LAB - Transform data using mapping data flows

In this tutorial, you'll use the Azure Data Factory user interface (UX) to create a pipeline that copies and transforms data from an Azure Data Lake Storage (ADLS) Gen2 source to an ADLS Gen2 sink using mapping data flow. The configuration pattern in this tutorial can be expanded upon when transforming data using mapping data flow

In this tutorial, you do the following steps:

- Create a data factory.
- Create a pipeline with a Data Flow activity.
- Build a mapping data flow with four transformations.
- Test run the pipeline.
- Monitor a Data Flow activity

Prerequisites

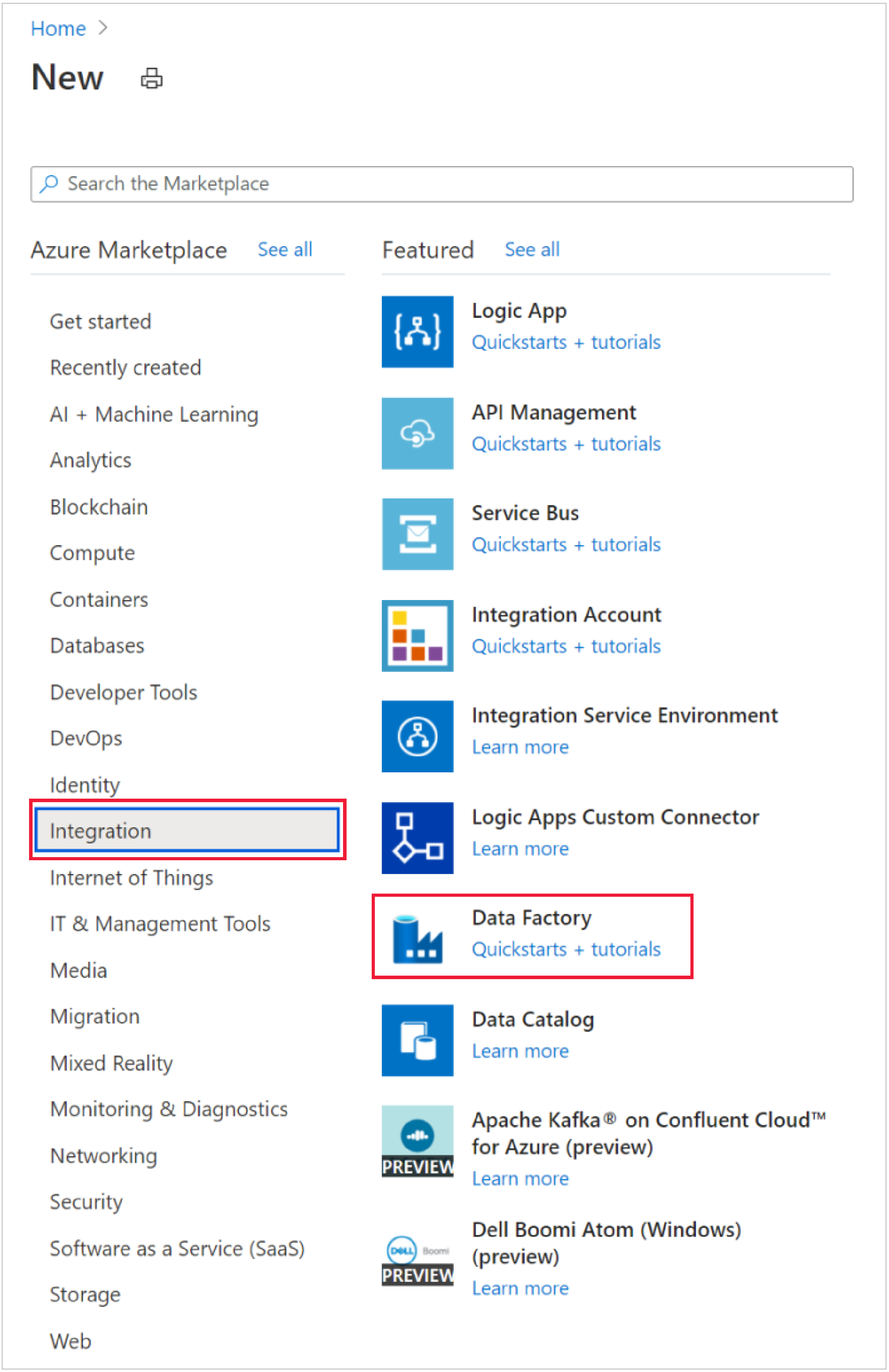
- **Azure subscription**. If you don't have an Azure subscription, create a free Azure account before you begin.
- **Azure storage account**. You use ADLS storage as a *source* and *sink* data stores. If you don't have a storage account, see Create an Azure storage account for steps to create one.

The file that we are transforming in this tutorial is MoviesDB.csv, which can be found here. To retrieve the file from GitHub, copy the contents to a text editor of your choice to save locally as a .csv file. To upload the file to your storage account, see Upload blobs with the Azure portal. The examples will be referencing a container named 'sample-data'.

Create a data factory

In this step, you create a data factory and open the Data Factory UX to create a pipeline in the data factory.

1. Open **Microsoft Edge** or **Google Chrome**. Currently, Data Factory UI is supported only in the Microsoft Edge and Google Chrome web browsers.
2. On the left menu, select **Create a resource** > **Integration** > **Data Factory**:


Home >

New

Search the Marketplace

Azure Marketplace See all

Get started
Recently created
AI + Machine Learning
Analytics
Blockchain
Compute
Containers
Databases
Developer Tools
DevOps
Identity
Integration
Internet of Things
IT & Management Tools
Media
Migration
Mixed Reality
Monitoring & Diagnostics
Networking
Security
Software as a Service (SaaS)
Storage
Web

Featured See all

Logic App
Quickstarts + tutorials

API Management
Quickstarts + tutorials

Service Bus
Quickstarts + tutorials

Integration Account
Quickstarts + tutorials

Integration Service Environment
Learn more

Logic Apps Custom Connector
Learn more

Data Factory
Quickstarts + tutorials

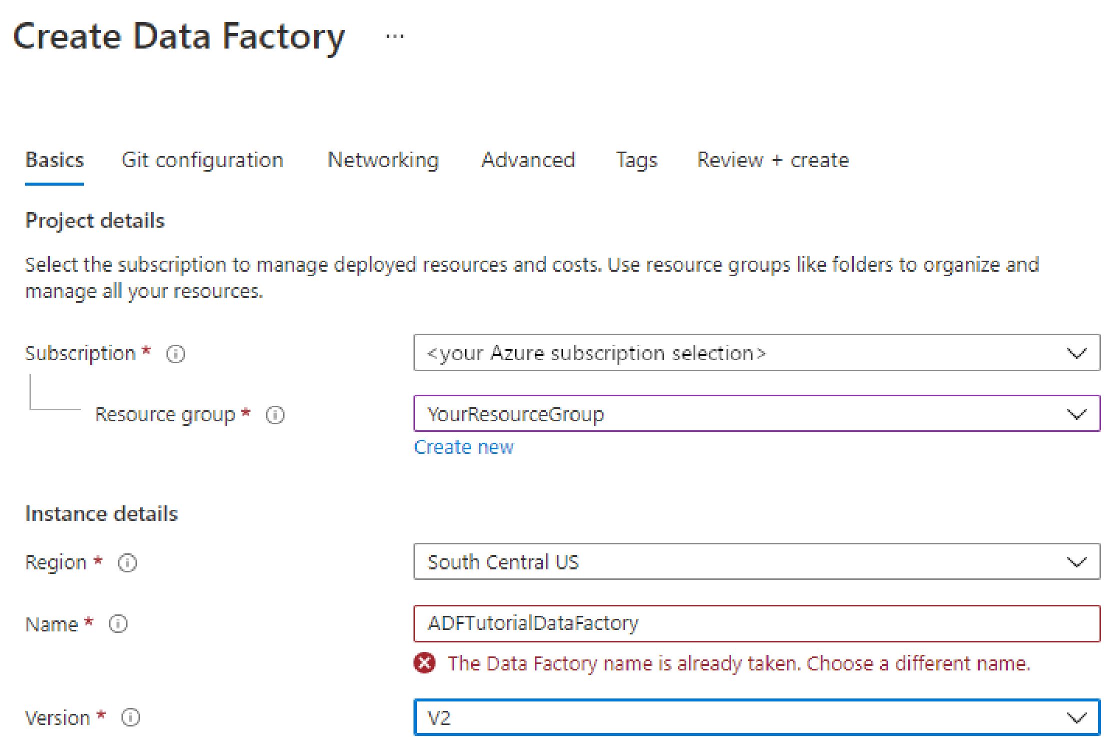
Data Catalog
Learn more

Apache Kafka® on Confluent Cloud™ for Azure (preview)
Learn more

Dell Boomi Atom (Windows) (preview)
Learn more

3. On the **New data factory** page, under **Name**, enter **ADFTutorialDataFactory**.

   The name of the Azure data factory must be *globally unique*. If you receive an error message about the name value, enter a different name for the data factory. (for example, yournameADFTutorialDataFactory). For naming rules for Data Factory artifacts, see Data Factory naming rules.


Create Data Factory ...

Basics | Git configuration | Networking | Advanced | Tags | Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : <your Azure subscription selection>

Resource group * : YourResourceGroup

Create new

Instance details

Region * : South Central US

Name * : ADFTutorialDataFactory

The Data Factory name is already taken. Choose a different name.

Version * : V2

4. Select the Azure **subscription** in which you want to create the data factory.
5. For **Resource Group**, take one of the following steps:

   a. Select **Use existing**, and select an existing resource group from the drop-down list.

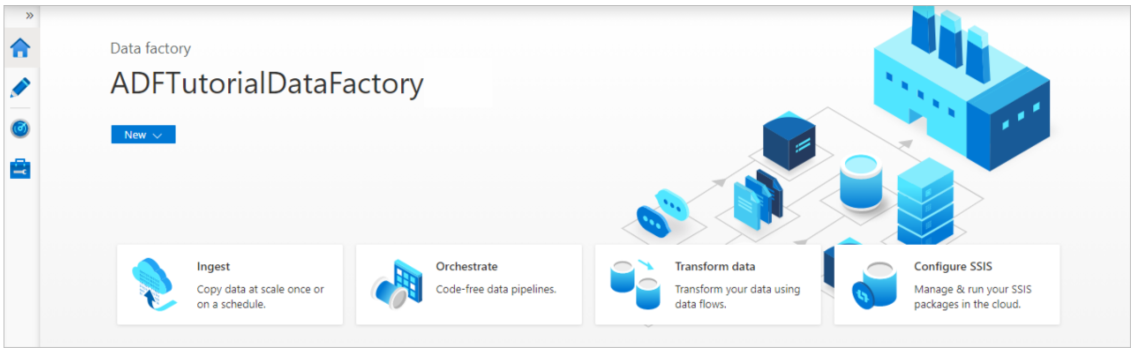
   b. Select **Create new**, and enter the name of a resource group.

