Azure Data Factory

Lab: Orchestrating Data Movement with Azure Data Factory

Pre-requisites:

- Azure Pass subscription
- Azure Data Lake Storage Gen2 storage account.
- Azure SQL Database

Lab objectives:

1. Ingest data using the Copy Activity
2. Use the Mapping Data Flow task to perform transformation

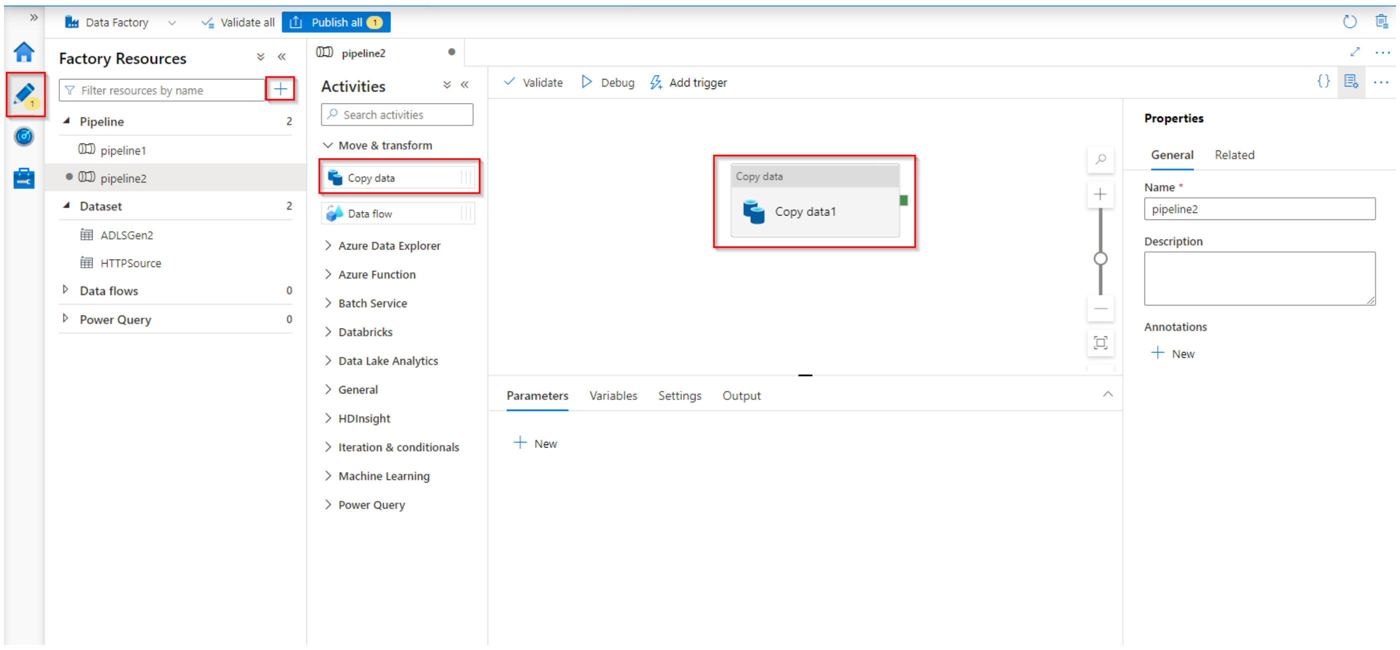
Exercise 1: Use the Mapping Data Flow task to perform transformation

The task for this exercise are as follows

1. Add the Copy Activity to the pipeline
2. Create a new HTTP dataset to use as a source
3. Create a new ADLS Gen2 sink
4. Test the Copy Activity

Task 1: Add the Copy Activity to the designer

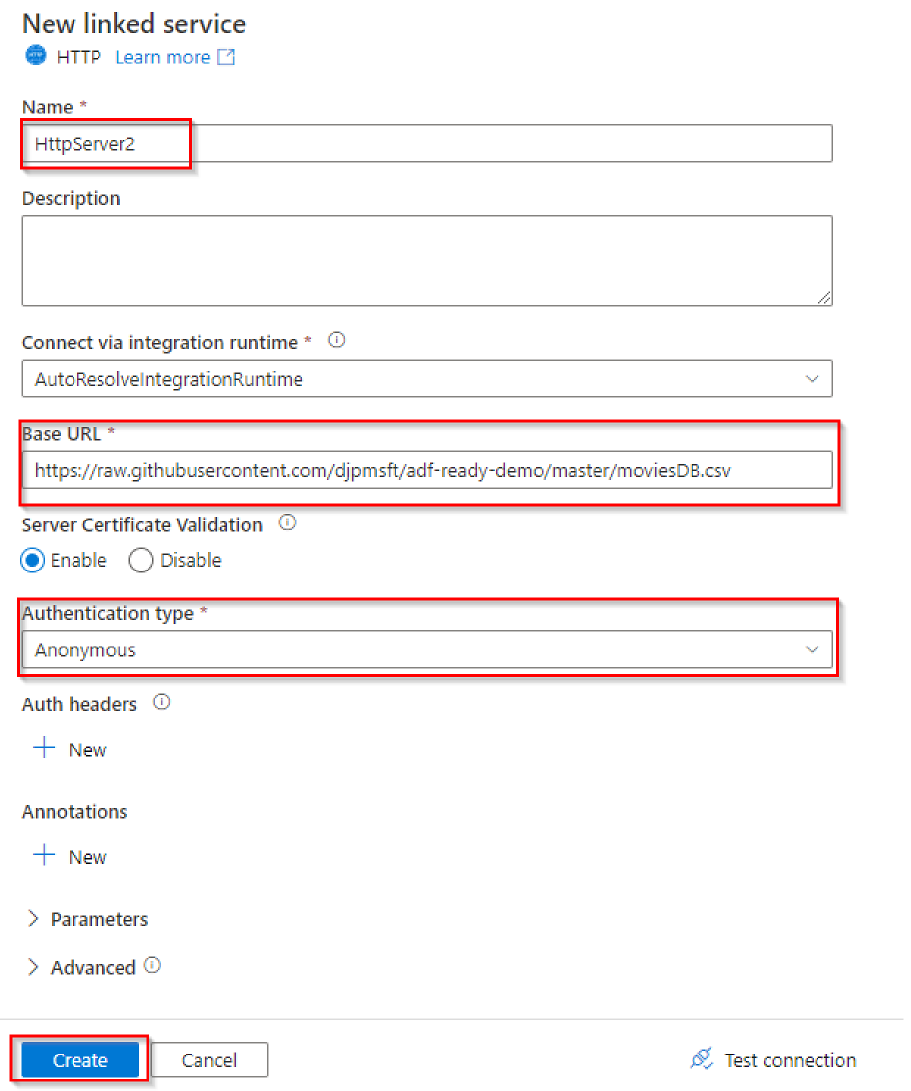
1. Launch Data Factory Studio from Azure Portal.
2. **Open the authoring canvas** If coming from the ADF homepage, click on the **pencil icon** on the left sidebar and select the **+ pipeline button** to open the authoring canvas and create a pipeline.
3. **Add a copy activity** In the Activities pane, open the **Move and Transform** accordion and drag the **Copy data** activity onto the pipeline canvas.



Task 2: Create a new HTTP dataset to use as a source

1. In the Source tab of the Copy activity settings, click **+ New**
2. In the data store list, select the **HTTP** tile and click **continue**
3. In the file format list, select the **DelimitedText** format tile and click **continue**
4. In **Set properties** blade, give your dataset an understandable name such as **HTTPSource** and click on the **Linked Service** dropdown. If you have not created your HTTP Linked Service, select **New**.
5. In the **New Linked Service (HTTP)** screen, copy the URL of the moviesDB csv file below in the **Base URL** textbox. You can access the data with no authentication required using the following endpoint:

   https://raw.githubusercontent.com/djpmsft/adf-ready-demo/master/moviesDB.csv


New linked service

HTTP Learn more

Name *

HttpServer2

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Base URL *

https://raw.githubusercontent.com/djpmsft/adf-ready-demo/master/moviesDB.csv

Server Certificate Validation

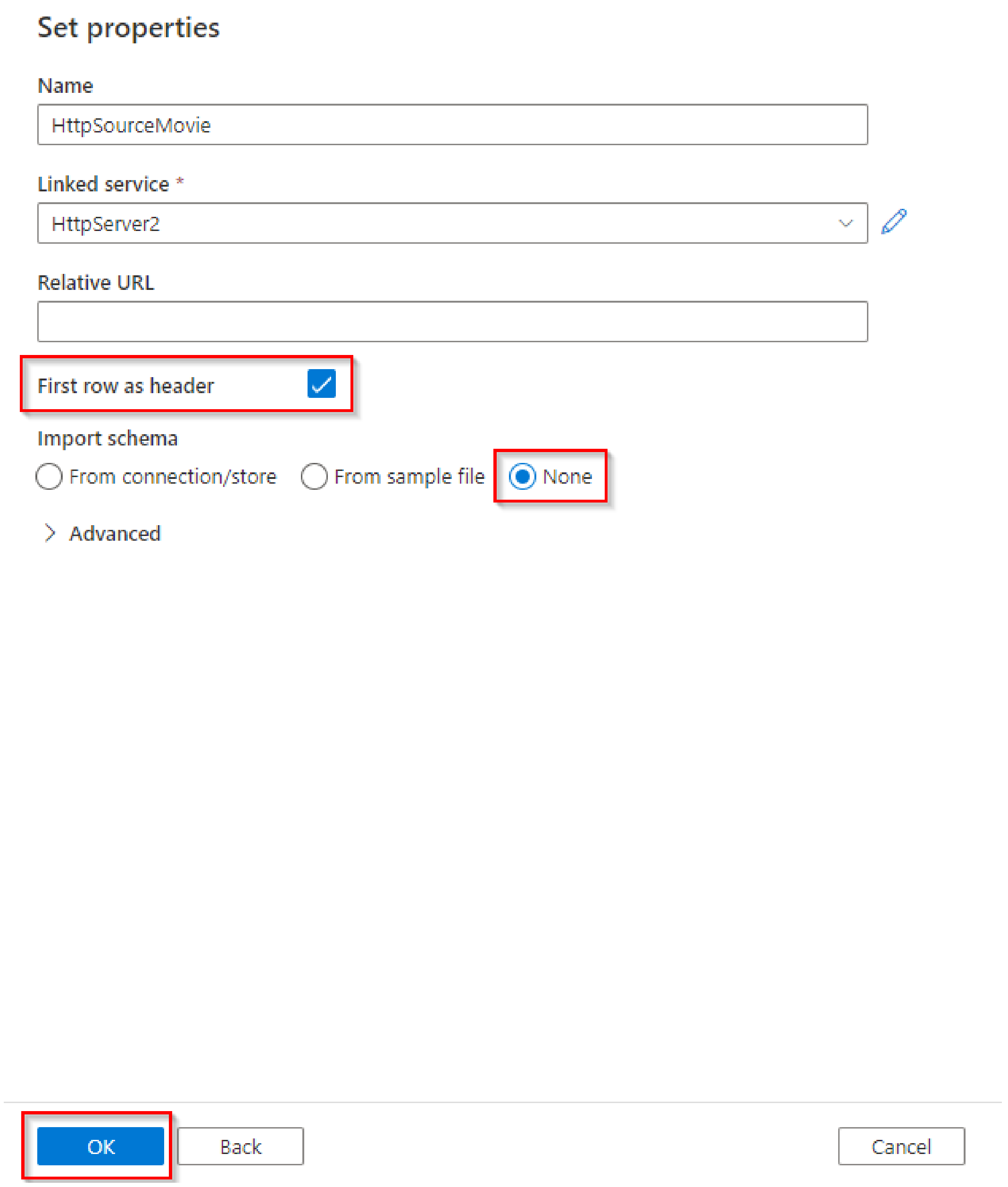
[✓ Enable] Disable

Authentication type *

Anonymous

6. In the **Authentication type** drop down, select **Anonymous**. and click on **Create**.

   - Once you have created and selected the linked service, specify the rest of your dataset settings. These settings specify how and where in your connection we want to pull the data. As the url is pointed at the file already, no relative endpoint is required. As the data has a header in the first row, set **First row as header** to be true and select Import schema from **connection/store** to pull the schema from the file itself. Select **Get** as the request method. You will see the following screen



