

100 XP

Introduction

1 minute

Azure Synapse Analytics includes serverless SQL pools, which are tailored for querying data in a data lake. With a serverless SQL pool you can use SQL code to query data in files of various common formats without needing to load the file data into database storage. This capability helps data analysts and data engineers analyze and process file data in the data lake using a familiar data processing language, without the need to create or maintain a relational database store.

After completing this module, you'll be able to:

- Identify capabilities and use cases for serverless SQL pools in Azure Synapse Analytics
- Query CSV, JSON, and Parquet files using a serverless SQL pool
- Create external database objects in a serverless SQL pool

Prerequisites

Before starting this module, you should have the following prerequisite skills and knowledge:

- Familiarity with the Microsoft Azure portal
- Familiarity with data lake and data warehouse concepts
- Experience of using SQL to query database tables

Next unit: Understand Azure Synapse serverless SQL pool capabilities and use cases

[Continue >](#)

How are we doing?

100 XP



Understand Azure Synapse serverless SQL pool capabilities and use cases

5 minutes

Azure Synapse Analytics is an integrated analytics service that brings together a wide range of commonly used technologies for processing and analyzing data at scale. One of the most prevalent technologies used in data solutions is SQL – an industry standard language for querying and manipulating data.

Serverless SQL pools in Azure Synapse Analytics

Azure Synapse SQL is a distributed query system in Azure Synapse Analytics that offers two kinds of runtime environments:

- **Serverless SQL pool:** on-demand SQL query processing, primarily used to work with data in a data lake.
- **Dedicated SQL pool:** Enterprise-scale relational database instances used to host data warehouses in which data is stored in relational tables.

In this module, we'll focus on serverless SQL pool, which provides a pay-per-query endpoint to query the data in your data lake. The benefits of using serverless SQL pool include:

- A familiar Transact-SQL syntax to query data in place without the need to copy or load data into a specialized store.
- Integrated connectivity from a wide range of business intelligence and ad-hoc querying tools, including the most popular drivers.
- Distributed query processing that is built for large-scale data, and computational functions - resulting in fast query performance.
- Built-in query execution fault-tolerance, resulting in high reliability and success rates even for long-running queries involving large data sets.
- No infrastructure to setup or clusters to maintain. A built-in endpoint for this service is provided within every Azure Synapse workspace, so you can start querying data as soon as the workspace is created.
- No charge for resources reserved, you're only charged for the data processed by queries you run.

When to use serverless SQL pools

Serverless SQL pool is tailored for querying the data residing in the data lake, so in addition to eliminating management burden, it eliminates a need to worry about ingesting the data into the system. You just point the query to the data that is already in the lake and run it.

Synapse SQL serverless resource model is great for unplanned or "bursty" workloads that can be processed using the always-on serverless SQL endpoint in your Azure Synapse Analytics workspace. Using the serverless pool helps when you need to know exact cost for each query executed to monitor and attribute costs.

! Note

Serverless SQL pool is an analytics system and is not recommended for OLTP workloads such as databases used by applications to store transactional data. Workloads that require millisecond response times and are looking to pinpoint a single row in a data set are not good fit for serverless SQL pool.

Common use cases for serverless SQL pools include:

- **Data exploration:** Data exploration involves browsing the data lake to get initial insights about the data, and is easily achievable with Azure Synapse Studio. You can browse through the files in your linked data lake storage, and use the built-in serverless SQL pool to automatically generate a SQL script to select TOP 100 rows from a file or folder just as you would do with a table in SQL Server. From there, you can apply projections, filtering, grouping, and most of the operation over the data as if the data were in a regular SQL Server table.
- **Data transformation:** While Azure Synapse Analytics provides great data transformations capabilities with Synapse Spark, some data engineers might find data transformation easier to achieve using SQL. Serverless SQL pool enables you to perform SQL-based data transformations; either interactively or as part of an automated data pipeline.
- **Logical data warehouse:** After your initial exploration of the data in the data lake, you can define external objects such as tables and views in a serverless SQL database. The data remains stored in the data lake files, but are abstracted by a relational schema that can be used by client applications and analytical tools to query the data as they would in a relational database hosted in SQL Server.

Next unit: Query files using a serverless SQL pool

✓ 100 XP

Query files using a serverless SQL pool

10 minutes

You can use a serverless SQL pool to query data files in various common file formats, including:

- Delimited text, such as comma-separated values (CSV) files.
- JavaScript object notation (JSON) files.
- Parquet files.

The basic syntax for querying is the same for all of these types of file, and is built on the OPENROWSET SQL function; which generates a tabular rowset from data in one or more files. For example, the following query could be used to extract data from CSV files.

SQL

```
SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.csv',
    FORMAT = 'csv') AS rows
```

The OPENROWSET function includes more parameters that determine factors such as:

- The schema of the resulting rowset
- Additional formatting options for delimited text files.

💡 Tip

You'll find the full syntax for the OPENROWSET function in the [Azure Synapse Analytics documentation](#).

The output from OPENROWSET is a rowset to which an alias must be assigned. In the previous example, the alias **rows** is used to name the resulting rowset.

The **BULK** parameter includes the full URL to the location in the data lake containing the data files. This can be an individual file, or a folder with a wildcard expression to filter the file types that should be included. The **FORMAT** parameter specifies the type of data being queried. The example above reads delimited text from all .csv files in the **files** folder.

ⓘ Note

This example assumes that the user has access to the files in the underlying store. If the files are protected with a SAS key or custom identity, you would need to [create a server-scoped credential](#).

As seen in the previous example, you can use wildcards in the **BULK** parameter to include or exclude files in the query. The following list shows a few examples of how this can be used:

- `https://mydatalake.blob.core.windows.net/data/files/file1.csv`: Only include `file1.csv` in the `files` folder.
- `https://mydatalake.blob.core.windows.net/data/files/file*.csv`: All .csv files in the `files` folder with names that start with "file".
- `https://mydatalake.blob.core.windows.net/data/files/*`: All files in the `files` folder.
- `https://mydatalake.blob.core.windows.net/data/files/**`: All files in the `files` folder, and recursively its subfolders.

You can also specify multiple file paths in the **BULK** parameter, separating each path with a comma.

Querying delimited text files

Delimited text files are a common file format within many businesses. The specific formatting used in delimited files can vary, for example:

- With and without a header row.
- Comma and tab-delimited values.
- Windows and Unix style line endings.
- Non-quoted and quoted values, and escaping characters.

Regardless of the type of delimited file you're using, you can read data from them by using the **OPENROWSET** function with the **csv** **FORMAT** parameter, and other parameters as required to handle the specific formatting details for your data. For example:

SQL

```
SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.csv',
    FORMAT = 'csv',
    PARSER_VERSION = '2.0',
    FIRSTROW = 2) AS rows
```

The **PARSER_VERSION** is used to determine how the query interprets the text encoding used in the files. Version 1.0 is the default and supports a wide range of file encodings, while version

2.0 supports fewer encodings but offers better performance. The **FIRSTROW** parameter is used to skip rows in the text file, to eliminate any unstructured preamble text or to ignore a row containing column headings.

Additional parameters you might require when working with delimited text files include:

- **FIELDTERMINATOR** - the character used to separate field values in each row. For example, a tab-delimited file separates fields with a TAB (`\t`) character. The default field terminator is a comma (,).
- **ROWTERMINATOR** - the character used to signify the end of a row of data. For example, a standard Windows text file uses a combination of a carriage return (CR) and line feed (LF), which is indicated by the code `\n`; while UNIX-style text files use a single line feed character, which can be indicated using the code `0x0a`.
- **FIELDQUOTE** - the character used to enclose quoted string values. For example, to ensure that the comma in the address field value *126 Main St, apt 2* isn't interpreted as a field delimiter, you might enclose the entire field value in quotation marks like this: "*126 Main St, apt 2*". The double-quote ("") is the default field quote character.

💡 Tip

For details of additional parameters when working with delimited text files, refer to the [Azure Synapse Analytics documentation](#).

Specifying the rowset schema

It's common for delimited text files to include the column names in the first row. The **OPENROWSET** function can use this to define the schema for the resulting rowset, and automatically infer the data types of the columns based on the values they contain. For example, consider the following delimited text:

text
product_id,product_name,list_price
123,Widget,12.99
124,Gadget,3.99

The data consists of the following three columns:

- **product_id** (integer number)
- **product_name** (string)
- **list_price** (decimal number)

You could use the following query to extract the data with the correct column names and appropriately inferred SQL Server data types (in this case INT, NVARCHAR, and DECIMAL)

SQL

```
SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.csv',
    FORMAT = 'csv',
    PARSER_VERSION = '2.0',
    HEADER_ROW = TRUE) AS rows
```

The **HEADER_ROW** parameter (which is only available when using parser version 2.0) instructs the query engine to use the first row of data in each file as the column names, like this:

product_id	product_name	list_price
123	Widget	12.9900
124	Gadget	3.9900

Now consider the following data:

text

```
123,Widget,12.99
124,Gadget,3.99
```

This time, the file doesn't contain the column names in a header row; so while the data types can still be inferred, the column names will be set to **C1**, **C2**, **C3**, and so on.

C1	C2	C3
123	Widget	12.9900
124	Gadget	3.9900

To specify explicit column names and data types, you can override the default column names and inferred data types by providing a schema definition in a **WITH** clause, like this:

SQL

```

SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.csv',
    FORMAT = 'csv',
    PARSER_VERSION = '2.0')
WITH (
    product_id INT,
    product_name VARCHAR(20) COLLATE Latin1_General_100_BIN2_UTF8,
    list_price DECIMAL(5,2)
) AS rows

```

This query produces the expected results:

product_id	product_name	list_price
123	Widget	12.99
124	Gadget	3.99

💡 Tip

When working with text files, you may encounter some incompatibility with UTF-8 encoded data and the collation used in the **master** database for the serverless SQL pool. To overcome this, you can specify a compatible collation for individual VARCHAR columns in the schema. See the [troubleshooting guidance](#) for more details.

Querying JSON files

JSON is a popular format for web applications that exchange data through REST interfaces or use NoSQL data stores such as Azure Cosmos DB. So, it's not uncommon to persist data as JSON documents in files in a data lake for analysis.

For example, a JSON file that defines an individual product might look like this:

JSON
<pre>{ "product_id": 123, "product_name": "Widget", "list_price": 12.99 }</pre>

To return product data from a folder containing multiple JSON files in this format, you could use the following SQL query:

SQL

```
SELECT doc
FROM
OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.json',
    FORMAT = 'csv',
    FIELDTERMINATOR ='0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
) WITH (doc NVARCHAR(MAX)) as rows
```

OPENROWSET has no specific format for JSON files, so you must use `csv` format with `FIELDTERMINATOR`, `FIELDQUOTE`, and `ROWTERMINATOR` set to `0x0b`, and a schema that includes a single `NVARCHAR(MAX)` column. The result of this query is a rowset containing a single column of JSON documents, like this:

doc

```
{"product_id":123,"product_name":"Widget","list_price": 12.99}
```

```
{"product_id":124,"product_name":"Gadget","list_price": 3.99}
```

To extract individual values from the JSON, you can use the `JSON_VALUE` function in the `SELECT` statement, as shown here:

SQL

```
SELECT JSON_VALUE(doc, '$.product_name') AS product,
       JSON_VALUE(doc, '$.list_price') AS price
FROM
OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.json',
    FORMAT = 'csv',
    FIELDTERMINATOR ='0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
) WITH (doc NVARCHAR(MAX)) as rows
```

This query would return a rowset similar to the following results:

product	price
Widget	12.99
Gadget	3.99

Querying Parquet files

Parquet is a commonly used format for big data processing on distributed file storage. It's an efficient data format that is optimized for compression and analytical querying.

In most cases, the schema of the data is embedded within the Parquet file, so you only need to specify the **BULK** parameter with a path to the file(s) you want to read, and a **FORMAT** parameter of *parquet*; like this:

SQL

```
SELECT TOP 100 *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/files/*.*',
    FORMAT = 'parquet') AS rows
```

Query partitioned data

It's common in a data lake to partition data by splitting across multiple files in subfolders that reflect partitioning criteria. This enables distributed processing systems to work in parallel on multiple partitions of the data, or to easily eliminate data reads from specific folders based on filtering criteria. For example, suppose you need to efficiently process sales order data, and often need to filter based on the year and month in which orders were placed. You could partition the data using folders, like this:

- /orders
 - /year=2020
 - /month=1
 - /01012020.parquet
 - /02012020.parquet
 - ...
 - /month=2
 - /01022020.parquet
 - /02022020.parquet

- ...
- ...
- /year=2021
 - /month=1
 - /01012021.parquet
 - /02012021.parquet
 - ...
 - ...

To create a query that filters the results to include only the orders for January and February 2020, you could use the following code:

SQL

```
SELECT *
FROM OPENROWSET(
    BULK
    'https://mydatalake.blob.core.windows.net/data/orders/year=*/month=*/.*.*',
    FORMAT = 'parquet') AS orders
WHERE orders.filepath(1) = '2020'
    AND orders.filepath(2) IN ('1','2');
```

The numbered filepath parameters in the WHERE clause reference the wildcards in the folder names in the BULK path -so the parameter 1 is the * in the *year=** folder name, and parameter 2 is the * in the *month=** folder name.

Next unit: Create external database objects

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



Create external database objects

6 minutes

You can use the OPENROWSET function in SQL queries that run in the default **master** database of the built-in serverless SQL pool to explore data in the data lake. However, sometimes you may want to create a custom database that contains some objects that make it easier to work with external data in the data lake that you need to query frequently.

Creating a database

You can create a database in a serverless SQL pool just as you would in a SQL Server instance. You can use the graphical interface in Synapse Studio, or a CREATE DATABASE statement. One consideration is to set the collation of your database so that it supports conversion of text data in files to appropriate Transact-SQL data types.

The following example code creates a database named *salesDB* with a collation that makes it easier to import UTF-8 encoded text data into VARCHAR columns.

SQL

```
CREATE DATABASE SalesDB  
    COLLATE Latin1_General_100_BIN2_UTF8
```

Creating an external data source

You can use the OPENROWSET function with a BULK path to query file data from your own database, just as you can in the **master** database; but if you plan to query data in the same location frequently, it's more efficient to define an external data source that references that location. For example, the following code creates a data source named *files* for the hypothetical <https://mydatalake.blob.core.windows.net/data/files/> folder:

SQL

```
CREATE EXTERNAL DATA SOURCE files  
WITH (  
    LOCATION = 'https://mydatalake.blob.core.windows.net/data/files/'  
)
```

One benefit of an external data source, is that you can simplify an OPENROWSET query to use the combination of the data source and the relative path to the folders or files you want to query:

SQL

```
SELECT *
FROM
OPENROWSET(
    BULK 'orders/*.csv',
    DATA_SOURCE = 'files',
    FORMAT = 'csv',
    PARSER_VERSION = '2.0'
) AS orders
```

In this example, the **BULK** parameter is used to specify the relative path for all .csv files in the **orders** folder, which is a subfolder of the **files** folder referenced by the data source.

Another benefit of using a data source is that you can assign a credential for the data source to use when accessing the underlying storage, enabling you to provide access to data through SQL without permitting users to access the data directly in the storage account. For example, the following code creates a credential that uses a shared access signature (SAS) to authenticate against the underlying Azure storage account hosting the data lake.

SQL

```
CREATE DATABASE SCOPED CREDENTIAL sqlcred
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=xxx...';
GO

CREATE EXTERNAL DATA SOURCE secureFiles
WITH (
    LOCATION = 'https://mydatalake.blob.core.windows.net/data/secureFiles/',
    CREDENTIAL = sqlcred
);
GO
```

💡 Tip

In addition to SAS authentication, you can define credentials that use *managed identity* (the Azure Active Directory identity used by your Azure Synapse workspace), a specific Azure Active Directory principal, or passthrough authentication based on the identity of the user running the query (which is the default type of authentication). To learn more about using credentials in a serverless SQL pool, see the [Control storage account access](#)

for serverless SQL pool in Azure Synapse Analytics article in Azure Synapse Analytics documentation.

Creating an external file format

While an external data source simplifies the code needed to access files with the OPENROWSET function, you still need to provide format details for the file being accessed; which may include multiple settings for delimited text files. You can encapsulate these settings in an external file format, like this:

SQL

```
CREATE EXTERNAL FILE FORMAT CsvFormat
    WITH (
        FORMAT_TYPE = DELIMITEDTEXT,
        FORMAT_OPTIONS(
            FIELD_TERMINATOR = ',',
            STRING_DELIMITER = ''
        )
    );
GO
```

After creating file formats for the specific data files you need to work with, you can use the file format to create external tables, as discussed next.

Creating an external table

When you need to perform a lot of analysis or reporting from files in the data lake, using the OPENROWSET function can result in complex code that includes data sources and file paths. To simplify access to the data, you can encapsulate the files in an external table; which users and reporting applications can query using a standard SQL SELECT statement just like any other database table. To create an external table, use the CREATE EXTERNAL TABLE statement, specifying the column schema as for a standard table, and including a WITH clause specifying the external data source, relative path, and external file format for your data.

SQL

```
CREATE EXTERNAL TABLE dbo.products
(
    product_id INT,
    product_name VARCHAR(20),
    list_price DECIMAL(5,2)
)
WITH
```

```
(  
    DATA_SOURCE = files,  
    LOCATION = 'products/*.csv',  
    FILE_FORMAT = CsvFormat  
);  
GO  
  
-- query the table  
SELECT * FROM dbo.products;
```

By creating a database that contains the external objects discussed in this unit, you can provide a relational database layer over files in a data lake, making it easier for many data analysts and reporting tools to access the data by using standard SQL query semantics.

Next unit: Exercise - Query files using a serverless SQL pool

[Continue >](#)

How are we doing?

Knowledge check

5 minutes

1. What function is used to read the data in files stored in a data lake? *

FORMAT

ROWSET

✗ Incorrect. The ROWSET is not a valid function.

OPENROWSET

✓ Correct. The OPENROWSET is used to read the data in files stored in a data lake.

2. What character in file path can be used to select all the file/folders that match rest of the path? *

&

*

✓ Correct. The asterisk character in file path can be used to select all the file or folders that match rest of the path.

/

3. Which external database object encapsulates the connection information to a file location in a data lake store? *

FILE FORMAT

DATA SOURCE

✓ Correct. a DATA SOURCE provides the connection information to the files in a data lake store.

EXTERNAL TABLE

✗ Incorrect. An EXTERNAL TABLE creates the table object without selecting data into it.



Summary

1 minute

Serverless SQL pools enable you to easily query files in data lake. You can query various file formats CSV, JSON, Parquet, and create external database objects to provide a relational abstraction layer over the raw files.

In this module, you've learned how to:

- Identify capabilities and use cases for serverless SQL pools in Azure Synapse Analytics
- Query CSV, JSON, and Parquet files using a serverless SQL pool
- Create external database objects in a serverless SQL pool

Learn more

To learn more about using serverless SQL pools to query files, refer to the [Azure Synapse Analytics documentation](#).

Module complete:

[Unlock achievement](#)

How are we doing? ★ ★ ★ ★ ★

100 XP

Introduction

1 minute

While SQL is commonly used by data analysts to query data and support analytical and reporting workloads, data engineers often need to use SQL to *transform* data; often as part of a data ingestion pipeline or extract, transform, and load (ETL) process.

In this module, you'll learn how to use `CREATE EXTERNAL TABLE AS SELECT` (CETAS) statements to transform data, and store the results in files in a data lake that can be queried through a relational table in a serverless SQL database or processed directly from the file system.

After completing this module, you'll be able to:

- Use a `CREATE EXTERNAL TABLE AS SELECT` (CETAS) statement to transform data.
- Encapsulate a CETAS statement in a stored procedure.
- Include a data transformation stored procedure in a pipeline.

Prerequisites

Before starting this module, you should have the following prerequisite skills and knowledge:

- Familiarity with Azure Synapse Analytics.
- Experience using Transact-SQL to query and manipulate data.

Next unit: Transform data files with the `CREATE EXTERNAL TABLE AS SELECT` statement

[Continue >](#)

How are we doing?

✓ 100 XP



Transform data files with the CREATE EXTERNAL TABLE AS SELECT statement

5 minutes

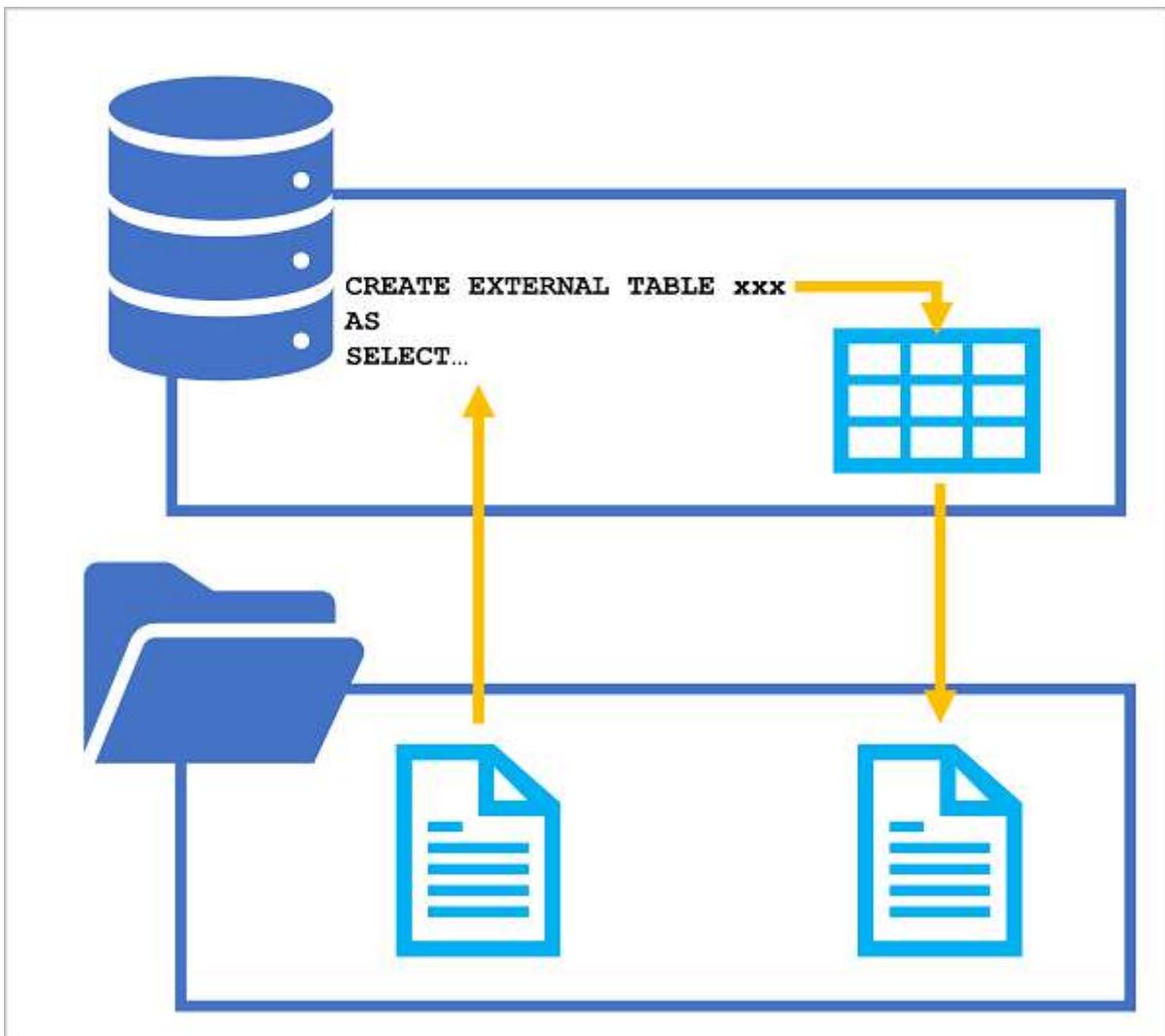
The SQL language includes many features and functions that enable you to manipulate data. For example, you can use SQL to:

- Filter rows and columns in a dataset.
- Rename data fields and convert between data types.
- Calculate derived data fields.
- Manipulate string values.
- Group and aggregate data.

Azure Synapse serverless SQL pools can be used to run SQL statements that transform data and persist the results as a file in a data lake for further processing or querying. If you're familiar with Transact-SQL syntax, you can craft a SELECT statement that applies the specific transformation you're interested in, and store the results of the SELECT statement in a selected file format with a metadata table schema that can be queried using SQL.

You can use a CREATE EXTERNAL TABLE AS SELECT (CETAS) statement in a dedicated SQL pool or serverless SQL pool to persist the results of a query in an external table, which stores its data in a file in the data lake.

The CETAS statement includes a SELECT statement that queries and manipulates data from any valid data source (which could be an existing table or view in a database, or an OPENROWSET function that reads file-based data from the data lake). The results of the SELECT statement are then persisted in an external table, which is a metadata object in a database that provides a relational abstraction over data stored in files. The following diagram illustrates the concept visually:



By applying this technique, you can use SQL to extract and transform data from files or tables, and store the transformed results for downstream processing or analysis. Subsequent operations on the transformed data can be performed against the relational table in the SQL pool database or directly against the underlying data files.

Creating external database objects to support CETAS

To use CETAS expressions, you must create the following types of object in a database for either a serverless or dedicated SQL pool. When using a serverless SQL pool, create these objects in a custom database (created using the `CREATE DATABASE` statement), not the **built-in** database.

External data source

An external data source encapsulates a connection to a file system location in a data lake. You can then use this connection to specify a relative path in which the data files for the external table created by the CETAS statement are saved.

If the source data for the CETAS statement is in files in the same data lake path, you can use the same external data source in the OPENROWSET function used to query it. Alternatively, you can create a separate external data source for the source files or use a fully qualified file path in the OPENROWSET function.

To create an external data source, use the CREATE EXTERNAL DATA SOURCE statement, as shown in this example:

SQL

```
CREATE EXTERNAL DATA SOURCE files
WITH (
    LOCATION = 'https://mydatalake.blob.core.windows.net/data/files/'
);
```

The previous example assumes that users running queries that use the external data source will have sufficient permissions to access the files. An alternative approach is to encapsulate a credential in the external data source so that it can be used to access file data without granting all users permissions to read it directly:

SQL

```
CREATE DATABASE SCOPED CREDENTIAL storagekeycred
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=xxx...';

CREATE EXTERNAL DATA SOURCE secureFiles
WITH (
    LOCATION = 'https://mydatalake.blob.core.windows.net/data/secureFiles/',
    CREDENTIAL = storagekeycred
);
```

💡 Tip

In addition to SAS authentication, you can define credentials that use *managed identity* (the Azure Active Directory identity used by your Azure Synapse workspace), a specific Azure Active Directory principal, or passthrough authentication based on the identity of the user running the query (which is the default type of authentication). To learn more about using credentials in a serverless SQL pool, see the [Control storage account access for serverless SQL pool in Azure Synapse Analytics](#) article in Azure Synapse Analytics documentation.

External file format

The CETAS statement creates a table with its data stored in files. You must specify the format of the files you want to create as an external file format.

To create an external file format, use the `CREATE EXTERNAL FILE FORMAT` statement, as shown in this example:

SQL

```
CREATE EXTERNAL FILE FORMAT ParquetFormat
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
);
```

Tip

In this example, the files will be saved in Parquet format. You can also create external file formats for other types of file. See [CREATE EXTERNAL FILE FORMAT \(Transact-SQL\)](#) for details.

Using the CETAS statement

After creating an external data source and external file format, you can use the CETAS statement to transform data and stored the results in an external table.

For example, suppose the source data you want to transform consists of sales orders in comma-delimited text files that are stored in a folder in a data lake. You want to filter the data to include only orders that are marked as "special order", and save the transformed data as Parquet files in a different folder in the same data lake. You could use the same external data source for both the source and destination folders as shown in this example:

SQL

```
CREATE EXTERNAL TABLE SpecialOrders
WITH (
    -- details for storing results
    LOCATION = 'special_orders/',
    DATA_SOURCE = files,
    FILE_FORMAT = ParquetFormat
)
AS
SELECT OrderID, CustomerName, OrderTotal
```

```

FROM
OPENROWSET(
    -- details for reading source files
    BULK 'sales_orders/*.csv',
    DATA_SOURCE = 'files',
    FORMAT = 'CSV',
    PARSER_VERSION = '2.0',
    HEADER_ROW = TRUE
) AS source_data
WHERE OrderType = 'Special Order';

```

The LOCATION and BULK parameters in the previous example are relative paths for the results and source files respectively. The paths are relative to the file system location referenced by the **files** external data source.

An important point to understand is that you **must** use an external data source to specify the location where the transformed data for the external table is to be saved. When file-based source data is stored in the same folder hierarchy, you can use the same external data source. Otherwise, you can use a second data source to define a connection to the source data or use the fully qualified path, as shown in this example:

SQL

```

CREATE EXTERNAL TABLE SpecialOrders
WITH (
    -- details for storing results
    LOCATION = 'special_orders/',
    DATA_SOURCE = files,
    FILE_FORMAT = ParquetFormat
)
AS
SELECT OrderID, CustomerName, OrderTotal
FROM
OPENROWSET(
    -- details for reading source files
    BULK
    'https://mystorage.blob.core.windows.net/data/sales_orders/*.csv',
    FORMAT = 'CSV',
    PARSER_VERSION = '2.0',
    HEADER_ROW = TRUE
) AS source_data
WHERE OrderType = 'Special Order';

```

Dropping external tables

If you no longer need the external table containing the transformed data, you can drop it from the database by using the **DROP EXTERNAL TABLE** statement, as shown here:

SQL

```
DROP EXTERNAL TABLE SpecialOrders;
```

However, it's important to understand that external tables are a metadata abstraction over the files that contain the actual data. Dropping an external table does *not* delete the underlying files.

Next unit: Encapsulate data transformations in a stored procedure

[Continue >](#)

How are we doing?

✓ 100 XP

Encapsulate data transformations in a stored procedure

4 minutes

While you can run a `CREATE EXTERNAL TABLE AS SELECT` (CETAS) statement in a script whenever you need to transform data, it's good practice to encapsulate the transformation operation in stored procedure. This approach can make it easier to operationalize data transformations by enabling you to supply parameters, retrieve outputs, and include additional logic in a single procedure call.

For example, the following code creates a stored procedure that drops the external table if it already exists before recreating it with order data for the specified year:

SQL

```
CREATE PROCEDURE usp_special_orders_by_year @order_year INT
AS
BEGIN

    -- Drop the table if it already exists
    IF EXISTS (
        SELECT * FROM sys.external_tables
        WHERE name = 'SpecialOrders'
    )
        DROP EXTERNAL TABLE SpecialOrders

    -- Create external table with special orders
    -- from the specified year
    CREATE EXTERNAL TABLE SpecialOrders
        WITH (
            LOCATION = 'special_orders/',
            DATA_SOURCE = files,
            FILE_FORMAT = ParquetFormat
        )
    AS
    SELECT OrderID, CustomerName, OrderTotal
    FROM
        OPENROWSET(
            BULK 'sales_orders/*.csv',
            DATA_SOURCE = 'files',
            FORMAT = 'CSV',
            PARSER_VERSION = '2.0',
            HEADER_ROW = TRUE
        ) AS source_data
    WHERE OrderType = 'Special Order'
```

```
    AND YEAR(OrderDate) = @order_year  
END
```

⚠ Note

As discussed previously, dropping an existing external table does not delete the folder containing its data files. You must explicitly delete the target folder if it exists before running the stored procedure, or an error will occur.

In addition to encapsulating Transact-SQL logic, stored procedures also provide the following benefits:

Reduces client to server network traffic

The commands in a procedure are executed as a single batch of code; which can significantly reduce network traffic between the server and client because only the call to execute the procedure is sent across the network.

Provides a security boundary

Multiple users and client programs can perform operations on underlying database objects through a procedure, even if the users and programs don't have direct permissions on those underlying objects. The procedure controls what processes and activities are performed and protects the underlying database objects; eliminating the requirement to grant permissions at the individual object level and simplifies the security layers.

Eases maintenance

Any changes in the logic or file system locations involved in the data transformation can be applied only to the stored procedure; without requiring updates to client applications or other calling functions.

Improved performance

Stored procedures are compiled the first time they're executed, and the resulting execution plan is held in the cache and reused on subsequent runs of the same stored procedure. As a result, it takes less time to process the procedure.

✓ 100 XP



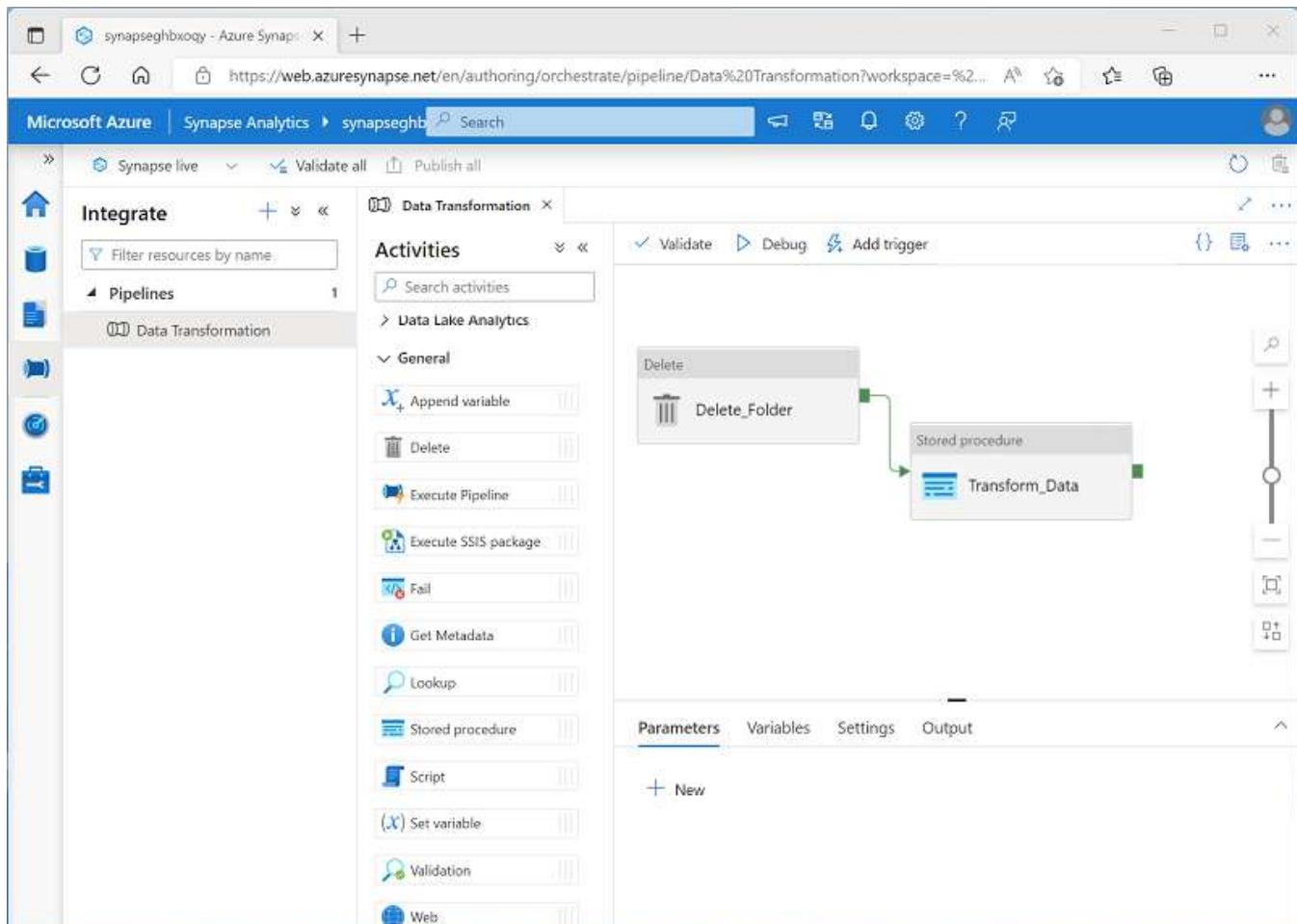
Include a data transformation stored procedure in a pipeline

3 minutes

Encapsulating a CREATE EXTERNAL TABLE AS SELECT (CETAS) statement in a stored procedure makes it easier for you to operationalize data transformations that you may need to perform repeatedly. In Azure Synapse Analytics and Azure Data Factory, you can create pipelines that connect to *linked services*, including Azure Data Lake Store Gen2 storage accounts that host data lake files, and serverless SQL pools; enabling you to call your stored procedures as part of an overall data extract, transform, and load (ETL) pipeline.

For example, you can create a pipeline that includes the following activities:

- A **Delete** activity that deletes the target folder for the transformed data in the data lake if it already exists.
- A **Stored procedure** activity that connects to your serverless SQL pool and runs the stored procedure that encapsulates your CETAS operation.



Creating a pipeline for the data transformation enables you to schedule the operation to run at specific times or based on specific events (such as new files being added to the source storage location).

 **Tip**

For more information about using the **Stored procedure** activity in a pipeline, see [Transform data by using the SQL Server Stored Procedure activity in Azure Data Factory](#) or [Synapse Analytics](#) in the Azure Data Factory documentation.

Next unit: Exercise - Transform files using a serverless SQL pool

[Continue >](#)

How are we doing?     

1. You need to store the results of a query in a serverless SQL pool as files in a data lake. Which SQL statement should you use? *

BULK INSERT

CREATE EXTERNAL TABLE AS SELECT

✓ Correct. CREATE EXTERNAL TABLE AS SELECT enables you to transform your data using Transact-SQL and store the query results in a data lake.

COPY

2. Which of the following file formats can you use to persist the results of a query? *

CSV only

Parquet only.

✗ Incorrect. Parquet is not the only supported file format.

CSV and Parquet.

✓ Correct. You can store files for an external table in CSV or Parquet format (as well as other formats).

3. You drop an existing external table from a database in a serverless SQL pool. What else must you do before recreating an external table with the same location? *

Delete the folder containing the data files for dropped table.

✓ Correct. Dropping an external table does not delete the underlying files.

Drop and recreate the database.

Create an Apache Spark pool.

Next unit: Summary

[Continue >](#)



Summary

1 minute

By using the CREATE EXTERNAL TABLE AS statement, you can use Azure Synapse serverless SQL pool to transform data as part of a data ingestion pipeline or an extract, transform, and load (ETL) process. The transformed data is persisted in files in the data lake with a relational table based on the file location; enabling you to work with the transformed data using SQL in the serverless SQL database, or directly in the file data lake.

In this lesson, you learned how to:

- Use a CREATE EXTERNAL TABLE AS SELECT (CETAS) statement to transform data.
- Encapsulate a CETAS statement in a stored procedure.
- Include a data transformation stored procedure in a pipeline.

💡 Tip

For more information about using the CETAS statement, see [CETAS with Synapse SQL](#) in the Azure Synapse Analytics documentation.

Module complete:

Unlock achievement

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

1 minute

Data analysts and engineers often find themselves forced to choose between the flexibility of storing data files in a data lake, with the advantages of a structured schema in a relational database. *Lake databases* in Azure Synapse Analytics provide a way to combine these two approaches and benefit from an explicit relational schema of tables, views, and relationships that is decoupled from file-based storage.

In this module, you'll learn how to:

- Understand lake database concepts and components
- Describe database templates in Azure Synapse Analytics
- Create a lake database

Prerequisites

Before starting this module, you should have the following prerequisite skills and knowledge:

- Familiarity with the Microsoft Azure portal
- Familiarity with data lake and data warehouse concepts
- Experience of using SQL to query database tables

Next unit: Understand lake database concepts

[Continue >](#)

How are we doing?

✓ 100 XP



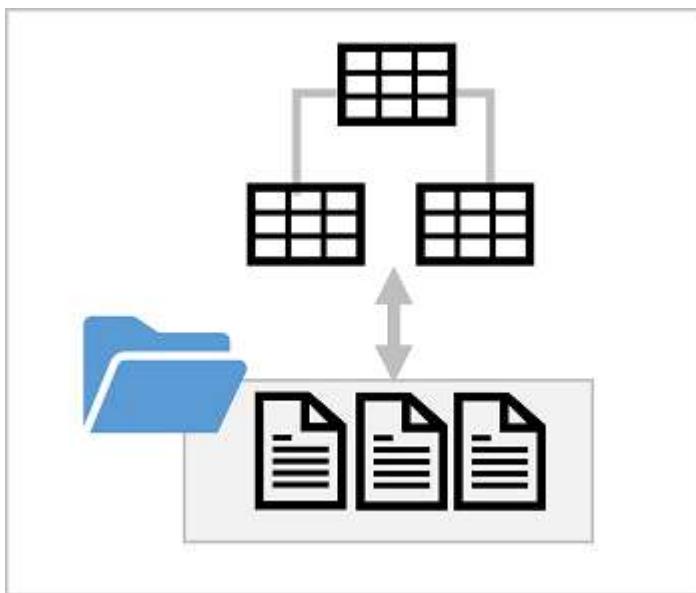
Understand lake database concepts

3 minutes

In a traditional relational database, the database schema is composed of tables, views, and other objects. Tables in a relational database define the entities for which data is stored - for example, a retail database might include tables for products, customers, and orders. Each entity consists of a set of attributes that are defined as columns in the table, and each column has a name and a data type. The data for the tables is stored in the database, and is tightly coupled to the table definition; which enforces data types, nullability, key uniqueness, and referential integrity between related keys. All queries and data manipulations must be performed through the database system.

In a data lake, there is no fixed schema. Data is stored in files, which may be structured, semi-structured, or unstructured. Applications and data analysts can work directly with the files in the data lake using the tools of their choice; without the constraints of a relational database system.

A *lake database* provides a relational metadata layer over one or more files in a data lake. You can create a lake database that includes definitions for tables, including column names and data types as well as relationships between primary and foreign key columns. The tables reference files in the data lake, enabling you to apply relational semantics to working with the data and querying it using SQL. However, the storage of the data files is decoupled from the database schema; enabling more flexibility than a relational database system typically offers.



Lake database schema

You can create a lake database in Azure Synapse Analytics, and define the tables that represent the entities for which you need to store data. You can apply proven data modeling principles to create relationships between tables and use appropriate naming conventions for tables, columns, and other database objects.

Azure Synapse Analytics includes a graphical database design interface that you can use to model complex database schema, using many of the same best practices for database design that you would apply to a traditional database.

Lake database storage

The data for the tables in your lake database is stored in the data lake as Parquet or CSV files. The files can be managed independently of the database tables, making it easier to manage data ingestion and manipulation with a wide variety of data processing tools and technologies.

Lake database compute

To query and manipulate the data through the tables you have defined, you can use an Azure Synapse serverless SQL pool to run SQL queries or an Azure Synapse Apache Spark pool to work with the tables using the Spark SQL API.

Next unit: Explore database templates

[Continue >](#)

How are we doing?

 100 XP 

Explore database templates

3 minutes

You can create a Lake database from an empty schema, to which you add definitions for tables and the relationships between them. However, Azure Synapse Analytics provides a comprehensive collection of database templates that reflect common schemas found in multiple business scenarios; including:

- Agriculture
- Automotive
- Banking
- Consumer goods
- Energy and commodity trading
- Freight and logistics
- Fund management
- Healthcare insurance
- Healthcare provider
- Manufacturing
- Retail
- *and many others...*

The screenshot shows the Microsoft Azure Synapse Analytics Gallery interface. At the top, there's a navigation bar with links for 'Microsoft Azure', 'Synapse Analytics', and 'synapses1tecy'. Below the navigation is a 'Gallery' section with tabs for 'Database templates', 'Datasets', 'Notebooks', 'SQL scripts', and 'Pipelines'. A search bar is also present. On the left, there's a vertical sidebar with icons for different services. The main area displays nine enterprise database templates in a grid:

- Consumer Goods**: For manufacturers or producers of goods bought and used by consumers.
- Energy & Commodity Trading**: For traders of energy, commodities, or carbon credits.
- Freight & Logistics**: For companies providing freight and logistics services.
- Fund Management**: For companies managing investment funds on behalf of investors.
- Genomics**: For companies acquiring and analyzing genomic data about human beings or other species.
- Healthcare Insurance**: For organizations providing insurance to cover healthcare needs (sometimes known as Payors).
- Healthcare Providers**: For organizations providing
- Life Insurance & Annuities**: For companies who provide life
- Manufacturing**: For companies engaged in discrete

At the bottom of the grid are two buttons: 'Continue' on the left and 'Close' on the right.

You can use one of the enterprise database templates as the starting point for creating your lake database, or you can start with a blank schema and add and modify tables from the templates as required.

Next unit: Create a lake database

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Create a lake database

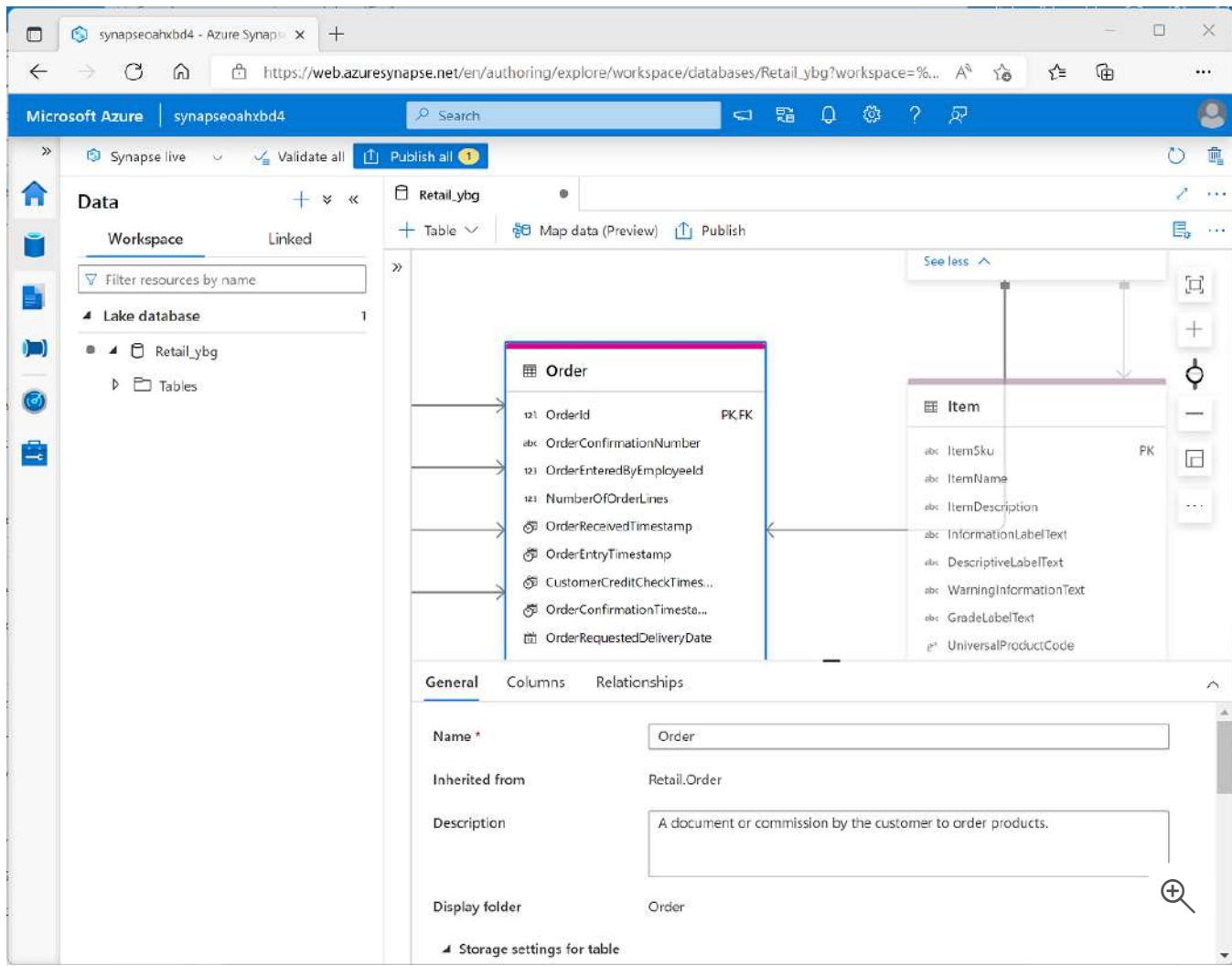
3 minutes

You can create a lake database using the lake database designer in Azure Synapse Studio. Start by adding a new lake database on the **Data** page, selecting a template from the gallery or starting with a blank lake database; and then add and customize tables using the visual database designer interface.

As you create each table, you can specify the type and location of the files you want to use to store the underlying data, or you can create a table from existing files that are already in the data lake. In most cases, it's advisable to store all of the database files in a consistent format within the same root folder in the data lake.

Database designer

The database designer interface in Azure Synapse Studio provides a drag-and-drop surface on which you can edit the tables in your database and the relationships between them.



Using the database designer, you can define the schema for your database by adding or removing tables and:

- Specifying the name and storage settings for each table.
- Specifying the names, key usage, nullability, and data types for each column.
- Defining relationships between key columns in tables.

When your database schema is ready for use, you can publish the database and start using it.

Next unit: Use a lake database

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP ➔

Use a lake database

3 minutes

After creating a lake database, you can store data files that match the table schemas in the appropriate folders in the data lake, and query them using SQL.

Using a serverless SQL pool

You can query a lake database in a SQL script by using a serverless SQL pool.

For example, suppose a lake database named **RetailDB** contains an **Customer** table. You could query it using a standard SELECT statement like this:

SQL

```
USE RetailDB;
GO

SELECT CustomerID, FirstName, LastName
FROM Customer
ORDER BY LastName;
```

There is no need to use an OPENROWSET function or include any additional code to access the data from the underlying file storage. The serverless SQL pool handles the mapping to the files for you.

Using an Apache Spark pool

In addition to using a serverless SQL pool, you can work with lake database tables using Spark SQL in an Apache Spark pool.

For example, you could use the following code to insert a new customer record into the **Customer** table.

SQL

```
%%sql
INSERT INTO `RetailDB`.`Customer` VALUES (123, 'John', 'Yang')
```

You could then use the following code to query the table:

SQL

```
%%sql  
SELECT * FROM `RetailDB`.`Customer` WHERE CustomerID = 123
```

Next unit: Exercise - Analyze data in a lake database

[Continue >](#)

How are we doing?

Knowledge check

5 minutes

1. Which if the following statements is true of a lake database? *

- Data is stored in a relational database store and cannot be directly accessed in the data lake files.
- Data is stored in files that cannot be queried using SQL.
- A relational schema is overlaid on the underlying files, and can be queried using a serverless SQL pool or a Spark pool.

✓ Correct. A lake database abstracts files with a relational schema that can be queried using SQL in a serverless SQL pool or a Spark pool.

2. You need to create a new lake database for a retail solution. What's the most efficient way to do this? *

- Create a sample database in Azure SQL Database and export the SQL scripts to create the schema for the lake database.
- Start with the Retail database template in Azure Synapse Studio, and adapt it as necessary.

✓ Correct. The Gallery in Azure Synapse Studio includes industry-proven database schema templates, including one for retail.

- Start with an empty database and create a normalized schema.

3. You have Parquet files in an existing data lake folder for which you want to create a table in a lake database. What should you do? *

- Use a CREATE EXTERNAL TABLE AS SELECT (CETAS) query to create the table.
- Convert the files in the folder to CSV format.
- Use the database designer to create a table based on the existing folder.

✓ Correct. You can add a table to a database from existing files.

Summary

1 minute

A lake database can provide the benefits of a relational schema and query interface with the flexibility of file storage in a data lake.

In this module, you learned how to:

- Understand lake database concepts and components
- Describe database templates in Azure Synapse Analytics
- Create a lake database

Learn more

To learn more about lake databases, refer to the [Azure Synapse Analytics documentation](#).

Module complete:

Unlock achievement

How are we doing? ★ ★ ★ ★ ★

100 XP

Introduction

3 minutes

In this lesson, you will learn how you can set up security when using Azure Synapse serverless SQL pools

After the completion of this lesson, you will be able to:

- Choose an authentication method in Azure Synapse serverless SQL pools
- Manage users in Azure Synapse serverless SQL pools
- Manage user permissions in Azure Synapse serverless SQL pools

Prerequisites

Before taking this lesson, it is recommended that the student is able to:

- Log into the Azure portal
- Explain the different components of Azure Synapse Analytics
- Use Azure Synapse Studio

Next unit: Choose an authentication method in Azure Synapse serverless SQL pools

[Continue >](#)

How are we doing?

100 XP

Choose an authentication method in Azure Synapse serverless SQL pools

3 minutes

Serverless SQL pool authentication refers to how users prove their identity when connecting to the endpoint. Two types of authentication are supported:

- **SQL Authentication**

This authentication method uses a username and password.

- **Azure Active Directory Authentication**

This authentication method uses identities managed by Azure Active Directory. For Azure AD users, multi-factor authentication can be enabled. Use Active Directory authentication (integrated security) whenever possible.

Authorization

Authorization refers to what a user can do within a serverless SQL pool database and is controlled by your user account's database role memberships and object-level permissions.

If SQL Authentication is used, the SQL user exists only in the serverless SQL pool and permissions are scoped to the objects in the serverless SQL pool. Access to securable objects in other services (such as Azure Storage) can't be granted to a SQL user directly since it only exists in scope of serverless SQL pool. The SQL user needs get authorization to access the files in the storage account.

If Azure Active Directory authentication is used, a user can sign in to a serverless SQL pool and other services, like Azure Storage, and can grant permissions to the Azure Active Directory user.

Access to storage accounts

A user that is logged into the serverless SQL pool service must be authorized to access and query the files in Azure Storage. Serverless SQL pool supports the following authorization types:

- Anonymous access

To access publicly available files placed on Azure storage accounts that allow anonymous access.

- Shared access signature (SAS)

Provides delegated access to resources in storage account. With a SAS, you can grant clients access to resources in storage account, without sharing account keys. A SAS gives you granular control over the type of access you grant to clients who have the SAS: validity interval, granted permissions, acceptable IP address range, acceptable protocol (https/http).

- Managed Identity.

Is a feature of Azure Active Directory (Azure AD) that provides Azure services for serverless SQL pool. Also, it deploys an automatically managed identity in Azure AD. This identity can be used to authorize the request for data access in Azure Storage. Before accessing the data, the Azure Storage administrator must grant permissions to Managed Identity for accessing the data. Granting permissions to Managed Identity is done the same way as granting permission to any other Azure AD user.

- User Identity

Also known as "pass-through", is an authorization type where the identity of the Azure AD user that logged into serverless SQL pool is used to authorize access to the data. Before accessing the data, Azure Storage administrator must grant permissions to Azure AD user for accessing the data. This authorization type uses the Azure AD user that logged into serverless SQL pool, therefore it's not supported for SQL user types.

Supported authorization types for database users can be found in the table below:

Authorization type	SQL user	Azure AD user
User Identity	Not supported	Supported
SAS	Supported	Supported
Managed Identity	Not supported	Supported

Supported storage and authorization types can be found in the table below:

Authorization type	Blob Storage	ADLS Gen1	ADLS Gen2
User Identity	Supported - SAS token can be used to access storage that is not protected with firewall	Not supported	Supported - SAS token can be used to access storage that is not protected with firewall
SAS	Supported	Supported	Supported
Managed Identity	Supported	Supported	Supported

Next unit: Manage users in Azure Synapse serverless SQL pools

[Continue >](#)

How are we doing?

Manage users in Azure Synapse serverless SQL pools

3 minutes

You can give administrator privileges to a user to Azure Synapse serverless SQL pool. To do this you should open the Azure Synapse workspace and do the following steps:

1. Go to **Manage** menu

2. Go to **Access control**

3. Click on **Add**

Access control

Grant others access to this workspace by assigning roles to users, groups, and/or service principals. [Learn more](#) 



4. Choose **Synapse Administrator**

Add role assignment

Grant others access to this workspace by assigning roles to users, groups, and/or service principals.
[Learn more](#)

Scope * ⓘ

Workspace Workspace item

Role * ⓘ

Synapse Administrator



Filter...

Synapse Administrator ⓘ

Synapse SQL Administrator ⓘ

Synapse Apache Spark Administrator ⓘ

Synapse Contributor (preview) ⓘ

Synapse Artifact Publisher (preview) ⓘ

Synapse Artifact User (preview) ⓘ

Synapse Compute Operator (preview) ⓘ

Synapse Credential User (preview) ⓘ



5. Select a User or Security group (a security group is the recommended option here)

6. Click **Apply**

Now this user or group is the administrator of the Azure Synapse workspace and serverless SQL pool.

Next unit: Manage user permissions in Azure Synapse serverless SQL pools

[Continue >](#)

How are we doing? ⭐ ⭐ ⭐ ⭐ ⭐

✓ 100 XP



Manage user permissions in Azure Synapse serverless SQL pools

3 minutes

To secure data, Azure Storage implements an access control model that supports both Azure role-based access control (Azure RBAC) and access control lists (ACLs) like Portable Operating System Interface for Unix (POSIX)

You can associate a security principal with an access level for files and directories. These associations are captured in an access control list (ACL). Each file and directory in your storage account has an access control list. When a security principal attempts an operation on a file or directory, an ACL check determines whether that security principal (user, group, service principal, or managed identity) has the correct permission level to perform the operation.

There are two kinds of access control lists:

- **Access ACLs**

Controls access to an object. Files and directories both have access ACLs.

- **Default ACLs**

Are templates of ACLs associated with a directory that determine the access ACLs for any child items that are created under that directory. Files do not have default ACLs.

Both access ACLs and default ACLs have the same structure.

The permissions on a container object are Read, Write, and Execute, and they can be used on files and directories as shown in the following table:

Levels of permissions

Permission	File	Directory
Read (R)	Can read the contents of a file	Requires Read and Execute to list the contents of the directory
Write (W)	Can write or append to a file	Requires Write and Execute to create child items in a directory

Permission	File	Directory
Execute (X)	Does not mean anything in the context of Data Lake Storage Gen2	Required to traverse the child items of a directory

Guidelines in setting up ACLs

Always use Azure Active Directory security groups as the assigned principal in an ACL entry. Resist the opportunity to directly assign individual users or service principals. Using this structure will allow you to add and remove users or service principals without the need to reapply ACLs to an entire directory structure. Instead, you can just add or remove users and service principals from the appropriate Azure AD security group.

There are many ways to set up groups. For example, imagine that you have a directory named **/LogData** which holds log data that is generated by your server. Azure Data Factory (ADF) ingests data into that folder. Specific users from the service engineering team will upload logs and manage other users of this folder, and various Databricks clusters will analyze logs from that folder.

To enable these activities, you could create a LogsWriter group and a LogsReader group. Then, you could assign permissions as follows:

- Add the LogsWriter group to the ACL of the **/LogData** directory with rwx permissions.
- Add the LogsReader group to the ACL of the **/LogData** directory with r-x permissions.
- Add the service principal object or Managed Service Identity (MSI) for ADF to the LogsWriters group.
- Add users in the service engineering team to the LogsWriter group.
- Add the service principal object or MSI for Databricks to the LogsReader group.

If a user in the service engineering team leaves the company, you could just remove them from the LogsWriter group. If you did not add that user to a group, but instead, you added a dedicated ACL entry for that user, you would have to remove that ACL entry from the **/LogData** directory. You would also have to remove the entry from all subdirectories and files in the entire directory hierarchy of the **/LogData** directory.

Roles necessary for serverless SQL pool users

For users which need **read only** access you should assign role named **Storage Blob Data Reader**.

For users which need **read/write** access you should assign role named **Storage Blob Data Contributor**. Read/Write access is needed if user should have access to create external table as select (CETAS).

! Note

If user has a role Owner or Contributor, that role is not enough. Azure Data Lake Storage gen 2 has super-roles which should be assigned.

Database level permission

To provide more granular access to the user, you should use Transact-SQL syntax to create logins and users.

To grant access to a user to a single serverless SQL pool database, follow the steps in this example:

1. Create LOGIN

SQL

```
use master  
CREATE LOGIN [alias@domain.com] FROM EXTERNAL PROVIDER;
```

2. Create USER

SQL

```
use yourdb -- Use your DB name  
CREATE USER alias FROM LOGIN [alias@domain.com];
```

3. Add USER to members of the specified role

SQL

```
use yourdb -- Use your DB name  
alter role db_datareader  
Add member alias -- Type USER name from step 2  
-- You can use any Database Role which exists  
-- (examples: db_owner, db_datareader, db_datawriter)  
-- Replace alias with alias of the user you would like to give access  
and domain with the company domain you are using.
```

Server level permission

1. To grant full access to a user to all serverless SQL pool databases, follow the step in this example:

SQL

```
CREATE LOGIN [alias@domain.com] FROM EXTERNAL PROVIDER;
ALTER SERVER ROLE sysadmin ADD MEMBER [alias@domain.com];
```

Next unit: Knowledge check

[Continue >](#)

How are we doing?



Knowledge check

3 minutes

1. Which authentication method would be the likeliest choice to use for an individual who needs to access your serverless SQL pool who works for an external organization? *

Local authentication.

SQL Authentication.

✓ Correct. SQL Authentication uses an authentication method of a username and password stored within the serverless SQL pool.

Azure Active Directory.

✗ Incorrect. Azure Active Directory uses identities managed by Azure Active Directory.

2. Which Azure Synapse Studio hub is where you assign administrator privileges to an Azure Synapse workspace? *

Manage.

✓ Correct. The manage hub is the area of Azure Synapse Studio where you assign administrator privileges to an Azure Synapse workspace.

Data.

Develop.

3. Which role enables a user to create external table as select (CETAS) against an Azure Data Lake Gen2 data store? *

Storage Blob Data Reader.

Storage Blob Data Contributor.

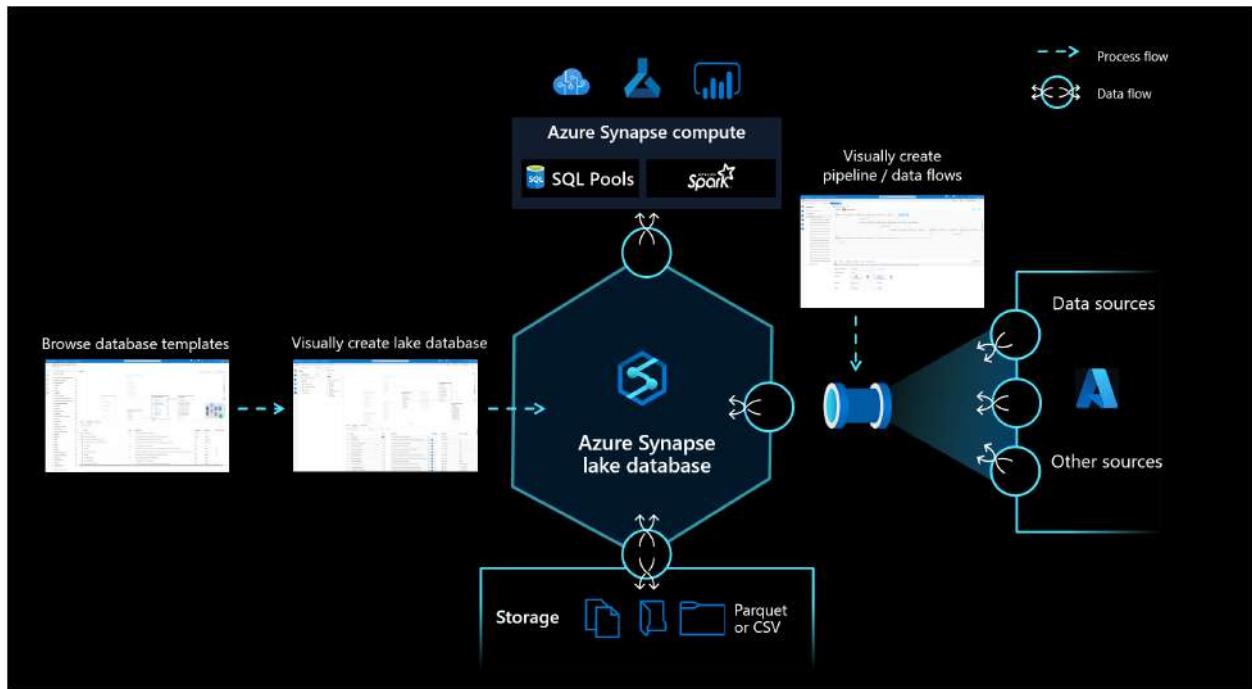
✓ Correct. This provides the read/write access required to execute a create external table as select statement.

Executor.

Lake database

Article • 03/04/2023

The lake database in Azure Synapse Analytics enables customers to bring together database design, meta information about the data that is stored and a possibility to describe how and where the data should be stored. Lake database addresses the challenge of today's data lakes where it is hard to understand how data is structured.



Database designer

The new database designer in Synapse Studio gives you the possibility to create a data model for your lake database and add additional information to it. Every Entity and Attribute can be described to provide more information about the model, which not only contains Entities but relationships as well. In particular, the inability to model relationships has been a challenge for the interaction on the data lake. This challenge is now addressed with an integrated designer that provides possibilities that have been available in databases but not on the lake. Also the capability to add descriptions and possible demo values to the model allows people who are interacting with it in the future to have information where they need it to get a better understanding about the data.

Data storage

Lake databases use a data lake on the Azure Storage account to store the data of the database. The data can be stored in Parquet, Delta or CSV format and different settings

can be used to optimize the storage. Every lake database uses a linked service to define the location of the root data folder. For every entity, separate folders are created by default within this database folder on the data lake. By default all tables within a lake database use the same format but the formats and location of the data can be changed per entity if that is requested.

Note

Publishing a lake database does not create any of the underlying structures or schemas needed to query the data in Spark or SQL. After publishing, load data into your lake database using **pipelines** to begin querying it.

Currently, Delta format support for lake databases is not supported in Synapse Studio.

The synchronization of lake database objects between storage and Synapse is one-directional. Be sure to perform any creation or schema modification of lake database objects using the database designer in Synapse Studio. If you instead make such changes from Spark or directly in storage, the definitions of your lake databases will become out of sync. If this happens, you may see old lake database definitions in the database designer. You will need to replicate and publish such changes in the database designer in order to bring your lake databases back in sync.

Database compute

The lake database is exposed in Synapse SQL serverless SQL pool and Apache Spark providing users with the capability to decouple storage from compute. The metadata that is associated with the lake database makes it easy for different compute engines to not only provide an integrated experience but also use additional information (for example, relationships) that was not originally supported on the data lake.

Next steps

Continue to explore the capabilities of the database designer using the links below.

- [Create lake database quick start](#)
- [Database templates Concept](#)

100 XP

Introduction

1 minute

Apache Spark is an open source parallel processing framework for large-scale data processing and analytics. Spark has become extremely popular in "big data" processing scenarios, and is available in multiple platform implementations; including Azure HDInsight, Azure Databricks, and Azure Synapse Analytics.

This module explores how you can use Spark in Azure Synapse Analytics to ingest, process, and analyze data from a data lake. While the core techniques and code described in this module are common to all Spark implementations, the integrated tools and ability to work with Spark in the same environment as other Synapse analytical runtimes are specific to Azure Synapse Analytics.

After completing this module, you'll be able to:

- Identify core features and capabilities of Apache Spark.
- Configure a Spark pool in Azure Synapse Analytics.
- Run code to load, analyze, and visualize data in a Spark notebook.

Next unit: Get to know Apache Spark

[Continue >](#)

How are we doing?

✓ 100 XP



Get to know Apache Spark

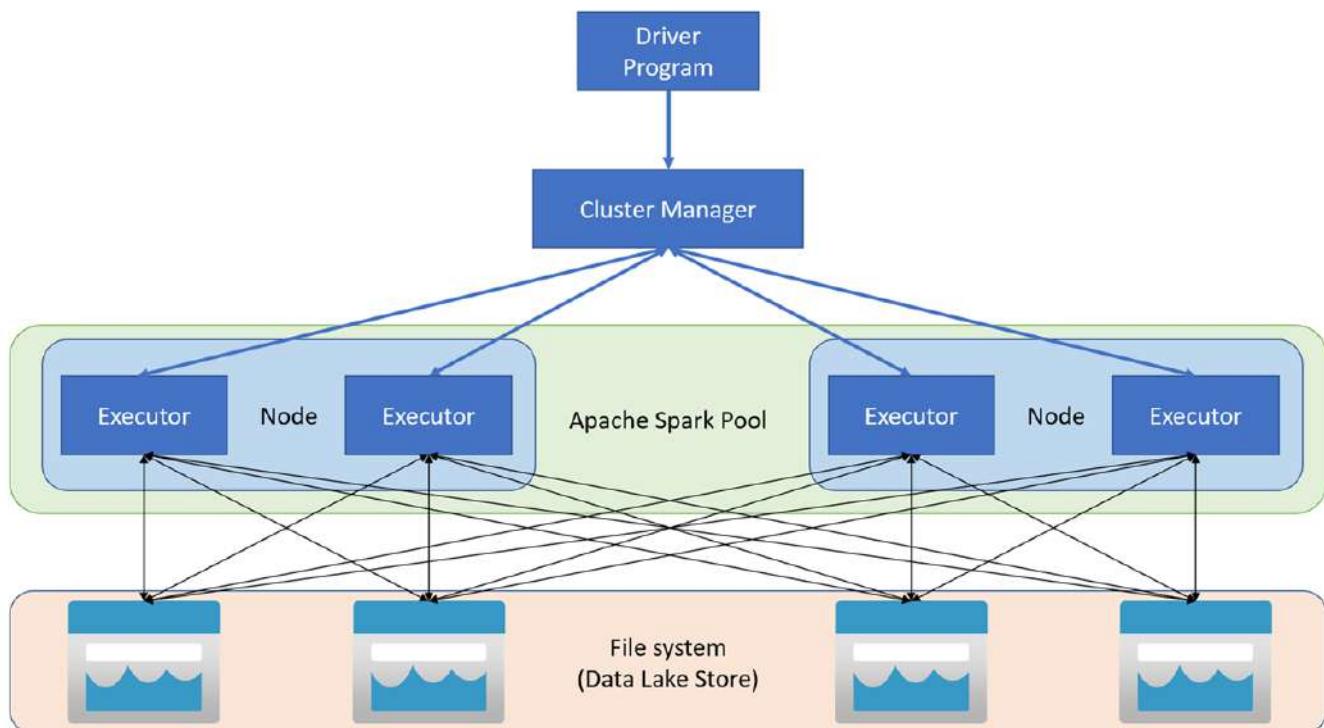
3 minutes

Apache Spark is distributed data processing framework that enables large-scale data analytics by coordinating work across multiple processing nodes in a cluster.

How Spark works

Apache Spark applications run as independent sets of processes on a cluster, coordinated by the *SparkContext* object in your main program (called the driver program). The *SparkContext* connects to the cluster manager, which allocates resources across applications using an implementation of Apache Hadoop YARN. Once connected, Spark acquires executors on nodes in the cluster to run your application code.

The *SparkContext* runs the main function and parallel operations on the cluster nodes, and then collects the results of the operations. The nodes read and write data from and to the file system and cache transformed data in-memory as *Resilient Distributed Datasets* (RDDs).



The *SparkContext* is responsible for converting an application to a *directed acyclic graph* (DAG). The graph consists of individual tasks that get executed within an executor process on the nodes. Each application gets its own executor processes, which stay up for the duration of the whole application and run tasks in multiple threads.

Spark pools in Azure Synapse Analytics

In Azure Synapse Analytics, a cluster is implemented as a *Spark pool*, which provides a runtime for Spark operations. You can create one or more Spark pools in an Azure Synapse Analytics workspace [by using the Azure portal](#), or [in Azure Synapse Studio](#). When defining a Spark pool, you can specify configuration options for the pool, including:

- A name for the spark pool.
- The size of virtual machine (VM) used for the nodes in the pool, including the option to use [hardware accelerated GPU-enabled nodes](#).
- The number of nodes in the pool, and whether the pool size is fixed or individual nodes can be brought online dynamically to *auto-scale* the cluster; in which case, you can specify the minimum and maximum number of active nodes.
- The version of the *Spark Runtime* to be used in the pool; which dictates the versions of individual components such as Python, Java, and others that get installed.

💡 Tip

For more information about Spark pool configuration options, see [Apache Spark pool configurations in Azure Synapse Analytics](#) in the Azure Synapse Analytics documentation.

Spark pools in an Azure Synapse Analytics Workspace are *serverless* - they start on-demand and stop when idle.

Next unit: Use Spark in Azure Synapse Analytics

[Continue >](#)

How are we doing?

100 XP

Use Spark in Azure Synapse Analytics

3 minutes

You can run many different kinds of application on Spark, including code in Python or Scala scripts, Java code compiled as a Java Archive (JAR), and others. Spark is commonly used in two kinds of workload:

- Batch or stream processing jobs to ingest, clean, and transform data - often running as part of an automated pipeline.
- Interactive analytics sessions to explore, analyze, and visualize data.

Running Spark code in notebooks

Azure Synapse Studio includes an integrated notebook interface for working with Spark. Notebooks provide an intuitive way to combine code with Markdown notes, commonly used by data scientists and data analysts. The look and feel of the integrated notebook experience within Azure Synapse Studio is similar to that of Jupyter notebooks - a popular open source notebook platform.

The screenshot shows the Azure Synapse Analytics workspace interface. On the left, there's a sidebar with icons for Home, Develop, Notebooks, Datasets, Pipelines, and Triggers. The main area is titled "Sales Notebook". It displays a code cell with the following Python code:

```
1  from pyspark.sql.types import *
2  from pyspark.sql.functions import *
3
4  orderSchema = StructType([
5    StructField("SalesOrderNumber", StringType()),
6    StructField("SalesOrderLineNumber", IntegerType()),
7    StructField("OrderDate", DateType()),
8    StructField("CustomerName", StringType()),
9    StructField("Email", StringType()),
10   StructField("Item", StringType()),
11   StructField("Quantity", IntegerType()),
12   StructField("UnitPrice", FloatType()),
13   StructField("Tax", FloatType())
14 ])
15
16 df = spark.read.load('abfss://files@datalake8xuhd0y.dfs.core.windows.net/sales/or
17   format='csv',
18   schema=orderSchema,
19 )
20 display(df.limit(10))
```

Below the code cell, a message indicates the command was executed successfully: "✓ 2 sec - Command executed in 2 sec 828 ms by graemesplace on 9:21:57 AM, 5/23/22". The notebook also shows job execution details: "Job execution Succeeded" with "Spark 2 executors 8 cores". There are buttons for "View in monitoring" and "Open Spark UI". At the bottom, there are tabs for "View", "Table" (which is selected), "Chart", and "Export results". A preview table shows the first two rows of the data:

SalesOrderNumber	SalesOrderLineNumber	OrderDate	CustomerName	Email
SO49171	1	2021-01-01	Mariah Foster	mariah21@adv...
SO49172	1	2021-01-01	Brian Howard	brian23@adv...

! Note

While usually used interactively, notebooks can be included in automated pipelines and run as an unattended script.

Notebooks consist of one or more *cells*, each containing either code or markdown. Code cells in notebooks have some features that can help you be more productive, including:

- Syntax highlighting and error support.
- Code auto-completion.
- Interactive data visualizations.
- The ability to export results.

💡 Tip

To learn more about working with notebooks in Azure Synapse Analytics, see the [Create, develop, and maintain Synapse notebooks in Azure Synapse Analytics](#) article in the Azure Synapse Analytics documentation.

Accessing data from a Synapse Spark pool

You can use Spark in Azure Synapse Analytics to work with data from various sources, including:

- A data lake based on the primary storage account for the Azure Synapse Analytics workspace.
- A data lake based on storage defined as a *linked service* in the workspace.
- A dedicated or serverless SQL pool in the workspace.
- An Azure SQL or SQL Server database (using the Spark connector for SQL Server)
- An Azure Cosmos DB analytical database defined as a *linked service* and configured using *Azure Synapse Link for Cosmos DB*.
- An Azure Data Explorer Kusto database defined as a *linked service* in the workspace.
- An external Hive metastore defined as a *linked service* in the workspace.

One of the most common uses of Spark is to work with data in a data lake, where you can read and write files in multiple commonly used formats, including delimited text, Parquet, Avro, and others.

Next unit: Analyze data with Spark

[Continue >](#)

How are we doing? 

✓ 100 XP



Analyze data with Spark

5 minutes

One of the benefits of using Spark is that you can write and run code in various programming languages, enabling you to use the programming skills you already have and to use the most appropriate language for a given task. The default language in a new Azure Synapse Analytics Spark notebook is *PySpark* - a Spark-optimized version of Python, which is commonly used by data scientists and analysts due to its strong support for data manipulation and visualization. Additionally, you can use languages such as *Scala* (a Java-derived language that can be used interactively) and *SQL* (a variant of the commonly used SQL language included in the *Spark SQL* library to work with relational data structures). Software engineers can also create compiled solutions that run on Spark using frameworks such as *Java* and *Microsoft .NET*.

Exploring data with dataframes

Natively, Spark uses a data structure called a *resilient distributed dataset* (RDD); but while you *can* write code that works directly with RDDs, the most commonly used data structure for working with structured data in Spark is the *dataframe*, which is provided as part of the *Spark SQL* library. Dataframes in Spark are similar to those in the ubiquitous *Pandas* Python library, but optimized to work in Spark's distributed processing environment.

⚠ Note

In addition to the Dataframe API, Spark SQL provides a strongly-typed *Dataset* API that is supported in Java and Scala. We'll focus on the Dataframe API in this module.

Loading data into a dataframe

Let's explore a hypothetical example to see how you can use a dataframe to work with data. Suppose you have the following data in a comma-delimited text file named **products.csv** in the primary storage account for an Azure Synapse Analytics workspace:

csv

ProductID	ProductName	Category	ListPrice
771	"Mountain-100 Silver, 38"	Mountain Bikes	3399.9900
772	"Mountain-100 Silver, 42"	Mountain Bikes	3399.9900

```
773,"Mountain-100 Silver, 44",Mountain Bikes,3399.9900
```

```
...
```

In a Spark notebook, you could use the following PySpark code to load the data into a dataframe and display the first 10 rows:

Python

```
%%pyspark
df = spark.read.load('abfss://container@store.dfs.core.windows.net/product-
s.csv',
    format='csv',
    header=True
)
display(df.limit(10))
```

The `%%pyspark` line at the beginning is called a *magic*, and tells Spark that the language used in this cell is PySpark. You can select the language you want to use as a default in the toolbar of the Notebook interface, and then use a magic to override that choice for a specific cell. For example, here's the equivalent Scala code for the products data example:

Scala

```
%%spark
val df = spark.read.format("csv").option("header",
"true").load("abfss://container@store.dfs.core.windows.net/products.csv")
display(df.limit(10))
```

The magic `%%spark` is used to specify Scala.

Both of these code samples would produce output like this:

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
...