   To learn about resource groups, see Use resource groups to manage your Azure resources.
6. Under **Version**, select **V2**.
7. Under **Location**, select a location for the data factory. Only locations that are supported are displayed in the drop-down list. Data stores (for example, Azure Storage and SQL Database) and computes (for example, Azure HDInsight) used by the data factory can be in other regions.
8. Select **Create**.
9. After the creation is finished, you see the notice in Notifications center. Select **Go to resource** to navigate to the Data factory page.
10. Select **Author & Monitor** to launch the Data Factory UI in a separate tab.

Create a pipeline with a Data Flow activity

In this step, you'll create a pipeline that contains a Data Flow activity.

1. On the home page of Azure Data Factory, select **Orchestrate**.

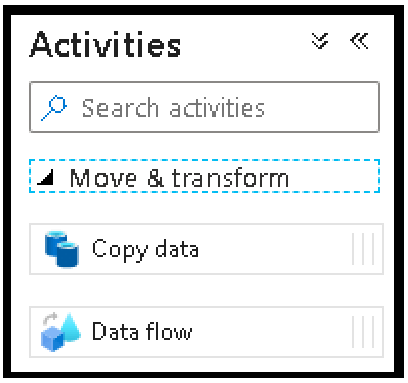

Data factory

ADFTutorialDataFactory

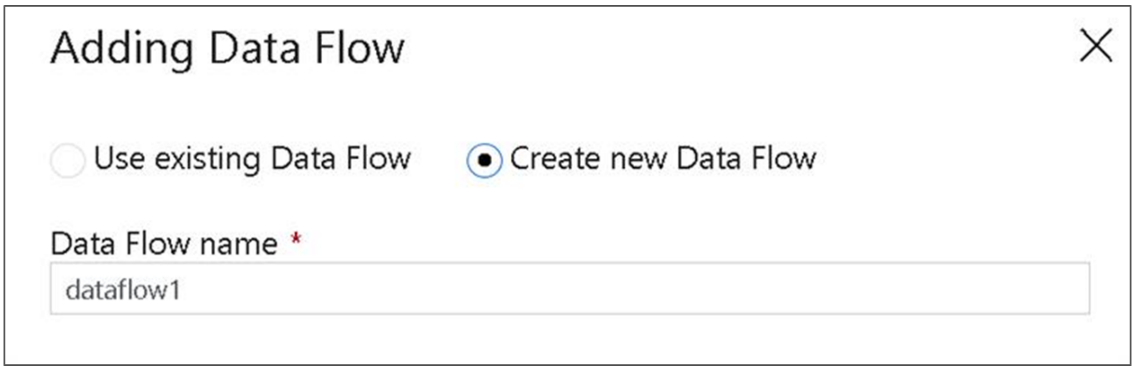
New
Ingest
Copy data at scale once or on a schedule.
Orchestrate
Code-free data pipelines.
Transform data
Transform your data using data flows.
Configure SSIS
Manage & run your SSIS packages in the cloud.

2. In the **General** tab for the pipeline, enter **TransformMovies** for **Name** of the pipeline.

3. In the **Activities** pane, expand the **Move and Transform** accordion. Drag and drop the **Data Flow** activity from the pane to the pipeline canvas.



4. In the **Adding Data Flow** pop-up, select **Create new Data Flow** and then name your data flow **TransformMovies**. Click Finish when done.

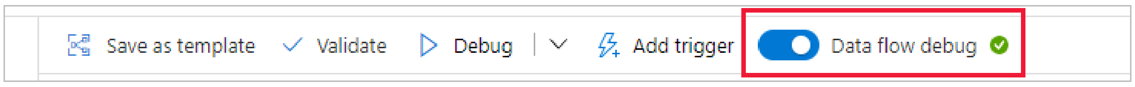

Adding Data Flow

Use existing Data Flow [✓ Create new Data Flow]

Data Flow name *

dataflow1

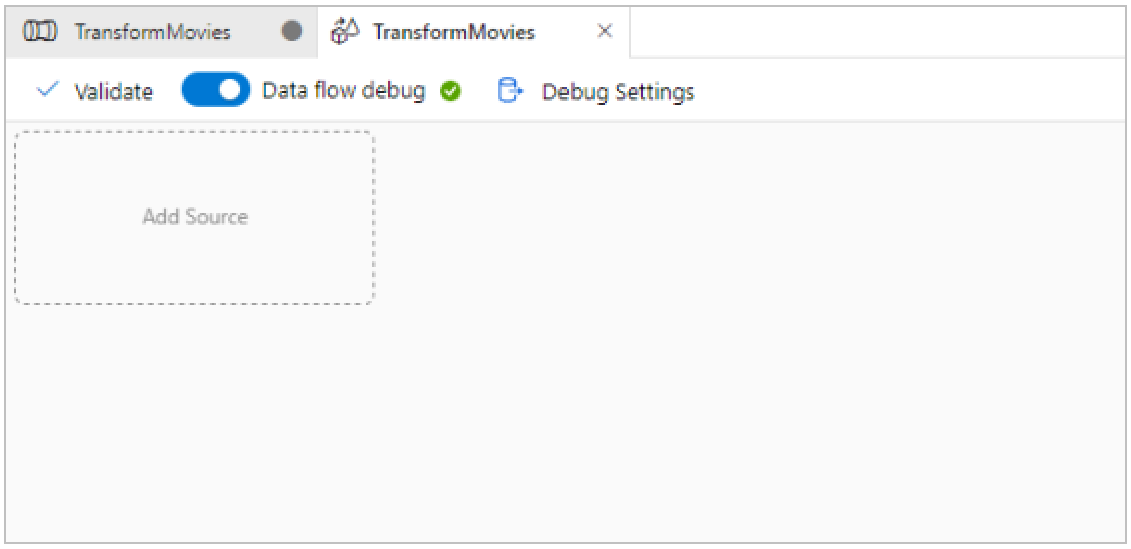
5. In the top bar of the pipeline canvas, slide the **Data Flow debug** slider on. Debug mode allows for interactive testing of transformation logic against a live Spark cluster. Data Flow clusters take 5-7 minutes to warm up and users are recommended to turn on debug first if they plan to do Data Flow development. For more information, see Debug Mode.



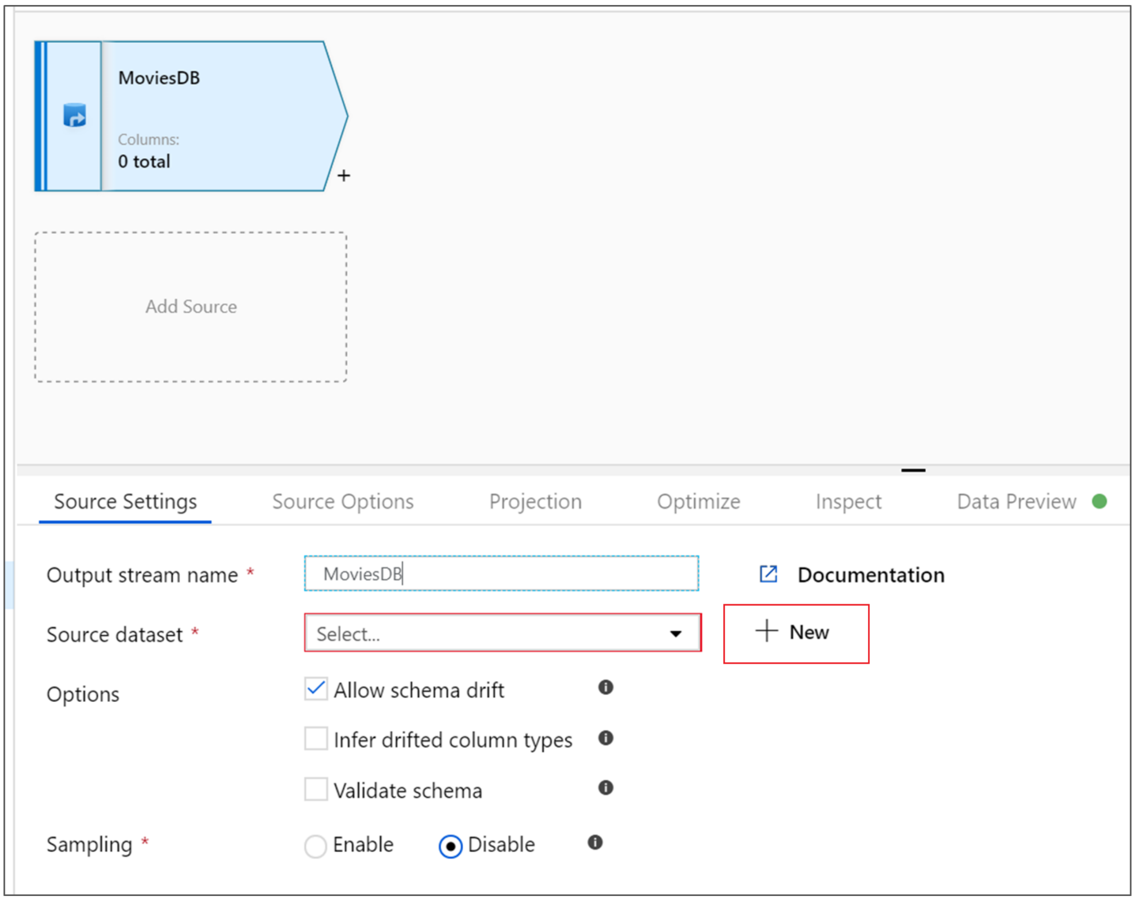
Build transformation logic in the data flow canvas

Once you create your Data Flow, you'll be automatically sent to the data flow canvas. In this step, you'll build a data flow that takes the moviesDB.csv in ADLS storage and aggregates the average rating of comedies from 1910 to 2000. You'll then write this file back to the ADLS storage.