Set properties

Name: HttpSourceMovie

Linked service *: HttpServer2

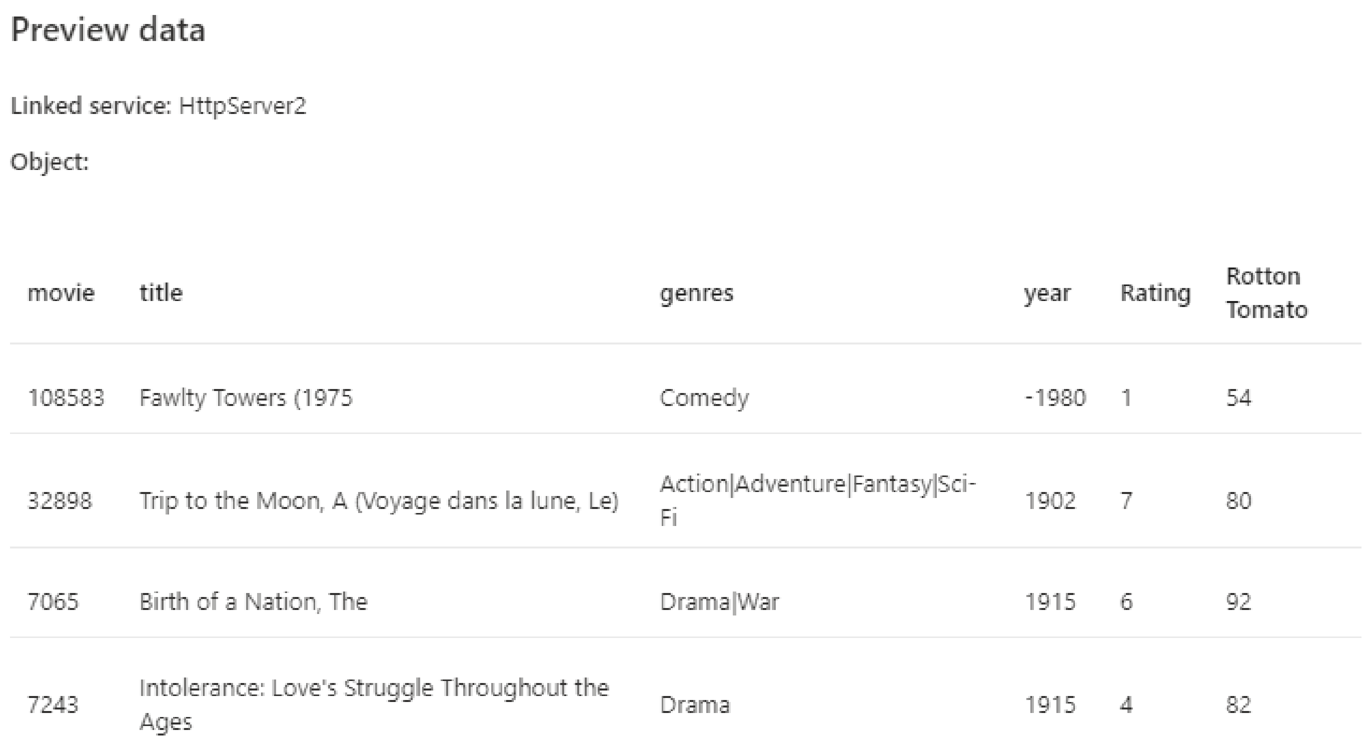
Relative URL:

First row as header: ✓

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

   - Click **OK** once completed.

To verify your dataset is configured correctly, click **Preview data** in the Source tab of the copy activity to get a small snapshot of your data.



Preview data

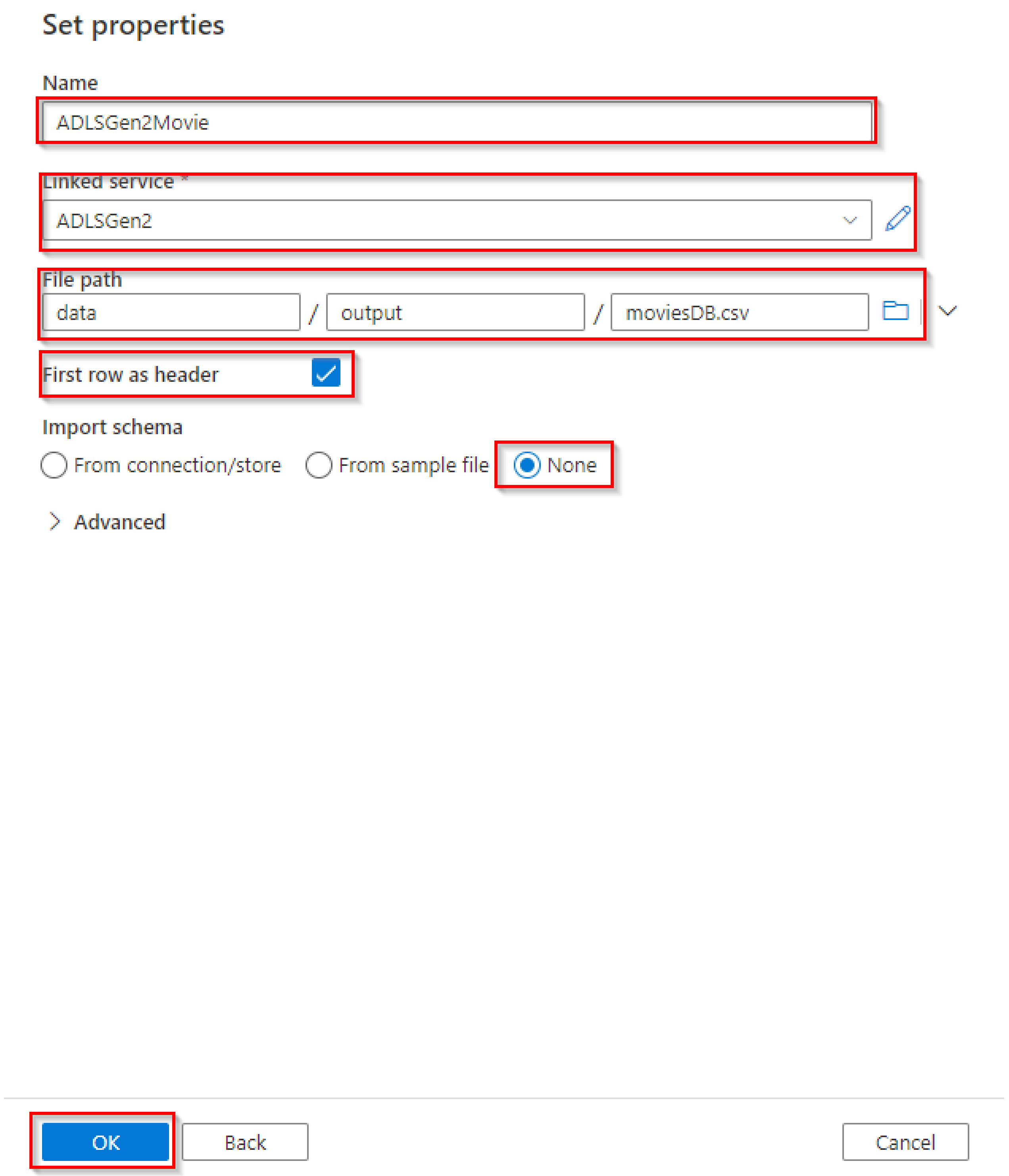
Linked service: HttpServer2

Object:

movie	title	genres	year	Rating	Rotton Tomato
108583	Fawlty Towers (1975	Comedy	-1980	1	54
32898	Trip to the Moon, A (Voyage dans la lune, Le)	Action|Adventure|Fantasy|Sci-Fi	1902	7	80
7065	Birth of a Nation, The	Drama|War	1915	6	92
7243	Intolerance: Love's Struggle Throughout the Ages	Drama	1915	4	82

Task 3: Create a new ADLS Gen2 dataset sink

1. Click on the **Sink tab**, and the click **+ New**
2. Select the **Azure Data Lake Storage Gen2** tile and click **Continue**.
3. Select the **DelimitedText** format tile and click **Continue**.
4. In Set Properties blade, give your dataset an understandable name such as **ADLSGen2Movie** and click on the **Linked Service** dropdown. If you have not created your ADLS Linked Service, select **New**.
5. In the New linked service (Azure Data Lake Storage Gen2) blade, select your authentication method as **Account key**, select your **Azure Subscription** and select your Storage account name of **datalakexx**. You will see a screen as follows:
6. Click on **Create**
7. Once you have configured your linked service, you enter the set properties blade. As you are writing to this dataset, you want to point the folder where you want moviesDB.csv copied to. In the example below, I am writing to folder **output** in the **data** container. Set **First row as header** to be true and Import schema set to **None**.



Set properties

Name: ADLSGen2Movie

Linked service *: ADLSGen2

File path: data / output / moviesDB.csv

First row as header: ✓

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

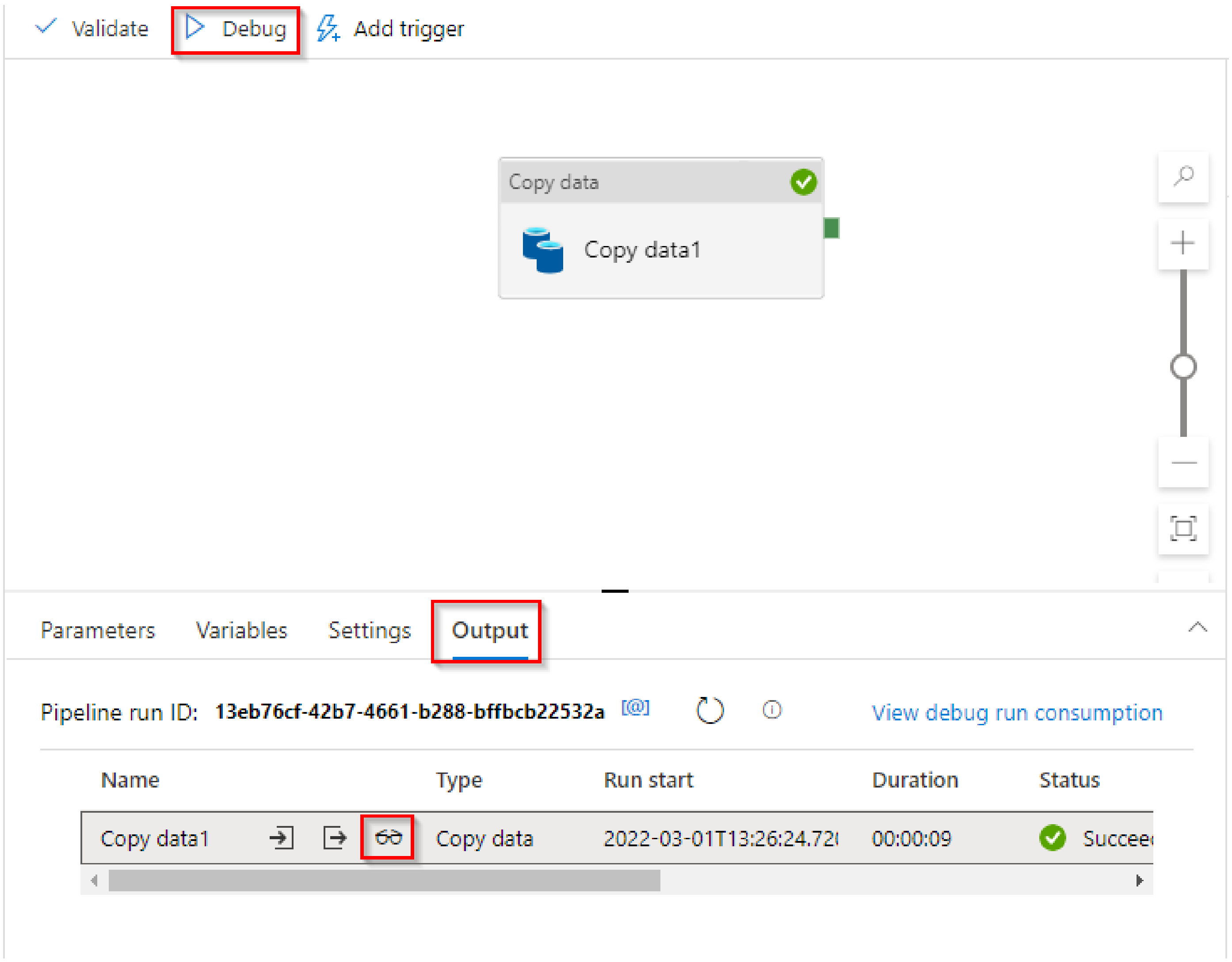
8. Click **OK** once completed.

