person: Miguel
7	null		null		null	
8	1/5/2020	Socks	6	9.99	59.94	
9	null	Shorts	4	\$15.25	61	
10	null	Jerseys	3	\$59.99	179.97	
11	null	Road Bike	2	\$1,250.49	2,500.98	
12	null		null		null	
13	null		null	Subtotal	2,741.95	Sales Person: Curt

The outcome that you'd expect is shown in the following image.

	Date	Item	Units	Unit Price	Total	Sales Person
1	1/1/2020	Brakes	15	\$4.25	63.75	Miguel
2	1/1/2020	Pedals	24	\$8.99	215.76	Miguel
3	1/1/2020	Wheels	20	\$24.52	490.4	Miguel
4	1/5/2020	Socks	6	9.99	59.94	Curt
5	1/5/2020	Shorts	4	\$15.25	61	Curt
6	1/5/2020	Jerseys	3	\$59.99	179.97	Curt
7	1/5/2020	Road Bike	2	\$1,250.49	2,500.98	Curt

Fill down

The fill down operation takes a column and traverses through the values in it to fill any null values in the next rows until it finds a new value. This process continues on a row-by-row basis until there are no more values in that column.

In the following example, you want to fill down on the **Date** column. To do that, you can right-click to select the **Date** column, and then select **Fill > Down**.

	Date	Item	Units	Unit Price	Total	Comments
1					null	
2	1/1/2020				63.75	
3					215.76	
4					490.4	
5					null	
6					769.91	Sales Person: Migu...
7					null	
8	1/5/2020				59.94	
9					61	
10					179.97	
11					2,500.98	
12					null	
13					2,741.95	Sales Person: Curt

A context menu is open over the last row (row 13). The menu items are:

- Remove columns
- Remove other columns
- Duplicate column
- Remove duplicates
- Remove errors
- Split column
- Replace values
- Replace errors
- Change type
- Transform column
- Group by
- Fill
 - Down
 - Up
- Unpivot columns
- Unpivot other columns
- Unpivot only selected columns
- Rename
- Move
- Drill down
- Add as new query

The result of that operation will look like the following image.

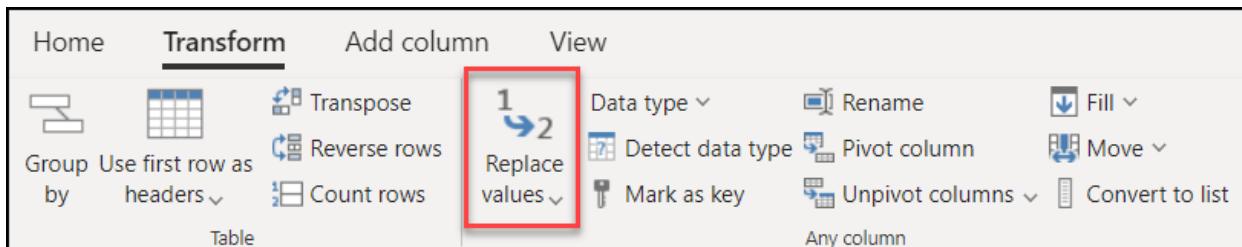
	Date	Item	Units	Unit Price	Total	Comments
1	null		null		null	
2	1/1/2020	Brakes	15	\$4.25	63.75	
3	1/1/2020	Pedals	24	\$8.99	215.76	
4	1/1/2020	Wheels	20	\$24.52	490.4	
5	1/1/2020		null		null	
6	1/1/2020		null	Subtotal	769.91	Sales Person: Migu...
7	1/1/2020		null		null	
8	1/5/2020	Socks	6	9.99	59.94	
9	1/5/2020	Shorts	4	\$15.25	61	
10	1/5/2020	Jerseys	3	\$59.99	179.97	
11	1/5/2020	Road Bike	2	\$1,250.49	2,500.98	
12	1/5/2020		null		null	
13	1/5/2020		null	Subtotal	2,741.95	Sales Person: Curt

Fill up

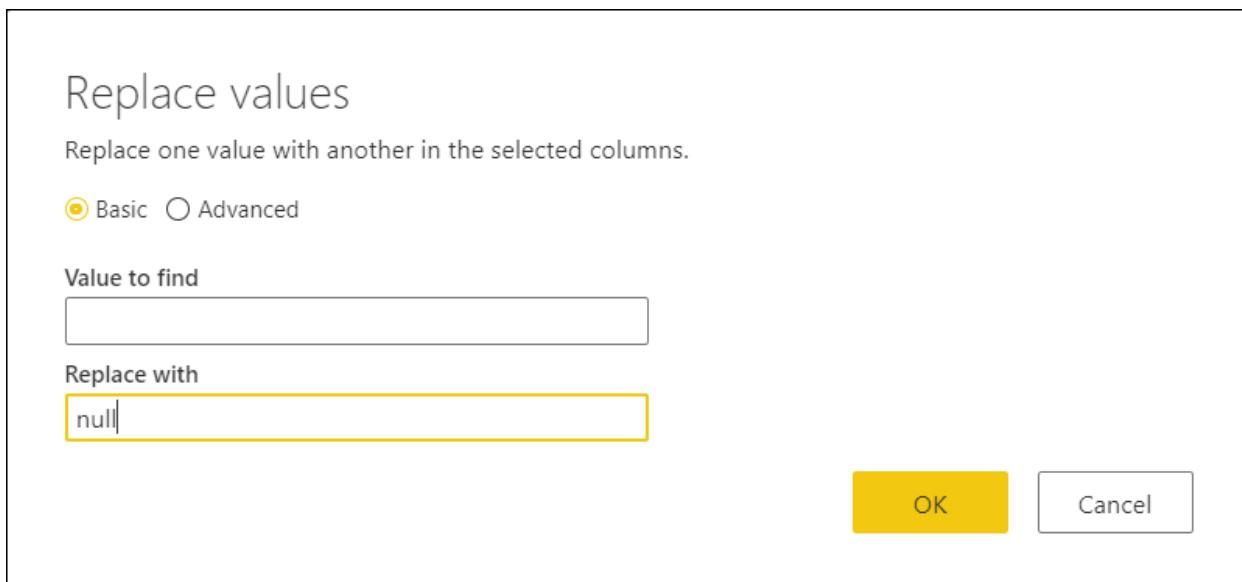
In the same way as the fill down operation, fill up works on a column. But by contrast, fill up finds the last value of the column and fills any null values in the previous rows until it finds a new value. Then the same process occurs for that value. This process continues until there are no more values in that column.

In the following example, you want to fill the **Comments** column from the bottom up. You'll notice that your **Comments** column doesn't have null values. Instead it has what appears to be empty cells. Before you can do the fill up operation, you need to transform those empty cells into null values: select the column, go to the

Transform tab, and then select Replace values.

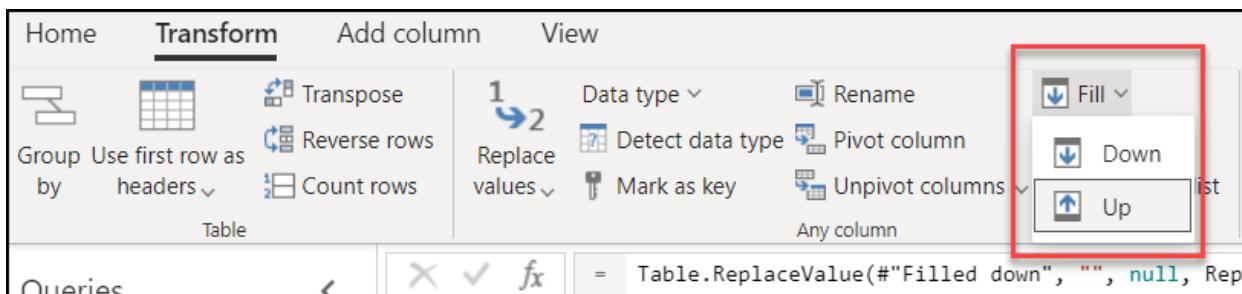


In the Replace values dialog box, leave Value to find blank. For Replace with, enter null.



More information: [Replace values](#)

After all empty cells are replaced with null, select the Comments column, go to the Transform tab, and then select Fill > Up.



The result of that operation will look like the following image.

	Date	Item	Units	Unit Price	Total	Comments
1	null		null		null	Sales Person: Miguel
2	1/1/2020	Brakes	15	\$4.25	63.75	Sales Person: Miguel
3	1/1/2020	Pedals	24	\$8.99	215.76	Sales Person: Miguel
4	1/1/2020	Wheels	20	\$24.52	490.4	Sales Person: Miguel
5	1/1/2020		null		null	Sales Person: Miguel
6	1/1/2020		null	Subtotal	769.91	Sales Person: Miguel
7	1/1/2020		null		null	Sales Person: Curt
8	1/5/2020	Socks	6	9.99	59.94	Sales Person: Curt
9	1/5/2020	Shorts	4	\$15.25	61	Sales Person: Curt
10	1/5/2020	Jerseys	3	\$59.99	179.97	Sales Person: Curt
11	1/5/2020	Road Bike	2	\$1,250.49	2,500.98	Sales Person: Curt
12	1/5/2020		null		null	Sales Person: Curt
13	1/5/2020		null	Subtotal	2,741.95	Sales Person: Curt

Cleaning up your table

1. Filter the **Units** column to show only rows that aren't equal to **null**.

The screenshot shows a table with columns: Date, Item, Units, Unit Price, Total, and Comments. A filter dialog is open over the 'Units' column. The dialog includes sorting options (Sort ascending, Sort descending), a remove empty button, and a number filters section. A search bar is at the top of the dialog. Below it is a list of numerical values from 1 to 24, each with a checkbox. Most checkboxes are checked, except for '(Select all)' and '(null)'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

	Date	Item	Units	Unit Price	Total	Comments
1	null					
2	1/1/2020	Brakes				
3	1/1/2020	Pedals				
4	1/1/2020	Wheels				
5	1/1/2020					
6	1/1/2020					
7	1/1/2020					
8	1/5/2020	Socks				
9	1/5/2020	Shorts				
10	1/5/2020	Jerseys				
11	1/5/2020	Road Bike				
12	1/5/2020					
13	1/5/2020					

2. Rename the **Comments** column as **Sales Person**.

The screenshot shows the same table as above, but the 'Comments' column has been renamed to 'Sales Person'. The new column header is highlighted with a yellow border.

	Date	Item	Units	Unit Price	Total	Sales Person
1	1/1/2020	Brakes	15	\$4.25	63.75	Sales Person: Miguel
2	1/1/2020	Pedals	24	\$8.99	215.76	Sales Person: Miguel
3	1/1/2020	Wheels	20	\$24.52	490.4	Sales Person: Miguel
4	1/5/2020	Socks	6	9.99	59.94	Sales Person: Curt
5	1/5/2020	Shorts	4	\$15.25	61	Sales Person: Curt
6	1/5/2020	Jerseys	3	\$59.99	179.97	Sales Person: Curt
7	1/5/2020	Road Bike	2	\$1,250.49	2,500.98	Sales Person: Curt

3. Remove the **Sales Person:** values from the **Sales Person** column so you only get the names of the salespeople.

Replace values

Replace one value with another in the selected columns.

Basic Advanced

Value to find

Sales Person:

Replace with

OK

Cancel

Now you should have exactly the table you were looking for.

	Date	Item	Units	Unit Price	Total	Sales Person
1	1/1/2020	Brakes	15	\$4.25	63.75	Miguel
2	1/1/2020	Pedals	24	\$8.99	215.76	Miguel
3	1/1/2020	Wheels	20	\$24.52	490.4	Miguel
4	1/5/2020	Socks	6	9.99	59.94	Curt
5	1/5/2020	Shorts	4	\$15.25	61	Curt
6	1/5/2020	Jerseys	3	\$59.99	179.97	Curt
7	1/5/2020	Road Bike	2	\$1,250.49	2,500.98	Curt

See also

[Replace values](#)

Sort columns

1/15/2022 • 2 minutes to read • [Edit Online](#)

You can sort a table in Power Query by one column or multiple columns. For example, take the following table with the columns named **Competition**, **Competitor**, and **Position**.

	A ^B _C Competition	A ^B _C Competitor	1 ² ₃ Position
1	1 - Opening	New York	2
2	3 - Final	South Carolina	1
3	2 - Main	New York	2
4	3 - Final	California	2
5	2 - Main	California	1
6	1 - Opening	Florida	1

Table with Competition, Competitor, and Position columns. The Competition column contains 1 - Opening in rows 1 and 6, 2 - Main in rows 3 and 5, and 3-Final in rows 2 and 4. The Position row contains a value of either 1 or 2 for each of the Competition values.

For this example, the goal is to sort this table by the **Competition** and **Position** fields in ascending order.

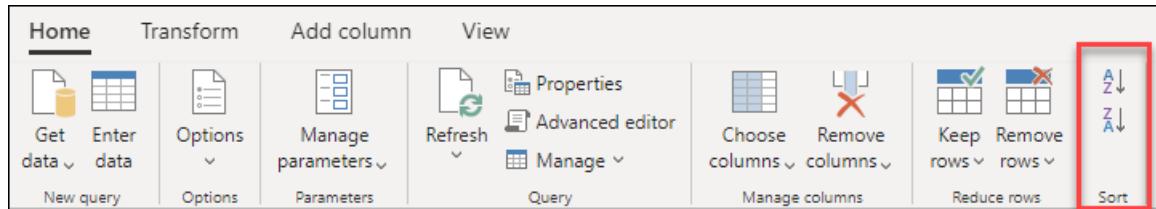
	A ^B _C Competition	1 ² ₃ A ^B _C Competitor	1 ² ₃ Position
1	1 - Opening	Florida	1
2	1 - Opening	New York	2
3	2 - Main	California	1
4	2 - Main	New York	2
5	3 - Final	South Carolina	1
6	3 - Final	California	2

Table with Competition, Competitor, and Position columns. The Competition column contains 1 - Opening in rows 1 and 2, 2 - Main in rows 3 and 4, and 3-Final in rows 5 and 6. The Position row contains, from top to bottom, a value of 1, 2, 1, 2, 1, and 2.

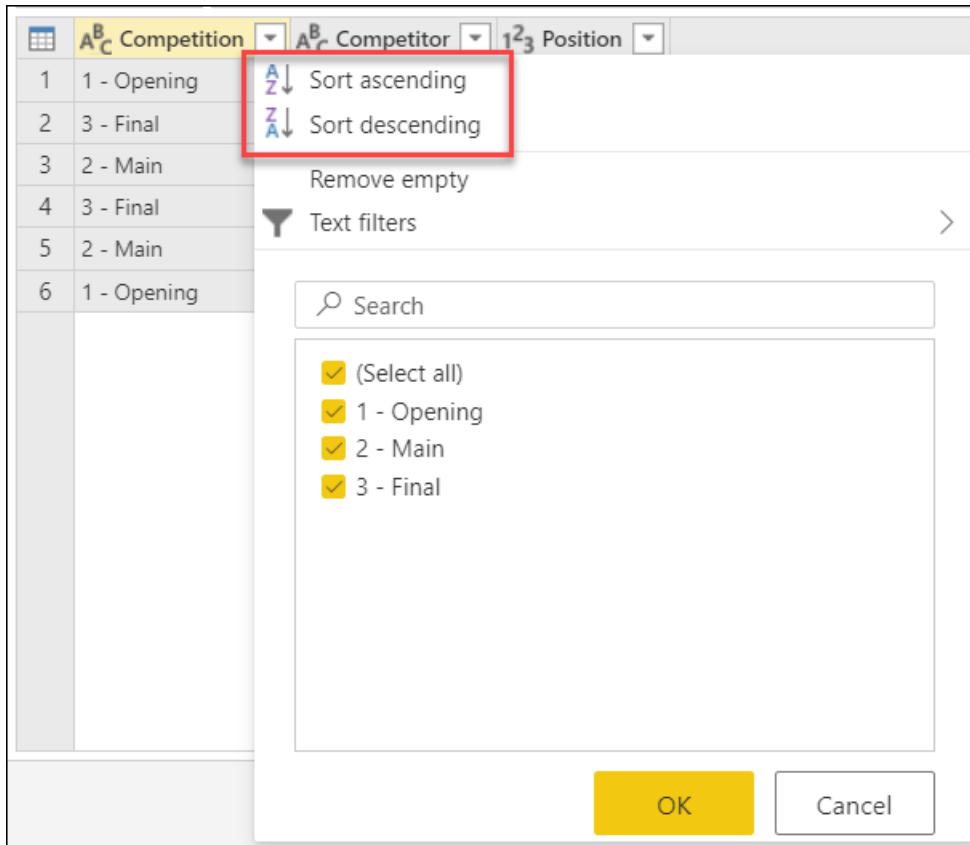
To sort a table by using columns

To sort the table, first select the column to be sorted. After the column has been selected, you can select the sort operation from one of two places:

- On the **Home** tab, in the **Sort** group, there are icons to sort your column in either ascending or descending order.



- From the column heading drop-down menu. Next to the name of the column there's a drop-down menu indicator ▾. When you select the icon, you'll see the option to sort the column.



In this example, first you need to sort the **Competition** column. You'll perform the operation by using the buttons in the **Sort** group on the **Home** tab. This action creates a new step in the **Applied steps** section named **Sorted rows**.

	A ^B _C Competition	A ^B _C Competitor	1 ² ₃ Position
1	1 - Opening	Florida	1
2	1 - Opening	New York	2
3	2 - Main	New York	2
4	2 - Main	California	1
5	3 - Final	California	2
6	3 - Final	South Carolina	1

Query settings >

Name: Table

Entity type: Custom

Applied steps:

- Source
- Changed column t...
- X Sorted rows**

A visual indicator, displayed as an arrow pointing up, gets added to the **Competitor** drop-down menu icon to show that the column is being sorted in ascending order.

Now you'll sort the **Position** field in ascending order as well, but this time you'll use the **Position** column heading drop-down menu.

The screenshot shows the Power BI Query Editor with a table containing six rows of data. The columns are labeled 'Competition', 'Competitor', and 'Position'. The 'Position' column has a dropdown arrow indicating it is sorted. A context menu is open over this column, showing options for sorting (Sort ascending, Sort descending), removing empty rows, applying number filters, and a search bar. Below the search bar, there is a section for selecting the sort order, with checkboxes for '(Select all)', '1', and '2'. At the bottom right of the menu are 'OK' and 'Cancel' buttons.

Notice that this action doesn't create a new **Sorted rows** step, but modifies it to perform both sort operations in one step. When you sort multiple columns, the order that the columns are sorted in is based on the order the columns were selected in. A visual indicator, displayed as a number to the left of the drop-down menu indicator, shows the place each column occupies in the sort order.

The screenshot shows the Power BI Query Editor with the same table structure. The 'Competition' and 'Competitor' columns now have their sort arrows highlighted with red boxes. The 'Position' column's sort arrow is also highlighted with a red box. The data remains the same as the previous screenshot.

	Competition	Competitor	Position
1	1 - Opening	Florida	1
2	1 - Opening	New York	2
3	2 - Main	California	1
4	2 - Main	New York	2
5	3 - Final	South Carolina	1
6	3 - Final	California	2

To clear a sort operation from a column

Do one of the following actions:

- Select the down arrow next to the column heading, and then select **Clear sort**.
- In **Applied steps** on the **Query Settings** pane, delete the **Sorted rows** step.

Rename columns

1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can rename columns to format the dataset in a clear and concise way.

As an example, let's start with a dataset that has two columns.

COLUMN 1	COLUMN 2
Panama	Panama
USA	New York
Canada	Toronto

The column headers are **Column 1** and **Column 2**, but you want to change those names to more friendly names for your columns.

- **Column 1** becomes **Country**
- **Column 2** becomes **City**

The end result that you want in Power Query looks like the following table.



	Country	City
1	Panama	Panama
2	USA	New York
3	Canada	Toronto

How to rename a column

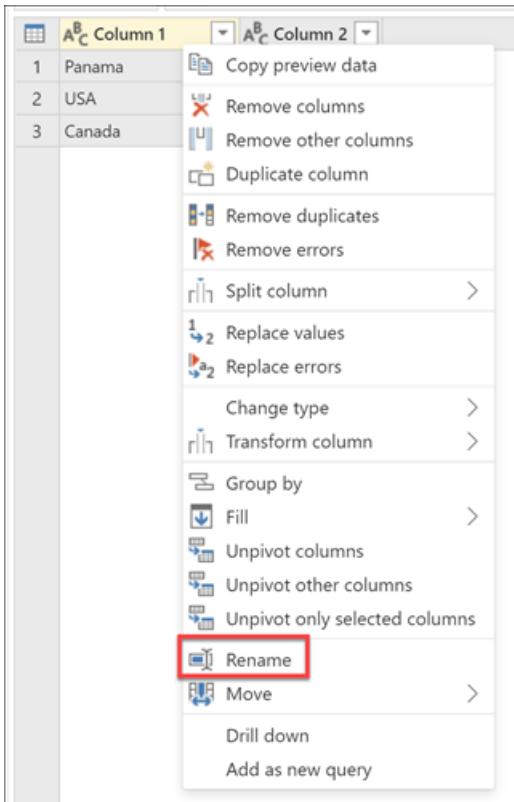
There are three ways to rename a column in Power Query.

- **Double-click the column header:** The double-click action immediately lets you rename the column.

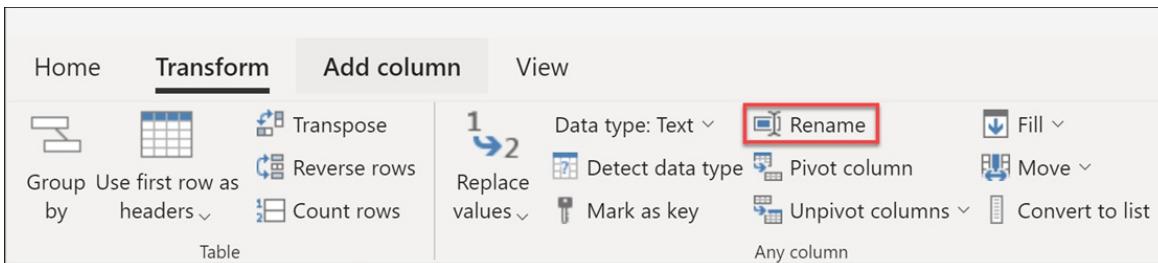


	Column 1	Column 2
1	Panama	Panama
2	USA	New York
3	Canada	Toronto

- **Right-click the column of your choice:** A contextual menu is displayed and you can select the **Rename** option to rename the selected column.



- **Rename option in the Transform tab:** In the **Transform** tab, under the **Any column** group, select the **Rename** option.



Avoiding duplicate column names

Power Query requires table column names to be unique across all columns. This means that if you try to rename a column to a column name that already exists in the table, an error with the message *Column Name Conflict* appears. You'll have to rename the column to something else.

For example, for the first sample table provided in this article, imagine that you try to rename both **Column 1** and **Column 2** to "Geography". An error message pops up that prevents you from renaming the second column to "Geography".

The screenshot shows the Power Query Editor interface. A table with three rows is visible. The first row has a header 'Geography'. A second row has a header 'Column 2'. The third row has a header 'A' and a value 'B'. A modal dialog box is displayed, titled 'Column name conflict'. It contains the message 'A column named "Geography" already exists.' and a yellow 'Close' button.

	Geography	Column 2
1	Panama	Panama
2	USA	New York
3	Canada	Toronto

Column name disambiguation

With many actions performed in Power Query that might result in a *Column Name Conflict*, Power Query tries to disambiguate by renaming all duplicate instances of the same column name. The way that Power Query renames these columns is by adding a suffix to the original column name that has a separator (commonly a dot or an underscore), and then a number that represents the instance of the duplicated column name in the order that it was found by Power Query. This renaming can often happen with actions such as, but not limited to:

- **Promoting your column headers from your first row:** For example, if you tried promoting the first row of the sample table in this article, Power Query renames the columns to **Panama** and **Panama_1**.

The screenshot shows the Power Query Editor with a table. The first row now has two headers: 'Panama' and 'Panama_1'. The second row has values 'USA' and 'New York'. The third row has values 'Canada' and 'Toronto'. The 'Panama' header is highlighted.

	Panama	Panama_1
1	USA	New York
2	Canada	Toronto

NOTE

To learn more about how to promote headers from your first row, go to [Promote or demote column headers](#).

- **Expanding a column with a field name that also exists in the current table:** This can happen, for example, when you perform a merge operation and the column with the merged table has field names that also exist in the table. When you try to expand the fields from that column, Power Query automatically tries to disambiguate to prevent *Column Name Conflict* errors.

	A ^B _C Country	A ^B _C City	1 ² ₃ Sales	Online Sales
1	Panama	Panama	20	[Table]
2	USA	New York	10	[Table]
3	Canada	Toronto	5	[Table]

A ^B _C Country	A ^B _C City	A ^B _C Sales
Panama	Panama	2



	A ^B _C Country	A ^B _C City	1 ² ₃ Sales	A ^B _C Country.1	A ^B _C City.1	A ^B _C Sales.1
1	Panama	Panama	20	Panama	Panama	2
2	USA	New York	10	USA	New York	3
3	Canada	Toronto	5	Canada	Toronto	4

Move columns

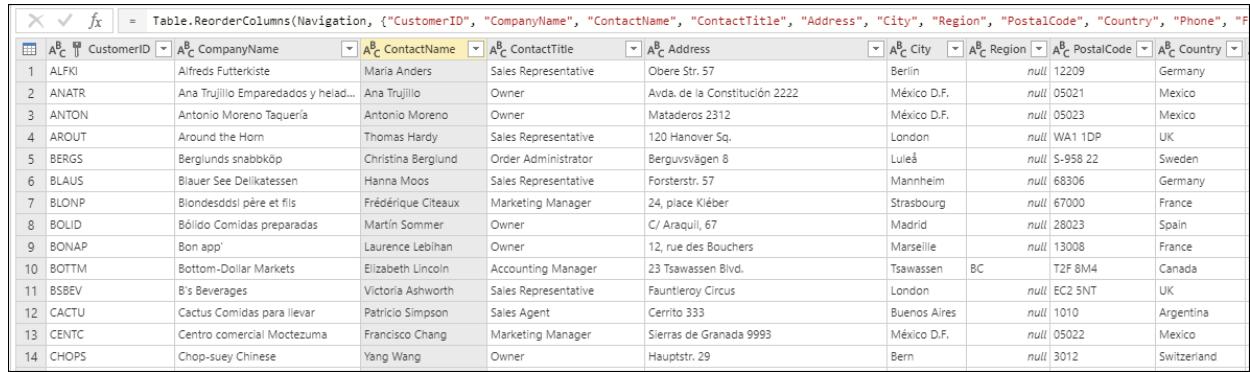
1/15/2022 • 2 minutes to read • [Edit Online](#)

A common process when preparing data is to move columns in the dataset.

To accomplish this move, you can either select the **Move** option or *drag and drop* the column.

Move option

The following example shows the different ways of moving columns. This example focuses on moving the Contact Name column.

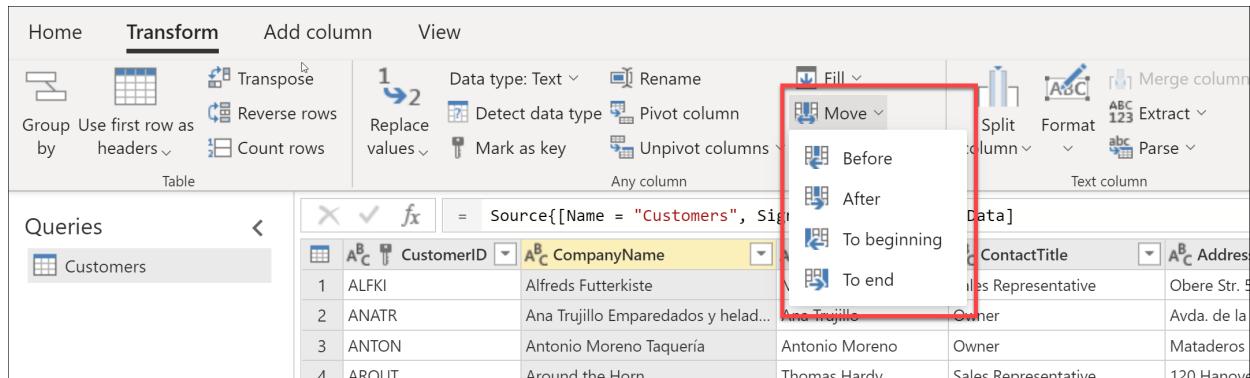


A screenshot of a table titled "Table.ReorderColumns(Navigation, {"CustomerID", "CompanyName", "ContactName", "ContactTitle", "Address", "City", "Region", "PostalCode", "Country", "Phone", "F..."}). The table contains 14 rows of data from the Northwind database. The "ContactName" column is highlighted in yellow. The columns are: CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, F...

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	F...
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	null	12209	Germany	
2	ANATR	Ana Trujillo Emparedados y helad...	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	null	05021	Mexico	
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	null	05023	Mexico	
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	null	WA1 1DP	UK	
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	null	9-958 22	Sweden	
6	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany	
7	BLONP	Blondesddsi père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	null	67000	France	
8	BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	null	28023	Spain	
9	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	null	13008	France	
10	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	
11	BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	null	EC2 5NT	UK	
12	CACTU	Cactus Comidas para llevar	Patricia Simpson	Sales Agent	Cerrito 333	Buenos Aires	null	1010	Argentina	
13	CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993	México D.F.	null	05022	Mexico	
14	CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	null	3012	Switzerland	

You move the column using the **Move** option. This option located in the **Any column** group under the **Transform** tab. In the **Move** option, the available choices are:

- Before
- After
- To beginning
- To end



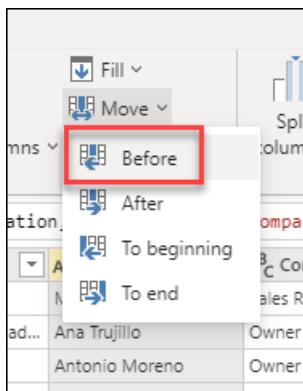
A screenshot of the Power BI ribbon showing the "Transform" tab selected. Under the "Transform" tab, the "Move" option is highlighted with a red box. The "Move" option has four sub-options: "Before", "After", "To beginning", and "To end". The "Move" option is located in the "Any column" group, which also includes "Fill", "Split column", "Format", and "Text column".

You can also find this option when you right-click a column.

A screenshot of the Power BI Data Editor showing a context menu for a column. The menu includes options like Copy preview data, Remove columns, Remove other columns, Duplicate column, Remove duplicates, Remove errors, Split column, Replace values, Replace errors, Change type, Transform column, Group by, Fill, Unpivot columns, Unpivot other columns, Unpivot only selected columns, Rename, Move, Before, After, To beginning, and To end. The 'Move' option is highlighted with a red box.

	CustomerID	CompanyName	ContactName	ContactTitle	Address
1	ALFKI	Alfreds Futterkiste	Sales Representative	Obere Str.	
2	ANATR	Ana Trujillo Emparedados y helados	Owner	Avda. de la Constitución 2222	
3	ANTON	Antonio Moreno Taquería	Owner	Mataderos 2312	
4	AROUT	Around the Horn	Sales Representative	120 Hanover Sq.	
5	BERGS	Berglunds snabbköp	Order Administrator	Berguvsvägen 8	
6	BLAUS	Blauer See Delikatessen	Sales Representative	Forsterstr. 57	
7	BLONP	Blondesddsl père et fils	Marketing Manager	24, place Kléber	
8	BOLID	Bólido Comidas preparadas	Owner	C/ Araquil, 67	
9	BONAP	Bon app'	Owner	12, rue des Bouchers	
10	BOTTM	Bottom-Dollar Markets	Accounting Manager	23 Tswassen Blvd.	
11	BSBEV	B's Beverages	Sales Representative	Fauntleroy Circus	
12	CACTU	Cactus Comidas para llevar	Sales Agent	Cerrito 333	
13	CENTC	Centro comercial Moctezuma	Marketing Manager	Sierras de Granada 9993	
14	CHOPS	Chop-suey Chinese	Owner	Hauptstr. 29	
15	COMMI	Comércio Mineiro	Sales Associate	Av. dos Lázares 12	
16	CONSH	Consolidated Holdings	Sales Representative	Berkeley Center	
17	DRACD	Drachenblut Delikatessen	Order Administrator	Walserweg 65	
18	DUMON	Du monde entier	Owner	67, rue de la Paix	
19	EASTC	Eastern Connection	Sales Agent	35 King George	
20	ERNSH	Ernst Handel	Owner	Strasse 55	
21	FAMIA	Familia Arquibaldo	Sales Representative	Östliche Allee 99	
22	FISSA	FISSA Fabrica Inter. S.A.	Owner	Alzaga 23	
23	FOLIG	Folies gourmandes	Sales Representative	au 12	
24					

If you want to move one column to the left, then select **Before**.

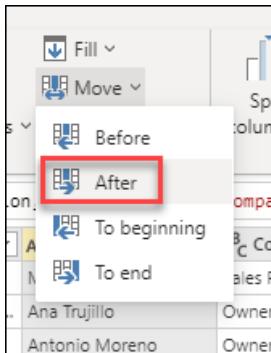


The new location of the column is now one column to the left of its original location.

A screenshot of the Power BI Data Editor showing the modified table structure. The columns have been reordered, demonstrating the result of moving a column to the left.

	CustomerID	ContactName	CompanyName	ContactTitle	Address	City	Region	PostalCode	Country
1	ALFKI	Maria Anders	Alfreds Futterkiste	Sales Representative	Obere Str. 57	Berlin	null	12209	Germany
2	ANATR	Ana Trujillo	Ana Trujillo Emparedados y helados	Owner	Avda. de la Constitución 2222	México D.F.	null	05021	Mexico
3	ANTON	Antonio Moreno	Antonio Moreno Taquería	Owner	Mataderos 2312	México D.F.	null	05023	Mexico
4	AROUT	Thomas Hardy	Around the Horn	Sales Representative	120 Hanover Sq.	London	null	W1 1DP	UK
5	BERGS	Christina Berglund	Berglunds snabbköp	Order Administrator	Berguvsvägen 8	Luleå	null	95-822	Sweden
6	BLAUS	Hanna Moos	Blauer See Delikatessen	Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany
7	BLONP	Frédérique Citeaux	Blondesddsl père et fils	Marketing Manager	24, place Kléber	Strasbourg	null	67000	France
8	BOLID	Martin Sommer	Bólido Comidas preparadas	Owner	C/ Araquil, 67	Madrid	null	28023	Spain
9	BONAP	Laurence Lebihan	Bon app'	Owner	12, rue des Bouchers	Marseille	null	13008	France
10	BOTTM	Elizabeth Lincoln	Bottom-Dollar Markets	Accounting Manager	23 Tswassen Blvd.	Tswassen	BC	T2F 8M4	Canada
11	BSBEV	Victoria Ashworth	B's Beverages	Sales Representative	Fauntleroy Circus	London	null	EC2 5NT	UK
12	CACTU	Patricia Simpson	Cactus Comidas para llevar	Sales Agent	Cerrito 333	Buenos Aires	null	1010	Argentina
13	CENTC	Francisco Chang	Centro comercial Moctezuma	Marketing Manager	Sierras de Granada 9993	México D.F.	null	05022	Mexico
14	CHOPS	Yang Wang	Chop-suey Chinese	Owner	Hauptstr. 29	Bern	null	3012	Switzerland

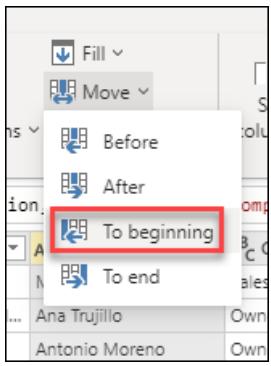
If you want to move one column to the right, then select **After**.



The new location of the column is now one column to the right of its original location.

A _C CustomerID	A _C CompanyName	A _C ContactTitle	A _C ContactName	A _C Address	A _C City	A _C Region	A _C PostalCode	A _C Country
1 ALFKI	Alfreds Futterkiste	Sales Representative	Maria Anders	Obere Str. 57	Berlin	null	12209	Germany
2 ANATR	Ana Trujillo Emparedados y helad...	Owner	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	null	05021	Mexico
3 ANTON	Antonio Moreno Taquería	Owner	Antonio Moreno	Mataderos 2312	México D.F.	null	05023	Mexico
4 AROUT	Around the Horn	Sales Representative	Thomas Hardy	120 Hanover Sq.	London	null	WA1 1DP	UK
5 BERGS	Berglunds snabbköp	Order Administrator	Christina Berglund	Berguvsvägen 8	Luleå	null	5-958 22	Sweden
6 BLAUS	Blauer See Delikatessen	Sales Representative	Hanna Moos	Forsterstr. 57	Mannheim	null	68306	Germany
7 BLONP	Blondesddsi père et fils	Marketing Manager	Frédérique Citeaux	24, place Kléber	Strasbourg	null	67000	France
8 BOLID	Bólido Comidas preparadas	Owner	Martín Sommer	C/ Araquil, 67	Madrid	null	28023	Spain
9 BONAP	Bon app'	Owner	Laurence Lebihan	12, rue des Bouchers	Marseille	null	13008	France
10 BOTTM	Bottom-Dollar Markets	Accounting Manager	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada
11 BSBEV	B's Beverages	Sales Representative	Victoria Ashworth	Faunteroy Circus	London	null	EC2 5NT	UK
12 CACTU	Cactus Comidas para llevar	Sales Agent	Patricia Simpson	Cerrito 333	Buenos Aires	null	1010	Argentina
13 CENTC	Centro comercial Móctezuma	Marketing Manager	Francisco Chang	Sierras de Granada 9993	México D.F.	null	05022	Mexico
14 CHOPS	Chop-suey Chinese	Owner	Yang Wang	Hauptstr. 29	Bern	null	3012	Switzerland

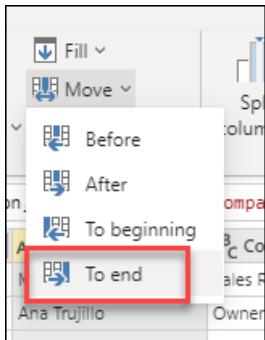
If you want to move the column to the most left space of the dataset, then select **To beginning**.



The new location of the column is now on the far left side of the table.

A _C ContactName	A _C CustomerID	A _C CompanyName	A _C ContactTitle	A _C Address	A _C City	A _C Region	A _C PostalCode	A _C Country
1 Maria Anders	ALFKI	Alfreds Futterkiste	Sales Representative	Obere Str. 57	Berlin	null	12209	Germany
2 Ana Trujillo	ANATR	Ana Trujillo Emparedados y helad...	Owner	Avda. de la Constitución 2222	México D.F.	null	05021	Mexico
3 Antonio Moreno	ANTON	Antonio Moreno Taquería	Owner	Mataderos 2312	México D.F.	null	05023	Mexico
4 Thomas Hardy	AROUT	Around the Horn	Sales Representative	120 Hanover Sq.	London	null	WA1 1DP	UK
5 Christina Berglund	BERGS	Berglunds snabbköp	Order Administrator	Berguvsvägen 8	Luleå	null	5-958 22	Sweden
6 Hanna Moos	BLAUS	Blauer See Delikatessen	Sales Representative	Forsterstr. 57	Mannheim	null	68306	Germany
7 Frédérique Citeaux	BLONP	Blondesddsi père et fils	Marketing Manager	24, place Kléber	Strasbourg	null	67000	France
8 Martín Sommer	BOLID	Bólido Comidas preparadas	Owner	C/ Araquil, 67	Madrid	null	28023	Spain
9 Laurence Lebihan	BONAP	Bon app'	Owner	12, rue des Bouchers	Marseille	null	13008	France
10 Elizabeth Lincoln	BOTTM	Bottom-Dollar Markets	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada
11 Victoria Ashworth	BSBEV	B's Beverages	Sales Representative	Faunteroy Circus	London	null	EC2 5NT	UK
12 Patricia Simpson	CACTU	Cactus Comidas para llevar	Sales Agent	Cerrito 333	Buenos Aires	null	1010	Argentina
13 Francisco Chang	CENTC	Centro comercial Móctezuma	Marketing Manager	Sierras de Granada 9993	México D.F.	null	05022	Mexico
14 Yang Wang	CHOPS	Chop-suey Chinese	Owner	Hauptstr. 29	Bern	null	3012	Switzerland

If you want to move the column to the most right space of the dataset, then select **To end**.



The new location of the column is now on the far right side of the table.

	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax	Orders	CustomerDemographics	ContactName
1	Representative	Obere Str. 57	Berlin	null	12209	Germany	030-0074321	030-0076545	[Table]	[Table]	Maria Anders
2	Manager	Avenida de la Constitución 2222	México D.F.	null	05021	Mexico	(5) 555-4729	(5) 555-3745	[Table]	[Table]	Ana Trujillo
3	Manager	Mataderos 2312	México D.F.	null	05023	Mexico	(5) 555-3932	null	[Table]	[Table]	Antonio Moreno
4	Representative	120 Hanover Sq.	London	null	WA1 1DP	UK	(171) 555-7788	(171) 555-6750	[Table]	[Table]	Thomas Hardy
5	Administrator	Berguvsvägen 8	Luleå	null	9958 22	Sweden	0921-12 34 65	0921-12 34 67	[Table]	[Table]	Christina Berglund
6	Representative	Forsterstr. 57	Mannheim	null	68306	Germany	0621-08460	0621-08924	[Table]	[Table]	Hanna Moos
7	Marketing Manager	24, place Kléber	Strasbourg	null	67000	France	86.60.15.31	86.60.15.32	[Table]	[Table]	Frédérique Citeaux
8	Manager	C/ Atocha, 67	Madrid	null	28023	Spain	(91) 555 22 82	(91) 555 91 99	[Table]	[Table]	Martin Sommer
9	Manager	12, rue des Bouchers	Marseille	null	13008	France	91.24.45.40	91.24.45.41	[Table]	[Table]	Laurence Lebihan
10	Marketing Manager	23 Tswassen Blvd.	Tswassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745	[Table]	[Table]	Elizabeth Lincoln
11	Representative	Fauntieroy Circus	London	null	EC2 5NT	UK	(171) 555-1212	null	[Table]	[Table]	Victoria Ashworth
12	Agent	Cerrito 333	Buenos Aires	null	1010	Argentina	(1) 135-5555	(1) 135-4892	[Table]	[Table]	Patricia Simpson
13	Marketing Manager	Sierras de Granada 9993	México D.F.	null	05022	Mexico	(5) 555-3392	(5) 555-7293	[Table]	[Table]	Francisco Chang
14	Manager	Hauptstr. 29	Bern	null	3012	Switzerland	0452-076545	null	[Table]	[Table]	Yang Wang

Drag and drop

Another way to move a column through the dataset is to drag and drop the column. Move the column to the place where you would like to place it.

Table.ReorderColumns(Navigat	rid	ABC ContactName
Alfreds Futterkiste		
Ana Trujillo Emparedados y helados		
Antonio Moreno Taquería		
Around the Horn		
Berglunds snabbköp		
Blauer See Delikatessen		
Blondesdds! père et fils		
Bólido Comidas preparadas		
Bon app'		

Go to column feature

If you want to find a specific column, then go to the **View** tab in the ribbon and select **Go to column**.

From there, you can specifically select the column you would like to view, which is especially useful if there are many columns.

Power Query - Edit queries

Home Transform Add column View

Data Schema view Go to column Advanced editor Advanced

Queries < Customers Employees

EmployeeID LastName FirstName Title

	EmployeeID	LastName	FirstName	Title
1	1	Davolio	Nancy	Sales Representative
2	2	Fuller	Andrew	Vice President
3	3	Leverling	Janet	Sales Representative
4	4	Peacock	Margaret	Sales Representative
5	5	Buchanan	Steven	Sales Manager
6	6	Suyama	Michael	Sales Representative
7	7	King	Robert	Sales Representative
8	8	Callahan	Laura	Inside Sales Co... Sales Representative
9	9	Dodsworth	Anne	Sales Representative

Go to column

Search:

Address EmployeeID LastName FirstName Title
 20th Ave. E, Apt. 2A Seattle WA 98122 USA
 V. Capital Way Tacoma WA 98401 USA
 Ioss Bay Blvd. Kirkland WA 98033 USA
 Old Redmond Rd. Redmond WA 98052 USA
 Everett Hill London null SW1 8JR UK
 Cherry House Miner Rd. London null EC2 7JR UK
 Sam Hollow Winchester W... London null RG1 9SP UK
 11th Ave. N.E. Seattle WA 98105 USA
 Windstooth Rd. London null WG2 7LT UK

Query settings >

Name: Employees

Entity type: Custom

Applied steps

Source:

Navigation 1

OK Cancel Save & close

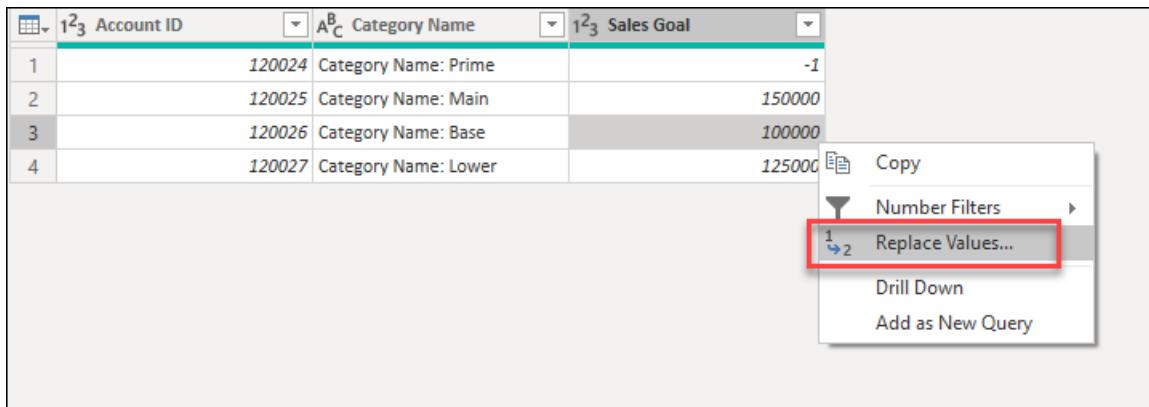
⚠ 2 warnings

Replace values and errors

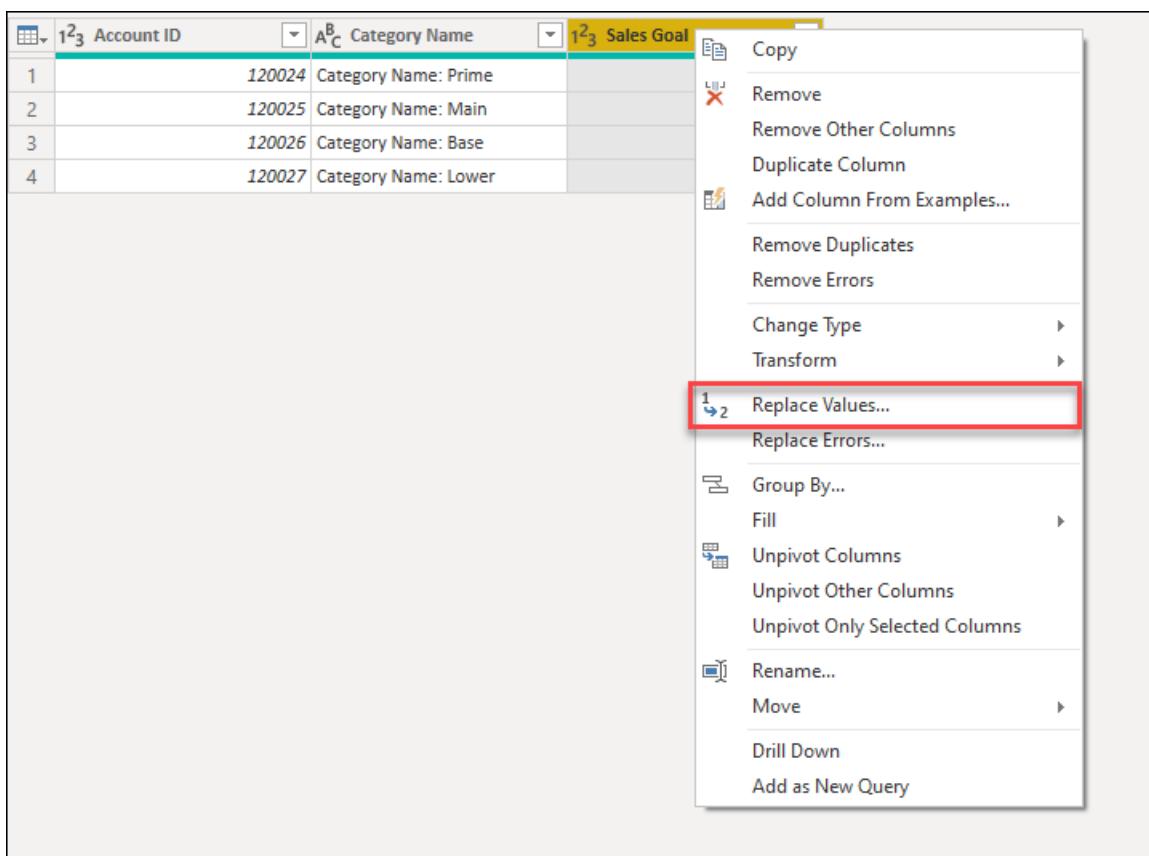
1/15/2022 • 2 minutes to read • [Edit Online](#)

With Power Query, you can replace one value with another value wherever that value is found in a column. The **Replace values** command can be found:

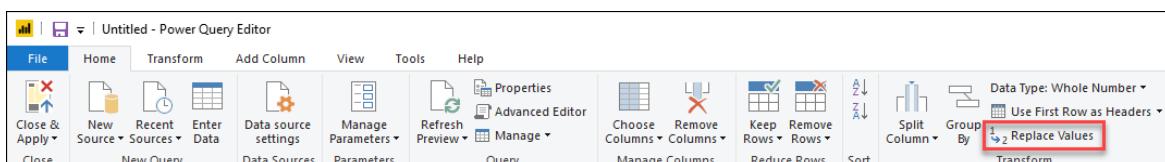
- On the cell shortcut menu. Right-click the cell to replace the selected value in the column with another value.



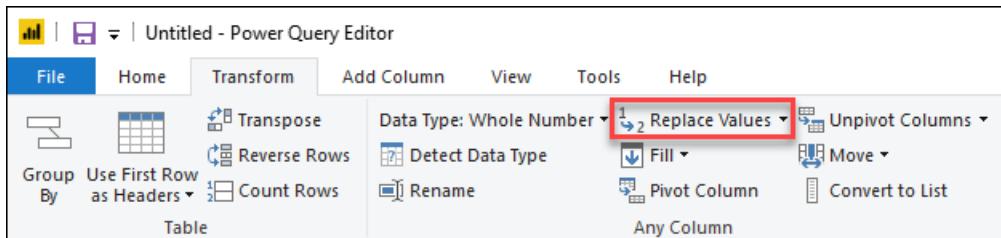
- On the column shortcut menu.



- On the Home tab, in the Transform group.



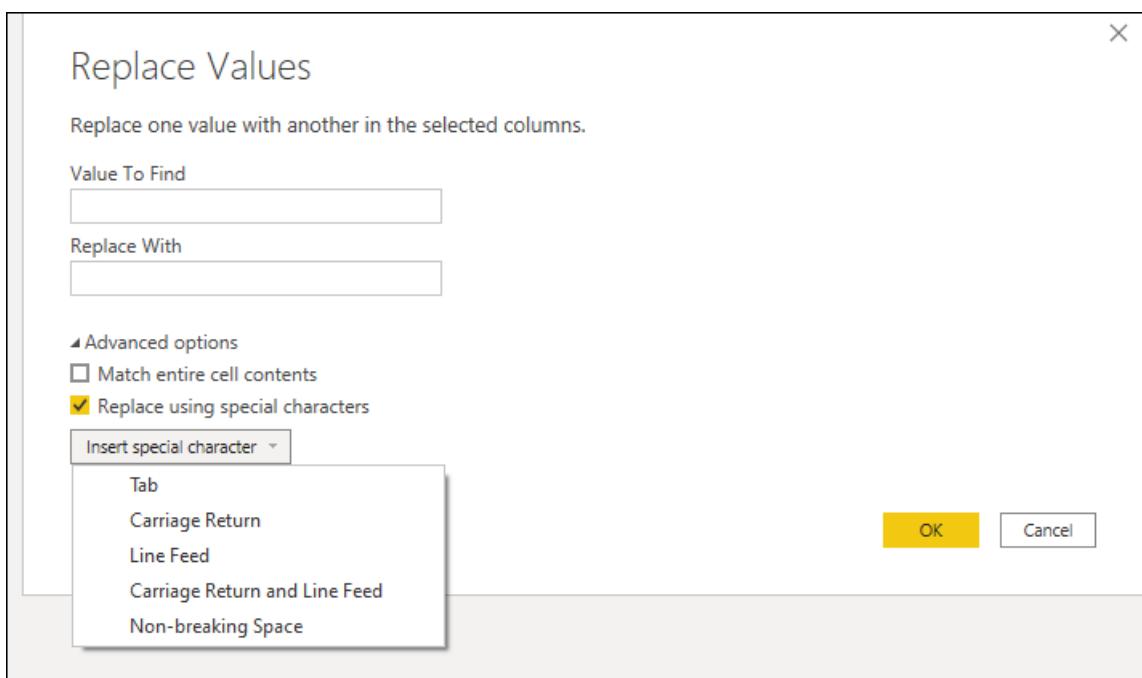
- On the Transform tab, in the Any column group.



The replace values operation has two modes:

- Replace entire cell contents:** This is the default behavior for non-text columns, where Power Query searches for and replaces the full contents of a cell. You can enable this mode for text columns by selecting **Advanced options**, and then selecting the **Match entire cell contents** check box.
- Replace instances of a text string:** This is the default behavior for text columns, where Power Query will search for a specific text string in all rows of a column and replace as many instances of the text string that it finds.

Advanced options are only available in columns of the Text data type. Within that set of options is the **Replace using special characters** option.

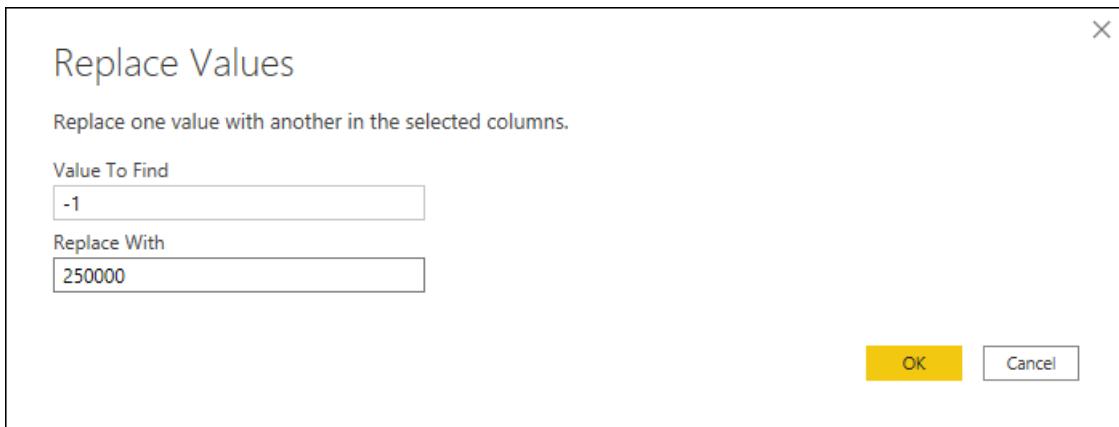


Replace entire cell contents

Imagine a table like the following, where you have columns for **Account ID**, **Category Name**, and **Sales Goal**.

	Account ID	Category Name	Sales Goal
1	120024	Category Name: Prime	-1
2	120025	Category Name: Main	150000
3	120026	Category Name: Base	100000
4	120027	Category Name: Lower	125000

The value of -1 in the **Sales Goal** column is an error in the source and needs to be replaced with the standard sales goal defined by the business for these instances, which is 250,000. To do that, right-click the -1 value, and then select **Replace values**. This action will bring up the Replace values dialog box with **Value to find** set to -1. Now all you need to do is enter 250000 in the **Replace with** box.

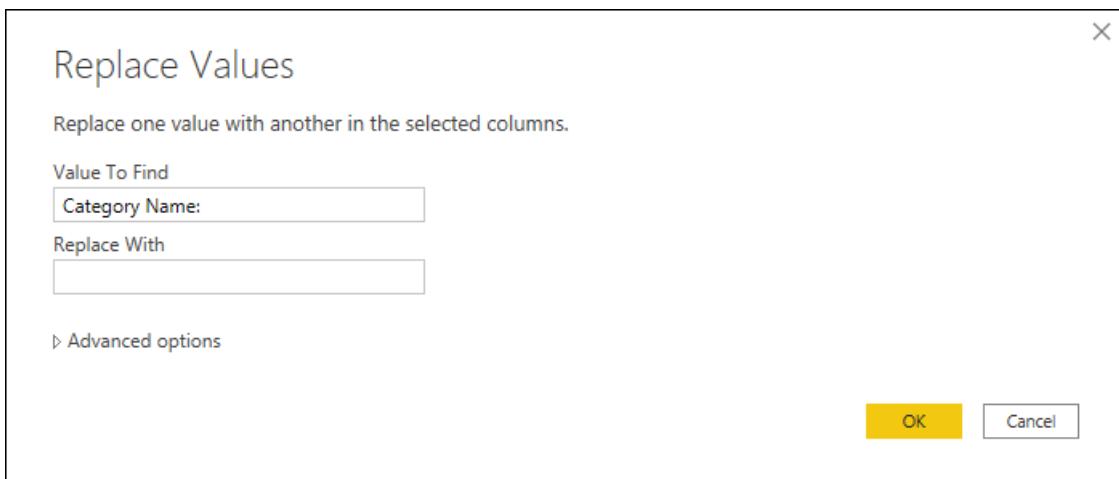


The outcome of that operation will give you the result that you're looking for.

	123 Account ID	A ^B Category Name	1.2 Sales Goal
1	120024	Category Name: Prime	250000
2	120025	Category Name: Main	150000
3	120026	Category Name: Base	100000
4	120027	Category Name: Lower	125000

Replace instances of a text string

Continuing with the previous table, let's say you want to remove the text string "Category Name:" from the **Category Name** column. To do that, go to the **Transform** group on the **Home** tab, and select **Replace values**. In the **Replace values** dialog box, enter the text string **Category Name:** (followed by a space) in the **Value to find** box, leave the **Replace with** box empty, and then select **OK**.



The result of that operation gives you the table in the following image.

	123 Account ID	A ^B Category Name	1.2 Sales Goal
1	120024	Prime	250000
2	120025	Main	150000
3	120026	Base	100000
4	120027	Lower	125000

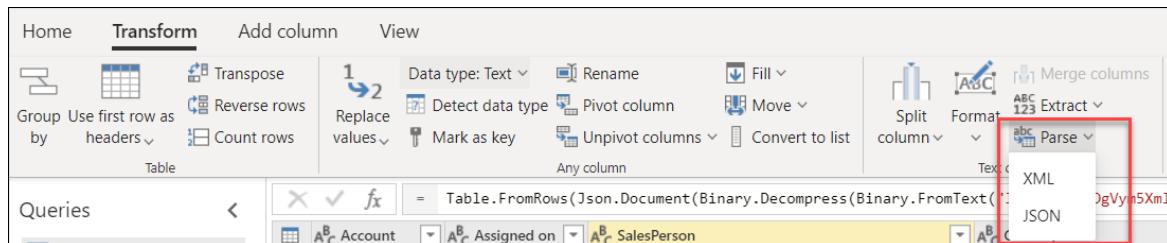
Parse text as JSON or XML

1/15/2022 • 2 minutes to read • [Edit Online](#)

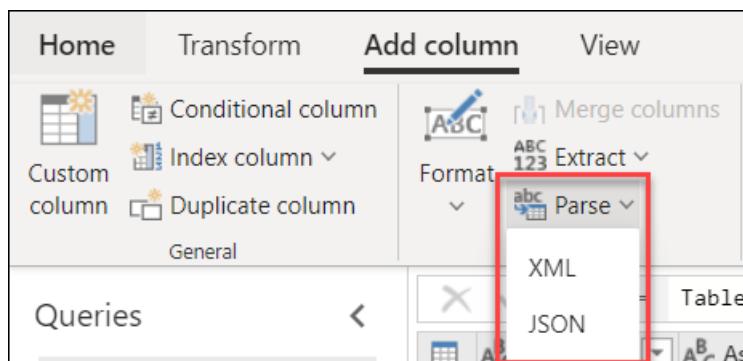
In Power Query, you can parse the contents of a column with text strings by identifying the contents as either a JSON or XML text string.

You can perform this parse operation by selecting the **Parse** button found inside the following places in the Power Query Editor:

- **Transform tab**—This button will transform the existing column by parsing its contents.



- **Add column tab**—This button will add a new column to the table parsing the contents of the selected column.



For this article, you'll be using the following sample table that contains the following columns that you need to parse:

- **SalesPerson**—Contains unparsed JSON text strings with information about the **FirstName** and **LastName** of the sales person, as in the following example.

```
{  
    "id" : 249319,  
    "FirstName": "Lesa",  
    "LastName": "Byrd"  
}
```

- **Country**—Contains unparsed XML text strings with information about the **Country** and the **Division** that the account has been assigned to, as in the following example.

```
<root>  
  <id>1</id>  
  <Country>USA</Country>  
  <Division>BI-3316</Division>  
</root>
```

The sample table looks as follows.

	A^BC Account	Assigned on	A^BC SalesPerson	A^BC Country
1	PTY-51578	6/5/2019	{ "id": 249319, "FirstName": "Les", "LastName": "Byrd" }	<root> <id>3</id> <Country>Panama</Country> <Division>BI-507</Division>...
2	USA-12475	11/12/2019	{ "id": 253212, "FirstName": "Kenny", "LastName": "Smith" }	<root> <id>1</id> <Country>USA</Country> <Division>BI-3316</Division>...
3	CA-35789	2/1/2020	{ "id": 112333, "FirstName": "Alberto", "LastName": "Gass" }	<root> <id>2</id> <Country>Canada</Country> <Division>BI-485</Division>...
4	MX-78933	3/24/2020	{ "id": 987514, "FirstName": "Alissa", "LastName": "Parker" }	<root> <id>5</id> <Country>Mexico</Country> <Division>BI-52</Division>...

The goal is to parse the above mentioned columns and expand the contents of those columns to get this output.

	A^BC Account	Assigned on	A^BC SalesPerson.FirstName	A^BC SalesPerson.LastName	A^BC Country.Country	A^BC Country.Division
1	PTY-51578	6/5/2019	Les	Byrd	Panama	BI-507
2	USA-12475	11/12/2019	Kenny	Smith	USA	BI-3316
3	CA-35789	2/1/2020	Alberto	Gass	Canada	BI-485
4	MX-78933	3/24/2020	Alissa	Parker	Mexico	BI-52

As JSON

Select the **SalesPerson** column. Then select **JSON** from the **Parse** dropdown menu inside the **Transform** tab. These steps will transform the **SalesPerson** column from having text strings to having **Record** values, as shown in the next image. You can select anywhere in the whitespace inside the cell of the **Record** value to get a detailed preview of the record contents at the bottom of the screen.



	A^BC Account	Assigned on	A^BC SalesPerson	A^BC Country
1	PTY-51578	6/5/2019	[Record]	<root> <id>3</id> <Country>Panama</Country> <Division>BI-507</Division>...
2	USA-12475	11/12/2019	[Record]	<root> <id>1</id> <Country>USA</Country> <Division>BI-3316</Division>...
3	CA-35789	2/1/2020	[Record]	<root> <id>2</id> <Country>Canada</Country> <Division>BI-485</Division>...
4	MX-78933	3/24/2020	[Record]	<root> <id>5</id> <Country>Mexico</Country> <Division>BI-52</Division>...

id	249319
FirstName	Les
LastName	Byrd

Select the expand icon next to the **SalesPerson** column header. From the expand columns menu, select only the **FirstName** and **LastName** fields, as shown in the following image.

Search

(Select all)
 id
 FirstName
 LastName

Use original column name as prefix

OK **Cancel**

	A^BC Account	Assigned on	A^BC SalesPerson	A^BC Country
1	PTY-51578	6/5/2019	[Record]	507
2	USA-12475	11/12/2019	[Record]	6<
3	CA-35789	2/1/2020	[Record]	485
4	MX-78933	3/24/2020	[Record]	52<

The result of that operation will give you the following table.

	ABC Account	Assigned on	ABC SalesPerson.FirstName	ABC SalesPerson.LastName	ABC Country
1	PTY-51578	6/5/2019	Lesa	Byrd	<root> <id>3</id> <Country>Panama</Country> <Division>BI-507</Division>
2	USA-12475	11/12/2019	Kenny	Smith	<root> <id>1</id> <Country>USA</Country> <Division>BI-3316</Division>
3	CA-35789	2/1/2020	Alberto	Gass	<root> <id>2</id> <Country>Canada</Country> <Division>BI-485</Division>
4	MX-78933	3/24/2020	Alissa	Parker	<root> <id>5</id> <Country>Mexico</Country> <Division>BI-52</Division>

As XML

Select the **Country** column. Then select the **XML** button from the **Parse** dropdown menu inside the **Transform** tab. These steps will transform the **Country** column from having text strings to having **Table** values as shown in the next image. You can select anywhere in the whitespace inside the cell of the **Table** value to get a detailed preview of the contents of the table at the bottom of the screen.

The screenshot shows a Power BI data view with four rows of data. The columns are: ABC Account, Assigned on, ABC SalesPerson.FirstName, ABC SalesPerson.LastName, and ABC Country. The 'Country' column contains text values like '<root> <id>3</id> <Country>Panama</Country> <Division>BI-507</Division>' for the first row. A red arrow points from the 'Country' column header to a small yellow [Table] icon in the cell for the second row. Below the main table, a preview window shows a single row with three columns: ABC id, ABC Country, and ABC Division. The 'ABC id' cell contains '3', 'ABC Country' contains 'Panama', and 'ABC Division' contains 'BI-507'.

Select the expand icon next to the **Country** column header. From the expand columns menu, select only the **Country** and **Division** fields, as shown in the following image.

The screenshot shows the 'Expand Columns' dialog box. It lists columns: ABC Account, Assigned on, ABC SalesPerson.FirstName, ABC SalesPerson.LastName, and ABC Country. The 'ABC Country' column is highlighted in yellow. To the right, there's a search bar and a list of columns with checkboxes: 'id' (unchecked), 'Country' (checked), and 'Division' (checked). At the bottom, there's a checkbox for 'Use original column name as prefix' (unchecked) and two buttons: 'OK' (yellow) and 'Cancel'.

You can define all the new columns as text columns. The result of that operation will give you the output table that you're looking for.

	ABC Account	Assigned on	ABC SalesPerson.FirstName	ABC SalesPerson.LastName	ABC Country.Country	ABC Country.Division
1	PTY-51578	6/5/2019	Lesa	Byrd	Panama	BI-507
2	USA-12475	11/12/2019	Kenny	Smith	USA	BI-3316
3	CA-35789	2/1/2020	Alberto	Gass	Canada	BI-485
4	MX-78933	3/24/2020	Alissa	Parker	Mexico	BI-52

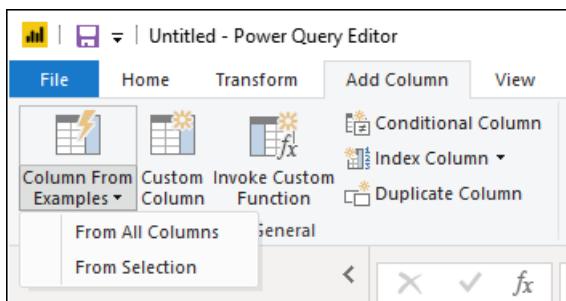
Add a column from examples

1/15/2022 • 3 minutes to read • [Edit Online](#)

When you add columns from examples, you can quickly and easily create new columns that meet your needs. This is useful for the following situations:

- You know the data you want in your new column, but you're not sure which transformation, or collection of transformations, will get you there.
- You already know which transformations you need, but you're not sure what to select in the UI to make them happen.
- You know all about the transformations you need by using a custom column expression in the M language, but one or more of those transformations aren't available in the UI.

The **Column from examples** command is located on the **Add column** tab, in the **General** group.



Add a new column from examples

In this example, you start with the table shown in the following image.

	A ^B C First Name	A ^B C Last Name	A ^B C CustomerID	A ^B C CompanyName	A ^B C Country	1 ² 3 Monthly Income
1	Maria	Anders	ALFKI	Alfreds Futterkiste	Germany	19500
2	Ana	Trujillo	ANATR	Ana Trujillo Emparedados y helados	Mexico	15500
3	Antonio	Moreno	ANTON	Antonio Moreno Taquería	Mexico	17500
4	Thomas	Hardy	AROUT	Around the Horn	UK	6000
5	Christina	Berglund	BERGS	Berglunds snabbköp	Sweden	14000
6	Hanna	Moos	BLAUS	Blauer See Delikatessen	Germany	9500
7	Frédérique	Citeaux	BLONP	Blondesddsl père et fils	France	17000
8	Martín	Sommer	BOLID	Bólido Comidas preparadas	Spain	8000
9	Laurence	Lebihan	BONAP	Bon app'	France	10500
10	Elizabeth	Lincoln	BOTTM	Bottom-Dollar Markets	Canada	17500
11	Victoria	Ashworth	BSBEV	B's Beverages	UK	13000
12	Patricia	Simpson	CACTU	Cactus Comidas para llevar	Argentina	14000
13	Francisco	Chang	CENTC	Centro comercial Moctezuma	Mexico	15000
14	Yng	Wang	CHOPS	Chop-suey Chinese	Switzerland	5000
15	Pedro	Afonso	COMMI	Comércio Mineiro	Brazil	18000
16	Elizabeth	Brown	CONSH	Consolidated Holdings	UK	12000
17	Sven	Ottlieb	DRACD	Drachenblut Delikatessen	Germany	8000
18	Janine	Labrune	DUMON	Du monde entier	France	6000
19	Ann	Devon	EASTC	Eastern Connection	UK	17500
20	Roland	Mendel	ERNSH	Ernst Handel	Austria	10500

Your goal in this example is to create two new columns:

- **Range:** Create bins for the **Monthly Income** column in discrete increments of 5,000.
- **Full Name:** Concatenate the **Last Name** and **First Name** columns to a single column.

5 COLUMNS, 20 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 4:27 AM

APPLIED STEPS

- Source
- Changed Type
- Inserted Range
- Inserted Merged Column
- Removed Columns

Column from examples, from selected columns

One of the options you have when creating your new column is to select which columns will be used in your calculations. For this example, you'll be creating the **Range** column from the values in the **Monthly Income** column.

To do this, select the **Monthly Income** column, select the **Column from examples** command, and then select **From selection**.

From Selection

APPLIED STEPS

- Source
- Changed Type
- Inserted Range
- Inserted Merged Column
- Removed Columns

The preview pane displays a new, editable column where you can enter your examples. For the first example, the value from the selected column is 19500. So in your new column, enter the text **15000 to 20000**, which is the bin where that value falls.

6 COLUMNS, 20 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 3:50 AM

When Power Query finds a matching transformation, it fills the transformation results into the remaining rows using light-colored text. You can also see the M formula text for the transformation above the table preview.

After you select OK, you'll see your new column as part of your query. You'll also see a new step added to your query.

7 COLUMNS, 20 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 4:16 AM

Column from examples, from all columns

The next goal is to create a **Full Name** column by using the values from the **First Name** and **Last Name** columns.

To do this, select the **Column from examples** command, and then select **From all columns**.

The screenshot shows the Power Query Editor interface. A red box highlights the 'From All Columns' button in the ribbon under the 'Add Column' tab. The main area displays a table with 20 rows of customer data. The columns are: First Name, Last Name, CustomerID, CompanyName, Country, Monthly Income, and Range. The 'Range' column contains ranges like '15000 15000 to 20000'. The 'Properties' pane on the right shows the query name is 'Query1'. The 'Applied Steps' pane shows the 'Inserted Range' step.

Now you'll enter your first **Full Name** example as **Enders, Maria**.

The screenshot shows the Power Query Editor after adding the 'Full Name' column. The 'Applied Steps' pane now includes 'Column From Examples'. The 'Properties' pane shows the 'Name' is 'Query1'. The 'Preview' pane at the bottom right shows the preview was downloaded at 4:16 AM.

First Name	Last Name	CustomerID	CompanyName	Country	Monthly Income	Range	Full Name
Maria	Anders	ALFKI	Alfreds Futterkiste	Germany	19500	15000 to 20000	Anders, Maria
Ana	Trujillo	ANATR	Ana Trujillo Emparedados y helados	Mexico	15500	15000 to 20000	Trujillo, Ana
Antonio	Moreno	ANTON	Antonio Moreno Taqueria	Mexico	17500	15000 to 20000	Moreno, Antonio
Thomas	Hardy	AROUT	Around the Horn	UK	6000	5000 to 10000	Hardy, Thomas
Christina	Berglund	BERGS	Berglunds snabbgöp	Sweden	14000	10000 to 15000	Berglund, Christina
Hanna	Moos	BLAUS	Blauer See Delikatessen	Germany	9500	5000 to 10000	Moos, Hanna
Frédérique	Citeaux	BLONP	Blondesdöd père et fils	France	17000	15000 to 20000	Citeaux, Frédérique
Martin	Sommer	BOLID	Bólido Comidas preparadas	Spain	8000	5000 to 10000	Sommer, Martin
Laurence	Lebihan	BONAP	Bon app'	France	10500	10000 to 15000	Lebihan, Laurence
Elizabeth	Lincoln	BOTTM	Bottom-Dollar Markets	Canada	17500	15000 to 20000	Lincoln, Elizabeth
Victoria	Ashworth	BSBEV	B's Beverages	UK	13000	10000 to 15000	Ashworth, Victoria
Patricia	Simpson	CACTU	Cactus Comidas para llevar	Argentina	14000	10000 to 15000	Simpson, Patricia
Francisco	Chang	CENTC	Centro comercial Moctezuma	Mexico	15000	15000 to 20000	Chang, Francisco
Yang	Wang	CHOPS	Chop-suey Chinese	Switzerland	5000	5000 to 10000	Wang, Yang
Pedro	Alonso	COMMI	Comerçio Mineiro	Brazil	18000	15000 to 20000	Alonso, Pedro
Elizabeth	Brown	CONSH	Consolidated Holdings	UK	12000	10000 to 15000	Brown, Elizabeth
Sven	Ottlieb	DRACD	Drechelnbut Delikatessen	Germany	8000	5000 to 10000	Ottlieb, Sven
Janine	Labrune	DUMON	Du monde entier	France	6000	5000 to 10000	Labrune, Janine
Ann	Devon	EASTC	Eastern Connection	UK	17500	15000 to 20000	Devon, Ann
Roland	Mendel	ERNSH	Ernst Handel	Austria	10500	10000 to 15000	Mendel, Roland

After you select **OK**, you'll see your new column as part of your query. You'll also see a new step added to your query.

The screenshot shows the Power Query Editor after selecting 'OK' in the 'Column From Examples' dialog. The 'Applied Steps' pane now includes 'Inserted Merged Column'. The 'Properties' pane shows the 'Name' is 'Query1'. The 'Preview' pane at the bottom right shows the preview was downloaded at 4:26 AM.

First Name	Last Name	CustomerID	CompanyName	Country	Monthly Income	Range	Full Name
Maria	Anders	ALFKI	Alfreds Futterkiste	Germany	19500	15000 to 20000	Anders, Maria
Ana	Trujillo	ANATR	Ana Trujillo Emparedados y helados	Mexico	15500	15000 to 20000	Trujillo, Ana
Antonio	Moreno	ANTON	Antonio Moreno Taqueria	Mexico	17500	15000 to 20000	Moreno, Antonio
Thomas	Hardy	AROUT	Around the Horn	UK	6000	5000 to 10000	Hardy, Thomas
Christina	Berglund	BERGS	Berglunds snabbgöp	Sweden	14000	10000 to 15000	Berglund, Christina
Hanna	Moos	BLAUS	Blauer See Delikatessen	Germany	9500	5000 to 10000	Moos, Hanna
Frédérique	Citeaux	BLONP	Blondesdöd père et fils	France	17000	15000 to 20000	Citeaux, Frédérique
Martin	Sommer	BOLID	Bólido Comidas preparadas	Spain	8000	5000 to 10000	Sommer, Martin
Laurence	Lebihan	BONAP	Bon app'	France	10500	10000 to 15000	Lebihan, Laurence
Elizabeth	Lincoln	BOTTM	Bottom-Dollar Markets	Canada	17500	15000 to 20000	Lincoln, Elizabeth
Victoria	Ashworth	BSBEV	B's Beverages	UK	13000	10000 to 15000	Ashworth, Victoria
Patricia	Simpson	CACTU	Cactus Comidas para llevar	Argentina	14000	10000 to 15000	Simpson, Patricia
Francisco	Chang	CENTC	Centro comercial Moctezuma	Mexico	15000	15000 to 20000	Chang, Francisco
Yang	Wang	CHOPS	Chop-suey Chinese	Switzerland	5000	5000 to 10000	Wang, Yang
Pedro	Alonso	COMMI	Comerçio Mineiro	Brazil	18000	15000 to 20000	Alonso, Pedro
Elizabeth	Brown	CONSH	Consolidated Holdings	UK	12000	10000 to 15000	Brown, Elizabeth
Sven	Ottlieb	DRACD	Drechelnbut Delikatessen	Germany	8000	5000 to 10000	Ottlieb, Sven
Janine	Labrune	DUMON	Du monde entier	France	6000	5000 to 10000	Labrune, Janine
Ann	Devon	EASTC	Eastern Connection	UK	17500	15000 to 20000	Devon, Ann
Roland	Mendel	ERNSH	Ernst Handel	Austria	10500	10000 to 15000	Mendel, Roland

Your last step is to remove the **First Name**, **Last Name**, and **Monthly Income** columns. Your final table now contains the **Range** and **Full Name** columns with all the data you produced in the previous steps.

5 COLUMNS, 20 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 4:27 AM

Tips and considerations

When providing examples, Power Query offers a helpful list of available fields, values, and suggested transformations for the selected columns. You can view this list by selecting any cell of the new column.

It's important to note that the **Column from examples** experience works only on the top 100 rows of your data preview. You can apply steps before the **Column from examples** step to create your own data sample. After the **Column from examples** column has been created, you can delete those prior steps; the newly created column won't be affected.

List of supported transformations

Many, but not all, transformations are available when you use **Column from examples**. The following list shows the supported transformations.

General

- Conditional Column

Reference

- Reference to a specific column, including trim, clean, and case transformations

Text transformations

- Combine (supports combination of literal strings and entire column values)
- Replace
- Length

- Extract
 - First Characters
 - Last Characters
 - Range
 - Text before Delimiter
 - Text after Delimiter
 - Text between Delimiters
 - Length
 - Remove Characters
 - Keep Characters

NOTE

All Text transformations take into account the potential need to trim, clean, or apply a case transformation to the column value.

Date transformations

- Day
- Day of Week
- Day of Week Name
- Day of Year
- Month
- Month Name
- Quarter of Year
- Week of Month
- Week of Year
- Year
- Age
- Start of Year
- End of Year
- Start of Month
- End of Month
- Start of Quarter
- Days in Month
- End of Quarter
- Start of Week
- End of Week
- Day of Month
- Start of Day
- End of Day

Time transformations

- Hour
- Minute
- Second
- To Local Time

NOTE

All Date and Time transformations take into account the potential need to convert the column value to Date, Time, or DateTime.

Number transformations

- Absolute Value
- Arccosine
- Arcsine
- Arctangent
- Convert to Number
- Cosine
- Cube
- Divide
- Exponent
- Factorial
- Integer Divide
- Is Even
- Is Odd
- Ln
- Base-10 Logarithm
- Modulo
- Multiply
- Round Down
- Round Up
- Sign
- Sine
- Square Root
- Square
- Subtract
- Sum
- Tangent
- Bucketing/Ranges

Add an index column

1/15/2022 • 2 minutes to read • [Edit Online](#)

The **Index column** command adds a new column to the table with explicit position values, and is usually created to support other transformation patterns.

The screenshot shows the Power Query ribbon with the 'Add column' tab selected. Under the 'Add column' tab, the 'Index column' option is highlighted with a red box. A dropdown menu is open, showing three options: 'From 0', 'From 1', and 'Custom...'. The main area of the Power Query editor shows a table with the following data:

	Date	Account	Sale	1/1/2020	A1000	2578	1/2/2020	B2000	4587
1	Date								
2		Account							
3			Sale						
4				1/1/2020					
5					A1000				
6						2578			
7							1/2/2020		
8								B2000	
9									4587

By default, the starting index will start from the value 0 and have an increment of 1 per row.

The screenshot shows the Power Query table after adding the 'Index' column. The 'Index' column is highlighted with a red box. The table now includes an additional column labeled 'Index' with the following values:

	Date	Account	Sale	1/1/2020	A1000	2578	1/2/2020	B2000	4587	Index
1	Date									0
2		Account								1
3			Sale							2
4				1/1/2020						3
5					A1000					4
6						2578				5
7							1/2/2020			6
8								B2000		7
9									4587	8

You can also configure the behavior of this step by selecting the **Custom** option and configuring two parameters:

- **Starting index:** Specifies the initial index value.
- **Increment:** Specifies how much to increment each index value.

Add index column

Add an index column with a specified starting index and increment.

Starting index

0

Increment

1

OK

Cancel

For the example in this article, you start with the following table that has only one column, but notice the data pattern in the column.

	Column1
1	Date
2	Account
3	Sale
4	1/1/2020
5	A1000
6	2578
7	1/2/2020
8	B2000
9	4587

Let's say that your goal is to transform that table into the one shown in the following image, with the columns **Date**, **Account**, and **Sale**.

	Date	Account	Sale
1	1/1/2020	A1000	2578
2	1/2/2020	B2000	4587

Step 1. Add an index column

You first need to add a new **Index** column to your table that starts from 0.

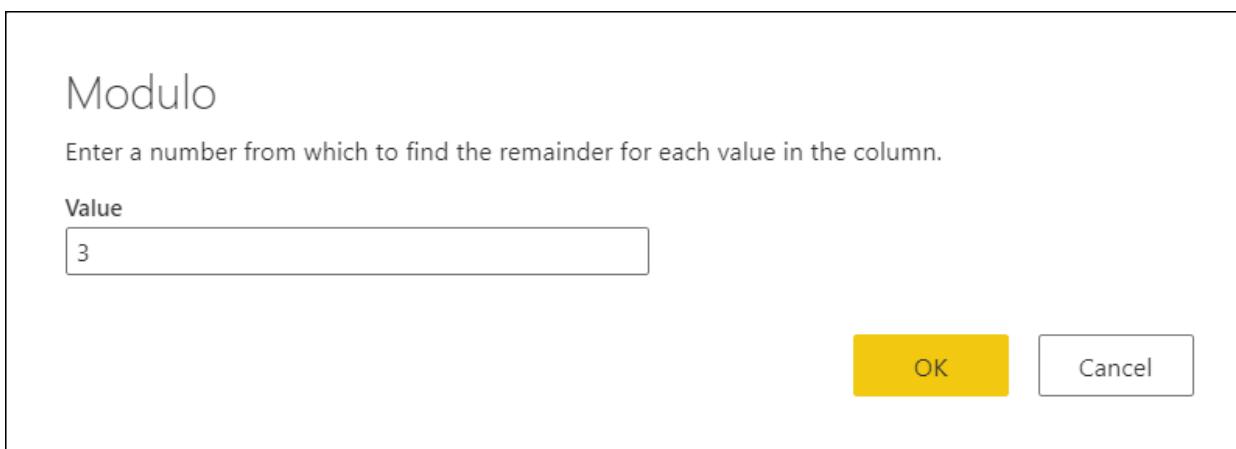
	Column1	Index
1	Date	0
2	Account	1
3	Sale	2
4	1/1/2020	3
5	A1000	4
6	2578	5
7	1/2/2020	6
8	B2000	7
9	4587	8

Step 2. Add a modulo column from the index column

The next step is to select the newly added index column, and then on the **Add column** tab, select Standard > Modulo.

A screenshot of the Power BI desktop application. The ribbon at the top has 'Home', 'Transform', 'Add column' (which is underlined), and 'View'. The 'Add column' tab is active. On the far left, there's a 'Queries' pane with a single query named 'Query'. Below it is a table with 9 rows and 3 columns, labeled '1.2 Index'. The first column contains values like 'Date', 'Account', etc., and the second column contains their indices (0, 1, 2, 3, 4, 5, 6, 7, 8). A context menu is open over the third column, showing options: 'Add', 'Multiply', 'Subtract', 'Divide', 'Divide (Integer)', 'Modulo' (which is highlighted with a red box), 'Percentage', and 'Percent of'. The status bar at the bottom shows the formula: 'Type", "Index", 0, 1)

In the **Modulo** dialog box, enter the number from which to find the remainder for each value in the column. In this case, your pattern repeats itself every three rows, so you'll enter 3.



The result of that operation will give you a new column named **Modulo**.

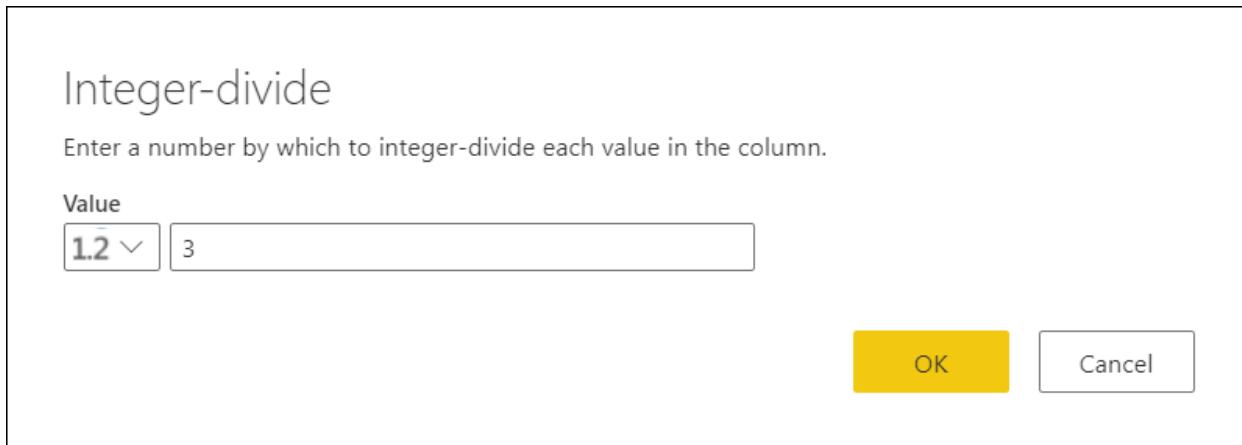
	1.2 Colu...	1.2 Index	1.2 Modulo
1	Date	0	0
2	Account	1	1
3	Sale	2	2
4	1/1/2020	3	0
5	A1000	4	1
6	2578	5	2
7	1/2/2020	6	0
8	B2000	7	1
9	4587	8	2

Step 3. Add an integer-divide column from the index column

Select the **Index** column, go to the **Add column** tab, and then select Standard > Divide (Integer).

The screenshot shows the Power BI desktop interface with the ribbon at the top. The 'Add column' tab is active. A context menu is open over a table named 'Table.AddColumn'. The menu includes options like Add, Multiply, Subtract, Divide, Divide (Integer) (which is highlighted with a red box), Modulo, Percentage, and Percent of.

In the **Integer-divide** dialog box, enter a number by which to divide each value in the column. In this case, your pattern repeats itself every three rows, so enter the value 3.



Remove the **Index** column, because you no longer need it. Your table now looks like the following image.

	Column1	1.2	Modulo	123	Integer-division
1	Date	0	0	0	0
2	Account	1	1	0	0
3	Sale	2	2	0	0
4	1/1/2020	0	0	1	1
5	A1000	1	1	1	1
6	2578	2	2	1	1
7	1/2/2020	0	0	2	2
8	B2000	1	1	2	2
9	4587	2	2	2	2

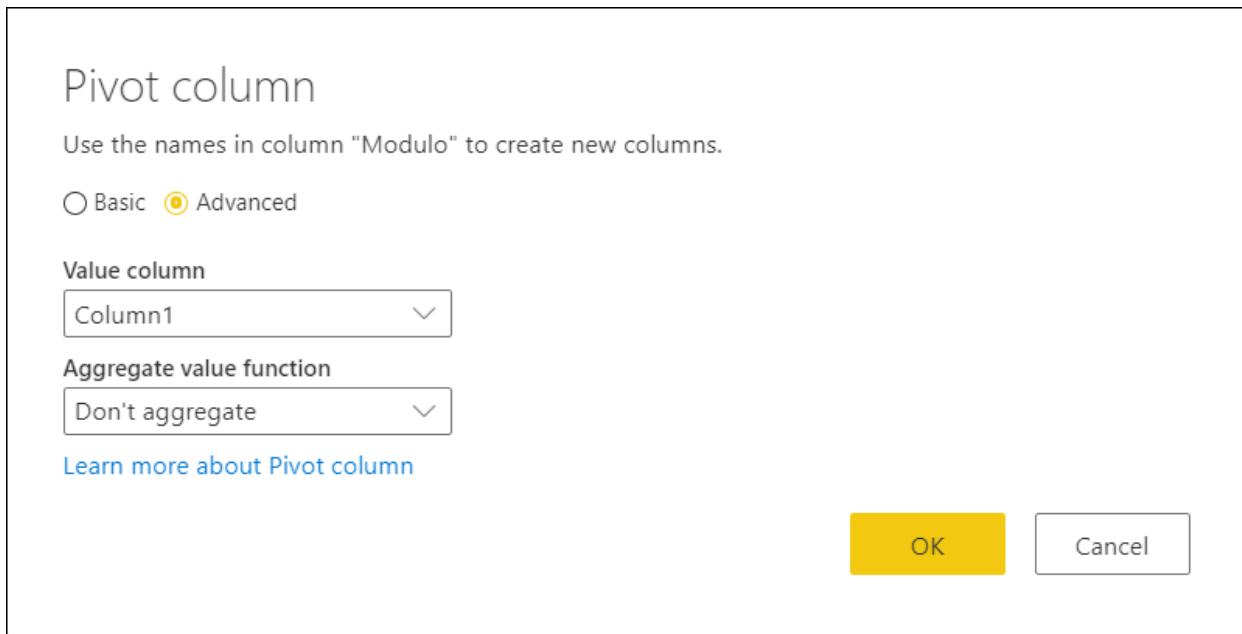
Step 4. Pivot a column

Your table now has three columns where:

- **Column1** contains the values that should be in the final table.
- **Modulo** provides the column position of the value (similar to the y coordinates of an xy chart).
- **Integer-division** provides the row position of the value (similar to the x coordinates of an xy chart).

To achieve the table you want, you need to pivot the **Modulo** column by using the values from **Column1** where

these values don't get aggregated. On the **Transform** tab, select the **Modulo** column, and then select **Pivot column** from the **Any column** group. In the **Pivot column** dialog box, select the **Advanced** option button. Make sure **Value column** is set to **Column1** and **Aggregate values function** is set to **Don't aggregate**.



More information: [Pivot columns](#)

The result of that operation will give you a table with four columns, as shown in the following image.

	1 ² ₃ Integer-division	A ^B _C 0	A ^B _C 1	A ^B _C 2
1	0	Date	Account	Sale
2	1	1/1/2020	A1000	2578
3	2	1/2/2020	B2000	4587

Step 5. Clean the table

You can now delete the **Integer-division** column and promote the first row of the table to become the headers of your table. More information: [Promote or demote column headers](#)

After defining the correct data types for your columns, you'll create a table that looks like the following table, with exactly the three columns that you needed and the shape that you were looking for.

	Date	A ^B _C Account	1 ² ₃ Sale
1	1/1/2020	A1000	2578
2	1/2/2020	B2000	4587

Add a custom column

1/15/2022 • 3 minutes to read • [Edit Online](#)

If you need more flexibility for adding new columns than the ones provided out of the box in Power Query, you can create your own custom column by using the Power Query M formula language.

Imagine that you have a table with the following set of columns.

	Date	Country	Units	Unit Price	Discount
1	2/1/2020	Panama	40	1.25	0%
2	3/1/2020	Panama	50	2.25	5%
3	2/1/2020	USA	150	3.45	10%
4	3/1/2020	USA	175	3.85	15%
5	2/1/2020	Canada	100	2.75	7%
6	3/1/2020	Canada	90	2.95	8%

Using the **Units**, **Unit Price**, and **Discount** columns, you'd like to create two new columns:

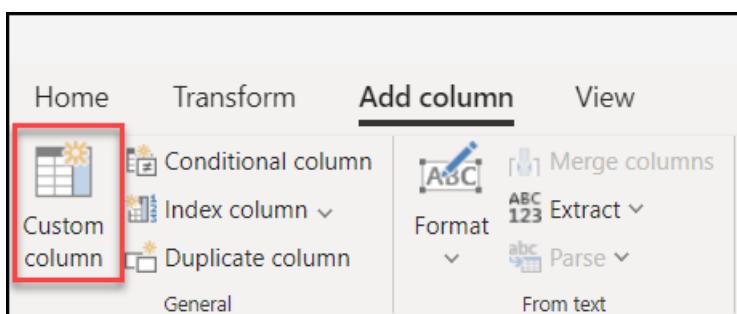
- **Total Sale before Discount**: Calculated by multiplying the **Units** column times the **Unit Price** column.
- **Total Sale after Discount**: Calculated by multiplying the **Total Sale before Discount** column by the net percentage value (one minus the discount value).

The goal is to create a table with new columns that looks like the following image.

	Date	Country	Units	Unit Price	Discount	Total Sale before Discount	Total Sale after Discount
1	2/1/2020	Panama	40	1.25	0%	50	50
2	3/1/2020	Panama	50	2.25	5%	112.5	106.88
3	2/1/2020	USA	150	3.45	10%	517.5	465.75
4	3/1/2020	USA	175	3.85	15%	673.75	572.69
5	2/1/2020	Canada	100	2.75	7%	275	255.75
6	3/1/2020	Canada	90	2.95	8%	265.5	244.26

Create a custom column

On the **Add column** tab, select **Custom column**.



The **Custom column** dialog box appears. This dialog box is where you define the formula to create your column.

Custom column

Add a column that is computed from other columns or values.

New column name

Custom

Custom column formula ①

=

Available column(s)

Date
Country
Units
Unit Price
Discount

Insert column

[Learn more about Power Query formulas](#)

OK

Cancel

The **Custom column** dialog box contains:

- An **Available columns** list on the right.
- The initial name of your custom column in the **New column name** box. You can rename this column.
- [Power Query M formula](#) in the **Custom column formula** box.

To add a new custom column, select a column from the **Available columns** list on the right side of the dialog box. Then select the **Insert column** button below the list to add it to the custom column formula. You can also add a column by selecting it in the list. Alternatively, you can write your own formula by using the Power Query M formula language in the **Custom column formula** box.

NOTE

If there's a syntax error when creating your custom column, you'll see a yellow warning icon, along with an error message and reason.

Adding the Total Sale before Discount column

The formula that you can use to create the **Total Sale before Discount** column is `[Units] * [Unit Price]`. The following image shows how it will look in the **Custom column** dialog box.

Custom column

Add a column that is computed from other columns or values.

New column name

Total Sale before Discount

Custom column formula ①

= [Units] * [Unit Price]

Available column(s)

Date
Country
Units
Unit Price
Discount

Insert column

[Learn more about Power Query formulas](#)

OK

Cancel

The result of that operation will add a new **Total Sale before Discount** column to your table and will look like the following image.

	Date	Country	Units	Unit Price	Discount	Total Sale before Discount
1	2/1/2020	Panama	40	1.25	0%	50
2	3/1/2020	Panama	50	2.25	5%	112.5
3	2/1/2020	USA	150	3.45	10%	517.5
4	3/1/2020	USA	175	3.85	15%	673.75
5	2/1/2020	Canada	100	2.75	7%	275
6	3/1/2020	Canada	90	2.95	8%	265.5

Adding the Total Sale after Discount column

The formula that you can use to create the **Total Sale before Discount** is

`[Total Sale before Discount]*(1-[Discount])`. The following image shows how it will look in your **Custom column** dialog box.

Custom column

Add a column that is computed from other columns or values.

New column name

Total Sale after Discount

Custom column formula ⓘ

= [Total Sale before Discount]* (1-[Discount])

Available column(s)

Date
Country
Units
Unit Price
Discount
Total Sale before Discount

Insert column

[Learn more about Power Query formulas](#)

OK

Cancel

The result of that operation will add a new **Total Sale after Discount** column to your table and will look like the following image.

	Date	Country	Units	Unit Price	Discount	Total Sale before Discount	Total Sale after Discount
1	2/1/2020	Panama	40	1.25	0%	50	50
2	3/1/2020	Panama	50	2.25	5%	112.5	106.875
3	2/1/2020	USA	150	3.45	10%	517.5	465.75
4	3/1/2020	USA	175	3.85	15%	673.75	572.688
5	2/1/2020	Canada	100	2.75	7%	275	255.75
6	3/1/2020	Canada	90	2.95	8%	265.5	244.26

Setting the column data types

Notice that your new columns don't have a data type defined yet. You can tell this by looking at the icon in the header of the column that has the data type icon (ABC123). You'll want to change the data types of both new columns to Currency.

1. Select both the **Total Sale before Discount** and **Total Sale after Discount** columns.

2. On the **Home** tab, in the **Transform** group, select **Data type** > **Currency**.

The screenshot shows the 'Power Query - Edit queries' dialog. In the ribbon, the 'Home' tab is selected. Under the 'Transform' group, the 'Data type' button is highlighted. A dropdown menu is open, showing various options like 'Decimal number', 'Percentage', 'Date/Time', etc., with 'Currency' selected. To the right of the dropdown, there are buttons for 'Merge queries', 'Append queries', 'Combine files', 'Map to standard', 'CDM', and 'AI insights'. Below the dropdown, the 'Applied steps' section lists the steps taken: 'Source', 'Changed Type', 'Added custom', and 'Added custom 1'. The main area shows a table with six rows and eight columns. The last two columns, 'Total Sale before Discount' and 'Total Sale after Discount', have their data type icons (ABC123) highlighted in red, indicating they are currently undefined.

After defining the data types for both columns, you'll create a table that looks like the following image.

	Date	Country	Units	Unit Price	Discount	Total Sale before Discount	Total Sale after Discount
1	2/1/2020	Panama	40	1.25	0%	50	50
2	3/1/2020	Panama	50	2.25	5%	112.5	106.88
3	2/1/2020	USA	150	3.45	10%	517.5	465.75
4	3/1/2020	USA	175	3.85	15%	673.75	572.69
5	2/1/2020	Canada	100	2.75	7%	275	255.75
6	3/1/2020	Canada	90	2.95	8%	265.5	244.26

Modify an existing custom column

Power Query adds your custom column to the table and adds the **Added custom** step to the **Applied steps** list in **Query settings**.

The screenshot shows the Power Query Editor interface. On the left, there's a preview pane displaying a table with columns: Date, Country, Units, Unit Price, % Discount, \$ Total Sale before Discount, and \$ Total Sale after Discount. The preview shows data for six rows from 1 to 6. On the right, the 'Query settings' dialog box is open. It has fields for 'Name' (set to 'Query') and 'Entity type' (set to 'Custom'). Below these, the 'Applied steps' list is shown, containing the following steps:

- Source
- Changed Type
- Added custom
- Added custom 1

Red arrows point to the 'Added custom' and 'Added custom 1' steps in the list. At the bottom of the dialog box are 'Cancel' and 'Save & close' buttons.

To modify your custom column, select the **Added custom** step in the **Applied steps** list.

The **Custom column** dialog box appears with the custom column formula you created.

Next steps

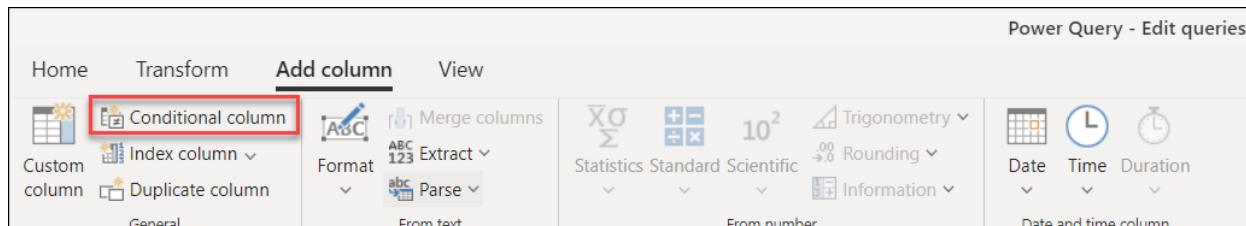
- You can create a custom column in other ways, such as creating a column based on examples you provide to Power Query Editor. More information: [Add a column from an example](#)
- For Power Query M reference information, go to [Power Query M function reference](#).

Add a conditional column

1/15/2022 • 2 minutes to read • [Edit Online](#)

With Power Query, you can create new columns whose values will be based on one or more conditions applied to other columns in your table.

The **Conditional column** command is located on the **Add column** tab, in the **General** group.



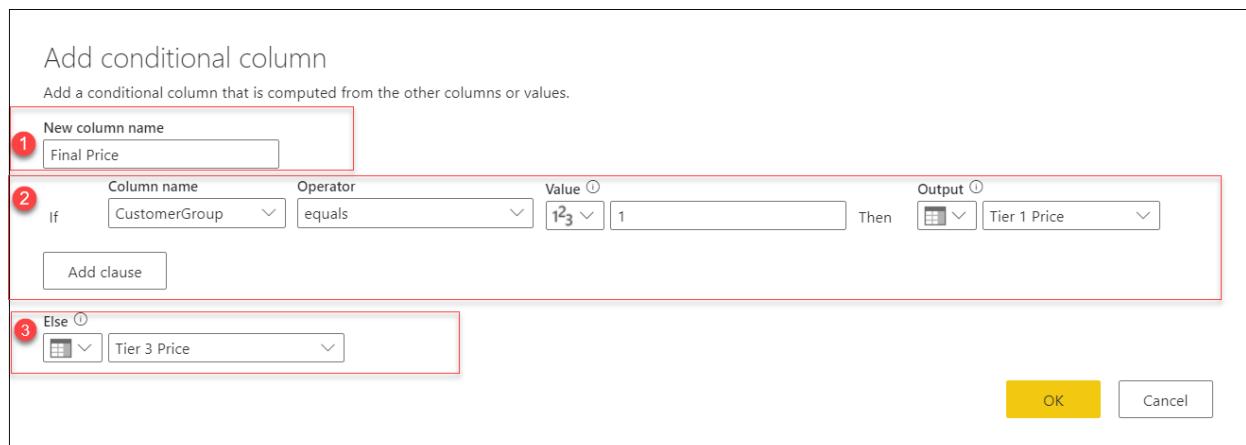
Adding a conditional column

In this example, you'll be using the table shown in the following image.

	Customer	CustomerGroup	Tier 1 Price	Tier 2 Price	Tier 3 Price	
1	USA-8431		1	5	8	14
2	PTY-507		4	10	15	26
3	USA-357		2	11	12	13
4	USA-8699		1	9	13	15
5	USA-0478		3	14	24	27

In this table, you have a field that gives you the **CustomerGroup**. You also have different prices applicable to that customer in the **Tier 1 Price**, **Tier 2 Price**, and **Tier 3 Price** fields. In this example, your goal is to create a new column with the name **Final Price** based on the value found in the **CustomerGroup** field. If the value in the **CustomerGroup** field is equal to 1, you'll want to use the value from the **Tier 1 Price** field; otherwise, you'll use the value from the **Tier 3 Price**.

To add this conditional column, select **Conditional column**. In the **Add conditional column** dialog box, you can define three sections numbered in the following image.



- New column name:** You can define the name of your new column. In this example, you'll use the name **Final Price**.
- Conditional clauses:** Here you define your conditional clauses. You can add more clauses by selecting **Add clause**. Each conditional clause will be tested on the order shown in the dialog box, from top to bottom. Each

clause has four parts:

- **Column name:** In the drop-down list, select the column to use for the conditional test. For this example, select **CustomerGroup**.
- **Operator:** Select the type of test or operator for the conditional test. In this example, the value from the **CustomerGroup** column has to be equal to 1, so select **equals**.
- **Value:** You can enter a value or select a column to be used for the conditional test. For this example, enter 1.
- **Output:** If the test is positive, the value entered here or the column selected will be the output. For this example, if the **CustomerGroup** value is equal to 1, your **Output** value should be the value from the **Tier 1 Price** column.

3. **Final Else clause:** If none of the clauses above yield a positive test, the output of this operation will be the one defined here, as a manually entered value or a value from a column. In this case, the output will be the value from the **Tier 3 Price** column.

The result of that operation will give you a new **Final Price** column.

	A ^B _C Customer	1 ² ₃ CustomerGroup	1 ² ₃ Tier 1 Price	1 ² ₃ Tier 2 Price	1 ² ₃ Tier 3 Price	ABC 1 ² ₃ Final Price
1	USA-8431		1	5	8	14
2	PTY-507		4	10	15	26
3	USA-357		2	11	12	13
4	USA-8699		1	9	13	15
5	USA-0478		3	14	24	27

NOTE

New conditional columns won't have a data type defined. You can add a new step to define a data type for this newly created column by following the steps described in [Data types in Power Query](#).

Adding and organizing multiple clauses

For this example, let's change your goal. Your new conditional clauses are:

- If the value from the **CustomerGroup** column is equal to 1, the **Output** will be the value from the **Tier 1 Price** column.
- If the value from the **CustomerGroup** column is equal to 2, the **Output** will be the value from the **Tier 2 Price** column.
- If none of the previous tests are positive, the **Output** will be the value from the **Tier 3 Price** column.

The screenshot shows the 'Add conditional column' dialog box. The 'New column name' is 'Final Price'. The first clause is 'If CustomerGroup equals 1 Then Tier 1 Price'. The second clause is 'Else if CustomerGroup equals 2 Then Tier 2 Price'. The third clause is 'Else Tier 3 Price'. The 'Name' field is set to 'Query' and 'Entity type' is set to 'Custom'. The 'Applied steps' pane shows 'Source' and 'Changed Type'.

NOTE

At the end of each clause, you can select the ellipsis button (...) to delete, move up, or move down the clause.

The result of that operation will give you the result that you're looking for.

	Customer	CustomerGroup	Tier 1 Price	Tier 2 Price	Tier 3 Price	Final Price
1	USA-8431		1	5	8	14
2	PTY-507		4	10	15	26
3	USA-357		2	11	12	13
4	USA-8699		1	9	13	15
5	USA-0478		3	14	24	27

Cluster values

1/15/2022 • 3 minutes to read • [Edit Online](#)

Cluster values automatically create groups with similar values using a fuzzy matching algorithm, and then maps each column's value to the best-matched group. This transform is very useful when you're working with data that has many different variations of the same value and you need to combine values into consistent groups.

Consider a sample table with an **id** column that contains a set of IDs and a **Person** column containing a set of variously spelled and capitalized versions of the names Miguel, Mike, William, and Bill.

	A ^B C id	A ^B C Person
1	1	miguel
2	2	Miguel
3	3	migueel
4	4	mike
5	5	Mike
6	6	William
7	7	Bill
8	8	billy
9	9	Miguel

In this example, the outcome you're looking for is a table with a new column that shows the right groups of values from the **Person** column and not all the different variations of the same words.

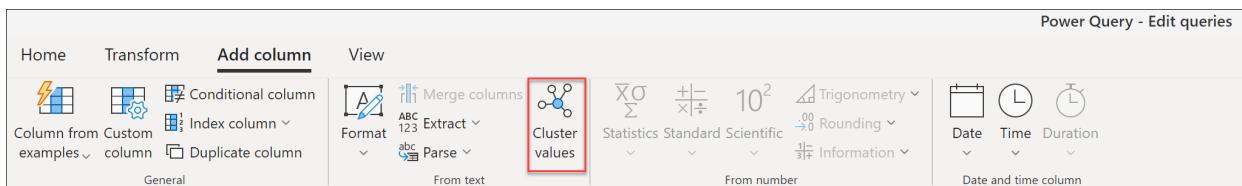
	12 id	ab Person	ab Cluster
1	1	miguel	Miguel
2	2	Miguel	Miguel
3	3	migueel	Miguel
4	4	mike	mike
5	5	Mike	mike
6	6	William	William
7	7	Bill	Bill
8	8	billy	Bill
9	9	Miguel	Miguel

NOTE

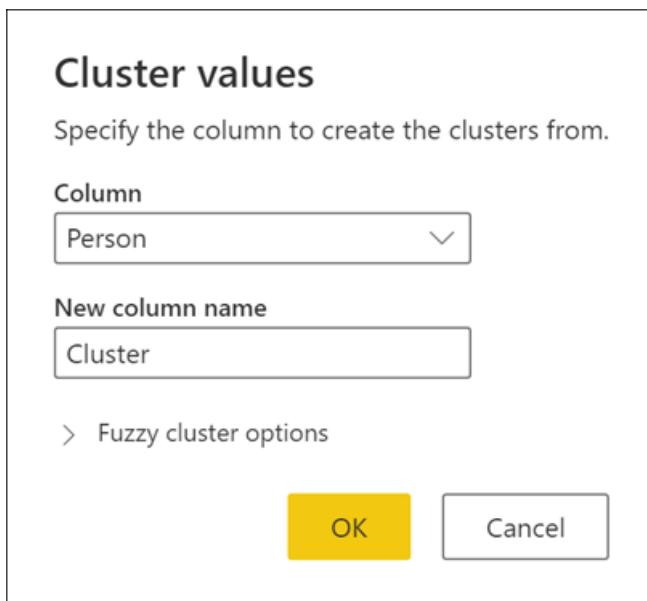
The Cluster values feature is available only for Power Query Online.

Create a Cluster column

To cluster values, first select the **Person** column, go to the **Add column** tab in the ribbon, and then select the **Cluster values** option.



In the **Cluster values** dialog box, confirm the column that you want to use to create the clusters from, and enter the new name of the column. For this case, name this new column **Cluster**.



The result of that operation yields the result shown in the next image.

	12	id	ab	Person	ab	Cluster
1		1	miguel	Miguel		
2		2	Miguel	Miguel		
3		3	migueel	Miguel		
4		4	mike	mike		
5		5	Mike	mike		
6		6	William	William		
7		7	Bill	Bill		
8		8	billy	Bill		
9		9	Miguel	Miguel		

NOTE

For each cluster of values, Power Query picks the most frequent instance from the selected column as the "canonical" instance. If multiple instances occur with the same frequency, Power Query picks the first one.

Using the fuzzy cluster options

The following options are available for clustering values in a new column:

- **Similarity threshold (optional)**: This option indicates how similar two values must be to be grouped together. The minimum setting of 0 causes all values to be grouped together. The maximum setting of 1 only allows values that match exactly to be grouped together. The default is 0.8.
- **Ignore case**: When comparing text strings, case is ignored. This option is enabled by default.
- **Group by combining text parts**: The algorithm tries to combine text parts (such as combining Micro and

soft into Microsoft) to group values.

- **Show similarity scores:** Shows similarity scores between the input values and computed representative values after fuzzy clustering.
- **Transformation table (optional):** You can select a transformation table that maps values (such as mapping MSFT to Microsoft) to group them together.

For this example, a new transformation table with the name **My transform table** is used to demonstrate how values can be mapped. This transformation table has two columns:

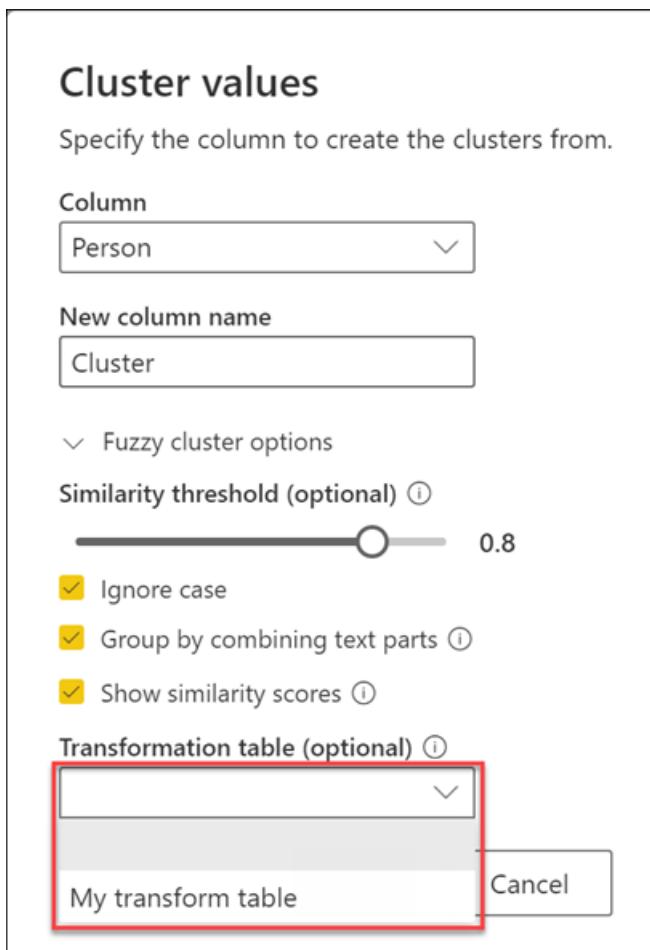
- **From:** The text string to look for in your table.
- **To:** The text string to use to replace the text string in the **From** column.

	A ^B _C From	A ^B _C To
1	mike	Miguel
2	William	Bill

IMPORTANT

It's important that the transformation table has the same columns and column names as shown in the previous image (they have to be named "From" and "To"), otherwise Power Query won't recognize this table as a transformation table, and no transformation will take place.

Using the previously created query, double-click the **Clustered values** step, then in the **Cluster values** dialog box, expand **Fuzzy cluster options**. Under *Fuzzy cluster options*, enable the **Show similarity scores** option. For **Transformation table (optional)**, select the query that has the transform table.



After selecting your transformation table and enabling the **Show similarity scores** option, select OK. The result of that operation will give you a table that contains the same id and **Person** columns as the original table,

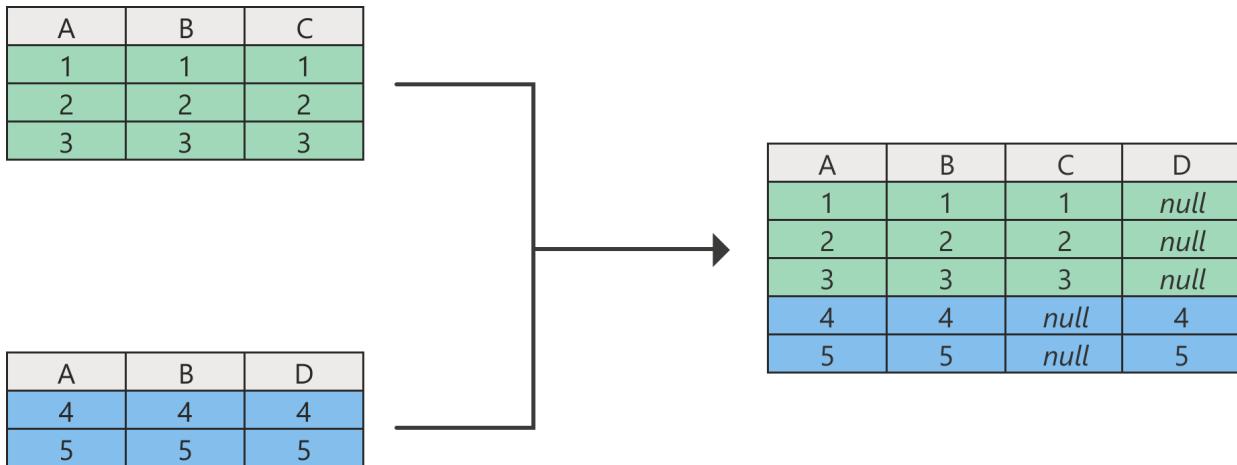
but also includes two new columns on the right called **Cluster** and **Person_Cluster_Similarity**. The **Cluster** column contains the properly spelled and capitalized versions of the names Miguel for versions of Miguel and Mike, and William for versions of Bill, Billy, and William. The **Person_Cluster_Similarity** column contains the similarity scores for each of the names.

	ab id	ab Person	ab Cluster	1.2 Person_Cluster_Similarity
1	1	miguel	Miguel	1
2	2	Miguel	Miguel	1
3	3	migueel	Miguel	0.88
4	4	mike	Miguel	0.95
5	5	Mike	Miguel	0.95
6	6	William	William	1
7	7	Bill	William	0.95
8	8	billy	William	0.91
9	9	Miguel	Miguel	1

Append queries

1/15/2022 • 2 minutes to read • [Edit Online](#)

The append operation creates a single table by adding the contents of one or more tables to another, and aggregates the column headers from the tables to create the schema for the new table.

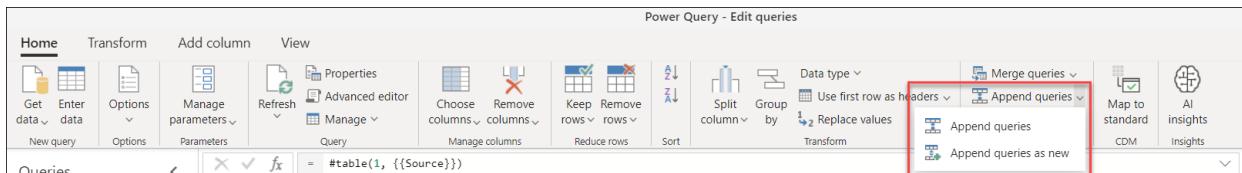


NOTE

When tables that don't have the same column headers are appended, all column headers from all tables are appended to the resulting table. If one of the appended tables doesn't have a column header from other tables, the resulting table shows *null* values in the respective column, as shown in the previous image in columns C and D.

You can find the **Append queries** command on the **Home** tab in the **Combine** group. On the drop-down menu, you'll see two options:

- **Append queries** displays the **Append** dialog box to add additional tables to the current query.
- **Append queries as new** displays the **Append** dialog box to create a new query by appending multiple tables.



The append operation requires at least two tables. The **Append** dialog box has two modes:

- **Two tables**: Combine two table queries together. This mode is the default mode.
- **Three or more tables**: Allow an arbitrary number of table queries to be combined.

NOTE

The tables will be appended in the order in which they're selected, starting with the **Primary table** for the **Two tables** mode and from the primary table in the **Tables to append** list for the **Three or more tables** mode.

Append two tables

For the example in this article, we'll use the following two tables with sample data:

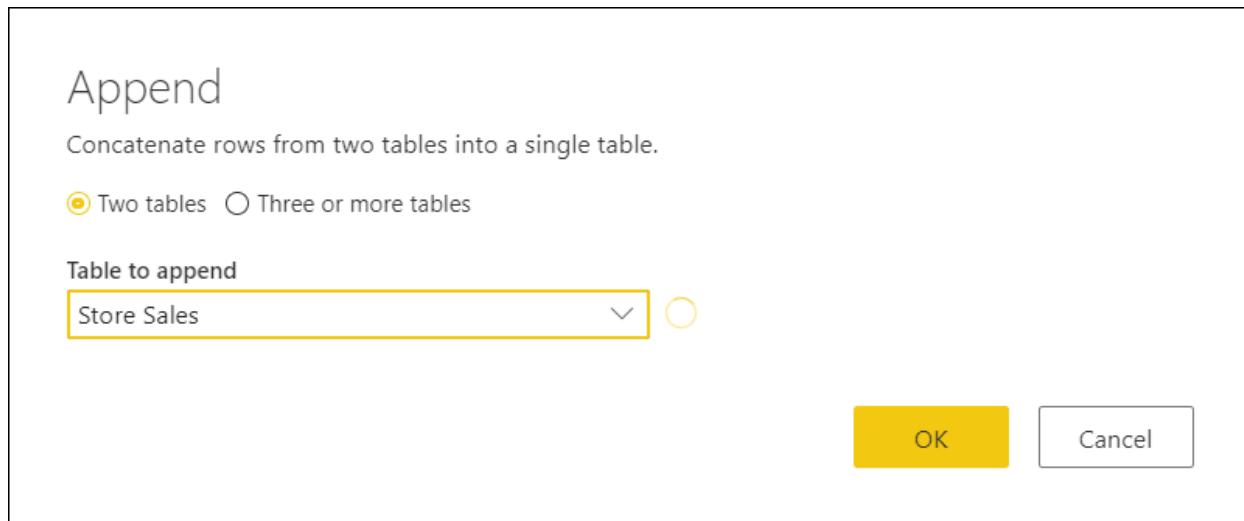
- **Online Sales:** Sales made through an online channel.

	Channel Name	Date	CustomerID	Units
1	Online	6/2/2020	US-187982	20
2	Online	6/2/2020	US-235789	30
3	Online	6/5/2020	US-187982	15
4	Online	6/5/2020	US-235789	40

- **Store Sales:** Sales made through the company's physical locations.

	Date	Units	Referer	CustomerID	Channel Name
1	6/4/2020	200	www.powerbi.com	e-78956	Store A
2	6/4/2020	300	www.powerquery.c...	e-78899	Store B
3	6/7/2020	100	www.xbox.com	e-75214	Store A
4	6/7/2020	70	www.microsoft.com	e-23658	Store B

To append these tables, first select the **Online Sales** table. On the Home tab, select **Append queries**, which creates a new step in the **Online Sales** query. The **Online Sales** table will be the primary table. The table to append to the primary table will be **Store Sales**.



Power Query performs the append operation based on the names of the column headers found on both tables, and not based on their relative position in the headers sections of their respective tables. The final table will have all columns from all tables appended.

In the event that one table doesn't have columns found in another table, *null*/values will appear in the corresponding column, as shown in the **Referer** column of the final query.

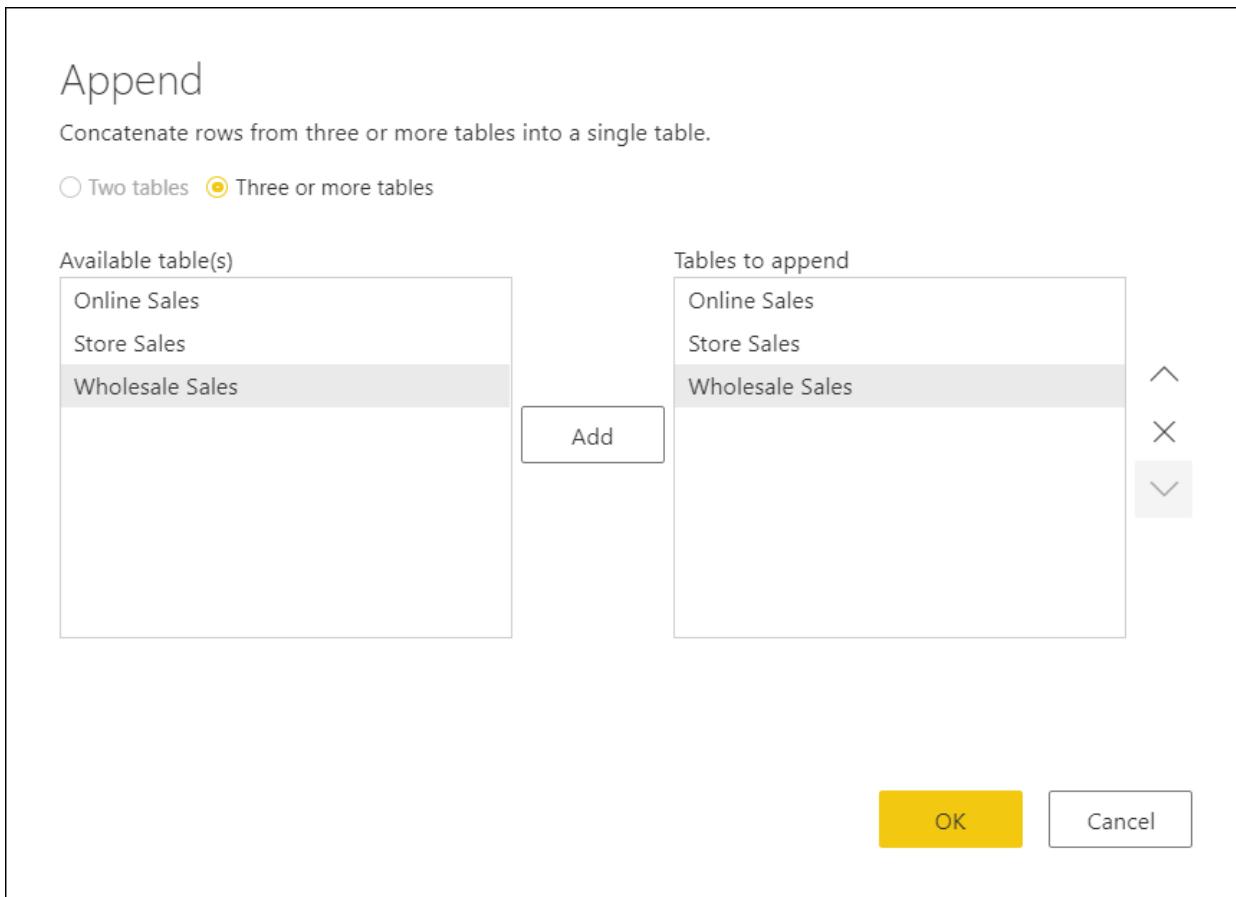
Channel Name	Date	CustomerID	Units	Referer
1 Online	6/2/2020	US-187982	20	null
2 Online	6/2/2020	US-235789	30	null
3 Online	6/5/2020	US-187982	15	null
4 Online	6/5/2020	US-235789	40	null
5 Store A	6/4/2020	e-78956	200	www.powerbi.com
6 Store B	6/4/2020	e-78899	300	www.powerquery.com
7 Store A	6/7/2020	e-75214	100	www.xbox.com
8 Store B	6/7/2020	e-23658	70	www.microsoft.com

Append three or more tables

In this example, you want to append not only the **Online Sales** and **Store Sales** tables, but also a new table named **Wholesale Sales**.

	Channel Name	Date	CustomerID	Units
1	Wholesale	6/1/2020	V-8884	5000
2	Wholesale	6/1/2020	V-1234	4000
3	Wholesale	6/1/2020	V-2943	6000
4	Wholesale	6/1/2020	V-8893	4000

The new approach for this example is to select **Append queries as new**, and then in the **Append** dialog box, select the **Three or more tables** option button. In the **Available table(s)** list, select each table you want to append, and then select **Add**. After all the tables you want appear in the **Tables to append** list, select **OK**.



After selecting **OK**, a new query will be created with all your tables appended.

Table.Combine({Online, #"Store Sales", Wholesale})

	Channel Name	Date	CustomerID	Units	Referer
1	Online	6/2/2020	US-187982	20	null
2	Online	6/2/2020	US-235789	30	null
3	Online	6/5/2020	US-187982	15	null
4	Online	6/5/2020	US-235789	40	null
5	Store A	6/4/2020	e-78956	200	www.powerbi.com
6	Store B	6/4/2020	e-78899	300	www.powerquery.com
7	Store A	6/7/2020	e-75214	100	www.xbox.com
8	Store B	6/7/2020	e-23658	70	www.microsoft.com
9	Wholesale	6/1/2020	V-8884	5000	null
10	Wholesale	6/1/2020	V-1234	4000	null
11	Wholesale	6/1/2020	V-2943	6000	null
12	Wholesale	6/1/2020	V-8893	4000	null

Combine files overview

1/15/2022 • 4 minutes to read • [Edit Online](#)

With Power Query, you can combine multiple files that have the same schema into a single logical table.

This feature is useful when you want to combine all the files you have in the same folder. For example, if you have a folder that contains monthly files with all the purchase orders for your company, you can combine these files to consolidate the orders into a single view.

Files can come from a variety of sources, such as (but not limited to):

- Local folders
- SharePoint sites
- Azure Blob storage
- Azure Data Lake Storage (Gen1 and Gen2)

When working with these sources, you'll notice that they share the same table schema, commonly referred to as the *file system view*. The following screenshot shows an example of the file system view.

	Content	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
1	[Binary]	.onetoc2	null	12/15/2014, 10:52:00 ...	10/31/2014, 9:07:00 PM	[Record]	
2	[Binary]	.one	null	10/31/2014, 9:29:00 PM	10/31/2014, 9:07:00 PM	[Record]	
3	[Binary]	.one	null	12/5/2014, 12:57:00 PM	10/31/2014, 9:28:00 PM	[Record]	
4	[Binary]	.pptm	null	2/13/2015, 10:01:00 AM	11/1/2014, 2:17:00 PM	[Record]	
5	[Binary]	.one	null	11/8/2014, 3:36:00 PM	11/8/2014, 3:30:00 PM	[Record]	
6	[Binary]	.one	null	11/28/2014, 12:47:00 ...	11/10/2014, 6:30:00 PM	[Record]	
7	[Binary]	.one	null	12/8/2014, 10:29:00 AM	12/5/2014, 12:57:00 PM	[Record]	
8	[Binary]	.one	null	12/15/2014, 11:41:00 ...	12/15/2014, 10:52:00 ...	[Record]	
9	[Binary]	.xlsx	null	1/12/2015, 2:27:00 AM	1/12/2015, 2:06:00 AM	[Record]	
10	[Binary]	.xlsx	null	8/17/2019, 5:13:00 PM	9/2/2015, 6:07:00 PM	[Record]	
11	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
12	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
13	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
14	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
15	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
16	[Binary]	.xlsx	null	9/2/2015, 6:07:00 PM	9/2/2015, 6:07:00 PM	[Record]	
17	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
18	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
19	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
20	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
21	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
22	[Binary]	.csv	null	1/9/2016, 1:52:00 PM	1/9/2016, 1:52:00 PM	[Record]	
23	[Binary]	.xlsx	null	2/5/2016, 12:47:00 PM	2/5/2016, 12:47:00 PM	[Record]	
24							

In the file system view, the **Content** column contains the binary representation of each file.

NOTE

You can filter the list of files in the file system view by using any of the available fields. It's good practice to filter this view to show only the files you need to combine, for example by filtering fields such as **Extension** or **Folder Path**. More information: [Folder](#)

Selecting any of the [Binary] values in the **Content** column automatically creates a series of navigation steps to that specific file. Power Query will try to interpret the binary by using one of the available connectors, such as Text/CSV, Excel, JSON, or XML.

Combining files takes place in the following stages:

- [Table preview](#)
- [Combine files dialog box](#)
- [Combined files output](#)

Table preview

When you connect to a data source by using any of the previously mentioned connectors, a table preview opens. If you're certain that you want to combine all the files in the folder, select **Combine** in the lower-right corner of the screen.

The screenshot shows the 'Content' table in the Power Query editor. The columns are: Content, Name, Extension, Date accessed, Date modified, Date created, Attributes, and Folder Path. The 'Content' column contains binary file types like .xlsx, .pptx, .pdf, etc. The 'Name' column lists file names such as '01-January.csv', '02-February.csv', etc. The 'Extension' column shows file extensions like .CSV, .PDF, etc. The 'Date accessed' column shows dates like 6/2/2016, 2/10/2016, etc. The 'Date modified' and 'Date created' columns show times like 10:31:51 AM, 1:48:29 PM, etc. The 'Attributes' column shows record types like [Record]. The 'Folder Path' column shows folder paths like 'C:\Users\Public\Documents\Power Query\Content'. At the bottom right of the table, there are buttons for 'Back', 'Cancel', 'Combine', and 'Transform data'.

Alternatively, you can select **Transform data** to access the Power Query Editor and create a subset of the list of files (for example, by using filters on the folder path column to only include files from a specific subfolder). Then combine files by selecting the column that contains the binaries in the **Content** column and then selecting either:

- The **Combine files** command in the **Combine** group on the **Home** tab.

The screenshot shows the Power Query ribbon with the 'Home' tab selected. In the 'Combine' group, the 'Combine files' icon is highlighted with a red box. Other icons in the group include 'Merge queries', 'Append queries', 'Map to standard', and 'AI insights'.

- The **Combine files** icon in the column header of the column that contains [Binary] values.

The screenshot shows the Power Query table with the 'Content' column header highlighted. The 'Content' column header has a red box around it, indicating it is the target for combining files. The other columns are: Name, Extension, Date accessed, Date modified, Date created, and Attributes.

Combine files dialog box

After you select the **Combine** or **Combine files** command, the **Combine files** dialog box opens and the following occurs:

1. Power Query analyzes the example file (by default, the first file in the list) and determines the correct file connector to use to open that file.
2. The dialog box provides the file connector experience exactly as if you were to connect directly to that example file.

- If you want to use a different file for the example file, you can choose it from the **Example file** drop-down menu.
- Optional: You can select **Skip files with errors** to exclude from the final output any files that result in errors.

In the following image, Power Query has detected that the first file has a .csv file name extension, so it uses the [Text/CSV](#) connector to interpret the file.

Combined files output

After the **Combine files** process is finished, Power Query automatically performs the following actions:

1. Creates an example query that performs all the required extraction steps for a single file. It uses the file that was selected as the example file in the **Combine files** dialog box.

This example query has the name **Transform Sample file** in the **Queries** pane.

2. Creates a function query that parameterizes the file/binary input to the example query. The example query and the function query are linked, so that changes to the example query are reflected in the function query.

These queries are listed in the **Helper queries** group.

3. Applies the function query to the original query with input binaries (for example, the folder query) so it applies the function query for binary inputs on each row, and then expands the resulting data extraction as top-level columns.

4. Creates a new group with the prefix **Transform file from** and the initial query as the suffix, and organizes all the components used to create these combined files in that group.

The screenshot shows the Power Query - Edit queries interface. In the top navigation bar, 'Home' is selected. The 'Queries' pane on the left lists several items, including 'Transform from Query1' (with a red box around it), 'Helper queries', 'Parameter', 'Transform file', and 'Query1'. The main area displays a table titled 'Table.TransformColumnTypes(#"Expanded table column", {"Source.Name", type text}, {"Month", type text}, {"Name", type text})'. The table has columns: Source.Name, Month, Name, 1.2 Australia, 1.2 Canada, 1.2 Central, 1.2 France, 1.2 Germany, and 1.2 Northeast. The data consists of 24 rows of product information. On the right side, there are 'Query settings' and 'Applied steps' sections.

You can easily combine all files within a given folder, as long as they have the same file type and structure (including the same columns). You can also apply additional transformation or extraction steps by modifying the automatically generated example query, without having to worry about modifying or creating additional function query steps.

NOTE

You can modify the steps inside the example query to change the function applied to each binary in your query. The example query is linked to the function, so any changes made to the example query will be reflected in the function query.

If any of the changes affect column names or column data types, be sure to check the last step of your output query.

Adding a **Change column type** step can introduce a step-level error that prevents you from visualizing your table.

More information: [Dealing with errors](#)

See also

[Combine CSV files](#)

Combine CSV files

1/15/2022 • 5 minutes to read • [Edit Online](#)

In Power Query, you can combine multiple files from a given data source. This article describes how the experience works when the files that you want to combine are CSV files. More information: [Combine files overview](#)

TIP

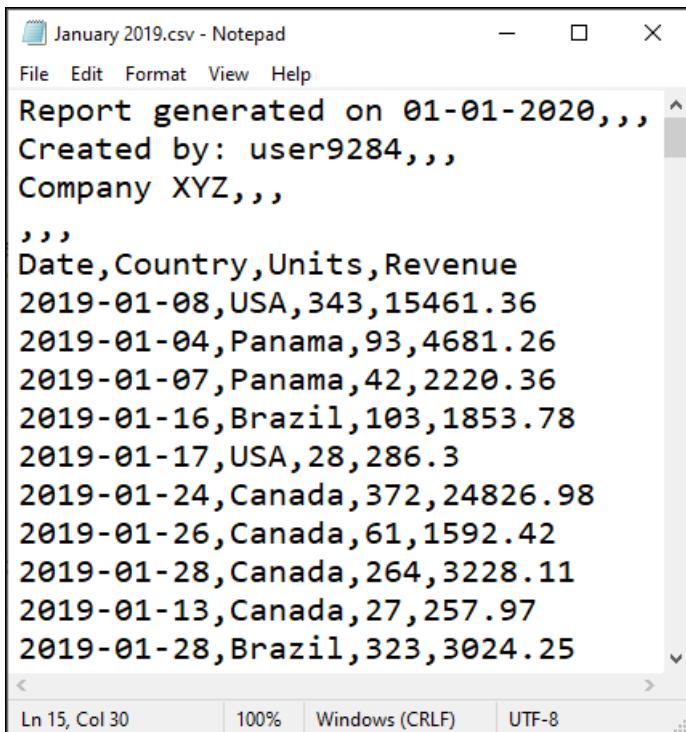
You can follow along with this example by downloading the sample files used in this article from [this download link](#). You can place those files in the data source of your choice, such as a local folder, SharePoint folder, Azure Blob storage, Azure Data Lake Storage, or other data source that provides the file system view.

For simplicity, the example in this article uses the Folder connector. More information: [Folder](#)

About the sample files used

To combine files, it's imperative that they all have the same structure and the same extension. All the files used in this example have the same structure and extension (.csv).

There are 12 CSV files, one for each month of the calendar year 2019. The following image shows the first 15 rows of the file for the month of January.



The screenshot shows a Notepad window titled "January 2019.csv - Notepad". The window contains the following text:

```
Report generated on 01-01-2020,,,  
Created by: user9284,,,  
Company XYZ,,,  
,,,  
Date,Country,Units,Revenue  
2019-01-08,USA,343,15461.36  
2019-01-04,Panama,93,4681.26  
2019-01-07,Panama,42,2220.36  
2019-01-16,Brazil,103,1853.78  
2019-01-17,USA,28,286.3  
2019-01-24,Canada,372,24826.98  
2019-01-26,Canada,61,1592.42  
2019-01-28,Canada,264,3228.11  
2019-01-13,Canada,27,257.97  
2019-01-28,Brazil,323,3024.25
```

The status bar at the bottom of the Notepad window shows "Ln 15, Col 30" and "100% Windows (CRLF) UTF-8".

The number of rows varies from file to file, but all files have a header section in the first four rows. They have column headers in the fifth row, and the data for the table begins in the sixth row and continues through all subsequent rows.

The goal is to combine all 12 files into a single table. This combined table contains the header row at the top of the table, and includes the source name, date, country, units, and revenue data for the entire year in separate columns after the header row.

The screenshot shows the Power Query - Edit queries dialog box. The left pane displays a list of queries: 'Transform file from...' [2], 'Helper queries' [3], 'Transform Sample file', and 'CSV Files'. The main area shows a table preview with columns: Source.Name, Date, Country, Units, and Revenue. The table contains 19 rows of data. The right pane shows 'Query settings' with 'Name' set to 'CSV Files' and 'Entity type' set to 'Custom'. Under 'Applied steps', several steps are listed, including 'Source', 'Filtered hidden files', 'Invoke custom fun...', 'Renamed columns', 'Removed other col...', 'Expanded table co...', and 'Changed column t...'. At the bottom are 'Cancel' and 'Save & close' buttons.

Table preview

When connecting to the folder that hosts the files that you want to combine—in this example, the name of that folder is **CSV Files**—you're shown the table preview dialog box, which displays your folder path in the upper-left corner. The data preview shows the file system view.

The screenshot shows the Power Query - Preview folder data dialog box. The path 'C:\Users\[REDACTED]\Documents\CSV Files' is shown in the top left. The main area is a table with columns: Content, Name, Extension, Date accessed, Date modified, Date created, Attributes, and Folder Path. The table lists 12 CSV files from April to September 2019. At the bottom are 'Back', 'Cancel', 'Combine', and 'Transform data' buttons.

For this example, select **Combine**.

NOTE

In a different situation, you might select **Transform data** to further filter and transform your data before combining the files. Selecting **Combine** is only recommended when you're certain that the folder contains only the files that you want to combine.

Combine files dialog box

After you select **Combine** in the table preview, the **Combine files** dialog box appears.

Power Query - Combine files

Select the object to be extracted from each file. [Learn more](#)

Example file:

First file

File origin	Delimiter	Data type detection	
1252: Western Europe...	Comma	Based on first 200 rows	
A^B_C Column1	A^B_C Column2	A^B_C Column3	A^B_C Column4

Report generated on 01-01-20...

Created by: user9284

Company XYZ

Date Country Units Revenue

2019-04-22 Brazil 153 2649.32

2019-04-14 Brazil 57 940.4

2019-04-26 Colombia 310 4408.22

2019-04-25 USA 90 2044.18

2019-04-23 Panama 204 16330.85

2019-04-07 USA 356 3772.26

2019-04-11 Colombia 122 2490.96

2019-04-27 Colombia 367 19933.19

2019-04-24 Panama 223 13834.04

2019-04-16 Colombia 159 3448.16

2019-04-09 Canada 258 11601.31

Skip files with errors

Back Cancel Transform data

NOTE

Power Query automatically detects what connector to use based on the first file found in the list. To learn more about the CSV connector, see [Text/CSV](#).

For this example, leave all the default settings (Example file set to First file, and the default values for File origin, Delimiter, and Data type detection).

Now select **Transform data** in the lower-right corner to go to the output query.

Output query

After selecting **Transform data** in the **Combine files** dialog box, you'll be taken back to the Power Query Editor in the query that you initially created from the connection to the local folder. The output query now contains the source file name in the left-most column, along with the data from each of the source files in the remaining columns.

The screenshot shows the Power Query interface with the 'Edit queries' window open. The 'Transform Sample file' query is selected. The table preview shows 19 rows of data from April 2019.csv. The 'Applied steps' pane on the right lists 'Changed column type'.

However, the data isn't in the correct shape. You need to remove the top four rows from each file before combining them. To make this change in each file before you combine them, select the **Transform Sample file** query in the **Queries** pane on the left side of your screen.

Modify the Transform Sample file query

In this **Transform Sample file** query, the values in the **Date** column indicate that the data is for the month of April, which has the year-month-day (YYYY-MM-DD) format. April 2019.csv is the first file that's displayed in the table preview.

The screenshot shows the Power Query interface with the 'Edit queries' window open. The 'Transform Sample file' query is selected. The table preview shows the correct data starting from row 5. The 'Applied steps' pane on the right lists 'Changed column type'.

You now need to apply a new set of transformations to clean the data. Each transformation will be automatically converted to a function inside the **Helper queries** group that will be applied to every file in the folder before combining the data from each file.

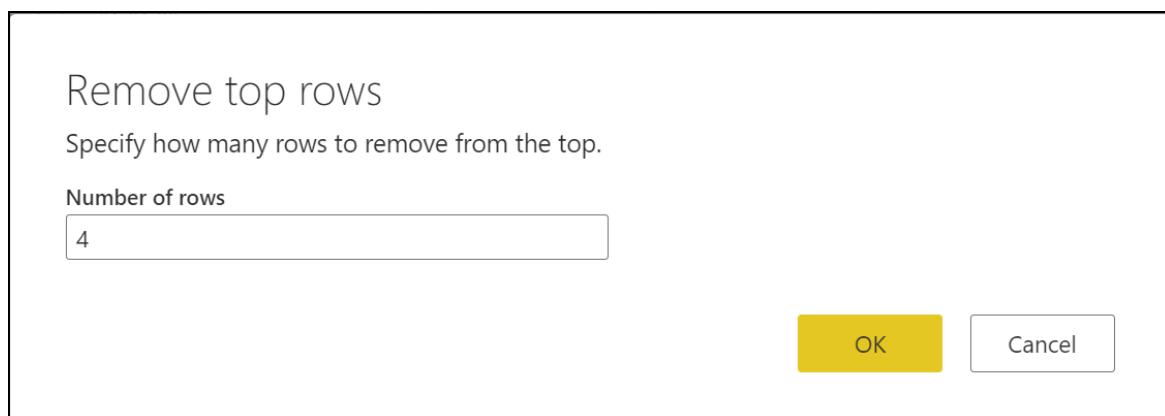
The transformations that need to be added to the **Transform Sample file** query are:

- Remove top rows:** To perform this operation, select the table icon menu in the upper-left corner of the table, and then select **Remove top rows**.

The screenshot shows the Power Query ribbon with the 'Column1' tab selected. A context menu is open over the first four rows of the table, listing various options like 'Use first row as headers' and 'Remove top rows'. The 'Remove top rows' option is highlighted with a red box.

Country	Units
Brazil	153
Brazil	57
Colombia	310
USA	90
Panama	204
USA	356
Colombia	122
Colombia	367
Panama	223
Colombia	159
Canada	258
Panama	325
Colombia	349
Panama	139

In the **Remove top rows** dialog box, enter **4**, and then select **OK**.



After selecting **OK**, your table will no longer have the top four rows.

Power Query - Edit queries

Home Transform Add column View

Get data Enter data Options Manage parameters Refresh Properties Advanced editor Manage... Choose columns Remove columns Keep rows Remove rows Sort Data type: Text Use first row as headers... Split column Group by Reduce rows Combine Map to standard CDM AI insights Insights

Queries

= Table.Skip(#"Changed column type", 4)

	A ^B Column1	A ^B Column2	A ^B Column3	A ^B Column4
1	Date	Country	Units	Revenue
2	2019-04-22	Brazil	153	2649.32
3	2019-04-14	Brazil	57	940.4
4	2019-04-26	Colombia	310	4408.22
5	2019-04-25	USA	90	2044.18
6	2019-04-23	Panama	204	16330.85
7	2019-04-07	USA	356	3772.26
8	2019-04-11	Colombia	122	2490.96
9	2019-04-27	Colombia	367	19933.19
10	2019-04-24	Panama	223	13834.04
11	2019-04-16	Colombia	159	3448.16
12	2019-04-08	Canada	258	14601.34
13	2019-04-14	Panama	325	11939.47
14	2019-04-01	Colombia	349	10844.36
15	2019-04-07	Panama	139	2890.93
16	2019-04-14	Colombia	360	22928.71
17	2019-04-03	Panama	69	2074.68
18	2019-04-17	Panama	361	16159.8
19	2019-04-30	Canada	26	2178.03

Query settings

Name: Transform Sample file Entity type: Custom

Applied steps

- Source
- Changed column type...
- Removed top rows

Cancel Save & close

2. **Use first row as headers:** Select the table icon again, and then select **Use first row as headers**.

Column1 Column2 Column3 Column4

Use first row as headers (highlighted with a red box)

	Country	Units	Revenue
1	Brazil	153	2649.32
2	Brazil	57	940.4
3	Colombia	310	4408.22
4	USA	90	2044.18
5	Panama	204	16330.85
6	USA	356	3772.26
7	Colombia	122	2490.96
8	Colombia	367	19933.19
9	Panama	223	13834.04
10	Colombia	159	3448.16
11	Canada	258	14601.34
12	Panama	325	11939.47
13	Colombia	349	10844.36
14	Panama	139	2890.93
15	Colombia	360	22928.71
16	Panama	69	2074.68
17	Panama	361	16159.8
18	Canada	26	2178.03

The result of that operation will promote the first row of the table to the new column headers.

The screenshot shows the Power Query interface with the 'Transform Sample file' query selected. The ribbon tabs include Home, Transform, Add column, and View. The 'Transform' tab is active. The main area displays a table with columns: Date, Country, Units, and Revenue. The 'Applied steps' pane on the right lists several steps: Source, Changed column t..., Removed top rows, Promoted headers, and a step that has been deleted (X). Buttons at the bottom right include 'Cancel' and 'Save & close'.

After this operation is completed, Power Query by default will try to automatically detect the data types of the columns and add a new **Changed column type** step.

Revising the output query

When you go back to the **CSV Files** query, you'll notice that the last step is giving you an error that reads "The column 'Column1' of the table wasn't found." The reason behind this error is that the previous state of the query was doing an operation against a column named **Column1**. But because of the changes made to the **Transform Sample file** query, this column no longer exists. More information: [Dealing with errors in Power Query](#)

The screenshot shows the Power Query interface with the 'CSV Files' query selected. The ribbon tabs are identical to the previous screenshot. The main area shows an error message: 'The column 'Column1' of the table wasn't found.' The 'Applied steps' pane lists several steps: Source, Filtered hidden files, Invoke custom fun..., Renamed columns, Removed other col..., Expanded table co..., and a step that has been deleted (X). A warning icon at the bottom left indicates '1 warning'. Buttons at the bottom right include 'Cancel' and 'Save & close'.

You can remove this last step of the query from the **Applied steps** pane by selecting the X delete icon on the left side of the name of the step. After deleting this step, your query will show the correct results.

Power Query - Edit queries

Queries

- Transform file fro... [2]
- Helper queries [3]
 - Transform Sample file
- CSV Files

Table.ExpandTableColumn(#"Removed other columns", "Transform")

Source.Name	Date	Country	Units	Revenue
1 April 2019.csv	4/22/2019	Brazil	153	2649.32
2 April 2019.csv	4/14/2019	Brazil	57	940.4
3 April 2019.csv	4/26/2019	Colombia	310	4408.22
4 April 2019.csv	4/25/2019	USA	90	2044.18
5 April 2019.csv	4/23/2019	Panama	204	16330.85
6 April 2019.csv	4/7/2019	USA	356	3772.26
7 April 2019.csv	4/11/2019	Colombia	122	2490.96
8 April 2019.csv	4/27/2019	Colombia	367	19933.19
9 April 2019.csv	4/24/2019	Panama	223	13834.04
10 April 2019.csv	4/16/2019	Colombia	159	3448.76
11 April 2019.csv	4/8/2019	Canada	258	14601.34
12 April 2019.csv	4/14/2019	Panama	325	11939.47
13 April 2019.csv	4/1/2019	Colombia	349	10844.36
14 April 2019.csv	4/7/2019	Panama	139	2890.93
15 April 2019.csv	4/14/2019	Colombia	360	22928.71
16 April 2019.csv	4/3/2019	Panama	69	2074.68
17 April 2019.csv	4/17/2019	Panama	361	16159.8
18 April 2019.csv	4/30/2019	Canada	26	2178.03
19 April 2019.csv	4/16/2019	Brazil	387	9041.12

Query settings

Name: CSV Files

Entity type: Custom

Applied steps:

- Source
- Filtered hidden files
- Invoke custom fun...
- Renamed columns
- Removed other col...
- Expanded table co...

1 warning

Cancel Save & close

However, notice that none of the columns derived from the files (Date, Country, Units, Revenue) have a specific data type assigned to them. Assign the correct data type to each column by using the following table.

COLUMN NAME	DATA TYPE
Date	Date
Country	Text
Units	Whole number
Revenue	Currency

After defining the data types for each column, you'll be ready to load the table.

Power Query - Edit queries X

Home Transform Add column View
Get data Enter data Options Manage parameters Refresh Properties Advanced editor
Choose columns Remove columns Keep rows Remove rows Sort
Transform Combine Map to standard CDM AI insights

Queries <

- Transform file fro... [2]
- Helper queries [3]
- Transform Sample file
- CSV Files

$$= \text{Table.TransformColumnTypes}(\#"\text{Expanded table column}", \{{"\text{Date}"},$$

Source.Name	Date	Country	Units	Revenue
1 April 2019.csv	4/22/2019	Brazil	153	2,649.32
2 April 2019.csv	4/14/2019	Brazil	57	940.4
3 April 2019.csv	4/26/2019	Colombia	310	4,408.22
4 April 2019.csv	4/25/2019	USA	90	2,044.18
5 April 2019.csv	4/23/2019	Panama	204	16,330.85
6 April 2019.csv	4/7/2019	USA	356	3,772.26
7 April 2019.csv	4/11/2019	Colombia	122	2,490.96
8 April 2019.csv	4/27/2019	Colombia	367	19,933.19
9 April 2019.csv	4/24/2019	Panama	223	13,834.04
10 April 2019.csv	4/16/2019	Colombia	159	3,448.16
11 April 2019.csv	4/8/2019	Canada	258	14,601.34
12 April 2019.csv	4/14/2019	Panama	325	11,939.47
13 April 2019.csv	4/1/2019	Colombia	349	10,844.36
14 April 2019.csv	4/7/2019	Panama	139	2,890.93
15 April 2019.csv	4/14/2019	Colombia	360	22,928.71
16 April 2019.csv	4/3/2019	Panama	69	2,074.68
17 April 2019.csv	4/17/2019	Panama	361	16,159.8
18 April 2019.csv	4/30/2019	Canada	26	2,178.03
19 April 2019.csv	4/16/2019	Brazil	387	9,041.12

Cancel
Save & close

NOTE

To learn how to define or change column data types, see [Data types](#).

Verification

To validate that all files have been combined, you can select the filter icon on the **Source.Name** column heading, which will display all the names of the files that have been combined. If you get the warning "List may be incomplete," select **Load more** at the bottom of the menu to display more available values in the column.

	Source.Name	Date	Country	Units	Revenue
1	April 2019.csv				
2	April 2019.csv				
3	April 2019.csv				
4	April 2019.csv				
5	April 2019.csv				
6	April 2019.csv				
7	April 2019.csv				
8	April 2019.csv				
9	April 2019.csv				
10	April 2019.csv				
11	April 2019.csv				
12	April 2019.csv				
13	April 2019.csv				
14	April 2019.csv				
15	April 2019.csv				
16	April 2019.csv				
17	April 2019.csv				
18	April 2019.csv				
19	April 2019.csv				
20	April 2019.csv	4/4/2019	Canada	222	7,975.43

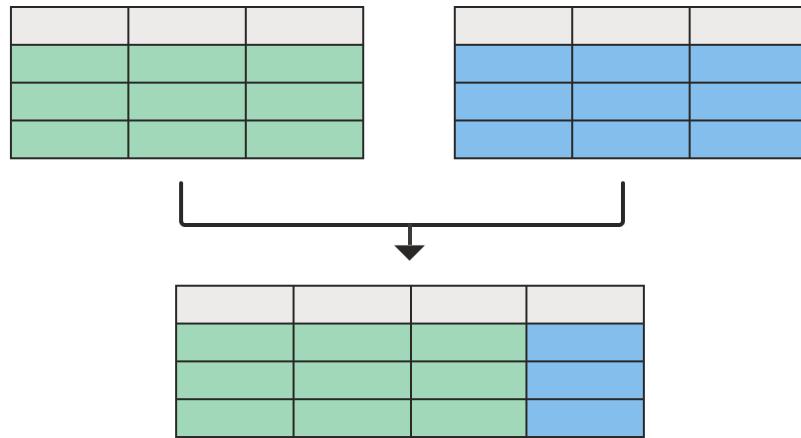
After you select **Load more**, all available file names will be displayed.

	Source.Name	Date	Country	Units	Revenue
1	April 2019.csv				
2	April 2019.csv				
3	April 2019.csv				
4	April 2019.csv				
5	April 2019.csv				
6	April 2019.csv				
7	April 2019.csv				
8	April 2019.csv				
9	April 2019.csv				
10	April 2019.csv				
11	April 2019.csv				
12	April 2019.csv				
13	April 2019.csv				
14	April 2019.csv				
15	April 2019.csv				
16	April 2019.csv				
17	April 2019.csv				
18	April 2019.csv	4/4/2019	Canada	222	7,975.43
19	April 2019.csv	4/8/2019	Brazil	23	4.18

Merge queries overview

1/15/2022 • 4 minutes to read • [Edit Online](#)

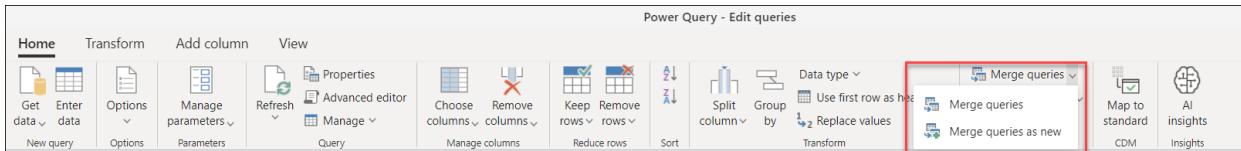
A merge queries operation joins two existing tables together based on matching values from one or multiple columns. You can choose to use different types of joins, depending on the output you want.



Merging queries

You can find the **Merge queries** command on the **Home** tab, in the **Combine** group. From the drop-down menu, you'll see two options:

- **Merge queries**: Displays the **Merge** dialog box, with the selected query as the left table of the merge operation.
- **Merge queries as new**: Displays the **Merge** dialog box without any preselected tables for the merge operation.



Identify tables for merging

The merge operation requires two tables:

- **Left table for merge**: The first selection, from top to bottom of your screen.
- **Right table for merge**: The second selection, from top to bottom of your screen.

Merge

Select tables and matching columns to create a merged table.

Left table for merge



No items selected for preview

Right table for merge



No items selected for preview

Join kind



Use fuzzy matching to perform the merge

➤ Fuzzy matching options

OK

Cancel

NOTE

The position—left or right—of the tables becomes very important when you select the correct join kind to use.

Select column pairs

After you've selected both the left and right tables, you can select the columns that drive the join between the tables. In the example below, there are two tables:

- **Sales:** The CountryID field is a key or an identifier from the Countries table.
- **Countries:** This table contains the CountryID and the name of the country.

Merge

Select tables and matching columns to create a merged table.

Left table for merge

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

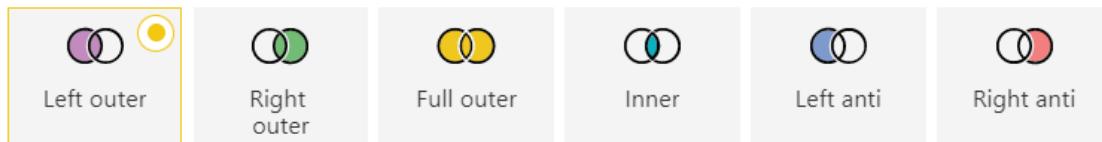


Right table for merge

CountryID	Country
1	USA
2	Canada
3	Panama



Join kind



Use fuzzy matching to perform the merge

➤ Fuzzy matching options

✓ The selection matches 4 of 4 rows from the first table

OK

Cancel

Merge dialog box with the Left table for merge set to Sales and the CountryID column selected, and the Right table for merge set to Countries and the CountryID column selected.

The goal is to join these tables by using the **CountryID** column from both tables, so you select the **CountryID** column from each table. After you make the selections, a message appears with an estimated number of matches at the bottom of the dialog box.

NOTE

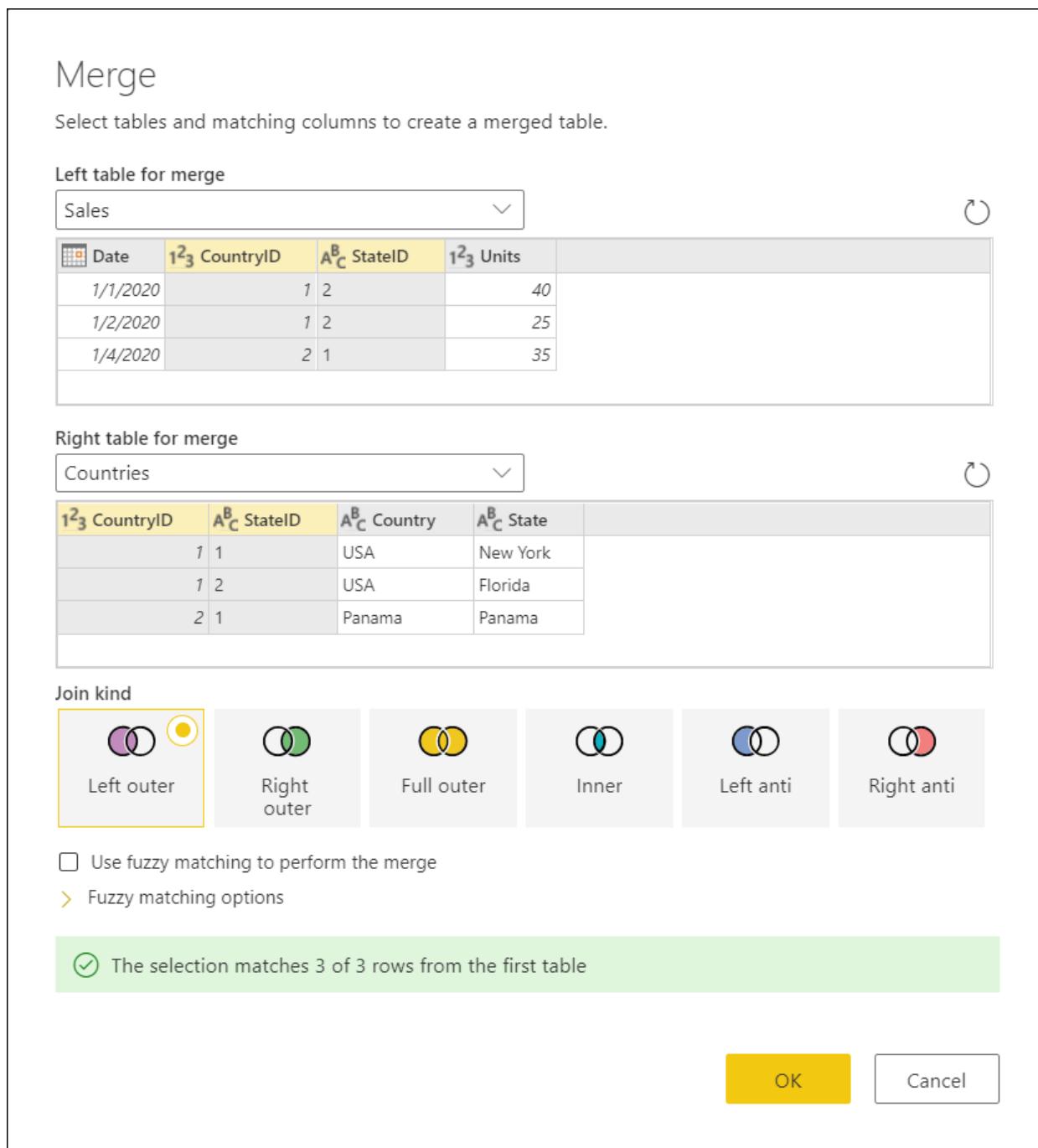
Although this example shows the same column header for both tables, this isn't a requirement for the merge operation. Column headers don't need to match between tables. However, it's important to note that the columns must be of the same data type, otherwise the merge operation might not yield correct results.

You can also select multiple columns to perform the join by selecting **Ctrl** as you select the columns. When you do so, the order in which the columns were selected is displayed in small numbers next to the column headings, starting with 1.

For this example, you have the **Sales** and **Countries** tables. Each of the tables has **CountryID** and **StateID**

columns, which you need to pair for the join between both columns.

First select the **CountryID** column in the **Sales** table, select **Ctrl**, and then select the **StateID** column. (This will show the small numbers in the column headings.) Next, perform the same selections in the **Countries** table. The following image shows the result of selecting those columns.



![Merge dialog box with the Left table for merge set to Sales, with the CountryID and StateID columns selected, and the Right table for merge set to Countries, with the CountryID and StateID columns selected. The Join kind is set to Left outer.]

Expand or aggregate the new merged table column

After selecting OK in the **Merge** dialog box, the base table of your query will have all the columns from your left table. Also, a new column will be added with the same name as your right table. This column holds the values corresponding to the right table on a row-by-row basis.

From here, you can choose to expand or aggregate the fields from this new table column, which will be the fields from your right table.

Table showing the merged Countries column on the right, with all rows containing a Table. The expand icon on the right of the Countries column header has been selected, and the expand menu is open. The expand menu has the Select all, CountryID, StateID, Country, and State selections selected. The Use original column name as prefix is also selected.

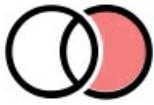
NOTE

Currently, the Power Query Online experience only provides the expand operation in its interface. The option to aggregate will be added later this year.

Join kinds

A *join kind* specifies how a merge operation will be performed. The following table describes the available join kinds in Power Query.

JOIN KIND	ICON	DESCRIPTION
Left outer		All rows from the left table, matching rows from the right table
Right outer		All rows from the right table, matching rows from the left table
Full outer		All rows from both tables
Inner		Only matching rows from both tables

JOIN KIND	ICON	DESCRIPTION
Left anti		Only rows from the left table
Right anti		Only rows from the right table

Fuzzy matching

You use fuzzy merge to apply fuzzy matching algorithms when comparing columns, to try to find matches across the tables you're merging. You can enable this feature by selecting the **Use fuzzy matching to perform the merge** check box in the **Merge** dialog box. Expand **Fuzzy matching options** to view all available configurations.

NOTE

Fuzzy matching is only supported for merge operations over text columns.

Left outer join

1/15/2022 • 2 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is a *left outer join*, which keeps all the rows from the left table and brings in any matching rows from the right table. More information: [Merge operations overview](#)

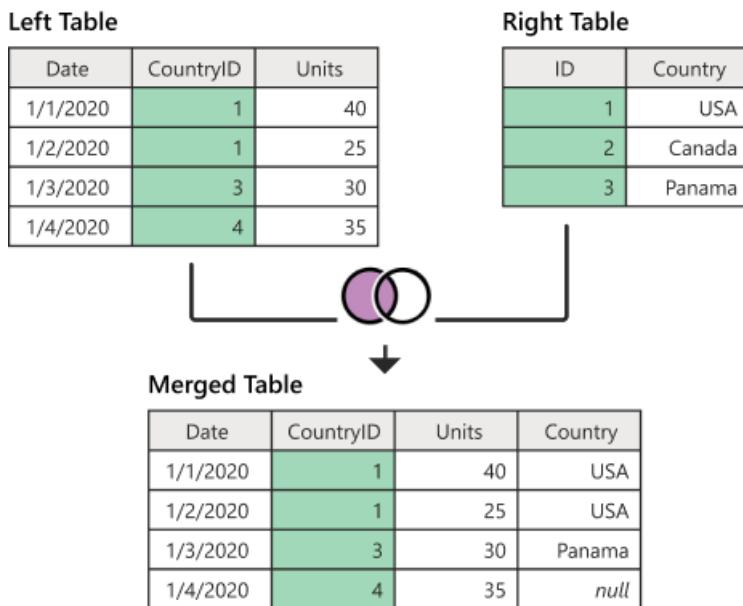


Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 4 in row 4. A table on the right contains ID and Country columns. The emphasized ID column contains values of 1 in row 1 (denoting USA), 2 in row 2 (denoting Canada), and 3 in row 3 (denoting Panama). A table below the first two tables contains Date, CountryID, Units, and Country columns. The table has four rows, with the top two rows containing the data for CountryID 1, one row for CountryID 3, and one row for Country ID 4. Since the right table didn't contain an ID of 4, the value of the fourth row in the Country column contains null.

This article uses sample data to show how to do a merge operation with the left outer join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	4	35

- **Countries:** This table is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	id	Country
1	1	USA
2	2	Canada
3	3	Panama

Countries table with id set to 1 in row 1, 2 in row 2, and 3 in row 3, and Country set to USA in row 1, Canada in row 2, and Panama in row 3.

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where the name of the country appears as a new **Country** column in the **Sales** table as long as the **CountryID** exists in the **Countries** table. If there are no matches between the left and right tables, a *null* value is the result of the merge for that row. In the following image, this is shown to be the case for **CountryID** 4, which was brought in from the **Sales** table.

	Date	CountryID	Units	Country
1	1/1/2020	1	40	USA
2	1/2/2020	1	25	USA
3	1/3/2020	3	30	Panama
4	1/4/2020	4	35	<i>null</i>

To do a left outer join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the **Merge** dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Left outer**.
6. Select **OK**.

Merge

Select a table and matching columns to create a merged table.

Sales

Date	CountryID	Units	
1/1/2020	1	40	
1/2/2020	1	25	
1/3/2020	3	30	
1/4/2020	4	35	



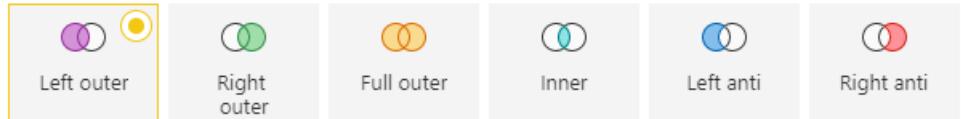
Right table for merge

Countries

id	Country
1	USA
2	Canada
3	Panama



Join kind



Use fuzzy matching to perform the merge

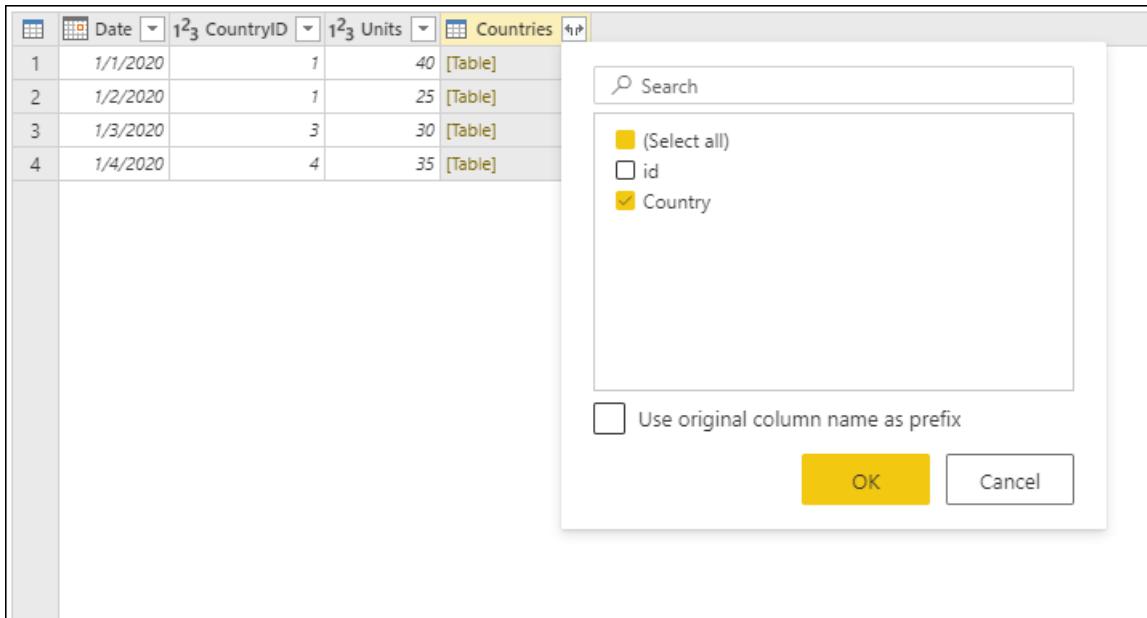
[Fuzzy matching options](#)

The selection matches 3 of 4 rows from the first table

OK

Cancel

From the newly created **Countries** column, expand the **Country** field. Don't select the **Use original column name as prefix** check box.



After performing this operation, you'll create a table that looks like the following image.

	Date	CountryID	Units	Country
1	1/1/2020	1	40	USA
2	1/2/2020	1	25	USA
3	1/3/2020	3	30	Panama
4	1/4/2020	4	35	null

Right outer join

1/15/2022 • 2 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is a *right outer join*, which keeps all the rows from the right table and brings in any matching rows from the left table. More information: [Merge operations overview](#)

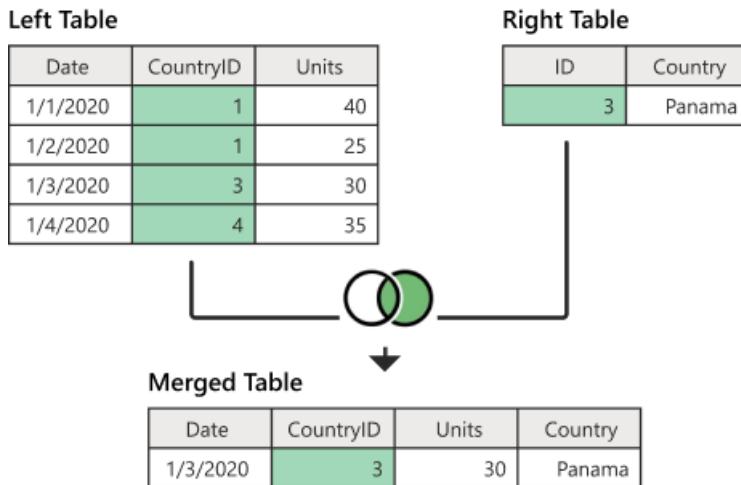


Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 4 in row 4. A table on the right contains ID and Country columns, with only one row. The emphasized ID column contains a value of 3 in row 1 (denoting Panama). A table below the first two tables contains Date, CountryID, Units, and Country columns. The table has one row, with the CountryID of 3 and the Country of Panama.

This article uses sample data to show how to do a merge operation with the right outer join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. The **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	4	35

- **Countries:** This table is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	id	Country
1	3	Panama

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where the name of the country appears as a new **Country** column in the **Sales** table. Because of how the right outer join works, all rows from the right table will be brought in, but only matching rows from the left table will be kept.

	Date	CountryID	Units	Country
1	1/3/2020	3	30	Panama

To do a right outer join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the **Merge** dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Right outer**.
6. Select **OK**.

Merge

Select a table and matching columns to create a merged table.

Sales

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right table for merge

id	Country
3	Panama

Join kind

- Left outer
- Right outer
- Full outer
- Inner
- Left anti
- Right anti

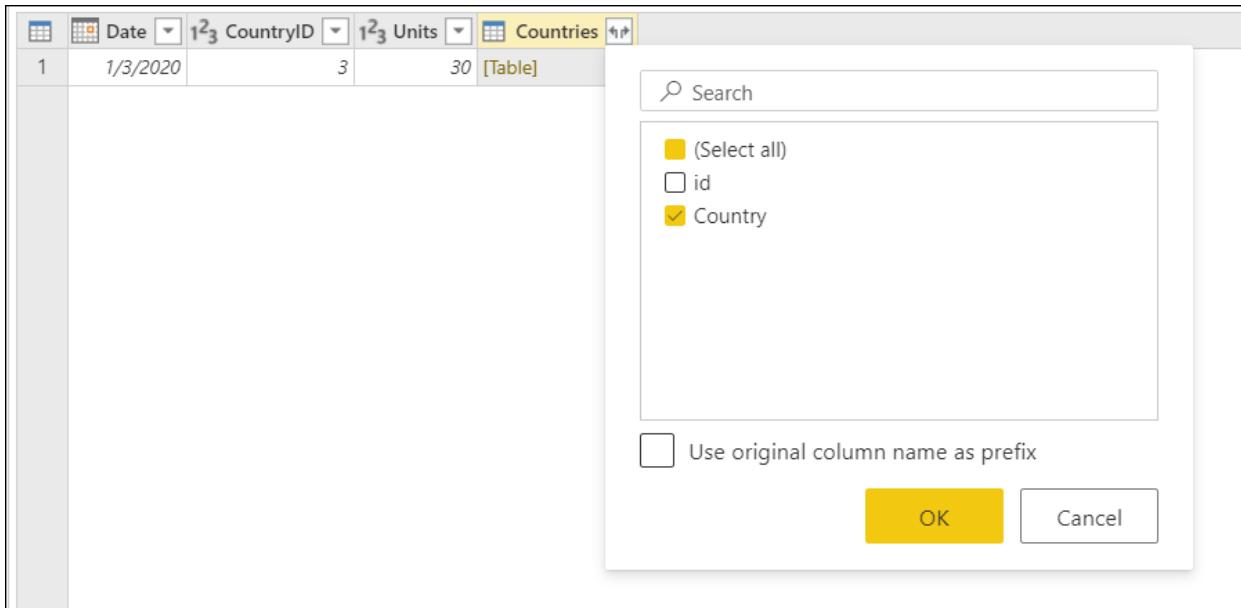
Use fuzzy matching to perform the merge

➤ Fuzzy matching options

✓ The selection matches 1 of 1 rows from the second table

From the newly created **Countries** column, expand the **Country** field. Don't select the **Use original column**

name as prefix check box.



After performing this operation, you'll create a table that looks like the following image.

	Date	CountryID	Units	Country
1	1/3/2020	3	30	Panama

Full outer join

1/15/2022 • 3 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is a *full outer join*, which brings in all the rows from both the left and right tables. More information: [Merge operations overview](#)

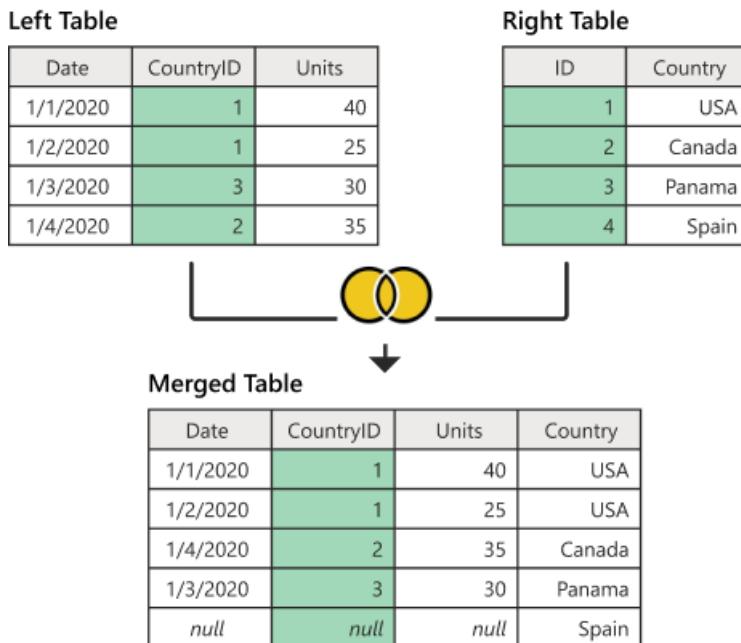


Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 2 in row 4. A table on the right contains ID and Country columns. The emphasized ID column contains values of 1 in row 1 (denoting USA), 2 in row 2 (denoting Canada), 3 in row 3 (denoting Panama), and 4 (denoting Spain) in row 4. A table below the first two tables contains Date, CountryID, Units, and Country columns. All rows have been rearranged in numerical order according to the CountryID value. The country associated with the CountryID number is shown in the Country column. Because the country ID for Spain wasn't contained in the left table, a new row is added, and the date, country ID, and units values for this row are set to null.

This article uses sample data to show how to do a merge operation with the full outer join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	2	35

- **Countries:** This is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	1	2	3	id	A	B	C	Country	D
1				1	USA				
2				2	Canada				
3				3	Panama				
4				4	Spain				

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where the name of the country appears as a new **Country** column in the **Sales** table. Because of how the full outer join works, all rows from both the left and right tables will be brought in, regardless of whether they only appear in one of the tables.

	Date	1	2	3	CountryID	1	2	3	Units	A	B	C	Country	D
1	1/1/2020				1				40	USA				
2	1/2/2020				1				25	USA				
3	1/4/2020				2				35	Canada				
4	1/3/2020				3				30	Panama				
5	null				null				null	Spain				

Full outer join final table with Date, a CountryID, and Units derived from the Sales table, and a Country column derived from the Countries table. A fifth row was added to contain data from Spain, but that row contains null in the Date, CountryID, and Units columns since those values did not exist for Spain in the Sales table.

To perform a full outer join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the Merge dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Full outer**.
6. Select **OK**

Merge

Select a table and matching columns to create a merged table.

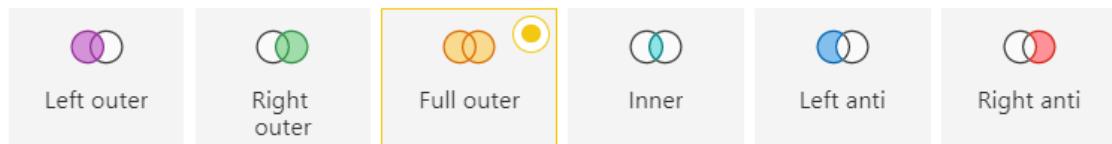
Sales

Date	CountryID	Units	
1/1/2020	1	40	
1/2/2020	1	25	
1/3/2020	3	30	
1/4/2020	2	35	

Right table for merge

Countries
1 USA
2 Canada
3 Panama
4 Spain

Join kind



Use fuzzy matching to perform the merge

➤ Fuzzy matching options

✓ The selection matches 4 of 4 rows from the first table, and 3 of 4 rows from the second table.

OK

Cancel

TIP

Take a closer look at the message at the bottom of the dialog box that reads "The selection matches 4 of 4 rows from the first table, and 3 of 4 rows from the second table." This message is crucial for understanding the result that you get from this operation.

In the **Countries** table, you have the Country Spain with **id** of 4, but there are no records for **CountryID** 4 in the **Sales** table. That's why only three of four rows from the right table found a match. All rows from the right table that didn't have matching rows from the left table will be grouped and shown in a new row in the output table with no values for the fields from the left table.

The screenshot shows a table with four columns: Date, CountryID, Units, and Countries. The Countries column contains a [Table] link. A red arrow points from the Countries link in the fourth row to a smaller table below it, which shows a single row with id 4 and Country Spain.

From the newly created **Countries** column after the merge operation, expand the **Country** field. Don't select the **Use original column name as prefix** check box.

The screenshot shows a table with four columns: Date, CountryID, Units, and Countries. The Countries column contains a [Table] link. A context menu is open over the Countries link, showing options like 'Search', '(Select all)', 'id', 'Country', and a checked 'Use original column name as prefix' checkbox. The 'Country' option is selected.

After performing this operation, you'll create a table that looks like the following image.

The screenshot shows a table with five columns: Date, CountryID, Units, and Country. The fifth row contains null values for Date, CountryID, and Units, and the Country column contains Spain.

Full outer join final table containing Date, a CountryID, and Units derived from the Sales table, and a Country column derived from the Countries table. A fifth row was added to contain data from Spain, but that row contains null in the Date, CountryID, and Units columns since those values didn't exist for Spain in the Sales table.

Inner join

1/15/2022 • 2 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is an *inner join*, which brings in only matching rows from both the left and right tables. More information: [Merge operations overview](#)

Left Table

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right Table

ID	Country
3	Panama
4	Spain

Merged Table

Date	CountryID	Units	Country
1/3/2020	3	30	Panama

Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 2 in row 4. A table on the right contains ID and Country columns. The emphasized ID column contains values of 3 in row 1 (denoting Panama) and 4 in row 2 (denoting Spain). A table below the first two tables contains Date, CountryID, Units, and Country columns, but only one row of data for Panama.

This article uses sample data to show how to do a merge operation with the inner join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	2	35

- **Countries:** This is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	id	Country
1	3	Panama
2	4	Spain

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where the name of the country appears as a new **Country** column in the **Sales** table. Because of how the inner join works, only matching rows from both the left and right tables will be brought in.

	Date	CountryID	Units	Country
1	1/3/2020	3	30	Panama

To perform an inner join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the **Merge** dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Inner**.
6. Select **OK**.

Merge

Select a table and matching columns to create a merged table.

Sales

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right table for merge

Countries	
id	Country
3	Panama
4	Spain

Join kind



Use fuzzy matching to perform the merge

➤ Fuzzy matching options

The selection matches 1 of 4 rows from the first table, and 1 of 2 rows from the second table.

OK

Cancel

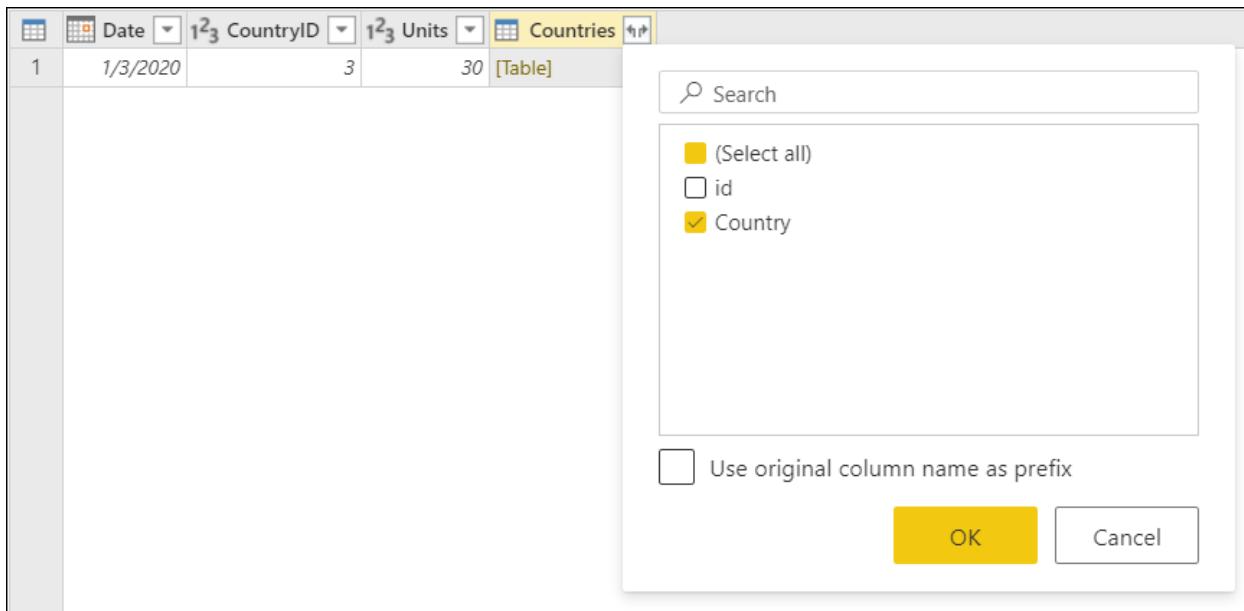
TIP

Take a closer look at the message at the bottom of the dialog box that reads "The selection matches 1 of 4 rows from the first table, and 1 of 2 rows from the second table." This message is crucial to understanding the result that you get from this operation.

In the **Sales** table, you have a **CountryID** of 1 and 2, but neither of these values are found in the **Countries** table. That's why the match only found one of four rows in the left (first) table.

In the **Countries** table, you have the **Country** Spain with the **id** 4, but there are no records for a **CountryID** of 4 in the **Sales** table. That's why only one of two rows from the right (second) table found a match.

From the newly created **Countries** column, expand the **Country** field. Don't select the **Use original column name as prefix** check box.



After performing this operation, you'll create a table that looks like the following image.

1	Date	CountryID	Units	Country
1	1/3/2020	3	30	Panama

Left anti join

1/15/2022 • 3 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is a *left anti join*, which brings in only rows from the left table that don't have any matching rows from the right table. More information: [Merge operations overview](#)

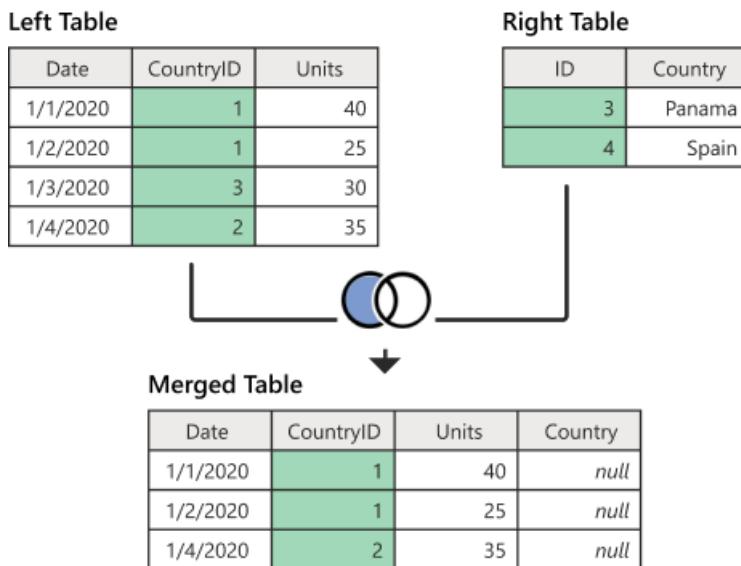


Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 2 in row 4. A table on the right contains ID and Country columns. The emphasized ID column contains values of 3 in row 1 (denoting Panama) and 4 in row 2 (denoting Spain). A table below the first two tables contains Date, CountryID, Units, and Country columns. The table has three rows, with two rows containing the data for CountryID 1, and one row for CountryID 2. Since none of the remaining CountryIDs match any of the countries in the right table, the rows in the Country column in the merged table all contain null.

This article uses sample data to show how to do a merge operation with the left anti join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	2	35

- **Countries:** This table is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	id	Country
1	3	Panama
2	4	Spain

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where only the rows from the left table that don't match any from the right table are kept.

	Date	CountryID	Units	Country
1	1/1/2020	1	40	null
2	1/2/2020	1	25	null
3	1/4/2020	2	35	null

Left anti join final table with Date, CountryID, Units, and Country column headers, and three rows of data of which the values for the Country column are all null.

To do a left anti join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the **Merge** dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Left anti**.
6. Select **OK**.

Merge

Select a table and matching columns to create a merged table.

Sales

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right table for merge

Countries

id	Country
3	Panama
4	Spain

Join kind



Left outer



Right
outer



Full outer



Inner



Left anti



Right anti

Use fuzzy matching to perform the merge

➤ Fuzzy matching options

✅ The selection excludes 1 of 4 rows from the first table

OK

Cancel

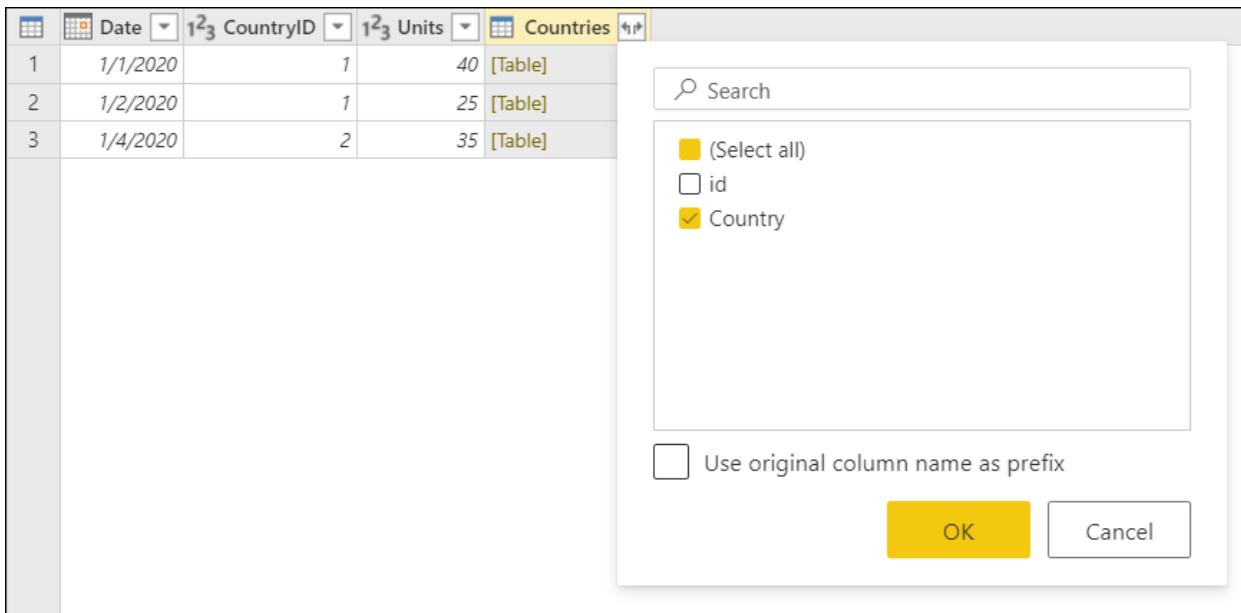
TIP

Take a closer look at the message at the bottom of the dialog box that reads "The selection excludes 1 of 4 rows from the first table." This message is crucial to understanding the result that you get from this operation.

In the **Sales** table, you have a **CountryID** of 1 and 2, but neither of them are found in the **Countries** table. That's why the match only found one of four rows in the left (first) table.

In the **Countries** table, you have the **Country** Spain with an **id** of 4, but there are no records for **CountryID** 4 in the **Sales** table. That's why only one of two rows from the right (second) table found a match.

From the newly created **Countries** column, expand the **Country** field. Don't select the **Use original column name as prefix** check box.



After doing this operation, you'll create a table that looks like the following image. The newly expanded **Country** field doesn't have any values. That's because the left anti join doesn't bring any values from the right table—it only keeps rows from the left table.

Date	CountryID	Units	Country
1/1/2020	1	40	null
1/2/2020	1	25	null
1/4/2020	2	35	null

Final table with Date, CountryID, Units, and Country column headers, and three rows of data of which the values for the Country column are all null.

Right anti join

1/15/2022 • 2 minutes to read • [Edit Online](#)

One of the join kinds available in the **Merge** dialog box in Power Query is a *right anti join*, which brings in only rows from the right table that don't have any matching rows from the left table. More information: [Merge operations overview](#)

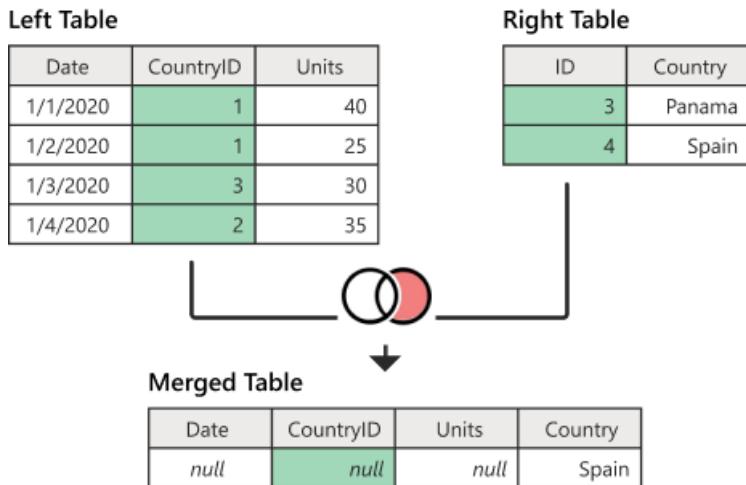


Figure shows a table on the left with Date, CountryID, and Units columns. The emphasized CountryID column contains values of 1 in rows 1 and 2, 3 in row 3, and 2 in row 4. A table on the right contains ID and Country columns. The emphasized ID column contains values of 3 in row 1 (denoting Panama) and 4 in row 2 (denoting Spain). A table below the first two tables contains Date, CountryID, Units, and Country columns. The table has one row, with the Date, CountryID and Units set to null, and the Country set to Spain.

This article uses sample data to show how to do a merge operation with the right anti join. The sample source tables for this example are:

- **Sales:** This table includes the fields **Date**, **CountryID**, and **Units**. **CountryID** is a whole number value that represents the unique identifier from the **Countries** table.

	Date	CountryID	Units
1	1/1/2020	1	40
2	1/2/2020	1	25
3	1/3/2020	3	30
4	1/4/2020	2	35

- **Countries:** This is a reference table with the fields **id** and **Country**. The **id** field represents the unique identifier for each record.

	id	Country
1	3	Panama
2	4	Spain

In this example, you'll merge both tables, with the **Sales** table as the left table and the **Countries** table as the right one. The join will be made between the following columns.

FIELD FROM THE SALES TABLE	FIELD FROM THE COUNTRIES TABLE
CountryID	id

The goal is to create a table like the following, where only the rows from the right table that don't match any from the left table are kept. As a common use case, you can find all the rows that are available in the right table but aren't found in the left table.

Date	CountryID	Units	Country
1 null	null	null	Spain

Right anti join final table with the Date, CountryID, Units, and Country header columns, containing one row with null in all columns except Country, which contains Spain.

To do a right anti join

1. Select the **Sales** query, and then select **Merge queries**.
2. In the **Merge** dialog box, under **Right table for merge**, select **Countries**.
3. In the **Sales** table, select the **CountryID** column.
4. In the **Countries** table, select the **id** column.
5. In the **Join kind** section, select **Right anti**.
6. Select **OK**.

Merge

Select a table and matching columns to create a merged table.

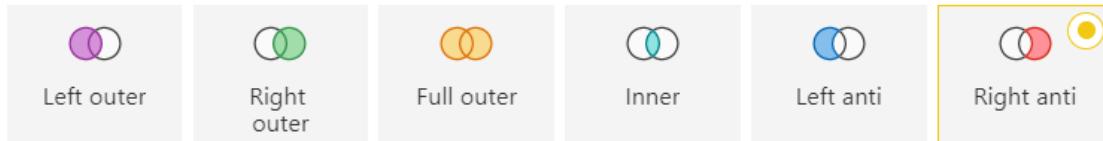
Sales

Date	CountryID	Units	
1/1/2020	1	40	
1/2/2020	1	25	
1/3/2020	3	30	
1/4/2020	2	35	

Right table for merge

Countries	
id	Country
3	Panama
4	Spain

Join kind



Use fuzzy matching to perform the merge

➤ Fuzzy matching options

The selection excludes 1 of 2 rows from the second table

OK

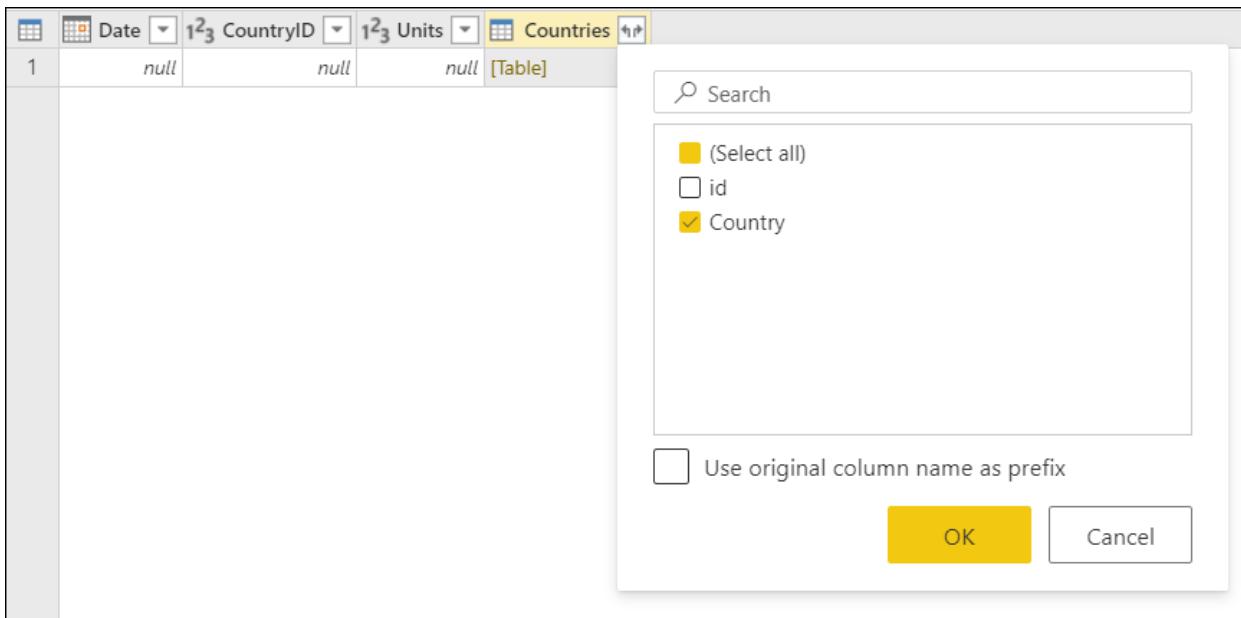
Cancel

TIP

Take a closer look at the message at the bottom of the dialog box that reads "The selection excludes 1 of 2 rows from the second table." This message is crucial to understanding the result that you get from this operation.

In the **Countries** table, you have the **Country** Spain with an **id** of 4, but there are no records for **CountryID** 4 in the **Sales** table. That's why only one of two rows from the right (second) table found a match. Because of how the right anti join works, you'll never see any rows from the left (first) table in the output of this operation.

From the newly created **Countries** column, expand the **Country** field. Don't select the **Use original column name as prefix** check box.



After performing this operation, you'll create a table that looks like the following image. The newly expanded **Country** field doesn't have any values. That's because the right anti join doesn't bring any values from the left table—it only keeps rows from the right table.

	Date	CountryID	Units	Country
1	null	null	null	Spain

Final table with the Date, CountryID, Units, and Country header columns, containing one row with null in all columns except Country, which contains Spain.

Fuzzy merge

1/15/2022 • 4 minutes to read • [Edit Online](#)

Fuzzy merge is a smart data preparation feature you can use to apply fuzzy matching algorithms when comparing columns, to try to find matches across the tables that are being merged.

You can enable fuzzy matching at the bottom of the **Merge** dialog box by selecting the **Use fuzzy matching to perform the merge** option button. More information: [Merge operations overview](#)

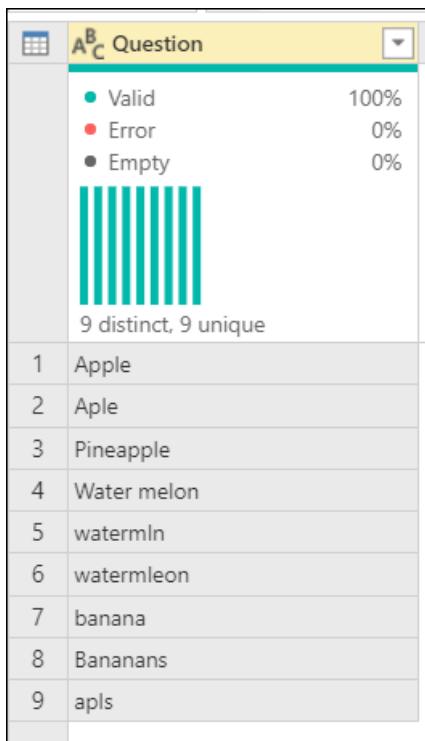
NOTE

Fuzzy matching is only supported on merge operations over text columns. Power Query uses the Jaccard similarity algorithm to measure the similarity between pairs of instances.

Sample scenario

A common use case for fuzzy matching is with freeform text fields, such as in a survey. For this article, the sample table was taken directly from an online survey sent to a group with only one question: *What is your favorite fruit?*

The results of that survey are shown in the following image.



Sample survey output table containing the column distribution graph showing nine distinct answers with all answers unique, and the answers to the survey with all the typos, plural or singular, and case problems.

The nine records reflect the survey submissions. The problem with the survey submissions is that some have typos, some are plural, some are singular, some are uppercase, and some are lowercase.

To help standardize these values, in this example you have a **Fruits** reference table.

A ^B Fruit	
● Valid	100%
● Error	0%
● Empty	0%
	
4 distinct, 4 unique	
1	Apple
2	Pineapple
3	Watermelon
4	Banana

Fruits reference table containing column distribution graph showing four distinct fruits with all fruits unique, and the list of fruits: apple, pineapple, watermelon, and banana.

NOTE

For simplicity, this **Fruits** reference table only includes the name of the fruits that will be needed for this scenario. Your reference table can have as many rows as you need.

The goal is to create a table like the following, where you've standardized all these values so you can do more analysis.

A ^B Question		A ^B Fruit
● Valid	100%	● Valid
● Error	0%	● Error
● Empty	0%	● Empty
	9 distinct, 9 unique	
1	Apple	Apple
2	Aple	Apple
3	Pineapple	Pineapple
4	Water melon	Watermelon
5	watermln	Watermelon
6	watermleon	Watermelon
7	banana	Banana
8	Bananans	Banana
9	apls	Apple

Sample survey output table with the Question column containing the column distribution graph showing nine distinct answers with all answers unique, and the answers to the survey with all the typos, plural or singular, and case problems, and also contains the Fruit column containing the column distribution graph showing four distinct answers with one unique answer and lists all of the fruits properly spelled, singular, and proper case.

Fuzzy merge

To do the fuzzy merge, you start by doing a merge. In this case, you'll use a left outer join, where the left table is the one from the survey and the right table is the **Fruits** reference table. At the bottom of the dialog box, select

the Use fuzzy matching to perform the merge check box.

Merge

Select a table and matching columns to create a merged table.

Survey

A ^B _C Question
Apple
Aple
Pineapple
Water melon
watermln

Right table for merge

A ^B _C Fruit
Apple
Pineapple
Watermelon
Banana

Join kind

Left outer Right outer Full outer Inner Left anti Right anti

Use fuzzy matching to perform the merge

> Fuzzy matching options

The selection matches 8 of 9 rows from the first table

OK Cancel

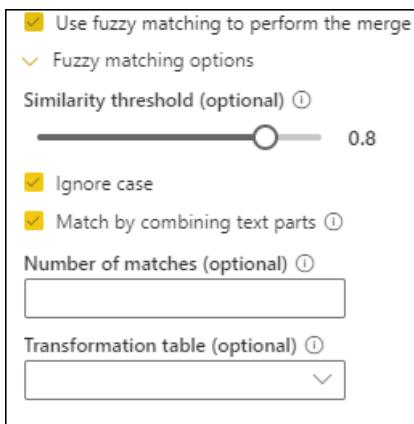
After you select OK, you can see a new column in your table because of this merge operation. If you expand it, you'll notice that there's one row that doesn't have any values in it. That's exactly what the dialog box message in the previous image stated when it said "The selection matches 8 of 9 rows from the first table."

A ^B Question	A ^B Fruit
Valid	100%
Error	0%
Empty	0%
9 distinct, 9 unique	5 distinct, 2 uni...
1 Apple	Apple
2 Aple	Apple
3 Pineapple	Pineapple
4 Water melon	Watermelon
5 watermin	Watermelon
6 watermleon	Watermelon
7 banana	Banana
8 Bananans	Banana
9 apls	null

Fruit column added to the Survey table, with all rows in the Question column expanded, except for row 9, which could not expand and the Fruit column contains null.

Fuzzy matching options

You can modify the **Fuzzy matching options** to tweak how the approximate match should be done. First, select the **Merge queries** command, and then in the **Merge** dialog box, expand **Fuzzy matching options**.



The available options are:

- **Similarity threshold (optional)**: A value between 0.00 and 1.00 that provides the ability to match records above a given similarity score. A threshold of 1.00 is the same as specifying an exact match criteria. For example, **Grapes** matches with **Graes** (missing the letter *p*) only if the threshold is set to less than 0.90. By default, this value is set to 0.80.
- **Ignore case**: Allows matching records no matter what the case of the text.
- **Match by combining text parts**: Allows combining text parts to find matches. For example, **Micro soft** is matched with **Microsoft** if this option is enabled.
- **Number of matches (optional)**: Specifies the maximum number of matching rows that can be returned for every input row.
- **Transformation table (optional)**: Allows matching records based on custom value mappings. For example, **Grapes** is matched with **Raisins** if a transformation table is provided where the **From** column contains **Grapes** and the **To** column contains **Raisins**.

Transformation table

For the example in this article, you can use a transformation table to map the value that has a missing pair. That

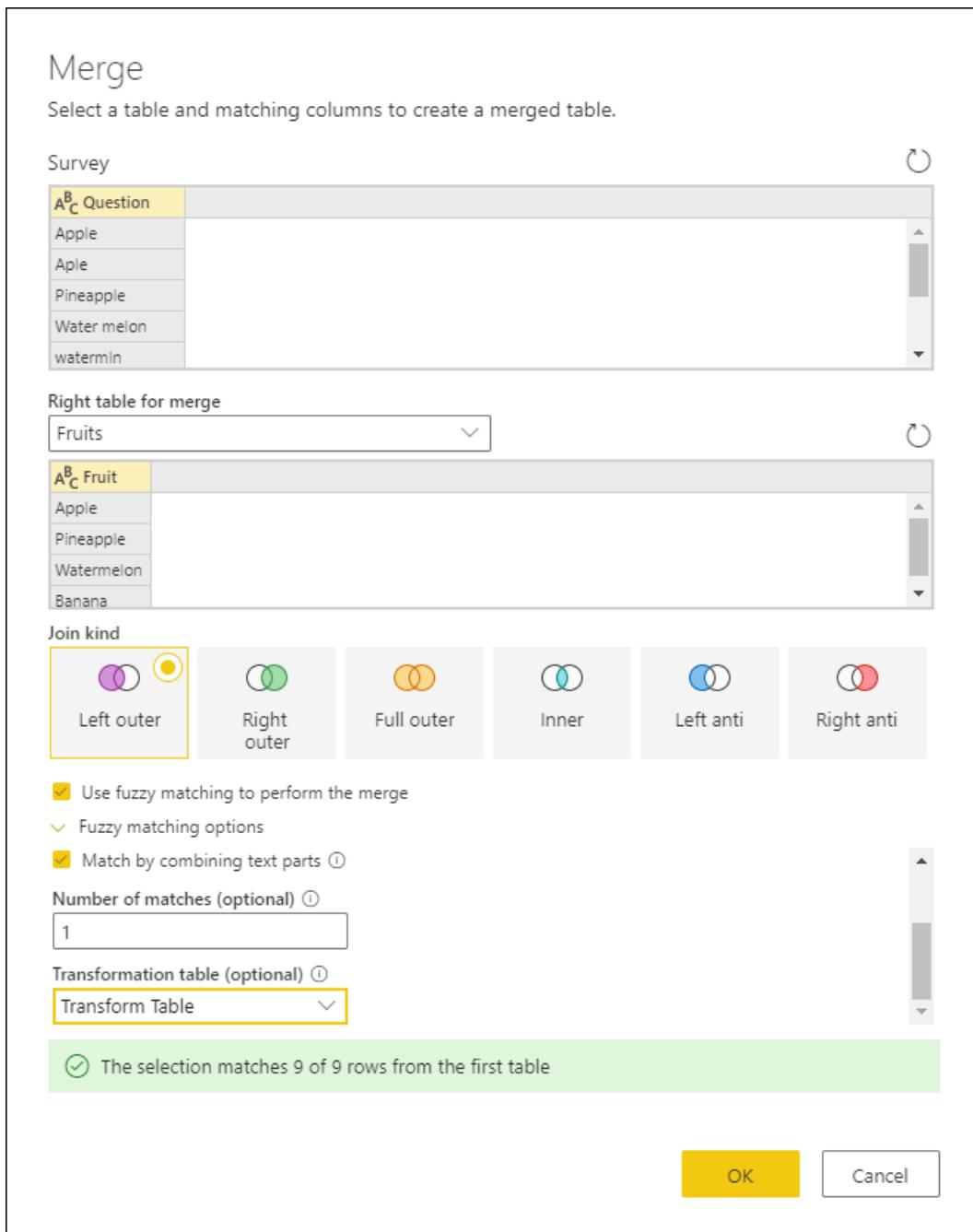
value is **apls**, which needs to be mapped to **Apple**. Your transformation table has two columns:

- **From** contains the values to find.
- **To** contains the values that will be used to replace the values found by using the **From** column.

For this article, the transformation table will look as follows:

FROM	TO
apls	Apple

You can go back to the **Merge** dialog box, and in **Fuzzy matching options** under **Number of matches (optional)**, enter **1**. Under **Transformation table (optional)**, select **Transform Table** from the drop-down menu.



After you select **OK**, you'll create a table that looks like the following image, with all values mapped correctly. Note how the example started with nine distinct values, but after the fuzzy merge, there are only four distinct values.

A ^B _C Question		A ^B _C Fruit	
● Valid	100%	● Valid	100%
● Error	0%	● Error	0%
● Empty	0%	● Empty	0%
	9 distinct, 9 unique		4 distinct, 1 unique
1 Apple		Apple	
2 Aple		Apple	
3 Pineapple		Pineapple	
4 Water melon		Watermelon	
5 watermln		Watermelon	
6 watermleon		Watermelon	
7 banana		Banana	
8 Bananans		Banana	
9 apls		Apple	

Fuzzy merge survey output table with the Question column containing the column distribution graph showing nine distinct answers with all answers unique, and the answers to the survey with all the typos, plural or singular, and case problems. Also contains the Fruit column with the column distribution graph showing four distinct answers with one unique answer and lists all of the fruits properly spelled, singular, and proper case.

Cross join

1/15/2022 • 2 minutes to read • [Edit Online](#)

A *cross join* is a type of join that returns the Cartesian product of rows from the tables in the join. In other words, it combines each row from the first table with each row from the second table.

This article demonstrates, with a practical example, how to do a cross join in Power Query.

Sample input and output tables

For this example, the sample source tables are:

- **Product:** A table with all the generic products that you sell.

	A	B	C	Product	▼
1				Shirt	
2				Jeans	
3				Leggings	

- **Colors:** A table with all the product variations, as colors, that you can have in your inventory.

	A	B	C	Colors	▼
1				Red	
2				Blue	
3				Black	
4				White	

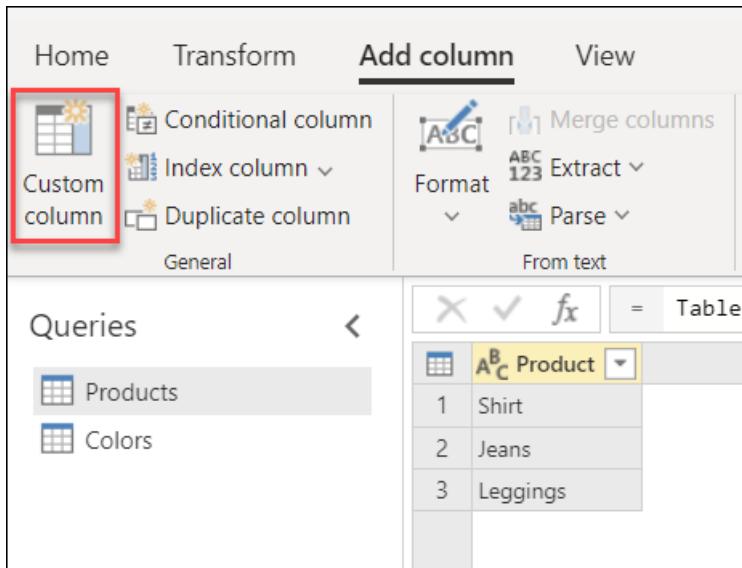
The goal is to perform a cross-join operation with these two tables to create a list of all unique products that you can have in your inventory, as shown in the following table. This operation is necessary because the **Product** table only contains the generic product name, and doesn't give the level of detail you need to see what product variations (such as color) there are.

	A	B	C	Product	▼	A	B	C	123	Colors	▼
1				Shirt						Red	
2				Shirt						Blue	
3				Shirt						Black	
4				Shirt						White	
5				Jeans						Red	
6				Jeans						Blue	
7				Jeans						Black	
8				Jeans						White	
9				Leggings						Red	
10				Leggings						Blue	
11				Leggings						Black	
12				Leggings						White	

Perform a cross join

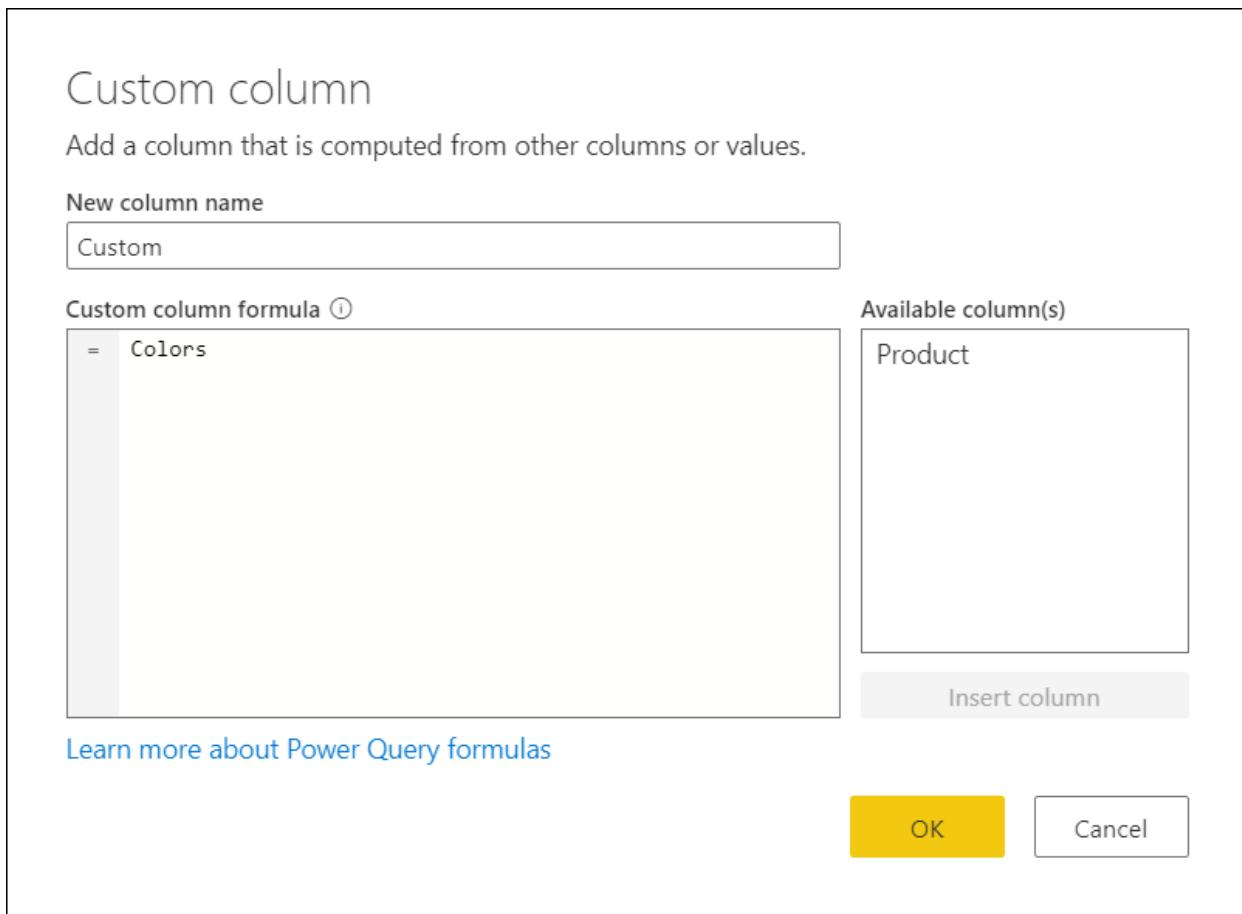
To do a cross-join operation in Power Query, first go to the **Product** table. From the **Add column** tab on the

ribbon, select **Custom column**. More information: [Add a custom column](#)



The screenshot shows the Power Query ribbon with the 'Add column' tab selected. Under the 'General' section, the 'Custom column' option is highlighted with a red box. Other options include 'Conditional column', 'Index column', 'Duplicate column', 'Format', 'Extract', and 'Parse'. To the right, there are buttons for 'Merge columns', 'From text', and a table preview showing three rows: '1 Shirt', '2 Jeans', and '3 Leggings'.

In the **Custom column** dialog box, enter whatever name you like in the **New column name** box, and enter `Colors` in the **Custom column formula** box.



The dialog box has the title 'Custom column'. It says 'Add a column that is computed from other columns or values.' Below that, the 'New column name' is set to 'Custom'. In the 'Custom column formula' section, the formula is set to `= Colors`. To the right, under 'Available column(s)', the 'Product' column is listed. At the bottom are 'OK' and 'Cancel' buttons.

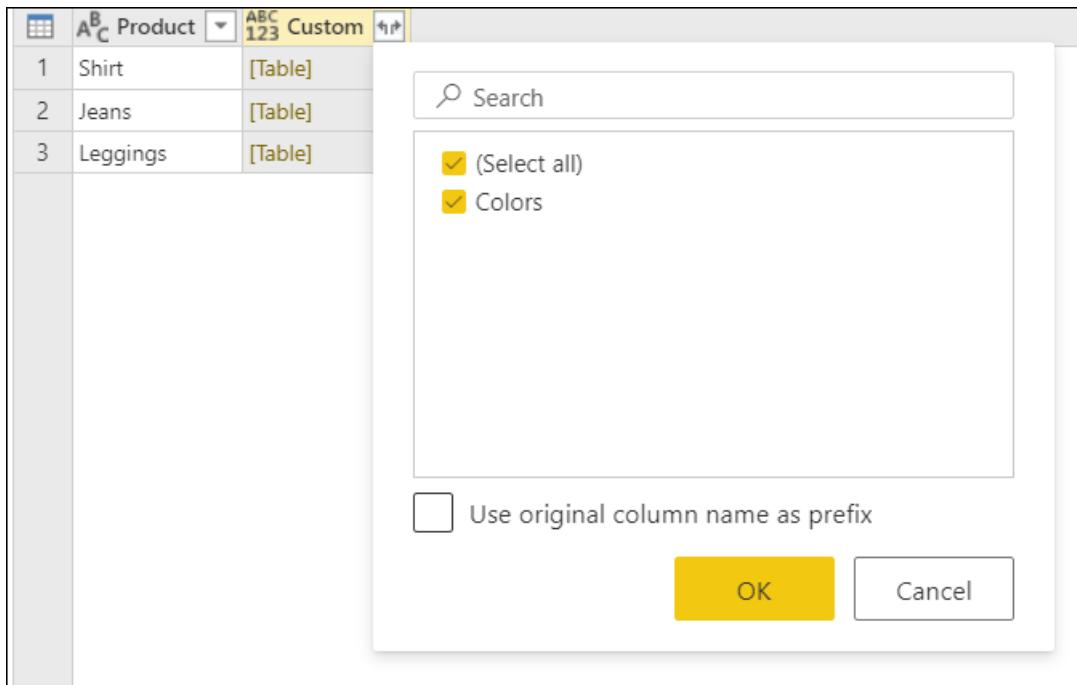
IMPORTANT

If your query name has spaces in it, such as **Product Colors**, the text that you need to enter in the **Custom column formula** section has to follow the syntax `#"Query name"`. For **Product Colors**, you need to enter `#"Product Colors"`

You can check the name of your queries in the **Query settings** pane on the right side of your screen or in the **Queries** pane on the left side.

After you select **OK** in the **Custom column** dialog box, a new column is added to the table. In the new column

heading, select **Expand**  to expand the contents of this newly created column, and then select **OK**.



After you select **OK**, you'll reach your goal of creating a table with all possible combinations of **Product** and **Colors**.

	A B C Product	ABC 123 Colors
1	Shirt	Red
2	Shirt	Blue
3	Shirt	Black
4	Shirt	White
5	Jeans	Red
6	Jeans	Blue
7	Jeans	Black
8	Jeans	White
9	Leggings	Red
10	Leggings	Blue
11	Leggings	Black
12	Leggings	White

Split columns by delimiter

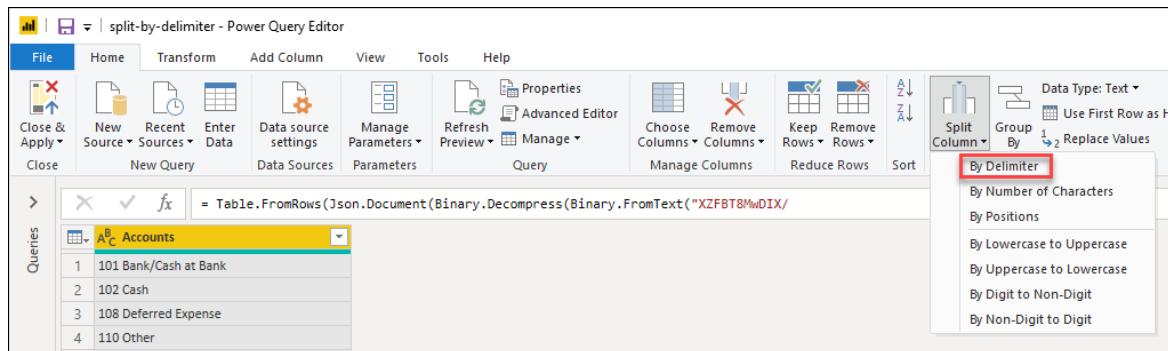
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by a delimiter.

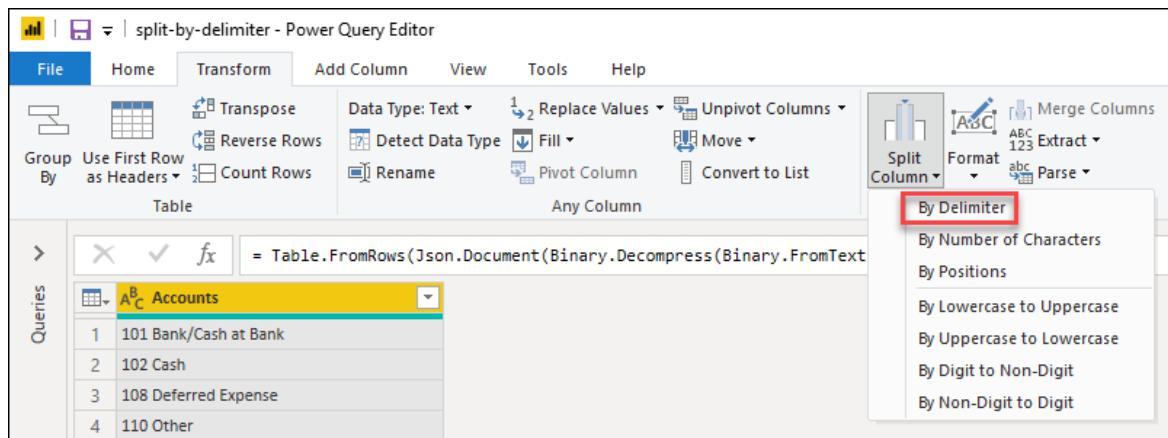
Where to find Split Columns > By Delimiter

You can find the **Split Columns: By Delimiter** option in three places:

- **Home tab**—under the **Split column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split column** dropdown menu inside the **Text column** group.



- **Right-click a column**—inside the **Split column** option.

A screenshot of the Microsoft Power BI Data Editor interface. A context menu is open over a column titled 'Accounts'. The 'Split Column' option is highlighted, and a submenu is displayed with several options: 'By Delimiter...', 'By Number of Characters...', 'By Positions...', 'By Lowercase to Uppercase', 'By Uppercase to Lowercase', 'By Digit to Non-Digit', and 'By Non-Digit to Digit'. The 'By Delimiter...' option is specifically highlighted with a red box.

Split columns by delimiter into columns

In this example, the initial table will be the one shown in the image below, with only one column for **Accounts**.

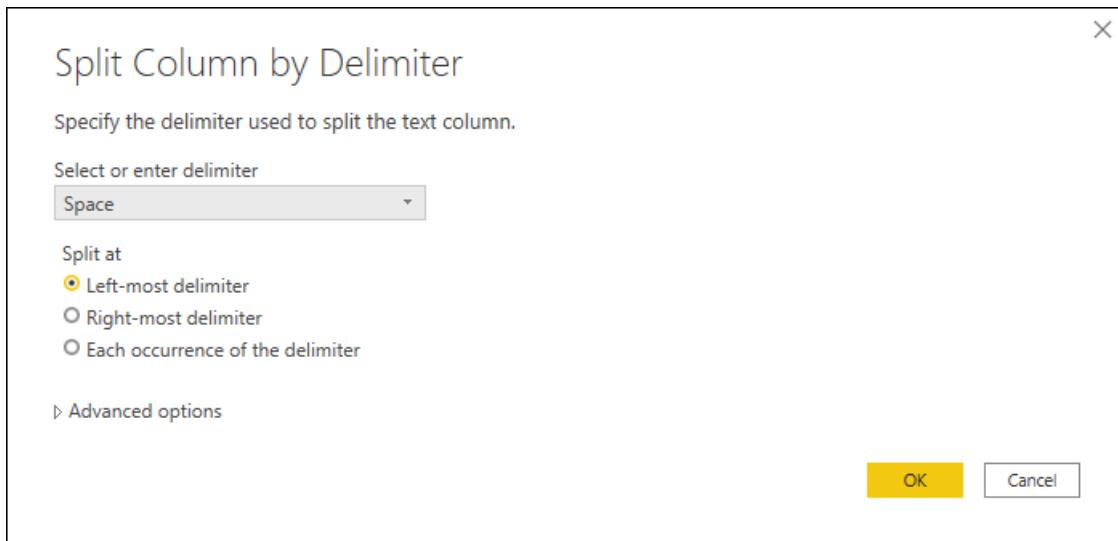
	Accounts
1	101 Bank/Cash at Bank
2	102 Cash
3	108 Deferred Expense
4	110 Other
5	112 Accounts Receivable
6	116 Supplies
7	130 Prepaid Insurance
8	157 Equipment
9	158 Accumulated Depreciation Equipment
10	200 Notes Payable
11	201 Accounts Payable
12	209 Unearned Service Revenue
13	230 Interest Payable
14	231 Deferred Gross profit
15	300 Owner's capital
16	311 Share Capital-Ordinary
17	320 Retained Earnings
18	330 Capital contributions
19	332 Dividends
20	350 Income Summary
21	360 Drawings (Distributions)
22	300 Dividend
23	310 Capital in excess of par

This column holds two values:

- Account number
- Account name

In this example, you want to split this column into two columns. The values are delimited by a space—the first space from left to right. To do this split, select the column, and then select the option to split the column by a delimiter. In **Split Column by Delimiter**, apply the following configuration:

- **Select or enter delimiter:** Space
- **Split at:** Left-most delimiter



The result of that operation will give you a table with the two columns that you're expecting.

	A ^B Accounts.1	A ^B Accounts.2
1	101	Bank/Cash at Bank
2	102	Cash
3	108	Deferred Expense
4	110	Other
5	112	Accounts Receivable
6	116	Supplies
7	130	Prepaid Insurance
8	157	Equipment
9	158	Accumulated Depreciation Equipment
10	200	Notes Payable
11	201	Accounts Payable
12	209	Unearned Service Revenue
13	230	Interest Payable
14	231	Deferred Gross profit
15	300	Owner's capital
16	311	Share Capital-Ordinary
17	320	Retained Earnings
18	330	Capital contributions
19	332	Dividends
20	350	Income Summary
21	360	Drawings (Distributions)
22	300	Dividend
23	310	Capital in excess of par

NOTE

Power Query will split the column into as many columns as needed. The name of the new columns will contain the same name as the original column. A suffix that includes a dot and a number that represents the split sections of the original column will be appended to the name of the new columns.

Split columns by delimiter into rows

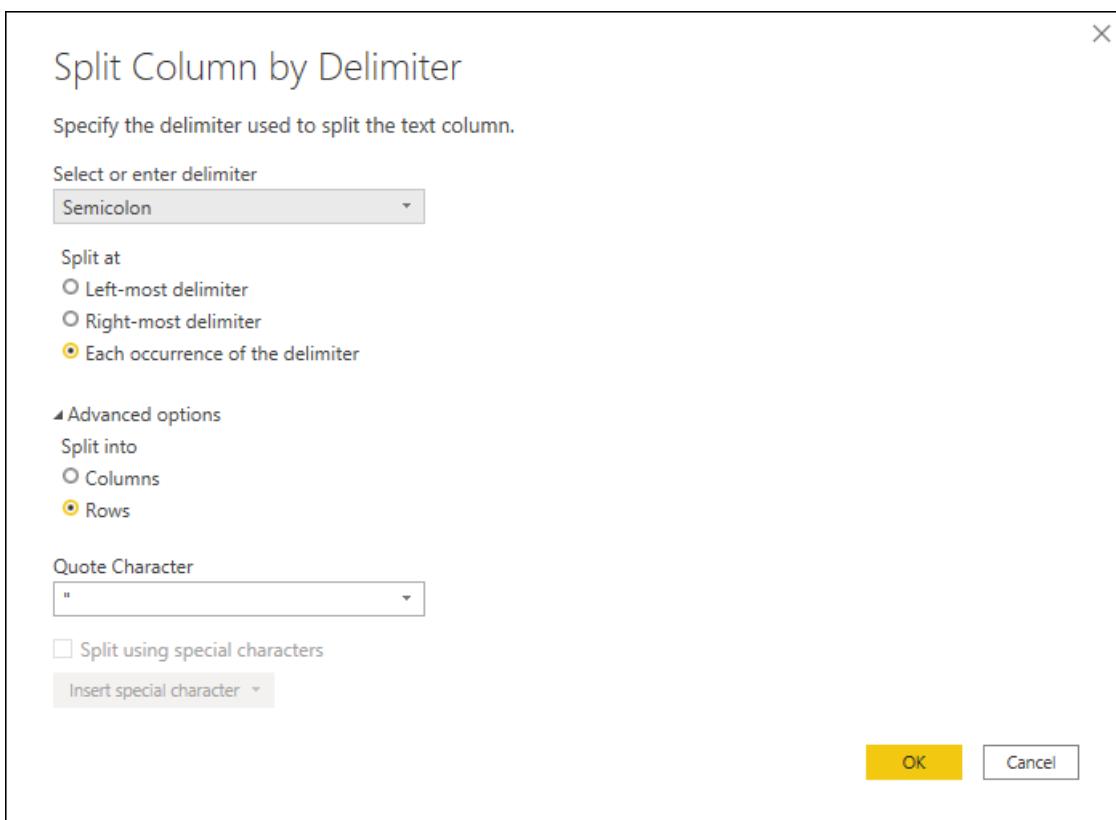
In this example, your initial table will be the one shown in the image below, with the columns **Cost Center** and **Accounts**.

A ^B Cost Center	A ^B Accounts
1 Panama	ACME,1090;Globex,2040;Lexcorp,1204;Wayne Enterprises,1090-4
2 USA	Wayne Enterprises,1090;ACME,2040;Lexcorp,1204
3 Canada	Lexcorp,1090;Wayne Enterprises,2040;ACME,1204

The **Accounts** column has values in pairs separated by a comma. These pairs are separated by a semicolon. The goal of this example is to split this column into new rows by using the semicolon as the delimiter.

To do that split, select the **Accounts** column. Select the option to split the column by a delimiter. In **Split Column by Delimiter**, apply the following configuration:

- **Select or enter delimiter:** Semicolon
- **Split at:** Each occurrence of the delimiter
- **Split into:** Rows



The result of that operation will give you a table with the same number of columns, but many more rows because the values inside the cells are now in their own cells.

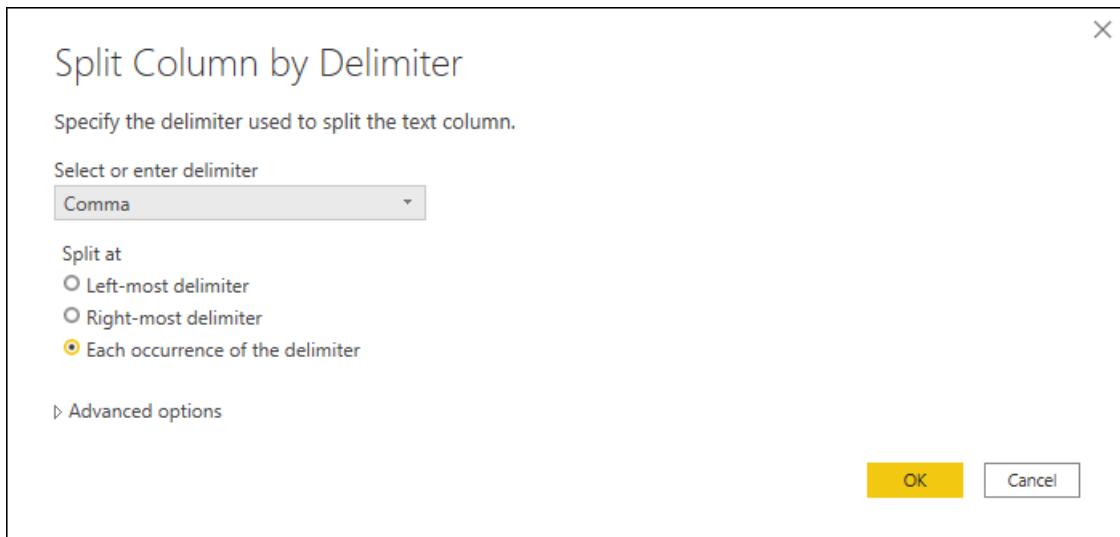
A ^B Cost Center	A ^B Accounts
1 Panama	ACME,1090
2 Panama	Globex,2040
3 Panama	Lexcorp,1204
4 Panama	Wayne Enterprises,1090-4
5 USA	Wayne Enterprises,1090
6 USA	ACME,2040
7 USA	Lexcorp,1204
8 Canada	Lexcorp,1090
9 Canada	Wayne Enterprises,2040
10 Canada	ACME,1204

Final Split

Your table still requires one last split column operation. You need to split the **Accounts** column by the first comma that it finds. This split will create a column for the account name and another one for the account number.

To do that split, select the **Accounts** column and then select **Split Column > By Delimiter**. Inside the **Split column** window, apply the following configuration:

- **Select or enter delimiter:** Comma
- **Split at:** Each occurrence of the delimiter



The result of that operation will give you a table with the three columns that you're expecting. You then rename the columns as follows:

PREVIOUS NAME	NEW NAME
Accounts.1	Account Name
Accounts.2	Account Number

Your final table looks like the one in the following image.

	Cost Center	Account Name	Account Number
1	Panama	ACME	1090
2	Panama	Globex	2040
3	Panama	Lexcorp	1204
4	Panama	Wayne Enterprises	1090-4
5	USA	Wayne Enterprises	1090
6	USA	ACME	2040
7	USA	Lexcorp	1204
8	Canada	Lexcorp	1090
9	Canada	Wayne Enterprises	2040
10	Canada	ACME	1204

Split columns by number of characters

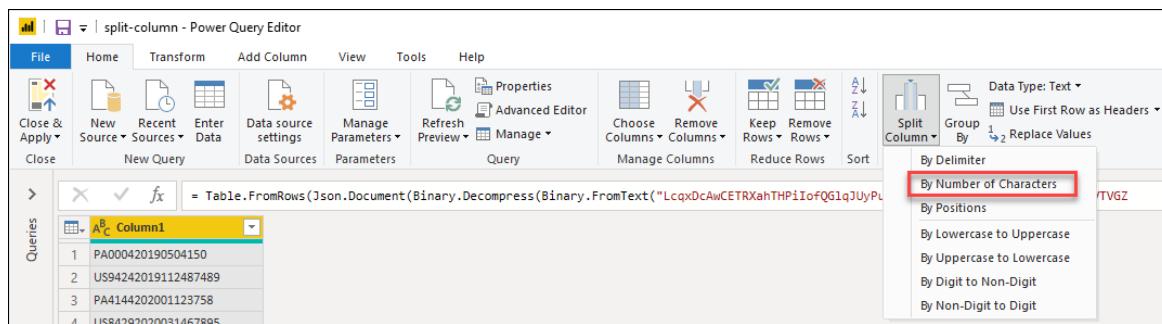
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by the number of characters.

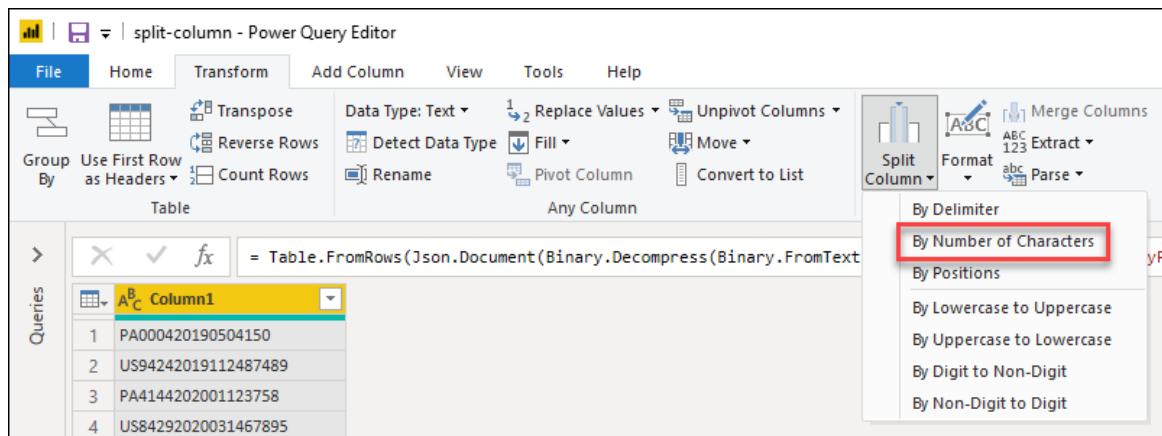
Where to find Split Columns > By Number of Characters

You can find the Split Columns > By Number of Characters option in three places:

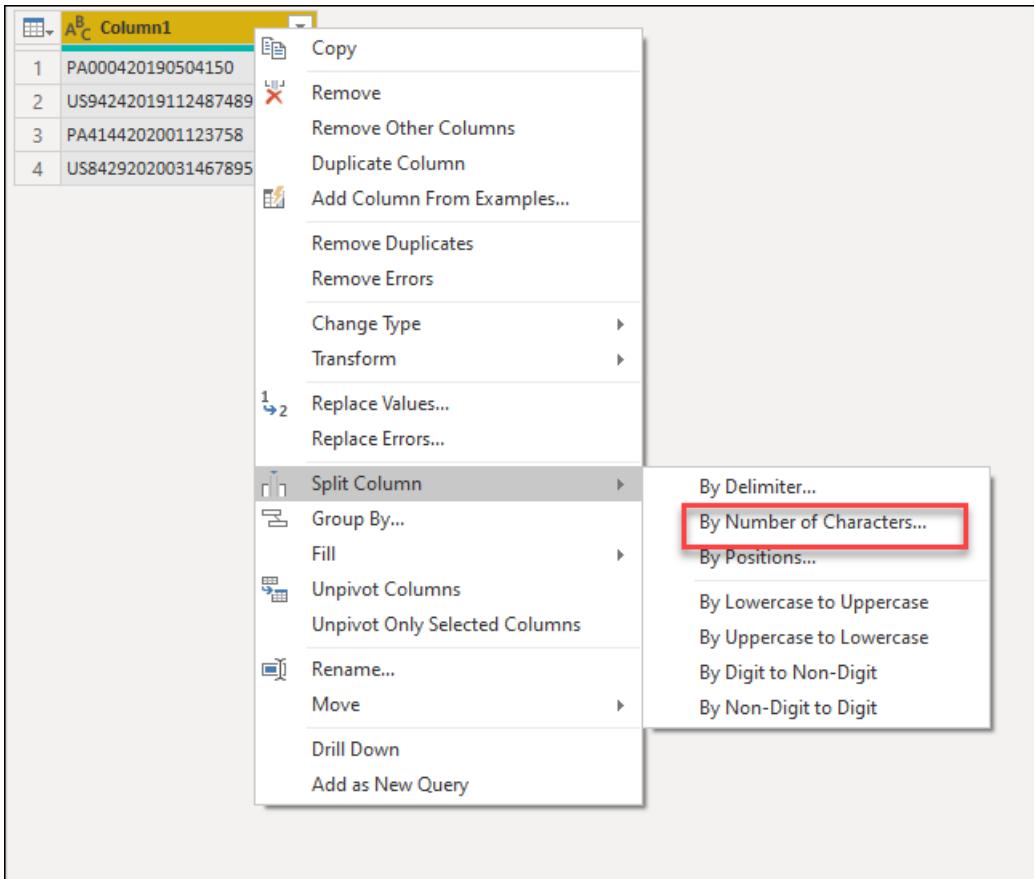
- Home tab—under the Split Column dropdown menu inside the Transform group.



- Transform tab—under the Split Column dropdown menu inside the Text Column group.



- Right-click a column—inside the Split Column option.



Split columns by number of characters into columns

The initial table for this example will be the one below, with only one column for **Column1**.

	Column1
1	PA000420190504150
2	US94242019112487489
3	PA4144202001123758
4	US84292020031467895

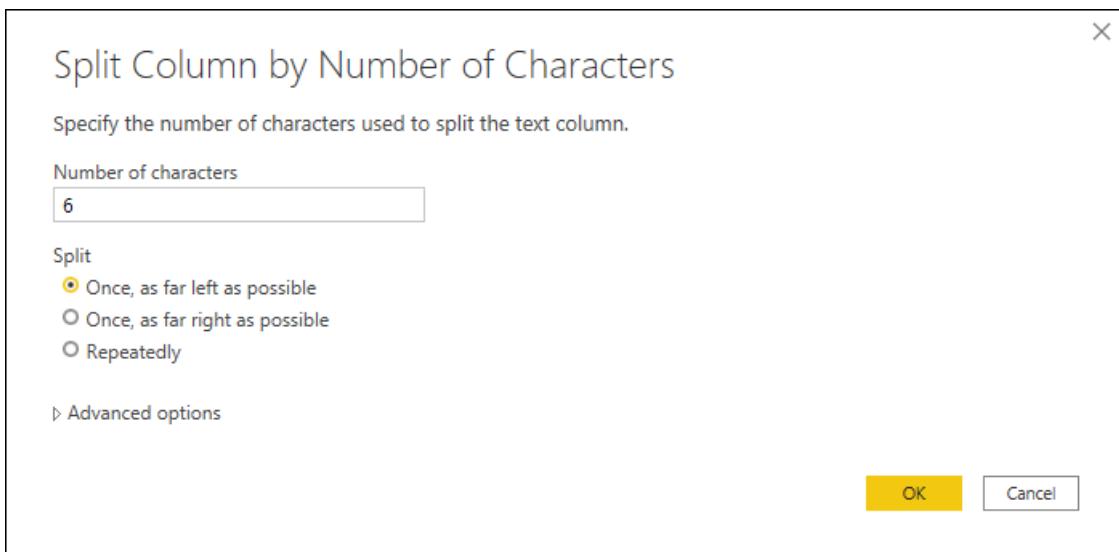
This column holds three values:

- **Account Name**—in the first six characters
- **Date**—in the following eight characters with the format `yyyymmdd`
- **Units**—the remaining characters

In this example, you want to split this column into three columns containing the values described in the list above.

To do this split, select the column and then select the option to split the column by the number of characters. In **Split column by Number of Characters**, apply the following configuration:

- **Number of characters:** 6
- **Split:** Once, as far left as possible



The result of that operation will give you a table with two columns. One for the account name and the other one that contains the combined values for the date and units.

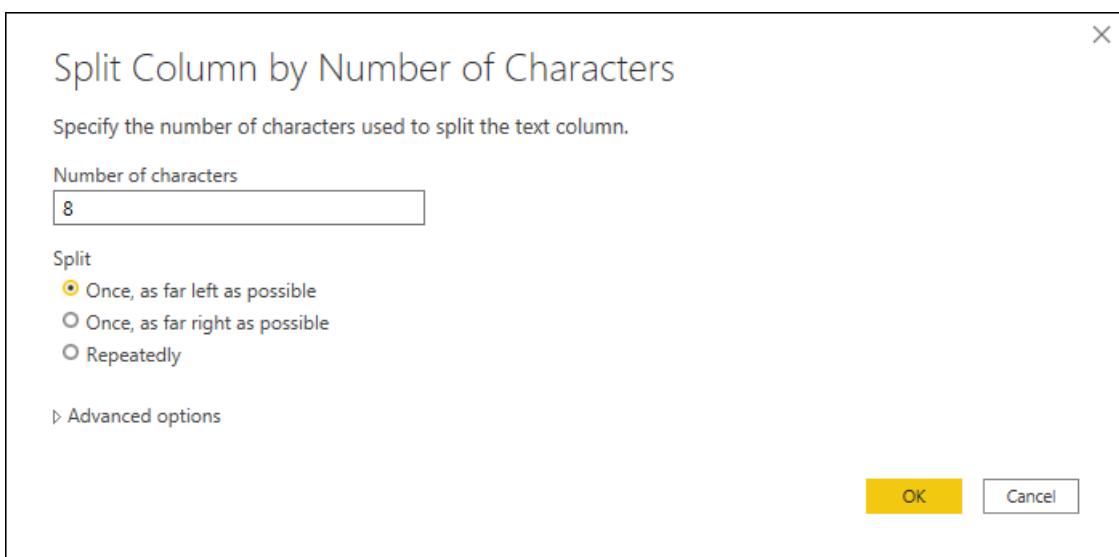
	A ^B _C Column1.1	A ^B _C Column1.2
1	PA0004	20190504150
2	US9424	2019112487489
3	PA4144	202001123758
4	US8429	2020031467895

NOTE

Power Query will split the column into only two columns. The name of the new columns will contain the same name as the original column. A suffix containing a dot and a number that represents the split section of the column will be appended to the names of the new columns.

Now continue to do the same operation over the new Column1.2 column, but with the following configuration:

- **Number of characters:** 8
- **Split:** Once, as far left as possible



The result of that operation will yield a table with three columns. Notice the new names of the two columns on the far right. **Column1.2.1** and **Column1.2.2** were automatically created by the split column operation.

A ^B C Column1.1	A ^B C Column1.2.1	A ^B C Column1.2.2
1 PA0004	20190504	150
2 US9424	20191124	87489
3 PA4144	20200112	3758
4 US8429	20200314	67895

You can now change the name of the columns and also define the data types of each column as follows:

ORIGINAL COLUMN NAME	NEW COLUMN NAME	DATA TYPE
Column1.1	Account Name	Text
Column1.2.1	Date	Date
Column1.2.2	Units	Whole Number

Your final table will look like the one in the following image.

A ^B C Account Name	Date	1 ² 3 Units
1 PA0004	5/4/2019	150
2 US9424	11/24/2019	87489
3 PA4144	1/12/2020	3758
4 US8429	3/14/2020	67895

Split columns by number of characters into rows

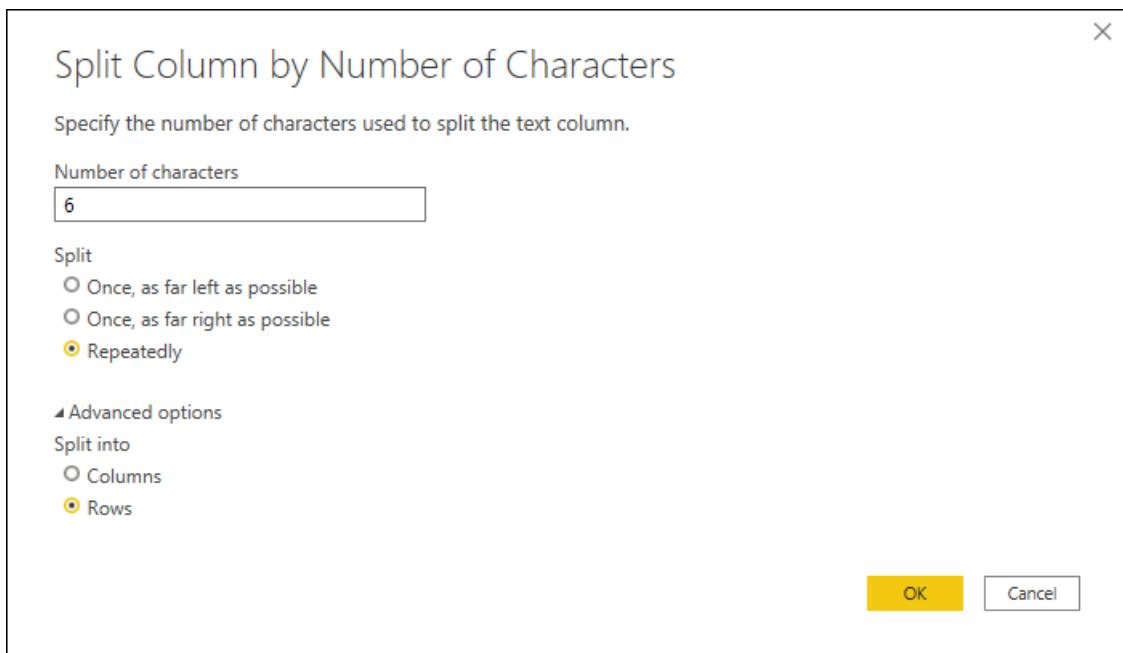
The initial table for this example will be the one below, with the columns **Group** and **Account**.

A ^B C Group	A ^B C Account
1 Group A	PA0004US9424
2 Group B	PA4144US8429

The **Account** column can hold multiple values in the same cell. Each value has the same length in characters, with a total of six characters. In this example, you want to split these values so you can have each account value in its own row.

To do that, select the **Account** column and then select the option to split the column by the number of characters. In **Split column by Number of Characters**, apply the following configuration:

- **Number of characters:** 6
- **Split:** Repeatedly
- **Split into:** Rows



The result of that operation will give you a table with the same number of columns, but many more rows because the fragments inside the original cell values in the **Account** column are now split into multiple rows.

	A ^B _C Group	A ^B _C Account
1	Group A	PA0004
2	Group A	US9424
3	Group B	PA4144
4	Group B	US8429

Split columns by positions

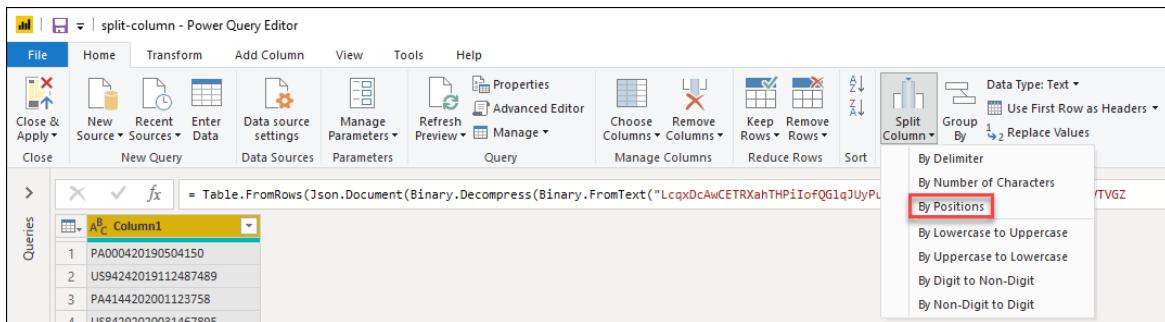
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by positions.

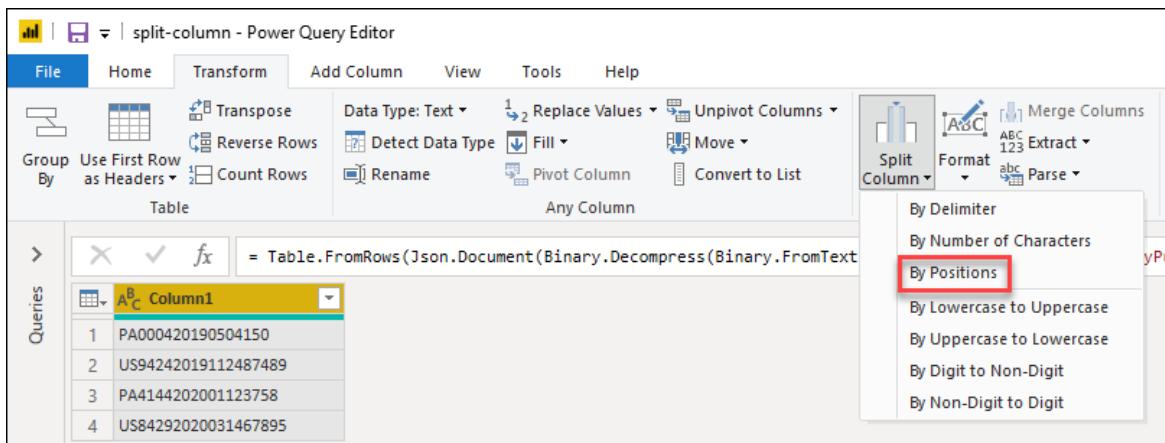
Where to find Split Columns > By Positions

You can find the **Split Columns > By Positions** option in three places:

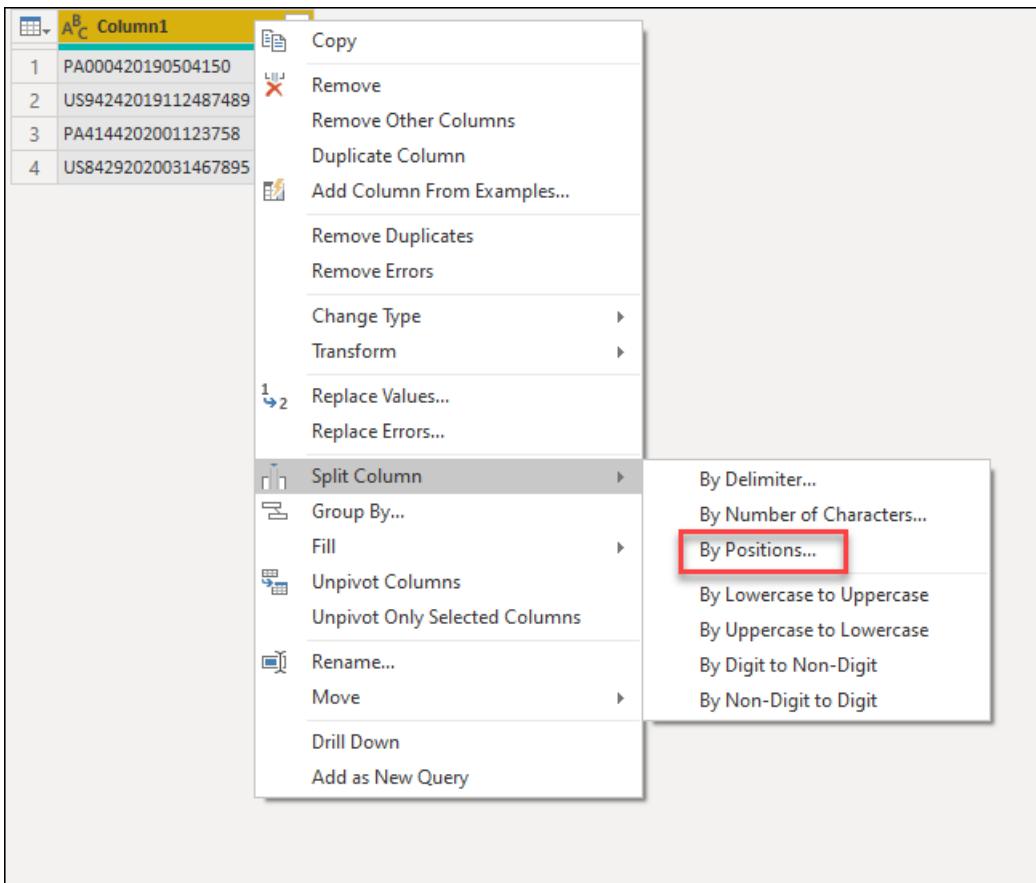
- **Home tab**—under the **Split Column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split Column** dropdown menu inside the **Text Column** group.



- **Right-click a column**—inside the **Split Column** option.



Split columns by positions into columns

The initial table for this example will be the one shown in the image below, with only one column for **Column1**.

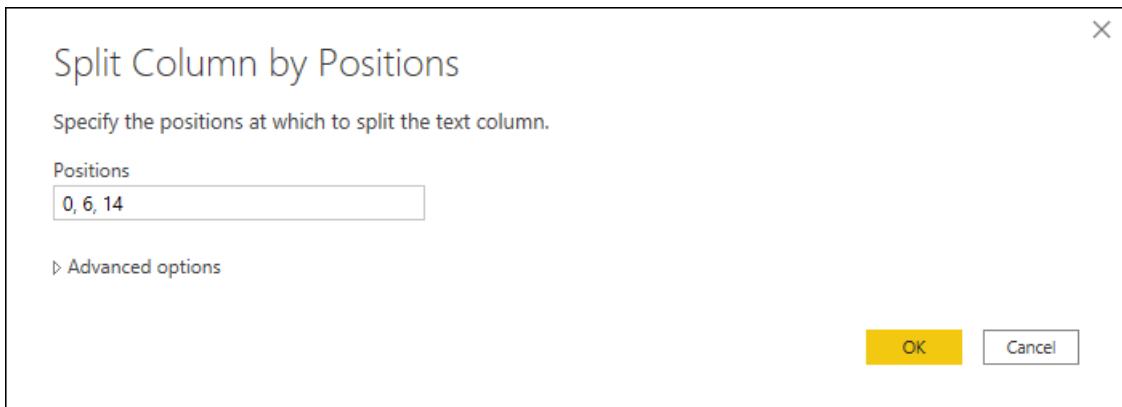
	A ^B _C Column1
1	PA000420190504150
2	US94242019112487489
3	PA4144202001123758
4	US84292020031467895

This column holds three values:

- **Account Name**—in the first six characters
- **Date**—in the next eight characters with the format yyymmdd
- **Units**—the rest of the characters

In this example, you want to split this column into the three columns made from the values in the list above. To do this split, select the column and then select the option to split the column by positions. In **Split Column by Positions**, apply the following configuration:

- **Positions:** 0,6,14
 - Positions are zero-based and comma-separated, where position zero is the start of the string.



NOTE

This operation will first start creating a column from position 0 to position 6, then from position 7 to position 14. There will be another column should there be values with a length of 16 or more characters in the current data preview contents.

The result of that operation will give you a table with three columns.

	A ^B _C Column1.1	A ^B _C Column1.2	A ^B _C Column1.3
1	PA0004	20190504	150
2	US9424	20191124	87489
3	PA4144	20200112	3758
4	US8429	20200314	67895

NOTE

Power Query will split the column into only two columns. The name of the new columns will contain the same name as the original column. A suffix created by a dot and a number that represents the split section of the column will be appended to the name of the new columns.

You can now change the name of the columns, and also define the data types of each column as follows:

ORIGINAL COLUMN NAME	NEW COLUMN NAME	DATA TYPE
Column1.1	Account Name	Text
Column1.2	Date	Date
Column1.3	Units	Whole Number

Your final table will look like the one in the following image.

	A ^B _C Account Name	Date	1 ² ₃ Units
1	PA0004	5/4/2019	150
2	US9424	11/24/2019	87489
3	PA4144	1/12/2020	3758
4	US8429	3/14/2020	67895

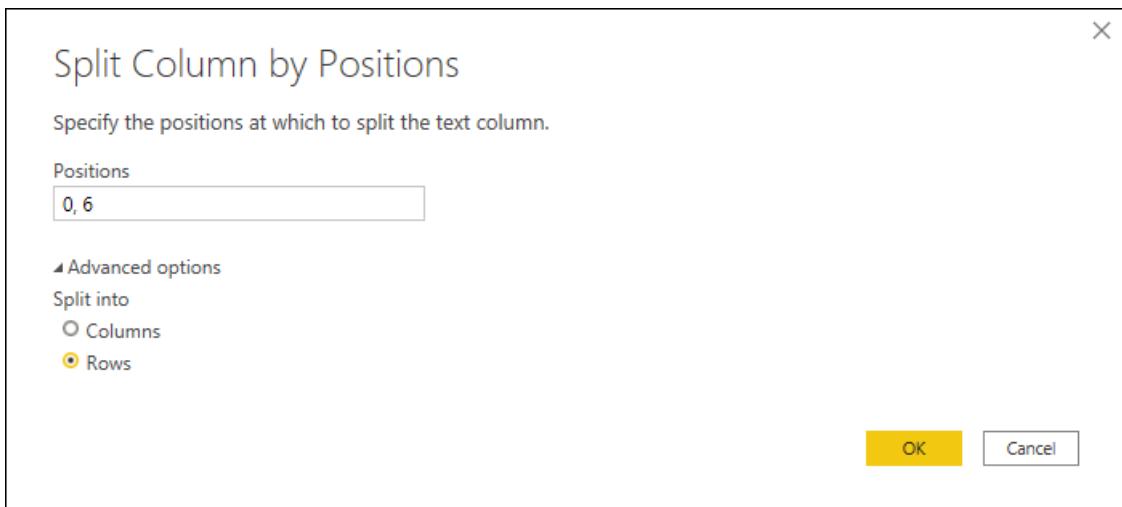
Split columns by positions into rows

The initial table for this example will be like the one in the image below, with the columns **Group** and **Account**.

A ^B C Group	A ^B C Account
1 Group A	PA0004US9424
2 Group B	PA4144US8429

The **Account** column can only hold two values in the same cell. Each value has the same length in characters, with a total of six characters. In this example, you want to split these values so you can have each account value in its own row. To do that, select the **Account** column and then select the option to split the column by positions. In **Split Column by Positions**, apply the following configuration:

- **Positions:** 0, 6
- **Split into:** Rows



NOTE

This operation will first start creating a column from position 0 to position 6. There will be another column should there be values with a length of 8 or more characters in the current data preview contents.

The result of that operation will give you a table with the same number of columns, but many more rows because the values inside the cells are now in their own cells.

A ^B C Group	A ^B C Account
1 Group A	PA0004
2 Group A	US9424
3 Group B	PA4144
4 Group B	US8429

Split columns by lowercase to uppercase

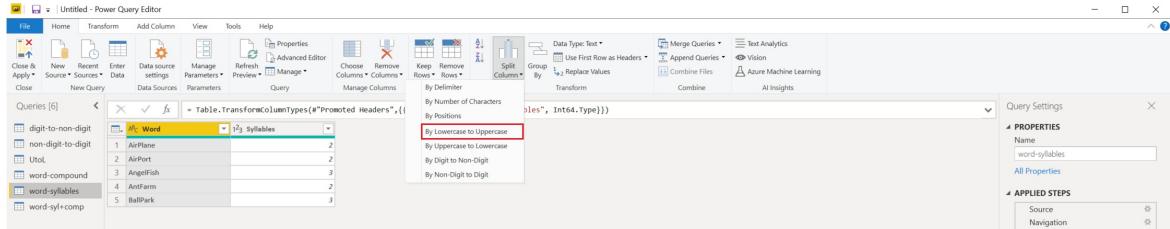
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. If your data contains CamelCased text or a similar pattern, then the column(s) selected can be split by every instance of the last lowercase letter to the next uppercase letter easily.

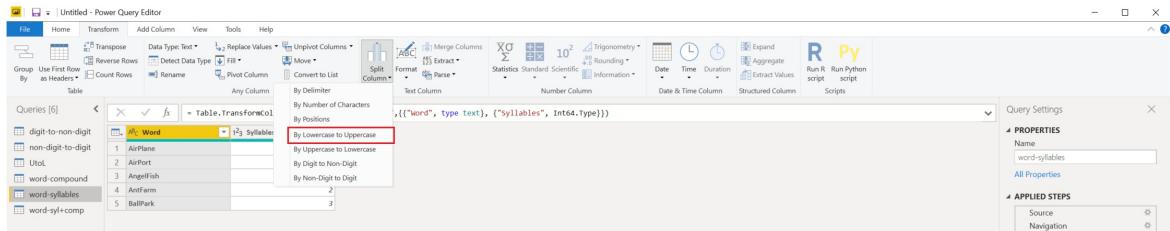
Where to find Split Columns > By Lowercase to Uppercase

You can find the **Split Columns: By Lowercase to Uppercase** option in three places:

- **Home tab**—under the **Split Column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split Column** dropdown menu inside the **Text Column** group.



- **Right-click a column**—inside the **Split Column** option.

The screenshot shows a Power BI desktop interface with a table containing five rows of data. The columns are labeled 'Word' and 'Syllables'. The 'Word' column contains values like 'AirPlane', 'AirPort', etc. The 'Syllables' column contains numerical values. A context menu is open over the 'Word' column, specifically over the first row. The menu path 'Split Column > By Lowercase to Uppercase' is highlighted with a red box.

Split columns by lowercase to uppercase into columns

The initial table in this example will be the one shown in the image below, with one column for **Word** and an extra column named **Syllables**. You'll only focus on the first column.

	AB_C Word	123 Syllables
1	AirPlane	2
2	AirPort	2
3	AngelFish	3
4	AntFarm	2
5	BallPark	3

This column holds two values in each row:

- **FirstWord**—The first half of the compound word.
- **SecondWord**—The second half of the compound word.

In this example, you want to split this column into the two columns described in the list above. Select the column and then select the option to split the column by lowercase to uppercase.

This single column will split into multiple columns, given every instance of the last lowercase letter to the next uppercase letter. In this case, it only splits into two columns.

Your final table will look like the following image.

	A ^B C Word.1	A ^B C Word.2	1 ² 3 Syllables
1	Air	Plane	2
2	Air	Port	2
3	Angel	Fish	3
4	Ant	Farm	2
5	Ball	Park	3

Split columns by uppercase to lowercase

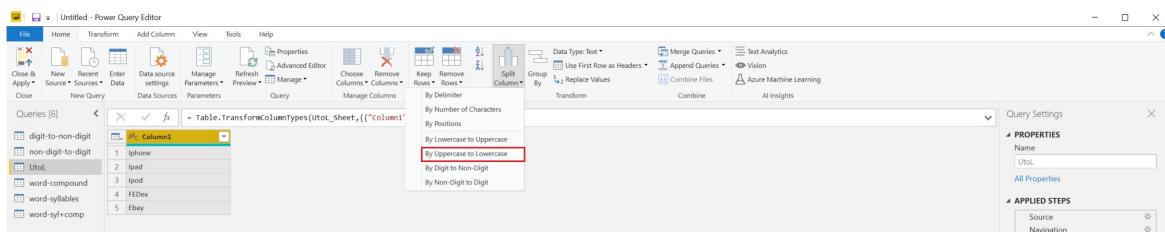
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by every instance of the last uppercase letter to the next lowercase letter.

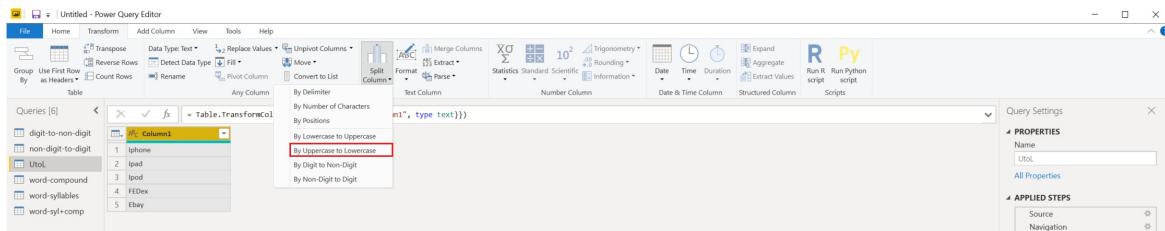
Where to find Split Columns > By Uppercase to Lowercase

You can find the **Split Columns > By Uppercase to Lowercase** option in three places:

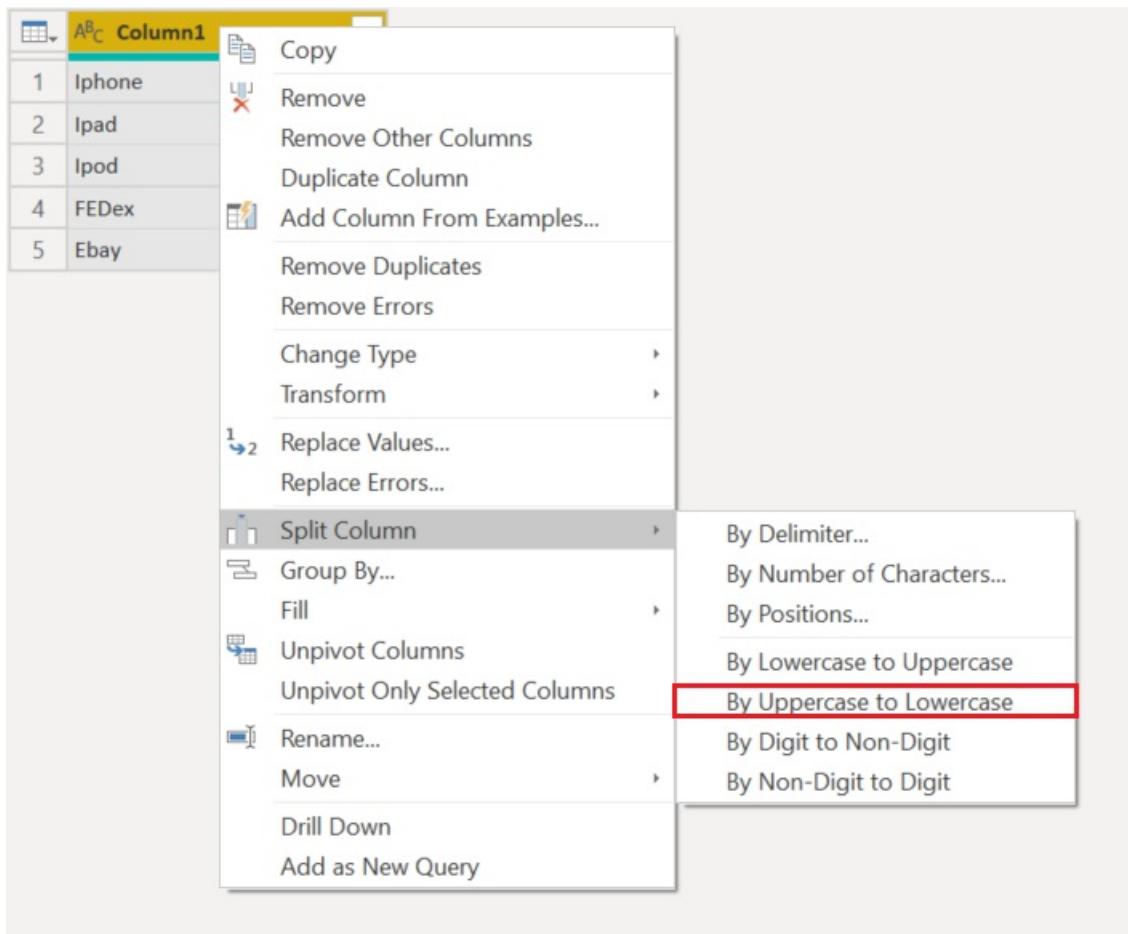
- **Home tab**—under the **Split Column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split Column** dropdown menu inside the **Text Column** group.



- **Right-click a column**—inside the **Split Column** option.



Split columns by uppercase to lowercase into columns

Your initial table in this example will be the one shown in the image below, with only one column for **Column1**.

	A ^B C Products/Brands
1	Iphone
2	Ipad
3	Ipod
4	FEDex
5	Ebay

This column holds two values:

- **FirstWord**—The first half of a product that is in camel case.
- **SecondWord**—The second half of a product that is in camel case.

In this example, you want to split this column into the two columns described in the list above. Select the column and then select the option to split the column by uppercase to lowercase.

This single column will split into multiple columns, given every instance of the last uppercase letter to the next lowercase letter. In this case, it only splits into two columns.

Your final table will look like the following image.

	A ^B C Products/Brands.1	A ^B C Products/Brands.2
1	I	phone
2	I	pad
3	I	pod
4	FED	ex
5	E	bay

Split columns by digit to non-digit

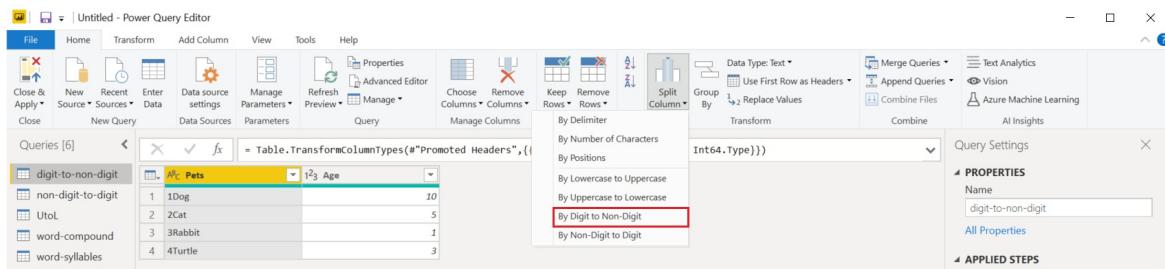
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by every instance of a digit followed by a non-digit.

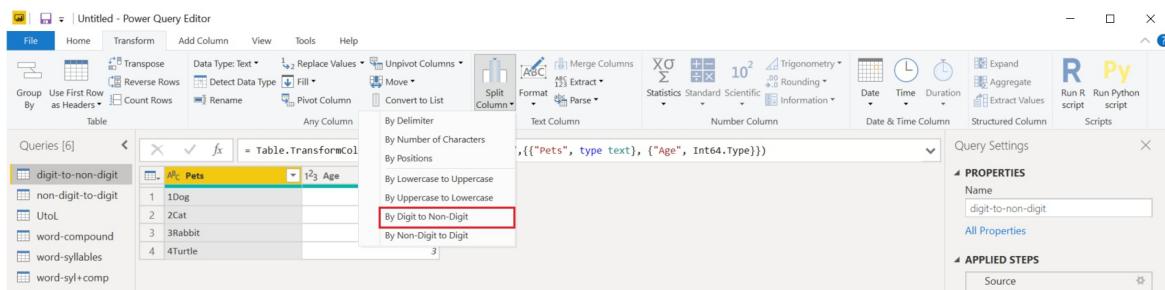
Where to find Split columns > By Digit to Non-Digit

You can find the **Split Columns: By Digit to Non-Digit** option in three places:

- **Home tab**—under the **Split Column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split Column** dropdown menu inside the **Text Column** group.



- **Right-click a column**—inside the **Split Column** option.

The screenshot shows a Power BI desktop interface with a table named 'Pets'. The 'Age' column is selected, indicated by a yellow header. A context menu is open, with 'Split Column' being the second item under the main menu. The 'By Digit to Non-Digit' option is highlighted with a red box.

	Pets	Age
1	1Dog	10
2	2Cat	5
3	3Rabbit	1
4	4Turtle	3

Split columns by digit to non-digit into columns

The initial table in this example will be the one shown in the image below, with a column for **Pets** and other extra columns. This example will only focus on the **Pets** column.

	Pets	Age
1	1Dog	10
2	2Cat	5
3	3Rabbit	1
4	4Turtle	3

This column holds two values in each row:

- **Rank**—The rank of the animal.
- **AnimalType**—The second part of the word is the type of animal.

In this example, you want to split this column into the two columns described in the list above. Select the column and then select the option to split the column by digit to non-digit.

This single column will split into multiple columns, given every instance of a digit followed with a non-digit. In this case, it only splits it into two.

Your final table will look like the following image.

	A ^B _C Pets.1	A ^B _C Pets.2	1 ² ₃ Age
1	1	Dog	10
2	2	Cat	5
3	3	Rabbit	1
4	4	Turtle	3

Split columns by non-digit to digit

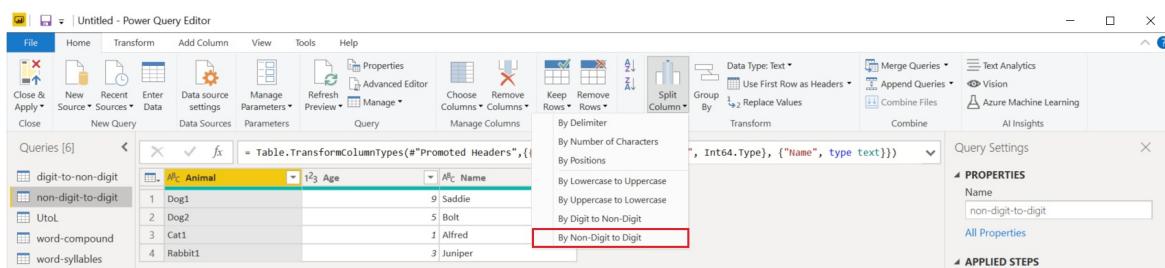
1/15/2022 • 2 minutes to read • [Edit Online](#)

In Power Query, you can split a column through different methods. In this case, the column(s) selected can be split by every instance of a non-digit followed by a digit.

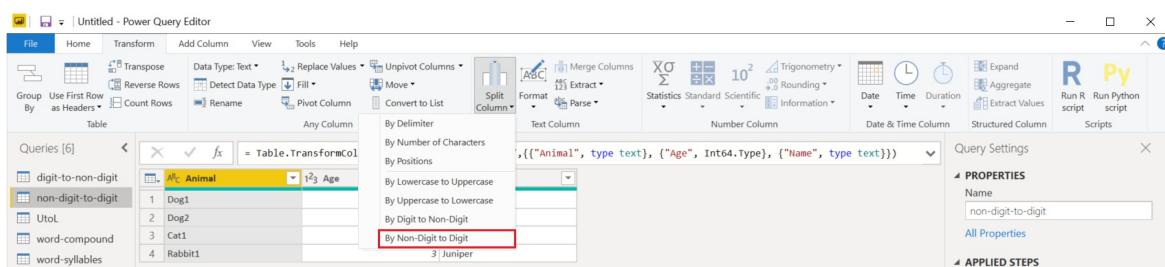
Where to find Split Columns > By Non-Digit to Digit

You can find the **Split Columns > By Non-Digit to Digit** option in three places:

- **Home tab**—under the **Split Column** dropdown menu inside the **Transform** group.



- **Transform tab**—under the **Split Column** dropdown menu inside the **Text Column** group.



- **Right-click a column**—inside the **Split Column** option.

The screenshot shows the Power BI desktop interface with a table containing four rows of animal data. A context menu is open over the first column, labeled 'Animal'. The 'Split Column' option is highlighted. A secondary menu is displayed under 'Split Column', listing several transformation methods. The 'By Non-Digit to Digit' option is specifically highlighted with a red border.

Split columns by non-digit to digit into columns

The initial table in this example will be the one shown in the image below, with a column for **Pets** and other extra columns. In this example, you'll only focus on the **Pets** column.

	ABC Animal	123 Age	ABC Name
1	Dog1		9 Saddie
2	Dog2		5 Bolt
3	Cat1		1 Alfred
4	Rabbit1		3 Juniper

This column holds two values in each row:

- **AnimalType**—The first part is the type of animal.
- **Number**—The animal number that came into the person's life.

In this example, you want to split this column into the two columns described in the list above. Select the column and then select the option to split the column by non-digit to digit.

This single column will split into multiple columns, given every instance of a digit followed by a non-digit. In this case, it only splits into two columns.

Your final table will look like the following image.

	A ^B _C Animal.1	A ^B _C Animal.2	1 ² ₃ Age	A ^B _C Name
1	Dog	1		9 Sadie
2	Dog	2		5 Bolt
3	Cat	1		1 Alfred
4	Rabbit	1		3 Juniper

What are dataflows?

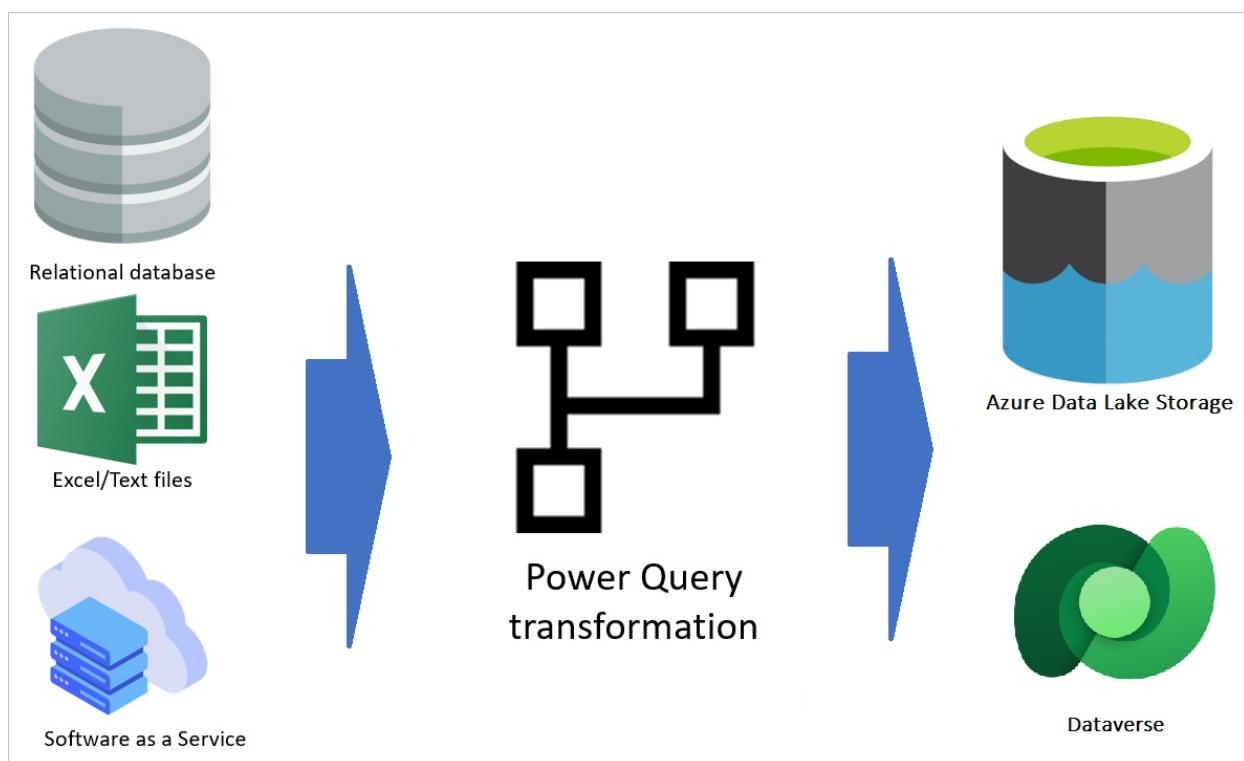
1/15/2022 • 4 minutes to read • [Edit Online](#)

Dataflows are a self-service, cloud-based, data preparation technology. Dataflows enable customers to ingest, transform, and load data into Microsoft Dataverse environments, Power BI workspaces, or your organization's Azure Data Lake Storage account. Dataflows are authored by using Power Query, a unified data connectivity and preparation experience already featured in many Microsoft products, including Excel and Power BI. Customers can trigger dataflows to run either on demand or automatically on a schedule; data is always kept up to date.

Dataflows can be created in multiple Microsoft products

Dataflows are featured in multiple Microsoft products and don't require a dataflow-specific license to be created or run. Dataflows are available in Power Apps, Power BI, and Dynamics 365 Customer Insights. The ability to create and run dataflows is bundled with those products' licenses. Dataflow features are mostly common across all products they're featured in, but some product-specific features might exist in dataflows created in one product versus another.

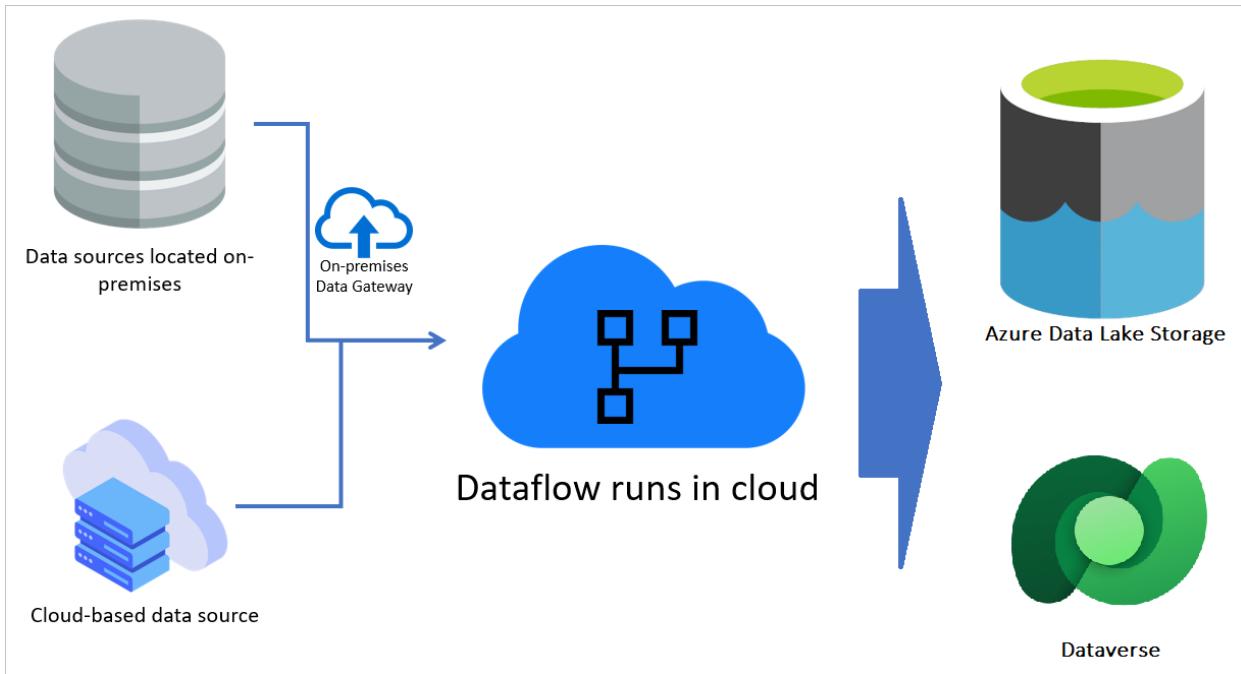
How does the dataflow function?



The previous image shows an overall view of how a dataflow is defined. A dataflow gets data from different data sources (more than 80 data sources are supported already). Then, based on the transformations configured with the Power Query authoring experience, the dataflow transforms the data by using the dataflow engine. Finally, the data is loaded to the output destination, which can be a Microsoft Power Platform environment, a Power BI workspace, or the organization's Azure Data Lake Storage account.

Dataflows run in the cloud

Dataflows are cloud-based. When a dataflow is authored and saved, its definition is stored in the cloud. A dataflow also runs in the cloud. However, if a data source is on-premises, an on-premises data gateway can be used to extract the data to the cloud. When a dataflow run is triggered, the data transformation and computation happens in the cloud, and the destination is always in the cloud.



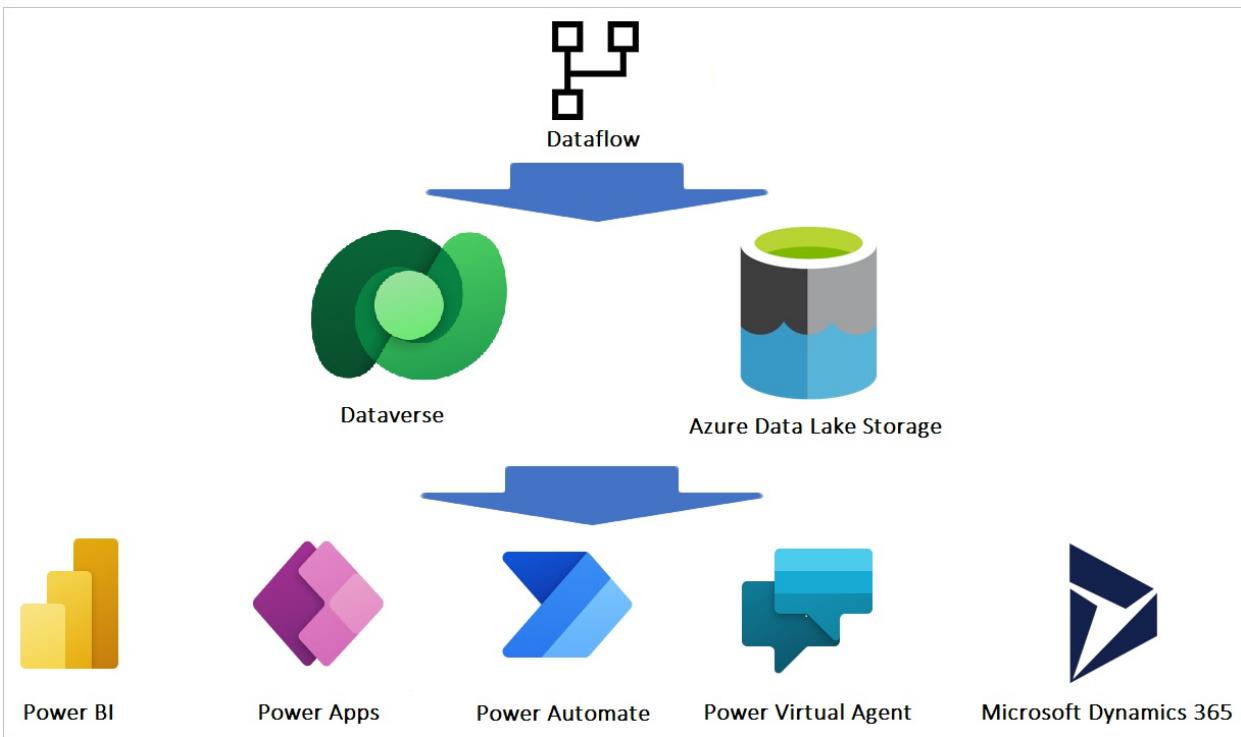
Dataflows use a powerful transformation engine

Power Query is the data transformation engine used in the dataflow. This engine is capable enough to support many advanced transformations. It also uses a straightforward, yet powerful, graphical user interface called Power Query Editor. You can use dataflows with this editor to develop your data integration solutions faster and more easily.

The screenshot shows the Microsoft Power Query - Edit queries interface. The top navigation bar includes Home, Transform, Add column, and View tabs. The ribbon has sections for Get data, Options, Manage parameters, Refresh, Advanced editor, Properties, Choose columns, Remove columns, Keep rows, Remove rows, Sort, Data type, Use first row as headers, Split column, Group by, Replace values, Merge queries, Append queries, Combine files, Map to standard, CDM, and AI insights. The main area displays a table titled "Table.RemoveColumns(#"Renamed Columns3", {"FiscalBaseDate"})". The table contains 26 rows of data, each with columns for Date, Year, Start of Year, End of Year, Month, Start of Month, End of Month, Days in Month, Day, Day Name, Day of Week, Day of Year, and Month N. The "Applied steps" pane on the right lists various transformations applied to the data, such as FromYear, ToYear, StartofFiscalYear, firstDayofWeek, FromDate, ToDate, Source, Converted to T, Renamed Column, Changed Type, Inserted Year, Inserted Start of Year, Inserted End of Year, Inserted Month, Inserted Start of Month, Inserted End of Month, Inserted Days, Inserted Day, Inserted Day of Month, Inserted Day of Year, and Inserted Month N.

Dataflow integration with Microsoft Power Platform and Dynamics 365

Because a dataflow stores the resulting entities in cloud-based storage, other services can interact with the data produced by dataflows.



For example, Power BI, Power Apps, Power Automate, Power Virtual Agents, and Dynamics 365 applications can get the data produced by the dataflow by connecting to Dataverse, a Power Platform dataflow connector, or directly through the lake, depending on the destination configured at dataflow creation time.

Benefits of dataflows

The following list highlights some of the benefits of using dataflows:

- A dataflow decouples the data transformation layer from the modeling and visualization layer in a Power BI solution.
- The data transformation code can reside in a central location, a dataflow, rather than be spread out among multiple artifacts.
- A dataflow creator only needs Power Query skills. In an environment with multiple creators, the dataflow creator can be part of a team that together builds the entire BI solution or operational application.
- A dataflow is product-agnostic. It's not a component of Power BI only; you can get its data in other tools and services.
- Dataflows take advantage of Power Query, a powerful, graphical, self-service data transformation experience.
- Dataflows run entirely in the cloud. No additional infrastructure is required.
- You have multiple options for starting to work with dataflows, using licenses for Power Apps, Power BI, and Customer Insights.
- Although dataflows are capable of advanced transformations, they're designed for self-service scenarios and require no IT or developer background.

Use-case scenarios for dataflows

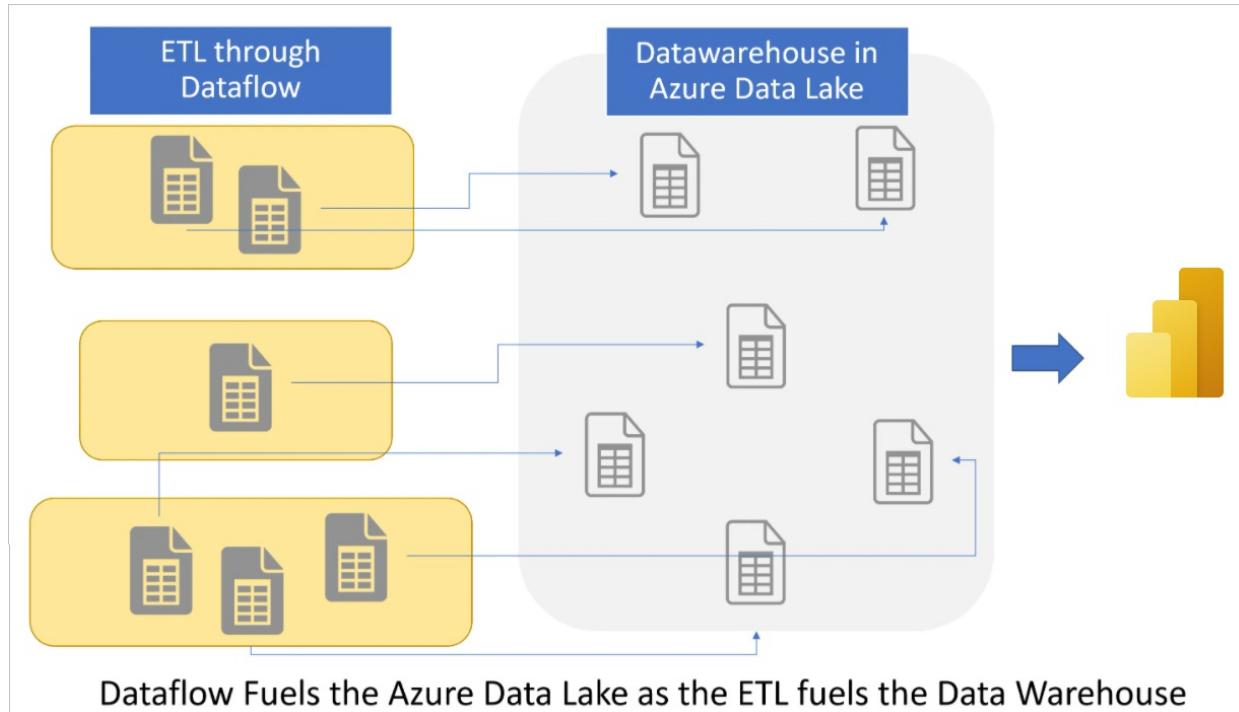
You can use dataflows for many purposes. The following scenarios provide a few examples of common use cases for dataflows.

Data migration from legacy systems

In this scenario, the decision has been made by an organization to use Power Apps for the new user interface experience rather than the legacy on-premises system. Power Apps, Power Automate, and AI Builder all use Dataverse as the primary data storage system. The current data in the existing on-premises system can be migrated into Dataverse by using a dataflow, and then these products can use that data.

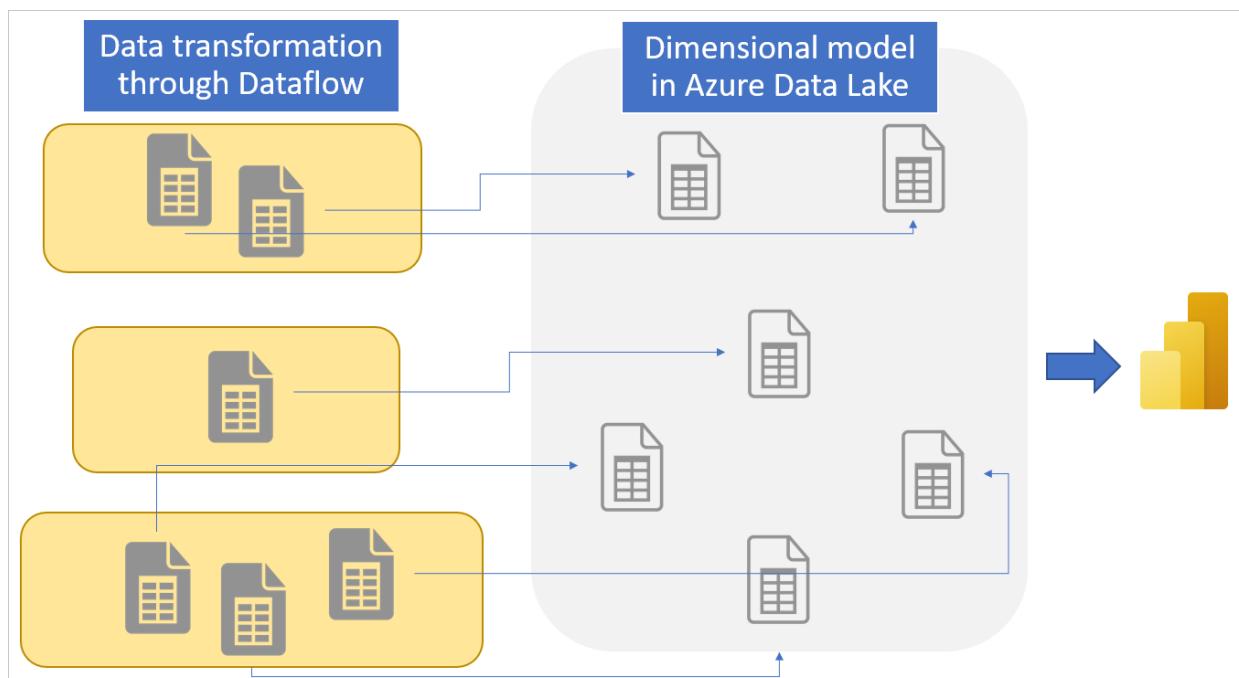
Using dataflows to build a data warehouse

You can use dataflows as a replacement for other extract, transform, load (ETL) tools to build a data warehouse. In this scenario, the data engineers of a company decide to use dataflows to build their star schema–designed data warehouse, including fact and dimension tables in Data Lake Storage. Then Power BI is used to generate reports and dashboards by getting data from the dataflows.