1. In the data flow canvas, add a source by clicking on the **Add Source** box.



2. Name your source **MoviesDB**. Click on **New** to create a new source dataset.



Output stream name * MoviesDB

Source dataset * Select...

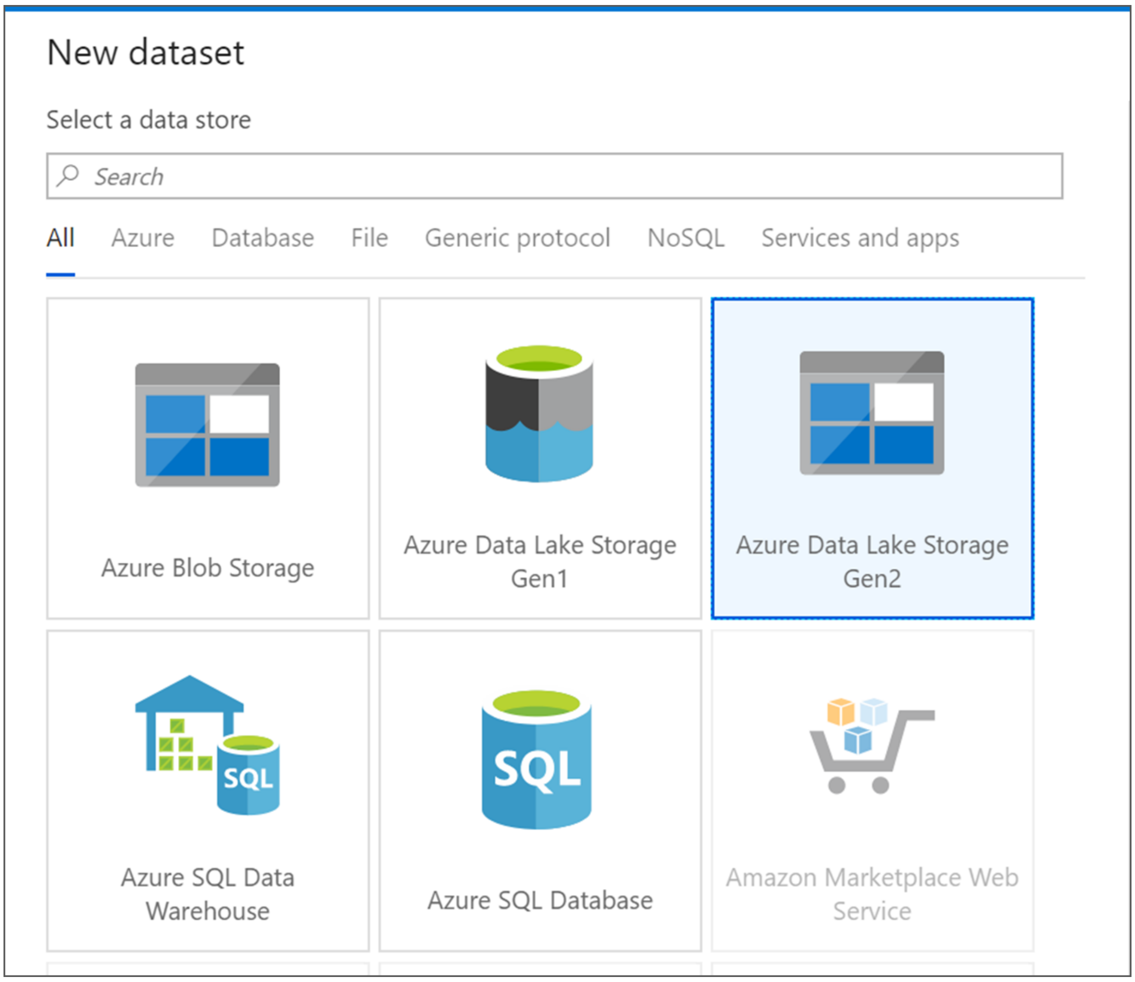
New

Documentation

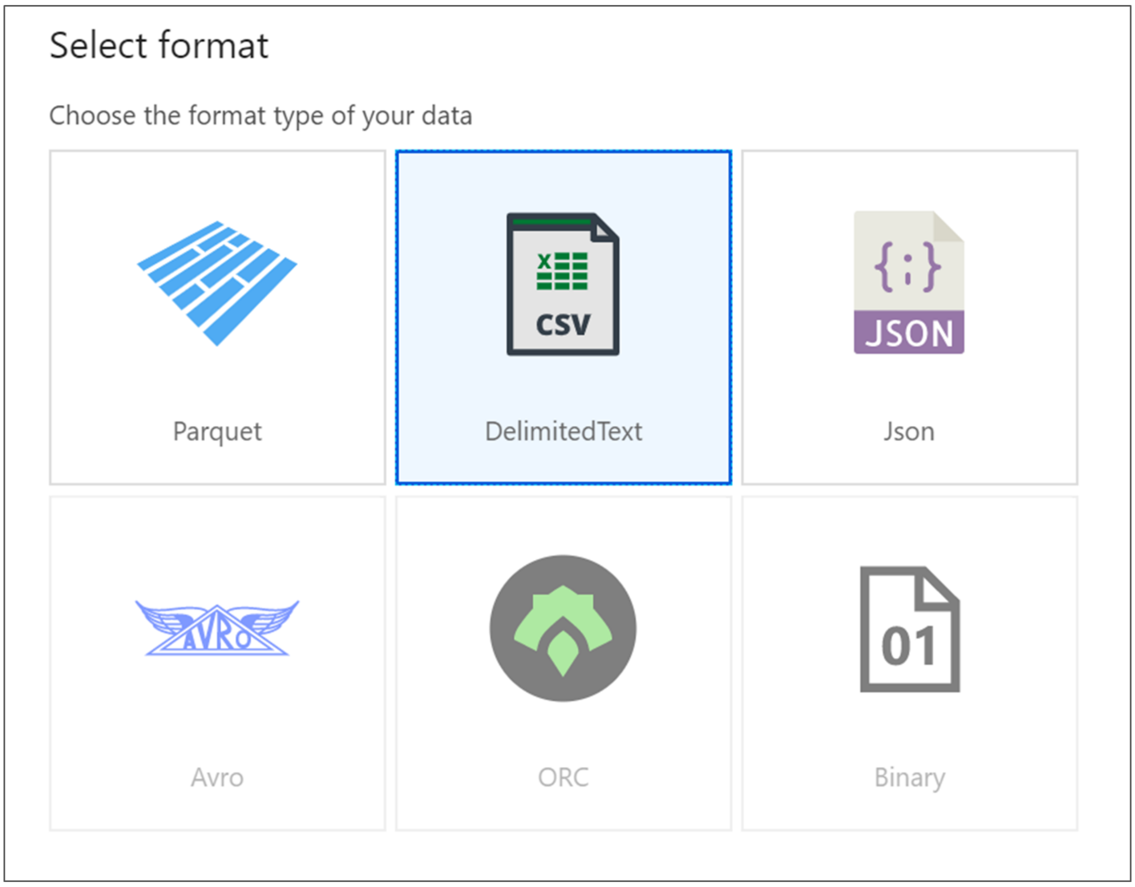
Options
- [✓] Allow schema drift
- [ ] Infer drifted column types
- [ ] Validate schema

Sampling * Enable [✓ Disable]

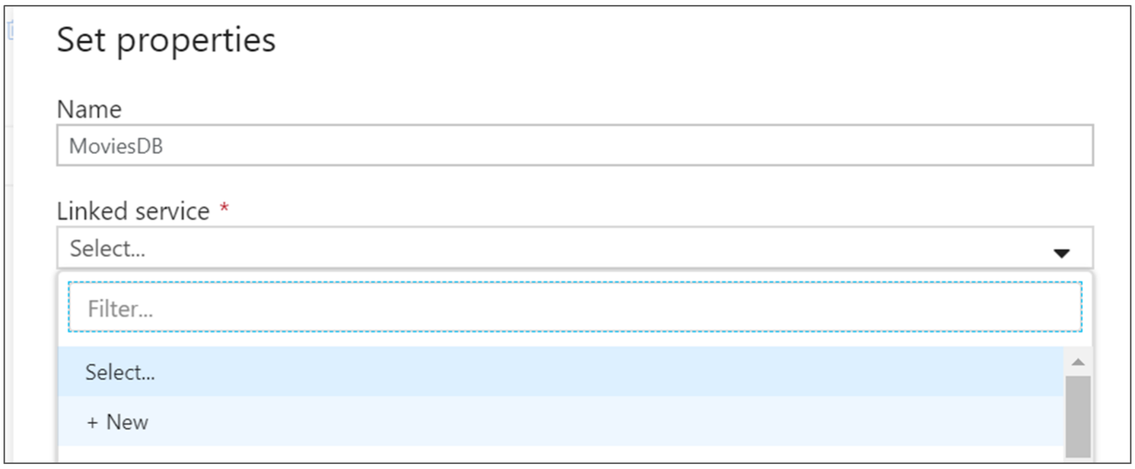
3. Choose **Azure Data Lake Storage Gen2**. Click Continue.