Task 4: Test the Copy Activity

At this point, you have fully configured your copy activity. To test it out, click on the **Debug** button at the top of the pipeline canvas. This will start a pipeline debug run.

1. To monitor the progress of a pipeline debug run, click on the **Output** tab of the pipeline

2. To view a more detailed description of the activity output, click on the eyeglasses icon. This will open up the copy monitoring screen which provides useful metrics such as Data read/written, throughput and in-depth duration statistics.



Pipeline run ID: 13eb76cf-42b7-4661-b288-bffbcb22532a

Name	Type	Run start	Duration	Status
Copy data1	Copy data	2022-03-01T13:26:24.72	00:00:09	Succee

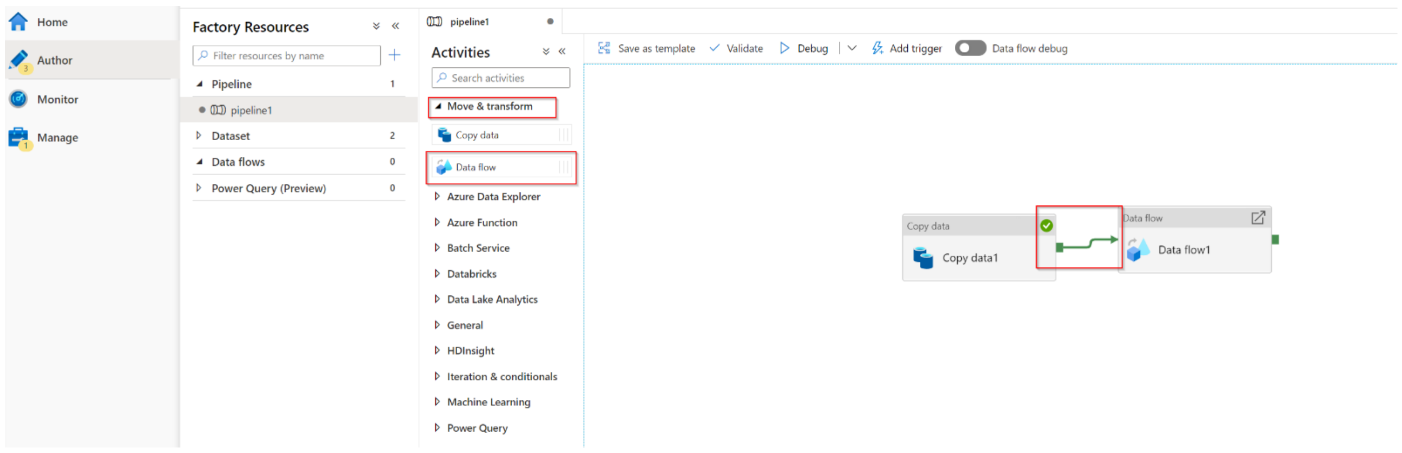
3. To verify the copy worked as expected, open up your ADLS gen2 storage account and check to see your file was written as expected

Exercise 2: Transforming Data with Mapping Data Flow

Tasks for this exercise are as follows:

1. Preparing the environment
2. Adding a Data Source
3. Using Mapping Data Flow transformation
4. Writing to a Data Sink
5. Running the Pipeline

Task 1: Preparing the environment

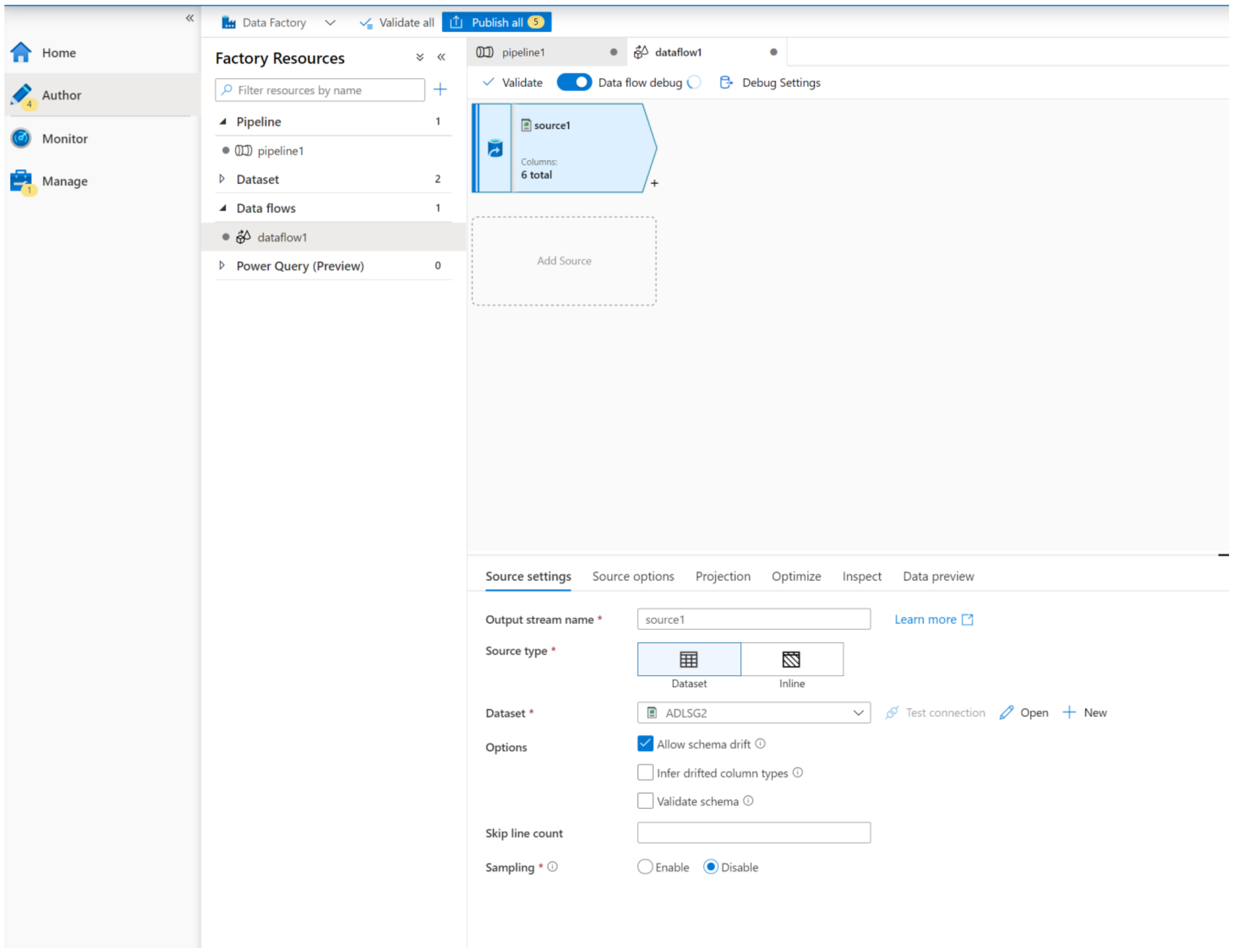
1. **Add a Data Flow activity** In the Activities pane, open the **Move and Transform** accordion and drag the **Data Flow** activity onto the pipeline canvas and connect with the **Copy data** activity we created earlier.
2. 

3. Turn the **Data Flow Debug** slider located at the top of the authoring module on, and click **OK** in the **Turn on data flow debug** screen that appears.

   NOTE: Data Flow clusters take 5-7 minutes to warm up.
4. Select the data flow activity in the pipeline workspace. In the lower pane, select the settings tab, click **+ New** for the variable **Dataflow**

Task 2: Adding a Data Source

1. **Add an ADLS source**: Click on the Mapping Data Flow object in the canvas. Go to the source settings tab. In the **Dataset** dropdown, select your **ADLSGen2** dataset used in your Copy activity



Output stream name * source1

Source type * [✓ Dataset] Inline

Dataset * ADLSG2

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

Skip line count

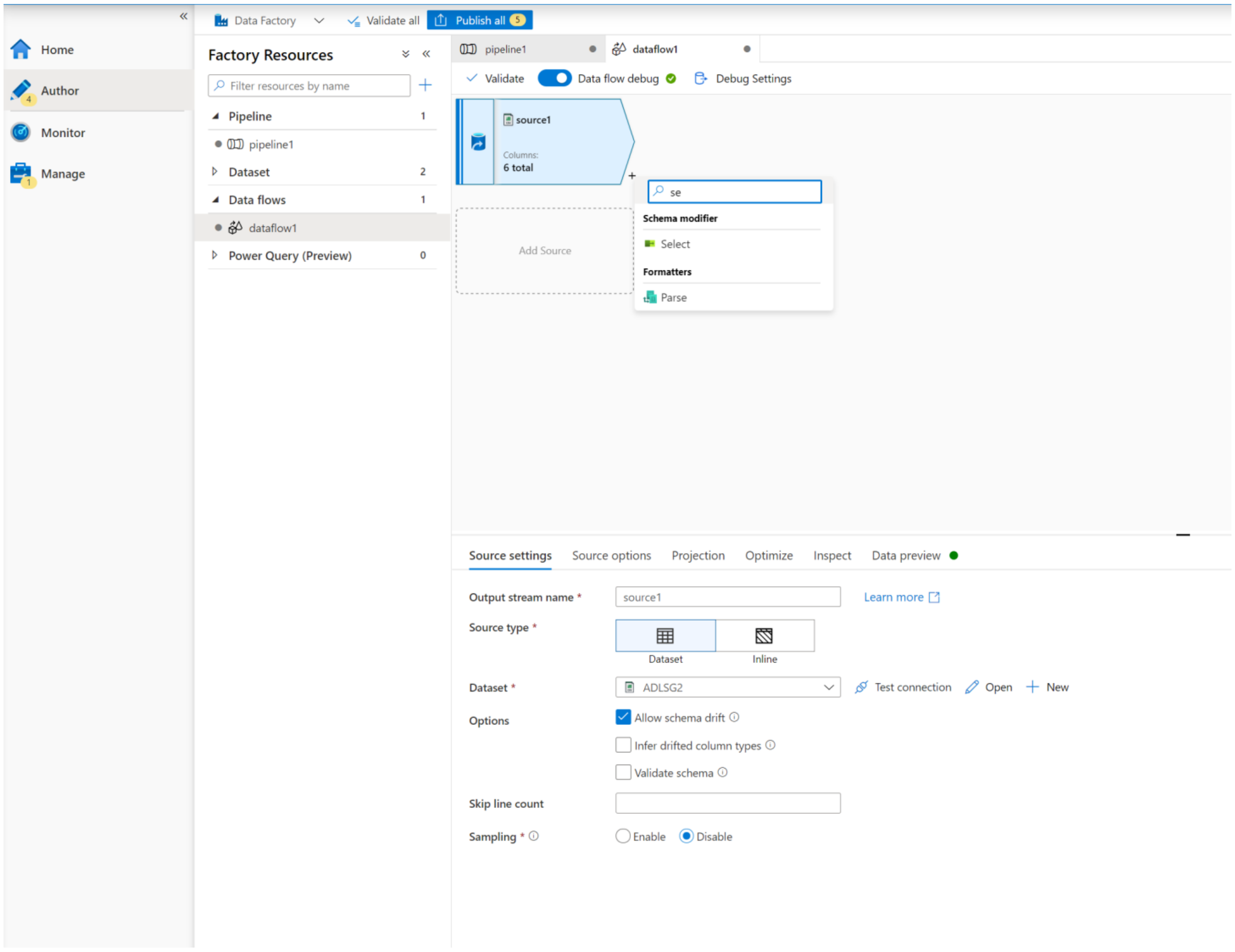
Sampling * Enable [✓ Disable]

   - If your dataset is pointing at a folder with other files, you may need to create another dataset or utilize parameterization to make sure only the moviesDB.csv file is read
   - If you have not imported your schema in your ADLS, but have already ingested your data, go to the dataset's 'Schema' tab and click 'Import schema' so that your data flow knows the schema projection.