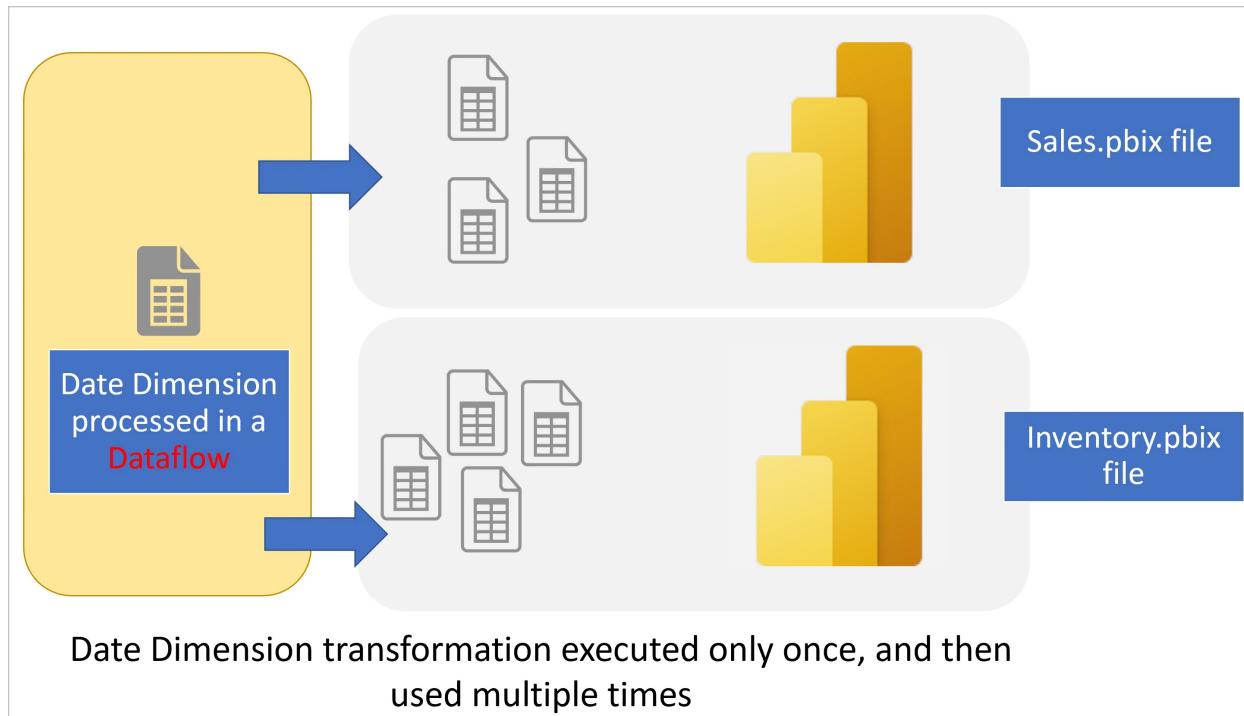
Using dataflows to build a dimensional model

You can use dataflows as a replacement for other ETL tools to build a dimensional model. For example, the data engineers of a company decide to use dataflows to build the star-schema designed dimensional model, including fact and dimension tables in Azure Data Lake Storage Gen2. Then Power BI is used to generate reports and dashboards by getting data from the dataflows.



Centralize data preparation and reuse of datasets across multiple Power BI solutions

If multiple Power BI solutions are using the same transformed version of a table, the process to create the table will be repeated multiple times. This increases the load on the source system, consumes more resources, and creates duplicate data with multiple points of failure. Instead, a single dataflow can be created to compute the data for all solutions. Power BI can then reuse the result of the transformation in all solutions. The dataflow, if used in such a way, can be part of a robust Power BI implementation architecture that avoids the Power Query code duplicates and reduces the maintenance costs of the data integration layer.



Next steps

The following articles provide further study materials for dataflows.

- [Create and use dataflows in Microsoft Power Platform](#)
- [Creating and using dataflows in Power BI](#)

Understanding the differences between dataflow types

1/15/2022 • 5 minutes to read • [Edit Online](#)

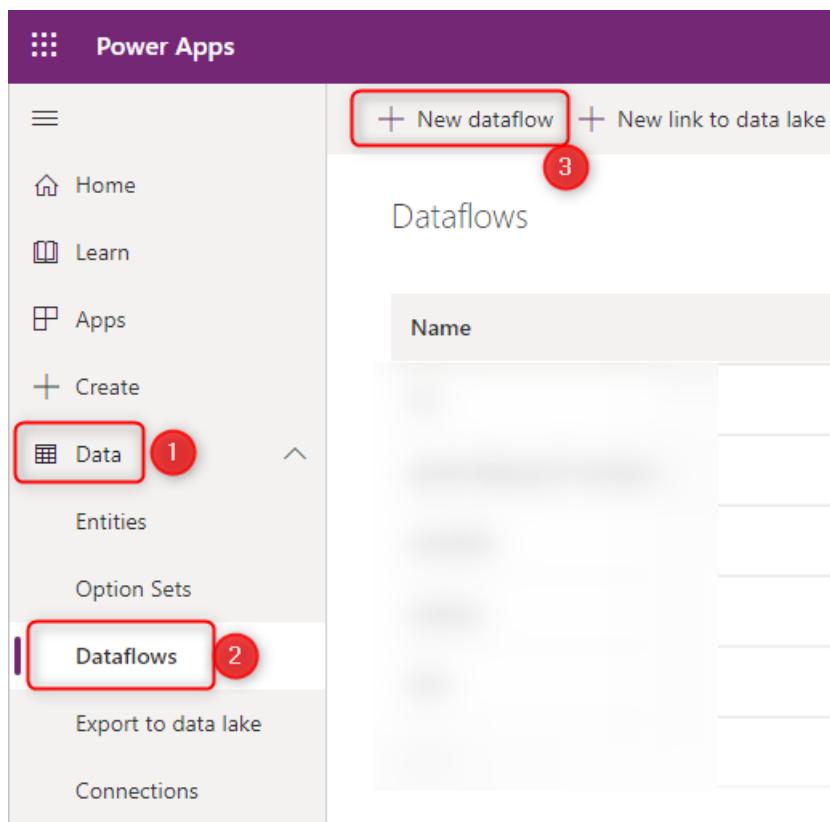
Dataflows are used to extract, transform, and load data to a storage destination where it can be leveraged for different scenarios. Because not all storage destinations share the same characteristics, some dataflow features and behaviors differ depending on the storage destination the dataflow loads data into. Before you create a dataflow, it's important to understand how the data is going to be used, and choose the storage destination according to the requirements of your solution.

Selecting a storage destination of a dataflow determines the dataflow's type. A dataflow that loads data into Dataverse tables is categorized as a *standard dataflow*. Dataflows that load data to analytical entities is categorized as an *analytical dataflow*.

Dataflows created in Power BI are always analytical dataflows. Dataflows created in Power Apps can either be standard or analytical, depending on your selection when creating the dataflow.

Standard dataflows

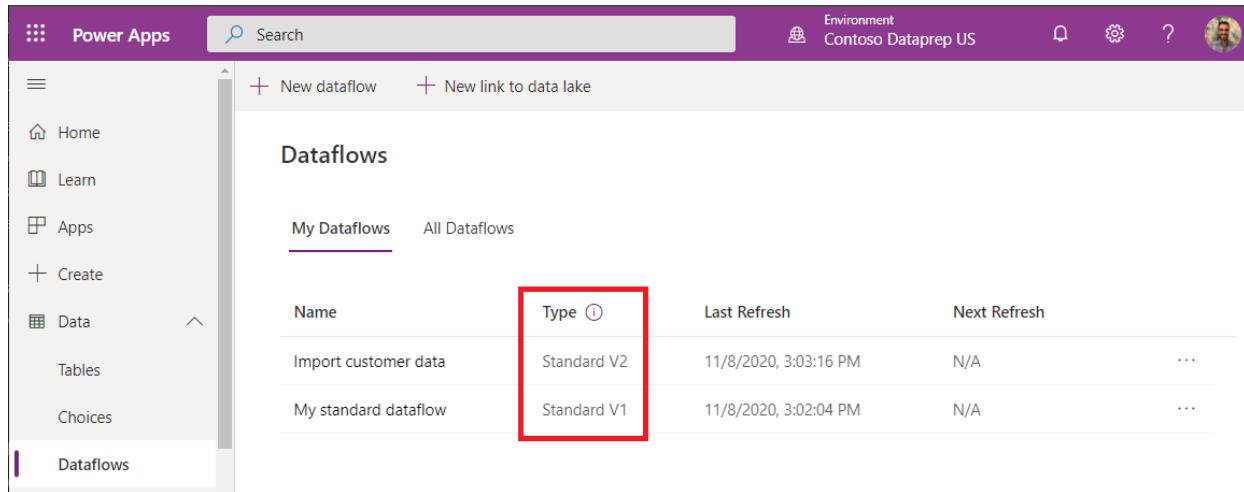
A standard dataflow loads data to Dataverse tables. Standard dataflows can only be created in Power Apps. One benefit of creating this type of dataflow is that any application that depends on data in Dataverse can work with the data created by standard dataflows. Typical applications that leverage Dataverse tables are Power Apps, Power Automate, AI Builder and Power Virtual Agents.



Standard dataflows versions

We've been working on significant updates to standard dataflows to improve their performance and reliability. These improvements will eventually be available to all standard dataflows. But in the interim, we'll differentiate between existing standard dataflows (version 1) and new standard dataflows (version 2) by adding a version

indicator in Power Apps.



The screenshot shows the Power Apps Dataflows page. On the left, there's a navigation sidebar with options like Home, Learn, Apps, Create, Data (Tables, Choices), and Dataflows. The main area is titled 'Dataflows' and has tabs for 'My Dataflows' (selected) and 'All Dataflows'. Below is a table with columns: Name, Type (with a help icon), Last Refresh, and Next Refresh. Two rows are listed: 'Import customer data' (Standard V2, last refreshed 11/8/2020, 3:03:16 PM, next refresh N/A) and 'My standard dataflow' (Standard V1, last refreshed 11/8/2020, 3:02:04 PM, next refresh N/A). The 'Type' column is highlighted with a red box.

Standard dataflow versions feature comparison

The following table lists the major features differences between standard dataflows V1 and V2, and provides information about each feature's behavior in each version.

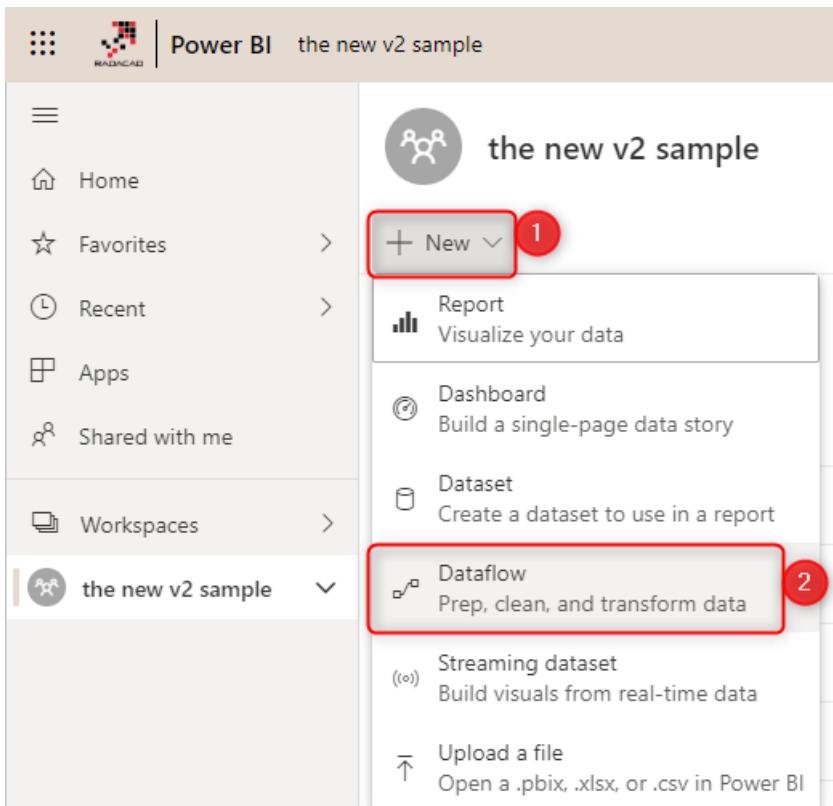
FEATURE	STANDARD V1	STANDARD V2
Maximum number of dataflows that can be saved with automatic schedule per customer tenant	50	Unlimited
Maximum number of records ingested per query/table	500,000	Unbounded. The maximum number of records that can be ingested per query or table now depends on Dataverse service protection limits at the time of ingestion.
Ingestion speed into Dataverse	Baseline performance	Improved performance by a few factors. Actual results may vary and depend on characteristics of the data ingested, and load on Dataverse service at the time of ingestion.
Incremental Refresh policy	Not supported	Supported
Resiliency	When Dataverse service protection limits are encountered, a record will be retried up to 3 times.	When Dataverse service protection limits are encountered, a record will be retried up to 3 times.

Analytical dataflows

An analytical dataflow loads data to storage types optimized for analytics—Azure Data Lake Storage. Microsoft Power Platform environments and Power BI workspaces provide customers with a managed analytical storage location that's bundled with those product licenses. In addition, customers can link their organization's Azure Data Lake storage account as a destination for dataflows.

Analytical dataflows are capable additional analytical features. For example, integration with Power BI's AI features or use of computed entities which will be discussed later.

You can create analytical dataflows in Power BI. By default, they'll load data to Power BI's managed storage. But you can also configure Power BI to store the data in the organization's Azure Data Lake Storage.



You can also create analytical dataflows in Power Apps and Dynamics 365 customer insights portals. When you're creating a dataflow in Power Apps portal, you can choose between Dataverse manages analytical storage or in your organization's Azure Data Lake Storage account.

New Dataflow

Name *

analytical sample dataflow

Options

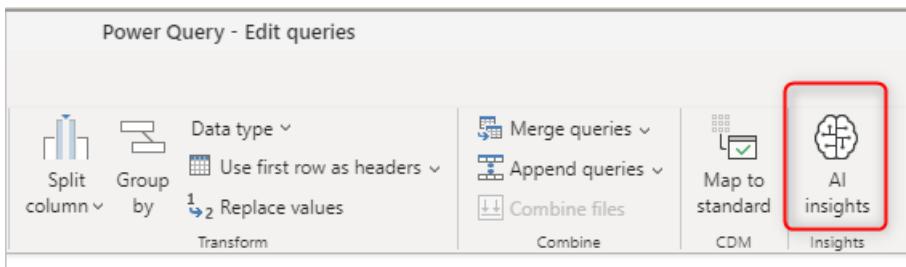
Analytical entities only

Dataflows allow users to extract, transform and load data from a wide range of data sources that can be used from Power Apps, Flow and analytics. [Learn more](#)

Create Cancel

AI Integration

Sometimes, depending on the requirement, you might need to apply some AI and machine learning functions on the data through the dataflow. These functionalities are available in Power BI dataflows and require a Premium workspace.



The following articles discuss how to use AI functions in a dataflow:

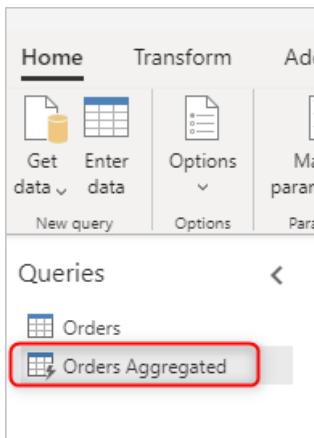
- [Azure Machine Learning integration in Power BI](#)
- [Cognitive Services in Power BI](#)
- [Automated Machine Learning in Power BI](#)

Note that the features listed above are Power BI specific and are not available when creating a dataflow in the Power Apps or Dynamics 365 customer insights portals.

Computed entities

One of the reasons to use a computed entity is the ability to process large amounts of data. The computed entity helps in those scenarios. If you have an entity in a dataflow, and another entity in the same dataflow uses the first entity's output, this will create a computed entity.

The computed entity helps with the performance of the data transformations. Instead of re-doing the transformations needed in the first entity multiple times, the transformation will be done only once in the computed entity. Then the result will be used multiple times in other entities.



To learn more about computed entities, see [Using computed entities on Power BI Premium](#).

Computed entities are available only in an analytical dataflow.

Standard vs. analytical dataflows

The following table lists some differences between a standard entity and an analytical entity.

OPERATION	STANDARD	ANALYTICAL
How to create	Power Platform dataflows	Power BI dataflows Power Platform dataflows by selecting the Analytical Entity checkbox when creating the dataflow

OPERATION	STANDARD	ANALYTICAL
Storage options	Dataverse	Azure Data Lake Storage internal for the Power BI dataflows Azure Data Lake Storage external attached to the Power BI or Power Platform dataflows
Power Query transformations	Yes	Yes
AI functions	No	Yes
Computed entity	No	Yes
Can be used in other applications	Yes, through Dataverse	Power BI dataflows: Only in Power BI Power Platform dataflows or Power BI external dataflows: Yes, through Azure Data Lake Storage
Mapping to standard Entity	Yes	Yes
Incremental load	Default incremental-load Possible to change using the Delete rows that no longer exist in the query output checkbox at the load settings	Default full-load Possible to set up incremental refresh by setting up the incremental refresh in the dataflow settings
Scheduled Refresh	Yes	Yes, the possibility of notifying the dataflow owners upon the failure

Scenarios to use each dataflow type

Here are some sample scenarios and best practice recommendations for each type of dataflow.

Cross-platform usage—standard dataflow

If your plan for building dataflows is to use stored data in multiple platforms (not only Power BI, but also other Microsoft Power Platform services, Dynamics 365, and so on), a standard dataflow is a great choice. Standard dataflows store the data in Dataverse, which you can access through many other platforms and services.

Heavy data transformations on large data tables—analytical dataflow

Analytical dataflows are an excellent option for processing large amounts of data. Analytical dataflows also enhance the computing power behind the transformation. Having the data stored in Azure Data Lake Storage increases the writing speed to a destination. Compared to Dataverse (which might have many rules to check at the time of data storage), Azure Data Lake Storage is faster for read/write transactions on a large amount of data.

AI features—analytical dataflow

If you're planning to use any AI functionality through the data transformation stage, you'll find it helpful to use an analytical dataflow because you can use all the supported AI features with this type of dataflow.

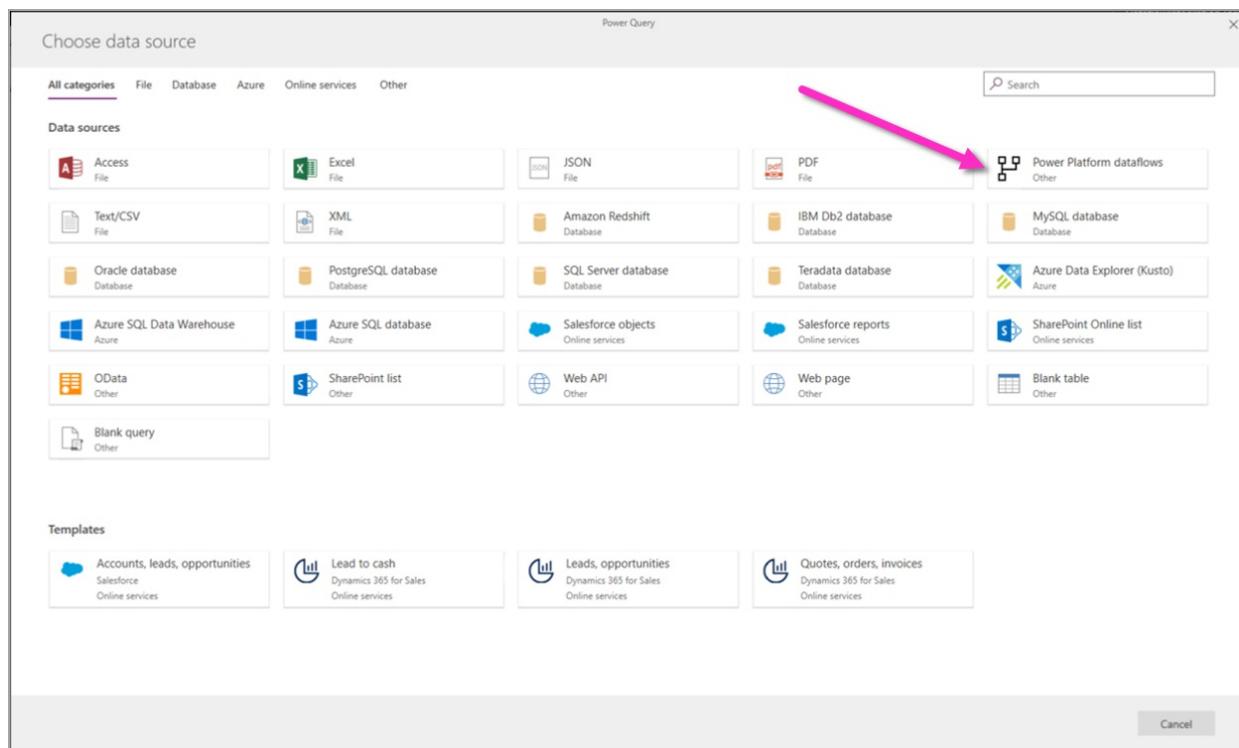
Create and use dataflows in Microsoft Power Platform

1/15/2022 • 5 minutes to read • [Edit Online](#)

Using dataflows with Microsoft Power Platform makes data preparation easier, and lets you reuse your data preparation work in subsequent reports, apps, and models.

In the world of ever-expanding data, data preparation can be difficult and expensive, consuming as much as 60 to 80 percent of the time and cost for a typical analytics project. Such projects can require wrangling fragmented and incomplete data, complex system integration, data with structural inconsistency, and a high skillset barrier.

To make data preparation easier and to help you get more value out of your data, Power Query and Power Platform dataflows were created.



With dataflows, Microsoft brings the self-service data preparation capabilities of Power Query into the Power BI and Power Apps online services, and expands existing capabilities in the following ways:

- **Self-service data prep for big data with dataflows:** Dataflows can be used to easily ingest, cleanse, transform, integrate, enrich, and schematize data from a large and ever-growing array of transactional and observational sources, encompassing all data preparation logic. Previously, extract, transform, load (ETL) logic could only be included within datasets in Power BI, copied over and over between datasets, and bound to dataset management settings.

With dataflows, ETL logic is elevated to a first-class artifact within Microsoft Power Platform services, and includes dedicated authoring and management experiences. Business analysts, BI professionals, and data scientists can use dataflows to handle the most complex data preparation challenges and build on each other's work, thanks to a revolutionary model-driven calculation engine, which takes care of all the transformation and dependency logic—cutting time, cost, and expertise to a fraction of what's traditionally been required for those tasks. You can create dataflows by using the well-known, self-service data preparation experience of Power Query. Dataflows are created and easily managed in app

workspaces or environments, in Power BI or Power Apps, respectively, enjoying all the capabilities these services have to offer, such as permission management and scheduled refreshes.

- **Load data to Dataverse or Azure Data Lake Storage:** Depending on your use case, you can store data prepared by Power Platform dataflows in the Dataverse or your organization's Azure Data Lake Storage account:
 - **Dataverse** lets you securely store and manage data that's used by business applications. Data within Dataverse is stored in a set of tables. A *table* is a set of rows (formerly referred to as records) and columns (formerly referred to as fields/attributes). Each column in the table is designed to store a certain type of data, for example, name, age, salary, and so on. Dataverse includes a base set of standard tables that cover typical scenarios, but you can also create custom tables specific to your organization and populate them with data by using dataflows. App makers can then use Power Apps and Power Automate to build rich applications that use this data.
 - **Azure Data Lake Storage** lets you collaborate with people in your organization using Power BI, Azure Data, and AI services, or using custom-built Line of Business Applications that read data from the lake. Dataflows that load data to an Azure Data Lake Storage account store data in Common Data Model folders. Common Data Model folders contain schematized data and metadata in a standardized format, to facilitate data exchange and to enable full interoperability across services that produce or consume data stored in an organization's Azure Data Lake Storage account as the shared storage layer.
- **Advanced Analytics and AI with Azure:** Power Platform dataflows store data in Dataverse or Azure Data Lake Storage—which means that data ingested through dataflows is now available to data engineers and data scientists to leverage the full power of Azure Data Services, such as Azure Machine Learning, Azure Databricks, and Azure SQL Data Warehouse for advanced analytics and AI. This enables business analysts, data engineers, and data scientists to collaborate on the same data within their organization.
- **Support for Common Data Model:** Common Data Model is a set of a standardized data schemas and a metadata system to allow consistency of data and its meaning across applications and business processes. Dataflows support Common Data Model by offering easy mapping from any data in any shape into the standard Common Data Model entities, such as Account and Contact. Dataflows also land the data, both standard and custom entities, in schematized Common Data Model form. Business analysts can take advantage of the standard schema and its semantic consistency, or customize their entities based on their unique needs. Common Data Model continues to evolve as part of the [Open Data Initiative](#).

Dataflow capabilities in Microsoft Power Platform services

Most dataflow capabilities are available in both Power Apps and Power BI. Dataflows are available as part of these services' plans. Some dataflow features are either product-specific or available in different product plans. The following table describes dataflow features and their availability.

DATAFLOW CAPABILITY	POWER APPS	POWER BI
Scheduled refresh	Up to 48 per day	Up to 48 per day
Maximum per entity refresh time	Up to 2 hours	Up to 2 hours
Dataflow authoring with Power Query Online	Yes	Yes

DATAFLOW CAPABILITY	POWER APPS	POWER BI
Dataflow management	In Power Apps admin portal	In Power BI admin portal
New connectors	Yes	Yes
Standardized schema / built-in support for the Common Data Model	Yes	Yes
Dataflows Data Connector in Power BI Desktop	For dataflows with Azure Data Lake Storage as the destination	Yes
Integration with the organization's Azure Data Lake Storage	Yes	Yes
Integration with Dataverse	Yes	No
Dataflow linked entities	For dataflows with Azure Data Lake Storage as the destination	Yes
Computed Entities (in-storage transformations using M)	For dataflows with Azure Data Lake Storage as the destination	Power BI Premium only
Dataflow incremental refresh	For dataflows with Azure Data Lake Storage as the destination, requires Power Apps Plan2	Power BI Premium only
Running on Power BI Premium capacity / parallel execution of transforms	No	Yes

More information about dataflows in Power Apps:

- [Self-service data prep in Power Apps](#)
- [Creating and using dataflows in Power Apps](#)
- [Connect Azure Data Lake Storage Gen2 for dataflow storage](#)
- [Add data to a table in Dataverse by using Power Query](#)
- Visit the Power Apps [dataflow community](#) and share what you're doing, ask questions, or [submit new ideas](#)
- Visit the Power Apps dataflow community forum and share what you're doing, ask questions, or [submit new ideas](#)

More information about dataflows in Power BI:

- [Self-service data prep in Power BI](#)
- [Create and use dataflows in Power BI](#)
- [Dataflows whitepaper](#)
- Detailed [video](#) of a dataflows walkthrough
- Visit the Power BI [dataflows community](#) and share what you're doing, ask questions, or [submit new ideas](#)

Next steps

The following articles go into more detail about common usage scenarios for dataflows.

- [Using incremental refresh with dataflows](#)
- [Creating computed entities in dataflows](#)
- [Connect to data sources for dataflows](#)
- [Link entities between dataflows](#)

For more information about Common Data Model and the Common Data Model folder standard, read the following articles:

- [Common Data Model - overview](#)
- [Common Data Model folders](#)
- [Common Data Model folder model file definition](#)

Create and use dataflows in Microsoft Teams (Preview)

1/15/2022 • 5 minutes to read • [Edit Online](#)

NOTE

We are rolling out dataflows for Microsoft Teams gradually. This feature might not be available in your region yet.

[Microsoft Dataverse for Teams](#) delivers a built-in, low-code data platform for Microsoft Teams. It provides relational data storage, rich data types, enterprise-grade governance, and one-click solution deployment. Dataverse for Teams enables everyone to easily build and deploy apps.

Before today, the way to get data into Dataverse for Teams was by manually adding data directly into a table. This process can be prone to errors and isn't scalable. But now, with self-service data prep you can find, clean, shape, and import your data into Dataverse for Teams.

With your organizational data already sitting in a different location, you can use Power Query dataflows to directly access your data through the connectors and load the data into Dataverse for Teams. When you update in your organizational data, you can refresh your dataflows by just one click and the data in Dataverse for Teams is updated too. You can also use the Power Query data transformations to easily validate and clean your data and enforce data quality for your Apps.

[Dataflows](#) were introduced to help organizations retrieve data from disparate sources and prepare it for consumption. You can easily create dataflows using the familiar, [self-service Power Query](#) experience to ingest, transform, integrate, and enrich data. When creating a dataflow, you'll connect to data, transform the data, and load data into Dataverse for Teams tables. Once the dataflow is created, it begins the process of importing data into the Dataverse table. Then you can start building apps to leverage that data.

Create a dataflow from the dataflows page

In this example, you're going to load data from an Excel file that's located on OneDrive into Dataverse for Teams.

1. Sign in to [Teams web version](#), and then select the link for **Power Apps**.
2. Select the **Build** tab, and then select **Dataflows (Preview)**.

Name	Last Refresh	Status
Dataflow of Miquella	5/19/2021, 4:20:00 PM	Published
New dataflow 1	5/19/2021, 11:48:32 AM	Published
New dataflow 2	N/A	Published
New dataflow 3	5/19/2021, 4:18:39 PM	Published

3. To create a new dataflow, select the **New** button.

The screenshot shows the Microsoft Teams interface with the 'Power Apps' app open. In the left sidebar, there are icons for Chat, Activity, Teams, Calendar, Calls, Files, and Power Apps. The main content area is titled 'Dataflows for Teams' and shows a list of items created for this team. The 'Dataflows (Preview)' tab is selected. At the bottom right of the list, there is a purple 'New' button, which is highlighted with a red box.

4. Select the Excel workbook connector.

The screenshot shows the 'Power Query - Choose data source' dialog box. It has tabs for 'All categories', 'File', 'Power Platform', 'Online services', and 'Other'. Under the 'Data sources' section, there are several options: 'Excel workbook' (highlighted with a red box), 'JSON File', 'PDF File', 'SharePoint folder File', 'Text/CSV File', 'XML File', 'SharePoint Online list Online services', 'OData Other', 'Blank table Other', and 'Blank query Other'. A search bar is at the top right, and a 'Cancel' button is at the bottom right.

5. Enter a URL address in **File path or URL**, or use the **Browse OneDrive** button to navigate through your OneDrive folders. Select the file you want, and then select the **Next** button. For more information about using the OneDrive connection or getting data, see [SharePoint and OneDrive for Business files import](#) or [Getting data from other sources](#).
6. In **Navigator**, select the tables that are present in your Excel file. If your Excel file has multiple sheets and tables, select only the tables you're interested in. When you're done, select **Transform data**.
7. Clean and transform your data using Power Query. You can use the out-of-the box transformations to delete missing values, delete unnecessary columns, or to filter your data. With Power Query, you can apply more than 300 different transformations on your data. To learn more about Power Query transformations, see [Use Power Query to transform data](#). After you're finished with preparing your data, select **Next**.

The screenshot shows the Microsoft Power Query interface within the Microsoft Teams application. The main area displays a table titled "1000 Sales Records" from an Excel workbook. The table has 12 columns and 12 rows of data. The columns are: Item Type, Sales Channel, Order Priority, Order Date, Ship Date, Units Sold, Unit Price, Unit Cost, Total Revenue, and Total Profit. The data includes various items like Cosmetics, Vegetables, Baby Food, Cereal, Fruits, and Snacks, with their respective sales details.

- In **Map tables**, select **Load to new table** to create a new table in Dataverse for Teams. You can also choose to load your data into an existing table. In the **Map tables** screen, you can also specify a **Unique primary name column** and an **Alternate key column (optional)**. In this example, leave these selections with the default values. To learn more about mapping your data and the different settings, see [Field mapping considerations for standard dataflows](#).

The screenshot shows the Microsoft Power Query interface within the Microsoft Teams application. The main area displays the "Power Query - Map tables" screen. The "Load settings" section is highlighted with a red box, showing three options: "Load to new table" (selected), "Load to existing table", and "Do not load". Below this, there are fields for "Table name" (set to "Sales Records"), "Table display name" (also set to "Sales Records"), and "Table description". The "Column mapping" section on the right lists source columns and destination columns for 12 data fields: Country, Item Type, Order Date, Order ID, Order Priority, Region, Sales Channel, Ship Date, Total Cost, Total Profit, Total Revenue, and Unit Cost. Each source column is mapped to a destination column of the same type (Text, Date only, Whole number, Text, Text, Text, Text, Date only, Floating point number, Floating point number, Floating point number, Floating point number).

- Select **Create** to finish your dataflow. Once you've created your dataflow, data begins loading into Dataverse for Teams. This process can take some time and you can use the management page to check the status. When a dataflow completes a run, its data is available to use.

Managing your dataflows

You can manage any dataflow you created from the **Dataflows (Preview)** tab. Here, you can see the status of all dataflows, when your dataflow was last refreshed, and take action from the action bar.

Items created for Dataflows for Teams			
Name	Last Refresh	Status	
Dataflow of Miquella	5/19/2021, 4:20:00 PM	Published	...
New dataflow 1	5/19/2021, 11:48:32 AM	Published	...
New dataflow 2	N/A	Published	...
New dataflow 3	5/19/2021, 4:18:39 PM	Published	...

In the **Last Refresh** column, you can see when your data was last refreshed. If your refresh failed, an error indication appears. If you select the error indication, the details of the error and recommended steps to address it appear.

In the **Status** column, you can see the current status of the dataflow. Possible states are:

- **Refresh in progress:** the dataflow is extracting, transforming, and loading your data from the source to the Dataverse Tables. This process can take several minutes depending on the complexity of transformations and data source's performance. We recommend that you check the status of the dataflow frequently.

To navigate to the action bar, select the three dots “...” next to your dataflow.

Items created for Dataflows for Teams			
Name	Last Refresh	Status	
Dataflow of Miquella	5/19/2021, 4:20:00 PM	Published	...
New dataflow 1	5/19/2021, 11:48:32 AM	Published	
New dataflow 2	N/A	Published	
New dataflow 3	5/19/2021, 4:18:39 PM	Published	

New

Edit
Rename
Refresh
Show refresh history
Delete

Here you can:

- **Edit** your dataflow if you want to change your transformation logic or mapping.
- **Rename** your dataflow. At creation, an autogenerated name is assigned.
- **Refresh** your dataflow. When you refresh your dataflows, the data will be updated.
- **Delete** your dataflow.
- **Show refresh history**. This gives you the results from the last refresh.

Select **Show refresh history** to see information about the last refresh of your dataflow. When the dataflow refresh is successful, you can see how many rows were added or updated in Dataverse. When your dataflow refresh wasn't successful, you can investigate why with the help of the error message.

Table	Status	Successful rows	Failed rows	Error message
Table	✓ Succeeded	4	0	

Dataflows in Teams is a lightweight version

Dataflows in Teams are a lightweight version of dataflows in the maker portal and can only load data into

Dataverse for Teams. Dataflows in Teams are optimized for a one-time import of data, but you can refresh your data manually through the refresh button in the dataflow management page. If you want full dataflows functionality, you can [upgrade your environment](#).

Supported data sources in dataflows in Teams are:

- Excel (OneDrive)
- Text/CSV (OneDrive)
- PDF (OneDrive)
- SharePoint Online Folder
- SharePoint Online list
- XML (OneDrive)
- JSON (OneDrive)
- OData
- Web API

NOTE

Dataflows in Teams don't support non-premises data sources, such as on premises file locations.

The following table lists the major feature differences between dataflows for Dataverse in Teams and dataflows for Dataverse.

DATAFLOW CAPABILITY	DATaverse for Teams	DATaverse
Standard dataflows	Yes	Yes
Analytical dataflows	No	Yes
Gateway support	No	Yes
Manual refresh	Yes	Yes
Scheduled refresh	No	Yes
Incremental refresh	No	Yes
Standard tables	No	Yes
Custom tables	Yes	Yes
Full PQ functionality	Yes	Yes
Supported connectors	Subset of connectors	All connectors
Small data volumes	Yes	Yes
Larger data volumes	No ¹	Yes

¹ Although there's no limitation on the amount of data you can load into Dataverse for Teams, for better performance in loading larger amounts of data, we recommend a Dataverse environment.

Overview of solution-aware dataflows

1/15/2022 • 3 minutes to read • [Edit Online](#)

When you include your dataflows in a solution, their definitions become portable, making it easier to move them from one environment to another, saving time required to author the dataflow.

A typical use case is for an independent software vendor (ISV) to develop a solution containing a dataflow, that extracts and transforms data from a data source to Dataverse tables, in a sandbox environment. The ISV would then move that dataflow and destination tables to a test environment to test with their test data source to validate that the solution works well and is ready for production. After testing completes, the ISV would provide the dataflow and tables to clients who will import them into their production environment to operate on client's data. This process is much easier when you add both the dataflows and tables they load data to into solutions, and then move the solutions and their contents between environments.

Dataflows added to a solution are known as *solution-aware* dataflows. You can add multiple dataflows to a single solution.

NOTE

- Only dataflows created in Power Platform environments can be solution-aware.
- The data loaded by dataflows to their destination isn't portable as part of solutions, only the dataflow definitions are. To recreate the data after a dataflow was deployed as part of a solution, you need to refresh the dataflow.

Add an existing dataflow to a solution

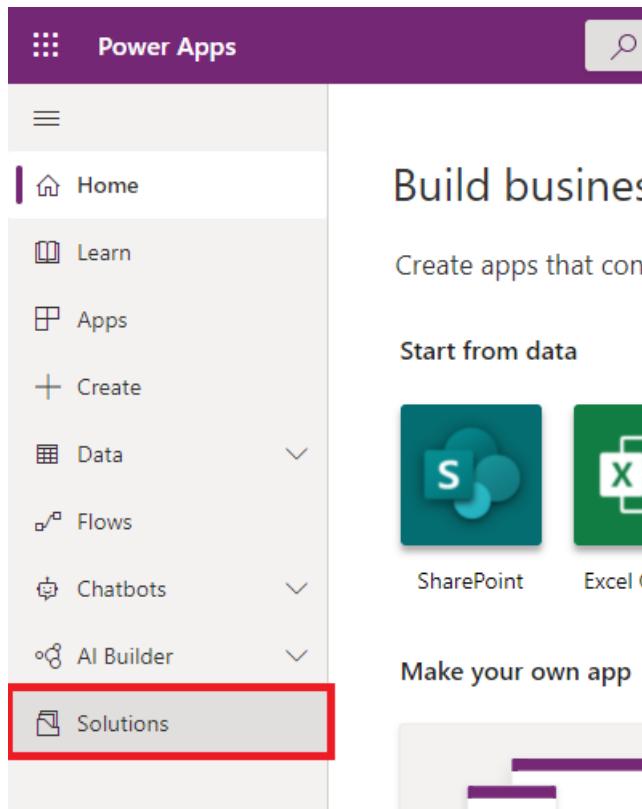
Follow these steps to add a dataflow to a solution.

Prerequisites

- You need to have created a solution before you can add a dataflow to it. More information: [Create solutions](#)
- You need to be the owner of at least one dataflow in the environment. More information: [Create dataflows](#)

Add the dataflow

1. Sign in to [Power Apps](#).
2. Select **Solutions** from the navigation bar.



3. Select the solution you'll add your dataflow to, and from the context menu select Edit.

This screenshot shows the 'Solutions' list in the Power Apps environment. The 'Contoso Solution' is selected, and its context menu is open, with the 'Edit' option highlighted with a red box. Other options in the menu include Delete, Export, Solution checker, Show dependencies, See history, Clone, Apply upgrade, Translations, and Settings. The table lists solutions by display name, name, created date, version, managed externally status, and solution check status.

4. Select Add Existing > Automation > Dataflow.

The screenshot shows the Microsoft Power Apps portal interface. On the left, there's a sidebar with 'Contoso Solution' selected. The main area has a search bar at the top. Below it, a navigation menu shows 'All (0)' selected. In the center, there's a list of objects: AI Model, App, Automation, Chatbot, Dashboard, Report, Security, Table, and More. Under 'Automation', 'Dataflow' is highlighted with a red box and circled. A message at the bottom says 'We didn't find anything to show here'.

5. Optional: If your dataflow loads data into a custom Dataverse table, add the custom table to the solution as well.

In this example, the dataflow you added to the solution loads data into a custom table called **Full Order Details**, which you want to also include in the solution with the dataflow.

Once both the dataflow and table it loads data to are added to the solution, it has the two artifacts added to the solution. In this case, the artifacts are **cr0c8_FullOrderDetails** and **Import Sales Data**.

The screenshot shows the Microsoft Power Apps portal interface. On the left, there's a sidebar with 'Contoso Solution' selected. The main area has a search bar at the top. Below it, a navigation menu shows 'All (2)' selected. In the center, there's a list of objects: Display name ↑, Name ↓, Type ↓, Ma... ↓, Last Modif... ↓, Owner ↓, Sta... ↓. The list contains two items: 'cr0c8_FullOrderDetails' (Table) and 'Import Sales Data' (Dataflow). A red box highlights the 'Tables' item in the navigation menu on the left.

To save your work, be sure to publish all customizations. Now, the solution is ready for you to export from the source environment and import to the destination environment.

Exporting and importing solutions containing dataflows

Exporting and importing solutions containing dataflows is identical to doing the same operations for other artifacts. For the most up-to-date instructions, go to the documentation on [exporting](#) and [importing](#) solutions.

Updating a dataflow's connections after solution import

For security reasons, credentials of connections used by dataflows aren't persisted by solutions. Once a dataflow is deployed as part of a solution, you'll need to edit its connections before it can be scheduled to run.

1. On the left navigation pane, select the down arrow next to **Dataverse** and select **Dataflows**. Identify the dataflow that was imported, and select **Edit** from the context menu.

The screenshot shows the Power Apps portal interface. The left sidebar is open, showing categories like Home, Learn, Apps, Create, Data (Tables, Choices), Dataflows (selected), Azure Synapse Link, Connections, Custom Connectors, Gateways, Flows, Chatbots, AI Builder, and Solutions. The main area displays a list of Dataflows under 'My Dataflows'. One dataflow, 'Import Sales Data', is selected. A context menu is open over this dataflow, with the 'Edit' option highlighted by a red box. Other options in the menu include Rename, Refresh, Edit refresh settings, Show refresh history, Delete, Edit incremental refresh, and Create copy.

2. In the Dataflow list, locate and double-click the dataflow that was added as part of the solution you've imported.
3. You'll be asked to enter credentials required for the dataflow.

The screenshot shows the Power Query - Edit queries interface. The left sidebar is identical to the previous screenshot. The main area is a query editor titled 'Power Query - Edit queries'. It shows a list of queries: Customers, Order_Details, Orders, Merge Customers and Or..., and Full Order Details. A warning message at the top states: 'Credentials are required to connect to the OData source. (Source at https://services.odata.org/V4/Northwind/Northwind.svc.)' with a 'Configure connection' button. The 'Transform' ribbon tab is selected. On the right side, there are sections for 'Query settings' (Properties, Name: Customers) and 'Applied steps' (Source, Navigation 1, Removed col...). At the bottom, there are buttons for Step, Next, and a progress bar indicating 'Completed (1.72 s)'.

Once the credentials for the connection have been updated, all queries that use that connection automatically load.

4. If your dataflow loads data in Dataverse tables, select **Next** to review the mapping configuration.

The screenshot shows the 'Power Query - Edit queries' interface. On the left, there's a navigation pane with options like Home, Learn, Apps, Create, Data, Tables, Choices, Dataflows, Azure Synapse Link, Connections, Custom Connectors, Gateways, Flows, Chatbots, AI Builder, and Solutions. The main area displays a table titled 'Table.RemoveColumns(#"Navigation 1", {"Orders", "CustomerDemographics"})'. The table contains 22 rows of data from the Northwind database. The 'Applied steps' section on the right shows a step named 'Navigation 1' which removed columns 'Orders' and 'CustomerDemographics'. A red box highlights the 'Next' button at the bottom right of the interface.

5. The mapping configuration is also saved as part of the solution. Since you also added the destination table to the solutions, there's no need to recreate the table in this environment and you can publish the dataflow.

The screenshot shows the 'Power Query - Map tables' interface. It displays a 'Queries' list on the left with items: Customers, Order_Details, Orders, Merge Customers and Orders, and Full Order Details. The 'Load settings' section shows 'Table name' as 'cr0c8_FullOrderDetails'. The 'Column mapping' section lists source columns (Address, City, CompanyName, ContactName, ContactTitle, Country, CustomerID, EmployeeID, Fax, Freight, (none), OrderDate) and their corresponding destination columns (cr0c8_Address, cr0c8_City, cr0c8_CompanyName, cr0c8_ContactName, cr0c8_ContactTitle, cr0c8_Country, cr0c8_CustomerID, cr0c8_EmployeeID, cr0c8_Fax, cr0c8_Freight, cr0c8_Name, cr0c8_OrderDate). A red box highlights the 'Publish' button at the bottom right of the interface.

That's it. Your dataflow now refreshes and loads data to the destination table.

Known limitations

- Dataflows can't be created from within solutions. To add a dataflow to a solution, follow the steps outlined in this article.
- Dataflows can't be edited directly from within solutions. Instead, the dataflow must be edited in the dataflows experience.
- Dataflows can't use connection references for any connector.
- Environment variables can't be used by dataflows.
- Dataflows don't support adding required components, such as custom tables they load data to. Instead, the custom table should be manually added to the solution.
- Dataflows can't be deployed by application users (service principals).

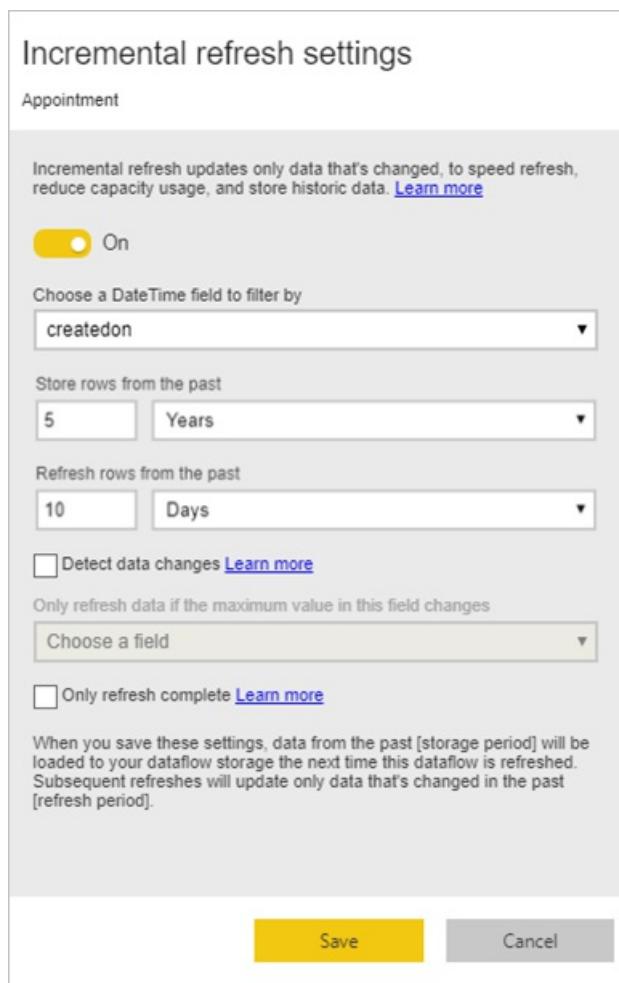
Using incremental refresh with dataflows

1/15/2022 • 8 minutes to read • [Edit Online](#)

With dataflows, you can bring large amounts of data into Power BI or your organization's provided storage. In some cases, however, it's not practical to update a full copy of source data in each refresh. A good alternative is **incremental refresh**, which provides the following benefits for dataflows:

- **Refresh occurs faster:** Only data that's changed needs to be refreshed. For example, refresh only the last five days of a 10-year dataflow.
- **Refresh is more reliable:** For example, it's not necessary to maintain long-running connections to volatile source systems.
- **Resource consumption is reduced:** Less data to refresh reduces overall consumption of memory and other resources.

Incremental refresh is available in dataflows created in Power BI and dataflows created in Power Apps. This article shows screens from Power BI, but these instructions apply to dataflows created in Power BI or in Power Apps.



Using incremental refresh in dataflows created in Power BI requires that the dataflow reside in a workspace in **Premium capacity**. Incremental refresh in Power Apps requires Power Apps Plan 2.

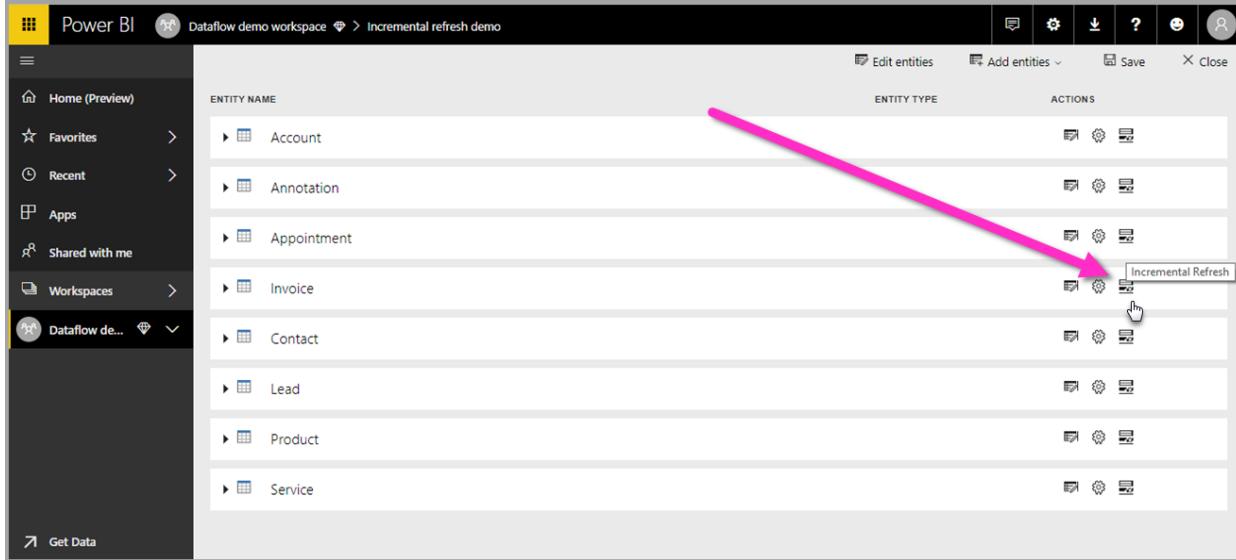
In either Power BI or Power Apps, using incremental refresh requires that source data ingested into the dataflow have a DateTime field on which incremental refresh can filter.

Configuring incremental refresh for dataflows

A dataflow can contain many entities. Incremental refresh is set up at the entity level, allowing one dataflow to hold both fully refreshed entities and incrementally refreshed entities.

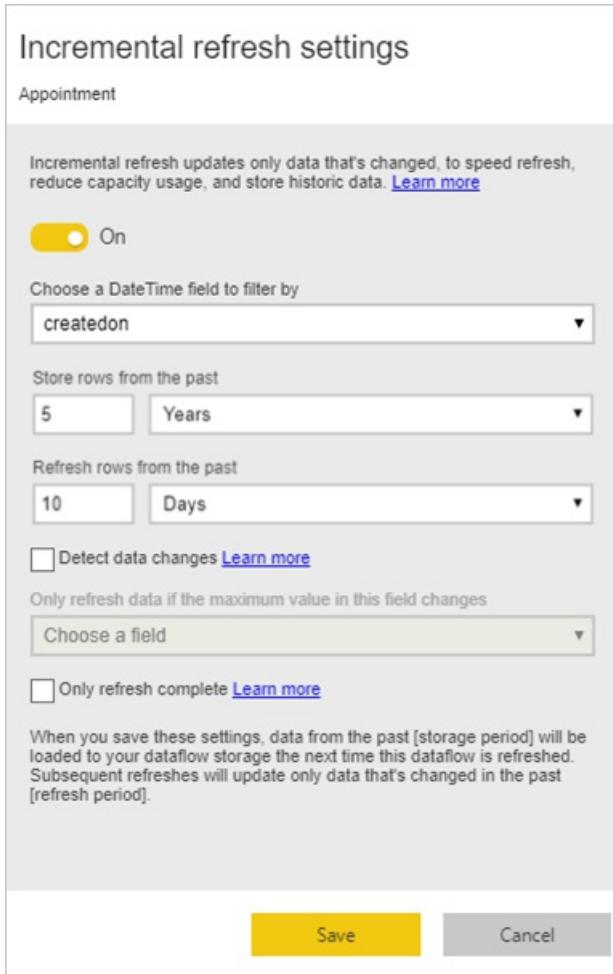
To set up an incremental refreshed entity, start by configuring your entity as you would any other entity.

After the dataflow is created and saved, select **Incremental refresh**  in the entity view, as shown in the following image.



The screenshot shows the Power BI Entity view. On the left is a navigation sidebar with Home (Preview), Favorites, Recent, Apps, Shared with me, Workspaces, and Dataflow demo workspace. The main area lists entities: Account, Annotation, Appointment, Invoice, Contact, Lead, Product, and Service. Each entity has columns for Entity Name, Entity Type, and Actions. The 'Actions' column for the Contact entity contains a gear icon, which is highlighted with a pink arrow pointing from the text above.

When you select the icon, the **Incremental refresh settings** window appears. Turn on incremental refresh.



Incremental refresh settings

Appointment

Incremental refresh updates only data that's changed, to speed refresh, reduce capacity usage, and store historic data. [Learn more](#)

On

Choose a DateTime field to filter by

Store rows from the past
 Years

Refresh rows from the past
 Days

Detect data changes [Learn more](#)

Only refresh data if the maximum value in this field changes

Only refresh complete [Learn more](#)

When you save these settings, data from the past [storage period] will be loaded to your dataflow storage the next time this dataflow is refreshed. Subsequent refreshes will update only data that's changed in the past [refresh period].

Save **Cancel**

The following list explains the settings in the **Incremental refresh settings** window.

- **Incremental refresh on/off toggle:** Turns the incremental refresh policy on or off for the entity.
- **Filter field drop-down:** Selects the query field on which the entity should be filtered for increments. This field only contains DateTime fields. You can't use incremental refresh if your entity doesn't contain a DateTime field.
- **Store/refresh rows from the past:** The example in the previous image illustrates these next few settings.

In this example, we define a refresh policy to store five years of data in total and incrementally refresh 10 days of data. Assuming that the entity is refreshed daily, the following actions are carried out for each refresh operation:

- Add a new day of data.
- Refresh 10 days, up to the current date.
- Remove calendar years that are older than five years before the current date. For example, if the current date is January 1, 2019, the year 2013 is removed.

The first dataflow refresh might take a while to import all five years, but subsequent refreshes are likely to be completed much more quickly.

- **Detect data changes:** An incremental refresh of 10 days is much more efficient than a full refresh of five years, but you might be able to do even better. When you select the **Detect data changes** check box, you can select a date/time column to identify and refresh only the days where the data has changed. This assumes such a column exists in the source system, which is typically for auditing purposes. The maximum value of this column is evaluated for each of the periods in the incremental range. If that data hasn't changed since the last refresh, there's no need to refresh the period. In the example, this might further reduce the days incrementally refreshed from 10 to perhaps 2.

TIP

The current design requires that the column used to detect data changes be persisted and cached into memory. You might want to consider one of the following techniques to reduce cardinality and memory consumption:

- Persist only the maximum value of this column at time of refresh, perhaps by using a Power Query function.
- Reduce the precision to a level that's acceptable given your refresh-frequency requirements.

- **Only refresh complete periods:** Imagine that your refresh is scheduled to run at 4:00 AM every day. If data appears in the source system during those first four hours of that day, you might not want to account for it. Some business metrics, such as barrels per day in the oil and gas industry, aren't practical or sensible to account for based on partial days.

Another example where only refreshing complete periods is appropriate is refreshing data from a financial system. Imagine a financial system where data for the previous month is approved on the 12th calendar day of the month. You can set the incremental range to one month and schedule the refresh to run on the 12th day of the month. With this option selected, the system will refresh January data (the most recent complete monthly period) on February 12.

NOTE

Dataflow incremental refresh determines dates according to the following logic: if a refresh is scheduled, incremental refresh for dataflows uses the time zone defined in the refresh policy. If no schedule for refreshing exists, incremental refresh uses the time from the computer running the refresh.

After incremental refresh is configured, the dataflow automatically alters your query to include filtering by date. You can edit the automatically generated query by using the advanced editor in Power Query to fine-tune or customize your refresh. Read more about incremental refresh and how it works in the following sections.

Incremental refresh and linked entities vs. computed entities

For *linked* entities, incremental refresh updates the source entity. Because linked entities are simply a pointer to the original entity, incremental refresh has no impact on the linked entity. When the source entity is refreshed according to its defined refresh policy, any linked entity should assume the data in the source is refreshed.

Computed entities are based on queries running over a data store, which can be another dataflow. As such, computed entities behave the same way as linked entities.

Because computed entities and linked entities behave similarly, the requirements and configuration steps are the same for both. One difference is that for computed entities, in certain configurations, incremental refresh can't run in an optimized fashion because of the way partitions are built.

Changing between incremental and full refresh

Dataflows support changing the refresh policy between incremental and full refresh. When a change occurs in either direction (full to incremental or incremental to full), the change affects the dataflow after the next refresh.

When moving a dataflow from full refresh to incremental, the new refresh logic updates the dataflow by adhering to the refresh window and increment as defined in the incremental refresh settings.

When moving a dataflow from incremental to full refresh, all data accumulated in the incremental refresh is overwritten by the policy defined in the full refresh. You must approve this action.

Time zone support in incremental refresh

Dataflow incremental refresh is dependent on the time at which it's run. The filtering of the query is dependent on the day on which it's run.

To accommodate those dependencies and to ensure data consistency, incremental refresh for dataflows implements the following heuristic for *refresh now* scenarios:

- In the case where a scheduled refresh is defined in the system, incremental refresh uses the time-zone settings from the scheduled refresh section. This ensures that whatever time zone the person refreshing the dataflow is in, it will always be consistent with the system's definition.
- If no scheduled refresh is defined, dataflows use the time zone from the computer of the user who's performing the refresh.

Incremental refresh can also be invoked by using APIs. In this case, the API call can hold a time-zone setting that's used in the refresh. Using APIs can be helpful for testing and validation purposes.

Incremental refresh implementation details

Dataflows use partitioning for incremental refresh. After XMLA-endpoints for Power BI Premium are available, the partitions become visible. Incremental refresh in dataflows keeps the minimum number of partitions to meet refresh policy requirements. Old partitions that go out of range are dropped, which maintains a rolling window. Partitions are opportunistically merged, reducing the total number of partitions required. This improves compression and, in some cases, can improve query performance.

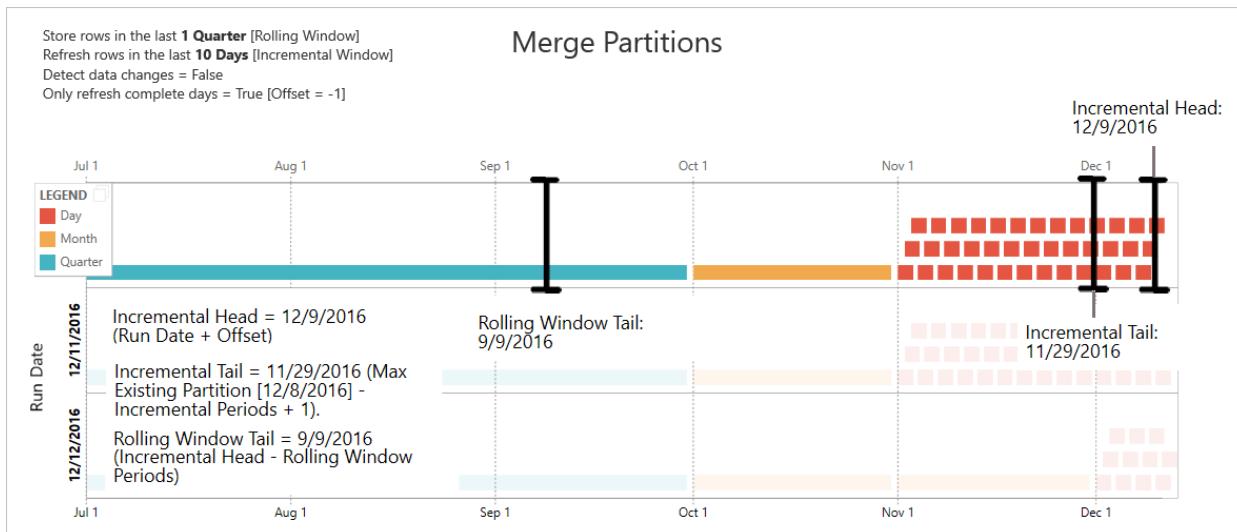
The examples in this section share the following refresh policy:

- Store rows in the last 1 Quarter

- Refresh rows in the last 10 Days
- Detect data changes = False
- Only refresh complete days = True

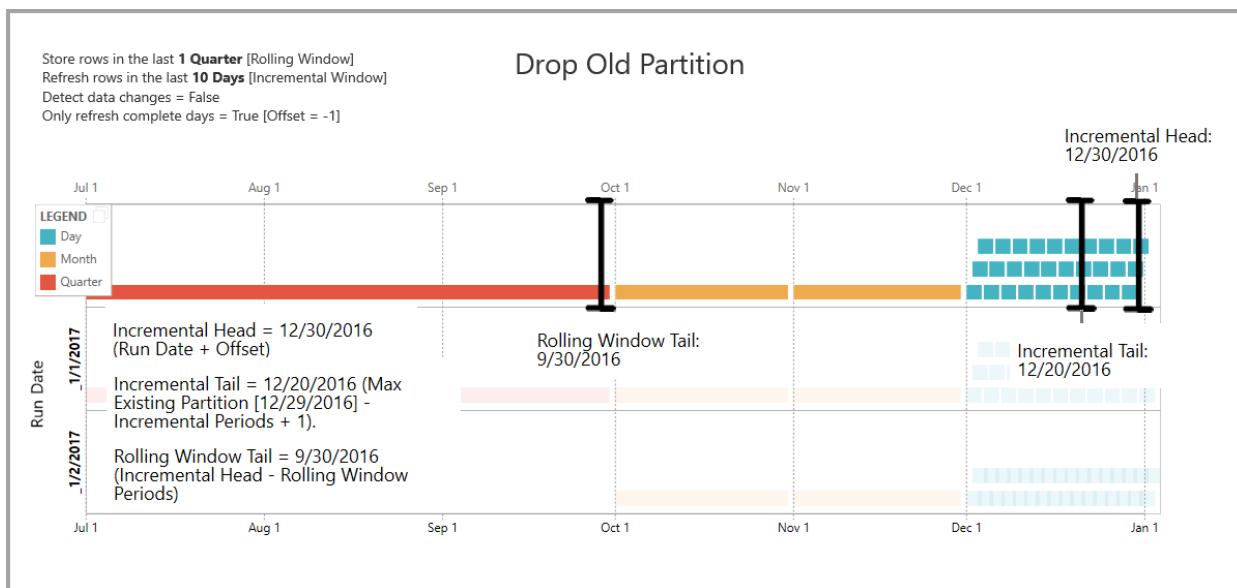
Merge partitions

In this example, day partitions are automatically merged to the month level after they go outside the incremental range. Partitions in the incremental range need to be maintained at daily granularity to allow only those days to be refreshed. The refresh operation with *Run Date 12/11/2016* merges the days in November, because they fall outside the incremental range.



Drop old partitions

Old partitions that fall outside the total range are removed. The refresh operation with *Run Date 1/2/2017* drops the partition for Q3 of 2016 because it falls outside the total range.



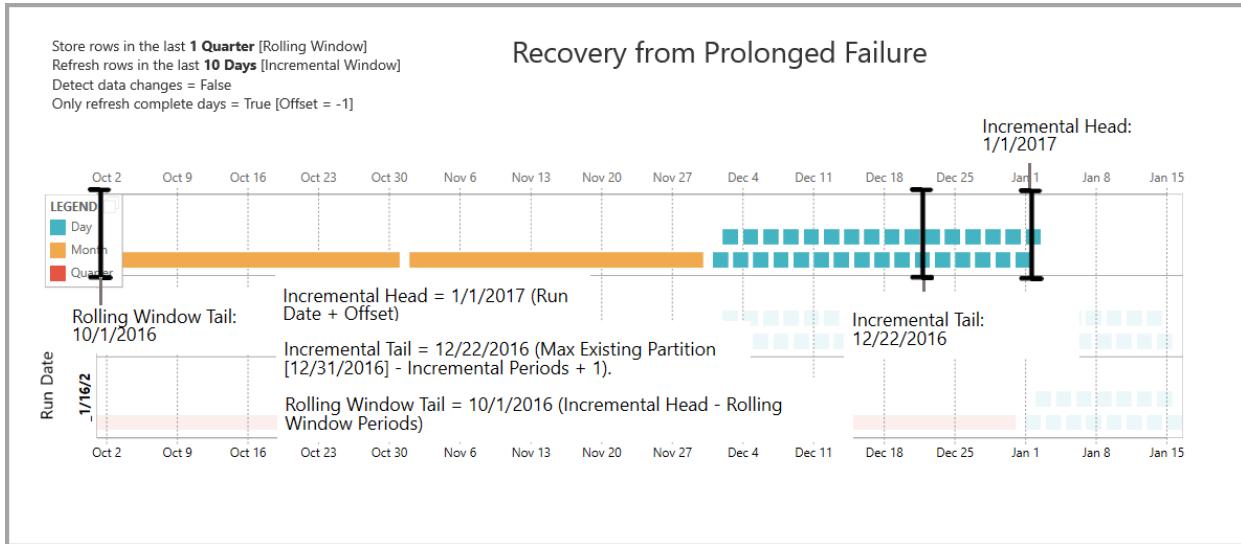
Recovery from prolonged failure

This example simulates how the system recovers gracefully from prolonged failure. Let's say refresh doesn't run successfully because data source credentials expired, and the issue takes 13 days to resolve. The incremental range is only 10 days.

The next successful refresh operation, with *Run Date 1/15/2017*, needs to backfill the missing 13 days and refresh them. It also needs to refresh the previous nine days because they weren't refreshed on the normal schedule. In other words, the incremental range is increased from 10 to 22 days.

The next refresh operation, with *Run Date 1/16/2017*, takes the opportunity to merge the days in December and

the months in Q4 of 2016.



Dataflow incremental refresh and datasets

Dataflow incremental refresh and dataset incremental refresh are designed to work in tandem. It's acceptable and supported to have an incrementally refreshing entity in a dataflow, fully loaded into a dataset, or a fully loaded entity in a dataflow incrementally loaded to a dataset.

Both approaches work according to your specified definitions in the refresh settings. More information: [Incremental refresh in Power BI Premium](#)

Considerations and limitations

Incremental refresh in Microsoft Power Platform dataflows is only supported in dataflows with an Azure Data Lake Storage account, not in dataflows with Dataverse as the destination.

See also

This article described incremental refresh for dataflows. Here are some more articles that might be useful:

- [Self-service data prep in Power BI](#)
- [Creating computed entities in dataflows](#)
- [Connect to data sources for dataflows](#)
- [Link entities between dataflows](#)
- [Create and use dataflows in Power BI](#)
- [Using dataflows with on-premises data sources](#)
- [Developer resources for Power BI dataflows](#)

For more information about Power Query and scheduled refresh, you can read these articles:

- [Query overview in Power BI Desktop](#)
- [Configuring scheduled refresh](#)

For more information about Common Data Model, you can read its overview article:

- [Common Data Model - overview](#)

Connect to data sources for dataflows

1/15/2022 • 3 minutes to read • [Edit Online](#)

With Microsoft Power BI and Power Platform dataflows, you can connect to many different data sources to create new dataflows, or add new entities to an existing dataflow.

This article describes how to create dataflows by using these data sources. For an overview of how to create and use dataflows, go to [Creating a dataflow](#) for Power BI service and [Create and use dataflows in Power Apps](#).

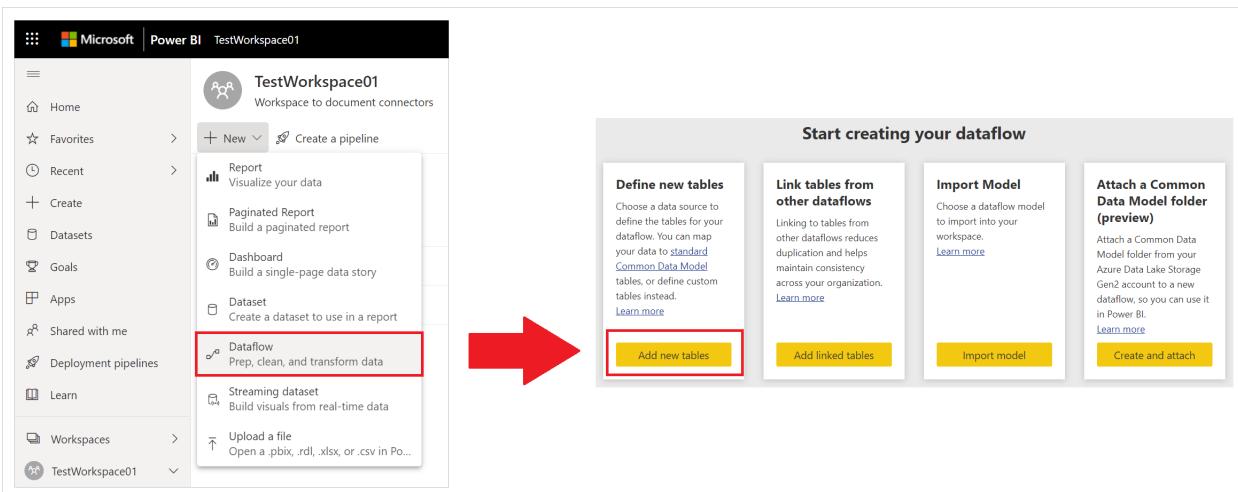
Create a dataflow from a data source

To create a dataflow from a data source, you'll first have to connect to your data.

- [Power BI service](#)
- [Power Apps](#)

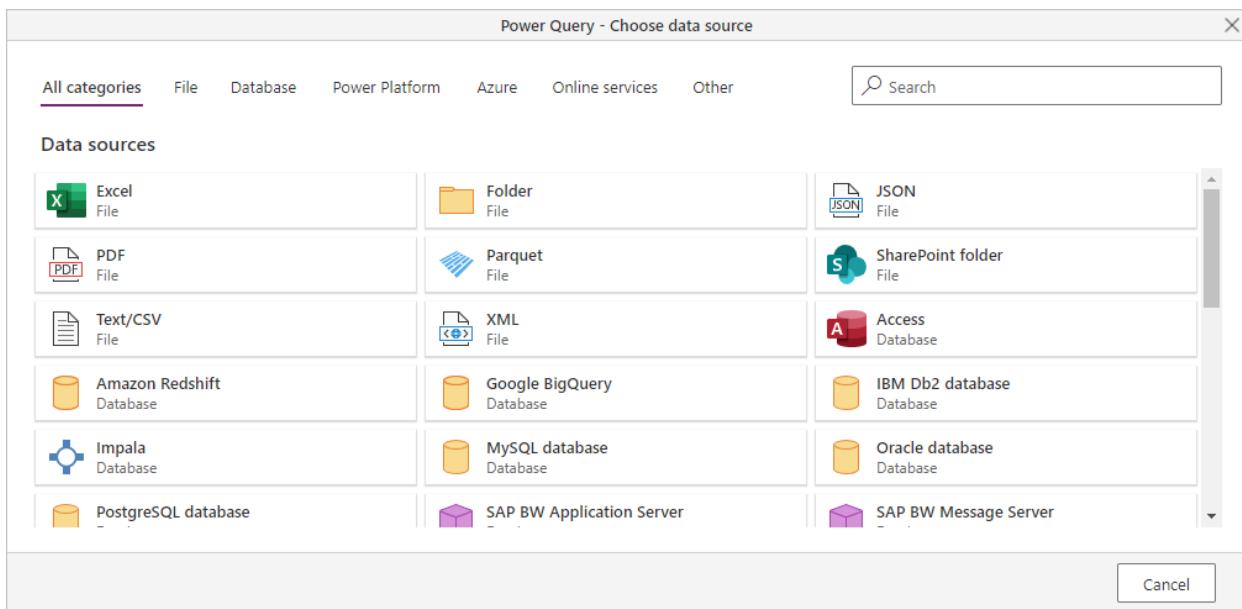
To connect to data in Power BI:

1. Open a workspace.
2. Select **New**.
3. Select **Dataflow** from the drop-down menu.
4. Under **Define new tables**, select **Add new tables**.



Data sources for dataflows

Once you've created the dataflow from the dataflow authoring tool, you'll be presented with the **Choose data source** dialog box.



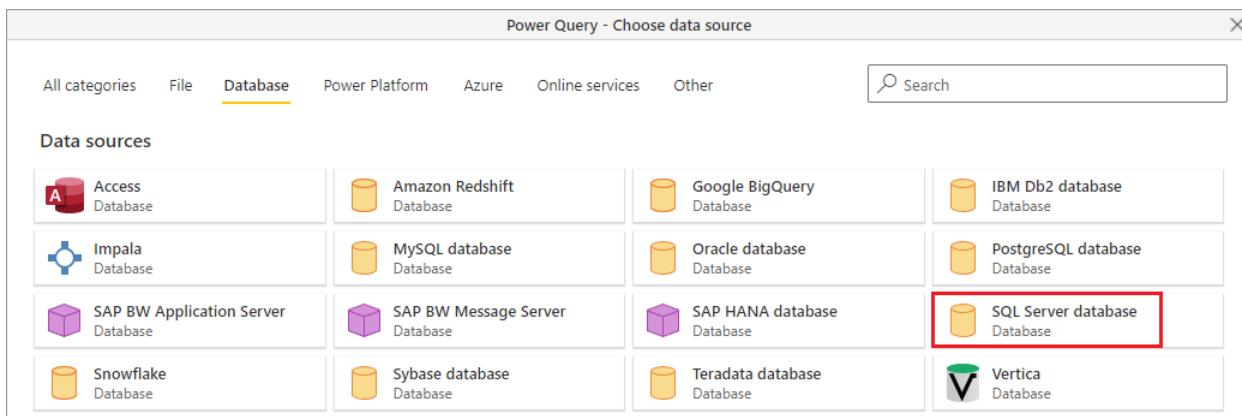
Data sources for dataflows are organized into the following categories, which appear as tabs in the **Choose data source** dialog box:

- All categories
- File
- Database
- Power Platform
- Azure
- Online Services
- Other

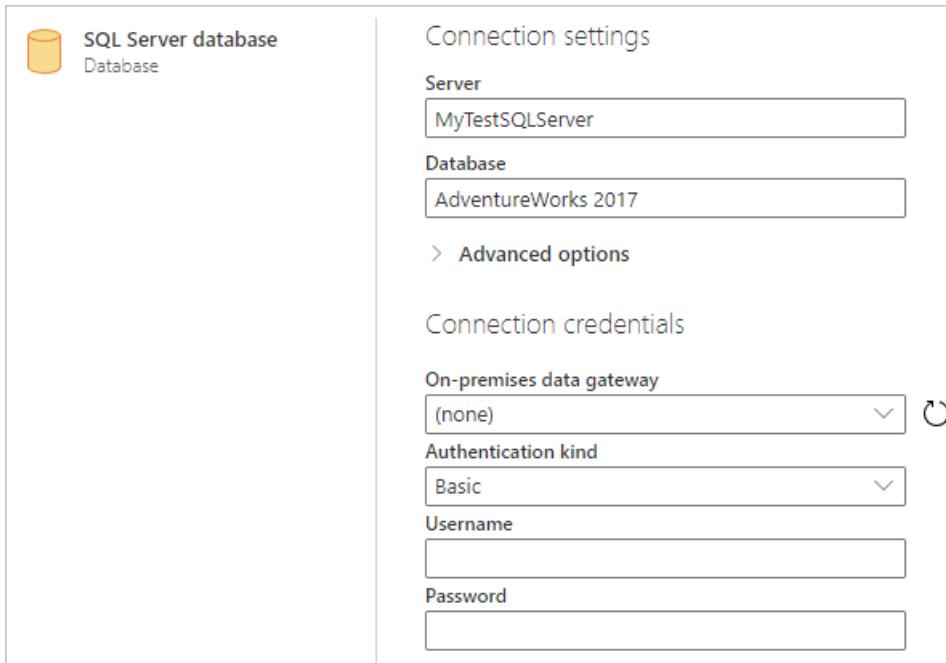
For a list of all of the supported data sources in Power Query, see [Connectors in Power Query](#).

Connect to a data source

To connect to a data source, select the data source. This section uses one example to show how the process works, but each data connection for dataflows is similar in process. Different connectors might require specific credentials or other information, but the flow is similar. In this example, **SQL Server database** is selected from the **Database** data connection category.



A connection window for the selected data connection is displayed. If credentials are required, you're prompted to provide them. The following image shows a server and database being entered to connect to a SQL Server database.



After the server URL or resource connection information is provided, enter the credentials to use for access to the data. You may also need to enter the name of an on-premises data gateway. Then select **Next**.

Power Query Online initiates and establishes the connection to the data source. It then presents the available tables from that data source in the **Navigator** window.

12	BusinessEntityID	ab Title	ab FirstName	ab MiddleName	ab LastName	ab Suffix	ab
	233	null	Magnus	E	Hedlund	nu	u
	234	null	Laura	F	Norman	nu	u
	234	null	Laura	F	Norman	nu	u
	235	null	Paula	M	Barreto de Mattos	nu	u
	236	null	Grant	N	Culbertson	nu	u
	237	null	Hao	O	Chen	nu	u
	238	null	Vidur	X	Luthra	nu	u
	239	null	Mindy	C	Martin	nu	u
	240	null	Willis	T	Johnson	nu	u
	241	null	David	J	Liu	nu	u
	242	null	Deborah	E	Poe	nu	u
	243	null	Candy	L	Spoon	nu	u
	244	null	Bryan	A	Walton	nu	u
	245	null	Barbara	C	Moreland	nu	u
	246	null	Dragan	K	Tomic	nu	u
	247	null	Janet	L	Sheperdigan	nu	u
	248	null	Mike	K	Seamans	nu	u

You can select tables and data to load by selecting the check box next to each in the left pane. To transform the data you've chosen, select **Transform data** from the bottom of the **Navigator** window. A Power Query Online dialog box appears, where you can edit queries and perform any other transformations you want to the selected data.

The screenshot shows the Power Query - Edit queries interface. The top ribbon has tabs for Home, Transform, Add column, and View. The Home tab is selected. The ribbon also includes Get data, Options, Manage parameters, Refresh, Properties, Advanced editor, Manage columns, Choose columns, Remove columns, Keep rows, Remove rows, Reduce rows, Sort, Split column, Group by, ABC 123 Data type: Whole number, Use first row as headers, and Replace values. The main area displays a table titled "HumanResources vEmplo...". The table has 11 columns: BusinessEntityID, ab Title, ab FirstName, ab MiddleName, ab LastName, ab Suffix, and ab Shift. Rows 1 through 14 are shown, with rows 15 and 16 partially visible. The bottom status bar indicates "Columns: 11 Rows: 99+ Completed (4.32 s)". On the right, the "Query settings" pane shows the name "HumanResources vEmplo..." and the applied step "Source". A "Navigation" step is also listed.

Connecting to additional data sources

There are additional data connectors that aren't shown in the Power BI dataflows user interface, but are supported with a few additional steps.

You can take the following steps to create a connection to a connector that isn't displayed in the user interface:

1. Open Power BI Desktop, and then select **Get data**.
2. Open Power Query Editor in Power BI Desktop, right-click the relevant query, and then select **Advanced Editor**, as shown in the following image. From there, you can copy the M script that appears in the **Advanced Editor** window.

The screenshot shows the Power Query Advanced Editor window. The title bar says "pbist_twitter hashtag_slicer". The main area contains the following M script:

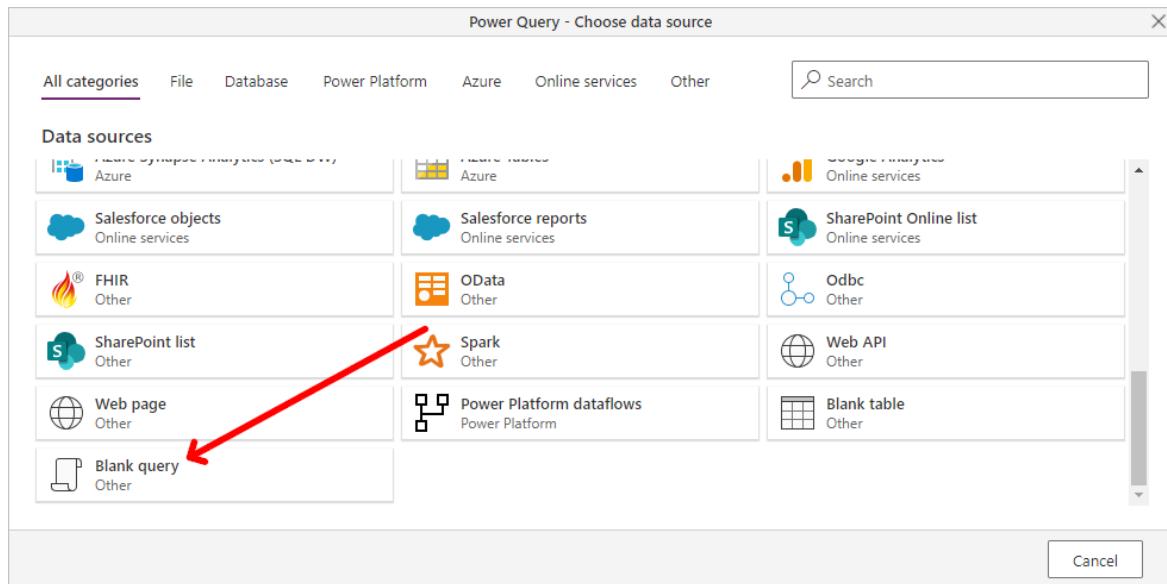
```

let
    Source = Sql.Database("yaronctestingdb1.database.windows.net", "twitterDB"),
    pbist_twitter_hashtag_slicer = Source{[Schema="pbist_twitter",Item="hashtag_slicer"]}[Data]
in
    pbist_twitter_hashtag_slicer

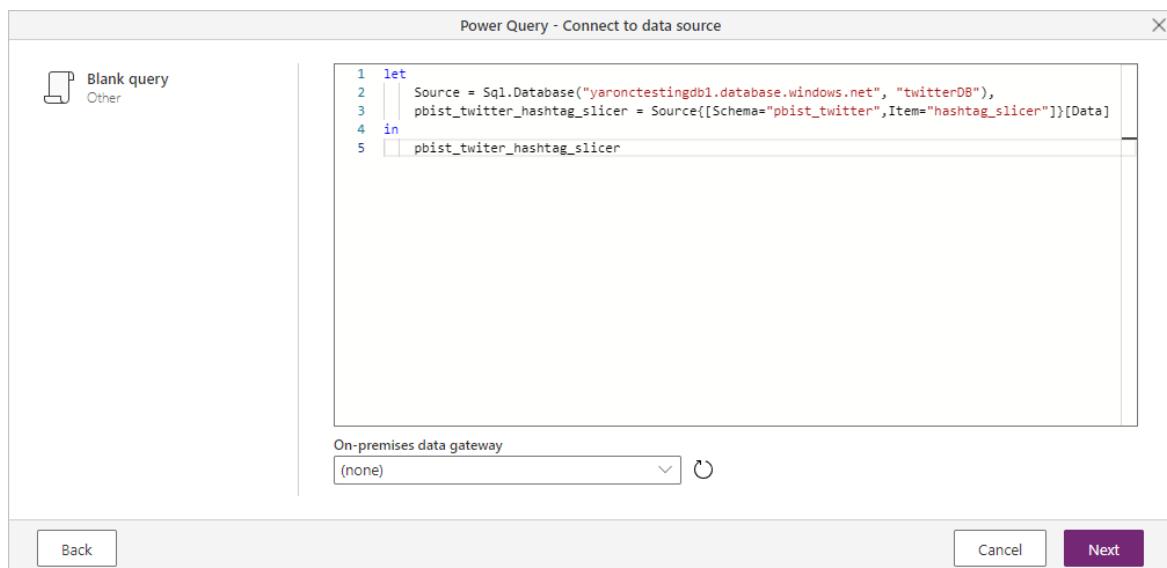
```

A message at the bottom left says "No syntax errors have been detected." At the bottom right are "Done" and "Cancel" buttons.

3. Open the Power BI dataflow, and then select **Get data** for a blank query.



4. Paste the copied query into the blank query for the dataflow.



Your script then connects to the data source you specified.

The following list shows which connectors you can currently use by copying and pasting the M query into a blank query:

- SAP Business Warehouse
- Azure Analysis Services
- Adobe Analytics
- ODBC
- OLE DB
- Folder
- SharePoint Online folder
- SharePoint folder
- Hadoop HDFS
- Azure HDInsight (HDFS)
- Hadoop file HDFS
- Informix (beta)

Next steps

This article showed which data sources you can connect to for dataflows. The following articles go into more detail about common usage scenarios for dataflows:

- [Self-service data prep in Power BI](#)
- [Using incremental refresh with dataflows](#)
- [Creating computed entities in dataflows](#)
- [Link entities between dataflows](#)

For information about individual Power Query connectors, go to the [connector reference list of Power Query connectors](#), and select the connector you want to learn more about.

Additional information about dataflows and related information can be found in the following articles:

- [Create and use dataflows in Power BI](#)
- [Using dataflows with on-premises data sources](#)
- [Developer resources for Power BI dataflows](#)
- [Dataflows and Azure Data Lake integration \(Preview\)](#)

For more information about Power Query and scheduled refresh, you can read these articles:

- [Query overview in Power BI Desktop](#)
- [Configuring scheduled refresh](#)

For more information about Common Data Model, you can read its overview article:

- [Common Data Model - overview](#)

What licenses do you need to use dataflows?

1/15/2022 • 4 minutes to read • [Edit Online](#)

Dataflows can be created in different portals, such as Power BI and the Power Apps, and can be of either analytical or standard type. In addition, some dataflow features are only available as Premium features. Considering the wide range of products that can use dataflows, and feature availability in each product or dataflow type, it's important to know what licensing options you need to use dataflows.

Creating dataflows in Power BI workspaces

If you want to create dataflows in Power BI workspaces, you need to have a paid Power BI Pro license. A Power BI free license won't give you the ability to create dataflows. Depending on the features you use, a Power BI Premium or embedded capacity is required.

A Power BI Pro account is available on a user-monthly basis. Multiple options are available for premium or embedded capacities.

Creating dataflows in Microsoft Power Platform environments

If you want to create dataflows in the Microsoft Power Platform environment a Power Apps (per-user or per-app) license is required.

If you want to create analytical dataflows that store data in your organization's Azure Data Lake Storage Gen2 account, you or your administrator will need access to an Azure subscription and an Azure Data Lake Storage Gen2 account.

Premium features

Some of the dataflow features are limited to premium licenses. If you want to use the enhanced compute engine to speed up your dataflow queries' performance over computed entities, or have the DirectQuery connection option to the dataflow, you need to have Power BI P1 or A3 or higher capacities.

AI capabilities in Power BI, linked entity, and computed entity are all premium functions that aren't available with a Power BI Pro account.

Features

The following table contains a list of features and the license needed for them to be available.

FEATURE	POWER BI	POWER APPS
Store data in Dataverse tables (standard dataflow)	N/A	Per app plan Per user plan
Store data in Azure Data Lake Storage (analytical dataflow)	Power BI Pro Power BI Premium	Yes, using analytical dataflows
Store data in customer provided Azure Data Lake Storage (analytical dataflow; bring your own Azure Data Lake Storage)	Power BI Pro Power BI Premium	Per app plan Per user plan

FEATURE	POWER BI	POWER APPS
The enhanced compute engine (running on Power BI Premium capacity / parallel execution of transforms)	Power BI Premium	N/A
DirectQuery connection to dataflow	Power BI Premium	N/A
AI capabilities in Power BI	Power BI Premium	N/A
Linked entities	Power BI Premium	Yes, using analytical dataflows
Computed entities (in-storage transformations using M)	Power BI premium	Yes, using analytical dataflows
Schedule refresh	Yes	Yes
Dataflow authoring with Power Query Online	Yes	Yes
Dataflow management	Power BI	Power Apps
New connectors	Yes	Yes
Standardized schema, built-in support for Common Data Model	Yes	Yes
Dataflows data connector in Power BI Desktop	Yes	Yes, using analytical dataflows
Dataflow incremental refresh	Power BI Premium	Yes, using analytical dataflows with Per user Plan

Limitations on each license

The preceding table shows what features each license will give you. The following sections provide details about some of the limitations of each license.

Power Apps licenses

If you use a Power Apps license to create dataflows, there's no limitation on the number of dataflows and entities you can create. However, there's a limitation on the size of Dataverse service you can use.

The Power Apps per-app plan covers up to a 50-MB database capacity. The Power Apps per-user plan allows you to have a database of 250-MB capacity.

Power BI Pro

Power BI Pro gives you the ability to create analytical dataflows, but not use any of the premium features. With a Power BI Pro account, you can't use linked or computed entities, you can't use AI capabilities in Power BI, and you can't use DirectQuery to connect to the dataflow. The storage for your dataflows is limited to the space left under your Power BI Pro account, which is a subset of 10-GB storage for all Power BI content.

Power BI Premium

If you use Power BI Premium (capacity-based licensing), you can use all the AI capabilities in Power BI, computed entities and linked entities, with the ability to have a DirectQuery connection to the dataflow, and you can use the enhanced compute engine. However, the dataflow created under a premium capacity license uses only the internal Azure Data Lake Storage, and won't be accessible by other platforms except Power BI itself. You can't create external dataflows just by having a Power BI Premium license; you need to have an Azure subscription for Azure Data Lake Storage as well.

Using your organization's Azure Data Lake Storage account for dataflow storage

To create dataflows that store data in your organization's Azure Data Lake Storage account, in addition to the product licenses above, you must have an Azure subscription. The amount of storage that can be used isn't limited by the dataflow or the product it was created in.

Next step

If you want to read more details about the concepts discussed in this article, follow any of the links below.

Pricing

- [Power BI pricing](#)
- [Power Apps pricing](#)
- [Azure Data Lake Storage Gen 2 pricing](#)

Features

- [Computed entities](#)
- [Linked entities](#)
- [AI capabilities in Power BI dataflows](#)
- [Standard vs. analytical dataflows](#)
- [The enhanced compute engine](#)

How to migrate queries from Power Query in the desktop (Power BI and Excel) to dataflows

1/15/2022 • 3 minutes to read • [Edit Online](#)

If you already have queries in Power Query, either in Power BI Desktop or in Excel, you might want to migrate the queries into dataflows. The migration process is simple and straightforward. In this article, you'll learn the steps to do so.

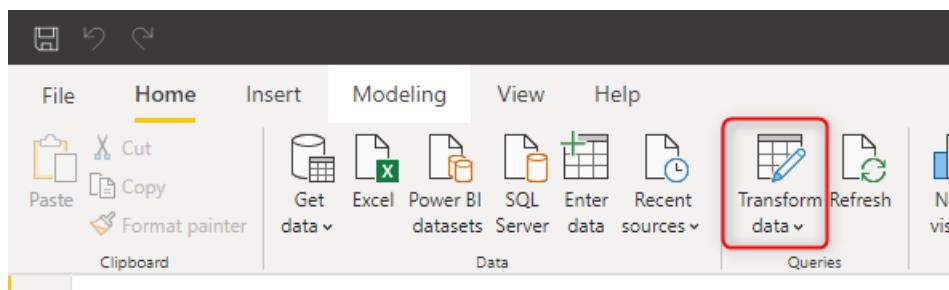
To learn how to create a dataflow in Microsoft Power Platform, go to [Create and use dataflows in Power Platform](#). To learn how to create a dataflow in Power BI, go to [Creating and using dataflows in Power BI](#).

Migrating the queries from the desktop

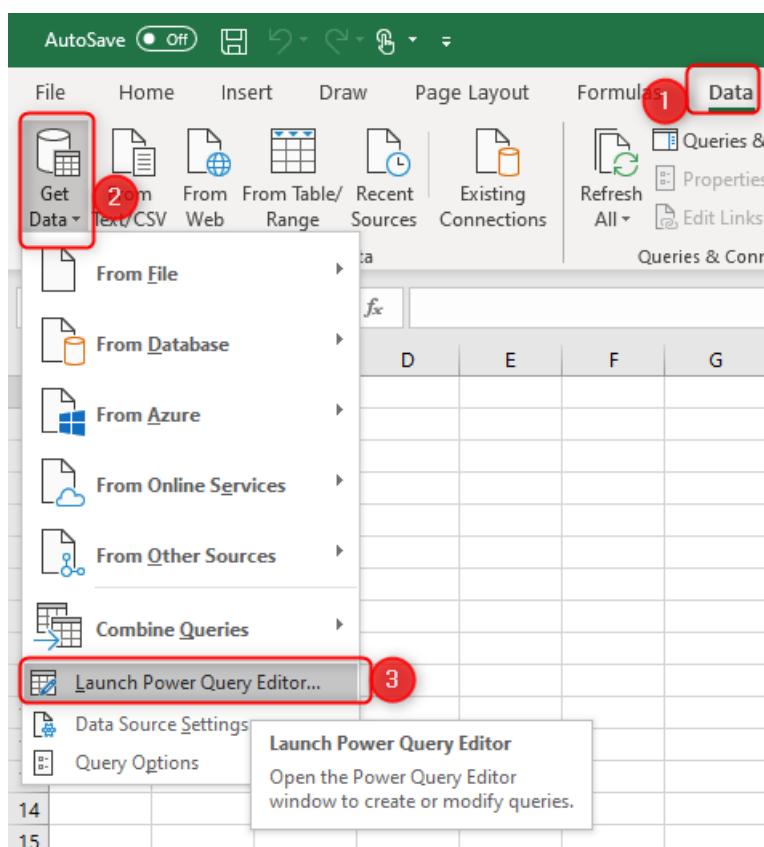
To migrate queries from Power Query in the desktop tools:

1. Open Power Query Editor:

- In Power BI Desktop on the Home tab, select **Transform data**.



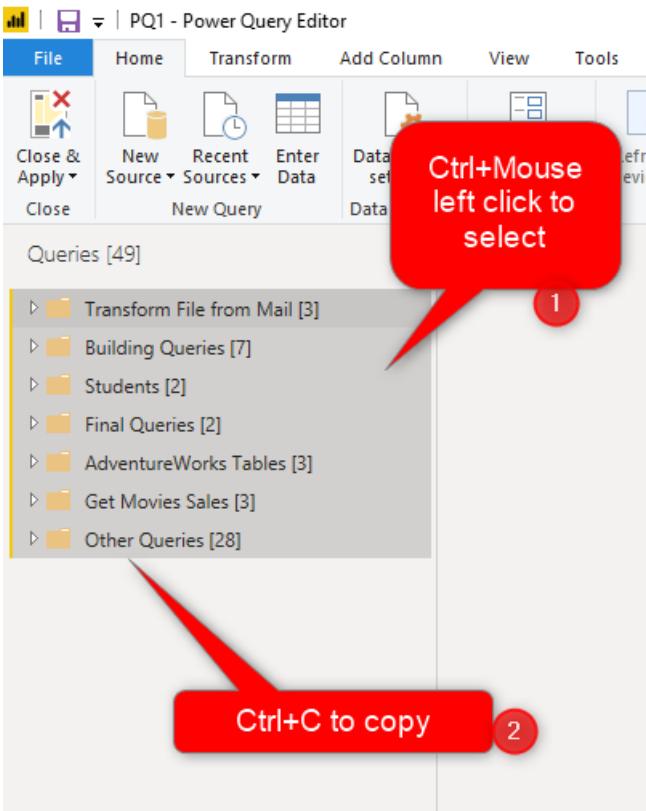
- In Excel on the Data tab, select **Get Data > Launch Power Query Editor**.



2. Copy the queries:

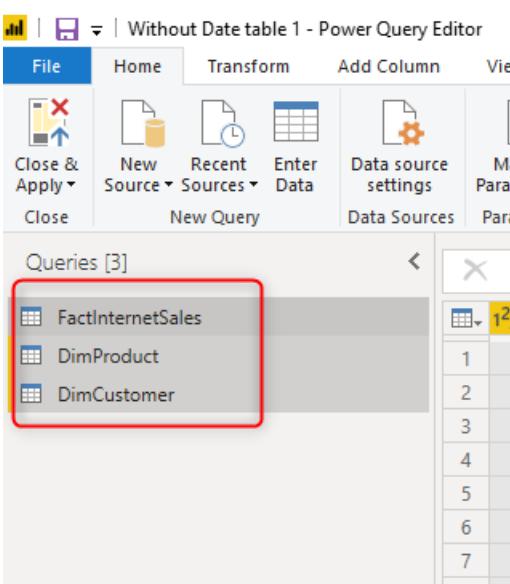
If you've organized your queries into folders (called *groups* in Power Query):

- In the **Queries** pane, select **Ctrl** as you select the folders you want to migrate to the dataflow.
- Select **Ctrl+C**.



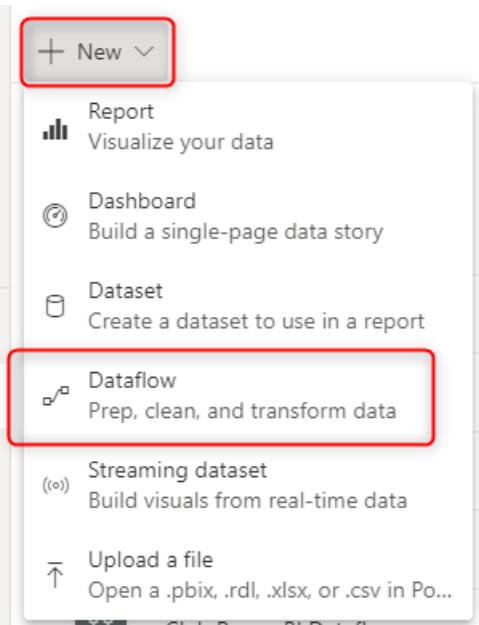
If you don't have folders:

- In the **Queries** pane, select **Ctrl** as you select the queries you want to migrate.
- Select **Ctrl+C**.



3. Paste the copied queries into a dataflow:

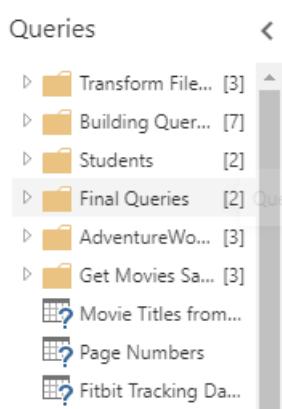
- Create a dataflow, if you don't have one already.



- b. Open the dataflow in Power Query Editor, and in the **Queries** pane, select **Ctrl+V** to paste the copied folders or queries.



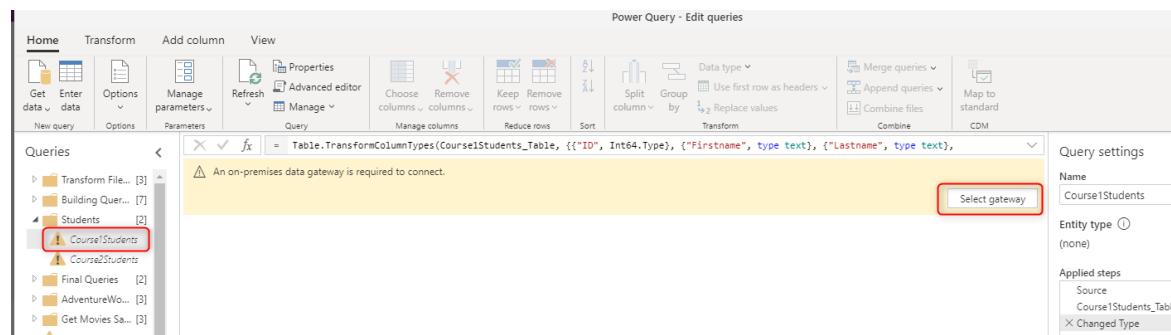
The image below shows an example of successfully copied folders.



4. Connect the on-premises data gateway.

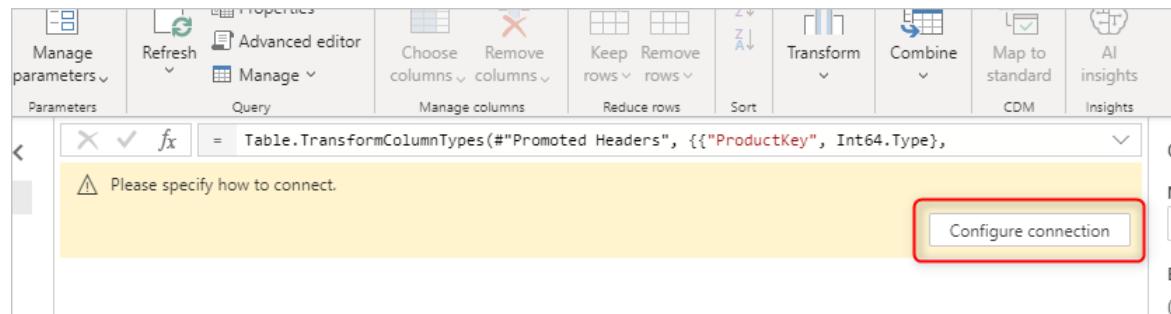
If your data source is an on-premises source, you need to perform an extra step. Examples of on-premises sources can be Excel files in a shared folder in a local domain, or a SQL Server database hosted in an on-premises server.

A dataflow, as a cloud-based service, requires the on-premises data gateway to connect to the on-premises data source. You need to [install and configure the gateway](#) for that source system, and then add [the data source for it](#). After you've completed these steps, you can select the on-premises data gateway when you create the entity in the dataflow.



The gateway isn't needed for data sources residing in the cloud, such as an Azure SQL database.

5. Configure the connection to the data source by selecting **Configure connection** and entering credentials or anything else you need to connect to the data source at this stage.



6. Verify the connection:

If you've done all the steps successfully, you'll see a preview of the data in the Power Query Editor.

Some Power Query Desktop functions require a gateway in Power Query Online

Some of the functions might require a gateway, even if their source is not located on-premises. Among these are functions such as `Web.BrowserContents` and `Web.Page`. If this happens, you might get an error message indicating which specific function isn't supported. The figure below shows an example of one of these scenarios.



If a scenario like this happens, you have two options. You can set up the gateway for that data source, or you can update the query in the Power Query Editor for the dataflow by using a set of steps that are supported without the need for the gateway.

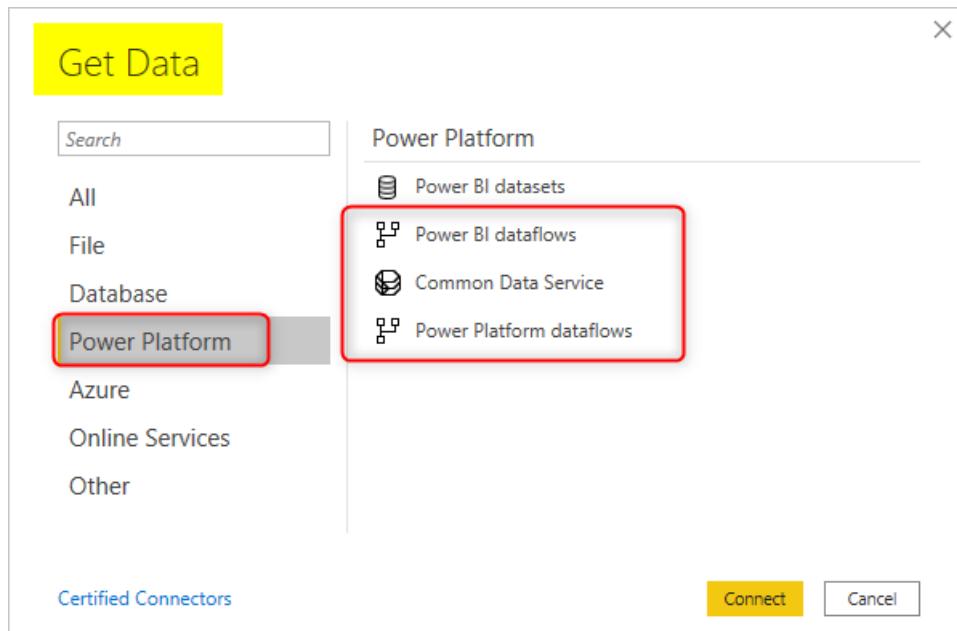
Refresh the dataflow entities

After migrating your queries to the dataflow, you must refresh the dataflow to get data loaded into these entities. You can refresh a dataflow manually or configure an automatic refresh based on the schedule of your choice.

A screenshot of the Power BI dataset list interface. At the top, there are tabs: 'All', 'Content', and 'Datasets + dataflows'. The 'Datasets + dataflows' tab is selected and highlighted with a yellow underline. Below the tabs is a table with columns: 'Name', 'Type', and 'Owner'. A single row is visible, representing a dataflow named 'migrate sample'. To the left of the name is a small icon. To the right of the name are two buttons: a circular arrow icon for refreshing and a clipboard icon for copying. Both of these buttons are enclosed in a red rectangular box. The 'Type' column shows 'Dataflow' and the 'Owner' column shows 'Reza Rad'.

Get data from Power Query Desktop

You can now get data from dataflow entities in Power BI Desktop by using the Power Platform dataflow or Dataverse connectors (depending on what type of dataflow you're using, analytical or standard). More information: [Connect to data created by Power Platform dataflows in Power BI Desktop](#)



Using an on-premises data gateway in Power Platform dataflows

1/15/2022 • 5 minutes to read • [Edit Online](#)

Install an on-premises data gateway to transfer data quickly and securely between a Power Platform dataflow and a data source that isn't in the cloud, such as an on-premises SQL Server database or an on-premises SharePoint site. You can view all gateways for which you have administrative permissions and manage permissions and connections for those gateways.

With a gateway, you can connect to on-premises data through these connections:

- SharePoint
- SQL Server
- Oracle
- Informix
- Filesystem
- DB2

Prerequisites

Power BI service

- A Power BI service account. Don't have one? [Sign up for 60 days free](#).
- Administrative permissions on a gateway. These permissions are provided by default for gateways you install. Administrators can grant other people permissions for gateways.

Power Apps

- A Power Apps account. Don't have one? [Sign up for 30 days free](#).
- Administrative permissions on a gateway. These permissions are provided by default for gateways you install. Administrators can grant other people permissions for gateways.
- A license that supports accessing on-premises data using an on-premises gateway. More information: "Connect to your data" row of the "Explore Power Apps plans" table in the [Power Apps pricing](#) page.
- Gateways and on-premises connections can only be created and used in the user's default environment. More information: [Working with environments and Microsoft Power Apps](#).

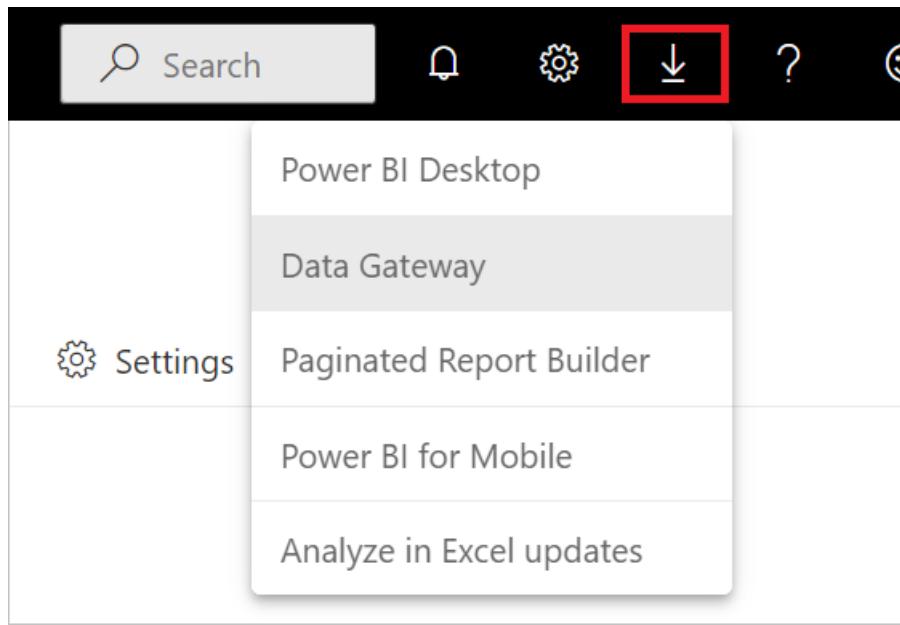
Install a gateway

You can install an on-premises data gateway directly from the online service.

Install a gateway from Power BI service

To install a gateway from Power BI service:

1. Select the downloads button in the upper right corner of Power BI service, and choose **Data Gateway**.



2. Install the gateway using the instructions provided in [Install an on-premises data gateway](#).

Install a gateway from Power Apps

To install a gateway from Power Apps:

1. In the left navigation pane of [powerapps.com](#), select Data > Gateways.

- Home
- Learn
- Apps
- Create
- Data ^
- Tables
- Choices
- Dataflows
- Azure Synapse Link
- Connections
- Custom Connectors
- Gateways

2. Select [New gateway](#).



You don't have a gateway installed.

To connect to your on-premises data, you can install one now. [Learn more](#)

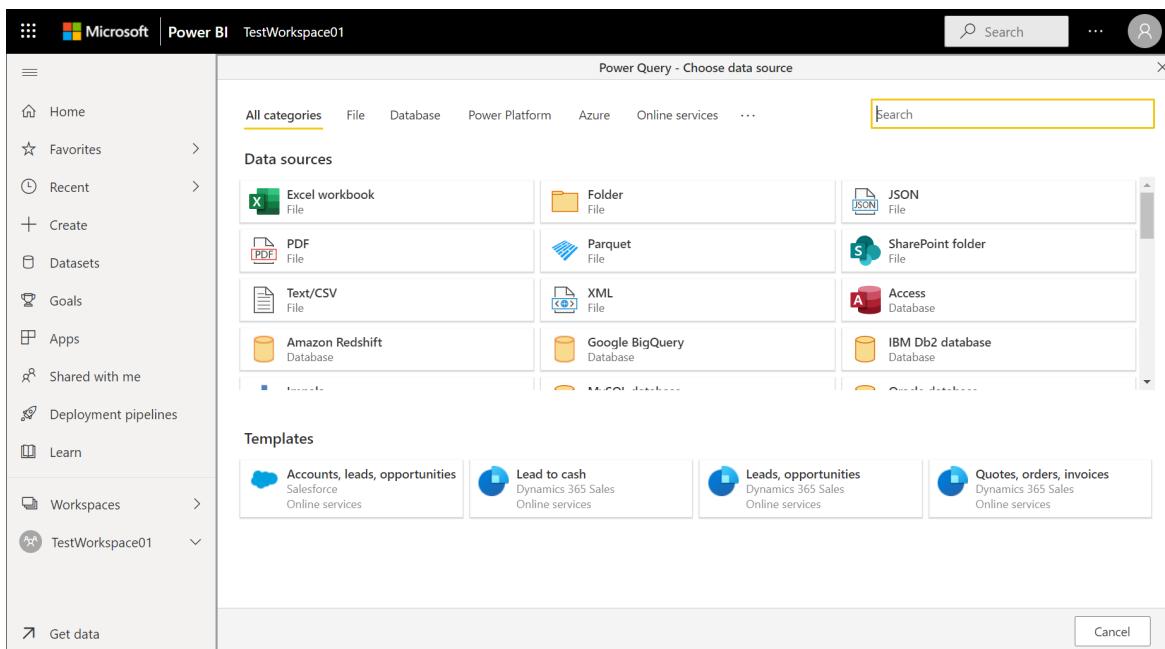
[New gateway](#)

3. In the **On-Premises Data Gateway** section, select [Download](#).

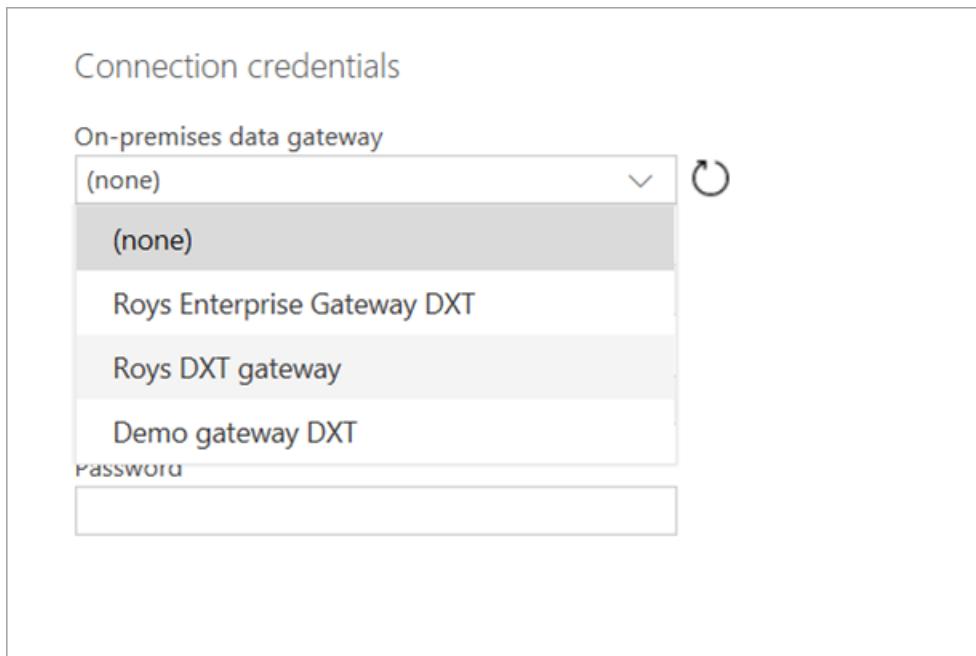
4. Install the gateway using the instructions provided in [Install an on-premises data gateway](#).

Use an on-premises data source in a dataflow

1. For instructions on how to create a new dataflow, go to [Create a dataflow from a data source](#).
2. Select an on-premises data source from the data sources list.



3. Provide the connection details for the enterprise gateway that will be used to access the on-premises data. You must select the gateway itself, and provide credentials for the selected gateway. Only gateways for which you're an administrator appear in the list.



You can change the enterprise gateway used for a given dataflow and change the gateway assigned to all of your queries using the dataflow authoring tool.

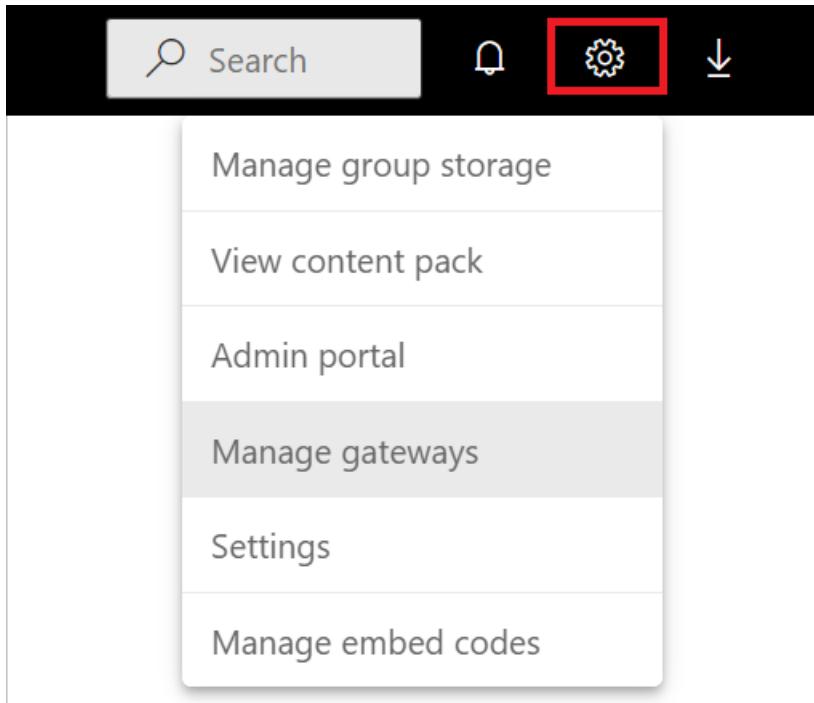
NOTE

The dataflow will try to find or create the required data sources using the new gateway. If it can't do so, you won't be able to change the gateway until all needed dataflows are available from the selected gateway.

View and manage gateway permissions

Power BI service gateway permissions

1. Select the setup button in the upper right corner of Power BI service, choose **Manage gateways**, and then select the gateway you want.



2. To add a user to the gateway, select the **Administrators** table and enter the email address of the user you would like to add as an administrator. Using gateways in dataflows requires Admin permission on the gateway. Admins have full control of the gateway, including adding users, setting permissions, creating connections to all available data sources, and deleting the gateway.

Power Apps gateway permissions

1. In the left navigation pane of [powerapps.com](#), select **Gateways** and then select the gateway you want.
2. To add a user to a gateway, select **Users**, specify a user or group, and then specify a permission level. Creating new data sources with a gateway in dataflows requires Admin permission on the gateway. Admins have full control of the gateway, including adding users, setting permissions, creating connections to all available data sources, and deleting the gateway.

View and manage gateway connections

Power BI service gateway connections

1. Select the setup button in the upper right corner of Power BI service, choose **Manage gateways**, and then select the gateway you want.
2. Perform the action that you want:
 - To view details and edit the settings, select **Gateway Cluster Settings**.
 - To add users as administrators of the gateway, select **Administrators**.
 - To add a data source to the gateway, select **Add Data Source**, enter a data source name and choose the data source type under **Data Source Settings**, and then enter the email address of the person who will use the data source.
 - To delete a gateway, select the ellipsis to the right of the gateway name and then select **Remove**.

Power Apps gateway connections

1. In the left navigation bar of [powerapps.com](#), select **Gateways**, and then choose the gateway you want.
2. Perform the action that you want:
 - To view details, edit the settings, or delete a gateway, select **Connections**, and then select a connection.
 - To share a connection, select **Share** and then add or remove users.

NOTE

You can only share some types of connections, such as a SQL Server connection. For more information, see [Share canvas-app resources in Power Apps](#).

For more information about how to manage a connection, see [Manage canvas-app connections in Power Apps](#).

Limitations

There are a few known limitations when using enterprise gateways and dataflows.

- Each dataflow can use only one gateway. As such, all queries should be configured using the same gateway.
- Changing the gateway impacts the entire dataflow.
- If several gateways are needed, the best practice is to build several dataflows (one for each gateway) and use the compute or table reference capabilities to unify the data.
- Dataflows are only supported using enterprise gateways. Personal gateways won't be available for selection in the drop-down lists and settings screens.
- Creating new data sources with a gateway in dataflows is only supported for people with *Admins* permissions. *Can use* and *Can use + share* permissions levels aren't currently supported.

Troubleshooting

When you attempt to use an on-premises data source to publish a dataflow, you might come across the following `MashupException` error:

```
AzureDataLakeStorage failed to get the response:  
'The underlying connection was closed: An unexpected error occurred on a send.'
```

This error usually occurs because you're attempting to connect to an Azure Data Lake Storage endpoint through a proxy, but you haven't properly configured the proxy settings for the on-premises data gateway. To learn more about how to configure these proxy settings, go to [Configure proxy settings for the on-premises data gateway](#).

For more information about troubleshooting issues with gateways, or configuring the gateway service for your network, go to the [On-premises data gateway documentation](#).

Next steps

- [Create and use dataflows in Power Apps](#)
- [Add data to a table in Microsoft Dataverse by using Power Query](#)
- [Connect Azure Data Lake Storage Gen2 for dataflow storage](#)

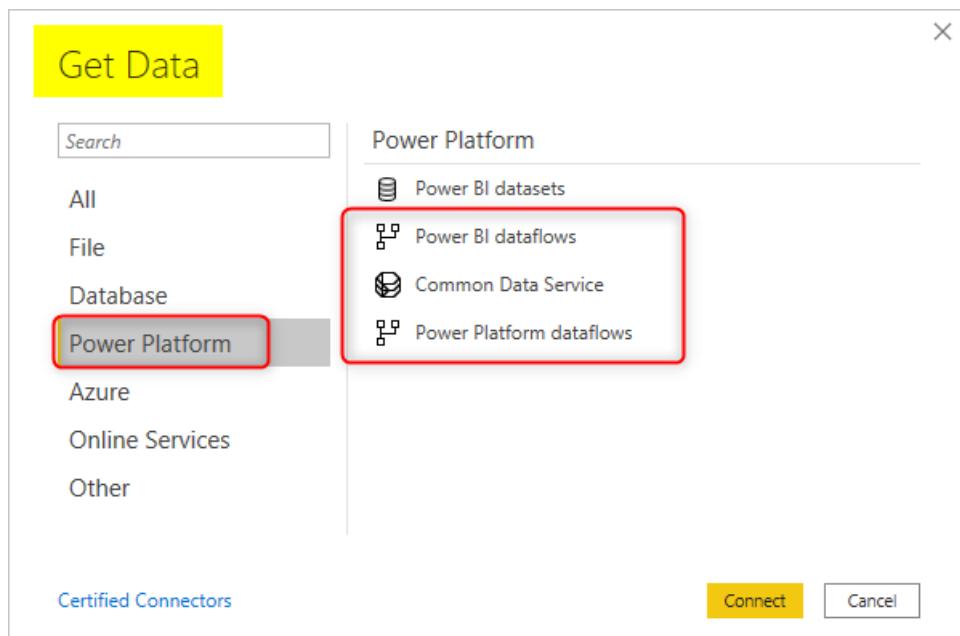
Using the output of Microsoft Power Platform dataflows from other Power Query experiences

1/15/2022 • 2 minutes to read • [Edit Online](#)

You can use the output of Microsoft Power Platform dataflows from the Power Query experience in other products. For example, in Power BI Desktop—or even in another dataflow—you can get data from the output of a dataflow. In this article, you'll learn how to do so.

Get data from dataflows in Power BI

If you're using Power BI for data analysis, you can get data from Power Platform dataflows or Power BI dataflows by choosing those data sources.



When you get data from a dataflow, the data is imported into the Power BI dataset. The dataset then needs to be refreshed. You can choose whether to perform a one-time refresh or an automatic refresh on a schedule you specify. Scheduled refreshes for the dataset can be configured in Power BI.

DirectQuery from dataflows

Power BI dataflows also support a DirectQuery connection. If the size of the data is so large that you don't want to import all of it into the Power BI dataset, you can create a DirectQuery connection. DirectQuery won't copy the data into the Power BI dataset. The tables in the Power BI dataset that get their data from a DirectQuery-sourced dataflow don't need a scheduled refresh, because their data will be fetched live from the dataflow.

To use DirectQuery for the dataflows, you need to enable the compute engine on your Power BI Premium capacity, and then refresh the dataflow before it can be consumed in DirectQuery mode. For more information, go to [Power BI Dataflows DirectQuery Support](#).

Dataflows can get data from other dataflows

If you'd like to reuse data created by one dataflow in another dataflow, you can do so by using the dataflow connector in the Power Query editor experience when you create the new dataflow.

When getting data from the output of another dataflow, a [linked entity](#) will be created. Linked entities provide a way to make data created in an upstream dataflow available in a downstream dataflow, without copying the data to the downstream dataflow. Because linked entities are just pointers to entities created in other dataflows, they're kept up to date by the refresh logic of the upstream dataflow. If both dataflows reside in the same workspace or environment, those dataflows will be refreshed together, to keep data in both dataflows always up to date. More information: [Link entities between dataflows](#)

Separating data transformation from data consumption

When you use the output of a dataflow in other dataflows or datasets, you can create an abstraction between the data transformation layer and the rest of the data model. This abstraction is important because it creates a multi-role architecture, in which the Power Query customer can focus on building the data transformations, and data modelers can focus on data modeling.

Next Steps

The following articles provide more details about related articles.

- [Creating and using dataflows in Power BI](#)
- [Link entities between dataflows in Power BI](#)
- [Connect to data created by Power BI dataflows in Power BI Desktop \(Beta\)](#)
- [Create and use dataflows in Power Platform](#)
- [Link entities between dataflows \(Power Platform\)](#)

Creating computed entities in dataflows

1/15/2022 • 3 minutes to read • [Edit Online](#)

You can perform *in-storage computations* when using dataflows with a Power BI Premium subscription. This lets you do calculations on your existing dataflows, and return results that enable you to focus on report creation and analytics.

The screenshot shows the 'Edit queries' interface in Power BI. At the top, there are several menu items: 'Get data', 'Refresh', 'Options', 'Manage columns', 'Transform table', 'Reduce rows', and a 'Power Query' button. Below the menu, there is a list of dataflows: 'Account', 'ServiceCalls', and 'ServiceCallsAggregated'. The 'ServiceCallsAggregated' item is highlighted with a yellow background and has a pink arrow pointing to it from the bottom-left. To the right of the list is a preview pane displaying a table with five rows. The table has two columns: 'accountid' and 'numberOfServiceC...'. The data is as follows:

	accountid	numberOfServiceC...
1	0011r00001mv6joAAA	87
2	0011r00001mv6jpAAA	87
3	0011r00001mv6jqAAA	96
4	0011r00001mv6jrAAA	87
5	0011r00001mv6jsAAA	87

To perform in-storage computations, you first must create the dataflow and bring data into that Power BI dataflow storage. After you have a dataflow that contains data, you can create *computed entities*, which are entities that do in-storage computations.

There are two ways you can connect dataflow data to Power BI:

- [Using self-service authoring of a dataflow](#)
- Using an external dataflow

The following sections describe how to create computed entities on your dataflow data.

How to create computed entities

After you have a dataflow with a list of entities, you can perform calculations on those entities.

In the dataflow authoring tool in the Power BI service, select **Edit entities**. Then right-click the entity you want to use as the basis for your computed entity and on which you want to perform calculations. On the shortcut menu, select **Reference**.

For the entity to be eligible as a computed entity, **Enable load** must be selected, as shown in the following image. Right-click the entity to display this shortcut menu.

Edit queries

A screenshot of the Power BI 'Edit queries' interface. On the left, there's a list of entities: Account, ServiceCalls, and ServiceCallsAggregated. The 'ServiceCalls' entity is selected and highlighted with a yellow background. A context menu is open over this entity, listing options: Delete, Rename, Enable load (which is checked), Reference, Duplicate, Move to group, and Advanced editor. To the right of the menu is a table view showing data from the ServiceCalls entity. The columns are labeled 'Column1' and 'Column2'. The data includes account IDs and call dates. A pink arrow points from the text in the previous paragraph to the 'Enable load' option in the menu.

	Column1	Column2
accountid	callDate	
0011r00001mv6joAAA	5/29/2030	
0011r00001mv6jpAAA	6/14/2030	
0011r00001mv6jqAAA	1/23/2030	
0011r00001mv6jrAAA	4/28/2030	
0011r00001mv6jsAAA	1/2/2021	
0011r00001mv6jtAAA	1/29/2030	
0011r00001mv6juAAA	3/28/2030	
0011r00001mv6jvAAA	2/6/2020	
0011r00001mv6jwAAA	1/20/2030	
11	0011r00001mv6jxAAA	1/16/2030
12	0011r00001mv6jyAAA	2/27/2030

By selecting **Enable load**, you create a new entity whose source is the referenced entity. The icon changes to the **computed** icon, as shown in the following image.

A screenshot of the Power BI 'Edit queries' interface. The list of entities now includes 'ServiceCallsAggregated', which is selected and highlighted with a yellow background. A context menu is open over this entity, listing options: Get data, Refresh, Options, Manage columns, Transform table, Reduce rows, and a 'Compute' option (indicated by a red arrow). To the right of the menu is a table view showing data from the ServiceCallsAggregated entity. The columns are labeled 'accountid' and 'numberOfServiceCalls'. The data includes account IDs and their respective service call counts. A pink arrow points from the text in the previous paragraph to the 'Compute' option in the menu.

	accountid	1.2 numberOfServiceC...
1	0011r00001mv6joAAA	87
2	0011r00001mv6jpAAA	87
3	0011r00001mv6jqAAA	96
4	0011r00001mv6jrAAA	87
5	0011r00001mv6jsAAA	87

Any transformation you do on this newly created entity will be run on the data that already resides in Power BI dataflow storage. That means that the query won't run against the external data source from which the data was imported (for example, the SQL database from which the data was pulled).

Example use cases

What kind of transformations can be done with computed entities? Any transformation that you usually specify by using the transformation user interface in Power BI, or the M editor, are all supported when performing in-

storage computation.

Consider the following example. You have an Account entity that contains the raw data for all the customers from your Dynamics 365 subscription. You also have ServiceCalls raw data from the service center, with data from the support calls that were performed from the different accounts on each day of the year.

Imagine you want to enrich the Account entity with data from ServiceCalls.

First you would need to aggregate the data from the ServiceCalls to calculate the number of support calls that were done for each account in the last year.

The screenshot shows the 'Edit queries' interface in Power Query. A 'Group by' dialog is open over a table of raw ServiceCall data. The table has columns: accountid (1 to 15), accountnumber (e.g., 0011r00001m), and createdon (3/11/2030). The 'Group by' dialog specifies grouping by 'accountid' and creating a new column 'numberOfServiceCalls' with the operation 'Count rows'. Buttons for 'OK' and 'Cancel' are visible at the bottom right of the dialog.

Next, you merge the Account entity with the ServiceCallsAggregated entity to calculate the enriched Account table.

The screenshot shows the 'Edit queries' interface in Power Query. A 'Merge' dialog is open, combining two tables: 'Account' and 'ServiceCallsAggregated'. The 'Account' table has columns: accountid (1 to 12), accountnumber (e.g., 0011r00001m), accountratingcode (e.g., Cold, Warm), and address1_address (Bill To). The 'ServiceCallsAggregated' table has columns: accountid (1 to 12) and a new column '12 numberOfServiceCalls' (values 87, 87, 96, 87, 87, 87). The 'Join kind' is set to 'Left outer (all from first, matching from sec...)'.

Then you can see the results, shown as EnrichedAccount in the following image.

ckexchange	telephone1	tickersymbol	transactioncurren...	websiteurl	1.2 numberOfServiceC...
1	(650) 867-3450		(null)	www.genepoint.com	83
2	(512) 757-6000	EDGE	(null)	http://edgecomm.com	87
3	+44 191 4956203	UOS	(null)	http://www.uos.com	86
4	(336) 222-7000	BTXT	(null)	www.burlington.com	87
5	(650) 450-8810	UOS	(null)	http://www.uos.com	83
6	(014) 427-4427	PYR	(null)	www.pyramid.com	96
7	(785) 241-6200		(null)	dickenson-consulting.com	87
8	(312) 596-1000	GHTL	(null)	www.grandhotels.com	87
9	(212) 642-5500	UOS	(null)	http://www.uos.com	87
10	(503) 421-7800	EXLT	(null)	www.expressl&t.net	97
11	(520) 773-9050		(null)	www.universityofarizona.c...	87
12	(415) 901-7000		(null)	www.sforce.com	83

And that's it—the transformation is done on the data in the dataflow that resides in your Power BI Premium subscription, not on the source data.

Considerations and limitations

It's important to note that if you remove the workspace from Power BI Premium capacity, the associated dataflow will no longer be refreshed.

When working with dataflows specifically created in an organization's Azure Data Lake Storage account, linked entities and computed entities only work properly when the entities reside in the same storage account. More information: [Connect Azure Data Lake Storage Gen2 for dataflow storage](#)

Linked entities are only available for dataflows created in Power BI and Power Apps. As a best practice, when doing computations on data joined by on-premises and cloud data, create a new entity to perform such computations. This provides a better experience than using an existing entity for computations, such as an entity that is also querying data from both sources and doing in-storage transformations.

See also

- [Computed entity scenarios and use cases](#)

This article described computed entities and dataflows. Here are some more articles that might be useful:

- [Self-service data prep in Power BI](#)
- [Using incremental refresh with dataflows](#)
- [Connect to data sources for dataflows](#)
- [Link entities between dataflows](#)

The following links provide additional information about dataflows in Power BI and other resources:

- [Create and use dataflows in Power BI](#)
- [Using dataflows with on-premises data sources](#)
- [Developer resources for Power BI dataflows](#)
- [Configure workspace dataflow settings \(Preview\)](#)
- [Add a CDM folder to Power BI as a dataflow \(Preview\)](#)
- [Connect Azure Data Lake Storage Gen2 for dataflow storage \(Preview\)](#)

For more information about Power Query and scheduled refresh, you can read these articles:

- [Query overview in Power BI Desktop](#)
- [Configuring scheduled refresh](#)

For more information about Common Data Model, you can read its overview article:

- [Common Data Model](#)

Link entities between dataflows

1/15/2022 • 5 minutes to read • [Edit Online](#)

With dataflows in Microsoft Power Platform, you can have a single organizational data storage source where business analysts can prep and manage their data once, and then reuse it between different analytics apps in the organization.

When you link entities between dataflows, you can reuse entities that have already been ingested, cleansed, and transformed by dataflows that are owned by others, without the need to maintain that data. The linked entities simply point to the entities in other dataflows, and do *not* copy or duplicate the data.

Linked entities are read-only, so if you want to create transformations for a linked entity, you must create a new computed entity with a reference to the linked entity.

Linked entity availability

Linked entity availability depends on whether you're using dataflows in Power BI or Power Apps. The following sections describe the details for each.

Linked entities in Power BI

To be refreshed, linked entities require a [Power BI Premium](#) subscription. Linked entities are available in any dataflow on a workspace that's hosted on Power BI Premium capacity. There are no limitations on the source dataflow.

Linked entities only work properly in new Power BI workspaces, and, likewise, all linked dataflows must be located in new workspaces. More information: [Create the new workspaces in Power BI](#)

NOTE

Entities differ based on whether they're standard entities or computed entities. Standard entities (often simply referred to as entities) query an external data source, such as a SQL database. Computed entities require Premium capacity on Power BI and run their transformations on data that's already in Power BI storage.

If your dataflow isn't located in a Premium capacity workspace, you can still reference a single query—or combine two or more queries—as long as the transformations aren't defined as in-storage transformations. Such references are considered standard entities. To do this, turn off the **Enable load** option for the referenced queries to prevent the data from being materialized and ingested into storage. From there, you can reference those **Enable load = false** queries, and set **Enable load** to **On** only for the resulting queries that you want to materialize.

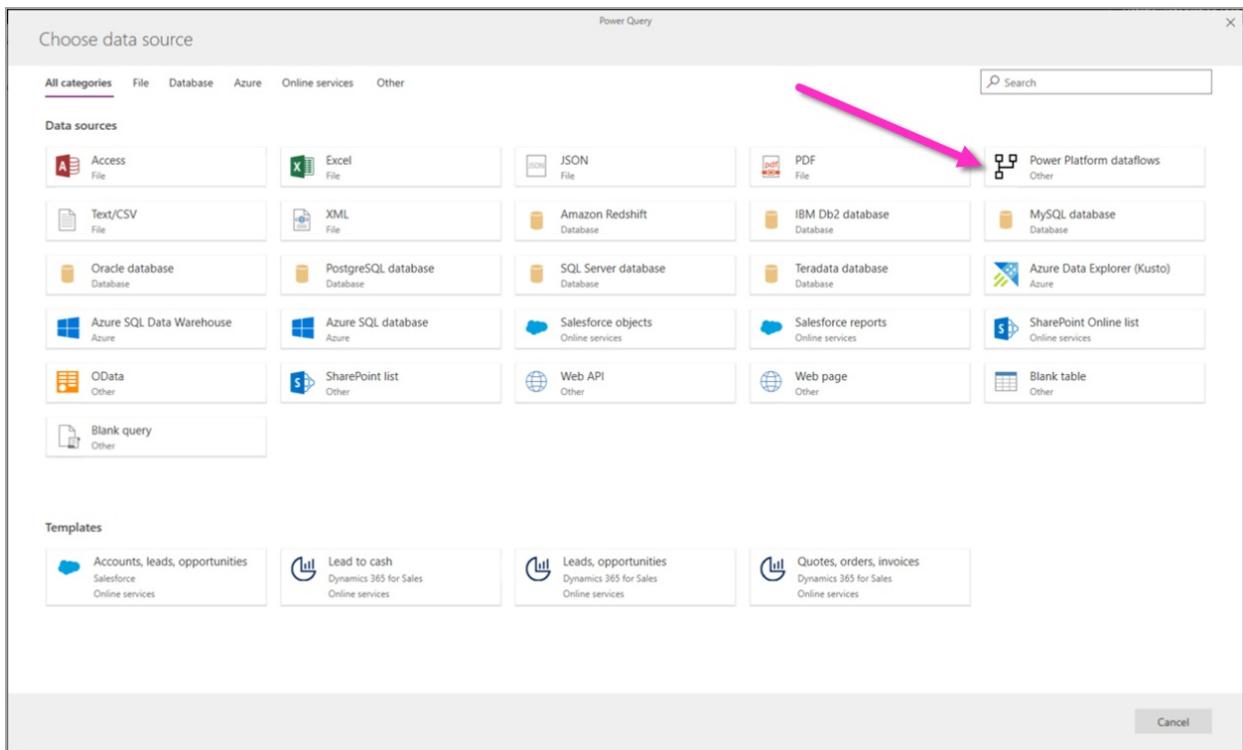
Linked entities in Power Apps

Linked entities are available in both Power Apps Plan 1 and Plan 2.

How to link entities between dataflows

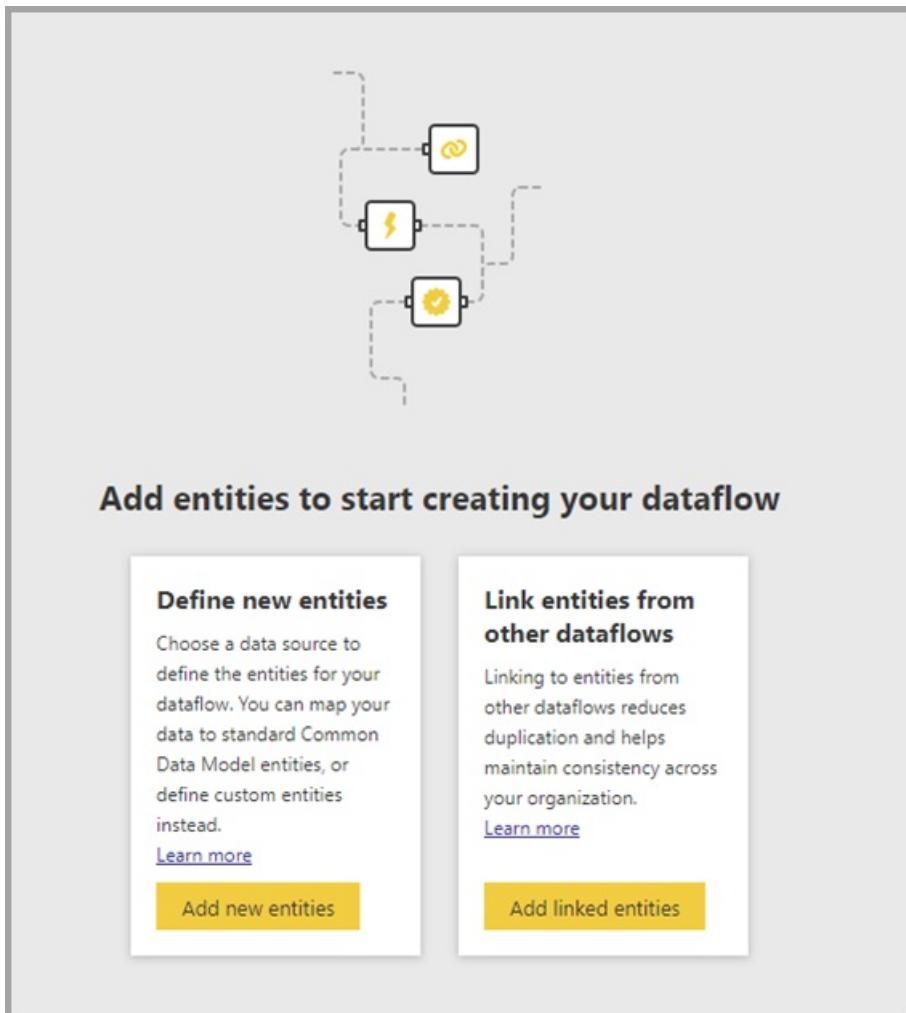
There are a few ways to link entities between dataflows. To link entities in Power BI, you must sign in with your Power BI credentials.

You can select **Get data** from the dataflow authoring tool, which displays a dialog box for selecting the categories and each data source. Then select the **Power Platform dataflows** connector.

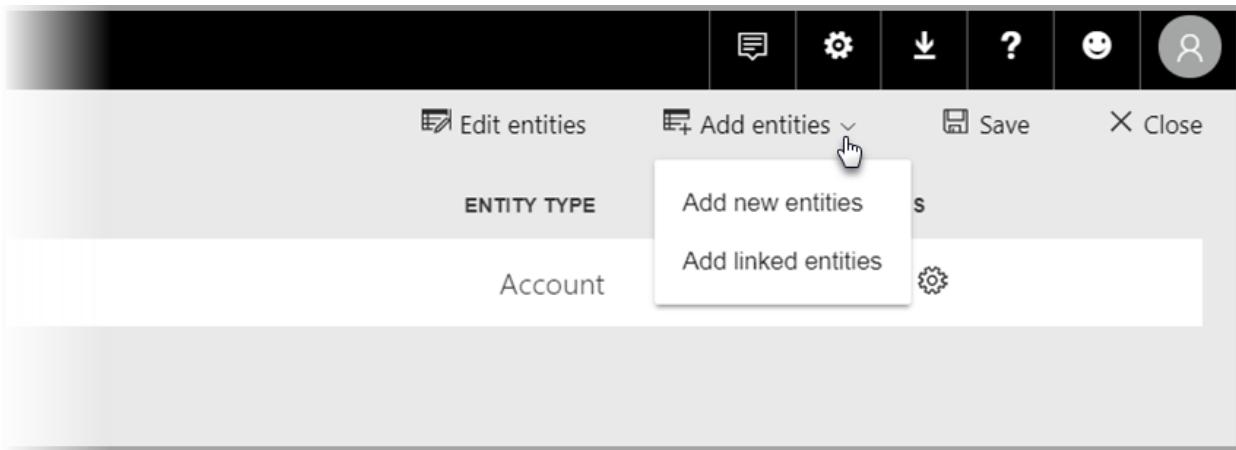


A connection window for the selected data connection is displayed. If credentials are required, you're prompted to provide them.

In Power BI, you can select **Add linked entities** from the dataflow authoring tool.



You can also select **Add linked entities** from the **Add entities** menu in the Power BI service.



A **Navigator** window opens, and you can choose a set of entities you can connect to. The window displays entities for which you have permissions across all workspaces and environments in your organization.

After you select your linked entities, they appear in the list of entities for your dataflow in the authoring tool, with a special icon identifying them as linked entities.

You can also view the source dataflow from the dataflow settings of your linked entity.

Refresh logic of linked entities

The refresh logic of linked entities differs slightly based on whether you're using Power BI or Power Apps, as described in the following sections.

Refresh logic in Power BI

The default refresh logic of linked entities depends on whether the source dataflow is in the same Power BI workspace as the destination dataflow. The following sections describe the behavior of each.

- **Links between workspaces:** Refresh for links from entities in different workspaces behaves like a link to an external data source. When the dataflow is refreshed, it takes the latest data for the entity from the source dataflow. If the source dataflow is refreshed, it doesn't automatically affect the data in the destination dataflow.
- **Links in the same workspace:** When data refresh occurs for a source dataflow, that event automatically triggers a refresh process for dependent entities in all destination dataflows in the same workspace, including any calculated entities based on them. All other entities in the destination dataflow are refreshed according to the dataflow schedule. Entities that depend on more than one source refresh their data whenever any of their sources are refreshed successfully.

NOTE

The entire refresh process is committed at once. Because of this, if the data refresh for the destination dataflow fails, the data refresh for the source dataflow fails as well.

Refresh logic in Power Apps

The refresh logic of linked entities in Power Apps behaves like an external data source. When the dataflow is refreshed, it takes the latest data for the entity from the source dataflow. If the source dataflow is refreshed, it doesn't automatically affect the data in the destination dataflow.

Permissions when viewing reports from dataflows

When creating a Power BI report that includes data based on a dataflow, you can see any linked entities only when you have access to the source dataflow.

Limitations and considerations

There are a few limitations to keep in mind when working with linked entities:

- An entity can be referenced by another dataflows. That reference entity can also be reference by other dataflows and so on, up to 5 times.
- Cyclical dependencies of linked entities aren't allowed.
- The dataflow must be in a [new Power BI workspace](#) or a Power Apps environment.
- A linked entity can't be joined with a regular entity that gets its data from an on-premises data source.
- When using M parameters to address linked entities, if the source dataflow is refreshed, it doesn't automatically affect the data in the destination dataflow.

Next steps

The following articles might be useful as you create or work with dataflows:

- [Self-service data prep in Power BI](#)
- [Using incremental refresh with dataflows](#)
- [Creating computed entities in dataflows](#)
- [Connect to data sources for dataflows](#)

The articles below provide more information about dataflows and Power BI:

- [Create and use dataflows in Power BI](#)
- [Using computed entities on Power BI Premium](#)
- [Using dataflows with on-premises data sources](#)
- [Developer resources for Power BI dataflows](#)

For more information about Power Query and scheduled refresh, you can read these articles:

- [Query overview in Power BI Desktop](#)
- [Configuring scheduled refresh](#)

For more information about Common Data Model, you can read its overview article:

- [Common Data Model - overview](#)

Connect Azure Data Lake Storage Gen2 for dataflow storage

1/15/2022 • 6 minutes to read • [Edit Online](#)

You can configure dataflows to store their data in your organization's Azure Data Lake Storage Gen2 account. This article describes the general steps necessary to do so, and provides guidance and best practices along the way.

IMPORTANT

Dataflow with Analytical tables feature utilizes the Azure Synapse Link for Dataverse service, which may offer varying levels of compliance, privacy, security, and data location commitments. For more information about Azure Synapse Link for Dataverse, go to the [blog article](#).

There are some advantages to configuring dataflows to store their definitions and datafiles in your data lake, such as:

- Azure Data Lake Storage Gen2 provides an enormously scalable storage facility for data.
- Dataflow data and definition files can be leveraged by your IT department's developers to leverage Azure data and artificial intelligence (AI) services as demonstrated in the GitHub samples from Azure data services.
- It enables developers in your organization to integrate dataflow data into internal applications and line-of-business solutions, using developer resources for dataflows and Azure.

Requirements

To use Azure Data Lake Storage Gen2 for dataflows, you need the following:

- A Power Apps environment. Any Power Apps plan will allow you to create dataflows with Azure Data Lake Storage Gen2 as a destination. You'll need to be authorized in the environment as a maker.
- An Azure subscription. You need an Azure subscription to use Azure Data Lake Storage Gen2.
- A resource group. Use a resource group you already have, or create a new one.
- An Azure storage account. The storage account must have the Data Lake Storage Gen2 feature enabled.

TIP

If you don't have an Azure subscription, [create a free trial account](#) before you begin.

Prepare your Azure Data Lake Storage Gen2 for Power Platform dataflows

Before you configure your environment with an Azure Data Lake Storage Gen2 account, you must create and configure a storage account. Here are the requirements for Power Platform dataflows:

1. The storage account must be created in the same Azure Active Directory tenant as your Power Apps tenant.
2. We recommend that the storage account is created in the same region as the Power Apps environment you plan to use it in. To determine where your Power Apps environment is, contact your environment admin.
3. The storage account must have the Hierarchical Name Space feature enabled.

4. You must be granted an Owner role on the storage account.

The following sections walk through the steps necessary to configure your Azure Data Lake Storage Gen2 account.

Create the storage account

Follow the steps in [Create an Azure Data Lake Storage Gen2 storage account](#).

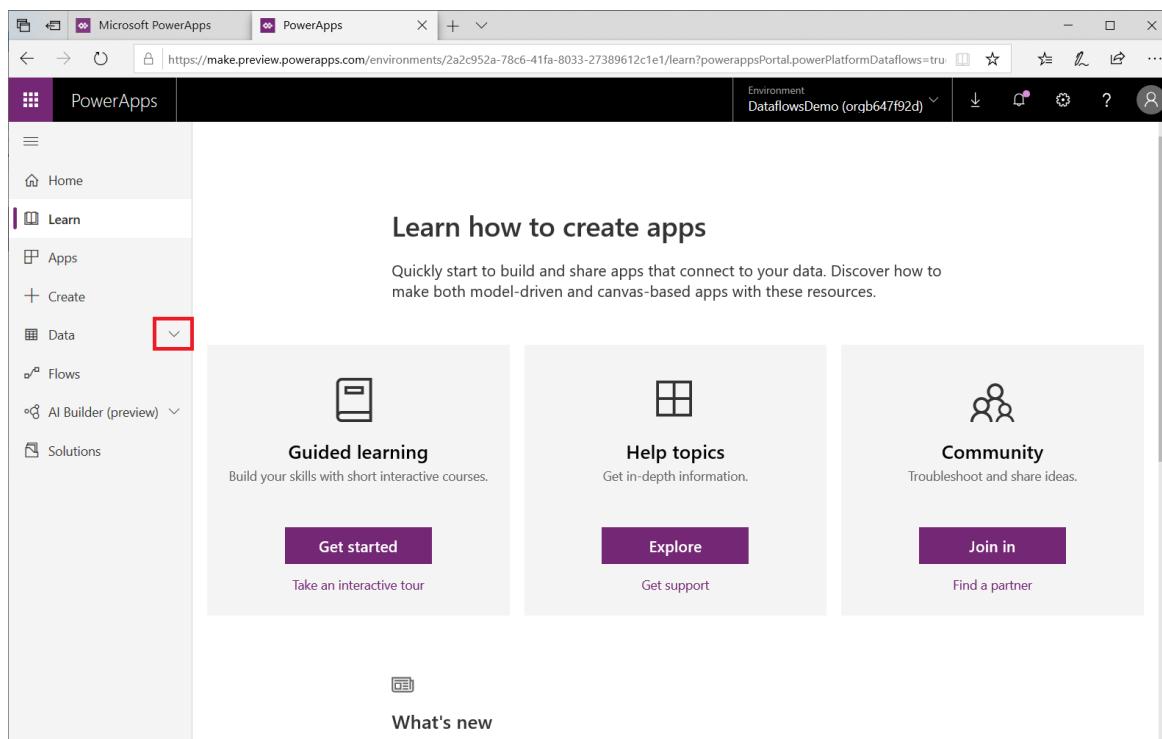
1. Make sure you select the same region as your environment and set your storage as StorageV2 (general purpose v2).
2. Make sure you enable the hierarchical namespace feature.
3. We recommend that you set the replication setting to Read-access geo-redundant storage (RA-GRS).

Connect your Azure Data Lake Storage Gen2 to Power Apps

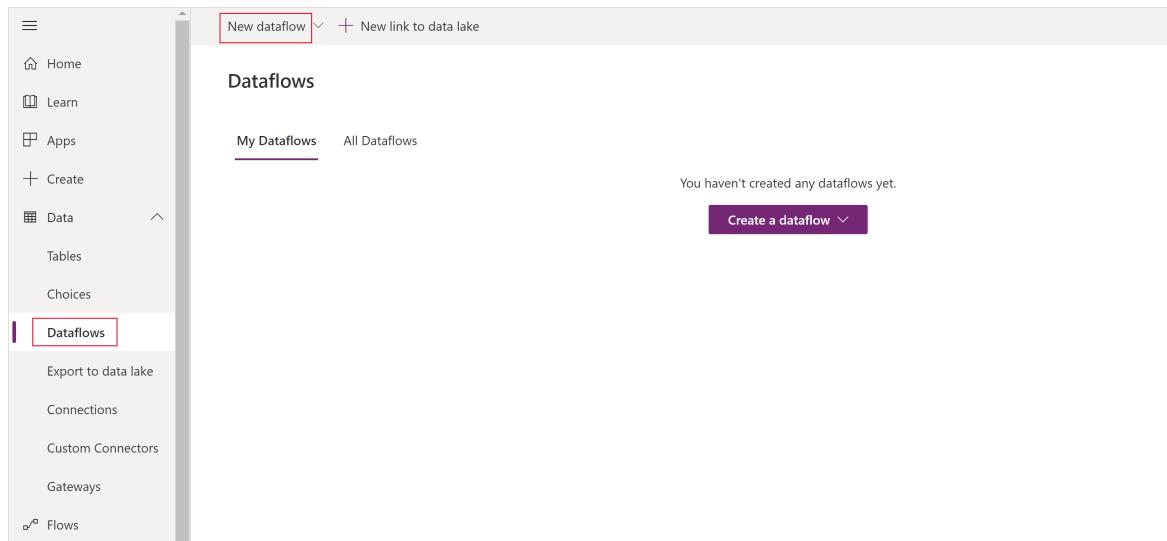
Once you've set up your Azure Data Lake Storage Gen2 account in the Azure portal, you're ready to connect it to a specific dataflow or a Power Apps environment. Connecting the lake to an environment allows other makers and admins in the environment to create dataflows that store their data in your organization's lake as well.

To connect your Azure Data Lake Storage Gen2 account with the dataflow, follow these steps:

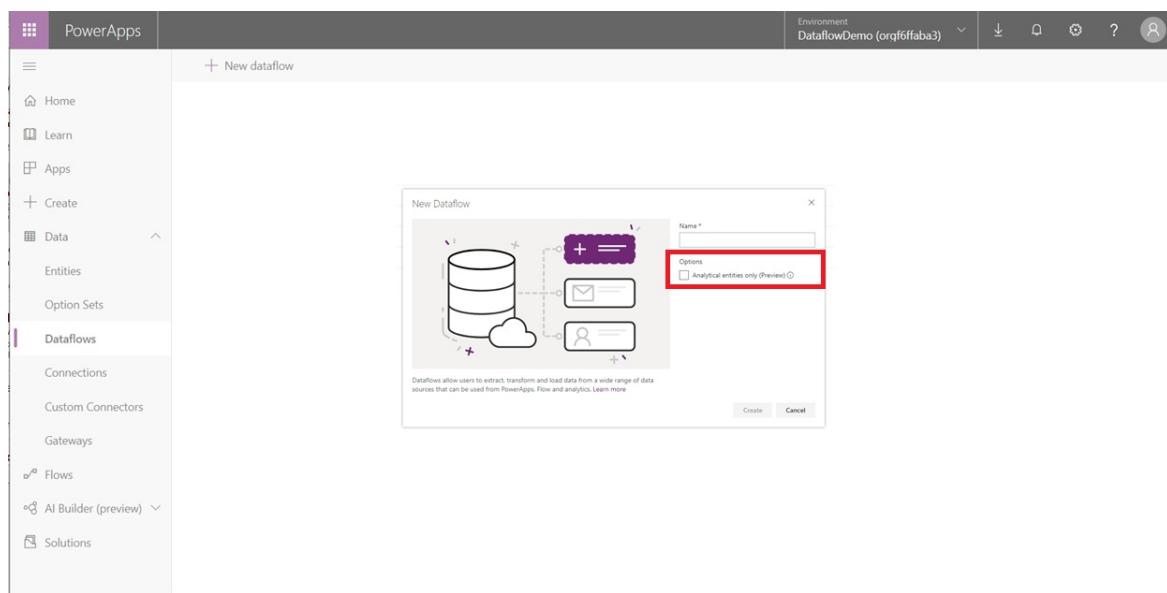
1. Sign in to [Power Apps](#), and verify which environment you're in. The environment switcher is located on the right side of the header.
2. On the left navigation pane, select the down arrow next to **Data**.



3. In the list that appears, select **Dataflows** and then on the command bar select **New dataflow**.



4. Select the analytical tables you want. These tables indicate what data you want to store in your organization's Azure Data Lake Store Gen2 account.

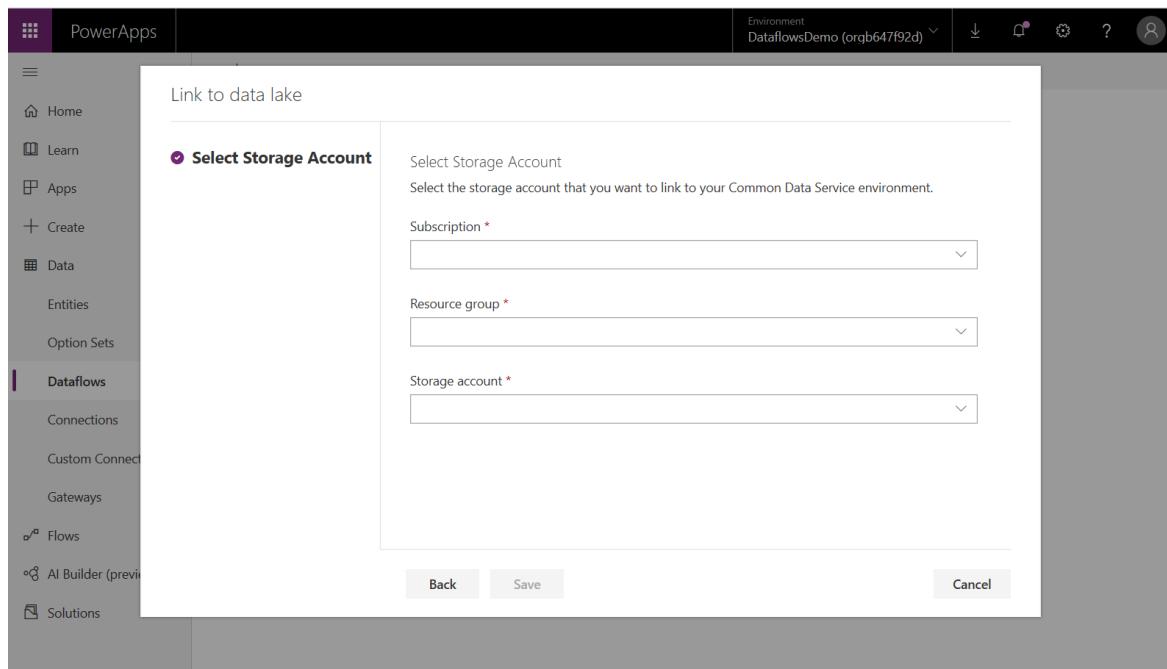


Select the storage account to use for dataflow storage

If a storage account hasn't yet been associated with the environment, a **Link to data lake** dialog box appears. You'll need to sign in and find the data lake you created in the previous steps. In this example, no data lake is associated with the environment and so a prompt occurs to add one.

1. Select storage account.

The **Select Storage Account** screen appears.



2. Select the **Subscription ID** of the storage account.
3. Select the **Resource group name** in which the storage account was created.
4. Enter the **Storage account name**.
5. Select **Save**.

Once these steps are successfully completed, your Azure Data Lake Storage Gen2 account is connected to Power Platform Dataflows and you can continue to create a dataflow.

Considerations and limitations

There are a few considerations and limitations to keep in mind when working with your dataflow storage:

- Linking an Azure Data Lake Store Gen2 account for dataflow storage isn't supported in the default environment.
- Once a dataflow storage location is configured for a dataflow, it can't be changed.
- By default, any member of the environment can access dataflow data using the Power Platform Dataflows Connector. However, only the owners of a dataflow can access its files directly in Azure Data Lake Storage Gen2. To authorize more people to access the dataflows data directly in the lake, you must authorize them to the dataflow's **CDM Folder** in the data lake or the data lake itself.
- When a dataflow is deleted, its **CDM Folder** in the lake will also be deleted.

IMPORTANT

You shouldn't change files created by dataflows in your organization's lake or add files to a dataflow's **CDM Folder**. Changing files might damage dataflows or alter their behavior and is not supported. Power Platform Dataflows only grants read access to files it creates in the lake. If you authorize other people or services to the filesystem used by Power Platform Dataflows, only grant them read access to files or folders in that filesystem.

Privacy notice

By enabling the creation of dataflows with Analytical tables in your organization, via the Azure Synapse Link for Dataverse service, details about the Azure Data Lake storage account, such as the name of the storage account, will be sent to and stored in the Azure Synapse Link for Dataverse service, which is currently located outside the

PowerApps compliance boundary and may employ lesser or different privacy and security measures than those typically in PowerApps. Note that you may remove the data lake association at any time to discontinue use of this functionality and your Azure Data Lake storage account details will be removed from the Azure Synapse Link for Dataverse service. Further information about Azure Synapse Link for Dataverse is available in [this article](#).

Frequently asked questions

What if I had previously created dataflows in my organization's Azure Data Lake Storage Gen2 and would like to change their storage location?

You can't change the storage location of a dataflow after it was created.

When can I change the dataflow storage location of an environment?

Changing the environment's dataflow storage location isn't currently supported.

Next steps

This article provided guidance about how to connect an Azure Data Lake Storage Gen2 account for dataflow storage.

For more information about dataflows, the Common Data Model, and Azure Data Lake Storage Gen2, go to these articles:

- [Self-service data prep with dataflows](#)
- [Creating and using dataflows in Power Apps](#)
- [Add data to a table in Microsoft Dataverse](#)

For more information about Azure storage, go to this article:

- [Azure Storage security guide](#)

For more information about the Common Data Model, go to these articles:

- [Common Data Model - overview](#)
- [Common Data Model folders](#)
- [CDM model file definition](#)

You can ask questions in the [Power Apps Community](#).

What is the storage structure for analytical dataflows?

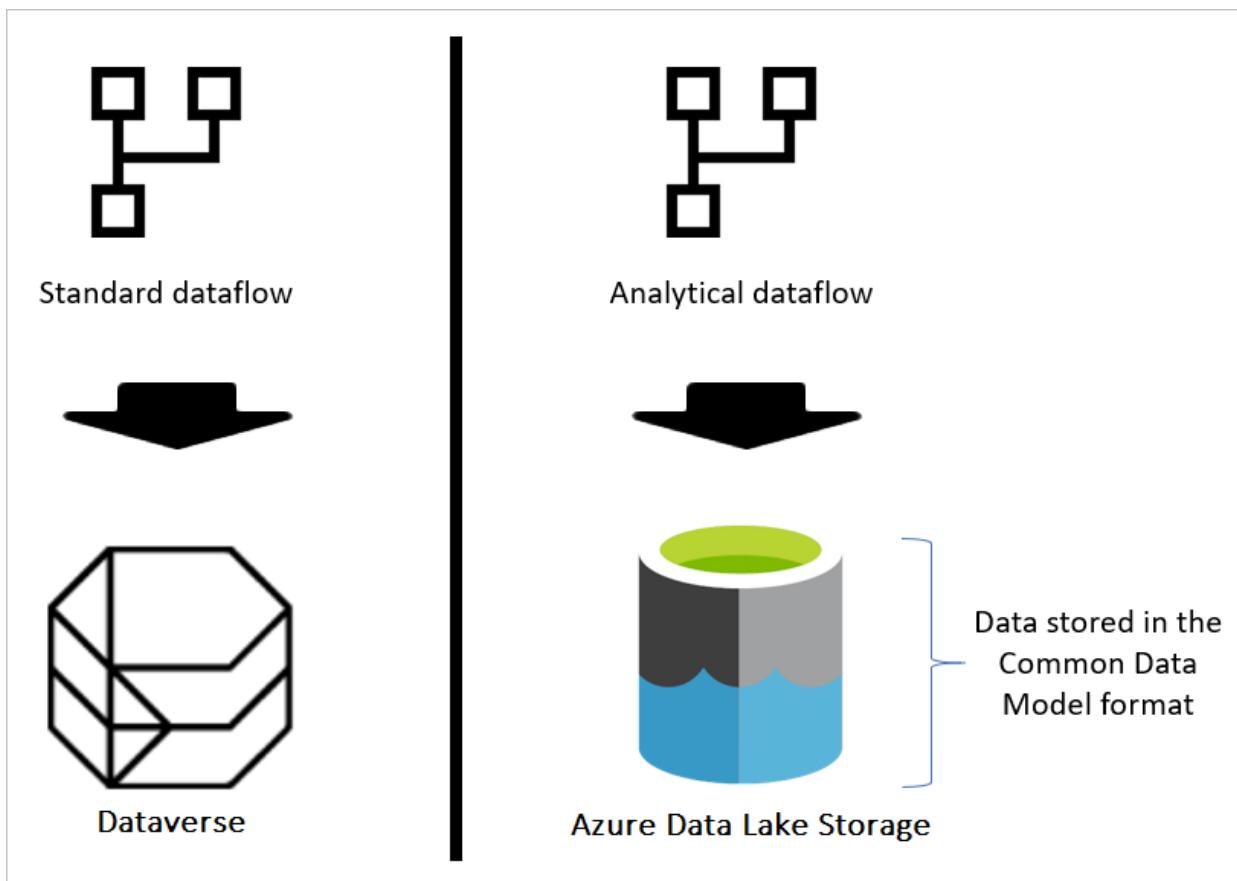
1/15/2022 • 3 minutes to read • [Edit Online](#)

Analytical dataflows store both data and metadata in Azure Data Lake Storage. Dataflows leverage a standard structure to store and describe data created in the lake, which is called Common Data Model folders. In this article, you'll learn more about the storage standard that dataflows use behind the scenes.

Storage needs a structure for an analytical dataflow

If the [dataflow is standard](#), then the data is stored in Dataverse. Dataverse is like a database system; it has the concept of tables, views, and so on. Dataverse is a structured data storage option used by standard dataflows.

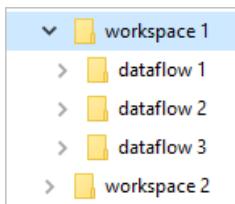
However, when the dataflow is [analytical](#), the data is stored in Azure Data Lake Storage. A dataflow's data and metadata is stored in a Common Data Model folder. Since a storage account might have multiple dataflows stored in it, a hierarchy of folders and subfolders has been introduced to help organize the data. Depending on the product the dataflow was created in, the folders and subfolders may represent workspaces (or environments), and then the dataflow's Common Data Model folder. Inside the Common Data Model folder, both schema and data of the dataflow entities are stored. This structure follows the standards defined for Common Data Model.



What is the Common Data Model storage structure?

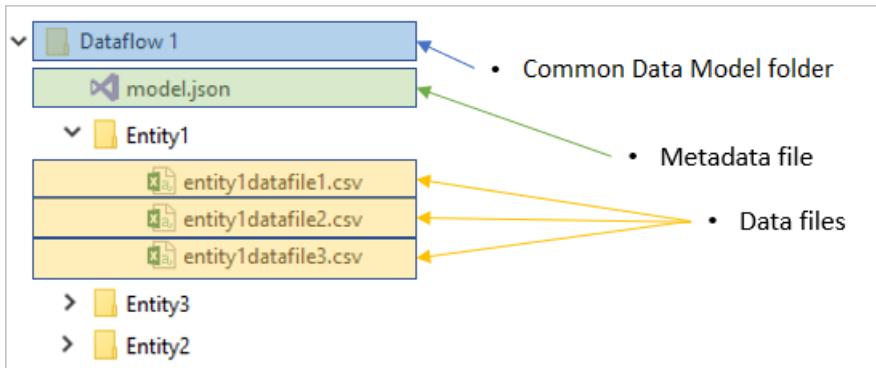
[Common Data Model](#) is a metadata structure defined to bring conformity and consistency for using data across multiple platforms. Common Data Model isn't data storage, it's the way that data is stored and defined.

Common Data Model folders define how an entity's schema and its data should be stored. In Azure Data Lake Storage, data is organized in folders. Folders can represent a workspace or environment. Under those folders, subfolders for each dataflow are created.



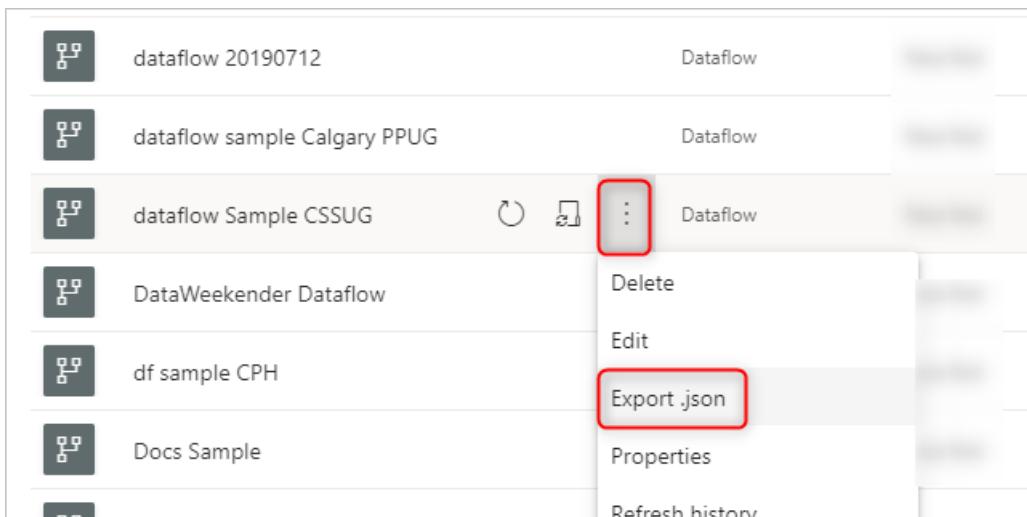
What's in a dataflow folder?

Each dataflow folder contains a subfolder for each entity and a metadata file named `model.json`.

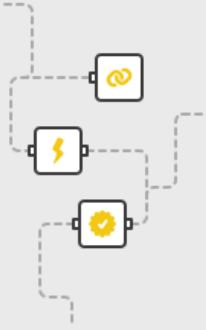


The metadata file: `model.json`

The `model.json` file is the metadata definition of the dataflow. This is the one file that contains all the dataflow metadata. It includes a list of entities, the columns, and their data types in each entity, the relationship between entities, and so on. You can export this file from a dataflow easily, even if you don't have access to the Common Data Model folder structure.



You can use this JSON file to migrate (or import) your dataflow into another workspace or environment.



Start creating your dataflow

Define new entities

Choose a data source to define the entities for your dataflow. You can map your data to [standard Common Data Model](#) entities, or define custom entities instead.

[Learn more](#)

[Add new entities](#)

Link entities from other dataflows

Linking to entities from other dataflows reduces duplication and helps maintain consistency across your organization.

[Learn more](#)

[Add linked entities](#)

Import Model

Choose a dataflow model to import into your workspace.

[Learn more](#)

[Import model](#)

Attach a Common Data Model folder (preview)

Attach a Common Data Model folder from your Azure Data Lake Storage Gen2 account to a new dataflow, so you can use it in Power BI.

[Learn more](#)

[Create and attach](#)

To learn exactly what the model.json metadata file includes, go to [The metadata file \(model.json\) for Common Data Model](#).

Data files

In addition to the metadata file, the dataflow folder includes other subfolders. A dataflow stores the data for each entity in a subfolder with the entity's name. Data for an entity might be split into multiple data partitions, stored in CSV format.

How to see or access Common Data Model folders

If you're using dataflows that use storage provided by the product they were created in, you won't have access to those folders directly. In such cases, getting data from the dataflows requires using the Microsoft Power Platform dataflow connector available in the [Get data](#) experience in the Power BI service, Power Apps, and Dynamics 35 Customer Insights products, or in Power BI Desktop.

Choose data source

All categories File Database Power Platform Azure Online services Other

Data sources

 Access File	 Excel File	 Folder File
 PDF File	 Parquet File	 SharePoint fo File
 XML File	 Amazon Redshift Database	 Google BigQu Database
 Impala Database	 MySQL database Database	 Oracle databa Database
 Power Platform dataflows Power Platform	 SAP BW Application Server Database	 SAP BW Mess Database
 SQL Server database Database	 Snowflake Database	 Teradata data Database
 Azure Data Lake Storage Gen2	 Azure HDInsight Spark	 Azure SQL Da

To learn how dataflows and the internal Data Lake Storage integration work, go to [Dataflows and Azure Data Lake integration \(Preview\)](#).

If your organization enabled dataflows to take advantage of its Data Lake Storage account and was selected as a load target for dataflows, you can still get data from the dataflow by using the Power Platform dataflow connector as mentioned above. But you can also access the dataflow's Common Data Model folder directly through the lake, even outside of Power Platform tools and services. Access to the lake is possible through the Azure portal, Microsoft Azure Storage Explorer, or any other service or experience that supports Azure Data Lake Storage. More information: [Connect Azure Data Lake Storage Gen2 for dataflow storage](#)

Next steps

- [Use the Common Data Model to optimize Azure Data Lake Storage Gen2](#)
- [The metadata file \(model.json\) for the Common Data Model](#)
- [Add a CDM folder to Power BI as a dataflow \(Preview\)](#)
- [Connect Azure Data Lake Storage Gen2 for dataflow storage](#)
- [Dataflows and Azure Data Lake Integration \(Preview\)](#)
- [Configure workspace dataflow settings \(Preview\)](#)

Dataflow storage options

1/15/2022 • 2 minutes to read • [Edit Online](#)

[Standard dataflows](#) always load data into Dataverse tables in an environment. [Analytical dataflows](#) always load data into Azure Data Lake Storage accounts. For both dataflow types, there's no need to provision or manage the storage. Dataflow storage, by default, is provided and managed by products the dataflow is created in.

Analytical dataflows allow an additional storage option: your organizations' Azure Data Lake Storage account. This option enables access to the data created by a dataflow directly through Azure Data Lake Storage interfaces. Providing your own storage account for analytical dataflows enables other Azure or line-of-business applications to leverage the data by connecting to the lake directly.

Dataflows that use built-in storage

By default, analytical dataflows will use the built-in Data Lake Storage; for example, when you create a dataflow in Power BI or Power Apps. Access to the output of this type of dataflow is only possible through the Microsoft Power Platform dataflows connector in Power BI Desktop, or from other dataflows.

Dataflows that use customer-provided storage

Before creating a dataflow that uses your organization's Data Lake Storage account, you must link the environment or workspace the dataflow was created in to your Data Lake Storage account. Depending on which product dataflow you're using (Power BI or Power Platform dataflows), the settings for connecting to an external Data Lake Storage subscription is linked in different places.

Linking Power BI to your organization's Azure Data Lake Storage

To configure Power BI dataflows to store data in your organization's Data Lake Storage, you need to follow the steps described in [Connect Azure Data Lake Storage Gen2 for dataflow storage](#) in the Power BI admin portal.

Linking a Power Platform environment to your organization's Azure Data Lake Storage

To configure dataflows created in Power Apps to store data in your organization's Azure Data Lake Storage, follow the steps in [Connect Azure Data Lake Storage Gen2 for dataflow storage](#) in Power Apps.

Known limitations

- After a dataflow is created, its storage location can't be changed.
- Linked and computed entities features are only available when both dataflows are in the same storage account.

The enhanced compute engine

In Power BI, in addition to the standard dataflow engine, an enhanced compute engine is available for the dataflows created in Power BI Premium workspaces. You can configure this setting in the Power BI admin portal, under the Premium capacity settings. The enhanced compute engine is available in Premium P1 or A3 capacities and above. The enhanced compute engine reduces the refresh time required for long-running extract, transform, load (ETL) steps over computed entities, such as joins, distinct, filters, and group by. It also provides the ability to perform DirectQuery over entities from the Power BI dataset. More information: [The enhanced compute engine](#)

Next steps

The articles below provide further information that can be helpful.

- [Connect Azure Data Lake Storage Gen2 for dataflow storage \(Power BI dataflows\)](#)
- [Connect Azure Data Lake Storage Gen2 for dataflow storage \(Power Platform dataflows\) -->](#)
- [Creating computed entities in dataflows](#)
- [The enhanced compute engine](#)
- [Understanding the differences between standard and analytical dataflows](#)

Computed entity scenarios and use cases

1/15/2022 • 5 minutes to read • [Edit Online](#)

There are benefits to using [computed entities](#) in a dataflow. This article describes use cases for computed entities and describes how they work behind the scenes.

What is a computed entity?

An entity represents the data output of a query created in a dataflow, after the dataflow has been refreshed. It represents data from a source and, optionally, the transformations that were applied to it. Sometimes, you might want to create new entities that are a function of a previously ingested entity.

Although it's possible to repeat the queries that created an entity and apply new transformations to them, this approach has drawbacks: data is ingested twice, and the load on the data source is doubled.

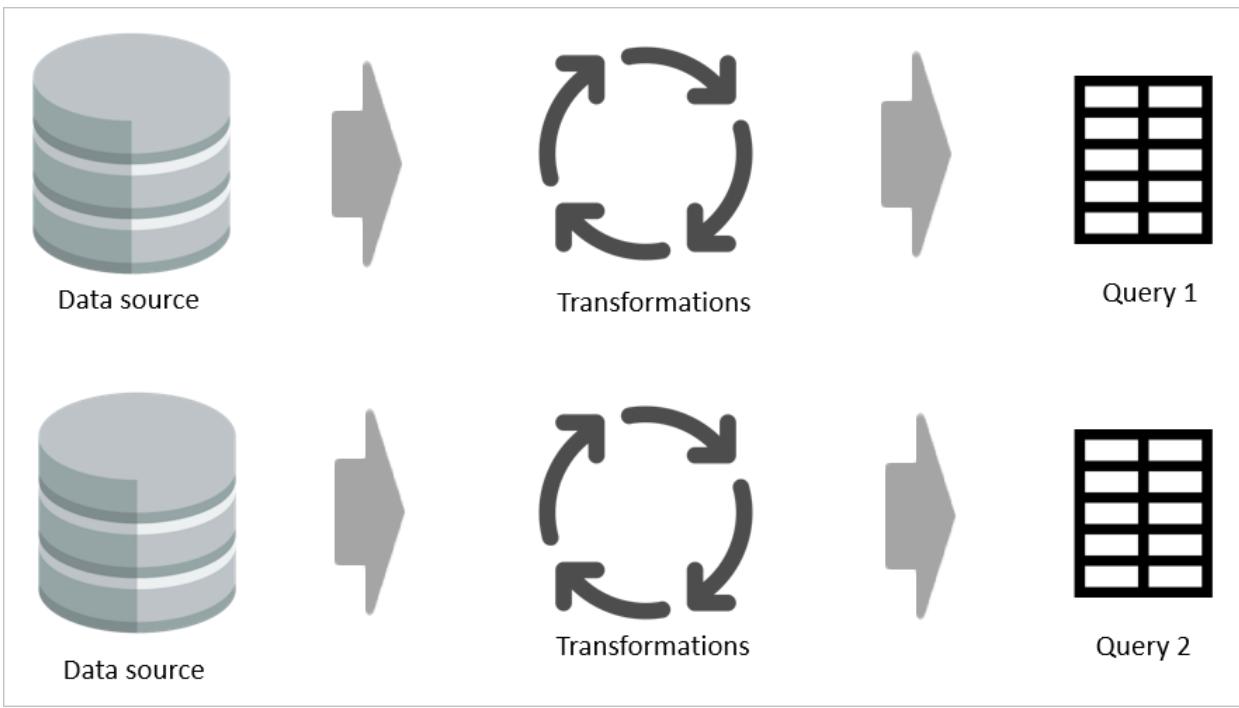
Computed entities solve both problems. Computed entities are similar to other entities in that they get data from a source and you can apply further transformations to create them. But their data originates from the storage dataflow used, and not the original data source. That is, they were previously created by a dataflow and then reused.

Computed entities can be created by referencing an entity in the same dataflow or by referencing an entity created in a different dataflow.

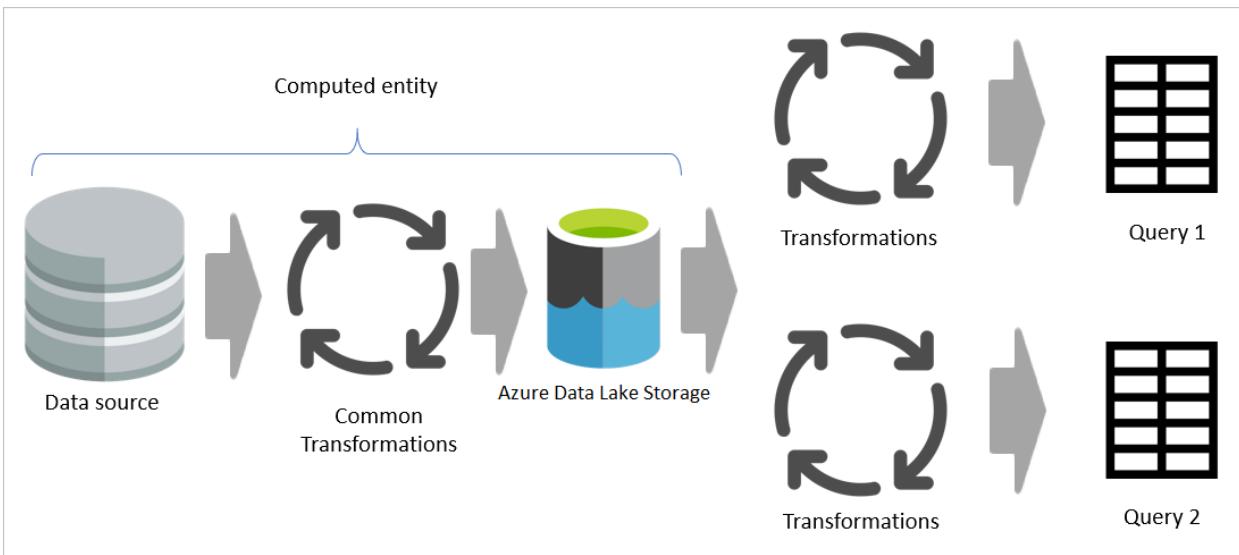
Why use a computed entity?

Performing all transformation steps in one entity can be slow. There can be many reasons for this slowdown—the data source might be slow, or the transformations that you're doing might need to be replicated in two or more queries. It might be advantageous to first ingest the data from the source and then reuse it in one or more entities. In such cases, you might choose to create two entities: one that gets data from the data source, and another—a computed entity—that applies additional transformations to data already written into the data lake used by a dataflow. This can increase performance and reusability of data, saving time and resources.

For example, if two entities share even a part of their transformation logic, without a computed entity the transformation will have to be done twice.



However, if a computed entity is used, then the common (shared) part of the transformation will be processed once and stored in Azure Data Lake Storage. The remaining transformations will then be processed from the output of the common transformation. Overall, this processing is much faster.



A computed entity provides one place as the source code for the transformation and speeds up the transformation because it need only be done once instead of multiple times. The load on the data source is also reduced.

Example scenario for using a computed entity

If you're building an aggregated table in Power BI to speed up the data model, you can build the aggregated table by referencing the original table and applying additional transformations to it. By using this approach, you don't need to replicate your transformation from the source (the part that is from the original table).

For example, the following figure shows an Orders entity.

The screenshot shows the Power Query - Edit queries interface with the Home tab selected. The ribbon at the top includes Home, Transform, Add column, View, Get data, Enter data, Options, Manage parameters, Refresh, Advanced editor, Properties, Choose columns, Remove columns, Keep rows, Remove rows, Sort, Data type, Split column, Group by, Replace values, Merge queries, Append queries, Combine files, and Combine. The main area displays a table of Orders data with columns: OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight, and ShipName. The first row is highlighted.

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
10248	VINET	5	7/4/1996, 12:00:00 AM	8/1/1996, 12:00:00 AM	7/16/1996, 12:00:00 AM	3	32.86	Vins
10249	TOMSP	6	7/5/1996, 12:00:00 AM	8/16/1996, 12:00:00 AM	7/10/1996, 12:00:00 AM	1	11.61	Toms
10250	HANAR	4	7/8/1996, 12:00:00 AM	8/5/1996, 12:00:00 AM	7/12/1996, 12:00:00 AM	2	65.83	Hanar
10251	VICTE	3	7/8/1996, 12:00:00 AM	8/5/1996, 12:00:00 AM	7/15/1996, 12:00:00 AM	1	41.34	Victua
10252	SUPRD	4	7/9/1996, 12:00:00 AM	8/6/1996, 12:00:00 AM	7/11/1996, 12:00:00 AM	2	51.3	Suprén
10253	HANAR	3	7/10/1996, 12:00:00 AM	7/24/1996, 12:00:00 AM	7/16/1996, 12:00:00 AM	2	58.17	Hanar
10254	CHOPS	5	7/11/1996, 12:00:00 AM	8/8/1996, 12:00:00 AM	7/23/1996, 12:00:00 AM	2	22.98	Chop-
10255	RICSU	9	7/12/1996, 12:00:00 AM	8/9/1996, 12:00:00 AM	7/15/1996, 12:00:00 AM	3	148.33	Richter
10256	WELLN	3	7/15/1996, 12:00:00 AM	8/12/1996, 12:00:00 AM	7/17/1996, 12:00:00 AM	2	13.97	Weilln
10257	HILAA	4	7/16/1996, 12:00:00 AM	8/13/1996, 12:00:00 AM	7/22/1996, 12:00:00 AM	3	81.91	HILAR
10258	ERNSH	1	7/17/1996, 12:00:00 AM	8/14/1996, 12:00:00 AM	7/23/1996, 12:00:00 AM	1	140.51	Ernst
10259	CENTC	4	7/18/1996, 12:00:00 AM	8/15/1996, 12:00:00 AM	7/25/1996, 12:00:00 AM	3	3.25	Centro
10260	OTTIK	4	7/19/1996, 12:00:00 AM	8/16/1996, 12:00:00 AM	7/29/1996, 12:00:00 AM	1	55.09	Ottile
10261	QUEDE	4	7/19/1996, 12:00:00 AM	8/16/1996, 12:00:00 AM	7/30/1996, 12:00:00 AM	2	3.05	Que D
10262	RATTG	8	7/22/1996, 12:00:00 AM	8/19/1996, 12:00:00 AM	7/25/1996, 12:00:00 AM	3	48.29	Rattig
10263	ERNSH	9	7/23/1996, 12:00:00 AM	8/20/1996, 12:00:00 AM	7/31/1996, 12:00:00 AM	3	146.06	Ernst

Using a reference from this entity, you can build a computed entity.

The screenshot shows the Power BI Query Editor interface. In the top-left corner, there's a 'Queries' section with two items: 'Orders' and 'Orders aggregated'. The 'Orders' item is highlighted with a red box and has a red circle with the number '1' above it. Below the queries, there's a context menu with several options: 'Paste' (with a clipboard icon), 'Delete' (with a red X icon), 'Rename' (with a document icon), 'Enable load' (with a checkmark icon), 'Duplicate' (with a clipboard icon), and 'Reference' (with a circular arrow icon). The 'Reference' option is highlighted with a red box and has a red circle with the number '2' below it. The entire interface is set against a light gray background.

Image showing how to create a computed entity from the Orders entity. First right-click the Orders entity in the Queries pane, select the Reference option from the drop-down menu, which creates the computed entity, which is renamed here to Orders aggregated.

The computed entity can have further transformations. For example, you can use **Group By** to aggregate the data at the customer level.

New query | Options | Parameters | Query

Queries <

Orders

Orders aggregated

① Computed entities require Premium to re

	A ^B C F	Custo... D G	1 ² 3 Count H I
1	VINET		5
2	TOMSP		6
3	HANAR		14
4	VICTE		10
5	SUPRD		12
6	CHOPS		8
7	RICSU		10
8	WELLI		9
9	HILAA		18
10	ERNSH		30
11	CENTC		1
12	OTTIK		10
13	QUEDE		9
14	PATTI		18

This means that the Orders Aggregated entity will be getting data from the Orders entity, and not from the data source again. Because some of the transformations that need to be done have already been done in the Orders

entity, performance is better and data transformation is faster.

Computed entity in other dataflows

You can also create a computed entity in other dataflows. It can be created by getting data from a dataflow with the Microsoft Power Platform dataflow connector.

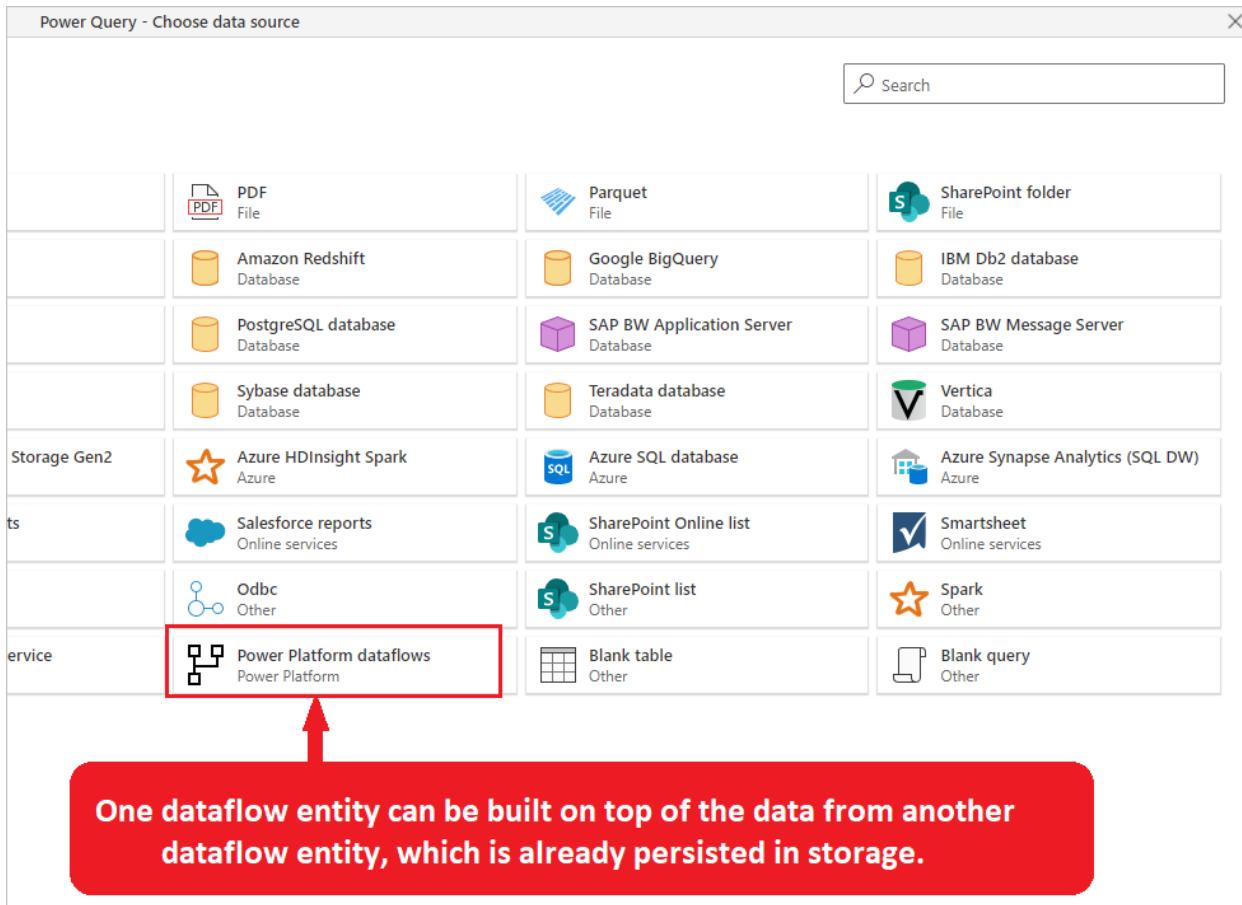


Image emphasizes the Power Platform dataflows connector from the Power Query choos data source window, with a description that states that one dataflow entity can be built on top of the data from another dataflow entity, which is already persisted in storage.

The concept of the computed entity is to have a table persisted in storage, and other tables sourced from it, so that you can reduce the read time from the data source and share some of the common transformations. This can be achieved by getting data from other dataflows through the dataflow connector or referencing another query in the same dataflow.

Computed entity: With transformations, or without?

Now that you know computed entities are great for improving performance of the data transformation, a good question to ask is whether transformations should always be deferred to the computed entity or whether they should be applied to the source entity. That is, should data always be ingested into one entity and then transformed in a computed entity? What are the pros and cons?

Load data without transformation for Text/CSV files

When a data source doesn't support query folding (such as Text/CSV files), there's little benefit in applying transformations when getting data from the source, especially if data volumes are large. The source entity should just load data from the Text/CSV file without applying any transformations. Then, computed entities can get data from the source entity and perform the transformation on top of the ingested data.

You might ask, what's the value of creating a source entity that only ingests data? Such an entity can still be

useful, because if the data from the source is used in more than one entity, it reduces the load on the data source. In addition, data can now be reused by other people and dataflows. Computed entities are especially useful in scenarios where the data volume is large, or when a data source is accessed through an on-premises data gateway, because they reduce the traffic from the gateway and the load on data sources behind them.

Doing some of the common transformations for a SQL table

If your data source supports query folding, it's good to perform some of the transformations in the source entity because the query will be folded to the data source, and only the transformed data will be fetched from it. This improves overall performance. The set of transformations that will be common in downstream computed entities should be applied in the source entity, so they can be folded to the source. Other transformations that only apply to downstream entities should be done in computed entities.

Best practices for designing and developing complex dataflows

1/15/2022 • 4 minutes to read • [Edit Online](#)

If the dataflow you're developing is getting bigger and more complex, here are some things you can do to improve on your original design.

Break it into multiple dataflows

Don't do everything in one dataflow. Not only does a single, complex dataflow make the data transformation process longer, it also makes it harder to understand and reuse the dataflow. Breaking your dataflow into multiple dataflows can be done by separating entities in different dataflows, or even one entity into multiple dataflows. You can use the concept of a computed entity or linked entity to build part of the transformation in one dataflow, and reuse it in other dataflows.

Split data transformation dataflows from staging/extraction dataflows

Having some dataflows just for extracting data (that is, [staging dataflows](#)) and others just for transforming data is helpful not only for creating a multilayered architecture, it's also helpful for reducing the complexity of dataflows. Some steps just extract data from the data source, such as get data, navigation, and data type changes. By separating the staging dataflows and transformation dataflows, you make your dataflows simpler to develop.

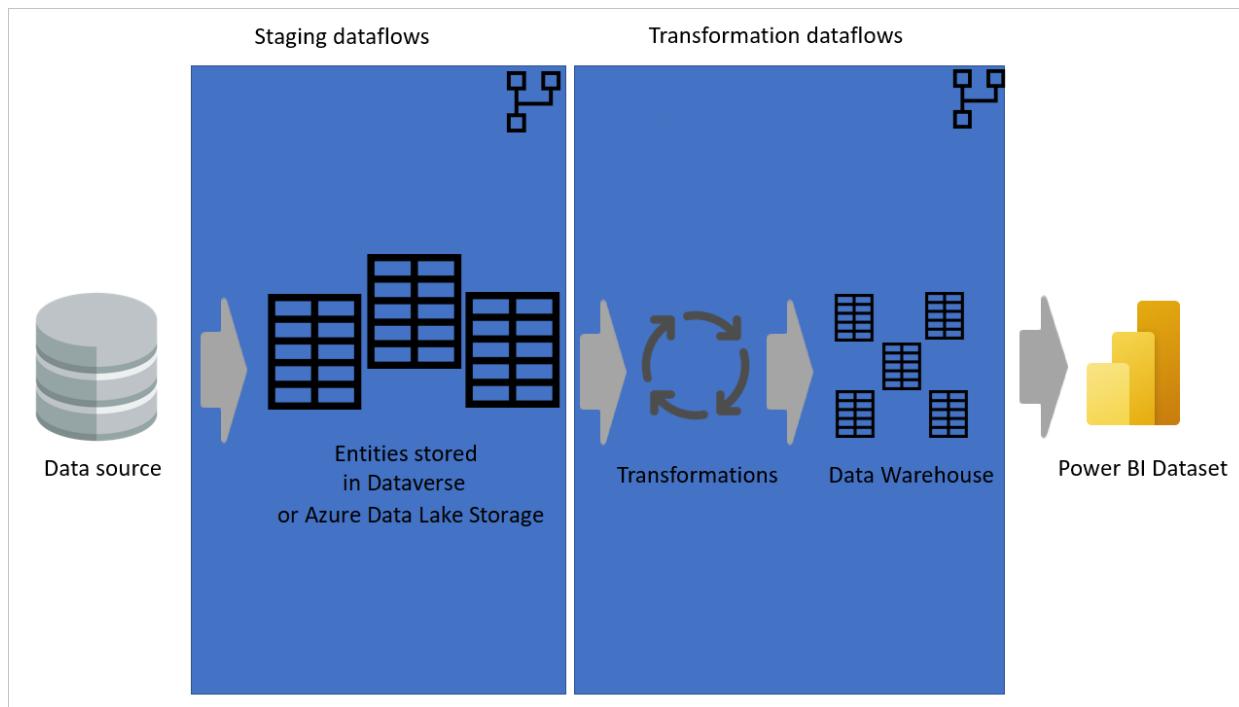


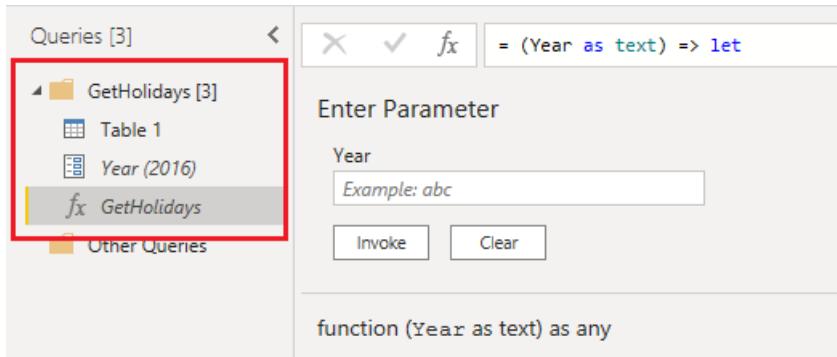
Image showing data being extracted from a data source to staging dataflows, where the entities are either stored in Dataverse or Azure Data Lake storage, then the data is moved to transformation dataflows where the data is transformed and converted to the data warehouse structure, and then the data is moved to the dataset.

Use custom functions

Custom functions are helpful in scenarios where a certain number of steps have to be done for a number of

queries from different sources. Custom functions can be developed through the graphical interface in Power Query Editor or by using an M script. Functions can be reused in a dataflow in as many entities as needed.

Having a custom function helps by having only a single version of the source code, so you don't have to duplicate the code. As a result, maintaining the Power Query transformation logic and the whole dataflow will be much easier. For more information, see the following blog post: [Custom Functions Made Easy in Power BI Desktop](#).



Place queries into folders

Using folders for queries helps to group related queries together. When developing the dataflow, spend a little more time to arrange queries in folders that make sense. Using this approach, you can find queries more easily in the future and maintaining the code will be much easier.

Use computed entities

Computed entities not only make your dataflow more understandable, they also provide better performance. When you use a computed entity, the other entities referenced from it are getting data from an "already-processed-and-stored" entity. The transformation will be much simpler and faster.

Take advantage of the enhanced compute engine

For dataflows developed in Power BI admin portal, ensure that you make use of the enhanced compute engine by performing joins and filter transformations first in a computed entity before doing other types of transformations.

Break many steps into multiple queries

It's hard to keep track of a large number of steps in one entity. Instead, you should break a large number of steps into multiple entities. You can use **Enable Load** for other queries and disable them if they're intermediate queries, and only load the final entity through the dataflow. When you have multiple queries with smaller steps in each, it's easier to use the dependency diagram and track each query for further investigation, rather than digging into hundreds of steps in one query.

Add properties for queries and steps

Documentation is the key to having easy-to-maintain code. In Power Query, you can add properties to the entities and also to steps. The text that you add in the properties will show up as a tooltip when you hover over that query or step. This documentation will help you maintain your model in the future. With a glance at a table or step, you can understand what's happening there, rather than rethinking and remembering what you've done in that step.

Ensure that capacity is in the same region

Dataflows don't currently support multiple countries or regions. The Premium capacity must be in the same region as your Power BI tenant.

Separate on-premises sources from cloud sources

We recommend that you create a separate dataflow for each type of source, such as on-premises, cloud, SQL Server, Spark, and Dynamics 365. Separating dataflows by source type facilitates quick troubleshooting and avoids internal limits when you refresh your dataflows.

Separate dataflows based on the scheduled refresh required for entities

If you have a sales transaction table that gets updated in the source system every hour and you have a product-mapping table that gets updated every week, break these two into two dataflows with different data refresh schedules.

Avoid scheduling refresh for linked entities in the same workspace

If you're regularly being locked out of your dataflows that contain linked entities, it might be caused by a corresponding, dependent dataflow in the same workspace that's locked during dataflow refresh. Such locking provides transactional accuracy and ensures that both dataflows are successfully refreshed, but it can block you from editing.

If you set up a separate schedule for the linked dataflow, dataflows can be refreshed unnecessarily and block you from editing the dataflow. There are two recommendations to avoid this:

- Don't set a refresh schedule for a linked dataflow in the same workspace as the source dataflow.
- If you want to configure a refresh schedule separately and want to avoid the locking behavior, move the dataflow to a separate workspace.

Best practices for reusing dataflows across environments and workspaces

1/15/2022 • 3 minutes to read • [Edit Online](#)

This article discusses a collection of best practices for reusing dataflows effectively and efficiently. Read this article to avoid design pitfalls and potential performance issues as you develop dataflows for reuse.

Separate data transformation dataflows from staging/extraction dataflows

If a dataflow performs all the actions, it's hard to reuse its entities in other dataflows or for other purposes. The best dataflows to reuse are those dataflows that do only a few actions. Creating dataflows that specialize in one specific task is one of the best ways to reuse them. If you have a set of dataflows that you use as [staging dataflows](#), their only action is to extract data as-is from the source system. These dataflows can be reused in multiple other dataflows.

If you have data transformation dataflows, you can split them into dataflows that do common transformations. Each dataflow can do just a few actions. These few actions per dataflow ensure that the output of that dataflow is reusable by other dataflows.

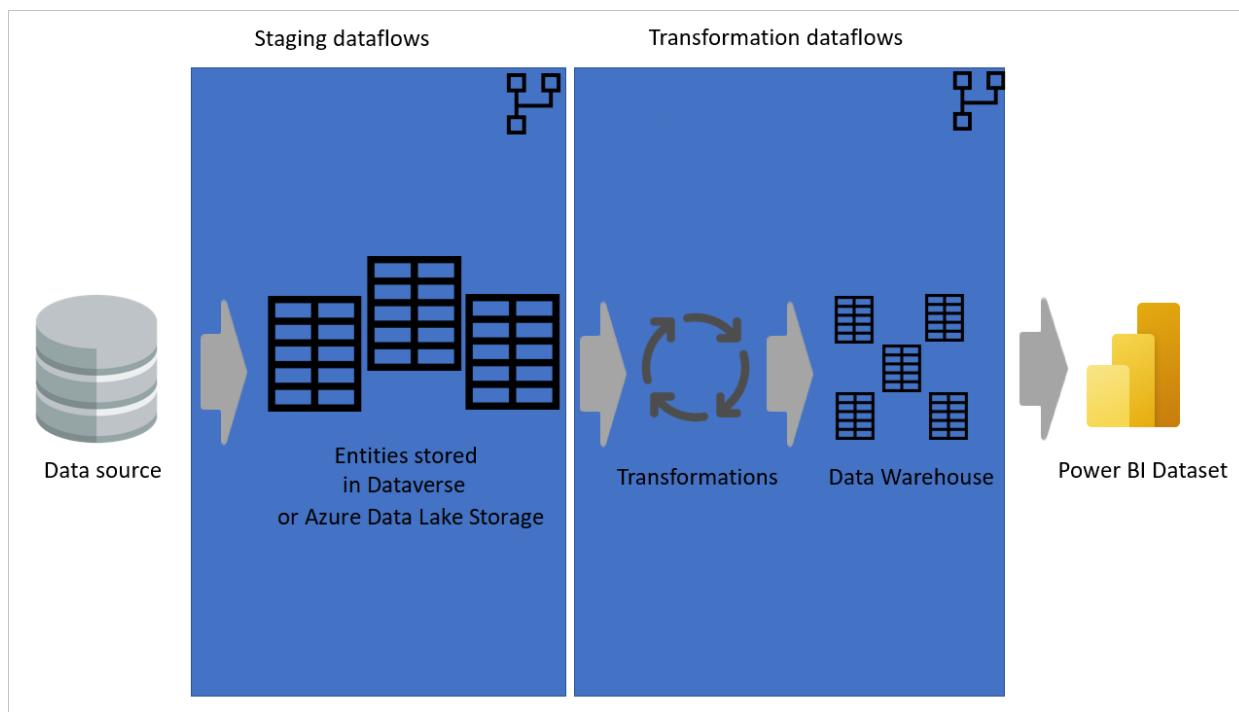
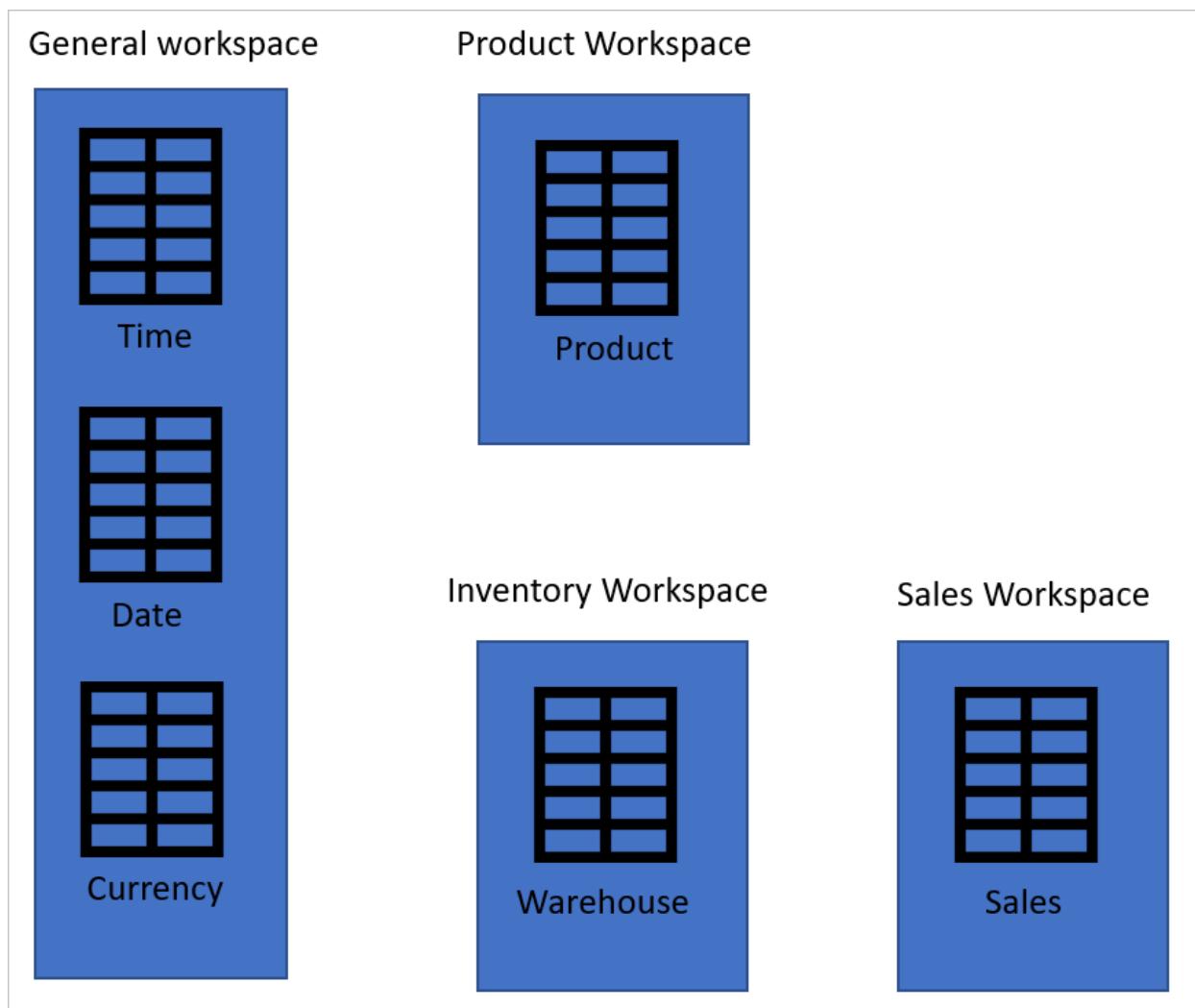


Image with data being extracted from a data source to staging dataflows, where the entities are either stored in Dataverse or Azure Data Lake storage, then the data is moved to transformation dataflows where the data is transformed and converted to the data warehouse structure, and then the data is loaded to a Power BI dataset.

Use multiple workspaces

Each workspace (or environment) is available only for members of that workspace. If you build all your dataflows in one workspace, you're minimizing the reuse of your dataflows. You can have some generic workspaces for dataflows that are processing company-wide entities. You can also have some workspace for dataflows to process entities across multiple departments. And you can also have some workspaces for

dataflows to be used only in specific departments.



Set the correct access levels on workspaces

To give access to dataflows in other workspaces to use the output of a dataflow in a workspace, you just need to give them View access in the workspace. To learn more about other roles in a Power BI workspace, go to [Roles in the new workspaces](#).

Endorsement on the dataflow in Power BI

There can be many dataflows created in a tenant organization, and it can be hard for the users to know which dataflow is most reliable. Authors of a dataflow, or those who have edit access to it, can endorse the dataflow at three levels: no endorsement, promoted, or certified.

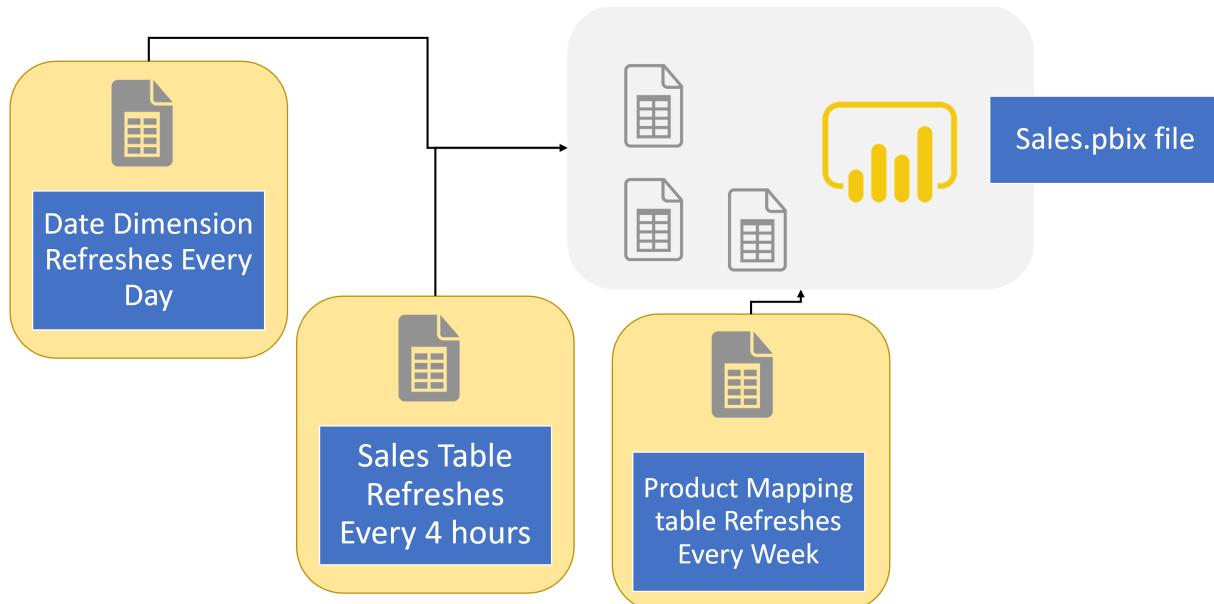
These levels of endorsement help users find reliable dataflows easier and faster. The dataflow with a higher endorsement level appears first. The Power BI administrator can delegate the ability to endorse dataflows to the certified level to other people. More information: [Endorsement - Promoting and certifying Power BI content](#)

Separate entities in multiple dataflows

You can have multiple entities in one dataflow. One of the reasons you might split entities in multiple dataflows is what you learned earlier in this article about separating the data ingestion and data transformation dataflows. Another good reason to have entities in multiple dataflows is when you want a different refresh schedule than

other tables.

In the example shown in the following image, the sales table needs to be refreshed every four hours. The date table needs to be refreshed only once a day to keep the current date record updated. And a product-mapping table just needs to be refreshed once a week. If you have all of these tables in one dataflow, you have only one refresh option for them all. However, if you split these tables into multiple dataflows, you can schedule the refresh of each dataflow separately.



Multiple Dataflows can have tables transformed with multiple schedule options

Good table candidates for dataflow entities

When you develop solutions using Power Query in the desktop tools, you might ask yourself: which of these tables are good candidates to be moved to a dataflow? The best tables to be moved to the dataflow are those that need to be used in more than one solution, or more than one environment or service. For example, the Date table shown in the following image needs to be used in two separate Power BI files. Instead of duplicating that table in each file, you can build the table in a dataflow as an entity, and reuse it in those Power BI files.



Date Dimension transformation executed only once, and then used multiple times

Best practices for creating a dimensional model using dataflows

1/15/2022 • 5 minutes to read • [Edit Online](#)

Designing a dimensional model is one of the most common tasks you can do with a dataflow. This article highlights some of the best practices for creating a dimensional model using a dataflow.

Staging dataflows

One of the key points in any data integration system is to reduce the number of reads from the source operational system. In the traditional data integration architecture, this reduction is done by creating a new database called a *staging database*. The purpose of the staging database is to load data as-is from the data source into the staging database on a regular schedule.

The rest of the data integration will then use the staging database as the source for further transformation and converting it to the dimensional model structure.

We recommended that you follow the same approach using dataflows. Create a set of dataflows that are responsible for just loading data as-is from the source system (and only for the tables you need). The result is then stored in the storage structure of the dataflow (either Azure Data Lake Storage or Dataverse). This change ensures that the read operation from the source system is minimal.

Next, you can create other dataflows that source their data from staging dataflows. The benefits of this approach include:

- Reducing the number of read operations from the source system, and reducing the load on the source system as a result.
- Reducing the load on data gateways if an on-premises data source is used.
- Having an intermediate copy of the data for reconciliation purpose, in case the source system data changes.
- Making the transformation dataflows source-independent.

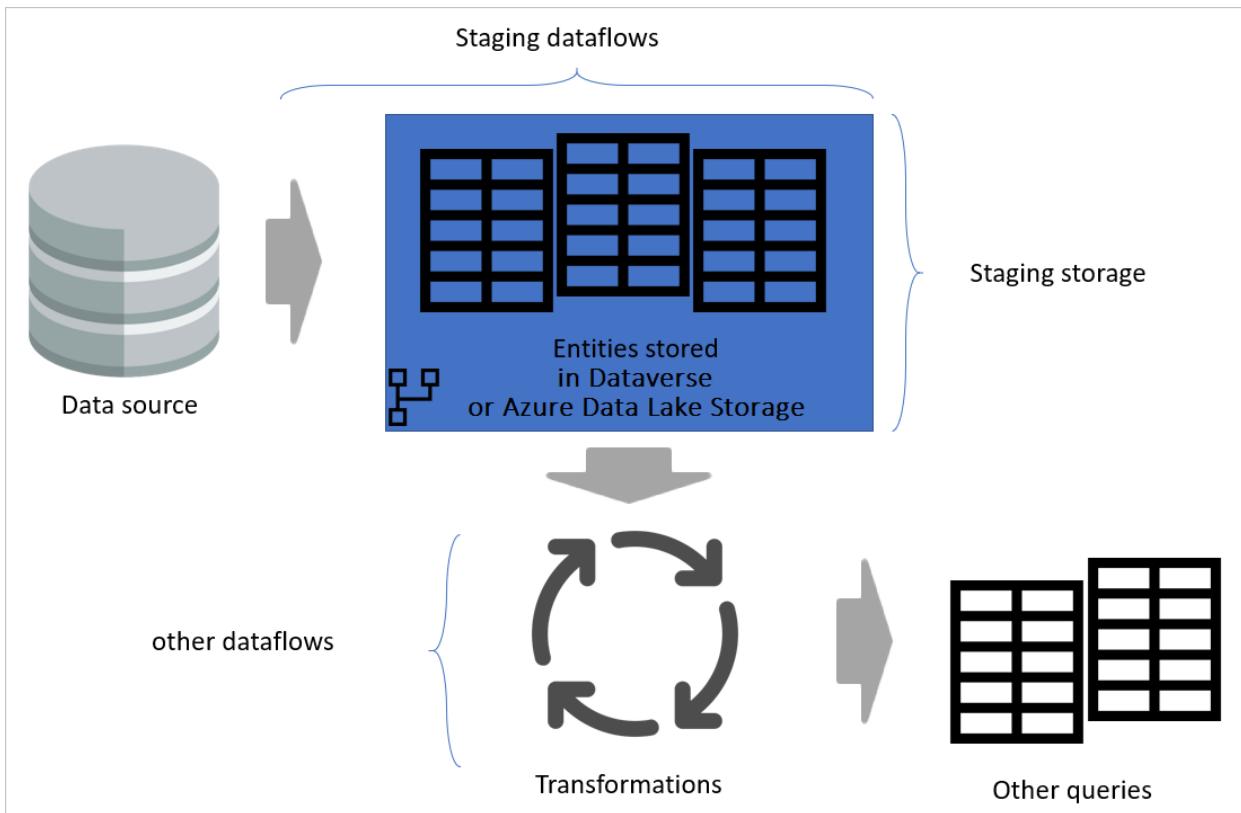
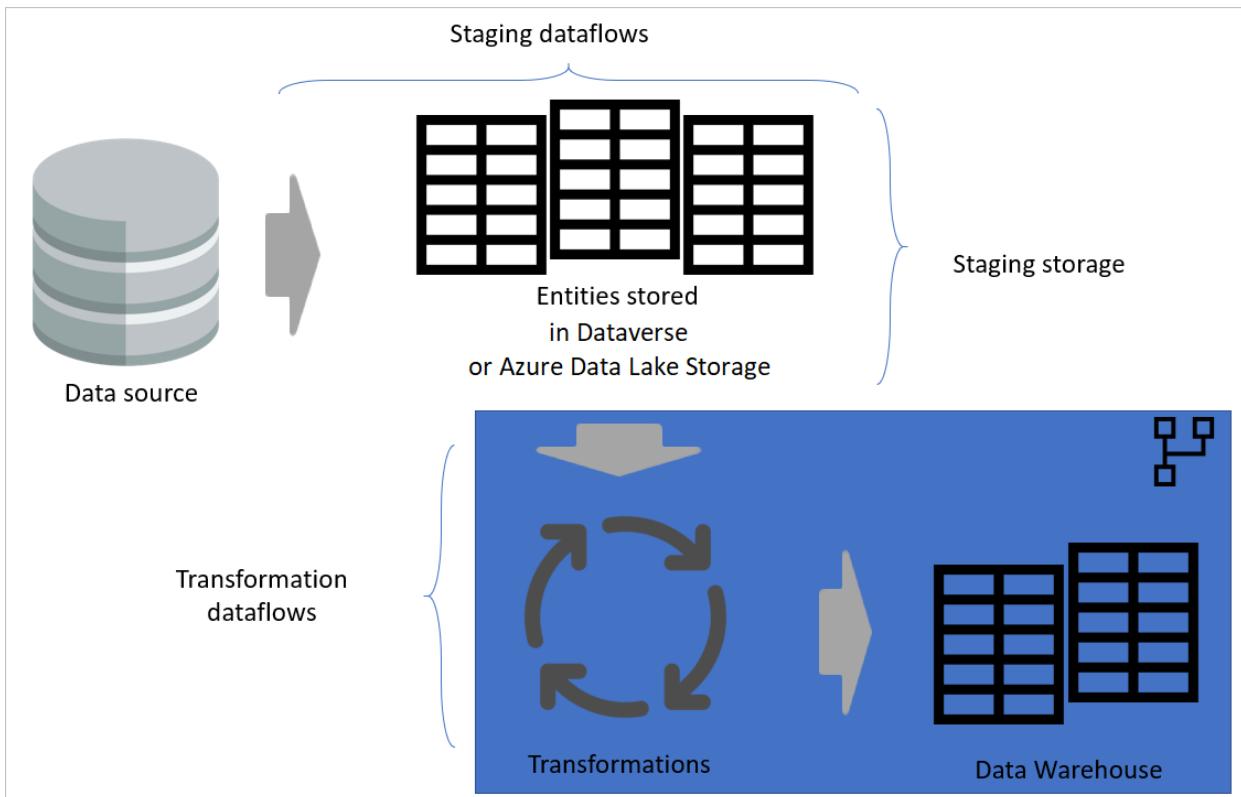


Image emphasizing staging dataflows and staging storage, and showing the data being accessed from the data source by the staging dataflow, and entities being stored in either Cadavers or Azure Data Lake Storage. The entities are then shown being transformed along with other dataflows, which are then sent out as queries.

Transformation dataflows

When you've separated your transformation dataflows from the staging dataflows, the transformation will be independent from the source. This separation helps if you're migrating the source system to a new system. All you need to do in that case is to change the staging dataflows. The transformation dataflows are likely to work without any problem, because they're sourced only from the staging dataflows.

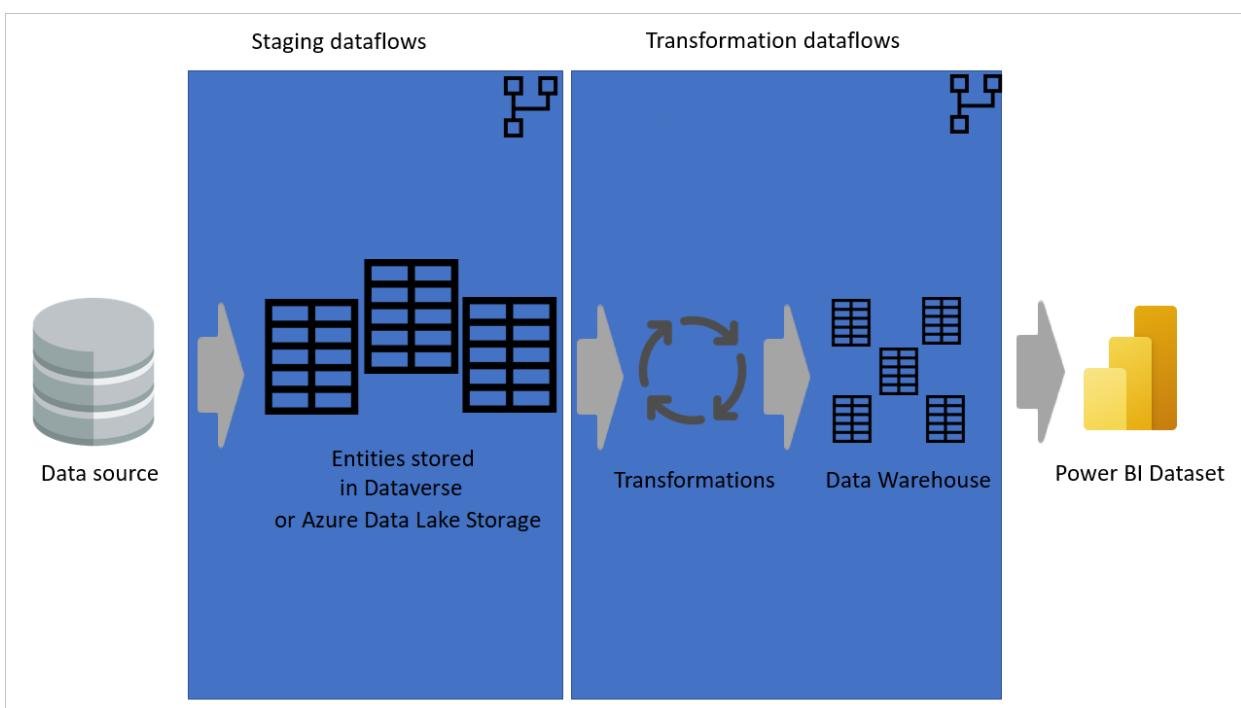
This separation also helps in case the source system connection is slow. The transformation dataflow won't need to wait for a long time to get records coming through a slow connection from the source system. The staging dataflow has already done that part, and the data will be ready for the transformation layer.



Layered Architecture

A layered architecture is an architecture in which you perform actions in separate layers. The staging and transformation dataflows can be two layers of a multi-layered dataflow architecture. Trying to do actions in layers ensures the minimum maintenance required. When you want to change something, you just need to change it in the layer in which it's located. The other layers should all continue to work fine.

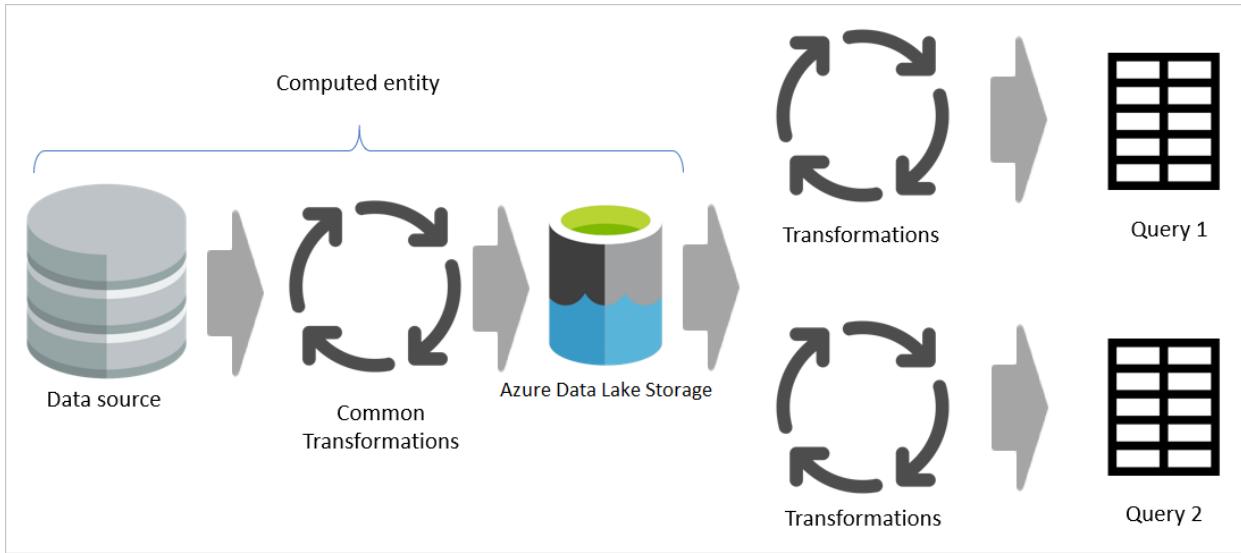
The following image shows a multi-layered architecture for dataflows in which their entities are then used in Power BI datasets.



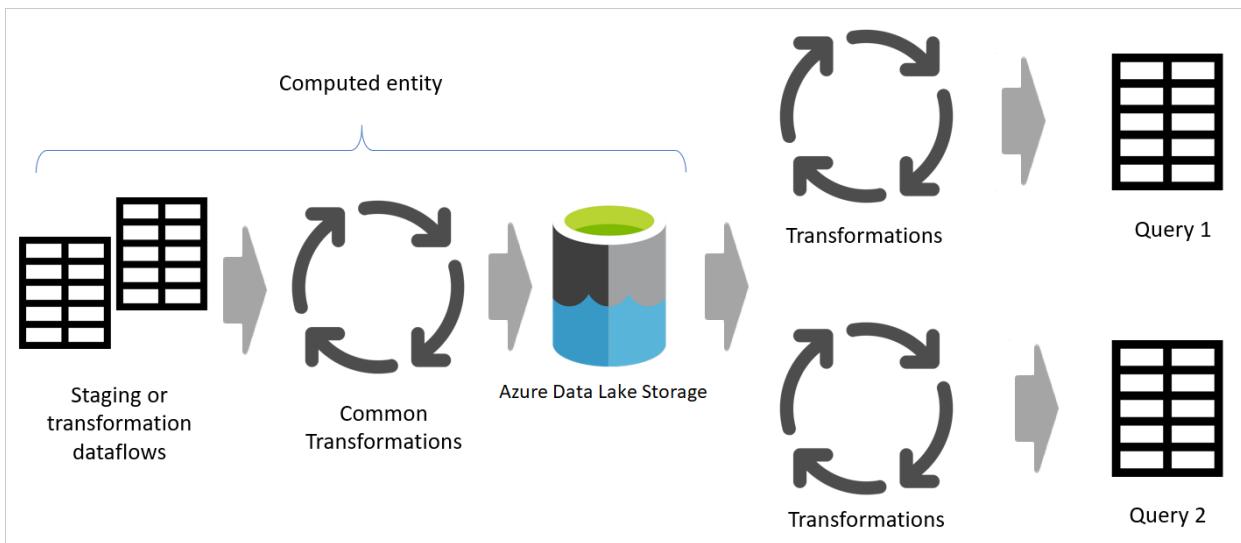
Use a computed entity as much as possible

When you use the result of a dataflow in another dataflow, you're using the concept of the computed entity,

which means getting data from an "already-processed-and-stored" entity. The same thing can happen inside a dataflow. When you reference an entity from another entity, you can use the computed entity. This is helpful when you have a set of transformations that need to be done in multiple entities, which are called *common transformations*.



In the previous image, the computed entity gets the data directly from the source. However, in the architecture of staging and transformation dataflows, it's likely that the computed entities are sourced from the staging dataflows.



Build a star schema

The best dimensional model is a star schema model that has dimensions and fact tables designed in a way to minimize the amount of time to query the data from the model, and also makes it easy to understand for the data visualizer.

It isn't ideal to bring data in the same layout of the operational system into a BI system. The data tables should be remodeled. Some of the tables should take the form of a dimension table, which keeps the descriptive information. Some of the tables should take the form of a fact table, to keep the aggregatable data. The best layout for fact tables and dimension tables to form is a star schema. More information: [Understand star schema and the importance for Power BI](#)

Use a unique key value for dimensions

When building dimension tables, make sure you have a key for each one. This key ensures that there are no

many-to-many (or in other words, "weak") relationships among dimensions. You can create the key by applying some transformation to make sure a column or a combination of columns is returning unique rows in the dimension. Then that combination of columns can be marked as a key in the entity in the dataflow.

The screenshot shows the Power BI Data Flow interface. In the top ribbon, the 'Transform' tab is selected. On the far right of the ribbon, there is a 'Mark as key' button, which is highlighted with a red box. Below the ribbon, a table named 'Date' is displayed. The first column of the table is labeled 'Date'. A red box highlights this column header. The table contains 10 rows of date information from January 1, 2018, to January 10, 2018, along with corresponding year, start of year, end of year, and month values.

Do an incremental refresh for large fact tables

Fact tables are always the largest tables in the dimensional model. We recommend that you reduce the number of rows transferred for these tables. If you have a very large fact table, ensure that you use incremental refresh for that entity. An incremental refresh can be done in the Power BI dataset, and also the dataflow entities.

You can use incremental refresh to refresh only part of the data, the part that has changed. There are multiple options to choose which part of the data to be refreshed and which part to be persisted. More information:

[Using incremental refresh with Power BI dataflows](#)

Referencing to create dimensions and fact tables

In the source system, you often have a table that you use for generating both fact and dimension tables in the data warehouse. These tables are good candidates for computed entities and also intermediate dataflows. The common part of the process—such as data cleaning, and removing extra rows and columns—can be done once. By using a reference from the output of those actions, you can produce the dimension and fact tables. This approach will use the computed entity for the common transformations.

The screenshot shows a context menu for a query named 'Orders aggregated'. The menu includes options like Paste, Delete, Rename, Enable load, Duplicate, Reference, and Move to group. The 'Reference' option is highlighted with a red box and circled with a red number 2. Other menu items like Paste, Delete, and Reference are numbered 1, 3 respectively. The 'Orders' query is also highlighted with a red box and circled with a red number 1.

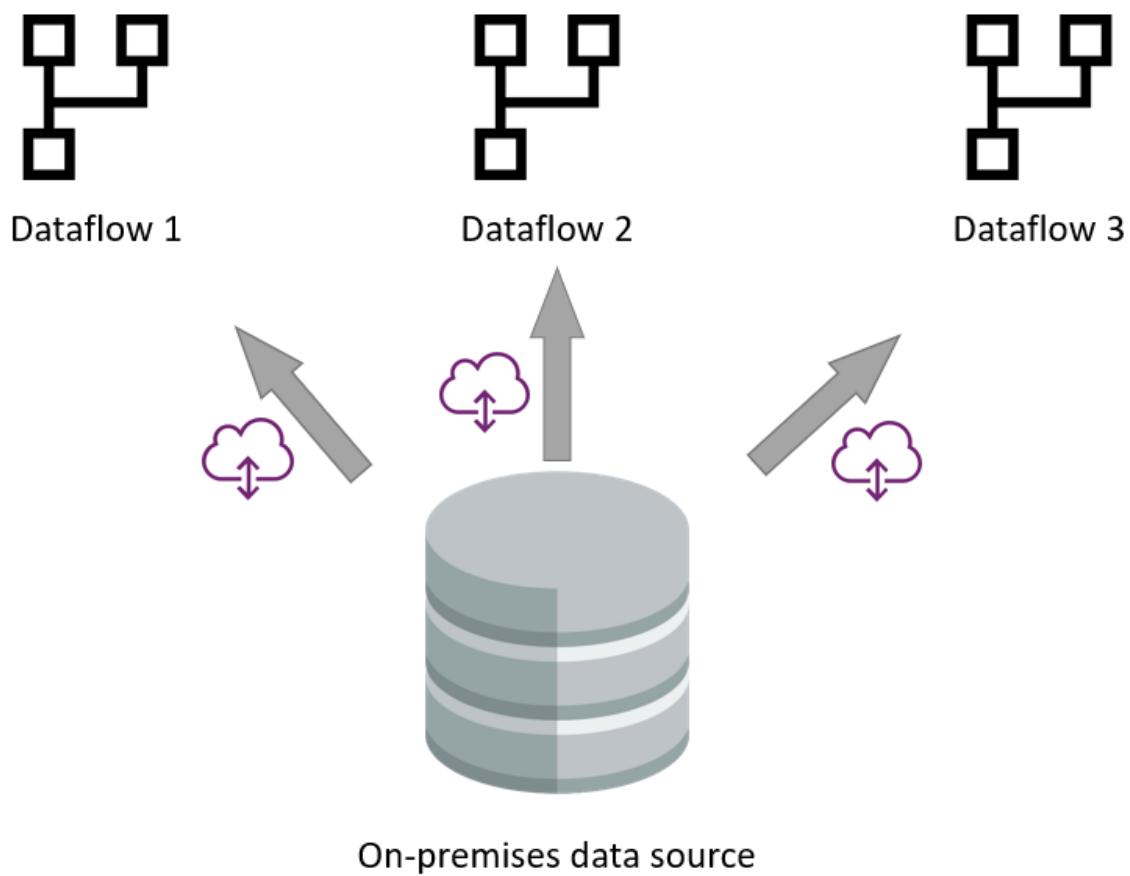
Improve performance and reusability by separating data ingestion from data transformation dataflows

1/15/2022 • 2 minutes to read • [Edit Online](#)

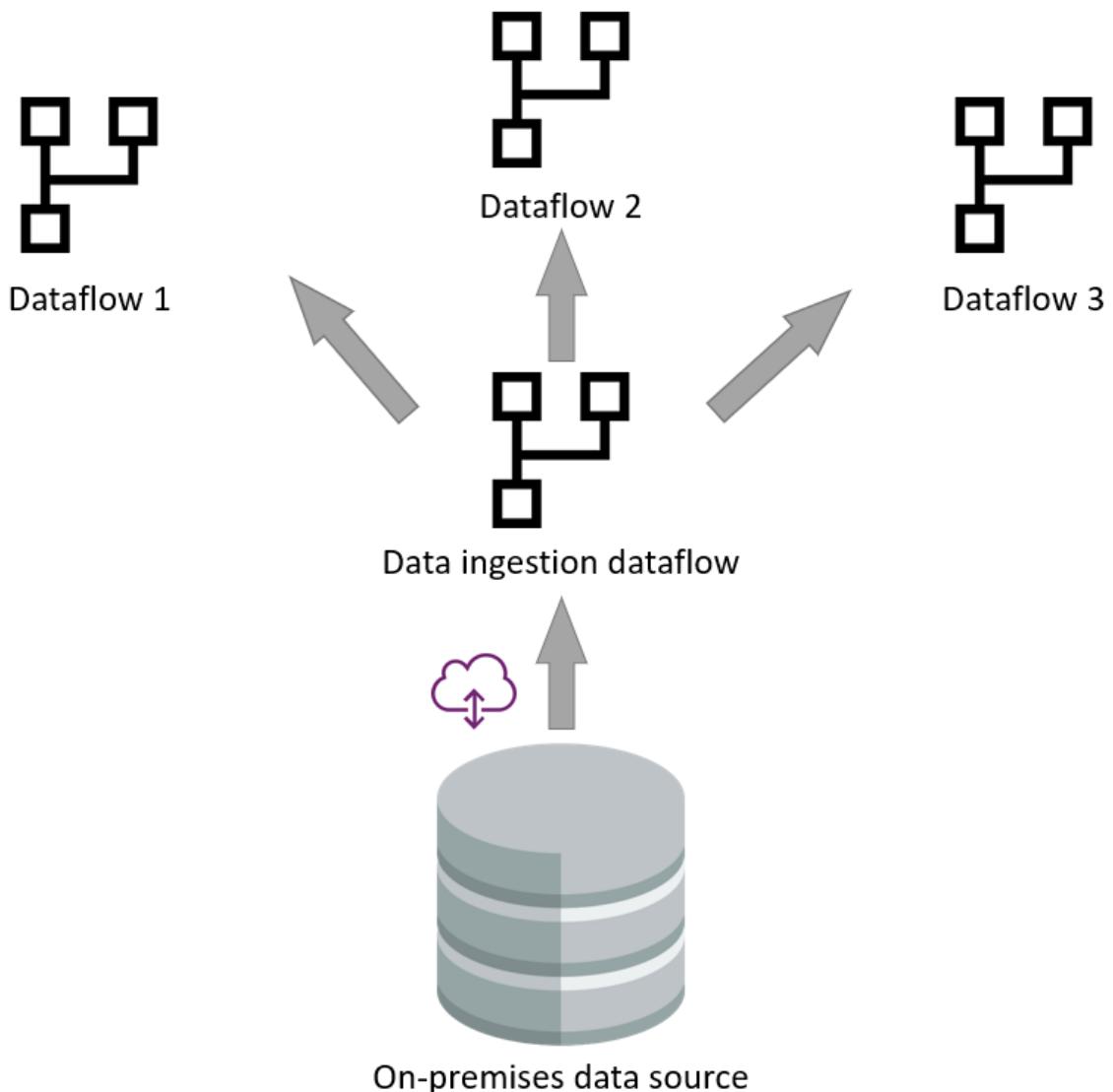
One of the best practices for dataflow implementations is separating the responsibilities of dataflows into two layers: data ingestion and data transformation. This pattern is specifically helpful when dealing with multiple queries of slower data sources in one dataflow, or multiple dataflows querying the same data sources. Instead of getting data from a slow data source again and again for each query, the data ingestion process can be done once, and the transformation can be done on top of that process. This article explains the process.

On-premises data source

In many scenarios, the on-premises data source is a slow data source. Especially considering that the gateway exists as the middle layer between the dataflow and the data source.



Using analytical dataflows for data ingestion minimizes the get data process from the source and focuses on loading data to Azure Data Lake Storage. Once in storage, other dataflows can be created that leverage the ingestion dataflow's output. The dataflow engine can read the data and do the transformations directly from the data lake, without contacting the original data source or gateway.



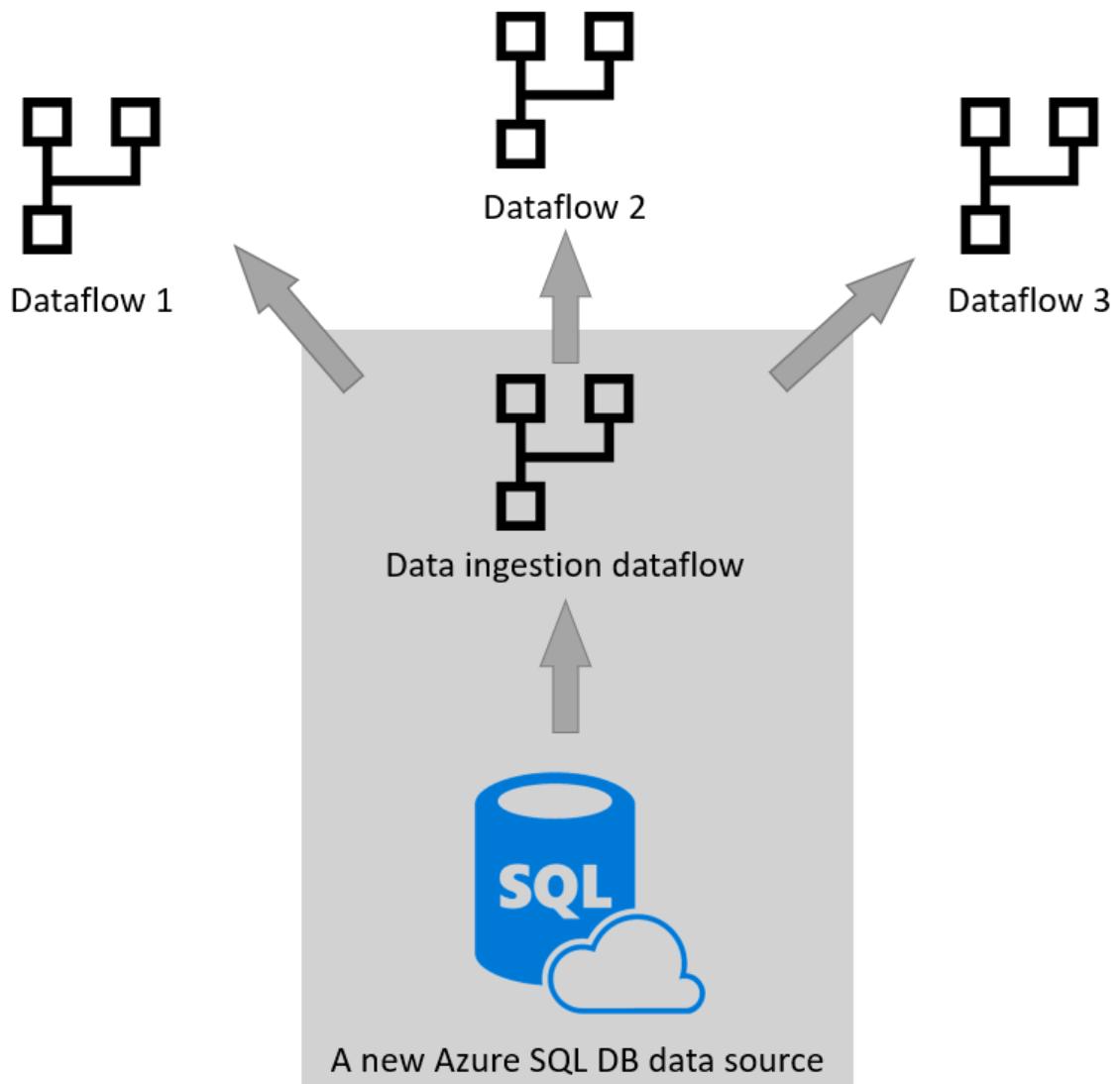
Slow data source

The same process is valid when a data source is slow. Some of the software as a service (SaaS) data sources perform slowly because of the limitations of their API calls.

Separation of the data ingestion and data transformation dataflows

The separation of the two layers—data ingestion and transformation—is helpful in the scenarios where the data source is slow. It helps to minimize the interaction with the data source.

This separation isn't only useful because of the performance improvement, it's also helpful for the scenarios where an old legacy data source system has been migrated to a new system. In those cases, only the data ingestion dataflows need to be changed. The data transformation dataflows remain intact for this type of change.



Reuse in other tools and services

Separation of data ingestion dataflows from data transformation dataflows is helpful in many scenarios. Another use case scenario for this pattern is when you want to use this data in other tools and services. For this purpose, it's better to use analytical dataflows and use your own Data Lake Storage as the storage engine. More information: [Analytical dataflows](#)

Optimize the data ingestion dataflow

Consider optimizing the data ingestion dataflow whenever possible. As an example, if all the data from the source isn't needed, and the data source supports query folding, then filtering data and getting only a required subset is a good approach. To learn more about query folding, go to [Power Query query folding](#).

Create the data ingestion dataflows as analytical dataflows

Consider creating your data ingestion dataflows as analytical dataflows. This especially helps other services and applications to use this data. This also makes it easier for the data transformation dataflows to get data from the analytical ingestion dataflow. To learn more, go to [Analytical dataflows](#).

Using the output of Microsoft Power Platform dataflows from other Azure data workloads

1/15/2022 • 2 minutes to read • [Edit Online](#)

Depending on the storage for the output of the Microsoft Power Platform dataflows, you can use that output in other Azure services.

The benefits of working with the output of Power Platform dataflows

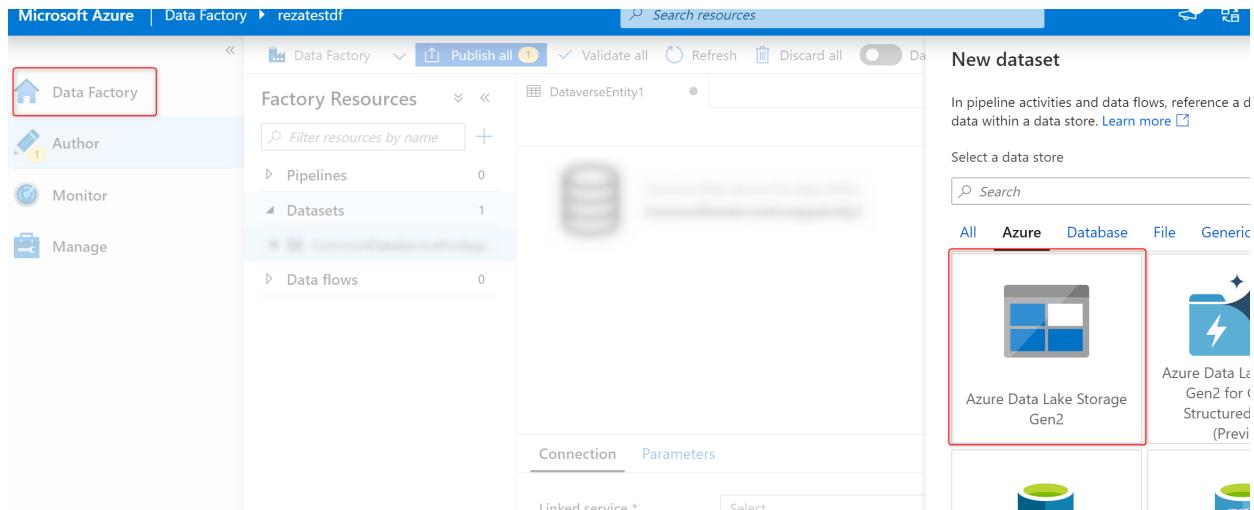
Using Power Platform dataflows, you can reshape, clean, and prepare data for further analysis and consumption. There are many other Azure data services that work with data as an input and provide actions.

- Azure Machine Learning can consume the output of dataflows and use it for machine learning scenarios (for example, predictive analysis).
- Azure Data Factory can get the output of dataflows on a much larger scale, combined with the data from big data sources, for advanced data integration solutions.
- Azure Databricks can consume the output of dataflows for applied data science algorithms and further AI with the big data scale in the Apache Spark back end.
- Other Azure data services can use the output of Power Platform dataflows to do further actions on that data.

Dataflows with external Azure Data Lake Storage

If you've connected an external Azure Data Lake Storage storage to the Power Platform dataflows, you can connect to it using any Azure services that have Azure Data Lake Storage as a source, such as Azure Machine Learning, Azure Data Factory, Azure Databricks, and Azure Analysis Services.

In any of these services, use Azure Data Lake Storage as the source. You'll be able to enter the details of your storage and connect to the data in it. The data is stored in CSV format, and is readable through any of these tools and services. The following screenshot shows how Azure Data Lake Storage is a source option for Azure Data Factory.



Dataflows with Dataverse

If you're using standard dataflows that store the data in Dataverse, you can still connect to Dataverse from many Azure services. The following image shows that in Azure Data Factory, the output of a dataflow from Dataverse

can be used as a source.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a navigation bar with 'Data Factory' (highlighted with a red box), 'Author', 'Monitor', and 'Manage'. The main area is titled 'Factory Resources' and shows a list of resources: Pipelines (0), Datasets (1), and Data flows (0). A dataset named 'DataverseEntity1' is selected, showing its details. To the right, a 'New dataset' dialog is open, titled 'New dataset'. It has a search bar with 'data' and a dropdown menu set to 'Database'. A result for 'Dataverse (Common Data Service for Apps)' is shown, with a red box highlighting the entire result card. Below the search bar, tabs for 'All', 'Azure', 'Database', and 'File' are visible.

Dataflows with internal Azure Data Lake Storage

When you use the internal Data Lake storage that's provided by Power Platform dataflows, that storage is exclusively limited to the Power Platform tools and isn't accessible from other Azure data workloads.

Mapping fields with relationships in standard dataflows

1/15/2022 • 2 minutes to read • [Edit Online](#)

In the standard dataflow, you can easily map fields from the dataflow query into Dataverse tables. However, if the Dataverse table has lookup or relationship fields, additional consideration is required to make sure this process works.

What is the relationship and why do you need that?

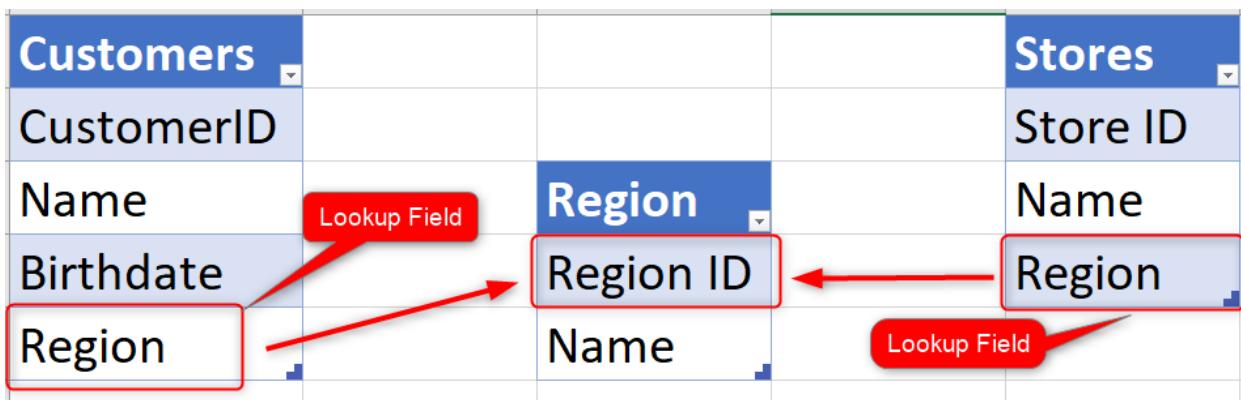
If you're coming from a database development background, you're familiar with the concept of a relationship between tables. However, many users of Microsoft Power Platform services aren't coming from that background. You might wonder what the relationship is, or why you should create a relationship between tables.

The tables and their relationship are fundamental concepts of designing a database. To learn everything about relationships is beyond the scope of this article. However, we'll discuss it in a general way here.

Let's say you want to store information about customers and their details, including region, in Dataverse. You can keep everything in one table. Your table can be called Customers, and it can contain fields, such as CustomerID, Name, Birthdate, and Region. Now imagine that you have another table that also has the store's information. This table can have fields, such as Store ID, Name, and Region. As you can see, the region is repeated in both tables. There's no single place where you can get all regions; some of the region's data is in the Customers table, and some of it's in the Stores table. If you ever build an application or a report from this information, you always have to combine the two regions' information into one.

Customers	Stores
CustomerID	Store ID
Name	Name
Birthdate	Region
Region	

What's done in the database design practice is to create a table for Region in scenarios like the one described above. This Region table would have a Region ID, Name, and other information about the region. The other two tables (Customers and Stores) will have links to this table using a field (which can be Region ID if we have the ID in both tables, or Name if it's unique enough to determine a region). This means having a relationship from the Stores and Customers table to the Region table.



In Dataverse, there are a number of ways to create a relationship. One way is to create a table, and then create a field in one table that's a relationship (or lookup) to another table, as described in the next section.

What are lookup or relationship fields?

In Dataverse, you can have a field defined as a lookup field, which points to another table.

Region

Display name *

Name *

Data type *

Region Lookup

Related table *

Required *

Optional

Searchable

Description

Advanced options

In the preceding image, the Region field is a lookup field to another table named Region Lookup. To learn more about different types of relationships, go to [Create a relationship between tables](#).

When the field mapping doesn't show the relationship fields

If you've created a lookup field in one table that points to another table, that field might not appear in the mapping of the dataflow. That's because **both** entities involved in the relationship require a Key field. This best practice would then make sure that the field is mappable in the table mappings of the dataflow.

Setting the Key field in the table

To set the key field in the table, go to the **Keys** tab, and add a key to a field that has unique values.

The screenshot shows the Power BI service interface with the 'Region Lookup' entity selected. The 'Keys' tab is highlighted in the ribbon. A red box labeled '2' surrounds the 'Add key' button. Another red box labeled '3' surrounds the 'NameKey' key configuration card, which includes fields for Display name, Name, and Fields.

After setting the key field, you can see the field in the mapping of the dataflow.

The screenshot shows the 'Power Query - Map entities' interface. In the 'Field mapping' section, there is a table with two rows. The first row maps 'CustomerID' to 'cre13_Name'. The second row, which is highlighted with a red box, maps 'Region' to 'cre13_Region.cre13_Name'. The 'Auto map' button is also visible.

Known limitations

- Mapping to [polymorphic lookup](#) fields is currently not supported.
- Mapping to a multi-level lookup field, a lookup that points to another tables' lookup field, is currently not supported.

Field mapping considerations for standard dataflows

1/15/2022 • 3 minutes to read • [Edit Online](#)

When you create dataflows that write their output to Dataverse, you can follow some guide lines and best practices to get the best outcome. In this article, some of those best practices are covered.

Set the key field in the entity

Having a primary key column in the entity helps in avoiding duplicates in the data rows. A primary key column is a column that's unique and deterministic of a data row in the entity. For example, in an Orders table, if the Order ID is the primary key column, you shouldn't have two rows with the same Order ID. Also, one Order ID, let's say an order with the ID 345, should only represent one row in the entity.

To choose the key column for the entity in Dataverse from the dataflow, you need to set the alternate key.

The following image shows how you can choose the key column to be populated from the source when you create a new entity in the dataflow.

The screenshot shows the 'Power Query - Map entities' interface. In the 'Field mapping' section, there are two dropdown menus. The first dropdown, labeled 'Unique primary name field', contains the value 'OrderID'. The second dropdown, labeled 'Alternate key fields (optional)', contains the value 'cre13_OrderID'. Both dropdowns have a downward arrow icon to their right. A red rectangle highlights the entire 'Field mapping' section, encompassing both dropdowns.

The primary name field that you see in the field mapping is for a label field; this field doesn't need to be unique. The field that's used in the entity for checking duplication will be the field that you set in the **Alternate Key** field.

Having a primary key in the entity ensures that even if you have duplicate data rows with the same value in the field that's mapped to the primary key, the duplicate entries won't be loaded into the entity, and the entity will always have a high quality of the data. Having an entity with a high quality of data is essential in building reporting solutions based on the entity.

The primary name field

The primary name field is a display field used in Dataverse. This field is used in default views to show the content of the entity in other applications. This field isn't the primary key field, and shouldn't be considered as that. This field can have duplicates, because it's a display field. The best practice, however, is to use a concatenated field to map to the primary name field, so the name is fully explanatory.

The alternate key field is what is used as the primary key.

What are good candidates for the key field

The key field is a unique value representing a unique row in the entity. It's important to have this field, because it helps you avoid having duplicate records in the entity. This field can come from three sources:

- The primary key in the source system (such as OrderID in the example above).
- A concatenated field created through Power Query transformations in the dataflow.

The screenshot shows the Power Query Editor interface. At the top, the ribbon has 'Home', 'Transform' (which is selected), and 'View'. The 'Transform' tab has a red box around the 'Add column' button (Step 2). Below the ribbon, there's a toolbar with various icons. One icon, 'Merge columns' (Step 3), has a red box around it. The main area shows a table with three columns: 'Territory', 'Region', and 'Merged Territory Region' (Step 4, highlighted with a red box). The 'Merged Territory Region' column contains values like 'Westboro,Eastern', 'Bedford,Eastern', etc.

	A ^B Territory	A ^B Region	A ^B Merged Territory Region
1	Westboro	Eastern	Westboro,Eastern
2	Bedford	Eastern	Bedford,Eastern
3	Georgetow	Eastern	Georgetow,Eastern
4	Boston	Eastern	Boston,Eastern
5	Cambridge	Eastern	Cambridge,Eastern
6	Braintree	Eastern	Braintree,Eastern
7	Providence	Eastern	Providence,Eastern
8	Hollis	Northern	Hollis,Northern
9	Portsmouth	Northern	Portsmouth,Northern
10	Wilton	Eastern	Wilton,Eastern

- A combination of fields to be selected in the **Alternate Key** option. A combination of fields used as a key field is also called a *composite key*.

This screenshot shows the 'Alternate key fields (optional)' dropdown in the Power BI Data Flow settings. It contains the text 'cre13_Region, cre13_Territory' with a dropdown arrow. Below the text, there are two checked checkboxes: 'cre13_Region' and 'cre13_Territory', each with up and down arrows for reordering.

Remove rows that no longer exist

If you want to have the data in your entity always synchronized with the data from the source system, choose the **Delete rows that no longer exist in the query output** option. However, be aware that this option slows down the dataflow because there's a need for a row comparison based on the primary key (alternate Key in the field mapping of the dataflow) for this action to occur.

Having this option checked means that if there's a data row in the entity that doesn't exist in the next dataflow refresh's query output, that row will be removed from the entity.

Queries	Load settings
Customers	<p>Reset load settings</p> <p>Entity name cre13_Customers_2</p> <p>Entity display name Customers</p> <p>Entity description</p> <p><input checked="" type="checkbox"/> Delete rows that no longer exist in the query output</p>

Known limitations

- Mapping to [polymorphic lookup](#) fields is currently not supported.
- Mapping to a multi-level lookup field, a lookup that points to another tables' lookup field, is currently not supported.
- Mapping to [Status](#) and [Status Reason](#) [fields](#) is currently not supported.

Security roles and permission levels in standard dataflows

1/15/2022 • 2 minutes to read • [Edit Online](#)

If someone in the team has created a dataflow and wants to share it with other team members, how does it work? What are the roles and permission level options available? This article takes you through the roles and permission levels related to standard dataflows.

Access to the environment

A standard dataflow stores data in Dataverse. Dataverse is located in an environment. Before accessing data stored in Dataverse, and also dataflows, you first need to have access to the environment.

Roles

There are multiple roles used to configure the security level for standard dataflows. The following table describes each role, along with the level of permission associated with that role.

SECURITY ROLE	PRIVILEGES	DESCRIPTION
Environment Maker	Create dataflows	Required to create any dataflow. Standard dataflows require additional roles depending on Dataverse tables permissions
Basic User	Write to non-custom entities	Has all the rights to work with non-custom entities
System Customizer	Create custom entities	Custom entities this user creates will be visible to this user only
Members of the environment	Get data from dataflows	Every member in the environment can get data from the dataflows in that environment

Row-level security isn't supported

The current version of standard dataflows doesn't support row-level security.

If you haven't heard of row-level security before, here's a quick introduction. If you have users with different levels of access to the same table, you can filter the data at the row level. For example, in the Orders table, you might have a SalesTerritory column, and you might want to filter the data in a way that users from California could only see records from the Orders table that belongs to California. This is possible through row-level security.

Steps to assign roles

The steps in the following procedure are sourced from [Configure user security to resources in an environment](#).

Verify that the user you want to assign a security role to is present in the environment. If not, add the user to the environment. You can assign a security role as part of the process of adding the user. More information: [Add users to an environment](#)

In general, a security role can only be assigned to users who are in the Enabled state. But if you need to assign a security role to users in the Disabled state, you can do so by enabling `allowRoleAssignmentOnDisabledUsers` in OrgDBOrgSettings.

To add a security role to a user who is already present in an environment:

1. Sign in to the [Power Platform admin center](#).
2. Select **Environments** > [select an environment] > **Settings** > **Users + permissions** > **Users**.
3. Select **Manage users in Dynamics 365**.

4. Select the user from the list of users in the environment, and then select **Manage roles**.

5. Assign one or more security roles to the user.

6. Select **OK**.

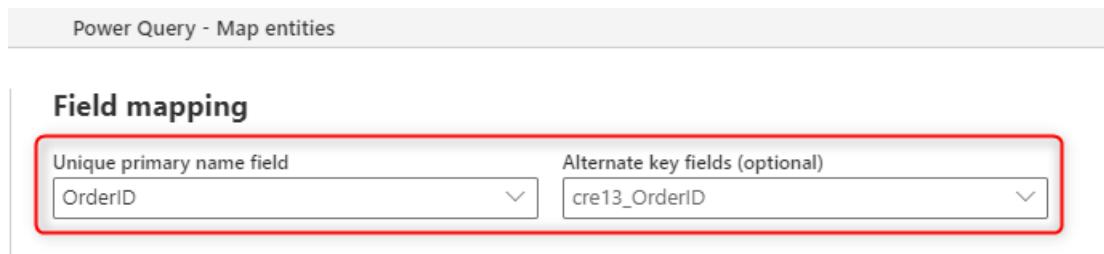
Sync your Excel data source with Dataverse using a dataflow

1/15/2022 • 3 minutes to read • [Edit Online](#)

One of the common scenarios that happens when you integrate data into Dataverse is keeping it synchronized with the source. Using the standard dataflow, you can load data into Dataverse. This article explains how you can keep the data synchronized with the source system.

The importance of the key column

If you're using a relational data base system as a source, normally you have key columns in the tables, and the data is in a proper format to be loaded into Dataverse. However, the data from the Excel files aren't always that clean. You often have an Excel file with sheets of data without having any key column. In [Field mapping considerations for standard dataflows](#), you can see that if there's a key column in the source, it can be easily used as the alternate key in the field mapping of the dataflow.



The screenshot shows the 'Power Query - Map entities' interface. Under the 'Field mapping' section, there are two dropdown menus. The first dropdown is labeled 'Unique primary name field' and contains the value 'OrderID'. The second dropdown is labeled 'Alternate key fields (optional)' and contains the value 'cre13_OrderID'. Both dropdowns have a downward arrow icon indicating they are expandable.

Having a key column is important for the table in Dataverse. The key column is the row identifier; this column contains unique values in each row. Having a key column helps in avoiding duplicate rows, and it also helps in synchronizing the data with the source system. If a row is removed from the source system, having a key column is helpful to find it and remove it from Dataverse as well.

Creating a key column

If you don't have a key column in your data source (Excel, text file, or any other sources), then you can generate one using the following method:

1. Clean your data.

The first step to create the key column is to remove all unnecessary rows, clean the data, remove empty rows, and remove any possible duplicates.

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7	ABC 123 Column8	ABC 123 Column9	ABC 123 Column10
1	Activities	null	null	null	null	null	null	null	null	null
2	Date	Calories Burned	Steps	Distance	Floors	Minutes Sedent...	Minutes Lightly Act...	Minutes Fairly Acti...	Minutes Very Acti...	Activity Calories
3	1/7/2016	4445	12346	9.87	3	994	202	92	152	3026
4	2/7/2016	4452	10805	8.23	1	1001	163	124	152	3005
5	3/7/2016	4273	6676	5.07	3	958	317	108	57	2917
6	4/7/2016	4147	6319	4.8	3	984	266	137	53	2724
7	5/7/2016	3446	4193	3.18	1	1130	206	66	38	1869
8	6/7/2016	4209	8613	6.74	4	992	278	92	78	2774
9	7/7/2016	4061	6276	4.76	4	1044	203	89	104	2559
10	8/7/2016	3924	6409	4.86	8	1031	235	120	54	2442
11	9/7/2016	4549	9942	7.55	16	980	169	149	142	3172
12	10/7/2016	4066	5761	4.37	0	1016	208	137	79	2609
13	11/7/2016	3587	5049	3.83	3	1114	190	75	61	1974
14	12/7/2016	4429	8263	6.27	15	950	283	128	79	3029
15	13-07-2016	4463	10761	8.33	2	980	238	95	127	3016
16	14-07-2016	3347	6170	4.68	12	1124	161	100	55	1840
17	15-07-2016	3744	5909	4.49	2	1103	155	87	95	2194
18	16-07-2016	4964	13972	10.63	10	928	172	160	180	3642
19	17-07-2016	4918	13568	10.3	13	889	204	202	145	3688
20	18-07-2016	4078	11933	9.06	12	1052	162	106	120	2527
21	19-07-2016	4417	12027	9.51	14	987	195	136	122	2973
22	20-07-2016	3883	6171	4.68	19	1026	282	68	64	2388
23	21-07-2016	4063	4790	3.64	1	1004	261	120	55	2617
24	22-07-2016	5680	12510	9.64	33	776	223	259	182	4524
25	23-07-2016	5390	12194	9.28	14	790	228	272	150	4239
26	24-07-2016	3861	5724	4.34	7	1088	175	95	82	2327
27	25-07-2016	4260	15196	11.56	14	1034	229	51	126	2736
28	26-07-2016	4012	13941	10.63	14	1087	154	55	144	2451
29	27-07-2016	4874	10686	8.11	18	907	243	148	142	3516



	Date	1 ² ₃ Calories Burned	1 ² ₃ Steps	1 ² ₃ Distance	1 ² ₃ Floors	1 ² ₃ Minutes Sedentary	1 ² ₃ Minutes Lightly Active	1 ² ₃ Minutes Fairly Active	1 ² ₃ Minutes Very Active
1	1/7/2016	4445	12346	9.87	3	994	202	92	152
2	2/7/2016	4452	10805	8.23	1	1001	163	124	152
3	3/7/2016	4273	6676	5.07	3	958	317	108	57
4	4/7/2016	4147	6319	4.8	3	984	266	137	53
5	5/7/2016	3446	4193	3.18	1	1130	206	66	38
6	6/7/2016	4209	8613	6.74	4	992	278	92	78
7	7/7/2016	4061	6276	4.76	4	1044	203	89	104
8	8/7/2016	3924	6409	4.86	8	1031	235	120	54
9	9/7/2016	4549	9942	7.55	16	980	169	149	142
10	10/7/2016	4066	5761	4.37	0	1016	208	137	79
11	11/7/2016	3587	5049	3.83	3	1114	190	75	61
12	12/7/2016	4429	8263	6.27	15	950	283	128	79
13	7/13/2016	4463	10761	8.33	2	980	238	95	127
14	7/14/2016	3347	6170	4.68	12	1124	161	100	55
15	7/15/2016	3744	5909	4.49	2	1103	155	87	95
16	7/16/2016	4964	13972	10.63	10	928	172	160	180
17	7/17/2016	4918	13568	10.3	13	889	204	202	145
18	7/18/2016	4078	11933	9.06	12	1052	162	106	120
19	7/19/2016	4417	12027	9.51	14	987	195	136	122
20	7/20/2016	3883	6171	4.68	19	1026	282	68	64
21	7/21/2016	4063	4790	3.64	1	1004	261	120	55
22	7/22/2016	5680	12510	9.64	33	776	223	259	182
23	7/23/2016	5390	12194	9.28	14	790	228	272	150
24	7/24/2016	3861	5724	4.34	7	1088	175	95	82
25	7/25/2016	4260	15196	11.56	14	1034	229	51	126
26	7/26/2016	4012	13941	10.63	14	1087	154	55	144
27	7/27/2016	4874	10686	8.11	18	907	243	148	142
28	7/28/2016	4058	5553	4.21	2	1011	189	164	76

2. Add an index column.

After the data is cleaned, the next step is to assign a key column to it. You can use **Add Index Column** from the **Add Column** tab for this purpose.

The screenshot shows the Power Query interface with the 'Edit queries' tab selected. The ribbon at the top has 'Home' and 'Transform' tabs, with 'Add column' highlighted. In the 'Custom' section of the ribbon, there is a 'Index column' button. A red circle labeled '1' is over 'Add column', and a red circle labeled '2' is over 'Index column'. A large red arrow points from the 'Index column' button to the 'Index' column in the table preview below.

When you add the index column, you have some options to customize it, for example, customizations on the starting number or the number of values to jump each time. The default start value is zero, and it increments one value each time.

Use the key column as the alternate key

Now that you have the key column(s), you can assign the dataflow's field mapping to the Alternate Key.

The screenshot shows the 'Map entities' section of Power Query. It has two dropdown menus: 'Unique primary name field' containing 'Index' and 'Alternate key fields' containing 'cre13_Index'. The 'Alternate key fields' dropdown is highlighted with a red box.

The setting is simple, you just need to set the alternate key. However, if you have multiple files or tables, it has one other step to consider.

If you have multiple files

If you have just one Excel file (or sheet or table), then the steps in the previous procedure are enough to set the alternate key. However, if you have multiple files (or sheets or tables) with the same structure (but with different data), then you to append them together.

If you're getting data from multiple Excel files, then the **Combine Files** option of Power Query will automatically append all the data together, and your output will look like the following image.

A	B	C	D
	Source.Name	Index	Date
	201605.xlsx	18	
	201605.xlsx	19	
	201605.xlsx	20	
	201605.xlsx	21	
	201605.xlsx	22	
	201605.xlsx	23	
	201605.xlsx	24	
	201605.xlsx	25	
	201605.xlsx	26	
	201605.xlsx	27	
	201605.xlsx	28	
	201605.xlsx	29	
	201605.xlsx	30	
	201606.xlsx	0	
	201606.xlsx	1	
	201606.xlsx	2	
	201606.xlsx	3	
	201606.xlsx	4	
	201606.xlsx	5	
	201606.xlsx	6	
	201606.xlsx	7	
	201606.xlsx	8	
	201606.xlsx	9	
	201606.xlsx	10	
	201606.xlsx	11	
	201606.xlsx	12	

As shown in the preceding image, besides the append result, Power Query also brings in the Source.Name column, which contains the file name. The Index value in each file might be unique, but it's not unique across multiple files. However, the combination of the Index column and the Source.Name column is a unique combination. Choose a composite alternate key for this scenario.

Alternate key fields

cre13_Index, cre13_SourceName

- cre13_ActivityCalories
- cre13_CaloriesBurned
- cre13_Date
- cre13_Distance
- cre13_Floors
- cre13_Index
- cre13_MinutesFairlyActive
- cre13_MinutesLightlyActive
- cre13_MinutesSedentary
- cre13_MinutesVeryActive
- cre13_SourceName
- cre13_Steps

Delete rows that no longer exists in the query output

The last step is to select the **Delete rows that no longer exist in the query output**. This option compares the data in the Dataverse table with the data coming from the source based on the alternate key (which might be a composite key), and remove the rows that no longer exist. As a result, your data in Dataverse will be always synchronized with your data source.

Queries

Customers

Load settings

[Reset load settings](#)

Entity name
cre13_Customers_2

Entity display name
Customers

Entity description



Delete rows that no longer exist in the query
output



Add data to a table in Microsoft Dataverse by using Power Query

1/15/2022 • 2 minutes to read • [Edit Online](#)

In this procedure, you'll create a table in [Dataverse](#) and fill that table with data from an OData feed by using Power Query. You can use the same techniques to integrate data from these online and on-premises sources, among others:

- SQL Server
- Salesforce
- IBM DB2
- Access
- Excel
- Web APIs
- OData feeds
- Text files

You can also filter, transform, and combine data before you load it into a new or existing table.

If you don't have a license for Power Apps, you can [sign up for free](#).

Prerequisites

Before you start to follow this article:

- Switch to an [environment](#) in which you can create tables.
- You must have a Power Apps per user plan or Power Apps per app plan.

Specify the source data

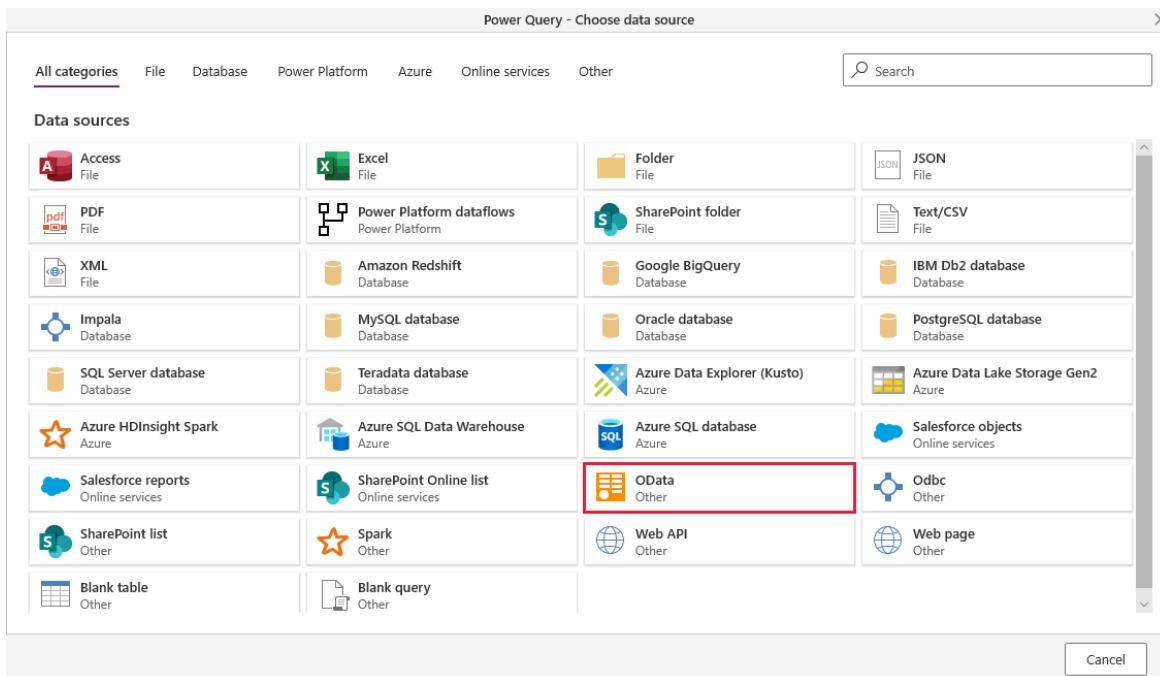
1. Sign in to [Power Apps](#).
2. In the navigation pane, select **Dataverse** to expand it, and then select **Tables**.

The screenshot shows the Microsoft Power Apps navigation pane. On the left, there's a sidebar with icons for Home, Learn, Apps, Create, and Dataverse. Under Dataverse, the Tables option is highlighted with a purple bar. The main area is titled "Tables" and lists various tables with columns for Name, Type, Customiza..., and Tags. Some tables listed include Account, Address, Appointment, Attachment, Business Unit, Contact, Currency, Dataflows Monitoring, Device Order, and Email. The "Tables" section is currently expanded.

Table ↑	Name	Type	Customiza...	Tags
Account	account	Standard	✓	Core
Address	customeraddress	Standard	✓	Standard
Appointment	appointment	Standard	✓	Productivity
Attachment	activitymimeattachment	Standard	✓	Productivity
Business Unit	businessunit	Standard	✓	Standard
Contact	contact	Standard	✓	Core
Currency	transactioncurrency	Standard	✓	Standard
Azure Synapse Link	cre6a_dataflowsmonit...	Custom	✓	Custom
Connections	cre6a_deviceorder	Custom	✓	Custom
Custom Connectors	cr21b_deviceorder	Custom	✓	Custom
Email	email	Standard	✓	Productivity

3. In the command menu, select **Data > Get data**.

4. In the list of data sources, select **OData**.



5. Under **Connection settings**, type or paste this URL, and then select **Next**:

```
https://services.odata.org/V4/Northwind/Northwind.svc/
```

6. In the list of tables, select the **Customers** check box, and then select **Next**.

The screenshot shows the 'Choose Data' dialog box. At the top left, it says 'Choose Data'. Below that is a 'Search' input field. Under 'Display Options', there is a dropdown menu showing 'OData [26]'. The main area lists various tables with checkboxes next to them. The 'Customers' checkbox is checked, while all other checkboxes are unchecked. The listed tables include: Alphabetical_list_of_p..., Categories, Category_Sales_for_1..., Current_Product_Lists, Customer_and_Suppli..., CustomerDemograph..., Customers, Employees, Invoices, Order_Details, Order_Details_Extend..., Order_Subtotals, Orders, Orders_Qries, Product_Sales_for_1997, Products, and Products_Above_Aver...

7. (optional) Modify the schema to suit your needs by choosing which columns to include, transforming the table in one or more ways, adding an index or conditional column, or making other changes.

8. In the lower-right corner, select **Next**.

Specify the target table

1. Under Load settings, select Load to new table.

Load settings

Load to new table
 Load to existing table
 Do not load

Table name
Customers

Table display name
Customers

Table description

Delete rows that no longer exist in the query output (i)

You can give the new table a different name or display name, but leave the default values to follow this tutorial exactly.

2. In the Unique primary name column list, select ContactName, and then select Next.

You can specify a different primary-name column, map a different column in the source table to each column in the table that you're creating, or both. You can also specify whether Text columns in your query output should be created as either Multiline Text or Single-Line Text in the Dataverse. To follow this tutorial exactly, leave the default column mapping.

3. Select Refresh manually for Power Query - Refresh Settings, and then select Publish.

4. Under Dataverse (near the left edge), select Tables to show the list of tables in your database.

The Customers table that you created from an OData feed appears as a custom table.

The screenshot shows the Microsoft Power Platform Admin Center interface. On the left, there's a navigation sidebar with options like Home, Learn, Apps, Create, Dataverse, Tables, Choices, Dataflows, Azure Synapse Link, Connections, Custom Connectors, and Gateways. The 'Tables' option is selected. The main area is titled 'Tables' and lists various entities. One entity, 'Customers', is highlighted with a red border. The columns in the list are Table, Name, Type, Customiza..., and Tags.

Table	Name	Type	Customiza...	Tags
Account	account	Standard	✓	Core
Address	customeraddress	Standard	✓	Standard
Appointment	appointment	Standard	✓	Productivity
Attachment	activitymimeattachment	Standard	✓	Productivity
Business Unit	businessunit	Standard	✓	Standard
Contact	contact	Standard	✓	Core
Currency	transactioncurrency	Standard	✓	Standard
Customers	cr216_customers	Custom	✓	Custom
Dataflows Monitoring	cre6a_dataflowsmonit...	Custom	✓	Custom
Device Order	cre6a_deviceorder	Custom	✓	Custom
Device Order	cr21b_deviceorder	Custom	✓	Custom

WARNING

If you use Power Query to add data to an existing table, all data in that table will be overwritten.

If you select **Load to existing table**, you can specify a table into which you add data from the **Customers** table. You could, for example, add the data to the **Account** table with which the Dataverse ships. Under **Column mapping**, you can further specify that data in the **ContactName** column from the **Customers** table should be added to the **Name** column in the **Account** table.

The screenshot shows the 'Power Query - Map tables' dialog. On the left, under 'Load settings', the 'Load to existing table' radio button is selected, and the 'Destination table' dropdown is set to 'Account'. Below that, 'Table display name' is set to 'Account' and 'Table description' is provided. At the bottom, there's a checkbox for 'Delete rows that no longer exist in the query output'. On the right, under 'Column mapping', it shows 'Key columns (none)'. Below that is a list of columns being mapped, with 'ContactName' mapped to 'Name'. There are other columns listed: 'Name', 'NumberOfEmployees', 'OnHoldTime', 'OwnershipCode', 'ParticipatesInWorkflow', 'PaymentTermsCode', 'PreferredAppointmentDayCode', 'PreferredAppointmentTimeCode', 'PreferredContactMethodCode', 'PreferredSystemUserId.AzureActiveDirectoryObjectId', 'PrimarySatorId', and 'PrimaryTwitterId'. A red box highlights the 'ContactName' field in the mapping list. At the bottom right, there are 'Cancel' and 'Next' buttons.

If an [error message about permissions](#) appears, contact your administrator.

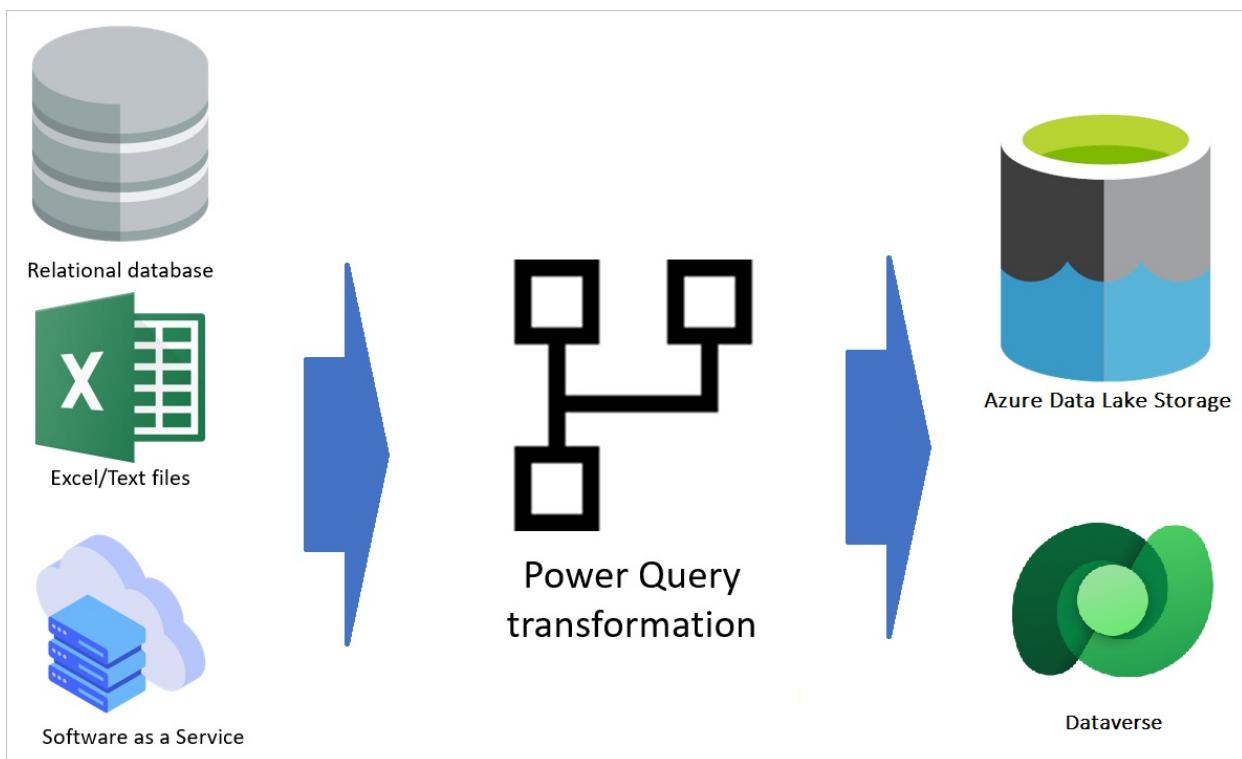
How Microsoft Power Platform dataflows and Azure Data Factory wrangling dataflows relate to each other

1/15/2022 • 2 minutes to read • [Edit Online](#)

Microsoft Power Platform dataflows and Azure Data Factory dataflows are often considered to be doing the same thing: extracting data from source systems, transforming the data, and loading the transformed data into a destination. However, there are differences in these two types of dataflows, and you can have a solution implemented that works with a combination of these technologies. This article describes this relationship in more detail.

Power Platform dataflows

Power Platform dataflows are data transformation services empowered by the Power Query engine and hosted in the cloud. These dataflows get data from different data sources and, after applying transformations, store it either in Dataverse or in Azure Data Lake Storage.



Data Factory wrangling dataflows

Data Factory is a cloud-based extract, transform, load (ETL) service that supports many different sources and destinations. There are two types of dataflows under this technology: mapping dataflows and wrangling dataflows. Wrangling dataflows are empowered by the Power Query engine for data transformation.

What do they have in common?

Power Platform dataflows and Data Factory wrangling dataflows are both useful for getting data from one or

more sources, applying transformations on the data by using Power Query, and loading the transformed data into destinations. In addition:

- Both are empowered by using Power Query data transformation.
- Both are cloud-based technologies.

What's the difference?

The main point is knowing their differences, because then you can think about scenarios where you'd want to use one or the other.

FEATURES	POWER PLATFORM DATAFLOWS	DATA FACTORY WRANGLING DATAFLOWS
Destinations	Dataverse or Azure Data Lake Storage	Many destinations (see the list here)
Power Query transformation	All Power Query functions are supported	A limited set of functions are supported (see the list here)
Sources	Many sources are supported	Only a few sources (see the list here)
Scalability	Depends on the Premium capacity and the use of the enhanced compute engine	Highly scalable

Which user persona is suited to which type of dataflow?

If you're a citizen application developer or citizen data analyst with small-scale to medium-scale data to be integrated and transformed, you'll find the Power Platform dataflows more convenient. The large number of transformations available, the ability to work with them without having developer knowledge, and the fact that dataflows can be authored, monitored, and edited in Power BI or Power Platform—all are reasons that make Power Platform dataflows a great data integration solution for this type of developer.

If you're a data developer who's dealing with big data and huge datasets, with a large number of rows to be ingested every time, you'll find the Data Factory wrangling dataflows a better tool for the job. Wrangling data flow translates M generated by the Power Query Online Mashup Editor into spark code for cloud scale execution. Working with the Azure portal to author, monitor, and edit wrangling dataflows requires a higher developer learning curve than the experience in Power Platform dataflows. Wrangling dataflows are best suited for this type of audience.

Power Automate templates for the dataflows connector

1/15/2022 • 2 minutes to read • [Edit Online](#)

The dataflows Power Automate connector can:

- Trigger a flow when a dataflow refresh completes.
- Take action to start a dataflow refresh.

This section discusses some use cases with provided tutorials to help you quickstart the use of this connector:

[Send notifications:](#)

- When a dataflow refresh status changes, send an email notification.
- When a dataflow succeeds or fails, send an email notification.
- When a dataflow refresh status changes, send a Teams notification.

[Open support tickets:](#)

- When a dataflow refresh fails, send a message to Azure Service Bus queue to open a support ticket.

[Refresh dataflows and datasets sequentially:](#)

- When you select a button, start a dataflow refresh.
- When an analytical dataflow refresh succeeds, start a standard dataflow refresh.
- When a dataflow refresh succeeds, start a Power BI dataset refresh.
- When a file in SharePoint is updated, start a dataflow refresh.

Save dataflow refresh metadata and build a dataflows monitoring dashboard:

- [When a dataflow refresh completes, save metadata to a Dataverse table](#)
- [When a dataflow refresh completes, save metadata to Excel online](#)
- [When a dataflow refresh completes, save metadata to a Power BI streaming dataset](#)

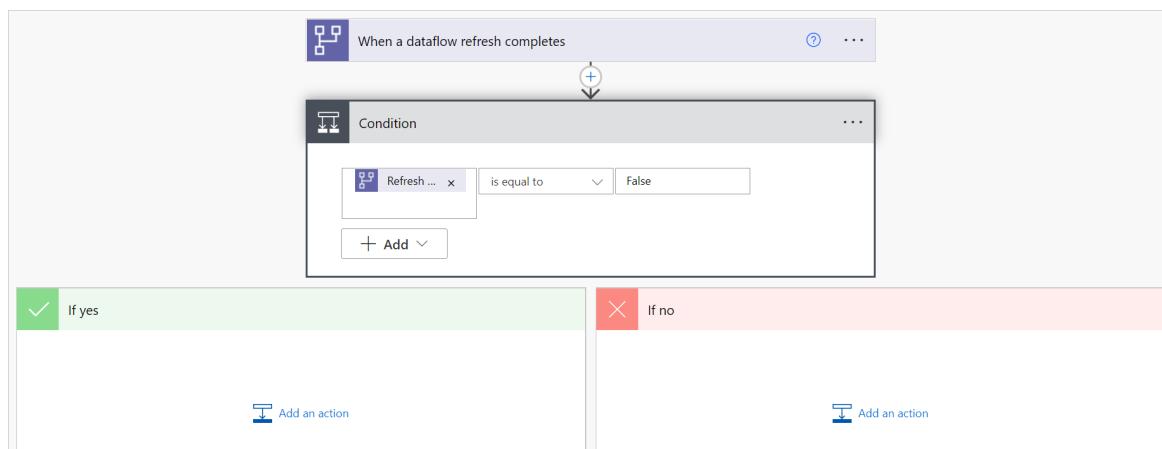
Send a notification when a dataflow refresh completes

1/15/2022 • 2 minutes to read • [Edit Online](#)

When your dataflow refresh completes, you or others who manage or depend on the dataflow might want to receive a notification to alert you of the dataflow refresh status. This way, you know your data is up to date and you can start getting new insights. Another common scenario addressed by this tutorial is notification after a dataflow fails. A notification allows you to start investigating the problem and alert people that depend on the data being successfully refreshed.

To set up a Power Automate notification that will be sent when a dataflow fails:

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.
5. Select **New step** to add an action to your flow.
6. Search for the **Condition** connector, and then select it.
7. Customize the **Condition** connector. Enter the following information:
 - a. In the first cell, add **Refresh Status** from the dataflow connector.
 - b. Leave the second cell as **is equal to**.
 - c. In the third cell, enter **False**.



8. In the **If Yes** section, select **Add an action**.
9. Search for the "Send an email notification (V3)" connector, and then select it.
10. Enter the email address and subject information.

11. Inside the body of the email, select the field next to **Body** and use **Dynamic content** to add dataflow information to the content of your email.

The screenshot shows the 'Send an email notification (V3)' configuration screen in Power Automate. In the 'Body' section, there are several dynamic content fields:

- Dataflow Name: Dataflow Name
- Dataflow ID: Dataflow Id
- Refresh Status: Refresh Status
- Refresh Type: Refresh Type
- Start Time: Start Time
- End Time: End Time

Below the body section is an 'Add dynamic content' button. To the right, a 'Dynamic content' pane is open, showing the available fields under the 'When a dataflow refresh completes' trigger:

- Dataflow Id
- Dataflow Name
- Refresh Type
- Start Time
- End Time
- Refresh Status

Open a ticket when a dataflow refresh fails

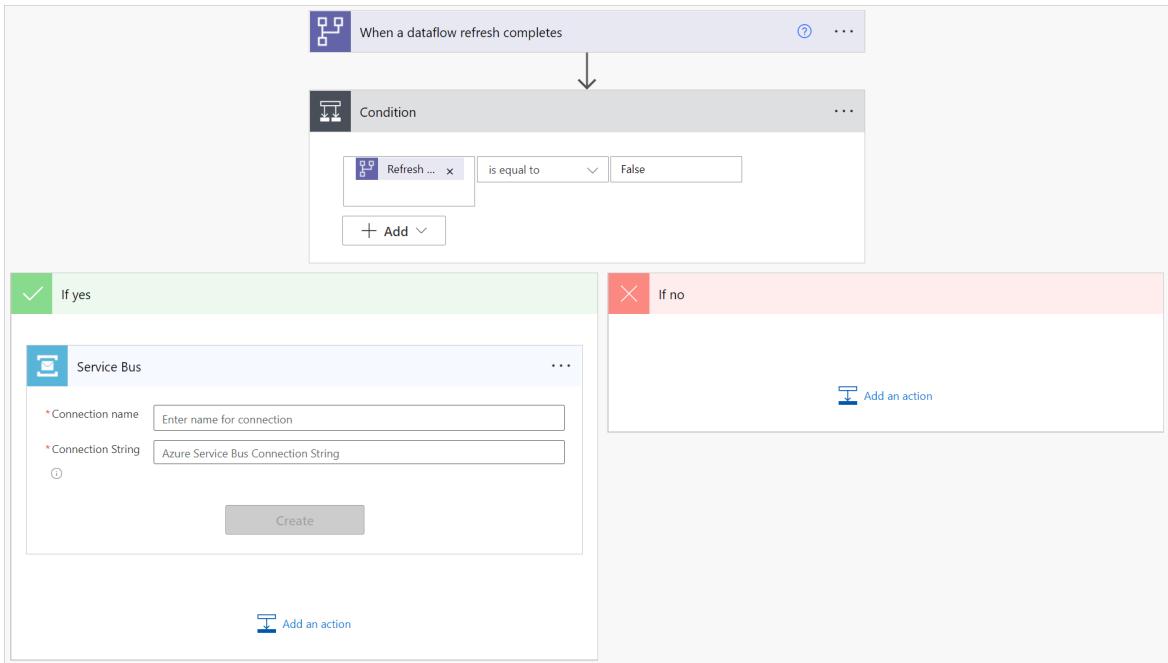
1/15/2022 • 2 minutes to read • [Edit Online](#)

When your dataflow refresh completes or has been taking longer than expected, you might want your support team to investigate. With this tutorial, you can automatically open a support ticket, create a message in a queue or Service Bus, or add an item to Azure DevOps to notify your support team.

In this tutorial, we make use of Azure Service Bus. For instructions on how to set up an Azure Service Bus and create a queue, go to [Use Azure portal to create a Service Bus namespace and a queue](#).

To automatically create a queue in Azure Service Bus:

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.
5. Select **New step** to add an action to your flow.
6. Search for the **Condition** connector, and then select it.
7. Customize the **Condition** connector. Enter the following information:
 - a. In the first cell, add **Refresh Status** from the dataflow connector.
 - b. Leave the second cell as **is equal to**.
 - c. In the third cell, enter **False**.
8. In the **If Yes** section, select **Add an action**.
9. Search for the "Send message" connector from Service Bus, and then select it.



10. Enter a **Connection name** for this message. In **Connection string**, enter the connection string that was generated when you created the Service Bus namespace. Then select **Create**.
11. Add dataflow information to the content of your message by selecting the field next to **Content**, and then select the dynamic content you want to use from **Dynamic content**.

Send message

*Queue/Topic name: Name of the queue or topic

Session Id: Identifier of the session

Content

- Dataflow Name: Dataflow Name
- Dataflow ID: Dataflow ID
- Refresh Status: Refresh Status
- Refresh Type: Refresh Type
- Start Time: Start Time
- End Time:

Add dynamic content +

Content Type: Content type of the message content

Properties: Key-value pairs for each brokered property

Show advanced options ▾

Add an action

Dynamic content

- When a dataflow refresh completes
- Dataflow Id: Id of the dataflow
- Dataflow Name: Name of the dataflow
- Refresh Type: Type of the dataflow refresh
- Start Time: Start time of the dataflow refresh
- End Time: Completion time of the dataflow refresh
- Refresh Status: Status of the dataflow refresh. Possible values are: 'Success', ...

Trigger dataflows and Power BI datasets sequentially

1/15/2022 • 2 minutes to read • [Edit Online](#)

There are two common scenarios for how you can use this connector to trigger multiple dataflows and Power BI datasets sequentially.

- Trigger the refresh of a standard dataflow after the successful completion of an analytical dataflow refresh.

If a single dataflow does every action, then it's hard to reuse its entities in other dataflows or for other purposes. The best dataflows to reuse are dataflows doing only a few actions, specializing in one specific task. If you have a set of dataflows as staging dataflows, and their only action is to extract data "as is" from the source system, these dataflows can be reused in multiple other dataflows. More information: [Best practices for reusing dataflows across environments and workspaces](#)

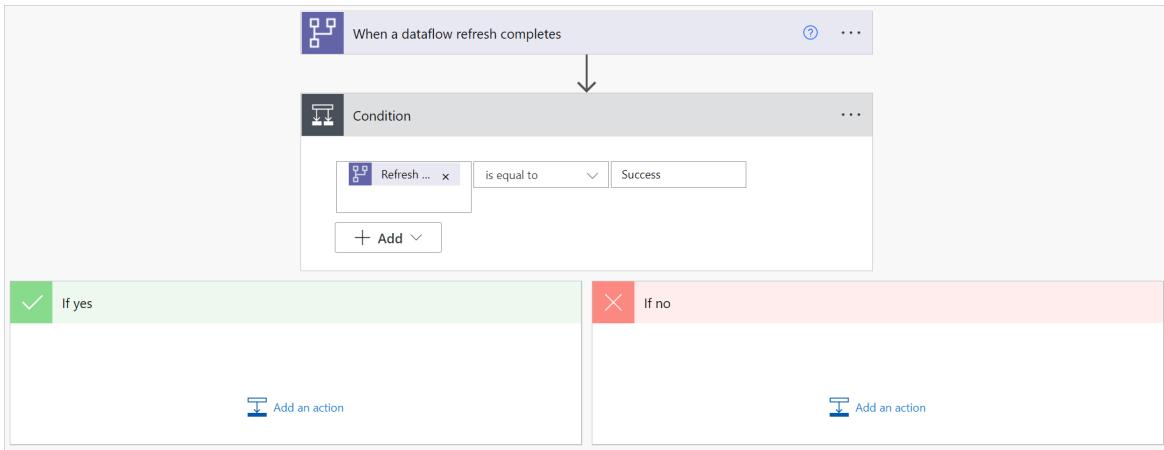
- Trigger the refresh of a Power BI dataset when a dataflow refresh completes successfully.