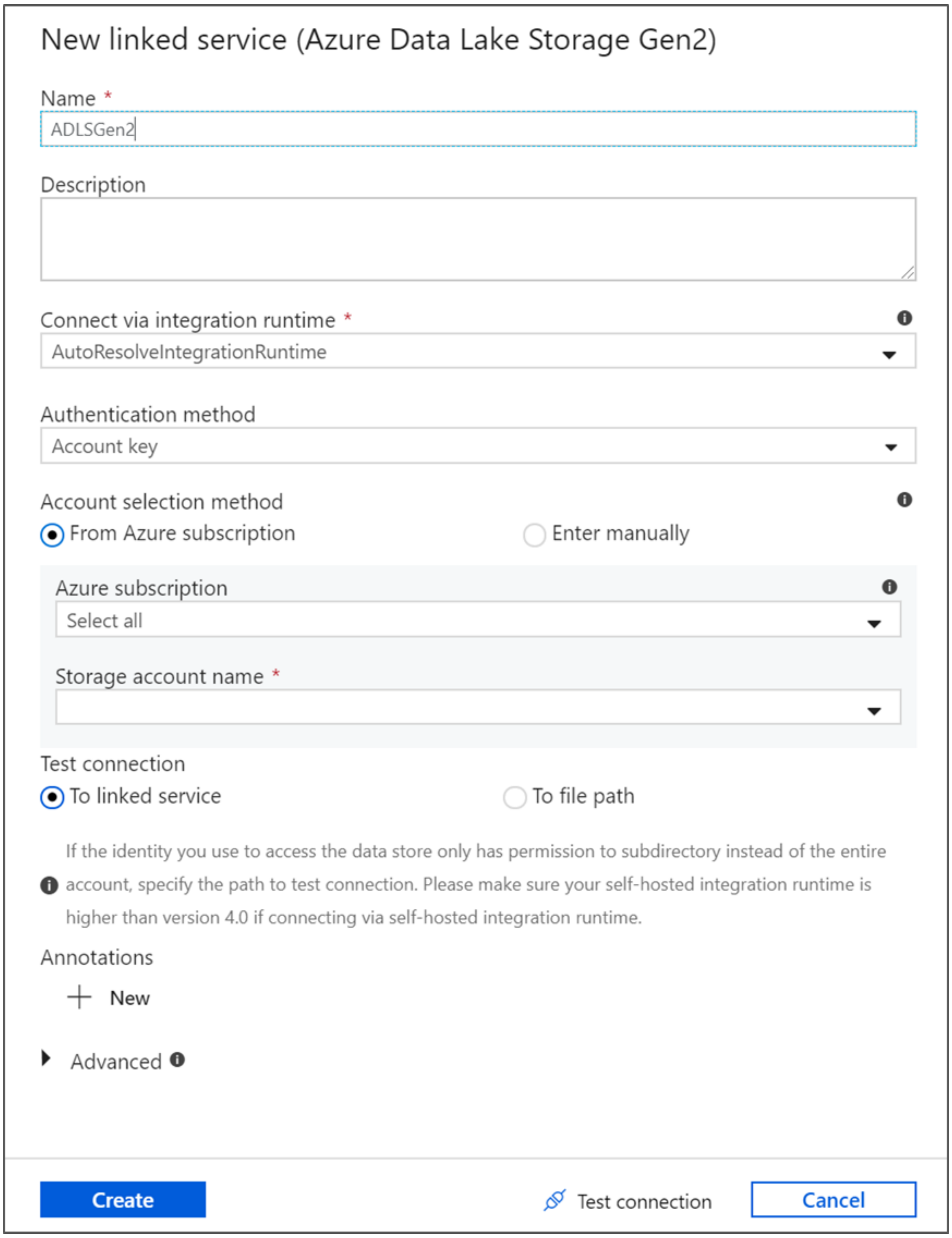
4. Choose **DelimitedText**. Click Continue.



5. Name your dataset **MoviesDB**. In the linked service dropdown, choose **New**.



6. In the linked service creation screen, name your ADLS gen2 linked service **ADLSGen2** and specify your authentication method. Then enter your connection credentials. In this tutorial, we're using Account key to connect to our storage account. You can click **Test connection** to verify your credentials were entered correctly. Click Create when finished.


New linked service (Azure Data Lake Storage Gen2)

Name *
ADLSGen2

Description

Connect via integration runtime *
AutoResolveIntegrationRuntime

Authentication method
Account key

Account selection method
[✓ From Azure subscription] Enter manually

Azure subscription
Select all

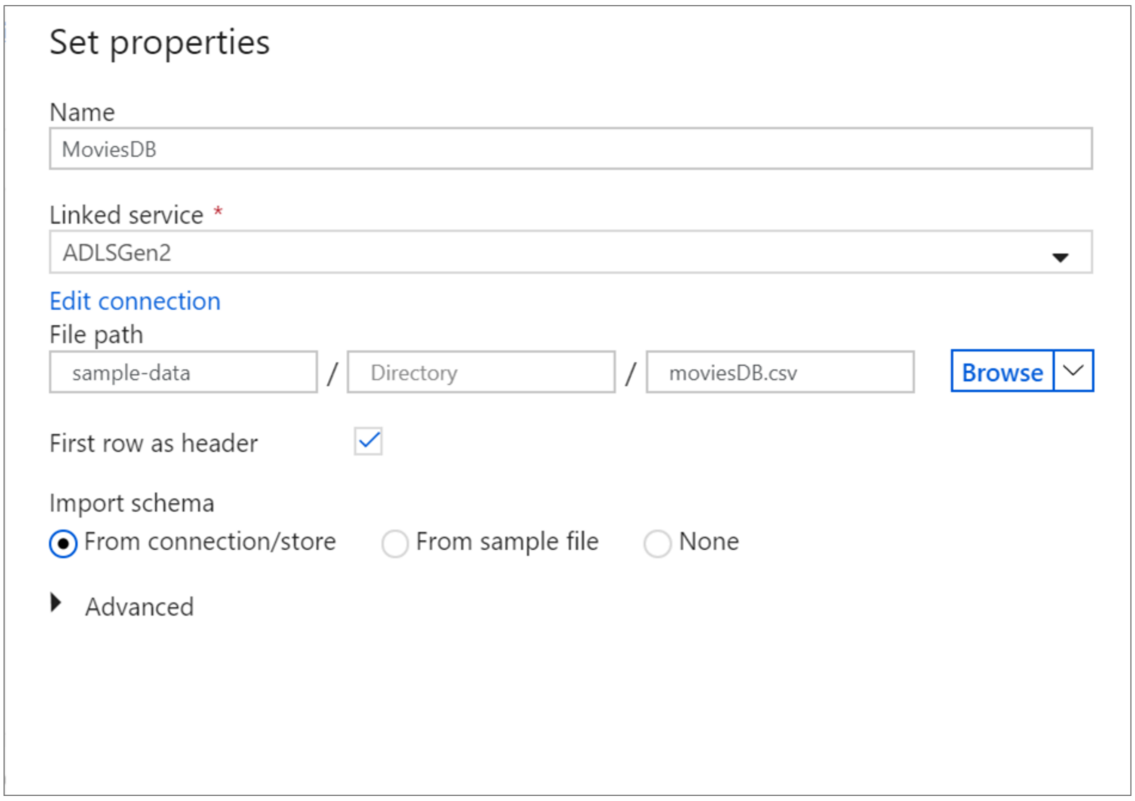
Storage account name *

Test connection
[✓ To linked service] To file path

If the identity you use to access the data store only has permission to subdirectory instead of the entire account, specify the path to test connection. Please make sure your self-hosted integration runtime is higher than version 4.0 if connecting via self-hosted integration runtime.

Annotations

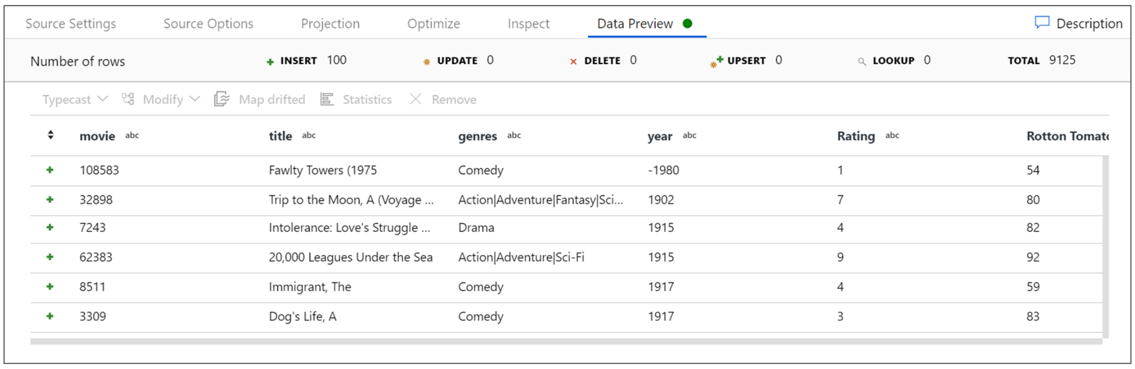
7. Once you're back at the dataset creation screen, enter where your file is located under the **File path** field. In this tutorial, the file moviesDB.csv is located in container sample-data. As the file has headers, check **First row as header**. Select **From connection/store** to import the header schema directly from the file in storage. Click OK when done.



Set properties

Name	MoviesDB
Linked service *	ADLSGen2
File path	sample-data / Directory / moviesDB.csv
First row as header	✓
Import schema	[✓ From connection/store] From sample file None

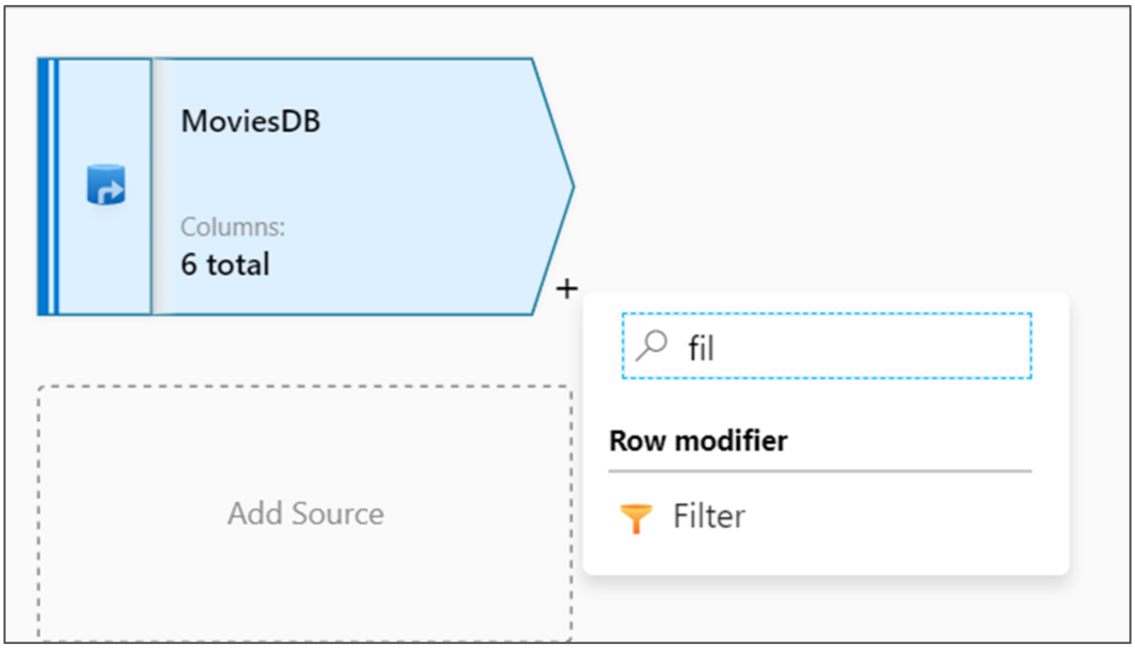
8. If your debug cluster has started, go to the **Data Preview** tab of the source transformation and click **Refresh** to get a snapshot of the data. You can use data preview to verify your transformation is configured correctly.