Specifying a dataframe schema

In the previous example, the first row of the CSV file contained the column names, and Spark was able to infer the data type of each column from the data it contains. You can also specify an explicit schema for the data, which is useful when the column names aren't included in the data file, like this CSV example:

CSV

```
771,"Mountain-100 Silver, 38",Mountain Bikes,3399.9900
772,"Mountain-100 Silver, 42",Mountain Bikes,3399.9900
773,"Mountain-100 Silver, 44",Mountain Bikes,3399.9900
...
```

The following PySpark example shows how to specify a schema for the dataframe to be loaded from a file named **product-data.csv** in this format:

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

productSchema = StructType([
    StructField("ProductID", IntegerType()),
    StructField("ProductName", StringType()),
    StructField("Category", StringType()),
    StructField("ListPrice", FloatType())
])

df = spark.read.load('abfss://container@store.dfs.core.windows.net/product-
data.csv',
    format='csv',
    schema=productSchema,
    header=False)
display(df.limit(10))
```

The results would once again be similar to:

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900

ProductID	ProductName	Category	ListPrice
...
...

Filtering and grouping dataframes

You can use the methods of the Dataframe class to filter, sort, group, and otherwise manipulate the data it contains. For example, the following code example uses the `select` method to retrieve the `ProductName` and `ListPrice` columns from the `df` dataframe containing product data in the previous example:

Python

```
pricelist_df = df.select("ProductID", "ListPrice")
```

The results from this code example would look something like this:

ProductID	ListPrice
771	3399.9900
772	3399.9900
773	3399.9900
...	...

In common with most data manipulation methods, `select` returns a new dataframe object.

💡 Tip

Selecting a subset of columns from a dataframe is a common operation, which can also be achieved by using the following shorter syntax:

```
pricelist_df = df["ProductID", "ListPrice"]
```

You can "chain" methods together to perform a series of manipulations that results in a transformed dataframe. For example, this example code chains the `select` and `where` methods

to create a new dataframe containing the **ProductName** and **ListPrice** columns for products with a category of **Mountain Bikes** or **Road Bikes**:

```
Python
```

```
bikes_df = df.select("ProductName",  
"ListPrice").where((df["Category"]=="Mountain Bikes") |  
(df["Category"]=="Road Bikes"))  
display(bikes_df)
```

The results from this code example would look something like this:

ProductName	ListPrice
Mountain-100 Silver, 38	3399.9900
Road-750 Black, 52	539.9900
...	...

To group and aggregate data, you can use the **groupBy** method and aggregate functions. For example, the following PySpark code counts the number of products for each category:

```
Python
```

```
counts_df = df.select("ProductID", "Category").groupBy("Category").count()  
display(counts_df)
```

The results from this code example would look something like this:

Category	count
Headsets	3
Wheels	14
Mountain Bikes	32
...	...

Using SQL expressions in Spark

The Dataframe API is part of a Spark library named Spark SQL, which enables data analysts to use SQL expressions to query and manipulate data.

Creating database objects in the Spark catalog

The Spark catalog is a metastore for relational data objects such as views and tables. The Spark runtime can use the catalog to seamlessly integrate code written in any Spark-supported language with SQL expressions that may be more natural to some data analysts or developers.

One of the simplest ways to make data in a dataframe available for querying in the Spark catalog is to create a temporary view, as shown in the following code example:

Python

```
df.createOrReplaceTempView("products")
```

A *view* is temporary, meaning that it's automatically deleted at the end of the current session. You can also create *tables* that are persisted in the catalog to define a database that can be queried using Spark SQL.

ⓘ Note

We won't explore Spark catalog tables in depth in this module, but it's worth taking the time to highlight a few key points:

- You can create an empty table by using the `spark.catalog.createTable` method. Tables are metadata structures that store their underlying data in the storage location associated with the catalog. Deleting a table also deletes its underlying data.
- You can save a dataframe as a table by using its `saveAsTable` method.
- You can create an *external* table by using the `spark.catalog.createExternalTable` method. External tables define metadata in the catalog but get their underlying data from an external storage location; typically a folder in a data lake. Deleting an external table does not delete the underlying data.

Using the Spark SQL API to query data

You can use the Spark SQL API in code written in any language to query data in the catalog. For example, the following PySpark code uses a SQL query to return data from the **products** view as a dataframe.

Python

```
bikes_df = spark.sql("SELECT ProductID, ProductName, ListPrice \
                      FROM products \
                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")  
display(bikes_df)
```

The results from the code example would look similar to the following table:

ProductID	ProductName	ListPrice
38	Mountain-100 Silver, 38	3399.9900
52	Road-750 Black, 52	539.9900
...

Using SQL code

The previous example demonstrated how to use the Spark SQL API to embed SQL expressions in Spark code. In a notebook, you can also use the `%%sql` magic to run SQL code that queries objects in the catalog, like this:

SQL

```
%%sql  
  
SELECT Category, COUNT(ProductID) AS ProductCount  
FROM products  
GROUP BY Category  
ORDER BY Category
```

The SQL code example returns a resultset that is automatically displayed in the notebook as a table, like the one below:

Category	ProductCount
Bib-Shorts	3

Category	ProductCount
Bike Racks	1
Bike Stands	1
...	...

Next unit: Visualize data with Spark

[Continue >](#)

How are we doing? ★ ★ ★ ★ ★

✓ 100 XP ➔

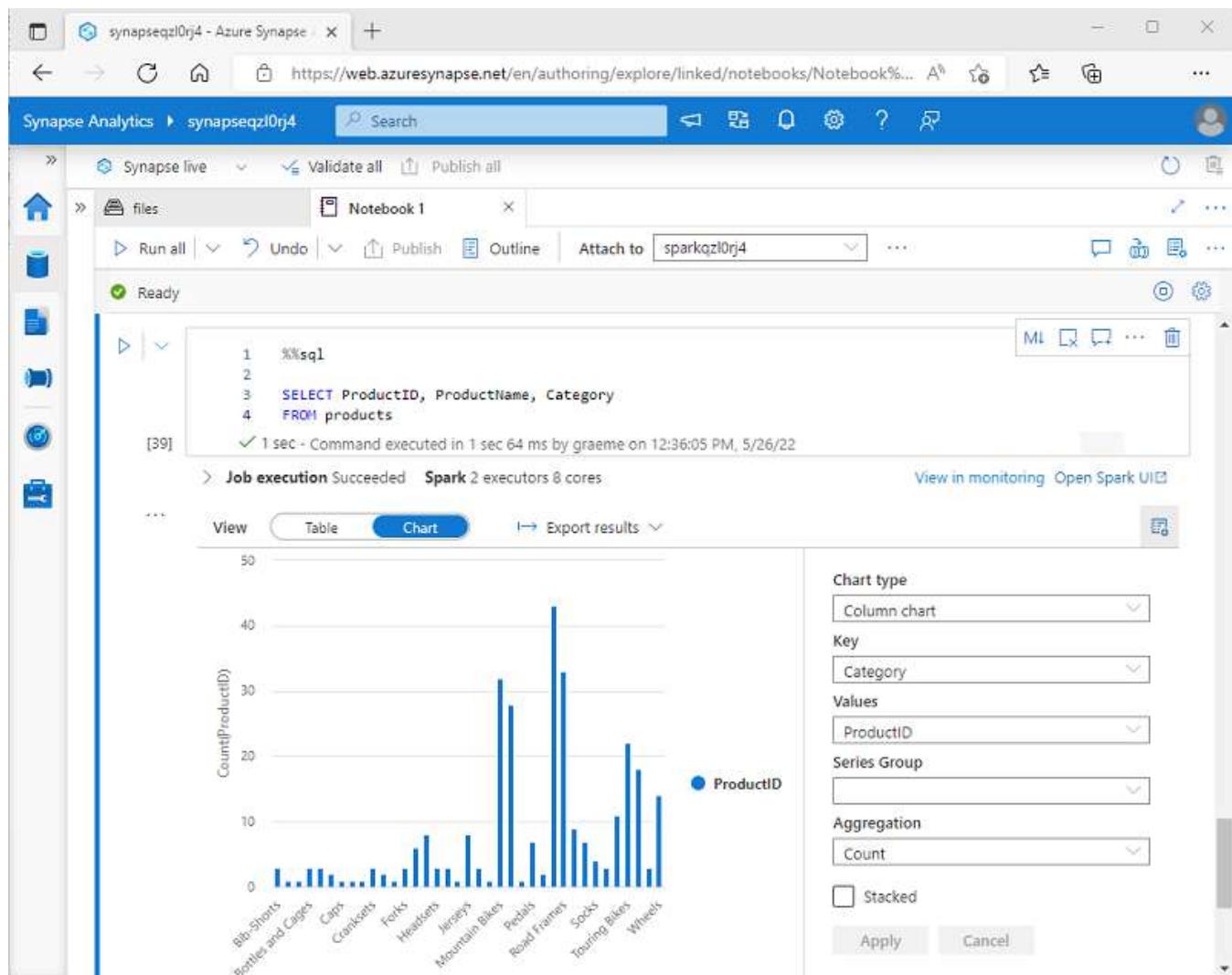
Visualize data with Spark

5 minutes

One of the most intuitive ways to analyze the results of data queries is to visualize them as charts. Notebooks in Azure Synapse Analytics provide some basic charting capabilities in the user interface, and when that functionality doesn't provide what you need, you can use one of the many Python graphics libraries to create and display data visualizations in the notebook.

Using built-in notebook charts

When you display a dataframe or run a SQL query in a Spark notebook in Azure Synapse Analytics, the results are displayed under the code cell. By default, results are rendered as a table, but you can also change the results view to a chart and use the chart properties to customize how the chart visualizes the data, as shown here:



The built-in charting functionality in notebooks is useful when you're working with results of a query that don't include any existing groupings or aggregations, and you want to quickly summarize the data visually. When you want to have more control over how the data is formatted, or to display values that you have already aggregated in a query, you should consider using a graphics package to create your own visualizations.

Using graphics packages in code

There are many graphics packages that you can use to create data visualizations in code. In particular, Python supports a large selection of packages; most of them built on the base **Matplotlib** library. The output from a graphics library can be rendered in a notebook, making it easy to combine code to ingest and manipulate data with inline data visualizations and markdown cells to provide commentary.

For example, you could use the following PySpark code to aggregate data from the hypothetical products data explored previously in this module, and use Matplotlib to create a chart from the aggregated data.

Python

```
from matplotlib import pyplot as plt

# Get the data as a Pandas dataframe
data = spark.sql("SELECT Category, COUNT(ProductID) AS ProductCount \
                  FROM products \
                  GROUP BY Category \
                  ORDER BY Category").toPandas()

# Clear the plot area
plt.clf()

# Create a Figure
fig = plt.figure(figsize=(12,8))

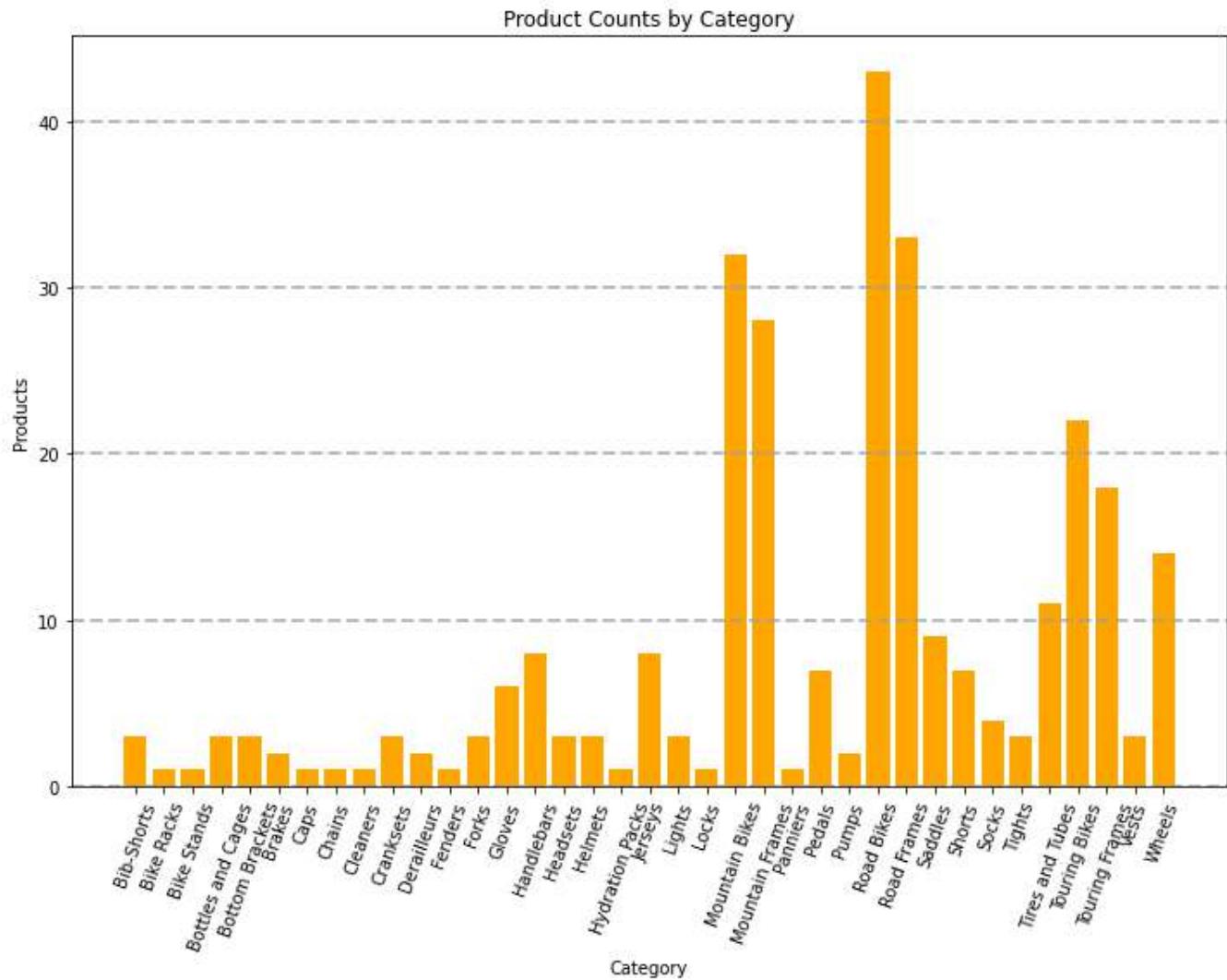
# Create a bar plot of product counts by category
plt.bar(x=data['Category'], height=data['ProductCount'], color='orange')

# Customize the chart
plt.title('Product Counts by Category')
plt.xlabel('Category')
plt.ylabel('Products')
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
plt.xticks(rotation=70)

# Show the plot area
plt.show()
```

The Matplotlib library requires data to be in a Pandas dataframe rather than a Spark dataframe, so the `toPandas` method is used to convert it. The code then creates a figure with a specified size and plots a bar chart with some custom property configuration before showing the resulting plot.

The chart produced by the code would look similar to the following image:



You can use the Matplotlib library to create many kinds of chart; or if preferred, you can use other libraries such as Seaborn to create highly customized charts.

Next unit: Exercise - Analyze data with Spark

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

3 minutes

Check your knowledge

1. Which definition best describes Apache Spark? *

A highly scalable relational database management system.

A virtual server with a Python runtime.

A distributed platform for parallel data processing using multiple languages.

✓ Correct. Spark provides a highly scalable distributed platform on which you can run code written in many languages to process data.

2. You need to use Spark to analyze data in a parquet file. What should you do? *

Load the parquet file into a dataframe.

✓ Correct. You can load data from files in many formats, including parquet, into a Spark dataframe.

Import the data into a table in a serverless SQL pool.

Convert the data to CSV format.

3. You want to write code in a notebook cell that uses a SQL query to retrieve data from a view in the Spark catalog. Which magic should you use? *

%%spark

%%pyspark

✗ Incorrect. The %%pyspark magic instructs Spark to interpret the code in the cell as Python.

%%sql

✓ Correct. The %%sql magic instructs Spark to interpret the code in the cell as SQL.

Next unit: Summary



Summary

1 minute

Apache Spark is a key technology used in big data analytics, and the Spark pool support in Azure Synapse Analytics enables you to combine big data processing in Spark with large-scale data warehousing in SQL.

In this module, you learned how to:

- Identify core features and capabilities of Apache Spark.
- Configure a Spark pool in Azure Synapse Analytics.
- Run code to load, analyze, and visualize data in a Spark notebook.

Module complete:

[Unlock achievement](#)

How are we doing?

100 XP

Introduction

1 minute

Apache Spark provides a powerful platform for performing data cleansing and transformation tasks on large volumes of data. By using the Spark *dataframe* object, you can easily load data from files in a data lake and perform complex modifications. You can then save the transformed data back to the data lake for downstream processing or ingestion into a data warehouse.

Azure Synapse Analytics provides Apache Spark pools that you can use to run Spark workloads to transform data as part of a data ingestion and preparation workload. You can use natively supported notebooks to write and run code on a Spark pool to prepare data for analysis. You can then use other Azure Synapse Analytics capabilities such as SQL pools to work with the transformed data.

Next unit: Modify and save dataframes

[Continue >](#)

How are we doing?

✓ 100 XP

Modify and save dataframes

5 minutes

Apache Spark provides the *dataframe* object as the primary structure for working with data. You can use dataframes to query and transform data, and persist the results in a data lake. To load data into a dataframe, you use the `spark.read` function, specifying the file format, path, and optionally the schema of the data to be read. For example, the following code loads data from all .csv files in the `orders` folder into a dataframe named `order_details` and then displays the first five records.

Python

```
order_details = spark.read.csv('/orders/*.csv', header=True,
inferSchema=True)
display(order_details.limit(5))
```

Transform the data structure

After loading the source data into a dataframe, you can use the `dataframe` object's methods and Spark functions to transform it. Typical operations on a `dataframe` include:

- Filtering rows and columns
- Renaming columns
- Creating new columns, often derived from existing ones
- Replacing null or other values

In the following example, the code uses the `split` function to separate the values in the `CustomerName` column into two new columns named `FirstName` and `LastName`. Then it uses the `drop` method to delete the original `CustomerName` column.

Python

```
from pyspark.sql.functions import split, col

# Create the new FirstName and LastName fields
transformed_df = order_details.withColumn("FirstName", split(col("Customer-
Name"), " ").getItem(0)).withColumn("LastName", split(col("CustomerName"),
" ").getItem(1))

# Remove the CustomerName field
transformed_df = transformed_df.drop("CustomerName")
```

```
display(transformed_df.limit(5))
```

You can use the full power of the Spark SQL library to transform the data by filtering rows, deriving, removing, renaming columns, and any applying other required data modifications.

Save the transformed data

After your DataFrame is in the required structure, you can save the results to a supported format in your data lake.

The following code example saves the DataFrame into a *parquet* file in the data lake, replacing any existing file of the same name.

Python

```
transformed_df.write.mode("overwrite").parquet('/transformed_data/orders.-  
parquet')  
print ("Transformed data saved!")
```

ⓘ Note

The Parquet format is typically preferred for data files that you will use for further analysis or ingestion into an analytical store. Parquet is a very efficient format that is supported by most large scale data analytics systems. In fact, sometimes your data transformation requirement may simply be to convert data from another format (such as CSV) to Parquet!

Next unit: Partition data files

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

Partition data files

5 minutes

Partitioning is an optimization technique that enables spark to maximize performance across the worker nodes. More performance gains can be achieved when filtering data in queries by eliminating unnecessary disk IO.

Partition the output file

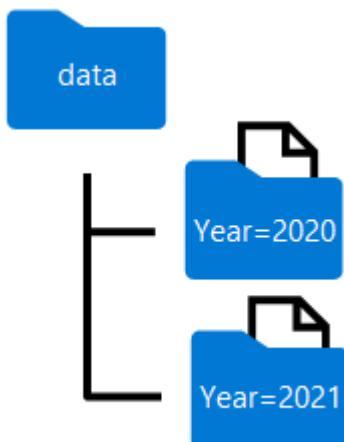
To save a dataframe as a partitioned set of files, use the **partitionBy** method when writing the data.

The following example creates a derived **Year** field. Then uses it to partition the data.

Python

```
from pyspark.sql.functions import year, col  
  
# Load source data  
df = spark.read.csv('/orders/*.csv', header=True, inferSchema=True)  
  
# Add Year column  
dated_df = df.withColumn("Year", year(col("OrderDate")))  
  
# Partition by year  
dated_df.write.partitionBy("Year").mode("overwrite").parquet("/data")
```

The folder names generated when partitioning a dataframe include the partitioning column name and value in a **column=value** format, as shown here:



ⓘ Note

You can partition the data by multiple columns, which results in a hierarchy of folders for each partitioning key. For example, you could partition the order in the example by year and month, so that the folder hierarchy includes a folder for each year value, which in turn contains a subfolder for each month value.

Filter parquet files in a query

When reading data from parquet files into a dataframe, you have the ability to pull data from any folder within the hierarchical folders. This filtering process is done with the use of explicit values and wildcards against the partitioned fields.

In the following example, the following code will pull the sales orders, which were placed in 2020.

Python

```
orders_2020 = spark.read.parquet('/partitioned_data/Year=2020')
display(orders_2020.limit(5))
```

ⓘ Note

The partitioning columns specified in the file path are omitted in the resulting dataframe. The results produced by the example query would not include a **Year** column - all rows would be from 2020.

Next unit: Transform data with SQL

[Continue >](#)

How are we doing? ⭐ ⭐ ⭐ ⭐ ⭐

✓ 100 XP



Transform data with SQL

5 minutes

The SparkSQL library, which provides the dataframe structure also enables you to use SQL as a way of working with data. With this approach, You can query and transform data in dataframes by using SQL queries, and persist the results as tables.

⚠ Note

Tables are metadata abstractions over files. The data is not stored in a relational table, but the table provides a relational layer over files in the data lake.

Define tables and views

Table definitions in Spark are stored in the *metastore*, a metadata layer that encapsulates relational abstractions over files. *External* tables are relational tables in the metastore that reference files in a data lake location that you specify. You can access this data by querying the table or by reading the files directly from the data lake.

⚠ Note

External tables are "loosely bound" to the underlying files and deleting the table *does not* delete the files. This allows you to use Spark to do the heavy lifting of transformation then persist the data in the lake. After this is done you can drop the table and downstream processes can access these optimized structures. You can also define *managed* tables, for which the underlying data files are stored in an internally managed storage location associated with the metastore. Managed tables are "tightly-bound" to the files, and dropping a managed table deletes the associated files.

The following code example saves a dataframe (loaded from CSV files) as an external table name `sales_orders`. The files are stored in the `/sales_orders_table` folder in the data lake.

Python

```
order_details.write.saveAsTable('sales_orders', format='parquet',
mode='overwrite', path='/sales_orders_table')
```

Use SQL to query and transform the data

After defining a table, you can use of SQL to query and transform its data. The following code creates two new derived columns named **Year** and **Month** and then creates a new table *transformed_orders* with the new derived columns added.

Python

```
# Create derived columns
sql_transform = spark.sql("SELECT *, YEAR(OrderDate) AS Year,
MONTH(OrderDate) AS Month FROM sales_orders")

# Save the results
sql_transform.write.partitionBy("Year","Month").saveAsTable('trans-
formed_orders', format='parquet', mode='overwrite', path='/trans-
formed_orders_table')
```

The data files for the new table are stored in a hierarchy of folders with the format of **Year=*NNNN* / Month=*N***, with each folder containing a parquet file for the corresponding orders by year and month.

Query the metastore

Because this new table was created in the metastore, you can use SQL to query it directly with the `%%sql` magic key in the first line to indicate that the SQL syntax will be used as shown in the following script:

SQL

```
%%sql

SELECT * FROM transformed_orders
WHERE Year = 2021
    AND Month = 1
```

Drop tables

When working with external tables, you can use the `DROP` command to delete the table definitions from the metastore without affecting the files in the data lake. This approach enables you to clean up the metastore after using SQL to transform the data, while making the transformed data files available to downstream data analysis and ingestion processes.

SQL

```
%%sql
```

```
DROP TABLE transformed_orders;  
DROP TABLE sales_orders;
```

Next unit: Exercise: Transform data with Spark in Azure Synapse Analytics

[Continue >](#)

How are we doing?

✓ 100 XP



Exercise: Transform data with Spark in Azure Synapse Analytics

30 minutes

Now it's your chance to use Spark to transform data for yourself. In this exercise, you'll use a Spark notebook in Azure Synapse Analytics to transform data in files.

! Note

To complete this lab, you will need an [Azure subscription](#) in which you have administrative access.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

Next unit: Knowledge check

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

1 minute

Linux foundation *Delta Lake* is an open-source storage layer for Spark that enables relational database capabilities for batch and streaming data. By using Delta Lake, you can implement a *data lakehouse* architecture in Spark to support SQL-based data manipulation semantics with support for transactions and schema enforcement. The result is an analytical data store that offers many of the advantages of a relational database system with the flexibility of data file storage in a data lake.

In this module, you'll learn how to:

- Describe core features and capabilities of Delta Lake.
- Create and use Delta Lake tables in a Synapse Analytics Spark pool.
- Create Spark catalog tables for Delta Lake data.
- Use Delta Lake tables for streaming data.
- Query Delta Lake tables from a Synapse Analytics SQL pool.

Note

The version of Delta Lake available in an Azure Synapse Analytics pool depends on the version of Spark specified in the pool configuration. The information in this module reflects Delta Lake version 1.0, which is installed with Spark 3.1.

Next unit: Understand Delta Lake

[Continue >](#)

How are we doing?

✓ 100 XP



Understand Delta Lake

5 minutes

Delta Lake is an open-source storage layer that adds relational database semantics to Spark-based data lake processing. Delta Lake is supported in Azure Synapse Analytics Spark pools for PySpark, Scala, and .NET code.

The benefits of using Delta Lake in a Synapse Analytics Spark pool include:

- **Relational tables that support querying and data modification.** With Delta Lake, you can store data in tables that support *CRUD* (create, read, update, and delete) operations. In other words, you can *select*, *insert*, *update*, and *delete* rows of data in the same way you would in a relational database system.
- **Support for *ACID* transactions.** Relational databases are designed to support transactional data modifications that provide *atomicity* (transactions complete as a single unit of work), *consistency* (transactions leave the database in a consistent state), *isolation* (in-process transactions can't interfere with one another), and *durability* (when a transaction completes, the changes it made are persisted). Delta Lake brings this same transactional support to Spark by implementing a transaction log and enforcing serializable isolation for concurrent operations.
- **Data versioning and *time travel*.** Because all transactions are logged in the transaction log, you can track multiple versions of each table row and even use the *time travel* feature to retrieve a previous version of a row in a query.
- **Support for batch and streaming data.** While most relational databases include tables that store static data, Spark includes native support for streaming data through the Spark Structured Streaming API. Delta Lake tables can be used as both *sinks* (destinations) and *sources* for streaming data.
- **Standard formats and interoperability.** The underlying data for Delta Lake tables is stored in Parquet format, which is commonly used in data lake ingestion pipelines. Additionally, you can use the serverless SQL pool in Azure Synapse Analytics to query Delta Lake tables in SQL.

💡 Tip

For more information about Delta Lake in Azure Synapse Analytics, see [What is Delta Lake](#) in the Azure Synapse Analytics documentation.

✓ 100 XP

Create Delta Lake tables

5 minutes

Delta lake is built on tables, which provide a relational storage abstraction over files in a data lake.

Creating a Delta Lake table from a dataframe

One of the easiest ways to create a Delta Lake table is to save a dataframe in the *delta* format, specifying a path where the data files and related metadata information for the table should be stored.

For example, the following PySpark code loads a dataframe with data from an existing file, and then saves that dataframe to a new folder location in *delta* format:

Python

```
# Load a file into a dataframe
df = spark.read.load('/data/mydata.csv', format='csv', header=True)

# Save the dataframe as a delta table
delta_table_path = "/delta/mydata"
df.write.format("delta").save(delta_table_path)
```

After saving the delta table, the path location you specified includes parquet files for the data (regardless of the format of the source file you loaded into the dataframe) and a `_delta_log` folder containing the transaction log for the table.

⚠ Note

The transaction log records all data modifications to the table. By logging each modification, transactional consistency can be enforced and versioning information for the table can be retained.

You can replace an existing Delta Lake table with the contents of a dataframe by using the `overwrite` mode, as shown here:

Python

```
new_df.write.format("delta").mode("overwrite").save(delta_table_path)
```

You can also add rows from a dataframe to an existing table by using the `append` mode:

Python

```
new_rows_df.write.format("delta").mode("append").save(delta_table_path)
```

Making conditional updates

While you can make data modifications in a dataframe and then replace a Delta Lake table by overwriting it, a more common pattern in a database is to insert, update or delete rows in an existing table as discrete transactional operations. To make such modifications to a Delta Lake table, you can use the `DeltaTable` object in the Delta Lake API, which supports `update`, `delete`, and `merge` operations. For example, you could use the following code to update the `price` column for all rows with a `category` column value of "Accessories":

Python

```
from delta.tables import *
from pyspark.sql.functions import *

# Create a deltaTable object
deltaTable = DeltaTable.forPath(spark, delta_table_path)

# Update the table (reduce price of accessories by 10%)
deltaTable.update(
    condition = "Category == 'Accessories'",
    set = { "Price": "Price * 0.9" })
```

The data modifications are recorded in the transaction log, and new parquet files are created in the table folder as required.

💡 Tip

For more information about using the Data Lake API, see [the Delta Lake API documentation](#).

Querying a previous version of a table

Delta Lake tables support versioning through the transaction log. The transaction log records modifications made to the table, noting the timestamp and version number for each transaction. You can use this logged version data to view previous versions of the table - a feature known as *time travel*.

You can retrieve data from a specific version of a Delta Lake table by reading the data from the delta table location into a dataframe, specifying the version required as a `versionAsOf` option:

Python

```
df = spark.read.format("delta").option("versionAsOf", 0).load(delta_table_path)
```

Alternatively, you can specify a timestamp by using the `timestampAsOf` option:

Python

```
df = spark.read.format("delta").option("timestampAsOf", '2022-01-01').load(delta_table_path)
```

Next unit: Create catalog tables

[Continue >](#)

How are we doing? ★ ★ ★ ★ ★

✓ 100 XP



Create catalog tables

6 minutes

So far we've considered Delta Lake table instances created from dataframes and modified through the Delta Lake API. You can also define Delta Lake tables as catalog tables in the Hive metastore for your Spark pool, and work with them using SQL.

External vs managed tables

Tables in a Spark catalog, including Delta Lake tables, can be *managed* or *external*; and it's important to understand the distinction between these kinds of table.

- A *managed* table is defined without a specified location, and the data files are stored within the storage used by the metastore. Dropping the table not only removes its metadata from the catalog, but also deletes the folder in which its data files are stored.
- An *external* table is defined for a custom file location, where the data for the table is stored. The metadata for the table is defined in the Spark catalog. Dropping the table deletes the metadata from the catalog, but doesn't affect the data files.

Creating catalog tables

There are several ways to create catalog tables.

Creating a catalog table from a dataframe

You can create managed tables by writing a dataframe using the `saveAsTable` operation as shown in the following examples:

Python

```
# Save a dataframe as a managed table
df.write.format("delta").saveAsTable("MyManagedTable")

## specify a path option to save as an external table
df.write.format("delta").option("path", "/mydata").saveAsTable("MyExternal-
Table")
```

Creating a catalog table using SQL

You can also create a catalog table by using the `CREATE TABLE` SQL statement with the `USING DELTA` clause, and an optional `LOCATION` parameter for external tables. You can run the statement using the SparkSQL API, like the following example:

Python

```
spark.sql("CREATE TABLE MyExternalTable USING DELTA LOCATION '/mydata'")
```

Alternatively you can use the native SQL support in Spark to run the statement:

SQL

```
%%sql  
  
CREATE TABLE MyExternalTable  
USING DELTA  
LOCATION '/mydata'
```

💡 Tip

The `CREATE TABLE` statement returns an error if a table with the specified name already exists in the catalog. To mitigate this behavior, you can use a `CREATE TABLE IF NOT EXISTS` statement or the `CREATE OR REPLACE TABLE` statement.

Defining the table schema

In all of the examples so far, the table is created without an explicit schema. In the case of tables created by writing a dataframe, the table schema is inherited from the dataframe. When creating an external table, the schema is inherited from any files that are currently stored in the table location. However, when creating a new managed table, or an external table with a currently empty location, you define the table schema by specifying the column names, types, and nullability as part of the `CREATE TABLE` statement; as shown in the following example:

SQL

```
%%sql  
  
CREATE TABLE ManagedSalesOrders  
(  
    Orderid INT NOT NULL,  
    OrderDate TIMESTAMP NOT NULL,
```

```
    CustomerName STRING,  
    SalesTotal FLOAT NOT NULL  
)  
USING DELTA
```

When using Delta Lake, table schemas are enforced - all inserts and updates must comply with the specified column nullability and data types.

Using the DeltaTableBuilder API

You can use the DeltaTableBuilder API (part of the Delta Lake API) to create a catalog table, as shown in the following example:

Python

```
from delta.tables import *  
  
DeltaTable.create(spark) \  
  .tableName("default.ManagedProducts") \  
  .addColumn("Productid", "INT") \  
  .addColumn("ProductName", "STRING") \  
  .addColumn("Category", "STRING") \  
  .addColumn("Price", "FLOAT") \  
  .execute()
```

Similarly to the CREATE TABLE SQL statement, the create method returns an error if a table with the specified name already exists. You can mitigate this behavior by using the `createIfNotExists` or `createOrReplace` method.

Using catalog tables

You can use catalog tables like tables in any SQL-based relational database, querying and manipulating them by using standard SQL statements. For example, the following code example uses a SELECT statement to query the `ManagedSalesOrders` table:

SQL

```
%%sql  
  
SELECT orderid, salestotal  
FROM ManagedSalesOrders
```

 **Tip**

For more information about working with Delta Lake, see [Table batch reads and writes](#) in the Delta Lake documentation.

Next unit: Use Delta Lake with streaming data

[Continue >](#)

How are we doing?

✓ 100 XP



Use Delta Lake with streaming data

6 minutes

All of the data we've explored up to this point has been static data in files. However, many data analytics scenarios involve *streaming* data that must be processed in near real time. For example, you might need to capture readings emitted by internet-of-things (IoT) devices and store them in a table as they occur.

Spark Structured Streaming

A typical stream processing solution involves constantly reading a stream of data from a *source*, optionally processing it to select specific fields, aggregate and group values, or otherwise manipulate the data, and writing the results to a *sink*.

Spark includes native support for streaming data through *Spark Structured Streaming*, an API that is based on a boundless dataframe in which streaming data is captured for processing. A Spark Structured Streaming dataframe can read data from many different kinds of streaming source, including network ports, real time message brokering services such as Azure Event Hubs or Kafka, or file system locations.

💡 Tip

For more information about Spark Structured Streaming, see [Structured Streaming Programming Guide](#) in the Spark documentation.

Streaming with Delta Lake tables

You can use a Delta Lake table as a source or a sink for Spark Structured Streaming. For example, you could capture a stream of real time data from an IoT device and write the stream directly to a Delta Lake table as a sink - enabling you to query the table to see the latest streamed data. Or, you could read a Delta Table as a streaming source, enabling you to constantly report new data as it is added to the table.

Using a Delta Lake table as a streaming source

In the following PySpark example, a Delta Lake table is used to store details of Internet sales orders. A stream is created that reads data from the Delta Lake table folder as new data is appended.

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Load a streaming dataframe from the Delta Table
stream_df = spark.readStream.format("delta") \
    .option("ignoreChanges", "true") \
    .load("/delta/internetorders")

# Now you can process the streaming data in the dataframe
# for example, show it:
stream_df.writeStream \
    .outputMode("append") \
    .format("console") \
    .start()
```

⚠ Note

When using a Delta Lake table as a streaming source, only *append* operations can be included in the stream. Data modifications will cause an error unless you specify the `ignoreChanges` or `ignoreDeletes` option.

After reading the data from the Delta Lake table into a streaming dataframe, you can use the Spark Structured Streaming API to process it. In the example above, the dataframe is simply displayed; but you could use Spark Structured Streaming to aggregate the data over temporal windows (for example to count the number of orders placed every minute) and send the aggregated results to a downstream process for near-real-time visualization.

Using a Delta Lake table as a streaming sink

In the following PySpark example, a stream of data is read from JSON files in a folder. The JSON data in each file contains the status for an IoT device in the format `{"device": "Dev1", "status": "ok"}`. New data is added to the stream whenever a file is added to the folder. The input stream is a boundless dataframe, which is then written in delta format to a folder location for a Delta Lake table.

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Create a stream that reads JSON data from a folder
inputPath = '/streamingdata/'
jsonSchema = StructType([
    StructField("device", StringType(), False),
    StructField("status", StringType(), False)
])
stream_df =
spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger",
1).json(inputPath)

# Write the stream to a delta table
table_path = '/delta/devicetable'
checkpoint_path = '/delta/checkpoint'
delta_stream = stream_df.writeStream.format("delta").option("checkpointLocation",
checkpoint_path).start(table_path)
```

ⓘ Note

The `checkpointLocation` option is used to write a checkpoint file that tracks the state of the stream processing. This file enables you to recover from failure at the point where stream processing left off.

After the streaming process has started, you can query the Delta Lake table to which the streaming output is being written to see the latest data. For example, the following code creates a catalog table for the Delta Lake table folder and queries it:

SQL

```
%%sql

CREATE TABLE DeviceTable
USING DELTA
LOCATION '/delta/devicetable';

SELECT device, status
FROM DeviceTable;
```

To stop the stream of data being written to the Delta Lake table, you can use the `stop` method of the streaming query:

Python

```
delta_stream.stop()
```



Tip

For more information about using Delta Lake tables for streaming data, see [Table streaming reads and writes](#) in the Delta Lake documentation.

Next unit: Use Delta Lake in a SQL pool

[Continue >](#)

How are we doing?

✓ 100 XP



Use Delta Lake in a SQL pool

5 minutes

Delta Lake is designed as a transactional, relational storage layer for Apache Spark; including Spark pools in Azure Synapse Analytics. However, Azure Synapse Analytics also includes a serverless SQL pool runtime that enables data analysts and engineers to run SQL queries against data in a data lake or a relational database.

! Note

You can only *query* data from Delta Lake tables in a serverless SQL pool; you can't *update*, *insert*, or *delete* data.

Querying delta formatted files with OPENROWSET

The serverless SQL pool in Azure Synapse Analytics includes support for reading delta format files; enabling you to use the SQL pool to query Delta Lake tables. This approach can be useful in scenarios where you want to use Spark and Delta tables to process large quantities of data, but use the SQL pool to run queries for reporting and analysis of the processed data.

In the following example, a SQL `SELECT` query reads delta format data using the `OPENROWSET` function.

SQL

```
SELECT *
FROM
    OPENROWSET(
        BULK 'https://mystore.dfs.core.windows.net/files/delta/mytable/',
        FORMAT = 'DELTA'
    ) AS deltadata
```

You could run this query in a serverless SQL pool to retrieve the latest data from the Delta Lake table stored in the specified file location.

You could also create a database and add a data source that encapsulates the location of your Delta Lake data files, as shown in this example:

SQL

```
CREATE DATABASE MyDB
    COLLATE Latin1_General_100_BIN2_UTF8;
GO;

USE MyDB;
GO

CREATE EXTERNAL DATA SOURCE DeltaLakeStore
WITH
(
    LOCATION = 'https://mystore.dfs.core.windows.net/files/delta/'
);
GO

SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'mytable',
    DATA_SOURCE = 'DeltaLakeStore',
    FORMAT = 'DELTA'
) as deltadata;
```

⚠ Note

When working with Delta Lake data, which is stored in Parquet format, it's generally best to create a database with a UTF-8 based collation in order to ensure string compatibility.

Querying catalog tables

The serverless SQL pool in Azure Synapse Analytics has shared access to databases in the Spark metastore, so you can query catalog tables that were created using Spark SQL. In the following example, a SQL query in a serverless SQL pool queries a catalog table that contains Delta Lake data:

SQL

```
-- By default, Spark catalog tables are created in a database named "default"
-- If you created another database using Spark SQL, you can use it here
USE default;

SELECT * FROM MyDeltaTable;
```

💡 Tip

For more information about using Delta Tables from a serverless SQL pool, see [Query Delta Lake files using serverless SQL pool in Azure Synapse Analytics](#) in the Azure Synapse Analytics documentation.

Next unit: Exercise - Use Delta Lake in Azure Synapse Analytics

[Continue >](#)

How are we doing?

✓ 200 XP



Knowledge check

5 minutes

1. Which of the following descriptions best fits Delta Lake? *

- A Spark API for exporting data from a relational database into CSV files.
- A relational storage layer for Spark that supports tables based on Parquet files.

✓ Correct. Delta Lake provides a relational storage layer in which you can create tables based on Parquet files in a data lake.

- A synchronization solution that replicates data between SQL pools and Spark pools.

✗ Incorrect. Delta Lake does not replicate data between SQL pools and Spark pools.

2. You've loaded a Spark dataframe with data, that you now want to use in a Delta Lake table. What format should you use to write the dataframe to storage? *

- CSV
- PARQUET

✗ Incorrect. Although Delta Lake tables are based on Parquet files, the format must also include a transaction log.

- DELTA

✓ Correct. Storing a dataframe in DELTA format creates parquet files for the data and the transaction log metadata necessary for Delta Lake tables.

3. What feature of Delta Lake enables you to retrieve data from previous versions of a table?
*

- Spark Structured Streaming

✗ Incorrect. Delta Lake tables do support Spark Structured Streaming, but that isn't what enables querying of previous versions.

- Time Travel

✓ Correct. The Time Travel feature is based on the transaction log, which enables you to specify a version number or timestamp for the data you want to retrieve.

Catalog Tables

4. You have a managed catalog table that contains Delta Lake data. If you drop the table, what will happen? *

The table metadata and data files will be deleted.

✓ Correct. The life-cycle of the metadata and data for a managed table are the same.

The table metadata will be removed from the catalog, but the data files will remain intact.

The table metadata will remain in the catalog, but the data files will be deleted.

5. When using Spark Structured Streaming, a Delta Lake table can be which of the following? *

Only a source

Only a sink

✗ Incorrect. A Delta Lake table can be a source or a sink.

Either a source or a sink

✓ Correct. A Delta Lake table can be a source or a sink.

Next unit: Summary

[Continue >](#)

How are we doing?     

RELATIONAL DATA WAREHOUSES ARE AT THE CENTER OF MOST ENTERPRISE BUSINESS INTELLIGENCE (BI)

solutions. While the specific details may vary across data warehouse implementations, a common pattern based on a denormalized, multidimensional schema has emerged as the standard design for a relational data warehouse.

Azure Synapse Analytics includes a highly scalable relational database engine that is optimized for data warehousing workloads. By using *dedicated SQL pools* in Azure Synapse Analytics, you can create databases that are capable of hosting and querying huge volumes of data in relational tables.

In this module, you'll learn how to:

- Design a schema for a relational data warehouse.
- Create fact, dimension, and staging tables.
- Use SQL to load data into data warehouse tables.
- Use SQL to query relational data warehouse tables.

Next unit: Design a data warehouse schema

[Continue >](#)

How are we doing?

Design a data warehouse schema

7 minutes

Like all relational databases, a data warehouse contains tables in which the data you want to analyze is stored. Most commonly, these tables are organized in a schema that is optimized for multidimensional modeling, in which numerical measures associated with events known as *facts* can be aggregated by the attributes of associated entities across multiple *dimensions*. For example, measures associated with a sales order (such as the amount paid or the quantity of items ordered) can be aggregated by attributes of the date on which the sale occurred, the customer, the store, and so on.

Tables in a data warehouse

A common pattern for relational data warehouses is to define a schema that includes two kinds of table: *dimension* tables and *fact* tables.

Dimension tables

Dimension tables describe business entities, such as products, people, places, and dates. Dimension tables contain columns for attributes of an entity. For example, a customer entity might have a first name, a last name, an email address, and a postal address (which might consist of a street address, a city, a postal code, and a country or region). In addition to attribute columns, a dimension table contains a unique key column that uniquely identifies each row in the table. In fact, it's common for a dimension table to include **two** key columns:

- a *surrogate* key that is specific to the data warehouse and uniquely identifies each row in the dimension table in the data warehouse - usually an incrementing integer number.
- An *alternate* key, often a *natural* or *business* key that is used to identify a specific instance of an entity in the transactional source system from which the entity record originated - such as a product code or a customer ID.

① Note

Why have two keys? There are a few good reasons:

- The data warehouse may be populated with data from multiple source systems, which can lead to the risk of duplicate or incompatible business keys.
- Simple numeric keys generally perform better in queries that join lots of tables - a common pattern in data warehouses.
- Attributes of entities may change over time - for example, a customer might change their address. Since the data warehouse is used to support historic reporting, you may want to retain a record for each instance of an entity at multiple points in time; so that, for example, sales orders for a specific customer are counted for the city where they lived at the time the order was placed. In this case, multiple customer records would have the same business key associated with the customer, but different surrogate keys for each discrete address where the customer lived at various times.

An example of a dimension table for customer might contain the following data:

CustomerKey	CustomerAltKey	Name	Email	Street	City	PostalCode	CountryRegion
123	I-543	Navin Jones	navin1@contoso.com	1 Main St.	Seattle	90000	United States
124	R-589	Mary Smith	mary2@contoso.com	234 190th Ave	Buffalo	50001	United States
125	I-321	Antoine Dubois	antoine1@contoso.com	2 Rue Jolie	Paris	20098	France
126	I-543	Navin Jones	navin1@contoso.com	24 125th Ave.	New York	50000	United States
...

① Note

Observe that the table contains two records for *Navin Jones*. Both records use the same alternate key to identify this person (*I-543*), but each record has a different surrogate key. From this, you can surmise that the customer moved from Seattle to New York. Sales made to the customer while living in Seattle are associated with the key *123*, while purchases made after moving to New York are recorded against record *126*.

In addition to dimension tables that represent business entities, it's common for a data warehouse to include a dimension table that represents *time*. This table enables data analysts to aggregate data over temporal intervals. Depending on the type of data you need to analyze, the lowest granularity (referred to as the *grain*) of a time dimension could represent times (to the hour, second, millisecond, nanosecond, or even lower), or dates.

An example of a time dimension table with a grain at the date level might contain the following data:

DateKey	DateAltKey	DayOfWeek	DayOfMonth	Weekday	Month	MonthName	Quarter	Year
19990101	01-01-1999	6	1	Friday	1	January	1	1999
...
20220101	01-01-2022	7	1	Saturday	1	January	1	2022
20220102	02-01-2022	1	2	Sunday	1	January	1	2022
...
20301231	31-12-2030	3	31	Tuesday	12	December	4	2030

The timespan covered by the records in the table must include the earliest and latest points in time for any associated events recorded in a related fact table. Usually there's a record for every interval at the appropriate grain in between.

Fact tables

Fact tables store details of observations or events; for example, sales orders, stock balances, exchange rates, or recorded temperatures. A fact table contains columns for numeric values that can be aggregated by dimensions. In addition to the numeric columns, a fact table contains key columns that reference unique keys in related dimension tables.

For example, a fact table containing details of sales orders might contain the following data:

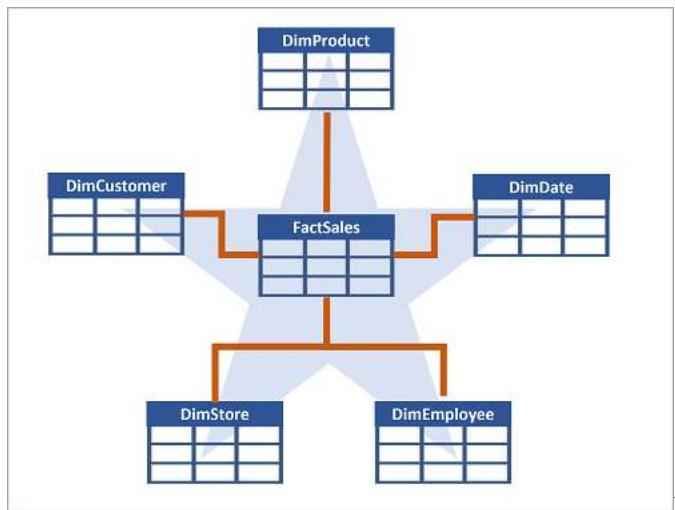
OrderDateKey	CustomerKey	StoreKey	ProductKey	OrderNo	LineItemNo	Quantity	UnitPrice	Tax	ItemTotal
20220101	123	5	701	1001	1	2	2.50	0.50	5.50
20220101	123	5	765	1001	2	1	2.00	0.20	2.20
20220102	125	2	723	1002	1	1	4.99	0.49	5.48
20220103	126	1	823	1003	1	1	7.99	0.80	8.79
...

A fact table's dimension key columns determine its grain. For example, the sales orders fact table includes keys for dates, customers, stores, and products. An order might include multiple products, so the grain represents line items for individual products sold in stores to customers on specific days.

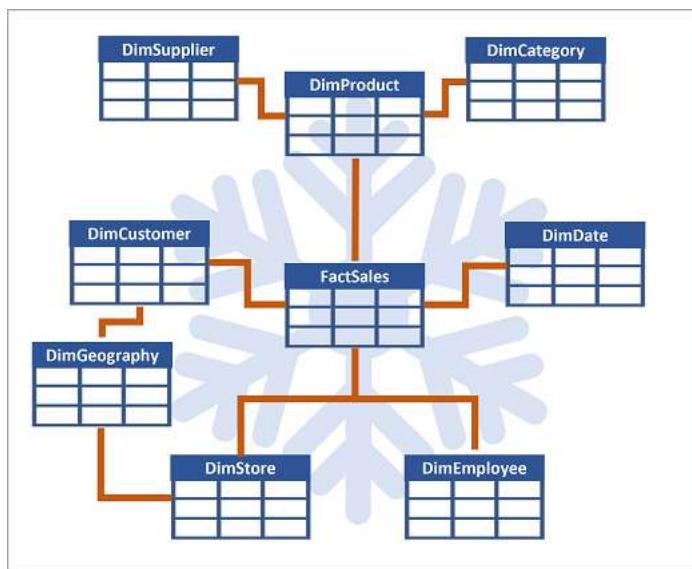
Data warehouse schema designs

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

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



The attributes of an entity can be used to aggregate measures in fact tables over multiple hierarchical levels - for example, to find total sales revenue by country or region, city, postal code, or individual customer. The attributes for each level can be stored in the same dimension table. However, when an entity has a large number of hierarchical attribute levels, or when some attributes can be shared by multiple dimensions (for example, both customers and stores have a geographical address), it can make sense to apply some normalization to the dimension tables and create a *snowflake* schema, as shown in the following example:



In this case, the **DimProduct** table has been normalized to create separate dimension tables for product categories and suppliers, and a **DimGeography** table has been added to represent geographical attributes for both customers and stores. Each row in the **DimProduct** table contains key values for the corresponding rows in the **DimCategory** and **DimSupplier** tables; and each row in the **DimCustomer** and **DimStore** tables contains a key value for the corresponding row in the **DimGeography** table.

Next unit: Create data warehouse tables

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP ➔

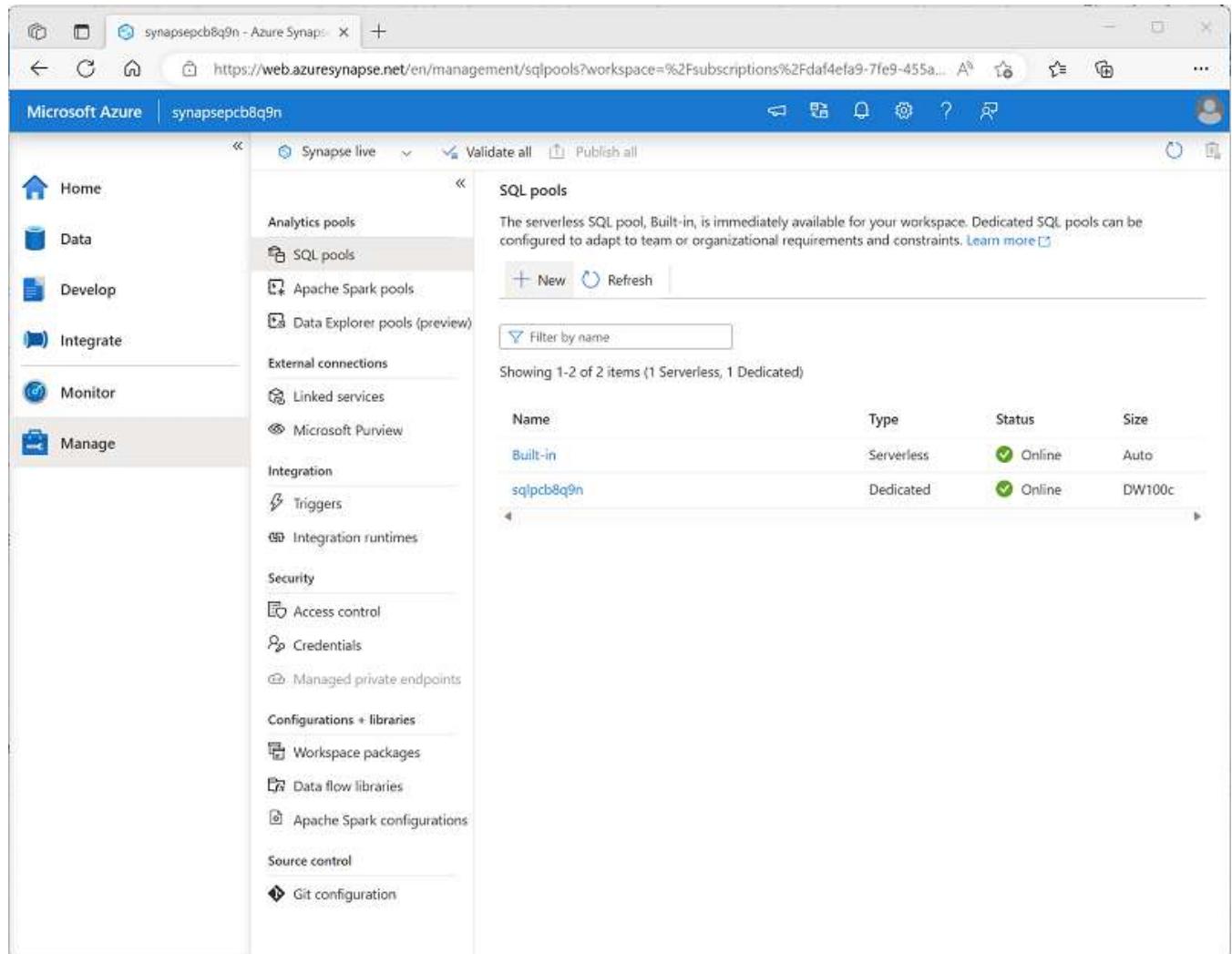
Create data warehouse tables

10 minutes

Now that you understand the basic architectural principles for a relational data warehouse schema, let's explore how to create a data warehouse.

Creating a dedicated SQL pool

To create a relational data warehouse in Azure Synapse Analytics, you must create a dedicated SQL Pool. The simplest way to do this in an existing Azure Synapse Analytics workspace is to use the **Manage** page in Azure Synapse Studio, as shown here:



The screenshot shows the Microsoft Azure Synapse Studio interface. The left sidebar has a 'Manage' tab selected. The main area is titled 'SQL pools'. It displays two items: 'Built-in' (Serverless, Online, Auto) and 'sqlpcb8q9n' (Dedicated, Online, DW100c). A 'New' button is available to create a new pool.

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
sqlpcb8q9n	Dedicated	Online	DW100c

When provisioning a dedicated SQL pool, you can specify the following configuration settings:

- A unique name for the dedicated SQL pool.

- A performance level for the SQL pool, which can range from *DW100c* to *DW30000c* and which determines the cost per hour for the pool when it's running.
- Whether to start with an empty pool or restore an existing database from a backup.
- The *collation* of the SQL pool, which determines sort order and string comparison rules for the database. (*You can't change the collation after creation*).

After creating a dedicated SQL pool, you can control its running state in the **Manage** page of Synapse Studio; pausing it when not required to prevent unnecessary costs.

When the pool is running, you can explore it on the **Data** page, and create SQL scripts to run in it.

Considerations for creating tables

To create tables in the dedicated SQL pool, you use the `CREATE TABLE` (or sometimes the `CREATE EXTERNAL TABLE`) Transact-SQL statement. The specific options used in the statement depend on the type of table you're creating, which can include:

- Fact tables
- Dimension tables
- Staging tables

! Note

The data warehouse is composed of *fact* and *dimension* tables as discussed previously. *Staging* tables are often used as part of the data warehousing loading process to ingest data from source systems.

When designing a star schema model for small or medium sized datasets you can use your preferred database, such as Azure SQL. For larger data sets you may benefit from implementing your data warehouse in Azure Synapse Analytics instead of SQL Server. It's important to understand some key differences when creating tables in Synapse Analytics.

Data integrity constraints

Dedicated SQL pools in Synapse Analytics don't support *foreign key* and *unique* constraints as found in other relational database systems like SQL Server. This means that jobs used to load data must maintain uniqueness and referential integrity for keys, without relying on the table definitions in the database to do so.

💡 Tip

For more information about constraints in Azure Synapse Analytics dedicated SQL pools, see [Primary key, foreign key, and unique key using dedicated SQL pool in Azure Synapse Analytics](#).

Indexes

While Synapse Analytics dedicated SQL pools support *clustered* indexes as found in SQL Server, the default index type is *clustered columnstore*. This index type offers a significant performance advantage when querying large quantities of data in a typical data warehouse schema and should be used where possible. However, some tables may include data types that can't be included in a clustered columnstore index (for example, VARBINARY(MAX)), in which case a clustered index can be used instead.

Tip

For more information about indexing in Azure Synapse Analytics dedicated SQL pools, see [Indexes on dedicated SQL pool tables in Azure Synapse Analytics](#).

Distribution

Azure Synapse Analytics dedicated SQL pools use a [massively parallel processing \(MPP\) architecture](#), as opposed to the symmetric multiprocessing (SMP) architecture used in most OLTP database systems. In an MPP system, the data in a table is distributed for processing across a pool of nodes. Synapse Analytics supports the following kinds of distribution:

- **Hash:** A deterministic hash value is calculated for the specified column and used to assign the row to a compute node.
- **Round-robin:** Rows are distributed evenly across all compute nodes.
- **Replicated:** A copy of the table is stored on each compute node.

The table type often determines which option to choose for distributing the table.

Table type	Recommended distribution option
Dimension	Use replicated distribution for smaller tables to avoid data shuffling when joining to distributed fact tables. If tables are too large to store on each compute node, use hash distribution.

Table type	Recommended distribution option
Fact	Use hash distribution with clustered columnstore index to distribute fact tables across compute nodes.
Staging	Use round-robin distribution for staging tables to evenly distribute data across compute nodes.

Tip

For more information about distribution strategies for tables in Azure Synapse Analytics, see [Guidance for designing distributed tables using dedicated SQL pool in Azure Synapse Analytics](#).

Creating dimension tables

When you create a dimension table, ensure that the table definition includes surrogate and alternate keys as well as columns for the attributes of the dimension that you want to use to group aggregations. It's often easiest to use an `IDENTITY` column to auto-generate an incrementing surrogate key (otherwise you need to generate unique keys every time you load data). The following example shows a `CREATE TABLE` statement for a hypothetical `DimCustomer` dimension table.

SQL

```
CREATE TABLE dbo.DimCustomer
(
    CustomerKey INT IDENTITY NOT NULL,
    CustomerAlternateKey NVARCHAR(15) NULL,
    CustomerName NVARCHAR(80) NOT NULL,
    EmailAddress NVARCHAR(50) NULL,
    Phone NVARCHAR(25) NULL,
    StreetAddress NVARCHAR(100),
    City NVARCHAR(20),
    PostalCode NVARCHAR(10),
    CountryRegion NVARCHAR(20)
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
```

(!) Note

If desired, you can create a specific *schema* as a namespace for your tables. In this example, the default **dbo** schema is used.

If you intend to use a *snowflake* schema in which dimension tables are related to one another, you should include the key for the *parent* dimension in the definition of the *child* dimension table. For example, the following SQL code could be used to move the geographical address details from the **DimCustomer** table to a separate **DimGeography** dimension table:

SQL

```
CREATE TABLE dbo.DimGeography
(
    GeographyKey INT IDENTITY NOT NULL,
    GeographyAlternateKey NVARCHAR(10) NULL,
    StreetAddress NVARCHAR(100),
    City NVARCHAR(20),
    PostalCode NVARCHAR(10),
    CountryRegion NVARCHAR(20)
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);

CREATE TABLE dbo.DimCustomer
(
    CustomerKey INT IDENTITY NOT NULL,
    CustomerAlternateKey NVARCHAR(15) NULL,
    GeographyKey INT NULL,
    CustomerName NVARCHAR(80) NOT NULL,
    EmailAddress NVARCHAR(50) NULL,
    Phone NVARCHAR(25) NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
```

Time dimension tables

Most data warehouses include a *time* dimension table that enables you to aggregate data by multiple hierarchical levels of time interval. For example, the following example creates a **DimDate** table with attributes that relate to specific dates.

SQL

```
CREATE TABLE dbo.DimDate
(
    DateKey INT NOT NULL,
    DateAltKey DATETIME NOT NULL,
    DayOfMonth INT NOT NULL,
    DayOfWeek INT NOT NULL,
    DayName NVARCHAR(15) NOT NULL,
    MonthOfYear INT NOT NULL,
    MonthName NVARCHAR(15) NOT NULL,
    CalendarQuarter INT NOT NULL,
    CalendarYear INT NOT NULL,
    FiscalQuarter INT NOT NULL,
    FiscalYear INT NOT NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
```

💡 Tip

A common pattern when creating a dimension table for dates is to use the numeric date in *DDMMYYYY* or *YYYYMMDD* format as an integer surrogate key, and the date as a DATE or DATETIME datatype as the alternate key.

Creating fact tables

Fact tables include the keys for each dimension to which they're related, and the attributes and numeric measures for specific events or observations that you want to analyze.

The following code example creates a hypothetical fact table named **FactSales** that is related to multiple dimensions through key columns (date, customer, product, and store)

SQL

```
CREATE TABLE dbo.FactSales
(
    OrderDateKey INT NOT NULL,
    CustomerKey INT NOT NULL,
    ProductKey INT NOT NULL,
    StoreKey INT NOT NULL,
    OrderNumber NVARCHAR(10) NOT NULL,
    OrderLineItem INT NOT NULL,
    OrderQuantity SMALLINT NOT NULL,
```

```
    UnitPrice DECIMAL NOT NULL,
    Discount DECIMAL NOT NULL,
    Tax DECIMAL NOT NULL,
    SalesAmount DECIMAL NOT NULL
)
WITH
(
    DISTRIBUTION = HASH(OrderNumber),
    CLUSTERED COLUMNSTORE INDEX
);
```

Creating staging tables

Staging tables are used as temporary storage for data as it's being loaded into the data warehouse. A typical pattern is to structure the table to make it as efficient as possible to ingest the data from its external source (often files in a data lake) into the relational database, and then use SQL statements to load the data from the staging tables into the dimension and fact tables.

The following code example creates a staging table for product data that will ultimately be loaded into a dimension table:

SQL

```
CREATE TABLE dbo.StageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ProductCategory NVARCHAR(200) NOT NULL,
    Color NVARCHAR(10),
    Size NVARCHAR(10),
    ListPrice DECIMAL NOT NULL,
    Discontinued BIT NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
);
```

Using external tables

In some cases, if the data to be loaded is in files with an appropriate structure, it can be more effective to create external tables that reference the file location. This way, the data can be read directly from the source files instead of being loaded into the relational store. The

following example, shows how to create an external table that references files in the data lake associated with the Synapse workspace:

```
SQL

-- External data source links to data lake location
CREATE EXTERNAL DATA SOURCE StagedFiles
WITH (
    LOCATION = 'https://mydatalake.blob.core.windows.net/data/stagedfiles/'
);
GO

-- External format specifies file format
CREATE EXTERNAL FILE FORMAT ParquetFormat
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
);
GO

-- External table references files in external data source
CREATE EXTERNAL TABLE dbo.ExternalStageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ProductCategory NVARCHAR(200) NOT NULL,
    Color NVARCHAR(10),
    Size NVARCHAR(10),
    ListPrice DECIMAL NOT NULL,
    Discontinued BIT NOT NULL
)
WITH
(
    DATA_SOURCE = StagedFiles,
    LOCATION = 'products/*.parquet',
    FILE_FORMAT = ParquetFormat
);
GO
```

 **Note**

For more information about using external tables, see [Use external tables with Synapse SQL](#) in the Azure Synapse Analytics documentation.

Next unit: Load data warehouse tables

Load data warehouse tables

10 minutes

At a basic level, loading a data warehouse is typically achieved by adding new data from files in a data lake into tables in the data warehouse. The `COPY` statement is an effective way to accomplish this task, as shown in the following example:

SQL

```
COPY INTO dbo.StageProducts
    (ProductID, ProductName, ProductCategory, Color, Size, ListPrice,
Discontinued)
FROM 'https://mydatalake.blob.core.windows.net/data/stagedfiles/prod-
ucts/*.parquet'
WITH
(
    FILE_TYPE = 'PARQUET',
    MAXERRORS = 0,
    IDENTITY_INSERT = 'OFF'
);
```

Considerations for designing a data warehouse load process

One of the most common patterns for loading a data warehouse is to transfer data from source systems to files in a data lake, ingest the file data into staging tables, and then use SQL statements to load the data from the staging tables into the dimension and fact tables. Usually data loading is performed as a periodic batch process in which inserts and updates to the data warehouse are coordinated to occur at a regular interval (for example, daily, weekly, or monthly).

In most cases, you should implement a data warehouse load process that performs tasks in the following order:

1. Ingest the new data to be loaded into a data lake, applying pre-load cleansing or transformations as required.
2. Load the data from files into staging tables in the relational data warehouse.
3. Load the dimension tables from the dimension data in the staging tables, updating existing rows or inserting new rows and generating surrogate key values as necessary.

4. Load the fact tables from the fact data in the staging tables, looking up the appropriate surrogate keys for related dimensions.
5. Perform post-load optimization by updating indexes and table distribution statistics.

After using the `COPY` statement to load data into staging tables, you can use a combination of `INSERT`, `UPDATE`, `MERGE`, and `CREATE TABLE AS SELECT (CTAS)` statements to load the staged data into dimension and fact tables.

 **Note**

Implementing an effective data warehouse loading solution requires careful consideration of how to manage surrogate keys, slowly changing dimensions, and other complexities inherent in a relational data warehouse schema. To learn more about techniques for loading a data warehouse, consider completing the [Load data into a relational data warehouse](#) module.

Next unit: Query a data warehouse

[Continue >](#)

How are we doing?     

Query a data warehouse

10 minutes

When the dimension and fact tables in a data warehouse have been loaded with data, you can use SQL to query the tables and analyze the data they contain. The Transact-SQL syntax used to query tables in a Synapse dedicated SQL pool is similar to SQL used in SQL Server or Azure SQL Database.

Aggregating measures by dimension attributes

Most data analytics with a data warehouse involves aggregating numeric measures in fact tables by attributes in dimension tables. Because of the way a star or snowflake schema is implemented, queries to perform this kind of aggregation rely on `JOIN` clauses to connect fact tables to dimension tables, and a combination of aggregate functions and `GROUP BY` clauses to define the aggregation hierarchies.

For example, the following SQL queries the `FactSales` and `DimDate` tables in a hypothetical data warehouse to aggregate sales amounts by year and quarter:

SQL

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

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

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

CalendarYear	CalendarQuarter	TotalSales
...

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

SQL

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

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

CalendarYear	CalendarQuarter	City	TotalSales
2020	1	Amsterdam	5982.53
2020	1	Berlin	2826.98
2020	1	Chicago	5372.72
...
2020	2	Amsterdam	7163.93
2020	2	Berlin	8191.12
2020	2	Chicago	2428.72
...
2020	3	Amsterdam	7261.92
2020	3	Berlin	4202.65
2020	3	Chicago	2287.87

CalendarYear	CalendarQuarter	City	TotalSales
...
2020	4	Amsterdam	8262.73
2020	4	Berlin	5373.61
2020	4	Chicago	7726.23
...
2021	1	Amsterdam	7261.28
2021	1	Berlin	3648.28
2021	1	Chicago	1027.27
...

Joins in a snowflake schema

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

SQL

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

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

ProductCategory	ItemsSold
Accessories	28271
Bits and pieces	5368

ProductCategory	ItemsSold
...	...

① Note

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

Using ranking functions

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

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

For example, consider the following query:

SQL

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

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

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

ⓘ Note

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

To learn more about ranking functions, see [Ranking Functions \(Transact-SQL\)](#) in the Azure Synapse Analytics documentation.

Retrieving an approximate count

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

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

SQL

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

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

CalendarYear	Orders
2019	239870
2020	284741
2021	309272
...	...

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

SQL

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

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

CalendarYear	ApproxOrders
2019	235552
2020	290436

CalendarYear	ApproxOrders
2021	304633
...	...

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

ⓘ Note

See the `APPROX_COUNT_DISTINCT` function documentation for more details.

Next unit: Exercise - Explore a data warehouse

[Continue >](#)

How are we doing?

✓ 200 XP



Knowledge check

5 minutes

1. In which of the following table types should an insurance company store details of customer attributes by which claims will be aggregated? *

 Staging table Dimension table

✓ Correct. Attributes of an entity by which numeric measures will be aggregated are stored in a dimension table.

 Fact table

2. You create a dimension table for product data, assigning a unique numeric key for each row in a column named `ProductKey`. The `ProductKey` is only defined in the data warehouse. What kind of key is `ProductKey`? *

 A surrogate key

✓ Correct. A surrogate key uniquely identifies each row in a dimension table, irrespective of keys used in source systems.

 An alternate key A business key

3. What distribution option would be best for a sales fact table that will contain billions of records? *

 HASH

✓ Correct. Hash distribution provides good read performance for a large table by distributing records across compute nodes based on the hash key.

 ROUND_ROBIN

X Incorrect. Round robin distribution distributes the data evenly but does not optimize queries on commonly used distribution key fields.

REPLICATE

4. You need to write a query to return the total of the **UnitsProduced** numeric measure in the **FactProduction** table aggregated by the **ProductName** attribute in the **FactProduct** table. Both tables include a **ProductKey** surrogate key field. What should you do? *

- Use two SELECT queries with a UNION ALL clause to combine the rows in the FactProduction table with those in the FactProduct table.
- Use a SELECT query against the FactProduction table with a WHERE clause to filter out rows with a ProductKey that doesn't exist in the FactProduct table.
- Use a SELECT query with a SUM function to total the UnitsProduced metric, using a JOIN on the ProductKey surrogate key to match the FactProduction records to the FactProduct records and a GROUP BY clause to aggregate by ProductName .

✓ Correct. To aggregate measures in a fact table by attributes in a dimension table, include an aggregate function for the measure, join the tables on the surrogate key, and group the results by the appropriate attributes.

5. You use the **RANK** function in a query to rank customers in order of the number of purchases they have made. Five customers have made the same number of purchases and are all ranked equally as 1. What rank will the customer with the next highest number of purchases be assigned? *

two

six

✓ Correct. There are five customers with a higher number of purchases, and RANK takes these into account.

one

6. You need to compare approximate production volumes by product while optimizing query response time. Which function should you use? *

COUNT

✗ Incorrect. The `COUNT` function will return accurate counts, but may take longer than other options.

NTILE

APPROX_COUNT_DISTINCT

✓ Correct. `APPROX_COUNT_DISTINCT` returns an approximate count within 2% of the actual count while optimizing for minimal response time.

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

1 minute

Many enterprise analytical solutions include a relational data warehouse. Data engineers are responsible for implementing ingestion solutions that load data into the data warehouse tables, usually on a regular schedule.

As a data engineer, you need to be familiar with the considerations and techniques that apply to loading a data warehouse. In this module, we'll focus on ways that you can use SQL to load data into tables in a dedicated SQL pool in Azure Synapse Analytics.

Next unit: Load staging tables

[Continue >](#)

How are we doing?

✓ 100 XP



Load staging tables

5 minutes

One of the most common patterns for loading a data warehouse is to transfer data from source systems to files in a data lake, ingest the file data into staging tables, and then use SQL statements to load the data from the staging tables into the dimension and fact tables. Usually data loading is performed as a periodic batch process in which inserts and updates to the data warehouse are coordinated to occur at a regular interval (for example, daily, weekly, or monthly).

Creating staging tables

Many organized warehouses have standard structures for staging the database and may even use a specific schema for staging the data. The following code example creates a staging table for product data that will ultimately be loaded into a dimension table:

ⓘ Note

This example creates a staging table in the default **dbo** schema. You can also create separate schemas for staging tables with a meaningful name, such as **stage** so architects and users understand the purpose of the schema.

SQL

```
CREATE TABLE dbo.StageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ProductCategory NVARCHAR(200) NOT NULL,
    Color NVARCHAR(10),
    Size NVARCHAR(10),
    ListPrice DECIMAL NOT NULL,
    Discontinued BIT NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
);
```

Using the COPY command

You can use the COPY statement to load data from the data lake, as shown in the following example:

! Note

This is generally the recommended approach to load staging tables due to its high performance throughput.

SQL

```
COPY INTO dbo.StageProduct
    (ProductID, ProductName, ...)
FROM 'https://mydatalake.../data/products*.parquet'
WITH
(
    FILE_TYPE = 'PARQUET',
    MAXERRORS = 0,
    IDENTITY_INSERT = 'OFF'
);
```

💡 Tip

To learn more about the COPY statement, see [COPY \(Transact-SQL\)](#) in the Transact-SQL documentation.

Using external tables

In some cases, if the data to be loaded is stored in files with an appropriate structure, it can be more effective to create external tables that reference the file location. This way, the data can be read directly from the source files instead of being loaded into the relational store. The following example, shows how to create an external table that references files in the data lake associated with the Azure Synapse Analytics workspace:

SQL

```
CREATE EXTERNAL TABLE dbo.ExternalStageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(10) NOT NULL,
    ...
)
```

```
WITH
(
    DATE_SOURCE = StagedFiles,
    LOCATION = 'folder_name/*.parquet',
    FILE_FORMAT = ParquetFormat
);
GO
```

 **Tip**

For more information about using external tables, see [Use external tables with Synapse SQL](#) in the Azure Synapse Analytics documentation.

Next unit: Load dimension tables

[Continue >](#)

How are we doing?     

✓ 100 XP



Load dimension tables

5 minutes

After staging dimension data, you can load it into dimension tables using SQL.

Using a CREATE TABLE AS (CTAS) statement

One of the simplest ways to load data into a new dimension table is to use a `CREATE TABLE AS (CTAS)` expression. This statement creates a new table based on the results of a `SELECT` statement.

SQL

```
CREATE TABLE dbo.DimProduct
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT ROW_NUMBER() OVER(ORDER BY ProdID) AS ProdKey,
    ProdID as ProdAltKey,
    ProductName,
    ProductCategory,
    Color,
    Size,
    ListPrice,
    Discontinued
FROM dbo.StageProduct;
```

ⓘ Note

You can't use `IDENTITY` to generate a unique integer value for the surrogate key when using a CTAS statement, so this example uses the `ROW_NUMBER` function to generate an incrementing row number for each row in the results ordered by the `ProductID` business key in the staged data.

You can also load a combination of new and updated data into a dimension table by using a `CREATE TABLE AS (CTAS)` statement to create a new table that `UNIONs` the existing rows from the dimension table with the new and updated records from the staging table. After creating

the new table, you can delete or rename the current dimension table, and rename the new table to replace it.

SQL

```
CREATE TABLE dbo.DimProductUpsert
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
)
AS
-- New or updated rows
SELECT stg.ProductID AS ProductBusinessKey,
       stg.ProductName,
       stg.ProductCategory,
       stg.Color,
       stg.Size,
       stg.ListPrice,
       stg.Discontinued
  FROM dbo.StageProduct AS stg
UNION ALL
-- Existing rows
SELECT dim.ProductBusinessKey,
       dim.ProductName,
       dim.ProductCategory,
       dim.Color,
       dim.Size,
       dim.ListPrice,
       dim.Discontinued
  FROM dbo.DimProduct AS dim
 WHERE NOT EXISTS
(   SELECT *
    FROM dbo.StageProduct AS stg
   WHERE stg.ProductId = dim.ProductBusinessKey
);
RENAME OBJECT dbo.DimProduct TO DimProductArchive;
RENAME OBJECT dbo.DimProductUpsert TO DimProduct;
```

While this technique is effective in merging new and existing dimension data, lack of support for IDENTITY columns means that it's difficult to generate a surrogate key.

💡 Tip

For more information, see [CREATE TABLE AS SELECT \(CTAS\)](#) in the Azure Synapse Analytics documentation.

Using an INSERT statement

When you need to load staged data into an existing dimension table, you can use an `INSERT` statement. This approach works if the staged data contains only records for new dimension entities (not updates to existing entities). This approach is much less complicated than the technique in the last section, which required a `UNION ALL` and then renaming table objects.

SQL

```
INSERT INTO dbo.DimCustomer
SELECT CustomerNo AS CustAltKey,
       CustomerName,
       EmailAddress,
       Phone,
       StreetAddress,
       City,
       PostalCode,
       CountryRegion
  FROM dbo.StageCustomers
```

! Note

Assuming the `DimCustomer` dimension table is defined with an `IDENTITY CustomerKey` column for the surrogate key (as described in the previous unit), the key will be generated automatically and the remaining columns will be populated using the values retrieved from the staging table by the `SELECT` query.

Next unit: Load time dimension tables

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



Load time dimension tables

3 minutes

Time dimension tables store a record for each time interval based on the *grain* by which you want to aggregate data over time. For example, a time dimension table at the *date* grain contains a record for each date between the earliest and latest dates referenced by the data in related fact tables.

The following code example shows how you can generate a sequence of time dimension values based on a *date* grain.

SQL

```
-- Create a temporary table for the dates we need
CREATE TABLE #TmpStageDate (DateVal DATE NOT NULL)

-- Populate the temp table with a range of dates
DECLARE @StartDate DATE
DECLARE @EndDate DATE
SET @StartDate = '2019-01-01'
SET @EndDate = '2023-12-31'
DECLARE @LoopDate = @StartDate
WHILE @LoopDate <= @EndDate
BEGIN
    INSERT INTO #TmpStageDate VALUES
    (
        @LoopDate
    )
    SET @LoopDate = DATEADD(dd, 1, @LoopDate)
END

-- Insert the dates and calculated attributes into the dimension table
INSERT INTO dbo.DimDate
SELECT CAST(CONVERT(VARCHAR(8), DateVal, 112) as INT), -- date key
       DateVal, --date alt key
       Day(DateVal) -- day number of month
       --, other derived temporal fields as required
FROM #TmpStageDate
GO

--Drop temporary table
DROP TABLE #TmpStageDate
```

Tip

Scripting this in SQL may be time-consuming in a dedicated SQL pool – it may be more efficient to prepare the data in Microsoft Excel or an external script and import it using the COPY statement.

As the data warehouse is populated in the future with new fact data, you periodically need to extend the range of dates related time dimension tables.

Next unit: Load slowly changing dimensions

[Continue >](#)

How are we doing?

Load slowly changing dimensions

5 minutes

In most relational data warehouses, you need to handle updates to dimension data and support what are commonly referred to as *slowly changing dimensions* (SCDs).

Types of slowly changing dimension

There are multiple kinds of slowly changing dimension, of which three are commonly implemented:

Type 0

Type 0 dimension data can't be changed. Any attempted changes fail.

DateKey	DateAltKey	Day	Month	Year
20230101	01-01-2023	Sunday	January	2023

Type 1

In *type 1* dimensions, the dimension record is updated in-place. Changes made to an existing dimension row apply to all previously loaded facts related to the dimension.

StoreKey	StoreAltKey	StoreName
123	EH199J	High Street Store Town Central Store

Type 2

In a *type 2* dimension, a change to a dimension results in a new dimension row. Existing rows for previous versions of the dimension are retained for historical fact analysis and the new row is applied to future fact table entries.

CustomerKey	CustomerAltKey	Name	Address	City	DateFrom	DateTo	IsCurrent
1211	jo@contoso.com	Jo Smith	999 Main St	Seattle	20190101	20230105	False
2996	jo@contoso.com	Jo Smith	1234 9th Ave	Boston	20230106		True

⚠ Note

Type 2 dimensions often include columns to track the effective time periods for each version of an entity, and/or a flag to indicate which row represents the current version of the entity. If you're using an incrementing surrogate key and you only need to track the most recently added version of an entity, then you may not need these columns; but before making that decision, consider how you'll look up the appropriate version of an entity when a new fact is entered based on the time at which the event the fact relates to occurred.

Combining INSERT and UPDATE statements

Logic to implement Type 1 and Type 2 updates can be complex, and there are various techniques you can use. For example, you could use a combination of UPDATE and INSERT statements.

SQL

```
-- New Customers
INSERT INTO dbo.DimCustomer
SELECT stg.*
FROM dbo.StageCustomers AS stg
WHERE NOT EXISTS
    (SELECT * FROM dbo.DimCustomer AS dim
     WHERE dim.CustomerAltKey = stg.CustNo)

-- Type 1 updates (name)
UPDATE dbo.DimCustomer
SET CustomerName = stg.CustomerName
FROM dbo.StageCustomers AS stg
WHERE dbo.DimCustomer.CustomerAltKey = stg.CustomerNo;

-- Type 2 updates (StreetAddress)
INSERT INTO dbo.DimCustomer
SELECT stg.*
FROM dbo.StageCustomers AS stg
JOIN dbo.DimCustomer AS dim
ON stg.CustNo = dim.CustomerAltKey
AND stg.StreetAddress <> dim.StreetAddress;
```

In the previous example, it's assumed that an incrementing surrogate key based on an IDENTITY column identifies each row, and that the highest value surrogate key for a given alternate key indicates the most recent or "current" instance of the dimension entity associated with that alternate key. In practice, many data warehouse designers include a Boolean column to indicate the current active instance of a changing dimension or use DateTime fields to indicate the active time periods for each version of the dimension instance. With these approaches, the logic for a type 2 change must include an INSERT of the new dimension row *and* an UPDATE to mark the current row as inactive.

Using a MERGE statement

As an alternative to using multiple INSERT and UPDATE statements, you can use a single MERGE statement to perform an "*upsert*" operation to insert new records and update existing ones.

SQL

```
MERGE dbo.DimProduct AS tgt
    USING (SELECT * FROM dbo.StageProducts) AS src
```

```
ON src.ProductID = tgt.ProductBusinessKey
WHEN MATCHED THEN
    -- Type 1 updates
    UPDATE SET
        tgt.ProductName = src.ProductName,
        tgt.ProductCategory = src.ProductCategory,
        tgt.Color = src.Color,
        tgt.Size = src.Size,
        tgt.ListPrice = src.ListPrice,
        tgt.Discontinued = src.Discontinued
WHEN NOT MATCHED THEN
    -- New products
    INSERT VALUES
        (src.ProductID,
        src.ProductName,
        src.ProductCategory,
        src.Color,
        src.Size,
        src.ListPrice,
        src.Discontinued);
```

ⓘ Note

For more information about the MERGE statement, see the [MERGE documentation for Azure Synapse Analytics](#).

Next unit: Load fact tables

[Continue >](#)

How are we doing? ★ ★ ★ ★ ★

100 XP



Load fact tables

3 minutes

Typically, a regular data warehouse load operation loads fact tables after dimension tables. This approach ensures that the dimensions to which the facts will be related are already present in the data warehouse.

The staged fact data usually includes the business (alternate) keys for the related dimensions, so your logic to load the data must look up the corresponding surrogate keys. When the data warehouse slowly changing dimensions, the appropriate version of the dimension record must be identified to ensure the correct surrogate key is used to match the event recorded in the fact table with the state of the dimension at the time the fact occurred.

In many cases, you can retrieve the latest "current" version of the dimension; but in some cases you might need to find the right dimension record based on DateTime columns that indicate the period of validity for each version of the dimension.

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

SQL

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

```
SalesAmount  
FROM dbo.StageSales AS stg
```

Next unit: Perform post load optimization

[Continue >](#)

How are we doing?

✓ 100 XP



Perform post load optimization

3 minutes

After loading new data into the data warehouse, it's a good idea to rebuild the table indexes and update statistics on commonly queried columns.

Rebuild indexes

The following example rebuilds all indexes on the **DimProduct** table.

SQL

```
ALTER INDEX ALL ON dbo.DimProduct REBUILD
```

💡 Tip

For more information about rebuilding indexes, see the [Indexes on dedicated SQL pool tables in Azure Synapse Analytics](#) article in the Azure Synapse Analytics documentation.

Update statistics

The following example creates statistics on the **ProductCategory** column of the **DimProduct** table:

SQL