Once your debug cluster is warmed up, verify your data is loaded correctly via the **Data preview** tab. Once you click the refresh button, Mapping Data Flow will show calculate a snapshot of what your data looks like when it is at each transformation.

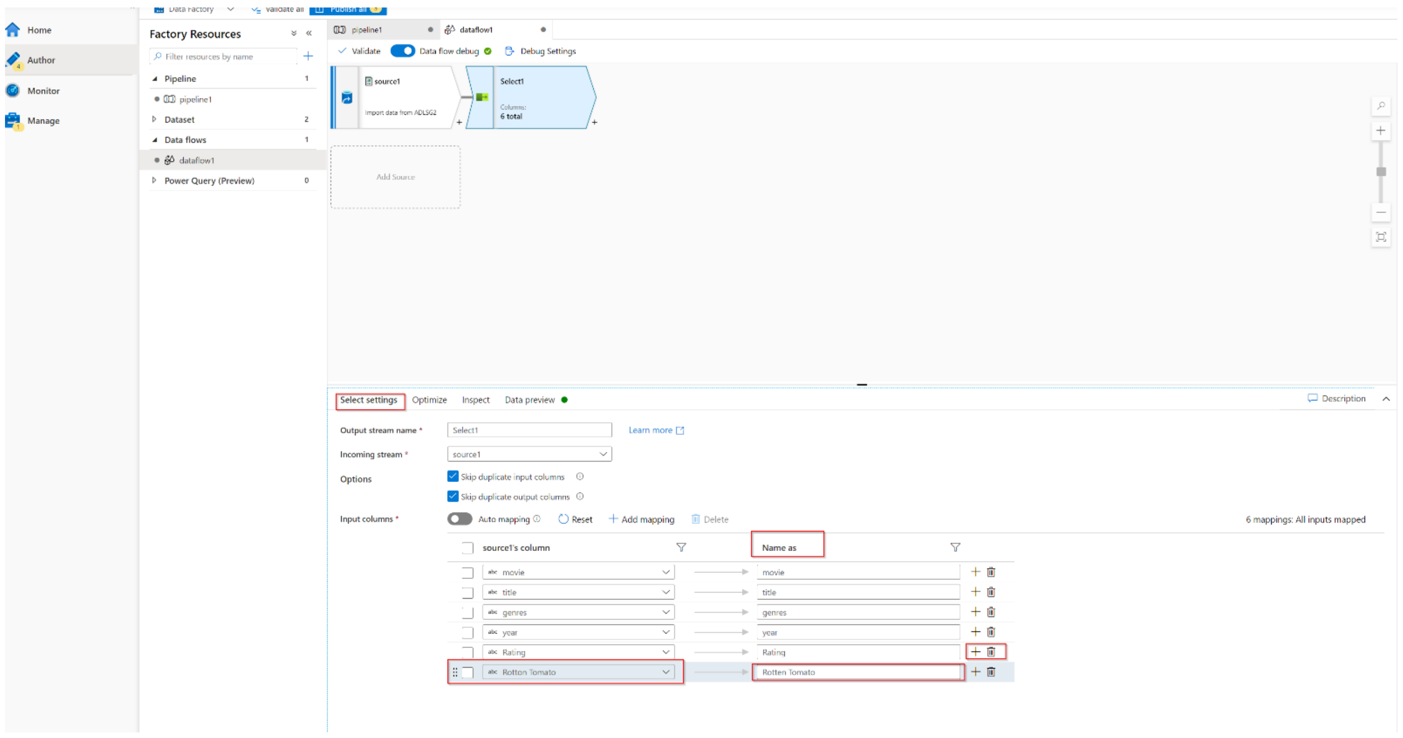
Task 3: Using Mapping Data Flow transformation

1. **Add a Select transformation to rename and drop a column**: In the preview of the data, you may have noticed that the "Rotton Tomatoes" column is misspelled. To correctly name it and drop the unused Rating column, you can add a Select transformation by clicking on the + icon next to your ADLS source node and choosing **Select** under Schema modifier.



Output stream name * source1 Learn more
Source type * [✓ Dataset] Inline
Dataset * ADLSG2 Test connection Open New
Options [✓ Allow schema drift] Infer drifted column types Validate schema
Skip line count
Sampling * Enable [✓ Disable]

In the **Name as** field, under the **Select settings** tab, change 'Rotton' to 'Rotten'. To drop the Rating column, hover over it and click on the trash can icon.



Output stream name * Select1 Learn more

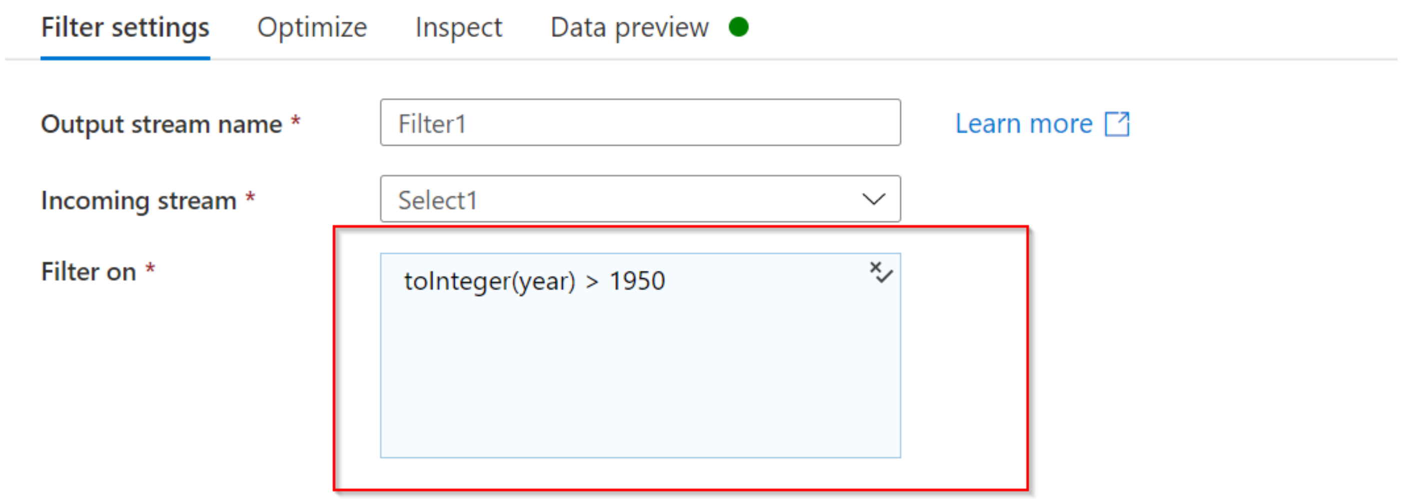
Incoming stream * source1

Options
- [x] Skip duplicate input columns
- [x] Skip duplicate output columns

Input columns * Auto mapping Reset + Add mapping Delete 6 mappings: All inputs mapped

source1's column	Name as
movie	movie
title	title
genres	genres
year	year
Rating	Rating
Rotton Tomato	Rotten Tomato

2. **Add a Filter Transformation to filter out unwanted years**: Say you are only interested in movies made after 1951. You can add a Filter transformation to specify a filter condition by clicking on the **+ icon** next to your Select transformation and choosing **Filter** under Row Modifier. Click on the **expression box** to open up the Expression builder and enter in your filter condition. Using the syntax of the Mapping Data Flow expression language, **toInteger(year) > 1950** will convert the string year value to an integer and filter rows if that value is above 1950.