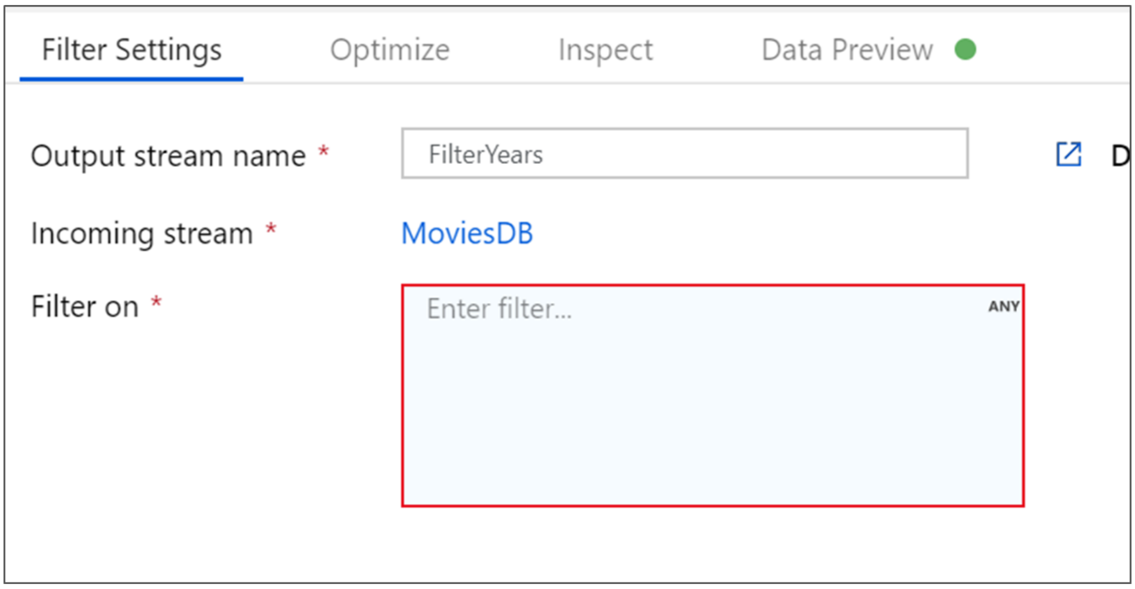
Number of rows INSERT 100 UPDATE 0 DELETE 0 UPSERT 0 LOOKUP 0 TOTAL 9125

	movie abc	title abc	genres abc	year abc	Rating abc	Rotton Tomat
+	108583	Fawlty Towers (1975	Comedy	-1980	1	54
+	32898	Trip to the Moon, A (Voyage ...	Action|Adventure|Fantasy|Sci...	1902	7	80
+	7243	Intolerance: Love's Struggle ...	Drama	1915	4	82
+	62383	20,000 Leagues Under the Sea	Action|Adventure|Sci-Fi	1915	9	92
+	8511	Immigrant, The	Comedy	1917	4	59
+	3309	Dog's Life, A	Comedy	1917	3	83

9. Next to your source node on the data flow canvas, click on the plus icon to add a new transformation. The first transformation you're adding is a **Filter**.



10. Name your filter transformation **FilterYears**. Click on the expression box next to **Filter on** to open the expression builder. Here you'll specify your filtering condition.



Output stream name * FilterYears

Incoming stream * MoviesDB

Filter on * Enter filter...

11. The data flow expression builder lets you interactively build expressions to use in various transformations. Expressions can include built-in functions, columns from the input schema, and user-defined parameters. For more information on how to build expressions, see Data Flow expression builder.

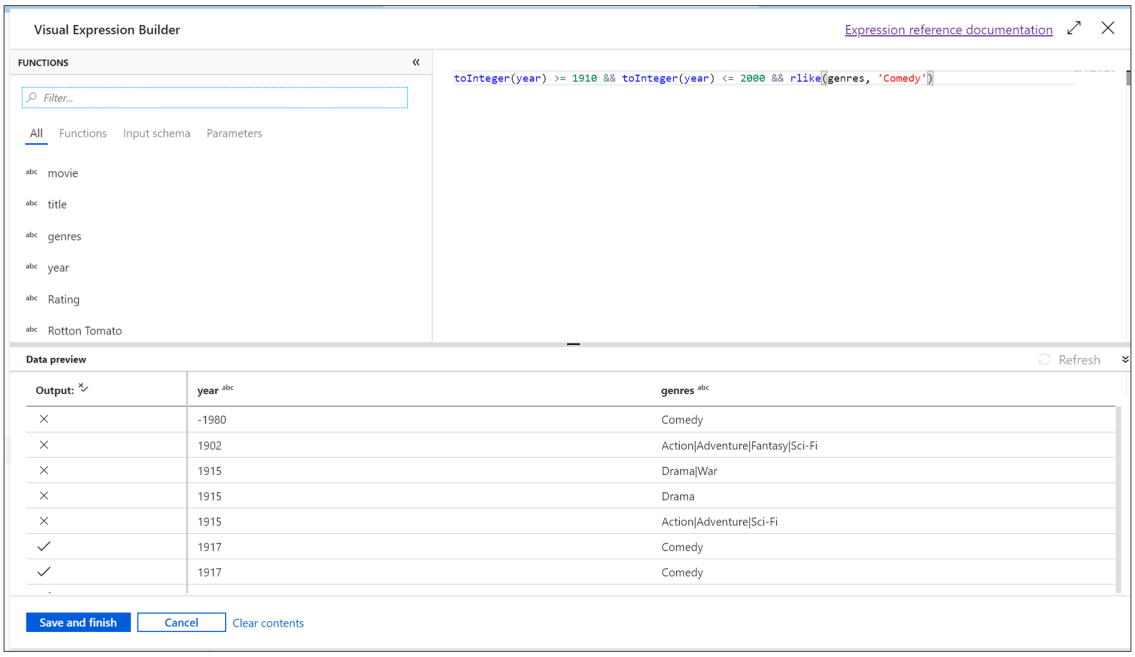
    In this tutorial, you want to filter movies of genre comedy that came out between the years 1910 and 2000. As year is currently a string, you need to convert it to an integer using the `toInteger()` function. Use the greater than or equals to (>=) and less than or equals to (<=) operators to compare against literal year values 1910 and 2000. Union these expressions together with the and (&&) operator. The expression comes out as:

    `toInteger(year) >= 1910 && toInteger(year) <= 2000`

    To find which movies are comedies, you can use the `rlike()` function to find pattern 'Comedy' in the column genres. Union the rlike expression with the year comparison to get:

    `toInteger(year) >= 1910 && toInteger(year) <= 2000 && rlike(genres, 'Comedy')`

    If you've a debug cluster active, you can verify your logic by clicking **Refresh** to see expression output compared to the inputs used. There's more than one right answer on how you can accomplish this logic using the data flow expression language.

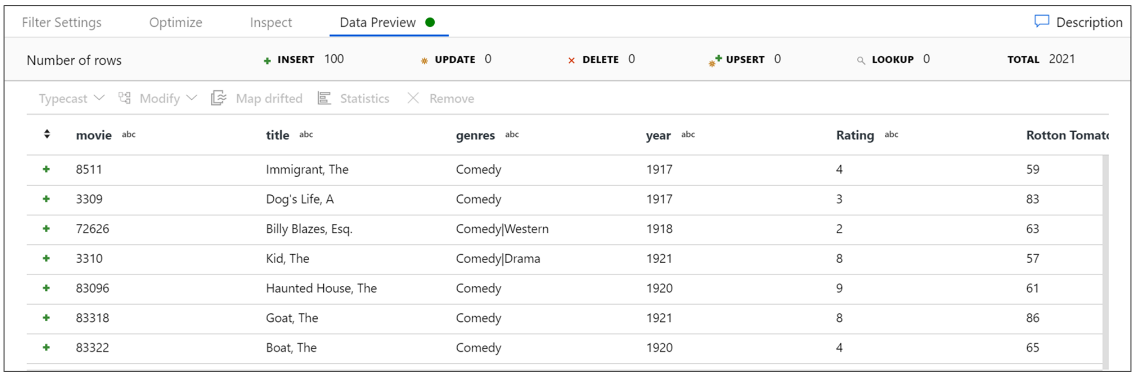


```
toInteger(year) >= 1910 && toInteger(year) <= 2000 && rlike(genres, 'Comedy')
```

Output:	year	genres
✗	-1980	Comedy
✗	1902	Action|Adventure|Fantasy|Sci-Fi
✗	1915	Drama|War
✗	1915	Drama
✗	1915	Action|Adventure|Sci-Fi
✓	1917	Comedy
✓	1917	Comedy

Click **Save and Finish** once you're done with your expression.

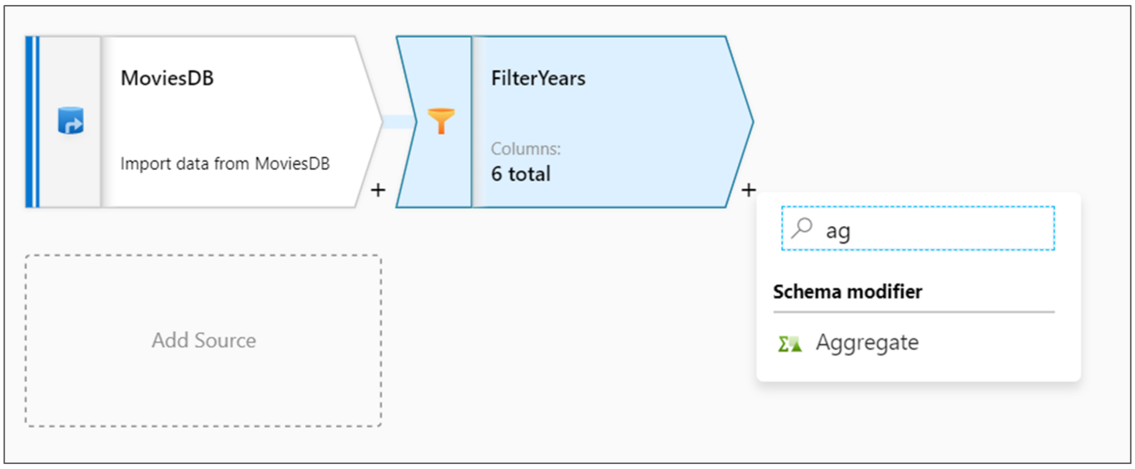
12. Fetch a **Data Preview** to verify the filter is working correctly.