If you want to ensure that your dashboard is up to date after a dataflow refreshes your data, you can use the connector to trigger the refresh of a Power BI dataset after your dataflow refreshes successfully.

This tutorial covers the first scenario.

To trigger dataflows sequentially:

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.
5. Select **New step** to add an action to your flow.
6. Search for the **Condition** connector, and then select it.
7. Customize the Condition connector. Enter the following information:
 - a. In the first cell, add **Refresh Status** from the dataflow connector.
 - b. Leave the second cell as **is equal to**.
 - c. In the third cell, enter **Success**.



8. In the If Yes section, select **Add an action**.
9. Search for the "Refresh a dataflow" connector, and then select it.
10. Customize the connector:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.

The screenshot shows the configuration screen for the 'Refresh a dataflow' action. It has three input fields:

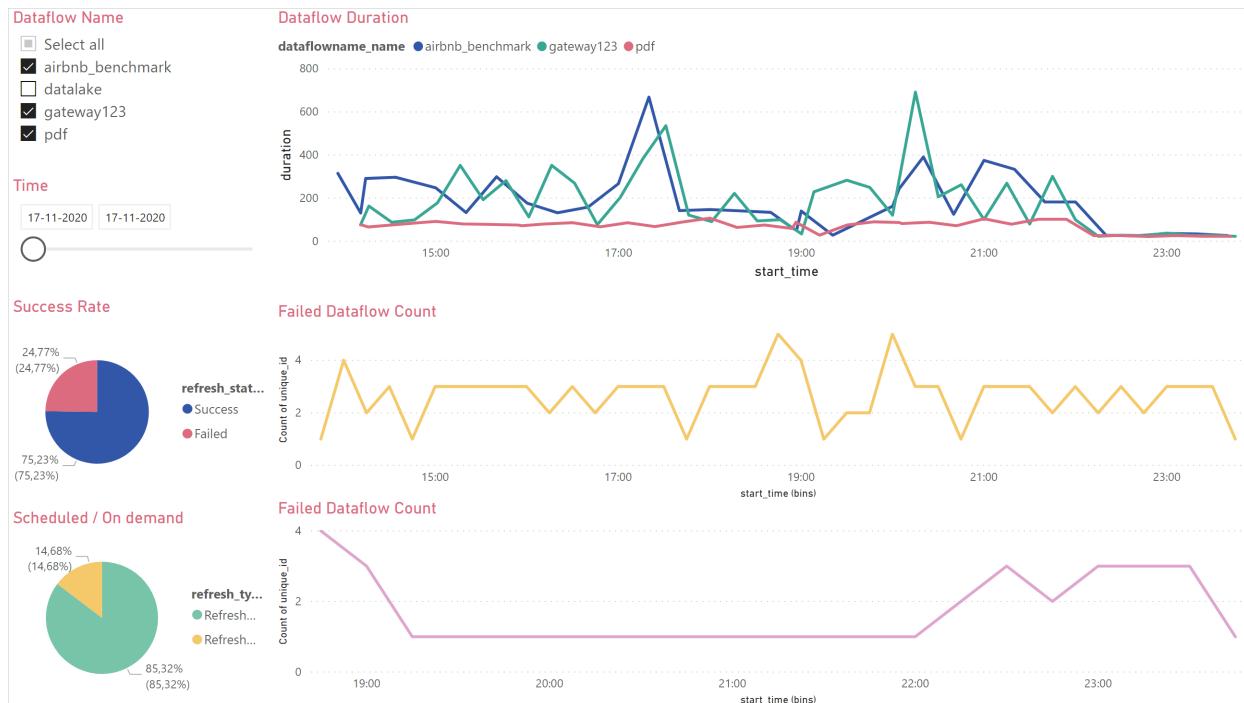
- * Group Type: Select workspace or environment
- * Group: The unique identifier of the workspace or environment
- * Dataflow: The unique identifier of the dataflow

 Below the fields is a placeholder 'Add an action' button.

Load data in a Dataverse table and build a dataflows monitoring report with Power BI

1/15/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to load data in a Dataverse table to create a dataflows monitoring report in Power BI.



You can use this dashboard to monitor your dataflows' refresh duration and failure count. With this dashboard, you can track any issues with your dataflows performance and share the data with others.

First, you'll create a new Dataverse table that stores all the metadata from the dataflow run. For every refresh of a dataflow, a record is added to this table. You can also store metadata for multiple dataflow runs in the same table. After the table is created, you'll connect the Power BI file to the Dataverse table.



Dataflow refresh



Save meta-data into Dataverse table



Visualize data with Power BI

Prerequisites

- [Power BI Desktop](#).
- A [Dataverse environment](#) with permissions to create new custom tables.
- A [Premium Power Automate License](#).
- A [Power BI dataflow](#) or [Power Platform dataflow](#).

Download the .pbit file

First, download the Dataverse [.pbit file](#).

Create a new table in Dataverse

1. Navigate to the [Power Apps portal](#).
2. On the left navigation pane expand **Data**, select **Tables**, and then select **New table**.

The screenshot shows the Microsoft Power Apps portal interface. On the left, there's a navigation pane with 'Tables' selected. In the center, a list of standard tables is shown. On the right, a 'New table' dialog box is open, prompting for the new table's details. The 'Name' field is pre-filled with 'cre6a'. The 'Primary Name Column' is set to 'Name'. There's also a checkbox for enabling attachments.

3. In the **New table** pane:
 - a. Enter **Dataflows Monitoring** in **Display name**.
 - b. Under **Primary Name Column**, enter **Dataflow name** in **Display name**.
 - c. Select **Create**.
4. Select **Add column** to repeat adding columns for the following values:
 - **Display name:** "Refresh Status", **Data type:** Text, **Required:** Required.
 - **Display name:** "Refresh Type", **Data type:** Text, **Required:** Required.
 - **Display name:** "Start Time", **Data type:** Date and Time, **Required:** Required.
 - **Display name:** "End Time", **Data type:** Date and Time, **Required:** Required.

The screenshot shows the Microsoft Power Apps interface with the 'Dataflows Monitoring' table selected. A 'Column properties' dialog is open on the right side, overlaid on the table. The 'Add column' button is highlighted with a red box. The dialog fields include:

- Display name ***: cre6a_
- Name ***: cre6a_
- Data type ***: Text
- Required ***: Optional
- Searchable**: checked
- Description**: (empty)
- Advanced options**: (button)

At the bottom of the dialog are 'Done' and 'Cancel' buttons.

Create a dataflow

If you don't already have one, create a dataflow. You can create a dataflow in either [Power BI dataflows](#) or [Power Apps dataflows](#).

Create a Power Automate flow

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type**: Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group**: Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow**: Select your dataflow by name.
5. Select **New step** to add an action to your flow.
6. Search for the "Add a new row" connector from Dataverse, and then select it.
7. In **Add a new row**, select **Choose a table** and then choose **Dataflows Monitoring** from the list.

When a dataflow refresh completes

Add a new row

Dataflows Monitorings

Required name field

End Time

Refresh Status

Refresh Type

Start Time

Show advanced options

8. For every required field, you need to add a dynamic value. This value is the output of the metadata of the dataflow that's run.

- Select the field next to **Dataflow Name** and then select **Dataflow Name** from the dynamic content.

Add a new row

Dataflows Monitorings

Dataflow Name

End Time

Refresh Status

Refresh Type

Start Time

Show advanced options

+ New step Save

Add dynamic content from the apps and connectors used in this flow.

Dynamic content Expression

Search dynamic content

When a dataflow refresh completes

Dataflow Id
Id of the dataflow

Dataflow Name
Name of the dataflow

Refresh Type
Type of the dataflow refresh

Start Time
Start time of the dataflow refresh

End Time
Completion time of the dataflow refresh

Refresh Status
Status of the dataflow refresh. Possible values are: 'Success', ...

- Repeat this process for all required fields.

The screenshot shows a Microsoft Power BI Dataflows Monitoring entity form. At the top, there's a green header bar with a circular icon, the text "Add a new row", a help icon, and a three-dot menu icon. Below the header are six input fields arranged in two columns. The left column contains: "Table name" (set to "Dataflows Monitorings"), "Dataflow Name" (with a small icon and placeholder "Dataflow Name x"), "End Time" (with a small icon and placeholder "End Time x"), "Refresh Status" (with a small icon and placeholder "Refresh Status x"), "Refresh Type" (with a small icon and placeholder "Refresh Type x"), and "Start Time" (with a small icon and placeholder "Start Time x"). Each field has a delete icon ("x") to its right. At the bottom of the form is a link "Show advanced options" followed by a downward arrow.

9. Save the flow.

Create a Power BI Report

1. Open the `.pbix` file.
2. Connect to your Dataverse entity **Dataflows Monitoring**.

In this dashboard, for every dataflow in your specified time interval, you can monitor:

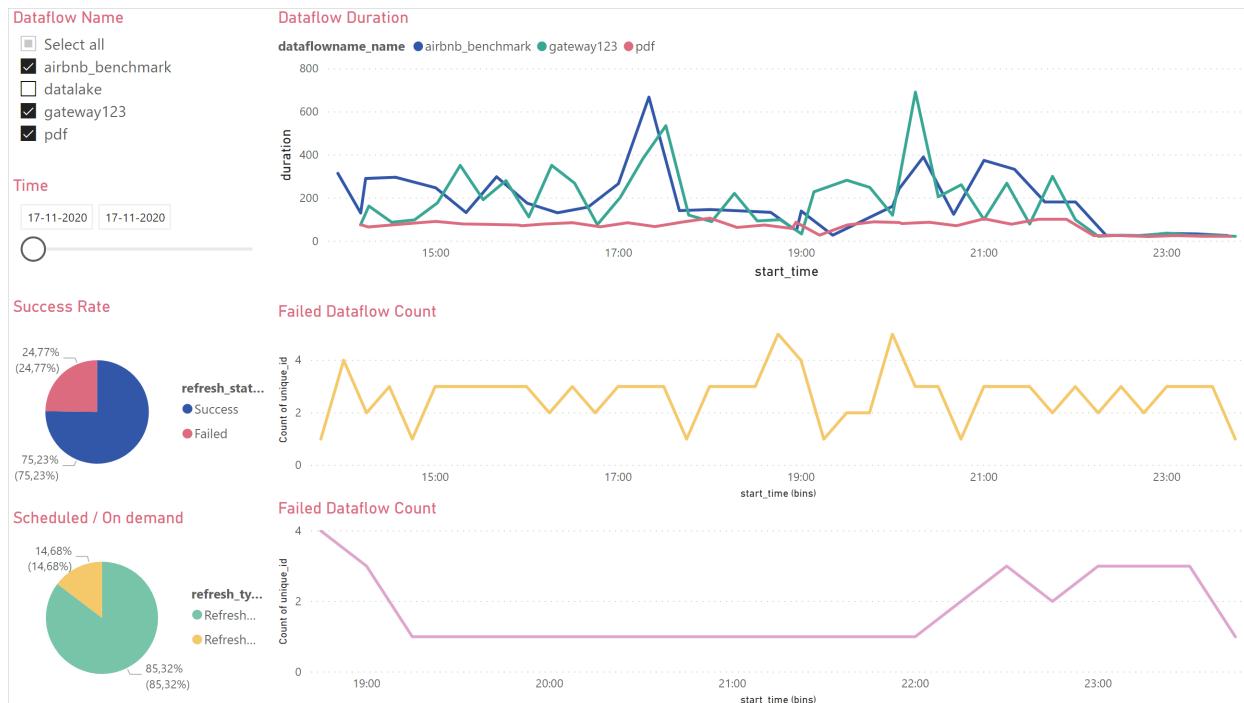
- The dataflow duration
- The dataflow count
- The dataflow failure count

The unique ID for every dataflow is generated by a merge between the dataflow name and the dataflow start time.

Load data in Excel Online and build a dataflows monitoring report with Power BI

1/15/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to use an Excel file and the dataflows connector in Power Automate to create a dataflows monitoring report in Power BI.



First, you'll download the Excel file and save it in OneDrive for Business or SharePoint. Next, you'll create a Power Automate connector that loads metadata from your dataflow to the Excel file in OneDrive for Business or SharePoint. Lastly, you'll connect a Power BI file to the Excel file to visualize the metadata and start monitoring the dataflows.

You can use this dashboard to monitor your dataflows' refresh duration and failure count. With this dashboard, you can track any issues with your dataflows performance and share the data with others.



Dataflow refresh



Save meta-data into Excel on
OneDrive for Business



Visualize data with Power BI

Prerequisites

- Microsoft Excel
- Power BI Desktop.
- A Premium Power Automate License
- OneDrive for Business.

- A [Power BI dataflow](#) or [Power Platform dataflow](#).

Download the .pbit file

First, download the [.pbit file](#).

Download the Excel file and save to OneDrive

Next, download the [.xlsx file](#) and save the file to a location on OneDrive for Business or SharePoint

Create a dataflow

If you don't already have one, create a dataflow. You can create a dataflow in either [Power BI dataflows](#) or [Power Apps dataflows](#).

Create a flow in Power Automate

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.
5. Select **New step** to add an action to your flow.
6. Search for the "Add a row into a table" connector from Excel Online (Business), and then select it.
7. Customize the connector. Enter the *Location* of the Excel file and the specific *Table* the data loads to.
 - **Location:** Select the location of the Excel file on OneDrive for Business or SharePoint.
 - **Document Library:** Select the library of the Excel file.
 - **File:** Select the file path to the Excel file.
 - **Table:** Select "Dataflow_monitoring".

The screenshot shows a Power Automate flow with two steps:

- When a dataflow refresh completes**: This step has three parameters:
 - * Group Type: Environment
 - * Group: Contoso
 - * Dataflow: DocTestDataflow
- Add a row into a table**: This step has four parameters:
 - * Location: Select from the drop-down or specify one of the following: - "me" - "Shar
 - * Document Library: Select a document library from the drop-down.
 - * File: Select an Excel file through File Browse.
 - * Table: Select a table from the drop-down.

8. Add dynamic values to the required fields.

For every required field, you need to add a dynamic value. This value is the output of the metadata of the dataflow run.

- Select the field next to **dataflowname_name**.
- Select **Dataflow Name** from the **Dynamic content** context box.

The screenshot shows the 'Add a row into a table' step with the 'dataflowname_name' field selected. A dynamic content context box is open, listing several variables related to the dataflow refresh. The 'Dataflow Name' variable is highlighted.

Field	Value
* Location	OneDrive for Business
* Document Library	OneDrive
* File	/Power Query/dataflow_monitoring.xlsx
* Table	Dataflow_monitoring
dataflowname_name	(Dynamic content)
refresh_status	
refresh_type	
start_time	
end_time	
Dataflow_id	

Dynamic content context box:

- Search dynamic content: When a dataflow refresh completes
- Dataflow Id: Id of the dataflow
- Dataflow Name**: Name of the dataflow (highlighted)
- Refresh Type: Type of the dataflow refresh
- Start Time: Start time of the dataflow refresh
- End Time: Completion time of the dataflow refresh
- Refresh Status: Status of the dataflow refresh. Possible values are: 'Success', '...

- Repeat this process for all required fields.

9. Save the flow.

Create a Power BI Report

1. Open the `.pbix` file.

2. Connect to your Excel file.

In this dashboard, for every dataflow in your specified time interval, you can monitor:

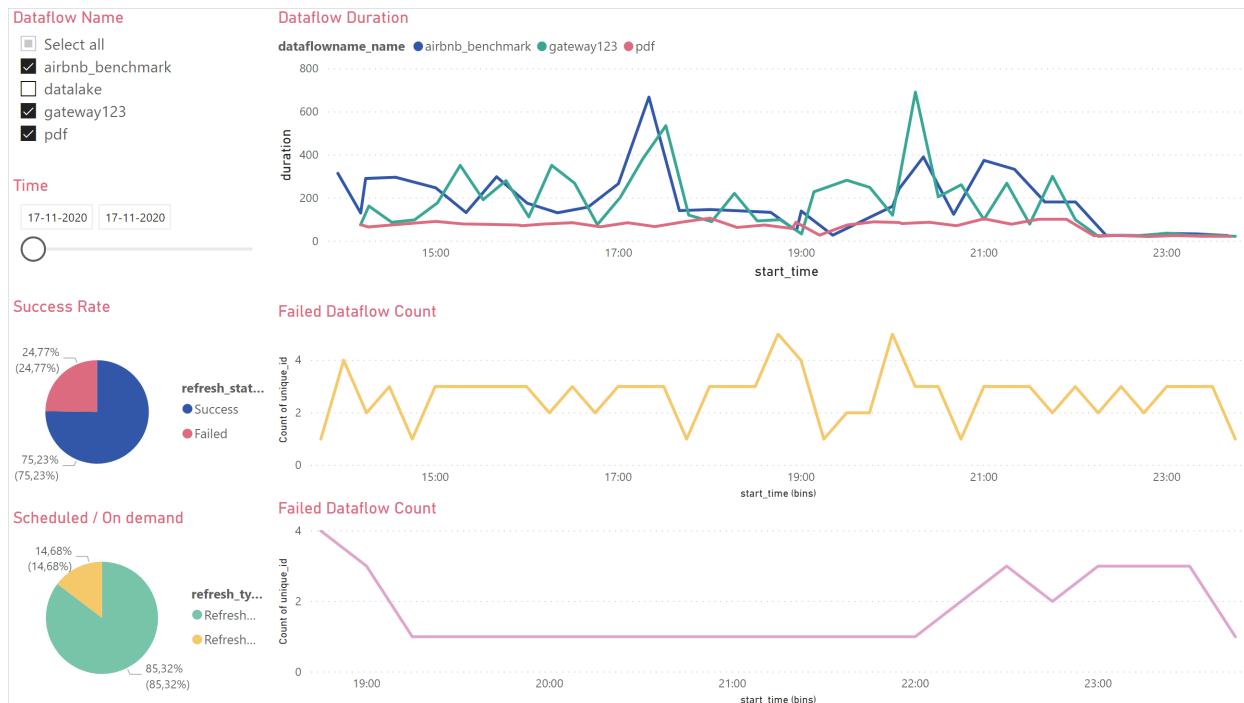
- The dataflow duration
- The dataflow count
- The dataflow failure count

The uniqueID for every dataflow is generated by a merge between the dataflow name and the dataflow start time.

Load data in a Power BI streaming dataset and build a dataflows monitoring report with Power BI

1/15/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to load data in a Power BI streaming dataset to create a dataflows monitoring report in Power BI.



First, you'll create a new streaming dataset in Power BI. This dataset collects all the metadata from the dataflow run, and for every refresh of a dataflow, a record is added to this dataset. You can run multiple dataflows all to the same dataset. Lastly, you can build a Power BI report on the data to visualize the metadata and start monitoring the dataflows.

You can use this dashboard to monitor your dataflows' refresh duration and failure count. With this dashboard, you can track any issues with your dataflows performance and share the data with others.



Dataflow refresh



Save meta-data to Power BI streaming dataset



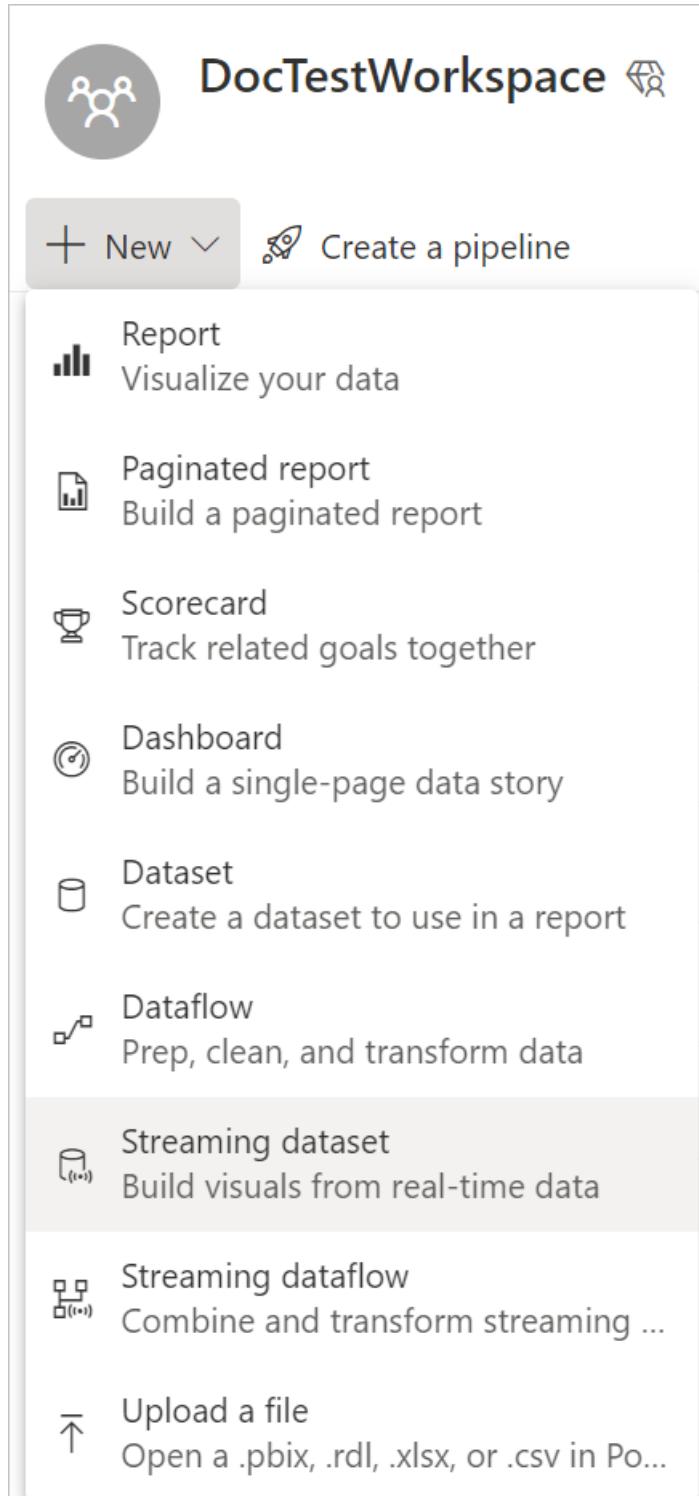
Visualize data with Power BI

Prerequisites

- A [Power BI Pro License](#).
- A [Premium Power Automate License](#)
- A [Power BI dataflow](#) or [Power Platform dataflow](#).

Create a new streaming dataset in Power BI

1. Navigate to [Power BI](#).
2. Open a workspace.
3. From the workspace, select **New > Streaming dataset**.



4. From **New streaming dataset**, select the **API** tile, and then select **Next**.

New streaming dataset

Choose the source of your data



API



AZURE STREAM



PUBNUB

5. In the new pane, turn **Historic data analysis** on.

6. Enter the following values, and then select **Create**.

- **Dataset Name:** "Dataflow Monitoring".
- **Value:** "Dataflow Name", **Data type:** Text.
- **Value:** "Dataflow ID", **Data type:** Text.
- **Value:** "Refresh Status", **Data type:** Text.
- **Value:** "Refresh Type", **Data type:** Text.
- **Value:** "Start Time", **Data type:** Date and Time.
- **Value:** "End Time", **Data type:** Date and Time.

New streaming dataset

Create a streaming dataset and integrate our API into your device or application to send data. [Learn more about the API](#).

* Required

Dataset name *

Dataflow Monitoring

Values from stream *

Dataflow Name

Text



Dataflow ID

Text



Refresh Status

Text



Refresh Type	Text	
Start Time	DateTime	
End Time	DateTime	
Enter a new value name	Text	

```
[  
 {  
   "Dataflow Name" : "AAAAAA555555",  
   "Dataflow ID" : "AAAAAA555555",  
   "Refresh Status" : "AAAAAA555555",  
   "Refresh Type" : "AAAAAA555555",  
   "Start Time" : "2021-11-16T16:18:58.871Z",  
   "End Time" : "2021-11-16T16:18:58.871Z"  
 }  
 ]
```

Historic data analysis

On

Back Create Cancel

Create a dataflow

If you do not already have one, create a dataflow. You can create a dataflow in either [Power BI dataflows](#) or [Power Apps dataflows](#).

Create a flow in Power Automate

1. Navigate to [Power Automate](#).
2. Select **Create > Automated cloud flow**.
3. Enter a flow name, and then search for the "When a dataflow refresh completes" connector. Select this connector from the list, and then select **Create**.
4. Customize the connector. Enter the following information on your dataflow:
 - **Group Type:** Select *Environment* when connecting to Power Apps and *Workspace* when connecting to Power BI.
 - **Group:** Select the Power Apps environment or the Power BI workspace your dataflow is in.
 - **Dataflow:** Select your dataflow by name.
5. Select **new step** to add an action to your flow.
6. Search for the connector "Add rows to a dataset" from Power BI, and then select it.

7. Customize the connector. Enter the following information:

- **Workspace ID:** Select the Power BI workspace that contains your streaming dataset.
- **Dataset:** Select the streaming dataset **Dataflow Monitoring** that you previously created in [Create a new streaming dataset in Power BI](#).
- **Table:** Select **RealTimeData**.

When a dataflow refresh completes

* Group Type Select workspace or environment

* Group The unique identifier of the workspace or environment

* Dataflow The unique identifier of the dataflow

Add rows to a dataset

* Workspace The unique identifier of the workspace.

* Dataset The unique identifier of the dataset.

* Table The name of the table.

8. Add dynamic values to the required fields.

For every required field, you need to add a dynamic value. This value is the output of the metadata of the dataflow run.

a. Select the field next to **Dataflow Name** and then select the lightning button.

b. Select **Dataflow Name** from the **Dynamic content** context box.

Add rows to a dataset

* Workspace DocTestWorkspace

* Dataset Dataflow Monitoring

* Table RealTimeData

Dataflow Name | Add dynamic content

Dataflow ID

Refresh Status

Refresh Type

Start Time

End Time

+ New step Save

Dynamic content Expression

Search dynamic content

When a dataflow refresh completes

Dataflow Id Id of the dataflow

Dataflow Name Name of the dataflow

Refresh Type Type of the dataflow refresh

Start Time Start time of the dataflow refresh

End Time Completion time of the dataflow refresh

Refresh Status Status of the dataflow refresh. Possible values are: 'Success', ...

c. Repeat this process for all required fields.

9. Save the flow.

Create a Power BI Report

1. Navigate to [Power BI](#).
2. Navigate to the streaming dataset (in this example, in the **DocTestWorkspace** workspace, from the **Dataflow Monitoring** dataset, select **Create Report**).
3. Create your own report on top of this data.

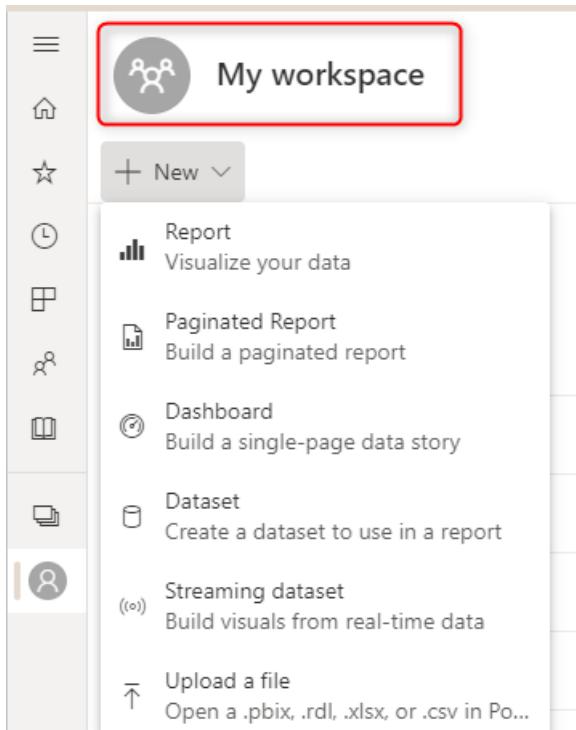
Troubleshooting dataflow issues: Creating dataflows

1/15/2022 • 2 minutes to read • [Edit Online](#)

This article explains some of the most common errors and issues you might get when you want to create a dataflow, and how to fix them.

I can't create a dataflow in My workspace

This problem happens when you try to create a dataflow in **My workspace** in Power BI.



Reason:

Creating dataflows in **My workspace** isn't supported.

Resolution:

Create your dataflows in [organizational workspaces](#). To learn how to create an organizational workspace, go to [Create the new workspaces in Power BI](#).

I can't create a dataflow in an organizational workspace where I have read-only rights

If you're a member of an organization workspace and you still can't create a dataflow, it might be because of your access rights in that workspace.

Reason:

You don't have edit rights in the workspace.

Resolution:

Ask the workspace administrators or members to give you an Admin, Member, or Contributor [role](#).

The screenshot shows the 'Access' section of the Power BI service. At the top, there's a yellow header with the text 'Access the new v2 sample'. Below it, a search bar says 'Enter email addresses'. A dropdown menu shows 'Member' selected. A list of users includes 'Member' (selected), 'Admin', 'Contributor', and 'Viewer'. The table below maps these names to their permissions: 'Member' is Admin, 'Contributor' is Contributor, and 'Viewer' has three dots. A red box highlights the dropdown and the list of users.

I can't create a dataflow in a workspace where I have edit rights

This problem happens when you're in an organizational workspace that you've created, or someone else has created and you have Admin, Member, or Contributor access. You want to create a dataflow in this scenario, but you can't.

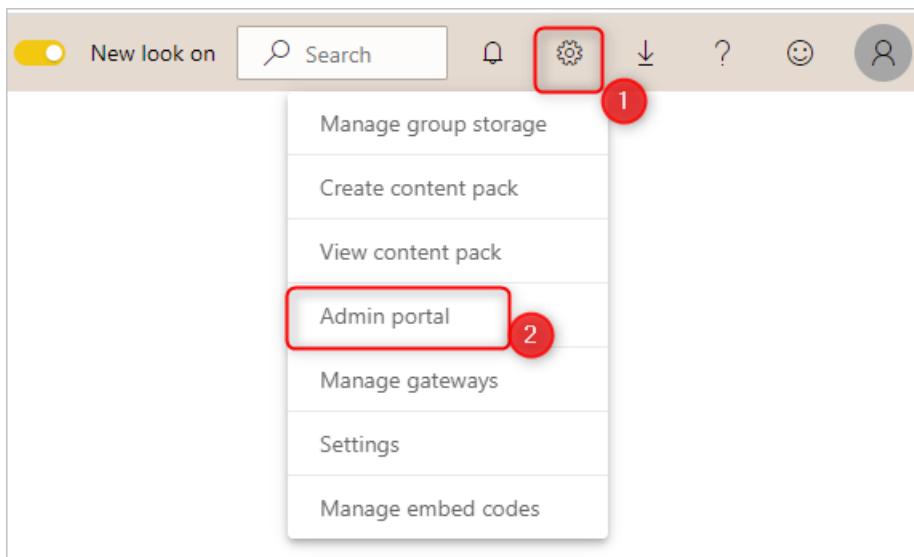
Reason:

The access to dataflow creation was disabled by the Power BI administrator.

Resolution:

Ask the Power BI tenant administrator to enable access for you by following these steps:

1. On the **Settings** menu in the Power BI service, select **Admin portal**.



2. On the left pane, select **Tenant settings**, and in the **Dataflow settings** section, turn on the toggle for **Enabled**. Then select **Apply**.

The screenshot shows the Microsoft Admin portal interface. On the left, a sidebar lists various administrative options: Usage metrics, Users, Audit logs, Tenant settings (which is highlighted with a red box and has a red number '1' next to it), Capacity settings, Refresh summary, Embed Codes, Organizational visuals, Dataflow settings, Workspaces, Custom branding, Protection metrics, and Featured content.

The main content area is titled "Developer settings" and contains two sections:

- Embed content in apps: Enabled for the entire organization
- Allow service principals to use Power BI APIs: Disabled for the entire organization

A red box highlights the "Dataflow settings" section, which includes the following configuration:

- Create and use dataflows: Unapplied changes (indicated by a red number '3' next to the status)

Below this, a note states: "Users in the organization can create and use dataflows. [Learn more](#)". There is a yellow "Enabled" toggle switch (indicated by a red number '2') and a "Dataflow settings" link.

At the bottom of the "Dataflow settings" section, there are "Apply" and "Cancel" buttons, with a red number '3' next to the "Apply" button. A tooltip below the "Apply" button says: "This setting applies to the entire organization".

At the very bottom of the main content area, there is a link to "Template app settings".

I only see limited options when I create a dataflow

When creating a dataflow, sometimes you don't see all the options that are available. For example, you might see only the options shown in the following image.

Start creating your dataflow

Define new entities

Choose a data source to define the entities for your dataflow. You can map your data to [standard Common Data Model entities](#), or define custom entities instead.

[Learn more](#)

Add new entities

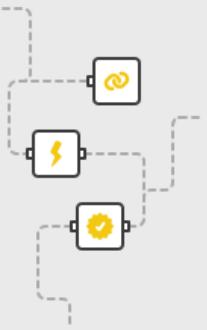
Import Model

Choose a dataflow model to import into your workspace.

[Learn more](#)

Import model

However, more options are actually available, as shown in the following image.



Start creating your dataflow

Define new entities

Choose a data source to define the entities for your dataflow. You can map your data to [standard Common Data Model](#) entities, or define custom entities instead.

[Learn more](#)

[Add new entities](#)

Link entities from other dataflows

Linking to entities from other dataflows reduces duplication and helps maintain consistency across your organization.

[Learn more](#)

[Add linked entities](#)

Import Model

Choose a dataflow model to import into your workspace.

[Learn more](#)

[Import model](#)

Attach a Common Data Model folder (preview)

Attach a Common Data Model folder from your Azure Data Lake Storage Gen2 account to a new dataflow, so you can use it in Power BI.

[Learn more](#)

[Create and attach](#)

Reason:

You're creating the dataflow in an old version of the Power BI workspace, called V1.

Resolution:

Upgrade your Power BI workspace to the new version (v2). More information: [Upgrade classic workspaces to the new workspaces in Power BI](#)

Edit workspace

Name

New WP

Privacy

Private - Only approved members can see what's inside

Members can edit Power BI content

Workspace members

Enter email addresses

Add

microsoft.com

Admin

▼



Advanced ^

Dedicated capacity ⓘ

Off

Upgrade this workspace (preview)

Get new features and improved security.

[Upgrade now](#) | [Learn more](#)

Delete workspace

Save

Cancel

Troubleshooting dataflow issues: Get data from a dataflow

1/15/2022 • 2 minutes to read • [Edit Online](#)

You might have created a dataflow but then had difficulty getting data from it (either by using Power Query in Power BI Desktop or from other dataflows). This article explains some of the most common problems with getting data from a dataflow.

Error: This table is empty

Let's assume that you're getting data from a dataflow (either in Power BI Desktop or in another dataflow), and you have access to that dataflow. Sometimes, however, when you get data from a dataflow in the above situation, you get a message in the **Navigator** window saying "This table is empty."

Reason:

The data wasn't loaded into the table.

Resolution:

In the desktop tools, such as Power Query in Excel and Power Query in Power BI Desktop, the loading of data into tables happens automatically (unless you disable it). This behavior is a bit different in Power Query in dataflows. In dataflow entities, the data won't be loaded unless you refresh the data.

You have to set up a scheduled refresh for a dataflow, or—if you want to just have a single refresh—use the manual refresh option.



After a dataflow is refreshed, the data in entities will be visible in the **Navigator** window of other tools and services.

More information: [Refreshing a dataflow in Power BI](#) and [Set the refresh frequency in Power Apps](#)

My Microsoft Power Platform dataflow isn't listed

Sometimes, you have a Microsoft Power Platform dataflow you created and also refreshed, but you still can't access it through the **Get data** command. This might be because the account that's trying to access the dataflow doesn't have access. However, if the account does have access to the dataflow, another reason might be the type of dataflow you're accessing.

You might receive the error message "We reached the end of the buffer" or "DataFormat.Error: We reached the end of the buffer".

Navigator

The screenshot shows the Power Platform Navigator interface. On the left, there's a tree view under 'Power Platform dataflows [2]'. One item in this tree has a red border around it. On the right, there's a large red-bordered box containing an 'Error' message: 'DataFormat.Error: We reached the end of the buffer.' with a warning icon.

Reason:

Only analytical dataflows can be used in a **Get data** operation from a dataflow.

Resolution:

If you've created a dataflow that stores data in Dataverse—that is, a standard dataflow—you can't see it by using the **Get data** operation from a dataflow. However, you can use [Get data from Dataverse](#) to access it. Or you can create an [analytical dataflow](#) instead, and then access it by using **Get data** from a dataflow.

The screenshot shows the 'Get Data' screen. On the left, there's a 'Search' bar and a sidebar with categories: 'Standard', 'Database', 'Power Platform' (which is selected), 'Azure', and 'Online Services'. A red speech bubble labeled 'Standard' points to the 'Standard' category. Another red speech bubble labeled 'Analytical' points to the 'Power Platform' category. On the right, there's a 'Power Platform' section with options: 'Power BI datasets', 'Power BI dataflows', 'Common Data Service (Legacy)', 'Dataverse' (which is highlighted with a red box), and 'Power Platform dataflows'.

I can't make a DirectQuery connection to the dataflow

If you intend to use the dataflow as a DirectQuery source, you might need to enable it first.

Reason:

The enhanced compute engine settings are disabled.

Resolution:

[Enable the enhanced compute engine](#), and then you'll have the option to connect to the dataflow by using

DirectQuery.

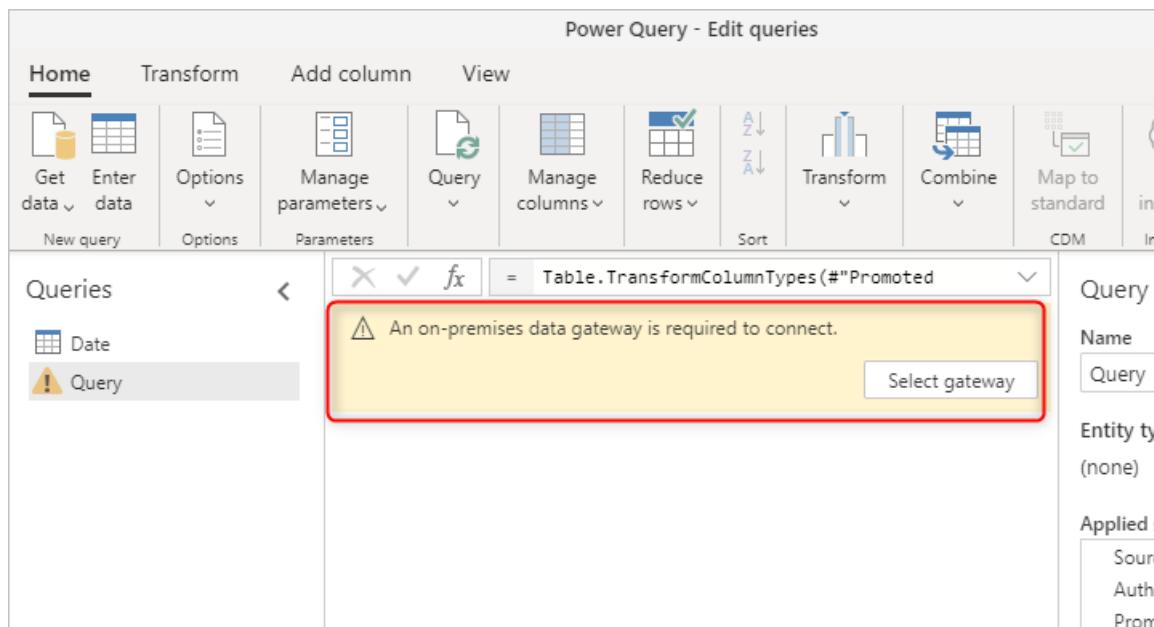
Troubleshooting dataflow issues: Connection to the data source

1/15/2022 • 2 minutes to read • [Edit Online](#)

When you create a dataflow, sometimes you get an error connecting to the data source. This error can be caused by the gateway, credentials, or other reasons. This article explains the most common connection errors and problems, and their resolution.

Error: An on-premises data gateway is required to connect

This problem can happen when you move a query from Power Query in desktop tools to Power Query in the dataflow, and you get the error "An on-premises data gateway is required to connect."



Reason:

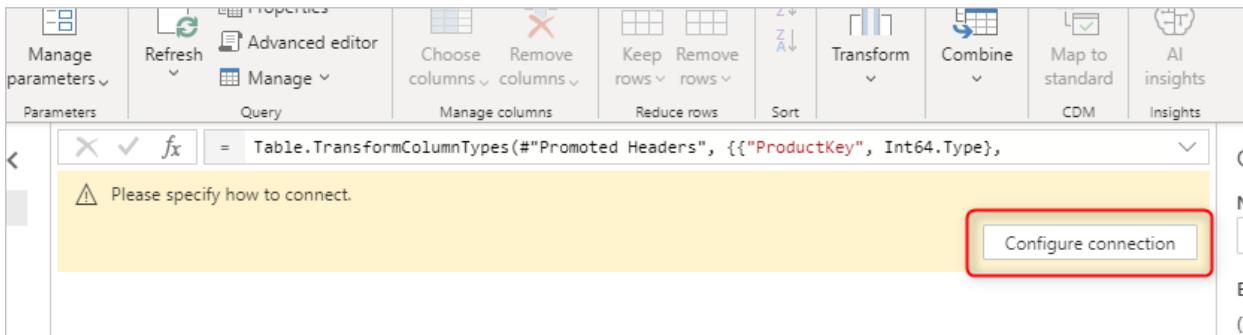
When your entity in the dataflow gets data from an on-premises data source, a gateway is needed for the connection, but the gateway hasn't been selected.

Resolution:

Select **Select gateway**. If the gateway hasn't been set up yet, see [Install an on-premises data gateway](#).

Error: Please specify how to connect

This problem happens when you're connected to a data source, but haven't set up the credentials or connection details yet. It can happen when you migrate queries into a dataflow.



Reason:

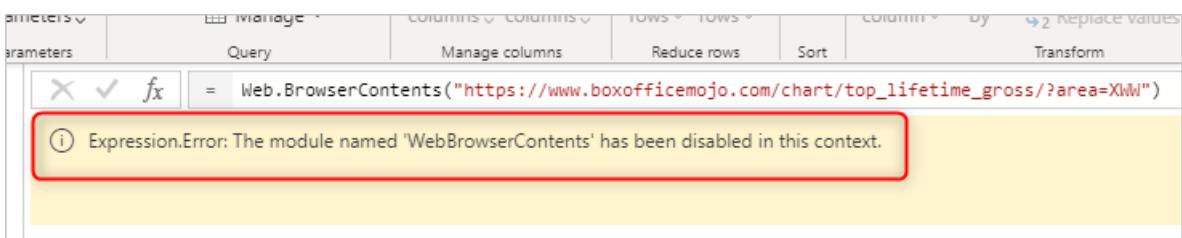
The connection details aren't set up correctly.

Resolution:

Select **Configure connection**. Set up the connection details and credentials.

Expression.Error: The module named 'xyz' has been disabled in this context

Sometimes, when you migrate your queries from Power Query in desktop tools to the dataflow, you get an error saying that a module is disabled in this context. One example of this situation is when your query uses functions such as `Web.Page` or `Web.BrowserContents`.



Reason:

Disabled modules are related to functions that require an on-premises data gateway connection to work. Even if the function is getting data from a webpage, because of some security compliance requirements, it needs to go through a gateway connection.

Resolution:

First, [install and set up an on-premises gateway](#). Then add a web data source for the web URL you're connecting to.

The screenshot shows the 'Data Source Settings' dialog. At the top, there are tabs for 'Data Source Settings' (which is selected) and 'Users'. Below the tabs, a green checkmark icon indicates 'Connection Successful'. The 'Data Source Name' field contains 'website'. A large red box highlights the 'Data Source Type' section, which is set to 'Web'. Under 'URL', the value is 'https://[REDACTED].com'. The 'Authentication Method' dropdown is set to 'Anonymous'. There is also a checked checkbox for 'Skip Test Connection'. At the bottom of the dialog are 'Apply' and 'Discard' buttons.

After adding the web data source, you can select the gateway in the dataflow from Options > Project options.

The screenshot shows the Power BI ribbon with the 'Home' tab selected. In the 'Transform' section, there is a 'Options' button with a dropdown menu. The 'Project options' item in this dropdown is highlighted with a red box. Other items in the dropdown include 'Manage parameters' and 'Global options'. The 'Queries' section below the ribbon shows three items: 'Service Calls Data', 'Query', and 'Query (2)'.

You might be asked to set up credentials. When you've set up the gateway and your credentials successfully, the modules will no longer be disabled."

The screenshot shows the Power BI ribbon with the 'Text tools' tab selected. The 'Transform' section includes buttons for 'To table', 'Split text', 'Format text', 'Extract', and 'Parse'. The 'Queries' section on the left lists 'Service Calls Data', 'Query', and 'Query (2)', with 'Query (2)' currently selected. The main area displays raw HTML code: '<html lang="en-US" class="jetpack-lazy-images-js-enabled"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-width, initial-scale=1"><link rel="profile" href="https://gmpg.org/xfn/11"><link rel="pingback" href="https://[REDACTED].com/xmlrpc.php">'.

Keyboard shortcuts in Power Query

1/15/2022 • 2 minutes to read • [Edit Online](#)

Keyboard shortcuts provide a quick way to navigate and allow users to work more efficiently. For users with mobility or vision disabilities, keyboard shortcuts can be easier than using the touchscreen, and are an essential alternative to using the mouse. The table in this article lists all the shortcuts available in Power Query Online.

When using the Query Editor in Power Query Online, you can press Ctrl+? or navigate to the **Keyboard shortcuts** button in the **Help** tab to view the list of keyboard shortcuts.

Query Editor

ACTION	KEYBOARD SHORTCUT
Get Data	Ctrl+Alt+D
Enter Data	Ctrl+Alt+T
Add custom column	Ctrl+Alt+C
Choose column	Ctrl+K
Go to column	Ctrl+G
Add column from examples	Ctrl+E
Blank query	Ctrl+M
Advanced editor	Ctrl+Shift+M
Refresh	Alt+F5

Data Preview

ACTION	KEYBOARD SHORTCUT
Copy cells/rows/columns	Ctrl+C
Select all cells	Ctrl+A
Select column	Ctrl+Space

When the focus is on the column header

ACTION	KEYBOARD SHORTCUT
Filter menu	Alt+Down arrow key
Change column type menu	Ctrl+Down arrow key

ACTION	KEYBOARD SHORTCUT
Move focus to column header on the left	Ctrl+Left arrow key
Move focus to column header on the right	Ctrl+Right arrow key
Select first column	Home
Select last column	End

When the focus is on the cell

ACTION	KEYBOARD SHORTCUT
Select first cell of the row	Home
Select last cell of the row	End
Select first cell of the column	Alt+Home
Select last cell of the column	Alt+End
Select first cell of the first row	Ctrl+Home
Select last cell of the last row	Ctrl+End
Select the cell one page up	Page up
Select the cell one page down	Page down

Editable Grid (Enter data/Blank table)

ACTION	KEYBOARD SHORTCUT
Copy cells/rows/columns	Ctrl+C
Paste cells/rows/columns	Ctrl+V
Save entered data	Ctrl+Enter
Show suggestions (when available)	Ctrl+Space

Multi-line text editor (Blank query/Advanced editor)

ACTION	KEYBOARD SHORTCUT
Toggle tab behavior	Ctrl+M

Diagram View

ACTION	KEYBOARD SHORTCUT
Expand selected query	Ctrl+Right arrow key
Collapse selected query	Ctrl+Left arrow key
Move focus from query level to step level	Alt+Down arrow key
Move focus from step level to query level	Esc
Expand all queries	Ctrl+Shift+Right arrow key
Collapse all queries	Ctrl+Shift+Left arrow key
Insert new step using + button (after selected step)	Ctrl+Alt+N
Highlight related queries	Ctrl+Alt+R
Select all queries	Ctrl+A
Copy queries	Ctrl+C
Paste queries	Ctrl+V

Queries pane

ACTION	KEYBOARD SHORTCUT
Select all queries	Ctrl+A
Copy queries	Ctrl+C
Paste queries	Ctrl+V
Select multiple consecutive queries	Ctrl+Up arrow key and Ctrl+Down arrow key

Best practices when working with Power Query

1/15/2022 • 11 minutes to read • [Edit Online](#)

This article contains some tips and tricks to make the most out of your data wrangling experience in Power Query.

Choose the right connector

Power Query offers a vast number of data connectors. These connectors range from data sources such as TXT, CSV, and Excel files, to databases such as Microsoft SQL Server, and popular SaaS services such as Microsoft Dynamics 365 and Salesforce. If you don't see your data source listed in the **Get Data** window, you can always use the ODBC or OLEDB connector to connect to your data source.

Using the best connector for the task will provide you with the best experience and performance. For example, using the SQL Server connector instead of the ODBC connector when connecting to a SQL Server database not only provides you with a much better **Get Data** experience, but the SQL Server connector also offers you features that can improve your experience and performance, such as query folding. To read more about query folding, see [Power Query query folding](#).

Each data connector follows a standard experience as explained in [Getting data](#). This standardized experience has a stage called **Data Preview**. In this stage, you're provided with a user-friendly window to select the data that you want to get from your data source, if the connector allows it, and a simple data preview of that data. You can even select multiple datasets from your data source through the **Navigator** window, as shown in the following image.

The screenshot shows the Power Query Navigator window. On the left, there's a tree view of available datasets under the 'AdventureWorks2012 [93]' connection. The 'HumanResources.vEmployee' dataset is selected. On the right, a preview grid displays the first 22 rows of the 'vEmployee' table. The columns are labeled: BusinessEntityID, Title, FirstName, and MiddleName. The data includes names like Ken, Terri, Roberto, Rob, Gail, Jossef, Dylan, Diane, Gigi, Michael, Ovidiu, Thierry, Janice, Michael, Sharon, David, Kevin, John, Mary, Wanida, Terry, and Sariya. At the bottom of the window, there are buttons for 'Select Related Tables', 'Load', 'Transform Data', and 'Cancel'.

BusinessEntityID	Title	FirstName	MiddleName
1	null	Ken	J
2	null	Terri	Lee
3	null	Roberto	
4	null	Rob	
5	Ms.	Gail	A
6	Mr.	Jossef	H
7	null	Dylan	A
8	null	Diane	L
9	null	Gigi	N
10	null	Michael	
11	null	Ovidiu	V
12	null	Thierry	B
13	Ms.	Janice	M
14	null	Michael	I
15	null	Sharon	B
16	null	David	M
17	null	Kevin	F
18	null	John	L
19	null	Mary	A
20	null	Wanida	M
21	null	Terry	J
22	null	Sariya	E

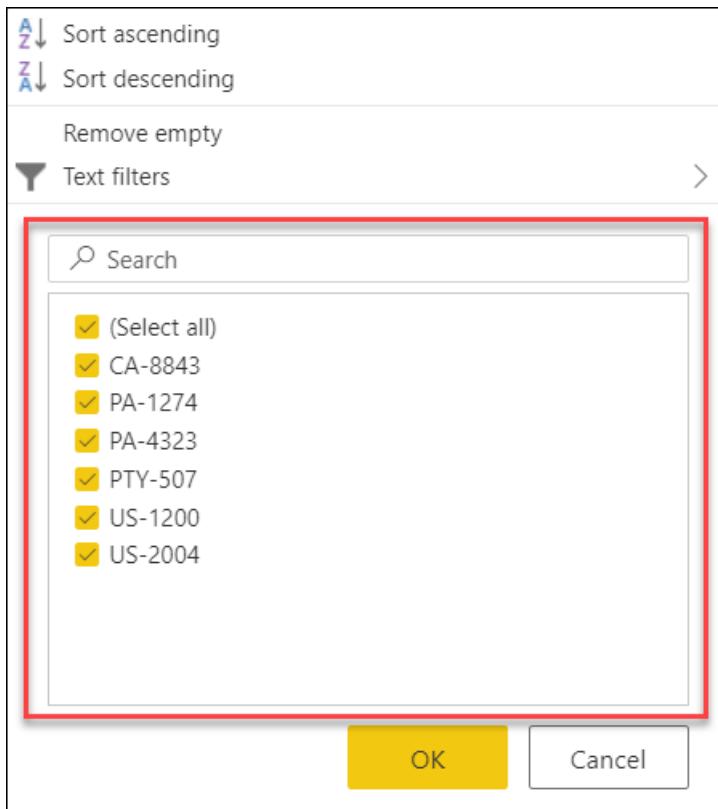
NOTE

To see the full list of available connectors in Power Query, see [Connectors in Power Query](#).

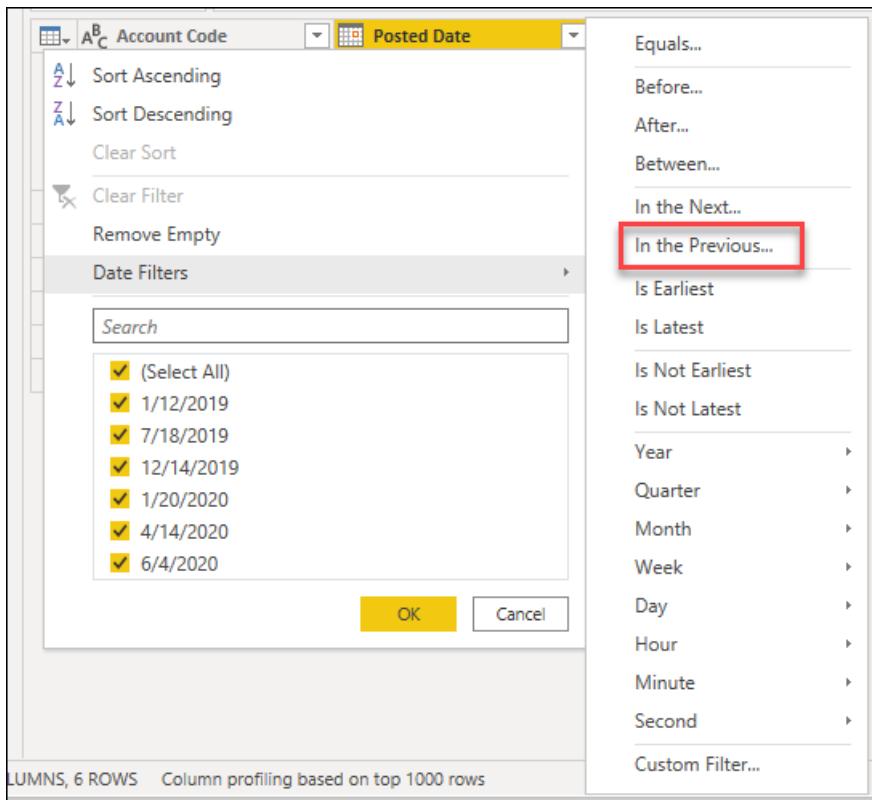
Filter early

It's always recommended to filter your data in the early stages of your query or as early as possible. Some connectors will take advantage of your filters through query folding, as described in [Power Query query folding](#). It's also a best practice to filter out any data that isn't relevant for your case. This will let you better focus on your task at hand by only showing data that's relevant in the data preview section.

You can use the auto filter menu that displays a distinct list of the values found in your column to select the values that you want to keep or filter out. You can also use the search bar to help you find the values in your column.



You can also take advantage of the type-specific filters such as **In the previous** for a date, datetime, or even date timezone column.



These type-specific filters can help you create a dynamic filter that will always retrieve data that's in the previous x number of seconds, minutes, hours, days, weeks, months, quarters, or years as showcased in the following image.

The screenshot shows the 'Filter rows' dialog in Power BI. It has a 'Basic' filter condition: 'Keep rows where "Date" is in the previous 90 days'. A dropdown menu is open next to 'days', listing 'years', 'quarters', 'months', 'weeks', 'days', 'hours', 'minutes', and 'seconds'. The 'days' option is currently selected.

NOTE

To learn more about filtering your data based on values from a column, see [Filter by values](#).

Do expensive operations last

Certain operations require reading the full data source in order to return *any* results, and will thus be slow to preview in the Power Query Editor. For example, if you perform a sort, it's possible that the first few sorted rows are at the end of the source data. So in order to return any results, the sort operation must first read *all* the rows.

Other operations (such as filters) do not need to read all the data before returning any results. Instead, they operate over the data in what's called a "streaming" fashion. The data "streams" by, and results are returned

along the way. In the Power Query Editor, such operations only need to read enough of the source data to populate the preview.

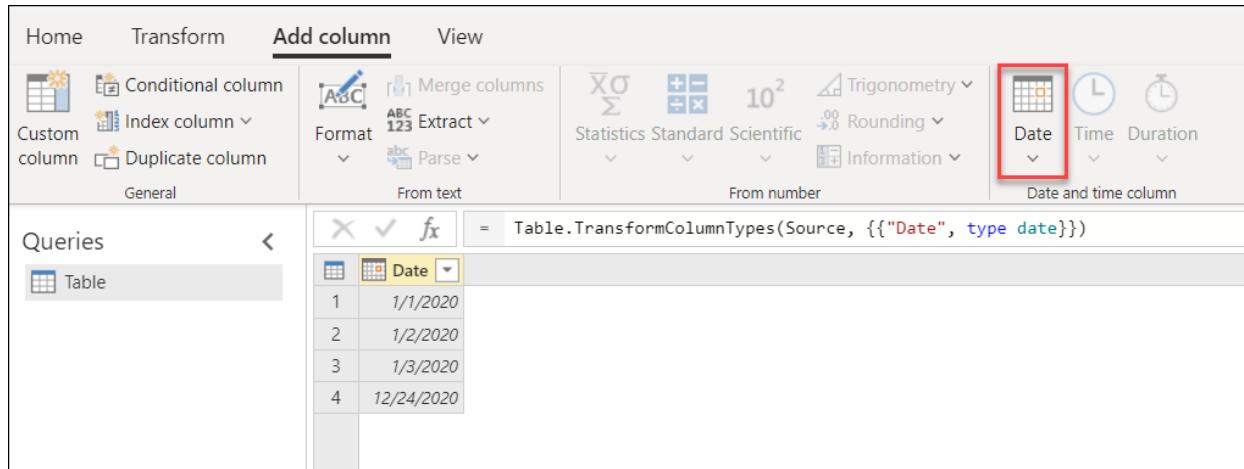
When possible, perform such streaming operations first, and do any more expensive operations last. This will help minimize the amount of time you spend waiting for the preview to render each time you add a new step to your query.

Temporarily work against a subset of your data

If adding new steps to your query in the Power Query Editor is slow, consider first doing a "Keep First Rows" operation and limiting the number of rows you're working against. Then, once you've added all the steps you need, remove the "Keep First Rows" step.

Use the correct data types

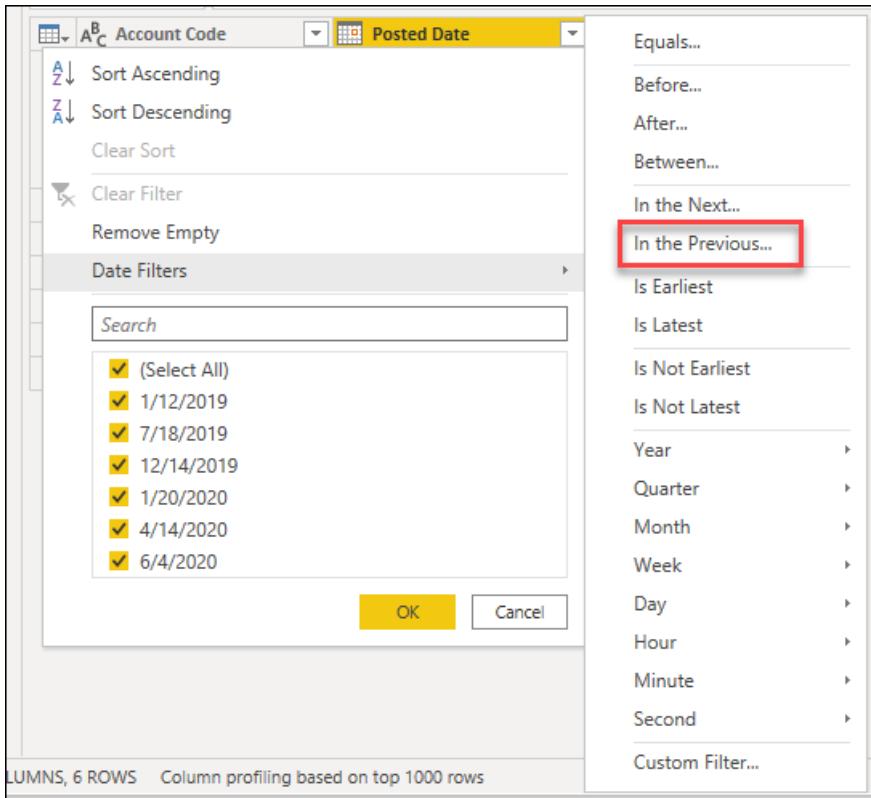
Some features in Power Query are contextual to the data type of the column selected. For example, when selecting a date column, the available options under the **Date and time column** group in the **Add Column** menu will be available. But if the column doesn't have a data type set, then these options will be greyed out.



The screenshot shows the Power Query Editor interface. The ribbon at the top has 'Home', 'Transform', 'Add column' (which is selected), and 'View'. Below the ribbon, there are several tabs: 'Conditional column', 'Index column', 'Duplicate column', 'Custom column', 'General', 'Merge columns', 'Format', 'Parse', 'From text', 'From number', 'Trigonometry', 'Rounding', 'Information', and 'Date and time column'. The 'Date and time column' tab is highlighted with a red box. Below the ribbon, there's a 'Queries' pane on the left with 'Table' selected. On the right, there's a formula bar with the formula: `= Table.TransformColumnTypes(Source, {"Date", type date})`. Below the formula is a table with four rows:

	Date
1	1/1/2020
2	1/2/2020
3	1/3/2020
4	12/24/2020

A similar situation occurs for the type-specific filters, since they're specific to certain data types. If your column doesn't have the correct data type defined, these type-specific filters won't be available.



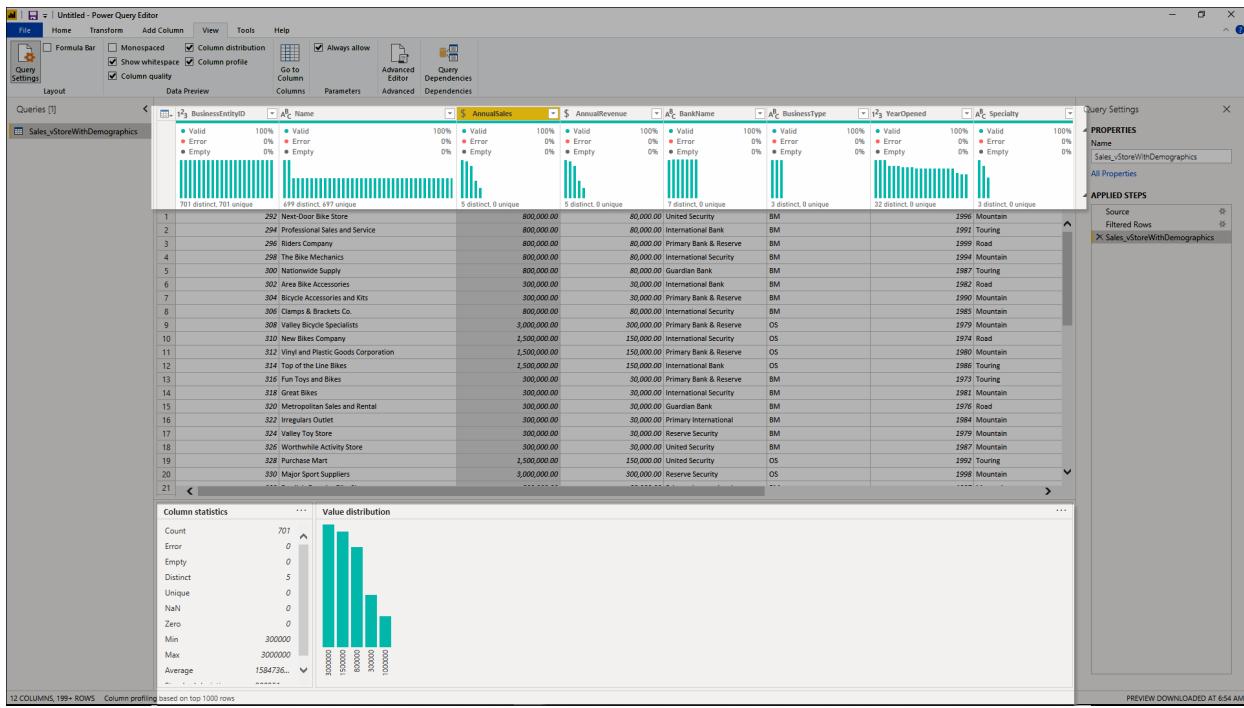
It's crucial that you always work with the correct data types for your columns. When working with structured data sources such as databases, the data type information will be brought from the table schema found in the database. But for unstructured data sources such as TXT and CSV files, it's important that you set the correct data types for the columns coming from that data source. By default, Power Query offers an automatic data type detection for unstructured data sources. You can read more about this feature and how it can help you in [Data types](#).

NOTE

To learn more about the importance of data types and how to work with them, see [Data types](#).

Explore your data

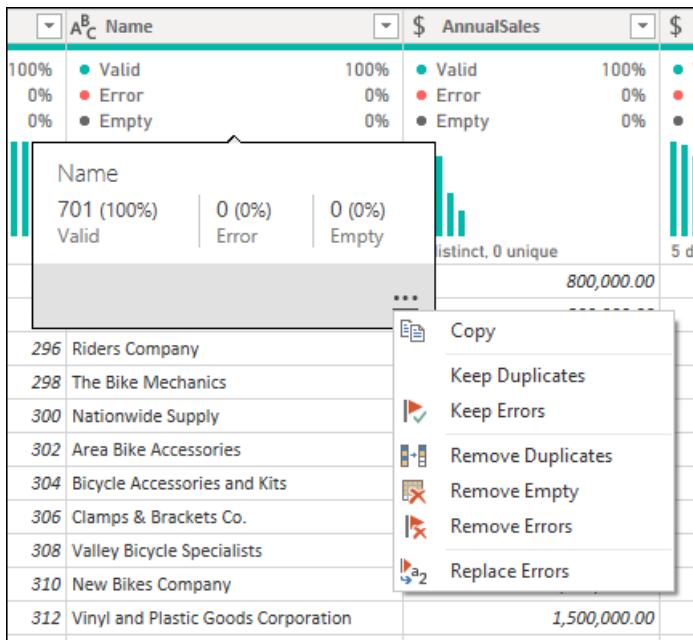
Before you start preparing your data and adding new transformation steps, we recommend that you enable the Power Query [data profiling tools](#) to easily discover information about your data.



These data profiling tools help you better understand your data. The tools provide you with small visualizations that show you information on a per column basis, such as:

- **Column quality**—Provides a small bar chart and three indicators with the representation of how many values in the column fall under the categories of valid, error, or empty values.
- **Column distribution**—Provides a set of visuals underneath the names of the columns that showcase the frequency and distribution of the values in each of the columns.
- **Column profile**—Provides a more thorough view of your column and the statistics associated to it.

You can also interact with these features, which will help you prepare your data.



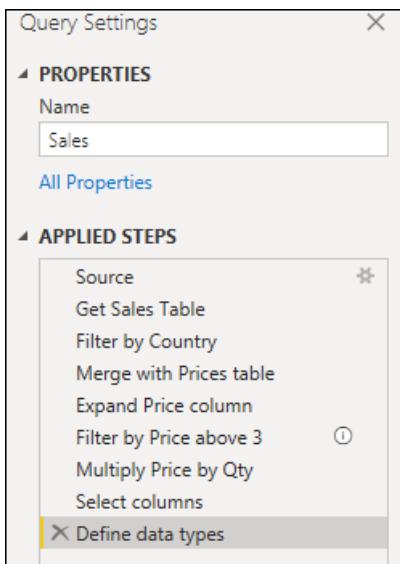
NOTE

To learn more about the data profiling tools, see [Data profiling tools](#).

Document your work

We recommend that you document your queries by renaming or adding a description to your steps, queries, or groups as you see fit.

While Power Query automatically creates a step name for you in the applied steps pane, you can also rename your steps or add a description to any of them.



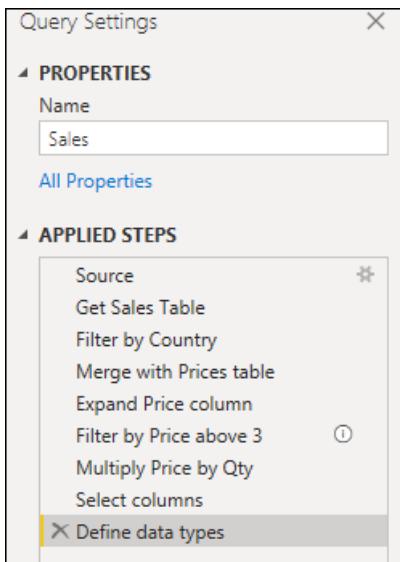
NOTE

To learn more about all the available features and components found inside the applied steps pane, see [Using the Applied steps list](#).

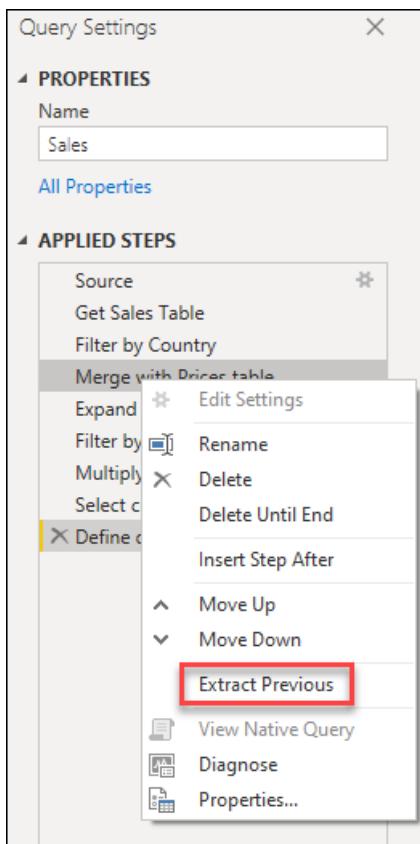
Take a modular approach

It's entirely possible to create a single query that contains all the transformations and calculations that you may need. But if the query contains a large number of steps, then it might be a good idea to split the query into multiple queries, where one query references the next. The goal of this approach is to simplify and decouple transformation phases into smaller pieces so they're easier to understand.

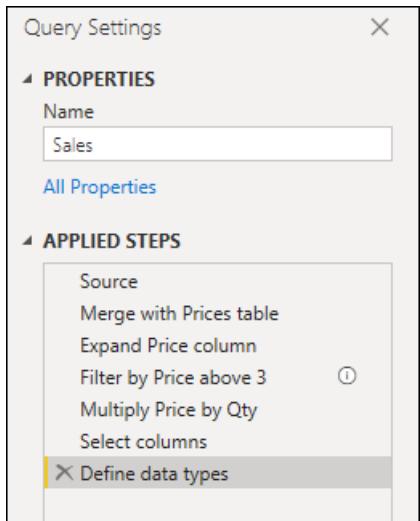
For example, say you have a query with the nine steps shown in the following image.



You could split this query into two at the **Merge with Prices table** step. That way it's easier to understand the steps that were applied to the sales query before the merge. To do this operation, you right-click the **Merge with Prices table** step and select the **Extract Previous** option.



You'll then be prompted with a dialog to give your new query a name. This will effectively split your query into two queries. One query will have all the queries before the merge. The other query will have an initial step that will reference your new query and the rest of the steps that you had in your original query from the **Merge with Prices table** step downward.



You could also leverage the use of query referencing as you see fit. But it's a good idea to keep your queries at a level that doesn't seem daunting at first glance with so many steps.

NOTE

To learn more about query referencing, see [Understanding the queries pane](#).

Create groups

A great way to keep your work organized is by leveraging the use of groups in the queries pane.

The screenshot shows the Power Query Editor interface. On the left, the 'Queries' pane lists three items: 'Customers' (selected), 'Customers', and 'Employees'. A context menu is open over the first 'Customers' item, with 'Move to group' highlighted. The 'Move to group' submenu shows 'Customers' as the current group, with other options like 'Queries (root)' and 'New group...' available. To the right, a table preview shows data from a source, with columns 'CustomerID' and 'CompanyName' visible.

The sole purpose of groups is to help you keep your work organized by serving as folders for your queries. You can create groups within groups should you ever need to. Moving queries across groups is as easy as drag and drop.

Try to give your groups a meaningful name that makes sense to you and your case.

NOTE

To learn more about all the available features and components found inside the queries pane, see [Understanding the queries pane](#).

Future-proofing queries

Making sure that you create a query that won't have any issues during a future refresh is a top priority. There are several features in Power Query to make your query resilient to changes and able to refresh even when some components of your data source changes.

It's a best practice to define the scope of your query as to what it should do and what it should account for in terms of structure, layout, column names, data types, and any other component that you consider relevant to the scope.

Some examples of transformations that can help you make your query resilient to changes are:

- If your query has a dynamic number of rows with data, but a fixed number of rows that serve as the footer that should be removed, you can use the **Remove bottom rows** feature.

NOTE

To learn more about filtering your data by row position, see [Filter a table by row position](#).

- If your query has a dynamic number of columns, but you only need to select specific columns from your dataset, you can use the **Choose columns** feature.

NOTE

To learn more about choosing or removing columns, see [Choose or remove columns](#).

- If your query has a dynamic number of columns and you need to unpivot only a subset of your columns, you can use the **unpivot only selected columns** feature.

NOTE

To learn more about the options to unpivot your columns, see [Unpivot columns](#).

- If your query has a step that changes the data type of a column, but some cells yield errors as the values don't conform to the desired data type, you could remove the rows that yielded error values.

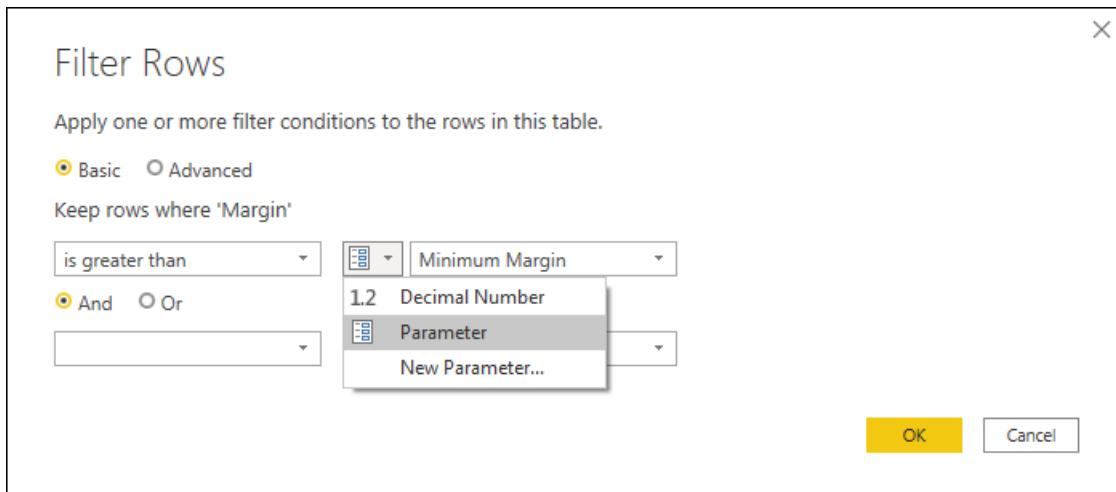
NOTE

To more about working and dealing with errors, see [Dealing with errors](#).

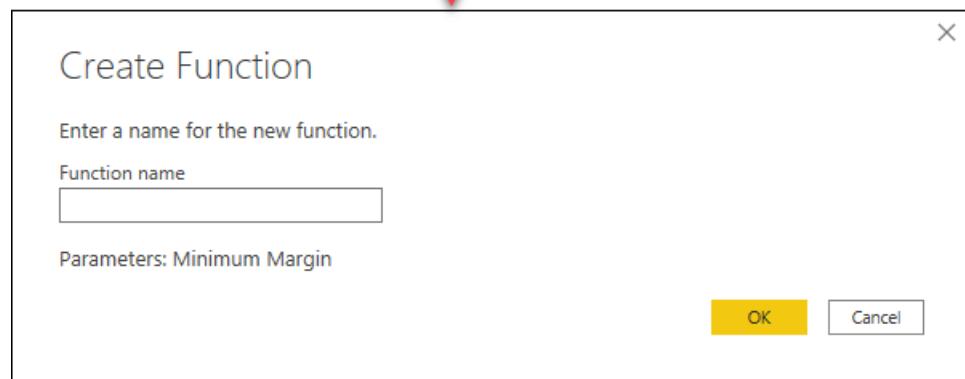
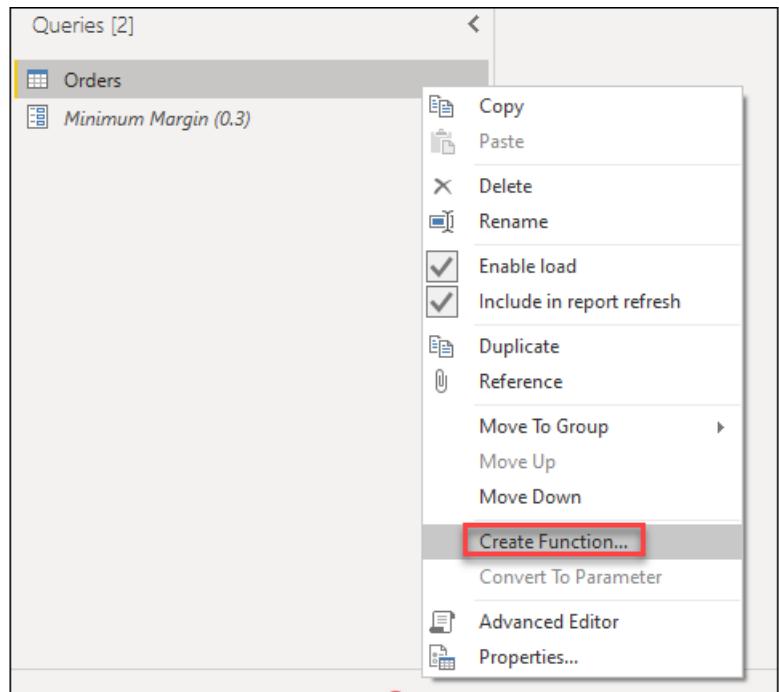
Use parameters

Creating queries that are dynamic and flexible is a best practice. Parameters in Power Query help you make your queries more dynamic and flexible. A parameter serves as a way to easily store and manage a value that can be reused in many different ways. But it's more commonly used in two scenarios:

- **Step argument**—You can use a parameter as the argument of multiple transformations driven from the user interface.

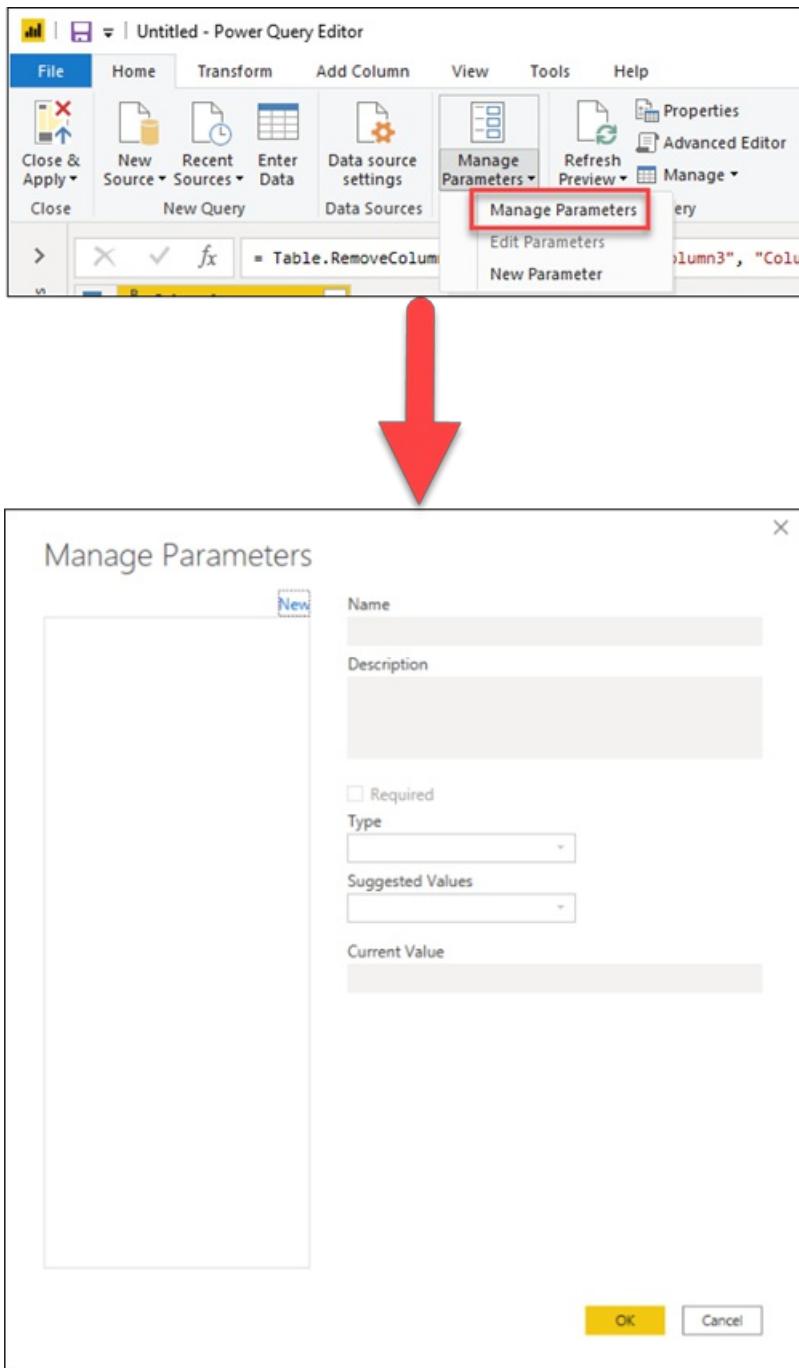


- **Custom Function argument**—You can create a new function from a query, and reference parameters as the arguments of your custom function.



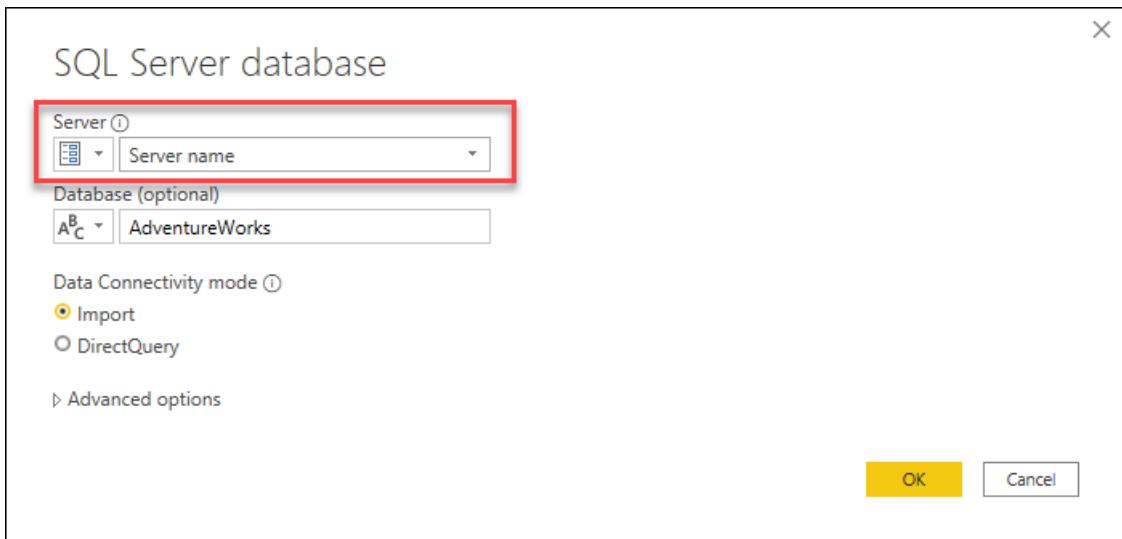
The main benefits of creating and using parameters are:

- Centralized view of all your parameters through the **Manage Parameters** window.



- Reusability of the parameter in multiple steps or queries.
- Makes the creation of custom functions straightforward and easy.

You can even use parameters in some of the arguments of the data connectors. For example, you could create a parameter for your server name when connecting to your SQL Server database. Then you could use that parameter inside the SQL Server database dialog.



If you change your server location, all you need to do is update the parameter for your server name and your queries will be updated.

NOTE

To learn more about creating and using parameters, see [Using parameters](#).

Create reusable functions

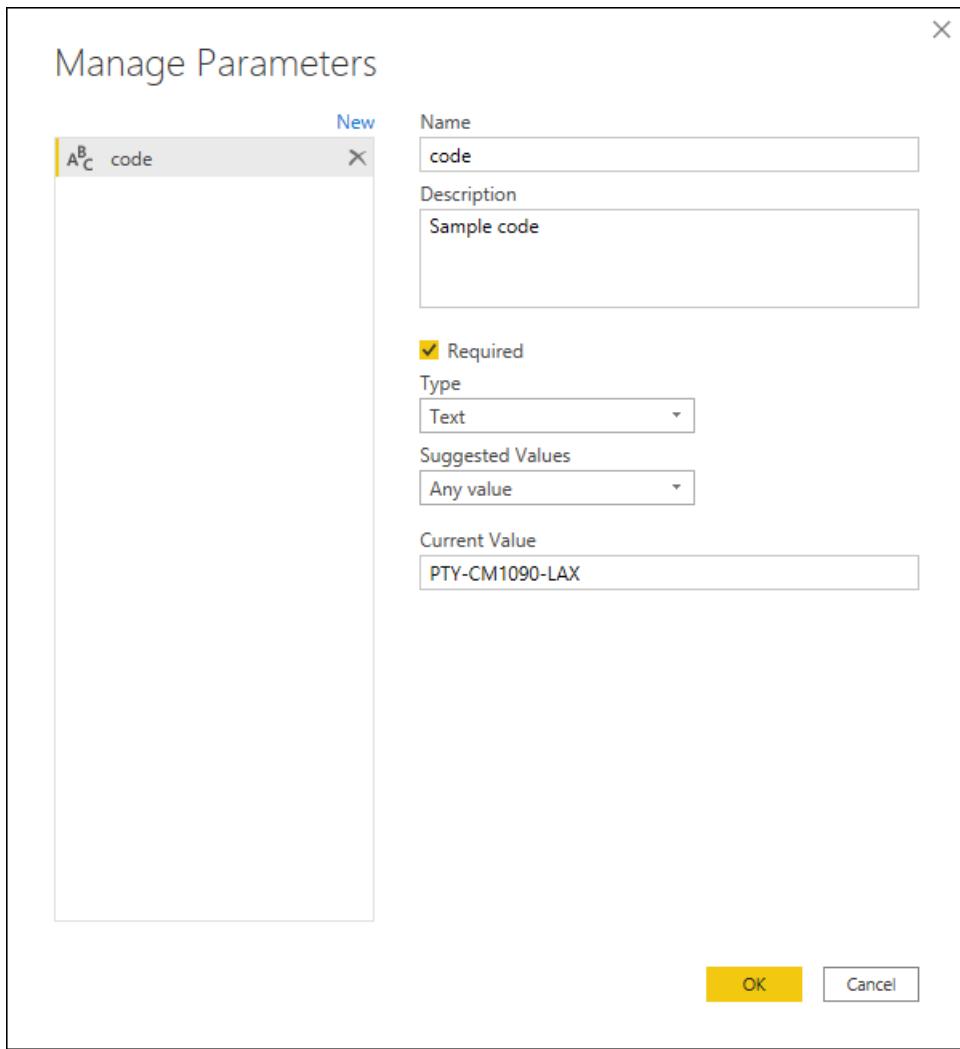
If you find yourself in a situation where you need to apply the same set of transformations to different queries or values, creating a Power Query custom function that can be reused as many times as you need could be beneficial. A Power Query custom function is a mapping from a set of input values to a single output value, and is created from native M functions and operators.

For example, say you have multiple queries or values that require the same set of transformations. You could create a custom function that later could be invoked against the queries or values of your choice. This custom function would save you time and help you in managing your set of transformations in a central location, which you can modify at any moment.

Power Query custom functions can be created from existing queries and parameters. For example, imagine a query that has several codes as a text string and you want to create a function that will decode those values.

A ^B C code	
	5 distinct, 5 unique
1	PTY-CM1090-LAX
2	LAX-CM701-PTY
3	PTY-CM4441-MIA
4	MIA-UA1257-LAX
5	LAX-XY2842-MIA

You start by having a parameter that has a value that serves as an example.



From that parameter, you create a new query where you apply the transformations that you need. For this case, you want to split the code PTY-CM1090-LAX into multiple components:

- **Origin** = PTY
- **Destination** = LAX
- **Airline** = CM
- **FlightID** = 1090

Origin	Destination	Airline	FlightID
1 distinct, 1 unique 1 PTY	1 distinct, 1 unique 1 LAX	1 distinct, 1 unique 1 CM	1 distinct, 1 unique 1 1090

You can then transform that query into a function by doing a right-click on the query and selecting **Create Function**. Finally, you can invoke your custom function into any of your queries or values, as shown in the following image.

The screenshot shows the Power Query Editor interface. On the left, the 'Queries [4]' pane lists 'fxDecode [3]' and 'Other Queries [1]'. The 'fxDecode' folder contains three items: 'code (PTY-CM1090-LAX)', 'Decode', and 'fxDecode'. The 'List of codes' item under 'Other Queries' is currently selected. In the center, a preview window shows the 'code' column with five distinct values. A floating dialog box titled 'Invoke Custom Function' is open, prompting the user to define a new column name ('fxDecode'), select a function query ('fxDecode'), and choose the source column ('code'). The 'OK' button is highlighted.

After a few more transformations, you can see that you've reached your desired output and leveraged the logic for such a transformation from a custom function.

The screenshot shows the final state of the Power Query Editor. The main area displays a table with five columns: 'code', 'Origin', 'Destination', 'Airline', and 'FlightID'. The 'code' column has been expanded into five separate columns. The 'Properties' pane on the right shows the 'Name' is 'List of codes' and the 'Applied Steps' list includes 'Source', 'Changed Type', 'Invoked Custom Function', 'Expanded fxDecode', and 'Changed Type1'.

NOTE

To learn more about how to create and use custom functions in Power Query from the article [Custom Functions](#).

Power Query feedback

1/15/2022 • 2 minutes to read • [Edit Online](#)

This article describes how to get support or submit feedback for Power Query.

For **Power Query connectors**, go to [Feedback and support for Power Query connectors](#).

For **Power Query documentation**, you can submit feedback through the **Submit and view feedback for - This page** link at the bottom of each article.

Support and troubleshooting

If you're finding an issue with Power Query, use the dedicated support channels for the product you're using Power Query in. For example, for Power BI, visit the [Power BI support page](#).

You can also use any of the following community resources:

- [Power Query on Microsoft Q&A](#)
- Community forums for the product you're using Power Query in. For example, for Power BI, this forum would be the [Power BI Community](#)
- [Power Query website resources](#)

Submitting feedback

To submit feedback about Power Query, provide the feedback to the "ideas" forum for the product you're using Power Query in. For example, for Power BI, visit the [Power BI ideas forum](#). If you have one, you can also provide feedback directly to your Microsoft account contact.

Power Query query folding

1/15/2022 • 4 minutes to read • [Edit Online](#)

This article targets data modelers developing models in Power Pivot or Power BI Desktop. It describes what Power Query query folding is, and why it's important in your data model designs. This article also describes the data sources and transformations that can achieve query folding, and how to determine that your Power Query queries can be folded—whether fully or partially.

Query folding is the ability for a Power Query query to generate a single query statement to retrieve and transform source data. The Power Query mashup engine strives to achieve query folding whenever possible for reasons of efficiency.

Query folding is an important topic for data modeling for several reasons:

- **Import model tables:** Data refresh will take place efficiently for Import model tables (Power Pivot or Power BI Desktop), in terms of resource utilization and refresh duration.
- **DirectQuery and Dual storage mode tables:** Each DirectQuery and Dual storage mode table (Power BI only) must be based on a Power Query query that can be folded.
- **Incremental refresh:** Incremental data refresh (Power BI only) will be efficient, in terms of resource utilization and refresh duration. In fact, the Power BI **Incremental Refresh** configuration window will notify you of a warning should it determine that query folding for the table can't be achieved. If it can't be achieved, the goal of incremental refresh is defeated. The mashup engine would then be required to retrieve all source rows, and then apply filters to determine incremental changes.

Query folding may occur for an entire Power Query query, or for a subset of its steps. When query folding cannot be achieved—either partially or fully—the Power Query mashup engine must compensate by processing data transformations itself. This process can involve retrieving source query results, which for large datasets is very resource intensive and slow.

We recommend that you strive to achieve efficiency in your model designs by ensuring query folding occurs whenever possible.

Sources that support folding

Most data sources that have the concept of a query language support query folding. These data sources can include relational databases, OData feeds (including SharePoint lists), Exchange, and Active Directory. However, data sources like flat files, blobs, and web typically do not.

Transformations that can achieve folding

Relational data source transformations that can be query folded are those that can be written as a single SELECT statement. A SELECT statement can be constructed with appropriate WHERE, GROUP BY, and JOIN clauses. It can also contain column expressions (calculations) that use common built-in functions supported by SQL databases.

Generally, the following list describes transformations that can be query folded.

- Removing columns.
- Renaming columns (SELECT column aliases).
- Filtering rows, with static values or Power Query parameters (WHERE clause predicates).
- Grouping and summarizing (GROUP BY clause).

- Expanding record columns (source foreign key columns) to achieve a join of two source tables (JOIN clause).
- Non-fuzzy merging of fold-able queries based on the same source (JOIN clause).
- Appending fold-able queries based on the same source (UNION ALL operator).
- Adding custom columns with *simple logic* (SELECT column expressions). Simple logic implies uncomplicated operations, possibly including the use of M functions that have equivalent functions in the SQL data source, like mathematic or text manipulation functions. For example, the following expressions return the year component of the **OrderDate** column value (to return a numeric value).

```
Date.Year([OrderDate])
```

- Pivoting and unpivoting (PIVOT and UNPIVOT operators).

Transformations that prevent folding

Generally, the following list describes transformations that prevent query folding. This list isn't intended to be an exhaustive list.

- Merging queries based on different sources.
- Appending (union-ing) queries based on different sources.
- Adding custom columns with *complex logic*. Complex logic implies the use of M functions that have no equivalent functions in the data source. For example, the following expressions format the **OrderDate** column value (to return a text value).

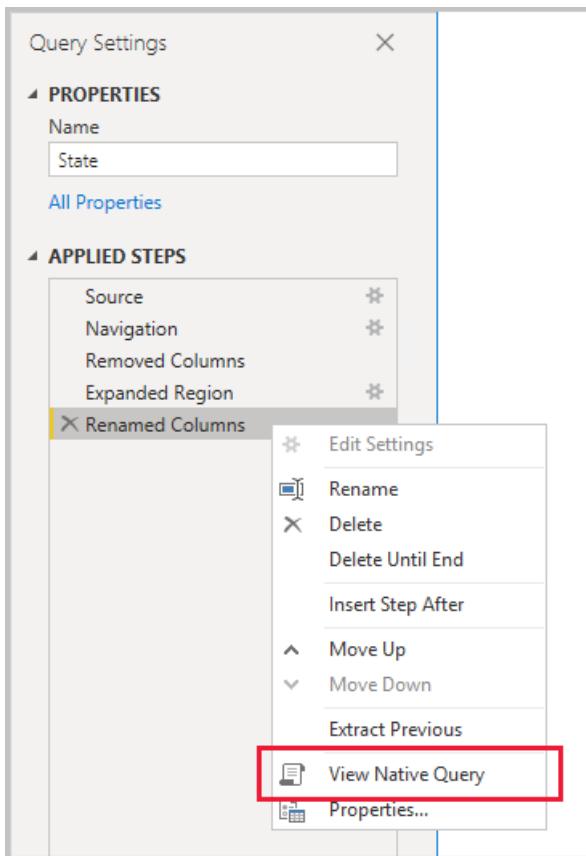
```
Date.ToText([OrderDate], "yyyy")
```

- Adding index columns.
- Changing a column data type.

Note that when a Power Query query encompasses multiple data sources, incompatibility of data source privacy levels can prevent query folding from taking place. For more information, see the [Power BI Desktop privacy levels](#) article.

Determine when a query can be folded

In the Power Query Editor window, it's possible to determine when a Power Query query can be folded. In the **Query Settings** pane, when you right-click the last applied step, if the **View Native Query** option is enabled (not greyed out), then the entire query can be folded.



NOTE

The **View Native Query** option is only available for certain relational DB/SQL generating connectors. It doesn't work for OData based connectors, for example, even though there is folding occurring on the backend. The Query Diagnostics feature is the best way to see what folding has occurred for non-SQL connectors (although the steps that fold aren't explicitly called out—you just see the resulting URL that was generated).

To view the folded query, you select the **View Native Query** option. You're then presented with the native query that Power Query will use to source data.

Native Query

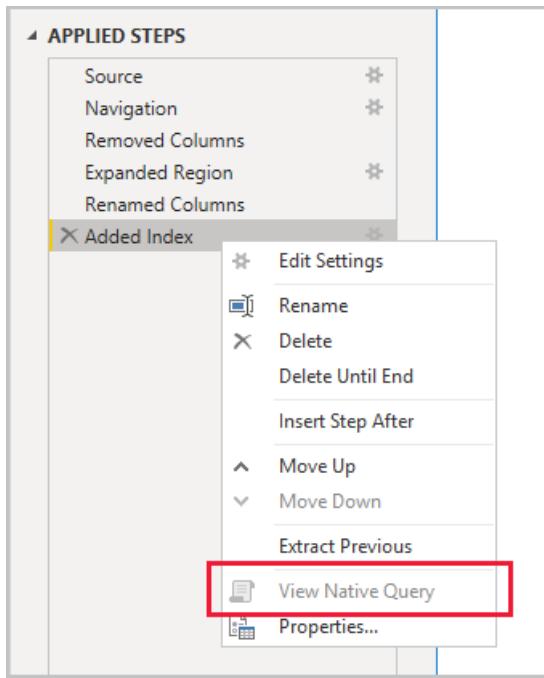
```

select [$Outer].[StateID] as [StateID],
       [$Outer].[StateCode] as [State Code],
       [$Outer].[StateName] as [State],
       [$Outer].[TimeZone] as [Time Zone],
       [$Inner].[RegionName] as [Region]
  from (
    select [...].[StateID] as [StateID],
           [...].[StateCode] as [StateCode],
           [...].[StateName] as [StateName],
           [...].[RegionID] as [RegionID2],
           [...].[TimeZone] as [TimeZone]
      from [dbo].[State] as [...]
  ) as [$Outer]
 left outer join [dbo].[Region] as [$Inner] on ([$Outer].[RegionID2] = [$Inner].[RegionID])

```

OK

If the **View Native Query** option isn't enabled (greyed out), this is evidence that not all query steps can be folded. However, it could mean that a subset of steps can still be folded. Working backwards from the last step, you can check each step to see if the **View Native Query** option is enabled. If so, then you've learned where, in the sequence of steps, that query folding could no longer be achieved.



Next steps

For more information about Query Folding and related articles, check out the following resources:

- [Best practice guidance for query folding](#)
- [Use composite models in Power BI Desktop](#)
- [Incremental refresh in Power BI Premium](#)
- [Using Table.View to Implement Query Folding](#)

How fuzzy matching works in Power Query?

1/15/2022 • 3 minutes to read • [Edit Online](#)

Power Query features such as [fuzzy merge](#), [cluster values](#), and [fuzzy grouping](#) use the same mechanisms to work as fuzzy matching.

This article goes over many scenarios that will show you how to take advantage of the options that fuzzy matching has with the goal of making 'fuzzy' clear.

Adjust the similarity threshold

The best scenario for applying the fuzzy match algorithm is when all text strings in a column contain only the strings that need to be compared and not extra components. For example, comparing `Apples` against `4ppl3s` yields higher similarity scores than comparing `Apples` to

`My favorite fruit, by far, is Apples. I simply love them!`.

This is because the word `Apples` in the second string is only a small part of the whole text string that yields a lower similarity score.

Take a look at the following dataset that consists of responses from a survey that had only one question "What is your favorite fruit?"

FRUIT
Blueberries
Blue berries are simply the best
Strawberries
Strawberries = <3
Apples
'sples
4ppl3s
Bananas
fav fruit is bananas
Banas
My favorite fruit, by far, is Apples. I simply love them!

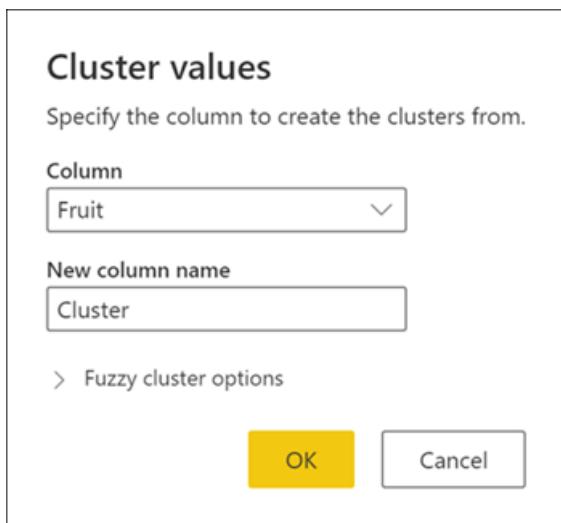
The survey provided one single textbox to input the value and with no validation.

Now you're tasked with clustering the values. To do that, you load the previous table of fruits into Power Query, select the column, and then select the option that reads *Cluster values* inside the **Add column** menu in the ribbon.

The screenshot shows the Power Query ribbon with the 'Add column' tab selected. In the 'View' section, the 'Cluster values' button is highlighted with a red box. Below the ribbon, the 'Queries [1]' pane shows a table named 'Fruits' with 11 rows of fruit-related text. The first row is selected and highlighted in yellow.

	ab Fruit
1	Blueberries
2	Blue berries are simply the best
3	Strawberries
4	Strawberries = <3
5	Apples
6	'sples
7	4ppl3s
8	Bananas
9	fav fruit is bananas
10	Banas
11	My favorite fruit, by far, is Apples. I simply love them!

The **Cluster values** dialog box appears where you can specify the name of the new column. Name this new column *Cluster* and select **Ok**.

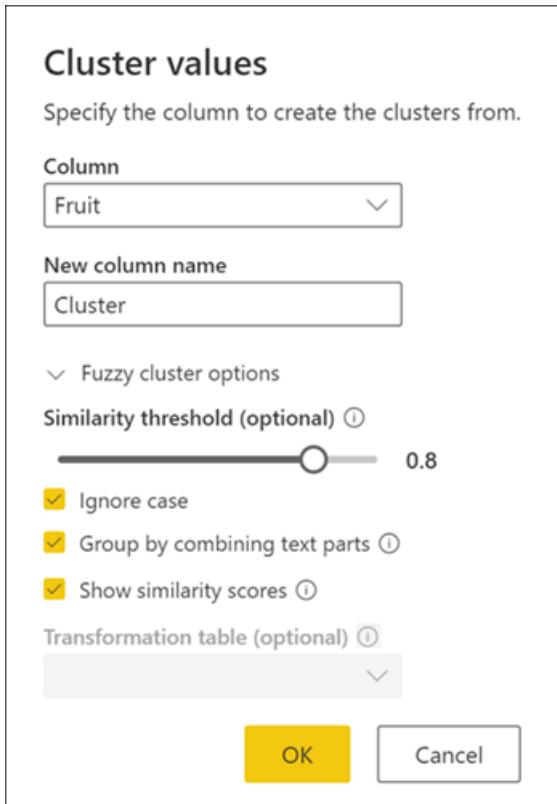


By default, Power Query will use a similarity threshold of 0.8 (or 80%) and the result of the previous operation will yield the following table with a new *Cluster* column:

	ab Fruit	ab Cluster
1	Blueberries	Blueberries
2	Blue berries are simply the best	Blue berries are simply the best
3	Strawberries	Strawberries
4	Strawberries = <3	Strawberries = <3
5	Apples	Apples
6	'sples	Apples
7	4ppl3s	Apples
8	Bananas	Bananas
9	fav fruit is bananas	fav fruit is bananas
10	Banas	Bananas
11	My favorite fruit, by far, is Apples. I simply love them!	My favorite fruit, by far, is Apples. I simply lov...

While the clustering has been done, it is not giving you the expected results for all the rows. Row number two (2) still has the value `Blue berries are simply the best`, but it should be clustered to `Blueberries`, and something similar happens to the text strings `Strawberries = <3`, `fav fruit is bananas`, and `My favorite fruit, by far, is Apples. I simply love them!`.

You wish to determine what's causing this clustering. To do this, you can double-click the *Clustered values* step to bring back the **Cluster values** window. Inside this window, expand the text that reads *Fuzzy cluster options* and enable the option that reads *Show similarity scores* as shown in the image below and hit the OK button:

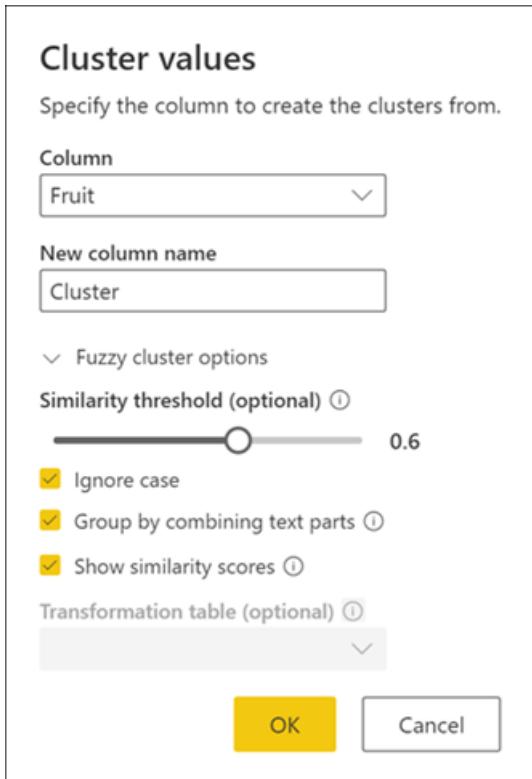


Enabling the *Show similarity scores* option will bring a new column to your table that shows you exactly the similarity score between the defined cluster and the original value.

	ab Fruit	ab Cluster	1.2 Fruit_Cluster_Similarity
1	Blueberries	Blueberries	1
2	Blue berries are simply the best	Blue berries are simply the best	1
3	Strawberries	Strawberries	1
4	Strawberries = <3	Strawberries = <3	1
5	Apples	Apples	1
6	'sples	Apples	0.91
7	4appl3s	Apples	0.91
8	Bananas	Bananas	1
9	fav fruit is bananas	fav fruit is bananas	1
10	Banas	Bananas	0.9
11	My favorite fruit, by far, is Apples. I simply love them!	My favorite fruit, by far, is Apples. I simply lov...	1

Upon closer inspection, you can see that Power Query couldn't find any other values within the similarity threshold for the text strings `Blue berries are simply the best`, `Strawberries = <3`, `fav fruit is bananas`, and `My favorite fruit, by far, is Apples. I simply love them!`.

You can go back to the **Cluster values** dialog box one more time by double-clicking the *Clustered values* step and changing the *Similarity threshold* from 0.8 to 0.6 as shown in the image below:



This change gets you closer to the result that you're looking for, except for the text string `My favorite fruit, by far, is Apples. I simply love them!`. This is because by changing the *Similarity threshold* value from 0.8 to 0.6 Power Query is now able to use the values with a similarity score that start from 0.6 all the way to 1.

	ab Fruit	ab Cluster	1.2 Fruit_Cluster_Similarity
1	Blueberries	Blueberries	1
2	Blue berries are simply the best	Blueberries	0.64
3	Strawberries	Strawberries	1
4	Strawberries = <3	Strawberries	0.75
5	Apples	Apples	1
6	'sples	Apples	0.91
7	4appl3s	Apples	0.91
8	Bananas	Bananas	1
9	fav fruit is bananas	Bananas	0.62
10	Banas	Bananas	0.9
11	My favorite fruit, by far, is Apples. I simply love them!	My favorite fruit, by far, is Apples. I simply love them!	1

NOTE

Power Query always uses the value closest to the threshold to define the clusters. The threshold defines the lower limit of similarity score that is acceptable to create assign the value to a cluster.

You can try again by changing the *Similarity score* from 0.6 to a lower number until you get the results that you're expecting. For this case, **change the *Similarity score* to 0.5**, which will yield the exact result that you're expecting with the text string `My favorite fruit, by far, is Apples. I simply love them!` now assigned to the cluster `Apples` as shown in the next image:

	ab Fruit	ab Cluster	1.2 Fruit_Cluster_Similarity
1	Blueberries	Blueberries	1
2	Blue berries are simply the best	Blueberries	0.64
3	Strawberries	Strawberries	1
4	Strawberries = <3	Strawberries	0.75
5	Apples	Apples	1
6	'sples	Apples	0.91
7	4appl3s	Apples	0.91
8	Bananas	Bananas	1
9	fav fruit is bananas	Bananas	0.62
10	Banas	Bananas	0.9
11	My favorite fruit, by far, is Apples. I simply love them!	Apples	0.54

NOTE

Currently only the [Cluster values](#) feature in Power Query Online will provide a new column with the similarity score.

Behind the scenes of the Data Privacy Firewall

1/15/2022 • 13 minutes to read • [Edit Online](#)

If you've used Power Query for any length of time, you've likely experienced it. There you are, querying away, when you suddenly get an error that no amount of online searching, query tweaking, or keyboard bashing can remedy. An error like:

```
Formula.Firewall: Query 'Query1' (step 'Source') references other queries or steps, so it may not directly access a data source. Please rebuild this data combination.
```

Or maybe:

```
Formula.Firewall: Query 'Query1' (step 'Source') is accessing data sources that have privacy levels which cannot be used together. Please rebuild this data combination.
```

These `Formula.Firewall` errors are the result of Power Query's Data Privacy Firewall (aka the Firewall), which at times may seem like it exists solely to frustrate data analysts the world over. Believe it or not, however, the Firewall serves an important purpose. In this article, we'll delve under the hood to better understand how it works. Armed with greater understanding, you'll hopefully be able to better diagnose and fix Firewall errors in the future.

What is it?

The purpose of the Data Privacy Firewall is simple: it exists to prevent Power Query from unintentionally leaking data between sources.

Why is this needed? I mean, you could certainly author some M that would pass a SQL value to an OData feed. But this would be intentional data leakage. The mashup author would (or at least should) know they were doing this. Why then the need for protection against unintentional data leakage?

The answer? Folding.

Folding?

Folding is a term that refers to converting expressions in M (such as filters, renames, joins, and so on) into operations against a raw data source (such as SQL, OData, and so on). A huge part of Power Query's power comes from the fact that PQ can convert the operations a user performs via its user interface into complex SQL or other backend data source languages, without the user having to know said languages. Users get the performance benefit of native data source operations, with the ease of use of a UI where all data sources can be transformed using a common set of commands.

As part of folding, PQ sometimes may determine that the most efficient way to execute a given mashup is to take data from one source and pass it to another. For example, if you're joining a small CSV file to a huge SQL table, you probably don't want PQ to read the CSV file, read the entire SQL table, and then join them together on your local computer. You probably want PQ to inline the CSV data into a SQL statement and ask the SQL database to perform the join.

This is how unintentional data leakage can happen.

Imagine if you were joining SQL data that included employee Social Security Numbers with the results of an external OData feed, and you suddenly discovered that the Social Security Numbers from SQL were being sent to the OData service. Bad news, right?

This is the kind of scenario the Firewall is intended to prevent.

How does it work?

The Firewall exists to prevent data from one source from being unintentionally sent to another source. Simple enough.

So how does it accomplish this mission?

It does this by dividing your M queries into something called partitions, and then enforcing the following rule:

- A partition may either access compatible data sources, or reference other partitions, but not both.

Simple...yet confusing. What's a partition? What makes two data sources "compatible"? And why should the Firewall care if a partition wants to access a data source and reference a partition?

Let's break this down and look at the above rule one piece at a time.

What's a partition?

At its most basic level, a partition is just a collection of one or more query steps. The most granular partition possible (at least in the current implementation) is a single step. The largest partitions can sometimes encompass multiple queries. (More on this later.)

If you're not familiar with steps, you can view them on the right of the Power Query Editor window after selecting a query, in the **Applied Steps** pane. Steps keep track of everything you've done to transform your data into its final shape.

Partitions that reference other partitions

When a query is evaluated with the Firewall on, the Firewall divides the query and all its dependencies into partitions (that is, groups of steps). Any time one partition references something in another partition, the Firewall replaces the reference with a call to a special function called `Value.Firewall`. In other words, the Firewall doesn't allow partitions to access each other randomly. All references are modified to go through the Firewall. Think of the Firewall as a gatekeeper. A partition that references another partition must get the Firewall's permission to do so, and the Firewall controls whether or not the referenced data will be allowed into the partition.

This all may seem pretty abstract, so let's look at an example.

Assume you have a query called `Employees`, which pulls some data from a SQL database. Assume you also have another query (`EmployeesReference`), which simply references `Employees`.

```
shared Employees = let
    Source = Sql.Database(...),
    EmployeesTable = ...
in
    EmployeesTable;

shared EmployeesReference = let
    Source = Employees
in
    Source;
```

These queries will end up divided into two partitions: one for the `Employees` query, and one for the `EmployeesReference` query (which will reference the `Employees` partition). When evaluated with the Firewall on, these queries will be rewritten like so:

```

shared Employees = let
    Source = Sql.Database(...),
    EmployeesTable = ...
in
    EmployeesTable;

shared EmployeesReference = let
    Source = Value.Firewall("Section1/Employees")
in
    Source;

```

Notice that the simple reference to the Employees query has been replaced by a call to `Value.Firewall`, which is provided the full name of the Employees query.

When EmployeesReference is evaluated, the call to `Value.Firewall("Section1/Employees")` is intercepted by the Firewall, which now has a chance to control whether (and how) the requested data flows into the EmployeesReference partition. It can do any number of things: deny the request, buffer the requested data (which prevents any further folding to its original data source from occurring), and so on.

This is how the Firewall maintains control over the data flowing between partitions.

Partitions that directly access data sources

Let's say you define a query Query1 with one step (note that this single-step query will correspond to one Firewall partition), and that this single step accesses two data sources: a SQL database table and a CSV file. How does the Firewall deal with this, since there's no partition reference, and thus no call to `Value.Firewall` for it to intercept? Let's review to the rule stated earlier:

- A partition may either access compatible data sources, or reference other partitions, but not both.

In order for your single-partition-but-two-data-sources query to be allowed to run, its two data sources must be "compatible". In other words, it needs to be okay for data to be shared between them. In terms of the Power Query UI, this means the privacy levels of the SQL and CSV data sources need to both be Public, or both be Organizational. If they are both marked Private, or one is marked Public and one is marked Organizational, or they are marked using some other combination of privacy levels, then it's not safe for them to both be evaluated in the same partition. Doing so would mean unsafe data leakage could occur (due to folding), and the Firewall would have no way to prevent it.

What happens if you try to access incompatible data sources in the same partition?

`Formula.Firewall: Query 'Query1' (step 'Source') is accessing data sources that have privacy levels which cannot be used together. Please rebuild this data combination.`

Hopefully you now better understand one of the error messages listed at the beginning of this article.

Note that this *compatibility* requirement only applies within a given partition. If a partition is referencing other partitions, the data sources from the referenced partitions don't have to be compatible with one another. This is because the Firewall can buffer the data, which will prevent any further folding against the original data source. The data will be loaded into memory and treated as if it came from nowhere.

Why not do both?

Let's say you define a query with one step (which will again correspond to one partition) that accesses two other queries (that is, two other partitions). What if you wanted, in the same step, to also directly access a SQL database? Why can't a partition reference other partitions and directly access compatible data sources?

As you saw earlier, when one partition references another partition, the Firewall acts as the gatekeeper for all the data flowing into the partition. To do so, it must be able to control what data is allowed in. If there are data sources being accessed within the partition, as well as data flowing in from other partitions, it loses its ability to be the gatekeeper, since the data flowing in could be leaked to one of the internally accessed data sources

without it knowing about it. Thus the Firewall prevents a partition that accesses other partitions from being allowed to directly access any data sources.

So what happens if a partition tries to reference other partitions and also directly access data sources?

Formula.Firewall: Query 'Query1' (step 'Source') references other queries or steps, so it may not directly access a data source. Please rebuild this data combination.

Now you hopefully better understand the other error message listed at the beginning of this article.

Partitions in-depth

As you can probably guess from the above information, how queries are partitioned ends up being incredibly important. If you have some steps that are referencing other queries, and other steps that access data sources, you now hopefully recognize that drawing the partition boundaries in certain places will cause Firewall errors, while drawing them in other places will allow your query to run just fine.

So how exactly do queries get partitioned?

This section is probably the most important for understanding why you're seeing Firewall errors, as well as understanding how to resolve them (where possible).

Here's a high-level summary of the partitioning logic.

- Initial Partitioning
 - Creates a partition for each step in each query
- Static Phase
 - This phase doesn't depend on evaluation results. Instead, it relies on how the queries are structured.
 - Parameter Trimming
 - Trims parameter-esque partitions, that is, any one that:
 - Doesn't reference any other partitions
 - Doesn't contain any function invocations
 - Isn't cyclic (that is, it doesn't refer to itself)
 - Note that "removing" a partition effectively includes it in whatever other partitions reference it.
 - Trimming parameter partitions allows parameter references used within data source function calls (for example, `Web.Contents(myUrl)`) to work, instead of throwing "partition can't reference data sources and other steps" errors.
 - Grouping (Static)
 - Partitions are merged, while maintaining separation between:
 - Partitions in different queries
 - Partitions that reference other partitions vs. those that don't
- Dynamic Phase
 - This phase depends on evaluation results, including information about data sources accessed by various partitions.
 - Trimming
 - Trims partitions that meet all the following requirements:
 - Doesn't access any data sources
 - Doesn't reference any partitions that access data sources
 - Isn't cyclic
 - Grouping (Dynamic)
 - Now that unnecessary partitions have been trimmed, try to create Source partitions that are as large as possible.
 - Merge all partitions with their input partitions if each of its inputs:
 - Is part of the same query

- o Doesn't reference any other partitions
- o Is only referenced by the current partition
- o Isn't the result (that is, final step) of a query
- o Isn't cyclic

What does all this mean?

Let's walk through an example to illustrate how the complex logic laid out above works.

Here's a sample scenario. It's a fairly straightforward merge of a text file (Contacts) with a SQL database (Employees), where the SQL server is a parameter (DbServer).

The three queries

Here's the M code for the three queries used in this example.

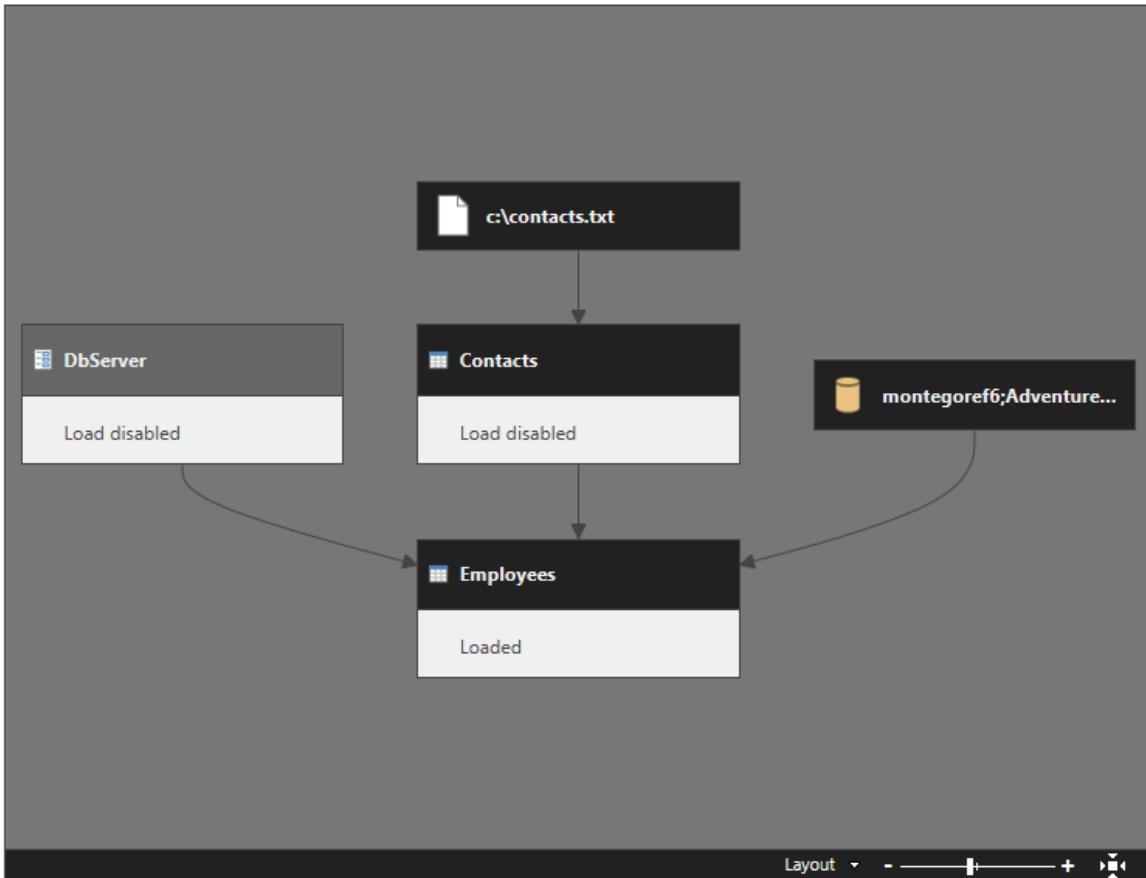
```
shared DbServer = "montegoref6" meta [IsParameterQuery=true, Type="Text", IsParameterQueryRequired=true];
```

```
shared Contacts = let
    Source = Csv.Document(File.Contents("C:\contacts.txt"),[Delimiter=" ", Columns=15, Encoding=1252, QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"ContactID", Int64.Type}, {"NameStyle", type logical}, {"Title", type text}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"Suffix", type text}, {"EmailAddress", type text}, {"EmailPromotion", Int64.Type}, {"Phone", type text}, {"PasswordHash", type text}, {"PasswordSalt", type text}, {"AdditionalContactInfo", type text}, {"rowguid", type text}, {"ModifiedDate", type datetime}}),
    in
    #"Changed Type";
```

```
shared Employees = let
    Source = Sql.Databases(DbServer),
    AdventureWorks = Source{[Name="AdventureWorks"]}[Data],
    HumanResources_Employee = AdventureWorks{[Schema="HumanResources", Item="Employee"]}[Data],
    #"Removed Columns" = Table.RemoveColumns(HumanResources_Employee,{"HumanResources.Employee(EmployeeID)", "HumanResources.Employee(ManagerID)", "HumanResources.EmployeeAddress", "HumanResources.EmployeeDepartmentHistory", "HumanResources.EmployeePayHistory", "HumanResources.JobCandidate", "Person.Contact", "Purchasing.PurchaseOrderHeader", "Sales.SalesPerson"}),
    #"Merged Queries" = Table.NestedJoin(#"Removed Columns", {"ContactID"}, Contacts, {"ContactID"}, "Contacts", JoinKind.LeftOuter),
    #"Expanded Contacts" = Table.ExpandTableColumn(#"Merged Queries", "Contacts", {"EmailAddress"}, {"EmailAddress"})
    in
    #"Expanded Contacts";
```

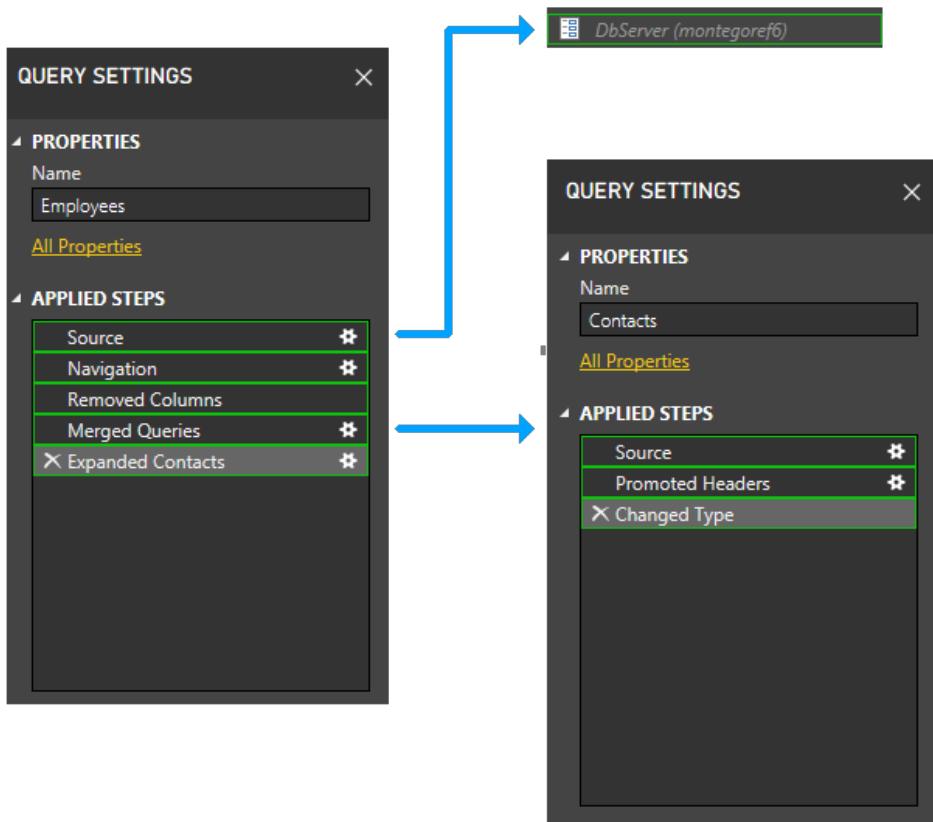
Here's a higher-level view, showing the dependencies.

Query Dependencies

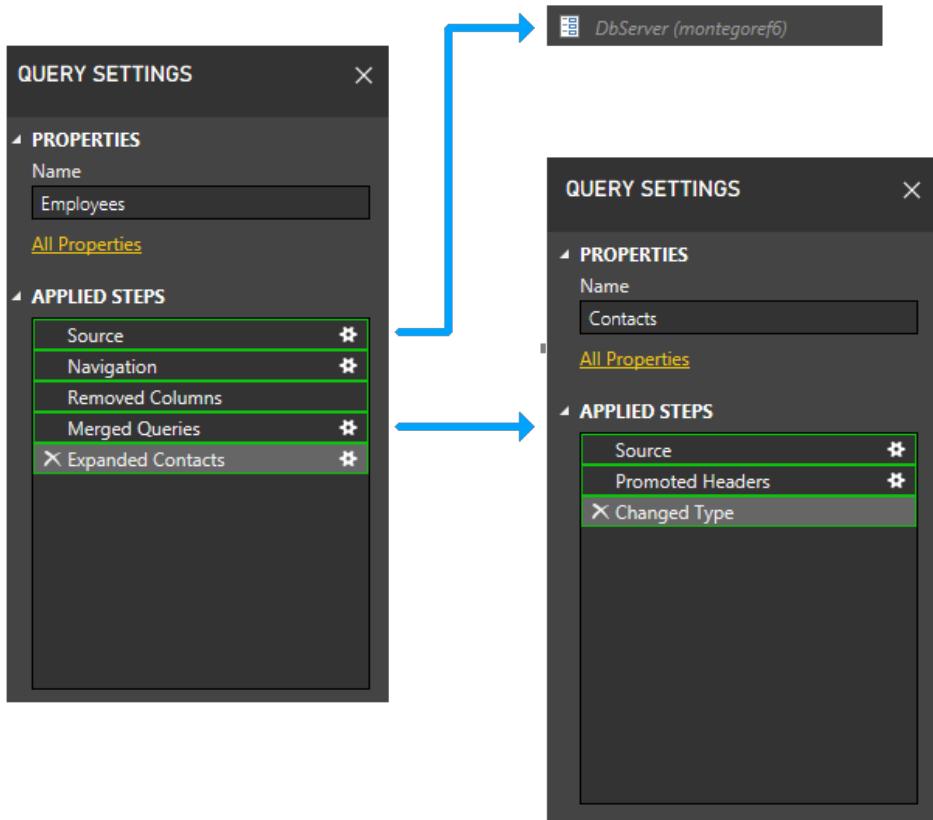


Let's partition

Let's zoom in a bit and include steps in the picture, and start walking through the partitioning logic. Here's a diagram of the three queries, showing the initial firewall partitions in green. Notice that each step starts in its own partition.

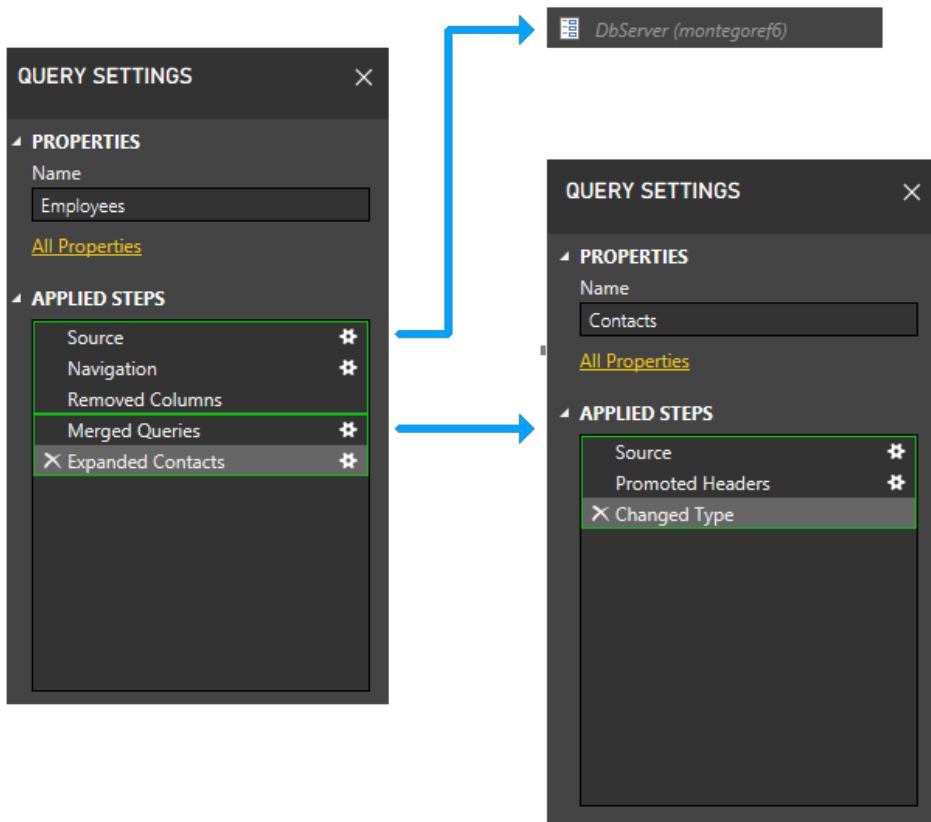


Next, we trim parameter partitions. Thus, DbServer gets implicitly included in the Source partition.



Now we perform the static grouping. This maintains separation between partitions in separate queries (note for instance that the last two steps of Employees don't get grouped with the steps of Contacts), as well as between

partitions that reference other partitions (such as the last two steps of Employees) and those that don't (such as the first three steps of Employees).

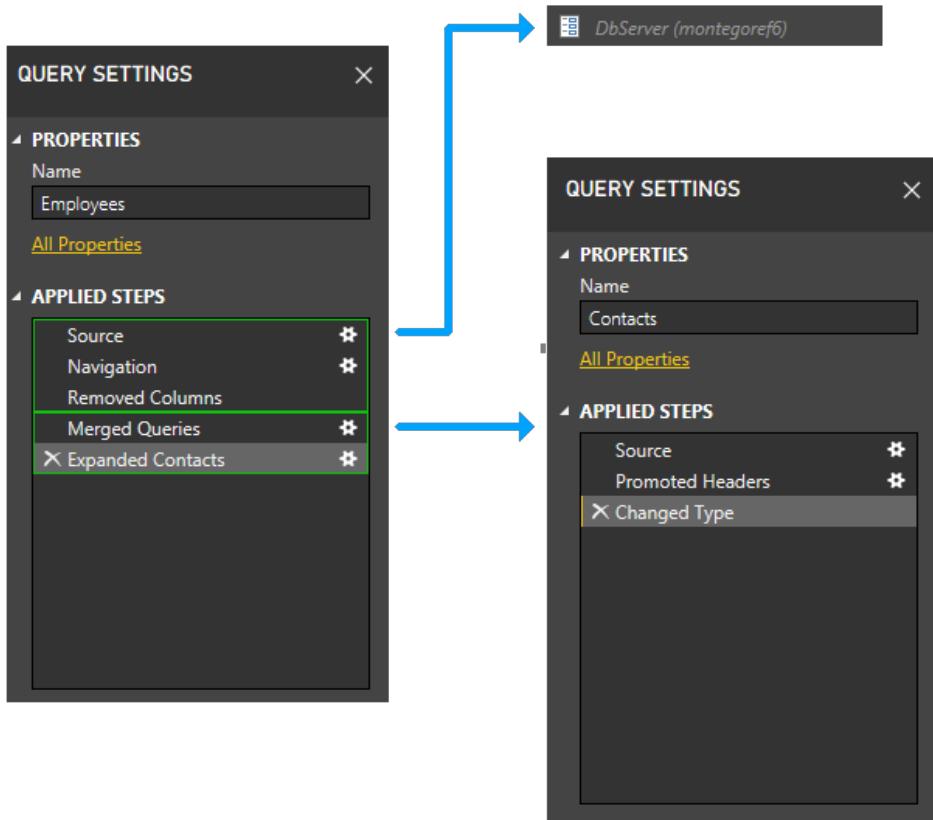


Now we enter the dynamic phase. In this phase, the above static partitions are evaluated. Partitions that don't access any data sources are trimmed. Partitions are then grouped to create source partitions that are as large as possible. However, in this sample scenario, all the remaining partitions access data sources, and there isn't any further grouping that can be done. The partitions in our sample thus won't change during this phase.

Let's pretend

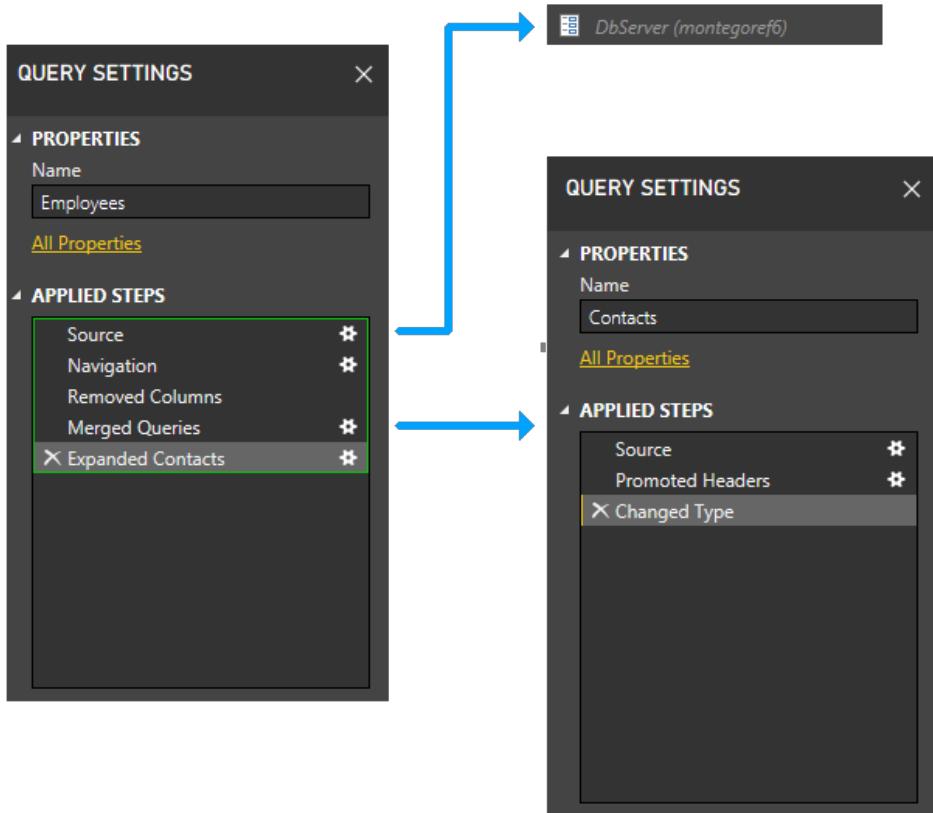
For the sake of illustration, though, let's look at what would happen if the Contacts query, instead of coming from a text file, were hard-coded in M (perhaps via the **Enter Data** dialog).

In this case, the Contacts query would not access any data sources. Thus, it would get trimmed during the first part of the dynamic phase.



With the Contacts partition removed, the last two steps of Employees would no longer reference any partitions except the one containing the first three steps of Employees. Thus, the two partitions would be grouped.

The resulting partition would look like this.



Example: Passing data from one data source to another

Okay, enough abstract explanation. Let's look at a common scenario where you're likely to encounter a Firewall error and the steps to resolve it.

Imagine you want to look up a company name from the Northwind OData service, and then use the company name to perform a Bing search.

First, you create a **Company** query to retrieve the company name.

```
let
  Source = OData.Feed("https://services.odata.org/V4/Northwind/Northwind.svc/", null,
  [Implementation="2.0"]),
  Customers_table = Source{[Name="Customers",Signature="table"]}[Data],
  CHOPS = Customers_table{[CustomerID="CHOPS"]}[CompanyName]
in
  CHOPS
```

Next, you create a **Search** query that references **Company** and passes it to Bing.

```
let
  Source = Text.FromBinary(Web.Contents("https://www.bing.com/search?q=" & Company))
in
  Source
```

At this point you run into trouble. Evaluating **Search** produces a Firewall error.

Formula.Firewall: Query 'Search' (step 'Source') references other queries or steps, so it may not directly access a data source. Please rebuild this data combination.

This is because the **Source** step of **Search** is referencing a data source ([bing.com](https://www.bing.com)) and also referencing another query/partition (**Company**). It is violating the rule mentioned above ("a partition may either access compatible data sources, or reference other partitions, but not both").

What to do? One option is to disable the Firewall altogether (via the Privacy option labeled **Ignore the Privacy Levels and potentially improve performance**). But what if you want to leave the Firewall enabled?

To resolve the error without disabling the Firewall, you can combine **Company** and **Search** into a single query, like this:

```
let
  Source = OData.Feed("https://services.odata.org/V4/Northwind/Northwind.svc/", null,
  [Implementation="2.0"]),
  Customers_table = Source{[Name="Customers",Signature="table"]}[Data],
  CHOPS = Customers_table{[CustomerID="CHOPS"]}[CompanyName],
  Search = Text.FromBinary(Web.Contents("https://www.bing.com/search?q=" & CHOPS))
in
  Search
```

Everything is now happening inside a *single* partition. Assuming that the privacy levels for the two data sources are compatible, the Firewall should now be happy, and you'll no longer get an error.

That's a wrap

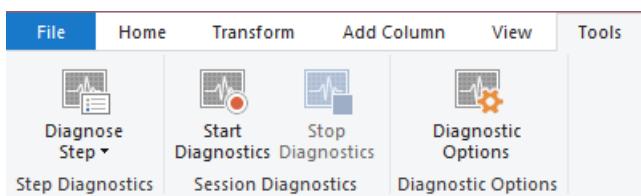
While there's much more that could be said on this topic, this introductory article is already long enough. Hopefully it's given you a better understanding of the Firewall, and will help you to understand and fix Firewall errors when you encounter them in the future.

Query Diagnostics

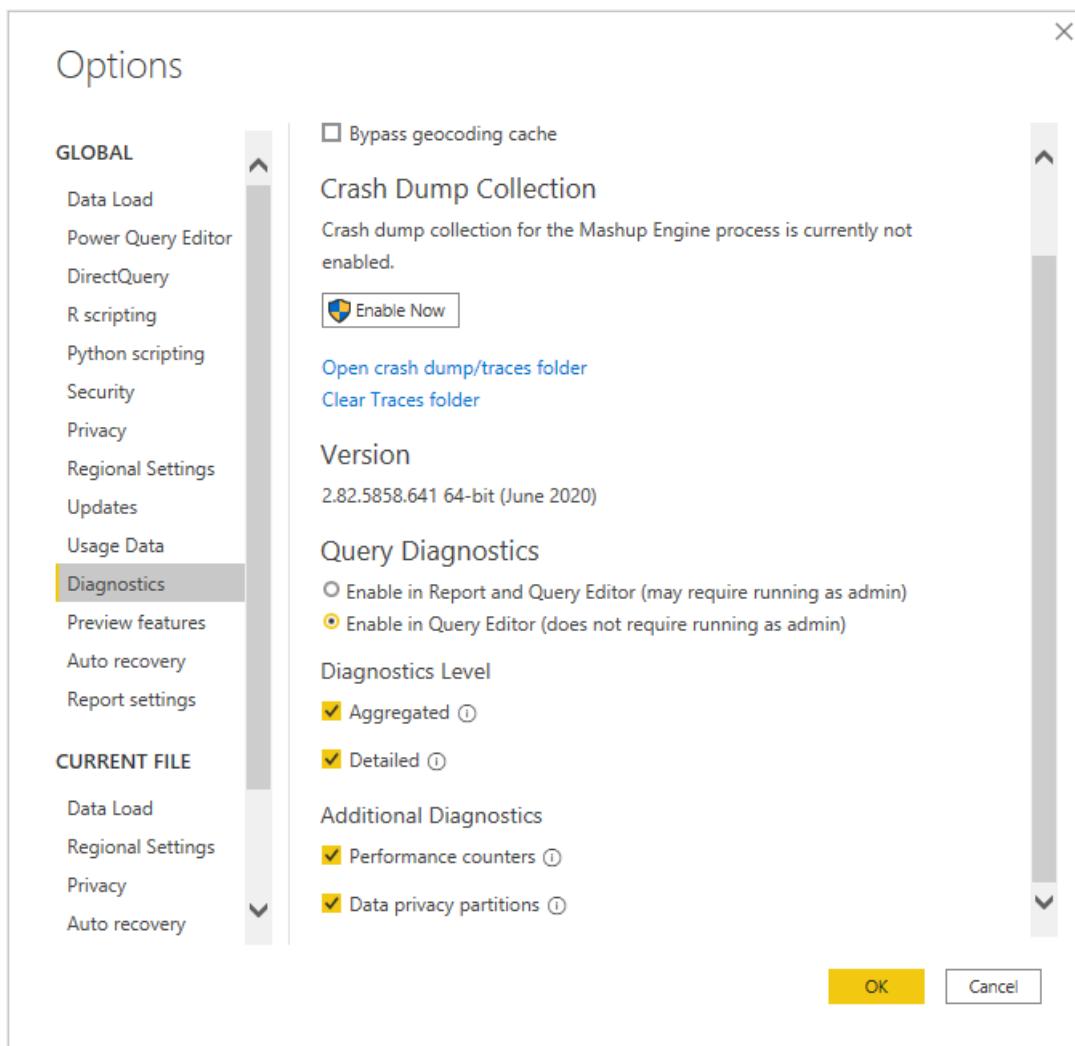
1/15/2022 • 11 minutes to read • [Edit Online](#)

With Query Diagnostics, you can achieve a better understanding of what Power Query is doing at authoring and at refresh time in Power BI Desktop. While we'll be expanding on this feature in the future, including adding the ability to use it during full refreshes, at this time you can use it to understand what sort of queries you're emitting, what slowdowns you might run into during authoring refresh, and what kind of background events are happening.

To use Query Diagnostics, go to the **Tools** tab in the Power Query Editor ribbon.



By default, Query Diagnostics might require administrative rights to run (depending on IT policy). If you find yourself unable to run Query Diagnostics, open the Power BI options page, and in the **Diagnostics** tab, select **Enable in Query Editor (does not require running as admin)**. This selection constrains you from being able to trace diagnostics when doing a full refresh into Power BI rather than the Power Query editor, but does allow you to still use it when previewing, authoring, and so on.



Whenever you start diagnostics, Power Query begins tracing any evaluations that you cause. The evaluation that most users think of is when you press refresh, or when you retrieve data for the first time, but there are many actions that can cause evaluations, depending on the connector. For example, with the SQL connector, when you retrieve a list of values to filter, that would kick off an evaluation as well—but it doesn't associate with a user query, and that's represented in the diagnostics. Other system-generated queries might include the navigator or the get data experience.

When you press **Diagnose Step**, Power Query runs a special evaluation of just the step you're looking at. It then shows you the diagnostics for that step, without showing you the diagnostics for other steps in the query. This can make it much easier to get a narrow view into a problem.

It's important that if you're recording all traces from **Start Diagnostics** that you press **Stop diagnostics**. Stopping the diagnostics allows the engine to collect the recorded traces and parse them into the proper output. Without this step, you'll lose your traces.

Types of Diagnostics

We currently provide three types of diagnostics, one of which has two levels of detail.

The first of these diagnostics are the primary diagnostics, which have a detailed view and a summarized view. The summarized view is aimed to give you an immediate insight into where time is being spent in your query. The detailed view is much deeper, line by line, and is, in general, only needed for serious diagnosing by power users.

For this view, some capabilities, like the Data Source Query column, are currently available only on certain connectors. We'll be working to extend the breadth of this coverage in the future.

Data privacy partitions provide you with a better understanding of the logical partitions used for data privacy.

NOTE

Power Query might perform evaluations that you may not have directly triggered. Some of these evaluations are performed in order to retrieve metadata so we can best optimize our queries or to provide a better user experience (such as retrieving the list of distinct values within a column that are displayed in the Filter Rows experience). Others might be related to how a connector handles parallel evaluations. At the same time, if you see in your query diagnostics repeated queries that you don't believe make sense, feel free to reach out through normal support channels—your feedback is how we improve our product.

Summarized vs. Detailed View

Query diagnostics provides two views: summarized and detailed. The summarized view "collapses" multiple related operations into a single operation. In this process, details collected by each operation are combined, and the exclusive durations are summed. No information is lost as part of this process.

The summarized view provides an overview of what occurred during an evaluation for easy high-level review. If further breakdown is wanted for a specific operation, the user can look at the group ID and view the corresponding operations that were grouped in the detail view.

Explaining Multiple Evaluations

When refreshing in the Power Query editor, there's a lot done behind the scenes to attempt to give you a fluent user experience. As an example, when you **Refresh Preview**, the evaluator will execute the final step of each given Query. But then in the background it sequentially runs n-1 steps, n-2, steps, and so on, so that if you step back through your steps, it's already available.

To provide higher performance, currently some caching happens so that it doesn't have to rerun every part of

the final query plan as it goes back through the steps. While this caching is useful for normal authoring, it means that you won't always get correct step comparison information because of later evaluations pulling on cached data.

Diagnostics Schema

Id

When analyzing the results of a recording, it's important to filter the recording session by Id, so that columns such as Exclusive Duration % make sense.

Id is a composite identifier. It's formed of two numbers—one before the dot, and one after. The first number is the same for all evaluations that resulted from a single user action. In other words, if you press refresh twice, there will be two different numbers leading the dot, one for each user activity taken. This numbering is sequential for a given diagnostics recording.

The second number represents an evaluation by the engine. This number is sequential for the lifetime of the process where the evaluation is queued. If you run multiple diagnostics recording sessions, you'll see this number continue to grow across the different sessions.

To summarize, if you start recording, press evaluation once, and stop recording, you'll have some number of Ids in your diagnostics. But since you only took one action, they'll all be 1.1, 1.2, 1.3, and so on.

The combination of the activityId and the evaluationId, separated by the dot, provides a unique identifier for an evaluation of a single recording session.

Query

The name of the Query in the left-hand pane of the Power Query editor.

Step

The name of the Step in the right-hand pane of the Power Query editor. Things like filter dropdowns generally associate with the step you're filtering on, even if you're not refreshing the step.

Category

The category of the operation.

Data Source Kind

This tells you what sort of data source you're accessing, such as SQL or Oracle.

Operation

The actual operation being performed. This operation can include evaluator work, opening connections, sending queries to the data source, and many more.

Start Time

The time that the operation started.

End Time

The time that the operation ended.

Exclusive Duration (%)

The Exclusive Duration column of an event is the amount of time the event was active. This contrasts with the "duration" value that results from subtracting the values in an event's Start Time column and End Time column. This "duration" value represents the total time that elapsed between when an event began and when it ended, which may include times the event was in a suspended or inactive state and another event was consuming resources.

Exclusive duration % adds up to approximately 100% within a given evaluation, as represented by the Id column.

For example, if you filter on rows with Id 1.x, the Exclusive Duration percentages would sum to approximately 100%. This isn't the case if you sum the Exclusive Duration % values of all rows in a given diagnostic table.

Exclusive Duration

The absolute time, rather than %, of exclusive duration. The total duration (that is, exclusive duration + time when the event was inactive) of an evaluation can be calculated in one of two ways:

- Find the operation called "Evaluation". The difference between End Time–Start Time results in the total duration of an event.
- Subtract the minimum start time of all operations in an event from the maximum end time. Note that in cases when the information collected for an event doesn't account for the total duration, an operation called "Trace Gaps" is generated to account for this time gap.

Resource

The resource you're accessing for data. The exact format of this resource depends on the data source.

Data Source Query

Power Query does something called *Folding*, which is the act of running as many parts of the query against the back-end data source as possible. In Direct Query mode (over Power Query), where enabled, only transforms that fold will run. In import mode, transforms that can't fold will instead be run locally.

The Data Source Query column allows you to see the query or HTTP request/response sent against the back-end data source. As you author your Query in the editor, many Data Source Queries will be emitted. Some of these are the actual final Data Source Query to render the preview, but others may be for data profiling, filter dropdowns, information on joins, retrieving metadata for schemas, and any number of other small queries.

In general, you shouldn't be concerned by the number of Data Source Queries emitted unless there are specific reasons to be concerned. Instead, you should focus on making sure the proper content is being retrieved. This column might also help determine if the Power Query evaluation was fully folded.

Additional Info

There's a lot of information retrieved by our connectors. Much of it is ragged and doesn't fit well into a standard column hierarchy. This information is put in a record in the additional info column. Information logged from custom connectors also appears here.

Row Count

The number of rows returned by a Data Source Query. Not enabled on all connectors.

Content Length

Content length returned by HTTP Requests, as commonly defined. This isn't enabled in all connectors, and it won't be accurate for connectors that retrieve requests in chunks.

Is User Query

A Boolean value that indicates if it's a query authored by the user and present in the left-hand pane, or if it was generated by some other user action. Other user actions can include things such as filter selection or using the navigator in the get data experience.

Path

Path represents the relative route of the operation when viewed as part of an interval tree for all operations within a single evaluation. At the top (root) of the tree, there's a single operation called *Evaluation* with path "0". The start time of this evaluation corresponds to the start of this evaluation as a whole. The end time of this evaluation shows when the whole evaluation finished. This top-level operation has an exclusive duration of 0, as its only purpose is to serve as the root of the tree.

Further operations branch from the root. For example, an operation might have "0/1/5" as a path. This path

would be understood as:

- 0: tree root
- 1: current operation's parent
- 5: index of current operation

Operation "0/1/5" might have a child node, in which case, the path has the form "0/1/5/8", with 8 representing the index of the child.

Group ID

Combining two (or more) operations won't occur if it leads to detail loss. The grouping is designed to approximate "commands" executed during the evaluation. In the detailed view, multiple operations share a Group Id, corresponding to the groups that are aggregated in the Summary view.

As with most columns, the group ID is only relevant within a specific evaluation, as filtered by the Id column.

Data Privacy Partitions Schema

Id

Same as the ID for the other query diagnostics results. The integer part represents a single activity ID, while the fractional part represents a single evaluation.

Partition Key

Corresponds to the Query/Step that's used as a firewall partition.

Firewall Group

Categorization that explains why this partition has to be evaluated separately, including details on the privacy level of the partition.

Accessed Resources

List of resource paths for all the resources accessed by this partition, and is in general uniquely identifying a data source.

Partition Inputs

List of partition keys upon which the current partition depends (this list could be used to build a graph).

Expression

The expression that gets evaluated on top of the partition's query/step. In several cases, it coincides with the query/step.

Start Time

Time when evaluation started for this partition.

End Time

Time when evaluation ended for this partition.

Duration

A value derived from End Time minus Start Time.

Exclusive Duration

If partitions are assumed to execute in a single thread, exclusive duration is the "real" duration that can be attributed to this partition.

Exclusive duration %

Exclusive duration as a percentage.

Diagnostics

This column only appears when the query diagnostics "Aggregated" or "Detailed" is also captured, allowing the user to correspond between the two diagnostics outputs.

Performance Counters Schema

When you run performance counters, every half second Power Query takes a snapshot of resource utilization. This snapshot isn't useful for very fast queries, but can be helpful for queries that use up a lot more resources.

% Processor Time

Percent of time spent by processors on the query. This percentage may reach above 100% because of multiple processors.

Total Processor Time

Total duration of processor time spent on the query.

IO Data Bytes per Second

Throughput speed of data received from the data source, expressed in bytes per second.

Commit (Bytes)

Amount of virtual memory reserved by the evaluation.

Working Set (Bytes)

Amount of memory reserved by the evaluation.

Additional Reading

[How to record diagnostics in various use cases](#)

[More about reading and visualizing your recorded traces](#)

[How to understand what query operations are folding using Query Diagnostics](#)

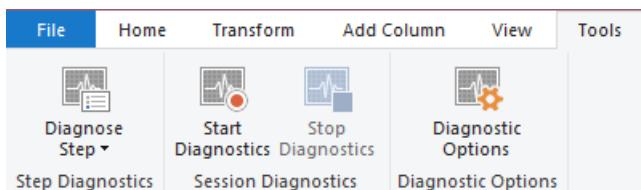
Recording Query Diagnostics in Power BI

1/15/2022 • 6 minutes to read • [Edit Online](#)

When authoring in Power Query, the basic workflow is that you connect to a data source, apply some transformations, potentially refresh your data in the Power Query editor, and then load it to the Power BI model. Once it's in the Power BI model, you may refresh it from time to time in Power BI Desktop (if you're using Desktop to view analytics), aside from any refreshes you do in the service.

While you may get a similar result at the end of an authoring workflow, refreshing in the editor, or refreshing in Power BI proper, very different evaluations are run by the software for the different user experiences provided. It's important to know what to expect when doing query diagnostics in these different workflows so you aren't surprised by the very different diagnostic data.

To start Query Diagnostics, go to the 'Tools' tab in the Power Query Editor ribbon. You're presented here with a few different options.



There are two primary options here, 'Diagnose Step' and 'Start Diagnostics' (paired with 'Stop Diagnostics'). The former will give you information on a query up to a selected step, and is most useful for understanding what operations are being performed locally or remotely in a query. The latter gives you more insight into a variety of other cases, discussed below.

Connector Specifics

It's important to mention that there is no way to cover all the different permutations of what you'll see in Query Diagnostics. There are lots of things that can change exactly what you see in results:

- Connector
- Transforms applied
- System that you're running on
- Network configuration
- Advanced configuration choices
- ODBC configuration

For the most broad coverage this documentation will focus on Query Diagnostics of the Northwind Customers table, both on SQL and OData. The OData notes use the public endpoint found at [the OData.org website](#), while you'll need to provide a SQL server for yourself. Many data sources will differ significantly from these, and will have connector specific documentation added over time.

Start / Stop Diagnostics

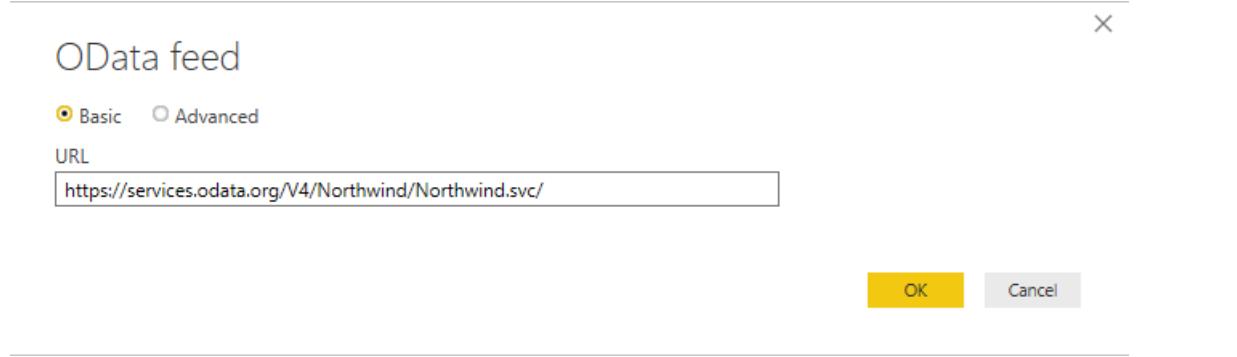
'Start Diagnostics' and 'Stop Diagnostics' are more broadly applicable than 'Diagnose Step', but will also give you a lot more information that you'll need to sort through. For example, starting diagnostics, refreshing a preview, and then stopping will give you equivalent information to running Diagnose Step on every step (due to how Power Query works in the editor to refresh each step independently).

To start recording, click 'Start Diagnostics', perform whatever evaluations you want (authoring, preview refresh, full refresh), and then click 'Stop Diagnostics'.

Authoring

The authoring workflow's primary difference is that it will generally generate more individual evaluations than seen in other workflows. As discussed in the primary Query Diagnostics article, these are a result of populating various user interfaces such as the navigator or filter dropdowns.

We're going to walk through an example. We're using the OData connector in this sample, but when reviewing the output we'll also look at the SQL version of the same database. For both data sources, we're going to connect to the data source via 'New Source', 'Recent Sources', or 'Get Data'. For the SQL connection you'll need to put in credentials for your server, but for the public OData endpoint you can put in the endpoint linked above.



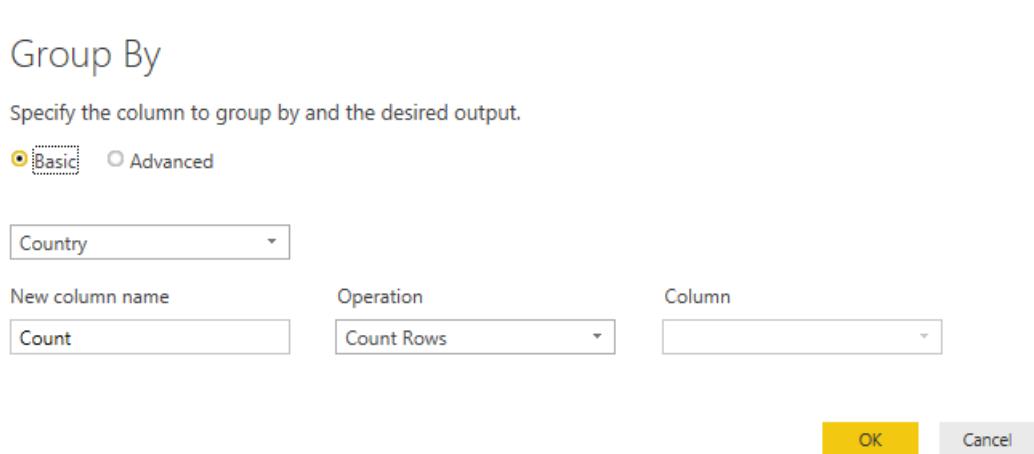
Once you connect and choose authentication, select the 'Customers' table from the OData service.

The screenshot shows the Power Query 'Navigator' interface. On the left, a tree view lists tables: 'Alphabetical_list_of_products', 'Categories', 'Category_Sales_for_1997', 'Current_Product_Lists', 'Customer_and_Suppliers_by_Cities', 'CustomerDemographics', 'Customers' (selected and highlighted with a checkmark), 'Employees', 'Invoices', 'Order_Details', 'Order_Details_Extendeds', 'Order_Subtotals', 'Orders', 'Orders_Qries', 'Product_Sales_for_1997', 'Products', 'Products_Above_Average_Prices', 'Products_by_Categories', and 'Regions'. On the right, a preview of the 'Customers' table is shown with columns: CustomerID, CompanyName, ContactName, and ContactTitle. The preview was downloaded on Sunday, October 6, 2019. At the bottom are 'Select Related Tables', 'OK', and 'Cancel' buttons.

This will present you with the Customers table in the Power Query interface. Let's say that we want to know how many Sales Representatives there are in different countries. First, right click on 'Sales Representative' under the 'Contact Title' column, mouse over 'Text Filters', and select 'Equals'.

A ^B _C ContactTitle	A ^B _C Address	A ^B _C City
Sales Representative	Obere Str. 57	Berlin
Owner	22	México D.F.
Owner		
Sales Representative		
Order Administrator		
Sales Representative		
Marketing Manager	27, rue du Marché-aux-	
Owner	C/ Araquil, 67	
Owner	12, rue des Bouchers	
Accounting Manager	23 Tsawassen Blvd.	
Sales Representative	Fauntleroy Circus	

Now, select 'Group By' from the Ribbon and do a grouping by 'Country', with your aggregate being a 'Count'.



This should present you with the same data you see below.

A ^B _C Country	1.2 Count
1 Germany	4
2 UK	3
3 Italy	1
4 Venezuela	1
5 USA	3
6 France	1
7 Mexico	1
8 Portugal	1
9 Argentina	1
10 Brazil	1

Finally, navigate back to the 'Tools' tab of the Ribbon and click 'Stop Diagnostics'. This will stop the tracing and build your diagnostics file for you, and the summary and detailed tables will appear on the left-hand side.

If you trace an entire authoring session, you will generally expect to see something like a source query evaluation, then evaluations related to the relevant navigator, then at least one query emitted for each step you apply (with potentially more depending on the exact UX actions taken). In some connectors, parallel evaluations will happen for performance reasons that will yield very similar sets of data.

Refresh Preview

When you have finished transforming your data, you have a sequence of steps in a query. When you press 'Refresh Preview' or 'Refresh All' in the Power Query editor, you won't see just one step in your query diagnostics. The reason for this is that refreshing in the Power Query Editor explicitly refreshes the query ending with the last step applied, and then steps back through the applied steps and refreshes for the query up to that point, back to the source.

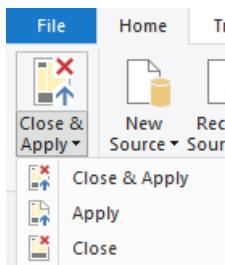
This means that if you have five steps in your query, including Source and Navigator, you will expect to see five different evaluations in your diagnostics. The first one, chronologically, will often (but not always) take the longest. This is due to two different reasons:

- It may potentially cache input data that the queries run after it (representing earlier steps in the User Query) can access faster locally.
- It may have transforms applied to it that significantly truncate how much data has to be returned.

Note that when talking about 'Refresh All' that it will refresh all queries and you'll need to filter to the ones you care about, as you might expect.

Full Refresh

Query Diagnostics can be used to diagnose the so-called 'final query' that is emitted during the Refresh in Power BI, rather than just the Power Query editor experience. To do this, you first need to load the data to the model once. If you are planning to do this, make sure that you realize that if you press 'Close and Apply' that the editor window will close (interrupting tracing) so you either need to do it on the second refresh, or click the dropdown icon under 'Close and Apply' and press 'Apply' instead.



Either way, make sure to press 'Start Diagnostics' on the Diagnostics section of the 'Tools' tab in the editor. Once you've done this refresh your model, or even just the table you care about.

```
New measure
New column
New quick measure
Refresh data
Edit query
Manage aggregations
Rename
Delete
Hide
View hidden
Unhide all
Collapse all
Expand all
```

Once it's done loading the data to model, press 'Stop' diagnostics.

You can expect to see some combination of metadata and data queries. Metadata calls grab the information it can about the data source. Data retrieval is about accessing the data source, emitting the final built up Data Source Query with folded down operations, and then performing whatever evaluations are missing on top, locally.

It's important to note that just because you see a resource (database, web endpoint, etc.) or a data source query in your diagnostics, it doesn't mean that it's necessarily performing network activity. Power Query may retrieve this information from its cache. In future updates, we will indicate whether or not information is being retrieved from the cache for easier diagnosis.

Diagnose Step

'Diagnose Step' is more useful for getting an insight into what evaluations are happening up to a single step, which can help you identify, up to that step, what performance is like as well as what parts of your query are being performed locally or remotely.

If you used 'Diagnose Step' on the query we built above, you'll find that it only returns 10 or so rows, and if we look at the last row with a Data Source Query we can get a pretty good idea of what our final emitted query to the data source will be. In this case, we can see that Sales Representative was filtered remotely, but the grouping (by process of elimination) happened locally.

The screenshot shows the Power Query Editor interface with the 'Diagnose Step' feature enabled. The main area displays a table of query steps, each with a duration, resource, data source query, additional info, row count, and content length. The right side features a 'Query Settings' pane with 'Properties' and 'Applied Steps' sections. The 'Applied Steps' section includes 'Source', 'Converted to Table', 'Expanded Column1', and 'Changed Type'. At the bottom, detailed OData requests and responses are shown for each step, illustrating the remote filtering of 'Sales Representative' and the local grouping by 'CustomerID'.

If you start and stop diagnostics and refresh the same query, we get 40 rows due to the fact that, as mentioned above, Power Query is getting information on every step, not just the final step. This makes it harder when you're just trying to get insight into one particular part of your query.

Additional Reading

[An introduction to the feature](#)

[More about reading and visualizing your recorded traces](#)

[How to understand what query operations are folding using Query Diagnostics](#)

Visualizing and Interpreting Query Diagnostics in Power BI

1/15/2022 • 4 minutes to read • [Edit Online](#)

Introduction

Once you've [recorded](#) the diagnostics you want to use, the next step is being able to understand what they say.

It's helpful to have a good understanding of what exactly each column in the query diagnostics schema means, which we're not going to repeat in this short tutorial. There's a full writeup of that [here](#).

In general, when building visualizations, it's better to use the full detailed table because regardless of how many rows it is, what you're probably looking at is some kind of depiction of how the time spent in different resources adds up, or what the native query emitted was.

As mentioned in our article on recording the diagnostics, I'm working with the OData and SQL traces for the same table (or very nearly so)--the Customers table from Northwind. In particular, I'm going to focus on common ask from our customers, as well as one of the most easy to interpret sets of traces: full refresh of the data model.

Building the visualizations

When going through traces, there are a lot of ways you can evaluate them. In this article we're going to focus on a two visualization split--one to show the details you care about, and the other to easily look at time contributions of various factors. For the first visualization, a table is used. You can pick any fields you like, but the ones recommended for an easy, high level look at what's going on are:

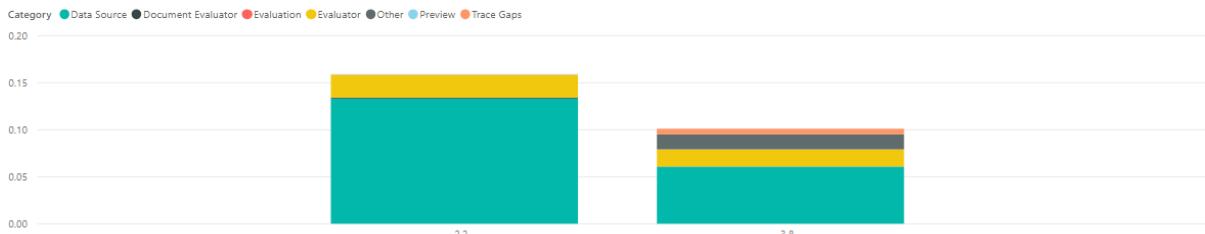
- [Id](#)
- [Start Time](#)
- [Query](#)
- [Step](#)
- [Data Source Query](#)
- [Exclusive Duration \(%\)](#)
- [Row Count](#)
- [Category](#)
- [Is User Query](#)
- [Path](#)

For the second visualization, one choice is to use a Stacked Column Chart. In the 'Axis' parameter, you might want to use 'Id' or 'Step'. If we're looking at the Refresh, because it doesn't have anything to do with steps in the Editor itself, we probably just want to look at 'Id'. For the 'Legend' parameter you should set 'Category' or 'Operation' (depending on the granularity you want). For the 'Value', set 'Exclusive Duration' (and make sure it's not the %, so that you get the raw duration value). Finally, for the Tooltip, set 'Earliest Start Time'.

Once your visualization is built, make sure you sort by 'Earliest Start Time' ascending so you can see the order things happen in.

Detailed Traces Table									
Id	Start Time	Query	Step	Exclusive Duration (%)	Row Count	Category	Is User Query	Path	Data Source Query
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.01	Document Evaluator	False	0/1		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.00	Evaluation	False	0		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.08	Evaluator	False	0/2		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.00	Evaluator	False	0/3		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.01	Evaluator	False	0/3/4		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.00	Evaluator	False	0/3/5		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.03	Evaluator	False	0/3/5/6		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.03	Evaluator	False	0/3/5/7		
2.2	10/15/2019 12:28:13 PM	Customers (2)	Grouped Rows	0.00	Data Source	False	0/3/5/7/8/9/10		

Exclusive Duration and Earliest Start Time by Id and Category



While your exact needs might differ, this combination of charts is a good place to start for looking at a lot of diagnostics files and for a number of purposes.

Interpreting the visualizations

As mentioned above, there's a lot of questions you can try to answer with query diagnostics, but the two that we see the most often are asking how time is spent, and asking what the query sent to the source is.

Asking how the time is spent is easy, and will be very similar for most connectors. A warning with query diagnostics, as mentioned elsewhere, is that you'll see drastically different capabilities depending on the connector. For example, many ODBC based connectors won't have an accurate recording of what query is sent to the actual back-end system, as Power Query only sees what it sends to the ODBC driver.

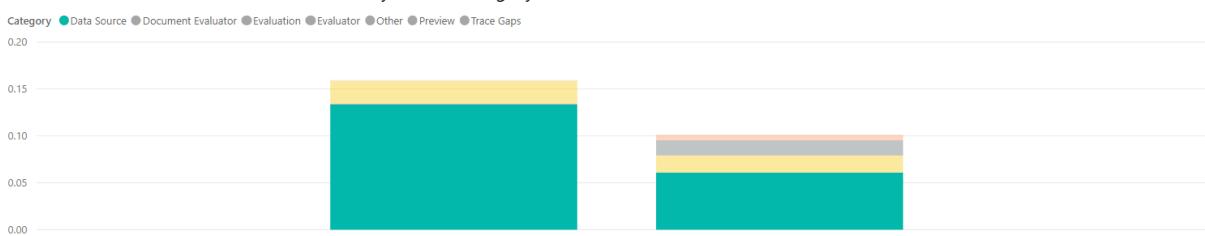
If we want to see how the time is spent, we can just look at the visualizations we built above.

Now, because the time values for the sample queries we're using here are so small, if we want to work with how Power BI reports time it's better if we convert the [Exclusive Duration](#) column to 'Seconds' in the Power Query editor. Once we do this, we can look at our chart and get a pretty decent idea of where time is spent.

For my OData results, I see in the image that the vast majority of the time spent was retrieving the data from source--if I click on the 'Data Source' item on the legend, it'll show me all of the different operations related to sending a query to the Data Source.

Detailed Traces Table		Data Source Query	
Exclusive Duration (%)	Path		
0.40	0/3/13/14/16	Request: GET https://services.odata.org/V4/Northwind/Northwind.svc/Customers?\$filter=ContactTitle eq 'Sales Representative'&\$select=CustomerID%2CCo	
		Response: https://services.odata.org/V4/Northwind/Northwind.svc/Customers?\$filter=ContactTitle eq 'Sales Representative'&\$select=CustomerID%2CCo HTTP/1.1 200 OK	
0.35	0/3/5/7/8/9	Request: https://services.odata.org/V4/Northwind/Northwind.svc/ HTTP/1.1 Content-Type: application/json;odata.metadata=minimal;q=1.0,application/json;odata=minimalmetadata;q=0.9,application/atomsvc+xml;q=0.8,application/a	

Exclusive Duration and Earliest Start Time by Id and Category



If we perform all the same operations and build similar visualizations, but with the SQL traces instead of the ODATA ones, we can see how the two data sources compare!

Detailed Traces Table

Id	Start Time	Query	Step	Exclusive Duration (%)	Row Count	Category	Is User Query	Path	Data Source Query	
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.01		Document Evaluator	False	0/1		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.00		Evaluation	False	0		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.09		Evaluator	False	0/2		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.00		Evaluator	False	0/3		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.00		Evaluator	False	0/3/4		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.00		Evaluator	False	0/3/5		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.06		Evaluator	False	0/3/5/6		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.44		Evaluator	False	0/3/5/7		
2.3	10/15/2019 12:44:37 PM	Customers (3)	Grouped Rows	0.00		Data Source	False	0/3/5/7/8		

Exclusive Duration and Earliest Start Time by Id and Category

Category ● Data Source ● Document Evaluator ● Evaluation ● Evaluator ● Other ● Preview ● SQL ● Trace Gaps



If we click the Data Source table, like with the ODATA diagnostics we can see the first evaluation (2.3 in this image) emits metadata queries, with the second evaluation actually retrieving the data we care about. Because we're retrieving very little data in this case the data pulled back takes very little time (less than a tenth of a second for the entire second evaluation to happen, with less than a twentieth of a second for data retrieval itself), but that won't be true in all cases.

As above, we can click the 'Data Source' category on the legend to see the emitted queries.

Digging into the data

Looking at paths

When you're looking at this, if it seems like time spent is strange--for example, on the OData query you might see that there's a Data Source Query with the following value:

```
https://services.odata.org/V4/Northwind/Northwind.svc/Customers?
$filter=ContactTitle%20eq%20%27Sales%20Representative%27&$select=CustomerID%2CCountry HTTP/1.1
Content-Type:
application/json;odata.metadata=minimal;q=1.0,application/json;odata=minimalmetadata;q=0.9,application/atoms
vc+xml;q=0.8,application/atom+xml;q=0.8,application/xml;q=0.7,text/plain;q=0.7

<Content placeholder>

Response:
Content-Type:
application/json;odata.metadata=minimal;q=1.0,application/json;odata=minimalmetadata;q=0.9,application/atoms
vc+xml;q=0.8,application/atom+xml;q=0.8,application/xml;q=0.7,text/plain;q=0.7
Content-Length: 435

<Content placeholder>
```

This Data Source Query is associated with an operation that only takes up, say, 1% of the Exclusive Duration. Meanwhile, there's a very similar one:

```
Request:
GET https://services.odata.org/V4/Northwind/Northwind.svc/Customers?$filter=ContactTitle eq 'Sales
Representative'&$select=CustomerID%2CCountry HTTP/1.1

Response:
https://services.odata.org/V4/Northwind/Northwind.svc/Customers?$filter=ContactTitle eq 'Sales
Representative'&$select=CustomerID%2CCountry
HTTP/1.1 200 OK
```

This Data Source Query is associated with an operation that takes up nearly 75% of the Exclusive Duration. If you turn on the [Path](#), you discover the latter is actually a child of the former. This means that the first query basically added very little time on its own, with the actual data retrieval being tracked by the 'inner' query.

These are extreme values, but they're within the bounds of what might be seen.

Understanding folding with Query Diagnostics

1/15/2022 • 2 minutes to read • [Edit Online](#)

One of the most common reasons to use Query Diagnostics is to have a better understanding of what operations were 'pushed down' by Power Query to be performed by the back-end data source, which is also known as 'folding'. If we want to see what folded, we can look at what is the 'most specific' query, or queries, that get sent to the back-end data source. We can look at this for both ODATA and SQL.

The operation that was described in the [article](#) on Recording Diagnostics does essentially four things:

- Connects to the data source
- Grabs the customer table
- Filters the Customer ID role to 'Sales Representative'
- Groups by 'Country'

Since the ODATA connector doesn't currently support folding COUNT() to the endpoint, and since this endpoint is somewhat limited in its operations as well, we don't expect that final step to fold. On the other hand, filtering is relatively trivial. This is exactly what we see if we look at the most specific query emitted above:

```
Request:  
GET https://services.odata.org/V4/Northwind/Northwind.svc/Customers?$filter=ContactTitle eq 'Sales  
Representative'&$select=CustomerID%2CCountry HTTP/1.1  
  
Response:  
https://services.odata.org/V4/Northwind/Northwind.svc/Customers?$filter=ContactTitle eq 'Sales  
Representative'&$select=CustomerID%2CCountry  
HTTP/1.1 200 OK
```

We can see we're filtering the table for ContactTitle equallying 'Sales Representative', and we're only returning two columns--Customer ID and Country. Country, of course, is needed for the grouping operation, which since it isn't being performed by the ODATA endpoint must be performed locally. We can conclude what folds and doesn't fold here.

Similarly, if we look at the specific and final query emitted in the SQL diagnostics, we see something slightly different:

```
count(1) as [Count]  
from  
(  
    select [__].[Country]  
    from [dbo].[Customers] as [__]  
    where [__].[ContactTitle] = 'Sales Representative' and [__].[ContactTitle] is not null  
) as [rows]  
group by [Country]
```

Here, we can see that Power Query creates a subselection where ContactTitle is filtered to 'Sales Representative', then groups by Country on this subselection. All of our operations folded.

Using Query Diagnostics, we can examine what kind of operations folded--in the future, we hope to make this capability easier to use.

Why does my query run multiple times?

1/15/2022 • 5 minutes to read • [Edit Online](#)

When refreshing in Power Query, there's a lot done behind the scenes to attempt to give you a smooth user experience, and to execute your queries efficiently and securely. However, in some cases you might notice that multiple data source requests are being triggered by Power Query when data is refreshed. Sometimes these requests are normal, but other times they can be prevented.

When multiple requests occur

The following sections describe a few instances when Power Query can send multiple requests to a data source.

Connector design

Connectors can make multiple calls to a data source for various reasons, including metadata, caching of results, pagination, and so on. This behavior is normal and is designed to work that way.

Multiple queries referencing a single data source

Multiple requests to the same data source can occur if multiple queries pull from that data source. These requests can happen even in a case where only one query references the data source. If that query is referenced by one or more other queries, then each query—along with all the queries it depends on—is evaluated independently.

In a desktop environment, a single refresh of all the tables in the data model is run using a single shared cache. Caching can reduce the likelihood of multiple requests to the same data source, since one query can benefit from the same request having already been run and cached for a different query. Even here, though, you can get multiple requests either because the data source isn't cached (for example, local CSV files), the request to the data source is different than a request that was already cached because of downstream operations (which can alter folding), the cache is too small (which is relatively unlikely), or because the queries are running at roughly the same time.

In a cloud environment, each query is refreshed using its own separate cache, so a query can't benefit from the same request having already been cached for a different query.

Folding

Sometimes Power Query's folding layer may generate multiple requests to a data source, based on the operations being performed downstream. In such cases, you might avoid multiple requests by using `Table.Buffer`. More information: [Buffer your table](#)

Loading to the Power BI Desktop model

In Power BI Desktop, Analysis Services (AS) refreshes data by using two evaluations: one to fetch the schema—which AS does by asking for zero rows—and one to fetch the data. If computing the zero-row schema requires fetching the data, then duplicate data source requests can occur.

Data privacy analysis

Data privacy does its own evaluations of each query to determine whether the queries are safe to run together. This evaluation can sometimes cause multiple requests to a data source. A telltale sign that a given request is coming from data privacy analysis is that it will have a "TOP 1000" condition (although not all data sources support such a condition). In general, disabling data privacy—assuming that's acceptable—would eliminate the "TOP 1000" or other data privacy-related requests during refresh. More information: [Disable the data privacy firewall](#)

Background data downloads (also known as “background analysis”)

Similar to the evaluations performed for data privacy, the Power Query editor by default will download a preview of the first 1000 rows of each query step. Downloading these rows helps ensure the data preview is ready to display as soon as a step is selected, but it can also cause duplicate data source requests. More information: [Disable background analysis](#)

Miscellaneous Power Query editor background tasks

Various Power Query editor background tasks can also trigger extra data source requests (for example, step folding analysis, column profiling, the automatic refresh of the 1000-row preview that Power Query triggers after loading results to Excel, and so on).

Isolating multiple queries

You can isolate instances of multiple queries by turning off specific parts of the query process to isolate where the duplicate requests are coming from. For example, if you start:

- In the Power Query editor
- With the firewall disabled
- With background analysis disabled
- With [column profiling](#) and any other background tasks disabled
- [Optional] Doing a `Table.Buffer`

In this example, you'll have only a single M evaluation that happens when you refresh the Power Query editor preview. If the duplicate requests occur at this point, then they're somehow inherent in the way the query is authored. If not, and if you enable the settings above one-by-one, you can then observe at what point the duplicate requests start occurring.

The following sections describe these steps in more detail.

Set up Power Query editor

You don't need to reconnect or recreate your query, just open the query you want to test in the Power Query editor. You can [duplicate the query](#) in the editor if you don't want to mess with the existing query.

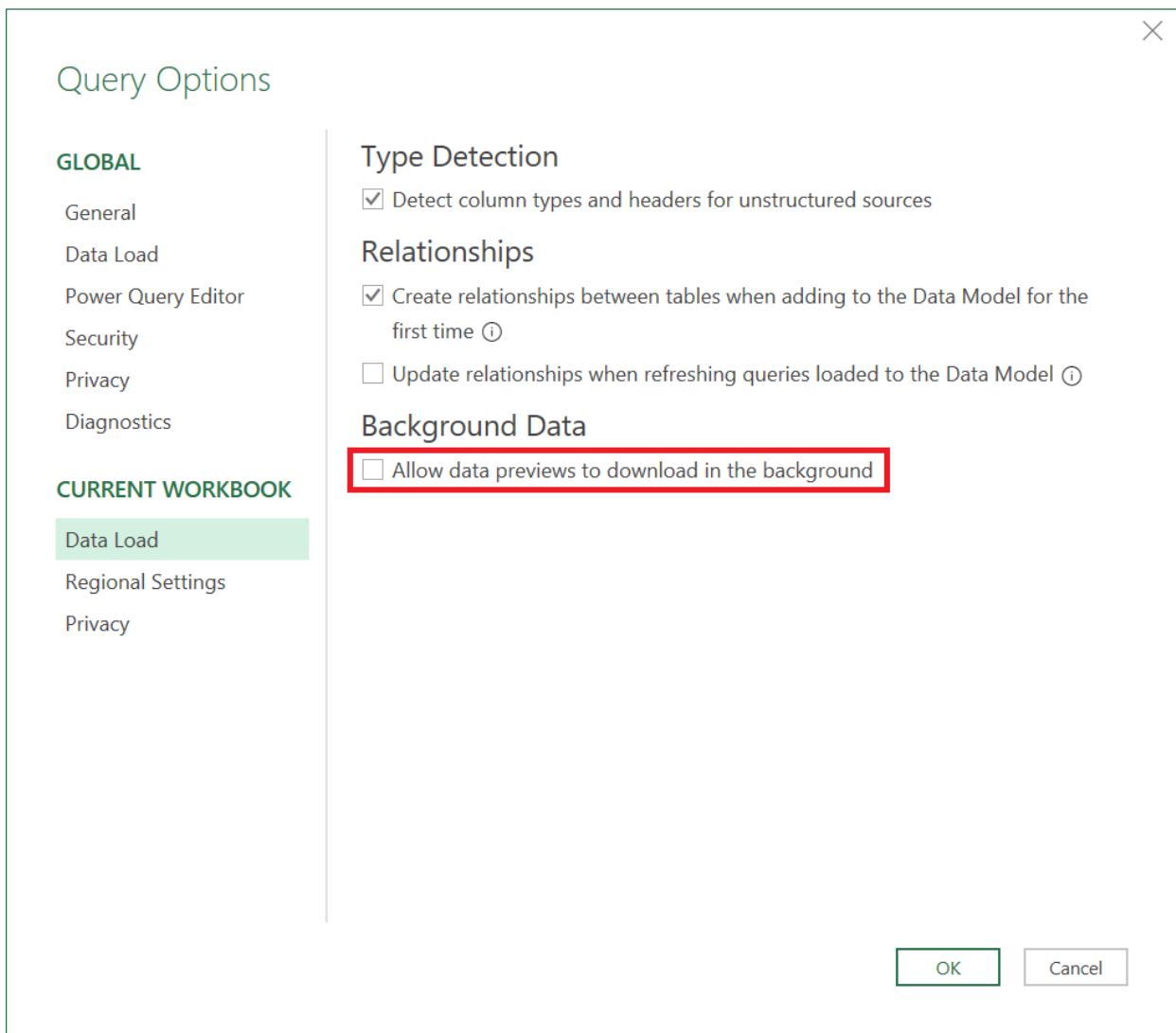
Disable the data privacy firewall

The next step is to disable the data privacy firewall. This step assumes you aren't concerned about data leakage between sources, so disabling the data privacy firewall can be done using the **Always ignore Privacy Level settings** described in [Set Fast Combine option](#) in Excel or using the **Ignore the Privacy levels and potentially improve performance** setting described in [Power BI Desktop privacy levels](#) in Power BI Desktop.

Be sure to undo this step before resuming normal testing.

Disable background analysis

The next step is to disable background analysis. Background analysis is controlled by the **Allow data preview to download in the background** setting described in [Disable Power Query background refresh](#) for Power BI. You can also disable this option in Excel.



Buffer your table

Optionally, you can also use `Table.Buffer` to force all the data to be read, which imitates what happens during a load. To use `Table.Buffer` in the Power Query editor:

1. In the Power Query editor formula bar, select the **fx** button to add a new step.

Segment	Country	Product
Government	United States of America	Paseo

2. In the formula bar, surround the name of the previous step with `Table.Buffer(<previous step name goes here>)`. For example, if the previous step was named `Source`, the formula bar will display `= Source`. Edit the step in the formula bar to say `= Table.Buffer(Source)`.

More information: [Table.Buffer](#)

Run the test

To run the test, do a refresh in the Power Query editor.

The screenshot shows the Microsoft Power Query Editor interface. The ribbon at the top has tabs for File, Home, Transform, Add Column, View, Tools, and Help. The Home tab is selected. Below the ribbon, there are several icons: Close & Apply, New, Recent, Enter, Data source settings, Manage Parameters, Refresh Preview (which is highlighted with a red box), Properties, Advanced Editor, Choose Columns, and Remove Column. The main area shows a query named "Financial Sample" with one row. The columns are labeled "Segment", "Country", and "Product". The "Segment" column contains "Government", the "Country" column contains "United States of America", and the "Product" column contains "Paseo".

	Segment	Country	Product
1	Government	United States of America	Paseo

Step folding indicators

1/15/2022 • 4 minutes to read • [Edit Online](#)

NOTE

Before reading this article, we recommended that you read [Query folding in Power Query](#) to better understand how folding works in Power Query.

Step folding indicators allow you to understand the steps that fold or not fold.

Using step folding indicators, when you make a change that breaks folding, it will become obvious. This will allow you to more easily resolve issues quickly, avoid performance issues in the first place, and have better insight into your queries. In most cases you run into, steps will fold, or not fold. There are many cases where the outcome isn't as obvious, discussed in the section on [Step diagnostics indicators](#) (Dynamic, Opaque, and Unknown) later in this article.

NOTE

The step folding indicators feature is available only for Power Query Online.

Interpreting step diagnostics

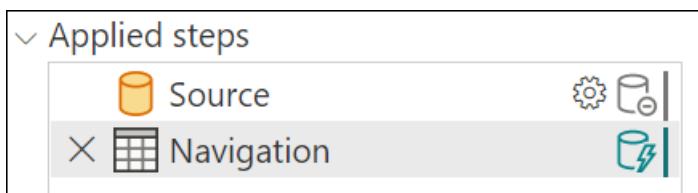
When looking at step diagnostics, the most important thing to understand is that the diagnostic state isn't sequential. In other words, the indicator for that step describes whether the query as a whole, up to that point, folds or not. If you have an indicator that shows that the query doesn't fold, followed by an indicator that shows it does fold, it means that every step up to that point does fold.

You can see an example of this even with a simple query against a SQL source.

Using the [AdventureWorks sample database](#), connect to the Products table and load data. Doing this through the Navigation experience will give the following query:

```
let
    Source = Sql.Database("ServerName", "AdventureWorks"),
    Navigation = Source{[Schema = "Production", Item = "Product"]}[Data]
in
    Navigation
```

If you look at how this shows up in step folding indicators, you can see that the first step doesn't fold, the second step is inconclusive, and the third step folds.



You can see that the initial steps don't fold, but the final step generated when you load data initially does fold. How the first steps (**Source**, and sometimes other **Navigation** steps) are handled depends on the connector. With SQL, for example, it's handled as a catalog table value, which doesn't fold. However, as soon as you select

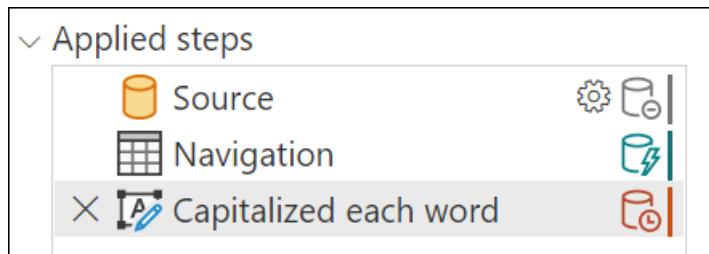
data for that connector it will.

Conversely, this can also mean that your query folds up to a point and then stops folding. Unlike in the case where you have a folding indicator for the step, which shows that everything folds when you have a not-folding indicator it doesn't mean that everything doesn't fold. Instead, it means that "not everything" folds. Generally, everything up to the last folding indicator will fold, with more operations happening after.

Modifying the example from above, you can give a transform that never folds - *Capitalize Each Word*.

```
let
    Source = Sql.Database("ServerName", "AdventureWorks"),
    Navigation = Source{[Schema = "Production", Item = "Product"]}[Data],
    #"Capitalized each word" = Table.TransformColumns(Navigation, {"Name", each Text.Proper(_), type text})
in
    #"Capitalized each word"
```

In step folding indicators, you will see that you have the exact same indicators as above, except the final step doesn't fold. Everything up to this final step will be performed on the data source, while the final step will be performed locally.



Step diagnostics indicators

Step folding indicators use an underlying query plan, and require it to be able to get information about the query to report it. Currently the query plan only supports tables, so some cases (lists, records, primitives) will not report as folding or not. Similarly, constant tables will report as opaque.

INDICATOR	ICON	DESCRIPTION
Folding		The folding indicator tells you that the query up to this step will be evaluated by the data source.
Not folding		The not-folding indicator tells you that some part of the query up to this step will be evaluated outside the data source. You can compare it with the last folding indicator, if there is one, to see if you can rearrange your query to be more performant.
Might fold		Might fold indicators are uncommon. They mean that a query 'might' fold. They indicate either that folding/not folding will be determined at runtime, when pulling results from the query, and that the query plan is dynamic. These will likely only appear with ODBC or OData connections.

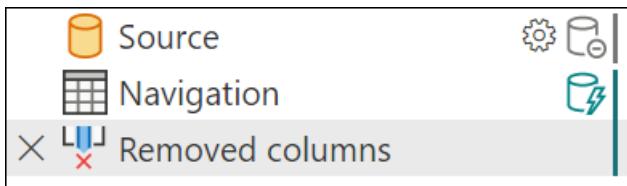
INDICATOR	ICON	DESCRIPTION
Opaque		Opaque indicators tell you that the resulting query plan is inconclusive for some reason. It generally indicates that there is a true 'constant' table, or that that transform or connector is not supported by the indicators and query plan tool.
Unknown		Unknown indicators represent an absence of query plan, either due to an error or attempting to run the query plan evaluation on something other than a table (such as a record, list, or primitive).

Example analysis

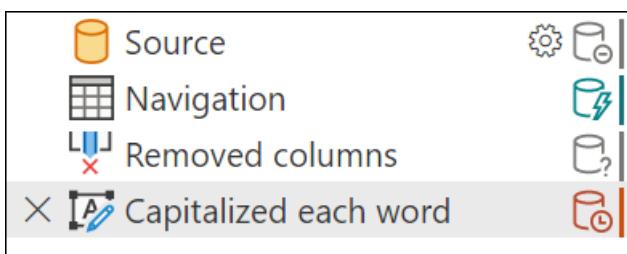
You can see an example by connecting to the Products table in Adventure Works (SQL). The initial load, similar to above, will look as follows:



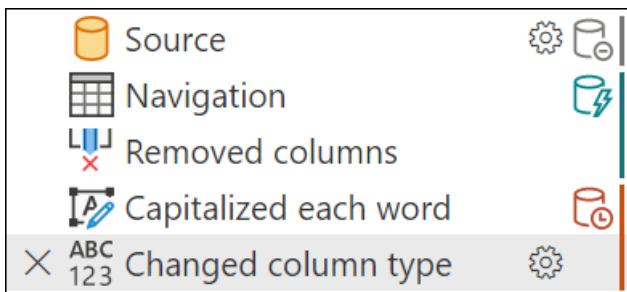
Adding more steps that fold will extend that green line. This is because this step also folds.



Adding a step that doesn't fold will show an indicator, for example, **Capitalize each word** will never fold. You can see that the indicator changes, showing that as of this step, it's stopped folding. As mentioned earlier, the previous steps will still fold.

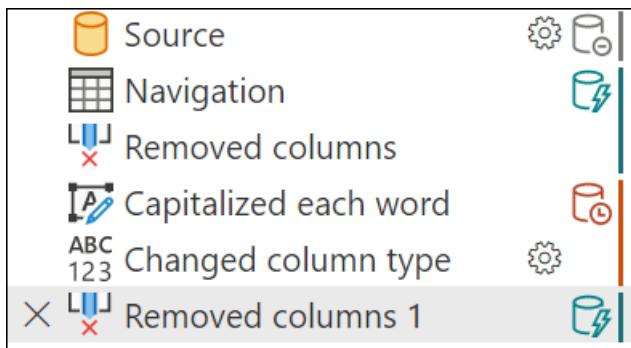


Adding more steps downstream that depend on **Capitalize each word** step will continue to not fold.



However, if you remove the column you applied the capitalization to so that the optimized query plan can all

fold once more, you'll get a result like this; although something like this is uncommon. This shows you how it's not just the order of steps, but the actual transformations that apply as well.



Query plan for Power Query (Preview)

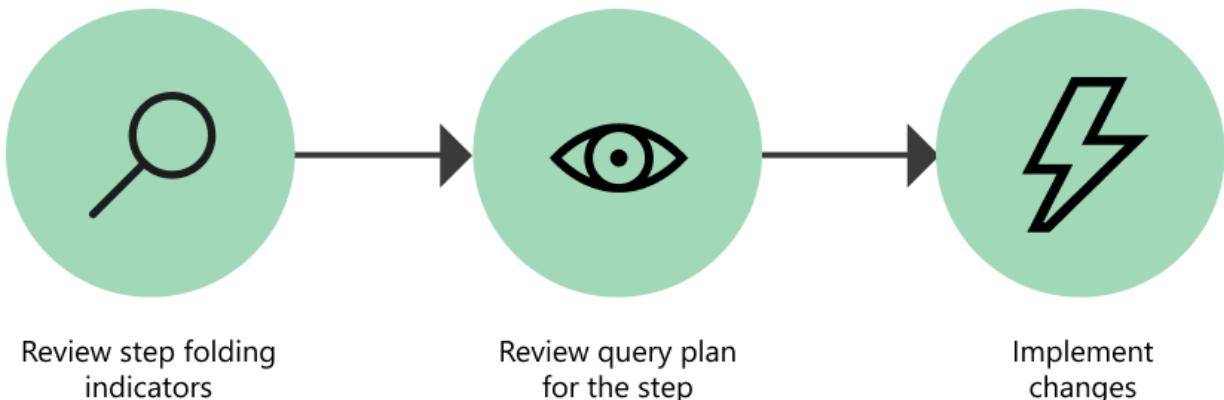
1/15/2022 • 7 minutes to read • [Edit Online](#)

Query plan for Power Query is a feature that provides a better view of your query's evaluation. It's useful to help determine why a particular query might not fold at a particular step.

Through a practical example, this article will demonstrate the main use case and potential benefits of using the query plan feature to review your query steps. The examples used in this article have been created using the AdventureWorksLT sample database for Azure SQL Server, which you can download from [AdventureWorks sample databases](#).

NOTE

The query plan feature for Power Query is only available in Power Query Online.



This article has been divided in a series of recommended steps in order to interpret the query plan. These steps are:

1. [Review the step folding indicators](#).
2. [Select the query step to review its query plan](#).
3. [Implement changes to your query](#).

Use the following steps to create the query in your own Power Query Online environment.

1. From **Power Query - Choose data source**, select **Blank query**.
2. Replace the blank query's script with the following query.

```
let
    Source = Sql.Database("servername", "database"),
    Navigation = Source{[Schema = "Sales", Item = "SalesOrderHeader"]}[Data],
    #"Removed other columns" = Table.SelectColumns(Navigation, {"SalesOrderID", "OrderDate",
    "SalesOrderNumber", "PurchaseOrderNumber", "AccountNumber", "CustomerID", "TotalDue"}),
    #"Filtered rows" = Table.SelectRows(#"Removed other columns", each [TotalDue] > 1000),
    #"Kept bottom rows" = Table.LastN(#"Filtered rows", 5)
in
    #"Kept bottom rows"
```

3. Change `servername` and `database` with the correct names for your own environment.
4. (Optional) If you're trying to connect to a server and database for an on-premises environment, be sure

to configure a gateway for that environment.

5. Select Next.

6. In the Power Query Editor, select **Configure connection** and provide the credentials to your data source.

NOTE

For more information about connecting to a SQL Server, go to [SQL Server database](#).

After following these steps, your query will look like the one in the following image.

The screenshot shows the Power Query Editor interface. The main area displays a table with the following data:

SalesOrderID	OrderDate	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID	TotalDue
74138	5/30/2014, 12:00:00 ...	SO74138	null	10-4030-026303	26303	1,381.07
74139	5/30/2014, 12:00:00 ...	SO74139	null	10-4030-013967	13967	2,650.32
74142	5/30/2014, 12:00:00 ...	SO74142	null	10-4030-022253	22253	1,297.80
74144	5/30/2014, 12:00:00 ...	SO74144	null	10-4030-012314	12314	2,673.06
74145	5/30/2014, 12:00:00 ...	SO74145	null	10-4030-013650	13650	2,634.40

The 'Query settings' pane on the right shows the query name is 'Query' and the applied steps include 'Kept bottom rows'.

This query connects to the SalesOrderHeader table, and selects a few columns from the last five orders with a **TotalDue** value above 1000.

NOTE

This article uses a simplified example to showcase this feature, but the concepts described in this article apply to all queries. We recommend that you have a good knowledge of query folding before reading the query plan. To learn more about query folding, go to [Query folding basics](#).

1. Review the step folding indicators

NOTE

Before reading this section, we recommend that you review the article on [Step folding indicators](#).

Your first step in this process is to review your query and pay close attention to the step folding indicators. The goal is to review the steps that are marked as not folded. Then you can see if making changes to the overall query could make those transformations fold completely.

The screenshot shows the 'Query settings' dialog with the following details:

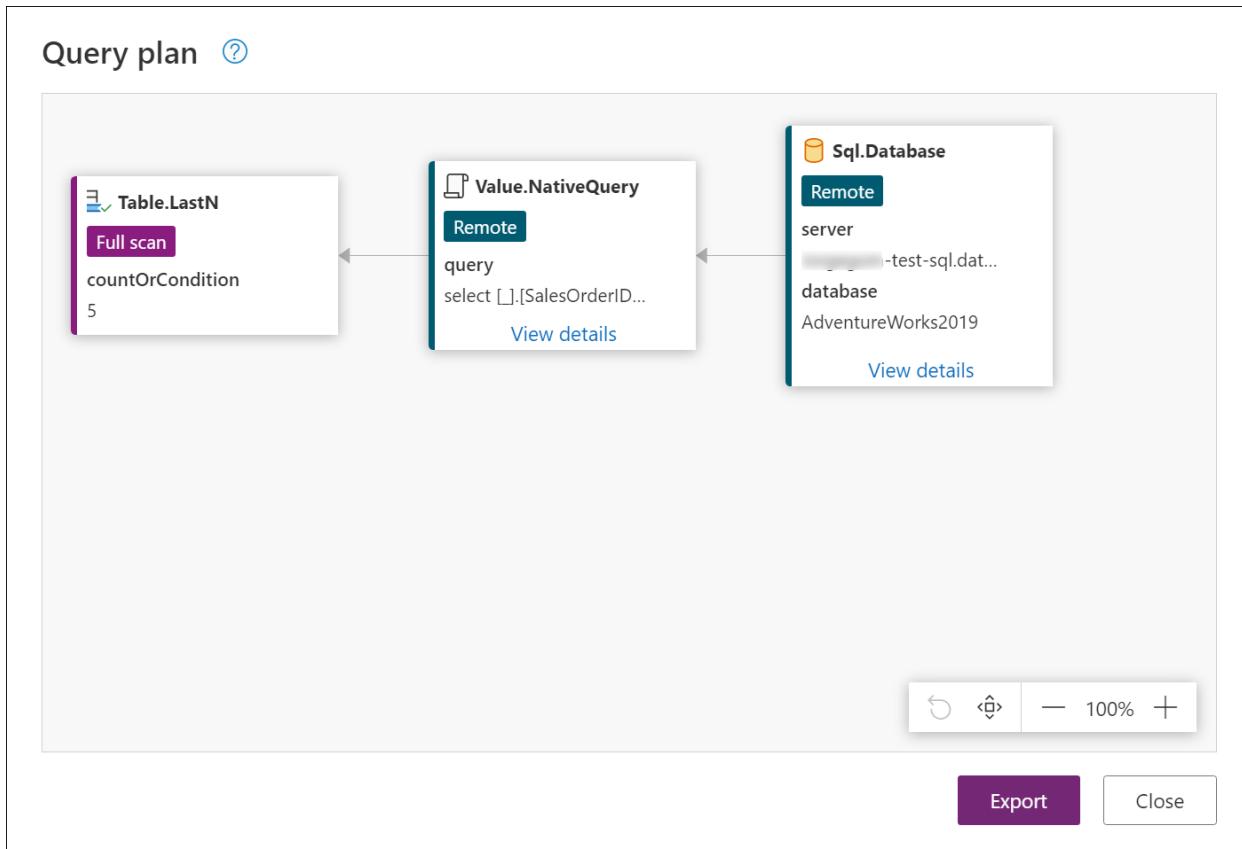
- Name:** Query
- Applied steps:**
 - Source (folded)
 - Navigation (folded)
 - Removed other columns (folded)
 - Filtered rows (folded)
 - Kept bottom rows (not folded)

For this example, the only step that can't be folded is **Kept bottom rows**, which is easy to identify through the *not folded* step indicator. This step is also the last step of the query.

The goal now is to review this step and understand what's being folded back to the data source and what can't be folded.

2. Select the query step to review its query plan

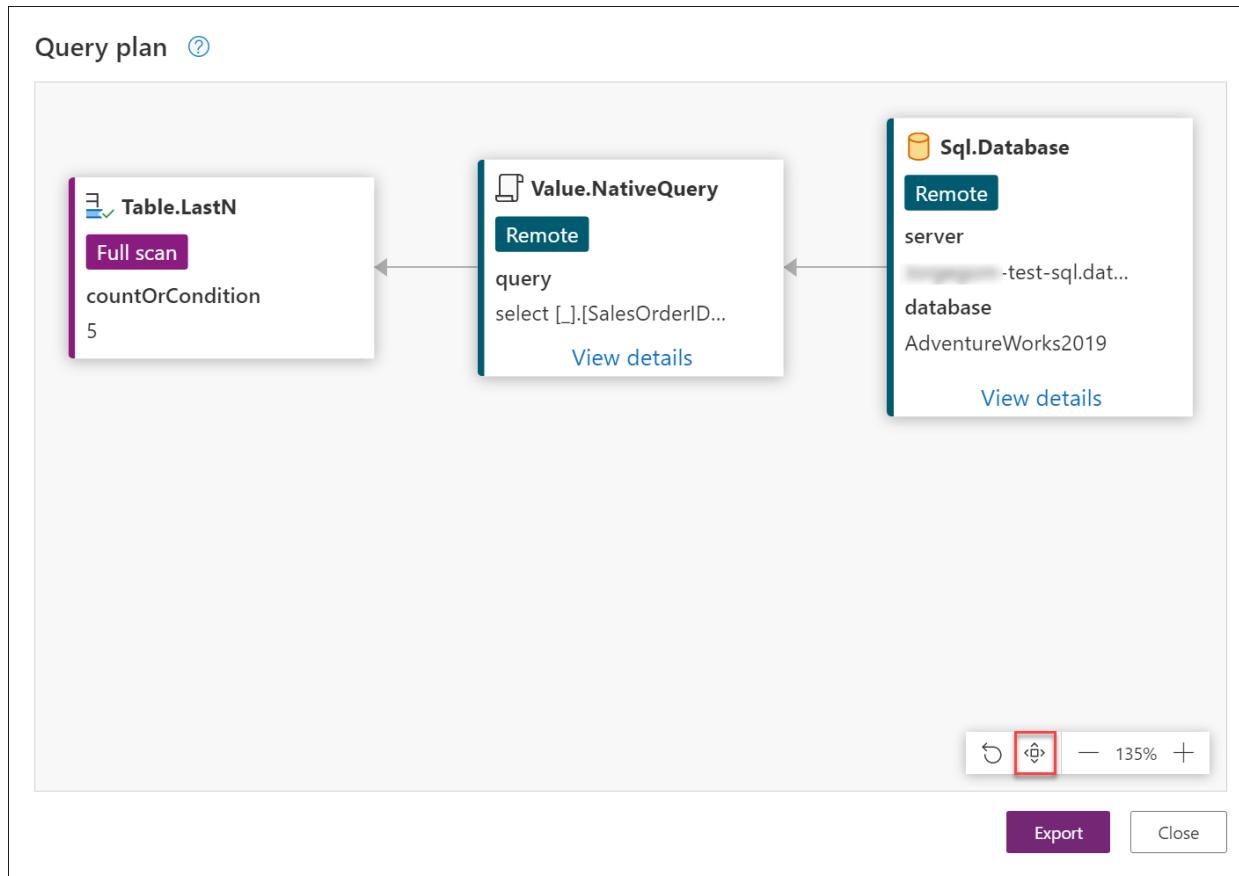
You've identified the **Kept bottom rows** step as a step of interest since it doesn't fold back to the data source. Right-click the step and select the **View Query plan** option. This action displays a new dialog that contains a diagram for the query plan of the selected step.



Power Query tries to optimize your query by taking advantage of lazy evaluation and query folding, as

mentioned in [Query folding basics](#). This query plan represents the optimized translation of your M query into the native query that's sent to the data source. It also includes any transforms that are performed by the Power Query Engine. The order in which the nodes appear follows the order of your query starting from the last step or output of your query, which is represented on the far left of the diagram and in this case is the `Table.LastN` node that represents the *Kept bottom rows* step.

At the bottom of the dialog, there's a bar with icons that help you zoom in or out of the query plan view, and other buttons to help you manage the view. For the previous image, the *Fit to view* option from this bar was used to better appreciate the nodes.



NOTE

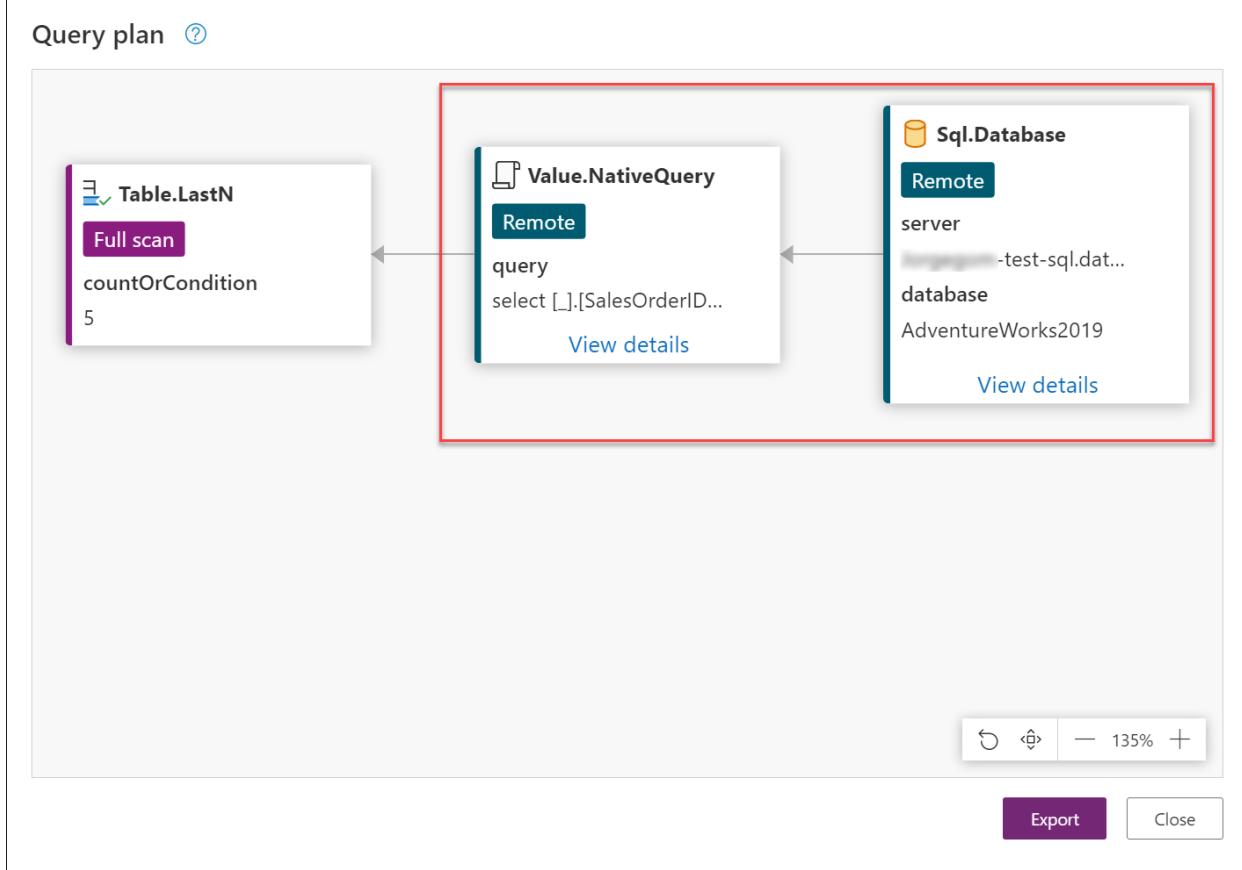
The query plan represents the optimized plan. When the engine is evaluating a query, it tries to fold all operators into a data source. In some cases, it might even do some internal reordering of the steps to maximize folding. With this in mind, the nodes/operators left in this optimized query plan typically contain the "folded" data source query and any operators that couldn't be folded and are evaluated locally.

Identify folded nodes from other nodes

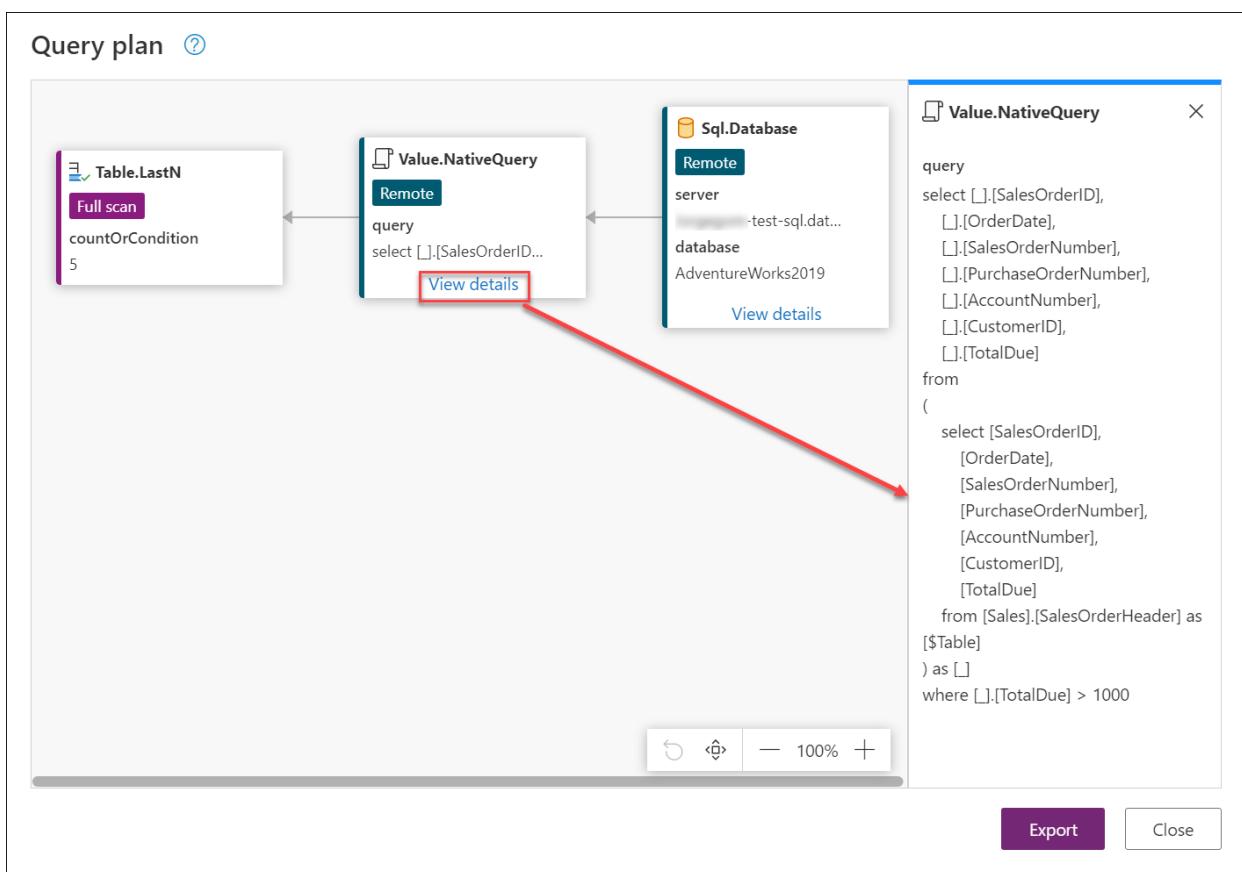
You can identify the nodes in this diagram as two groups:

- Folded nodes:** This node can be either `Value.NativeQuery` or "data source" nodes such as `Sql.Database`. These can also be identified with the label *remote* under their function name.
- Non-folded nodes:** Other table operators, such as `Table.SelectRows`, `Table.SelectColumns`, and other functions that couldn't be folded. These can also be identified with the labels *Full scan* and *Streaming*.

The following image shows the folded nodes inside the red rectangle. The rest of the nodes couldn't be folded back to the data source. You'll need to review the rest of the nodes since the goal is to attempt to have those nodes fold back to the data source.



You can select **View details** at the bottom of some nodes to display extended information. For example, the details of the **Value.NativeQuery** node show the native query (in SQL) that will be sent to the data source.



The query shown here might not be exactly the same query sent to the data source, but it's a good approximation. For this case, it tells you exactly what columns will be queried from the SalesOrderHeader table and then how it will filter that table using the TotalDue field to only get rows where the value for that field is larger than 1000. The node next to it, **Table.LastN**, is calculated locally by the Power Query engine, as it can't be

folded.

NOTE

The operators might not exactly match the functions used in the query's script.

Review non-folded nodes and consider actions to make your transform fold

You've now determined which nodes couldn't be folded and will be evaluated locally. This case only has the `Table.LastN` node, but in other scenarios it could have many more.

The goal is to apply changes to your query so that the step can be folded. Some of the changes you might implement could range from rearranging your steps to applying an alternative logic to your query that's more explicit to the data source. This doesn't mean that all queries and all operations are foldable by applying some changes. But it's a good practice to determine through trial and error if your query could be folded back.

Since the data source is a SQL Server database, if the goal is to retrieve the last five orders from the table, then a good alternative would be to take advantage of the `TOP` and `ORDER BY` clauses in SQL. Since there's no `BOTTOM` clause in SQL, the `Table.LastN` transform in PowerQuery can't be translated into SQL. You could remove the `Table.LastN` step and replace it with:

- **A sort descending step** by the `SalesOrderID` column in the table, since this column determines which order goes first and which has been entered last.
- **Select the top five rows** since the table has been sorted, this transform accomplishes the same as if it was a *Kept bottom rows* (`Table.LastN`).

This alternative is equivalent to the original query. While this alternative in theory seems good, you need to make the changes to see if this alternative will make this node fully fold back to the data source.

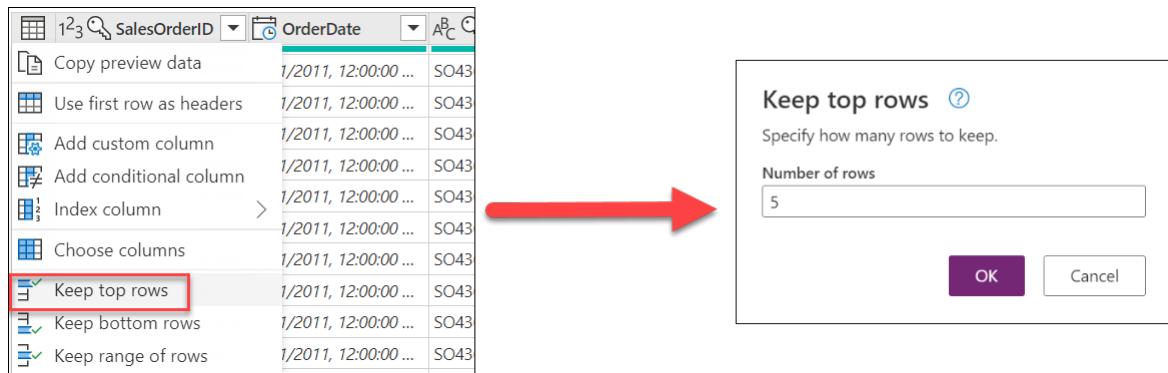
3. Implement changes to your query

Implement the alternative discussed in the previous section:

1. Close the query plan dialog and go back to the Power Query Editor.
2. Remove the *Kept bottom rows* step.
3. Sort the `SalesOrderID` column in descending order.

The screenshot shows the Power Query Editor interface with a table preview. The 'SalesOrderID' column header has a context menu open. The menu items are: 'Sort ascending' (with a downward arrow icon), 'Sort descending' (with a downward arrow icon and a red box around it), 'Remove empty', and 'Number filters'. Below the menu is a search bar labeled 'Search'.

4. Select the table icon on the top-left corner of the data preview view and select the option that reads *Keep top rows*. In the dialog, pass the number five as the argument and hit OK.



After implementing the changes, check the step folding indicators again and see if it's giving you a folded indicator.

The screenshot shows the Power Query interface with the 'Edit queries' window open. The 'Applied steps' pane on the right side lists a single step named 'Kept top rows'. The main area displays a table with 5 rows of data, each containing columns like SalesOrderID, OrderDate, SalesOrderNumber, PurchaseOrderNumber, AccountNumber, CustomerID, and TotalDue. The table has a header row and 4 data rows.

Now it's time to review the query plan of the last step, which is now **Keep top rows**. Now there are only folded nodes. Select **View details** under `Value.NativeQuery` to verify which query is being sent to the database.

The screenshot shows the Power BI Query Editor with a 'Value.NativeQuery' node selected. The node is expanded, showing its connection details: 'Sql.Database', 'Remote', 'server: -test-sql.dat...', 'database: AdventureWorks2019'. To the right, the expanded node shows the native query code:

```

query
select top 5 [SalesOrderID], [OrderDate], [SalesOrderNumber], [PurchaseOrderNumber], [AccountNumber], [CustomerID], [TotalDue]
from (
    select [SalesOrderID], [OrderDate], [SalesOrderNumber], [PurchaseOrderNumber], [AccountNumber], [CustomerID], [TotalDue]
    from [Sales].[SalesOrderHeader] as [$Table]
) as [L]
where [TotalDue] > 1000
  
```

While this article is suggesting what alternative to apply, the main goal is for you to learn how to use the query plan to investigate query folding. This article also provides visibility of what's being sent to your data source and what transforms will be done locally.

You can adjust your code to see the impact that it has in your query. By using the step folding indicators, you'll also have a better idea of which steps are preventing your query from folding.

Using parameters

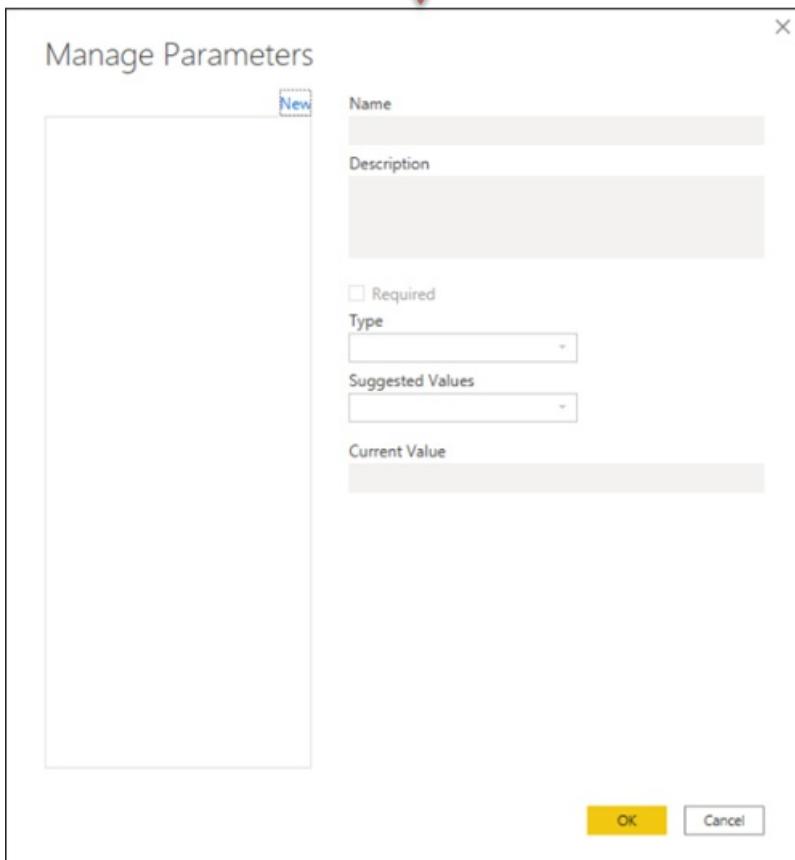
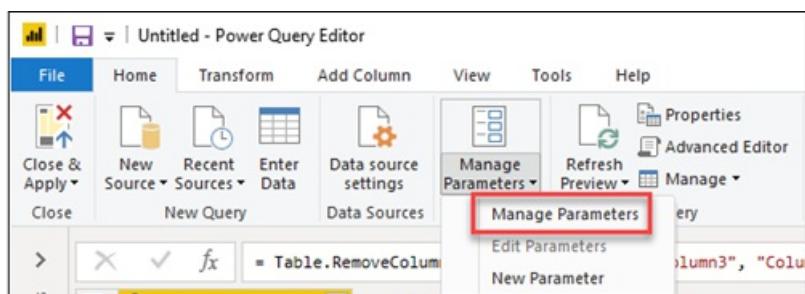
1/15/2022 • 5 minutes to read • [Edit Online](#)

A parameter serves as a way to easily store and manage a value that can be reused.

Parameters give you the flexibility to dynamically change the output of your queries depending on their value, and can be used for:

- Changing the argument values for particular transforms and data source functions
- Inputs in custom functions

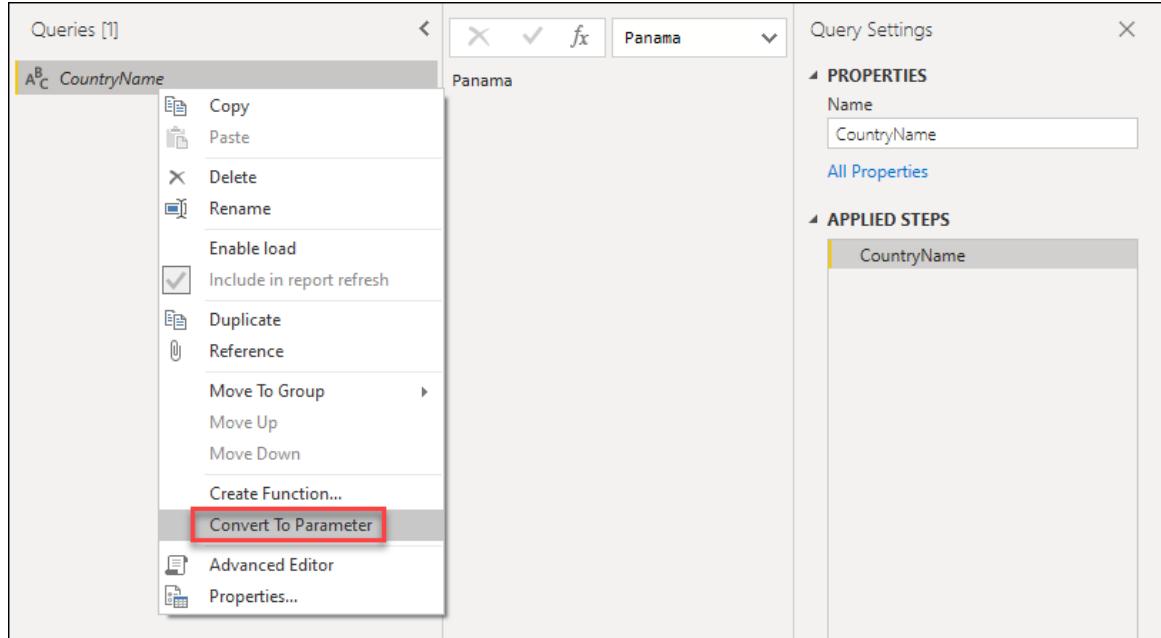
You can easily manage your parameters inside the **Manage Parameters** window. You can get to the **Manage Parameters** window by selecting the **Manage Parameters** option inside **Manage Parameters** in the **Home** tab.



Creating a parameter

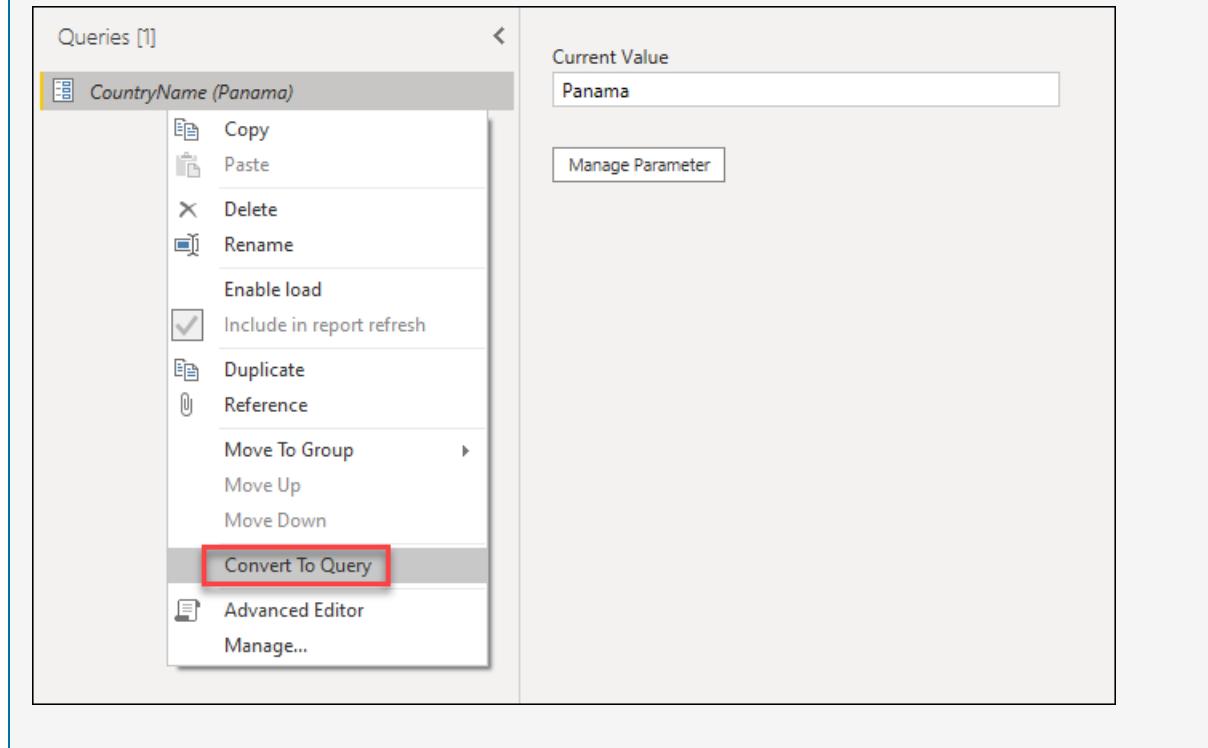
Power Query provides two easy ways to create parameters:

- **From an existing query**—You can easily right-click a query whose value is a simple non-structured constant such as, but not limited to, a date, text, or number, and select **Convert to Parameter**.

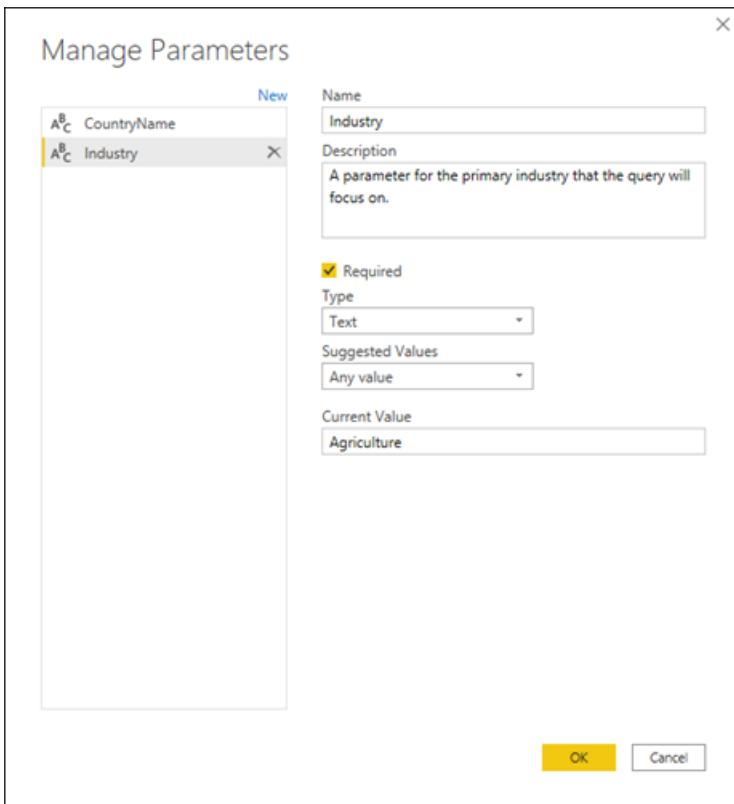


NOTE

You can also convert a parameter to a query by right-clicking the parameter and then selecting **Convert To Query**, as shown in the following image.



- **Using the Manage Parameters window**—You can select the **New Parameter** option from the dropdown menu of **Manage Parameters** in the **Home** tab, or you can launch the **Manage Parameters** window and select in the **New** button on the top to create a parameter. You can fill in this form and select **OK** to create a new parameter.



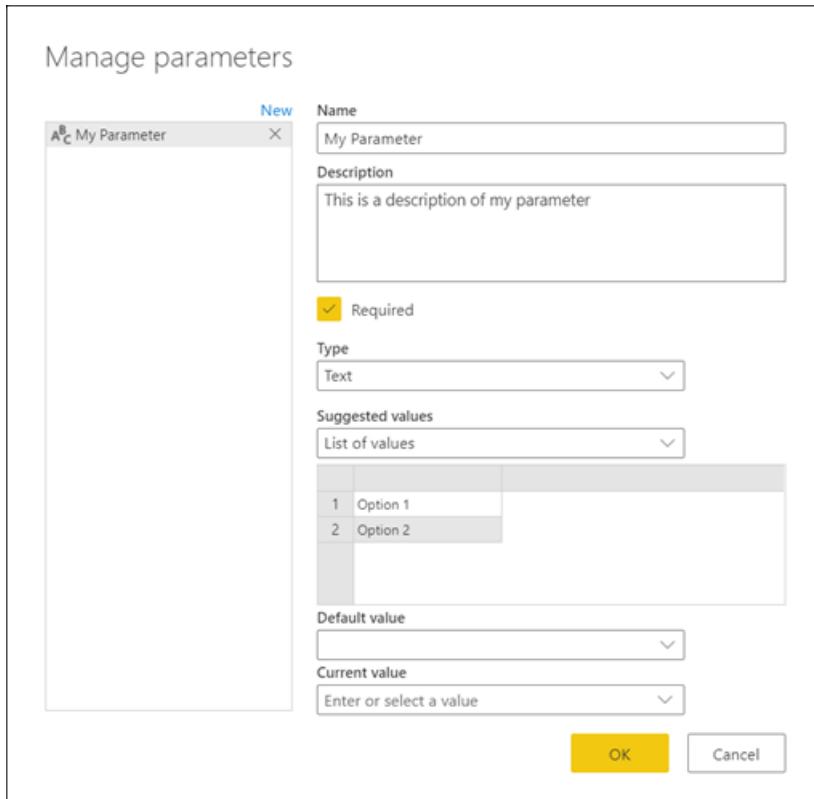
After creating the parameter, you can always go back to the **Manage Parameters** window to modify any of your parameters at any moment.

Parameter properties

A parameter stores a value that can be used for transformations in Power Query. Apart from the name of the parameter and the value that it stores, it also has other properties that provide metadata to it. The properties of a parameter are as follows.

- **Name**—Provide a name for this parameter that lets you easily recognize and differentiate it from other parameters you might create.
- **Description**—The description is displayed next to the parameter name when parameter information is displayed, helping users who are specifying the parameter value to understand its purpose and its semantics.
- **Required**—The checkbox indicates whether subsequent users can specify whether a value for the parameter must be provided.
- **Type**—We recommend that you always set up the data type of your parameter. You can learn more about the importance of data types from the [Data types](#) article.
- **Suggested Values**—Provides the user with suggestions to select a value for the **Current Value** from the available options:
 - **Any value**—The current value can be any manually entered value.
 - **List of values**—Provides you with a simple table-like experience so you can define a list of suggested values that you can later select from for the **Current Value**. When this option is selected, a new option called **Default Value** will be made available. From here you can select what should be the default value for this parameter, which will be the default value shown to the user when referencing the parameter. This value isn't the same as the **Current Value**, which is the value that's stored inside the parameter and can be passed as an argument in transformations. Using the *List of values* will enable a drop-down menu to be displayed in the **Default Value** and

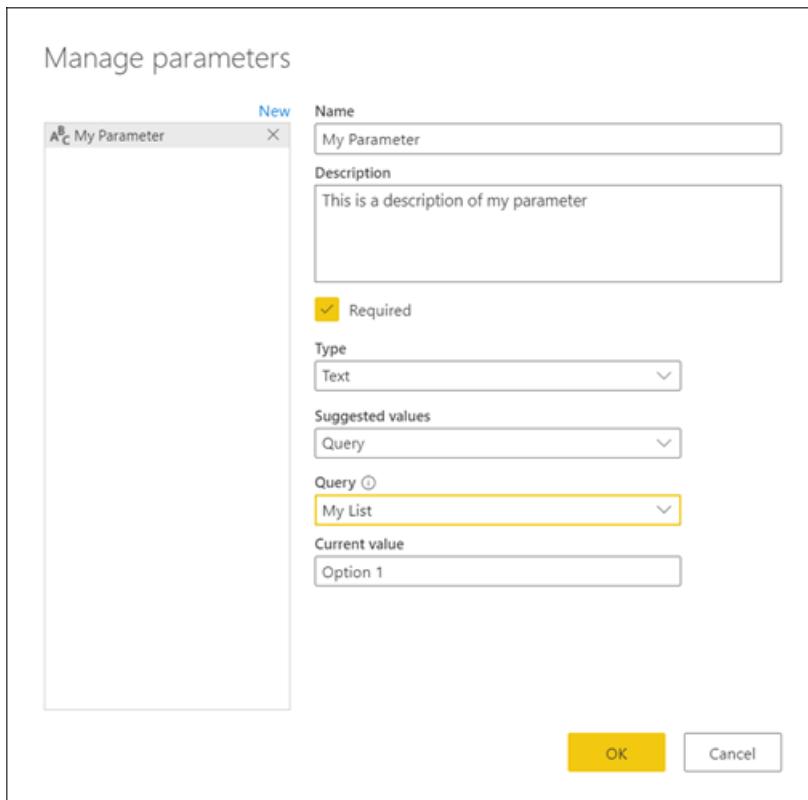
Current Value fields, where you can pick one of the values from the suggested list of values.



NOTE

You can still manually type any value that you want to pass to the parameter. The list of suggested values only serves as simple suggestions.

- **Query**—Uses a list query (a query whose output is a list) to provide the list of suggested values that you can later select for the **Current Value**.



- **Current Value**—The value that will be stored in this parameter.

Where to use parameters

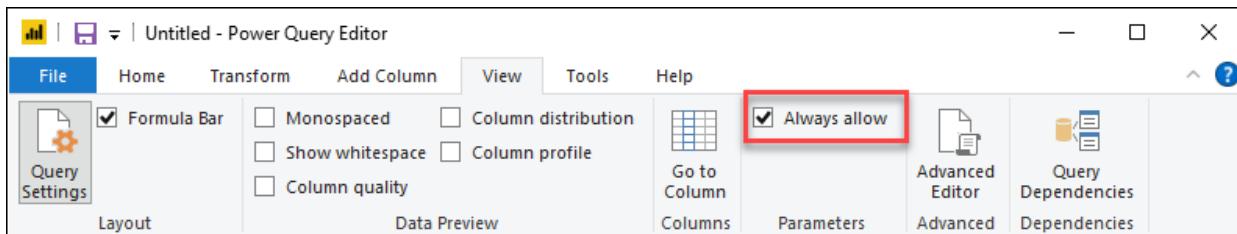
A parameter can be used in many different ways, but it's more commonly used in two scenarios:

- **Step argument**—You can use a parameter as the argument of multiple transformations driven from the user interface (UI).
- **Custom Function argument**—You can create a new function from a query and reference parameters as the arguments of your custom function.

In the next sections, you'll see an example for these two scenarios.

Step argument

To enable this feature, first go to the **View** tab in the Power Query Editor and enable the **Always allow** option in the **Parameters** group.



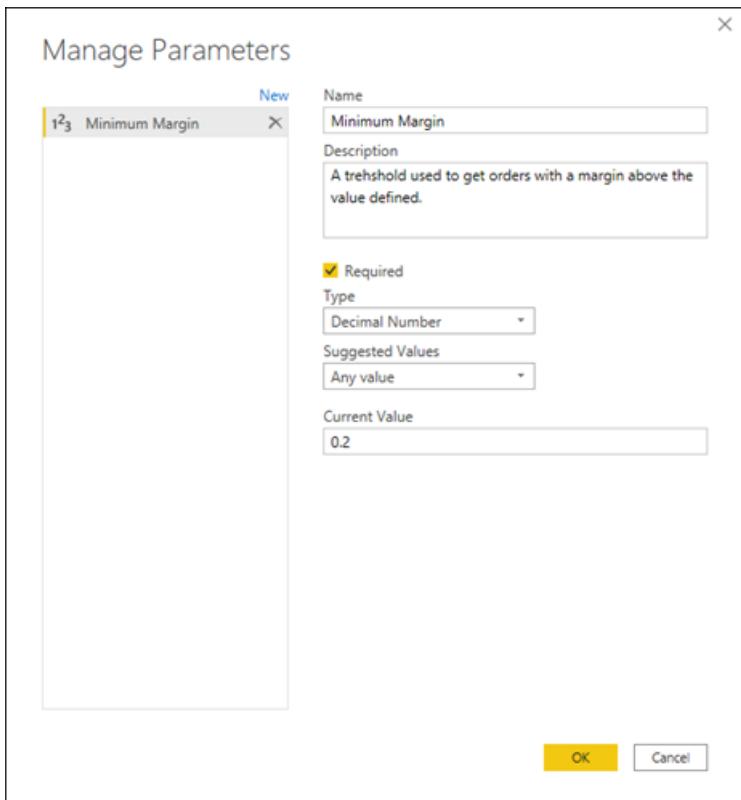
NOTE

This feature is currently not available in Power Query Online.

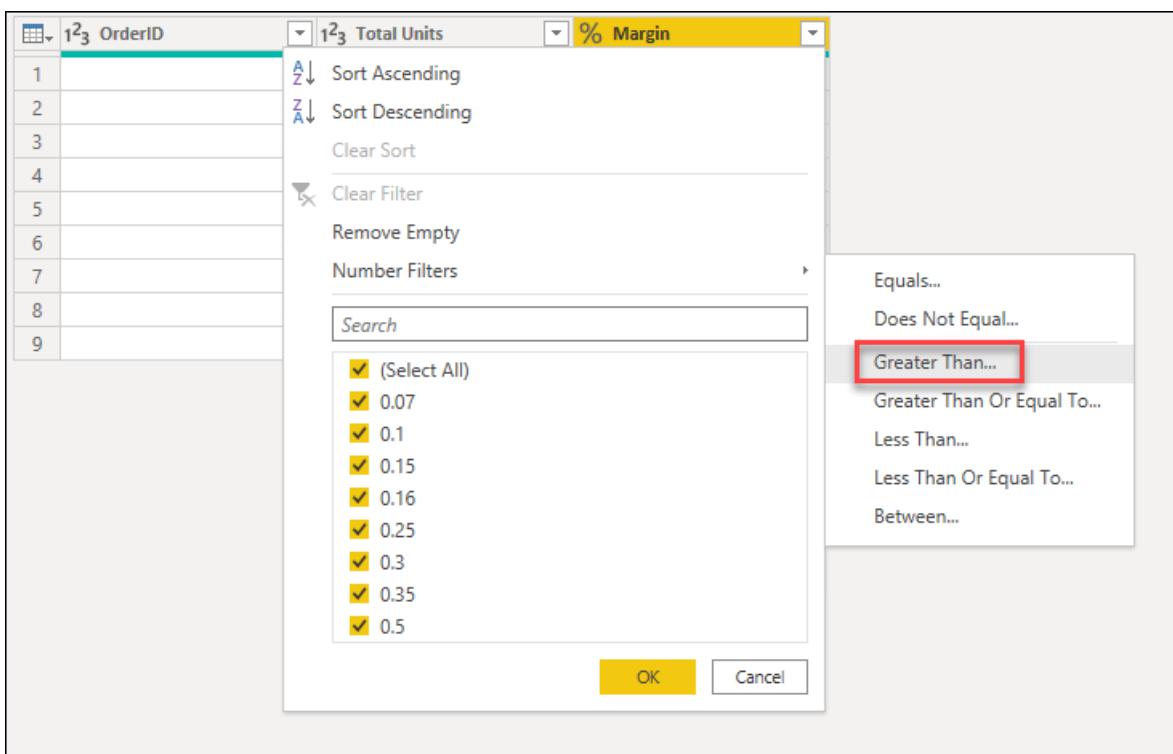
For example purposes, you can see the following **Orders** query with the fields **OrderID**, **Units**, and **Margin**.

OrderID	Total Units	Margin
1	204	10.00%
2	457	7.00%
3	568	15.00%
4	745	25.00%
5	125	30.00%
6	245	50.00%
7	687	16.00%
8	999	16.00%
9	777	35.00%

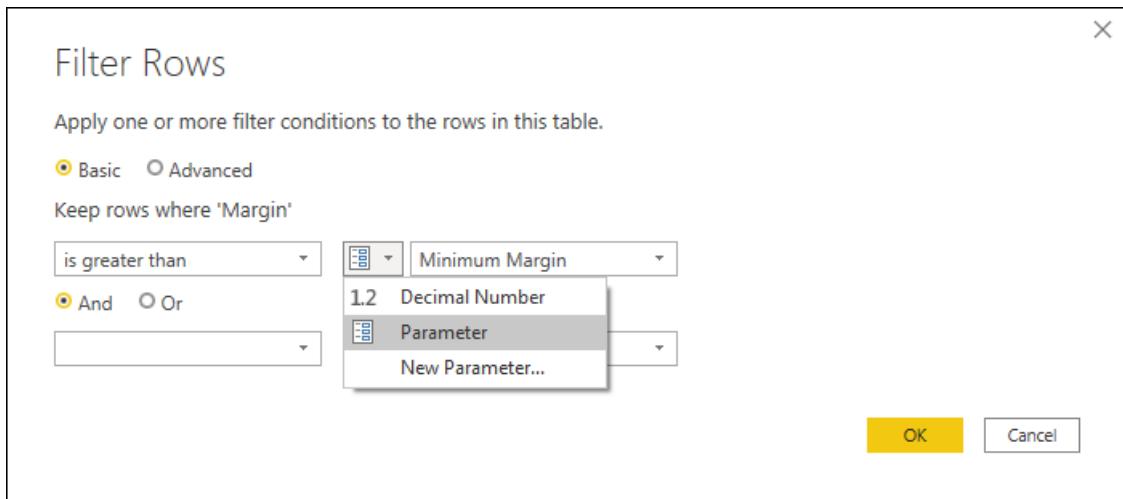
You can create a new parameter with the name **Minimum Margin** with a **Decimal Number** type and a **Current Value** of 0.2, as shown in the next image.



You can go to the Orders query, and in the Margin field select the Greater Than filter option.



In the **Filter Rows** window, you'll see a button with a data type for the field selected. You can select the **Parameter** option from the dropdown menu for this button. From the field selection right next to the data type button, you can select the parameter that you want to pass to this argument. In this case, it's the **Minimum Margin** parameter.



After you select **OK**, you can see that your table has been filtered using the **Current Value** for your parameter.

	OrderID	Total Units	% Margin
1	745	25	25.00%
2	125	30	30.00%
3	245	100	50.00%
4	777	2500	35.00%

If you modify the **Current Value** of your **Minimum Margin** parameter to be 0.3, you can immediately see how your orders query gets updated and shows you only the rows where the **Margin** is above 30%.

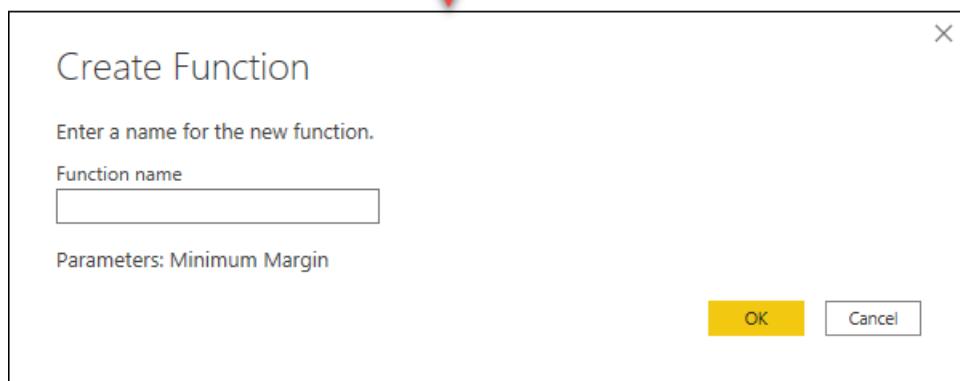
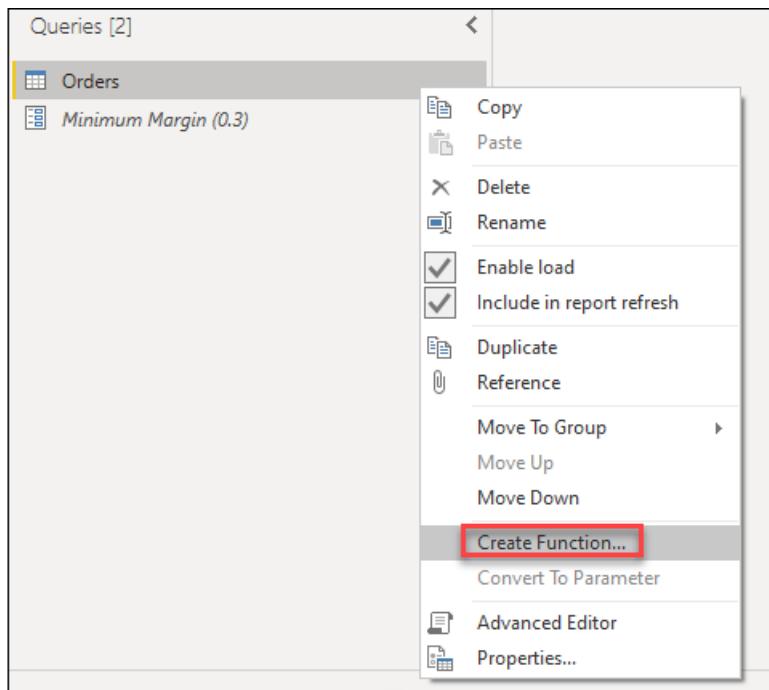
	OrderID	Total Units	% Margin
1	245	100	50.00%
2	777	2500	35.00%

TIP

Multiple transformations in Power Query offer this experience where you can select your parameter from a dropdown. So we recommend that you always look for it and take advantage of what parameters can offer you.

Custom function argument

With Power Query, you can create a custom function from an existing query with a simple click. Following the previous example, you can right-click the **Orders** query and select **Create Function**, which will launch a new **Create Function** window. In this window, you can name your new function and it will tell you the parameters being referenced in your query. These parameters will be used as the parameters for the custom function.



You can name this new function however you want. For demonstration purposes, the name of this new function will be **MyFunction**. After you select **OK**, a new group will be created in the **Queries** pane using the name of your new function. In this group, you'll find the parameters being used for the function, the query that was used to create the function, and the function itself.

The screenshot shows the Power BI Query Editor interface. On the left, the 'Queries [3]' pane lists three items: 'MyFunction [3]', 'Orders', and 'Minimum Margin (0.3)'. The 'MyFunction' item is expanded, showing its definition: `= (#"Minimum Margin" as number) => let`. In the center, the 'Enter Parameter' dialog is open, prompting for a 'Minimum Margin' value. The 'Invoke' button is highlighted. On the right, the 'Query Settings' pane shows the 'Name' field set to 'MyFunction' under the 'PROPERTIES' tab. Under 'APPLIED STEPS', the 'Source' step is selected.

You can test this new function by entering a value, such as 0.4, in the field underneath the **Minimum Margin** label. Then select the **Invoke** button. This will create a new query with the name **Invoked Function**, effectively passing the value 0.4 to be used as the argument for the function and giving you only the rows where the margin is above 40%.

The screenshot shows the Power BI Query Editor after invoking the function. The 'Queries [4]' pane now includes 'Invoked Function'. The main area displays a preview of the data with three columns: OrderID, Total Units, and Margin. The first row shows OrderID 1, Total Units 245, and Margin 50.00%. The status bar at the bottom indicates '3 COLUMNS, 1 ROW' and 'Column profiling based on top 1000 rows'.

OrderID	Total Units	Margin
1	245	50.00%

You can learn more about how to create custom functions from the article [Creating a custom function](#).

Error handling

1/15/2022 • 4 minutes to read • [Edit Online](#)

Similar to how Excel and the DAX language have an `IFERROR` function, Power Query has its own syntax to test and catch errors.

As mentioned in the article on [dealing with errors in Power Query](#), errors can appear either at the step or cell level. This article will focus on how you can catch and manage errors based on our own specific logic.

NOTE

To demonstrate this concept, this article will use an Excel Workbook as its data source. The concepts showcased here apply to all values in Power Query and not only the ones coming from an Excel Workbook.

Applying conditional logic based on errors

The sample data source for this demonstration is an Excel Workbook with the following table:

	A	B	C
1	Account	Standard Rate	Special Rate
2	USA-0004	50	25
3	USA-1524	#NULL!	10
4	USA-4889	35	20
5	PTY-1257	#REF!	12
6	PTY-0507	40	33
7	PTY-1234	#DIV/0!	77

This table from an Excel Workbook has Excel errors such as `#NULL!`, `#REF!`, and `#DIV/0!` in the **Standard Rate** column. When you import this table into the Power Query Editor, the following image shows how it will look.

	ABC 123 Account	ABC 123 Standard Rate	ABC 123 Special Rate
1	USA-0004	50	25
2	USA-1524	[Error]	10
3	USA-4889	35	20
4	PTY-1257	[Error]	12
5	PTY-0507	40	33
6	PTY-1234	[Error]	77

Notice how the errors from the Excel workbook are shown with the `[Error]` value in each of the cells.

In this case, the goal is to create a new **Final Rate** column that will use the values from the **Standard Rate** column. If there are any errors, then it will use the value from the correspondent **Special Rate** column.

Add custom column with `try` **and** `otherwise` **syntax**

To create a new custom column, go to the **Add column** menu and select **Custom column**. In the **Custom column** window, enter the formula `try [Standard Rate] otherwise [Special Rate]`. Name this new column **Final Rate**.

Custom column

Add a column that is computed from other columns or values.

New column name

Final Rate

Custom column formula ⓘ

```
= try [Standard Rate] otherwise [Special Rate]
```

Available column(s)

Account
Standard Rate
Special Rate

Insert column

[Learn more about Power Query formulas](#)

OK

Cancel

The formula above will try to evaluate the **Standard Rate** column and will output its value if no errors are found. If errors are found in the **Standard Rate** column, then the output will be the value defined after the `otherwise` statement, which in this case is the **Special Rate** column.

After adding the correct data types to all of the columns in the table, the following image shows how the final table looks.

	A ^B Account	1 ² 3 Standard Rate	1 ² 3 Special Rate	1 ² 3 Final Rate
1	USA-0004	50	25	50
2	USA-1524	[Error]	10	10
3	USA-4889	35	20	35
4	PTY-1257	[Error]	12	12
5	PTY-0507	40	33	40
6	PTY-1234	[Error]	77	77

Catching an error with `try` and applying custom conditional logic

Using the same sample data source as the previous section, the new goal is to create a new column for the **Final Rate**. If the value from the **Standard Rate** exists, then that value will be used. Otherwise the value from the **Special Rate** column will be used, except for the rows with any `#REF!` error.

NOTE

The sole purpose of excluding the `#REF!` error is for demonstration purposes. With the concepts showcased in this article, you can target any error reasons, messages, or details of your choice.

When you select any of the whitespace next to the error value, you get the details pane at the bottom of the screen. The details pane contains both the error reason, `DataFormat.Error`, and the error message,

Invalid cell value '#REF!' :

The screenshot shows a Power BI interface with a table containing three columns: 'ABC 123 Account', 'ABC 123 Standard Rate', and 'ABC 123 Special Rate'. The second row's 'Standard Rate' cell contains the text '[Error]'. Below the table, a yellow callout box displays the message '(i) DataFormat.Error: Invalid cell value '#REF!'.' with a 'Show details' button.

You can only select one cell at a time, so you can effectively only see the error components of one error value at a time. This is where you'll create a new custom column and use the `try` expression.

Add custom column with `try` syntax

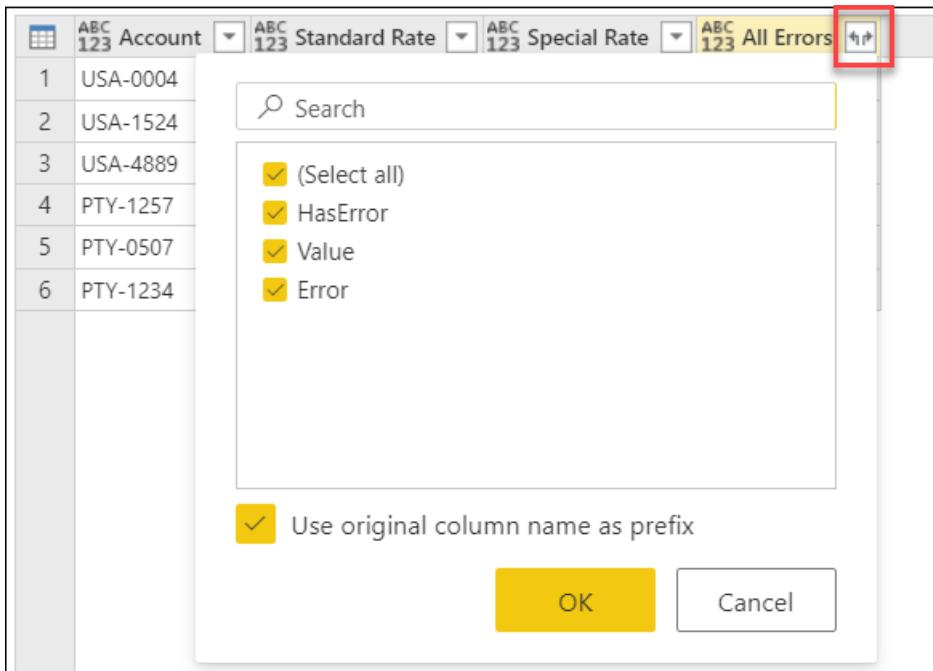
To create a new custom column, go to the **Add column** menu and select **Custom column**. In the **Custom column** window, enter the formula `try [Standard Rate]`. Name this new column **All Errors**.

The screenshot shows the 'Custom column' dialog box. It includes fields for 'New column name' (set to 'All Errors'), 'Custom column formula' (containing the formula `= try [Standard Rate]`), and a 'Available column(s)' list (showing 'Account', 'Standard Rate' (selected), and 'Special Rate'). At the bottom are 'OK' and 'Cancel' buttons, and a link to 'Learn more about Power Query formulas'.

The `try` expression converts values and errors into a record value that indicates whether the `try` expression handled an error or not, as well as the proper value or the error record.

	ABC 123 Account	ABC 123 Standard Rate	ABC 123 Special Rate	ABC 123 All Errors
1	USA-0004	50	25	[Record]
2	USA-1524	[Error]	10	[Record]
3	USA-4889	35	20	[Record]
4	PTY-1257	[Error]	12	[Record]
5	PTY-0507	40	33	[Record]
6	PTY-1234	[Error]	77	[Record]

You can expand this newly created column with record values and look at the available fields to be expanded by selecting the icon next to the column header.



This operation will expose three new fields:

- **All Errors.HasError**—displays whether the value from the **Standard Rate** column had an error or not.
- **All Errors.Value**—if the value from the **Standard Rate** column had no error, this column will display the value from the **Standard Rate** column. For values with errors this field won't be available, and during the expand operation this column will have `null` values.
- **All Errors.Error**—if the value from the **Standard Rate** column had an error, this column will display the error record for the value from the **Standard Rate** column. For values with no errors, this field won't be available, and during the expand operation this column will have `null` values.

	ABC 123 Account	ABC 123 Standard Rate	ABC 123 Special Rate	ABC 123 All Errors.HasError	ABC 123 All Errors.Value	ABC 123 All Errors.Error
1	USA-0004	50	25	FALSE	50	null
2	USA-1524	[Error]		10	TRUE	null [Record]
3	USA-4889		35	20	FALSE	35 null
4	PTY-1257	[Error]		12	TRUE	null [Record]
5	PTY-0507		40	33	FALSE	40 null
6	PTY-1234	[Error]		77	TRUE	null [Record]

Reason	DataFormat.Error
Message	Invalid cell value '#N...'
Detail	null

For further investigation, you can expand the **All Errors.Error** column to get the three components of the error record:

- Error reason
- Error message
- Error detail

After doing the expand operation, the **All Errors.Error.Message** field displays the specific error message that tells you exactly what Excel error each cell has. The error message is derived from the **Error Message** field of the error record.

	ABC 123 Account	ABC 123 Standard Rate	ABC 123 Special Rate	ABC 123 All Errors.HasError	ABC 123 All Errors.Value	ABC 123 All Errors.Error.Reason	ABC 123 All Errors.Error.Message	ABC 123 All Errors.Error.Detail
1	USA-0004	50	25	FALSE	50	null	null	null
2	USA-1524	[Error]		10	TRUE	null	DataFormat.Error	Invalid cell value '#NULL!'. null
3	USA-4889		35	20	FALSE	35	null	null
4	PTY-1257	[Error]		12	TRUE	null	DataFormat.Error	Invalid cell value '#REF!'. null
5	PTY-0507		40	33	FALSE	40	null	null
6	PTY-1234	[Error]		77	TRUE	null	DataFormat.Error	Invalid cell value '#DIV/0!'. null

Add a conditional column

Now with each error message in a new column, you can create a new conditional column with the name **Final Rate** and the following clauses:

- If the value in the **All Errors.Errors.Message** column equals `null`, then the output will be the value from the **Standard Rate** column.
- Else, if the value in the **All Errors.Errors.Message** column equals `Invalid cell value '#REF!'.`, then the output will be the value from the **Special Rate** column.
- Else, `null`.

Add conditional column

Add a conditional column that is computed from the other columns or values.

New column name
Final Rate

Column name	Operator	Value ⓘ	Output ⓘ
If	equals	ABC 123	Then Standard Rate
Else if	does not equal	ABC 123	Then Special Rate
Invalid cell value '#REF!'. Else ⓘ ABC 123 null			
Add clause			
OK Cancel			

After keeping only the **Account**, **Standard Rate**, **Special Rate**, and **Final Rate** columns, and adding the correct data type for each column, the following image shows what the final table looks like.

	A C Account	1 2 3 Standard Rate	1 2 3 Special Rate	1 2 3 Final Rate
1	USA-0004	50	25	50
2	USA-1524	[Error]	10	10
3	USA-4889	35	20	35
4	PTY-1257	[Error]	12	null
5	PTY-0507	40	33	40
6	PTY-1234	[Error]	77	77

More resources

- [Understanding and working with errors in Power Query](#)
- [Add a Custom column in Power Query](#)
- [Add a Conditional column in Power Query](#)

Import data from a database using native database query

1/15/2022 • 4 minutes to read • [Edit Online](#)

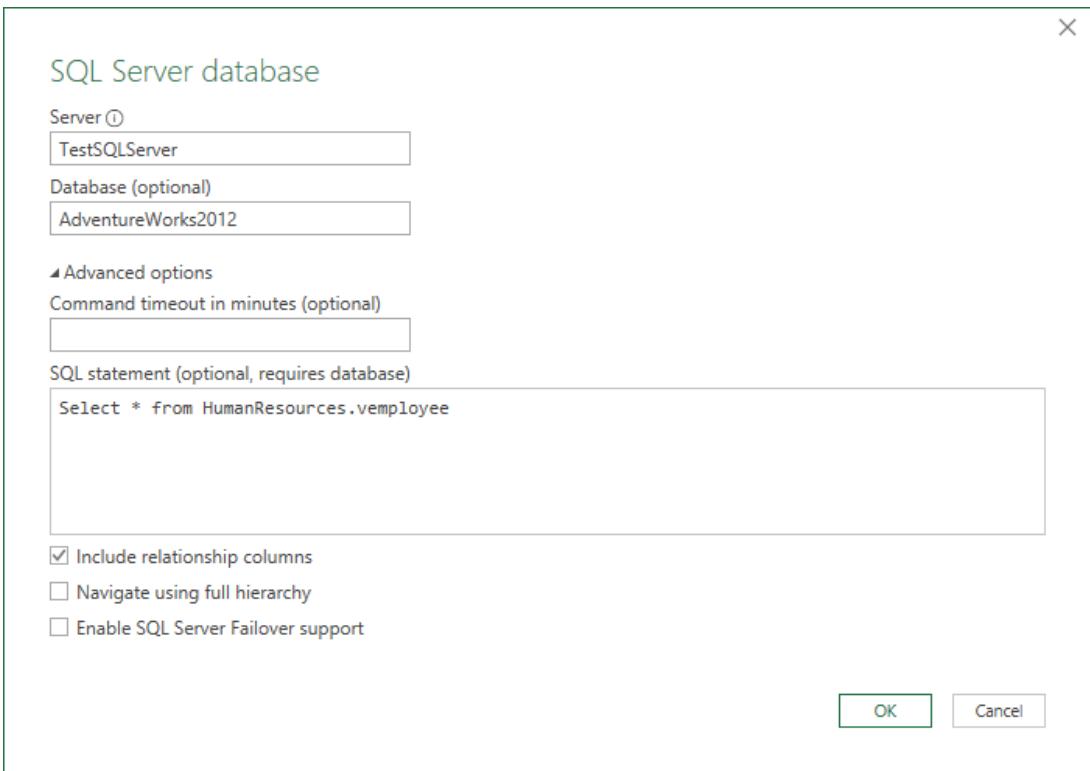
Power Query gives you the flexibility to import data from wide variety of databases that it supports. It can run native database queries, which can save you the time it takes to build queries using the Power Query interface. This feature is especially useful for using complex queries that already exist—and that you might not want to or know how to rebuild using the Power Query interface.

NOTE

One intent of native database queries is to be non-side effecting. However, Power Query does not guarantee that the query will not affect the database. If you run a native database query written by another user, you will be prompted to ensure that you're aware of the queries that will be evaluated with your credentials. For more information, see [Native database query security](#).

Power Query enables you to specify your native database query in a text box under **Advanced options** when connecting to a database. In the example below, you'll import data from a SQL Server database using a native database query entered in the **SQL statement** text box. The procedure is similar in all other databases with native database query that Power Query supports.

1. Connect to a SQL Server database using Power Query. Select the **SQL Server database** option in the connector selection.
2. In the **SQL Server database** popup window:
 - a. Specify the **Server** and **Database** where you want to import data from using native database query.
 - b. Under **Advanced options**, select the **SQL statement** field and paste or enter your native database query, then select **OK**.



3. If this is the first time you're connecting to this server, you'll see a prompt to select the authentication mode to connect to the database. Select an appropriate authentication mode, and continue.

NOTE

If you don't have access to the data source (both Server and Database), you'll see a prompt to request access to the server and database (if access-request information is specified in Power BI for the data source).

4. If the connection is established, the result data is returned in the Power Query Editor.

Shape the data as you prefer, then select **Apply & Close** to save the changes and import the data.

Connectors that support native database queries

The following Power Query connectors support native database queries.

CONNECTOR	TYPE OF NATIVE DATABASE QUERY
Amazon Redshift	SQL statement
Azure Analysis Services database	MDX or DAX query
Azure Database for PostgreSQL	SQL statement
Azure Cosmos DB	SQL statement
Azure SQL Data Warehouse	SQL statement
Azure SQL database	SQL statement
DataWorld.Dataset	dwSQL

CONNECTOR	TYPE OF NATIVE DATABASE QUERY
Dataverse	SQL statement
Essbase	MDX statement
FHIR	FHIR Search
IBM Db2 database	SQL statement
IBM Informix database (Beta)	SQL statement
MySQL database	SQL statement
ODBC	SQL statement
OLE DB	SQL statement
Oracle database	SQL statement
PostgreSQL	SQL statement
SAP HANA database	SQL statement
Snowflake	SQL statement
SQL Server Analysis Services database	MDX or DAX query
SQL Server database	SQL statement
TIBCO(R) Data Virtualization (Beta)	SQL statement
Vena (Beta)	Model Query (MQL)

Limitations and issues

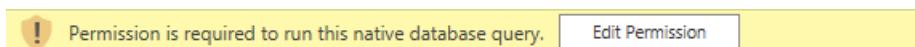
Before using native database query, you should be aware of the limitations and issues that you may meet.

Query folding

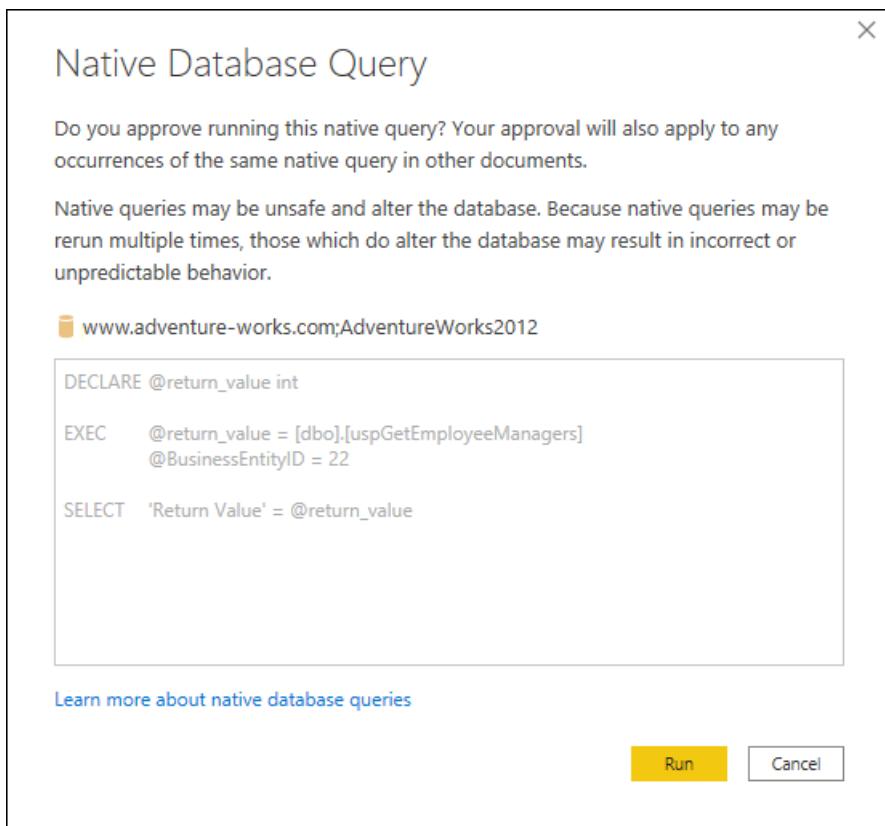
Query folding while using a native database query is limited to the PostgreSQL connector only. No other connectors support query folding if you use a native database query. Also, for folding to work in the PostgreSQL connector, the native database query you enter has to work as a subquery.

Native database query security

Sometimes, when you attempt to use a query created by another user or through the Advanced Editor or formula bar—essentially any other path outside of the connector dialogs where the native query input box is shown—you may get a message that says:



If you see this message, select **Edit Permission**. This selection will open the **Native Database Query** dialog box. You'll be given an opportunity to either run the native database query, or cancel the query.



By default, if you run a native database query outside of the connector dialogs, you'll be prompted each time you run a different query text to ensure that the query text that will be executed is approved by you.

NOTE

Native database queries that you insert in your get data operation won't ask you whether you want to run the query or not. They'll just run.

You can turn off the native database query security messages if the native database query is run in either Power BI Desktop or Excel. To turn off the security messages:

1. If you're using Power BI Desktop, under the **File** tab, select **Options and settings > Options**.
If you're using Excel, under the **Data** tab, select **Get Data > Query Options**.
2. Under **Global** settings, select **Security**.
3. Clear **Require user approval for new native database queries**.
4. Select **OK**.

You can also revoke the approval of any native database queries that you've previously approved for a given data source in either Power BI Desktop or Excel. To revoke the approval:

1. If you're using Power BI Desktop, under the **File** tab, select **Options and settings > Data source settings**.
If you're using Excel, under the **Data** tab, select **Get Data > Data Source Settings**.
2. In the **Data source settings** dialog box, select **Global permissions**. Then select the data source containing the native database queries whose approval you want to revoke.
3. Select **Edit permissions**.
4. In the **Edit permissions** dialog box, under **Native Database Queries**, select **Revoke Approvals**.

Data source settings

Manage settings for data source

Data sources in current file

[Search data source settings](#)

 testsqserver

 testsqserver;Adventure

[Edit Permissions...](#)

[Clear Perm](#)

Edit Permissions

 testsqserver;AdventureWorks2017

Credentials

Type: Not Specified

[Edit...](#)

[Delete](#)

Encryption

Encrypt connections

Privacy Level

[None](#)

Native Database Queries

You have approved 1 Native Query for this source

[Revoke Approvals](#)

[OK](#)

[Cancel](#)

[Close](#)

Create Power Microsoft Platform dataflows from queries in Microsoft Excel (Preview)

1/15/2022 • 2 minutes to read • [Edit Online](#)

[This topic is pre-release documentation and is subject to change.]

You can create Microsoft Power Platform dataflows from queries in Microsoft Excel workbooks to take advantage of cloud-powered dataflows refreshing and processing the data at regular intervals instead of performing these operations manually in Excel.

This article walks you through how to export queries from Excel into a Power Query template that can then be imported into Power Platform dataflow to create a dataflow.

NOTE

The preview feature for creating Power Query templates from queries feature is only available to Office Insiders. For more information on the Office insider program, see [Office Insider](#).

Overview

Working with large datasets or long-running queries can be cumbersome every time you have to manually trigger a data refresh in Excel because it takes resources from your computer to do this, and you have to wait until the computation is done to get the latest data. Moving these data operations into a Power Platform dataflow is an effective way to free up your computer's resources and to have the latest data easily available for you to consume in Excel.

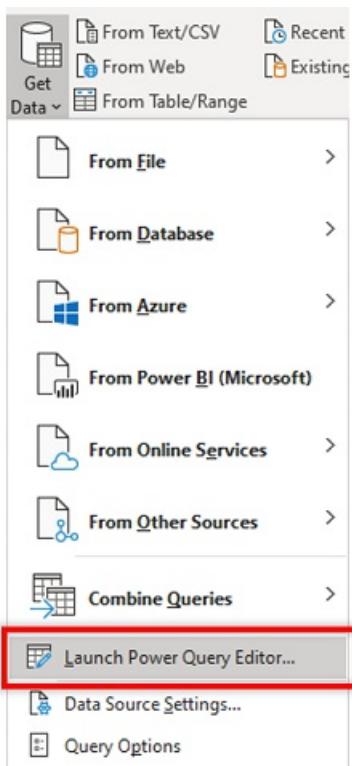
It only takes two quick steps to do this:

1. Exporting queries in Excel to a Power Query template
2. Creating a Power Platform dataflow from the Power Query template

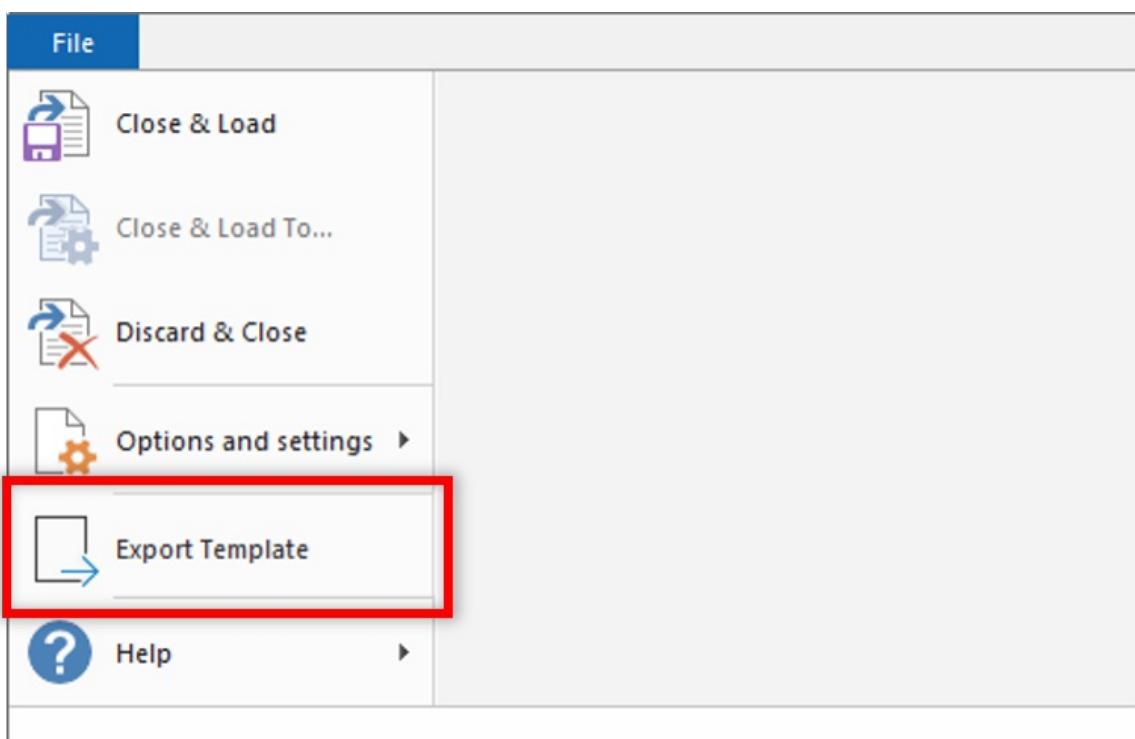
Exporting queries in Excel to a Power Query template

The first step is to create a Power Query template with your queries in Excel.

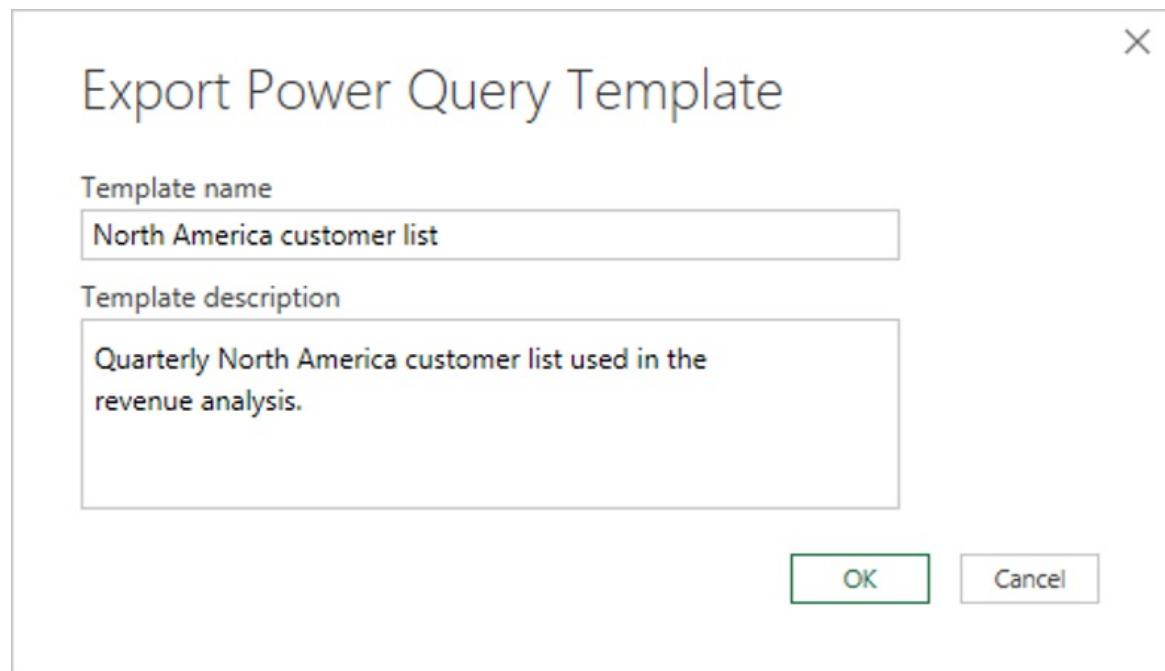
1. Start the Power Query editor from **Data tab > Get Data > Launch Power Query Editor**.



2. Once Power Query loads, select File > Export Template.

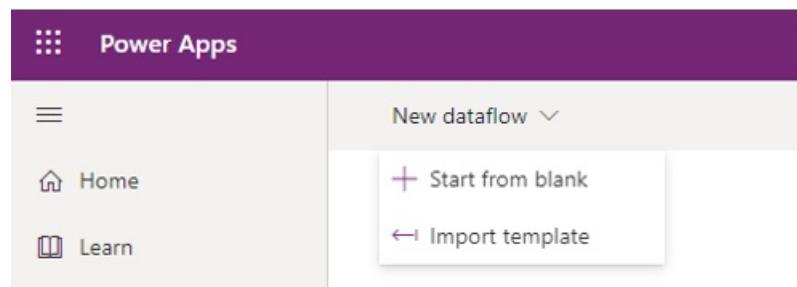


3. The template requires basic information such as a name and a description before it can be saved locally on your computer.



Creating a Power Platform dataflow from the Power Query template

1. Sign in to [Power Apps](#).
2. In the left navigation pane, select Data > Dataflows.
3. From the toolbar, select New dataflow > Import template.



4. Select the Power Query template you created earlier. The dataflow name will prepopulate with the template name provided. Once you're done with the dataflow creation screen, select **Next** to see your queries from Excel in the query editor.
5. From this point, go through the normal dataflow creation and configuration process so you can further transform your data, set refresh schedules on the dataflow, and any other dataflow operation possible. For more information on how to configure and create Power Platform dataflows, see [Create and use dataflows](#).

See also

[Create and use dataflows in Power Apps](#)

Optimize Power Query when expanding table columns

1/15/2022 • 3 minutes to read • [Edit Online](#)

The simplicity and ease of use that allows Power BI users to quickly gather data and generate interesting and powerful reports to make intelligent business decisions also allows users to easily generate poorly performing queries. This often occurs when there are two tables that are related in the way a foreign key relates SQL tables or SharePoint lists. (For the record, this issue isn't specific to SQL or SharePoint, and occurs in many backend data extraction scenarios, especially where schema is fluid and customizable.) There's also nothing inherently wrong with storing data in separate tables that share a common key—in fact this is a fundamental tenet of database design and normalization. But it does imply a better way to expand the relationship.

Consider the following example of a SharePoint customer list.

Primary Customer List				
Title	Location	Location:Zip	Location:Anchor City	Location:State
Topographic Oceanus	90125	90125	Big Sky	LM
Tamarack	59937	59937	Whitefish	MT
Skunkworks Ltd.	80122	80122	Littleton	CO
Contoso	98052	98052	Redmond	WA
4500 The Team	21412	21412	Annapolis	MD

And the following location list it refers to.

Secondary Location List		
State	Anchor City	Zip
WA	Redmond	98052
CA	Los Angeles	90189
IA	Council Bluffs	51501
MT	Bozeman	59715
MT	Whitefish	59937
MD	Annapolis	21412

When first connecting to the list, the location shows up as a record.

The screenshot shows the Power BI Query Editor interface. On the left, there's a 'Queries [1]' pane with a single item named 'Primary Customer List'. To the right, the main area displays a table with three columns: 'ABC 123 Title' (containing values like 'Topographic Oceanus', 'Tamarack', etc.), 'ABC 123 LocationId' (containing values like '11', '6', etc.), and 'Location' (containing values like 'Record', 'Record', etc.). A tooltip above the table indicates the formula: `= Table.SelectColumns(#"Renamed Columns", {"Title", "LocationId", "Location"})`.

This top-level data is gathered through a single HTTP call to the SharePoint API (ignoring the metadata call), which you can see in any web debugger.

1290	200	HTTP	Tunnel to	microsoft.sharepoint.com:443
1291	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/\$metadata
1294	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items

When you expand the record, you see the fields joined from the secondary table.

The screenshot shows the Power BI Query Editor interface. The 'Primary Customer List' query now includes an expanded column 'Location'. The table has four columns: 'ABC 123 Title', 'ABC 123 Location.Zip' (containing values like '90125', '59937', etc.), 'ABC 123 Location.Anchor City' (containing values like 'Big Sky', 'Whitefish', etc.), and 'ABC 123 Location.State' (containing values like 'LM', 'MT', etc.). A tooltip above the table indicates the formula: `= Table.ExpandRecordColumn(#"Removed Other Columns", "Location", {"Zip", "Anchor City", "State"})`.

When expanding related rows from one table to another, the default behavior of Power BI is to generate a call to `Table.ExpandTableColumn`. You can see this in the generated formula field. Unfortunately, this method generates an individual call to the second table for every row in the first table.

#	Result	Protocol	Host	URL
1286	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/web/lists?\$select=Id,Title,Items
1291	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/\$metadata
1294	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items
1297	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items
1299	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(1)/Location
1300	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(2)/Location
1301	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(3)/Location
1302	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(4)/Location
1303	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(5)/Location
1306	200	HTTPS	microsoft.sharepoint.com	/teams/bas/_api/Web/Lists(guid'00000000-0000-0000-0000-000000000000')/Items(6)/Location

This increases the number of HTTP calls by one for each row in the primary list. This may not seem like a lot in the above example of five or six rows, but in production systems where SharePoint lists reach hundreds of thousands of rows, this can cause a significant experience degradation.

When queries reach this bottleneck, the best mitigation is to avoid the call-per-row behavior by using a classic table join. This ensures that there will be only one call to retrieve the second table, and the rest of the expansion can occur in memory using the common key between the two tables. The performance difference can be massive in some cases.

First, start with the original table, noting the column you want to expand, and ensuring you have the ID of the item so that you can match it. Typically the foreign key is named similar to the display name of the column with `Id` appended. In this example, it's `LocationId`.

Queries [1]

	Title	LocationId	Location
1	Topographic Oceanus	11	Record
2	Tamarack	6	Record
3	Skunkworks Ltd.	8	Record
4	Contoso	2	Record
5	4500 The Team	7	Record

Second, load the secondary table, making sure to include the **Id**, which is the foreign key. Right-click on the Queries panel to create a new query.

Queries [2]

	Id	Zip	Anchor City	State
1		98052	Redmond	WA
2		90189	Los Angeles	CA
3		51501	Council Bluffs	IA
4		59715	Bozeman	MT
5		59937	Whitefish	MT
6		21412	Annapolis	MD
7		80122	Littleton	CO
8		02112	Boston	MA
9		69105	North Platte	NE
10		90125	Big Sky	LM

Finally, join the two tables using the respective column names that match. You can typically find this field by first expanding the column, then looking for the matching columns in the preview.

	Title	LocationId	Location.Id	Location.Zip	Location.Anchor City	Location.State
1	Topographic Oceanus		11	11 90125	Big Sky	LM
2	Tamarack		6	6 59937	Whitefish	MT
3	Skunkworks Ltd.		8	8 80122	Littleton	CO
4	Contoso		2	2 98052	Redmond	WA
5	4500 The Team		7	7 21412	Annapolis	MD

In this example, you can see that **LocationId** in the primary list matches **Id** in the secondary list. The UI renames this to **Location.Id** to make the column name unique. Now let's use this information to merge the tables.

By right-clicking on the query panel and selecting **New Query > Combine > Merge Queries as New**, you see a friendly UI to help you combine these two queries.

Queries [2]

	Id	Zip	Anchor City
1		98052	Redmond
2		90189	Los Angeles
3		51501	Council Bluffs

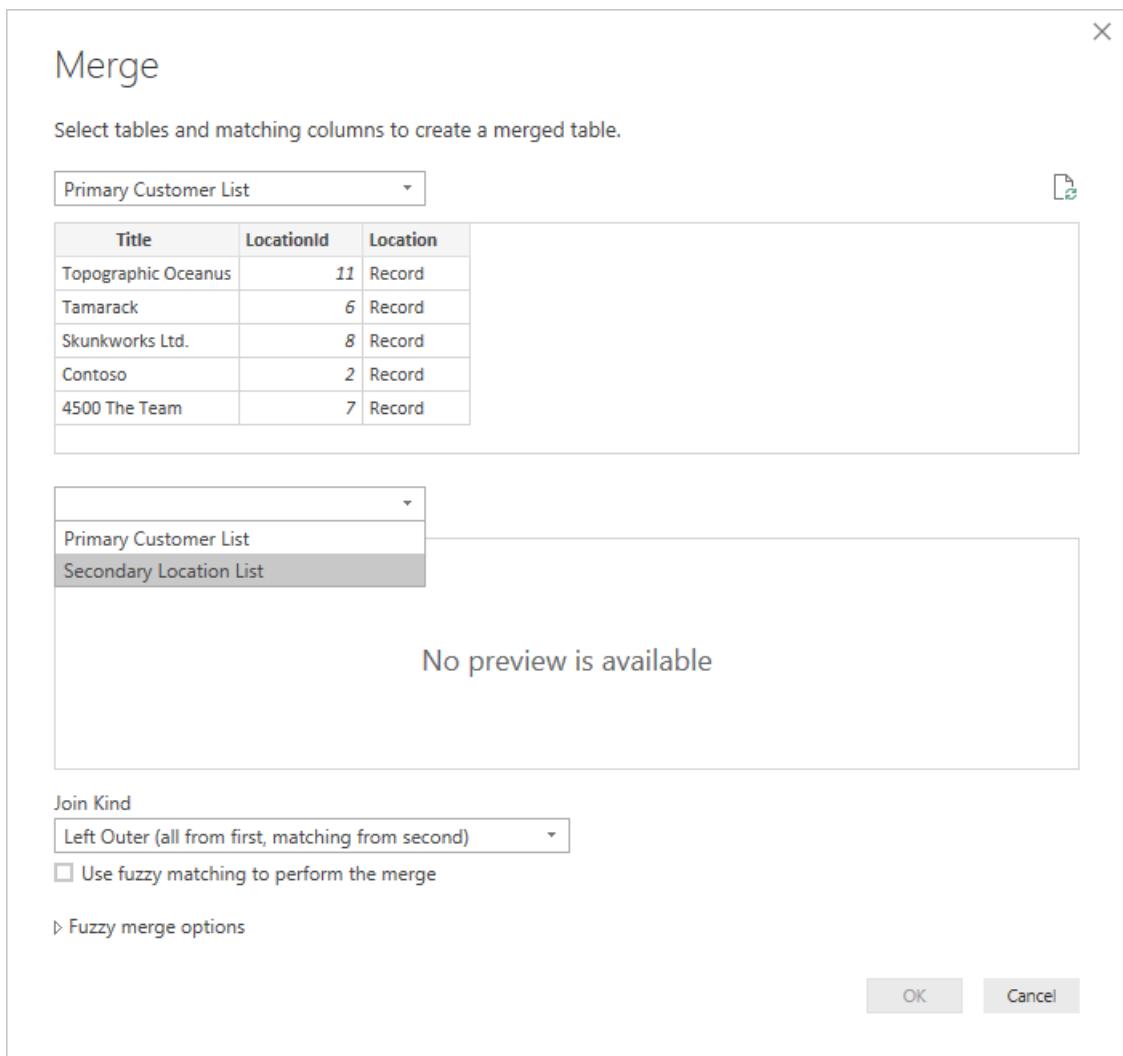
Paste

- New Query
- New Parameter...
- New Group...
- Expand All
- Collapse All

Merge Queries as New

Recent Sources

Select each table from the drop-down to see a preview of the query.



Once you've selected both tables, select the column that joins the tables logically (in this example, it's **LocationId** from the primary table and **Id** from the secondary table). The dialog will instruct you how many of the rows match using that foreign key. You'll likely want to use the default join kind (left outer) for this kind of data.

Merge

Select tables and matching columns to create a merged table.

Primary Customer List			
Title	LocationId	Location	
Topographic Oceanus	11	Record	
Tamarack	6	Record	
Skunkworks Ltd.	8	Record	
Contoso	2	Record	
4500 The Team	7	Record	

Secondary Location List			
Id	Zip	Anchor City	State
2	98052	Redmond	WA
3	90189	Los Angeles	CA
4	51501	Council Bluffs	IA
5	59715	Bozeman	MT
6	59937	Whitefish	MT

Join Kind

Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

► Fuzzy merge options

The selection matches 6 of 6 rows from the first table.

Select OK and you'll see a new query, which is the result of the join. Expanding the record now doesn't imply additional calls to the backend.

Queries [3]	X	Y	f(x)	= Table.RemoveColumns(#"Expanded Secondary Location List", {"LocationId"})
Primary Customer List				ABC 123 Title ABC 123 Secondary Location List.Zip ABC 123 Secondary Location List.Anchor City ABC 123 Secondary Location List.State
Secondary Location List				ABC 123 Title ABC 123 Secondary Location List.Zip ABC 123 Secondary Location List.Anchor City ABC 123 Secondary Location List.State
Merge1				ABC 123 Title ABC 123 Secondary Location List.Zip ABC 123 Secondary Location List.Anchor City ABC 123 Secondary Location List.State

Refreshing this data will result in only two calls to SharePoint—one for the primary list, and one for the secondary list. The join will be performed in memory, significantly reducing the number of calls to SharePoint.

This approach can be used for any two tables in PowerQuery that have a matching foreign key.

NOTE

SharePoint user lists and taxonomy are also accessible as tables, and can be joined in exactly the way described above, provided the user has adequate privileges to access these lists.

Enabling Microsoft Edge (Chromium) for OAuth authentication in Power BI Desktop

1/15/2022 • 2 minutes to read • [Edit Online](#)

If you're using OAuth authentication to connect to your data, the OAuth dialog in Power Query uses the Microsoft Internet Explorer 11 embedded control browser. However, certain web services, such as QuickBooks Online, Salesforce Reports, and Salesforce Objects no longer support Internet Explorer 11.

October 2021 Power BI release

NOTE

If you are using an earlier release of Power BI, go to [December 2020 Power BI Release](#).

As of October of 2021, Power BI Desktop now uses [Microsoft Edge WebView2](#), by default, for OAuth authentication for all connectors. However, you can change the default behavior using environment variables.

- To disable the use of WebView2 for specific connectors, set `PQ_ExtendEdgeChromiumOAuthDenyList` with the name(s) of the connector(s) you want to disable. Multiple connectors are separated by semicolons.

```
setx PQ_ExtendEdgeChromiumOAuthDenyList MyExtension1;MyExtension2
```

- To disable the use of WebView2, set `PQ_DisableEdgeChromiumOAuth` to true.

```
setx PQ_DisableEdgeChromiumOAuth true
```

December 2020 Power BI release

As of December of 2020, Power BI Desktop uses [Microsoft Edge WebView2](#) for OAuth authentication with certain connectors. These connectors are:

- GitHub
- QuickBooks Online
- Salesforce Reports
- Salesforce Objects
- Smartsheet
- Twilio
- Zendesk

On your Power BI Desktop machine, you can get WebView2 control either by installing the new Edge (Chromium) browser (at least beta) from <https://www.microsoftedgeinsider.com/download>, or by installing the [WebView2 redist package](#).

All other connectors will use Internet Explorer 11 by default unless the settings are overridden using environment variables.

- To enable WebView2 for all connectors, set `PQ_EdgeChromiumOAuthAllowListAll` to true:

```
setx PQ_EdgeChromiumOAuthAllowListAll    true
```

- To enable WebView2 for specific connectors, set `PQ_ExtendEdgeChromiumOAuthAllowList` with the name(s) of the connector(s) you want to enable. Multiple connectors are separated by semicolons.

```
setx PQ_ExtendEdgeChromiumOAuthAllowList    MyExtension1;MyExtension2
```

- To disable the use of WebView2, set `PQ_DisableEdgeChromiumOAuth` to true.

```
setx PQ_DisableEdgeChromiumOAuth    true
```

Connectors in Power Query

1/15/2022 • 7 minutes to read • [Edit Online](#)

The following table contains a list of all the connectors currently available for Power Query. For those connectors that have a reference page in this document, a link is provided under the connector icon and name.

A checkmark indicates the connector is currently supported in the listed service; an X indicates that the connector is not currently supported in the listed service.

The connectors are listed in alphabetical order in separate tables for each letter in the alphabet. Use the **In this article** list on the right side of this article to go to any of the alphabetized tables.

NOTE

The Excel column in the following table indicates all connectors that are available on at least one version of Excel. However, not all Excel versions support all of these indicated Power Query connectors. For a complete list of the Power Query connectors supported by all versions of Excel, go to [Power Query data sources in Excel versions](#).

A

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Access Database By Microsoft	✓	✓	✓	✓	✓	✓
 Active Directory By Microsoft	✓	✓	✓	✗	✗	✓
 Acterys (Beta) By Acterys	✗	✓	✗	✗	✗	✗
 Actian (Beta) By Actian	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Adobe Analytics By Microsoft	✗	✓	✗	✗	✗	✗
 Amazon Athena By Amazon	✗	✓	✗	✗	✗	✗
 Amazon Redshift By Microsoft	✗	✓	✓	✓	✓	✗
 Anaplan By Anaplan	✗	✓	✗	✗	✗	✗
 appFigures (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Asana By Asana	✗	✓	✗	✗	✗	✗
 Assemble Views By Autodesk	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 AtScale cubes (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Automation Anywhere By Automation Anywhere	✗	✓	✗	✗	✗	✗
 Automy Data Analytics (Beta) By ACEROALTY	✗	✓	✗	✗	✗	✗
 Azure Analysis Services database By Microsoft	✗	✓	✗	✓	✓	✗
 Azure Blob Storage By Microsoft	✓	✓	✓	✓	✓	✓
 Azure CosmosDB v1 (Beta) By Microsoft	✗	✓	✗	✗	✗	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Azure CosmosDB v2 (Beta) By Microsoft	✗	✓	✗	✗	✗	✓
 Azure Cost Management By Microsoft	✗	✓	✗	✗	✗	✗
 Azure Databricks By Databricks	✗	✓	✗	✗	✗	✗
 Azure Data Explorer (Beta) By Microsoft	✓	✓	✓	✓	✓	✗
 Azure Data Lake Storage Gen1 By Microsoft	✓	✓	✗	✗	✗	✓
 Azure Data Lake Storage Gen2 (Beta) By Microsoft	✗	✓	✓	✓	✓	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Azure DevOps (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Azure DevOps Server (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Azure HDInsight (HDFS) By Microsoft	✓	✓	✗	✗	✗	✓
 Azure HDInsight Spark By Microsoft	✗	✓	✓	✓	✓	✓
 Azure Synapse Analytics (SQL DW) By Microsoft	✓	✓	✓	✓	✓	✓
 Azure Synapse Analytics workspace (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Azure SQL database By Microsoft	✓	✓	✓	✓	✓	✓
 Azure Table Storage By Microsoft	✓	✓	✓	✓	✓	✓
 Azure Time Series Insights (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

B

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 BI Connector By Guidanz	✗	✓	✗	✗	✗	✗
 BI360 By Solver Global	✗	✓	✗	✗	✗	✗
 Bloomberg Data and Analytics By Bloomberg	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 BQE Core (Beta) By BQE	✗	✓	✗	✗	✗	✗

C

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Cognite Data Fusion (Beta) By Cognite	✗	✓	✗	✗	✗	✗
 Cherwell (Beta) By Cherwell	✗	✓	✗	✗	✗	✗
 Common Data Service (legacy) By Microsoft	✗	✓	✗	✗	✗	✗

D

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Data.World - Get Dataset (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Data Virtuality (Beta) By Data Virtuality	✗	✓	✗	✗	✗	✗
 Dataverse By Microsoft	✗	✓	✓	✓	✓	✗
 Delta Sharing (Beta) By Databricks	✗	✓	✗	✗	✗	✗
 Denodo By Denodo	✗	✓	✗	✗	✗	✗
 Dremio By Dremio	✗	✓	✗	✗	✗	✗
 Dynamics 365 (online) By Microsoft	✓	✓	✗	✗	✗	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Dynamics 365 Business Central By Microsoft	✗	✓	✗	✗	✗	✗
 Dynamics 365 Business Central (on- premises) By Microsoft	✗	✓	✗	✗	✗	✗
 Dynamics 365 Customer Insights (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Dynamics NAV By Microsoft	✗	✓	✗	✗	✗	✗

E

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 eWay-CRM By eWay- CRM	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Emigo Data Source By Sagra	✗	✓	✗	✗	✗	✗
 Entersoft Business Suite (Beta) By Entersoft	✗	✓	✗	✗	✗	✗
 EQuIS (Beta) By EarthSoft	✗	✓	✗	✗	✗	✗
 Essbase By Microsoft	✗	✓	✗	✗	✗	✗
 Exasol By Exasol	✗	✓	✗	✗	✗	✗
 Excel By Microsoft	✓	✓	✓	✓ ¹	✓	✓

¹ Available in [dataflows for Microsoft Teams](#).