```
CREATE STATISTICS productcategory_stats  
ON dbo.DimProduct(ProductCategory);
```

💡 Tip

For more information about updating statistics, see the [Table statistics for dedicated SQL pool in Azure Synapse Analytics](#) article in the Azure Synapse Analytics documentation.

✓ 200 XP



Knowledge check

3 minutes

Choose the best response for each of the questions, then select **Check your answers**.

1. In which order should you load tables in the data warehouse? *

Staging tables, then dimension tables, then fact tables

✓ That's correct. The correct order of operations is stage, populate dimensions to transform keys and SCDs, then load facts with numeric values.

Staging tables, then fact tables, then dimension tables

✗ That's incorrect. Fact tables should be loaded last with the appropriate surrogate keys from dimensions.

Dimension tables, then staging tables, then fact tables

2. Which command should you use to load a staging table with data from files in the data lake? *

COPY

✓ That's Correct. With existing tables, the copy command is the most efficient way to populate the warehouse from the data lake.

LOAD

INSERT

3. When a customer changes their phone number, the change should be made in the existing row for that customer in the dimension table. What type of slowly changing dimension does this scenario require? *

Type 0

Type 1

✓ That's correct. In this case, simply changing the number with no history is appropriate.

Type 2

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

3 minutes

In this module, you will learn some of the features you can use to manage and monitor Azure Synapse Analytics.

At the end of this module, you will

- Scale compute resources in Azure Synapse Analytics
- Pause compute in Azure Synapse Analytics
- Manage workloads in Azure Synapse Analytics
- Use Azure Advisor to review recommendations
- Use Dynamic Management Views to identify and troubleshoot query performance

Prerequisites

Before taking this module, it is recommended that the student is able to:

- Log into the Azure portal
- Create a Synapse Analytics Workspace
- Create an Azure Synapse Analytics SQL Pool

Next unit: Scale compute resources in Azure Synapse Analytics

[Continue >](#)

How are we doing?

✓ 100 XP ➔

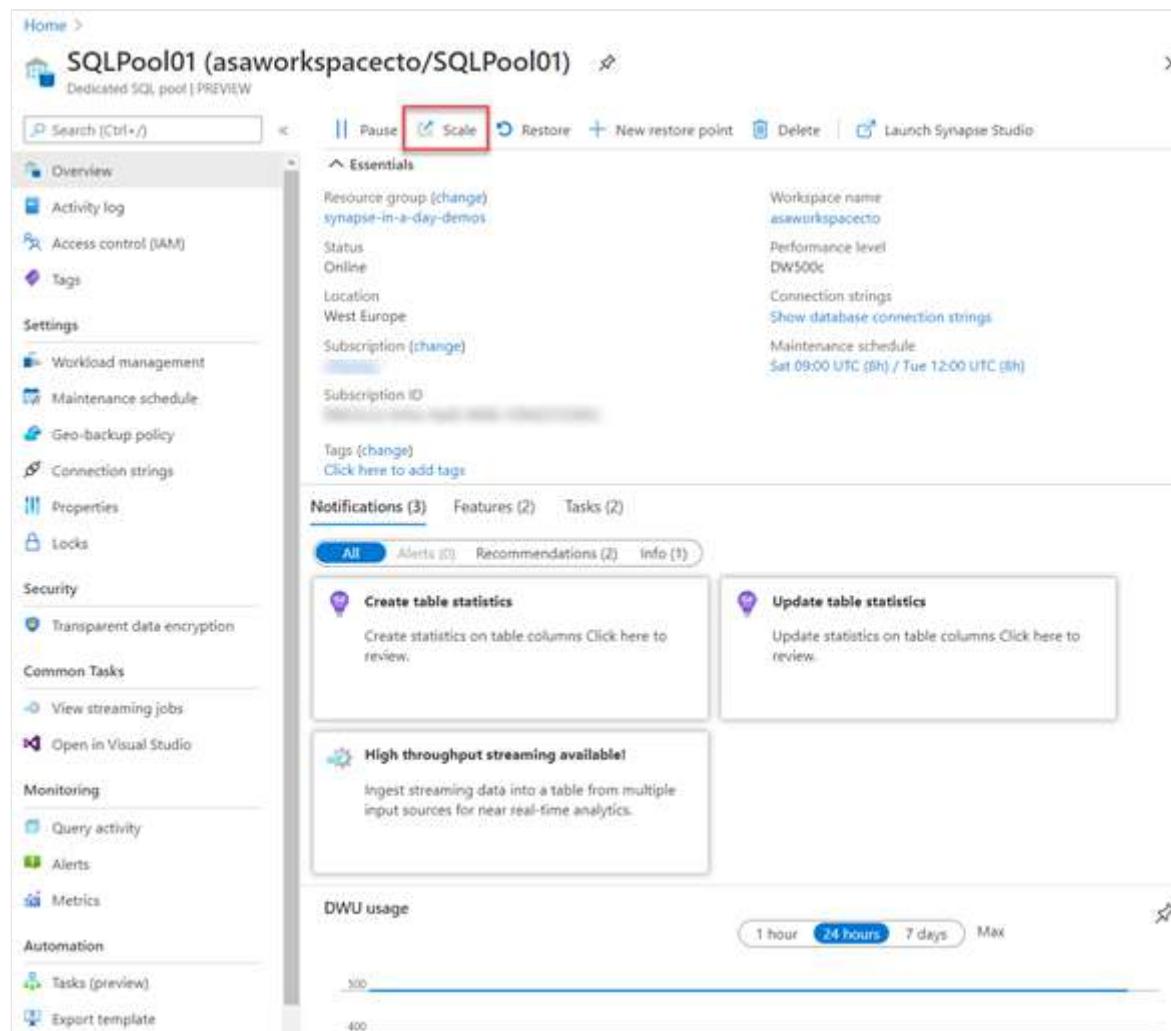
Scale compute resources in Azure Synapse Analytics

8 minutes

One of the key management features that you have at your disposal within Azure Synapse Analytics, is the ability to scale the compute resources for SQL or Spark pools to meet the demands of processing your data. In SQL pools, the unit of scale is an abstraction of compute power that is known as a data warehouse unit. Compute is separate from storage, which enables you to scale compute independently of the data in your system. This means you can scale up and scale down the compute power to meet your needs.

You can scale a Synapse SQL pool either through the Azure portal, Azure Synapse Studio or programmatically using TSQL or PowerShell.

In the Azure portal, you can click on **scale** icon



The screenshot shows the Azure portal interface for managing a SQL pool named 'SQLPool01' within a workspace 'asaworkspacecto'. The top navigation bar includes buttons for Pause, Scale (which is highlighted with a red box), Restore, New restore point, Delete, and Launch Synapse Studio. The main content area displays the 'Essentials' section with details like Resource group, Status (Online), Location (West Europe), Subscription, and Tags. Below this are sections for Notifications (3), Features (2), and Tasks (2). A 'DWU usage' chart at the bottom shows usage over the last 24 hours, with a blue line fluctuating between 300 and 400 DWUs. Callout boxes provide links to 'Create table statistics', 'Update table statistics', and 'High throughput streaming available!'

And then you can adjust the **slider** to scale the SQL Pool

The screenshot shows the 'Workload management' section of the Azure Synapse Analytics portal. At the top, there's a navigation bar with 'Save', 'Discard', 'New workload group', 'Scale', 'Refresh', 'View queries', 'Help', and 'Feedback'. Below the navigation is a 'Scale your system' slider with a green icon. The slider has a value of 20, a minimum of 5, and a maximum of DW500c. A red box highlights this slider area. Below the slider, there are fields for 'Supported concurrency' (set to 1) and 'Min. allowable resources % per request' (set to 1). The unit is 'USD / hour'. Under 'Workload groups', there's a search bar 'Filter by name...' and a dropdown 'User-defined workload groups : All'. A table below shows parameters and effective values for workload groups, with one row for 'Workload group' (DW500c). The table includes columns for 'Workload group', 'Min. resources', 'Min. resources', 'Cap resources', 'Concurrency range', 'Min. resources', 'Min. resources', 'Cap resources', 'Max. resources', and 'Classifiers'. The effective values row shows the same values as the parameters row. A large hexagonal icon is centered on the page, and a message says 'No user-defined workload groups to display. Create "New workload group" or change filter to see system-defined workload groups.'

Another option to scale is within Azure Synapse Studio, click on the **scale** icon:

The screenshot shows the 'Manage' section of the Azure Synapse Studio interface. On the left, there's a sidebar with icons for Home, Data, Develop, Integrate, Monitor, and Manage. The 'Manage' icon is selected. In the main area, under 'Analytics pools', there's a 'SQL pools' section. It says 'Serverless SQL pool is immediately available for your workspace. Dedicated SQL pools can be configured to adapt to team or organizational requirements and constraints.' There are buttons for '+ New', 'Refresh', and '...'. A search bar 'Search to filter items' is also present. Below this, it says 'Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)'. A table lists two items: 'Built-in' (Serverless, Online) and 'SQLPool01' (Dedicated, Online). The 'SQLPool01' row has a red box around its 'scale' icon.

And then move the **slider** as follows:

Scale

SQLPool01

Configure the settings that best align to the workload needs on the dedicated SQL pool. [Learn more about performance levels](#)

Performance level



DW500c

Estimated price ⓘ

Est. cost per hour

You can also make the modification using Transact-SQL

SQL

```
ALTER DATABASE mySampleDataWarehouse  
MODIFY (SERVICE_OBJECTIVE = 'DW300c');
```

Or by using PowerShell

PowerShell

```
Set-AzSqlDatabase -ResourceGroupName "resourcegroupname" -DatabaseName  
"mySampleDataWarehouse" -ServerName "sqlpoolservername" -  
RequestedServiceObjectiveName "DW300c"
```

Scaling Apache Spark pools in Azure Synapse Analytics

Apache Spark pools for Azure Synapse Analytics uses an **Autoscale** feature that automatically scales the number of nodes in a cluster instance up and down. During the creation of a new Spark pool, a minimum and maximum number of nodes can be set when **Autoscale** is selected. Autoscale then monitors the resource requirements of the load and scales the number of nodes up or down. To enable the Autoscale feature, complete the following steps as part of the normal pool creation process:

1. On the **Basics** tab, select the **Enable autoscale** checkbox.

2. Enter the desired values for the following properties:

- Min number of nodes.
- Max number of nodes.

The initial number of nodes will be the minimum. This value defines the initial size of the instance when it's created. The minimum number of nodes can't be fewer than three.

You can also modify this in the Azure portal, you can click on **auto-scale settings** icon

The screenshot shows the Azure portal interface for managing a Spark pool named "SparkPool01". The top navigation bar includes "Home > asaworkspacecto >". The main title is "SparkPool01 (asaworkspacecto/SparkPool01)". Below the title, there are several tabs: "Overview" (selected), "Activity log", "Access control (IAM)", "Tags", "Settings" (selected), "Packages", "Spark configuration", "Properties", and "Locks". On the right side, under the "Essentials" section, there are details: Resource group (change) "synapse-in-a-day-mos", Status "Succeeded", Location "West Europe", Subscription (change) [blue bar], Subscription ID [redacted], and Tags (change) "Click here to add tags". At the top right, there are buttons for "Auto-pause settings", "Auto-scale settings" (which is highlighted with a red box), "Refresh", and "Delete". A search bar at the top left contains "Search (Ctrl+ /)".

Choose the node size and the number of nodes



Auto-scale Settings

SparkPool01



Configure the settings that best align with the workload on the Apache Spark pool.

Autoscale * ⓘ

Enabled [Disabled](#)

Node size family

MemoryOptimized

Node size *

Small (4 vCPU / 32 GB)



Number of nodes *

3



4

Estimated price ⓘ

Est. cost per hour

[View pricing details](#)

Force new settings ⓘ

Immediately apply settings change and cancel all active applications.

[Apply](#)

[Cancel](#)

and for Azure Synapse Studio as follows

The screenshot shows the Azure portal interface. On the left, there's a navigation bar with icons for Home, Data, Develop, Integrate, Monitor, and Manage. The 'Manage' icon is highlighted. The main content area has a header 'Analytics pools' with sub-options: SQL pools, Apache Spark pools (which is selected and highlighted in blue), External connections, Linked services, Integration, Triggers, Integration runtimes, Security, Access control, Credentials, and Managed private endpoints. To the right, under 'Apache Spark pool', it says 'Apache Spark pools can be finely tuned to run different kinds of Apache Spark workloads using specific configuration libraries, permissions, etc.' with a 'Learn more' link. Below that are buttons for '+ New' and 'Refresh'. A search bar says 'Search to filter items'. It shows 'Showing 1 of 1 item' with a table row for 'SparkPool01'. The 'Size' column indicates 'Small (4 vCores / 32 GB) - 3 to 4 nodes'. A red box highlights the edit icon (pencil) in the 'Actions' column of the table.

And Choose the node size and the number of nodes

The screenshot shows the 'Autoscale settings' page for 'SparkPool01'. It includes a section for 'Node size family' set to 'MemoryOptimized', a dropdown for 'Node size' set to 'Small (4 vCores / 32 GB)', and an 'Autoscale' toggle switch set to 'Enabled'. Below these are sections for 'Number of nodes' (set to 3) and 'Estimated price' (with a 'Est. cost per hour' button). A green bar at the bottom indicates the estimated cost per hour.

Autoscale continuously monitors the Spark instance and collects the following metrics:

Metric	Description
Total Pending CPU	The total number of cores required to start execution of all pending nodes.

Metric	Description
Total Pending Memory	The total memory (in MB) required to start execution of all pending nodes.
Total Free CPU	The sum of all unused cores on the active nodes.
Total Free Memory	The sum of unused memory (in MB) on the active nodes.
Used Memory per Node	The load on a node. A node on which 10 GB of memory is used, is considered under more load than a worker with 2 GB of used memory.

The following conditions will then autoscale the memory or CPU

Scale-up	Scale-down
Total pending CPU is greater than total free CPU for more than 1 minute.	Total pending CPU is less than total free CPU for more than 2 minutes.
Total pending memory is greater than total free memory for more than 1 minute.	Total pending memory is less than total free memory for more than 2 minutes.

The scaling operation can take between 1 -5 minutes. During an instance where there is a scale down process, Autoscale will put the nodes in decommissioning state so that no new executors can launch on that node.

The running jobs will continue to run and finish. The pending jobs will wait to be scheduled as normal with fewer available nodes.

Next unit: Pause compute in Azure Synapse Analytics

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

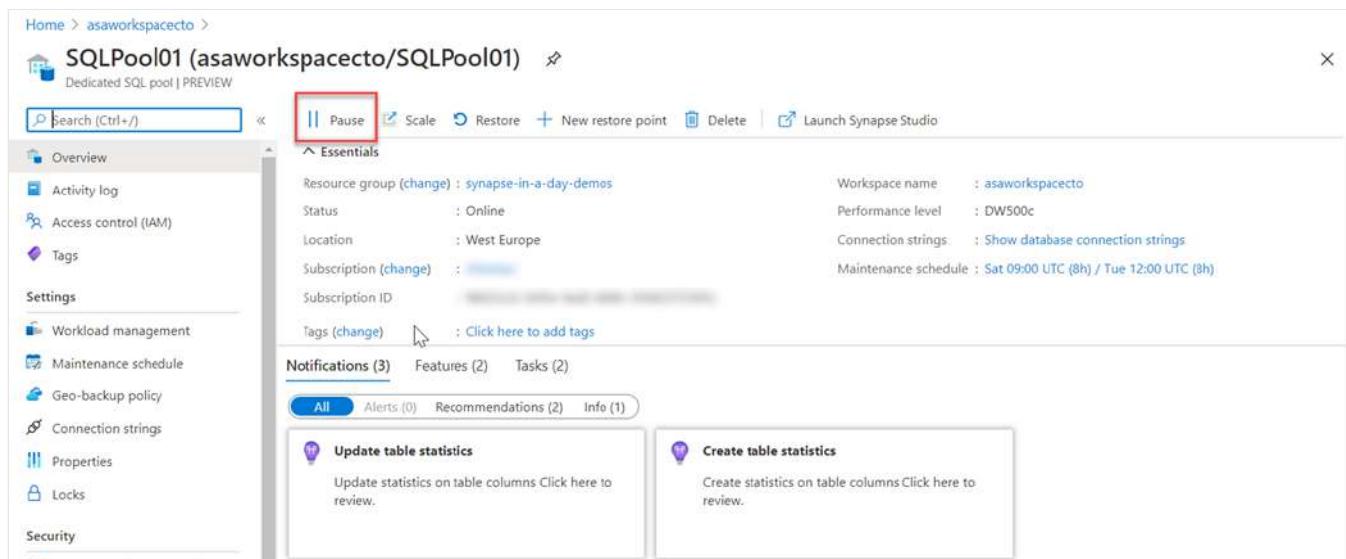
✓ 100 XP

Pause compute in Azure Synapse Analytics

3 minutes

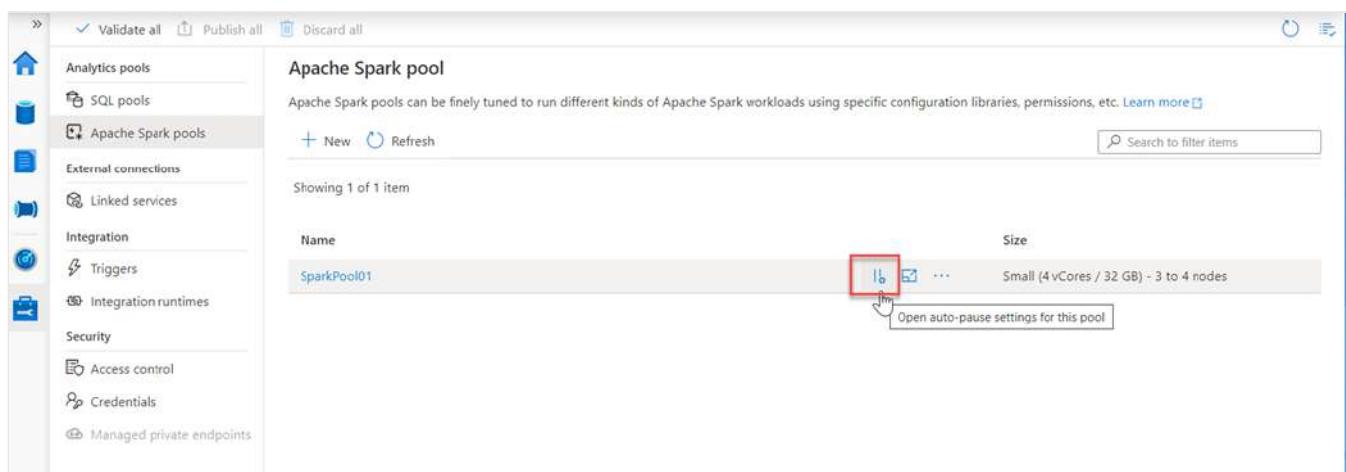
When performing the batch movement of data to populate a data warehouse, it is typical for the data engineer to understand the schedule on which the data loads take place. In these circumstances, you may be able to predict the periods of downtime in the data loading and querying process and take advantage of the pause operations to minimize your costs.

In the Azure portal you can use the Pause command within the dedicated SQL pool



The screenshot shows the Azure portal interface for managing a dedicated SQL pool named 'SQLPool01'. The top navigation bar includes a 'Pause' button, which is highlighted with a red box. Below the navigation bar, there's a search bar and several management options like 'Scale', 'Restore', 'New restore point', 'Delete', and 'Launch Synapse Studio'. The main content area displays basic pool details such as resource group, status, location, subscription, workspace name (asaworkspaceto), performance level (DWS00c), connection strings, and maintenance schedule. On the left, a sidebar lists various settings like workload management, maintenance schedule, geo-backup policy, connection strings, properties, locks, and security. At the bottom, there are sections for notifications, features, and tasks, along with buttons for 'All', 'Alerts (0)', 'Recommendations (2)', and 'Info (1)'.

And this can also be used within Azure Synapse Studio for Apache Spark pools too, in the Manage hub.



The screenshot shows the 'Manage' hub in Azure Synapse Studio. The left sidebar lists categories like Analytics pools, SQL pools, External connections, Integration, Triggers, Integration runtimes, Security, Access control, Credentials, and Managed private endpoints. The main panel is titled 'Apache Spark pool' and contains a sub-section for 'Apache Spark pools'. It states that these pools can be finely tuned to run different kinds of Apache Spark workloads using specific configuration libraries, permissions, etc. A 'Learn more' link is provided. Below this, there are buttons for '+ New' and 'Refresh'. A search bar at the top right allows filtering items. The main list shows one item named 'SparkPool01'. To the right of this item, there are three icons: a pause button (highlighted with a red box), a copy icon, and a more options icon. A tooltip for the pause icon says 'Open auto-pause settings for this pool'.

Which allows you to enable it, and set the number of minutes idle

Auto-pause settings



Configure the auto-pause settings for the Apache Spark pool.

Auto-pause * ⓘ

Enabled

Disabled

Number of minutes idle *

15

Next unit: Manage workloads in Azure Synapse Analytics

[Continue >](#)

How are we doing? ⭐ ⭐ ⭐ ⭐ ⭐

100 XP



Manage workloads in Azure Synapse Analytics

10 minutes

Azure Synapse Analytics allows you to create, control and manage resource availability when workloads are competing. This allows you to manage the relative importance of each workload when waiting for available resources.

To facilitate faster load times, you can create a workload classifier for the load user with the “importance” set to above_normal or High. Workload importance ensures that the load takes precedence over other waiting tasks of a lower importance rating. Use this in conjunction with your own workload group definitions for workload isolation to manage minimum and maximum resource allocations during peak and quiet periods.

Dedicated SQL pool workload management in Azure Synapse consists of three high-level concepts:

- Workload Classification
- Workload Importance
- Workload Isolation

These capabilities give you more control over how your workload utilizes system resources.

Workload classification

Workload management classification allows workload policies to be applied to requests through assigning resource classes and importance.

While there are many ways to classify data warehousing workloads, the simplest and most common classification is load and query. You load data with insert, update, and delete statements. You query the data using selects. A data warehousing solution will often have a workload policy for load activity, such as assigning a higher resource class with more resources. A different workload policy could apply to queries, such as lower importance compared to load activities.

You can also subclassify your load and query workloads. Subclassification gives you more control of your workloads. For example, query workloads can consist of cube refreshes, dashboard queries or ad-hoc queries. You can classify each of these query workloads with

different resource classes or importance settings. Load can also benefit from subclassification. Large transformations can be assigned to larger resource classes. Higher importance can be used to ensure key sales data is loaded before weather data or a social data feed.

Not all statements are classified as they do not require resources or need importance to influence execution. DBCC commands, BEGIN, COMMIT, and ROLLBACK TRANSACTION statements are not classified.

Workload importance

Workload importance influences the order in which a request gets access to resources. On a busy system, a request with higher importance has first access to resources. Importance can also ensure ordered access to locks. There are five levels of importance: low, below_normal, normal, above_normal, and high. Requests that don't set importance are assigned the default level of normal. Requests that have the same importance level have the same scheduling behavior that exists today.

Workload isolation

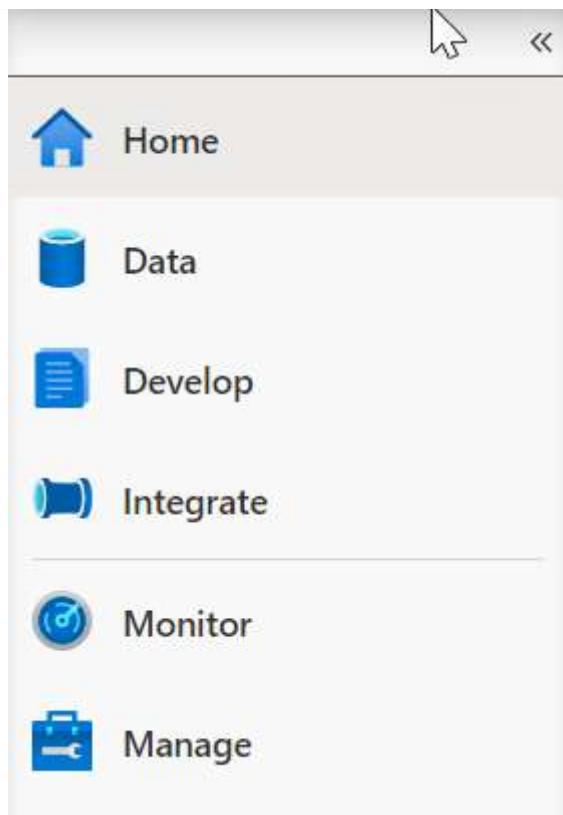
Workload isolation reserves resources for a workload group. Resources reserved in a workload group are held exclusively for that workload group to ensure execution. Workload groups also allow you to define the amount of resources that are assigned per request, much like resource classes do. Workload groups give you the ability to reserve or cap the amount of resources a set of requests can consume. Finally, workload groups are a mechanism to apply rules, such as query timeout, to requests.

You can perform the following steps to implement workload management

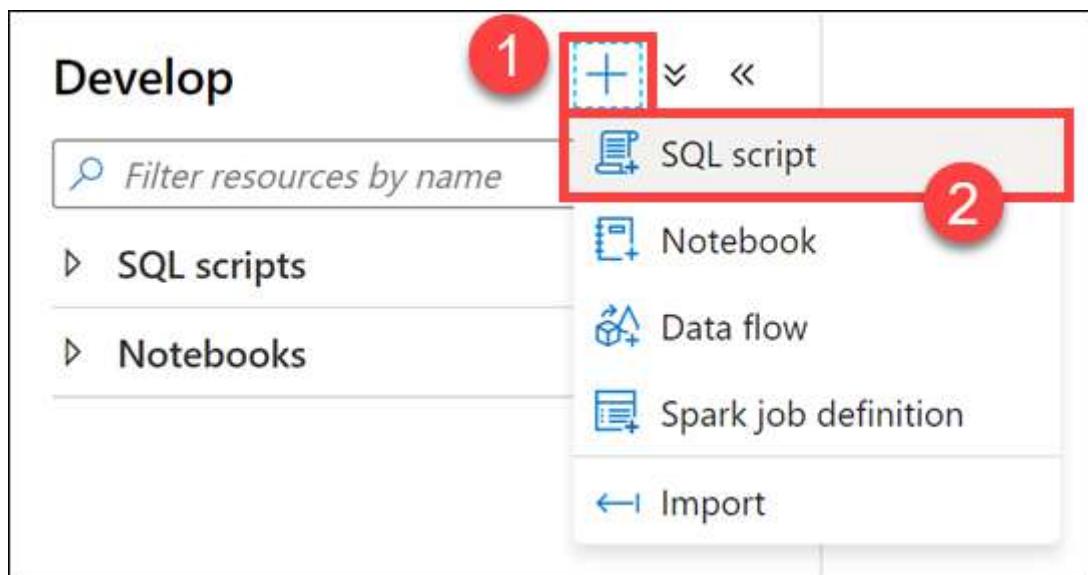
Create a workload classifier to add importance to certain queries

Your organization has asked you if there is a way to mark queries executed by the CEO as more important than others, so they don't appear slow due to heavy data loading or other workloads in the queue. You decide to create a workload classifier and add importance to prioritize the CEO's queries.

1. Select the Develop hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



3. In the toolbar menu, connect to the **SQL Pool** database to execute the query.



4. In the query window, replace the script with the following to confirm that there are no queries currently being run by users logged in as `asa.sql.workload01`, representing the CEO of the organization or `asa.sql.workload02` representing the data analyst working on the project:

```
SQL
```

```
--First, let's confirm that there are no queries currently being run  
by users logged in workload01 or workload02
```

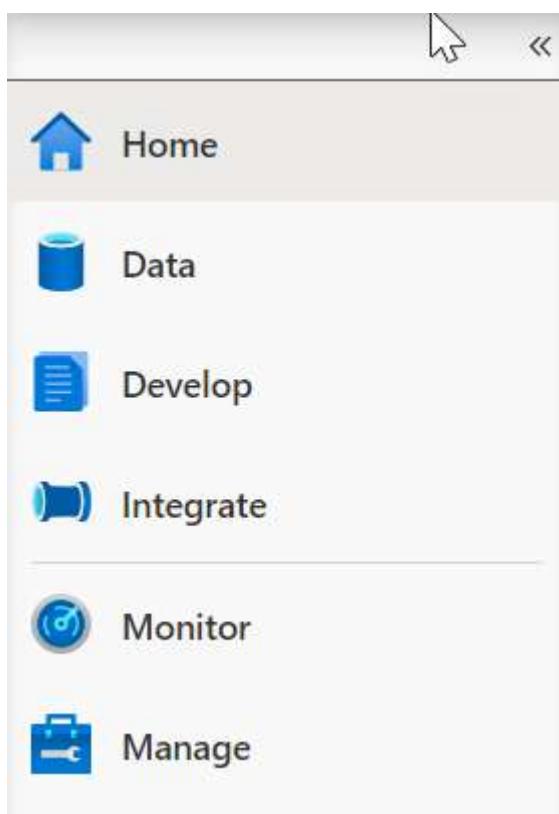
```
SELECT s.login_name, r.[Status], r.Importance, submit_time,  
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s  
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id  
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and  
Importance  
is not NULL AND r.[status] in ('Running','Suspended')  
--and submit_time>dateadd(minute,-2,getdate())  
ORDER BY submit_time ,s.login_name
```

5. Select **Run** from the toolbar menu to execute the SQL command.

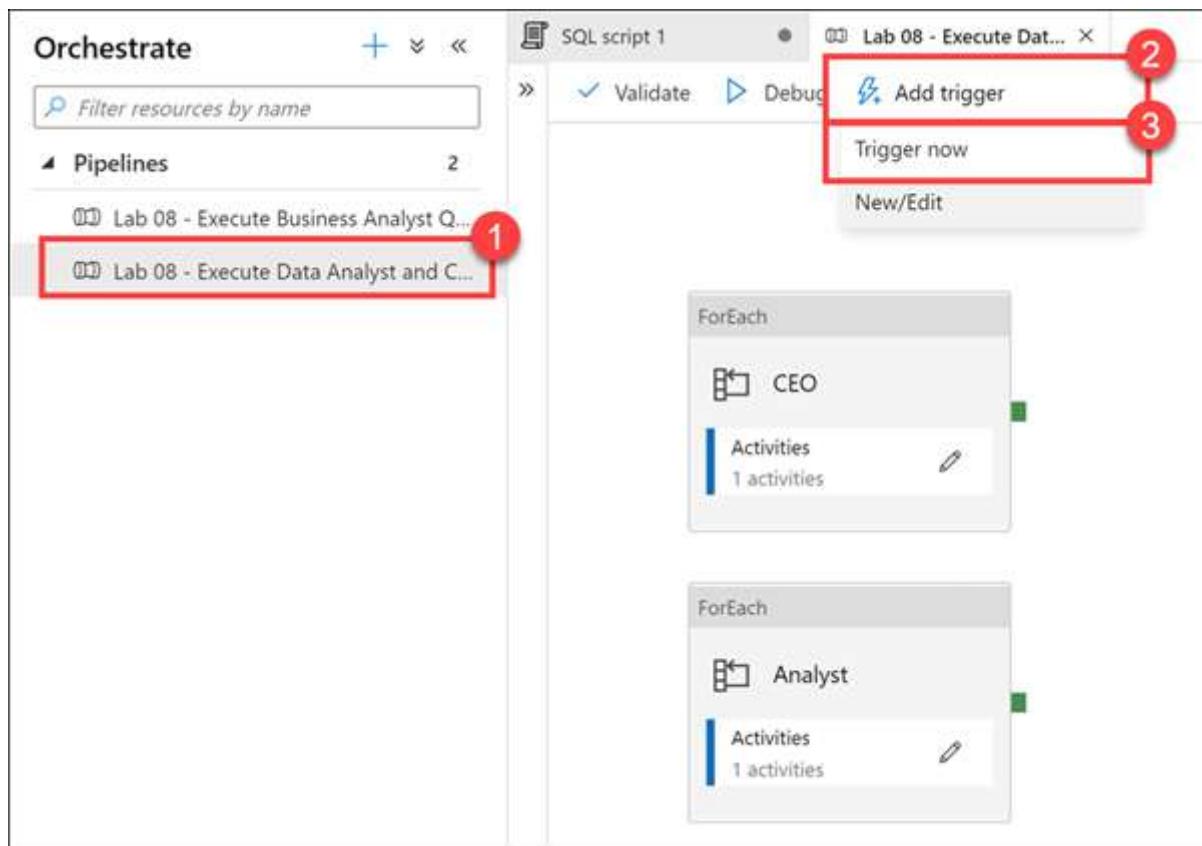


Now that we have confirmed that there are no running queries, we need to flood the system with queries and see what happens for `asa.sql.workload01` and `asa.sql.workload02`. To do this, we'll run a Azure Synapse Pipeline which triggers queries.

6. Select the **Integrate** hub.



7. Select the **Lab 08 - Execute Data Analyst and CEO Queries Pipeline** (1), which will run / trigger the `asa.sql.workload01` and `asa.sql.workload02` queries. Select **Add trigger** (2), then **Trigger now** (3). In the dialog that appears, select **OK**.



8. Let's see what happened to all the queries we just triggered as they flood the system. In the query window, replace the script with the following:

SQL

```
SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time
, s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and
Importance
is not NULL AND r.[status] in ('Running','Suspended') and
submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status
```

9. Select **Run** from the toolbar menu to execute the SQL command.



You should see an output similar to the following:

```
3 WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance  
4 is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(minute,-  
5 ORDER BY submit_time ,status
```

Results Messages

View Table Chart Export results

Search

Login_name	Status	Importance	Subm
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-

Notice that the **Importance** level for all queries is set to **normal**.

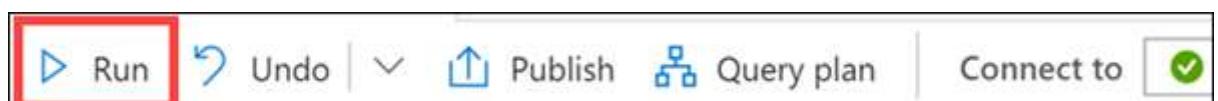
10. We will give our `asa.sql.workload01` user queries priority by implementing the **Workload Importance** feature. In the query window, replace the script with the following:

```
SQL

IF EXISTS (SELECT * FROM sys.workload_management_workload_classifiers
WHERE name = 'CEO')
BEGIN
    DROP WORKLOAD CLASSIFIER CEO;
END
CREATE WORKLOAD CLASSIFIER CEO
    WITH (WORKLOAD_GROUP = 'largerc'
    ,MEMBERNAME = 'asa.sql.workload01',IMPORTANCE = High);
```

We are executing this script to create a new **Workload Classifier** named `CEO` that uses the `largerc` Workload Group and sets the **Importance** level of the queries to `High`.

11. Select **Run** from the toolbar menu to execute the SQL command.



12. Let's flood the system again with queries and see what happens this time for `asa.sql.workload01` and `asa.sql.workload02` queries. To do this, we'll run an Azure Synapse Pipeline which triggers queries. Select the Integrate Tab, run the **Lab 08 - Execute Data Analyst and CEO Queries** Pipeline, which will run / trigger the `asa.sql.workload01` and `asa.sql.workload02` queries.

13. In the query window, replace the script with the following to see what happens to the `asa.sql.workload01` queries this time:

SQL

```
SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time
, s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and
Importance
is not NULL AND r.[status] in ('Running','Suspended') and
submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status desc
```

14. Select **Run** from the toolbar menu to execute the SQL command.



You should see an output similar to the following:

```
3 WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importanc  
4 is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(m  
5 ORDER BY submit_time ,status desc
```

Results Messages

View

Table

Chart

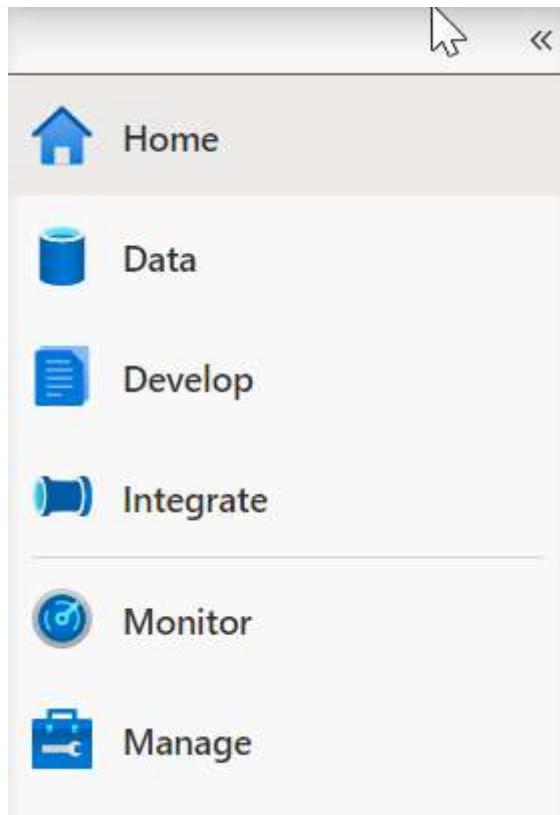
Export results ▾

🔍 Search

Login_name	Status	Importance
asa.sql.workload01	Suspended	high
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal
asa.sql.workload01	Suspended	high
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal

Notice that the queries executed by the `asa.sql.workload01` user have a **high** importance.

15. Select the Monitor hub.



16. Select Pipeline runs (1), and then select Cancel recursive (2) for each running Lab 08 pipelines, marked In progress (3). This will help speed up the remaining tasks.

Pipeline name	Run start	Run end	Duration	Triggered by	Status
Lab 08 - Execute Data Analyst	9/8/20, 11:19:04 PM	--	00:13:47	Manual trigger	In progress
Lab 08 - Execute Data Analyst	9/8/20, 11:23:35 PM	--	00:16:16	Manual trigger	In progress
Lab 08 - Execute Data Analyst	9/8/20, 11:12:26 PM	--	00:20:25	Manual trigger	In progress
Lab 08 - Execute Data Analyst	9/8/20, 11:03:19 PM	--	00:29:32	Manual trigger	In progress
Setup - Load SQL Pool	9/8/20, 2:49:12 PM	9/8/20, 2:49:47 PM	00:00:34	Manual trigger	Succeeded
Setup - Load SQL Pool (global)	9/8/20, 1:54:04 PM	9/8/20, 2:33:05 PM	00:39:01	Manual trigger	Succeeded

Reserve resources for specific workloads through workload isolation

Workload isolation means resources are reserved, exclusively, for a workload group. Workload groups are containers for a set of requests and are the basis for how workload management, including workload isolation, is configured on a system. A simple workload management configuration can manage data loads and user queries.

In the absence of workload isolation, requests operate in the shared pool of resources. Access to resources in the shared pool is not guaranteed and is assigned on an importance basis.

Given the workload requirements provided by Tailwind Traders, you decide to create a new workload group called CEODemo to reserve resources for queries executed by the CEO.

Let's start by experimenting with different parameters.

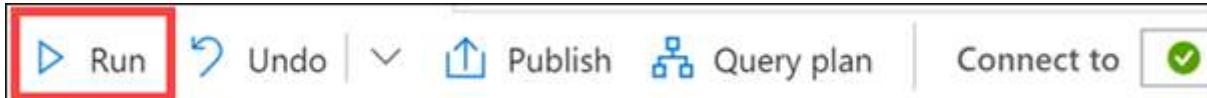
1. In the query window, replace the script with the following:

```
SQL

IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_groups
where name = 'CEODemo')
BEGIN
    Create WORKLOAD GROUP CEODemo WITH
    ( MIN_PERCENTAGE_RESOURCE = 50          -- integer value
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 25 --
    ,CAP_PERCENTAGE_RESOURCE = 100
    )
END
```

The script creates a workload group called `CEODemo` to reserve resources exclusively for the workload group. In this example, a workload group with a `MIN_PERCENTAGE_RESOURCE` set to 50% and `REQUEST_MIN_RESOURCE_GRANT_PERCENT` set to 25% is guaranteed 2 concurrency.

2. Select **Run** from the toolbar menu to execute the SQL command.



3. In the query window, replace the script with the following to create a Workload Classifier called `CEODreamDemo` that assigns a workload group and importance to incoming requests:

```
SQL

IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_classifiers
where name = 'CEODreamDemo')
BEGIN
    Create Workload Classifier CEODreamDemo with
    ( Workload_Group ='CEODemo',MemberName='asa.sql.workload02',IMPORTANCE = BELOW_NORMAL);
END
```

This script sets the Importance to `BELLOW_NORMAL` for the `asa.sql.workload02` user, through the new `CEODreamDemo` Workload Classifier.

4. Select **Run** from the toolbar menu to execute the SQL command.



5. In the query window, replace the script with the following to confirm that there are no active queries being run by asa.sql.workload02 (suspended queries are OK):

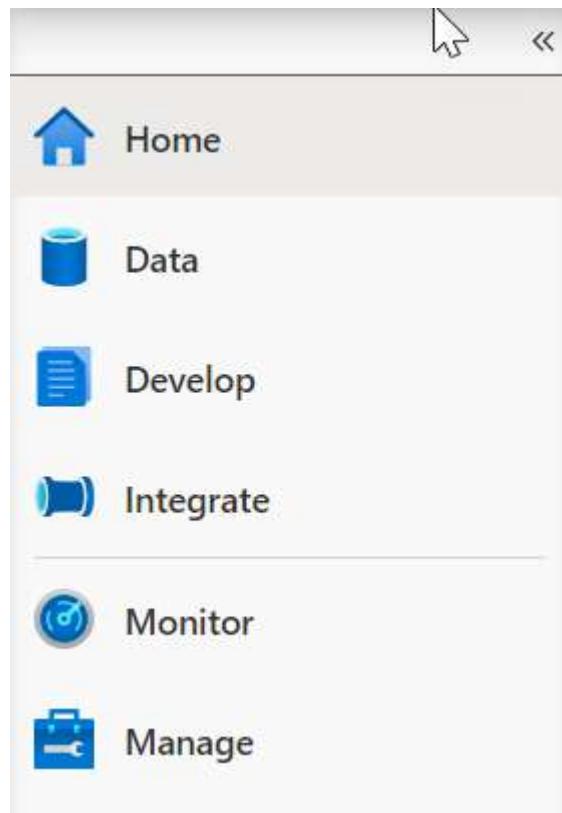
SQL

```
SELECT s.login_name, r.[Status], r.Importance, submit_time,  
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s  
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id  
WHERE s.login_name IN ('asa.sql.workload02') and Importance  
is not NULL AND r.[status] in ('Running','Suspended')  
ORDER BY submit_time, status
```

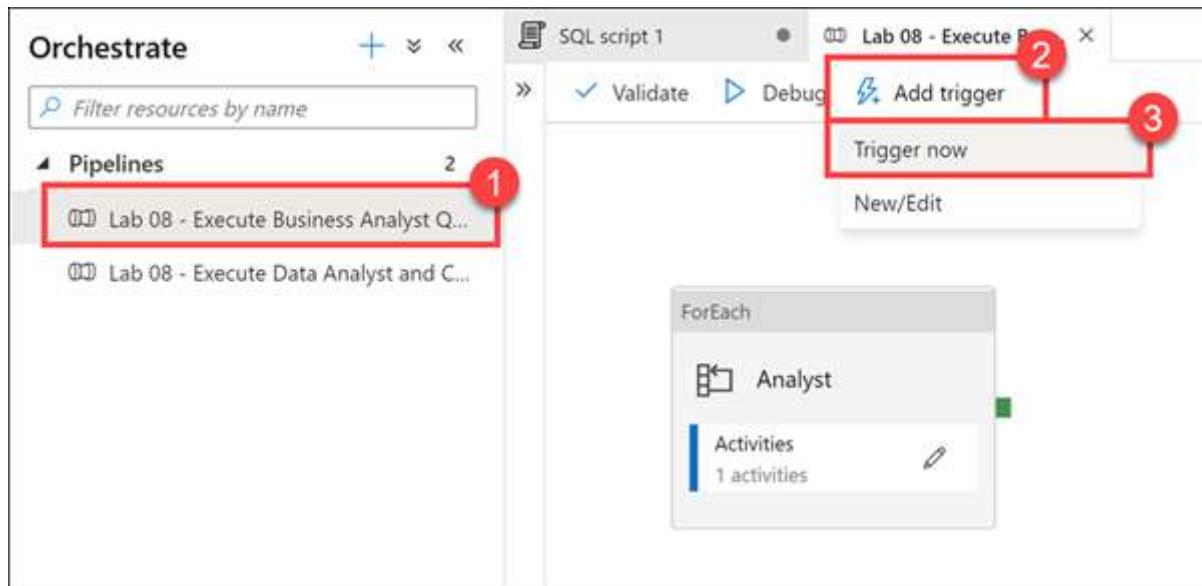
6. Select **Run** from the toolbar menu to execute the SQL command.



7. Select the **Integrate** hub.



8. Select the **Lab 08 - Execute Business Analyst Queries Pipeline** (1), which will run / trigger asa.sql.workload02 queries. Select **Add trigger** (2), then **Trigger now** (3). In the dialog that appears, select **OK**.



9. In the query window, replace the script with the following to see what happened to all the `asa.sql.workload02` queries we just triggered as they flood the system:

SQL

```
SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

10. Select Run from the toolbar menu to execute the SQL command.



You should see an output similar to the following that shows the importance for each session set to `below_normal`:

```

4 WHERE s.login_name IN ('asa.sql.workload02') and Importance
5 is not NULL AND r.[status] in ('Running','Suspended')
6 ORDER BY submit_time, status

```

Results Messages

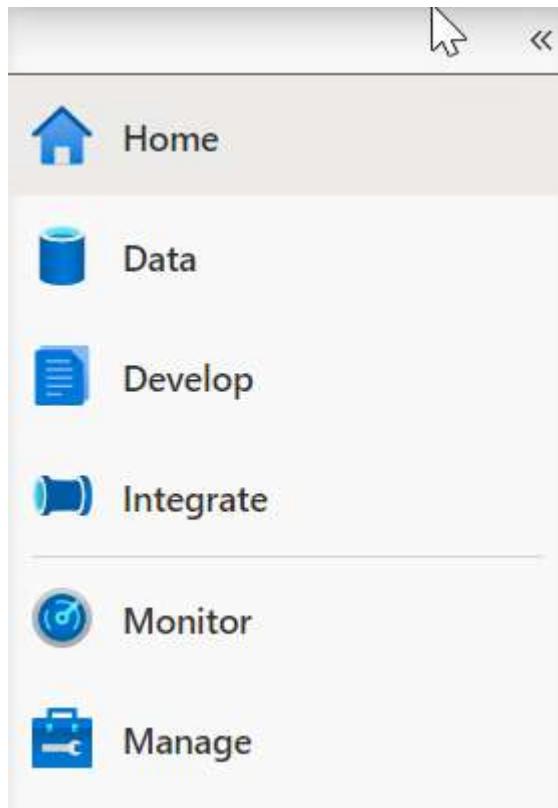
View **Table** Chart Export results

Search

Login_name	Status	Importance	Subn
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020

Notice that the running scripts are executed by the `asa.sql.workload02` user (1) with an Importance level of `below_normal` (2). We have successfully configured the business analyst queries to execute at a lower importance than the CEO queries. We can also see that the `CEOdreamDemo` Workload Classifier works as expected.

11. Select the Monitor hub.



12. Select Pipeline runs (1), and then select Cancel recursive (2) for each running Lab 08 pipelines, marked In progress (3). This will help speed up the remaining tasks.

Pipeline runs

Triggered Debug Rerun Cancel Refresh Edit columns

End time : Last 24 hours (9/7/20 11:38 PM - 9/8/20 11:38 PM) Time zone : Eastern Time (US & Canada) (UT... Status : All status Runs : Latest runs

Showing 1 - 10 items

Pipeline name	Run start	Run end	Duration	Triggered by	Status
Lab 08 - Execute B... (1)	9/9/20, 10:03:02 AM	--	00:05:24	Manual trigger	In progress (3)
Lab 08 - Execute Data Analyst ... (2)	9/8/20, 11:39:04 PM	9/8/20, 11:39:22 PM	00:20:18	Manual trigger	Cancelled (4)
Lab 08 - Execute Data Analyst ... (3)	9/8/20, 11:16:35 PM	9/8/20, 11:39:19 PM	00:22:44	Manual trigger	Cancelled (5)

13. Return to the query window under the **Develop** hub. In the query window, replace the script with the following to set 3.25% minimum resources per request:

SQL

```
IF EXISTS (SELECT * FROM sys.workload_management_workload_classifiers
where group_name = 'CEODemo')
BEGIN
    Drop Workload Classifier CEODreamDemo
    DROP WORKLOAD GROUP CEODemo
    --- Creates a workload group 'CEODemo'.
    Create WORKLOAD GROUP CEODemo WITH
        (MIN_PERCENTAGE_RESOURCE = 26 -- integer value
         ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 3.25 -- factor of 26
        (guaranteed more than 4 concurrencies)
         ,CAP_PERCENTAGE_RESOURCE = 100
        )
    --- Creates a workload Classifier 'CEODreamDemo'.
    Create Workload Classifier CEODreamDemo with
        (Workload_Group ='CEODemo',MemberName='asa.sql.workload02',IMPORTANCE = BELOW_NORMAL);
END
```

① Note

Configuring workload containment implicitly defines a maximum level of concurrency. With a CAP_PERCENTAGE_RESOURCE set to 60% and a REQUEST_MIN_RESOURCE_GRANT_PERCENT set to 1%, up to a 60-concurrency level is allowed for the workload group. Consider the method included below for determining the maximum concurrency: [Max Concurrency] = [CAP_PERCENTAGE_RESOURCE] / [REQUEST_MIN_RESOURCE_GRANT_PERCENT]

14. Select **Run** from the toolbar menu to execute the SQL command.



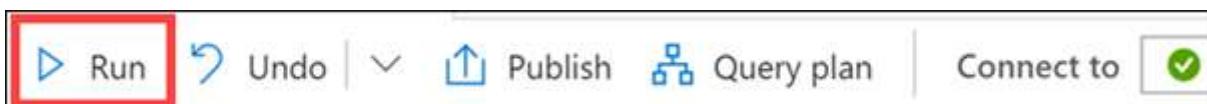
15. Let's flood the system again and see what happens for `asa.sql.workload02`. To do this, we will run an Azure Synapse Pipeline which triggers queries. Select the Integrate Tab. Run the **Lab 08 - Execute Business Analyst Queries** Pipeline, which will run / trigger `asa.sql.workload02` queries.

16. In the query window, replace the script with the following to see what happened to all of the `asa.sql.workload02` queries we just triggered as they flood the system:

SQL

```
SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

17. Select **Run** from the toolbar menu to execute the SQL command.



After several moments (up to a minute), we should see several concurrent executions by the `asa.sql.workload02` user running at **below_normal** importance. We have validated that the modified Workload Group and Workload Classifier works as expected.

```
4 WHERE s.login_name IN ('asa.sql.workload02') and Importance  
5 is not NULL AND r.[status] in ('Running', 'Suspended')  
6 ORDER BY submit_time, status
```

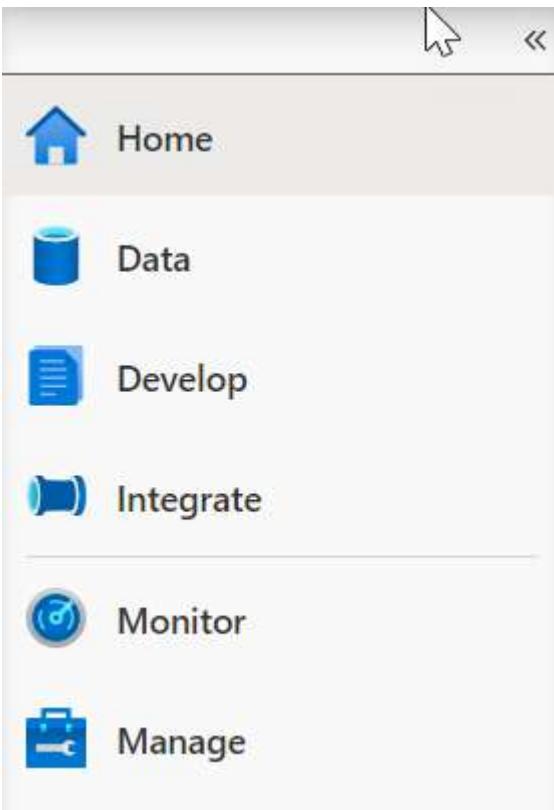
Results Messages

View Table Chart Export results

Search

Login_name	Status	Importance
asa.sql.workload02	Running	below_normal

18. Select the Monitor hub.



19. Select Pipeline runs (1), and then select Cancel recursive (2) for each running Lab 08 pipelines, marked In progress (3). This will help speed up the remaining tasks.

Pipeline runs

Triggered Debug Rerun Cancel Refresh Edit columns

End time : Last 24 hours (9/7/20 4:26 PM - 9/8/20 4:26 PM) Time zone : Eastern Time (US & Canada) (UT...) Status : All status Runs : Latest runs

Showing 1 - 9 items

Pipeline name	Run start	Run end	Duration	Triggered by	Status
Lab 08 - Execute Data Analyst...	9/8/20, 11:19:04 PM	--	00:13:47	Manual trigger	In progress
Lab 08 - Execute Data Analyst...	9/8/20, 11:20:35 PM	--	00:16:16	Manual trigger	In progress
Lab 08 - Execute Data Analyst...	9/8/20, 11:22:26 PM	--	00:20:25	Manual trigger	In progress
Lab 08 - Execute Data Analyst...	9/8/20, 11:03:19 PM	--	00:29:32	Manual trigger	In progress
Setup - Load SQL Pool	9/8/20, 2:49:12 PM	9/8/20, 2:49:47 PM	00:00:34	Manual trigger	Succeeded
Setup - Load SQL Pool (global)	9/8/20, 1:54:04 PM	9/8/20, 2:33:05 PM	00:39:01	Manual trigger	Succeeded

Next unit: Use Azure Advisor to review recommendations

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

Use Azure Advisor to review recommendations

9 minutes

Azure Advisor provides you with personalized messages that provide information on best practices to optimize the setup of your Azure services. It analyzes your resource configuration and usage telemetry and then recommends solutions that can help you improve the cost effectiveness, performance, Reliability (formerly called High availability), and security of your Azure resources.

The Advisor may appear when you log into the Azure portal, but you can also access the Advisor by selecting Advisor in the navigation menu.

The screenshot shows the Azure portal interface. On the left, the navigation menu is open, with the 'Advisor' option highlighted and a red box drawn around it. The main content area displays the 'All services' dashboard. At the top, there's a search bar and several quick access icons: Resource groups, Subscriptions, All resources, Service providers, Azure Active Directory, Cognitive Services, and Security Center. Below these are sections for 'Services' and 'Resources'. The 'Resources' section contains a table with columns for 'Name' and 'Type'. The table lists various Azure resources, including Synapse workspace, Dedicated SQL pool, Apache Spark pool, Storage account, Resource group, and Subscription. One row in the table is highlighted with a blue background.

Name	Type
asawspaceto (preview)	Synapse workspace
l01 (asawspaceto/SQLPool01) (preview)	Dedicated SQL pool
pool01 (asawspaceto/SparkPool01) (preview)	Apache Spark pool
ectows (preview)	Synapse workspace
awsctodd1	Storage account
aparseRG	Resource group
-in-a-day-demos	Resource group
pacemanagedrg-bf3e530	Resource group
o	Subscription
37ffe99f499a6	Storage account

On accessing Advisor, a dashboard is presented that provides recommendations in the following areas:

- Cost
- Security

- Reliability
- Operational excellence
- Performance

The screenshot shows the Azure Advisor Overview dashboard. On the left, there's a sidebar with links for Home, Overview, Advisor Score (preview), Recommendations (Cost, Security, Reliability, Operational excellence, Performance, All recommendations), Monitoring (Alerts (Preview), Recommendation digests), and Settings (Configuration). The main area has a header with Search, Feedback, Download options, and a Try the new Advisor Score (preview) button. Below the header, there's a message: "Try the new Advisor score experience to better prioritize recommendations and measure impact." The dashboard is divided into several cards:

- Cost:** Shows 9 USD savings/yr.* with 1 Recommendation (0 High impact, 0 Medium impact, 1 Low impact).
- Security:** Shows 16 Recommendations (7 High impact, 4 Medium impact, 5 Low impact).
- Reliability:** Shows a green checkmark indicating "You are following all of our reliability recommendations".
- Operational excellence:** Shows a green checkmark indicating "You are following all of our operational excellence recommendations".
- Performance:** Shows 2 Recommendations (2 High impact, 0 Medium impact, 0 Low impact) and 1 Impacted resource.
- Tips & tricks:** A card with a rocket icon and the text "Try Advisor Score" and "Preview the new Advisor score experience today. Easily prioritize recommendations, track progress, and measure impact." It includes "Try now" and download buttons for PDF and CSV.

You can click on any of the dashboard items for more information. In the following example, the performance dashboard item is showing more information on two high impact items in Azure Synapse Analytics.

The screenshot shows the Azure Advisor Performance dashboard. The sidebar is identical to the Overview dashboard. The main area has a header with Search, Feedback, Download options, and buttons for Create alert, Create recommendation digest, and Try the new Advisor Score (preview). There's also a message about subscriptions: "Subscriptions: 1 of 15 selected – Don't see a subscription? Open Directory + Subscription settings". Below this is a search bar with "chtestao" and dropdowns for Active and No grouping. The dashboard displays recommendations by impact:

Total recommendations	Recommendations by impact	Impacted resources
2	2 High impact, 0 Medium impact, 0 Low impact	1

Below this is a table of recommendations:

Impact	Description	Potential benefits	Impacted resources	Last updated
High	Update statistics on table columns	Increase query performance	data warehouse	11/23/2020, 06:04 AM
High	Create statistics on table columns	Increase query performance	data warehouse	11/23/2020, 06:04 AM

You can also click on each item to get even more information that can help you resolve the issue. In the following example, this is the information that is shown when clicking on the

Create statistics on table columns recommendation.

The screenshot shows the Azure Advisor interface for creating table statistics. At the top, there are links for Home > Advisor > Create table statistics. Below this, there are buttons for Feedback, Download as CSV, Download as PDF, and Create alert. A section titled 'Recommendation details' explains that missing table statistics can impact query performance. Under 'Impacted resources', there is a table with one row:

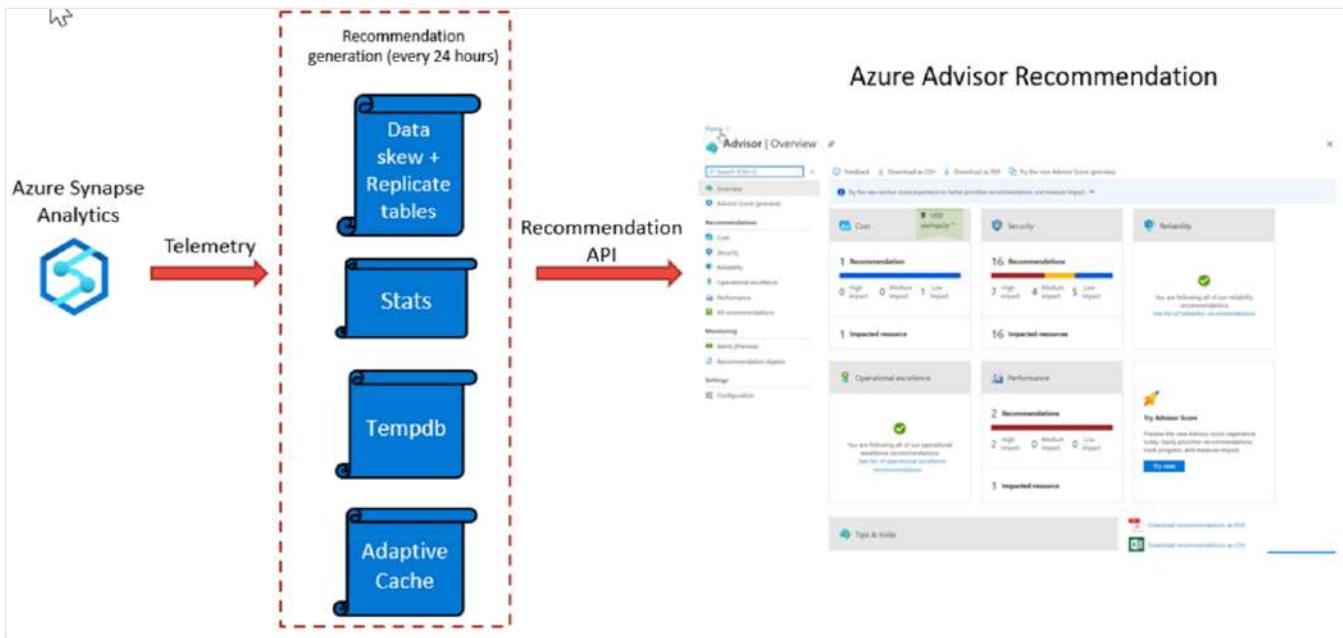
Select	Sql data warehouse	Recommended actions	Subscription	Total impacted tables	Last updated	Action
<input type="checkbox"/>	sqlpool01	View impacted tables Create table statistics	[redacted]	3	11/23/2020, 06:04 AM	Postpone Dismiss

In this screen, you can click on the **view impacted tables** to see which tables are being impacted specifically, and there are also links to the help in the Azure documentation that you can use to get more understanding of the issue.

How Azure Synapse Analytics works with Azure Advisor

Azure Advisor recommendations are free, and the recommendations are based on telemetry data that is generated by Azure Synapse Analytics. The telemetry data that is captured by Azure Synapse Analytics include

- Data Skew and replicated table information.
- Column statistics data.
- TempDB utilization data.
- Adaptive Cache.



Azure Advisor recommendations are checked every 24 hours, as the recommendation API is queried against the telemetry generated from with Azure Synapse Analytics, and the recommendation dashboards are then updated to reflect the information that the telemetry has generated. This can then be viewed in the Azure Advisor dashboard.

Next unit: Use dynamic management views to identify and troubleshoot query performance

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Use dynamic management views to identify and troubleshoot query performance

8 minutes

Dynamic Management Views provide a programmatic experience for monitoring the Azure Synapse Analytics SQL pool activity by using the Transact-SQL language. The views that are provided, not only enable you to troubleshoot and identify performance bottlenecks with the workloads working on your system, but they are also used by other services such as Azure Advisor to provide recommendations about Azure Synapse Analytics.

There are over 90 Dynamic Management Views that can queried against dedicated SQL pools to retrieve information about the following areas of the service:

- Connection information and activity
- SQL execution requests and queries
- Index and statistics information
- Resource blocking and locking activity
- Data movement service activity
- Errors

The following is an example of monitoring query execution of the Azure Synapse Analytics SQL pools. The first step involves checking the connections against the server first, before checking the query execution activity.

Monitoring connections

All logins to your data warehouse are logged to sys.dm_pdw_exec_sessions. The session_id is the primary key and is assigned sequentially for each new logon.

SQL

```
-- Other Active Connections
SELECT * FROM sys.dm_pdw_exec_sessions where status <> 'Closed' and session_id <> session_id();
```

Monitor query execution

All queries executed on SQL pool are logged to sys.dm_pdw_exec_requests. The request_id uniquely identifies each query and is the primary key for this DMV. The request_id is assigned sequentially for each new query and is prefixed with QID, which stands for query ID. Querying this DMV for a given session_id shows all queries for a given logon.

Step 1

The first step is to identify the query you want to investigate

```
SQL

-- Monitor active queries
SELECT *
FROM sys.dm_pdw_exec_requests
WHERE status not in ('Completed','Failed','Cancelled')
    AND session_id <> session_id()
ORDER BY submit_time DESC;

-- Find top 10 queries longest running queries
SELECT TOP 10 *
FROM sys.dm_pdw_exec_requests
ORDER BY total_elapsed_time DESC;
```

From the preceding query results, **note the Request ID** of the query that you would like to investigate.

Queries in the **Suspended** state can be queued due to a large number of active running queries. These queries also appear in the sys.dm_pdw_waits waits query with a type of UserConcurrencyResourceType. For information on concurrency limits, see Memory and concurrency limits or Resource classes for workload management. Queries can also wait for other reasons such as for object locks. If your query is waiting for a resource, see Investigating queries waiting for resources further down in this article.

To simplify the lookup of a query in the sys.dm_pdw_exec_requests table, use LABEL to assign a comment to your query, which can be looked up in the sys.dm_pdw_exec_requests view.

```
SQL

-- Query with Label
SELECT *
FROM sys.tables
OPTION (LABEL = 'My Query')
;
```

```
-- Find a query with the Label 'My Query'  
-- Use brackets when querying the label column, as it is a key word  
SELECT *  
FROM sys.dm_pdw_exec_requests  
WHERE [label] = 'My Query';
```

Step 2

Use the Request ID to retrieve the queries distributed SQL (DSQL) plan from sys.dm_pdw_request_steps

SQL

```
-- Find the distributed query plan steps for a specific query.  
-- Replace request_id with value from Step 1.
```

```
SELECT * FROM sys.dm_pdw_request_steps  
WHERE request_id = 'QID#####'  
ORDER BY step_index;
```

When a DSQL plan is taking longer than expected, the cause can be a complex plan with many DSQL steps or just one step taking a long time. If the plan is many steps with several move operations, consider optimizing your table distributions to reduce data movement.

The [Table distribution](#) article explains why data must be moved to solve a query. The article also explains some distribution strategies to minimize data movement.

To investigate further details about a single step, the operation_type column of the long-running query step and note the **Step Index**:

- Proceed with Step 3 for **SQL operations**: OnOperation, RemoteOperation, ReturnOperation.
- Proceed with Step 4 for **Data Movement operations**: ShuffleMoveOperation, BroadcastMoveOperation, TrimMoveOperation, PartitionMoveOperation, MoveOperation, CopyOperation.

Step 3

Use the Request ID and the Step Index to retrieve details from sys.dm_pdw_sql_requests, which contains execution information of the query step on all of the distributed databases.

SQL

```
-- Find the distribution run times for a SQL step.  
-- Replace request_id and step_index with values from Step 1 and 3.  
  
SELECT * FROM sys.dm_pdw_sql_requests  
WHERE request_id = 'QID#####' AND step_index = 2;
```

When the query step is running, DBCC PDW_SHOWEXECUTIONPLAN can be used to retrieve the SQL Server estimated plan from the SQL Server plan cache for the step running on a particular distribution.

SQL

```
-- Find the SQL Server execution plan for a query running on a specific SQL  
pool or control node.  
-- Replace distribution_id and spid with values from previous query.  
  
DBCC PDW_SHOWEXECUTIONPLAN(1, 78);
```

Step 4

Use the Request ID and the Step Index to retrieve information about a data movement step running on each distribution from sys.dm_pdw_dms_workers.

SQL

```
-- Find information about all the workers completing a Data Movement Step.  
-- Replace request_id and step_index with values from Step 1 and 3.  
  
SELECT * FROM sys.dm_pdw_dms_workers  
WHERE request_id = 'QID#####' AND step_index = 2;
```

- Check the total_elapsed_time column to see if a particular distribution is taking longer than others for data movement.
- For the long-running distribution, check the rows_processed column to see if the number of rows being moved from that distribution is larger than others. If so, this finding might indicate skew of your underlying data. One cause for data skew is distributing on a column with many NULL values (whose rows will all land in the same distribution). Prevent slow queries by avoiding distribution on these types of columns or filtering your query to eliminate NULLs when possible.

If the query is running, you can use DBCC PDW_SHOWEXECUTIONPLAN to retrieve the SQL Server estimated plan from the SQL Server plan cache for the currently running SQL Step within a particular distribution.

SQL

```
-- Find the SQL Server estimated plan for a query running on a specific SQL  
pool Compute or control node.  
-- Replace distribution_id and spid with values from previous query.
```

```
DBCC PDW_SHOWEXECUTIONPLAN(55, 238);
```

Dynamic Management Views (DMV) only contains 10,000 rows of data. On heavily utilized systems this means that data held in this table may be lost with hours, or even minutes as data is managed in a first in, first out system. As a result you can potentially lose meaningful information that can help you diagnose query performance issues on your system. In this situation, you should use the Query Store.

You can also monitor additional aspects of Azure Synapse SQL pools including:

- Monitoring waits
- Monitoring tempdb
- Monitoring memory
- Monitoring transaction log
- Monitoring PolyBase

You can view information about [monitoring these areas here](#)

Next unit: Knowledge check

[Continue >](#)

How are we doing?

✓ 200 XP



Knowledge check

3 minutes

1. Which ALTER DATABASE statement parameter allows a dedicated SQL pool to scale? *

 SCALE.

✗ Incorrect. SCALE is not a valid ALTER DATABASE parameter

 MODIFY

✓ Correct. MODIFY is used to scale a dedicated SQL pool.

 CHANGE.

2. Which workload management feature influences the order in which a request gets access to resources? *

 Workload classification.

✗ Incorrect. Workload classification allows workload policies to be applied to requests through assigning resource classes and importance..

 Workload importance.

✓ Correct. Workload importance influences the order in which a request gets access to resources. On a busy system, a request with higher importance has first access to resources.

 Workload isolation.

3. Which Dynamic Management View enables the view of the active connections against a dedicated SQL pool? *

 sys.dm_pdw_exec_requests.

✓ Correct. sys.dm_pdw_exec_requests enables you to view the active connections against a dedicated SQL pool

 sys.dm_pdw_dms_workers.



DBCC PDW_SHOWEXECUTIONPLAN.

- Incorrect. DBCC PDW_SHOWEXECUTIONPLAN Shows an execution plan of a query, and is not a Dynamic Management View.

Next unit: Summary

[Continue >](#)

How are we doing?

Introduction

3 minutes

In this module, you will learn how to approach and implement security to protect your data with Azure Synapse Analytics.

In this module, you will:

- Understand network security options for Azure Synapse Analytics
- Configure Conditional Access
- Configure Authentication
- Manage authorization through column and row level security
- Manage sensitive data with Dynamic Data masking
- Implement encryption in Azure Synapse Analytics
- Understand advanced data security options for Azure Synapse Analytics

Prerequisites

Before taking this module, it is recommended that the student is able to:

- Log into the Azure portal
- Create a Synapse Analytics Workspace
- Create an Azure Synapse Analytics SQL Pool

Next unit: Understand network security options for Azure Synapse Analytics

[Continue >](#)

How are we doing?     

100 XP

Understand network security options for Azure Synapse Analytics

5 minutes

There are a range of network security steps that you should consider to secure Azure Synapse Analytics. One of the first aspects that you will consider is securing access to the service itself. This can be achieved by creating the following network objects including:

- Firewall rules
- Virtual networks
- Private endpoints

Firewall rules

Firewall rules enable you to define the type of traffic that is allowed or denied access to an Azure Synapse workspace using the originating IP address of the client that is trying to access the Azure Synapse Workspace. IP firewall rules configured at the workspace level apply to all public endpoints of the workspace including dedicated SQL pools, serverless SQL pool, and the development endpoint.

You can choose to allow connections from all IP addresses as you are creating the Azure Synapse Workspaces, although this is not recommended as it does not allow for control access to the workspace. Instead, within the Azure portal, you can configure specific IP address ranges and associate them with a rule name so that you have greater control.

The screenshot shows the Azure portal interface for managing a Synapse workspace named 'asaworkspacetoto'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Settings (selected), Analytics pools, Security (selected), Firewalls (highlighted with a red box), Managed identities, Private endpoint connections ..., Azure SQL Auditing, Azure Defender for SQL, Monitoring, Alerts, Metrics, Automation, and Tasks (preview). The main content area displays the 'Firewalls' settings. A status message at the top says, 'The IPs listed below will have full access to Synapse workspace 'asaworkspacetoto''. Below it is a toggle switch labeled 'Allow Azure services and resources to access this workspace' with options 'ON' and 'OFF' (set to OFF). Underneath is a table with columns 'Rule name', 'Start IP', and 'End IP', each containing an empty input field.

Make sure that the firewall on your network and local computer allows outgoing communication on TCP ports 80, 443 and 1443 for Synapse Studio.

Also, you need to allow outgoing communication on UDP port 53 for Synapse Studio. To connect using tools such as SSMS and Power BI, you must allow outgoing communication on TCP port 1433.

Virtual networks

Azure Virtual Network (VNet) enables private networks in Azure. VNet enables many types of Azure resources, such as Azure Synapse Analytics, to securely communicate with other virtual networks, the internet, and on-premises networks. When you create your Azure Synapse workspace, you can choose to associate it to a Microsoft Azure Virtual Network. The Virtual Network associated with your workspace is managed by Azure Synapse. This Virtual Network is called a Managed workspace Virtual Network.

Using a managed workspace virtual network provides the following benefits:

- With a Managed workspace Virtual Network, you can offload the burden of managing the Virtual Network to Azure Synapse.
- You don't have to configure inbound NSG rules on your own Virtual Networks to allow Azure Synapse management traffic to enter your Virtual Network. Misconfiguration of these NSG rules causes service disruption for customers.
- You don't need to create a subnet for your Spark clusters based on peak load.
- Managed workspace Virtual Network along with Managed private endpoints protects against data exfiltration. You can only create Managed private endpoints in a workspace that has a Managed workspace Virtual Network associated with it.
- it ensures that your workspace is network isolated from other workspaces.

If your workspace has a Managed workspace Virtual Network, Data integration and Spark resources are deployed in it. A Managed workspace Virtual Network also provides user-level isolation for Spark activities because each Spark cluster is in its own subnet.

Dedicated SQL pool and serverless SQL pool are multi-tenant capabilities and therefore reside outside of the Managed workspace Virtual Network. Intra-workspace communication to dedicated SQL pool and serverless SQL pool use Azure private links. These private links are automatically created for you when you create a workspace with a Managed workspace Virtual Network associated to it.

You can only choose to enable managed virtual networks as you are creating the Azure Synapse Workspaces.

Create Synapse workspace



*Basics *Security **Networking** Tags Summary

Configure networking settings for your workspace.

Allow connections from all IP addresses

Azure Synapse Studio and other client tools will only be able to connect to the workspace endpoints if this setting is allowed. Connections from specific IP addresses or all Azure services can be allowed/disallowed after the workspace is provisioned.

Allow connections from all IP addresses to your workspace's endpoints. You can restrict this to just Azure datacenter IP addresses and/or specific IP address ranges after creating the workspace.

Allow connections from all IP addresses

Managed virtual network

Choose whether you want a Synapse-managed virtual network dedicated for your Azure Synapse workspace. [Learn more](#) ⓘ

Enable managed virtual network ⓘ

Private endpoints

Azure Synapse Analytics enables you to connect up its various components through endpoints. You can set up managed private endpoints to access these components in a secure manner known as private links. This can only be achieved in an Azure Synapse workspace with a Managed workspace Virtual Network. Private link enables you to access Azure services (such as Azure Storage and Azure Cosmos DB) and Azure hosted customer/partner services from your Azure Virtual Network securely.

When you use a private link, traffic between your Virtual Network and workspace traverses entirely over the Microsoft backbone network. Private Link protects against data exfiltration risks. You establish a private link to a resource by creating a private endpoint.

Private endpoint uses a private IP address from your Virtual Network to effectively bring the service into your Virtual Network. Private endpoints are mapped to a specific resource in Azure and not the entire service. Customers can limit connectivity to a specific resource approved by their organization. You can manage the private endpoints in the Azure Synapse Studio manage hub.

Microsoft Azure | Synapse Analytics > test

» Publish all ✓ Validate all Discard all

Analytics pools

- SQL pools
- Apache Spark pools

External connections

- Linked services

Integration

- Triggers
- Integration runtimes

Security

- Access control
- Credentials

Managed private endpoints

Managed private endpoints

Managed private endpoint uses a private IP address from within Managed Virtual Network to connect to an Azure resource or your own private link service. Connected managed private endpoints listed below provide access to Azure resources or private link services. [Learn more](#)

New **Refresh**

Showing 1 - 2 of 2 items

Name	Provisioning status	Approval status	VNet name	Possible Linked Services	Linked resources
synapse-ws-sql-test	Succeeded	Approved	default	1	/subscriptions/...
synapse-ws-sqlOnDemand...	Succeeded	Approved	default	0	/subscriptions/...

Next unit: Configure Conditional Access

Continue >

How are we doing? ★ ★ ★ ★ ★

100 XP



Configure Conditional Access

7 minutes

Conditional Access is a feature that enables you to define the conditions under which a user can connect to your Azure subscription and access services. Conditional Access provides an additional layer of security that can be used in combination with authentication to strengthen the security access to your network.

Conditional Access policies at their simplest are if-then statements, if a user wants to access a resource, then they must complete an action. As an example, if a Data Engineer wishes to access services in Azure Synapse Analytics, they may be requested by the Conditional Access policy to perform an additional step of multifactor authentication (MFA) to complete the authentication to get onto the service

Conditional Access policies use signals as a basis to determine if Conditional Access should first be applied. Common signals include:

- User or group membership names
- IP address information
- Device platforms or type
- Application access requests
- Real-time and calculated risk detection
- Microsoft Cloud App Security (MCAS)

Based on these signals, you can then choose to block access. The alternative is you can grant access, and at the same time request that the user perform an additional action including:

- Perform multifactor authentication
- Use a specific device to connect

Given the amount of data that could potentially be stored, Azure Synapse Analytics dedicated SQL pools supports Conditional Access to provide protection for your data. It does require that Azure Synapse Analytics is configured to support Azure Active Directory, and that if you chose multifactor authentication, that the tool you are using support it.

To configure Conditional Access, you can perform the following steps:

1. Sign in to the Azure portal, select **Azure Active Directory**, and then select **Conditional Access**.

Microsoft Azure azatest

azatest Azure Active Directory

Classic portal Switch directory Delete directory

Users and groups

Enterprise applications

USERS SIGN-INS

80
60
40
20
0

May 7 May 14 May 21 May 28

Recommended

Sync with Windows Server AD

Sync users and groups from your on-premises directory to your Azure AD

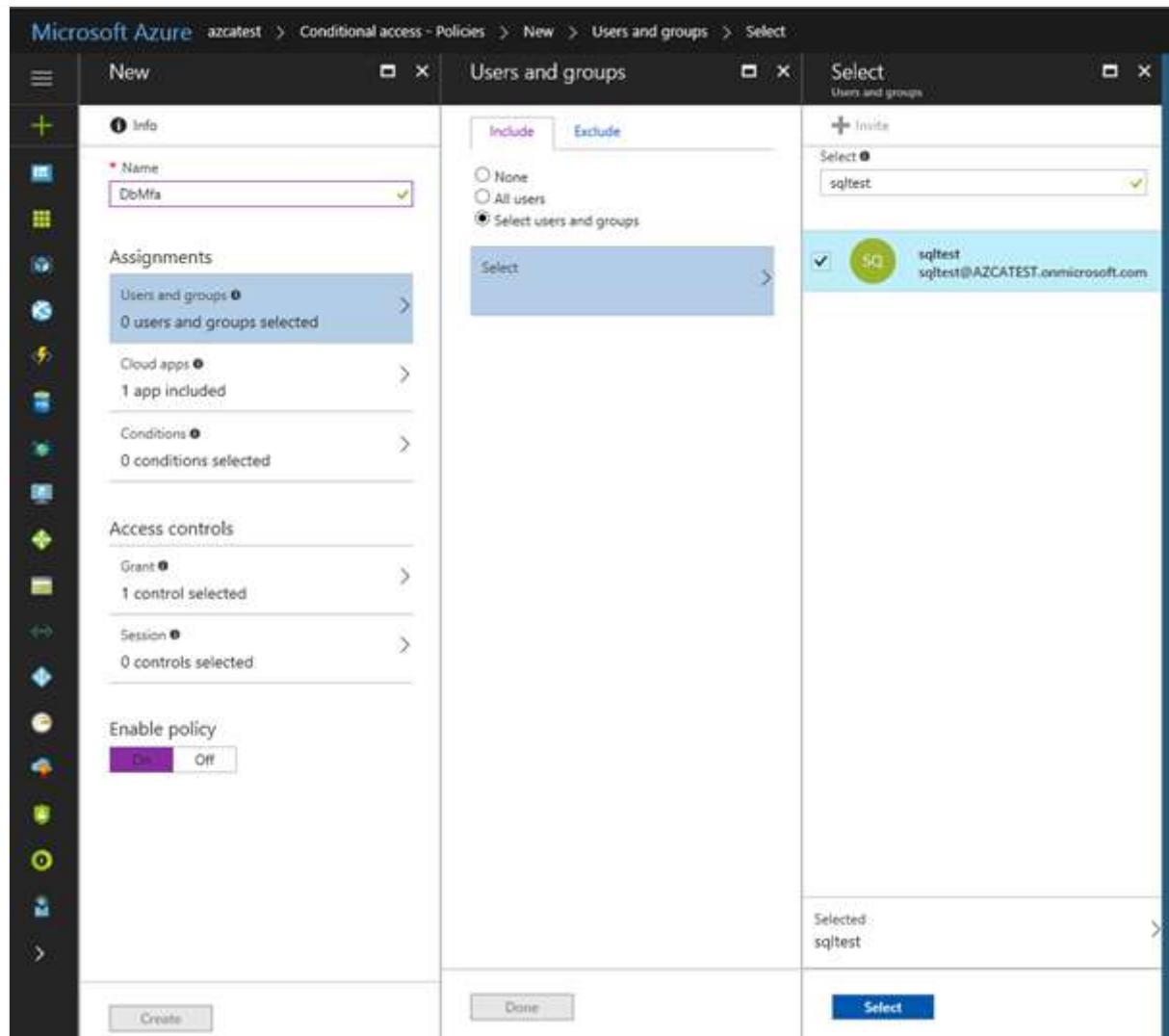
Self-service password reset

Enable your users to reset their forgotten passwords

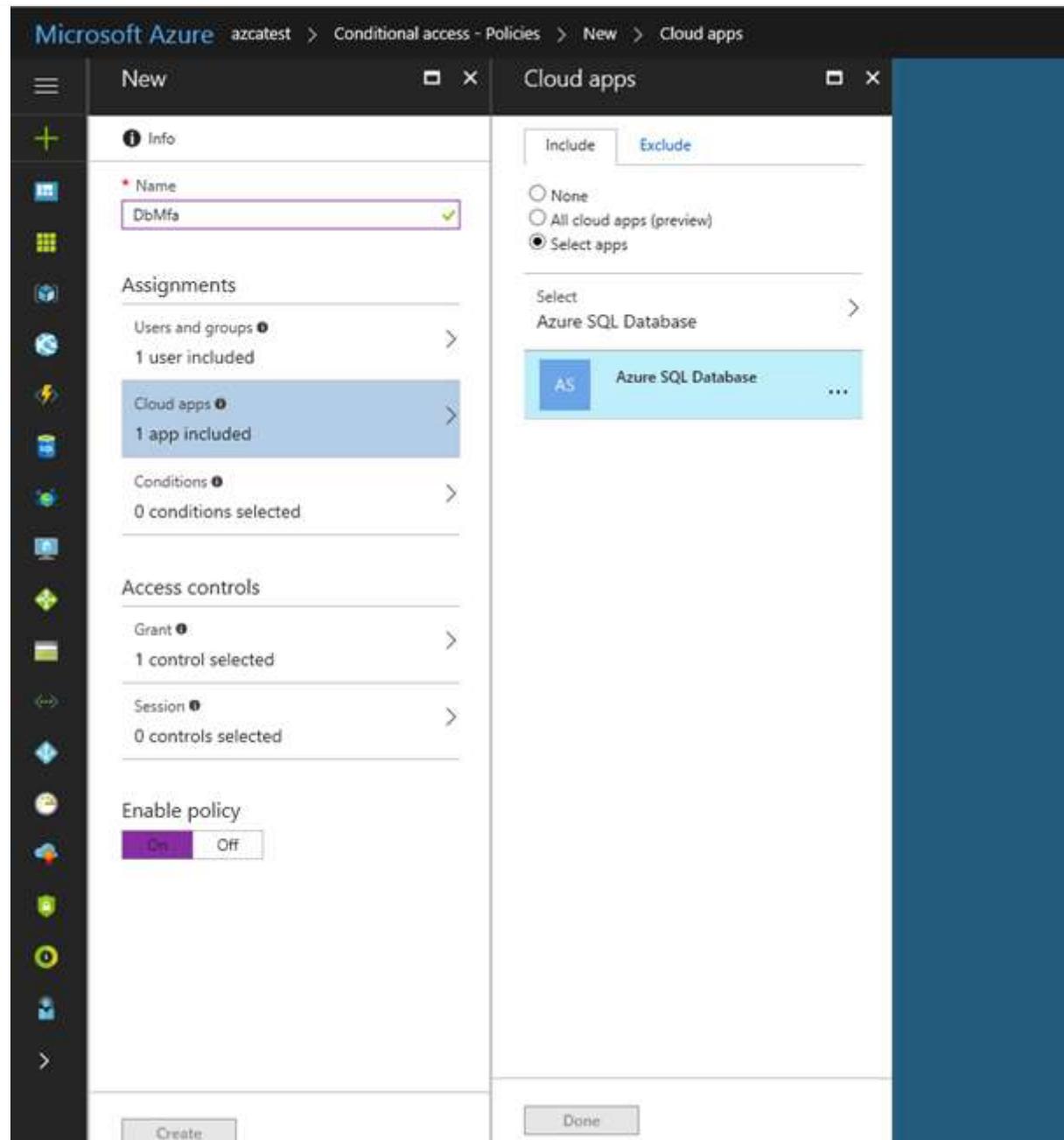
App registrations

2

2. In the **Conditional Access-Policies** blade, click **New policy**, provide a name, and then click **Configure rules**.
3. Under **Assignments**, select **Users and groups**, check **Select users and groups**, and then select the user or group for Conditional Access. Click **Select**, and then click **Done** to accept your selection.



4. Select **Cloud apps**, click **Select apps**. You see all apps available for Conditional Access. Select **Azure SQL Database**, at the bottom click **Select**, and then click **Done**.



5. If you can't find **Azure SQL Database** listed in the following third screenshot, complete the following steps:

- Connect to your database in Azure SQL Database by using SSMS with an Azure AD admin account.
- Execute `CREATE USER [user@yourtenant.com] FROM EXTERNAL PROVIDER.`
- Sign into Azure AD and verify that Azure SQL Database, SQL Managed Instance, or Azure Synapse are listed in the applications in your Azure AD instance.

6. Select **Access controls**, select **Grant**, and then check the policy you want to apply. For this example, we select **Require multifactor authentication**.

The screenshot shows the Microsoft Azure Conditional Access - Policies interface. On the left, a sidebar lists various policy types with corresponding icons. The main area is split into two tabs: 'New' (left) and 'Grant' (right).

New Tab (Left):

- Info:** A form with a required field 'Name' containing 'DbMfa'. Below it is a green checkmark icon.
- Assignments:** A section with three items:
 - Users and groups:** 1 user included.
 - Cloud apps:** 1 app included.
 - Conditions:** 0 conditions selected.
- Access controls:** A section with two items:
 - Grant:** 1 control selected. This item is highlighted with a blue background.
 - Session:** 0 controls selected.
- Enable policy:** A switch button set to 'On'.

Grant Tab (Right):

Select the controls to be enforced.

Block access
 Grant access

Require multi-factor authentication ⓘ

Require device to be marked as compliant ⓘ

Require domain joined device ⓘ

For multiple controls

Require all the selected controls
 Require one of the selected controls (preview)

Buttons at the bottom:

- Create
- Select

Next unit: Configure authentication

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Configure authentication

8 minutes

Authentication is the process of validating credentials as you access resources in a digital infrastructure. This ensures that you can validate that an individual, or a service that wants to access a service in your environment can prove who they are. Azure Synapse Analytics provides several different methods for authentication.

What needs to be authenticated

There are a variety of scenarios that means that authentication must take place to protect the data that is stored in your Azure Synapse Analytics estate.

The common form of authentication is that of individuals who want to access the data in the service. This is typically seen as an individual providing a username and password to authenticate against a service. However, this is also becoming more sophisticated with authentication requests working in combination with Conditional Access policies to further secure the authentication process with additional security steps.

What is less obvious is the fact that services must authenticate with other services so that they can operate seamlessly. An example of this is using an Azure Synapse Spark or serverless SQL pool to access data in an Azure Data Lake store. An authentication mechanism must take place in the background to ensure that Azure Synapse Analytics can access the data in the data lake in an authenticated manner.

Finally, there are situations where users and services operate together at the same time. Here you have a combination of both user and service authentication taking place under the hood to ensure that the user is getting access to the data seamlessly. An example of this is using Power BI to view reports in a dashboard that is being serviced by a dedicated SQL pool. Here you have multiple levels of authentication taking place that needs to be managed.

Types of security

The following are the types of authentication that you should be aware of when working with Azure Synapse Analytics.

Azure Active Directory

Azure Active Directory is a directory service that allows you to centrally maintain objects that can be secured. The objects can include user accounts and computer accounts. An employee of an organization will typically have a user account that represents them in the organizations Azure Active Directory tenant, and they then use the user account with a password to authenticate against other resources that are stored within the directory using a process known as single sign-on.

The power of Azure Active Directory is that they only have to log in once, and Azure Active Directory will manage access to other resources based on the information held within it using pass through authentication. If a user and an instance of Azure Synapse Analytics are part of the same Azure Active Directory, it is possible for the user to access Azure Synapse Analytics without an apparent login. If managed correctly, this process is seamless as the administrator would have given the user authorization to access Azure Synapse Analytics dedicated SQL pool as an example.

In this situation, it is normal for an Azure Administrator to create the user accounts and assign them to the appropriate roles and groups in Azure Active Directory. The Data Engineer will then add the user, or a group to which the user belongs to access a dedicated SQL pool.

Managed identities

Managed identity for Azure resources is a feature of Azure Active Directory. The feature provides Azure services with an automatically managed identity in Azure AD. You can use the Managed Identity capability to authenticate to any service that supports Azure Active Directory authentication.

Managed identities for Azure resources are the new name for the service formerly known as Managed Service Identity (MSI). A system-assigned managed identity is created for your Azure Synapse workspace when you create the workspace.

Azure Synapse also uses the managed identity to integrate pipelines. The managed identity lifecycle is directly tied to the Azure Synapse workspace. If you delete the Azure Synapse workspace, then the managed identity is also cleaned up.

The workspace managed identity needs permissions to perform operations in the pipelines. You can use the object ID or your Azure Synapse workspace name to find the managed identity when granting permissions.

You can retrieve the managed identity in the Azure portal. Open your Azure Synapse workspace in Azure portal and select **Overview** from the left navigation. The managed identity's object ID is displayed to in the main screen.

The screenshot shows the 'Overview' tab of a Synapse workspace named 'asaworkspacetoto'. The 'Managed identity object ID' field is highlighted with a red box. Other visible fields include 'Resource group (change)', 'Status', 'Location', 'Subscription (change)', 'Managed virtual network', 'Managed Identity object ...', 'Workspace web URL', and 'Tags (change)'. The 'Getting started' section contains links to 'Open Synapse Studio' and 'Read documentation'.

The managed identity information will also show up when you create a linked service that supports managed identity authentication from Azure Synapse Studio.

Launch **Azure Synapse Studio** and select the **Manage** tab from the left navigation. Then select **Linked services** and choose the **+ New** option to create a new linked service.

In the **New linked service** window, type **Azure Data Lake Storage Gen2**. Select the **Azure Data Lake Storage Gen2** resource type from the list below and choose **Continue**.

In the next window, choose **Managed Identity** for **Authentication method**. You'll see the managed identity's **Name** and **Object ID**.

New linked service (Azure Data Lake Storage Gen2)

i Choose a name for your linked service. This name cannot be updated later.

Name *

AzureDataLakeStorage1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime



Authentication method

Managed Identity



Account selection method

From Azure subscription Enter manually



Azure subscription

Select all



Storage account name *



Managed identity name:

Managed identity object ID:

Grant workspace service managed identity access to your Azure Data Lake Storage Gen2.

[Learn more](#)

Test connection



To linked service To file path

Annotations

New

Name

Advanced

Create

Back

Test connection

Cancel

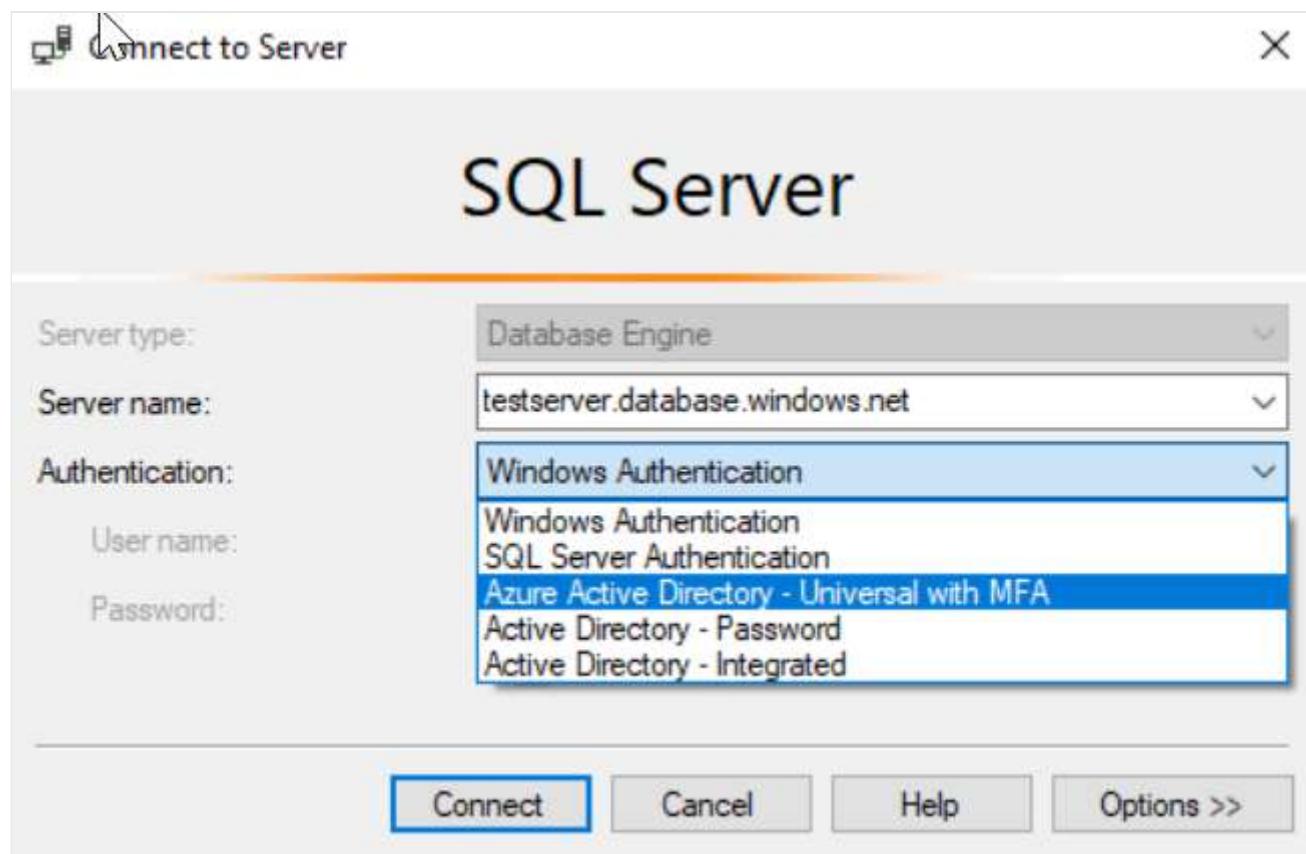
SQL Authentication

For user accounts that are not part of an Azure Active directory, then using SQL Authentication will be an alternative. In this instance, a user is created in the instance of a dedicated SQL pool. If the user in question requires administrator access, then the details of the user are held in the master database. If administrator access is not required, you can create a user in a specific database. A user then connects directly to the Azure Synapse Analytics dedicated SQL pool where they are prompted to use a username and password to access the service.

This approach is typically useful for external users who need to access the data, or if you are using third party or legacy applications against the Azure Synapse Analytics dedicated SQL pool

Multifactor authentication

Synapse SQL support connections from SQL Server Management Studio (SSMS) using Active Directory Universal Authentication.



This enables you to operate in environments that use Conditional Access policies that enforce multifactor authentication as part of the policy.

Keys

If you are unable to use a managed identity to access resources such as Azure Data Lake then you can use storage account keys and shared access signatures.

With a storage account key. Azure creates two of these keys (primary and secondary) for each storage account you create. The keys give access to everything in the account. You'll find the storage account keys in the Azure portal view of the storage account. Just select **Settings**, and then click **Access keys**.

As a best practice, you shouldn't share storage account keys, and you can use Azure Key Vault to manage and secure the keys.

Azure Key Vault is a secret store: a centralized cloud service for storing app secrets - configuration values like passwords and connection strings that must remain secure at all times. Key Vault helps you control your apps' secrets by keeping them in a single central location and providing secure access, permissions control, and access logging.

The main benefits of using Key Vault are:

- Separation of sensitive app information from other configuration and code, reducing risk of accidental leaks
- Restricted secret access with access policies tailored to the apps and individuals that need them
- Centralized secret storage, allowing required changes to happen in only one place
- Access logging and monitoring to help you understand how and when secrets are accessed

Secrets are stored in individual vaults, which are Azure resources used to group secrets together. Secret access and vault management is accomplished via a REST API, which is also supported by all of the Azure management tools as well as client libraries available for many popular languages. Every vault has a unique URL where its API is hosted.

Shared access signatures

If an external third-party application needs access to your data, you'll need to secure their connections without using storage account keys. For untrusted clients, use a shared access signature (SAS). A shared access signature is a string that contains a security token that can be attached to a URI. Use a shared access signature to delegate access to storage objects and specify constraints, such as the permissions and the time range of access. You can give a customer a shared access signature token.

Types of shared access signatures

You can use a service-level shared access signature to allow access to specific resources in a storage account. You'd use this type of shared access signature, for example, to allow an app to retrieve a list of files in a file system or to download a file.

Use an account-level shared access signature to allow access to anything that a service-level shared access signature can allow, plus additional resources and abilities. For example, you can use an account-level shared access signature to allow the ability to create file systems.

Next unit: Manage authorization through column and row level security

[Continue >](#)

How are we doing?

✓ 100 XP



Manage authorization through column and row level security

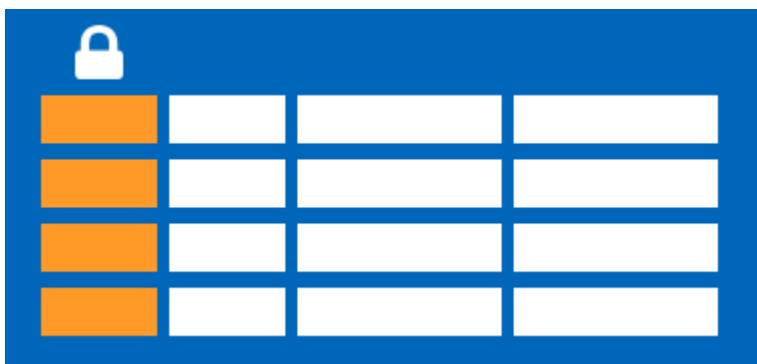
6 minutes

In this topic, we'll go through how you can manage authorization through column and row level security within Azure Synapse Analytics. We'll start off by talking about column level security in Azure Synapse Analytics, and finish with row level security.

Column level security in Azure Synapse Analytics

Generally speaking, column level security is simplifying a design and coding for the security in your application. It allows you to restrict column access in order to protect sensitive data. For example, if you want to ensure that a specific user 'Leo' can only access certain columns of a table because he's in a specific department. The logic for 'Leo' only to access the columns specified for the department he works in, is a logic that is located in the database tier, rather than on the application level data tier. If he needs to access data from any tier, the database should apply the access restriction every time he tries to access data from another tier. The reason for doing so is to make sure that your security is reliable and robust since we're reducing the surface area of the overall security system. Column level security will also eliminate the necessity for the introduction of view, where you would filter out columns, to impose access restrictions on 'Leo'

The way to implement column level security is by using the GRANT T-SQL statement. Using this statement, SQL and Azure Active Directory (AAD) support the authentication.



Syntax

The syntax to use for implementing column level security looks as follows:

```
syntaxsql
```

```
GRANT <permission> [ ,...n ] ON
    [ OBJECT :: ][ schema_name ]. object_name [ ( column [ ,...n ] ) ] // specifying the column access
        TO <database_principal> [ ,...n ]
        [ WITH GRANT OPTION ]
        [ AS <database_principal> ]
<permission> ::=

    SELECT
    | UPDATE
<database_principal> ::=
    Database_user // specifying the database user
    | Database_role // specifying the database role
    | Database_user_mapped_to_Windows_User
    | Database_user_mapped_to_Windows_Group
```

So when would you use column-level security? Let's say that you are a financial services firm, and can only have an account manager allowed to have access to a customer's social security number, phone number, or other personally identifiable information. It is imperative to distinguish the role of an account manager versus the manager of the account managers.

Another use case might be related to the Healthcare Industry. Let's say you have a specific health care provider. This healthcare provider only wants doctors and nurses to be able to access medical records. The billing department should not have access to view this data. Column-level security might be the option to use.

So how does column level security distinguishes from row-level security? Let's look into that.

Row level security in Azure Synapse Analytics

Row-level security (RLS) can help you to create a group membership or execution context in order to control not just columns in a database table, but actually, the rows. RLS, just like column-level security, can simply help and enable your design and coding of your application security. However, compared to column-level security where it's focused on the columns (parameters), RLS helps you implement restrictions on data row access. Let's say that your employee can only access rows of data that are important to the department, you should implement RLS. If you want to restrict, for example, customer data access that is only relevant to the company, you can implement RLS. The restriction on the access of the rows is a logic that is located in the database tier, rather than on the application level data tier. If 'Leo' needs to access data from any tier, the database should apply the access restriction every time he tries to access data from another tier. The reason for doing so is to make sure that your security is reliable and robust since we're reducing the surface area of the overall security system.

The way to implement RLS is by using the CREATE SECURITY POLICY[!INCLUDEtsql] statement. The predicates are created as inline table-valued functions. It is imperative to understand that within Azure Synapse, only supports filter predicates. If you need to use a block predicate, you won't be able to find support at this moment within Azure synapse.



Description of row level security in relation to filter predicates

RLS within Azure Synapse supports one type of security predicates, which are Filter predicates, not block predicates.

What filter predicates do, is silently filtering the rows that are available for reading operations such as SELECT, UPDATE, DELETE.

The access to row-level data in a table is restricted as an inline table-valued function, which is a security predicate. This table-valued function will then be invoked and enforced by the security policy that you need. An application is not aware of rows that are filtered from the result set for filter predicates. So what will happen is that if all rows are filtered, a null set is returned.

When you are using filter predicates, it will be applied when data is read from the base table. The filter predicate affects all get operations such as SELECT, DELETE, UPDATE. You are unable to select or delete rows that have been filtered. It is not possible for you to update a row that has been filtered. What you can do, is update rows in a way that they will be filtered afterward.

Use cases

We've already mentioned some use cases for RLS. Another use case might where you have created a multi-tenant application where you create a policy where logical separations of a tenant's data rows from another tenant's data rows are enforced. In order to implement this efficiently, it is highly recommended to store data for many tenants in a single table.

When we look at RLS filter predicates, they are functionally equivalent to appending a **WHERE** clause. The predicate can be as sophisticated as business practices dictate, or the clause can be as simple as `WHERE TenantId = 42`.

When we look at RLS more formally, RLS introduces predicate based access control. The reason why RLS can be used for predicate access control is that it is a flexible, centralized, predicate-

based evaluation. The filter predicate can be based on metadata or any other criteria you would determine as appropriate. The predicate is used as a criterion to determine if the user has the appropriate access to the data based on user attributes. Label-based access control can be implemented by using predicate-based access control.

Permissions

If you want to create, alter or drop the security policies, you would have to use the **ALTER ANY SECURITY POLICY** permission. The reason for that is when you are creating or dropping a security policy it requires **ALTER** permissions on the schema.

In addition to that, there are other permissions required for each predicate that you would add:

- **SELECT** and **REFERENCES** permissions on the inline table-valued function being used as a predicate.
- **REFERENCES** permission on the table that you target to be bound to the policy.
- **REFERENCES** permission on every column from the target table used as arguments.

Once you've set up the security policies, they will apply to all the users (including dbo users in the database) Even though DBO users can alter or drop security policies, their changes to the security policies can be audited. If you have special circumstances where highly privileged users, like a sysadmin or db_owner, need to see all rows to troubleshoot or validate data, you would still have to write the security policy in order to allow that.

If you have created a security policy where **SCHEMABINDING = OFF**, in order to query the target table, the user must have the **SELECT** or **EXECUTE** permission on the predicate function. They also need permissions to any additional tables, views, or functions used within the predicate function. If a security policy is created with **SCHEMABINDING = ON** (the default), then these permission checks are bypassed when users query the target table.

Best practices

There are some best practices to take in mind when you want to implement RLS. We recommended creating a separate schema for the RLS objects. RLS objects in this context would be the predicate functions, and security policies. Why is that a best practice? It helps to separate the permissions that are required on these special objects from the target tables. In addition to that, separation for different policies and predicate functions may be needed in multi-tenant-databases. However, it is not a standard for every case.

Another best practice to bear in mind is that the **ALTER ANY SECURITY POLICY** permission should only be intended for highly privileged users (such as a security policy manager). The security policy manager should not require **SELECT** permission on the tables they protect.

In order to avoid potential runtime errors, you should take into mind type conversions in predicate functions that you write. Also, you should try to avoid recursion in predicate functions. The reason for this is to avoid performance degradation. Even though the query optimizer will try to detect the direct recursions, there is no guarantee to find the indirect recursions. With indirect recursion, we mean where a second function calls the predicate function.

It would also be recommended to avoid the use of excessive table joins in predicate functions. This would maximize performance.

Generally speaking when it comes to the logic of predicates, you should try to avoid logic that depends on session-specific SET options. Even though this is highly unlikely to be used in practical applications, predicate functions whose logic depends on certain session-specific **SET** options can leak information if users are able to execute arbitrary queries. For example, a predicate function that implicitly converts a string to **datetime** could filter different rows based on the **SET DATEFORMAT** option for the current session.

Next unit: Exercise - Manage authorization through column and row level security

[Continue >](#)

How are we doing?     

✓ 100 XP



Exercise - Manage authorization through column and row level security

9 minutes

In this exercise, examples are shown how you can manage authorization through column and row level security.

An example of column level security

The following example shows how to restrict TestUser from accessing the SSN column of the Membership table:

Create Membership table with SSN column used to store social security numbers:

SQL

```
CREATE TABLE Membership
(MemberID int IDENTITY,
 FirstName varchar(100) NULL,
 SSN char(9) NOT NULL,
 LastName varchar(100) NOT NULL,
 Phone varchar(12) NULL,
 Email varchar(100) NULL);
```

Allow TestUser to access all columns except for the SSN column, which has the sensitive data:

SQL

```
GRANT SELECT ON Membership(MemberID, FirstName, LastName, Phone, Email) TO
TestUser;
```

Queries executed as TestUser will fail if they include the SSN column:

SQL

```
SELECT * FROM Membership;

-- Msg 230, Level 14, State 1, Line 12
-- The SELECT permission was denied on the column 'SSN' of the object 'Membership', database 'CLS_TestDW', schema 'dbo'.
```

An example of row level security

This scenario gives you an example for row level security on an Azure Synapse external table.

This short example creates three users and an external table with six rows. It then creates an inline table-valued function and a security policy for the external table. The example shows how select statements are filtered for the various users.

Prerequisites

- You must have a SQL pool. See [Create a Synapse SQL pool](#)
- The server hosting your SQL pool must be registered with AAD and you must have an Azure storage account with Storage Blob Contributor permissions. Follow the steps [here](#).
- Create a file system for your Azure Storage account. Use Storage Explorer to view your storage account. Right click on containers and select *Create file system*.

Once you have the prerequisites in place, create three user accounts that will demonstrate different access capabilities.

SQL

```
--run in master
CREATE LOGIN Manager WITH PASSWORD = '<user_password>'
GO
CREATE LOGIN Sales1 WITH PASSWORD = '<user_password>'
GO
CREATE LOGIN Sales2 WITH PASSWORD = '<user_password>'
GO

--run in master and your SQL pool database
CREATE USER Manager FOR LOGIN Manager;
CREATE USER Sales1 FOR LOGIN Sales1;
CREATE USER Sales2 FOR LOGIN Sales2 ;
```

Create a table to hold data.

SQL

```
CREATE TABLE Sales
(
    OrderID int,
    SalesRep sysname,
    Product varchar(10),
    Qty int
);
```

Populate the table with six rows of data, showing three orders for each sales representative.

SQL

```
INSERT INTO Sales VALUES (1, 'Sales1', 'Valve', 5);
INSERT INTO Sales VALUES (2, 'Sales1', 'Wheel', 2);
INSERT INTO Sales VALUES (3, 'Sales1', 'Valve', 4);
INSERT INTO Sales VALUES (4, 'Sales2', 'Bracket', 2);
INSERT INTO Sales VALUES (5, 'Sales2', 'Wheel', 5);
INSERT INTO Sales VALUES (6, 'Sales2', 'Seat', 5);
-- View the 6 rows in the table
SELECT * FROM Sales;
```

Create an Azure Synapse external table from the Sales table you just created.

SQL

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<user_password>';

CREATE DATABASE SCOPED CREDENTIAL msi_cred WITH IDENTITY = 'Managed Service
Identity';

CREATE EXTERNAL DATA SOURCE ext_datasource_with_abfss WITH (TYPE = hadoop,
LOCATION =
'abfss://<file_system_name@storage_account>.dfs.core.windows.net',
CREDENTIAL = msi_cred);

CREATE EXTERNAL FILE FORMAT MSIFormat WITH (FORMAT_TYPE=DELIMITEDTEXT);

CREATE EXTERNAL TABLE Sales_ext WITH (LOCATION='<your_table_name>',
DATA_SOURCE=ext_datasource_with_abfss, FILE_FORMAT=MSIFormat,
REJECT_TYPE=Percentage, REJECT_SAMPLE_VALUE=100, REJECT_VALUE=100)
AS SELECT * FROM sales;
```

Grant SELECT for the three users on the external table Sales_ext that you created.

SQL

```
GRANT SELECT ON Sales_ext TO Sales1;
GRANT SELECT ON Sales_ext TO Sales2;
GRANT SELECT ON Sales_ext TO Manager;
```

Create a new schema, and an inline table-valued function, you may have completed this in example A. The function returns 1 when a row in the SalesRep column is the same as the user executing the query (@SalesRep = USER_NAME()) or if the user executing the query is the Manager user (USER_NAME() = 'Manager').

SQL

```
CREATE SCHEMA Security;
GO

CREATE FUNCTION Security.fn_securitypredicate(@SalesRep AS sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS fn_securitypredicate_result
WHERE @SalesRep = USER_NAME() OR USER_NAME() = 'Manager';
```

Create a security policy on your external table using the inline table-valued function as a filter predicate. The state must be set to ON to enable the policy.

SQL

```
CREATE SECURITY POLICY SalesFilter_ext
ADD FILTER PREDICATE Security.fn_securitypredicate(SalesRep)
ON dbo.Sales_ext
WITH (STATE = ON);
```

Now test the filtering predicate, by selecting from the Sales_ext external table. Sign in as each user, Sales1, Sales2, and manager. Run the following command as each user.

SQL

```
SELECT * FROM Sales_ext;
```

The Manager should see all six rows. The Sales1 and Sales2 users should only see their sales.

Alter the security policy to disable the policy.

SQL

```
ALTER SECURITY POLICY SalesFilter_ext
WITH (STATE = OFF);
```

Now the Sales1 and Sales2 users can see all six rows.

Connect to the Azure Synapse database to clean up resources

SQL

```
DROP USER Sales1;
DROP USER Sales2;
DROP USER Manager;
```

```
DROP SECURITY POLICY SalesFilter_ext;
DROP TABLE Sales;
DROP EXTERNAL TABLE Sales_ext;
DROP EXTERNAL DATA SOURCE ext_datasource_with_abfss ;
DROP EXTERNAL FILE FORMAT MSIFormat;
DROP DATABASE SCOPED CREDENTIAL msi_cred;
DROP MASTER KEY;
```

Connect to logical master to clean up resources.

SQL

```
DROP LOGIN Sales1;
DROP LOGIN Sales2;
DROP LOGIN Manager;
```

Next unit: Manage sensitive data with Dynamic Data Masking

[Continue >](#)

How are we doing?

100 XP



Manage sensitive data with Dynamic Data Masking

8 minutes

Azure SQL Database, Azure SQL Managed Instance, and Azure Synapse Analytics support Dynamic Data Masking. Dynamic Data Masking ensures limited data exposure to nonprivileged users, such that they can't see the data that is being masked. It also helps you in preventing unauthorized access to sensitive information that has minimal impact on the application layer. Dynamic Data Masking is a policy-based security feature. It will hide the sensitive data in a result set of a query that runs over designated database fields.

Let's give you an example how it works. Let's say you work at a bank as a service representative in a call center. Due to compliance, any caller must identify themselves by providing several digits of their credit card number. In this scenario, the full credit card number shouldn't be fully exposed to the service representative in the call center. You can define a masking rule that masks all but the last four digits of a credit card number so that you would get a query that only gives as a result the last four digits of the credit card number. This is just one example that could be equally applied to a variety of personal data such that compliance isn't violated. For Azure Synapse Analytics, the way to set up a Dynamic Data Masking policy is using PowerShell or the REST API. The configuration of the Dynamic Data Masking policy can be done by the Azure SQL Database admin, server admin, or SQL Security Manager roles.

In Azure Synapse Analytics, you can find Dynamic Data Masking here;

The screenshot shows the Azure portal interface for managing a Dedicated SQL pool named 'SQLPool01'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Settings (Workload management, Maintenance schedule, Geo-backup policy, Connection strings, Properties, Locks), Security (Auditing, Data Discovery & Classification, Dynamic Data Masking, Security Center, Transparent data encryption). The 'Dynamic Data Masking' link is highlighted with a red box. The main content area displays 'Masking rules' with a table showing columns: Mask name and Mask Function. A message states 'You haven't created any masking rules.' Below this is a section for 'SQL users excluded from masking (administrators are always excluded)' with a dropdown menu set to 'SQL users excluded from masking (administrators are always excluded)'. A table titled 'Recommended fields to mask' lists fields from various schemas and tables, each with an 'Add mask' button.

Looking into Dynamic Data Masking Policies:

- **SQL users excluded from Dynamic Data Masking Policies**

The following SQL users or Azure AD identities can get unmasked data in the SQL query results. Users with administrator privileges are always excluded from masking, and will see the original data without any mask.

- **Masking rules** - Masking rules are a set of rules that define the designated fields to be masked including the masking function that is used. The designated fields can be defined using a database schema name, table name, and column name.
- **Masking functions** - Masking functions are a set of methods that control the exposure of data for different scenarios.

Set up Dynamic Data Masking for your database in Azure Synapse Analytics using PowerShell cmdlets

In this part, we're going to look into Dynamic Data Masking for a database in Azure Synapse Analytics using PowerShell cmdlets.

- Data masking policies

- Get-AzSqlDatabaseDataMaskingPolicy

The Get-AzSqlDatabaseDataMaskingPolicy gets the data masking policy for a database.

The syntax for the Get-AzSqlDatabaseDataMaskingPolicy in PowerShell is as follows:

```
Get-AzSqlDatabaseDataMaskingPolicy [-ServerName] <String> [-DatabaseName]
<String>
[-ResourceGroupName] <String> [-DefaultProfile <IAzureContextContainer>]
[-WhatIf] [-Confirm]
[<CommonParameters>]
```

What the **Get-AzSqlDatabaseDataMaskingPolicy** cmdlet does, is getting the data masking policy of an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the database:

- *ResourceGroupName*: name of the resource group you deployed the database in
- *ServerName*: sql server name
- *DatabaseName* : name of the database

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

- Set-AzSqlDatabaseDataMaskingPolicy

The Set-AzSqlDatabaseDataMaskingPolicy sets data masking for a database.

The syntax for the Set-AzSqlDatabaseDataMaskingPolicy in PowerShell is as follows:

```
Set-AzSqlDatabaseDataMaskingPolicy [-PassThru] [-PrivilegedUsers <String>]
[-DataMaskingState <String>]
[-ServerName] <String> [-DatabaseName] <String> [-ResourceGroupName]
<String>
[-DefaultProfile <IAzureContextContainer>] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

What the **Set-AzSqlDatabaseDataMaskingPolicy** cmdlet does is setting the data masking policy for an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the database:

- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database

In addition, you'll need to set the *DataMaskingState* parameter to specify whether data masking operations are enabled or disabled.

If the cmdlet succeeds and the *PassThru* parameter is used, it will return an object describing the current data masking policy in addition to the database identifiers.

Database identifiers can include, **ResourceGroupName**, **ServerName**, and **DatabaseName**.

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

- Data masking rules
- Get-AzSqlDatabaseDataMaskingRule

The **Get-AzSqlDatabaseDataMaskingRule** Gets the data masking rules from a database.

The syntax for the **Get-AzSqlDatabaseDataMaskingRule** in PowerShell is as follows:

```
Get-AzSqlDatabaseDataMaskingRule [-SchemaName <String>] [-TableName <String>]
[-ColumnName <String>] [-ServerName] <String> [-DatabaseName] <String> [-ResourceGroupName] <String>
[-DefaultProfile <IAzureContextContainer>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

What the **Get-AzSqlDatabaseDataMaskingRule** cmdlet does it getting either a specific data masking rule or all of the data masking rules for an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the database:

- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database

You'd also have to specify the *RuleId* parameter to specify which rule this cmdlet returns.

If you don't provide *RuleId*, all the data masking rules for that Azure SQL database are returned.

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

- `New-AzSqlDatabaseDataMaskingRule`

The `New-AzSqlDatabaseDataMaskingRule` creates a data masking rule for a database.

The syntax for the `New-AzSqlDatabaseDataMaskingRule` in PowerShell is as follows:

```
 New-AzSqlDatabaseDataMaskingRule -MaskingFunction <String> [-PrefixSize <UInt32>] [-ReplacementString <String>] [-SuffixSize <UInt32>] [-NumberFrom <Double>] [-NumberTo <Double>] [-PassThru] -SchemaName <String> -TableName <String> -ColumnName <String> [-ServerName] <String> [-DatabaseName] <String> [-ResourceGroupName] <String> [-DefaultProfile <IAzureContextContainer>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

What the `New-AzSqlDatabaseDataMaskingRule` cmdlet does is creating a data masking rule for an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the rule:

- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database

Providing the *TableName* and *ColumnName* is necessary in order to specify the target of the rule.

The *MaskingFunction* parameter is necessary to define how the data is masked.

If *MaskingFunction* has a value of Number or Text, you can specify the *NumberFrom* and *NumberTo* parameters, for number masking, or the *PrefixSize*, *ReplacementString*, and *SuffixSize* for text masking.

If the command succeeds and the *PassThru* parameter is used, the cmdlet returns an object describing the data masking rule properties in addition to the rule identifiers.

Rule identifiers can be, for example, *ResourceGroupName*, *ServerName*, *DatabaseName*, and *RuleID*.

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

- `Remove-AzSqlDatabaseDataMaskingRule`

The Remove-AzSqlDatabaseDataMaskingRule removes a data masking rule from a database.

The syntax for the Remove-AzSqlDatabaseDataMaskingRule in PowerShell is as follows:

```
Remove-AzSqlDatabaseDataMaskingRule [-PassThru] [-Force] -SchemaName <String> -TableName <String> -ColumnName <String> [-ServerName] <String> [-DatabaseName] <String> [-ResourceGroupName] <String> [-DefaultProfile <IAzureContextContainer>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

What the **Remove-AzSqlDatabaseDataMaskingRule** cmdlet does, is it removes a specific data masking rule from an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the rule that needs to be removed:

- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database
- *RuleId* : identifier of the rule

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

- Set-AzSqlDatabaseDataMaskingRule

The Set-AzSqlDatabaseDataMaskingRule Sets the properties of a data masking rule for a database.

The syntax for the Set-AzSqlDatabaseDataMaskingRule in PowerShell is as follows:

```
Set-AzSqlDatabaseDataMaskingRule [-MaskingFunction <String>] [-PrefixSize <UInt32>] [-ReplacementString <String>] [-SuffixSize <UInt32>] [-NumberFrom <Double>] [-NumberTo <Double>] [-PassThru] -SchemaName <String> -TableName <String> -ColumnName <String> [-ServerName] <String> [-DatabaseName] <String> [-ResourceGroupName] <String> [-DefaultProfile <IAzureContextContainer>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

What the **Set-AzSqlDatabaseDataMaskingRule** cmdlet does is setting a data masking rule for an Azure SQL database.

To use this cmdlet in PowerShell, you'd have to specify the following parameters to identify the rule:

- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database
- *RuleId* : identifier of the rule

You can provide any of the parameters of *SchemaName*, *TableName*, and *ColumnName* to retarget the rule.

Specify the *MaskingFunction* parameter to modify how the data is masked.

If you specify a value of Number or Text for *MaskingFunction*, you can specify the *NumberFrom* and *NumberTo* parameters for number masking or the *PrefixSize*, *ReplacementString*, and *SuffixSize* parameters for text masking.

If the command succeeds, and if you specify the *PassThru* parameter, the cmdlet returns an object that describes the data masking rule properties and the rule identifiers.

Rule identifiers can be, **ResourceGroupName**, **ServerName**, **DatabaseName**, and **RuleId**.

This cmdlet is also supported by the SQL Server Stretch Database service on Azure.

Set up Dynamic Data Masking for your database in Azure Synapse Analytics using the REST API

For setting up Dynamic Data Masking in Azure Synapse Analytics, you can also make use of the REST API. It will enable you to programmatically manage data masking policy and rules.

The REST API will support the following operations:

- Data masking policies
- Create Or Update

The Create Or Update masking policy using the REST API will create or update a database data masking policy.

In HTTP the following request can be made: > **Note:** The date of the API will change over time and the version you use will be determined by your needs and the functionality required.

HTTP

```
GET https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/dat
```

The following parameters need to be passed through:

- *SubscriptionID*: the ID of the subscription
 - *ResourceGroupName*: name of the resource group that you deployed the database in
 - *ServerName* : sql server name
 - *DatabaseName* : name of the database
 - *dataMaskingPolicyName*: the name of the data masking policy
 - *api version*: version of the api that is used.
- Get

The Get policy, gets a database data masking policy.

In HTTP the following request can be made:

HTTP

```
GET https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/databases/{databaseName}/dataMaskingPolicies/Default?api-version=2021-06-01
```

The following parameters need to be passed through:

- *SubscriptionID*: the ID of the subscription
 - *ResourceGroupName*: name of the resource group that you deployed the database in
 - *ServerName* : sql server name
 - *DatabaseName* : name of the database
 - *dataMaskingPolicyName*: the name of the data masking policy
 - *api version*: version of the api that is used.
- Data masking rules
- Create Or Update

The Create or Update masking rule creates or updates a database data masking rule.

In HTTP the following request can be made:

HTTP

```
PUT https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/dat
```

```
abases/{databaseName}/dataMaskingPolicies/Default/rules/{dataMaskingRule-  
Name}?api-version=2021-06-01
```

The following parameters need to be passed through:

- *SubscriptionID*: the ID of the subscription
 - *ResourceGroupName*: name of the resource group that you deployed the database in
 - *ServerName* : sql server name
 - *DatabaseName* : name of the database
 - *dataMaskingPolicyName*: the name of the data masking policy
 - *dataMaskingRuleName*: the name of the rule for data masking
 - *api version*: version of the api that is used.
-
- List By Database

The List By Database request gets a list of database data masking rules.

In HTTP the following request can be made:

HTTP

```
GET https://management.azure.com/subscriptions/{subscriptionId}/resource-  
Groups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/dat  
abases/{databaseName}/dataMaskingPolicies/Default/rules?api-version=2021-  
06-01
```

The following parameters need to be passed through:

- *SubscriptionID*: the ID of the subscription
- *ResourceGroupName*: name of the resource group that you deployed the database in
- *ServerName* : sql server name
- *DatabaseName* : name of the database
- *dataMaskingPolicyName*: the name of the data masking policy
- *api version*: version of the api that is used.

Next unit: Implement encryption in Azure Synapse Analytics

Continue >

How are we doing? ★ ★ ★ ★ ★

100 XP

Implement encryption in Azure Synapse Analytics

3 minutes

In this section, we will go through Transparent Data Encryption and TokenLibrary for Apache Spark.

What is transparent data encryption

Transparent data encryption (TDE) is an encryption mechanism to help you protect Azure Synapse Analytics. It will protect Azure Synapse Analytics against threats of malicious offline activity. The way TDE will do so is by encrypting data at rest. TDE performs real-time encryption as well as decryption of the database, associated backups, and transaction log files at rest without you having to make changes to the application. In order to use TDE for Azure Synapse Analytics, you will have to manually enable it.

What TDE does is performing I/O encryption and decryption of data at the page level in real time. When a page is read into memory, it is decrypted. It is encrypted before writing it to disk. TDE encrypts the entire database storage using a symmetric key called a Database Encryption Key (DEK). When you start up a database, the encrypted Database Encryption Key is decrypted. The DEK will then be used for decryption and re-encryption of the database files in the SQL Server database engine. The DEK is protected by the Transparent Data Encryption Protector. This protector can be either a service-managed certificate, which is referred to as service-managed transparent data encryption, or an asymmetric key that is stored in Azure Key Vault (customer-managed transparent data encryption).

What is important to understand is that for Azure Synapse Analytics, this TDE protector is set on the server level. There it is inherited by all the databases that are attached or aligned to that server. The term server refers both to server and instance.

Service-managed transparent data encryption

As stated above, the DEK that is protected by the Transparent Encryption protector can be service-managed certificates which we call service-managed TDE. When you look in Azure, that default setting means that the DEK is protected by a built-in certificate unique for each server with encryption algorithm AES256. When a database is in a geo-replicated relationship then

primary and the geo-secondary database are protected by the primary database's parent server key. If the databases are connected to the same server, they will also have the same built-in AES 256 certificate. As Microsoft we automatically rotate the certificates in compliance with the internal security policy. The root key is protected by a Microsoft internal secret store. Microsoft also seamlessly moves and manages the keys as needed for geo-replication and restores.

Transparent data encryption with bring your own key for customer-managed transparent data encryption

As stated above, the DEK that is protected by the Transparent Data Encryption Protector can also be customer managed by bringing an asymmetric key that is stored in Azure Key Vault (customer-managed transparent data encryption). This is also referred to as Bring Your Own Key (BYOK) support for TDE. When this is the scenario that is applicable to you, the TDE Protector that encrypts the DEK is a customer-managed asymmetric key. It is stored in your own and managed Azure Key Vault. Azure Key Vault is Azure's cloud-based external key management system. This managed key never leaves the key vault. The TDE Protector can be generated by the key vault. Another option is to transfer the TDE Protector to the key vault from, for example, an on-premise hardware security module (HSM) device. Azure Synapse Analytics needs to be granted permissions to the customer-owned key vault in order to decrypt and encrypt the DEK. If permissions of the server to the key vault are revoked, a database will be inaccessible, and all data is encrypted.

By using Azure Key Vault integration for TDE, you have control over the key management tasks such as key rotations, key backups, and key permissions. It also enables you to audit and report on all the TDE protectors when using the Azure Key Vault functionality. The reason for using Key Vault is that it provides you with a central key management system where tightly monitored HSMs are leveraged. It also enables you to separate duties of management of keys and data in order to meet compliance with security policies.

Manage transparent data encryption in the Azure portal.

For Azure Synapse Analytics, you can manage TDE for the database in the Azure portal after you've signed in with the Azure Administrator or Contributor account. The TDE settings can be found under your user database.

The screenshot shows the Azure portal interface for managing a Dedicated SQL pool. The top navigation bar includes 'Save', 'Discard', 'Refresh', and 'Feedback' buttons. The main content area displays information about transparent data encryption, stating it encrypts databases, backups, and logs at rest without changes to the application. A 'Learn more' link is provided. Below this, a section titled 'Data encryption' has a switch that is currently set to 'OFF'. The 'Encryption status' is shown as 'Unencrypted'. On the left side, there's a navigation menu with sections like 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Settings' (which is expanded to show 'Workload management', 'Maintenance schedule', 'Geo-backup policy', 'Connection strings', 'Properties', and 'Locks'), 'Security' (which is expanded to show 'Auditing', 'Data Discovery & Classification', 'Dynamic Data Masking', 'Security Center', and 'Transparent data encryption'), 'Common Tasks' (which is expanded to show 'Open in Visual Studio'), and 'Open in Visual Studio'.

It is by default that the service-managed TDE is used and therefore a TDE certificate is automatically generated for the server that contains that database.

Moving a transparent data encryption protected database

In some use cases you need to move a database that is protected with TDE. Within Azure, there is no need to decrypt the databases. The TDE settings on the source database or primary database, will be inherited on the target. Some of the operations within Azure that inherited the TDE are:

- Geo-restore
- Self-service point-in-time restore
- Restoration of a deleted database
- Active geo-replication
- Creation of a database copy
- Restore of backup file to Azure SQL Managed Instance

If you export a TDE-protected database, the exported content is not encrypted. This will be stored in an unencrypted BACPAC file. You need to make sure that you protect this BACPAC file and enable TDE as soon as the import of the bacpac file in the new database is finished.

Securing your credentials through linked services with TokenLibrary for Apache Spark

It is quite a common pattern to access data from external sources. Unless the external data source allows anonymous access, it is highly likely that you need to secure your connection with a credential, secret, or connection string.

Within Azure Synapse Analytics, the integration process is simplified by providing linked services. Doing so, the connection details can be stored in the linked service or an Azure Key Vault. If the Linked Service is created, Apache spark can reference the linked service to apply the connection information in your code. When you want to access files from the Azure Data Lake Storage Gen 2 within your Azure Synapse Analytics Workspace, it uses AAD passthrough for the authentication. Therefore, there is no need to use TokenLibrary. However, to connect to other linked services, you are enabled to make a direct call to the TokenLibrary.

An example can be found below: In order to connect to other linked services, you are enabled to make a direct call to TokenLibrary by retrieving the connection string. In order to retrieve the connection string, use the **getConnectionString** function and pass in the **linked service name**.

Scala

```
// Scala  
// retrieve connectionstring from TokenLibrary  
  
import com.microsoft.azure.synapse.tokenlibrary.TokenLibrary  
  
val connectionString: String = TokenLibrary.getConnectionString("<LINKED  
SERVICE NAME>")  
println(connectionString)
```

Python

```
# Python  
# retrieve connectionstring from TokenLibrary  
  
from pyspark.sql import SparkSession  
  
sc = SparkSession.builder.getOrCreate()  
token_library = sc._jvm.com.microsoft.azure.synapse.tokenlibrary.TokenLi-  
brary
```

```
connection_string = token_library.getConnectionString("<LINKED SERVICE NAME>")
print(connection_string)
```

If you want to Get the connection string as map and parse specific values from a key in the connection string, you can find an example below:

To parse specific values from a *key=value* pair in the connection string such as

DefaultEndpointsProtocol=https;AccountName=<AccountName>;AccountKey=<AccountKey>

use the `getConnectionStringAsMap` function and pass the key to return the value.

Scala

```
// Linked services can be used for storing and retrieving credentials (e.g,
// account key)
// Example connection string (for storage):
"DefaultEndpointsProtocol=https;AccountName=<accountname>;AccountKey=<ac-
countkey>"
import com.microsoft.azure.synapse.tokenlibrary.TokenLibrary

val accountKey: String = TokenLibrary.getConnectionStringAsMap("<LINKED
SERVICE NAME>").get("<KEY NAME>")
println(accountKey)
```

Python

```
# Linked services can be used for storing and retrieving credentials (e.g,
# account key)
# Example connection string (for storage):
"DefaultEndpointsProtocol=https;AccountName=<accountname>;AccountKey=<ac-
countkey>"
from pyspark.sql import SparkSession

sc = SparkSession.builder.getOrCreate()
token_library = sc._jvm.com.microsoft.azure.synapse.tokenlibrary.TokenLi-
brary
accountKey = token_library.getConnectionStringAsMap("<LINKED SERVICE
NAME>").get("<KEY NAME>")
print(accountKey)
```

Next unit: Knowledge check

✓ 200 XP



Knowledge check

3 minutes

1. You want to configure a private endpoint. You open up Azure Synapse Studio, go to the manage hub, and see that the private endpoints are greyed out. Why is the option not available? *



Azure Synapse Studio doesn't support the creation of private endpoints.

X Incorrect. Azure Synapse Studio does support the creation of private endpoints.



A Conditional Access policy has to be defined first.



A managed virtual network hasn't been created.

✓ Correct. In order to create a private endpoint, you first must create a managed virtual network.

2. You require an Azure Synapse Analytics Workspace to access an Azure Data Lake Store using the benefits of the security provided by Azure Active Directory. What is the best authentication method to use? *



Storage account keys.



Shared access signatures.



Managed identities.

✓ Correct. Managed identities provide Azure services with an automatically managed identity in Azure Active Directory. You can use the Managed Identity capability to authenticate to any service that supports Azure Active Directory authentication.

Next unit: Summary

100 XP

Introduction

1 minute

With the wide range of data stores available in Azure, there's the need to manage and orchestrate the movement data between them. In fact, you'll usually want to automate *extract, transform, and load* (ETL) workloads as a regular process in a wider enterprise analytical solution. *Pipelines* are a mechanism for defining and orchestrating data movement activities. In this module, you'll be introduced to Azure Synapse Analytics pipelines, their component parts, and how to implement and run a pipeline in Azure Synapse Studio.

Note

Azure Synapse Analytics pipelines are built on the same technology as Azure Data Factory, and offer a similar authoring experience. The authoring processes described in this module are also applicable to Azure Data Factory. For a detailed discussion of the differences between Azure Synapse Analytics pipelines and Azure Data Factory, see [Data integration in Azure Synapse Analytics versus Azure Data Factory](#).

Next unit: Understand pipelines in Azure Synapse Analytics

[Continue >](#)

How are we doing?

✓ 100 XP



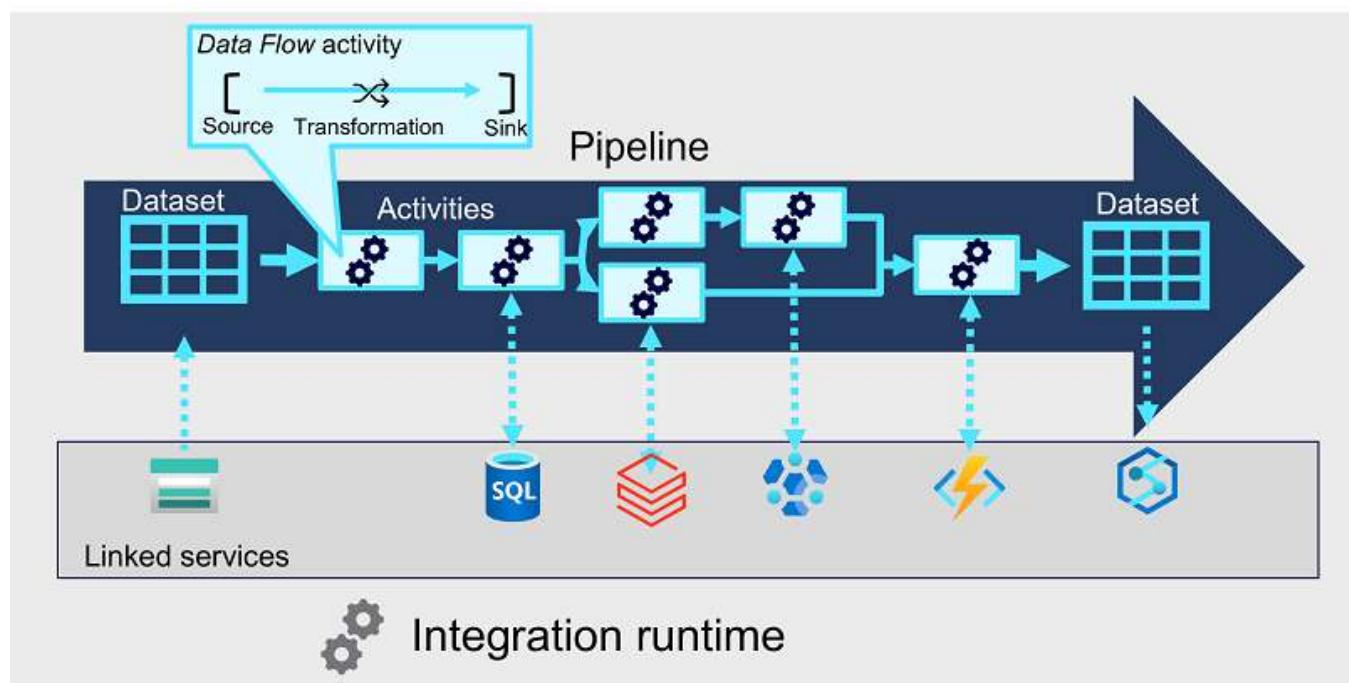
Understand pipelines in Azure Synapse Analytics

6 minutes

Pipelines in Azure Synapse Analytics encapsulate a sequence of *activities* that perform data movement and processing tasks. You can use a pipeline to define data transfer and transformation activities, and orchestrate these activities through control flow activities that manage branching, looping, and other typical processing logic. The graphical design tools in Azure Synapse Studio enable you to build complex pipelines with minimal or no coding required.

Core pipeline concepts

Before building pipelines in Azure Synapse Analytics, you should understand a few core concepts.



Activities

Activities are the executable tasks in a pipeline. You can define a flow of activities by connecting them in a sequence. The outcome of a particular activity (success, failure, or completion) can be used to direct the flow to the next activity in the sequence.

Activities can encapsulate data transfer operations, including simple data copy operations that extract data from a source and load it to a target (or *sink*), as well as more complex data flows that apply transformations to the data as part of an *extract, transfer, and load* (ETL) operation. Additionally, there are activities that encapsulate processing tasks on specific systems, such as running a Spark notebook or calling an Azure function. Finally, there are *control flow* activities that you can use to implement loops, conditional branching, or manage variable and parameter values.

Integration runtime

The pipeline requires compute resources and an execution context in which to run. The pipeline's *integration runtime* provides this context, and is used to initiate and coordinate the activities in the pipeline.

Linked services

While many of the activities are run directly in the integration runtime for the pipeline, some activities depend on external services. For example, a pipeline might include an activity to run a notebook in Azure Databricks or to call a stored procedure in Azure SQL Database. To enable secure connections to the external services used by your pipelines, you must define *linked services* for them.

 **Note**

Linked services are defined at the Azure Synapse Analytics workspace level, and can be shared across multiple pipelines.

Datasets

Most pipelines process data, and the specific data that is consumed and produced by activities in a pipeline is defined using *datasets*. A dataset defines the schema for each data object that will be used in the pipeline, and has an associated linked service to connect to its source. Activities can have datasets as inputs or outputs.

 **Note**

Similarly to linked services, datasets are defined at the Azure Synapse Analytics workspace level, and can be shared across multiple pipelines.

✓ 100 XP

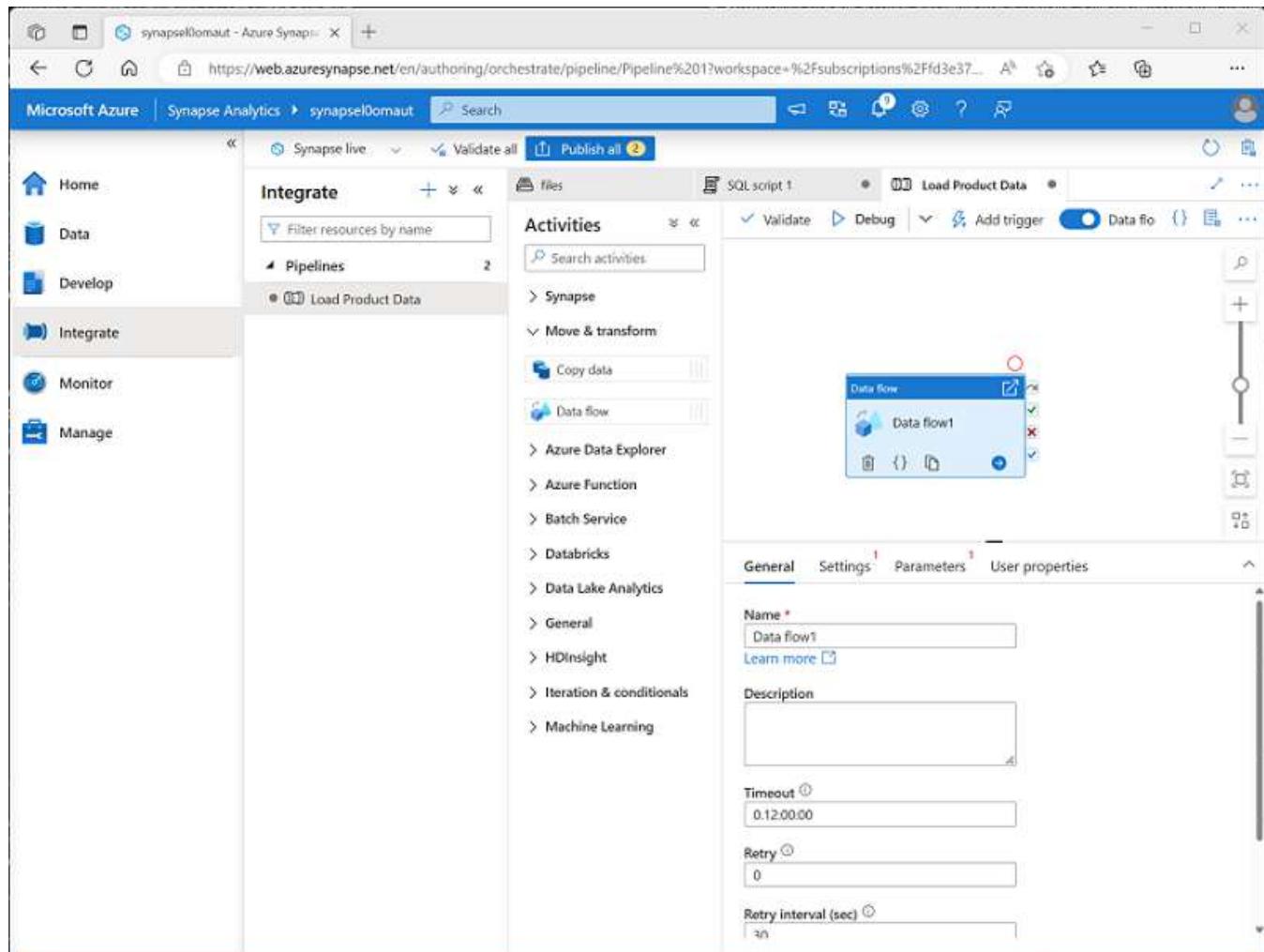


Create a pipeline in Azure Synapse Studio

5 minutes

You can create a pipeline in Azure Synapse Studio by using shortcuts on the **Home** page, but the primary place where pipelines are created and managed is the **Integrate** page.

When you create a pipeline in Azure Synapse Studio, you can use the graphical design interface.



The pipeline designer includes a set of activities, organized into categories, which you can drag onto a visual design canvas. You can select each activity on the canvas and use the properties pane beneath the canvas to configure the settings for that activity.

To define the logical sequence of activities, you can connect them by using the **Succeeded**, **Failed**, and **Completed** dependency conditions, which are shown as small icons on the right-hand edge of each activity.

Defining a pipeline with JSON

While the graphical development environment is the preferred way to create a pipeline, you can also create or edit the underlying JSON definition of a pipeline. The following code example shows the JSON definition of a pipeline that includes a **Copy Data** activity:

JSON

```
{  
  "name": "CopyPipeline",  
  "properties": {  
    "description": "Copy data from a blob to Azure SQL table",  
    "activities": [  
      {  
        "name": "CopyFromBlobToSQL",  
        "type": "Copy",  
        "inputs": [  
          {  
            "name": "InputDataset"  
          }  
        ],  
        "outputs": [  
          {  
            "name": "OutputDataset"  
          }  
        ],  
        "typeProperties": {  
          "source": {  
            "type": "BlobSource"  
          },  
          "sink": {  
            "type": "SqlSink",  
            "writeBatchSize": 10000,  
            "writeBatchTimeout": "60:00:00"  
          }  
        },  
        "policy": {  
          "retry": 2,  
          "timeout": "01:00:00"  
        }  
      }  
    ]  
  }  
}
```

Next unit: Define data flows

✓ 100 XP



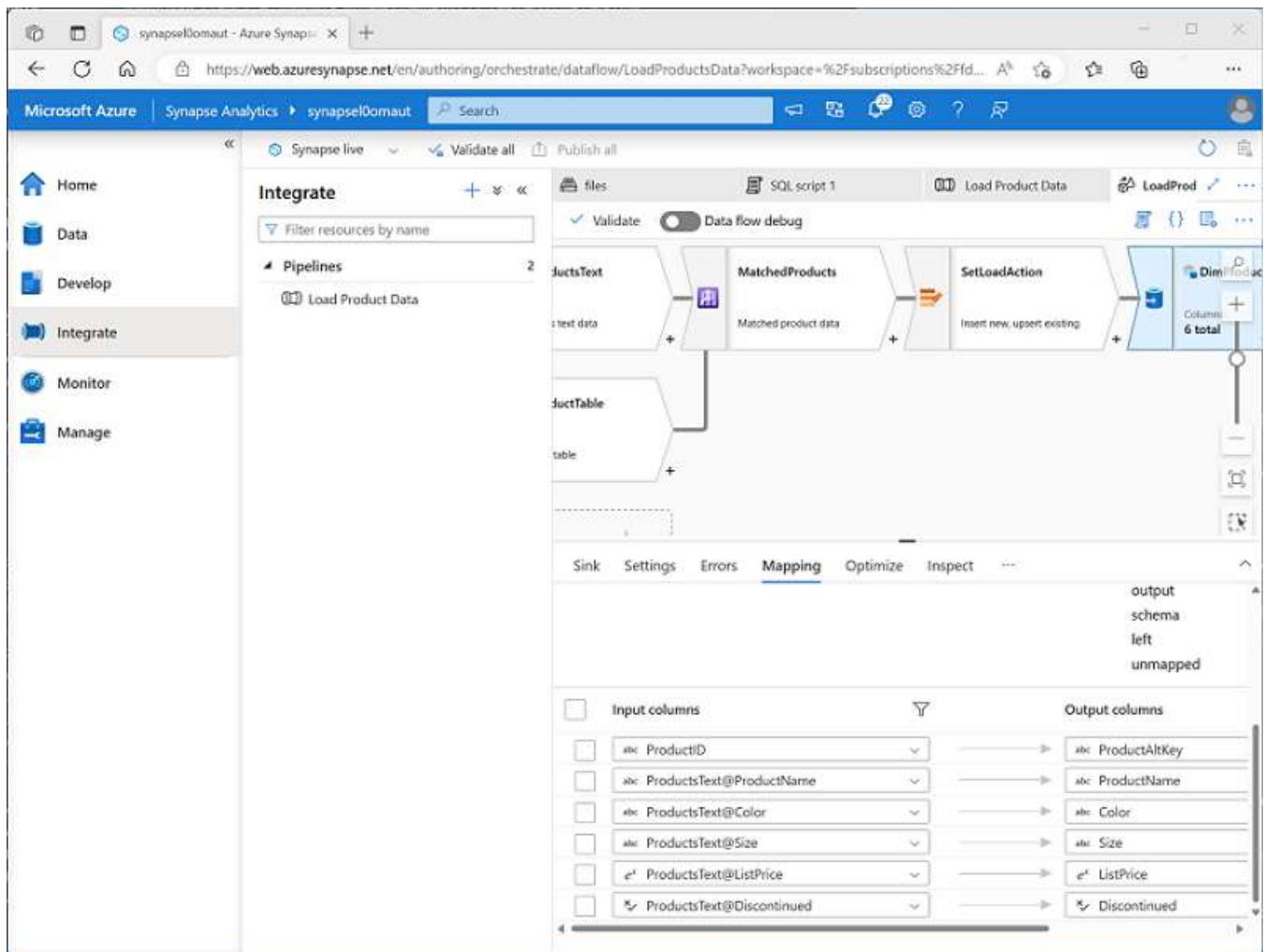
Define data flows

5 minutes

A **Data Flow** is a commonly used activity type to define data flow and transformation. Data flows consist of:

- **Sources** - The input data to be transferred.
- **Transformations** – Various operations that you can apply to data as it streams through the data flow.
- **Sinks** – Targets into which the data will be loaded.

When you add a **Data Flow** activity to a pipeline, you can open it in a separate graphical design interface in which to create and configure the required data flow elements.



An important part of creating a data flow is to define mappings for the columns as the data flows through the various stages, ensuring column names and data types are defined appropriately. While developing a data flow, you can enable the **Data flow debug** option to

pass a subset of data through the flow, which can be useful to test that your columns are mapped correctly.

 **Tip**

To learn more about implementing a Data Flow activity, see [Data Flow activity in Azure Data Factory and Azure Synapse Analytics](#) in the Azure documentation.

Next unit: Run a pipeline

[Continue >](#)

How are we doing?     

✓ 100 XP ➔

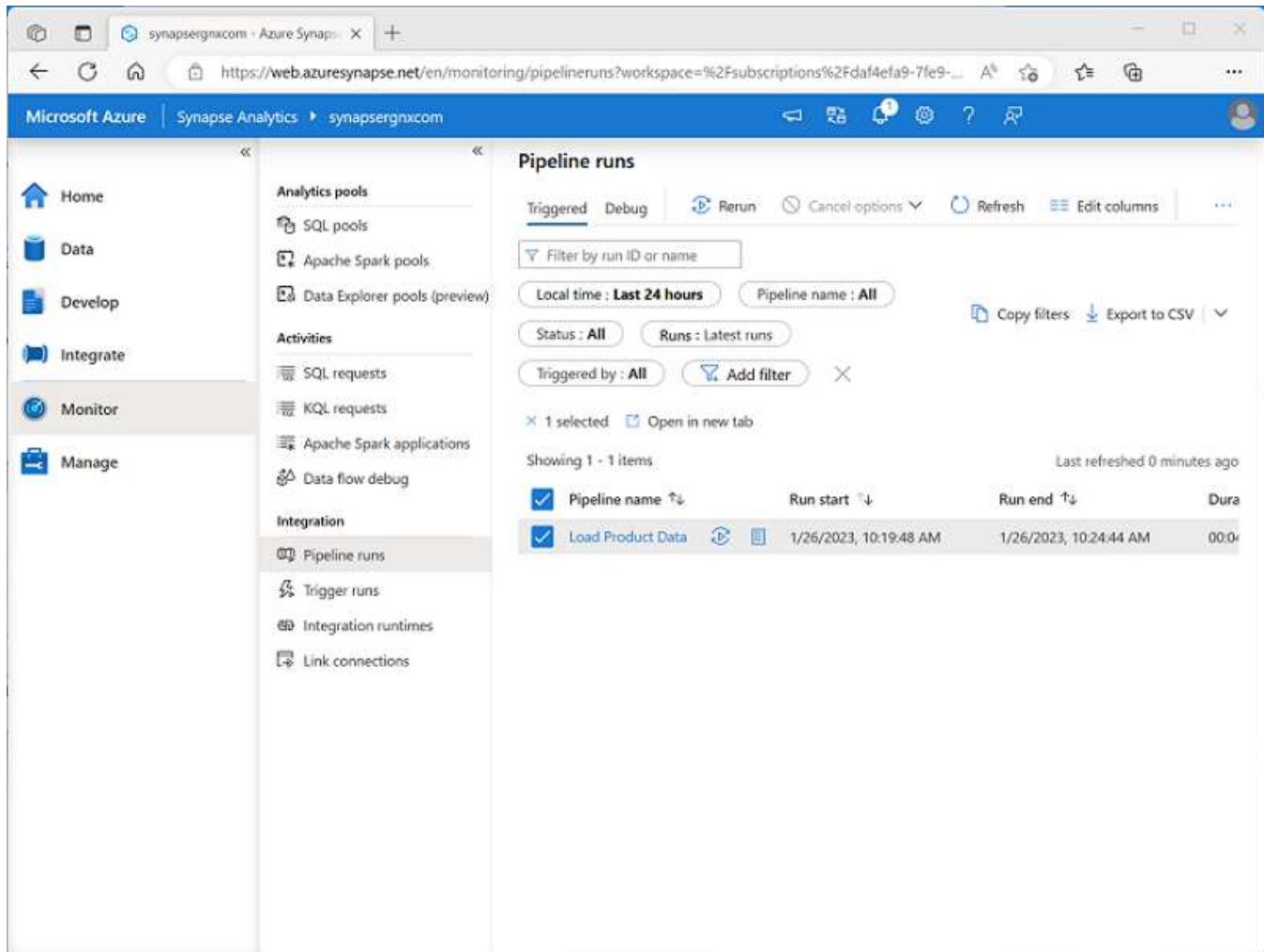
Run a pipeline

5 minutes

When you're ready, you can publish a pipeline and use a trigger to run it. Triggers can be defined to run the pipeline:

- Immediately
- At explicitly scheduled intervals
- In response to an event, such as new data files being added to a folder in a data lake.

You can monitor each individual run of a pipeline in the **Monitor** page in Azure Synapse Studio.



The screenshot shows the 'Pipeline runs' section of the Azure Synapse Analytics studio. The left sidebar has 'Monitor' selected under the 'Analytics pools' category. The main area displays a table of pipeline runs. The first run listed is 'Load Product Data', which was triggered at 1/26/2023, 10:19:48 AM, completed at 1/26/2023, 10:24:44 AM, and had a duration of 00:05. The table includes columns for Pipeline name, Run start, Run end, and Duration. Filter options at the top include 'Triggered' (selected), 'Debug', 'Rerun', 'Cancel options', 'Refresh', 'Edit columns', and date/time filters for Local time and Pipeline name. A 'Copy filters' and 'Export to CSV' button are also present.

The ability to monitor past and ongoing pipeline runs is useful for troubleshooting purposes. Additionally, when combined with the ability to integrate Azure Synapse Analytics and Microsoft Purview, you can use pipeline run history to track data lineage data flows.

Tip

To learn more about integration between Azure Synapse Analytics and Microsoft Purview, consider completing the [Integrate Microsoft Purview and Azure Synapse Analytics](#) module.

Next unit: Exercise - Build a data pipeline in Azure Synapse Analytics

[Continue >](#)

How are we doing?     

✓ 200 XP



Knowledge check

3 minutes

1. What does a pipeline use to access external data source and processing resources? *

Data Explorer pools

✗ Incorrect. Data Explorer pools are used to run Kusto queries.

Linked services

✓ Correct. Linked services encapsulate connections to external services.

External tables

2. What kind of object should you add to a data flow to define a target to which data is loaded? *

Source

Transformation

✗ Incorrect. You can use various transformations to modify data in a data flow.

Sink

✓ Correct. A sink represents a target in a data flow.

3. What must you create to run a pipeline at scheduled intervals? *

A control flow

A trigger

✓ Correct. A trigger is used to initiate a pipeline run.

An activity

Next unit: Summary

100 XP

Introduction

1 minute

With Azure Synapse Analytics pipelines, you can orchestrate data transfer and transformation activities and build data integration solutions across multiple systems. When you're working with analytical data in a data lake, Apache Spark provides a scalable, distributed processing platform that you can use to process huge volumes of data efficiently.

The **Synapse Notebook** activity enables you to run data processing code in Spark notebooks as a task in a pipeline; making it possible to automate big data processing and integrate it into *extract, transform, and load* (ETL) workloads.

Next unit: Understand Synapse Notebooks and Pipelines

[Continue >](#)

How are we doing?

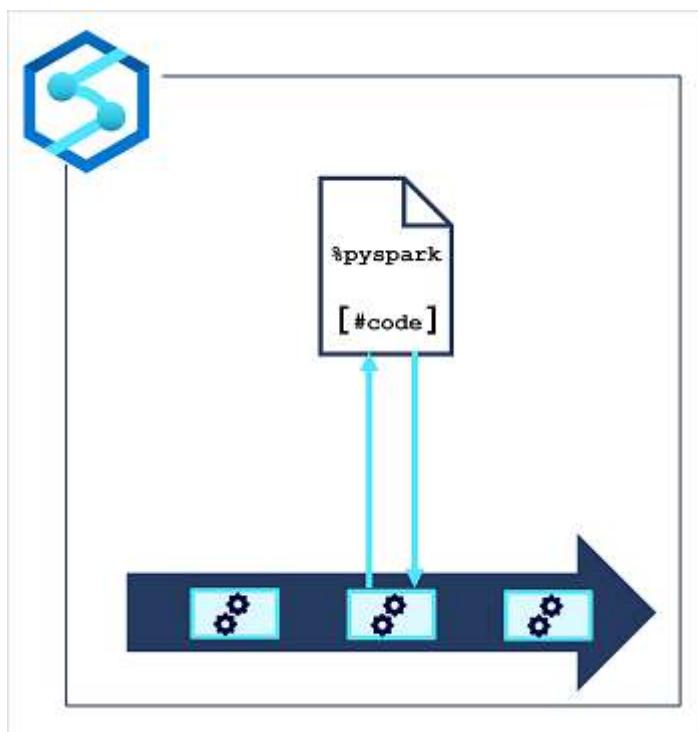
✓ 100 XP



Understand Synapse Notebooks and Pipelines

5 minutes

Azure Synapse Pipelines enable you to create, run, and manage data integration and data flow activities. While many of these activities are built-into the Azure Synapse Pipeline platform and run natively in the integration runtime for your pipeline, you can also use external processing resources to perform specific tasks. One such external resource is an Apache Spark pool in your Azure Synapse Analytics workspace on which you can run code in a notebook.



It's common in big data analytics solutions for data engineers to use Spark notebooks for initial data exploration and interactive experimentation when designing data transformation processes. When the transformation logic has been completed, you can perform some final code optimization and refactoring for maintainability, and then include the notebook in a pipeline. The pipeline can then be run on a schedule or in response to an event (such as new data files being loaded into the data lake).

The notebook is run on a Spark pool, which you can configure with the appropriate compute resources and Spark runtime for your specific workload. The pipeline itself is run in an integration runtime that orchestrates the activities in the pipeline, coordinating the external services needed to run them.

Tip

There are several best practices that can help make working with Spark notebooks more efficient and effective. Some of these include:

- Keep your code organized: Use clear and descriptive variable and function names, and organize your code into small, reusable chunks.
- Cache intermediate results: Spark allows you to cache intermediate results, which can significantly speed up the performance of your notebook.
- Avoid unnecessary computations: Be mindful of the computations you are performing and try to avoid unnecessary steps. For example, if you only need a subset of your data, filter it out before running any further computations.
- Avoid using collect() unless necessary: When working with large datasets, it is often better to perform operations on the entire dataset rather than bringing the data into the driver node using the collect() method.
- Use Spark UI for monitoring and debugging: Spark's web-based user interface (UI) provides detailed information about the performance of your Spark jobs, including task execution times, input and output data sizes, and more.
- Keep your dependencies version-consistent and updated: when working with Spark, it is important to keep dependencies version-consistent across your cluster and to use the latest version of Spark and other dependencies if possible.

Next unit: Use a Synapse notebook activity in a pipeline

[Continue >](#)

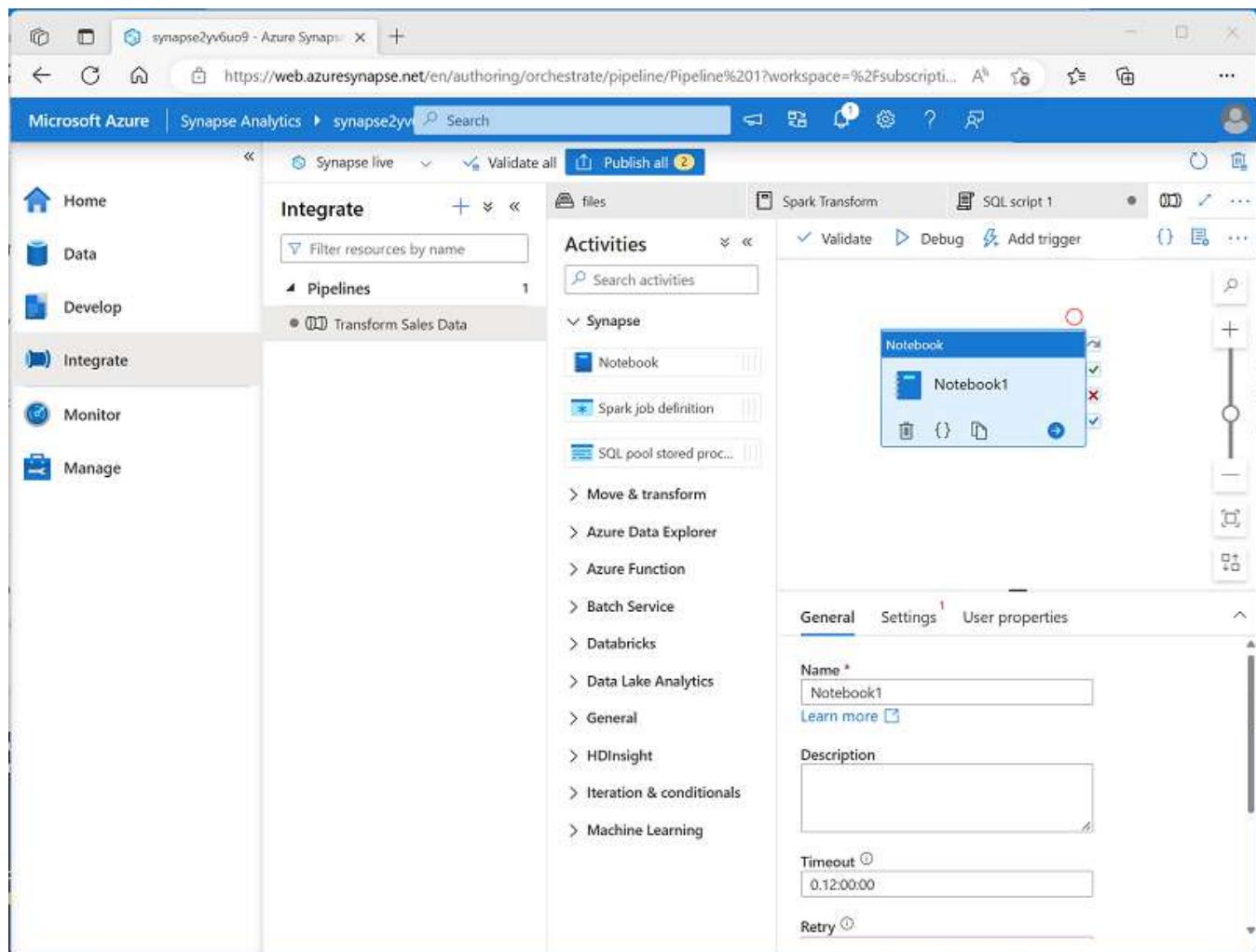
How are we doing?     

✓ 100 XP

Use a Synapse notebook activity in a pipeline

3 minutes

To run a Spark notebook in a pipeline, you must add a notebook activity and configure it appropriately. You'll find the **Notebook** activity in the **Synapse** section of the activities pane in the Azure Synapse Analytics pipeline designer.



Tip

You can also add a notebook to a pipeline from within the notebook editor.

To configure the notebook activity, edit the settings in the properties pane beneath the pipeline designer canvas. Notebook activity specific settings include:

- **Notebook:** The notebook you want to run. You can select an existing notebook in your Azure Synapse Analytics workspace, or create a new one.
 - **Spark pool:** The Apache Spark pool on which the notebook should be run.
 - **Executor size:** The node size for the worker nodes in the pool, which determines the number of processor cores and the amount of memory allocated to worker nodes.
 - **Dynamically allocate executors:** Configures Spark dynamic allocation, enabling the pool to automatically scale up and down to support the workload.
 - **Min executors:** The minimum number of executors to be allocated.
 - **Max executors:** The maximum number of executors to be allocated.
 - **Driver size:** The node size for the driver node.
-

Next unit: Use parameters in a notebook

[Continue >](#)

How are we doing?

✓ 100 XP ➔

Use parameters in a notebook

5 minutes

Parameters enable you to dynamically pass values for variables in the notebook each time it's run. This approach provides flexibility, enabling you to adjust the logic encapsulated in the notebook for each run.

Create a *parameters* cell in the notebook

To define the parameters for a notebook, you declare and initialize variables in a cell, which you then configure as a **Parameters** cell by using the toggle option in the notebook editor interface.