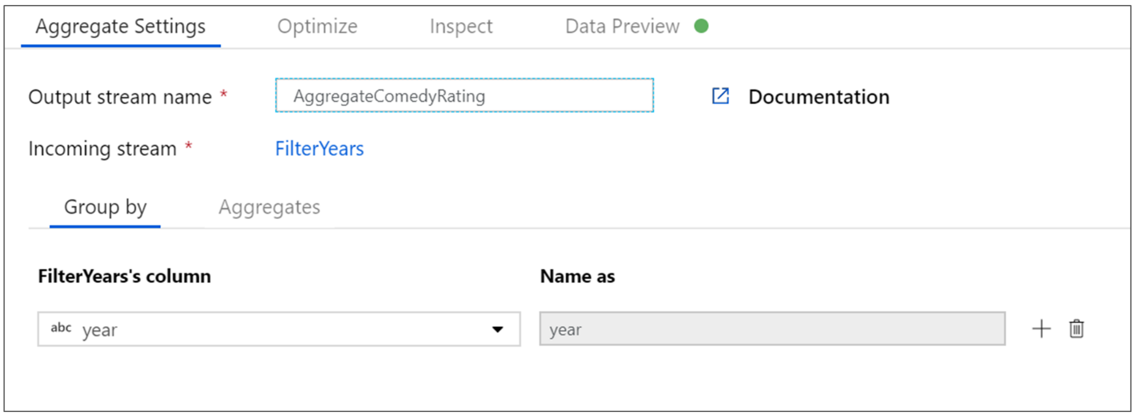
Number of rows INSERT 100 UPDATE 0 DELETE 0 UPSERT 0 LOOKUP 0 TOTAL 2021

	movie abc	title abc	genres abc	year abc	Rating abc	Rotton Tomat
+	8511	Immigrant, The	Comedy	1917	4	59
+	3309	Dog's Life, A	Comedy	1917	3	83
+	72626	Billy Blazes, Esq.	Comedy|Western	1918	2	63
+	3310	Kid, The	Comedy|Drama	1921	8	57
+	83096	Haunted House, The	Comedy	1920	9	61
+	83318	Goat, The	Comedy	1921	8	86
+	83322	Boat, The	Comedy	1920	4	65

13. The next transformation you'll add is an **Aggregate** transformation under **Schema modifier**.



14. Name your aggregate transformation **AggregateComedyRatings**. In the **Group by** tab, select **year** from the dropdown to group the aggregations by the year the movie came out.

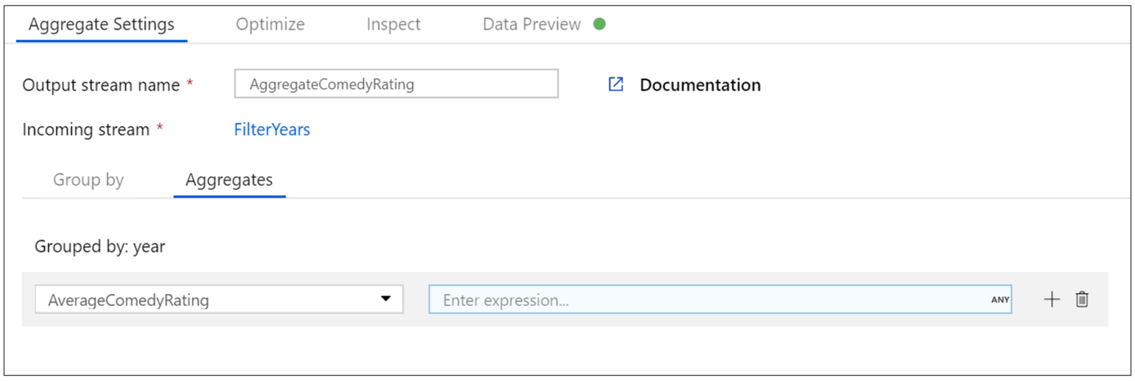


Output stream name * AggregateComedyRating

Incoming stream * FilterYears

FilterYears's column	Name as
abc year	year

15. Go to the **Aggregates** tab. In the left text box, name the aggregate column **AverageComedyRating**. Click on the right expression box to enter the aggregate expression via the expression builder.



Output stream name * AggregateComedyRating

Incoming stream * FilterYears

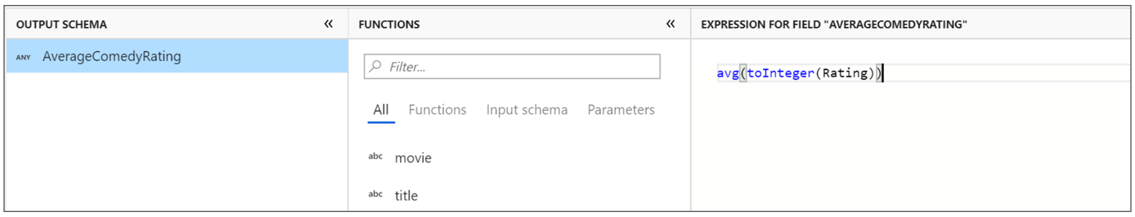
Grouped by: year

AverageComedyRating Enter expression...

16. To get the average of column **Rating**, use the `avg()` aggregate function. As **Rating** is a string and `avg()` takes in a numerical input, we must convert the value to a number via the `toInteger()` function. This is expression looks like:

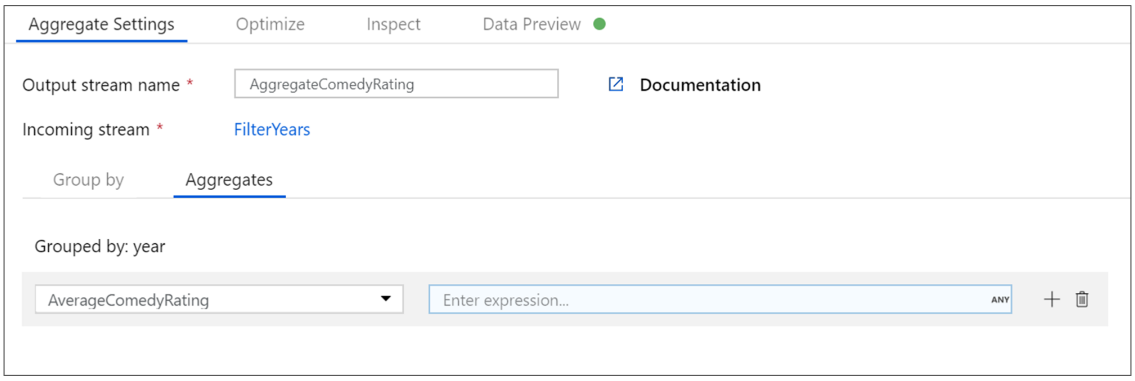
    `avg(toInteger(Rating))`

    Click **Save and Finish** when done.



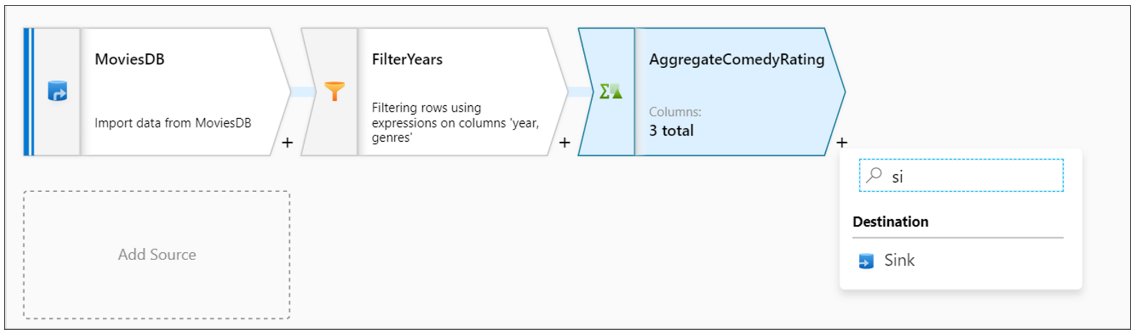
```
avg(toInteger(Rating))
```

17. Go to the **Data Preview** tab to view the transformation output. Notice only two columns are there, **year** and **AverageComedyRating**.

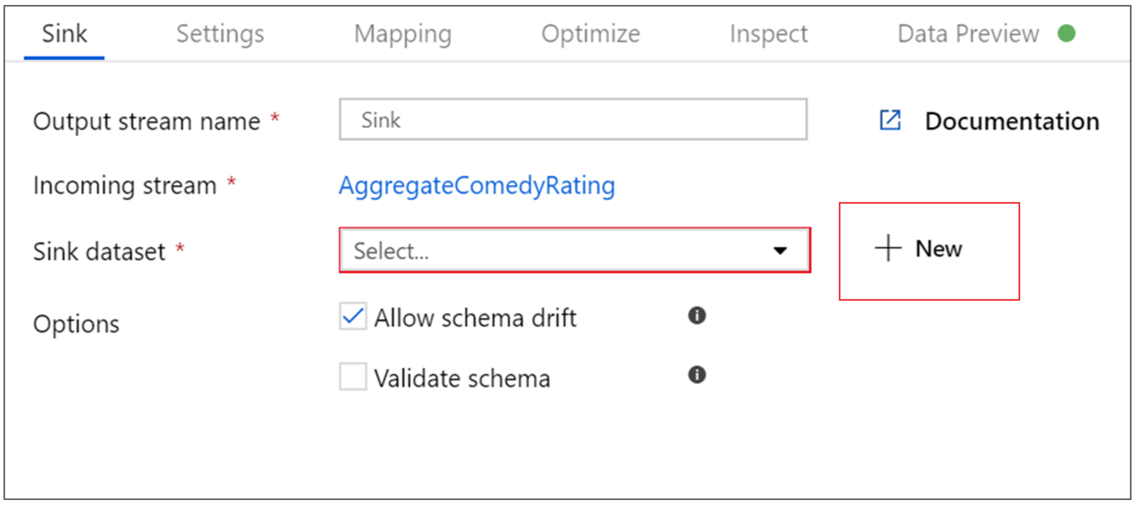


Output stream name * AggregateComedyRating
Incoming stream * FilterYears
Grouped by: year
AverageComedyRating | Enter expression...