F

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 FactSet Analytics (Beta) By FactSet	✗	✓	✗	✗	✗	✗
 FHIR By Microsoft	✗	✓	✗	✓	✓	✗
 Folder By Microsoft	✓	✓	✓	✓	✓	✓

G

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Github (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Google Analytics By Microsoft	✗	✓	✗	✓	✗	✗
 Google BigQuery By Microsoft	✗	✓	✓	✓	✓	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Google Sheets (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

H

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Hadoop File (HDFS) By Microsoft	✓	✓	✗	✗	✗	✗
 HDInsight Interactive Query By Microsoft	✗	✓	✗	✗	✗	✗
 Hexagon PPM Smart API By Hexagon PPM	✗	✓	✗	✗	✗	✗
 Hive LLAP By Microsoft	✗	✓	✗	✗	✗	✗

|

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 IBM DB2 database By Microsoft	✓	✓	✓	✓	✓	✗
 IBM Informix database (Beta) By Microsoft	✗	✓	✗	✗	✗	✓
 IBM Netezza By Microsoft	✗	✓	✗	✗	✗	✗
 Impala By Microsoft	✗	✓	✗	✓	✓	✗
 Indexima (Beta) By Indexima	✗	✓	✗	✗	✗	✗
 Industrial App Store By Intelligent Plant	✗	✓	✗	✗	✗	✗
 Information Grid (Beta) By Luminis	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 InterSystems IRIS (Beta) By Intersystems	✗	✓	✗	✗	✗	✗
 Intune Data Warehouse (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

J

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Jamf Pro (Beta) By Jamf	✗	✓	✗	✗	✗	✗
 Jethro (Beta) By JethroData	✗	✓	✗	✗	✗	✗
 JSON By Microsoft	✓	✓	✓	✓ ¹	✓	✓

¹ Available in [dataflows for Microsoft Teams](#).

K

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Kognitwin (Beta) By Kongsberg	✗	✓	✗	✗	✗	✗
 Kyligence Kyligence By Kyligence	✗	✓	✗	✗	✗	✗

L

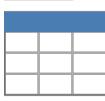
CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Linkar PICK Style/MultiValue Databases (Beta) By Kosday Solutions	✗	✓	✗	✗	✗	✗
 LinkedIn Sales Navigator (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

M

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Marketo (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 MarkLogic By MarkLogic	✗	✓	✗	✗	✗	✗
 MariaDB By MariaDB	✗	✓	✗	✗	✗	✗
 Microsoft Azure Consumption Insights (Beta) (Deprecated) By Microsoft	✗	✓	✗	✗	✗	✗
 Microsoft Exchange By Microsoft	✓	✓	✗	✗	✗	✓
 Microsoft Exchange Online By Microsoft	✓	✓	✓	✓	✓	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Microsoft Graph Security (Deprecated) By Microsoft	✗	✗	✗	✗	✗	✗
 MicroStrategy for Power BI By MicroStrategy	✗	✓	✗	✗	✗	✗
 Mixpanel (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 MySQL database By Microsoft	✓	✓	✓	✓	✓	✓

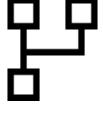
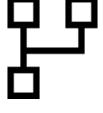
CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 OData Feed By Microsoft	✓	✓	✓	✓ ¹	✓	✓
 ODBC By Microsoft	✓	✓	✓	✓	✓	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 OLE DB By Microsoft	✓	✓	✗	✗	✗	✓
 Oracle database By Microsoft	✓	✓	✓	✓	✓	✓

¹ Available in [dataflows for Microsoft Teams](#).

P

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Parquet By Microsoft	✗	✓ ³	✗	✓	✓	✗
 Palantir Foundry By Palantir	✗	✓	✗	✗	✗	✗
 Paxata By Paxata	✗	✓	✗	✗	✗	✗
 PDF By Microsoft	✓	✓ ²	✓ ²	✓ ¹	✓	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Planview Enterprise One - CTM (Beta) By Planview	✗	✓	✗	✗	✗	✗
 Planview Enterprise One - PRM (Beta) By Planview	✗	✓	✗	✗	✗	✗
 PostgreSQL database By Microsoft	✓	✓	✓	✓	✓	✓
 Power BI dataflows (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Power BI datasets By Microsoft	✓	✓	✗	✗	✗	✗
 Power Platform dataflows By Microsoft	✗	✓	✗	✓	✓	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Product Insights (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Projectplace for Power BI (Beta) By Planview	✗	✓	✗	✗	✗	✗
 Python Script By Microsoft	✗	✓	✗	✗	✗	✗

¹ Available in [dataflows for Microsoft Teams](#).

² The PDF connector isn't supported in Power BI Premium.

³ The Parquet connector isn't supported in the 32-bit version of Power BI Desktop.

Q

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 QubolePres to Beta By Qubole	✗	✓	✗	✗	✗	✗
 Quickbooks Online (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Quick Base By Quick Base	✗	✓	✗	✗	✗	✗

R

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 R Script By Microsoft	✗	✓	✗	✗	✗	✗
 Roamler (Beta) By Roamler	✗	✓	✗	✗	✗	✗

S

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Salesforce Objects By Microsoft	✓	✓	✓	✓	✓	✓
 Salesforce Reports By Microsoft	✓	✓	✓	✓	✓	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 SAP Business Warehouse Application Server By Microsoft	✗	✓	✓	✓	✓	✓
 SAP Business Warehouse Message Server By Microsoft	✗	✓	✓	✓	✓	✗
 SAP HANA database By Microsoft	✓	✓	✓	✓	✓	✓
 SIS-CC SDMX By SIS-CC	✗	✓	✗	✗	✗	✗
 SharePoint folder By Microsoft	✓	✓	✓	✓	✓	✗
 SharePoint list By Microsoft	✓	✓	✓	✓	✓	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 SharePoint Online list By Microsoft	✓	✓	✓	✓ ¹	✓	✗
 Shortcuts Business Insights (Beta) By Shortcuts	✗	✓	✗	✗	✗	✗
 Sitelimprove By Sitelimprove	✗	✓	✗	✗	✗	✗
 Smartsheet By Microsoft	✗	✓	✓	✗	✗	✗
 Snowflake By Microsoft	✗	✓	✗	✓	✓	✗
 SoftOneBI (Beta) By SoftOne	✗	✓	✗	✗	✗	✗
 Solver By BI360	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Spark By Microsoft	✗	✓	✓	✓	✓	✗
 SparkPost (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Spigit (Beta) By Spigit	✗	✓	✗	✗	✗	✗
 Starburst Enterprise (Beta) By Starburst Data	✗	✓	✗	✗	✗	✗
 SQL Server Analysis Services database By Microsoft	✓	✓	✗	✓	✓	✗
 SQL Server database By Microsoft	✓	✓	✓	✓	✓	✓
 SumTotal (Beta) By SumTotal	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 SurveyMon key (Beta) By SurveyMonke y	✗	✓	✗	✗	✗	✗
 SweetIQ (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Sybase Database By Microsoft	✓	✓	✓	✗	✗	✓

¹ Available in [dataflows for Microsoft Teams](#).

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 TeamDesk (Beta) By ForeSoft	✗	✓	✗	✗	✗	✗
 Tenforce (Smart)List By Tenforce	✗	✓	✗	✗	✗	✗
 Teradata database By Microsoft	✓	✓	✓	✓	✓	✓

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Text/CSV By Microsoft	✓	✓	✓	✓ ¹	✓	✓
 TIBCO(R) Data Virtualizatio n By TIBCO	✗	✓	✗	✗	✗	✗
 Twilio (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

¹ Available in [dataflows for Microsoft Teams](#).

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Usercube (Beta) By Usercube	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Vena (Beta) By Vena	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Vertica By Microsoft	✗	✓	✓	✗	✗	✗
 Vessel Insights (Beta) By Kongsberg	✗	✓	✗	✗	✗	✗

W

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Web By Microsoft	✓	✓	✓	✓ ¹	✓	✗
 Webtrends Analytics (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Witivio (Beta) By Witivio	✗	✓	✗	✗	✗	✗
 Workforce Dimensions (Beta) (Deprecated) By Kronos	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Workplace Analytics (Beta) By Microsoft	✗	✓	✗	✗	✗	✗

¹ Available in [dataflows for Microsoft Teams](#).

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 XML By Microsoft	✓	✓	✓	✓ ¹	✓	✓

¹ Available in [dataflows for Microsoft Teams](#).

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Zendesk (Beta) By Microsoft	✗	✓	✗	✗	✗	✗
 Zoho Creator (Beta) By Zoho	✗	✓	✗	✗	✗	✗

CONNECTOR	EXCEL	POWER BI (DATASETS)	POWER BI (DATAFLOWS)	POWER APPS (DATAFLOWS)	CUSTOMER INSIGHTS (DATAFLOWS)	ANALYSIS SERVICES
 Zucchetti HR Infinity (Beta) By Zucchetti	✗	✓	✗	✗	✗	✗

Next steps

- [Power BI data sources \(datasets\)](#)
- [Connect to data sources for Power BI dataflows](#)
- [Available data sources \(Dynamics 365 Customer Insights\)](#)
- [Data sources supported in Azure Analysis Services](#)

Access database

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Anonymous Windows Basic Organizational Account
Function Reference Documentation	Access.Database

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

If you're connecting to an Access database from Power Query Online, the system that contains the on-premises data gateway must have the 64-bit version of the [Access Database Engine 2010 OLEDB provider](#) installed.

If you're loading an Access database to Power BI Desktop, the versions of the Access Database Engine 2010 OLEDB provider and Power BI Desktop on that machine must match (that is, either 32-bit or 64-bit). For more information, go to [Import Access database to Power BI Desktop](#).

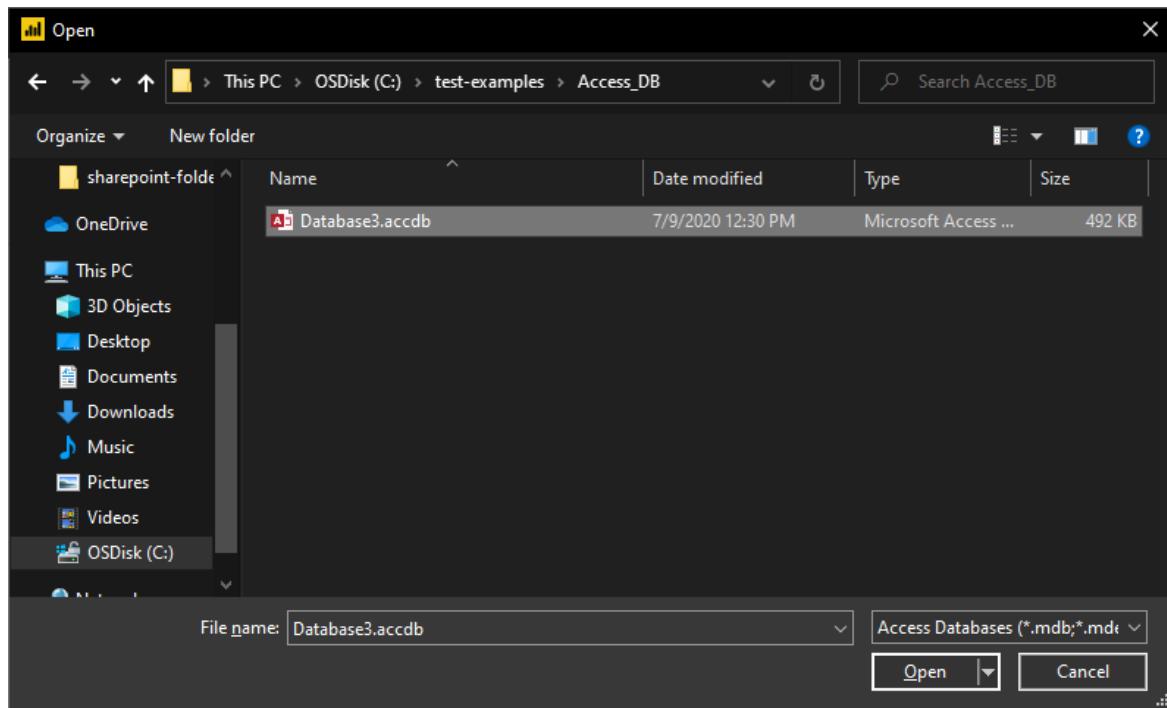
Capabilities Supported

- Import

Connect to an Access database from Power Query Desktop

To make the connection from Power Query desktop:

- Select the **Access database** option in the connector selection.
- Browse for and select the Access database you want to load. Then select **Open**.



If the Access database is online, use the [Web connector](#) to connect to the database.

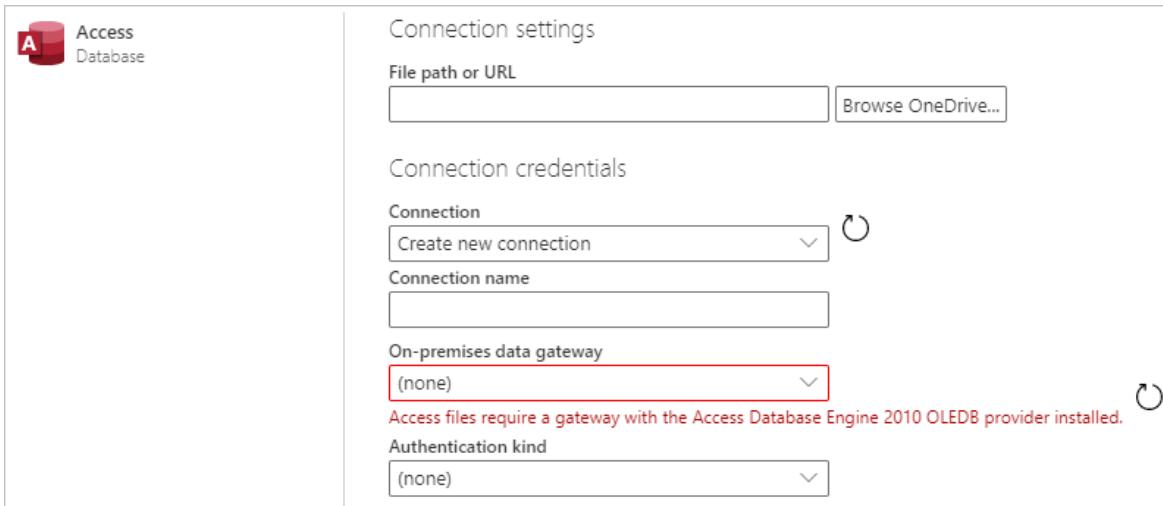
3. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

ID	Field1	Field2	Field3	Field4	Field5
1	Segment	Country	Product	Discount Band	
2	Government	United States of America	Paseo	Low	3
3	Government	France	Amarilla	None	2
4	Government	Germany	Velo	Low	2
5	Government	Germany	Amarilla	Low	2
6	Government	Germany	Velo	Low	2
7	Government	Germany	VTT	Low	2
8	Government	Canada	Carretera	Low	2
9	Government	Canada	Paseo	Low	2
10	Government	France	Amarilla	Medium	2
11	Government	France	Carretera	Low	2
12	Government	France	Paseo	Low	2
13	Government	France	Velo	Low	2
14	Government	France	VTT	Low	2
15	Government	Mexico	VTT	Low	1
16	Government	Canada	Paseo	None	1
17	Government	United States of America	VTT	High	2
18	Government	Germany	Amarilla	Low	1
19	Government	France	Velo	Medium	2
20	Government	France	Amarilla	Medium	2

Connect to an Access database from Power Query Online

To make the connection from Power Query desktop:

1. Select the **Access database** option in the connector selection.
2. In the **Access database** dialog that appears, provide the path to the Access database.



3. Enter the file path or URL address to the Access database.

4. Select the name of your on-premises data gateway.

NOTE

You must select an on-premises data gateway for this connector, whether the Access database is on your local network or on a web site.

5. Select the type of credentials for the connection to the Access database in **Authentication kind**.

6. Enter your credentials.

7. Select **Next** to continue.

8. In **Navigator**, select the data you require, and then select **Transform data** to continue transforming the data in Power Query Editor.

Troubleshooting

Connect to local file from Power Query Online

When you attempt to connect to a local Access database using Power Query Online, you must select an on-premises data gateway, even if your Access database is online.

On-premises data gateway error

A 64-bit version of the Access Database Engine 2010 OLEDB provider must be installed on your on-premises

data gateway machine to be able to load Access database files. If you already have a 64-bit version of Microsoft Office installed on the same machine as the gateway, the Access Database Engine 2010 OLEDB provider is already installed. If not, you can download the driver from the following location:

<https://www.microsoft.com/download/details.aspx?id=13255>

Import Access database to Power BI Desktop

In some cases, you may get a `The 'Microsoft.ACE.OLEDB.12.0' provider is not registered` error when attempting to import an Access database file to Power BI Desktop. This error may be caused by using mismatched bit versions of Power BI Desktop and the Access Database Engine 2010 OLEDB provider. For more information about how you can fix this mismatch, see [Troubleshoot importing Access and Excel .xls files in Power BI Desktop](#).

Adobe Analytics

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI Desktop
Authentication Types Supported	Organizational account
Function Reference Documentation	AdobeAnalytics.Cubes

Prerequisites

Before you can sign in to Adobe Analytics, you must have an Adobe Analytics account (username/password).

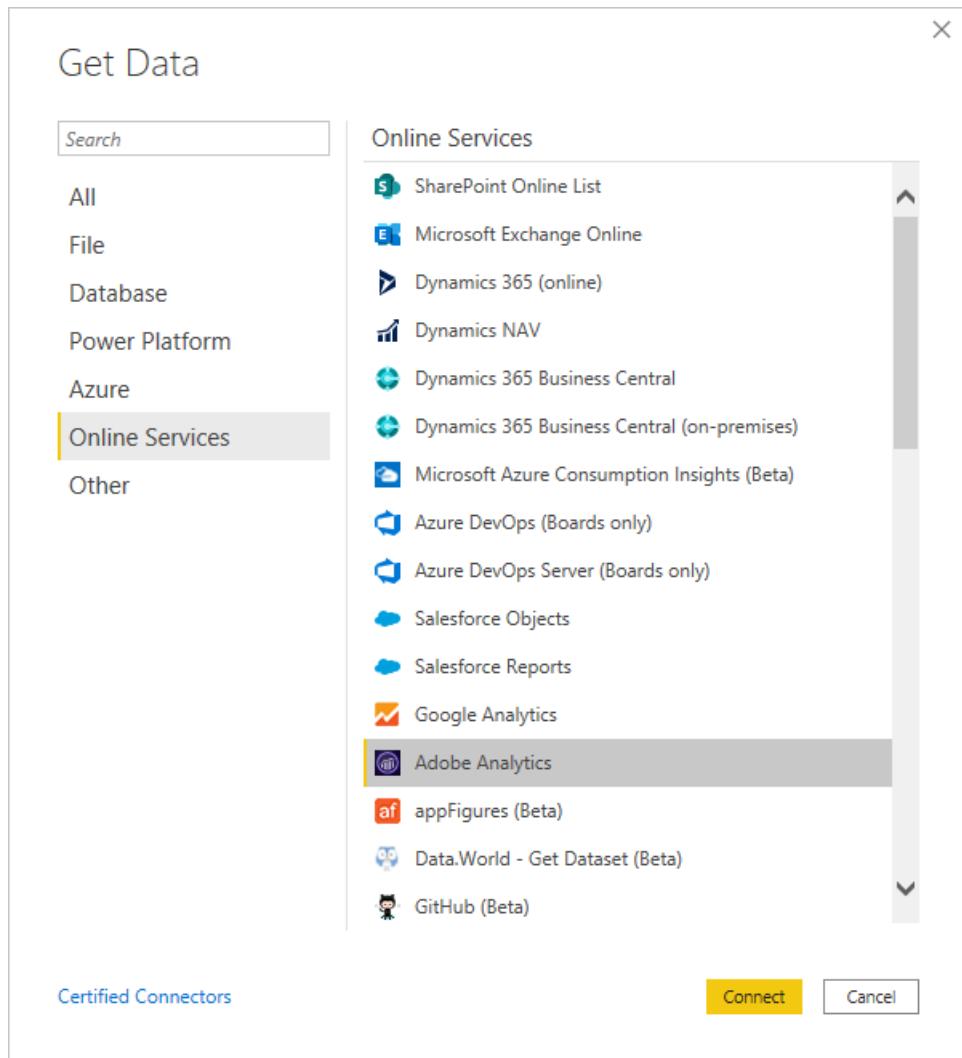
Capabilities Supported

- Import

Connect to Adobe Analytics data

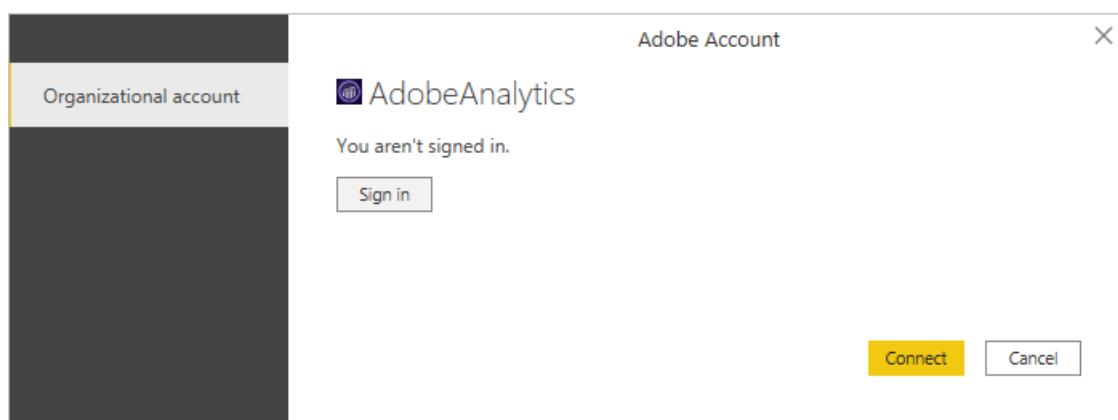
To connect to Adobe Analytics data:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **Online Services** from the categories on the left, select **Adobe Analytics**, and then select **Connect**.



2. If this is the first time you're getting data through the Adobe Analytics connector, a third-party notice will be displayed. Select **Don't warn me again with this connector** if you don't want this message to be displayed again, and then select **Continue**.

3. To sign in to your Adobe Analytics account, select **Sign in**.



4. In the Adobe Analytics window that appears, provide your credentials to sign in to your Adobe Analytics account. You can either supply a username (which is usually an email address), or select **Continue with Google** or **Continue with Facebook**.

 For your protection, please verify your identity.



Sign in

New user? [Create an account](#)

Email address

[Continue](#)

Or

 Continue with Google

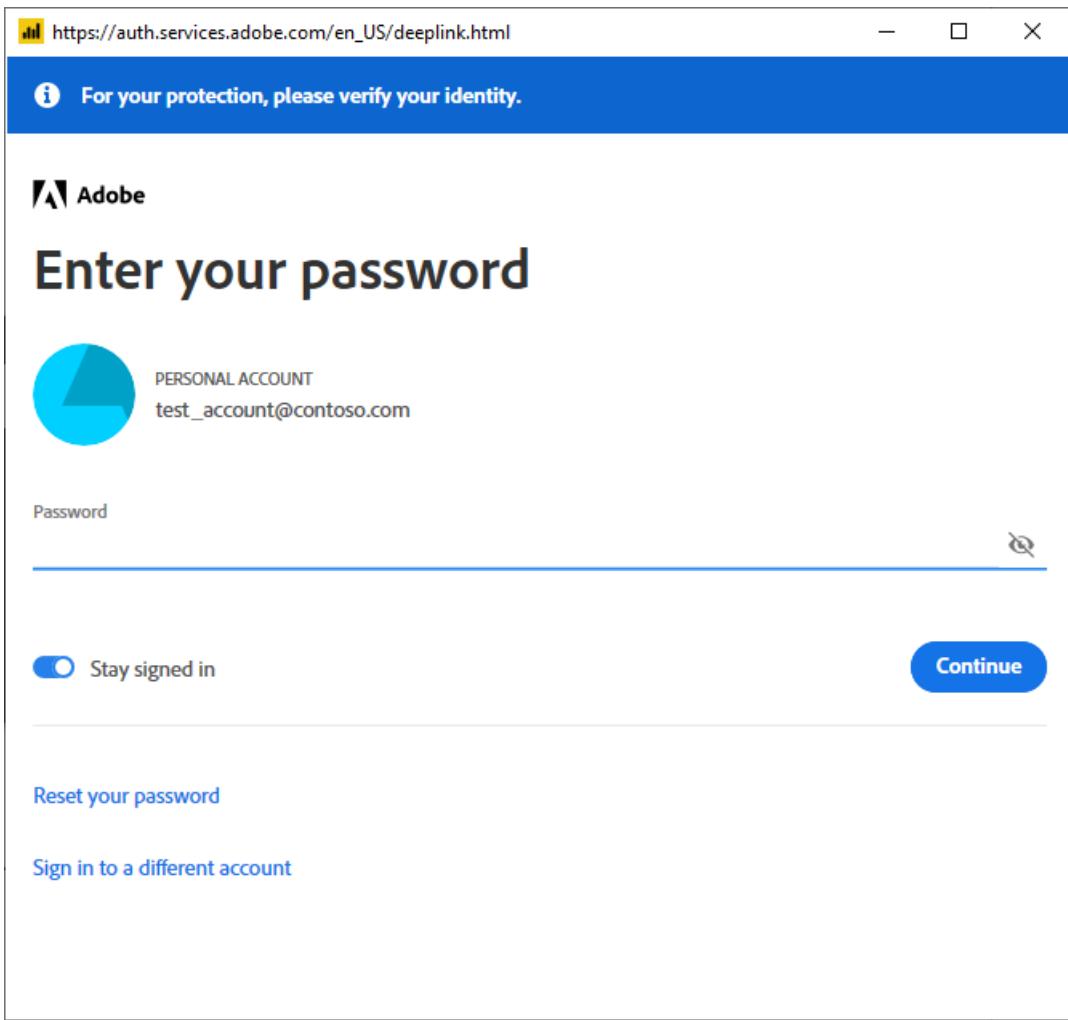
 Continue with Facebook

 Continue with Apple

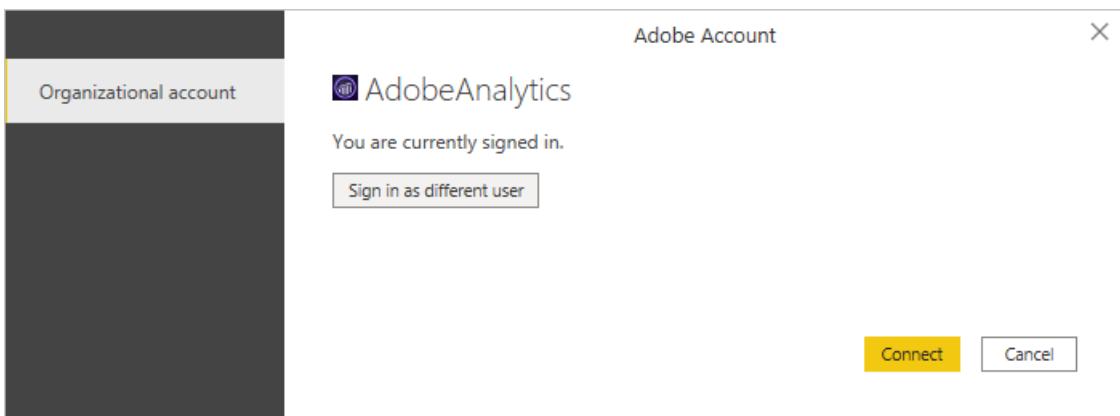
Protected by reCAPTCHA and subject to the Google [Privacy Policy](#) and [Terms of Service](#).

If you entered an email address, select **Continue**.

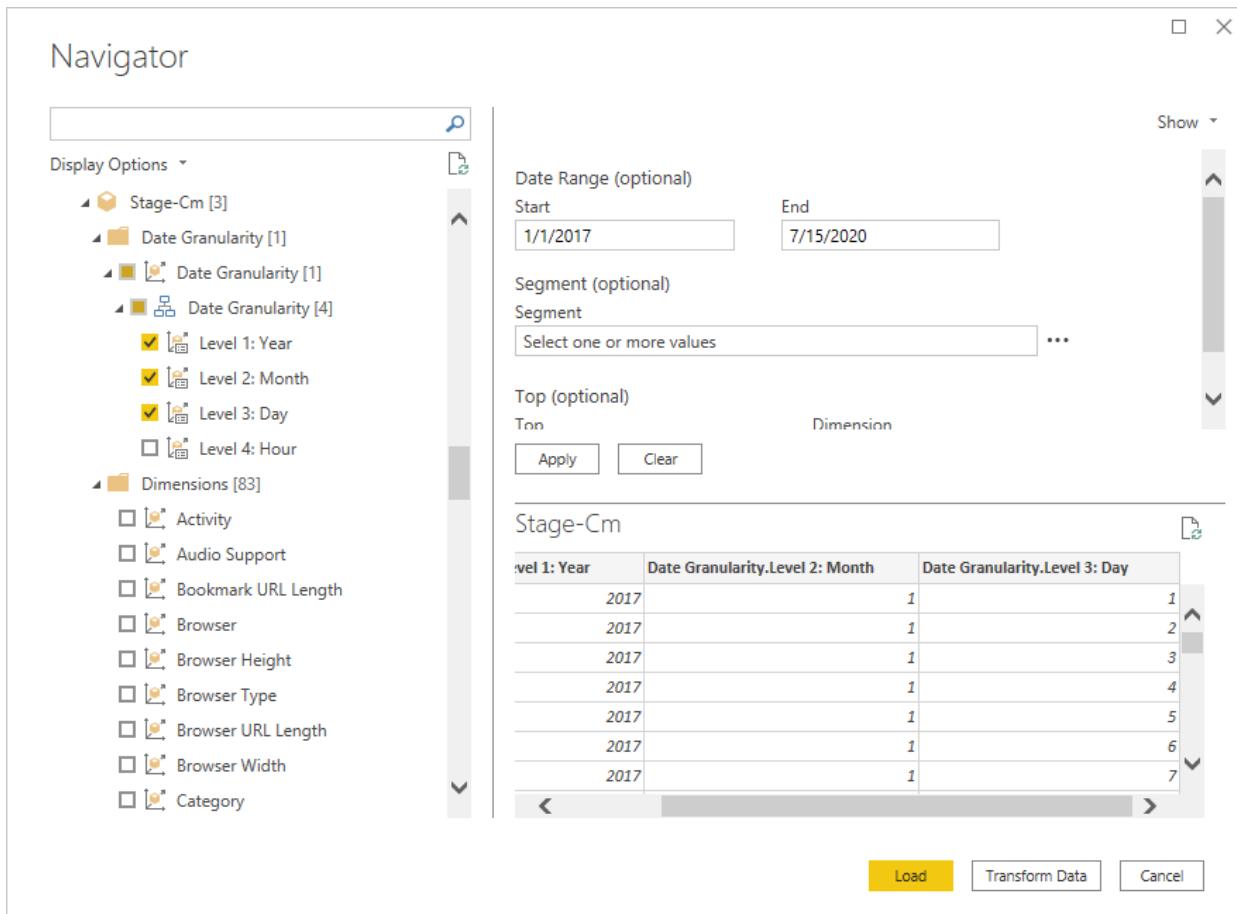
5. Enter your Adobe Analytics password and select **Continue**.



6. Once you've successfully signed in, select **Connect**.

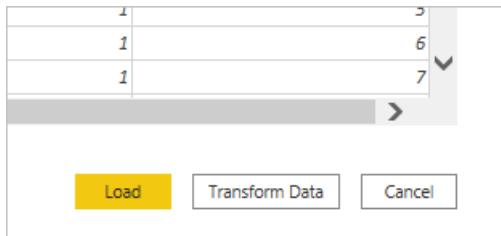


Once the connection is established, you can preview and select multiple dimensions and measures within the **Navigator** dialog box to create a single tabular output.



You can also provide any optional input parameters required for the selected items. For more information about these parameters, see [Optional input parameters](#).

You can **Load** the selected table, which brings the entire table into Power BI Desktop, or you can select **Transform Data** to edit the query, which opens Power Query Editor. You can then filter and refine the set of data you want to use, and then load that refined set of data into Power BI Desktop.



Optional input parameters

When you've selected the Adobe Analytics data you want to load or transform in the Power Query Navigator dialog box, you can also limit the amount of data by selecting a set of optional input parameters.

Show ▾

Date Range (optional)

Start End

Segment (optional)

Segment ...

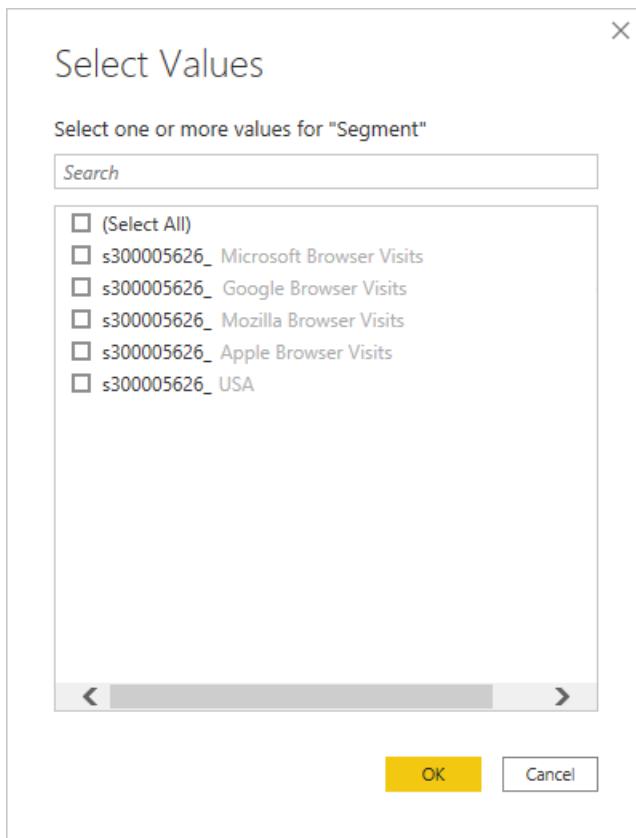
Top (optional)

Top ... Dimension ...

Apply **Clear**

These input parameters are:

- Date Range—filter with a reporting range between a start date and an end date that you set.
- Segment—filter the data based on all segments contained in the data, or only those segments you select. To change the list of segments, select the ellipsis to the right of the **Segment** list box, then choose the segments you want. By default, all segments are included in the data.



- Top—filter the data based on the top items for the dimension. You can enter a value in the **Top** text box, or select the ellipsis next to the text box to select some default values. By default, all items are selected.
- Dimension—filter the data based on the selected dimension. By default, all dimensions are selected. Custom Adobe dimension filters are not currently supported in the Power Query user interface, but can be defined by hand as M parameters in the query. For more information, see [Using Query Parameters in Power BI Desktop](#).

Limitations and issues

You should be aware of the following limitations and issues associated with accessing Adobe Analytics data.

- Adobe Analytics has a built-in limit of 50 K rows returned per API call.
- If the number of API calls exceeds four per second, a warning will be issued. If the number exceeds five per second, an error message will be returned. For more information about these limits and the associated messages, see [Web Services Error Codes](#).
- The API request timeout through adobe.io is currently 60 seconds.
- The default rate limit for an Adobe Analytics Company is 120 requests per minute per user (the limit is enforced as 12 requests every 6 seconds).

Import from Adobe Analytics will stop and display an error message whenever the Adobe Analytics connector hits any of the API limits listed above.

When accessing your data using the Adobe Analytics connector, follow the guidelines provided under the [Best Practices](#) heading.

For additional guidelines on accessing Adobe Analytics data, see [Recommended usage guidelines](#).

Next steps

You may also find the following Adobe Analytics information useful:

- [Adobe Analytics 1.4 APIs](#)
- [Adobe Analytics Reporting API](#)
 - [Metrics](#)
 - [Elements](#)
 - [Segments](#)
- [GetReportSuites](#)
- [Adobe Analytics support](#)

Amazon Athena

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Amazon, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Amazon website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	DSN configuration Organizational account

Prerequisites

- An [Amazon Web Services \(AWS\) account](#)
- [Permissions](#) to use Athena
- Customers must install the [Amazon Athena ODBC driver](#) before using the connector

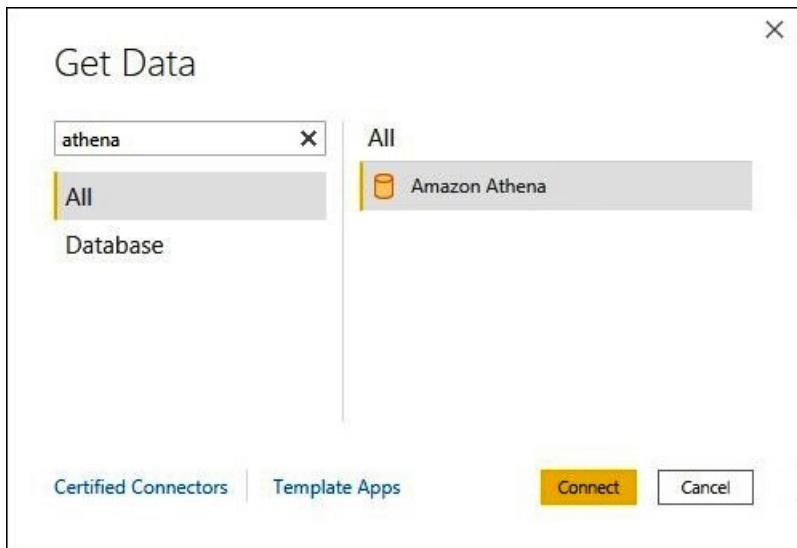
Capabilities supported

- Import
- DirectQuery

Connect to Amazon Athena

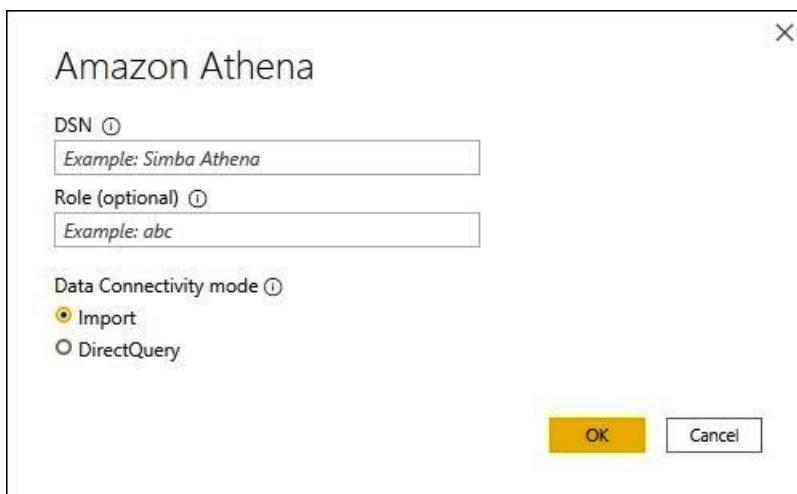
To connect to Athena data:

1. Launch Power BI Desktop.
2. In the **Home** tab, select **Get Data**.
3. In the search box, enter **Athena**.
4. Select **Amazon Athena**, and then select **Connect**.



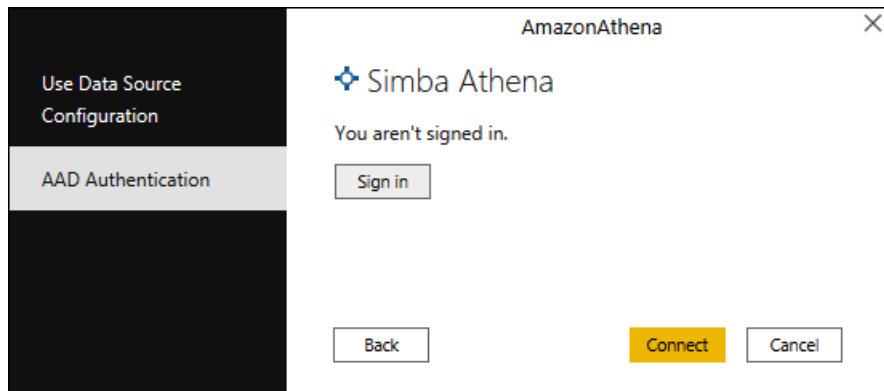
5. On the **Amazon Athena** connection page, enter the following information:

- For **DSN**, enter the name of the ODBC DSN that you want to use. For instructions on configuring your DSN, go to the [ODBC driver documentation](#).
- For **Data Connectivity mode**, choose a mode that's appropriate for your use case, following these general guidelines:
 - For smaller datasets, choose **Import**. When using import mode, Power BI works with Athena to import the contents of the entire dataset for use in your visualizations.
 - For larger datasets, choose **DirectQuery**. In DirectQuery mode, no data is downloaded to your workstation. While you create or interact with a visualization, Microsoft Power BI works with Athena to dynamically query the underlying data source so that you're always viewing current data. More information: [Use DirectQuery in Power BI Desktop](#)



6. Select **OK**.

7. At the prompt to configure data source authentication, select either **Use Data Source Configuration** or **AAD Authentication**. Enter any required sign-in information. Then select **Connect**.



Your data catalog, databases, and tables appear in the **Navigator** dialog box.

The screenshot shows the 'Navigator' dialog box. On the left, the 'Display Options' pane shows a tree view with nodes like 'demo-dsn [1]', 'AwsDataCatalog [3]', 'default [8]', 'demo-datasets [2]' (selected), 'iris' (selected), 'demo_datasets', and 'sampledb [5]'. The main area displays a preview of the 'iris' dataset with columns: species, sepal_length, sepal_width, petal_length, and petal_width. The data table contains 50 rows of Iris flower measurements. At the bottom are 'Load', 'Transform Data', and 'Cancel' buttons.

8. In the **Display Options** pane, select the check box for the dataset that you want to use.
9. If you want to transform the dataset before you import it, go to the bottom of the dialog box and select **Transform Data**. This selection opens the Power Query Editor so that you can filter and refine the set of data you want to use.
10. Otherwise, select **Load**. After the load is complete, you can create visualizations like the one in the following image. If you selected **DirectQuery**, Power BI issues a query to Athena for the visualization that you requested.

File **Home** **Insert** **Modeling** **View** **Help**

Clipboard **Data** **Queries** **Insert** **Calculations** **Share**

Tabular View

species	petal_length	petal_width	sepal_length	sepal_width
setosa	73.10	12.30	250.30	171.40
versicolor	213.00	66.30	296.80	138.50
virginica	277.60	101.30	329.40	148.70

Measurements

● sepal_width ● sepal_length ● petal_width ● petal_length

The chart displays the total measurement values for each species, broken down by category. The total height of the bars increases from approximately 450 for setosa to about 850 for virginica and 700 for versicolor.

Species	sepal_width	sepal_length	petal_width	petal_length	Total
setosa	~100	~150	~10	~10	~450
virginica	~150	~150	~50	~250	~850
versicolor	~100	~150	~10	~250	~700

Amazon Redshift

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Dynamics 365 Customer Insights
Authentication Types Supported	Amazon Redshift Basic Microsoft account Organizational account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

- An [Amazon Web Services \(AWS\) account](#)

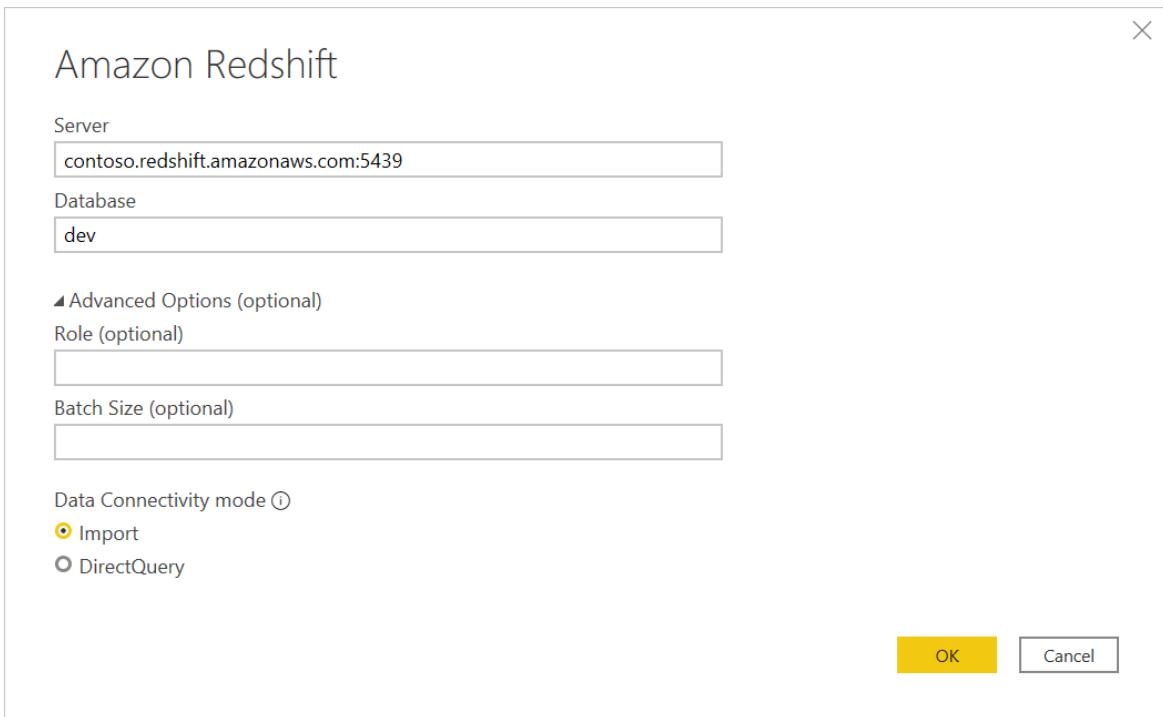
Capabilities supported

- Import
- DirectQuery (Power BI Desktop only)
- Advanced options
 - Roles
 - Batch size

Connect to Amazon Redshift data from Power Query Desktop

To connect to Amazon Redshift data:

1. Select the **Amazon Redshift** option in the **Get Data** selection.
2. In **Server**, enter the server name where your data is located. As part of the Server field, you can also specify a port in the following format: *ServerURL:Port*. In **Database**, enter the name of the Amazon Redshift database you want to access. In this example, `contoso.redshift.amazonaws.com:5439` is the server name and port number, `dev` is the database name, and **Data Connectivity mode** is set to **Import**.

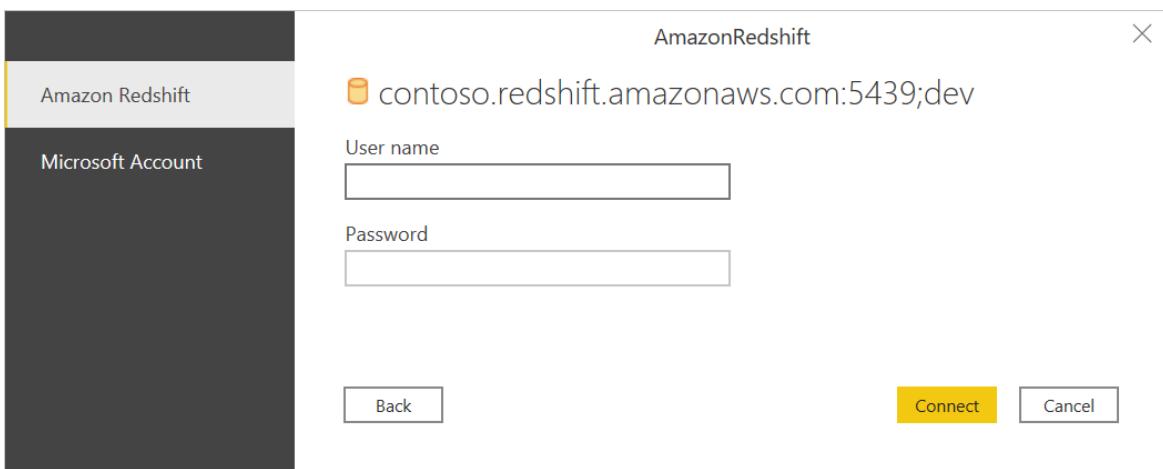


You can also choose some optional advanced options for your connection. More information: [Connect using advanced options](#)

Select either the **Import** or **DirectQuery** data connectivity mode.

After you have finished filling in and selecting all the options you need, select **OK**.

3. If this is the first time you're connecting to this database, enter your credentials in the **User name** and **Password** boxes of the Amazon Redshift authentication type. Then select **Connect**.



More information: [Authentication with a data source](#)

4. Once you successfully connect, a **Navigator** window appears and displays the data available on the server. Choose one or more of the elements you want to import.

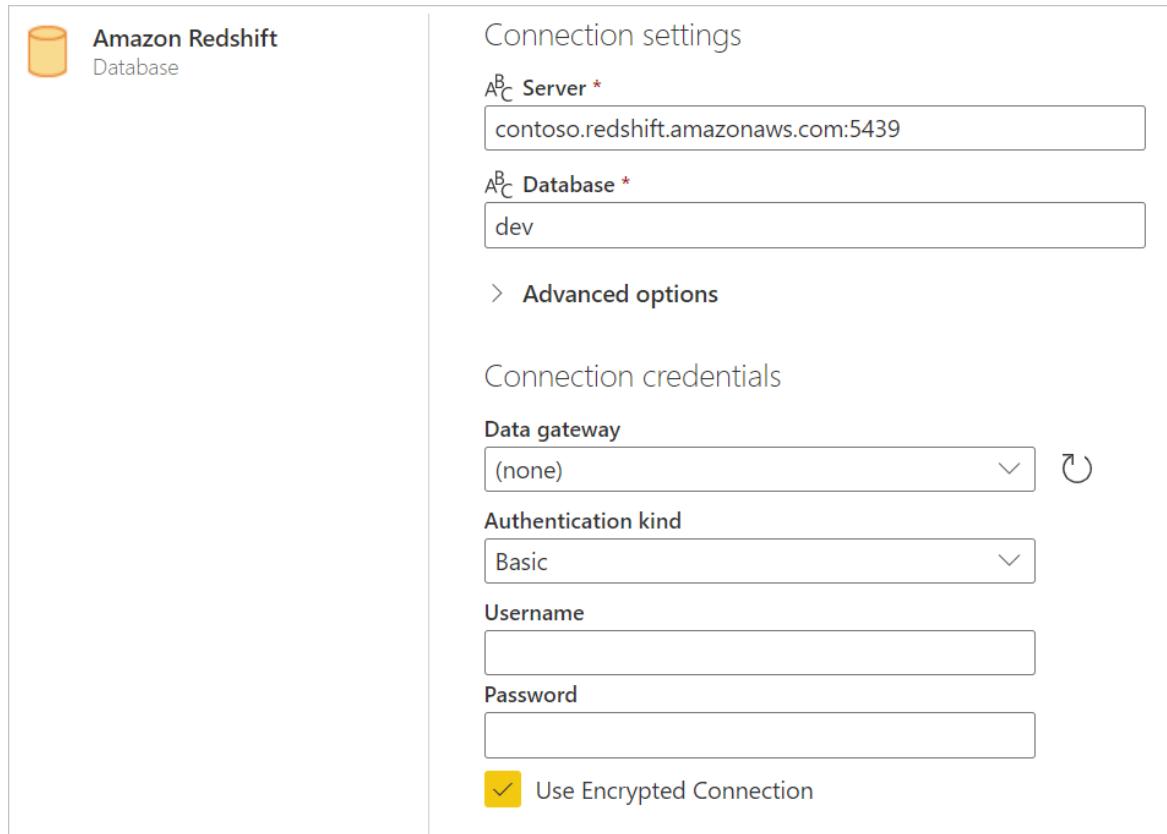
The screenshot shows the Power Query Navigator window. On the left, there's a tree view of available databases and tables. Under 'contoso.redshift.amazonaws.com', 'northwind [11]', and 'orders', the 'order details' table has a checked checkbox. The main area displays the 'orders' table with columns: orderid, customerid, employeeid, orderdate, requireddate, and shippeddate. The table contains 30 rows of data. At the bottom right of the table view, there are three buttons: 'Load' (yellow), 'Transform Data' (white), and 'Cancel'.

- Once you've selected the elements you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

Connect to Amazon Redshift data from Power Query Online

To connect to Amazon Redshift data:

- Select the **Amazon Redshift** option in the **Power Query - Choose data source** page.
- In **Server**, enter the server name where your data is located. As part of the Server field, you can also specify a port in the following format: *ServerURL:Port*. In **Database**, enter the name of the Amazon Redshift database you want to access. In this example, `contoso.redshift.amazonaws.com:5439` is the server name and port number, and `dev` is the database name.



The screenshot shows the 'Amazon Redshift Database' connection settings. It includes fields for 'Server' (contoso.redshift.amazonaws.com:5439), 'Database' (dev), and 'Advanced options'. Below this is a section for 'Connection credentials' with fields for 'Data gateway' (set to '(none)'), 'Authentication kind' (Basic), 'Username', 'Password', and a checked 'Use Encrypted Connection' checkbox.

You can also choose some optional advanced options for your connection. More information: [Connect using advanced options](#)

3. If needed, select the on-premises data gateway in **Data gateway**.
4. Select the type of authentication you want to use in **Authentication kind**, and then enter your credentials.
5. Select or clear **Use Encrypted Connection** depending on whether you want to use an encrypted connection or not.
6. Select **Next** to continue.
7. In **Navigator**, select the data you require, and then select **Transform data**. This selection opens the Power Query Editor so that you can filter and refine the set of data you want to use.

Connect using advanced options

Power Query provides a set of advanced options that you can add to your query if needed.

The following table describes all of the advanced options you can set in Power Query.

ADVANCED OPTION	DESCRIPTION
Role	Provides an Amazon Resource Name (ARN), which uniquely identifies AWS resources.
Batch size	Specifies the maximum number of rows to retrieve at a time from the server when fetching data. A small number translates into more calls to the server when retrieving a large dataset. A large number of rows may improve performance, but could cause high memory usage. The default value is 100 rows.

ADVANCED OPTION	DESCRIPTION
SQL Statement	For information, go to Import data from a database using native database query .

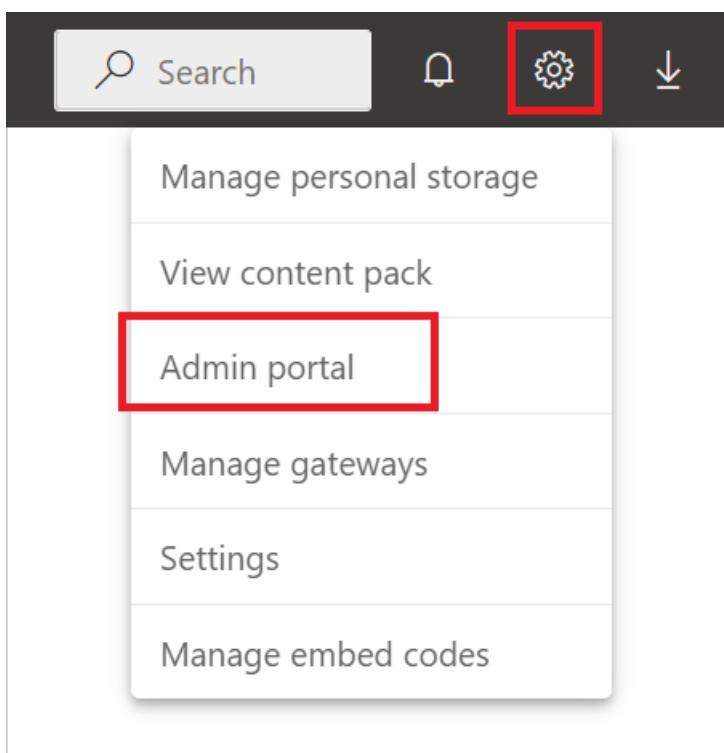
Enable Azure AD Single Sign-On (SSO) for Amazon Redshift

We support Azure AD SSO through both Power BI Service (cloud) and also through the on-premises data gateway. For more information about enabling Azure AD SSO for all connectors, go to [Overview of single sign-on \(SSO\) for on-premises data gateways in Power BI](#).

Azure AD Single Sign-On (SSO) through Power BI service

To configure a new connection in Power BI service:

1. In Power BI service, select **Admin portal** from the settings list.



2. Enable the **Redshift SSO** option.

◀ **Redshift SSO**
Enabled for the entire organization

Enable SSO capability for Redshift. By enabling, user access token information, including name and email, will be sent to Redshift for authentication.

Enabled

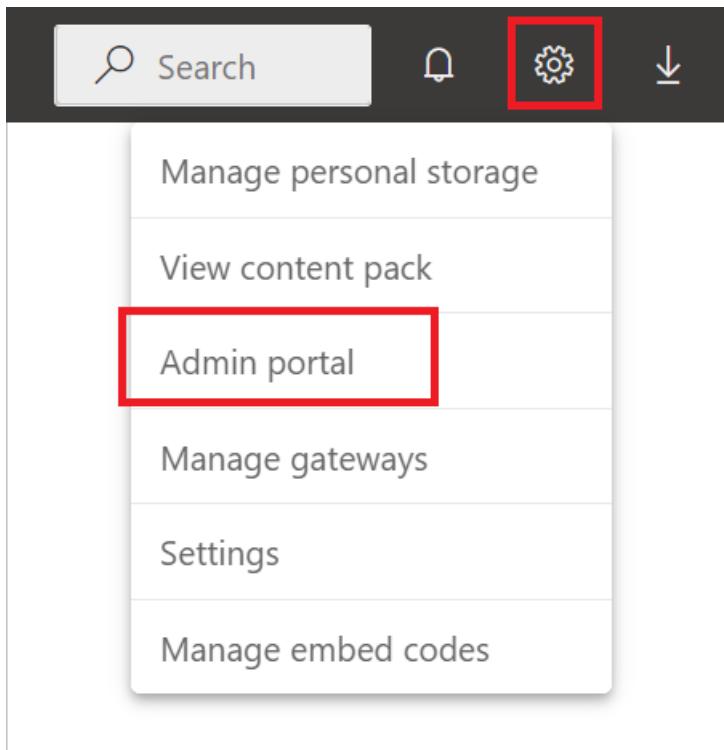
Apply **Cancel**

ⓘ This setting applies to the entire organization

Azure AD Single Sign-On (SSO) for Amazon Redshift with an on-premises data gateway

Before you can enable Azure AD SSO for Amazon Redshift, you must first enable Azure AD SSO for all data sources that support Azure AD SSO with an on-premises data gateway:

1. In Power BI service, select **Admin portal** from the settings list.



2. Under **Tenant settings**, enable **Azure AD Single-Sign On (SSO) for Gateway**.

A screenshot of the "Azure AD Single Sign-On (SSO) for Gateway" settings dialog. It shows a section titled "Enabled for the entire organization" with a description: "Enable AAD SSO via the on-premises data gateway for applicable data sources. By enabling user access token information including name and email will be sent to these data sources for authentication via the on-premises data gateway." Below this is a yellow toggle switch labeled "Enabled". At the bottom of the dialog are two buttons: "Apply" and "Cancel". A note at the bottom states: "ⓘ This setting applies to the entire organization".

Once you've enabled Azure AD SSO for all data sources, then enable Azure AD SSO for Amazon Redshift:

3. Also enable the **Redshift SSO** option.

4 Redshift SSO

Enabled for the entire organization

Enable SSO capability for Redshift. By enabling, user access token information, including name and email, will be sent to Redshift for authentication.

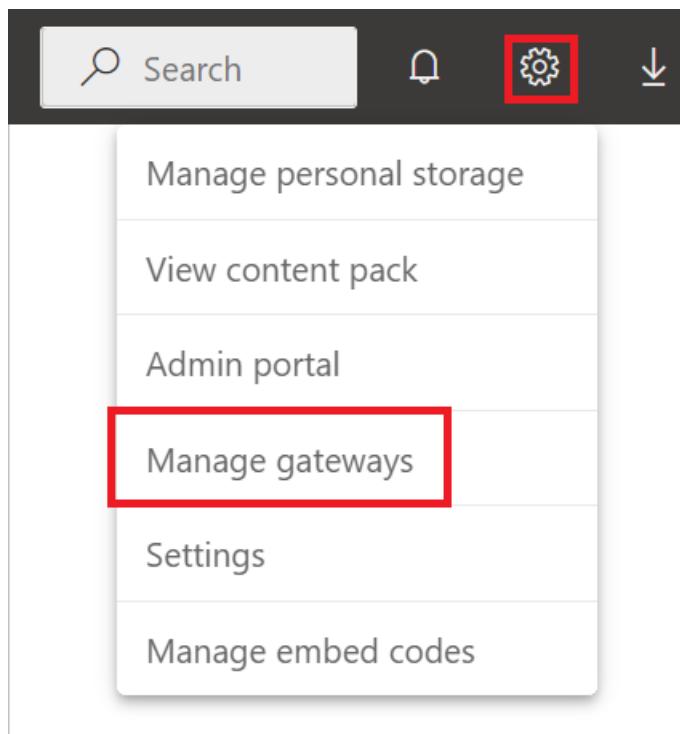


Apply

Cancel

(i) This setting applies to the entire organization

4. Select **Manage gateways** from the settings list.



5. Select a gateway, and then select **Choose Data Source**.

6. Under the **Data Source Settings** tab, enter a value in **Role**. The **Role** parameter is required when using AAD and needs to be specified in **Advanced Settings**.

Also select **Use SSO via Azure AD for DirectQuery queries**.

Data Source Settings

Users

Data Source Name

Data Source Type

Server

Database

Authentication Method

Skip Test Connection

✓Advanced settings

Role

Batch Size

Use SSO via Azure AD for DirectQuery queries

This setting will only be applied for DirectQuery datasets. Import will use the Username and Password specified in the data source details. [Learn more](#)

Connection Encryption setting for this data source

Privacy Level setting for this data source

Anaplan

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Anaplan, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Anaplan website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Basic
Function Reference Documentation	-

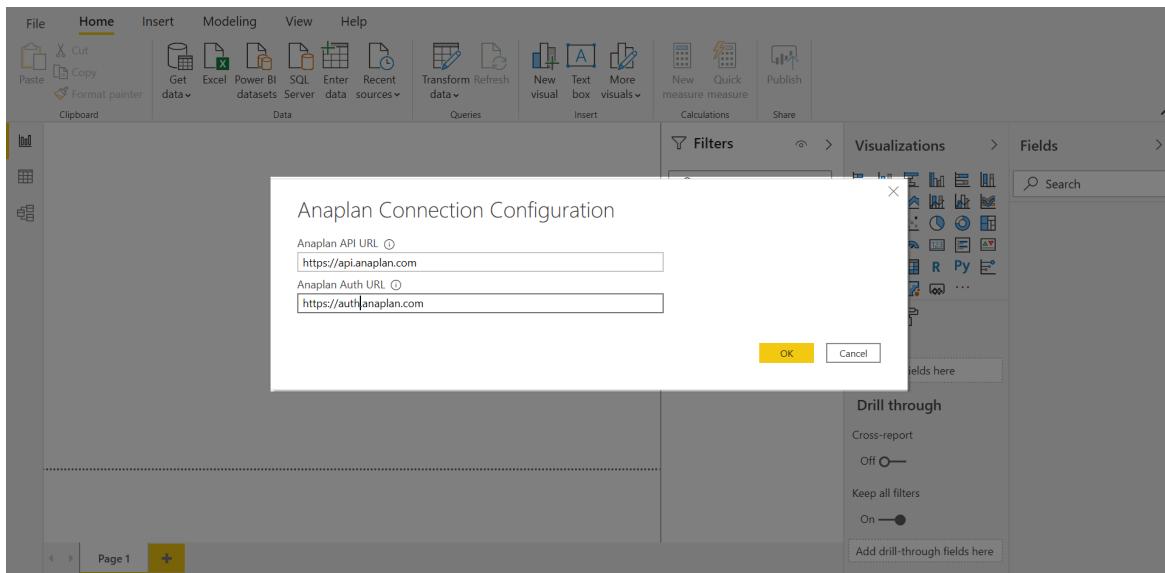
Capabilities supported

The connector runs through Anaplan public data integration APIs and allows you to load all Anaplan models (aside from archived ones) and saved export actions into Power BI.

Connect to Anaplan from Power Query Desktop

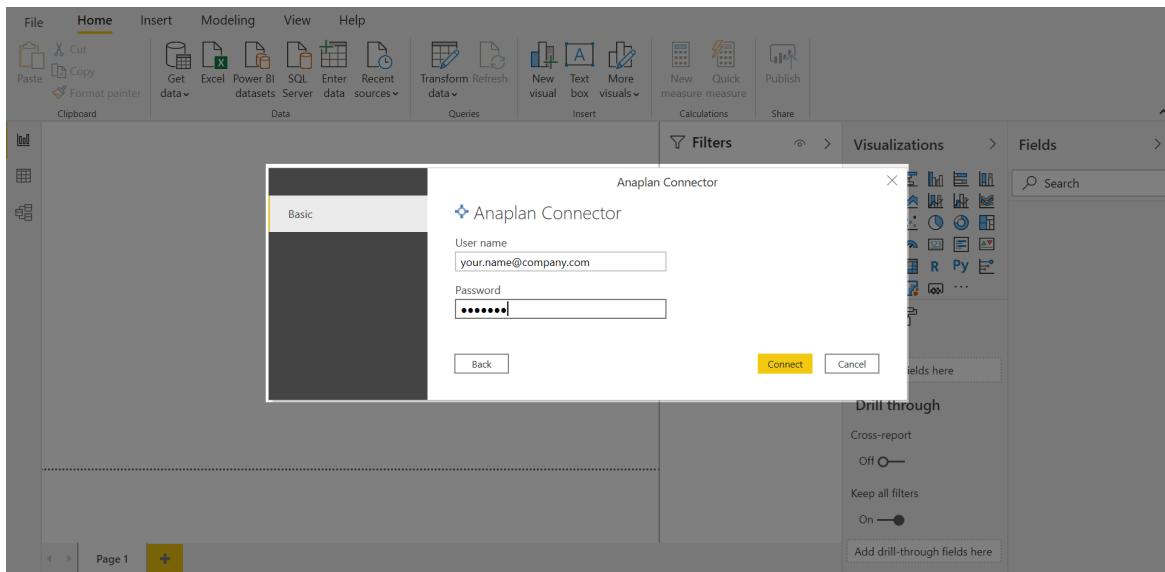
To connect to Anaplan data:

1. Select **Anaplan** from the product-specific data connector list, and then select **Connect**.
2. In the Anaplan Connector Configuration screen, enter the API and Authentication URLs:
 - **Anaplan API URL:** <https://api.anaplan.com>
 - **Anaplan Auth URL:** <https://auth.anaplan.com>



After you've entered the API and Auth URL, select **Ok**.

3. Sign in to the connector to verify your access to an Anaplan workspace.



Once you've succeeded, select **Connect**.

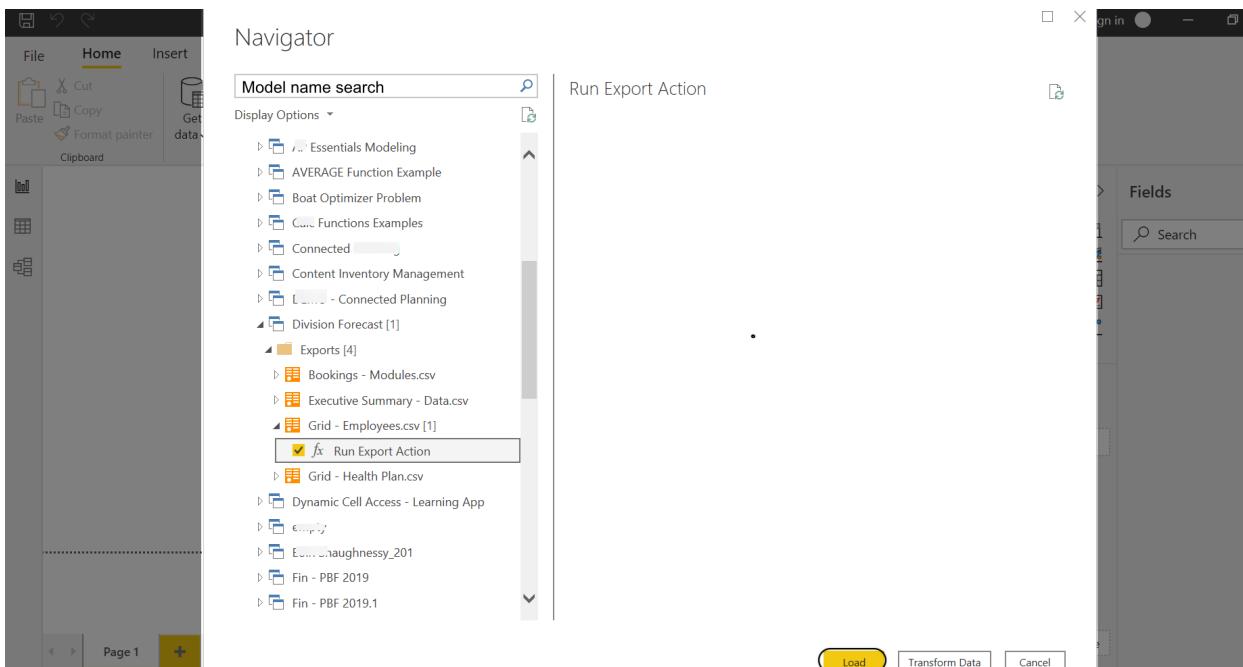
Run an export action

The Anaplan connector leverages export actions to download data from your Anaplan model. First, ensure that you have an export action set.

When you run an export action:

- Only exports that output .csv and .txt files are supported.
- With every export action run, you need to wait ten minutes to repeat the same export action. The time is calculated from one export run completion until the next export run begins. The 10 minute wait does not apply to different exports.
- If you don't see the export action in the Power BI connector, check your role and the export actions in your model.

To run an export action, use the **Navigator** dialog to locate your export.



1. Search your Anaplan models to find and set your export. You can also locate for your model name via the search field.
2. Check the box next to **fx Run Export Action** to select your export.
 - When you select the **fx Run Export Action**, this does not trigger the export run. Instead this selection downloads the last version of the exported Anaplan data for preview.
 - A preview displays in the right panel. If the Anaplan export is set to **Admins only**, model users may see a blank preview, but the export will run as normal.
 - You'll see the preview the next time you set an integration with same export.
3. Select **Load** to trigger the export run. The **Load** dialog displays and your data loads.

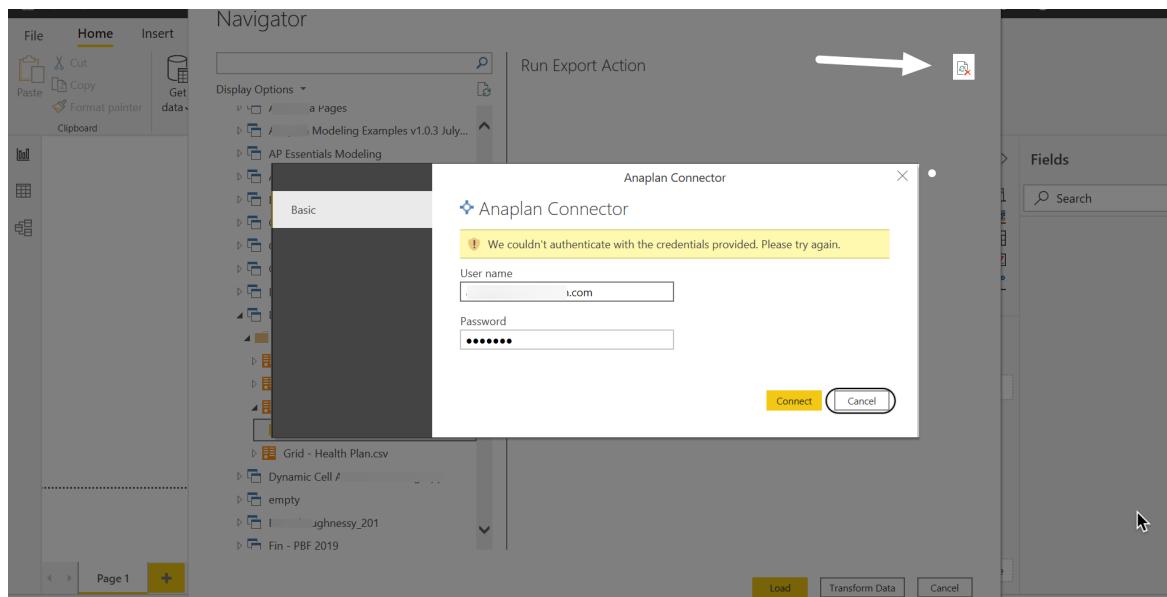
Troubleshooting

If you get a connector related error message, first, try refreshing.

Credential error in the Navigator

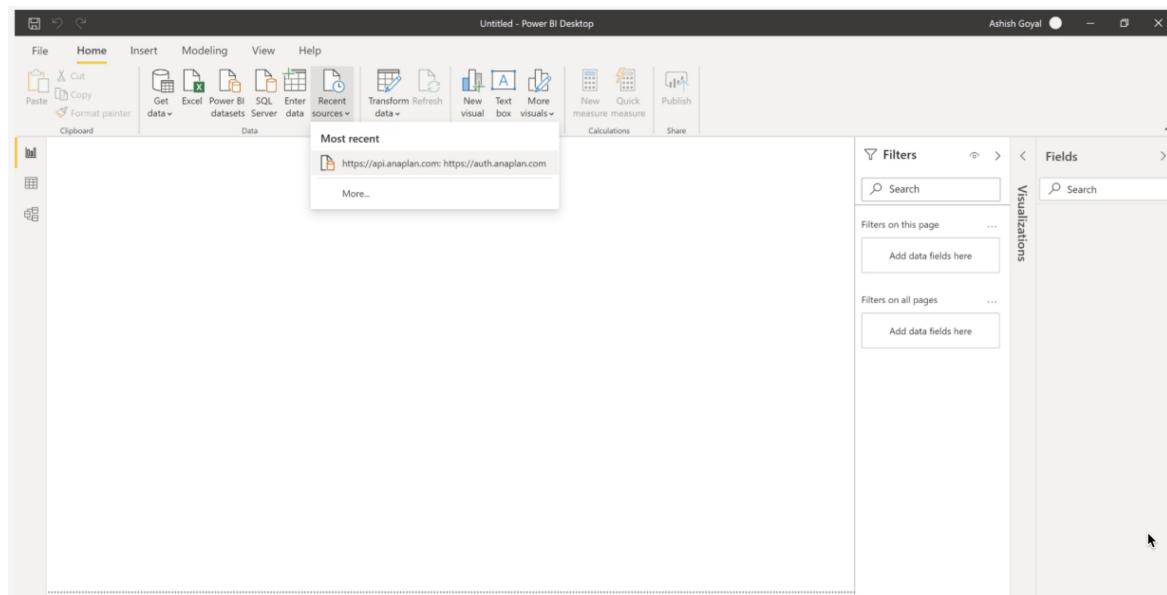
Do one of the following:

- Clear cache within Power BI (**File, Options, Clear cache**) and restart the connector, or
- Select **Cancel** and select **Refresh** (top right).

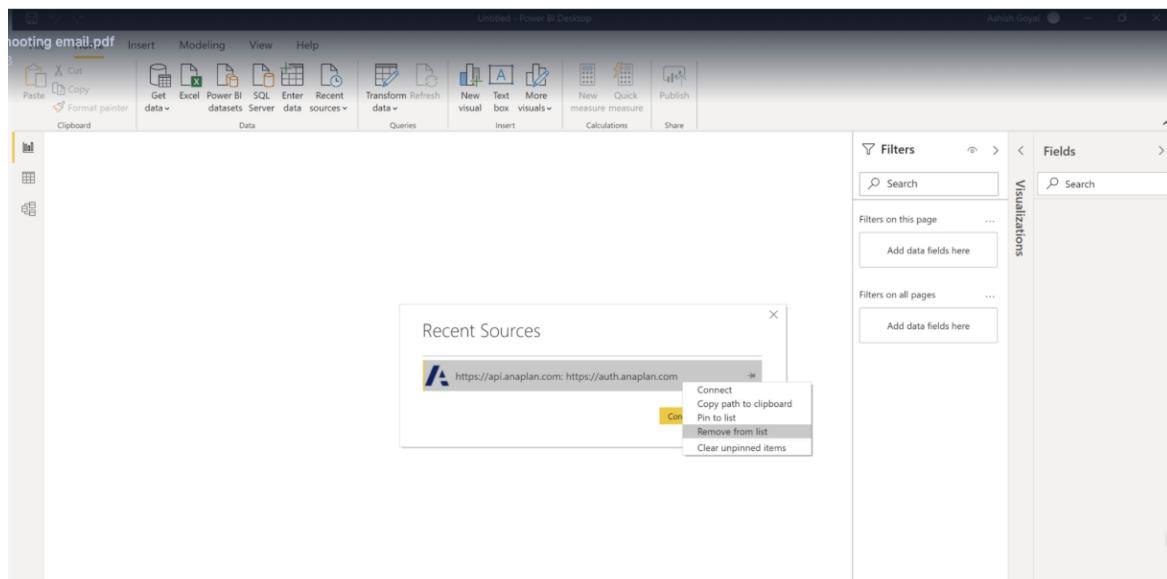


If you still receive a credential error after you clear cache, also clear your recent sources.

1. Select Recent sources



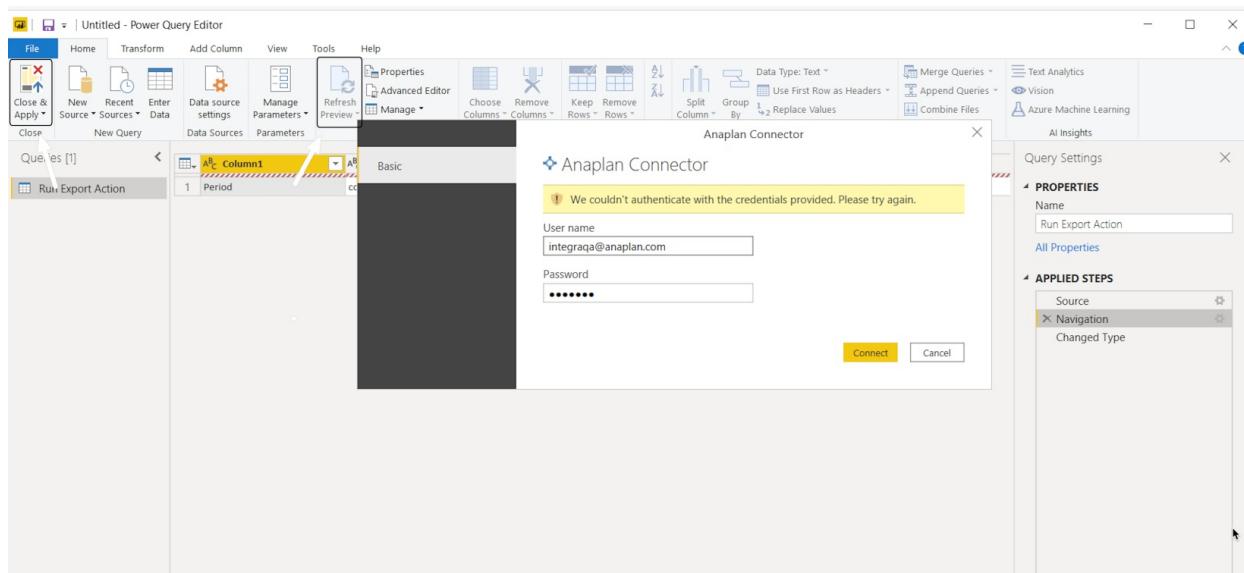
2. Select Remove from list



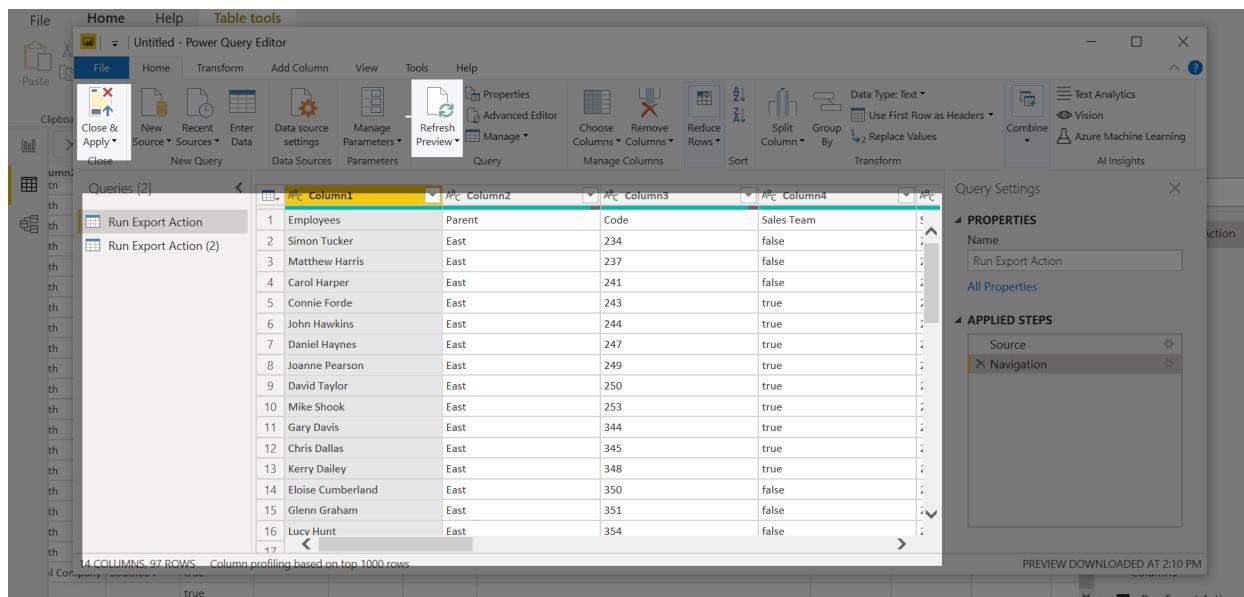
3. Establish the connection to the export again, and your data refreshes.

Credential error in the Power Query editor

If you encounter a credential error in the Power Query editor, select **Close & Apply** or **Refresh Preview** to refresh the data.

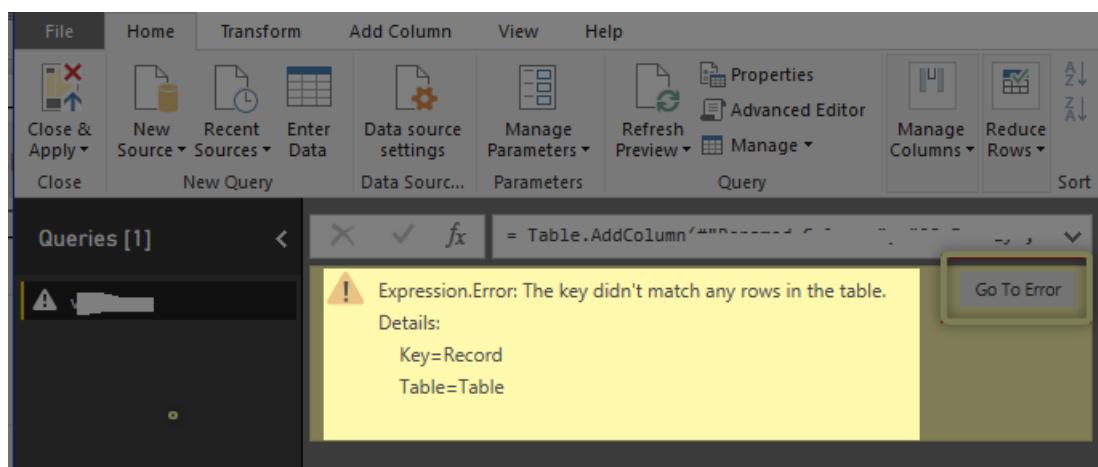


Your data will refresh, resolving the error.



Power Query expression error

If you encounter a Power Query expression error, select **Refresh Preview** to refresh the data.



Your data will refresh, resolving the error.

File Home Help Table tools

Clipboard Paste Close & Apply New Source Sources Data Data source settings Refresh Preview Advanced Editor Manage Parameters Choose Columns Remove Columns Reduce Rows Sort Split Column Group By Data Type: Text Use First Row as Headers Combine Text Analytics Vision Azure Machine Learning AI Insights

Queries [2]

Run Export Action Run Export Action (2)

A ₁ Column1	A ₁ Column2	A ₁ Column3	A ₁ Column4
1 Employees	Parent	Code	Sales Team
2 Simon Tucker	East	234	false
3 Matthew Harris	East	237	true
4 Carol Harper	East	241	true
5 Connie Forde	East	243	true
6 John Hawkins	East	244	true
7 Daniel Haynes	East	247	true
8 Joanne Pearson	East	249	true
9 David Taylor	East	250	true
10 Mike Shook	East	253	true
11 Gary Davis	East	344	true
12 Chris Dallas	East	345	true
13 Kerry Dailey	East	348	true
14 Eloise Cumberland	East	350	false
15 Glenn Graham	East	351	false
16 Lucy Hunt	East	354	false
17			

14 COLUMNS, 97 ROWS Column profiling based on top 1000 rows.

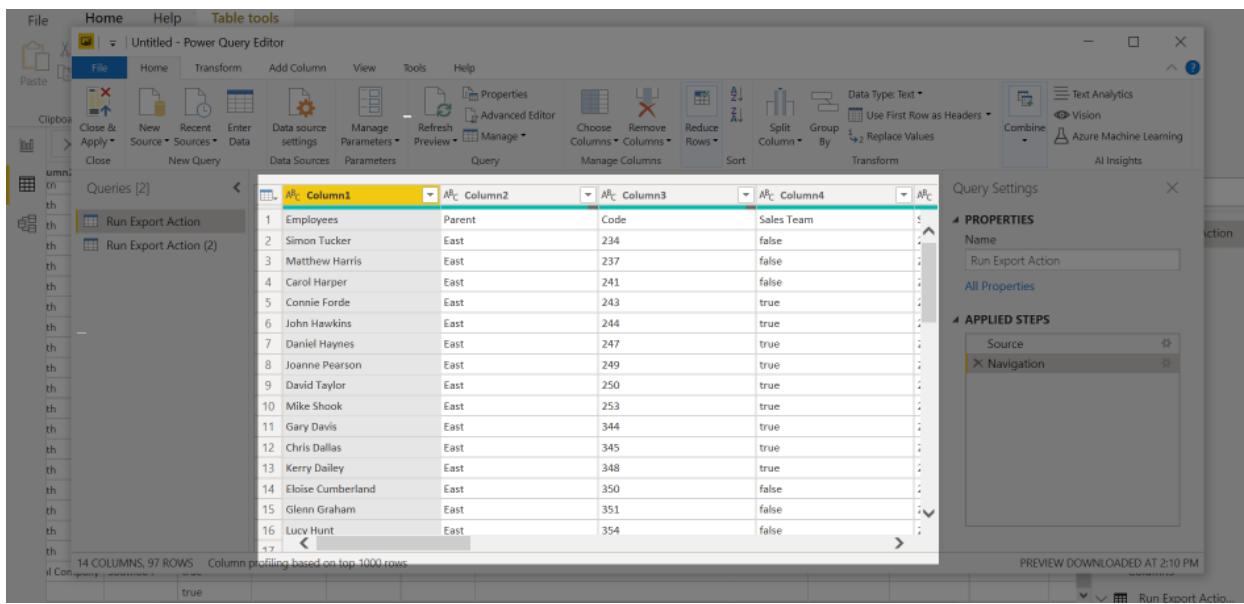
Run Export Action

Query Settings

PROPERTIES Name: Run Export Action All Properties

APPLIED STEPS Source: Navigation

PREVIEW DOWNLOADED AT 2:10 PM



Assemble Views

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Autodesk, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Autodesk website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	GA
Products	Power BI (Datasets)
Authentication Types Supported	Autodesk Account
Function Reference Documentation	-

Prerequisites

To use the Assemble Views connector, you must have an Autodesk account with a username and password, and be a member of at least one project in Assemble.

You'll also need at least one view associated with the Assemble project.

Capabilities supported

- Import

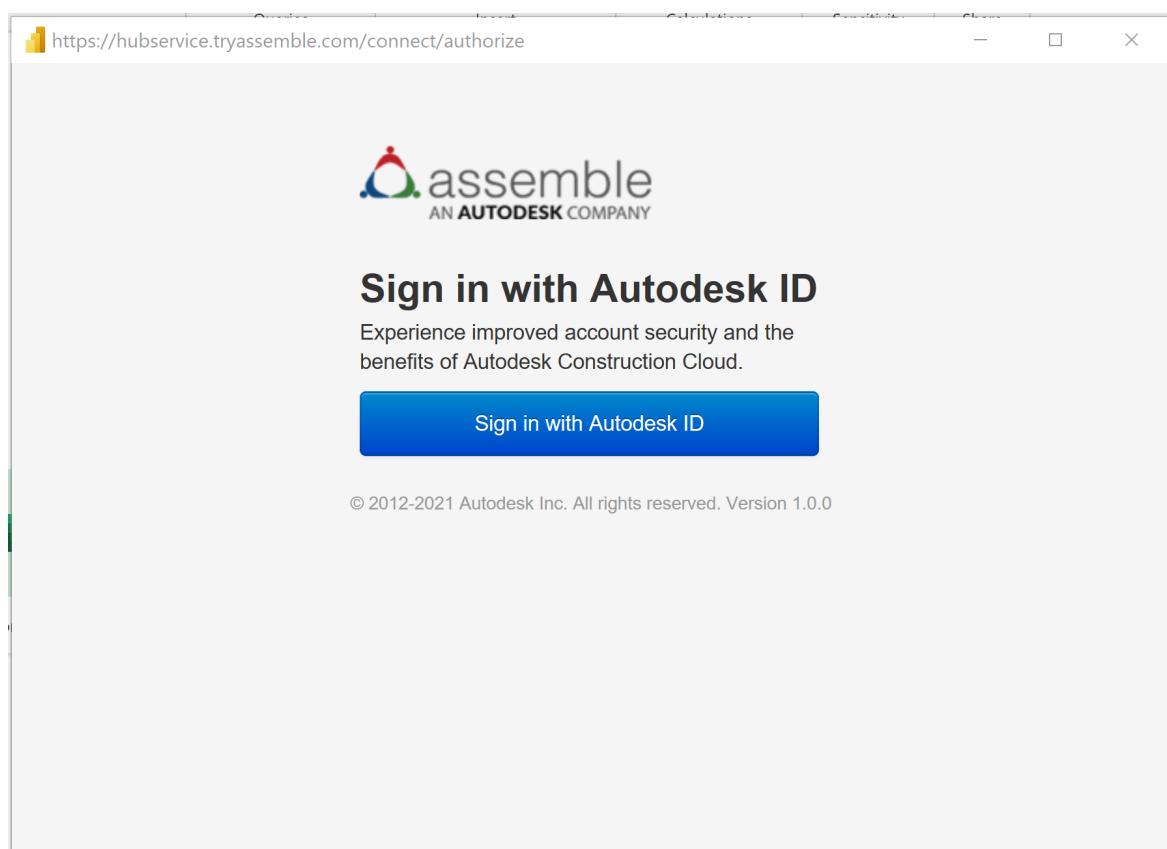
Connect to Assemble Views from Power Query Desktop

To connect to Assemble data:

1. Select **Assemble Views** from the **Get Data** experience under the **Online Services** category, and then select **Connect**.
2. In **Assemble Views**, enter your site's URL to sign in. For example, use <https://example.tryassemble.com>.
 - a. (Optional) Select a date from which you want to load the data. Leaving this entry blank results in the latest data being pulled each time you refresh.



- b. Once you've entered the URL, select **OK** to continue.
3. Select **Sign in** to sign in to your Autodesk account.



- Once you've successfully signed in, select **Connect**.
4. In the **Navigator** dialog box, select the Assemble Views that you want to load. For each project, there's a single item for view images named **[Your Project] View Thumbnails**. Select this option if you want to include images in your report. Select **Transform Data** to continue to Power Query.

Mechanical JEA

Display Options ▾

- Autodesk Hospital (UselThursday Demo)...
 - Mechanical JEA
 - Piping JEA
 - Steve's View 05.06
 - Structural Framing DGM
 - Structural Framing Eddie
 - Yvonne's View
 - Autodesk Hospital (UselThursday De...)

Rows

Record
Record
Record
Record
Record
Record
Record
Record
Record
Record

5. In Power Query, you'll see a single column named **Rows**. On the header of the column, select the button with two arrows pointing in opposite directions to expand your rows.

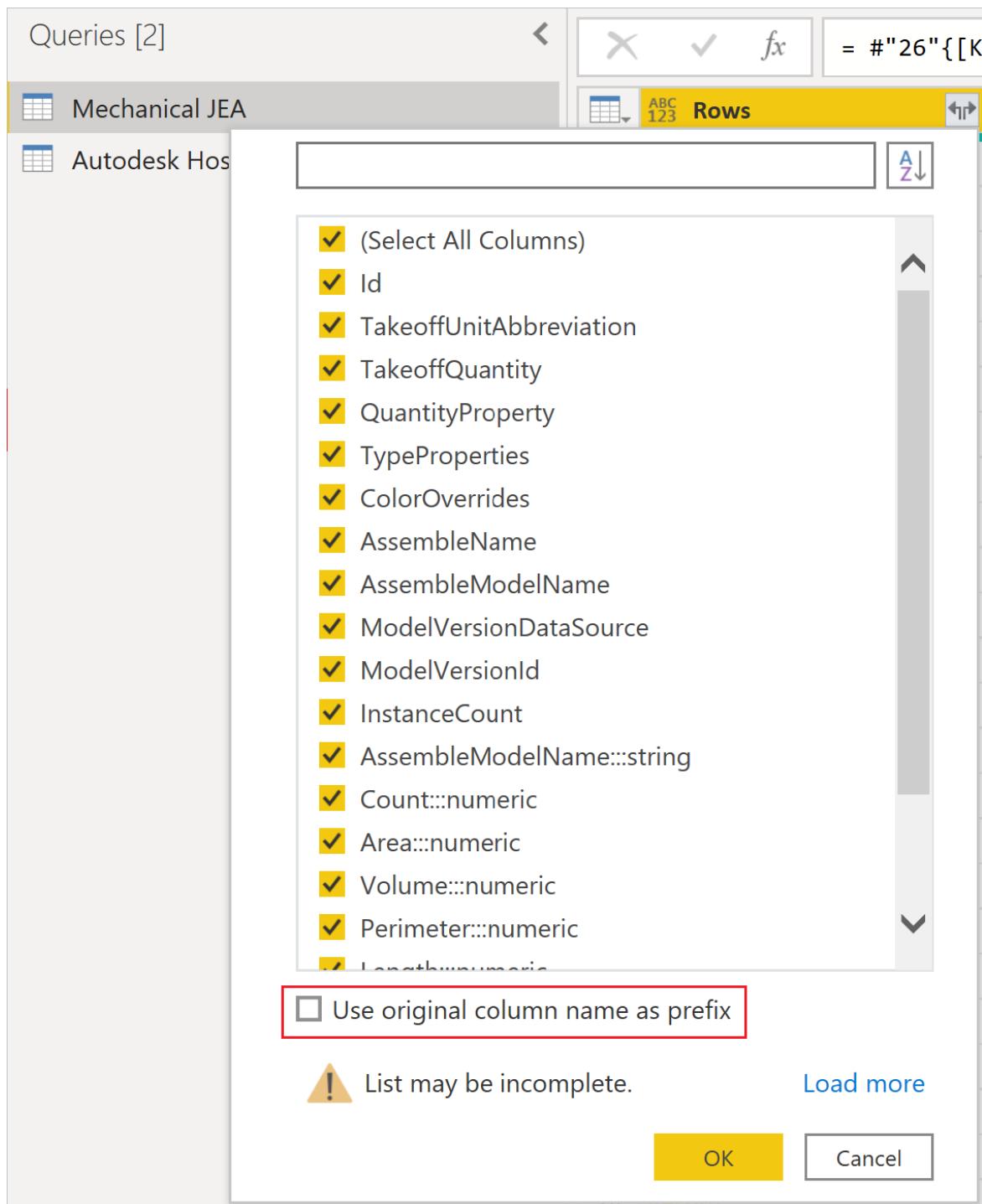
Queries [2]

- Mechanical JEA
- Autodesk Hospital (UselThursday Demo)...

ABC 123 Rows

	Rows
1	Record
2	Record
3	Record
4	Record
5	...

- a. Uncheck **Use original column name as prefix** and select **OK** for each view data query you've selected.

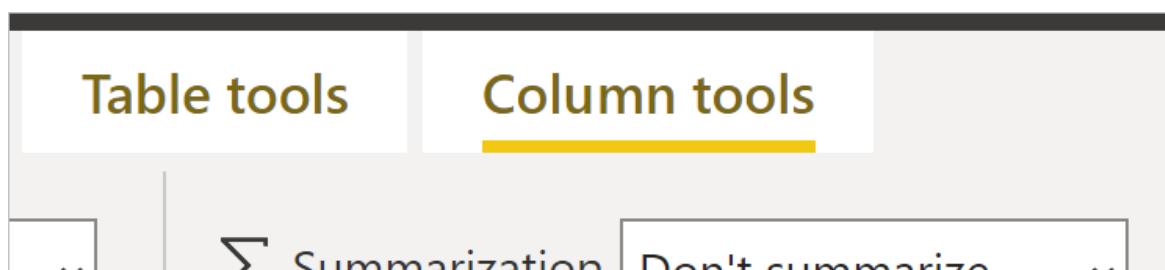


b. Select **Close & Apply** to load the datasets.

6. (Optional) If you have chosen to load images, you'll need to update the **Data category** for the image field.

a. Expand the **[Your Project] View Thumbnails** table, and then select the **Image** field. This selection opens the **Column tools** tab.

b. Open the **Data category** drop-down and select **Image URL**. You can now drag and drop the Image field into your report visuals.



Summarization Don't summarize

Data category Uncategorized

Address

Place

City

County

State or Province

Postal code

Country

Continent

Latitude

Longitude

Web URL

Image URL

Barcode

Build visualizations

or drag fields from the sidebar into the visualization area.

The screenshot shows a user interface for building visualizations. At the top, there are two dropdown menus: 'Summarization' (set to 'Don't summarize') and 'Data category' (set to 'Uncategorized'). A vertical sidebar on the left lists several data categories: Address, Place, City, County, State or Province, Postal code, Country, Continent, Latitude, Longitude, Web URL, Image URL, and Barcode. In the main area, there is a large text box containing the placeholder 'Build visualizations' and a smaller text box below it containing 'or drag fields from the sidebar into the visualization area.' A small icon of a document with a dashed border is visible near the bottom of the sidebar.

Known issues and limitations

- Views with greater than 100,000 rows may not load depending on the number of fields included in the view. To avoid this limitation, we suggest breaking large views into multiple smaller views and appending the queries in your report, or creating relationships in your data model.

- The **view images** feature currently only supports thumbnail sized images because of a row size limitation in Power BI.

Automy Data Analytics (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by ACEROYALTY, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the ACEROYALTY website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI Desktop
Authentication Types Supported	Automy Report Token

Prerequisites

Before you can sign in to Automy Data Analytics, you must have an Automy Report Token.

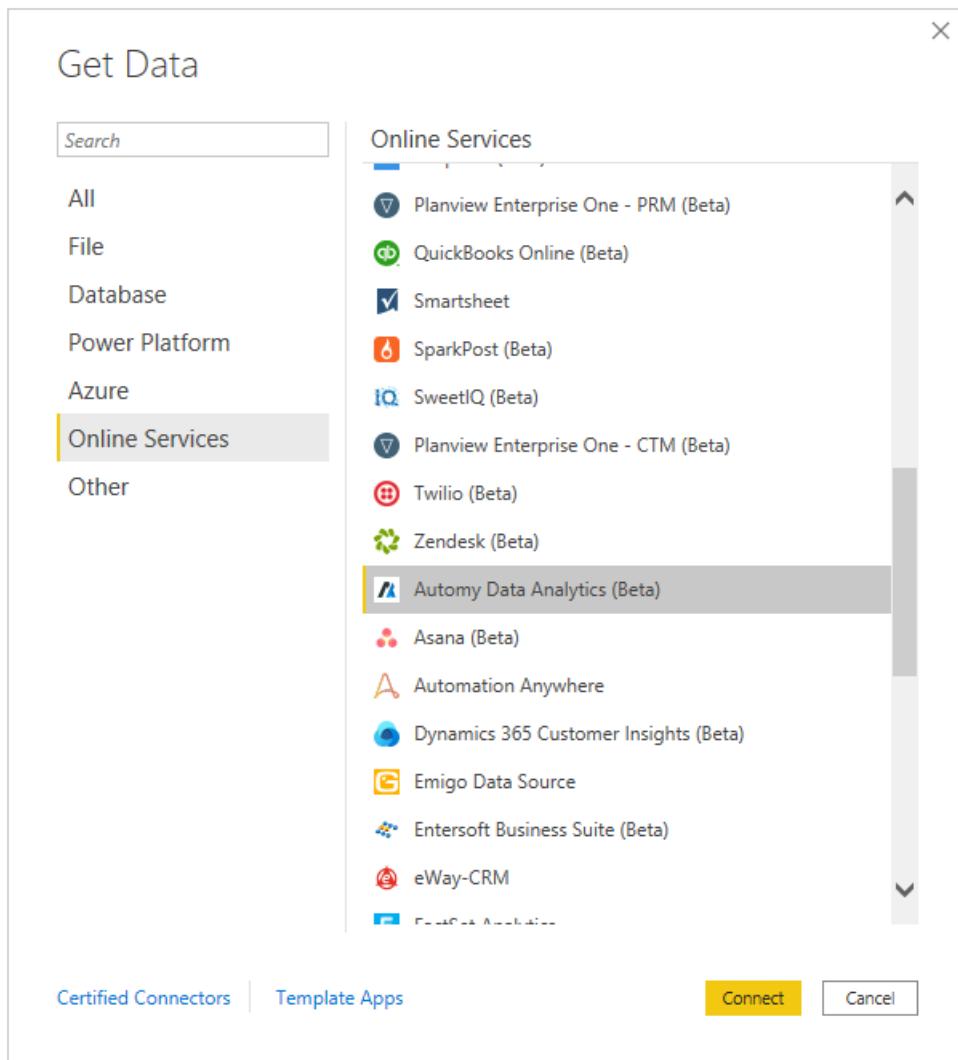
Capabilities Supported

- Import

Connect to Automy Data Analytics data

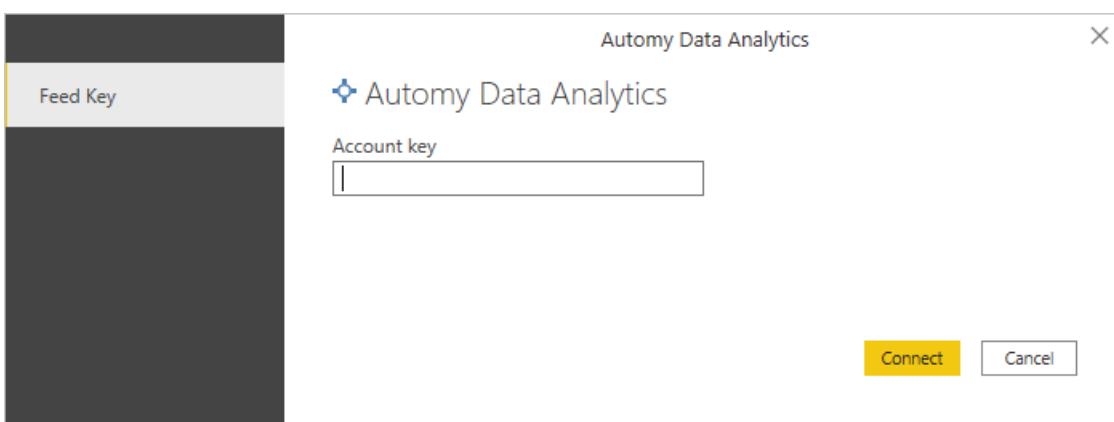
To connect to Automy Data Analytics data:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **Online Services** from the categories on the left, select **Automy Data Analytics**, and then select **Connect**.



2. If this is the first time you're connecting to the Automy Data Analytics connector, a third-party notice will be displayed. Select **Don't warn me again with this connector** if you don't want this message to be displayed again, and then select **Continue**.

3. Sign in to the connector with API Key to verify your access to Automy.



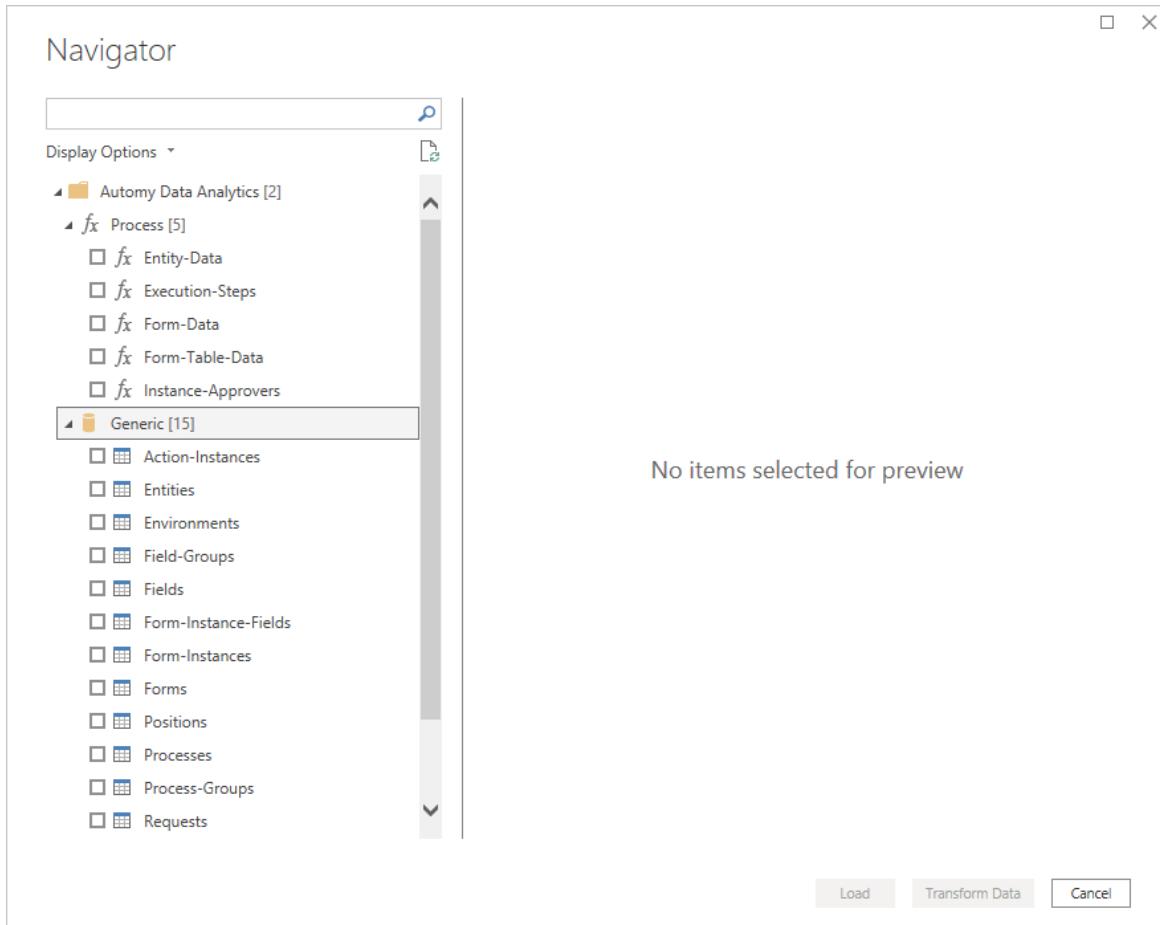
Once you've succeeded, select **Connect**.

4. In the Automy Data Analytics window that appears, select the correct parameters to prepare the connection. Select the type of report and data type and completed the token information, and then select **Ok**.

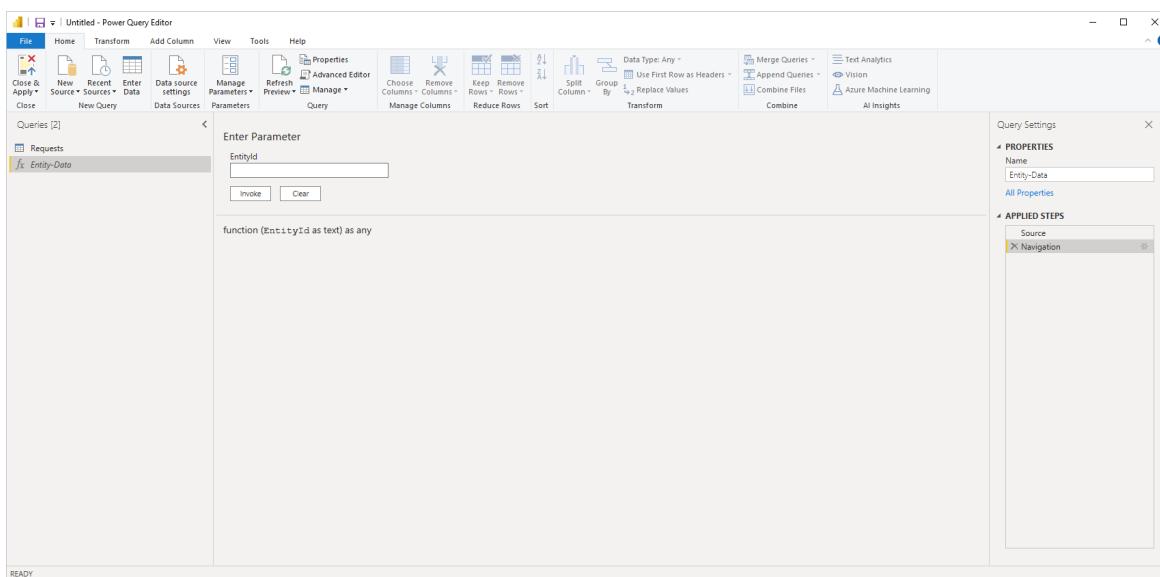
NOTE

You can generate an authentication token for reports using the configuration option in Automy.

5. In the **Navigator** dialog box, select the Automy tables you want. You can then either load or transform the data.



If you're selecting functions, be sure to select **Transform Data** so that you can add parameters to the functions you've selected. More information: [Using parameters](#)



Limitations and issues

Users should be aware of the following limitations and issues associated with accessing Automy Data Analytics

data.

- Automy Data Analytics has a built-in limit of 100,000 rows returned per connection.
- The default rate limit for an Automy Data Analytics Company is 120 requests per minute per user.

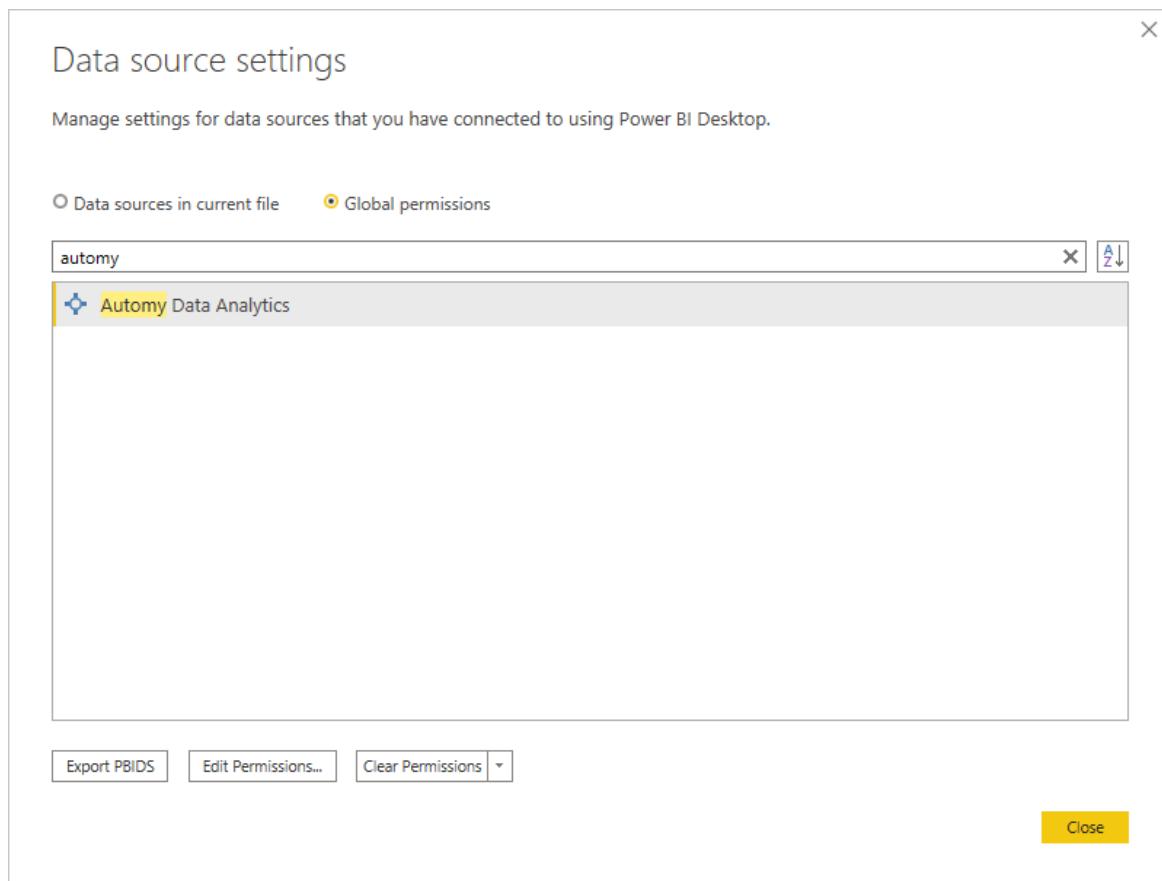
Import from Automy Data Analytics will stop and display an error message whenever the Automy Data Analytics connector reaches any of the limits listed above.

For more guidelines on accessing Automy Data Analytics, contact support@automy.global.

Credential error in the Navigator

If a credential error occurs in the Navigator, clear your recent data source settings.

1. In Power BI Desktop, select **File > Data source settings**.



2. Select the data source, and then select **Clear permissions**. Establish the connection to the navigation again.

Azure Cosmos DB v2 (Beta)

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Feed Key

NOTE

The Azure Cosmos DB V2 connector release has been delayed. We recommend continuing to use the Azure Cosmos DB V1 connector.

Prerequisites

- An [Azure Cosmos DB](#) account

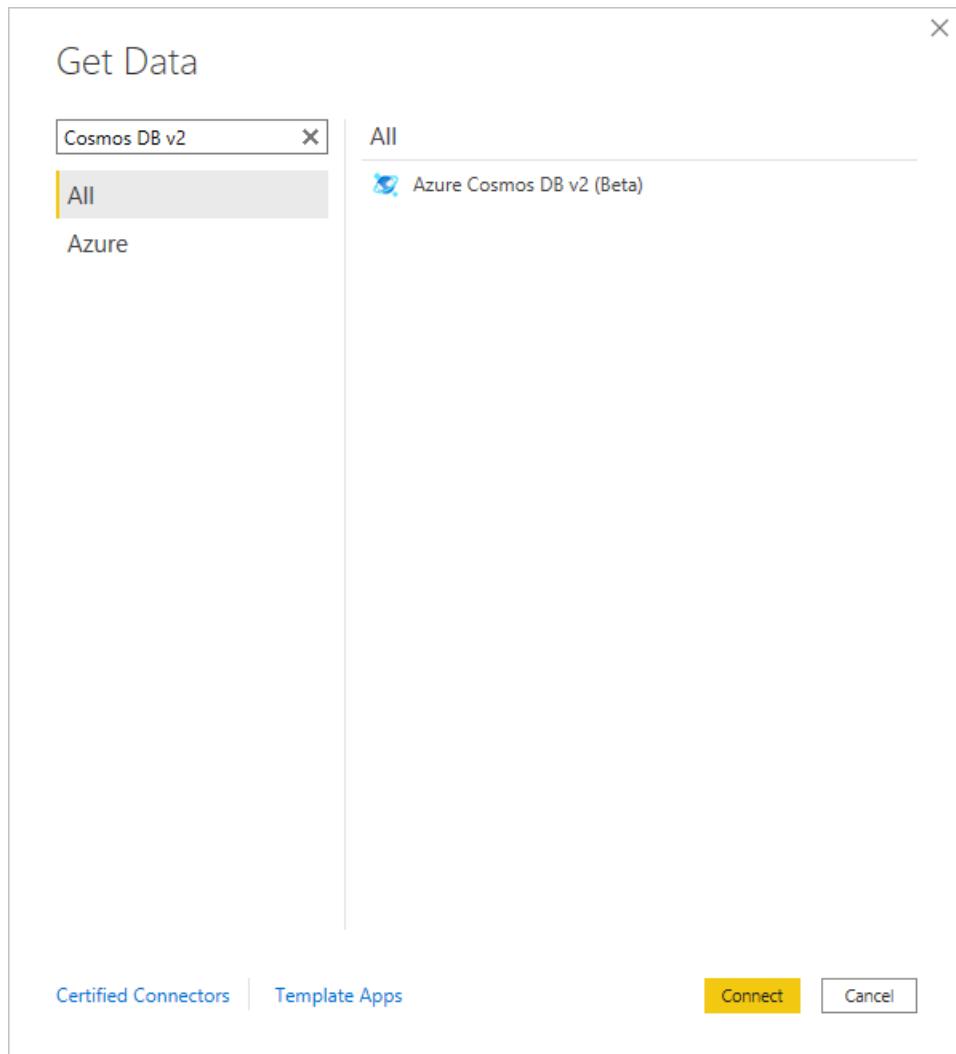
Capabilities supported

- Import
- DirectQuery

Connect to Azure Cosmos DB

To connect to Azure Cosmos DB data:

1. Launch Power BI Desktop.
2. In the **Home** tab, select **Get Data**.
3. In the search box, enter **Cosmos DB v2**.
4. Select **Azure Cosmos DB v2**, and then select **Connect**.



5. On the **Azure Cosmos DB v2** connection page, for **Cosmos Endpoint**, enter the URI of the Azure Cosmos DB account that you want to use. For **Data Connectivity mode**, choose a mode that's appropriate for your use case, following these general guidelines:

- For smaller datasets, choose **Import**. When using import mode, Power BI works with Cosmos DB to import the contents of the entire dataset for use in your visualizations.

NOTE

For **Import** mode to be setup properly, you need to have both the **Advanced Passdown** and **PBI Mode** advanced options set to **0**. More information: [Connect using advanced options](#)

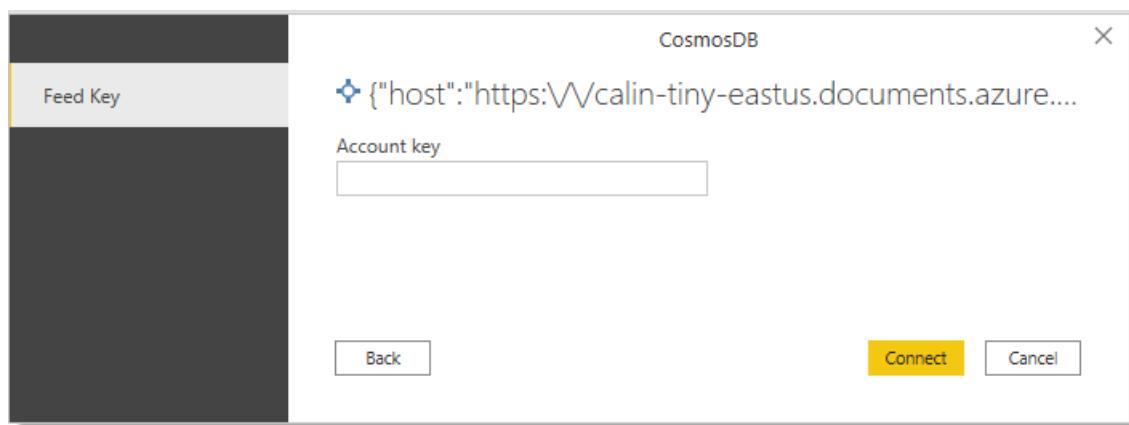
- For larger datasets, choose **DirectQuery**. In DirectQuery mode, no data is downloaded to your workstation. While you create or interact with a visualization, Microsoft Power BI works with Cosmos DB to dynamically query the underlying data source so that you're always viewing current data. More information: [Use DirectQuery in Power BI Desktop](#)

NOTE

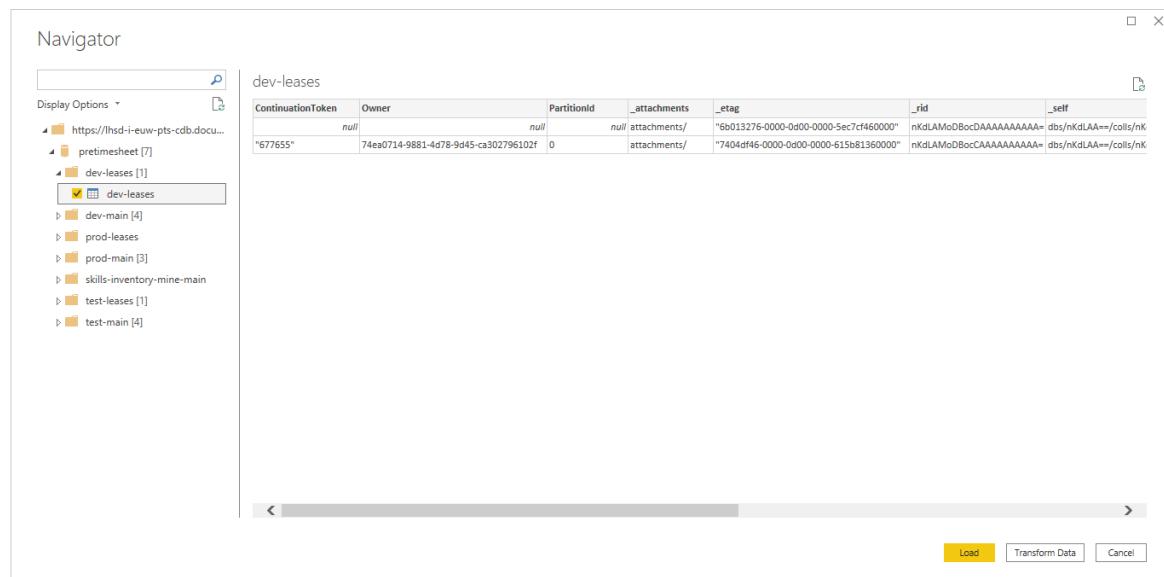
For **DirectQuery** mode to be setup properly, you need to have both the **Advanced Passdown** and **PBI Mode** advanced options set to **1**. More information: [Connect using advanced options](#)



6. Select OK.
 7. At the prompt to configure data source authentication, enter the Account Key. Then select Connect.



Your data catalog, databases, and tables appear in the **Navigator** dialog box.



8. In the **Display Options** pane, select the check box for the dataset that you want to use.
 9. If you want to transform the dataset before you import it, go to the bottom of the dialog box and select **Transform Data**. This selection opens the Power Query editor so that you can filter and refine the set of data you want to use. Also, you could fine-tune the connector options by modifying the passed arguments.

The screenshot shows the Power Query Advanced Editor window. The title bar says "Advanced Editor". The main area contains the following M code:

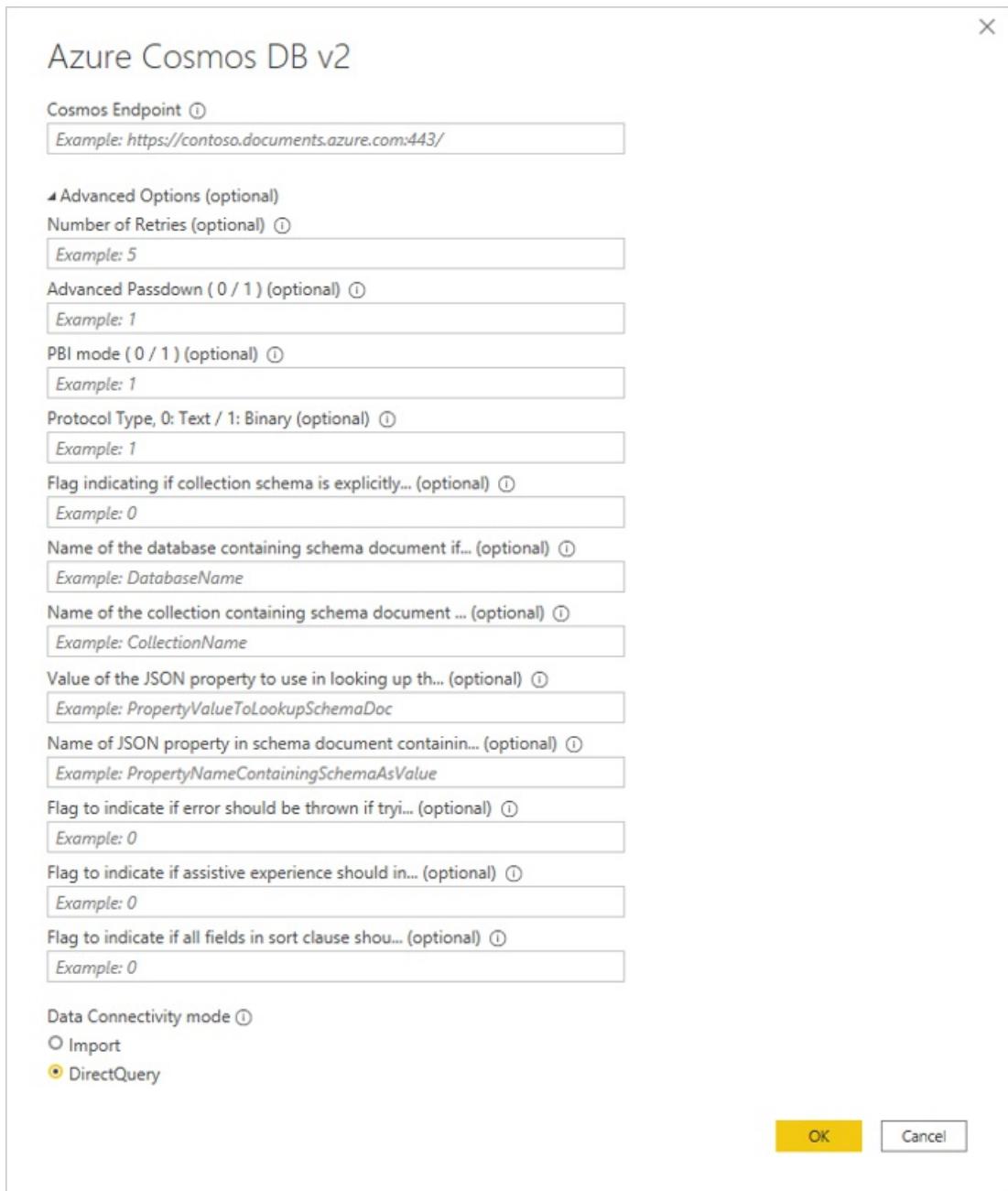
```
let
    Source = CosmosDB.Contents( "https://calin-tiny-eastus.documents.azure.com:443/",
        [ NUMBER_OF_RETRIES="5",
            ADVANCED_PASSDOWN=null,
            PBI_MODE=null,
            PROTOCOL=null,
            SCHEMA_IN_COLLECTION=null,
            DB_WITH_SCHEMA=null,
            COLL_WITH_SCHEMA=null,
            KC=null,
            KV=null,
            SC=null,
            TREAT_FULL_SORTING_MISSES_AS_ERRORS=null,
            REPORT_DEVELOPER_MODE_ON=null,
            FULL_SORTING_ON=null
        ],
        calin_tiny_db_Database = Source{[Name="calin_tiny_db",Kind="Database"]}[Data],
        calin_tiny_coll_Schema = calin_tiny_db_Database{[Name="calin_tiny_coll",Kind="Schema"]}[Data],
        calin_tiny_coll_Table = calin_tiny_coll_Schema{[Name="calin_tiny_coll",Kind="Table"]}[Data]
    in
        calin_tiny_coll_Table
```

Below the code, a green checkmark icon indicates "No syntax errors have been detected." At the bottom right are "Done" and "Cancel" buttons.

10. Otherwise, select **Load**. After the load is complete, you can create visualizations. If you selected **DirectQuery**, Power BI issues a query to Cosmos DB for the visualization that you requested.

Connect using advanced options

Power Query Desktop provides a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
Number of Retries	How many times to retry if there are HTTP return codes of 408 - Request Timeout, 412 - Precondition Failed, or 429 - Too Many Requests. The default number of retries is 5.
Advanced Passdown	Attempt to pass down whenever possible. Set to 0 for false or 1 for true. The default value is 1.
PBI mode	Indicates whether the ODBC Driver's behavior is tailored towards the PBI flow support. Set to 0 for false or 1 for true. The default value is 1.
Protocol Type	The format of the data exchanged with Cosmos DB. Set to 0 for text or 1 for binary data. The default value is 1.

ADVANCED OPTION	DESCRIPTION
Flag indicating if collection schema is explicitly stated as a document	More information: Schema in a document
Name of the database containing schema document if explicitly specified	More information: Schema in a document
Name of the collection containing schema document if explicitly specified	More information: Schema in a document
Name of JSON property to use in looking up the schema document	More information: Schema in a document
Value of the JSON property to use in looking up the schema document	More information: Schema in a document
Name of JSON property in schema document containing the collection schema	More information: Schema in a document
Flag to indicate if error should be thrown if trying to sort more columns than composite index limit	Detects whether the target collection has a Composite Index matching the Sorted Sequence of Columns. Default value is 1 (true).
Flag to indicate if assistive experience should interject if optimal composite indices aren't defined for Sort Passdown	When detecting an error in the six Schema in a document options, prompt whether the JSON of the Composite Index definition will be copied into the clipboard. The contents of the clipboard could then be pasted in the composite index definition in the Cosmos DB Portal. Use this option in the development phase. Default value is 0 (false).
Flag to indicate if all fields in sort clause should be passed down	Indicates if all fields in the sort clause should be passed down. Otherwise, only the field sorted on in a Power BI report or the first field specified in M will be passed down as an optimization. Sorting depends on the composite indexes defined for the collection. Currently the Cosmos DB containers have a maximum of eight composite indexes that can be defined. Default value is 0 (false).
Rest API Version	Sets the REST API version to use. Possible values are <code>2015-12-16</code> or <code>2018-12-31</code> . The default value is <code>2018-12-31</code> . This value can only be set in an advanced query.

Schema in a document

NOTE

Currently, this section contains preliminary information. Additional information will be added before the connector is officially released.

Flag indicating if collection schema is explicitly stated as a document (default 0, that is, no schema as document)

Azure Cosmos DB v2

Flag indicating if collection schema is explicitly... (optional) 1

Name of the database containing schema document if... (optional) calin_tiny_db

Name of the collection containing schema document ... (optional) calin_tiny_coll

Name of JSON property to use in looking up the sch... (optional) id

Value of the JSON property to use in looking up th... (optional) schemaSpecItem

Name of JSON property in schema document containin... (optional) value

```

    {
      "id": "schemaSpecItem",
      "DeviceId": "schemaSpecItem",
      "value": {
        "SchemaMapVersion": "3.0.0",
        "DataSourceSchemaMapVersion": "",
        "DataSource": "DocumentDB",
        "Delimiter": "~~~SchemaMap",
        "SourceTypeMapping": {
          "1": [
            "1"
          ]
        }
      }
    }
  
```

- Name of the database containing schema document if explicitly specified
- Name of the collection containing schema document if explicitly specified
- Name of JSON property to use in looking up the schema document
- Value of the JSON property to use in looking up the schema document
- Name of JSON property in schema document containing the collection schema

Instructions, limitations, and known issues

You should be aware of the following instructions, limitations, and known issues associated with accessing the current version of Azure Cosmos DB v2 data.

Instructions

- When using this connector in import mode, set both **Advanced Passdown** and **PBI Mode** to 0 (`ADVANCED_PASSDOWN="0", PBI_MODE="0"`).
- Don't publish reports on Power BI service that have the Report Developer Mode enabled (`REPORT_DEVELOPER_MODE_ON="1"`).
- Use the following best practices when working with new large collections in DirectQuery mode:
 - Temporarily enable the Report Developer Mode (`REPORT_DEVELOPER_MODE_ON="1"`). Enabling this mode allows the discovery of the Data Layout by loading a very small dataset from Cosmos DB.
 - Once the collection is prepared with the necessary Composite Indexes and the useful Data Engineering aspects related to the Data Shapes are determined, you can switch back to the Regular Mode (`REPORT_DEVELOPER_MODE_ON="0"`) and start the Data Engineering activities targeting the Datasets in their entirety.

Limitations

- Reports must be filtered on Partition Keys defined on the underlying Cosmos DB Container.
- If you need to sort on more than one column (`FULL_SORTING_ON="1"`), you need to consider that the sorting will be delegated to Cosmos DB, which doesn't sort on fields that aren't part of Composite Indexes.
- To assist with the creation of the necessary Composite Indexes, while designing the report in PBI Desktop, the Report Developer Mode needs to be enabled (`REPORT_DEVELOPER_MODE_ON="1"`), which will prompt to Copy to

Clipboard the JSON text that could be pasted in the Cosmos DB Portal when specifying the Cosmos DB Collection Composite Index.

Known issues in DirectQuery mode

- Reports with more than eight columns won't work in DirectQuery mode.
- Aggregate functions aren't passed down. The effect is that SQL expressions Passing Down COUNT, SUM, and so on, will fail and not display a number.

Azure SQL database

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release state	General Availability
Products supported	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication types supported	Windows (Power BI Desktop, Excel, Power Query Online with gateway) Database (Power BI Desktop, Excel) Microsoft Account (all) Basic (Power Query Online)
Function reference docs	Sql.Database Sql.Databases

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

By default, Power BI installs an OLE DB driver for Azure SQL database. However, for optimal performance, we recommend that the customer installs the [SQL Server Native Client](#) before using the Azure SQL database connector. SQL Server Native Client 11.0 and SQL Server Native Client 10.0 are both supported in the latest version.

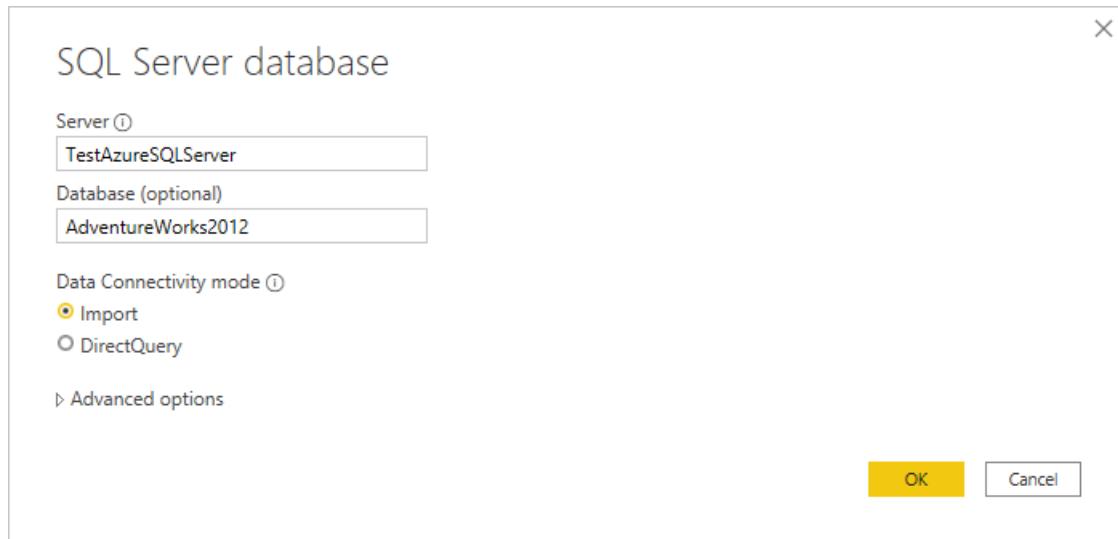
Capabilities supported

- Import
- DirectQuery (Power BI only)
- Advanced options
 - Command timeout in minutes
 - Native SQL statement
 - Relationship columns
 - Navigate using full hierarchy
 - SQL Server failover support

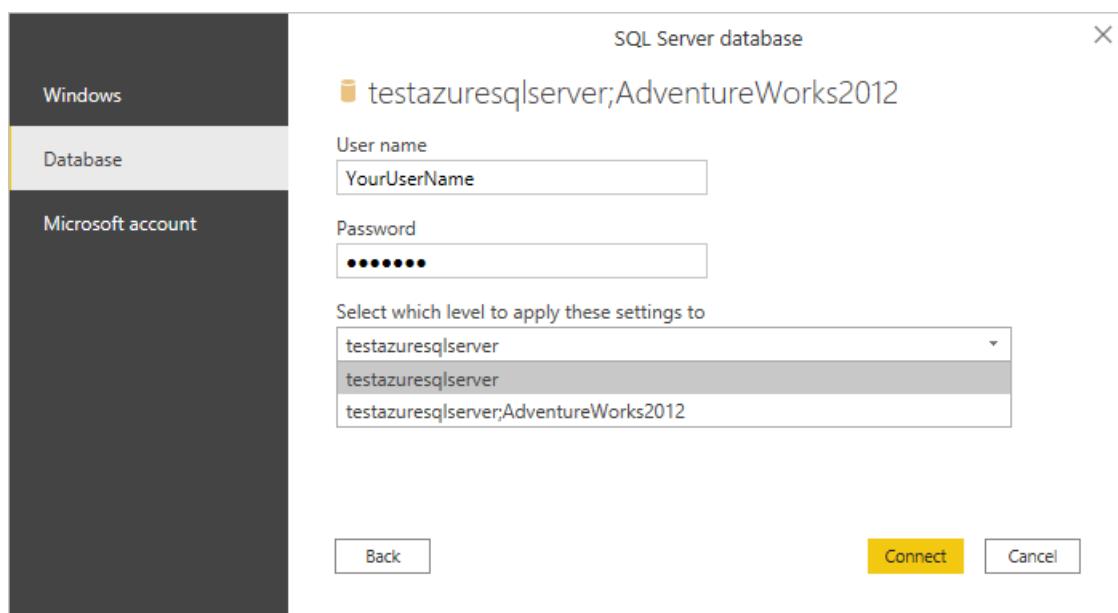
Connect to Azure SQL database from Power Query Desktop

To connect to an Azure SQL database from Power Query Desktop, take the following steps:

1. Select the **Azure SQL database** option in the connector selection.
2. In **SQL Server database**, provide the name of the server and database (optional).



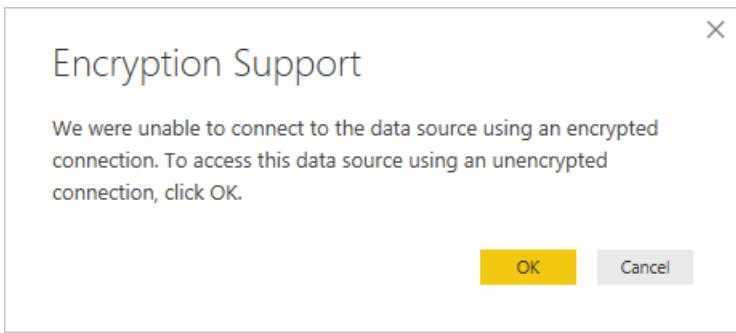
3. Select either the **Import** or **DirectQuery** data connectivity mode.
4. Optionally, you can select and enter advanced options that will modify the connection query, such as a command timeout or a native query (SQL statement). For information: [Connect using advance options](#)
5. Select **OK**.
6. If this is the first time you're connecting to this database, select the authentication type, input your credentials, and select the level to apply the authentication settings to. Then select **Connect**.



For more information about authentication methods, go to [Authentication with a data source](#).

NOTE

If the connection is not encrypted, you'll be prompted with the following message.



Select **OK** to connect to the database by using an unencrypted connection, or follow the instructions in [Enable encrypted connections to the Database Engine](#) to set up encrypted connections to Azure SQL database.

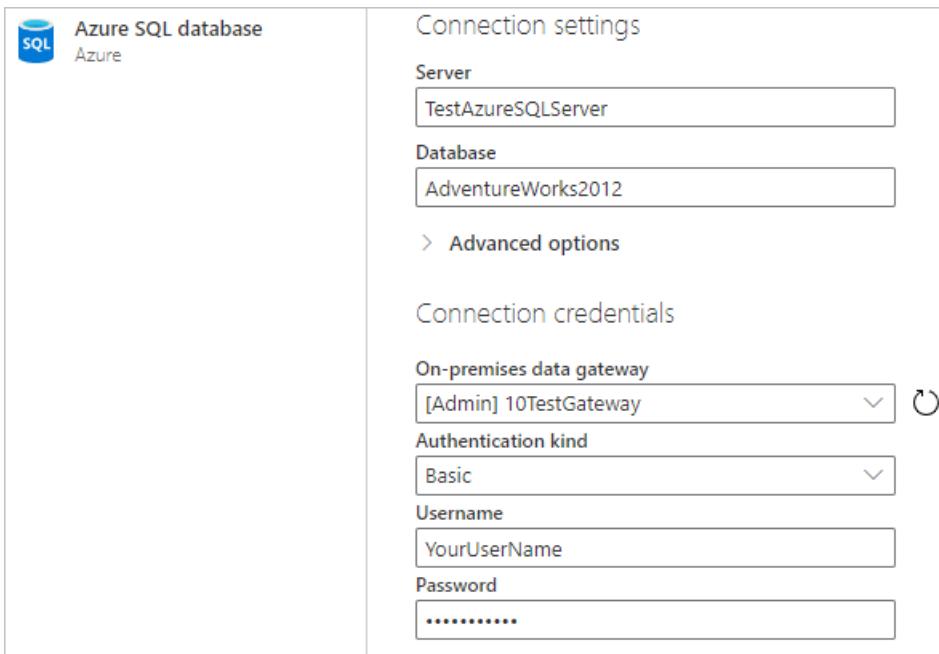
7. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

A screenshot of the Microsoft Power Query Editor interface. On the left, the "Navigator" pane shows a tree view of available tables. The "HumanResources.Employee" table is selected, indicated by a checked checkbox next to its name. Other tables listed include Sales.vStoreWithContacts, Sales.vStoreWithDemographics, AWBuildVersion, DatabaseLog, ErrorLog, HumanResources.Department, HumanResources.Employee, HumanResources.EmployeeDepartment, HumanResources.EmployeePayHistory, HumanResources.JobCandidate, HumanResources.Shift, Person.Address, and Person.AddressType. On the right, the main area displays the "HumanResources.Employee" table data. The columns are: BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, and JobTitle. The data consists of 15 rows of employee information. At the bottom of the editor are three buttons: "Select Related Tables", "Load" (highlighted in yellow), "Transform Data", and "Cancel".

Connect to Azure SQL database from Power Query Online

To connect to an Azure SQL database from Power Query Online, take the following steps:

1. Select the **Azure SQL database** option in the connector selection.
2. In **Azure SQL database**, provide the name of the server and database.



You can also select and enter advanced options that will modify the connection query, such as a command timeout or a native query (SQL statement). More information: [Connect using advanced options](#)

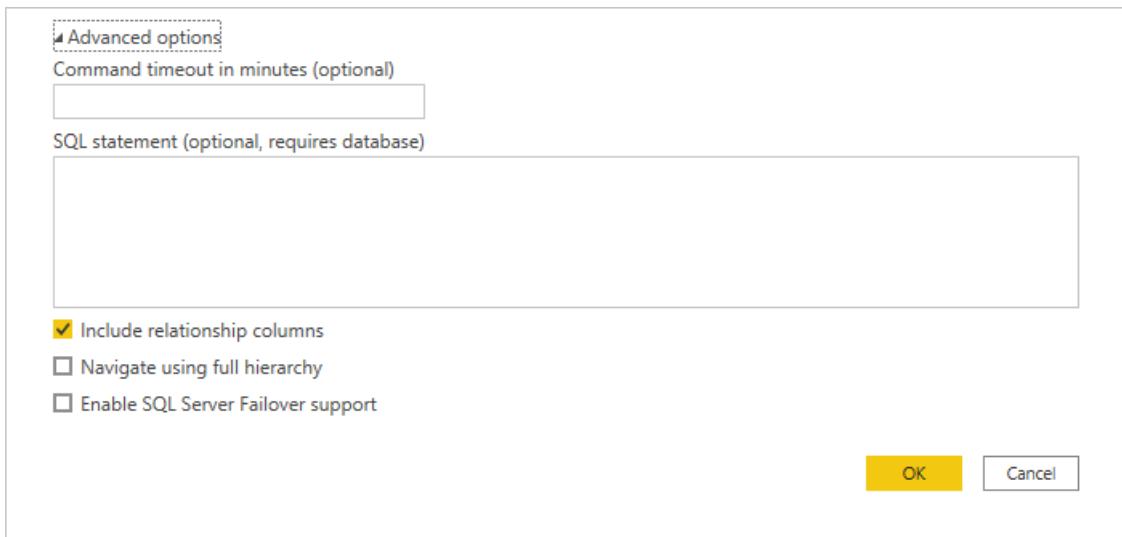
3. If this is the first time you're connecting to this database, select the authentication kind and input your credentials.
4. If necessary, select the name of your on-premises data gateway.
5. If the connection is not encrypted, clear the **Use Encrypted Connection** check box.
6. Select **Next** to continue.
7. In **Navigator**, select the data you require, and then select **Transform data**.

The screenshot shows the 'Power Query - Choose data' dialog. On the left, there's a tree view of available tables: Sales.vSalesPersonSales..., Sales.vStoreWithAddress..., Sales.vStoreWithContacts..., Sales.vStoreWithDemog..., AWBuildVersion, DatabaseLog, ErrorLog, HumanResources.Depar..., HumanResources.Employ..., HumanResources.Employ..., HumanResources.Employ..., and HumanResources.JobCa... . The 'HumanResources.Employee' table is selected. On the right, the table data is displayed in a grid format with columns: BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, and Job. The data consists of 15 rows of employee information. At the bottom, there are 'Back', 'Cancel', and 'Transform data' buttons.

BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	Job
1	295847284	adventure-works\ken0		null	null Chie
2	245797967	adventure-works\terri0	/1/	1 Vice	
3	509647174	adventure-works\roberto0	/1/1/	2 Engi	
4	112457891	adventure-works\rob0	/1/1/1/	3 Seni	
5	695256908	adventure-works\gail0	/1/1/2/	3 Desi	
6	998320692	adventure-works\jossef0	/1/1/3/	3 Desi	
7	134969118	adventure-works\dylan0	/1/1/4/	3 Rest	
8	811994146	adventure-works\diane1	/1/1/4/1/	4 Rest	
9	658797903	adventure-works\gigi0	/1/1/4/2/	4 Rest	
10	879342154	adventure-works\michael6	/1/1/4/3/	4 Rest	
11	974026903	adventure-works\ovidiu0	/1/1/5/	3 Seni	
12	480168528	adventure-works\thierry0	/1/1/5/1/	4 Tool	
13	486228782	adventure-works\janice0	/1/1/5/2/	4 Tool	
14	42487730	adventure-works\michael8	/1/1/6/	3 Seni	
15	56920285	adventure-works\sharon0	/1/1/7/	3 Desi	

Connect using advanced options

Both Power Query Desktop and Power Query Online provide a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop and Power Query Online.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.
Enable SQL Server Failover support	If checked, when a node in the Azure SQL failover group isn't available, Power Query moves from that node to another when failover occurs. If cleared, no failover occurs.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your Azure SQL database.

Troubleshooting

Always Encrypted columns

Power Query doesn't support 'Always Encrypted' columns.

Azure Synapse Analytics (SQL DW)

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Windows (Power BI Desktop, Excel, online service with gateway) Database (Power BI Desktop, Excel) Microsoft Account (all) Basic (online service)
Function Reference Documentation	Sql.Database Sql.Databases

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

By default, Power BI installs an OLE DB driver for Azure Synapse Analytics (SQL DW). However, for optimal performance, we recommend that the customer installs the [SQL Server Native Client](#) before using the Azure Synapse Analytics (SQL DW) connector. SQL Server Native Client 11.0 and SQL Server Native Client 10.0 are both supported in the latest version.

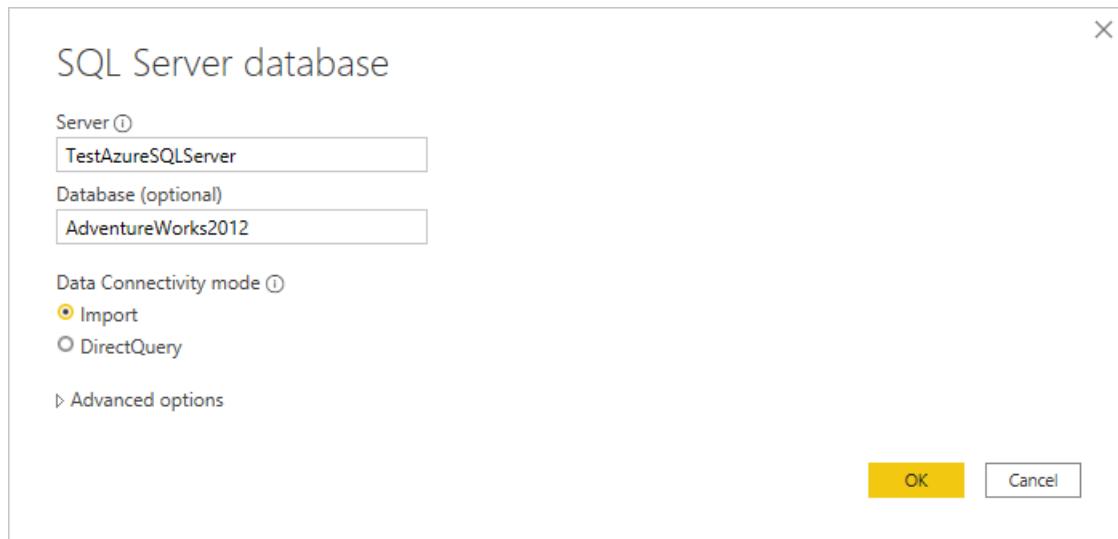
Capabilities Supported

- Import
- DirectQuery (Power BI only)
- Advanced options
 - Command timeout in minutes
 - Native SQL statement
 - Relationship columns
 - Navigate using full hierarchy
 - SQL Server failover support

Connect to Azure Synapse Analytics (SQL DW) from Power Query Desktop

To make the connection from Power Query Desktop:

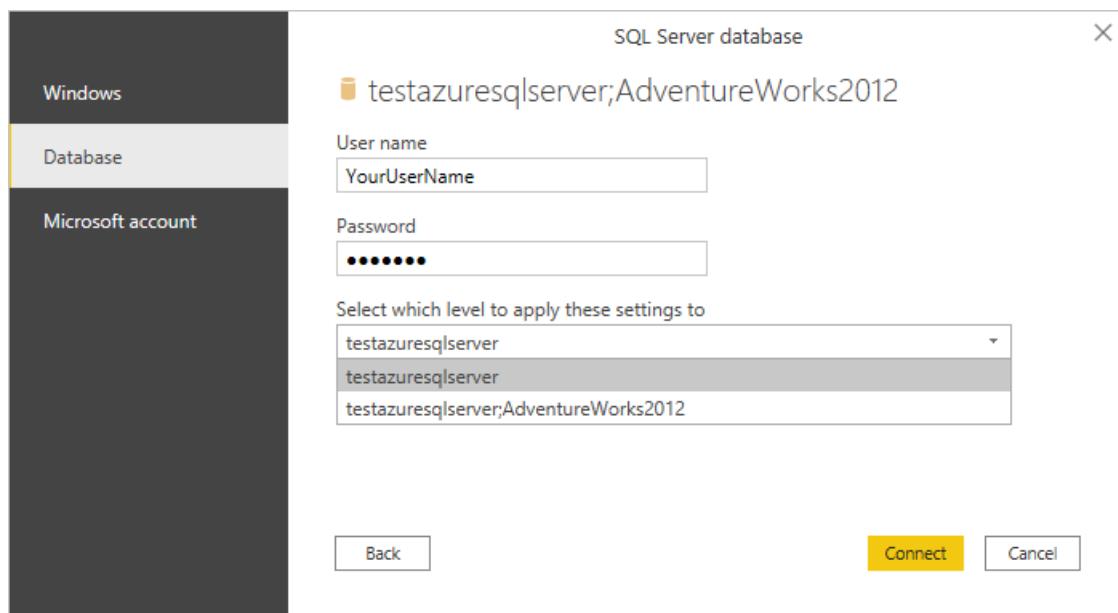
1. Select the **Azure Synapse Analytics (SQL DW)** option in the connector selection.
2. In the **SQL Server database** dialog that appears, provide the name of the server and database (optional). In this example, `TestAzureSqlServer` is the server name and `AdventureWorks2012` is the database.



3. Select either the **Import** or **DirectQuery** data connectivity mode.

You can also select and enter advanced options that will modify the connection query, such as a command timeout or a native query (SQL statement). More information: [Connect using advanced options](#)

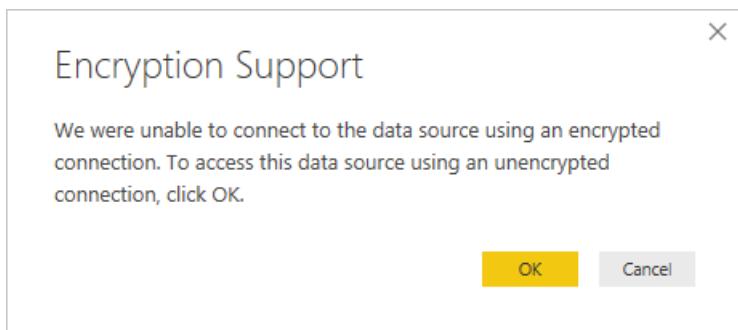
4. Select **OK**.
5. If this is the first time you're connecting to this database, select the authentication type, input your credentials, and select the level to apply the authentication settings to. Then select **Connect**.



For more information about authentication methods, go to [Authentication with a data source](#).

NOTE

If the connection is not encrypted, you'll be prompted with the following dialog.



Select **OK** to connect to the database by using an unencrypted connection, or follow the instructions in [Enable encrypted connections to the Database Engine](#) to set up encrypted connections to Azure Synapse Analytics (SQL DW).

6. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

Connect to Azure Synapse Analytics (SQL DW) from Power Query Online

To make the connection from Power Query Online:

1. Select the **Azure Synapse Analytics (SQL DW)** option in the connector selection.
2. In the **Azure Synapse Analytics (SQL DW)** dialog that appears, provide the name of the server and database (optional). In this example, `TestAzureSQLServer` is the server name and `AdventureWorks2012` is the database.

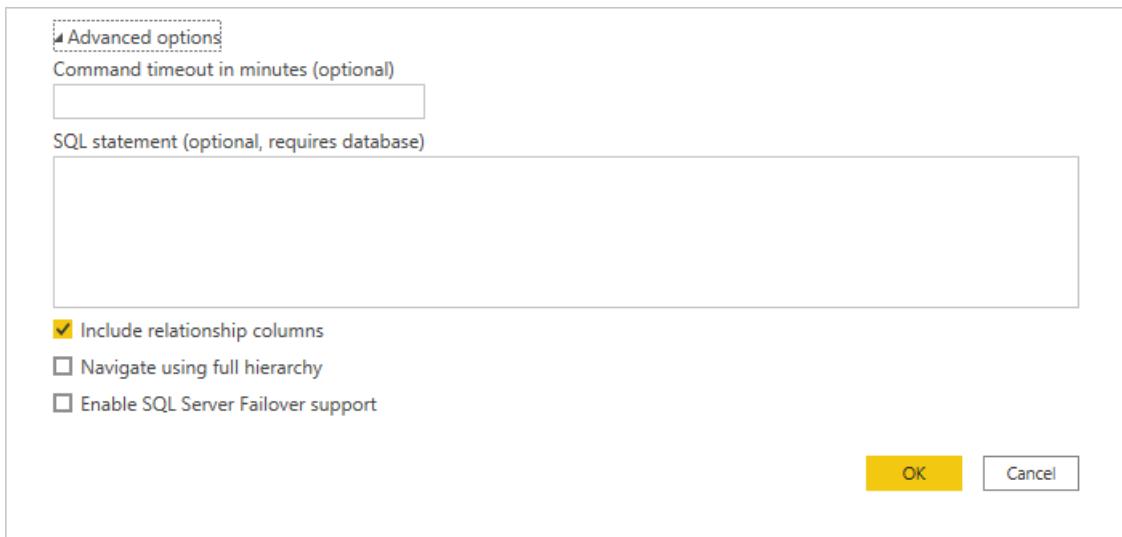
The screenshot shows the 'Azure Synapse Analytics (SQL DW)' connection settings dialog. It includes fields for 'Server' (TestAzureSQLServer), 'Database' (AdventureWorks2012), and a link to 'Advanced options'. Below this is the 'Connection credentials' section, which contains fields for 'Connection' (Create new connection), 'Connection name' (testazuresqlserver;AdventureWorks2012), 'On-premises data gateway' (none), 'Authentication kind' (Basic), 'Username' (empty), 'Password' (empty), 'Privacy Level' (None), and a checked 'Use Encrypted Connection' checkbox.

You can also select and enter advanced options that will modify the connection query, such as a command timeout or a native query (SQL statement). More information: [Connect using advanced options](#)

3. If this is the first time you're connecting to this database, select the authentication kind and input your credentials.
4. If required, select the name of your on-premises data gateway.
5. If the connection is not encrypted, clear the **Use Encrypted Connection** check box.
6. Select **Next** to continue.
7. In **Navigator**, select the data you require, and then select **Transform data**.

Connect using advanced options

Both Power Query Desktop and Power Query Online provide a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop and Power Query Online.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.
Enable SQL Server Failover support	If checked, when a node in the Azure SQL failover group isn't available, Power Query moves from that node to another when failover occurs. If cleared, no failover occurs.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your Azure SQL database.

Troubleshooting

Always Encrypted columns

Power Query doesn't support 'Always Encrypted' columns.

Azure Synapse Analytics workspace (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Public Review
Products	Power BI (Datasets)
Authentication Types Supported	Organizational account Service account

NOTE

This Azure Synapse Analytics workspace connector doesn't replace the Azure Synapse Analytics (SQL DW) connector. This connector makes exploring data in Synapse workspaces more accessible. Some capabilities aren't present in this connector, including native query and DirectQuery support.

NOTE

This connector supports access to all data in your Synapse workspace, including Synapse Serverless, Synapse on-demand, and Spark tables.

Prerequisites

Before you can sign in to Synapse workspaces, you must have access to [Azure Synapse Analytics Workspace](#).

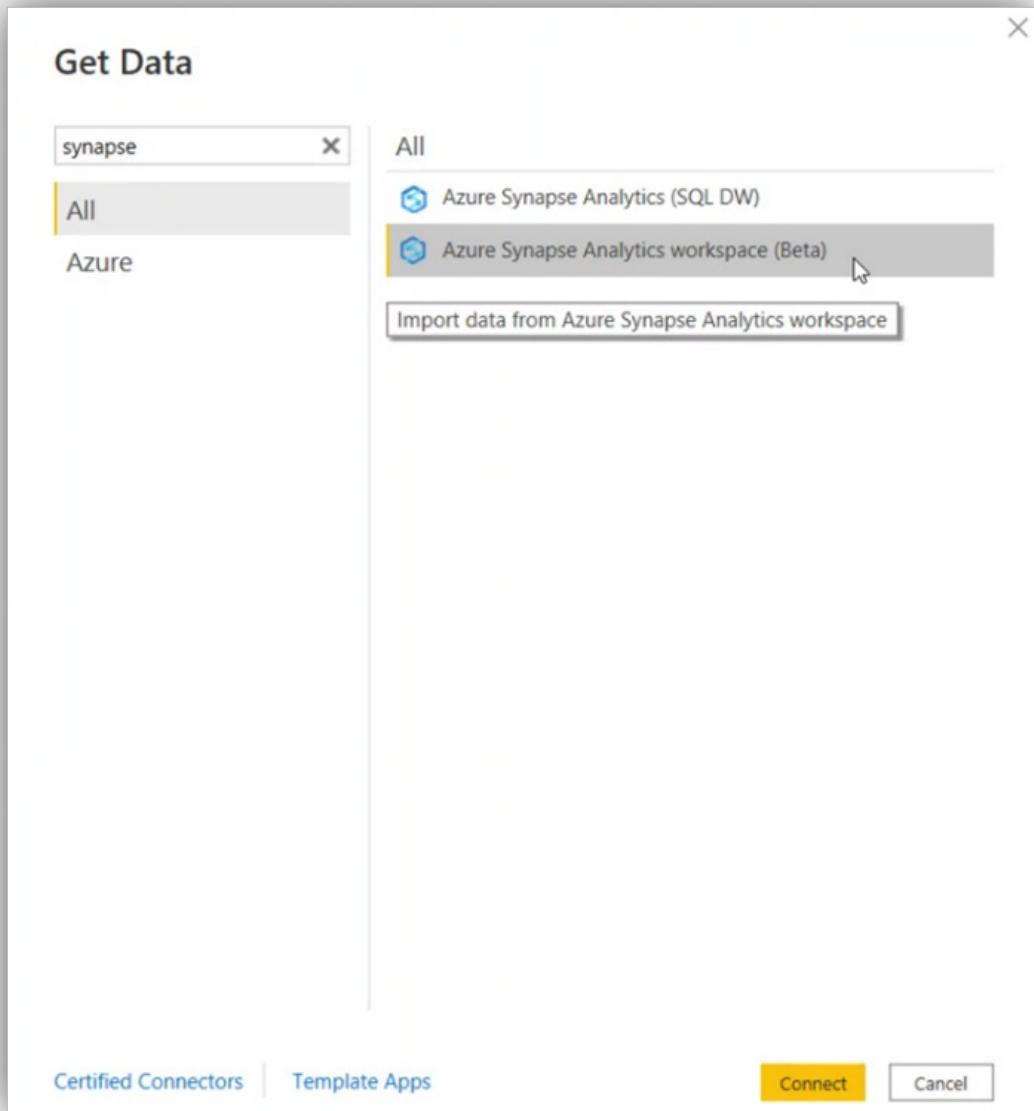
Capabilities Supported

- Import

Connect to Synapse workspace data from Power Query Desktop

To connect to Synapse workspace data:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **Azure Synapse Analytics workspace (Beta)**. Then select **Connect**.



2. To sign in to your Synapse account, select **Sign in**.
3. In the **Sign in with Microsoft** window that appears, provide your credentials to sign in to your Synapse account. Then select **Next**.
4. Once you've successfully signed in, select **Connect**.

Once the connection is established, you'll see a list of the workspaces you have access to. Drill through the workspaces, databases, and tables.

You can **Load** the selected table, which brings the entire table into Power BI Desktop, or you can select **Transform Data** to edit the query, which opens Power Query Editor. You can then filter and refine the set of data you want to use, and then load that refined set of data into Power BI Desktop.

Troubleshooting

I don't see my Synapse workspace in the connector

The Synapse connector is using [Azure role-based access control \(RBAC\)](#) to find the Synapse workspaces you have access to.

If your access is only defined in [Synapse RBAC](#), you might not see the workspace.

Make sure your access is defined by Azure RBAC to ensure all Synapse workspaces are displayed.

Bloomberg Data and Analytics

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Bloomberg, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Bloomberg website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Organizational account

Prerequisites

Your organization must subscribe to Bloomberg PORT Enterprise and you must be a Bloomberg Anywhere user and have a Bloomberg biometric authentication device (B-Unit).

Capabilities Supported

- Import

Connect to Bloomberg Data and Analytics

To connect to Bloomberg Data and Analytics:

1. In Power BI Desktop, select **Home > Get Data**.
2. Select **Other** from the categories on the left, select **Bloomberg Data and Analytics**, and then select **Connect**.

Get Data

- All
- File
- Database
- Power Platform
- Azure
- Online Services
- Other

- Web
- SharePoint list
- OData Feed
- Active Directory
- Microsoft Exchange
- Hadoop File (HDFS)
- Spark
- Hive LLAP
- R script
- Python script
- ODBC
- OLE DB
- B Bloomberg Data and Analytics (Beta)**
- Acterys : Model Automation & Planning (Beta)
- Anaplan Connector (Beta)
- Automation Anywhere (Beta)

[Certified Connectors](#) | [Template Apps](#)

[Connect](#) [Cancel](#)

3. If this is the first time you're connecting to the Bloomberg Data and Analytics connector, a third-party notice will be displayed. Select **Don't warn me again with this connector** if you don't want this message to be displayed again, and then select **Continue**.
4. Enter a Bloomberg Query Language (BQL) query to specify what data you want to get. To learn more about BQL, contact your Bloomberg Sales representative. Select **OK**.

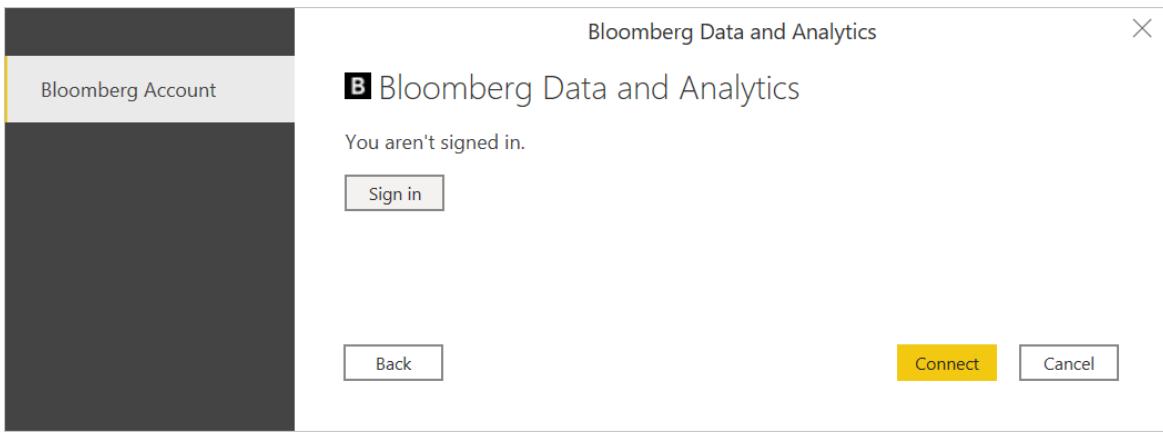
Bloomberg Data and Analytics v1.0.15

BQL Query

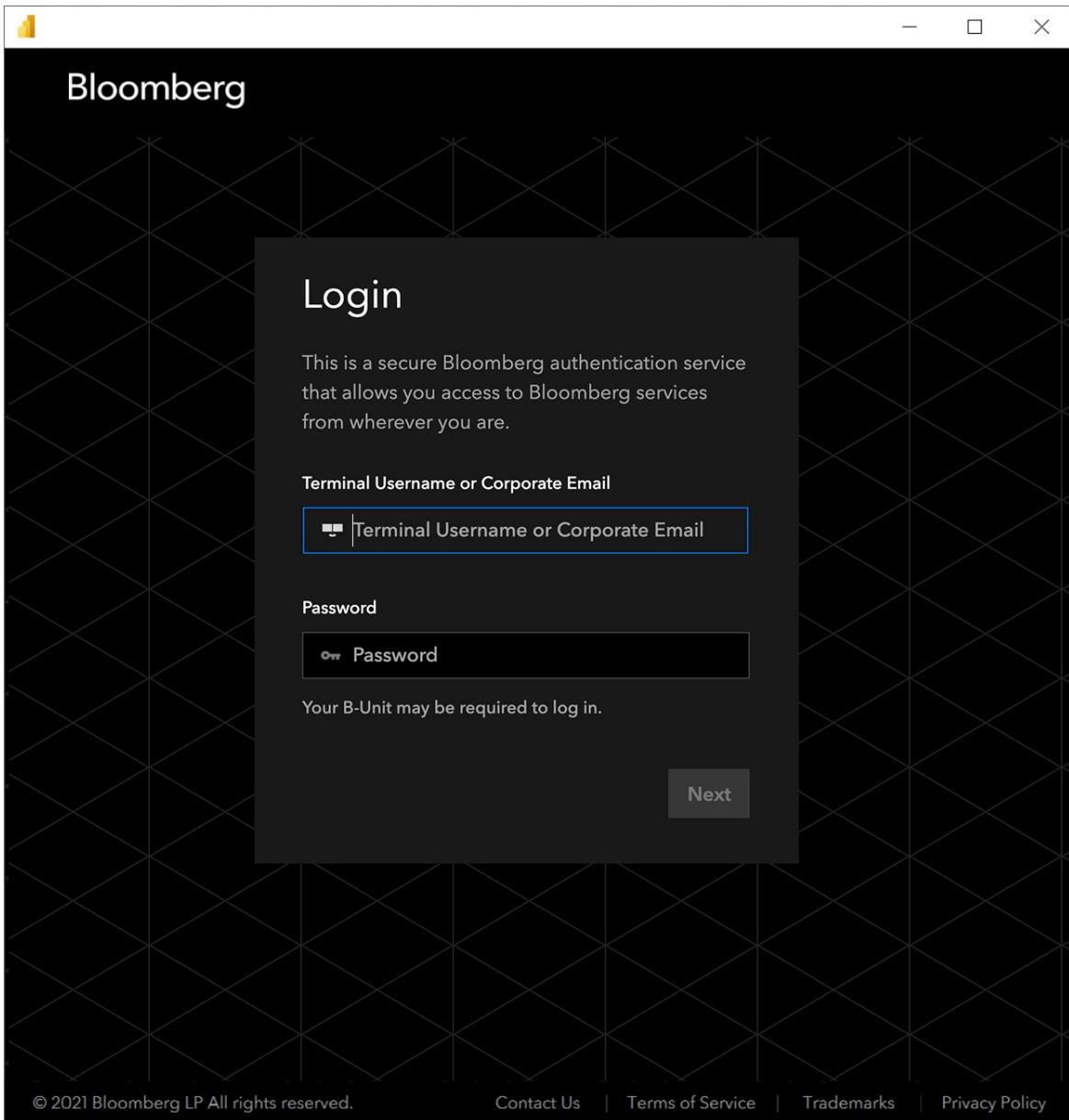
```
let( #Data = port_all( report_name='BIINT_TE', classification_level = 'Portfol
```

[OK](#) [Cancel](#)

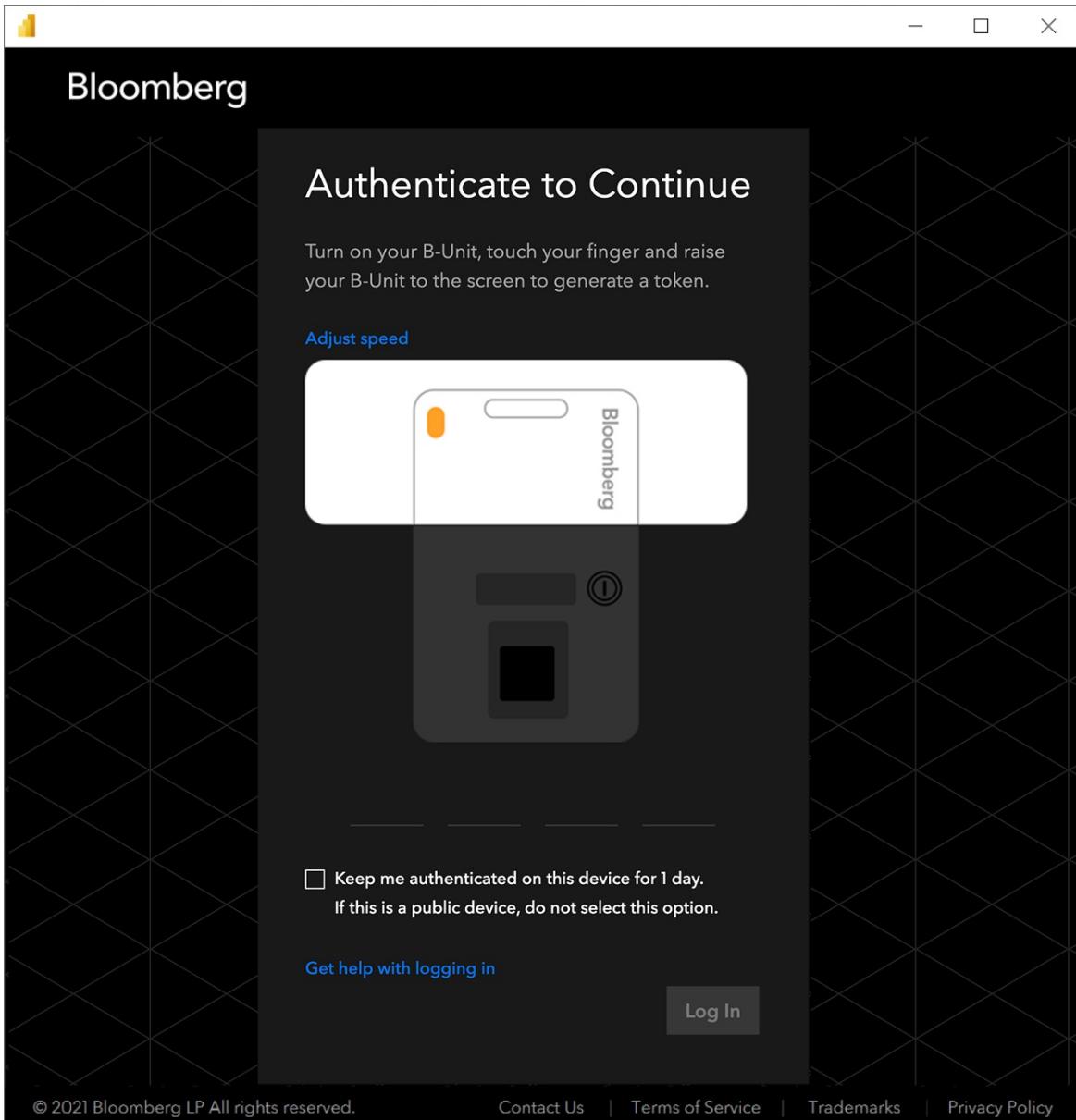
5. To sign in to your Bloomberg account, select **Sign in**.



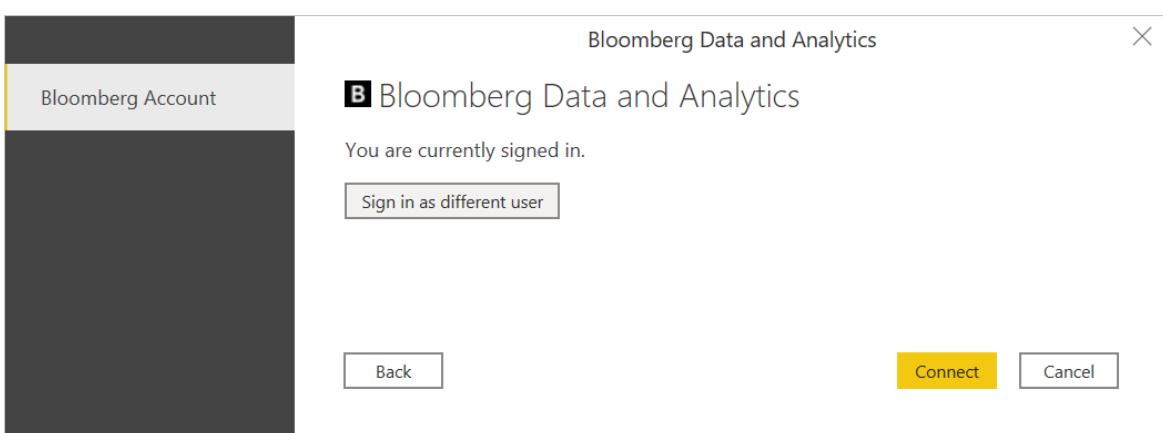
6. In the window that appears, provide your credentials to sign in to your Bloomberg account. If you entered an email address and a password, select **Next**.



7. Enter your B-Unit code and select **Log In**.



8. Once you've successfully signed in, select **Connect**.



Once the connection is established, you will see data available for preview in **Navigator**.

Navigator

Display Options

let(#Data = port_all(report_name='BIINT_TE',cl...)

#Risk

#Weight

#Risk

ID	#Risk
BIINT_PORT_DEMO	5.05964853
BIINT_PORT_DEMO	24.8853001

You can **Load** the selected table, or you can select **Transform Data** to edit the query, which opens Power Query Editor. You can then filter and refine the set of data you want to use, and then load that refined set of data into Power BI Desktop.



BQE Core (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by BQE, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the BQE website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI (Datasets)
Authentication Types Supported	BQE Core Account
Function Reference Documentation	-

Prerequisites

To use the BQE Core Power BI connector, you must have a BQE Core account with username and password.

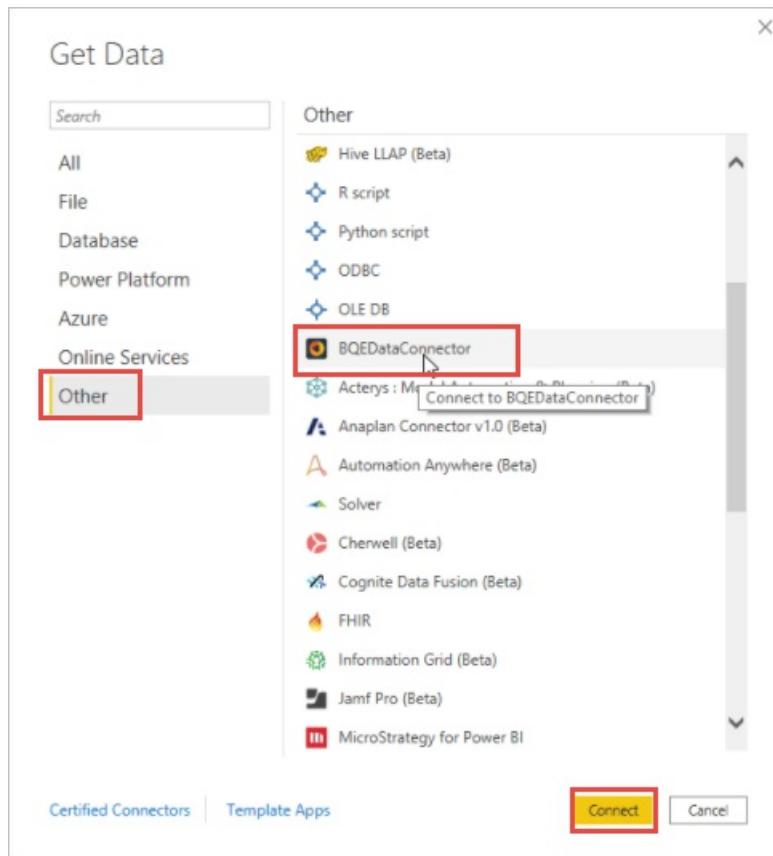
Capabilities supported

- Import

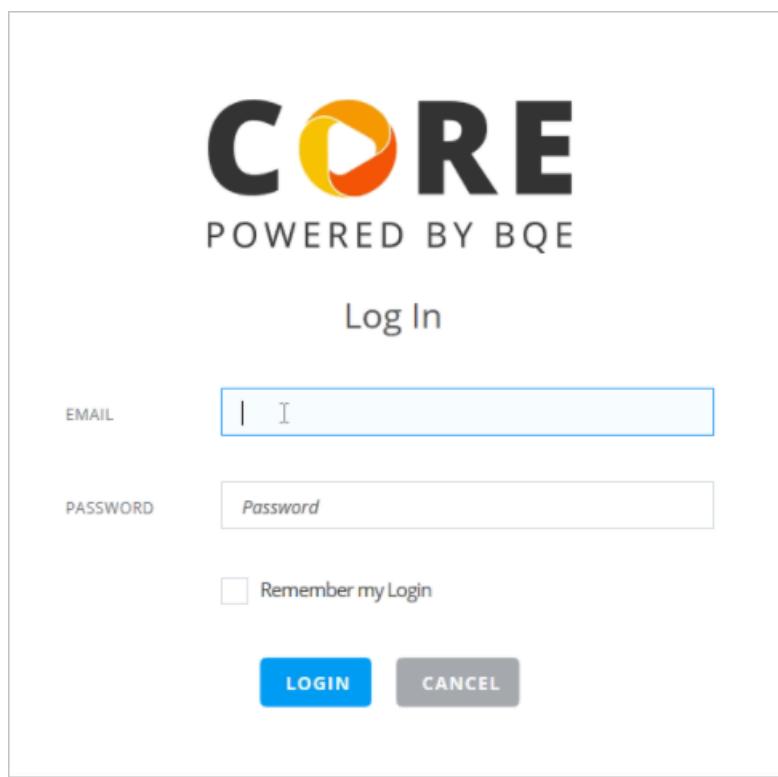
Connect to BQE Core from Power Query Desktop

To connect to BQE Core data:

1. Launch Power BI Desktop and enter the **Get Data** experience.
2. From the **Other** category, select **BQEDataConnector**, and then select **Connect**.



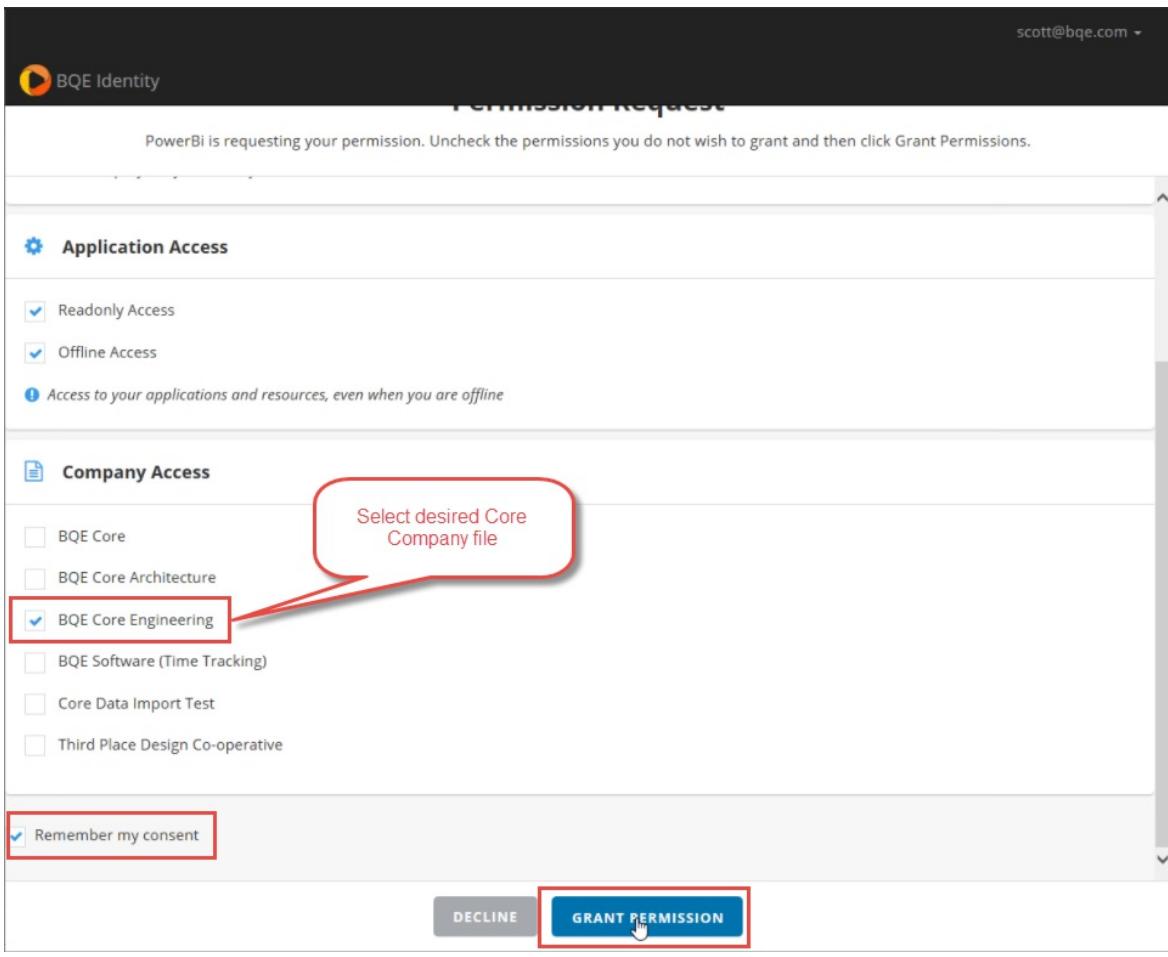
3. Select **Sign In**. You'll be prompted to sign in to Core.



4. In the sign in screen, enter your Core email and password. Select **Login**.

5. You'll then be prompted to select your Core company file.

- a. Select the Core company file you want to use.
- b. (Optional) If you select **Remember my consent**, the next time you connect to this Core company file you won't need to grant permission again.
- c. Select **Grant Permission**.



6. Select **Connect**, and then select a module. For reference, review the API Reference under the [Core API Documentation](#).
7. From the Navigator, select the tables to load, and then select **Transform Data** to transform the data in Power Query.

Dataverse

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Dynamics 365 Customer Insights Power Apps (Dataflows)
Authentication types	Organizational account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

You must have a Dataverse environment with maker permissions to access the portal, and read permissions to access data within tables.

To use the Dataverse connector, the **TDS endpoint** setting must be enabled in your environment. More information: [Manage feature settings](#)

To use the Dataverse connector, TCP ports 1433 and/or 5558 need to be open to connect. If only port 5558 is enabled, you must append that port number to the Dataverse environment URL, such as `yourenvironmentid.crm.dynamics.com:5558`. More information: [SQL Server connection issue due to closed ports](#)

Capabilities supported

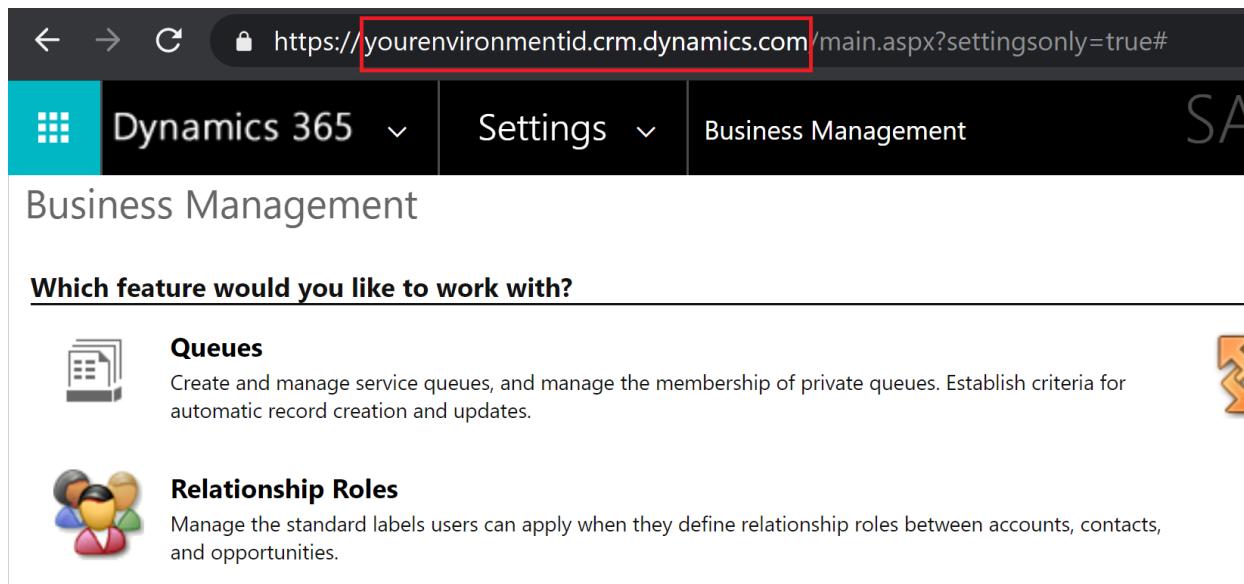
- Server URL
- Advanced
 - Reorder columns
 - Add display column

Finding your Dataverse environment URL

Open [Power Apps](#). In the upper right of the Power Apps page, select the environment you're going to connect to. Select the  settings icon, and then select **Advanced settings**.

In the new browser tab that opens, copy the root of the URL. This root URL is the unique URL for your environment. The URL will be in the format of `https://<yourenvironmentid>.crm.dynamics.com/`. **Make sure you remove https:// and the trailing / from the URL before pasting it to connect to your**

environment. Keep this URL somewhere handy so you can use it later, for example, when you create Power BI reports.



The screenshot shows a browser window with the URL <https://yourenvironmentid.crm.dynamics.com/main.aspx?settingsonly=true#>. The page is titled 'Business Management'. At the top, there's a navigation bar with 'Dynamics 365' and 'Settings'. Below the navigation, the title 'Business Management' is displayed. A section titled 'Which feature would you like to work with?' contains two items: 'Queues' and 'Relationship Roles'. Each item has a small icon and a brief description. The 'Queues' section includes a red box around the URL in the address bar.

Queues
Create and manage service queues, and manage the membership of private queues. Establish criteria for automatic record creation and updates.

Relationship Roles
Manage the standard labels users can apply when they define relationship roles between accounts, contacts, and opportunities.

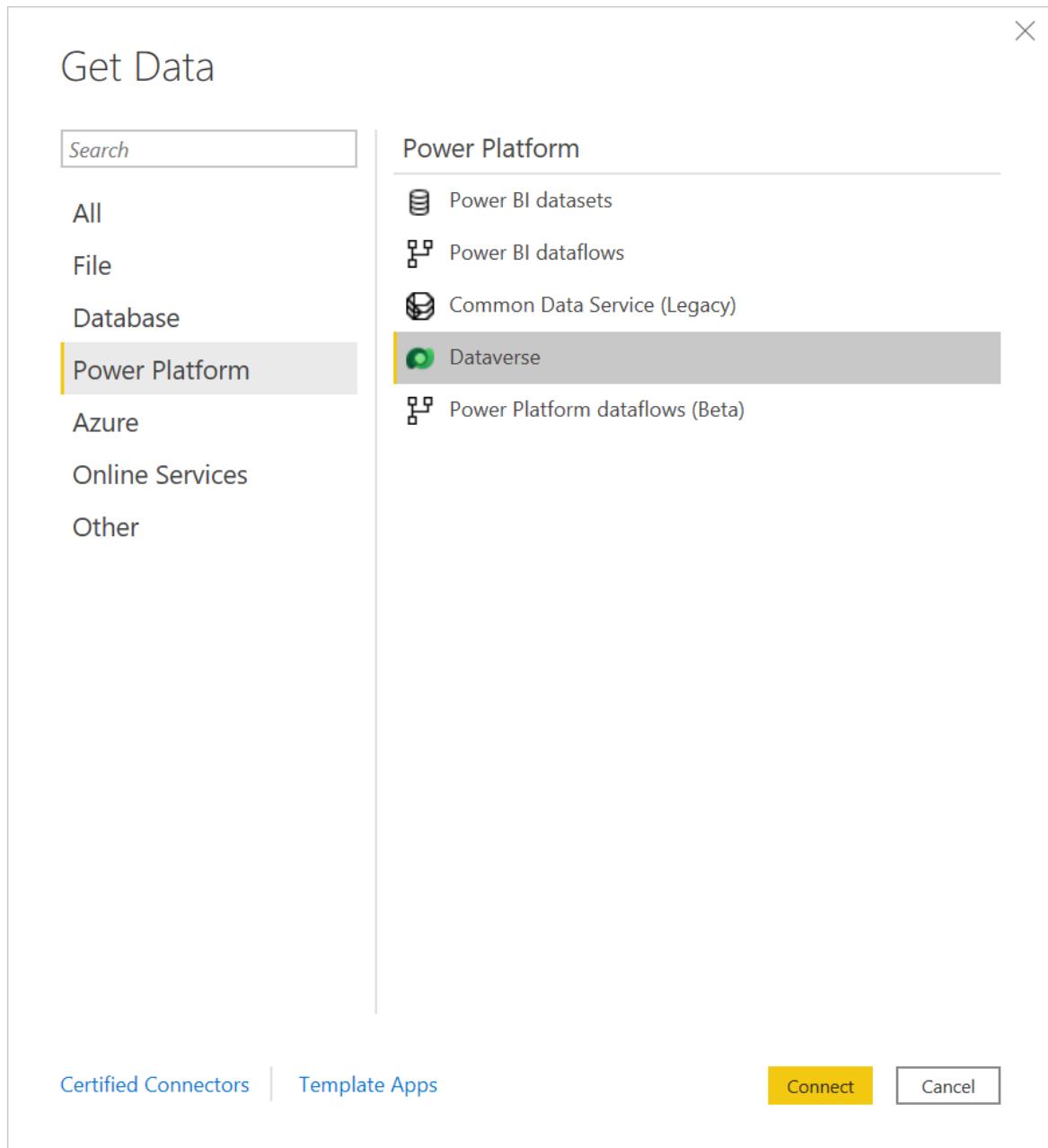
Connect to Dataverse from Power BI Desktop

NOTE

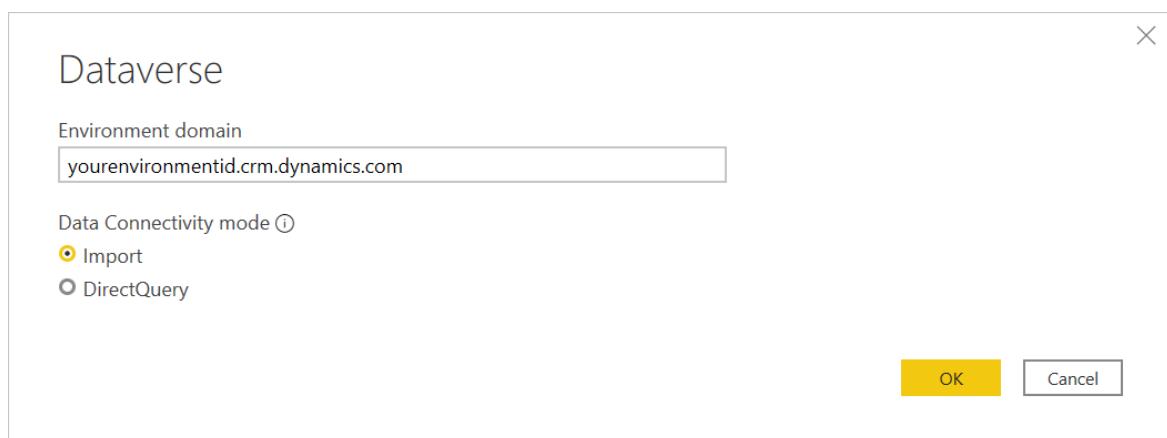
The Power Query Dataverse connector is mostly suited towards analytics workloads, not bulk data extraction. More information: [Alternative Dataverse connections](#)

To connect to Dataverse from Power BI Desktop:

1. Select **Get data** from the **Home** tab.
2. In the **Get Data** dialog box, select **Power Platform > Dataverse**, and then select **Connect**.



3. Enter the Dataverse environment URL of the data you want to load. Use the format <yourenvironmentid>.crm.dynamics.com. Be sure to remove the https:// prefix and / suffix from the URL before entering the name in **Environment domain**. More information: [Finding your Dataverse environment URL](#)



4. Select one of the following Data Connectivity mode options:
- **Import:** We recommend that you import data to Power BI wherever possible. With this mode, data is

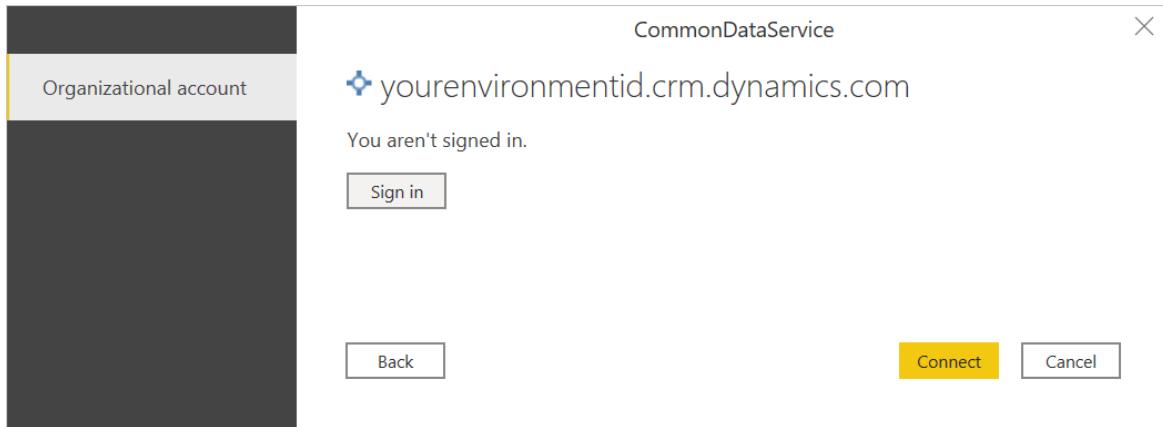
cached in the Power BI service and imported on a scheduled interval.

- **DirectQuery**: Connects directly to the data in Dataverse. Use this mode for real-time data retrieval. This mode can also more strictly enforce the Dataverse security model. More information: [DirectQuery model guidance in Power BI Desktop](#)

When you've finished filling in the information, select **OK**.

5. If this attempt is the first time you're connecting to this site, select **Sign in** and input your credentials.

Then select **Connect**.



6. In **Navigator**, select the data you require, then either load or transform the data.

A screenshot of the Power BI Navigator interface. The left pane shows a tree view of entities from the Dataverse connection, with 'ApplicationUser' selected. The right pane displays a table titled ' ApplicationUser ' showing four rows of data:

	statecode	statecode_display	statuscode	statuscode_display
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active

At the bottom are 'Load' (highlighted in yellow), 'Transform Data', and 'Cancel' buttons.

Connect to Dataverse from Power Query Online

To connect to Dataverse from Power Query Online:

1. From the **Data sources** page, select **Common Data Service (Legacy)**.

Data sourcesCommon Data Service (Legacy)
Power PlatformPower BI dataflows
Power PlatformPower Platform data
Power Platform

2. Enter the server URL address of the data you want to load.

Common Data Service (Legacy)
Power Platform

Connection settings

Server Url *

https://yourenvironmentid.crm.dynamics.com

> Advanced options

Connection credentials

On-premises data gateway

(none)

Authentication kind

Organizational account

You are not signed in. Please sign in.

Sign in

3. If necessary, enter an on-premises data gateway if you're going to be using on-premises data. For example, if you're going to combine data from Dataverse and an on-premises SQL Server database.

4. Sign in to your organizational account.

5. When you've successfully signed in, select **Next**.

6. In the navigation page, select the data you require, and then select **Transform Data**.

Limitations and issues

When to use the Common Data Service (Legacy) connector

Dataverse is the direct replacement for the Common Data Service connector. However, there may be times when it's necessary to choose the Common Data Service (Legacy) connector instead of the Dataverse connector:

- If you're connecting to data using Power Apps, you'll still have to use the Common Data Service (Legacy) connector.
- If you're accessing large datasets that are greater than 80 MB, you'll still have to use the Common Data Service (Legacy) connector.
- If you want paging of the query results and want to build reports that use the image data type, you'll still have to use the Common Data Service (Legacy) connector.

Also, there are certain Tabular Data Stream (TDS) data types that are supported in OData when using Common Data Service (Legacy) that aren't supported in Dataverse. The supported and unsupported data types are listed in [How Dataverse SQL differs from Transact-SQL \(Preview\)](#).

All of these features will be added to the Dataverse connector in the future, at which time the Common Data Service (Legacy) connector will be deprecated.

Dataverse performance and throttling limits

For information about performance and throttling limits for Dataverse connections, go to [Requests limits and allocations](#). These limitations apply to both the Dataverse connector and the [OData Feed](#) connector when accessing the same endpoint.

Table retrieval rate

As a guideline, most default tables will be retrieved at a rate of approximately 500 rows per second using the Dataverse connector. Take this rate into account when deciding whether you want to connect to Dataverse or export to data lake. If you require faster retrieval rates, consider using the Export to data lake feature or Tabular Data Stream (TDS) endpoint. For more information, go to [Alternative Dataverse connections](#).

Alternative Dataverse connections

There are several alternative ways of extracting and migrating data from Dataverse:

- Use the OData connector to move data in and out of Dataverse. For more information on how to migrate data between Dataverse environments using the dataflows OData connector, go to [Migrate data between Dataverse environments using the dataflows OData connector](#).
- Use the **Export to data lake** feature in Power Apps to extract data from Dataverse into Azure Data Lake Storage, which can then be used to run analytics. For more information about the export to data lake feature, go to [Exporting Dataverse data to Azure Data Lake is Generally Available](#).
- Use the Tabular Data Stream (TDS) Protocol endpoint to access read-only data in Dataverse. For more information about this preview feature and a video on how it works, go to [Tabular Data Stream \(TDS\) Protocol endpoint for Dataverse](#).

NOTE

Both the Dataverse connector and the OData APIs are meant to serve analytical scenarios where data volumes are relatively small. The recommended approach for bulk data extraction is "Export to Data Lake". The TDS endpoint is a better option than the Dataverse connector and OData endpoint, but is currently in Preview.

SQL Server connection issue due to closed ports

When connecting with the Dataverse connector, you might encounter an **Unable to connect** error indicating that a network or instance-specific error occurred while establishing a connection to SQL Server. This error is likely caused by the TCP ports 1433 and 5558 being blocked during connection. To troubleshoot the blocked port error, go to [Blocked ports](#).

Using native database queries with Dataverse

You can connect to Dataverse using a custom SQL statement or a [native database query](#). While there's no user interface for this experience, you can enter the query using the Power Query Advanced Editor. In order to use a native database query, a **Database** must be specified as the Source.

```
Source = CommonDataService.Database([DATABASE URL])
```

Once a database source has been defined, you can specify a native query using the [Value.NativeQuery](#) function.

```
myQuery = Value.NativeQuery(Source, [QUERY], null, [EnableFolding=true])
```

Altogether, the query will look like this.

```
let
    Source = CommonDataService.Database("[DATABASE]"),
    myQuery = Value.NativeQuery(Source, "[QUERY]", null, [EnableFolding=true])
in
    myQuery
```

Note that misspelling a column name may result in an error message about query folding instead of missing column.

Analyze data in Azure Data Lake Storage Gen2 by using Power BI

1/15/2022 • 4 minutes to read • [Edit Online](#)

In this article you'll learn how to use Power BI Desktop to analyze and visualize data that is stored in a storage account that has a hierarchical namespace (Azure Data Lake Storage Gen2).

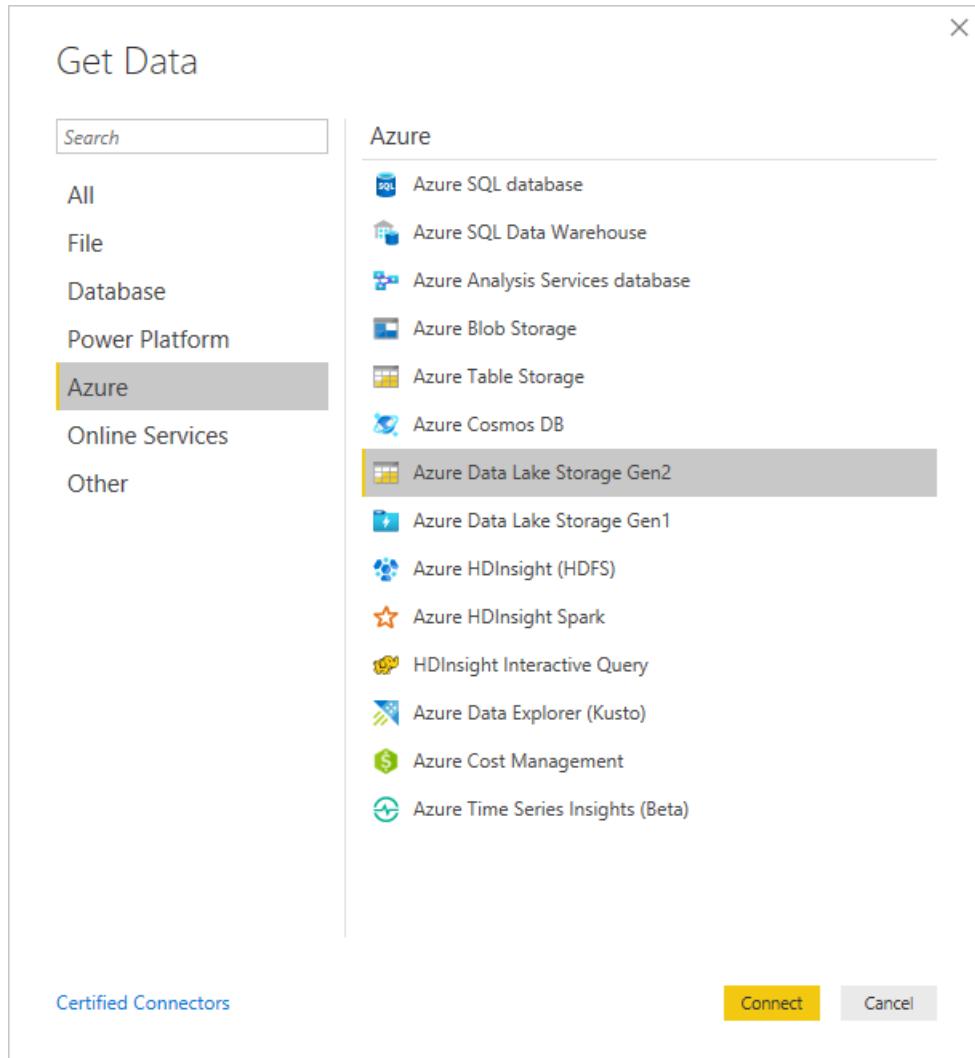
Prerequisites

Before you begin this tutorial, you must have the following prerequisites:

- An Azure subscription. See [Get Azure free trial](#).
- A storage account that has a hierarchical namespace. Follow [these](#) instructions to create one. This article assumes that you've created a storage account named `myadlsg2`.
- You are granted one of the following roles for the storage account: **Blob Data Reader**, **Blob Data Contributor**, or **Blob Data Owner**.
- A sample data file named `Drivers.txt` located in your storage account. You can download this sample from [Azure Data Lake Git Repository](#), and then upload that file to your storage account.
- **Power BI Desktop**. You can download this from the [Microsoft Download Center](#).

Create a report in Power BI Desktop

1. Launch Power BI Desktop on your computer.
2. From the **Home** tab of the Ribbon, select **Get Data**, and then select **More**.
3. In the **Get Data** dialog box, select **Azure > Azure Data Lake Store Gen2**, and then select **Connect**.

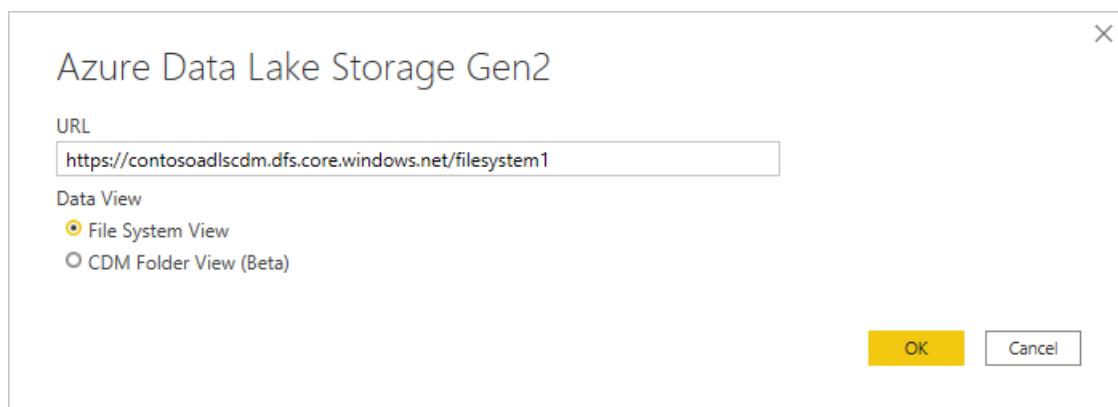


4. In the **Azure Data Lake Storage Gen2** dialog box, you can provide the URL to your Azure Data Lake Storage Gen2 account, filesystem, or subfolder using the container endpoint format. URLs for Data Lake Storage Gen2 have the following pattern:

```
https://<accountname>.dfs.core.windows.net/<filesystemname>/<subfolder>
```

You can also select whether you want to use the file system view or the Common Data Model folder view.

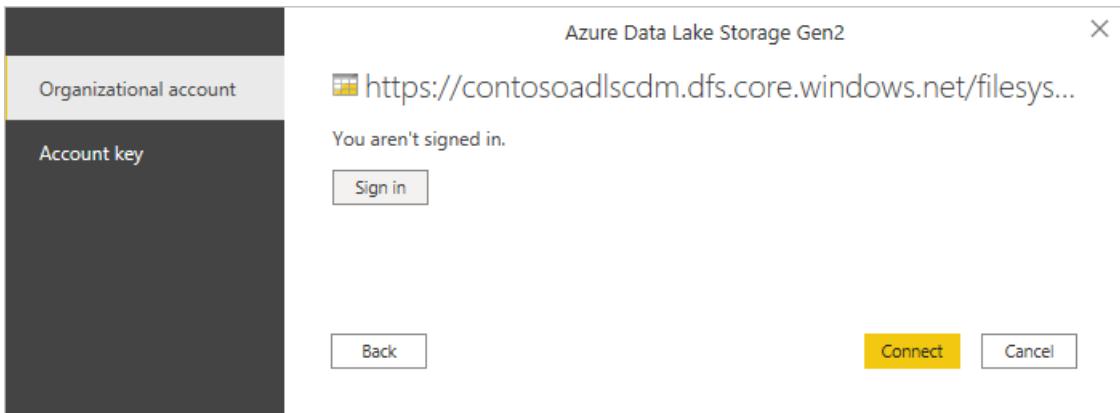
Select **OK** to continue.



5. If this is the first time you're using this URL address, you'll be asked to select the authentication method.

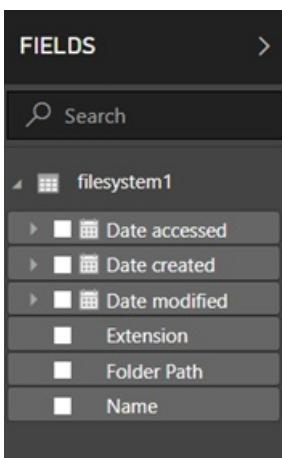
If you select the Organizational account method, select **Sign in** to sign into your storage account. You'll be redirected to your organization's sign in page. Follow the prompts to sign into the account. After you've successfully signed in, select **Connect**.

If you select the Account key method, enter your account key and then select **Connect**.



6. The next dialog box shows all files under the URL you provided in step 4 above, including the file that you uploaded to your storage account. Verify the information, and then select **Load**.

7. After the data has been successfully loaded into Power BI, you'll see the following fields in the **Fields** tab.

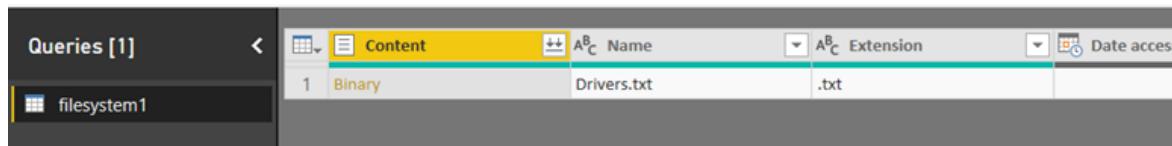


However, to visualize and analyze the data, you might prefer the data to be available using the following fields.

0	1	2	3	4	5	6
1	Maria Anders	Obere Str. 57	Berlin	12209	Germany	
2	Ana Trujillo	Avda. de la Constituci�n 2222	M�xico D.F.	5021	Mexico	

In the next steps, you'll update the query to convert the imported data to the desired format.

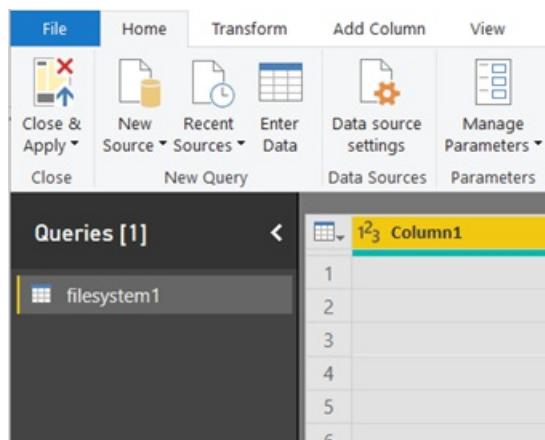
8. From the Home tab on the ribbon, select **Edit Queries**.



9. In the **Query Editor**, under the **Content** column, select **Binary**. The file will automatically be detected as CSV and you should see an output as shown below. Your data is now available in a format that you can use to create visualizations.

A screenshot of the Power BI Query Editor showing the data loaded into the 'filesystem1' query. The data is displayed in a table with columns: Column1, Column2, Column3, Column4, Column5, Column6, and Column7. The table contains 15 rows of data. The 'QUERY SETTINGS' pane on the right shows the source is 'Changed Type'. The 'PROPERTIES' section shows the name is 'filesystem1'. The 'APPLIED STEPS' section shows the source, navigation, and imported CSV steps.

10. From the Home tab on the ribbon, select **Close & Apply**.



11. Once the query is updated, the **Fields** tab will show the new fields available for visualization.

The screenshot shows the 'FIELDS' tab in a data modeling interface. A tree view displays a single item named 'filesystem1'. Underneath it, there are eight columns labeled 'Column1' through 'Column8'. Each column entry includes a small icon and a dropdown arrow.

12. Now you can create a pie chart to represent the drivers in each city for a given country. To do so, make the following selections.

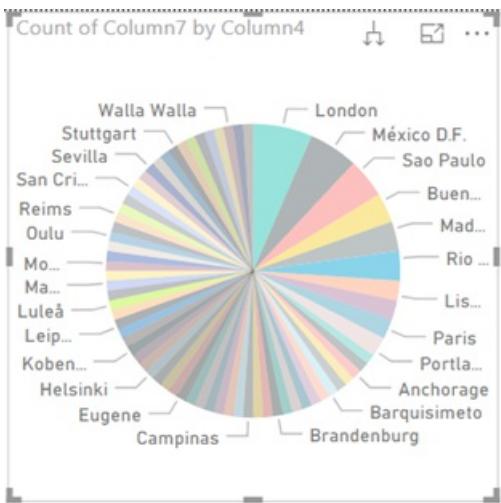
From the **Visualizations** tab, select the symbol for a pie chart.

The screenshot shows the 'Visualizations' tab. It contains a grid of icons representing different types of charts and data visualizations. One specific icon, which looks like a pie chart, is highlighted with a yellow box.

In this example, the columns you're going to use are Column 4 (name of the city) and Column 7 (name of the country). Drag these columns from the **Fields** tab to the **Visualizations** tab as shown below.

This screenshot shows the 'Visualizations' tab on the left and the 'Fields' tab on the right. In the 'Visualizations' tab, there's a 'Details' section where 'Column4' is listed with a red circle containing '1'. Below it is a 'Values' section with 'Count of Column7' and a red circle containing '2'. In the 'Filters' section, 'Column7' is listed with a red circle containing '3'. In the 'Fields' tab, 'Column4' and 'Column7' are selected, indicated by yellow checkmarks. Red arrows numbered 1, 2, and 3 point from the respective sections in the 'Visualizations' tab to the corresponding selected columns in the 'Fields' tab.

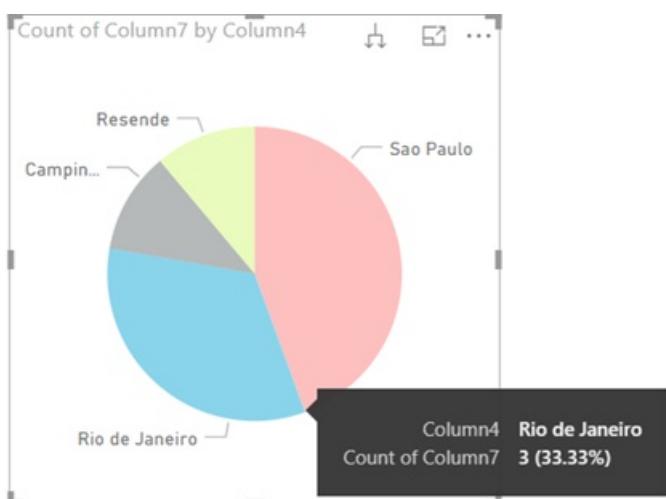
The pie chart should now resemble the one shown below.



13. By selecting a specific country from the page level filters, you can now see the number of drivers in each city of the selected country. For example, under the **Visualizations** tab, under **Page level filters**, select **Brazil**.

Column7	Count
is Brazil	9
Filter type	
Basic filtering	
Select all	
Argentina	3
Austria	2
Belgium	2
<input checked="" type="checkbox"/> Brazil	9
Canada	3
Denmark	2
Finland	2
France	11
<input type="checkbox"/> Require single selection	

14. The pie chart is automatically updated to display the drivers in the cities of Brazil.



15. From the **File** menu, select **Save** to save the visualization as a Power BI Desktop file.

Publish report to Power BI service

After you've created the visualizations in Power BI Desktop, you can share it with others by publishing it to the Power BI service. For instructions on how to do that, see [Publish from Power BI Desktop](#).

Troubleshooting

Currently, in Power Query Online, the Azure Data Lake Storage Gen2 connector only supports paths with container, and not subfolder or file. For example, `https://<accountname>.dfs.core.windows.net/<container>` will work, while `https://<accountname>.dfs.core.windows.net/<container>/<filename>` or `https://<accountname>.dfs.core.windows.net/<container>/<subfolder>` will fail.

Microsoft doesn't support dataflow or dataset refresh using OAuth2 authentication when the Azure Data Lake Storage Gen 2 (ADLS) account is in a different tenant. This limitation only applies to ADLS when the authentication method is OAuth2, that is, when you attempt to connect to a cross-tenant ADLS using an Azure AD account. In this case, we recommend that you use a different authentication method that is not OAuth2/AAD, such as the Key authentication method.

Dataverse

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Dynamics 365 Customer Insights Power Apps (Dataflows)
Authentication types	Organizational account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

You must have a Dataverse environment with maker permissions to access the portal, and read permissions to access data within tables.

To use the Dataverse connector, the **TDS endpoint** setting must be enabled in your environment. More information: [Manage feature settings](#)

To use the Dataverse connector, TCP ports 1433 and/or 5558 need to be open to connect. If only port 5558 is enabled, you must append that port number to the Dataverse environment URL, such as `yourenvironmentid.crm.dynamics.com:5558`. More information: [SQL Server connection issue due to closed ports](#)

Capabilities supported

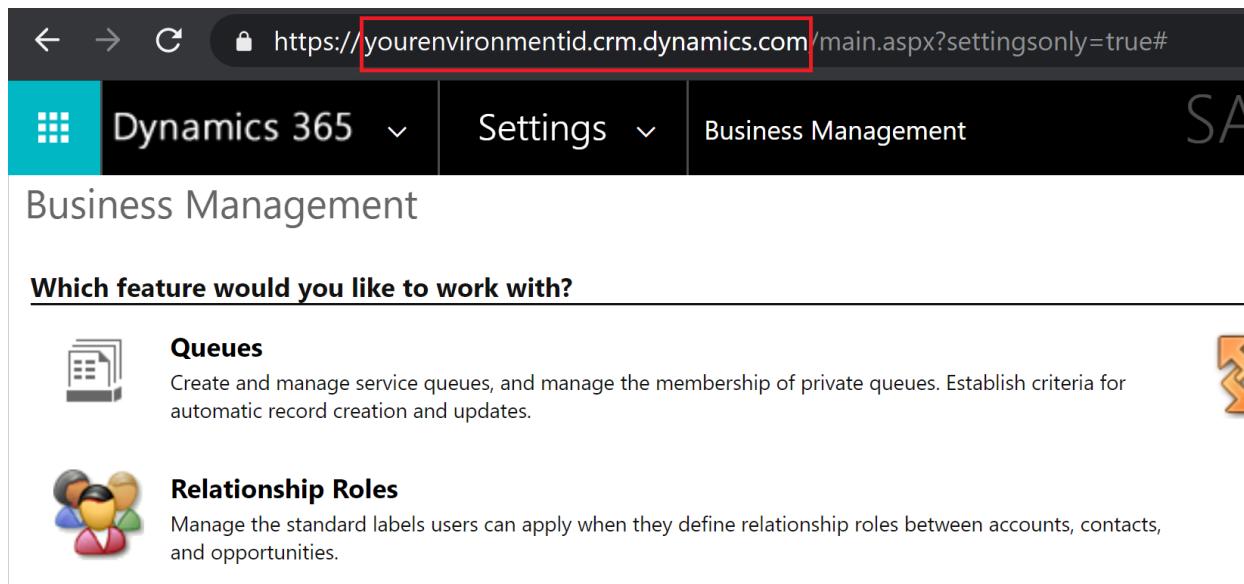
- Server URL
- Advanced
 - Reorder columns
 - Add display column

Finding your Dataverse environment URL

Open [Power Apps](#). In the upper right of the Power Apps page, select the environment you're going to connect to. Select the  settings icon, and then select **Advanced settings**.

In the new browser tab that opens, copy the root of the URL. This root URL is the unique URL for your environment. The URL will be in the format of `https://<yourenvironmentid>.crm.dynamics.com/`. **Make sure you remove https:// and the trailing / from the URL before pasting it to connect to your**

environment. Keep this URL somewhere handy so you can use it later, for example, when you create Power BI reports.



The screenshot shows a browser window with the URL <https://yourenvironmentid.crm.dynamics.com/main.aspx?settingsonly=true#> highlighted by a red box. The page title is "Business Management". The top navigation bar includes "Dynamics 365", "Settings", and "Business Management". Below the title, the heading "Which feature would you like to work with?" is followed by two items: "Queues" and "Relationship Roles".

Queues
Create and manage service queues, and manage the membership of private queues. Establish criteria for automatic record creation and updates.

Relationship Roles
Manage the standard labels users can apply when they define relationship roles between accounts, contacts, and opportunities.

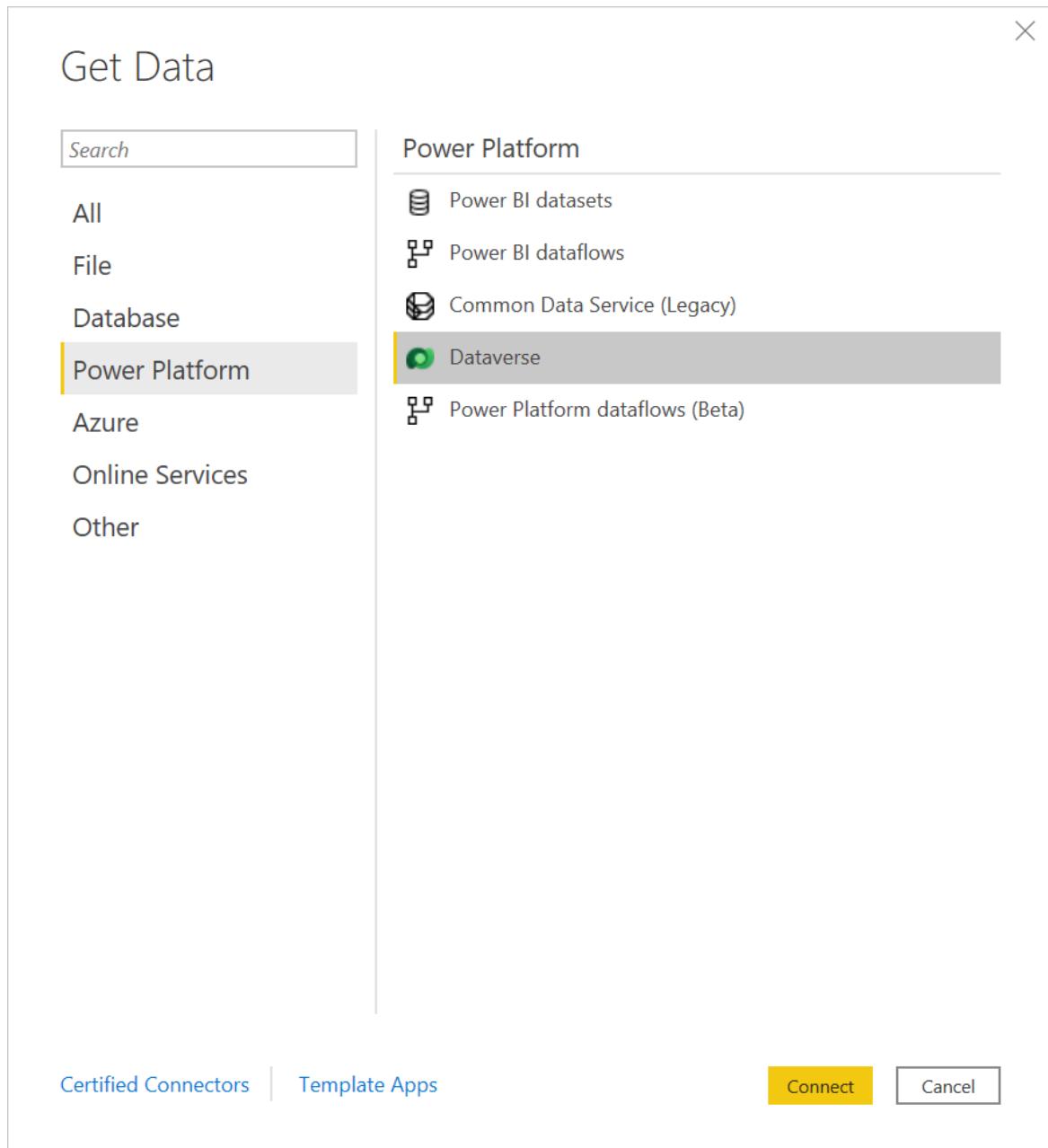
Connect to Dataverse from Power BI Desktop

NOTE

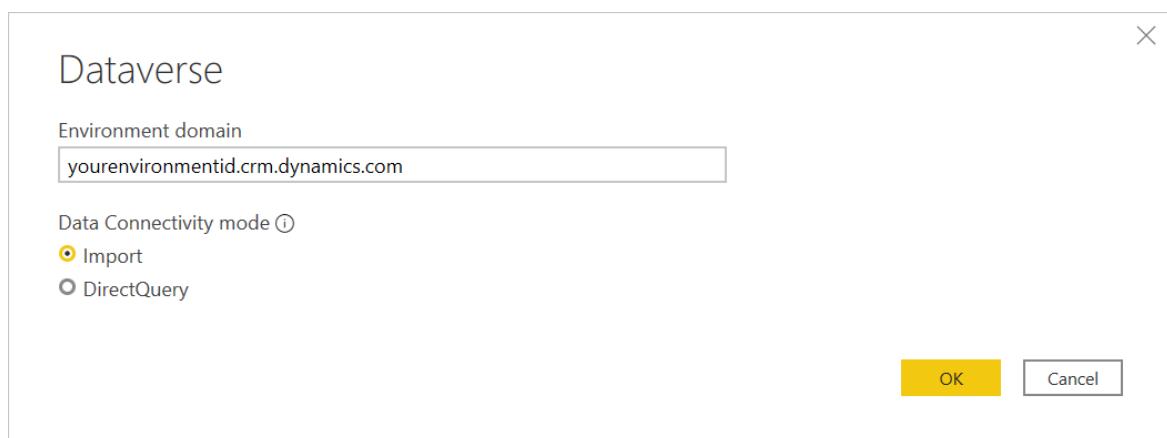
The Power Query Dataverse connector is mostly suited towards analytics workloads, not bulk data extraction. More information: [Alternative Dataverse connections](#)

To connect to Dataverse from Power BI Desktop:

1. Select **Get data** from the **Home** tab.
2. In the **Get Data** dialog box, select **Power Platform > Dataverse**, and then select **Connect**.



3. Enter the Dataverse environment URL of the data you want to load. Use the format <yourenvironmentid>.crm.dynamics.com. Be sure to remove the https:// prefix and / suffix from the URL before entering the name in **Environment domain**. More information: [Finding your Dataverse environment URL](#)



4. Select one of the following Data Connectivity mode options:
- **Import:** We recommend that you import data to Power BI wherever possible. With this mode, data is

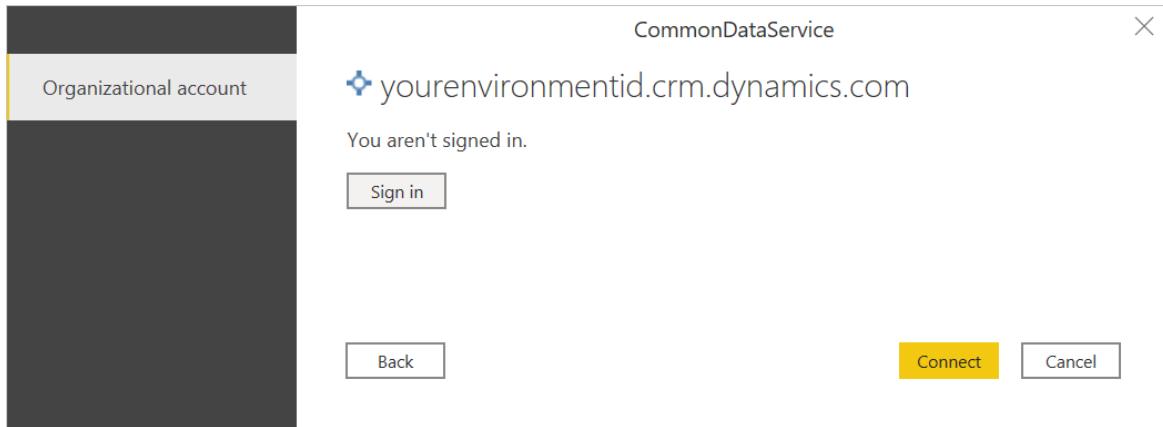
cached in the Power BI service and imported on a scheduled interval.

- **DirectQuery**: Connects directly to the data in Dataverse. Use this mode for real-time data retrieval. This mode can also more strictly enforce the Dataverse security model. More information: [DirectQuery model guidance in Power BI Desktop](#)

When you've finished filling in the information, select **OK**.

5. If this attempt is the first time you're connecting to this site, select **Sign in** and input your credentials.

Then select **Connect**.



6. In **Navigator**, select the data you require, then either load or transform the data.

	statecode	statecode_display	statuscode	statuscode_display
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active
:1db51667	0	Active	1	Active

Connect to Dataverse from Power Query Online

To connect to Dataverse from Power Query Online:

1. From the **Data sources** page, select **Common Data Service (Legacy)**.

Data sourcesCommon Data Service (Legacy)
Power PlatformPower BI dataflows
Power PlatformPower Platform data
Power Platform

2. Enter the server URL address of the data you want to load.

Common Data Service (Legacy)
Power Platform

Connection settings

Server Url *

https://yourenvironmentid.crm.dynamics.com

> Advanced options

Connection credentials

On-premises data gateway

(none)

Authentication kind

Organizational account

You are not signed in. Please sign in.

Sign in

3. If necessary, enter an on-premises data gateway if you're going to be using on-premises data. For example, if you're going to combine data from Dataverse and an on-premises SQL Server database.

4. Sign in to your organizational account.

5. When you've successfully signed in, select **Next**.

6. In the navigation page, select the data you require, and then select **Transform Data**.

Limitations and issues

When to use the Common Data Service (Legacy) connector

Dataverse is the direct replacement for the Common Data Service connector. However, there may be times when it's necessary to choose the Common Data Service (Legacy) connector instead of the Dataverse connector:

- If you're connecting to data using Power Apps, you'll still have to use the Common Data Service (Legacy) connector.
- If you're accessing large datasets that are greater than 80 MB, you'll still have to use the Common Data Service (Legacy) connector.
- If you want paging of the query results and want to build reports that use the image data type, you'll still have to use the Common Data Service (Legacy) connector.

Also, there are certain Tabular Data Stream (TDS) data types that are supported in OData when using Common Data Service (Legacy) that aren't supported in Dataverse. The supported and unsupported data types are listed in [How Dataverse SQL differs from Transact-SQL \(Preview\)](#).

All of these features will be added to the Dataverse connector in the future, at which time the Common Data Service (Legacy) connector will be deprecated.

Dataverse performance and throttling limits

For information about performance and throttling limits for Dataverse connections, go to [Requests limits and allocations](#). These limitations apply to both the Dataverse connector and the [OData Feed](#) connector when accessing the same endpoint.

Table retrieval rate

As a guideline, most default tables will be retrieved at a rate of approximately 500 rows per second using the Dataverse connector. Take this rate into account when deciding whether you want to connect to Dataverse or export to data lake. If you require faster retrieval rates, consider using the Export to data lake feature or Tabular Data Stream (TDS) endpoint. For more information, go to [Alternative Dataverse connections](#).

Alternative Dataverse connections

There are several alternative ways of extracting and migrating data from Dataverse:

- Use the OData connector to move data in and out of Dataverse. For more information on how to migrate data between Dataverse environments using the dataflows OData connector, go to [Migrate data between Dataverse environments using the dataflows OData connector](#).
- Use the **Export to data lake** feature in Power Apps to extract data from Dataverse into Azure Data Lake Storage, which can then be used to run analytics. For more information about the export to data lake feature, go to [Exporting Dataverse data to Azure Data Lake is Generally Available](#).
- Use the Tabular Data Stream (TDS) Protocol endpoint to access read-only data in Dataverse. For more information about this preview feature and a video on how it works, go to [Tabular Data Stream \(TDS\) Protocol endpoint for Dataverse](#).

NOTE

Both the Dataverse connector and the OData APIs are meant to serve analytical scenarios where data volumes are relatively small. The recommended approach for bulk data extraction is "Export to Data Lake". The TDS endpoint is a better option than the Dataverse connector and OData endpoint, but is currently in Preview.

SQL Server connection issue due to closed ports

When connecting with the Dataverse connector, you might encounter an **Unable to connect** error indicating that a network or instance-specific error occurred while establishing a connection to SQL Server. This error is likely caused by the TCP ports 1433 and 5558 being blocked during connection. To troubleshoot the blocked port error, go to [Blocked ports](#).

Using native database queries with Dataverse

You can connect to Dataverse using a custom SQL statement or a [native database query](#). While there's no user interface for this experience, you can enter the query using the Power Query Advanced Editor. In order to use a native database query, a **Database** must be specified as the Source.

```
Source = CommonDataService.Database([DATABASE URL])
```

Once a database source has been defined, you can specify a native query using the [Value.NativeQuery](#) function.

```
myQuery = Value.NativeQuery(Source, [QUERY], null, [EnableFolding=true])
```

Altogether, the query will look like this.

```
let
    Source = CommonDataService.Database("[DATABASE]"),
    myQuery = Value.NativeQuery(Source, "[QUERY]", null, [EnableFolding=true])
in
    myQuery
```

Note that misspelling a column name may result in an error message about query folding instead of missing column.

Delta Sharing (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Databricks, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Databricks website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI (Datasets)
Authentication Types Supported	Key (Bearer Token)

Prerequisites

If you use Power BI Desktop you need to install the November release of Power BI Desktop or later. [Download the latest version](#).

The data provider sends an activation URL from which you can download a credentials file that grants you access to the shared data.

After downloading the credentials file, open it with a text editor to retrieve the endpoint URL and the token.

For detailed information about Delta Sharing, visit [Access data shared with you using Delta Sharing](#).

Capabilities supported

- Import

Connect to Databricks Delta Sharing in Power BI Desktop

To connect to Databricks using the Delta Sharing connector, use the following steps:

1. Open Power BI Desktop.
2. Navigate to the **Get Data** menu and search for **Delta Sharing**.
3. Select the connector and then select **Connect**.
4. Enter the endpoint URL retrieved from the credentials file in the Delta Sharing Server URL field.
5. Optionally, in the **Advanced Options** tab you can set a **Row Limit** for the maximum number of rows you can download. This is set to 1 million rows by default.
6. Select **OK**.

7. In the **Authentication** dialog box, enter the token retrieved from the credentials file in the **Bearer Token** field.
8. Select **Connect**.

Limitations and considerations

This section describes any limitations or considerations of the Delta Sharing connector.

You need to make sure that the data loaded with the Delta Sharing connector fits in the memory of your machine. To ensure this, the connector limits the number of imported rows to the Row Limit set by the user.

EQuIS (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by EarthSoft, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the EarthSoft website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Preview
Products	Power BI (Datasets)
Authentication Types Supported	Basic Web API (API Token) Organizational Account
Function Reference Documentation	-

Prerequisites

To use the EQuIS connector, you must have a valid user account in an EQuIS Enterprise site (version 7.0.0.19300 or later) that includes a REST API license. Your user account must be a member of the REST API role. To verify user account configuration, go to the **Roles** tab in your [user profile](#) and verify that you are a member of the REST API role.

Capabilities supported

- Import

Connect to EQuIS from Power BI Desktop

To connect to an EQuIS Enterprise site from Power BI Desktop, take the following steps:

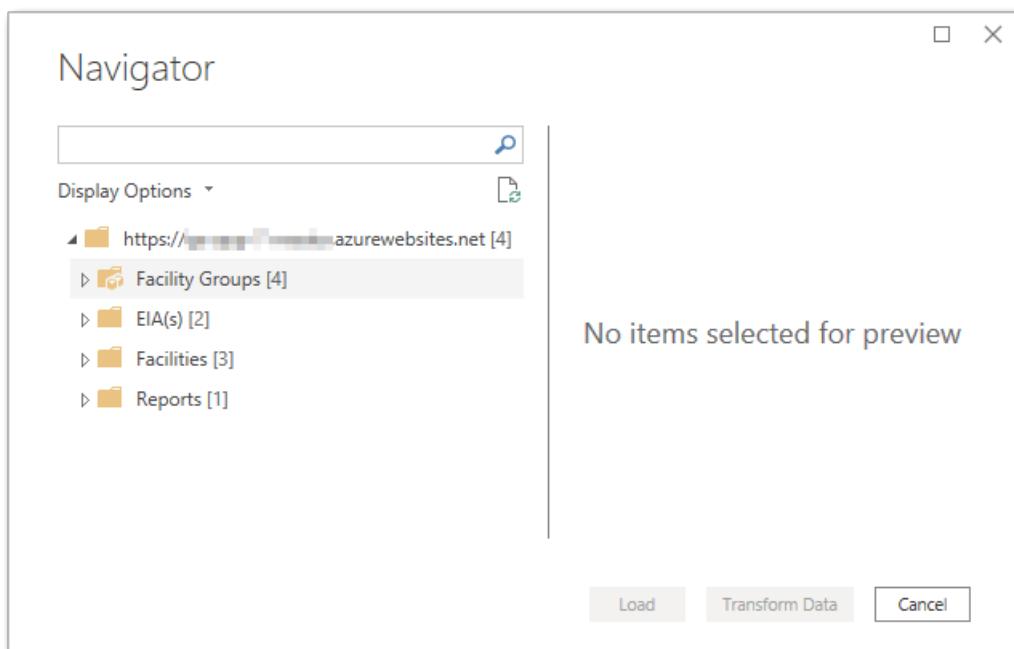
1. Select the EQuIS connector in the connector list, then select **Connect**.
2. Enter the URL of the EQuIS Enterprise site you are connecting to, then select **OK**.



3. Select the appropriate type of authentication:

- **Basic:** Enter your EQuIS username and password for the given EQuIS Enterprise site.
- **API Token:** Enter an API Token that you generated in EQuIS Enterprise (visit [User Profile](#)).
- **Organizational Account:** If your EQuIS Enterprise site is appropriately configured, you may authenticate with Azure Active Directory

4. In **Navigator**, browse to the dataset or report you want to load, then select **Load** or **Transform Data**. Visit [Using EQuIS Data](#) for more information about available datasets.



Additional Information

- For best functionality and performance, EarthSoft recommends that you use the EQuIS connector with the latest build of EQuIS Enterprise.
- When using reports in a facility group, non-administrator users must have permission to all facilities contained in the facility group.
- Only "grid" reports will be available in the **Navigator**.
- All datasets consumed by the EQuIS connector will use camelCase for column names.
- The current version of the EQuIS connector will retrieve a dataset in a single request and is limited to 1,048,576 rows in a single dataset (this limitation might be removed in a future version of the connector).

Essbase

1/15/2022 • 16 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Basic (Username/Password)
Function Reference Documentation	Essbase.Cubes

Prerequisites

None

Capabilities Supported

- Import
- Direct Query
- Advanced options
 - Command timeout in minutes
 - Server
 - Application
 - MDX statement

Connect to Essbase from Power Query Desktop

To connect to an Essbase server:

1. Select the **Essbase** option in the **Get Data** experience.
2. Enter the URL to the Oracle Essbase Hyperion server. Typically, the URL looks like `http://[hostname]:[port number]/aps/XMLA`. The components of the URL are:
 - The `hostname` (for example, `yourservername.domain.com`) is the hostname or IP address of the Oracle Hyperion Application Provider Service (APS) server for your in-house system.
 - The `port number` (for example, 19000) is the port number the APS server is listening to for XMLA requests.
 - The last portion of the URL, the path (that is, `/aps/XMLA`), is case-sensitive and must be specified exactly as shown.



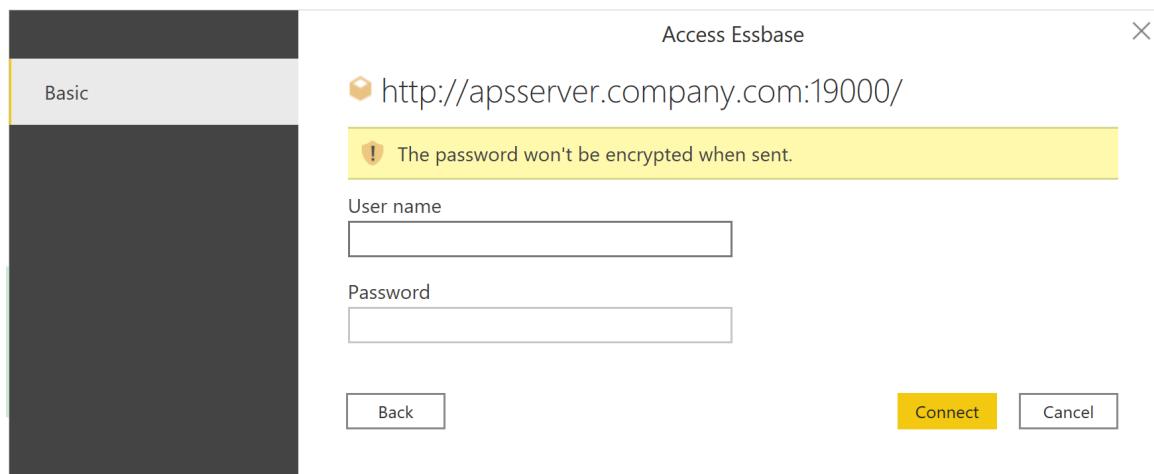
Some example URLs would be:

- `http://apsserver.company.com:19000/aps/XMLA` —Using fully qualified host name with default port 19000.
- `http://hypserver01:13080/aps/XMLA` —Using a not fully qualified host name and port 13080.
- `http://10.10.10.10/aps/XMLA` —Using an IP address and port 80—changed from default 19000.

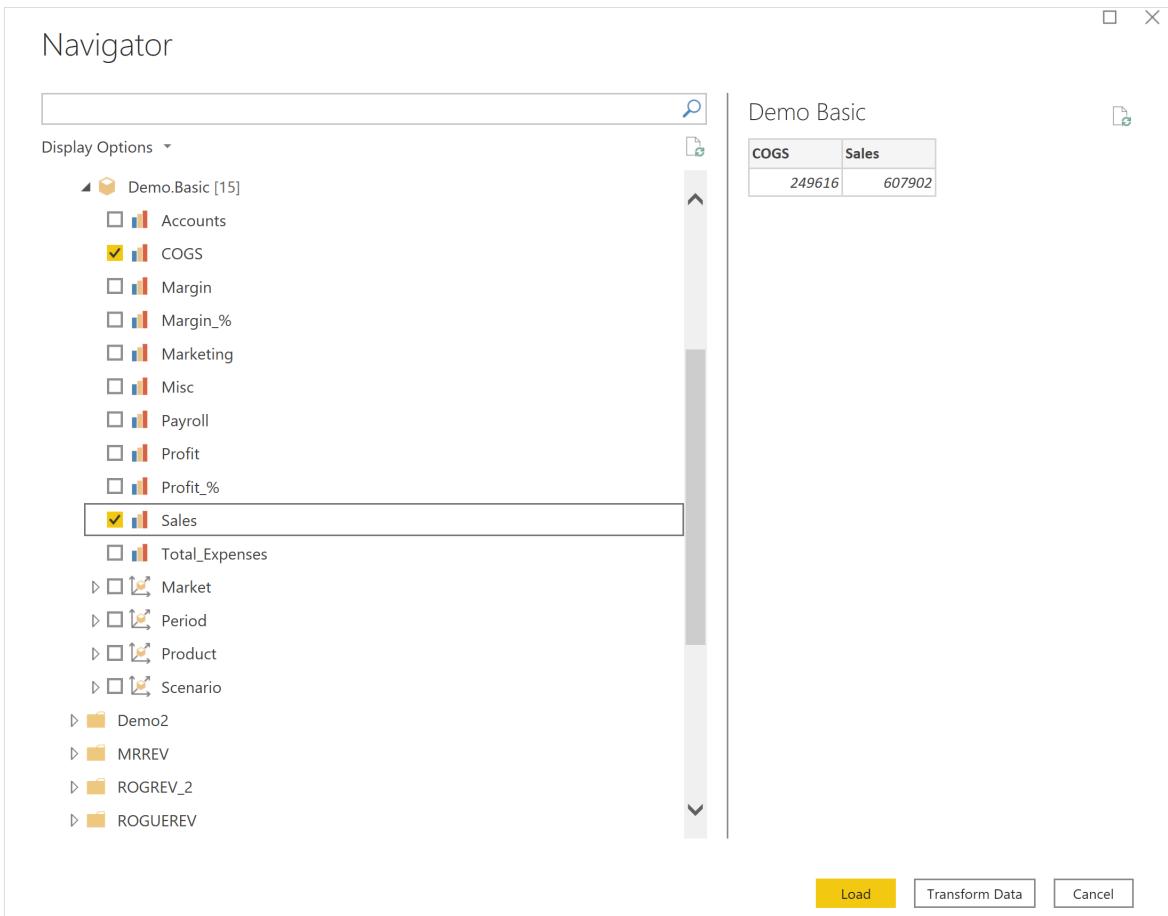
3. Select either the **Import** or **DirectQuery** data connectivity mode. More information: [Use DirectQuery in Power BI Desktop](#)

Optionally, enter values in any advanced options that you want to use to modify the connection query. More information: [Connect using advanced options](#)

4. The first time you connect to a data source (identified by each unique URL), you'll be prompted to enter account credentials. Enter the **User name** and **Password** for the connection. More information: [Authentication with a data source](#)



5. In **Navigator**, select the data you require. Then, either select **Transform data** to transform the data in Power Query Editor, or **Load** to load the data in Power BI.



Connect using advanced options

Power Query provides a set of advanced options that you can add to your query if needed. The following table lists all of the advanced options you can set in Power Query.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	Lets you set the maximum time a command is allowed to run before Power BI abandons the call. If the command timeout is reached, Power BI may retry two more times before completely abandoning the call. This setting is helpful for querying large amounts of data. The default value of the command timeout is 140 seconds.
Server	The name of the server where the optional MDX statement is to run. This value is case sensitive.
Application	The name of the application where the optional MDX statement is to run. This value is case sensitive.

ADVANCED OPTION	DESCRIPTION
MDX statement	<p>Optionally provides a specific MDX statement to the Oracle Essbase server to execute. Normally, Power BI interactively determines the measures and dimensions of the cube to return. However, by specifying the MDX statement, the results of that particular MDX statement will be loaded. When you specify the MDX statement, you must also provide the Server (for example, <code>essbaseserver-1</code>) and Application (for example, <code>Sample</code>) advanced options to indicate where the MDX statement is to run. Also, you can only use the MDX statement in conjunction with Data Connectivity mode set to Import.</p> <p>In general, the MDX generated by SmartView or one accepted by Essbase Admin is not 100% compatible with Power BI.</p> <p>PowerBI requires measures to be specified on a 0 axis in an MDX query. In addition, level numbering is reversed in XMLA. The least granular level is level 0, 1, and so on in XML, but the opposite in Essbase "directly". So if level numbers are used explicitly in the MDX query, they need to be adjusted.</p>

Data Connectivity mode ⓘ

Import

DirectQuery

▲ Advanced options

Command timeout in minutes (optional)

Server

Application

MDX statement (optional)

```
SELECT
  {
    [Margin],
    [Profit],
    [Sales]
  } ON 0,
```

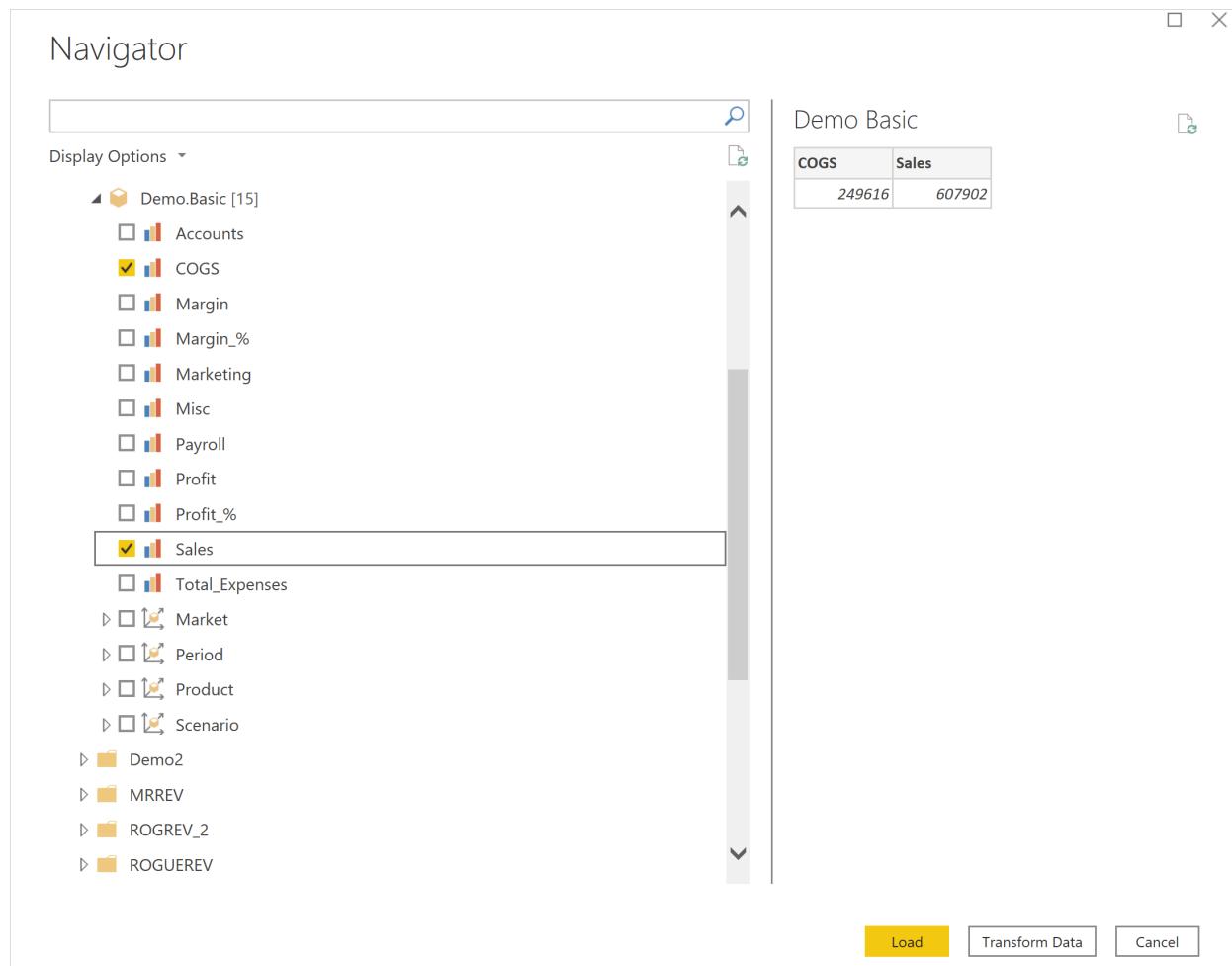
OK Cancel

Using data source navigator when importing data

When **Data Connectivity mode** is set to **Import**, the data source navigator loads the servers configured for the APS server you've provided in the URL. Expanding a server node in the tree reveals the available applications. Expanding an application node reveals the available databases (also known as cubes). Expanding a database node reveals the available measures and dimensions. The dimension can be further expanded to reveal the levels in the hierarchy for each dimension.

Choose a measure and all (or specific) dimension levels by selecting the checkbox next to the name. A preview of the data is provided in the pane on the right. You can select the **Load** button to retrieve the data associated

with the selection or select the **Transform Data** button to set further filters on the data before loading it in Power BI.



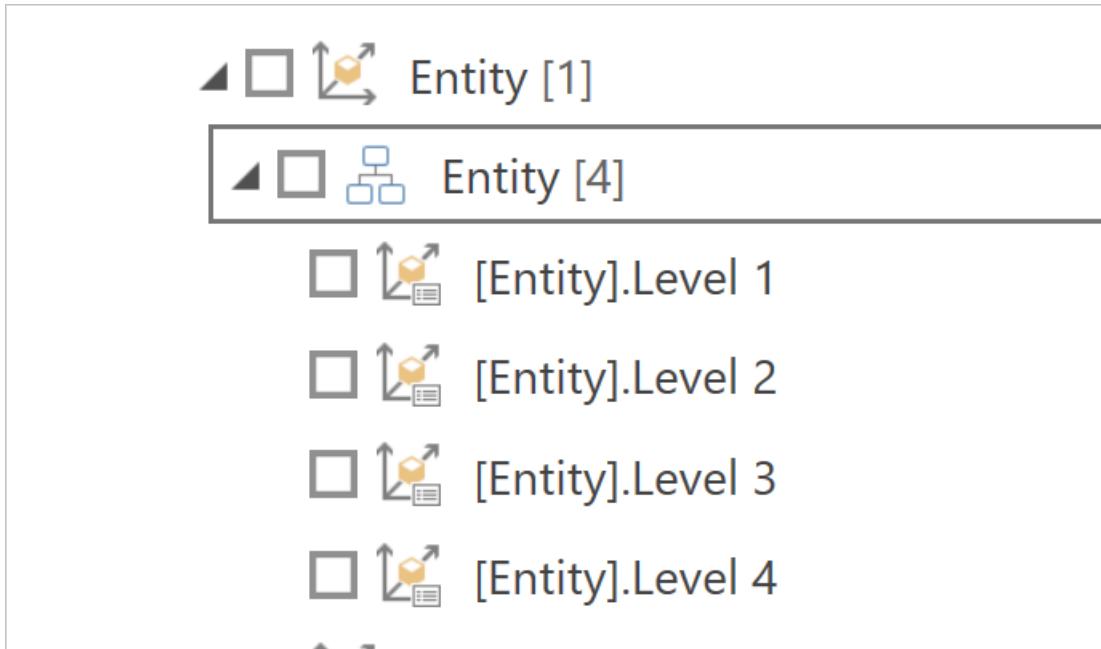
Differences in display compared with Essbase administration

When expanding a dimension hierarchy in the navigator, you might notice that it looks different when compared to using the Essbase Administration Services control panel.

As an example, the following image shows the expanded Entity dimension hierarchy from Essbase Administration Services.



While in the Power Query navigator, the same Entity being expanded appears like this:



Be aware that this look is a stylistic decision and that there are no differences in data. The levels in the Power Query navigator correspond to the hierarchical level.

In the example above, Level 1 would contain "R_ReportingsUnits", "Adjustment Entity Input" and "No_Entity". Level 2 would contain "R_Americas", "R_EMEA", "R_AsiaPacific", "1_ReportingsUnits_Adjustment", "CALA_HFM_Input", "CALA_Total", and so on.

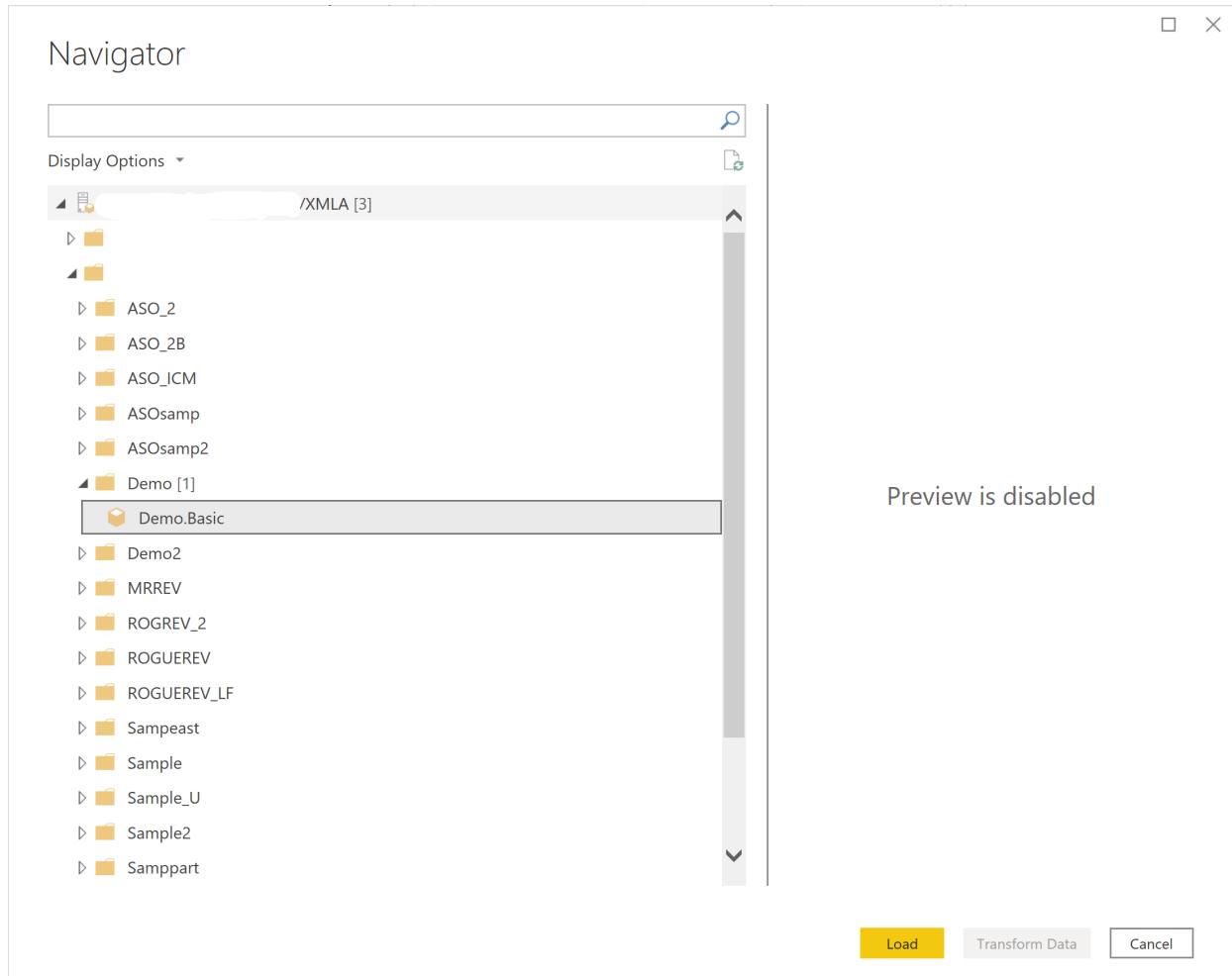
The reason is because the navigator in Power Query is limited to 10,000 members to display, and there can be millions or billions of members underneath a hierarchy. Even for the case of no member display limit (such as with Power Query Online), navigating and selecting every individual member in a tree format with so many

possible values quickly becomes tedious and difficult to use.

So, the grouping of the hierarchical levels makes it easier to select what to import, and the subsequent report generation can use filters to target only the members the end user wants.

Using data source navigator for DirectQuery

When a **Data Connectivity mode** of DirectQuery is chosen, the data source navigator loads the servers configured for the APS server you've provided in the URL. Expanding a server node in the tree reveals the available applications. Expanding an application node reveals the available databases (also known as cubes).



Known limitations

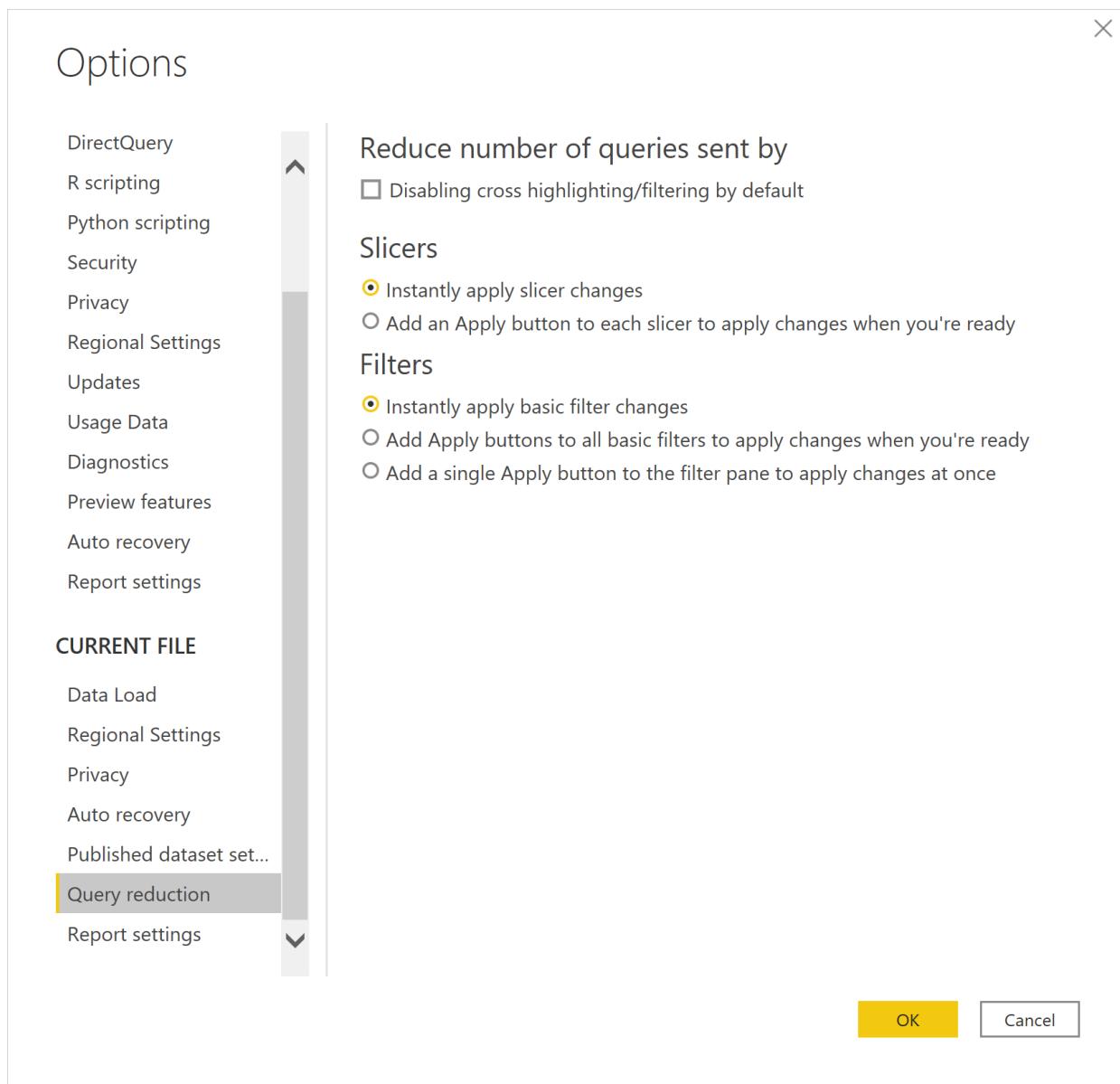
The Essbase connector doesn't support measure hierarchies. All measures are displayed at the same level. You can still select all the measures that you need. The search field can be used to narrow down the displayed measures if there are large numbers of measures.

Performance considerations

Interacting with Power BI in DirectQuery mode is very dynamic. When selecting a checkbox to include a measure or dimension level in the visualization, Power BI Desktop generates a query and sends it to the Oracle Essbase server to get the results. Power BI is optimized to cache any repeated queries to improve performance. But if any new query is generated, it's sent to the Oracle Essbase server to produce a new result. Depending on the number of selected measures, dimension levels, and the filters applied, the query might get sent more quickly than the Oracle Essbase server can respond. To improve performance and increase responsiveness, consider the following three methods to optimize your interaction with the Oracle Essbase server.

Query reductions options

There are three options to reduce the number of queries sent. In Power BI Desktop, select the **File** tab, then select **Options and settings > Options**, and then select **Query reductions** under the **Current File** section.



Selecting the **Disabling cross highlighting/filtering by default** option under **Reduce number of queries sent by** disables cross highlighting/filtering by default. When disabled, member lists in the filter don't get updated when filtering members in other levels of the same dimension. Selecting the **Slicer selections** option under **Show an Apply button and only send queries once** for section displays the **Apply** button when a slicer selection is changed. Selecting the **Filter selections** option under **Show an Apply button and only send queries once** for section displays the **Apply** button when a filter selection is changed.

NOTE

These options apply only to the current file you are working on. **Current File** option settings are saved with the file and restored when opening the same file.

Iterative filter application when adding dimension levels in import mode

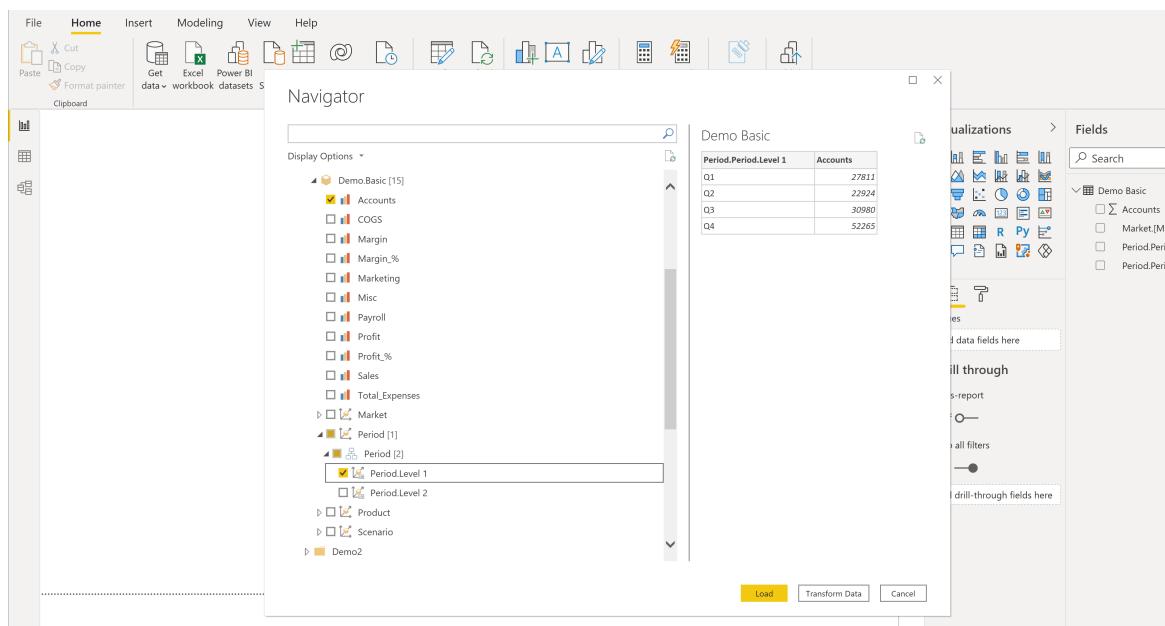
When interacting in import mode with a multidimensional cube data source like Oracle's Essbase, Power Query

initially displays the measures, dimensions, and dimension levels in the database **Navigator** dialog box. However, while Power BI makes it easy to select and visualize data it can, at times, lead to retrieving too much data from the server.

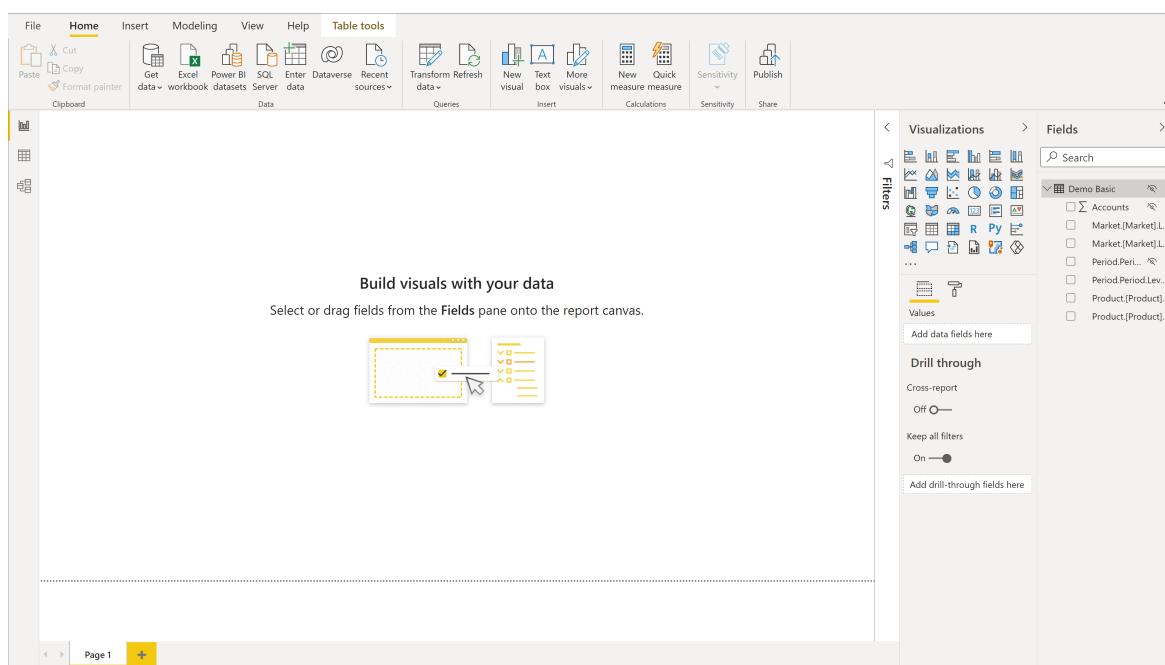
The following procedure demonstrates how to reduce the chances of retrieving more data than is necessary when importing data into Power BI by iteratively applying filters on dimension members at each level.

Connecting to the Oracle Essbase data source

1. Follow the instructions in [Connect to Essbase from Power Query Desktop](#) to connect to an Essbase server using import mode.
2. Expand the tree to drill down to your desired server, application, and database until it exposes the measures and dimensions for your database. For now, select your measures and only one dimension level. Pick the most important dimension level. In later steps, you'll build the result by incrementally adding more dimensions levels.



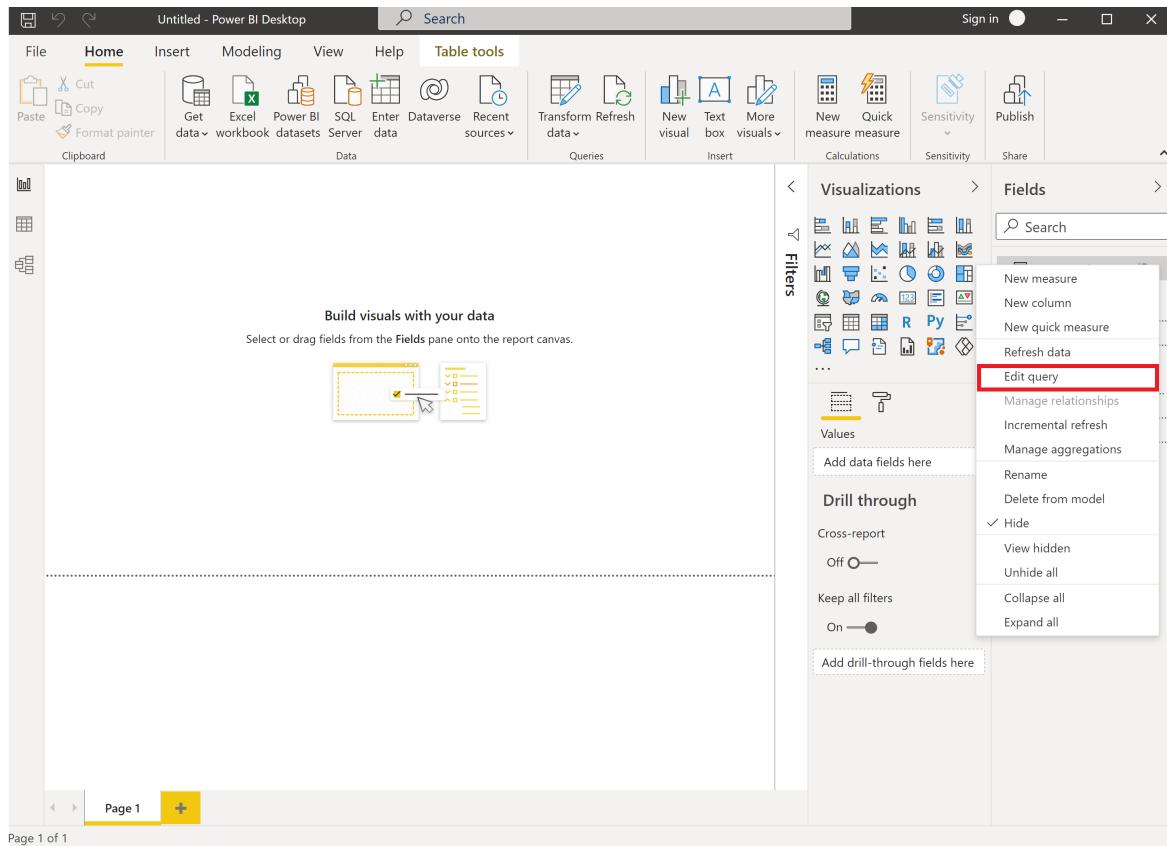
3. Select **Load** to import the selected measures and dimension level.



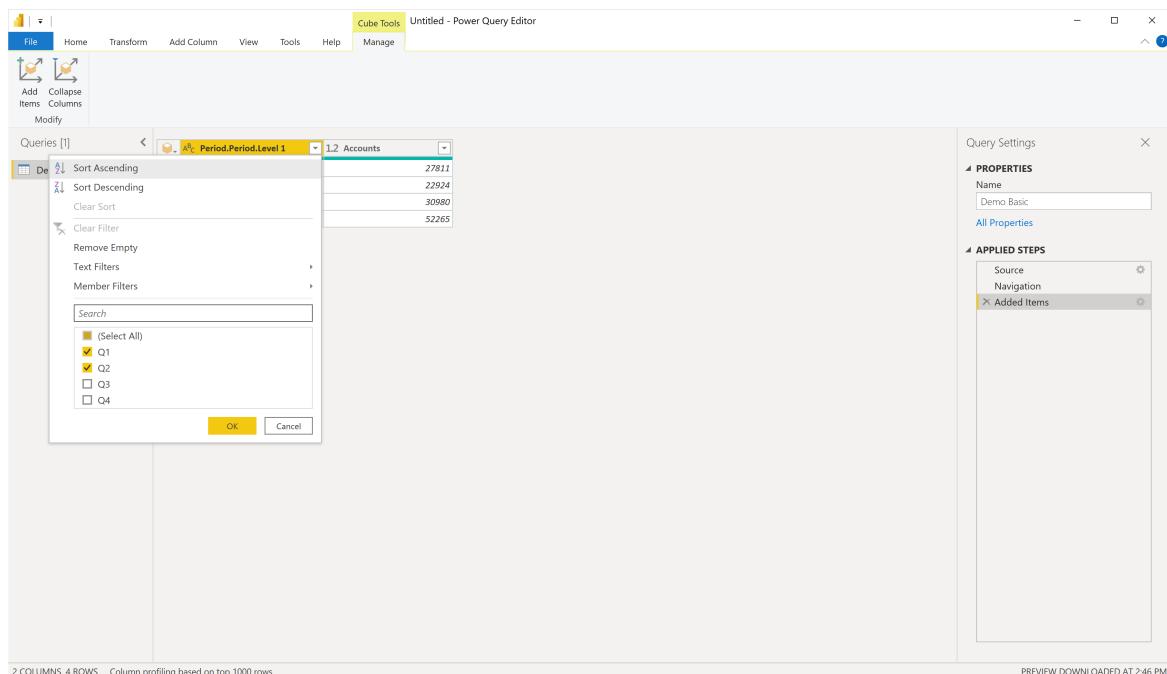
Editing queries to add more dimension levels

Now that you have a starting point, you can start adding more dimension levels and applying filters as you go.

1. Select **Edit Queries** on the Power BI Desktop ribbon to start the process.



2. If you have members you want to filter on in the initial dimension, select the column properties button to display the list of available dimension members at this level. Select only the dimension members you need at this level and then select **OK** to apply the filter.



3. The resulting data is now updated with the applied filter. **Applied Steps** now contains a new step (**Filtered Rows**) for the filter you set. You can select the settings button for the step to modify the filter at a later time.

Untitled - Power Query Editor

Cube Tools

File Home Transform Add Column View Tools Help Manage

Queries [1]

Demo Basic

Period.PeriodLevel 1 Accounts

1	Q2	22924
2	Q1	27811

Query Settings

PROPERTIES

Name: Demo Basic

APPLIED STEPS

- Source
- Navigation
- Added Items
- Filter Rows

PREVIEW DOWNLOADED AT 2:46 PM

- Now you'll add a new dimension level. In this case, you're going to add the next level down for the same dimension you initially chose. Select **Add Items** on the ribbon to bring up the **Navigator** dialog box.

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help Cube Tools

Queries [1]

Demo Basic

Period.PeriodLevel 1 Accounts

1	Q2	2313
2	Q2	3641
3	Q2	1211
4	Q1	4301
5	Q1	4766
6	Q1	1594

Query Settings

PROPERTIES

Name: Demo Basic

APPLIED STEPS

- Source
- Navigation
- Added Items
- Filtered Rows
- Added Items1
- Filtered Rows1
- Added Items2
- Removed Columns
- Added Items3
- Removed Columns1

PREVIEW DOWNLOADED AT 2:46 PM

- Navigate to the same dimension, but this time select the next level below the first level. Then select **OK** to add the dimension level to the result.

Add Items

Select dimensions and measures to add to the query.

Demo Basic [15]

- Accounts
- COGS
- Margin
- Margin_%
- Marketing
- Misc
- Payroll
- Profit
- Profit_%
- Sales
- Total_Expenses
- Market
- Period [1]
- PeriodLevel [2]
 - PeriodLevel 1
 - PeriodLevel 2
- Product
- Scenario

There are no variables available for this data source.

OK Cancel

Query Settings

Properties

Name: Demo Basic

All Properties

Applied Steps

Source
Navigation
Added Items
Filtered Rows
Added Items1
Added Items2
Filtered Rows1
Removed Columns

2 COLUMNS, 2 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 2:46 PM

6. The result grid now has the data from the new dimension level. Notice that because you've applied a filter at the top level, only the related members in the second level are returned.

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help Manage

Cube Tools

Queries [1]

Demo Basic

Period.Period.Level 1	Accounts	Period.Period.Level 2
1 Q2		627 Apr
2 Q2		893 Apr
3 Q2		793 Apr
4 Q2		556 May
5 Q2		841 May
6 Q2		972 May
7 Q2		792 Jun
8 Q2		959 Jun
9 Q2		843 Jun
10 Q1		1567 Jan
11 Q1		1431 Jan
12 Q1		1303 Jan
13 Q1		1259 Feb
14 Q1		1165 Feb
15 Q1		958 Feb
16 Q1		768 Mar
17 Q1		1012 Mar
18 Q1		957 Mar

Query Settings

Properties

Name: Demo Basic

All Properties

Applied Steps

Source
Navigation
Added Items
Added Items1
Added Items2
Removed Columns
Added Items3
Added Items5
Filtered Rows
Removed Columns1
Filtered Rows1
Added Items6
Filtered Rows2
Removed Columns2

3 COLUMNS, 18 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 2:46 PM

7. You can now apply a filter to the second-level dimension as you did for the first level.

The screenshot shows the Power Query Editor interface. In the center, there is a table with three columns: 'Period.Period.Level 1', 'Accounts', and 'Period.Period.Level 2'. The 'Period.Period.Level 1' column contains values Q2, Q2, Q2, Q1, Q1, and Q1. The 'Accounts' column contains numerical values. The 'Period.Period.Level 2' column contains month abbreviations: Apr, Apr, Apr, Jan, Jan, and Jan. A filter dialog is open over the table, showing a dropdown menu with options like 'Sort Ascending', 'Sort Descending', 'Clear Sort', 'Clear Filter', 'Remove Empty', 'Text Filters', and 'Member Filters'. Below this is a search bar and a list of months: (Select All), Apr, Feb, Jan, Jun, Mar, and May. The 'Jan' option is checked. At the bottom of the dialog are 'OK' and 'Cancel' buttons. To the right of the table is the 'Query Settings' pane, which includes sections for 'PROPERTIES' (Name: Demo Basic) and 'APPLIED STEPS'. The 'APPLIED STEPS' section lists several steps: Source, Navigation, Added Items, Filtered Rows, Added Items1, Filtered Rows1, Added Items2, Removed Columns, Added Items3, Removed Columns1, and Added Items4. 'Added Items1' is currently highlighted.

8. In this way, each subsequent step ensures only the members and data you need are retrieved from the server.

This screenshot is similar to the previous one but shows a different state of the query. The table now has six rows, each containing a value from the 'Period.Period.Level 1' column and a corresponding value from the 'Period.Period.Level 2' column. The 'Accounts' column remains hidden. The 'APPLIED STEPS' pane shows a longer list of steps, indicating more filtering has occurred. The steps listed are: Source, Navigation, Added Items, Filtered Rows, Added Items1, Filtered Rows1, Added Items2, Removed Columns, Added Items3, Removed Columns1, and Added Items4. 'Added Items4' is currently highlighted.

9. Now let's add a new dimension level by repeating the previous steps. Select **Add Items** on the ribbon bar again.

The screenshot shows the Power Query Editor interface. The main area displays a table with three columns and six rows. The first column is labeled 'Period.Period.Level 1', the second is 'Accounts', and the third is 'Period.Period.Level 2'. The data is as follows:

Period.Period.Level 1	Accounts	Period.Period.Level 2
1 Q2		2313 Apr
2 Q2		3641 Apr
3 Q2		1211 Apr
4 Q1		4301 Jan
5 Q1		4766 Jan
6 Q1		1594 Jan

The 'APPLIED STEPS' pane on the right lists the following steps:

- Source
- Navigation
- Added Items
- Filtered Rows
- Added Items1
- Filtered Rows1
- Added Items2
- Removed Columns
- Added Items3
- Removed Columns1
- Added Items4** (highlighted)

At the bottom left, it says '3 COLUMNS, 6 ROWS Column profiling based on top 1000 rows'. At the bottom right, it says 'PREVIEW DOWNLOADED AT 2:46 PM'.

10. Navigate to the dimension level you want, select it, and then select **OK** to add the dimension level to the result.

The screenshot shows the Power Query Editor with the 'Add Items' dialog box open. The dialog box contains a list of dimensions and measures. Under the 'Market' dimension, the 'Market[1]' item is selected. The 'OK' button at the bottom right of the dialog box is highlighted with a yellow box.

The 'APPLIED STEPS' pane on the right shows the following steps:

- Source
- Navigation
- Added Items
- Filtered Rows
- Added Items1
- Added Items2
- Filtered Rows1
- Removed Columns
- Added Items3** (highlighted)

At the bottom left, it says '3 COLUMNS, 2 ROWS Column profiling based on top 1000 rows'. At the bottom right, it says 'PREVIEW DOWNLOADED AT 2:46 PM'.

11. The new dimension level is added to the result.

The screenshot shows the Power Query Editor interface with the 'Cube Tools' tab selected. A query named 'Demo Basic' is displayed, showing a table with three columns: 'Period.Period.Level 1', '1..2 Accounts', and 'Market.[Market].Level 1'. The table contains 18 rows of data. The 'APPLIED STEPS' pane on the right shows a step named 'Added Items2'.

Period.Period.Level 1	1..2 Accounts	Market.[Market].Level 1
1 Q2	2313 Apr	East
2 Q2	3641 Apr	West
3 Q2	1211 Apr	South
4 Q2	2369 May	East
5 Q2	4280 May	West
6 Q2	1780 May	South
7 Q2	2594 Jun	East
8 Q2	3528 Jun	West
9 Q2	1202 Jun	South
10 Q1	4301 Jan	East
11 Q1	4766 Jan	West
12 Q1	1594 Jan	South
13 Q1	3382 Feb	East
14 Q1	3879 Feb	West
15 Q1	1455 Feb	South
16 Q1	2737 Mar	East
17 Q1	4272 Mar	West
18 Q1	1425 Mar	South

12. Apply a filter to this dimension level, as needed.

The screenshot shows the Power Query Editor interface with the 'Cube Tools' tab selected. A query named 'Demo Basic' is displayed, showing a table with three columns. A filter dialog is open over the table, specifically for the 'Market.[Market].Level 1' column. The dialog shows a list of filters: 'Select All' (checked), 'East' (checked), 'South' (unchecked), and 'West' (unchecked). The 'APPLIED STEPS' pane on the right shows a step named 'Added Items2'.

13. Observe the result.

Untitled - Power Query Editor

Queries [1]

Demo Basic

```
= Table.SelectRows(#"Added Items2", each Cube.AttributeMemberId([#"Market.[Market].Level 1"]) = "[East]" meta [DisplayName = "Period.Period.Level 1", OriginalName = "Period.Period.Level 1", Type = "Table", ValueCount = 2])
```

	A [¶] Period.Period.Level 1	A [¶] Accounts	A [¶] Period.Period.Level 2	A [¶] Market.[Market].Level 1
1	Q2	2313	Apr	East
2	Q1	4301	Jan	East

4 COLUMNS, 2 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 2:46 PM

Query Settings

PROPERTIES
Name: Demo Basic
All Properties

APPLIED STEPS

- Source
- Navigation
- Added Items
- Filtered Rows
- Added Items1
- Added Items2
- Filtered Rows1

Applying your changes and loading the data

- When you've added all the dimension levels you want and have set all the required filters, select **Close** in the upper right corner to close the editor.

Untitled - Power Query Editor

Queries [1]

Demo Basic

```
= Table.SelectRows(#"Added Items2", each Cube.AttributeMemberId([#"Market.[Market].Level 1"]) = "[East]" meta [DisplayName = "Period.Period.Level 2", OriginalName = "Period.Period.Level 2", Type = "Table", ValueCount = 30])
```

	A [¶] Period.Period.Level 2	A [¶] Market.[Market].Level 1	A [¶] Market.[Market].Level 2	A [¶] Product.[Product].Level 1	A [¶] Product.[Product].Level 2
1	4 Apr	East	New_York	Audio	Stereo
2	0 Apr	East	New_York	Audio	Compact_Disc
3	0 Apr	East	New_York	Visual	Television
4	2 Apr	East	New_York	Visual	VCR
5	9 Apr	East	New_York	Visual	Camera
6	9 Apr	East	Boston	Audio	Stereo
7	8 Apr	East	Boston	Audio	Compact_Disc
8	0 Apr	East	Boston	Visual	Television
9	8 Apr	East	Boston	Visual	VCR
10	6 Apr	East	Boston	Visual	Camera
11	9 Apr	East	Chicago	Audio	Stereo
12	4 Apr	East	Chicago	Audio	Compact_Disc
13	5 Apr	East	Chicago	Visual	Television
14	5 Apr	East	Chicago	Visual	VCR
15	8 Apr	East	Chicago	Visual	Camera
16	7 Apr	West	San_Francisco	Audio	Stereo
17	5 Apr	West	San_Francisco	Audio	Compact_Disc
18	0 Apr	West	San_Francisco	Visual	Television
19	6 Apr	West	San_Francisco	Visual	VCR
20	0 Apr	West	San_Francisco	Visual	Camera
21	7 Apr	West	Seattle	Audio	Stereo
22	8 Apr	West	Seattle	Audio	Compact_Disc
23	4 Apr	West	Seattle	Visual	Television
24	9 Apr	West	Seattle	Visual	VCR
25	7 Apr	West	Seattle	Visual	Camera
26	2 Apr	West	Denver	Audio	Stereo
27	5 Apr	West	Denver	Audio	Compact_Disc
28	6 Apr	West	Denver	Visual	Television
29	5 Apr	West	Denver	Visual	VCR
30					

7 COLUMNS, 100 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 2:46 PM

Query Settings

PROPERTIES
Name: Demo Basic
All Properties

APPLIED STEPS

- Source
- Navigation
- Added Items
- Filtered Rows
- Added Items1
- Added Items2
- Removed Columns
- Added Items3
- Removed Columns1
- Added Items4
- Added Items5
- Added Items6
- Filtered Rows2

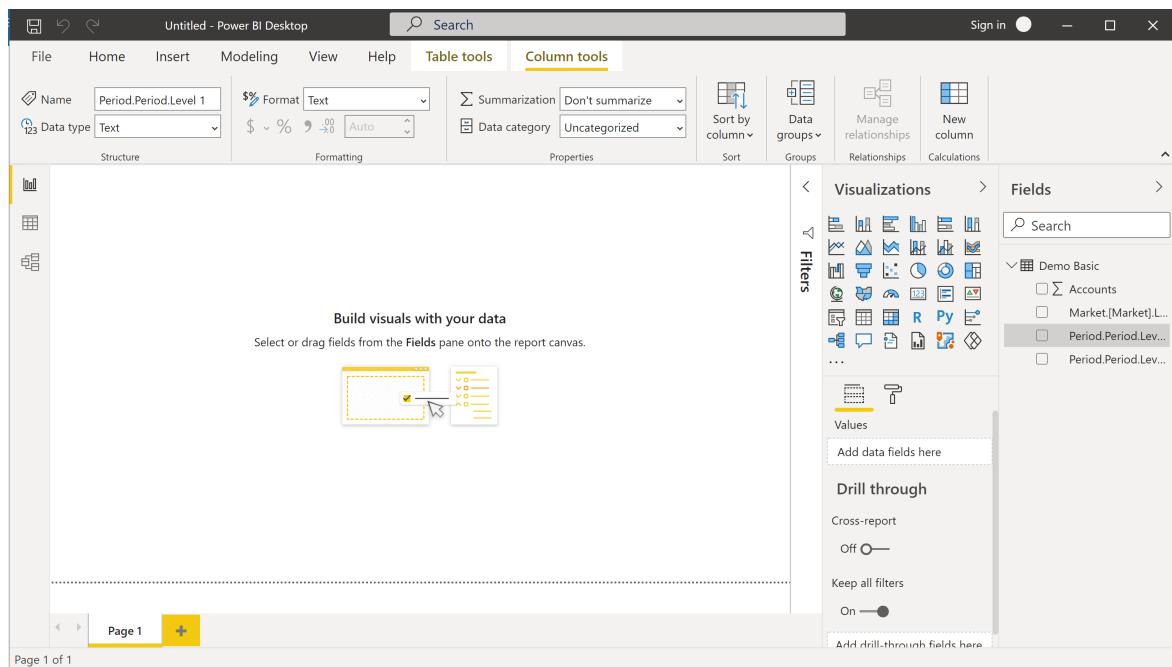
- Select **Yes** to apply your changes.

The screenshot shows the Power Query Editor interface with a preview of 1000 rows of data. The data is joined from three tables: Period, Market, and Product. A confirmation dialog box is displayed in the center, asking 'Do you want to apply your changes now?' with the 'Yes' button highlighted.

3. Wait for the changes to be applied.

The screenshot shows the Power BI desktop interface with the 'Table tools' ribbon selected. A 'Load' dialog box is open, showing the status 'Evaluating...'. The Fields pane on the right displays the new dimension levels: Accounts, Period.Period, and Period.PeriodLevel.

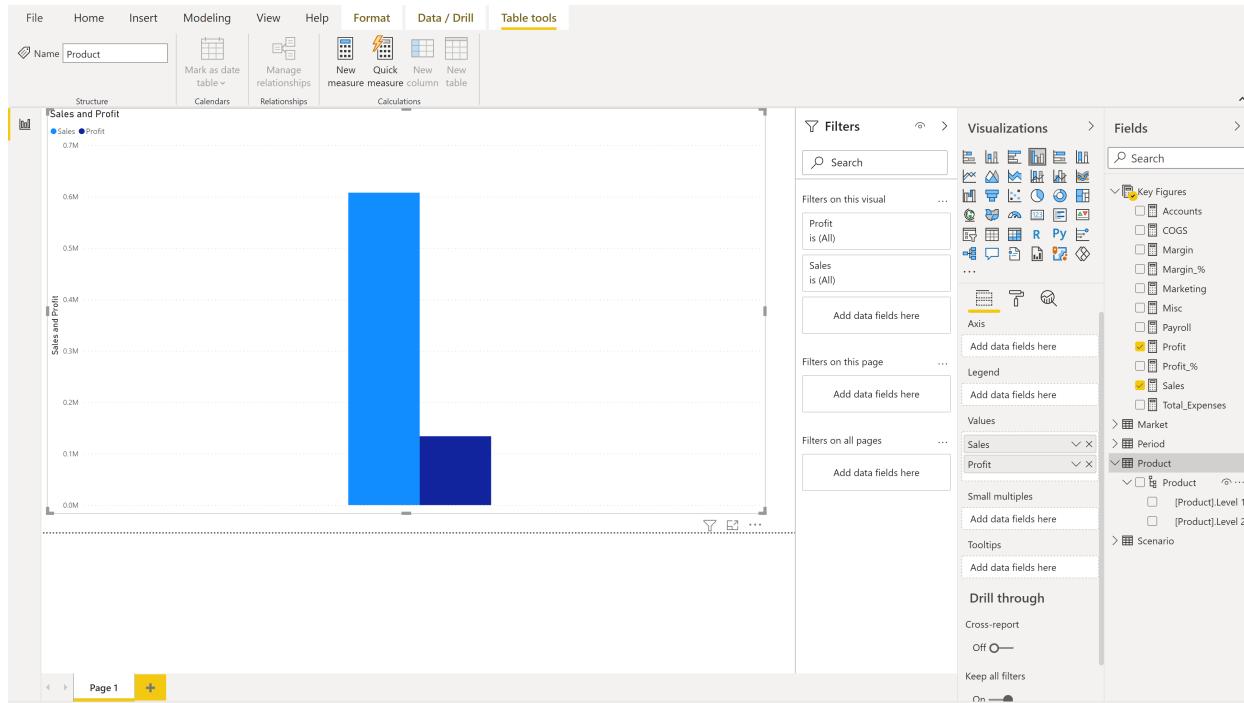
4. Observe the new dimension levels in the Fields pane.



You're now ready to create reports and visualizations.

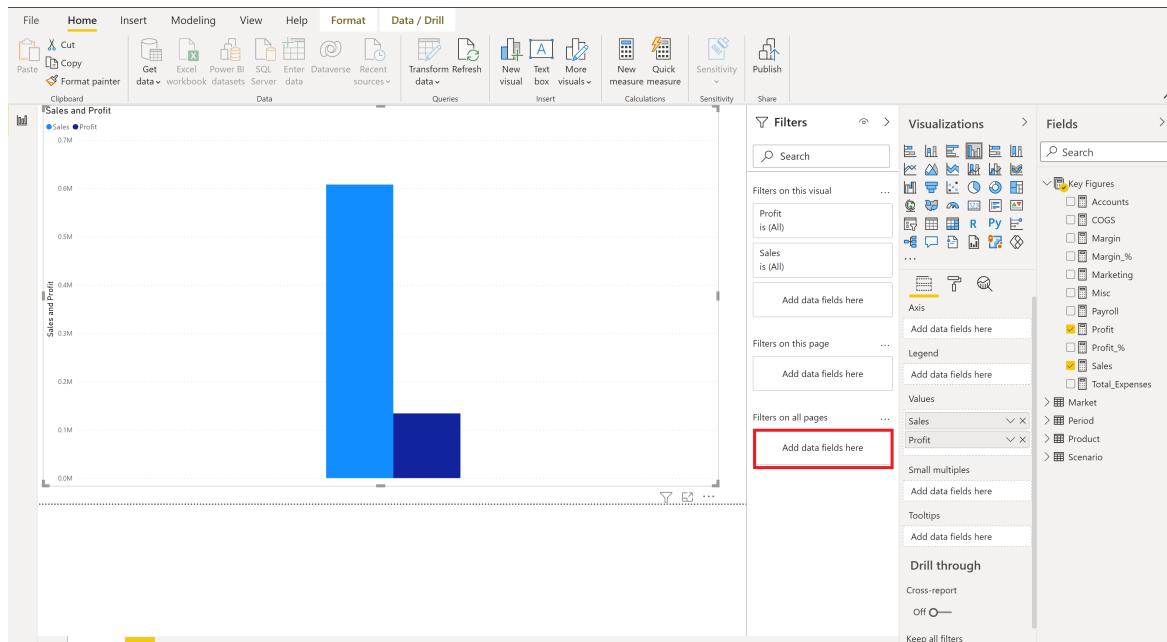
Iterative filter application when adding dimension levels in DirectQuery mode

When interacting in DirectQuery mode with a multidimensional cube data source (like Oracle's Essbase), Power BI displays the cube's dimensions and levels in the **Fields** pane.