```
1 import uuid
2
3 # Variable for unique folder name
4 folderName = uuid.uuid4()
```

Press shift + enter to run

Parameters

A screenshot of a Jupyter Notebook cell. The cell contains Python code: `import uuid`, `# Variable for unique folder name`, and `folderName = uuid.uuid4()`. Below the code, there is a note: "Press shift + enter to run". In the top right corner of the cell, there is a toolbar with icons for copy, cut, paste, and delete. To the left of the cell, there is a vertical toolbar with a play button and a dropdown menu. The word "Parameters" is visible at the bottom right of the cell area.

Initializing a variable ensures that it has a default value, which will be used if the parameter isn't set in the notebook activity.

Set *base parameters* for the notebook activity

After defining a parameters cell in the notebook, you can set values to be used when the notebook is run by a notebook activity in a pipeline. To set parameter values, expand and edit the **Base parameters** section of the settings for the activity.

✓ Validate ▶ Debug ⚡ Add trigger

Notebook

Run Spark Transform

General Settings User properties

Notebook Spark Transform Open New

Base parameters

+ New Delete

Name	Type	Value
folderName	String	@pipeline().RunId

Spark pool spark2yv6uo9

Executor size Small(4 vCores, 28GB memory)

Dynamically allocate executors Enabled

Min executors

Max executors

Driver size Small(4 vCores, 28GB memory)

You can assign explicit parameter values, or use an expression to assign a dynamic value. For example, the expression `@pipeline().RunId` returns the unique identifier for the current run of the pipeline.

Next unit: Exercise - Use an Apache Spark notebook in a pipeline

[Continue >](#)

How are we doing? ★ ★ ★ ★ ★

Check your knowledge

1. What kind of pool is required to run a Synapse notebook in a pipeline? *

A Dedicated SQL pool

X Incorrect: A dedicated SQL pool is used to host a relational data warehouse.

A Data Explorer pool

An Apache Spark pool

✓ Correct: Notebooks are run in an Apache Spark pool.

2. What kind of pipeline activity encapsulates a Synapse notebook? *

Notebook activity

✓ Correct: Use a Notebook activity to run a Synapse notebook

HDInsight Spark activity

X Incorrect: An HDInsight Spark activity is used to run a Spark job on an Azure HDInsight cluster

Script activity

3. A notebook cell contains variable declarations. How can you use them as parameters? *

Add a %%Spark magic at the beginning of the cell

X Incorrect: a %%Spark magic would configure the cell to be run as Scala code.

Toggle the *Parameters cell* setting for the cell

✓ Correct: Toggle the Parameters cell option to use variables as parameters.

Use the var keyword for each variable declaration

Next unit: Summary

[Continue >](#)

100 XP

Introduction

1 minute

Hybrid Transactional / Analytical Processing (HTAP) is a style of data processing that combines transactional data processing, such as is typically found in a business application, with analytical processing, such as is used in a business intelligence (BI) or reporting solution. The data access patterns and storage optimizations used in these two kinds of workload are very different, so usually a complex extract, transform, and load (ETL) process is required to copy data out of transactional systems and into analytical systems; adding complexity and latency to data analysis. In an HTAP solution, the transactional data is replicated automatically, with low-latency, to an analytical store, where it can be queried without impacting the performance of the transactional system.

In Azure Synapse Analytics, HTAP capabilities are provided by multiple **Azure Synapse Link** services, each connecting a commonly used transactional data store to your Azure Synapse Analytics workspace and making the data available for processing using Spark or SQL.

After completing this module, you'll be able to:

- Describe Hybrid Transactional / Analytical Processing patterns.
- Identify Azure Synapse Link services for HTAP.

Next unit: Understand hybrid transactional and analytical processing patterns

[Continue >](#)

How are we doing?

Understand hybrid transactional and analytical processing patterns

6 minutes

Many business application architectures separate transactional and analytical processing into separate systems with data stored and processed on separate infrastructures. These infrastructures are commonly referred to as OLTP (online transaction processing) systems working with operational data, and OLAP (online analytical processing) systems working with historical data, with each system optimized for their specific task.

OLTP systems are optimized for dealing with discrete system or user requests immediately and responding as quickly as possible.

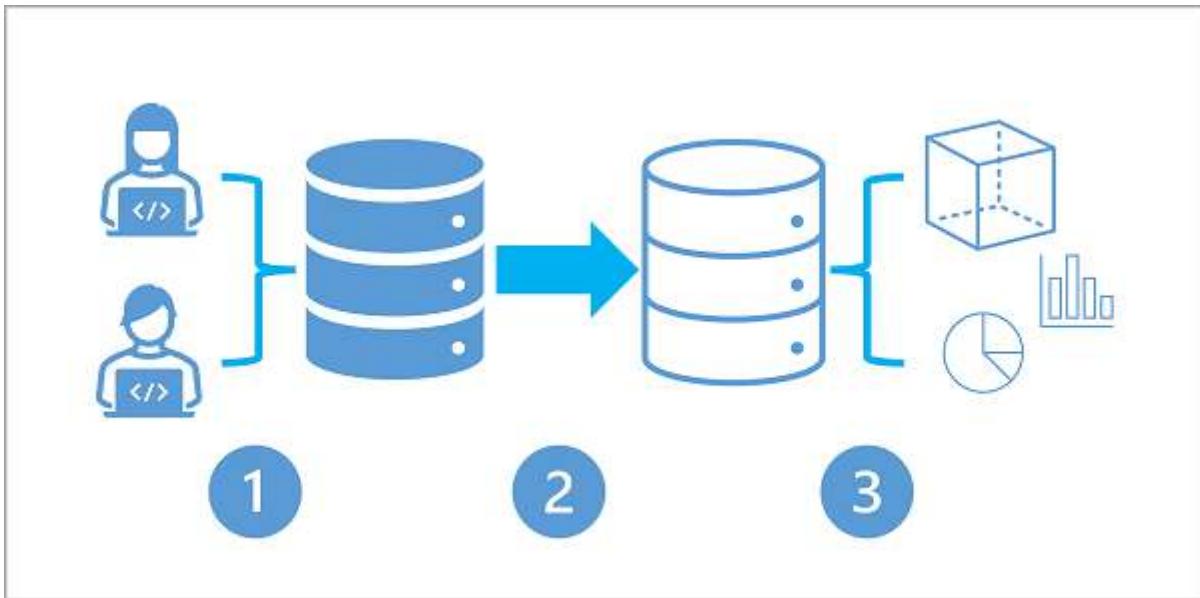
OLAP systems are optimized for the analytical processing, ingesting, synthesizing, and managing large sets of historical data. The data processed by OLAP systems largely originates from OLTP systems and needs to be loaded into the OLAP systems by ETL (Extract, Transform, and Load) batch processes.

Due to their complexity and the need to physically copy large amounts of data, this approach creates a delay in data being available to analyze in OLAP systems.

Hybrid Transactional / Analytical Processing (HTAP)

As more businesses move to digital processes, they increasingly recognize the value of being able to respond to opportunities by making faster and well-informed decisions. HTAP (Hybrid Transactional/Analytical processing) enables business to run advanced analytics in near-real-time on data stored and processed by OLTP systems.

The following diagram illustrates the generalized pattern of an HTAP architecture:



1. A business application processes user input and stores data in a transactional database that is optimized for a mix of data reads and writes based on the application's expected usage profile.
2. The application data is automatically replicated to an analytical store with low latency.
3. The analytical store supports data modeling, analytics, and reporting without impacting the transactional system.

Next unit: Describe Azure Synapse Link

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆



Describe Azure Synapse Link

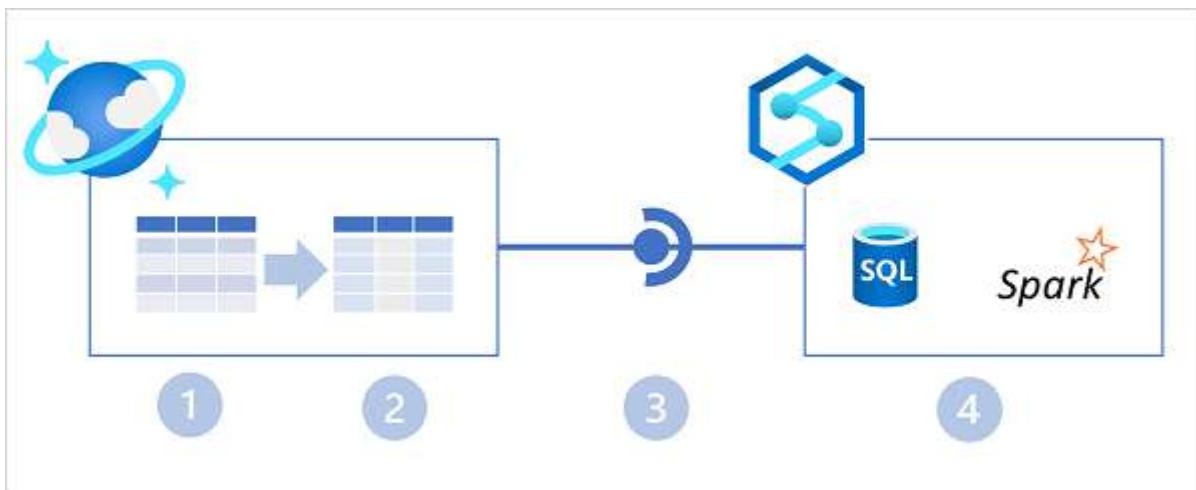
6 minutes

HTAP solutions are supported in Azure Synapse Analytics through **Azure Synapse Link**; a general term for a set of linked services that support HTAP data synchronization into your Azure Synapse Analytics workspace.

Azure Synapse Link for Cosmos DB

Azure Cosmos DB is a global-scale NoSQL data service in Microsoft Azure that enables applications to store and access operational data by using a choice of application programming interfaces (APIs).

Azure Synapse Link for Azure Cosmos DB is a cloud-native HTAP capability that enables you to run near-real-time analytics over operational data stored in a Cosmos DB container. Azure Synapse Link creates a tight seamless integration between Azure Cosmos DB and Azure Synapse Analytics.



In the diagram above, the following key features of the Azure Synapse Link for Cosmos DB architecture are illustrated:

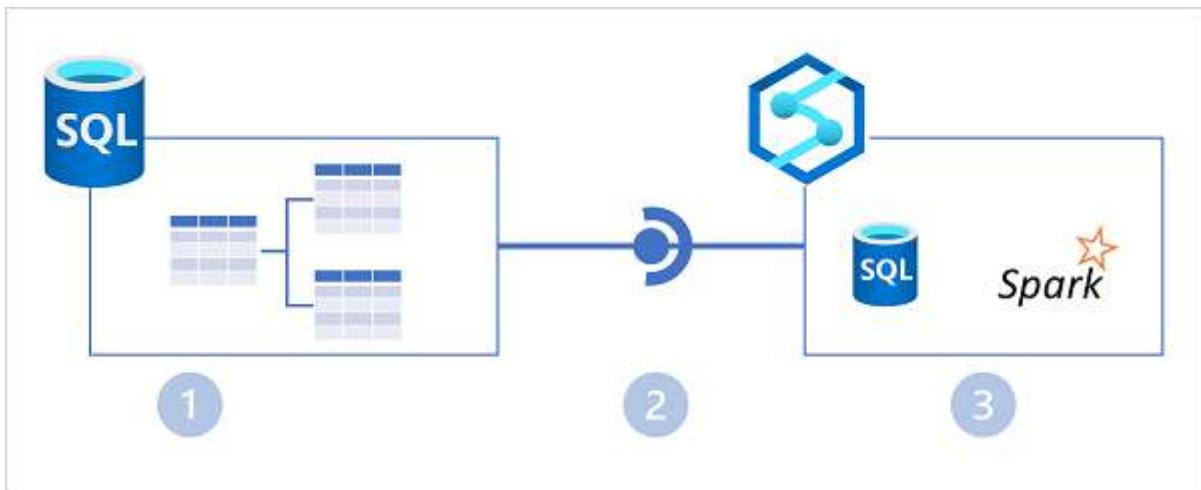
1. An Azure Cosmos DB container provides a row-based transactional store that is optimized for read/write operations.
2. The container also provides a column-based analytical store that is optimized for analytical workloads. A fully managed autosync process keeps the data stores in sync.
3. Azure Synapse Link provides a linked service that connects the analytical store enabled container in Azure Cosmos DB to an Azure Synapse Analytics workspace.

4. Azure Synapse Analytics provides Synapse SQL and Apache Spark runtimes in which you can run code to retrieve, process, and analyze data from the Azure Cosmos DB analytical store without impacting the transactional data store in Azure Cosmos DB.

Azure Synapse Link for SQL

Microsoft SQL Server is a popular relational database system that powers business applications in some of the world's largest organizations. Azure SQL Database is a cloud-based platform-as-a-service database solution based on SQL Server. Both of these relational database solutions are commonly used as operational data stores.

Azure Synapse Link for SQL enables HTAP integration between data in SQL Server or Azure SQL Database and an Azure Synapse Analytics workspace.



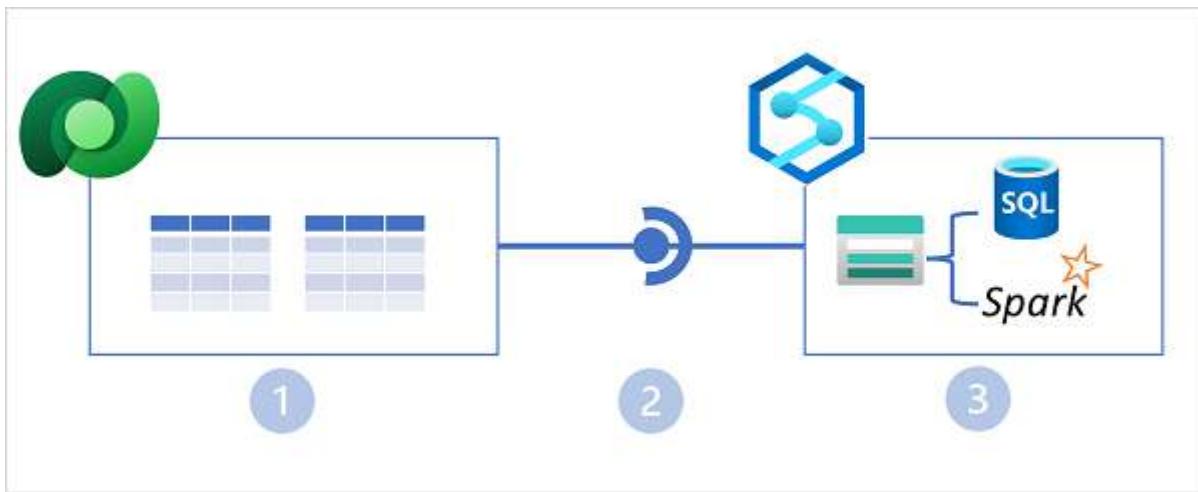
In the diagram above, the following key features of the Azure Synapse Link for SQL architecture are illustrated:

1. An Azure SQL Database or SQL Server instance contains a relational database in which transactional data is stored in tables.
2. Azure Synapse Link for SQL replicates the table data to a dedicated SQL pool in an Azure Synapse workspace.
3. The replicated data in the dedicated SQL pool can be queried in the dedicated SQL pool, or connected to as an external source from a Spark pool without impacting the source database.

Azure Synapse Link for Dataverse

Microsoft Dataverse is data storage service within the Microsoft Power Platform. You can use Dataverse to store business data in tables that are accessed by Power Apps, Power BI, Power Virtual Agents, and other applications and services across Microsoft 365, Dynamics 365, and Azure.

Azure Synapse Link for Dataverse enables HTAP integration by replicating table data to Azure Data Lake storage, where it can be accessed by runtimes in Azure Synapse Analytics - either directly from the data lake or through a Lake Database defined in a serverless SQL pool.



In the diagram above, the following key features of the Azure Synapse Link for Dataverse architecture are illustrated:

1. Business applications store data in Microsoft Dataverse tables.
2. Azure Synapse Link for Dataverse replicates the table data to an Azure Data Lake Gen2 storage account associated with an Azure Synapse workspace.
3. The data in the data lake can be used to define tables in a lake database and queried using a serverless SQL pool, or read directly from storage using SQL or Spark.

Next unit: Knowledge check

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 200 XP



Knowledge check

5 minutes

1. Which of the following descriptions matches a hybrid transactional/analytical processing (HTAP) architecture. *

- Business applications store data in an operational data store, which is also used to support analytical queries for reporting.
- Business applications store data in an operational data store, which is synchronized with low latency to a separate analytical store for reporting and analysis.

✓ Correct. an HTAP solution replicates operational data to an analytical store, enabling you to perform analytics and reporting without impacting the performance of the operational system.

- Business applications store operational data in an analytical data store that is optimized for queries to support reporting and analysis.

2. You want to use Azure Synapse Analytics to analyze operational data stored in a Cosmos DB for NoSQL container. Which Azure Synapse Link service should you use? *

- Azure Synapse Link for SQL
- Azure Synapse Link for Dataverse

✗ Incorrect. Azure Synapse Link for Dataverse integrates with the Dataverse Power Platform data store.

- Azure Synapse Link for Azure Cosmos DB

✓ Correct. Azure Synapse Link for Azure Cosmos DB integrates with multiple Azure Cosmos DB APIs, including Azure Cosmos DB for NoSQL.

3. You plan to use Azure Synapse Link for Dataverse to analyze business data in your Azure Synapse Analytics workspace. Where is the replicated data from Dataverse stored? *

- In an Azure Synapse dedicated SQL pool



In an Azure Data Lake Gen2 storage container.

✓ Correct. Azure Synapse Link for Dataverse replicates data to an Azure Data Lake Gen2 storage account.



In an Azure Cosmos DB container.

Next unit: Summary

[Continue >](#)

How are we doing?

Introduction

1 minute

Azure Synapse Analytics Link for Cosmos DB enables *hybrid transactional/analytical processing* (HTAP) integration between Azure Cosmos DB and Azure Synapse Analytics. By using this HTAP solution, organizations can make operational data in Azure Cosmos DB available for analysis and reporting in Azure Synapse Analytics in near-real time without the need to develop a complex ETL pipeline.

In this module, you'll learn how to:

- Configure an Azure Cosmos DB account to use Azure Synapse Link.
- Create an analytical store enabled container.
- Create a linked service for Azure Cosmos DB.
- Analyze linked data using Spark.
- Analyze linked data using Synapse SQL.

Next unit: Enable Cosmos DB account to use Azure Synapse Link

[Continue >](#)

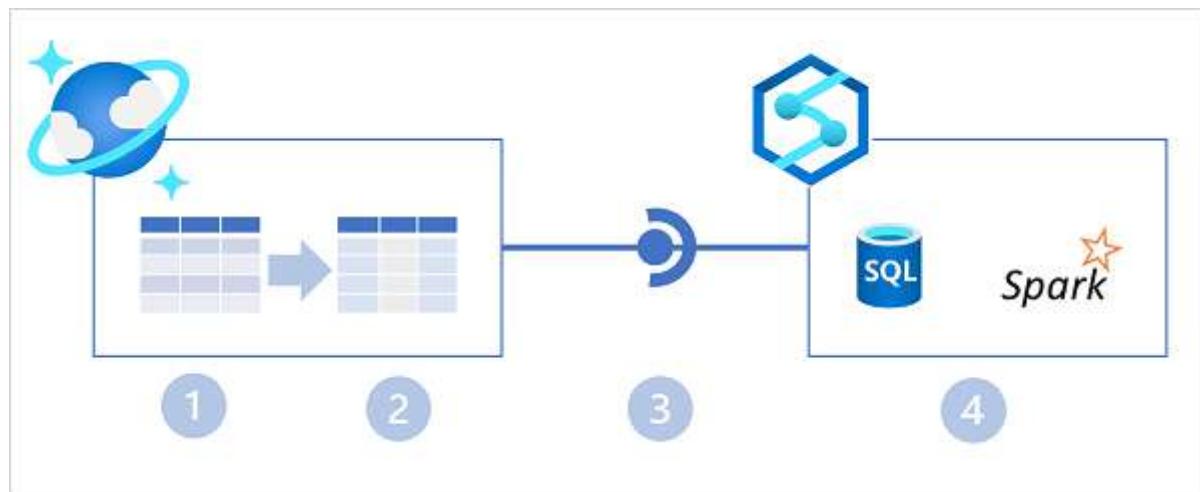
How are we doing?



Enable Cosmos DB account to use Azure Synapse Link

5 minutes

Azure Synapse Link for Azure Cosmos DB is a cloud-native HTAP capability that enables integration between Azure Cosmos DB and Azure Synapse Analytics.



In the diagram above, the following key features of the Azure Synapse Link for Cosmos DB architecture are illustrated:

1. An Azure Cosmos DB container provides a row-based transactional store that is optimized for read/write operations.
2. The container also provides a column-based analytical store that is optimized for analytical workloads. A fully managed autosync process keeps the data stores in sync.
3. Azure Synapse Link provides a linked service that connects the analytical store enabled container in Azure Cosmos DB to an Azure Synapse Analytics workspace.
4. Azure Synapse Analytics provides Synapse SQL and Apache Spark runtimes in which you can run code to retrieve, process, and analyze data from the Azure Cosmos DB analytical store without impacting the transactional data store in Azure Cosmos DB.

Enabling Azure Synapse Link in Azure Cosmos DB

The first step in using Azure Synapse Link for Cosmos DB is to enable it in an Azure Cosmos DB account. Azure Synapse Link is supported in the following types of Azure Cosmos DB account:

- Azure Cosmos DB for NoSQL
- Azure Cosmos DB for MongoDB

- Azure Cosmos DB for Apache Gremlin (*preview*)

You can enable Azure Synapse Link in the Azure portal page for your Cosmos DB account, or by using the Azure CLI or Azure PowerShell from a command line or in a script.

Using the Azure portal

In the Azure portal, you can enable Azure Synapse Link for a Cosmos DB account on the **Azure Synapse Link** page in the **Integrations** section, as shown below.

The screenshot shows the Azure portal interface for enabling Azure Synapse Link on a specific Cosmos DB account. The left sidebar lists various account settings like Replicate data globally, Default consistency, and CORS. The Integrations section is expanded, showing options for Power BI, Azure Synapse Link (which is selected and highlighted in grey), Add Azure Cognitive Search, and Add Azure Function. The main content area is titled 'Enable Azure Synapse Link' and contains a sub-section 'Enable Azure Synapse Link for your containers'. It provides instructions on how Synapse Link enables real-time analytics over operational data and allows selecting specific containers for analysis. A large blue 'Enable' button is prominently displayed. Navigation buttons 'Search', 'Home', 'Microsoft Azure', and a search bar are visible at the top.

Tip

For Azure Cosmos DB for NoSQL accounts, there's also a link on the **Data Explorer** page.

Using the Azure CLI

To enable Azure Synapse Link using the Azure CLI, run the `az cosmosdb create` command (to create a new Cosmos DB account) or `az cosmosdb update` command (to configure an existing

Cosmos DB account) with the `--enable-analytical-storage true` parameter. For example, the following command updates an existing Cosmos DB account named **my-cosmos-db** to enable Azure Synapse Link.

```
az cosmosdb update --name my-cosmos-db --resource-group my-rg --enable-analytical-storage true
```

To enable Azure Synapse Link for an Azure Cosmos DB for Apache Gremlin account, include the `--capabilities EnableGremlin` parameter.

Using Azure PowerShell

To enable Azure Synapse Link using Azure PowerShell, run the `New-AzCosmosDBAccount` cmdlet (to create a new Cosmos DB account) or `Update-AzCosmosDBAccount` cmdlet (to configure an existing Cosmos DB account) with the `-EnableAnalyticalStorage 1` parameter. For example, the following command updates an existing Cosmos DB account named **my-cosmos-db** to enable Azure Synapse Link.

```
Update-AzCosmosDBAccount -Name "my-cosmos-db" -ResourceGroupName "my-rg" -EnableAnalyticalStorage 1
```

Considerations for enabling Azure Synapse Link

When planning to enable Azure Synapse Link for a Cosmos DB account, consider the following facts:

- After enabling Azure Synapse Link for an account, you can't disable it.
- Enabling Azure Synapse Link doesn't start synchronization of operational data to an analytical store - you must also create or update a container with support for an analytical store.
- When enabling Azure Synapse Link for a Cosmos DB for NoSQL account using the Azure CLI or PowerShell, you can use the `--analytical-storage-schema-type` (Azure CLI) or `-AnalyticalStorageSchemaType` (PowerShell) parameter to specify the schema type as `WellDefined` (default) or `FullFidelity`. For a Cosmos DB for MongoDB account, the default (and only supported) schema type is `FullFidelity`.

- After a schema type has been assigned, you can't change it.

 Note

You'll learn more about the analytical store and its schema types in the next unit.

Next unit: Create an analytical store enabled container

[Continue >](#)

How are we doing?     

✓ 100 XP



Create an analytical store enabled container

5 minutes

After enabling Azure Synapse Link in an Azure Cosmos DB account, you can create or update a container with support for an analytical store.

An analytical store is a column-based store within the same container as a row-based operational store. An *auto-sync* process synchronizes changes in the operational store to the analytical store; from where it can be queried without incurring processing overhead in the operational store.

Analytical store schema types

As the data from the operational store is synchronized to the analytical store, the schema is updated dynamically to reflect the structure of the documents being synchronized. The specific behavior of this dynamic schema maintenance depends on the analytical store schema type configured for the Azure Cosmos DB account. Two types of schema representation are supported:

- **Well-defined:** The default schema type for an Azure Cosmos DB for NoSQL account.
- **Full fidelity:** The default (and only supported) schema type for an Azure Cosmos DB for MongoDB account.

The analytical store receives JSON data from the operational store and organizes it into a column-based structure. In a well-defined schema, the first non-null occurrence of a JSON field determines the data type for that field. Subsequent occurrences of the field that aren't compatible with the assigned data type aren't ingested into the analytical store.

For example, consider the following two JSON documents:

JSON

```
{"productID": 123, "productName": "Widget"}  
{"productID": "124", "productName": "Wotsit"}
```

The first document determines that the `productID` field is a numeric (integer) value. When the second document is encountered, its `productID` field has a string value, and so isn't imported

into the analytical store. The document and the rest of its field is imported, but the incompatible field is dropped. The following columns represent the data in the analytical store:

productID	productName
123	Widget
	Wotsit

In a full fidelity schema, the data type is appended to each instance of the field, with new columns created as necessary; enabling the analytical store to contain multiple occurrences of a field, each with a different data type, as shown in the following table:

productID.int32	productName.string	productID.string
123	Widget	
	Wotsit	124

 **Note**

For more information, see [What is Azure Cosmos DB analytical store?](#).

Enabling analytical store support in a container

You can enable analytical store support when creating a new container or for an existing container. To enable analytical store support, you can use the Azure portal, or you can use the Azure CLI or Azure PowerShell from a command line or in a script.

Using the Azure portal

To enable analytical store support when creating a new container in the Azure portal, select the **On** option for **Analytical Store**, as shown here:

The screenshot shows the Microsoft Azure Data Explorer interface for the 'graeme-cosmos-db' account. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, etc. The main area is titled 'NOSQL API' and shows sections for DATA and NOTEBOOKS. A modal window titled 'New Container' is open, prompting for a 'Container id' (set to 'my-container'), 'Indexing' (set to 'Automatic'), and a 'Partition key' (set to '/productID'). The 'Analytical store' section, which includes a radio button for 'On' or 'Off', is highlighted with a red box. At the bottom right of the modal is a blue 'OK' button.

Alternatively, you can enable analytical store support for an existing container in the **Azure Synapse Link** page in the **Integrations** section of the page for your Cosmos DB account, as shown here:

The screenshot shows the Microsoft Azure portal interface for managing an Azure Cosmos DB account named 'graeme-cosmos-db'. On the left, there's a sidebar with various options like Dedicated Gateway, Keys, Advisor Recommendations, Microsoft Defender for Cloud, Identity, Locks, Power BI, Add Azure Cognitive Search, Add Azure Function, Browse, Scale, Settings, Document Explorer, Query Explorer, and Script Explorer. The 'Integrations' section is expanded, and 'Azure Synapse Link' is selected. The main content area has a heading 'Enable Azure Synapse Link' with a status message 'Account enabled' and a green checkmark. Below it, another section titled 'Enable Azure Synapse Link for your containers' allows selecting specific containers. Two containers, 'my-db' and 'my-container', are checked. At the bottom right of this section is a blue button labeled 'Enable Synapse Link on your container(s)'. The URL in the browser bar is https://portal.azure.com/.

Using the Azure CLI

To use the Azure CLI to enable analytical store support in an Azure Cosmos DB for NoSQL container, run the `az cosmosdb sql container create` command (to create a new container) or `az cosmosdb sql container update` command (to configure an existing container) with the `--analytical-storage-ttl` parameter, assigning a retention time for analytical data. Specifying an `-analytical-storage-ttl` parameter of `-1` enables permanent retention of analytical data. For example, the following command creates a new container named **my-container** with analytical store support.

```
az cosmosdb sql container create --resource-group my-rg --account-name my-cosmos-db --database-name my-db --name my-container --partition-key-path "/productID" --analytical-storage-ttl -1
```

For an Azure Cosmos DB for MongoDB account, you can use the `az cosmosdb mongodb collection create` or `az cosmosdb mongodb collection update` command with the `--analytical-storage-ttl` parameter. For an Azure Cosmos DB for Apache Gremlin account,

use the `az cosmosdb gremlin graph create` or `az cosmosdb gremlin graph update` command with the `--analytical-storage-ttl` parameter.

Using Azure PowerShell

To use Azure PowerShell to enable analytical store support in an Azure Cosmos DB for NoSQL container, run the `New-AzCosmosDBSqlContainer` cmdlet (to create a new container) or `Update-AzCosmosDBSqlContainer` cmdlet (to configure an existing container) with the `-AnalyticalStorageTtl` parameter, assigning a retention time for analytical data. Specifying an `-AnalyticalStorageTtl` parameter of `-1` enables permanent retention of analytical data. For example, the following command creates a new container named `my-container` with analytical store support.

```
New-AzCosmosDBSqlContainer -ResourceGroupName "my-rg" -AccountName "my-cosmos-db" -DatabaseName "my-db" -Name "my-container" -PartitionKeyKind "hash" -PartitionKeyPath "/productID" -AnalyticalStorageTtl -1
```

For an Azure Cosmos DB for MongoDB API account, use the `New-AzCosmosDBMongoDBCcollection` or `Update-AzCosmosDBMongoDBCcollection` cmdlet with the `-AnalyticalStorageTtl` parameter.

Considerations for enabling analytical store support

Analytical store support can't be disabled without deleting the container. Setting the analytical store TTL value to `0` or `null` effectively disables the analytical store by no longer synchronizing new items to it from the operational store and deleting items already synchronized from the analytical store. After setting this value to `0`, you can't re-enable analytical store support in the container.

Next unit: Create a linked service for Cosmos DB

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



Create a linked service for Cosmos DB

5 minutes

When you have an Azure Cosmos DB container with analytical store support, you can create a linked service in an Azure Synapse Analytics workspace to connect to it.

To create a linked service to an Azure Cosmos DB analytical data store, use Azure Synapse Studio, and add a linked service on the **Data** page by selecting the **Connect to external data** option, as shown here:

The screenshot shows the Azure Synapse Analytics Data page. On the left, there's a navigation sidebar with icons for Home, Data (which is selected), Develop, Integrate, Monitor, and Manage. The main area has a title 'Connect to external data' with a sub-instruction: 'Once a connection is created, the underlying data of that connection will be available for analysis in the Data hub or for pipeline activities in the Integrate hub.' Below this, there are five options arranged in a grid: 'Azure Blob Storage', 'Azure Cosmos DB for MongoDB' (which is highlighted with a blue border), 'Azure Cosmos DB for NoSQL', 'Azure Data Explorer (Kusto)', and 'Azure Data Lake Storage Gen2'. At the bottom of the dialog are 'Continue' and 'Cancel' buttons.

As you complete the steps to create your linked service, select the type of Azure Cosmos DB account and then assign your linked service a meaningful name and provide the necessary information to connect to your Azure Cosmos DB database.

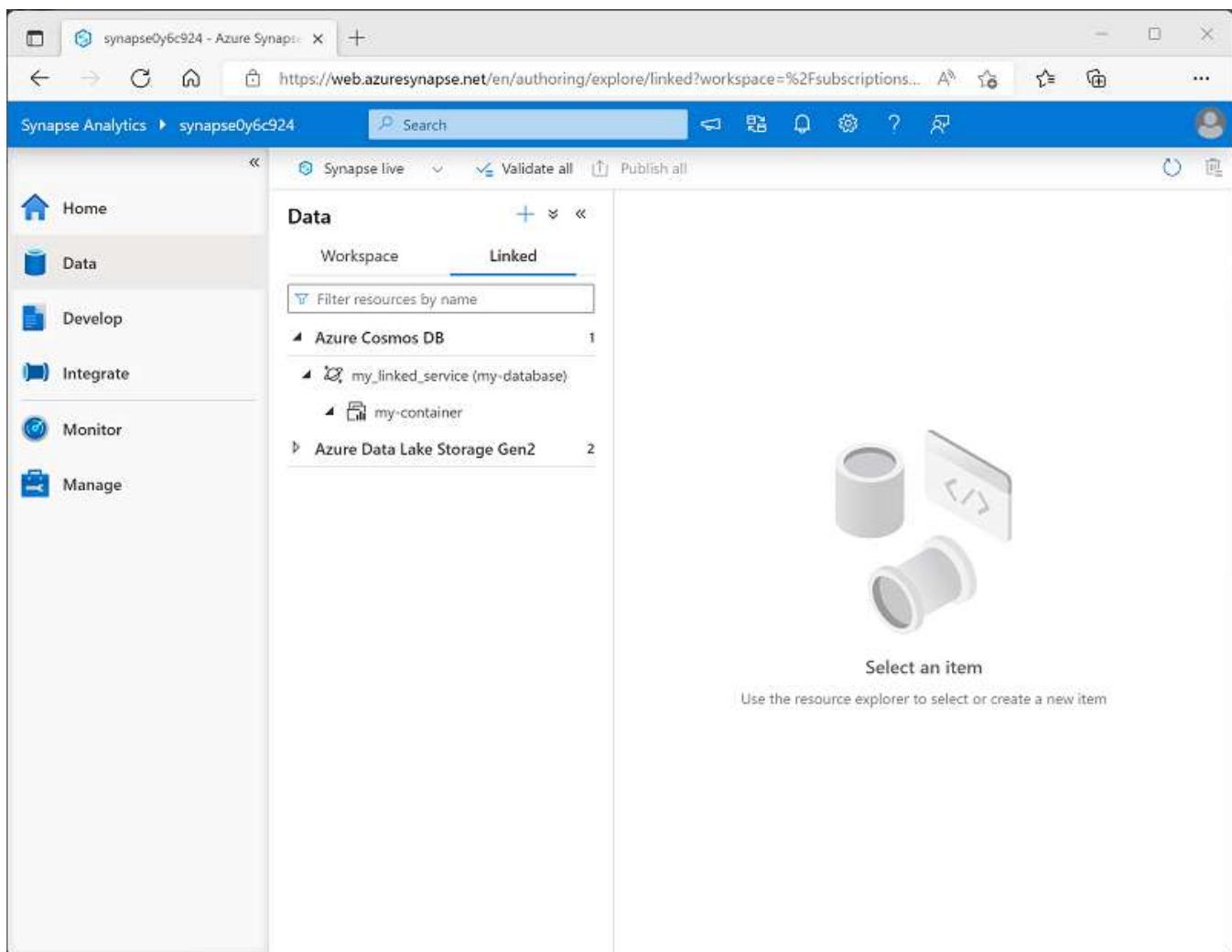
To connect to the Azure Cosmos DB database, you can use any of the following authentication options:

- **Account key:** Specify an authentication key for your Cosmos DB account.
- **Service Principal:** Use the identity of the Azure Synapse Analytics service.
- **System Assigned Managed Identity:** Use system-assigned managed identity.
- **User Managed Identity:** Use a user-defined managed identity.

💡 Tip

For more information about using managed identities in Azure Active Directory, see [What are managed identities for Azure resources?](#)

After creating a linked service, the Azure Cosmos DB database and its containers will be shown in the **Data** page of Azure Synapse Studio, as shown here:



❗ Note

The user interface differentiates between containers with analytical store support and those without by using the following icons:

Analytical store enabled



Analytical store not enabled



You can query a container without an analytical store, but you won't benefit from the advantages of an HTAP solution that offloads analytical query overhead from the operational data store.

Next unit: Query Cosmos DB data with Spark

[Continue >](#)

How are we doing?

✓ 100 XP



Query Cosmos DB data with Spark

5 minutes

After you've added a linked service for your analytical store enabled Azure Cosmos DB database, you can use it to query the data using a Spark pool in your Azure Synapse Analytics workspace.

Loading Azure Cosmos DB analytical data into a dataframe

For initial exploration or quick analysis of data from an Azure Cosmos DB linked service, it's often easiest to load data from a container into a dataframe using a Spark-supported language like PySpark (A Spark-specific implementation of Python) or Scala (a Java-based language often used on Spark).

For example, the following PySpark code could be used to load a dataframe named `df` from the data in the `my-container` container connected to using the `my_linked_service` linked service, and display the first 10 rows of data:

Python

```
df = spark.read  
    .format("cosmos.olap")\  
    .option("spark.synapse.linkedService", "my_linked_service")\  
    .option("spark.cosmos.container", "my-container")\  
    .load()  
  
display(df.limit(10))
```

Let's suppose the `my-container` container is used to store items similar to the following example:

JSON

```
{  
  "productID": 123,  
  "productName": "Widget",  
  "id": "7248f072-11c3-42b1-a368-...",  
  "_rid": "mjMaAL...==",  
  "_self": "dbs/mjM...==/colls/mjMaAL...=/docs/mjMaAL...==/",  
  "_etag": "\"54004b09-0000-2300-...\"",
```

```
        "_attachments": "attachments/",
        "_ts": 1655414791
    }
```

The output from the PySpark code would be similar to the following table:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-421a-a0f7...	5400ca09-0000-2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5...	5400ca09-0000-2300-...
...

The data is loaded from the analytical store in the container, not from the operational store; ensuring that there's no querying overhead on the operational store. The fields in the analytical data store include the application-defined fields (in this case **productID** and **productName**) and automatically created metadata fields.

After loading the dataframe, you can use its native methods to explore the data. For example, the following code creates a new dataframe containing only the **productID** and **productName** columns, ordered by the **productName**:

Python

```
products_df = df.select("productID", "productName").orderBy("productName")
display(products_df.limit(10))
```

The output of this code would look similar this table:

productID	productName
125	Thingumy
123	Widget
124	Wotsit
...	...

Writing a dataframe to a Cosmos DB container

In most HTAP scenarios, you should use the linked service to read data into Spark from the analytical store. However you *can* write the contents of a dataframe to the container as shown in the following example:

Python

```
mydf.write.format("cosmos.oltp")\
    .option("spark.synapse.linkedService", "my_linked_service")\
    .option("spark.cosmos.container", "my-container")\
    .mode('append')\
    .save()
```

! Note

Writing a dataframe to a container updates the *operational* store and can have an impact on its performance. The changes are then synchronized to the analytical store.

Using Spark SQL to query Azure Cosmos DB analytical data

Spark SQL is a Spark API that provides SQL language syntax and relational database semantics in a Spark pool. You can use Spark SQL to define metadata for tables that can be queried using SQL.

For example, the following code creates a table named **Products** based on the hypothetical container used in the previous examples:

SQL

```
%%sql

-- Create a logical database in the Spark metastore
CREATE DATABASE mydb;

USE mydb;

-- Create a table from the Cosmos DB container
CREATE TABLE products using cosmos.olap options (
    spark.synapse.linkedService 'my_linked_service',
    spark.cosmos.container 'my-container'
);

-- Query the table
SELECT productID, productName
FROM products;
```

💡 Tip

The `%%sql` keyword at the beginning of the code is a *magic* that instructs the Spark pool to run the code as SQL rather than the default language (which is usually set to PySpark).

By using this approach, you can create a logical database in your Spark pool that you can then use to query the analytical data in Azure Cosmos DB to support data analysis and reporting workloads without impacting the operational store in your Azure Cosmos DB account.

Next unit: Query Cosmos DB with Synapse SQL

[Continue >](#)

How are we doing? ★ ★ ★ ★ ★

✓ 100 XP



Query Cosmos DB with Synapse SQL

5 minutes

In addition to using a Spark pool, you can also query an Azure Cosmos DB analytical container by using a built-in *serverless* SQL pool in Azure Synapse Analytics. To do this, you can use the OPENROWSET SQL function to connect to the linked service for your Azure Cosmos DB database.

Using OPENROWSET with an authentication key

By default, access to an Azure Cosmos DB account is authenticated by an authentication key. You can use this key as part of a connection string in an OPENROWSET statement to connect through a linked service from a SQL pool, as shown in the following example:

SQL

```
SELECT *
FROM OPENROWSET(
    'CosmosDB',
    'Account=my-cosmos-db;Database=my-db;Key=abcd1234....==',
    [my-container]) AS products_data
```

💡 Tip

You can find a primary and secondary key for your Cosmos DB account on its **Keys** page in the Azure portal.

The results of this query might look something like the following, including metadata and application-defined fields from the items in the Azure Cosmos DB container:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-	5400ca09-0000-

_rid	_ts	productID	productName	id	_etag
				421a-a0f7-...	2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5-...	5400ca09-0000-2300-...
...

The data is retrieved from the analytical store, and the query doesn't impact the operational store.

Using OPENROWSET with a credential

Instead of including the authentication key in each call to OPENROWSET, you can define a *credential* that encapsulates the authentication information for your Cosmos DB account, and use the credential in subsequent queries. To create a credential, use the CREATE CREDENTIAL statement as shown in this example:

SQL

```
CREATE CREDENTIAL my_credential
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'abcd1234....==';
```

With the credential in place, you can use it in an OPENROWSET function like this:

SQL

```
SELECT *
FROM OPENROWSET(PROPRIETOR = 'CosmosDB',
                 CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                 OBJECT = 'my-container',
                 SERVER_CREDENTIAL = 'my_credential'
) AS products_data
```

Once again, the results include metadata and application-defined fields from the analytical store:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-421a-a0f7...	5400ca09-0000-2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5...	5400ca09-0000-2300-...
...

Specifying a schema

The `OPENROWSET` syntax includes a `WITH` clause that you can use to define a schema for the resulting rowset. You can use this to specify individual fields and assign data types as shown in the following example:

SQL

```
SELECT *
FROM OPENROWSET(PROVIDER = 'CosmosDB',
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                OBJECT = 'my-container',
                SERVER_CREDENTIAL = 'my_credential'
)
WITH (
    productID INT,
    productName VARCHAR(20)
) AS products_data
```

In this case, assuming the fields in the analytical store include `productID` and `productName`, the resulting rowset will resemble the following table:

productID	productName
123	Widget
124	Wotsit
125	Thingumy
...	...

You can of course specify individual column names in the `SELECT` clause (for example, `SELECT productID, productName ...`), so this ability to specify individual columns may seem of limited use. However, consider cases where the source JSON documents stored in the operational store include multiple levels of fields, as show in the following example:

JSON
<pre>{ "productID": 126, "productName": "Sprocket", "supplier": { "supplierName": "Contoso", "supplierPhone": "555-123-4567" } "id": "62588f072-11c3-42b1-a738-...", "_rid": "mjMaAL...==", ... }</pre>

The `WITH` clause supports the inclusion of explicit JSON paths, enabling you to handle nested fields and to assign aliases to field names; as shown in this example:

SQL
<pre>SELECT * FROM OPENROWSET(PROVIDER = 'CosmosDB', CONNECTION = 'Account=my-cosmos-db;Database=my-db', OBJECT = 'my-container', SERVER_CREDENTIAL = 'my_credential') WITH (ProductNo INT '\$.productID', ProductName VARCHAR(20) '\$.productName', Supplier VARCHAR(20) '\$.supplier.supplierName',</pre>

```
SupplierPhoneNo VARCHAR(15) '$.supplier.supplierPhone'  
 ) AS products_data
```

The results of this query would include the following row for product 126:

ProductNo	ProductName	Supplier	SupplierPhoneNo
126	Sprocket	Contoso	555-123-4567

Creating a view in a database

If you need to query the same data frequently, or you need to use reporting and visualization tools that rely on SELECT statements that don't include the OPENROWSET function, you can use a *view* to abstract the data. To create a view, you should create a new database in which to define it (user-defined views in the **master** database aren't supported), as shown in the following example:

SQL

```
CREATE DATABASE sales_db  
    COLLATE Latin1_General_100_BIN2_UTF8;  
GO;  
  
USE sales_db;  
GO;  
  
CREATE VIEW products  
AS  
SELECT *  
FROM OPENROWSET(PROPRIER = 'CosmosDB',  
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',  
                OBJECT = 'my-container',  
                SERVER_CREDENTIAL = 'my_credential'  
)  
WITH (  
    ProductNo INT '$.productID',  
    ProductName VARCHAR(20) '$.productName',  
    Supplier VARCHAR(20) '$.supplier.supplierName',  
    SupplierPhoneNo VARCHAR(15) '$.supplier.supplierPhone'  
) AS products_data  
GO
```

 Tip

When creating a database that will access data in Cosmos DB, it's best to use a UTF-8 based collation to ensure compatibility with strings in Cosmos DB.

After the view has been created, users and client applications can query it like any other SQL view or table:

SQL

```
SELECT * FROM products;
```

Considerations for Serverless SQL pools and Azure Cosmos DB

When planning to use a serverless SQL pool to query data in an Azure Cosmos DB analytical store, consider the following best practices:

- Provision your Azure Cosmos DB analytical storage and any client applications (for example Microsoft Power BI) in the same region as serverless SQL pool.

Azure Cosmos DB containers can be replicated to multiple regions. If you have a multi-region container, you can specify a region parameter in the OPENROWSET connection string to ensure queries are sent to a specific regional replica of the container.

- When working with string columns, use the OPENROWSET function with the explicit WITH clause and specify an appropriate data length for the string data.

Next unit: Exercise - Implement Azure Synapse Link for Cosmos DB

[Continue >](#)

How are we doing?

✓ 200 XP

Knowledge check

5 minutes

1. You have an Azure Cosmos DB for NoSQL account and an Azure Synapse Analytics workspace. What must you do first to enable HTAP integration with Azure Synapse Analytics? *

Configure global replication in Azure Cosmos DB.

X Incorrect. Global replication helps scale Cosmos DB, but it's not required for HTAP integration with Azure Synapse Analytics.

Create a dedicated SQL pool in Azure Synapse Analytics.

Enable Azure Synapse Link in Azure Cosmos DB.

✓ Correct. The first step in setting up HTAP integration is to enable Azure Synapse Link in Azure Cosmos DB.

2. You have an existing container in a Cosmos DB core (SQL) database. What must you do to enable analytical queries over Azure Synapse Link from Azure Synapse Analytics? *

Delete and recreate the container.

Enable Azure Synapse Link in the container to create an analytical store.

✓ Correct. Before a container can be used for analytical queries, you need to enable Synapse link for the container; which creates an analytical store.

Add an item to the container.

X Incorrect. Adding an item to a container does not enable Synapse Link integration.

3. You plan to use a Spark pool in Azure Synapse Analytics to query an existing analytical store in Azure Cosmos DB. What must you do? *

Create a linked service for the Azure Cosmos DB database where the analytical store enabled container is defined.

✓ Correct. A linked service that connects to the Azure Cosmos DB account containing the analytical store enabled container is required.

- Disable automatic pausing for the Spark pool in Azure Synapse Analytics.
- Install the Azure Cosmos DB SDK for Python package in the Spark pool.

✗ Incorrect. The Azure Cosmos DB SDK for Python is not required for Synapse Link integration.

4. You're writing PySpark code to load data from an Azure Cosmos DB analytical store into a dataframe. What **format** should you specify? *

- `cosmos.json`
- `cosmos.olap`

✓ Correct. `cosmos.olap` is the appropriate format to read data from a Cosmos DB analytical store.

- `cosmos.sql`

✗ Incorrect. `cosmos.sql` isn't a valid format.

5. You're writing a SQL code in a serverless SQL pool to query an analytical store in Azure Cosmos DB. What function should you use? *

- `OPENDATASET`
- `ROW`

✗ Incorrect. While `ROW` is a valid SQL function, it isn't used to query external data.

- `OPENROWSET`

✓ Correct. `OPENROWSET` is used to query external data, including analytical stores in Cosmos DB.

Next unit: Summary

[Continue >](#)

100 XP

Introduction

1 minute

Azure Synapse Link for SQL is a *hybrid transactional / analytical processing* (HTAP) capability in Azure Synapse Analytics that you can use to synchronize transactional data in Azure SQL Database or Microsoft SQL Server with a dedicated SQL pool in Azure Synapse Analytics. This synchronization enables you to perform near real-time analytical workloads on operational data with minimal impact on the transactional store used by business applications.

In this module, you'll learn how to:

- Understand key concepts and capabilities of Azure Synapse Link for SQL.
- Configure Azure Synapse Link for Azure SQL Database.
- Configure Azure Synapse Link for Microsoft SQL Server.

Next unit: What is Azure Synapse Link for SQL?

[Continue >](#)

How are we doing?



What is Azure Synapse Link for SQL?

5 minutes

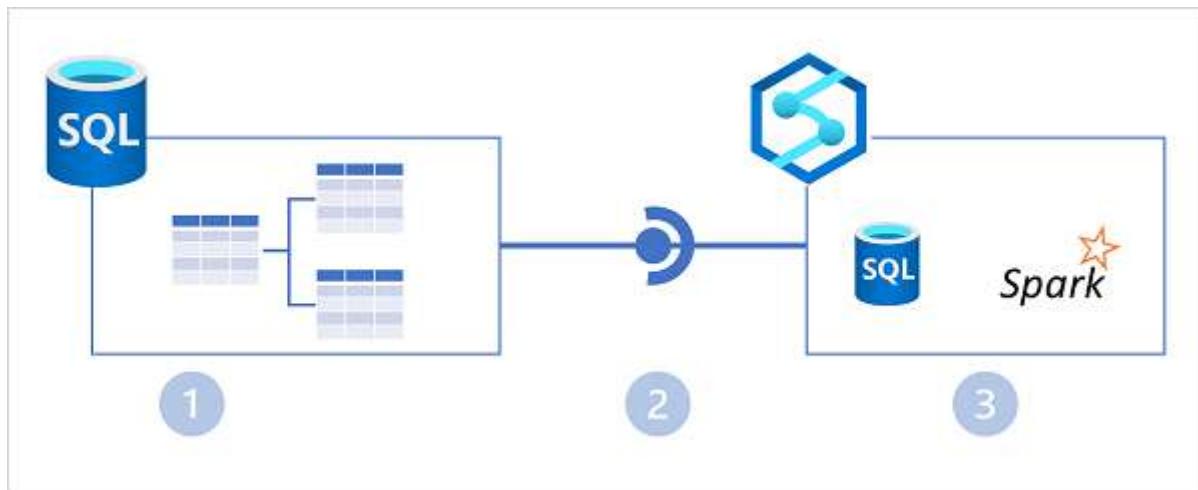
Many organizations use a relational database in Azure SQL Database or Microsoft SQL Server to support business applications. These databases are optimized for transactional workloads that store and manipulate operational data. Performing analytical queries on the data in these databases to support reporting and data analysis incurs resource contention that can be detrimental to application performance.

A traditional approach to resolving this problem is to implement an *extract, transform, and load* (ETL) solution that loads data from the operational data store into an analytical store as a batch operation at regular intervals. While this solution supports the analytical workloads required for reporting and data analysis, it suffers from the following limitations:

- The ETL process can be complex to implement and operate.
- The analytical store is only updated at periodic intervals, so reporting doesn't reflect the most up-to-date operational data.

Azure Synapse Link for SQL

Azure Synapse Link for SQL addresses the limitations of a traditional ETL process by automatically replicating changes made to tables in the operational database to corresponding tables in an analytical database. After the initial synchronization process, the changes are replicated in near real-time without the need for a complex ETL batch process.



In the diagram above, the following key features of the Azure Synapse Link for SQL architecture are illustrated:

1. An Azure SQL Database or SQL Server 2022 instance contains a relational database in which transactional data is stored in tables.
2. Azure Synapse Link for SQL replicates the table data to a dedicated SQL pool in an Azure Synapse workspace.
3. The replicated data in the dedicated SQL pool can be queried in the dedicated SQL pool, or connected to as an external source from a Spark pool without impacting the source database.

Source and target databases

Azure Synapse Link for SQL supports the following source databases (used as operational data stores):

- Azure SQL Database
- Microsoft SQL Server 2022

 **Note**

Azure Synapse link for SQL is not supported for Azure SQL Managed Instance.

The target database (used as an analytical data store) must be a dedicated SQL pool in an Azure Synapse Analytics workspace.

The implementation details for Azure Synapse Link vary between the two types of data source, but the high-level principle is the same - changes made to tables in the source database are synchronized to the target database.

Change feed

Azure Synapse Link for SQL uses the *change feed* feature in Azure SQL Database and Microsoft SQL Server 2022 to capture changes to the source tables. All data modifications are recorded in the transaction log for the source database. The change feed feature monitors the log and applies the same data modifications in the target database. In the case of Azure SQL Database, the modifications are made directly to the target database. When using Azure Synapse Link for SQL Server, the changes are recorded in files and saved to a *landing zone* in Azure Data Lake Gen2 storage before being applied to the target database.

 **Note**

Change feed is similar to the *change data capture* (CDC) feature in SQL Server. The key difference is that CDC is used to reproduce data modifications in a table in the same

database as the modified table. Change feed caches the data modification in memory and forwards it to Azure Synapse Analytics.

After implementing Azure Synapse Link for SQL, you can use system views and stored procedures in your Azure SQL Database or SQL Server database to monitor and manage change feed activity.

 **Tip**

Learn more:

- For more information about change feed, see [Azure Synapse Link for SQL change feed](#).
- To learn more about monitoring and managing change feed, see [Manage Azure Synapse Link for SQL Server and Azure SQL Database](#).

Next unit: Configure Azure Synapse Link for Azure SQL Database

[Continue >](#)

How are we doing?     

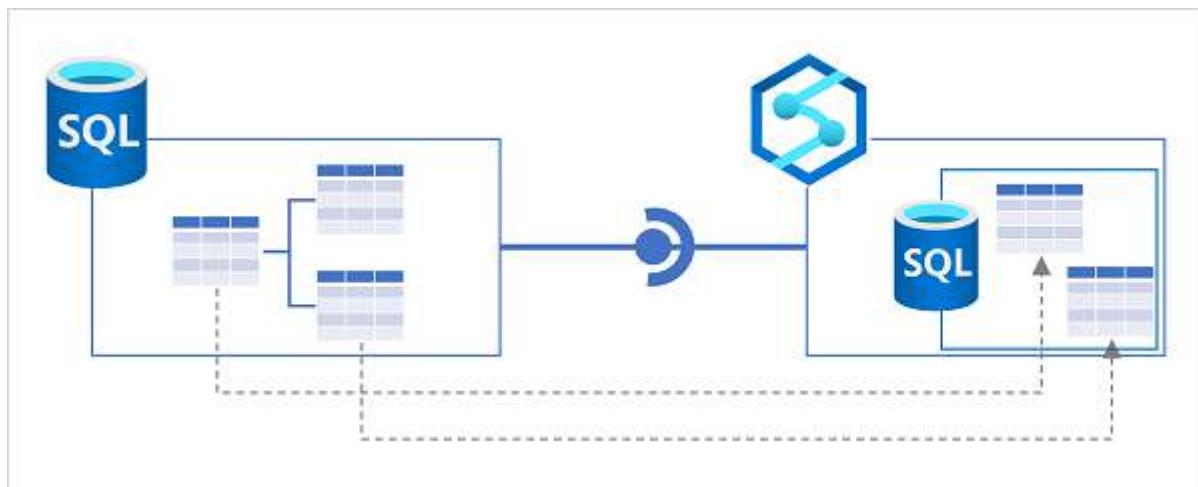
✓ 100 XP

Configure Azure Synapse Link for Azure SQL Database

5 minutes

Azure SQL Database is a platform-as-a-service (PaaS) relational database service based on the SQL Server database engine. It's commonly used in cloud-native applications as a scalable, secure, and easy to manage relational database store for operational data.

Azure Synapse Link for Azure SQL Database uses a *link connection* to map one or more tables in an Azure SQL Database instance to tables in a dedicated SQL pool in Azure Synapse Analytics. When the link connection is started, the tables are initialized by copying a snapshot of the source tables to the target tables. Subsequently, the change feed process applies all modifications made in the source tables to the target tables.



Implementing Azure Synapse Link for Azure SQL Database

To use Azure Synapse Link for Azure SQL Database, you need to configure some settings in your Azure SQL Database server, before creating a link connection in Azure Synapse Analytics.

Configure Azure SQL Database

Before you can use Azure SQL Database as a source for a linked connection in Azure Synapse Analytics, you must ensure the following settings are configured in the Azure SQL Database server that hosts the database you want to synchronize:

- **System assigned managed identity** - enable this option so that your Azure SQL Database server uses a system assigned managed identity.
- **Firewall rules** - ensure that Azure services can access your Azure SQL Database server.

In addition to these server-level settings, if you plan to configure the link connection from Azure Synapse Analytics to use a managed identity when connecting to Azure SQL Database, you must create a user for the workspace identity in the database and add it to the **db_owner** role, as shown in the following code example:

SQL

```
CREATE USER my_synapse_workspace FROM EXTERNAL PROVIDER;
ALTER ROLE [db_owner] ADD MEMBER my_synapse_workspace;
```



If you intend to use SQL authentication, you can omit this step.

Prepare the target SQL pool

Azure Synapse Link for Azure SQL Database synchronizes the source data to tables in a dedicated SQL pool in Azure Synapse Analytics. You therefore need to create and start a dedicated SQL pool in your Azure Synapse Analytics workspace before you can create the link connection.

The database associated with the dedicated SQL pool must include the appropriate schema for the target table. If source tables are defined in a schema other than the default **dbo** schema, you must create a schema of the same name in the dedicated SQL pool database:

SQL

```
CREATE SCHEMA myschema;
```

Create a link connection

To create a linked connection, add a **linked connection** on the **Integrate** page in Azure Synapse Studio. You'll need to:

1. Select or create a *linked service* for your Azure SQL Database. You can create this separately ahead of time, or as part of the process of creating a linked connection for

Azure Synapse Link. You can use a managed identity or SQL authentication to connect the linked service to Azure SQL Database.

2. Select the tables in the source database that you want to include in the linked connection.
3. Select the target dedicated SQL pool in which the target tables should be created.
4. Specify the number of CPU cores you want to use to process synchronization. Four driver cores will be used in addition to the number of cores you specify.

After creating the linked connection, you can configure the mappings between the source and target tables. In particular, you can specify the table structure (index) type and distribution configuration for the target tables.

 **Note**

Some data types in your source tables may not be supported by specific dedicated SQL pool index types. For example, you cannot use a clustered columnstore index for tables that include VARBINARY(MAX) columns. You can map such tables to a *heap* (an unindexed table) in the dedicated SQL pool.

When the linked connection is configured appropriately, you can start it to initialize synchronization. The source tables are initially copied to the target database as snapshots, and then subsequent data modifications are replicated.

 **Tip**

Learn more:

- For more information about Synapse Link for Azure SQL Database, see [Azure Synapse Link for Azure SQL Database](#).
- To learn about limitations and restrictions that apply to Synapse Link for Azure SQL Database, see [Known limitations and issues with Azure Synapse Link for SQL](#).
- For a step-by-step guide to setting up Synapse Link for Azure SQL Database, see [Get started with Azure Synapse Link for Azure SQL Database](#). You'll also get a chance to try configuring Synapse Link for Azure SQL Database in the exercise, later in this module.

Next unit: Configure Azure Synapse Link for SQL Server 2022

✓ 100 XP

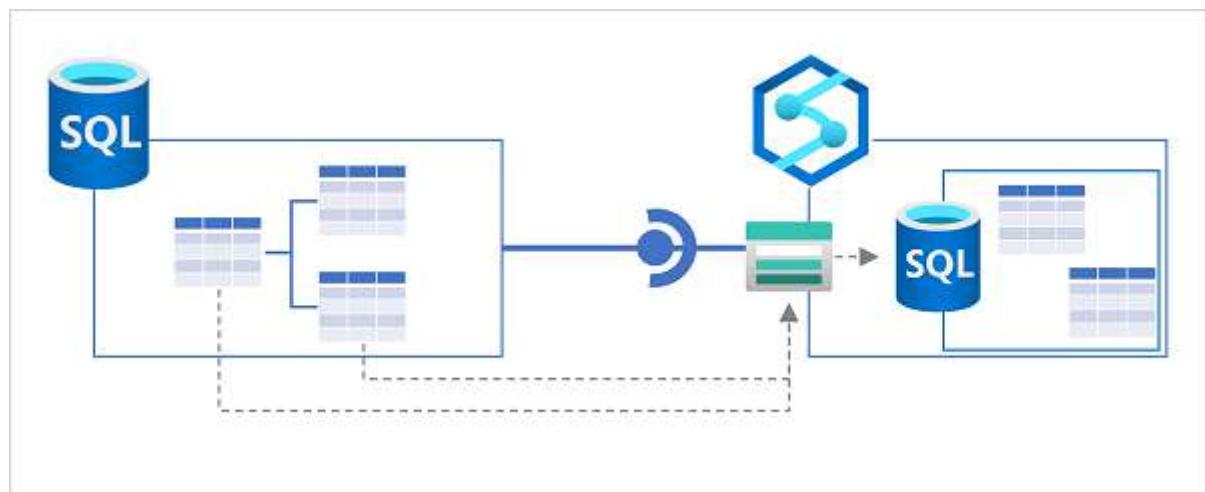


Configure Azure Synapse Link for SQL Server 2022

5 minutes

Microsoft SQL Server is one of the world's most commonly used relational database systems. SQL Server 2022 is the latest release, and includes many enhancements and new features; including the ability to be used as a source for Azure Synapse Link.

Azure Synapse Link for SQL Server uses a *link connection* to map one or more tables in an Azure SQL Database instance to tables in a dedicated SQL pool in Azure Synapse Analytics. When the link connection is started, the tables are initialized by copying a .parquet file for each source table to a *landing zone* in Azure Data Lake Storage Gen2; from where the data is imported into tables in the dedicated SQL pool. Subsequently, the change feed process copies all changes as .csv files to the landing zone where they're applied to the target tables.



Synchronization between SQL Server (which can be on-premises or in a private network) and Azure Synapse Analytics is achieved through a self-hosted integration runtime. An integration runtime is a software agent that handles secure connectivity when using Azure Data Factory or Azure Synapse Analytics to transfer data across networks. It must be installed on a Microsoft Windows computer with direct access to your SQL Server instance.

Tip

For more information about using a self-hosted integration runtime to work with Azure Synapse Analytics, see [Create and configure a self-hosted integration runtime](#).

Implementing Azure Synapse Link for SQL Server 2022

To use Azure Synapse Link for SQL Server 2022, you need to create storage for the landing zone in Azure and configure your SQL Server instance before creating a link connection in Azure Synapse Analytics.

Create landing zone storage

You need to create an Azure Data Lake Storage Gen2 account in your Azure subscription to use as a landing zone. You can't use the default storage for your Azure Synapse Analytics workspace.

Tip

For more information about provisioning an Azure Data Lake Storage Gen2 account, see [Create a storage account to use with Azure Data Lake Storage Gen2](#).

Create a master key in the SQL Server database

To support Azure Synapse Link, your SQL Server database must contain a master key. You can use a CREATE MASTER KEY SQL statement like the following example to create one:

SQL

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'my$ecretPa$$w0rd';
```

Create a dedicated SQL pool in Azure Synapse Analytics

In your Azure Synapse Analytics workspace, you need to create a dedicated SQL pool where the target tables will be created. You also need to create master key in this database by using the following SQL statement:

SQL

```
CREATE MASTER KEY
```

Create a linked service for the SQL Server source database

Next, in Azure Synapse Analytics, create a linked service for your SQL Server database. When you do this, you need to specify the self-hosted integration runtime to be used for connectivity between SQL Server and Azure Synapse Analytics. If you haven't already configured a self-hosted integration runtime, you can create one now, and then download and install the agent onto a Windows machine in the network where your SQL Server instance is located.

Create a linked service for your Data Lake Storage Gen2 account

In addition to the linked service for SQL Server, you need a linked service for the Data Lake Storage Gen2 account that will be used as a landing zone. To support this, you need to add the managed identity of your Azure Synapse Analytics Workspace to the **Storage Blob Data Contributor** role for your storage account and configure the linked service to use the managed identity for authentication.

Create a link connection for Azure Synapse Link

Finally, you're ready to create a link connection for Azure Synapse Link data synchronization. As you do so, you'll specify the service link for the SQL Server source database, the individual tables to be replicated, the number of CPU cores to be used for the synchronization process, and the Azure Data Lake Storage Gen2 linked service and folder location for the landing zone.

After the link connection is created, you can start it to initialize synchronization. After a short time, the tables will be available to query in the dedicated SQL pool, and will be kept in sync with modifications in the source database by the change feed process.

💡 Tip

Learn more:

- For more information about Synapse Link for SQL Server 2022, see [Azure Synapse Link for SQL Server 2022](#).
- To learn about limitations and restrictions that apply to Synapse Link for Azure SQL Database, see [Known limitations and issues with Azure Synapse Link for SQL](#).
- For a step-by-step guide to setting up Synapse Link for SQL Server 2022, see [Get started with Azure Synapse Link for SQL Server 2022](#).

Next unit: Exercise - Implement Azure Synapse Link for SQL

✓ 200 XP



Knowledge check

3 minutes

1. From which of the following data sources can you use Azure Synapse Link for SQL to replicate data to Azure Synapse Analytics? *

Azure Cosmos DB

✗ Incorrect. Azure Synapse Link does not support Cosmos DB. You can use Azure Synapse Link for Cosmos DB instead.

SQL Server 2022

✓ Correct. You can use Azure Synapse Link for SQL to replicate data from SQL Server 2022.

Azure SQL Managed Instance

2. What must you create in your Azure Synapse Analytics workspace as a target database for Azure Synapse Link for Azure SQL Database? *

A serverless SQL pool

An Apache Spark pool

✗ Incorrect. You cannot use a Spark pool for Azure Synapse Link.

A dedicated SQL pool

✓ Correct. Use a dedicated SQL pool as a target database.

3. You plan to use Azure Synapse Link for SQL to replicate tables from SQL Server 2022 to Azure Synapse Analytics. What additional Azure resource must you create? *

An Azure Storage account with an Azure Data Lake Storage Gen2 container

✓ Correct. An Azure Data Lake Storage Gen2 account is required to be used as a landing zone when using Azure Synapse Link for SQL Server 2022.

An Azure Key Vault containing the SQL Server admin password

X Incorrect. You do not need Azure key Vault to use Azure Synapse Link for SQL.

- An Azure Application Insights resource
-

Next unit: Summary

[Continue >](#)

How are we doing?     

100 XP

Introduction

1 minute

Today, massive amounts of real-time data are generated by connected applications, Internet of Things (IoT) devices and sensors, and various other sources. The proliferation of streaming data sources has made the ability to consume and make informed decisions from these data in near-real-time an operational necessity for many organizations.

Some typical examples of streaming data workloads include:

- Online stores analyzing real-time clickstream data to provide product recommendations to consumers as they browse the website.
- Manufacturing facilities using telemetry data from IoT sensors to remotely monitor high-value assets.
- Credit card transactions from point-of-sale systems being scrutinized in real-time to detect and prevent potentially fraudulent activities.

Azure Stream Analytics provides a cloud-based stream processing engine that you can use to filter, aggregate, and otherwise process a real-time stream of data from various sources. The results of this processing can then be used to trigger automated activity by a service or application, generate real-time visualizations, or integrate streaming data into an enterprise analytics solution.

In this module, you'll learn how to get started with Azure Stream Analytics, and use it to process a stream of event data.

Next unit: Understand data streams

[Continue >](#)

How are we doing?

Understand data streams

5 minutes

A data stream consists of a perpetual series of data, typically related to specific point-in-time events. For example, a stream of data might contain details of messages submitted to a social media micro-blogging site, or a series of environmental measurements recorded by an internet-connected weather sensor. Streaming data analytics is most often used to better understand change over time. For example, a marketing organization may perform sentiment analysis on social media messages to see if an advertising campaign results in more positive comments about the company or its products, or an agricultural business might monitor trends in temperature and rainfall to optimize irrigation and crop harvesting.

Common goals for stream analytics include

- Continuously analyzing data to report issues or trends.
- Understanding component or system behavior under various conditions to help plan future enhancements.
- Triggering specific actions or alerts when certain events occur or thresholds are exceeded.

Characteristics of stream processing solutions

Stream processing solutions typically exhibit the following characteristics:



1. The source data stream is *unbounded* - data is added to the stream perpetually.
2. Each data record in the stream includes *temporal* (time-based) data indicating when the event to which the record relates occurred (or was recorded).
3. Aggregation of streaming data is performed over temporal *windows* - for example, recording the number of social media posts per minute or the average rainfall per hour.
4. The results of streaming data processing can be used to support real-time (or *near real-time*) automation or visualization, or persisted in an analytical store to be combined with other data.

other data for historical analysis. Many solutions combine these approaches to support both real-time and historical analytics.

Next unit: Understand event processing

[Continue >](#)

How are we doing? 

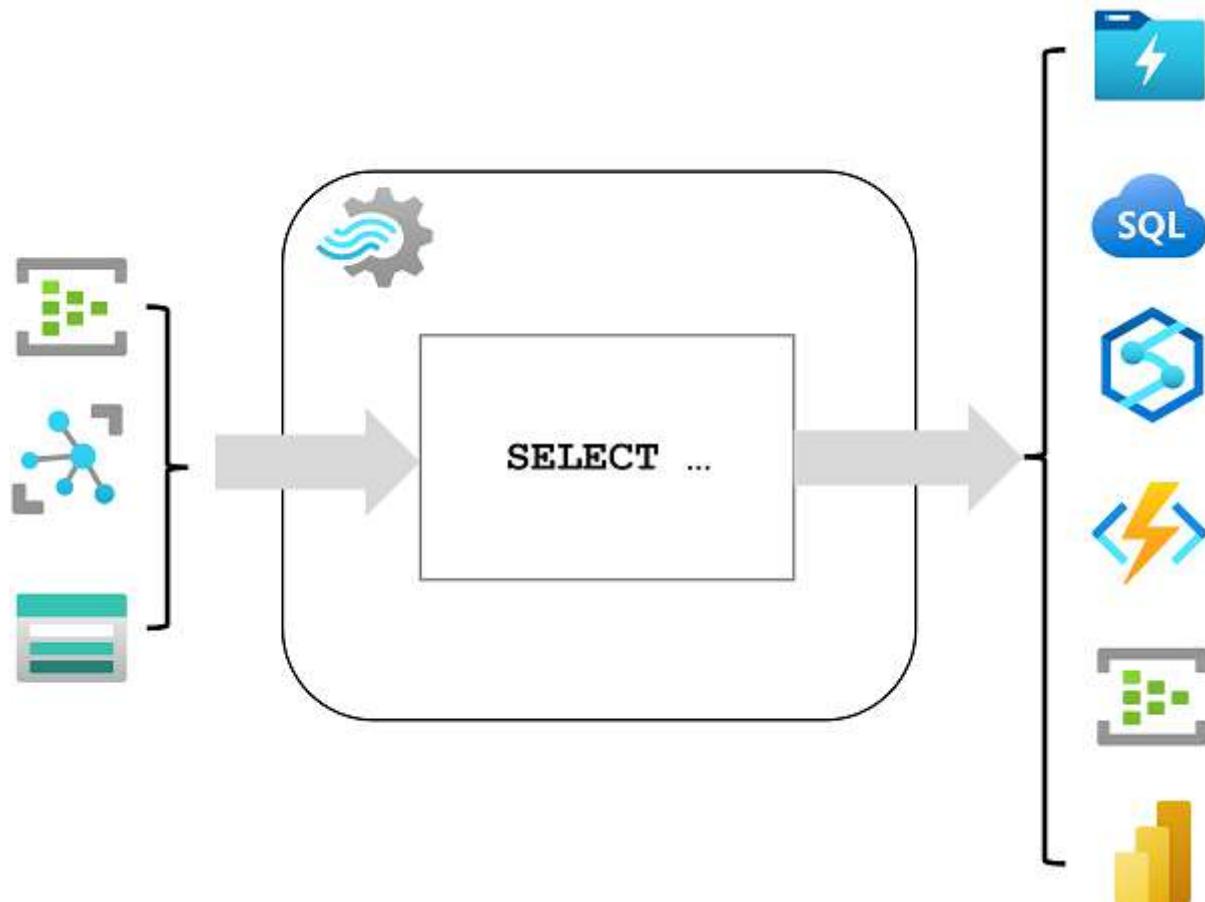
✓ 100 XP

Understand event processing

5 minutes

Azure Stream Analytics is a service for complex event processing and analysis of streaming data. Stream Analytics is used to:

- Ingest data from an *input*, such as an Azure event hub, Azure IoT Hub, or Azure Storage blob container.
- Process the data by using a *query* to select, project, and aggregate data values.
- Write the results to an *output*, such as Azure Data Lake Gen 2, Azure SQL Database, Azure Synapse Analytics, Azure Functions, Azure event hub, Microsoft Power BI, or others.



Once started, a Stream Analytics query will run perpetually, processing new data as it arrives in the input and storing results in the output.

Stream Analytics guarantees *exactly once* event processing and *at-least-once* event delivery, so events are never lost. It has built-in recovery capabilities in case the delivery of an event fails. Also, Stream Analytics provides built-in checkpointing to maintain the state of your job and produces repeatable results. Because Azure Stream Analytics is a platform-as-a-service (PaaS) solution, it's fully managed and highly reliable. Its built-in integration with various sources and

destinations and provides a flexible programmability model. The Stream Analytics engine enables in-memory compute, so it offers high performance.

Azure Stream Analytics jobs and clusters

The easiest way to use Azure Stream Analytics is to create a Stream Analytics *job* in an Azure subscription, configure its input(s) and output(s), and define the query that the job will use to process the data. The query is expressed using structured query language (SQL) syntax, and can incorporate static reference data from multiple data sources to supply lookup values that can be combined with the streaming data ingested from an input.

If your stream process requirements are complex or resource-intensive, you can create a Stream Analytics *cluster*, which uses the same underlying processing engine as a Stream Analytics job, but in a dedicated tenant (so your processing is not affected by other customers) and with configurable scalability that enables you to define the right balance of throughput and cost for your specific scenario.

Inputs

Azure Stream Analytics can ingest data from the following kinds of input:

- Azure Event Hubs
- Azure IoT Hub
- Azure Blob storage
- Azure Data Lake Storage Gen2

Inputs are generally used to reference a source of streaming data, which is processed as new event records are added. Additionally, you can define *reference* inputs that are used to ingest static data to augment the real-time event stream data. For example, you could ingest a stream of real-time weather observation data that includes a unique ID for each weather station, and augment that data with a static reference input that matches the weather station ID to a more meaningful name.

Outputs

Outputs are destinations to which the results of stream processing are sent. Azure Stream Analytics supports a wide range of outputs, which can be used to:

- Persist the results of stream processing for further analysis; for example by loading them into a data lake or data warehouse.

- Display a real-time visualization of the data stream; for example by appending data to a dataset in Microsoft Power BI.
- Generate filtered or summarized events for downstream processing; for example by writing the results of stream processing to an event hub.

Queries

The stream processing logic is encapsulated in a query. Queries are defined using SQL statements that *SELECT* data fields *FROM* one or more inputs, filter or aggregate the data, and write the results *INTO* an output. For example, the following query filters the events from the **weather-events** input to include only data from events with a **temperature** value less than 0, and writes the results to the **cold-temps** output:

SQL

```
SELECT observation_time, weather_station, temperature  
INTO cold-temps  
FROM weather-events TIMESTAMP BY observation_time  
WHERE temperature < 0
```

A field named **EventProcessedUtcTime** is automatically created to define the time when the event is processed by your Azure Stream Analytics query. You can use this field to determine the timestamp of the event, or you can explicitly specify another DateTime field by using the *TIMESTAMP BY* clause, as shown in this example. Depending on the input from which the streaming data is read, one or more potential timestamp fields may be created automatically; for example, when using an *Event Hubs* input, a field named **EventQueuedUtcTime** is generated to record the time when the event was received in the event hub queue.

The field used as a timestamp is important when aggregating data over temporal windows, which is discussed next.

Next unit: Understand window functions

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



Understand window functions

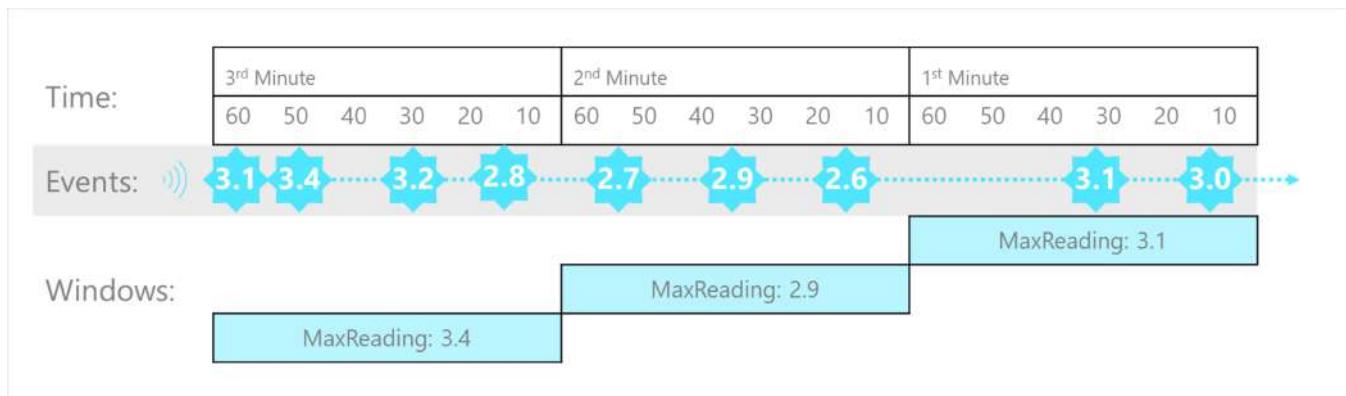
6 minutes

A common goal of stream processing is to aggregate events into temporal intervals, or *windows*. For example, to count the number of social media posts per minute or to calculate the average rainfall per hour.

Azure Stream Analytics includes native support for [five kinds of temporal windowing functions](#). These functions enable you to define temporal intervals into which data is aggregated in a query. The supported windowing functions are [Tumbling](#), [Hopping](#), [Sliding](#), [Session](#), and [Snapshot](#).

Tumbling

Tumbling window functions segment a data stream into a contiguous series of fixed-size, non-overlapping time segments and operate against them. Events can't belong to more than one tumbling window.



The Tumbling window example, represented by the following query, finds the maximum reading value in each one-minute window. Windowing functions are applied in Stream Analytics jobs using the `GROUP BY` clause of the query syntax. The `GROUP BY` clause in the following query contains the `TumblingWindow()` function, which specifies a one-minute window size.

SQL

```
SELECT DateAdd(minute,-1,System.TimeStamp) AS WindowStart,  
       System.TimeStamp() AS WindowEnd,  
       MAX(Reading) AS MaxReading  
INTO
```

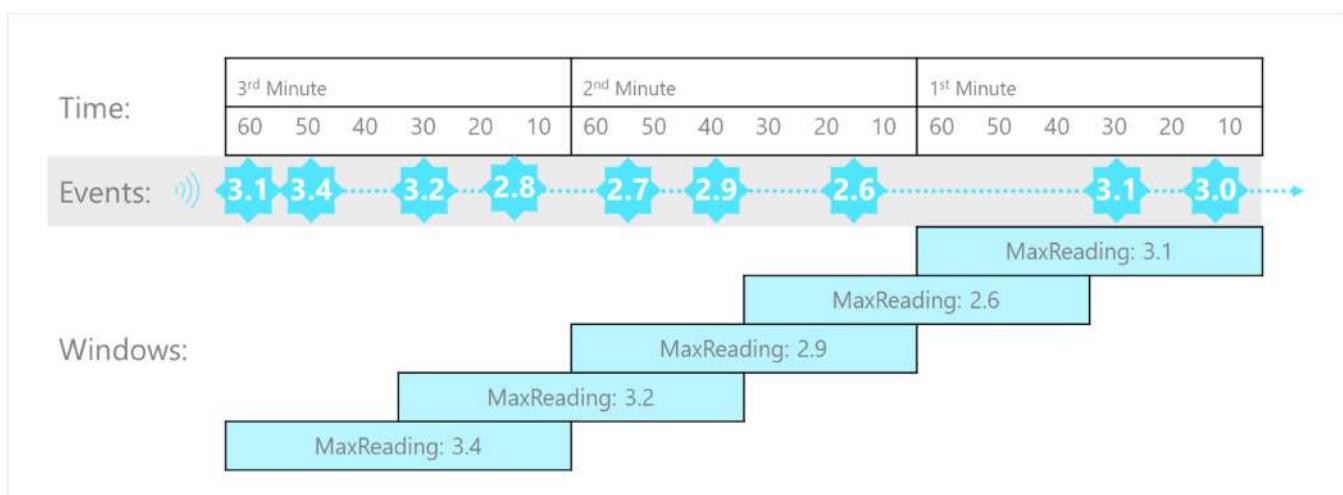
```

[output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY TumblingWindow(minute, 1)

```

Hopping

Hopping window functions model scheduled overlapping windows, jumping forward in time by a fixed period. It's easiest to think of them as Tumbling windows that can overlap and be emitted more frequently than the window size. In fact, tumbling windows are simply a hopping window whose `hop` is equal to its `size`. When you use Hopping windows, events can belong to more than one window result set.



To create a hopping window, you must specify three parameters. The first parameter indicates the time unit, such as second, minute, or hour. The following parameter sets the window size, which designates how long each window lasts. The final required parameter is the hop size, which specifies how much each window moves forward relative to the previous one. An optional fourth parameter denoting the offset size may also be used.

The following query demonstrates using a `HoppingWindow()` where the `timeunit` is set to `second`. The `windowsize` is 60 seconds, and the `hopsize` is 30 seconds. This query outputs an event every 30 seconds containing the maximum reading value that occurred over the last 60 seconds.

SQL

```

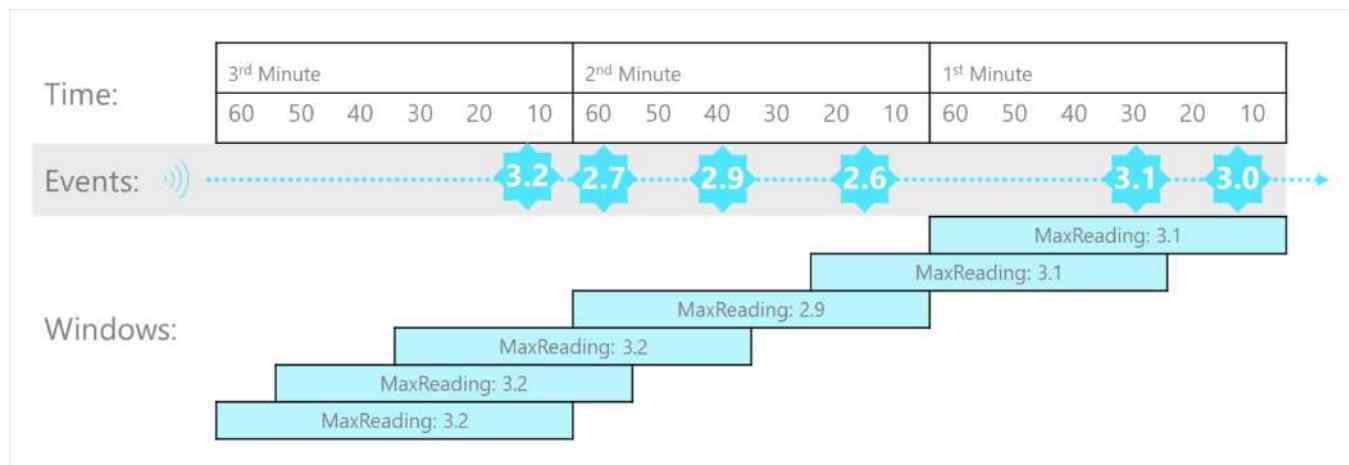
SELECT DateAdd(second,-60,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime

```

```
GROUP BY HoppingWindow(second, 60, 30)
```

Sliding

Sliding windows generate events for points in time when the content of the window actually changes. This function model limits the number of windows that need to be considered. Azure Stream Analytics outputs events for only those points in time when an event entered or exited the window. As such, every window contains a minimum of one event. Events in Sliding windows can belong to more than one sliding window, similar to Hopping windows.



The following query uses the `SlidingWindow()` function to find the maximum reading value in each one-minute window in which an event occurred.

SQL

```
SELECT DateAdd(minute,-1,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY SlidingWindow(minute, 1)
```

Session

Session window functions cluster together events that arrive at similar times, filtering out periods of time where there's no data. It has three primary parameters: timeout, maximum duration, and partitioning key (optional).



The occurrence of the first event starts a session window. Suppose another event occurs within the specified timeout from the last ingested event. In that case, the window will be extended to incorporate the new event. However, if no other events occur within the specified timeout period, the window will be closed at the timeout. If events keep happening within the specified timeout, the session window will extend until the maximum duration is reached.

The following query measures user session length by creating a `SessionWindow` over clickstream data with a `timeoutsize` of 20 seconds and a `maximumdurationsize` of 60 seconds.

SQL

```
SELECT DateAdd(second,-60,System.Timestamp) AS WindowStart,
       System.Timestamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY SessionWindow(second, 20, 60)
```

Snapshot

Snapshot windows groups events by identical timestamp values. Unlike other windowing types, a specific window function isn't required. You can employ a snapshot window by specifying the `System.Timestamp()` function to your query's `GROUP BY` clause.



For example, the following query finds the maximum reading value for events that occur at precisely the same time.

SQL

```
SELECT System.TimeStamp() AS WindowTime,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY System.Timestamp()
```

`System.Timestamp()` is considered in the `GROUP BY` clause as a snapshot window definition because it groups events into a window based on the equality of timestamps.

Next unit: Exercise - Get started with Azure Stream Analytics

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 200 XP ➔

Knowledge check

3 minutes

Check your knowledge

1. Which definition of stream processing is correct? *

- Data is processed continually as new data records arrive.

✓ Correct. Stream processing is used to continually process new data as it arrives.

- Data is collected in a temporary store, and all records are processed together as a batch.
- Data that is incomplete or contains errors is redirected to separate storage for correction by a human operator.

2. You need to process a stream of sensor data, aggregating values over one minute windows and storing the results in a data lake. Which service should you use? *

- Azure SQL Database

- Azure Cosmos DB

- Azure Stream Analytics

✓ Correct. Azure Stream Analytics is a stream processing engine.

3. You want to aggregate event data by contiguous, fixed-length, non-overlapping temporal intervals. What kind of window should you use? *

- Sliding

✗ Incorrect. Sliding windows are not contiguous.

- Session

- Tumbling

- ✓ Correct. Tumbling windows define contiguous, fixed-length, non-overlapping windows.
-

Next unit: Summary

[Continue >](#)

How are we doing?

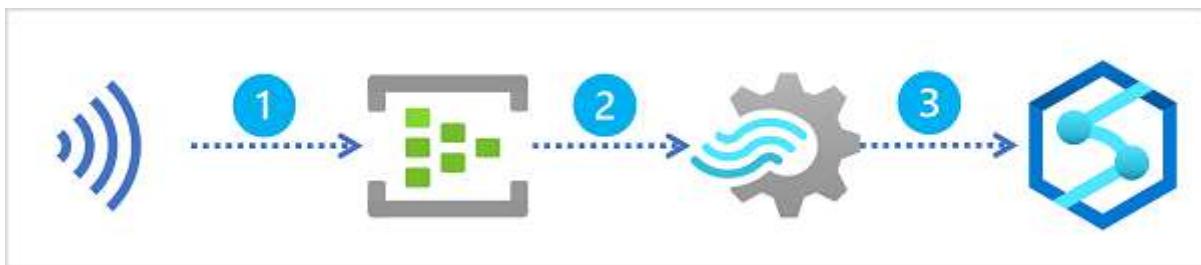
✓ 100 XP ➔

Introduction

1 minute

Suppose a retail company captures real-time sales transaction data from an e-commerce website, and wants to analyze this data along with more static data related to products, customers, and employees. A common way to approach this problem is to ingest the stream of real-time data into a data lake or data warehouse, where it can be queried together with data that is loaded using batch processing techniques.