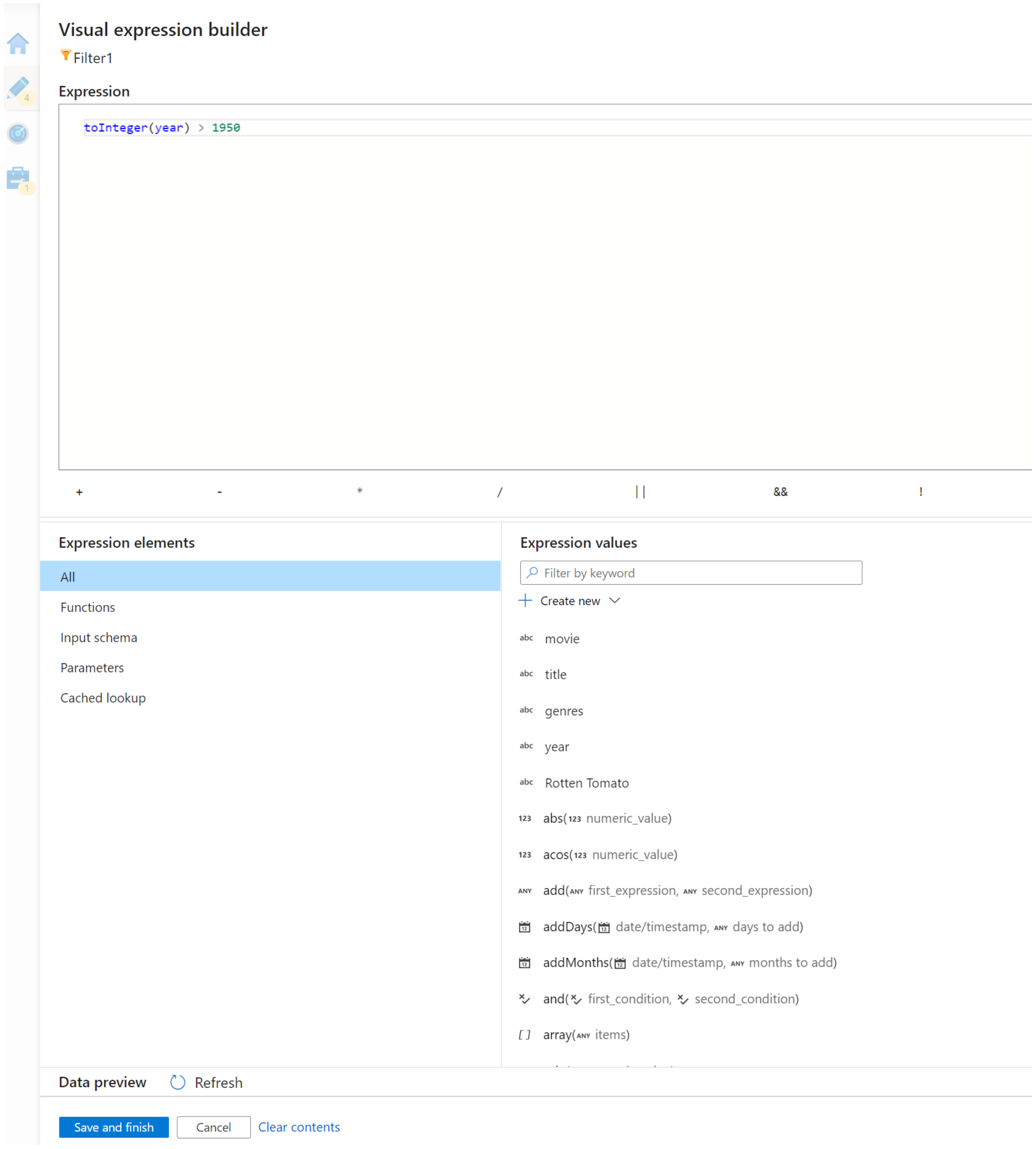
Output stream name * Filter1 Learn more

Incoming stream * Select1

Filter on *

```
toInteger(year) > 1950
```

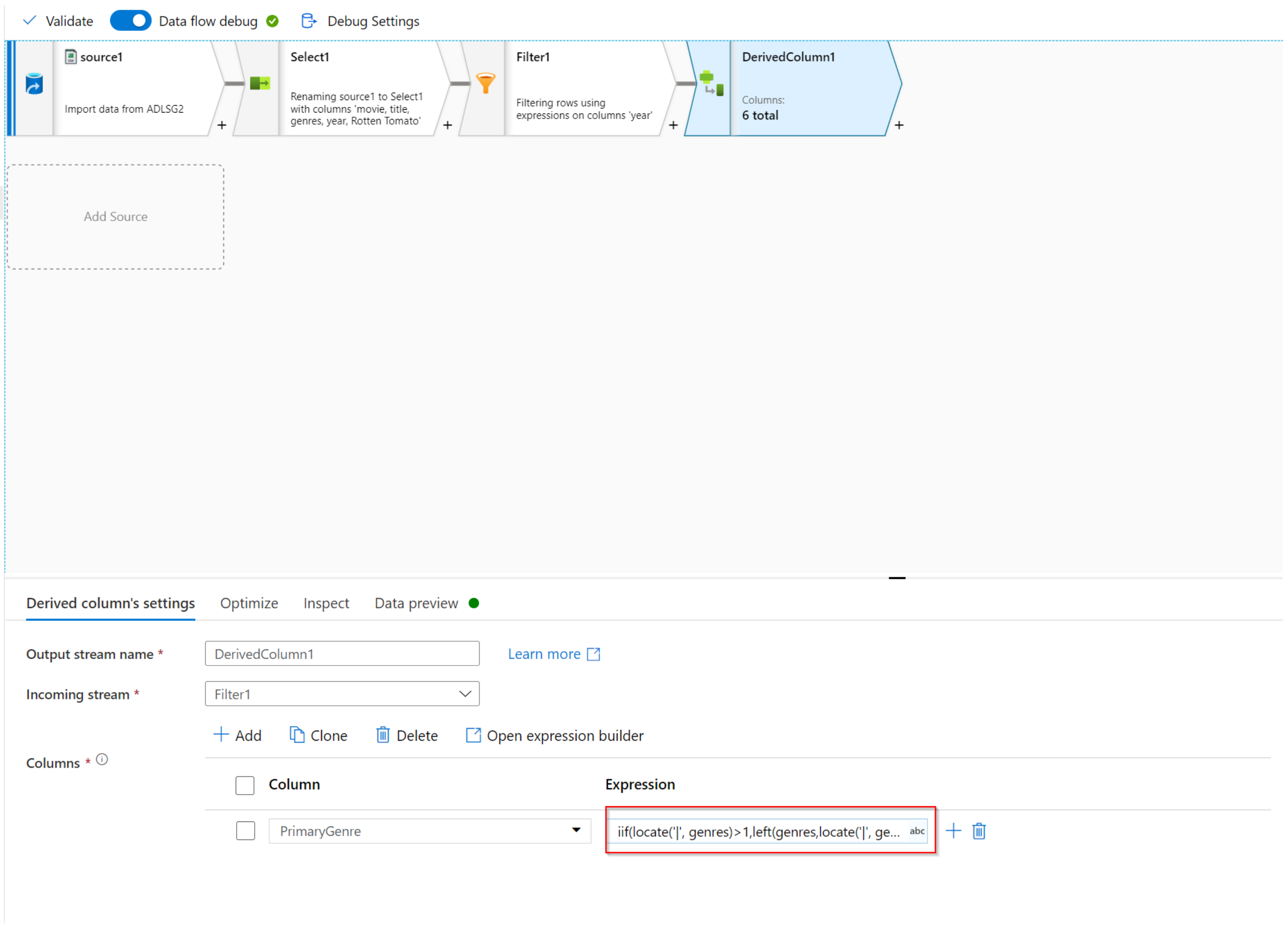
When you clicked on **open expression builder** you can verify your condition is working properly. This will also show by a check mark in the **Filter on** textbox.



```
toInteger(year) > 1950
```

3. **Add a Derive Transformation to calculate primary genre**: As you may have noticed, the genres column is a string delimited by a '|' character. If you only care about

the *first* genre in each column, you can derive a new column named **PrimaryGenre** via the Derived Column transformation by clicking on the **+ icon** next to your Filter transformation and choosing **Derived Column** under Schema Modifier. Similar to the filter transformation, the derived column uses the Mapping Data Flow expression builder to specify the values of the new column.



Output stream name * DerivedColumn1

Incoming stream * Filter1

Columns *

Column	Expression
PrimaryGenre	iif(locate('|', genres)>1,left(genres,locate('|', ge...

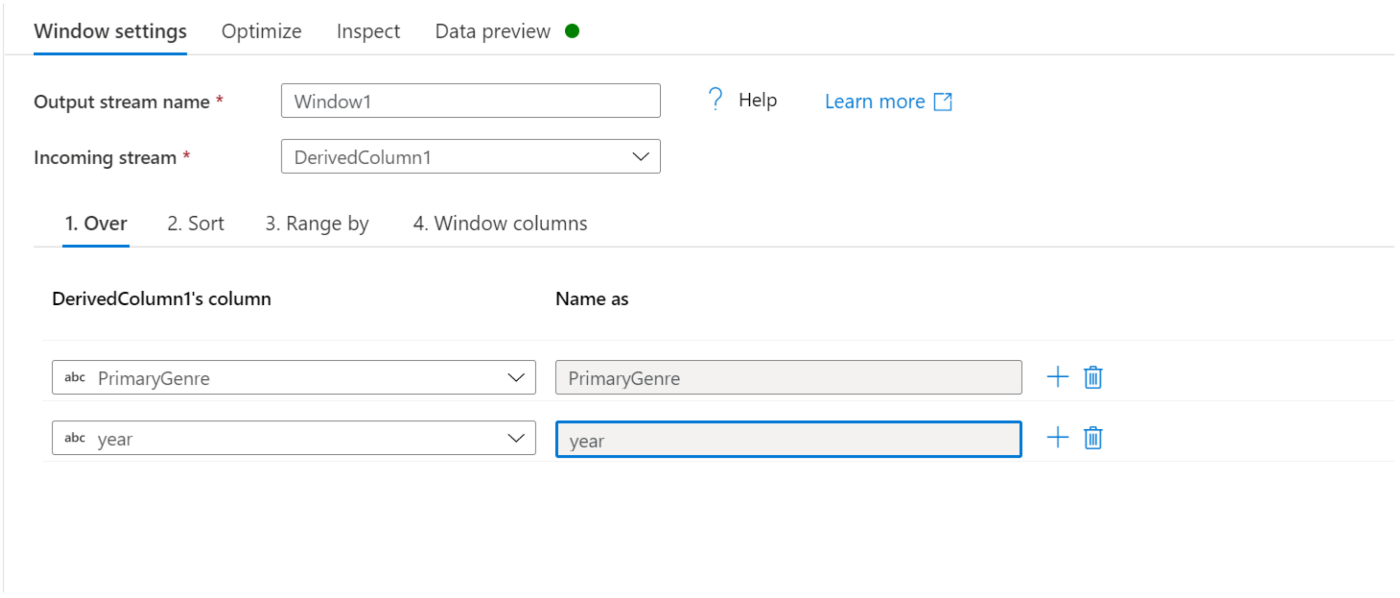
In this scenario, you are trying to extract the first genre from the genres column which is formatted as 'genre1|genre2|...|genreN'. Use the **locate** function to get the first 1-based index of the '|' in the genres string. Using the **iif** function, if this index is greater than 1, the primary genre can be calculated via the **left** function which returns all characters in a string to the left of an index. Otherwise, the PrimaryGenre value is equal to the genres field. You can verify the output via the expression builder's Data preview pane.

- In the **Derived column's** settings tab, click **+Add** then, **Add column**, to add a column named **PrimaryGenre**.
- Under **Expression** open the **Expression builder**.

- Write iif(locate('|', genres)>1,left(genres,locate('|', genres)-1),genres)
- Select **Save and finish**

4. **Rank movies via a Window Transformation** Say you are interested in how a movie ranks within its year for its specific genre. You can add a Window transformation to define window-based aggregations by clicking on the **+ icon** next to your Derived Column transformation and clicking **Window** under Schema modifier. To accomplish this, specify what you are windowing over, what you are sorting by, what the range is, and how to calculate your new window columns. In this example, we will window over PrimaryGenre and year with an unbounded range, sort by Rotten Tomato descending, a calculate a new column called RatingsRank which is equal to the rank each movie has within its specific genre-year.

   - In the **Window settings** pane under the **Over** tab, select **PrimaryGenre** and add **year** by clicking on **+** and selecting **year** from the dropdown.

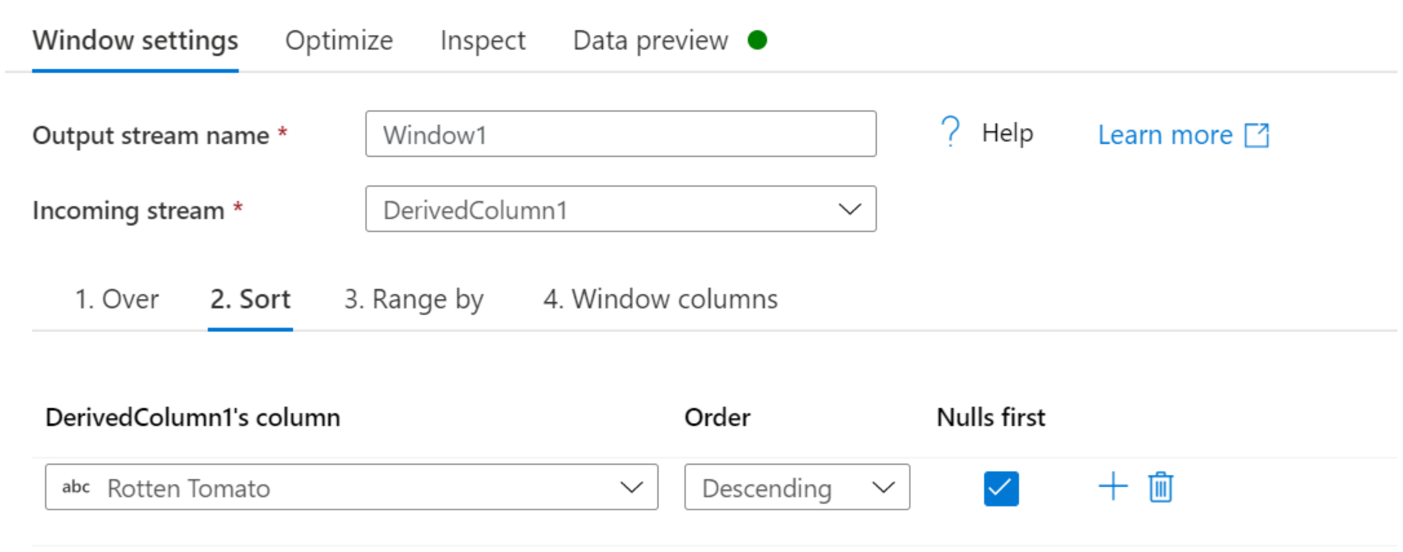


Output stream name * Window1

Incoming stream * DerivedColumn1

DerivedColumn1's column	Name as
abc PrimaryGenre	PrimaryGenre
abc year	year

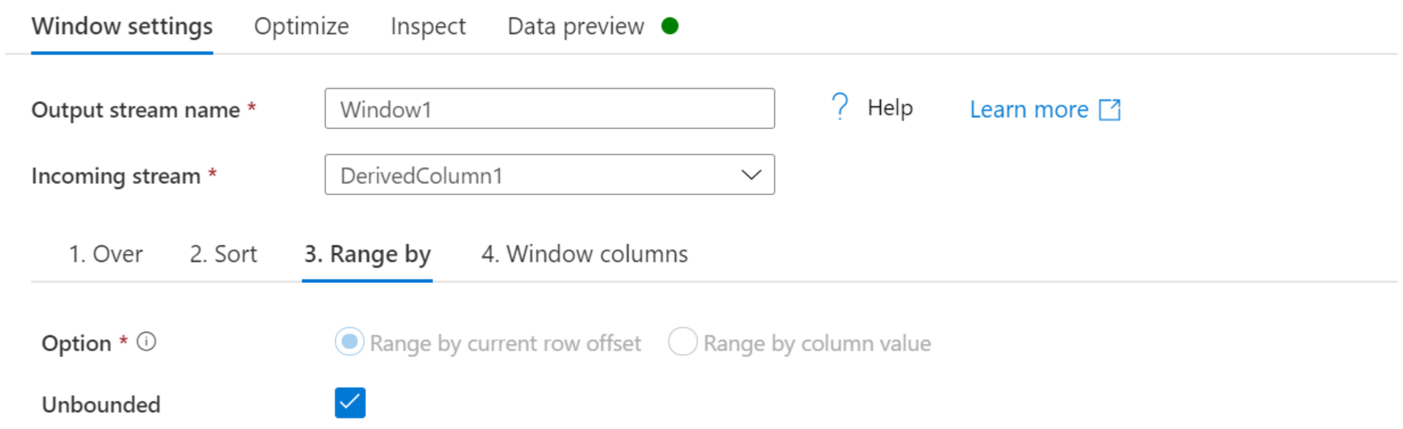
   - In the **Sort settings** pane, select the **Rotten Tomato** column, select **Descending** under **Order** and check **Nulls first**