To view and filter based on dimension members:

1. Drag-and-drop a dimension level from the Fields pane over to the Filters pane. You can drag the dimension level to the **Add data fields here** area under **Filters on this visual**, **Filters on this page**, or **Filters on all pages**, depending on your needs.

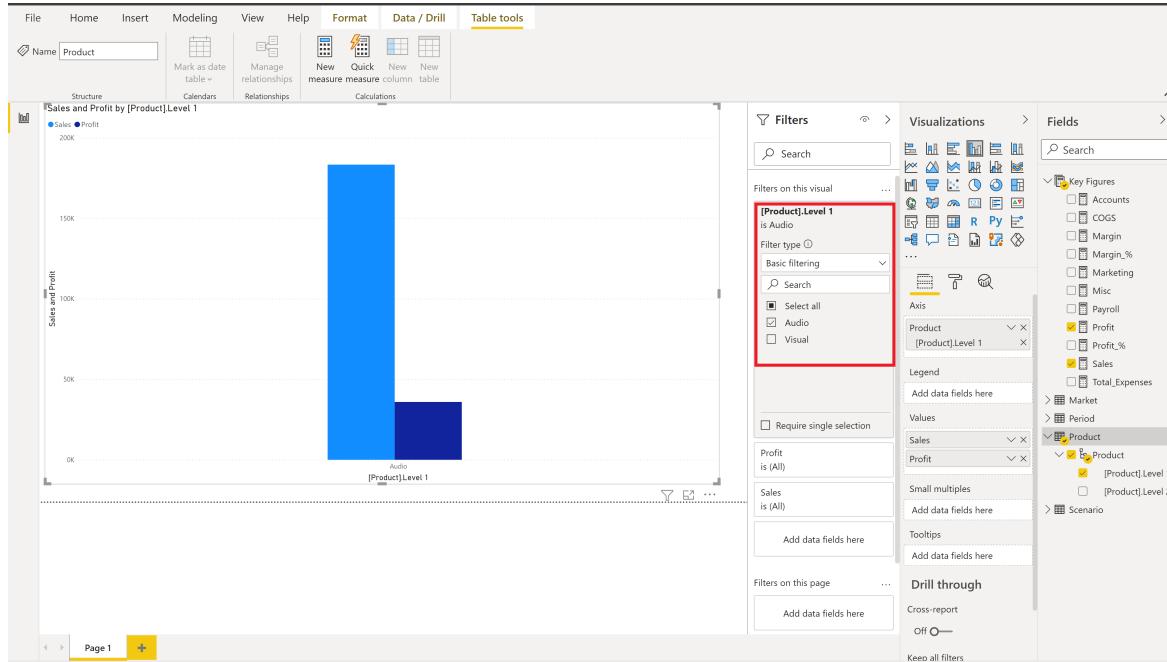


2. Once a dimension's level is in the **Filter** pane and the filter type is set to **Basic filtering**, you'll notice that the members of that dimension's level are displayed as a list of available filters.

3. You can check the members you want to include in your result.

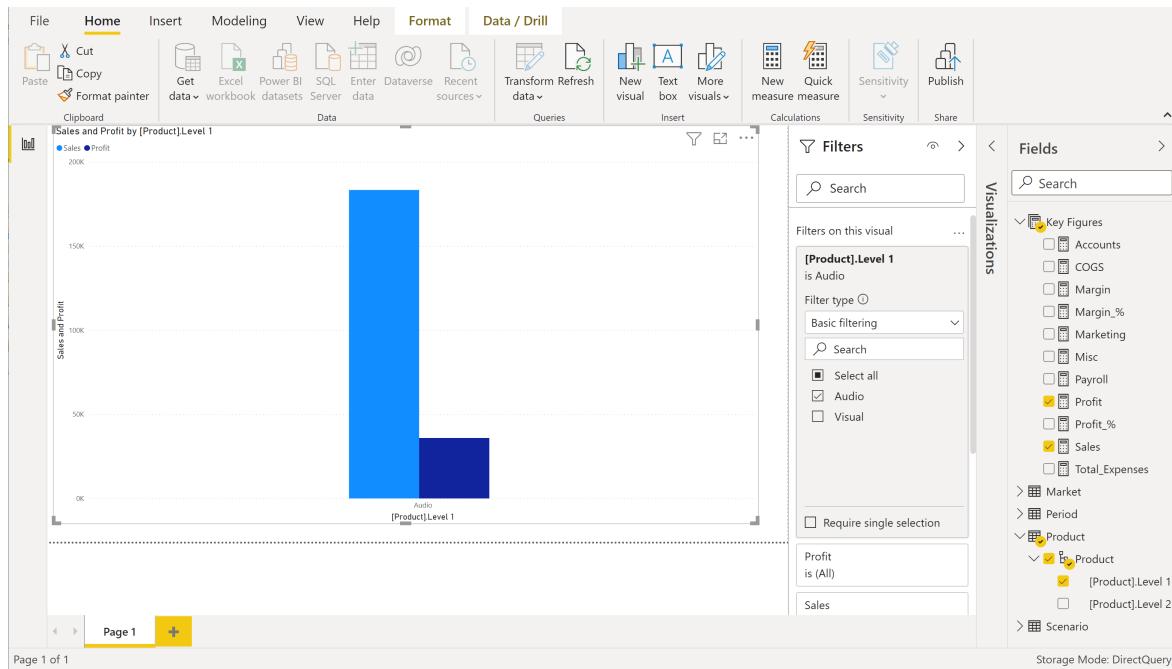
Or you can select the **Select all** option, then uncheck the members you don't want to include in your result.

Type some characters in the search field for that filter to find members in the list.

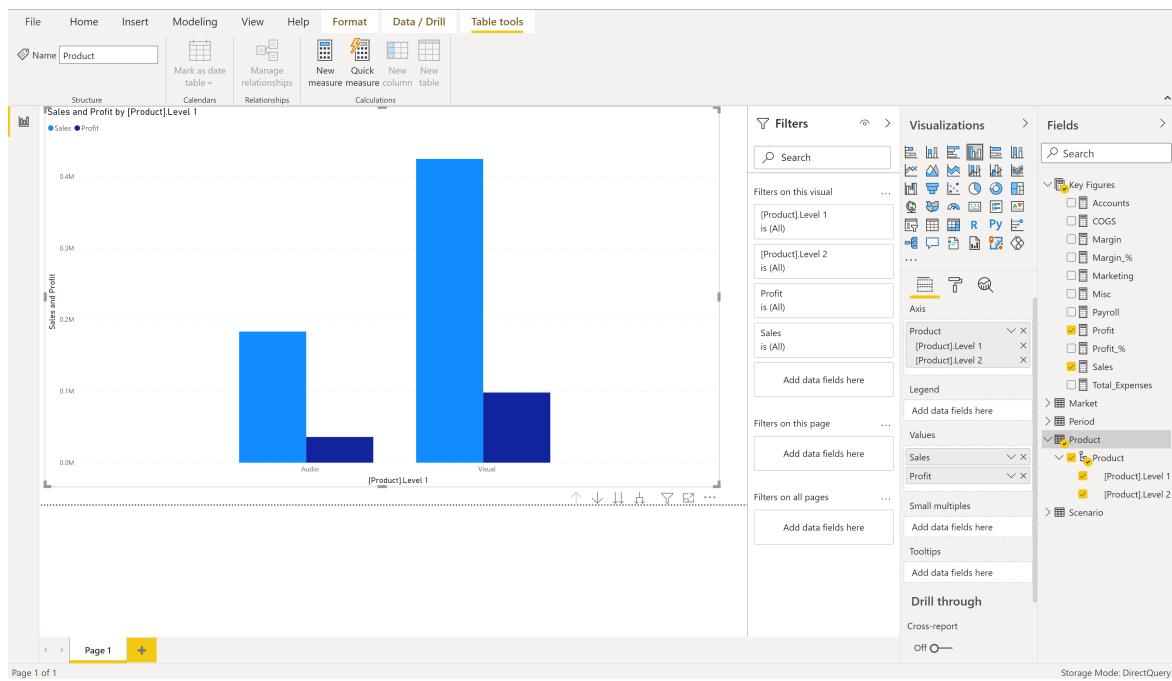


4. When you have filters for two or more levels of the same dimension, you'll notice that selecting members from a higher level in the dimension changes the members available in the lower levels of that dimension.

This cross highlighting/filtering behavior can be disabled by checking the **Disabling cross highlighting/filtering by default** option, as described in [Query reductions options](#).



5. When you've finished choosing the members you want in the dimension level filter, it's a good time to add that dimension level to your visualization. Check the matching dimension level in the **Fields** pane and it's then added to your current visualization.



For more information about adding filters, go to [Add a filter to a report in Power BI](#).

Troubleshooting

This section outlines common issues that you might come across, and includes troubleshooting steps to address the issues.

Connection issues

Symptom 1

Power BI Desktop returns the error message "Unable to connect to the remote server".

Resolution

1. Ensure the Essbase Analytic Provider Services (APS) server is configured correctly for the Provider

Servers and Standalone Servers in the Essbase Administration Service (EAS) console. More information: [Configuring Essbase Clusters](#)

2. Ensure that the URL is correct.

- Check to ensure the hostname and or IP address is correct.
- Check to ensure the provided port is correct.
- Check to ensure the http (not https) protocol is specified.
- Check to ensure the case is correct for the /aps/XMLA path in the URL.

3. If there's a firewall between Power BI Desktop and the provided hostname, check to ensure the provided hostname and port can pass outbound through your firewall.

Validation

Trying to connect again won't show the error and the Cube and member list is in the navigation pane. You can also select and display in preview in Import mode.

Symptom 2

Power BI Desktop returns the error message "We couldn't authenticate with the credentials provided. Please try again."

Resolution

Ensure the provided username and password are correct. Reenter their values carefully. The password is case-sensitive.

Validation

After correcting the username and password, you should be able to display the members and the value in the preview or be able to load the data.

Symptom 3

Power BI Desktop returns the error message "Data at the root level is invalid. Line 1, position 1."

Resolution

Ensure the Essbase Analytic Provider Services (APS) server is configured correctly for the Provider Servers and Standalone Servers in the Essbase Administration Service (EAS) console. More information: [Configuring Essbase Clusters](#).

Validation

Trying to connect again won't show the error and the Cube and member list is displayed in the navigation pane. You can also select and display in the preview in Import mode.

Symptom 4

Once successfully connected to the Oracle Essbase Analytic Provider Services (APS) server, there are servers listed below the URL node in the data source navigator. However, when you expand a server node, no applications are listed below that server node.

Resolution

We recommend configuring the Oracle Hyperion server to define the provider and standalone servers through the Essbase Administration Service (EAS) console. Refer to section Addendum: [Registering Provider and Standalone Servers in Essbase Administration Service \(EAS\) Console](#).

Validation

Trying to connect again won't show the error and you can see the Cube and member list in the navigation pane. You can also select and display in the preview in Import mode.

Time out or large data issue

Symptom 1

Power Query returns the error message "The operation has timed out"

Resolution

1. Ensure the network is stable and there's a reliable network path to the Essbase Analytic Provider Services (APS) server provided in the data source URL.
2. If there's a possibility that the query to the service could return a large amount of data, specify a long (or longer) command timeout interval. If possible, add filters to your query to reduce the amount of data returned. For example, select only specific members of each dimension you want returned.

Validation

Retry to load the data and if the problem persists, try to increase to a longer timeout interval or filter the data further. If the problem still persists, try the resolution on [Symptoms 3](#).

Symptom 2

The query returns the error message "Internal error: Query is allocating too large memory (> 4GB) and cannot be executed. Query allocation exceeds allocation limits."

Resolution

The query you're trying to execute is producing results greater than the Oracle Essbase server can handle. Supply or increase the filters on the query to reduce the amount of data the server will return. For example, select specific members for each level of each dimension or set numeric limits on the value of measures.

Validation

Retry to load the data and if the problem persists, try to increase to a longer timeout interval or filter the data further. If the problem still persists, try the resolution on [Symptoms 3](#).

Symptom 3

Essbase Analytic Provider Services (APS) or Essbase server indicates a large number of connections with long running sessions.

Resolution

When the connectivity mode is DirectQuery, it's easy to select measures or dimension levels to add to the selected visualization. However, each new selection creates a new query and a new session to the Essbase Analytic Provider Services (APS)/Essbase server. There are a few ways to ensure a reduced number of queries or to reduce the size of each query result. Review [Performance Considerations](#) to reduce the number of times the server is queried and to also reduce the size of query results.

Validation

Retry to load the data.

Key not matching when running MDX

Symptom

An MDX statement returns the error message "The key didn't match any rows in the table".

Resolution

It's likely that the value or the case of the Server and Application fields don't match. Select the **Edit** button and correct the value and case of the Server and Application fields.

Validation

Retry to load the data.

Unable to get cube issue - MDX

Symptom

An MDX statement returns the error message "Unable to get the cube name from the statement. Check the format used for specifying the cube name".

Resolution

Ensure the database name in the MDX statement's FROM clause is fully qualified with the application and database name, for example, [Sample.Basic]. Select the **Edit** button and correct the fully qualified database name in the MDX statement's FROM clause.

Validation

Retry to load the data.

Essbase Error (1260060) issue - MDX

Symptom

An MDX statement returns the error message "Essbase Error (1260060): The cube name XXXX does not match with current application/database"

Resolution

Ensure the application name and the fully qualified database name in the FROM clause match. Select the **Edit** button and correct either the application name or the fully qualified database name in the MDX statement's FROM clause

Validation

Retry to load the data.

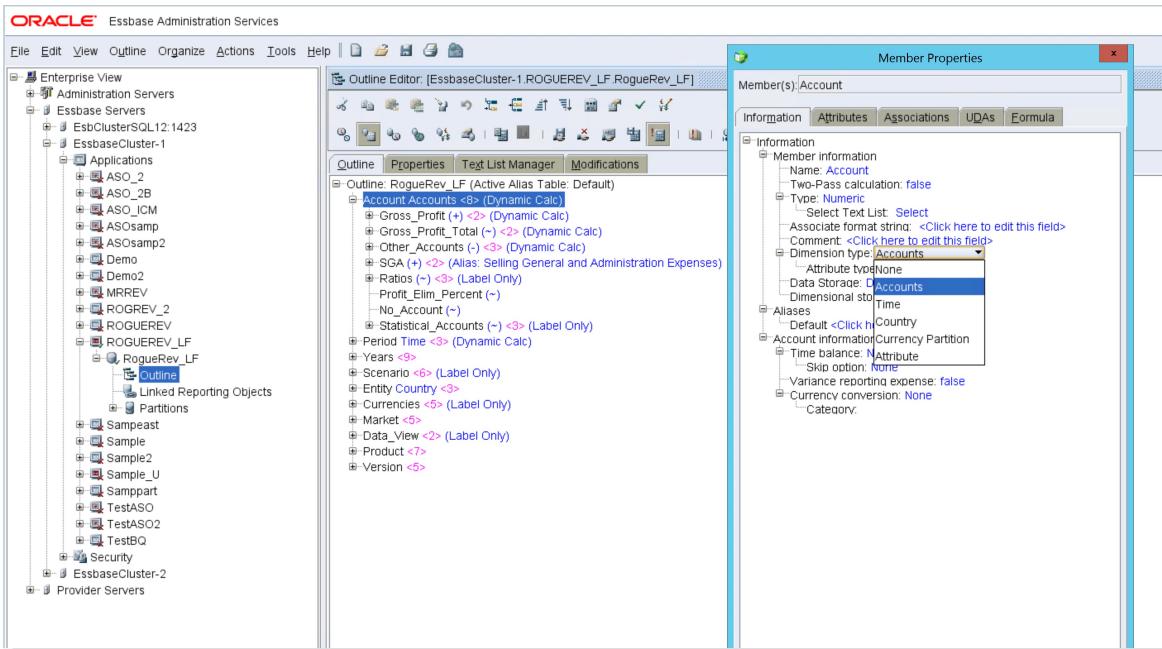
Essbase Error (1200549): Repeated dimension [Measures] in MDX query

Symptom

Loading a dimension returns the error message "Essbase Error (1200549): Repeated dimension [Measures] in MDX query".

Resolution

1. Sign in to the Essbase server, open the Essbase Administration Services Console and sign in with an admin user (or whoever has permissions over the problematic database).
2. Navigate to the Essbase server > application > database with the problematic "Measures" dimension.
3. Unlock the outline of the database and edit it.
4. Determine which dimension should be the "Accounts" dimension type. Right-click it and select **Edit member properties....**
5. Select the Dimension Type field and set it to **Accounts**. Select **OK**.



6. Verify and Save the outline.

Validation

Retry to load the dimension.

Excel

1/15/2022 • 7 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Anonymous (online) Basic (online) Organizational account (online)
Function Reference Documentation	Excel.Workbook Excel.CurrentWorkbook

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

To connect to a legacy workbook (such as .xls or .xlsb), the Access Database Engine OLEDB (or ACE) provider is required. To install this provider, go to the [download page](#) and install the relevant (32 bit or 64 bit) version. If you don't have it installed, you'll see the following error when connecting to legacy workbooks:

The 'Microsoft.ACE.OLEDB.12.0' provider is not registered on the local machine. The 32-bit (or 64-bit) version of the Access Database Engine OLEDB provider may be required to read this type of file. To download the client software, visit the following site: <https://go.microsoft.com/fwlink/?LinkId=285987>.

ACE can't be installed in cloud service environments. So if you're seeing this error in a cloud host (such as Power Query Online), you'll need to use a gateway that has ACE installed to connect to the legacy Excel files.

Capabilities Supported

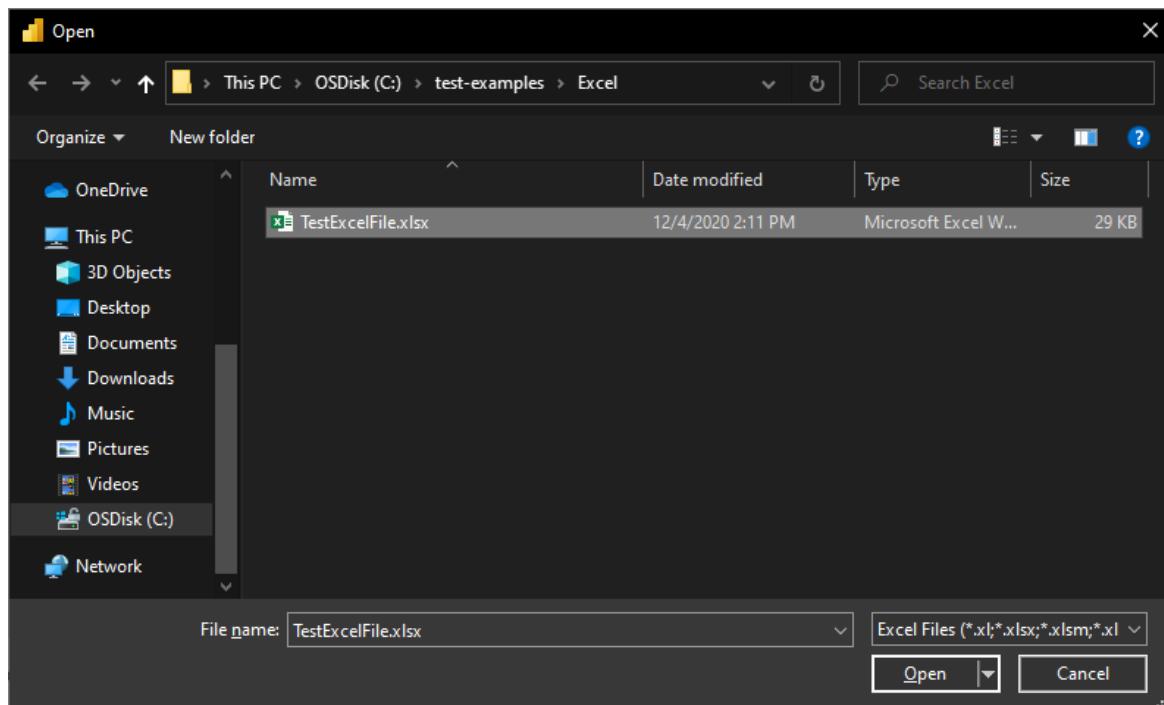
- Import

Connect to an Excel workbook from Power Query Desktop

To make the connection from Power Query Desktop:

1. Select the **Excel** option in the connector selection.

2. Browse for and select the Excel workbook you want to load. Then select **Open**.



If the Excel workbook is online, use the [Web connector](#) to connect to the workbook.

3. In **Navigator**, select the workbook information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

A screenshot of the Power Query Navigator interface. On the left, there's a tree view under 'Display Options' showing 'TestExcelFile.xlsx [3]' with 'Financial_Sample_1' checked. To the right, a preview pane titled 'Financial_Sample_1' shows a table with four columns: Segment, Country, Product, and Discount Band. The data includes rows for Government segments in USA, France, Germany, and Canada, with products like Paseo, Amarilla, Velo, and VTT, and discount bands ranging from Low to Medium. At the bottom are 'Load', 'Transform Data', and 'Cancel' buttons.

Connect to an Excel workbook from Power Query Online

To make the connection from Power Query Online:

1. Select the **Excel** option in the connector selection.
2. In the Excel dialog box that appears, provide the path to the Excel workbook.

3. If necessary, select an on-premises data gateway to access the Excel workbook.
4. If this is the first time you've accessed this Excel workbook, select the authentication kind and sign in to your account (if needed).
5. In **Navigator**, select the workbook information you want, and then **Transform Data** to continue transforming the data in Power Query Editor.

ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7
Segment	Country	Product	Discount Band	Units Sold	Manufacturing Pri...	Sale Price
Government	United States of Ameri...	Paseo	Low	3450	10	350
Government	France	Amarilla	None	2750	260	350
Government	Germany	Velo	Low	2966	120	350
Government	Germany	Amarilla	Low	2966	260	350
Government	Germany	Velo	Low	2877	120	350
Government	Germany	VTT	Low	2877	250	350
Government	Canada	Carretera	Low	2852	3	350
Government	Canada	Paseo	Low	2852	10	350
Government	France	Amarilla	Medium	2876	260	350
Government	France	Carretera	Low	2155	3	350
Government	France	Paseo	Low	2155	10	350
Government	France	Velo	Low	2177	120	350
Government	France	VTT	Low	2177	250	350
Government	Mexico	VTT	Low	1940	250	350

Troubleshooting

Numeric precision (or "Why did my numbers change?")

When importing Excel data, you may notice that certain number values seem to change slightly when imported into Power Query. For example, if you select a cell containing 0.049 in Excel, this number is displayed in the formula bar as 0.049. But if you import the same cell into Power Query and select it, the preview details display it as 0.049000000000000002 (even though in the preview table it's formatted as 0.049). What's going on here?

The answer is a bit complicated, and has to do with how Excel stores numbers using something called *binary floating-point notation*. The bottom line is that there are certain numbers that Excel can't represent with 100% precision. If you crack open the .xlsx file and look at the actual value being stored, you'll see that in the .xlsx file, 0.049 *is* actually stored as 0.049000000000000002. This is the value Power Query reads from the .xlsx, and thus the value that appears when you select the cell in Power Query. (For more information on numeric precision in Power Query, go to the "Decimal number" and "Fixed decimal number" sections of [Data types in Power Query](#).)

Connecting to an online Excel workbook

If you want to connect to an Excel document hosted in Sharepoint, you can do so via the [Web](#) connector in Power BI Desktop, Excel, and Dataflows, and also with the Excel connector in Dataflows. To get the link to the file:

1. Open the document in Excel Desktop.
2. Open the **File** menu, select the **Info** tab, and then select **Copy Path**.
3. Copy the address into the **File Path or URL** field, and remove the `?web=1` from the end of the address.

Legacy ACE connector

Power Query reads legacy workbooks (such as .xls or .xslb) use the Access Database Engine (or ACE) OLEDB provider. Because of this, you may come across unexpected behaviors when importing legacy workbooks that don't occur when importing OpenXML workbooks (such as .xlsx). Here are some common examples.

Unexpected value formatting

Because of ACE, values from a legacy Excel workbook might be imported with less precision or fidelity than you expect. For example, imagine your Excel file contains the number 1024.231, which you've formatted for display as "1,024.23". When imported into Power Query, this value is represented as the text value "1,024.23" instead of as the underlying full-fidelity number (1024.231). This is because, in this case, ACE doesn't surface the underlying value to Power Query, but only the value as it's displayed in Excel.

Unexpected null values

When ACE loads a sheet, it looks at the first eight rows to determine the data types of the columns. If the first eight rows aren't representative of the later rows, ACE may apply an incorrect type to that column and return nulls for any value that doesn't match the type. For example, if a column contains numbers in the first eight rows (such as 1000, 1001, and so on) but has non-numerical data in later rows (such as "100Y" and "100Z"), ACE concludes that the column contains numbers, and any non-numeric values are returned as null.

Inconsistent value formatting

In some cases, ACE returns completely different results across refreshes. Using the example described in the [formatting section](#), you might suddenly see the value 1024.231 instead of "1,024.23". This difference can be caused by having the legacy workbook open in Excel while importing it into Power Query. To resolve this problem, close the workbook.

Missing or incomplete Excel data

Sometimes Power Query fails to extract all the data from an Excel Worksheet. This failure is often caused by the Worksheet having **incorrect dimensions** (for example, having dimensions of `A1:C200` when the actual data occupies more than three columns or 200 rows).

How to diagnose incorrect dimensions

To view the dimensions of a Worksheet:

1. Rename the .xlsx file with a .zip extension.
2. Open the file in File Explorer.
3. Navigate into `xl\worksheets`.
4. Copy the xml file for the problematic sheet (for example, `Sheet1.xml`) out of the zip file to another location.
5. Inspect the first few lines of the file. If the file is small enough, open it in a text editor. If the file is too large to be opened in a text editor, run the following command from a Command Prompt: `more Sheet1.xml`.
6. Look for a `<dimension .../>` tag (for example, `<dimension ref="A1:C200" />`).

If your file has a dimension attribute that points to a single cell (such as `<dimension ref="A1" />`), Power Query uses this attribute to find the starting row and column of the data on the sheet.

However, if your file has a dimension attribute that points to multiple cells (such as `<dimension ref="A1:AJ45000" />`), Power Query uses this range to find the starting row and column **as well as the ending row and column**. If this range doesn't contain all the data on the sheet, some of the data won't be

loaded.

How to fix incorrect dimensions

You can fix issues caused by incorrect dimensions by doing one of the following actions:

- Open and resave the document in Excel. This action will overwrite the incorrect dimensions stored in the file with the correct value.
- Ensure the tool that generated the Excel file is fixed to output the dimensions correctly.
- Update your M query to ignore the incorrect dimensions. As of the December 2020 release of Power Query, `Excel.Workbook` now supports an `InferSheetDimensions` option. When true, this option will cause the function to ignore the dimensions stored in the Workbook and instead determine them by inspecting the data.

Here's an example of how to provide this option:

```
Excel.Workbook(File.Contents("C:\MyExcelFile.xlsx"), [DelayTypes = true, InferSheetDimensions = true])
```

Sluggish or slow performance when loading Excel data

Slow loading of Excel data can also be caused by incorrect dimensions. However, in this case, the slowness is caused by the dimensions being much larger than they need to be, rather than being too small. Overly large dimensions will cause Power Query to read a much larger amount of data from the Workbook than is actually needed.

To fix this issue, you can refer to [Locate and reset the last cell on a worksheet](#) for detailed instructions.

Poor performance when loading data from SharePoint

When retrieving data from Excel on your machine or from SharePoint, consider both the volume of the data involved, as well as the complexity of the workbook.

You'll notice performance degradation when retrieving very large files from SharePoint. However, this is only one part of the problem. If you have significant business logic in an Excel file being retrieved from SharePoint, this business logic may have to execute when you refresh your data, which could cause complicated calculations. Consider aggregating and pre-calculating data, or moving more of the business logic out of the Excel layer and into the Power Query layer.

Errors when using the Excel connector to import CSV files

Even though CSV files can be opened in Excel, they're not Excel files. Use the [Text/CSV connector](#) instead.

FHIR

1/15/2022 • 2 minutes to read • [Edit Online](#)

Fast Healthcare Interoperability Resources ([FHIR®](#)) is a new standard for healthcare data interoperability. Healthcare data is represented as resources such as `Patient`, `Observation`, `Encounter`, and so on, and a REST API is used for querying healthcare data served by a FHIR server. The Power Query connector for FHIR can be used to import and shape data from a FHIR server.

If you don't have a FHIR server, you can provision the [Azure API for FHIR](#).

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous Azure Active Directory

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

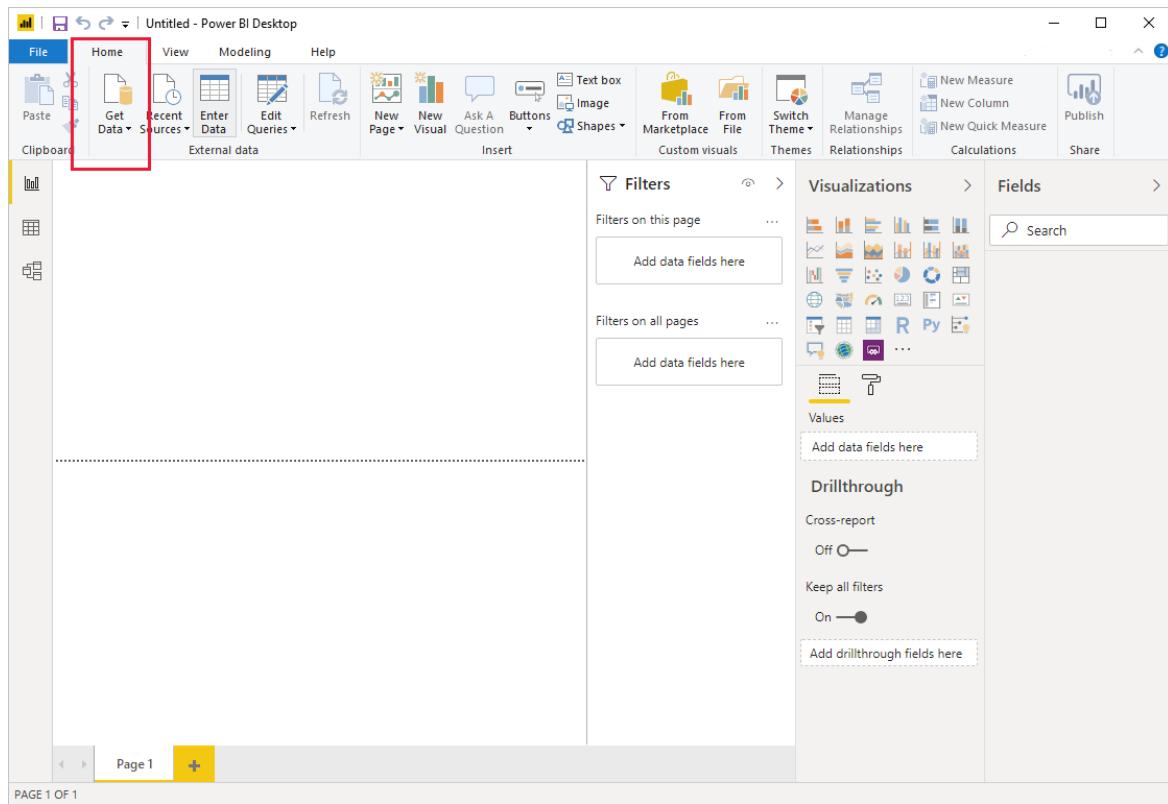
Capabilities Supported

- Import

Connect to a FHIR server from Power Query Desktop

To make a connection to a FHIR server, take the following steps:

1. Select the **Get Data** button.



2. Select Other > FHIR, and then select Connect.

Get Data

Search

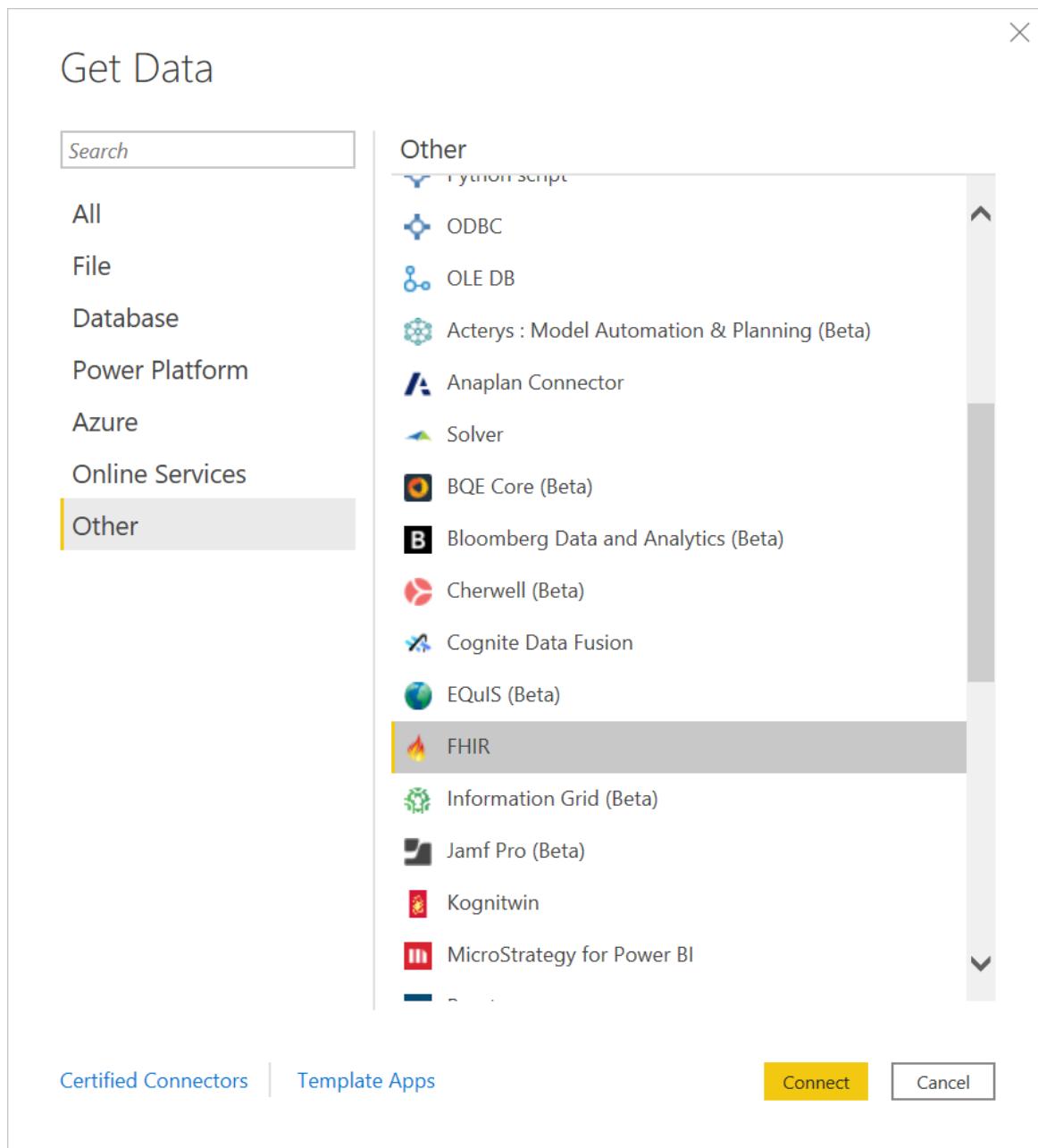
- All
- File
- Database
- Power Platform
- Azure
- Online Services
- Other

Other

- Python Script
- ODBC
- OLE DB
- Acterys : Model Automation & Planning (Beta)
- Anaplan Connector
- Solver
- BQE Core (Beta)
- B Bloomberg Data and Analytics (Beta)
- C Cherwell (Beta)
- D Cognite Data Fusion
- E EQuIS (Beta)
- F FHIR
- G Information Grid (Beta)
- H Jamf Pro (Beta)
- I Kognitwin
- J MicroStrategy for Power BI

Certified Connectors | Template Apps

Connect Cancel



3. Enter the URL for your FHIR server.

From Fhir.Contents

URL ⓘ

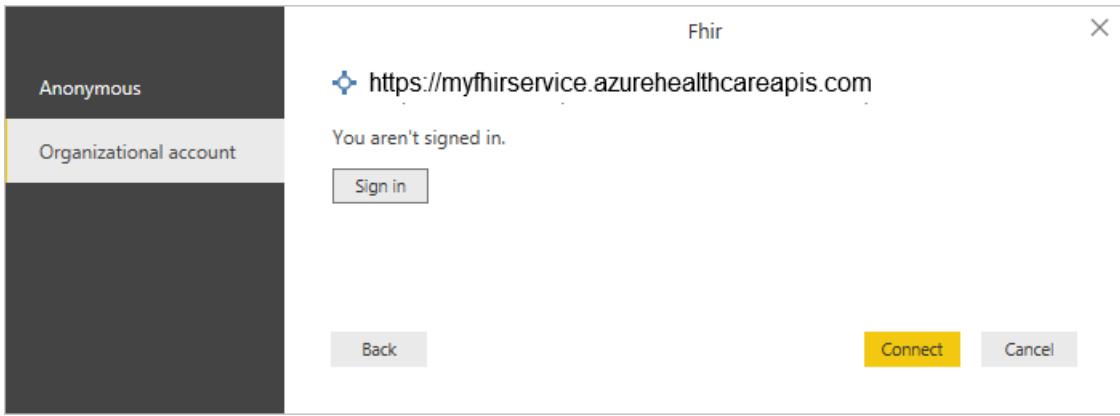
Query (optional)
Example: Patient?address-postalcode=98052

OK Cancel

You can optionally enter an initial query for the FHIR server, if you know exactly what data you're looking for.

Select OK to proceed.

4. Decide on your authentication scheme.



The connector supports "Anonymous" for FHIR servers with no access controls (for example, public test servers (like <http://test.fhir.org/r4>) or Azure Active Directory authentication. Go to [FHIR connector authentication](#) for details.

5. Select the resources you're interested in.

id	meta	implicitRules	language	<re
e11037dd-9b76-42bb-92cb-9109a7b4ab71	Record		null	null Pat
69d45781-e1a9-4585-8403-909d3d0119e3	Record		null	null Pat
c1a6c861-a4f0-4699-aeaa-cf352fed7d11	Record		null	null Pat
72812a51-49e5-4149-a4e4-220dab513a31	Record		null	null Pat
10040468-4d8d-47ef-b230-9d2104258685	Record		null	null Pat
2d59118f-1ab1-4ed0-8805-cfb8c6f8225	Record		null	null Pat
a4a14bb0-dc7e-4dfa-af5b-03eed5462445	Record		null	null Pat
fadd999d-fc4b-4373-b6ee-a9c2bc9932cd	Record		null	null Pat
9ae8155e-d925-4416-b428-263368cd4626	Record		null	null Pat
4ae59bcc-92d8-4e59-9bb1-627c533626e3	Record		null	null Pat
24e65740-3cd6-4c29-86c4-64481ba4ac1e	Record		null	null Pat
8c85b049-7b60-4ed8-a199-d4f878d682e0	Record		null	null Pat
274c739c-3761-4187-b5e2-dd6adbe20c4f	Record		null	null Pat
23e32e02-5b94-4ce0-9f9a-7fd55005233c	Record		null	null Pat
8b40afed-4da3-49d1-aeff-4aadef61c9ae	Record		null	null Pat
221b5dd1-6ff1-4d14-939d-e476979be467	Record		null	null Pat
a1bd87a8-67d5-45e5-93b7-3d79132b6525	Record		null	null Pat
efcdb09b-7964-49af-81ca-fed92139d0ab	Record		null	null Pat
ee1fd943-8357-46ff-8dcf-cc3637c3c719	Record		null	null Pat
7d3941c7-0492-4785-a0a0-0484ff695319	Record		null	null Pat

The data in the preview has been truncated due to size limits.

Select **Transform Data** to shape the data.

6. Shape the data as needed, for example, expand the postal code.

The screenshot shows the Power Query Editor interface with the 'Patient' query selected. A context menu is open over the 'address' column, with the 'Expand' option highlighted. The 'Applied Steps' pane on the right shows the 'Navigation' step applied to the source. The status bar at the bottom indicates 'PREVIEW DOWNLOADED AT 12:30 PM'.

7. Save the query when shaping is complete.

The screenshot shows the Power Query Editor interface with the 'Patient' query selected. The 'Close & Apply' button in the ribbon is highlighted in red. The 'Applied Steps' pane on the right shows the 'Expanded address' step applied to the query. The status bar at the bottom indicates 'PREVIEW DOWNLOADED AT 12:30 PM'.

8. Create dashboards with data, for example, make a plot of the patient locations based on postal code.

The screenshot shows the Power BI Desktop interface. On the left is a map of a region with several blue dots representing data points. The top ribbon has tabs like File, Home, View, Modeling, Help, Format, and Data/Drill. The Home tab is selected. The ribbon also includes sections for External data, Insert, Custom visuals, Themes, Relationships, Calculations, and Share. In the center, there's a 'Filters' pane with three sections: 'Filters on this visual', 'Filters on this page', and 'Filters on all pages'. The 'Filters on this visual' section contains a dropdown for 'address.postalCode' with the value '(All)'. The 'Filters on this page' section has a dropdown for 'address.postalCode' which is highlighted with a red box. The 'Filters on all pages' section also has a dropdown for 'address.postalCode'. To the right is a 'Visualizations' pane showing various chart and map icons. Below it is a 'Fields' pane with a search bar and a tree view of fields under a 'Patient' category, including 'address.postalCode' (which is checked), 'active', 'birthDate', 'gender', 'id', 'implicitRules', and 'language'. At the bottom, there are navigation buttons for 'Page 1' and '+', and a status bar showing 'PAGE 1 OF 1'.

Connect to a FHIR server from Power Query Online

To make a connection to a FHIR server, take the following steps:

1. In Power Query - Choose data source, select the Other category, and then select FHIR.

The screenshot shows the 'Power Query - Choose data source' dialog. At the top, there are tabs for All categories, File, Database, Power Platform, Azure, Online services, and Other. The 'Other' tab is selected. A search bar is on the right. Below the tabs, there's a section titled 'Data sources' with eight items arranged in a grid:

- FHIR (Other)
- OData (Other)
- Odbc (Other)
- SharePoint list (Other)
- Spark (Other)
- Web API (Other)
- Web page (Other)
- Blank table (Other)
- Blank query (Other) - This item is highlighted with a red box.

At the bottom right of the dialog is a 'Cancel' button.

2. In the FHIR dialog, enter the URL for your FHIR server.



FHIR
Other

Connection settings

A_C URL *

https://myfhirserver.azurehealthcareapis.com

A_C Query

Example: Patient?address-postalcode=98052

Connection credentials

Data gateway

(none)



Authentication kind

Organizational account

You are not signed in. Please sign in.

Sign in

You can optionally enter an initial query for the FHIR server, if you know exactly what data you're looking for.

3. If necessary, include the name of your on-premises data gateway.
4. Select the **Organizational account** authentication kind, and select **Sign in**. Enter your credentials when asked.
5. Select **Next** to proceed.
6. Select the resources you're interested in.

Power Query - Choose data

A _C id	A _C meta	A _C implicitRules	A _C language	A _C <r
44f610e-96c2-4802-b857-4861f1802522	[Record]	null	null	Pat
4b0cc362-6a4f-4661-bd9b-ea21db4c0b10	[Record]	null	null	Pat
1d13e89c-3092-436b-9031-f18d2edcd00f	[Record]	null	null	Pat
3a5bc54c-6088-485a-a3bf-7c58bc3ff3ca	[Record]	null	null	Pat
6d88dc36-430c-45bf-a99f-2623cacb101d	[Record]	null	null	Pat
0b71b012-1e12-4f7d-afe7-bff593e7a7ba	[Record]	null	null	Pat
88cd3d4d-ae7d-443a-bdfd-117dac017e5a	[Record]	null	null	Pat
e77b1548-bf33-4721-8a8a-daa54c15c442	[Record]	null	null	Pat
02b9b740-618d-4978-b3d1-ef36ed8c5843	[Record]	null	null	Pat
82fb617f-0579-40e6-81ad-ae5939f1a0c7	[Record]	null	null	Pat
c63ad7c9-9106-427a-93f0-d5433b31bdd4	[Record]	null	null	Pat
371d495a-a385-4d57-80f3-37c844e1ac8f	[Record]	null	null	Pat
c8cbffff-874a-4b9f-b36b-7a2251d3a723	[Record]	null	null	Pat
8188ca80-3fb5-4b4d-b347-c73c5239924d	[Record]	null	null	Pat
478e360c-d860-40c1-928d-ef81392110cf	[Record]	null	null	Pat
d6c07704-14d6-4dd1-a5de-16bddb7029...	[Record]	null	null	Pat
51e87235-452e-4f52-9fed-b0625f07f208	[Record]	null	null	Pat
dfddcd4-9641-4756-8c10-6ab03e02229e	[Record]	null	null	Pat
7ab73a3b-e094-4fed-958e-5b8f268bc913	[Record]	null	null	Pat
518d9bc9-55ee-4cff-b299-9984593be0ad	[Record]	null	null	Pat
859e17db-bd0a-4e18-95d5-8110c74322...	[Record]	null	null	Pat
30b2d2b4-9e49-4a86-be7c-18664fcacd1d	[Record]	null	null	Pat
8bd8a546-1692-496b-9115-5e03c25c69fc	[Record]	null	null	Pat

Back Cancel Transform data

Select **Transform data** to shape the data.

7. Shape the data as needed, for example, expand the postal code.

Completed (3.90 s) Columns: 25 Rows: 99+

Step Cancel Save & close

8. Save the query when shaping is complete.

Completed (4.67 s) Columns: 25 Rows: 99+

Step Cancel Save & close

NOTE

In some cases, query folding can't be obtained purely through data shaping with the graphical user interface (GUI), as shown in the previous image. To learn more about query folding when using the FHIR connector, see [FHIR query folding](#).

Next Steps

In this article, you've learned how to use the Power Query connector for FHIR to access FHIR data. Next explore the authentication features of the Power Query connector for FHIR.

[FHIR connector authentication](#)

FHIR® and the FHIR Flame icon are the registered trademarks of HL7 and are used with the permission of HL7. Use of the FHIR trademark does not constitute endorsement of this product by HL7.

FHIR connector authentication

1/15/2022 • 2 minutes to read • [Edit Online](#)

This article explains authenticated access to FHIR servers using the Power Query connector for FHIR. The connector supports anonymous access to publicly accessible FHIR servers and authenticated access to FHIR servers using Azure Active Directory authentication. The [Azure API for FHIR](#) is secured with Azure Active Directory.

NOTE

If you are connecting to a FHIR server from an online service, such as Power BI service, you can only use an organizational account.

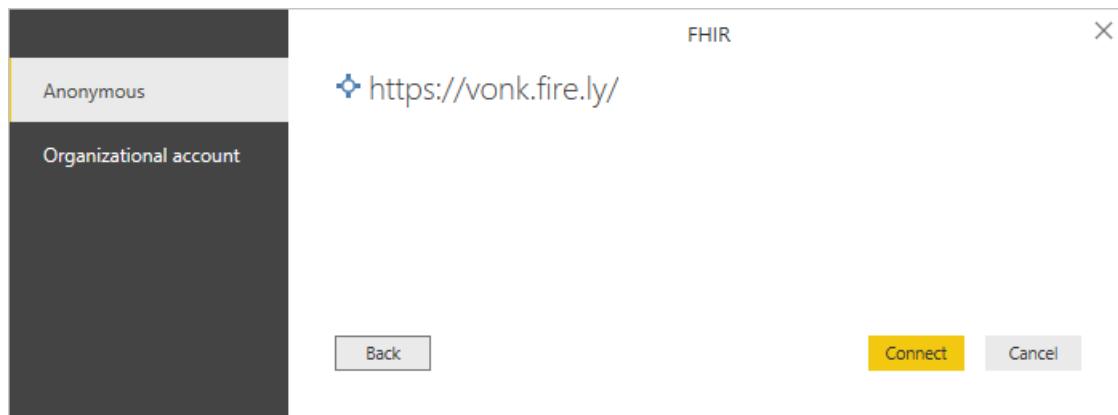
Anonymous access

There are many [publicly accessible FHIR servers](#). To enable testing with these public servers, the Power Query connector for FHIR supports the "Anonymous" authentication scheme. For example to access the public <https://vonk.fire.ly> server:

1. Enter the URL of the public Vonk server.



2. Select **Anonymous** authentication scheme.



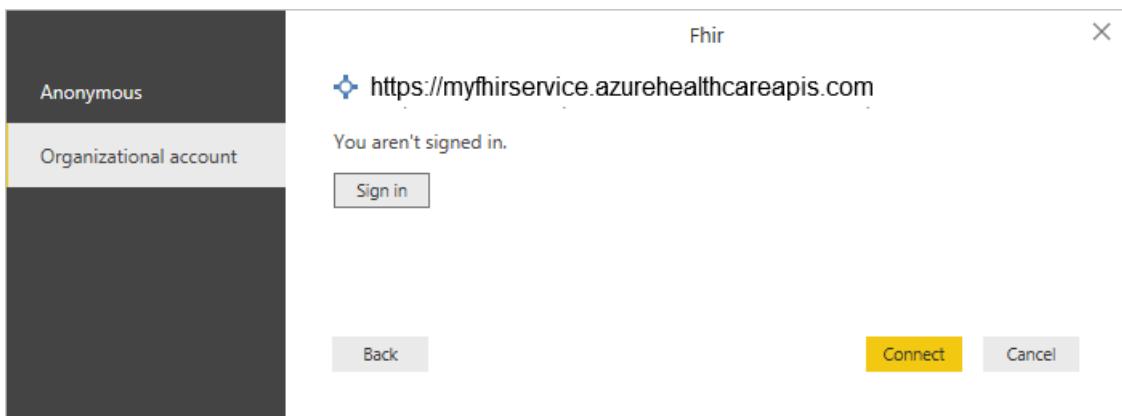
After that, follow the steps to [query and shape your data](#).

Azure Active Directory (organizational) authentication

The Power Query connector for FHIR supports OAuth authentication for FHIR servers that are secured with

Azure Active Directory.

To use Azure Active Directory authentication, select **Organizational account** when connecting.



There are some restrictions to be aware of:

- The expected Audience for the FHIR server **must** be equal to the base URL of the FHIR server. For the [Azure API for FHIR](#), you can set this when you [provision the FHIR service](#) or later in the portal.
- If your FHIR server doesn't return a `WWW-Authenticate` challenge header with an `authorization_uri` field on failed authorization, you must use an organizational account to sign in. You can't use a guest account in your active directory tenant. For the Azure API for FHIR, you **must** use an Azure Active Directory organizational account.
- If your FHIR service isn't the Azure API for FHIR (for example, if you're running the [open source Microsoft FHIR server for Azure](#)), you'll have registered an [Azure Active Directory resource application](#) for the FHIR server. You must pre-authorize the Power BI client application to be able to access this resource application.

A screenshot of the Azure portal. The URL in the address bar is 'https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade/ProtectAnAPI/'. The left sidebar shows 'Expose an API' selected under 'API permissions'. The main content area shows the 'Expose an API' blade for a registered application. It displays the 'Application ID URI' as 'https://[REDACTED].azurehealthcareapis.com'. Under 'Scopes defined by this API', there is one scope listed: 'https://[REDACTED].azurehealthcareapis.com/user_impersonation' with 'Admins and users' as the 'Who Can Consent' and 'Access https://[REDACTED].azu...' as the 'User Consent Display Name'. Under 'Authorized client applications', there is a table with one entry: 'Client Id' is 'a672d62c-fc7b-4e81-a576-e60dc46e951d' and 'Scopes' is '1'. A red box highlights the 'Add a client application' button and the Client Id field.

The client ID for the Power BI client is `a672d62c-fc7b-4e81-a576-e60dc46e951d`.

- The Power Query (for example, Power BI) client will only request a single scope: `user_impersonation`. This scope must be available and the FHIR server can't rely on other scopes.

Next steps

In this article, you've learned how to use the Power Query connector for FHIR authentication features. Next, explore query folding.

[FHIR Power Query folding](#)

FHIR query folding

1/15/2022 • 4 minutes to read • [Edit Online](#)

Power Query folding is the mechanism used by a Power Query connector to turn data transformations into queries that are sent to the data source. This allows Power Query to off-load as much of the data selection as possible to the data source rather than retrieving large amounts of unneeded data only to discard it in the client. The Power Query connector for FHIR includes query folding capabilities, but due to the nature of [FHIR search](#), special attention must be given to the Power Query expressions to ensure that query folding is performed when possible. This article explains the basics of FHIR Power Query folding and provides guidelines and examples.

FHIR and query folding

Suppose you are constructing a query to retrieve "Patient" resources from a FHIR server and you are interested in patients born before the year 1980. Such a query could look like:

```
let
    Source = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null),
    Patient1 = Source{[Name="Patient"]}[Data],
    "#Filtered Rows" = Table.SelectRows(Patient1, each [birthDate] < #date(1980, 1, 1))
in
    "#Filtered Rows"
```

Instead of retrieving all Patient resources from the FHIR server and filtering them in the client (Power BI), it's more efficient to send a query with a search parameter to the FHIR server:

```
GET https://myfhirserver.azurehealthcareapis.com/Patient?birthdate=lt1980-01-01
```

With such a query, the client would only receive the patients of interest and would not need to discard data in the client.

In the example of a birth date, the query folding is straightforward, but in general it is challenging in FHIR because the search parameter names don't always correspond to the data field names and frequently multiple data fields will contribute to a single search parameter.

For example, let's consider the `Observation` resource and the `category` field. The `Observation.category` field is a `CodeableConcept` in FHIR, which has a `coding` field, which have `system` and `code` fields (among other fields). Suppose you're interested in vital-signs only, you would be interested in Observations where `Observation.category.coding.code = "vital-signs"`, but the FHIR search would look something like `https://myfhirserver.azurehealthcareapis.com/Observation?category=vital-signs`.

To be able to achieve query folding in the more complicated cases, the Power Query connector for FHIR matches Power Query expressions with a list of expression patterns and translates them into appropriate search parameters. The expression patterns are generated from the FHIR specification.

This matching with expression patterns works best when any selection expressions (filtering) is done as early as possible in data transformation steps before any other shaping of the data.

NOTE

To give the Power Query engine the best chance of performing query folding, you should do all data selection expressions before any shaping of the data.

Query folding example

To illustrate efficient query folding, we'll walk through the example of getting all vital signs from the Observation resource. The intuitive way to do this would be to first expand the `Observation.category` field and then expand `Observation.category.coding` and then filter. The query would look something like this:

```
// Inefficient Power Query
let
    Source = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null),
    Observation = Source{[Name="Observation"]}[Data],
    ExpandCategory = Table.ExpandTableColumn(Observation, "category", {"coding"}, {"category.coding"}),
    ExpandCoding = Table.ExpandTableColumn(ExpandCategory, "category.coding", {"system", "code"}, {"category.coding.system", "category.coding.code"}),
    FilteredRows = Table.SelectRows(ExpandCoding, each ([category.coding.code] = "vital-signs"))
in
    FilteredRows
```

Unfortunately, the Power Query engine no longer recognized that as a selection pattern that maps to the `category` search parameter, but if you restructure the query to:

```
// Efficient Power Query allowing folding
let
    Source = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null),
    Observation = Source{[Name="Observation"]}[Data],
    FilteredObservations = Table.SelectRows(Observation, each Table.MatchesAnyRows([category], each Table.MatchesAnyRows([coding], each [code] = "vital-signs"))),
    ExpandCategory = Table.ExpandTableColumn(FilteredObservations, "category", {"coding"}, {"category.coding"}),
    ExpandCoding = Table.ExpandTableColumn(ExpandCategory, "category.coding", {"system", "code"}, {"category.coding.system", "category.coding.code"})
in
    ExpandCoding
```

The search query `/Observation?category=vital-signs` will be sent to the FHIR server, which will reduce the amount of data that the client will receive from the server.

While the first and the second Power Query expressions will result in the same data set, the latter will, in general, result in better query performance. It's important to note that the second, more efficient, version of the query can't be obtained purely through data shaping with the graphical user interface (GUI). It's necessary to write the query in the "Advanced Editor".

The initial data exploration can be done with the GUI query editor, but it's recommended that the query be refactored with query folding in mind. Specifically, selective queries (filtering) should be performed as early as possible.

Finding folding patterns

The Power Query connector for FHIR will only be able to perform query folding if the Power Query expressions map to known search parameters as defined by the FHIR specification. If you're wondering if query folding is possible, we recommend that you consult the [FHIR specification](#). Each resource will list a set of search parameters towards the bottom of the specification page. You can also consult the [folding query patterns](#) page

for examples of how to write foldable Power Query Expressions for FHIR.

Debugging query folding

If you're trying to determine if a given Power Query expression is being folded and what the resulting FHIR search expression is, you can start [Fiddler](#) while shaping queries in Power BI Desktop.

Summary

Query folding provides more efficient Power Query expressions. A properly crafted Power Query will enable query folding and thus off-load much of the data filtering burden to the data source.

Next steps

In this article, you've learned how to use query folding in the Power Query connector for FHIR. Next, explore the list of FHIR Power Query folding patterns.

[FHIR Power Query folding patterns](#)

FHIR query folding patterns

1/15/2022 • 9 minutes to read • [Edit Online](#)

This article describes Power Query patterns that will allow effective query folding in FHIR. It assumes that you are familiar with using the [Power Query connector for FHIR](#) and understand the basic motivation and principles for [Power Query folding in FHIR](#).

How to use this document

The list of examples in this document is not exhaustive and does not cover all the search parameters that queries will fold to. However, we provide examples of the types of queries and parameters you might encounter. When you are constructing a filter query expression, consider whether the parameter you would like to filter on is:

- A primitive type (like `Patient.birthDate`)
- A complex type, which would be a record in Power Query (like `Patient.meta`)
- An array of primitive types, which would be a list in Power Query (like `Patient.meta.profile`)
- An array of complex types, which would be a table in Power Query (like `observation.code.coding`, which has a number of columns)

And then consult the list of examples below. There are also examples of combining these types of filtering patterns in multi-level, nested filtering statements. Finally, this article provides more complicated filtering expressions that fold to [composite search parameters](#).

In each example you'll find a filtering expression (`Table.SelectRows`) and right above each filtering statement a comment `// Fold: ...` explaining what search parameters and values the expression will fold to.

Filtering on primitive types

Root properties are at the root of a resource and typically of a primitive type (string, date, and so on), but they can also be coding fields (for example `Encoding.class`). This section shows examples of searching different types of primitive root level properties.

Filtering patients by birth date:

```
let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
    // Fold: "birthdate=lt1980-01-01"
    FilteredPatients = Table.SelectRows(Patients, each [birthDate] < #date(1980, 1, 1))
in
    FilteredPatients
```

Filtering Patients by birth date range using `and`, only the 1970s:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
[Data],

    // Fold: "birthdate=ge1970-01-01&birthdate=lt1980-01-01"
    FilteredPatients = Table.SelectRows(Patients, each [birthDate] < #date(1980, 1, 1) and [birthDate] >=
#date(1970, 1, 1))
in
    FilteredPatients

```

Filtering Patients by birthdate using `or`, not the 1970s:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
[Data],

    // Fold: "birthdate=ge1980-01-01,lt1970-01-01"
    FilteredPatients = Table.SelectRows(Patients, each [birthDate] >= #date(1980, 1, 1) or [birthDate] <
#date(1970, 1, 1))
in
    FilteredPatients

```

Alternative search for active patients:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
[Data],

    // Fold: "active=true"
    FilteredPatients = Table.SelectRows(Patients, each [active])
in
    FilteredPatients

```

Alternative search for patients where active not true (could include missing):

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
[Data],

    // Fold: "active:not=true"
    FilteredPatients = Table.SelectRows(Patients, each [active] <> true)
in
    FilteredPatients

```

Filtering to keep only male patients:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
[Data],

    // Fold: "gender=male"
    FilteredPatients = Table.SelectRows(Patients, each [gender] = "male")
in
    FilteredPatients

```

Filtering to keep only patients that are not male (includes other):

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
        // Fold: "gender:not=male"
        FilteredPatients = Table.SelectRows(Patients, each [gender] <> "male")
in
    FilteredPatients

```

Filtering Observations with status final (code):

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation" ]}
    [Data],
        // Fold: "status=final"
        FilteredObservations = Table.SelectRows(Observations, each [status] = "final")
in
    FilteredObservations

```

Filtering on complex types

Filtering on last updated:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
        // Fold: "_lastUpdated=2010-12-31T11:56:02.000+00:00"
        FilteredPatients = Table.SelectRows(Patients, each [meta][lastUpdated] = #datetimezone(2010, 12, 31, 11,
56, 2, 0, 0))
in
    FilteredPatients

```

Filtering Encounter based on class system and code (coding):

```

let
    Encounters = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Encounter" ]}
    [Data],
        // Fold: "class=s|c"
        FilteredEncounters = Table.SelectRows(Encounters, each [class][system] = "s" and [class][code] = "c")
in
    FilteredEncounters

```

Filtering Encounter based on code (coding):

```

let
    Encounters = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Encounter" ]}
    [Data],
        // Fold: "class=c"
        FilteredEncounters = Table.SelectRows(Encounters, each [class][code] = "c")
in
    FilteredEncounters

```

Filtering Encounter based on class system only (coding):

```

let
    Encounters = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Encounter" ]}
    [Data],
    // Fold: "class=s"
    FilteredEncounters = Table.SelectRows(Encounters, each [class][system] = "s")
in
    FilteredEncounters

```

Filter Observations based on `Observation.subject.reference` (reference):

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation" ]}
    [Data],
    // Fold: "subject=Patient/1234"
    FilteredObservations = Table.SelectRows(Observations, each [subject][reference] = "Patient/1234")
in
    FilteredObservations

```

Filter Observations based on variations in `Observation.subject.reference` (reference):

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation" ]}
    [Data],
    // Fold: "subject=1234,Patient/1234,https://myfhirservice/Patient/1234"
    FilteredObservations = Table.SelectRows(Observations, each [subject][reference] = "1234" or [subject]
    [reference] = "Patient/1234" or [subject][reference] = "https://myfhirservice/Patient/1234")
in
    FilteredObservations

```

Filtering on Quantity equal value (quantity):

```

let
    ChargeItems = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "ChargeItem" ]}
    [Data],
    // Fold: "quantity=1"
    FilteredChargeItems = Table.SelectRows(ChargeItems, each [quantity][value] = 1)
in
    FilteredChargeItems

```

Filtering on Quantity greater than value (quantity):

```

let
    ChargeItems = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "ChargeItem" ]}
    [Data],
    // Fold: "quantity=gt1.001"
    FilteredChargeItems = Table.SelectRows(ChargeItems, each [quantity][value] > 1.001)
in
    FilteredChargeItems

```

Filtering on Quantity with value system and code (quantity):

```

let
    ChargeItems = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "ChargeItem"]
}][Data], 

    // Fold: "quantity=lt1.001|s|c"
    FilteredChargeItems = Table.SelectRows(ChargeItems, each [quantity][value] < 1.001 and [quantity]
[system] = "s" and [quantity][code] = "c")
in
    FilteredChargeItems

```

Filtering on period, starts after (period):

```

let
    Consents = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Consent" ]}
[Data], 

    // Fold: "period=sa2010-01-01T00:00:00.000+00:00"
    FiltertedConsents = Table.SelectRows(Consents, each [provision][period][start] > #datetimezone(2010, 1,
1, 0, 0, 0, 0))
in
    FiltertedConsents

```

Filtering on period, ends before (period):

```

let
    Consents = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Consent" ]}
[Data], 

    // Fold: "period=eb2010-01-01T00:00:00.000+00:00"
    FiltertedConsents = Table.SelectRows(Consents, each [provision][period][end] < #datetimezone(2010, 1, 1,
0, 0, 0, 0))
in
    FiltertedConsents

```

Filtering text field of complex types:

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"
}][Data], 

    // Fold: "code:text=t"
    FilteredObservations = Table.SelectRows(Observations, each [code][text] = "t")
in
    FilteredObservations

```

Filtering on text field (starts with):

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"
}][Data], 

    // Fold: "code:text=t"
    FilteredObservations = Table.SelectRows(Observations, each Text.StartsWith([code][text], "t"))
in
    FilteredObservations

```

Filtering on lists properties

Filtering Patients on profile:

```
let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
    // Fold: "_profile=http://myprofile"
    FilteredPatients = Table.SelectRows(Patients, each List.MatchesAny([meta][profile], each _ =
    "http://myprofile"))
in
    FilteredPatients
```

Filtering AllergyIntolerance on category:

```
let
    AllergyIntolerances = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name =
    "AllergyIntolerance" ]}[Data],
    // Fold: "category=food"
    FilteredAllergyIntolerances = Table.SelectRows(AllergyIntolerances, each List.MatchesAny([category],
    each _ = "food"))
in
    FilteredAllergyIntolerances
```

Filtering AllergyIntolerance on missing category:

```
let
    AllergyIntolerances = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name =
    "AllergyIntolerance" ]}[Data],
    // Fold: "category:missing=true"
    FilteredAllergyIntolerances = Table.SelectRows(AllergyIntolerances, each List.MatchesAll([category],
    each _ = null))
in
    FilteredAllergyIntolerances
```

Filtering AllergyIntolerance on simpler form of missing category:

```
let
    AllergyIntolerances = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name =
    "AllergyIntolerance" ]}[Data],
    // Fold: "category:missing=true"
    FilteredAllergyIntolerances = Table.SelectRows(AllergyIntolerances, each [category] = null)
in
    FilteredAllergyIntolerances
```

Filtering on table properties

Filtering Patients on exact family name:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}[Data], 

    // Fold: "family:exact=Johnson"
    FilteredPatients = Table.SelectRows(Patients, each Table.MatchesAnyRows([name], each [family] = 
"Johnson"))
in
    FilteredPatients

```

Filtering on Patients where family name starts with:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}[Data], 

    // Fold: "family=John"
    FilteredPatients = Table.SelectRows(Patients, each Table.MatchesAnyRows([name], each 
Text.StartsWith([family], "John")))
in
    FilteredPatients

```

Filtering Patients on family name starts with `John` or `Paul`:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}[Data], 

    // Fold: "family=John,Paul"
    FilteredPatients = Table.SelectRows(Patients, each Table.MatchesAnyRows([name], each 
Text.StartsWith([family], "John") or Text.StartsWith([family], "Paul")))
in
    FilteredPatients

```

Filtering Patients on family name starts with `John` and given starts with `Paul`:

```

let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}[Data], 

    // Fold: "family=John&given=Paul"
    FilteredPatients = Table.SelectRows(
        Patients,
        each
            Table.MatchesAnyRows([name], each Text.StartsWith([family], "John")) and
            Table.MatchesAnyRows([name], each List.MatchesAny([given], each Text.StartsWith(_, "Paul"))))
in
    FilteredPatients

```

Filtering on Goal due date:

```

let
    Goals = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Goal" ]}[Data], 

    // Fold: "target-date=gt2020-03-01"
    FilteredGoals = Table.SelectRows(Goals, each Table.MatchesAnyRows([target], each [due][date] > 
#date(2020,3,1)))
in
    FilteredGoals

```

Filtering Patient on identifier:

```
let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
        // Fold: "identifier=s|v"
        FilteredPatients = Table.SelectRows(Patients, each Table.MatchesAnyRows([identifier], each [system] =
    "s" and _[value] = "v"))
in
    FilteredPatients
```

Filtering on Observation code (CodeableConcept):

```
let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation" ]}
    [Data],
        // Fold: "code=s|c"
        FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([code][coding], each
    [system] = "s" and [code] = "c"))
in
    FilteredObservations
```

Filtering on Observation code and text (CodeableConcept):

```
let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation" ]}
    [Data],
        // Fold: "code:text=t&code=s|c"
        FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([code][coding], each
    [system] = "s" and [code] = "c") and [code][text] = "t")
in
    FilteredObservations
```

Filtering multi-level nested properties

Filtering Patients on family name starts with `John` and given starts with `Paul`:

```
let
    Patients = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Patient" ]}
    [Data],
        // Fold: "family=John&given=Paul"
        FilteredPatients =
            Table.SelectRows(
                Patients,
                each
                    Table.MatchesAnyRows([name], each Text.StartsWith([family], "John")) and
                    Table.MatchesAnyRows([name], each List.MatchesAny([given], each Text.StartsWith(_, "Paul"))))
in
    FilteredPatients
```

Filtering only vital signs from Observations:

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]
}][Data],


    // Fold: "category=vital-signs"
    FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([category], each
Table.MatchesAnyRows([coding], each [code] = "vital-signs")))
in
    FilteredObservations

```

Filtering Observations on category coding with system and code:

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]
}][Data],


    // Fold: "category=s|c"
    FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([category], each
Table.MatchesAnyRows([coding], each [system] = "s" and [code] = "c")))
in
    FilteredObservations

```

Filtering Observations on multiple categories (OR):

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]
}][Data],


    // Fold: "category=s1|c1,s2|c2"
    FilteredObservations =
        Table.SelectRows(
            Observations,
            each
                Table.MatchesAnyRows(
                    [category],
                    each
                        Table.MatchesAnyRows(
                            [coding],
                            each
                                ([system] = "s1" and [code] = "c1") or
                                ([system] = "s2" and [code] = "c2")))
in
    FilteredObservations

```

Filtering nested list in table:

```

let
    AuditEvents = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "AuditEvent"]
}][Data],


    // Fold: "policy=http://mypolicy"
    FilteredAuditEvents = Table.SelectRows(AuditEvents, each Table.MatchesAnyRows([agent], each
List.MatchesAny([policy], each _ = "http://mypolicy")))
in
    FilteredAuditEvents

```

Filtering with composite search parameters

FHIR has [composite search](#) parameters that allow filtering on multiple fields on a complex type within a resource or at the root of the resource at the same time. For example, one can search for Observations with

specific code *and* a specific value (a `code-value-quantity` search parameter). The Power Query connector for FHIR will attempt to recognize filtering expressions that map to such composite search parameters. This section lists some examples of these patterns. In the context of analyzing FHIR data, it is especially the composite search parameters on the `Observation` resource that are of interest.

Filtering Observations on code and value quantity, body height greater than 150:

```
let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]}[Data],
    
    // Fold: "code-value-quantity=http://loinc.org|8302-2$gt150"
    FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([code][coding], each
    [system] = "http://loinc.org" and [code] = "8302-2") and [value][Quantity][value] > 150)
in
    FilteredObservations
```

Filtering on Observation component code and value quantity, systolic blood pressure greater than 140:

```
let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]}[Data],
    
    // Fold: "component-code-value-quantity=http://loinc.org|8480-6$gt140"
    FilteredObservations = Table.SelectRows(Observations, each Table.MatchesAnyRows([component], each
    Table.MatchesAnyRows([code][coding], each [system] = "http://loinc.org" and [code] = "8480-6") and [value]
    [Quantity][value] > 140))
in
    FilteredObservations
```

Filtering on multiple component code value quantities (AND), diastolic blood pressure greater than 90 *and* systolic blood pressure greater than 140:

```
let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]}[Data],
    
    // Fold: "component-code-value-quantity=http://loinc.org|8462-4$gt90&component-code-value-
    quantity=http://loinc.org|8480-6$gt140"
    FilteredObservations =
        Table.SelectRows(
            Observations,
            each
                Table.MatchesAnyRows(
                    [component],
                    each
                        Table.MatchesAnyRows(
                            [code][coding], each [system] = "http://loinc.org" and [code] =
                            "8462-4") and [value][Quantity][value] > 90) and
                        Table.MatchesAnyRows([component], each Table.MatchesAnyRows([code][coding], each
                            [system] = "http://loinc.org" and [code] = "8480-6") and [value][Quantity][value] > 140))
in
    FilteredObservations
```

Filtering on multiple component code value quantities (OR), diastolic blood pressure greater than 90 *or* systolic blood pressure greater than 140:

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]}[Data], 

    // Fold: "component-code-value-quantity=http://loinc.org|8462-4$gt90,http://loinc.org|8480-6$gt140"
    FilteredObservations =
        Table.SelectRows(
            Observations,
            each
                Table.MatchesAnyRows(
                    [component],
                    each
                        (Table.MatchesAnyRows([code][coding], each [system] = "http://loinc.org" and [code] = "8462-4") and [value][Quantity][value] > 90) or
                            Table.MatchesAnyRows([code][coding], each [system] = "http://loinc.org" and [code] = "8480-6") and [value][Quantity][value] > 140 )
            in
                FilteredObservations

```

Filtering Observations on code value quantities on root of resource or in component array:

```

let
    Observations = Fhir.Contents("https://myfhirserver.azurehealthcareapis.com", null){[Name = "Observation"]}[Data], 

    // Fold: "combo-code-value-quantity=http://loinc.org|8302-2$gt150"
    FilteredObservations =
        Table.SelectRows(
            Observations,
            each
                (Table.MatchesAnyRows([code][coding], each [system] = "http://loinc.org" and [code] = "8302-2") and [value][Quantity][value] > 150) or
                    (Table.MatchesAnyRows([component], each Table.MatchesAnyRows([code][coding], each [system] = "http://loinc.org" and [code] = "8302-2") and [value][Quantity][value] > 150))
            in
                FilteredObservations

```

Summary

Query folding turns Power Query filtering expressions into FHIR search parameters. The Power Query connector for FHIR recognizes certain patterns and attempts to identify matching search parameters. Recognizing those patterns will help you write more efficient Power Query expressions.

Next steps

In this article, we reviewed some classes of filtering expressions that will fold to FHIR search parameters. Next read about establishing relationships between FHIR resources.

[FHIR Power Query relationships](#)

FHIR Relationships

1/15/2022 • 2 minutes to read • [Edit Online](#)

This article describes how to establish [relationships](#) between tables that have been imported using the Power Query connector for FHIR.

Introduction

FHIR resources are related to each other, for example, an `Observation` that references a subject (`Patient`):

```
{  
    "resourceType": "Observation",  
    "id": "1234",  
    "subject": {  
        "reference": "Patient/456"  
    }  
  
    // ... other fields  
}
```

Some of the resource reference fields in FHIR can refer to multiple different types of resources (for example, `Practitioner` or `Organization`). To facilitate an easier way to resolve references, the Power Query connector for FHIR adds a synthetic field to all imported resources called `<referenceId>`, which contains a concatenation of the resource type and the resource ID.

To establish a relationship between two tables, you can connect a specific reference field on a resource to the corresponding `<referenceId>` field on the resource you would like it linked to. In simple cases, Power BI will even detect this for you automatically.

Establishing FHIR relationships in Power BI

In this section, we'll show an example of establishing a relationship between the `Observation.subject.reference` field and the `<referenceId>` field on `Patient`.

1. When importing data, select the `Patient` and `Observation` tables:

Navigator

Display Options ▾

- MolecularSequence
- NamingSystem
- NutritionOrder
- Observation
- ObservationDefinition
- OperationDefinition
- OperationOutcome
- Organization
- OrganizationAffiliation
- Patient
- PaymentNotice
- PaymentReconciliation
- Person
- PlanDefinition
- Practitioner
- PractitionerRole
- Procedure
- Provenance
- Questionnaire
- QuestionnaireResponse

Observation

id	meta	implicitRules	language	<re
ee652fed-2ac9-471b-a83b-206f988801e7	Record		null	null Ob:
5104be56-5e8b-4466-855a-38aed002d87e	Record		null	null Ob:
8e9e8f2b-28ec-42c3-976f-4a9c2472f896	Record		null	null Ob:
4a089b8a-b0a0-46a9-92da-c8b653aa2e71	Record		null	null Ob:
08568cc1-c116-41d8-9ad2-fd1586cd286f	Record		null	null Ob:
516396ca-fec8-4a8e-aa75-bb9c8359b6a8	Record		null	null Ob:
06190a3f-0a5d-4aae-83fb-ae1403c4fe3b	Record		null	null Ob:
c4bf28c3-cba-4dc4-83a5-739a537cf95a	Record		null	null Ob:
d9c25b12-4756-460b-9786-987dc455fd51	Record		null	null Ob:
c55a7a46-f7db-4e1f-bd81-cd8c56594b03	Record		null	null Ob:
37fb0aa8-f59c-4e39-89b8-cebce2f65dac	Record		null	null Ob:
65832e35-cb93-4589-97ed-d46c1750edc3	Record		null	null Ob:
2eb4862e-a7ad-406f-9196-8204761f7b7b	Record		null	null Ob:
4d0139f1-faf8-4762-bb16-bbad068f3f5f	Record		null	null Ob:
f3fab696-7e93-43d6-9fed-991d65262a75	Record		null	null Ob:

The data in the preview has been truncated due to size limits.

Load Transform Data Cancel

Then select **Transform Data**.

2. Expand the **subject** column on **observation** to reveal **subject.reference**:

Untitled - Power Query Editor

File Home Transform Add Column View Help

Queries [2]

Patient

Observation

Source[[Name="Observation"]][Data]

subject

Search Columns to Expand

(Select All Columns)

reference

type

identifier

display

Use original column name as prefix

OK Cancel

Properties

Name: Observation

All Properties

Applied Steps

Source Navigation

33 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 9:09 AM

After expanding, you should see the list of subject references:

The screenshot shows the Power Query Editor interface. The main area displays a table with columns: code, subject.reference, focus, and encounter. The 'subject.reference' column contains many rows starting with 'Patient/e11037dd-9b76-42bb-92cb-9109a7b4ab...'. The 'APPLIED STEPS' pane on the right shows the step 'Expanded subject'.

3. Make any other modifications you need to the query and save the modified query.

4. Select **Manage Relationships** in the Power BI client:

The screenshot shows the Power BI Desktop interface. The ribbon has the 'Modeling' tab selected. The 'Relationships' button in the ribbon is highlighted with a red box. The main workspace is empty, showing a large gray area with a few small icons at the bottom.

5. Establish the relationship. In this simple example, Power BI will likely have detected the relationship automatically:

Manage relationships

Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	Observation (subject.reference)	Patient (<referenceld>)

New...

Autodetect...

Edit...

Delete

Close

If not, you can add it manually:

Manage relationships

Active	From: Table (Column)	To: Table (Column)
There are no relationships defined yet.		

New...

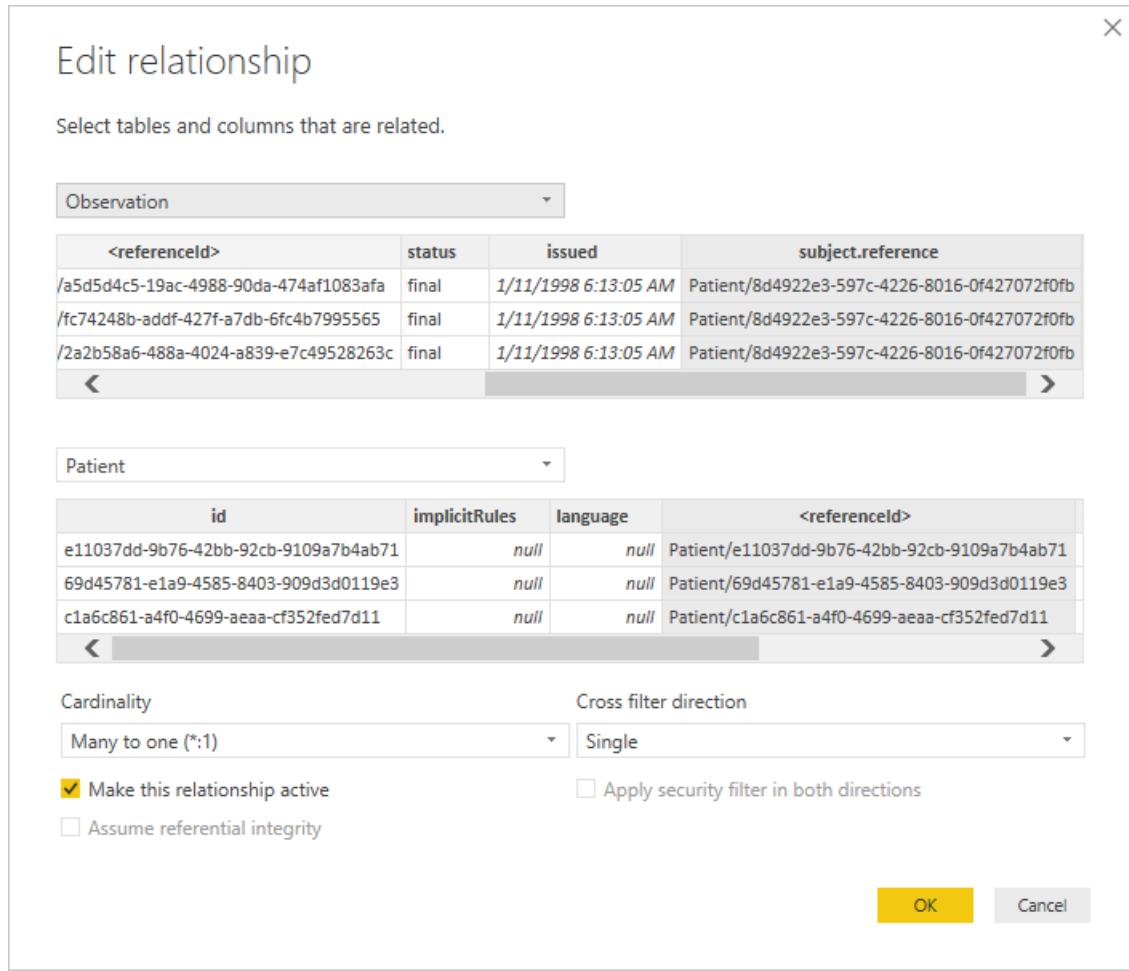
Autodetect...

Edit...

Delete

Close

You can edit the details of the relationship:



Summary

Resources in FHIR are related. These relationships need to be established on data imported with the Power Query connector for FHIR. The `<referenceId>` field is a synthetic field added to all imported FHIR data that will help establish the relationships.

Next steps

In this article, you've learned how to establish relationships between tables imported with the Power Query connector for FHIR. Next, explore query folding with the Power Query connector for FHIR.

FHIR Power Query folding

Folder

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Windows
Function Reference Documentation	Folder.Contents , Folder.Files

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

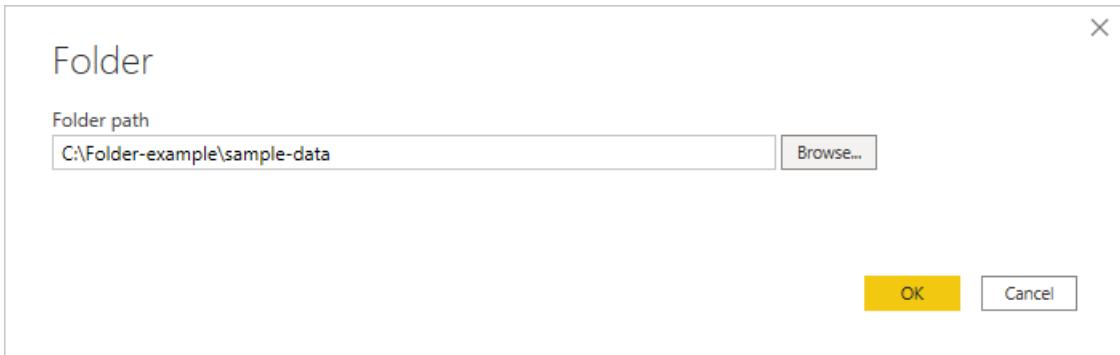
Capabilities supported

- Folder path
- Combine
 - Combine and load
 - Combine and transform

Connect to a folder from Power Query Desktop

To connect to a folder from Power Query Desktop:

1. Select the **Folder** option in the connector selection.
2. Enter the path to the folder you want to load, or select **Browse** to browse to the folder you want to load.
Then select **OK**.



When you select the folder you want to use, the file information about all of the files in that folder are displayed. Also, file information about any files in any subfolders is also displayed.

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	Financial Sample 1.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:43:47 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 2.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:45:00 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 3.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:46:10 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 4.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:47:23 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 5.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:28 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\new-samples\
Binary	Financial Sample 6.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:49 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\new-samples\

3. Select **Combine & Transform Data** to combine the data in the files of the selected folder and load the data in the Power Query Editor for editing. Select **Combine & Load** to load the data from all of the files in the folder directly into your app. Or select **Transform Data** to load the folder data as-is in the Power Query Editor.

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder P
Binary	Financial Sample 1.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:43:47 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 2.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:45:00 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 3.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:46:10 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 4.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:47:23 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\
Binary	Financial Sample 5.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:28 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\new-samples\
Binary	Financial Sample 6.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:49 AM	4/29/2020 9:15:41 AM	Record	C:\Folder-example\sample-data\new-samples\

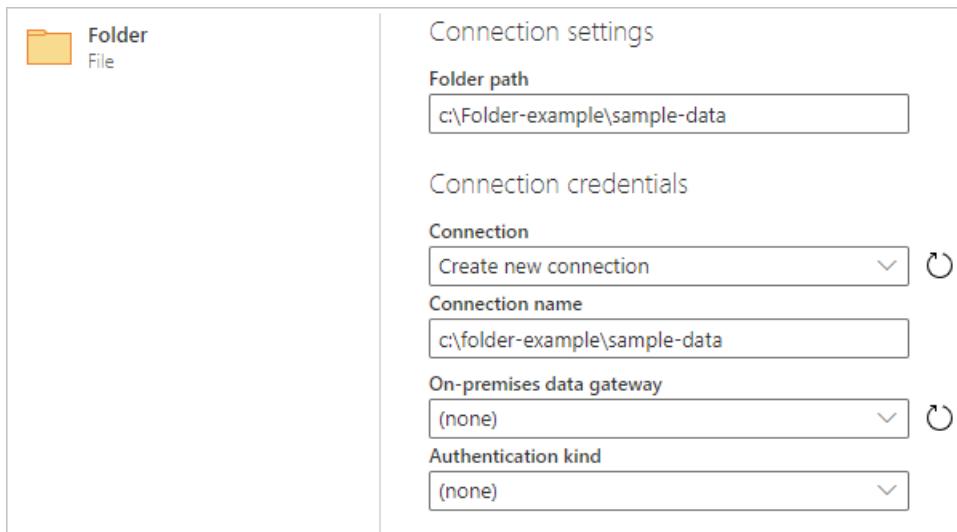
NOTE

The **Combine & Transform Data** and **Combine & Load** buttons are the easiest ways to combine data found in the files of the folder you specify. You could also use the **Load** button (in Power BI Desktop only) or the **Transform Data** buttons to combine the files as well, but that requires more manual steps.

Connect to a folder from Power Query Online

To connect to a folder from Power Query Online:

1. Select the **Folder** option in the connector selection.
2. Enter the path to the folder you want to load.



3. Enter the name of an on-premises data gateway that you'll use to access the folder.
4. Select the authentication kind to connect to the folder. If you select the **Windows** authentication kind, enter your credentials.
5. Select **Next**.
6. In the **Navigator** dialog box, select **Combine** to combine the data in the files of the selected folder and load the data into the Power Query Editor for editing. Or select **Transform data** to load the folder data as-is in the Power Query Editor.

The screenshot shows the 'Power Query - Preview folder data' dialog box. It displays a preview of files from 'c:\Folder-example\sample-data'. The table has columns: Content, Name, Extension, Date accessed, Date modified, Date created, Attributes, and Folder Path. The preview shows six CSV files named 'Financial Sample 1....', 'Financial Sample 2....', 'Financial Sample 3....', 'Financial Sample 4....', 'Financial Sample 5....', and 'Financial Sample 6....'. At the bottom of the dialog are buttons for 'Back', 'Cancel', 'Combine', and 'Transform data'.

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
[Binary]	Financial Sample 1....	.CSV	12/8/2020, 6:44:30 AM	4/28/2020, 8:43:47 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\
[Binary]	Financial Sample 2....	.CSV	9/25/2020, 11:19:08 ...	4/28/2020, 8:44:59 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\
[Binary]	Financial Sample 3....	.CSV	10/23/2020, 9:59:19 ...	4/28/2020, 8:46:09 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\
[Binary]	Financial Sample 4....	.CSV	10/23/2020, 9:59:19 ...	4/28/2020, 8:47:22 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\
[Binary]	Financial Sample 5....	.CSV	4/29/2020, 9:15:41 AM	4/28/2020, 8:48:27 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\new-sampl...
[Binary]	Financial Sample 6....	.CSV	4/29/2020, 9:15:41 AM	4/28/2020, 8:48:49 AM	4/29/2020, 9:15:41 A...	[Record]	c:\Folder-example\sample-data\new-sampl...

Troubleshooting

Combining files

When you combine files using the folder connector, all the files in the folder and its subfolders are processed the same way, and the results are then combined. The way the files are processed is determined by the example file you select. For example, if you select an Excel file and choose a table called "Table1", then all the files will be treated as Excel files that contain a table called "Table1".

To ensure that combining the files works properly, make sure that all the files in the folder and its subfolders have the same file format and structure. If you need to exclude some of the files, first select **Transform data** instead of **Combine** and filter the table of files in the Power Query Editor before **combining**.

For more information about combining files, go to [Combine files in Power Query](#).

Google Analytics

1/15/2022 • 5 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power Apps (Dataflows)
Authentication Types Supported	Google Account
Function Reference Documentation	GoogleAnalytics.Accounts

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

NOTE

Effective July 2021, Google will discontinue support for sign-ins to Google accounts from embedded browser frameworks. Due to this change, you will need to [update](#) your Power BI Desktop version to June 2021 to support signing in to Google.

NOTE

This connector uses V4 of the Google Analytics API.

Prerequisites

Before you can sign in to Google Analytics, you must have a Google Analytics account (username/password).

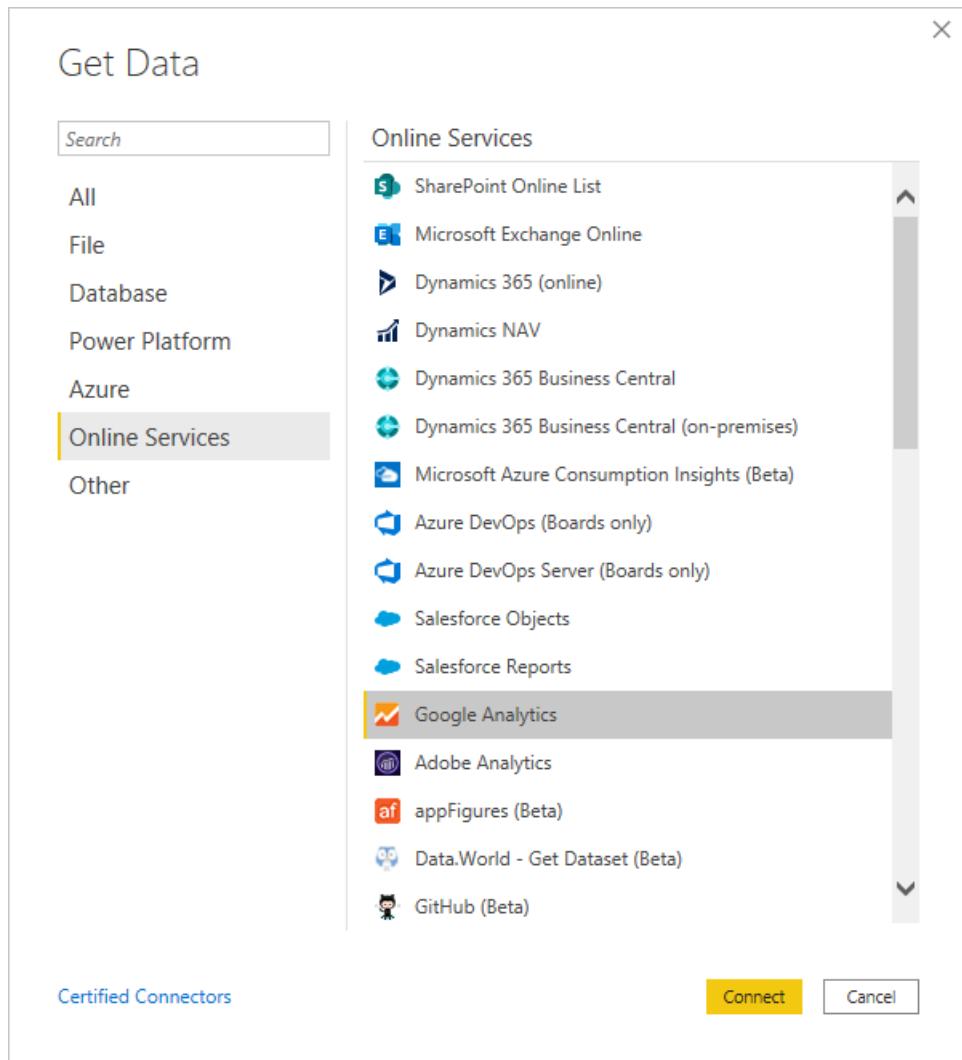
Capabilities Supported

- Import

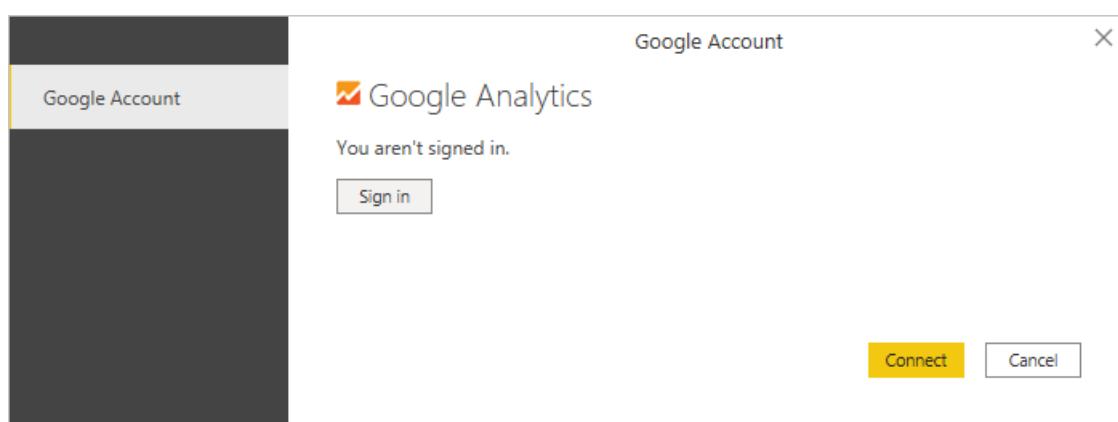
Connect to Google Analytics data from Power Query Desktop

To connect to Google Analytics data:

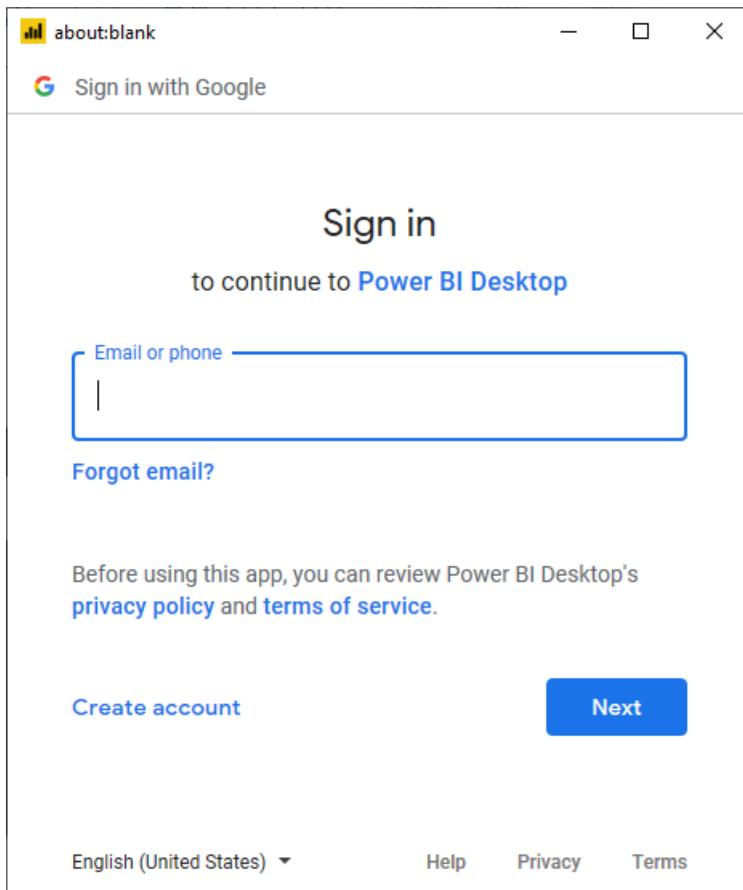
1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **Online Services** from the categories on the left, and then select **Google Analytics**. Then select **Connect**.



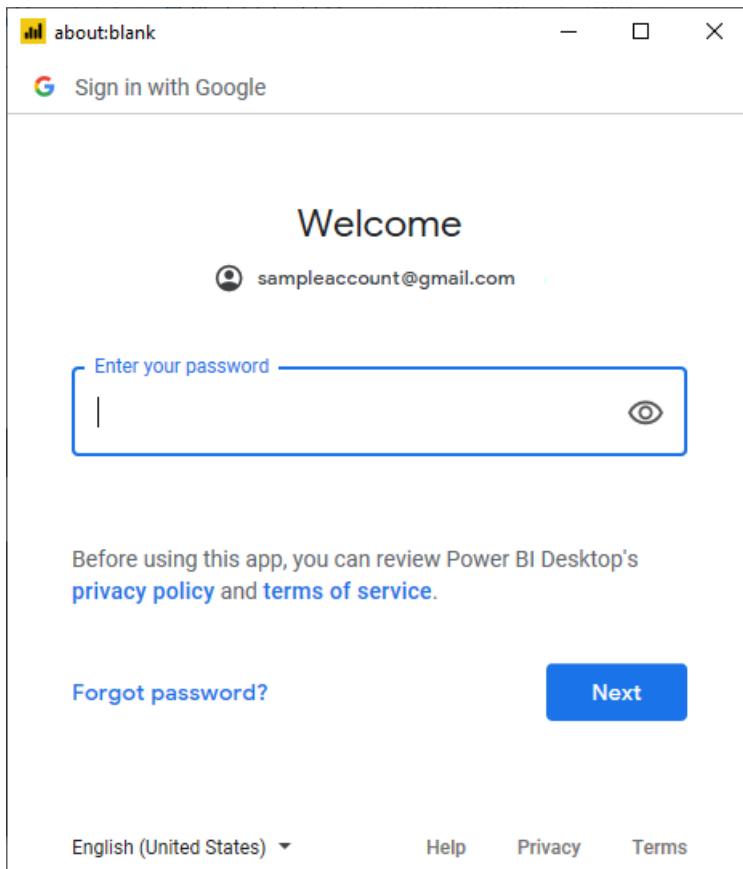
2. If this is the first time you're getting data through the Google Analytics connector, a third-party notice is displayed. Select **Don't warn me again with this connector** if you don't want this message to be displayed again. Then select **Continue**.
3. To sign in to your Google Analytics account, select **Sign in**.



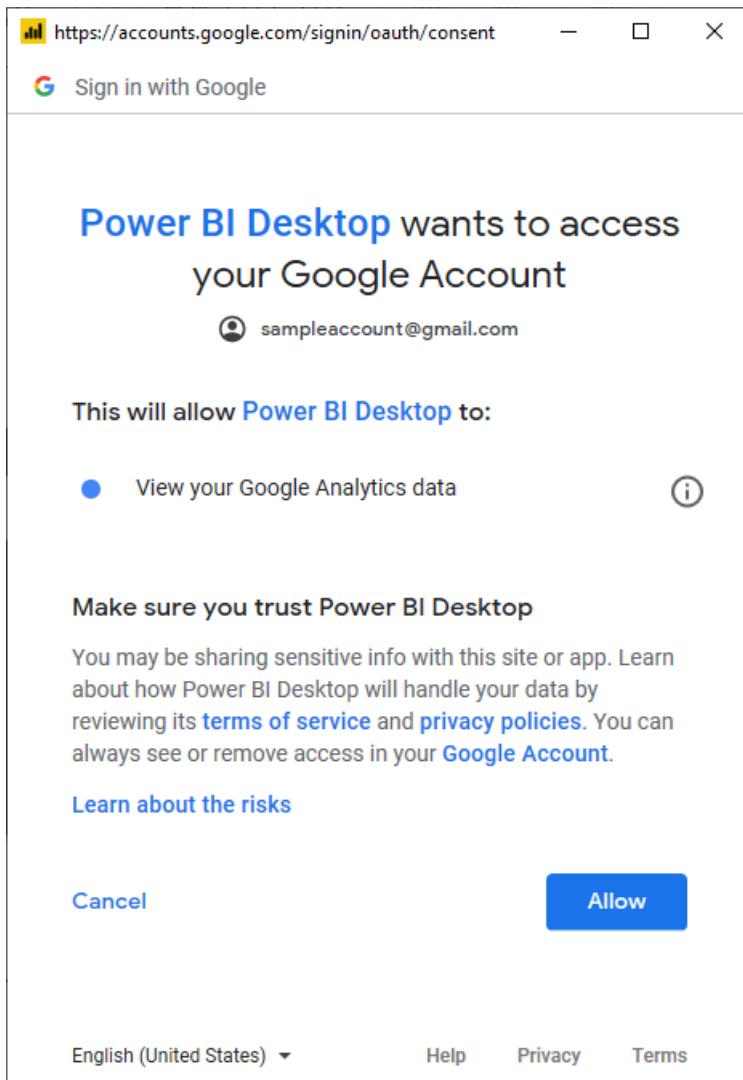
4. In the **Sign in with Google** window that appears, provide your credentials to sign in to your Google Analytics account. You can either supply an email address or phone number. Then select **Next**.



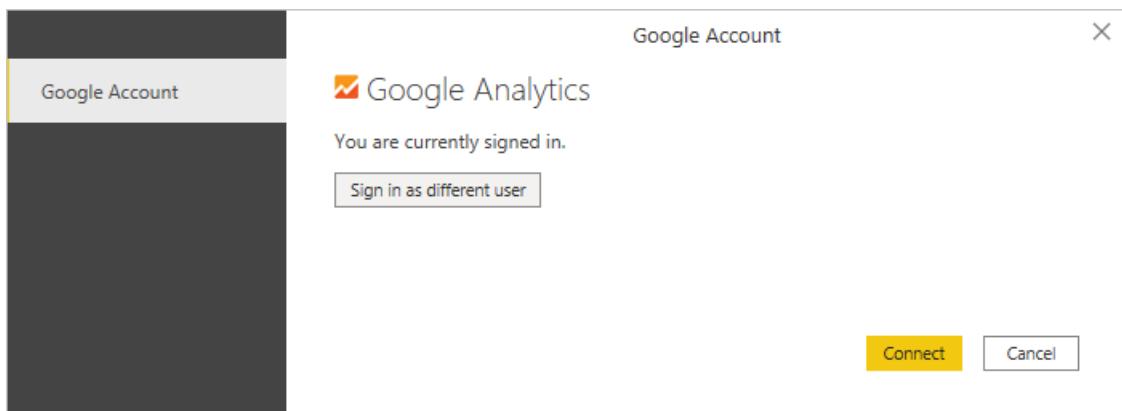
5. Enter your Google Analytics password and select **Next**.



6. When asked if you want Power BI Desktop to access your Google account, select **Allow**.

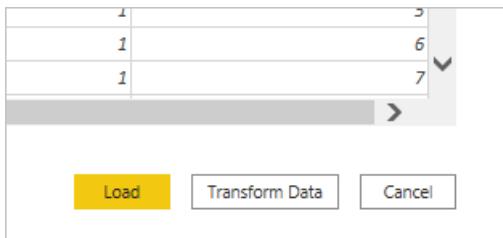


7. Once you've successfully signed in, select **Connect**.



Once the connection is established, you'll see a list of the accounts you have access to. Drill through the account, properties, and views to see a selection of values, categorized in display folders.

You can **Load** the selected table, which brings the entire table into Power BI Desktop, or you can select **Transform Data** to edit the query, which opens Power Query Editor. You can then filter and refine the set of data you want to use, and then load that refined set of data into Power BI Desktop.



Connect to Google Analytics data from Power Query Online

To connect to Google Analytics data:

1. Select **Google Analytics** from the **Power Query - Choose data source** page.
2. From the connection page, enter a connection name and choose an on-premises data gateway if necessary.

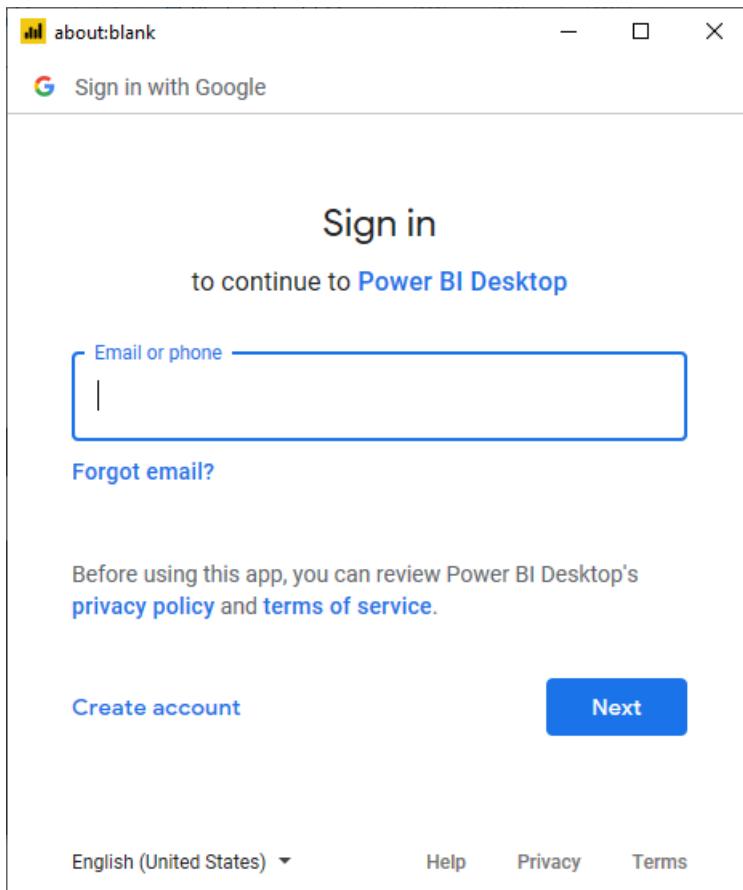
The screenshot shows the 'Connection credentials' page for Google Analytics. On the left, there's a sidebar with the Google Analytics logo and 'Online services'. Below it, a '3rd Party' button is highlighted. The main area has the following fields:

- Connection:** Create new connection (dropdown with a refresh icon)
- Connection name:** GoogleAnalytics
- Data gateway:** (none) (dropdown with a refresh icon)
- Authentication kind:** Organizational account (dropdown with a refresh icon)
- A message: You are not signed in. Please sign in. with a **Sign in** button.
- Privacy Level:** None (dropdown with a refresh icon)

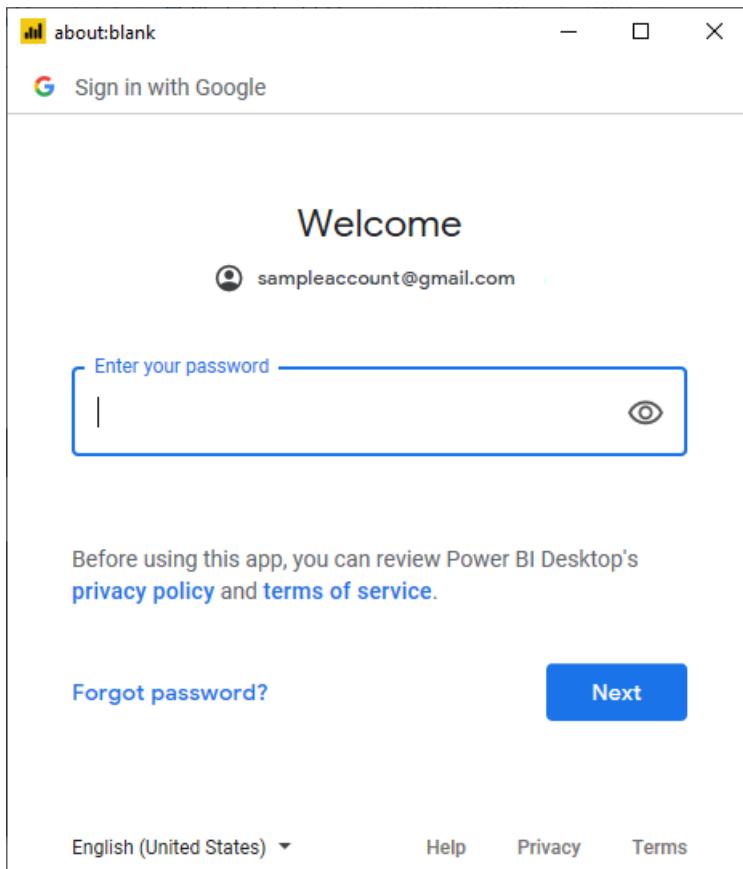
3. Select **Sign in** to sign in to your Google account.
4. In the **Sign in with Google** window that appears, provide your credentials to sign in to your Google Analytics account. You can either supply an email address or phone number. Then select **Next**.

NOTE

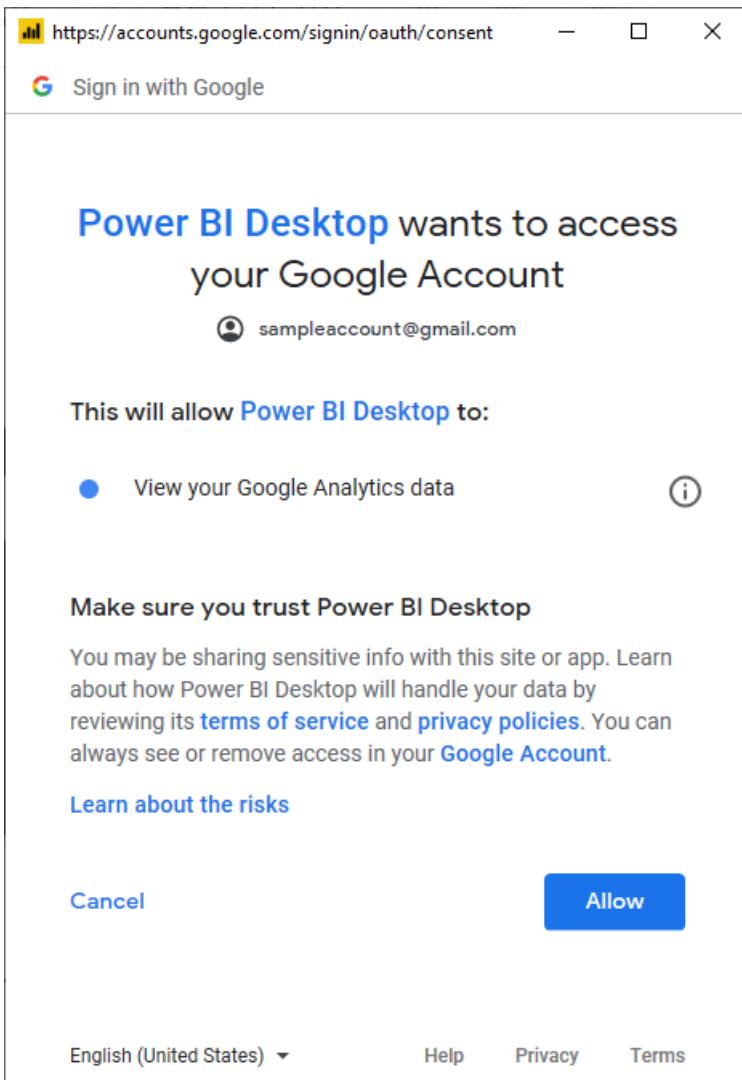
Currently, the Google Analytics sign-in dialog boxes indicate that you are signing in to Power Query Desktop. This wording will be changed in the future.



5. Enter your Google Analytics password and select **Next**.



6. When asked if you want Power BI Desktop to access your Google account, select **Allow**.



7. Once you've successfully signed in, select **Next**.

Once the connection is established, you'll see a list of the accounts you have access to. Drill through the account, properties, and views to see a selection of values, categorized in display folders.

8. Select **Transform data** to edit the query in Power Query Editor. You can then filter and refine the set of data you want to use, and then load that refined set of data into Power Apps.

Limitations and issues

You should be aware of the following limitations and issues associated with accessing Adobe Analytics data.

Google Analytics quota limits for Power BI

The standard limitations and quotas for Google Analytics AP requests is documented in [Limits and Quotas on API Requests](#). However, Power BI Desktop and Power BI service allow you to use the following enhanced number of queries.

- Queries per day: 1,500,000
- Queries per 100 seconds: 4,000

Troubleshooting

Validating Unexpected Data

When date ranges are very large, Google Analytics will return only a subset of values. You can use the process described in this section to understand what dates are being retrieved, and manually edit them. If you need more data, you can append multiple queries with different date ranges. If you're not sure you're getting back the

data you expect to see, you can also use [Data Profiling](#) to get a quick look at what's being returned.

To make sure that the data you're seeing is the same as you would get from Google Analytics, you can execute the query yourself in Google's interactive tool. To understand what data Power Query is retrieving, you can use [Query Diagnostics](#) to understand what query parameters are being sent to Google Analytics.

If you follow the instructions for Query Diagnostics and run **Diagnose Step** on any **Added Items**, you can see the generated results in the **Diagnostics Data Source Query** column. We recommend running this with as few additional operations as possible on top of your initial connection to Google Analytics, to make sure you're not losing data in a Power Query transform rather than what's being retrieved from Google Analytics.

Depending on your query, the row containing the emitted API call to Google Analytics may not be in the same place. But for a simple Google Analytics only query, you'll generally see it as the last row that has content in that column.

In the **Data Source Query** column, you'll find a record with the following pattern:

```
Request:  
GET https://www.googleapis.com/analytics/v3/data/ga?ids=ga:<GA  
Id>&metrics=ga:users&dimensions=ga:source&start-date=2009-03-12&end-date=2020-08-11&start-index=1&max-  
results=1000&quotaUser=<User>%40gmail.com HTTP/1.1  
  
<Content placeholder>  
  
Response:  
HTTP/1.1 200 OK  
Content-Length: -1  
  
<Content placeholder>
```

From this record, you can see you have your [Analytics view \(profile\) ID](#), your list of [metrics](#) (in this case, just `ga:users`), your list of [dimensions](#) (in this case, just referral source), the [start-date](#) and [end-date](#), the [start-index](#), [max-results](#) (set to 1000 for the editor by default), and the [quotaUser](#).

You can copy these values into the [Google Analytics Query Explorer](#) to validate that the same data you're seeing returned by your query is also being returned by the API.

If your error is around a date range, you can easily fix it. Go into the [Advanced Editor](#). You'll have an M query that looks something like this (at a minimum—there may be other transforms on top of it).

```
let  
    Source = GoogleAnalytics.Accounts(),  
    #"<ID>" = Source{[Id="<ID>"]}[Data],  
    #"UA-<ID>-1" = #"<ID>"{[Id="UA-<ID>-1"]}[Data],  
    #"<View ID>" = #"UA-<ID>-1"{[Id="<View ID>"]}[Data],  
    #"Added Items" = Cube.Transform(#"<View ID>",  
        {  
            {Cube.AddAndExpandDimensionColumn, "ga:source", {"ga:source"}, {"Source"}},  
            {Cube.AddMeasureColumn, "Users", "ga:users"}  
        })  
in  
    #"Added Items"
```

You can do one of two things. If you have a **Date** column, you can filter on the Date. This is the easier option. If you don't care about breaking it up by date, you can Group afterwards.

If you don't have a **Date** column, you can manually manipulate the query in the Advanced Editor to add one and filter on it. For example:

```

let
    Source = GoogleAnalytics.Accounts(),
    #"<ID>" = Source{[Id=<ID>]}[Data],
    #>UA-<ID>-1" = #"<ID>{[Id="UA-<ID>-1"]}[Data],
    #><View ID>" = #>UA-<ID>-1"{[Id=<View ID>"]}[Data],
    #>Added Items" = Cube.Transform(#><View ID>",
    {
        {Cube.AddAndExpandDimensionColumn, "ga:date", {"ga:date"}, {"Date"}},
        {Cube.AddAndExpandDimensionColumn, "ga:source", {"ga:source"}, {"Source"}},
        {Cube.AddMeasureColumn, "Organic Searches", "ga:organicSearches"}
    }),
    #>Filtered Rows" = Table.SelectRows(#>Added Items", each [Date] >= #date(2019, 9, 1) and [Date] <=
    #date(2019, 9, 30))
in
#>Filtered Rows"

```

Next steps

- [Google Analytics Dimensions & Metrics Explorer](#)
- [Google Analytics Core Reporting API](#)

Google BigQuery

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Customer Insights (Dataflows)
Authentication Types Supported	Organizational account Service account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

NOTE

Effective July 2021, Google will discontinue support for sign-ins to Google accounts from embedded browser frameworks. Due to this change, you will need to [update](#) your Power BI Desktop version to June 2021 to support signing in to Google.

Prerequisites

You'll need a Google account or a Google service account to sign in to Google BigQuery.

Capabilities supported

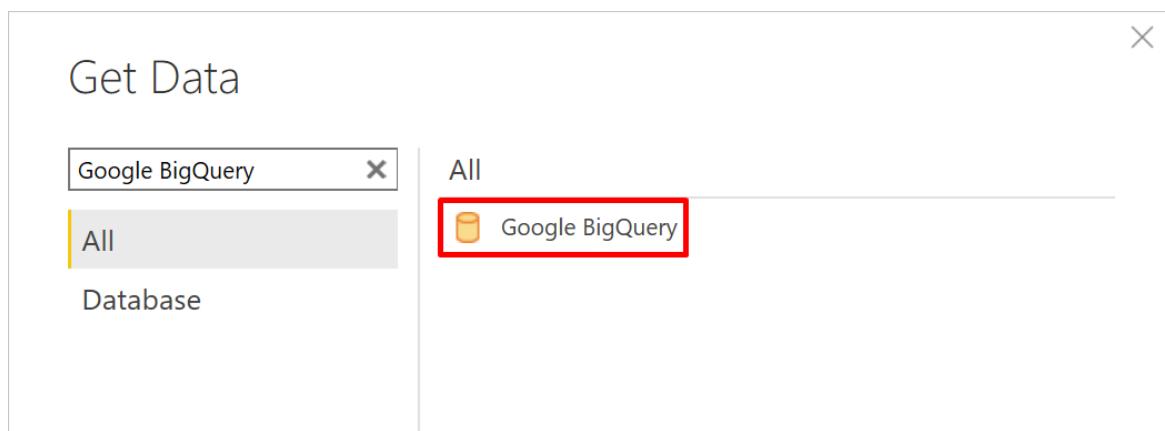
- Import
- DirectQuery (Power BI Desktop only)

Connect to Google BigQuery data from Power Query Desktop

To connect to Google BigQuery from Power Query Desktop, take the following steps:

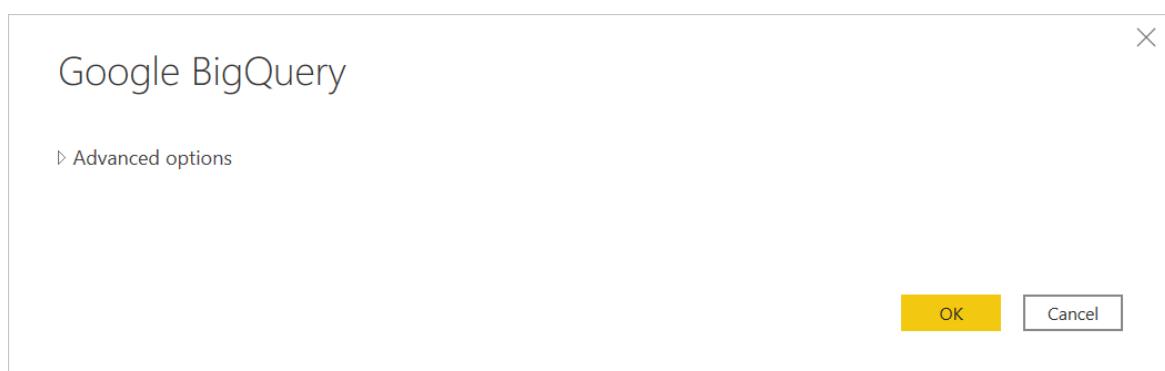
1. In the Get Data experience, search for and select **Google BigQuery**.

Get Data

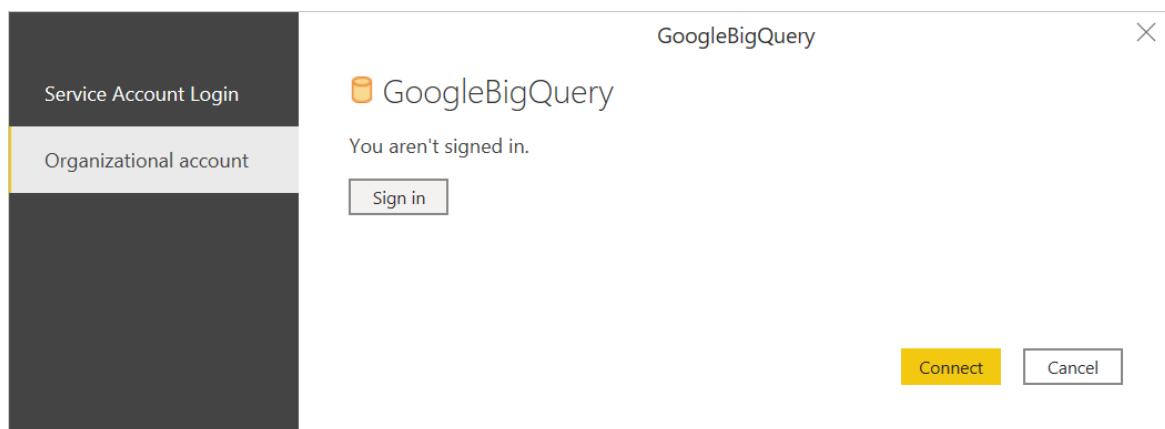


2. If you want to use any advance options, select **Advanced options**. Otherwise, select OK to continue.

More information: [Connect using advanced options](#)



3. The Google BigQuery connector supports connecting through an organizational account or a service account sign-in. In this example, you'll use the organizational account to sign in. Select **Sign In** to continue.



You can also sign in using a Google service account. In this case, select **Service Account Login** and enter your service account email and your service account JSON key file contents. Then select **Connect**.

4. A **Sign in with Google** dialog appears. Select your Google account and approve connecting to Power BI Desktop.

 Sign in with Google

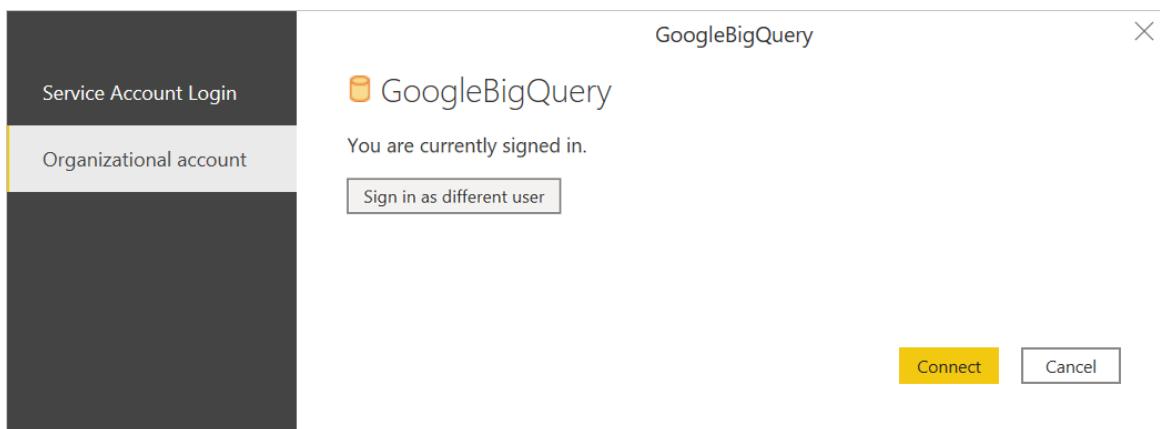
Choose an account

to continue to [Power BI Desktop](#)

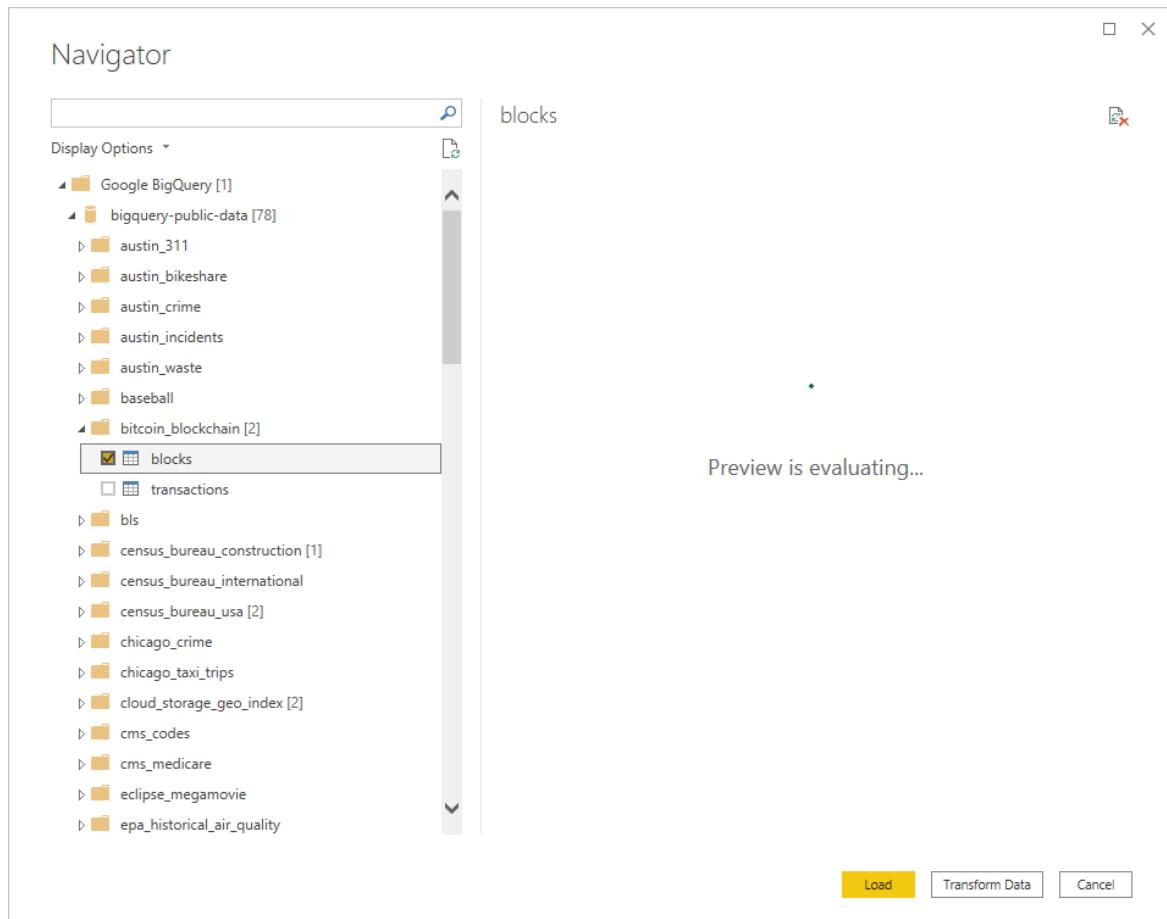
 [Use another account](#)

Before using this app, you can review Power BI Desktop's [privacy policy](#) and Terms of Service.

5. Once signed in, select **Connect** to continue.



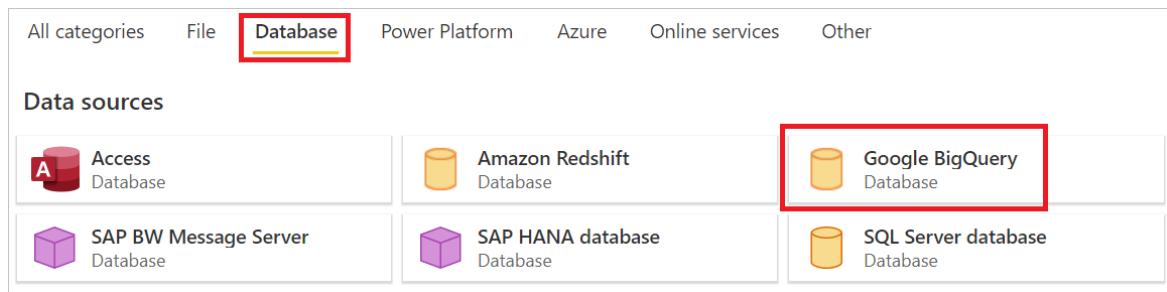
6. Once you successfully connect, a **Navigator** window appears and displays the data available on the server. Select your data in the navigator. Then select either **Transform Data** to transform the data in Power Query or **Load** to load the data in Power BI Desktop.



Connect to Google BigQuery data from Power Query Online

To connect to Google BigQuery from Power Query Online, take the following steps:

1. In the Get Data experience, select the **Database** category, and then select **Google BigQuery**.



2. In the **Google BigQuery Database** dialog, you may need to either create a new connection or select an existing connection. If you're using on-premises data, select an on-premises data gateway. Then select **Sign in**.



Google BigQuery

Database

Connection settings

> Advanced options

Connection credentials

Data gateway

(none)



Authentication kind

Basic



Username

Password

3. A **Sign in with Google** dialog appears. Select your Google account and approve connecting.

NOTE

Although the sign in dialog box says you'll continue to Power BI Desktop once you've signed in, you'll be sent to your online app instead.



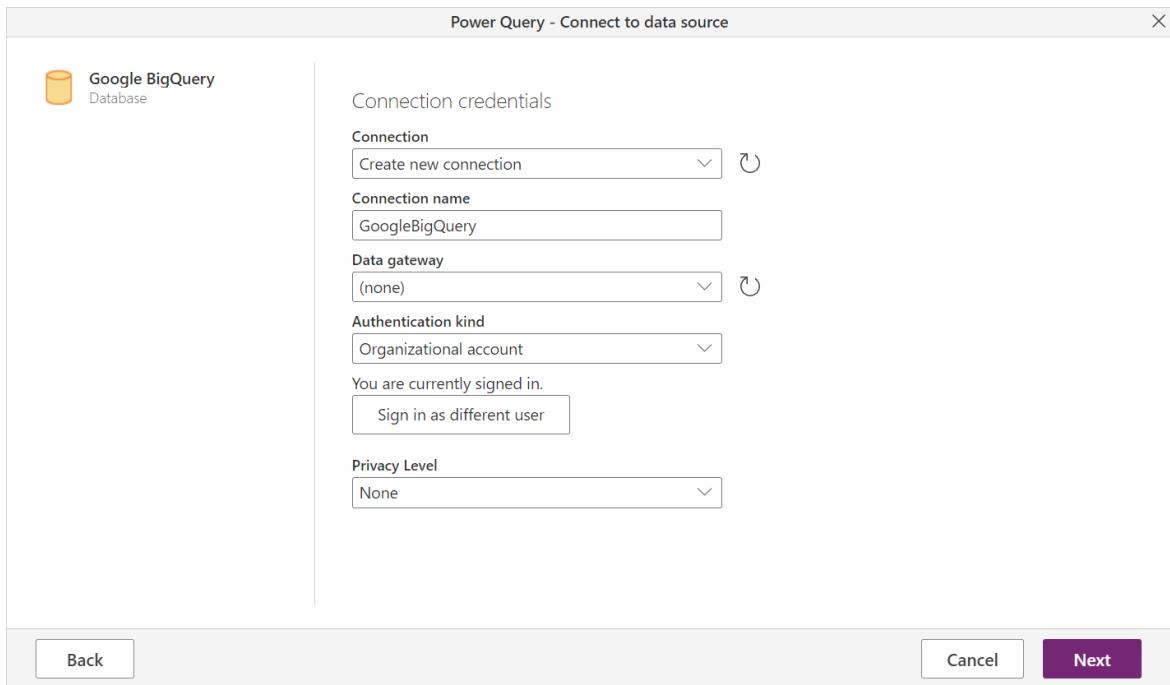
Choose an account

to continue to [Power BI Desktop](#)

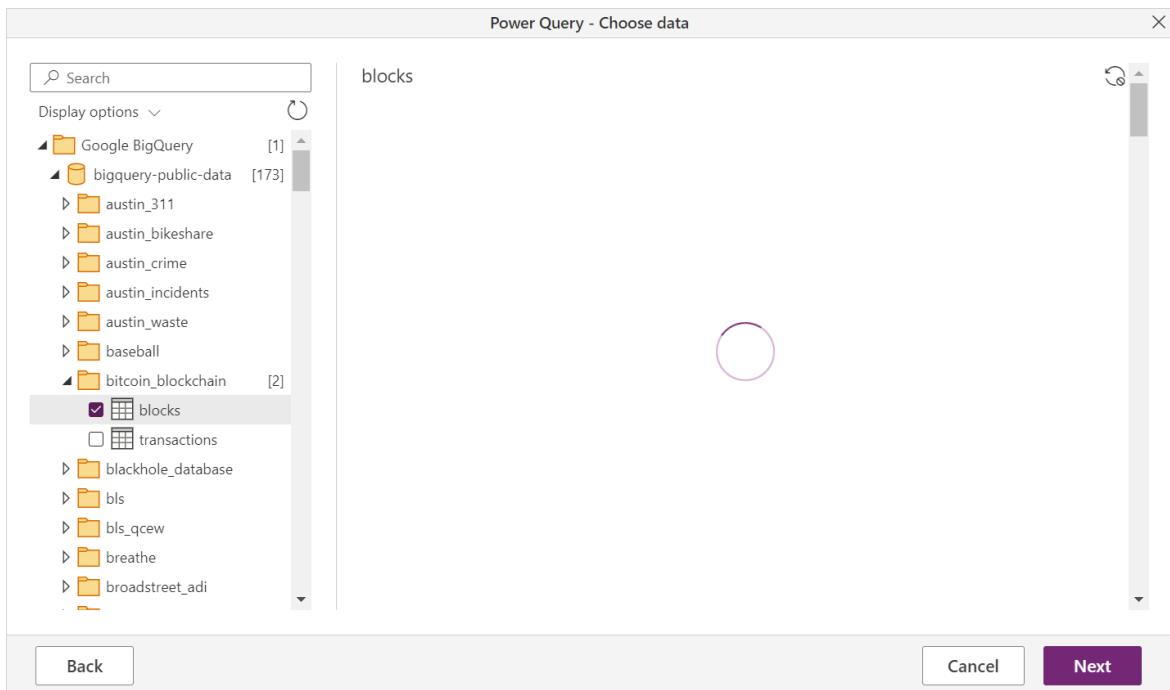
 [Use another account](#)

Before using this app, you can review Power BI Desktop's
[privacy policy](#) and Terms of Service.

4. If you want to use any advance options, select **Advanced options**. More information: [Connect using advanced options](#)
5. Once signed in, select **Next** to continue.

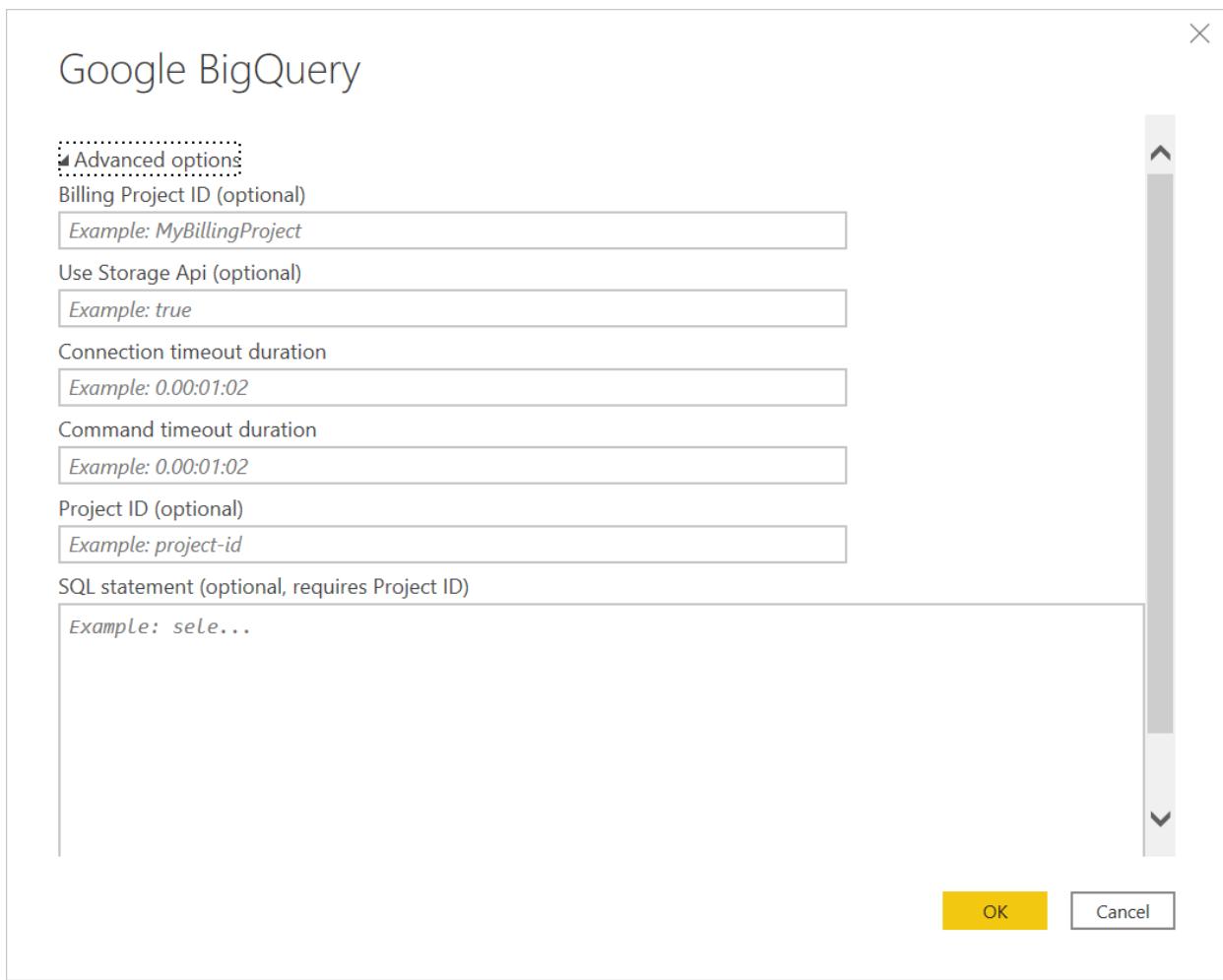


6. Once you successfully connect, a **Navigator** window appears and displays the data available on the server. Select your data in the navigator. Then select **Next** to transform the data in Power Query.



Connect using advanced options

Both Power Query Desktop and Power Query Online provide a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop and Power Query Online.

ADVANCED OPTION	DESCRIPTION
Billing Project ID	A project against which Power Query will run queries. Permissions and billing are tied to this project.
Use Storage Api	A flag that enables using the Storage API of Google BigQuery . This option is true by default. This option can be set to false to not use the Storage API and use REST APIs instead.
Connection timeout duration	The standard connection setting (in seconds) that controls how long Power Query waits for a connection to complete. You can change this value if your connection doesn't complete before 15 seconds (the default value.)
Command timeout duration	How long Power Query waits for a query to complete and return results. The default depends on the driver default. You can enter another value in minutes to keep the connection open longer.
Project ID	The project that you want to run native queries on. This option is only available in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
SQL statement	For information, go to Import data from a database using native database query . In this version of native database query functionality, you need to use fully qualified table names in the format <code>Database.Schema.Table</code> , for example <code>SELECT * FROM DEMO_DB.PUBLIC.DEMO_TABLE</code> . This option is only available in Power Query Desktop.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your Google BigQuery data.

Limitations and considerations

This section describes any limitations or considerations of the Google BigQuery connector.

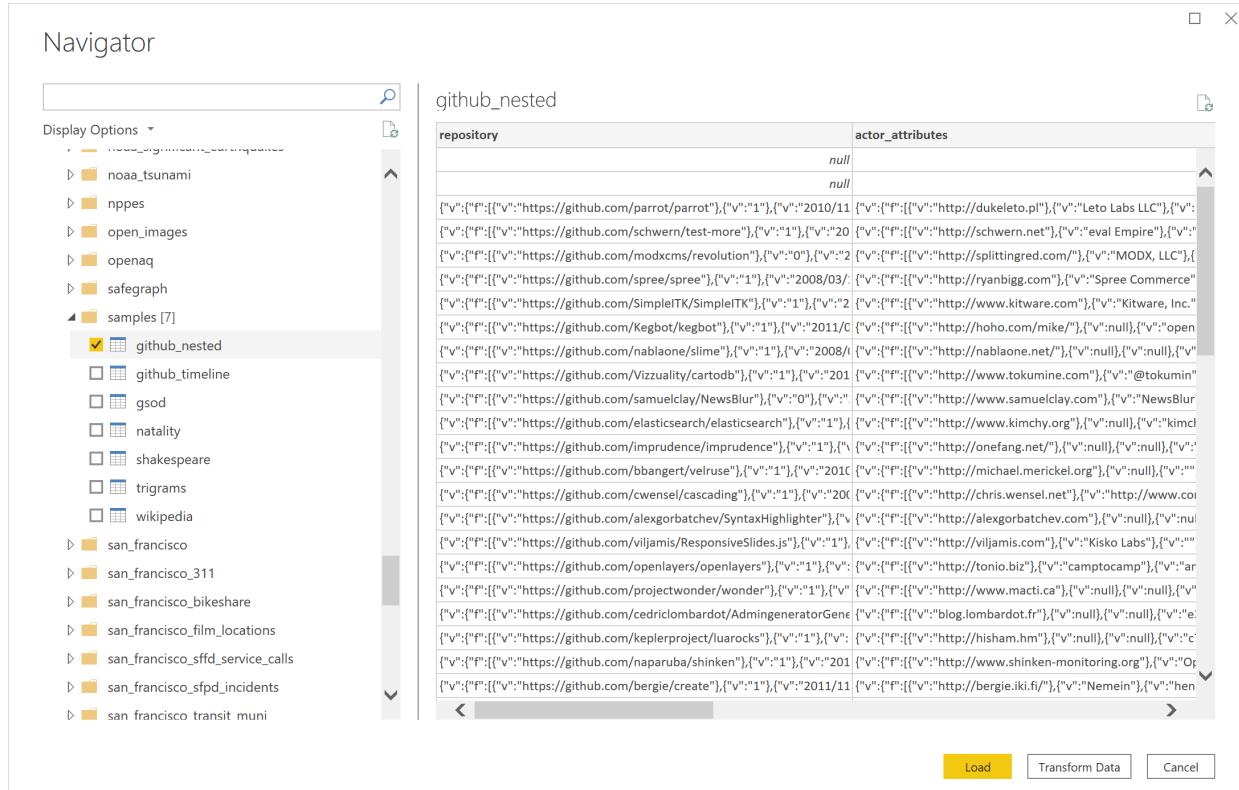
Connecting to Google Big Query in Power BI Desktop

For more information about limitations and considerations when connecting to Google Big Query, go to [Considerations and limitations](#).

Nested fields

To optimize performance considerations, Google BigQuery does well with large datasets when denormalized, flattened, and nested.

The Google BigQuery connector supports nested fields, which are loaded as text columns in JSON format.



The screenshot shows the Power Query Editor interface with the 'Navigator' pane on the left and the 'Power Query Editor' pane on the right. The 'Navigator' pane lists various datasets, with 'github_nested' selected. The 'Power Query Editor' pane displays a table named 'github_nested' with two columns: 'repository' and 'actor_attributes'. The 'repository' column contains a single row with the value 'null'. The 'actor_attributes' column contains a large JSON string representing nested fields. At the bottom of the editor, there are three buttons: 'Load', 'Transform Data', and 'Cancel'.

repository	actor_attributes
null	<pre>{"v": [{"f": [{"v": "https://github.com/parrot/parrot"}, {"v": "1"}, {"v": "2010/11"}], "v": [{"v": [{"f": [{"v": "http://dukeleto.pl"}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/schwern/test-more"}, {"v": "1"}, {"v": "20"}, {"v": [{"f": [{"v": "http://schwern.net"}]}]}]}, {"v": [{"v": [{"f": [{"v": "eval Empire"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/modxcms/modx"}, {"v": "0"}, {"v": "2"}, {"v": [{"f": [{"v": "http://splittingred.com"}]}]}]}, {"v": [{"v": [{"f": [{"v": "MODX, LLC"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/spree/spree"}, {"v": "1"}, {"v": "2008/03"}, {"v": [{"f": [{"v": "http://ryanbigg.com"}]}]}, {"v": [{"v": [{"f": [{"v": "Spree Commerce"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/SimpleTK/SimpleTK"}, {"v": "1"}, {"v": "2"}, {"v": [{"f": [{"v": "http://www.kitware.com"}]}]}, {"v": [{"v": [{"f": [{"v": "Kitware, Inc."}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/Kegbot/kegbot"}, {"v": "1"}, {"v": "2011/1"}, {"v": [{"f": [{"v": "http://hoho.com/mike/"}]}]}, {"v": [{"v": [{"f": [{"v": "open"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/nablaone/slime"}, {"v": "1"}, {"v": "2008/03"}, {"v": [{"f": [{"v": "http://nablaone.net/"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/Vizzuality/cartodb"}, {"v": "1"}, {"v": "2011/1"}, {"v": [{"f": [{"v": "http://www.tokumine.com"}]}]}, {"v": [{"v": [{"f": [{"v": "@tokumin"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/samuelclay/NewsBlur"}, {"v": "0"}, {"v": [{"f": [{"v": "http://www.samuelclay.com"}]}]}, {"v": [{"v": [{"f": [{"v": "NewsBlur"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/elasticsearch/elasticsearch"}, {"v": "1"}, {"v": [{"f": [{"v": "http://www.kimchy.org"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/imprudence/imprudence"}, {"v": "1"}, {"v": [{"f": [{"v": "http://onefang.net"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/bbangert/velruse"}, {"v": "1"}, {"v": "2010/1"}, {"v": [{"f": [{"v": "http://michael.merickel.org"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/cwensel/cascading"}, {"v": "1"}, {"v": "2010/1"}, {"v": [{"f": [{"v": "http://chris.wensel.net"}]}]}]}, {"v": [{"v": [{"f": [{"v": "http://www.co"}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/alexborbatchev/SyntaxHighlighter"}, {"v": "1"}, {"v": [{"f": [{"v": "http://alexborbatchev.com"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/viljamis/ResponsiveSlides.js"}, {"v": "1"}, {"v": [{"f": [{"v": "http://viljamis.com"}]}]}, {"v": [{"v": [{"f": [{"v": "Kisko Labs"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/openlayers/openlayers"}, {"v": "1"}, {"v": [{"f": [{"v": "http://tonio.biz"}]}]}, {"v": [{"v": [{"f": [{"v": "campocamp"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/problemwonder"}, {"v": "1"}, {"v": [{"f": [{"v": "http://www.macti.ca"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/cedriclombardot/AdmingeneratorGen"}, {"v": "1"}, {"v": [{"f": [{"v": "http://blog.lombardot.fr"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/keplerproject/uarocks"}, {"v": "1"}, {"v": [{"f": [{"v": "http://hisham.hm"}]}]}, {"v": [{"v": [{"f": [{"v": "null"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/narapuba/shinken"}, {"v": "1"}, {"v": "2011/1"}, {"v": [{"f": [{"v": "http://www.shinken-monitoring.org"}]}]}, {"v": [{"v": [{"f": [{"v": "Open"}]}]}]}]}, {"v": [{"v": [{"f": [{"v": "https://github.com/bergie/create"}, {"v": "1"}, {"v": "2011/1"}, {"v": [{"f": [{"v": "http://bergie.iki.fi/"}]}]}, {"v": [{"v": [{"f": [{"v": "Nemein"}]}]}]}]}]}]</pre>

Users should select **Transform Data** and then use the JSON parsing capabilities in the Power Query Editor to extract the data.

- Under the **Transforms** ribbon tab, the **Text Column** category, select **Parse** and then **JSON**.
- Extract the JSON record fields using the **Expand Column** option.

Setting up a Google service account

For more information on setting up or using Google service accounts, go to [Creating and managing service account keys](#) in the Google docs.

Authenticating through a Google service account

When authenticating through a Google service account in Power BI Desktop, there's a specific credential format that's required by the connector.

- Service Account Email: must be in email format
- Service Account JSON key file contents: once this JSON key is downloaded, all new lines must be removed from the file so that the contents are in one line. Once the JSON file is in that format, the contents can be pasted into this field.

When authenticating through a Google service account in Power BI service, users need to use "Basic" authentication. The **Username** field maps to the **Service Account Email** field above, and the **Password** field maps to the **Service Account JSON key file contents** field above. The format requirements for each credential remain the same in both Power BI Desktop and Power BI service.

Unable to authenticate with Google BigQuery Storage API

The Google BigQuery connector uses [Google BigQuery Storage API](#) by default. This feature is controlled by the advanced option called [UseStorageApi](#). You might encounter issues with this feature if you use granular permissions. In this scenario, you might see the following error message or fail to get any data from your query:

```
ERROR [HY000] [Microsoft][BigQuery] (131) Unable to authenticate with Google BigQuery Storage API. Check your account permissions
```

You can resolve this issue by adjusting the user permissions for the BigQuery Storage API correctly. These storage API permissions are required to access data correctly with BigQueryStorage API:

- `bigquery.readsessions.create` : Creates a new read session via the BigQuery Storage API.
- `bigquery.readsessions.getData` : Reads data from a read session via the BigQuery Storage API.
- `bigquery.readsessions.update` : Updates a read session via the BigQuery Storage API.

These permissions typically are provided in the `BigQuery.User` role. More information, [Google BigQuery Predefined roles and permissions](#)

If the above steps don't resolve the problem, you can disable the BigQuery Storage API.

Google Sheets (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI Desktop
Authentication Types Supported	Organizational account
Function Reference Documentation	-

Prerequisites

Before you can use the Google Sheets connector, you must have a Google account and have access to the Google Sheet you're trying to connect to.

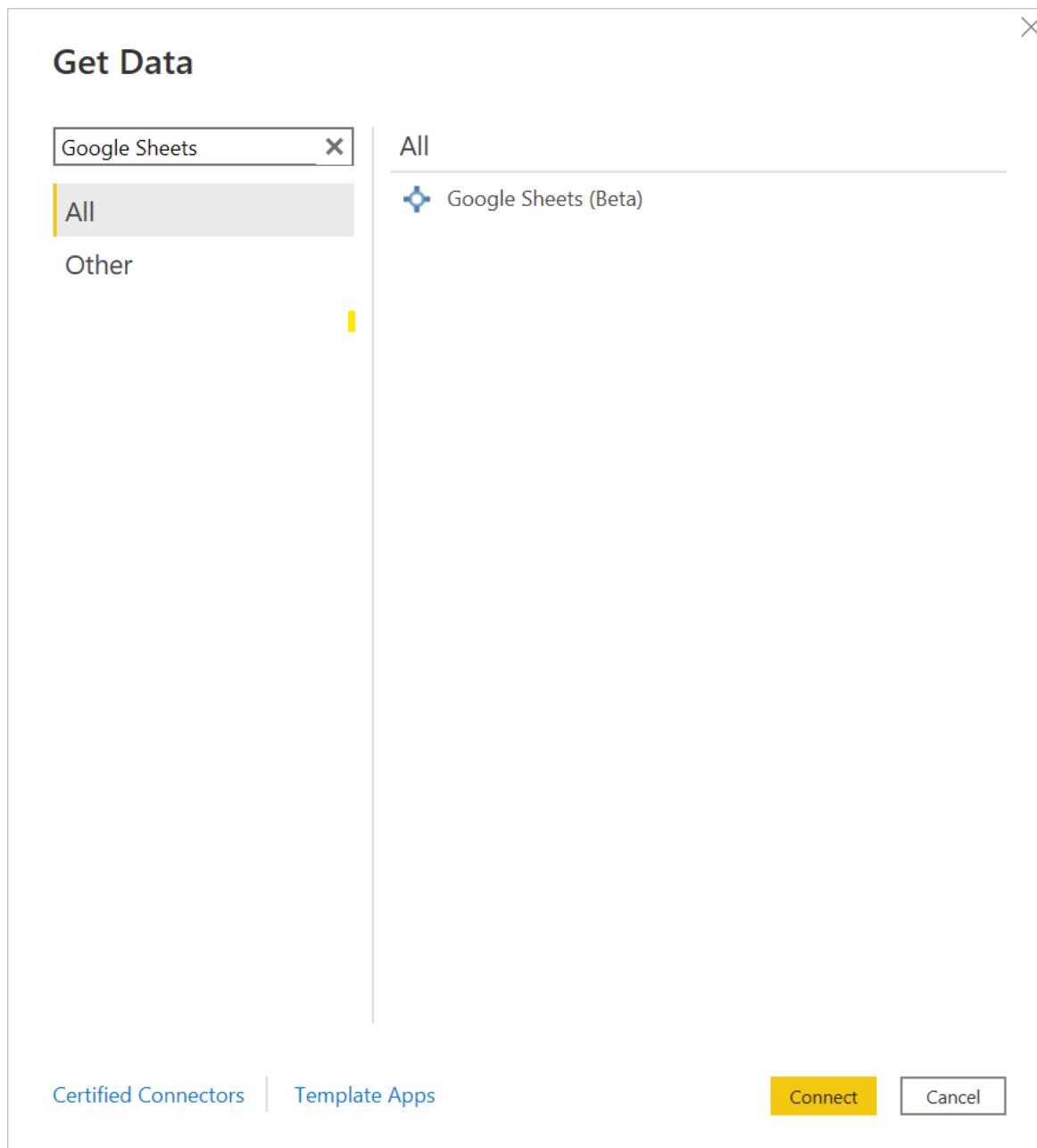
Capabilities Supported

- Import

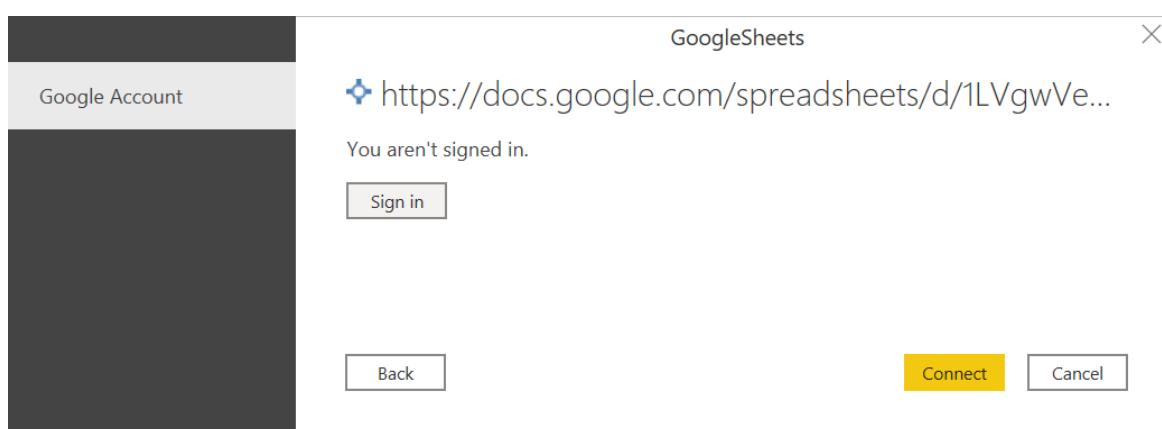
Connect to Google Sheets data from Power Query Desktop

To connect to Google Sheets from Power Query Desktop, take the following steps:

1. In the Get Data experience, search for and select **Google Sheets**.



2. You'll be prompted for a Google Sheets URL. Copy and paste the URL from your browser address bar into the input prompt.
3. The Google Sheets connector supports connecting through an organizational (Google) account. Select **Sign In** to continue.



4. A **Sign in with Google** dialog appears in an external browser window. Select your Google account and approve connecting to Power BI Desktop.



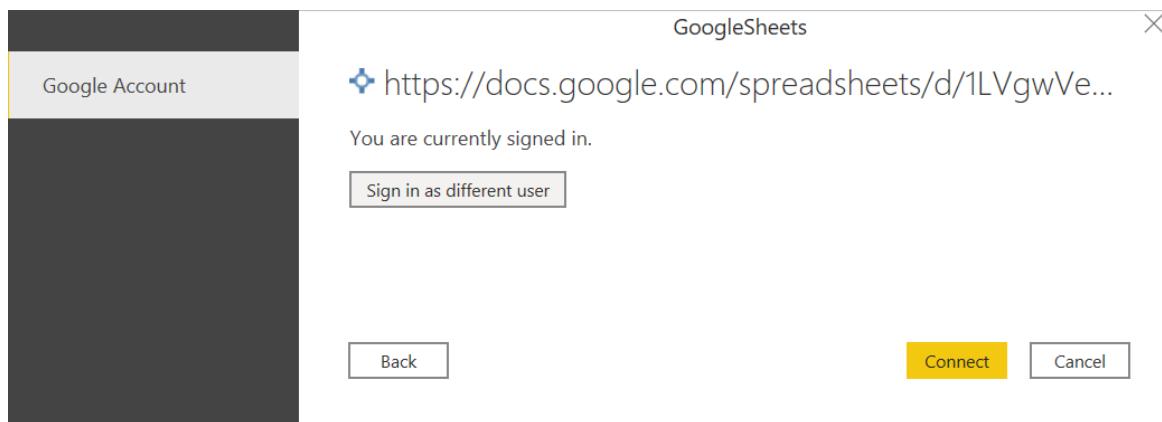
Choose an account

to continue to [Power BI Desktop](#)

[Use another account](#)

Before using this app, you can review Power BI Desktop's [privacy policy](#) and Terms of Service.

5. Once signed in, select **Connect** to continue.



6. Once you successfully connect, a **Navigator** window appears and displays the data available on the server. Select your data in the navigator. Then select either **Transform Data** to transform the data in Power Query or **Load** to load the data in Power BI Desktop.

Limitations and considerations

This section describes any limitations or considerations of the Google Sheets connector.

Power Query Online

This connector isn't yet available in Power Query Online and dataflows experiences.

Shared drive support

This connector does support connecting to shared drives.

Multiple connections

This connector uses a different ResourcePath for every Google Sheet URL. You'll need to authenticate to every new resource path and URL, but you might not need to sign into Google multiple times if the previous sessions remain active.

Understanding URL parsing

The connector first checks for the signature of the URL, ensuring it starts with

`https://docs.google.com/spreadsheets/d/`. The connector then parses the Google Spreadsheet ID from the URL to include in the Google Sheets API call. The rest of the URL isn't used. Each Google Sheet connection is tied to the submitted URL, which will act as the ResourcePath.

Hive LLAP

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Basic (Username/Password) Windows
Function Reference Documentation	—

Prerequisites

An Apache Hive LLAP username and password.

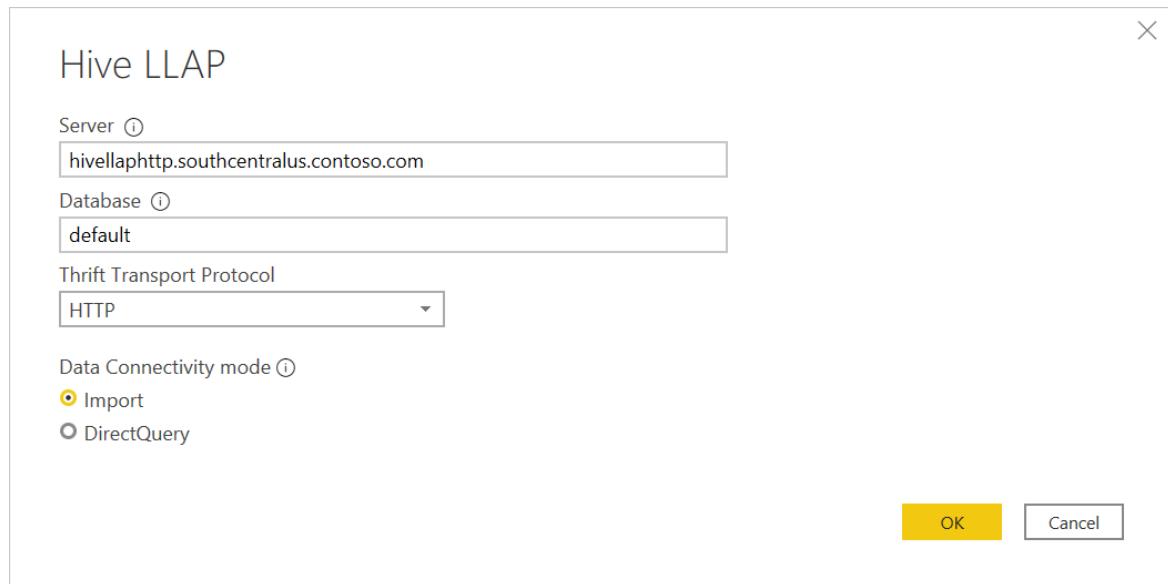
Capabilities Supported

- Import
- Direct Query
- Thrift Transport Protocol
 - HTTP
 - Standard

Connect to Hive LLAP data from Power Query Desktop

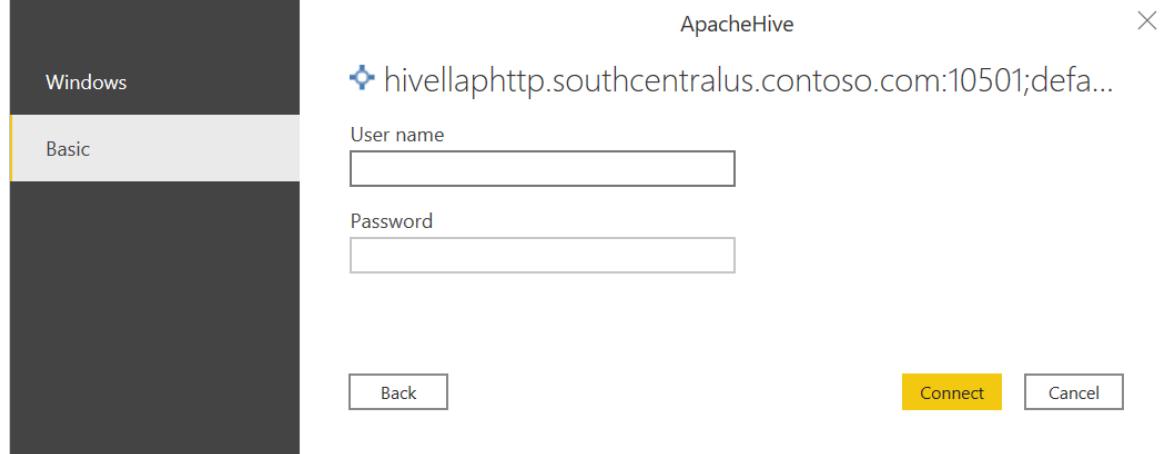
To connect to an Apache Hive LLAP server:

1. Select the **Hive LLAP** option from **Get Data**.
2. Enter the **URL** to the Adobe Hive LLAP server. You can also enter an optional port number. Typically, the URL looks like `http://[hostname]:[port number]`. The components of the URL are:
 - The `hostname` (for example, `hivellaphttp.southcentralus.contoso.com`) is the hostname or IP address of the Apache Hive server.
 - The `port number` (for example, 10500) is the port number for the Apache Hive server. If the `port number` isn't specified, the default value is 10501 for the HTTP transport protocol and 10500 for the standard transport protocol.



3. In **Thrift Transport Protocol**, select either **Standard** for TCP mode, or **HTTP** for HTTP mode.
4. Select either the **Import** or **DirectQuery** data connectivity mode. More information: [Use DirectQuery in Power BI Desktop](#)
5. Select **OK** to continue.
6. The first time you connect to a data source (identified by each unique URL), you'll be prompted to enter account credentials. Select the appropriate type of authentication and enter your credentials for the connection.
 - **Windows:** Select this authentication type if you want to connect using Windows authentication.
 - **Basic:** Select this authentication type if you want to connect using Apache Hive LLAP authentication. Enter your Apache Hive LLAP **User name** and **Password**.

More information: [Authentication with a data source](#).



7. Select **Connect** to connect to the Apache Hive LLAP data.
8. In **Navigator**, select the data you require. Then select either **Transform data** to transform the data in Power Query Editor or **Load** to load the data in Power BI Desktop.

The screenshot shows the Power BI Navigator interface. On the left, there's a tree view of data sources under a folder named 'hivellaphptt.southcentralus.contoso.com'. The 'sample_07' item is selected and highlighted with a yellow background. To the right, a preview pane displays a table titled 'sample_07' with four columns: 'code', 'description', 'total_emp', and 'salary'. The table lists various occupation codes and their descriptions, along with their respective employee counts and salaries. At the bottom of the preview pane are three buttons: 'Load' (highlighted in yellow), 'Transform Data', and 'Cancel'.

Kerberos-based single sign-on (SSO) for Hive LLAP

The Hive LLAP connector now supports Kerberos-based single sign-on (SSO).

To use this feature:

1. Sign in to your Power BI account, and navigate to the **Gateway management** page.
2. Add a new data source under the gateway cluster you want to use.
3. Select the connector in the **Data Source Type** list.
4. Expand the **Advanced Settings** section.
5. Select the option to **Use SSO via Kerberos for DirectQuery queries** or **Use SSO via Kerberos for DirectQuery and Import queries**.

ADD DATA SOURCE

GATEWAY CLUSTERS

- TestGateway23
- TestDataSource001
- New data source

Test all connections

Data Source Settings Users

Data Source Name: Kerberos SSO Hive LLAP

Data Source Type: Hive LLAP

Server:

Database:

Thrift Transport Protocol:

Authentication Method: Select an authentication method

Skip Test Connection

Advanced settings

Use SSO via Kerberos for DirectQuery queries
This setting will only be applied for DirectQuery datasets. Import will use the Username and Password specified in the data source details. [Learn more](#)

Use SSO via Kerberos for DirectQuery And Import queries
For Import, it will use the Dataset owner's credentials. [Learn more](#)

Connection Encryption setting for this data source: Encrypted

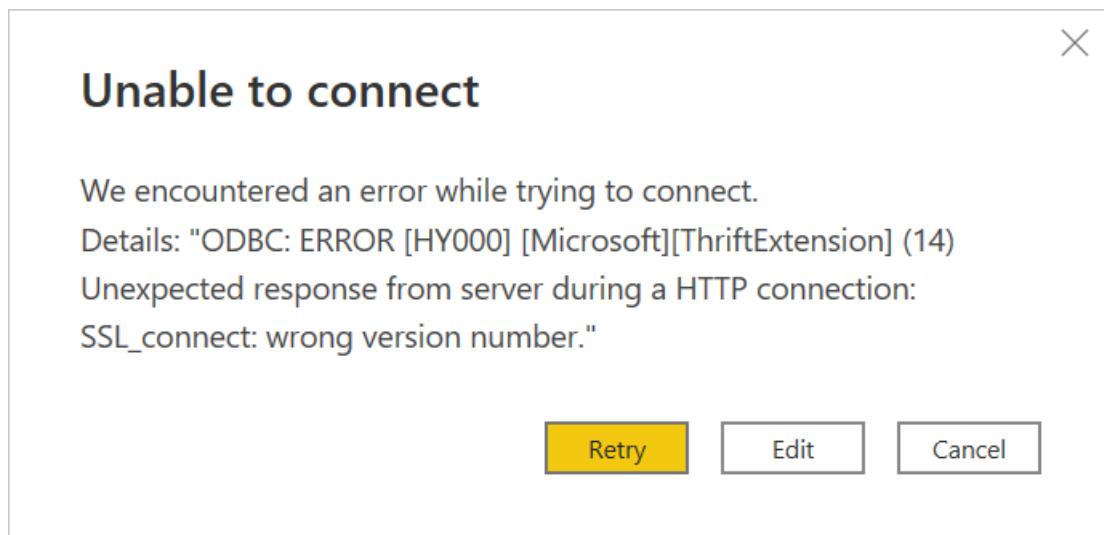
Privacy Level setting for this data source: Organizational

Add **Discard**

More information, [Configure Kerberos-based SSO from Power BI service to on-premises data sources](#)

Troubleshooting

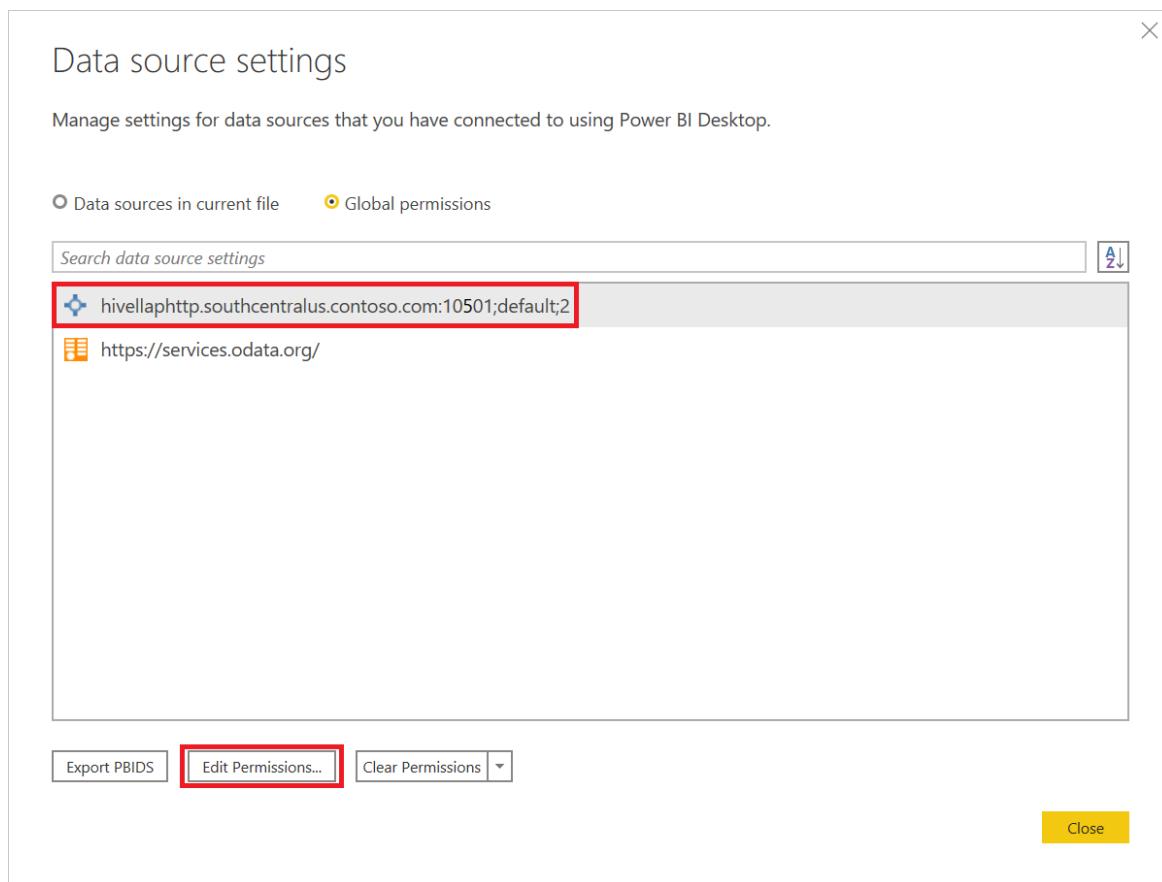
You might come across the following "SSL_connect" error after entering the authentication information for the connector and selecting **Connect**.



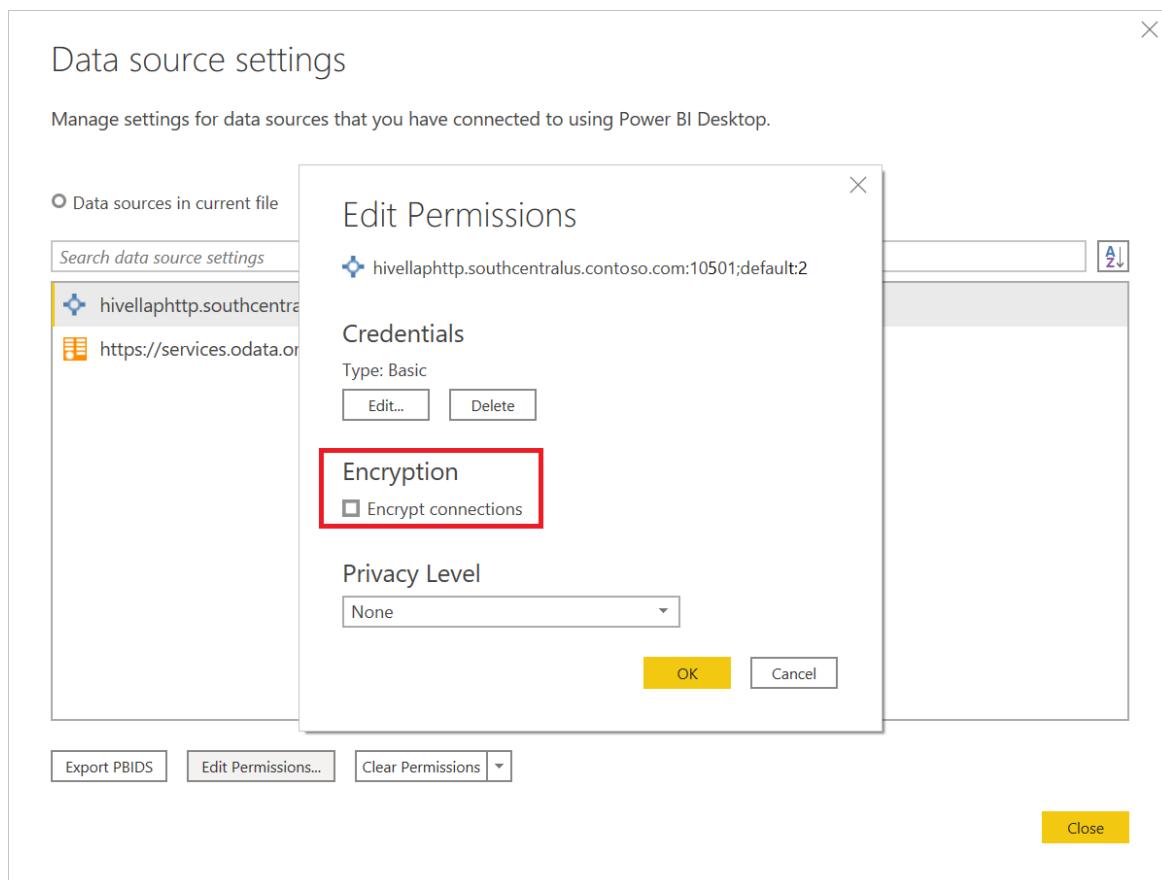
If this error occurs:

1. In Power BI Desktop, select **Files > Options and settings > Data source settings**.

2. In **Data source settings**, select the Hive LLAP source you created, and then select **Edit Permissions**.



3. In **Edit Permissions**, under **Encryption**, clear the **Encrypt connections** check box.



4. Select **OK**, and then in **Data source settings**, select **Close**.

5. Redo the steps in [Connect to Hive LLAP data from Power Query Desktop](#).

IBM Db2 database

1/15/2022 • 10 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Basic Database Windows
Function Reference Documentation	DB2.Database

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

By default, the IBM Db2 database connector uses the Microsoft driver to connect to your data. If you choose to use the IBM driver in the advanced options in Power Query Desktop, you must first install the IBM Db2 driver for .NET on the machine used to connect to the data. The name of this driver changes from time to time, so be sure to install the IBM Db2 driver that works with .NET. For instructions on how to download, install, and configure the IBM Db2 driver for .NET, go to [Download initial Version 11.5 clients and drivers](#). More information: [Driver limitations, Ensure the IBM Db2 driver is installed](#)

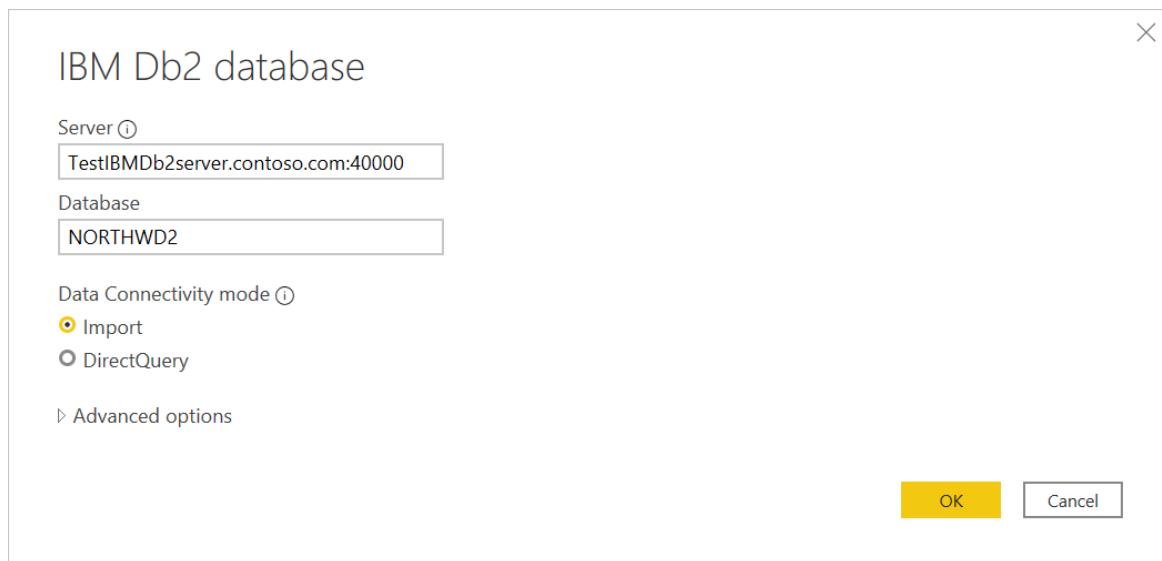
Capabilities Supported

- Import
- DirectQuery (Power BI Desktop only)
- Advanced options
 - Driver (IBM or Microsoft)
 - Command timeout in minutes
 - Package collection
 - SQL statement
 - Include relationship columns
 - Navigate using full hierarchy

Connect to an IBM Db2 database from Power Query Desktop

To make the connection, take the following steps:

1. Select the **IBM Db2 database** option from **Get Data**.
2. Specify the IBM Db2 server to connect to in **Server**. If a port is required, specify it by using the format *ServerName:Port*, where *Port* is the port number. Also, enter the IBM Db2 database you want to access in **Database**. In this example, the server name and port are `TestIBMDb2server.contoso.com:4000` and the IBM Db2 database being accessed is `NORTHWD2`.



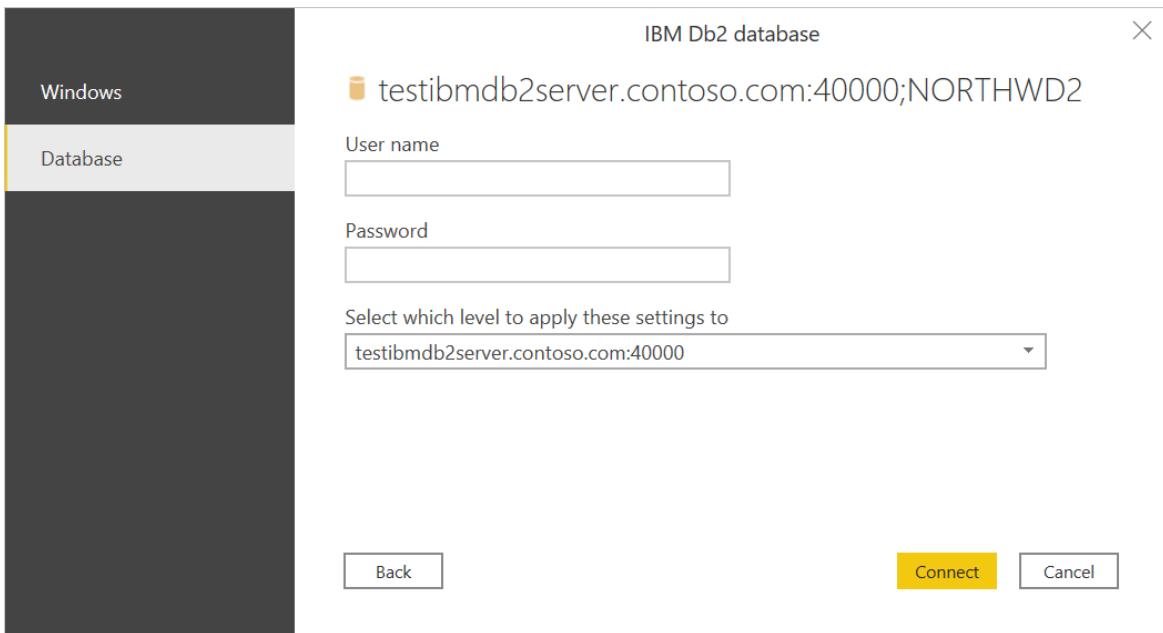
3. If you're connecting from Power BI Desktop, select either the **Import** or **DirectQuery** data connectivity mode. The rest of these example steps use the Import data connectivity mode. To learn more about DirectQuery, go to [Use DirectQuery in Power BI Desktop](#).

NOTE

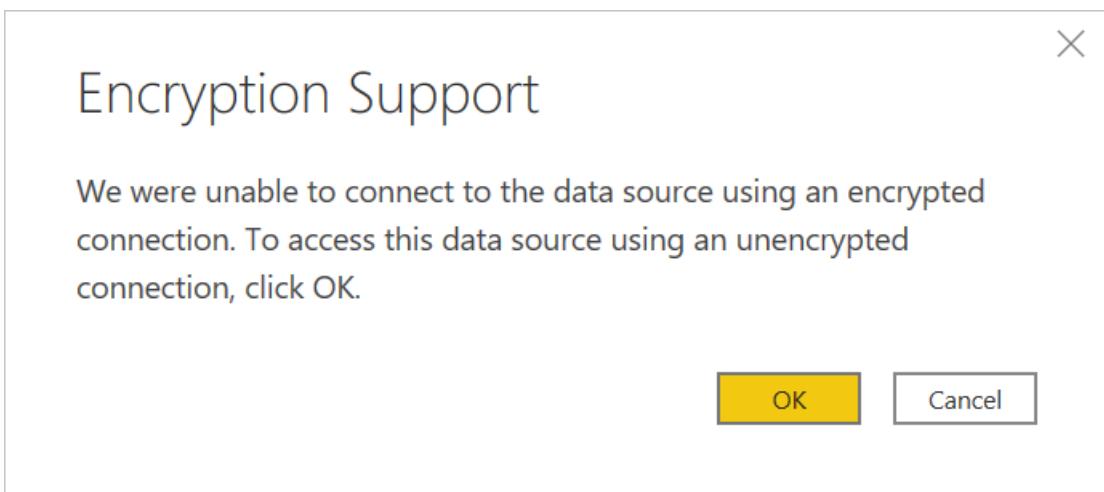
By default, the IBM Db2 database dialog box uses the Microsoft driver during sign in. If you want to use the IBM driver, open **Advanced options** and select **IBM**. More information: [Connect using advanced options](#)

If you select **DirectQuery** as your data connectivity mode, the **SQL statement** in the advanced options will be disabled. DirectQuery currently does not support query push down on top of a native database query for the IBM Db2 connector.

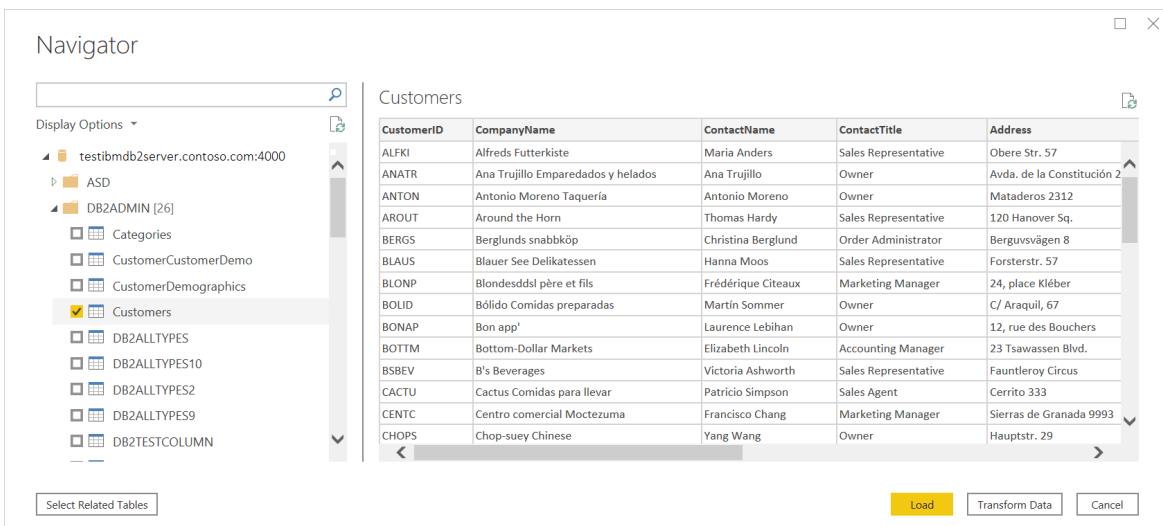
4. Select **OK**.
5. If this is the first time you're connecting to this IBM Db2 database, select the authentication type you want to use, enter your credentials, and then select **Connect**. For more information about authentication, go to [Authentication with a data source](#).



By default, Power Query attempts to connect to the IBM Db2 database using an encrypted connection. If Power Query can't connect using an encrypted connection, an "unable to connect" dialog box will appear. To connect using an unencrypted connection, select OK.



6. In **Navigator**, select the data you require, then either select **Load** to load the data or **Transform Data** to transform the data.



Connect to an IBM Db2 database from Power Query Online

To make the connection, take the following steps:

1. Select the **IBM Db2 database** option in the **Power Query - Connect to data source** page.
2. Specify the IBM Db2 server to connect to in **Server**. If a port is required, specify it by using the format *ServerName:Port*, where *Port* is the port number. Also, enter the IBM Db2 database you want to access in **Database**. In this example, the server name and port are `TestIBMDb2server.contoso.com:4000` and the IBM Db2 database being accessed is `NORTHWD2`
3. Select the name of your on-premises data gateway.

NOTE

You must select an on-premises data gateway for this connector, whether the IBM Db2 database is on your local network or online.

4. If this is the first time you're connecting to this IBM Db2 database, select the type of credentials for the connection in **Authentication kind**. Choose **Basic** if you plan to use an account that's created in the IBM Db2 database instead of Windows authentication.
5. Enter your credentials.
6. Select **Use Encrypted Connection** if you want to use an encrypted connection, or clear the option if you want to use an unencrypted connection.

The screenshot shows the 'Connection settings' and 'Connection credentials' sections of the Power Query - Connect to data source page for an IBM Db2 database.

Connection settings

- Server ***: TestIBMDb2Server.contoso.com:4000
- Database ***: NORTHWD2

Connection credentials

- Data gateway ***: [On-premises][Admin] TestGateway23
- Authentication kind**: Basic
- Username**: Db2Admin
- Password**: (redacted)
- Use Encrypted Connection**

7. Select **Next** to continue.
8. In **Navigator**, select the data you require, then select **Transform data** to transform the data in Power Query Editor.

Power Query - Choose data

The screenshot shows the 'Power Query - Choose data' dialog. On the left, there's a tree view of data sources: 'IBM Db2 database' (23 items), 'ASD' (1 item), and 'DB2ADMIN' (26 items). Under 'DB2ADMIN', several tables are listed: 'Categories', 'CustomerCustomerDe...', 'CustomerDemographics', 'Customers' (selected), 'DB2ALLTYPES', 'DB2ALLTYPES10', 'DB2ALLTYPES2', and 'DB2ALLTYPES9'. A 'Select related tables' button is at the bottom of this list. On the right, a preview of the 'Customers' table is shown with columns: CustomerID, CompanyName, ContactName, ContactTitle, and Address. The data includes rows for ALFKI, ANATR, ANTON, AROUT, BERGS, BLAUS, BLONP, BOLID, BONAP, BOTTM, BSBEV, CACTU, CENTC, and CHOPS. At the bottom right are 'Cancel' and 'Transform data' buttons.

Connect using advanced options

Power Query provides a set of advanced options that you can add to your query if needed.

Advanced options

Driver

IBM

Microsoft (requires .Net 4.5 or later)

Command timeout in minutes (optional)

Package collection (optional, Microsoft driver only)

SQL statement (optional, requires database)

Include relationship columns

Navigate using full hierarchy

The following table lists all of the advanced options you can set in Power Query.

ADVANCED OPTION	DESCRIPTION
Driver	Determines which driver is used to connect to your IBM Db2 database. The choices are IBM and Windows (default). If you select the IBM driver, you must first ensure that the IBM Db2 driver for .NET is installed on your machine. This option is only available in Power Query Desktop. More information: Ensure the IBM Db2 driver is installed
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer.

ADVANCED OPTION	DESCRIPTION
Package collection	Specifies where to look for packages. Packages are control structures used by Db2 when processing an SQL statement, and will be automatically created if necessary. By default, this option uses the value <code>NULLID</code> . Only available when using the Microsoft driver. More information: DB2 packages: Concepts, examples, and common problems
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your IBM Db2 database.

Issues and limitations

Driver limitations

The Microsoft driver is the same one used in Microsoft Host Integration Server, called the "ADO.NET Provider for DB2". The IBM driver is the IBM Db/2 driver that works with .NET. The name of this driver changes from time to time, so be sure it's the one that works with .NET, which is different from the IBM Db2 drivers that work with OLE/DB, ODBC, or JDBC.

You can choose to use either the Microsoft driver (default) or the IBM driver if you're using Power Query Desktop. Currently, Power Query Online only uses the Microsoft driver. Each driver has its limitations.

- Microsoft driver
 - Doesn't support Transport Layer Security (TLS)
- IBM driver
 - The IBM Db2 database connector, when using the IBM Db2 driver for .NET, doesn't work with Mainframe or IBM i systems
 - Doesn't support DirectQuery

Microsoft provides support for the Microsoft driver, but not for the IBM driver. However, if your IT department already has it set up and configured on your machines, your IT department should know how to troubleshoot the IBM driver.

Native queries not supported in DirectQuery

When you select DirectQuery as the data connectivity mode in Power Query Desktop, the SQL statement text box in the advanced options is disabled. It's disabled because the Power Query IBM Db2 connector doesn't currently support query push down on top of a native database query.

Troubleshooting

Ensure the IBM Db2 driver is installed

If you choose to use the IBM Db2 driver for Power Query Desktop, you first have to download, install, and configure the driver on your machine. To ensure the IBM Db2 driver has been installed:

1. Open Windows PowerShell on your machine.
2. Enter the following command:

```
[System.Data.Common.DbProviderFactories]::GetFactoryClasses() | ogv
```

3. In the dialog box that opens, you should see the following name in the **InvariantName** column:

```
IBM.Data.DB2
```

If this name is in the **InvariantName** column, the IBM Db2 driver has been installed and configured correctly.

SQLCODE -805 and SQLCODE -551 error codes

When attempting to connect to an IBM Db2 database, you may sometimes come across the common error SQLCODE -805, which indicates the package isn't found in the **NULLID** or other collection (specified in the Power Query **Package connection** configuration). You may also encounter the common error SQLCODE -551, which indicates you can't create packages because you lack package binding authority.

Typically, SQLCODE -805 is followed by SQLCODE -551, but you'll see only the second exception. In reality, the problem is the same. You lack the authority to bind the package to either **NULLID** or the specified collection.

Typically, most IBM Db2 administrators don't provide bind package authority to end users—especially in an IBM z/OS (mainframe) or IBM i (AS/400) environment. Db2 on Linux, Unix, or Windows is different in that user accounts have bind privileges by default, which create the MSCS001 (Cursor Stability) package in the user's own collection (name = user login name).

If you don't have bind package privileges, you'll need to ask your Db2 administrator for package binding authority. With this package binding authority, connect to the database and fetch data, which will auto-create the package. Afterwards, the administrator can revoke the packaging binding authority. Also, afterwards, the administrator can "bind copy" the package to other collections—to increase concurrency, to better match your internal standards for where packages are bound, and so on.

When connecting to IBM Db2 for z/OS, the Db2 administrator can do the following steps.

1. Grant authority to bind a new package to the user with one of the following commands:
 - GRANT BINDADD ON SYSTEM TO <*authorization_name*>
 - GRANT PACKADM ON <*collection_name*> TO <*authorization_name*>
2. Using Power Query, connect to the IBM Db2 database and retrieve a list of schemas, tables, and views.
The Power Query IBM Db2 database connector will auto-create the package **NULLID.MSCS001**, and then grant execute on the package to public.
3. Revoke authority to bind a new package to the user with one of the following commands:
 - REVOKE BINDADD FROM <*authorization_name*>
 - REVOKE PACKADM ON <*collection_name*> FROM <*authorization_name*>

When connecting to IBM Db2 for Linux, Unix, or Windows, the Db2 administrator can do the following steps.

1. GRANT BINDADD ON DATABASE TO USER <*authorization_name*>.
2. Using Power Query, connect to the IBM Db2 database and retrieve a list of schemas, tables, and views.
The Power Query IBM Db2 connector will auto-create the package **NULLID.MSCS001**, and then grant execute on the package to public.
3. REVOKE BINDADD ON DATABASE FROM USER <*authorization_name*>.

4. GRANT EXECUTE ON PACKAGE <collection.package> TO USER <authorization_name>.

When connecting to IBM Db2 for i, the Db2 administrator can do the following steps.

1. WRKOBJ QSYS/CRTSQLPKG. Type "2" to change the object authority.
2. Change authority from *EXCLUDE to PUBLIC or <authorization_name>.
3. Afterwards, change authority back to *EXCLUDE.

SQLCODE -360 error code

When attempting to connect to the IBM Db2 database, you may come across the following error:

```
Microsoft Db2 Client: The host resource could not be found. Check that the Initial Catalog value matches the host resource name. SQLSTATE=HY000 SQLCODE=-360
```

This error message indicates that you didn't put the right value in for the name of the database.

SQLCODE -1336 error code

```
The specified host could not be found.
```

Double check the name, and confirm that the host is reachable. For example, use [ping](#) in a command prompt to attempt to reach the server and ensure the IP address is correct, or use [telnet](#) to communicate with the server.

SQLCODE -1037 error code

```
Host is reachable, but is not responding on the specified port.
```

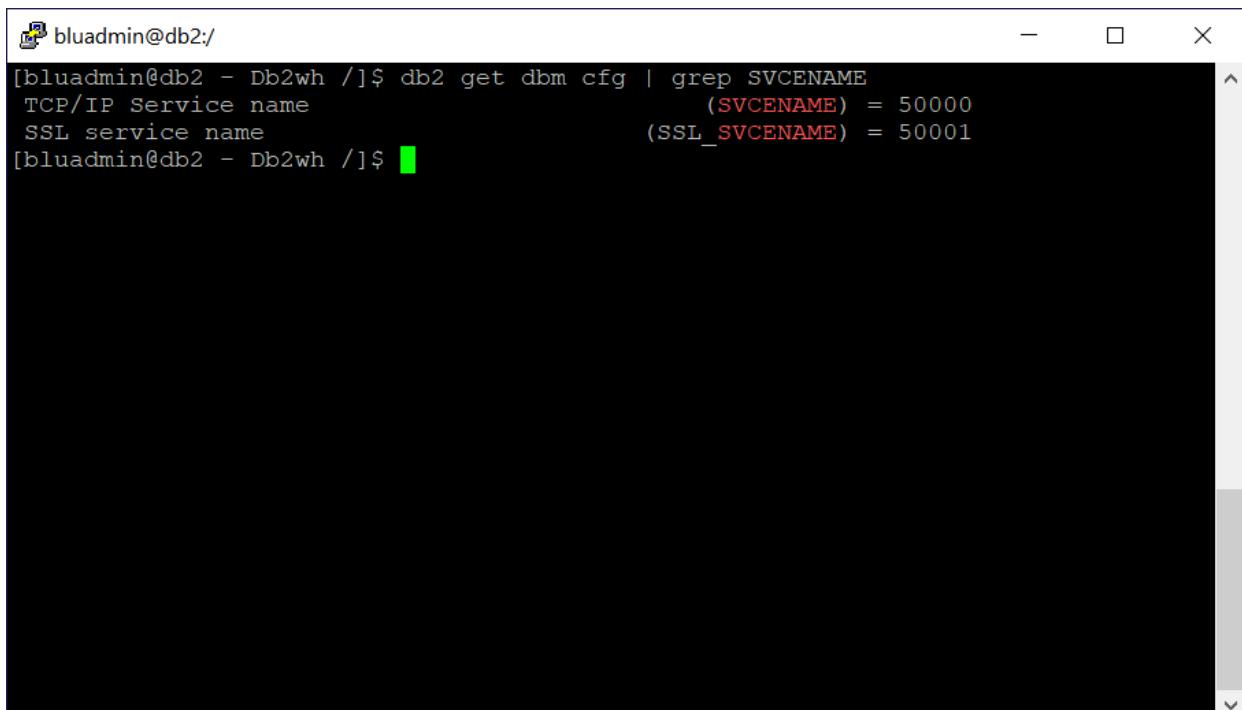
The port is specified at the end of the server name, separated by a colon. If omitted, the default value of 50000 is used.

To find the port Db2 is using for Linux, Unix, and Windows, run this command:

```
db2 get dbm cfg | findstr SVCENAME
```

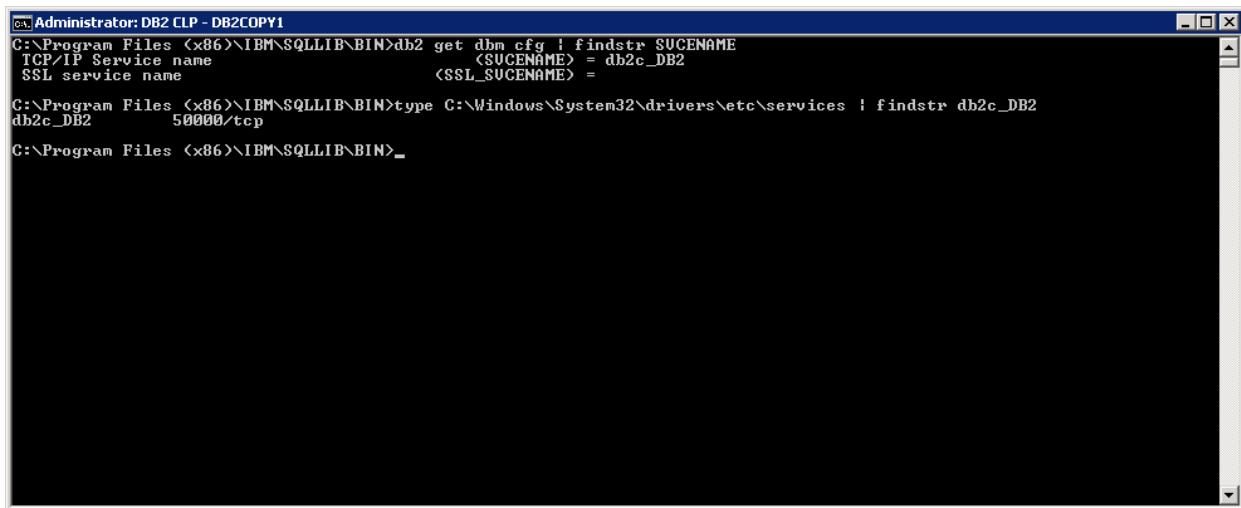
Look in the output for an entry for SVCENAME (and SSL_SVCENAME for TLS encrypted connections). If this value is a number, that's the port. Otherwise cross reference the value with the system's "services" table. You can usually find this at /etc/services, or at c:\windows\system32\drivers\etc\services for Windows.

The following screenshot shows the output of this command in Linux/Unix.



```
bluadmin@db2:/ [bluadmin@db2 - Db2wh /]$ db2 get dbm cfg | grep SVCENAME
TCP/IP Service name          (SVCENAME) = 50000
SSL service name              (SSL_SVCENAME) = 50001
[bluadmin@db2 - Db2wh /]$
```

The following screenshot shows the output of this command in Windows.

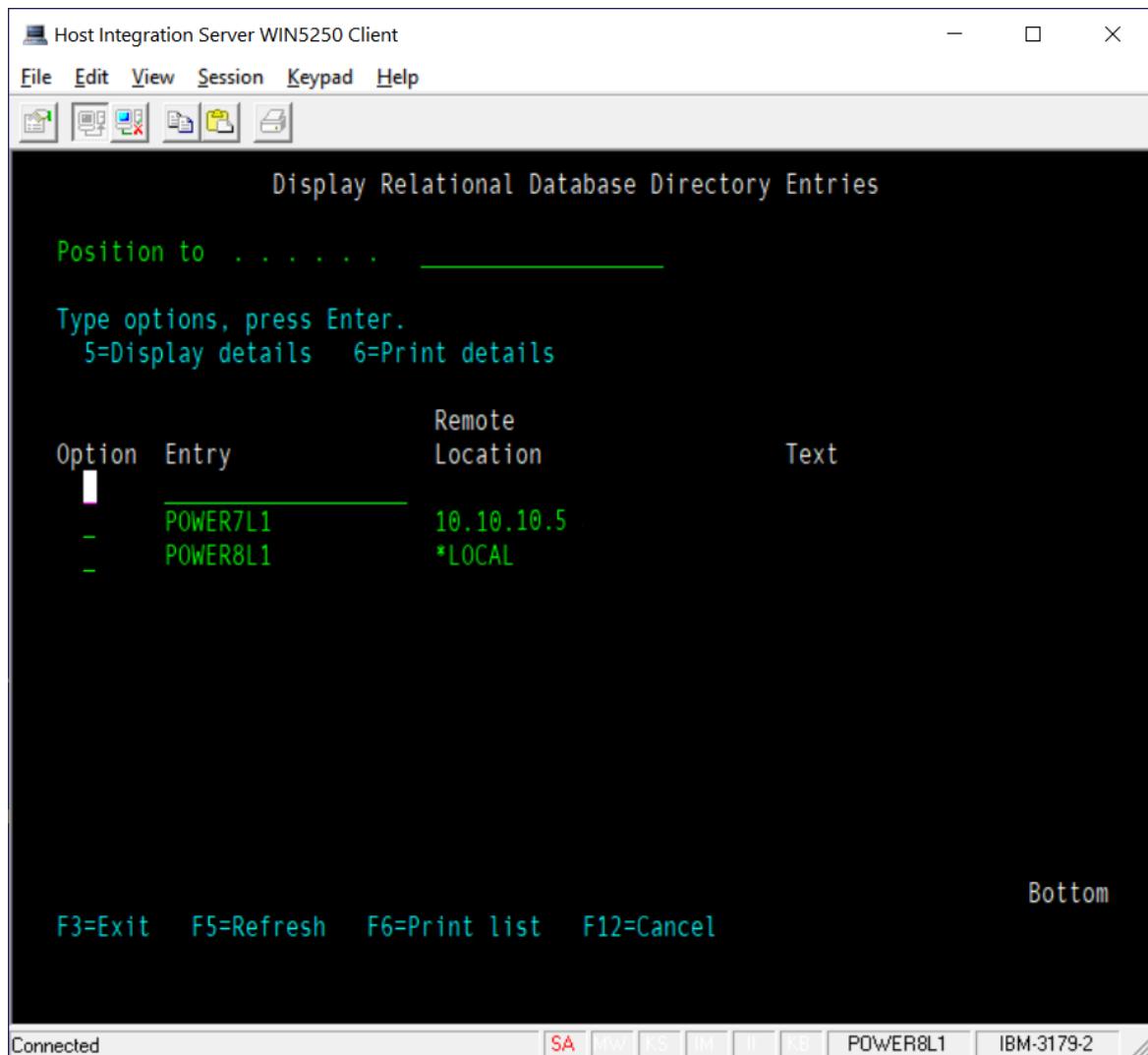


```
C:\>Administrator: DB2 CLP - DB2COPY1
C:\Program Files <x86>\IBM\SQLLIB\BIN>db2 get dbm cfg | findstr SUCENAME
TCP/IP Service name           <SUCENAME> = db2c_DB2
SSL service name              <SSL_SUCENAME> =
C:\Program Files <x86>\IBM\SQLLIB\BIN>
```

Determine database name

To determine the database name to use:

1. On IBM i, run `DSPRDBDIR`.



Option	Entry	Remote Location	Text
-	POWER7L1	10.10.10.5	
-	POWER8L1	*LOCAL	

F3=Exit F5=Refresh F6=Print list F12=Cancel

Connected SA POWER8L1 IBM-3179-2

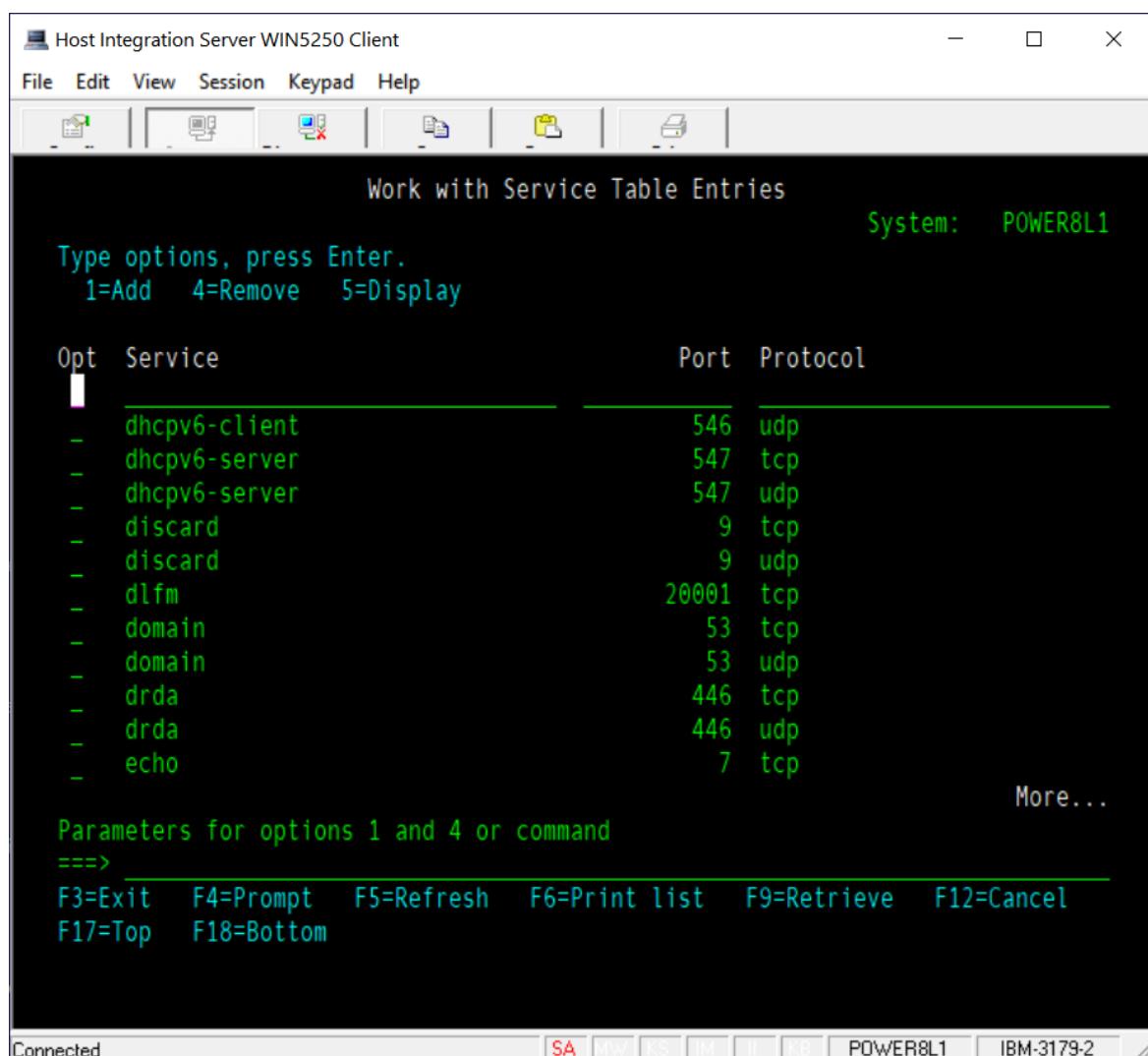
2. One of the entries will have a **Remote Location** of ***LOCAL**. This entry is the one to use.

Determine port number

The Microsoft driver connects to the database using the Distributed Relational Database Architecture (DRDA) protocol. The default port for DRDA is port 446. Try this value first.

To find for certain what port the DRDA service is running on:

1. Run the IBM i command `WRKSRVTBLE`.
2. Scroll down until you find the entries for DRDA.



Opt	Service	Port	Protocol
-	dhcpv6-client	546	udp
-	dhcpv6-server	547	tcp
-	dhcpv6-server	547	udp
-	discard	9	tcp
-	discard	9	udp
-	dlfm	20001	tcp
-	domain	53	tcp
-	domain	53	udp
-	drda	446	tcp
-	drda	446	udp
-	echo	7	tcp

Type options, press Enter.
1=Add 4=Remove 5=Display

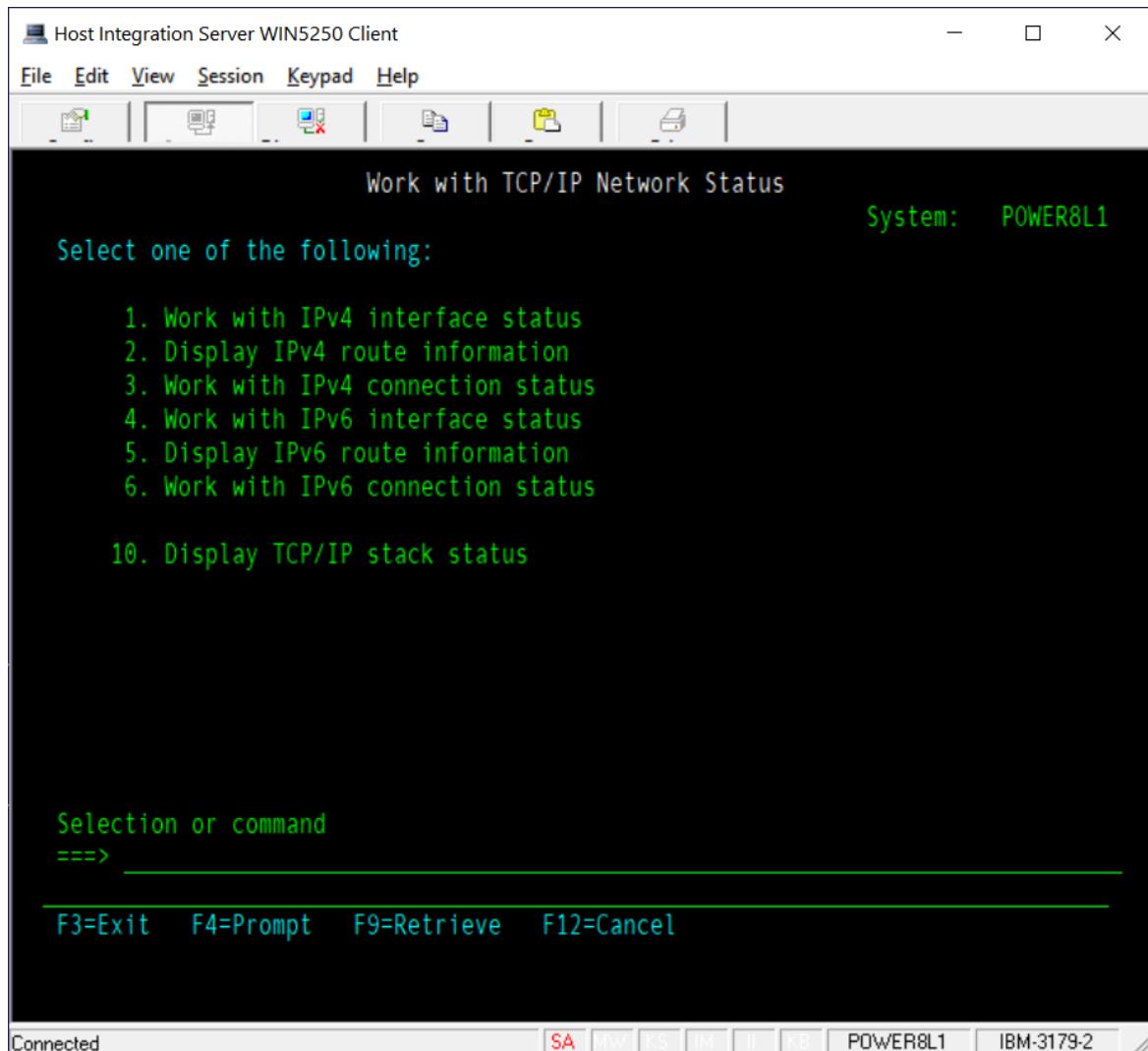
More...

Parameters for options 1 and 4 or command
==>

F3=Exit F4=Prompt F5=Refresh F6=Print list F9=Retrieve F12=Cancel
F17=Top F18=Bottom

Connected SA IBM-3179-2 POWER8L1 IBM-3179-2

3. To confirm that the DRDA service is up and listening on that port, run `NETSTAT`.



4. Choose either option 3 (for IPv4) or option 6 (for IPv6).
5. Press F14 to see the port numbers instead of names, and scroll until you see the port in question. It should have an entry with a state of "Listen".

Host Integration Server WIN5250 Client

File Edit View Session Keypad Help

Work with IPv4 Connection Status System: POWER8L1

Type options, press Enter.

3=Enable debug 4=End 5=Display details 6=Disable debug
8=Display jobs

Opt	Remote	Remote	Local		
	Address	Port	Port	Idle Time	State
-	*	*	21	044:56:18	Listen
-	*	*	23	000:43:49	Listen
-	*	*	25	045:24:08	Listen
-	*	*	137	++++++	Listen
-	*	*	137	000:01:05	*UDP
-	*	*	138	000:01:04	*UDP
-	*	*	139	045:24:48	Listen
-	*	*	389	++++++	Listen
-	*	*	427	000:02:42	*UDP
-	*	*	445	000:41:19	Listen
-	*	*	446	000:08:14	Listen
-	*	*	447	++++++	Listen

More...
F3=Exit F5=Refresh F9=Command line F11=Display byte counts F12=Cancel
F15=Subset F20=Work with IPv6 connections F24=More keys

Connected SA MW RS IM IL LB POWER8L1 IBM-3179-2

More information

- HIS - Microsoft OLE DB Provider for DB2

JSON

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Anonymous Basic (Web only) Organizational Account Web API (Web only) Windows
Function Reference Documentation	Json.Document

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

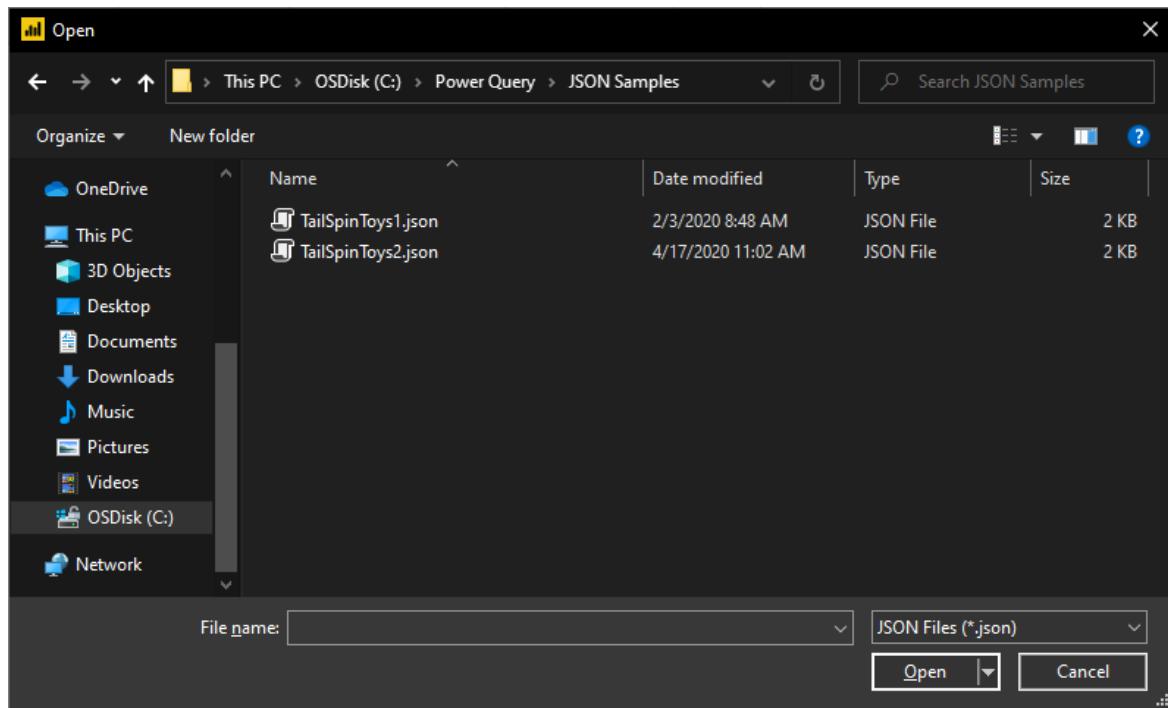
Capabilities supported

- Import

Load a local JSON file from Power Query Desktop

To load a local JSON file:

1. Select the **JSON** option in the **Get Data** selection. This selection launches a local file browser where you can select your JSON file.



2. Select Open to open the file.

Loading the JSON file will automatically launch the Power Query Editor. Power Query uses automatic table detection to seamlessly flatten the JSON data into a table. From the editor, you can then continue to transform the data if you want, or you can just close and apply. More information: [Automatic table detection from JSON files](#)

Load a local JSON file from Power Query Online

To load a local JSON file:

- From the Data sources page, select **JSON**.
- Enter the path to the local JSON file.

Connection settings

File path or URL *

3. Select an on-premises data gateway from **Data gateway**.

4. If authentication is required, enter your credentials.

5. Select **Next**.

Loading the JSON file will automatically launch the Power Query Editor. Power Query uses automatic table detection to seamlessly flatten the JSON data into a table. From the editor, you can then continue to transform the data if you want, or you can just save and close to load the data. More information: [Automatic table detection from JSON files](#)

Power Query - Edit queries

Home Transform Add column View Help

Get data Enter data Options Manage parameters Refresh Properties Advanced editor Choose columns Remove columns Keep rows Remove rows Filter rows Reduce rows Sort Transform Combine

Queries [1]

Completed (2.24 s) Columns: 38 Rows: 1

Query settings

Name: Query

Entity type: Custom

Applied steps:

- Source
- Converted to ...
- Expanded price
- Expanded sal...
- Expanded shi...
- Changed colu...

Step Cancel Save & close

Load from the web

To load a JSON file from the web, select the [Web connector](#), enter the web address of the file, and follow any credential prompts.

Automatic table detection from JSON files

Importing data from JSON files (or Web APIs) can be challenging for end users. Here is an example of JSON file with multiple levels of nested data.

```
{ TailSpinToys1.json } X
C: > test-examples > JSON > { TailSpinToys1.json } > ...
1  {
2    "additionalImageLinks": [
3      "http://www.tailspintoys.com/images?id=123def",
4      "http://www.tailspintoys.com/images?id=567def"
5    ],
6    "adult": "False",
7    "adwordsRedirect": "http://contoso.com/hury",
8    "ageGroup": "kids",
9    "availability": "in stock",
10   "brand": "Tailspin",
11   "channel": "online",
12   "color": "Blue",
13   "condition": "new",
14   "contentLanguage": "en",
15   "description": "Charming sundress perfect for lunch out on the town.",
16   "expirationDate": "2016-07-14T07:27:06-08:00",
17   "gender": "female",
18   "id": "Online:en:US:sku1234",
19   "identifierExists": "True",
20   "imageLink": "http://www.tailspintoys.com/images?id=123abc",
21   "itemGroupId": "abc123def456",
22   "kind": "content#product",
23   "link": "http://www.tailspintoys.com/girls/apparel?id=9d0s-a934",
24   "material": "cotton",
25   "offerId": "sku1234",
26   "onlineOnly": "False",
27   "price": {
28     "currency": "USD",
29     "value": 38.000000
30   },
31   "productType": "Apparel & Accessories > Clothing > Dresses",
32   "salePrice": {
33     "currency": "USD",
34     "value": 25.000000
35   },
36   "salePriceEffectiveDate": "2016-06-14T08:00:00-08:00/2016-06-21T17:00:00-08:00",
37   "shipping": [
38     {
39       "country": "US",
40       "price": {
41         "currency": "USD",
42         "value": 3.00
43       }
44     }
45   ]
46 }
```

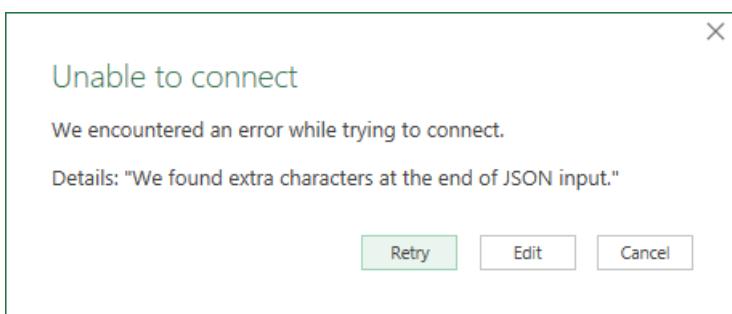
With the addition of automatic table detection capabilities, using the JSON connector in Power Query will automatically apply transformation steps to flatten the JSON data into a table. Previously, users had to flatten records and lists manually.

The screenshot shows the Power Query Editor interface. In the center, there's a preview pane displaying a table with columns: 'APC salePrice.currency', 'salePrice.value', and 'APC salePriceEffectiveDate'. Below the preview, it says '38 COLUMNS, 1 ROW' and 'Column profiling based on top 1000 rows'. On the right, the 'Query Settings' pane is open, showing 'PROPERTIES' with a 'Name' field containing 'TailSpinToys1'. The 'APPLIED STEPS' section is highlighted with a red box, listing the following steps:

- Source
- Converted to Table
- Expanded price
- Expanded salePrice
- Expanded shippingWeight
- Changed Type

Troubleshooting

If you see the following message, it might be because the file is invalid, for example, it's not really a JSON file, or is malformed. Or you might be trying to load a JSON Lines file.



If you're trying to load a JSON Lines file, the following sample M code converts all JSON Lines input to a single flattened table automatically:

```
let
    // Read the file into a list of lines
    Source = Table.FromColumns({Lines.FromBinary(File.Contents("C:\json-lines-example.json"), null, null)}),
    // Transform each line using Json.Document
    #"Transformed Column" = Table.TransformColumns(Source, {"Column1", Json.Document})
in
    #"Transformed Column"
```

You'll then need to use an *Expand* operation to combine the lines together.

Mailchimp (Deprecated)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Deprecated
Products	-
Authentication Types Supported	-
Function Reference Documentation	-

Deprecation

This connector is deprecated, and won't be supported soon. We recommend you transition off existing connections using this connector, and don't use this connector for new connections.

Microsoft Azure Consumption Insights (Beta) (Deprecated)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Deprecated
Products	-
Authentication Types Supported	Organizational account
Function Reference Documentation	-

Deprecation

NOTE

This connector is deprecated because of end of support for the Microsoft Azure Consumption Insights service. We recommend that users transition off existing connections using this connector, and don't use this connector for new connections.

Transition instructions

Users are instructed to use the certified Microsoft Azure Cost Management connector as a replacement. The table and field names are similar and should offer the same functionality.

Timeline

The Microsoft Azure Consumption Insights service will stop working in December 2021. Users should transition off the Microsoft Azure Consumption Insights connector to the Microsoft Azure Cost Management connector by December 2021.

Microsoft Graph Security (Deprecated)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Deprecated
Products	-
Authentication Types Supported	Organizational account
Function Reference Documentation	-

Deprecation

NOTE

This connector is deprecated. We recommend that you transition off existing connections using this connector, and don't use this connector for new connections.

MySQL database

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Windows (Power BI Desktop, Excel, online service with gateway) Database (Power BI Desktop, Excel) Basic (online service with gateway)
Function Reference Documentation	MySQL.Database

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

Users need to install the [Oracle MySQL Connector/.NET](#) package prior to using this connector in Power BI Desktop. This component must also be installed on the machine running the on-premises data gateway in order to use this connector in Power Query Online (dataflows) or Power BI Service.

Capabilities Supported

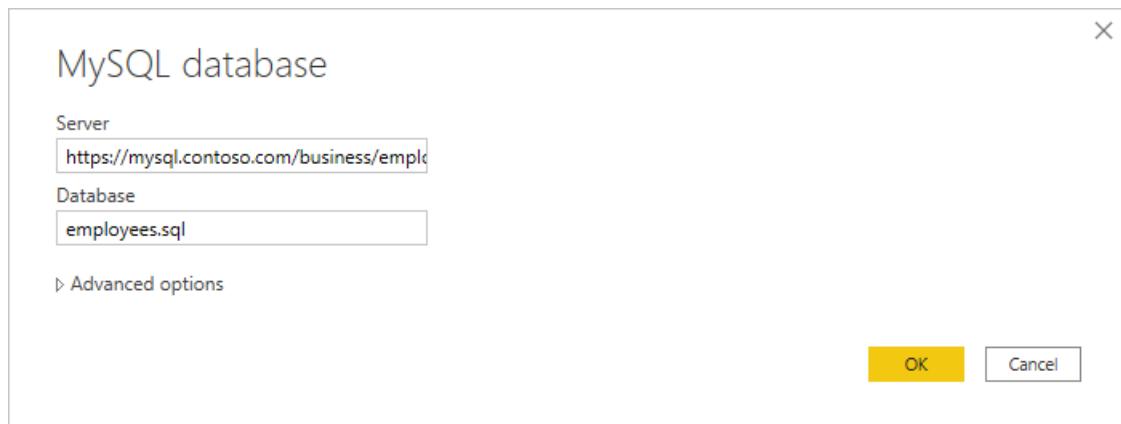
- Import
- Advanced options
 - Command timeout in minutes
 - Native SQL statement
 - Relationship columns
 - Navigate using full hierarchy

Connect to MySQL database from Power Query Desktop

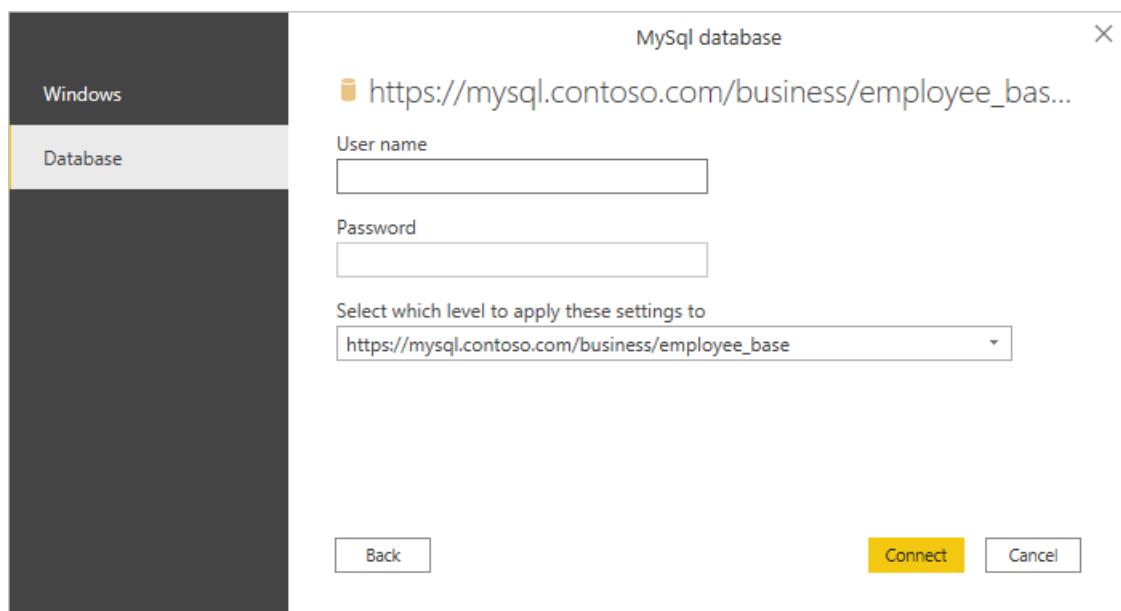
To make the connection, take the following steps:

1. Select the **MySQL database** option in the connector selection.

2. In the **MySQL database** dialog, provide the name of the server and database.



3. Select the **Database** authentication type and input your MySQL credentials in the **User name** and **Password** boxes.

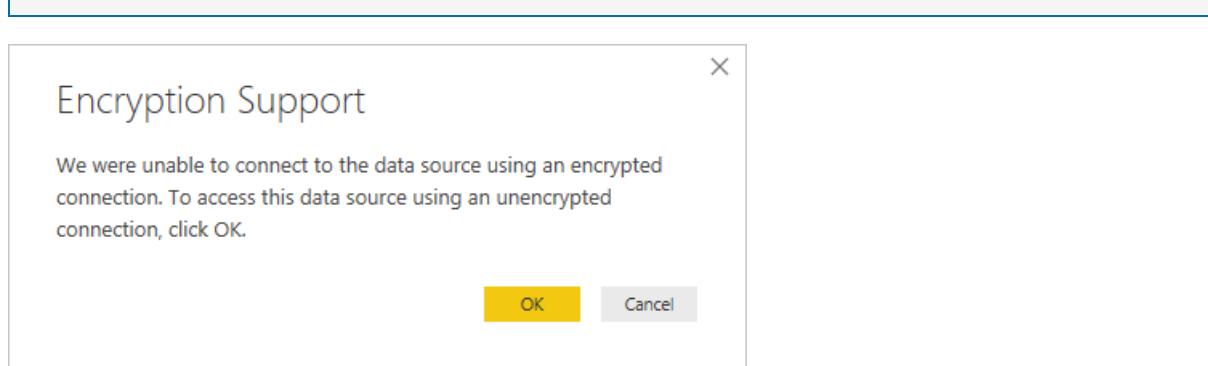


4. Select the level to apply your credentials to.

5. Once you're done, select **OK**.

NOTE

If the connection is not encrypted, you'll be prompted with the following dialog.



Select **OK** to connect to the database by using an unencrypted connection, or follow the [instructions](#) to set up encrypted connections to SQL Server.

6. In **Navigator**, select the data you require, then either load or transform the data.

The screenshot shows the 'Navigator' dialog in Power Query Online. On the left, a tree view lists tables from a MySQL database, with 'employees.departments' selected. On the right, a preview of the 'employees.departments' table is shown, displaying columns: dept_no, dept_name, employees.dept_emp, and employees.dept_manager. The table contains 9 rows of department data. At the bottom, there are buttons for 'Select Related Tables', 'Load' (highlighted in yellow), 'Transform Data', and 'Cancel'.

Connect to MySQL database from Power Query Online

To make the connection, take the following steps:

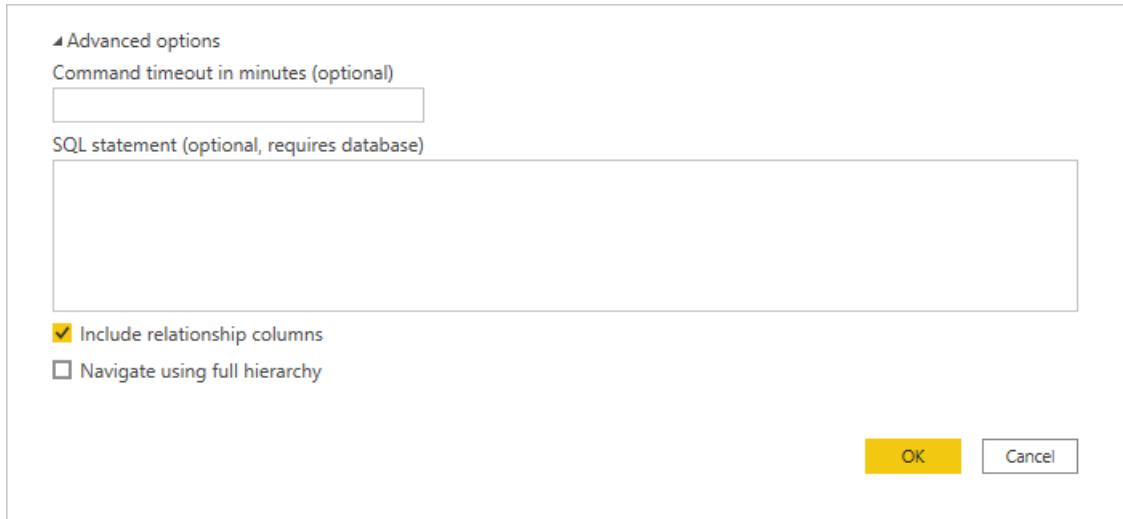
1. Select the **MySQL database** option in the connector selection.
2. In the **MySQL database** dialog, provide the name of the server and database.

The screenshot shows the 'MySQL database' connection settings dialog. It includes fields for 'Server' (set to 'https://mysql.contoso.com/business/employee_...'), 'Database' (set to 'employees.sql'), 'On-premises data gateway' (set to '[Admin] 10TestGateway'), 'Authentication kind' (set to 'Basic'), and two empty boxes for 'Username' and 'Password'. A checkbox for 'Use Encrypted Connection' is checked.

3. If necessary, include the name of your on-premises data gateway.
4. Select the **Basic** authentication kind and input your MySQL credentials in the **Username** and **Password** boxes.
5. If your connection isn't encrypted, clear **Use Encrypted Connection**.
6. Select **Next** to connect to the database.
7. In **Navigator**, select the data you require, then select **Transform data** to transform the data in Power Query Editor.

Connect using advanced options

Power Query Desktop provides a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop to connect to your MySQL database.

OData Feed

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Anonymous Windows Basic (requires Gateway) Web API Organizational Account
Function Reference Documentation	OData.Feed , ODataOmitValues.Nulls

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Capabilities supported

- Basic
- Advanced
 - URL parts
 - Open type columns
- Select related tables

NOTE

Microsoft Graph is not supported. More information: [Lack of Support for Microsoft Graph in Power Query](#)

Load data from an OData Feed in Power Query Desktop

To load data from an OData Feed in Power Query Desktop:

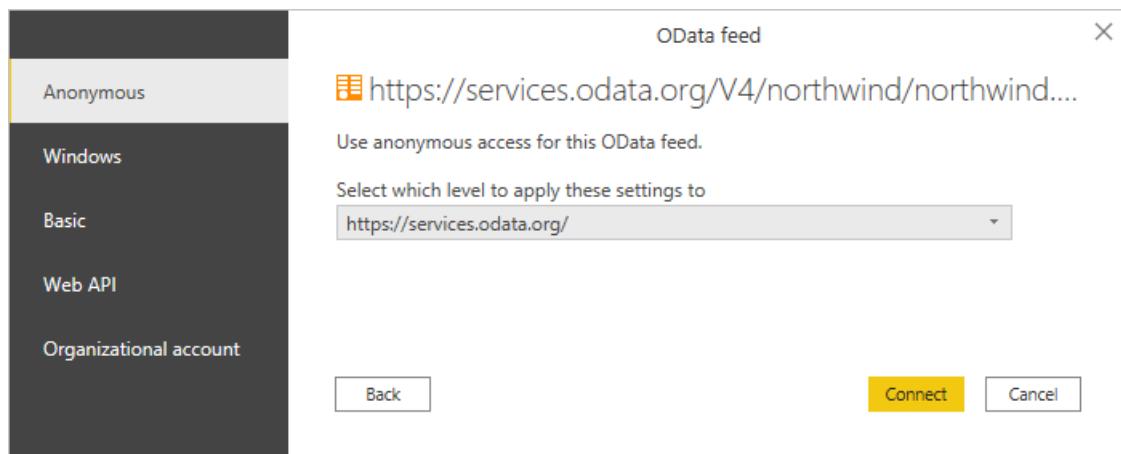
1. Select the **OData** or **OData Feed** option in the connector selection.
2. Choose the **Basic** button and enter a URL address in the text box. This URL should be the root of the OData service you want to connect to. For example, enter

<http://services.odata.org/V4/northwind/northwind.svc/>. Then select **OK**.

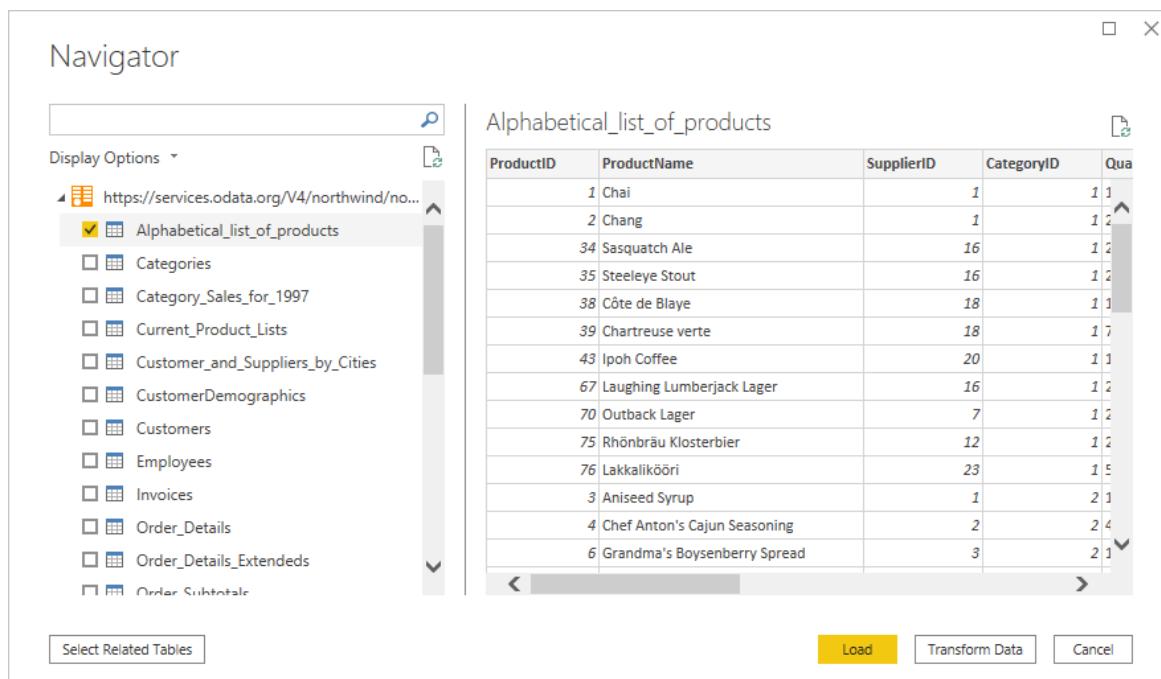


If the URL address you enter is invalid, a warning icon will appear next to the URL textbox.

3. If this is the first time you're connecting using the OData Feed, select the authentication type, input your credentials (if necessary), and select the level to apply the authentication settings to. Then select **Connect**.



4. From the **Navigator** dialog, you can select a table, then either transform the data in the Power Query Editor by selecting **Transform Data**, or load the data by selecting **Load**.

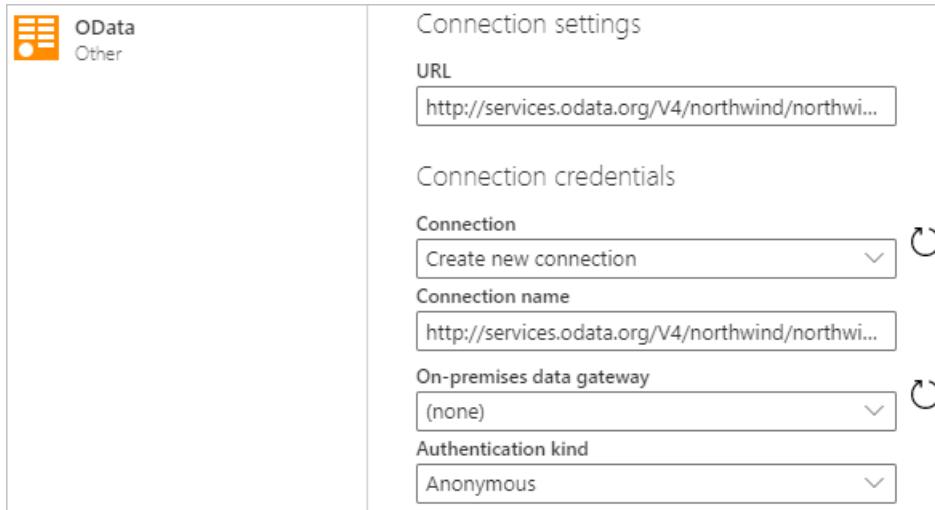


If you have multiple tables that have a direct relationship to one or more of the already selected tables, you can select the **Select Related Tables** button. When you do, all tables that have a direct relationship to one or more of the already selected tables will be imported as well.

Load data from an OData Feed in Power Query Online

To load data from an OData Feed in Power Query Online:

1. Select the **OData** or **OData Feed** option in the connector selection.
2. In the OData dialog that appears, enter a URL in the text box.



3. If this is the first time you're connecting using the OData Feed, select the authentication kind and enter your credentials (if necessary). Then select **Next**.
4. From the **Navigator** dialog, you can select a table, then transform the data in the Power Query Editor by selecting **Transform Data**.

The screenshot shows the 'Navigator' dialog. On the left, there's a tree view of tables under the URL 'https://services.odata.org/V4/northwind/no...'. The 'Alphabetical_list_of_products' table is selected and expanded, showing its schema and data. On the right, a preview grid displays the first 21 rows of the 'Alphabetical_list_of_products' table. At the bottom, there are buttons for 'Select Related Tables', 'Load' (highlighted in yellow), 'Transform Data', and 'Cancel'.

ProductID	ProductName	SupplierID	CategoryID	Qua
1	Chai	1	1 1	
2	Chang	1	1 2	
34	Sasquatch Ale	16	1 2	
35	Steeleye Stout	16	1 2	
38	Côte de Blaye	18	1 1	
39	Chartreuse verte	18	1 7	
43	Ipoh Coffee	20	1 1	
67	Laughing Lumberjack Lager	16	1 2	
70	Outback Lager	7	1 2	
75	Rhönbräu Klosterbier	12	1 2	
76	Lakkalikööri	23	1 5	
3	Aniseed Syrup	1	2 1	
4	Chef Anton's Cajun Seasoning	2	2 4	
6	Grandma's Boysenberry Spread	3	2 1	

If you have multiple tables that have a direct relationship to one or more of the already selected tables, you can select the **Select Related Tables** button. When you do, all tables that have a direct relationship to one or more of the already selected tables will be imported as well.

Connecting to Microsoft Graph

Connecting to [Microsoft Graph REST APIs](#) from Power Query isn't recommended or supported. See this [article](#) for more information.

Known Issues and Limitations

Joins

Due to the architecture of OData and other web connectors, joins can be non-performant. While you have the option to use navigation columns when merging between tables from an OData source, you don't have this option when merging with non-OData sources.

If you are seeing performance issues when merging an OData source, you should apply [Table.Buffer](#) to your OData query in the Advanced Editor, before you merge the data.

Test Connection issues

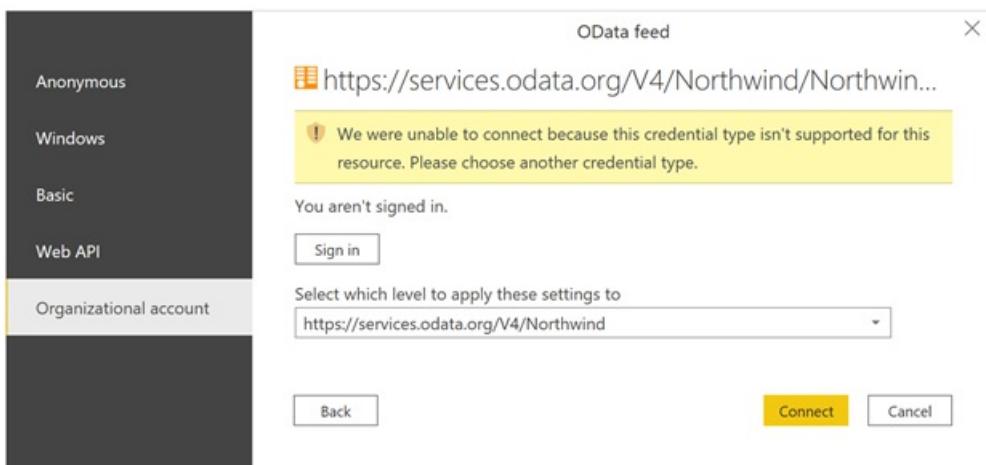
In cases where you're passing in a URL to the OData connector that's not just the service root, for example, if you have a filter on the URL, when you set up refresh in the service you should select **Skip Test Connection**.

Authenticating to arbitrary services

Some services support the ability for the OData connector to authenticate with OAuth/AAD authentication out of the box. However, this won't work in most cases.

When attempting to authenticate, if you see the following error:

"We were unable to connect because this credential type isn't supported for this resource. Please choose another credential type."



Contact the service owner. They'll either need to change the authentication configuration or build a custom connector.

Maximum URL length

If you're using the OData feed connector to connect to a SharePoint list, SharePoint online list, or Project Online, the maximum URL length for these connections is approximately 2100 characters. Exceeding the character limit results in an 401 error. This maximum URL length is built in the SharePoint front end and can't be changed.

To get around this limitation, start with the root OData endpoint and then navigate and filter inside Power Query. Power Query filters this URL locally when the URL is too long for SharePoint to handle. For example, start with:

```
OData.Feed("https://contoso.sharepoint.com/teams/sales/_api/ProjectData")
```

instead of

```
OData.Feed("https://contoso.sharepoint.com/teams/sales/_api/ProjectData/Projects?
select=_x0031_MetricName...etc...")
```

ODBC

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Database (Username/Password) Windows Default or Custom
Function Reference Documentation	Odbc.DataSource Odbc.Query

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

Before you get started, make sure you've properly configured the connection in the [Windows ODBC Data Source Administrator](#). The exact process here will depend on the driver.

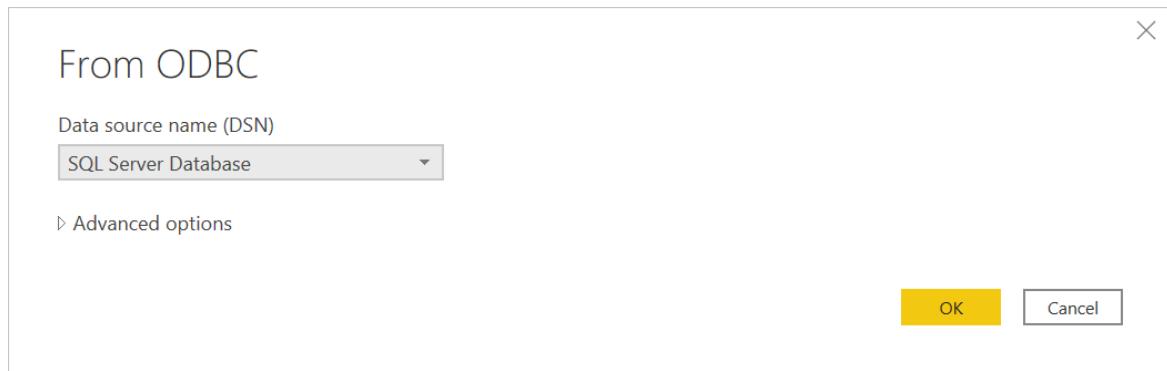
Capabilities Supported

- Import
- Advanced options
 - Connection string (non-credential properties)
 - SQL statement
 - Supported row reduction clauses

Connect to an ODBC data source from Power Query Desktop

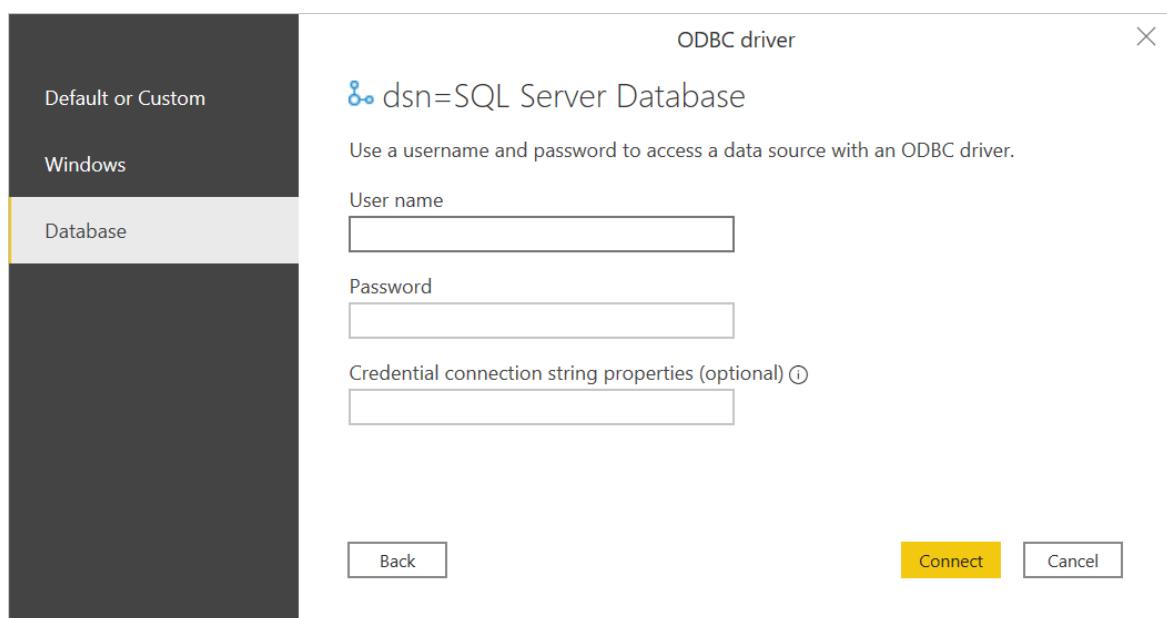
To make the connection, take the following steps:

1. Select the **ODBC** option in the **Get Data** selection.
2. In **From ODBC**, select the data source name (DSN) from the **Data source name (DSN)** drop-down box. In this example, a DSN name of **SQL Server Database** was selected.



You can also choose **Advanced options** to enter more optional connection information. More information: [Connect using advanced options](#)

3. Once you're done, select **OK**.
4. If this is the first time you are connecting to this database, select the authentication type and input your credentials when prompted.



The authentication types available are:

- **Default or Custom:** Select this authentication type when you don't specify any credentials if you're using DSN configured with a username and password. Or, if you need to include credentials as connection string properties.
- **Windows:** Select this authentication type if you want to connect using Windows authentication. Optionally, include any connection string properties you need.
- **Database:** Select this authentication type to use a username and password to access a data source with an ODBC driver. Optionally, include any connection string properties you need. This is the default selection.

More information: [Authentication with a data source](#)

5. Once you are done, select **Connect**.
6. In the **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

The screenshot shows the Power Query Navigator window. On the left, there's a tree view of data sources and tables. Under 'ODBC (dsn=SQL Server Database) [4]', 'vEmployee' is selected. On the right, the 'vEmployee' table is displayed in a grid format with columns: BusinessEntityID, Title, FirstName, MiddleName, LastName, and Suffix. The data includes rows for various employees like Gary, Brian, Alan, William, Sairaj, A. Scott, Chris, Karen, Sean, Tengiz, Zainal, Sean, and Mark. At the bottom, there are buttons for 'Select Related Tables', 'Load', 'Transform Data', and 'Cancel'.

Connect to an ODBC data source from Power Query Online

To make the connection, take the following steps:

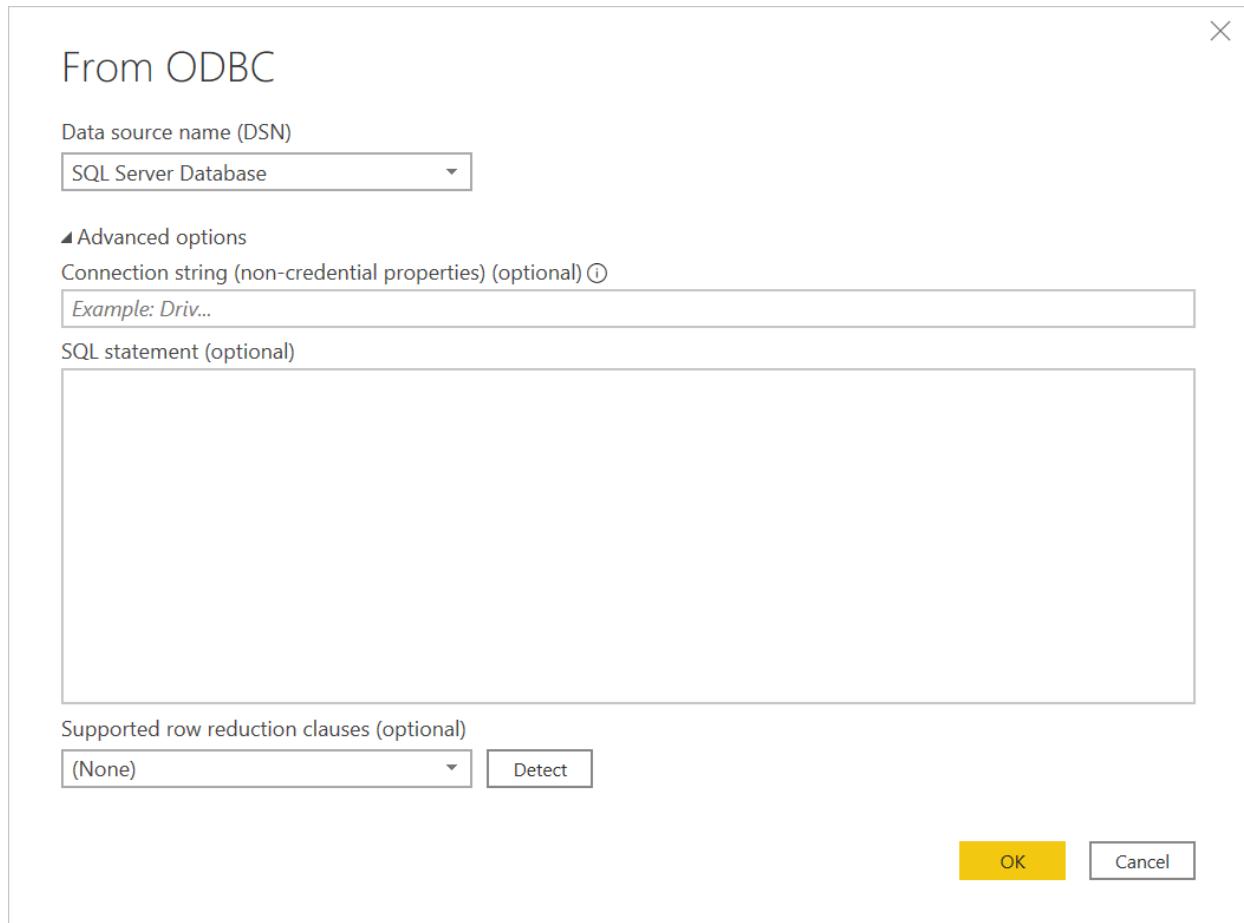
1. From the **Data sources** page, select **ODBC**.
2. In the ODBC page, enter your ODBC connection string. In this example, the connection string is `dsn=SQL Server Database`.

The screenshot shows the 'Connection settings' page for an ODBC connection. On the left, there are links for 'Odbc' and 'Other'. The main area has a section for 'ODBC connection string *' containing the value 'dsn=SQL Server Database'. Below it is a link to 'Advanced options'. The next section is 'Connection credentials' with fields for 'Data gateway *' (set to '[On-premises][Admin] TestGateway23'), 'Authentication kind' (set to 'Windows'), 'Username' (set to 'domain\alias'), and 'Password'. At the bottom is a checked checkbox for 'Use Encrypted Connection'.

3. If needed, select an on-premises data gateway in **Data gateway**.
4. Choose the authentication kind you'll use to sign in, and then enter your credentials.
5. Select **Next**.
6. In the **Navigator**, select the database information you want, and then select **Transform data** to continue transforming the data in Power Query Editor.

Connect using advanced options

Power Query provides a set of advanced options that you can add to your query if needed.



ADVANCED OPTION	DESCRIPTION
Connection string (non-credential properties)	<p>Provides an optional connection string that can be used instead of the Data source name (DSN) selection in Power BI Desktop. If Data source name (DSN) is set to (None), you can enter a connection string here instead. For example, the following connection strings are valid:</p> <p><code>dsn= <myDSN></code> or <code>driver= <myDriver>;port= <myPortNumber>;server= <myServer>;database= <myDatabase>;</code> The <code>{ }</code> characters can be used to escape special characters. Keys for connection strings will vary between different ODBC drivers. Consult your ODBC driver provider for more information about valid connection strings.</p>
SQL statement	<p>Provides a SQL statement, depending on the capabilities of the driver. Ask your vendor for more information, or go to Import data from a database using native database query.</p>
Supported row reduction clauses	<p>Enables folding support for <code>Table.FirstN</code>. Select Detect to find supported row reduction clauses, or select from one of the drop down options (TOP, LIMIT and OFFSET, LIMIT, or ANSI SQL-compatible). This option is not applicable when using a native SQL statement. Only available in Power Query Desktop.</p>

Oracle database

1/15/2022 • 5 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Windows (desktop/online) Database (desktop) Basic (online)
Function Reference Documentation	Oracle.Database

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

Supported Oracle versions:

- Oracle Server 9 and later
- Oracle Data Access Client (ODAC) software 11.2 and later

Before you can connect to an Oracle database using Power Query, you need to install the Oracle client software v8.1.7 or greater on your computer. To install the 32-bit Oracle client software, go to [32-bit Oracle Data Access Components \(ODAC\) with Oracle Developer Tools for Visual Studio \(12.1.0.2.4\)](#). To install the 64-bit Oracle client, go to [64-bit ODAC 12c Release 4 \(12.1.0.2.4\) Xcopy for Windows x64](#).

NOTE

Choose a version of Oracle Data Access Client (ODAC) that's compatible with your Oracle Server. For instance, ODAC 12.x doesn't always support Oracle Server version 9. Choose the Windows installer of the Oracle Client. During the setup of the Oracle client, make sure you enable *Configure ODP.NET and/or Oracle Providers for ASP.NET* at machine-wide level by selecting the corresponding checkbox during the setup wizard. Some versions of the Oracle client wizard selects the checkbox by default, others don't. Make sure that checkbox is selected so that Power Query can connect to your Oracle database.

To connect to an Oracle database with the [on-premises data gateway](#), the correct Oracle client software must be

installed on the computer running the gateway. The Oracle client software you use depends on the Oracle server version, but will always match the 64-bit gateway. For more information, go to [Manage your data source - Oracle](#).

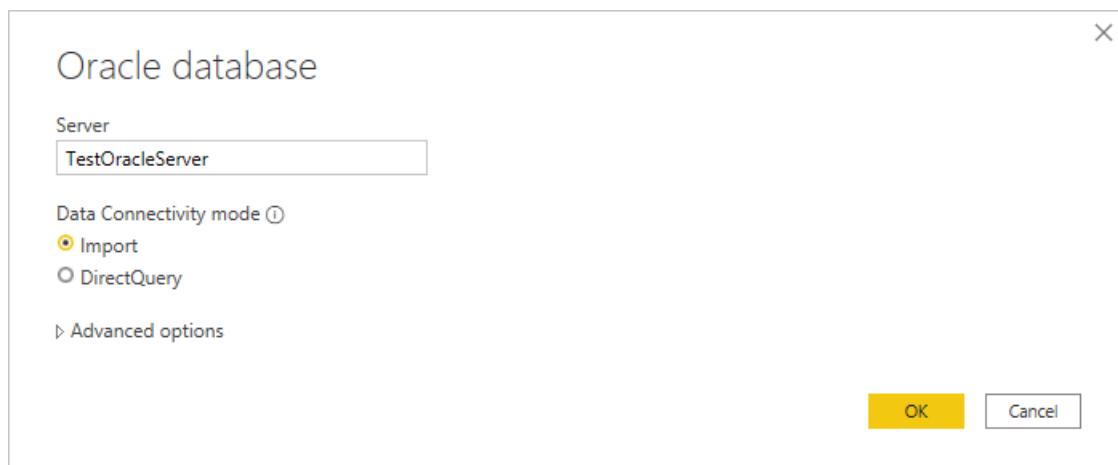
Capabilities Supported

- Import
- DirectQuery
- Advanced options
 - Command timeout in minutes
 - SQL statement
 - Include relationship columns
 - Navigate using full hierarchy

Connect to an Oracle database from Power Query Desktop

To make the connection, take the following steps:

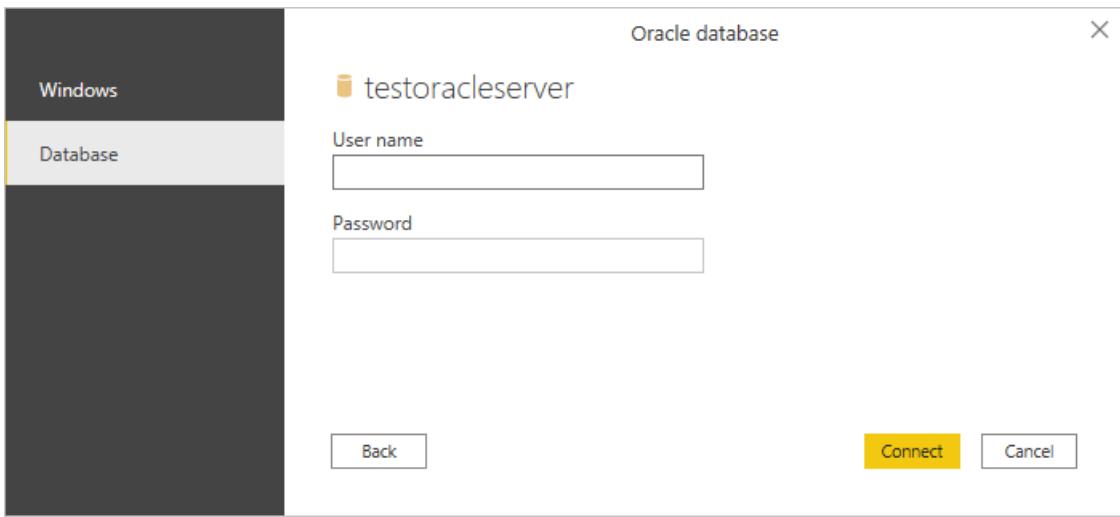
1. Select the **Oracle database** option in the connector selection.
2. Specify the Oracle Server to connect to in **Server**. If a SID is required, specify it by using the format *ServerName/SID*, where *SID* is the unique name of the database. If the *ServerName/SID* format doesn't work, use *ServerName/ServiceName*, where *ServiceName* is the alias you use to connect.



NOTE

If you are using a local database, or autonomous database connections, you may need to place the server name in quotation marks to avoid connection errors.

3. If you're connecting from Power BI Desktop, select either the **Import** or **DirectQuery** data connectivity mode. The rest of these example steps use the Import data connectivity mode. To learn more about DirectQuery, go to [Use DirectQuery in Power BI Desktop](#).
4. If this is the first time you're connecting to this Oracle database, select the authentication type you want to use, and then enter your credentials. For more information about authentication, go to [Authentication with a data source](#).

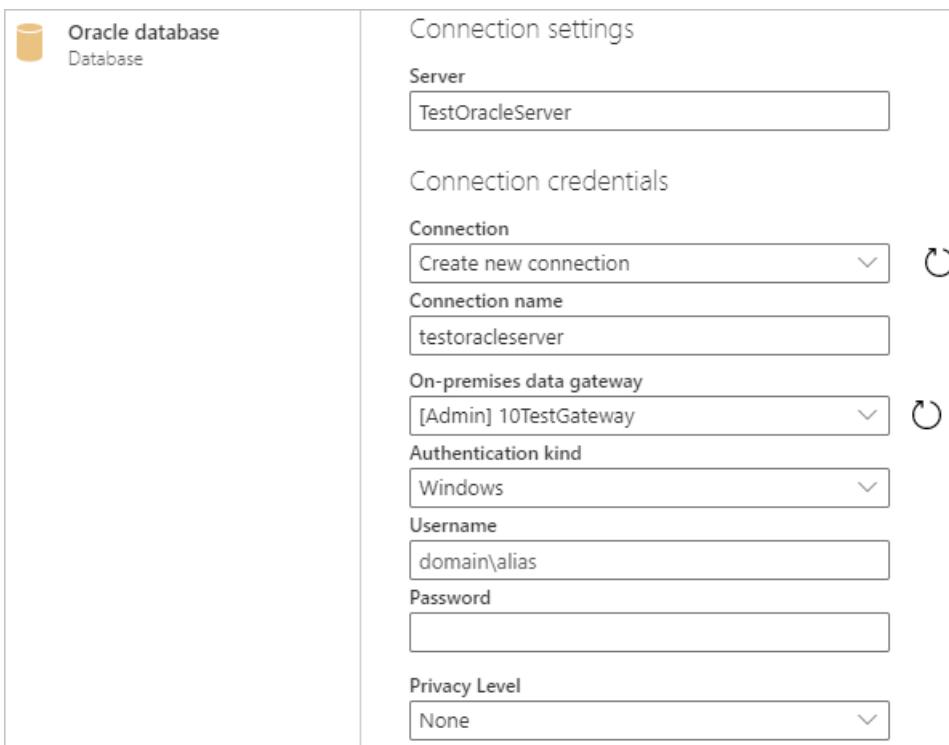


5. In **Navigator**, select the data you require, then either select **Load** to load the data or **Transform Data** to transform the data.

Connect to an Oracle database from Power Query Online

To make the connection, take the following steps:

1. Select the **Oracle database** option in the data sources selection.
2. In the **Oracle database** dialog that appears, specify the Oracle Server to connect to in **Server**. If a SID is required, specify it by using the format *ServerName/SID*, where *SID* is the unique name of the database. If the *ServerName/SID* format doesn't work, use *ServerName/ServiceName*, where *ServiceName* is the alias you use to connect.



3. Select the name of your on-premises data gateway.

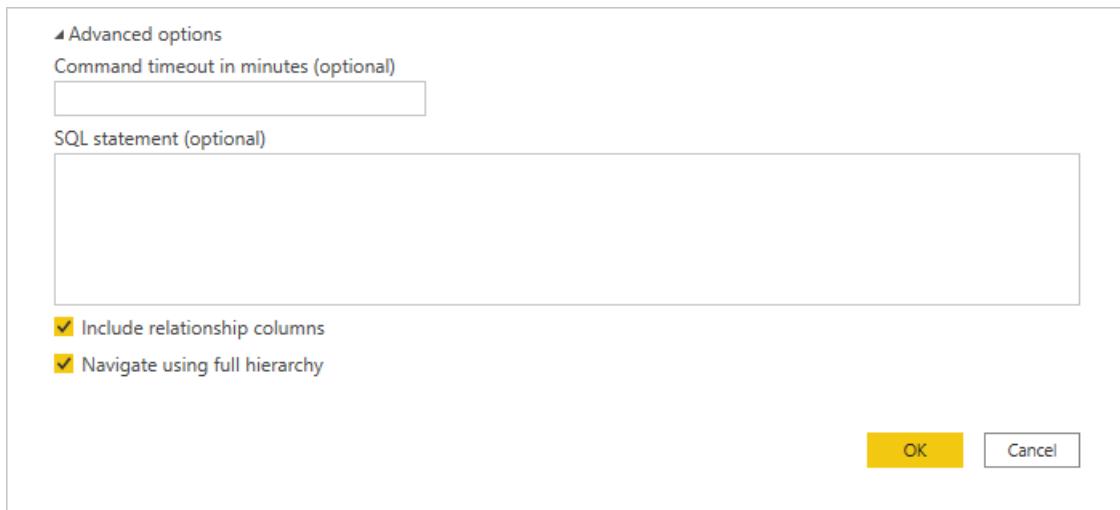
NOTE

You must select an on-premises data gateway for this connector, whether the Oracle database is on your local network or on a web site.

4. If this is the first time you're connecting to this Oracle database, select the type of credentials for the connection in **Authentication kind**. Choose **Basic** if you plan to use an account that's created within Oracle instead of Windows authentication.
5. Enter your credentials.
6. Select **Next** to continue.
7. In **Navigator**, select the data you require, then select **Transform data** to transform the data in Power Query Editor.

Connect using advanced options

Power Query Desktop provides a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop to connect to your Oracle database.

Troubleshooting

You might come across any of several errors from Oracle when the naming syntax is either incorrect or not

configured properly:

- ORA-12154: TNS: could not resolve the connect identifier specified.
- ORA-12514: TNS: listener does not currently know of service requested in connect descriptor.
- ORA-12541: TNS: no listener.
- ORA-12170: TNS: connect timeout occurred.
- ORA-12504: TNS: listener was not given the SERVICE_NAME in CONNECT_DATA.

These errors might occur if the Oracle client either isn't installed or isn't configured properly. If it's installed, verify the tnsnames.ora file is properly configured and you're using the proper net_service_name. You also need to make sure the net_service_name is the same between the machine that uses Power BI Desktop and the machine that runs the gateway. For more information, see [Prerequisites](#).

You might also come across a compatibility issue between the Oracle server version and the Oracle Data Access Client version. Typically, you want these versions to match, as some combinations are incompatible. For instance, ODAC 12.x doesn't support Oracle Server version 9.

If you downloaded Power BI Desktop from the Microsoft Store, you might be unable to connect to Oracle databases because of an Oracle driver issue. If you come across this issue, the error message returned is: *Object reference not set*. To address the issue, do one of these steps:

- Download Power BI Desktop from the Download Center instead of Microsoft Store.
- If you want to use the version from Microsoft Store: on your local computer, copy oraons.dll from `12.X.X\client_X` to `12.X.X\client_X\bin`, where X represents version and directory numbers.

If you see the error message, *Object reference not set*, in the Power BI Gateway when you connect to an Oracle database, follow the instructions in [Manage your data source - Oracle](#).

If you're using Power BI Report Server, consult the guidance in the [Oracle Connection Type](#) article.

Next steps

[Optimize Power Query when expanding table columns](#)

PDF

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous (online) Basic (online) Organizational account (online) Windows (online)
Function Reference Documentation	Pdf.Tables

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

NOTE

PDF is not supported in Power BI Premium.

Prerequisites

None.

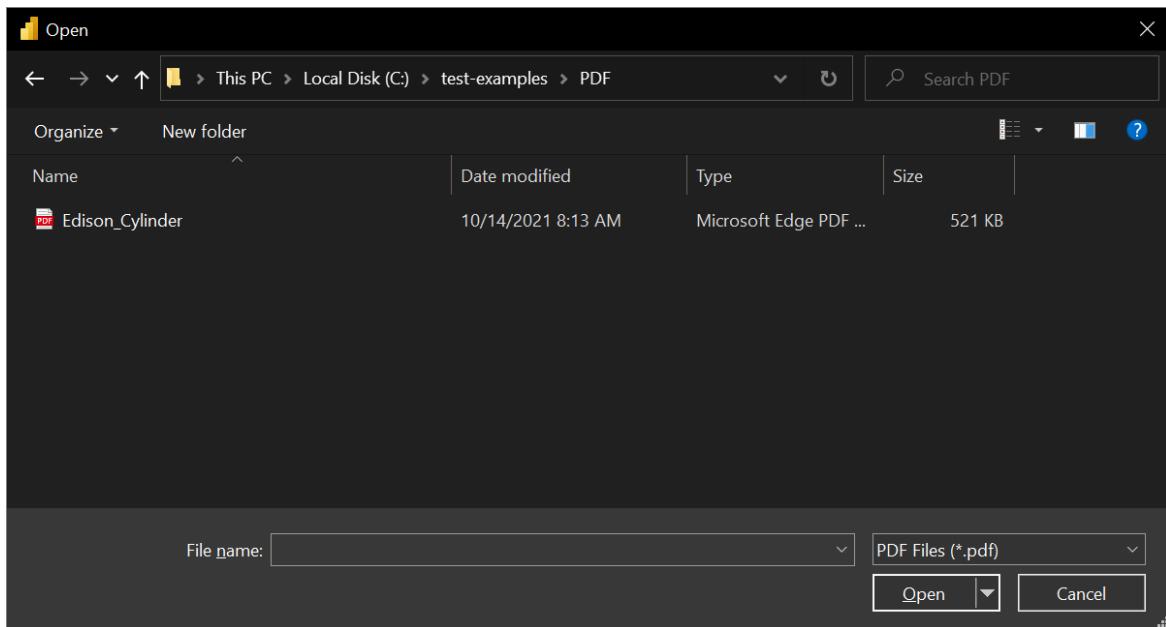
Capabilities Supported

- Import

Connect to a PDF file from Power Query Desktop

To make the connection from Power Query Desktop:

- Select the **PDF** option in the connector selection.
- Browse for and select the PDF file you want to load. Then select **Open**.



If the PDF file is online, use the [Web connector](#) to connect to the file.

3. In **Navigator**, select the file information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

Label No.	Title	Performed by:
3	Vocal Sextette From Lucia	
126	Cello Solo, Nina	Kronold
179	Are You Coming Home Tonight	Anthony & Harrison
217	Garden Melody, Violin	A. Spalding
232	Dream Of The Tyrolienne	Venetian Trio
265	Superba Lancers - 1st And 2nd Figures	Band
266	Superba Lancers - 3rd And 4th Figures	Band
268	Petunia QuadRille - 1st And 2nd Figures	Band
274	Dublin Daisies Two-Step	Band
275	Lucky Moon Three-Step	Band
303	My Old Kentucky Home	Knickerbocker Quartet
321	He Was A Wonderful Man	Jones & Murray
363	Medley Of Emmett's Yodel Songs	Watson
395	That Mesmerizing Mendelssohn Tune	Collins & Harlan
422	Down In Turkey Hollow	Golden & Hughes

Connect to a PDF file from Power Query Online

To make the connection from Power Query Online:

1. Select the **PDF** option in the connector selection.
2. In the PDF dialog box that appears, either provide the file path or the URL to the location of the PDF file. If you're loading a local file, you can also select **Upload file (Preview)** to browse to the local file or drag and drop the file.

 PDF
File

Connection settings

Link to file Upload file (Preview) (i)

File path or URL *

Browse OneDrive...

Connection credentials

Data gateway *

▼
↻

Authentication kind

▼

Username

Password

3. If necessary, select an on-premises data gateway to access the PDF file.
4. If this is the first time you've accessed this PDF file, select the authentication kind and sign in to your account (if needed).
5. In **Navigator**, select the file information you want, and then select **Transform Data** to continue transforming the data in Power Query Editor.

Power Query - Choose data X

Search

Display options
↻

Table001 (Page 1)

#	Label No.	Title	Performed by:	Duration	Type	Take/Mold	Condition
3		Vocal Sextette From Lucia	Kronold	Four minute	Black Wax	0.39	Good
126		Cello Solo, Nina		Four minute	Black Wax	0.7	Good
179		Are You Coming Home Tonight	Anthony & Harrison	Four minute	Black Wax	.9	Good
217		Garden Melody, Violin	A. Spalding	Four minute	Black Wax	0.17	Poor
232		Dream Of The Tyrolienne	Venetian Trio	Four minute	Black Wax	0.11	Good
265		Superba Lancers - 1st And 2nd Figures	Band	Four minute	Black Wax	0.8	Good
266		Superba Lancers - 3rd And 4th Figures	Band	Four minute	Black Wax	0.7	Good
268		Petunia Quadrille - 1st And 2nd Figures	Band	Four minute	Black Wax	.10	Good
274		Dublin Daisies Two-Step	Band	Four minute	Black Wax	0.8	Good
275		Lucky Moon Three-Step	Band	Four minute	Black Wax	0.11	Poor
303		My Old Kentucky Home	Knickerbocker Quartet	Four minute	Black Wax	10	Good
321		He Was A Wonderful Man	Jones & Murray	Four minute	Black Wax	.12	Good
363		Medley Of Emmett's Yodle Songs	Watson	Four minute	Black Wax	.11	Good
395		That Mesmerizing Mendelssohn Tune	Collins & Harlan	Four minute	Black Wax	.9	Fair
422		Down In Turkey Hollow	Golden & Hughes	Four minute	Black Wax	.11	Good
553		Buck Dance Medley (Accorion)	Kimmble	Four minute	Black Wax	.9	Good

Back
Cancel
Transform data

PostgreSQL

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Database (Username/Password)
Function Reference Documentation	PostgreSQL.Database

NOTE

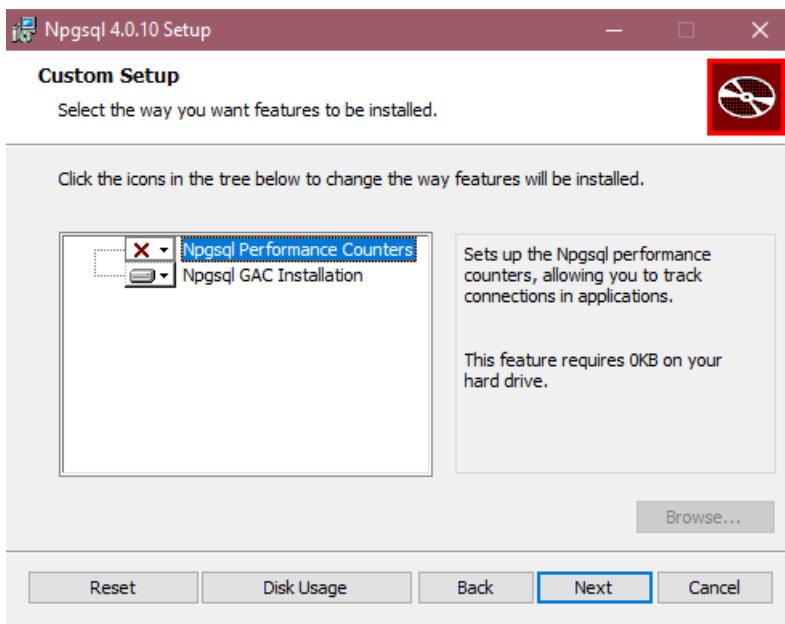
Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

As of the December 2019 release, Npgsql 4.0.10 shipped with Power BI Desktop and no additional installation is required. GAC Installation overrides the version provided with Power BI Desktop, which will be the default. Refreshing is supported both through the cloud in the Power BI Service and also on premise through the Gateway. In the Power BI service, Npgsql 4.0.10 will be used, while on premise refresh will use the local installation of Npgsql, if available, and otherwise use Npgsql 4.0.10.

For Power BI Desktop versions released before December 2019, you must install the Npgsql provider on your local machine. To install the Npgsql provider, go to the [releases page](#) and download the relevant release. The provider architecture (32-bit or 64-bit) needs to match the architecture of the product where you intend to use the connector. When installing, make sure that you select Npgsql GAC Installation to ensure Npgsql itself is added to your machine.

We recommend Npgsql 4.0.10. Npgsql 4.1 and up won't work due to .NET version incompatibilities.



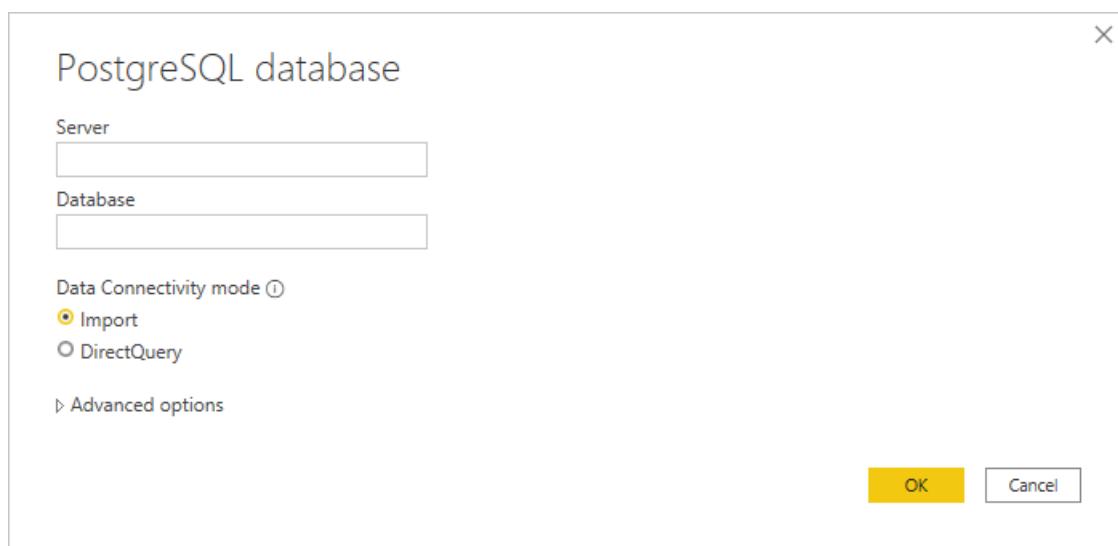
Capabilities Supported

- Import
- DirectQuery (Power BI only)
- Advanced options
 - Command timeout in minutes
 - Native SQL statement
 - Relationship columns
 - Navigate using full hierarchy

Connect to a PostgreSQL database from Power Query Desktop

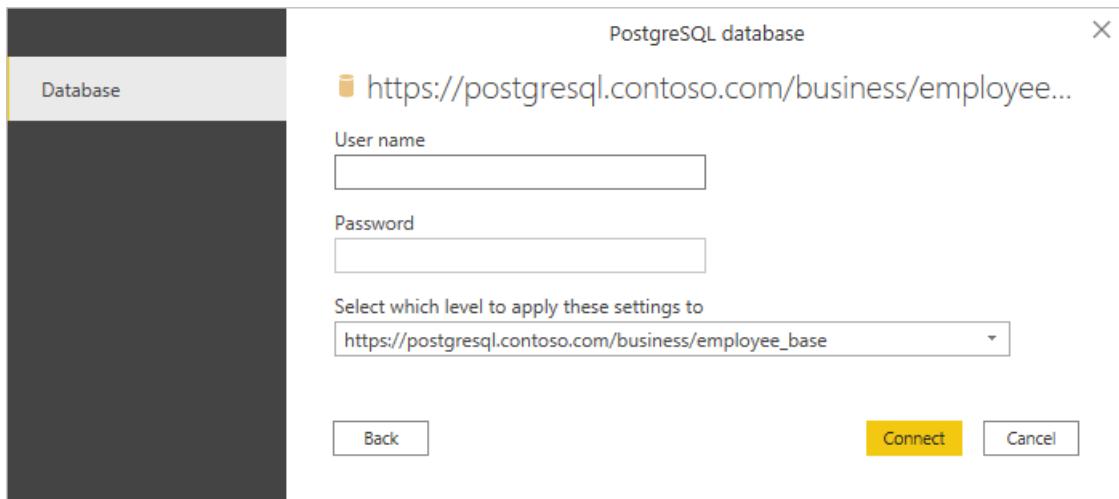
Once the matching Npgsql provider is installed, you can connect to a PostgreSQL database. To make the connection, take the following steps:

1. Select the **PostgreSQL database** option in the connector selection.
2. In the **PostgreSQL database** dialog that appears, provide the name of the server and database.



3. Select either the **Import** or **DirectQuery** data connectivity mode.
4. If this is the first time you're connecting to this database, input your PostgreSQL credentials in the **User**

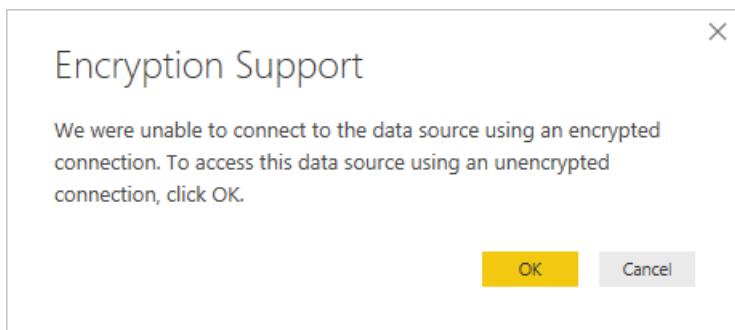
name and Password boxes of the Database authentication type. Select the level to apply the authentication settings to. Then select **Connect**.



For more information about using authentication methods, go to [Authentication with a data source](#).

NOTE

If the connection is not encrypted, you'll be prompted with the following message.



Select **OK** to connect to the database by using an unencrypted connection, or follow the instructions in [Enable encrypted connections to the Database Engine](#) to set up encrypted connections to PostgreSQL database.

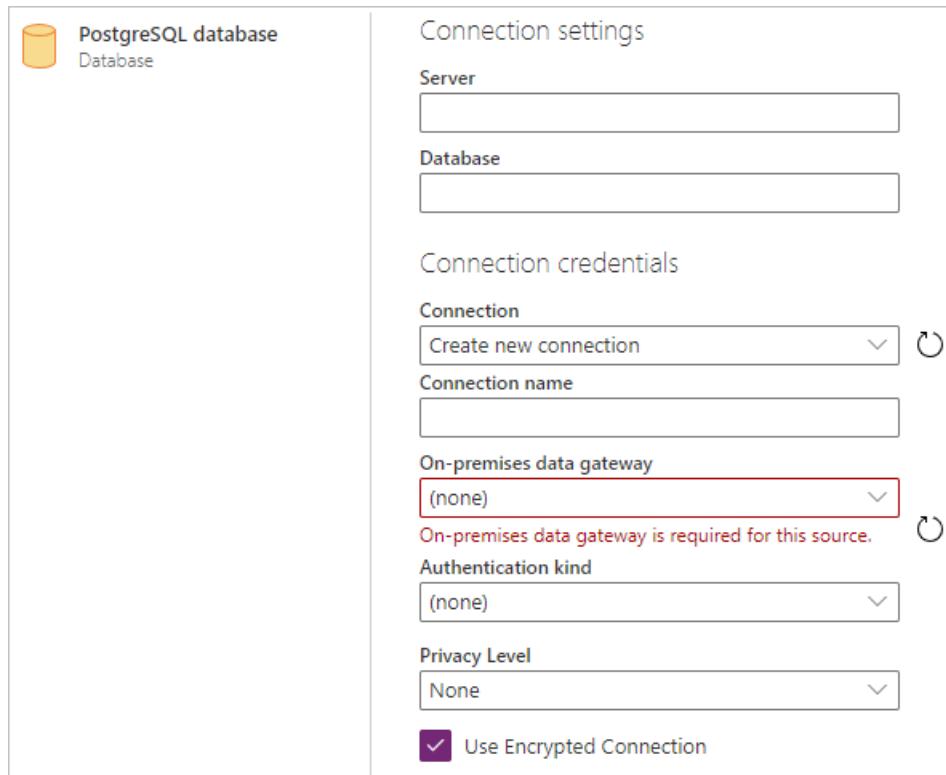
5. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

A screenshot of the Power BI Navigator interface. On the left, there's a tree view of available tables: Sales.vStoreWithContacts, Sales.vStoreWithDemographics, AWBuildVersion, DatabaseLog, ErrorLog, HumanResources.Department, HumanResources.Employee (selected), HumanResources.EmployeeDepartmentHistory, HumanResources.EmployeePayHistory, HumanResources.JobCandidate, HumanResources.Shift, Person.Address, and Person.AddressType. The main area shows the "HumanResources.Employee" table with columns: BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, and JobTitle. The table has 16 rows of data. At the bottom, there are buttons for "Select Related Tables", "Load" (highlighted in yellow), "Transform Data", and "Cancel".

Connect to a PostgreSQL database from Power Query Online

To make the connection, take the following steps:

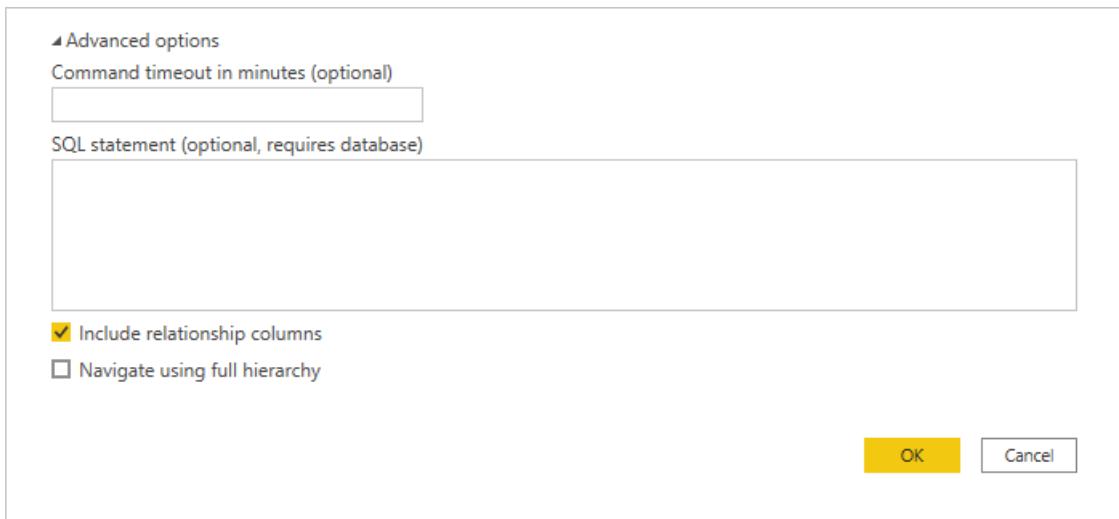
1. Select the **PostgreSQL database** option in the connector selection.
2. In the **PostgreSQL database** dialog that appears, provide the name of the server and database.



3. Select the name of the on-premises data gateway you want to use.
4. Select the **Basic** authentication kind and input your MySQL credentials in the **Username** and **Password** boxes.
5. If your connection isn't encrypted, clear **Use Encrypted Connection**.
6. Select **Next** to connect to the database.
7. In **Navigator**, select the data you require, then select **Transform data** to transform the data in Power Query Editor.

Connect using advanced options

Power Query Desktop provides a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, the navigator displays only the tables whose columns and rows contain data.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop to connect to your PostgreSQL database.

Native query folding

By default, native query folding is enabled. Operations that are capable of folding will be applied on top of your native query according to normal Import or Direct Query logic. Native Query folding isn't applicable with optional parameters present in [Value.NativeQuery\(\)](#).

In the rare case that folding doesn't work with native query folding enabled, you can disable it. To disable native query folding, set the `EnableFolding` flag to `false` for [Value.NativeQuery\(\)](#) in the advanced editor.

Sample: `Value.NativeQuery(target as any, query, null, [EnableFolding=false])`

Troubleshooting

Your native query may throw the following error:

We cannot fold on top of this native query. Please modify the native query or remove the 'EnableFolding' option.

A basic trouble shooting step is to check if the query in [Value.NativeQuery\(\)](#) throws the same error with a `limit 1` clause around it:

```
select * from (query) _ limit 1
```

QuickBooks Online (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

The Power BI QuickBooks Online connector enables connecting to your QuickBooks Online account and viewing, analyzing, and reporting on your company QuickBooks data in Power BI.

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI (Datasets)
Authentication Types Supported	QuickBooks Online account

WARNING

QuickBooks Online has deprecated support for Internet Explorer 11, which Power Query Desktop uses for authentication to online services. To be able to log in to Quickbooks Online from Power BI Desktop, go to [Enabling Microsoft Edge \(Chromium\) for OAuth Authentication in Power BI Desktop](#).

Prerequisites

To use the QuickBooks Online connector, you must have a QuickBooks Online account username and password.

The QuickBooks Online connector uses the QuickBooks ODBC driver. The QuickBooks ODBC driver is shipped with Power BI Desktop and no additional installation is required.

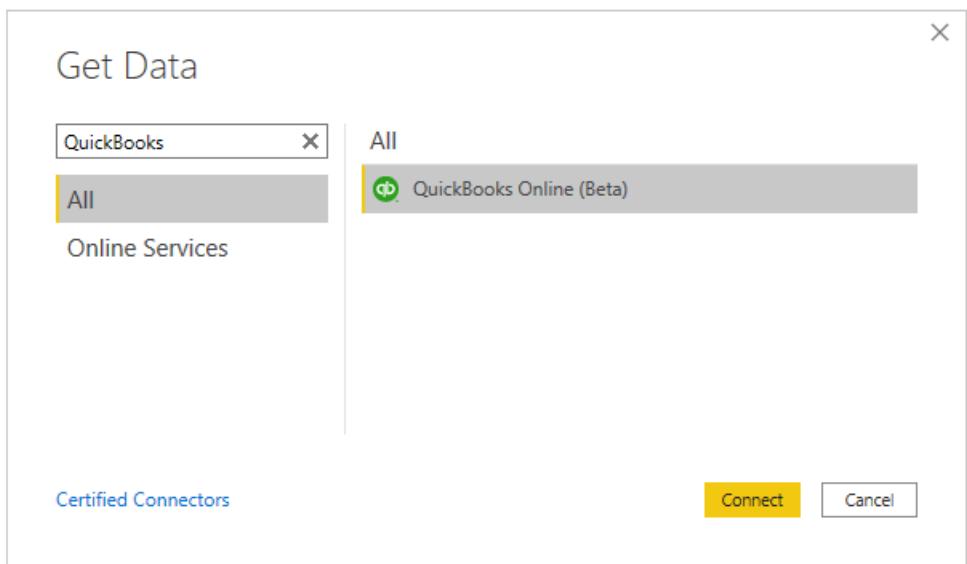
Capabilities Supported

- Import

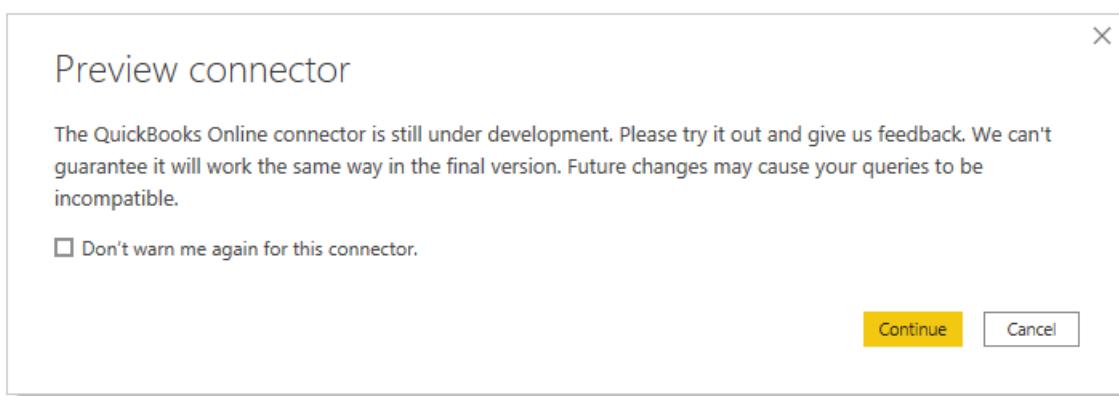
Connect to QuickBooks Online

To connect to QuickBooks Online:

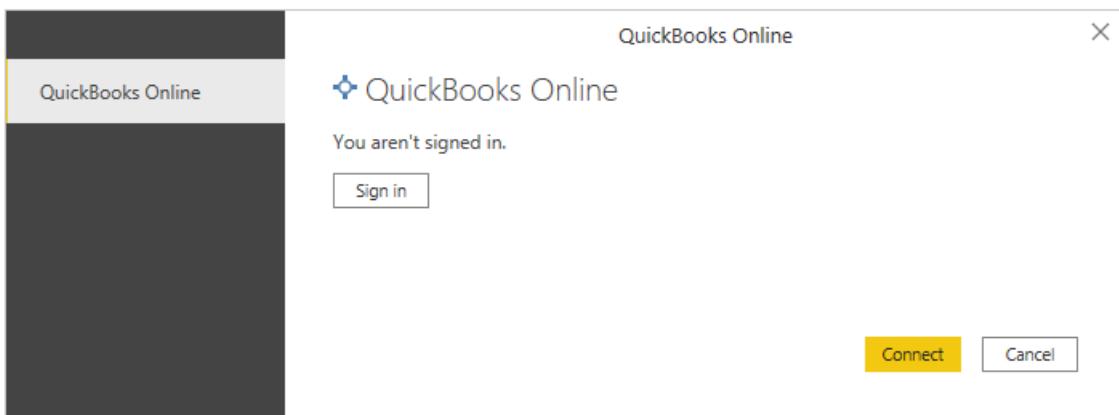
1. In the **Get Data** dialog box, enter **QuickBooks** in the **Search** box, select **QuickBooks Online (Beta)** from the product-specific data connector list, and then select **Connect**.



2. Select **Continue** in the Preview connector message.



3. Select **Sign in** to sign into your QuickBooks Online account.



4. In the following dialog, enter your QuickBooks credentials. You may be required to provide 2FA (two factor authentication code) as well.

about:blank

Intuit

✓ turbotax qb quickbooks mint

Sign In

One account for everything Intuit, including QuickBooks. [Learn more](#)

 Sign in with Google

or

Email or user ID

Password

×

Please enter password.

Remember me

 **Sign In**

By clicking **Sign In**, you agree to our [Terms](#) and have read and acknowledge our [US Privacy Statement](#).
Updated on January 1, 2020

[I forgot my user ID or password](#)
[New to Intuit? Create an account.](#)

Invisible reCAPTCHA by Google [Privacy Policy](#) and [Terms of Use](#).

Legal Privacy Security
© 2020 Intuit, Inc. All rights reserved. Intuit, QuickBooks, QB, TurboTax, ProConnect and Mint are registered trademarks of Intuit Inc.
Terms and conditions, features, support, pricing, and service options subject to change without notice.

5. In the following dialog, select a company and then select **Next**.

<https://appcenter.intuit.com/app/connect/oauth2>

Welcome [REDACTED] (Not You?)

Let's get you set up to use Microsoft Power BI

Search for one of your companies below to use with the app.

Search for a company

Search for a company ▾

No, thanks

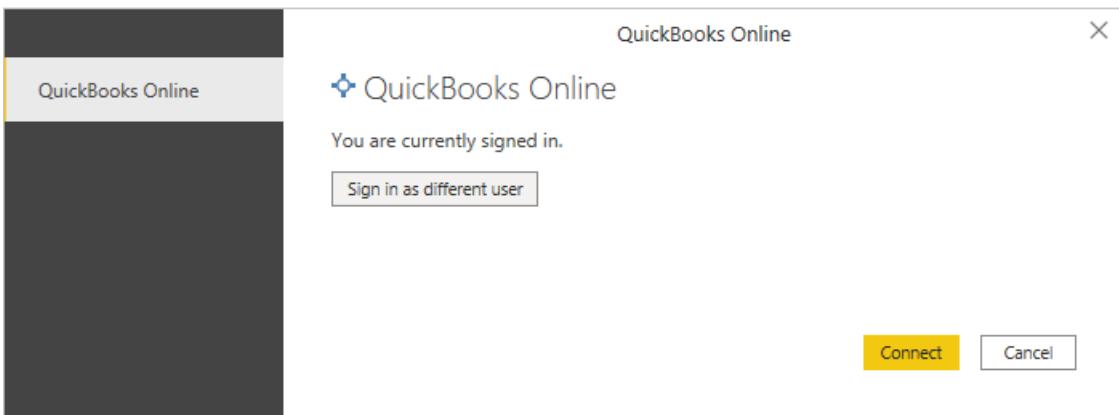
Next

intuit

©2020 Intuit Inc. All rights reserved.

turbotax quickbooks proconnect mint

- Once you've successfully signed in, select **Connect**.



- In the **Navigator** dialog box, select the QuickBooks tables you want to load. You can then either load or transform the data.

The screenshot shows the Microsoft Power BI Navigator window. On the left, there's a tree view of data sources and tables. Under 'QuickBooks Online [113]', 'Account' is selected and highlighted with a yellow checkmark. Other visible nodes include 'Attachable', 'Attachable_AttachableRef', 'Attachable_AttachableRef_CustomField', 'Bill', 'Bill_Account_Based_Expense_Line', 'Bill_Item_Based_Expense_Line', 'Bill_LinkedTxn', 'Bill_Payment', 'Bill_TxnTaxDetail_TaxLine', 'BillPayment_Line', 'BillPayment_Line_LinkedTxn', 'Budget', 'Budget_Detail', 'Class', 'Company_Currency', 'Company_Info', 'CompanyCurrency_CustomField', 'CompanyInfo_NameValue', 'Credit_Memo', 'CreditMemo_CustomField', 'CreditMemo_Description_Line', 'CreditMemo_Discount_Line', 'CreditMemo_Group_Individual_Item_Li...', 'CreditMemo_Group_Item_Line', 'CreditMemo_Sales_Item_Line', 'CreditMemo_Subtotal_Line', 'CreditMemo_TxnTaxDetail_TaxLine', and 'Customer'. The main pane on the right is titled 'Account' and contains a table with columns: Id, SyncToken, Create_Time, Last_Updated_Time, Name, and SubAccou. The table has 30 rows of data. A note at the bottom says: 'The data in the preview has been truncated due to size limits.' At the bottom right are buttons for Load, Transform Data, and Cancel.

ID	SyncToken	Create_Time	Last_Updated_Time	Name	SubAccou
1	0	2020-01-01 2:42:05 PM	2020-01-01 2:42:05 PM	Services	
10	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Dues & Subscriptions	
11	0	2020-01-01 2:42:07 PM	2020-01-06 10:17:24 AM	Insurance	
12	0	2020-01-01 2:42:07 PM	2020-01-06 10:26:24 AM	Legal & Professional Fees	
13	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Meals and Entertainment	
14	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Miscellaneous	
15	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Office Expenses	
16	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Promotional	
17	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Rent or Lease	
19	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Stationery & Printing	
2	0	2020-01-01 2:42:05 PM	2020-01-01 2:42:05 PM	Retained Earnings	
20	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Supplies	
21	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Taxes & Licenses	
22	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Travel	
23	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Travel Meals	
24	0	2020-01-01 2:42:07 PM	2020-01-06 10:30:53 AM	Utilities	
25	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Interest Earned	
26	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Other Portfolio Income	
27	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Penalties & Settlements	
28	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Disposal Fees	
29	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Equipment Rental	
3	0	2020-01-01 2:42:07 PM	2020-01-01 2:42:07 PM	Prepaid Expenses	

The data in the preview has been truncated due to size limits.

Load Transform Data Cancel

Known issues

Beginning on August 1, 2020, Intuit will no longer support Microsoft Internet Explorer 11 (IE 11) for QuickBooks Online. When you use OAuth2 for authorizing QuickBooks Online, after August 1, 2020, only the following browsers will be supported:

- Microsoft Edge
- Mozilla Firefox
- Google Chrome
- Safari 11 or newer (Mac only)

For more information, see [Alert: Support for IE11 deprecating on July 31, 2020 for Authorization screens](#).

For information about current Microsoft Edge support in Power BI Desktop, go to [Enabling Microsoft Edge \(Chromium\) for OAuth Authentication in Power BI Desktop](#).

Salesforce Objects

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Salesforce account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

WARNING

By default, Salesforce does not support Internet Explorer 11, which is used as part of the authentication experience to online services in Power Query Desktop. Please opt-in for [extended support for accessing Lightning Experience Using Microsoft Internet Explorer 11](#). You may also want to review Salesforce documentation on [configuring Internet Explorer](#). At this time, users will be impaired from authenticating, but stored credentials should continue to work until their existing authentication tokens expire. To resolve this, go to [Enabling Microsoft Edge \(Chromium\) for OAuth Authentication in Power BI Desktop](#).

Prerequisites

To use the Salesforce Objects connector, you must have a Salesforce account username and password.

Also, Salesforce API access should be enabled. To verify access settings, go to your personal Salesforce page, open your profile settings, and search for and make sure the **API Enabled** checkbox is selected. Note that Salesforce trial accounts don't have API access.

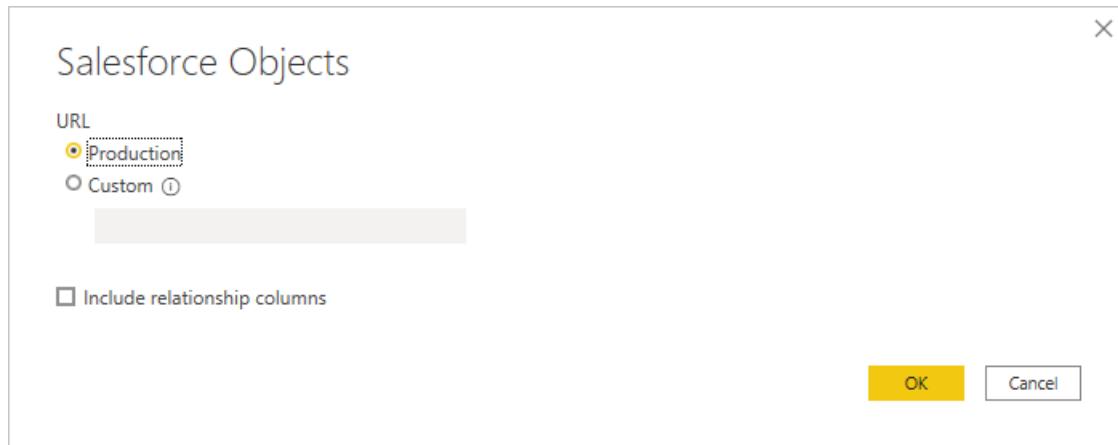
Capabilities Supported

- Production
- Custom
 - Custom domains
 - CNAME record redirects
 - Relationship columns

Connect to Salesforce Objects from Power Query Desktop

To connect to Salesforce Objects data:

1. Select **Salesforce Objects** from the product-specific data connector list, and then select **Connect**.
2. In **Salesforce Objects**, choose the **Production** URL if you use the Salesforce production URL (<https://www.salesforce.com>) to sign in.

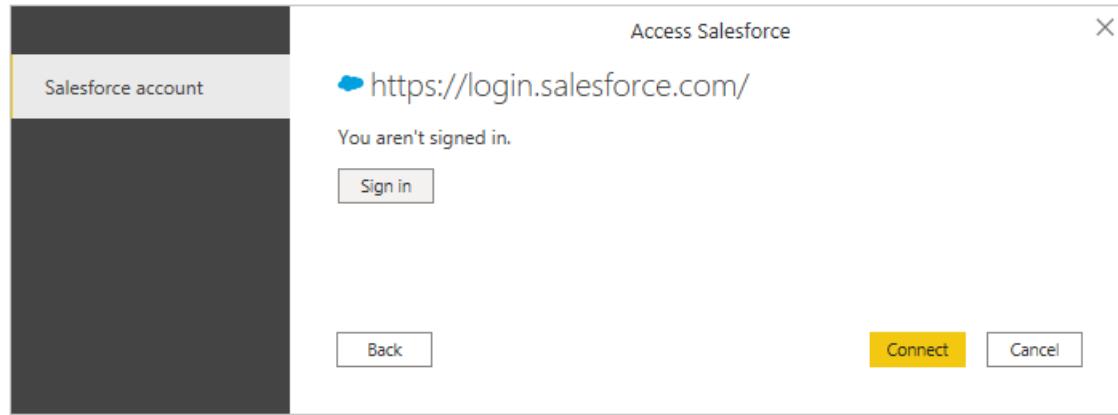


You can also select **Custom** and enter a custom URL to sign in. This custom URL might be a custom domain you've created within Salesforce, such as <https://contoso.salesforce.com>. You can also use the custom URL selection if you're using your own CNAME record that redirects to Salesforce.

Also, you can select **Include relationship columns**. This selection alters the query by including columns that might have foreign-key relationships to other tables. If this box is unchecked, you won't see those columns.

Once you've selected the URL, select **OK** to continue.

3. Select **Sign in** to sign in to your Salesforce account.



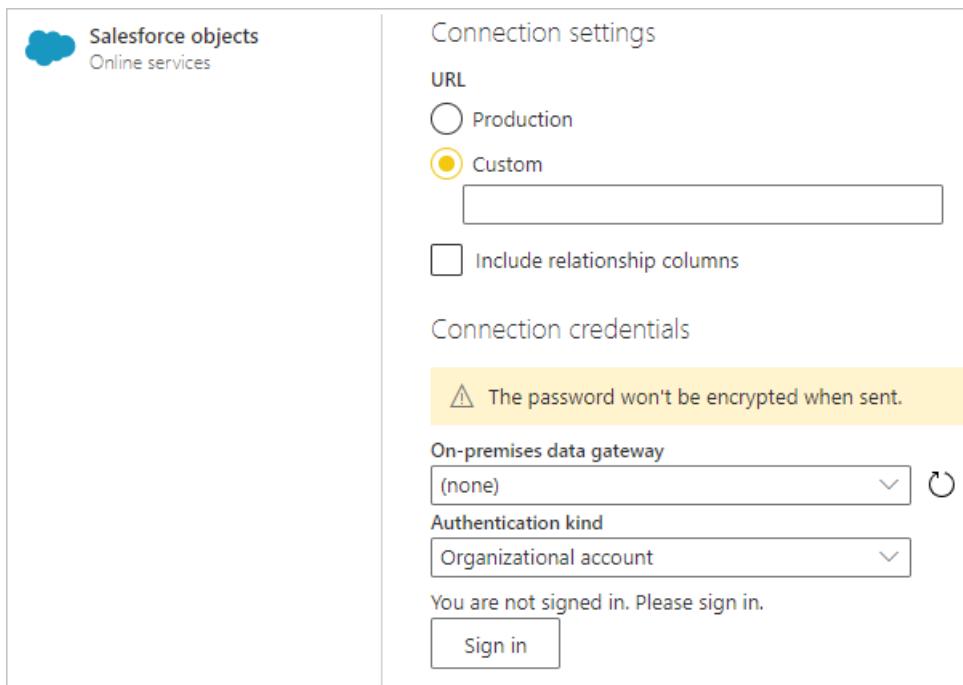
Once you've successfully signed in, select **Connect**.

4. If this is the first time you've signed in using a specific app, you'll be asked to verify your authenticity by entering a code sent to your email address. You'll then be asked whether you want the app you're using to access the data. For example, you'll be asked if you want to allow Power BI Desktop to access your Salesforce data. Select **Allow**.
5. In the **Navigator** dialog box, select the Salesforce Objects you want to load. You can then either select **Load** to load the data or select **Transform Data** to transform the data.

Connect to Salesforce Objects from Power Query Online

To connect to Salesforce Objects data:

1. Select **Salesforce objects** from the product-specific data connector list, and then select **Connect**.
2. In **Salesforce objects**, choose the URL you want to use to connect. Select the **Production** URL if you use the Salesforce production URL (<https://www.salesforce.com>) to sign in.



NOTE

Currently, you may need to select the **Custom** URL, enter <https://www.salesforce.com> in the text box, and then select **Production** to connect to your data.

You can also select Custom and enter a custom URL to sign in. This custom URL might be a custom domain you've created within Salesforce, such as <https://contoso.salesforce.com>. You can also use the custom URL selection if you're using your own CNAME record that redirects to Salesforce.

Also, you can select **Include relationship columns**. This selection alters the query by including columns that might have foreign-key relationships to other tables. If this box is unchecked, you won't see those columns.

3. If this is the first time you've made this connection, select an on-premises data gateway, if needed.
4. Select **Sign in** to sign in to your Salesforce account. Once you've successfully signed in, select **Next**.
5. In the **Navigator** dialog box, select the Salesforce Objects you want to load. Then select **Transform Data** to transform the data.

Known issues and limitations

- There's a limit on the number of fields a query to Salesforce can contain. The limit varies depending on the type of the columns, the number of computed columns, and so on. When you receive the `Query is either selecting too many fields or the filter conditions are too complicated` error, it means that your query exceeds the limit. To avoid this error, use the **Select Query advanced** option and specify fields that you really need.
- Salesforce session settings can block this integration. Ensure that the setting **Lock sessions to the IP address from which they originated** is disabled.

- Salesforce API access should be enabled. To verify access settings, go to profile settings for the current user and search for "API Enabled" checkbox.
- Salesforce trial accounts don't have API access.
- Custom fields of type "Picklist (Multi-Select)" are not supported by "Create record" and "Update record" operations.
- Lightning URLs aren't supported.

For more information about Salesforce internal API limits, go to [Salesforce Developer Limits and Allocations Quick Reference](#).

Salesforce Reports

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Salesforce account

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

WARNING

By default, Salesforce does not support Internet Explorer 11, which is used as part of the authentication experience to online services in Power Query Desktop. Please opt-in for [extended support for accessing Lightning Experience Using Microsoft Internet Explorer 11](#). You may also want to review Salesforce documentation on [configuring Internet Explorer](#). At this time, users will be impaired from authenticating, but stored credentials should continue to work until their existing authentication tokens expire. To resolve this, go to [Enabling Microsoft Edge \(Chromium\) for OAuth Authentication in Power BI Desktop](#).

Prerequisites

To use the Salesforce Reports connector, you must have a Salesforce account username and password.

Also, Salesforce API access should be enabled. To verify access settings, go to your personal Salesforce page, open your profile settings, and search for and make sure the **API Enabled** checkbox is selected. Note that Salesforce trial accounts don't have API access.

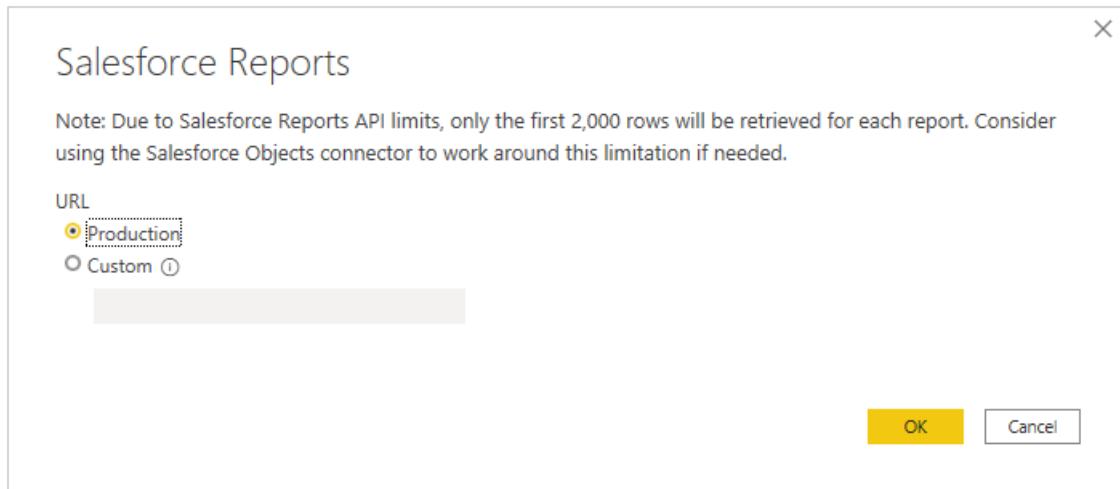
Capabilities Supported

- Production
- Custom
 - Custom domains
 - CNAME record redirects

Connect to Salesforce Reports from Power Query Desktop

To connect to Salesforce Reports data:

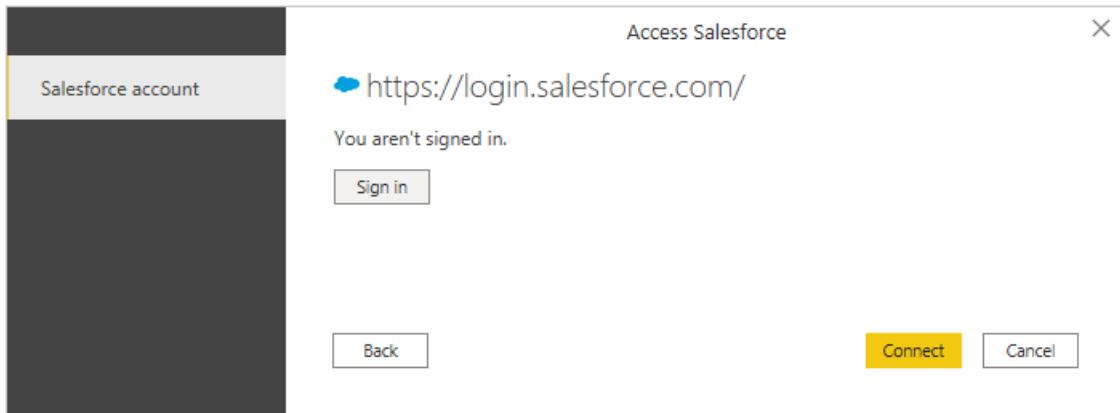
1. Select **Salesforce Reports** from the product-specific data connector list, and then select **Connect**.
2. In **Salesforce Reports**, choose the **Production** URL if you use the Salesforce production URL (<https://www.salesforce.com>) to sign in.



You can also select **Custom** and enter a custom URL to sign in. This custom URL might be a custom domain you've created within Salesforce, such as <https://contoso.salesforce.com>. You can also use the custom URL selection if you're using your own CNAME record that redirects to Salesforce.

Once you've selected the URL, select **OK** to continue.

3. Select **Sign in** to sign in to your Salesforce account.



Once you've successfully signed in, select **Connect**.

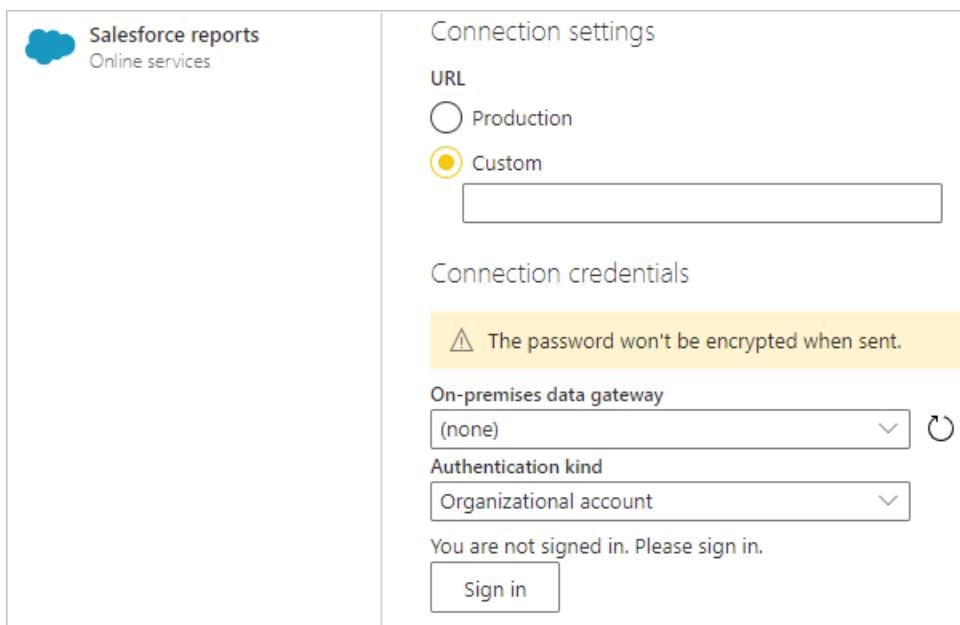
4. If this is the first time you've signed in using a specific app, you'll be asked to verify your authenticity by entering a code sent to your email address. You'll then be asked whether you want the app you're using to access the data. For example, you'll be asked if you want to allow Power BI Desktop to access your Salesforce data. Select **Allow**.
5. In the **Navigator** dialog box, select the Salesforce Reports you want to load. You can then either select **Load** to load the data or select **Transform Data** to transform the data.

Connect to Salesforce Reports from Power Query Online

To connect to Salesforce Reports data:

1. Select **Salesforce reports** from the product-specific data connector list, and then select **Connect**.

2. In **Salesforce reports**, choose the URL you want to use to connect. Select the **Production** URL if you use the Salesforce production URL (<https://www.salesforce.com>) to sign in.



NOTE

Currently, you may need to select the **Custom** URL, enter <https://www.salesforce.com> in the text box, and then select **Production** to connect to your data.

You can also select **Custom** and enter a custom URL to sign in. This custom URL might be a custom domain you've created within Salesforce, such as <https://contoso.salesforce.com>. You can also use the custom URL selection if you're using your own CNAME record that redirects to Salesforce.

Also, you can select **Include relationship columns**. This selection alters the query by including columns that might have foreign-key relationships to other tables. If this box is unchecked, you won't see those columns.

3. If this is the first time you've made this connection, select an on-premises data gateway, if needed.
4. Select **Sign in** to sign in to your Salesforce account. Once you've successfully signed in, select **Next**.
5. In the **Navigator** dialog box, select the Salesforce Reports you want to load. Then select **Transform Data** to transform the data.

Known issues and limitations

- There's a limit on the number of fields a query to Salesforce can contain. The limit varies depending on the type of the columns, the number of computed columns, and so on. When you receive an `Query is either selecting too many fields or the filter conditions are too complicated` error, it means that your query exceeds the limit. To avoid this error, use the **Select Query** advanced option and specify fields that you really need.
- Salesforce session settings can block this integration. Ensure that the setting **Lock sessions to the IP address from which they originated** is disabled.
- The number of rows you can access in Salesforce Reports is limited by Salesforce to 2000 rows. As a workaround for this issue, you can use the **Salesforce Objects** connector in Power BI Desktop to retrieve all the rows from individual tables and recreate reports you'd like. The Object connector doesn't have the 2000-row limit.

- Salesforce API access should be enabled. To verify access settings, go to profile settings for the current user and search for "API Enabled" checkbox.
- Salesforce trial accounts don't have API access.
- Lightning URLs aren't supported.

For more information about Salesforce internal API limits, go to [Salesforce Developer Limits and Allocations Quick Reference](#).

SAP Business Warehouse Application Server

1/15/2022 • 5 minutes to read • [Edit Online](#)

NOTE

The SAP Business Warehouse (BW) Application Server connector is now certified for SAP BW/4HANA as of June 2020.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Analysis Services
Authentication Types Supported	Windows (desktop) Database (desktop) Basic (online)
Function Reference Documentation	SapBusinessWarehouse.Cubes sapbusinesswarehouseexecutionmode.datastream SapBusinessWarehouseExecutionMode.BasXml SapBusinessWarehouseExecutionMode.BasXmlGzip

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

You'll need an SAP account to sign in to the website and download the drivers. If you're unsure, contact the SAP administrator in your organization. The drivers need to be installed on the gateway machine.

You can use either version 1.0 of the SAP Business Warehouse (BW) Application Server connector or the Implementation 2.0 SAP connector in Power Query Desktop. The following sections describe the installation of each version, in turn. You can choose one or the other connector when connecting to an SAP BW Application Server from Power BI Desktop.

BW 7.3, BW 7.5 and BW/4HANA 2.0 is supported.

NOTE

We suggest you use the Implementation 2.0 SAP connector whenever possible because it provides significant performance, functionality, and reliability improvements over 1.0.

NOTE

Power Query Online uses the version 2.0 SAP BW Application Server connector by default. However, version 1.0 of the SAP BW Application Server connector works in the M Engine level if you really need to use it.

Prerequisites for version 1.0

To use version 1.0 of the SAP BW Application Server connector, you must install the *SAP NetWeaver* library on your local computer. You can get the SAP NetWeaver library from your SAP administrator or directly from the [SAP Software Download Center](#). Since the SAP Software Download Center changes its structure frequently, more specific guidance for navigating that site isn't available. The SAP NetWeaver library is usually included in the SAP Client Tools installation.

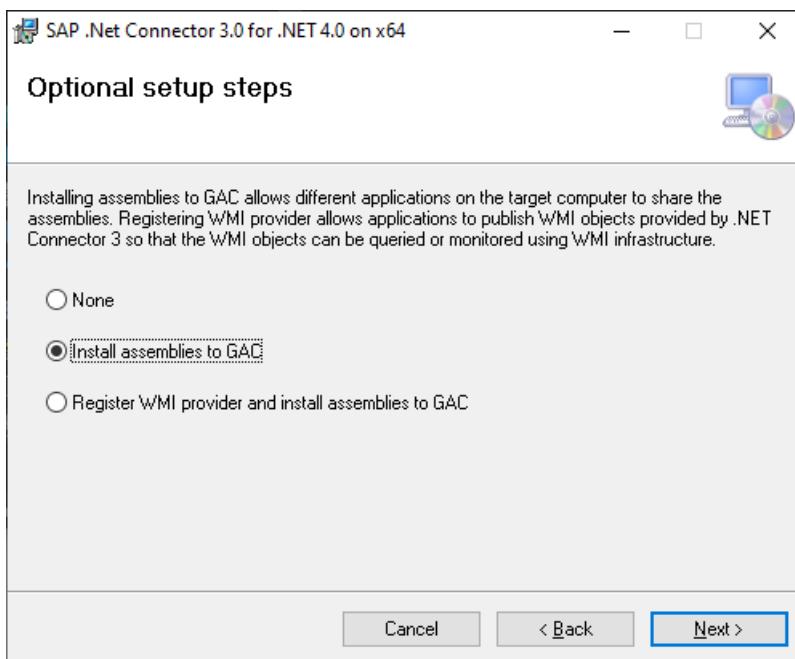
You can search for *SAP Note #1025361* to get the download location for the most recent version. Make sure the architecture for the SAP NetWeaver library (32-bit or 64-bit) matches your Power BI Desktop installation. Install all files included in the *SAP NetWeaver RFC SDK* according to the SAP Note.

Prerequisites for version 2.0

To use the version 2.0 SAP BW Application Server connector in Power BI Desktop or Power Query Online, you must install the SAP .NET Connector 3.0. Access to the download requires a valid S-user. Contact your SAP Basis team to get the SAP .NET Connector 3.0. You can download the [SAP .NET Connector 3.0](#) from SAP. The connector comes in 32-bit and 64-bit versions. Choose the version that matches your Power BI Desktop installation. For Power Query Online, choose the 64-bit version. Currently, the website lists two versions for .NET 4.0 framework:

- SAP Connector for Microsoft .NET 3.0.23.0 for Windows 32 bit (x86) as a zip file (6,928 KB), May 28, 2020
- SAP Connector for Microsoft .NET 3.0.23.0 for Windows 64 bit (x64) as a zip file (7,225 KB), May 28, 2020

When you install, in **Optional setup steps**, make sure you select **Install assemblies to GAC**.



NOTE

If you want to use version 1 of the SAP BW Application Server connector, you must use the *SAP NetWeaver* library. For more information about installing version 1, see [Prerequisites for version 1.0](#). We recommend using the Implementation 2.0 SAP BW Application Server connector whenever possible.

Capabilities Supported

- Import
- Direct Query
- Implementation
 - 2.0 (Requires SAP .NET Connector 3.0)
 - 1.0 (Requires NetWeaver RFC)
- Advanced
 - Language code
 - Execution mode
 - Batch size
 - MDX statement
 - Enable characteristic structures

Connect to an SAP BW Application Server from Power Query Desktop

To connect to an SAP BW Application Server:

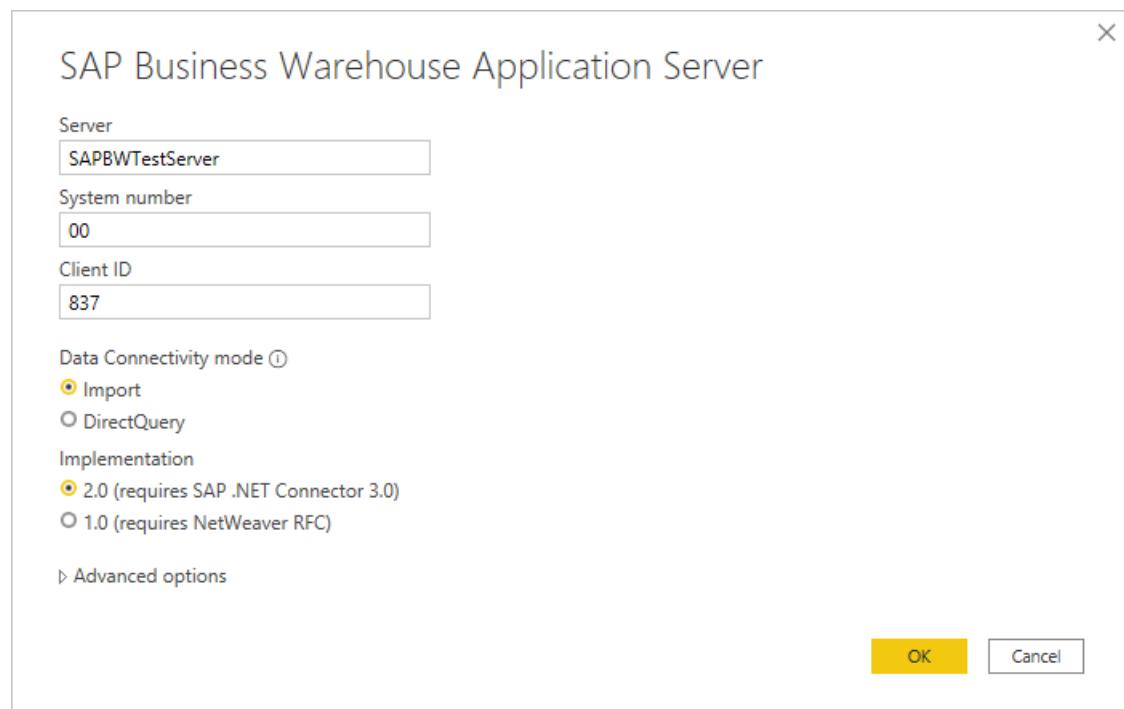
1. From the **Home** tab of Power BI Desktop, select **Get Data > SAP Business Warehouse Application Server**.
2. Enter the server name, system number, and client ID of the SAP BW Application Server you want to connect to. This example uses **SAPBWTTestServer** as the server name, a system number of **00**, and a client ID of **837**.

The rest of this example describes how to import your data into Power Query Desktop, which is the default setting for **Data Connectivity mode**. If you want to use DirectQuery to load your data, see [Connect to SAP Business Warehouse by using DirectQuery in Power BI](#).

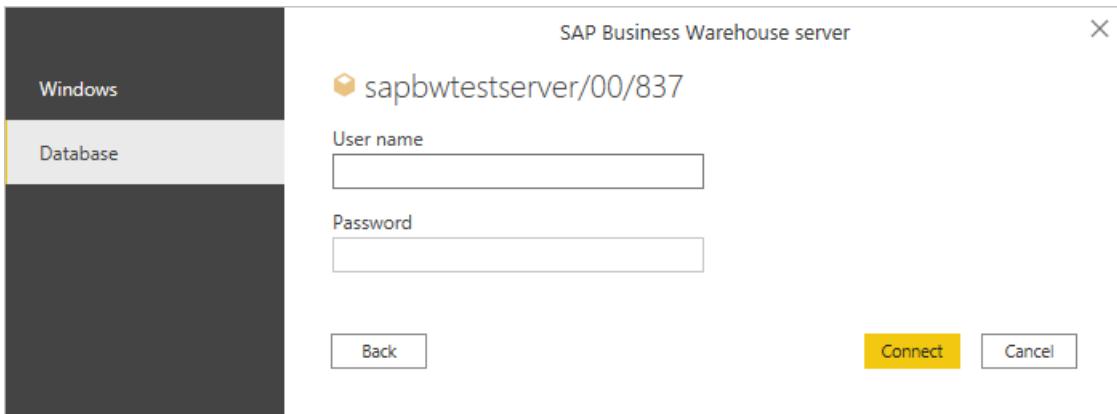
This example also uses the default **Implementation of 2.0 (requires SAP .NET Connector 3.0)**. If you want to use version 1 of the SAP BW Application Server connector, select **1.0 (requires NetWeaver RFC)**.

If you want to use any of the advanced options for this connector to fine-tune your query, go to [Use advanced options](#).

When you've finished filling in the relevant information, select **OK**.



3. When accessing the database for the first time, the SAP BW Application Server requires database user credentials. Power Query Desktop offers two authentication modes for SAP BW connections—user name/password authentication (Database), and Windows authentication (single sign-on). SAML authentication isn't currently supported. Select either **Windows** or **Database**. If you select **Database** authentication, enter your user name and password. If you select **Windows** authentication, go to [Windows Authentication and single sign-on](#) to learn more about the requirements for Windows authentication.



Then select **Connect**.

For more information about authentication, go to [Authentication with a data source](#).

4. From the **Navigator** dialog box, select the items you want to use. When you select one or more items from the server, the **Navigator** dialog box creates a preview of the output table. For more information about navigating the SAP BW Application Server query objects in Power Query, go to [Navigate the query objects](#).

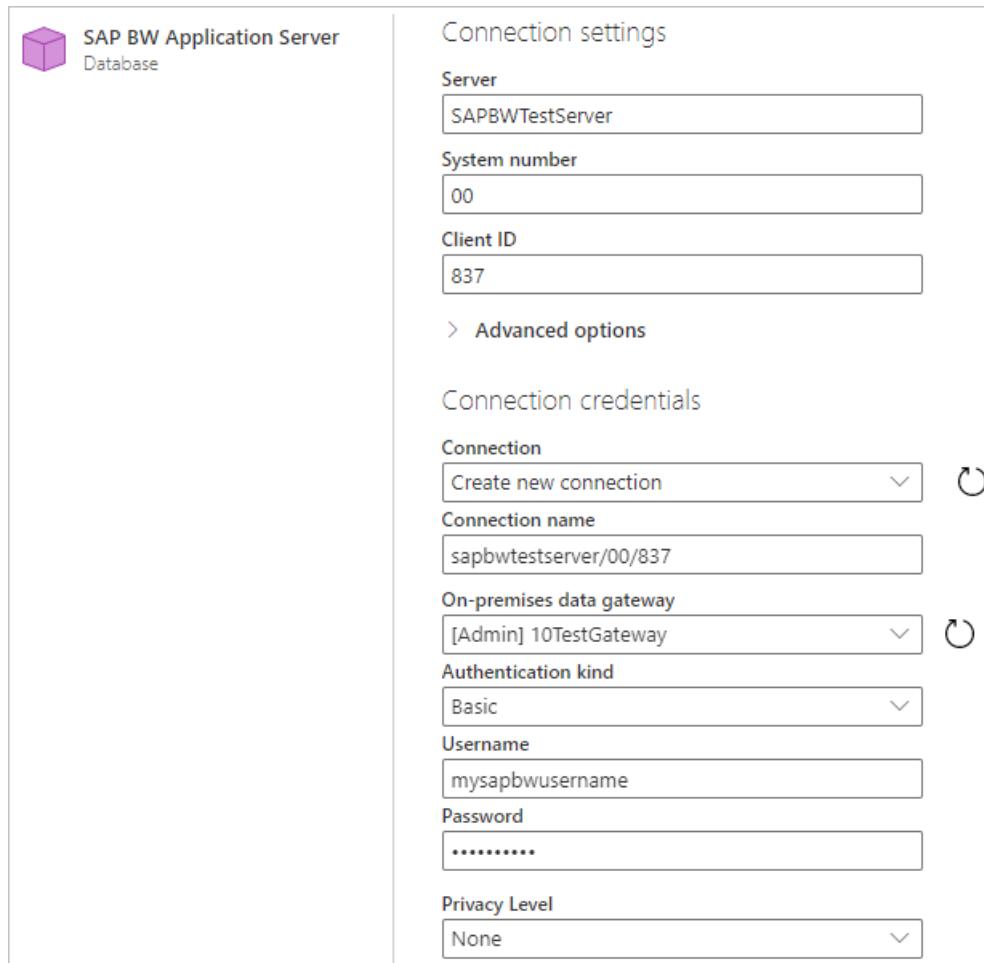
Purchasing Overview		
Vendor.Vendor Level 01	Vendor.Vendor Level 01.Vendor Level 01.UniqueName	Vendor
Sunny Electronics Inc.	[0D_VENDOR].[0000001000]	
PAQ_Germany GmbH	[0D_VENDOR].[0000001001]	
Jotachi Deutschland AG	[0D_VENDOR].[0000001050]	
Computer 3000 Inc.	[0D_VENDOR].[0000001300]	
CompSmart Inc.	[0D_VENDOR].[0000001370]	
Technik und Systeme GmbH	[0D_VENDOR].[0000001500]	
Phoenix Supplies	[0D_VENDOR].[0000001650]	
Pyramid Systems	[0D_VENDOR].[0000002000]	
HiTech Corp	[0D_VENDOR].[0000002100]	
Omnium	[0D_VENDOR].[0000002210]	
Asia Technologies	[0D_VENDOR].[0000002500]	
Computer Competence Center AG	[0D_VENDOR].[0000003000]	
PC Warehouse	[0D_VENDOR].[0000003010]	
CompuMax Corp	[0D_VENDOR].[0000003090]	
C.E.B Paris	[0D_VENDOR].[0000003176]	
Becker Components AG	[0D_VENDOR].[0000003201]	
Générale Electronique	[0D_VENDOR].[0000003300]	
Logo Systems	[0D_VENDOR].[0000003390]	
SAPSOTA Corp	[0D_VENDOR].[0000003500]	
ABC Technology	[0D_VENDOR].[0000003510]	
Superminus	[0D_VENDOR].[0000003660]	
Hatushiba Co. Ltd	[0D_VENDOR].[0000004151]	
Marvick Inc.	[0D_VENDOR].[0000005510]	

5. From the **Navigator** dialog box, you can either transform the data in the Power Query Editor by selecting **Transform Data**, or load the data by selecting **Load**.

Connect to an SAP BW Application Server from Power Query Online

To connect to an SAP BW Application Server from Power Query Online:

1. From the **Data sources** page, select **SAP BW Application Server**.
2. Enter the server name, system number, and client ID of the SAP BW Application Server you want to connect to. This example uses **SAPBWTTestServer** as the server name, a system number of **00**, and a client ID of **837**.
3. Select the [on-premises data gateway](#) you want to use to connect to the data.
4. Set **Authentication Kind** to **Basic**. Enter your user name and password.



5. You can also select from a set of [advanced options](#) to fine-tune your query.
6. Select **Next** to connect.
7. From the **Navigator** dialog box, select the items you want to use. When you select one or more items from the server, the **Navigator** dialog box creates a preview of the output table. For more information about navigating the SAP BW Application Server query objects in Power Query, go to [Navigate the query objects](#).
8. From the **Navigator** dialog box, you can transform the data in the Power Query Editor by selecting **Transform Data**.

Power Query - Choose data

Purchasing Overview

1..2 Invoiced Amount	1..2 Delivery Time	A ^B _C Vendor Level 01	A ^B _C Vendor Level 01.Vendor Level 01.UniqueName	A ^B _C Vendor Level 01.Key	A ^B _C Vendor Le...
5588152	420	Sunny Electronics Inc.	[OD_VENDOR].[0000001000]	1000	Sunny Elec...
9132897	378	PAQ Germany GmbH	[OD_VENDOR].[0000001001]	1001	PAQ Germi...
4309632	288	Jotachi Deutschland AG	[OD_VENDOR].[0000001050]	1050	Jotachi De...
4420784	198	Computer 3000 Inc.	[OD_VENDOR].[0000001300]	1300	Computer...
13285140	414	CompSmart Inc.	[OD_VENDOR].[0000001370]	1370	CompSmar...
3331875	150	Technik und Systeme GmbH	[OD_VENDOR].[0000001500]	1500	Technik un...
2604640	165	Phoenix Supplies	[OD_VENDOR].[0000001650]	1650	Phoenix Su...
5162710	360	Pyramid Systems	[OD_VENDOR].[0000002000]	2000	Pyramid Sy...
2537990	216	HiTech Corp	[OD_VENDOR].[0000002100]	2100	HiTech Cor...
5929962	648	Omnium	[OD_VENDOR].[0000002210]	2210	Omnium...
460502	108	Asia Technologies	[OD_VENDOR].[0000002500]	2500	Asia Techn...
5066768	270	Computer Competence Center ...	[OD_VENDOR].[0000003000]	3000	Computer C...
2346024	363	PC Warehouse	[OD_VENDOR].[0000003010]	3010	PC Wareho...
10730400	528	CompuMax Corp	[OD_VENDOR].[0000003090]	3090	CompuMax ...
2021691	270	C.E.B Paris	[OD_VENDOR].[0000003176]	3176	C.E.B Paris...
16654265	984	Becker Components AG	[OD_VENDOR].[0000003201]	3201	Becker Com...
7725896	378	Générale Electronique	[OD_VENDOR].[0000003300]	3300	Générale E...
4177492	459	Logo Systems	[OD_VENDOR].[0000003390]	3390	Logo Syste...
341640	3	SAPSOTA Corp	[OD_VENDOR].[0000003500]	3500	SAPSOTA C...

Back Cancel Transform data

See also

- [Navigate the query objects](#)
- [SAP Business Warehouse fundamentals](#)
- [Use advanced options](#)
- [SAP Business Warehouse connector troubleshooting](#)

SAP Business Warehouse Message Server

1/15/2022 • 5 minutes to read • [Edit Online](#)

NOTE

The SAP Business Warehouse (BW) Message Server connector is now certified for SAP BW/4HANA as of June 2020.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows)
Authentication Types Supported	Windows (desktop) Database (desktop) Basic (online)
Function Reference Documentation	SapBusinessWarehouse.Cubes sapbusinesswarehouseexecutionmode.datastream SapBusinessWarehouseExecutionMode.BasXml SapBusinessWarehouseExecutionMode.BasXmlGzip

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

You'll need an SAP account to sign in to the website and download the drivers. If you're unsure, contact the SAP administrator in your organization.

You can use either version 1.0 of the SAP Business Warehouse (BW) Message Server connector or the Implementation 2.0 SAP connector in Power Query Desktop. The following sections describe the installation of each version, in turn. You can choose one or the other connector when connecting to an SAP BW Message Server from Power BI Desktop. We suggest you use the Implementation 2.0 SAP connector whenever possible.

NOTE

We suggest you use the Implementation 2.0 SAP connector whenever possible because it provides significant performance, functionality, and reliability improvements over 1.0.

NOTE

Power Query Online uses the version 2.0 SAP BW Message Server connector by default. However, version 1.0 of the SAP BW Message Server connector works in the M Engine level if you really need to use it.

Prerequisites for version 1.0

To use version 1.0 of the SAP BW Message Server connector, you must install the *SAP NetWeaver* library on your local computer. You can get the SAP NetWeaver library from your SAP administrator or directly from the [SAP Software Download Center](#). Since the SAP Software Download Center changes its structure frequently, more specific guidance for navigating that site isn't available. The SAP NetWeaver library is usually included in the SAP Client Tools installation.

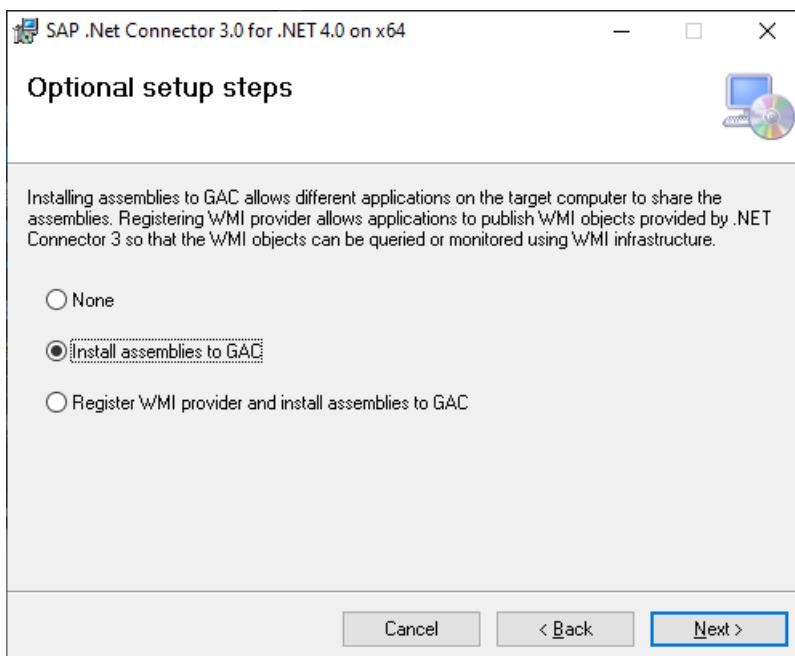
You can search for *SAP Note #1025361* to get the download location for the most recent version. Make sure the architecture for the SAP NetWeaver library (32-bit or 64-bit) matches your Power BI Desktop installation. Install all files included in the *SAP NetWeaver RFC SDK* according to the SAP Note.

Prerequisites for version 2.0

To use the version 2.0 SAP BW Message Server connector in Power BI Desktop or Power Query Online, you must install the SAP .NET Connector 3.0. Access to the download requires a valid S-user. Contact your SAP Basis team to get the SAP .NET Connector 3.0. You can download the [SAP .NET Connector 3.0](#) from SAP. The connector comes in 32-bit and 64-bit versions. Choose the version that matches your Power BI Desktop installation. For Power Query Online, choose the 64-bit version. Currently, the website lists two versions for .NET 4.0 framework:

- SAP Connector for Microsoft .NET 3.0.23.0 for Windows 32 bit (x86) as a zip file (6,928 KB), May 28, 2020
- SAP Connector for Microsoft .NET 3.0.23.0 for Windows 64 bit (x64) as a zip file (7,225 KB), May 28, 2020

When you install, in **Optional setup steps**, make sure you select **Install assemblies to GAC**.



NOTE

If you want to use version 1.0 of the SAP BW Message Server connector, you must use the *SAP NetWeaver* library. For more information about installing version 1.0, see [Prerequisites for version 1.0](#). We recommend using the Implementation 2.0 SAP BW Message Server connector whenever possible.

Capabilities Supported

- Import
- Direct Query
- Implementation
 - 2.0 (Requires SAP .NET Connector 3.0)
 - 1.0 (Requires NetWeaver RFC)
- Advanced
 - Language code
 - Execution mode
 - Batch size
 - MDX statement
 - Enable characteristic structures

Connect to an SAP BW Message Server from Power Query Desktop

To connect to an SAP BW Message Server:

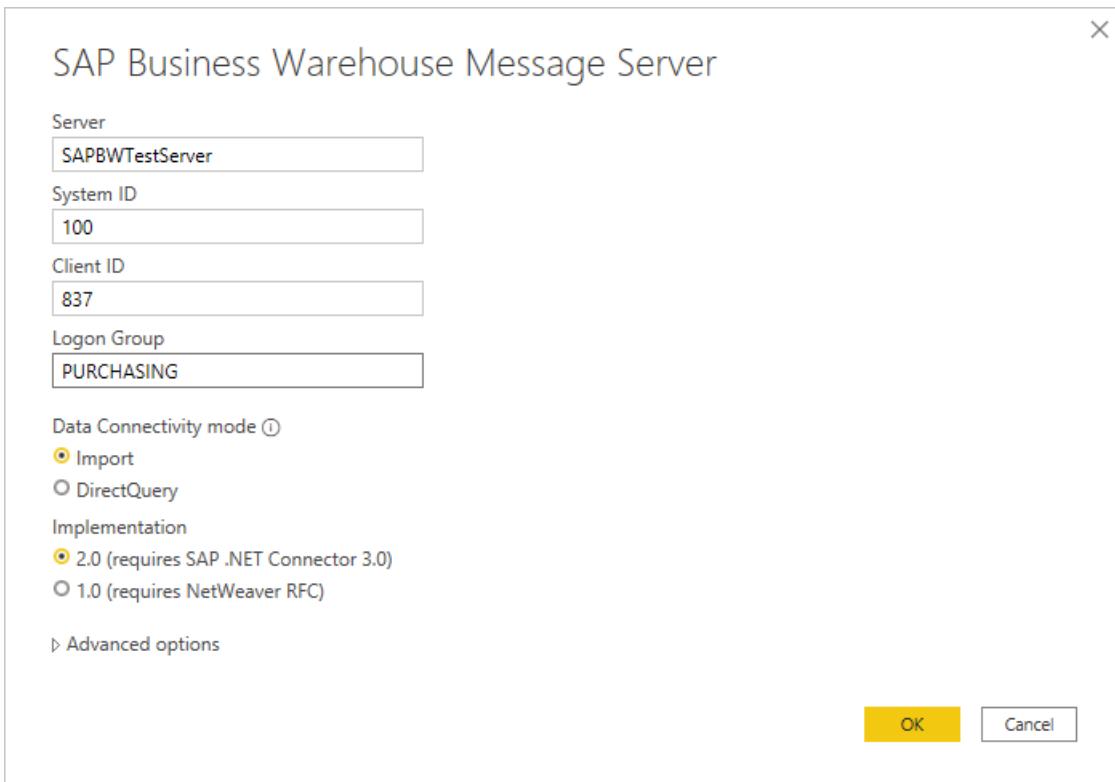
1. From the **Home** tab of Power BI Desktop, select **Get Data > SAP Business Warehouse Message Server**.
2. Enter the server, system number, client ID, and logon group of the SAP BW Message Server you want to connect to. This example uses `SAPBWTTestServer` as the server name, a system number of `100`, a client ID of `837`, and a logon group of `PURCHASING`.

The rest of this example describes how to import your data into Power Query Desktop, which is the default setting for **Data Connectivity mode**. If you want to use DirectQuery to load your data, see [Connect to SAP Business Warehouse by using DirectQuery in Power BI](#).

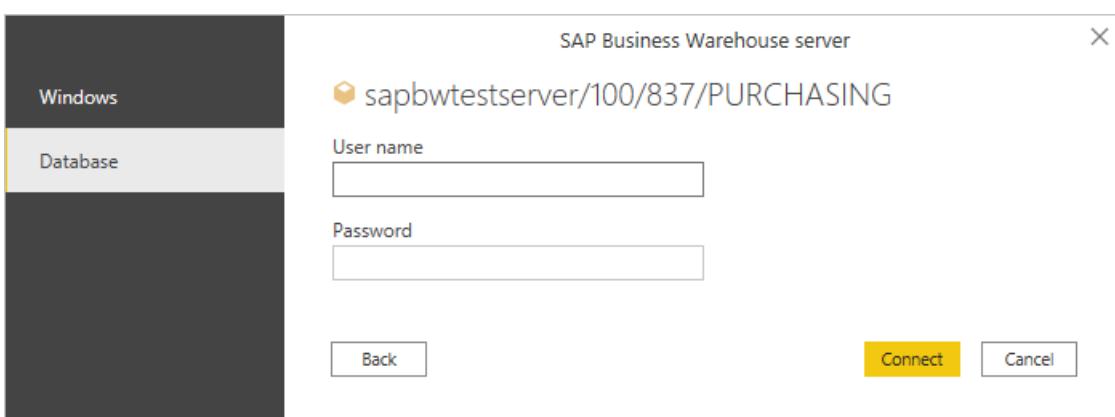
This example also uses the default **Implementation of 2.0 (requires SAP .NET Connector 3.0)**. If you want to use version 1 of the SAP BW Message Server connector, select **1.0 (requires NetWeaver RFC)**.

If you want to use any of the advanced options for this connector to fine-tune your query, go to [Use advanced options](#).

When you've finished filling in the relevant information, select **OK**.



- When accessing the database for the first time, the SAP BW Message Server requires database user credentials. Power Query Desktop offers two authentication modes for SAP BW connections—user name/password authentication (Database), and Windows authentication (single sign-on). SAML authentication isn't currently supported. Select either **Windows** or **Database**. If you select **Database** authentication, enter your user name and password. If you select **Windows** authentication, go to [Windows Authentication and single sign-on](#) to learn more about the requirements for Windows authentication.



Then select **Connect**.

For more information about authentication, go to [Authentication with a data source](#).

- From the **Navigator** dialog box, select the items you want to use. When you select one or more items from the server, the **Navigator** dialog box creates a preview of the output table. For more information about navigating the SAP BW Message Server query objects in Power Query, go to [Navigate the query objects](#).

Purchasing Overview

Vendor.Vendor Level 01	Vendor.Vendor Level 01.Vendor Level 01.UniqueName
Sunny Electronics Inc.	[OD_VENDOR].[0000001000]
PAQ_Germany GmbH	[OD_VENDOR].[0000001001]
Jotachi Deutschland AG	[OD_VENDOR].[0000001050]
Computer 3000 Inc.	[OD_VENDOR].[0000001300]
CompSmart Inc.	[OD_VENDOR].[0000001370]
Technik und Systeme GmbH	[OD_VENDOR].[0000001500]
Phoenix Supplies	[OD_VENDOR].[0000001650]
Pyramid Systems	[OD_VENDOR].[0000002000]
HiTech Corp	[OD_VENDOR].[0000002100]
Omnium	[OD_VENDOR].[0000002210]
Asia Technologies	[OD_VENDOR].[0000002500]
Computer Competence Center AG	[OD_VENDOR].[0000003000]
PC Warehouse	[OD_VENDOR].[0000003010]
CompuMax Corp	[OD_VENDOR].[0000003090]
C.E.B Paris	[OD_VENDOR].[0000003176]
Becker Components AG	[OD_VENDOR].[0000003201]
Générale Electronique	[OD_VENDOR].[0000003300]
Logo Systems	[OD_VENDOR].[0000003390]
SAPSOTA Corp	[OD_VENDOR].[0000003500]
ABC Technology	[OD_VENDOR].[0000003510]
Superminus	[OD_VENDOR].[0000003660]
Hatushiba Co. Ltd	[OD_VENDOR].[0000004151]
Marwick Inc.	[OD_VENDOR].[0000005510]

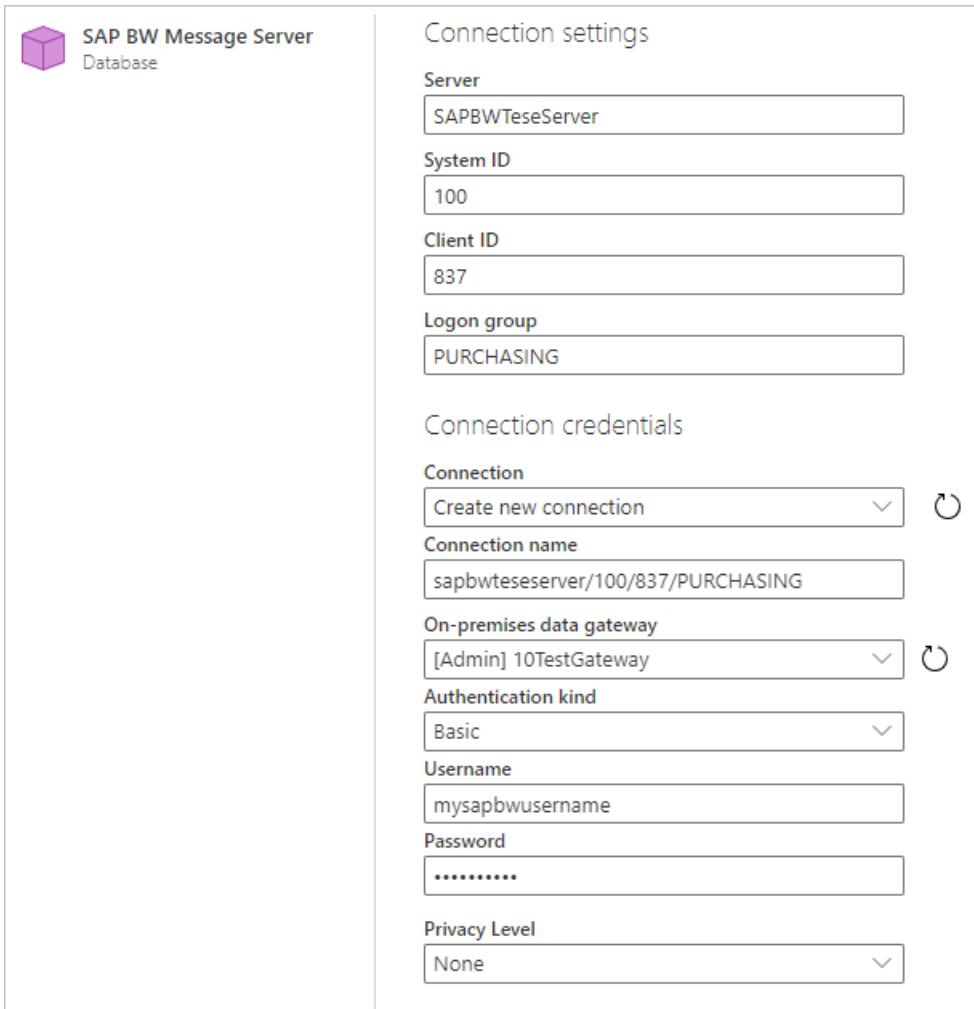
Load Transform Data Cancel

- From the **Navigator** dialog box, you can either transform the data in the Power Query Editor by selecting **Transform Data**, or load the data by selecting **Load**.

Connect to an SAP BW Message Server from Power Query Online

To connect to an SAP BW Message Server from Power Query Online:

- From the **Data sources** page, select **SAP BW Message Server**.
- Enter the server, system number, client ID, and logo group of the SAP BW Message Server you want to connect to. This example uses **SAPBWTTestServer** as the server name, a system number of **100**, a client ID of **837**, and a logon group of **PURCHASING**.
- Select the **on-premises data gateway** you want to use to connect to the data.
- Set **Authentication Kind** to **Basic**. Enter your user name and password.



5. You can also select from a set of [advanced options](#) to fine-tune your query.
6. Select **Next** to connect.
7. From the **Navigator** dialog box, select the items you want to use. When you select one or more items from the server, the **Navigator** dialog box creates a preview of the output table. For more information about navigating the SAP BW Message Server query objects in Power Query, go to [Navigate the query objects](#).
8. From the **Navigator** dialog box, you can transform the data in the Power Query Editor by selecting **Transform Data**.

Power Query - Choose data

Purchasing Overview

1.2 Invoiced Amount	1.2 Delivery Time	A ^B _C Vendor Level 01	A ^B _C Vendor Level 01.Vendor Level 01.UniqueName	A ^B _C Vendor Level 01.Key	A ^B _C Vendor Le...
5588152	420	Sunny Electronics Inc.	[OD_VENDOR].[0000001000]	1000	Sunny Elec...
9132897	378	PAQ Germany GmbH	[OD_VENDOR].[0000001001]	1001	PAQ Germ...
4309632	288	Jotachi Deutschland AG	[OD_VENDOR].[0000001050]	1050	Jotachi De...
4420784	198	Computer 3000 Inc.	[OD_VENDOR].[0000001300]	1300	Computer...
13285140	414	CompSmart Inc.	[OD_VENDOR].[0000001370]	1370	CompSmar...
3331875	150	Technik und Systeme GmbH	[OD_VENDOR].[0000001500]	1500	Technik un...
2604640	165	Phoenix Supplies	[OD_VENDOR].[0000001650]	1650	Phoenix Su...
5162710	360	Pyramid Systems	[OD_VENDOR].[0000002000]	2000	Pyramid Sy...
2537990	216	HiTech Corp	[OD_VENDOR].[0000002100]	2100	HiTech Cor...
5929362	648	Omnium	[OD_VENDOR].[0000002210]	2210	Omnium...
460502	108	Asia Technologies	[OD_VENDOR].[0000002500]	2500	Asia Techn...
5066768	270	Computer Competence Center ...	[OD_VENDOR].[0000003000]	3000	Computer C...
2346024	363	PC Warehouse	[OD_VENDOR].[0000003010]	3010	PC Wareho...
10730400	528	CompuMax Corp	[OD_VENDOR].[0000003090]	3090	CompuMax...
2021691	270	C.E.B Paris	[OD_VENDOR].[0000003176]	3176	C.E.B Paris...
16654265	984	Becker Components AG	[OD_VENDOR].[0000003201]	3201	Becker Com...
7725896	378	Générale Electronique	[OD_VENDOR].[0000003300]	3300	Générale E...
4177492	459	Logo Systems	[OD_VENDOR].[0000003390]	3390	Logo Syste...
341640	3	SAPSOTA Corp	[OD_VENDOR].[0000003500]	3500	SAPSOTA C...

Back Cancel Transform data

See also

- Navigate the query objects
- SAP Business Warehouse fundamentals
- Use advanced options
- SAP Business Warehouse connector troubleshooting

SAP BW fundamentals

1/15/2022 • 5 minutes to read • [Edit Online](#)

This article describes basic terminology used when describing interactions between the SAP BW server and Power Query. It also includes information about tools that you may find useful when using the Power Query SAP BW connector.

Integration Architecture

From a technical point of view, the integration between applications and SAP BW is based on the so-called Online Analytical Processing (OLAP) Business Application Programming Interfaces (BAPI).

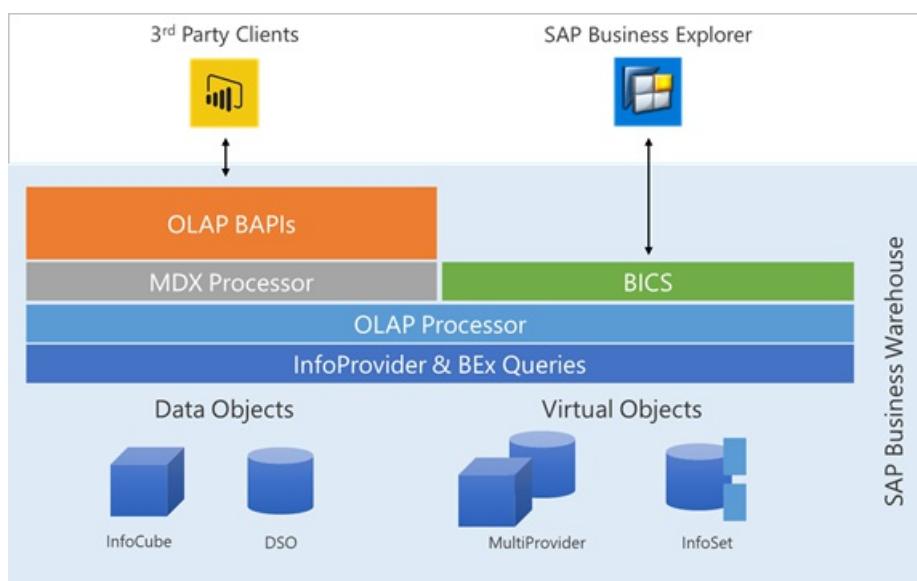
The OLAP BAPIs are delivered with SAP BW and provide 3rd-parties and developers with standardized interfaces that enable them to access the data and metadata of SAP BW with their own front-end tools.

Applications of all types can be connected with an SAP BW server using these methods.

The OLAP BAPIs are implemented in SAP BW as RFC-enabled function modules and are invoked by applications over SAP's RFC protocol. This requires the [NetWeaver RFC Library or SAP .NET Connector](#) to be installed on the application's machine.

The OLAP BAPIs provide methods for browsing metadata and master data, and also for passing MDX statements for execution to the MDX Processor.

The OLAP Processor is responsible for retrieving, processing, and formatting the data from the SAP BW source objects, which are further described in [SAP BW data source](#) and [Data objects in SAP BW](#).



SAP Business Explorer and other SAP tools use a more direct interface to the SAP BW OLAP Processor called Business Intelligence Consumer Services, commonly known as BICS. BICS isn't available for 3rd party tools.

SAP BW data sources

The OLAP BAPIs provide 3rd party applications with access to SAP BW InfoProviders and BEx Queries.

Typically, when a 3rd party tool like Power Query connects using the OLAP BAPIs, SAP BW first responds with a list of catalogs available in the SAP BW system.

There's one catalog with the technical name **\$INFOCUBE** that contains all InfoProviders in the SAP BW system. This catalog is shown as a node in the navigator of Power Query. By expanding this node in the navigator, you can select from the available InfoProviders in the SAP BW system.

The other catalogs represent InfoProviders for which at least one Query exists. By expanding one of these nodes in the navigator, you can select from the available queries associated with the InfoProvider.

BEx Queries offer some advantages and additional functionality to create customized data sources to meet end-user requirements. For example, you can parameterize queries with variables that can limit the data set to what's important to the end user. Or, you can recalculate key figures using formulas.

Although BEx Queries have advantages as data sources (go to [Performance considerations](#)), you don't need a Query for every report. You'll need to weigh the cost of developing and maintaining additional Queries against their reporting requirements.

Data objects in SAP BW

SAP BW comes with built-in tools for creating data models based on different data objects. It's helpful to have a rudimentary understanding of how data is represented in SAP BW and the terminology. The main data objects in SAP BW are briefly introduced here:

- **InfoProvider** is the generic term for a Business Intelligence (BI) object into which data is loaded or which provides views of data. InfoProviders can be queried with client tools, such as Business Explorer (or BEx) and also with Power Query.

InfoProviders can be seen as uniform data providers from the viewpoint of a query definition. Their data can therefore be analyzed in a uniform way.

- **InfoCube** is a type of InfoProvider. An InfoCube describes, from an analysis point of view, a self-contained dataset, for a business-orientated area, for example Purchasing. You can analyze an InfoCube directly as an InfoProvider with analysis and reporting tools, including Power BI or Power Platform apps.

An InfoCube consists of a set of relational tables that are arranged according to an enhanced star schema. This means there's a (large) fact table that contains the key figures for the InfoCube, and also several (smaller) dimension tables that surround it.

- **Key figure** is an operational attribute that indicates a numerical measure such as amount, weight, quantity, and so on.
- **Dimension** is a grouping of related **characteristics** under a single generic term. For example, the *Customer* dimension could be made up of the *Customer Number*, the *Customer Group*, and the levels of the customer hierarchy.

A *Sales* dimension could contain the characteristics *Sales Person*, *Sales Group*, and *Sales Office*.

A *Time* dimension could have the characteristics *Day* (in the form YYYYMMDD), *Week* (in the form YYYY.WW), *Month* (in the form YYYY.MM), *Year* (in the form YYYY) and *Fiscal Period* (in the form YYYY.PPP).

- **Characteristics** refer to master data with their **attributes** and **text descriptions**, and in some cases **hierarchies**. The characteristics of an InfoCube are stored in dimensions.

For example, the *Customer* dimension could have the characteristics *Sold-to-party*, *Ship-to-party*, and *Payer*.

The characteristic *Sold-to-party* could have the attributes *Country*, *Region*, *City*, *Street*, and *Industry*. The text description of the characteristic would be the *Name* of the *Sold-to-party*.

In MDX query terms, the attributes of characteristics are also referred to as **properties**.

- **InfoObjects** is the generic term for all characteristics and key figures. All InfoObjects are maintained independently of the InfoCube in SAP BW. InfoObjects are the smallest units of Business Intelligence (BI). Using InfoObjects, information can be stored and mapped in a structured form. This is required for constructing InfoProviders. InfoObjects with attributes or texts can themselves be InfoProviders.
- **DataStore Object (DSO)** serves as a storage location for consolidated and cleansed transaction data or master data on a document (atomic) level. Unlike the multidimensional data in InfoCubes, the data in DataStore objects is stored in transparent, flat database tables. The system doesn't create separate fact tables or dimension tables for DSOs. Data in DSOs can be evaluated using a BEx query.
- **MultiProviders** are a special type of InfoProvider that combine data from several InfoProviders. They're then available for reporting. MultiProviders don't contain any data, their data comes exclusively from the InfoProviders upon which they're based. MultiProviders can be based upon any combination of InfoProviders, including InfoCubes, DataStore Objects, InfoObjects, or InfoSets.
- **InfoSets** are a special type of InfoProvider that doesn't store data physically. InfoSets describe data that's based on joining the tables of other InfoProviders like DataStore Objects, standard InfoCubes, or InfoObjects with master data characteristics. InfoSets can be useful when you have to build a report spanning two or more different data targets in SAP BW.

Composite Providers are a new data object in SAP BW systems that run on HANA, that is, SAP BW 7.5 or BW4/HANA. A composite provider is based on a JOIN or UNION of other InfoProviders or Analytic Indexes. Data in Composite Providers can be evaluated using a BEx query.

See also

- [Navigate the query objects](#)

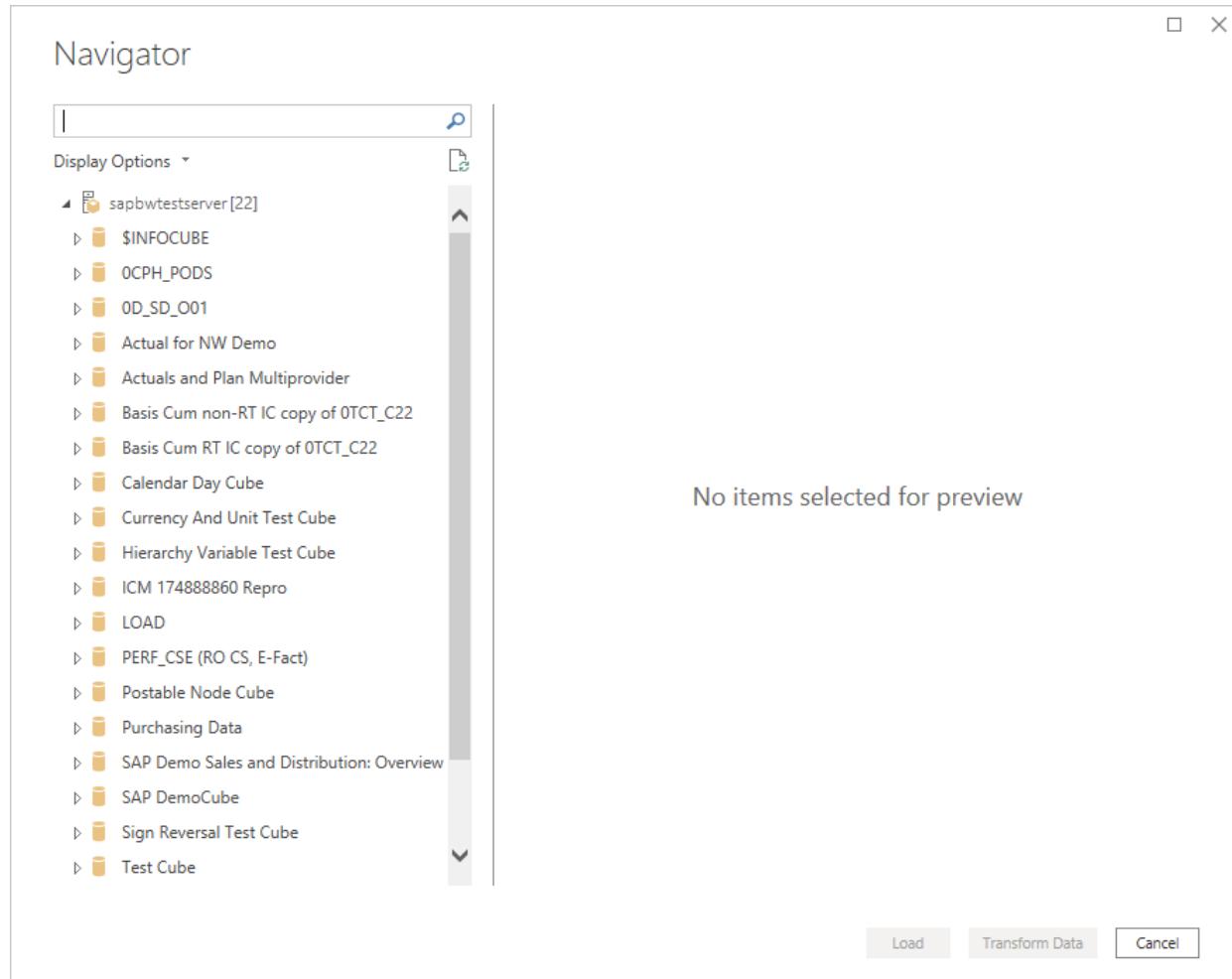
Navigate the query objects

1/15/2022 • 6 minutes to read • [Edit Online](#)

After you connect to your SAP BW instance, the **Navigator** dialog box will show a list of available catalogs in the selected server.

You'll see one catalog folder with the name **\$INFOCUBE**. This folder contains all InfoProviders in the SAP BW system.

The other catalog folders represent InfoProviders in SAP BW for which at least one query exists.



The **Navigator** dialog box displays a hierarchical tree of data objects from the connected SAP BW system. The following table describes the types of objects.

SYMBOL	DESCRIPTION
	Your SAP BW server
	Catalog—either \$INFOCUBE or an InfoProvider

SYMBOL	DESCRIPTION
	InfoCube or a BEx Query
	Key figure
	Characteristic
	Characteristic level
	Property (Attribute)
	Hierarchy

Navigator

Display Options ▾

- Only selected items
- Enable data previews
- Technical names

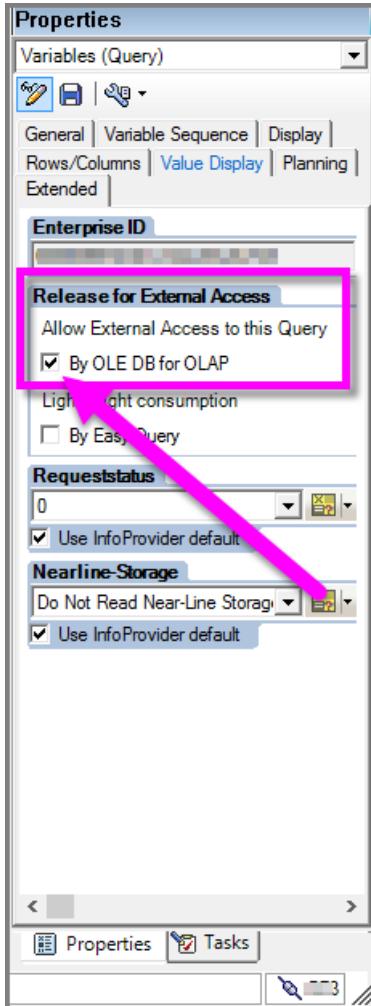
- ▷ Calendar Year
- ▷ Calendar Year/Month
- ▲ Country [4]
 - Country Level 01
 - ▷ Continent
 - ▲ PM_Country [3]
 - Level 01
 - Level 02
 - Level 03
 - ▷ Properties [3]
- ▲ Material [2]
 - Material Level 01
 - ▲ Properties [5]
 - Key
 - Material group (Key)
 - Material group (Name)
 - Material Level 01.UniqueName

Purchasing Overview

Country.Level 02	Country.Level 03	Material.Material Level 01	Material
EUROPE	Germany	Casing Notebook Speedy I CN	
EUROPE	Germany	Motherboard Notebook Speedy I CN	
EUROPE	Germany	Processor Notebook Speedy I CN	
EUROPE	Germany	Casing Notebook Speedy II CN	
EUROPE	Germany	Motherboard Notebook Speedy II CN	
EUROPE	Germany	Processor Notebook Speedy II CN	
EUROPE	Germany	Casing Terminal P400 CN	
EUROPE	Germany	Motherboard Terminal P400 CN	
EUROPE	Germany	Harddrive Terminal P400 CN	
EUROPE	Germany	Casing Terminal P600 CN	
EUROPE	Germany	Motherboard Terminal P600 CN	
EUROPE	Germany	Harddrive Terminal P600 CN	
EUROPE	Germany	Casing Monitor flat 17 CN	
EUROPE	Germany	Picture Tube Monitor flat 17 CN	
EUROPE	Germany	Logic Monitor flat 17 CN	
EUROPE	Germany	Casing Monitor flat 21 CN	
EUROPE	Germany	Picture Tube Monitor flat 21 CN	
EUROPE	Germany	Logic Monitor flat 21 CN	
EUROPE	France	Casing Notebook Speedy I CN	
EUROPE	France	Motherboard Notebook Speedy I CN	
EUROPE	France	Processor Notebook Speedy I CN	
EUROPE	France	Casing Notebook Speedy II CN	
EUROPE	France	Motherboard Notebook Speedy II CN	

NOTE

The navigator shows InfoCubes and BEx queries. For BEx queries, you may need to go into Business Explorer, open the desired query and check **Allow External Access to this Query: By OLE DB for OLAP** for the query to be available in the navigator.



NOTE

In Power BI Desktop, objects below an InfoCube or BEx Query node, such as the key figures, characteristics, and properties are only shown in Import connectivity mode, not in DirectQuery mode. In DirectQuery mode, all the available objects are mapped to a Power BI model and will be available for use in any visual.

In the navigator, you can select from different display options to view the available query objects in SAP BW:

- **Only selected items:** This option limits the objects shown in the list to just the selected items. By default, all query objects are displayed. This option is useful for a review of the objects that you included in your query. Another approach to viewing selected items is to select the column names in the preview area.
- **Enable data previews:** This value is the default. This option allows you to control whether a preview of the data should be displayed on the right-hand side in the **Navigator** dialog box. Disabling data previews reduces the amount of server interaction and response time. In Power BI Desktop, data preview is only available in Import connectivity mode.
- **Technical names:** SAP BW supports the notion of *technical names* for query objects, as opposed to the descriptive names that are shown by default. Technical names uniquely identify an object within SAP BW.

With the option selected, the technical names will appear next to the descriptive name of the object.

Characteristic hierarchies

A characteristic will always have at least one characteristic level (Level 01), even when no hierarchy is defined on the characteristic. The **Characteristic Level 01** object contains all members for the characteristic as a flat list of values.

Characteristics in SAP BW can have more than one hierarchy defined. For those characteristics, you can only select one hierarchy or the **Level 01** object.

For characteristics with hierarchies, the properties selected for that characteristic will be included for each selected level of the hierarchy.

The screenshot shows the SAP BW Navigator interface. On the left, the 'Display Options' tree view is expanded to show 'Country [4]' with its children: 'Country Level 01', 'Continent', 'PM_Country [3]' (which has 'Level 01', 'Level 02', and 'Level 03' checked), and 'Properties [3]' (with 'Name' checked). Other nodes like 'Material [2]' are collapsed. To the right, a table titled 'Purchasing Overview' displays data for 'Country.Level 01' and 'Country.Level 02'. The table has four columns: 'Country.Level 01', 'Country.Level 02', 'Country.Level 01.Name', and 'Country.Level 02.Name'. Two rows are shown: one for 'REGION' with 'EUROPE' in both columns, and another for 'REGION' with 'AMERICA' in both columns. At the bottom are 'Load', 'Transform Data', and 'Cancel' buttons.

Measure properties

When you pick a measure, you have an option to select the units/currency, formatted value, and format string. In the screenshot below, it's useful to get the formatted value for COGS. This helps us follow the same formatting standard across all the reports.

NOTE

Measure properties are not available in Power BI Desktop in DirectQuery mode.

The screenshot shows the SAP BW Navigator interface. On the left, the 'Display Options' tree view is expanded to show 'Accounting Freight Cost - COGS [24]' with its children: 'COGS' (checked), 'Gross Weight', 'Net Weight', and 'Measure Properties [3]' (with 'Format string', 'Formatted Value', and 'Unit of measure' checked). To the right, a table titled 'Accounting Freight Cost - COGS' displays data for 'COGS'. The table has four columns: 'COGS', 'COGS.Format string', 'COGS.Formatted Value', and 'COGS.Unit of measure'. One row is shown: '16099490278' with '#,##0.00' in the format string and '16,099,490,277.91' in the formatted value. At the top right are 'Apply' and 'Clear' buttons.

Flattening of multi-dimensional data

Based on the selected objects and properties in the navigator, Power Query constructs an MDX statement that is sent for execution to SAP BW. The MDX statement returns a flattened data set that can be further manipulated using the Power Query Editor.

Power Query uses a newer interface that is available in SAP BW version 7.01 or higher. The interface reduces memory consumption and the result set is not restricted by the number of cells.

The flattened data set is aggregated in SAP BW at the level of the selected characteristics and properties.

Even with these improvements, the resulting dataset can become very large and time-consuming to process.

Performance recommendation

Only include the characteristics and properties that you ultimately need. Aim for higher levels of aggregation, that is, do you need Material-level details in your report, or is MaterialGroup-level enough? What hierarchy levels are required in Power BI? Try to create smaller datasets, with higher levels of aggregation, or multiple smaller datasets, that can be joined together later.

Query parameters

Queries in SAP BW can have dynamic filters defined that allow you to restrict the data set that's returned by the query. In the BEx Query Designer, this type of dynamic filter can be defined with what's called a *Characteristic Restriction* and assigning a *Variable* to that restriction. Variables on a query can be required or optional, and they're available to the user in the navigator.

When you select an SAP BW query with characteristic restrictions in the Power Query navigator, you'll see the variables displayed as parameters above the data preview area.

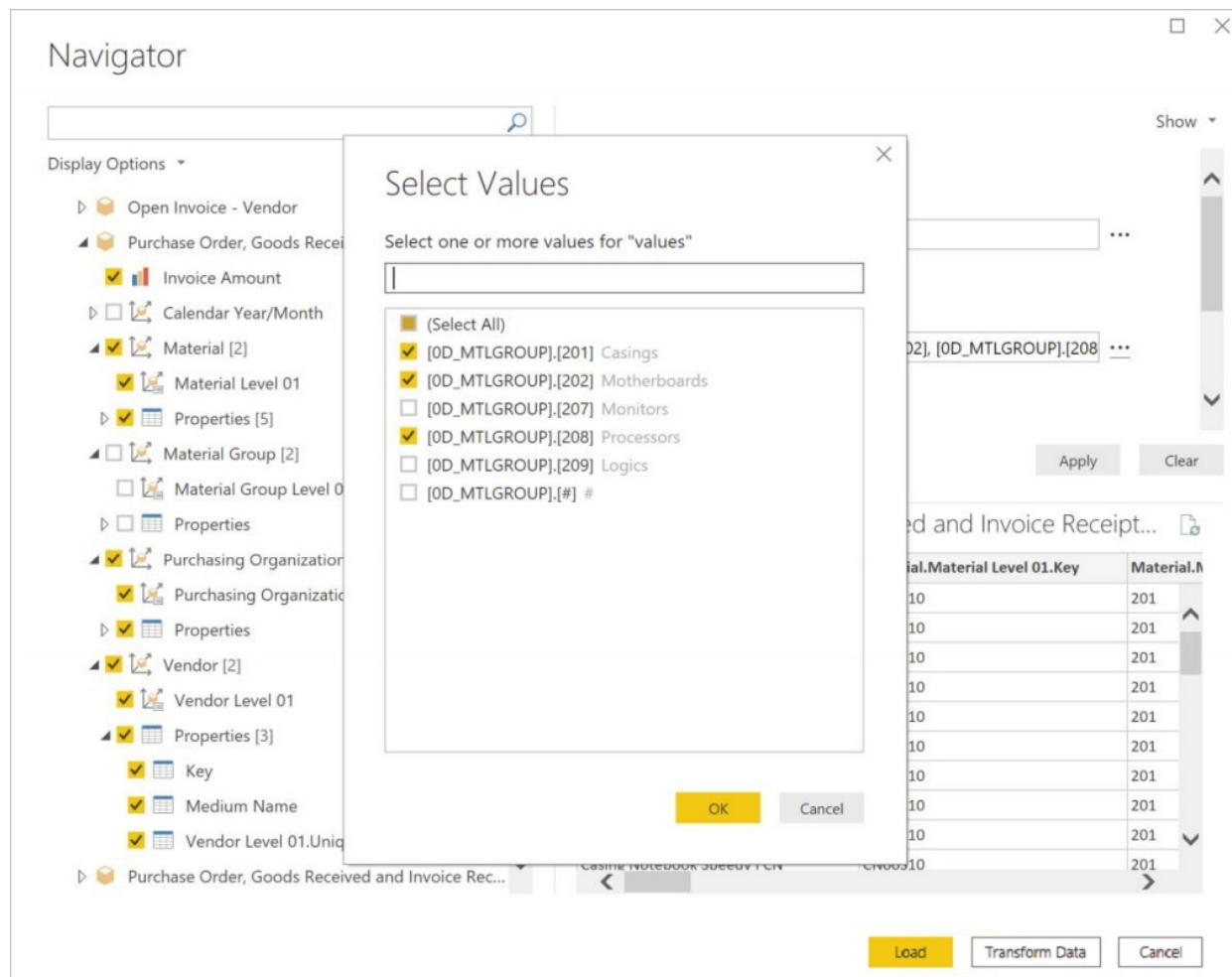
The screenshot shows the SAP BW Power Query Navigator window. On the left, the 'Navigator' pane lists various objects and properties under 'Display Options'. Several items are checked, including 'Purchase Order, Goods Received and Invoice Rec...', 'Invoice Amount', 'Material [2]', 'Material Level 01', 'Properties [5]', 'Material Group [2]', 'Material Group Level 01', 'Properties', 'Purchasing Organization [2]', 'Vendor [2]', 'Vendor Level 01', 'Properties [3]', 'Key', 'Medium Name', and 'Vendor Level 01.UniqueName'. In the center, there are three input fields labeled 'Material (optional) values', 'Material Group (optional) values', and 'Calendar Month/Year (optional)'. Below these is a 'Show' dropdown menu. To the right, a preview table titled 'Purchase Order, Goods Received and Invoice Receipt...' displays data for 'Material.Material Level 01' and 'Material.Material Level 01.Key'. The table contains 15 rows of data, each with columns for 'Material.Material Level 01' (e.g., 'Casing Notebook Speedy I CN'), 'Material.Material Level 01.Key' (e.g., 'CN00510'), and 'Material.B' (e.g., '201'). At the bottom of the preview table are 'Load', 'Edit', and 'Cancel' buttons.

Using the **Show** selector, you can display all parameters that are defined on the query, or just the required ones.

The query shown in the previous image has several optional parameters, including one for **Material Group**. You can select one or more material groups to only return purchasing information for the selected values, that is, casings, motherboards, and processors. You can also type the values directly into the values field. For variables with multiple entries, comma-separated values are expected, in this example it would look like

[0D_MTLGROUP].[201], [0D_MTLGROUP].[202], [0D_MTLGROUP].[208] .

The value # means unassigned; in the example any data record without an assigned material group value.



Performance recommendation

Filters based on parameter values get processed in the SAP BW data source, not in Power BI. This type of processing can have performance advantages for larger datasets when loading or refreshing SAP BW data into Power BI. The time it takes to load data from SAP BW into Power BI increases with the size of the dataset, for example, the number of columns and rows in the flattened result set. To reduce the number of columns, only select the key figures, characteristics, and properties in the navigator that you eventually want to see.

Similarly, to reduce the number of rows, use the available parameters on the query to narrow the dataset, or to split up a larger dataset into multiple, smaller datasets that can be joined together in the Power BI Desktop data model.

In many cases, it may also be possible to work with the author of the BEx Query in SAP BW to clone and modify an existing query and optimize it for performance by adding additional characteristic restrictions or removing unnecessary characteristics.

Loading SAP data into Power Query

Once you've selected the SAP data set you want in the navigator, you can import the data into Power Query Editor. Select **Transform Data** to launch the Power Query Editor, where you can perform additional data transformation and filtering steps.

Screenshot of the Power Query Editor showing a query named "Purchase Order, Goods..." which uses a cube transformation to filter data. The query settings show a parameter "Purchase Order, Goods Received and Inv". Applied steps include "Source" and "Added Items".

	A ^B _C Material.Material Level 01	A ^B _C Material.Material Level 01.Material Level 01.UniqueName	A ^B _C Material.Material Level 01.Key	A ^B _C Material.Material Level 01.M
1	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
2	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
3	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
4	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
5	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
6	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
7	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
8	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
9	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
10	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
11	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
12	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
13	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
14	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
15	Casing Notebook Speedy I CN	[OD_MATERIAL].[CN00510]	CN00510	Casing Notebook Speedy I CN
16	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
17	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
18	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
19	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
20	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
21	Motherboard Notebook Speedy I CN	[OD_MATERIAL].[CN00511]	CN00511	Motherboard Notebook Spec
22				

15 COLUMNS, 205 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 9:56 AM

In the example above, a parameter was used to only bring back records with a **Material Group** of casings, motherboards, and processors.

In Power Query Desktop, you can also select **Load** to bring the entire data set from SAP BW into Power BI Desktop. Power BI Desktop will take you to the **Report** view where you can begin visualizing the data or make further modifications using the **Data** or **Relationships** views.

See also

- [Transform and filter an SAP BW dataset](#)

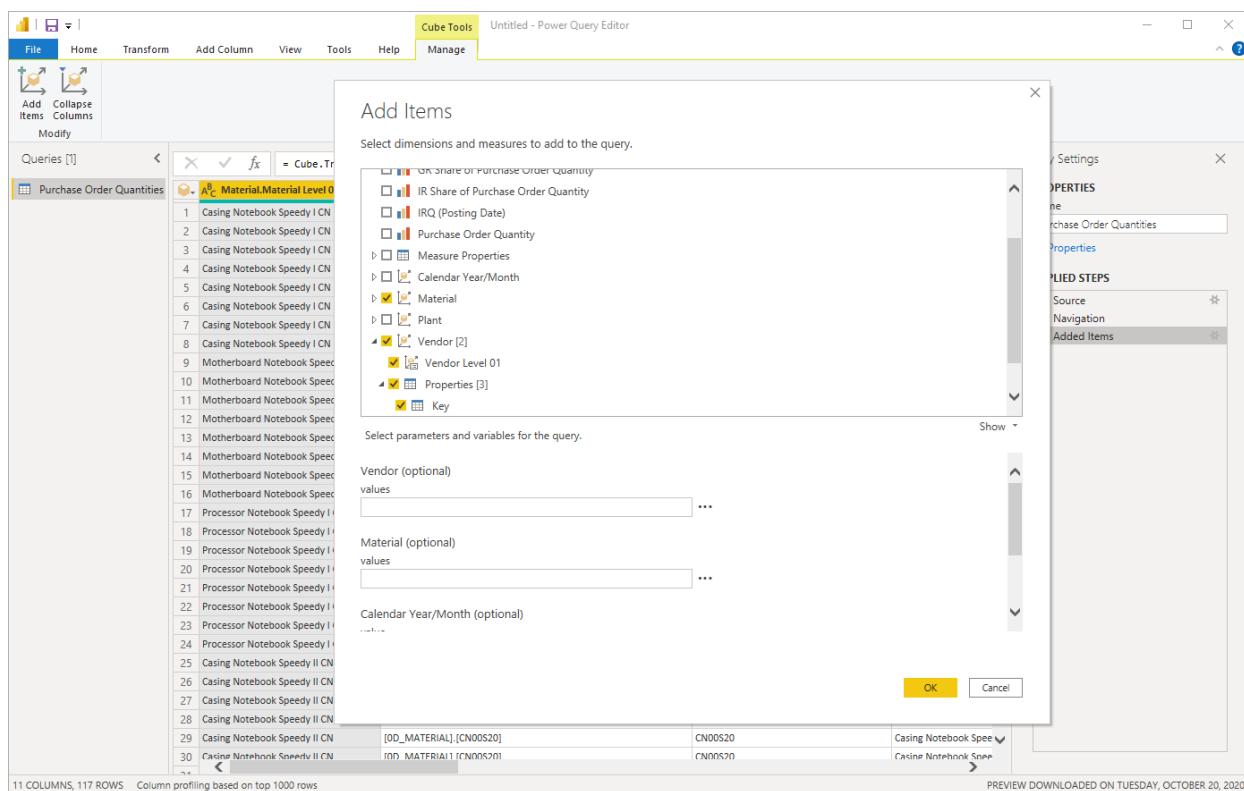
Transform and filter an SAP BW dataset

1/15/2022 • 2 minutes to read • [Edit Online](#)

With Power Query Editor, you can apply additional data transformations and filtering steps before you bring the dataset from SAP BW into the Power BI Desktop or Microsoft Power Platform data model.

In Power Query Editor, the *Applied Steps* for the query are shown in the **Query Settings** pane on the right. To modify or review a step, select the gear icon next to a step.

For example, if you select the gear icon next to **Added Items**, you can review the selected data objects in SAP BW, or modify the specified query parameters. This way it's possible to filter a dataset using a characteristic that isn't included in the result set.



You can apply additional filters on the dataset by selecting the drop-down menu for one of the columns.

The screenshot shows the Power Query Editor interface with the following details:

- File**, **Home**, **Transform**, **Add Column**, **View**, **Help**, **Cube Tools** (selected), **Untitled - Power Query Editor**.
- Queries [1]**: Sales and Distribution: Overview.
- Table Headers**: A[1] Calendar Year/Month, A[2] Country.Level 01, A[3] Country.Level 02, A[4] Country.Level 03, A[5] 1.2 Cost stats currency, A[6] 1.2 Gross weight.
- Applied Steps** pane:
 - PROPERTIES**: Name = Sales and Distribution: Overview, All Properties.
 - APPLIED STEPS**: Source, Navigation, **Added Items** (highlighted).
- Context Menu** (over the first row):
 - (Select All)
 - APR 2003
 - APR 2004
 - AUG 2003
 - AUG 2004
 - DEC 2003
 - DEC 2004
 - FEB 2003
 - FEB 2004
 - JAN 2003
 - JAN 2004
 - JUL 2003
 - JUL 2004
 - JUN 2003
 - JUN 2004
 - MAR 1030
 - MAR 2003
 - MAJ 2004
- Table Data Preview** (first few rows):

1	MAR 1030	Not Assigned Country (s)	Australia	4268607	
2	JAN 2003	REGION	Germany	49328079	
3	JAN 2003	REGION	France	17969040	
4	JAN 2003	REGION	England	24809029	
5	JAN 2003	REGION	USA	26032927	
6	JAN 2003	REGION	Canada	6451493	
7	FEB 2003	REGION	Germany	28700489	
8	FEB 2003	REGION	France	7233747	
9	FEB 2003	REGION	England	12050663	
10	FEB 2003	AMERICA	USA	34116345	
11	FEB 2003	AMERICA	Canada	10304981	
12	MAR 2003	EUROPE	Germany	37991347	
13	MAR 2003	EUROPE	France	11000854	
14	MAR 2003	EUROPE	England	13851025	
15	MAR 2003	REGION	USA	40193811	
16	MAR 2003	REGION	Canada	9643892	
17	APR 2003	REGION	Germany	12026591	
18	APR 2003	REGION	France	9475899	
19	APR 2003	REGION	England	18365679	
20	APR 2003	REGION	USA	42272146	
21	APR 2003	REGION	Canada	6156023	
22	MAY 2003	REGION	Germany	16588484	
23	MAY 2003	REGION	France	12613687	
24	MAY 2003	REGION	England	22157663	
25	MAY 2003	REGION	USA	25350099	
26	MAY 2003	REGION	Canada	11126587	
27	JUN 2003	REGION	Germany	14826079	
28	JUN 2003	REGION	France	25962516	
29	JUN 2003	REGION	England	19604233	

Another easy way to set a filter is to right-click on one of the values in the table, then select **Member Filters** or **Text Filters**.

The screenshot shows the Power Query Editor interface with the following details:

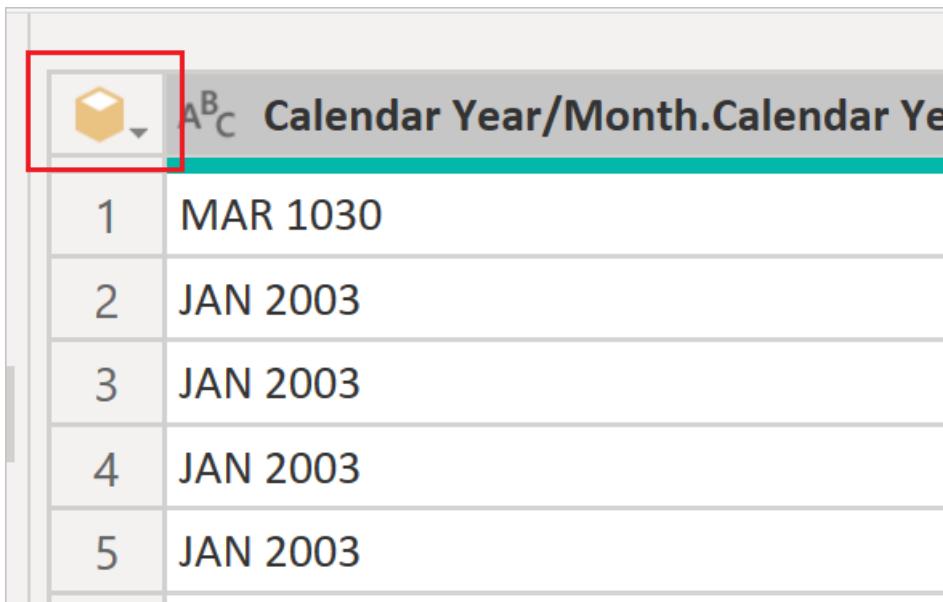
- File**, **Home**, **Transform**, **Add Column**, **View**, **Help**, **Cube Tools** (selected), **Untitled - Power Query Editor**.
- Queries [1]**: Sales and Distribution: Overview.
- Table Headers**: A[1] Calendar Year/Month, A[2] Country.Level 01, A[3] Country.Level 02, A[4] Country.Level 03, A[5] 1.2 Cost stats currency, A[6] 1.2 Gross weight.
- Applied Steps** pane:
 - PROPERTIES**: Name = Sales and Distribution: Overview, All Properties.
 - APPLIED STEPS**: Source, Navigation, **Added Items** (highlighted).
- Context Menu** (over the first row):
 - (Select All)
 - APR 2003
 - APR 2004
 - AUG 2003
 - AUG 2004
 - DEC 2003
 - DEC 2004
 - FEB 2003
 - FEB 2004
 - JAN 2003
 - JAN 2004
 - JUL 2003
 - JUL 2004
 - JUN 2003
 - JUN 2004
 - MAR 1030
 - MAR 2003
 - MAJ 2004
- Table Data Preview** (first few rows):

1	MAR 1030	Not Assigned Country (s)	Australia	4268607	
2	JAN 2003	REGION	Germany	49328079	
3	JAN 2003	REGION	France	17969040	
4	JAN 2003	REGION	England	24809029	
5	JAN 2003	REGION	USA	26032927	
6	JAN 2003	REGION	Canada	6451493	
7	FEB 2003	REGION	Germany	28700489	
8	FEB 2003	REGION	France	7233747	
9	FEB 2003	REGION	England	12050663	
10	FEB 2003	AMERICA	USA	34116345	
11	FEB 2003	AMERICA	Canada	10304981	
12	MAR 2003	EUROPE	Germany	37991347	
13	MAR 2003	EUROPE	France	11000854	
14	MAR 2003	EUROPE	England	13851025	
15	MAR 2003	REGION	USA	40193811	
16	MAR 2003	REGION	Canada	9643892	
17	APR 2003	REGION	Germany	12026591	
18	APR 2003	REGION	France	9475899	
19	APR 2003	REGION	England	18365679	
20	APR 2003	REGION	USA	42272146	
21	APR 2003	REGION	Canada	6156023	
22	MAY 2003	REGION	Germany	16588484	
23	MAY 2003	REGION	France	12613687	
24	MAY 2003	REGION	England	22157663	
25	MAY 2003	REGION	USA	25350099	
26	MAY 2003	REGION	Canada	11126587	
27	JUN 2003	REGION	Germany	14826079	
28	JUN 2003	REGION	France	25962516	
29	JUN 2003	REGION	England	19604233	

For example, you could filter the dataset to only include records for Calendar Year/Month FEB 2003, or apply a text filter to only include records where Calendar Year/Month contains 2003.

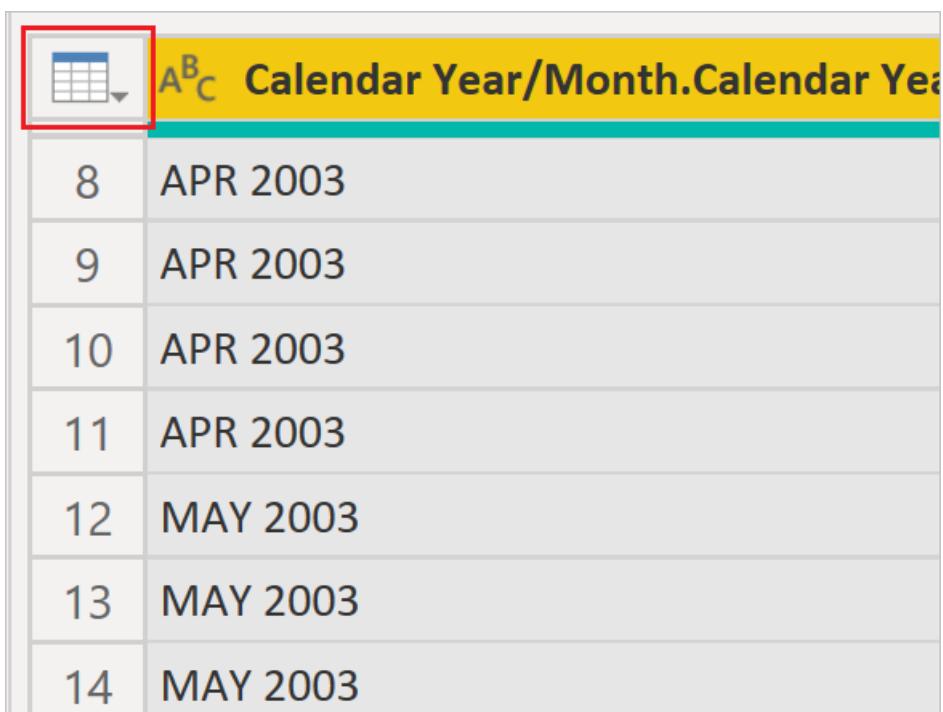
Not every filter will get folded into the query against SAP BW. You can determine if a filter is folded into the query by examining the icon in the top-left corner of the data table, directly above the number 1 of the first data record.

If the icon is a cube, then the filter is applied in the query against the SAP BW system.



	A ^B C	Calendar Year/Month.Calendar Ye
1	MAR	1030
2	JAN	2003
3	JAN	2003
4	JAN	2003
5	JAN	2003

If the icon is a table, then the filter isn't part of the query and only applied to the table.



	A ^B C	Calendar Year/Month.Calendar Ye
8	APR	2003
9	APR	2003
10	APR	2003
11	APR	2003
12	MAY	2003
13	MAY	2003
14	MAY	2003

Behind the UI of Power Query Editor, code is generated based on the M formula language for data mashup queries.

You can view the generated M code with the **Advanced Editor** option in the **View** tab.

```

let
    Source = SapBusinessWarehouse.Cubes("bw0.my-organization.com", "00", "800", [ScaleMeasures=false]),
    #> "00_PU_C01" = Source[[Name="00_PU_C01"]][Data],
    #> "00_PU_C01/00_PU_C01_Q0013" = "#00_PU_C01[[id='00_PU_C01/00_PU_C01_Q0013']][Data],
    #> "Added Items" = Cube.Transform("#00_PU_C01/00_PU_C01_Q0013",
        {
            {Cube.ApplyParameter, "[00_MTLGRP]", {[["00_MTLGROUP].[201]", "[00_MTLGROUP].[202]", "[00_MTLGROUP].[208"]]}},
            {Cube.AddAndExpandDimensionColumn, "[00_MATERIAL]", {[00_MATERIAL].[LEVEL01]}, {"Material.Material Level 01"}},
            {Table.AddColumn, "Material.Material Level 01.UniqueName", each Cube.AttributeMemberProperty([Material.Material Level 01.Key]), {Table.AddColumn, "Material.Material Level 01.Key", each Cube.AttributeMemberProperty([Material.Material Level 01.Key])}},
            {Table.AddColumn, "Material.Material Level 01.Medium Name", each Cube.AttributeMemberProperty([Material.Material Level 01.Medium Name]), {Table.AddColumn, "Material.Material Level 01.Medium Name", each Cube.AttributeMemberProperty([Material.Material Level 01.Medium Name])}},
            {Table.AddColumn, "Material.Material Level 01.Material group (Key)", each Cube.AttributeMemberProperty([Material.Material Level 01.Material group (Key)]), {Table.AddColumn, "Material.Material Level 01.Material group (Key)", each Cube.AttributeMemberProperty([Material.Material Level 01.Material group (Key)])}},
            {Table.AddColumn, "Material.Material Level 01.Purchasing Organization", each Cube.AttributeMemberProperty([Material.Material Level 01.Purchasing Organization]), {Table.AddColumn, "Material.Material Level 01.Purchasing Organization", each Cube.AttributeMemberProperty([Material.Material Level 01.Purchasing Organization])}},
            {Table.AddColumn, "Purchasing Organization.Purchasing Organization Level 01.Key", each Cube.AttributeMemberProperty([Purchasing Organization.Level01]), {Table.AddColumn, "Purchasing Organization.Purchasing Organization Level 01.Key", each Cube.AttributeMemberProperty([Purchasing Organization.Level01])}},
            {Table.AddColumn, "Purchasing Organization.Purchasing Organization Level 01.Medium Name", each Cube.AttributeMemberProperty([Purchasing Organization.Level01.Medium Name]), {Table.AddColumn, "Purchasing Organization.Purchasing Organization Level 01.Medium Name", each Cube.AttributeMemberProperty([Purchasing Organization.Level01.Medium Name])}},
            {Table.AddColumn, "Vendor.Vendor Level 01.Key", each Cube.AttributeMemberProperty([Vendor.Vendor Level 01.Key]), {Table.AddColumn, "Vendor.Vendor Level 01.Key", each Cube.AttributeMemberProperty([Vendor.Vendor Level 01.Key])}},
            {Table.AddColumn, "Vendor.Vendor Level 01.Medium Name", each Cube.AttributeMemberProperty([Vendor.Vendor Level 01.Medium Name]), {Table.AddColumn, "Vendor.Vendor Level 01.Medium Name", each Cube.AttributeMemberProperty([Vendor.Vendor Level 01.Medium Name])}},
            {Table.AddColumn, "Invoice Amount", "[Measures].[#EFUZH0P10X72MBPOVBYISwN]"}
        })
in
    "Added Items"

```

No syntax errors have been detected.

Done Cancel

To see a description for each function or to test it, right-click on the existing SAP BW query in the **Queries** pane and select **Create Function**. In the formula bar at the top, enter:

= <*function name*>

where <*function name*> is the name of the function you want to see described. The following example shows the description of the `Cube.Transform` function.

Queries [1]

fx Func

= `Cube.Transform`

`Cube.Transform`

Applies the list cube functions, `transforms`, on the `cube`.

Enter Parameters

`cube`

`transforms`

Unspecified

Choose Column...

Invoke Clear

function (cube as table, transforms as list) as table

The following examples show some descriptions of various cube functions:

- `Cube.Transform` : Applies the list of functions or transforms on the cube.
- `Cube.ApplyParameter` : Applies the specified values for a parameter.
- `Cube.DisplayFolders` : Returns a nested tree of objects representing the display folder hierarchy of the cube.
- `Cube.Parameters` : Returns a table with the set of parameters that can be applied to the cube.
- `Cube.Dimensions` : Returns a table with the set of dimensions for the cube.
- `Cube.Measures` : Returns a table with the set of measures for the cube.

See also

- [Power Query M formula language reference](#)
- [Implementation details](#)

Implementation details

1/15/2022 • 11 minutes to read • [Edit Online](#)

This article describes conversion information and specific features available in Implementation 2 of the Power Query SAP Business Warehouse connector.

New options for Implementation 2.0

Implementation 2.0 supports the following options:

- *ExecutionMode* specifies the MDX interface used to execute queries on the server. The following options are valid:
 - `SapBusinessWarehouseExecutionMode.BasXml`
 - `SapBusinessWarehouseExecutionMode.BasXmlGzip`
 - `SapBusinessWarehouseExecutionMode.DataStream`

The default value is `SapBusinessWarehouseExecutionMode.BasXmlGzip`.

Using `SapBusinessWarehouseExecutionMode.BasXmlGzip` may improve performance when experiencing high latency for large datasets.

- *BatchSize* specifies the maximum number of rows to retrieve at a time when executing an MDX statement. A small number translates into more calls to the server while retrieving a large dataset. A large number of rows may improve performance, but could cause memory issues on the SAP BW server. The default value is 50000 rows.
- *EnableStructures* indicates whether characteristic structures are recognized. The default value for this option is false. Affects the list of objects available for selection. Not supported in Native query mode.

The *ScaleMeasures* option has been deprecated in this implementation. The behavior is now the same as setting *ScaleMeasures* to false, always showing unscaled values.

Additional improvements for Implementation 2.0

The following list describes some of the additional improvements that come with the new implementation:

- Improved performance.
- Ability to retrieve several million rows of data, and fine-tuning through the batch size parameter.
- Ability to switch execution modes.
- Support for compressed mode. Especially beneficial for high latency connections or large datasets.
- Improved detection of `Date` variables.
- Expose `Date` (ABAP type DATS) and `Time` (ABAP type TIMS) dimensions as dates and times respectively, instead of text values. More information: [Support for typed dates in SAP BW](#)
- Better exception handling. Errors that occur in BAPI calls are now surfaced.
- Column folding in BasXml and BasXmlGzip modes. For example, if the generated MDX query retrieves 40 columns but the current selection only needs 10, this request will be passed onto the server to retrieve a smaller dataset.

Changing existing reports to use Implementation 2.0

Changing existing reports to use Implementation 2.0 is only possible in import mode. Follow these steps:

1. Open an existing report, select **Edit Queries** in the ribbon, and then select the SAP Business Warehouse query to update.
2. Right-click the query and select **Advanced Editor**.
3. In the **Advanced Editor**, change the `SapBusinessWarehouse.Cubes` call as follows:

Determine whether the query already contains an option record, such as the following example.

```
Source = SapBusinessWarehouse.Cubes("someserver", "00", "900",
[ScaleMeasures=false])

Source = SapBusinessWarehouse.Cubes("sapbw73", "00", "900", [Query="SELECT {
[Measures].[006EI86RZI4D2RTX6HQP12JIY] }ON 0, NON EMPTY CROSSJOIN ... ON 1 FROM
[0D_SD_C03/REP_20180117030589]"])
```

If so, add the `Implementation 2.0` option, and remove the `ScaleMeasures` option, if present, as shown.

```
Source = SapBusinessWarehouse.Cubes("someserver", "00", "900",
[Implementation="2.0"])

Source = SapBusinessWarehouse.Cubes("sapbw73", "00", "900",
[Implementation="2.0", Query="SELECT {
[Measures].[006EI86RZI4D2RTX6HQP12JIY] }ON 0, NON EMPTY CROSSJOIN ... ON 1 FROM
[0D_SD_C03/REP_20180117030589]"])
```

If the query doesn't already include an options record, just add it. For the following option:

```
Source = SapBusinessWarehouse.Cubes("someserver", "00", "900")
```

Just change it to:

```
Source = SapBusinessWarehouse.Cubes("someserver", "00", "900",
[Implementation="2.0"])
```

Every effort has been made to make Implementation 2.0 of the SAP BW connector compatible with version 1. However, there may be some differences because of the different SAP BW MDX execution modes being used. To resolve any discrepancies, try switching between execution modes.

Support for typed dates in SAP BW

Implementation 2.0 of the SAP BW connector includes support for typed dates and times. If you query a report that has dimensions with ABAP types, DATS, or TIMS, they can now be output as dates instead of text.

The limitations for using this functionality are:

- Only available in Implementation 2.0 of the SAP BW connector.
- Only available in Import mode.
- The account used to connect to the SAP BW server should have enough permissions to call `BAPI_IOBJ_GETDETAIL`.

```

let
    Source = SapBusinessWarehouse.Cubes("sapbwtestserver", "00", "837",
[ExecutionMode=SapBusinessWarehouseExecutionMode.BasXmlGzip, Implementation="2.0"]),
    #"$INFOCUBE" = Source{[Name="$INFOCUBE"]}[Data],
    #"$0D_DECU" = #"$INFOCUBE"{[Id="$0D_DECU"]}[Data],
    #"Added Items" = Cube.Transform(#"$0D_DECU",
    {
        {Cube.AddAndExpandDimensionColumn, "[0CALDAY]", {[0CALDAY].[LEVEL01]}, {"Calendar day.Calendar day Level 01"}},
        {Table.AddColumn, "Calendar day.Calendar day Level 01.Key", each
            Cube.AttributeMemberProperty([Calendar day.Calendar day Level 01], "[20CALDAY"]),
            {Cube.AddMeasureColumn, "Billed Quantity", "[Measures].[0D_INV_QTY]"}
        })
    in
        #"Added Items"

```

You'll need to add the key in to access the typed date. For example, if there's a dimension attribute called [0CALDAY], you'll need to add the key [20CALDAY] to get the typed value.

In the example above, this means that:

- Calendar day.Calendar day Level 01 [0CALDAY] will be text (a caption). (Added by default when the dimension is added.)
- Calendar day.Calendar day Level 01.Key [20CALDAY] will be a date (must be manually selected).

To manually add the key in Import mode, just expand **Properties** and select the key.

Add Items

Select dimensions and measures to add to the query.

The screenshot shows the 'Add Items' dialog with a tree view of available dimensions and measures. Under the 'Properties' node, the 'Key' column is selected and highlighted with a red border. Other visible items include 'Billed Quantity', 'Costs (SAP Demo)', 'Net Value', 'Tax Amount', 'Cal. Year/Quarter', 'Calendar day [2]', 'Calendar day Level 01', 'Calendar day Level 01.UniqueName', and several other dimension nodes like 'Calendar Year', 'Company code', etc.

The key column will be of type date, and can be used for filtering. Filtering on this column will fold to the server.

Support for SAP BW features

The following table lists all SAP BW features that aren't fully supported or will behave differently when using the Power Query SAP BW connector.

FEATURE	DESCRIPTION
Local calculations	<p>Local calculations defined in a BEX Query will change the numbers as displayed through tools like Bex Analyzer. However, they aren't reflected in the numbers returned from SAP through the public MDX interface.</p> <p>As such, the numbers seen in Power Query won't necessarily match those for a corresponding visual in an SAP tool.</p> <p>For instance, when connecting to a query cube from a BEx query that sets the aggregation to be Cumulated (for example, running sum), Power Query would get back the base numbers, ignoring that setting. An analyst could then apply a running sum calculation locally in, for example, Power BI, but would need to exercise caution in how the numbers are interpreted if this isn't done.</p>
Aggregations	<p>In some cases (particularly when dealing with multiple currencies), the aggregate numbers returned by the SAP public interface don't match those shown by SAP tools.</p> <p>As such, the numbers seen in Power Query won't necessarily match those for a corresponding visual in an SAP tool.</p> <p>For instance, totals over different currencies would show as "##" in Bex Analyzer, but the total would get returned by the SAP public interface, without any information that such an aggregate number is meaningless. Thus the number (aggregating, say, \$, EUR, and AUD) would get displayed by Power Query.</p>
Currency formatting	<p>Any currency formatting (for example, \$2,300 or 4000 AUD) isn't reflected in Power Query.</p>
Units of measure	<p>Units of measure (for example, 230 KG) aren't reflected in Power Query.</p>
Key versus text (short, medium, long)	<p>For an SAP BW characteristic like CostCenter, the navigator will show a single item Cost Center Level 01. Selecting this item will include the default text for Cost Center in the field list. Also, the Key value, Short Name, Medium Name, and Long Name values are available for selection in the Properties node for the characteristic (if maintained in SAP BW).</p> <p>Note that this only applies to Import connectivity mode. For DirectQuery mode, only the default text will be included in the data set.</p>
Attributes	<p>The attributes of a characteristic will be available for selection in the Properties for the characteristic. This only applies to Import connectivity mode. For DirectQuery mode, attributes won't be available.</p>

FEATURE	DESCRIPTION
Multiple hierarchies of a characteristic	<p>In SAP a characteristic can have multiple hierarchies. Then in tools like BEx Analyzer, when a characteristic is included in a query, the user can select the hierarchy to use.</p> <p>In Power BI, the various hierarchies can be seen in the field list as different hierarchies on the same dimension. However, selecting multiple levels from two different hierarchies on the same dimension will result in empty data being returned by SAP.</p>
Treatment of ragged hierarchies	<p>SAP BW supports ragged hierarchies, where levels can be missed, for example:</p> <pre style="margin-left: 40px;">Continent Americas Canada USA Not Assigned Australia</pre> <p>In Power BI, this appears with (Blank) at the missing level:</p> <pre style="margin-left: 40px;">Continent Americas Canada USA Not Assigned (Blank) Australia</pre>
Scaling factor/reverse sign	<p>In SAP, a key figure can have a scaling factor (for example, 1000) defined as a formatting option, meaning that all displays will be scaled by that factor.</p> <p>It can similarly have a property set that reverses the sign. Use of such a key figure in Power BI (in a visual, or as part of a calculation) will result in the unscaled number being used (and the sign isn't reversed). The underlying scaling factor isn't available. In Power BI visuals, the scale units shown on the axis (K,M,B) can be controlled as part of the visual formatting.</p>
Hierarchies where levels appear/disappear dynamically	<p>Initially when connecting to SAP BW, the information on the levels of a hierarchy will be retrieved, resulting in a set of fields in the field list. This is cached, and if the set of levels changes, then the set of fields doesn't change until Refresh is invoked.</p> <p>This is only possible in Power BI Desktop. Such a Refresh to reflect changes to the levels cannot be invoked in the Power BI service after Publish.</p>
Default filter	<p>A BEX query can include Default Filters, which will be applied automatically by SAP Bex Analyzer. These aren't exposed, and so the equivalent usage in Power Query won't apply the same filters by default.</p>

FEATURE	DESCRIPTION
Hidden Key figures	A BEx query can control visibility of Key Figures, and those that are hidden won't appear in SAP BEx Analyzer. This isn't reflected through the public API, and so such hidden key figures will still appear in the field list. However, they can then be hidden within Power Query.
Numeric formatting	Any numeric formatting (number of decimal positions, decimal point, and so on) won't automatically be reflected in Power Query. However, it's possible to then control such formatting within Power Query.
Hierarchy versioning	SAP BW allows different versions of a hierarchy to be maintained, for example, the cost center hierarchy in 2007 versus 2008. Only the latest version will be available in Power Query, as information on versions isn't exposed by the public API.
Time-dependent hierarchies	When using Power Query, time-dependent hierarchies are evaluated at the current date.
Currency conversion	SAP BW supports currency conversion, based on rates held in the cube. Such capabilities aren't exposed by the public API, and are therefore not available in Power Query.
Sort Order	The sort order (by Text, or by Key) for a characteristic can be defined in SAP. This sort order isn't reflected in Power Query. For example, months might appear as "April", "Aug", and so on. It isn't possible to change this sort order in Power Query.
Technical names	In the navigator, the characteristic/measure names (descriptions) and technical names can both be displayed using the Display Options selector. The field list contains the characteristic/measure names (descriptions).
End user language setting	The locale used to connect to SAP BW is set as part of the connection details, and doesn't reflect the locale of the final report consumer.
Text Variables	SAP BW allows field names to contain placeholders for variables (for example, "\$YEARS Actuals") that would then get replaced by the selected value. For example, the field appears as "2016 Actuals" in BEx tools, if the year 2016 were selected for the variable. The column name in Power Query won't be changed depending on the variable value, and so would appear as "\$YEARS Actuals". However, the column name can then be changed in Power Query.
Customer Exit Variables	Customer Exit variables aren't exposed by the public API, and are therefore not supported by Power Query.

Performance Considerations

The following table provides a summary list of suggestions to improve performance for data load and refresh from SAP BW.

SUGGESTION	DESCRIPTION
Limit characteristics and properties (attribute) selection	The time it takes to load data from SAP BW into Power Query increases with the size of the dataset, that is, the number of columns and rows in the flattened result set. To reduce the number of columns, only select the characteristics and properties in the navigator that you eventually want to see in your report or dashboard.
Make use of parameters	Using filters/parameters contributes to reducing the size of the result set, which significantly improves query runtimes. Parameters are especially valuable when used with large dimensions, where there's many members, such as customers, materials, or document numbers.
Limit number of key figures	Selecting many key figures from a BEx query/BW model can have a significant performance impact during query execution because of the time being spent on loading metadata for units. Only include the key figures that you need in Power Query.
Split up very large queries into multiple, smaller queries	For very large queries against InfoCubes or BEx queries, it may be beneficial to split up the query. For example, one query might be getting the key figures, while another query (or several other queries) is getting the characteristics data. You can join the individual query results in Power Query.
Avoid Virtual Providers (MultiProviders or InfoSets)	VirtualProviders are similar to structures without persistent storage. They are useful in many scenarios, but can show slower query performance because they represent an additional layer on top of actual data.
Avoid use of navigation attributes in BEx query	A query with a navigation attribute has to run an additional join, compared with a query with the same object as a characteristic in order to arrive at the values.
Use RSRT to monitor and troubleshoot slow running queries	Your SAP Admin can use the Query Monitor in SAP BW (transaction RSRT) to analyze performance issues with SAP BW queries. Review SAP note 1591837 for more information.
Avoid Restricted Key Figures and Calculated Key Figures	Both are computed during query execution and can slow down query performance.
Consider using incremental refresh to improve performance	Power BI refreshes the complete dataset with each refresh. If you're working with large volume of data, refreshing the full dataset on each refresh may not be optimal. In this scenario, you can use incremental refresh, so you're refreshing only a subset of data. For more details, go to Incremental refresh in Power BI .

See also

- [SAP Business Warehouse Application Server](#)
- [SAP Business Warehouse Message Server](#)

- Import vs. DirectQuery for SAP BW

Import vs. DirectQuery for SAP BW

1/15/2022 • 7 minutes to read • [Edit Online](#)

NOTE

This article discusses the differences between Import and DirectQuery modes in Power BI Desktop. For a description of using Import mode in Power Query Desktop or Power Query Online, go to the following sections:

SAP BW Application Server connector:

- [Connect to an SAP BW Application Server from Power Query Desktop](#)
- [Connect to an SAP BW Application Server from Power Query Online](#)

SAP BW Message Server connector:

- [Connect to an SAP BW Message Server from Power Query Desktop](#)
- [Connect to an SAP BW Message Server from Power Query Online](#)

With Power Query, you can connect to a wide variety of data sources, including online services, databases, different file formats, and others. If you are using Power BI Desktop, you can connect to these data sources in two different ways: either import the data into Power BI, or connect directly to data in the source repository, which is known as DirectQuery. When you connect to an SAP BW system, you can also choose between these two connectivity modes. For a complete list of data sources that support DirectQuery, refer to [Power BI data sources](#).

The main differences between the two connectivity modes are outlined here, as well as guidelines and limitations, as they relate to SAP BW connections. For additional information about DirectQuery mode, go to [Using DirectQuery in Power BI](#).

Import Connections

When you connect to a data source with Power BI Desktop, the navigator will allow you to select a set of tables (for relational sources) or a set of source objects (for multidimensional sources).

For SAP BW connections, you can select the objects you want to include in your query from the tree displayed. You can select an InfoProvider or BEx query for an InfoProvider, expand its key figures and dimensions, and select specific key figures, characteristics, attributes (properties), or hierarchies to be included in your query.

The selection defines a query that will return a flattened data set consisting of columns and rows. The selected characteristics levels, properties and key figures will be represented in the data set as columns. The key figures are aggregated according to the selected characteristics and their levels. A preview of the data is displayed in the navigator. You can edit these queries in Power Query prior to loading the data, for example to apply filters, or aggregate the data, or join different tables.

When the data defined by the queries is loaded, it will be imported into the Power BI in-memory cache.

As you start creating your visuals in Power BI Desktop, the imported data in the cache will be queried. The querying of cached data is very fast and changes to the visuals will be reflected immediately.

However, the user should take care when building visuals that further aggregate the data, when dealing with non-additive measures. For example, if the query imported each *Sales Office*, and the *Growth %* for each one, then if the user built a visual that will *Sum* the *Growth %* values across all *Sales Offices*, that aggregation will be performed locally, over the cached data. The result wouldn't be the same as requesting the overall *Growth %*

from SAP BW, and is probably not what's intended. To avoid such accidental aggregations, it's useful to set the **Default Summarization** for such columns to **Do not summarize**.

If the data in the underlying source changes, it won't be reflected in your visuals. It will be necessary to do a **Refresh**, which will reimport the data from the underlying source into the Power BI cache.

When you publish a report (.pbix file) to the Power BI service, a dataset is created and uploaded to the Power BI server. The imported data in the cache is included with that dataset. While you work with a report in the Power BI service, the uploaded data is queried, providing a fast response time and interactivity. You can set up a scheduled refresh of the dataset, or re-import the data manually. For on-premise SAP BW data sources, it's necessary to configure an on-premises data gateway. Information about installing and configuring the on-premises data gateway can be found in the following documentation:

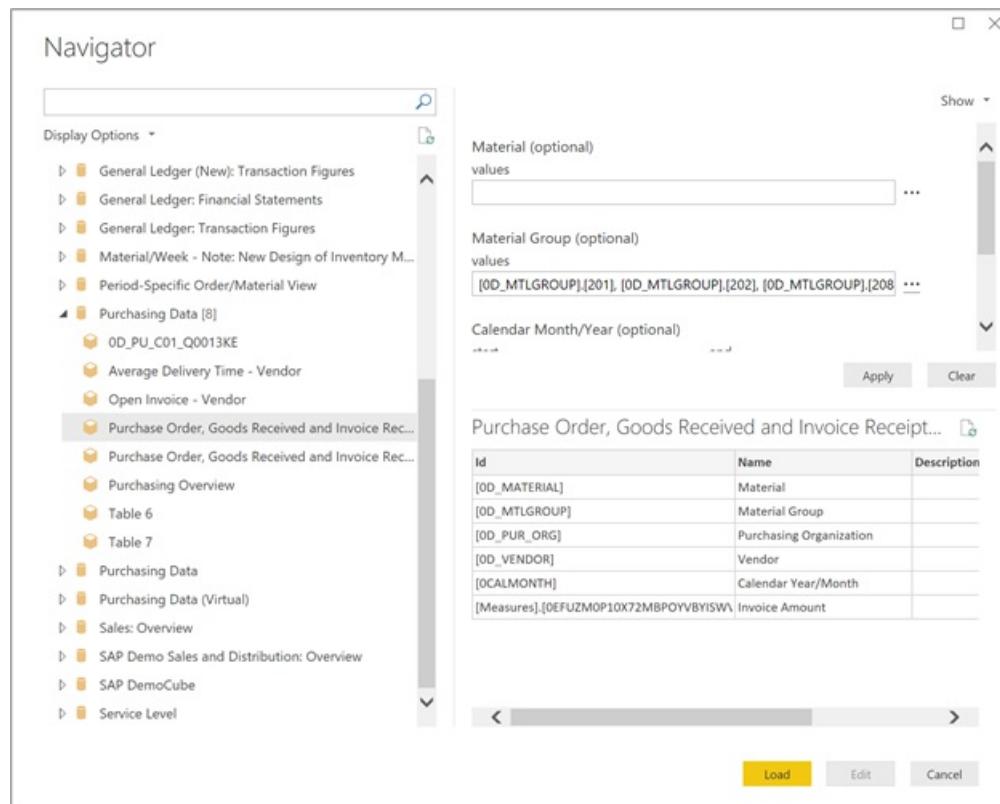
- [On-premises data gateway documentation](#)
- [Manage gateway data source in Power BI](#)
- [Data source management in Power Platform](#)

DirectQuery Connections

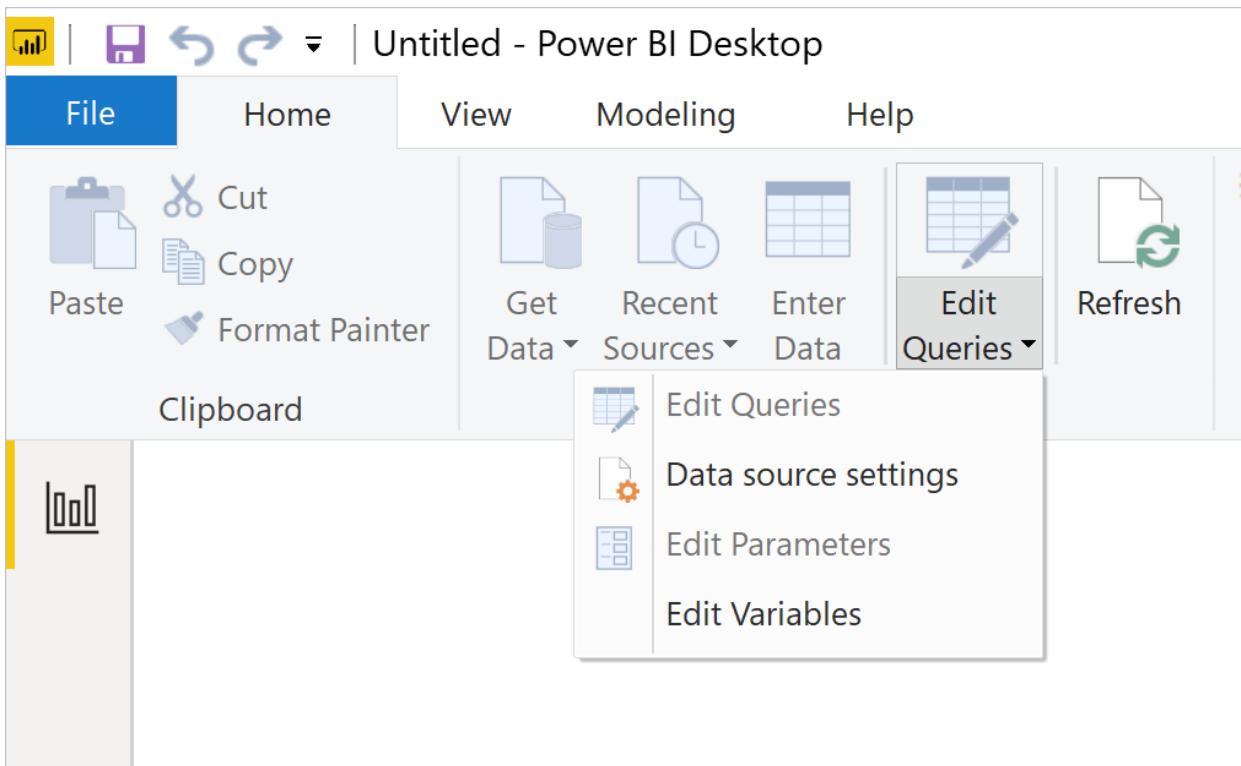
The navigation experience is slightly different when connecting to an SAP BW source in DirectQuery mode. The navigator will still display a list of available InfoProviders and BEx queries in SAP BW, however no Power BI query is defined in the process. You'll select the source object itself, that is, the InfoProvider or BEx query, and see the field list with the characteristics and key figures once you connect.

For SAP BW queries with variables, you can enter or select values as parameters of the query. Select the **Apply** button to include the specified parameters in the query.

Instead of a data preview, the metadata of the selected InfoCube or BEx Query is displayed. Once you select the **Load** button in **Navigator**, no data will be imported.



You can make changes to the values for the SAP BW query variables with the **Edit Queries** option on the Power BI Desktop ribbon.



As you start creating your visuals in Power BI Desktop, the underlying data source in SAP BW is queried to retrieve the required data. The time it takes to update a visual depends on the performance of the underlying SAP BW system.

Any changes in the underlying data won't be immediately reflected in your visuals. It will still be necessary to do a **Refresh**, which will rerun the queries for each visual against the underlying data source.

When you publish a report to the Power BI service, it will again result in the creation of a dataset in the Power BI service, just as for an import connection. However, no data is included with that dataset.

While you work with a report in the Power BI service, the underlying data source is queried again to retrieve the necessary data. For DirectQuery connections to your SAP BW and SAP HANA systems, you must have an [on-premises data gateway](#) installed and the [data source registered with the gateway](#).

For SAP BW queries with variables, end users can edit parameters of the query.

NOTE

For the end user to edit parameters, the dataset needs to be published to a premium workspace, in DirectQuery mode, and single sign-on (SSO) needs to be enabled.

General Recommendations

You should import data to Power BI whenever possible. Importing data takes advantage of the high-performance query engine of Power BI and provides a highly interactive and fully featured experience over your data.

However, DirectQuery provides the following advantages when connecting to SAP BW:

- Provides the ability to access SAP BW data using SSO, to ensure that security defined in the underlying SAP BW source is always applied. When accessing SAP BW using SSO, the user's data access permissions in SAP will apply, which may produce different results for different users. Data that a user isn't authorized to view will be trimmed by SAP BW.
- Ensures that the latest data can easily be seen, even if it's changing frequently in the underlying SAP BW

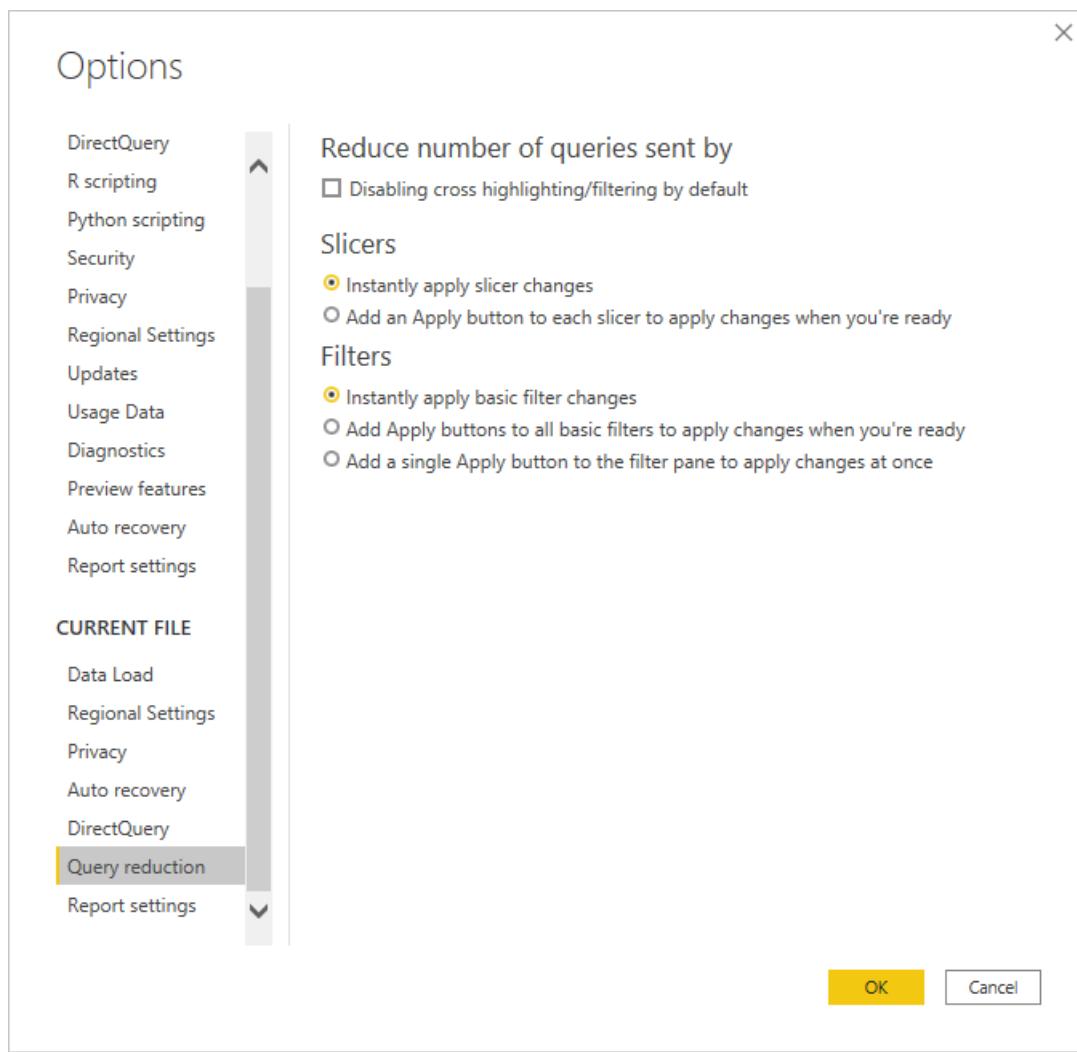
source.

- Ensures that complex measures can easily be handled, where the source SAP BW is always queried for the aggregate data, with no risk of unintended and misleading aggregates over imported caches of the data.
- Avoids caches of data being extracted and published, which might violate data sovereignty or security policies that apply.

Using DirectQuery is generally only feasible when the underlying data source can provide interactive queries for the typical aggregate query within seconds and is able to handle the query load that will be generated.

Additionally, the list of limitations that accompany use of DirectQuery should be considered, to ensure your goals can still be met.

If you're working with either very large datasets or encounter slow SAP BW query response time in DirectQuery mode, Power BI provides options in the report to send fewer queries, which makes it easier to interact with the report. To access these options in Power BI Desktop, go to **File > Options and settings > Options**, and select **Query reduction**.



You can disable cross-highlighting throughout your entire report, which reduces the number of queries sent to SAP BW. You can also add an **Apply** button to slicers and filter selections. You can make as many slicer and filter selections as you want, but no queries will be sent to SAP BW until you select the **Apply** button. Your selections will then be used to filter all your data.

These changes will apply to your report while you interact with it in Power BI Desktop, as well as when your users consume the report in the Power BI service.

In the Power BI service, the query cache for DirectQuery connections is updated on a periodic basis by querying

the data source. By default, this update happens every hour, but it can be configured to a different interval in dataset settings. For more information, go to [Data refresh in Power BI](#).

Also, many of the general best practices described in [Using DirectQuery in Power BI](#) apply equally when using DirectQuery over SAP BW. Additional details specific to SAP BW are described in [Connect to SAP Business Warehouse by using DirectQuery in Power BI](#).

See also

- [Windows authentication and single sign-on](#)

Windows authentication and single sign-on

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following information about Windows authentication and single sign-on applies only to Power Query Desktop. For more information about using Windows authentication and single sign-on in Power Query Desktop, go to [Overview of single sign-on \(SSO\) for gateways in Power BI](#).

For Windows-based authentication and single sign-on functionality, your SAP BW server must be configured for sign in using Secure Network Communications (SNC). SNC is a mechanism provided by the SAP system that enables application-level security on data exchanged between a client, such as Power BI Desktop, and the SAP BW server. SNC works with different external security products and offers features that the SAP system doesn't directly provide, including single sign-on.

In addition to your SAP BW server being configured for SNC sign in, your SAP user account needs to be configured with an SNC name (transaction SU01 in your SAP system).

For more detailed information, go to [Secure Network Communication](#), and the chapter *Single Sign-On Configuration* in this document.

Secure Login is a software solution by SAP that allows customers to benefit from the advantages of SNC without having to set up a public-key infrastructure (PKI). Secure Login allows users to authenticate with Windows Active Directory credentials.

Secure Login requires the installation of the Secure Login Client on your Power BI Desktop machine. The installation package is named SAPSetupSCL.EXE and can be obtained from the [SAP Service Marketplace](#) (requires SAP customer credentials).

For further information, go to [Secure Login](#).

1. In the **SAP Business Warehouse server** dialog box, select the **Windows** tab.
2. Select to either use your current Windows credentials or specify alternate Windows credentials.
3. Enter the **SNC Partner Name**. This name is the configured SNC name in the SAP BW server's security token. You can retrieve the SNC name with transaction **RZ11** (Profile Parameter Maintenance) in SAPGUI and parameter name **snc/identity/as**.

For X.509 certificate security tokens, the format is:

p:<X.509 Distinguished Name>

Example (values are case-sensitive): **p:CN=BW0, OU=BI, O=MyOrg, C=US**

For Kerberos security tokens, the format is:

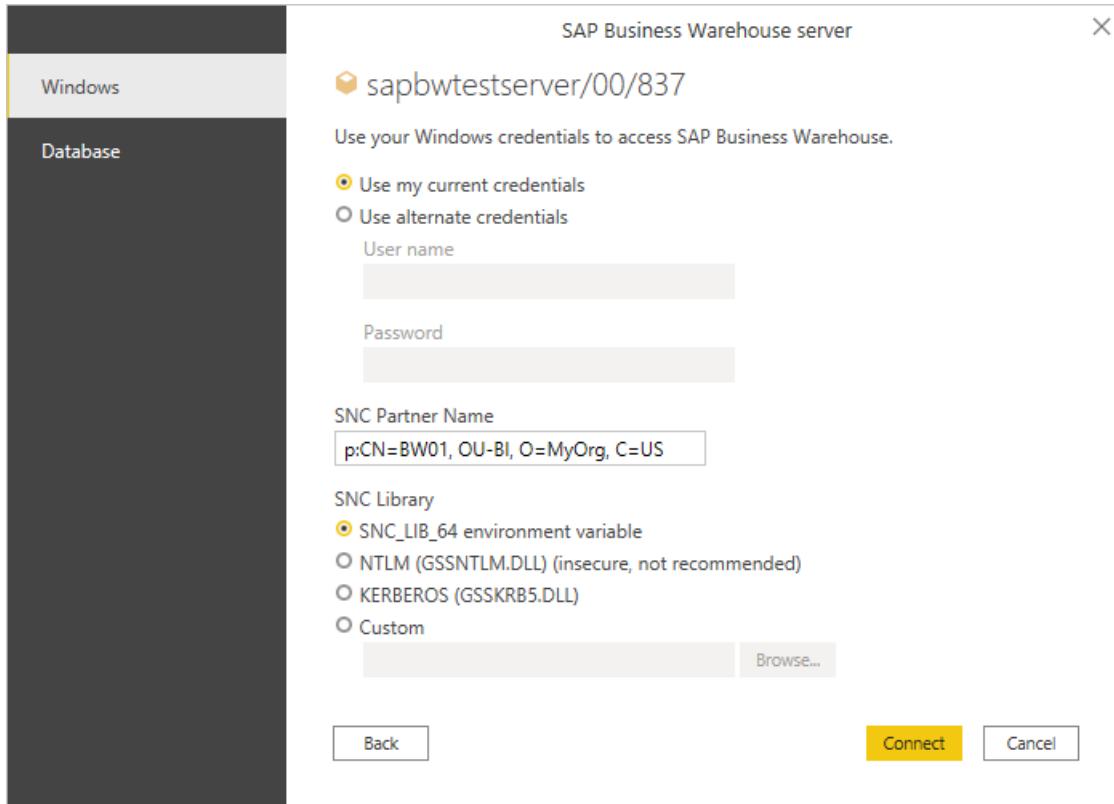
p:CN=<service_User_Principal_Name>

Example (values are case-sensitive): **p:CN=SAPServiceBW0@BWSERVER.MYORG.COM**

4. Select the **SNC Library** that your SAP BW environment has been configured for.
 - The **SNC_LIB** or **SNC_LIB_64** option will check the corresponding environment variable on your machine and use the DLL that's specified there.

- The **NTLM** and **KERBEROS** options will expect the corresponding DLL to be in a folder that's been specified in the **PATH** variable on your local machine. The libraries for 32-bit systems are **GSSNTLM.DLL** (for NTLM) and **GSSKRB5.DLL** (for Kerberos). The libraries for 64-bit systems are **GX64NTLM.DLL** (for NTLM) and **GX64KRB5.DLL** (for Kerberos).
- The **Custom** option allows for the use of a custom developed library.

Validate the settings with your SAP Administrator.



5. Select **Connect**.

See also

- [Use advanced options](#)

Use advanced options

1/15/2022 • 4 minutes to read • [Edit Online](#)

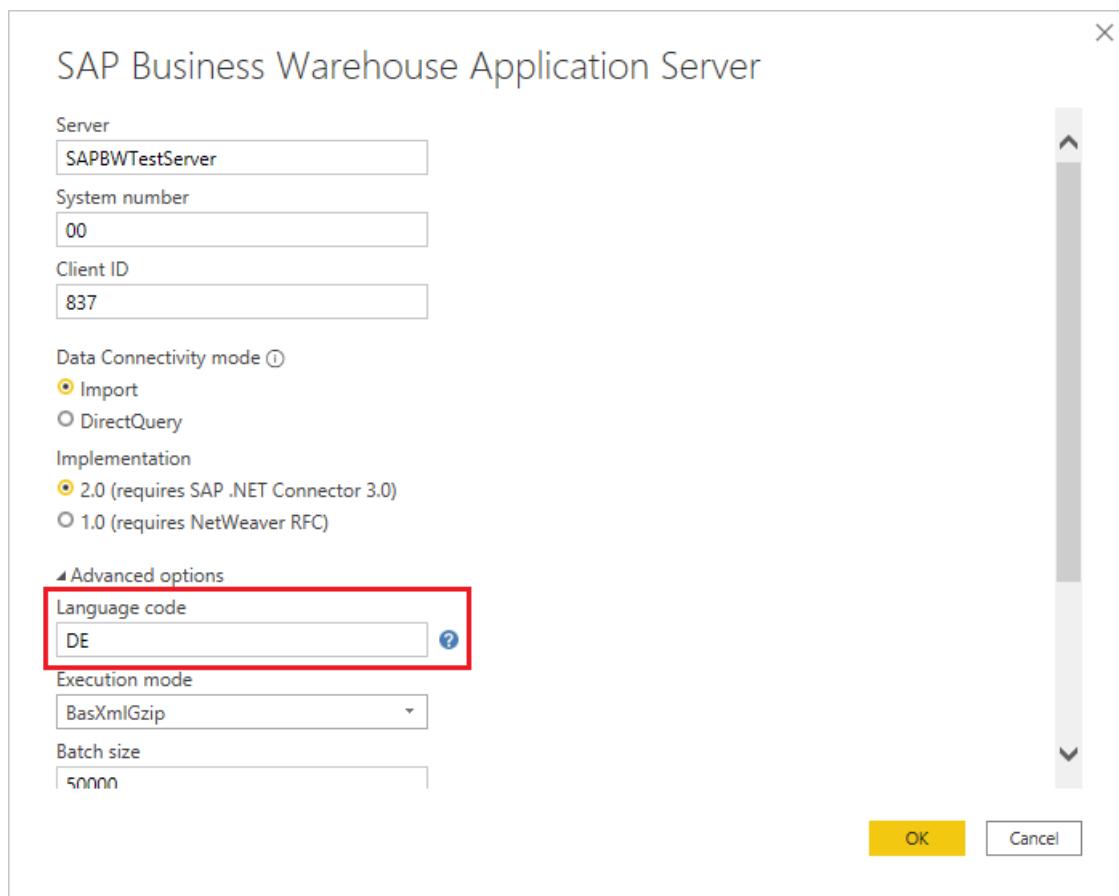
When you create a connection to an SAP Business Warehouse server, you can optionally specify a language code, execution mode, batch size, and an MDX Statement. Also, you can select whether you want to enable characteristic structures.

NOTE

Although the images in this article illustrate the advanced options in the SAP Business Warehouse Application Server connector, they work the same way in the SAP Business Warehouse Message Server connector.

Language code

You can optionally specify a language code when establishing a connection to the SAP BW server.

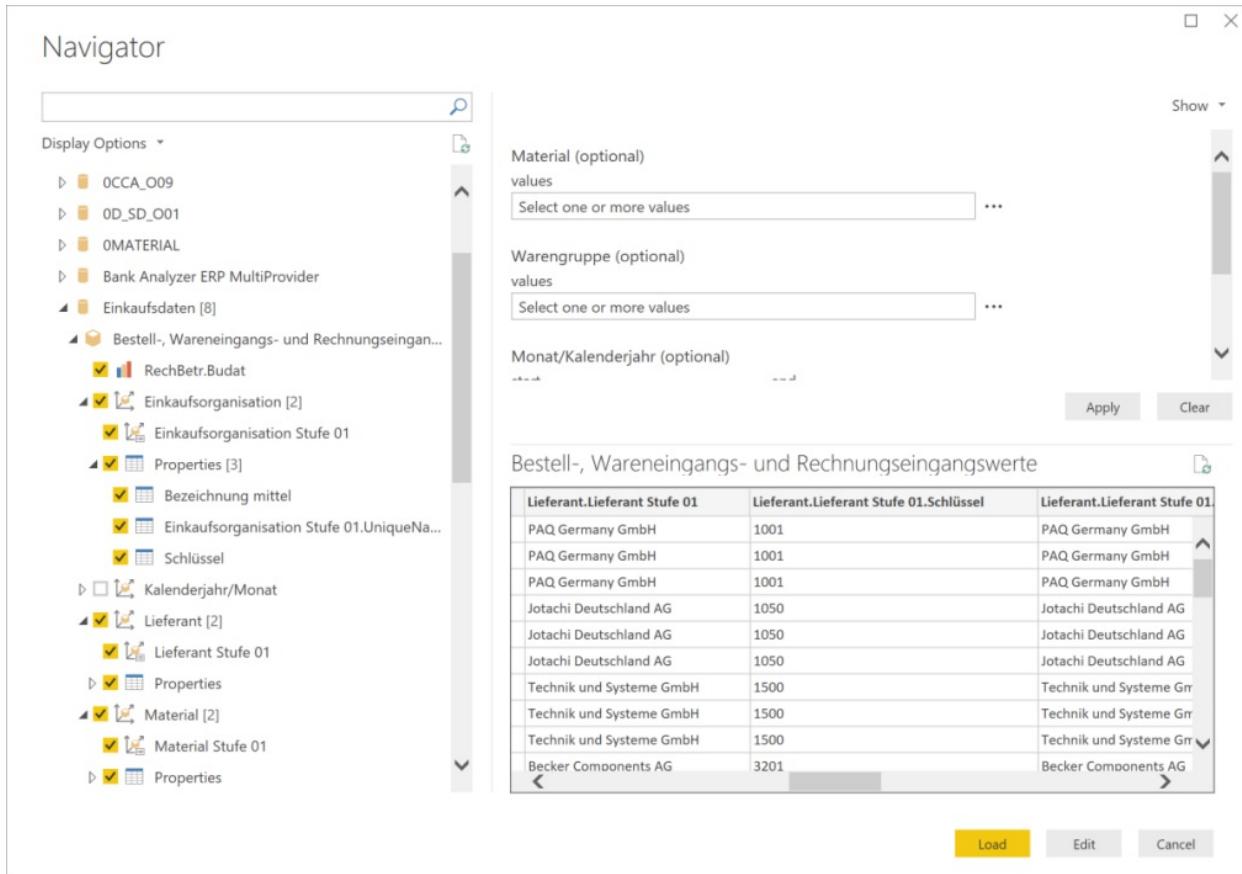


The expected value is a two-letter language code as defined in the SAP system. In Power Query Desktop, select the Help icon (question mark) next to the Language Code field for a list of valid values.

After you set the language code, Power Query displays the descriptive names of the data objects in SAP BW in the specified language, including the field names for the selected objects.

NOTE

Not all listed languages might be configured in your SAP BW system, and object descriptions might not be translated in all languages.



If no language code is specified, the default locale from the **Options** dialog will be used and mapped to a valid SAP language code. To view or override the current locale in Power BI Desktop, open the **File > Options and settings > Options** dialog box and select **Current File > Regional settings**. To view or override the current locale in Power Query Online, open the **Home > Options > Project options** dialog box. If you do override the locale, your setting gets persisted in your M query and would be honored if you copy-paste your query from Power Query Desktop to Power Query Online.

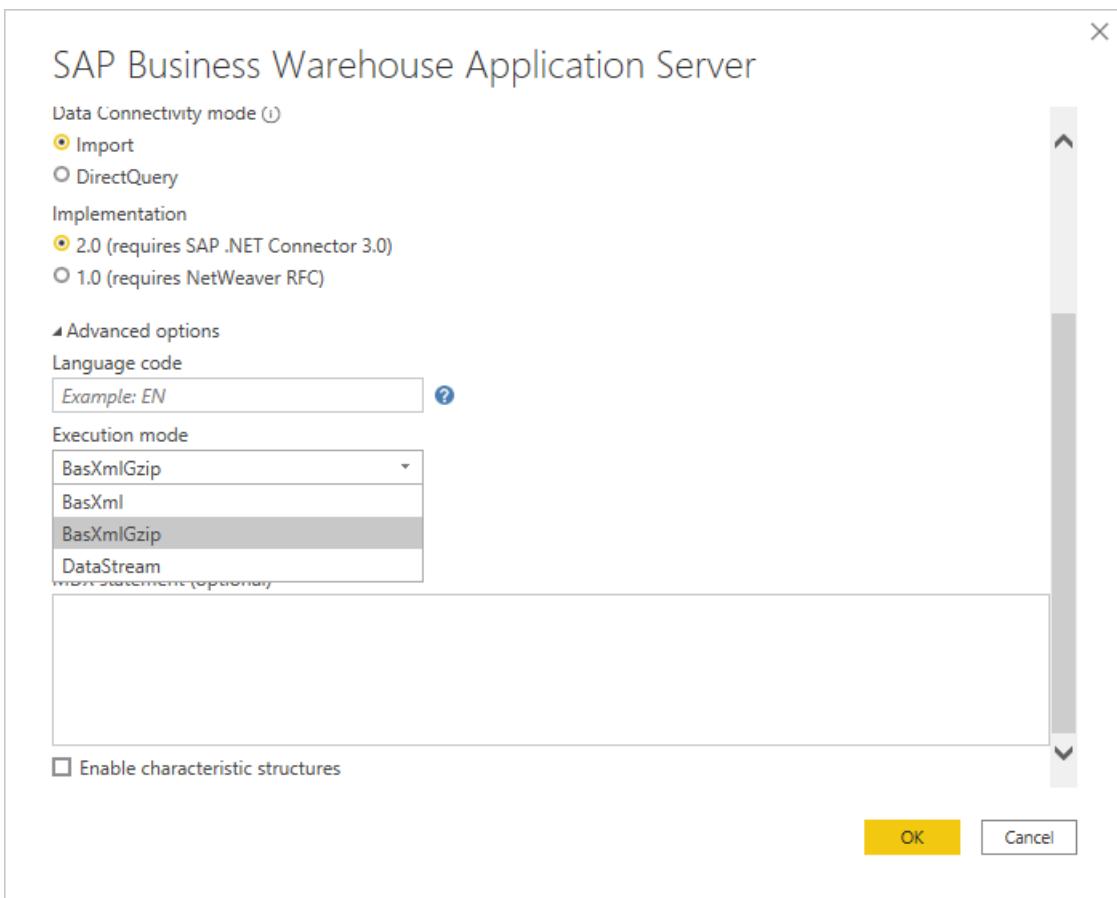
Execution mode

NOTE

Execution mode can't be changed in version 1.0 of the SAP BW connector.

The **Execution mode** option specifies the MDX interface is used to execute queries on the server. The following options are valid:

- **BasXml**: Specifies the *bXML flattening mode* option for MDX execution in SAP Business Warehouse.
- **BasXmlGzip**: Specifies the *Gzip compressed bXML flattening mode* option for MDX execution in SAP Business Warehouse. This option is recommended for low latency or high volume queries. The default value for the execution mode option.
- **DataStream**: Specifies the *DataStream flattening mode* option for MDX execution in SAP Business Warehouse.



Batch size

NOTE

Batch size can't be changed in version 1.0 of the SAP BW connector.

Specifies the maximum number of rows to retrieve at a time when executing an MDX statement. A small number translates into more calls to the server when retrieving a large dataset. A large number of rows may improve performance, but could cause memory issues on the SAP BW server. The default value is 50000 rows.

MDX Statement

NOTE

The MDX statement option is not available in Power Query Online.

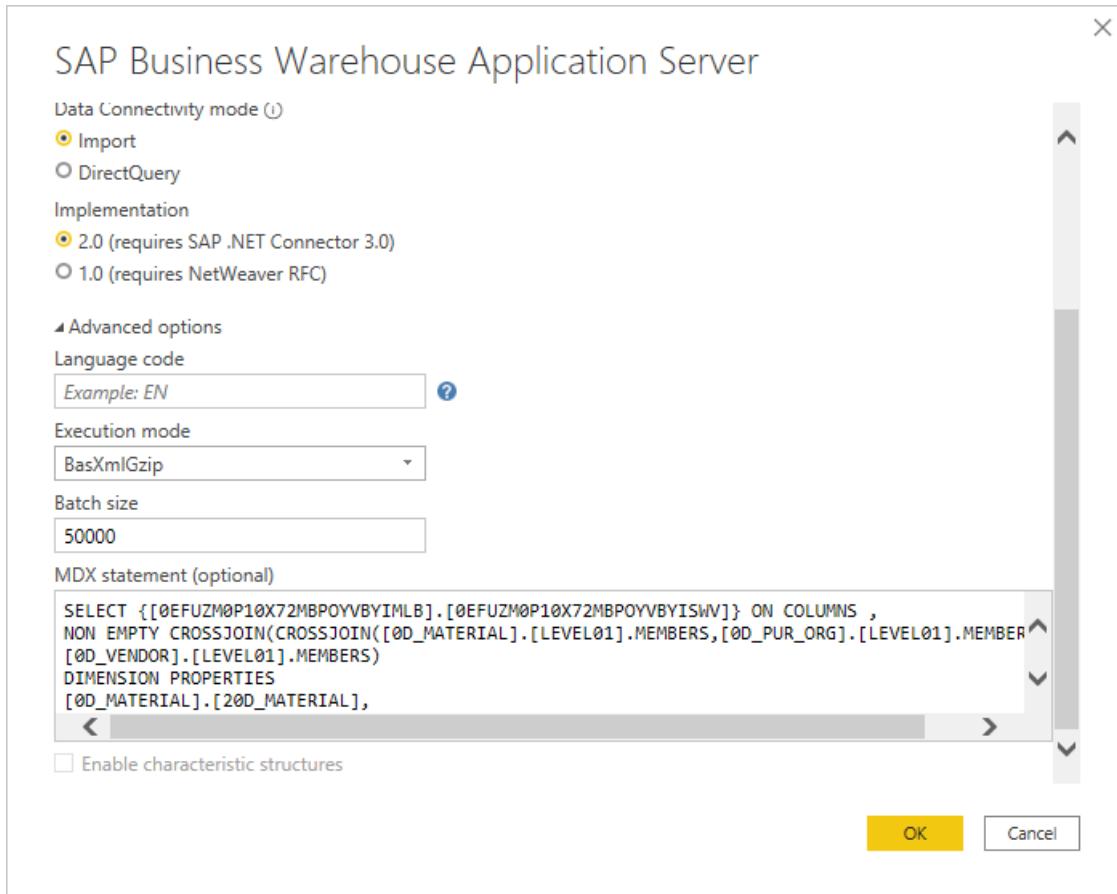
Instead of using the navigator to browse through and select from available data objects in SAP BW, a user who's familiar with the MDX query language can specify an MDX statement for direct execution in SAP BW. However, be aware that no further query folding will be applied when using a custom MDX statement.

The statement for the example used here would look as shown in the following sample, based on the technical names of the objects and properties in SAP BW.

```

SELECT {[0EFUZM0P10X72MBPOYVBYIMLB].[0EFUZM0P10X72MBPOYVBYISWV]} ON COLUMNS ,
NON EMPTY CROSSJOIN(CROSSJOIN([0D_MATERIAL].[LEVEL01].MEMBERS,[0D_PUR_ORG].[LEVEL01].MEMBERS) ,
[0D_VENDOR].[LEVEL01].MEMBERS)
DIMENSION PROPERTIES
[0D_MATERIAL].[20D_MATERIAL],
[0D_MATERIAL].[50D_MATERIAL],
[0D_PUR_ORG].[20D_PUR_ORG],
[0D_PUR_ORG].[50D_PUR_ORG],
[0D_VENDOR].[20D_VENDOR],
[0D_VENDOR].[50D_VENDOR] ON ROWS FROM [0D_PU_C01/0D_PU_C01_Q0013]

```



The SAP BW connector will display a preview of the data that is returned by the MDX statement. You can then either select **Load** to load the data (Power Query Desktop only), or select **Transform Data** to further manipulate the data set in the Power Query Editor.

[OD_MATERIAL].[LEVEL01].[50D_MATERIAL]	[OD_PUR_ORG].[LEVEL01].[20D_PUR_ORG]	[OD_PUR_ORG].[LEVEL01].[50D_PUR_ORG]	[OD_
Casing Notebook Speedy I CN	1512	Frankfurt	3201
Casing Notebook Speedy I CN	1514	Berlin	3201
Casing Notebook Speedy I CN	1516	Munich	3201
Casing Notebook Speedy I CN	1518	Hamburg	3201
Casing Notebook Speedy I CN	1552	London	2100
Casing Notebook Speedy I CN	1552	London	3500
Casing Notebook Speedy I CN	1554	Birmingham	2100
Casing Notebook Speedy I CN	1592	Paris	3300
Casing Notebook Speedy I CN	1612	New York	2500
Casing Notebook Speedy I CN	1612	New York	4151
Casing Notebook Speedy I CN	1614	San Francisco	4151
Casing Notebook Speedy I CN	1662	Toronto	3090
Casing Notebook Speedy I CN	1662	Toronto	3510
Casing Notebook Speedy I CN	1664	Vancouver	3090
Casing Notebook Speedy I CN	1664	Vancouver	3510

◀ ▶

To validate and troubleshoot an MDX statement, SAP BW provides the *MDXTTEST* transaction for SAP GUI for Windows users. Further, the MDXTTEST transaction can be a useful tool for analyzing server errors or performance concerns as a result of processing that occurs within the SAP BW system.

For more detailed information on this transaction, go to [MDX Test Environment](#).

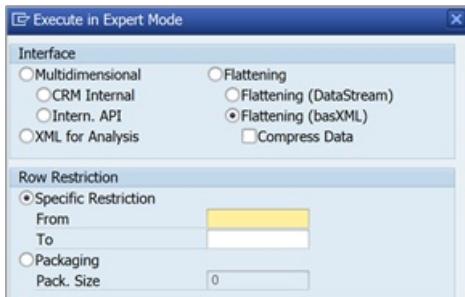
```

SELECT {[GEFUEMOP10X72MBPOVBYIMLB].[GEFUEMOP10X72MBPOVBYISMV]} ON COLUMNS, NON EMPTY Crossjoin Crossjoin([OD_MATERIAL].[LEVEL01].Members, [OD_PUR_ORG].[LEVEL01].Members), [OD_VENDOR].[LEVEL01].Members DIMENSION PROPERTIES [OD_MATERIAL].[20D_MATERIAL], [OD_MATERIAL].[50D_MATERIAL], [OD_PUR_ORG].ON ROWS FROM [OD_PU_C01/OD_PU_C01_Q0013]

```

MDXTTEST can also be used to construct an MDX statement. The transaction screen includes panels on the left that assist the user in browsing to a query object in SAP BW and generating an MDX statement.

The transaction offers different execution modes/interfaces for the MDX statement. Select *Flattening (basXML)* to mimic how Power Query would execute the query in SAP BW. This interface in SAP BW creates the row set dynamically using the selections of the MDX statement. The resulting dynamic table that's returned to Power Query Desktop has a very compact form that reduces memory consumption.



The transaction will display the result set of the MDX statement and useful runtime metrics.

Enable characteristic structures

The **Enable characteristic structures** selection changes the way characteristic structures are displayed in the navigator. A structure is an SAP BW object that can be used when building BEX queries. In the BEX UX they look like the following image.

The screenshot shows the SAP BEx Query Designer interface with the title bar "BEx Query Designer - Query: Structure Report". The menu bar includes "Query", "Edit", "View", "Tools", and "Help". Below the menu is a toolbar with various icons for file operations. On the left, the "infoProvider" pane displays the structure of the ZSP_IC1 provider, under "Key Figures" and "Dimensions". The main workspace is titled "Rows/Columns" and contains four sections: "Free Characteristics", "Columns", "Rows", and "Preview". The "Columns" section has a red box around it, showing "Key Figures" with items "ZSP_REVENUE" and "ZSP_QUANTITY". The "Rows" section also has a red box around it, showing "Structure" with items "Customer" and "Material". The "Preview" section shows a table with columns "ZSP_REVE" and "ZSP_QUAN" and rows "Customer" and "Material". The right side features a "Properties" pane with tabs for "Free Characteristics", "General", and "Preview". The "General" tab is active, stating "No properties for the current selected objects". A message at the bottom left says "To display properties for one or more objects of the selection, choose one or more objects of the selection".

If the **Enable characteristic structures** selection is clear (default), the connector will produce a cartesian product of each dimension on the structure with each available measure. For example:

▲  **CharacteristicStructure [12]**

-  **Calgary Net Value stat curr**
-  **Calgary Number of documents**
-  **Frankfurt Net Value stat curr**
-  **Frankfurt Number of documents**
-  **Munich Net Value stat curr**
-  **Munich Number of documents**

If selected, the connector produces only the available measures. For example:

▲  **CharacteristicStructure [9]**

-  **Net Value stat curr**
-  **Number of documents**

See also

- [Navigate the query objects](#)
- [Transform and filter SAP BW dataset](#)
- [SAP Business Warehouse connector troubleshooting](#)

SAP Business Warehouse connector troubleshooting

1/15/2022 • 11 minutes to read • [Edit Online](#)

This article provides troubleshooting situations (and possible solutions) for working with the SAP Business Warehouse (BW) connector.

Collect SAP BW advanced traces

NOTE

Collecting a trace of a query sent to the SAP BW server requires some options and settings that can only be provided by using Power BI Desktop. If you don't already have a copy of Power BI Desktop, you can obtain a copy at the [Microsoft Download Center](#). You can set all of the required options and settings for advanced traces using this free version.

Many times when an error occurs, it may be advantageous to collect a trace of the query that was sent to the SAP BW server and its response. The following procedure shows how to set up advanced traces for issues that occur using the SAP BW connector.

1. Close Power BI Desktop if it's running.
2. Create a new environment variable:
 - a. From the Windows Control Panel, select **System > Advanced System Settings**.
You could also open a **Command Prompt** and enter **sysdm.cpl**.
 - b. In **System Properties**, select the **Advanced** tab, and then select **Environment Variables**.
 - c. In **Environment Variables**, under **System Variables**, select **New**.
 - d. In **New System Variable**, under **Variable name**, enter **PBI_EnableSapBwTracing** and under **Variable value**, enter **true**.
 - e. Select **OK**.

When this advanced tracing is activated, an additional folder called **SapBw** will be created in the **Traces** folder. See the rest of this procedure for the location of the **Traces** folder.

3. Open Power BI Desktop.
4. Clear the cache before capturing.
 - a. In Power BI desktop, select the **File** tab.
 - b. Select **Options and settings > Options**.
 - c. Under **Global** settings, choose **Data Load**.
 - d. Select **Clear Cache**.
5. While you're still in **Options and settings**, enable tracing.
 - a. Under **Global** settings, choose **Diagnostics**.
 - b. Select **Enable tracing**.
6. While you're still in **Options and settings > Global > Diagnostics**, select **Open crash dump/traces folder**. Ensure the folder is clear before capturing new traces.
7. Reproduce the issue.

8. Once done, close Power BI Desktop so the logs are flushed to disk.
9. You can view the newly captured traces under the **SapBw** folder (the **Traces** folder that contains the **SapBw** folder is shown by selecting **Open crash dump/traces folder** on the **Diagnostics** page in Power BI Desktop).
10. Make sure you deactivate this advanced tracing once you're done, by either removing the environment variable or setting **PBI_EnableSapBwTracing** to false.

Collect SAP BW advanced traces with CPIC traces

If you're investigating authentication or single sign-on (SSO) issues, use the same procedure as described in [Collect SAP BW advanced traces](#), except in step 2d, enter the following additional system variables and values:

- **CPIC_TRACE**—3
- **CPIC_TRACE_DIR**—a valid folder, for example: E:\traces\CPIC

The rest of the procedure remains the same. You can view the CPIC traces in the folder you specified in the **CPIC_TRACE_DIR** environment variable. You can also view the regular traces under the **SapBw** folder.

Also make sure you deactivate this advanced tracing once you're done, by either removing the environment variables or setting **BI_EnableSapBwTracing** to false and **CPIC_TRACE** to 0.

Perform clean installation of SAP .NET connector

If it becomes necessary to reinstall the SAP .NET connector:

1. Remove (uninstall) the SAP .NET Connector.
2. After removing, verify that the SAP .NET Connector isn't installed in the Global Assembly Cache (GAC), by making sure the following paths do NOT exist or do NOT contain DLLs:
 - 32 bit GAC:

C:\Windows\Microsoft.NET\assembly\GAC_32\sapnco\v4.0_3.0.0.42__50436dca5c7f7d23
C:\Windows\Microsoft.NET\assembly\GAC_32\sapnco_utils\v4.0_3.0.0.42__50436dca5c7f7d23
 - 64 bit GAC:

C:\Windows\Microsoft.NET\assembly\GAC_64\sapnco\v4.0_3.0.0.42__50436dca5c7f7d23
C:\Windows\Microsoft.NET\assembly\GAC_64\sapnco_utils\v4.0_3.0.0.42__50436dca5c7f7d23
3. Verify that the binaries aren't in **Program Files**. Make sure the following locations do NOT exist or are empty:

C:\Program Files\SAP\SAP_DotNetConnector3_Net40_x64
C:\Program Files (x86)\sap\SAP_DotNetConnector3_Net40_x86
4. Reinstall the connector, and remember to select the **Install assemblies to GAC** option. We recommend you use the latest, 3.0.23.

Troubleshooting error messages

SAP BW ErrorCode method not found

```
Method not found: 'Int32 SAP.Middleware.Connector.RfcBaseException.get_ErrorCode()'
```

This error is thrown when an error occurs on the SAP BW server and the SAP .NET connector tries to retrieve information about that error. However, this error may be hiding the real error. This error can occur when:

- Using an old version of the SAP .NET connector.
- Multiple versions of the SAP .NET connector are installed.
- The SAP .NET connector was installed twice, once in the Global Assembly Cache (GAC) and once not in the GAC.

Follow the instructions under [Perform clean installation of SAP .NET connector](#) to reinstall the connector.

This won't solve the problem, but will provide the actual error message.

Exception: The type initializer for 'Microsoft.Mashup.Engine1.Library.SapBusinessWarehouse.SapBwMicrosoftProviderFactoryService' threw an exception.

Follow instructions under [Perform clean installation of SAP .NET connector](#) to reinstall the connector.

This connector requires one or more additional components

If you receive this error message, use the following troubleshooting steps:

1. Verify that the version of the SAP .NET connector is installed in the correct bit length. If you have Power BI Desktop 64-bit installed, make sure you installed the 64-bit SAP .NET connector.
2. Verify that, while installing the SAP .NET Connector, the [Install assemblies to GAC](#) was checked. To verify GAC is installed, open Windows Explorer and go to:

C:\Windows\Microsoft.NET\assembly\GAC_64\sapnco

For example, the full path might be:

C:\Windows\Microsoft.NET\assembly\GAC_64\sapnco\v4.0_3.0.0.42_50436dca5c7f7d23\sapnco.dll

If you installed the 32-bit version of the SAP .NET connector, it would be

C:\Windows\Microsoft.NET\assembly\GAC_32\sapnco\v4.0_3.0.0.42_50436dca5c7f7d23\sapnco.dll (and you'd need a 32-bit version of Power BI Desktop).

Another way to check the GAC is to use gacutil (one of the options for disabling strong name signing). You'd need to run it from a 64-bit command prompt. You can check the contents of the GAC by opening a command prompt, navigating to the gacutil.exe path and executing:

```
gacutil -l
```

For example, in the output you should see:

```
sapnco, Version=3.0.0.42, Culture=neutral, PublicKeyToken=50436dca5c7f7d23,  
processorArchitecture=AMD64 sapnco_utils, Version=3.0.0.42, Culture=neutral,  
PublicKeyToken=50436dca5c7f7d23, processorArchitecture=AMD64
```

"No RFC authorization for function ..."

Implementation 2.0 requires access to the following BAPIs. To resolve, contact the SAP Basis team and request permissions to these BAPIs and RFCs for the user.

- Connectivity:
 - RFC_PING
 - RFC_METADATA_GET
- MDX execution:
 - RSR_MDX_CREATE_OBJECT

- BAPI_MDDATASET_CREATE_OBJECT
- BAPI_MDDATASET_SELECT_DATA
- BAPI_MDDATASET_DELETE_OBJECT
- RSR_MDX_GET_AXIS_INFO
- RSR_MDX_GET_AXIS_DATA
- RSR_MDX_GET_CELL_DATA
- BAPI_MDDATASET_GET_AXIS_INFO
- BAPI_MDDATASET_GET_AXIS_DATA
- BAPI_MDDATASET_GET_CELL_DATA
- ExecutionMode flattening:
 - RSR_MDX_GET_FLAT_DATA
 - RSR_MDX_GET_FS_DATA
 - BAPI_MDDATASET_GET_FLAT_DATA
 - BAPI_MDDATASET_GET_FS_DATA
- ExecutionMode streaming:
 - BAPI_MDDATASET_GET_STREAMDATA
 - BAPI_MDDATASET_GET_STREAMINFO
- ExecutionMode BasXml:
 - RSR_MDX_BXML_GET_DATA
 - RSR_MDX_BXML_GET_GZIP_DATA
 - RSR_MDX_BXML_GET_INFO
 - RSR_MDX_BXML_SET_BINDING
- Metadata:
 - BAPI_MDPROVIDER_GET_DIMENSIONS
 - BAPI_MDPROVIDER_GET_CATALOGS
 - BAPI_MDPROVIDER_GET_CUBES
 - BAPI_MDPROVIDER_GET_MEASURES
 - BAPI_MDPROVIDER_GET_HIERARCHYS
 - BAPI_MDPROVIDER_GET_LEVELS
 - BAPI_MDPROVIDER_GET_PROPERTIES
 - BAPI_MDPROVIDER_GET_MEMBERS
 - BAPI_MDPROVIDER_GET_VARIABLES
- Information:
 - BAPI_IOBJ_GETDETAIL (required for typed dimensions (DATS, TIMS))
 - BAPI_USER_GET_DETAIL (only used for flattening interface)
 - RFC_READ_TABLE (required for catalog names and certain variable values calls)
- Might be called by the underlying SAP .NET connector:
 - RFC_GET_FUNCTION_INTERFACE
 - FUNCTION_IMPORT_INTERFACE
 - DDIF_FIELDINFO_GET
 - SYSTEM_FINISH_ATTACH_GUI
 - BGRFC_DEST_CONFIRM
 - BGRFC_CHECK_UNIT_STATE_SERVER
 - BGRFC_DEST_SHIP

- ARFC_DEST_SHIP
- RFC_FUNCTION_SEARCH
- RFC_SYSTEM_INFO
- RFC_SET_REG_SERVER_PROPERTY
- RFC_DOCU
- SEO_GET_CLIF_REMOTE
- SYSTEM_PREPARE_ATTACH_GUI
- API_CLEAR_TID
- ARFC_DEST_CONFIRM

Method not found 'Int32 SAP.Middleware.Connector.RfcBaseException.get_ErrorCode()

Verify that the SAP .NET connector is installed properly. Refer to [Perform clean installation of SAP .NET connector](#).

This error appears when the installed version in the GAC is lower than the expected 3.0.18.0 version. [SAP Note 2417315](#) discusses this scenario.

Connection string properties set by the connector

When *both* SNC Partner Name and SNC Library are provided, the SAP BW Application Server connector (implementation 2.0) will set these properties in the connection string:

- SNC_MODE—SncModeApply
- SNC_LIB—with the library path specified; if it's an environment variable, it's expanded at this point
- SNC_PARTNERNAME—with the value provided
- SNC_QOP = RfcConfigParameters.RfcSncQOP.Default

These are used for both SAP BW Application Server and SAP BW Message Server connections.

For both connection types, the connector sets:

- LANG (Language)
- CLIENT

For the SAP BW Application Server connection, the connector sets:

- ASHOST (AppServerHost)
- SYSNR (SystemNumber)

For SAP BW Message Server connections, the connector sets:

- MSHOST (MessageServerHost)
- SYSID (SystemID)
- GROUP (LogonGroup)

Invalid MDX command with <internal>

This error comes directly from the SAP BW server. Implementation 1 of the connector, based on Netweaver RFC, didn't expose these errors to the user, returning an empty result set instead.

This issue is discussed in the following SAP Notes. Access to these notes requires an S-user. Contact your SAP Basis team to apply the relevant fixes for this issue.

- [1084454—MDX: System error "Invalid MDX command with <internal>"](#)
- [1282785—MDX: System error "Invalid MDX command with <internal>"](#)
- [401607—Errors at MDX command with CROSSJOIN in the slicer](#)
- [1786009—Invalid MDX when using a member formula with special char](#)

Additionally, for other similar errors, you can review the contents of the following SAP notes, and apply them as appropriate for your environment:

- [1142664—MDX: Composite SAP Note about performance improvements](#)
- [1156101—MDX: Composite SAP Note about incorrect data](#)

Issues and limitations

Changing variable names on an SAP cube puts DirectQuery report in a broken, unrecoverable state

The following symptoms occur:

- Error message— [Expression.Error] The import [XXXX] matches no exports.
- In the logs— Message: [Expression.Error] The key didn't match any rows in the table.
- StackTrace:

```
at Microsoft.Mashup.Engine1.Runtime.TableValue.get_Item(Value key)
at
Microsoft.Mashup.Engine1.Library.Cube.CubeParametersModule.Cube.ApplyParameterFunctionValue.GetParameterValue(CubeValue cubeValue, Value parameter)
at
Microsoft.Mashup.Engine1.Library.Cube.CubeParametersModule.Cube.ApplyParameterFunctionValue.TypedInvoke(TableValue cube, Value parameter, Value arguments)
Detail: [Key = [Id = "\[!V000004\]"], Table = #table({...}, {...})]
```

One possible workaround is to:

1. Make a copy of the PBIX file (as things might break).
2. Add an environment variable called PBI_AlwaysEnableQueryEditor with a value of true. This setting will allow access to the query editor even in DirectQuery mode.

NOTE

This environment variable is unsupported, so should only be used as outlined here.

3. Right-click on the "Cube" query and select **Advanced Editor**.
4. The query there should have a line that starts with "{Cube.ApplyParameter, "[!V000004]" (*the missing parameter*)}. Remove that line.
5. Select **Done**.
6. Close the Power Query Editor.
7. Refresh the affected visual.

If the above workaround doesn't work, the only alternative fix is for you to recreate the report.

Numeric data from SAP BW

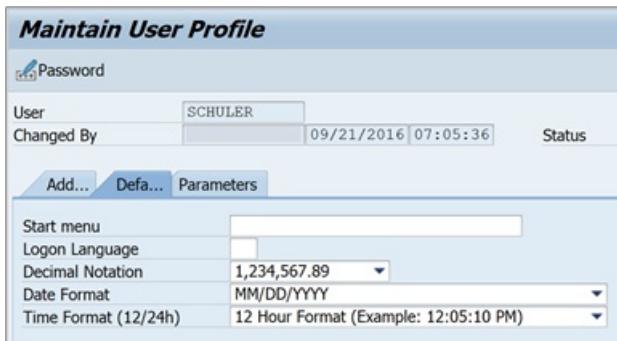
NOTE

The following information only applies when using Implementation 1.0 of the SAP BW connector or Implementation 2.0 of the SAP BW connector with Flattening mode (when ExecutionMode=67).

User accounts in SAP BW have default settings for how decimal or date/time values are formatted when

displayed to the user in the SAP GUI.

The default settings are maintained in the SAP system in the User Profile for an account, and the user can view or change these settings in the SAP GUI with the menu path **System > User Profile > Own Data**.



Power BI Desktop queries the SAP system for the decimal notation of the connected user and uses that notation to format decimal values in the data from SAP BW.

SAP BW returns decimal data with either a `,` (comma) or a `.` (dot) as the decimal separator. To specify which of those SAP BW should use for the decimal separator, the driver used by Power BI Desktop makes a call to `BAPI_USER_GET_DETAIL`. This call returns a structure called `DEFAULTS`, which has a field called `DCPFM` that stores *Decimal Format Notation*. The field takes one of the following values:

- `' '` (space) = Decimal point is comma: N.NNN,NN
- `'X'` = Decimal point is period: N,NNN.NN
- `'Y'` = Decimal point is N NNN NNN,NN

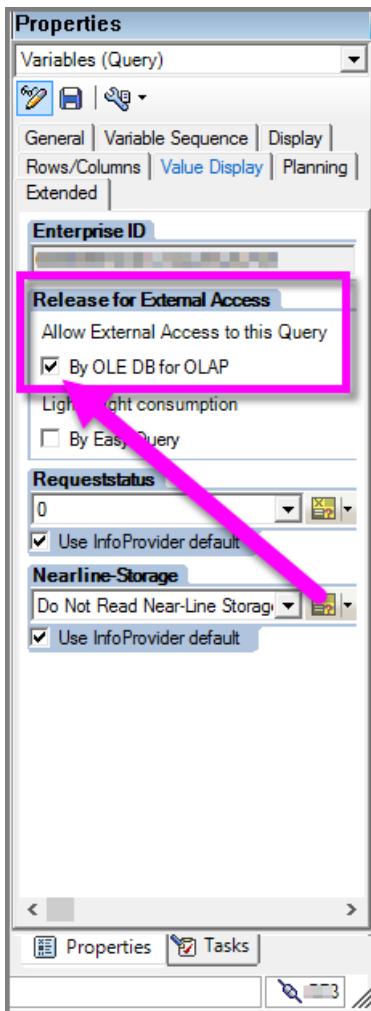
Customers who have reported this issue found that the call to `BAPI_USER_GET_DETAIL` is failing for a particular user, which is showing the incorrect data, with an error message similar to the following message:

```
You are not authorized to display users in group TI:  
<item>  
  <TYPE>E</TYPE>  
  <ID>01</ID>  
  <NUMBER>512</NUMBER>  
  <MESSAGE>You are not authorized to display users in group TI</MESSAGE>  
  <LOG_NO/>  
  <LOG_MSG_NO>000000</LOG_MSG_NO>  
  <MESSAGE_V1>TI</MESSAGE_V1>  
  <MESSAGE_V2/>  
  <MESSAGE_V3/>  
  <MESSAGE_V4/>  
  <PARAMETER/>  
  <ROW>0</ROW>  
  <FIELD>BNAME</FIELD>  
  <SYSTEM>CLNTPW1400</SYSTEM>  
</item>
```

To solve this error, users must ask their SAP admin to grant the SAP BW user being used in Power BI the right to execute `BAPI_USER_GET_DETAIL`. It's also worth verifying that the user has the required `DCPFM` value, as described earlier in this troubleshooting solution.

Connectivity for SAP BEx queries

You can perform BEx queries in Power BI Desktop by enabling a specific property, as shown in the following image:



MDX interface limitation

A limitation of the MDX interface is that long variables lose their technical name and are replaced by V00000#.

No data preview in Navigator window

In some cases, the **Navigator** dialog box doesn't display a data preview and instead provides an *object reference not set to an instance of an object* error message.

SAP users need access to specific BAPI function modules to get metadata and retrieve data from SAP BW's InfoProviders. These modules include:

- BAPI_MDPROVIDER_GET_CATALOGS
- BAPI_MDPROVIDER_GET_CUBES
- BAPI_MDPROVIDER_GET_DIMENSIONS
- BAPI_MDPROVIDER_GET_HIERARCHYS
- BAPI_MDPROVIDER_GET_LEVELS
- BAPI_MDPROVIDER_GET_MEASURES
- BAPI_MDPROVIDER_GET_MEMBERS
- BAPI_MDPROVIDER_GET_VARIABLES
- BAPI_IOBJ_GETDETAIL

To solve this issue, verify that the user has access to the various MDPROVIDER modules and BAPI_IOBJ_GETDETAIL. To further troubleshoot this or similar issues, you can enable tracing. Select **File > Options and settings > Options**. In Options, select **Diagnostics**, then select **Enable tracing**. Attempt to retrieve data from SAP BW while tracing is active, and examine the trace file for more detail.

Memory Exceptions

In some cases, you might encounter one of the following memory errors:

- Message: No more memory available to add rows to an internal table.
- Message: [DataSource.Error] SAP Business Warehouse: The memory request for [number] bytes could not be complied with.
- Message: The memory request for [number] bytes could not be complied with.

These memory exceptions are from the SAP BW server and are due to the server running out of available memory to process the query. This might happen when the query returns a large set of results or when the query is too complex for the server to handle, for example, when a query has many crossjoins.

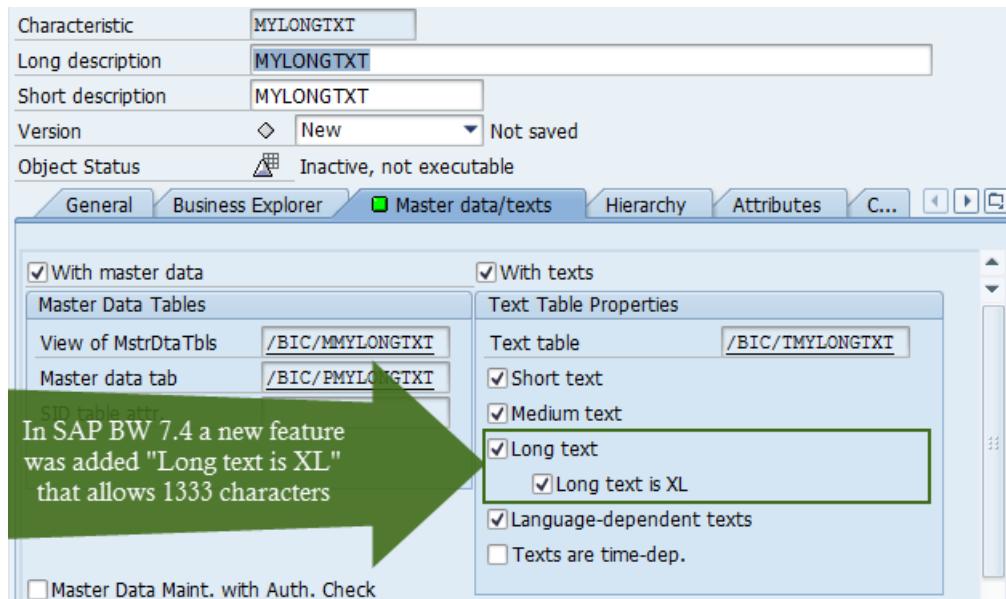
To resolve this error, the recommendation is to simplify the query or divide it into smaller queries. If possible, push more aggregation to the server. Alternatively, contact your SAP Basis team to increase the resources available in the server.

Loading text strings longer than 60 characters in Power BI Desktop fails

In some cases you may find that text strings are being truncated to 60 characters in Power BI Desktop.

First, follow the instructions in [2777473 - MDX: FAQ for Power BI accessing BW or BW/4HANA](#) and see if that resolves your issue.

Because the Power Query SAP Business Warehouse connector uses the MDX interface provided by SAP for 3rd party access, you'll need to contact SAP for possible solutions as they own the layer between the MDX interface and the SAP BW server. Ask how "long text is XL" can be specified for your specific scenario.



SAP HANA database

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Analysis Services
Authentication Types Supported	Basic Database Windows
Function Reference Documentation	SapHana.Database

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

You'll need an SAP account to sign in to the website and download the drivers. If you're unsure, contact the SAP administrator in your organization.

To use SAP HANA in Power BI Desktop or Excel, you must have the SAP HANA ODBC driver installed on the local client computer for the SAP HANA data connection to work properly. You can download the SAP HANA Client tools from [SAP Development Tools](#), which contains the necessary ODBC driver. Or you can get it from the [SAP Software Download Center](#). In the Software portal, search for the SAP HANA CLIENT for Windows computers. Since the SAP Software Download Center changes its structure frequently, more specific guidance for navigating that site isn't available. For instructions about installing the SAP HANA ODBC driver, see [Installing SAP HANA ODBC Driver on Windows 64 Bits](#).

To use SAP HANA in Excel, you must have either the 32-bit or 64-bit SAP HANA ODBC driver (depending on whether you're using the 32-bit or 64-bit version of Excel) installed on the local client computer.

This feature is only available in Excel for Windows if you have Office 2019 or a [Microsoft 365 subscription](#). If you're a Microsoft 365 subscriber, [make sure you have the latest version of Office](#).

HANA 1.0 SPS 12rev122.09, 2.0 SPS 3rev30 and BW/4HANA 2.0 is supported.

Capabilities Supported

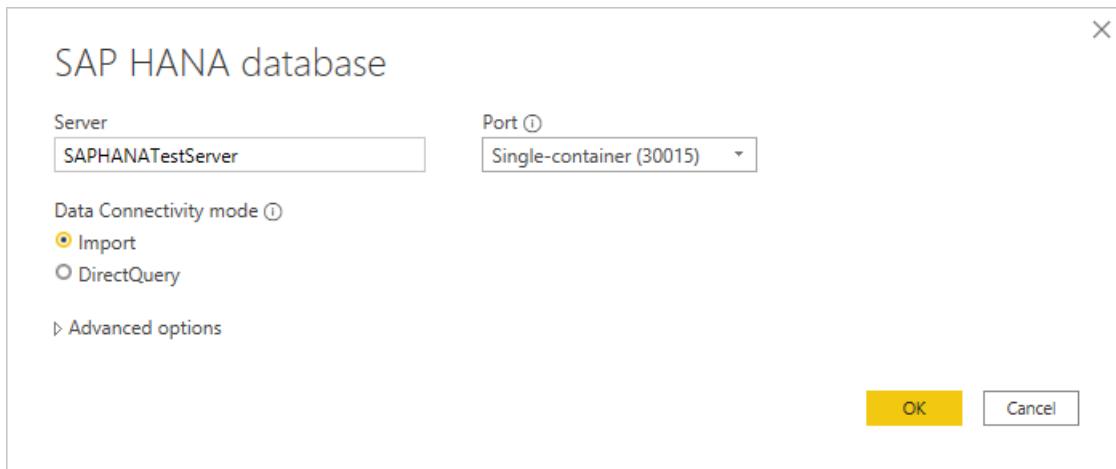
- Import

- Direct Query
- Advanced
 - SQL Statement

Connect to an SAP HANA database from Power Query Desktop

To connect to an SAP HANA database from Power Query Desktop:

1. Select **Get Data > SAP HANA database** in Power BI Desktop or **From Database > From SAP HANA Database** in the **Data** ribbon in Excel.
2. Enter the name and port of the SAP HANA server you want to connect to. The example in the following figure uses **SAPHANATestServer** on port **30015**.



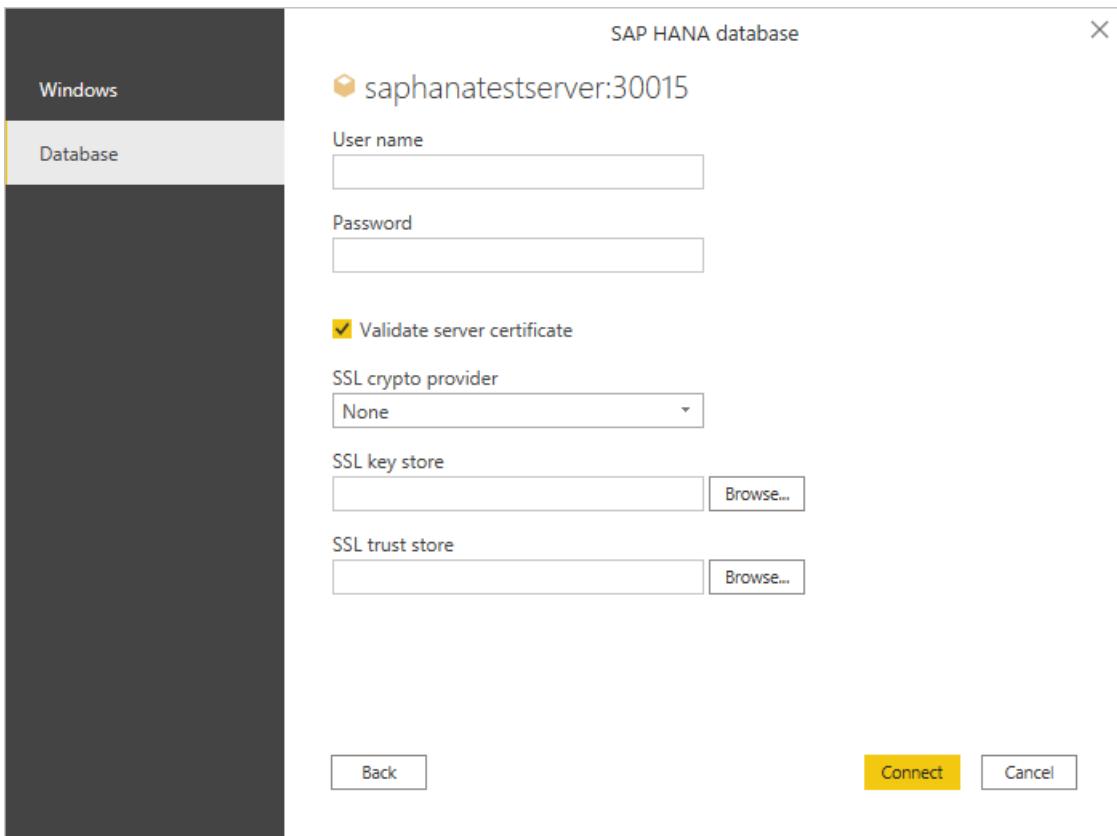
By default, the port number is set to support a single container database. If your SAP HANA database can contain more than one multitenant database container, select **Multi-container system database (30013)**. If you want to connect to a tenant database or a database with a non-default instance number, select **Custom** from the **Port** drop-down menu.

If you're connecting to an SAP HANA database from Power BI Desktop, you're also given the option of selecting either **Import** or **DirectQuery**. The example in this article uses **Import**, which is the default (and the only mode for Excel). For more information about connecting to the database using DirectQuery in Power BI Desktop, see [Connect to SAP HANA data sources by using DirectQuery in Power BI](#).

If you select **Advanced options**, you can also enter an SQL statement. For more information on using this SQL statement, see [Import data from a database using native database query](#).

Once you've entered all of your options, select **OK**.

3. If you are accessing a database for the first time, you'll be asked to enter your credentials for authentication. In this example, the SAP HANA server requires database user credentials, so select **Database** and enter your user name and password. If necessary, enter your server certificate information.



Also, you may need to validate the server certificate. For more information about using validate server certificate selections, see [Using SAP HANA encryption](#). In Power BI Desktop and Excel, the validate server certificate selection is enabled by default. If you've already set up these selections in ODBC Data Source Administrator, clear the **Validate server certificate** check box. To learn more about using ODBC Data Source Administrator to set up these selections, see [Configure SSL for ODBC client access to SAP HANA](#).

For more information about authentication, see [Authentication with a data source](#).

Once you've filled in all required information, select **Connect**.

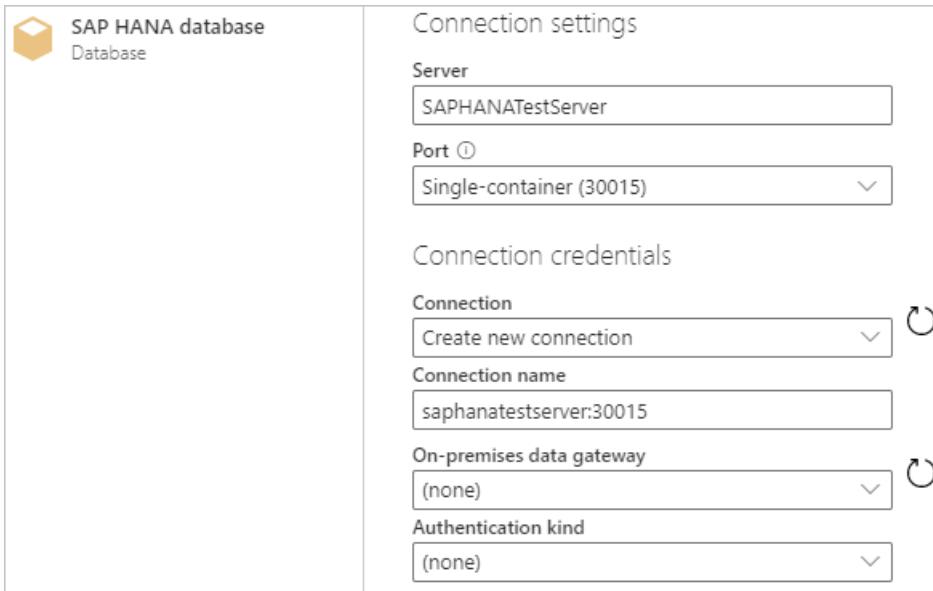
4. From the **Navigator** dialog box, you can either transform the data in the Power Query editor by selecting **Transform Data**, or load the data by selecting **Load**.

Connect to an SAP HANA database Power Query Online

To connect to SAP HANA data from Power Query Online:

1. From the **Data sources** page, select **SAP HANA database**.
2. Enter the name and port of the SAP HANA server you want to connect to. The example in the following figure uses `SAPHANATestServer` on port `30015`.

If you want to connect to a tenant database or a database with a non-default instance number, select **Custom** from the **Port** drop-down menu.



3. Select the name of the on-premises data gateway to use for accessing the database.

NOTE

You must use an on-premises data gateway with this connector, whether your data is local or online.

4. Choose the authentication kind you want to use to access your data. You'll also need to enter a username and password.

NOTE

Currently, Power Query Online does not support Windows authentication. Windows authentication support is planned to become available in a few months.

5. Select **Next** to continue.
6. From the **Navigator** dialog box, you can either transform the data in the Power Query editor by selecting **Transform Data**, or load the data by selecting **Load**.

Supported features for SAP HANA

The following list shows the supported features for SAP HANA. Not all features listed here are supported in all implementations of the SAP HANA database connector.

- Both the Power BI Desktop and Excel connector for an SAP HANA database use the SAP ODBC driver to provide the best user experience.
- In Power BI Desktop, SAP HANA supports both DirectQuery and Import options.
- Power BI Desktop supports HANA information models, such as Analytic and Calculation Views, and has optimized navigation.
- With SAP HANA, you can also use SQL commands in the native database query SQL statement to connect to Row and Column Tables in HANA Catalog tables, which is not included in the Analytic/Calculation Views provided by the Navigator experience. You can also use the [ODBC connector](#) to query these tables.
- Power BI Desktop includes Optimized Navigation for HANA Models.

- Power BI Desktop supports SAP HANA Variables and Input parameters.
- Power BI Desktop supports HDI-container-based Calculation Views.
 - To access your HDI-container-based Calculation Views in Power BI, ensure that the HANA database users you use with Power BI have permission to access the HDI runtime container that stores the views you want to access. To grant this access, create a Role that allows access to your HDI container. Then assign the role to the HANA database user you'll use with Power BI. (This user must also have permission to read from the system tables in the _SYS_BI schema, as usual.) Consult the official SAP documentation for detailed instructions on how to create and assign database roles. This [SAP blog post](#) may be a good place to start.
 - There are currently some limitations for HANA variables attached to HDI-based Calculation Views. These limitations are because of errors on the HANA side. First, it isn't possible to apply a HANA variable to a shared column of an HDI-container-based Calculation View. To fix this limitation, upgrade to HANA 2 version 37.02 and onwards or to HANA 2 version 42 and onwards. Second, multi-entry default values for variables and parameters currently don't show up in the Power BI UI. An error in SAP HANA causes this limitation, but SAP hasn't announced a fix yet.

Next steps

- [Enable encryption for SAP HANA](#)

The following articles contain more information that you may find useful when connecting to an SAP HANA database.

- [Manage your data source - SAP HANA](#)
- [Use Kerberos for single sign-on \(SSO\) to SAP HANA](#)

Enable encryption for SAP HANA

1/15/2022 • 4 minutes to read • [Edit Online](#)

We recommend that you encrypt connections to an SAP HANA server from Power Query Desktop and Power Query Online. You can enable HANA encryption using both OpenSSL and SAP's proprietary CommonCryptoLib (formerly known as sapcrypto) library. SAP recommends using CommonCryptoLib, but basic encryption features are available using either library.

This article provides an overview of enabling encryption using OpenSSL, and references some specific areas of the SAP documentation. We update content and links periodically, but for comprehensive instructions and support, always refer to the official SAP documentation. If you want to set up encryption using CommonCryptoLib instead of OpenSSL, see [How to Configure TLS/SSL in SAP HANA 2.0](#) For steps on how to migrate from OpenSSL to CommonCryptoLib, see [SAP Note 2093286](#) (s-user required).

NOTE

The setup steps for encryption detailed in this article overlap with the setup and configuration steps for SAML SSO. Whether you choose OpenSSL or CommonCryptoLib as your HANA server's encryption provider, make sure that your choice is consistent across SAML and encryption configurations.

There are four phases to enabling encryption for SAP HANA using OpenSSL. We cover these phases next. For more information, see [Securing the Communication between SAP HANA Studio and SAP HANA Server through SSL](#).

Use OpenSSL

Ensure your HANA server is configured to use OpenSSL as its cryptographic provider. Replace the missing path information below with the server ID (sid) of your HANA server.

```
sslcryptoprocessor commoncrypto
sslforce false
sslinternalkeystore sapsrv_internal.pse
sslinternaltruststore sapsrv_internal.pse
sslinternalvalidatecertificate true
sslkeystore sapsrv.pse
sslmaxprotocolversion MAX
sslminprotocolversion TLS10
ssltruststore sapsrv.pse
sslcipherlist <sid>
```

Create a certificate signing request

Create an X509 certificate signing request for the HANA server.

1. Using SSH, connect to the Linux machine that the HANA server runs on as <sid>adm.
2. Go to the Home directory `/_usr/sap/<sid>/home`.
3. Create a hidden directory with the name `_.ssl` if one doesn't already exist.
4. Execute the following command:

```
openssl req -newkey rsa:2048 -days 365 -sha256 -keyout Server\_Key.pem -out Server\_Req.pem -nodes
```

This command creates a certificate signing request and private key. Once signed, the certificate is valid for a year (see the -days parameter). When prompted for the common name (CN), enter the fully qualified domain name (FQDN) of the computer the HANA server is installed on.

Get the certificate signed

Get the certificate signed by a certificate authority (CA) that is trusted by the client(s) you'll use to connect to the HANA server.

1. If you already have a trusted company CA (represented by CA_Cert.pem and CA_Key.pem in the following example), sign the certificate request by running the following command:

```
openssl x509 -req -days 365 -in Server\_Req.pem -sha256 -extfile /etc/ssl/openssl.cnf -extensions usr\_cert -CA CA\_Cert.pem -CAkey CA\_Key.pem -CAcreateserial -out Server\_Cert.pem
```

If you don't already have a CA you can use, you can create a root CA yourself by following the steps outlined in [Securing the Communication between SAP HANA Studio and SAP HANA Server through SSL](#).

2. Create the HANA server certificate chain by combining the server certificate, key, and the CA's certificate (the key.pem name is the convention for SAP HANA):

```
cat Server\_Cert.pem Server\_Key.pem CA\_Cert.pem \> key.pem
```

3. Create a copy of CA_Cert.pem named trust.pem (the trust.pem name is the convention for SAP HANA):

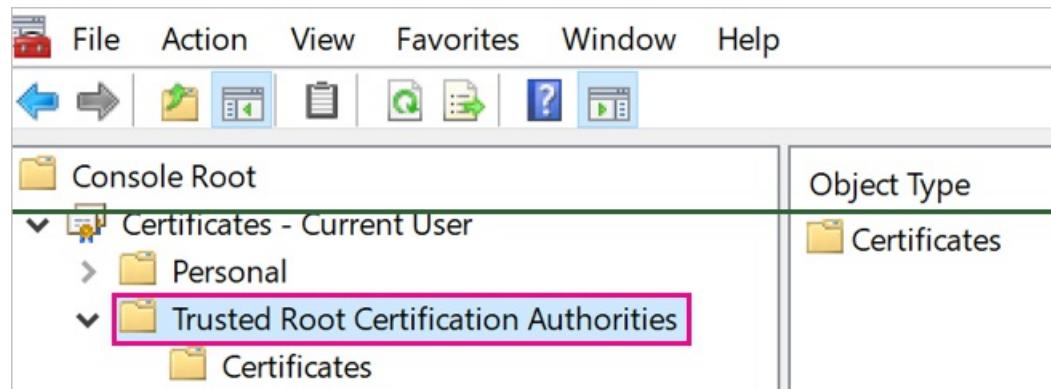
```
cp CA\_Cert.pem trust.pem
```

4. Restart the HANA server.

5. Verify the trust relationship between a client and the CA you used to sign the SAP HANA server's certificate.

The client must trust the CA used to sign the HANA server's X509 certificate before an encrypted connection can be made to the HANA server from the client's machine.

There are various ways to ensure this trust relationship exists using Microsoft Management Console (mmc) or the command line. You can import the CA's X509 certificate (trust.pem) into the **Trusted Root Certification Authorities** folder for the user that will establish the connection, or into the same folder for the client machine itself, if that is desirable.



You must first convert trust.pem into a .crt file before you can import the certificate into the Trusted Root Certification Authorities folder, for example by executing the following OpenSSL command:

```
openssl x509 -outform der -in your-cert.pem -out your-cert.crt
```

For information about using OpenSSL for the conversion, see the [OpenSSL documentation](#).

Test the connection

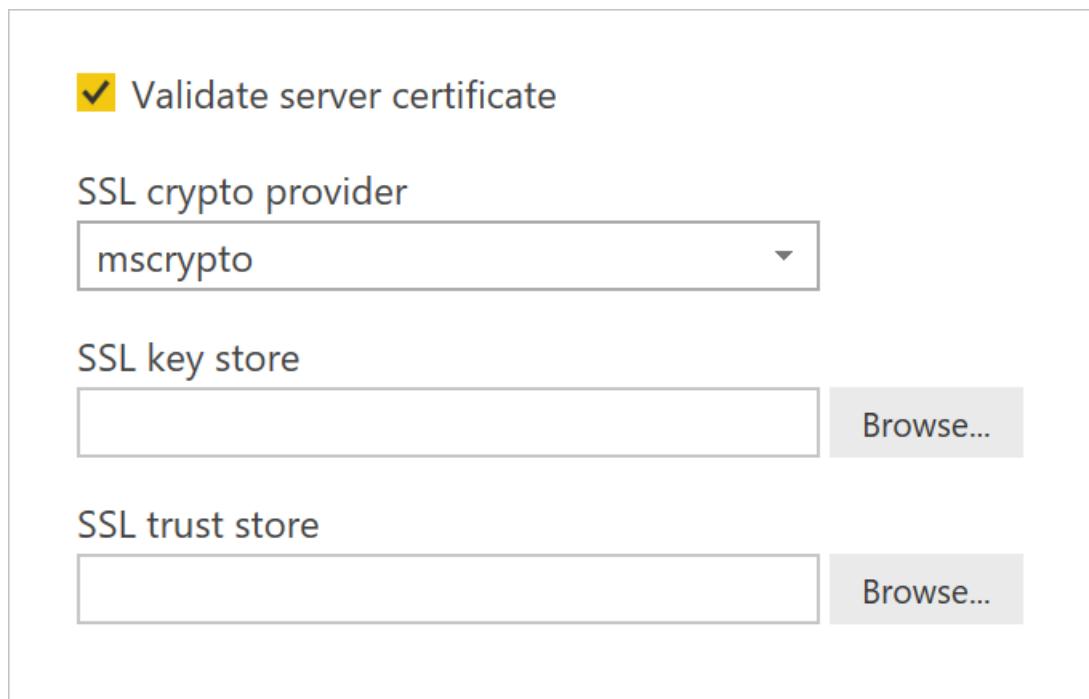
Before you can validate a server certificate in the Power BI service online, you must have a data source already set up for the [on-premises data gateway](#). If you don't already have a data source set up to test the connection, you'll have to create one. To set up the data source on the gateway:

1. From the Power BI service, select the  setup icon.
2. From the drop-down list, select **Manage gateways**.
3. Select the ellipsis (...) next to the name of the gateway you want to use with this connector.
4. From the drop-down list, select **Add data source**.
5. In **Data Source Settings**, enter the data source name you want to call this new source in the **Data Source Name** text box.
6. In **Data Source Type**, select **SAP HANA**.
7. Enter the server name in **Server**, and select the authentication method.
8. Continue following the instructions in the next procedure.

Test the connection in Power BI Desktop or the Power BI service.

1. In Power BI Desktop or in the **Data Source Settings** page of the Power BI service, ensure that **Validate server certificate** is enabled before attempting to establish a connection to your SAP HANA server. For **SSL crypto provider**, select **mscrypto** if you've followed the OpenSSL setup steps and **commoncrypto** if you've configured that library as your crypto provider. Leave the **SSL key store** and **SSL trust store** fields blank.

- Power BI Desktop



- Power BI service

Validate Server Certificate
Yes
SSL Crypto Provider
mscrypto
Key Store
Only applicable when validating SSL server certificate without mscrypto
Trust Store (on Gateway machine)
Only applicable when validating SSL server certificate without mscrypto

2. Verify that you can successfully establish an encrypted connection to the server with the **Validate server certificate** option enabled, by loading data in Power BI Desktop or refreshing a published report in Power BI service.

You'll note that only the **SSL crypto provider** information is required. However, your implementation might require that you also use the key store and trust store. For more information about these stores and how to create them, see [Client-Side TLS/SSL Connection Properties \(ODBC\)](#).

Additional information

- [Server-Side TLS/SSL Configuration Properties for External Communication \(JDBC/ODBC\)](#)

Next steps

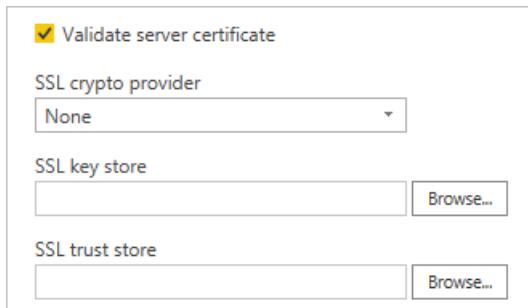
- [Configure SSL for ODBC client access to SAP HANA](#)

Configure SSL for ODBC client access to SAP HANA

1/15/2022 • 2 minutes to read • [Edit Online](#)

If you're connecting to an SAP HANA database from Power Query Online, you may need to set up various property values to connect. These properties could be the SSL crypto provider, an SSL key store, and an SSL trust store. You may also require that the connection be encrypted. In this case, you can use the ODBC Data Source Administrator application supplied with Windows to set up these properties.

In Power BI Desktop and Excel, you can set up these properties when you first sign in using the Power Query SAP HANA database connector. The **Validate server certificate** selection in the authentication dialog box is enabled by default. You can then enter values in the **SSL crypto provider**, **SSL key store**, and **SSL trust store** properties in this dialog box. However, all of the validate server certificate selections in the authentication dialog box in Power BI Desktop and Excel are optional. They're optional in case you want to use ODBC Data Source Administrator to set them up at the driver level.



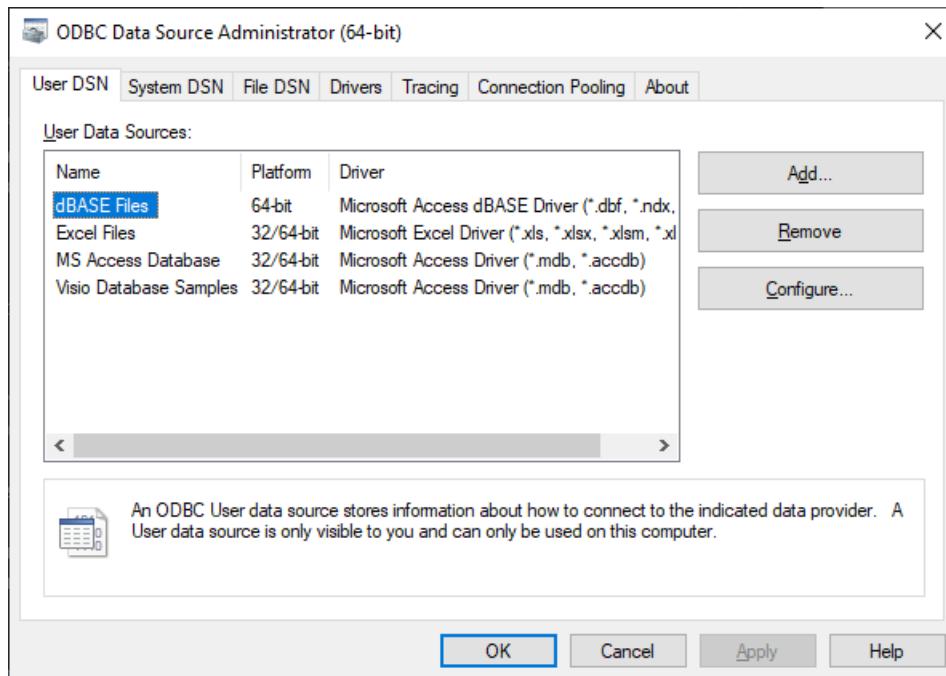
NOTE

You must have the proper SAP HANA ODBC driver (32-bit or 64-bit) installed before you can set these properties in ODBC Data Source Administrator.

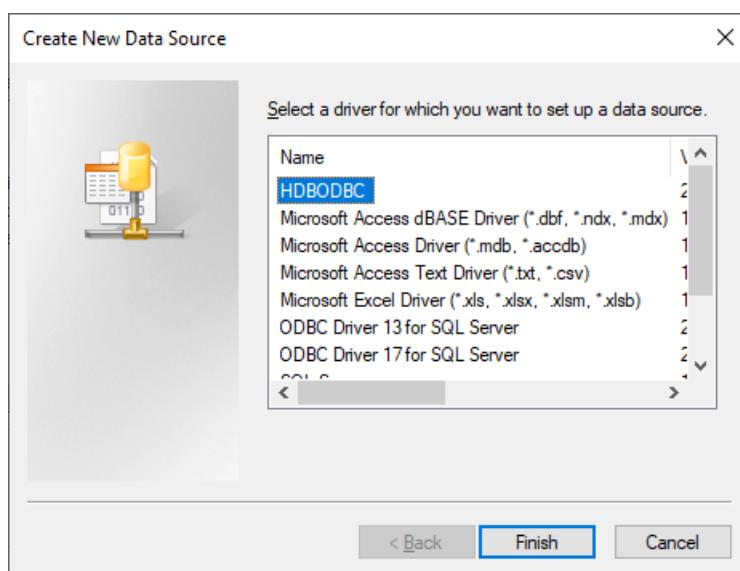
If you're going to use ODBC Data Source Administrator to set up the SSL crypto provider, SSL key store, and SSL trust store in Power BI or Excel, clear the **Validate server certificate** check box when presented with the authentication dialog box.

To use ODBC Data Source Administrator to set up the validate server certificate selections:

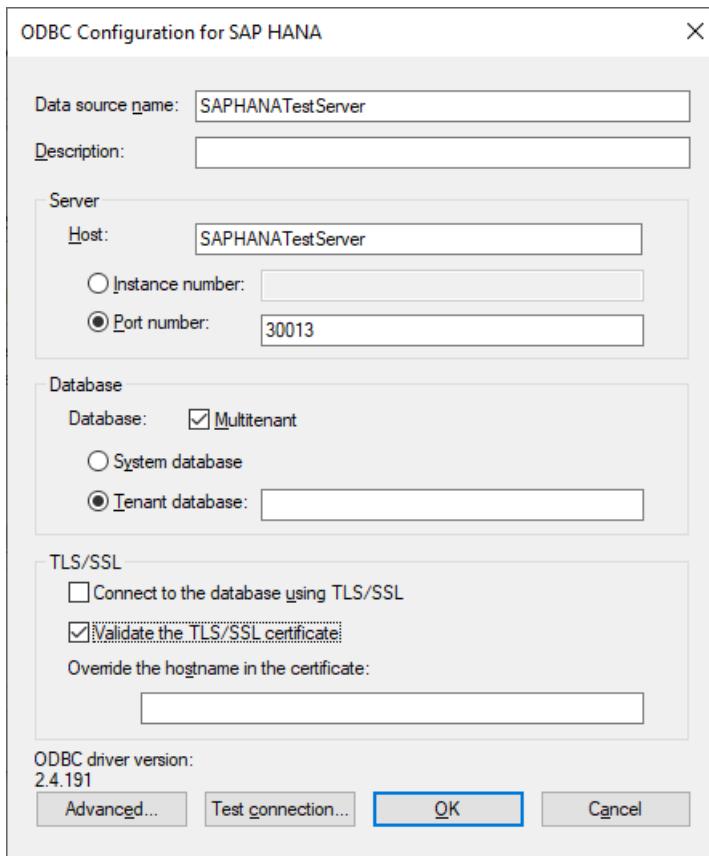
1. From the Windows Start menu, select **Windows Administrative Tools > ODBC Data Sources**. If you're using a 32-bit version of Power BI Desktop or Excel, open ODBC Data Sources (32-bit), otherwise open ODBC Data Sources (64-bit).



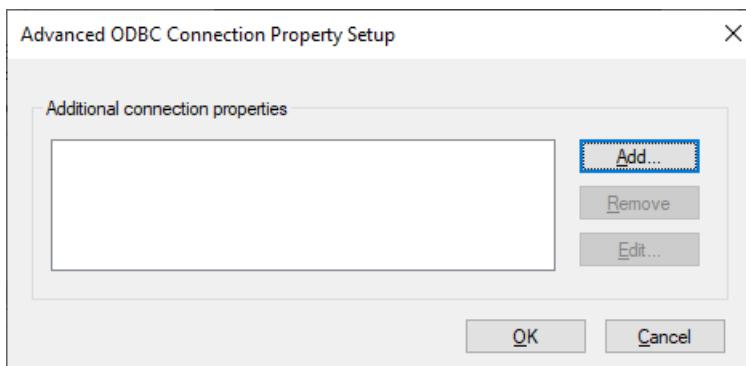
2. In the User DSN tab, select Add.
3. In the Create New Data Source dialog box, select the HDBODBC driver, and then select Finish.



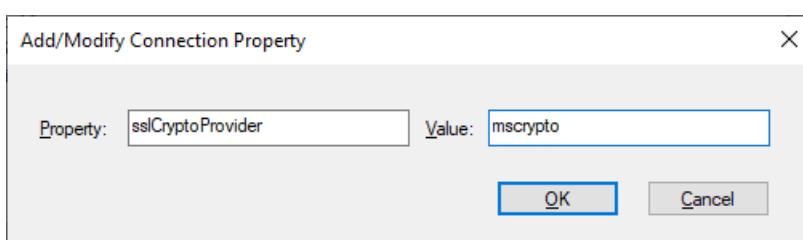
4. In the ODBC Configuration for SAP HANA dialog box, enter a Data source name. Then enter your server and database information, and select Validate the TLS/SSL certificate.



5. Select the **Advanced** button.
6. In the **Advanced ODBC Connection Property Setup** dialog box, select the **Add** button.



7. In the **Add/Modify Connection Property** dialog box, enter **sslCryptoProvider** in the **Property** text box.
8. In the **Value** text box, enter the name of the crypto provider you'll be using: either **sapcrypto**, **commoncrypto**, **openssl**, or **mscrypto**.



9. Select **OK**.
10. You can also add the optional **sslKeyStore** and **sslTrustStore** properties and values if necessary. If the connection must be encrypted, add **ENCRYPT** as the property and **TRUE** as the value.
11. In the **Advanced ODBC Connection Property Setup** dialog box, select **OK**.

12. To test the connection you've set up, select **Test connection** in the ODBC Configuration for SAP HANA dialog box.

13. When the test connection has completed successfully, select **OK**.

For more information about the SAP HANA connection properties, see [Server-Side TLS/SSL Configuration Properties for External Communication \(JDBC/ODBC\)](#).

NOTE

If you select **Validate server certificate** in the SAP HANA authentication dialog box in Power BI Desktop or Excel, any values you enter in **SSL crypto provider**, **SSL key store**, and **SSL trust store** in the authentication dialog box will override any selections you've set up using ODBC Data Source Administrator.

Next steps

- [SAP HANA database connector troubleshooting](#)

Troubleshooting

1/15/2022 • 2 minutes to read • [Edit Online](#)

The following section describes some issues that may occur while using the Power Query SAP HANA connector, along with some possible solutions.

Known issues and limitations

There are a few limitations to using SAP HANA, shown below:

- NVARCHAR strings are truncated to a maximum length of 4000 Unicode characters.
- SMALLDECIMAL isn't supported.
- VARBINARY isn't supported.
- Valid Dates are between 1899/12/30 and 9999/12/31.

Error: This connector requires one or more additional components to be installed

The connector looks for the driver in the registry, so if the driver wasn't properly installed it won't show up.

The registry key is:

```
HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI\ODBC Drivers
```

If you're on a 64-bit machine, but Excel or Power BI Desktop is 32-bit (like the screenshots below), you can check for the driver in the WOW6432 node instead:

```
HKEY_LOCAL_MACHINE\Software\WOW6432Node\ODBC\ODBCINST.INI\ODBC Drivers
```

Note that the driver needs to match the bit version of your Excel or Power BI Desktop. If you're using:

- 32-bit Excel/Power BI Desktop, you'll need the 32-bit ODBC driver (HDBODBC32).
- 64-bit Excel/Power BI Desktop, you'll need the 64-bit ODBC driver (HDBODBC).

The driver is usually installed by running hdbsetup.exe.

Finally, the driver should also show up as "ODBC DataSources 32-bit" or "ODBC DataSources 64-bit".

Collect SAP HANA ODBC Driver traces

To capture an SAP HANA trace:

1. Open a command-line window.
2. Depending on your installation, you may need to go to C:\Program Files instead of C:\Program Files (x86). The command might also be hdbodbc_cons.exe instead of hdbodb_cons32.exe.
3. Type the following commands:

```
cd C:\Program Files (x86)\sap\hdbclient  
hdbodbc_cons32.exe config trace api on  
hdbodbc_cons32.exe config trace sql on  
hdbodbc_cons32.exe config trace debug on  
hdbodbc_cons32.exe config trace short on
```

```
hdbodbc_cons32.exe config trace packet 99999999999999  
hdbodbc_cons32.exe config trace filename D:\tmp\odbctraces\hana-%p.html  
hdbodbc_cons32.exe trace refresh  
hdbodbc_cons32.exe show all
```

4. Open Power BI, clear the cache, and rerun the scenario.

5. Once done, zip up the traces:

- From the **Log File Path** in the **Tracing** tab of the ODBC Data Source Administrator.
- From the HANA trace based on the path configured with the command **hdbodbc_cons32.exe config trace filename**.

6. Disable tracing by using the following command:

```
hdbodbc_cons.exe trace off
```

When capturing an SAP HANA trace, note the following considerations:

- The trace commands should be run as the user that will be running the Mashup process that accesses the SAP HANA server.
- The trace file path you specify should be writable by the user that runs the Mashup process.

For example:

- To capture non-SSO connections from gateway, make sure you use the gateway service user. That is, run the command-line window as the gateway user when you want to execute the hdbodbc_cons.exe calls. Make sure that the gateway server user can write to the log file location you specify.
- To capture SSO connections from Gateway, use the SSO user.

SAP HANA: insufficient privilege

This message might be because of:

- The user legitimately not having enough privileges on the view they're trying to access.
- The following known issue:

Issue: Not able to connect to SAP Hana from PBI Desktop using SAP client 2.0 37.02, but if you downgrade the client version to 1.00.120.128, it works.

```
ERROR MESSAGE: External error: ERROR [S1000] [SAP AG][LIBODBCHDB DLL][HDBODBC] General error;258  
insufficient privilege: [2950] user is not authorized
```

- Response from SAP:

```
*****  
The issue you are experiencing is a regression of the bug which was applied on HANA 1.0 122.29, HANA 2.0 37.02 and 42.  
There was a problem on copying session variables among nodes, so we have changed a behavior.  
From this change, when there was a change on session variables, session user and session schema information are also applied on the another node.  
So the query in your case was executed as INVMD1 user, but its user was changed to _SYS_REPO on the another node.  
Since _SYS_REPO user does not have a privilege, the issue occurred.
```

To avoid this issue(or a workaround), there should be no session variable changes in between query executions.
From the trace logs that you provided, we see a session variable 'client' was set.
If this variable is not necessary for the query execution, removing it can be a workaround.
However, if this session variable is necessary for the query execution, this workaround cannot be applied.

We are still struggling to make a patch on this issue, but it should be delivered to the next SPS03 revision.(our targeting revision)
This can be changed due to a schedule and we will inform you if there is a change on the schedule.

Unfortunately, this is an SAP issue so you'll need to wait for a fix from SAP.

SharePoint folder

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous Microsoft Account Windows
Function Reference Documentation	SharePoint.Contents SharePoint.Files

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

NOTE

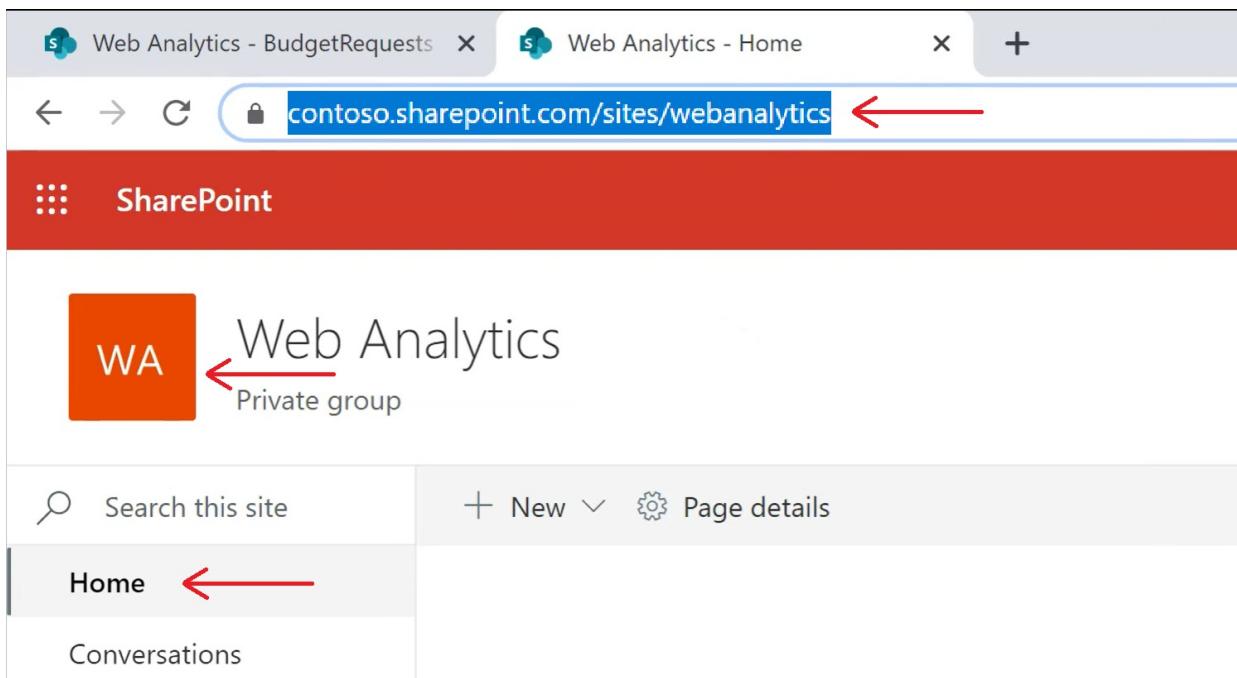
AAD/OAuth for SharePoint on-premises isn't supported using the on-premises data gateway.

Capabilities supported

- Folder path
- Combine
 - Combine and load
 - Combine and transform

Determine the site URL

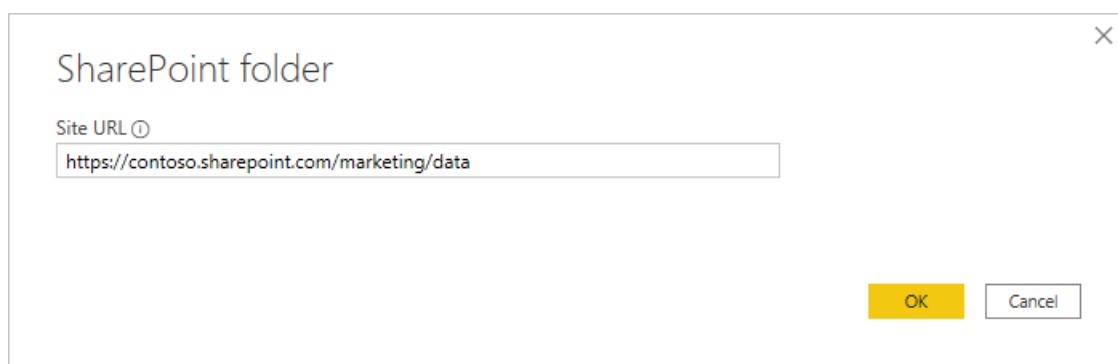
When you're connecting to a SharePoint site, you'll be asked to enter the site URL. To find the site URL that contains your SharePoint folder, first open a page in SharePoint. From a page in SharePoint, you can usually get the site address by selecting **Home** in the navigation pane, or the icon for the site at the top. Copy the address from your web browser's address bar and save for later.



Connect to a SharePoint folder from Power Query Desktop

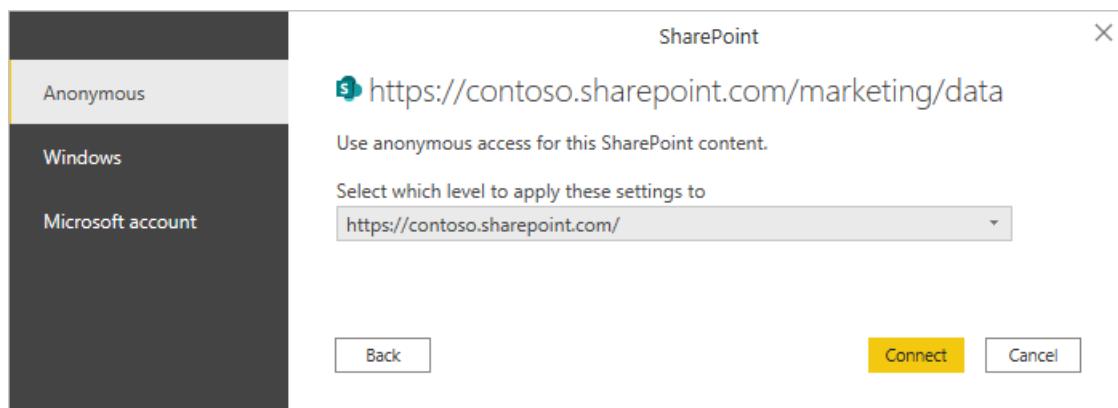
To connect to a SharePoint folder:

1. From **Get Data**, select **SharePoint folder**.
2. Paste the SharePoint site URL you copied in [Determine the site URL](#) to the **Site URL** text box in the **SharePoint folder** dialog box. In this example, the site URL is `https://contoso.sharepoint.com/marketing/data`. If the site URL you enter is invalid, a warning icon will appear next to the URL text box.



Select **OK** to continue.

3. If this is the first time you've visited this site address, select the appropriate authentication method. Enter your credentials and choose which level to apply these settings to. Then select **Connect**.



For more information about authentication methods and level settings, go to [Authentication with a data source](#).

- When you select the SharePoint folder you want to use, the file information about all of the files in that SharePoint folder are displayed. In addition, file information about any files in any subfolders is also displayed.

The screenshot shows a Power Query Editor window with the URL <https://contoso.sharepoint.com/marketing/data>. Below the URL is a table with the following data:

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	
Binary	Financial Sample 1.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:43:47 AM	4/29/2020 9:15:41 AM	Record	https://
Binary	Financial Sample 2.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:45:00 AM	4/29/2020 9:15:41 AM	Record	https://
Binary	Financial Sample 3.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:46:10 AM	4/29/2020 9:15:41 AM	Record	https://
Binary	Financial Sample 4.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:47:23 AM	4/29/2020 9:15:41 AM	Record	https://
Binary	Financial Sample 5.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:28 AM	4/29/2020 9:15:41 AM	Record	https://
Binary	Financial Sample 6.csv	.csv	4/29/2020 9:15:41 AM	4/28/2020 8:48:49 AM	4/29/2020 9:15:41 AM	Record	https://

At the bottom of the window are buttons: **Combine**, **Load**, **Transform Data**, and **Cancel**.

- Select **Combine & Transform Data** to combine the data in the files of the selected SharePoint folder and load the data into the Power Query Editor for editing. Or select **Combine & Load** to load the data from all of the files in the SharePoint folder directly into your app.

The screenshot shows a Power Query Editor window with the same table of CSV files. The **Combine** button is highlighted. A dropdown menu is open below it, showing three options: **Combine & Transform Data**, **Combine & Load**, and **Load**.

NOTE

The **Combine & Transform Data** and **Combine & Load** buttons are the easiest ways to combine data found in the files of the SharePoint folder you specify. You could also use the **Load** button or the **Transform Data** buttons to combine the files as well, but that requires more manual steps.

Connect to a SharePoint folder from Power Query Online

To connect to a SharePoint folder:

- From the **Data sources** page, select **SharePoint folder**.
- Paste the SharePoint site URL you copied in [Determine the site URL](#) to the **Site URL** text box in the **SharePoint folder** dialog box. In this example, the site URL is <https://contoso.sharepoint.com/marketing/data>.

SharePoint folder

File

Learn more

Connection settings

Site URL *

https://contoso.sharepoint.com/marketing/data

Connection credentials

Data gateway

(none)

Authentication kind

Microsoft account

You are not signed in. Please sign in.

Sign in

3. If the SharePoint folder is on-premises, enter the name of an on-premises data gateway.
4. Select the authentication kind, and enter any credentials that are required.
5. Select **Next**.
6. When you select the SharePoint folder you want to use, the file information about all of the files in that SharePoint folder are displayed. In addition, file information about any files in any subfolders is also displayed.

Power Query - Preview folder data

https://contoso.sharepoint.com/marketing/data

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes
[Binary]	Financial Sample 4....	.csv	null	8/11/2021, 2:55:35 ...	8/11/2021, 2:55:35 ...	[Record] https://
[Binary]	Financial Sample 3....	.csv	null	8/11/2021, 2:55:35 ...	8/11/2021, 2:55:35 ...	[Record] https://
[Binary]	Financial Sample 2....	.csv	null	8/11/2021, 2:55:35 ...	8/11/2021, 2:55:35 ...	[Record] https://
[Binary]	Financial Sample 1....	.csv	null	8/11/2021, 2:55:35 ...	8/11/2021, 2:55:35 ...	[Record] https://
[Binary]	Financial Sample 5....	.csv	null	8/11/2021, 2:55:36 ...	8/11/2021, 2:55:36 ...	[Record] https://
[Binary]	Financial Sample 6....	.csv	null	8/11/2021, 2:55:36 ...	8/11/2021, 2:55:36 ...	[Record] https://

Back Cancel Combine Transform data

7. Select **Combine** to combine the data in the files of the selected SharePoint folder and load the data into the Power Query Editor for editing.

NOTE

The **Combine** button is the easiest way to combine data found in the files of the SharePoint folder you specify. You could also use the **Transform Data** buttons to combine the files as well, but that requires more manual steps.

Troubleshooting

Combining files

All of the files in the SharePoint folder you select will be included in the data to be combined. If you have data

files located in a subfolder of the SharePoint folder you select, all of these files will also be included. To ensure that combining the file data works properly, make sure that all of the files in the folder and the subfolders have the same schema.

In some cases, you might have multiple folders on your SharePoint site containing different types of data. In this case, you'll need to delete the unnecessary files. To delete these files:

1. In the list of files from the SharePoint folder you chose, select **Transform Data**.

The screenshot shows a 'Transform Data' dialog box. At the top, the URL 'https://contoso.sharepoint.com/marketing/data' is displayed. Below it is a table with columns: Content, Name, Extension, Date accessed, Date modified, Date created, and Attributes. The table lists various files like 'Open Notebook.onetoc2', 'Meetings.one', etc. A note at the bottom says 'The data in the preview has been truncated due to size limits.' At the bottom right, there are buttons for 'Combine', 'Load', 'Transform Data' (which is highlighted with a red box), and 'Cancel'.

2. In the Power Query editor, scroll down to find the files you want to keep.

The screenshot shows the Power Query Editor interface. The ribbon tabs include File, Home, Transform, Add Column, and View. The main area displays a table of files from 'SharePoint.Files'. A 'Query Settings' pane on the right shows 'Name: Query1' and 'Source' under 'APPLIED STEPS'. The 'Transform' tab is selected in the ribbon. A note at the bottom says 'PREVIEW DOWNLOADED AT 2:43 PM'.

3. In the example shown in the screenshot above, the required files are the last rows in the table. Select **Remove Rows**, enter the value of the last row before the files to keep (in this case 903), and select **OK**.

The screenshot shows the Power Query Editor interface with the 'Home' ribbon selected. In the 'Transform' tab, the 'Remove Rows' button is highlighted with a red box. A modal dialog box titled 'Remove Top Rows' is open, asking 'Specify how many rows to remove from the top.' A text input field contains the value '903'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

- Once you've removed all the unnecessary files, select **Combine Files** from the **Home** ribbon to combine the data from all of the remaining files.

For more information about combining files, go to [Combine files in Power Query](#).

Filename special characters

If a filename contains certain special characters, it may lead to authentication errors because of the filename being truncated in the URL. If you're getting unusual authentication errors, make sure all of the filenames you're using don't contain any of the following special characters.

% \$

If these characters are present in the filename, the file owner must rename the file so that it does NOT contain any of these characters.

SharePoint list

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Anonymous Windows Microsoft Account
Function Reference Documentation	SharePoint.Contents SharePoint.Files SharePoint.Tables

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

NOTE

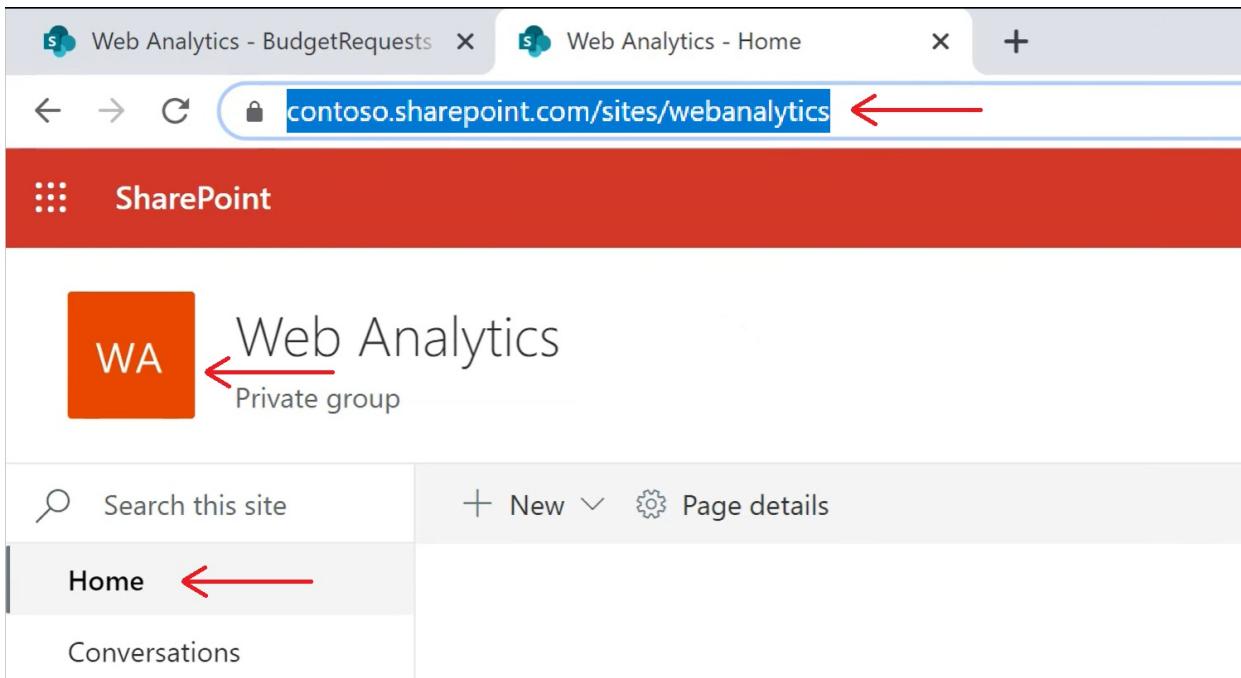
AAD/OAuth for SharePoint on-premises isn't supported using the on-premises data gateway.

Capabilities supported

- Site URL

Determine the site URL

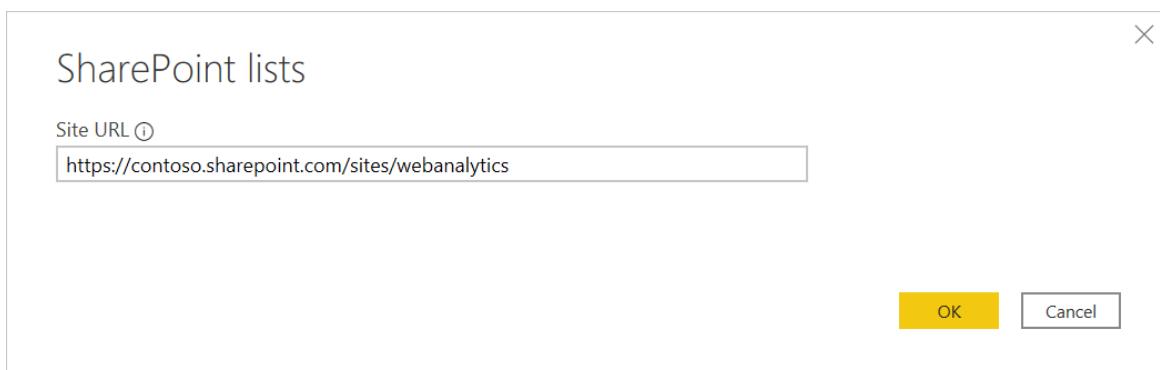
When you're connecting to a SharePoint site, you'll be asked to enter the site URL. To find the site URL that contains your SharePoint list, first open a page in SharePoint. From a page in SharePoint, you can usually get the site address by selecting **Home** in the navigation pane, or the icon for the site at the top. Copy the address from your web browser's address bar and save for later.



Connect to a SharePoint list from Power Query Desktop

To connect to a SharePoint list:

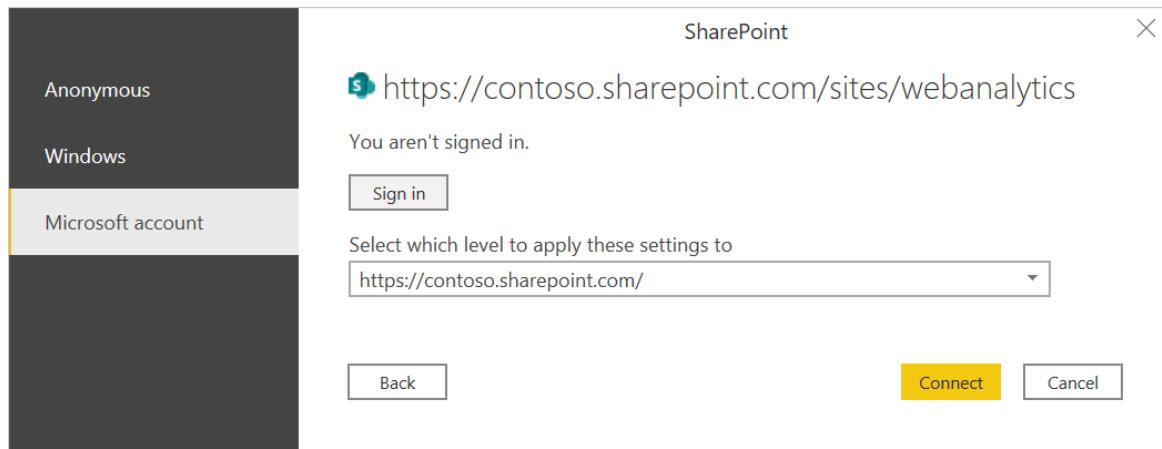
1. From **Get Data**, select **SharePoint List**.
2. Paste the SharePoint site URL you copied in [Determine the site URL](#) to the **Site URL** field in the open dialog box.



If the URL address you enter is invalid, a warning icon will appear next to the **Site URL** textbox.

Select **OK** to continue.

3. If this is the first time you've visited this site address, select the appropriate authentication method. Enter your credentials and chose which level to apply these settings to. Then select **Connect**.



For more information about authentication methods and level settings, go to [Authentication with a data source](#).

4. From the **Navigator**, you can select a location, then either transform the data in the Power Query editor by selecting **Transform Data**, or load the data by selecting **Load**.

A screenshot of the Power Query Navigator interface. On the left, there is a tree view of SharePoint sites and lists. The "Site Pages" node under "https://contoso.sharepoint.com/site/webanalytics" is selected, indicated by a checked checkbox. The right side shows a preview of the "Site Pages" data in a table format. The table has four columns: "FileSystemObjectType", "Id", "ServerRedirectedEmbedUri", and "ServerRedirectedEmbed". The data consists of 12 rows, each with an "Id" value from 1 to 12 and a "ServerRedirectedEmbedUri" value of "null". A note below the table says "The data in the preview has been truncated due to size limits." At the bottom are three buttons: "Load" (highlighted in yellow), "Transform Data", and "Cancel".

Connect to a SharePoint list from Power Query Online

To connect to a SharePoint list:

1. From the **Data sources** page, select **SharePoint list**.
2. Paste the SharePoint site URL you copied in [Determine the site URL](#) to the **Site URL** field in the open dialog box.

SharePoint list

Connection settings

Site URL *
https://contoso.sharepoint.com/sites/webanalytics

Connection credentials

Data gateway
(none)

Authentication kind
Anonymous

3. Enter the name of an on-premises data gateway if needed.
4. Select the authentication kind, and enter any credentials that are required.
5. Select **Next**.
6. From the **Navigator**, you can select a location, then transform the data in the Power Query editor by selecting **Next**.

ABC 123 FileSystemObjectType	ABC 123 Id	ABC 123 ServerRedirectedEmbedUri	ABC 123 ServerRedirectedEmbedUrl	ABC 123 ID	ABC 123 ContentType
0	1		null	1	0x0100C5A270CF473
0	2		null	2	0x0100C5A270CF473
0	3		null	3	0x0100C5A270CF473
0	4		null	4	0x0100C5A270CF473
0	5		null	5	0x0100C5A270CF473
0	6		null	6	0x0100C5A270CF473
0	7		null	7	0x0100C5A270CF473
0	8		null	8	0x0100C5A270CF473
0	9		null	9	0x0100C5A270CF473
0	10		null	10	0x0100C5A270CF473
0	11		null	11	0x0100C5A270CF473
0	12		null	12	0x0100C5A270CF473
0	13		null	13	0x0100C5A270CF473
0	14		null	14	0x0100C5A270CF473
0	15		null	15	0x0100C5A270CF473
0	16		null	16	0x0100C5A270CF473
0	17		null	17	0x0100C5A270CF473

Power Query - Choose data

MarketingData

Display options ▾

Back Cancel Next

Troubleshooting

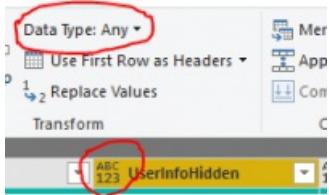
Use root SharePoint address

Make sure you supply the root address of the SharePoint site, without any subfolders or documents. For example, use link similar to the following: <https://contoso.sharepoint.com/teams/ObjectModel/>

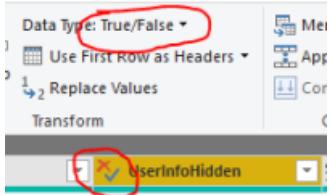
Inconsistent behavior around boolean data

When using the SharePoint list connector, Boolean values are represented inconsistently as TRUE/FALSE or 1/0 in Power BI Desktop and Power BI service environments. This may result in wrong data, incorrect filters, and empty visuals.

This issue only happens when the **Data Type** is not explicitly set for a column in the Query View of Power BI Desktop. You can tell that the data type isn't set by seeing the "ABC 123" image on the column and "Any" data type in the ribbon as shown below.



The user can force the interpretation to be consistent by explicitly setting the data type for the column through the Power Query Editor. For example, the following image shows the column with an explicit Boolean type.



Using OData to access a SharePoint List

If you use an OData feed to access a SharePoint List, there's an approximately 2100 character limitation to the URL you use to connect. More information: [Maximum URL length](#)

Next steps

[Optimize Power Query when expanding table columns](#)

SharePoint Online list

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous Windows Microsoft Account
Function Reference Documentation	SharePoint.Contents SharePoint.Files SharePoint.Tables

NOTE

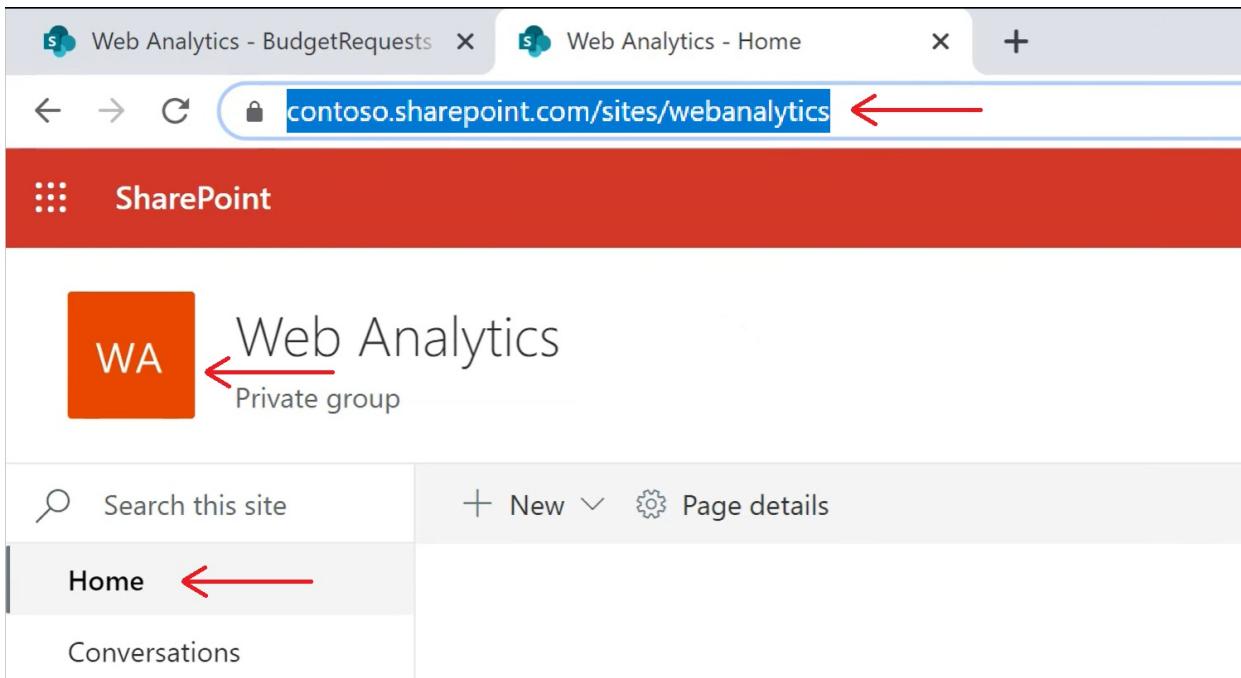
Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Capabilities supported

- Site URL

Determine the site URL

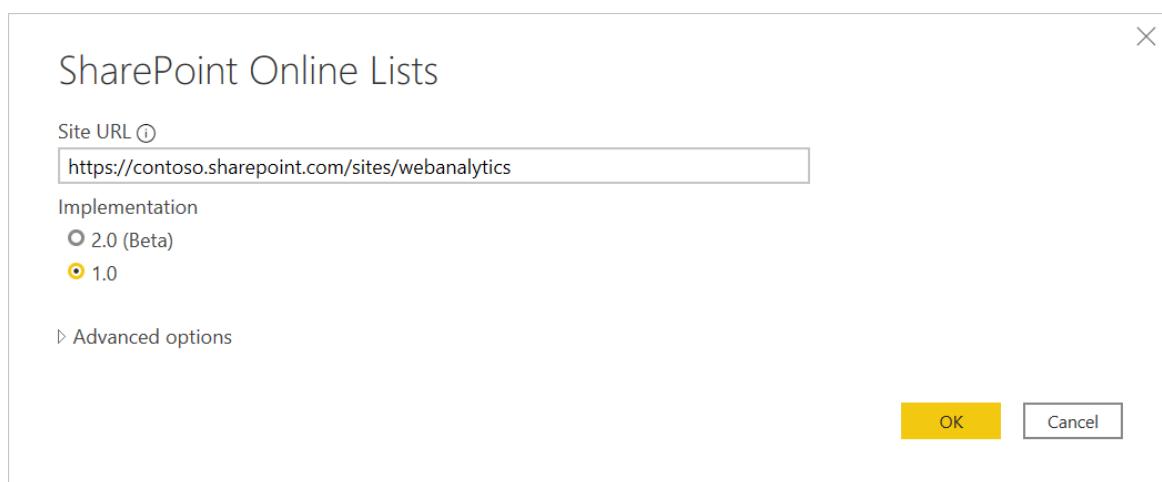
When you're connecting to a SharePoint site, you'll be asked to enter the site URL. To find the site URL that contains your SharePoint Online list, first open a page in SharePoint. From a page in SharePoint, you can usually get the site address by selecting **Home** in the navigation pane, or the icon for the site at the top. Copy the address from your web browser's address bar and save for later.



Connect to a SharePoint Online list from Power Query Desktop

To connect to a SharePoint Online list:

1. From Get Data, select SharePoint Online list.
2. Paste the SharePoint site URL you copied in [Determine the site URL](#) to the Site URL field in the open dialog box.

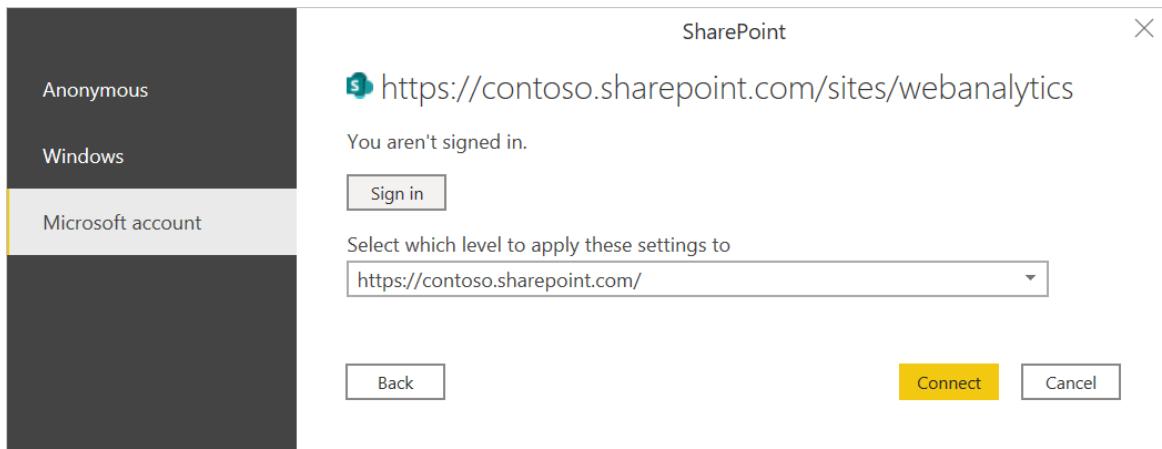


If the URL address you enter is invalid, a warning icon will appear next to the Site URL textbox.

You can also select either the 1.0 implementation of this connector or the beta 2.0 implementation. More information: [Connect to SharePoint Online list v2.0 \(Beta\)](#)

Select OK to continue.

3. If this is the first time you've visited this site address, select the appropriate authentication method. Enter your credentials and chose which level to apply these settings to. Then select Connect.



For more information about authentication methods and level settings, go to [Authentication with a data source](#).

4. From the **Navigator**, you can select a location, then either transform the data in the Power Query editor by selecting **Transform Data**, or load the data by selecting **Load**.

A screenshot of the Power Query Navigator interface. On the left, there is a tree view of a SharePoint site structure under the URL https://contoso.sharepoint.com/sites/webanalytics. The "MarketingData" list is selected and highlighted with a yellow checkmark. On the right, the preview pane shows a table titled "MarketingData" with columns: FileSystemObjectType, Id, ServerRedirectedEmbedUri, ServerRedirectedEmbedUrl, ID.1, and ContentTypeId. The data consists of 10 rows, each with an Id from 1 to 10. A note at the bottom of the preview pane states: "The data in the preview has been truncated due to size limits." At the bottom of the interface, there are three buttons: "Load" (highlighted in yellow), "Transform Data", and "Cancel".

Connect to a SharePoint Online list from Power Query Online

To connect to a SharePoint Online list:

1. From the **Data sources** page, select **SharePoint Online list**.
2. Paste the SharePoint site URL you copied in [Determine the site URL](#) to the **Site URL** field in the open dialog box.

The screenshot shows the 'Connection settings' page for a SharePoint Online list. On the left, there's a sidebar with a 'SharePoint Online list' icon and the text 'Online services'. Below that is a 'Learn more' link. The main area has 'Site URL *' set to 'https://contoso.sharepoint.com/sites/webanalytics'. Under 'Connection credentials', it shows 'Data gateway' as '(none)' and 'Authentication kind' as 'Microsoft account'. A message says 'You are not signed in. Please sign in.' with a 'Sign in' button.

3. Enter the name of an on-premises data gateway if needed.
4. Select the authentication kind, and enter any credentials that are required.
5. Select **Next**.
6. From the **Navigator**, you can select a location, then transform the data in the Power Query editor by selecting **Transform data**.

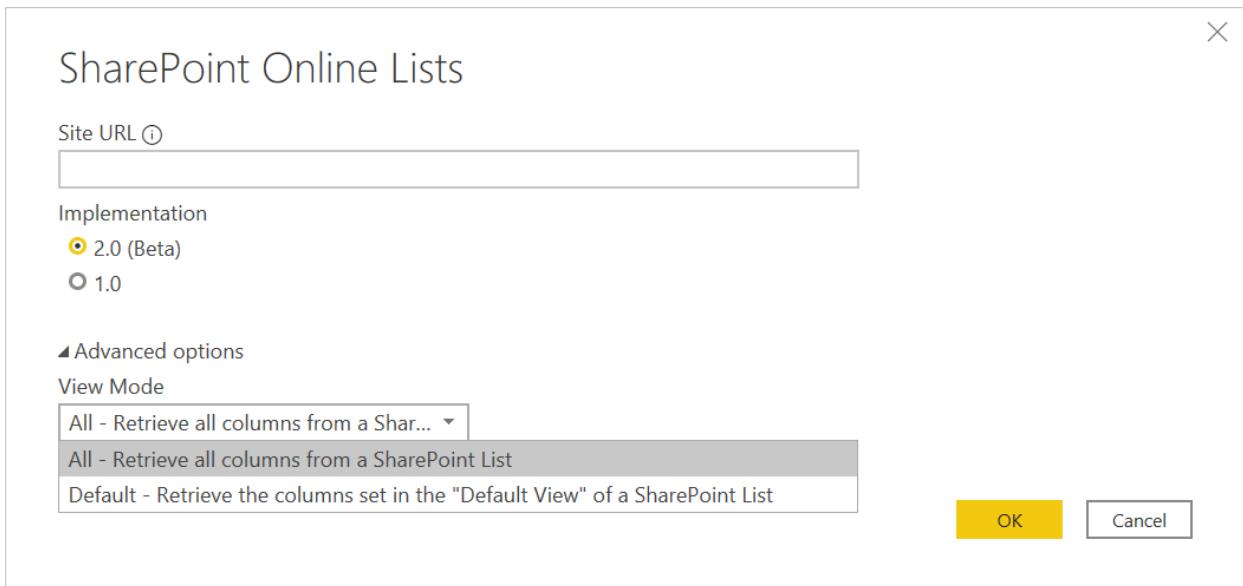
The screenshot shows the 'Power Query - Choose data' dialog. On the left is a 'Navigator' pane listing various SharePoint Online list items like 'SharePoint Online list [21]', 'appdata', 'appfiles', etc., with 'MarketingData' selected. The main area shows a table titled 'MarketingData' with columns: FileSystemObjectType, Id, ServerRedirectedEmbedUri, ServerRedirectedEmbedUrl, ID, and ContentType. The table contains 18 rows of data. At the bottom are 'Back', 'Cancel', and 'Transform data' buttons.

FileSystemObjectType	Id	ServerRedirectedEmbedUri	ServerRedirectedEmbedUrl	ID	ContentType
	0	null		1	0x0100C5A270CF4
	0	null		2	0x0100C5A270CF4
	0	null		3	0x0100C5A270CF4
	0	null		4	0x0100C5A270CF4
	0	null		5	0x0100C5A270CF4
	0	null		6	0x0100C5A270CF4
	0	null		7	0x0100C5A270CF4
	0	null		8	0x0100C5A270CF4
	0	null		9	0x0100C5A270CF4
	0	null		10	0x0100C5A270CF4
	0	null		11	0x0100C5A270CF4
	0	null		12	0x0100C5A270CF4
	0	null		13	0x0100C5A270CF4
	0	null		14	0x0100C5A270CF4
	0	null		15	0x0100C5A270CF4
	0	null		16	0x0100C5A270CF4
	0	null		17	0x0100C5A270CF4

Connect to SharePoint Online list v2.0 (Beta)

In the October 2020 release of Power BI Desktop, we introduced an updated version of the SharePoint Online list connector. This connector has improved APIs and greater usability, but isn't backwards compatible with usage of the 1.0 connector version.

To access it, you'll enter the same connector screen through step 2 in [Connect to a SharePoint Online list from Power Query Desktop](#). However, make sure you select **2.0 (Beta)** under **Implementation** if it isn't already selected.



With this update to the connector, we're making available two different views for the same data:

- **All**
- **Default**

The **All** view includes all user created and system defined columns. You can see what columns are included in the following screen.

View Name:

Default Items

Web address of this view:

<https://contoso.sharepoint.com/sites/webanalytics/Lists/Sample List/DefaultItems.aspx> 

This view appears by default when visitors follow a link to this list. If you want to delete this view, first make another view the default.

Display	Column Name	Position from Left
<input checked="" type="checkbox"/>	Title	1 ▾
<input checked="" type="checkbox"/>	Sample Column	2 ▾
<input type="checkbox"/>	App Created By	3 ▾
<input type="checkbox"/>	App Modified By	4 ▾
<input type="checkbox"/>	Attachments	5 ▾
<input type="checkbox"/>	Compliance Asset Id	6 ▾
<input type="checkbox"/>	Content Type	7 ▾
<input type="checkbox"/>	Created	8 ▾
<input type="checkbox"/>	Created By	9 ▾
<input type="checkbox"/>	Edit (link to edit item)	10 ▾
<input type="checkbox"/>	Folder Child Count	11 ▾
<input type="checkbox"/>	ID	12 ▾
<input type="checkbox"/>	Item Child Count	13 ▾
<input type="checkbox"/>	Item is a Record	14 ▾
<input type="checkbox"/>	Label applied by	15 ▾
<input type="checkbox"/>	Label setting	16 ▾
<input type="checkbox"/>	Modified	17 ▾
<input type="checkbox"/>	Modified By	18 ▾
<input type="checkbox"/>	Retention label	19 ▾
<input type="checkbox"/>	Retention label Applied	20 ▾
<input type="checkbox"/>	Title	21 ▾
<input type="checkbox"/>	Title (linked to item)	22 ▾
<input type="checkbox"/>	Type (icon linked to document)	23 ▾
<input type="checkbox"/>	Version	24 ▾

The default view is what you'll see when looking at the list online in whichever view you've set as *Default* in your settings. If you edit this view to add or remove either user created or system defined columns, or by creating a new view and setting it as default, these changes will propagate through the connector.

The screenshot shows the 'Sample List' settings page in SharePoint. It includes sections for List Information, General Settings, Permissions and Management, Communications, Columns, and Views. A detailed table of columns is also present.

Column (click to edit)	Type	Required
Title	Single line of text	
Modified	Date and Time	
Created	Date and Time	
Sample Column	Single line of text	
Created By	Person or Group	
Modified By	Person or Group	

Troubleshooting

Use root SharePoint address

Make sure you supply the root address of the SharePoint site, without any subfolders or documents. For example, use a link similar to `https://contoso.sharepoint.com/teams/ObjectModel/`.

Timezone issues

When using the SharePoint Online list (v1.0) connector, you may notice that timezone data doesn't match what you would expect from your browser. The SharePoint web-based client does a local timezone conversion based on the browser's knowledge of the user's timezone.

The backend API for SharePoint uses UTC time and sends this UTC time directly to Power BI. Power BI doesn't convert this UTC time, but reports it to the user.

To get time into local time, the user must do the same conversion the SharePoint client does. An example of the column operations that would do this are:

```
#"Changed Type" = Table.TransformColumnTypes(#"Renamed Columns",{{"Datewithtime", type datetimezone}}),
#"Timezone Shifted" = Table.TransformColumns(#"Changed Type", {"Datewithtime", DateTimeZone.ToLocal})
```

The first operation changes the type to `datetimezone`, and the second operation converts it to the computer's local time.

SharePoint join limit

This issue is limited to the SharePoint Online list v2.0 connector

The SharePoint Online list v2.0 connector uses a different API than the v1.0 connector and, as such, is subject to a maximum of 12 join operations per query, as documented in the [SharePoint Online documentation](#) under **List view lookup threshold**. This issue will manifest as SharePoint queries failing when more than 12 columns are accessed simultaneously from a SharePoint list.

Using OData to access a SharePoint Online list

If you use an OData feed to access a SharePoint Online list, there's an approximately 2100 character limitation to the URL you use to connect. More information: [Maximum URL length](#)

SIS-CC SDMX

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by the Statistical Information System Collaboration Community (SIS-CC), the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the SIS-CC website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Anonymous
Function Reference Documentation	—

Prerequisites

Before you get started, make sure you've properly configured the URL from the service provider's API. The exact process here will depend on the service provider.

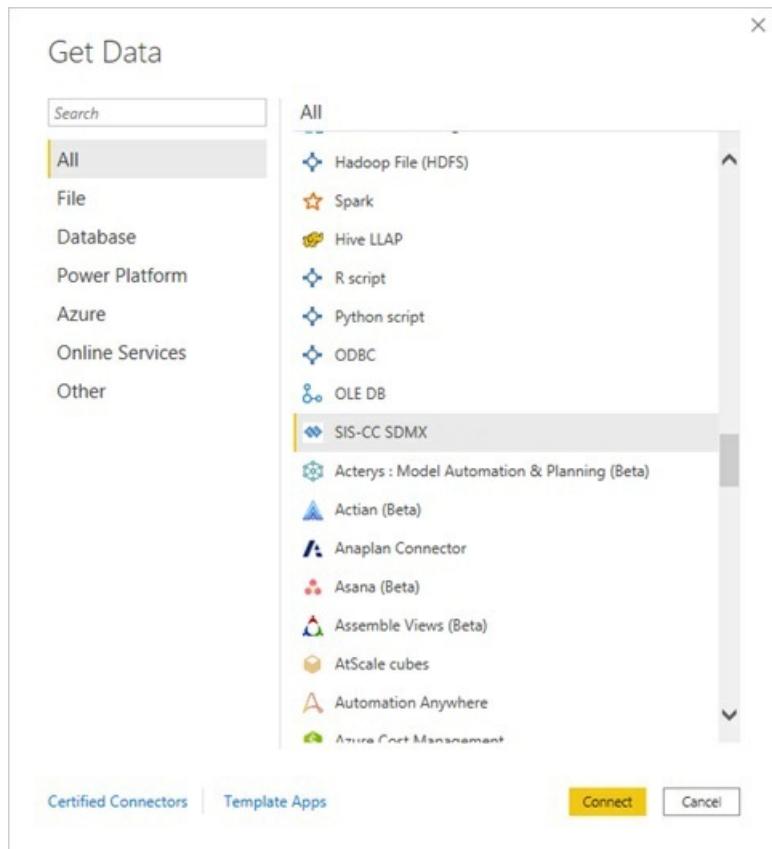
Capabilities supported

Import of SDMX-CSV 2.1 format. Other formats aren't supported.

Connection instructions

To connect to SDMX Web Service data:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **All** from the categories on the left, and then select **SIS-CC SDMX**. Then select **Connect**.

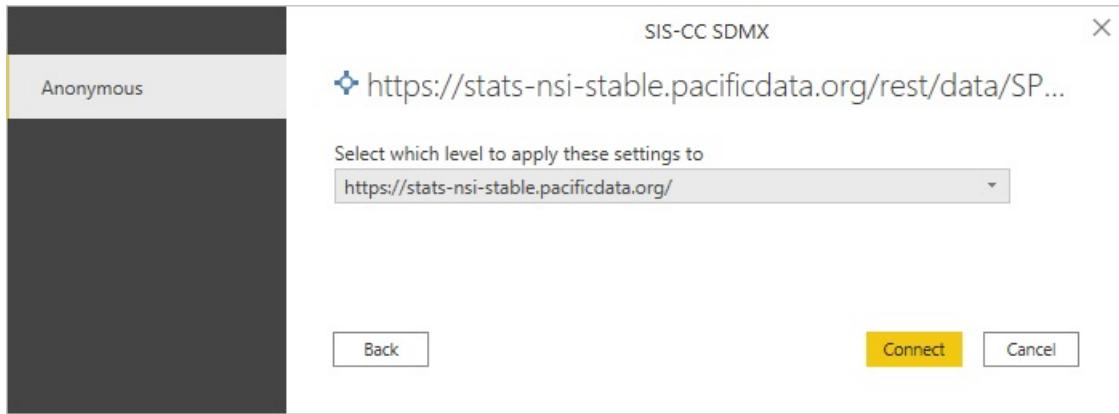


2. Fill in the parameters:

- In the **Data query URL**, enter an SDMX REST data query URL (the web service must support the SDMX-CSV format).
- In **Display format**, select one of the options:
 - Show codes and labels; example: FREQ: Frequency
 - Show codes; example: FREQ
 - Show labels; example: Frequency
 - Optionally, enter a language preference in **Label language preference** using an IETF BCP 47 tag



- If this is the first time you're connecting to the REST web service in the previous step **Data query URL**, this authentication step is displayed. As the connection is **Anonymous**, select **Connect**



4. Select **Load** to import the data into Power BI, or **Transform Data** to edit the query in Power Query Editor where you can refine the query before loading into Power BI.

Limitations and issues

This version of the connector doesn't support importing data formats SDMX-ML or SDMX-JSON.

Next steps

If you want to submit a feature request or contribute to the open-source project, then go to the [Gitlab project site](#).

Snowflake

1/15/2022 • 4 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows)
Authentication Types Supported	Database (Username/Password), AAD
Function Reference Documentation	-

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

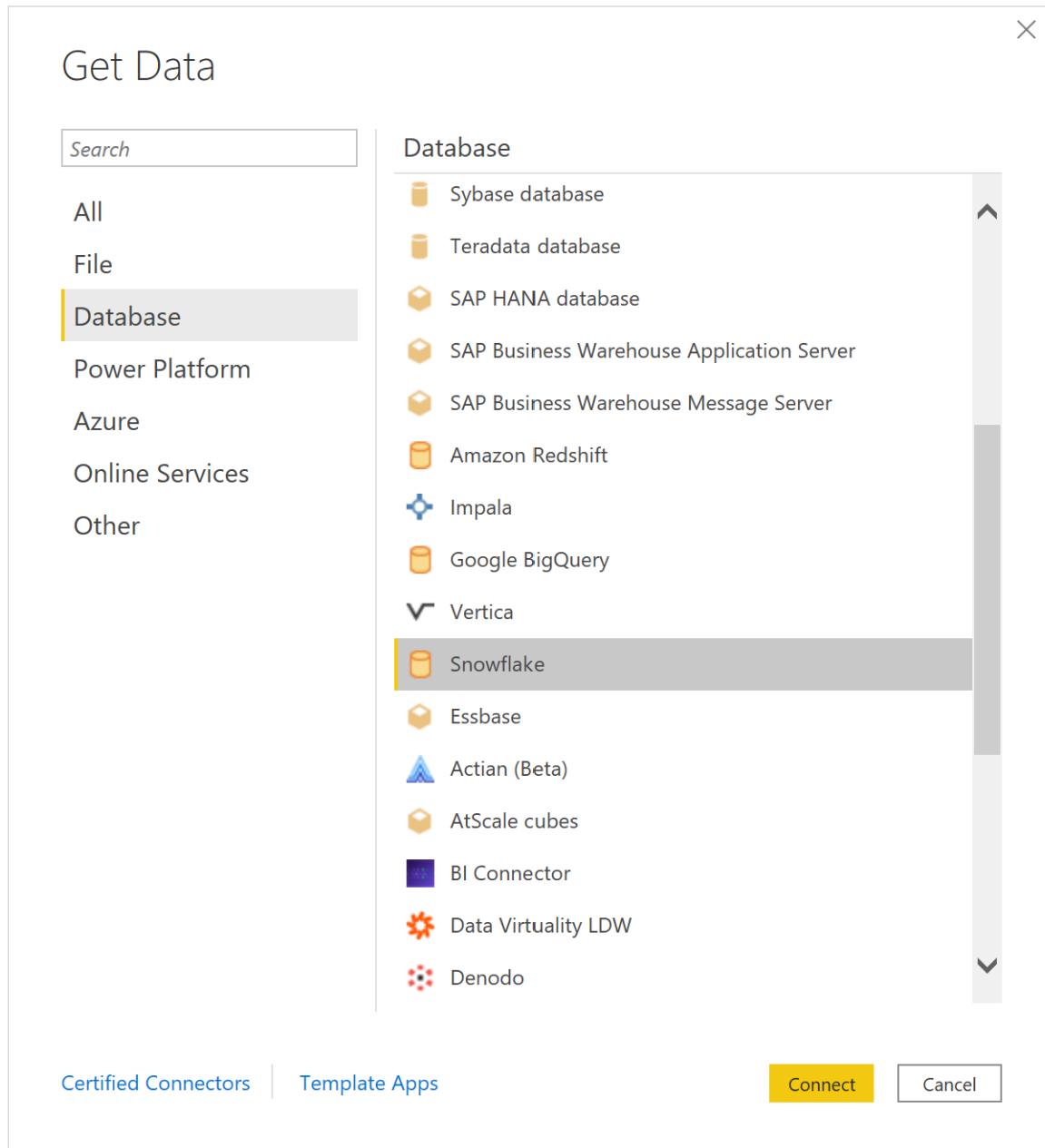
Capabilities Supported

- Import
- DirectQuery (Power BI only)
- Advanced options
 - Specify a text value to use as Role name
 - Native SQL statement
 - Relationship columns
 - Connection timeout in seconds
 - Command timeout in seconds

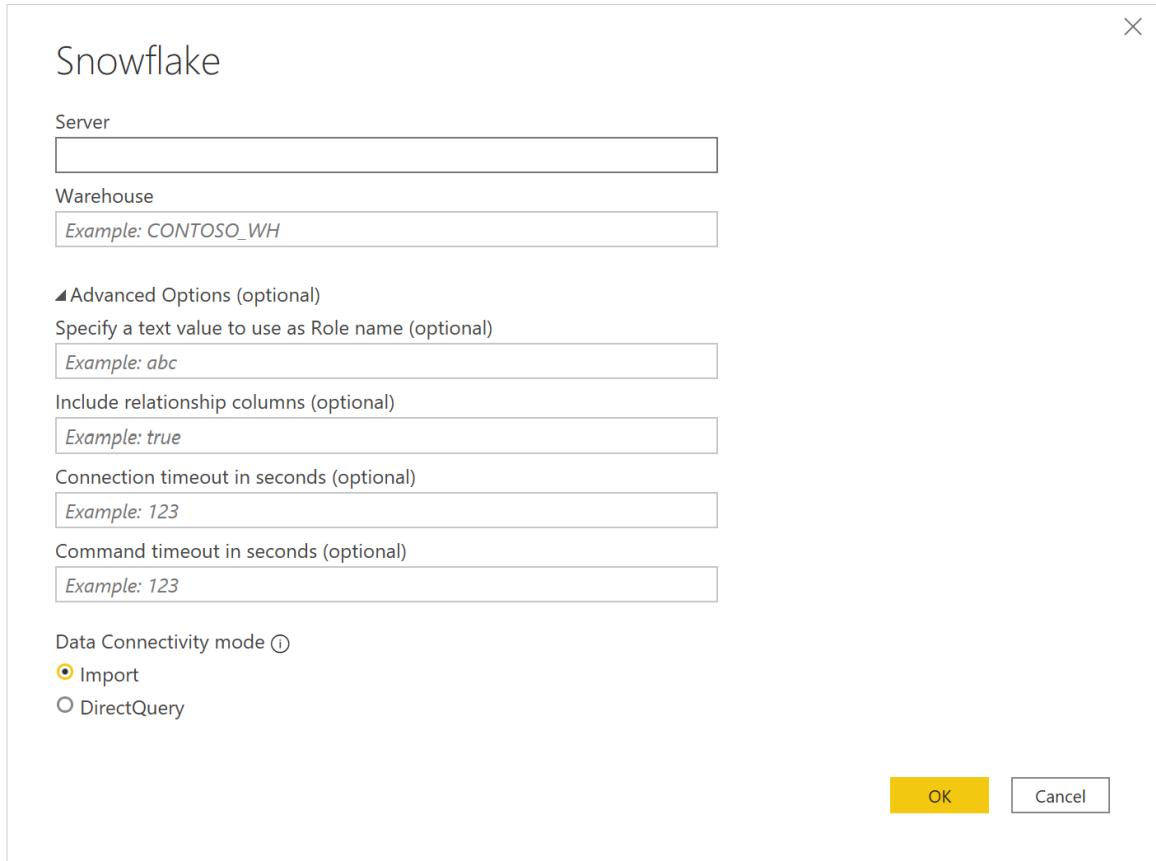
Connect to Snowflake data warehouse from Power Query Desktop

To make the connection to a **Snowflake** computing warehouse, take the following steps:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop, select **Database** from the categories on the left, select **Snowflake**, and then select **Connect**.



2. In the **Snowflake** window that appears, enter the name of your Snowflake server in **Server** and the name of your Snowflake computing warehouse in **Warehouse**.

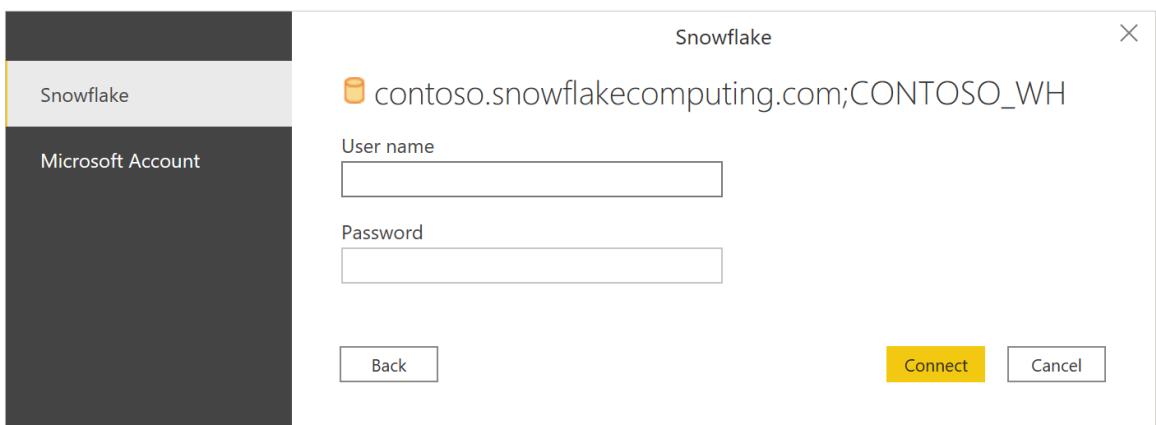


3. Optionally, enter values in any advanced options that you want to use to modify the connection query, such as a text value to use as a Role name or a command timeout. More information: [Connect using advanced options](#)
4. Select **Import** to import data directly into Power BI, or select **DirectQuery**. More information: [Use DirectQuery in Power BI Desktop](#)

NOTE

Azure Active Directory (Azure AD) Single Sign-On (SSO) only supports DirectQuery.

5. Select **OK**.
6. To sign in to your Snowflake computing warehouse, enter your username and password, and then select **Connect**.



NOTE

Once you enter your username and password for a particular **Snowflake** server, Power BI Desktop uses those same credentials in subsequent connection attempts. You can modify those credentials by going to **File > Options and settings > Data source settings**. More information: [Change the authentication method](#)

If you want to use the Microsoft account option, the Snowflake AAD integration must be configured on the Snowflake side. For more information: [Power BI SSO to Snowflake - Getting Started](#)

7. In **Navigator** select one or multiple elements to import and use in Power BI Desktop. Then select either **Load** to load the table in Power BI Desktop, or **Edit** to open the Power Query Editor where you can filter and refine the set of data you want to use, and then load that refined set of data into Power BI Desktop.

Connect to a Snowflake database from Power Query Online

To make the connection, take the following steps:

1. Select the **Snowflake** option in the connector selection.
2. In the **Snowflake** dialog that appears, enter the name of the server and warehouse.

The screenshot shows the 'Connection settings' dialog for connecting to a Snowflake database. On the left, there's a 'Snowflake Database' icon. The main area has two sections: 'Connection settings' and 'Connection credentials'. Under 'Connection settings', there are fields for 'Server' (with placeholder 'Example: contoso.snowflakecomputing.com') and 'Warehouse' (with placeholder 'Example: CONTOSO_WH'). Below these are 'Advanced options' (which are collapsed), 'Specify a text value to use as Role name' (empty field), and 'Include relationship columns' (checkbox checked). There are also fields for 'Connection timeout in seconds' and 'Command timeout in seconds', both set to '1.2'. Under 'Connection credentials', there's a dropdown for 'On-premises data gateway' (set to '(none)') with a refresh icon, an 'Authentication kind' dropdown (set to 'Basic'), and fields for 'Username' and 'Password'.

3. Enter any values in the advanced options you want to use. If there are any advanced options not represented in the UI, you can edit them in the **Advanced Editor** in Power Query later.
4. Enter your connection credentials, including selecting or creating a new connection, which gateway you

would like to use, and a username and password (only the Basic authentication kind is supported in Power Query Online).

5. Select **Next** to connect to the database.
6. In **Navigator**, select the data you require, then select **Transform data** to transform the data in Power Query Editor.

Connect using advanced options

Power Query provides a set of advanced options that you can add to your query if needed.

The following table lists all of the advanced options you can set in Power Query.

ADVANCED OPTION	DESCRIPTION
Role name	Specifies the role that the report uses via the driver. This role must be available to the user, otherwise no role will be set.
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Connection timeout in seconds	Specifies how long to wait for a response when interacting with the Snowflake service before returning an error. Default is 0 (no timeout).
Command timeout in seconds	Specifies how long to wait for a query to complete before returning an error. Default is 0 (no timeout).
SQL Statement	For information, go to Import data from a database using native database query . In this version of native database query functionality, you need to use fully qualified table names in the format Database.Schema.Table , for example SELECT * FROM DEMO_DB.PUBLIC.DEMO_TABLE .

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your Snowflake database.

Power BI and Snowflake Webinar

Power BI and Snowflake partnered to host a hands-on webinar on August 25th, 2021 to deep-dive into [Attaining Consumer Insights with Snowflake and Microsoft Power BI](#).

In this hands on lab, learn how to access all relevant data from a single source and turn data into insights through this connector.

- Access first party data seamlessly in Snowflake
- Leverage the Data Marketplace to query live data from 3rd party providers
- Set up self-serve analytics in Power BI

View the webinar on-demand using the above link!

Additional information

- [Connect to Snowflake in Power BI Service](#)

SoftOne BI (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by SoftOne, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the SoftOne website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Preview
Products	Power BI (Datasets)
Authentication Types Supported	Basic (Soft1/Atlantis Web Services)

Prerequisites

You'll need to have the Soft1 ERP/CRM or Atlantis ERP product installed with a licensed SoftOne BI connector module. A web account must be configured in the application with access to the SoftOne BI Connector service. This account information and your installation serial number will be validated during authentication by the SoftOne BI connector.

The SoftOne BI connector is supported from Soft1 Series 5 version 500.521.11424 or later and Atlantis ERP version 3.3.2697.1 or later.

Capabilities supported

- Import

Connection instructions

SoftOne provides many templates as Power BI template files (.pbit) that you can use or customize which will provide you with a start to your BI project. For example, Sales & Collections, and Finance.

To connect in Power BI Desktop using a new report, follow the steps below. If you're connecting from a report created using one of the SoftOne BI templates, see [Using a provided template](#) later in this article.

Connect to your Soft1 or Atlantis data store from scratch

To load data from your installation with Power Query Desktop:

1. Select **Get Data > More... > Online Services** in Power BI Desktop and search for **SoftOne BI**. Select **Connect**.

Get Data

- All
- File
- Database
- Power Platform
- Azure
 - Online Services
 - Other

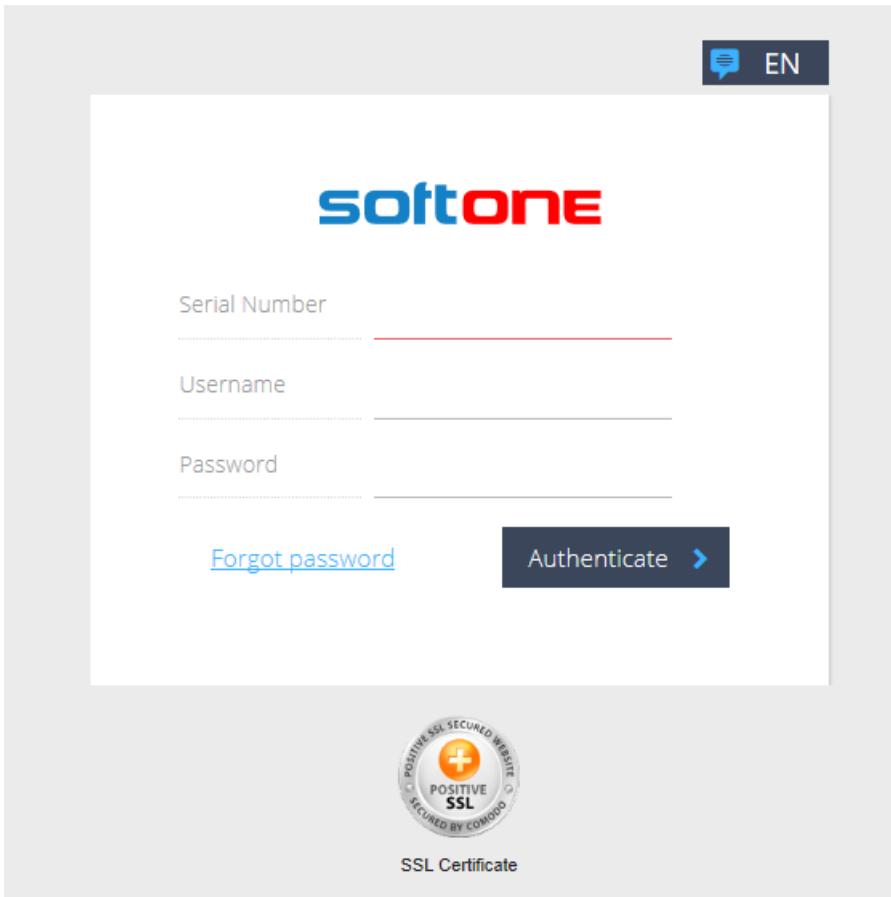
Online Services

- SparkPost (Beta)
- SweetIQ (Beta)
- Planview Enterprise One - CTM (Beta)
- Twilio (Beta)
- Zendesk (Beta)
- SoftOne BI (Beta)**
- Asana (Beta)
- Dynamics 365 Customer Insights (Beta)
- Emigo Data Source
- Entersoft Business Suite (Beta)
- eWay-CRM (Beta)
- FactSet Analytics
- Palantir Foundry
- Hexagon PPM Smart® API
- Industrial App Store
- Intune Data Warehouse (Beta)

Certified Connectors | Template Apps

Connect **Cancel**

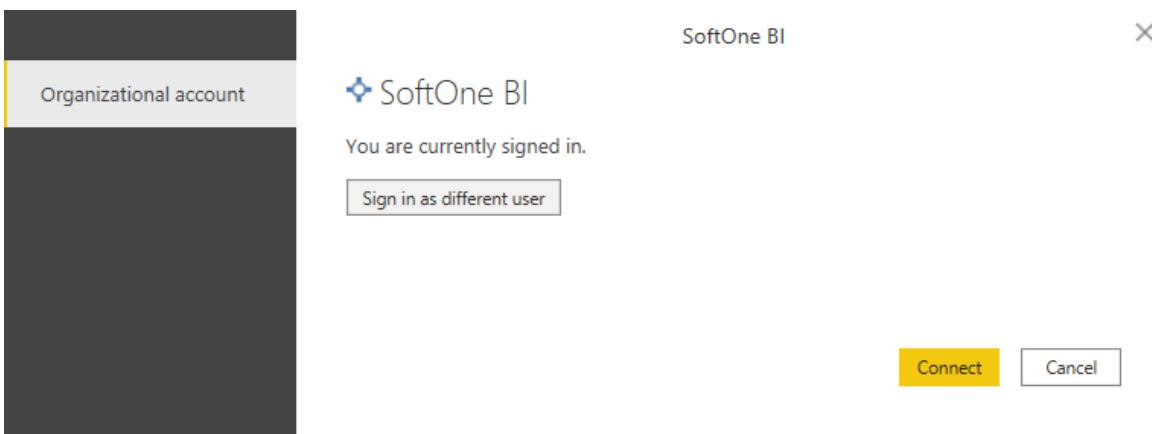
2. Select **Sign in**. An authentication form will display.



NOTE

- If you enter incorrect credentials, you'll receive a message stating that your sign in failed due to invalid credentials.
- If the SoftOne BI Connector is not activated, or the Web Account that you're using is not configured with the service, you'll receive a message stating that access is denied because the selected module is not activated.

3. After signing in with SoftOne Web Services, you can connect to your data store.



Selecting **Connect** will take you to the navigation table and display the available tables from the data store from which you may select the data required.

4. In the navigator, you should now see the tables in your data store. Fetching the tables can take some time.

Navigator

The screenshot shows the Power BI Desktop interface with the 'Navigator' tab selected. On the left, there's a navigation pane titled 'Display Options' with a search bar and a refresh icon. Below it is a tree view of data sources. A folder icon next to 'SoftOne BI [42]' indicates it contains 42 items. Underneath are various table names, each with a small icon and a dropdown arrow. To the right of the tree view is a large empty area with the text 'No items selected for preview'. At the bottom right are three buttons: 'Load', 'Transform Data', and 'Cancel'.

No items selected for preview

Load Transform Data Cancel

You must have uploaded the data from your Soft1 or Atlantis installation (per the product documentation) to see any tables. If you haven't uploaded your data, you won't see any tables displayed in the Navigation Table.

Navigator

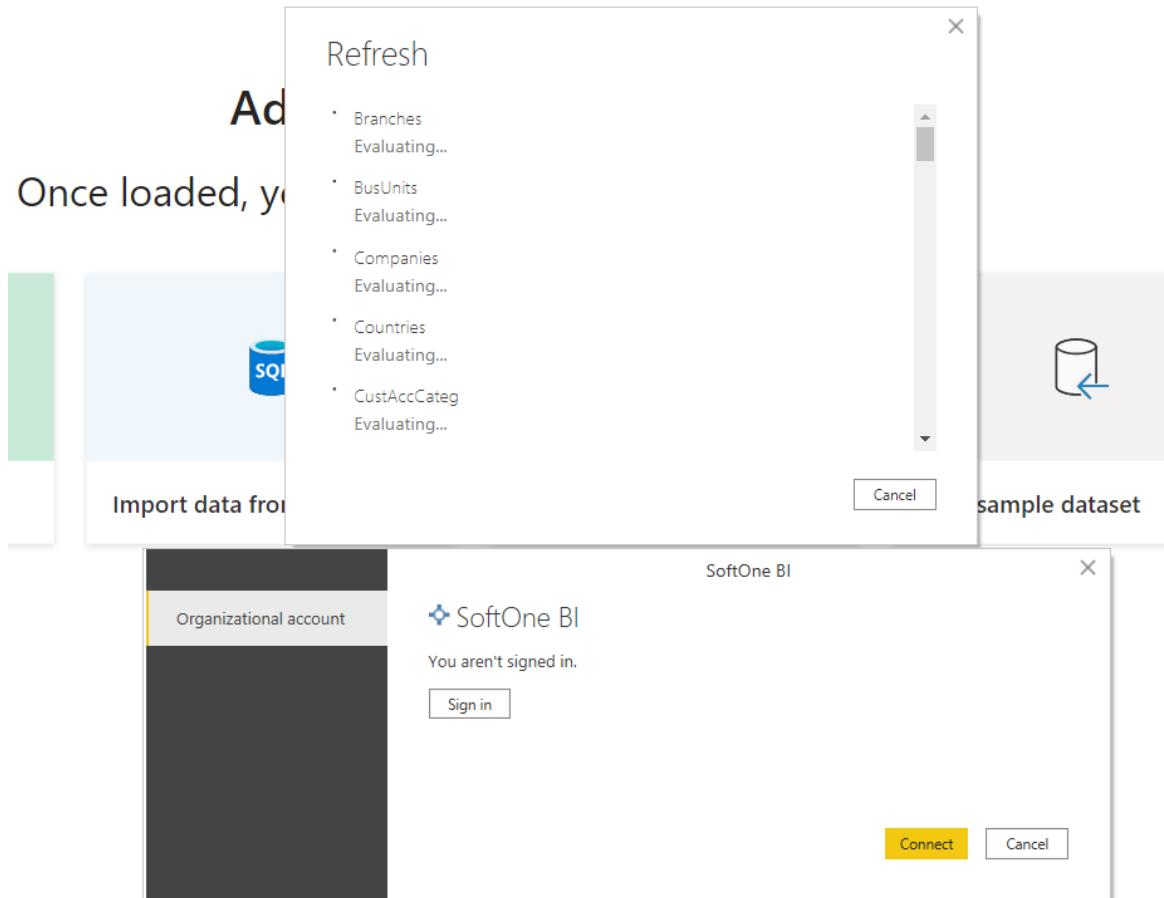
This screenshot shows the same Power BI Desktop interface as above, but with a different tree view. Only one item, 'SoftOne BI', is visible under the 'Display Options' section. The rest of the interface is identical, with the main area showing 'No items selected for preview' and the standard 'Load', 'Transform Data', and 'Cancel' buttons at the bottom right.

No items selected for preview

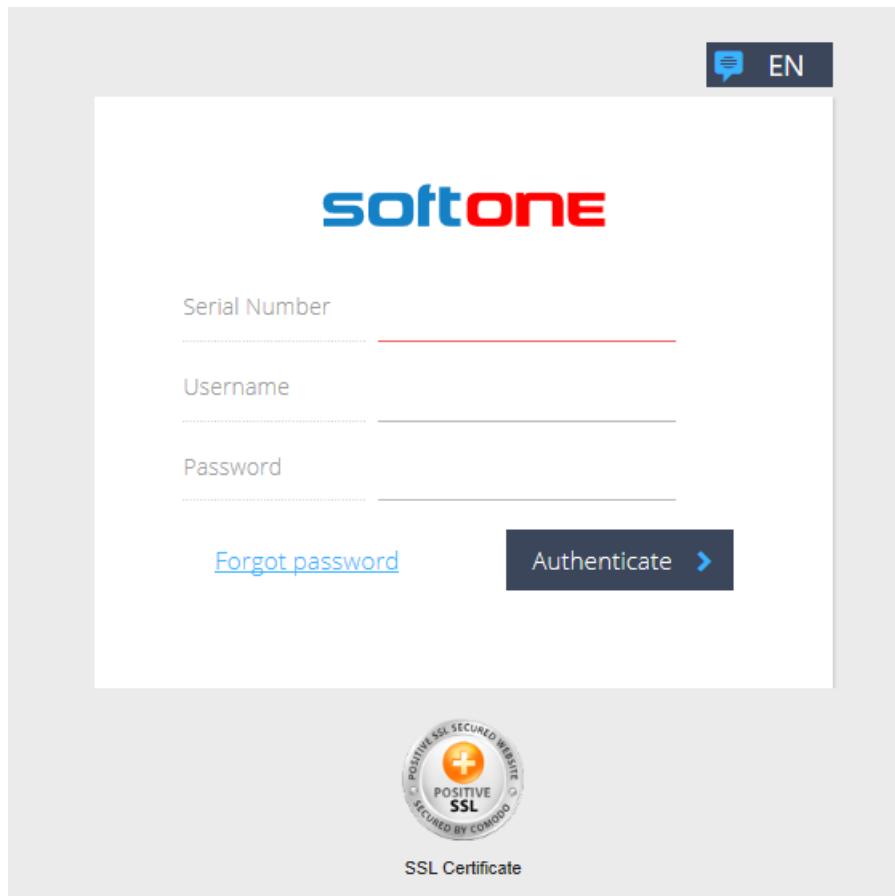
In this case, you'll need to go back to your application and upload your data.

Using a provided template

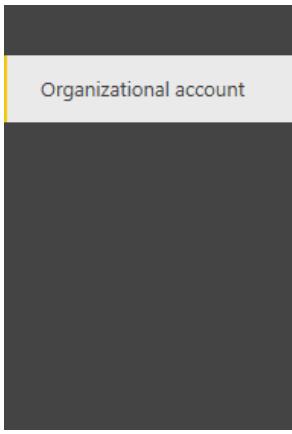
1. Open the selected template, Power BI Desktop will attempt to load the data from the data store, and will prompt for credentials.



2. Select **Sign in** and enter your credentials (Serial number, username, and password).



3. Once you're authenticated, select **Connect**.



SoftOne BI

X

Organizational account

SoftOne BI

You are currently signed in.

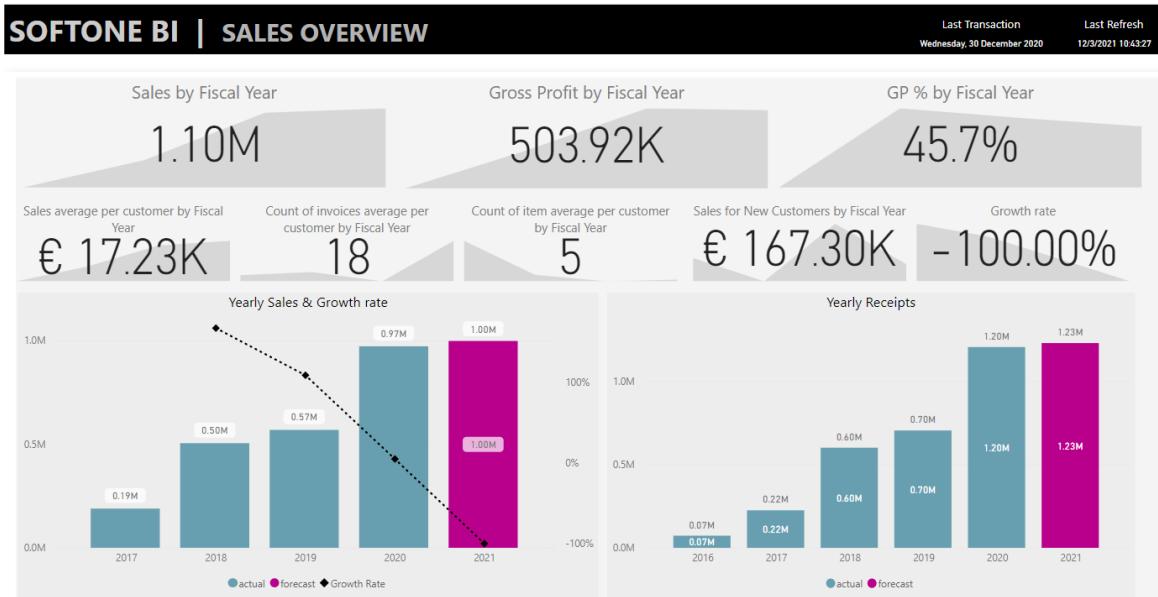
Sign in as different user

Connect

Cancel

Power BI Desktop will fetch the data from the data store.

- After the refresh has completed, you're ready to start customizing the report or to publish it as is to the Power BI Service.



IMPORTANT

If you're working with more than one Soft1/Atlantis installation, then when switching between data stores, you must clear the SoftOne BI credentials saved by Power BI Desktop.

SQL Server

1/15/2022 • 3 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Authentication Types Supported	Database (Username/Password) Windows
M Function Reference	Sql.Database Sql.Databases

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

By default, Power BI installs an OLE DB driver for SQL Server. However, for optimal performance, we recommend that the customer installs the [SQL Server Native Client](#) before using the SQL Server connector. SQL Server Native Client 11.0 and SQL Server Native Client 10.0 are both supported in the latest version.

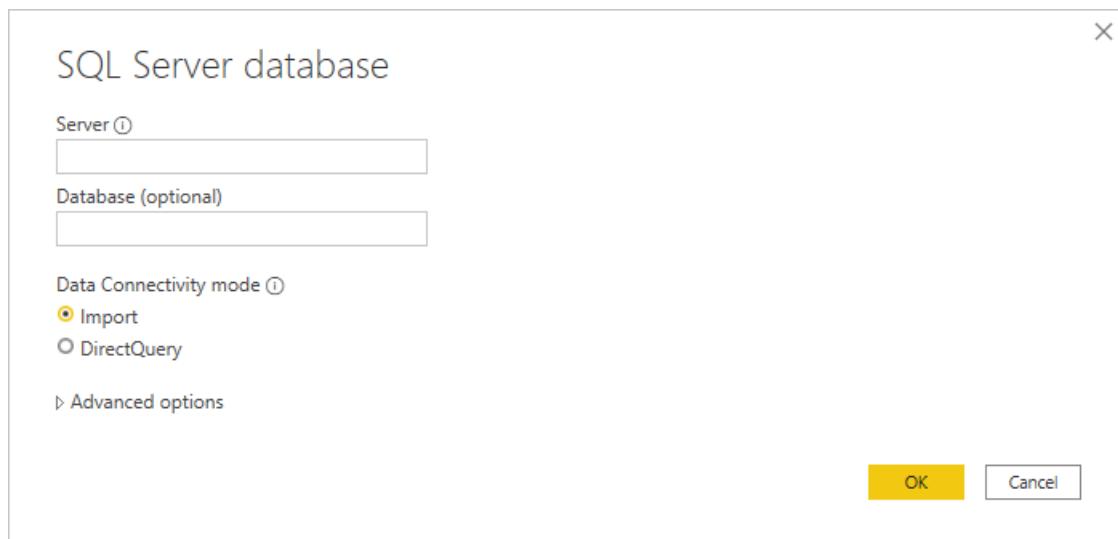
Capabilities Supported

- Import
- DirectQuery (Power BI Desktop)
- Advanced options
 - Command timeout in minutes
 - Native SQL statement
 - Relationship columns
 - Navigate using full hierarchy
 - SQL Server failover support

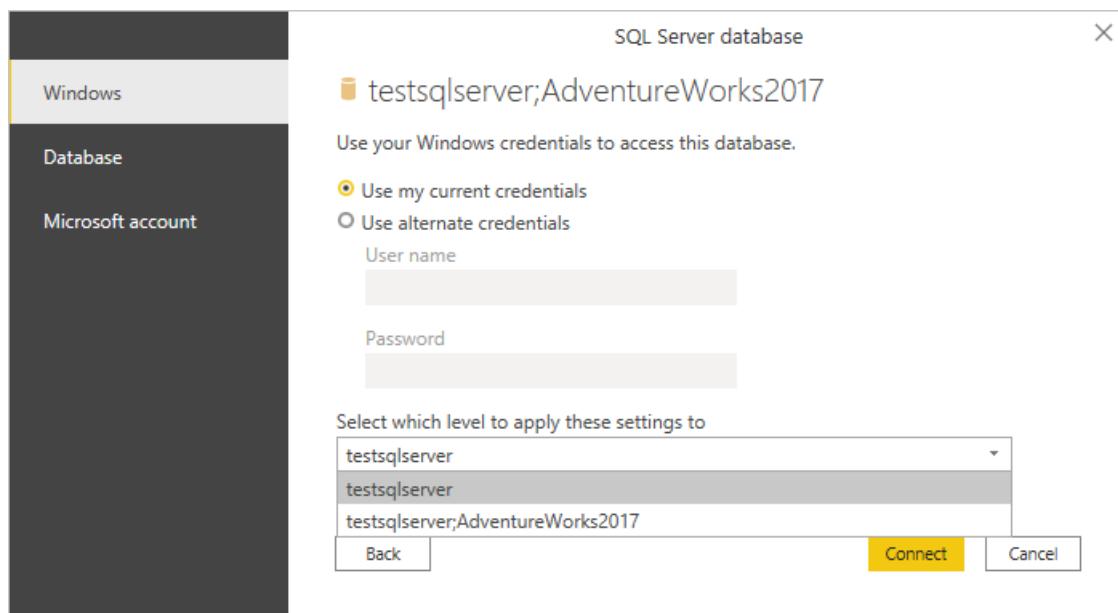
Connect to SQL Server database from Power Query Desktop

To make the connection, take the following steps:

1. Select the **SQL Server database** option in the connector selection.
2. In the **SQL Server database** dialog that appears, provide the name of the server and database (optional).

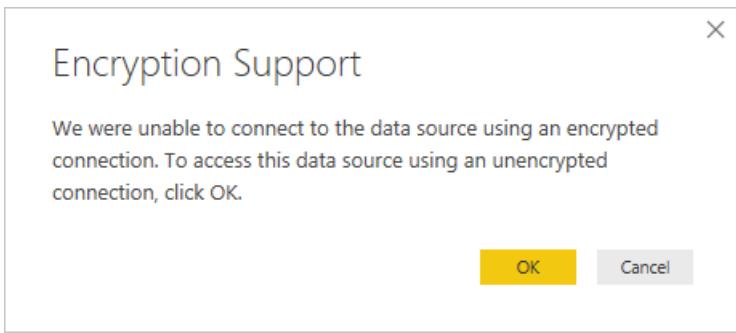


3. Select either the **Import** or **DirectQuery** data connectivity mode (Power BI Desktop only).
4. Select **OK**.
5. If this is the first time you're connecting to this database, select the authentication type, input your credentials, and select the level to apply the authentication settings to. Then select **Connect**.



NOTE

If the connection is not encrypted, you'll be prompted with the following dialog.



Select **OK** to connect to the database by using an unencrypted connection, or follow these [instructions](#) to setup encrypted connections to SQL Server.

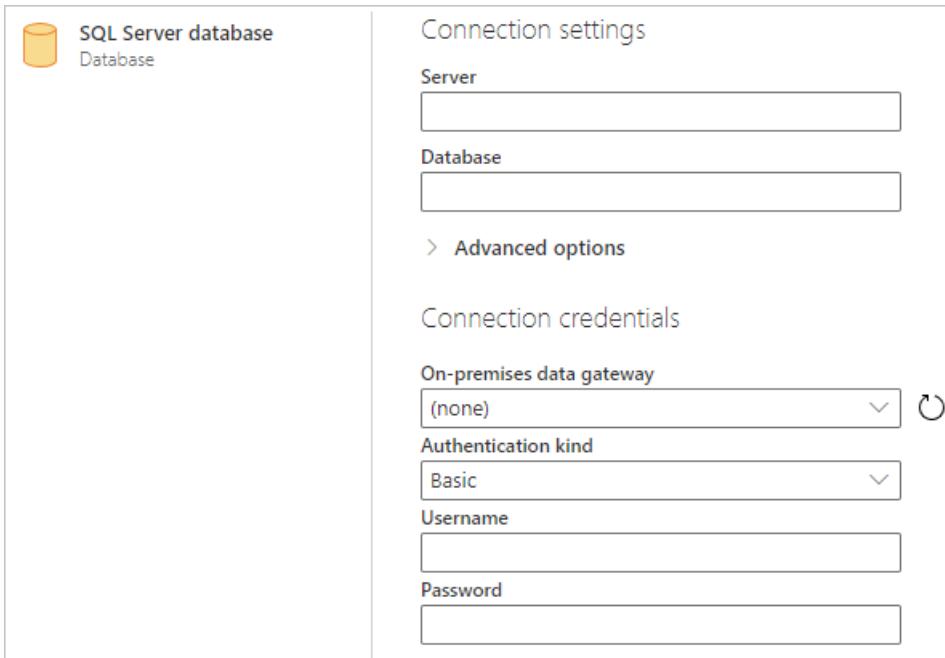
6. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

A screenshot of the Power Query Navigator interface. On the left, there's a tree view of available tables: Sales.vStoreWithContacts, Sales.vStoreWithDemographics, AWBuildVersion, DatabaseLog, ErrorLog, HumanResources.Department, HumanResources.Employee (which is selected, indicated by a checked checkbox), HumanResources.EmployeeDepartment, HumanResources.EmployeePayHistory, HumanResources.JobCandidate, HumanResources.Shift, Person.Address, and Person.AddressType. On the right, a preview of the "HumanResources.Employee" table is shown with 16 rows of data. The columns are: BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, and JobTitle. The "JobTitle" column shows values like "null Chi", "1 Vice", "2 Eng", "3 Sen", "3 Des", "3 Des", "3 Des", "3 Res", "4 Res", "4 Res", "4 Res", "3 Sen", "4 Toc", "4 Toc", "3 Sen", "3 Des", and "4 Man". At the bottom of the interface are buttons for "Select Related Tables", "Load" (highlighted in yellow), "Transform Data", and "Cancel".

Connect to SQL Server database from Power Query Online

To make the connection, take the following steps:

1. Select the **SQL Server database** option in the connector selection.
2. In the **SQL Server database** dialog that appears, provide the name of the server and database (optional).



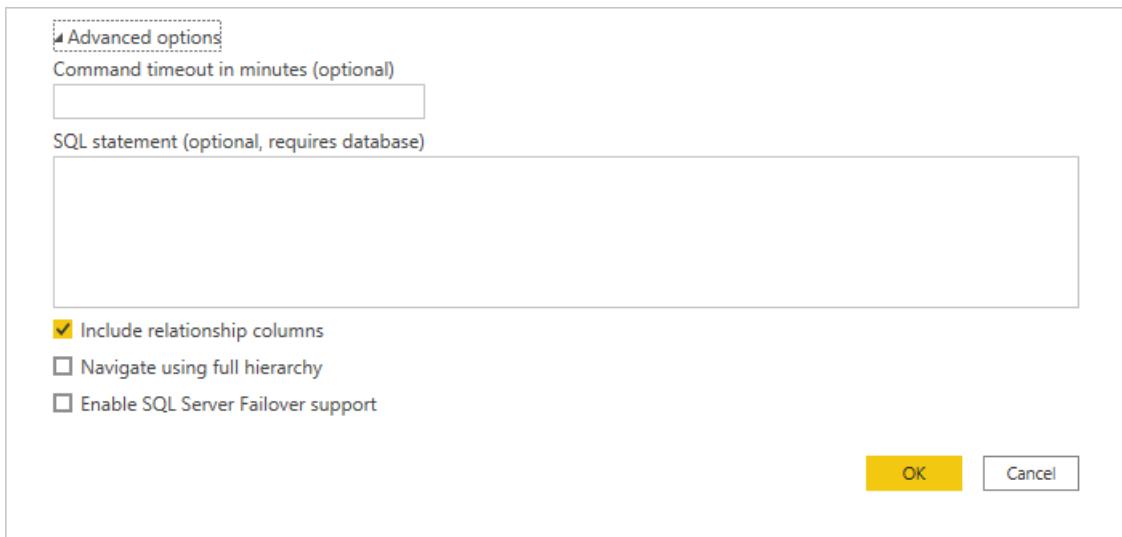
3. If needed, select an on-premises data gateway.
4. If this is the first time you're connecting to this database, select the authentication kind and input your credentials.
5. If the connection is not encrypted, and the connection dialog contains a **Use Encrypted Connection** check box, clear the check box.
6. Select **Next** to continue.

7. In **Navigator**, select the data you require, and then select **Transform data**.

BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	Job
1	295847284	adventure-works\ken0		null	null Chie
2	245797967	adventure-works\terri0	/1/	1 Vice	
3	509647174	adventure-works\roberto0	/1/1/	2 Engi	
4	112457891	adventure-works\rob0	/1/1/1/	3 Seni	
5	695256908	adventure-works\gail0	/1/1/2/	3 Desi	
6	998320692	adventure-works\jossef0	/1/1/3/	3 Desi	
7	134969118	adventure-works\dylan0	/1/1/4/	3 Rese	
8	811994146	adventure-works\diane1	/1/1/4/1/	4 Rese	
9	658797903	adventure-works\gigi0	/1/1/4/2/	4 Rese	
10	879342154	adventure-works\michael6	/1/1/4/3/	4 Rese	
11	974026903	adventure-works\ovidiu0	/1/1/5/	3 Seni	
12	480168528	adventure-works\thierry0	/1/1/5/1/	4 Tool	
13	486228782	adventure-works\janice0	/1/1/5/2/	4 Tool	
14	42487730	adventure-works\michael8	/1/1/6/	3 Seni	
15	56920285	adventure-works\sharon0	/1/1/7/	3 Desi	

Connect using advanced options

Both Power Query Desktop and Power Query Online provide a set of advanced options that you can add to your query if needed.



The following table lists all of the advanced options you can set in Power Query Desktop and Power Query Online.

ADVANCED OPTION	DESCRIPTION
Command timeout in minutes	If your connection lasts longer than 10 minutes (the default timeout), you can enter another value in minutes to keep the connection open longer. This option is only available in Power Query Desktop.
SQL statement	For information, go to Import data from a database using native database query .
Include relationship columns	If checked, includes columns that might have relationships to other tables. If this box is cleared, you won't see those columns.
Navigate using full hierarchy	If checked, the Navigator displays the complete hierarchy of tables in the database you're connecting to. If cleared, Navigator displays only the tables whose columns and rows contain data.
Enable SQL Server Failover support	If checked, when a node in the SQL Server failover group isn't available, Power Query moves from that node to another when failover occurs. If cleared, no failover will occur.

Once you've selected the advanced options you require, select **OK** in Power Query Desktop or **Next** in Power Query Online to connect to your SQL Server database.

Troubleshooting

Always Encrypted columns

Power Query doesn't support 'Always Encrypted' columns.

Next steps

[Optimize Power Query when expanding table columns](#)

Stripe (Deprecated)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Deprecated
Products	-
Authentication Types Supported	-
Function Reference Documentation	-

Deprecation

This connector is deprecated, and won't be supported soon. We recommend you transition off existing connections using this connector, and don't use this connector for new connections.

SumTotal (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by SumTotal, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the SumTotal website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI (Datasets)
Authentication types	OAuth 2.0
Function Reference Documentation	-

Prerequisites

You must have a SumTotal hosted environment with standard permissions to access the portal, and read permissions to access data in tables.

Capabilities supported

- Import

Finding your SumTotal hosted URL

Copy the SumTotal hosted root URL in full. This root URL is the unique URL specific to your instance. The URL will be in the format of <https://<yourdomain>.sumtotal.host/>. Make sure not to copy the rest of the URL. Keep this URL somewhere handy so you can use it later.

Connect to SumTotal BI from Power BI Desktop

NOTE

The Power Query SumTotal connector is currently only suited towards OData API endpoints. For more information, see the [SumTotal OData API specification](#).

To connect to SumTotal from Power BI Desktop:

1. In the **Get Data** experience, select **SumTotal** in the **Other** category, and then select **Connect**.

2. Enter the server URL address of the data you want to load.

NOTE

You'll be prompted with a script error; this is expected and loads the JS/CSS scripts that the login form uses. Select Yes.

3. When the table is loaded in **Navigator**, you'll be presented with the list of OData API entities that are currently supported by the connector. You can select to load one or multiple entities.
4. When you've finished selecting entities, select **Load** to load the data directly in Power BI desktop, or select **Transform Data** to transform the data.

NOTE

If this is the first time you're connecting to this site, select **Sign in** and input your credentials. Then select **Connect**.

Known issues and limitations

This section describes any limitations or considerations of the SumTotal connector.

SumTotal OData API performance and throttling limits

For information about OData API performance and throttling limits for SumTotal connections, see the [SumTotal's OData API specification](#) under the **Important Notes** section. These limitations apply to both the SumTotal connector (which uses the OData API) and the 'actual' OData Hosted API when accessing the same endpoints.

Table retrieval rate

Most default tables are retrieved at approximately 1000 rows per second using the SumTotal connector. If you require faster retrieval rates, consider using the **RowVersionId** filter parameter. You can pass this parameter directly to the environment hosted URL by appending it as a query string parameter, for example, <https://{{host}}.sumtotalsystems.com/?rowVersionId=1234>.

Text/CSV

1/15/2022 • 6 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Function Reference Documentation	File.Contents Lines.FromBinary Csv.Document

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

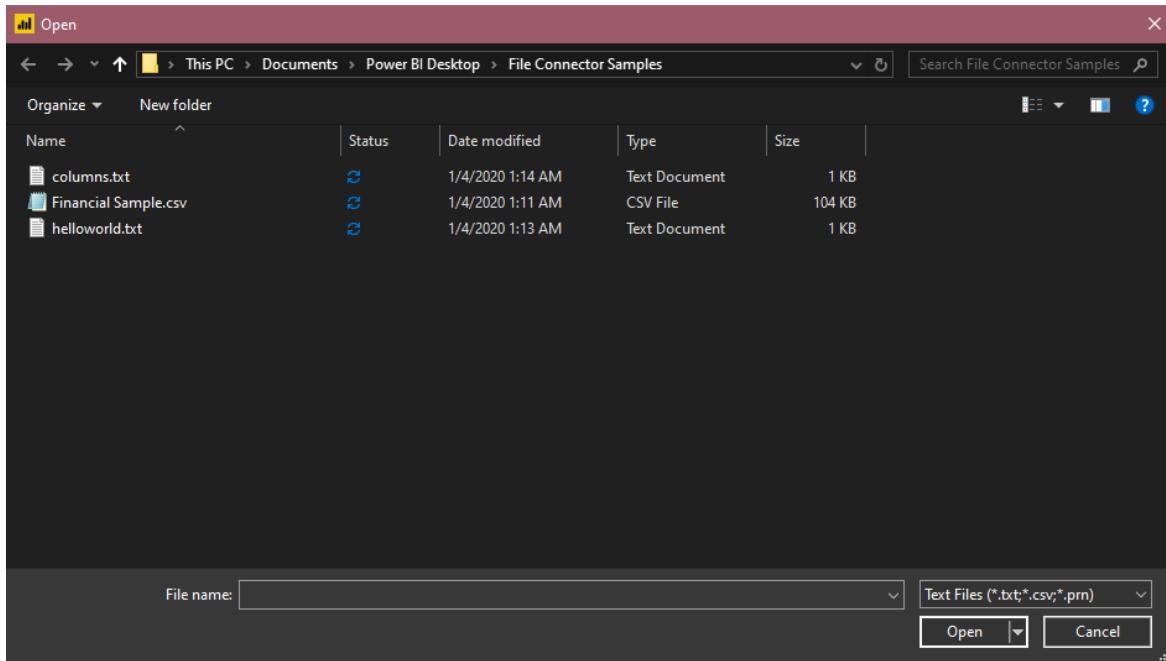
Capabilities supported

- Import

Connect to local text/CSV file from Power Query Desktop

To load a local text or CSV file:

1. Select the **Text/CSV** option in **Get Data**. This action launches a local file browser where you can select your text file.



Select Open to open the file.

- From the **Navigator**, you can either transform the data in the Power Query Editor by selecting **Transform Data**, or load the data by selecting **Load**.

Segment	Country	Product	Discount Band	Units Sold	Manufacturing Price	Sale Price	Gross Sales	Discounts	Sales	COGS
Government	United States of America	Paseo	Low	3450	10.00	350.00	1,207,500.00	48,300.00	1,159,200.00	897,000.00
Government	France	Amarilla	None	2750	260.00	350.00	962,500.00	0.00	962,500.00	715,000.00
Government	Germany	Velo	Low	2966	120.00	350.00	1,038,100.00	20,762.00	1,017,338.00	771,160.00
Government	Germany	Amarilla	Low	2966	260.00	350.00	1,038,100.00	20,762.00	1,017,338.00	771,160.00
Government	Germany	Velo	Low	2877	120.00	350.00	1,006,950.00	20,139.00	986,811.00	748,020.00
Government	Germany	VTT	Low	2877	250.00	350.00	1,006,950.00	20,139.00	986,811.00	748,020.00
Government	Canada	Carretera	Low	2852	3.00	350.00	998,200.00	19,964.00	978,236.00	741,520.00
Government	Canada	Paseo	Low	2852	10.00	350.00	998,200.00	19,964.00	978,236.00	741,520.00
Government	France	Amarilla	Medium	2876	260.00	350.00	1,006,600.00	70,462.00	936,138.00	747,760.00
Government	France	Carretera	Low	2155	3.00	350.00	754,250.00	7,542.50	746,707.50	560,300.00
Government	France	Paseo	Low	2155	10.00	350.00	754,250.00	7,542.50	746,707.50	560,300.00
Government	France	Velo	Low	2177	120.00	350.00	761,950.00	30,478.00	731,472.00	566,020.00

Connect to text/CSV file from Power Query Online

To load a local text or CSV file:

- From the **Data sources** page, select **Text/CSV**.
- In **Connection settings**, enter a file path to the local text or CSV file you want.

3. Select an on-premises data gateway from **Data gateway**.
4. Enter a username and password.
5. Select **Next**.
6. From the **Navigator**, select **Transform Data** to begin transforming the data in the Power Query Editor.

Segment	Country	Product	Discount Band	1.2 Units Sold	\$ Manufacturing Price	\$ Sale Price	\$ Gross Sales	\$ Discounts	\$ Sales	\$ COGS	\$ Profit
Government	United States of Amer...	Paseo	Low	3450	10	350	1,207,500	48,300	1,159,200	897,000	26
Government	France	Amarilla	None	2750	260	350	962,500	0	962,500	75,000	24
Government	Germany	Velo	Low	2966	120	350	1,038,100	20,762	1,017,338	771,160	24
Government	Germany	Amarilla	Low	2966	260	350	1,038,100	20,762	1,017,338	771,160	24
Government	Germany	Velo	Low	2877	120	350	1,006,950	20,139	986,811	748,020	23
Government	Germany	VTT	Low	2877	250	350	1,006,950	20,139	986,811	748,020	23
Government	Canada	Carretera	Low	2852	3	350	998,200	19,964	978,236	741,520	23
Government	Canada	Paseo	Low	2852	10	350	998,200	19,964	978,236	741,520	23
Government	France	Amarilla	Medium	2876	260	350	1,006,600	70,462	936,138	747,760	18
Government	France	Carretera	Low	2155	3	350	754,250	7,542.5	746,707.5	560,300	186
Government	France	Paseo	Low	2155	10	350	754,250	7,542.5	746,707.5	560,300	186
Government	France	Velo	Low	2177	120	350	761,950	30,478	731,472	566,020	16
Government	France	VTT	Low	2177	250	350	761,950	30,478	731,472	566,020	16
Government	Mexico	VTT	Low	1940	250	350	679,000	13,580	665,420	504,400	16
Government	Canada	Paseo	None	1725	10	350	603,750	0	603,750	448,500	15

Load from the web

To load a text or CSV file from the web, select the [Web connector](#), enter the web address of the file, and follow any credential prompts.

Text/CSV delimiters

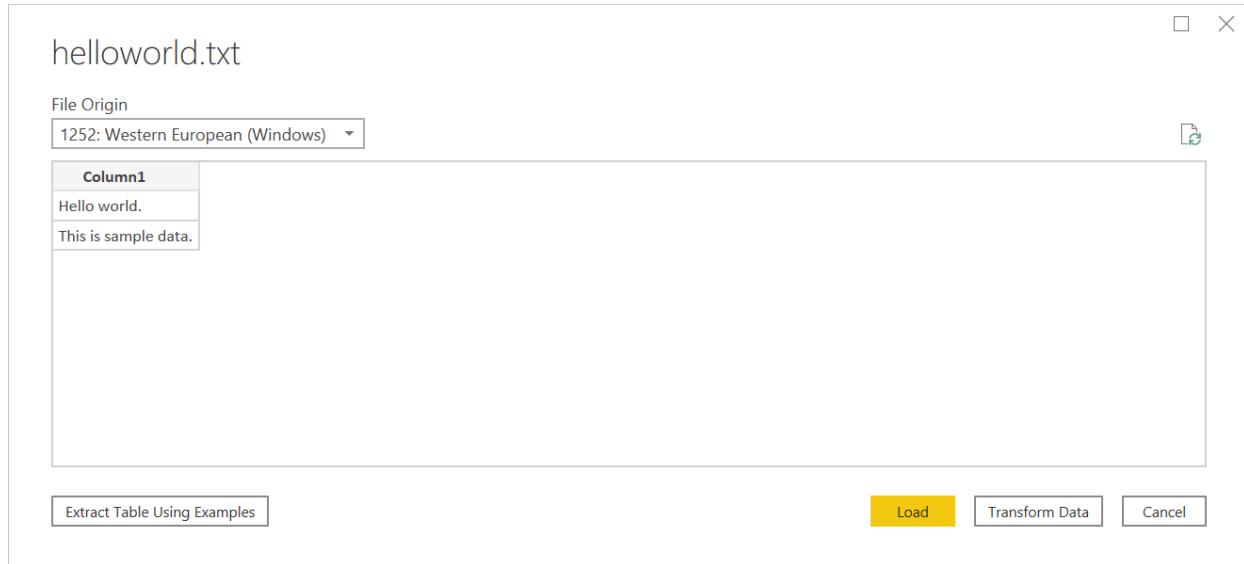
Power Query will treat CSVs as structured files with a comma as a delimiter—a special case of a text file. If you choose a text file, Power Query will automatically attempt to determine if it has delimiter separated values, and what that delimiter is. If it can infer a delimiter, it will automatically treat it as a structured data source.

Unstructured Text

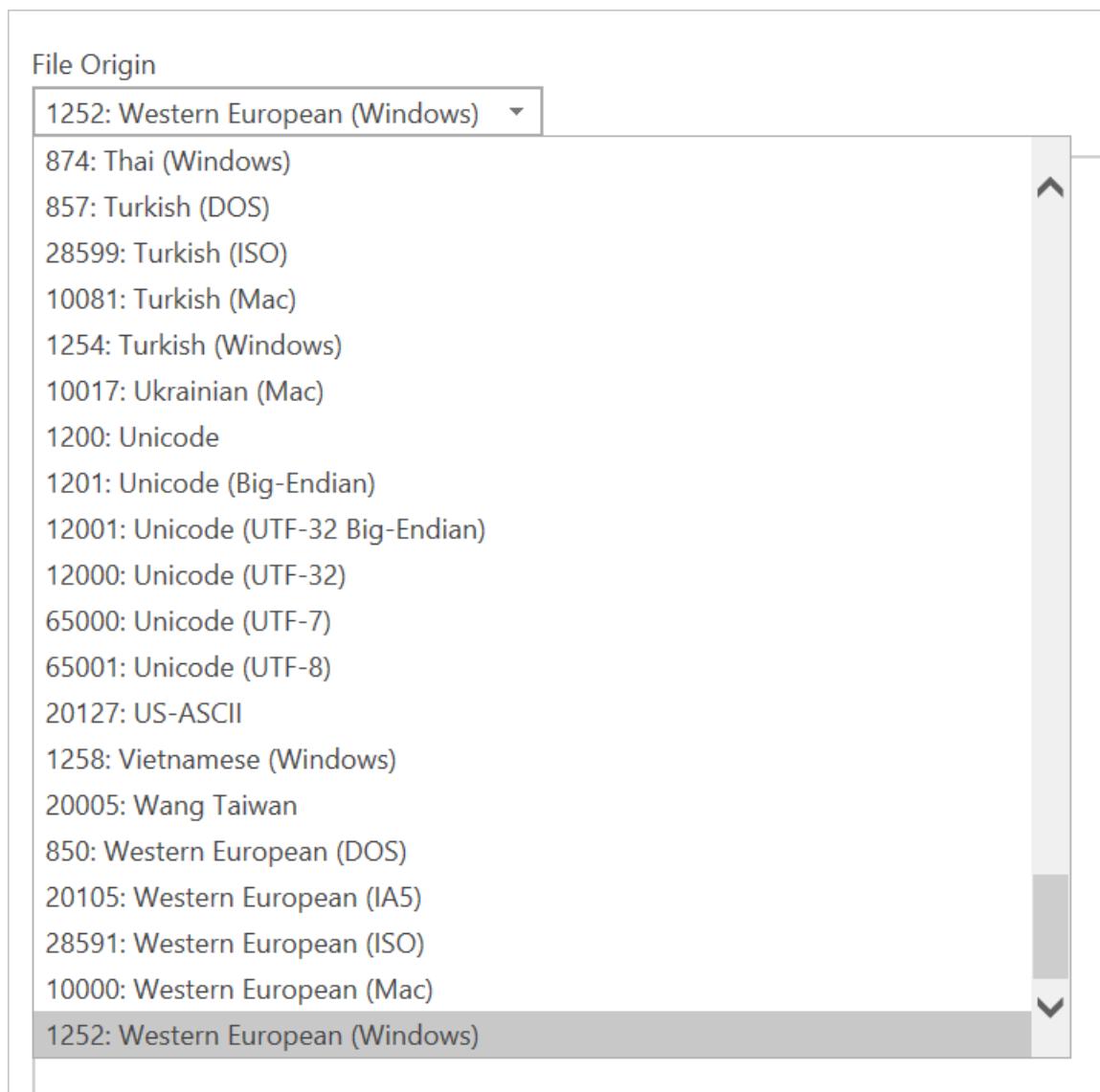
If your text file doesn't have structure, you'll get a single column with a new row per line encoded in the source text. As a sample for unstructured text, you can consider a notepad file with the following contents:

```
Hello world.  
This is sample data.
```

When you load it, you're presented with a navigation screen that loads each of these lines into their own row.



There's only one thing you can configure on this dialog, which is the **File Origin** dropdown select. This dropdown lets you select [which character set](#) was used to generate the file. Currently, character set isn't inferred, and UTF-8 will only be inferred if it starts with a [UTF-8 BOM](#).



CSV

You can find a sample CSV file [here](#).

In addition to file origin, CSV also supports specifying the delimiter and how data type detection will be handled.

Financial Sample.csv

File Origin: 1252: Western European (Windows) Delimiter: Comma Data Type Detection: Based on first 200 rows

Segment	Country	Product	Discount Band	Units Sold	Manufacturing Price	Sale Price	Gross Sales	Discounts	Sales	COGS
Government	United States of America	Paseo	Low	3450	10.00	350.00	1,207,500.00	48,300.00	1,159,200.00	897,000.00
Government	France	Amarilla	None	2750	260.00	350.00	962,500.00	0.00	962,500.00	715,000.00
Government	Germany	Velo	Low	2966	120.00	350.00	1,038,100.00	20,762.00	1,017,338.00	771,160.00
Government	Germany	Amarilla	Low	2966	260.00	350.00	1,038,100.00	20,762.00	1,017,338.00	771,160.00
Government	Germany	Velo	Low	2877	120.00	350.00	1,006,950.00	20,139.00	986,811.00	748,020.00
Government	Germany	VTT	Low	2877	250.00	350.00	1,006,950.00	20,139.00	986,811.00	748,020.00
Government	Canada	Carretera	Low	2852	3.00	350.00	998,200.00	19,964.00	978,236.00	741,520.00
Government	Canada	Paseo	Low	2852	10.00	350.00	998,200.00	19,964.00	978,236.00	741,520.00
Government	France	Amarilla	Medium	2876	260.00	350.00	1,006,600.00	70,462.00	936,138.00	747,760.00
Government	France	Carretera	Low	2155	3.00	350.00	754,250.00	7,542.50	746,707.50	560,300.00
Government	France	Paseo	Low	2155	10.00	350.00	754,250.00	7,542.50	746,707.50	560,300.00
Government	France	Velo	Low	2177	120.00	350.00	761,950.00	30,478.00	731,472.00	566,020.00

Extract Table Using Examples Load Transform Data Cancel

Delimiters available include colon, comma, equals sign, semicolon, space, tab, a custom delimiter (which can be any string), and a fixed width (splitting up text by some standard number of characters).

Delimiter

- Comma
- Colon
- Comma
- Equals Sign
- Semicolon
- Space
- Tab
- Custom--
- Fixed Width--

Carretera Low 2852

The final dropdown allows you to select how you want to handle data type detection. It can be done based on the first 200 rows, on the entire data set, or you can choose to not do automatic data type detection and instead let all columns default to 'Text'. Warning: if you do it on the entire data set it may cause the initial load of the data in the editor to be slower.

Data Type Detection

- Based on first 200 rows
- Based on entire dataset
- Do not detect data types

200.00 350.00 982,500.

Since inference can be incorrect, it's worth double checking settings before loading.

Structured Text

When Power Query can detect structure to your text file, it will treat the text file as a delimiter separated value file, and give you the same options available when opening a CSV—which is essentially just a file with an extension indicating the delimiter type.

For example, if you save the following example as a text file, it will be read as having a tab delimiter rather than unstructured text.

The screenshot shows the 'Get Data' dialog in Power Query. The file path is 'columns.txt'. The 'File Origin' dropdown is set to '1252: Western European (Windows)'. The 'Delimiter' dropdown is set to 'Tab', which is highlighted. The 'Data Type Detection' dropdown is set to 'Based on first 200 rows'. Below these settings, there is a preview area showing a table with three columns: 'Column 1', 'Column 2', and 'Column 3'. The first row contains 'This is a string.' and '1' followed by 'ABC123'. The second row contains 'This is also a string.' and '2' followed by 'DEF456'. At the bottom of the dialog are three buttons: 'Extract Table Using Examples' (disabled), 'Load' (highlighted in yellow), and 'Transform Data'.

This can be used for any kind of other delimiter-based file.

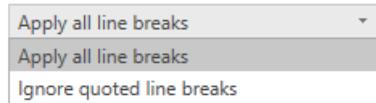
Editing Source

When editing the source step, you'll be presented with a slightly different dialog than when initially loading. Depending on what you are currently treating the file as (that is, text or csv) you'll be presented with a screen with a variety of dropdowns.

The screenshot shows the 'Comma-Separated Values' dialog. It has two radio button options: 'Basic' (selected) and 'Advanced'. The 'File path' field contains 'FILEPATH' with a 'Browse...' button. The 'Open file as' dropdown is set to 'Csv Document'. The 'File origin' dropdown is set to '1252: Western European (Windows)'. The 'Line breaks' dropdown is set to 'Apply all line breaks'. The 'Delimiter' dropdown is set to 'Comma'. At the bottom are 'OK' and 'Cancel' buttons.

The **Line breaks** dropdown will allow you to select if you want to apply line breaks that are inside quotes or not.

Line breaks



For example, if you edit the 'structured' sample provided above, you can add a line break.

Column 1	Column 2	Column 3
This is a string.	1 "ABC 123"	
This is also a string.	2 "DEF456"	

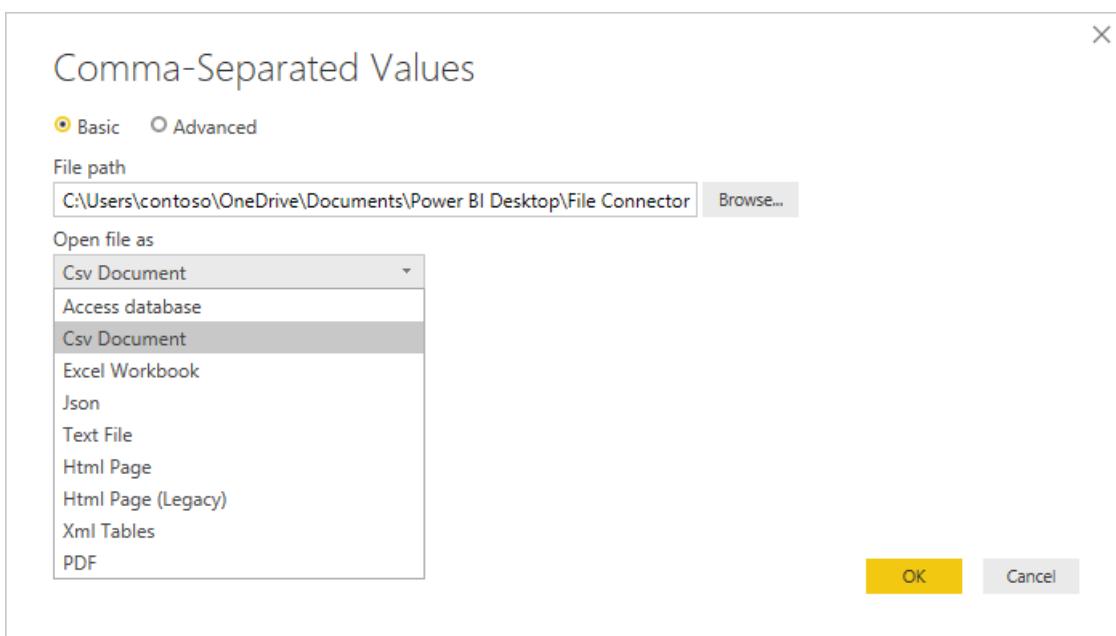
If **Line breaks** is set to **Ignore quoted line breaks**, it will load as if there was no line break (with an extra space).

	Column1	Column2	Column3
1	Column 1	Column 2	Column 3
2	This is a string.	1	ABC 123
3	This is also a string.	2	DEF456

If **Line breaks** is set to **Apply all line breaks**, it will load an extra row, with the content after the line breaks being the only content in that row (exact output may depend on structure of the file contents).

	Column1	Column2	Column3
1	Column 1	Column 2	Column 3
2	This is a string.	1	ABC
3	123"		
4	This is also a string.	2	DEF456

The **Open file as** dropdown will let you edit what you want to load the file as—important for troubleshooting. For structured files that aren't technically CSVs (such as a tab separated value file saved as a text file), you should still have **Open file as** set to CSV. This setting also determines which dropdowns are available in the rest of the dialog.



Text/CSV by Example

Text/CSV By Example in Power Query is a generally available feature in Power BI Desktop. When you use the Text/CSV connector, you'll see an option to **Extract Table Using Examples** on the bottom-left corner of the navigator.

The screenshot shows the 'Extract Table Using Examples' dialog box. At the top, it displays the file name 'Customers_Contoso.txt'. Below that is a 'File Origin' dropdown set to '1252: Western European (Windows)'. The main area contains a preview of data in a table format with one column labeled 'Column1'. The data includes:
List of Customers from March 2020
Last updated 3/16/2020

Alfreds Futterkiste
Maria Anders
Sales Representative
030-0074321

Ana Trujillo Emparedados y helados
Ana Trujillo
Owner
(5) 555-4729

Around the Horn
Thomas Hardy
Sales Representative
(171) 555-7788

Blauer See Delikatessen
Hanna Moos

A tooltip at the bottom left states: 'The data in the preview has been truncated due to size limits.'

At the bottom right are three buttons: 'Load' (yellow), 'Transform Data', and 'Cancel'.

When you select that button, you'll be taken into the **Extract Table Using Examples** page. On this page, you specify sample output values for the data you'd like to extract from your Text/CSV file. After you enter the first cell of the column, other cells in the column are filled out. For the data to be extracted correctly, you may need to enter more than one cell in the column. If some cells in the column are incorrect, you can fix the first incorrect cell and the data will be extracted again. Check the data in the first few cells to ensure that the data has been extracted successfully.

NOTE

We recommend that you enter the examples in column order. Once the column has successfully been filled out, create a new column and begin entering examples in the new column.

Extract Table Using Examples - Customers_Contoso.txt

File Origin
1252: Western European (Windows)

List of Customers from March 2020
Last updated 3/16/2020

Alfreds Futterkiste
Maria Anders
Sales Representative
030-0074321

Ana Trujillo Emparedados y helados
Ana Trujillo
Owner
(5) 555-4729

	Company Name	Contact Name	Contact Title	*
1	Alfreds Futterkiste	Maria Anders	Sales Representative	
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	
3	Around the Horn	Thomas Hardy	Owner	
4	Blauer See Delikatessen	Hanna Moos	Owner (5)	
5	Bon app'	Laurence Lebihan	Owner (5) 555-4729	
6	Bottom-Dollar Markets	Elizabeth Lincoln	Owner (5) 555-4729	Around the Horn
7	B's Beverages	Victoria Ashworth	Owner (5) 555-4729	Around the Horn
8	Cactus Comidas para llevar	Patricia Simpson	Owner (5) 555-4729	Around the Horn
9	Centro comercial Moctezuma	Francisco Chang	Owner (5) 555-4729	Around the Horn Thomas
10	Chop-suey Chinese	Yang Wang	Owner (5) 555-4729	Around the Horn Thomas Hardy
11	Consolidated Holdings	Elizabeth Brown	Sales Representative	
12	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	
13	Du monde entier	Janine Labrune	Owner	
14	Eastern Connection	Ann Devon	Sales Agent	
15	Ernst Handel	Roland Mendel	Sales Manager	
16	Familia Arquibaldo	Aria Cruz	Marketing Assistant	
17	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accounting Manager	
18	Frankenversand	Peter Franken	Marketing Manager	
19	France restauration	Carine Schmitt	Marketing Manager	
20	Franchi S.p.A.	Paolo Accorti	Sales Representative	
21	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Sales Manager	
22	Gourmet Lanches	Andre Fonseca	Sales Associate	

Load Transform Data Cancel

Once you're done constructing that table, you can either select to load or transform the data. Notice how the resulting queries contain a detailed breakdown of all the steps that were inferred for the data extraction. These steps are just regular query steps that you can customize as needed.

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Advanced Editor Properties Choose Columns Remove Columns Keep Rows Remove Rows Manage Columns Reduce Rows Sort Split Column Group By Use First Row as Headers Data Type: Text

Combine Text Analytics Vision Azure Machine Learning AI Insights

Queries [1] Customers_Contoso

= Table.TransformColumnTypes(#"Renamed Columns",{{"Customer Name", type text}, {"Contact Name", type text}, {"Contact Title", type text}})

	Customer Name	Contact Name	Contact Title
1	Alfreds Futterkiste	Maria Anders	Sales Representative
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
3	Around the Horn	Thomas Hardy	Sales Representative
4	Blauer See Delikatessen	Hanna Moos	Sales Representative
5	Bon app'	Laurence Lebihan	Owner
6	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager
7	B's Beverages	Victoria Ashworth	Sales Representative
8	Cactus Comidas para llevar	Patricia Simpson	Sales Agent
9	Centro comercial Moctezuma	Francisco Chang	Marketing Manager
10	Chop-suey Chinese	Yang Wang	Owner
11	Consolidated Holdings	Elizabeth Brown	Sales Representative
12	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator
13	Du monde entier	Janine Labrune	Owner
14	Eastern Connection	Ann Devon	Sales Agent
15	Ernst Handel	Roland Mendel	Sales Manager
16	Familia Arquibaldo	Aria Cruz	Marketing Assistant
17	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accounting Manager
18	Frankenversand	Peter Franken	Marketing Manager
19	France restauration	Carine Schmitt	Marketing Manager
20	Franchi S.p.A.	Paolo Accorti	Sales Representative
21	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Sales Manager
22	Gourmet Lanches	Andre Fonseca	Sales Associate

3 COLUMNS, 29 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 12:14 PM

NOTE

This feature is being released soon in Power Query Online. More information: [Text/CSV by example data extraction](#)

Troubleshooting

Loading Files from the Web

If you're requesting text/csv files from the web and also promoting headers, and you're retrieving enough files that you need to be concerned with potential throttling, you should consider wrapping your `Web.Contents` call with `Binary.Buffer()`. In this case, buffering the file before promoting headers will cause the file to only be requested once.

Unstructured text being interpreted as structured

In rare cases, a document that has similar comma numbers across paragraphs might be interpreted to be a CSV. If this issue happens, edit the **Source** step in the Query Editor, and select **Text** instead of **CSV** in the **Open File As** dropdown select.

Error: Connection closed by host

When loading Text/CSV files from a web source and also promoting headers, you might sometimes encounter the following errors: “An existing connection was forcibly closed by the remote host” or

“Received an unexpected EOF or 0 bytes from the transport stream.” These errors might be caused by the host employing protective measures and closing a connection which might be temporarily paused, for example, when waiting on another data source connection for a join or append operation. To work around these errors, try adding a `Binary.Buffer` (recommended) or `Table.Buffer` call, which will download the file, load it into memory, and immediately close the connection. This should prevent any pause during download and keep the host from forcibly closing the connection before the content is retrieved.

The following example illustrates this workaround. This buffering needs to be done before the resulting table is passed to `Table.PromoteHeaders`.

- Original:

```
Csv.Document(Web.Contents("https://.../MyFile.csv"))
```

- With `Binary.Buffer`:

```
Csv.Document(Binary.Buffer(Web.Contents("https://.../MyFile.csv")))
```

- With `Table.Buffer`:

```
Table.Buffer(Csv.Document(Web.Contents("https://.../MyFile.csv")))
```

TIBCO(R) Data Virtualization

1/15/2022 • 3 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by TIBCO, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the TIBCO website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Database (Username/Password) Windows Authentication (NTLM/Kerberos)
Function Reference Documentation	—

Prerequisites

To access the TIBCO eDelivery site, you must have purchased TIBCO software. There's no TIBCO license required for the TIBCO(R) Data Virtualization (TDV) software—a TIBCO customer only needs to have a valid contract in place. If you don't have access, then you'll need to contact the TIBCO admin in your organization.

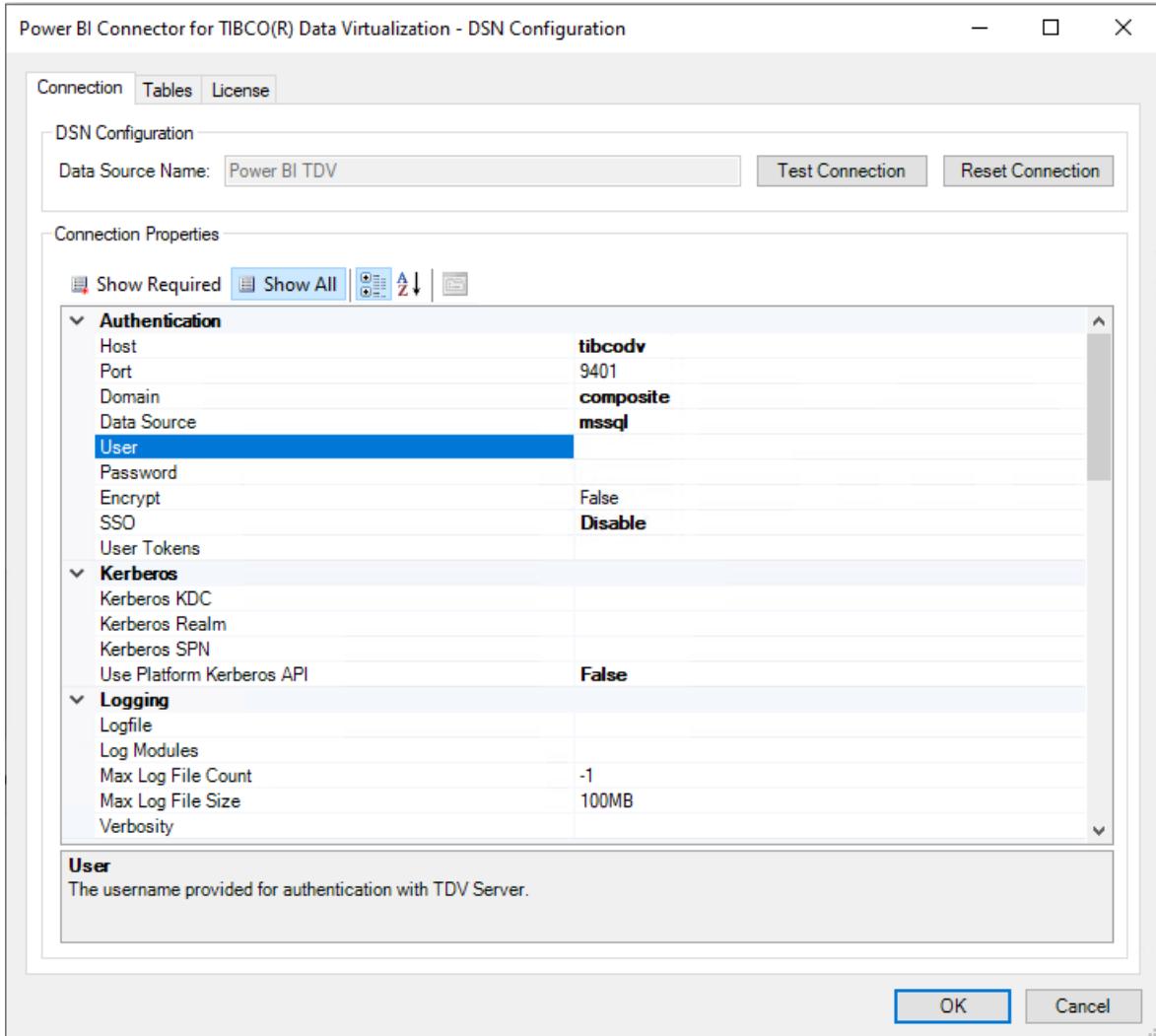
The Power BI Connector for TIBCO(R) Data Virtualization must first be downloaded from <https://edelivery.tibco.com> and installed on the machine running Power BI Desktop. The eDelivery site downloads a ZIP file (for example, TIB_tdv_drivers_<VERSION>.all.zip*zip where <VERSION>=TDV Version) that contains an installer program that installs all TDV client drivers, including the Power BI Connector.



Once the connector is installed, configure a data source name (DSN) to specify the connection properties needed to connect to the TIBCO(R) Data Virtualization server.

NOTE

The DSN architecture (32-bit or 64-bit) needs to match the architecture of the product where you intend to use the connector.



NOTE

Power BI Connector for TIBCO(R) Data Virtualization is the driver used by the TIBCO(R) Data Virtualization connector to connect Power BI Desktop to TDV.

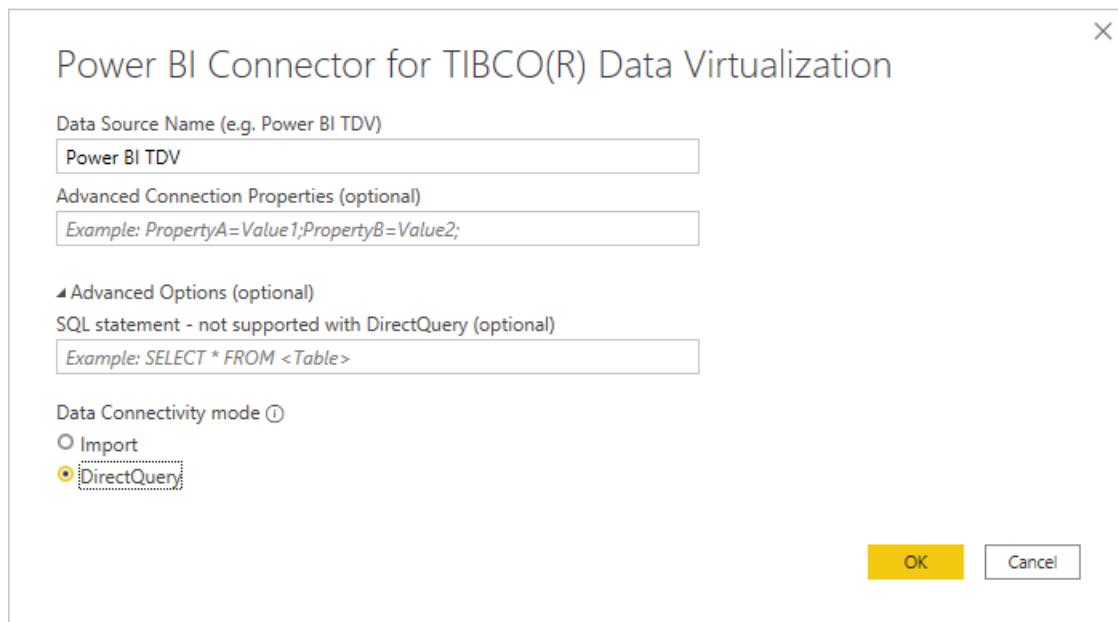
Capabilities Supported

- Import
- DirectQuery (Power BI Desktop only)
- Advanced Connection Properties
- Advanced
 - Native SQL statement

Connect to a TIBCO(R) Data Virtualization server from Power Query Desktop

Once the matching connector is installed and a DSN is configured, you can connect to a TIBCO(R) Data Virtualization server. To make the connection, take the following steps:

1. Select the **TIBCO(R) Data Virtualization** option in the connector selection.
2. In the **Power BI Connector for TIBCO(R) Data Virtualization** dialog that appears, provide the Data Source Name.



3. Select either the **Import** or **DirectQuery** data connectivity mode.
4. If this is the first time you're connecting to this database, select the authentication type. If applicable, enter the needed credentials. Then select **Connect**.
 - **Anonymous**: Uses the credentials stored in the DSN.
 - **Basic**: Username/password are submitted when creating the connection.
 - **Windows**: Authenticates using the current Windows user. This authentication type requires the SSO connection property to be set. When using Kerberos, the **Use Platform Kerberos API** must be set to true to use the current Windows user.



5. In **Navigator**, select the database information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

productid	productname	productdescription	categoryid	serialnumber	unitprice	reorder
1	Maxtific 40GB ATA133 7200	Maxtific Storage 40 GB	1	221-887-3458	89.99	
2	Mega Zip 750MB USB 2.0	Mega Zip 750 MB	1	5-76-9876	187.67	
3	Mega Zip 250MB	Parallel External w/o Disk	1	5-75-9924	140.95	
4	Maxtific120 GB ATA133 7200	Maxtific Storage 120 GB	1	221-889-3460	151.95	
5	Mega Zip 250MB SCSI External	Mega Zip 250 MB	2	9-2244-038	144.95	
6	Maxtific 60GB ATA133 7200	Maxtific Storage 60 GB	1	220-768-7642	109.99	
7	RedPagoda PCI 56K	Fax Modem	5	RDP-8546	14.95	
8	Deluxe RedPagoda PCI V.90 56K	Voice Modem	5	RDP-9864	18	
9	Acme Super Memory	Acme Memory	4	MEM-212	25	
10	Lexington Z23	Lexington Color Jet Printer	6	PRNT-LEX35	99	
11	Printerific C42UX	Inkjet Printer	6	PRNT-RFC764	129.99	
12	Processor Queen IV 2.4 GHz	Queen 2.4 GHz	8	CPU-7543	215.99	
13	Processor King 2.0 GHz	King 2.0 GHz	8	CPU-C538	99.99	
14	Rad Expert 9000 PRO 128 MB	Rad Video Card	9	VID-RAD89388	129.99	
15	Rad 2000 PRO 32 MB	Rad Video Card	9	VID-RAD76459	40	
16	Force Pro M64 32MB	Force Pro Video Card	9	VID-FP765549	41	
17	Acme 950MHz w/o fan	Acme Processor	8	CPU-ACM8733	39.99	
18	Acme 40GB ATA100 7200	Acme Storage 40 GB	1	HD-ACM4483-2	85.95	
19	Widget 1	Widget model	7	W1	22	
20	Widget 2	Widget model	7	W2	22	
21	Widget 3	Widget model	7	W3	22	
22	Widget 4	Widget model	7	W4	22	
23	Widget 5	Widget model	7	W5	22	

Connect using advanced options

Power Query Desktop provides a set of advanced options that you can add to your query if needed.

The following table lists all of the advanced options you can set in Power Query Desktop.

ADVANCED OPTION	DESCRIPTION
Advanced Connection Properties	Takes a semi-colon separated list of connection properties. This option is used to specify other connection properties not configured in the DSN.
SQL statement	For information, go to Import data from a database using native database query .

Once you've selected the advanced options you require, select **OK** in Power Query Desktop to connect to your TIBCO(R) Data Virtualization Server.

Kerberos-based single sign-on (SSO) for TIBCO(R) Data Virtualization

The TIBCO(R) Data Virtualization connector now supports Kerberos-based single sign-on (SSO).

To use this feature:

1. Sign in to your Power BI account, and navigate to the **Gateway management** page.
2. Add a new data source under the gateway cluster you want to use.
3. Select the connector in the **Data Source Type** list.
4. Expand the **Advanced Settings** section.
5. Select the option to **Use SSO via Kerberos for DirectQuery queries** or **Use SSO via Kerberos for DirectQuery and Import queries**.

The screenshot shows the 'Add Data Source' dialog. On the left, under 'GATEWAY CLUSTERS', 'TestGateway23' is expanded, showing 'TestDataSource001' and a 'New data source' button. A 'Test all connections' button is also present. The main area has two tabs: 'Data Source Settings' (selected) and 'Users'. Under 'Data Source Settings', the 'Data Source Name' is 'New data source', 'Data Source Type' is 'Power BI Connector for TIBCO(R) Data Virtualization', and 'Data Source Name (e.g. Power BI TDV)' is empty. The 'Authentication Method' dropdown is set to 'Select an authentication method'. There is a checkbox for 'Skip Test Connection'. Under 'Advanced settings', there is a 'Advanced Connection Properties' section, a note about SQL statements, and two checkboxes for Kerberos-based SSO: 'Use SSO via Kerberos for DirectQuery queries' (with a note about DirectQuery datasets) and 'Use SSO via Kerberos for DirectQuery And Import queries' (with a note about import using dataset owner credentials). The 'Privacy Level setting for this data source' is set to 'Organizational'. At the bottom are 'Add' and 'Discard' buttons.

More information: [Configure Kerberos-based SSO from Power BI service to on-premises data sources](#)

Usercube

1/15/2022 • 2 minutes to read • [Edit Online](#)

NOTE

The following connector article is provided by Usercube, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the Usercube website and use the support channels there.

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets)
Authentication Types Supported	Client Credentials
Function Reference Documentation	—

Prerequisites

You must have a Usercube instance with the *PowerBI* option.

Capabilities supported

- Import

Connect to Usercube from Power Query Desktop

To connect to a Usercube server:

1. Launch Power BI Desktop and enter the **Get Data** experience.
2. From the **Other** category, select **Usercube**, and then select **Connect**.

Get Data

Search

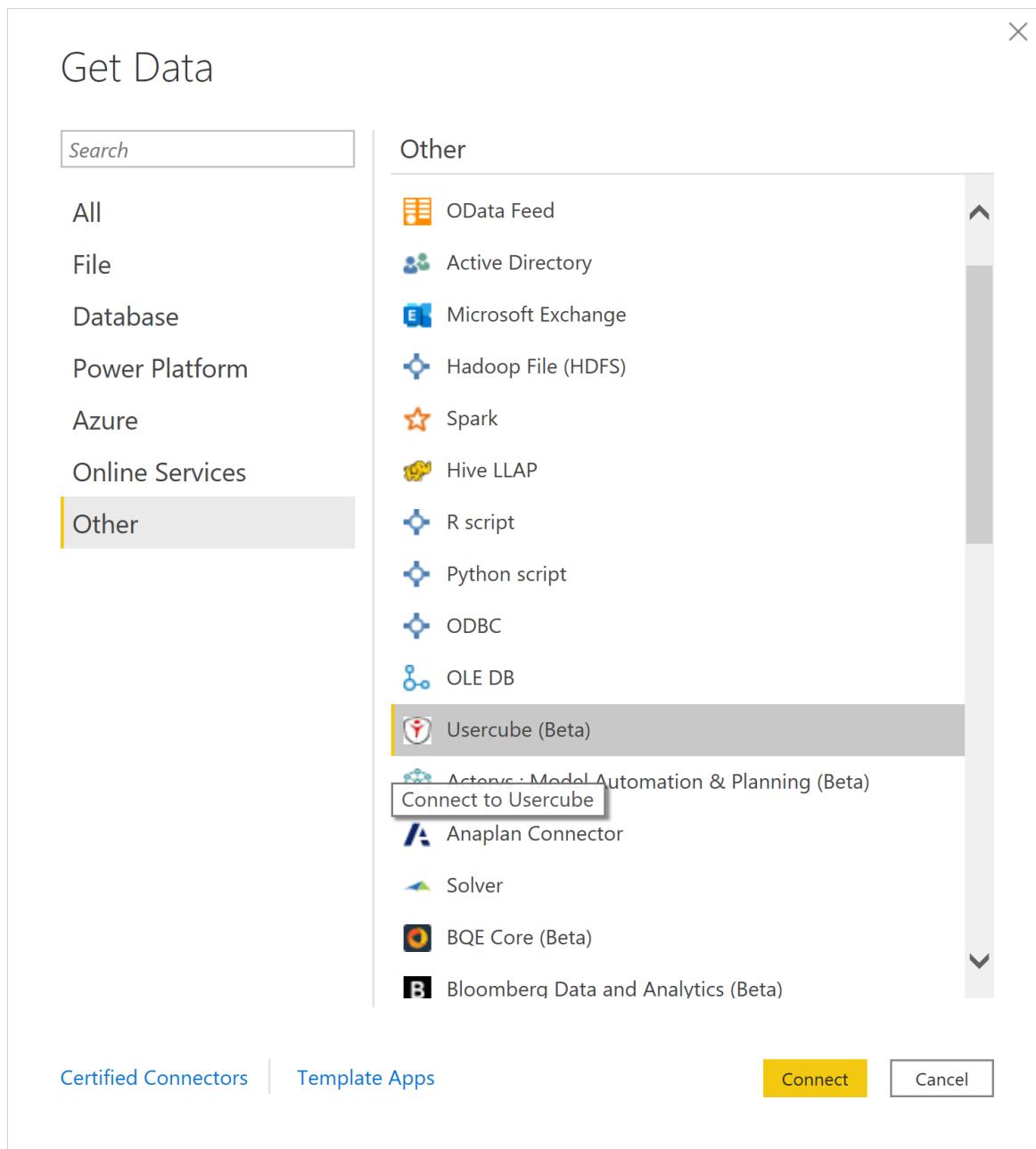
- All
- File
- Database
- Power Platform
- Azure
- Online Services
- Other

Other

- OData Feed
- Active Directory
- Microsoft Exchange
- Hadoop File (HDFS)
- Spark
- Hive LLAP
- R script
- Python script
- ODBC
- OLE DB
- Usercube (Beta)
- Actvys : Model Automation & Planning (Beta)
- Anaplan Connector
- Solver
- BQE Core (Beta)
- Bloomberg Data and Analytics (Beta)

Certified Connectors | Template Apps

Connect Cancel



3. You'll be prompted to specify your Usercube server's URL.

Usercube

Server URL ⓘ

OK Cancel



4. Enter the client credentials. The *Client Id* must be built from the *Identifier* of an *OpenIdClient* element. This element is defined in the configuration of your Usercube instance. To this identifier, you must concatenate the @ character and the domain name of the Usercube instance.



5. In **Navigator**, select the data you require. Then, either select **Transform data** to transform the data in the Power Query Editor, or choose **Load** to load the data in Power BI.

AD User (nominative) Id	Display Name (AD User (nominative))	accountExpires (AD Use...
1	robert.roy@acme.com	132834636000000000
2	adm.olivier.david@acme.com	9223372036854775807
3	veronique.paul@acme.com	9223372036854775807
4	francoise.lacroix@acme.com	9223372036854775807
5	julien.morel1@acme.com	9223372036854775807
6	guillaume.garcia@acme.com	9223372036854775807
7	jean.lemaitre@acme.com	132845184000000000
8	philippe.rey@acme.com	9223372036854775807
9	alain.payet@acme.com	9223372036854775807
10	patrick.huet1@acme.com	9223372036854775807

The data in the preview has been truncated due to size limits.

Web

1/15/2022 • 9 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous Windows Basic Web API Organizational Account
Function Reference Documentation	Web.Page Web.BrowserContents

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

- Internet Explorer 10

Capabilities supported

- Basic
- Advanced
 - URL parts
 - Command timeout
 - HTTP request header parameters

Load Web data using Power Query Desktop

To load data from a web site with Power Query Desktop:

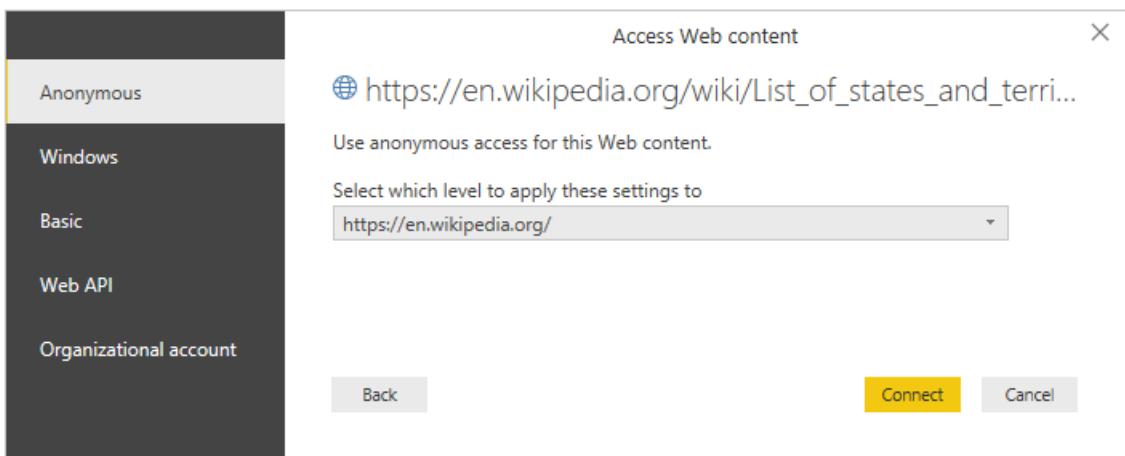
1. Select **Get Data > Web** in Power BI or **From Web** in the **Data** ribbon in Excel.
2. Choose the **Basic** button and enter a URL address in the text box. For example, enter `https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States`. Then select **OK**.



If the URL address you enter is invalid, a  warning icon will appear next to the URL textbox.

If you need to construct a more advanced URL before you connect to the website, go to [Load Web data using an advanced URL](#).

3. Select the authentication method to use for this web site. In this example, select **Anonymous**. Then select the level to you want to apply these settings to—in this case, <https://en.wikipedia.org/>. Then select **Connect**.



The available authentication methods for this connector are:

- **Anonymous**: Select this authentication method if the web page doesn't require any credentials.
- **Windows**: Select this authentication method if the web page requires your Windows credentials.
- **Basic**: Select this authentication method if the web page requires a basic user name and password.
- **Web API**: Select this method if the web resource that you're connecting to uses an API Key for authentication purposes.
- **Organizational account**: Select this authentication method if the web page requires organizational account credentials.

NOTE

When uploading the report to the Power BI service, only the **anonymous**, **Windows** and **basic** authentication methods are available.

The level you select for the authentication method determines what part of a URL will have the authentication method applied to it. If you select the top-level web address, the authentication method you select here will be used for that URL address or any subaddress within that address. However, you might not want to set the top URL address to a specific authentication method because different subaddresses could require different authentication methods. For example, if you were accessing two

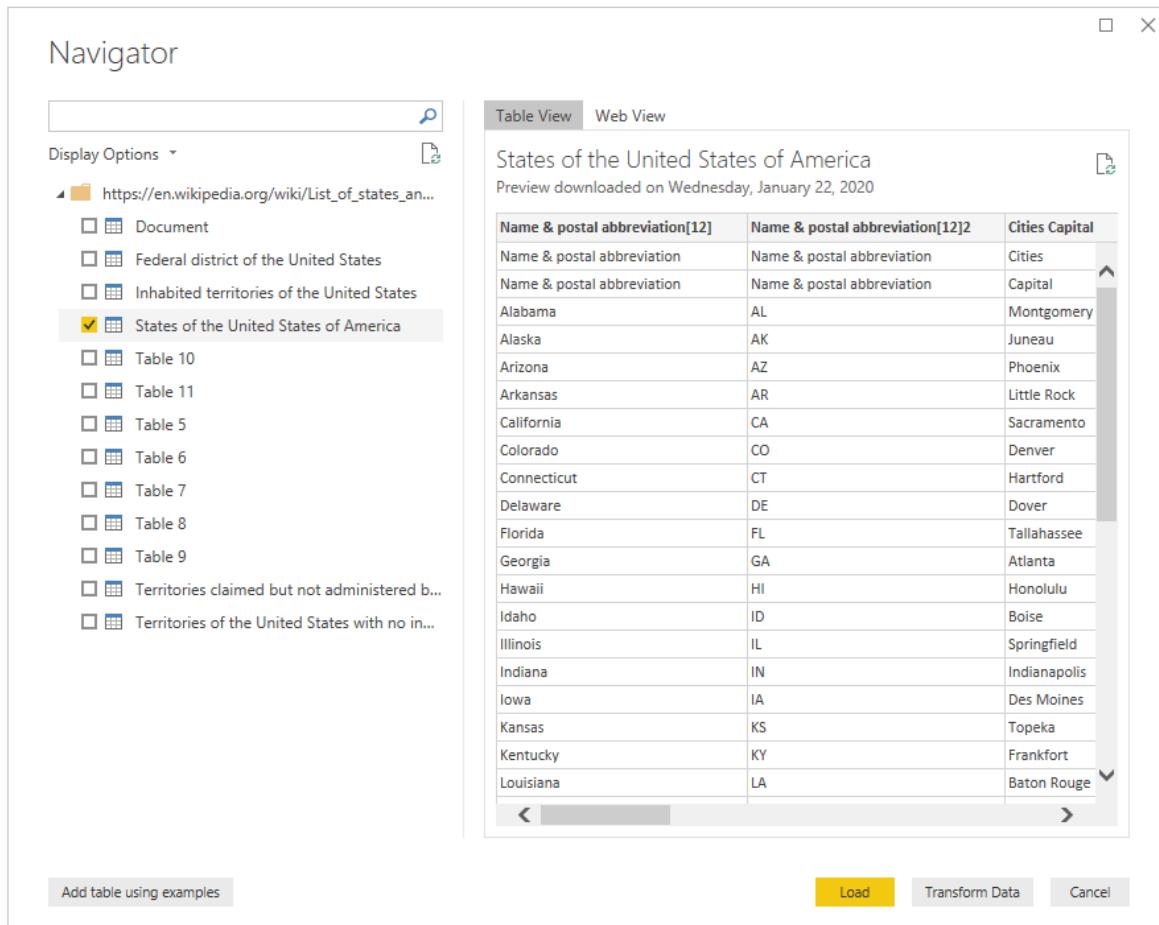
separate folders of a single SharePoint site and wanted to use different Microsoft Accounts to access each one.

Once you've set the authentication method for a specific web site address, you won't need to select the authentication method for that URL address or any subaddress again. For example, if you select the <https://en.wikipedia.org/> address in this dialog, any web page that begins with this address won't require that you select the authentication method again.

NOTE

If you need to change the authentication method later, go to [Changing the authentication method](#).

4. From the **Navigator** dialog, you can select a table, then either transform the data in the Power Query editor by selecting **Transform Data**, or load the data by selecting **Load**.



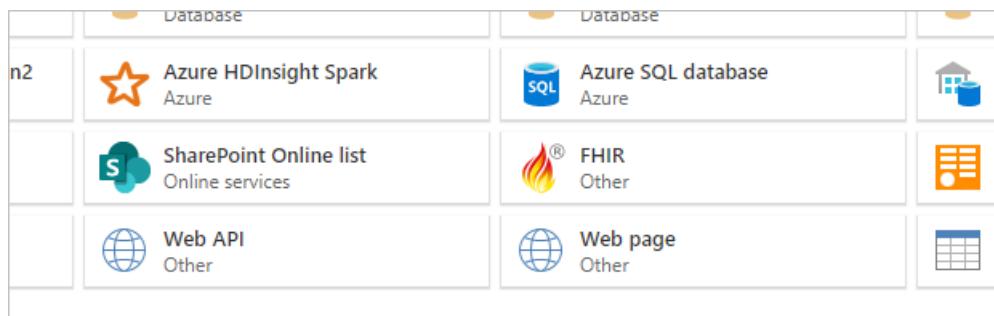
The right side of the **Navigator** dialog displays the contents of the table you select to transform or load. If you're uncertain which table contains the data you're interested in, you can select the **Web View** tab. The web view lets you see the entire contents of the web page, and highlights each of the tables that have been detected on that site. You can select the check box above the highlighted table to obtain the data from that table.

On the lower left side of the **Navigator** dialog, you can also select the **Add table using examples** button. This selection presents an interactive window where you can preview the content of the web page and enter sample values of the data you want to extract. For more information on using this feature, go to [Get webpage data by providing examples](#).

Load Web data using Power Query Online

To load data from a web site with Power Query Online:

- From the Get Data dialog box, select either Web page or Web API.



In most cases, you'll want to select the Web page connector. For security reasons, you'll need to use an [on-premises data gateway](#) with this connector. The Web Page connector requires a gateway because HTML pages are retrieved using a browser control, which involves potential security concerns. This concern isn't an issue with Web API connector, as it doesn't use a browser control.

In some cases, you might want to use a URL that points at either an API or a file stored on the web. In those scenarios, the Web API connector (or file-specific connectors) would allow you to move forward without using an on-premises data gateway.

Also note that if your URL points to a file, you should [use the specific file connector](#) instead of the Web page connector.

- Enter a URL address in the text box. For this example, enter

`https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States`.

The screenshot shows the 'Connection settings' section of the 'Web page' configuration. It includes fields for 'URL' (containing 'List_of_states_and_territories_of_the_United_States') and 'On-premises data gateway' (set to '(none)'). A note below states: 'The Web Page connector requires a gateway. If you want to access a non-HTML resource without the gateway, please use the Web API connector or one of the File connectors instead.'

- Select the name of your on-premises data gateway.

The screenshot shows the 'Connection settings' section of the 'Web page' configuration after selecting a gateway. The 'On-premises data gateway' dropdown now shows '[Admin] 10TestGateway'.

- Select the authentication method you'll use to connect to the web page.

The available authentication methods for this connector are:

- **Anonymous:** Select this authentication method if the web page doesn't require any credentials.
- **Windows:** Select this authentication method if the web page requires your Windows credentials.
- **Basic:** Select this authentication method if the web page requires a basic user name and password.
- **Organizational account:** Select this authentication method if the web page requires organizational account credentials.

Once you've chosen the authentication method, select **Next**.

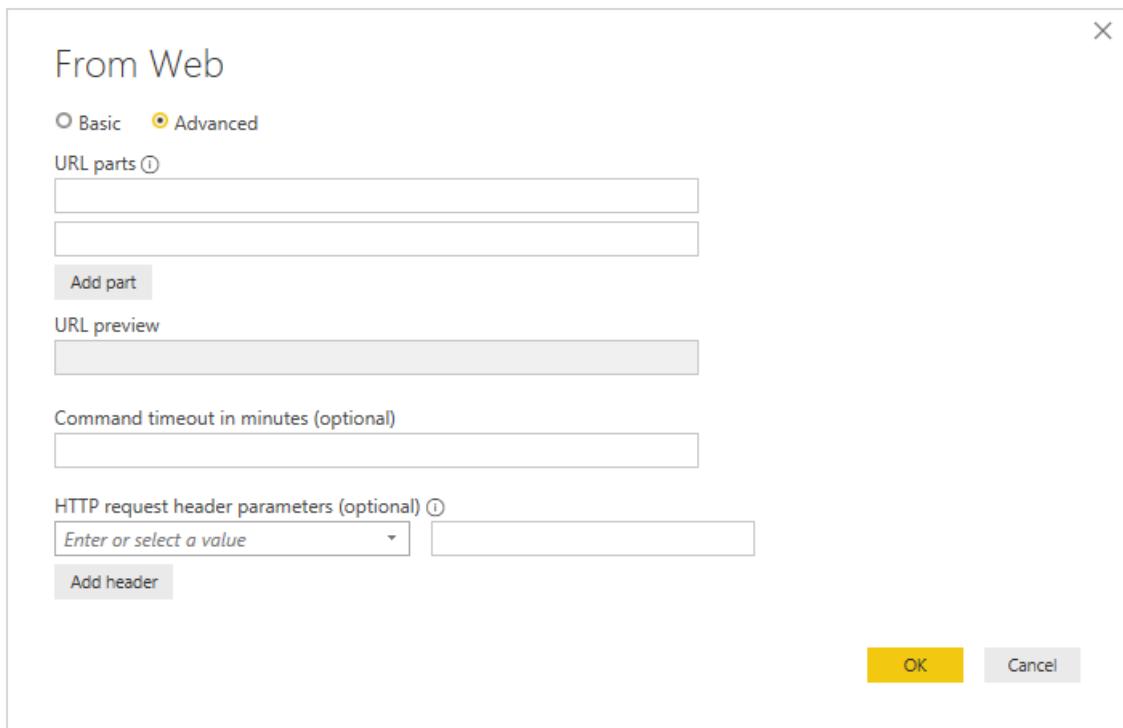
5. From the **Navigator** dialog, you can select a table, then transform the data in the Power Query Editor by selecting **Transform Data**.

Flag, name and postal abbreviation[12]	Flag, name and postal abbreviation[12]2	Cities Capital
Flag, name and postal abbreviation	Flag, name and postal abbreviation	Cities
Flag, name and postal abbreviation	Flag, name and postal abbreviation	Capital
Alabama	AL	Montgomery
Alaska	AK	Juneau
Arizona	AZ	Phoenix
Arkansas	AR	Little Rock
California	CA	Sacramento
Colorado	CO	Denver
Connecticut	CT	Hartford
Delaware	DE	Dover
Florida	FL	Tallahassee
Georgia	GA	Atlanta
Hawaii	HI	Honolulu
Idaho	ID	Boise
Illinois	IL	Springfield
Indiana	IN	Indianapolis
Iowa	IA	Des Moines
Kansas	KS	Topeka
Kentucky	KY	Frankfort
Louisiana	LA	Baton Rouge

Load Web data using an advanced URL

When you select **Get Data > From Web** in Power Query Desktop, in most instances you'll enter URLs in the **Basic** setting. However, in some cases you may want to assemble a URL from its separate parts, set a timeout for

the connection, or provide individualized URL header data. In this case, select the **Advanced** option in the **From Web** dialog box.



Use the **URL parts** section of the dialog to assemble the URL you want to use to get data. The first part of the URL in the **URL parts** section most likely would consist of the scheme, authority, and path of the URI (for example, `http://contoso.com/products/`). The second text box could include any queries or fragments that you would use to filter the information provided to the web site. If you need to add more than one part, select **Add part** to add another URL fragment text box. As you enter each part of the URL, the complete URL that will be used when you select **OK** is displayed in the **URL preview** box.

Depending on how long the POST request takes to process data, you may need to prolong the time the request continues to stay connected to the web site. The default timeout for both POST and GET is 100 seconds. If this timeout is too short, you can use the optional **Command timeout in minutes** to extend the number of minutes you stay connected.

You can also add specific request headers to the POST you send to the web site using the optional **HTTP request header parameters** drop-down box. The following table describes the request headers you can select.

REQUEST HEADER	DESCRIPTION
Accept	Specifies the response media types that are acceptable.
Accept-Charset	Indicates which character sets are acceptable in the textual response content.
Accept-Encoding	Indicates what response content encodings are acceptable in the response.
Accept-Language	Indicates the set of natural languages that are preferred in the response.
Cache-Control	Indicates the caching policies, specified by directives, in client requests and server responses.

REQUEST HEADER	DESCRIPTION
Content-Type	Indicates the media type of the content.
If-Modified-Since	Conditionally determines if the web content has been changed since the date specified in this field. If the content hasn't changed, the server responds with only the headers that have a 304 status code. If the content has changed, the server will return the requested resource along with a status code of 200.
Prefer	Indicates that particular server behaviors are preferred by the client, but aren't required for successful completion of the request.
Range	Specifies one or more subranges of the selected representation data.
Referer	Specifies a URI reference for the resource from which the target URI was obtained.

Import files from the web

Normally when you import a local on-premises file in Power Query Desktop, you'll use the specific file-type connector to import that file, for example, the JSON connector to import a JSON file or the CSV connector to import a CSV file. However, if you're using Power Query Desktop and the file you want to import is located on the web, you must use the Web connector to import that file. As in the local case, you'll then be presented with the table that the connector loads by default, which you can then either Load or Transform.

The following file types are supported by the Web Connector:

- [Access database](#)
- [CSV document](#)
- [Excel workbook](#)
- [JSON](#)
- [Text file](#)
- [HTML page](#)
- [XML tables](#)
- [PDF](#)

For example, you could use the following steps to import a JSON file on the <https://contoso.com/products> web site:

1. From the **Get Data** dialog box, select the **Web** connector.
2. Choose the **Basic** button and enter the address in the **URL** box, for example:

http://contoso.com/products/Example_JSON.json



3. Select **OK**.
4. If this is the first time you're visiting this URL, select **Anonymous** as the authentication type, and then select **Connect**.
5. Power Query Editor will now open with the data imported from the JSON file. Select the **View** tab in the Power Query Editor, then select **Formula Bar** to turn on the formula bar in the editor.

additionalImageLinks	List
adult	False
adwordsRedirect	http://contoso.com/hury
ageGroup	kids
availability	in stock

As you can see, the Web connector returns the web contents from the URL you supplied, and then automatically wraps the web contents in the appropriate document type specified by the URL (`Json.Document` in this example).

Handling dynamic web pages

Web pages that load their content dynamically might require special handling. If you notice sporadic errors in your web queries, it's possible that you're trying to access a dynamic web page. One common example of this type of error is:

1. You refresh the site.
2. You see an error (for example, "the column 'Foo' of the table wasn't found").
3. You refresh the site again.
4. No error occurs.

These kinds of issues are usually due to timing. Pages that load their content dynamically can sometimes be inconsistent since the content can change after the browser considers loading complete. Sometimes `Web.BrowserContents` downloads the HTML after all the dynamic content has loaded. Other times the changes are still in progress when it downloads the HTML, leading to sporadic errors.

The solution is to pass the `WaitFor` option to `Web.BrowserContents`, which indicates either a selector or a length of time that should be waited for before downloading the HTML.

How can you tell if a page is dynamic? Usually it's pretty simple. Open the page in a browser and watch it load. If the content shows up right away, it's a regular HTML page. If it appears dynamically or changes over time, it's a dynamic page.

See also

- [Extract data from a Web page by example](#)
- [Troubleshooting the Power Query Web connector](#)

Get webpage data by providing examples

1/15/2022 • 2 minutes to read • [Edit Online](#)

Getting data from a web page lets users easily extract data from web pages. Often however, data on Web pages aren't in tidy tables that are easy to extract. Getting data from such pages can be challenging, even if the data is structured and consistent.

There's a solution. With the *Get Data from Web by example* feature, you can essentially show Power Query data you want to extract by providing one or more examples within the connector dialog. Power Query gathers other data on the page that match your examples. With this solution you can extract all sorts of data from Web pages, including data found in tables *and* other non-table data.

Name	Price
1 UNO®	\$9.99
2 MONOPOLY PL...	\$14.99
3 Zombie Party	\$9.99
4 ARCADE GAME...	\$1.99
5 Let Them Come	\$7.99
6 Rare Replay	\$25.99
7 ARCADE GAME...	\$7.99
8 ARCADE GAME...	\$1.99
9 Chess Ultra	\$12.49
10 MONOPOLY F...	\$19.99
11 ARCADE GAME...	\$1.99
12 Babylon 2055 P...	\$4.99
13 Jeopardy!	\$19.99
14 ARCADE GAME...	\$1.99
15 Q*Bert REBOO...	\$9.99
16 Pure Chess Gra...	\$12.49
17 Atari Flashback...	\$19.99
18 Deem-Omn...	Kd.99

NOTE

Prices listed in the images are for example purposes only.

Using Get Data from Web by example

Select the **Web** option in the connector selection, and then select **Connect** to continue.

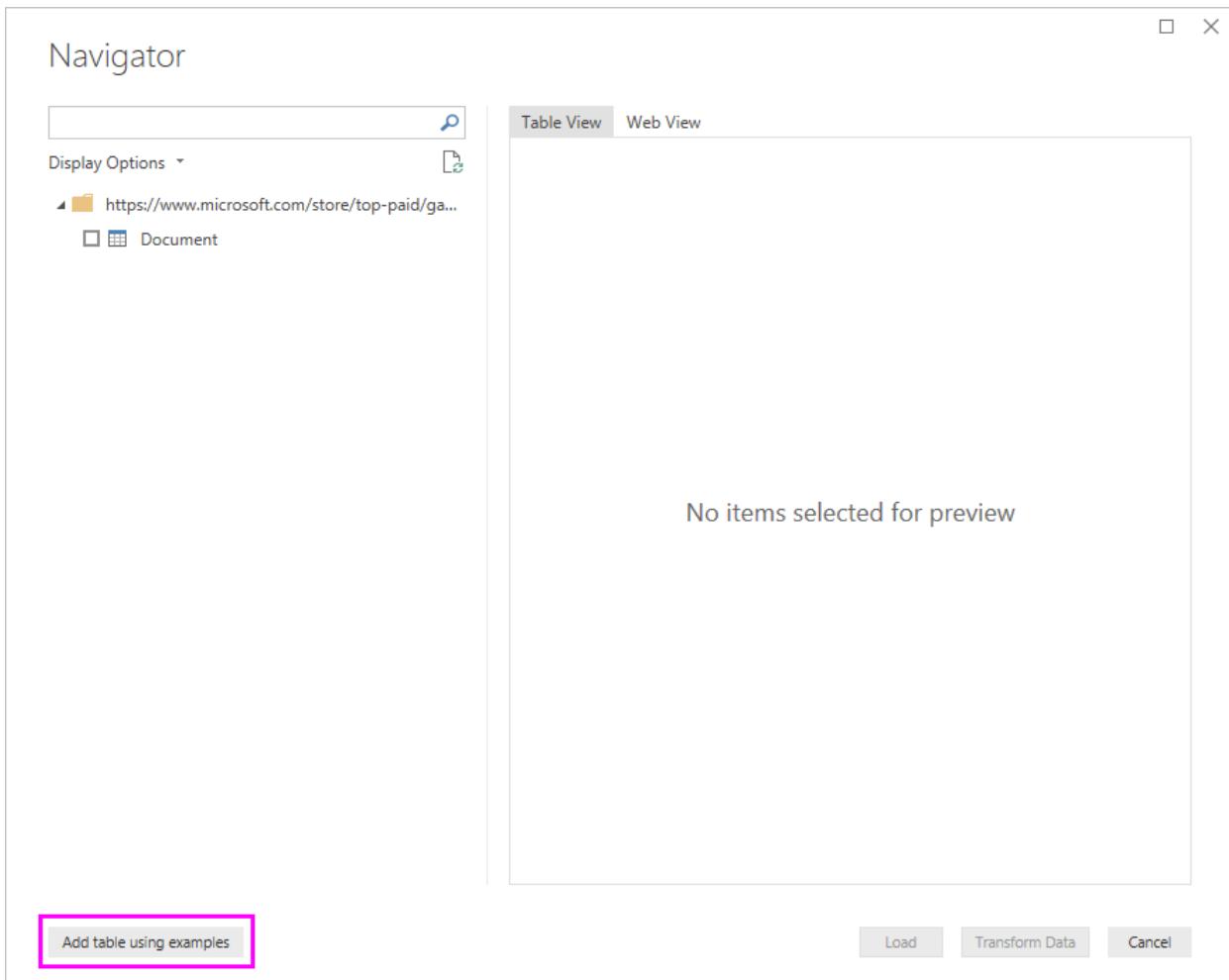
In **From Web**, enter the URL of the Web page from which you'd like to extract data. In this article, we'll use the Microsoft Store Web page, and show how this connector works.

If you want to follow along, you can use the [Microsoft Store URL](https://www.microsoft.com/store/top-paid/games/xbox?category=classics) that we use in this article:

<https://www.microsoft.com/store/top-paid/games/xbox?category=classics>



When you select **OK**, you're taken to the **Navigator** dialog box where any autodetected tables from the Web page are presented. In the case shown in the image below, no tables were found. Select **Add table using examples** to provide examples.



Add table using examples presents an interactive window where you can preview the content of the Web page. Enter sample values of the data you want to extract.

In this example, you'll extract the *Name* and *Price* for each of the games on the page. You can do that by specifying a couple of examples from the page for each column. As you enter examples, Power Query extracts data that fits the pattern of example entries using smart data extraction algorithms.

From Web

Showing 1 - 46 of 46 results

Category	Item	Rating	Price
Best-rated	UNO®	★★★★★	\$9.99
Coming Soon	MONOPOLY PLUS	★★★★★	\$14.99*
Game Demos	ARCADE GAME SERIES: PAC-MAN	★★★★★	\$29.99*
Game Previews	Rare Replay	★★★★★	\$3.99
Most Played			
New			
Total			

Site feedback

Questions? Talk to an expert

Column1	Column2	*
1 UNO®	\$9.99	
2 Monopoly Plus	\$14.99	
3 ARCADE GAME...	\$3.99	
4 Rare Replay	\$29.99	
5 MONOPOLY F...	\$19.99	
6 ARCADE GAME...	\$3.99	
7 ARCADE GAME...	\$3.99	
8 Chess Ultra	\$12.49	
9 ARCADE GAME...	\$7.99	
10 Jeopardy!	\$19.99	
11 Babylon 2055 P...	\$4.99	
12 Zombie Party	\$9.99	
13 ARCADE GAME...	\$3.99	
14 The Disney Af...	\$10.00	

NOTE

Value suggestions only include values less than or equal to 128 characters in length.

Once you're happy with the data extracted from the Web page, select OK to go to Power Query Editor. You can then apply more transformations or shape the data, such as combining this data with other data sources.

Queries [1]

Table 1

Column1 Column2

1 UNO®	\$9.99
2 MONOPOLY PLUS	\$14.99
3 ARCADE GAME SERIES: PAC-MAN	\$3.99
4 The Disney Afternoon Collection	\$19.99
5 MONOPOLY FAMILY FUN PACK	\$19.99
6 ARCADE GAME SERIES: Ms. PAC-MAN	\$3.99
7 ARCADE GAME SERIES 3-in-1 Pack	\$7.99
8 ARCADE GAME SERIES: GALAGA	\$3.99
9 Paranaautical Activity	\$9.99
10 Rare Replay	Included

2 COLUMNS, 90 ROWS Column profiling based on top 1000 rows

Query Settings

Properties

Name: Table 1

All Properties

Applied Steps

Source: Extracted Table From Html

Changed Type

PREVIEW DOWNLOADED AT 3:05 PM

See also

- [Add a column from examples](#)
- [Shape and combine data](#)
- [Getting data](#)
- [Troubleshooting the Power Query Web connector](#)

Troubleshooting the Web connector

1/15/2022 • 5 minutes to read • [Edit Online](#)

Connecting to Microsoft Graph

Connecting to [Microsoft Graph REST APIs](#) from Power Query isn't recommended or supported. See this [article](#) for more information.

Using a gateway with the Web connector

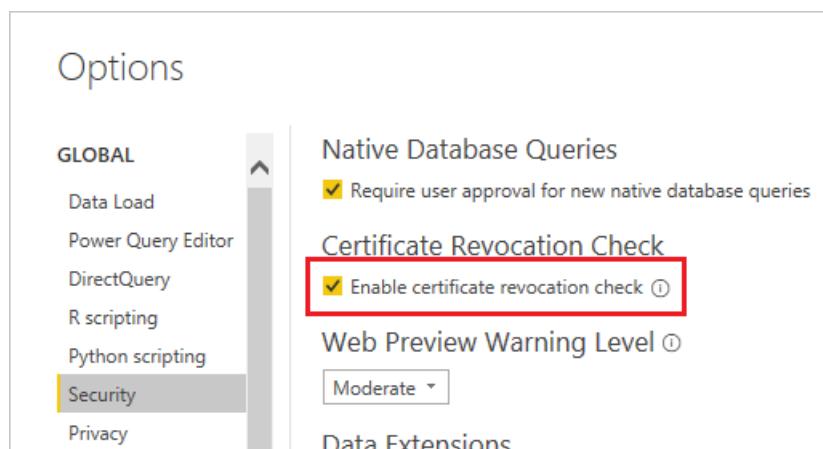
Every use of the Web connector to get data from an HTML page requires the use of an on-premises data gateway when published to a Cloud service, whether it's Power BI Service for datasets or dataflows, or Power Apps for dataflows. (Currently, Dynamics 365 Customer Insights doesn't support the use of a gateway.)

If you receive a `Please specify how to connect` error message when attempting to connect to an HTML page using the Web connector, ensure that you have Internet Explorer 10 or later installed on the machine that hosts your on-premises data gateway.

Capturing web requests and certificate revocation

We've strengthened the security of web connections to protect your data. However, this means that certain scenarios, like capturing web requests with Fiddler, will no longer work by default. To enable those scenarios:

1. Open Power BI Desktop.
2. Under the **File** tab, select **Options and settings > Options**.
3. In **Options**, under **Global > Security**, uncheck **Enable certificate revocation check**.



4. Select **OK**.
5. Restart Power BI Desktop.

IMPORTANT

Be aware that unchecking **Enable certificate revocation check** will make web connections less secure.

To set this scenario in Group Policy, use the "DisableCertificateRevocationCheck" key under the registry path "Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft Power BI Desktop". Setting

"DisableCertificateRevocationCheck" to 0 will always enable the check (stopping Fiddler and similar software from working) and setting "DisableCertificateRevocationCheck" to 1 will always disable the check (enabling Fiddler and similar software).

Changing the authentication method

In some cases, you may need to change the authentication method you use to access a particular site. If this change is necessary, see [Change the authentication method](#).

Limitations on Web connector authentication for HTML content

NOTE

The limitations described in this section only apply to HTML web pages. Opening other kinds of files from the web using this connector isn't affected by these limitations.

The legacy Power Query Web connector automatically creates a `Web.Page` query that supports authentication. The only limitation occurs if you select Windows authentication in the authentication method dialog box. In this case, the **Use my current credentials** selection works correctly, but **Use alternate credentials** won't authenticate.

The new version of the Web connector (currently available in Power BI Desktop) automatically creates a `Web.BrowserContents` query. Such queries currently only support anonymous authentication. In other words, the new Web connector can't be used to connect to a source that requires non-anonymous authentication. This limitation applies to the `Web.BrowserContents` function, regardless of the host environment.

Currently, Power BI Desktop automatically uses the `Web.BrowserContents` function. The `Web.Page` function is still used automatically by Excel and Power Query Online. Power Query Online does support `Web.BrowserContents` using an on-premises data gateway, but you currently would have to enter such a formula manually. When *Web By Example* becomes available in Power Query Online in mid-October 2020, this feature will use `Web.BrowserContents`.

The `Web.Page` function requires that you have Internet Explorer 10 installed on your computer. When refreshing a `Web.Page` query via an on-premises data gateway, the computer containing the gateway must have Internet Explorer 10 installed. If you use only the `Web.BrowserContents` function, you don't need to have Internet Explorer 10 installed on your computer or the computer containing the on-premises data gateway.

In cases where you need to use `Web.Page` instead of `Web.BrowserContents` because of authentication issues, you can still manually use `Web.Page`.

In Power BI Desktop, you can use the older `Web.Page` function by clearing the **New web table inference** preview feature:

- Under the **File** tab, select **Options and settings > Options**.
- In the **Global** section, select **Preview features**.
- Clear the **New web table inference** preview feature, and then select **OK**.
- Restart Power BI Desktop.

NOTE

Currently, you can't turn off the use of `Web.BrowserContents` in Power BI Desktop optimized for Power BI Report Server.

You can also get a copy of a `Web.Page` query from Excel. To copy the code from Excel:

1. Select **From Web** from the Data tab.
2. Enter the address in the **From Web** dialog box, and then select OK.
3. In **Navigator**, choose the data you want to load, and then select **Transform Data**.
4. In the **Home** tab of Power Query, select **Advanced Editor**.
5. In the **Advanced Editor**, copy the M formula.
6. In the app that uses `Web.BrowserContents`, select the **Blank Query** connector.
7. If you're copying to Power BI Desktop:
 - a. In the **Home** tab, select **Advanced Editor**.
 - b. Paste the copied `Web.Page` query in the editor, and then select **Done**.
8. If you're copying to Power Query Online:
 - a. In the **Blank Query**, paste the copied `Web.Page` query in the blank query.
 - b. Select an on-premises data gateway to use.
 - c. Select **Next**.

You can also manually enter the following code into a blank query. Ensure that you enter the address of the web page you want to load.

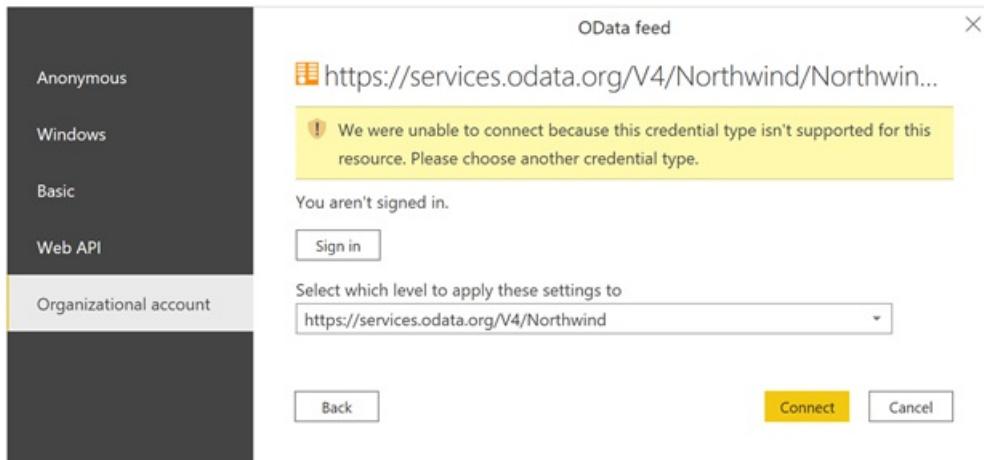
```
let
    Source = Web.Page(Web.Contents("your address here")),
    Navigation = Source{0}[Data]
in
    Navigation
```

Authenticating to arbitrary services

Some services support the ability for the Web connector to authenticate with OAuth/AAD authentication out of the box. However, this won't work in most cases.

When attempting to authenticate, if you see the following error:

"We were unable to connect because this credential type isn't supported for this resource. Please choose another credential type."



Please contact the service owner. They will either need to change the authentication configuration or build a custom connector.

Web connector uses HTTP 1.1 to communicate

The Power Query Web connector communicates with a data source using HTTP 1.1. If your data source is expecting to communicate using HTTP 1.0, you might receive an error, such as `500 Internal Server Error`.

It's not possible to switch Power Query to use HTTP 1.0. Power Query always sends an `Expect:100-continue` when there's a body to avoid passing a possibly large payload when the initial call itself might fail (for example, due to a lack of permissions). Currently, this behavior can't be changed.

See also

- [Power Query Web connector](#)
- [Troubleshooting the Power Query Web connector](#)

Workforce Dimensions (Beta) (Deprecated)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Deprecated
Products	-
Authentication Types Supported	-
Function Reference Documentation	-

Deprecation

NOTE

This connector is deprecated because of end of support for the connector. We recommend that users transition off existing connections using this connector, and don't use this connector for new connections.

XML

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights Analysis Services
Function Reference Documentation	Xml.Tables Xml.Document

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

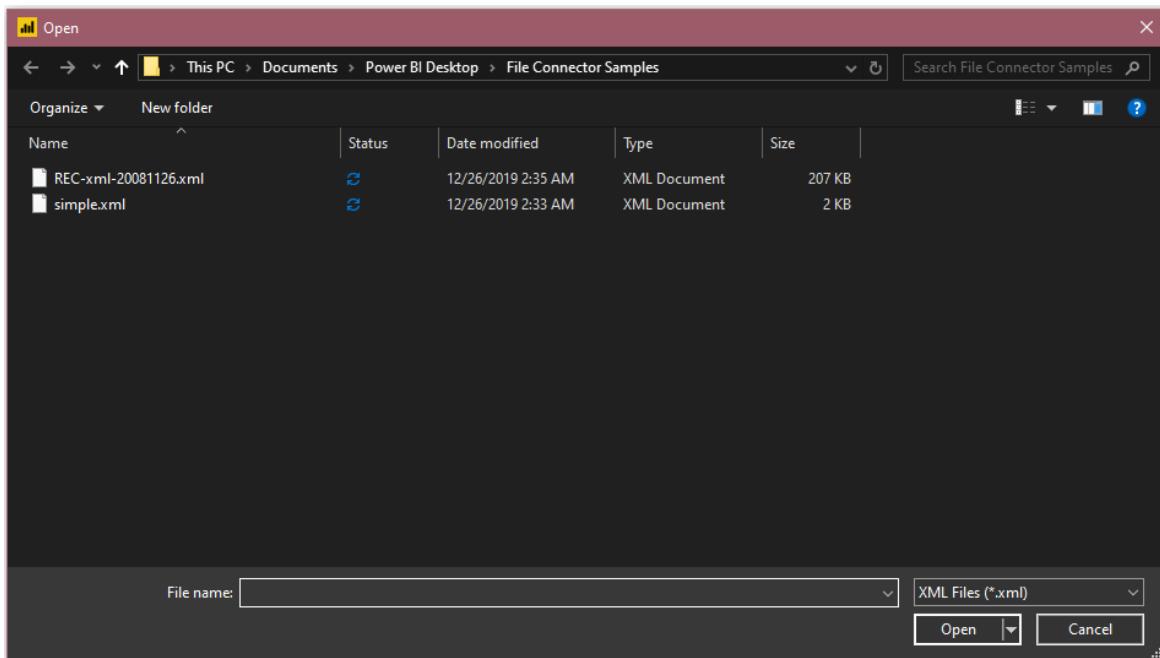
Capabilities supported

- Import

Load a local XML file from Power Query Desktop

To load a local XML file:

1. Select the **XML** option in the **Get Data** selection. This action will launch a local file browser and allow you to select your XML file.



2. Browse to the directory containing the local XML file you want to load, and then select **Open**.
3. In **Navigator**, select the data you want, and then either select **Load** to load the data, or **Transform Data** to continue transforming the data in Power Query Editor.

A screenshot of the Power BI Navigator interface. The title bar says "Navigator". On the left, there is a tree view under "Display Options" with a search icon. It shows a folder named "simple.xml [1]" which is expanded to reveal a table named "food". The "food" table has three columns: "name", "price", and "description". The data in the table is as follows:

name	price	description
Belgian Waffles	5.95	Two of our famous Belgian Waffles with plenty of i
Strawberry Belgian Waffles	7.95	Light Belgian waffles covered with strawberries an
Berry-Berry Belgian Waffles	8.95	Light Belgian waffles covered with an assortment c
French Toast	4.5	Thick slices made from our homemade sourdough
Homestyle Breakfast	6.95	Two eggs, bacon or sausage, toast, and our ever-pe

At the bottom of the Navigator window are buttons for "Load", "Transform Data", and "Cancel".

Load a local XML file from Power Query Online

To load a local XML file:

1. From the **Data sources** page, select **XML**.
2. Enter the path to the local XML file.

Connection settings

File path or URL *
C:\test-examples\XML\books.xml [Browse OneDrive...](#)

Connection credentials

Data gateway *
[On-premises][Admin] TestGateway23 [Edit](#)

Authentication kind
Windows

Username
domain\alias

Password

3. Select an on-premises data gateway from **Data gateway**.

4. If authentication is required, enter your credentials.

5. Select **Next**.

Loading the XML file will automatically launch the Power Query Editor. From the editor, you can then transform the data if you want, or you can just save and close to load the data.

Power Query - Edit queries

Home Transform Add column View Help

Get data Enter data Options Manage parameters Refresh Properties Advanced editor Manage Query Choose columns Remove columns Keep rows Remove rows Filter rows Reduce rows Sort Transform Combine Map to entity CDM AI insights Insights

Queries [1]

Query

Completed (1.64 s) Columns: 4 Rows: 1

Query settings

Name: Query
Entity type: Custom
Applied steps: Source

Cancel **Save & close**

Load an XML file from the web

If you want to load an XML file from the web, instead of selecting the XML connector you can select the Web connector. Paste in the address of the desired file and you'll be prompted with an authentication selection, since you're accessing a website instead of a static file. If there's no authentication, you can just select **Anonymous**. As in the local case, you'll then be presented with the table that the connector loads by default, which you can Load or Transform.

Troubleshooting

Data Structure

Due to the fact that many XML documents have ragged or nested data, you may have to do extra data shaping

to get it in the sort of form that will make it convenient to do analytics. This holds true whether you use the UI accessible `Xm1.Tables` function, or the `Xm1.Document` function. Depending on your needs, you may find you have to do more or less data shaping.

Text versus nodes

If your document contains a mixture of text and non-text sibling nodes, you may encounter issues.

For example if you have a node like this:

```
<abc>
  Hello <i>world</i>
</abc>
```

`Xm1.Tables` will return the "world" portion but ignore "Hello". Only the element(s) are returned, not the text. However, `Xm1.Document` will return "Hello <i>world</i>". The entire inner node is turned to text, and structure isn't preserved.

Zendesk (Beta)

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

ITEM	DESCRIPTION
Release State	Beta
Products	Power BI (Datasets)
Authentication Types Supported	Zendesk account

Prerequisites

Before you can sign in to Zendesk, you must have a Zendesk account (username/password).

Capabilities Supported

- Import

Connect to Zendesk data

To connect to Zendesk data:

1. Select **Get Data** from the **Home** ribbon in Power BI Desktop. Select **Online Services** from the categories on the left, select **Zendesk (Beta)**, and then select **Connect**.

Get Data

- All
- File
- Database
- Power Platform
- Azure
- Online Services
- Other

Online Services

- Mixpanel (Beta)
- Planview Enterprise One - PRM (Beta)
- QuickBooks Online (Beta)
- Smartsheet
- SparkPost (Beta)
- SweetIQ (Beta)
- Planview Enterprise One - CTM (Beta)
- Twilio (Beta)
- Zendesk (Beta)
- Asana (Beta)
- Assemble Views (Beta)
- Automation Anywhere
- Dynamics 365 Customer Insights (Beta)
- Emigo Data Source
- Entersoft Business Suite (Beta)
- eWay-CRM

[Certified Connectors](#) | [Template Apps](#)

[Connect](#) [Cancel](#)

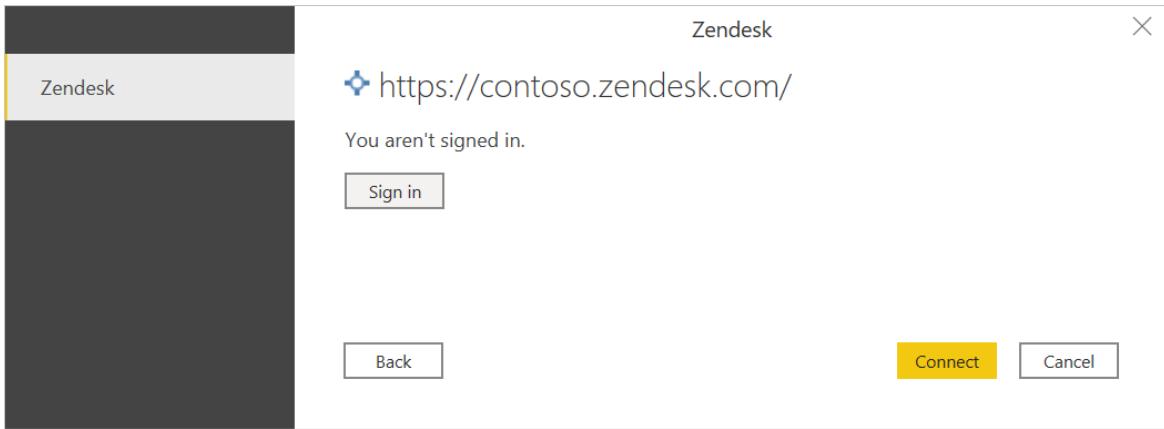
2. If this is the first time you're getting data through the Zendesk connector, a preview connector notice will be displayed. Select **Don't warn me again with this connector** if you don't want this message to be displayed again, and then select **Continue**.
3. Enter the Zendesk URL location that you want to access, and then select **OK**.

Zendesk

Zendesk URL

[OK](#) [Cancel](#)

4. To sign in to your Zendesk account, select **Sign in**.



5. In the Zendesk window that appears, provide your credentials to sign in to your Zendesk account.

The image shows a "Sign in to Contoso" form. It has two input fields: "Email" and "Password", both with placeholder text. Below the fields is a large blue "Sign in" button. Underneath the button are links for "I am a Customer" and "Forgot my password". At the bottom is a link for "Privacy Policy".

Sign in to Contoso

Email

Password

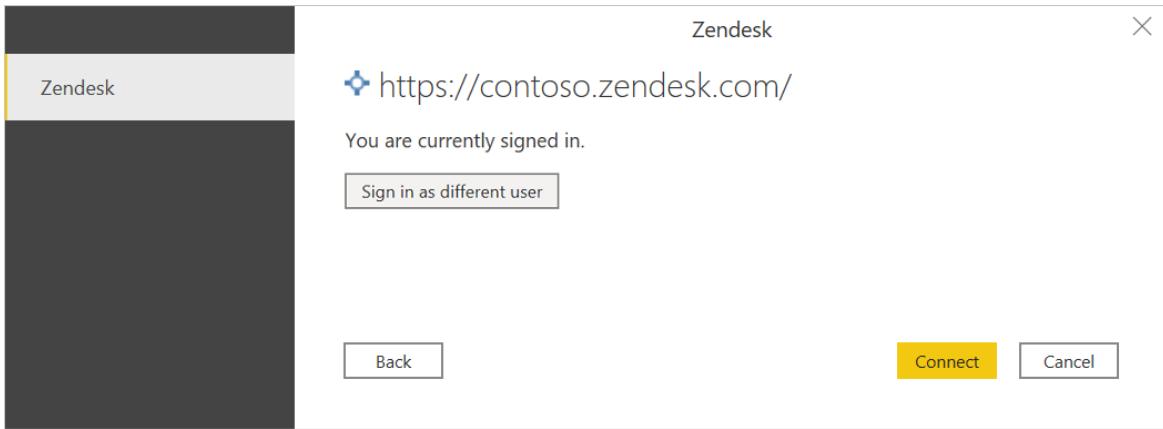
Sign in

I am a Customer

Forgot my password

Privacy Policy

6. Select **Sign in**.
7. Once you've successfully signed in, select **Connect**.



8. In **Navigator**, select the information you want, then either select **Load** to load the data or **Transform Data** to continue transforming the data in Power Query Editor.

Limitations and issues

You should be aware of the following limitations and issues associated with accessing Zendesk data.

- Zendesk returns a 422 error status if the instance returns more than 1000 rows.

Power Query Online Limits

1/15/2022 • 2 minutes to read • [Edit Online](#)

Summary

Power Query Online is integrated into a variety of Microsoft products. Since these products target different scenarios, they may set different limits for Power Query Online usage.

Limits are enforced at the beginning of query evaluations. Once an evaluation is underway, only timeout limits are imposed.

Limit Types

Hourly Evaluation Count: The maximum number of evaluation requests a user can issue during any 60 minute period

Daily Evaluation Time: The net time a user can spend evaluating queries during any 24 hour period

Concurrent Evaluations: The maximum number of evaluations a user can have running at any given time

Authoring Limits

Authoring limits are the same across all products. During authoring, query evaluations return previews that may be subsets of the data. Data is not persisted.

Hourly Evaluation Count: 1000

Daily Evaluation Time: Currently unrestricted

Per Query Timeout: 10 minutes

Refresh Limits

During refresh (either scheduled or on-demand), query evaluations return complete results. Data is typically persisted in storage.

PRODUCT INTEGRATION	HOURLY EVALUATION COUNT (#)	DAILY EVALUATION TIME (HOURS)	CONCURRENT EVALUATIONS (#)
Microsoft Flow (SQL Connector—Transform Data Using Power Query)	500	2	5
Dataflows in PowerApps.com (Trial)	500	2	5
Dataflows in PowerApps.com (Production)	1000	8	10
Data Integration in PowerApps.com Admin Portal	1000	24	10
Dataflows in PowerBI.com	1000	100	10

Common Issues

1/15/2022 • 6 minutes to read • [Edit Online](#)

Preserving sort

You might assume that if you sort your data, any downstream operations will preserve the sort order.

For example, if you sort a sales table so that each store's largest sale is shown first, you might expect that doing a "Remove duplicates" operation will return only the top sale for each store. And this operation might, in fact, appear to work. However, this behavior isn't guaranteed.

Because of the way Power Query optimizes certain operations, including skipping them or offloading them to data sources (which can have their own unique ordering behavior), sort order isn't guaranteed to be preserved through aggregations (such as `Table.Group`), merges (such as `Table.NestedJoin`), or duplicate removal (such as `Table.Distinct`).

There are a number of ways to work around this. Here are two suggestions:

- Perform a sort *after* applying the downstream operation. For example, when grouping rows, sort the nested table in each group before applying further steps. Here's some sample M code that demonstrates this approach:

```
Table.Group(Sales_SalesPerson, {"TerritoryID"}, {{"SortedRows", each Table.Sort(_, {"SalesYTD", Order.Descending})}})
```
- Buffer the data (using `Table.Buffer`) before applying the downstream operation. In some cases, this operation will cause the downstream operation to preserve the buffered sort order.

Data type inference

Sometimes Power Query may incorrectly detect a column's data type. This is due to the fact that Power Query infers data types using only the first 200 rows of data. If the data in the first 200 rows is somehow different than the data after row 200, Power Query can end up picking the wrong type. (Be aware that an incorrect type won't always produce errors. Sometimes the resulting values will simply be incorrect, making the issue harder to detect.)

For example, imagine a column that contains integers in the first 200 rows (such as all zeroes), but contains decimal numbers after row 200. In this case, Power Query will infer the data type of the column to be Whole Number (`Int64.Type`). This inference will result in the decimal portions of any non-integer numbers being truncated.

Or imagine a column that contains textual date values in the first 200 rows, and other kinds of text values after row 200. In this case, Power Query will infer the data type of the column to be Date. This inference will result in the non-date text values being treated as type conversion errors.

Because type detection works on the first 200 rows, but Data Profiling can operate over the entire dataset, you can consider using the Data Profiling functionality to get an early indication in the Query Editor about Errors (from type detection or any number of other reasons) beyond the top N rows.

Connections forcibly closed by the remote host

When connecting to various APIs, you might get the following warning:

```
Data source error: Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host
```

If you run into this error, it's most likely a networking issue. Generally, the first people to check with are the owners of the data source you're attempting to connect to. If they don't think they're the ones closing the connection, then it's possible something along the way is (for example, a proxy server, intermediate routers/gateways, and so on).

Whether this only reproduces with any data or only larger data sizes, it's likely that there's a network timeout somewhere on the route. If it's only with larger data, customers should consult with the data source owner to see if their APIs support paging, so that they can split their requests into smaller chunks. Failing that, alternative ways to extract data from the API (following data source best practices) should be followed.

TLS RSA cipher suites are deprecated

Effective October 30, 2020, the following cipher suites are being deprecated from our servers.

- "TLS_RSA_WITH_AES_256_GCM_SHA384"
- "TLS_RSA_WITH_AES_128_GCM_SHA256"
- "TLS_RSA_WITH_AES_256_CBC_SHA256"
- "TLS_RSA_WITH_AES_128_CBC_SHA256"

The following list are the supported cipher suites:

- "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256"
- "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"
- "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
- "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
- "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
- "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384"
- "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256"
- "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"

Cipher suites are used to encrypt messages to secure a network connection between clients/servers and other servers. We're removing the above list of cipher suites to comply with our current security protocols. Beginning March 1, 2021, customers can only use our [standard cipher suites](#).

These are the cipher suites the server you connect to must support to connect from Power Query Online or Power BI.

In Power Query Desktop (Power BI, Excel), we don't control your cipher suites. If you're trying to connect to Power Platform (for example Power Platform Dataflows) or the Power BI Service, you'll need one of those cipher suites enabled on your OS. You may either upgrade the [Windows version](#) or update the [Windows TLS registry](#) to make sure that your server endpoint supports one of these ciphers.

To verify that your server complies with the security protocol, you can perform a test using a TLS cipher and scanner tool. One example might be [SSLLABS](#).

Customers must upgrade their servers before March 1, 2021. For more information about configuring TLS Cipher Suite order, see [Manage Transport Layer Security \(TLS\)](#).

Certificate revocation

An upcoming version of Power BI Desktop will cause SSL connections failure from Desktop when any certificates in the SSL chain are missing certificate revocation status. This is a change from the current state, where revocation only caused connection failure in the case where the certificate was explicitly revoked. Other certificate issues might include invalid signatures, and certificate expiration.

As there are configurations in which revocation status may be stripped, such as with corporate proxy servers, we'll be providing another option to ignore certificates that don't have revocation information. This option will allow situations where revocation information is stripped in certain cases, but you don't want to lower security entirely, to continue working.

It isn't recommended, but users will continue to be able to turn off revocation checks entirely.

Error: Evaluation was canceled

Power Query will return the message "Evaluation was canceled" when background analysis is disabled and the user switches between queries or closes the Query Editor while a query is in the process of refreshing.

Error: The key didn't match any rows in the table

There are many reasons why Power Query may return an error that **the key didn't match any rows in the table**. When this error happens, the Mashup Engine is unable to find the table name it's searching for. Reasons why this error may happen include:

- The table name has been changed, for example in the data source itself.
- The account used to access the table doesn't have sufficient privileges to read the table.
- There may be multiple credentials for a single data source, which [isn't supported in Power BI Service](#). This error may happen, for example, when the data source is a cloud data source and multiple accounts are being used to access the data source at the same time with different credentials. If the data source is on-premises, you'll need to use the on-premises data gateway.

Limitation: Domain-joined requirement for gateway machines when using Windows authentication

Using Windows authentication with an on-premises gateway requires the gateway machine to be domain joined. This applies to any connections that are set up with "Windows authentication through the gateway". Windows accounts that will be used to access a data source might require read access to the shared components in the Windows directory and the gateway installation.

Limitation: Cross tenant OAuth2 refresh isn't supported in Power BI service

If you want to connect to a data source from Power BI service using OAuth2, the data source must be in the same tenant as Power BI service. Currently, multi-tenant connection scenarios aren't supported with OAuth2.

Review script changes in Power Query Online

1/15/2022 • 2 minutes to read • [Edit Online](#)

Background

Due to the way that queries are stored in Power Query Online, there are cases where manually entered M script (generally comments) is lost. The 'Review Script Changes' pane provides a diff experience highlighting the changes, which allows users to understand what changes are being made. Users can then accept the changes or rearrange their script to fix it.

There are three notable cases that may cause this experience.

Script for ribbon transforms

Ribbon transforms always generate the same M script, which may be different than the way they are manually entered. This should always be equivalent script. Contact support if this is not the case.

Comments

Comments always have to be inside the `Let ... in` expression, and above a step. This will be shown in the user interface as a 'Step property'. We lose all other comments. Comments that are written on the same line as one step, but above another step (for example, after the comma that trails every step) will be moved down.

Removing script errors

In certain cases, your script will be updated if it results in a syntax error by escaping your script (for example, when using the formula bar).

Experience

When you commit a query, Power Query Online will evaluate it to see if the 'stored' version of the script differs at all from what you have submitted. If it does, it will present you with a 'Review Script Changes' dialog box that will allow you to accept or cancel.

- If you accept, the changes will be made to your query.
- If you cancel, you might rewrite your query to make sure that you move your comments properly, or rearrange however else you want.

Review script changes

The original script needs to be modified due to limitations in the supported script patterns. To learn more about script restrictions and how to avoid them, [click here](#).

Original script

```
1-//This will be lost
2-let //This will be kept and moved down a line.
3--//This will be kept. This is a great place to write out long comments,
4--such as an overall explanation of what's happening in your query.
5- Source = 1, //This will be kept and moved down a line
6- Calc = Source+1//This will be lost
7--//This will be lost
8-in //This will be lost
9---//This will be lost
10- Calc //This will be lost
11--//This will be lost
```

Modified script

```
1+let
2+ // This will be kept and moved down a line.
3+ // This will be kept. This is a great place to write out long comments,
4+ // such as an overall explanation of what's happening in your query.
5+ Source = 1,
6+ // This will be kept and moved down a line
7+ Calc = Source + 1
8+in
9+Calc
```

Accept

Cancel

Power Query connector feedback

1/15/2022 • 2 minutes to read • [Edit Online](#)

This article describes how to submit feedback for Power Query connectors. It's important to distinguish between Microsoft-owned connectors and non-Microsoft-owned connectors, as the support and feedback channels are different.

To confirm whether a connector is Microsoft-owned, visit the [connector reference](#). Only connectors marked as "By Microsoft" are Microsoft-owned connectors.



Microsoft-owned connectors

This section outlines instructions to receive support or submit feedback on Microsoft-owned connectors.

Support and troubleshooting

If you're finding an issue with a Power Query connector, use the dedicated support channels for the product you're using Power Query connectors in. For example, for Power BI, visit the [Power BI support page](#).

If you're seeking help with using Microsoft-owned Power Query connectors, visit one of the following resources.

- [Power Query on Microsoft Q&A](#)
- Community forums for the product you're using Power Query in. For example, for Power BI, this forum would be the [Power BI Community](#)
- [Power Query website resources](#)

Submitting feedback

To submit feedback about a Microsoft-owned connector, provide the feedback to the "ideas" forum for the product you're using Power Query connectors in. For example, for Power BI, visit the [Power BI ideas forum](#). If you have one, you can also provide feedback directly to your Microsoft account contact.

Non-Microsoft-owned connectors

This section outlines instructions to receive support or submit feedback on non-Microsoft-owned connectors.

Support and troubleshooting

For non-Microsoft-owned connectors, support and troubleshooting questions should go to the connector owner through their support channels. For example, for a Contoso-owned connector, you should submit a request through the Contoso support channels.

You can also engage the Power Query community resources indicated above for Microsoft-owned connectors, in case a member of the community can assist.

Submitting feedback

As non-Microsoft-owned connectors are managed and updated by the respective connector owner, feedback should be sent directly to the connector owner. For example, to submit feedback about a Contoso-owned connector, you should directly submit feedback to Contoso.

Capture web requests with Fiddler

1/15/2022 • 2 minutes to read • [Edit Online](#)

When diagnosing issues that might occur when Power Query communicates with your data, you might be asked to supply a Fiddler trace. The information provided by Fiddler can be of significant use when troubleshooting connectivity issues.

NOTE

This article assumes that you are already familiar with how Fiddler works in general.

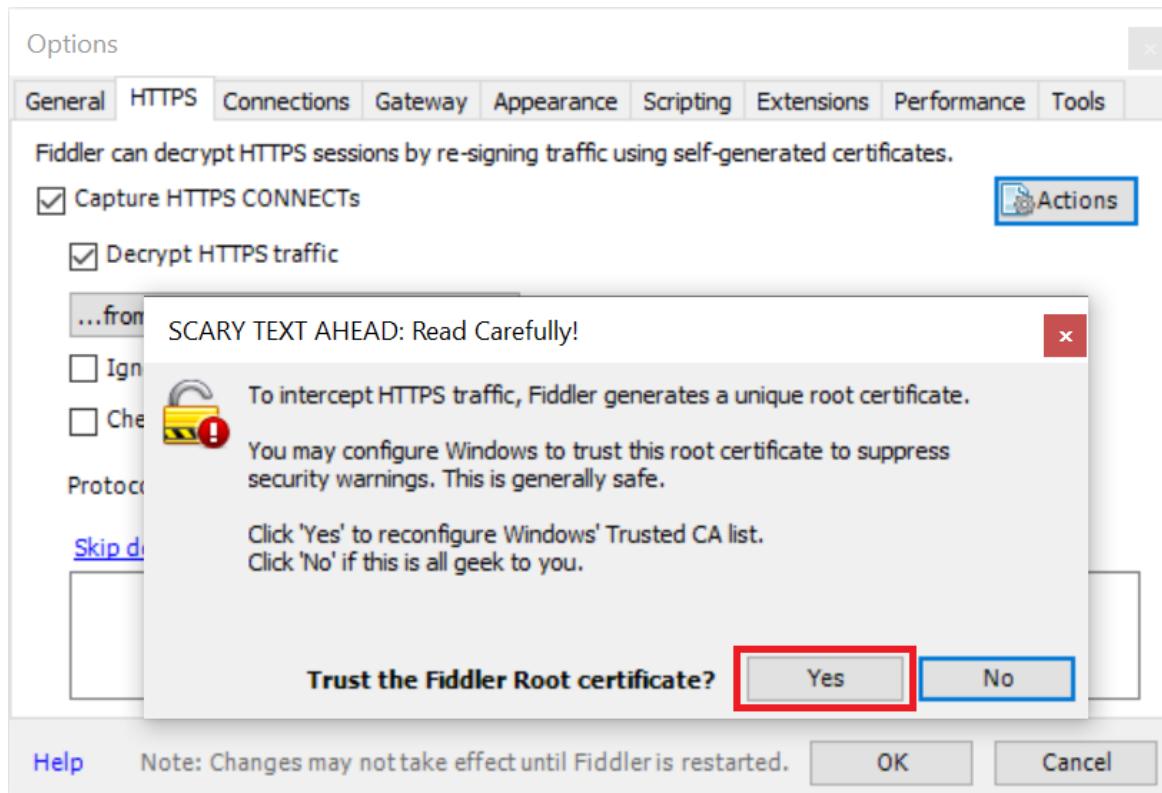
Set up Fiddler to capture secure HTTP addresses

Before you can begin capturing web requests issued by Power Query, you must first enable Fiddler to capture secure HTTP addresses (`https://`) on your device.

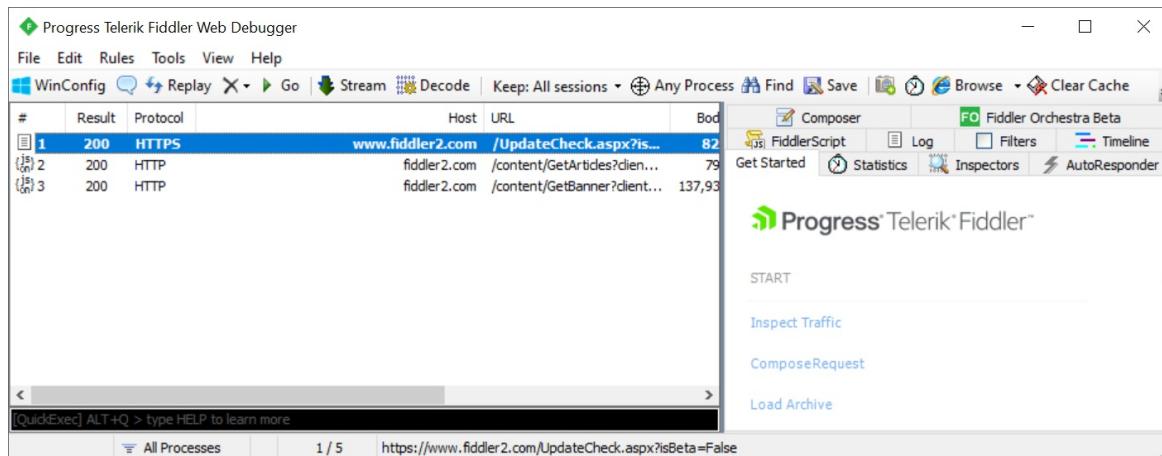
If you don't already have Fiddler installed, download and install [Fiddler](#) now. Be sure to install Fiddler on the system where the issue is occurring.

To set up Fiddler to capture secure HTTP addresses:

1. Open Fiddler.
2. Under **File**, clear the check mark next to **Capture Traffic**.
3. Select **Tools > Options**.
4. Open the **HTTPS** tab.
5. Select **Capture HTTPS CONNECTs**.
6. Select **Decrypt HTTPS traffic**.
7. In the root certificate dialog box, select **Yes**.



8. When asked to confirm that you want to add the certificate to your PCs Trusted Root List, select Yes.
9. In the HTTPS tab, select OK.
10. In the Fiddler traffic pane, select one of the current traces, and then press **Ctrl + X**. This action clears all of the current traces from the traffic pane.



Guidelines for capturing web requests

Because Fiddler captures all network traffic during the recorded session, be sure to close all other apps and web pages before capturing a fiddler trace. Closing all other apps and web pages clears away most extra web traffic that's not associated with the issue you're trying to capture.

Once you've closed all other apps and web pages not associated with the issue, clear the Fiddler traffic pane as described in step 10 in the previous procedure. Then select **File > Capture traffic** just before starting the operation in question. After the issue occurs, immediately clear the check mark next to **File > Capture traffic** to stop the capture.

These actions minimize the number of messages we have to dig through, and also helps focus the investigation. It also avoids capturing other potentially sensitive information that you don't want to share.

If you're only running Power Query and Fiddler, this minimum setup should yield a sequence of HTTP requests

and responses from whatever backend you're communicating with, for example Power BI service, SharePoint, or Azure. The requests, responses, headers, response codes, and sometimes the payload will all provide clues we can use to troubleshoot your issue.

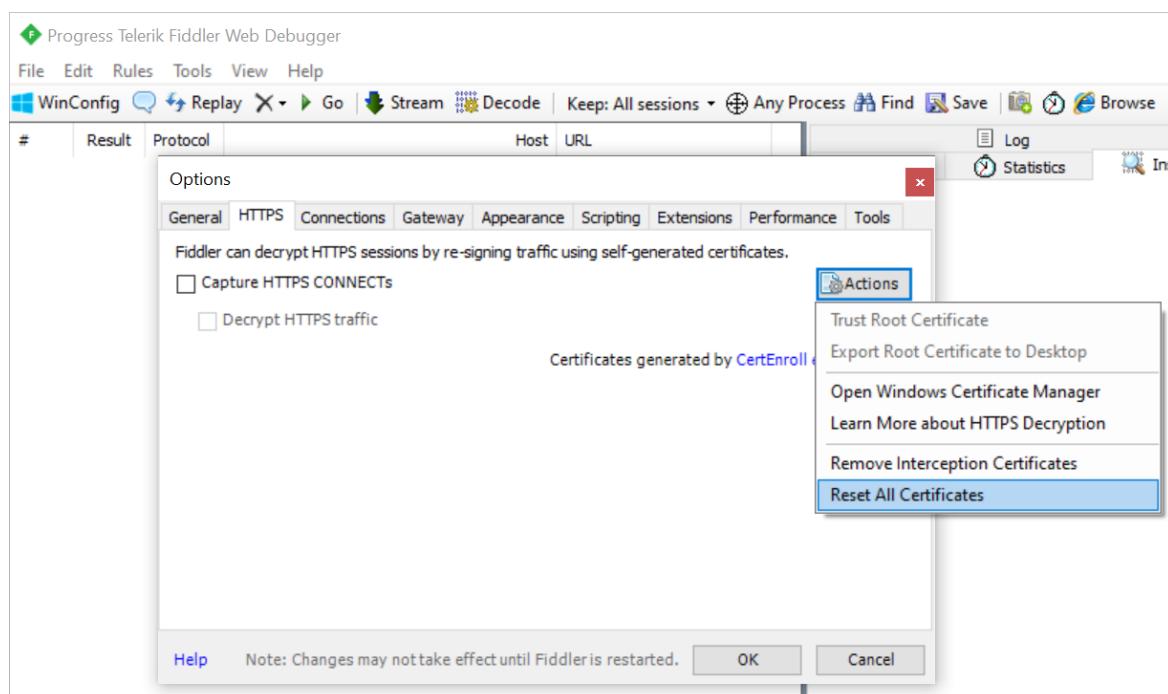
To save the capture session to a log file, select **File > Save > All Sessions**. You might also be asked to compress the log file (.zip) before sending it.

Return your system to its original configuration

Once you've finished providing Fiddler log files and troubleshooting has completed, you'll want to return your system to its original configuration and remove Fiddler as the middleman.

To return your system to its original configuration:

1. In Fiddler, select **Tools > Options**.
2. In **Options**, open the **HTTPS** tab.
3. Clear the check mark next to **Capture HTTPS CONNECTS**.
4. Select **Actions**.
5. Select **Reset All Certificates**.



6. In **Reset All Certificates**, select **OK**.
7. In **Do you want to allow this app to make changes to your device?**, select **Yes**.
8. In **TrustCert Success**, select **OK**.
9. In **Root Certificate Store**, select **Yes**.
10. If the root certificate dialog box appears, close the dialog box without selecting **Yes** or **No**.
11. In **Success**, select **OK**.
12. In **Options**, select **OK**.

See also

- [Query diagnostics](#)
- [Power Query feedback](#)
- [Getting started with Fiddler Classic](#)

Installing the Power Query SDK

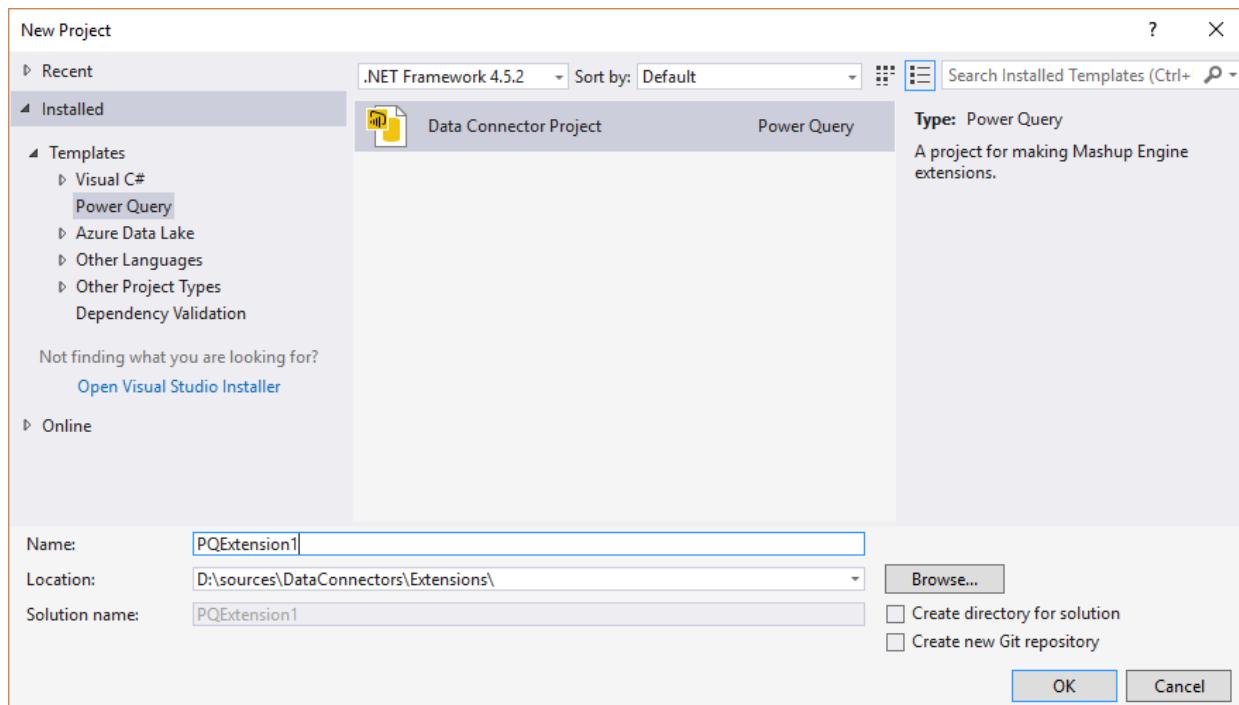
1/15/2022 • 2 minutes to read • [Edit Online](#)

Quickstart

NOTE

The steps to enable extensions changed in the June 2017 version of Power BI Desktop.

1. Install the [Power Query SDK](#) from the Visual Studio Marketplace.
2. Create a new data connector project.
3. Define your connector logic.
4. Build the project to produce an extension file.
5. Copy the extension file into [Documents]/Power BI Desktop/Custom Connectors.
6. Check the option (**Not Recommended**) **Allow any extension to load without validation or warning** in Power BI Desktop (under *File / Options and settings / Options / Security / Data Extensions*).
7. Restart Power BI Desktop.



Distribution of Data Connectors

Power BI Desktop users can download extension files and place them in a known directory (steps described above). Power BI Desktop will automatically load the extensions on restart.

Additional links and resources

- [M Library Functions](#)
- [M Language Specification](#)
- [Power BI Developer Center](#)
- [Data Connector Tutorial](#)

Step by step

Creating a new extension in Visual Studio

Installing the Power Query SDK for Visual Studio will create a new Data Connector project template in Visual Studio.

This creates a new project containing the following files:

- Connector definition file (<connectorName>.pq)
- A query test file (<connectorName>.query.pq)
- A string resource file (resources.resx)
- PNG files of various sizes used to create icons

Your connector definition file will start with an empty Data Source description. See the Data Source Kind section later in this document for details.

Testing in Visual Studio

The Power Query SDK provides basic query execution capabilities, allowing you to test your extension without having to switch over to Power BI Desktop. See [Query File](#) for more details.

Build and deploy from Visual Studio

Building your project will produce your .pqx file.

Data Connector projects don't support custom post build steps to copy the extension file to your [Documents]\Microsoft Power BI Desktop\Custom Connectors directory. If this is something you want to do, you may want to use a third party Visual Studio extension, such as Auto Deploy.

Extension files

Power Query extensions are bundled in a ZIP file and given a .mez file extension. At runtime, Power BI Desktop will load extensions from the [Documents]\Microsoft Power BI Desktop\Custom Connectors.

NOTE

In an upcoming change the default extension will be changed from .mez to .pqx.

Extension file format

Extensions are defined within an M section document. A section document has a slightly different format from the query document(s) generated in Power Query. Code you import from Power Query typically requires modification to fit into a section document, but the changes are minor. Section document differences you should be aware of include:

- They begin with a section declaration (for example, `section HelloWorld;`).
- Each expression ends with a semi-colon (for example, `a = 1;` or `b = let c = 1 + 2 in c;`).
- All functions and variables are local to the section document, unless they are marked as shared. Shared functions become visible to other queries/functions, and can be thought of as the exports for your extension (that is, they become callable from Power Query).

More information about M section documents can be found in the M Language specification.

Query File

In addition to the extension file, Data Connector projects can have a query file (name.query.pq). This file can be used to run test queries within Visual Studio. The query evaluation will automatically include your extension code, without having to register your .pqx file, allowing you to call/test any shared functions in your extension code.

The query file can contain a single expression (for example, `HelloWorld.Contents()`), a `let` expression (such as what Power Query would generate), or a section document.

Starting to develop custom connectors

1/15/2022 • 2 minutes to read • [Edit Online](#)

To get you up to speed with Power Query, this page lists some of the most common questions.

What software do I need to get started with the Power Query SDK?

You need to install the [Power Query SDK](#) in addition to Visual Studio. To be able to test your connectors, you should also have Power BI installed.

What can you do with a custom connector?

Custom connectors allow you to create new data sources or customize and extend an existing source. Common use cases include:

- Creating a business analyst-friendly view for a REST API.
- Providing branding for a source that Power Query supports with an existing connector (such as an OData service or ODBC driver).
- Implementing OAuth v2 authentication flow for a SaaS offering.
- Exposing a limited or filtered view over your data source to improve usability.
- Enabling DirectQuery for a data source using an ODBC driver.

Custom connectors are only available in Power BI Desktop and Power BI Service through the use of an on-premises data gateway. More information: [TripPin 9 - Test Connection](#)

Creating your first connector: Hello World

1/15/2022 • 2 minutes to read • [Edit Online](#)

Hello World sample

This sample provides a simple data source extension that can be run in Visual Studio, and loaded in Power BI Desktop. As an overview, this sample shows the following:

- Exporting function (`HelloWorld.Contents`), which takes an option text parameter.
- Defining a data source kind that:
 - Declares that it uses Implicit (anonymous) authentication.
 - Uses string resources that allow for localization.
 - Declaring UI metadata so the extension can show up in the Power BI Desktop Get Data dialog.

Following the instructions in [Installing the PowerQuery SDK](#), create a new project called "HelloWorld" and copy in the following M code, and then follow the rest of the instructions to be able to open it in PowerBI.

In the following connector definition you'll find:

- A `section` statement.
- A data source function with metadata establishing it as a data source definition with the Kind `HelloWorld` and Publish `HelloWorld.Publish`.
- An `Authentication` record declaring that implicit (anonymous) is the only authentication type for this source.
- A publish record declaring that this connection is in Beta, what text to load from the resx file, the source image, and the source type image.
- A record associating icon sizes with specific PNGs in the build folder.

```

[DataSource.Kind="HelloWorld", Publish="HelloWorld.Publish"]
shared HelloWorld.Contents = (optional message as text) =>
    let
        message = if (message <> null) then message else "Hello world"
    in
        message;

HelloWorld = [
    Authentication = [
        Implicit = []
    ],
    Label = Extension.LoadString("DataSourceLabel")
];

HelloWorld.Publish = [
    Beta = true,
    ButtonText = { Extension.LoadString("FormulaTitle"), Extension.LoadString("FormulaHelp") },
    SourceImage = HelloWorld.Icons,
    SourceTypeImage = HelloWorld.Icons
];

HelloWorld.Icons = [
    Icon16 = { Extension.Contents("HelloWorld16.png"), Extension.Contents("HelloWorld20.png"), Extension.Contents("HelloWorld24.png"), Extension.Contents("HelloWorld32.png") },
    Icon32 = { Extension.Contents("HelloWorld32.png"), Extension.Contents("HelloWorld40.png"), Extension.Contents("HelloWorld48.png"), Extension.Contents("HelloWorld64.png") }
];

```

Once you've built the file and copied it to the correct directory, following the instructions in [Installing the PowerQuery SDK](#) tutorial, open PowerBI. You can search for "hello" to find your connector in the **Get Data** dialog.

This step will bring up an authentication dialog. Since there's no authentication options and the function takes no parameters, there's no further steps in these dialogs.

Press **Connect** and the dialog will tell you that it's a "Preview connector", since `Beta` is set to true in the query. Since there's no authentication, the authentication screen will present a tab for Anonymous authentication with no fields. Press **Connect** again to finish.

Finally, the query editor will come up showing what you expect—a function that returns the text "Hello world".

For the fully implemented sample, see the [Hello World Sample](#) in the Data Connectors sample repo.

TripPin Tutorial

1/15/2022 • 2 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

This tutorial uses a public OData service ([TripPin](#)) as a reference source. Although this lesson requires the use of the M engine's OData functions, subsequent lessons will use [Web.Contents](#), making it applicable to (most) REST APIs.

Prerequisites

The following applications will be used throughout this tutorial:

- [Power BI Desktop](#), May 2017 release or later
- [Power Query SDK for Visual Studio](#)
- [Fiddler](#)—Optional, but recommended for viewing and debugging requests to your REST service

It's strongly suggested that you review:

- [Installing the PowerQuery SDK](#)
- [Start developing custom connectors](#)
- [Creating your first connector: Hello World](#)
- [Handling Data Access](#)
- [Handling Authentication](#)

NOTE

You can also start trace logging of your work at any time by enabling diagnostics, which is described later on in this tutorial. More information: [Enabling diagnostics](#)

Parts

PART	LESSON	DETAILS
1	OData	Create a simple Data Connector over an OData service
2	Rest	Connect to a REST API that returns a JSON response
3	Nav Tables	Provide a navigation experience for your source
4	Data Source Paths	How credentials are identified for your data source
5	Paging	Read with a paged response from a web service

PART	LESSON	DETAILS
6	Enforcing Schema	Enforce table structure and column data types
7	Advanced Schema	Dynamically enforce table structure using M types and external metadata
8	Diagnostics	Add detailed tracing to the connector
9	Test Connection	Implement a TestConnection handler to enable refresh through the gateway
10	Basic query Folding	Implement basic query folding handlers

TripPin Part 1 - Data Connector for an OData Service

1/15/2022 • 5 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

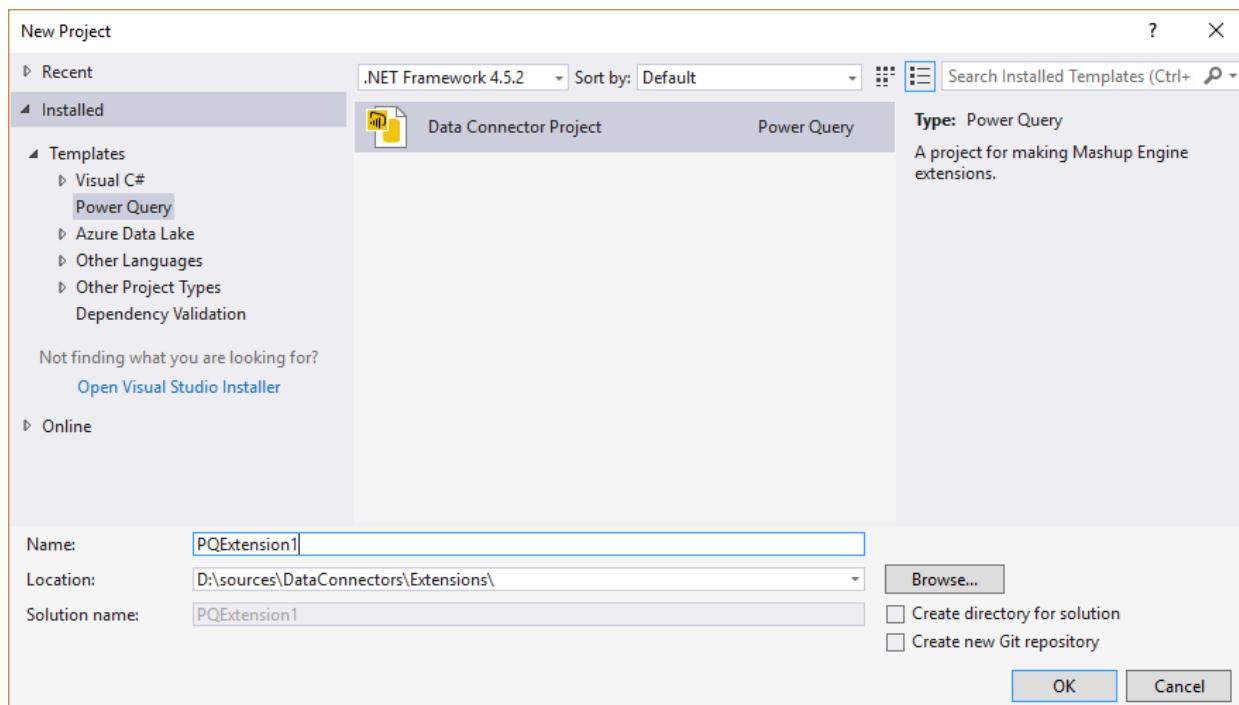
In this lesson, you will:

- Create a new Data Connector project using the Visual Studio SDK
- Author a base function to pull data from a source
- Test your connector in Visual Studio
- Register your connector in Power BI Desktop

Creating a Basic OData Connector

In this section, you will create a new Data Connector project, provide some basic information, and test it in Visual Studio.

Open Visual Studio, and create a new Project. Under the Power Query folder, select the Data Connector project. For this sample, set the project name to `TripPin`.



Open the TripPin.pq file and paste in the following connector definition.

```

section TripPin;

[DataSource.Kind="TripPin", Publish="TripPin.Publish"]
shared TripPin.Feed = Value.ReplaceType(TripPinImpl, type function (url as Uri.Type) as any);

TripPinImpl = (url as text) =>
let
    source = OData.Feed(url)
in
    source;

// Data Source Kind description
TripPin = [
    Authentication = [
        Anonymous = []
    ],
    Label = "TripPin Part 1 - OData"
];

// Data Source UI publishing description
TripPin.Publish = [
    Beta = true,
    Category = "Other",
    ButtonText = { "TripPin OData", "TripPin OData" }
];

```

This connector definition contains:

- A Data Source definition record for the TripPin connector
- A declaration that Implicit (Anonymous) is the only authentication type for this source
- A function (`TripPinImpl`) with an implementation that calls `OData.Feed`
- A shared function (`TripPin.Feed`) that sets the parameter type to `Uri.Type`
- A Data Source publishing record that will allow the connector to appear in the Power BI Get Data dialog

Open the `TripPin.query.pq` file. Replace the current contents with a call to your exported function.

```
TripPin.Feed("https://services.odata.org/v4/TripPinService/")
```

Select the **Start** button to launch the M Query utility.

The `<project>.query.pq` file is used to test out your extension without having to deploy it to your Power BI Desktop's bin folder. Selecting the **Start** button (or pressing F5) automatically compiles your extension and launches the M Query utility.

Running your query for the first time results in a credential error. In Power Query, the hosting application would convert this error into a credential prompt. In Visual Studio, you'll receive a similar prompt that calls out which data source is missing credentials and its data source path. Select the shortest of the data source paths (`https://services.odata.org/`)—this will apply your credential to all URLs under this path.

Select the **Anonymous** credential type, and then select **Set Credential**.

M Query Output

Output Log Errors Credentials

Errors

Credentials are required to connect to the TripPin source. (Source at [http://services.odata.org/TripPinRESTierService/\(S\(dh0deax3pfzop331wyar1fep\)\)](http://services.odata.org/TripPinRESTierService/(S(dh0deax3pfzop331wyar1fep))).)

Data Source Kind:
TripPin

Data Source Path:

Credential Type:

Privacy:

Set Credential

Select OK to close the dialog, and then select the Start button once again. You see a query execution status dialog, and finally a Query Result table showing the data returned from your query.

MQuery Output

Output Log Errors Credentials

Query Result

Name	Data	Signature
Airlines	[Table]	table
Airports	[Table]	table
GetNearestAirport	[Function]	function (lat as number, lon as number) as record
GetPersonWithMostFriends	[Function]	function () as record
Me	[Table]	singleton
NewComePeople	[Table]	table
People	[Table]	table

You can try out a few different OData URLs in the test file to see what how different results are returned. For example:

- <https://services.odata.org/v4/TripPinService/Me>
- [https://services.odata.org/v4/TripPinService/GetPersonWithMostFriends\(\)](https://services.odata.org/v4/TripPinService/GetPersonWithMostFriends())
- <https://services.odata.org/v4/TripPinService/People>

The TripPin.query.pq file can contain single statements, let statements, or full section documents.

```
let
    Source = TripPin.Feed("https://services.odata.org/v4/TripPinService/"),
    People = Source{[Name="People"]}[Data],
    SelectColumns = Table.SelectColumns(People, {"UserName", "FirstName", "LastName"})
in
    SelectColumns
```

Open **Fiddler** to capture HTTP traffic, and run the query. You should see a few different requires to services.odata.org, generated by the mashup container process. You can see that accessing the root URL of the service results in a 302 status and a redirect to the longer version of the URL. Following redirects is another behavior you get “for free” from the base library functions.

One thing to note if you look at the URLs is that you can see the query folding that happened with the **SelectColumns** statement.

[https://services.odata.org/v4/TripPinService/People?\\$select=UserName%2CFirstName%2CLastName](https://services.odata.org/v4/TripPinService/People?$select=UserName%2CFirstName%2CLastName)

If you add more transformations to your query, you can see how they impact the generated URL.

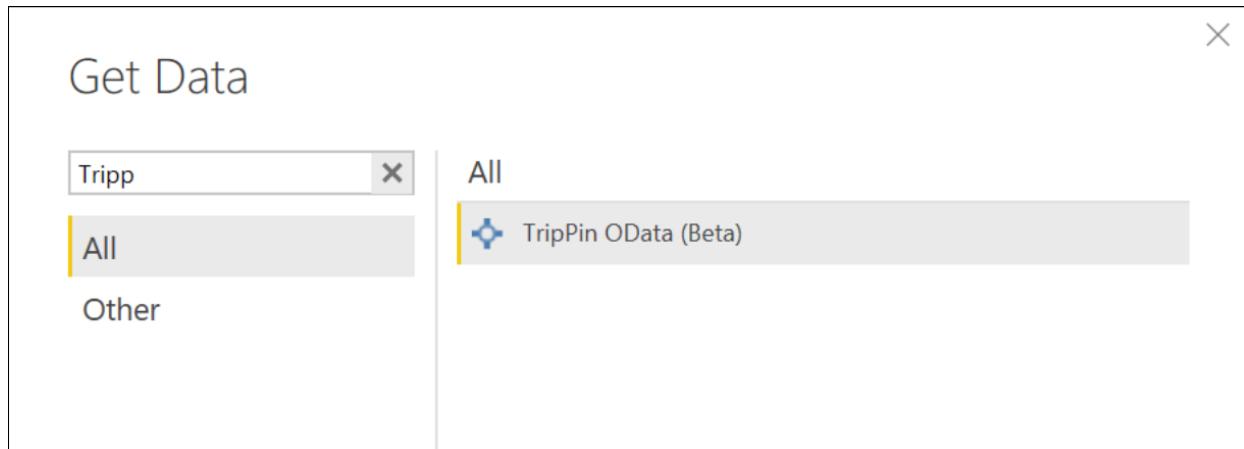
This behavior is important to note. Even though you did not implement explicit folding logic, your connector inherits these capabilities from the `OData.Feed` function. M statements are compose-able—filter contexts will flow from one function to another, whenever possible. This is similar in concept to the way data source functions used within your connector inherit their authentication context and credentials. In later lessons, you'll replace the use of `OData.Feed`, which has native folding capabilities, with `Web.Contents`, which does not. To get the same level of capabilities, you'll need to use the `Table.View` interface and implement your own explicit folding logic.

Loading Your Extension in Power BI Desktop

To use your extension in Power BI Desktop, you'll need to copy your connector project's output file (`TripPin.mez`) to your Custom Connectors directory.

1. In Visual Studio, select **Build | Build Solution (F6)** from the menu bar. This will generate the .mez file for your project. By default, this will go in your project's bin\Debug folder.
2. Create a `[My Documents]\Power BI Desktop\Custom Connectors` directory.
3. Copy the extension file into this directory.
4. Check the option **(Not Recommended) Allow any extension to load without validation or warning** in Power BI Desktop (under **File > Options and settings > Options > Security > Data Extensions**).
5. Restart Power BI Desktop.
6. Select **Get Data > More** to bring up the **Get Data** dialog.

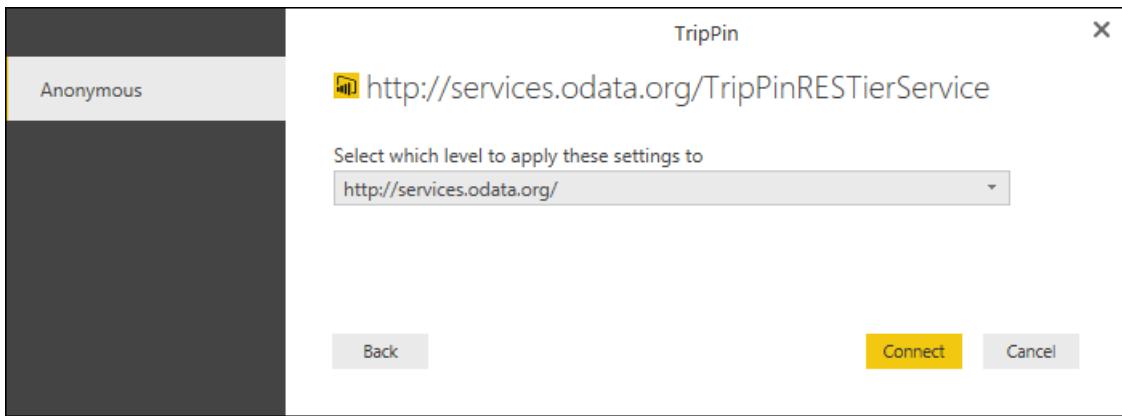
You can locate your extension by typing its name into the search box.



Select the function name, and select **Connect**. A third-party message appears—select **Continue** to continue. The function invocation dialog now appears. Enter the root URL of the service (`https://services.odata.org/v4/TripPinService/`), and select **OK**.



Since this is the first time you are accessing this data source, you'll receive a prompt for credentials. Check that the shortest URL is selected, and then select **Connect**.



Notice that instead of getting a simple table of data, the navigator appears. This is because the [OData.Feed](#) function returns a table with special metadata on top of it that the Power Query experience knows to display as a navigation table. This walkthrough will cover how you can create and customize your own navigation table in a future lesson.

AirlineCode	Name
AA	American Airlines
FM	Shanghai Airline
MU	China Eastern Airlines
AF	Air France
AZ	Alitalia
AC	Air Canada
OS	Austrian Airlines
TK	Turkish Airlines
JL	Japan Airlines
SQ	Singapore Airlines
KE	Korean Air
CZ	China Southern
AK	AirAsia

Select the **Me** table, and then select **Transform Data**. Notice that the columns already have types assigned (well, most of them). This is another feature of the underlying [OData.Feed](#) function. If you watch the requests in [Fiddler](#), you'll see that you've fetched the service's \$metadata document. The engine's OData implementation does this automatically to determine the service's schema, data types, and relationships.

Conclusion

This lesson walked you through the creation of a simple connector based on the [OData.Feed](#) library function. As you saw, very little logic is needed to enable a fully functional connector over the [OData](#) base function. Other extensibility enabled functions, such as [ODBC.DataSource](#), provide similar capabilities.

In the next lesson, you'll replace the use of [OData.Feed](#) with a less capable function—[Web.Contents](#). Each lesson will implement more connector features, including paging, metadata/schema detection, and query folding to the OData query syntax, until your custom connector supports the same range of capabilities as [OData.Feed](#).

Next steps

[TripPin Part 2 - Data Connector for a REST Service](#)

TripPin Part 2 - Data Connector for a REST Service

1/15/2022 • 7 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Create a base function that calls out to a REST API using [Web.Contents](#)
- Learn how to set request headers and process a JSON response
- Use Power BI Desktop to wrangle the response into a user friendly format

This lesson converts the OData based connector for the [TripPin service](#) (created in the [previous lesson](#)) to a connector that resembles something you'd create for any RESTful API. OData is a RESTful API, but one with a fixed set of conventions. The advantage of OData is that it provides a schema, data retrieval protocol, and standard query language. Taking away the use of [OData.Feed](#) will require us to build these capabilities into the connector ourselves.

Recap of the OData Connector

Before you remove the OData functions from your connector, let's do a quick review of what it currently does (mostly behind the scenes) to retrieve data from the service.

Open the TripPin connector project from [Part 1](#) in Visual Studio. Open the Query file and paste in the following query:

```
TripPin.Feed("https://services.odata.org/v4/TripPinService/Me")
```

Open Fiddler and then select the Start button in Visual Studio.

In Fiddler, you'll see three requests to the server:

#	Result	Protocol	Host	URL
25	200	HTTP	services.odata.org	/TripPinRESTierService/(S(njbl4y1givihcajsba2xiorr))/Me
26	200	HTTP	services.odata.org	/TripPinRESTierService/(S(njbl4y1givihcajsba2xiorr))/\$metadata
27	204	HTTP	services.odata.org	/TripPinRESTierService/(S(njbl4y1givihcajsba2xiorr))/Me/BestFriend

- `/Me` —the actual URL you are requesting.
- `/$metadata` —a call automatically made by the `OData.Feed` function to determine schema and type information about the response.
- `/Me/BestFriend` —one of the fields that was (eagerly) pulled when you listed the `/Me` singleton. In this case the call resulted in a `204 No Content` status.

M evaluation is mostly lazy. In most cases, data values are only retrieved/pulled when they are needed. There are scenarios (like the `/Me/BestFriend` case) where a value is pulled eagerly. This tends to occur when type information is needed for a member, and the engine has no other way to determine the type than to retrieve the value and inspect it. Making things lazy (that is, avoiding eager pulls) is one of the key aspects to making an M connector performant.

Note the request headers that were sent along with the requests and the JSON format of the response of the `/Me` request.

```
{
    "@odata.context": "https://services.odata.org/v4/TripPinService/$metadata#Me",
    "UserName": "aprilcline",
    "FirstName": "April",
    "LastName": "Cline",
    "MiddleName": null,
    "Gender": "Female",
    "Age": null,
    "Emails": [ "April@example.com", "April@contoso.com" ],
    "FavoriteFeature": "Feature1",
    "Features": [ ],
    "AddressInfo": [
        {
            "Address": "P.O. Box 555",
            "City": {
                "Name": "Lander",
                "CountryRegion": "United States",
                "Region": "WY"
            }
        }
    ],
    "HomeAddress": null
}
```

When the query finishes evaluating, the M Query Output window should show the Record value for the Me singleton.

MQuery Output

Name	Value
UserName	aprilcline
FirstName	April
LastName	Cline
MiddleName	
Gender	Female
Age	
Emails	[List]
AddressInfo	[List]
HomeAddress	
FavoriteFeature	Feature1
Features	[List]
Friends	[Table]
BestFriend	
Trips	[Table]
GetFavoriteAirline	
GetFriendsTrips	
UpdatePersonLastName	

If you compare the fields in the output window with the fields returned in the raw JSON response, you'll notice a mismatch. The query result has additional fields (Friends , Trips , GetFriendsTrips) that don't appear anywhere in the JSON response. The [OData.Feed](#) function automatically appended these fields to the record based on the schema returned by \$metadata. This is a good example of how a connector might augment and/or reformat the response from the service to provide a better user experience.

Creating a Basic REST Connector

You'll now be adding a new exported function to your connector that calls [Web.Contents](#).

To be able to make successful web requests to the OData service, however, you'll have to set some [standard OData headers](#). You'll do this by defining a common set of headers as a new variable in your connector:

```
DefaultRequestHeaders = [
    #"Accept" = "application/json;odata.metadata=minimal", // column name and values only
    #"OData-MaxVersion" = "4.0"                                // we only support v4
];
```

You'll change your implementation of your `TripPin.Feed` function so that rather than using `OData.Feed`, it uses `Web.Contents` to make a web request, and parses the result as a JSON document.

```
TripPinImpl = (url as text) =>
    let
        source = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
        json = Json.Document(source)
    in
        json;
```

You can now test this out in Visual Studio using the query file. The result of the /Me record now resembles the raw JSON that you saw in the Fiddler request.

If you watch Fiddler when running the new function, you'll also notice that the evaluation now makes a single web request, rather than three. Congratulations—you've achieved a 300% performance increase! Of course, you've now lost all the type and schema information, but there's no need to focus on that part just yet.

Update your query to access some of the TripPin Entities/Tables, such as:

- <https://services.odata.org/v4/TripPinService/Airlines>
- <https://services.odata.org/v4/TripPinService/Airports>
- <https://services.odata.org/v4/TripPinService/Me/Trips>

You'll notice that the paths that used to return nicely formatted tables now return a top level "value" field with an embedded [List]. You'll need to do some transformations on the result to make it usable for Power BI scenarios.

The screenshot shows the 'MQuery Output' window with a yellow header bar. Below the header are tabs: 'Output' (which is selected), 'Log', 'Errors', and 'Credentials'. The main area is titled 'Query Result' and contains a table with two rows:

Name	Value
@odata.context	http://services.odata.org/TripPinRESTierService/(S(njbl4y1givihcajsba2xiorr))/\$.metadata#Collection(Microsoft.OData.Service
value	[List]

Authoring Transformations in Power Query

While it is certainly possible to author your M transformations by hand, most people prefer to use Power Query to shape their data. You'll open your extension in Power BI Desktop and use it to design queries to turn the output into a more user friendly format. Rebuild your solution, copy the new extension file to your Custom Data Connectors directory, and relaunch Power BI Desktop.

Start a new Blank Query, and paste the following into the formula bar:

```
= TripPin.Feed("https://services.odata.org/v4/TripPinService/Airlines")
```

Be sure to include the = sign.

Manipulate the output until it looks like the original OData feed—a table with two columns: AirlineCode and Name.

	AirlineCode	Name
1	AA	American Airlines
2	FM	Shanghai Airline
3	MU	China Eastern Airlines

The resulting query should look something like this:

```
let
    Source = TripPin.Feed("https://services.odata.org/v4/TripPinService/Airlines"),
    value = Source[value],
    toTable = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    expand = Table.ExpandRecordColumn(toTable, "Column1", {"AirlineCode", "Name"}, {"AirlineCode", "Name"})
in
    expand
```

Give the query a name ("Airlines").

Create a new Blank Query. This time, use the `TripPin.Feed` function to access the /Airports entity. Apply transforms until you get something similar to the share shown below. The matching query can also be found below—give this query a name ("Airports") as well.

	Name	IcaoCode	IataCode	Address	City	CountryRegion	Region	Latitude	Longitude
1	San Francisco International Airport	KSFO	SFO	South McDonnell Road, San Francisco, CA 94128	San Francisco	United States	California	37.61888889	-122.3747222
2	Los Angeles International Airport	KLAX	LAX	1 World Way, Los Angeles, CA, 90045	Los Angeles	United States	California	33.9425	-118.4080556
3	Shanghai Hongqiao International Airport	ZSSS	SHA	Hongqiao Road 2550, Changning District	Shanghai	China	Shanghai	31.1977778	121.3861111
4	Beijing Capital International Airport	ZBAA	PEK	Airport Road, Chaoyang District, Beijing, 100621	Beijing	China	Beijing	40.08	116.5844444
5	John F. Kennedy International Airport	KJFK	JFK	Jamaica, New York, NY 11430	New York City	United States	New York	40.63972222	-73.77888889

```
let
    Source = TripPin.Feed("https://services.odata.org/v4/TripPinService/Airports"),
    value = Source[value],
    #"Converted to Table" = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Expanded Column1" = Table.ExpandRecordColumn(#"Converted to Table", "Column1", {"Name", "IcaoCode", "IataCode", "Location"}, {"Name", "IcaoCode", "IataCode", "Location"}),
    #"Expanded Location" = Table.ExpandRecordColumn(#"Expanded Column1", "Location", {"Address", "Loc", "City"}, {"Address", "Loc", "City"}),
    #"Expanded City" = Table.ExpandRecordColumn(#"Expanded Location", "City", {"Name", "CountryRegion", "Region"}, {"Name.1", "CountryRegion", "Region"}),
    #"Renamed Columns" = Table.RenameColumns(#"Expanded City", {"Name.1", "City"}),
    #"Expanded Loc" = Table.ExpandRecordColumn(#"Renamed Columns", "Loc", {"coordinates"}, {"coordinates"}),
    #"Added Custom" = Table.AddColumn(#"Expanded Loc", "Latitude", each [coordinates]{1}),
    #"Added Custom1" = Table.AddColumn(#"Added Custom", "Longitude", each [coordinates]{0}),
    #"Removed Columns" = Table.RemoveColumns(#"Added Custom1", {"coordinates"}),
    #"Changed Type" = Table.TransformColumnTypes(#"Removed Columns", {"Name", type text}, {"IcaoCode", type text}, {"IataCode", type text}, {"Address", type text}, {"City", type text}, {"CountryRegion", type text}, {"Region", type text}, {"Latitude", type number}, {"Longitude", type number})
in
    #"Changed Type"
```

You can repeat this process for additional paths under the service. Once you're ready, move onto the next step of creating a (mock) navigation table.

Simulating a Navigation Table

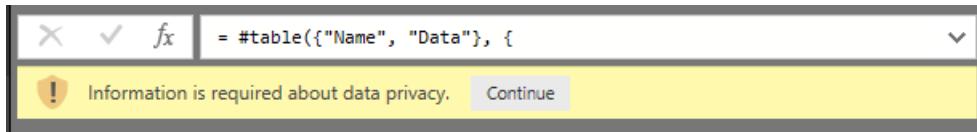
Now you are going to build a table (using M code) that presents your nicely formatted TripPin entities.

Start a new Blank Query and bring up the Advanced Editor.

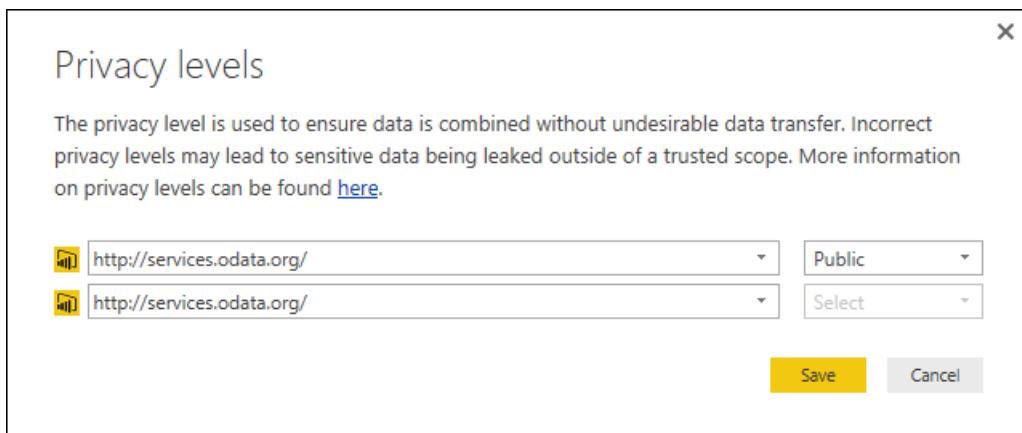
Paste in the following query:

```
let
  source = #table({ "Name", "Data" }, [
    { "Airlines", Airlines },
    { "Airports", Airports }
  ])
in
  source
```

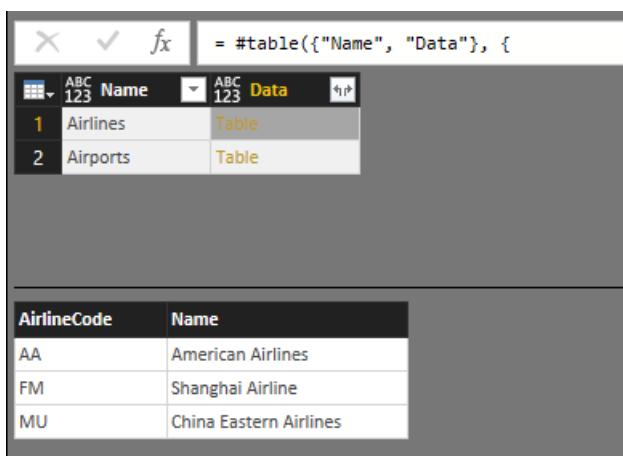
If you have not set your Privacy Levels setting to "Always ignore Privacy level settings" (also known as "Fast Combine") you'll see a privacy prompt.



Privacy prompts appear when you're combining data from multiple sources and have not yet specified a privacy level for the source(s). Select the **Continue** button and set the privacy level of the top source to **Public**.



Select **Save** and your table will appear. While this isn't a navigation table yet, it provides the basic functionality you need to turn it into one in a subsequent lesson.



Data combination checks do not occur when accessing multiple data sources from within an extension. Since all data source calls made from within the extension inherit the same authorization context, it is assumed they are "safe" to combine. Your extension will always be treated as a single data source when it comes to data combination rules. Users would still receive the regular privacy prompts when combining your source with other M sources.

If you run Fiddler and click the **Refresh Preview** button in the Query Editor, you'll notice separate web requests for each item in your navigation table. This indicates that an eager evaluation is occurring, which isn't ideal when

building navigation tables with a lot of elements. Subsequent lessons will show how to build a proper navigation table that supports lazy evaluation.

Conclusion

This lesson showed you how to build a simple connector for a REST service. In this case, you turned an existing OData extension into a standard REST extension (using [Web.Contents](#)), but the same concepts apply if you were creating a new extension from scratch.

In the next lesson, you'll take the queries created in this lesson using Power BI Desktop and turn them into a true navigation table within the extension.

Next steps

[TripPin Part 3 - Navigation Tables](#)

TripPin Part 3 - Navigation Tables

1/15/2022 • 4 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Create a navigation table for a fixed set of queries
- Test the navigation table in Power BI Desktop

This lesson adds a navigation table to the TripPin connector created in the [previous lesson](#). When your connector used the `odata.Feed` function ([Part 1](#)), you received the navigation table “for free”, as derived from the OData service’s `$metadata` document. When you moved to the `Web.Contents` function ([Part 2](#)), you lost the built-in navigation table. In this lesson, you’ll take a set of fixed queries you created in Power BI Desktop and add the appropriate metadata for Power Query to popup the **Navigator** dialog for your data source function.

See the [Navigation Table documentation](#) for more information about using navigation tables.

Defining Fixed Queries in the Connector

A simple connector for a REST API can be thought of as a fixed set of queries, each returning a table. These tables are made discoverable through the connector’s navigation table. Essentially, each item in the navigator is associated with a specific URL and set of transformations.

You’ll start by copying the queries you wrote in Power BI Desktop (in the previous lesson) into your connector file. Open the TripPin Visual Studio project, and paste the Airlines and Airports queries into the `TripPin.pq` file. You can then turn those queries into functions that take a single text parameter:

```

GetAirlinesTable = (url as text) as table =>
    let
        source = TripPin.Feed(url & "Airlines"),
        value = source[value],
        toTable = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
        expand = Table.ExpandRecordColumn(toTable, "Column1", {"AirlineCode", "Name"}, {"AirlineCode", "Name"})
    in
        expand;

GetAirportsTable = (url as text) as table =>
    let
        source = TripPin.Feed(url & "Airports"),
        value = source[value],
        #"Converted to Table" = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
        #"Expanded Column1" = Table.ExpandRecordColumn(#"Converted to Table", "Column1", {"Name", "IcaoCode", "IataCode", "Location"}, {"Name", "IcaoCode", "IataCode", "Location"}),
        #"Expanded Location" = Table.ExpandRecordColumn(#"Expanded Column1", "Location", {"Address", "Loc", "City"}, {"Address", "Loc", "City"}),
        #"Expanded City" = Table.ExpandRecordColumn(#"Expanded Location", "City", {"Name", "CountryRegion", "Region"}, {"Name.1", "CountryRegion", "Region"}),
        #"Renamed Columns" = Table.RenameColumns(#"Expanded City", {[{"Name.1", "City"}]}),
        #"Expanded Loc" = Table.ExpandRecordColumn(#"Renamed Columns", "Loc", {"coordinates"}, {"coordinates"}),
        #"Added Custom" = Table.AddColumn(#"Expanded Loc", "Latitude", each [coordinates]{1}),
        #"Added Custom1" = Table.AddColumn(#"Added Custom", "Longitude", each [coordinates]{0}),
        #"Removed Columns" = Table.RemoveColumns(#"Added Custom1", {"coordinates"}),
        #"Changed Type" = Table.TransformColumnTypes(#"Removed Columns", [{"Name", type text}, {"IcaoCode", type text}, {"IataCode", type text}, {"Address", type text}, {"City", type text}, {"CountryRegion", type text}, {"Region", type text}, {"Latitude", type number}, {"Longitude", type number}])
    in
        #"Changed Type";

```

Next you'll import the mock navigation table query you wrote that creates a fixed table linking to these data set queries. Call it `TripPinNavTable`:

```

TripPinNavTable = (url as text) as table =>
    let
        source = #table({"Name", "Data"}, [
            { "Airlines", GetAirlinesTable(url) },
            { "Airports", GetAirportsTable(url) }
        ])
    in
        source;

```

Finally you'll declare a new shared function, `TripPin.Contents`, that will be used as your main data source function. You'll also remove the `Publish` value from `TripPin.Feed` so that it no longer shows up in the **Get Data** dialog.

```

[DataSource.Kind="TripPin"]
shared TripPin.Feed = Value.ReplaceType(TripPinImpl, type function (url as Uri.Type) as any);

[DataSource.Kind="TripPin", Publish="TripPin.Publish"]
shared TripPin.Contents = Value.ReplaceType(TripPinNavTable, type function (url as Uri.Type) as any);

```

NOTE

Your extension can mark multiple functions as `shared`, with or without associating them with a `DataSource.Kind`. However, when you associate a function with a specific `DataSource.Kind`, each function **must** have the same set of *required* parameters, with the same name and type. This is because the data source function parameters are combined to make a 'key' used for looking up cached credentials.

You can test your `TripPin.Contents` function using your `TripPin.query.pq` file. Running the following test query will give you a credential prompt, and a simple table output.

```
TripPin.Contents("https://services.odata.org/v4/TripPinService/")
```

The screenshot shows the 'M Query Output' window. At the top, there are tabs for 'Output' (which is selected), 'Log', 'Errors', and 'Credentials'. Below the tabs, the title 'Query Result' is centered. The main area displays a table with two rows. The first row has two columns: 'Name' and 'Data'. The second row contains two entries: 'Airlines' and '[Table]'. The third row contains 'Airports' and '[Table]'. The fourth row is empty.

Name	Data
Airlines	[Table]
Airports	[Table]

Creating a Navigation Table

You'll use the handy `Table.ToNavigationTable` function to format your static table into something that Power Query will recognize as a Navigation Table.

```
Table.ToNavigationTable = (
    table as table,
    keyColumns as list,
    nameColumn as text,
    dataColumn as text,
    itemKindColumn as text,
    itemNameColumn as text,
    isLeafColumn as text
) as table =>
    let
        tableType = Value.Type(table),
        newTableType = Type.AddTableKey(tableType, keyColumns, true) meta
        [
            NavigationTable.NameColumn = nameColumn,
            NavigationTable.DataColumn = dataColumn,
            NavigationTable.ItemKindColumn = itemKindColumn,
            Preview.DelayColumn = itemNameColumn,
            NavigationTable.IsLeafColumn = isLeafColumn
        ],
        navigationTable = Value.ReplaceType(table, newTableType)
    in
        navigationTable;
```

After copying this into your extension file, you'll update your `TripPinNavTable` function to add the navigation

table fields.

```
TripPinNavTable = (url as text) as table =>
    let
        source = #table({ "Name", "Data", "ItemKind", "ItemName", "IsLeaf"}, {
            { "Airlines", GetAirlinesTable(url), "Table", "Table", true },
            { "Airports", GetAirportsTable(url), "Table", "Table", true }
        }),
        navTable = Table.ToNavigationTable(source, { "Name" }, "Name", "Data", "ItemKind", "ItemName",
        "IsLeaf")
    in
        navTable;
```

Running your test query again will give you a similar result as last time—with a few more columns added.

The screenshot shows the 'M Query Output' window in Visual Studio. The title bar says 'M Query Output'. Below it is a tab bar with 'Output' (which is selected and highlighted in blue), 'Log', 'Errors', and 'Credentials'. The main area is titled 'Query Result' and contains a table with five columns: 'Name', 'Data', 'ItemKind', 'ItemName', and 'IsLeaf'. There are two rows in the table. The first row has 'Airlines' in the 'Name' column, 'Table' in 'Data', 'Table' in 'ItemKind', 'Table' in 'ItemName', and 'True' in 'IsLeaf'. The second row has 'Airports' in the 'Name' column, 'Table' in 'Data', 'Table' in 'ItemKind', 'Table' in 'ItemName', and 'True' in 'IsLeaf'. All other cells in the table are empty.

NOTE

You will not see the **Navigator** window appear in Visual Studio. The **M Query Output** window always displays the underlying table.

If you copy your extension over to your Power BI Desktop custom connector and invoke the new function from the **Get Data** dialog, you'll see your navigator appear.

The screenshot shows the 'Navigator' pane in Power BI Desktop. At the top left is a search bar with a magnifying glass icon. Below it is a 'Display Options' dropdown menu. Underneath is a navigation tree. The root node is a folder icon followed by the URL 'http://services.odata.org/TripPinRESTierService...'. Below this are two expanded nodes: 'Airlines' and 'Airports'. The 'Airlines' node is currently selected, indicated by a blue border. To the right of the navigation tree is a detailed view of the 'Airlines' table. The table has two columns: 'AirlineCode' and 'Name'. It contains three rows: 'AA' (American Airlines), 'FM' (Shanghai Airline), and 'MU' (China Eastern Airlines).

If you right click on the root of the navigation tree and select **Edit**, you'll see the same table as you did within Visual Studio.

The screenshot shows the Microsoft Power Query Editor interface. The top ribbon has tabs for File, Home, Transform, Add Column, and View. The Home tab is selected. The ribbon contains various icons for file operations like Close & Apply, New Source, Refresh, and Transform functions like Choose Columns, Remove Rows, and Sort.

The main area displays a query titled "TripPin.Contents("http://services.odata.org/TripPinRESTierService/(\$")". It shows a table with two rows:

	Name	Data	ItemKind	ItemName	IsLeaf
1	Airlines	Table	Table	Table	TRUE
2	Airports	Table	Table	Table	TRUE

The bottom status bar indicates "5 COLUMNS, 2 ROWS" and "PREVIEW DOWNLOADED AT 11:17 AM".

The right side features a "Query Settings" pane with sections for "PROPERTIES" (Name: http://services.odata.org/TripPinRESTierS) and "APPLIED STEPS" (Source).

Conclusion

In this tutorial, you added a [Navigation Table](#) to your extension. Navigation Tables are a key feature that make connectors easier to use. In this example your navigation table only has a single level, but the Power Query UI supports displaying navigation tables that have multiple dimensions (even when they are ragged).

Next steps

[TripPin Part 4 - Data Source Paths](#)

TripPin Part 4 - Data Source Paths

1/15/2022 • 6 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Simplify the connection logic for your connector
- Improve the navigation table experience

This lesson simplifies the connector built in the [previous lesson](#) by removing its required function parameters, and improving the user experience by moving to a dynamically generated navigation table.

For an in-depth explanation of how credentials are identified, see the [Data Source Paths section of Handling Authentication](#).

Data Source Paths

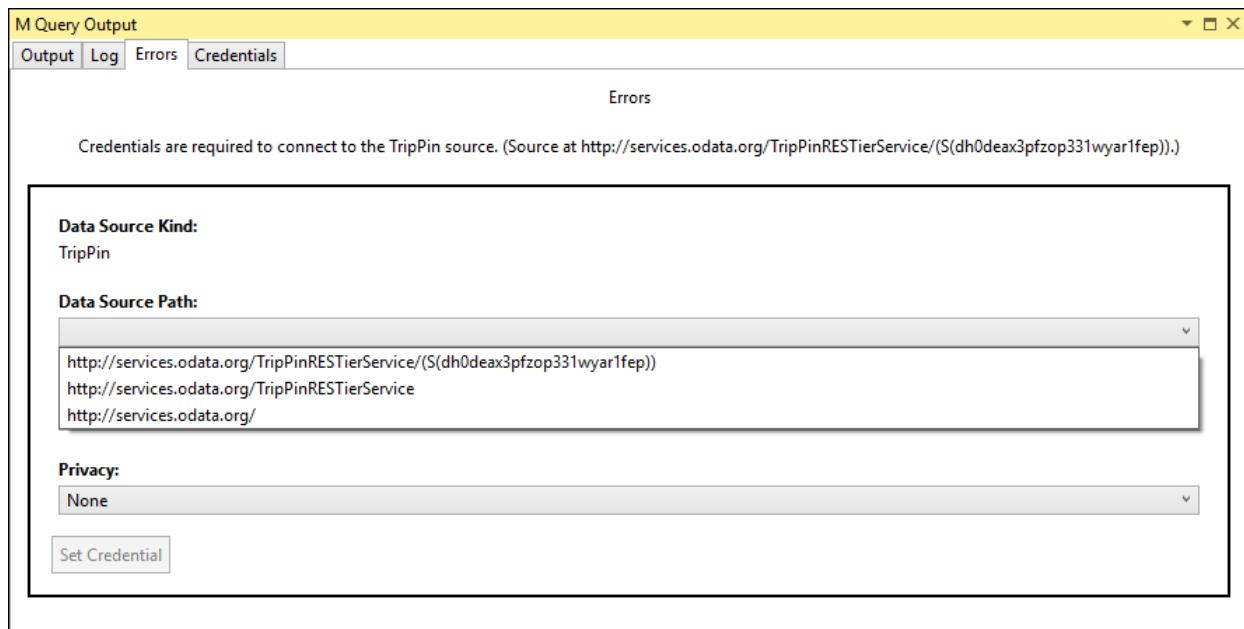
When invoking a [data source function](#), the M engine identifies which credentials to use during an evaluation by doing a lookup based on the [Data Source Kind](#) and [Data Source Path](#) values.

In the [previous lesson](#) you shared two data source functions, both with a single *Uri.Type* parameter.

```
[DataSource.Kind="TripPin"]
shared TripPin.Feed = Value.ReplaceType(TripPinImpl, type function (url as Uri.Type) as any);

[DataSource.Kind="TripPin", Publish="TripPin.Publish"]
shared TripPin.Contents = Value.ReplaceType(TripPinNavTable, type function (url as Uri.Type) as any);
```

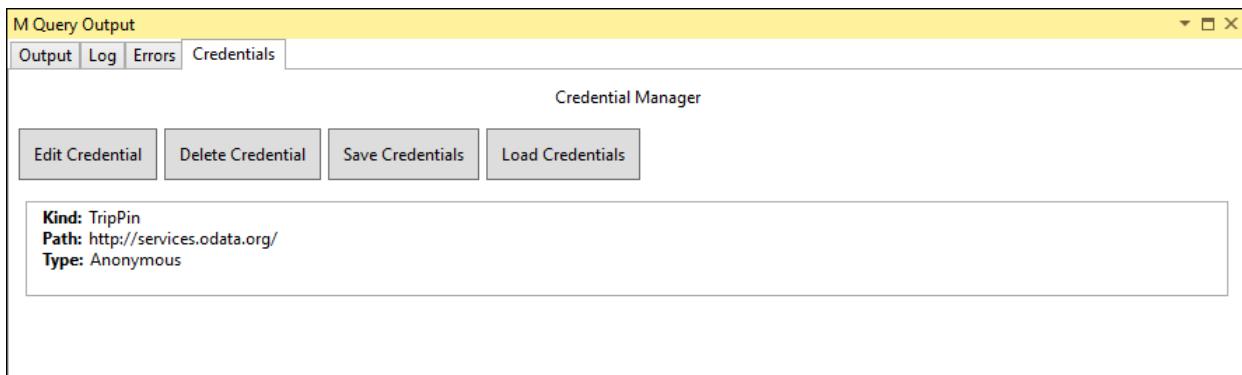
The first time you run a query that uses one of the functions, you'll receive a credential prompt with drop downs that lets you select a path and an authentication type.



If you run the same query again, with the same parameters, the M engine is able to locate the cached

credentials, and no credential prompt is shown. If you modify the `url` argument to your function so that the base path no longer matches, a new credential prompt is displayed for the new path.

You can see any cached credentials on the Credentials table in the **M Query Output** window.



Depending on the type of change, modifying the parameters of your function will likely result in a credential error.

Simplifying the Connector

You'll now simplify your connector by removing the parameters for your data source function (`TripPin.Contents`). You'll also remove the `shared` qualifier for `TripPin.Feed`, and leave it as an internal-only function.

One of the design philosophies of Power Query is to keep the initial data source dialog as simple as possible. If at all possible, you should provide the user with choices at the Navigator level, rather than the connection dialog. If a user provided value can be determined programmatically, consider adding it as the top level of your navigation table rather than a function parameter.

For example, when connecting to a relational database, you might need server, database, and table names. Once you know the server to connect to, and credentials have been provided, you could use the database's API to fetch a list of databases, and a list of tables contained within each database. In this case, to keep your initial connect dialog as simple as possible, only the server name should be a required parameter—`Database` and `Table` would be levels of your navigation table.

Since the TripPin service has a fixed URL endpoint, you don't need to prompt the user for any values. You'll remove the `url` parameter from your function, and define a `BaseUrl`/variable in your connector.

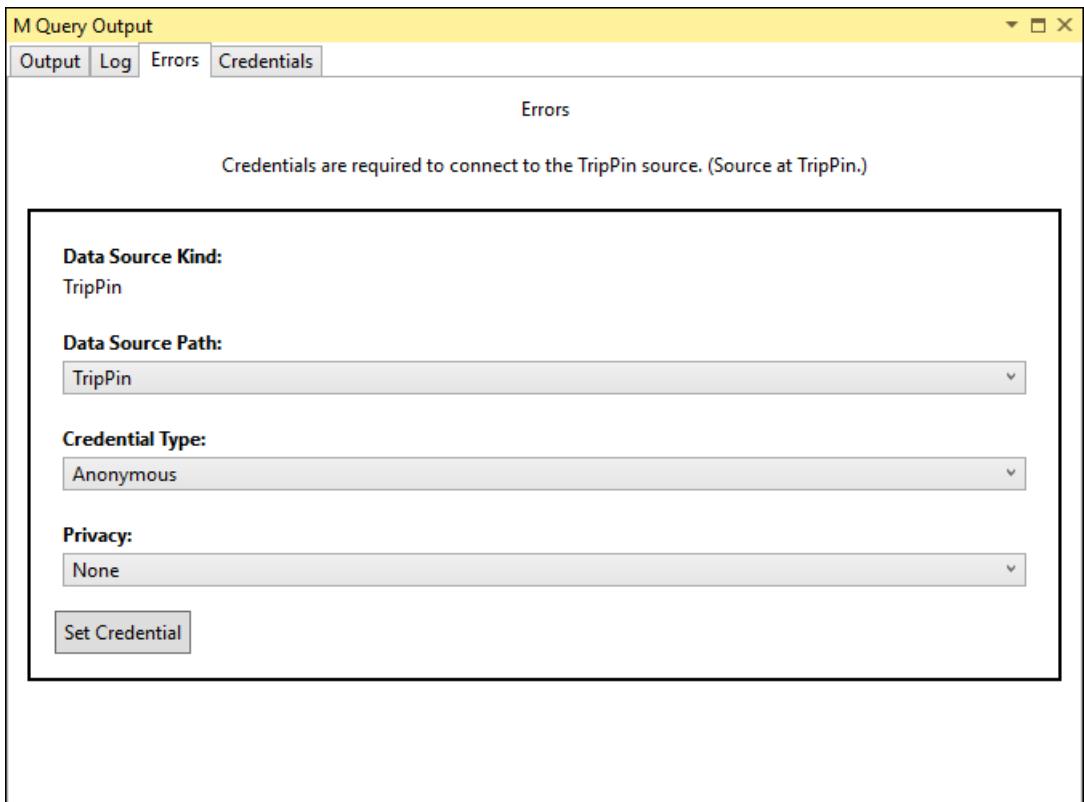
```
BaseUrl = "https://services.odata.org/v4/TripPinService/";

[DataSource.Kind="TripPin", Publish="TripPin.Publish"]
shared TripPin.Contents = () => TripPinNavTable(BaseUrl) as table;
```

You'll keep the `TripPin.Feed` function, but no longer make it shared, no longer associate it with a Data Source Kind, and simplify its declaration. From this point on, you'll only use it internally within this section document.

```
TripPin.Feed = (url as text) =>
    let
        source = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
        json = Json.Document(source)
    in
        json;
```

If you update the `TripPin.Contents()` call in your `TripPin.query.pq` file and run it in Visual Studio, you'll see a new credential prompt. Note that there is now a single Data Source Path value—TripPin.



Improving the Navigation Table

In the [first tutorial](#) you used the built-in `odata` functions to connect to the TripPin service. This gave you a really nice looking navigation table, based on the TripPin service document, with no additional code on your side. The `OData.Feed` function automatically did the hard work for you. Since you're "roughing it" by using `Web.Contents` rather than `OData.Feed`, you'll need to recreate this navigation table yourself.

The screenshot shows the 'Navigator' window. On the left, there is a tree view of available services and entities:

- http://services.odata.org/TripPinRESTierS...
 - Airlines
 - Airports
 - Me
 - NewComePeople
 - People
 - GetNearestAirport
 - GetPersonWithMostFriends

On the right, a table named 'Airlines' is displayed with the following data:

AirlineCode	Name
AA	American Airlines
FM	Shanghai Airline
MU	China Eastern Airlines

At the bottom of the window, there are buttons for 'Select Related Tables', 'Load', 'Edit', and 'Cancel'.

You're going to make the following changes:

1. Define a list of items to show in your navigation table
2. Do away with the entity specific functions (`GetAirlineTables` and `GetAirportsTable`)

Generating a Navigation Table from a List

You'll list the entities you want to expose in the navigation table, and build the appropriate URL to access them. Since all of the entities are under the same root path, you'll be able build these URLs dynamically.

To simplify the example, you'll only expose the three entity sets (Airlines, Airports, People), which would be exposed as Tables in M, and skip the singleton (Me) which would be exposed as a Record. You'll skip adding the functions until a later lesson.

```
RootEntities = {
    "Airlines",
    "Airports",
    "People"
};
```

You then update your `TripPinNavTable` function to build the table a column at a time. The [Data] column for each entity is retrieved by calling `TripPin.Feed` with the full URL to the entity.

```
TripPinNavTable = (url as text) as table =>
let
    entitiesAsTable = Table.FromList(RootEntities, Splitter.SplitByNothing()),
    rename = Table.RenameColumns(entitiesAsTable, {"Column1", "Name"}),
    // Add Data as a calculated column
    withData = Table.AddColumn(rename, "Data", each TripPin.Feed(Uri.Combine(url, [Name])), Uri.Type),
    // Add ItemKind and ItemName as fixed text values
    withItemKind = Table.AddColumn(withData, "ItemKind", each "Table", type text),
    withItemName = Table.AddColumn(withItemKind, "ItemName", each "Table", type text),
    // Indicate that the node should not be expandable
    withIsLeaf = Table.AddColumn(withItemName, "IsLeaf", each true, type logical),
    // Generate the nav table
    navTable = Table.ToNavigationTable(withIsLeaf, {"Name", "Name", "Data", "ItemKind", "ItemName", "IsLeaf"})
in
    navTable;
```

When dynamically building URL paths, make sure you're clear where your forward slashes (/) are! Note that `Uri.Combine` uses the following rules when combining paths:

- When the `relativeUri` parameter starts with a /, it will replace the entire path of the `baseUri` parameter
- If the `relativeUri` parameter *does not* start with a / and `baseUri` ends with a /, the path is appended
- If the `relativeUri` parameter *does not* start with a / and `baseUri` *does not* end with a /, the last segment of the path is replaced

The following image shows examples of this:

	baseUri	relativeUri	Uri.Combine
1	http://services.odata.org/TripPinRESTierService/	/Airports	http://services.odata.org/Airports
2	http://services.odata.org/TripPinRESTierService/	Airports	http://services.odata.org/TripPinRESTierService/Airports
3	http://services.odata.org/TripPinRESTierService	Airports	http://services.odata.org/Airports

Remove the Entity Specific Functions

To make your connector easier to maintain, you'll remove the entity specific formatting functions you used in the previous lesson—`GetAirlineTables` and `GetAirportsTable`. Instead, you'll update `TripPin.Feed` to process the JSON response in a way that will work for all of your entities. Specifically, you take the `value` field of the returned OData JSON payload, and convert it from a list of records to a table.

```

TripPin.Feed = (url as text) =>
let
    source = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
    json = Json.Document(source),
    // The response is a JSON record - the data we want is a list of records in the "value" field
    value = json[value],
    asTable = Table.FromList(value, Splitter.SplitByNothing()),
    // expand all columns from the record
    fields = Record.FieldNames(Table.FirstValue(asTable, [Empty = null])),
    expandAll = Table.ExpandRecordColumn(asTable, "Column1", fields)
in
    expandAll;

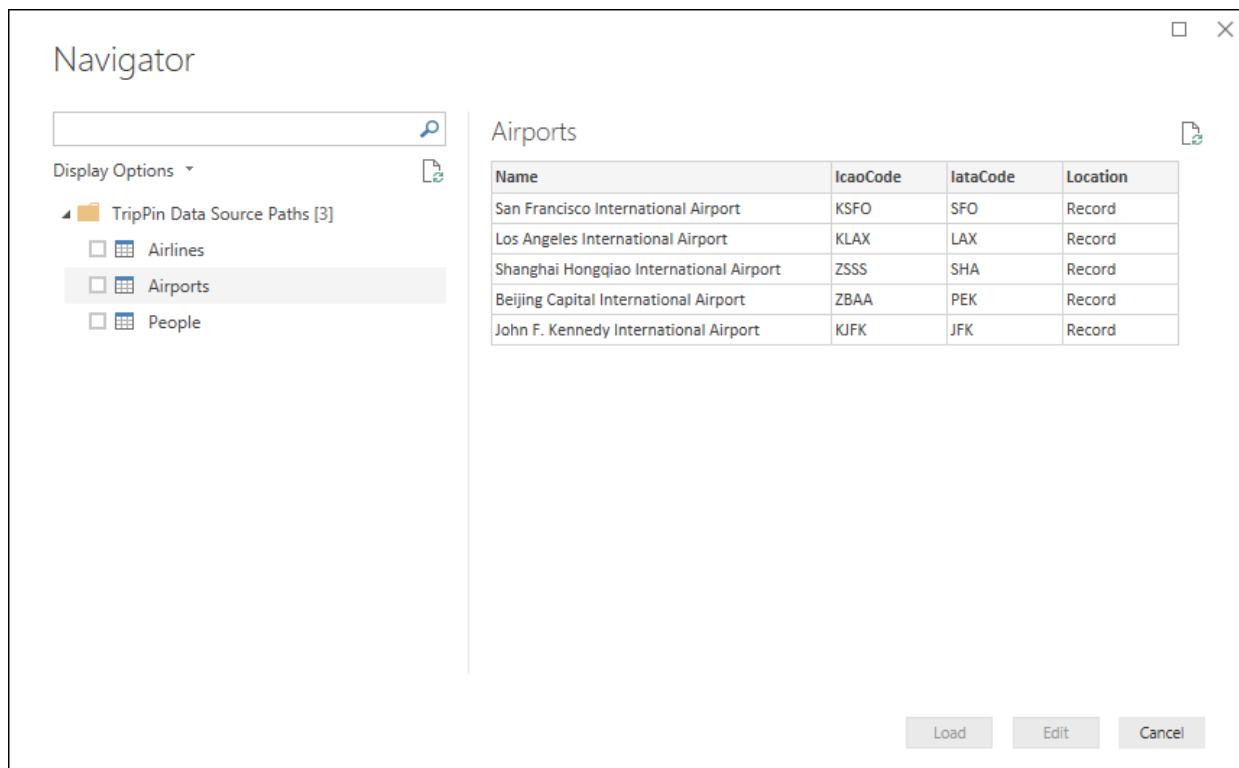
```

NOTE

A disadvantage of using a generic approach to process your entities is that you lose the nice formating and type information for your entities. A later section in this tutorial shows how to enforce schema on REST API calls.

Conclusion

In this tutorial, you cleaned up and simplified your connector by fixing your Data Source Path value, and moving to a more flexible format for your navigation table. After completing these steps (or using the sample code in this directory), the `TripPin.Contents` function returns a navigation table in Power BI Desktop.



Next steps

[TripPin Part 5 - Paging](#)

TripPin Part 5 - Paging

1/15/2022 • 7 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Add paging support to the connector

Many Rest APIs will return data in "pages", requiring clients to make multiple requests to stitch the results together. Although there are some common conventions for pagination (such as [RFC 5988](#)), it generally varies from API to API. Thankfully, TripPin is an OData service, and the [OData standard](#) defines a way of doing pagination using `odata.nextLink` values returned in the body of the response.

To simplify [previous iterations](#) of the connector, the `TripPin.Feed` function was not *page aware*. It simply parsed whatever JSON was returned from the request and formatted it as a table. Those familiar with the OData protocol might have noticed that a number of incorrect assumptions were made on the [format of the response](#) (such as assuming there is a `value` field containing an array of records).

In this lesson you'll improve your response handling logic by making it page aware. Future tutorials will make the page handling logic more robust and able to handle multiple response formats (including errors from the service).

NOTE

You do not need to implement your own paging logic with connectors based on [OData.Feed](#), as it handles it all for you automatically.

Paging Checklist

When implementing paging support, you'll need to know the following things about your API:

- How do you request the next page of data?
- Does the paging mechanism involve calculating values, or do you extract the URL for the next page from the response?
- How do you know when to stop paging?
- Are there parameters related to paging that you should be aware of? (such as "page size")

The answer to these questions will impact the way you implement your paging logic. While there is some amount of code reuse across paging implementations (such as the use of `Table.GenerateByPage`), most connectors will end up requiring custom logic.

NOTE

This lesson contains paging logic for an OData service, which follows a specific format. Check the documentation for your API to determine the changes you'll need to make in your connector to support its paging format.

Overview of OData Paging

OData paging is driven by [nextLink annotations](#) contained within the response payload. The nextLink value contains the URL to the next page of data. You'll know if there is another page of data by looking for an `odata.nextLink` field in outermost object in the response. If there's no `odata.nextLink` field, you've read all of your data.

```
{  
    "odata.context": "...",  
    "odata.count": 37,  
    "value": [  
        { },  
        { },  
        { }  
    ],  
    "odata.nextLink": "...?skiptoken=342r89"  
}
```

Some OData services allow clients to supply a [max page size preference](#), but it is up to the service whether or not to honor it. Power Query should be able to handle responses of any size, so you don't need to worry about specifying a page size preference—you can support whatever the service throws at you.

More information about [Server-Driven Paging](#) can be found in the OData specification.

Testing TripPin

Before fixing your paging implementation, confirm the current behavior of the extension from the [previous tutorial](#). The following test query will retrieve the People table and add an index column to show your current row count.

```
let  
    source = TripPin.Contents(),  
    data = source{[Name="People"]}[Data],  
    withRowCount = Table.AddIndexColumn(data, "Index")  
in  
    withRowCount
```

Turn on fiddler, and run the query in Visual Studio. You'll notice that the query returns a table with 8 rows (index 0 to 7).

Query Result										
@odata.id	@odata.etag	@odata.editLink	UserName	FirstName	LastName	Emails	AddressInfo	Gender	Concurrency	Index
			russellwhyte	Russell	Whyte	[List]	[List]	Male	636348013653063483	0
			scottketchum	Scott	Ketchum	[List]	[List]	Male	636348013653063483	1
			ronaldmundy	Ronald	Mundy	[List]	[List]	Male	636348013653063483	2
			javieralfred	Javier	Alfred	[List]	[List]	Male	636348013653063483	3
			willieashmore	Willie	Ashmore	[List]	[List]	Male	636348013653063483	4
			vincentcalabrese	Vincent	Calabrese	[List]	[List]	Male	636348013653063483	5
			clydeguess	Clyde	Guess	[List]	[List]	Male	636348013653063483	6
			keithpinckney	Keith	Pinckney	[List]	[List]	Male	636348013653063483	7

If you look at the body of the response from fiddler, you'll see that it does in fact contain an `odata.nextLink` field, indicating that there are more pages of data available.

```
{
    "@odata.context": "https://services.odata.org/V4/TripPinService/$metadata#People",
    "@odata.nextLink": "https://services.odata.org/v4/TripPinService/People?%24skiptoken=8",
    "value": [
        { },
        { },
        { },
        { }
    ]
}
```

Implementing Paging for TripPin

You're now going to make the following changes to your extension:

1. Import the common `Table.GenerateByPage` function
2. Add a `GetAllPagesByNextLink` function that uses `Table.GenerateByPage` to glue all pages together
3. Add a `GetPage` function that can read a single page of data
4. Add a `GetNextLink` function to extract the next URL from the response
5. Update `TripPin.Feed` to use the new page reader functions

NOTE

As stated earlier in this tutorial, paging logic will vary between data sources. The implementation here tries to break up the logic into functions that should be reusable for sources that use *next links* returned in the response.

`Table.GenerateByPage`

The `Table.GenerateByPage` function can be used to efficiently combine multiple 'pages' of data into a single table. It does this by repeatedly calling the function passed in as the `getNextPage` parameter, until it receives a `null`. The function parameter must take a single argument, and return a `nullable table`.

```
getNextPage = (lastPage) as nullable table => ...
```

Each call to `getNextPage` receives the output from the previous call.

```
// The getNextPage function takes a single argument and is expected to return a nullable table
Table.GenerateByPage = (getNextPage as function) as table =>
    let
        listOfPages = List.Generate(
            () => getNextPage(null),           // get the first page of data
            (lastPage) => lastPage <> null,    // stop when the function returns null
            (lastPage) => getNextPage(lastPage) // pass the previous page to the next function call
        ),
        // concatenate the pages together
        tableOfPages = Table.FromList(listOfPages, Splitter.SplitByNothing(), {"Column1"}),
        firstRow = tableOfPages{0}?
    in
        // if we didn't get back any pages of data, return an empty table
        // otherwise set the table type based on the columns of the first page
        if (firstRow = null) then
            Table.FromRows({})
        else
            Value.ReplaceType(
                Table.ExpandTableColumn(tableOfPages, "Column1", Table.ColumnNames(firstRow[Column1])),
                Value.Type(firstRow[Column1])
            );
    
```

Some notes about `Table.GenerateByPage`:

- The `getNextPage` function will need to retrieve the next page URL (or page number, or whatever other values are used to implement the paging logic). This is generally done by adding `meta` values to the page before returning it.
- The columns and table type of the combined table (i.e. all pages together) are derived from the first page of data. The `getNextPage` function should normalize each page of data.
- The first call to `getNextPage` receives a null parameter.
- `getNextPage` must return null when there are no pages left.

Implementing GetAllPagesByNextLink

The body of your `GetAllPagesByNextLink` function implements the `getNextPage` function argument for `Table.GenerateByPage`. It will call the `GetPage` function, and retrieve the URL for the next page of data from the `NextLink` field of the `meta` record from the previous call.

```
// Read all pages of data.
// After every page, we check the "NextLink" record on the metadata of the previous request.
// Table.GenerateByPage will keep asking for more pages until we return null.
GetAllPagesByNextLink = (url as text) as table =>
    Table.GenerateByPage((previous) =>
        let
            // if previous is null, then this is our first page of data
            nextLink = if (previous = null) then url else Value.Metadata(previous)[NextLink]?,
            // if NextLink was set to null by the previous call, we know we have no more data
            page = if (nextLink <> null) then GetPage(nextLink) else null
        in
            page
    );

```

Implementing GetPage

Your `GetPage` function will use `Web.Contents` to retrieve a single page of data from the TripPin service, and convert the response into a table. It passes the response from `Web.Contents` to the `GetNextLink` function to extract the URL of the next page, and sets it on the `meta` record of the returned table (page of data).

This implementation is a slightly modified version of the `TripPin.Feed` call from the previous tutorials.

```
GetPage = (url as text) as table =>
    let
        response = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
        body = Json.Document(response),
        nextLink = GetNextLink(body),
        data = Table.FromRecords(body[value])
    in
        data meta [NextLink = nextLink];
```

Implementing GetNextLink

Your `GetNextLink` function simply checks the body of the response for an `@odata.nextLink` field, and returns its value.

```
// In this implementation, 'response' will be the parsed body of the response after the call to
Json.Document.
// Look for the '@odata.nextLink' field and simply return null if it doesn't exist.
GetNextLink = (response) as nullable text => Record.FieldOrDefault(response, "@odata.nextLink");
```

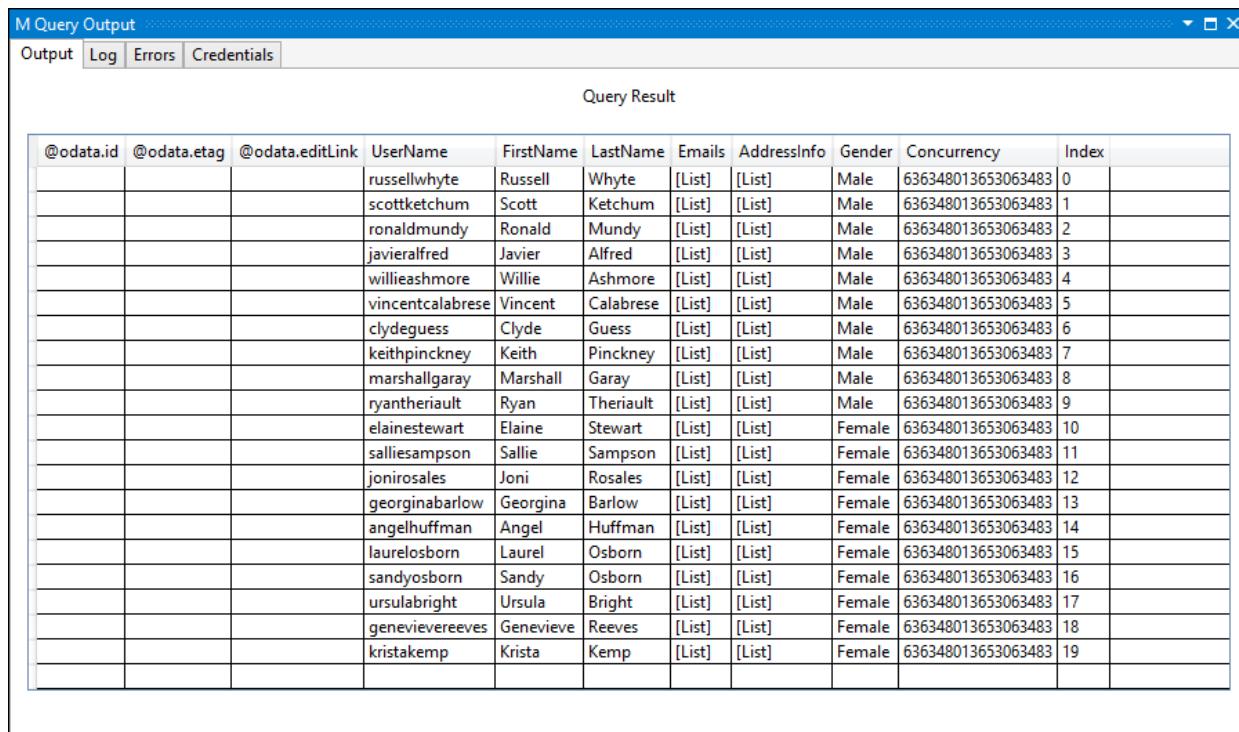
Putting it all Together

The final step to implement your paging logic is to update `TripPin.Feed` to use the new functions. For now, you're simply calling through to `GetAllPagesByNextLink`, but in subsequent tutorials, you'll be adding new

capabilities (such as enforcing a schema, and query parameter logic).

```
TripPin.Feed = (url as text) as table => GetAllPagesByNextLink(url);
```

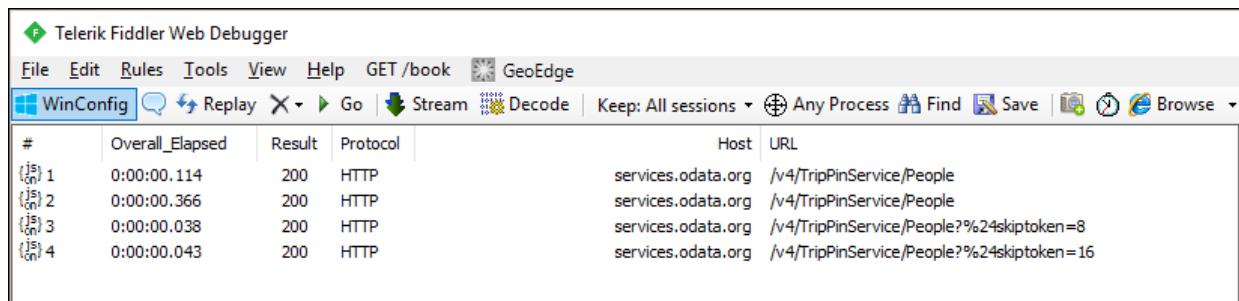
If you re-run the same [test query](#) from earlier in the tutorial, you should now see the page reader in action. You should also see that you have 20 rows in the response rather than 8.



The screenshot shows the 'M Query Output' interface. At the top, there are tabs for 'Output' (which is selected), 'Log', 'Errors', and 'Credentials'. Below this is a section titled 'Query Result' containing a table with 20 rows of data. The columns are: @odata.id, @odata.etag, @odata.editLink, UserName, FirstName, LastName, Emails, AddressInfo, Gender, Concurrency, and Index. The data represents 20 different people with their names, genders, and other details.

@odata.id	@odata.etag	@odata.editLink	UserName	FirstName	LastName	Emails	AddressInfo	Gender	Concurrency	Index
			russellwhyte	Russell	Whyte	[List]	[List]	Male	636348013653063483	0
			scottketchum	Scott	Ketchum	[List]	[List]	Male	636348013653063483	1
			ronaldmundy	Ronald	Mundy	[List]	[List]	Male	636348013653063483	2
			javieralfred	Javier	Alfred	[List]	[List]	Male	636348013653063483	3
			willieashmore	Willie	Ashmore	[List]	[List]	Male	636348013653063483	4
			vincentcalabrese	Vincent	Calabrese	[List]	[List]	Male	636348013653063483	5
			clydegueess	Clyde	Guess	[List]	[List]	Male	636348013653063483	6
			keithpinckney	Keith	Pinckney	[List]	[List]	Male	636348013653063483	7
			marshallgaray	Marshall	Garay	[List]	[List]	Male	636348013653063483	8
			ryantheriault	Ryan	Theriault	[List]	[List]	Male	636348013653063483	9
			elainestewart	Elaine	Stewart	[List]	[List]	Female	636348013653063483	10
			salliesampson	Sallie	Sampson	[List]	[List]	Female	636348013653063483	11
			jonirosales	Joni	Rosales	[List]	[List]	Female	636348013653063483	12
			georginabarlow	Georgina	Barlow	[List]	[List]	Female	636348013653063483	13
			angelhuffman	Angel	Huffman	[List]	[List]	Female	636348013653063483	14
			laurelosborn	Laurel	Osborn	[List]	[List]	Female	636348013653063483	15
			sandyosborn	Sandy	Osborn	[List]	[List]	Female	636348013653063483	16
			ursulabright	Ursula	Bright	[List]	[List]	Female	636348013653063483	17
			genevievereeves	Genevieve	Reeves	[List]	[List]	Female	636348013653063483	18
			kristakemp	Krista	Kemp	[List]	[List]	Female	636348013653063483	19

If you look at the requests in fiddler, you should now see separate requests for each page of data.



The screenshot shows the Telerik Fiddler Web Debugger interface. The top menu includes File, Edit, Rules, Tools, View, Help, and a 'GET /book' item. The main window displays a list of network requests. There are four entries, each showing a timestamp, overall elapsed time, result code, protocol, host, and URL. The host for all requests is services.odata.org, and the URLs are related to the 'TripPinService/People' endpoint with different query parameters.

#	Overall_Elapsed	Result	Protocol	Host	URL
{\n} 1	0:00:00.114	200	HTTP	services.odata.org	/v4/TripPinService/People
{\n} 2	0:00:00.366	200	HTTP	services.odata.org	/v4/TripPinService/People
{\n} 3	0:00:00.038	200	HTTP	services.odata.org	/v4/TripPinService/People?%24skiptoken=8
{\n} 4	0:00:00.043	200	HTTP	services.odata.org	/v4/TripPinService/People?%24skiptoken=16

NOTE

You'll notice duplicate requests for the first page of data from the service, which is not ideal. The extra request is a result of the M engine's schema checking behavior. Ignore this issue for now and resolve it in the [next tutorial](#), where you'll apply an explicit schema.

Conclusion

This lesson showed you how to implement pagination support for a Rest API. While the logic will likely vary between APIs, the pattern established here should be reusable with minor modifications.

In the next lesson, you'll look at how to apply an explicit schema to your data, going beyond the simple `text` and `number` data types you get from `Json.Document`.

Next steps

TripPin Part 6 - Schema

TripPin Part 6 - Schema

1/15/2022 • 11 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Define a fixed schema for a REST API
- Dynamically set data types for columns
- Enforce a table structure to avoid transformation errors due to missing columns
- Hide columns from the result set

One of the big advantages of an OData service over a standard REST API is its [\\$metadata definition](#). The \$metadata document describes the data found on this service, including the schema for all of its Entities (Tables) and Fields (Columns). The `OData.Feed` function uses this schema definition to automatically set data type information—so instead of getting all text and number fields (like you would from `Json.Document`), end users will get dates, whole numbers, times, and so on, providing a better overall user experience.

Many REST APIs don't have a way to programmatically determine their schema. In these cases, you'll need to include schema definitions within your connector. In this lesson you'll define a simple, hardcoded schema for each of your tables, and enforce the schema on the data you read from the service.

NOTE

The approach described here should work for many REST services. [Future lessons](#) will build upon this approach by recursively enforcing schemas on structured columns (record, list, table), and provide sample implementations that can programmatically generate a schema table from CSDL or [JSON Schema](#) documents.

Overall, enforcing a schema on the data returned by your connector has multiple benefits, such as:

- Setting the correct data types
- Removing columns that don't need to be shown to end users (such as internal IDs or state information)
- Ensuring that each page of data has the same shape by adding any columns that might be missing from a response (a common way for REST APIs to indicate a field should be null)

Viewing the Existing Schema with Table.Schema

The connector created in the [previous lesson](#) displays three tables from the TripPin service—`Airlines`, `Airports`, and `People`. Run the following query to view the `Airlines` table:

```
let
    source = TripPin.Contents(),
    data = source{[Name="Airlines"]}[Data]
in
    data
```

In the results you'll see four columns returned:

- `@odata.id`

- @odata.editLink
 - AirlineCode
 - Name

@odata.id	@odata.editLink	AirlineCode	Name
		AA	American Airlines
		FM	Shanghai Airline
		MU	China Eastern Airlines
		AF	Air France
		AZ	Alitalia
		AC	Air Canada
		OS	Austrian Airlines
		TK	Turkish Airlines
		JL	Japan Airlines
		SQ	Singapore Airlines
		KE	Korean Air
		CZ	China Southern
		AK	AirAsia
		HX	Hong Kong Airlines
		EK	Emirates

The "@odata.*" columns are part of OData protocol, and not something you'd want or need to show to the end users of your connector. `AirlineCode` and `Name` are the two columns you'll want to keep. If you look at the schema of the table (using the handy `TableSchema` function), you can see that all of the columns in the table have a data type of `Any.Type`.

```
let
    source = TripPin.Contents(),
    data = source{[Name="Airlines"]}[Data]
in
    Table.Schema(data)
```

Name	Position	TypeName	Kind	IsNullable
@odata.id	0	Any.Type	any	<input checked="" type="checkbox"/>
@odata.editLink	1	Any.Type	any	<input checked="" type="checkbox"/>
AirlineCode	2	Any.Type	any	<input checked="" type="checkbox"/>
Name	3	Any.Type	any	<input checked="" type="checkbox"/>
				<input type="checkbox"/>

`Table.Schema` returns a lot of metadata about the columns in a table, including names, positions, type information, and many advanced properties, such as Precision, Scale, and MaxLength. Future lessons will provide design patterns for setting these advanced properties, but for now you need only concern yourself with the ascribed type (`TypeName`), primitive type (`Kind`), and whether the column value might be null (`IsNullable`).

Defining a Simple Schema Table

Your schema table will be composed of two columns:

COLUMN	DETAILS
Name	The name of the column. This must match the name in the results returned by the service.
Type	The M data type you're going to set. This can be a primitive type (<code>text</code> , <code>number</code> , <code>datetime</code> , and so on), or an ascribed type (<code>Int64.Type</code> , <code>Currency.Type</code> , and so on).

The hardcoded schema table for the `Airlines` table will set its `AirlineCode` and `Name` columns to `text`, and looks like this:

```
Airlines = #table({ "Name", "Type"}, {  
    {"AirlineCode", type text},  
    {"Name", type text}  
});
```

The `Airports` table has four fields you'll want to keep (including one of type `record`):

```
Airports = #table({ "Name", "Type"}, {  
    {"IcaoCode", type text},  
    {"Name", type text},  
    {"IataCode", type text},  
    {"Location", type record}  
});
```

Finally, the `People` table has seven fields, including lists (`Emails`, `AddressInfo`), a *nullable* column (`Gender`), and a column with an *ascribed type* (`Concurrency`).

```
People = #table({ "Name", "Type"}, {  
    {"UserName", type text},  
    {"FirstName", type text},  
    {"LastName", type text},  
    {"Emails", type list},  
    {"AddressInfo", type list},  
    {"Gender", type nullable text},  
    {"Concurrency", Int64.Type}  
})
```

The SchemaTransformTable Helper Function

The `SchemaTransformTable` helper function described below will be used to enforce schemas on your data. It takes the following parameters:

PARAMETER	TYPE	DESCRIPTION
table	table	The table of data you'll want to enforce your schema on.
schema	table	The schema table to read column information from, with the following type: <pre>type table [Name = text, Type = type]</pre>

PARAMETER	TYPE	DESCRIPTION
enforceSchema	number	<p>(optional) An enum that controls behavior of the function.</p> <p>The default value (<code>EnforceSchema.Strict = 1</code>) ensures that the output table will match the schema table that was provided by adding any missing columns, and removing extra columns.</p> <p>The <code>EnforceSchema.IgnoreExtraColumns = 2</code> option can be used to preserve extra columns in the result.</p> <p>When <code>EnforceSchema.IgnoreMissingColumns = 3</code> is used, both missing columns and extra columns will be ignored.</p>

The logic for this function looks something like this:

1. Determine if there are any missing columns from the source table.
2. Determine if there are any extra columns.
3. Ignore structured columns (of type `list`, `record`, and `table`), and columns set to `type any`.
4. Use `Table.TransformColumnTypes` to set each column type.
5. Reorder columns based on the order they appear in the schema table.
6. Set the type on the table itself using `Value.ReplaceType`.

NOTE

The last step to set the table type will remove the need for the Power Query UI to infer type information when viewing the results in the query editor. This removes the double request issue you saw at the [end of the previous tutorial](#).

The following helper code can be copy and pasted into your extension:

```

EnforceSchema.Strict = 1;           // Add any missing columns, remove extra columns, set table type
EnforceSchema.IgnoreExtraColumns = 2; // Add missing columns, do not remove extra columns
EnforceSchema.IgnoreMissingColumns = 3; // Do not add or remove columns

SchemaTransformTable = (table as table, schema as table, optional enforceSchema as number) as table =>
    let
        // Default to EnforceSchema.Strict
        _enforceSchema = if (enforceSchema <> null) then enforceSchema else EnforceSchema.Strict,

        // Applies type transforms to a given table
        EnforceTypes = (table as table, schema as table) as table =>
            let
                map = (t) => if Type.Is(t, type list) or Type.Is(t, type record) or t = type any then null
            else t,
                mapped = Table.TransformColumns(schema, {"Type", map}),
                omitted = Table.SelectRows(mapped, each [Type] <> null),
                existingColumns = Table.ColumnNames(table),
                removeMissing = Table.SelectRows(omitted, each List.Contains(existingColumns, [Name])),
                primitiveTransforms = Table.ToRows(removeMissing),
                changedPrimitives = Table.TransformColumnTypes(table, primitiveTransforms)
            in
                changedPrimitives,

            // Returns the table type for a given schema
            SchemaToTableType = (schema as table) as type =>
                let
                    toList = List.Transform(schema[Type], (t) => [Type=t, Optional=false]),
                    toRecord = Record.FromList(toList, schema[Name]),
                    toType = Type.ForRecord(toRecord, false)
                in
                    type table (toType),

            // Determine if we have extra/missing columns.
            // The enforceSchema parameter determines what we do about them.
            schemaNames = schema[Name],
            foundNames = Table.ColumnNames(table),
            addNames = List.RemoveItems(schemaNames, foundNames),
            extraNames = List.RemoveItems(foundNames, schemaNames),
            tmp = Text.NewGuid(),
            added = Table.AddColumn(table, tmp, each []),
            expanded = Table.ExpandRecordColumn(added, tmp, addNames),
            result = if List.IsEmpty(addNames) then table else expanded,
            fullList =
                if (_enforceSchema = EnforceSchema.Strict) then
                    schemaNames
                else if (_enforceSchema = EnforceSchema.IgnoreMissingColumns) then
                    foundNames
                else
                    schemaNames & extraNames,

            // Select the final list of columns.
            // These will be ordered according to the schema table.
            reordered = Table.SelectColumns(result, fullList, MissingField.Ignore),
            enforcedTypes = EnforceTypes(reordered, schema),
            withType = if (_enforceSchema = EnforceSchema.Strict) then Value.ReplaceType(enforcedTypes,
SchemaToTableType(schema)) else enforcedTypes
        in
            withType;

```

Updating the TripPin Connector

You'll now make the following changes to your connector to make use of the new schema enforcement code.

1. Define a master schema table (`SchemaTable`) that holds all of your schema definitions.
2. Update the `TripPin.Feed`, `GetPage`, and `GetAllPagesByNextLink` to accept a `schema` parameter.

3. Enforce your schema in `GetPage`.
4. Update your navigation table code to wrap each table with a call to a new function (`GetEntity`)—this will give you more flexibility to manipulate the table definitions in the future.

Master schema table

You'll now consolidate your schema definitions into a single table, and add a helper function (`GetSchemaForEntity`) that lets you look up the definition based on an entity name (for example, `GetSchemaForEntity("Airlines")`)

```
SchemaTable = #table({{"Entity", "SchemaTable"}, {
    {"Airlines", #table({{"Name", "Type"}, {
        {"AirlineCode", type text},
        {"Name", type text}
    }}),
    {"Airports", #table({{"Name", "Type"}, {
        {"IcaoCode", type text},
        {"Name", type text},
        {"IataCode", type text},
        {"Location", type record}
    }}),
    {"People", #table({{"Name", "Type"}, {
        {"UserName", type text},
        {"FirstName", type text},
        {"LastName", type text},
        {"Emails", type list},
        {"AddressInfo", type list},
        {"Gender", type nullable text},
        {"Concurrency", Int64.Type}
    }})
});;

GetSchemaForEntity = (entity as text) as table => try SchemaTable[[Entity=entity]][SchemaTable] otherwise
error "Couldn't find entity: '" & entity & "'";
```

Adding schema support to data functions

You'll now add an optional `schema` parameter to the `TripPin.Feed`, `GetPage`, and `GetAllPagesByNextLink` functions. This will allow you to pass down the schema (when you want to) to the paging functions, where it will be applied to the results you get back from the service.

```
TripPin.Feed = (url as text, optional schema as table) as table => ...
GetPage = (url as text, optional schema as table) as table => ...
GetAllPagesByNextLink = (url as text, optional schema as table) as table => ...
```

You'll also update all of the calls to these functions to make sure that you pass the schema through correctly.

Enforcing the schema

The actual schema enforcement will be done in your `GetPage` function.

```

GetPage = (url as text, optional schema as table) as table =>
    let
        response = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
        body = Json.Document(response),
        nextLink = GetNextLink(body),
        data = Table.FromRecords(body[value]),
        // enforce the schema
        withSchema = if (schema <> null) then SchemaTransformTable(data, schema) else data
    in
        withSchema meta [NextLink = nextLink];

```

[Note] This `GetPage` implementation uses `Table.FromRecords` to convert the list of records in the JSON response to a table. A major downside to using `Table.FromRecords` is that it assumes all records in the list have the same set of fields. This works for the TripPin service, since the OData records are guaranteed to contain the same fields, but this might not be the case for all REST APIs. A more robust implementation would use a combination of `Table.FromList` and `Table.ExpandRecordColumn`. Later tutorials will change the implementation to get the column list from the schema table, ensuring that no columns are lost or missing during the JSON to M translation.

Adding the `GetEntity` function

The `GetEntity` function will wrap your call to `TripPin.Feed`. It will look up a schema definition based on the entity name, and build the full request URL.

```

GetEntity = (url as text, entity as text) as table =>
    let
        fullUrl = Uri.Combine(url, entity),
        schemaTable = GetSchemaForEntity(entity),
        result = TripPin.Feed(fullUrl, schemaTable)
    in
        result;

```

You'll then update your `TripPinNavTable` function to call `GetEntity`, rather than making all of the calls inline. The main advantage to this is that it will let you continue modifying your entity building code, without having to touch your nav table logic.

```

TripPinNavTable = (url as text) as table =>
    let
        entitiesAsTable = Table.FromList(RootEntities, Splitter.SplitByNothing()),
        rename = Table.RenameColumns(entitiesAsTable, {[{"Column1", "Name"}]}),
        // Add Data as a calculated column
        withData = Table.AddColumn(rename, "Data", each GetEntity(url, [Name]), type table),
        // Add ItemKind and ItemName as fixed text values
        withItemKind = Table.AddColumn(withData, "ItemKind", each "Table", type text),
        withItemName = Table.AddColumn(withItemKind, "ItemName", each "Table", type text),
        // Indicate that the node should not be expandable
        withIsLeaf = Table.AddColumn(withItemName, "IsLeaf", each true, type logical),
        // Generate the nav table
        navTable = Table.ToNavigationTable(withIsLeaf, {"Name"}, "Name", "Data", "ItemKind", "ItemName",
        "IsLeaf")
    in
        navTable;

```

Putting it all Together

Once all of the code changes are made, compile and re-run the test query that calls `Table.Schema` for the Airlines table.

```

let
    source = TripPin.Contents(),
    data = source{[Name="Airlines"]}[Data]
in
    Table.Schema(data)

```

You now see that your Airlines table only has the two columns you defined in its schema:

Name	Position	TypeName	Kind	IsNullable
AirlineCode	0	Text.Type	text	<input type="checkbox"/>
Name	1	Text.Type	text	<input type="checkbox"/>
				<input type="checkbox"/>

If you run the same code against the People table...

```

let
    source = TripPin.Contents(),
    data = source{[Name="People"]}[Data]
in
    Table.Schema(data)

```

You'll see that the ascribed type you used (`Int64.Type`) was also set correctly.

Name	Position	TypeName	Kind	IsNullable
UserName	0	Text.Type	text	<input type="checkbox"/>
FirstName	1	Text.Type	text	<input type="checkbox"/>
LastName	2	Text.Type	text	<input type="checkbox"/>
Emails	3	List.Type	list	<input type="checkbox"/>
AddressInfo	4	List.Type	list	<input type="checkbox"/>
Gender	5	Text.Type	text	<input checked="" type="checkbox"/>
Concurrency	6	Int64.Type	number	<input type="checkbox"/>
				<input type="checkbox"/>

An important thing to note is that this implementation of `SchemaTransformTable` doesn't modify the types of `list` and `record` columns, but the `Emails` and `AddressInfo` columns are still typed as `list`. This is because `Json.Document` will correctly map JSON arrays to M lists, and JSON objects to M records. If you were to expand the list or record column in Power Query, you'd see that all of the expanded columns will be of type any. Future tutorials will improve the implementation to recursively set type information for nested complex types.

Conclusion

This tutorial provided a sample implementation for enforcing a schema on JSON data returned from a REST service. While this sample uses a simple hardcoded schema table format, the approach could be expanded upon by dynamically building a schema table definition from another source, such as a JSON schema file, or metadata service/endpoint exposed by the data source.

In addition to modifying column types (and values), your code is also setting the correct type information on the table itself. Setting this type information benefits performance when running inside of Power Query, as the user experience always attempts to infer type information to display the right UI queues to the end user, and the inference calls can end up triggering additional calls to the underlying data APIs.

If you view the People table using the [TripPin connector from the previous lesson](#), you'll see that all of the columns have a 'type any' icon (even the columns that contain lists):

		ABC 123 @odata.editLink	ABC 123 UserName	ABC 123 FirstName	ABC 123 LastName	ABC 123 Emails	ABC 123 AddressInfo	ABC 123 Gender	ABC 123 Concurrency
1	8951"	http://services.odata.org/V4/TripPinService/People('russellwhyte')	russellwhyte	Russell	Whyte	List	List	Male	6.36389E+17
2	8951"	http://services.odata.org/V4/TripPinService/People('scottketchum')	scottketchum	Scott	Ketchum	List	List	Male	6.36389E+17
3	8951"	http://services.odata.org/V4/TripPinService/People('ronaldmundy')	ronaldmundy	Ronald	Mundy	List	List	Male	6.36389E+17
4	8951"	http://services.odata.org/V4/TripPinService/People('javieralfred')	javieralfred	Javier	Alfred	List	List	Male	6.36389E+17
5	8951"	http://services.odata.org/V4/TripPinService/People('willieashmore')	willieashmore	Willie	Ashmore	List	List	Male	6.36389E+17
6	8951"	http://services.odata.org/V4/TripPinService/People('vincentcalabrese')	vincentcalabrese	Vincent	Calabrese	List	List	Male	6.36389E+17
7	8951"	http://services.odata.org/V4/TripPinService/People('clydegues')	clydegues	Clyde	Guess	List	List	Male	6.36389E+17
8	8951"	http://services.odata.org/V4/TripPinService/People('keithpinckney')	keithpinckney	Keith	Pinckney	List	List	Male	6.36389E+17
9	8951"	http://services.odata.org/V4/TripPinService/People('marshallgaray')	marshallgaray	Marshall	Garay	List	List	Male	6.36389E+17
10	8951"	http://services.odata.org/V4/TripPinService/People('ryantheriault')	ryantheriault	Ryan	Theriault	List	List	Male	6.36389E+17
11	8951"	http://services.odata.org/V4/TripPinService/People('elainestewart')	elainestewart	Elaine	Stewart	List	List	Female	6.36389E+17
12	8951"	http://services.odata.org/V4/TripPinService/People('salliesampson')	salliesampson	Sallie	Sampson	List	List	Female	6.36389E+17
13	8951"	http://services.odata.org/V4/TripPinService/People('jonirosales')	jonirosales	Joni	Rosales	List	List	Female	6.36389E+17
14	8951"	http://services.odata.org/V4/TripPinService/People('georginabarlow')	georginabarlow	Georgina	Barlow	List	List	Female	6.36389E+17
15	8951"	http://services.odata.org/V4/TripPinService/People('angelhuffman')	angelhuffman	Angel	Huffman	List	List	Female	6.36389E+17
16	8951"	http://services.odata.org/V4/TripPinService/People('laurelosborn')	laurelosborn	Laurel	Osborn	List	List	Female	6.36389E+17
17	8951"	http://services.odata.org/V4/TripPinService/People('sandyosborn')	sandyosborn	Sandy	Osborn	List	List	Female	6.36389E+17
18	8951"	http://services.odata.org/V4/TripPinService/People('ursulabright')	ursulabright	Ursula	Bright	List	List	Female	6.36389E+17
19	8951"	http://services.odata.org/V4/TripPinService/People('genevieveerees')	genevieveerees	Genevieve	Reeves	List	List	Female	6.36389E+17
20	8951"	http://services.odata.org/V4/TripPinService/People('kristakemp')	kristakemp	Krista	Kemp	List	List	Female	6.36389E+17

Running the same query with the TripPin connector from this lesson, you'll now see that the type information is displayed correctly.

		= Source{[Name="People"]}[Data]	ABC 123 UserName	ABC 123 FirstName	ABC 123 LastName	Emails	AddressInfo	ABC 123 Gender	ABC 123 Concurrency
1	russellwhyte	Russell	Whyte	List	List	Male	6.36389E+17		
2	scottketchum	Scott	Ketchum	List	List	Male	6.36389E+17		
3	ronaldmundy	Ronald	Mundy	List	List	Male	6.36389E+17		
4	javieralfred	Javier	Alfred	List	List	Male	6.36389E+17		
5	willieashmore	Willie	Ashmore	List	List	Male	6.36389E+17		
6	vincentcalabrese	Vincent	Calabrese	List	List	Male	6.36389E+17		
7	clydegues	Clyde	Guess	List	List	Male	6.36389E+17		
8	keithpinckney	Keith	Pinckney	List	List	Male	6.36389E+17		
9	marshallgaray	Marshall	Garay	List	List	Male	6.36389E+17		
10	ryantheriault	Ryan	Theriault	List	List	Male	6.36389E+17		
11	elainestewart	Elaine	Stewart	List	List	Female	6.36389E+17		
12	salliesampson	Sallie	Sampson	List	List	Female	6.36389E+17		
13	jonirosales	Joni	Rosales	List	List	Female	6.36389E+17		
14	georginabarlow	Georgina	Barlow	List	List	Female	6.36389E+17		
15	angelhuffman	Angel	Huffman	List	List	Female	6.36389E+17		
16	laurelosborn	Laurel	Osborn	List	List	Female	6.36389E+17		
17	sandyosborn	Sandy	Osborn	List	List	Female	6.36389E+17		
18	ursulabright	Ursula	Bright	List	List	Female	6.36389E+17		
19	genevieveerees	Genevieve	Reeves	List	List	Female	6.36389E+17		
20	kristakemp	Krista	Kemp	List	List	Female	6.36389E+17		

Next steps

[TripPin Part 7 - Advanced Schema with M Types](#)

TripPin Part 7 - Advanced Schema with M Types

1/15/2022 • 7 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Enforce a table schema using [M Types](#)
- Set types for nested records and lists
- Refactor code for reuse and unit testing

In the previous lesson you defined your table schemas using a simple "Schema Table" system. This schema table approach works for many REST APIs/Data Connectors, but services that return complete or deeply nested data sets might benefit from the approach in this tutorial, which leverages the [M type system](#).

This lesson will guide you through the following steps:

1. Adding unit tests
2. Defining custom M types
3. Enforcing a schema using types
4. Refactoring common code into separate files

Adding Unit Tests

Before you start making use of the advanced schema logic, you'll add a set of unit tests to your connector to reduce the chance of inadvertently breaking something. Unit testing works like this:

1. Copy the common code from the [UnitTest sample](#) into your `TripPin.query.pq` file
2. Add a section declaration to the top of your `TripPin.query.pq` file
3. Create a *shared* record (called `TripPin.UnitTest`)
4. Define a `Fact` for each test
5. Call `Facts.Summarize()` to run all of the tests
6. Reference the previous call as the shared value to ensure that it gets evaluated when the project is run in Visual Studio

```

section TripPinUnitTests;

shared TripPin.UnitTesting =
[
    // Put any common variables here if you only want them to be evaluated once
    RootTable = TripPin.Contents(),
    Airlines = RootTable{[Name="Airlines"]}[Data],
    Airports = RootTable{[Name="Airports"]}[Data],
    People = RootTable{[Name="People"]}[Data],

    // Fact(<Name of the Test>, <Expected Value>, <Actual Value>)
    // <Expected Value> and <Actual Value> can be a literal or let statement
    facts =
    {
        Fact("Check that we have three entries in our nav table", 3, Table.RowCount(RootTable)),
        Fact("We have Airline data?", true, not Table.IsEmpty(Airlines)),
        Fact("We have People data?", true, not Table.IsEmpty(People)),
        Fact("We have Airport data?", true, not Table.IsEmpty(Airports)),
        Fact("Airlines only has 2 columns", 2, List.Count(Table.ColumnNames(Airlines))),
        Fact("Airline table has the right fields",
            {"AirlineCode", "Name"}, Record.FieldNames(Type.RecordFields(Type.TableRow(Value.Type(Airlines)))))
    }
},
report = Facts.Summarize(facts)
][report];

```

Clicking run on the project will evaluate all of the Facts, and give you a report output that looks like this:

Query Result		
Result	Notes	Details
Success	All 6 Passed !!! ✓	100% success rate
Success ✓	Check that we have three entries in our nav table	(3 = 3)
Success ✓	We have Airline data?	(true = true)
Success ✓	We have People data?	(true = true)
Success ✓	We have Airport data?	(true = true)
Success ✓	Airlines only has 2 columns	(2 = 2)
Success ✓	Airline table has the right fields	({"AirlineCode", "Name"} = {"AirlineCode", "Name"})

Using some principles from [test-driven development](#), you'll now add a test that currently fails, but will soon be reimplemented and fixed (by the end of this tutorial). Specifically, you'll add a test that checks one of the nested records (Emails) you get back in the People entity.

```
Fact("Emails is properly typed", type text, Type.ListItem(Value.Type(People{0}[Emails])))
```

If you run the code again, you should now see that you have a failing test.

Result	Notes	Details
Success ✓	Check that we have three entries in our nav table	(3 = 3)
Success ✓	We have Airline data?	(true = true)
Success ✓	We have People data?	(true = true)
Success ✓	We have Airport data?	(true = true)
Success ✓	Airlines only has 2 columns	(2 = 2)
Success ✓	Airline table has the right fields	({"AirlineCode", "Name"} = {"AirlineCode", "Name"})
Failure ⚡	Emails is properly typed	(type text <> type any)

Now you just need to implement the functionality to make this work.

Defining Custom M Types

The schema enforcement approach in the [previous lesson](#) used "schema tables" defined as Name/Type pairs. It works well when working with flattened/relational data, but didn't support setting types on nested records/tables/lists, or allow you to reuse type definitions across tables/entities.

In the TripPin case, the data in the People and Airports entities contain structured columns, and even share a type (`Location`) for representing address information. Rather than defining Name/Type pairs in a schema table, you'll define each of these entities using custom M type declarations.

Here is a quick refresher about types in the M language from the [Language Specification](#):

A **type value** is a value that **classifies** other values. A value that is classified by a type is said to **conform** to that type. The M type system consists of the following kinds of types:

- Primitive types, which classify primitive values (`binary`, `date`, `datetime`, `datetimezone`, `duration`, `list`, `logical`, `null`, `number`, `record`, `text`, `time`, `type`) and also include a number of abstract types (`function`, `table`, `any`, and `none`)
- Record types, which classify record values based on field names and value types
- List types, which classify lists using a single item base type
- Function types, which classify function values based on the types of their parameters and return values
- Table types, which classify table values based on column names, column types, and keys
- Nullable types, which classifies the value `null` in addition to all the values classified by a base type
- Type types, which classify values that are types

Using the raw JSON output you get (and/or looking up the definitions in the service's `$metadata`), you can define the following record types to represent OData complex types:

```

LocationType = type [
    Address = text,
    City = CityType,
    Loc = LocType
];

CityType = type [
    CountryRegion = text,
    Name = text,
    Region = text
];

LocType = type [
    #"type" = text,
    coordinates = {number},
    crs = CrsType
];

CrsType = type [
    #"type" = text,
    properties = record
];

```

Note how the `LocationType` references the `CityType` and `LocType` to represent its structured columns.

For the top level entities (that you want represented as Tables), you define *table types*:

```

AirlinesType = type table [
    AirlineCode = text,
    Name = text
];

AirportsType = type table [
    Name = text,
    IataCode = text,
    Location = LocationType
];

PeopleType = type table [
    UserName = text,
    FirstName = text,
    LastName = text,
    Emails = {text},
    AddressInfo = {nullable LocationType},
    Gender = nullable text,
    Concurrency = Int64.Type
];

```

You then update your `SchemaTable` variable (which you use as a "lookup table" for entity to type mappings) to use these new type definitions:

```

SchemaTable = #table({{"Entity", "Type"}, {
    {"Airlines", AirlinesType },
    {"Airports", AirportsType },
    {"People", PeopleType}
}});

```

Enforcing a Schema Using Types

You'll rely on a common function (`Table.ChangeType`) to enforce a schema on your data, much like you used `SchemaTransformTable` in the [previous lesson](#). Unlike `SchemaTransformTable`, `Table.ChangeType` takes in an actual

M table type as an argument, and will apply your schema *recursively* for all nested types. Its signature looks like this:

```
Table.ChangeType = (table, tableType as type) as nullable table => ...
```

The full code listing for the `Table.ChangeType` function can be found in the [Table.ChangeType.pqm](#) file.

NOTE

For flexibility, the function can be used on tables, as well as lists of records (which is how tables would be represented in a JSON document).

You then need to update the connector code to change the `schema` parameter from a `table` to a `type`, and add a call to `Table.ChangeType` in `GetEntity`.

```
GetEntity = (url as text, entity as text) as table =>
    let
        fullUrl = Uri.Combine(url, entity),
        schema = GetSchemaForEntity(entity),
        result = TripPin.Feed(fullUrl, schema),
        appliedSchema = Table.ChangeType(result, schema)
    in
        appliedSchema;
```

`GetPage` is updated to use the list of fields from the schema (to know the names of what to expand when you get the results), but leaves the actual schema enforcement to `GetEntity`.

```
GetPage = (url as text, optional schema as type) as table =>
    let
        response = Web.Contents(url, [ Headers = DefaultRequestHeaders ]),
        body = Json.Document(response),
        nextLink = GetNextLink(body),

        // If we have no schema, use Table.FromRecords() instead
        // (and hope that our results all have the same fields).
        // If we have a schema, expand the record using its field names
        data =
            if (schema <> null) then
                Table.FromRecords(body[value])
            else
                let
                    // convert the list of records into a table (single column of records)
                    asTable = Table.FromList(body[value], Splitter.SplitByNothing(), {"Column1"}),
                    fields = Record.FieldNames(Type.RecordFields(Type.TableRow(schema))),
                    expanded = Table.ExpandRecordColumn(asTable, fields)
                in
                    expanded
    in
        data meta [NextLink = nextLink];
```

Confirming that nested types are being set

The definition for your `PeopleType` now sets the `Emails` field to a list of text (`{text}`). If you're applying the types correctly, the call to `Type.ListItem` in your unit test should now be returning `type text` rather than `type any`.

Running your unit tests again show that they are now all passing.

Query Result		
Result	Notes	Details
Success	All 7 Passed !!! ✓	100% success rate
Success ✓	Check that we have three entries in our nav table	(3 = 3)
Success ✓	We have Airline data?	(true = true)
Success ✓	We have People data?	(true = true)
Success ✓	We have Airport data?	(true = true)
Success ✓	Airlines only has 2 columns	(2 = 2)
Success ✓	Airline table has the right fields	({"AirlineCode", "Name"} = {"AirlineCode", "Name"})
Success ✓	Emails is properly typed	(type text = type text)

Refactoring Common Code into Separate Files

NOTE

The M engine will have improved support for referencing external modules/common code in the future, but this approach should carry you through until then.

At this point, your extension almost has as much "common" code as TripPin connector code. In the future these **common functions** will either be part of the built-in standard function library, or you'll be able to reference them from another extension. For now, you refactor your code in the following way:

1. Move the reusable functions to separate files (.pqm).
2. Set the **Build Action** property on the file to **Compile** to make sure it gets included in your extension file during the build.
3. Define a function to load the code using [Expression.Evaluate](#).
4. Load each of the common functions you want to use.

The code to do this is included in the snippet below:

```
Extension.LoadFunction = (name as text) =>
    let
        binary = Extension.Contents(name),
        asText = Text.FromBinary(binary)
    in
        Expression.Evaluate(asText, #shared);

Table.ChangeType = Extension.LoadFunction("Table.ChangeType.pqm");
Table.GenerateByPage = Extension.LoadFunction("Table.GenerateByPage.pqm");
Table.ToNavigationTable = Extension.LoadFunction("Table.ToNavigationTable.pqm");
```

Conclusion

This tutorial made a number of improvements to the way you enforce a schema on the data you get from a REST API. The connector is currently hard coding its schema information, which has a performance benefit at runtime, but is unable to adapt to changes in the service's metadata overtime. Future tutorials will move to a purely dynamic approach that will infer the schema from the service's \$metadata document.

In addition to the schema changes, this tutorial added Unit Tests for your code, and refactored the common helper functions into separate files to improve overall readability.

Next steps

[TripPin Part 8 - Adding Diagnostics](#)

TripPin Part 8 - Adding Diagnostics

1/15/2022 • 8 minutes to read • [Edit Online](#)

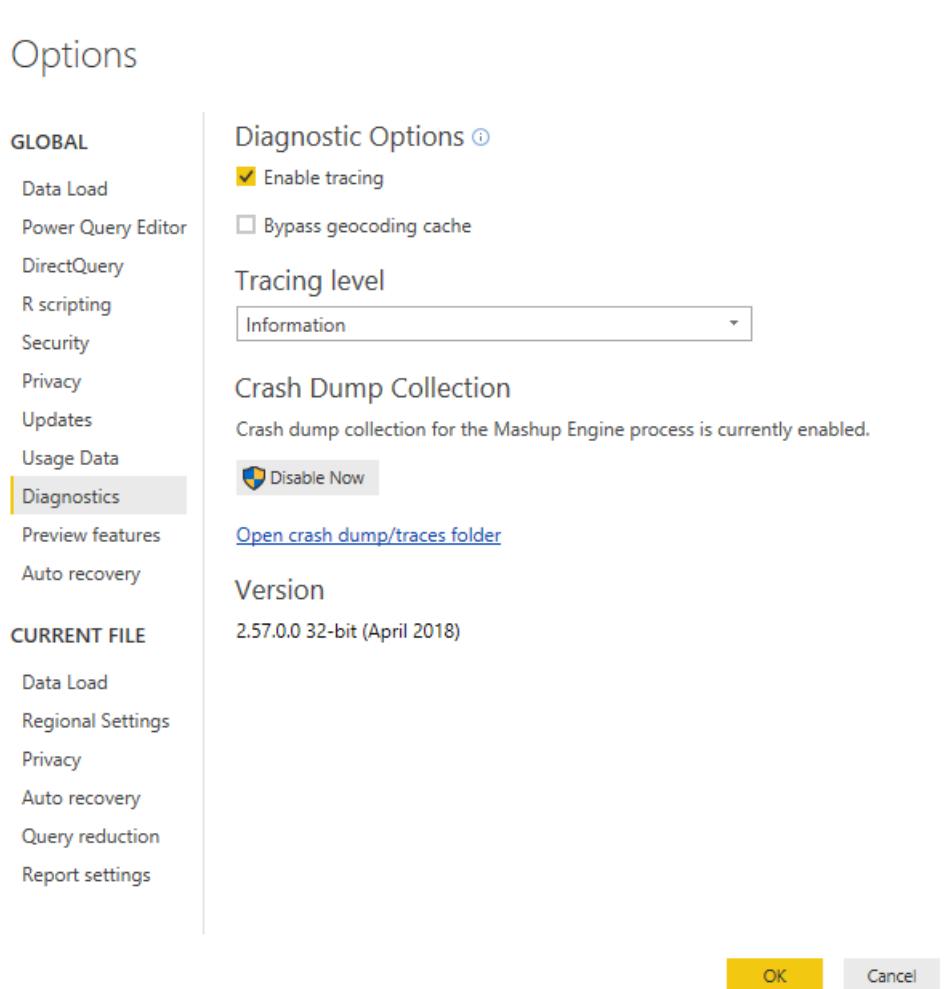
This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Learn about the [Diagnostics.Trace](#) function
- Use the Diagnostics helper functions to add trace information to help debug your connector

Enabling diagnostics

Power Query users can enable trace logging by selecting the checkbox under Options | Diagnostics.

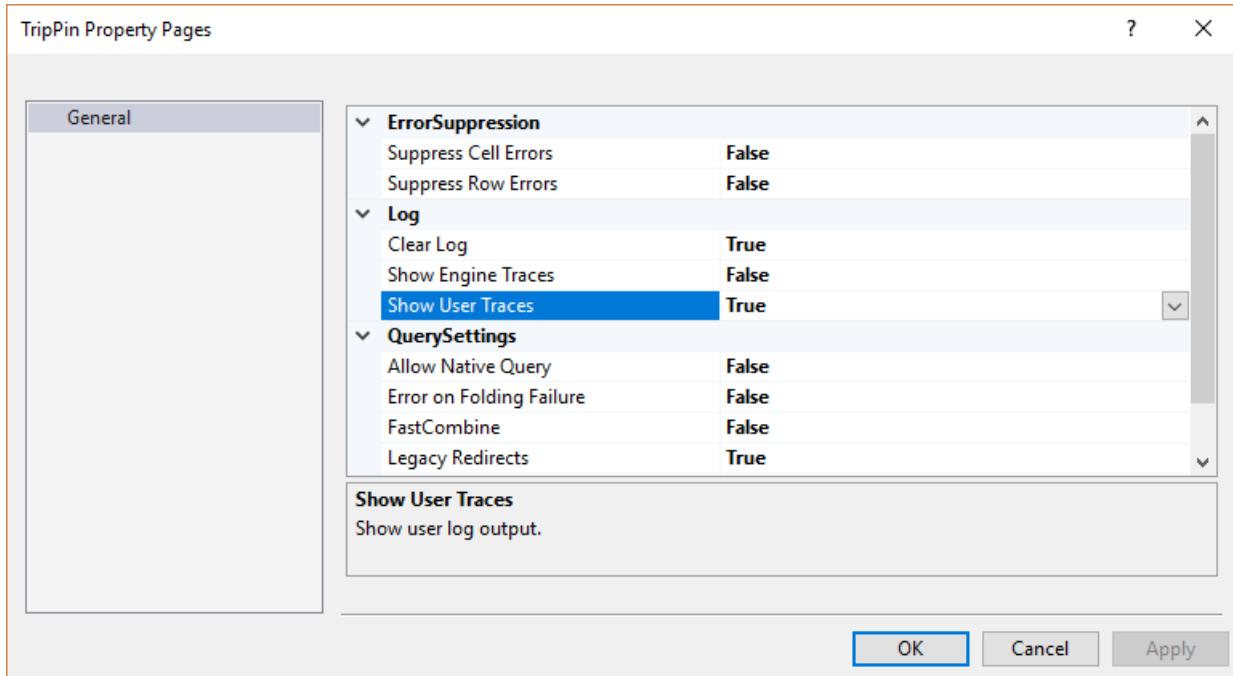


Once enabled, any subsequent queries will cause the M engine to emit trace information to log files located in a fixed user directory.

When running M queries from within the Power Query SDK, tracing is enabled at the project level. On the project properties page, there are three settings related to tracing:

- **Clear Log**—when this is set to `true`, the log will be reset/cleared when you run your queries. We recommend you keep this set to `true`.

- **Show Engine Traces**—this setting controls the output of built-in traces from the M engine. These traces are generally only useful to members of the Power Query team, so you'll typically want to keep this set to `false`.
- **Show User Traces**—this setting controls trace information output by your connector. You'll want to set this to `true`.



Once enabled, you'll start seeing log entries in the M Query Output window, under the Log tab.

Diagnostics.Trace

The [Diagnostics.Trace](#) function is used to write messages into the M engine's trace log.

```
Diagnostics.Trace = (traceLevel as number, message as text, value as any, optional delayed as nullable logical as any) => ...
```

IMPORTANT

M is a functional language with lazy evaluation. When using `Diagnostics.Trace`, keep in mind that the function will only be called if the expression it's a part of is actually evaluated. Examples of this can be found later in this tutorial.

The `traceLevel` parameter can be one of the following values (in descending order):

- `TraceLevel.Critical`
- `TraceLevel.Error`
- `TraceLevel.Warning`
- `TraceLevel.Information`
- `TraceLevel.Verbose`

When tracing is enabled, the user can select the maximum level of messages they would like to see. All trace messages of this level and under will be output to the log. For example, if the user selects the "Warning" level, trace messages of `TraceLevel.Warning`, `TraceLevel.Error`, and `TraceLevel.Critical` would appear in the logs.

The `message` parameter is the actual text that will be output to the trace file. Note that the text will not contain the `value` parameter unless you explicitly include it in the text.

The `value` parameter is what the function will return. When the `delayed` parameter is set to `true`, `value` will

be a zero parameter function that returns the actual value you're evaluating. When `delayed` is set to `false`, `value` will be the actual value. An example of how this works can be [found below](#).

Using `Diagnostics.Trace` in the TripPin connector

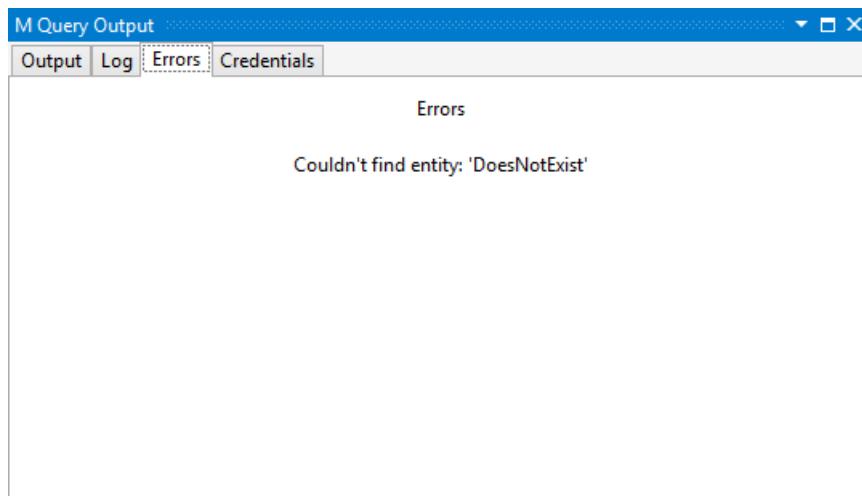
For a practical example of using `Diagnostics.Trace` and the impact of the `delayed` parameter, update the TripPin connector's `GetSchemaForEntity` function to wrap the `error` exception:

```
GetSchemaForEntity = (entity as text) as type =>
    try
        SchemaTable{[Entity=entity]}[Type]
    otherwise
        let
            message = Text.Format("Couldn't find entity: '#{0}'", {entity})
        in
            Diagnostics.Trace(TraceLevel.Error, message, () => error message, true);
```

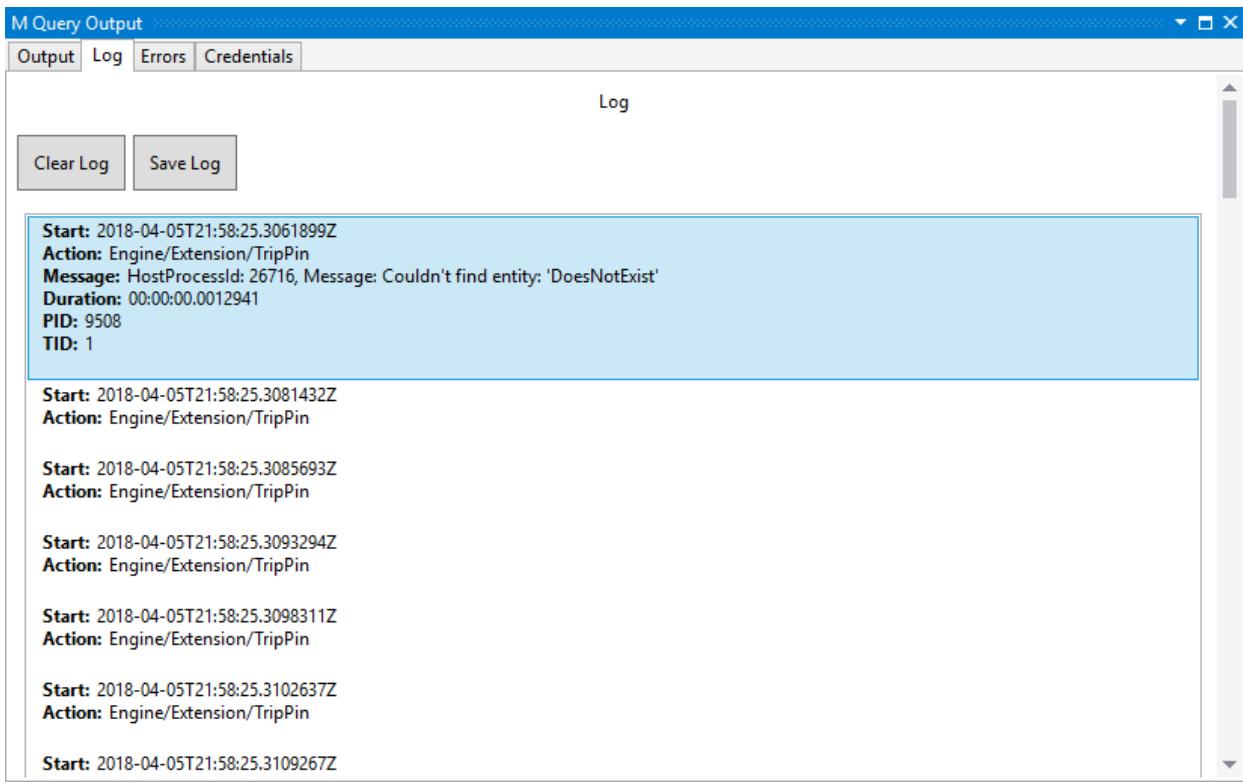
You can force an error during evaluation (for test purposes!) by passing an invalid entity name to the `GetEntity` function. Here you change the `withData` line in the `TripPinNavTable` function, replacing `[Name]` with `"DoesNotExist"`.

```
TripPinNavTable = (url as text) as table =>
    let
        // Use our schema table as the source of top level items in the navigation tree
        entities = Table.SelectColumns(SchemaTable, {"Entity"}),
        rename = Table.RenameColumns(entities, {[{"Entity", "Name"}]}),
        // Add Data as a calculated column
        withData = Table.AddColumn(rename, "Data", each GetEntity(url, "DoesNotExist"), type table),
        // Add ItemKind and ItemName as fixed text values
        withItemKind = Table.AddColumn(withData, "ItemKind", each "Table", type text),
        withItemName = Table.AddColumn(withItemKind, "ItemName", each "Table", type text),
        // Indicate that the node should not be expandable
        withIsLeaf = Table.AddColumn(withItemName, "IsLeaf", each true, type logical),
        // Generate the nav table
        navTable = Table.ToNavigationTable(withIsLeaf, {"Name"}, "Name", "Data", "ItemKind", "ItemName",
        "IsLeaf")
    in
        navTable;
```

Enable tracing for your project, and run your test queries. On the `Errors` tab you should see the text of the error you raised:



Also, on the `Log` tab, you should see the same message. Note that if you use different values for the `message` and `value` parameters, these would be different.



Also note that the `Action` field of the log message contains the name (Data Source Kind) of your extension (in this case, `Engine/Extension/TripPin`). This makes it easier to find the messages related to your extension when there are multiple queries involved and/or system (mashup engine) tracing is enabled.

Delayed evaluation

As an example of how the `delayed` parameter works, you'll make some modifications and run the queries again.

First, set the `delayed` value to `false`, but leave the `value` parameter as-is:

```
Diagnostics.Trace(TraceLevel.Error, message, () => error message, false);
```

When you run the query, you'll receive an error that "We cannot convert a value of type Function to type Type", and not the actual error you raised. This is because the call is now returning a `function` value, rather than the value itself.

Next, remove the function from the `value` parameter:

```
Diagnostics.Trace(TraceLevel.Error, message, error message, false);
```

When you run the query, you'll receive the correct error, but if you check the **Log** tab, there will be no messages. This is because the `error` ends up being raised/evaluated *during* the call to `Diagnostics.Trace`, so the message is never actually output.

Now that you understand the impact of the `delayed` parameter, be sure to reset your connector back to a working state before proceeding.

Diagnostic helper functions in `Diagnostics.pqm`

The `Diagnostics.pqm` file included in this project contains a number of helper functions that make tracing easier. As shown in the [previous tutorial](#), you can include this file in your project (remembering to set the Build Action to *Compile*), and then load it in your connector file. The bottom of your connector file should now look something like the code snippet below. Feel free to explore the various functions this module provides, but in

this sample, you'll only be using the `Diagnostics.LogValue` and `Diagnostics.LogFailure` functions.

```
// Diagnostics module contains multiple functions. We can take the ones we need.  
Diagnostics = Extension.LoadFunction("Diagnostics.pqm");  
Diagnostics.LogValue = Diagnostics[LogValue];  
Diagnostics.LogFailure = Diagnostics[LogFailure];
```

Diagnostics.LogValue

The `Diagnostics.LogValue` function is a lot like `Diagnostics.Trace`, and can be used to output the value of what you're evaluating.

```
Diagnostics.LogValue = (prefix as text, value as any) as any => ...
```

The `prefix` parameter is prepended to the log message. You'd use this to figure out which call output the message. The `value` parameter is what the function will return, and will also be written to the trace as a text representation of the M value. For example, if `value` is equal to a `table` with columns A and B, the log will contain the equivalent `#table` representation:

```
#table({{"A", "B"}, {"row1 A", "row1 B"}, {"row2 A", "row2 B"}})
```

NOTE

Serializing M values to text can be an expensive operation. Be aware of the potential size of the values you are outputting to the trace.

NOTE

Most Power Query environments will truncate trace messages to a maximum length.

As an example, you'll update the `TripPin.Feed` function to trace the `url` and `schema` arguments passed into the function.

```
TripPin.Feed = (url as text, optional schema as type) as table =>  
let  
    _url = Diagnostics.LogValue("Accessing url", url),  
    _schema = Diagnostics.LogValue("Schema type", schema),  
    //result = GetAllPagesByNextLink(url, schema)  
    result = GetAllPagesByNextLink(_url, _schema)  
in  
    result;
```

Note that you have to use the new `_url` and `_schema` values in the call to `GetAllPagesByNextLink`. If you used the original function parameters, the `Diagnostics.LogValue` calls would never actually be evaluated, resulting in no messages written to the trace. *Functional programming is fun!*

When you run your queries, you should now see new messages in the log.

Accessing url:

M Query Output

Output Log Errors Credentials

Log

Clear Log **Save Log**

```

Start: 2018-04-05T21:01:25.1781124Z
Action: Engine/Extension/TripPin
Message: HostProcessId: 26716, Message: Accessing url: "http://services.odata.org/v4/TripPinService/Airlines"
Duration: 00:00:00.0000240
PID: 7796
TID: 1

Start: 2018-04-05T21:01:25.4088308Z
Action: Engine/Extension/TripPin

Start: 2018-04-05T21:01:25.5058279Z
Action: Engine/Extension/TripPin

```

Schema type:

M Query Output

Output Log Errors Credentials

Log

Clear Log **Save Log**

```

Start: 2018-04-06T15:48:31.1976744Z
Action: Engine/Extension/TripPin

Start: 2018-04-06T15:48:31.2190450Z
Action: Engine/Extension/TripPin
Message: HostProcessId: 26716, Message: Schema type: type table [AirlineCode = text, Name = text]
Duration: 00:00:00.0000225
PID: 7796
TID: 1

Start: 2018-04-06T15:48:31.4969434Z
Action: Engine/Extension/TripPin

Start: 2018-04-06T15:48:31.5014955Z
Action: Engine/Extension/TripPin

Start: 2018-04-06T15:48:31.5920417Z
Action: Engine/Extension/TripPin

Start: 2018-04-06T15:48:31.6089830Z
Action: Engine/Extension/TripPin

```

Note that you see the serialized version of the `schema` parameter `type`, rather than what you'd get when you do a simple `Text.FromValue` on a type value (which results in "type").

Diagnostics.LogFailure

The `Diagnostics.LogFailure` function can be used to wrap function calls, and will only write to the trace if the function call fails (that is, returns an `error`).

```
Diagnostics.LogFailure = (text as text, function as function) as any => ...
```

Internally, `Diagnostics.LogFailure` adds a `try` operator to the `function` call. If the call fails, the `text` value is written to the trace before returning the original `error`. If the `function` call succeeds, the result is returned

without writing anything to the trace. Since M errors don't contain a full stack trace (that is, you typically only see the message of the error), this can be useful when you want to pinpoint where the error was actually raised.

As a (poor) example, modify the `withData` line of the `TripPinNavTable` function to force an error once again:

```
withData = Table.AddColumn(rename, "Data", each Diagnostics.LogError("Error in GetEntity", () => GetEntity(url, "DoesNotExist")), type table),
```

In the trace, you can find the resulting error message containing your `text`, and the original error information.

The screenshot shows the 'M Query Output' window with the 'Log' tab selected. The log contains several entries, with the last one being highlighted in blue. The highlighted entry is as follows:

Start: 2018-04-06T16:39:12.4396537Z
Action: Engine/Extension/TripPin
Message: HostProcessId: 26716, Message: Error in GetEntity: [Reason = "Expression.Error", Message = "Couldn't find entity: 'DoesNotExist'", Detail = null]
Duration: 00:00:00.0002472
PID: 9508
TID: 1

Be sure to reset your function to a working state before proceeding with the next tutorial.

Conclusion

This brief (but important!) lesson showed you how to make use of the diagnostic helper functions to log to the Power Query trace files. When used properly, these functions are extremely useful in debugging issues within your connector.

NOTE

As a connector developer, it is your responsibility to ensure that you do not log sensitive or personally identifiable information (PII) as part of your diagnostic logging. You must also be careful to not output too much trace information, as it can have a negative performance impact.

Next steps

[TripPin Part 9 - TestConnection](#)

TripPin Part 9 - TestConnection

1/15/2022 • 4 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you'll:

- Add a `TestConnection` handler
- Configure the on-premises data gateway (personal mode)
- Test scheduled refresh through the Power BI service

Custom connector support was added to the April 2018 release of the [personal on-premises data gateway](#). This new (preview) functionality allows for Scheduled Refresh of reports that make use of your custom connector.

This tutorial will cover the process of enabling your connector for refresh, and provide a quick walkthrough of the steps to configure the gateway. Specifically you'll:

1. Add a `TestConnection` handler to your connector
2. Install the On-Premises Data Gateway in Personal mode
3. Enable Custom Connector support in the Gateway
4. Publish a workbook that uses your connector to PowerBI.com
5. Configure scheduled refresh to test your connector

See [Handling Gateway Support](#) for more information on the `TestConnection` handler.

Background

There are three prerequisites for configuring a data source for scheduled refresh using PowerBI.com:

- **The data source is supported:** This means that the target gateway environment is aware of all of the functions contained in the query you want to refresh.
- **Credentials are provided:** To present the right credential entry dialog, Power BI needs to know the support authentication mechanism for a given data source.
- **The credentials are valid:** After the user provides credentials, they're validated by calling the data source's `TestConnection` handler.

The first two items are handled by registering your connector with the gateway. When the user attempts to configure scheduled refresh in PowerBI.com, the query information is sent to your personal gateway to determine if any data sources that aren't recognized by the Power BI service (that is, custom ones that you created) are available there. The third item is handled by invoking the `TestConnection` handler defined for your data source.

Adding a `TestConnection` handler

The `TestConnection` handler is added to the `Data Source Kind` declaration record (the same place you declare its supported authentication type(s)). The handler is a `function` with a single parameter of type `any`, which returns a `list`. The first value in the list is the function that will be called to actually test the connection. This is generally the same as your main data source function. In some cases you may need to expose a separate `shared` function to provide an efficient connection test, however, this should generally be avoided.

Since the TripPin data source function has no required arguments, the implementation for TestConnection is fairly simple:

```
// Data Source Kind description
TripPin = [
    // TestConnection is required to enable the connector through the Gateway
    TestConnection = (dataSourcePath) => { "TripPin.Contents" },
    Authentication = [
        Anonymous = []
    ],
    Label = "TripPin Part 9 - TestConnection"
];
```

NOTE

Future versions of the Power Query SDK will provide a way to validate the TestConnection handler from Visual Studio. Currently, the only mechanism that uses TestConnection is the on-premises data gateway.

Enabling custom connectors in the personal gateway

Download and install the [on-premises data gateway](#). When you run the installer, select the personal mode.

After installation is complete, launch the gateway and sign into Power BI. The sign-in process will automatically register your gateway with the Power BI services. Once signed in, perform the following steps:

1. Select the **Connectors** tab.
2. Select the switch to enable support for **Custom data connectors**.
3. Select the directory you want to load custom connectors from. This will usually be the same directory that you'd use for Power BI Desktop, but the value is configurable.
4. The page should now list all extension files in your target directory.

? X



On-premises data gateway (personal mode)

Status

Service Settings

Diagnostics

Network

Connectors

Custom data connectors



[Learn more](#)

Name

TripPin

Load custom data connectors from folder:

C:\Users\ [REDACTED] \Documents\Power BI Desktop\Custom Conn

...

[Close](#)

See the [online documentation](#) for more information about the gateway.

Testing scheduled refresh

Open Power BI Desktop and create a report that imports data using the TripPin connector.

Display Options ▾

- ▲ TripPin Advanced Schema [3]
 - Airlines
 - Airports
 - People

People

UserName	FirstName	LastName	Emails	AddressInfo
russellwhyte	Russell	Whyte	List	List
scottketchum	Scott	Ketchum	List	List
ronaldmundy	Ronald	Mundy	List	List
javieralfred	Javier	Alfred	List	List
willieashmore	Willie	Ashmore	List	List
vincentcalabrese	Vincent	Calabrese	List	List
clydeguess	Clyde	Guess	List	List
keithpinckney	Keith	Pinckney	List	List
marshallgaray	Marshall	Garay	List	List
ryantheriault	Ryan	Theriault	List	List
elainestewart	Elaine	Stewart	List	List
salliesampson	Sallie	Sampson	List	List
jonirosales	Joni	Rosales	List	List
georginabarlow	Georgina	Barlow	List	List
angelhuffman	Angel	Huffman	List	List
laurelosborn	Laurel	Osborn	List	List
sandyosborn	Sandy	Osborn	List	List
ursulabright	Ursula	Bright	List	List
genevievereeves	Genevieve	Reeves	List	List
kristakemp	Krista	Kemp	List	List

◀ ▶

Load Edit Cancel

Add one or more visuals to your report page (optional), and then publish the report to PowerBI.com.

After publishing, go to PowerBI.com and find the dataset for the report you just published. Select the ellipses, and then select **Schedule Refresh**. Expand the **Gateway connection** and **Data source credentials** sections.

Settings for TripPin

[Refresh history](#)

▲ Gateway connection

To use a data gateway, make sure the computer is online and the data source is added in [Manage Gateways](#).

Use your data gateway (personal mode)
(Online, running on [REDACTED]) Delete Gateway

Use an on-prem data gateway

Apply Discard

▲ Data source credentials

✖ Your data source can't be refreshed because the credentials are invalid. Please update your credentials and try again.

TripPin Edit credentials

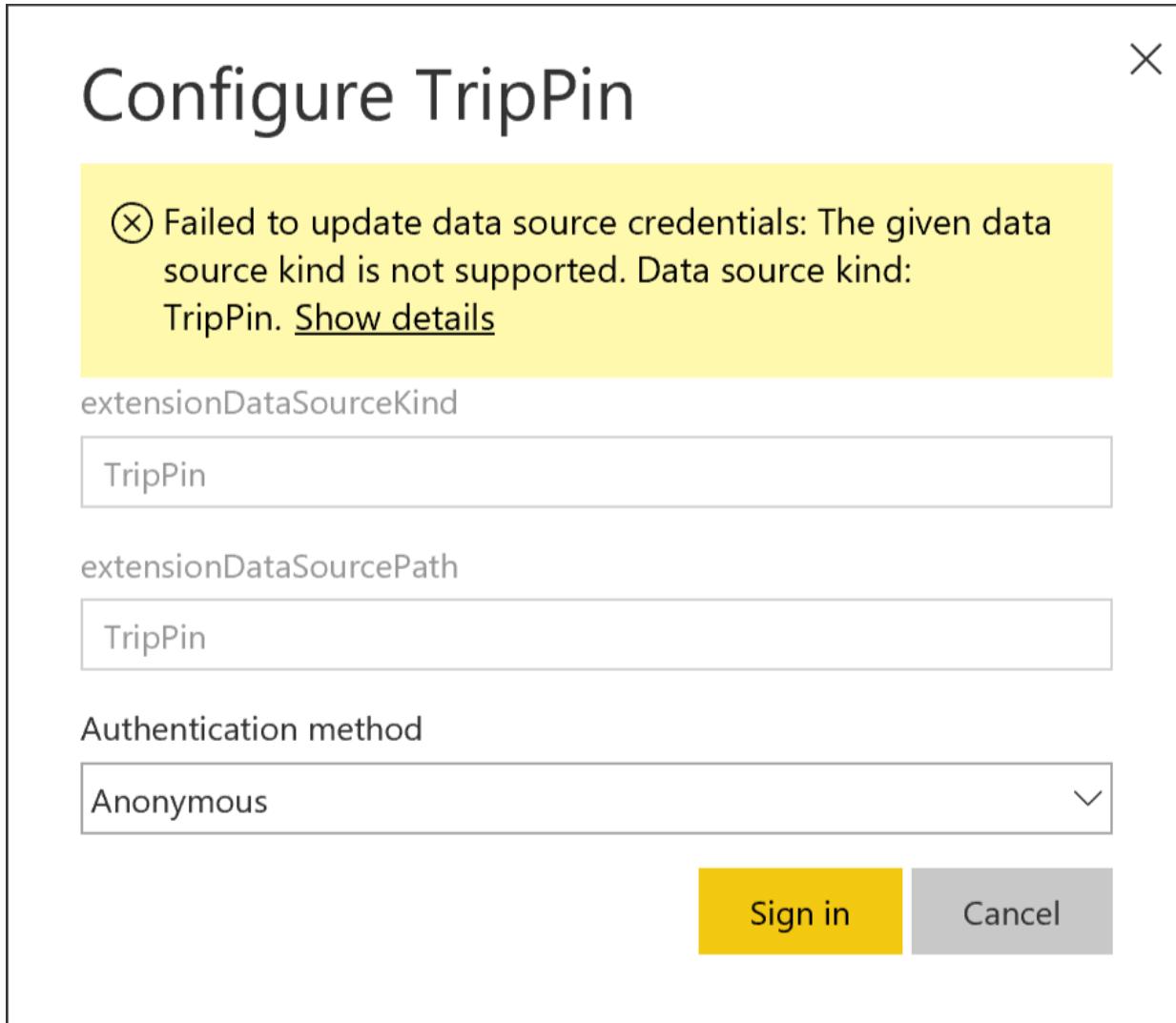
NOTE

If the dataset configuration page says that the report contains unknown data sources, your gateway/custom connector might not be configured properly. Go to the personal gateway configuration UI and make sure that there are no errors next to the TripPin connector. You may need to restart the gateway (on the **Service Settings** tab) to pick up the latest configuration.

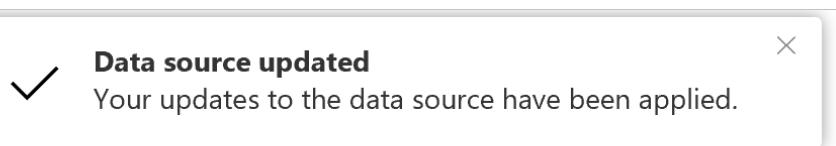
Select the **Edit credentials** link to bring up the authentication dialog, and then select sign-in.

NOTE

If you receive an error similar to the one below ("Failed to update data source credentials"), you most likely have an issue with your TestConnection handler.



After a successful call to `TestConnection`, the credentials will be accepted. You can now schedule refresh, or select the dataset ellipse and then select **Refresh Now**. You can select the **Refresh history** link to view the status of the refresh (which generally takes a few minutes to get kicked off).

Alerts Subscriptions D 
✓ **Data source updated**
Your updates to the data source have been applied.

Settings for TripPin

[Refresh history](#)

▲ **Gateway connection**

To use a data gateway, make sure the computer is online and the data source is added in [Manage Gateways](#).

Use your data gateway (personal mode)
([Online](#), running on [REDACTED]) [Delete Gateway](#)

Use an on-prem data gateway

[Apply](#) [Discard](#)

▲ **Data source credentials**

TripPin [Edit credentials](#)

► **Parameters**

► **Scheduled refresh**

► **Q&A and Cortana**

► **Featured Q&A questions**

Conclusion

Congratulations! You now have a production ready custom connector that supports automated refresh through the Power BI service.

Next steps

[TripPin Part 10 - Query Folding](#)

TripPin Part 10—Basic Query Folding

1/15/2022 • 16 minutes to read • [Edit Online](#)

This multi-part tutorial covers the creation of a new data source extension for Power Query. The tutorial is meant to be done sequentially—each lesson builds on the connector created in previous lessons, incrementally adding new capabilities to your connector.

In this lesson, you will:

- Learn the basics of query folding
- Learn about the `Table.View` function
- Replicate OData query folding handlers for:
 - `$top`
 - `$skip`
 - `$count`
 - `$select`
 - `$orderby`

One of the powerful features of the M language is its ability to push transformation work to underlying data source(s). This capability is referred to as *Query Folding* (other tools/technologies also refer to similar function as Predicate Pushdown, or Query Delegation). When creating a custom connector that uses an M function with built-in query folding capabilities, such as `OData.Feed` or `Odbc.DataSource`, your connector will automatically inherit this capability for free.

This tutorial will replicate the built-in query folding behavior for OData by implementing function handlers for the `Table.View` function. This part of the tutorial will implement some of the *easier* handlers to implement (that is, ones that don't require expression parsing and state tracking).

To understand more about the query capabilities that an OData service might offer, see [OData v4 URL Conventions](#).

NOTE

As stated above, the `OData.Feed` function will automatically provide query folding capabilities. Since the TripPin series is treating the OData service as a regular REST API, using `Web.Contents` rather than `OData.Feed`, you'll need to implement the query folding handlers yourself. For real world usage, we recommend that you use `OData.Feed` whenever possible.

See the [Table.View documentation](#) for more information about query folding in M.

Using Table.View

The `Table.View` function allows a custom connector to override default transformation handlers for your data source. An implementation of `Table.View` will provide a function for one or more of the supported handlers. If a handler is unimplemented, or returns an `error` during evaluation, the M engine will fall back to its default handler.

When a custom connector uses a function that doesn't support implicit query folding, such as `Web.Contents`, default transformation handlers will always be performed locally. If the REST API you are connecting to supports

query parameters as part of the query, `Table.View` will allow you to add optimizations that allow transformation work to be pushed to the service.

The `Table.View` function has the following signature:

```
Table.View(table as nullable table, handlers as record) as table
```

Your implementation will wrap your main data source function. There are two required handlers for `Table.View`:

- `GetType` —returns the expected `table type` of the query result
- `GetRows` —returns the actual `table` result of your data source function

The simplest implementation would be similar to the following:

```
TripPin.SuperSimpleView = (url as text, entity as text) as table =>
    Table.View(null, [
        GetType = () => Value.Type(GetRows()),
        GetRows = () => GetEntity(url, entity)
    ]);

```

Update the `TripPinNavTable` function to call `TripPin.SuperSimpleView` rather than `GetEntity`:

```
WithData = Table.AddColumn(rename, "Data", each TripPin.SuperSimpleView(url, [Name]), type table),
```

If you re-run the unit tests, you'll see that the behavior of your function hasn't changed. In this case your `Table.View` implementation is simply passing through the call to `GetEntity`. Since you haven't implemented any transformation handlers (yet), the original `url` parameter remains untouched.

Initial Implementation of `Table.View`

The above implementation of `Table.View` is simple, but not very useful. The following implementation will be used as your baseline—it doesn't implement any folding functionality, but has the scaffolding you'll need to do it.

```

TripPin.View = (baseUrl as text, entity as text) as table =>
    let
        // Implementation of Table.View handlers.
        //
        // We wrap the record with Diagnostics.WrapHandlers() to get some automatic
        // tracing if a handler returns an error.
        //
        View = (state as record) => Table.View(null, Diagnostics.WrapHandlers([
            // Returns the table type returned by GetRows()
            GetType = () => CalculateSchema(state),

            // Called last - retrieves the data from the calculated URL
            GetRows = () =>
                let
                    finalSchema = CalculateSchema(state),
                    finalUrl = CalculateUrl(state),

                    result = TripPin.Feed(finalUrl, finalSchema),
                    appliedType = Table.ChangeType(result, finalSchema)
                in
                    appliedType,

                //
                // Helper functions
                //
                // Retrieves the cached schema. If this is the first call
                // to CalculateSchema, the table type is calculated based on
                // the entity name that was passed into the function.
                CalculateSchema = (state) as type =>
                    if (state[Schema]? = null) then
                        GetSchemaForEntity(entity)
                    else
                        state[Schema],

                // Calculates the final URL based on the current state.
                CalculateUrl = (state) as text =>
                    let
                        urlWithEntity = Uri.Combine(state[Url], state[Entity])
                    in
                        urlWithEntity
                ))
            in
                View([Url = baseUrl, Entity = entity]);

```

If you look at the call to `Table.View`, you'll see an additional wrapper function around the `handlers` record—`Diagnostics.WrapHandlers`. This helper function is found in the `Diagnostics` module (that was introduced in a previous tutorial), and provides you with a useful way to automatically trace any errors raised by individual handlers.

The `GetType` and `GetRows` functions have been updated to make use of two new helper functions—`CalculateSchema` and `CaculateUrl`. Right now the implementations of those functions are fairly straightforward—you'll notice they contain parts of what was previously done by the `GetEntity` function.

Finally, you'll notice that you're defining an internal function (`View`) that accepts a `state` parameter. As you implement more handlers, they will recursively call the internal `View` function, updating and passing along `state` as they go.

Update the `TripPinNavTable` function once again, replacing the call to `TripPin.SuperSimpleView` with a call to the new `TripPin.View` function, and re-run the unit tests. You won't see any new functionality yet, but you now have a solid baseline for testing.

Implementing Query Folding

Since the M engine will automatically fall back to local processing when a query can't be folded, you must take some additional steps to validate that your `Table.View` handlers are working correctly.

The manual way to validate folding behavior is to watch the URL requests your unit tests make using a tool like Fiddler. Alternatively, the diagnostic logging you added to `TripPin.Feed` will emit the full URL being run, which *should* include the OData query string parameters your handlers will add.

An automated way to validate query folding is to force your unit test execution to fail if a query doesn't fully fold. You can do this by opening the project properties, and setting **Error on Folding Failure** to **True**. With this setting enabled, any query that requires local processing results in the following error:

We couldn't fold the expression to the source. Please try a simpler expression.

You can test this out by adding a new `Fact` to your unit test file that contains one or more table transformations.

```
// Query folding tests
Fact("Fold $top 1 on Airlines",
    #table( type table [AirlineCode = text, Name = text] , [{"AA", "American Airlines"}] ),
    Table.FirstN(Airlines, 1)
)
```

NOTE

The **Error on Folding Failure** setting is an "all or nothing" approach. If you want to test queries that aren't designed to fold as part of your unit tests, you'll need to add some conditional logic to enable/disable tests accordingly.

The remaining sections of this tutorial will each add a new `Table.View` handler. You'll be taking a [Test Driven Development \(TDD\)](#) approach, where you first add failing unit tests, and then implement the M code to resolve them.

Each handler section below will describe the functionality provided by the handler, the OData equivalent query syntax, the unit tests, and the implementation. Using the scaffolding code described above, each handler implementation requires two changes:

- Adding the handler to `Table.View` that will update the `state` record.
- Modifying `calculateUrl` to retrieve the values from the `state` and add to the url and/or query string parameters.

Handling `Table.FirstN` with `OnTake`

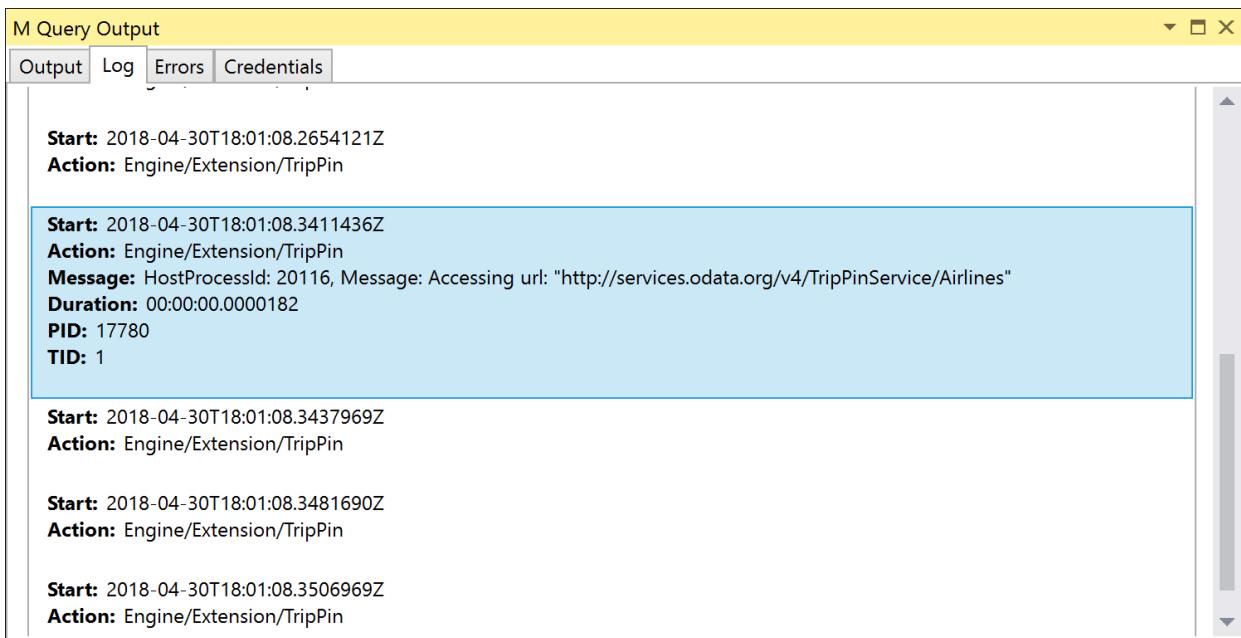
The [OnTake handler](#) receives a `count` parameter, which is the maximum number of rows to take. In OData terms, you can translate this to the `$top` query parameter.

You'll use the following unit tests:

```
// Query folding tests
Fact("Fold $top 1 on Airlines",
    #table( type table [AirlineCode = text, Name = text] , [{"AA", "American Airlines"}] ),
    Table.FirstN(Airlines, 1)
),
Fact("Fold $top 0 on Airports",
    #table( type table [Name = text, IataCode = text, Location = record] , {} ),
    Table.FirstN(Airports, 0)
),
```

These tests both use `Table.FirstN` to filter to the result set to the first X number of rows. If you run these tests

with **Error on Folding Failure** set to `False` (the default), the tests should succeed, but if you run Fiddler (or check the trace logs), you'll see that the request you send doesn't contain any OData query parameters.



The screenshot shows the 'M Query Output' window with the 'Log' tab selected. It displays several log entries related to the TripPin extension:

- Start:** 2018-04-30T18:01:08.2654121Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T18:01:08.3411436Z
Action: Engine/Extension/TripPin
Message: HostProcessId: 20116, Message: Accessing url: "http://services.odata.org/v4/TripPinService/Airlines"
Duration: 00:00:00.0000182
PID: 17780
TID: 1
- Start:** 2018-04-30T18:01:08.3437969Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T18:01:08.3481690Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T18:01:08.3506969Z
Action: Engine/Extension/TripPin

If you set **Error on Folding Failure** to `True`, they will fail with the "Please try a simpler expression." error. To fix this, you'll define your first Table.View handler for `OnTake`.

The `OnTake` handler looks like this:

```
OnTake = (count as number) =>
  let
    // Add a record with Top defined to our state
    newState = state & [ Top = count ]
  in
    @View(newState),
```

The `calculateUrl` function is updated to extract the `Top` value from the `state` record, and set the right parameter in the query string.

```
// Calculates the final URL based on the current state.
CalculateUrl = (state) as text =>
  let
    urlWithEntity = Uri.Combine(state[Url], state[Entity]),

    // Uri.BuildQueryString requires that all field values
    // are text literals.
    defaultQueryString = [],

    // Check for Top defined in our state
    qsWithTop =
      if (state[Top]? <> null) then
        // add a $top field to the query string record
        defaultQueryString & [ #"$top" = Number.ToText(state[Top]) ]
      else
        defaultQueryString,

    encodedQueryString = Uri.BuildQueryString(qsWithTop),
    finalUrl = urlWithEntity & "?" & encodedQueryString
  in
    finalUrl
```

Rerunning the unit tests, you can see that the URL you are accessing now contains the `$top` parameter. (Note

that due to URL encoding, `$top` appears as `%24top`, but the OData service is smart enough to convert it automatically).

The screenshot shows the 'M Query Output' window with tabs for 'Output', 'Log', 'Errors', and 'Credentials'. The 'Log' tab is selected, showing the following entries:

- Action:** engine/extension/trippin
Start: 2018-04-30T19:22:43.1213089Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T19:22:43.1297467Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T19:22:43.1679684Z
Action: Engine/Extension/TripPin
Message: HostProcessId: 20116, Message: Accessing url: "http://services.odata.org/v4/TripPinService/People?%24top=2"
Duration: 00:00:00.0000459
PID: 24328
TID: 1
- Start:** 2018-04-30T19:22:43.1782248Z
Action: Engine/Extension/TripPin
- Start:** 2018-04-30T19:22:43.2195863Z
Action: Engine/Extension/TripPin

Handling Table.Skip with OnSkip

The [OnSkip handler](#) is a lot like [OnTake](#). It receives a `count` parameter, which is the number of rows to skip from the result set. This translates nicely to the OData `$skip` query parameter.

Unit tests:

```
// OnSkip
Fact("Fold $skip 14 on Airlines",
    #table( type table [AirlineCode = text, Name = text] , [{"EK", "Emirates"}] ),
    Table.Skip(Airlines, 14)
),
Fact("Fold $skip 0 and $top 1",
    #table( type table [AirlineCode = text, Name = text] , [{"AA", "American Airlines"}] ),
    Table.FirstN(Table.Skip(Airlines, 0), 1)
),
```

Implementation:

```
// OnSkip - handles the Table.Skip transform.
// The count value should be >= 0.
OnSkip = (count as number) =>
    let
        newState = state & [ Skip = count ]
    in
        @View(newState),
```

Matching updates to `CalculateUrl`:

```
qsWithSkip =
    if (state[Skip]? <> null) then
        qsWithTop & [ "#$skip" = Number.ToString(state[Skip]) ]
    else
        qsWithTop,
```

Handling Table.SelectColumns with OnSelectColumns

The [OnSelectColumns](#) handler is called when the user selects or removes columns from the result set. The

handler receives a `list` of `text` values, representing the column(s) to be selected. In OData terms, this operation will map to the `$select` query option. The advantage of folding column selection becomes apparent when you are dealing with tables with many columns. The `$select` operator will remove unselected columns from the result set, resulting in more efficient queries.

Unit tests:

```
// OnSelectColumns
Fact("Fold $select single column",
    #table( type table [AirlineCode = text] , {"AA"} ),
    Table.FirstN(Table.SelectColumns(Airlines, {"AirlineCode"}), 1)
),
Fact("Fold $select multiple column",
    #table( type table [UserName = text, FirstName = text, LastName = text], {"russellwhyte", "Russell", "Whyte"} ),
    Table.FirstN(Table.SelectColumns(People, {"UserName", "FirstName", "LastName"}), 1)
),
Fact("Fold $select with ignore column",
    #table( type table [AirlineCode = text] , {"AA"} ),
    Table.FirstN(Table.SelectColumns(Airlines, {"AirlineCode", "DoesNotExist"}, MissingField.Ignore), 1)
),
```

The first two tests select different numbers of columns with `Table.SelectColumns`, and include a `Table.FirstN` call to simplify the test case.

NOTE

If the test were to simply return the column names (using `Table.ColumnNames`) and not any data, the request to the OData service will never actually be sent. This is because the call to `GetType` will return the schema, which contains all of the information the M engine needs to calculate the result.

The third test uses the `MissingField.Ignore` option, which tells the M engine to ignore any selected columns that don't exist in the result set. The `OnSelectColumns` handler does not need to worry about this option—the M engine will handle it automatically (that is, missing columns won't be included in the `columns` list).

NOTE

The other option for `Table.SelectColumns`, `MissingField.UseNull`, requires a connector to implement the `OnAddColumn` handler. This will be done in a subsequent lesson.

The implementation for `OnSelectColumns` does two things:

- Adds the list of selected columns to the `state`.
- Re-calculates the `Schema` value so you can set the right table type.

```

OnSelectColumns = (columns as list) =>
    let
        // get the current schema
        currentSchema = CalculateSchema(state),
        // get the columns from the current schema (which is an M Type value)
        rowRecordType = Type.RecordFields(Type.TableRow(currentSchema)),
        existingColumns = Record.FieldNames(rowRecordType),
        // calculate the new schema
        columnsToRemove = List.Difference(existingColumns, columns),
        updatedColumns = Record.RemoveFields(rowRecordType, columnsToRemove),
        newSchema = type table (Type.ForRecord(updatedColumns, false))
    in
        @View(state &
            [
                SelectColumns = columns,
                Schema = newSchema
            ]
        ),

```

`CalculateUrl` is updated to retrieve the list of columns from the state, and combine them (with a separator) for the `$select` parameter.

```

// Check for explicitly selected columns
qsWithSelect =
    if (state[SelectColumns]? <> null) then
        qsWithSkip & [ #"$select" = Text.Combine(state[SelectColumns], ",") ]
    else
        qsWithSkip,

```

Handling Table.Sort with OnSort

The `OnSort` handler receives a `list` of `record` values. Each record contains a `Name` field, indicating the name of the column, and an `Order` field which is equal to `Order.Ascending` or `Order.Descending`. In OData terms, this operation will map to the `$orderby` query option. The `$orderby` syntax has the column name followed by `asc` or `desc` to indicate ascending or descending order. When sorting on multiple columns, the values are separated with a comma. Note that if the `columns` parameter contains more than one item, it is important to maintain the order in which they appear.

Unit tests:

```

// OnSort
Fact("Fold $orderby single column",
    #table( type table [AirlineCode = text, Name = text], {"TK", "Turkish Airlines"}),
    Table.FirstN(Table.Sort(Airlines, {"AirlineCode", Order.Descending}), 1)
),
Fact("Fold $orderby multiple column",
    #table( type table [UserName = text], {"javieralfred"}),
    Table.SelectColumns(Table.FirstN(Table.Sort(People, {"LastName", Order.Ascending}, {"UserName", Order.Descending}), 1), {"UserName"})
)

```

Implementation:

```

// OnSort - receives a list of records containing two fields:
//   [Name] - the name of the column to sort on
//   [Order] - equal to Order.Ascending or Order.Descending
// If there are multiple records, the sort order must be maintained.
//
// OData allows you to sort on columns that do not appear in the result
// set, so we do not have to validate that the sorted columns are in our
// existing schema.
OnSort = (order as list) =>
    let
        // This will convert the list of records to a list of text,
        // where each entry is "<columnName> <asc|desc>"
        sorting = List.Transform(order, (o) =>
            let
                column = o[Name],
                order = o[Order],
                orderText = if (order = Order.Ascending) then "asc" else "desc"
            in
                column & " " & orderText
        ),
        orderBy = Text.Combine(sorting, ", ")
    in
        @View(state & [ OrderBy = orderBy ]),

```

Updates to `CalculateUrl`:

```

qsWithOrderBy =
    if (state[OrderBy]? <> null) then
        qsWithSelect & [ #"orderby" = state[OrderBy] ]
    else
        qsWithSelect,

```

Handling `Table.RowCount` with `GetRowCount`

Unlike the other query handlers you've implemented, the `GetRowCount` handler will return a single value—the number of rows expected in the result set. In an M query, this would typically be the result of the `Table.RowCount` transform. You have a few different options on how to handle this as part of an OData query.

- The `$count query parameter`, which returns the count as a separate field in the result set.
- The `/$count path segment`, which will return **only** the total count, as a scalar value.

The downside to the query parameter approach is that you still need to send the entire query to the OData service. Since the count comes back inline as part of the result set, you'll have to process the first page of data from the result set. While this is still more efficient than reading the entire result set and counting the rows, it's probably still more work than you want to do.

The advantage of the path segment approach is that you'll only receive a single scalar value in the result. This makes the entire operation a lot more efficient. However, as described in the OData specification, the `/$count` path segment will return an error if you include other query parameters, such as `$top` or `$skip`, which limits its usefulness.

In this tutorial, you'll implement the `GetRowCount` handler using the path segment approach. To avoid the errors you'd get if other query parameters are included, you'll check for other state values, and return an "unimplemented error" (`...`) if you find any. Returning any error from a `Table.View` handler tells the M engine that the operation cannot be folded, and it should fallback to the default handler instead (which in this case would be counting the total number of rows).

First, add a simple unit test:

```
// GetRowCount
Fact("Fold $count", 15, Table.RowCount(Airlines)),
```

Since the `/$count` path segment returns a single value (in plain/text format) rather than a JSON result set, you'll also have to add a new internal function (`TripPin.Scalar`) for making the request and handling the result.