18. Next, you want to add a **Sink** transformation under **Destination**.

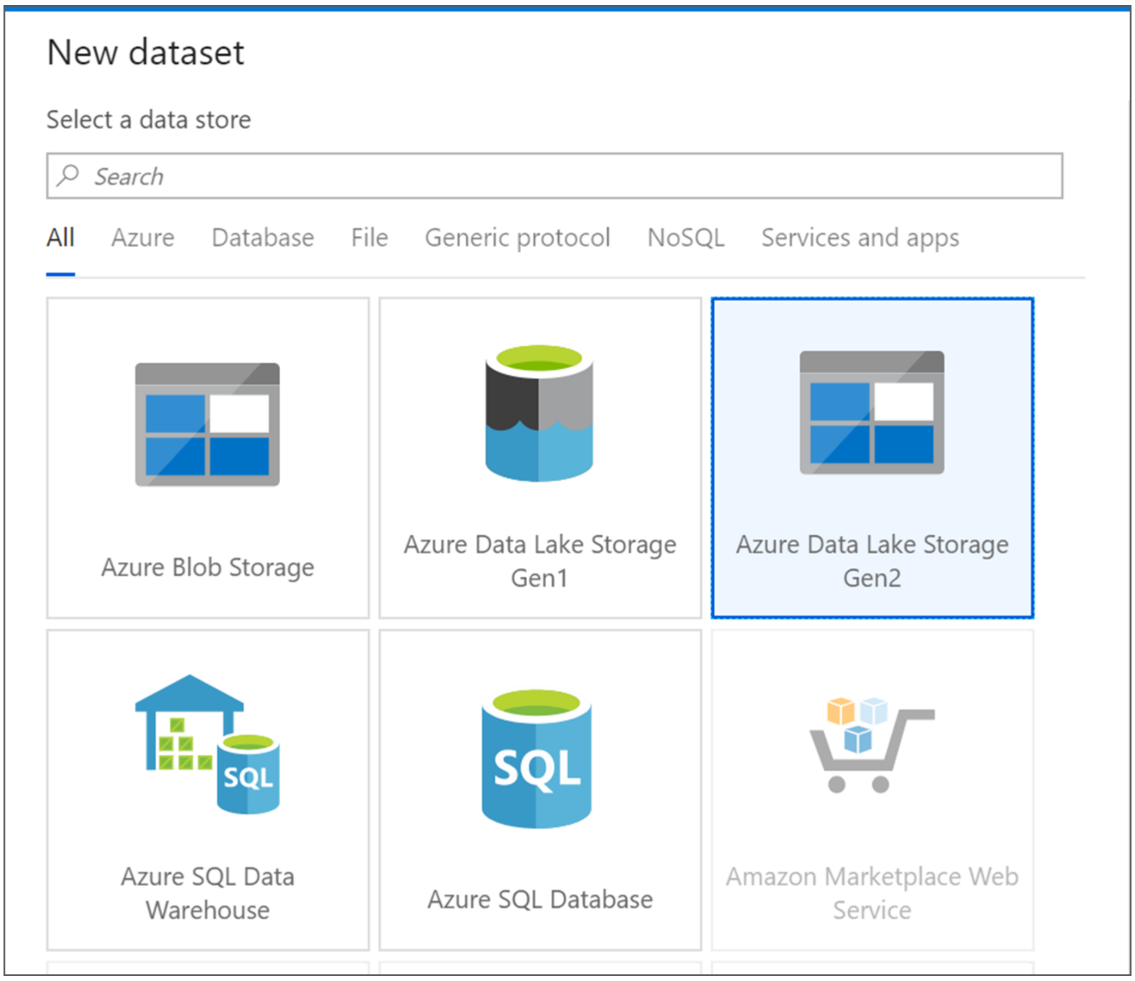

MoviesDB
Import data from MoviesDB
FilterYears
Filtering rows using expressions on columns 'year, genres'
AggregateComedyRating
Columns:
3 total
Add Source
si
Destination
Sink

19. Name your sink **Sink**. Click **New** to create your sink dataset.

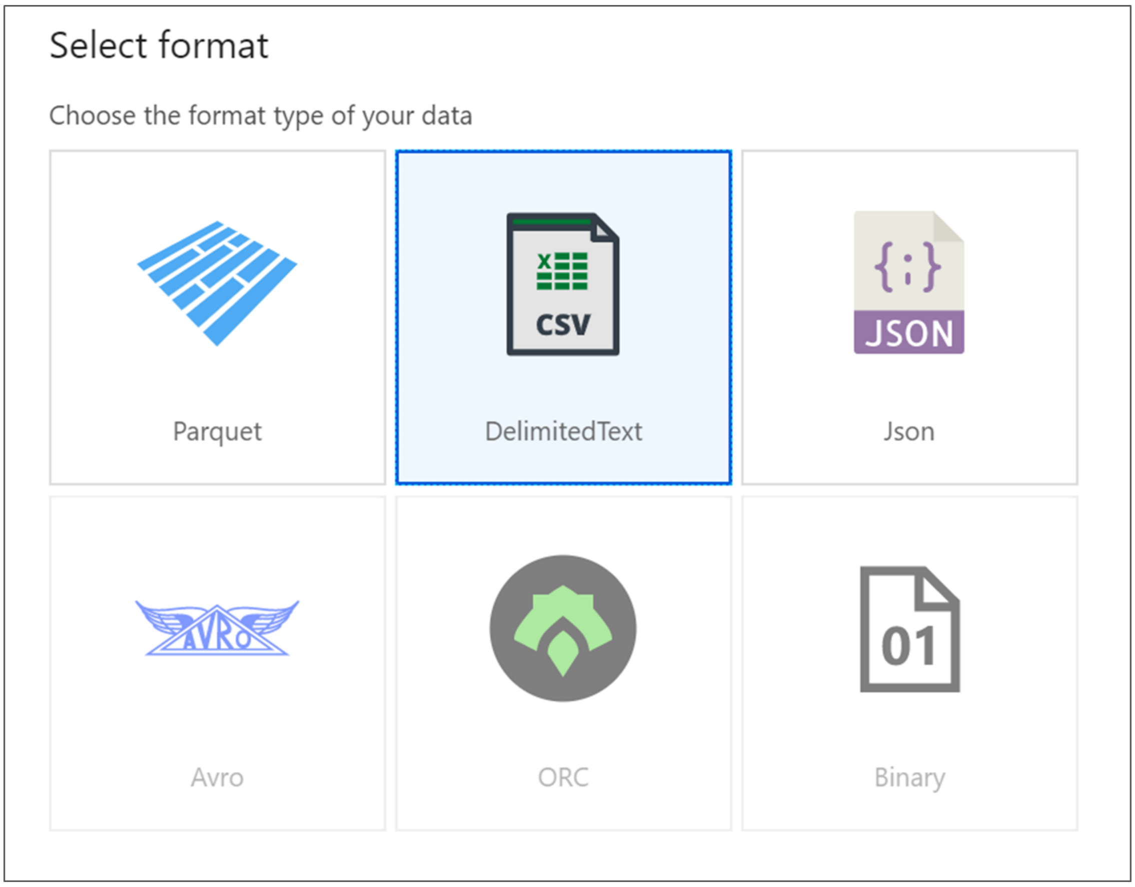


Output stream name * Sink
Incoming stream * AggregateComedyRating
Sink dataset * Select...
Options ✓ Allow schema drift
Validate schema

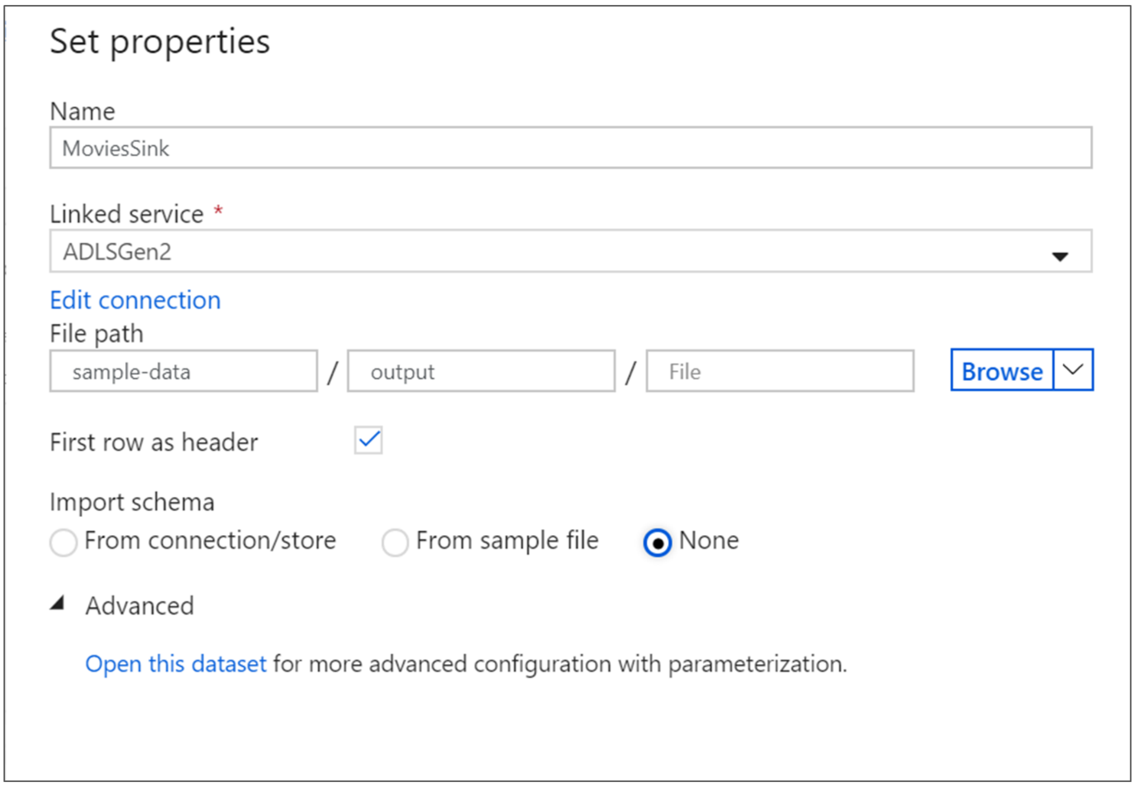
20. Choose **Azure Data Lake Storage Gen2**. Click Continue.