Microsoft Azure Synapse Analytics provides a comprehensive enterprise data analytics platform, into which real-time data captured in Azure Event Hubs or Azure IoT Hub, and processed by Azure Stream Analytics can be loaded.



A typical pattern for real-time data ingestion in Azure consists of the following sequence of service integrations:

1. A real-time source of data is captured in an event ingestor, such as Azure Event Hubs or Azure IoT Hub.
2. The captured data is perpetually filtered and aggregated by an Azure Stream Analytics query.
3. The results of the query are loaded into a data lake or data warehouse in Azure Synapse Analytics for subsequent analysis.

In this module, you'll explore multiple ways in which you can use Azure Stream Analytics to ingest real-time data into Azure Synapse Analytics.

Next unit: Stream ingestion scenarios

[Continue >](#)



Stream ingestion scenarios

5 minutes

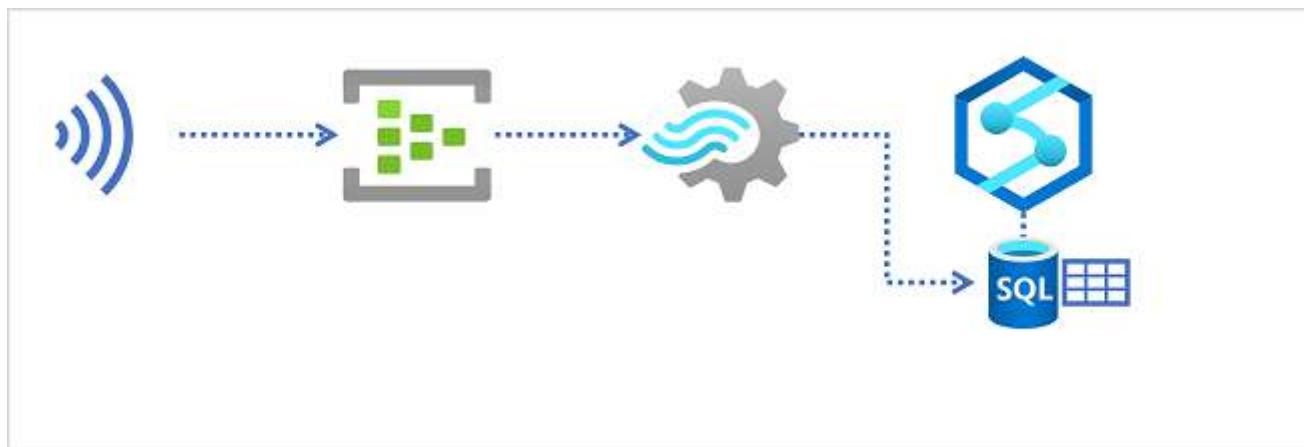
Azure Synapse Analytics provides multiple ways to analyze large volumes of data. Two of the most common approaches to large-scale data analytics are:

- **Data warehouses** - relational databases, optimized for distributed storage and query processing. Data is stored in tables and queried using SQL.
- **Data lakes** - distributed file storage in which data is stored as files that can be processed and queried using multiple runtimes, including Apache Spark and SQL.

Data warehouses in Azure Synapse Analytics

Azure Synapse Analytics provides dedicated SQL pools that you can use to implement enterprise-scale relational data warehouses. Dedicated SQL pools are based on a *massively parallel processing* (MPP) instance of the Microsoft SQL Server relational database engine in which data is stored and queried in tables.

To ingest real-time data into a relational data warehouse, your Azure Stream Analytics query must write its results to an output that references the table into which you want to load the data.

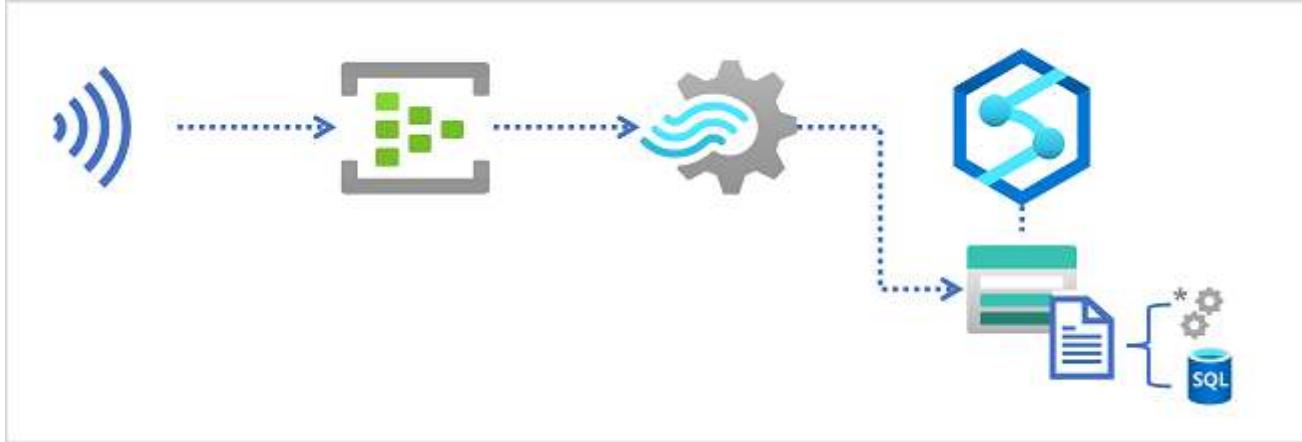


Data lakes in Azure Synapse Analytics

An Azure Synapse Analytics workspace typically includes at least one storage service that is used as a data lake. Most commonly, the data lake is hosted in an Azure Storage account using a container configured to support Azure Data Lake Storage Gen2. Files in the data lake are

organized hierarchically in directories (folders), and can be stored in multiple file formats, including delimited text (such as comma-separated values, or CSV), Parquet, and JSON.

When ingesting real-time data into a data lake, your Azure Stream Analytics query must write its results to an output that references the location in the Azure Data Lake Gen2 storage container where you want to save the data files. Data analysts, engineers, and scientists can then process and query the files in the data lake by running code in an Apache Spark pool, or by running SQL queries using a serverless SQL pool.



Next unit: Configure inputs and outputs

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Configure inputs and outputs

9 minutes

All Azure Stream Analytics jobs include at least one input and output. In most cases, inputs reference sources of streaming data (though you can also define inputs for static reference data to augment the streamed event data). Outputs determine where the results of the stream processing query will be sent. In the case of data ingestion into Azure Synapse Analytics, the output usually references an Azure Data Lake Storage Gen2 container or a table in a dedicated SQL pool database.

Streaming data inputs

Inputs for streaming data consumed by Azure Stream Analytics can include:

- Azure Event Hubs
- Azure IoT Hubs
- Azure Blob or Data Lake Gen 2 Storage

Depending on the specific input type, the data for each streamed event includes the event's data fields as well as input-specific metadata fields. For example, data consumed from an Azure Event Hubs input includes an **EventEnqueuedUtcTime** field indicating the time when the event was received in the event hub.

Note

For more information about streaming inputs, see [Stream data as input into Stream Analytics](#) in the Azure Stream Analytics documentation.

Azure Synapse Analytics outputs

If you need to load the results of your stream processing into a table in a dedicated SQL pool, use an **Azure Synapse Analytics** output. The output configuration includes the identity of the dedicated SQL pool in an Azure Synapse Analytics workspace, details of how the Azure Stream Analytics job should establish an authenticated connection to it, and the existing table into which the data should be loaded.

Authentication to Azure Synapse Analytics is usually accomplished through SQL Server authentication, which requires a username and password. Alternatively, you can use a managed identity to authenticate. When using an Azure Synapse Analytics output, your Azure Stream Analytics job configuration must include an Azure Storage account in which authentication metadata for the job is stored securely.

 **Note**

For more information about using an Azure Synapse Analytics output, see [Azure Synapse Analytics output from Azure Stream Analytics](#) in the Azure Stream Analytics documentation.

Azure Data Lake Storage Gen2 outputs

If you need to write the results of stream processing to an Azure Data Lake Storage Gen2 container that hosts a data lake in an Azure Synapse Analytics workspace, use a **Blob storage/ADLS Gen2** output. The output configuration includes details of the storage account in which the container is defined, authentication settings to connect to it, and details of the files to be created. You can specify the file format, including CSV, JSON, Parquet, and Delta formats. You can also specify custom patterns to define the folder hierarchy in which the files are saved - for example using a pattern such as *YYYY/MM/DD* to generate a folder hierarchy based on the current year, month, and day.

You can specify minimum and maximum row counts for each batch, which determines the number of output files generated (each batch creates a new file). You can also configure the *write mode* to control when the data is written for a time window - appending each row as it arrives or writing all rows once (which ensures "exactly once" delivery).

 **Note**

For more information about using a Blob storage/ADLS Gen2 output, see [Blob storage and Azure Data Lake Gen2 output from Azure Stream Analytics](#) in the Azure Stream Analytics documentation.

Next unit: Define a query to select, filter, and aggregate data

✓ 100 XP

Define a query to select, filter, and aggregate data

5 minutes

After defining the input(s) and output(s) for your Azure Stream Analytics job, you can define a **query** to process the incoming data from an input and write the results to an output.

Selecting input fields

The simplest approach to ingesting streaming data into Azure Synapse Analytics is to capture the required field values for every event using a **SELECT...INTO** query, as shown here:

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
```

Tip

When using an **Azure Synapse Analytics** output to write the results to a table in a dedicated SQL pool, the schema of the results produced by the query must match the table into which the data is to be loaded. You can use **AS** clauses to rename fields, and cast them to alternative (compatible) data types as necessary.

Filtering event data

In some cases, you might want to filter the data to include only specific events by adding a **WHERE** clause. For example, the following query writes data only for events with a negative **ReadingValue** field value.

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
WHERE ReadingValue < 0
```

Aggregating events over temporal windows

A common pattern for streaming queries is to aggregate event data over temporal (time-based) intervals, or *windows*. To accomplish this, you can use a **GROUP BY** clause that includes a Window function defining the kind of window you want to define (for example, *tumbling*, *hopping*, or *sliding*).

Tip

For more information about window functions, see [Introduction to Stream Analytics windowing functions](#) in the Azure Stream Analytics documentation.

The following example groups streaming sensor readings into 1 minute tumbling (serial, non-overlapping) windows, recording the start and end time of each window and the maximum reading for each sensor. The **HAVING** clause filters the results to include only windows where at least one event occurred.

SQL

```
SELECT
    DateAdd(second, -60, System.Timestamp) AS StartTime,
    System.Timestamp AS EndTime,
    SensorID,
    MAX(ReadingValue) AS MaxReading
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
GROUP BY SensorID, TumblingWindow(second, 60)
HAVING COUNT(*) >= 1
```

Tip

For more information about common patterns for queries, see [Common query patterns in Azure Stream Analytics](#) in the Azure Stream Analytics documentation.

Next unit: Run a job to ingest data

[Continue >](#)

How are we doing?

✓ 100 XP



Run a job to ingest data

3 minutes

When you've created and saved your query, you can run the Azure Stream Analytics job to process events in the input(s) and write the results to output(s). Once started, the query will run perpetually until stopped; constantly ingesting new event data into your Azure Synapse Analytics workspace (into a table in relational data warehouse or files in a data lake, depending on the output type).

Working with ingested data

You can work with the ingested streaming data like any other data in Azure Synapse Analytics, combining it with data ingested using batch processing techniques or synchronized from operational data sources by using Azure Synapse Link.

Querying data in a relational data warehouse

If you used an Azure Synapse Analytics output to ingest the results of your stream processing job into a table in a dedicated SQL pool, you can query the table using a SQL query, just like any other table. The results of the query will always include the latest data to be ingested at the time the query is run. Your data warehouse can include tables for streaming data as well as tables for batch ingested data, enabling you to join real-time and batch data for historical analytics.

For example, the following SQL code could be used to query a table named **factSensorReadings** that contains the results of stream processing, and combine it with a **dimDate** table containing detailed data about the dates on which readings were captured.

SQL

```
SELECT d.Weekday, s.SensorID, AVG(s.SensorReading) AS AverageReading
FROM factSensorReadings AS s
JOIN dimDate AS d
    ON CAST(s.ReadingTime AS DATE) = d.DateKey
GROUP BY d.Weekday, s.SensorID
```

💡 Tip

To Learn more about using a dedicated SQL pool to analyze data in a data warehouse, see the [Analyze data in a relational data warehouse](#) module on Microsoft Learn.

Querying data in a data lake

As streaming data is ingested into files in a data lake, you can query those files by using a serverless SQL pool in Azure Synapse Analytics. For example, the following query reads all fields from all Parquet files under the **sensors** folder in the **data** file system container.

SQL

```
SELECT *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/sensors/*',
    FORMAT = 'parquet') AS rows
```

💡 Tip

To Learn more about using serverless SQL pools to query files in a data lake, see the [Use Azure Synapse serverless SQL pool to query files in a data lake](#) module on Microsoft Learn.

You can also query the data lake by using code running in an Apache Spark pool, as shown in this example:

Python

```
%%pyspark
df =
spark.read.load('abfss://data@datalake.dfs.core.windows.net/sensors/*',
format='parquet'
)
display(df)
```

💡 Tip

To Learn more about using Apache Spark pools to query files in a data lake, see the [Analyze data with Apache Spark in Azure Synapse Analytics](#) module on Microsoft Learn.

✓ 200 XP ➔

Knowledge check

3 minutes

Check your knowledge

1. Which type of output should you use to ingest the results of an Azure Stream Analytics job into a dedicated SQL pool table in Azure Synapse Analytics? *

Azure Synapse Analytics

✓ Correct. An Azure Synapse Analytics output writes data to a table in an Azure Synapse Analytics dedicated SQL pool.

Blob storage/ADLS Gen2

Azure Event Hubs

2. Which type of output should be used to ingest the results of an Azure Stream Analytics job into files in a data lake for analysis in Azure Synapse Analytics? *

Azure Synapse Analytics

Blob storage/ADLS Gen2

✓ Correct. A Blob storage/ADLS Gen2 output writes data to files in a data lake.

Azure Event Hubs

✗ Incorrect. An Azure Event Hubs output does not write data to files in a data lake.

Next unit: Summary

Continue >

100 XP

Introduction

2 minutes

Microsoft Power BI is used by organizations all around the world to create dynamic, interactive data visualizations that reveal insights on which important business decisions are based. Access to timely data can be the difference between failure and success, so the ability to capture and visualize data in real-time, or as near as possible, is critical in many scenarios.

Azure Stream Analytics provides a way to process a stream of real-time data from an input such as Azure Event Hubs, and direct the results to an output. One possible output is a Power BI dataset, from which dashboards can consume data for real-time visualization.



In this module, we'll examine how to use Azure Stream Analytics to process a stream of real-time data, and send the results to a Power BI dataset for visualization.

Next unit: Use a Power BI output in Azure Stream Analytics

[Continue >](#)

How are we doing?

✓ 100 XP



Use a Power BI output in Azure Stream Analytics

6 minutes

All Azure Stream Analytics jobs include at least one input and output. In most cases, inputs reference sources of streaming data (though you can also define inputs for static reference data to augment the streamed event data). Outputs determine where the results of the stream processing query will be sent. To support real-time data visualization, you can use a **Power BI** output.

Streaming data inputs

Inputs for streaming data consumed by Azure Stream Analytics can include:

- Azure Event Hubs
- Azure IoT Hubs
- Azure Blob or Data Lake Gen 2 Storage

Depending on the specific input type, the data for each streamed event includes the event's data fields and input-specific metadata fields. For example, data consumed from an Azure Event Hubs input includes an **EventEnqueuedUtcTime** field indicating the time when the event was received in the event hub.

ⓘ Note

For more information about streaming inputs, see [Stream data as input into Stream Analytics](#) in the Azure Stream Analytics documentation.

Power BI outputs

You can use a Power BI output to write the results of a Stream Analytics query to a table in a Power BI streaming dataset, from where it can be visualized in a dashboard. When adding a Power BI output to a Stream Analytics job, you need to specify the following properties:

- **Output alias:** A name for the output that can be used in a query.

- **Group workspace:** The Power BI workspace in which you want to create the resulting dataset.
- **Dataset name:** The name of the dataset to be generated by the output. You shouldn't pre-create this dataset as it will be created automatically (replacing any existing dataset with the same name).
- **Table name:** The name of the table to be created in the dataset.
- **Authorize connection:** You must authenticate the connection to your Power BI tenant so that the Stream Analytics job can write data to the workspace.

ⓘ Note

For more information about Power BI outputs, see [Power BI output from Azure Stream Analytics](#) in the Azure Stream Analytics documentation.

Next unit: Create a query for real-time visualization

[Continue >](#)

How are we doing?

✓ 100 XP

Create a query for real-time visualization

6 minutes

To send streaming data to Power BI, your Azure Stream Analytics job uses a query that writes its results to a Power BI output. A simple query that forwards event data from an event hub directly to Power BI might look something like this:

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [powerbi-output]
FROM
    [eventhub-input] TIMESTAMP BY EventEnqueuedUtcTime
```

The results of the query determine the schema of the table in the output dataset in Power BI.

Alternatively, you might use your query to filter and/or aggregate the data, sending only relevant or summarized data to the Power BI dataset. For example, the following query calculates the maximum reading for each sensor other than sensor 0 for each consecutive minute in which an event occurs.

SQL

```
SELECT
    DateAdd(second, -60, System.Timestamp) AS StartTime,
    System.Timestamp AS EndTime,
    SensorID,
    MAX(ReadingValue) AS MaxReading
INTO
    [powerbi-output]
FROM
    [eventhub-input] TIMESTAMP BY EventEnqueuedUtcTime
WHERE SensorID <> 0
GROUP BY SensorID, TumblingWindow(second, 60)
HAVING COUNT(*) > 1
```

When working with window functions (such as the `TumblingWindow` function in the previous example), consider that Power BI is capable of handling a call every second. Additionally, streaming visualizations support packets with a maximum size of 15 KB. As a general rule, use

window functions to ensure data is sent to Power BI no more frequently than every second, and minimize the fields included in the results to optimize the size of the data load.

 **Note**

For more information about Power BI output limitations, see [Power BI output from Azure Stream Analytics](#) in the Azure Stream Analytics documentation.

Next unit: Create real-time data visualizations in Power BI

[Continue >](#)

How are we doing?     

✓ 100 XP



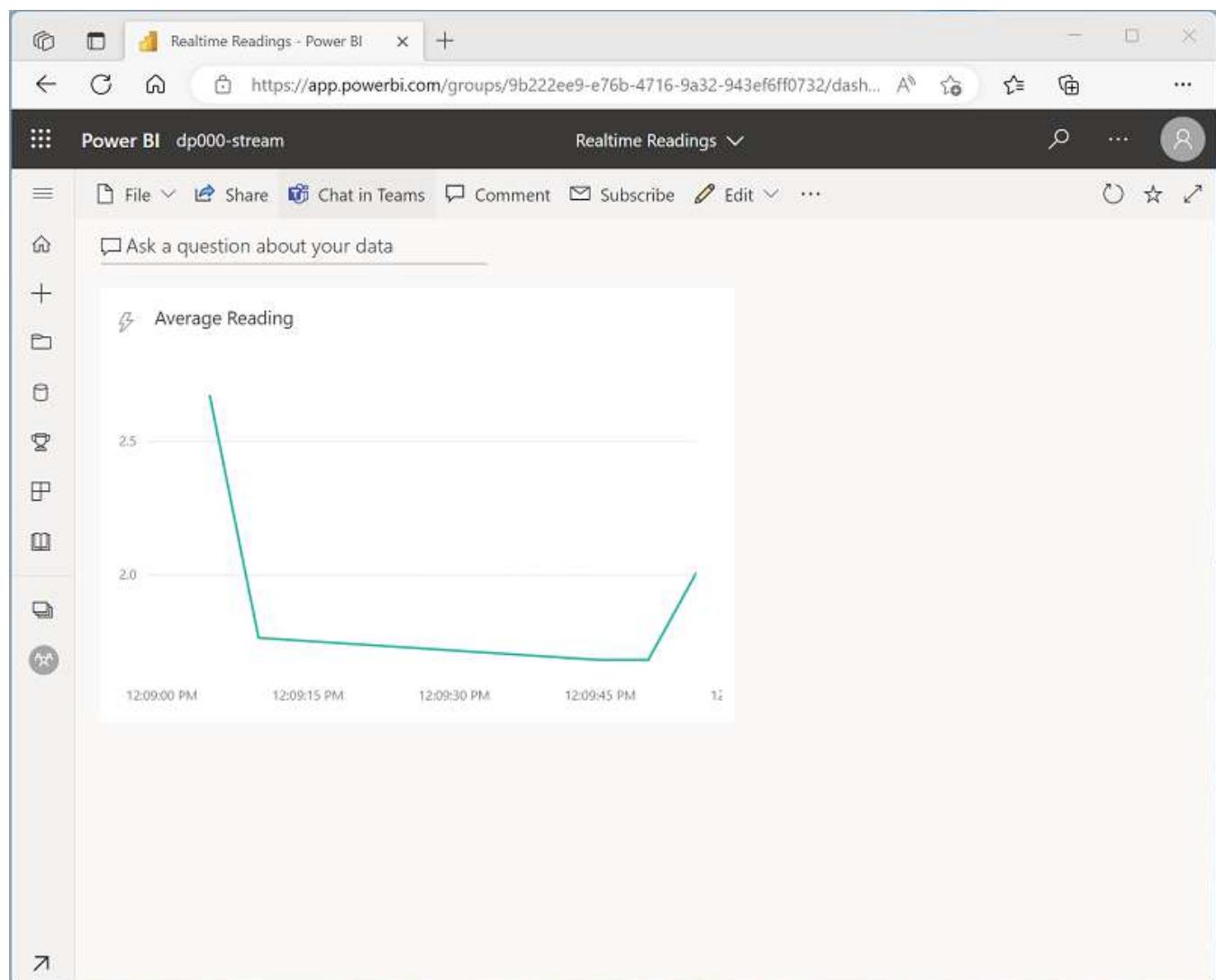
Create real-time data visualizations in Power BI

5 minutes

When you successfully run an Azure Stream Analytics job that sends results to a Power BI output, a streaming dataset containing a single table is created in the Power BI workspace specified for the output. The table contains the data produced by the Stream Analytics query.

Creating real-time visualizations in a dashboard

To visualize data in real-time, you can create a *dashboard* with a real-time visualization tile. Real-time visualizations on a dashboard show data from a streaming dataset, and are updated dynamically as new data flows into the dataset.



✓ 200 XP



Knowledge check

3 minutes

Check your knowledge

1. Which type of Azure Stream Analytics output should you use to support real-time visualizations in Microsoft Power BI? *

Azure Synapse Analytics

✗ Incorrect. An Azure Synapse Analytics output writes data to a table in an Azure Synapse Analytics dedicated SQL pool.

Azure Event Hubs

Power BI

✓ Correct. A Power BI output creates a dataset with a table of streaming data in a Power BI workspace.

2. You want to use an output to write the results of a Stream Analytics query to a table named device-events in a dataset named realtime-data in a Power BI workspace named analytics workspace. What should you do? *

Create only the workspace. The dataset and table will be created automatically.

✓ Correct. The dataset and table will be created dynamically by the output.

Create the workspace and dataset. The table will be created automatically.

✗ Incorrect. The dataset and table will be created dynamically by the output.

Create the workspace, dataset, and table before creating the output.

3. You want to create a visualization that updates dynamically based on a table in a streaming dataset in Power BI. What should you do? *

Create a report from the dataset.

Create a dashboard with a tile based on the streaming dataset.

✓ **Correct. A dashboard with a tile based on a streaming dataset updates dynamically as new data arrives.**

Export the streaming dataset to Excel and create a report from the Excel workbook.

✗ **Incorrect. A report based on an Excel workbook does not update dynamically.**

Next unit: Summary

[Continue >](#)

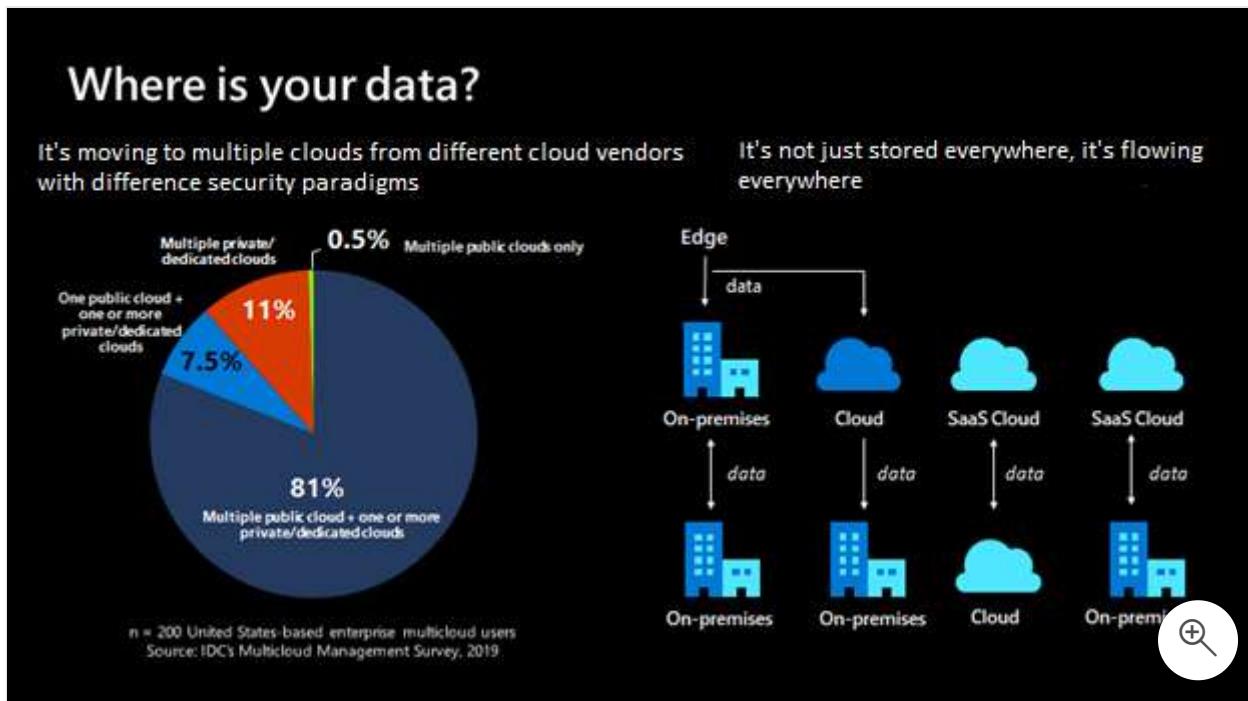
How are we doing?

Introduction

2 minutes

As the volume and variety of data increases, the challenges of good data governance are likely to become more difficult. Digital transformation technologies have resulted in new data sources. How do users know what data is available? How do administrators manage data when they might not know what type of data exists and where it's stored? Does the data contain sensitive or personal information?

All these questions aren't easy to answer without insights into the data and the source of storage. Before you can develop data-governance plans for storage and usage, you need to understand the data your organization uses.



Example scenario

As a user or producer of data, you might be a business or technical data analyst, data scientist, or data engineer. You probably spend significant time on manual processes to annotate, catalog, and find trusted data sources.

Because there's no central location to register data sources, you might be unaware of a data source unless you come into contact with it as part of another process.

Writing metadata descriptions for data sources is often a wasted effort. Client applications typically ignore descriptions that are stored in the data source. Creating documentation for data sources is difficult because you must keep documentation in sync with data sources. Users also might not trust documentation that they think is out of date.

Without the ability to track data from end to end, you must spend time tracing problems created by data pipelines that other teams own. If you make changes to your datasets, you can accidentally affect related reports that are business or mission critical.

Microsoft Purview is designed to address these issues and help enterprises get the most value from their existing information assets. The catalog makes data sources easy to discover and understand by the users who manage the data.



What will we be doing?

This high-level overview of Microsoft Purview helps you discover the key aspects that make it the tool of choice for mapping out your enterprise data. You learn how it can help you:

- Manage and govern your data across various platforms and locations.
- Map out your data landscape.
- Classify sensitive data.
- Empower customers to find trustworthy data.

What's the main goal?

By the end of this session, you'll be able to decide if Microsoft Purview is the right choice to help you manage your enterprise data environment and your various data sources.

Next unit: What is Microsoft Purview?

[Continue >](#)

How are we doing?

100 XP

What is Microsoft Purview?

3 minutes

Let's start with a few definitions and a quick tour of the core features of Microsoft Purview.

What's Microsoft Purview?

Microsoft Purview is a unified data-governance service that helps you manage and govern your on-premises, multicloud, and software-as-a-service (SaaS) data. You can easily create a broad, up-to-date map of your data landscape with:

- Automated data discovery.
- Sensitive data classification.
- End-to-end data lineage.

You can also allow data users to find valuable, trustworthy data.

Microsoft Purview is designed to help enterprises get the most value from their existing information assets. With this cloud-based service, you can register your data sources to help you discover and manage them. Your data sources remain in place, but a copy of the metadata for the source is added to Microsoft Purview.

You can register a wide range of sources in Azure and across your multicloud data estate in Microsoft Purview. These sources include Azure Data Lake Storage, AWS, Azure SQL Database on-premises and in the cloud, and many more.

Microsoft Purview has three main elements:

Microsoft Purview Data Map: The data map provides a structure for your data estate in Microsoft Purview, where you can map your existing data stores into groups and hierarchies. In the data map, you can grant users and teams access to these groups so that they have access to find relevant data stores. The data map can then scan your data stores and gather metadata, like schema and data types. It can also identify sensitive data types so that you can keep track of them in your data estate.

The screenshot shows the 'Sources' section of the Microsoft Purview Data Catalog. On the left is a vertical toolbar with icons for Home, Register, New collection, Refresh, Filter by name, and a lightbulb. The main area displays five data sources in a grid:

- AzureDataLakeStore-r6l** (Azure Data Lake Storage Gen1): Includes a lightning bolt icon, a 'View details' button, and edit, copy, and delete icons.
- AzureFileStorage-pX1** (Azure Files): Includes a blue folder icon, a 'View details' button, and edit, copy, and delete icons.
- CosmosDB-P8b** (Azure Cosmos DB (SQL API)): Includes a globe icon, a 'View details' button, and edit, copy, and delete icons.
- AzureBlob-B5Z** (Azure Blob Storage): Includes a folder icon, a 'View details' button, and edit, copy, and delete icons.
- AzureSqlDatabase-2vb** (Azure SQL Database): Includes a SQL icon, a 'View details' button, and edit, copy, and delete icons.

Microsoft Purview Data Catalog: The data catalog allows your users to browse the metadata stored in the data map so that they can find reliable data and understand its context. For example, users can see where the data comes from and who are the experts they can contact about that data source. The data catalog also integrates with other Azure products, like the Azure Synapse Analytics workspace, so that users can search for the data they need from the applications they need it in.

Microsoft Purview Data Estate Insights: Insights offer a high-level view into your data catalog, covering these key facets:

- **Data stewardship:** A report on how curated your data assets are so that you can track your governance progress.
- **Catalog adoption:** A report on the number of active users in your data catalog, their top searches, and your most viewed assets.
- **Asset insights:** A report on the data estate and source-type distribution. You can view by source type, classification, and file size. View the insights as a graph or key performance

indicators.

- **Scan insights:** A report that provides information on the health of your scans (successes, failures, or canceled).
- **Glossary insights:** A status report on the glossary to help users understand the distribution of glossary terms by status, or to view how the terms are attached to assets.
- **Classification insights:** A report that shows where classified data is located. It allows security administrators to understand the types of information found in their organization's data estate.
- **Sensitivity insights:** A report that focuses on sensitivity labels found during scans. Security administrators can make use of this information to ensure security is appropriate for the data estate.

Get a bird's-eye view of sensitive data with Data Estate Insights

- Reports on Assets, Scans, Glossary, Classification, and Labeling

Next unit: How Microsoft Purview works

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP



How Microsoft Purview works

5 minutes

Here's where we take a look at how Microsoft Purview works. In this unit, you learn the core operational theory behind the functioning of Microsoft Purview for mapping and scanning your data sources. The key areas we focus on include how to:

- Load data in the data map.
- Browse and search information in the data catalog.

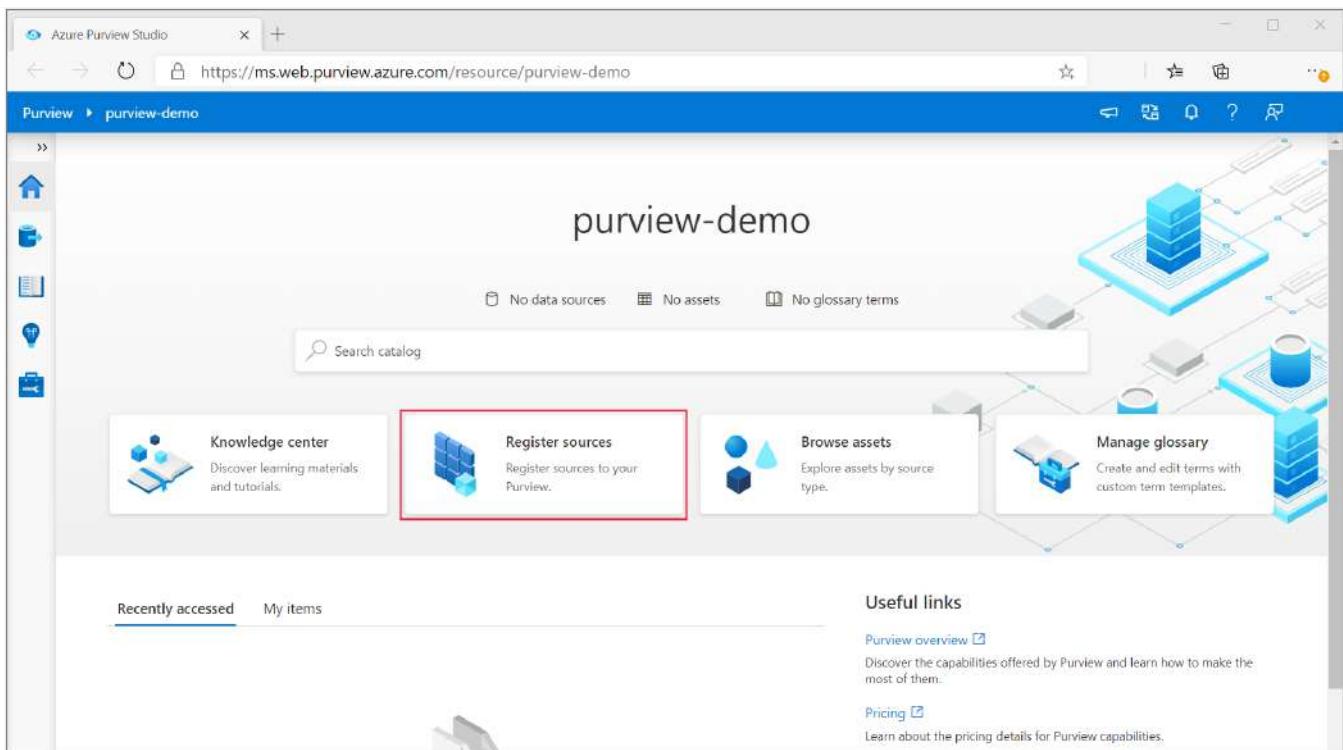
Load data in the data map

The Microsoft Purview Data Map is a unified map of your data assets and their relationships. As one cohesive map, it's easier for you and your users to visualize and govern. It also houses the metadata that underpins the Microsoft Purview Data Catalog and Data Estate Insights. It scales up and down to meet your enterprise compliance requirements. You can use it to govern your data estate in a way that makes the most sense for your business.

Source data

Sourcing your data starts with a process where you register data sources. Microsoft Purview supports an array of data sources that span on-premises, multicloud, and software-as-a-service (SaaS) options. You register the various data sources so that Microsoft Purview is aware of them. The data remains in its location and isn't migrated to any other platform.

After you have a Microsoft Purview service configured in Azure, you use the Microsoft Purview governance portal to register your data sources.



Each type of data source you choose requires specific information to complete the registration. For example, if your data sources reside in your Azure subscription, you choose the necessary subscription and storage account name. The following image is an example of choosing an Azure Blob Storage source.

Register sources (Azure Blob Storage)

Name *

Azure subscription

Storage account name *

Endpoint

Select a collection

After registration, you scan the data source. Scanning ingests metadata about your data source into the Microsoft Purview Data Map. Each data source has specific requirements for authenticating and configuration to permit scanning of the assets in that data source.

For example, if you have data stored in an Amazon S3 standard bucket, you need to provide a configuration for the connection. For this service, you use Microsoft Purview to provide a Microsoft account with secure access to AWS, where the Microsoft Purview scanner will run. The Microsoft Purview scanner uses this access to your Amazon S3 buckets to read your data. The scanner then reports the results (including only the metadata and classification) back to Azure. You can use the Microsoft Purview classification and labeling reports to analyze and review your data scan results.

 **Note**

Check the [Microsoft Purview connector for Amazon S3 documentation](#) for region support related to AWS S3 sources.

In Microsoft Purview, there are a few options to use for authentication when the service needs to scan data sources. Some of these options are:

- Microsoft Purview managed identity
- Account key (using Azure Key Vault)
- SQL authentication (using Key Vault)
- Service principal (using Key Vault)

Map data

The data map is the foundational platform for Microsoft Purview. The data map consists of:

- Data assets.
- Data lineage.
- Data classifications.
- Business context.

Customers create a knowledge graph of data that comes in from a range of sources. Microsoft Purview makes it easy to register and automatically scan and classify data at scale. Within the data map, you can identify the type of data source, along with other details around security and scanning.

The data map uses collections to organize these details. Collections are a way of grouping data assets into logical categories to simplify management and discovery of assets within the catalog. You also use collections to manage access to the metadata that's available in the data map.

Select **Map view** in the Microsoft Purview governance portal to display the data sources in a graphical view, along with the collections you created for them.

The screenshot shows the 'Sources' section of the Microsoft Purview Governance portal. It displays a grid of data source configurations. On the left, there's a sidebar with icons for Home, Sources, Scan, Metrics, Classifications, and Access control. The main area shows three collections: 'Sales' (Collection), 'Web-store-front' (Collection), and 'HumanResources' (Collection). Each collection has a 'View details' button. Below each collection is a list of registered sources: 'CosmosDB-P8b' (Azure Cosmos DB (SQL API)), 'AzureBlob-B5Z' (Azure Blob Storage), 'AzureDataLakeStore-r6l' (Azure Data Lake Storage Gen1), 'AzureSqlDatabase-2vb' (Azure SQL Database), and 'AzureFileStorage-pX1' (Azure Files). Each source has its own 'View details' button.

Scan data

After you register your data sources, you need to run a scan to access the metadata and browse the asset information. Before you can scan the data sources, you're required to enter the credentials for these sources. You can use Azure Key Vault to store the credentials for security and ease of access by your scan rules. The Microsoft Purview governance portal comes with existing system scan rule sets that you can select when you create a new scan rule. You can also specify a custom scan rule set.

A scan rule set is a container for grouping scan rules together to use the same rules repeatedly. A scan rule set lets you select file types for schema extraction and classification. It also lets you define new custom file types. You might create a default scan rule set for each of your data source types. Then you can use these scan rule sets by default for all scans within your company.

For example, you might want to scan only the .csv files in an Azure Data Lake Storage account. Or you might want to check your data only for credit card numbers rather than all the possible classifications. You might also want users with the right permissions to create other scan rule sets with different configurations based on business need.

The screenshot shows the 'Scan rule sets' section of the Microsoft Purview Governance portal. The left sidebar includes 'General', 'Scan rule sets' (selected), 'Integration runtimes', 'Metrics', 'Metadata management', 'Classifications', 'Classification rules', 'Resource sets', 'Pattern rules', 'Lineage connections', 'Data Factory', 'Data Share', 'Security and access', 'Access control', and 'Credentials'. The main area shows a table of scan rule sets:

Name	Source type	Version
AzureFileService	Azure Files	Version 2 (Update)
AzureDataExplorer	Azure Data Explorer (Kusto)	Version 1 (Update)
AdlGen2	Azure Data Lake Storage Gen2	Version 2 (Update)
AdlsGen1	Azure Data Lake Storage Gen1	Version 2 (Update)
AzureSqlDatabase	Azure SQL Database	Version 1 (Update)
AzureSynapseSQL	Azure Synapse Analytics	Version 1 (Update)
AzureCosmosDB	Azure Cosmos DB (SQL API)	Version 1 (Update)
AzureSqlDataWarehouse	Azure Dedicated SQL Pool (formerly SQL DW)	Version 1 (Update)
AzureSqlDatabaseManagedInstance	Azure SQL Database Managed Instance	Version 1 (Update)
AmazonS3	Amazon S3	Version 1 (Update)
AzureStorage	Azure Blob Storage	Version 1 (Update)
SqlServer	SQL Server	Version 1 (Update)

Classification

Metadata is used to help describe the data that's being scanned and made available in the catalog. During the configuration of a scan set, you can specify classification rules to apply during the scan that also serve as metadata. The classification rules fall under five major categories:

- **Government:** Attributes such as government identity cards, driver license numbers, and passport numbers.
- **Financial:** Attributes such as bank account numbers or credit card numbers.
- **Personal:** Personal information such as a person's age, date of birth, email address, and phone number.
- **Security:** Attributes like passwords that can be stored.
- **Miscellaneous:** Attributes not included in the other categories.

You can use several system classifications to classify your data. These classifications align with the sensitive information types in the Microsoft Purview compliance portal. You can also create custom classifications to identify other important or sensitive information types in your data estate.

General		Classifications		
		New	Edit	Delete
		System	Custom	
These are the system provided classifications.				
<input type="text"/> Filter by name...				
Display name		Formal name	Description	
ABA Routing Number		MICROSOFT.FINANCIALUS.ABA_ROUTING_NUMB...	ABA Routing Number	
Age of an individual		MICROSOFT.PERSONAL.AGE	Age of an individual	
Argentina National Identity (DNI) Number		MICROSOFT.GOVERNMENT.ARGENITNA.DNI_NU...	Argentina National Identity (DNI) Number	
Australia Bank Account Number		MICROSOFT.FINANCIALAUSTRALIA.BANK_ACCOU...	Australia Bank Account Number	
Australia Business Number		MICROSOFT.GOVERNMENT.AUSTRALIA.BUSINESS...	Australia Business Number	
Australia Company Number		MICROSOFT.GOVERNMENT.AUSTRALIA.COMPANY...	Australia Company Number	
Australia Driver's License Number		MICROSOFT.GOVERNMENT.AUSTRALIA.DRIVERS_L...	Australia Driver's License Number	
Australia Medical Account Number		MICROSOFT.GOVERNMENT.AUSTRALIA.MEDICAL_...	Australia Medical Account Number	
Australia Passport Number		MICROSOFT.GOVERNMENT.AUSTRALIA.PASSPORT...	Australia Passport Number	
Australia Tax File Number		MICROSOFT.GOVERNMENT.AUSTRALIA.TAX_FILE...	Australia Tax File Number	
Austria Driver's License Number		MICROSOFT.GOVERNMENT.AUSTRIA.DRIVERS.LIC...	Austria Driver's License Number	
Austria Identity Card		MICROSOFT.GOVERNMENT.AUSTRIA.IDENTITY.CA...	Austria Identity Card	

After you register a data source, you can enrich its metadata. With proper access, you can annotate a data source by providing descriptions, ratings, tags, glossary terms, identifying experts, or other metadata for requesting data-source access. This descriptive metadata supplements the structural metadata, such as column names and data types, that's registered from the data source.

Discovering and understanding data sources and their use is the primary purpose of registering the sources. If you're an enterprise user, you might need data for business intelligence, application development, data science, or any other task where the right data is

required. You can use the data catalog discovery experience to quickly find data that matches your needs. You can evaluate the data for its fitness for the purpose and then open the data source in your tool of choice.

At the same time, you can contribute to the catalog by tagging, documenting, and annotating data sources that have already been registered. You can also register new data sources, which are then discovered, evaluated, and used by the community of catalog users.

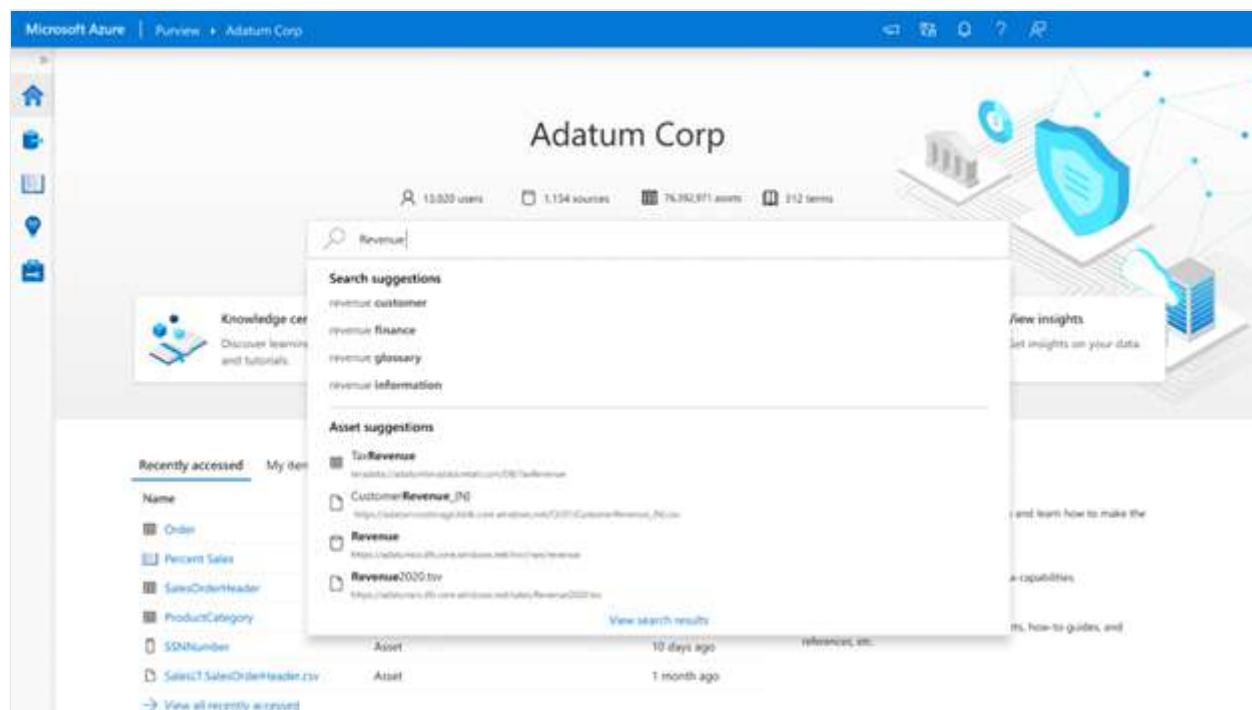
Browse and search

Microsoft Purview allows you to search information from the data map by using the Microsoft Purview Data Catalog. You can perform text-based search and browse through results by using filters like data source type, tags, ratings, or collection.

You can use business context to search information from the Microsoft Purview catalog. You can define business glossaries and bulk import existing ones, too. You can also apply business context onto assets in the data map. By using a metamodel, you can define business processes in your environment and associate your data sources with those processes. Users can then apply these business contexts to browse and search for information in the data catalog.

Discovery enables you to use:

- Semantic search and browse.
- Business glossary and workflows.
- Data lineage with sources, owners, transformations, and lifecycle.

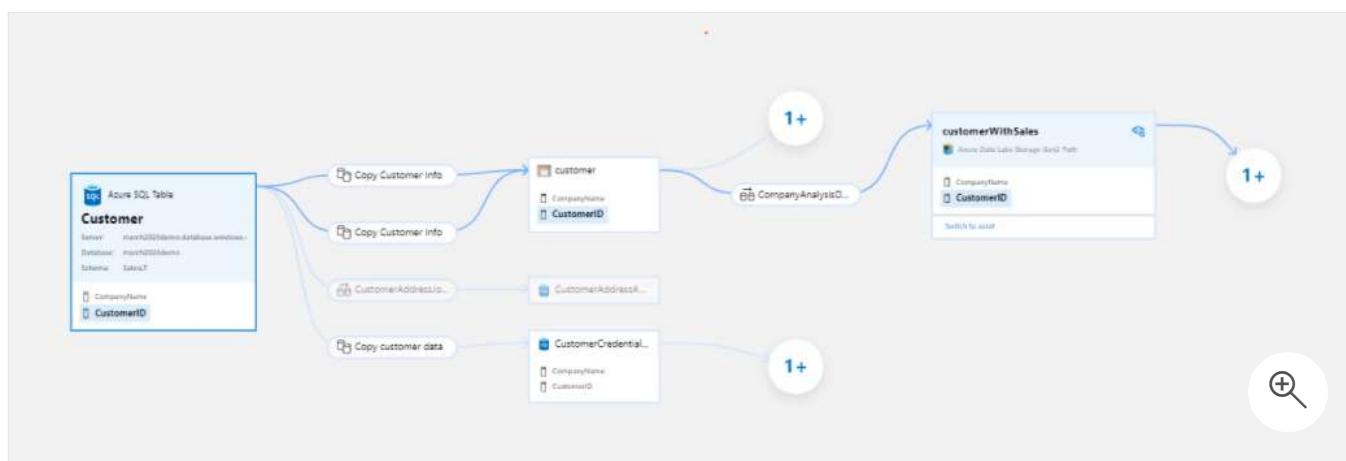


Data lineage

The concept of data lineage focuses on the lifecycle of data. The lifecycle concerns itself with the various stages data might go through. Data is sourced, moved, and stored throughout its lifecycle. Data might also undergo transformations in the extract, load, and transform/extract, transform, and load (ELT/ETL) operations.

Data lineage can offer insights into the data lifecycle by looking at the data pipeline. You can use the lineage to identify the root cause of data issues, perform data quality analysis, and verify compliance.

Microsoft Purview represents this data lineage in a visual form by showing data movement from source to destination.



100 XP

When to use Microsoft Purview

5 minutes

In this unit, we discuss how you can decide if Microsoft Purview is the right choice for your data governance and discovery needs. The criteria that indicate whether Microsoft Purview will meet your requirements are:

- Discovery
- Governance

Let's take a look at the criteria and see how Microsoft Purview can help address the needs in those specific areas.

Discovery

Without a central location to register data sources, you might be unaware of a data source unless you come into contact with it as part of another process.

Unless you know the location of a data source, you can't connect to the data by using a client application. You're required to know the connection string or path.

The intended use of the data is hidden to you unless you know the location of a data source's documentation. Data sources and documentation might live in several places and be utilized through different kinds of experiences.

Governance

As the data in your organization grows, the task of discovering, protecting, and governing that data becomes more difficult. Data is stored in different locations, which might be required for compliance reasons. The data might contain sensitive information such as credit card numbers, social security numbers, or other personal information.

Compliance with company security policies, government regulations, and customer needs are critical considerations for data governance. Understanding which data sources contain sensitive information is key to knowing where protections are needed and how to guard against access to this sensitive data.

General		Classifications																																									
		+ New Edit Delete Refresh																																									
		System Custom																																									
These are the system provided classifications.																																											
<input type="text"/> Filter by name...		<table border="1"> <thead> <tr> <th>Display name</th> <th>Formal name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ABA Routing Number</td> <td>MICROSOFT.FINANCIAL.US.ABA_ROUTING_NUMB...</td> <td>ABA Routing Number</td> </tr> <tr> <td>Age of an individual</td> <td>MICROSOFT.PERSONAL.AGE</td> <td>Age of an individual</td> </tr> <tr> <td>Argentina National Identity (DNI) Number</td> <td>MICROSOFT.GOVERNMENT.ARGENITNA.DNI_NU...</td> <td>Argentina National Identity (DNI) Number</td> </tr> <tr> <td>Australia Bank Account Number</td> <td>MICROSOFT.FINANCIAL.AUSTRALIA.BANK_ACCOU...</td> <td>Australia Bank Account Number</td> </tr> <tr> <td>Australia Business Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.BUSINESS....</td> <td>Australia Business Number</td> </tr> <tr> <td>Australia Company Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.COMPANY...</td> <td>Australia Company Number</td> </tr> <tr> <td>Australia Driver's License Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.DRIVERS_L...</td> <td>Australia Driver's License Number</td> </tr> <tr> <td>Australia Medical Account Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.MEDICAL_...</td> <td>Australia Medical Account Number</td> </tr> <tr> <td>Australia Passport Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.PASSPORT...</td> <td>Australia Passport Number</td> </tr> <tr> <td>Australia Tax File Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRALIA.TAX_FILE...</td> <td>Australia Tax File Number</td> </tr> <tr> <td>Austria Driver's License Number</td> <td>MICROSOFT.GOVERNMENT.AUSTRIA.DRIVERS.LIC...</td> <td>Austria Driver's License Number</td> </tr> <tr> <td>Austria Identity Card</td> <td>MICROSOFT.GOVERNMENT.AUSTRIA.IDENTITY.CA...</td> <td>Austria Identity Card</td> </tr> </tbody> </table>			Display name	Formal name	Description	ABA Routing Number	MICROSOFT.FINANCIAL.US.ABA_ROUTING_NUMB...	ABA Routing Number	Age of an individual	MICROSOFT.PERSONAL.AGE	Age of an individual	Argentina National Identity (DNI) Number	MICROSOFT.GOVERNMENT.ARGENITNA.DNI_NU...	Argentina National Identity (DNI) Number	Australia Bank Account Number	MICROSOFT.FINANCIAL.AUSTRALIA.BANK_ACCOU...	Australia Bank Account Number	Australia Business Number	MICROSOFT.GOVERNMENT.AUSTRALIA.BUSINESS....	Australia Business Number	Australia Company Number	MICROSOFT.GOVERNMENT.AUSTRALIA.COMPANY...	Australia Company Number	Australia Driver's License Number	MICROSOFT.GOVERNMENT.AUSTRALIA.DRIVERS_L...	Australia Driver's License Number	Australia Medical Account Number	MICROSOFT.GOVERNMENT.AUSTRALIA.MEDICAL_...	Australia Medical Account Number	Australia Passport Number	MICROSOFT.GOVERNMENT.AUSTRALIA.PASSPORT...	Australia Passport Number	Australia Tax File Number	MICROSOFT.GOVERNMENT.AUSTRALIA.TAX_FILE...	Australia Tax File Number	Austria Driver's License Number	MICROSOFT.GOVERNMENT.AUSTRIA.DRIVERS.LIC...	Austria Driver's License Number	Austria Identity Card	MICROSOFT.GOVERNMENT.AUSTRIA.IDENTITY.CA...	Austria Identity Card
Display name	Formal name	Description																																									
ABA Routing Number	MICROSOFT.FINANCIAL.US.ABA_ROUTING_NUMB...	ABA Routing Number																																									
Age of an individual	MICROSOFT.PERSONAL.AGE	Age of an individual																																									
Argentina National Identity (DNI) Number	MICROSOFT.GOVERNMENT.ARGENITNA.DNI_NU...	Argentina National Identity (DNI) Number																																									
Australia Bank Account Number	MICROSOFT.FINANCIAL.AUSTRALIA.BANK_ACCOU...	Australia Bank Account Number																																									
Australia Business Number	MICROSOFT.GOVERNMENT.AUSTRALIA.BUSINESS....	Australia Business Number																																									
Australia Company Number	MICROSOFT.GOVERNMENT.AUSTRALIA.COMPANY...	Australia Company Number																																									
Australia Driver's License Number	MICROSOFT.GOVERNMENT.AUSTRALIA.DRIVERS_L...	Australia Driver's License Number																																									
Australia Medical Account Number	MICROSOFT.GOVERNMENT.AUSTRALIA.MEDICAL_...	Australia Medical Account Number																																									
Australia Passport Number	MICROSOFT.GOVERNMENT.AUSTRALIA.PASSPORT...	Australia Passport Number																																									
Australia Tax File Number	MICROSOFT.GOVERNMENT.AUSTRALIA.TAX_FILE...	Australia Tax File Number																																									
Austria Driver's License Number	MICROSOFT.GOVERNMENT.AUSTRIA.DRIVERS.LIC...	Austria Driver's License Number																																									
Austria Identity Card	MICROSOFT.GOVERNMENT.AUSTRIA.IDENTITY.CA...	Austria Identity Card																																									

Apply the criteria

Let's take a look at how Microsoft Purview can address the data discovery and governance criteria.

Does Microsoft Purview help with data discovery?

Do you require a solution or centralized location to register data sources? Often, users might be unaware of a data source unless they come into contact with it as part of another process. Microsoft Purview can help to provide a solution.

After you've registered data sources in the Microsoft Purview governance portal and displayed them in the data map, you can set up scanning of those data sources. The metadata that's returned catalogs the data in those sources. In this way, it's easier for users to discover what the data sources contain. The metadata is indexed to make each data source easy to discover via search. It's also more understandable to the users who discover it.

Users can contribute to the catalog by tagging, documenting, and annotating data sources that have already been registered. They can register new data sources so that other catalog users can discover, understand, and utilize them.

The screenshot shows the Microsoft Purview Sources page. At the top, there's a navigation bar with 'Purview' and 'purview-demo'. A search bar says 'Search assets'. On the left, there's a sidebar with icons for Home, Sources (selected), Collections, Assets, and Glossary. The main area is titled 'Sources' with a 'Register' button, a 'New collection' button, and a 'Refresh' button. It shows 'Showing 1 source' with a table. The table has columns: Name, Source type, Collection, Source id, Scans, and Registered on. One row is shown: 'blob-storage' (Azure Blob Storage) with a 'New scan' status and registered on '11/24/20 11:45 ...'. A 'List view' dropdown is at the top right. The entire screenshot is framed by a red border.

Does Microsoft Purview help with data governance?

Microsoft Purview can scan and automatically classify data in files and tables. Microsoft Purview classifies data by Bloom Filter and RegEx. Bloom Filter classifications include attributes for city, country/region, place, and person information. RegEx classifications cover attributes that include categories like bank information (ABA routing numbers or country/region-specific banking account numbers), passport numbers, and country/region-specific identification numbers. You can find the [full list of supported classifications](#) in the documentation for Microsoft Purview.

Microsoft Purview also uses predefined Data Plane roles to help control who has access to the information in Microsoft Purview. For access, users can use the Microsoft Purview governance portal only if they're placed in at least one of the three supported roles. When a Microsoft Purview account is created, no one but the creator can access the account or use its APIs. New users must be put in one or more of the following roles:

- **Purview Data Reader role:** Has access to the Microsoft Purview governance portal and can read all content in Microsoft Purview except for scan bindings.
- **Purview Data Curator role:** Has access to the Microsoft Purview governance portal and can read all content in Microsoft Purview except for scan bindings. Can edit information about assets, classification definitions, and glossary terms. Can also apply classifications and glossary terms to assets.
- **Purview Data Source Administrator role:** Doesn't have access to the Microsoft Purview governance portal because the user must also be in the Data Reader or Data Curator roles. Can manage all aspects of scanning data into Microsoft Purview. Doesn't have read or write access to content in Microsoft Purview beyond those tasks related to scanning.

These roles are assigned by using the collections where your data sources are registered. You can grant users access to the data they might need without granting them access to the entire

data estate. By assigning roles, you can promote resource discoverability while still protecting sensitive information.

Next unit: Knowledge check

[Continue >](#)

How are we doing?

✓ 200 XP



Knowledge check

15 minutes

Choose the best response for each of the following questions, and then select **Check your answers**.

1. What does Microsoft Purview do with the data it discovers from your registered sources? *

*



It catalogs and classifies the data that's scanned.

✓ **Correct. It creates a broad, up-to-date map of your data landscape with automated data discovery, sensitive data classification, and end-to-end data lineage. It empowers data consumers to find valuable, trustworthy data.**



It moves the data to your Azure subscription, automatically creating the necessary storage accounts.

✗ **Incorrect. Microsoft Purview doesn't move the data. It leaves the data in its original storage location but maintains a link to the data.**



It performs data transformations to match your on-premises schemas.

2. Where would you register your data sources for use in Microsoft Purview? *



On the Overview tab of the Microsoft Purview account page.

✗ **Incorrect. The overview page for the Microsoft Purview account provides information about that account only.**



On the Managed Resources tab of the Microsoft Purview account page.



In the Microsoft Purview governance portal.

✓ **Correct. The Microsoft Purview governance portal is where you register your data sources.**

3. What aspect of Microsoft Purview is used to configure the data discovery for your data sources? *



Scan rules

✓ Correct. In a Microsoft Purview catalog, you can create scan rule sets to enable you to quickly scan data sources in your organization.



Collections



Classifications

✗ Incorrect. Classifications are used to identify key attributes of the data in the data sources. The five categories are government, financial, personal, security, or miscellaneous.

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

2 minutes

Microsoft Purview is a unified data governance service that helps you manage and govern your on-premises, multi-cloud, and software-as-a-service (SaaS) data. Data professionals can easily create a holistic, up-to-date map of the entire data landscape. Microsoft Purview includes automated data discovery, sensitive data classification, and end-to-end data lineage. Microsoft Purview can empower data analysts and other data consumers to find valuable, trustworthy data.

Imagine that you're a new Data Analyst at Contoso. In your second week, the sales manager desperately asks for the latest inventory and sales data for an impromptu review. After clarifying the requirements, you know you need to quickly find accurate assets to create a report. With the help of Microsoft Purview data catalog, you'll be able to search, browse, and discover assets. More importantly, you'll be able to validate that you're using the right data sources for your reports.

Learning objectives

In this module, you will:

- Browse and search data catalog assets.
- Use data catalog assets with Power BI.
- Use Microsoft Purview in Azure Synapse Studio.

Next unit: Search for assets

[Continue >](#)

How are we doing?

✓ 100 XP ➔

Search for assets

10 minutes

Microsoft Purview offers a central place to discover and understand assets to use in your day-to-day activities. This central place, Microsoft Purview data catalog, provides advanced search capabilities to quickly find the right assets and information. Using keywords, business terms and Microsoft Purview data catalog functionalities, you can find the assets needed to build and design reports.

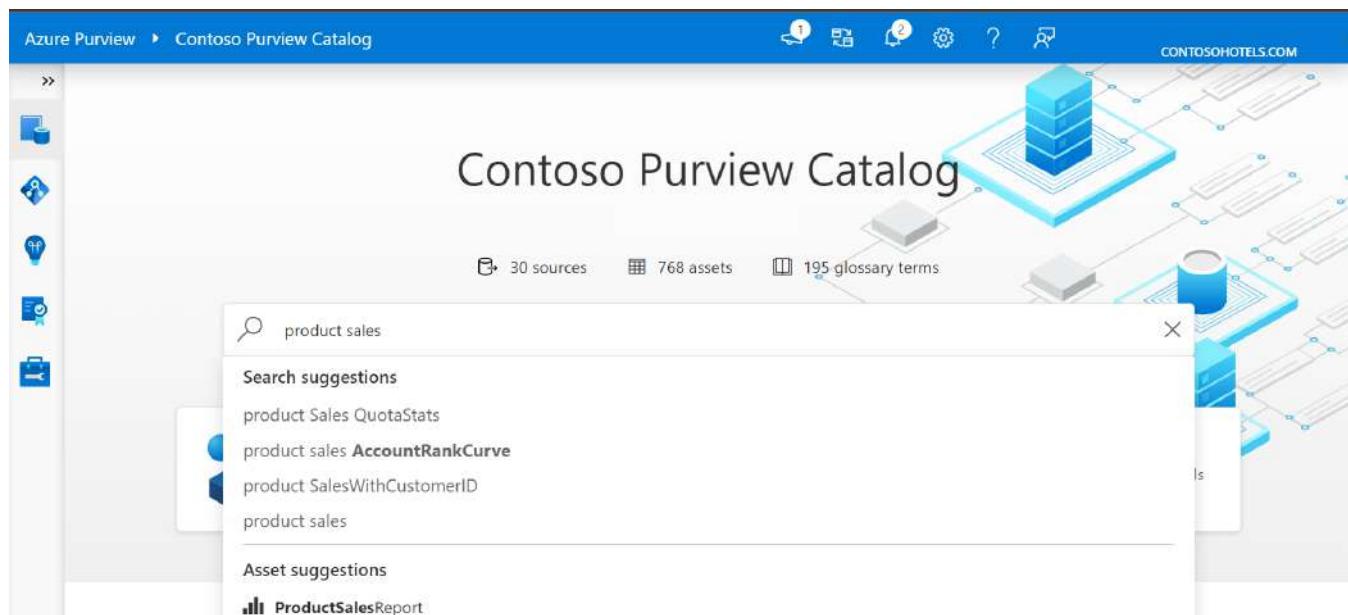
As a data analyst looking for assets, you'll be searching the Microsoft Purview data catalog. This assumes that the *Microsoft Purview data Map* has been created by your organization. The data map provides the foundation for data discovery. The data map captures metadata about enterprise data in analytics and operations systems on-premises and in the cloud and must be established before the data catalog can be searched.

ⓘ Note

Learn more about the [Microsoft Purview data map components](#).

Search the Microsoft Purview data catalog

From the Microsoft Purview Studio home page, users can type relevant keywords to start discovering assets. In this scenario, you're looking for "product sales."



The screenshot below displays the search result, with all assets corresponding to the keywords entered in the search engine.

The screenshot shows the Microsoft Purview Data Catalog interface. The left sidebar has a tree view with 'Data Catalog' at the top, followed by 'Search results for product sales'. Below this are sections for 'Object Type', 'Collection', and 'Classification', each with checkboxes and counts (e.g., 66 for Sales). The main pane displays search results for 'product sales' with 1-25 out of 153 results shown. The results include:

- Product Mix.csv** (Azure Blob Path)
- Sales Dashboard-SQLDB** (Power BI Dataset, Secret)
- SalesWithCustomerID.csv** (Azure Blob Path)
- SalesWithCustomerID** (Azure SQL Table, certified)
Table with sales raw information.
- ReportTable_Prod** (Azure SQL Table, certified)
Report Table in production environment to create Power BI report for sales people. To...

You can fine-tune your search using the filters on the left side of the page.

You can filter by:

- Source type (and instance if needed)
- Object type
- Classification
- Glossary term
- If needed, more options are available like Collection, Contact and Label

The screenshot shows the Azure Purview Data Catalog interface. At the top, there's a navigation bar with 'Azure Purview' and 'Contoso Purview Catalog'. A search bar contains the text 'product sales'. On the left, there's a sidebar with icons for 'Data catalog', 'Source type', 'Object Type' (with 'Tables' selected), and 'Collection'. The main area displays search results for 'product sales' with two items:

- ReportTable_Prod** (certified) - Azure SQL Table: Report Table in production environment to create Power BI report for sales people. Table de reporting ...
- Product** - Azure SQL Table

Below the results, there are sections for 'Classification' and 'Glossary term', both of which are highlighted with red boxes. The 'Classification' section lists various categories like 'Franser Product' (selected), 'Franser SIN', 'Person's Name', etc. The 'Glossary term' section lists terms like 'Products', 'Sales', 'Customer Product', 'Product client' (selected), 'Consumer Products', etc.

You've been instructed to connect to sources like Azure SQL tables. In the result displayed below, two assets are displayed. To use the correct asset, it's possible to browse each asset to dig for more detailed information. Alternatively, you can rely on the work done by the data stewards who have labeled certified assets for the organization.

Microsoft Azure | Azure Purview | Contoso Purview Catalog

Search results for product sales

Source type : Azure SQL Database | Instance : all | Clear all filters

Filter by keyword

Object Type

- Dashboards
- Data pipelines
- Files
- Folders
- Reports
- Stored procedures
- Tables

Collection

Showing 1-2 out of 2 results

ReportTable_Prod (certified)

Azure SQL Table

Report Table in production environment to create Power BI report for sales people.

Product

Azure SQL Table

Before using this asset to create your report, you need to verify more details and validate where data comes from to populate this asset. Select the asset to access more information.

Understand a single asset

Asset overview

Select an asset to see the *overview*. The overview displays information at a glance, including a description, asset classification, schema classification, collection path, asset hierarchy, and glossary terms.

Microsoft Azure | Azure Purview | Contoso Purview Catalog

Search results "product sales"

ReportTable_Prod (Certified)

Azure SQL Table

Edit | Select for bulk edit | Request access | Refresh | Delete

Overview Properties Schema Lineage Contacts Related

Updated on March 15, 2022 10:16 PM UTC by

Asset description

Report Table in production environment to create Power BI report for sales people.

Classifications (4)

Credit Card Number, Person's Name, Franner Product, Numero De Carte De Credit

Schema classifications (9)

Canada Social Insurance Number, Country/Region, Credit Card Number, EU Mobile Phone Number, EU Phone Number, Person's Name, U.S. Phone Number, Franner Product, Franner SIN

Fully qualified name

Collection path

- Contoso Purview Catalog
- PurviewNinja
- Cloud_Data
- EMEA
- NinjaSales

Hierarchy

- Azure SQL Server
- Azure SQL Database
- Azure SQL Schema
- ReportTable_Prod

Azure SQL Table

Glossary terms (6)

Customer Product, Product client, Products, Sales, Saleswithtax

The *asset description* provides a brief explanation of the purpose of an asset. Data stewards have made data analysts lives easier in the screenshot below, by noting that this is the correct resource to use for sales reporting.



ReportTable_Prod

Certified

Azure SQL Table

[Edit](#) [Select for bulk edit](#) [Request access](#) [Refresh](#) [Delete](#)[Overview](#) [Properties](#) [Schema](#) [Lineage](#) [Contacts](#) [Related](#)

Asset description

Report Table in production environment to create Power BI report for sales people.

Beneath the description, you'll see the *asset classification* and *schema classification*.

Data classification, in the context of Microsoft Purview, is a way of categorizing data assets by assigning unique logical labels or classes. Classification is based on the business context of the data. For example, you might classify assets by Passport Number, Driver's License Number, Credit Card Number, SWIFT Code, Person's Name, and so on. Asset classifications can be automatically applied during a scan or applied manually.

Note

Microsoft Purview comes with more than 200 classifications out of the box. For a full list of classifications, see [System classifications in Microsoft Purview](#).

The overview tab reflects both asset level classifications and column level classifications that have been applied, which you can also view as part of the schema.



ReportTable_Prod

Certified

Azure SQL Table

[Edit](#) [Select for bulk edit](#) [Request access](#) [Refresh](#) [Delete](#)[Overview](#) [Properties](#) [Schema](#) [Lineage](#) [Contacts](#) [Related](#)

Asset description

Report Table in production environment to create Power BI report for sales people.

Classifications (4)

[Credit Card Number](#) [Person's Name](#) [Franmer Product](#) [Numero De Carte De Credit](#)

Schema classifications (9)

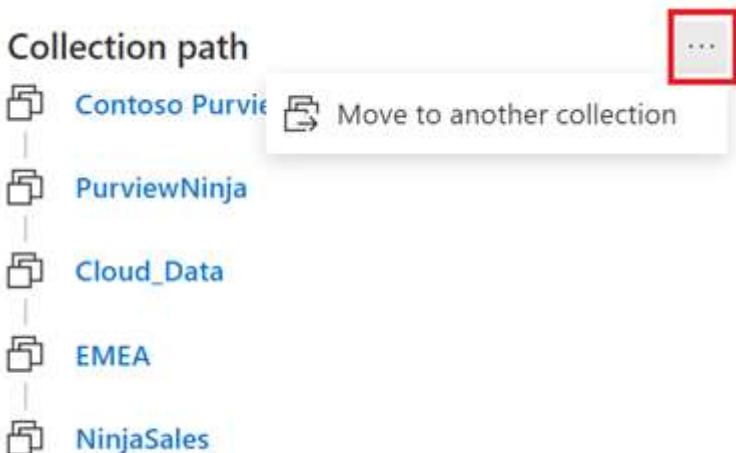
[Canada Social Insurance Number](#) [Country/Region](#) [Credit Card Number](#) [EU Mobile Phone Number](#) [EU Phone Number](#)
[Person's Name](#) [U.S. Phone Number](#) [Franmer Product](#) [Franmer SIN](#)

Important

You may notice that the classifications displayed above are sensitive or contain personally identifiable information (PII). data encryption is done at the source level, and Microsoft Purview stores only the metadata. It does not preview data.

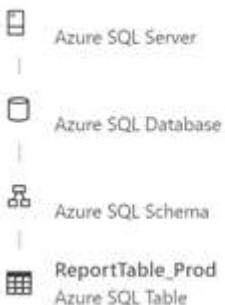
You can also view the *collection path*, *hierarchy* and *glossary terms* on the right side of the overview tab.

The *collection path* refers to the location of the asset inside Microsoft Purview. You have the option to move an asset to another collection.



You can view the full asset hierarchy within the overview tab. As an example: if you navigate to a SQL table, then you can see the schema, database, and the server the table belongs to.

Hierarchy



Glossary terms are a managed vocabulary for business terms that can be used to categorize and relate assets across your environment. For example, terms like 'customer,' 'buyer,' 'cost center,' or any terms that give your data context for your users. You can view the glossary terms for an asset in the overview section, and you can add a glossary term on an asset by editing the asset.

Glossary terms (6)

Customer Product

Produit client

Products

Sales

Saleswithtax

SSN

! Note

For more information, see the [business glossary page](#).

Asset schema

The *schema* view of the asset includes more granular details about the asset, such as column names, data types, column level classifications, terms, and descriptions.

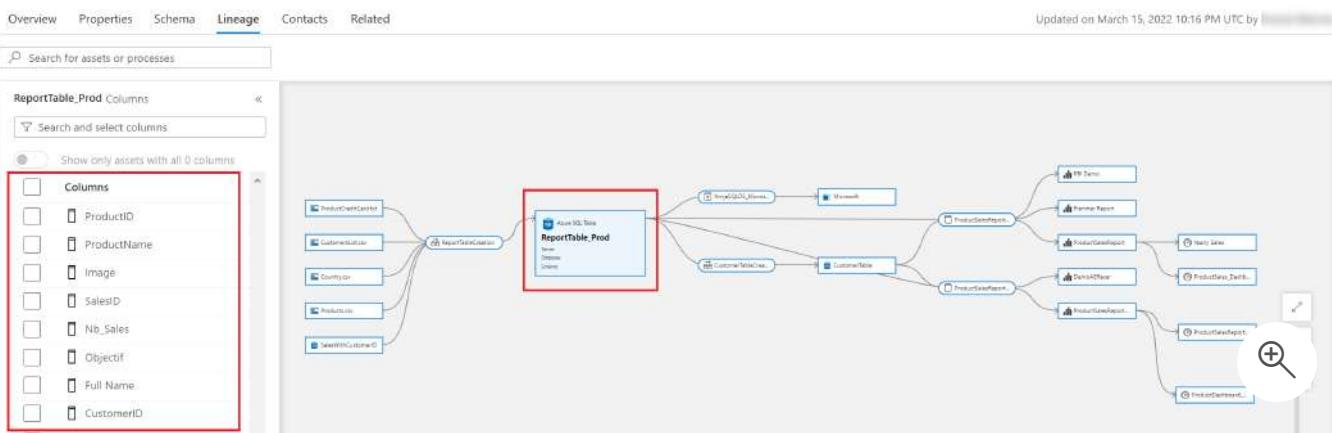
ReportTable_Prod Certified				
Azure SQL Table				
Edit Select for bulk edit Request access Refresh Delete Edit columns				
Overview	Properties	Schema	Lineage	Contacts
Related				Updated
<p>Info The schema was modified by on 03/15/2022, 10:16:53 PM UTC.</p> <p>Filter by name</p>				
Showing 17 of 17 items				
Column name	Classifications	Glossary terms	Data type	Asset description
CardNumber	Credit Card Number	Payments	nvarchar	Card number compliant with Luhn test
CardType		Payments	nvarchar	Card type (VISA, MASTERCARD, AMEX,...)
Country	Country/Region		nvarchar	Country Name
CountryID			nvarchar	Unique Identifier for country
CustomerID			nvarchar	Unique identifier for customer
Flag			nvarchar	Link to flag picture stored in data lake
Full Name	Person's Name	Personal Information	nvarchar	Full Name
Image		Consumer Products	nvarchar	Link to product picture stored in data lake
Nb_Sales		Sales	int	Technical name from database
Nom	Person's Name	Personal Information	nvarchar	Name
Objectif		Sales	int	Objective in CAD (with taxes)
Phone	EU Mobile Phone Number +2 More	Personal Information	nvarchar	Phone number
Prenom	Person's Name	Personal Information	nvarchar	First Name
ProductID			nvarchar	Unique Identifier for product

Asset lineage

Asset lineage gives you a clear view of how the asset is populated and where data comes from. *Data lineage* is broadly understood as the lifecycle that spans the data's origin, and where it moves over time across the data estate. Data lineage is important to analysts because it enables understanding of where data is coming from, what upstream changes may have occurred, and how it flows through the enterprise data systems.

A single view on the asset lineage tab displays the data flow to and from the asset. Asset lineage can also help you understand how the asset was built and how the asset is used inside the organization.

The columns pane on the left side of the lineage tab allows users to select and track columns as they flow through the lineage. For example, if you select the column Full Name, you can see how the Full Name field was created and where the information comes from.



! Note

The lineage view is a powerful way to understand the transformation process an asset has undergone. Learn more about the [lineage experience in Microsoft Purview data catalog](#)

Asset contacts and related assets

Asset *contacts* provide you contact details of experts or dataset owners with any questions. As a new analyst searching for the right data sources for your report, you may find these individuals helpful.

**ReportTable_Prod**

Certified

Azure SQL Table



Edit



Select for bulk edit



Request access



Refresh



Delete

[Overview](#)[Properties](#)[Schema](#)[Lineage](#)[Contacts](#)[Related](#)

Here are the people you can contact about this asset:

Experts (2)



Owners (2)



If needed, you can also navigate through the technical hierarchy of assets that are related to the current asset you're viewing.

» Data catalog > Search results "product sales" >

ReportTable_Prod Certified
Azure SQL Table

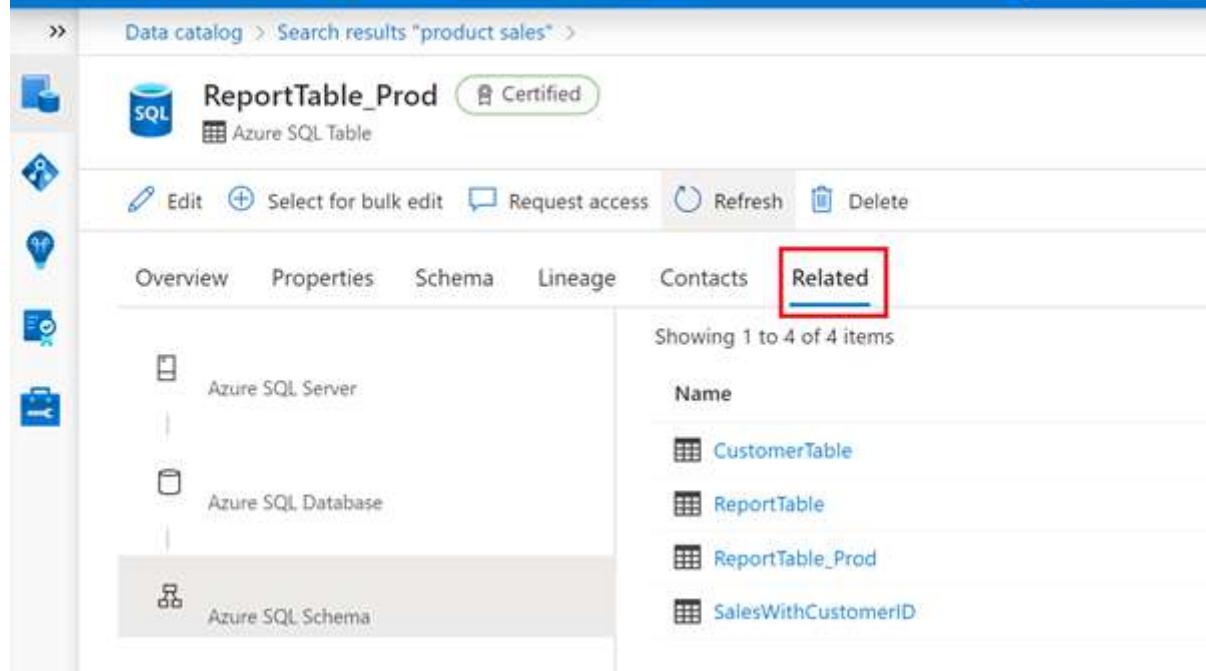
Edit Select for bulk edit Request access Refresh Delete

Overview Properties Schema Lineage Contacts **Related**

Showing 1 to 4 of 4 items

Name
CustomerTable
ReportTable
ReportTable_Prod
SalesWithCustomerID

Azure SQL Server
Azure SQL Database
Azure SQL Schema



The ability to search the Microsoft Purview data catalog has the potential to break down data silos and enable the next level of enterprise analytics.

Next unit: Browse assets

[Continue >](#)

How are we doing?

✓ 100 XP ➔

Browse assets

4 minutes

Searching a data catalog is a great tool for data discovery you know what you're looking for. Often, you may not know how your data estate is structured. The Microsoft Purview data catalog offers a browse experience that enables exploration of available data, either by collection or by exploring the hierarchy of each data source in the catalog.

Browse by collection or source type

If you're new to an organization or department, you may want to familiarize yourself with the contents of the data estate. From the Microsoft Purview Studio home page, select the "Browse assets" tile to browse either by collection or by source type.

The screenshot shows the Microsoft Purview Studio home page for the 'Contoso Purview Catalog'. At the top, there's a navigation bar with 'Azure Purview' and 'Contoso Purview Catalog'. Below the navigation bar, the title 'Contoso Purview Catalog' is displayed. There are three main tiles: 'Browse assets' (highlighted with a red box), 'Manage glossary', and 'Knowledge center'. Below these tiles, there's a 'Recently accessed' section showing a single item named 'ReportTable_Prod' last updated a day ago. On the right, there's a 'Links' section with a link to 'Azure Purview overview'.

Here you can specify whether you'd like to browse by collection or by source type.

Browse by collection allows you to explore the different collections you're a data reader or curator for. You'll only see collections you have access to. Select a collection to get a list of assets in that collection with the facets and filters available in search.

The screenshot shows the Microsoft Purview Data Catalog interface. The left sidebar has a tree view with 'Data catalog > AdventureWorks > North America > Sub collection(s)'. Under 'Sub collection(s)', there are 'Related' items: 'North America (Parent)', 'Curated Zone', and 'Raw Zone'. Below that is a section titled 'Narrow results by:' with dropdown menus for 'Classification', 'Contact', 'Content type', and 'Label'. The main pane displays a list of assets under 'Source Systems': 'Address' (Azure SQL Table), 'Customer' (Azure SQL Table), 'SalesOrderDetail' (Azure SQL Table), 'ProductModelProductDescription' (Azure SQL Table), and 'CustomerAddress' (Azure SQL Table). At the bottom of the main pane, there is a navigation bar with 'Previous', 'Page 1 of 1', and 'Next'.

💡 Tip

Collections are a tool to manage ownership and access control across assets and data sources. They also organize assets and sources into categories that are customized to match the business flow. See [Create and manage collections in Microsoft Purview](#) to learn more.

Browse by source type allows you to explore the hierarchies of data sources using an explorer view.

After selecting a tile associated with a data source type, you'll see a list of assets belonging to that type. From there, you'll be able to use the explorer view to see parent and child assets.

Name	Type	Owner	Action
Address	Azure SQL Table	-	...
Address_SchemaChange	Azure SQL Table	-	...
Customer	Azure SQL Table	-	...
CustomerAddress	Azure SQL Table	-	...
Product	Azure SQL Table	-	...
ProductCategory	Azure SQL Table	-	...
ProductDescription	Azure SQL Table	-	...
ProductModel	Azure SQL Table	-	...
ProductModelProductDescription	Azure SQL Table	-	...
SalesOrderDetail	Azure SQL Table	-	...
SalesOrderHeader	Azure SQL Table	-	...
vGetAllCategories	Azure SQL View	-	...
vProductAndDescription	Azure SQL View	-	...
vProductModelCatalogDescription	Azure SQL View	-	...
VW_ColumnDescription	Azure SQL View	-	...
VW_ColumnDescriptionNew	Azure SQL View	-	...

The Microsoft Purview data catalog browse experience enables analysts or data consumers to explore what data is available in many different ways. Microsoft Purview has the ability to enable users access to data you may not have known about before. The possibilities are endless so long as your organization's data stewards have scanned and classified data across the estate.

✓ 100 XP ➔

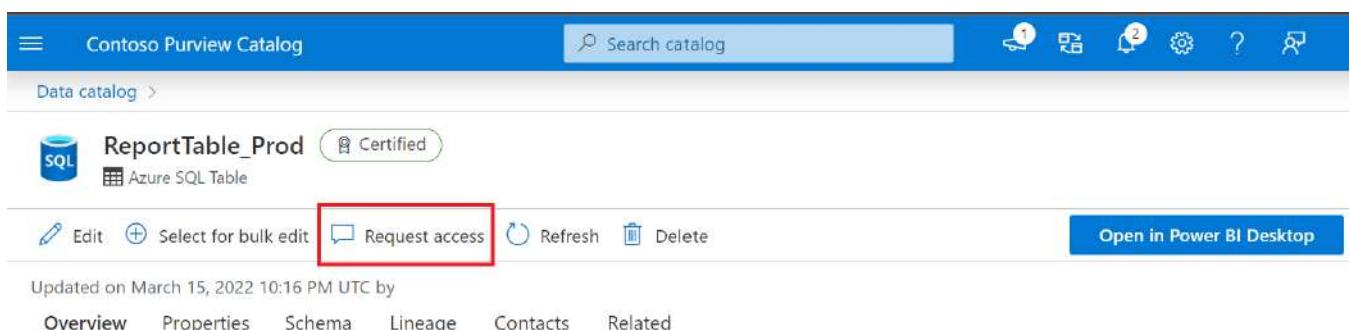
Use assets with Power BI

3 minutes

The integration of Microsoft Purview and Power BI makes it possible to gain a more complete understanding of the data across your estate.

Request access to assets

In your search or browsing session, you may come across assets that you don't have access to. Microsoft Purview makes it simple to request access directly from the Data Catalog by using the "Request access" button. Requesting access will kick off a workflow that manages requests and approvals.



The screenshot shows the Microsoft Purview Catalog interface. At the top, there's a blue header bar with the title 'Contoso Purview Catalog'. Below the header, a search bar says 'Search catalog'. On the right side of the header are several icons. The main content area shows a data asset named 'ReportTable_Prod' which is 'Certified' and an 'Azure SQL Table'. Below the asset name are several buttons: 'Edit', 'Select for bulk edit', 'Request access' (which is highlighted with a red box), 'Refresh', and 'Delete'. To the right of these buttons is a blue button labeled 'Open in Power BI Desktop'. Underneath the asset, it says 'Updated on March 15, 2022 10:16 PM UTC by'. Below this, there are tabs for 'Overview' (which is underlined in blue), 'Properties', 'Schema', 'Lineage', 'Contacts', and 'Related'. In the 'Asset description' section, it says 'Report Table in production environment to create Power BI report for sales people.' In the 'Classifications (4)' section, there are three categories: 'Credit Card Number', 'Person's Name', and 'Frammer Product'. To the right, there's a 'Collection path' tree view starting from 'Contoso Purview Catalog' and branching down through 'Purview', 'Cloud_Data', and 'EMEA'. There's also a small '...' icon next to the collection path.

Build a Power BI report using data discovered in Purview

Working as a new analyst, you've taken the time to search and browse assets and now you'd like to use those trusted assets in a Power BI report. Purview makes it simple, with the ability to open the asset in Power BI desktop.

The screenshot shows the Microsoft Purview Catalog interface. At the top, there's a search bar labeled 'Search catalog'. Below the search bar, the page title is 'Contoso Purview Catalog' and the sub-page title is 'Data catalog >'. A blue icon representing an Azure SQL Table is next to the asset name 'ReportTable_Prod'. A 'Certified' badge is also present. Below the asset name, there are several buttons: 'Edit', 'Select for bulk edit', 'Request access', 'Refresh', and 'Delete'. A red box highlights the 'Open in Power BI Desktop' button. To the right of the main content area, there's a 'Collection path' sidebar with a tree structure: Contoso Purview Catalog > Purview > Cloud_Data > EMEA > Sales. Below the sidebar, there's a red arrow pointing upwards towards the 'Open in Power BI Desktop' button.

Asset description

Report Table in production environment to create Power BI report for sales people.

Classifications (4) ^①

Credit Card Number Person's Name Franmer Product
Numero De Carte De Credit

Collection path

Contoso Purview Catalog
Purview
Cloud_Data
EMEA
Sales

Selecting Open in Power BI Desktop initiates the download of a Power BI Data Source file (PBIDS) you can open with Power BI Desktop. PBIDS files contain a connection to the data source, so all you need to do is enter credentials upon opening the file and you're ready to start building.

Scan a Power BI tenant

In addition to using Purview to find trusted data sources in the data estate to build reports, you can also scan your Power BI tenant to manage and catalog assets. The metadata of Power BI assets, and information about their lineage across Power BI workspaces and their connections to data sources, are then available in Microsoft Purview.

! Note

See [Connect to and manage a Power BI tenant in Microsoft Purview](#) for more details.

Next unit: Integrate with Azure Synapse Analytics

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



Integrate with Azure Synapse Analytics

4 minutes

Microsoft Purview can be integrated directly into Azure Synapse. If Azure Synapse Studio is massively deployed in your organization, you can get the data catalog experience directly in Azure Synapse Studio.

This integrated experience allows you to discover Microsoft Purview assets, interact with them through Synapse capabilities, and push lineage information to Microsoft Purview.

ⓘ Note

To connect an Microsoft Purview Account to a Synapse workspace, you need 2 types of permissions. You need a contributor role in Synapse workspace from Azure portal identity and access management (IAM). You also need access to that Microsoft Purview Account. For more information, see [Microsoft Purview permissions](#).

Let's imagine you need to find and understand some assets before working with them in pipelines or notebooks. From Azure Synapse Studio, you can easily query your Microsoft Purview data catalog.

In Azure Synapse Studio, from the **Data** blade on the left, select **Purview** in the dropdown next to the search bar.

The screenshot shows the Microsoft Azure Synapse Studio interface. On the left, there is a sidebar with icons for Home, Data (which is selected and highlighted with a red box), Develop, Integrate, Monitor, and Manage. The main area has a header with 'Microsoft Azure' and 'azuresynapse'. Below the header is a message about optional cookies. To the right of the message is a dropdown menu with 'Purview' and 'Workspace' options; 'Purview' is highlighted with a red box and a red arrow points to it. A search bar says 'Search your organization data'. The central part of the screen shows a 'Data' section with tabs for 'Workspace' (selected) and 'Linked'. It includes a 'Filter resources by name' input field and a list of resources: 'Lake database' (1 item) and 'SQL database' (2 items). To the right, there is a sidebar titled 'Recently accessed' which lists files: 'Movies.csv', 'AllMovies', 'ActorsClean.csv', and 'Actors.csv'. At the bottom of the sidebar is a 'View search results' link.

Search for the asset that exists in Purview. Imagine you're looking for movie files. Enter the keyword **movie** in the search bar, and fine tune your search by selecting **Files** as the object

type and Raw as the collection.

The screenshot shows the Microsoft Azure Synapse Studio interface. In the top navigation bar, 'Purview' is selected, and a search bar contains the query 'movie'. The left sidebar shows a 'Data' section with 'Workspace' selected. Under 'Object Type', 'Files' is checked. Under 'Collection', 'RAW' is checked. The main pane displays search results for 'movie', showing three items: 'Movies.csv' (Azure Blob Path), 'MovieActors.csv' (Azure Blob Path), and 'OnlineMovieMappings.csv' (Azure Blob Path). Each item has a small icon and a 'View details' link.

Select the first asset "Movies.csv" to get asset details. Because you are in Azure Synapse Studio, you can also leverage Azure Synapse capabilities.

The screenshot shows the Microsoft Azure Synapse Studio interface, specifically the asset details for 'Movies.csv'. The top navigation bar shows 'Purview' and the search bar still contains 'movie'. The left sidebar shows 'Data' with 'Workspace' selected. The main pane shows the asset details for 'Movies.csv', which is located in 'Azure Blob Storage | Blob'. Below the asset name are buttons for 'Edit', 'Select for bulk edit', 'Refresh', 'Delete', and 'Connect'. A 'Develop' dropdown menu is open, showing options: 'New SQL script', 'New notebook', and 'New data flow'. The 'Overview' tab is selected. Other tabs include 'Properties', 'Schema', 'Lineage', 'Contacts', and 'Related'. The 'Asset description' section states 'No description for this asset.' The 'Classifications' section states 'No classifications for this asset.' The 'Schema classifications' section states 'No classifications for this asset.' The 'Fully qualified name' section shows the path 'movie/Movies.csv'.

For instance, you can use Azure Synapse serverless to query your assets. Select **Develop**, **New SQL Script** and **Select top 100**.

The screenshot shows the Microsoft Azure Synapse Analytics Data workspace interface. On the left, there's a sidebar with icons for Home, Data, Workbooks, and Pipelines. The main area is titled 'Purview search' and shows search results for 'Movies'. It lists a single item: 'Movies.csv' from 'Azure Blob Storage | Blob'. Below this, there's a ribbon with 'Edit', 'Select for bulk edit', 'Refresh', 'Delete', 'Connect', and 'Develop' (which is currently selected). A context menu is open under 'Develop', showing options like 'New SQL script', 'Select top 100', 'New notebook', and 'New data flow'. Below the ribbon, tabs for 'Overview', 'Properties', 'Schema', 'Lineage', 'Contacts', and 'Related' are visible. Under 'Asset description', it says 'No description for this asset.' Under 'Classifications', it says 'No classifications for this asset.'

Double check you're connected to your serveless instance and select **Run** to execute the script and get an overview of your data.

This screenshot shows the same workspace environment as the previous one, but now with a SQL script being run. The script is as follows:

```
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://<storage-acct>.blob.core.windows.net/raw/Movies/Movies.csv',
        FORMAT = 'CSV',
        PARSE_VERSION = '2.0',
        HEADER_ROW = TRUE
    ) AS [result]
```

The 'Run' button is highlighted with a red box. Below the script, the 'Results' tab is selected, showing a table of movie data. The table has columns: MovieID, MovieTitle, Category, Rating, RunTimeMin, and ReleaseDate. The data is as follows:

MovieID	MovieTitle	Category	Rating	RunTimeMin	ReleaseDate
2f49d5af-b5c1-4b87-b426-0028...	Card Spark	Romance	R	103	02-13-2018
6260de1c-81bc-48b8-be49-007d...	Cube Lake	Science Fiction	PG-13	186	03-13-2018
3f0e40a9-330f-40a9-968d-008a6...	Brick Destiny	Science Fiction	R	182	02-13-2018
de31d122-4694-4c89-8e91-00b8...	Magical Dogs	Comedy	R	128	12-05-2017
fce022f4-135c-4c3c-8d9b-00d4d...	Western Data	Family	PG	97	11-28-2017
ca1eb6ca-9d55-41c9-a9ff-0100a...	Data Destiny	Science Fiction	R	167	08-08-2017
672b5a1c-f5b3-45a0-9919-011b...	Space Case	Romance	PG	94	08-15-2017

After reviewing data, you can use the asset, for example, adding to a new dataflow in Azure Synapse.

The screenshot shows two windows side-by-side. On the left is the Microsoft Purview search interface, displaying a file named 'Movies.csv' under 'Search results "movie" > Movies.csv'. On the right is the Azure Synapse Analytics 'Integrate' blade, specifically the 'Pipelines' section. A red arrow points from the 'New data flow' button in the Purview search interface towards the 'New data flow' button in the 'Integrate' blade.

(!) Note

See [Connect an Microsoft Purview Account](#) for detailed information about integrating Microsoft Purview into Azure Synapse Analytics.

Next unit: Knowledge check

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 200 XP



Knowledge check

3 minutes

Choose the best response for each of the questions below. Then select **Check your answers**.

Check your knowledge

1. What feature of Microsoft Purview can analysts and other data consumers use to find trustworthy data for reports? *

Data map.

Data catalog.

✓ Correct. Microsoft Purview data catalog helps users find trusted data sources by browsing and searching data assets across a data estate.

Data policies.

✗ Incorrect. Data access policies in Microsoft Purview enable you to manage access to different data systems.

2. If an analyst is looking for a specific asset by name and type, what is the most efficient way to find that asset in the data catalog? *

Browse assets.

Manage glossary.

✗ Incorrect. The business glossary is a definition of terms specific to a domain of knowledge that is commonly used, communicated, and shared in organizations as they are conducting business.

Search catalog.

✓ Correct. Searching the data catalog can help users quickly find assets they're looking for.

3. How can users download Power BI data source files that contain connections to assets discovered in Microsoft Purview? *



In the Microsoft Purview data catalog, in the asset view.

✓ Correct. There's a button in the asset view to open the asset in Power BI desktop.



In the Microsoft Purview insights report.



Users can't download Power BI data source files from Microsoft Purview.

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

3 minutes

The Microsoft Purview Data Catalog offers a browse experience that enables users to explore available data. Users can explore the data catalog either by collection or through traversing the hierarchy of each data source. The first step in understanding the contents of your data map is registering and scanning data, after which you can classify data for easy identification of assets to use for reporting.

Learning objectives

In this module, you will:

- Describe asset classification in Microsoft Purview.

Next unit: Register and scan data

[Continue >](#)

How are we doing?

100 XP

Register and scan data

10 minutes

Registration and scanning of data enables discoverability of data across an estate.

Before you can register and scan data, it's important to understand the concept of collections. In Microsoft Purview Data Catalog, collections are key concept because they drive permissions and asset protection. Collections are also used to understand data estate health and catalog usage and adoption, as featured in the data stewardship section of your [Data Estate Insights](#).

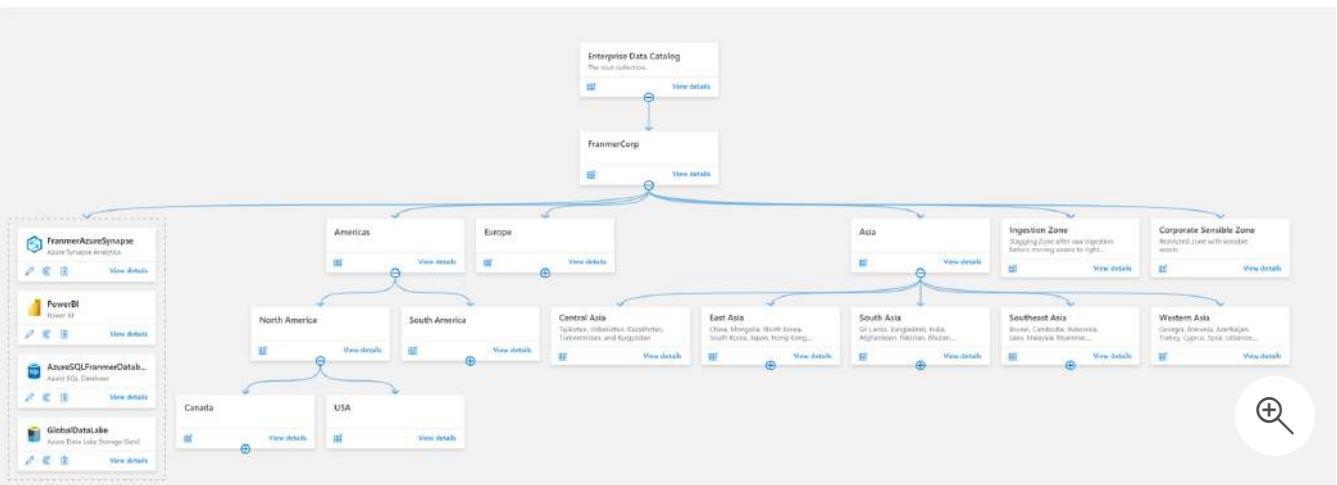
Collections

The data map is at the core of Microsoft Purview, which keeps an up-to-date map of assets and their metadata across your data estate. To hydrate the data map, you need to register and scan your data sources, which is done at the collection level. Collections support organizational mapping of metadata. By using collections, you can manage and maintain data sources, scans, and assets in a hierarchy instead of a flat structure. Collections allow you to build a custom hierarchical model of your data landscape based on how your organization plans to use Microsoft Purview to govern your landscape.

Collections also provide a security boundary for your metadata in the data map. Access to collections, data sources, and metadata is set up and maintained based on the collection's hierarchy in Microsoft Purview, following a least-privilege model:

- Users have the minimum amount of access they need to do their jobs.
- Users don't have access to sensitive data that they don't need.

Data sources are registered at the collection level. Scan results can then be sent to this collection or a sub collection. The image below displays the structure of a collection.



Tip

Learn more about [Microsoft Purview collections architectures and best practices](#).

Register and scan data sources

Data governance use begins at collection level, with the registration of data sources in Microsoft Purview governance portal. Microsoft Purview supports an array of data sources. Data teams (analysts, engineers, and scientists) may not be actively registering and scanning data in Microsoft Purview, but it's critical that data consumers understand governance efforts. Registering and scanning assets requires **Data Curator** permissions.

Important

Data registered and scanned in Microsoft Purview only collects metadata information. Data remains in its location and isn't migrated to any other platform.

Register a data source

Registering a data source is done from within the Azure portal. Once you have a Microsoft Purview service configured in Azure, you use the Microsoft Purview governance portal to register your data sources.

To register a data source, you'll select the icon to register a data source as displayed in the image below. Selecting this icon will give you access to all data source connectors.

Below is a small sample of available connectors in Microsoft Purview Data Catalog. See [supported data sources and file types](#) for an up-to-date list of supported data sources and connectors.

The screenshot shows the Microsoft Purview Enterprise Data Catalog interface. On the left, there's a tree view of collections: 'Enterprise Data Catalog' (root collection) and 'FranmerCorp'. Under 'FranmerCorp', there's a '+ Add' button and a 'View details' link. On the right, a grid of tiles displays various Azure services as data sources:

Azure Multiple	Azure Synapse Analytics	Azure Blob Storage
Azure Cosmos DB (SQL API)	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2	Azure Database for MySQL	Azure Database for PostgreSQL
Azure Dedicated SQL Pool (formerly SQL DW)	Azure Files	Azure SQL Database
Azure SQL Managed Instance	SQL Server on Azure Arc-enabled servers	

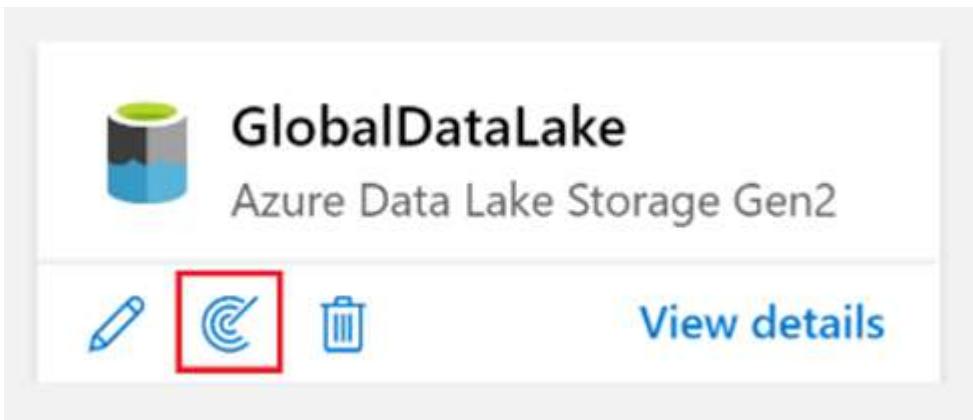
Registering a data source is straightforward, you need to complete the required fields. Authentication will be done during the scanning phase.

Each type of data source you choose will require specific information to complete the registration. For example, if your data sources reside in your Azure subscription, you'll choose the necessary subscription and storage account name.

Scan a data source

Once you have data sources registered in the Microsoft Purview governance portal and displayed in the data map, you can set up scanning. The scanning process can be triggered to run immediately or can be scheduled to run on a periodic basis to keep your Microsoft Purview account up to date.

Scanning assets is as simple as selecting **New scan** from the resource as displayed in the data map.



You'll now need to configure your scan and assign the following details:

- Assign a friendly name.
- Define which [integration runtime](#) to use to perform the scan.
- [Create credentials](#) to authenticate to your registered data sources.
- Choose a collection to send scan results.

After the basic configuration, you'll *scope* your scan, which allows you to choose just a specific zone of your data source. For instance, if you have a collection called "Raw" in your data map, you can define the scope to scan only the raw container of your data lake.

After configuring and scoping your scan, you'll define the *scan rule set*. A scan rule set is a container for grouping a set of scan rules together so that you can easily associate them with a scan. For example, you might create a default scan rule set for each of your data source types, and then use these scan rule sets by default for all scans within your company. You might also want users with the right permissions to create other scan rule sets with different configurations based on business need.

Once a scan is complete, you can refer to the scan details to view information about the number of scans completed, assets detected, assets classified, Scan information. It's a good place to monitor scan progress, including success or failure.

The screenshot shows the 'GlobalDataLake' source in the Purview Data Catalog overview page. It displays metrics like 4 scans, 108 discovered assets, and 32 classified assets. It also shows the source was registered on 05/02/2022 at 4:49:38 PM, belongs to the 'Enterprise Data Catalog' collection path under 'FranmerCorp', and has a source hierarchy for 'Microsoft Azure Sponsorship 2' and 'Purview_Demo' resource group. A search bar is also visible.



Tip

Refer to [Scanning best practices](#) for more information on scanning assets.

Roles and permissions

Permissions in Microsoft Purview are assigned at **collection** level. Collections are used to organize assets and sources and can be thought of as a logical grouping of data assets.

Data teams looking to discover and use data need to be assigned the **Data Reader** role in a collection in Microsoft Purview. The Data Reader role enables users to find assets, but doesn't enable users to edit anything. The **Data Curator** role is required to edit information about assets, assign classifications, and associate assets with glossary entries. To set up scans via the Microsoft Purview Governance Portal, individuals need to be either a data curator on the collection or data curator and data source administrator where the source is registered.

When a Microsoft Purview account is created, it starts with a root collection that has the same name as the Microsoft Purview account itself. The creator of the Microsoft Purview account is automatically added as a Collection Admin, who can then assign Data Source Admin, Data Curator, and Data Reader on this root collection, and can edit and manage this collection.



Tip

Learn more about [Microsoft Purview permissions and access](#).

Next unit: Classify and label data

[Continue >](#)

How are we doing?

100 XP

Classify and label data

6 minutes

Glossary terms, classifications and labels are all annotations to a data asset. Each of them have a different meaning in the context of the data catalog.

What is data classification?

Classifications are annotations that can be assigned to entities. The flexibility of classifications enables you to use them for multiple scenarios such as:

- understanding the nature of data stored in the data assets
- defining access control policies

Classification is based on the business context of the data. For example, you might classify assets by Passport Number, Driver's License Number, Credit Card Number, SWIFT Code, Person's Name, and so on. Microsoft Purview has more than 200 system classifiers today. Users can also define their own classifiers in the data catalog. As part of the scanning process, classifications are automatically detected and applied as metadata within the Purview Data Catalog.

Classification rules

In Microsoft Purview, you can apply system or custom classifications on a file, table, or column asset. Microsoft Purview makes use of Regex patterns and bloom filters to classify data. These classifications are then associated with the metadata discovered in the Azure Purview Data Catalog.

Metadata is used to help describe the data that is being scanned and made available in the catalog. During the configuration of a scan set, you can specify classification rules to apply during the scan that will also serve as metadata. The existing classification rules fall under five major categories:

- Government - covers attributes such as government identity cards, driver license numbers, passport numbers, etc.
- Financial - covers attributes such as bank account numbers or credit card numbers.
- Personal - personal information such as a person's age, date of birth, email address, phone number, etc.

- Security - attributes like passwords that may be stored.
- Miscellaneous - attributes not covered in the other categories.

Why classify data?

A good data governance strategy includes a process to classify data to understand its level of confidentiality, determine if the data source is compliant with various regulations, or how long to retain it for. Classification in Microsoft Purview makes data assets easier to understand, search, and govern. Classification can also help you implement measures to protect sensitive data.

Once a classification is tagged to a data source after a scan, you can generate reports and insights to gain a stronger understanding of your data estate. Because classification is based on the business context of the data, it can help bridge the gap between the business and the data team.

Data classification: system vs. custom classification

Microsoft Purview supports both system and custom classifications. There are over +200 system classifications available in Microsoft Purview today. Data teams need to know that if necessary classifications aren't available out of the box, they can work with the data stewards to create custom classifications, to meet their own organizational data governance requirements.

Important

For the entire list of available system classifications, see [Supported classifications in Microsoft Purview](#).

Who creates custom classifications?

Purview Data Curators can create, update, and delete custom classifiers and classification rules. Purview Data Readers can only view classifiers and classification rules.

In practical terms, Data Curators may not be members of the data team. It is however critical that data team members understand classification to be able to successfully work together and govern data across an organization.

What are data labels?

The Microsoft Purview Data Map supports labeling structured and unstructured data stored across various data sources. This may sound familiar to you from other Microsoft technologies - and may be known as sensitivity labels. The data map extends the use of sensitivity labels from Microsoft Purview Information Protection to assets stored in infrastructure cloud locations and structured data sources.

Labels are defined in Microsoft Purview Information Protection, and you can extend the application to Microsoft Purview Data Catalog.

The screenshot below shows both data classification and label in the Microsoft Purview Data Catalog. You can see that this Azure SQL table has a column called "CreditCard":

- Classified as "Credit Card Number" because scan detected numbers corresponding to credit card pattern rules.
- Labeled as "Confidential – Finance" because credit card number was defined in your organization as confidential information (and this label brings encryption).

The screenshot shows the Microsoft Purview Data Catalog interface for an Azure SQL table named "ReportTable". The table has 19 items. The "Schema" tab is selected. A red box highlights the "CardNumber" column under the "Franmer_Card" row. The "Classifications" column for "CardNumber" contains a button labeled "Credit Card Number". The "Sensitivity label" column for "CardNumber" contains a button labeled "Confidential - Finance". Other columns like "CardType", "CardVerification", "Country", and "CountryName" have empty classification and sensitivity label fields.

Column name	Classifications	Sensitivity label
Franmer_Card		
CardNumber	Credit Card Number	Confidential - Finance
CardType		
CardVerification		
Country	Country/Region	
CountryName	Country/Region	
ExpiryMonth		

Next unit: Search the data catalog

100 XP



Search the data catalog

3 minutes

A data catalog search can empower business and data analysts to find and interpret data. The data catalog provides intelligent recommendations based on data relationships, business context, and search history. The Purview data catalog can assist data teams by adding business context to assets to drive analytics, AI and ML initiatives.

The data catalog can be searched by keyword, object type, collection, classification, contact, label, or assigned term. Results can then be sorted by relevance or name.

The screenshot shows the 'Data catalog > Search results for Bing' interface. On the left, there are several filter panels with red arrows pointing to them:

- Filter by keyword:** A text input field.
- Object Type:** A dropdown menu with options like Dashboards, Data pipelines, Files, Folders, Glossary terms, Reports, Stored procedures, and Tables.
- Collection:** A dropdown menu with options like Contoso (Purview4All).
- Classification:** A dropdown menu with options like Country/Region (2) and World Cities (1).
- Contact:** A dropdown menu with options like Paul (3).
- Label:** A dropdown menu with options like Confidential (2).
- Assigned term:** A dropdown menu with options like Contoso Child (1).

At the top, there are filters for 'Source type : all', 'Instance : all', and a 'Clear all filters' button. To the right, it says 'Showing 1-3 out of 3 results' and 'Sort by: Relevance'. The results list three datasets:

- QueriesByCountry**: Azure Data Lake Storage Gen2 Resource Set. Curated from Bing search logs. Last updated 3 months ago.
- QueriesByState**: Azure Data Lake Storage Gen2 Resource Set. Curated from Bing search logs. Last updated 3 months ago.
- BingCoronavirusQuerySet**: Azure Data Lake Storage Gen2 Path. Last updated 21 days ago.

✓ 200 XP



Knowledge check

3 minutes

Choose the best response for each of the questions below. Then select **Check your answers**.

Check your knowledge

1. What level are user permissions set at in Microsoft Purview? *

Tenant.

X Incorrect. Permissions aren't set at tenant level in Microsoft Purview.

Data catalog.

Collection.

✓ Correct. User permissions are set at the collection level in Microsoft Purview.

2. What are the two types of classification in Microsoft Purview? *

System classifications and custom classifications.

✓ Correct. Microsoft Purview contains out of the box, system classifications and custom classifications.

Microsoft Information Protection Sensitivity Labels and system classifications.

X Incorrect. Microsoft Information Protection Sensitivity Labels can be applied in Microsoft Purview, but they aren't a type of classification.

Custom classifications and user-defined classifications.

3. If a data analyst is looking for a specific resource for reporting, what should they use? *

Purview Data Catalog to search.

✓ Correct. Searching the Purview Data Catalog will help analysts find resources.

The business glossary.

-
- Import into Power BI and create a custom report.
-

Next unit: Summary

[Continue >](#)

How are we doing?     

100 XP

Introduction

3 minutes

As the landscape of enterprise data continues to grow, it's critical to get an accurate view of your organization's data. Microsoft Purview and Power BI integration enables you to scan your entire Power BI tenant to search and browse Power BI assets, explore enhanced dataset metadata, trace end-to-end data lineage, and drill-down into datasets in Power BI for further analysis.

Learning objectives

In this module, you will:

- Register and scan a Power BI tenant.
- Use the search and browse functions to find data assets.
- Describe the schema details and data lineage tracing of Power BI data assets.

Next unit: Register and scan a Power BI tenant

[Continue >](#)

How are we doing?

✓ 100 XP



Register and scan a Power BI tenant

4 minutes

To get an understanding of what is going on in your Power BI tenant, you can perform a full scan in Microsoft Purview to view the schema and lineage of assets across all workspaces. After, you can schedule incremental scans on workspaces that have changed since the previous scan.

There are a few pre-requisite steps required to scan your Power BI tenant in Microsoft Purview.

💡 Tip

If you need to create a Microsoft Purview account, see the [quickstart guide](#) to create a Microsoft Purview account in the Azure Portal.

Establish a connection between Microsoft Purview and Power BI

Microsoft Purview can connect to and scan Power BI either in the same tenant or across tenants. You'll need to set up authentication either by using a Managed Identity or a Delegated Authentication.

❗ Note

See [Register and scan a Power BI tenant](#) to learn more about the set-up and authentication of Power BI connections in same and cross-tenant scenarios.

Authenticate to Power BI tenant

Give Microsoft Purview permissions to access your Power BI tenant.

If you're using **Managed Identity** to authenticate to Power BI, you'll need to create a security group in Azure Active Directory, and add your Microsoft Purview managed identity to this security group.



New Group ...

Got feedback?

Group type * ⓘ

Security ✓

Group name * ⓘ

Purview ✓

Group description ⓘ

Enter a description for the group

Azure AD roles can be assigned to the group ⓘ

Yes

No

Membership type * ⓘ

Assigned ✓

Owners

1 owner selected

Members

1 member selected

If a security group containing the Purview managed identity already exists, you can proceed to configuring the Power BI tenant.

Configure Power BI tenant

Next you need to enable access to Power BI by Microsoft Purview in Power BI itself. This is done by enabling *Allow service principals to use read-only Power BI admin APIs* in the Power BI admin portal.

Admin API settings

Allow service principals to use read-only Power BI admin APIs

Unapplied changes

Web apps registered in Azure Active Directory (Azure AD) will use an assigned service principal to access read-only Power BI Admin APIs without a signed in user. To allow an app to use service principal authentication, its service principal must be included in an allowed security group. By including the service principal in the allowed security group, you're giving the service principal read-only access to all the information available through Power BI admin APIs (current and future). For example, Power BI user names and emails, dataset and report detailed metadata. [Learn more](#)

Enabled

Apply to:

The entire organization

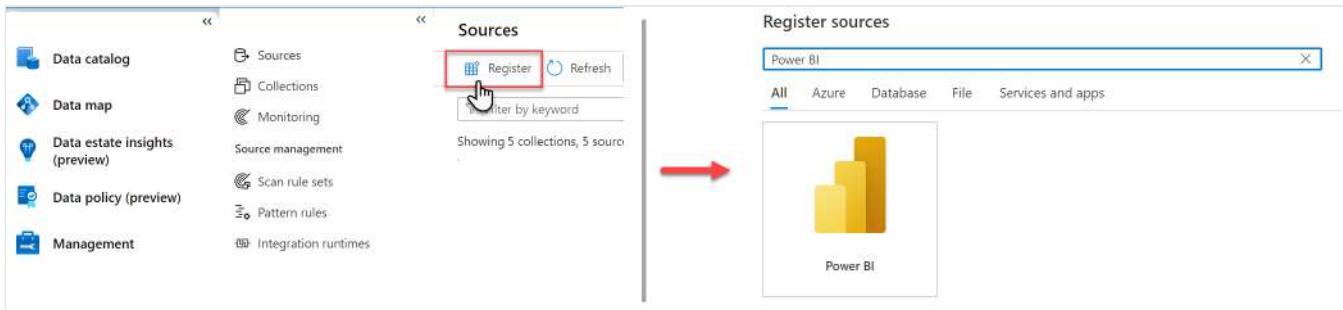
Specific security groups

Purview Enter security groups

✓ Applying changes
Tenant settings changes will be applied within the next 15 minutes.

Register and scan Power BI

Now that you've got access set up in both Microsoft Purview and Power BI, you can register and scan your Power BI tenant.



After registering the Power BI tenant, initiate the scan by selecting **New scan**. Give your scan a name and step through the interface, where you'll be able to exclude personal workspaces, confirm integration runtime and credentials, and select a collection. Test the connection to ensure authentication is set up properly.

Scan "PowerBI-Nod"

Name *
Scan-1

Personal workspaces *
 Include Exclude

Changes to the scan configuration will reset the upcoming scan to be a full scan for this data source. [Learn more](#)

Connect via integration runtime * ⓘ
Azure AutoResolveIntegrationRuntime

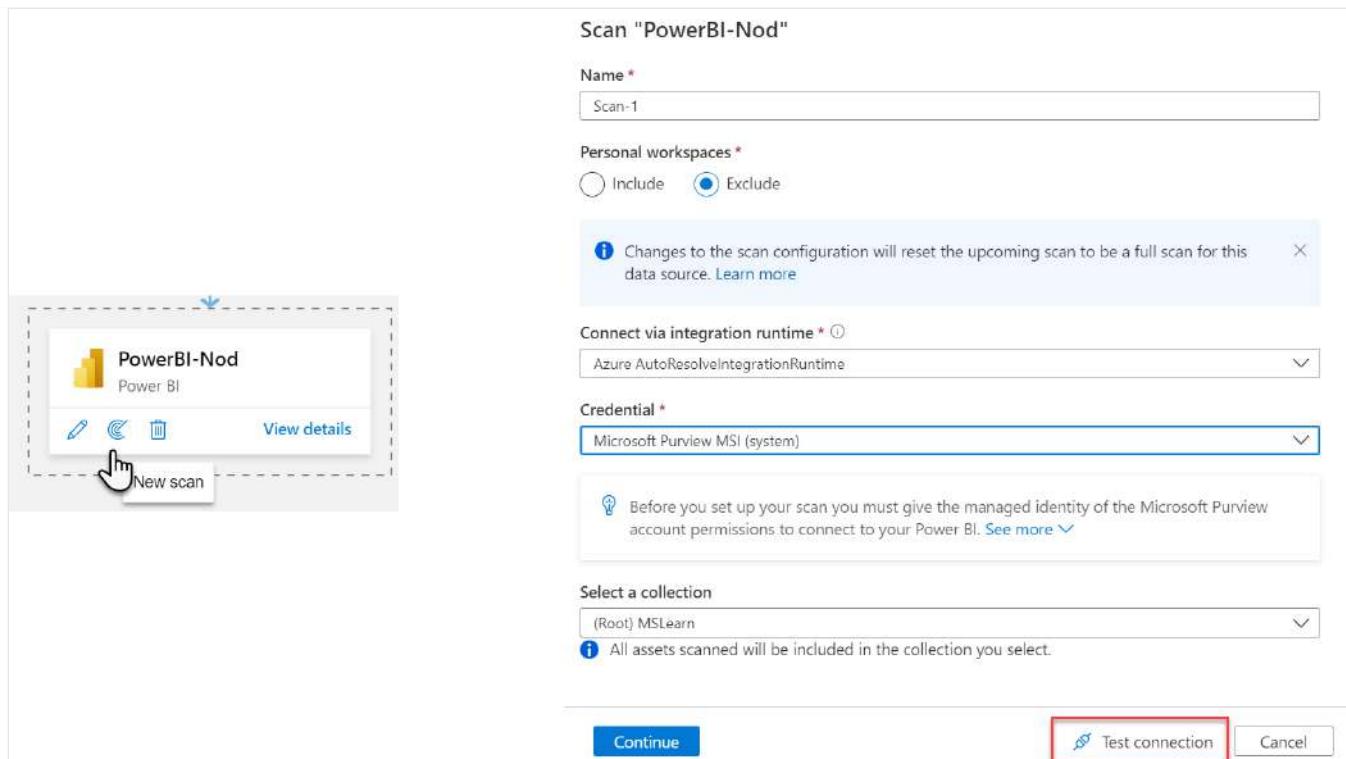
Credential *
Microsoft Purview MSI (system)

Before you set up your scan you must give the managed identity of the Microsoft Purview account permissions to connect to your Power BI. [See more](#)

Select a collection
(Root) MSLearn

All assets scanned will be included in the collection you select.

[Continue](#) [Test connection](#) [Cancel](#)



! Note

If you're performing the scan, you must be both a Data Source Administrator and a Data Reader. See **Access control in the Microsoft Purview Data Map** for details on permissions.

You're able to track the progress of the scan in the data map, and once the scan is complete, you'll be able to search and browse the contents of your entire Power BI tenant!

PowerBI-4Learn



New scan Edit source Delete source Refresh

Overview Scans

Source ID: <https://app.powerbi.com/>

Scans

1



Discovered assets

1



Classified assets

0



Recent scans

Scan name	Last run status	Scan rule set
Scan-C4B	In progress	-
→ See all applied scans		

If you're having any issues with scanning your Power BI tenant, see [Troubleshoot Power BI tenant scans in Microsoft Purview](#) for details and helpful hints.

Next unit: Search and browse Power BI assets

[Continue >](#)

How are we doing?

✓ 100 XP ➔

Search and browse Power BI assets

4 minutes

After data is registered and scanned, analysts and data consumers need to be able to find data, view enhanced metadata, and track data lineage. Search and browse in the Purview Data Catalog enables you to quickly find trustworthy data.

After scanning your Power BI tenant, you'll see those assets appear in the search results, including underlying data sources.

Search the Microsoft Purview Data Catalog

From the Microsoft Purview Governance Portal, you can type relevant keywords to start discovering assets. In this scenario, you're looking for "sales."

The screenshot shows the Microsoft Purview Data Catalog interface with the title "Purview4All" at the top. Below the title, there are three statistics: "6 sources", "139 assets", and "58 glossary terms". A search bar contains the keyword "sales". Under the search bar, a section titled "Your recent searches" lists "(⌚) sales". Below this, a section titled "Search suggestions" lists several items: "SalesLT SalesOrderDetail", "SalesLT vGetAllCategories", "SalesLT ProductModelProductDescription", and "SalesPerson".

The screenshot below displays the search result, with all assets corresponding to the keywords entered in the search engine. Notice the appearance of Power BI assets.

The screenshot shows the Microsoft Purview Data Catalog interface. On the left, there's a sidebar with filters for Source type (all), Instance (all), and a 'Clear all filters' button. Below these are sections for Object Type (Dashboards, Data pipelines, Files, Folders, Reports, Stored procedures, Tables), Collection (Sales, Contoso Purview Catalog, IT), and Classification (Franmer Product, Credit Card Number, Country/Region, Person's Name). The main area displays search results for 'product sales'. It shows 1-25 out of 153 results, sorted by Relevance. The results include:

- Product Mix.csv** (Azure Blob Path)
- Sales Dashboard-SQLDB** (Secret, Power BI Dataset)
- SalesWithCustomerID.csv** (Azure Blob Path)
- SalesWithCustomerID** (certified, Azure SQL Table)
Table with sales raw information.
- ReportTable_Prod** (certified, Azure SQL Table)
Report Table in production environment to create Power BI report for sales people. Ta...

You can fine-tune your search using the filters on the left side of the page. Filters available include source type, keyword, object type, collection, classification, contact, label, and glossary term.

Search results for sales

Source type : all Instance : all

Filter by keyword

Object Type

- Dashboards
- Data pipelines
- Files
- Folders
- Glossary terms
- Reports
- Stored procedures
- Tables

Collection

- Finance Purview4All (3)
- Purview4All (3)

Classification

Contact

Label

Assigned term

Browse the Microsoft Purview Data Catalog

Searching for specific assets is great if you know what you're looking for, but analysts and data consumers may not know exactly how their data estate is structured. The browse experience enables you to explore what data is available, either by collection or through traversing the hierarchy of each data source in the catalog.

To access the browse experience, select **Browse assets** from the governance portal home page.

Purview4All

The dashboard shows a summary of data assets: 5 sources, 139 assets, and 58 glossary terms. It includes a search bar labeled "Search catalog" and three main cards:

- Browse assets**: Explore assets by source type and collection. This card is highlighted with a red border.
- Manage glossary**: Create and edit terms with custom term templates.
- Knowledge center**: Discover learning materials and tutorials.

You can browse the data catalog either by collection or by source type, depending on your needs. Browsing by either collection or source type allows you to see assets you have access to. Once you find the asset you're looking for, you can select it to see details on schema, lineage, and a detailed classification list.

Browse assets

Showing 5 collections

Name	Description	Assets
Purview4All	The root collection.	12
Contoso		14
Finance	Logical collection to store Finance-related data	18
HR		67

Browse assets

Custom source types

Uniquely, browsing by source type allows you to see the hierarchies of data sources using an explorer view. This is a helpful and familiar way to navigate to see lists of scanned assets.

! Note

Assets in Purview are organized by collection and permissions are granted at collection level. Both searching and browsing require data reader permissions. See [Access control in the Microsoft Purview Data Map](#) for details on permissions.

Select an asset to see details about the properties, schema, lineage, contacts, and related assets.

The screenshot shows the Microsoft Purview Data Catalog interface. At the top, it says "Data catalog > Browse assets >". Below that, there's a card for a "Customer" Azure SQL Table. The card includes icons for edit, bulk edit, request access, refresh, and delete. Below the card, there are tabs for Overview (which is selected), Properties, Schema, Lineage, Contacts, and Related. The Overview tab contains sections for Asset description, Classifications, Schema classifications, Fully qualified name, Collection path, Hierarchy, and Glossary terms. The Asset description section says "No description for this asset." The Classifications section says "No classifications for this asset." The Schema classifications section lists three items: "Email Address", "Person's Name", and "U.S. Phone Number". The Fully qualified name section shows "mssql://". The Collection path section shows a tree structure: Purview4All > Finance. The Hierarchy section shows a tree structure: pvlab > Azure SQL Server > pvlab > Azure SQL Database > SalesLT > Azure SQL Schema > Customer > Azure SQL Table. The Glossary terms section says "No glossary terms for this asset."

Next unit: View Power BI metadata and lineage

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 100 XP



View Power BI metadata and lineage

4 minutes

Purview and Power BI together are powerful, enhancing the ability of the search and browse features to see both the schema and lineage of Power BI assets.

Extend your search with enhanced metadata

Metadata scanning facilitates governance by making it possible to catalog and report on the metadata of your organization's Power BI artifacts. The results of metadata scanning are displayed on the schema tab of the asset.

ⓘ Note

Metadata scanning must be enabled in the Power BI admin portal. See [Set up metadata scanning in your organization](#) to learn more.

After performing a search in the Purview Governance Portal, select a Power BI asset from your search result to see the sensitivity labels and endorsement metadata. Additional business metadata includes the dataset user configuration, create datetime, and description.

Under the Schema tab, you can see the list of all the tables, columns, and measures created inside the Power BI dataset.

The screenshot shows the 'Schema' tab of the Power BI Opportunity Report. The table lists various fields with their details:

Field name	Classifications	Sensitivity label	Glossary terms	Data type	Asset description
CountryRegionBudget			Sales without taxes	table	Budget per country without taxes
Actual Revenue			Sales without taxes	Double	Actual revenue
Actual Revenue MoM%					Actual Revenue MoM%
Country	Country/Region			String	Country name
Growth Year on Year			Sales without taxes	String	Growth Year on Year
Multiplier				Double	Accelerator
Region	Country/Region			String	Region name
Target Revenue			Sales without taxes	Int64	Target revenue without taxes
Variance				Double	Variance
Variance to Target Sales with Disc...					Variance to Target Sales with Disc...
DimAccount				table	Account name
DimAccountOwner	Person's Name			table	Account owner name
DimDate				table	Date
Discount %				table	Discount in percent
Margin			Sales without taxes; Sales	table	Margin
prmAnalytic				table	Power BI field parameter
Product Mix	Franmer Product			table	Product mix
Actual Mix (\$)				Double	
Product	Franmer Product		Product, Customer product, Product client	String	
Product Category	Franmer Product		Customer product, Product client, Product	String	
Total Mix (\$)				Int64	

For more detail, selecting a particular field within the schema tab will take you to the details for that field. You can then view the overview, properties, lineage, contacts, and related assets for that particular field.

Power BI Column

Edit Select for bulk edit Refresh Delete

Overview Properties Contacts

Filter by property key Show properties without a value

Showing 6 of 12 properties

Properties

dataType	String
displayName	Variance
isHidden	false
isMeasure	false
qualifiedName	https://app.powerbi.com/

Related assets

Table	dimAccount
-------	----------------------------

Metadata scanning requires no special license. It works for all of your tenant metadata, including items that are located in non-Premium workspaces.

If you'd like more information about assets, you also have the option open the Power BI dataset in the Power BI service for further analytics, root-cause investigation, impact analysis, management tasks, and dataset enrichment.

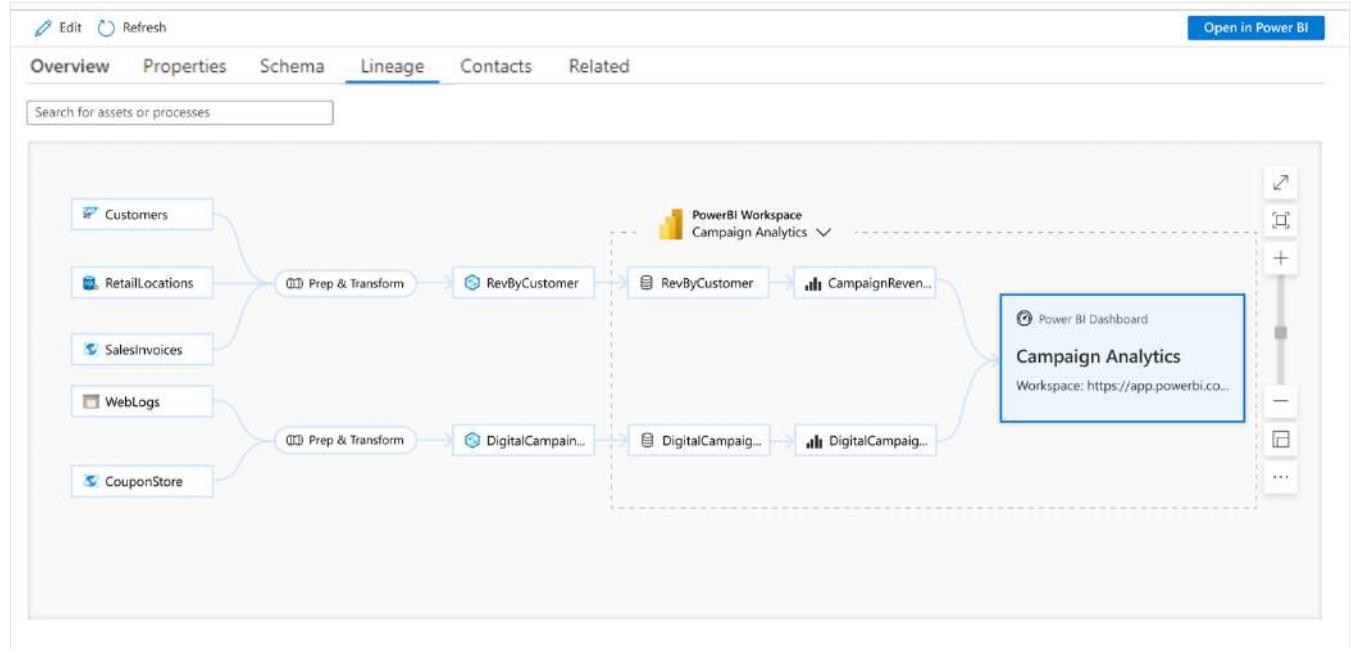
Extend your search with lineage

If you're using the search and browse features in Microsoft Purview to find assets for reporting or to troubleshoot existing assets, you likely need more information on where data actually comes from, and what transformation steps it has undergone. The lineage view displays the flow of data from the source through to Power BI assets, including dataflows, datasets, reports, and dashboards.

Although you can track [data lineage in Power BI](#), this information is limited to the items in a single workspace. Lineage in Purview enables you to view the movement of data across more than one workspace, in a single view.

Lineage enables easy troubleshooting and deeper analysis of analytics projects. You're able to look both up and down-stream, to perform either root cause or impact analysis.

For example, you can detect the Azure Synapse Analytics pipeline that is responsible for the transformation of the data upstream of Power BI.



In the Purview Governance Portal, lineage is displayed from the asset you're currently viewing.

Next unit: Knowledge check

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

✓ 200 XP ➔

Knowledge check

3 minutes

Choose the best response for each of the questions below. Then select **Check your answers**.

Check your knowledge

1. What are the prerequisite steps to register and scan a Power BI tenant in Microsoft Purview? *

- Set up authentication between Purview and Power BI, and configure the Power BI tenant.

✓ **Correct. Authentication must be established using either Managed Identity or Delegated Authentication, and the Power BI tenant must be configured to allow service principles to use the read-only API.**

- Set up and deploy a Power BI data gateway.

- Configure the Power BI tenant only.

2. What steps are required in the Power BI admin portal to enable the scanning and display of enhanced metadata? *

- There are no other steps required. Enhanced metadata displays by default.

✗ **Incorrect. There is a required setting in the Power BI admin portal that needs to be enabled.**

- Enable enhanced metadata scanning in the Azure portal.

- Enable an admin API setting in the Power BI admin portal.

✓ **Correct. Enabling the admin API settings will enable enhanced metadata scanning.**

3. What details of an asset would be helpful in performing an impact analysis? *

- Lineage.

✓ Correct. Viewing lineage details will help users perform an impact analysis.



Properties.

✗ Incorrect. The properties of the asset aren't sufficient to perform an impact analysis.



Schema.

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

1 minute

Microsoft Purview is a cloud service that provides the basis of a *data governance* solution in which you can catalog, classify, and track data assets across a large-scale data estate.

Azure Synapse Analytics is a cloud-scale data analytics suite that supports data ingestion and transformation, distributed big data processing and exploration with SQL and Spark, and enterprise data warehousing.

When combined, Microsoft Purview and Azure Synapse Analytics can be used to create a comprehensive solution for reliable, massively scalable data analytics with rich data asset discovery and lineage tracking capabilities.

In this module you'll learn how to:

- Catalog Azure Synapse Analytics database assets in Microsoft Purview.
- Configure Microsoft Purview integration in Azure Synapse Analytics.
- Search the Microsoft Purview catalog from Synapse Studio.
- Track data lineage in Azure Synapse Analytics pipelines activities.

Next unit: Catalog Azure Synapse Analytics data assets in Microsoft Purview

[Continue >](#)

How are we doing?

100 XP



Catalog Azure Synapse Analytics data assets in Microsoft Purview

8 minutes

Azure Synapse Analytics is a platform for cloud-scale analytics workloads that process data in multiple sources; including:

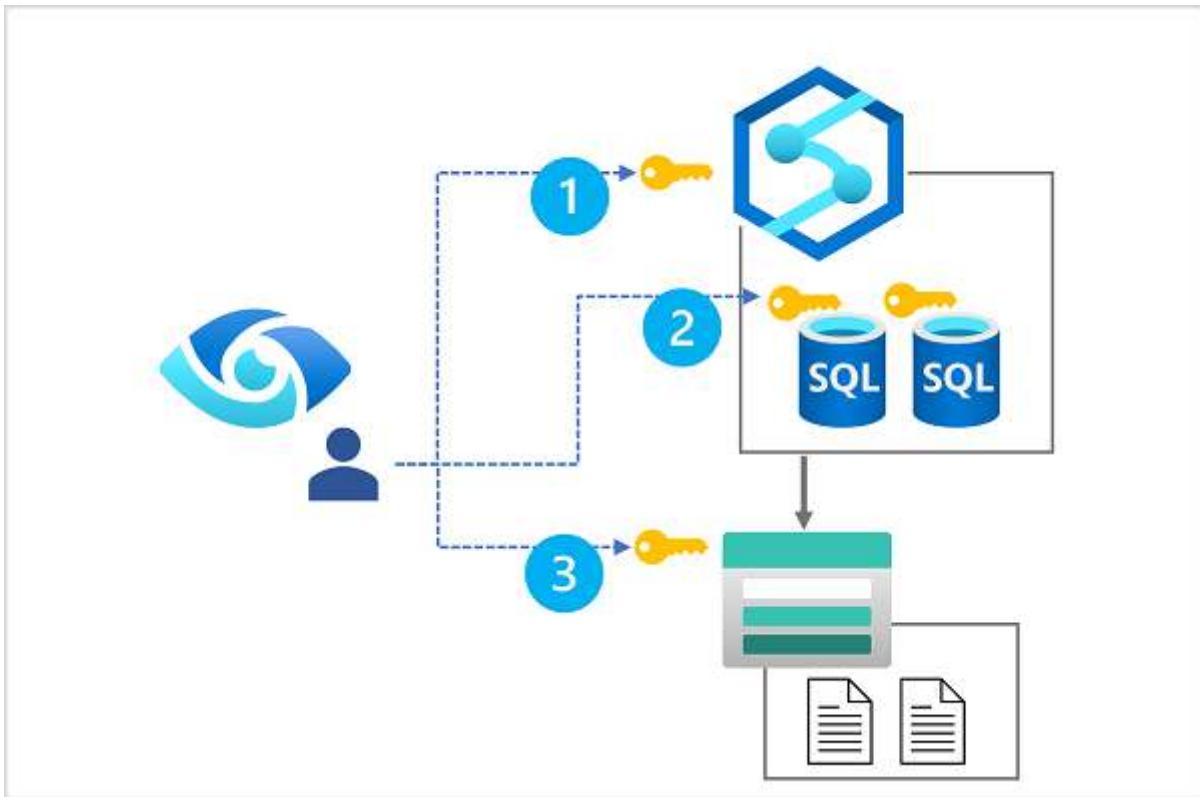
- Relational databases in serverless and dedicated SQL pools
- Files in Azure Data Lake Storage Gen2

A comprehensive data analytics solution can include many folders and files in a data lake, and multiple databases that each contain many tables, each with multiple fields. For a data analyst, finding and understanding the data assets associated with a Synapse Analytics workspace can present a significant challenge before any analysis or reporting can even begin.

Microsoft Purview can help in this scenario by cataloging the data assets in a *data map*, and enabling data stewards to add metadata, categorization, subject matter contact details, and other information that helps data analysts identify and understand data.

Configure data access for Microsoft Purview

In order to scan the data assets in the data lake storage and databases used in your Azure Synapse Workspace, Microsoft Purview must have appropriate permissions to read the data. In practice, this means that the account used by your Microsoft Purview account (usually a system-assigned managed identity that is created when Microsoft Purview is provisioned) needs to be a member of the appropriate role-based access control (RBAC) and database roles.



The diagram shows that Microsoft Purview requires role membership that permits the following access:

1. Read access to the Azure Synapse workspace (achieved through membership of the **Reader** role for the Azure Synapse Workspace resource in the Azure subscription).
2. Read access to each SQL database that will be scanned (achieved through membership of the **db_datareader** fixed database role in each database).
3. Read access to data lake storage (achieved through membership of the **Storage Blob Data Reader** role for the Azure Storage account hosting the Azure Data Lake Storage Gen2 container for the data lake).

Tip

Learn more:

- For more information about RBAC in Microsoft Azure, see [What is Azure role-based access control \(Azure RBAC\)?](#)
- For more information about database-level roles in Azure Synapse Analytics SQL pools, see [Database-level roles](#).

You'll get a chance to assign RBAC and SQL database role membership to support Microsoft Purview data access for yourself in the exercise later in this module.

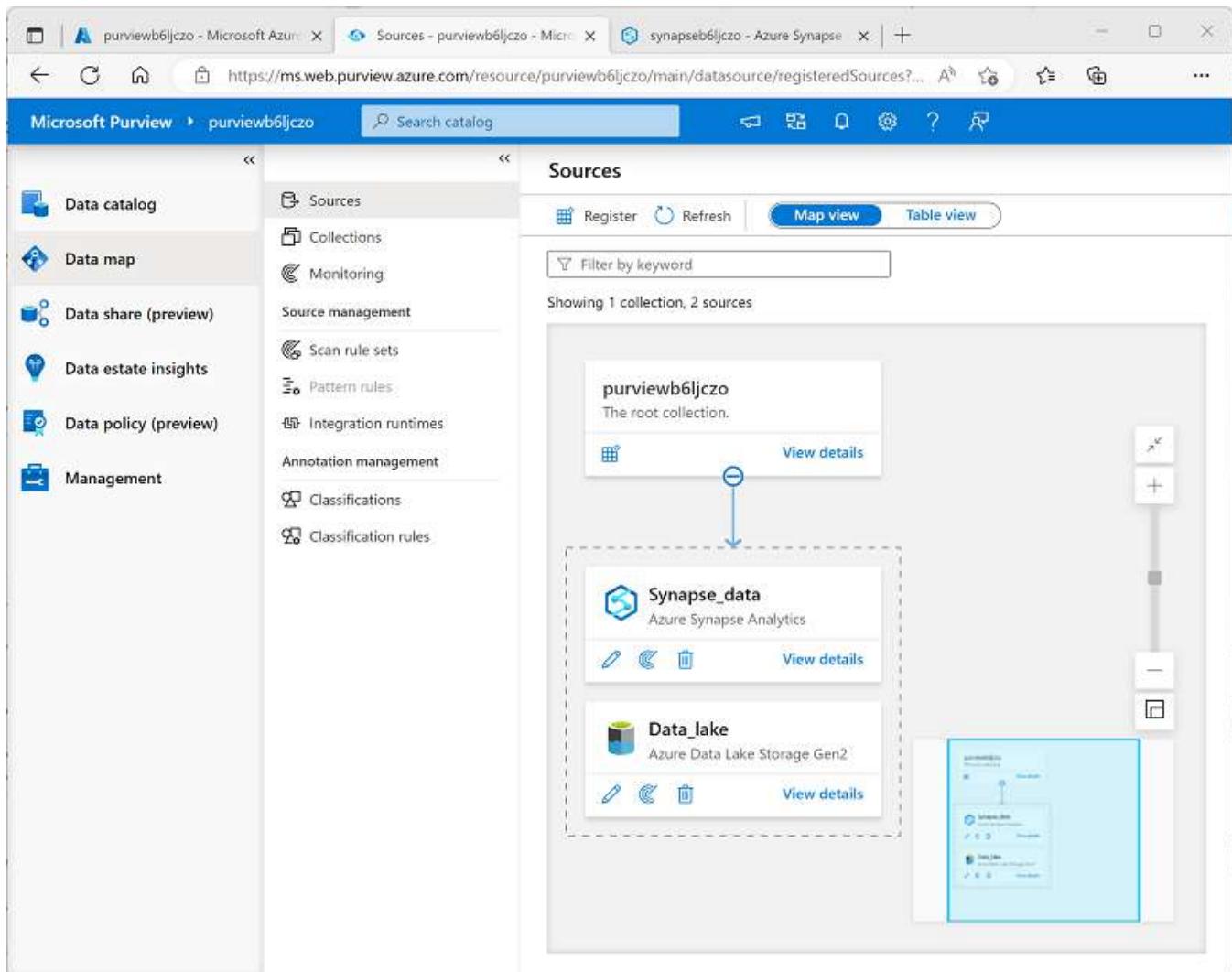
Register and scan data sources

Microsoft Purview supports the creation of a *data map* that catalogs data assets in *collections* by scanning registered *sources*. Collections form a hierarchy of logical groupings of related data assets, under a root collection that is created when you provision a Microsoft Purview account. You can use the Microsoft Purview Governance Portal to create and manage collections in your account.

To include assets from a particular data source, you need to register the source in a collection. Microsoft Purview supports many kinds of source, including:

- **Azure Synapse Analytics** - One or more SQL databases in a Synapse Analytics workspace.
- **Azure Data Lake Storage Gen2** - Blob containers used to host folders and files in a data lake.

To catalog assets used in an Azure Synapse Analytics workspace, you can register one or both of these sources in a collection, as shown here:



After registering the sources where your data assets are stored, you can scan each source to catalog the assets it contains. You can scan each source interactively, and you can schedule

period scans to keep the data map up to date.

Tip

To learn more about registering and scanning sources, see [Scans and ingestion in Microsoft Purview](#).

You'll get a chance to register and scan sources for an Azure Synapse Analytics workspace in the exercise later in this module.

View and manage cataloged data assets

As each scan finds data assets in the registered sources, they're added to the associated collection in the data catalog. You can query the data catalog in the Microsoft Purview Governance Portal to view and filter the data assets, as shown here:

The screenshot shows the Microsoft Purview Governance Portal interface. The left sidebar has a navigation menu with items: Data catalog, Data map, Data share (preview), Data estate insights, Data policy (preview), and Management. The main content area is titled "Browse assets" and shows a list of cataloged data assets under the collection "purviewb6ljczo". The assets listed are "products" (Azure Synapse Dedicated SQL Table), "products.csv" (Azure Data Lake Storage Gen2 | File), and "products_csv" (Azure Synapse Serverless SQL View). On the right side, there are filters for "Source type : all" and "Instance : all", and a "Clear all filters" button. Below the filters, it says "Showing 1-3 out of 3 results" and "Sort by: Name". There are also sections for "Narrow results by" (Object Type: Dashboards, Data pipelines, Files, Folders, Reports, Stored procedures, Tables; Classification; Contact; Label) and a "Page" navigation bar at the bottom.

Data assets include items in the registered data stores at multiple levels. For example, assets from an Azure Synapse Analytics source include databases, schemas, tables, and individual

fields; and assets from an Azure Data Lake Storage Gen 2 source include containers, folders, and files.

You can view and edit the properties of each asset to add contextual information such as descriptions, contacts for expert help, and other useful metadata. Data assets can also be classified using built-in or custom classifications that match specific patterns of data field to common types of data - for example, passport numbers, credit card numbers, and others.

 **Tip**

To learn more about data asset classification, see [Data classification in the Microsoft Purview governance portal](#).

Next unit: Connect Microsoft Purview to an Azure Synapse Analytics workspace

[Continue >](#)

How are we doing?     

✓ 100 XP



Connect Microsoft Purview to an Azure Synapse Analytics workspace

5 minutes

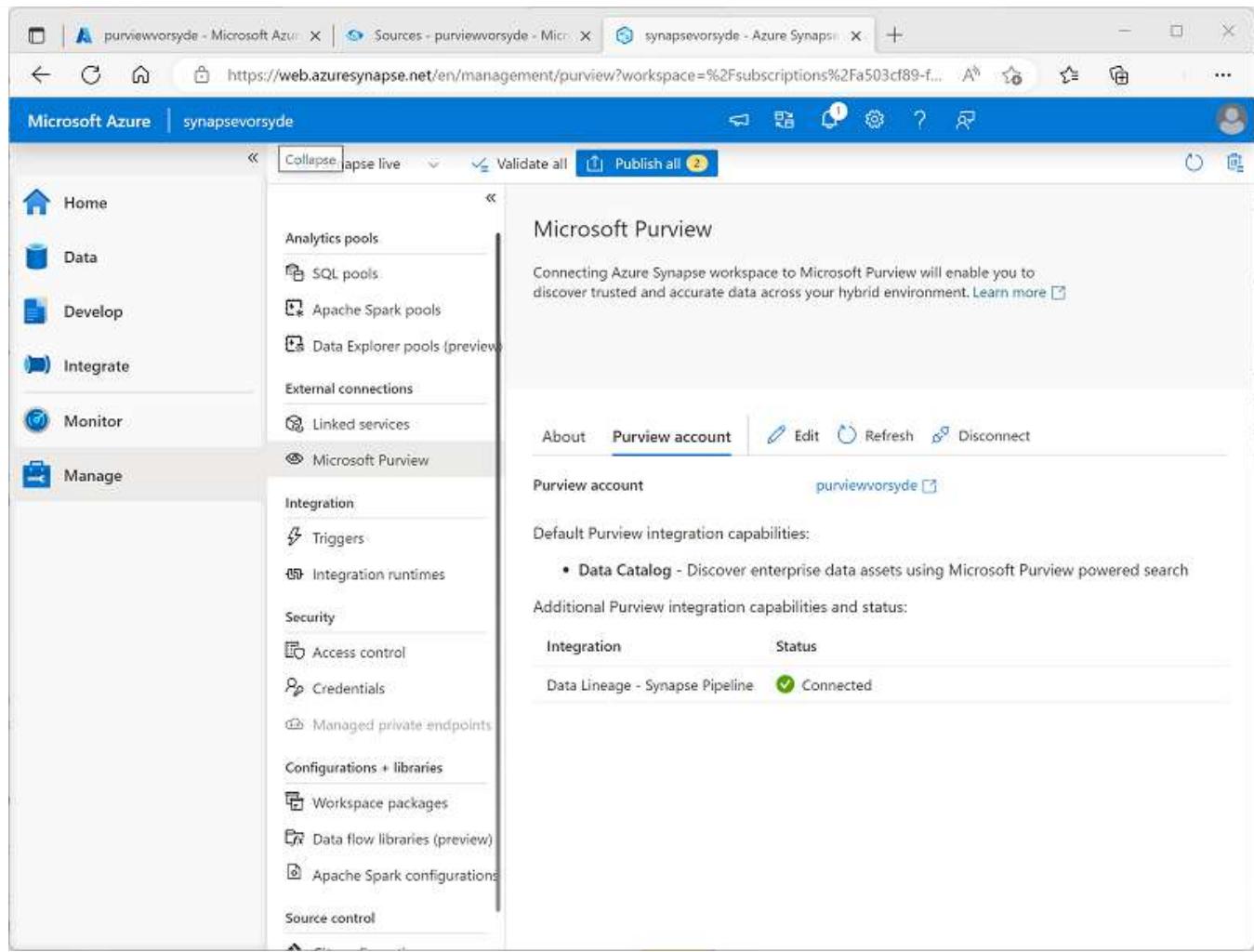
So far, you've learned how you can use Azure Synapse Analytics data stores as sources for a Microsoft Purview catalog; which is similar in most respects to using any other data source.

What sets Azure Synapse Analytics apart from many other data sources is the ability to configure direct integration between an Azure Synapse Analytics workspace and a Microsoft Purview account. By linking your workspace to a Purview account, you can:

- Search the Purview catalog in the Synapse Studio user interface.
- Push details of data pipeline activities to Purview in order to track data lineage information.

Connect a Purview account to a Synapse Analytics workspace

You connect a Microsoft Purview account to an Azure Synapse Analytics workspace on the [Manage](#) page of Synapse Studio, as shown here:



Security considerations

To connect a Purview account by using the Synapse Studio interface, you require **Collection Administrator** access to the Purview account's root collection. After successfully connecting the account, the managed identity used by your Azure Synapse Analytics workspace will be added to the collection's **Data Curator** role.

If your Microsoft Purview account is behind a firewall, you need to create a managed endpoint, and configure the connection to access Purview using that endpoint. For more information, see [Access a secured Microsoft Purview account from Azure Synapse Analytics](#).

Tip

To learn more about connecting Azure Synapse Analytics to Microsoft Purview, see [QuickStart: Connect a Synapse workspace to a Microsoft Purview account](#).

You'll get a chance to connect an Azure Synapse Analytics workspace to a Microsoft Purview account in the exercise later in this module.

✓ 100 XP



Search a Purview catalog in Synapse Studio

4 minutes

After connecting an Azure Synapse Analytics workspace to a Microsoft Purview account, you can search the Purview catalog from Synapse Studio. This ability to discover and examine data assets from across the enterprise can greatly assist data engineers, data analysts, and other consumers of data by providing a curated catalog of documented data sources for analysis and reporting.

Search the Purview catalog in Synapse Studio

You can search the catalog from a connected Purview account by using the **Search** bar in the **Data**, **Develop**, or **Integrate** pages in Synapse Studio, as shown here:

The screenshot shows the Microsoft Synapse Studio interface. On the left, there's a navigation sidebar with icons for Home, Data, Develop, Integrate, Monitor, and Manage. The 'Integrate' tab is currently selected. In the center, there's a search bar with the placeholder 'Search suggestions' and a dropdown menu showing results like 'products', 'products csv', 'products products csv', and 'products'. Below this, under 'Asset suggestions', there are four items: 'products' (with a link to 'https://datalakeb6ljczo.dfs.core.windows.net/files/products/'), 'products' (with a link to 'mssql://synapseb6ljczo.sql.azuresynapse.net/sqlb6ljczo/dba/products'), 'products.csv' (with a link to 'https://datalakeb6ljczo.dfs.core.windows.net/files/products/products.csv'), and 'products_csv' (with a link to 'mssql://synapseb6ljczo-on-demand.sql.azuresynapse.net/lakedb/dba/produc...'). To the right of the search interface, there's a large text area containing SQL script code. At the bottom, there are two tabs: 'Results' and 'Messages'. The 'Results' tab displays a message 'No items to show' and a note to 'Try creating a new item using the + button above.' The 'Messages' tab shows a magnifying glass icon and the message 'No results to show' followed by 'Your query yielded no displayable results'. A status bar at the bottom right indicates '00:00:15 Query executed successfully.'

The *search results* interface, and the details for each asset found reflect the user interface in the Microsoft Purview Governance Portal, ensuring that the data discovery and examination experience in Synapse Studio is consistent for users of Microsoft Purview in its own portal.

 **Tip**

For more information about searching the Purview catalog in Synapse Studio, see [Discover, connect, and explore data in Synapse using Microsoft Purview](#).

You'll get a chance to try searching a connected Purview account for yourself in the exercise later in this module.

Next unit: Track data lineage in pipelines

[Continue >](#)

How are we doing?     

100 XP

Track data lineage in pipelines

4 minutes

In a typical large-scale analytics solution, data is transferred and transformed across multiple systems until it's loaded into an analytical data store for reporting and analysis. Tracking the *lineage* of data as moves across the enterprise is an important factor in determining the provenance, trustworthiness, and recency of data assets used to inform analysis and decision making.

Generate and view data lineage information

In Azure Synapse Analytics, data movement and transformation is managed by using *pipelines*, which consist of an orchestrated set of *activities* that operate on data. The design and implementation of pipelines is too large a subject to cover in depth in this module, but a key point to be aware of is that there are two activity types available in Synapse Analytics pipelines that automatically generate data lineage information in a connected Purview catalog:

- The **Copy Data** activity
- The **Data Flow** activity

Running a pipeline that includes either of these activities in a workspace with a connected Purview account will result in the creation or update of data assets with lineage information. The assets recorded include:

- The source from which the data is extracted.
- The activity used to transfer the data.
- The destination where the data is stored.

In the Microsoft Purview Governance Portal, you can open the assets in the Purview catalog, and view the lineage information as shown here:

The screenshot shows the Microsoft Purview Data Catalog interface. On the left, there's a sidebar with icons for Data catalog, Data map, Data share (preview), Data estate insights (preview), Data policy (preview), and Management. The main area is titled 'Copy_lxr' and 'Azure Synapse Copy Activity'. It has tabs for Overview, Properties, Lineage, Contacts, and Related. The Lineage tab is selected, showing a lineage path from 'products.csv' to 'products'. Below this, there's a list of columns: ProductID, ProductName, Category, and ListPrice. The pipeline details section shows the activity type as 'Azure Synapse Copy Activity' with a workspace of 'synapsevorsyde-AzureSynapse' and a pipeline of '/subscriptions/.../pipelines/copy_lxr'. There are also 'See details' and 'Open in Azure Synapse Analytics' buttons.

You can also view the lineage for a pipeline activity in Synapse Studio.

💡 Tip

For more information about tracking data lineage for Azure Synapse Analytics pipelines in Microsoft Purview, see [How to get lineage from Azure Synapse Analytics into Microsoft Purview](#).

You'll get a chance to generate and view data lineage from a Synapse Analytics pipeline in the exercise later in this module.

Next unit: Exercise - Integrate Azure Synapse Analytics and Microsoft Purview

[Continue >](#)

✓ 200 XP



Knowledge check

4 minutes

1. You want to scan data assets in a dedicated SQL pool in your Azure Synapse Analytics workspace. What kind of source should you register in Microsoft Purview? *

Azure Synapse Analytics

✓ Correct. An Azure Synapse Analytics source is used to scan database in SQL pools in a workspace.

Azure Data Lake Storage Gen2

Azure SQL Database

2. You want to scan data assets in the default data lake used by your Azure Synapse Analytics workspace. What kind of source should you register in Microsoft Purview? *

Azure Synapse Analytics

Azure Data Lake Storage Gen2

✓ Correct. Azure Synapse Analytics uses an Azure Data Lake Storage Gen2 storage account for the default data lake storage.

Azure Cosmos DB

3. You want data analysts using Synapse Studio to be able to find data assets that are registered in a Microsoft Purview collection. What should you do? *

Register an Azure Synapse Analytics source in the Purview account

Add a Data Explorer pool to the Synapse Workspace

Connect the Purview account to the Synapse analytics workspace

✓ Correct. Connecting the Purview account to the workspace makes it searchable in Synapse Studio.

4. Which of the following pipeline activities records data lineage data in a connected Purview account? *

Get Metadata

Copy Data

✓ Correct. A Copy Data activity and its data lineage from source to target is recorded in a connected Purview account.

Lookup

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP

Introduction

1 minute

Azure Databricks is a fully managed, cloud-based data analytics platform, which empowers developers to accelerate AI and innovation by simplifying the process of building enterprise-grade data applications. Built as a joint effort by Microsoft and the team that started Apache Spark, Azure Databricks provides data science, engineering, and analytical teams with a single platform for big data processing and machine learning.

By combining the power of Databricks, an end-to-end, managed Apache Spark platform optimized for the cloud, with the enterprise scale and security of Microsoft's Azure platform, Azure Databricks makes it simple to run large-scale Spark workloads.

In this module, you'll learn how to:

- Provision an Azure Databricks workspace.
- Identify core workloads and personas for Azure Databricks.
- Describe key concepts of an Azure Databricks solution.

Next unit: Get started with Azure Databricks

[Continue >](#)

How are we doing?

100 XP



Get started with Azure Databricks

3 minutes

Azure Databricks is a cloud-based distributed platform for data processing built on Apache Spark. Databricks was designed to unify data science, data engineering, and business data analytics on Spark by creating an easy to use environment that enables users to spend more time working effectively with data, and less time focused on managing clusters and infrastructure. As the platform has evolved, it has kept up to date with the latest advances in the Spark runtime and added usability features to support common data workloads in a single, centrally managed interface.

Azure Databricks is hosted on the Microsoft Azure cloud platform, and integrated with Azure services such as Azure Active Directory, Azure Storage, Azure Synapse Analytics, and Azure Machine Learning. Organizations can apply their existing capabilities with the Databricks platform, and build fully integrated data analytics solutions that work with cloud infrastructure used by other enterprise applications.

Creating an Azure Databricks workspace

To use Azure Databricks, you must create an Azure Databricks *workspace* in your Azure subscription. You can accomplish this by:

- Using the Azure portal user interface.
- Using an Azure Resource Manager (ARM) or Bicep template.
- Using the `New-AzDatabricksWorkspace` Azure PowerShell cmdlet
- Using the `az databricks workspace create` Azure command line interface (CLI) command.

When you create a workspace, you must specify one of the following pricing tiers:

- **Standard** - Core Apache Spark capabilities with Azure AD integration.
- **Premium** - Role-based access controls and other enterprise-level features.
- **Trial** - A 14-day free trial of a premium-level workspace

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Workspace name *

Region *

Pricing Tier *
Standard (Apache Spark, Secure with Azure AD)
Premium (+ Role-based access controls)
Trial (Premium - 14-Days Free DBUs)

[Review + create](#) [< Previous.](#) [Next : Networking >](#)

Using the Azure Databricks portal

After you've provisioned an Azure Databricks workspace, you can use the Azure Databricks portal to work with data and compute resources. The Azure Databricks portal is a web-based user interface through which you can create and manage workspace resources (such as Spark clusters) and use notebooks and queries to work with data in files and tables.

The screenshot shows a Microsoft Azure Databricks notebook titled "Notebook for data exploration". The notebook interface includes a left sidebar with navigation links like "New", "Workspace", "Recents", "Data", "Workflows", "Compute", "SQL", "Data Engineering", "Machine Learning", "Marketplace", "Partner Connect", "Disable new UI", "Provide feedback", and "Collapse menu". The main workspace displays two command cells. Command 1 contains Python code to read a CSV file from DBFS and display its contents. Command 2 shows the resulting DataFrame as a table, listing 295 rows of product data. The table has columns: ProductID, ProductName, Category, and ListPrice. The data shows various mountain bike models and their prices.

	ProductID	ProductName	Category	ListPrice
1	771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
2	772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
3	773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
4	774	Mountain-100 Silver, 48	Mountain Bikes	3399.9900
5	775	Mountain-100 Black, 38	Mountain Bikes	3374.9900
6	776	Mountain-100 Black, 42	Mountain Bikes	3374.9900
7	777	Mountain-100 Black, 44	Mountain Bikes	3374.9900

Next unit: Identify Azure Databricks workloads

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Identify Azure Databricks workloads

3 minutes

Azure Databricks is a comprehensive platform that offers many data processing capabilities. While you can use the service to support any workload that requires scalable data processing, Azure Databricks is optimized for three specific types of data workload and associated user personas:

- Data Science and Engineering
- Machine Learning
- SQL*

**SQL workloads are only available in premium tier workspaces.*

Data Science and Engineering

Azure Databricks provides Apache Spark based processing and analysis of large volumes of data in a data lake. Data engineers, data scientists, and data analysts can use interactive notebooks to run code in Python, Scala, SparkSQL, or other languages to cleanse, transform, aggregate, and analyze data.

The screenshot shows the Azure Databricks Data Ingestion interface. On the left, a sidebar menu includes options like New, Workspace, Recents, Data, Workflows, Compute, SQL, Data Engineering, Job Runs, Data Ingestion (which is selected), Delta Live Tables, Machine Learning, Marketplace, Partner Connect, Disable new UI, and Provide feedback. The main content area is titled 'Add data' with a 'Preview' tab. It features a 'Data sources' section with a search bar. Under 'From local files (1)', there's a 'Create or modify table' button with a note: 'Upload tabular data files to create a new table or replace an existing one'. Below this is a 'Native integrations (11)' section with a grid of icons and names: Azure Blob Storage, Azure Data Lake, Cassandra, Snowflake, JDBC, Kafka, Elasticsearch, MongoDB, Postgres, MySQL, and DBFS. At the bottom, it says 'Fivetran data sources (177) See all available ingest partners in Partner Connect'.

Machine Learning

Azure Databricks supports machine learning workloads that involve data exploration and preparation, training and evaluating machine learning models, and serving models to generate predictions for applications and analyses. Data scientists and ML engineers can use AutoML to quickly train predictive models, or apply their skills with common machine learning frameworks such as SparkML, Scikit-Learn, PyTorch, and Tensorflow. They can also manage the end-to-end machine learning lifecycle with MLFlow.

The screenshot shows the Azure Databricks interface with the URL <https://adb-1363117955737105.5.azure.databricks.net/>. The left sidebar is open, showing the 'Experiments' section selected under 'Machine Learning'. The main content area displays an 'MLflow Model' page for a registered model named 'model'. The model's full path is listed as 'Full Path: dbfs:/databricks/mlflow-tracking/1110075481086604/75673901...'. A 'Register Model' button is visible. The page includes sections for 'MLflow Model', 'Model schema', and 'Make Predictions'. The 'Model schema' section shows input and output fields. The 'Make Predictions' section contains sample Python code for predicting on a Spark DataFrame.

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control and deploy as a REST endpoint for real time serving.

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (5)	
Outputs (1)	

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/75673901dba74105bed7de6e9aa8539f/model'

# Load model as a Spark UDF. Override result_type if the model does not return double values
```

SQL

Azure Databricks supports SQL-based querying for data stored in tables in a *SQL Warehouse*. This capability enables data analysts to query, aggregate, summarize, and visualize data using familiar SQL syntax and a wide range of SQL-based data analytical tools.

The screenshot shows the Azure Databricks interface. On the left, a sidebar menu includes options like 'New', 'Workspace', 'Recents', 'Data', 'Workflows', 'Compute', 'SQL', 'SQL Editor' (which is selected), 'Queries', 'Dashboards', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', 'Machine Learning', 'Marketplace', 'Partner Connect', 'Disable new UI', and 'Provide feedback'. A message at the bottom of the sidebar says 'Disable new schema browser'. The main area has tabs for 'Data' and 'SQL'. In the 'SQL' tab, a query is running: 'SELECT * FROM products'. The results table shows 7 rows of data from the 'products' table, with columns: #, ProductID, ProductName, Category, and ListPrice. The data is as follows:

#	ProductID	ProductName	Category	ListPrice
1	771	Mountain-100 Silver, 38	Mountain Bikes	3399.99
2	772	Mountain-100 Silver, 42	Mountain Bikes	3399.99
3	773	Mountain-100 Silver, 44	Mountain Bikes	3399.99
4	774	Mountain-100 Silver, 48	Mountain Bikes	3399.99
5	775	Mountain-100 Black, 38	Mountain Bikes	3374.99
6	776	Mountain-100 Black, 42	Mountain Bikes	3374.99
7	777	Mountain-100 Black, 44	Mountain Bikes	3374.99

At the bottom of the results table, it says '14 s 442 ms | 295 rows returned' and 'Refreshed 3 minutes ago'.

! Note

SQL Warehouses are only available in *premium* Azure Databricks workspaces.