```
// Similar to TripPin.Feed, but is expecting back a scalar value.
// This function returns the value from the service as plain text.
TripPin.Scalar = (url as text) as text =>
    let
        _url = Diagnostics.LogValue("TripPin.Scalar url", url),

        headers = DefaultRequestHeaders & [
            #"Accept" = "text/plain"
        ],

        response = Web.Contents(_url, [ Headers = headers ]),
        toText = Text.FromBinary(response)
    in
        toText;
```

The implementation will then use this function (if no other query parameters are found in the `state`):

```
GetRowCount = () as number =>
    if (Record.FieldCount(Record.RemoveFields(state, {"Url", "Entity", "Schema"}, MissingField.Ignore)) > 0)
then
    ...
else
    let
        newState = state & [ RowCountOnly = true ],
        finalUrl = CalculateUrl(newState),
        value = TripPin.Scalar(finalUrl),
        converted = Number.FromText(value)
    in
        converted,
```

The `CalculateUrl` function is updated to append `/$count` to the URL if the `RowCountOnly` field is set in the `state`.

```
// Check for $count. If all we want is a row count,
// then we add /$count to the path value (following the entity name).
urlWithRowCount =
    if (state[RowCountOnly]? = true) then
        urlWithEntity & "/$count"
    else
        urlWithEntity,
```

The new `Table.RowCount` unit test should now pass.

To test the fallback case, you'll add another test that forces the error. First, add a helper method that checks the result of a `try` operation for a folding error.

```
// Returns true if there is a folding error, or the original record (for logging purposes) if not.  
Test.IsFoldingError = (tryResult as record) =>  
    if ( tryResult[HasError]? = true and tryResult[Error][Message] = "We couldn't fold the expression to the  
data source. Please try a simpler expression.") then  
        true  
    else  
        tryResult;
```

Then add a test that uses both `Table.RowCount` and `Table.FirstN` to force the error.

```
// test will fail if "Fail on Folding Error" is set to false  
Fact("Fold $count + $top *error*", true, Test.IsFoldingError(try Table.RowCount(Table.FirstN(Airlines,  
3))),
```

An important note here is that this test will now return an error if **Error on Folding Error** is set to `false`, because the `Table.RowCount` operation will fall back to the local (default) handler. Running the tests with **Error on Folding Error** set to `true` will cause `Table.RowCount` to fail, and allows the test to succeed.

Conclusion

Implementing `Table.View` for your connector adds a significant amount of complexity to your code. Since the M engine can process all transformations locally, adding `Table.View` handlers does not enable new scenarios for your users, but will result in more efficient processing (and potentially, happier users). One of the main advantages of the `Table.View` handlers being optional is that it allows you to incrementally add new functionality without impacting backwards compatibility for your connector.

For most connectors, an important (and basic) handler to implement is `OnTake` (which translates to `$top` in OData), as it limits the amount of rows returned. The Power Query experience will always perform an `OnTake` of `1000` rows when displaying previews in the navigator and query editor, so your users might see significant performance improvements when working with larger data sets.

GitHub Connector Sample

1/15/2022 • 7 minutes to read • [Edit Online](#)

The GitHub M extension shows how to add support for an OAuth 2.0 protocol authentication flow. You can learn more about the specifics of GitHub's authentication flow on the [GitHub Developer site](#).

Before you get started creating an M extension, you need to register a new app on GitHub, and replace the `client_id` and `client_secret` files with the appropriate values for your app.

Note about compatibility issues in Visual Studio: *The Power Query SDK uses an Internet Explorer based control to popup OAuth dialogs. GitHub has deprecated its support for the version of IE used by this control, which will prevent you from completing the permission grant for your app if run from within Visual Studio. An alternative is to load the extension with Power BI Desktop and complete the first OAuth flow there. After your application has been granted access to your account, subsequent logins will work fine from Visual Studio.*

OAuth and Power BI

OAuth is a form of credentials delegation. By logging in to GitHub and authorizing the "application" you create for GitHub, the user is allowing your "application" to login on their behalf to retrieve data into Power BI. The "application" must be granted rights to retrieve data (get an `access_token`) and to refresh the data on a schedule (get and use a `refresh_token`). Your "application" in this context is your Data Connector used to run queries within Power BI. Power BI stores and manages the `access_token` and `refresh_token` on your behalf.

NOTE

To allow Power BI to obtain and use the `access_token`, you must specify the redirect url as <https://oauth.powerbi.com/views/oauthredirect.html>.

When you specify this URL and GitHub successfully authenticates and grants permissions, GitHub will redirect to PowerBI's `oauthredirect` endpoint so that Power BI can retrieve the `access_token` and `refresh_token`.

How to register a GitHub app

Your Power BI extension needs to login to GitHub. To enable this, you register a new OAuth application with GitHub at <https://github.com/settings/applications/new>.

1. `Application name` : Enter a name for the application for your M extension.
2. `Authorization callback URL` : Enter <https://oauth.powerbi.com/views/oauthredirect.html>.
3. `Scope` : In GitHub, set scope to `user, repo`.

NOTE

A registered OAuth application is assigned a unique Client ID and Client Secret. The Client Secret should not be shared. You get the Client ID and Client Secret from the GitHub application page. Update the files in your Data Connector project with the Client ID (`client_id` file) and Client Secret (`client_secret` file).

How to implement GitHub OAuth

This sample will walk you through the following steps:

1. Create a Data Source Kind definition that declares it supports OAuth.
2. Provide details so the M engine can start the OAuth flow (`StartLogin`).
3. Convert the code received from GitHub into an access_token (`FinishLogin` and `TokenMethod`).
4. Define functions that access the GitHub API (`GithubSample.Contents`).

Step 1 - Create a Data Source definition

A Data Connector starts with a `record` that describes the extension, including its unique name (which is the name of the record), supported authentication type(s), and a friendly display name (label) for the data source. When supporting OAuth, the definition contains the functions that implement the OAuth contract—in this case, `StartLogin` and `FinishLogin`.

```
//  
// Data Source definition  
//  
GithubSample = [  
    Authentication = [  
        OAuth = [  
            StartLogin = StartLogin,  
            FinishLogin = FinishLogin  
        ]  
    ],  
    Label = Extension.LoadString("DataSourceLabel")  
];
```

Step 2 - Provide details so the M engine can start the OAuth flow

The GitHub OAuth flow starts when you direct users to the <https://github.com/login/oauth/authorize> page. For the user to login, you need to specify a number of query parameters:

NAME	TYPE	DESCRIPTION
client_id	string	Required. The client ID you received from GitHub when you registered.
redirect_uri	string	The URL in your app where users will be sent after authorization. See details below about redirect urls. For M extensions, the <code>redirect_uri</code> must be "https://oauth.powerbi.com/views/oauthredirect.html".
scope	string	A comma separated list of scopes. If not provided, scope defaults to an empty list of scopes for users that don't have a valid token for the app. For users who do already have a valid token for the app, the user won't be shown the OAuth authorization page with the list of scopes. Instead, this step of the flow will automatically complete with the same scopes that were used last time the user completed the flow.
state	string	An un-guessable random string. It's used to protect against cross-site request forgery attacks.

The following code snippet describes how to implement a `StartLogin` function to start the login flow. A `StartLogin` function takes a `resourceUrl`, `state`, and `display` value. In the function, create an `AuthorizeUrl` that concatenates the GitHub authorize URL with the following parameters:

- `client_id`: You get the client ID after you register your extension with GitHub from the GitHub application page.
- `scope`: Set scope to "`user, repo`". This sets the authorization scope (that is, what your app wants to access) for the user.
- `state`: An internal value that the M engine passes in.
- `redirect_uri`: Set to <https://oauth.powerbi.com/views/oauthredirect.html>.

```
StartLogin = (resourceUrl, state, display) =>
    let
        AuthorizeUrl = "https://github.com/login/oauth/authorize?" & Uri.BuildQueryString([
            client_id = client_id,
            scope = "user, repo",
            state = state,
            redirect_uri = redirect_uri])
    in
    [
        LoginUri = AuthorizeUrl,
        CallbackUri = redirect_uri,
        WindowHeight = windowHeight,
        WindowWidth = windowWidth,
        Context = null
    ];

```

If this is the first time the user is logging in with your app (identified by its `client_id` value), they'll see a page that asks them to grant access to your app. Subsequent login attempts will simply ask for their credentials.

Step 3 - Convert the code received from GitHub into an access_token

If the user completes the authentication flow, GitHub redirects back to the Power BI redirect URL with a temporary code in a `code` parameter, as well as the state you provided in the previous step in a `state` parameter. Your `FinishLogin` function will extract the code from the `callbackUri` parameter, and then exchange it for an access token (using the `TokenMethod` function).

```
FinishLogin = (context, callbackUri, state) =>
    let
        Parts = Uri.Parts(callbackUri)[Query]
    in
        TokenMethod(Parts[code]);
```

To get a GitHub access token, you pass the temporary code from the GitHub Authorize Response. In the `TokenMethod` function, you formulate a POST request to GitHub's `access_token` endpoint (https://github.com/login/oauth/access_token). The following parameters are required for the GitHub endpoint:

NAME	TYPE	DESCRIPTION
client_id	string	Required. The client ID you received from GitHub when you registered.
client_secret	string	Required. The client secret you received from GitHub when you registered.

NAME	TYPE	DESCRIPTION
code	string	Required. The code you received in <code>FinishLogin</code> .
redirect_uri	string	The URL in your app where users will be sent after authorization. See details below about redirect URLs.

Here are the details used parameters for the [Web.Contents](#) call.

ARGUMENT	DESCRIPTION	VALUE
url	The URL for the Web site.	<code>https://github.com/login/oauth/access_token</code>
options	A record to control the behavior of this function.	Not used in this case
Query	Programmatically add query parameters to the URL.	<pre>Content = Text.ToBinary(Uri.BuildQueryString([client_id = client_id, client_secret = client_secret, code = code, redirect_uri = redirect_uri]))</pre> <p>Where</p> <ul style="list-style-type: none"> • <code>client_id</code> : Client ID from GitHub application page. • <code>client_secret</code> : Client secret from GitHub application page. • <code>code</code> : Code in GitHub authorization response. • <code>redirect_uri</code> : The URL in your app where users will be sent after authorization.
Headers	A record with additional headers for the HTTP request.	<pre>Headers= [#"Content-type" = "application/x-www-form-urlencoded", #"Accept" = "application/json"]</pre>

This code snippet describes how to implement a `TokenMethod` function to exchange an auth code for an access token.

```

TokenMethod = (code) =>
let
    Response = Web.Contents("https://Github.com/login/oauth/access_token", [
        Content = Text.ToBinary(Uri.BuildQueryString([
            client_id = client_id,
            client_secret = client_secret,
            code = code,
            redirect_uri = redirect_uri])),
        Headers=[#"Content-type" = "application/x-www-form-urlencoded",#"Accept" =
"application/json"]]),
    Parts = Json.Document(Response)
in
    Parts;

```

The JSON response from the service will contain an access_token field. The `TokenMethod` method converts the JSON response into an M record using `Json.Document`, and returns it to the engine.

Sample response:

```
{
    "access_token": "e72e16c7e42f292c6912e7710c838347ae178b4a",
    "scope": "user,repo",
    "token_type": "bearer"
}
```

Step 4 - Define functions that access the GitHub API

The following code snippet exports two functions (`GithubSample.Contents` and `GithubSample.PagedTable`) by marking them as `shared`, and associates them with the `GithubSample` Data Source Kind.

```

[DataSource.Kind="GithubSample", Publish="GithubSample.UI"]
shared GithubSample.Contents = Value.ReplaceType(Github.Contents, type function (url as Uri.Type) as any);

[DataSource.Kind="GithubSample"]
shared GithubSample.PagedTable = Value.ReplaceType(Github.PagedTable, type function (url as Uri.Type) as nullable table);

```

The `GithubSample.Contents` function is also published to the UI (allowing it to appear in the **Get Data** dialog). The `Value.ReplaceType` function is used to set the function parameter to the `Url.Type` ascribed type.

By associating these functions with the `GithubSample` data source kind, they'll automatically use the credentials that the user provided. Any M library functions that have been enabled for extensibility (such as `Web.Contents`) will automatically inherit these credentials as well.

For more details on how credential and authentication works, see [Handling Authentication](#).

Sample URL

This connector is able to retrieve formatted data from any of the GitHub v3 REST API endpoints. For example, the query to pull all commits to the Data Connectors repo would look like this:

```
GithubSample.Contents("https://api.github.com/repos/microsoft/dataconnectors/commits")
```

List of Samples

1/15/2022 • 2 minutes to read • [Edit Online](#)

We maintain a list of samples on the DataConnectors repo on GitHub. Each of the links below links to a folder in the sample repository. Generally these folders include a readme, one or more .pq / .query.pq files, a project file for Visual Studio, and in some cases icons. To open these files in Visual Studio, make sure you've set up the SDK properly, and run the .mproj file from the cloned or downloaded folder.

Functionality

SAMPLE	DESCRIPTION	LINK
Hello World	This simple sample shows the basic structure of a connector.	GitHub Link
Hello World with Docs	Similar to the Hello World sample, this sample shows how to add documentation to a shared function.	GitHub Link
Navigation Tables	This sample provides two examples of how to create a navigation table for your data connector using the <code>Table.ToTable</code> function.	GitHub Link
Unit Testing	This sample shows how you can add simple unit testing to your <code><extension>.query.pq</code> file.	GitHub Link
Relationships	This sample demonstrates the declaration of table relationships that will be detected by Power BI Desktop.	GitHub Link

OAuth

SAMPLE	DESCRIPTION	LINK
GitHub	This sample corresponds to the GitHub connector tutorial .	GitHub Link

ODBC

SAMPLE	DESCRIPTION	LINK
SQL	This connector sample serves as a template for ODBC connectors.	GitHub Link
Redshift	This connector sample uses the Redshift ODBC driver, and is based on the connector template.	GitHub Link

SAMPLE	DESCRIPTION	LINK
Hive LLAP	This connector sample uses the Hive ODBC driver, and is based on the connector template.	GitHub Link
Snowflake	This connector sample uses the Snowflake ODBC driver, and is based on the connector template.	GitHub Link
Impala	This connector sample uses the Cloudera Impala ODBC driver, and is based on the connector template.	GitHub Link
Direct Query for SQL	This sample creates an ODBC-based custom connector that enables Direct Query for SQL Server.	GitHub Link

TripPin

SAMPLE	DESCRIPTION	LINK
Part 1	This sample corresponds to TripPin Tutorial Part 1 - OData .	GitHub Link
Part 2	This sample corresponds to TripPin Tutorial Part 2 - REST .	GitHub Link
Part 3	This sample corresponds to TripPin Tutorial Part 3 - Navigation Tables .	GitHub Link
Part 4	This sample corresponds to TripPin Tutorial Part 4 - Data Source Paths .	GitHub Link
Part 5	This sample corresponds to TripPin Tutorial Part 5 - Paging .	GitHub Link
Part 6	This sample corresponds to TripPin Tutorial Part 6 - Enforcing Schema .	GitHub Link
Part 7	This sample corresponds to TripPin Tutorial Part 7 - Advanced Schema with M Types .	GitHub Link
Part 8	This sample corresponds to TripPin Tutorial Part 8 - Adding Diagnostics .	GitHub Link
Part 9	This sample corresponds to TripPin Tutorial Part 9 - Test Connection .	GitHub Link
Part 10	This sample corresponds to TripPin Tutorial Part 10 - Basic Query Folding .	GitHub Link

List of Samples

1/15/2022 • 2 minutes to read • [Edit Online](#)

We maintain a list of samples on the DataConnectors repo on GitHub. Each of the links below links to a folder in the sample repository. Generally these folders include a readme, one or more .pq / .query.pq files, a project file for Visual Studio, and in some cases icons. To open these files in Visual Studio, make sure you've set up the SDK properly, and run the .mproj file from the cloned or downloaded folder.

Functionality

SAMPLE	DESCRIPTION	LINK
Hello World	This simple sample shows the basic structure of a connector.	GitHub Link
Hello World with Docs	Similar to the Hello World sample, this sample shows how to add documentation to a shared function.	GitHub Link
Navigation Tables	This sample provides two examples of how to create a navigation table for your data connector using the <code>Table.ToTable</code> function.	GitHub Link
Unit Testing	This sample shows how you can add simple unit testing to your <code><extension>.query.pq</code> file.	GitHub Link
Relationships	This sample demonstrates the declaration of table relationships that will be detected by Power BI Desktop.	GitHub Link

OAuth

SAMPLE	DESCRIPTION	LINK
GitHub	This sample corresponds to the GitHub connector tutorial .	GitHub Link

ODBC

SAMPLE	DESCRIPTION	LINK
SQL	This connector sample serves as a template for ODBC connectors.	GitHub Link
Redshift	This connector sample uses the Redshift ODBC driver, and is based on the connector template.	GitHub Link

SAMPLE	DESCRIPTION	LINK
Hive LLAP	This connector sample uses the Hive ODBC driver, and is based on the connector template.	GitHub Link
Snowflake	This connector sample uses the Snowflake ODBC driver, and is based on the connector template.	GitHub Link
Impala	This connector sample uses the Cloudera Impala ODBC driver, and is based on the connector template.	GitHub Link
Direct Query for SQL	This sample creates an ODBC-based custom connector that enables Direct Query for SQL Server.	GitHub Link

TripPin

SAMPLE	DESCRIPTION	LINK
Part 1	This sample corresponds to TripPin Tutorial Part 1 - OData .	GitHub Link
Part 2	This sample corresponds to TripPin Tutorial Part 2 - REST .	GitHub Link
Part 3	This sample corresponds to TripPin Tutorial Part 3 - Navigation Tables .	GitHub Link
Part 4	This sample corresponds to TripPin Tutorial Part 4 - Data Source Paths .	GitHub Link
Part 5	This sample corresponds to TripPin Tutorial Part 5 - Paging .	GitHub Link
Part 6	This sample corresponds to TripPin Tutorial Part 6 - Enforcing Schema .	GitHub Link
Part 7	This sample corresponds to TripPin Tutorial Part 7 - Advanced Schema with M Types .	GitHub Link
Part 8	This sample corresponds to TripPin Tutorial Part 8 - Adding Diagnostics .	GitHub Link
Part 9	This sample corresponds to TripPin Tutorial Part 9 - Test Connection .	GitHub Link
Part 10	This sample corresponds to TripPin Tutorial Part 10 - Basic Query Folding .	GitHub Link

List of Samples

1/15/2022 • 2 minutes to read • [Edit Online](#)

We maintain a list of samples on the DataConnectors repo on GitHub. Each of the links below links to a folder in the sample repository. Generally these folders include a readme, one or more .pq / .query.pq files, a project file for Visual Studio, and in some cases icons. To open these files in Visual Studio, make sure you've set up the SDK properly, and run the .mproj file from the cloned or downloaded folder.

Functionality

SAMPLE	DESCRIPTION	LINK
Hello World	This simple sample shows the basic structure of a connector.	GitHub Link
Hello World with Docs	Similar to the Hello World sample, this sample shows how to add documentation to a shared function.	GitHub Link
Navigation Tables	This sample provides two examples of how to create a navigation table for your data connector using the <code>Table.ToTable</code> function.	GitHub Link
Unit Testing	This sample shows how you can add simple unit testing to your <code><extension>.query.pq</code> file.	GitHub Link
Relationships	This sample demonstrates the declaration of table relationships that will be detected by Power BI Desktop.	GitHub Link

OAuth

SAMPLE	DESCRIPTION	LINK
GitHub	This sample corresponds to the GitHub connector tutorial .	GitHub Link

ODBC

SAMPLE	DESCRIPTION	LINK
SQL	This connector sample serves as a template for ODBC connectors.	GitHub Link
Redshift	This connector sample uses the Redshift ODBC driver, and is based on the connector template.	GitHub Link

SAMPLE	DESCRIPTION	LINK
Hive LLAP	This connector sample uses the Hive ODBC driver, and is based on the connector template.	GitHub Link
Snowflake	This connector sample uses the Snowflake ODBC driver, and is based on the connector template.	GitHub Link
Impala	This connector sample uses the Cloudera Impala ODBC driver, and is based on the connector template.	GitHub Link
Direct Query for SQL	This sample creates an ODBC-based custom connector that enables Direct Query for SQL Server.	GitHub Link

TripPin

SAMPLE	DESCRIPTION	LINK
Part 1	This sample corresponds to TripPin Tutorial Part 1 - OData .	GitHub Link
Part 2	This sample corresponds to TripPin Tutorial Part 2 - REST .	GitHub Link
Part 3	This sample corresponds to TripPin Tutorial Part 3 - Navigation Tables .	GitHub Link
Part 4	This sample corresponds to TripPin Tutorial Part 4 - Data Source Paths .	GitHub Link
Part 5	This sample corresponds to TripPin Tutorial Part 5 - Paging .	GitHub Link
Part 6	This sample corresponds to TripPin Tutorial Part 6 - Enforcing Schema .	GitHub Link
Part 7	This sample corresponds to TripPin Tutorial Part 7 - Advanced Schema with M Types .	GitHub Link
Part 8	This sample corresponds to TripPin Tutorial Part 8 - Adding Diagnostics .	GitHub Link
Part 9	This sample corresponds to TripPin Tutorial Part 9 - Test Connection .	GitHub Link
Part 10	This sample corresponds to TripPin Tutorial Part 10 - Basic Query Folding .	GitHub Link

Additional connector functionality

1/15/2022 • 7 minutes to read • [Edit Online](#)

This article provides information about different types of additional connector functionality that connector developers might want to invest in. For each type, this article outlines availability and instructions to enable the functionality.

Authentication

While implementing authentication is covered in the [authentication](#) article, there are other methods that connector owners might be interested in offering.

Windows authentication

Windows authentication is supported. To enable Windows-based authentication in your connector, add the following line in the **Authentication** section of your connector.

```
Windows = [ SupportsAlternateCredentials = true ]
```

This change will expose Windows authentication as an option in the Power BI Desktop authentication experience. The **SupportsAlternateCredentials** flag will expose the option to "Connect using alternative credentials". After this flag is enabled, you can specify explicit Windows account credentials (username and password). You can use this feature to test impersonation by providing your own account credentials.

Single sign-on authentication

This section outlines options available for implementing single sign-on (SSO) functionality into your certified connector. Currently, there is no support for "plug and play" extensibility for SSO. Enabling SSO would require changes and collaboration both on the Microsoft and data source or connector sides, so reach out to your Microsoft contact prior to starting work.

Azure Active Directory SSO

Azure Active Directory (Azure AD)-based SSO is supported in cloud scenarios. The data source must accept Azure AD access tokens, as the Power BI Azure AD user token will be exchanged with a data source token from Azure AD. If you have a certified connector, reach out to your Microsoft contact to learn more.

Kerberos SSO

Kerberos-based single sign-on is supported in gateway scenarios. The data source must support Windows authentication. Generally, these scenarios involve Direct Query-based reports, and a connector based on an ODBC driver. The primary requirements for the driver are that it can determine Kerberos configuration settings from the current thread context, and that it supports thread-based user impersonation. The gateway must be [configured](#) to support Kerberos Constrained Delegation (KCD). An example can be found in the [Impala](#) sample connector.

Power BI will send the current user information to the gateway. The gateway will use Kerberos Constrained Delegation to invoke the query process as the impersonated user.

After making the above changes, the connector owner can test the following scenarios to validate functionality.

- In Power BI Desktop: Windows impersonation (current user)
- In Power BI Desktop: Windows impersonation using alternate credentials
- In the gateway: Windows impersonation using alternate credentials, by pre-configuring the data source with Windows account credentials in the Gateway Power BI Admin portal.

Connector developers can also use this procedure to test their implementation of Kerberos-based SSO.

1. Set up an on-premises data gateway with single sign-on enabled using instructions in the [Power BI Kerberos SSO documentation](#) article.
2. Validate the setup by testing with SQL Server and Windows accounts. Set up the [SQL Server Kerberos configuration manager](#). If you can use Kerberos SSO with SQL Server then your Power BI data gateway is properly set up to enable Kerberos SSO for other data sources as well.
3. Create an application (for example, a command-line tool) that connects to your server through your ODBC driver. Ensure that your application can use Windows authentication for the connection.
4. Modify your test application so that it can take a username (UPN) as an argument and use the [WindowsIdentity](#) constructor with it. Once complete, with the privileges granted to the gateway account set up in Step 1, you should be able to obtain the user's [AccessToken](#) property and [impersonate](#) this token.
5. Once you've made the changes to your application, ensure that you can use impersonation to load and connect to your service through the ODBC driver. Ensure that data can be retrieved. If you want to use native C or C++ code instead, you'll need to use [LsaLoginUser](#) to retrieve a token with just the username and use the [KERB_S4U_LOGON](#) option.

After this functionality is validated, Microsoft will make a change to thread the UPN from the Power BI Service down through the gateway. Once at the gateway, it will essentially act the same way as your test application to retrieve data.

Reach out to your Microsoft contact prior to starting work to learn more on how to request this change.

SAML SSO

SAML-based SSO is often not supported by end data sources and isn't a recommended approach. If your scenario requires the use of SAML-based SSO, reach out to your Microsoft contact or visit our documentation to [learn more](#).

Native database query support

Some Power Query connectors offer end users the ability to specify [native database queries](#) under **Advanced options** in the connection experience. Custom connector developers may be interested in offering native database query support in their connector.

Allowing users to run a custom SQL statement through an ODBC-based custom connector

Scenario: An end user can run custom SQL statements through their ODBC-based connector. The statement would be run in Import Mode, and there is no need for the transformations to fold.

Status: This feature isn't currently supported in our extensibility SDK. The product team is investigating the feasibility of this scenario. Without the extensibility of the security model, we don't recommend connectors expose native query functionality unless through one of the workarounds below.

Workarounds: If the data source is able to use the generic ODBC connector that currently supports native database query, this use is recommended. However, there may be cases where the generic ODBC connectivity scenario might not work, for example, if authentication needs to be implemented at the connector level.

In those cases, the connector developer can opt to use generic ODBC functionality with the [Odbc.Query](#) function instead of a custom connector. Unlike [Odbc.DataSource](#), which allows the custom connector to override driver settings and improve query folding behavior, [Odbc.Query](#) simply runs the query as provided and doesn't benefit from the custom connector wrapper.

If the data source can enforce read-only access and you'd like to proceed with exposing [Odbc.Query](#) functionality for your connector, we recommend that you provide a second data source function with its own

Publish record, and have two entries in the Get Data dialog (`DataSource.Database`, `DataSource.Query`). The `Odbc.Query` function would only support Import mode in Power BI, not Direct Query. The distinction is recommended, since combining `Odbc.Query` (which doesn't support query folding) and `Odbc.DataSource` (which does support query folding) may confuse end users. Also be sure to clearly distinguish the naming of your two Publish records to clearly communicate to users which function to use for native query.

If the data source doesn't enforce a read-only access, the connector must also leverage our [native database query security model](#) feature. Note that the Native Database Query prompt doesn't work in Visual Studio SDK. When you try to run `Extension.Query` in Visual Studio, you'll receive an error.

```
The evaluation requires a permission that has not been provided. Data source kind: 'Extension'. Data source path: 'test'. Permission kind: 'Native Query'
```

You'll need to conduct testing in Power BI Desktop.

The following connector code example exposes two functions, one that accepts a native query and one that doesn't.

```
section Extension;

// This function would call Odbc.DataSource
[DataSource.Kind = "Extension"]
shared Extension.DataSource = (server as text) => server;

// This function would call Odbc.Query
[DataSource.Kind = "Extension"]
shared Extension.Query = (server as text, query as text) => query;

Extension = [
    // MakeResourcePath overrides the default Data Source Path creation logic that serializes
    // all required parameters as a JSON encoded value. This is required to keep the data source
    // path the same between the Extension.DataSource and Extension.Query functions. Alternatively,
    // you can provide a function documentation type and use DataSource.Path = false for the query
    // parameter to exclude it from the data source path calculation.
    Type="Custom",
    MakeResourcePath = (server) => server,
    ParseResourcePath = (resource) => { resource },

    // Use NativeQuery to enable a Native Database Query prompt in the Power Query user experience.
    NativeQuery = (optional query) => query,
    Authentication=[Anonymous=null]
];
```

When evaluated, if the parameter names of the data source function can be mapped to the parameter names of the `NativeQuery` function on the data source definition, and the `NativeQuery` function returns text, then the call site generates a native query prompt. In this case, `Extension.Query("server", "select 1")` generates a challenge for the native query text `select 1`, while `Extension.DataSource("server")` won't generate a native query challenge.

Allowing users to use Direct Query over a custom SQL statement

Scenario: An end user can use Direct Query over native database queries.

Status: This feature is not currently supported in our extensibility SDK. The product team is investigating this scenario and expect that this scenario may eventually be possible for connectors with ODBC drivers and end data sources supporting ANSI SQL92 "pass through" mode.

Workarounds: None.

Handling Authentication

1/15/2022 • 12 minutes to read • [Edit Online](#)

Authentication Kinds

An extension can support one or more kinds of Authentication. Each authentication kind is a different type of credential. The authentication UI displayed to end users in Power Query is driven by the type of credential(s) that an extension supports.

The list of supported authentication types is defined as part of an extension's [Data Source Kind](#) definition. Each Authentication value is a record with specific fields. The following table lists the expected fields for each kind. All fields are required unless marked otherwise.

AUTHENTICATION KIND	FIELD	DESCRIPTION
Implicit		The Implicit (anonymous) authentication kind doesn't have any fields.
OAuth	StartLogin	<p>Function that provides the URL and state information for starting an OAuth flow.</p> <p>See the Implementing an OAuth Flow section below.</p>
	FinishLogin	Function that extracts the access_token and other properties related to the OAuth flow.
	Refresh	(optional) Function that retrieves a new access token from a refresh token.
	Logout	(optional) Function that invalidates the user's current access token.
	Label	(optional) A text value that allows you to override the default label for this AuthenticationKind.
Aad	AuthorizationUri	<p><code>text</code> value or function that returns the Azure AD authorization endpoint (example: "https://login.microsoftonline.com/common/oauth2/au").</p> <p>See the Azure Active Directory authentication section below.</p>
	Resource	<code>text</code> value or function that returns the Azure AD resource value for your service.
UsernamePassword	UsernameLabel	(optional) A text value to replace the default label for the <i>Username</i> text box on the credentials UI.
	PasswordLabel	(optional) A text value to replace the default label for the <i>Password</i> text box on the credentials UI.

AUTHENTICATION KIND	FIELD	DESCRIPTION
	Label	(optional) A text value that allows you to override the default label for this AuthenticationKind.
Windows	UsernameLabel	(optional) A text value to replace the default label for the <i>Username</i> text box on the credentials UI.
	PasswordLabel	(optional) A text value to replace the default label for the <i>Password</i> text box on the credentials UI.
	Label	(optional) A text value that allows you to override the default label for this AuthenticationKind.
Key	KeyLabel	(optional) A text value to replace the default label for the <i>API Key</i> text box on the credentials UI.
	Label	(optional) A text value that allows you to override the default label for this AuthenticationKind.

The sample below shows the Authentication record for a connector that supports OAuth, Key, Windows, Basic (Username and Password), and anonymous credentials.

Example:

```
Authentication = [
    OAuth = [
        StartLogin = StartLogin,
        FinishLogin = FinishLogin,
        Refresh = Refresh,
        Logout = Logout
    ],
    Key = [],
    UsernamePassword = [],
    Windows = [],
    Implicit = []
]
```

Accessing the Current Credentials

The current credentials can be retrieved using the `Extension.CurrentCredential()` function.

M data source functions that have been enabled for extensibility will automatically inherit your extension's credential scope. In most cases, you won't need to explicitly access the current credentials, however, there are exceptions, such as:

- Passing in the credential in a custom header or query string parameter (such as when you're using the API Key auth type)
- Setting connection string properties for ODBC or ADO.NET extensions
- Checking custom properties on an OAuth token
- Using the credentials as part of an OAuth v1 flow

The `Extension.CurrentCredential()` function returns a record object. The fields it contains will be authentication type specific. See the following table for details.

FIELD	DESCRIPTION	USED BY

FIELD	DESCRIPTION	USED BY
AuthenticationKind	Contains the name of the authentication kind assigned to this credential (UsernamePassword, OAuth, and so on).	All
Username	Username value	UsernamePassword, Windows
Password	Password value. Typically used with UsernamePassword, but it is also set for Key.	Key, UsernamePassword, Windows
access_token	OAuth access token value.	OAuth
Properties	A record containing other custom properties for a given credential. Typically used with OAuth to store additional properties (such as the refresh_token) returned with the access_token during the authentication flow.	OAuth
Key	The API key value. Note, the key value is also available in the Password field as well. By default, the mashup engine will insert this in an Authorization header as if this value were a basic auth password (with no username). If this is not the behavior you want, you must specify the ManualCredentials = true option in the options record.	Key
EncryptConnection	A logical value that determined whether to require an encrypted connection to the data source. This value is available for all Authentication Kinds, but will only be set if EncryptConnection is specified in the Data Source definition.	All

The following code sample accesses the current credential for an API key and uses it to populate a custom header (`x-APIKey`).

Example:

```
MyConnector.Raw = (_url as text) as binary =>
let
    apiKey = Extension.CurrentCredential()[Key],
    headers = [
        #"x-APIKey" = apiKey,
        Accept = "application/vnd.api+json",
        #"Content-Type" = "application/json"
    ],
    request = Web.Contents(_url, [ Headers = headers, ManualCredentials = true ])
in
    request
```

Implementing an OAuth Flow

The OAuth authentication type allows an extension to implement custom logic for their service. To do this, an extension will provide functions for `StartLogin` (returning the authorization URI to initiate the OAuth flow) and `FinishLogin` (exchanging the authorization code for an access token). Extensions can optionally implement `Refresh` (exchanging a refresh token for a new access token) and `Logout` (expiring the current refresh and access tokens) functions as well.

NOTE

Power Query extensions are evaluated in applications running on client machines. Data Connectors *should not* use confidential secrets in their OAuth flows, as users may inspect the extension or network traffic to learn the secret. See the [Proof Key for Code Exchange by OAuth Public Clients RFC](#) (also known as PKCE) for further details on providing flows that don't rely on shared secrets. A [sample](#) implementation of this flow can be found on our GitHub site.

There are two sets of OAuth function signatures; the original signature that contains a minimal number of parameters, and an advanced signature that accepts additional parameters. Most OAuth flows can be implemented using the original signatures. You can also mix and match signature types in your implementation. The function calls are matches based on the number of parameters (and their types). The parameter names are not taken into consideration.

See the [Github](#) sample for more details.

Original OAuth Signatures

```
StartLogin = (dataSourcePath, state, display) => ...;  
FinishLogin = (context, callbackUri, state) => ...;  
Refresh = (dataSourcePath, refreshToken) => ...;  
Logout = (accessToken) => ...;
```

Advanced OAuth Signatures

Notes about the advanced signatures:

- All signatures accept a `clientApplication` record value, which is reserved for future use.
- All signatures accept a `dataSourcePath` (also referred to as `resourceUrl` in most samples).
- The `Refresh` function accepts an `oldCredential` parameter, which is the previous `record` returned by your `FinishLogin` function (or previous call to `Refresh`).

```
StartLogin = (clientApplication, dataSourcePath, state, display) => ...;  
FinishLogin = (clientApplication, dataSourcePath, context, callbackUri, state) => ...;  
Refresh = (clientApplication, dataSourcePath, oldCredential) => ...;  
Logout = (clientApplication, dataSourcePath, accessToken) => ...;
```

Azure Active Directory authentication

The `Aad` authentication kind is a specialized version of OAuth for Azure Active Directory. It uses the same Azure AD client as the built-in Power Query connectors that support Organization Account authentication.

NOTE

If your data source requires scopes other than `user_impersonation`, or is incompatible with the use of `user_impersonation`, then you should use the `OAuth` authentication kind.

NOTE

If you implement your own OAuth flow for Azure AD, users who have enabled [Conditional Access](#) for their tenant might encounter issues when refreshing using the Power BI service. This won't impact gateway-based refresh, but would impact a certified connector that supports refresh from the Power BI service. Users might run into a problem stemming from the connector using a [public client application](#) when configuring web-based credentials through the Power BI service. The access token generated by this flow will ultimately be used on a different computer (that is, the Power BI service in an Azure data center, not on the company's network) than the one used to originally authenticate (that is, the computer of the user who configures the data source credentials on the company's network). The built-in `Aad` type works around this problem by using a different Azure AD client when configuring credentials in the Power BI service. This option won't be available to connectors that use the `OAuth` authentication kind.

Most connectors will need to provide values for the `AuthorizationUri` and `Resource` fields. Both fields can be `text` values, or a single argument function that returns a `text value`.

```
AuthorizationUri = "https://login.microsoftonline.com/common/oauth2/authorize"
```

```
AuthorizationUri = (dataSourcePath) => FunctionThatDeterminesAadEndpointFromDataSourcePath(dataSourcePath)
```

```
Resource = "77256ee0-fe79-11ea-adc1-0242ac120002" // Azure AD resource value for your service - Guid or URL
```

```
Resource = (dataSourcePath) => FunctionThatDeterminesResourceFromDataSourcePath(dataSourcePath)
```

Connectors that use a [Uri based identifier](#) do not need to provide a `Resource` value. By default, the value will be equal to the root path of the connector's Uri parameter. If the data source's Azure AD resource is different than the domain value (for example, it uses a GUID), then a `Resource` value needs to be provided.

Aad authentication kind samples

In this case, the data source supports global cloud Azure AD using the common tenant (no Azure B2B support).

```
Authentication = [
  Aad = [
    AuthorizationUri = "https://login.microsoftonline.com/common/oauth2/authorize",
    Resource = "77256ee0-fe79-11ea-adc1-0242ac120002" // Azure AD resource value for your service - Guid
  or URL
  ]
]
```

In this case, the data source supports tenant discovery based on OpenID Connect (OIDC) or similar protocol. This allows the connector to determine the correct Azure AD endpoint to use based on one or more parameters in the data source path. This dynamic discovery approach allows the connector to support Azure B2B.

```

// Implement this function to retrieve or calculate the service URL based on the data source path parameters
GetServiceRootFromDataSourcePath = (dataSourcePath) as text => ...;

GetAuthorizationUrlFromWwwAuthenticate = (url as text) as text =>
    let
        // Sending an unauthenticated request to the service returns
        // a 302 status with WWW-Authenticate header in the response. The value will
        // contain the correct authorization_uri.
        //
        // Example:
        // Bearer authorization_uri="https://login.microsoftonline.com/{tenant_guid}/oauth2/authorize"
        responseCodes = {302, 401},
        endpointResponse = Web.Contents(url, [
            ManualCredentials = true,
            ManualStatusHandling = responseCodes
        ])
    in
        if (List.Contains(responseCodes, Value.Metadata(endpointResponse)[Response.Status]?)) then
            let
                headers = Record.FieldOrDefault(Value.Metadata(endpointResponse), "Headers", []),
                wwwAuthenticate = Record.FieldOrDefault(headers, "WWW-Authenticate", ""),
                split = Text.Split(Text.Trim(wwwAuthenticate), " "),
                authorizationUri = List.First(List.Select(split, each Text.Contains(_, "authorization_uri="))), null)
            in
                if (authorizationUri <> null) then
                    // Trim and replace the double quotes inserted before the url
                    Text.Replace(Text.Trim(Text.Trim(Text.AfterDelimiter(authorizationUri, "="))), ","),
                    """", """
                else
                    error Error.Record("DataSource.Error", "Unexpected WWW-Authenticate header format or
value during authentication."), [
                        "#WWW-Authenticate" = wwwAuthenticate
                    ])
            else
                error Error.Unexpected("Unexpected response from server during authentication.");
        else
            <... snip ...>

        Authentication = [
            Aad = [
                AuthorizationUri = (dataSourcePath) =>
                    GetAuthorizationUrlFromWwwAuthenticate(
                        GetServiceRootFromDataSourcePath(dataSourcePath)
                    ),
                Resource = "https://myAadResourceValue.com", // Azure AD resource value for your service - Guid or
                URL
            ]
        ]
    
```

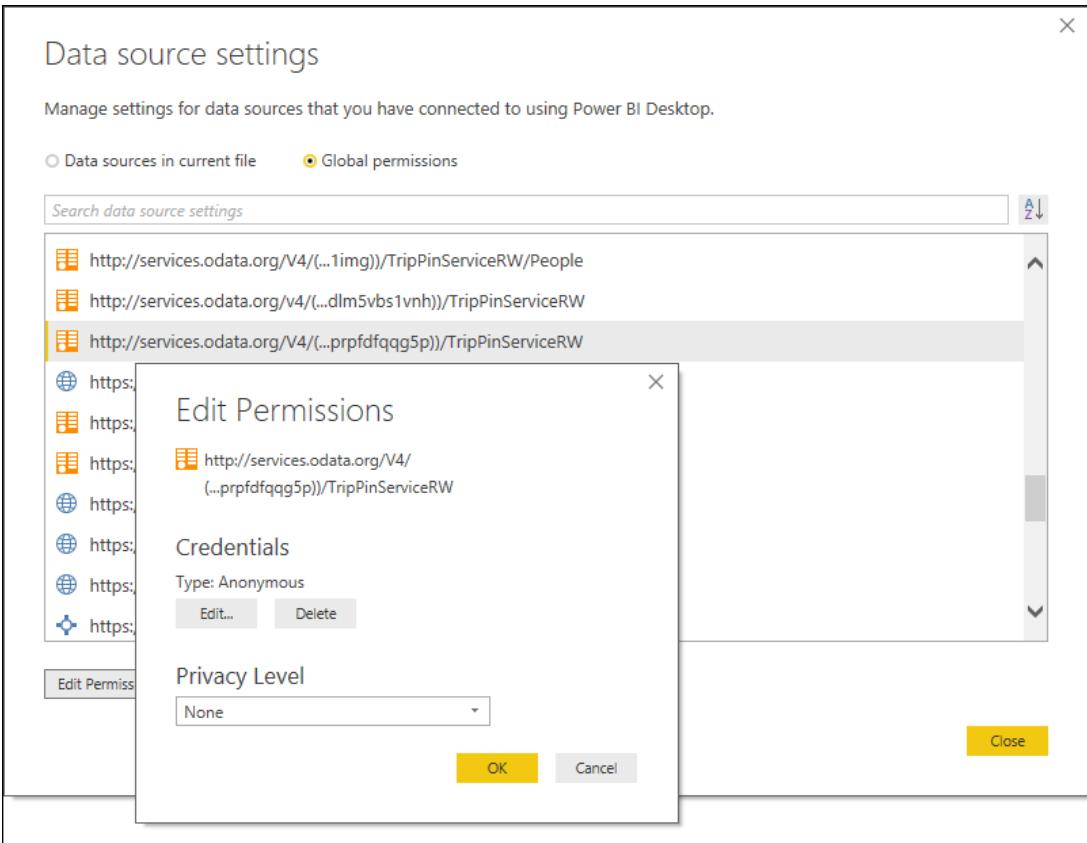
Data Source Paths

The M engine identifies a data source using a combination of its *Kind* and *Path*. When a data source is encountered during a query evaluation, the M engine will try to find matching credentials. If no credentials are found, the engine returns a special error that results in a credential prompt in Power Query.

The *Kind* value comes from the [Data Source Kind](#) definition.

The *Path* value is derived from the *required parameters* of your [data source function](#). Optional parameters aren't factored into the data source path identifier. As a result, all data source functions associated with a data source kind must have the same parameters. There's special handling for functions that have a single parameter of type [Uri.Type](#). See the [section below](#) for details.

You can see an example of how credentials are stored in the **Data source settings** dialog in Power BI Desktop. In this dialog, the Kind is represented by an icon, and the Path value is displayed as text.



NOTE

If you change your data source function's required parameters during development, previously stored credentials will no longer work (because the path values no longer match). You should delete any stored credentials any time you change your data source function parameters. If incompatible credentials are found, you may receive an error at runtime.

Data Source Path Format

The `Path` value for a data source is derived from the data source function's required parameters. Required parameters can be excluded from the path by adding `DataSource.Path = false` to the function's metadata ([see below](#)).

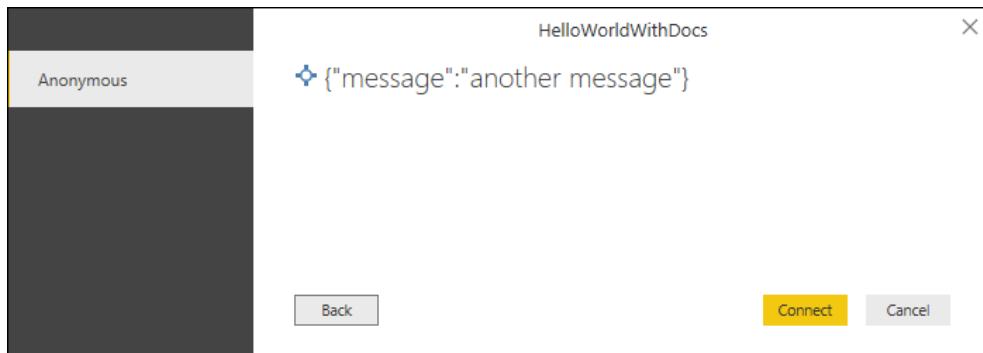
By default, you can see the actual string value in the Data source settings dialog in Power BI Desktop, and in the credential prompt. If the Data Source Kind definition has included a `Label` value, you'll see the label value instead.

For example, the data source function in the [HelloWorldWithDocs sample](#) has the following signature:

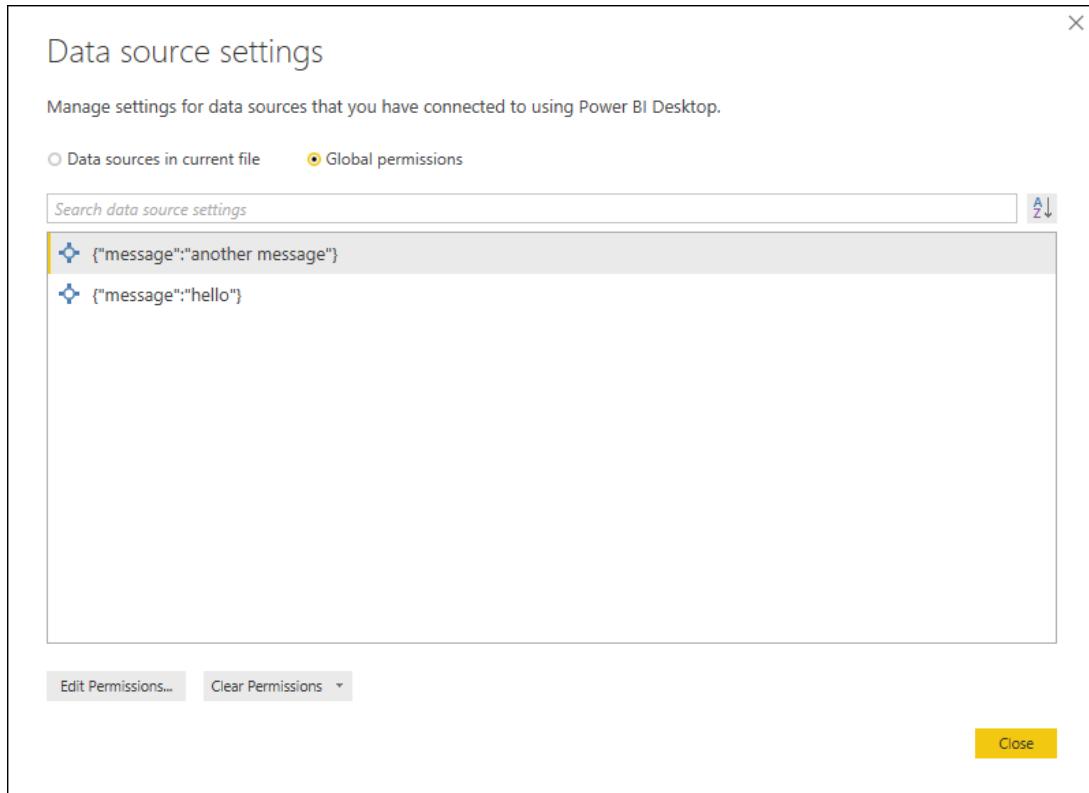
```
HelloWorldWithDocs.Contents = (message as text, optional count as number) as table => ...
```

The function has a single required parameter (`message`) of type `text`, and will be used to calculate the data source path. The optional parameter (`count`) would be ignored. The path would be displayed

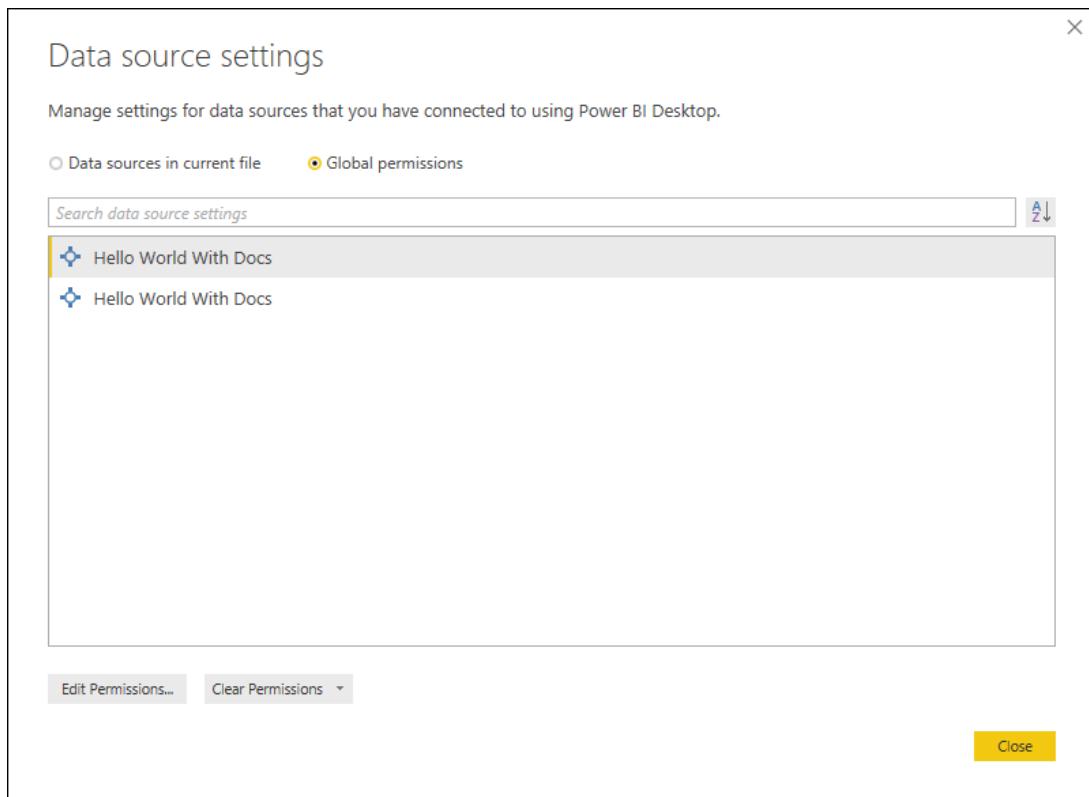
Credential prompt:



Data source settings UI:



When a Label value is defined, the data source path value wouldn't be shown:



NOTE

We currently recommend you *do not* include a Label for your data source if your function has required parameters, as users won't be able to distinguish between the different credentials they've entered. We are hoping to improve this in the future (that is, allowing data connectors to display their own custom data source paths).

Excluding Required Parameters from your Data Source Path

If you want a function parameter to be required, but not to be included as part of your data source path, you can add `DataSource.Path = false` to the function documentation metadata. This property can be added to one or

more parameters for your function. This field removes the value from your data source path (meaning that it will no longer be passed to your `TestConnection` function), so it should only be used for parameters that aren't required to identify your data source, or distinguish between user credentials.

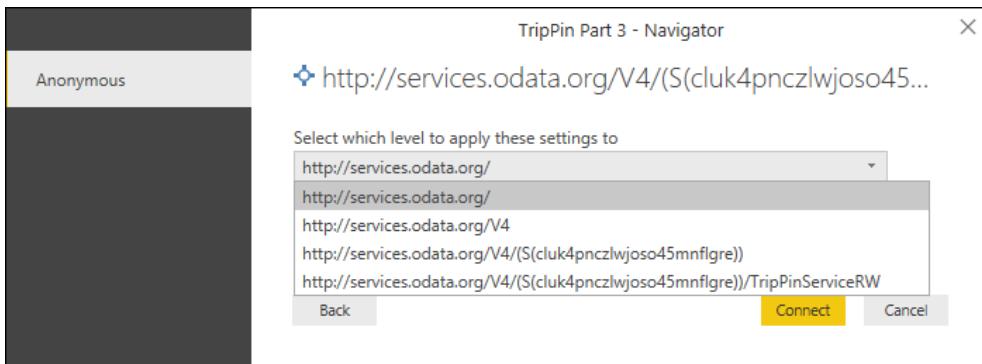
For example, the connector in the [HelloWorldWithDocs sample](#) would require different credentials for different `message` values. Adding `DataSource.Path = false` to the `message` parameter removes it from the data source path calculation, effectively making the connector a "singleton". All calls to `HelloWorldWithDocs.Contents` are treated as the same data source, and the user will only provide credentials once.

```
HelloWorldType = type function (
    message as (type text meta [
        DataSource.Path = false,
        Documentation.FieldCaption = "Message",
        Documentation.FieldDescription = "Text to display",
        Documentation.SampleValues = {"Hello world", "Hola mundo"}
    ]),
    optional count as (type number meta [
        Documentation.FieldCaption = "Count",
        Documentation.FieldDescription = "Number of times to repeat the message",
        Documentation.AllowedValues = { 1, 2, 3 }
    ]))
as table meta [
    Documentation.Name = "Hello - Name",
    Documentation.LongDescription = "Hello - Long Description",
    Documentation.Examples = {[[
        Description = "Returns a table with 'Hello world' repeated 2 times",
        Code = "HelloWorldWithDocs.Contents(\"Hello world\", 2)",
        Result = "#table({\"Column1\"}, {\"Hello world\"}, {\"Hello world\"})"
    ], [
        Description = "Another example, new message, new count!",
        Code = "HelloWorldWithDocs.Contents(\"Goodbye\", 1)",
        Result = "#table({\"Column1\"}, {\"Goodbye\"})"
    ]]}
];

```

Functions with a Uri parameter

Because data sources with a Uri based identifier are so common, there's special handling in the Power Query UI when dealing with Uri based data source paths. When a Uri-based data source is encountered, the credential dialog provides a drop-down allowing the user to select the base path, rather than the full path (and all paths in between).



As `Uri.Type` is an *ascribed type* rather than a *primitive type* in the M language, you'll need to use the `Value.ReplaceType` function to indicate that your text parameter should be treated as a Uri.

```
shared GithubSample.Contents = Value.ReplaceType(Github.Contents, type function (url as Uri.Type) as any);
```

Additional types of authentication

For information on additional types of authentication not covered in this article, such as Kerberos-based single sign-on, visit the [additional connector functionality](#) article to learn more.

Handling Data Access

1/15/2022 • 3 minutes to read • [Edit Online](#)

Data Source Functions

A Data Connector wraps and customizes the behavior of a [data source function in the M Library](#). For example, an extension for a REST API would make use of the `Web.Contents` function to make HTTP requests. Currently, a limited set of data source functions have been enabled to support extensibility.

- [Web.Contents](#)
- [OData.Feed](#)
- [Odbc.DataSource](#)
- [AdoDotNet.DataSource](#)
- [OleDb.DataSource](#)

Example:

```
[DataSource.Kind="HelloWorld", Publish="HelloWorld.Publish"]
shared HelloWorld.Contents = (optional message as text) =>
    let
        message = if (message <> null) then message else "Hello world"
    in
        message;
```

Data Source Kind

Functions marked as `shared` in your extension can be associated with a specific data source by including a `DataSource.Kind` metadata record on the function with the name of a Data Source definition record. The Data Source record defines the authentication types supported by your data source, and basic branding information (like the display name / label). The name of the record becomes is unique identifier.

Functions associated with a data source must have the same required function parameters (including name, type, and order). Functions for a specific Data Source Kind can only use credentials associated with that Kind. Credentials are identified at runtime by performing a lookup based on the combination of the function's required parameters. For more information about how credentials are identified, see [Data Source Paths](#).

Example:

```
HelloWorld = [
    Authentication = [
        Implicit = []
    ],
    Label = Extension.LoadString("DataSourceLabel")
];
```

Properties

The following table lists the fields for your Data Source definition record.

FIELD	TYPE	DETAILS
Authentication	record	Specifies one or more types of authentication supported by your data source. At least one kind is required. Each kind will be displayed as an option in the Power Query credential prompt. For more information, see Authentication Kinds .
Label	text	(optional) Friendly display name for this extension in credential dialogs.
SupportsEncryption	logical	(optional) When true, the UI will present the option to connect to the data source using an encrypted connection. This is typically used for data sources with a non-encrypted fallback mechanism (generally ODBC or ADO.NET based sources).

Publish to UI

Similar to the (Data Source)[#data-source-kind] definition record, the Publish record provides the Power Query UI the information it needs to expose this extension in the **Get Data** dialog.

Example:

```

HelloWorld.Publish = [
    Beta = true,
    ButtonText = { Extension.LoadString("FormulaTitle"), Extension.LoadString("FormulaHelp") },
    SourceImage = HelloWorld(Icons,
    SourceTypeImage = HelloWorld(Icons
];
HelloWorld(Icons = [
    Icon16 = { Extension.Contents("HelloWorld16.png"), Extension.Contents("HelloWorld20.png"),
Extension.Contents("HelloWorld24.png"), Extension.Contents("HelloWorld32.png") },
    Icon32 = { Extension.Contents("HelloWorld32.png"), Extension.Contents("HelloWorld40.png"),
Extension.Contents("HelloWorld48.png"), Extension.Contents("HelloWorld64.png") }
];

```

Properties

The following table lists the fields for your Publish record.

FIELD	TYPE	DETAILS
ButtonText	list	List of text items that will be displayed next to the data source's icon in the Power BI Get Data dialog.

FIELD	TYPE	DETAILS
Category	text	Where the extension should be displayed in the Get Data dialog. Currently the only category values with special handling are <code>Azure</code> and <code>Database</code> . All other values will end up under the Other category.
Beta	logical	(optional) When set to true, the UI will display a Preview/Beta identifier next to your connector name and a warning dialog that the implementation of the connector is subject to breaking changes.
LearnMoreUrl	text	(optional) Url to website containing more information about this data source or connector.
SupportsDirectQuery	logical	(optional) Enables Direct Query for your extension. This is currently only supported for ODBC extensions.
SourceImage	record	(optional) A record containing a list of binary images (sourced from the extension file using the Extension.Contents method). The record contains two fields (<code>Icon16</code> , <code>Icon32</code>), each with its own list. Each icon should be a different size.
SourceTypeImage	record	(optional) Similar to <code>SourceImage</code> , except the convention for many out of the box connectors is to display a sheet icon with the source specific icon in the bottom right corner. Having a different set of icons for <code>SourceTypeImage</code> is optional—many extensions simply reuse the same set of icons for both fields.

Enabling Direct Query for an ODBC based connector

1/15/2022 • 21 minutes to read • [Edit Online](#)

Overview

Using M's built-in [Odbc.DataSource](#) function is the recommended way to create custom connectors for data sources that have an existing ODBC driver and/or support a SQL query syntax. Wrapping the [Odbc.DataSource](#) function will allow your connector to inherit default query folding behavior based on the capabilities reported by your driver. This will enable the M engine to generate SQL statements based on filters and other transformations defined by the user within the Power Query experience, without having to provide this logic within the connector itself.

ODBC extensions can optionally enable Direct Query mode, allowing Power BI to dynamically generate queries at runtime without pre-caching the user's data model.

NOTE

Enabling Direct Query support raises the difficulty and complexity level of your connector. When Direct Query is enabled, Power BI will prevent the M engine from compensating for operations that cannot be fully pushed to the underlying data source.

This section builds on the concepts presented in the M Extensibility Reference, and assumes familiarity with the creation of a basic Data Connector.

Refer to the [SqlODBC sample](#) for most of the code examples in the sections below. Additional samples can be found in the ODBC samples directory.

ODBC Extensibility Functions

The M engine provides two ODBC related data source functions: [Odbc.DataSource](#), and [Odbc.Query](#).

The [Odbc.DataSource](#) function provides a default navigation table with all databases, tables, and views from your system, supports query folding, and allows for a range of customization options. The majority of ODBC based extensions will use this as their primary extensibility function. The function accepts two arguments—a connection string, and an options record to provide behavior overrides.

The [Odbc.Query](#) function allows you to execute SQL statements through an ODBC driver. It acts as a passthrough for query execution. Unlike the [Odbc.DataSource](#) function, it doesn't provide query folding functionality, and requires that SQL queries be provided by the connector (or end user). When building a custom connector, this function is typically used internally to run queries to retrieve metadata that might not be exposed through regular ODBC channels. The function accepts two arguments—a connection string, and a SQL query.

Parameters for your Data Source Function

Custom connectors can accept any number of function arguments, but to remain consistent with the built-in data source functions shipped with Power Query, the following guidelines are recommended:

- Require the minimal set of parameters used to establish a connection to your server. The less parameters end users need to provide, the easier your connector will be to use.

- Although you can define parameters with a fixed number of values (that is, a dropdown list in the UI), parameters are entered before the user is authenticated. Any values that can be discovered programmatically after the user is authenticated (such as catalog or database name) should be selectable through the Navigator. The default behavior for the [Odbc.DataSource](#) function will be to return a hierarchical navigation table consisting of Catalog (Database), Schema, and Table names, although this can be overridden within your connector.
- If you feel your users will typically know what values to enter for items they would select from the Navigator (such as the database name), make these parameters optional. Parameters that can be discovered programmatically should not be made required.
- The last parameter for your function should be an optional record called "options". This parameter typically allows advanced users to set common ODBC related properties (such as CommandTimeout), set behavior overrides specific to your connector, and allows for future extensibility without impacting backwards compatibility for your function.
- Security/credential related arguments MUST never be part of your data source function parameters, as values entered in the connect dialog will be persisted to the user's query. Credential related parameters should be specified as part of the connector's supported Authentication methods.

By default, all required parameters for your data source function are factored into the Data Source Path value used to identify user credentials.

Note that while the UI for the built-in [Odbc.DataSource](#) function provides a dropdown that allows the user to select a DSN, this functionality is not available through extensibility. If your data source configuration is complex enough to require a fully customizable configuration dialog, it's recommended you require your end users to pre-configure a system DSN, and have your function take in the DSN name as a text field.

Parameters for Odbc.DataSource

The [Odbc.DataSource](#) function takes two parameters—a connectionString for your driver, and an options record that lets you override various driver behaviors. Through the options record you can override capabilities and other information reported by the driver, control the navigator behavior, and affect the SQL queries generated by the M engine.

The supported options records fields fall into two categories—those that are public / always available, and those that are only available in an extensibility context.

The following table describes the public fields in the options record.

FIELD	DESCRIPTION
CommandTimeout	A duration value that controls how long the server-side query is allowed to run before it's cancelled. Default: 10 minutes
ConnectionTimeout	A duration value that controls how long to wait before abandoning an attempt to make a connection to the server. Default: 15 seconds

FIELD	DESCRIPTION
CreateNavigationProperties	<p>A logical value that sets whether to generate navigation properties on the returned tables. Navigation properties are based on foreign key relationships reported by the driver, and show up as "virtual" columns that can be expanded in the query editor, creating the appropriate join.</p> <p>If calculating foreign key dependencies is an expensive operation for your driver, you may want to set this value to false.</p> <p>Default: true</p>
HierarchicalNavigation	<p>A logical value that sets whether to view the tables grouped by their schema names. When set to false, tables will be displayed in a flat list under each database.</p> <p>Default: false</p>
SqlCompatibleWindowsAuth	<p>A logical value that determines whether to produce a SQL Server compatible connection string when using Windows Authentication—Trusted_Connection=Yes.</p> <p>If your driver supports Windows Authentication, but requires additional or alternative settings in your connection string, you should set this value to false and use the CredentialConnectionString option record field described below.</p> <p>Default: true</p>

The following table describes the options record fields that are only available through extensibility. Fields that aren't simple literal values are described in subsequent sections.

FIELD	DESCRIPTION
AstVisitor	<p>A record containing one or more overrides to control SQL query generation. The most common usage of this field is to provide logic to generate a LIMIT/OFFSET clause for drivers that don't support TOP.</p> <p>Fields include:</p> <ul style="list-style-type: none"> • Constant • LimitClause <p>See the AstVisitor section for more information.</p>

FIELD	DESCRIPTION
CancelQueryExplicitly	<p>A logical value that instructs the M engine to explicitly cancel any running calls through the ODBC driver before terminating the connection to the ODBC server.</p> <p>This field is useful in situations where query execution is managed independently of the network connections to the server, for example in some Spark deployments. In most cases, this value doesn't need to be set because the query in the server is canceled when the network connection to the server is terminated.</p> <p>Default: false</p>
ClientConnectionPooling	<p>A logical value that enables client-side connection pooling for the ODBC driver. Most drivers will want to set this value to true.</p> <p>Default: false</p>
CredentialConnectionString	<p>A text or record value used to specify credential related connection string properties.</p> <p>See the Credential section for more information.</p>
HideNativeQuery	<p>A logical value that controls whether your connector allows native SQL statements to be passed in by a query using the Value.NativeQuery() function.</p> <p>Note: this functionality is currently not exposed in the Power Query user experience. Users would need to manually edit their queries to take advantage of this capability.</p> <p>Default: false</p>
ImplicitTypeConversions	<p>A table value containing implicit type conversions supported by your driver or backend server. Values in this table are additive to the conversions reported by the driver itself.</p> <p>This field is typically used in conjunction with the SQLGetTypeInfo field when overriding data type information reported by the driver.</p> <p>See the ImplicitTypeConversions section for more information.</p>
OnError	<p>An error handling function that receives an errorRecord parameter of type record.</p> <p>Common uses of this function include handling SSL connection failures, providing a download link if your driver isn't found on the system, and reporting authentication errors.</p> <p>See the OnError section for more information.</p>

FIELD	DESCRIPTION
SoftNumbers	<p>Allows the M engine to select a compatible data type when conversion between two specific numeric types isn't declared as supported in the SQL_CONVERT_* capabilities.</p> <p>Default: false</p>
SqlCapabilities	<p>A record providing various overrides of driver capabilities, and a way to specify capabilities that aren't expressed through ODBC 3.8.</p> <p>See the SqlCapabilities section for more information.</p>
SQLColumns	<p>A function that allows you to modify column metadata returned by the SQLColumns function.</p> <p>See the SQLColumns section for more information.</p>
SQLGetFunctions	<p>A record that allows you to override values returned by calls to SQLGetFunctions.</p> <p>A common use of this field is to disable the use of parameter binding, or to specify that generated queries should use CAST rather than CONVERT.</p> <p>See the SQLGetFunctions section for more information.</p>
SQLGetInfo	<p>A record that allows you to override values returned by calls to SQLGetInfo.</p> <p>See the SQLGetInfo section for more information.</p>
SQLGetTypeInfo	<p>A table, or function that returns a table, that overrides the type information returned by SQLGetTypeInfo.</p> <p>When the value is set to a table, the value completely replaces the type information reported by the driver. SQLGetTypeInfo won't be called.</p> <p>When the value is set to a function, your function will receive the result of the original call to SQLGetTypeInfo, allowing you to modify the table.</p> <p>This field is typically used when there's a mismatch between data types reported by SQLGetTypeInfo and SQLColumns.</p> <p>See the SQLGetTypeInfo section for more information.</p>
SQLTables	<p>A function that allows you to modify the table metadata returned by a call to SQLTables.</p> <p>See the SQLTables section for more information.</p>

FIELD	DESCRIPTION
TolerateConcatOverflow	<p>Allows concatenation of text values to occur even if the result might be truncated to fit within the range of an available type.</p> <p>For example, when concatenating a VARCHAR(4000) field with a VARCHAR(4000) field on a system that supports a maximize VARCHAR size of 4000 and no CLOB type, the concatenation will be folded even though the result might get truncated.</p> <p>Default: false</p>
UseEmbeddedDriver	<p>(internal use): A logical value that controls whether the ODBC driver should be loaded from a local directory (using new functionality defined in the ODBC 4.0 specification). This is generally only set by connectors created by Microsoft that ship with Power Query.</p> <p>When set to false, the system ODBC driver manager will be used to locate and load the driver.</p> <p>Most connectors should not need to set this field.</p> <p>Default: false</p>

Overriding AstVisitor

The AstVisitor field is set through the [Odbc.DataSource](#) options record. It's used to modify SQL statements generated for specific query scenarios.

NOTE

Drivers that support `LIMIT` and `OFFSET` clauses (rather than `TOP`) will want to provide a LimitClause override for AstVisitor.

Constant

Providing an override for this value has been deprecated and may be removed from future implementations.

LimitClause

This field is a function that receives two `Int64.Type` arguments (skip, take), and returns a record with two text fields (Text, Location).

```
LimitClause = (skip as nullable number, take as number) as record => ...
```

The skip parameter is the number of rows to skip (that is, the argument to OFFSET). If an offset is not specified, the skip value will be null. If your driver supports `LIMIT`, but does not support `OFFSET`, the LimitClause function should return an unimplemented error (...) when skip is greater than 0.

The take parameter is the number of rows to take (that is, the argument to LIMIT).

The `Text` field of the result contains the SQL text to add to the generated query.

The `Location` field specifies where to insert the clause. The following table describes supported values.

Value	Description	Example
AfterQuerySpecification	<p>LIMIT clause is put at the end of the generated SQL.</p> <p>This is the most commonly supported LIMIT syntax.</p>	<pre>SELECT a, b, c FROM table WHERE a > 10 LIMIT 5</pre>
BeforeQuerySpecification	LIMIT clause is put before the generated SQL statement.	<pre>LIMIT 5 ROWS SELECT a, b, c FROM table WHERE a > 10</pre>
AfterSelect	LIMIT goes after the SELECT statement, and after any modifiers (such as DISTINCT).	<pre>SELECT DISTINCT LIMIT 5 a, b, c FROM table WHERE a > 10</pre>
AfterSelectBeforeModifiers	LIMIT goes after the SELECT statement, but before any modifiers (such as DISTINCT).	<pre>SELECT LIMIT 5 DISTINCT a, b, c FROM table WHERE a > 10</pre>

The following code snippet provides a LimitClause implementation for a driver that expects a LIMIT clause, with an optional OFFSET, in the following format: [OFFSET <offset> ROWS] LIMIT <row_count>

```
LimitClause = (skip, take) =>
  let
    offset = if (skip > 0) then Text.Format("OFFSET #{0} ROWS", {skip}) else "",
    limit = if (take <> null) then Text.Format("LIMIT #{0}", {take}) else ""
  in
  [
    Text = Text.Format("#{0} #{1}", {offset, limit}),
    Location = "AfterQuerySpecification"
  ]
```

The following code snippet provides a LimitClause implementation for a driver that supports LIMIT, but not OFFSET. Format: LIMIT <row_count> .

```
LimitClause = (skip, take) =>
  if (skip > 0) then error "Skip/Offset not supported"
  else
  [
    Text = Text.Format("LIMIT #{0}", {take}),
    Location = "AfterQuerySpecification"
  ]
```

Overriding SqlCapabilities

FIELD	DETAILS
FractionalSecondsScale	<p>A number value ranging from 1 to 7 that indicates the number of decimal places supported for millisecond values. This value should be set by connectors that want to enable query folding over datetime values.</p> <p>Default: null</p>
PrepareStatements	<p>A logical value that indicates that statements should be prepared using SQLPrepare.</p> <p>Default: false</p>
SupportsTop	<p>A logical value that indicates the driver supports the TOP clause to limit the number of returned rows.</p> <p>Default: false</p>
StringLiteralEscapeCharacters	<p>A list of text values that specify the character(s) to use when escaping string literals and LIKE expressions.</p> <p>Ex. {"\r\n"}</p> <p>Default: null</p>
SupportsDerivedTable	<p>A logical value that indicates the driver supports derived tables (sub-selects).</p> <p>This value is assumed to be true for drivers that set their conformance level to SQL_SC_SQL92_FULL (reported by the driver or overridden with the Sql92Conformance setting (see below)). For all other conformance levels, this value defaults to false.</p> <p>If your driver doesn't report the SQL_SC_SQL92_FULL compliance level, but does support derived tables, set this value to true.</p> <p>Note that supporting derived tables is required for many Direct Query scenarios.</p>
SupportsNumericLiterals	<p>A logical value that indicates whether the generated SQL should include numeric literals values. When set to false, numeric values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsStringLiterals	<p>A logical value that indicates whether the generated SQL should include string literals values. When set to false, string values will always be specified using Parameter Binding.</p> <p>Default: false</p>

FIELD	DETAILS
SupportsOdbcDateLiterals	<p>A logical value that indicates whether the generated SQL should include date literals values. When set to false, date values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimeLiterals	<p>A logical value that indicates whether the generated SQL should include time literals values. When set to false, time values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimestampLiterals	<p>A logical value that indicates whether the generated SQL should include timestamp literals values. When set to false, timestamp values will always be specified using Parameter Binding.</p> <p>Default: false</p>

Overriding SQLColumns

`SQLColumns` is a function handler that receives the results of an ODBC call to [SQLColumns](#). The source parameter contains a table with the data type information. This override is typically used to fix up data type mismatches between calls to `SQLGetTypeInfo` and `SQLColumns`.

For details of the format of the source table parameter, go to [SQLColumns Function](#).

Overriding SQLGetFunctions

This field is used to override SQLFunctions values returned by an ODBC driver. It contains a record whose field names are equal to the FunctionId constants defined for the ODBC [SQLGetFunctions](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#).

FIELD	DETAILS
SQL_CONVERT_FUNCTIONS	<p>Indicates which function(s) are supported when doing type conversions. By default, the M Engine will attempt to use the CONVERT function. Drivers that prefer the use of CAST can override this value to report that only SQL_FN_CVT_CAST (numeric value of 0x2) is supported.</p>
SQL_API_SQLBINDCOL	<p>A logical (true/false) value that indicates whether the Mashup Engine should use the SQLBindCol API when retrieving data. When set to false, SQLGetData is used instead.</p> <p>Default: false</p>

The following code snippet provides an example explicitly telling the M engine to use CAST rather than CONVERT.

```

SQLGetFunctions = [
    SQL_CONVERT_FUNCTIONS = 0x2 /* SQL_FN_CVT_CAST */
]

```

Overriding SQLGetInfo

This field is used to override SQLGetInfo values returned by an ODBC driver. It contains a record whose fields are names equal to the InfoType constants defined for the ODBC [SQLGetInfo](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#). The full list of InfoTypes that are checked can be found in the Mashup Engine trace files.

The following table contains commonly overridden SQLGetInfo properties:

FIELD	DETAILS
SQL_SQL_CONFORMANCE	<p>An integer value that indicates the level of SQL-92 supported by the driver:</p> <ul style="list-style-type: none"> (1) SQL_SC_SQL92_ENTRY = Entry level SQL-92 compliant. (2) SQL_SC_FIPS127_2_TRANSITIONAL = FIPS 127-2 transitional level compliant. (4) SQL_SC_SQL92_INTERMEDIATE = Intermediate level SQL-92 compliant. (8) SQL_SC_SQL92_FULL = Full level SQL-92 compliant. <p>Note that in Power Query scenarios, the connector will be used in a Read Only mode. Most drivers will want to report a SQL_SC_SQL92_FULL compliance level, and override specific SQL generation behavior using the SQLGetInfo and SQLGetFunctions properties.</p>
SQL_SQL92_PREDICATES	<p>A bitmask enumerating the predicates supported in a SELECT statement, as defined in SQL-92.</p> <p>See the SQL_SP_* constants in the ODBC specification.</p>
SQL_AGGREGATE_FUNCTIONS	<p>A bitmask enumerating support for aggregation functions.</p> <p>SQL_AF_ALL SQL_AF_AVG SQL_AF_COUNT SQL_AF_DISTINCT SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM</p> <p>See the SQL_AF_* constants in the ODBC specification.</p>

FIELD	DETAILS
SQL_GROUP_BY	<p>A integer value that specifies the relationship between the columns in the GROUP BY clause and the non-aggregated columns in the select list:</p> <p>SQL_GB_COLLATE = A COLLATE clause can be specified at the end of each grouping column.</p> <p>SQL_GB_NOT_SUPPORTED = GROUP BY clauses are not supported.</p> <p>SQL_GB_GROUP_BY_EQUALS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It cannot contain any other columns. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT.</p> <p>SQL_GB_GROUP_BY_CONTAINS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It can contain columns that are not in the select list. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>SQL_GB_NO_RELATION = The columns in the GROUP BY clause and the select list are not related. The meaning of non-grouped, non-aggregated columns in the select list is data source-dependent. For example, SELECT DEPT, SALARY FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>See the SQL_GB_* constants in the ODBC specification.</p>

The following helper function can be used to create bitmask values from a list of integer values:

```
Flags = (flags as list) =>
    let
        Loop = List.Generate(
            ()=> [i = 0, Combined = 0],
            each [i] < List.Count(flags),
            each [i = [i]+1, Combined =*Number.BitwiseOr([Combined], flags{i})],
            each [Combined]),
        Result = List.Last(Loop, 0)
    in
        Result;
```

Overriding SQLGetTypeInfo

`SQLGetTypeInfo` can be specified in two ways:

- A fixed `table` value that contains the same type information as an ODBC call to `sqlGetTypeInfo`.
- A function that accepts a table argument, and returns a table. The argument will contain the original results of the ODBC call to `sqlGetTypeInfo`. Your function implementation can modify/add to this table.

The first approach is used to completely override the values returned by the ODBC driver. The second approach is used if you want to add to or modify these values.

For details of the format of the types table parameter and expected return value, see the [SQLGetTypeInfo function reference](#).

SQLGetTypeInfo using a static table

The following code snippet provides a static implementation for SQLGetTypeInfo.

```

SQLGetTypeInfo = #table(
    { "TYPE_NAME",      "DATA_TYPE", "COLUMN_SIZE", "LITERAL_PREF", "LITERAL_SUFFIX", "CREATE_PARAS",
    "NULLABLE", "CASE_SENSITIVE", "SEARCHABLE", "UNSIGNED_ATTRIBUTE", "FIXED_PREC_SCALE", "AUTO_UNIQUE_VALUE",
    "LOCAL_TYPE_NAME", "MINIMUM_SCALE", "MAXIMUM_SCALE", "SQL_DATA_TYPE", "SQL_DATETIME_SUB", "NUM_PREC_RADIX",
    "INTERNAL_PRECISION", "USER_DATA_TYPE" }, {
        { "char",           1,       65535,      "",      "",      "max. length",
1,           1,           3,       null,      0,      null,
"char",       null,       null,      -8,      null,
0,           0,           0,       null,      10,      null,
        { "int8",          -5,       19,      "",      "",      null,
1,           0,           2,       0,      -5,      0,
"int8",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "bit",           -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bit",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "bool",          -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bool",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "date",           9,       10,      "",      "",      null,
1,           0,           2,       null,      0,      null,
"date",       null,       null,      9,      1,
0,           0,           0,       null,      null,
        { "numeric",         3,       28,      null,      null,      null,
1,           0,           2,       0,      0,
"numeric",       0,       0,       null,      null,
0,           0,           0,       null,      10,
        { "float8",          8,       15,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "float8",          6,       17,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "uuid",          -11,       37,      null,      null,      null,
1,           0,           2,       null,      0,
"uuid",       null,       null,      -11,      null,
0,           0,           0,       null,      null,
        { "int4",            4,       10,      null,      null,      null,
1,           0,           2,       0,      0,
"int4",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "text",            -1,       65535,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"text",       null,       null,      -10,      null,
0,           0,           0,       null,      null,
        { "lo",              -4,       255,      "",      "",      null,
1,           0,           2,       null,      0,
"lo",       null,       null,      -4,      null,
0,           0,           0,       null,      null,
        { "numeric",          2,       28,      null,      null,      "precision, scale",
1,           0,           2,       0,      10,
"numeric",       0,       6,       2,      null,
0,           0,           0,       null,      10,
        { "float4",            7,       9,       null,      null,      null,
1,           0,           2,       0,      10,
"float4",       null,       null,      7,      null,
0,           0,           0,       null,      2,
        { "int2",            5,       19,      null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "int2",            -6,       5,       null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
    }
}

```

```

0,          0      },
0, { "timestamp", 11, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "date", 91, 10, null, "", "", null,
1, 0, 2, null, 9, 1, null,
"date", null, }, 1,
0, 0 }, null,
{ "timestamp", 93, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "bytea", -3, 255, null, "", "", null,
1, 0, 2, null, 0, null,
"bytea", null, }, null,
0, 0 }, null,
{ "varchar", 12, 65535, null, "", "", "max. length",
1, 0, 2, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "char", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"char", null, }, null,
0, 0 }, null,
{ "text", -10, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"text", null, }, null,
0, 0 }, null,
{ "varchar", -9, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "bpchar", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"bpchar", null, }, null,
0, 0 } }
);

```

SQLGetTypeInfo using a function

The following code snippets append the `bpchar` type to the existing types returned by the driver.

```

SQLGetTypeInfo = (types as table) as table =>
let
    newTypes = #table(
    {
        "TYPE_NAME",
        "DATA_TYPE",
        "COLUMN_SIZE",
        "LITERAL_PREF",
        "LITERAL_SUFFIX",
        "CREATE_PARAS",
        "NULLABLE",
        "CASE_SENSITIVE",
        "SEARCHABLE",
        "UNSIGNED_ATTRIBUTE",
        "FIXED_PREC_SCALE",
        "AUTO_UNIQUE_VALUE",
        "LOCAL_TYPE_NAME",
        "MINIMUM_SCALE",
        "MAXIMUM_SCALE",
        "SQL_DATA_TYPE",
        "SQL_DATETIME_SUB",
        "NUM_PREC_RADIX",
        "INTERNAL_PRECISION",
        "USER_DATA_TYPE"
    },
    // we add a new entry for each type we want to add
    {
        {
            "bpchar",
            -8,
            65535,
            """",
            """",
            "max. length",
            1,
            1,
            3,
            null,
            0,
            null,
            "bpchar",
            null,
            null,
            -9,
            null,
            null,
            0,
            0
        }
    )),
    append = Table.Combine({types, newTypes})
in
    append;

```

Setting the Connection String

The connection string for your ODBC driver is set using the first argument to the [Odbc.DataSource](#) and/or [Odbc.Query](#) functions. The value can be text, or an M record. When using the record, each field in the record will become a property in the connection string. All connection strings will require a Driver field (or DSN field if you require users to pre-configure a system level DSN). Credential related properties will be set separately (see below). Other properties will be driver specific.

The code snippet below shows the definition of a new data source function, creation of the `ConnectionString` record, and invocation of the [Odbc.DataSource](#) function.

```
[DataSource.Kind="SqlODBC", Publish="SqlODBC.Publish"]
shared SqlODBC.Contents = (server as text) =>
    let
        ConnectionString = [
            Driver = "SQL Server Native Client 11.0",
            Server = server,
            MultiSubnetFailover = "Yes",
            ApplicationIntent = "ReadOnly",
            APP = "PowerBICustomConnector"
        ],
        OdbcDatasource = Odbc.DataSource(ConnectionString)
    in
        OdbcDatasource;
```

Troubleshooting and Testing

To enable tracing in Power BI Desktop:

1. Go to **File > Options and settings > Options**.
2. Select on the **Diagnostics** tab.
3. Select the **Enable tracing** option.
4. Select the **Open traces folder** link (should be `%LOCALAPPDATA%/Microsoft/Power BI Desktop/Traces`).
5. Delete existing trace files.
6. Perform your tests.
7. Close Power BI Desktop to ensure all log files are flushed to disk.

Here are steps you can take for initial testing in Power BI Desktop:

1. Close Power BI Desktop.
2. Clear your trace directory.
3. Open Power BI desktop, and enable tracing.
4. Connect to your data source, and select Direct Query mode.
5. Select a table in the navigator, and select **Edit**.
6. Manipulate the query in various ways, including:

- Take the First N rows (for example, 10).
- Set equality filters on different data types (int, string, bool, and so on).
- Set other range filters (greater than, less than).
- Filter on NULL / NOT NULL.
- Select a sub-set of columns.
- Aggregate / Group By different column combinations.
- Add a column calculated from other columns ($[C] = [A] + [B]$).
- Sort on one column, multiple columns. 7. Expressions that fail to fold will result in a warning bar. Note the failure, remove the step, and move to the next test case. Details about the cause of the failure should be emitted to the trace logs. 8. Close Power BI Desktop. 9. Copy the trace files to a new directory. 10. Use the recommend Power BI workbook to parse and analyze the trace files.

Once you have simple queries working, you can then try Direct Query scenarios (for example, building reports in the Report Views). The queries generated in Direct Query mode will be significantly more complex (that is, use of sub-selects, COALESCE statements, and aggregations).

Concatenation of strings in Direct Query mode

The M engine does basic type size limit validation as part of its query folding logic. If you are receiving a folding

error when trying to concatenate two strings that potentially overflow the maximum size of the underlying database type:

1. Ensure that your database can support up-conversion to CLOB types when string concat overflow occurs.
2. Set the `TolerateConcatOverflow` option for `Odbc.DataSource` to `true`.

The [DAX CONCATENATE function](#) is currently not supported by Power Query/ODBC extensions. Extension authors should ensure string concatenation works through the query editor by adding calculated columns (`[stringCol1] & [stringCol2]`). When the capability to fold the CONCATENATE operation is added in the future, it should work seamlessly with existing extensions.

Enabling Direct Query for an ODBC based connector

1/15/2022 • 21 minutes to read • [Edit Online](#)

Overview

Using M's built-in [Odbc.DataSource](#) function is the recommended way to create custom connectors for data sources that have an existing ODBC driver and/or support a SQL query syntax. Wrapping the [Odbc.DataSource](#) function will allow your connector to inherit default query folding behavior based on the capabilities reported by your driver. This will enable the M engine to generate SQL statements based on filters and other transformations defined by the user within the Power Query experience, without having to provide this logic within the connector itself.

ODBC extensions can optionally enable Direct Query mode, allowing Power BI to dynamically generate queries at runtime without pre-caching the user's data model.

NOTE

Enabling Direct Query support raises the difficulty and complexity level of your connector. When Direct Query is enabled, Power BI will prevent the M engine from compensating for operations that cannot be fully pushed to the underlying data source.

This section builds on the concepts presented in the M Extensibility Reference, and assumes familiarity with the creation of a basic Data Connector.

Refer to the [SqlODBC sample](#) for most of the code examples in the sections below. Additional samples can be found in the ODBC samples directory.

ODBC Extensibility Functions

The M engine provides two ODBC related data source functions: [Odbc.DataSource](#), and [Odbc.Query](#).

The [Odbc.DataSource](#) function provides a default navigation table with all databases, tables, and views from your system, supports query folding, and allows for a range of customization options. The majority of ODBC based extensions will use this as their primary extensibility function. The function accepts two arguments—a connection string, and an options record to provide behavior overrides.

The [Odbc.Query](#) function allows you to execute SQL statements through an ODBC driver. It acts as a passthrough for query execution. Unlike the [Odbc.DataSource](#) function, it doesn't provide query folding functionality, and requires that SQL queries be provided by the connector (or end user). When building a custom connector, this function is typically used internally to run queries to retrieve metadata that might not be exposed through regular ODBC channels. The function accepts two arguments—a connection string, and a SQL query.

Parameters for your Data Source Function

Custom connectors can accept any number of function arguments, but to remain consistent with the built-in data source functions shipped with Power Query, the following guidelines are recommended:

- Require the minimal set of parameters used to establish a connection to your server. The less parameters end users need to provide, the easier your connector will be to use.

- Although you can define parameters with a fixed number of values (that is, a dropdown list in the UI), parameters are entered before the user is authenticated. Any values that can be discovered programmatically after the user is authenticated (such as catalog or database name) should be selectable through the Navigator. The default behavior for the [Odbc.DataSource](#) function will be to return a hierarchical navigation table consisting of Catalog (Database), Schema, and Table names, although this can be overridden within your connector.
- If you feel your users will typically know what values to enter for items they would select from the Navigator (such as the database name), make these parameters optional. Parameters that can be discovered programmatically should not be made required.
- The last parameter for your function should be an optional record called "options". This parameter typically allows advanced users to set common ODBC related properties (such as CommandTimeout), set behavior overrides specific to your connector, and allows for future extensibility without impacting backwards compatibility for your function.
- Security/credential related arguments MUST never be part of your data source function parameters, as values entered in the connect dialog will be persisted to the user's query. Credential related parameters should be specified as part of the connector's supported Authentication methods.

By default, all required parameters for your data source function are factored into the Data Source Path value used to identify user credentials.

Note that while the UI for the built-in [Odbc.DataSource](#) function provides a dropdown that allows the user to select a DSN, this functionality is not available through extensibility. If your data source configuration is complex enough to require a fully customizable configuration dialog, it's recommended you require your end users to pre-configure a system DSN, and have your function take in the DSN name as a text field.

Parameters for Odbc.DataSource

The [Odbc.DataSource](#) function takes two parameters—a connectionString for your driver, and an options record that lets you override various driver behaviors. Through the options record you can override capabilities and other information reported by the driver, control the navigator behavior, and affect the SQL queries generated by the M engine.

The supported options records fields fall into two categories—those that are public / always available, and those that are only available in an extensibility context.

The following table describes the public fields in the options record.

FIELD	DESCRIPTION
CommandTimeout	A duration value that controls how long the server-side query is allowed to run before it's cancelled. Default: 10 minutes
ConnectionTimeout	A duration value that controls how long to wait before abandoning an attempt to make a connection to the server. Default: 15 seconds

FIELD	DESCRIPTION
CreateNavigationProperties	<p>A logical value that sets whether to generate navigation properties on the returned tables. Navigation properties are based on foreign key relationships reported by the driver, and show up as "virtual" columns that can be expanded in the query editor, creating the appropriate join.</p> <p>If calculating foreign key dependencies is an expensive operation for your driver, you may want to set this value to false.</p> <p>Default: true</p>
HierarchicalNavigation	<p>A logical value that sets whether to view the tables grouped by their schema names. When set to false, tables will be displayed in a flat list under each database.</p> <p>Default: false</p>
SqlCompatibleWindowsAuth	<p>A logical value that determines whether to produce a SQL Server compatible connection string when using Windows Authentication—Trusted_Connection=Yes.</p> <p>If your driver supports Windows Authentication, but requires additional or alternative settings in your connection string, you should set this value to false and use the CredentialConnectionString option record field described below.</p> <p>Default: true</p>

The following table describes the options record fields that are only available through extensibility. Fields that aren't simple literal values are described in subsequent sections.

FIELD	DESCRIPTION
AstVisitor	<p>A record containing one or more overrides to control SQL query generation. The most common usage of this field is to provide logic to generate a LIMIT/OFFSET clause for drivers that don't support TOP.</p> <p>Fields include:</p> <ul style="list-style-type: none"> • Constant • LimitClause <p>See the AstVisitor section for more information.</p>

FIELD	DESCRIPTION
CancelQueryExplicitly	<p>A logical value that instructs the M engine to explicitly cancel any running calls through the ODBC driver before terminating the connection to the ODBC server.</p> <p>This field is useful in situations where query execution is managed independently of the network connections to the server, for example in some Spark deployments. In most cases, this value doesn't need to be set because the query in the server is canceled when the network connection to the server is terminated.</p> <p>Default: false</p>
ClientConnectionPooling	<p>A logical value that enables client-side connection pooling for the ODBC driver. Most drivers will want to set this value to true.</p> <p>Default: false</p>
CredentialConnectionString	<p>A text or record value used to specify credential related connection string properties.</p> <p>See the Credential section for more information.</p>
HideNativeQuery	<p>A logical value that controls whether your connector allows native SQL statements to be passed in by a query using the Value.NativeQuery() function.</p> <p>Note: this functionality is currently not exposed in the Power Query user experience. Users would need to manually edit their queries to take advantage of this capability.</p> <p>Default: false</p>
ImplicitTypeConversions	<p>A table value containing implicit type conversions supported by your driver or backend server. Values in this table are additive to the conversions reported by the driver itself.</p> <p>This field is typically used in conjunction with the SQLGetTypeInfo field when overriding data type information reported by the driver.</p> <p>See the ImplicitTypeConversions section for more information.</p>
OnError	<p>An error handling function that receives an errorRecord parameter of type record.</p> <p>Common uses of this function include handling SSL connection failures, providing a download link if your driver isn't found on the system, and reporting authentication errors.</p> <p>See the OnError section for more information.</p>

FIELD	DESCRIPTION
SoftNumbers	<p>Allows the M engine to select a compatible data type when conversion between two specific numeric types isn't declared as supported in the SQL_CONVERT_* capabilities.</p> <p>Default: false</p>
SqlCapabilities	<p>A record providing various overrides of driver capabilities, and a way to specify capabilities that aren't expressed through ODBC 3.8.</p> <p>See the SqlCapabilities section for more information.</p>
SQLColumns	<p>A function that allows you to modify column metadata returned by the SQLColumns function.</p> <p>See the SQLColumns section for more information.</p>
SQLGetFunctions	<p>A record that allows you to override values returned by calls to SQLGetFunctions.</p> <p>A common use of this field is to disable the use of parameter binding, or to specify that generated queries should use CAST rather than CONVERT.</p> <p>See the SQLGetFunctions section for more information.</p>
SQLGetInfo	<p>A record that allows you to override values returned by calls to SQLGetInfo.</p> <p>See the SQLGetInfo section for more information.</p>
SQLGetTypeInfo	<p>A table, or function that returns a table, that overrides the type information returned by SQLGetTypeInfo.</p> <p>When the value is set to a table, the value completely replaces the type information reported by the driver. SQLGetTypeInfo won't be called.</p> <p>When the value is set to a function, your function will receive the result of the original call to SQLGetTypeInfo, allowing you to modify the table.</p> <p>This field is typically used when there's a mismatch between data types reported by SQLGetTypeInfo and SQLColumns.</p> <p>See the SQLGetTypeInfo section for more information.</p>
SQLTables	<p>A function that allows you to modify the table metadata returned by a call to SQLTables.</p> <p>See the SQLTables section for more information.</p>

FIELD	DESCRIPTION
TolerateConcatOverflow	<p>Allows concatenation of text values to occur even if the result might be truncated to fit within the range of an available type.</p> <p>For example, when concatenating a VARCHAR(4000) field with a VARCHAR(4000) field on a system that supports a maximize VARCHAR size of 4000 and no CLOB type, the concatenation will be folded even though the result might get truncated.</p> <p>Default: false</p>
UseEmbeddedDriver	<p>(internal use): A logical value that controls whether the ODBC driver should be loaded from a local directory (using new functionality defined in the ODBC 4.0 specification). This is generally only set by connectors created by Microsoft that ship with Power Query.</p> <p>When set to false, the system ODBC driver manager will be used to locate and load the driver.</p> <p>Most connectors should not need to set this field.</p> <p>Default: false</p>

Overriding AstVisitor

The AstVisitor field is set through the [Odbc.DataSource](#) options record. It's used to modify SQL statements generated for specific query scenarios.

NOTE

Drivers that support `LIMIT` and `OFFSET` clauses (rather than `TOP`) will want to provide a LimitClause override for AstVisitor.

Constant

Providing an override for this value has been deprecated and may be removed from future implementations.

LimitClause

This field is a function that receives two `Int64.Type` arguments (skip, take), and returns a record with two text fields (Text, Location).

```
LimitClause = (skip as nullable number, take as number) as record => ...
```

The skip parameter is the number of rows to skip (that is, the argument to OFFSET). If an offset is not specified, the skip value will be null. If your driver supports `LIMIT`, but does not support `OFFSET`, the LimitClause function should return an unimplemented error (...) when skip is greater than 0.

The take parameter is the number of rows to take (that is, the argument to LIMIT).

The `Text` field of the result contains the SQL text to add to the generated query.

The `Location` field specifies where to insert the clause. The following table describes supported values.

Value	Description	Example
AfterQuerySpecification	<p>LIMIT clause is put at the end of the generated SQL.</p> <p>This is the most commonly supported LIMIT syntax.</p>	<pre>SELECT a, b, c FROM table WHERE a > 10 LIMIT 5</pre>
BeforeQuerySpecification	LIMIT clause is put before the generated SQL statement.	<pre>LIMIT 5 ROWS SELECT a, b, c FROM table WHERE a > 10</pre>
AfterSelect	LIMIT goes after the SELECT statement, and after any modifiers (such as DISTINCT).	<pre>SELECT DISTINCT LIMIT 5 a, b, c FROM table WHERE a > 10</pre>
AfterSelectBeforeModifiers	LIMIT goes after the SELECT statement, but before any modifiers (such as DISTINCT).	<pre>SELECT LIMIT 5 DISTINCT a, b, c FROM table WHERE a > 10</pre>

The following code snippet provides a LimitClause implementation for a driver that expects a LIMIT clause, with an optional OFFSET, in the following format: [OFFSET <offset> ROWS] LIMIT <row_count>

```
LimitClause = (skip, take) =>
  let
    offset = if (skip > 0) then Text.Format("OFFSET #{0} ROWS", {skip}) else "",
    limit = if (take <> null) then Text.Format("LIMIT #{0}", {take}) else ""
  in
  [
    Text = Text.Format("#{0} #{1}", {offset, limit}),
    Location = "AfterQuerySpecification"
  ]
```

The following code snippet provides a LimitClause implementation for a driver that supports LIMIT, but not OFFSET. Format: LIMIT <row_count> .

```
LimitClause = (skip, take) =>
  if (skip > 0) then error "Skip/Offset not supported"
  else
  [
    Text = Text.Format("LIMIT #{0}", {take}),
    Location = "AfterQuerySpecification"
  ]
```

Overriding SqlCapabilities

FIELD	DETAILS
FractionalSecondsScale	<p>A number value ranging from 1 to 7 that indicates the number of decimal places supported for millisecond values. This value should be set by connectors that want to enable query folding over datetime values.</p> <p>Default: null</p>
PrepareStatements	<p>A logical value that indicates that statements should be prepared using SQLPrepare.</p> <p>Default: false</p>
SupportsTop	<p>A logical value that indicates the driver supports the TOP clause to limit the number of returned rows.</p> <p>Default: false</p>
StringLiteralEscapeCharacters	<p>A list of text values that specify the character(s) to use when escaping string literals and LIKE expressions.</p> <p>Ex. {"\r\n"}</p> <p>Default: null</p>
SupportsDerivedTable	<p>A logical value that indicates the driver supports derived tables (sub-selects).</p> <p>This value is assumed to be true for drivers that set their conformance level to SQL_SC_SQL92_FULL (reported by the driver or overridden with the Sql92Conformance setting (see below)). For all other conformance levels, this value defaults to false.</p> <p>If your driver doesn't report the SQL_SC_SQL92_FULL compliance level, but does support derived tables, set this value to true.</p> <p>Note that supporting derived tables is required for many Direct Query scenarios.</p>
SupportsNumericLiterals	<p>A logical value that indicates whether the generated SQL should include numeric literals values. When set to false, numeric values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsStringLiterals	<p>A logical value that indicates whether the generated SQL should include string literals values. When set to false, string values will always be specified using Parameter Binding.</p> <p>Default: false</p>

FIELD	DETAILS
SupportsOdbcDateLiterals	<p>A logical value that indicates whether the generated SQL should include date literals values. When set to false, date values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimeLiterals	<p>A logical value that indicates whether the generated SQL should include time literals values. When set to false, time values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimestampLiterals	<p>A logical value that indicates whether the generated SQL should include timestamp literals values. When set to false, timestamp values will always be specified using Parameter Binding.</p> <p>Default: false</p>

Overriding SQLColumns

`SQLColumns` is a function handler that receives the results of an ODBC call to [SQLColumns](#). The source parameter contains a table with the data type information. This override is typically used to fix up data type mismatches between calls to `SQLGetTypeInfo` and `SQLColumns`.

For details of the format of the source table parameter, go to [SQLColumns Function](#).

Overriding SQLGetFunctions

This field is used to override SQLFunctions values returned by an ODBC driver. It contains a record whose field names are equal to the FunctionId constants defined for the ODBC [SQLGetFunctions](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#).

FIELD	DETAILS
SQL_CONVERT_FUNCTIONS	<p>Indicates which function(s) are supported when doing type conversions. By default, the M Engine will attempt to use the CONVERT function. Drivers that prefer the use of CAST can override this value to report that only SQL_FN_CVT_CAST (numeric value of 0x2) is supported.</p>
SQL_API_SQLBINDCOL	<p>A logical (true/false) value that indicates whether the Mashup Engine should use the SQLBindCol API when retrieving data. When set to false, SQLGetData is used instead.</p> <p>Default: false</p>

The following code snippet provides an example explicitly telling the M engine to use CAST rather than CONVERT.

```

SQLGetFunctions = [
    SQL_CONVERT_FUNCTIONS = 0x2 /* SQL_FN_CVT_CAST */
]

```

Overriding SQLGetInfo

This field is used to override SQLGetInfo values returned by an ODBC driver. It contains a record whose fields are names equal to the InfoType constants defined for the ODBC [SQLGetInfo](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#). The full list of InfoTypes that are checked can be found in the Mashup Engine trace files.

The following table contains commonly overridden SQLGetInfo properties:

FIELD	DETAILS
SQL_SQL_CONFORMANCE	<p>An integer value that indicates the level of SQL-92 supported by the driver:</p> <ul style="list-style-type: none"> (1) SQL_SC_SQL92_ENTRY = Entry level SQL-92 compliant. (2) SQL_SC_FIPS127_2_TRANSITIONAL = FIPS 127-2 transitional level compliant. (4) SQL_SC_SQL92_INTERMEDIATE = Intermediate level SQL-92 compliant. (8) SQL_SC_SQL92_FULL = Full level SQL-92 compliant. <p>Note that in Power Query scenarios, the connector will be used in a Read Only mode. Most drivers will want to report a SQL_SC_SQL92_FULL compliance level, and override specific SQL generation behavior using the SQLGetInfo and SQLGetFunctions properties.</p>
SQL_SQL92_PREDICATES	<p>A bitmask enumerating the predicates supported in a SELECT statement, as defined in SQL-92.</p> <p>See the SQL_SP_* constants in the ODBC specification.</p>
SQL_AGGREGATE_FUNCTIONS	<p>A bitmask enumerating support for aggregation functions.</p> <p>SQL_AF_ALL SQL_AF_AVG SQL_AF_COUNT SQL_AF_DISTINCT SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM</p> <p>See the SQL_AF_* constants in the ODBC specification.</p>

FIELD	DETAILS
SQL_GROUP_BY	<p>A integer value that specifies the relationship between the columns in the GROUP BY clause and the non-aggregated columns in the select list:</p> <p>SQL_GB_COLLATE = A COLLATE clause can be specified at the end of each grouping column.</p> <p>SQL_GB_NOT_SUPPORTED = GROUP BY clauses are not supported.</p> <p>SQL_GB_GROUP_BY_EQUALS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It cannot contain any other columns. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT.</p> <p>SQL_GB_GROUP_BY_CONTAINS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It can contain columns that are not in the select list. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>SQL_GB_NO_RELATION = The columns in the GROUP BY clause and the select list are not related. The meaning of non-grouped, non-aggregated columns in the select list is data source-dependent. For example, SELECT DEPT, SALARY FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>See the SQL_GB_* constants in the ODBC specification.</p>

The following helper function can be used to create bitmask values from a list of integer values:

```
Flags = (flags as list) =>
    let
        Loop = List.Generate(
            ()=> [i = 0, Combined = 0],
            each [i] < List.Count(flags),
            each [i = [i]+1, Combined =*Number.BitwiseOr([Combined], flags{i})],
            each [Combined]),
        Result = List.Last(Loop, 0)
    in
        Result;
```

Overriding SQLGetTypeInfo

`SQLGetTypeInfo` can be specified in two ways:

- A fixed `table` value that contains the same type information as an ODBC call to `sqlGetTypeInfo`.
- A function that accepts a table argument, and returns a table. The argument will contain the original results of the ODBC call to `sqlGetTypeInfo`. Your function implementation can modify/add to this table.

The first approach is used to completely override the values returned by the ODBC driver. The second approach is used if you want to add to or modify these values.

For details of the format of the types table parameter and expected return value, see the [SQLGetTypeInfo function reference](#).

SQLGetTypeInfo using a static table

The following code snippet provides a static implementation for SQLGetTypeInfo.

```

SQLGetTypeInfo = #table(
    { "TYPE_NAME",      "DATA_TYPE", "COLUMN_SIZE", "LITERAL_PREF", "LITERAL_SUFFIX", "CREATE_PARAS",
    "NULLABLE", "CASE_SENSITIVE", "SEARCHABLE", "UNSIGNED_ATTRIBUTE", "FIXED_PREC_SCALE", "AUTO_UNIQUE_VALUE",
    "LOCAL_TYPE_NAME", "MINIMUM_SCALE", "MAXIMUM_SCALE", "SQL_DATA_TYPE", "SQL_DATETIME_SUB", "NUM_PREC_RADIX",
    "INTERNAL_PRECISION", "USER_DATA_TYPE" }, {
        { "char",           1,       65535,      "",      "",      "max. length",
1,           1,           3,       null,      0,      null,
"char",       null,       null,      -8,      null,
0,           0,           0,       null,      10,      null,
        { "int8",          -5,       19,      "",      "",      null,
1,           0,           2,       0,      -5,      0,
"int8",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "bit",           -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bit",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "bool",          -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bool",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "date",           9,       10,      "",      "",      null,
1,           0,           2,       null,      0,      null,
"date",       null,       null,      9,      1,
0,           0,           0,       null,      null,
        { "numeric",         3,       28,      null,      null,      null,
1,           0,           2,       0,      0,
"numeric",       0,       0,       null,      null,
0,           0,           0,       null,      10,
        { "float8",          8,       15,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "float8",          6,       17,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "uuid",          -11,       37,      null,      null,      null,
1,           0,           2,       null,      0,
"uuid",       null,       null,      -11,      null,
0,           0,           0,       null,      null,
        { "int4",            4,       10,      null,      null,      null,
1,           0,           2,       0,      0,
"int4",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "text",            -1,       65535,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"text",       null,       null,      -10,      null,
0,           0,           0,       null,      null,
        { "lo",              -4,       255,      "",      "",      null,
1,           0,           2,       null,      0,
"lo",       null,       null,      -4,      null,
0,           0,           0,       null,      null,
        { "numeric",          2,       28,      null,      null,      "precision, scale",
1,           0,           2,       0,      10,
"numeric",       0,       6,       2,      null,
0,           0,           0,       null,      10,
        { "float4",            7,       9,       null,      null,      null,
1,           0,           2,       0,      10,
"float4",       null,       null,      7,      null,
0,           0,           0,       null,      2,
        { "int2",            5,       19,      null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "int2",            -6,       5,       null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
    }
}

```

```

0,          0      },
0, { "timestamp", 11, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "date", 91, 10, null, "", "", null,
1, 0, 2, null, 9, 1, null,
"date", null, }, 1,
0, 0 }, null,
{ "timestamp", 93, 26, null, "", "", null,
1, 0, 2, 38, 9, 3, null,
"timestamp", 0, }, null,
0, { "bytea", -3, 255, null, "", "", null,
1, 0, 2, null, 0, null,
"bytea", null, }, null,
0, 0 }, null,
{ "varchar", 12, 65535, null, "", "", "max. length",
1, 0, 2, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "char", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"char", null, }, null,
0, 0 }, null,
{ "text", -10, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"text", null, }, null,
0, 0 }, null,
{ "varchar", -9, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "bpchar", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"bpchar", null, }, null,
0, 0 } }
);

```

SQLGetTypeInfo using a function

The following code snippets append the `bpchar` type to the existing types returned by the driver.

```

SQLGetTypeInfo = (types as table) as table =>
let
    newTypes = #table(
    {
        "TYPE_NAME",
        "DATA_TYPE",
        "COLUMN_SIZE",
        "LITERAL_PREF",
        "LITERAL_SUFFIX",
        "CREATE_PARAS",
        "NULLABLE",
        "CASE_SENSITIVE",
        "SEARCHABLE",
        "UNSIGNED_ATTRIBUTE",
        "FIXED_PREC_SCALE",
        "AUTO_UNIQUE_VALUE",
        "LOCAL_TYPE_NAME",
        "MINIMUM_SCALE",
        "MAXIMUM_SCALE",
        "SQL_DATA_TYPE",
        "SQL_DATETIME_SUB",
        "NUM_PREC_RADIX",
        "INTERNAL_PRECISION",
        "USER_DATA_TYPE"
    },
    // we add a new entry for each type we want to add
    {
        {
            "bpchar",
            -8,
            65535,
            """",
            """",
            "max. length",
            1,
            1,
            3,
            null,
            0,
            null,
            "bpchar",
            null,
            null,
            -9,
            null,
            null,
            0,
            0
        }
    )),
    append = Table.Combine({types, newTypes})
in
    append;

```

Setting the Connection String

The connection string for your ODBC driver is set using the first argument to the [Odbc.DataSource](#) and/or [Odbc.Query](#) functions. The value can be text, or an M record. When using the record, each field in the record will become a property in the connection string. All connection strings will require a Driver field (or DSN field if you require users to pre-configure a system level DSN). Credential related properties will be set separately (see below). Other properties will be driver specific.

The code snippet below shows the definition of a new data source function, creation of the `ConnectionString` record, and invocation of the [Odbc.DataSource](#) function.

```
[DataSource.Kind="SqlODBC", Publish="SqlODBC.Publish"]
shared SqlODBC.Contents = (server as text) =>
    let
        ConnectionString = [
            Driver = "SQL Server Native Client 11.0",
            Server = server,
            MultiSubnetFailover = "Yes",
            ApplicationIntent = "ReadOnly",
            APP = "PowerBICustomConnector"
        ],
        OdbcDatasource = Odbc.DataSource(ConnectionString)
    in
        OdbcDatasource;
```

Troubleshooting and Testing

To enable tracing in Power BI Desktop:

1. Go to **File > Options and settings > Options**.
2. Select on the **Diagnostics** tab.
3. Select the **Enable tracing** option.
4. Select the **Open traces folder** link (should be `%LOCALAPPDATA%/Microsoft/Power BI Desktop/Traces`).
5. Delete existing trace files.
6. Perform your tests.
7. Close Power BI Desktop to ensure all log files are flushed to disk.

Here are steps you can take for initial testing in Power BI Desktop:

1. Close Power BI Desktop.
2. Clear your trace directory.
3. Open Power BI desktop, and enable tracing.
4. Connect to your data source, and select Direct Query mode.
5. Select a table in the navigator, and select **Edit**.
6. Manipulate the query in various ways, including:

- Take the First N rows (for example, 10).
- Set equality filters on different data types (int, string, bool, and so on).
- Set other range filters (greater than, less than).
- Filter on NULL / NOT NULL.
- Select a sub-set of columns.
- Aggregate / Group By different column combinations.
- Add a column calculated from other columns ($[C] = [A] + [B]$).
- Sort on one column, multiple columns. 7. Expressions that fail to fold will result in a warning bar. Note the failure, remove the step, and move to the next test case. Details about the cause of the failure should be emitted to the trace logs. 8. Close Power BI Desktop. 9. Copy the trace files to a new directory. 10. Use the recommend Power BI workbook to parse and analyze the trace files.

Once you have simple queries working, you can then try Direct Query scenarios (for example, building reports in the Report Views). The queries generated in Direct Query mode will be significantly more complex (that is, use of sub-selects, COALESCE statements, and aggregations).

Concatenation of strings in Direct Query mode

The M engine does basic type size limit validation as part of its query folding logic. If you are receiving a folding

error when trying to concatenate two strings that potentially overflow the maximum size of the underlying database type:

1. Ensure that your database can support up-conversion to CLOB types when string concat overflow occurs.
2. Set the `TolerateConcatOverflow` option for `Odbc.DataSource` to `true`.

The [DAX CONCATENATE function](#) is currently not supported by Power Query/ODBC extensions. Extension authors should ensure string concatenation works through the query editor by adding calculated columns (`[stringCol1] & [stringCol2]`). When the capability to fold the CONCATENATE operation is added in the future, it should work seamlessly with existing extensions.

Enabling Direct Query for an ODBC based connector

1/15/2022 • 21 minutes to read • [Edit Online](#)

Overview

Using M's built-in [Odbc.DataSource](#) function is the recommended way to create custom connectors for data sources that have an existing ODBC driver and/or support a SQL query syntax. Wrapping the [Odbc.DataSource](#) function will allow your connector to inherit default query folding behavior based on the capabilities reported by your driver. This will enable the M engine to generate SQL statements based on filters and other transformations defined by the user within the Power Query experience, without having to provide this logic within the connector itself.

ODBC extensions can optionally enable Direct Query mode, allowing Power BI to dynamically generate queries at runtime without pre-caching the user's data model.

NOTE

Enabling Direct Query support raises the difficulty and complexity level of your connector. When Direct Query is enabled, Power BI will prevent the M engine from compensating for operations that cannot be fully pushed to the underlying data source.

This section builds on the concepts presented in the M Extensibility Reference, and assumes familiarity with the creation of a basic Data Connector.

Refer to the [SqlODBC sample](#) for most of the code examples in the sections below. Additional samples can be found in the ODBC samples directory.

ODBC Extensibility Functions

The M engine provides two ODBC related data source functions: [Odbc.DataSource](#), and [Odbc.Query](#).

The [Odbc.DataSource](#) function provides a default navigation table with all databases, tables, and views from your system, supports query folding, and allows for a range of customization options. The majority of ODBC based extensions will use this as their primary extensibility function. The function accepts two arguments—a connection string, and an options record to provide behavior overrides.

The [Odbc.Query](#) function allows you to execute SQL statements through an ODBC driver. It acts as a passthrough for query execution. Unlike the [Odbc.DataSource](#) function, it doesn't provide query folding functionality, and requires that SQL queries be provided by the connector (or end user). When building a custom connector, this function is typically used internally to run queries to retrieve metadata that might not be exposed through regular ODBC channels. The function accepts two arguments—a connection string, and a SQL query.

Parameters for your Data Source Function

Custom connectors can accept any number of function arguments, but to remain consistent with the built-in data source functions shipped with Power Query, the following guidelines are recommended:

- Require the minimal set of parameters used to establish a connection to your server. The less parameters end users need to provide, the easier your connector will be to use.

- Although you can define parameters with a fixed number of values (that is, a dropdown list in the UI), parameters are entered before the user is authenticated. Any values that can be discovered programmatically after the user is authenticated (such as catalog or database name) should be selectable through the Navigator. The default behavior for the [Odbc.DataSource](#) function will be to return a hierarchical navigation table consisting of Catalog (Database), Schema, and Table names, although this can be overridden within your connector.
- If you feel your users will typically know what values to enter for items they would select from the Navigator (such as the database name), make these parameters optional. Parameters that can be discovered programmatically should not be made required.
- The last parameter for your function should be an optional record called "options". This parameter typically allows advanced users to set common ODBC related properties (such as CommandTimeout), set behavior overrides specific to your connector, and allows for future extensibility without impacting backwards compatibility for your function.
- Security/credential related arguments MUST never be part of your data source function parameters, as values entered in the connect dialog will be persisted to the user's query. Credential related parameters should be specified as part of the connector's supported Authentication methods.

By default, all required parameters for your data source function are factored into the Data Source Path value used to identify user credentials.

Note that while the UI for the built-in [Odbc.DataSource](#) function provides a dropdown that allows the user to select a DSN, this functionality is not available through extensibility. If your data source configuration is complex enough to require a fully customizable configuration dialog, it's recommended you require your end users to pre-configure a system DSN, and have your function take in the DSN name as a text field.

Parameters for Odbc.DataSource

The [Odbc.DataSource](#) function takes two parameters—a connectionString for your driver, and an options record that lets you override various driver behaviors. Through the options record you can override capabilities and other information reported by the driver, control the navigator behavior, and affect the SQL queries generated by the M engine.

The supported options records fields fall into two categories—those that are public / always available, and those that are only available in an extensibility context.

The following table describes the public fields in the options record.

FIELD	DESCRIPTION
CommandTimeout	A duration value that controls how long the server-side query is allowed to run before it's cancelled. Default: 10 minutes
ConnectionTimeout	A duration value that controls how long to wait before abandoning an attempt to make a connection to the server. Default: 15 seconds

FIELD	DESCRIPTION
CreateNavigationProperties	<p>A logical value that sets whether to generate navigation properties on the returned tables. Navigation properties are based on foreign key relationships reported by the driver, and show up as "virtual" columns that can be expanded in the query editor, creating the appropriate join.</p> <p>If calculating foreign key dependencies is an expensive operation for your driver, you may want to set this value to false.</p> <p>Default: true</p>
HierarchicalNavigation	<p>A logical value that sets whether to view the tables grouped by their schema names. When set to false, tables will be displayed in a flat list under each database.</p> <p>Default: false</p>
SqlCompatibleWindowsAuth	<p>A logical value that determines whether to produce a SQL Server compatible connection string when using Windows Authentication—Trusted_Connection=Yes.</p> <p>If your driver supports Windows Authentication, but requires additional or alternative settings in your connection string, you should set this value to false and use the CredentialConnectionString option record field described below.</p> <p>Default: true</p>

The following table describes the options record fields that are only available through extensibility. Fields that aren't simple literal values are described in subsequent sections.

FIELD	DESCRIPTION
AstVisitor	<p>A record containing one or more overrides to control SQL query generation. The most common usage of this field is to provide logic to generate a LIMIT/OFFSET clause for drivers that don't support TOP.</p> <p>Fields include:</p> <ul style="list-style-type: none"> • Constant • LimitClause <p>See the AstVisitor section for more information.</p>

FIELD	DESCRIPTION
CancelQueryExplicitly	<p>A logical value that instructs the M engine to explicitly cancel any running calls through the ODBC driver before terminating the connection to the ODBC server.</p> <p>This field is useful in situations where query execution is managed independently of the network connections to the server, for example in some Spark deployments. In most cases, this value doesn't need to be set because the query in the server is canceled when the network connection to the server is terminated.</p> <p>Default: false</p>
ClientConnectionPooling	<p>A logical value that enables client-side connection pooling for the ODBC driver. Most drivers will want to set this value to true.</p> <p>Default: false</p>
CredentialConnectionString	<p>A text or record value used to specify credential related connection string properties.</p> <p>See the Credential section for more information.</p>
HideNativeQuery	<p>A logical value that controls whether your connector allows native SQL statements to be passed in by a query using the Value.NativeQuery() function.</p> <p>Note: this functionality is currently not exposed in the Power Query user experience. Users would need to manually edit their queries to take advantage of this capability.</p> <p>Default: false</p>
ImplicitTypeConversions	<p>A table value containing implicit type conversions supported by your driver or backend server. Values in this table are additive to the conversions reported by the driver itself.</p> <p>This field is typically used in conjunction with the SQLGetTypeInfo field when overriding data type information reported by the driver.</p> <p>See the ImplicitTypeConversions section for more information.</p>
OnError	<p>An error handling function that receives an errorRecord parameter of type record.</p> <p>Common uses of this function include handling SSL connection failures, providing a download link if your driver isn't found on the system, and reporting authentication errors.</p> <p>See the OnError section for more information.</p>

FIELD	DESCRIPTION
SoftNumbers	<p>Allows the M engine to select a compatible data type when conversion between two specific numeric types isn't declared as supported in the SQL_CONVERT_* capabilities.</p> <p>Default: false</p>
SqlCapabilities	<p>A record providing various overrides of driver capabilities, and a way to specify capabilities that aren't expressed through ODBC 3.8.</p> <p>See the SqlCapabilities section for more information.</p>
SQLColumns	<p>A function that allows you to modify column metadata returned by the SQLColumns function.</p> <p>See the SQLColumns section for more information.</p>
SQLGetFunctions	<p>A record that allows you to override values returned by calls to SQLGetFunctions.</p> <p>A common use of this field is to disable the use of parameter binding, or to specify that generated queries should use CAST rather than CONVERT.</p> <p>See the SQLGetFunctions section for more information.</p>
SQLGetInfo	<p>A record that allows you to override values returned by calls to SQLGetInfo.</p> <p>See the SQLGetInfo section for more information.</p>
SQLGetTypeInfo	<p>A table, or function that returns a table, that overrides the type information returned by SQLGetTypeInfo.</p> <p>When the value is set to a table, the value completely replaces the type information reported by the driver. SQLGetTypeInfo won't be called.</p> <p>When the value is set to a function, your function will receive the result of the original call to SQLGetTypeInfo, allowing you to modify the table.</p> <p>This field is typically used when there's a mismatch between data types reported by SQLGetTypeInfo and SQLColumns.</p> <p>See the SQLGetTypeInfo section for more information.</p>
SQLTables	<p>A function that allows you to modify the table metadata returned by a call to SQLTables.</p> <p>See the SQLTables section for more information.</p>

FIELD	DESCRIPTION
TolerateConcatOverflow	<p>Allows concatenation of text values to occur even if the result might be truncated to fit within the range of an available type.</p> <p>For example, when concatenating a VARCHAR(4000) field with a VARCHAR(4000) field on a system that supports a maximize VARCHAR size of 4000 and no CLOB type, the concatenation will be folded even though the result might get truncated.</p> <p>Default: false</p>
UseEmbeddedDriver	<p>(internal use): A logical value that controls whether the ODBC driver should be loaded from a local directory (using new functionality defined in the ODBC 4.0 specification). This is generally only set by connectors created by Microsoft that ship with Power Query.</p> <p>When set to false, the system ODBC driver manager will be used to locate and load the driver.</p> <p>Most connectors should not need to set this field.</p> <p>Default: false</p>

Overriding AstVisitor

The AstVisitor field is set through the [Odbc.DataSource](#) options record. It's used to modify SQL statements generated for specific query scenarios.

NOTE

Drivers that support `LIMIT` and `OFFSET` clauses (rather than `TOP`) will want to provide a LimitClause override for AstVisitor.

Constant

Providing an override for this value has been deprecated and may be removed from future implementations.

LimitClause

This field is a function that receives two `Int64.Type` arguments (skip, take), and returns a record with two text fields (Text, Location).

```
LimitClause = (skip as nullable number, take as number) as record => ...
```

The skip parameter is the number of rows to skip (that is, the argument to OFFSET). If an offset is not specified, the skip value will be null. If your driver supports `LIMIT`, but does not support `OFFSET`, the LimitClause function should return an unimplemented error (...) when skip is greater than 0.

The take parameter is the number of rows to take (that is, the argument to LIMIT).

The `Text` field of the result contains the SQL text to add to the generated query.

The `Location` field specifies where to insert the clause. The following table describes supported values.

Value	Description	Example
AfterQuerySpecification	<p>LIMIT clause is put at the end of the generated SQL.</p> <p>This is the most commonly supported LIMIT syntax.</p>	<pre>SELECT a, b, c FROM table WHERE a > 10 LIMIT 5</pre>
BeforeQuerySpecification	LIMIT clause is put before the generated SQL statement.	<pre>LIMIT 5 ROWS SELECT a, b, c FROM table WHERE a > 10</pre>
AfterSelect	LIMIT goes after the SELECT statement, and after any modifiers (such as DISTINCT).	<pre>SELECT DISTINCT LIMIT 5 a, b, c FROM table WHERE a > 10</pre>
AfterSelectBeforeModifiers	LIMIT goes after the SELECT statement, but before any modifiers (such as DISTINCT).	<pre>SELECT LIMIT 5 DISTINCT a, b, c FROM table WHERE a > 10</pre>

The following code snippet provides a LimitClause implementation for a driver that expects a LIMIT clause, with an optional OFFSET, in the following format: [OFFSET <offset> ROWS] LIMIT <row_count>

```
LimitClause = (skip, take) =>
  let
    offset = if (skip > 0) then Text.Format("OFFSET #{0} ROWS", {skip}) else "",
    limit = if (take <> null) then Text.Format("LIMIT #{0}", {take}) else ""
  in
  [
    Text = Text.Format("#{0} #{1}", {offset, limit}),
    Location = "AfterQuerySpecification"
  ]
```

The following code snippet provides a LimitClause implementation for a driver that supports LIMIT, but not OFFSET. Format: LIMIT <row_count> .

```
LimitClause = (skip, take) =>
  if (skip > 0) then error "Skip/Offset not supported"
  else
  [
    Text = Text.Format("LIMIT #{0}", {take}),
    Location = "AfterQuerySpecification"
  ]
```

Overriding SqlCapabilities

FIELD	DETAILS
FractionalSecondsScale	<p>A number value ranging from 1 to 7 that indicates the number of decimal places supported for millisecond values. This value should be set by connectors that want to enable query folding over datetime values.</p> <p>Default: null</p>
PrepareStatements	<p>A logical value that indicates that statements should be prepared using SQLPrepare.</p> <p>Default: false</p>
SupportsTop	<p>A logical value that indicates the driver supports the TOP clause to limit the number of returned rows.</p> <p>Default: false</p>
StringLiteralEscapeCharacters	<p>A list of text values that specify the character(s) to use when escaping string literals and LIKE expressions.</p> <p>Ex. {"\r\n"}</p> <p>Default: null</p>
SupportsDerivedTable	<p>A logical value that indicates the driver supports derived tables (sub-selects).</p> <p>This value is assumed to be true for drivers that set their conformance level to SQL_SC_SQL92_FULL (reported by the driver or overridden with the Sql92Conformance setting (see below)). For all other conformance levels, this value defaults to false.</p> <p>If your driver doesn't report the SQL_SC_SQL92_FULL compliance level, but does support derived tables, set this value to true.</p> <p>Note that supporting derived tables is required for many Direct Query scenarios.</p>
SupportsNumericLiterals	<p>A logical value that indicates whether the generated SQL should include numeric literals values. When set to false, numeric values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsStringLiterals	<p>A logical value that indicates whether the generated SQL should include string literals values. When set to false, string values will always be specified using Parameter Binding.</p> <p>Default: false</p>

FIELD	DETAILS
SupportsOdbcDateLiterals	<p>A logical value that indicates whether the generated SQL should include date literals values. When set to false, date values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimeLiterals	<p>A logical value that indicates whether the generated SQL should include time literals values. When set to false, time values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimestampLiterals	<p>A logical value that indicates whether the generated SQL should include timestamp literals values. When set to false, timestamp values will always be specified using Parameter Binding.</p> <p>Default: false</p>

Overriding SQLColumns

`SQLColumns` is a function handler that receives the results of an ODBC call to [SQLColumns](#). The source parameter contains a table with the data type information. This override is typically used to fix up data type mismatches between calls to `SQLGetTypeInfo` and `SQLColumns`.

For details of the format of the source table parameter, go to [SQLColumns Function](#).

Overriding SQLGetFunctions

This field is used to override SQLFunctions values returned by an ODBC driver. It contains a record whose field names are equal to the FunctionId constants defined for the ODBC [SQLGetFunctions](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#).

FIELD	DETAILS
SQL_CONVERT_FUNCTIONS	<p>Indicates which function(s) are supported when doing type conversions. By default, the M Engine will attempt to use the CONVERT function. Drivers that prefer the use of CAST can override this value to report that only SQL_FN_CVT_CAST (numeric value of 0x2) is supported.</p>
SQL_API_SQLBINDCOL	<p>A logical (true/false) value that indicates whether the Mashup Engine should use the SQLBindCol API when retrieving data. When set to false, SQLGetData is used instead.</p> <p>Default: false</p>

The following code snippet provides an example explicitly telling the M engine to use CAST rather than CONVERT.

```

SQLGetFunctions = [
    SQL_CONVERT_FUNCTIONS = 0x2 /* SQL_FN_CVT_CAST */
]

```

Overriding SQLGetInfo

This field is used to override SQLGetInfo values returned by an ODBC driver. It contains a record whose fields are names equal to the InfoType constants defined for the ODBC [SQLGetInfo](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#). The full list of InfoTypes that are checked can be found in the Mashup Engine trace files.

The following table contains commonly overridden SQLGetInfo properties:

FIELD	DETAILS
SQL_SQL_CONFORMANCE	<p>An integer value that indicates the level of SQL-92 supported by the driver:</p> <ul style="list-style-type: none"> (1) SQL_SC_SQL92_ENTRY = Entry level SQL-92 compliant. (2) SQL_SC_FIPS127_2_TRANSITIONAL = FIPS 127-2 transitional level compliant. (4) SQL_SC_SQL92_INTERMEDIATE = Intermediate level SQL-92 compliant. (8) SQL_SC_SQL92_FULL = Full level SQL-92 compliant. <p>Note that in Power Query scenarios, the connector will be used in a Read Only mode. Most drivers will want to report a SQL_SC_SQL92_FULL compliance level, and override specific SQL generation behavior using the SQLGetInfo and SQLGetFunctions properties.</p>
SQL_SQL92_PREDICATES	<p>A bitmask enumerating the predicates supported in a SELECT statement, as defined in SQL-92.</p> <p>See the SQL_SP_* constants in the ODBC specification.</p>
SQL_AGGREGATE_FUNCTIONS	<p>A bitmask enumerating support for aggregation functions.</p> <p>SQL_AF_ALL SQL_AF_AVG SQL_AF_COUNT SQL_AF_DISTINCT SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM</p> <p>See the SQL_AF_* constants in the ODBC specification.</p>

FIELD	DETAILS
SQL_GROUP_BY	<p>A integer value that specifies the relationship between the columns in the GROUP BY clause and the non-aggregated columns in the select list:</p> <p>SQL_GB_COLLATE = A COLLATE clause can be specified at the end of each grouping column.</p> <p>SQL_GB_NOT_SUPPORTED = GROUP BY clauses are not supported.</p> <p>SQL_GB_GROUP_BY_EQUALS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It cannot contain any other columns. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT.</p> <p>SQL_GB_GROUP_BY_CONTAINS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It can contain columns that are not in the select list. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>SQL_GB_NO_RELATION = The columns in the GROUP BY clause and the select list are not related. The meaning of non-grouped, non-aggregated columns in the select list is data source-dependent. For example, SELECT DEPT, SALARY FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>See the SQL_GB_* constants in the ODBC specification.</p>

The following helper function can be used to create bitmask values from a list of integer values:

```
Flags = (flags as list) =>
    let
        Loop = List.Generate(
            ()=> [i = 0, Combined = 0],
            each [i] < List.Count(flags),
            each [i = [i]+1, Combined =*Number.BitwiseOr([Combined], flags{i})],
            each [Combined]),
        Result = List.Last(Loop, 0)
    in
        Result;
```

Overriding SQLGetTypeInfo

`SQLGetTypeInfo` can be specified in two ways:

- A fixed `table` value that contains the same type information as an ODBC call to `sqlGetTypeInfo`.
- A function that accepts a table argument, and returns a table. The argument will contain the original results of the ODBC call to `sqlGetTypeInfo`. Your function implementation can modify/add to this table.

The first approach is used to completely override the values returned by the ODBC driver. The second approach is used if you want to add to or modify these values.

For details of the format of the types table parameter and expected return value, see the [SQLGetTypeInfo function reference](#).

SQLGetTypeInfo using a static table

The following code snippet provides a static implementation for SQLGetTypeInfo.


```

0,          0      },
0, { "timestamp", 11, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "date", 91, 10, null, "", "", null,
1, 0, 2, null, 9, 1, null,
"date", null, }, 1,
0, 0 }, null,
{ "timestamp", 93, 26, null, "", "", null,
1, 0, 2, 38, 9, 3, null,
"timestamp", 0, }, null,
0, { "bytea", -3, 255, null, "", "", null,
1, 0, 2, null, 0, null,
"bytea", null, }, null,
0, 0 }, null,
{ "varchar", 12, 65535, null, "", "", "max. length",
1, 0, 2, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "char", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"char", null, }, null,
0, 0 }, null,
{ "text", -10, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"text", null, }, null,
0, 0 }, null,
{ "varchar", -9, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "bpchar", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"bpchar", null, }, null,
0, 0 } }
);

```

SQLGetTypeInfo using a function

The following code snippets append the `bpchar` type to the existing types returned by the driver.

```

SQLGetTypeInfo = (types as table) as table =>
let
    newTypes = #table(
    {
        "TYPE_NAME",
        "DATA_TYPE",
        "COLUMN_SIZE",
        "LITERAL_PREF",
        "LITERAL_SUFFIX",
        "CREATE_PARAS",
        "NULLABLE",
        "CASE_SENSITIVE",
        "SEARCHABLE",
        "UNSIGNED_ATTRIBUTE",
        "FIXED_PREC_SCALE",
        "AUTO_UNIQUE_VALUE",
        "LOCAL_TYPE_NAME",
        "MINIMUM_SCALE",
        "MAXIMUM_SCALE",
        "SQL_DATA_TYPE",
        "SQL_DATETIME_SUB",
        "NUM_PREC_RADIX",
        "INTERNAL_PRECISION",
        "USER_DATA_TYPE"
    },
    // we add a new entry for each type we want to add
    {
        {
            "bpchar",
            -8,
            65535,
            """",
            """",
            "max. length",
            1,
            1,
            3,
            null,
            0,
            null,
            "bpchar",
            null,
            null,
            -9,
            null,
            null,
            0,
            0
        }
    )),
    append = Table.Combine({types, newTypes})
in
    append;

```

Setting the Connection String

The connection string for your ODBC driver is set using the first argument to the [Odbc.DataSource](#) and/or [Odbc.Query](#) functions. The value can be text, or an M record. When using the record, each field in the record will become a property in the connection string. All connection strings will require a Driver field (or DSN field if you require users to pre-configure a system level DSN). Credential related properties will be set separately (see below). Other properties will be driver specific.

The code snippet below shows the definition of a new data source function, creation of the `ConnectionString` record, and invocation of the [Odbc.DataSource](#) function.

```
[DataSource.Kind="SqlODBC", Publish="SqlODBC.Publish"]
shared SqlODBC.Contents = (server as text) =>
    let
        ConnectionString = [
            Driver = "SQL Server Native Client 11.0",
            Server = server,
            MultiSubnetFailover = "Yes",
            ApplicationIntent = "ReadOnly",
            APP = "PowerBICustomConnector"
        ],
        OdbcDatasource = Odbc.DataSource(ConnectionString)
    in
        OdbcDatasource;
```

Troubleshooting and Testing

To enable tracing in Power BI Desktop:

1. Go to **File > Options and settings > Options**.
2. Select on the **Diagnostics** tab.
3. Select the **Enable tracing** option.
4. Select the **Open traces folder** link (should be `%LOCALAPPDATA%/Microsoft/Power BI Desktop/Traces`).
5. Delete existing trace files.
6. Perform your tests.
7. Close Power BI Desktop to ensure all log files are flushed to disk.

Here are steps you can take for initial testing in Power BI Desktop:

1. Close Power BI Desktop.
2. Clear your trace directory.
3. Open Power BI desktop, and enable tracing.
4. Connect to your data source, and select Direct Query mode.
5. Select a table in the navigator, and select **Edit**.
6. Manipulate the query in various ways, including:

- Take the First N rows (for example, 10).
- Set equality filters on different data types (int, string, bool, and so on).
- Set other range filters (greater than, less than).
- Filter on NULL / NOT NULL.
- Select a sub-set of columns.
- Aggregate / Group By different column combinations.
- Add a column calculated from other columns ($[C] = [A] + [B]$).
- Sort on one column, multiple columns. 7. Expressions that fail to fold will result in a warning bar. Note the failure, remove the step, and move to the next test case. Details about the cause of the failure should be emitted to the trace logs. 8. Close Power BI Desktop. 9. Copy the trace files to a new directory. 10. Use the recommend Power BI workbook to parse and analyze the trace files.

Once you have simple queries working, you can then try Direct Query scenarios (for example, building reports in the Report Views). The queries generated in Direct Query mode will be significantly more complex (that is, use of sub-selects, COALESCE statements, and aggregations).

Concatenation of strings in Direct Query mode

The M engine does basic type size limit validation as part of its query folding logic. If you are receiving a folding

error when trying to concatenate two strings that potentially overflow the maximum size of the underlying database type:

1. Ensure that your database can support up-conversion to CLOB types when string concat overflow occurs.
2. Set the `TolerateConcatOverflow` option for `Odbc.DataSource` to `true`.

The [DAX CONCATENATE function](#) is currently not supported by Power Query/ODBC extensions. Extension authors should ensure string concatenation works through the query editor by adding calculated columns (`[stringCol1] & [stringCol2]`). When the capability to fold the CONCATENATE operation is added in the future, it should work seamlessly with existing extensions.

Enabling Direct Query for an ODBC based connector

1/15/2022 • 21 minutes to read • [Edit Online](#)

Overview

Using M's built-in [Odbc.DataSource](#) function is the recommended way to create custom connectors for data sources that have an existing ODBC driver and/or support a SQL query syntax. Wrapping the [Odbc.DataSource](#) function will allow your connector to inherit default query folding behavior based on the capabilities reported by your driver. This will enable the M engine to generate SQL statements based on filters and other transformations defined by the user within the Power Query experience, without having to provide this logic within the connector itself.

ODBC extensions can optionally enable Direct Query mode, allowing Power BI to dynamically generate queries at runtime without pre-caching the user's data model.

NOTE

Enabling Direct Query support raises the difficulty and complexity level of your connector. When Direct Query is enabled, Power BI will prevent the M engine from compensating for operations that cannot be fully pushed to the underlying data source.

This section builds on the concepts presented in the M Extensibility Reference, and assumes familiarity with the creation of a basic Data Connector.

Refer to the [SqlODBC sample](#) for most of the code examples in the sections below. Additional samples can be found in the ODBC samples directory.

ODBC Extensibility Functions

The M engine provides two ODBC related data source functions: [Odbc.DataSource](#), and [Odbc.Query](#).

The [Odbc.DataSource](#) function provides a default navigation table with all databases, tables, and views from your system, supports query folding, and allows for a range of customization options. The majority of ODBC based extensions will use this as their primary extensibility function. The function accepts two arguments—a connection string, and an options record to provide behavior overrides.

The [Odbc.Query](#) function allows you to execute SQL statements through an ODBC driver. It acts as a passthrough for query execution. Unlike the [Odbc.DataSource](#) function, it doesn't provide query folding functionality, and requires that SQL queries be provided by the connector (or end user). When building a custom connector, this function is typically used internally to run queries to retrieve metadata that might not be exposed through regular ODBC channels. The function accepts two arguments—a connection string, and a SQL query.

Parameters for your Data Source Function

Custom connectors can accept any number of function arguments, but to remain consistent with the built-in data source functions shipped with Power Query, the following guidelines are recommended:

- Require the minimal set of parameters used to establish a connection to your server. The less parameters end users need to provide, the easier your connector will be to use.

- Although you can define parameters with a fixed number of values (that is, a dropdown list in the UI), parameters are entered before the user is authenticated. Any values that can be discovered programmatically after the user is authenticated (such as catalog or database name) should be selectable through the Navigator. The default behavior for the [Odbc.DataSource](#) function will be to return a hierarchical navigation table consisting of Catalog (Database), Schema, and Table names, although this can be overridden within your connector.
- If you feel your users will typically know what values to enter for items they would select from the Navigator (such as the database name), make these parameters optional. Parameters that can be discovered programmatically should not be made required.
- The last parameter for your function should be an optional record called "options". This parameter typically allows advanced users to set common ODBC related properties (such as CommandTimeout), set behavior overrides specific to your connector, and allows for future extensibility without impacting backwards compatibility for your function.
- Security/credential related arguments MUST never be part of your data source function parameters, as values entered in the connect dialog will be persisted to the user's query. Credential related parameters should be specified as part of the connector's supported Authentication methods.

By default, all required parameters for your data source function are factored into the Data Source Path value used to identify user credentials.

Note that while the UI for the built-in [Odbc.DataSource](#) function provides a dropdown that allows the user to select a DSN, this functionality is not available through extensibility. If your data source configuration is complex enough to require a fully customizable configuration dialog, it's recommended you require your end users to pre-configure a system DSN, and have your function take in the DSN name as a text field.

Parameters for Odbc.DataSource

The [Odbc.DataSource](#) function takes two parameters—a connectionString for your driver, and an options record that lets you override various driver behaviors. Through the options record you can override capabilities and other information reported by the driver, control the navigator behavior, and affect the SQL queries generated by the M engine.

The supported options records fields fall into two categories—those that are public / always available, and those that are only available in an extensibility context.

The following table describes the public fields in the options record.

FIELD	DESCRIPTION
CommandTimeout	A duration value that controls how long the server-side query is allowed to run before it's cancelled. Default: 10 minutes
ConnectionTimeout	A duration value that controls how long to wait before abandoning an attempt to make a connection to the server. Default: 15 seconds

FIELD	DESCRIPTION
CreateNavigationProperties	<p>A logical value that sets whether to generate navigation properties on the returned tables. Navigation properties are based on foreign key relationships reported by the driver, and show up as "virtual" columns that can be expanded in the query editor, creating the appropriate join.</p> <p>If calculating foreign key dependencies is an expensive operation for your driver, you may want to set this value to false.</p> <p>Default: true</p>
HierarchicalNavigation	<p>A logical value that sets whether to view the tables grouped by their schema names. When set to false, tables will be displayed in a flat list under each database.</p> <p>Default: false</p>
SqlCompatibleWindowsAuth	<p>A logical value that determines whether to produce a SQL Server compatible connection string when using Windows Authentication—Trusted_Connection=Yes.</p> <p>If your driver supports Windows Authentication, but requires additional or alternative settings in your connection string, you should set this value to false and use the CredentialConnectionString option record field described below.</p> <p>Default: true</p>

The following table describes the options record fields that are only available through extensibility. Fields that aren't simple literal values are described in subsequent sections.

FIELD	DESCRIPTION
AstVisitor	<p>A record containing one or more overrides to control SQL query generation. The most common usage of this field is to provide logic to generate a LIMIT/OFFSET clause for drivers that don't support TOP.</p> <p>Fields include:</p> <ul style="list-style-type: none"> • Constant • LimitClause <p>See the AstVisitor section for more information.</p>

FIELD	DESCRIPTION
CancelQueryExplicitly	<p>A logical value that instructs the M engine to explicitly cancel any running calls through the ODBC driver before terminating the connection to the ODBC server.</p> <p>This field is useful in situations where query execution is managed independently of the network connections to the server, for example in some Spark deployments. In most cases, this value doesn't need to be set because the query in the server is canceled when the network connection to the server is terminated.</p> <p>Default: false</p>
ClientConnectionPooling	<p>A logical value that enables client-side connection pooling for the ODBC driver. Most drivers will want to set this value to true.</p> <p>Default: false</p>
CredentialConnectionString	<p>A text or record value used to specify credential related connection string properties.</p> <p>See the Credential section for more information.</p>
HideNativeQuery	<p>A logical value that controls whether your connector allows native SQL statements to be passed in by a query using the Value.NativeQuery() function.</p> <p>Note: this functionality is currently not exposed in the Power Query user experience. Users would need to manually edit their queries to take advantage of this capability.</p> <p>Default: false</p>
ImplicitTypeConversions	<p>A table value containing implicit type conversions supported by your driver or backend server. Values in this table are additive to the conversions reported by the driver itself.</p> <p>This field is typically used in conjunction with the SQLGetTypeInfo field when overriding data type information reported by the driver.</p> <p>See the ImplicitTypeConversions section for more information.</p>
OnError	<p>An error handling function that receives an errorRecord parameter of type record.</p> <p>Common uses of this function include handling SSL connection failures, providing a download link if your driver isn't found on the system, and reporting authentication errors.</p> <p>See the OnError section for more information.</p>

FIELD	DESCRIPTION
SoftNumbers	<p>Allows the M engine to select a compatible data type when conversion between two specific numeric types isn't declared as supported in the SQL_CONVERT_* capabilities.</p> <p>Default: false</p>
SqlCapabilities	<p>A record providing various overrides of driver capabilities, and a way to specify capabilities that aren't expressed through ODBC 3.8.</p> <p>See the SqlCapabilities section for more information.</p>
SQLColumns	<p>A function that allows you to modify column metadata returned by the SQLColumns function.</p> <p>See the SQLColumns section for more information.</p>
SQLGetFunctions	<p>A record that allows you to override values returned by calls to SQLGetFunctions.</p> <p>A common use of this field is to disable the use of parameter binding, or to specify that generated queries should use CAST rather than CONVERT.</p> <p>See the SQLGetFunctions section for more information.</p>
SQLGetInfo	<p>A record that allows you to override values returned by calls to SQLGetInfo.</p> <p>See the SQLGetInfo section for more information.</p>
SQLGetTypeInfo	<p>A table, or function that returns a table, that overrides the type information returned by SQLGetTypeInfo.</p> <p>When the value is set to a table, the value completely replaces the type information reported by the driver. SQLGetTypeInfo won't be called.</p> <p>When the value is set to a function, your function will receive the result of the original call to SQLGetTypeInfo, allowing you to modify the table.</p> <p>This field is typically used when there's a mismatch between data types reported by SQLGetTypeInfo and SQLColumns.</p> <p>See the SQLGetTypeInfo section for more information.</p>
SQLTables	<p>A function that allows you to modify the table metadata returned by a call to SQLTables.</p> <p>See the SQLTables section for more information.</p>

FIELD	DESCRIPTION
TolerateConcatOverflow	<p>Allows concatenation of text values to occur even if the result might be truncated to fit within the range of an available type.</p> <p>For example, when concatenating a VARCHAR(4000) field with a VARCHAR(4000) field on a system that supports a maximize VARCHAR size of 4000 and no CLOB type, the concatenation will be folded even though the result might get truncated.</p> <p>Default: false</p>
UseEmbeddedDriver	<p>(internal use): A logical value that controls whether the ODBC driver should be loaded from a local directory (using new functionality defined in the ODBC 4.0 specification). This is generally only set by connectors created by Microsoft that ship with Power Query.</p> <p>When set to false, the system ODBC driver manager will be used to locate and load the driver.</p> <p>Most connectors should not need to set this field.</p> <p>Default: false</p>

Overriding AstVisitor

The AstVisitor field is set through the [Odbc.DataSource](#) options record. It's used to modify SQL statements generated for specific query scenarios.

NOTE

Drivers that support `LIMIT` and `OFFSET` clauses (rather than `TOP`) will want to provide a LimitClause override for AstVisitor.

Constant

Providing an override for this value has been deprecated and may be removed from future implementations.

LimitClause

This field is a function that receives two `Int64.Type` arguments (skip, take), and returns a record with two text fields (Text, Location).

```
LimitClause = (skip as nullable number, take as number) as record => ...
```

The skip parameter is the number of rows to skip (that is, the argument to OFFSET). If an offset is not specified, the skip value will be null. If your driver supports `LIMIT`, but does not support `OFFSET`, the LimitClause function should return an unimplemented error (...) when skip is greater than 0.

The take parameter is the number of rows to take (that is, the argument to LIMIT).

The `Text` field of the result contains the SQL text to add to the generated query.

The `Location` field specifies where to insert the clause. The following table describes supported values.

Value	Description	Example
AfterQuerySpecification	<p>LIMIT clause is put at the end of the generated SQL.</p> <p>This is the most commonly supported LIMIT syntax.</p>	<pre>SELECT a, b, c FROM table WHERE a > 10 LIMIT 5</pre>
BeforeQuerySpecification	LIMIT clause is put before the generated SQL statement.	<pre>LIMIT 5 ROWS SELECT a, b, c FROM table WHERE a > 10</pre>
AfterSelect	LIMIT goes after the SELECT statement, and after any modifiers (such as DISTINCT).	<pre>SELECT DISTINCT LIMIT 5 a, b, c FROM table WHERE a > 10</pre>
AfterSelectBeforeModifiers	LIMIT goes after the SELECT statement, but before any modifiers (such as DISTINCT).	<pre>SELECT LIMIT 5 DISTINCT a, b, c FROM table WHERE a > 10</pre>

The following code snippet provides a LimitClause implementation for a driver that expects a LIMIT clause, with an optional OFFSET, in the following format: [OFFSET <offset> ROWS] LIMIT <row_count>

```
LimitClause = (skip, take) =>
  let
    offset = if (skip > 0) then Text.Format("OFFSET #{0} ROWS", {skip}) else "",
    limit = if (take <> null) then Text.Format("LIMIT #{0}", {take}) else ""
  in
  [
    Text = Text.Format("#{0} #{1}", {offset, limit}),
    Location = "AfterQuerySpecification"
  ]
```

The following code snippet provides a LimitClause implementation for a driver that supports LIMIT, but not OFFSET. Format: LIMIT <row_count> .

```
LimitClause = (skip, take) =>
  if (skip > 0) then error "Skip/Offset not supported"
  else
  [
    Text = Text.Format("LIMIT #{0}", {take}),
    Location = "AfterQuerySpecification"
  ]
```

Overriding SqlCapabilities

FIELD	DETAILS
FractionalSecondsScale	<p>A number value ranging from 1 to 7 that indicates the number of decimal places supported for millisecond values. This value should be set by connectors that want to enable query folding over datetime values.</p> <p>Default: null</p>
PrepareStatements	<p>A logical value that indicates that statements should be prepared using SQLPrepare.</p> <p>Default: false</p>
SupportsTop	<p>A logical value that indicates the driver supports the TOP clause to limit the number of returned rows.</p> <p>Default: false</p>
StringLiteralEscapeCharacters	<p>A list of text values that specify the character(s) to use when escaping string literals and LIKE expressions.</p> <p>Ex. {"\r\n"}</p> <p>Default: null</p>
SupportsDerivedTable	<p>A logical value that indicates the driver supports derived tables (sub-selects).</p> <p>This value is assumed to be true for drivers that set their conformance level to SQL_SC_SQL92_FULL (reported by the driver or overridden with the Sql92Conformance setting (see below)). For all other conformance levels, this value defaults to false.</p> <p>If your driver doesn't report the SQL_SC_SQL92_FULL compliance level, but does support derived tables, set this value to true.</p> <p>Note that supporting derived tables is required for many Direct Query scenarios.</p>
SupportsNumericLiterals	<p>A logical value that indicates whether the generated SQL should include numeric literals values. When set to false, numeric values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsStringLiterals	<p>A logical value that indicates whether the generated SQL should include string literals values. When set to false, string values will always be specified using Parameter Binding.</p> <p>Default: false</p>

FIELD	DETAILS
SupportsOdbcDateLiterals	<p>A logical value that indicates whether the generated SQL should include date literals values. When set to false, date values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimeLiterals	<p>A logical value that indicates whether the generated SQL should include time literals values. When set to false, time values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimestampLiterals	<p>A logical value that indicates whether the generated SQL should include timestamp literals values. When set to false, timestamp values will always be specified using Parameter Binding.</p> <p>Default: false</p>

Overriding SQLColumns

`SQLColumns` is a function handler that receives the results of an ODBC call to [SQLColumns](#). The source parameter contains a table with the data type information. This override is typically used to fix up data type mismatches between calls to `SQLGetTypeInfo` and `SQLColumns`.

For details of the format of the source table parameter, go to [SQLColumns Function](#).

Overriding SQLGetFunctions

This field is used to override SQLFunctions values returned by an ODBC driver. It contains a record whose field names are equal to the FunctionId constants defined for the ODBC [SQLGetFunctions](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#).

FIELD	DETAILS
SQL_CONVERT_FUNCTIONS	<p>Indicates which function(s) are supported when doing type conversions. By default, the M Engine will attempt to use the CONVERT function. Drivers that prefer the use of CAST can override this value to report that only SQL_FN_CVT_CAST (numeric value of 0x2) is supported.</p>
SQL_API_SQLBINDCOL	<p>A logical (true/false) value that indicates whether the Mashup Engine should use the SQLBindCol API when retrieving data. When set to false, SQLGetData is used instead.</p> <p>Default: false</p>

The following code snippet provides an example explicitly telling the M engine to use CAST rather than CONVERT.

```

SQLGetFunctions = [
    SQL_CONVERT_FUNCTIONS = 0x2 /* SQL_FN_CVT_CAST */
]

```

Overriding SQLGetInfo

This field is used to override SQLGetInfo values returned by an ODBC driver. It contains a record whose fields are names equal to the InfoType constants defined for the ODBC [SQLGetInfo](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#). The full list of InfoTypes that are checked can be found in the Mashup Engine trace files.

The following table contains commonly overridden SQLGetInfo properties:

FIELD	DETAILS
SQL_SQL_CONFORMANCE	<p>An integer value that indicates the level of SQL-92 supported by the driver:</p> <ul style="list-style-type: none"> (1) SQL_SC_SQL92_ENTRY = Entry level SQL-92 compliant. (2) SQL_SC_FIPS127_2_TRANSITIONAL = FIPS 127-2 transitional level compliant. (4) SQL_SC_SQL92_INTERMEDIATE = Intermediate level SQL-92 compliant. (8) SQL_SC_SQL92_FULL = Full level SQL-92 compliant. <p>Note that in Power Query scenarios, the connector will be used in a Read Only mode. Most drivers will want to report a SQL_SC_SQL92_FULL compliance level, and override specific SQL generation behavior using the SQLGetInfo and SQLGetFunctions properties.</p>
SQL_SQL92_PREDICATES	<p>A bitmask enumerating the predicates supported in a SELECT statement, as defined in SQL-92.</p> <p>See the SQL_SP_* constants in the ODBC specification.</p>
SQL_AGGREGATE_FUNCTIONS	<p>A bitmask enumerating support for aggregation functions.</p> <p>SQL_AF_ALL SQL_AF_AVG SQL_AF_COUNT SQL_AF_DISTINCT SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM</p> <p>See the SQL_AF_* constants in the ODBC specification.</p>

FIELD	DETAILS
SQL_GROUP_BY	<p>A integer value that specifies the relationship between the columns in the GROUP BY clause and the non-aggregated columns in the select list:</p> <p>SQL_GB_COLLATE = A COLLATE clause can be specified at the end of each grouping column.</p> <p>SQL_GB_NOT_SUPPORTED = GROUP BY clauses are not supported.</p> <p>SQL_GB_GROUP_BY_EQUALS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It cannot contain any other columns. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT.</p> <p>SQL_GB_GROUP_BY_CONTAINS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It can contain columns that are not in the select list. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>SQL_GB_NO_RELATION = The columns in the GROUP BY clause and the select list are not related. The meaning of non-grouped, non-aggregated columns in the select list is data source-dependent. For example, SELECT DEPT, SALARY FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>See the SQL_GB_* constants in the ODBC specification.</p>

The following helper function can be used to create bitmask values from a list of integer values:

```
Flags = (flags as list) =>
    let
        Loop = List.Generate(
            ()=> [i = 0, Combined = 0],
            each [i] < List.Count(flags),
            each [i = [i]+1, Combined =*Number.BitwiseOr([Combined], flags{i})],
            each [Combined]),
        Result = List.Last(Loop, 0)
    in
        Result;
```

Overriding SQLGetTypeInfo

`SQLGetTypeInfo` can be specified in two ways:

- A fixed `table` value that contains the same type information as an ODBC call to `sqlGetTypeInfo`.
- A function that accepts a table argument, and returns a table. The argument will contain the original results of the ODBC call to `sqlGetTypeInfo`. Your function implementation can modify/add to this table.

The first approach is used to completely override the values returned by the ODBC driver. The second approach is used if you want to add to or modify these values.

For details of the format of the types table parameter and expected return value, see the [SQLGetTypeInfo function reference](#).

SQLGetTypeInfo using a static table

The following code snippet provides a static implementation for SQLGetTypeInfo.

```

SQLGetTypeInfo = #table(
    { "TYPE_NAME",      "DATA_TYPE", "COLUMN_SIZE", "LITERAL_PREF", "LITERAL_SUFFIX", "CREATE_PARAS",
    "NULLABLE", "CASE_SENSITIVE", "SEARCHABLE", "UNSIGNED_ATTRIBUTE", "FIXED_PREC_SCALE", "AUTO_UNIQUE_VALUE",
    "LOCAL_TYPE_NAME", "MINIMUM_SCALE", "MAXIMUM_SCALE", "SQL_DATA_TYPE", "SQL_DATETIME_SUB", "NUM_PREC_RADIX",
    "INTERNAL_PRECISION", "USER_DATA_TYPE" }, {
        { "char",           1,       65535,      "",      "",      "max. length",
1,           1,           3,       null,      0,      null,
"char",       null,       null,      -8,      null,
0,           0,           0,       null,      10,      null,
        { "int8",          -5,       19,      "",      "",      null,
1,           0,           2,       0,      -5,      0,
"int8",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "bit",           -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bit",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "bool",          -7,       1,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"bool",       null,       null,      -7,      null,
0,           0,           0,       null,      null,
        { "date",           9,       10,      "",      "",      null,
1,           0,           2,       null,      0,      null,
"date",       null,       null,      9,      1,
0,           0,           0,       null,      null,
        { "numeric",         3,       28,      null,      null,      null,
1,           0,           2,       0,      0,
"numeric",       0,       0,       null,      null,
0,           0,           0,       null,      10,
        { "float8",          8,       15,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "float8",          6,       17,      null,      null,      null,
1,           0,           2,       0,      0,
"float8",       null,       null,      6,      null,
0,           0,           0,       null,      2,
        { "uuid",          -11,       37,      null,      null,      null,
1,           0,           2,       null,      0,
"uuid",       null,       null,      -11,      null,
0,           0,           0,       null,      null,
        { "int4",            4,       10,      null,      null,      null,
1,           0,           2,       0,      0,
"int4",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "text",            -1,       65535,      "",      "",      null,
1,           1,           3,       null,      0,      null,
"text",       null,       null,      -10,      null,
0,           0,           0,       null,      null,
        { "lo",              -4,       255,      "",      "",      null,
1,           0,           2,       null,      0,
"lo",       null,       null,      -4,      null,
0,           0,           0,       null,      null,
        { "numeric",          2,       28,      null,      null,      "precision, scale",
1,           0,           2,       0,      10,
"numeric",       0,       6,       2,      null,
0,           0,           0,       null,      10,
        { "float4",            7,       9,       null,      null,      null,
1,           0,           2,       0,      10,
"float4",       null,       null,      7,      null,
0,           0,           0,       null,      2,
        { "int2",            5,       19,      null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
        { "int2",            -6,       5,       null,      null,      null,
1,           0,           2,       0,      10,
"int2",       0,       0,       null,      null,
0,           0,           0,       null,      2,
    }
}

```

```

0,          0      },
0, { "timestamp", 11, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "date", 91, 10, null, "", "", null,
1, 0, 2, null, 9, 1, null,
"date", null, }, 1,
0, 0 }, null,
{ "timestamp", 93, 26, null, "", "", null,
1, 0, 2, 38, 9, 3, null,
"timestamp", 0, }, null,
0, { "bytea", -3, 255, null, "", "", null,
1, 0, 2, null, 0, null,
"bytea", null, }, null,
0, 0 }, null,
{ "varchar", 12, 65535, null, "", "", "max. length",
1, 0, 2, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "char", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"char", null, }, null,
0, 0 }, null,
{ "text", -10, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"text", null, }, null,
0, 0 }, null,
{ "varchar", -9, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "bpchar", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"bpchar", null, }, null,
0, 0 } }
);

```

SQLGetTypeInfo using a function

The following code snippets append the `bpchar` type to the existing types returned by the driver.

```

SQLGetTypeInfo = (types as table) as table =>
let
    newTypes = #table(
    {
        "TYPE_NAME",
        "DATA_TYPE",
        "COLUMN_SIZE",
        "LITERAL_PREF",
        "LITERAL_SUFFIX",
        "CREATE_PARAS",
        "NULLABLE",
        "CASE_SENSITIVE",
        "SEARCHABLE",
        "UNSIGNED_ATTRIBUTE",
        "FIXED_PREC_SCALE",
        "AUTO_UNIQUE_VALUE",
        "LOCAL_TYPE_NAME",
        "MINIMUM_SCALE",
        "MAXIMUM_SCALE",
        "SQL_DATA_TYPE",
        "SQL_DATETIME_SUB",
        "NUM_PREC_RADIX",
        "INTERNAL_PRECISION",
        "USER_DATA_TYPE"
    },
    // we add a new entry for each type we want to add
    {
        {
            "bpchar",
            -8,
            65535,
            """",
            """",
            "max. length",
            1,
            1,
            3,
            null,
            0,
            null,
            "bpchar",
            null,
            null,
            -9,
            null,
            null,
            0,
            0
        }
    )),
    append = Table.Combine({types, newTypes})
in
    append;

```

Setting the Connection String

The connection string for your ODBC driver is set using the first argument to the [Odbc.DataSource](#) and/or [Odbc.Query](#) functions. The value can be text, or an M record. When using the record, each field in the record will become a property in the connection string. All connection strings will require a Driver field (or DSN field if you require users to pre-configure a system level DSN). Credential related properties will be set separately (see below). Other properties will be driver specific.

The code snippet below shows the definition of a new data source function, creation of the `ConnectionString` record, and invocation of the [Odbc.DataSource](#) function.

```
[DataSource.Kind="SqlODBC", Publish="SqlODBC.Publish"]
shared SqlODBC.Contents = (server as text) =>
    let
        ConnectionString = [
            Driver = "SQL Server Native Client 11.0",
            Server = server,
            MultiSubnetFailover = "Yes",
            ApplicationIntent = "ReadOnly",
            APP = "PowerBICustomConnector"
        ],
        OdbcDatasource = Odbc.DataSource(ConnectionString)
    in
        OdbcDatasource;
```

Troubleshooting and Testing

To enable tracing in Power BI Desktop:

1. Go to **File > Options and settings > Options**.
2. Select on the **Diagnostics** tab.
3. Select the **Enable tracing** option.
4. Select the **Open traces folder** link (should be `%LOCALAPPDATA%/Microsoft/Power BI Desktop/Traces`).
5. Delete existing trace files.
6. Perform your tests.
7. Close Power BI Desktop to ensure all log files are flushed to disk.

Here are steps you can take for initial testing in Power BI Desktop:

1. Close Power BI Desktop.
2. Clear your trace directory.
3. Open Power BI desktop, and enable tracing.
4. Connect to your data source, and select Direct Query mode.
5. Select a table in the navigator, and select **Edit**.
6. Manipulate the query in various ways, including:

- Take the First N rows (for example, 10).
- Set equality filters on different data types (int, string, bool, and so on).
- Set other range filters (greater than, less than).
- Filter on NULL / NOT NULL.
- Select a sub-set of columns.
- Aggregate / Group By different column combinations.
- Add a column calculated from other columns ($[C] = [A] + [B]$).
- Sort on one column, multiple columns. 7. Expressions that fail to fold will result in a warning bar. Note the failure, remove the step, and move to the next test case. Details about the cause of the failure should be emitted to the trace logs. 8. Close Power BI Desktop. 9. Copy the trace files to a new directory. 10. Use the recommend Power BI workbook to parse and analyze the trace files.

Once you have simple queries working, you can then try Direct Query scenarios (for example, building reports in the Report Views). The queries generated in Direct Query mode will be significantly more complex (that is, use of sub-selects, COALESCE statements, and aggregations).

Concatenation of strings in Direct Query mode

The M engine does basic type size limit validation as part of its query folding logic. If you are receiving a folding

error when trying to concatenate two strings that potentially overflow the maximum size of the underlying database type:

1. Ensure that your database can support up-conversion to CLOB types when string concat overflow occurs.
2. Set the `TolerateConcatOverflow` option for `Odbc.DataSource` to `true`.

The [DAX CONCATENATE function](#) is currently not supported by Power Query/ODBC extensions. Extension authors should ensure string concatenation works through the query editor by adding calculated columns (`[stringCol1] & [stringCol2]`). When the capability to fold the CONCATENATE operation is added in the future, it should work seamlessly with existing extensions.

Enabling Direct Query for an ODBC based connector

1/15/2022 • 21 minutes to read • [Edit Online](#)

Overview

Using M's built-in [Odbc.DataSource](#) function is the recommended way to create custom connectors for data sources that have an existing ODBC driver and/or support a SQL query syntax. Wrapping the [Odbc.DataSource](#) function will allow your connector to inherit default query folding behavior based on the capabilities reported by your driver. This will enable the M engine to generate SQL statements based on filters and other transformations defined by the user within the Power Query experience, without having to provide this logic within the connector itself.

ODBC extensions can optionally enable Direct Query mode, allowing Power BI to dynamically generate queries at runtime without pre-caching the user's data model.

NOTE

Enabling Direct Query support raises the difficulty and complexity level of your connector. When Direct Query is enabled, Power BI will prevent the M engine from compensating for operations that cannot be fully pushed to the underlying data source.

This section builds on the concepts presented in the M Extensibility Reference, and assumes familiarity with the creation of a basic Data Connector.

Refer to the [SqlODBC sample](#) for most of the code examples in the sections below. Additional samples can be found in the ODBC samples directory.

ODBC Extensibility Functions

The M engine provides two ODBC related data source functions: [Odbc.DataSource](#), and [Odbc.Query](#).

The [Odbc.DataSource](#) function provides a default navigation table with all databases, tables, and views from your system, supports query folding, and allows for a range of customization options. The majority of ODBC based extensions will use this as their primary extensibility function. The function accepts two arguments—a connection string, and an options record to provide behavior overrides.

The [Odbc.Query](#) function allows you to execute SQL statements through an ODBC driver. It acts as a passthrough for query execution. Unlike the [Odbc.DataSource](#) function, it doesn't provide query folding functionality, and requires that SQL queries be provided by the connector (or end user). When building a custom connector, this function is typically used internally to run queries to retrieve metadata that might not be exposed through regular ODBC channels. The function accepts two arguments—a connection string, and a SQL query.

Parameters for your Data Source Function

Custom connectors can accept any number of function arguments, but to remain consistent with the built-in data source functions shipped with Power Query, the following guidelines are recommended:

- Require the minimal set of parameters used to establish a connection to your server. The less parameters end users need to provide, the easier your connector will be to use.

- Although you can define parameters with a fixed number of values (that is, a dropdown list in the UI), parameters are entered before the user is authenticated. Any values that can be discovered programmatically after the user is authenticated (such as catalog or database name) should be selectable through the Navigator. The default behavior for the [Odbc.DataSource](#) function will be to return a hierarchical navigation table consisting of Catalog (Database), Schema, and Table names, although this can be overridden within your connector.
- If you feel your users will typically know what values to enter for items they would select from the Navigator (such as the database name), make these parameters optional. Parameters that can be discovered programmatically should not be made required.
- The last parameter for your function should be an optional record called "options". This parameter typically allows advanced users to set common ODBC related properties (such as CommandTimeout), set behavior overrides specific to your connector, and allows for future extensibility without impacting backwards compatibility for your function.
- Security/credential related arguments MUST never be part of your data source function parameters, as values entered in the connect dialog will be persisted to the user's query. Credential related parameters should be specified as part of the connector's supported Authentication methods.

By default, all required parameters for your data source function are factored into the Data Source Path value used to identify user credentials.

Note that while the UI for the built-in [Odbc.DataSource](#) function provides a dropdown that allows the user to select a DSN, this functionality is not available through extensibility. If your data source configuration is complex enough to require a fully customizable configuration dialog, it's recommended you require your end users to pre-configure a system DSN, and have your function take in the DSN name as a text field.

Parameters for Odbc.DataSource

The [Odbc.DataSource](#) function takes two parameters—a connectionString for your driver, and an options record that lets you override various driver behaviors. Through the options record you can override capabilities and other information reported by the driver, control the navigator behavior, and affect the SQL queries generated by the M engine.

The supported options records fields fall into two categories—those that are public / always available, and those that are only available in an extensibility context.

The following table describes the public fields in the options record.

FIELD	DESCRIPTION
CommandTimeout	A duration value that controls how long the server-side query is allowed to run before it's cancelled. Default: 10 minutes
ConnectionTimeout	A duration value that controls how long to wait before abandoning an attempt to make a connection to the server. Default: 15 seconds

FIELD	DESCRIPTION
CreateNavigationProperties	<p>A logical value that sets whether to generate navigation properties on the returned tables. Navigation properties are based on foreign key relationships reported by the driver, and show up as "virtual" columns that can be expanded in the query editor, creating the appropriate join.</p> <p>If calculating foreign key dependencies is an expensive operation for your driver, you may want to set this value to false.</p> <p>Default: true</p>
HierarchicalNavigation	<p>A logical value that sets whether to view the tables grouped by their schema names. When set to false, tables will be displayed in a flat list under each database.</p> <p>Default: false</p>
SqlCompatibleWindowsAuth	<p>A logical value that determines whether to produce a SQL Server compatible connection string when using Windows Authentication—Trusted_Connection=Yes.</p> <p>If your driver supports Windows Authentication, but requires additional or alternative settings in your connection string, you should set this value to false and use the CredentialConnectionString option record field described below.</p> <p>Default: true</p>

The following table describes the options record fields that are only available through extensibility. Fields that aren't simple literal values are described in subsequent sections.

FIELD	DESCRIPTION
AstVisitor	<p>A record containing one or more overrides to control SQL query generation. The most common usage of this field is to provide logic to generate a LIMIT/OFFSET clause for drivers that don't support TOP.</p> <p>Fields include:</p> <ul style="list-style-type: none"> • Constant • LimitClause <p>See the AstVisitor section for more information.</p>

FIELD	DESCRIPTION
CancelQueryExplicitly	<p>A logical value that instructs the M engine to explicitly cancel any running calls through the ODBC driver before terminating the connection to the ODBC server.</p> <p>This field is useful in situations where query execution is managed independently of the network connections to the server, for example in some Spark deployments. In most cases, this value doesn't need to be set because the query in the server is canceled when the network connection to the server is terminated.</p> <p>Default: false</p>
ClientConnectionPooling	<p>A logical value that enables client-side connection pooling for the ODBC driver. Most drivers will want to set this value to true.</p> <p>Default: false</p>
CredentialConnectionString	<p>A text or record value used to specify credential related connection string properties.</p> <p>See the Credential section for more information.</p>
HideNativeQuery	<p>A logical value that controls whether your connector allows native SQL statements to be passed in by a query using the Value.NativeQuery() function.</p> <p>Note: this functionality is currently not exposed in the Power Query user experience. Users would need to manually edit their queries to take advantage of this capability.</p> <p>Default: false</p>
ImplicitTypeConversions	<p>A table value containing implicit type conversions supported by your driver or backend server. Values in this table are additive to the conversions reported by the driver itself.</p> <p>This field is typically used in conjunction with the SQLGetTypeInfo field when overriding data type information reported by the driver.</p> <p>See the ImplicitTypeConversions section for more information.</p>
OnError	<p>An error handling function that receives an errorRecord parameter of type record.</p> <p>Common uses of this function include handling SSL connection failures, providing a download link if your driver isn't found on the system, and reporting authentication errors.</p> <p>See the OnError section for more information.</p>

FIELD	DESCRIPTION
SoftNumbers	<p>Allows the M engine to select a compatible data type when conversion between two specific numeric types isn't declared as supported in the SQL_CONVERT_* capabilities.</p> <p>Default: false</p>
SqlCapabilities	<p>A record providing various overrides of driver capabilities, and a way to specify capabilities that aren't expressed through ODBC 3.8.</p> <p>See the SqlCapabilities section for more information.</p>
SQLColumns	<p>A function that allows you to modify column metadata returned by the SQLColumns function.</p> <p>See the SQLColumns section for more information.</p>
SQLGetFunctions	<p>A record that allows you to override values returned by calls to SQLGetFunctions.</p> <p>A common use of this field is to disable the use of parameter binding, or to specify that generated queries should use CAST rather than CONVERT.</p> <p>See the SQLGetFunctions section for more information.</p>
SQLGetInfo	<p>A record that allows you to override values returned by calls to SQLGetInfo.</p> <p>See the SQLGetInfo section for more information.</p>
SQLGetTypeInfo	<p>A table, or function that returns a table, that overrides the type information returned by SQLGetTypeInfo.</p> <p>When the value is set to a table, the value completely replaces the type information reported by the driver. SQLGetTypeInfo won't be called.</p> <p>When the value is set to a function, your function will receive the result of the original call to SQLGetTypeInfo, allowing you to modify the table.</p> <p>This field is typically used when there's a mismatch between data types reported by SQLGetTypeInfo and SQLColumns.</p> <p>See the SQLGetTypeInfo section for more information.</p>
SQLTables	<p>A function that allows you to modify the table metadata returned by a call to SQLTables.</p> <p>See the SQLTables section for more information.</p>

FIELD	DESCRIPTION
TolerateConcatOverflow	<p>Allows concatenation of text values to occur even if the result might be truncated to fit within the range of an available type.</p> <p>For example, when concatenating a VARCHAR(4000) field with a VARCHAR(4000) field on a system that supports a maximize VARCHAR size of 4000 and no CLOB type, the concatenation will be folded even though the result might get truncated.</p> <p>Default: false</p>
UseEmbeddedDriver	<p>(internal use): A logical value that controls whether the ODBC driver should be loaded from a local directory (using new functionality defined in the ODBC 4.0 specification). This is generally only set by connectors created by Microsoft that ship with Power Query.</p> <p>When set to false, the system ODBC driver manager will be used to locate and load the driver.</p> <p>Most connectors should not need to set this field.</p> <p>Default: false</p>

Overriding AstVisitor

The AstVisitor field is set through the [Odbc.DataSource](#) options record. It's used to modify SQL statements generated for specific query scenarios.

NOTE

Drivers that support `LIMIT` and `OFFSET` clauses (rather than `TOP`) will want to provide a LimitClause override for AstVisitor.

Constant

Providing an override for this value has been deprecated and may be removed from future implementations.

LimitClause

This field is a function that receives two `Int64.Type` arguments (skip, take), and returns a record with two text fields (Text, Location).

```
LimitClause = (skip as nullable number, take as number) as record => ...
```

The skip parameter is the number of rows to skip (that is, the argument to OFFSET). If an offset is not specified, the skip value will be null. If your driver supports `LIMIT`, but does not support `OFFSET`, the LimitClause function should return an unimplemented error (...) when skip is greater than 0.

The take parameter is the number of rows to take (that is, the argument to LIMIT).

The `Text` field of the result contains the SQL text to add to the generated query.

The `Location` field specifies where to insert the clause. The following table describes supported values.

Value	Description	Example
AfterQuerySpecification	<p>LIMIT clause is put at the end of the generated SQL.</p> <p>This is the most commonly supported LIMIT syntax.</p>	<pre>SELECT a, b, c FROM table WHERE a > 10 LIMIT 5</pre>
BeforeQuerySpecification	LIMIT clause is put before the generated SQL statement.	<pre>LIMIT 5 ROWS SELECT a, b, c FROM table WHERE a > 10</pre>
AfterSelect	LIMIT goes after the SELECT statement, and after any modifiers (such as DISTINCT).	<pre>SELECT DISTINCT LIMIT 5 a, b, c FROM table WHERE a > 10</pre>
AfterSelectBeforeModifiers	LIMIT goes after the SELECT statement, but before any modifiers (such as DISTINCT).	<pre>SELECT LIMIT 5 DISTINCT a, b, c FROM table WHERE a > 10</pre>

The following code snippet provides a LimitClause implementation for a driver that expects a LIMIT clause, with an optional OFFSET, in the following format: [OFFSET <offset> ROWS] LIMIT <row_count>

```
LimitClause = (skip, take) =>
  let
    offset = if (skip > 0) then Text.Format("OFFSET #{0} ROWS", {skip}) else "",
    limit = if (take <> null) then Text.Format("LIMIT #{0}", {take}) else ""
  in
  [
    Text = Text.Format("#{0} #{1}", {offset, limit}),
    Location = "AfterQuerySpecification"
  ]
```

The following code snippet provides a LimitClause implementation for a driver that supports LIMIT, but not OFFSET. Format: LIMIT <row_count> .

```
LimitClause = (skip, take) =>
  if (skip > 0) then error "Skip/Offset not supported"
  else
  [
    Text = Text.Format("LIMIT #{0}", {take}),
    Location = "AfterQuerySpecification"
  ]
```

Overriding SqlCapabilities

FIELD	DETAILS
FractionalSecondsScale	<p>A number value ranging from 1 to 7 that indicates the number of decimal places supported for millisecond values. This value should be set by connectors that want to enable query folding over datetime values.</p> <p>Default: null</p>
PrepareStatements	<p>A logical value that indicates that statements should be prepared using SQLPrepare.</p> <p>Default: false</p>
SupportsTop	<p>A logical value that indicates the driver supports the TOP clause to limit the number of returned rows.</p> <p>Default: false</p>
StringLiteralEscapeCharacters	<p>A list of text values that specify the character(s) to use when escaping string literals and LIKE expressions.</p> <p>Ex. {"\r\n"}</p> <p>Default: null</p>
SupportsDerivedTable	<p>A logical value that indicates the driver supports derived tables (sub-selects).</p> <p>This value is assumed to be true for drivers that set their conformance level to SQL_SC_SQL92_FULL (reported by the driver or overridden with the Sql92Conformance setting (see below)). For all other conformance levels, this value defaults to false.</p> <p>If your driver doesn't report the SQL_SC_SQL92_FULL compliance level, but does support derived tables, set this value to true.</p> <p>Note that supporting derived tables is required for many Direct Query scenarios.</p>
SupportsNumericLiterals	<p>A logical value that indicates whether the generated SQL should include numeric literals values. When set to false, numeric values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsStringLiterals	<p>A logical value that indicates whether the generated SQL should include string literals values. When set to false, string values will always be specified using Parameter Binding.</p> <p>Default: false</p>

FIELD	DETAILS
SupportsOdbcDateLiterals	<p>A logical value that indicates whether the generated SQL should include date literals values. When set to false, date values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimeLiterals	<p>A logical value that indicates whether the generated SQL should include time literals values. When set to false, time values will always be specified using Parameter Binding.</p> <p>Default: false</p>
SupportsOdbcTimestampLiterals	<p>A logical value that indicates whether the generated SQL should include timestamp literals values. When set to false, timestamp values will always be specified using Parameter Binding.</p> <p>Default: false</p>

Overriding SQLColumns

`SQLColumns` is a function handler that receives the results of an ODBC call to [SQLColumns](#). The source parameter contains a table with the data type information. This override is typically used to fix up data type mismatches between calls to `SQLGetTypeInfo` and `SQLColumns`.

For details of the format of the source table parameter, go to [SQLColumns Function](#).

Overriding SQLGetFunctions

This field is used to override SQLFunctions values returned by an ODBC driver. It contains a record whose field names are equal to the FunctionId constants defined for the ODBC [SQLGetFunctions](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#).

FIELD	DETAILS
SQL_CONVERT_FUNCTIONS	<p>Indicates which function(s) are supported when doing type conversions. By default, the M Engine will attempt to use the CONVERT function. Drivers that prefer the use of CAST can override this value to report that only SQL_FN_CVT_CAST (numeric value of 0x2) is supported.</p>
SQL_API_SQLBINDCOL	<p>A logical (true/false) value that indicates whether the Mashup Engine should use the SQLBindCol API when retrieving data. When set to false, SQLGetData is used instead.</p> <p>Default: false</p>

The following code snippet provides an example explicitly telling the M engine to use CAST rather than CONVERT.

```

SQLGetFunctions = [
    SQL_CONVERT_FUNCTIONS = 0x2 /* SQL_FN_CVT_CAST */
]

```

Overriding SQLGetInfo

This field is used to override SQLGetInfo values returned by an ODBC driver. It contains a record whose fields are names equal to the InfoType constants defined for the ODBC [SQLGetInfo](#) function. Numeric constants for each of these fields can be found in the [ODBC specification](#). The full list of InfoTypes that are checked can be found in the Mashup Engine trace files.

The following table contains commonly overridden SQLGetInfo properties:

FIELD	DETAILS
SQL_SQL_CONFORMANCE	<p>An integer value that indicates the level of SQL-92 supported by the driver:</p> <ul style="list-style-type: none"> (1) SQL_SC_SQL92_ENTRY = Entry level SQL-92 compliant. (2) SQL_SC_FIPS127_2_TRANSITIONAL = FIPS 127-2 transitional level compliant. (4) SQL_SC_SQL92_INTERMEDIATE = Intermediate level SQL-92 compliant. (8) SQL_SC_SQL92_FULL = Full level SQL-92 compliant. <p>Note that in Power Query scenarios, the connector will be used in a Read Only mode. Most drivers will want to report a SQL_SC_SQL92_FULL compliance level, and override specific SQL generation behavior using the SQLGetInfo and SQLGetFunctions properties.</p>
SQL_SQL92_PREDICATES	<p>A bitmask enumerating the predicates supported in a SELECT statement, as defined in SQL-92.</p> <p>See the SQL_SP_* constants in the ODBC specification.</p>
SQL_AGGREGATE_FUNCTIONS	<p>A bitmask enumerating support for aggregation functions.</p> <p>SQL_AF_ALL SQL_AF_AVG SQL_AF_COUNT SQL_AF_DISTINCT SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM</p> <p>See the SQL_AF_* constants in the ODBC specification.</p>

FIELD	DETAILS
SQL_GROUP_BY	<p>A integer value that specifies the relationship between the columns in the GROUP BY clause and the non-aggregated columns in the select list:</p> <p>SQL_GB_COLLATE = A COLLATE clause can be specified at the end of each grouping column.</p> <p>SQL_GB_NOT_SUPPORTED = GROUP BY clauses are not supported.</p> <p>SQL_GB_GROUP_BY_EQUALS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It cannot contain any other columns. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT.</p> <p>SQL_GB_GROUP_BY_CONTAINS_SELECT = The GROUP BY clause must contain all non-aggregated columns in the select list. It can contain columns that are not in the select list. For example, SELECT DEPT, MAX(SALARY) FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>SQL_GB_NO_RELATION = The columns in the GROUP BY clause and the select list are not related. The meaning of non-grouped, non-aggregated columns in the select list is data source-dependent. For example, SELECT DEPT, SALARY FROM EMPLOYEE GROUP BY DEPT, AGE.</p> <p>See the SQL_GB_* constants in the ODBC specification.</p>

The following helper function can be used to create bitmask values from a list of integer values:

```
Flags = (flags as list) =>
    let
        Loop = List.Generate(
            ()=> [i = 0, Combined = 0],
            each [i] < List.Count(flags),
            each [i = [i]+1, Combined =*Number.BitwiseOr([Combined], flags{i})],
            each [Combined]),
        Result = List.Last(Loop, 0)
    in
        Result;
```

Overriding SQLGetTypeInfo

`SQLGetTypeInfo` can be specified in two ways:

- A fixed `table` value that contains the same type information as an ODBC call to `sqlGetTypeInfo`.
- A function that accepts a table argument, and returns a table. The argument will contain the original results of the ODBC call to `sqlGetTypeInfo`. Your function implementation can modify/add to this table.

The first approach is used to completely override the values returned by the ODBC driver. The second approach is used if you want to add to or modify these values.

For details of the format of the types table parameter and expected return value, see the [SQLGetTypeInfo function reference](#).

SQLGetTypeInfo using a static table

The following code snippet provides a static implementation for SQLGetTypeInfo.


```

0,          0      },
0, { "timestamp", 11, 26, null, "", "", null,
1, 0, 2, 38, 9, 0, null,
"timestamp", 0, }, 3,
0, { "date", 91, 10, null, "", "", null,
1, 0, 2, null, 9, 1, null,
"date", null, }, 1,
0, 0 }, null,
{ "timestamp", 93, 26, null, "", "", null,
1, 0, 2, 38, 9, 3, null,
"timestamp", 0, }, null,
0, { "bytea", -3, 255, null, "", "", null,
1, 0, 2, null, 0, null,
"bytea", null, }, null,
0, 0 }, null,
{ "varchar", 12, 65535, null, "", "", "max. length",
1, 0, 2, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "char", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"char", null, }, null,
0, 0 }, null,
{ "text", -10, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"text", null, }, null,
0, 0 }, null,
{ "varchar", -9, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"varchar", null, }, null,
0, 0 }, null,
{ "bpchar", -8, 65535, null, "", "", "max. length",
1, 1, 3, null, 0, null,
"bpchar", null, }, null,
0, 0 } }
);

```

SQLGetTypeInfo using a function

The following code snippets append the `bpchar` type to the existing types returned by the driver.

```

SQLGetTypeInfo = (types as table) as table =>
let
    newTypes = #table(
    {
        "TYPE_NAME",
        "DATA_TYPE",
        "COLUMN_SIZE",
        "LITERAL_PREF",
        "LITERAL_SUFFIX",
        "CREATE_PARAS",
        "NULLABLE",
        "CASE_SENSITIVE",
        "SEARCHABLE",
        "UNSIGNED_ATTRIBUTE",
        "FIXED_PREC_SCALE",
        "AUTO_UNIQUE_VALUE",
        "LOCAL_TYPE_NAME",
        "MINIMUM_SCALE",
        "MAXIMUM_SCALE",
        "SQL_DATA_TYPE",
        "SQL_DATETIME_SUB",
        "NUM_PREC_RADIX",
        "INTERNAL_PRECISION",
        "USER_DATA_TYPE"
    },
    // we add a new entry for each type we want to add
    {
        {
            "bpchar",
            -8,
            65535,
            """",
            """",
            "max. length",
            1,
            1,
            3,
            null,
            0,
            null,
            "bpchar",
            null,
            null,
            -9,
            null,
            null,
            0,
            0
        }
    )),
    append = Table.Combine({types, newTypes})
in
    append;

```

Setting the Connection String

The connection string for your ODBC driver is set using the first argument to the [Odbc.DataSource](#) and/or [Odbc.Query](#) functions. The value can be text, or an M record. When using the record, each field in the record will become a property in the connection string. All connection strings will require a Driver field (or DSN field if you require users to pre-configure a system level DSN). Credential related properties will be set separately (see below). Other properties will be driver specific.

The code snippet below shows the definition of a new data source function, creation of the `ConnectionString` record, and invocation of the [Odbc.DataSource](#) function.

```
[DataSource.Kind="SqlODBC", Publish="SqlODBC.Publish"]
shared SqlODBC.Contents = (server as text) =>
    let
        ConnectionString = [
            Driver = "SQL Server Native Client 11.0",
            Server = server,
            MultiSubnetFailover = "Yes",
            ApplicationIntent = "ReadOnly",
            APP = "PowerBICustomConnector"
        ],
        OdbcDatasource = Odbc.DataSource(ConnectionString)
    in
        OdbcDatasource;
```

Troubleshooting and Testing

To enable tracing in Power BI Desktop:

1. Go to **File > Options and settings > Options**.
2. Select on the **Diagnostics** tab.
3. Select the **Enable tracing** option.
4. Select the **Open traces folder** link (should be `%LOCALAPPDATA%/Microsoft/Power BI Desktop/Traces`).
5. Delete existing trace files.
6. Perform your tests.
7. Close Power BI Desktop to ensure all log files are flushed to disk.

Here are steps you can take for initial testing in Power BI Desktop:

1. Close Power BI Desktop.
2. Clear your trace directory.
3. Open Power BI desktop, and enable tracing.
4. Connect to your data source, and select Direct Query mode.
5. Select a table in the navigator, and select **Edit**.
6. Manipulate the query in various ways, including:

- Take the First N rows (for example, 10).
- Set equality filters on different data types (int, string, bool, and so on).
- Set other range filters (greater than, less than).
- Filter on NULL / NOT NULL.
- Select a sub-set of columns.
- Aggregate / Group By different column combinations.
- Add a column calculated from other columns ($[C] = [A] + [B]$).
- Sort on one column, multiple columns. 7. Expressions that fail to fold will result in a warning bar. Note the failure, remove the step, and move to the next test case. Details about the cause of the failure should be emitted to the trace logs. 8. Close Power BI Desktop. 9. Copy the trace files to a new directory. 10. Use the recommend Power BI workbook to parse and analyze the trace files.

Once you have simple queries working, you can then try Direct Query scenarios (for example, building reports in the Report Views). The queries generated in Direct Query mode will be significantly more complex (that is, use of sub-selects, COALESCE statements, and aggregations).

Concatenation of strings in Direct Query mode

The M engine does basic type size limit validation as part of its query folding logic. If you are receiving a folding

error when trying to concatenate two strings that potentially overflow the maximum size of the underlying database type:

1. Ensure that your database can support up-conversion to CLOB types when string concat overflow occurs.
2. Set the `TolerateConcatOverflow` option for `Odbc.DataSource` to `true`.

The [DAX CONCATENATE function](#) is currently not supported by Power Query/ODBC extensions. Extension authors should ensure string concatenation works through the query editor by adding calculated columns (`[stringCol1] & [stringCol2]`). When the capability to fold the CONCATENATE operation is added in the future, it should work seamlessly with existing extensions.

Handling Resource Path

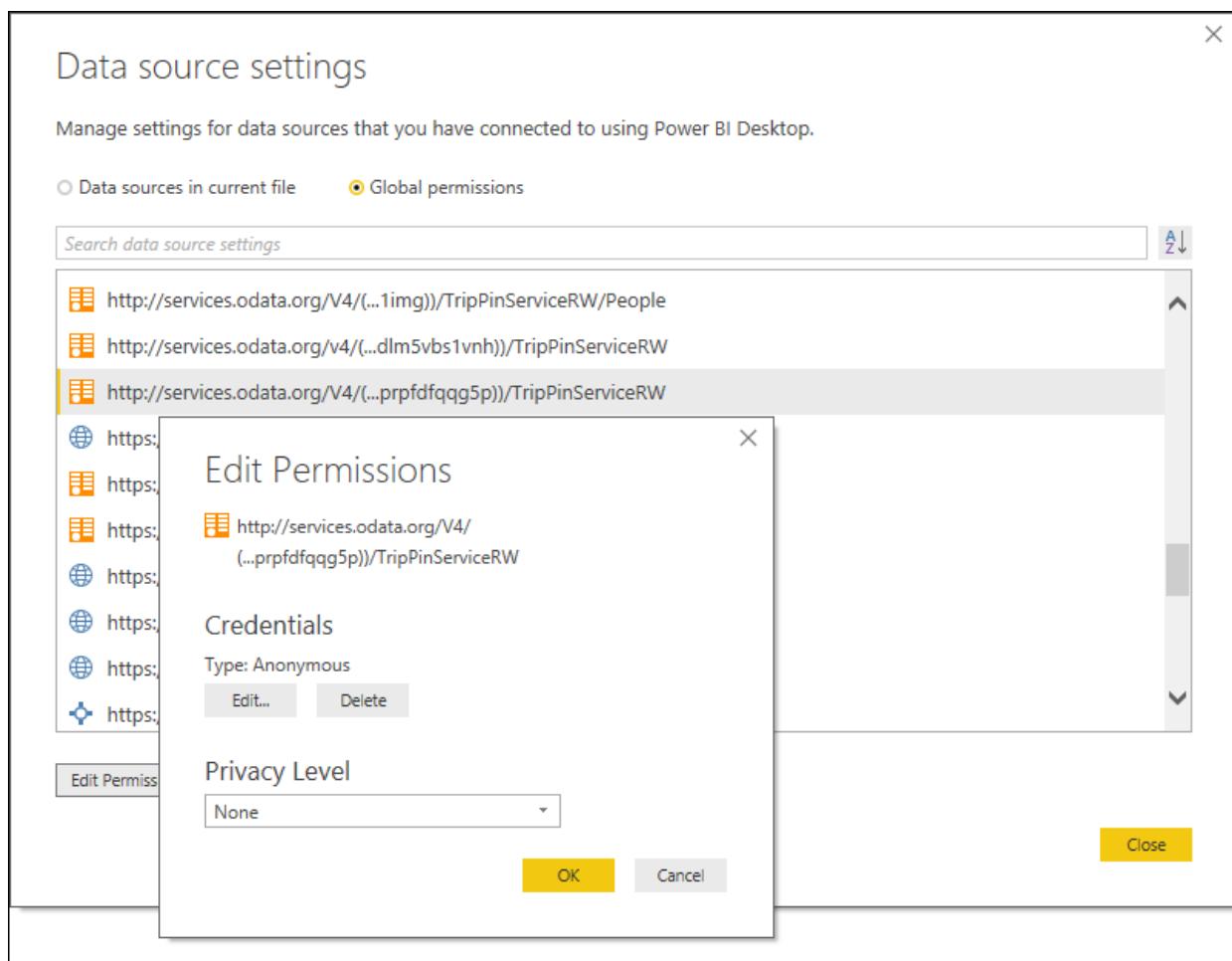
1/15/2022 • 2 minutes to read • [Edit Online](#)

The M engine identifies a data source using a combination of its *Kind* and *Path*. When a data source is encountered during a query evaluation, the M engine will try to find matching credentials. If no credentials are found, the engine returns a special error that results in a credential prompt in Power Query.

The *Kind* value comes from [Data Source Kind](#) definition.

The *Path* value is derived from the *required parameters* of your data source function(s). Optional parameters aren't factored into the data source path identifier. As a result, all data source functions associated with a data source kind must have the same parameters. There's special handling for functions that have a single parameter of type `Uri.Type`. See below for further details.

You can see an example of how credentials are stored in the [Data source settings](#) dialog in Power BI Desktop. In this dialog, the Kind is represented by an icon, and the Path value is displayed as text.



[Note] If you change your data source function's required parameters during development, previously stored credentials will no longer work (because the path values no longer match). You should delete any stored credentials any time you change your data source function parameters. If incompatible credentials are found, you may receive an error at runtime.

Data Source Path Format

The *Path* value for a data source is derived from the data source function's required parameters.

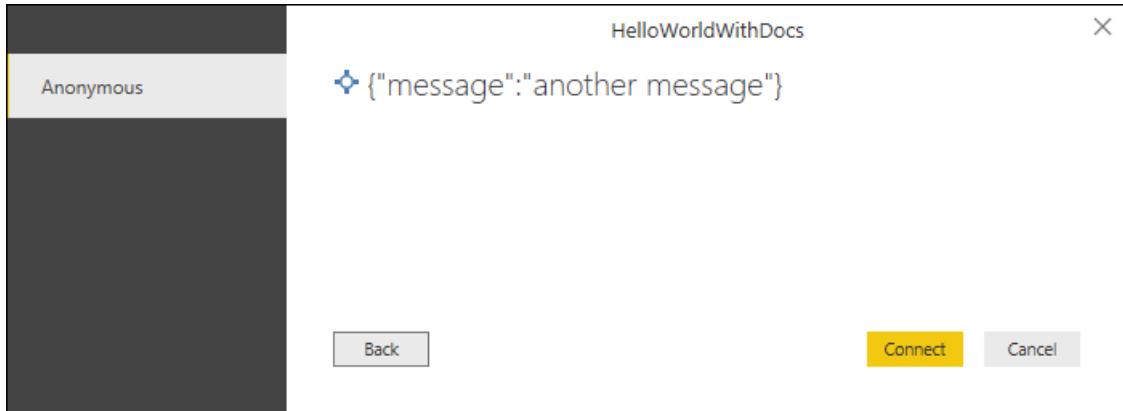
By default, you can see the actual string value in the **Data source settings** dialog in Power BI Desktop, and in the credential prompt. If the Data Source Kind definition has included a `Label` value, you'll see the label value instead.

For example, the data source function in the [HelloWorldWithDocs sample](#) has the following signature:

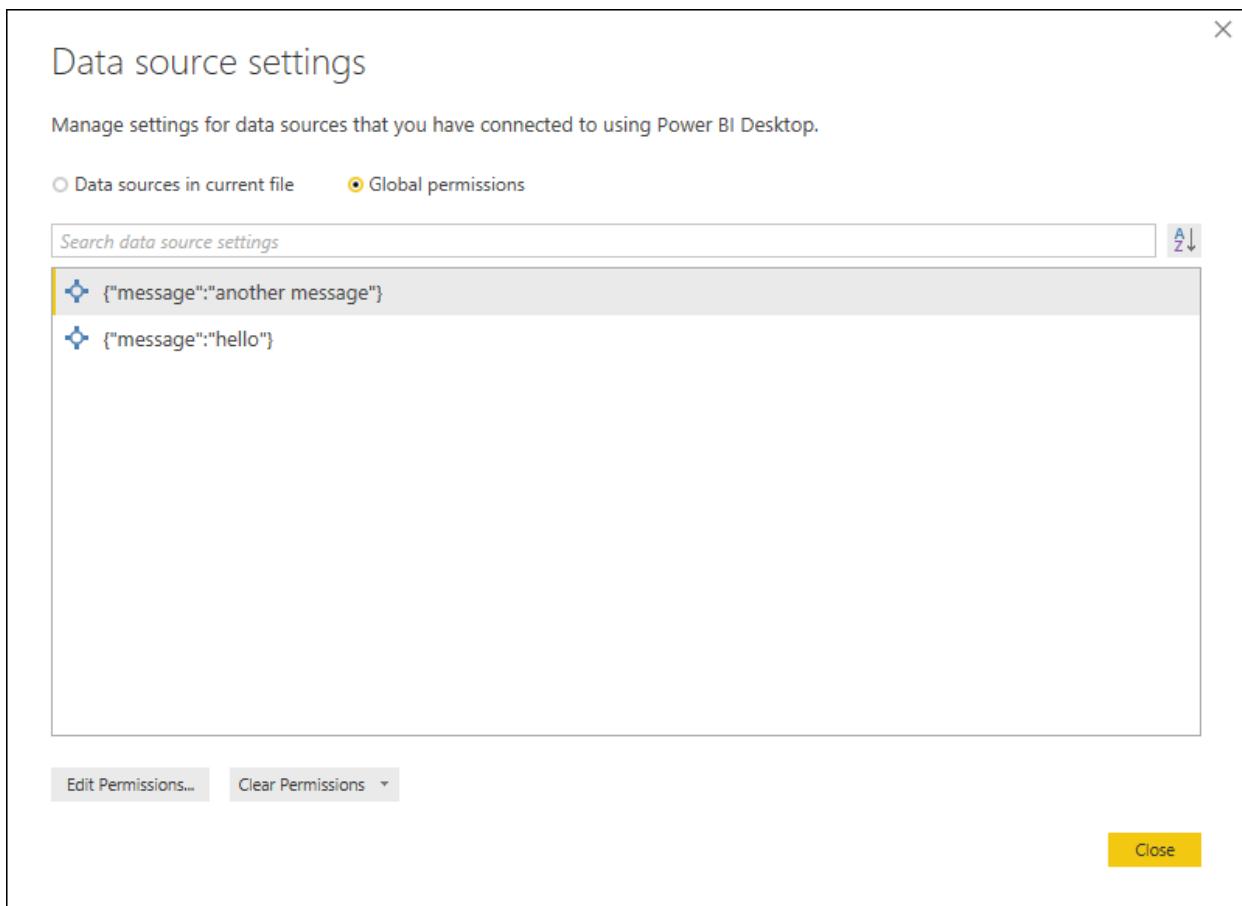
```
HelloWorldWithDocs.Contents = (message as text, optional count as number) as table => ...
```

The function has a single required parameter (`message`) of type `text`, and will be used to calculate the data source path. The optional parameter (`count`) will be ignored. The path would be displayed as follows:

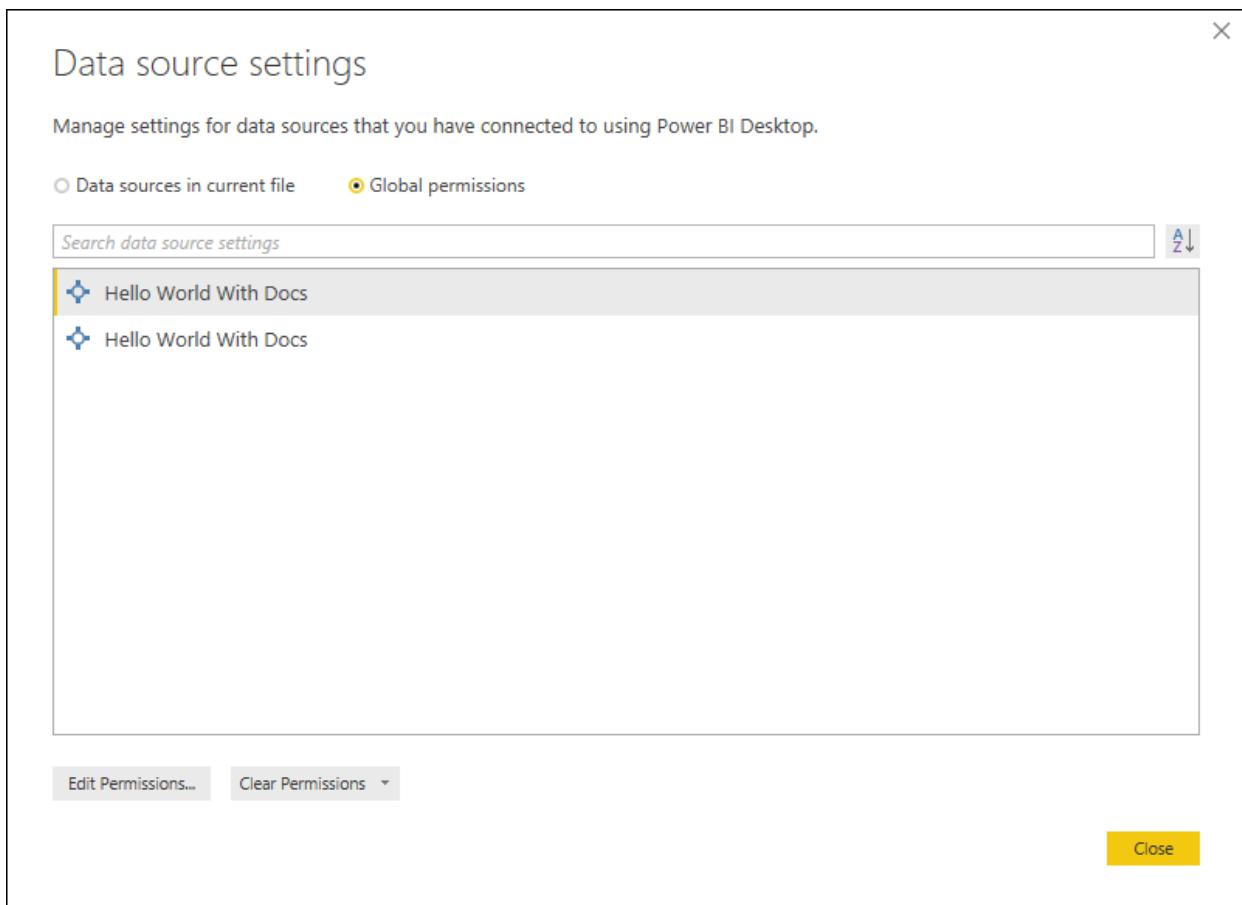
Credential prompt:



Data source settings UI:



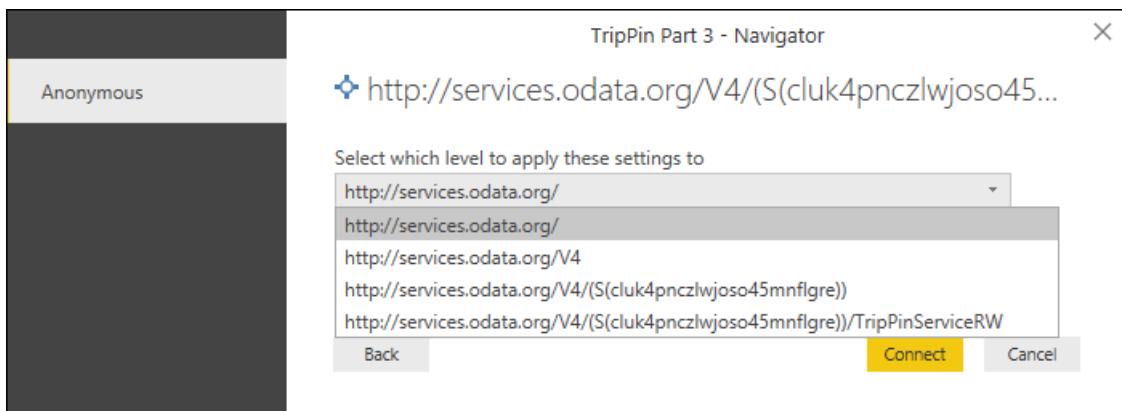
When a Label value is defined, the data source path value won't be shown:



[Note] We currently recommend that you *do not* include a Label for your data source if your function has required parameters, as users won't be able to distinguish between the different credentials they've entered. We are hoping to improve this in the future (that is, allowing data connectors to display their own custom data source paths).

Functions with a Uri parameter

Because data sources with a Uri-based identifier are so common, there's special handling in the Power Query UI when dealing with Uri-based data source paths. When a Uri-based data source is encountered, the credential dialog provides a dropdown allowing the user to select the base path, rather than the full path (and all paths in-between).



As `Uri.Type` is an *ascribed type* rather than a *primitive type* in the M language, you'll need to use the `Value.ReplaceType` function to indicate that your text parameter should be treated as a Uri.

```
shared GithubSample.Contents = Value.ReplaceType(Github.Contents, type function (url as Uri.type) as any);
```

Paging

1/15/2022 • 2 minutes to read • [Edit Online](#)

REST APIs typically have some mechanism to transmit large volumes of records broken up into *pages* of results. Power Query has the flexibility to support many different paging mechanisms. However, since each paging mechanism is different, some amount of modification of the paging examples is likely to be necessary to fit your situation.

Typical Patterns

The heavy lifting of compiling all page results into a single table is performed by the [Table.GenerateByPage\(\)](#) helper function, which can generally be used with no modification. The code snippets presented in the

[Table.GenerateByPage\(\)](#) helper function section describe how to implement some common paging patterns.

Regardless of pattern, you'll need to understand:

1. How do you request the next page of data?
2. Does the paging mechanism involve calculating values, or do you extract the URL for the next page from the response?
3. How do you know when to stop paging?
4. Are there parameters related to paging (such as "page size") that you should be aware of?

Handling Transformations

1/15/2022 • 3 minutes to read • [Edit Online](#)

For situations where the data source response isn't presented in a format that Power BI can consume directly, Power Query can be used to perform a series of transformations.

Static Transformations

In most cases, the data is presented in a consistent way by the data source: column names, data types, and hierarchical structure are consistent for a given endpoint. In this situation it's appropriate to always apply the same set of transformations to get the data in a format acceptable to Power BI.

An example of static transformation can be found in the [TripPin Part 2 - Data Connector for a REST Service](#) tutorial when the data source is treated as a standard REST service:

```
let
    Source = TripPin.Feed("https://services.odata.org/v4/TripPinService/Airlines"),
    value = Source[value],
    toTable = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    expand = Table.ExpandRecordColumn(toTable, "Column1", {"AirlineCode", "Name"}, {"AirlineCode", "Name"})
in
    expand
```

The transformations in this example are:

1. `Source` is a Record returned from a call to `TripPin.Feed(...)`.
2. You pull the value from one of `Source`'s key-value pairs. The name of the key is `value`, and you store the result in a variable called `value`.
3. `value` is a list, which you convert to a table. Each element in `value` becomes a row in the table, which you can call `toTable`.
4. Each element in `value` is itself a Record. `toTable` has all of these in a single column: `"Column1"`. This step pulls all data with key `"AirlineCode"` into a column called `"AirlineCode"` and all data with key `"Name"` into a column called `"Name"`, for each row in `toTable`. `"Column1"` is replaced by these two new columns.

At the end of the day you're left with data in a simple tabular format that Power BI can consume and easily render:

	AirlineCode	Name
1	AA	American Airlines
2	FM	Shanghai Airline
3	MU	China Eastern Airlines

It's important to note that a sequence of static transformations of this specificity are only applicable to a *single* endpoint. In the example above, this sequence of transformations will only work if `"AirlineCode"` and `"Name"` exist in the REST endpoint response, since they are hard-coded into the M code. Thus, this sequence of transformations may not work if you try to hit the `/Event` endpoint.

This high level of specificity may be necessary for pushing data to a navigation table, but for more general data access functions it's recommended that you only perform transformations that are appropriate for all endpoints.

NOTE

Be sure to test transformations under a variety of data circumstances. If the user doesn't have any data at the `/airlines` endpoint, do your transformations result in an empty table with the correct schema? Or is an error encountered during evaluation? See [TripPin Part 7: Advanced Schema with M Types](#) for a discussion on unit testing.

Dynamic Transformations

More complex logic is sometimes needed to convert API responses into stable and consistent forms appropriate for Power BI data models.

Inconsistent API Responses

Basic M control flow (if statements, HTTP status codes, try...catch blocks, and so on) are typically sufficient to handle situations where there are a handful of ways in which the API responds.

Determining Schema On-The-Fly

Some APIs are designed such that multiple pieces of information must be combined to get the correct tabular format. Consider Smartsheet's `/sheets` endpoint response, which contains an array of column names and an array of data rows. The Smartsheet Connector is able to parse this response in the following way:

```
raw = Web.Contents(...),
columns = raw[columns],
columnTitles = List.Transform(columns, each [title]),
columnTitlesWithRowNumber = List.InsertRange(columnTitles, 0, {"RowNumber"}),

RowAsList = (row) =>
    let
        listOfCells = row[cells],
        cellValuesList = List.Transform(listOfCells, each if Record.HasFields(_, "value") then [value]
            else null),
        rowNumberFirst = List.InsertRange(cellValuesList, 0, {row[rowNumber]}) 
    in
        rowNumberFirst,

listOfRows = List.Transform(raw[rows], each RowAsList(_)),
result = Table.FromRows(listOfRows, columnTitlesWithRowNumber)
```

1. First deal with column header information. You can pull the `title` record of each column into a List, prepending with a `RowNumber` column that you know will always be represented as this first column.
2. Next you can define a function that allows you to parse a row into a List of cell `value`s. You can again prepend `rowNumber` information.
3. Apply your `RowAsList()` function to each of the `row`s returned in the API response.
4. Convert the List to a table, specifying the column headers.

Handling Transformations

1/15/2022 • 3 minutes to read • [Edit Online](#)

For situations where the data source response isn't presented in a format that Power BI can consume directly, Power Query can be used to perform a series of transformations.

Static Transformations

In most cases, the data is presented in a consistent way by the data source: column names, data types, and hierarchical structure are consistent for a given endpoint. In this situation it's appropriate to always apply the same set of transformations to get the data in a format acceptable to Power BI.

An example of static transformation can be found in the [TripPin Part 2 - Data Connector for a REST Service](#) tutorial when the data source is treated as a standard REST service:

```
let
    Source = TripPin.Feed("https://services.odata.org/v4/TripPinService/Airlines"),
    value = Source[value],
    toTable = Table.FromList(value, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    expand = Table.ExpandRecordColumn(toTable, "Column1", {"AirlineCode", "Name"}, {"AirlineCode", "Name"})
in
    expand
```

The transformations in this example are:

1. `Source` is a Record returned from a call to `TripPin.Feed(...)`.
2. You pull the value from one of `Source`'s key-value pairs. The name of the key is `value`, and you store the result in a variable called `value`.
3. `value` is a list, which you convert to a table. Each element in `value` becomes a row in the table, which you can call `toTable`.
4. Each element in `value` is itself a Record. `toTable` has all of these in a single column: `"Column1"`. This step pulls all data with key `"AirlineCode"` into a column called `"AirlineCode"` and all data with key `"Name"` into a column called `"Name"`, for each row in `toTable`. `"Column1"` is replaced by these two new columns.

At the end of the day you're left with data in a simple tabular format that Power BI can consume and easily render:

	AirlineCode	Name
1	AA	American Airlines
2	FM	Shanghai Airline
3	MU	China Eastern Airlines

It's important to note that a sequence of static transformations of this specificity are only applicable to a *single* endpoint. In the example above, this sequence of transformations will only work if `"AirlineCode"` and `"Name"` exist in the REST endpoint response, since they are hard-coded into the M code. Thus, this sequence of transformations may not work if you try to hit the `/Event` endpoint.

This high level of specificity may be necessary for pushing data to a navigation table, but for more general data access functions it's recommended that you only perform transformations that are appropriate for all endpoints.

NOTE

Be sure to test transformations under a variety of data circumstances. If the user doesn't have any data at the `/airlines` endpoint, do your transformations result in an empty table with the correct schema? Or is an error encountered during evaluation? See [TripPin Part 7: Advanced Schema with M Types](#) for a discussion on unit testing.

Dynamic Transformations

More complex logic is sometimes needed to convert API responses into stable and consistent forms appropriate for Power BI data models.

Inconsistent API Responses

Basic M control flow (if statements, HTTP status codes, try...catch blocks, and so on) are typically sufficient to handle situations where there are a handful of ways in which the API responds.

Determining Schema On-The-Fly

Some APIs are designed such that multiple pieces of information must be combined to get the correct tabular format. Consider Smartsheet's `/sheets` endpoint response, which contains an array of column names and an array of data rows. The Smartsheet Connector is able to parse this response in the following way:

```
raw = Web.Contents(...),
columns = raw[columns],
columnTitles = List.Transform(columns, each [title]),
columnTitlesWithRowNumber = List.InsertRange(columnTitles, 0, {"RowNumber"}),

RowAsList = (row) =>
    let
        listOfCells = row[cells],
        cellValuesList = List.Transform(listOfCells, each if Record.HasFields(_, "value") then [value]
            else null),
        rowNumberFirst = List.InsertRange(cellValuesList, 0, {row[rowNumber]})
    in
        rowNumberFirst,

listOfRows = List.Transform(raw[rows], each RowAsList(_)),
result = Table.FromRows(listOfRows, columnTitlesWithRowNumber)
```

1. First deal with column header information. You can pull the `title` record of each column into a List, prepending with a `RowNumber` column that you know will always be represented as this first column.
2. Next you can define a function that allows you to parse a row into a List of cell `value`s. You can again prepend `rowNumber` information.
3. Apply your `RowAsList()` function to each of the `row`s returned in the API response.
4. Convert the List to a table, specifying the column headers.

Handling Schema

1/15/2022 • 7 minutes to read • [Edit Online](#)

Depending on your data source, information about data types and column names may or may not be provided explicitly. OData REST APIs typically handle this using the [\\$metadata definition](#), and the Power Query [OData.Feed](#) method automatically handles parsing this information and applying it to the data returned from an [OData source](#).

Many REST APIs don't have a way to programmatically determine their schema. In these cases you'll need to include a schema definition in your connector.

Simple Hardcoded Approach

The simplest approach is to hardcode a schema definition into your connector. This is sufficient for most use cases.

Overall, enforcing a schema on the data returned by your connector has multiple benefits, such as:

- Setting the correct data types.
- Removing columns that don't need to be shown to end users (such as internal IDs or state information).
- Ensuring that each page of data has the same shape by adding any columns that might be missing from a response (REST APIs commonly indicate that fields should be null by omitting them entirely).

Viewing the Existing Schema with [Table.Schema](#)

Consider the following code that returns a simple table from the [TripPin OData sample service](#):

```
let
    url = "https://services.odata.org/TripPinWeb ApiService/Airlines",
    source = Json.Document(Web.Contents(url))[value],
    asTable = Table.FromRecords(source)
in
    asTable
```

NOTE

TripPin is an OData source, so realistically it would make more sense to simply use the [OData.Feed](#) function's automatic schema handling. In this example you'll be treating the source as a typical REST API and using [Web.Contents](#) to demonstrate the technique of hardcoding a schema by hand.

This table is the result:

	ABC 123 AirlineCode	ABC 123 Name
1	AA	American Airlines
2	FM	Shanghai Airline
3	MU	China Eastern Airlines

You can use the handy [Table.Schema](#) function to check the data type of the columns:

```

let
    url = "https://services.odata.org/TripPinWeb ApiService/Airlines",
    source = Json.Document(Web.Contents(url))[value],
    asTable = Table.FromRecords(source)
in
    Table.Schema(asTable)

```

	Name	Position	TypeName	Kind	IsNullable
1	AirlineCode	0	Any.Type	any	TRUE
2	Name	1	Any.Type	any	TRUE

Both AirlineCode and Name are of `any` type. `Table.Schema` returns a lot of metadata about the columns in a table, including names, positions, type information, and many advanced properties such as Precision, Scale, and MaxLength. For now you should only concern yourself with the ascribed type (`TypeName`), primitive type (`kind`), and whether the column value might be null (`IsNullable`).

Defining a Simple Schema Table

Your schema table will be composed of two columns:

COLUMN	DETAILS
Name	The name of the column. This must match the name in the results returned by the service.
Type	The M data type you're going to set. This can be a primitive type (text, number, datetime, and so on), or an ascribed type (Int64.Type, Currency.Type, and so on).

The hardcoded schema table for the `Airlines` table will set its `AirlineCode` and `Name` columns to `text` and looks like this:

```

Airlines = #table({"Name", "Type"}, {
    {"AirlineCode", type text},
    {"Name", type text}
})

```

As you look to some of the other endpoints, consider the following schema tables:

The `Airports` table has four fields you'll want to keep (including one of type `record`):

```

Airports = #table({"Name", "Type"}, {
    {"IcaoCode", type text},
    {"Name", type text},
    {"IataCode", type text},
    {"Location", type record}
})

```

The `People` table has seven fields, including `list<S(Emails, AddressInfo)`, a `nullable` column (`Gender`), and a column with an *ascribed type* (`Concurrency`):

```

People = #table({ "Name", "Type"}, {
    {"UserName", type text},
    {"FirstName", type text},
    {"LastName", type text},
    {"Emails", type list},
    {"AddressInfo", type list},
    {"Gender", type nullable text},
    {"Concurrency", Int64.Type}
})

```

You can put all of these tables into a single master schema table `SchemaTable`:

```

SchemaTable = #table({ "Entity", "SchemaTable"}, {
    {"Airlines", Airlines},
    {"Airports", Airports},
    {"People", People}
})

```

	ABC 123 Entity	ABC 123 SchemaTable
1	Airlines	Table
2	Airports	Table
3	People	Table
Name Type		
AirlineCode	Type	
Name	Type	

The SchemaTransformTable Helper Function

The `SchemaTransformTable` helper function described below will be used to enforce schemas on your data. It takes the following parameters:

PARAMETER	TYPE	DESCRIPTION
table	table	The table of data you'll want to enforce your schema on.
schema	table	<p>The schema table to read column info from, with the following type:</p> <pre>type table [Name = text, Type = type]</pre>

PARAMETER	TYPE	DESCRIPTION
enforceSchema	number	<p>(optional) An enum that controls behavior of the function. The default value (<code>EnforceSchema.Strict = 1</code>) ensures that the output table will match the schema table that was provided by adding any missing columns, and removing extra columns.</p> <p>The <code>EnforceSchema.IgnoreExtraColumns = 2</code> option can be used to preserve extra columns in the result.</p> <p>When <code>EnforceSchema.IgnoreMissingColumns = 3</code> is used, both missing columns and extra columns will be ignored.</p>

The logic for this function looks something like this:

1. Determine if there are any missing columns from the source table.
2. Determine if there are any extra columns.
3. Ignore structured columns (of type `list`, `record`, and `table`), and columns set to type `any`.
4. Use `Table.TransformColumnTypes` to set each column type.
5. Reorder columns based on the order they appear in the schema table.
6. Set the type on the table itself using `Value.ReplaceType`.

NOTE

The last step to set the table type will remove the need for the Power Query UI to infer type information when viewing the results in the query editor, which can sometimes result in a double-call to the API.

Putting It All Together

In the greater context of a complete extension, the schema handling will take place when a table is returned from the API. Typically this functionality takes place at the lowest level of the paging function (if one exists), with entity information passed through from a navigation table.

Because so much of the implementation of paging and navigation tables is context-specific, the complete example of implementing a hardcoded schema-handling mechanism won't be shown here. [This TripPin example](#) demonstrates how an end-to-end solution might look.

Sophisticated Approach

The hardcoded implementation discussed above does a good job of making sure that schemas remain consistent for simple JSON responses, but it's limited to parsing the first level of the response. Deeply nested data sets would benefit from the following approach, which takes advantage of M Types.

Here is a quick refresh about types in the M language from the [Language Specification](#):

A **type value** is a value that **classifies** other values. A value that is classified by a type is said to **conform** to that type. The M type system consists of the following kinds of types:

- Primitive types, which classify primitive values (`binary`, `date`, `datetime`, `datetimezone`, `duration`,

`list`, `logical`, `null`, `number`, `record`, `text`, `time`, `type`) and also include a number of abstract types (`function`, `table`, `any`, and `none`).

- Record types, which classify record values based on field names and value types.
- List types, which classify lists using a single item base type.
- Function types, which classify function values based on the types of their parameters and return values.
- Table types, which classify table values based on column names, column types, and keys.
- Nullable types, which classify the value null in addition to all the values classified by a base type.
- Type types, which classify values that are types.

Using the raw JSON output you get (and/or by looking up the definitions in the [service's \\$metadata](#)), you can define the following record types to represent OData complex types:

```
LocationType = type [
    Address = text,
    City = CityType,
    Loc = LocType
];

CityType = type [
    CountryRegion = text,
    Name = text,
    Region = text
];

LocType = type [
    "#type" = text,
    coordinates = {number},
    crs = CrsType
];

CrsType = type [
    "#type" = text,
    properties = record
];
```

Notice how `LocationType` references the `CityType` and `LocType` to represent its structured columns.

For the top-level entities that you'll want represented as Tables, you can define *table types*:

```
AirlinesType = type table [
    AirlineCode = text,
    Name = text
];
AirportsType = type table [
    Name = text,
    IataCode = text,
    Location = LocationType
];
PeopleType = type table [
    UserName = text,
    FirstName = text,
    LastName = text,
    Emails = {text},
    AddressInfo = {nullable LocationType},
    Gender = nullable text,
    Concurrency Int64.Type
];
```

You can then update your `SchemaTable` variable (which you can use as a lookup table for entity-to-type mappings) to use these new type definitions:

```
SchemaTable = #table({ "Entity", "Type" }, {
    {"Airlines", AirlinesType},
    {"Airports", AirportsType},
    {"People", PeopleType}
});
```

You can rely on a common function (`Table.ChangeType`) to enforce a schema on your data, much like you used `SchemaTransformTable` in the earlier exercise. Unlike `SchemaTransformTable`, `Table.ChangeType` takes an actual M table type as an argument, and will apply your schema *recursively* for all nested types. Its signature is:

```
Table.ChangeType = (table, tableType as type) as nullable table => ...
```

NOTE

For flexibility, the function can be used on tables as well as lists of records (which is how tables are represented in a JSON document).

You'll then need to update the connector code to change the `schema` parameter from a `table` to a `type`, and add a call to `Table.ChangeType`. Again, the details for doing so are very implementation-specific and thus not worth going into in detail here. [This extended TripPin connector example](#) demonstrates an end-to-end solution implementing this more sophisticated approach to handling schema.

Status Code Handling with `Web.Contents`

1/15/2022 • 2 minutes to read • [Edit Online](#)

The `Web.Contents` function has some built in functionality for dealing with certain HTTP status codes. The default behavior can be overridden in your extension using the `ManualStatusHandling` field in the [options record](#).

Automatic retry

`Web.Contents` will automatically retry requests that fail with one of the following status codes:

CODE	STATUS
408	Request Timeout
429	Too Many Requests
503	Service Unavailable
504	Gateway Timeout
509	Bandwidth Limit Exceeded

Requests will be retried up to 3 times before failing. The engine uses an exponential back-off algorithm to determine how long to wait until the next retry, unless the response contains a `Retry-after` header. When the header is found, the engine will wait the specified number of seconds before the next retry. The minimum supported wait time is 0.5 seconds, and the maximum value is 120 seconds.

NOTE

The `Retry-after` value must be in the `delta-seconds` format. The `HTTP-date` format is currently not supported.

Authentication exceptions

The following status codes will result in a credentials exception, causing an authentication prompt asking the user to provide credentials (or re-login in the case of an expired OAuth token).

CODE	STATUS
401	Unauthorized
403	Forbidden

NOTE

Extensions are able to use the `ManualStatusHandling` option with status codes 401 and 403, which is not something that can be done in `Web.Contents` calls made outside of an extension context (that is, directly from Power Query).

Redirection

The follow status codes will result in an automatic redirect to the URI specified in the `Location` header. A missing `Location` header will result in an error.

CODE	STATUS
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
307	Temporary Redirect

NOTE

Only status code 307 will keep a `POST` request method. All other redirect status codes will result in a switch to `GET`.

Wait-Retry Pattern

1/15/2022 • 2 minutes to read • [Edit Online](#)

In some situations a data source's behavior does not match that expected by Power Query's [default HTTP code handling](#). The examples below show how to work around this situation.

In this scenario you'll be working with a REST API that occasionally returns a 500 status code, indicating an internal server error. In these instances, you could wait a few seconds and retry, potentially a few times before you give up.

ManualStatusHandling

If `Web.Contents` gets a 500 status code response, it throws a `DataSource.Error` by default. You can override this behavior by providing a list of codes as an optional argument to `Web.Contents`:

```
response = Web.Contents(url, [ManualStatusHandling={404, 500}])
```

By specifying the status codes in this way, Power Query will continue to process the web response as normal. However, normal response processing is often not appropriate in these cases. You'll need to understand that an abnormal response code has been received and perform special logic to handle it. To determine the response code that was returned from the web service, you can access it from the `meta` Record that accompanies the response:

```
responseCode = Value.Metadata(response)[Response.Status]
```

Based on whether `responseCode` is 200 or 500, you can either process the result as normal, or follow your wait-retry logic that you'll flesh out in the next section.

NOTE

We recommended that you use `Binary.Buffer` to force Power Query to cache the `Web.Contents` results if you'll be implementing complex logic such as the Wait-Retry pattern shown here. This prevents Power Query's multi-threaded execution from making multiple calls with potentially inconsistent results.

Value.WaitFor

`Value.WaitFor()` is a standard [helper function](#) that can usually be used with no modification. It works by building a List of retry attempts.

producer Argument

This contains the task to be (possibly) retried. It's represented as a function so that the iteration number can be used in the `producer` logic. The expected behavior is that `producer` will return `null` if a retry is determined to be necessary. If anything other than `null` is returned by `producer`, that value is in turn returned by `Value.WaitFor`.

delay Argument

This contains the logic to execute between retries. It's represented as a function so that the iteration number can be used in the `delay` logic. The expected behavior is that `delay` returns a Duration.

count Argument (optional)

A maximum number of retries can be set by providing a number to the `count` argument.

Putting It All Together

The following example shows how `ManualStatusHandling` and `Value.WaitFor` can be used to implement a delayed retry in the event of a 500 response. Wait time between retries here is shown as doubling with each try, with a maximum of 5 retries.

```
let
  waitForResult = Value.WaitFor(
    (iteration) =>
      let
        result = Web.Contents(url, [ManualStatusHandling = {500}]),
        buffered = Binary.Buffer(result),
        status = Value.Metadata(result)[Response.Status],
        actualResult = if status = 500 then null else buffered
      in
        actualResult,
    (iteration) => #duration(0, 0, 0, Number.Power(2, iteration)),
    5)
in
  waitForResult,
```

Handling Unit Testing

1/15/2022 • 2 minutes to read • [Edit Online](#)

For both simple and complex connectors, adding unit tests is a best practice and highly recommended.

Unit testing is accomplished in the context of Visual Studio's [Power Query SDK](#). Each test is defined as a `Fact` that has a name, an expected value, and an actual value. In most cases, the "actual value" will be an M expression that tests part of your expression.

Consider a very simple extension that exports three functions:

```
section UnitTesting;

shared UnitTesting.ReturnsABC = () => "ABC";
shared UnitTesting.Returns123 = () => "123";
shared UnitTesting.ReturnTableWithFiveRows = () => Table.Repeat(#table({{"a"}},{{1}}),5);
```

This unit test code is made up of a number of Facts, and a bunch of common code for the unit test framework (`ValueToText`, `Fact`, `Facts`, `Facts.Summarize`). The following code provides an example set of Facts (see [UnitTesting.query.pq](#) for the common code):

```
section UnitTestingTests;

shared MyExtension.UnitTesting =
[
    // Put any common variables here if you only want them to be evaluated once

    // Fact(<Name of the Test>, <Expected Value>, <Actual Value>)
    facts =
    {
        Fact("Check that this function returns 'ABC'",           // name of the test
              "ABC",                                         // expected value
              UnitTesting.ReturnsABC()                      // expression to evaluate (let or single statement)
        ),
        Fact("Check that this function returns '123'",          // name of the test
              "123",                                         // expected value
              UnitTesting.Returns123()                     // expression to evaluate (let or single statement)
        ),
        Fact("Result should contain 5 rows",
              5,
              Table.RowCount(UnitTesting.ReturnTableWithFiveRows())
        ),
        Fact("Values should be equal (using a let statement)",
              "Hello World",
              let
                  a = "Hello World"
              in
                  a
        )
    },
    report = Facts.Summarize(facts)
][report];
```

Running the sample in Visual Studio will evaluate all of the Facts and give you a visual summary of the pass rates:

M Query Output

Output	Log	Errors	Credentials
Query Result			
Result	Notes	Details	
Success	All 4 Passed !!! ✓	100% success rate	
Success ✓	Check that this function returns 'ABC'	("ABC" = "ABC")	
Success ✓	Check that this function returns '123'	("123" = "123")	
Success ✓	Result should contain 5 rows	(5 = 5)	
Success ✓	Values should be equal (using a let statement)	("Hello World" = "Hello World")	

Implementing unit testing early in the connector development process enables you to follow the principles of test-driven development. Imagine that you need to write a function called `Uri.GetHost` that returns only the host data from a URI. You might start by writing a test case to verify that the function appropriately performs the expected function:

```
Fact("Returns host from URI",
    "https://bing.com",
    Uri.GetHost("https://bing.com/subpath/query?param=1&param2=hello")
),
Fact("Handles port number appropriately",
    "https://bing.com:8080",
    Uri.GetHost("https://bing.com:8080/subpath/query?param=1&param2=hello")
)
```

Additional tests can be written to ensure that the function appropriately handles edge cases.

An early version of the function might pass some but not all tests:

```
Uri.GetHost = (url) =>
  let
    parts = Uri.Parts(url)
  in
    parts[Scheme] & "://" & parts[Host]
```

M Query Output

Output	Log	Errors	Credentials
Query Result			
Result	Notes	Details	
⊖	5 Passed ⓘ 1 Failed ⓘ	83% success rate	
Success ✓	Check that this function returns 'ABC'	("ABC" = "ABC")	
Success ✓	Check that this function returns '123'	("123" = "123")	
Success ✓	Result should contain 5 rows	(5 = 5)	
Success ✓	Values should be equal (using a let statement)	("Hello World" = "Hello World")	
Success ✓	Returns host from URI	("https://bing.com" = "https://bing.com")	
Failure ⊖	Handles port number appropriately	("https://bing.com:8080" <> "https://bing.com")	

The [final version of the function](#) should pass all unit tests. This also makes it easy to ensure that future updates to the function do not accidentally remove any of its basic functionality.

M Query Output

Output Log Errors Credentials

Query Result

Result	Notes	Details
Success	All 6 Passed !!! ✓	100% success rate
Success ✓	Check that this function returns 'ABC'	("ABC" = "ABC")
Success ✓	Check that this function returns '123'	("123" = "123")
Success ✓	Result should contain 5 rows	(5 = 5)
Success ✓	Values should be equal (using a let statement)	("Hello World" = "Hello World")
Success ✓	Returns host from URI	("https://bing.com" = "https://bing.com")
Success ✓	Handles port number appropriately	("https://bing.com:8080" = "https://bing.com:8080")

Helper Functions

1/15/2022 • 10 minutes to read • [Edit Online](#)

This topic contains a number of helper functions commonly used in M extensions. These functions may eventually be moved to the official M library, but for now can be copied into your extension file code. You shouldn't mark any of these functions as `shared` within your extension code.

Navigation Tables

Table.ToTable

This function adds the table type metadata needed for your extension to return a table value that Power Query can recognize as a Navigation Tree. See [Navigation Tables](#) for more information.

```
Table.ToTable = (
    table as table,
    keyColumns as list,
    nameColumn as text,
    dataColumn as text,
    itemKindColumn as text,
    itemNameColumn as text,
    isLeafColumn as text
) as table =>
    let
        tableType = Value.Type(table),
        newTableType = Type.AddTableKey(tableType, keyColumns, true) meta
        [
            NavigationTable.NameColumn = nameColumn,
            NavigationTable.DataColumn = dataColumn,
            NavigationTable.ItemKindColumn = itemKindColumn,
            Preview.DelayColumn = itemNameColumn,
            NavigationTable.IsLeafColumn = isLeafColumn
        ],
        navigationTable = Value.ReplaceType(table, newTableType)
    in
        navigationTable;
```

PARAMETER	DETAILS
table	Your navigation table.
keyColumns	List of column names that act as the primary key for your navigation table.
nameColumn	The name of the column that should be used as the display name in the navigator.
dataColumn	The name of the column that contains the Table or Function to display.
itemKindColumn	The name of the column to use to determine the type of icon to display. Valid values for the column are <code>Table</code> and <code>Function</code> .

PARAMETER	DETAILS
itemNameColumn	The name of the column to use to determine the type of tooltip to display. Valid values for the column are <code>Table</code> and <code>Function</code> .
isLeafColumn	The name of the column used to determine if this is a leaf node, or if the node can be expanded to contain another navigation table.

Example usage:

```
shared MyExtension.Contents = () =>
    let
        objects = #table(
            {"Name", "Key", "Data", "ItemKind", "ItemName",
            "IsLeaf"}, {
                {"Item1", "item1", #table({ "Column1"}, { "Item1"}), "Table", "Table", true},
                {"Item2", "item2", #table({ "Column1"}, { "Item2"}), "Table", "Table", true},
                {"Item3", "item3", FunctionCallThatReturnsATable(), "Table", "Table", true},
                {"MyFunction", "myfunction", AnotherFunction.Contents(), "Function", "Function", true}
            }),
        NavTable = Table.ToNavigationTable(objects, {"Key"}, "Name", "Data", "ItemKind", "ItemName",
        "IsLeaf")
    in
        NavTable;
```

URI Manipulation

Uri.FromParts

This function constructs a full URL based on individual fields in the record. It acts as the reverse of [Uri.Parts](#).

```
Uri.FromParts = (parts) =>
    let
        port = if (parts[Scheme] = "https" and parts[Port] = 443) or (parts[Scheme] = "http" and parts[Port]
        = 80) then "" else ":" & Text.From(parts[Port]),
        div1 = if Record.FieldCount(parts[Query]) > 0 then "?" else "",
        div2 = if Text.Length(parts[Fragment]) > 0 then "#" else "",
        uri = Text.Combine({parts[Scheme], "://", parts[Host], port, parts[Path], div1,
        Uri.BuildQueryString(parts[Query]), div2, parts[Fragment]}))
    in
        uri;
```

Uri.GetHost

This function returns the scheme, host, and default port (for HTTP/HTTPS) for a given URL. For example,

`https://bing.com/subpath/query?param=1¶m2=hello` would become `https://bing.com:443`.

This is particularly useful for building `ResourcePath`.

```
Uri.GetHost = (url) =>
    let
        parts = Uri.Parts(url),
        port = if (parts[Scheme] = "https" and parts[Port] = 443) or (parts[Scheme] = "http" and parts[Port]
        = 80) then "" else ":" & Text.From(parts[Port])
    in
        parts[Scheme] & "://" & parts[Host] & port;
```

ValidateUrlScheme

This function checks if the user entered an HTTPS URL and raises an error if they don't. This is required for user entered URLs for certified connectors.

```
ValidateUrlScheme = (url as text) as text => if (Uri.Parts(url)[Scheme] <> "https") then error "Url scheme must be HTTPS" else url;
```

To apply it, just wrap your `url` parameter in your data access function.

```
DataAccessFunction = (url as text) as table =>
let
    _url = ValidateUrlScheme(url),
    source = Web.Contents(_url)
in
    source;
```

Retrieving Data

Value.WaitFor

This function is useful when making an asynchronous HTTP request and you need to poll the server until the request is complete.

```
Value.WaitFor = (producer as function, interval as function, optional count as number) as any =>
let
    list = List.Generate(
        () => {0, null},
        (state) => state{0} <> null and (count = null or state{0} < count),
        (state) => if state{1} <> null then {null, state{1}} else {1 + state{0}, Function.InvokeAfter(() => producer(state{0}), interval(state{0}))},
        (state) => state{1})
in
    List.Last(list);
```

Table.GenerateByPage

This function is used when an API returns data in an incremental/paged format, which is common for many REST APIs. The `getNextPage` argument is a function that takes in a single parameter, which will be the result of the previous call to `getNextPage`, and should return a `nullable table`.

```
getNextPage = (lastPage) as nullable table => ...`
```

`getNextPage` is called repeatedly until it returns `null`. The function will collate all pages into a single table. When the result of the first call to `getNextPage` is null, an empty table is returned.

```

// The getNextPage function takes a single argument and is expected to return a nullable table
Table.GenerateByPage = (getNextPage as function) as table =>
    let
        listOfPages = List.Generate(
            () => getNextPage(null),           // get the first page of data
            (lastPage) => lastPage <> null,    // stop when the function returns null
            (lastPage) => getNextPage(lastPage) // pass the previous page to the next function call
        ),
        // concatenate the pages together
        tableOfPages = Table.FromList(listOfPages, Splitter.SplitByNothing(), {"Column1"}),
        firstRow = tableOfPages{0}?
    in
        // if we didn't get back any pages of data, return an empty table
        // otherwise set the table type based on the columns of the first page
        if (firstRow = null) then
            Table.FromRows({})
        else
            Value.ReplaceType(
                Table.ExpandTableColumn(tableOfPages, "Column1", Table.ColumnNames(firstRow[Column1])),
                Value.Type(firstRow[Column1])
            );

```

Additional notes:

- The `getNextPage` function will need to retrieve the next page URL (or page number, or whatever other values are used to implement the paging logic). This is generally done by adding `meta` values to the page before returning it.
- The columns and table type of the combined table (that is, all pages together) are derived from the first page of data. The `getNextPage` function should normalize each page of data.
- The first call to `getNextPage` receives a null parameter.
- `getNextPage` must return null when there are no pages left.

An example of using this function can be found in the [Github sample](#), and the [TripPin paging sample](#).

```

Github.PagedTable = (url as text) => Table.GenerateByPage((previous) =>
    let
        // If we have a previous page, get its Next link from metadata on the page.
        next = if (previous <> null) then Value.Metadata(previous)[Next] else null,
        // If we have a next link, use it, otherwise use the original URL that was passed in.
        urlToUse = if (next <> null) then next else url,
        // If we have a previous page, but don't have a next link, then we're done paging.
        // Otherwise retrieve the next page.
        current = if (previous <> null and next = null) then null else Github.Contents(urlToUse),
        // If we got data back from the current page, get the link for the next page
        link = if (current <> null) then Value.Metadata(current)[Next] else null
    in
        current meta [Next=link];

```

SchemaTransformTable

```

EnforceSchema.Strict = 1;           // Add any missing columns, remove extra columns, set table type
EnforceSchema.IgnoreExtraColumns = 2; // Add missing columns, do not remove extra columns
EnforceSchema.IgnoreMissingColumns = 3; // Do not add or remove columns

SchemaTransformTable = (table as table, schema as table, optional enforceSchema as number) as table =>
    let
        // Default to EnforceSchema.Strict
        _enforceSchema = if (enforceSchema <> null) then enforceSchema else EnforceSchema.Strict,

        // Applies type transforms to a given table
        EnforceTypes = (table as table, schema as table) as table =>
            let
                map = (t) => if Type.Is(t, type list) or Type.Is(t, type record) or t = type any then null
            else t,
                mapped = Table.TransformColumns(schema, {"Type", map}),
                omitted = Table.SelectRows(mapped, each [Type] <> null),
                existingColumns = Table.ColumnNames(table),
                removeMissing = Table.SelectRows(omitted, each List.Contains(existingColumns, [Name])),
                primitiveTransforms = Table.ToRows(removeMissing),
                changedPrimitives = Table.TransformColumnTypes(table, primitiveTransforms)
            in
                changedPrimitives,

            // Returns the table type for a given schema
            SchemaToTableType = (schema as table) as type =>
                let
                    toList = List.Transform(schema[Type], (t) => [Type=t, Optional=false]),
                    toRecord = Record.FromList(toList, schema[Name]),
                    toType = Type.ForRecord(toRecord, false)
                in
                    type table (toType),

            // Determine if we have extra/missing columns.
            // The enforceSchema parameter determines what we do about them.
            schemaNames = schema[Name],
            foundNames = Table.ColumnNames(table),
            addNames = List.RemoveItems(schemaNames, foundNames),
            extraNames = List.RemoveItems(foundNames, schemaNames),
            tmp = Text.NewGuid(),
            added = Table.AddColumn(table, tmp, each []),
            expanded = Table.ExpandRecordColumn(added, tmp, addNames),
            result = if List.IsEmpty(addNames) then table else expanded,
            fullList =
                if (_enforceSchema = EnforceSchema.Strict) then
                    schemaNames
                else if (_enforceSchema = EnforceSchema.IgnoreMissingColumns) then
                    foundNames
                else
                    schemaNames & extraNames,

            // Select the final list of columns.
            // These will be ordered according to the schema table.
            reordered = Table.SelectColumns(result, fullList, MissingField.Ignore),
            enforcedTypes = EnforceTypes(reordered, schema),
            withType = if (_enforceSchema = EnforceSchema.Strict) then Value.ReplaceType(enforcedTypes,
SchemaToTableType(schema)) else enforcedTypes
        in
            withType;

```

Table.ChangeType

```

let
    // table should be an actual Table.Type, or a List.Type of Records
    Table.ChangeType = (table, tableType as type) as nullable table =>
        // we only operate on table types
        if (not Type.Is(tableType, type table)) then error "type argument should be a table type" else

```

```

// if we have a null value, just return it
if (table = null) then table else
let
    columnsForType = Type.RecordFields(Type.TableRow(tableType)),
    columnsAsTable = Record.ToTable(columnsForType),
    schema = Table.ExpandRecordColumn(columnsAsTable, "Value", {"Type"}, {"Type"}),
    previousMeta = Value.Metadata(tableType),

    // make sure we have a table
    parameterType = Value.Type(table),
    _table =
        if (Type.Is(parameterType, type table)) then table
        else if (Type.Is(parameterType, type list)) then
            let
                asTable = Table.FromList(table, Splitter.SplitByNothing(), {"Column1"}),
                firstValueType = Value.Type(Table.FirstValue(asTable, null)),
                result =
                    // if the member is a record (as expected), then expand it.
                    if (Type.Is(firstValueType, type record)) then
                        Table.ExpandRecordColumn(asTable, "Column1", schema[Name])
                    else
                        error Error.Record("Error.Parameter", "table argument is a list, but not a
list of records", [ValueType = firstValueType ])
            in
                if (List.IsEmpty(table)) then
                    #table({"a"}, {})
                else result
        else
            error Error.Record("Error.Parameter", "table argument should be a table or list of
records", [ValueType = parameterType]),

    reordered = Table.SelectColumns(_table, schema[Name], MissingField.UseNull),

    // process primitive values - this will call Table.TransformColumnTypes
    map = (t) => if Type.Is(t, type table) or Type.Is(t, type list) or Type.Is(t, type record) or t
= type any then null else t,
    mapped = Table.TransformColumns(schema, {"Type", map}),
    omitted = Table.SelectRows(mapped, each [Type] <> null),
    existingColumns = Table.ColumnNames(reordered),
    removeMissing = Table.SelectRows(omitted, each List.Contains(existingColumns, [Name])),
    primativeTransforms = Table.ToRows(removeMissing),
    changedPrimatives = Table.TransformColumnTypes(reordered, primativeTransforms),

    // Get the list of transforms we'll use for Record types
    recordColumns = Table.SelectRows(schema, each Type.Is([Type], type record)),
    recordTypeTransformations = Table.AddColumn(recordColumns, "RecordTransformations", each (r) =>
Record.ChangeType(r, [Type]), type function),
    recordChanges = Table.ToRows(Table.SelectColumns(recordTypeTransformations, {"Name",
"RecordTransformations"})),

    // Get the list of transforms we'll use for List types
    listColumns = Table.SelectRows(schema, each Type.Is([Type], type list)),
    listTransforms = Table.AddColumn(listColumns, "ListTransformations", each (t) =>
List.ChangeType(t, [Type]), Function.Type),
    listChanges = Table.ToRows(Table.SelectColumns(listTransforms, {"Name",
"ListTransformations"})),

    // Get the list of transforms we'll use for Table types
    tableColumns = Table.SelectRows(schema, each Type.Is([Type], type table)),
    tableTransforms = Table.AddColumn(tableColumns, "TableTransformations", each (t) =>
@Table.ChangeType(t, [Type]), Function.Type),
    tableChanges = Table.ToRows(Table.SelectColumns(tableTransforms, {"Name",
"TableTransformations"})),

    // Perform all of our transformations
    allColumnTransforms = recordChanges & listChanges & tableChanges,
    changedRecordTypes = if (List.IsEmpty(allColumnTransforms)) then changedPrimatives else
Table.TransformColumns(changedPrimatives, allColumnTransforms, null, MissingField.Ignore),

```

```

        // set final type
        withType = Value.ReplaceType(changedRecordTypes, tableType)
    in
        if (List.IsEmpty(Record.FieldNames(columnsForType))) then table else withType meta previousMeta,
    end

    // If given a generic record type (no predefined fields), the original record is returned
    Record.ChangeType = (record as record, recordType as type) =>
        let
            // record field format is [ fieldName = [ Type = type, Optional = logical], ... ]
            fields = try Type.RecordFields(recordType) otherwise error "Record.ChangeType: failed to get
    record fields. Is this a record type?",
            fieldNames = Record.FieldNames(fields),
            fieldTable = Record.ToTable(fields),
            optionalFields = Table.SelectRows(fieldTable, each [Value][Optional])[Name],
            requiredFields = List.Difference(fieldNames, optionalFields),
            // make sure all required fields exist
            withRequired = Record.SelectFields(record, requiredFields, MissingField.UseNull),
            // append optional fields
            withOptional = withRequired & Record.SelectFields(record, optionalFields, MissingField.Ignore),
            // set types
            transforms = GetTransformsForType(recordType),
            withTypes = Record.TransformFields(withOptional, transforms, MissingField.Ignore),
            // order the same as the record type
            reorder = Record.ReorderFields(withTypes, fieldNames, MissingField.Ignore)
        in
            if (List.IsEmpty(fieldNames)) then record else reorder,
        end

    List.ChangeType = (list as list, listType as type) =>
        if (not Type.Is(listType, type list)) then error "type argument should be a list type" else
        let
            listItemType = Type.ListItem(listType),
            transform = GetTransformByType(listItemType),
            modifiedValues = List.Transform(list, transform),
            typed = Value.ReplaceType(modifiedValues, listType)
        in
            typed,
        end

    // Returns a table type for the provided schema table
    Schema.ToTableType = (schema as table) as type =>
        let
            toList = List.Transform(schema[Type], (t) => [Type=t, Optional=false]),
            toRecord = Record.FromList(toList, schema[Name]),
            toType = Type.ForRecord(toRecord, false),
            previousMeta = Value.Metadata(schema)
        in
            type table (toType) meta previousMeta,
        end

    // Returns a list of transformations that can be passed to Table.TransformColumns, or
    Record.TransformFields
    // Format: {"Column", (f) => ...} .... ex: {"A", Number.From}
    GetTransformsForType = (_type as type) as list =>
        let
            fieldsOrColumns = if (Type.Is(_type, type record)) then Type.RecordFields(_type)
                            else if (Type.Is(_type, type table)) then
                                Type.RecordFields(Type.TableRow(_type))
                            else error "GetTransformsForType: record or table type expected",
            toTable = Record.ToTable(fieldsOrColumns),
            transformColumn = Table.AddColumn(toTable, "Transform", each GetTransformByType([Value][Type]),
Function.Type),
            transformMap = Table.ToRows(Table.SelectColumns(transformColumn, {"Name", "Transform"}))
        in
            transformMap,
        end

    GetTransformByType = (_type as type) as function =>
        if (Type.Is(_type, type number)) then Number.From
        else if (Type.Is(_type, type text)) then Text.From
        else if (Type.Is(_type, type date)) then Date.From
        else if (Type.Is(_type, type datetime)) then DateTime.From
        else if (Type.Is(_type, type duration)) then Duration.From
    end

```

```
else if (Type.Is(_type, type datetimezone)) then DateTimeZone.From
else if (Type.Is(_type, type logical)) then Logical.From
else if (Type.Is(_type, type time)) then Time.From
else if (Type.Is(_type, type record)) then (t) => if (t <> null) then @Record.ChangeType(t, _type)
else t
else if (Type.Is(_type, type table)) then (t) => if (t <> null) then @Table.ChangeType(t, _type)
else t
else if (Type.Is(_type, type list)) then (t) => if (t <> null) then @List.ChangeType(t, _type) else
t
else (t) => t
in
Table.ChangeType
```

Handling errors

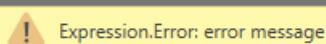
1/15/2022 • 2 minutes to read • [Edit Online](#)

Errors in Power Query generally halt query evaluation and display a message to the user.

Throwing an error with the `error` expression

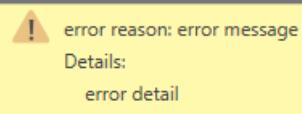
Throwing an error in Power Query is as simple as invoking the `error` expression.

```
let
    Source = "foo",
    Output = error "error message"
in
    Output
```



The `Error.Record` function can be used for more control.

```
let
    Source = "foo",
    Output = error Error.Record("error reason", "error message", "error detail")
in
    Output
```



Catching an error with `try` and `otherwise`

The `try` expression converts values and errors into a record value that indicates whether the `try` expression handled an error or not, as well as the proper value of the error record.

If no error is found, the following record is returned from the `try` expression:

```
try "foo"
```

HasError	FALSE
Value	foo

If an error is found, the following record is returned from the `try` expression:

```
try "foo"+1
```

HasError	TRUE
Error	Record

The Error record contains **Reason**, **Message**, and **Detail** fields.

Reason	Expression.Error
Message	We cannot apply operator + to types Text and Number.
Detail	Record

Depending on the error, the **Detail** field may contain additional information.

The `otherwise` clause can be used with a `try` expression to perform some action if an error occurs:

```
try "foo"+1 otherwise "There was an error"
```

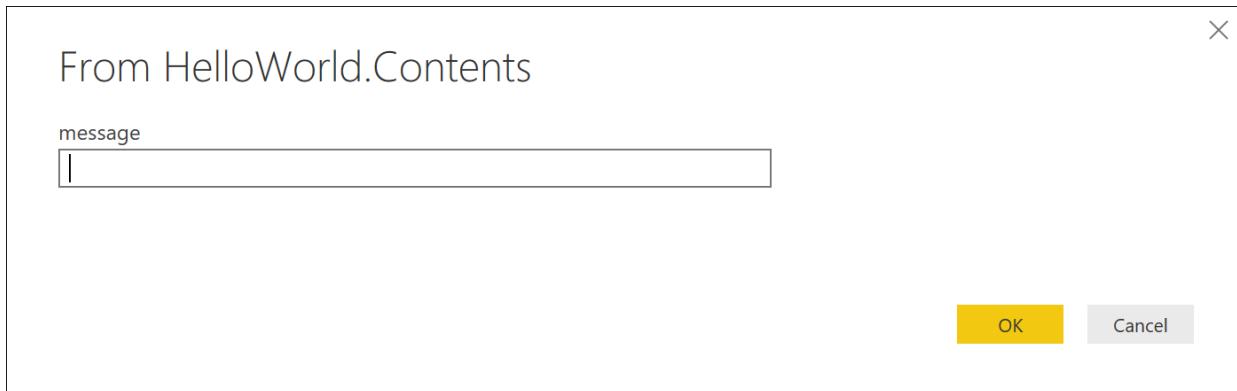
is a more compact form of:

```
result = try "foo"+1,  
if result[HasError] then "There was an error" else result[Value]
```

Adding Function Documentation

1/15/2022 • 3 minutes to read • [Edit Online](#)

Power Query will automatically generate an invocation UI for you based on the arguments for your function. By default, this UI will contain the name of your function, and an input for each of your parameters.



Similarly, evaluating the name of your function, without specifying parameters, will display information about it.



You might notice that built-in functions typically provide a better user experience, with descriptions, tooltips, and even sample values. You can take advantage of this same mechanism by defining specific meta values on your function type. This topic describes the meta fields that are used by Power Query, and how you can make use of them in your extensions.

The screenshot shows the Power Query Editor interface. On the left, there's a sidebar titled "Queries [1]" with a single item "Query1". The main area has a title bar with icons for close, save, and refresh, followed by the text "= Csv.Document". Below this is the function documentation for "Csv.Document". It describes the function as returning the contents of a CSV document as a table. It details parameters: "source" (optional), "columns" (optional), "delimiter" (optional), "extraValues" (optional) with an example of "123", and "encoding" (optional). At the bottom, it shows the M language signature: `function (source as any, optional columns as any, optional delimiter as any, optional extraValues as nullable number, optional encoding as nullable TextEncoding.Type) as table`. There are "Invoke" and "Clear" buttons at the bottom of the parameter input area.

Function Types

You can provide documentation for your function by defining custom *type* values. The process looks like this:

1. Define a type for each parameter.
2. Define a type for your function.
3. Add various `Documentation.*` fields to your types metadata record.
4. Call `Value.ReplaceType` to ascribe the type to your shared function.

You can find more information about types and metadata values in the [M Language Specification](#).

Using this approach allows you to supply descriptions and display names for your function, as well as individual parameters. You can also supply sample values for parameters, as well as defining a preset list of values (turning the default text box control into a drop down).

The Power Query experience retrieves documentation from meta values on the type of your function, using a combination of calls to `Value.Type`, `Type.FunctionParameters`, and `Value.Metadata`.

Function Documentation

The following table lists the Documentation fields that can be set in the metadata for your *function*. All fields are optional.

FIELD	TYPE	DETAILS
-------	------	---------

FIELD	TYPE	DETAILS
Documentation.Examples	list	List of record objects with example usage of the function. Only displayed as part of the function info. Each record should contain the following optional text fields: <code>Description</code> , <code>Code</code> , and <code>Result</code> .
Documentation.LongDescription	text	Full description of what the function does, displayed in the function info.
Documentation.Name	text	Text to display across the top of the function invocation dialog.

Parameter Documentation

The following table lists the Documentation fields that can be set in the metadata for your *function parameters*. All fields are optional.

FIELD	TYPE	DETAILS
Documentation.AllowedValues	list	List of valid values for this parameter. Providing this field will change the input from a textbox to a drop down list. Note, this doesn't prevent a user from manually editing the query to supply alternative values.
Documentation.FieldCaption	text	Friendly display name to use for the parameter.
Documentation.FieldDescription	text	Description to show next to the display name.
Documentation.SampleValues	list	List of sample values to be displayed (as faded text) inside of the text box.
Formatting.IsMultiLine	boolean	Allows you to create a multi-line input, for example for pasting in native queries.
Formatting.IsCode	boolean	Formats the input field for code, commonly with multi-line inputs. Uses a code-like font rather than the standard font.

Basic Example

The following code snippet (and resulting dialogs) are from the [HelloWorldWithDocs](#) sample.

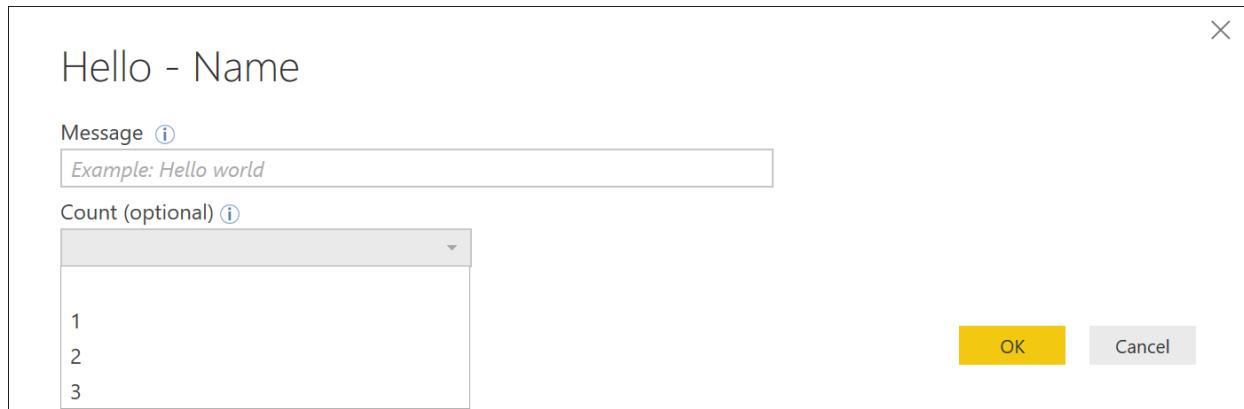
```
[DataSource.Kind="HelloWorldWithDocs", Publish="HelloWorldWithDocs.Publish"]
shared HelloWorldWithDocs.Contents = Value.ReplaceType(HelloWorldImpl, HelloWorldType);

HelloWorldType = type function (
    message as (type text meta [
        Documentation.FieldCaption = "Message",
        Documentation.FieldDescription = "Text to display",
        Documentation.SampleValues = {"Hello world", "Hola mundo"}
    ]),
    optional count as (type number meta [
        Documentation.FieldCaption = "Count",
        Documentation.FieldDescription = "Number of times to repeat the message",
        Documentation.AllowedValues = { 1, 2, 3 }
    )));
as table meta [
    Documentation.Name = "Hello - Name",
    Documentation.LongDescription = "Hello - Long Description",
    Documentation.Examples = {[[
        Description = "Returns a table with 'Hello world' repeated 2 times",
        Code = "HelloWorldWithDocs.Contents(""Hello world"", 2)",
        Result = "#table({""Column1""}, {""Hello world""}, {""Hello world""})"
    ],[
        Description = "Another example, new message, new count!",
        Code = "HelloWorldWithDocs.Contents(""Goodbye"", 1)",
        Result = "#table({""Column1""}, {""Goodbye""})"
    ]}]
];
];

HelloWorldImpl = (message as text, optional count as number) as table =>
let
    _count = if (count <> null) then count else 5,
    listOfMessages = List.Repeat({message}, _count),
    table = Table.FromList(listOfMessages, Splitter.SplitByNothing())
in
    table;
```

This code results in the following dialogs in Power BI.

Function invocation



Function info

The screenshot shows the Power BI formula editor interface. The title bar says "HelloWorldWithDocs.Contents". The main area contains the following content:

Enter Parameters

Message
Example: Hello world

Count (optional)

Invoke Clear

```
function (message as text, optional count as nullable number) as table
```

Example: Returns a table with 'Hello world' repeated 2 times

Usage:
HelloWorldWithDocs.Contents("Hello world", 2)

Output:
#table({ "Column1" }, { {"Hello world"}, {"Hello world"} })

Example: Another example, new message, new count!

Usage:
HelloWorldWithDocs.Contents("Goodbye", 1)

Output:
#table({ "Column1" }, { {"Goodbye"} })

Multi-Line Example

```
[DataSource.Kind="HelloWorld", Publish="HelloWorld.Publish"]
shared HelloWorld.Contents =
    let
        HelloWorldType = type function (
            message1 as (type text meta [
                Documentation.FieldCaption = "Message 1",
                Documentation.FieldDescription = "Text to display for message 1",
                Documentation.SampleValues = {"Hello world"},
                Formatting.IsMultiLine = true,
                Formatting.IsCode = true
            ]),
            message2 as (type text meta [
                Documentation.FieldCaption = "Message 2",
                Documentation.FieldDescription = "Text to display for message 2",
                Documentation.SampleValues = {"Hola mundo"},
                Formatting.IsMultiLine = true,
                Formatting.IsCode = false
            )) as text,
            HelloWorldFunction = (message1 as text, message2 as text) as text => message1 & message2
        in
            Value.ReplaceType(HelloWorldFunction, HelloWorldType);
```

This code (with associated publish information, etc.) results in the following dialogue in Power BI. New lines will be represented in text with '#(If)', or 'line feed'.

Enter Parameters

Message 1 ⓘ

Example: Hello world

Message 2 ⓘ

Example: Hola mundo

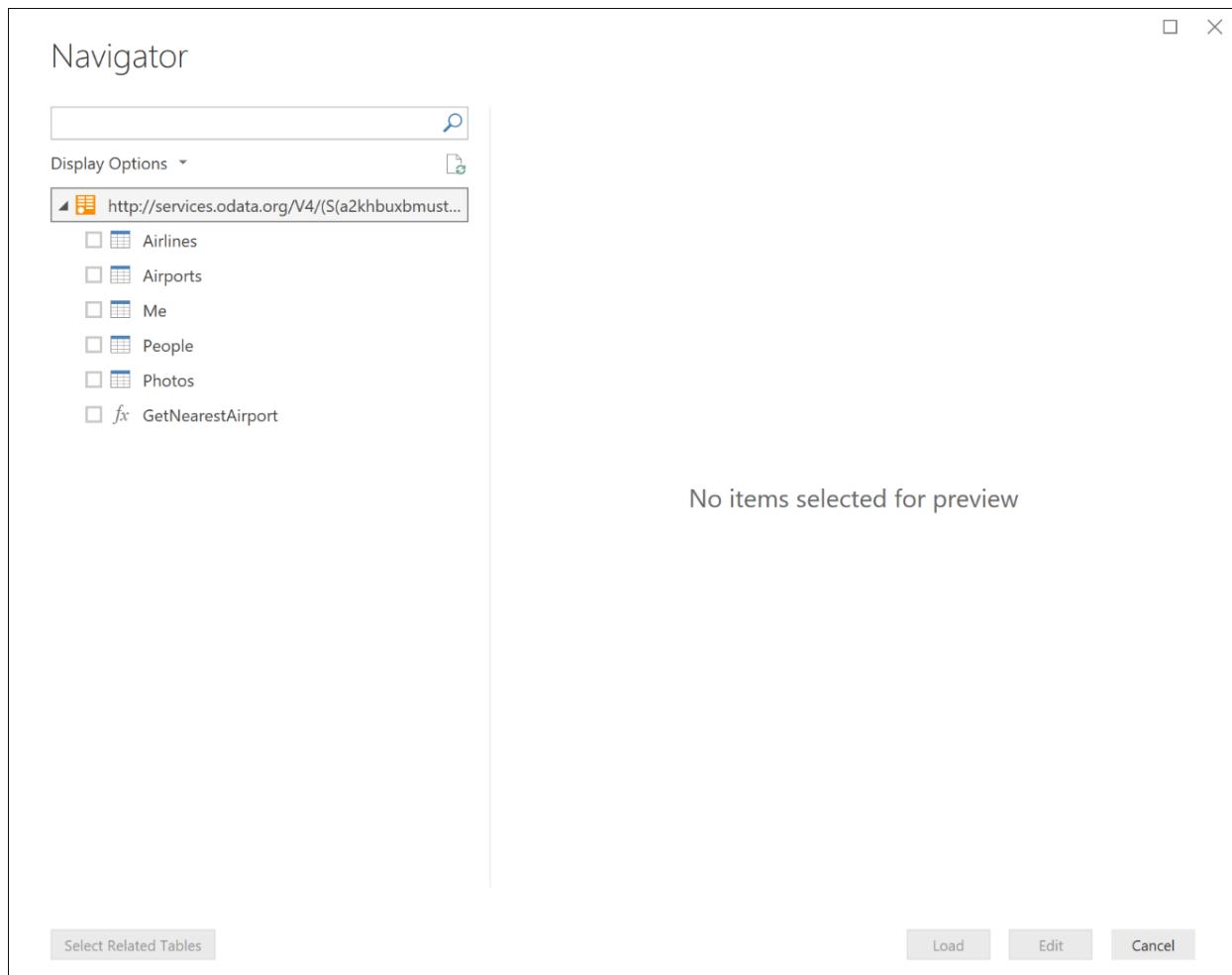
OK

Cancel

Handling Navigation

1/15/2022 • 3 minutes to read • [Edit Online](#)

Navigation Tables (or nav tables) are a core part of providing a user-friendly experience for your connector. The Power Query experience displays them to the user after they've entered any required parameters for your data source function, and have authenticated with the data source.



Behind the scenes, a nav table is just a regular M Table value with specific metadata fields defined on its Type. When your data source function returns a table with these fields defined, Power Query will display the navigator dialog. You can actually see the underlying data as a Table value by right-clicking on the root node and selecting Edit.

Table.ToNavigationTable

You can use the `Table.ToNavigationTable` function to add the table type metadata needed to create a nav table.

NOTE

You currently need to copy and paste this function into your M extension. In the future it will likely be moved into the M standard library.

The following table describes the parameters for this function:

PARAMETER	DETAILS
table	Your navigation table.
keyColumns	List of column names that act as the primary key for your navigation table.
nameColumn	The name of the column that should be used as the display name in the navigator.
dataColumn	The name of the column that contains the Table or Function to display.
itemKindColumn	The name of the column to use to determine the type of icon to display. See below for the list of valid values for the column.
itemNameColumn	The name of the column to use to determine the preview behavior. This is typically set to the same value as itemKind.
isLeafColumn	The name of the column used to determine if this is a leaf node, or if the node can be expanded to contain another navigation table.

The function adds the following metadata to the table type:

FIELD	PARAMETER
NavigationTable.NameColumn	nameColumn
NavigationTable.DataColumn	dataColumn
NavigationTable.ItemKindColumn	itemKindColumn
NavigationTable.IsLeafColumn	isLeafColumn
Preview.DelayColumn	itemNameColumn

Values for ItemKind

Each of the following item kind values provide a different icon in the navigation table.

- Feed
- Cube
- CubeDatabase
- CubeView
- CubeViewFolder
- Database
- DatabaseServer
- Dimension
- Table
- Folder
- Function

- View
- Sheet
- Subcube
- DefinedName
- Record

The image below shows the icons for item kinds in Power BI Desktop.

Navigator



Display Options ▾



◀ NavTable Icons [16]

- ▷ CubeViewFolder
- ▷ Folder
- ▷ View
- ▷ Table
- ▷ Sheet
- ▷ Record
- ▷ Function
- ▷ Feed
- ▷ Dimension
- ▷ DefinedName
- ▷ DatabaseServer
- ▷ Database
- ▷ CubeView
- ▷ CubeDatabase
- ▷ Cube
- ▷ Subcube

Examples

Flat navigation table

The following code sample displays a flat nav table with three tables and a function.

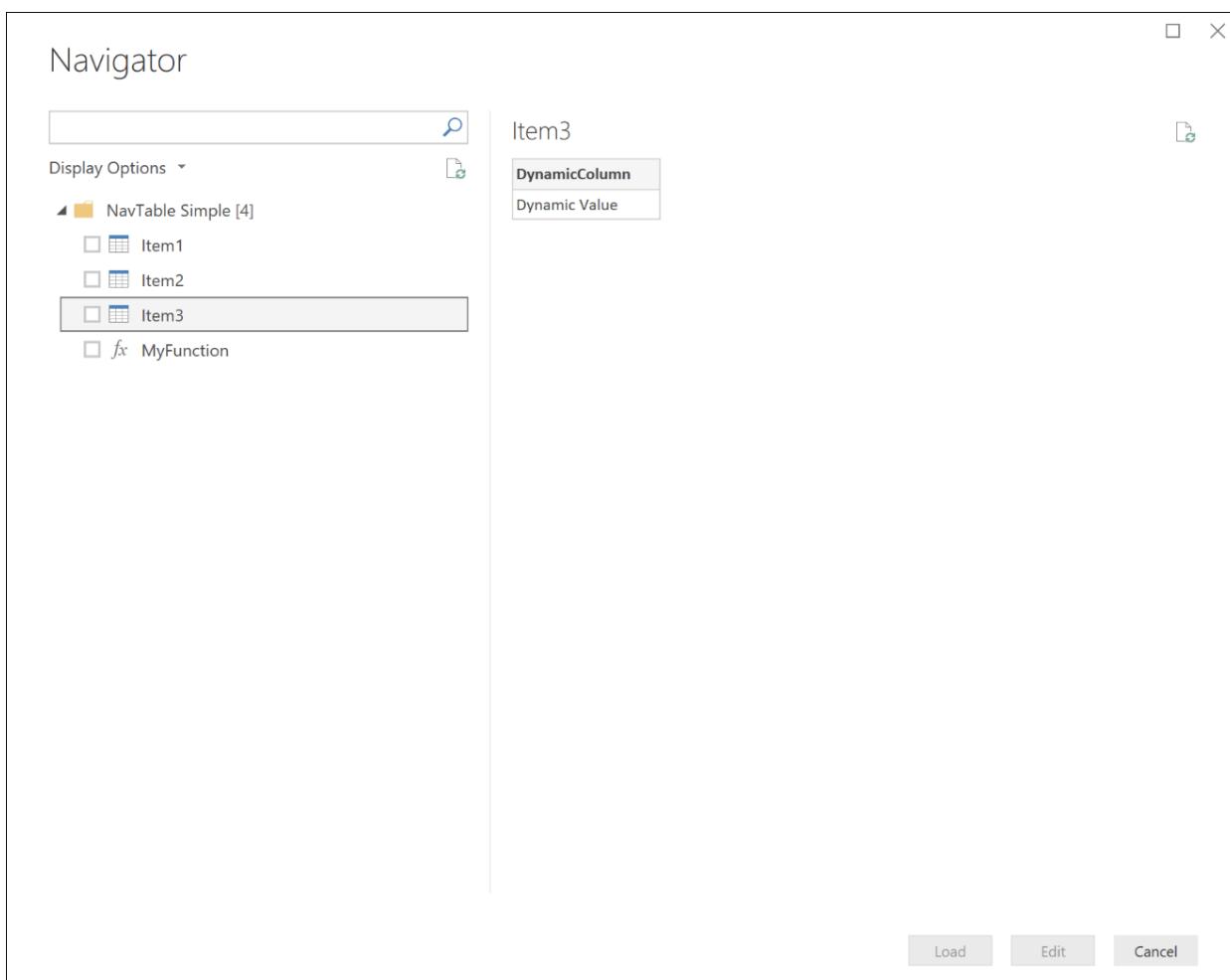
```

shared NavigationTable.Simple = () =>
    let
        objects = #table(
            {"Name", "Key", "Data", "ItemKind", "ItemName", "IsLeaf"}, {
                {"Item1", "item1", #table({ "Column1"}, { "Item1"}), "Table", "Table", true},
                {"Item2", "item2", #table({ "Column1"}, { "Item2"}), "Table", "Table", true},
                {"Item3", "item3", FunctionCallThatReturnsATable(), "Table", "Table", true},
                {"MyFunction", "myfunction", AnotherFunction.Contents, "Function", "Function", true}
            })
        NavTable = Table.ToNavigationTable(objects, {"Key"}, "Name", "Data", "ItemKind", "ItemName", "IsLeaf")
    in
        NavTable;

shared FunctionCallThatReturnsATable = () =>
    #table({ "DynamicColumn"}, { "Dynamic Value"});

```

This code will result in the following Navigator display in Power BI Desktop:



Multi-level navigation table

It is possible to use nested navigation tables to create a hierarchical view over your data set. You do this by setting the `IsLeaf` value for that row to `false` (which marks it as a node that can be expanded), and format the `Data` column to also be another nav table.

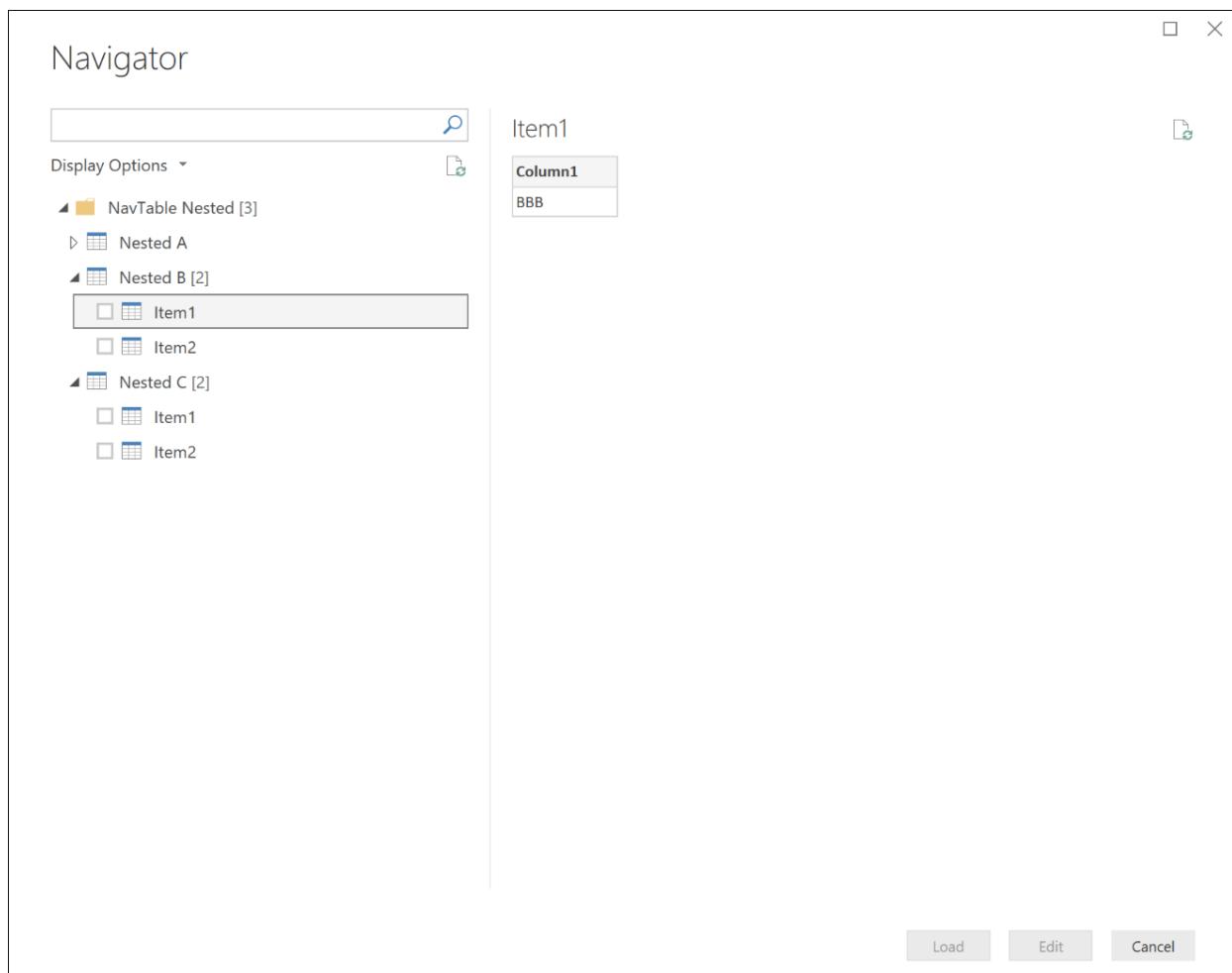
```

shared NavigationTable.Nested = () as table =>
let
    objects = #table(
        {"Name", "Key", "Data", "ItemKind", "ItemName", "IsLeaf"},{
            {"Nested A", "n1", CreateNavTable("AAA"), "Table", "Table", false},
            {"Nested B", "n2", CreateNavTable("BBB"), "Table", "Table", false},
            {"Nested C", "n3", CreateNavTable("CCC"), "Table", "Table", false}
        })
    NavTable = Table.ToNavigationTable(objects, {"Key"}, "Name", "Data", "ItemKind", "ItemName",
    "IsLeaf")
in
    NavTable;

CreateNavTable = (message as text) as table =>
let
    objects = #table(
        {"Name", "Key", "Data", "ItemKind", "ItemName", "IsLeaf"},{
            {"Item1", "item1", #table({"Column1"}, {{message}}), "Table", "Table", true},
            {"Item2", "item2", #table({"Column1"}, {{message}}), "Table", "Table", true}
        })
    NavTable = Table.ToNavigationTable(objects, {"Key"}, "Name", "Data", "ItemKind", "ItemName",
    "IsLeaf")
in
    NavTable;

```

This code would result in the following Navigator display in Power BI Desktop:



Dynamic Navigation Tables

More complex functionality can be built from these basics. While all of the above examples show hard-coded entities in the nav table, it's easy to see how a nav table could be generated dynamically based on entities that are available to a given user. A few key considerations for dynamic navigation tables include:

- [Error handling](#) to ensure a good experience for users that don't have access to certain endpoints.
- Node evaluation is lazy by default; leaf nodes are not evaluated until the parent node is expanded. Certain implementations of multi-level dynamic nav tables may result in eager evaluation of the entire tree. Be sure to monitor the number of calls that Power Query is making as it initially renders the navigation table. For example, `Table.InsertRows` is 'lazier' than `Table.FromRecords`, as it does not need to evaluate its arguments.

Handling Gateway Support

1/15/2022 • 2 minutes to read • [Edit Online](#)

Test Connection

Custom Connector support is available in both Personal and Standard modes of the [on-premises data gateway](#). Both gateway modes support **Import**. **Direct Query** is only supported in Standard mode. OAuth for custom connectors via gateways is currently supported only for gateway admins but not other data source users.

The method for implementing `TestConnection` functionality is likely to change while the Power BI Custom Data Connector functionality is in preview.

To support scheduled refresh through the on-premises data gateway, your connector **must** implement a `TestConnection` handler. The function is called when the user is configuring credentials for your source, and used to ensure they are valid. The `TestConnection` handler is set in the [Data Source Kind](#) record, and has the following signature:

```
(dataSourcePath) as list => ...
```

Where `dataSourcePath` is the [Data Source Path](#) value for your function, and the return value is a list composed of:

- The name of the function to call (this function must be marked as `#shared`, and is usually your primary data source function).
- One or more arguments to pass to your function.

If the invocation of the function results in an error, `TestConnection` is considered to have failed, and the credential won't be persisted.

NOTE

As stated above, the function name provided by `TestConnection` must be a `shared` member.

Example: Connector with no required arguments

The code snippet below implements `TestConnection` for a data source with no required parameters (such as the one found in the [TripPin tutorial](#)). Connectors with no required parameters (referred to as 'Singletons') do not need any user provided input to test a connection (other than credentials). In this case, the `dataSourcePath` value would be equal to the name of the Data Source Kind, and can be ignored. The `TripPin.Contents` function is invoked with no additional parameters.

```
TripPin = [
    TestConnection = (dataSourcePath) => { "TripPin.Contents" },
    Authentication = [
        Anonymous = []
    ],
    Label = "TripPin"
];
```

Example: Connector with a URL parameter

If your data source function has a single required parameter of the type `Uri.Type`, its `dataSourcePath` will be equal to the URL provided by the user. The snippet below shows the `TestConnection` implementation from the [Github Sample](#).

```
GithubSample = [
    TestConnection = (dataSourcePath) => {"GithubSample.Contents", dataSourcePath},
    Authentication = [
        OAuth = [
            StartLogin = StartLogin,
            FinishLogin = FinishLogin,
            Label = Extension.LoadString("AuthenticationLabel")
        ]
    ],
];
];
```

Example: Connector with required parameters

If your data source function has multiple parameters, or a single non-URL parameter, then the `dataSourcePath` value will be a JSON string containing the parameters. The snippet below comes from the [DirectQueryForSQL](#) sample.

```
DirectSQL = [
    TestConnection = (dataSourcePath) =>
        let
            json = Json.Document(dataSourcePath),
            server = json[server],
            database = json[database]
        in
            { "DirectSQL.Database", server, database },
    Authentication = [
        Windows = [],
        UsernamePassword = []
    ],
    Label = "Direct Query for SQL"
];
];
```

Handling Power Query Connector Signing

1/15/2022 • 3 minutes to read • [Edit Online](#)

In Power BI, the loading of custom connectors is limited by your choice of security setting. As a general rule, when the security for loading custom connectors is set to 'Recommended', the custom connectors won't load at all, and you have to lower it to make them load.

The exception to this is trusted, 'signed connectors'. Signed connectors are a special format of custom connector, a .pqx instead of .mez file, which has been signed with a certificate. The signer can provide the user or the user's IT department with a thumbprint of the signature, which can be put into the registry to securely indicate trusting a given connector.

The following steps enable you to use a certificate (with an explanation on how to generate one if you don't have one available) and sign a custom connector with the 'MakePQX' tool.

NOTE

If you need help creating a self-signed certificate to test these instructions, go to the Microsoft documentation on [New-SelfSignedCertificate in PowerShell](#).

NOTE

If you need help exporting your certificate as a pfx, go to [Export-PfxCertificate](#).

1. Download [MakePQX](#).
2. Extract the MakePQX folder in the included zip to the target you want.
3. To run it, call MakePQX in the command line. It requires the other libraries in the folder, so you can't copy just the one executable. Running without any parameters will return the help information.

Usage: **MakePQX [options] [command]**

Options:

OPTIONS	DESCRIPTION
-? -h --help	Show help information

Commands:

COMMAND	DESCRIPTION
pack	Create a pqx file.
sign	Signs an unsigned pqx, or countersigns if pqx is already signed. Use the --replace option to replace the existing signature.
verify	Verify the signature status on a pqx file. Return value will be non-zero if the signature is invalid.

There are three commands in MakePQX. Use **MakePQX [command] --help** for more information about a command.

Pack

The **Pack** command takes a mez file and packs it into a ppx file, which can be signed. The ppx file is also able to support some capabilities that will be added in the future.

Usage: **MakePQX pack [options]**

Options:

OPTION	DESCRIPTION
-? -h --help	Show help information.
-mz --mez	Input extension file.
-c --certificate	Certificate (.pfx) used to sign the extension file.
-p --password	Password for the certificate file.
-t --target	Output file name. Defaults to the same name as the input file.

Example

```
C:\Users\cpoke\Downloads\MakePQX>MakePQX.exe pack -mz  
"C:\Users\cpoke\OneDrive\Documents\Power BI Desktop\Custom Connectors\HelloWorld.mez" -t  
"C:\Users\cpoke\OneDrive\Documents\Power BI Desktop\Custom  
Connectors\HelloWorldSigned.pqx"
```

Sign

The **Sign** command signs your ppx file with a certificate, giving it a thumbprint that can be checked for trust by Power BI clients with the higher security setting. This command takes a ppx file and returns the same ppx file, signed.

Usage: **MakePQX sign [arguments] [options]**

Arguments:

ARGUMENT	DESCRIPTION
<ppx file>	The path to the ppx file.

Options:

OPTION	DESCRIPTION
-c --certificate	Certificate (.pfx) used to sign the extension file.
-p --password	Password for the certificate file.
-r --replace	Replace existing signature instead of countersigning.

OPTION	DESCRIPTION
-? -h --help	Show help information.

Example

```
C:\Users\cpope\Downloads\MakePQX> MakePQX sign
"C:\Users\cpope\OneDrive\Documents\Power BI Desktop\Custom
Connectors\HelloWorldSigned.pqx" --certificate ContosoTestCertificate.pfx --password password
```

Verify

The **Verify** command verifies that your module has been properly signed, and is showing the Certificate status.

Usage: **MakePQX verify [arguments] [options]**

Arguments:

ARGUMENT	DESCRIPTION
<pqx file>	The path to the pqx file.

Options:

OPTION	DESCRIPTION
-q --quiet	Hides signature verification output.
-? -h --help	Show help information.

Example

```
C:\Users\cpope\Downloads\MakePQX> MakePQX verify
"C:\Users\cpope\OneDrive\Documents\Power BI Desktop\Custom
Connectors\HelloWorldSigned.pqx"
```

```
{
  "SignatureStatus": "Success",
  "CertificateStatus": [
    {
      "Issuer": "CN=Colin Popell",
      "Thumbprint": "16AF59E4BE5384CD860E230ED4AED474C2A3BC69",
      "Subject": "CN=Colin Popell",
      "NotBefore": "2019-02-14T22:47:42-08:00",
      "NotAfter": "2020-02-14T23:07:42-08:00",
      "Valid": false,
      "Parent": null,
      "Status": "UntrustedRoot"
    }
  ]
}
```

Trusting signed connectors in Power BI Desktop

Once you've verified your signature, you can provide the thumbprint to the end user to list as trusted. You can read about how to provide the thumbprint in the [Power BI Documentation](#).

Power Query Connector Certification

1/15/2022 • 7 minutes to read • [Edit Online](#)

NOTE

This article describes the requirements and process to submit a Power Query custom connector for certification. Read the entire article closely before starting the certification process.

Introduction

Certifying a Power Query custom connector makes the connector available publicly, out-of-box, within Power BI Desktop. Certified connectors are supported in PowerBI.com and all versions of Power BI Premium, except dataflows. Certification is governed by Microsoft's Connector Certification Program, where Microsoft works with partner developers to extend the data connectivity capabilities of Power BI.

Certified connectors are:

- Maintained by the partner developer
- Supported by the partner developer
- Certified by Microsoft
- Distributed by Microsoft

We work with partners to try to make sure that they have support in maintenance, but customer issues with the connector itself will be directed to the partner developer.

Certified connectors are bundled out-of-box in Power BI Desktop. Custom connectors need to be loaded in Power BI Desktop, as described in [Loading your extension in Power BI Desktop](#). Both can be refreshed through Power BI Desktop or Power BI Service through using an on-premises data gateway by implementing a [TestConnection](#).

Certified connectors with a `TestConnection` implementation also support end-to-end refresh through the cloud (Power BI Service) without the need of an on-premises data gateway. The Power BI service environment essentially hosts a "cloud gateway" that runs similar to the on-premises gateway. After certification, we will deploy your connector to this environment so that it's available to all Power BI customers. There are additional requirements for connectors that need to use additional components, such as an ODBC-based driver. Be sure to reach out to your Microsoft contact if your connector requires the use of additional components.

Custom Connector Security and Signing

As M is a versatile language that, as seen in [Handling Authentication](#), has the capacity to interact with stored credentials, we need to give users a way to only allow trusted connectors to run.

From a developer's perspective, developers need to [self-sign](#) their custom connector and provide their users with the information (thumbprint) to securely load it.

From a user's perspective, users need to use the thumbprint from the developer to securely [trust and load the custom connector](#) for use. Alternatively, users can opt to lower their security settings to allow loading of code not certified by Microsoft or another developer, but this is not recommended.

Certification Overview

Prerequisites

To ensure the best experience for our customers, we only consider connectors that meet a set of prerequisites for certification:

- The connector must be for a public product.
- The connector must be considered code-complete for a initial release version. The program allows for frequent iterations and updates. Note that Microsoft doesn't directly offer technical consulting for development of custom connectors. However as part of the program, Microsoft can recommend resources for developers to further engage with. Once registered with the program below, reach out to your Microsoft contact to learn more.
- The developer must provide an estimate for usage. We suggest that developers of connectors for very boutique products use our [connector self-signing capabilities](#) to provide them directly to the customer.
- The connector must be already made available to customers directly to fulfill a user need or business scenario. This can be done using a Private Preview program by distributing the completed connector directly to end users and organizations through [self-signing](#). Each user or organization should be able to provide feedback and validation that there's a business need for the connector and that the connector is working sucessfully to fulfill their business requirements.
- The connector must be working successfully at an anticipated level of usage by customers.
- There must be a thread in the [Power BI Ideas forum](#) driven by customers to indicate demand to make the connector publicly available in Power BI Desktop. There is no set threshold of engagement. However the more engagement, the stronger the evidenced demand for the connector.

These prerequisites exist to ensure that connectors undergoing certification have significant customer and business need to be used and supported post-certification.

Process and Timelines

Certified connectors are released with monthly Power BI Desktop releases, so the deadlines for each release work back from each Power BI Desktop release date. The expected duration of the certification process from registration to release varies depending on the quality and complexity of the connector submission, and is outlined in the following steps:

- **Registration:** notification of intent to certify your custom connector. This must occur by the 15th of the month, two months before the targeted Power BI desktop release.
 - For example, for the April Power BI Desktop release, the deadline would be February 15th.
- **Submission:** submission of connector files for Microsoft review. This must occur by the 1st of the month before the targeted Power BI desktop release.
 - For example, for the April Power BI Desktop release, the deadline would be March 1st.
- **Technical Review:** finalization of the connector files, passing Microsoft review and certification. This must occur by the 15th of the month before the targeted Power BI Desktop release.
 - For example, for the April Power BI Desktop release, the deadline would be March 15th.

Due to the complexity of the technical reviews and potential delays, rearchitecture, and testing issues, we highly recommend submitting early with a long lead time for the initial release and certification. If you feel like your connector is important to deliver to a few customers with minimal overhead, we recommend [self-signing](#) and providing it that way.

Certification Requirements

We have a certain set of requirements for certification. We recognize that not every developer can meet these requirements, and we're hoping to introduce a feature set that will handle developer needs in short order.

Submission Files (Artifacts)

Please ensure the connector files that you submit include all of the following:

- Connector (.mez) file
 - The .mez file should follow style standards and be named similarly to the product or service name. It should not include words like "Power BI", "Connector" or "API".
 - Name the .mez file: `ProductName.mez`
- Power BI Desktop (.pbix) file for testing
 - We require a sample Power BI report (.pbix) to test your connector with.
 - The report should include at least one query to test each item in your navigation table.
 - If there's no set schema (for example, databases), the report needs to include a query for each "type" of table that the connector may handle.
- Test account to your data source
 - We will use the test account to test and troubleshoot your connector.
 - Provide a test account that is persistent, so we can use the same account to certify any future updates.
- Testing instructions
 - Provide any documentation on how to use the connector and test its functionality.
- Links to external dependencies (for example, ODBC drivers)

Features and Style

The connector must follow a set of feature and style rules to meet a usability standard consistent with other certified connectors.

- The connector MUST:
 - Use Section document format.
 - Have [version adornment](#) on section.
 - Provide [function documentation metadata](#).
 - Have [TestConnection handler](#).
 - Follow naming conventions (for example, `DataSourceKind.FunctionName`). It should not include words like "Power BI", "Connector" or "API".
- The `FunctionName` should make sense for the domain (for example "Contents", "Tables", "Document", "Databases", and so on).
- The connector SHOULD:
 - Have icons.
 - Provide a navigation table.
 - Place strings in a `resources.resx` file. URLs and values should be hardcoded in the connector code and not be placed in the `resources.resx` file.

Security

There are specific security considerations that your connector must handle.

- If `Extension.CurrentCredentials()` is used:

- Is the usage required? If so, where do the credentials get sent to?
- Are the requests guaranteed to be made through HTTPS?
 - You can use the [HTTPS enforcement helper function](#).
- If the credentials are sent using `Web.Contents()` via GET:
 - Can it be turned into a POST?
 - If GET is required, the connector MUST use the `CredentialQueryString` record in the `Web.Contents()` options record to pass in sensitive credentials.
- If [Diagnostics.* functions](#) are used:
 - Validate what is being traced; data **must not contain PII or large amounts of unnecessary data**.
 - If you implemented significant tracing in development, you should implement a variable or feature flag that determines if tracing should be on. This must be **turned off** prior to submitting for certification.
- If `Expression.Evaluate()` is used:
 - Validate where the expression is coming from and what it is (that is, can dynamically construct calls to `Extension.CurrentCredentials()` and so on).
 - The `Expression` should not be user provided nor take user input.
 - The `Expression` should not be dynamic (that is, retrieved from a web call).

Registering for Certification

If you're interested in pursuing certification of your custom connector, ensure that your scenario and connector meet the [prerequisites](#) and [requirements](#) outlined in this article. Failure to do so will cause delays in certification as our team will require you to fix any issues or inconsistencies prior to moving forward with certification.

Ensure that your connector is code complete and has been tested in both authoring in Power BI Desktop, and refreshing and consumption in Power BI Service. Ensure you have tested full end-to-end refresh in Power BI Service through the use of an on-premises data gateway.

To get started, complete our [registration form](#), and a Microsoft contact will reach out to begin the process.

Template Apps (Recommended)

Once you've developed a connector to a data source, consider helping customers get up and running quickly by creating a [template app](#). A template app provides customers a prebuilt report connected to their data that they can use out-of-the-box or customize as necessary.

NOTE

Template apps do not support connectors that require a gateway.

Power Query connector submission

1/15/2022 • 3 minutes to read • [Edit Online](#)

Introduction

This article provides instructions for how to submit your Power Query custom connector for certification. Do not submit your connector for certification unless you've been directed to by your Microsoft contact.

Prerequisites

After you've been approved for certification, ensure that your connector meets the [certification requirements](#) and follows all feature, style, and security guidelines. Prepare the [submission artifacts](#) for submission.

Submitting to the Connector Certification Portal

The Connector Certification Portal is an experience within [ISV Studio](#) that allows Microsoft partners and ISVs to submit, view, and manage connectors submitted for certification. After submission, communication between Microsoft and the partner connector developer will be through the Connector Certification Portal.

Initial Submission

1. Navigate to [ISV Studio](#) and log in with your work Microsoft account. Personal accounts aren't supported in this experience.
2. Select the **Connector certification** tab on the left to launch the Connector Certification Portal experience.
3. Select **Power Query** as your connector type.
4. Read and agree to our connector Partner Agreement. If there is a separate agreement governing the relationship between your organization and Microsoft, let your Microsoft contact know.
5. Upload your `.mez` file and complete the form with information on your connector. Submit the form to finish the connector submission process. Once submitted, you can use the Activity Control experience on the right to communicate with your Microsoft contact.
6. Read the guidelines for providing [documentation](#) for your custom connector. Create a Markdown (`.md`) file following the custom connector documentation guidelines, leveraging examples from existing documentation if needed. This step is crucial to ensure users know how to use your connector. Once complete, submit the Markdown (`.md`) file in the Activity Control as an attachment.
7. Also in the Activity Control, share a paragraph introducing your connector and explaining its value proposition to users and customers. This entry will be submitted as part of the Power BI blog in the upcoming month's post, announcing the release of your connector.
8. If you would like to add teammates to manage your connector, let your Microsoft contact know.

After your connector code review is complete, our team will schedule a demo for your team to walk us through the connector experience. We will be validating the following scenarios:

- You can successfully connect to the service.
- Your connector correctly displays navigation tables.
- Your connector correctly authenticates with the expected authentication method.
- You can successfully refresh the connector.

- You can successfully publish to the service.
- You can successfully refresh end-to-end through the gateway.

Ensure that these scenarios are working successfully before the demo so that the demo goes smooth and doesn't block the release of your connector.

Updates

Updates to your connector submission can be made at any time, except when your connector is in the process of production deployment. When you are submitting an update, ensure that you submit an update to your existing submission, rather than creating a new submission.

1. Navigate to the **Connector certification** experience within [ISV Studio](#).
2. From the list of connectors you manage, select the connector submission to view its connector versions.
3. For an update to a certified connector, select the link to submit a new version in the panel on the right, on top of the existing connector versions. For an update to an existing connector version undergoing certification, select the most recent connector version and on the bottom left, select the **Submit an update** button.
4. You can upload a new version of artifacts and complete the submission form again.
5. After submitting the connector form, in the **Activity Control** chat feature on the right, submit a short changelog explaining the connector update. This information should be public and written in a customer-facing way, as it will be included in the next Power BI Desktop blog update.

Providing user documentation for your custom connector

1/15/2022 • 4 minutes to read • [Edit Online](#)

Once you've finished designing your Power Query custom connector, you'll need to submit an article that provides instructions on how to use your connector for publication on docs.microsoft.com. This article discusses the layout of such an article and how to format the text of your article.

Article layout

This section describes the general layout of the Power Query connector articles. Your custom connector article should follow this general layout.

Support note

Right after the title of the article, insert the following note.

NOTE

The following connector article is provided by <*company name*>, the owner of this connector and a member of the Microsoft Power Query Connector Certification Program. If you have questions regarding the content of this article or have changes you would like to see made to this article, visit the <*company name*> website and use the support channels there.

Replace <*company name*> with your company name.

Summary table

After the support note, provide a summary table that contains the following information:

- **Release state:** Indicates whether the connector is in preview or general availability. Use either "Preview" or "General Availability".
- **Products supported:** Lists the products that can use your custom connector.
- **Authentication types supported:** Lists the authentication types your custom connector supports.
- **Function reference docs:** Lists any M formula language docs that are associated with your custom connector.

Summary

Item	Description
Release State	General Availability
Products	Power BI (Datasets) Power BI (Dataflows) Power Apps (Dataflows) Excel Dynamics 365 Customer Insights
Authentication Types Supported	Anonymous Windows Basic Web API Organizational Account
Function Reference Documentation	Web.Page Web.BrowserContents

If your custom connector is implemented on various products, but has different capabilities on these products, include the following or similar note in your article after the summary table.

NOTE

Some capabilities may be present in one product but not others due to deployment schedules and host-specific capabilities.

Prerequisites

If your custom connector requires that other applications be installed on the system running your connector or requires that a set-up procedure be done before using your custom connector, you must include a Prerequisites section that describes these installation and set-up procedures. This section will also include any information about setting up various versions of your connector (if applicable).

Capabilities supported

This section should contain a list of the capabilities supported by your custom connector. These capabilities are usually a bulleted list that indicates if the connector supports Import and DirectQuery modes, and also any advanced options that are available in the initial dialog box that appears after the user selects your connector in **Get data**.

Connection instructions

This section contains the procedures required to connect to data. If your custom connector is only used in Power Query Desktop, only one procedure is required. However, if your custom connector is used on both Power Query Desktop and Power Query Online, you must supply a separate procedure in separate sections for each instance. That is, if your custom connector is only used by Power Query Desktop, you'll have one procedure starting with a second order heading and a single step-by-step procedure. If your custom connector is used by both Power Query Desktop and Power Query Online, you'll have two procedures. Each procedure starts with a second order heading, and contains a separate step-by-step procedure under each heading. For examples of each of these types of procedures, go to [Example connector articles](#).

The procedure is made up of a numbered list that includes each step required to fill in the information needed to provide a normal connection (not requiring advance options) to the data.

Connect using advanced options (optional)

If your custom connector contains advanced options that can be used to connect to the data, this information should be covered in a separate section of the documentation. Each of the advanced options should be documented, and the purpose of each advanced option explained in this section.

Troubleshooting (optional)

If you know of any common errors that may occur with your custom connector, you can add a troubleshooting section to describe ways to either fix the error, or work around the error. This section can also include information on any known limitations of your connector or the retrieval of data. You can also include any known issues with using your connector to connect to data.

Additional instructions (optional)

Any other instructions or information about your connector that hasn't been covered by the previous sections can go in this section.

Article location and format

Your article should be made available on GitHub under the Connectors folder in the Power Query docs repo: <https://github.com/MicrosoftDocs/powerquery-docs/tree/master/powerquery-docs/Connectors>. Ensure that you also add a link to your article in the [list of connectors](#) referencing the correct logo image uploaded to `/Connectors/media/index` folder. Lastly, ensure that you add a link to your article in the table of contents file (`TOC.yml`). Certified connectors will only be available under [Power BI \(Datasets\)](#).

See the [Microsoft Docs contributor guide](#) on how you can contribute to our repo.

The article should be formatted and submitted as a Markdown file. It should use the Microsoft style for describing procedures and the UI layout.

The following articles include instructions on formatting your document in Markdown, and the Microsoft style that you should follow when authoring your article:

- [Docs Markdown reference](#)
- [Microsoft Writing Style Guide](#)

Example connector articles

Here's a couple of example Power Query connector articles that you can use as a starting point:

- [Adobe Analytics](#) (Power Query Desktop only)
- [Azure SQL database](#) (Power Query Desktop and Power Query Online)