21. Choose **DelimitedText**. Click Continue.



22. Name your sink dataset **MoviesSink**. For linked service, choose the ADLS gen2 linked service you created in step 6. Enter an output folder to write your data to. In this tutorial, we're writing to folder 'output' in container 'sample-data'. The folder doesn't need to exist beforehand and can be dynamically created. Set **First row as header** as true and select **None** for **Import schema**. Click Finish.



Set properties

Name: MoviesSink

Linked service *: ADLSGen2

Edit connection

File path: sample-data / output / File

First row as header: ✓

Import schema: From connection/store From sample file [✓ None]

Advanced

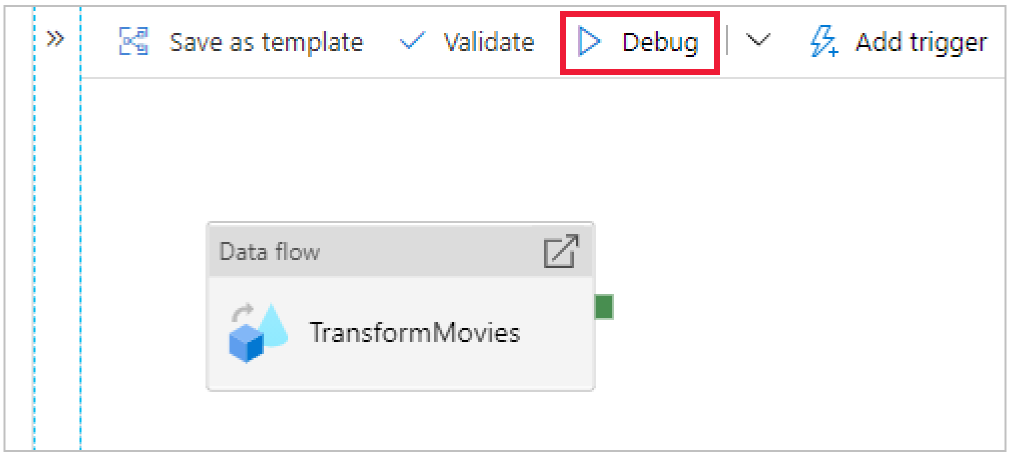
Open this dataset for more advanced configuration with parameterization.

Now you've finished building your data flow. You're ready to run it in your pipeline.

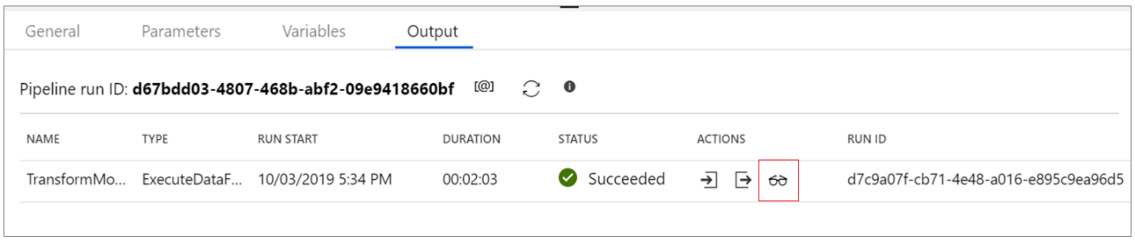
Running and monitoring the Data Flow

You can debug a pipeline before you publish it. In this step, you're going to trigger a debug run of the data flow pipeline. While data preview doesn't write data, a debug run will write data to your sink destination.

1. Go to the pipeline canvas. Click **Debug** to trigger a debug run.



2. Pipeline debug of Data Flow activities uses the active debug cluster but still take at least a minute to initialize. You can track the progress via the **Output** tab. Once the run is successful, click on the eyeglasses icon to open the monitoring pane.

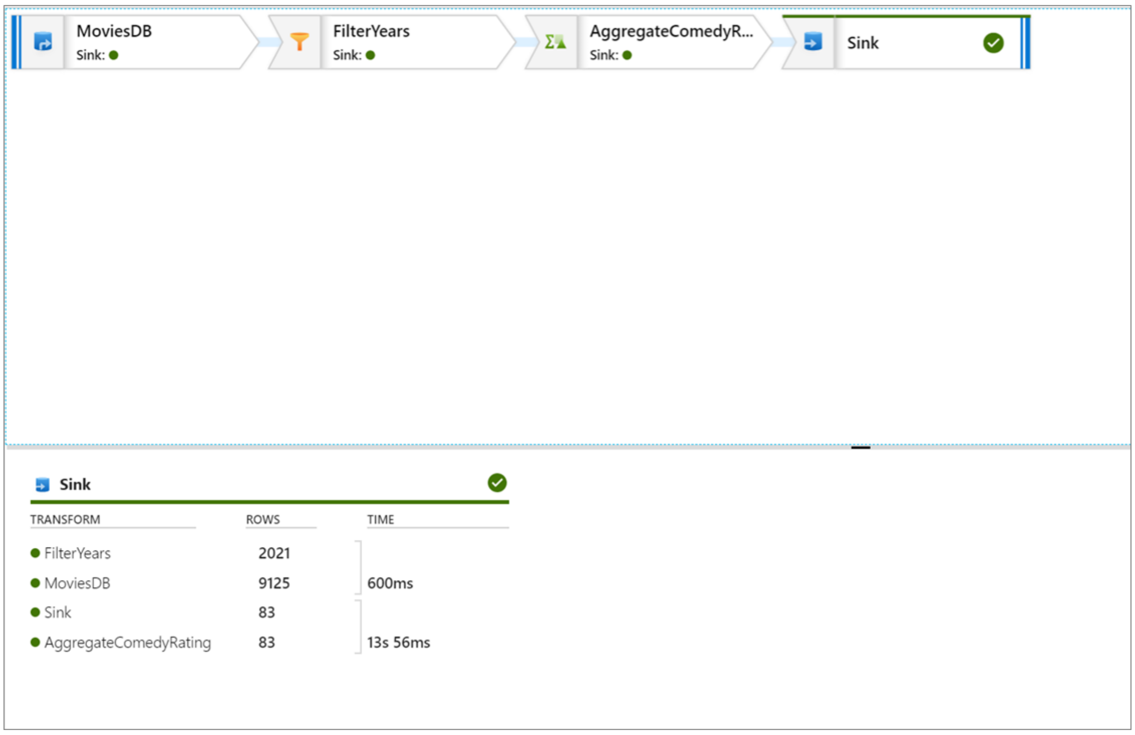


General | Parameters | Variables | Output

Pipeline run ID: **d67bdd03-4807-468b-abf2-09e9418660bf**

NAME	TYPE	RUN START	DURATION	STATUS	ACTIONS	RUN ID
TransformMo...	ExecuteDataF...	10/03/2019 5:34 PM	00:02:03	Succeeded		d7c9a07f-cb71-4e48-a016-e895c9ea96d5

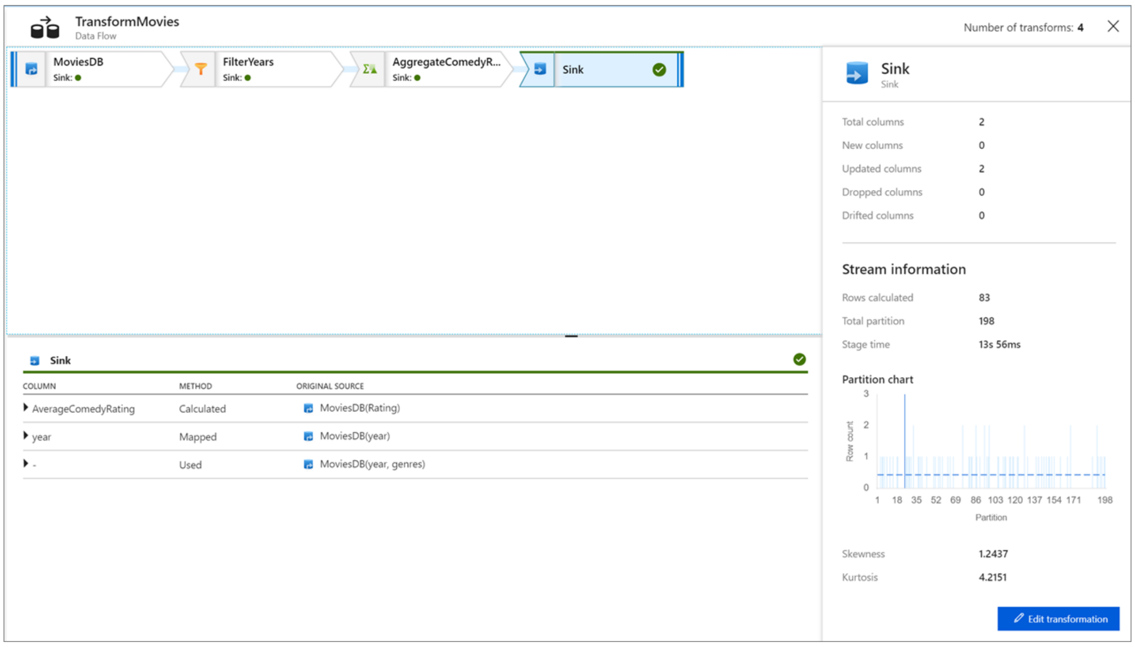
3. In the monitoring pane, you can see the number of rows and time spent in each transformation step.



Sink

TRANSFORM	ROWS	TIME
● FilterYears	2021	
● MoviesDB	9125	600ms
● Sink	83	
● AggregateComedyRating	83	13s 56ms

4. Click on a transformation to get detailed information about the columns and partitioning of the data.



Sink
Sink

Total columns	2
New columns	0
Updated columns	2
Dropped columns	0
Drifted columns	0

Stream information

Rows calculated	83
Total partition	198
Stage time	13s 56ms

Partition chart

Skewness	1.2437
Kurtosis	4.2151

Sink

COLUMN	METHOD	ORIGINAL SOURCE
AverageComedyRating	Calculated	MoviesDB(Rating)
year	Mapped	MoviesDB(year)
-	Used	MoviesDB(year, genres)

If you followed this tutorial correctly, you should have written 83 rows and 2 columns into your sink folder. You can verify the data is correct by checking your blob storage.

Next steps

The pipeline in this tutorial runs a data flow that aggregates the average rating of comedies from 1910 to 2000 and writes the data to ADLS. You learned how to:

- Create a data factory.
- Create a pipeline with a Data Flow activity.
- Build a mapping data flow with four transformations.
- Test run the pipeline.
- Monitor a Data Flow activity