Output stream name *	Window1
Incoming stream *	DerivedColumn1

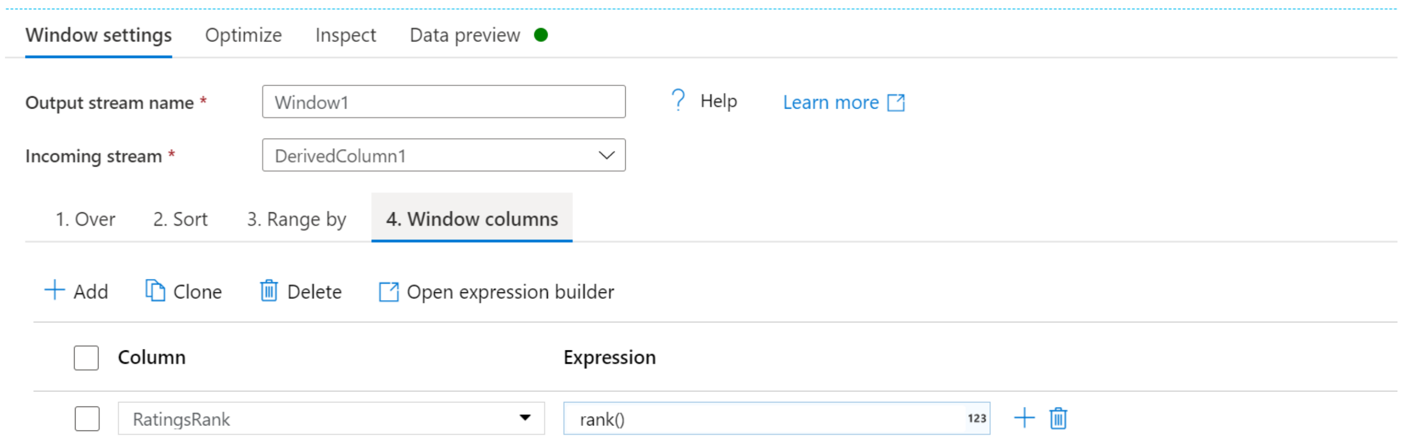
DerivedColumn1's column	Order	Nulls first
abc Rotten Tomato	Descending	✓

- In the **Range by settings** pane, leave all settings per default.



Output stream name *	Window1
Incoming stream *	DerivedColumn1
Option *	[✓ Range by current row offset] Range by column value
Unbounded	✓

- In the **Window columns settings** pane, rename the blank column to **RatingsRank** and enter as expression **rank()**

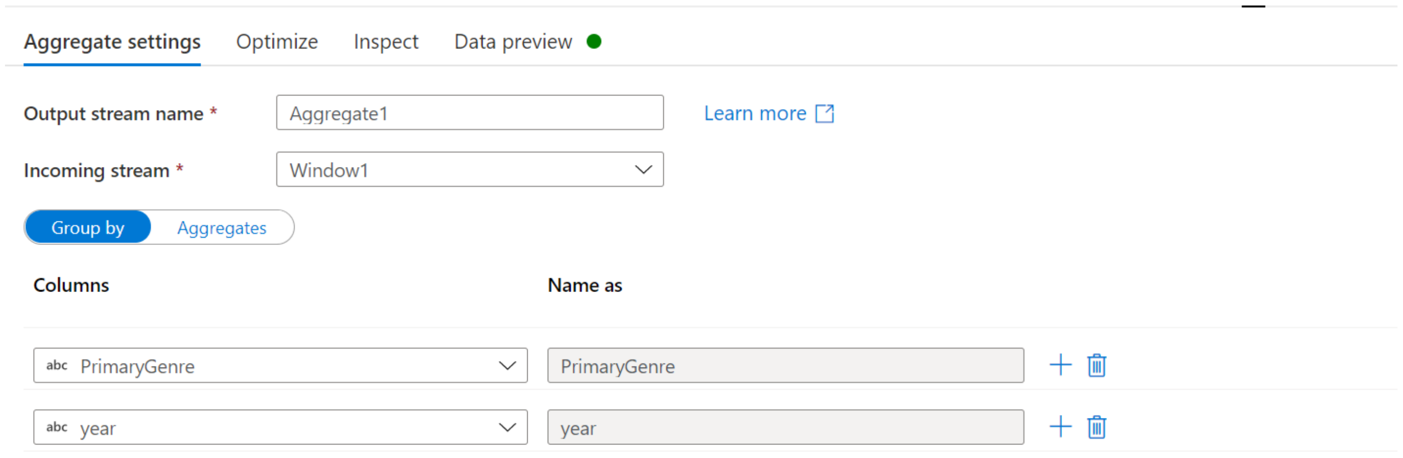

Window settings | Optimize | Inspect | Data preview ●

Output stream name * Window1
Incoming stream * DerivedColumn1

1. Over | 2. Sort | 3. Range by | 4. Window columns

Column	Expression
RatingsRank	rank()

5. **Aggregate ratings with an Aggregate Transformation**: Now that you have gathered and derived all your required data, we can add an Aggregate transformation to calculate metrics based on a desired group by clicking on the **+ icon** next to your Window transformation and clicking **Aggregate** under Schema modifier. As you did in the window transformation, lets group movies by PrimaryGenre and year
   - Under the **Aggregate settings** tab, select **Group by**.
   - Using the dropdown select the column **Primary Genre** and add the **year** column by clicking **+**, and dropdown.


Aggregate settings | Optimize | Inspect | Data preview ●

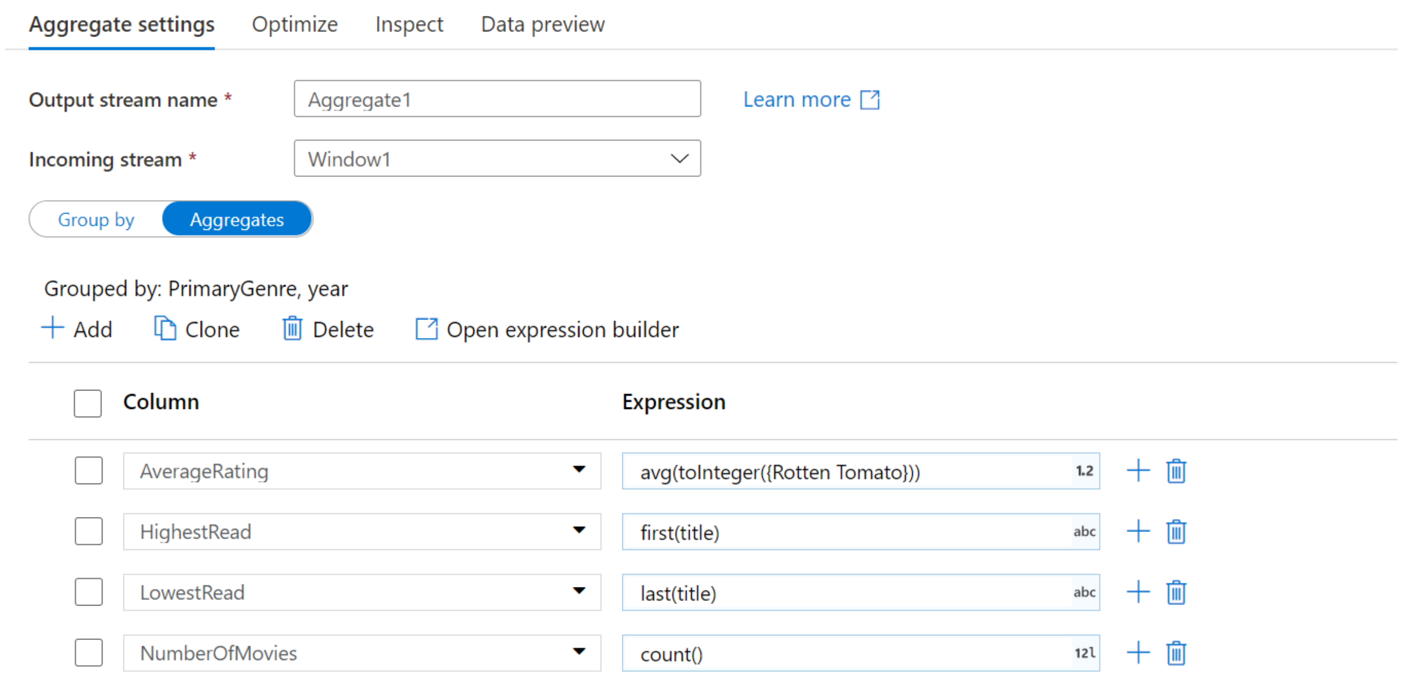
Output stream name * Aggregate1
Incoming stream * Window1

Group by | Aggregates

Columns	Name as
abc PrimaryGenre	PrimaryGenre
abc year	year

In the Aggregates tab, you can aggregations calculated over the specified group by columns. For every genre and year, lets get the average Rotten Tomatoes rating, the highest and lowest rated movie (utilizing the windowing function) and the number of movies that are in each group. Aggregation significantly reduces the amount of rows in your transformation stream and only propagates the group by and aggregate columns specified in the transformation.

- Under the **Aggregate settings** tab, now select **Aggregates**. Add the following columns by clicking **+** and then **Add column**, with their respective expressions:
  - AverageRating: avg(toInteger({Rotten Tomato}))
  - HighestRead: first(title)
  - LowestRead: last(title)
  - NumberOfMovies: count()



Output stream name * Aggregate1

Incoming stream * Window1

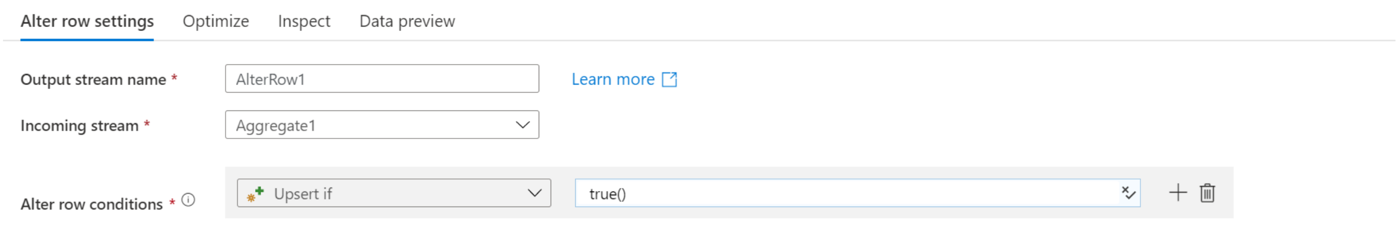
Grouped by: PrimaryGenre, year

Column	Expression
AverageRating	avg(toInteger({Rotten Tomato}))
HighestRead	first(title)
LowestRead	last(title)
NumberOfMovies	count()

- To see how the aggregate transformation changes your data, use the Data Preview tab

6. **Specify Upsert condition via an Alter Row Transformation** If you are writing to a tabular sink, you can specify insert, delete, update and upsert policies on rows using the Alter Row transformation by clicking on the **+ icon** next to your Aggregate transformation and clicking **Alter Row** under Row modifier. Since you are always inserting and updating, you can specify that all rows will always be upserted.