Next unit: Understand key concepts

[Continue >](#)

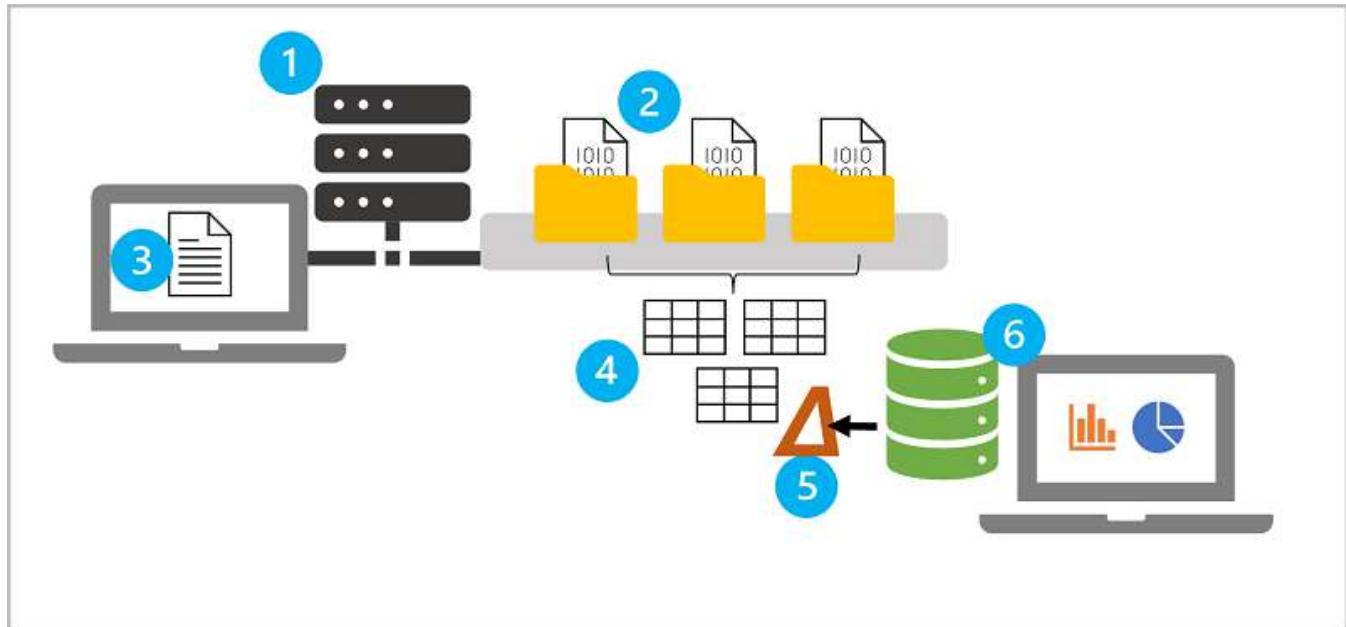
How are we doing? ☆ ☆ ☆ ☆ ☆



Understand key concepts

3 minutes

Azure Databricks is an amalgamation of multiple technologies that enable you to work with data at scale. Before using Azure Databricks, there are some key concepts that you should understand.



1. **Apache Spark clusters** - Spark is a distributed data processing solution that makes use of *clusters* to scale processing across multiple compute *nodes*. Each Spark cluster has a *driver* node to coordinate processing jobs, and one or more *worker* nodes on which the processing occurs. This distributed model enables each node to operate on a subset of the job in parallel; reducing the overall time for the job to complete. To learn more about clusters in Azure Databricks, see [Clusters](#) in the Azure Databricks documentation.
2. **Databricks File System (DBFS)** - While each cluster node has its own local file system (on which operating system and other node-specific files are stored), the nodes in a cluster have access to a shared, distributed file system in which they can access and operate on data files. The *Databricks File System* (DBFS) enables you to mount cloud storage and use it to work with and persist file-based data. To learn more about DBFS, see [Databricks File System \(DBFS\)](#) in the Azure Databricks documentation.
3. **Notebooks** - One of the most common ways for data analysts, data scientists, data engineers, and developers to work with Spark is to write code in *notebooks*. Notebooks provide an interactive environment in which you can combine text and graphics in *Markdown* format with cells containing code that you run interactively in the notebook

session. To learn more about notebooks, see [Notebooks in the Azure Databricks documentation](#).

4. **Hive metastore** - *Hive* is an open source technology used to define a relational abstraction layer of tables over file-based data. The tables can then be queried using SQL syntax. The table definitions and details of the file system locations on which they're based is stored in the metastore for a Spark cluster. A *Hive metastore* is created for each cluster when it's created, but you can configure a cluster to use an existing external metastore if necessary - see [Metastores](#) in the Azure Databricks documentation for more details.
5. **Delta Lake** - *Delta Lake* builds on the relational table schema abstraction over files in the data lake to add support for SQL semantics commonly found in relational database systems. Capabilities provided by Delta Lake include transaction logging, data type constraints, and the ability to incorporate streaming data into a relational table. To learn more about Delta Lake, see [Delta Lake Guide](#) in the Azure Databricks documentation.
6. **SQL Warehouses** - *SQL Warehouses* are relational compute resources with endpoints that enable client applications to connect to an Azure Databricks workspace and use SQL to work with data in tables. The results of SQL queries can be used to create data visualizations and dashboards to support business analytics and decision making. SQL Warehouses are only available in *premium* tier Azure Databricks workspaces. To learn more about SQL Warehouses, see [SQL Warehouses](#) in the Azure Databricks documentation.

Next unit: Exercise - Explore Azure Databricks

[Continue >](#)

How are we doing?

Knowledge check

3 minutes

1. You plan to create an Azure Databricks workspace and use it to work with a SQL Warehouse. Which of the following pricing tiers can you select? *

Enterprise

X Incorrect. There is no Enterprise tier for Azure Databricks.

Standard

Premium

✓ Correct. Premium tier is required for SQL Warehouses.

2. You've created an Azure Databricks workspace in which you plan to use code to process data files. What must you create in the workspace? *

A SQL Warehouse

A Spark cluster

✓ Correct. A Spark cluster is required to process data using code in notebooks.

A Windows Server virtual machine

3. You want to use Python code to interactively explore data in a text file that you've uploaded to your Azure Databricks workspace. What should you create? *

A SQL query

An Azure function

A Notebook

✓ Correct. A notebook is used to interactively explore data.

Next unit: Summary

Continue >

100 XP

Introduction

1 minute

Azure Databricks offers a highly scalable platform for data analytics and processing using Apache Spark.

Spark is a flexible platform that supports many different programming languages and APIs. Most data processing and analytics tasks can be accomplished using the **Dataframe API**, which is what we'll focus on in this module.

In this module, you'll learn how to:

- Describe key elements of the Apache Spark architecture.
- Create and configure a Spark cluster.
- Describe use cases for Spark.
- Use Spark to process and analyze data stored in files.
- Use Spark to visualize data.

Next unit: Get to know Spark

[Continue >](#)

How are we doing?

Get to know Spark

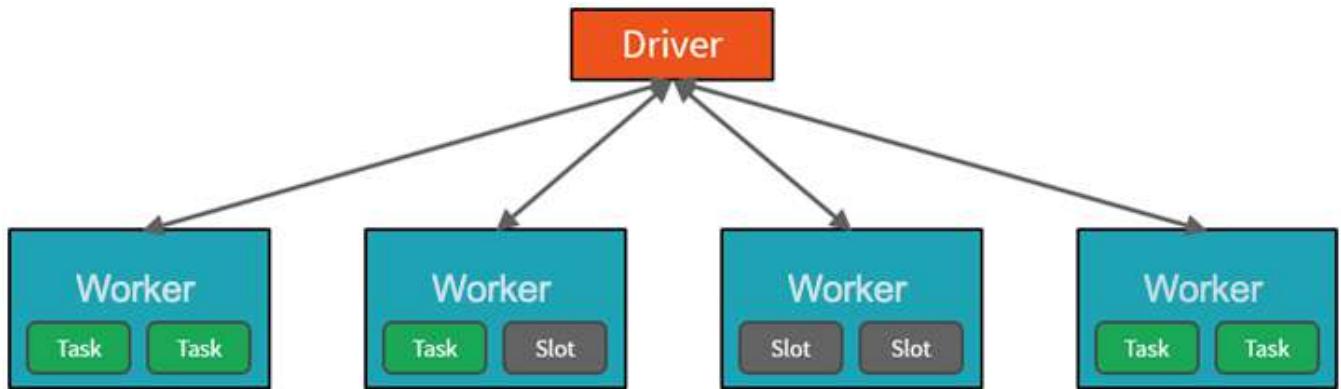
4 minutes

To gain a better understanding of how to process and analyze data with Apache Spark in Azure Databricks, it's important to understand the underlying architecture.

High-level overview

From a high level, the Azure Databricks service launches and manages Apache Spark clusters within your Azure subscription. Apache Spark clusters are groups of computers that are treated as a single computer and handle the execution of commands issued from notebooks. Clusters enable processing of data to be parallelized across many computers to improve scale and performance. They consist of a Spark *driver* and *worker* nodes. The driver node sends work to the worker nodes and instructs them to pull data from a specified data source.

In Databricks, the notebook interface is typically the driver program. This driver program contains the main loop for the program and creates distributed datasets on the cluster, then applies operations to those datasets. Driver programs access Apache Spark through a *SparkSession* object regardless of deployment location.



Microsoft Azure manages the cluster, and auto-scales it as needed based on your usage and the setting used when configuring the cluster. Auto-termination can also be enabled, which allows Azure to terminate the cluster after a specified number of minutes of inactivity.

Spark jobs in detail

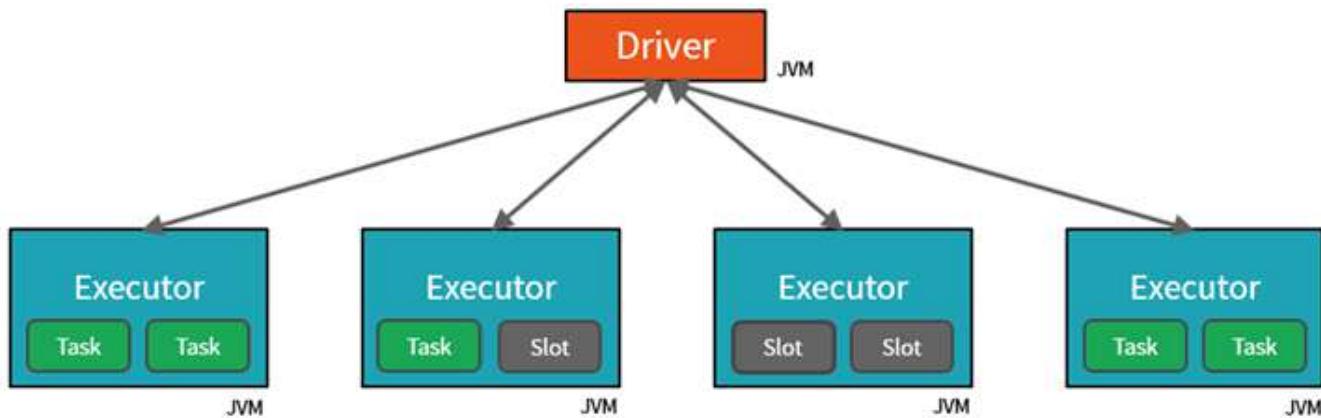
Work submitted to the cluster is split into as many independent jobs as needed. This is how work is distributed across the Cluster's nodes. Jobs are further subdivided into tasks. The input to a job is partitioned into one or more partitions. These partitions are the unit of work for

each slot. In between tasks, partitions may need to be reorganized and shared over the network.

The secret to Spark's high performance is parallelism. Scaling *vertically* (by adding resources to a single computer) is limited to a finite amount of RAM, Threads and CPU speeds; but clusters scale *horizontally*, adding new nodes to the cluster as needed.

Spark parallelizes jobs at two levels:

- The first level of parallelization is the *executor* - a Java virtual machine (JVM) running on a worker node, typically, one instance per node.
- The second level of parallelization is the *slot* - the number of which is determined by the number of cores and CPUs of each node.
- Each executor has multiple slots to which parallelized tasks can be assigned.



The JVM is naturally multi-threaded, but a single JVM, such as the one coordinating the work on the driver, has a finite upper limit. By splitting the work into tasks, the driver can assign units of work to *slots in the executors on worker nodes for parallel execution. Additionally, the driver determines how to partition the data so that it can be distributed for parallel processing. So, the driver assigns a partition of data to each task so that each task knows which piece of data it is to process. Once started, each task will fetch the partition of data assigned to it.

Jobs and stages

Depending on the work being performed, multiple parallelized jobs may be required. Each job is broken down into *stages*. A useful analogy is to imagine that the job is to build a house:

- The first stage would be to lay the foundation.
- The second stage would be to erect the walls.
- The third stage would be to add the roof.

Attempting to do any of these steps out of order just doesn't make sense, and may in fact be impossible. Similarly, Spark breaks each job into stages to ensure everything is done in the right order.

✓ 100 XP ➔

Create a Spark cluster

3 minutes

You can create one or more clusters in your Azure Databricks workspace by using the Azure Databricks portal.

The screenshot shows the 'Cluster Details - Databricks' page in a browser. The URL is https://adb-1363117955737105.5.azure.databricks.net/?o=1363117955737... . The left sidebar has a 'Compute' section selected. The main area shows 'My Cluster' configuration with tabs for Configuration, Notebooks (0), Libraries, Event log, Spark UI, Driver logs, Metrics, Apps, and Spark compute UI - Master. Under Configuration, there are sections for Policy (Unrestricted), Access mode (No isolation shared), Performance (Databricks Runtime Version: 13.3 LTS ML (includes Apache Spark 3.4.0, Scala 2.12), Use Photon Acceleration checked), Node type (Standard_DS3_v2, 14 GB Memory, 4 Cores), and a checkbox for Terminate after 30 minutes of inactivity. A summary panel on the right indicates 1 Driver, Runtime, and Standard_DS3_v2. The UI tab is selected.

When creating the cluster, you can specify configuration settings, including:

- A name for the cluster.
- A *cluster mode*, which can be:
 - *Standard*: Suitable for single-user workloads that require multiple worker nodes.
 - *High Concurrency*: Suitable for workloads where multiple users will be using the cluster concurrently.
 - *Single Node*: Suitable for small workloads or testing, where only a single worker node is required.
- The version of the *Databricks Runtime* to be used in the cluster; which dictates the version of Spark and individual components such as Python, Scala, and others that get installed.

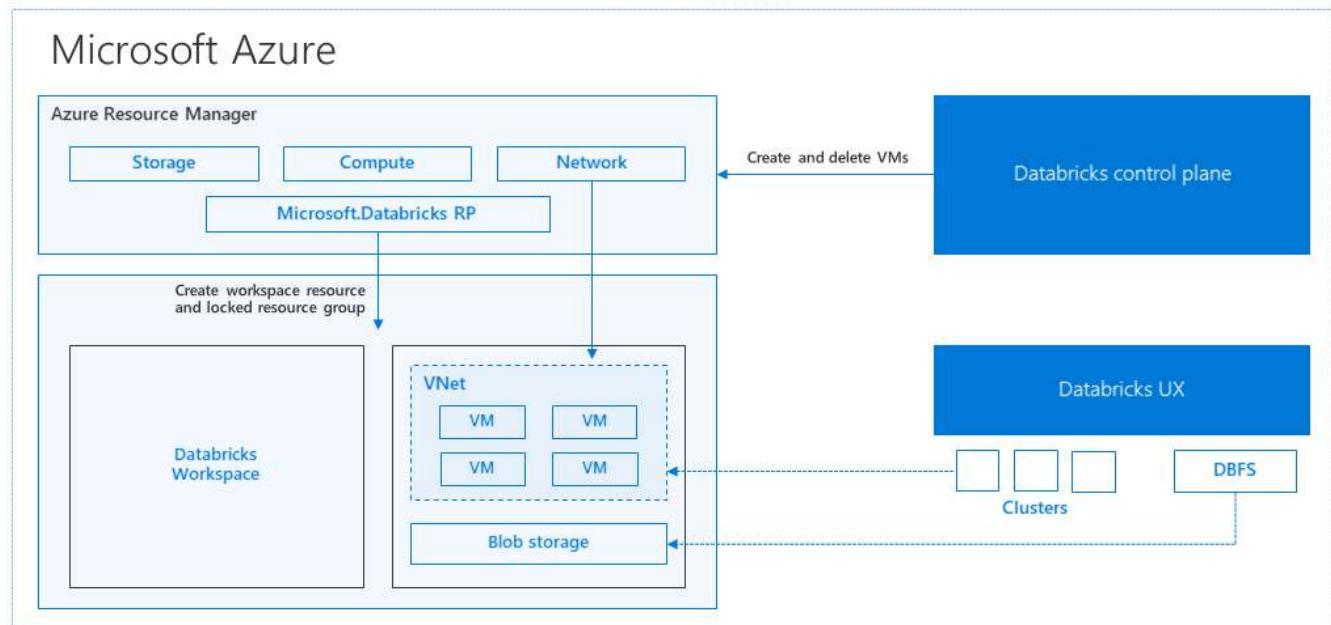
- The type of virtual machine (VM) used for the worker nodes in the cluster.
- The minimum and maximum number of worker nodes in the cluster.
- The type of VM used for the driver node in the cluster.
- Whether the cluster supports *autoscaling* to dynamically resize the cluster.
- How long the cluster can remain idle before being shut down automatically.

How Azure manages cluster resources

When you create an Azure Databricks workspace, a *Databricks appliance* is deployed as an Azure resource in your subscription. When you create a cluster in the workspace, you specify the types and sizes of the virtual machines (VMs) to use for both the driver and worker nodes, and some other configuration options, but Azure Databricks manages all other aspects of the cluster.

The Databricks appliance is deployed into Azure as a managed resource group within your subscription. This resource group contains the driver and worker VMs for your clusters, along with other required resources, including a virtual network, a security group, and a storage account. All metadata for your cluster, such as scheduled jobs, is stored in an Azure Database with geo-replication for fault tolerance.

Internally, Azure Kubernetes Service (AKS) is used to run the Azure Databricks control-plane and data-planes via containers running on the latest generation of Azure hardware (Dv3 VMs), with NVMe SSDs capable of blazing 100us latency on high-performance Azure virtual machines with accelerated networking. Azure Databricks utilizes these features of Azure to further improve Spark performance. After the services within your managed resource group are ready, you can manage the Databricks cluster through the Azure Databricks UI and through features such as auto-scaling and auto-termination.



 Note

You also have the option of attaching your cluster to a *pool* of idle nodes to reduce cluster startup time. For more information, see **Pools** in the Azure Databricks documentation.

Next unit: Use Spark in notebooks

[Continue >](#)

How are we doing?     

100 XP



Use Spark in notebooks

6 minutes

You can run many different kinds of application on Spark, including code in Python or Scala scripts, Java code compiled as a Java Archive (JAR), and others. Spark is commonly used in two kinds of workload:

- Batch or stream processing jobs to ingest, clean, and transform data - often running as part of an automated pipeline.
- Interactive analytics sessions to explore, analyze, and visualize data.

Running Spark code in notebooks

Azure Databricks includes an integrated notebook interface for working with Spark. Notebooks provide an intuitive way to combine code with Markdown notes, commonly used by data scientists and data analysts. The look and feel of the integrated notebook experience within Azure Databricks is similar to that of Jupyter notebooks - a popular open source notebook platform.

The screenshot shows the Microsoft Azure Databricks interface. On the left is a sidebar with navigation links like 'New', 'Workspace', 'Recents', 'Data', 'Workflows', 'Compute', 'SQL', 'Data Engineering', 'Machine Learning', 'Marketplace', 'Partner Connect', 'Disable new UI', 'Provide feedback', and 'Collapse menu'. The main area is titled 'My Notebook' and 'Python'. It contains two code cells: 'Cmd 1' and 'Cmd 2'. 'Cmd 1' shows a single line of Python code: 'df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/user.name@contoso.com/products.csv")'. 'Cmd 2' shows the output of the code: '(2) Spark Jobs' and 'df1: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 2 more fields]'. Below this is a table view with columns 'ProductID', 'ProductName', 'Category', and 'ListPrice'. The data includes rows for Mountain-100 Silver and Mountain-100 Black models. At the bottom of the notebook area, it says '295 rows | 0.72 seconds runtime' and 'Refreshed 3 minutes ago'. A note at the bottom states 'Command took 0.72 seconds'.

Notebooks consist of one or more *cells*, each containing either code or markdown. Code cells in notebooks have some features that can help you be more productive, including:

- Syntax highlighting and error support.
- Code auto-completion.
- Interactive data visualizations.
- The ability to export results.

Tip

To learn more about working with notebooks in Azure Databricks, see the [Notebooks](#) article in the Azure Databricks documentation.

Next unit: Use Spark to work with data files

[Continue >](#)

✓ 100 XP



Use Spark to work with data files

5 minutes

One of the benefits of using Spark is that you can write and run code in various programming languages, enabling you to use the programming skills you already have and to use the most appropriate language for a given task. The default language in a new Azure Databricks Spark notebook is *PySpark* - a Spark-optimized version of Python, which is commonly used by data scientists and analysts due to its strong support for data manipulation and visualization. Additionally, you can use languages such as *Scala* (a Java-derived language that can be used interactively) and *SQL* (a variant of the commonly used SQL language included in the *Spark SQL* library to work with relational data structures). Software engineers can also create compiled solutions that run on Spark using frameworks such as *Java*.

Exploring data with dataframes

Natively, Spark uses a data structure called a *resilient distributed dataset* (RDD); but while you *can* write code that works directly with RDDs, the most commonly used data structure for working with structured data in Spark is the *dataframe*, which is provided as part of the *Spark SQL* library. Dataframes in Spark are similar to those in the ubiquitous *Pandas* Python library, but optimized to work in Spark's distributed processing environment.

⚠ Note

In addition to the Dataframe API, Spark SQL provides a strongly-typed *Dataset* API that is supported in Java and Scala. We'll focus on the Dataframe API in this module.

Loading data into a dataframe

Let's explore a hypothetical example to see how you can use a dataframe to work with data. Suppose you have the following data in a comma-delimited text file named **products.csv** in the **data** folder in your Databricks File System (DBFS) storage:

csv

```
ProductID,ProductName,Category,ListPrice
771,"Mountain-100 Silver, 38",Mountain Bikes,3399.9900
772,"Mountain-100 Silver, 42",Mountain Bikes,3399.9900
```

```
773,"Mountain-100 Silver, 44",Mountain Bikes,3399.9900
```

```
...
```

In a Spark notebook, you could use the following PySpark code to load the data into a dataframe and display the first 10 rows:

Python

```
%pyspark
df = spark.read.load('/data/products.csv',
    format='csv',
    header=True
)
display(df.limit(10))
```

The `%pyspark` line at the beginning is called a *magic*, and tells Spark that the language used in this cell is PySpark. Here's the equivalent Scala code for the products data example:

Scala

```
%spark
val df = spark.read.format("csv").option("header",
"true").load("/data/products.csv")
display(df.limit(10))
```

The magic `%spark` is used to specify Scala.

💡 Tip

You can also select the language you want to use for each cell in the Notebook interface.

Both of the examples shown previously would produce output like this:

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
...

Specifying a dataframe schema

In the previous example, the first row of the CSV file contained the column names, and Spark was able to infer the data type of each column from the data it contains. You can also specify an explicit schema for the data, which is useful when the column names aren't included in the data file, like this CSV example:

CSV

```
771,"Mountain-100 Silver, 38",Mountain Bikes,3399.9900
772,"Mountain-100 Silver, 42",Mountain Bikes,3399.9900
773,"Mountain-100 Silver, 44",Mountain Bikes,3399.9900
...
```

The following PySpark example shows how to specify a schema for the dataframe to be loaded from a file named **product-data.csv** in this format:

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

productSchema = StructType([
    StructField("ProductID", IntegerType()),
    StructField("ProductName", StringType()),
    StructField("Category", StringType()),
    StructField("ListPrice", FloatType())
])

df = spark.read.load('/data/product-data.csv',
    format='csv',
    schema=productSchema,
    header=False)
display(df.limit(10))
```

The results would once again be similar to:

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900

ProductID	ProductName	Category	ListPrice
...
...

Filtering and grouping dataframes

You can use the methods of the Dataframe class to filter, sort, group, and otherwise manipulate the data it contains. For example, the following code example uses the `select` method to retrieve the `ProductName` and `ListPrice` columns from the `df` dataframe containing product data in the previous example:

Python

```
pricelist_df = df.select("ProductID", "ListPrice")
```

The results from this code example would look something like this:

ProductID	ListPrice
771	3399.9900
772	3399.9900
773	3399.9900
...	...

In common with most data manipulation methods, `select` returns a new dataframe object.

💡 Tip

Selecting a subset of columns from a dataframe is a common operation, which can also be achieved by using the following shorter syntax:

```
pricelist_df = df["ProductID", "ListPrice"]
```

You can "chain" methods together to perform a series of manipulations that results in a transformed dataframe. For example, this example code chains the `select` and `where` methods

to create a new dataframe containing the **ProductName** and **ListPrice** columns for products with a category of **Mountain Bikes** or **Road Bikes**:

```
Python
```

```
bikes_df = df.select("ProductName",  
"ListPrice").where((df["Category"]=="Mountain Bikes") |  
(df["Category"]=="Road Bikes"))  
display(bikes_df)
```

The results from this code example would look something like this:

ProductName	ListPrice
Mountain-100 Silver, 38	3399.9900
Road-750 Black, 52	539.9900
...	...

To group and aggregate data, you can use the **groupBy** method and aggregate functions. For example, the following PySpark code counts the number of products for each category:

```
Python
```

```
counts_df = df.select("ProductID", "Category").groupBy("Category").count()  
display(counts_df)
```

The results from this code example would look something like this:

Category	count
Headsets	3
Wheels	14
Mountain Bikes	32
...	...

Using SQL expressions in Spark

The Dataframe API is part of a Spark library named Spark SQL, which enables data analysts to use SQL expressions to query and manipulate data.

Creating database objects in the Spark catalog

The Spark catalog is a metastore for relational data objects such as views and tables. The Spark runtime can use the catalog to seamlessly integrate code written in any Spark-supported language with SQL expressions that may be more natural to some data analysts or developers.

One of the simplest ways to make data in a dataframe available for querying in the Spark catalog is to create a temporary view, as shown in the following code example:

Python

```
df.createOrReplaceTempView("products")
```

A *view* is temporary, meaning that it's automatically deleted at the end of the current session. You can also create *tables* that are persisted in the catalog to define a database that can be queried using Spark SQL.

ⓘ Note

We won't explore Spark catalog tables in depth in this module, but it's worth taking the time to highlight a few key points:

- You can create an empty table by using the `spark.catalog.createTable` method. Tables are metadata structures that store their underlying data in the storage location associated with the catalog. Deleting a table also deletes its underlying data.
- You can save a dataframe as a table by using its `saveAsTable` method.
- You can create an *external* table by using the `spark.catalog.createExternalTable` method. External tables define metadata in the catalog but get their underlying data from an external storage location; typically a folder in a data lake. Deleting an external table does not delete the underlying data.

Using the Spark SQL API to query data

You can use the Spark SQL API in code written in any language to query data in the catalog. For example, the following PySpark code uses a SQL query to return data from the **products** view as a dataframe.

Python

```
bikes_df = spark.sql("SELECT ProductID, ProductName, ListPrice \
                      FROM products \
                      WHERE Category IN ('Mountain Bikes', 'Road Bikes')")  
display(bikes_df)
```

The results from the code example would look similar to the following table:

ProductName	ListPrice
Mountain-100 Silver, 38	3399.9900
Road-750 Black, 52	539.9900
...	...

Using SQL code

The previous example demonstrated how to use the Spark SQL API to embed SQL expressions in Spark code. In a notebook, you can also use the `%sql` magic to run SQL code that queries objects in the catalog, like this:

SQL

```
%sql  
  
SELECT Category, COUNT(ProductID) AS ProductCount  
FROM products  
GROUP BY Category  
ORDER BY Category
```

The SQL code example returns a resultset that is automatically displayed in the notebook as a table, like the one below:

Category	ProductCount
Bib-Shorts	3

Category	ProductCount
Bike Racks	1
Bike Stands	1
...	...

Next unit: Visualize data

[Continue >](#)

How are we doing?

✓ 100 XP



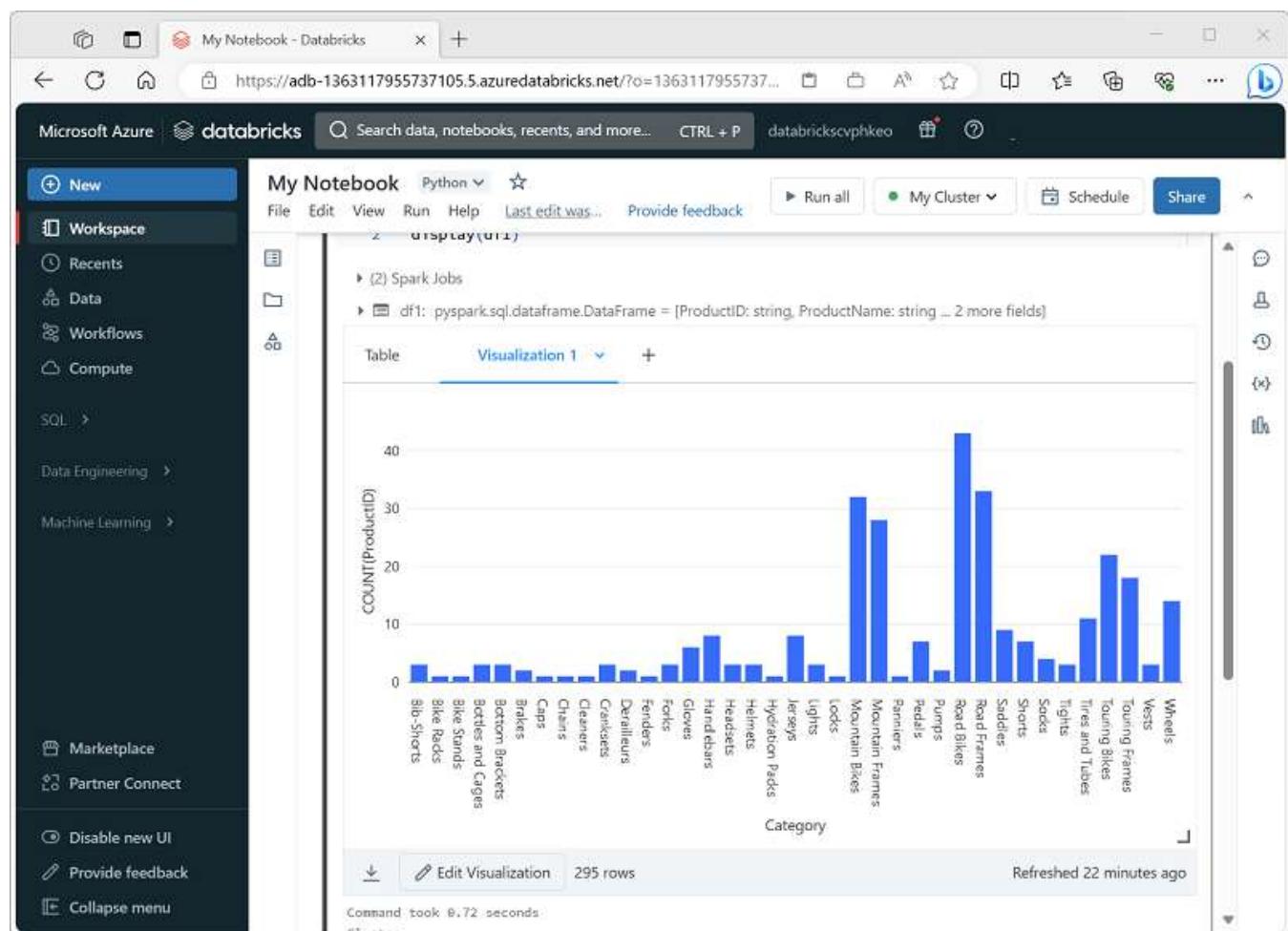
Visualize data

6 minutes

One of the most intuitive ways to analyze the results of data queries is to visualize them as charts. Notebooks in Azure Databricks provide charting capabilities in the user interface, and when that functionality doesn't provide what you need, you can use one of the many Python graphics libraries to create and display data visualizations in the notebook.

Using built-in notebook charts

When you display a dataframe or run a SQL query in a Spark notebook in Azure Databricks, the results are displayed under the code cell. By default, results are rendered as a table, but you can also view the results as a visualization and customize how the chart displays the data, as shown here:



The built-in visualization functionality in notebooks is useful when you want to quickly summarize the data visually. When you want to have more control over how the data is

formatted, or to display values that you have already aggregated in a query, you should consider using a graphics package to create your own visualizations.

Using graphics packages in code

There are many graphics packages that you can use to create data visualizations in code. In particular, Python supports a large selection of packages; most of them built on the base **Matplotlib** library. The output from a graphics library can be rendered in a notebook, making it easy to combine code to ingest and manipulate data with inline data visualizations and markdown cells to provide commentary.

For example, you could use the following PySpark code to aggregate data from the hypothetical products data explored previously in this module, and use Matplotlib to create a chart from the aggregated data.

Python

```
from matplotlib import pyplot as plt

# Get the data as a Pandas dataframe
data = spark.sql("SELECT Category, COUNT(ProductID) AS ProductCount \
                  FROM products \
                  GROUP BY Category \
                  ORDER BY Category").toPandas()

# Clear the plot area
plt.clf()

# Create a Figure
fig = plt.figure(figsize=(12,8))

# Create a bar plot of product counts by category
plt.bar(x=data['Category'], height=data['ProductCount'], color='orange')

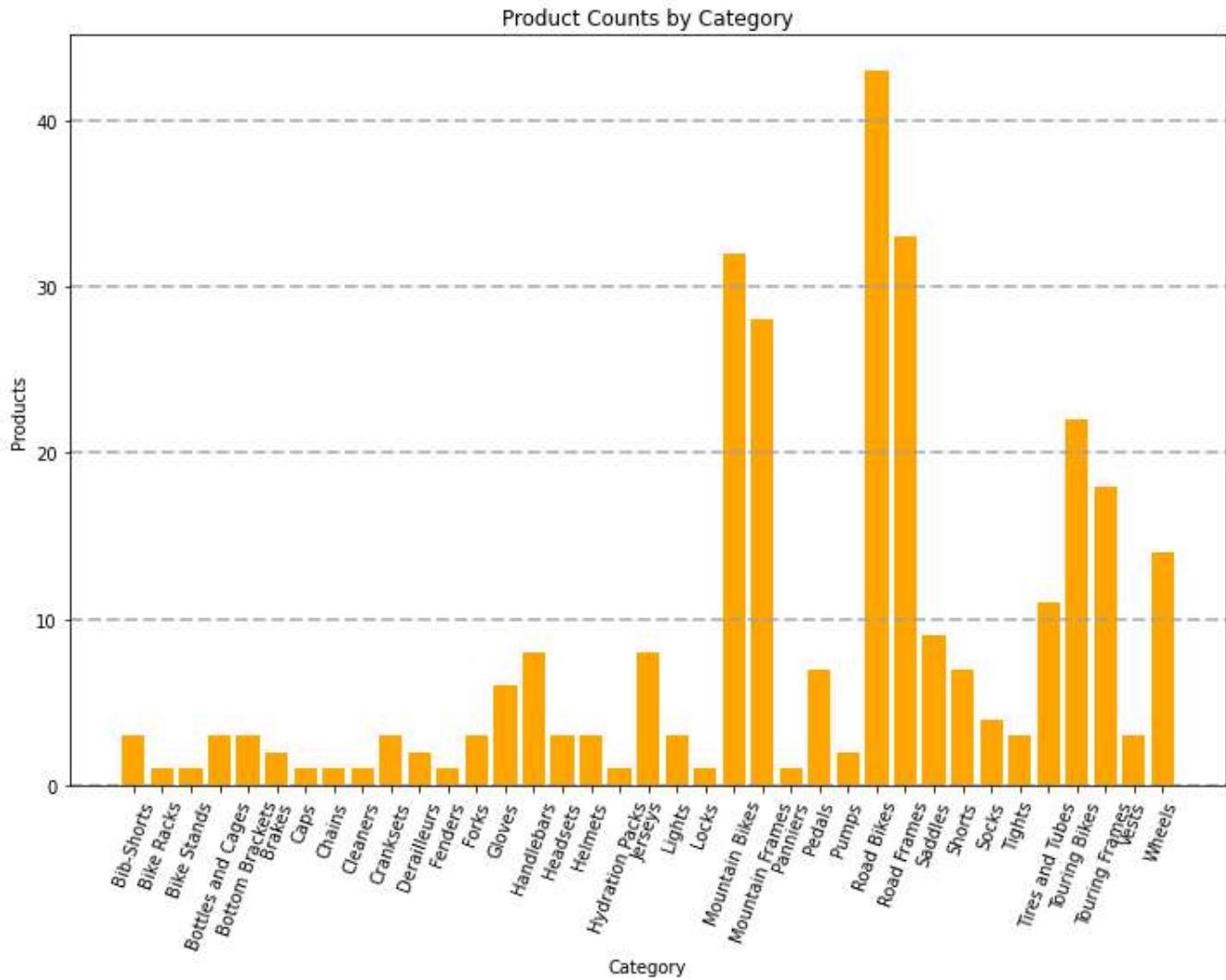
# Customize the chart
plt.title('Product Counts by Category')
plt.xlabel('Category')
plt.ylabel('Products')
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
plt.xticks(rotation=70)

# Show the plot area
plt.show()
```

The Matplotlib library requires data to be in a Pandas dataframe rather than a Spark dataframe, so the **toPandas** method is used to convert it. The code then creates a figure with a specified

size and plots a bar chart with some custom property configuration before showing the resulting plot.

The chart produced by the code would look similar to the following image:



You can use the Matplotlib library to create many kinds of chart; or if preferred, you can use other libraries such as Seaborn to create highly customized charts.

! Note

The Matplotlib and Seaborn libraries may already be installed on Databricks clusters, depending on the Databricks Runtime for the cluster. If not, or if you want to use a different library that is not already installed, you can add it to the cluster. See [Cluster Libraries](#) in the Azure Databricks documentation for details.

Next unit: Exercise - Use Spark in Azure Databricks

Check your knowledge

1. Which definition best describes Apache Spark? *



A highly scalable relational database management system.

X Incorrect. Spark supports some features of relational databases, but it is not primarily a relational database engine.



A virtual server with a Python runtime.



A distributed platform for parallel data processing using multiple languages.

✓ Correct. Spark provides a highly scalable distributed platform on which you can run code written in many languages to process data.

2. You need to use Spark to analyze data in a parquet file. What should you do? *



Load the parquet file into a dataframe.

✓ Correct. You can load data from files in many formats, including parquet, into a Spark dataframe.



Import the data into a table in a serverless SQL pool.

X Incorrect. You do not need to load the data into a SQL pool in order to analyze it with Spark.



Convert the data to CSV format.

3. You want to write code in a notebook cell that uses a SQL query to retrieve data from a view in the Spark catalog. Which magic should you use? *



%spark



%pyspark



%sql

✓ Correct. The %sql magic instructs Spark to interpret the code in the cell as SQL.

Next unit: Summary

Continue >

100 XP

Introduction

1 minute

Linux foundation *Delta Lake* is an open-source storage layer for Spark that enables relational database capabilities for batch and streaming data. By using Delta Lake, you can implement a *data lakehouse* architecture in Spark to support SQL-based data manipulation semantics with support for transactions and schema enforcement. The result is an analytical data store that offers many of the advantages of a relational database system with the flexibility of data file storage in a data lake.

In this module, you'll learn how to:

- Describe core features and capabilities of Delta Lake.
- Create and use Delta Lake tables in Azure Databricks.
- Create Spark catalog tables for Delta Lake data.
- Use Delta Lake tables for streaming data.

Note

The version of Delta Lake available in an Azure Databricks cluster depends on the version of the Databricks Runtime being used. The information in this module reflects Delta Lake version 1.2.1, which is installed with Databricks Runtime version 10.5 - 11.0.

Next unit: Get Started with Delta Lake

[Continue >](#)

How are we doing?

100 XP

Get Started with Delta Lake

3 minutes

Delta Lake is an open-source storage layer that adds relational database semantics to Spark-based data lake processing. Delta Lake is supported in Azure Synapse Analytics Spark pools for PySpark, Scala, and .NET code.

The benefits of using Delta Lake in Azure Databricks include:

- **Relational tables that support querying and data modification.** With Delta Lake, you can store data in tables that support *CRUD* (create, read, update, and delete) operations. In other words, you can *select*, *insert*, *update*, and *delete* rows of data in the same way you would in a relational database system.
- **Support for *ACID* transactions.** Relational databases are designed to support transactional data modifications that provide *atomicity* (transactions complete as a single unit of work), *consistency* (transactions leave the database in a consistent state), *isolation* (in-process transactions can't interfere with one another), and *durability* (when a transaction completes, the changes it made are persisted). Delta Lake brings this same transactional support to Spark by implementing a transaction log and enforcing serializable isolation for concurrent operations.
- **Data versioning and *time travel*.** Because all transactions are logged in the transaction log, you can track multiple versions of each table row, and even use the *time travel* feature to retrieve a previous version of a row in a query.
- **Support for batch and streaming data.** While most relational databases include tables that store static data, Spark includes native support for streaming data through the Spark Structured Streaming API. Delta Lake tables can be used as both *sinks* (destinations) and *sources* for streaming data.
- **Standard formats and interoperability.** The underlying data for Delta Lake tables is stored in Parquet format, which is commonly used in data lake ingestion pipelines.

Tip

For more information about Delta Lake in Azure Databricks, see the [Delta Lake guide](#) in the Azure Databricks documentation.

Next unit: Create Delta Lake tables

✓ 100 XP



Create Delta Lake tables

5 minutes

Delta lake is built on tables, which provide a relational storage abstraction over files in a data lake.

Creating a Delta Lake table from a dataframe

One of the easiest ways to create a Delta Lake table is to save a dataframe in the *delta* format, specifying a path where the data files and related metadata information for the table should be stored.

For example, the following PySpark code loads a dataframe with data from an existing file, and then saves that dataframe to a new folder location in *delta* format:

Python

```
# Load a file into a dataframe
df = spark.read.load('/data/mydata.csv', format='csv', header=True)

# Save the dataframe as a delta table
delta_table_path = "/delta/mydata"
df.write.format("delta").save(delta_table_path)
```

After saving the delta table, the path location you specified includes parquet files for the data (regardless of the format of the source file you loaded into the dataframe) and a `_delta_log` folder containing the transaction log for the table.

⚠ Note

The transaction log records all data modifications to the table. By logging each modification, transactional consistency can be enforced and versioning information for the table can be retained.

You can replace an existing Delta Lake table with the contents of a dataframe by using the `overwrite` mode, as shown here:

Python

```
new_df.write.format("delta").mode("overwrite").save(delta_table_path)
```

You can also add rows from a dataframe to an existing table by using the `append` mode:

Python

```
new_rows_df.write.format("delta").mode("append").save(delta_table_path)
```

Making conditional updates

While you can make data modifications in a dataframe and then replace a Delta Lake table by overwriting it, a more common pattern in a database is to insert, update or delete rows in an existing table as discrete transactional operations. To make such modifications to a Delta Lake table, you can use the `DeltaTable` object in the Delta Lake API, which supports `update`, `delete`, and `merge` operations. For example, you could use the following code to update the `price` column for all rows with a `category` column value of "Accessories":

Python

```
from delta.tables import *
from pyspark.sql.functions import *

# Create a deltaTable object
deltaTable = DeltaTable.forPath(spark, delta_table_path)

# Update the table (reduce price of accessories by 10%)
deltaTable.update(
    condition = "Category == 'Accessories'",
    set = { "Price": "Price * 0.9" })
```

The data modifications are recorded in the transaction log, and new parquet files are created in the table folder as required.

💡 Tip

For more information about using the Data Lake API, see [the Delta Lake API documentation](#).

Querying a previous version of a table

Delta Lake tables support versioning through the transaction log. The transaction log records modifications made to the table, noting the timestamp and version number for each transaction. You can use this logged version data to view previous versions of the table - a feature known as *time travel*.

You can retrieve data from a specific version of a Delta Lake table by reading the data from the delta table location into a dataframe, specifying the version required as a `versionAsOf` option:

Python

```
df = spark.read.format("delta").option("versionAsOf", 0).load(delta_table_path)
```

Alternatively, you can specify a timestamp by using the `timestampAsOf` option:

Python

```
df = spark.read.format("delta").option("timestampAsOf", '2022-01-01').load(delta_table_path)
```

Next unit: Create and query catalog tables

[Continue >](#)

How are we doing? 

✓ 100 XP



Create and query catalog tables

5 minutes

So far we've considered Delta Lake table instances created from dataframes and modified through the Delta Lake API. You can also define Delta Lake tables as catalog tables in the Hive metastore for your Spark cluster, and work with them using SQL.

External vs managed tables

Tables in a Spark catalog, including Delta Lake tables, can be *managed* or *external*; and it's important to understand the distinction between these kinds of table.

- A *managed* table is defined without a specified location, and the data files are stored within the storage used by the metastore. Dropping the table not only removes its metadata from the catalog, but also deletes the folder in which its data files are stored.
- An *external* table is defined for a custom file location, where the data for the table is stored. The metadata for the table is defined in the Spark catalog. Dropping the table deletes the metadata from the catalog, but doesn't affect the data files.

Creating catalog tables

There are several ways to create catalog tables.

Creating a catalog table from a dataframe

You can create managed tables by writing a dataframe using the `saveAsTable` operation as shown in the following examples:

Python

```
# Save a dataframe as a managed table
df.write.format("delta").saveAsTable("MyManagedTable")

## specify a path option to save as an external table
df.write.format("delta").option("path", "/mydata").saveAsTable("MyExternal-
Table")
```

Creating a catalog table using SQL

You can also create a catalog table by using the `CREATE TABLE` SQL statement with the `USING DELTA` clause, and an optional `LOCATION` parameter for external tables. You can run the statement using the SparkSQL API, like the following example:

Python

```
spark.sql("CREATE TABLE MyExternalTable USING DELTA LOCATION '/mydata'")
```

Alternatively you can use the native SQL support in Spark to run the statement:

SQL

```
%sql  
  
CREATE TABLE MyExternalTable  
USING DELTA  
LOCATION '/mydata'
```

💡 Tip

The `CREATE TABLE` statement returns an error if a table with the specified name already exists in the catalog. To mitigate this behavior, you can use a `CREATE TABLE IF NOT EXISTS` statement or the `CREATE OR REPLACE TABLE` statement.

Defining the table schema

In all of the examples so far, the table is created without an explicit schema. In the case of tables created by writing a dataframe, the table schema is inherited from the dataframe. When creating an external table, the schema is inherited from any files that are currently stored in the table location. However, when creating a new managed table, or an external table with a currently empty location, you define the table schema by specifying the column names, types, and nullability as part of the `CREATE TABLE` statement; as shown in the following example:

SQL

```
%sql  
  
CREATE TABLE ManagedSalesOrders  
(  
    Orderid INT NOT NULL,  
    OrderDate TIMESTAMP NOT NULL,
```

```
    CustomerName STRING,  
    SalesTotal FLOAT NOT NULL  
)  
USING DELTA
```

When using Delta Lake, table schemas are enforced - all inserts and updates must comply with the specified column nullability and data types.

Using the DeltaTableBuilder API

You can use the DeltaTableBuilder API (part of the Delta Lake API) to create a catalog table, as shown in the following example:

Python

```
from delta.tables import *  
  
DeltaTable.create(spark) \  
  .tableName("default.ManagedProducts") \  
  .addColumn("Productid", "INT") \  
  .addColumn("ProductName", "STRING") \  
  .addColumn("Category", "STRING") \  
  .addColumn("Price", "FLOAT") \  
  .execute()
```

Similarly to the CREATE TABLE SQL statement, the create method returns an error if a table with the specified name already exists. You can mitigate this behavior by using the createIfNotExists or createOrReplace method.

Using catalog tables

You can use catalog tables like tables in any SQL-based relational database, querying and manipulating them by using standard SQL statements. For example, the following code example uses a SELECT statement to query the **ManagedSalesOrders** table:

SQL

```
%sql  
  
SELECT orderid, salestotal  
FROM ManagedSalesOrders
```

 Tip

For more information about working with Delta Lake, see [Table batch reads and writes](#) in the Delta Lake documentation.

Next unit: Use Delta Lake for streaming data

[Continue >](#)

How are we doing?

Use Delta Lake for streaming data

5 minutes

All of the data we've explored up to this point has been static data in files. However, many data analytics scenarios involve *streaming* data that must be processed in near real time. For example, you might need to capture readings emitted by internet-of-things (IoT) devices and store them in a table as they occur.

Spark Structured Streaming

A typical stream processing solution involves constantly reading a stream of data from a *source*, optionally processing it to select specific fields, aggregate and group values, or otherwise manipulate the data, and writing the results to a *sink*.

Spark includes native support for streaming data through *Spark Structured Streaming*, an API that is based on a boundless dataframe in which streaming data is captured for processing. A Spark Structured Streaming dataframe can read data from many different kinds of streaming source, including network ports, real time message brokering services such as Azure Event Hubs or Kafka, or file system locations.

💡 Tip

For more information about Spark Structured Streaming, see [Structured Streaming Programming Guide](#) in the Spark documentation.

Streaming with Delta Lake tables

You can use a Delta Lake table as a source or a sink for Spark Structured Streaming. For example, you could capture a stream of real time data from an IoT device and write the stream directly to a Delta Lake table as a sink - enabling you to query the table to see the latest streamed data. Or, you could read a Delta Table as a streaming source, enabling you to constantly report new data as it is added to the table.

Using a Delta Lake table as a streaming source

In the following PySpark example, a Delta Lake table is used to store details of Internet sales orders. A stream is created that reads data from the Delta Lake table folder as new data is appended.

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Load a streaming dataframe from the Delta Table
stream_df = spark.readStream.format("delta") \
    .option("ignoreChanges", "true") \
    .load("/delta/internetorders")

# Now you can process the streaming data in the dataframe
# for example, show it:
stream_df.show()
```

(!) Note

When using a Delta Lake table as a streaming source, only *append* operations can be included in the stream. Data modifications will cause an error unless you specify the `ignoreChanges` or `ignoreDeletes` option.

After reading the data from the Delta Lake table into a streaming dataframe, you can use the Spark Structured Streaming API to process it. In the example above, the dataframe is simply displayed; but you could use Spark Structured Streaming to aggregate the data over temporal windows (for example to count the number of orders placed every minute) and send the aggregated results to a downstream process for near-real-time visualization.

Using a Delta Lake table as a streaming sink

In the following PySpark example, a stream of data is read from JSON files in a folder. The JSON data in each file contains the status for an IoT device in the format
`{"device": "Dev1", "status": "ok"}` New data is added to the stream whenever a file is added to the folder. The input stream is a boundless dataframe, which is then written in delta format to a folder location for a Delta Lake table.

Python

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Create a stream that reads JSON data from a folder
```

```
streamFolder = '/streamingdata/'  
jsonSchema = StructType([  
    StructField("device", StringType(), False),  
    StructField("status", StringType(), False)  
)  
stream_df =  
spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger",  
1).json(inputPath)  
  
# Write the stream to a delta table  
table_path = '/delta/devicetable'  
checkpoint_path = '/delta/checkpoint'  
delta_stream = stream_df.writeStream.format("delta").option("checkpointLo-  
cation", checkpoint_path).start(table_path)
```

ⓘ Note

The `checkpointLocation` option is used to write a checkpoint file that tracks the state of the stream processing. This file enables you to recover from failure at the point where stream processing left off.

After the streaming process has started, you can query the Delta Lake table to which the streaming output is being written to see the latest data. For example, the following code creates a catalog table for the Delta Lake table folder and queries it:

SQL

```
%sql  
  
CREATE TABLE DeviceTable  
USING DELTA  
LOCATION '/delta/devicetable';  
  
SELECT device, status  
FROM DeviceTable;
```

To stop the stream of data being written to the Delta Lake table, you can use the `stop` method of the streaming query:

Python

```
delta_stream.stop()
```

💡 Tip

For more information about using Delta Lake tables for streaming data, see [Table streaming reads and writes](#) in the Delta Lake documentation.

Next unit: Exercise - Use Delta Lake in Azure Databricks

[Continue >](#)

How are we doing?

Knowledge check

3 minutes

1. Which of the following descriptions best fits Delta Lake? *



A Spark API for exporting data from a relational database into CSV files.

X Incorrect. Delta Lake does not export data from a relational database into CSV files.



A relational storage layer for Spark that supports tables based on Parquet files.

✓ Correct. Delta Lake provides a relational storage layer in which you can create tables based on Parquet files in a data lake.



A synchronization solution that replicates data between SQL Server and Spark clusters.

2. You've loaded a Spark dataframe with data, that you now want to use in a Delta Lake table. What format should you use to write the dataframe to storage? *



CSV



PARQUET

X Incorrect. Although Delta Lake tables are based on Parquet files, the format must also include a transaction log.



DELTA

✓ Correct. Storing a dataframe in DELTA format creates parquet files for the data and the transaction log metadata necessary for Delta Lake tables.

3. What feature of Delta Lake enables you to retrieve data from previous versions of a table? *



Spark Structured Streaming



Time Travel

✓ Correct. The Time Travel feature is based on the transaction log, which enables you to specify a version number or timestamp for the data you want to retrieve.



4. You have a managed catalog table that contains Delta Lake data. If you drop the table, what will happen? *



The table metadata and data files will be deleted.

✓ Correct. The life-cycle of the metadata and data for a managed table are the same.



The table metadata will be removed from the catalog, but the data files will remain intact.

✗ Incorrect. The life-cycle of the metadata and data for a managed table are the same.



The table metadata will remain in the catalog, but the data files will be deleted.

5. When using Spark Structured Streaming, a Delta Lake table can be which of the following? *



Only a source



Only a sink

✗ Incorrect. A Delta Lake table can be a source or a sink.



Either a source or a sink

✓ Correct. A Delta Lake table can be a source or a sink.

Next unit: Summary

[Continue >](#)

How are we doing?

100 XP



Introduction

1 minute

Data analysts often use SQL to query relational data and create reports and dashboards. Azure Databricks provides support for SQL-based data analytics through SQL Warehouses and the SQL persona.

In this module, you'll learn how to:

- Create and configure SQL Warehouses in Azure Databricks.
- Create databases and tables.
- Create queries and dashboards.

Note

SQL Warehouses and the SQL persona are available in *premium-tier* Azure Databricks workspaces.

Next unit: Get started with SQL Warehouses

[Continue >](#)

How are we doing?

100 XP

Get started with SQL Warehouses

3 minutes

SQL Warehouses (formerly known as SQL Endpoints) provide a relational database interface for data in Azure Databricks. The data is stored in files that are abstracted by Delta tables in a hive metastore, but from the perspective of the user or client application, the SQL Warehouse behaves like a relational database.

Creating a SQL Warehouse

When you create a premium-tier Azure Databricks workspace, it includes a default SQL Warehouse named **Starter Warehouse**, which you can use to explore sample data and get started with SQL-based data analytics in Azure Databricks. You can modify the configuration of the default SQL Warehouse to suit your needs, or you can create more SQL Warehouses in your workspace.

You can manage the SQL Warehouses in your Azure Databricks workspace by using the Azure Databricks portal in the **SQL** persona view.

The screenshot shows the Azure Databricks Compute interface. On the left, there's a sidebar with a 'New' button and sections for Workspace, Recents, Data, Workflows, and Compute. Under SQL, 'SQL Warehouses' is selected. The main area is titled 'Compute' and has tabs for All-purpose compute, Job compute, and SQL warehouses (which is selected). It includes filters for Filter SQL warehouses, Only my SQL warehouses, Created by, Size, and a 'Create SQL warehouse' button. A table lists existing SQL warehouses: Starter Warehouse, created by 'databricks', 2X-Small size, 1/1 active, and Pro type. At the bottom right, there are navigation buttons for 1, 20 / page, and a dropdown.

SQL Warehouse configuration settings

When you create or configure a SQL Warehouse, you can specify the following settings:

- **Name:** A name used to identify the SQL Warehouse.
- **Cluster size:** Choose from a range of standard sizes to control the number and size of compute resources used to support the SQL Warehouse. Available sizes range from *2X-Small* (a single worker node) to *4X-Large* (256 worker nodes). For more information, see [Cluster size](#) in the Azure Databricks documentation.
- **Auto Stop:** The amount of time the cluster will remain running when idle before being stopped. Idle clusters continue to incur charges when running.
- **Scaling:** The minimum and maximum number of clusters used to distribute query processing.
- **Type:** You can create a SQL Warehouse that uses *serverless* compute for fast, cost-effective on-demand provisioning. Alternatively, you can create a *Pro* or *Classic* SQL warehouse.

! Note

You can create a SQL Warehouse with any available size, but if you have insufficient quota for the number of cores required to support your choice in the region where Azure

Databricks is provisioned, the SQL Warehouse will fail to start.

Next unit: Create databases and tables

[Continue >](#)

How are we doing?

✓ 100 XP



Create databases and tables

3 minutes

After creating and starting a SQL Warehouse, you can start to work with data in tables.

Database schema

All SQL Warehouses contain a default database schema named **default**. You can use create tables in this schema in order to analyze data. However, if you need to work with multiple tables in a relational schema, or you have multiple analytical workloads where you want to manage the data (and access to it) separately, you can create custom database schema. To create a database, use the SQL editor to run a `CREATE DATABASE` or `CREATE SCHEMA` SQL statement. These statements are equivalent, but `CREATE SCHEMA` is preferred, as shown in this example:

SQL

```
CREATE SCHEMA salesdata;
```

💡 Tip

For more information, see [CREATE SCHEMA](#) in the Azure Databricks documentation.

Tables

You can use the user interface in the Azure Databricks portal to upload delimited data, or import data from a wide range of common data sources. The imported data is stored in files in Databricks File System (DBFS) storage, and a Delta table is defined for it in the Hive metastore.

If the data files already exist in storage, or you need to define an explicit schema for the table, you can use a `CREATE TABLE` SQL statement. For example, the following code creates a table named **salesorders** in the **salesdata** database, based on the `/data/sales/` folder in DBFS storage.

SQL

```
CREATE TABLE salesdata.salesorders
(
    orderid INT,
    orderdate DATE,
    customerid INT,
    ordertotal DECIMAL
)
USING DELTA
LOCATION '/data/sales/';
```

Tip

For more information, see [CREATE TABLE](#) in the Azure Databricks documentation.

Next unit: Create queries and dashboards

[Continue >](#)

How are we doing?     

✓ 100 XP



Create queries and dashboards

3 minutes

Azure Databricks SQL is primarily designed for data analytics and visualization workloads. To support these workloads, users can create *queries* to retrieve and summarize data from tables, and *dashboards* to share visualizations of the data.

Queries

You can use the SQL Editor in the Azure Databricks portal to create a query based on any valid SQL SELECT statement, and then save the query with a meaningful name to be retrieved and run later.

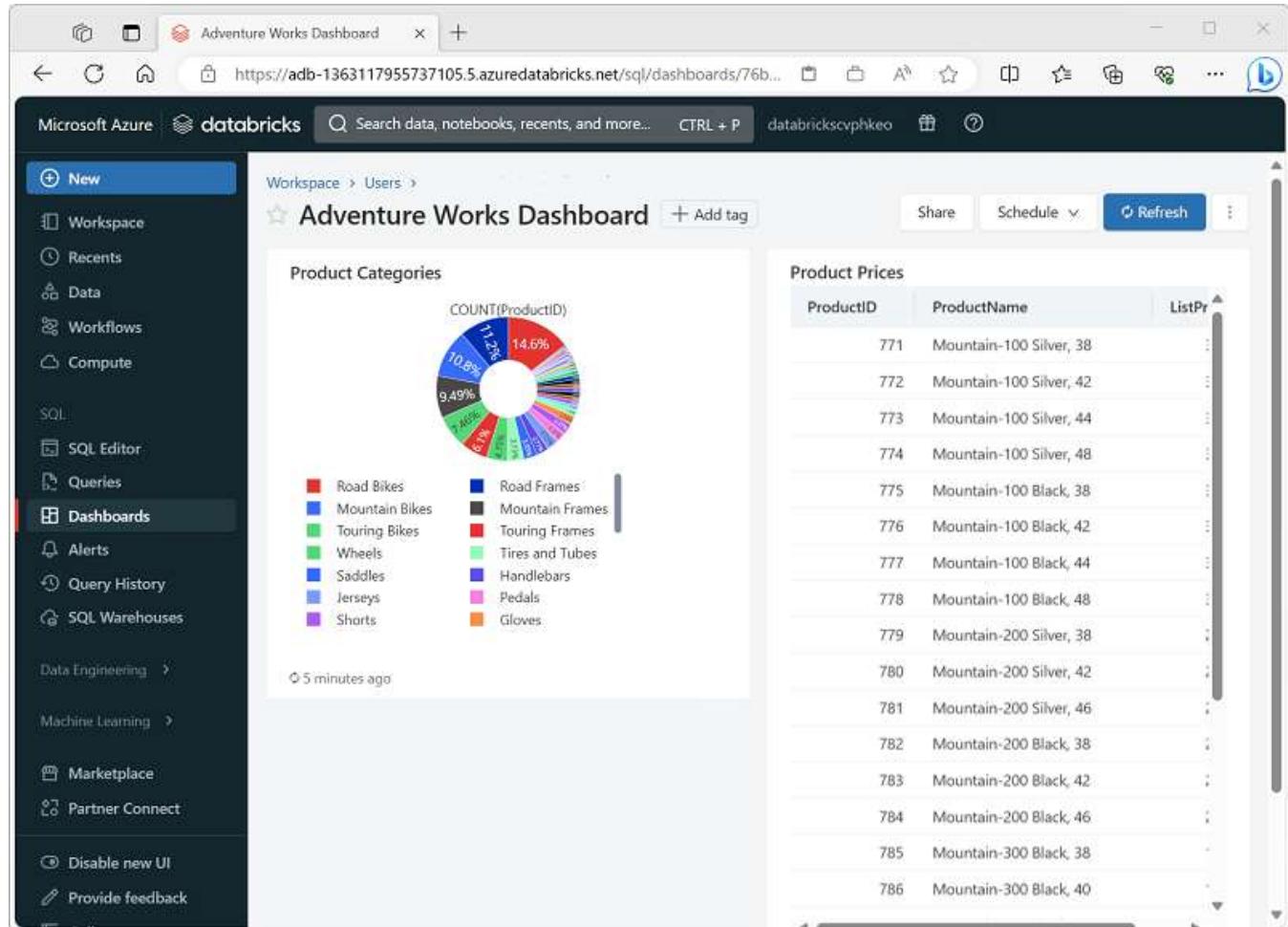
The screenshot shows the Azure Databricks SQL Editor interface. On the left, there's a sidebar with navigation links like 'New', 'Workspace', 'Recents', 'Data', 'Workflows', 'Compute', 'SQL', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', 'Machine Learning', 'Marketplace', 'Partner Connect', 'Disable new UI', and 'Provide feedback'. The main area has tabs for 'Data' and 'SQL Editor'. In the 'SQL Editor' tab, a query is being run: 'SELECT * FROM products'. The results table shows data for 295 rows, with columns: #, ProductID, ProductName, Category, and ListPrice. The data includes various mountain bike models like 'Mountain-100 Silver' and 'Mountain-100 Black' at different price points. At the bottom, it says '14 s 442 ms | 295 rows returned' and 'Refreshed 3 minutes ago'.

#	ProductID	ProductName	Category	ListPrice
1	771	Mountain-100 Silver, 38	Mountain Bikes	3399.99
2	772	Mountain-100 Silver, 42	Mountain Bikes	3399.99
3	773	Mountain-100 Silver, 44	Mountain Bikes	3399.99
4	774	Mountain-100 Silver, 48	Mountain Bikes	3399.99
5	775	Mountain-100 Black, 38	Mountain Bikes	3374.99
6	776	Mountain-100 Black, 42	Mountain Bikes	3374.99
7	777	Mountain-100 Black, 44	Mountain Bikes	3374.99

After saving the query, you can schedule it to be run automatically at regular intervals to refresh the data, or you can open it and run it interactively.

Dashboards

Dashboards enable you to display the results of queries, either as tables of data or as graphical visualizations.



You can create multiple visualizations in a dashboard and share it with users in your organization. As with individual queries, you can schedule the dashboard to refresh its data periodically, and notify subscribers by email that new data is available.

Next unit: Exercise - Use a SQL Warehouse in Azure Databricks

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP



Exercise - Use a SQL Warehouse in Azure Databricks

30 minutes

Now it's your chance to explore Azure Databricks SQL for yourself. In this exercise, you'll use a SQL Warehouse in Azure Databricks to query tables and create a dashboard.

Note

To complete this lab, you will need an [Azure subscription](#) in which you have administrative access.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

Next unit: Knowledge check

[Continue >](#)

How are we doing?

✓ 200 XP



Knowledge check

3 minutes

1. Which of the following workloads is best suited for Azure Databricks SQL? *

Running Scala code in notebooks to transform data.

X Incorrect. Using notebooks to transform data is more suited to the Azure Databricks Data Science and Engineering persona.

Querying and visualizing data in relational tables.

✓ Correct. Azure Databricks SQL is optimized for SQL-based querying and data visualization.

Training and deploying machine learning models.

2. Which statement should you use to create a database in a SQL warehouse? *

CREATE VIEW

CREATE SCHEMA

✓ Correct. The CREATE SCHEMA statement is used to create a database.

CREATE GROUP

3. You need to share data visualizations, including charts and tables of data, with users in your organization. What should you create? *

A table

X Incorrect. Creating a table does not enable you to share data visualizations.

A query

A dashboard

✓ Correct. A dashboard can be used to share data visualizations with other users.

100 XP

Introduction

1 minute

Azure Databricks enables data engineers to use code in notebooks to ingest and process data. While notebooks are designed to be used interactively, you can use them to encapsulate activities in a data ingestion or processing pipeline that is orchestrated using Azure Data Factory.

In this module, you'll learn how to:

- Describe how Azure Databricks notebooks can be run in a pipeline.
- Create an Azure Data Factory linked service for Azure Databricks.
- Use a Notebook activity in a pipeline.
- Pass parameters to a notebook.

Note

While this module focuses on using Azure Databricks notebooks in an Azure Data Factory pipeline, the same principles and techniques apply when using an Azure Synapse Analytics pipeline.

Next unit: Understand Azure Databricks notebooks and pipelines

[Continue >](#)

How are we doing?

✓ 100 XP

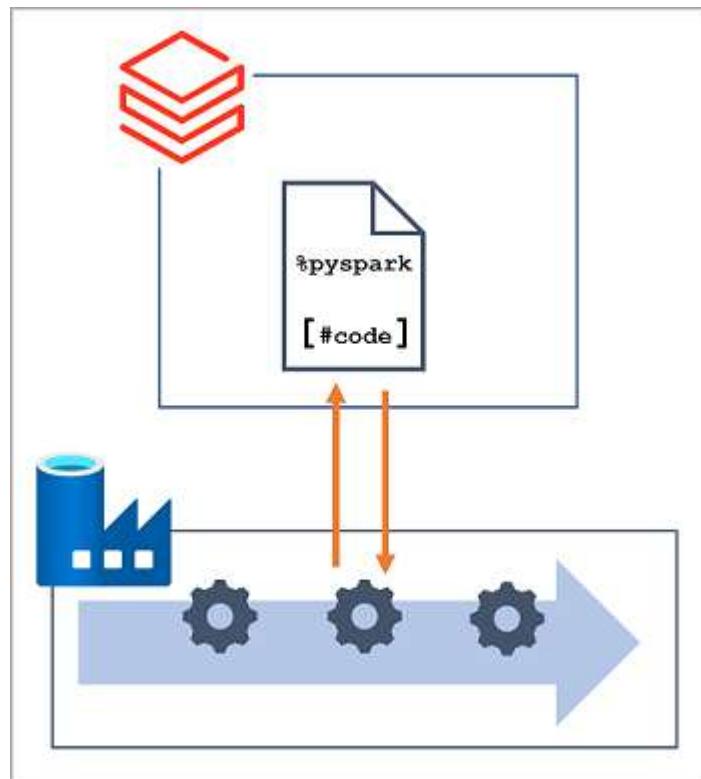


Understand Azure Databricks notebooks and pipelines

3 minutes

In Azure Databricks, you can use notebooks to run code written in Python, Scala, SQL, and other languages to ingest and process data. Notebooks provide an interactive interface in which you can run individual code cells and use Markdown to include notes and annotations.

In many data engineering solutions, code that is written and tested interactively can later be incorporated into an automated data processing workload. On Azure, such workloads are often implemented as *pipelines* in Azure Data Factory, in which one or more *activities* are used to orchestrate a series of tasks that can be run on-demand, at scheduled intervals, or in response to an event (such as new data being loaded into a folder in a data lake). Azure Data Factory supports a **Notebook** activity that can be used to automate the unattended execution of a notebook in an Azure Databricks workspace.



! Note

The same **Notebook** activity is available in pipelines built in Azure Synapse Analytics.

✓ 100 XP



Create a linked service for Azure Databricks

5 minutes

To run notebooks in an Azure Databricks workspace, the Azure Data Factory pipeline must be able to connect to the workspace; which requires authentication. To enable this authenticated connection, you must perform two configuration tasks:

1. Generate an *access token* for your Azure Databricks workspace.
2. Create a *linked service* in your Azure Data Factory resource that uses the access token to connect to Azure Databricks.

Generating an access token

An access token provides an authentication method for Azure Databricks as an alternative to credentials on the form of a user name and password. You can generate access tokens for applications, specifying an expiration period after which the token must be regenerated and updated in the client applications.

To create an Access token, use the **Generate new token** option on the **Developer** tab of the **User Settings** page in Azure Databricks portal.

The screenshot shows the Microsoft Azure Databricks Settings interface. The left sidebar contains navigation links for Workspace, Recents, Data, Workflows, Compute, SQL, Machine Learning, Marketplace, Partner Connect, Disable new UI, and Provide feedback. The main content area is titled 'Access tokens' under 'User settings > Developer'. It includes a sub-navigation bar with 'User', 'Profile', 'Preferences', 'Developer' (which is selected), 'Linked accounts', and 'Notifications'. A note states: 'Personal access tokens can be used for secure authentication to the [Databricks API](#) instead of passwords.' A blue button labeled 'Generate new token' is present. Below this is a table header with columns 'Comment', 'Creation' (sorted by descending date), and 'Expiration'. A message at the bottom of the table says 'No tokens exist.'

Creating a linked service

To connect to Azure Databricks from Azure Data Factory, you need to create a linked service for **Azure Databricks** compute. You can create a linked service in the **Linked services** page in the **Manage** section of Azure Data Factory Studio.

The screenshot shows the 'New linked service' dialog in the Azure Data Factory interface. The 'Compute' tab is selected. Under the 'Compute' tab, there is a search bar labeled 'Search'. Below the search bar, there are three rows of icons representing different services:

- Row 1: Azure Batch (blue icon), Azure Data Lake Analytics (blue icon with a lightning bolt), Azure Databricks (red icon with three stacked hexagons).
- Row 2: Azure Function (yellow and blue icon with a lightning bolt), Azure HDInsight (blue icon with a hexagonal cluster), Azure Machine Learning (blue icon with a stylized tree or graph).
- Row 3: Azure Machine Learning (blue icon with a globe and a flask), Azure Synapse Analytics (blue icon with a hexagon and a circular arrow).

At the bottom of the dialog are two buttons: 'Continue' (blue) and 'Cancel' (white).

When you create an **Azure Databricks** linked service, you must specify the following configuration settings:

Setting	Description
Name	A unique name for the linked service
Description	A meaningful description
Integration runtime	The integration runtime used to run activities in this linked service. See Integration runtime in Azure Data Factory for more details.
Azure subscription	The Azure subscription in which Azure Databricks is provisioned
Databricks workspace	The Azure Databricks workspace

Setting	Description
Cluster	The Spark cluster on which activity code will be run. You can have Azure Databricks dynamically provision a <i>job cluster</i> on-demand or you can specify an existing cluster in the workspace.
Authentication type	How the linked connection will be authenticated by Azure Databricks. For example, using an access token (in which case, you need to specify the access token you generated for your workspace).
Cluster configuration	The Databricks runtime version, Python version, worker node type, and number of worker nodes for your cluster.

Next unit: Use a Notebook activity in a pipeline

[Continue >](#)

How are we doing?

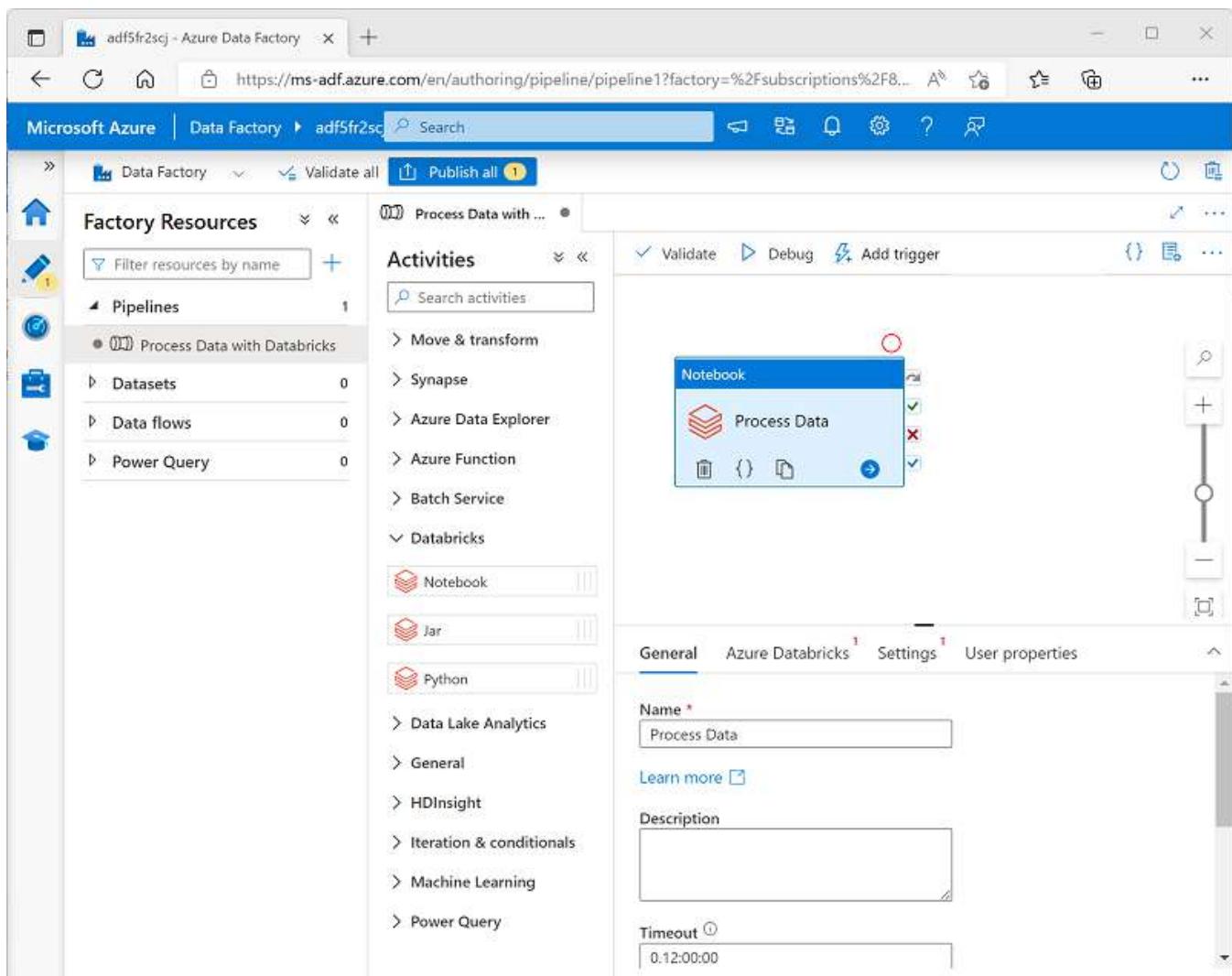
✓ 100 XP

Use a Notebook activity in a pipeline

5 minutes

After you've created a linked service in Azure Data Factory for your Azure Databricks workspace, you can use it to define the connection for a **Notebook** activity in a pipeline.

To use a **Notebook** activity, create a pipeline and from the **Databricks** category, add a **Notebook** activity to the pipeline designer surface.



Use the following properties of the **Notebook** activity to configure it:

Category	Setting	Descriptions
General	Name	A unique name for the activity.
	Description	A meaningful description.

Category	Setting	Descriptions
	Timeout	How long the activity should run before automatically canceling.
	Retries	How many times should Azure Data Factory try before failing.
	Retry interval	How long to wait before retrying.
	Secure input and output	Determines if input and output values are logged.
Azure Databricks	Azure Databricks linked service	The linked service for the Azure Databricks workspace containing the notebook.
Settings	Notebook path	The path to the notebook file in the Workspace.
	Base parameters	Used to pass parameters to the notebook.
	Append libraries	Required code libraries that aren't installed by default.
User properties		Custom user-defined properties.

Running a pipeline

When the pipeline containing the **Notebook** activity is published, you can run it by defining a trigger. You can then monitor pipeline runs in the **Monitor** section of Azure Data Factory Studio.

Next unit: Use parameters in a notebook

[Continue >](#)

100 XP

Use parameters in a notebook

3 minutes

You can use parameters to pass variable values to a notebook from the pipeline. Parameterization enables greater flexibility than using hard-coded values in the notebook code.

Using parameters in a notebook

To define and use parameters in a notebook, use the `dbutils.widgets` library in your notebook code.

For example, the following Python code defines a variable named `folder` and assigns a default value of `data`:

Python

```
dbutils.widgets.text("folder", "data")
```

To retrieve a parameter value, use the `get` function, like this:

Python

```
folder = dbutils.widgets.get("folder")
```

The `get` function will retrieve the value for the specific parameter that was passed to the notebook. If no such parameter was passed, it will get the default value of the variable you declared previously.

Passing output values

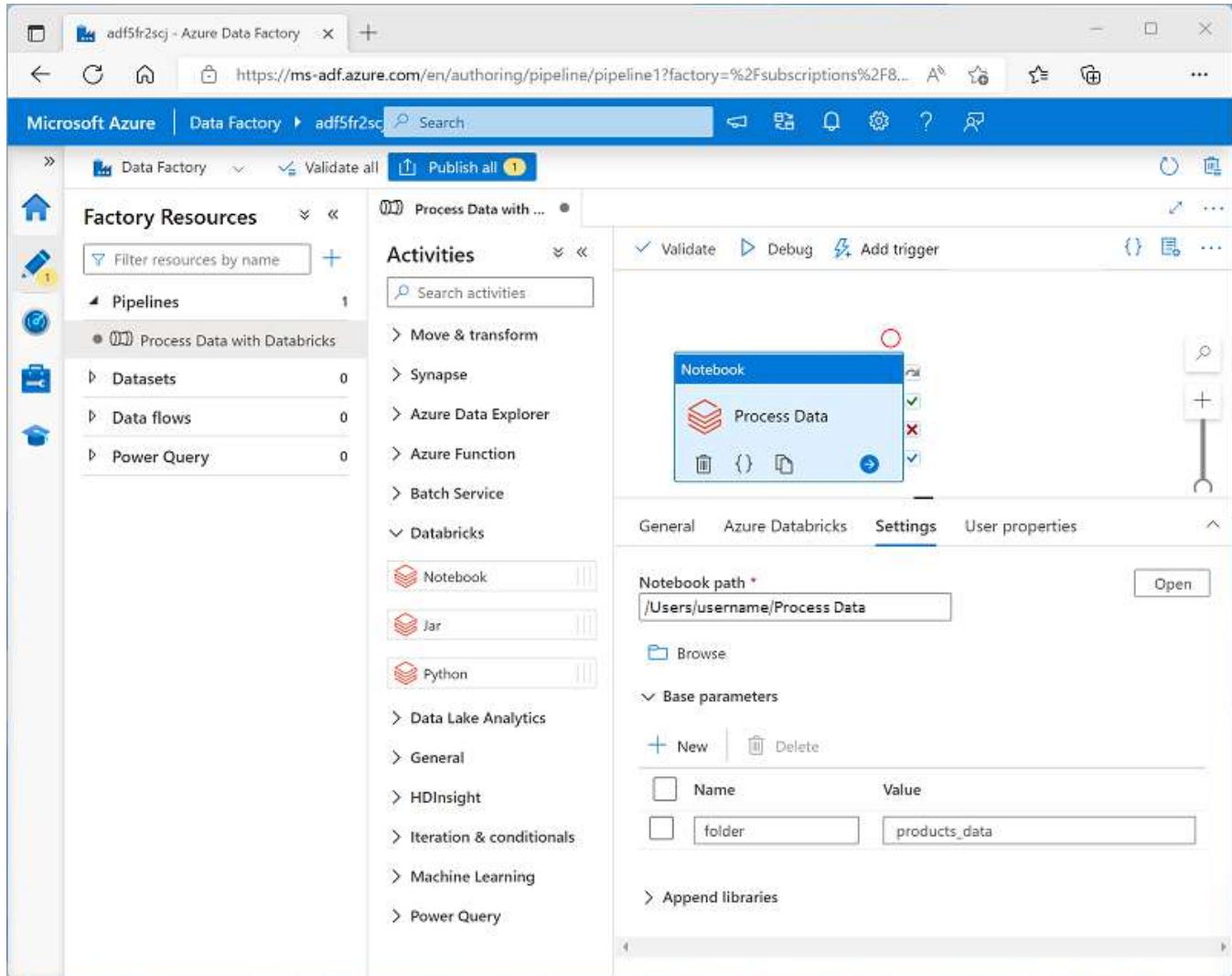
In addition to using parameters that can be passed *in* to a notebook, you can pass values *out* to the calling application by using the `notebook.exit` function, as shown here:

Python

```
path = "dbfs:/{}products.csv".format(folder)
dbutils.notebook.exit(path)
```

Setting parameter values in a pipeline

To pass parameter values to a **Notebook** activity, add each parameter to the activity's **Base parameters**, as shown here:



In this example, the parameter value is explicitly specified as a property of the **Notebook** activity. You could also define a *pipeline* parameter and assign its value dynamically to the **Notebook** activity's base parameter; adding a further level of abstraction.

Tip

For more information about using parameters in Azure Data Factory, see [How to use parameters, expressions and functions in Azure Data Factory](#) in the Azure Data Factory documentation.

Next unit: Exercise - Run an Azure Databricks Notebook with Azure Data Factory

✓ 200 XP ➔

Knowledge check

3 minutes

1. You want to connect to an Azure Databricks workspace from Azure Data Factory. What must you define in Azure Data Factory? *

A global parameter

A linked service

✓ Correct. An Azure Databricks linked service is required to connect to an Azure Databricks workspace.

A customer managed key

2. You need to run a notebook in the Azure Databricks workspace referenced by a linked service. What type of activity should you add to a pipeline? *

Notebook

✓ Correct. Use a Notebook activity to run an Azure Databricks notebook.

Python

Jar

3. You need to use a parameter in a notebook. Which library should you use to define parameters with default values and get parameter values that are passed to the notebook? *

notebook

✗ Incorrect. You can use the notebook library to exit the notebook and return an output value.

argparse

dbutils.widget

✓ Correct. Use the dbutils.widget library to define and read parameters in an Azure Databricks notebook.