   - From the dropdown next to **Alter row conditions** in the **Alter row settings** tab, please select **Upsert if**. In the expression write **true()**


Alter row settings | Optimize | Inspect | Data preview

Output stream name * AlterRow1 Learn more

Incoming stream * Aggregate1

Alter row conditions * Upsert if true()

Task 4: Writing to a Data Sink

1. **Write to a Azure SQL Database Sink**: Now that you have finished all your transformation logic, you are ready to write to a Sink.

    i. Add a **Sink** by clicking on the **+ icon** next to your Alter row transformation and clicking **Sink** under Destination.

    ii. In the Sink tab, create a new data warehouse dataset via the **+ New button** next to **Dataset**.

    iii. Select **Azure SQL Database** from the tile list and click **Continue**

    iv. Select **+New** under **Linked service**. Configure your Azure SQL Database connection to connect to the SQLDB database.

    v. **Account selection method**: **From Azure subscription**

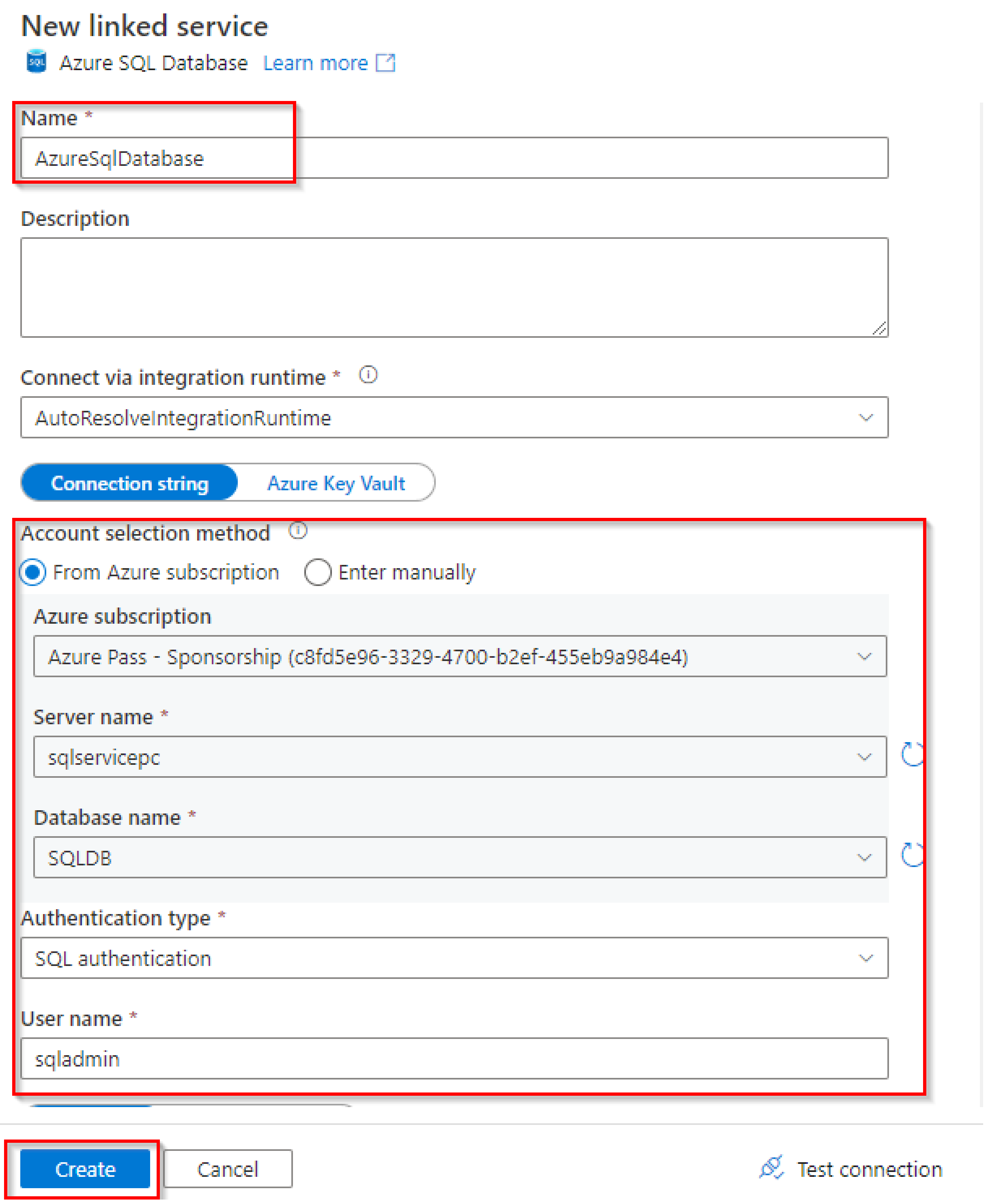
    vi. **Azure subscription**: select the subscription used for this lab.

    vii. **Server name**: select your **sqlservicexx** server.

    viii. **Database name**: **SQLDB**

    ix. **Authentication type**: **SQL authentication**

    x. For **username** use your server admin username, for **Password** use the corresponding password you provided, when setting up the service.


New linked service

Azure SQL Database Learn more

Name *

AzureSqlDatabase

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Connection string | Azure Key Vault

Account selection method

[✓ From Azure subscription] Enter manually

Azure subscription

Azure Pass - Sponsorship (c8fd5e96-3329-4700-b2ef-455eb9a984e4)

Server name *

sqlservicepc

Database name *

SQLDB

Authentication type *

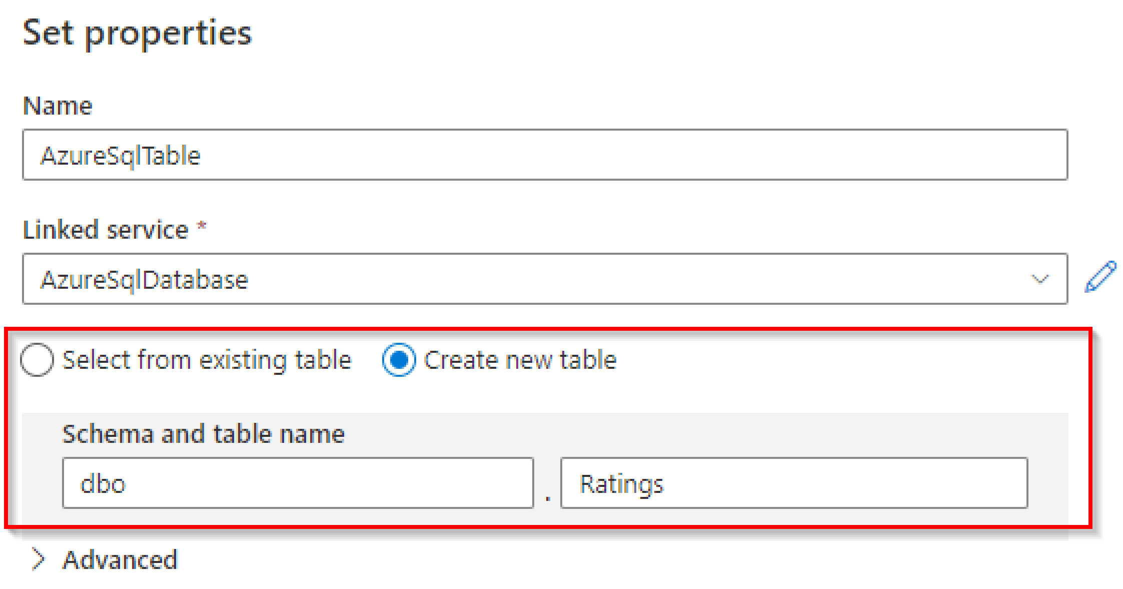
SQL authentication

User name *

sqladmin

xi. Click **Create** when finished.

xii. In the **Set properties** page, select **Create new table** and enter in the schema of **dbo** and the table name of **Ratings**. Click **OK** once completed.



Set properties

Name: AzureSqlTable

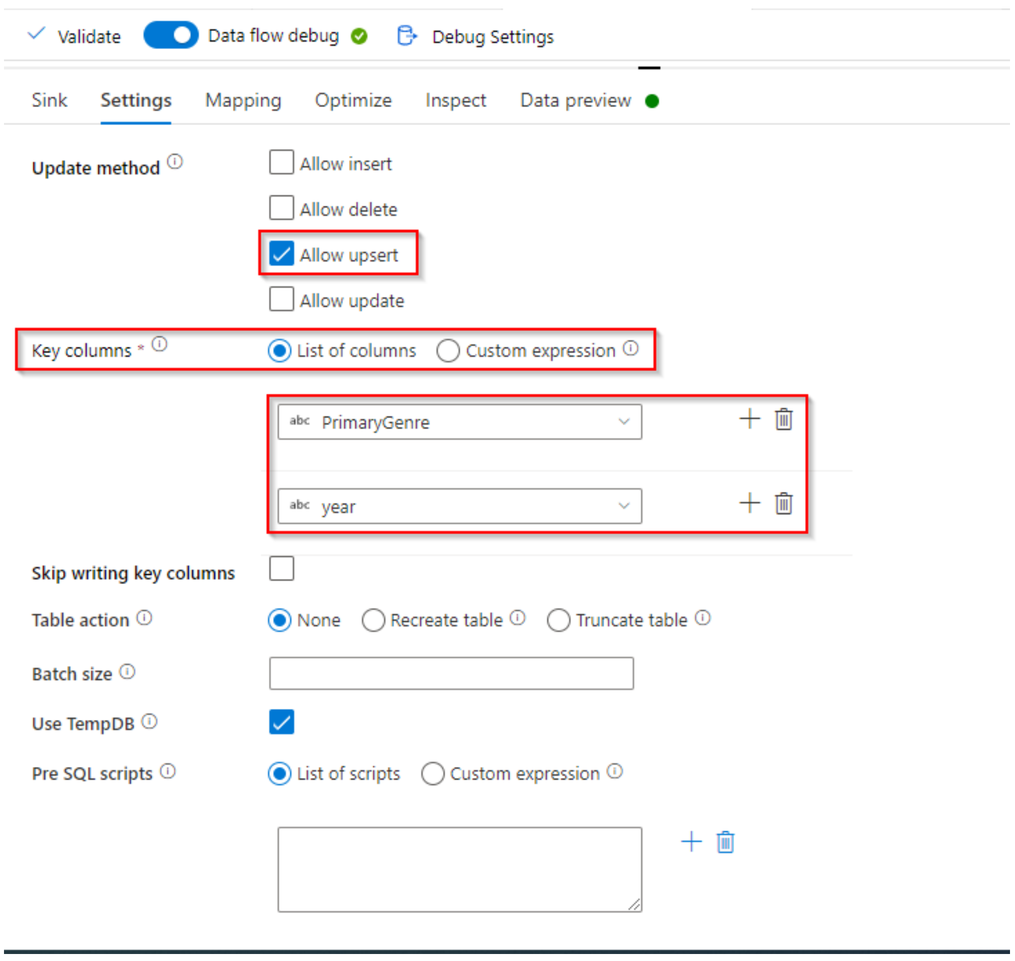
Linked service *: AzureSqlDatabase

( ) Select from existing table [✓ Create new table]

Schema and table name: dbo . Ratings

Advanced

xiii. Since an upsert condition was specified, you need to go to the Settings tab and select **Allow upsert**.



Update method: [ ] Allow insert, [ ] Allow delete, [✓] Allow upsert, [ ] Allow update

Key columns *: [✓ List of columns] ( ) Custom expression

abc PrimaryGenre

abc year

Skip writing key columns: [ ]

Table action: [✓ None] ( ) Recreate table ( ) Truncate table

Batch size:

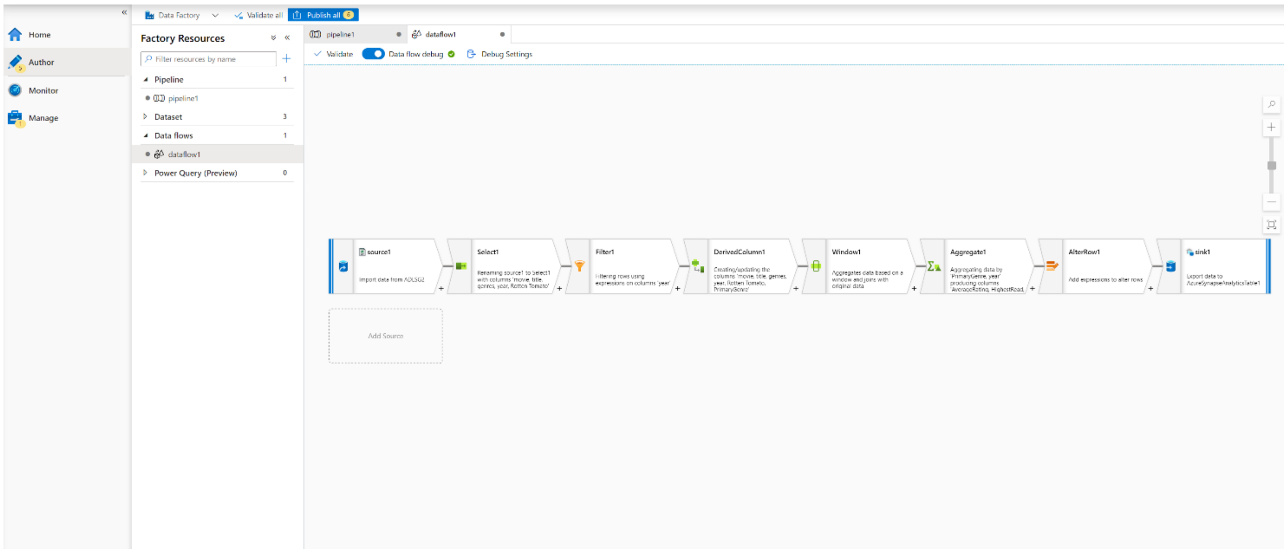
Use TempDB: [✓]

Pre SQL scripts: [✓ List of scripts] ( ) Custom expression

xiv. For **Key columns** select **List of Columns** and add through **+** the two columns PrimaryGenre and year. based on key columns PrimaryGenre and year.

xv. In the **Mapping** pane make sure you untick **Auto mapping**.

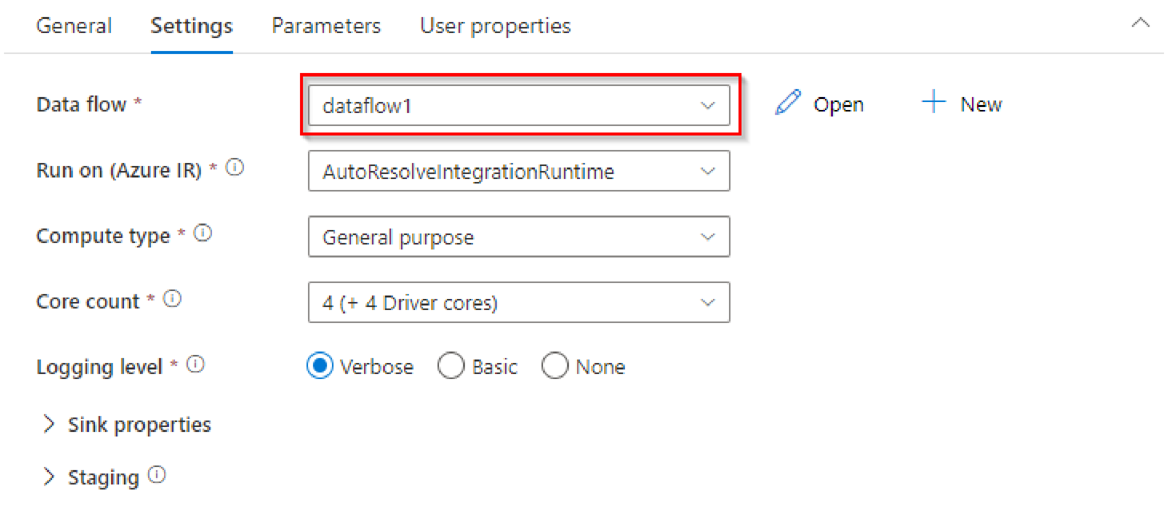
At this point, You have finished building your 8 transformation Mapping Data Flow. It's time to publish and run the pipeline and see the results!



Data Factory
Validate all
Publish all
Home
Author
Monitor
Manage
Factory Resources
Filter resources by name
Pipeline 1
pipeline1
Dataset 3
Data flows 1
dataflow1
Power Query (Preview) 0
pipeline1
dataflow1
Validate
Data flow debug
Debug Settings
source1
Import data from ADLSG2
Select1
Renaming source1 to Select1 with columns 'movie, title, genres, year, Rotten Tomato'
Filter1
Filtering rows using expressions on columns 'year'
DerivedColumn1
Creating/updating the columns 'movie, title, genres, year, Rotten Tomato, PrimaryGenre'
Window1
Aggregates data based on a window and joins with original data
Aggregate1
Aggregating data by 'PrimaryGenre, year' producing columns 'AverageRating, HighestRead,
AlterRow1
Add expressions to alter rows
sink1
Export data to AzureSynapseAnalyticsTable1
Add Source

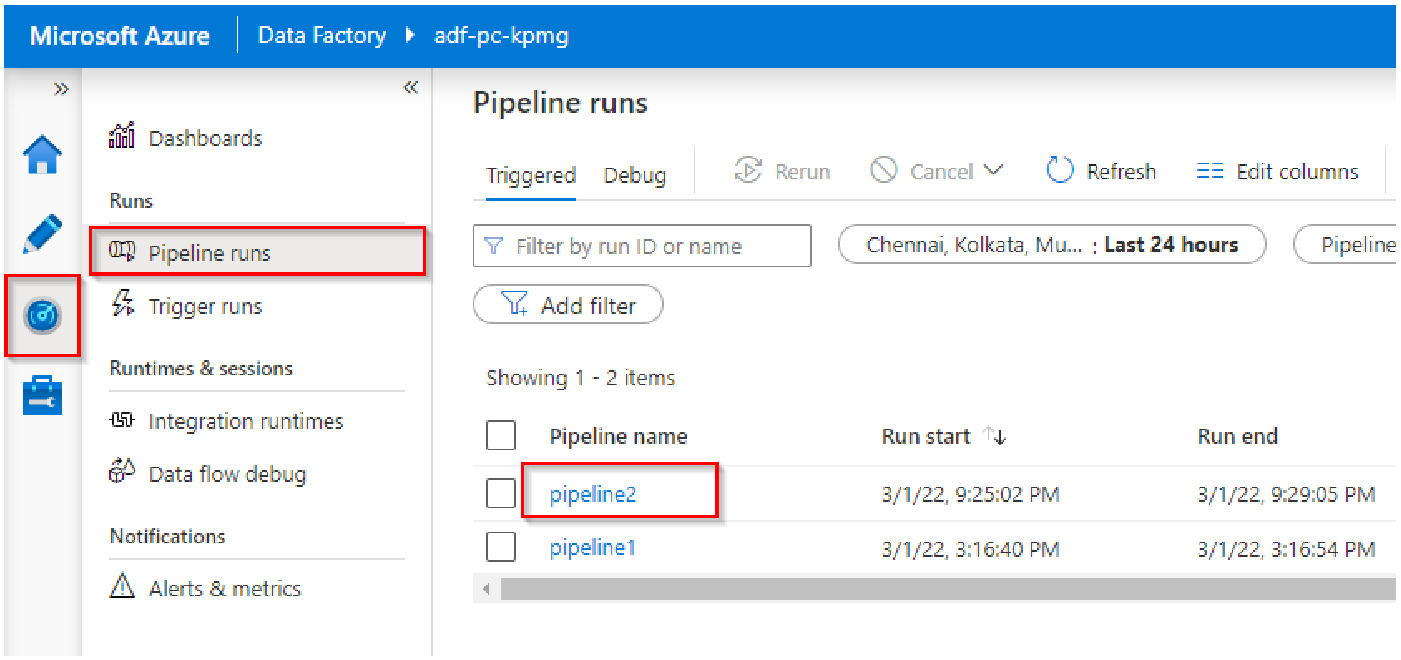
Task 5: Running the Pipeline

1. Go to the pipeline tab in the canvas and trigger the pipeline.



Data flow * dataflow1
Run on (Azure IR) * AutoResolveIntegrationRuntime
Compute type * General purpose
Core count * 4 (+ 4 Driver cores)
Logging level * [✓ Verbose] Basic None

2. Go to monitor, under pipeline runs select the pipeline activity.

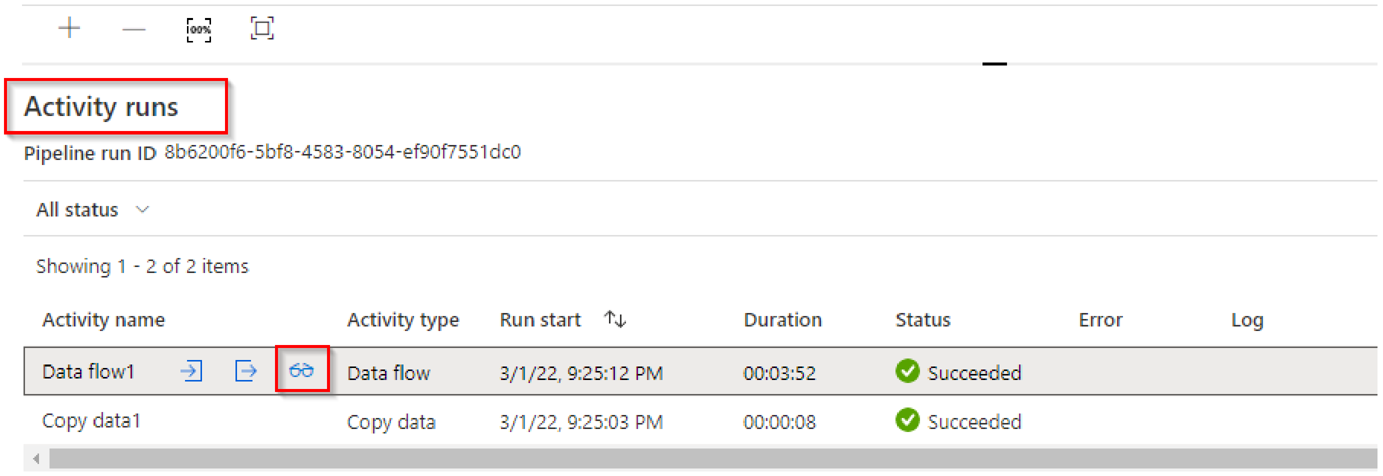


Pipeline runs

Showing 1 - 2 items

Pipeline name	Run start	Run end
pipeline2	3/1/22, 9:25:02 PM	3/1/22, 9:29:05 PM
pipeline1	3/1/22, 3:16:40 PM	3/1/22, 3:16:54 PM

3. Select the pipeline activity run select eyeglass icon.



Activity runs

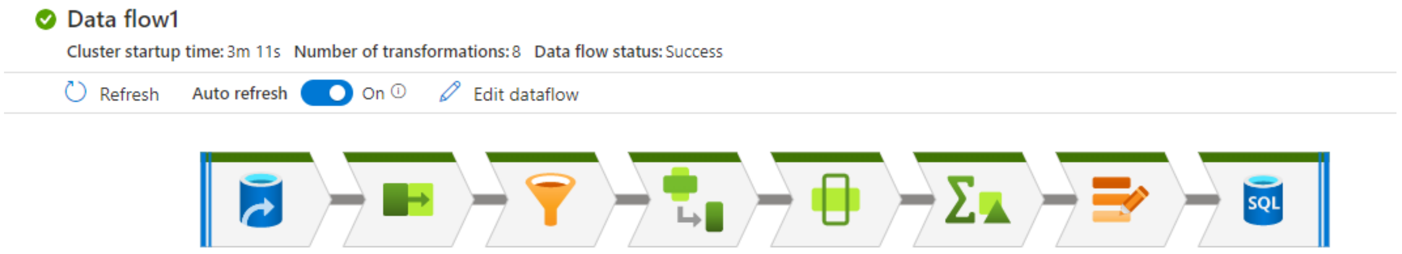
Pipeline run ID 8b6200f6-5bf8-4583-8054-ef90f7551dc0

All status

Showing 1 - 2 of 2 items

Activity name	Activity type	Run start	Duration	Status	Error	Log
Data flow1	Data flow	3/1/22, 9:25:12 PM	00:03:52	✓ Succeeded		
Copy data1	Copy data	3/1/22, 9:25:03 PM	00:00:08	✓ Succeeded		

4. Monitor the status of each transformation.



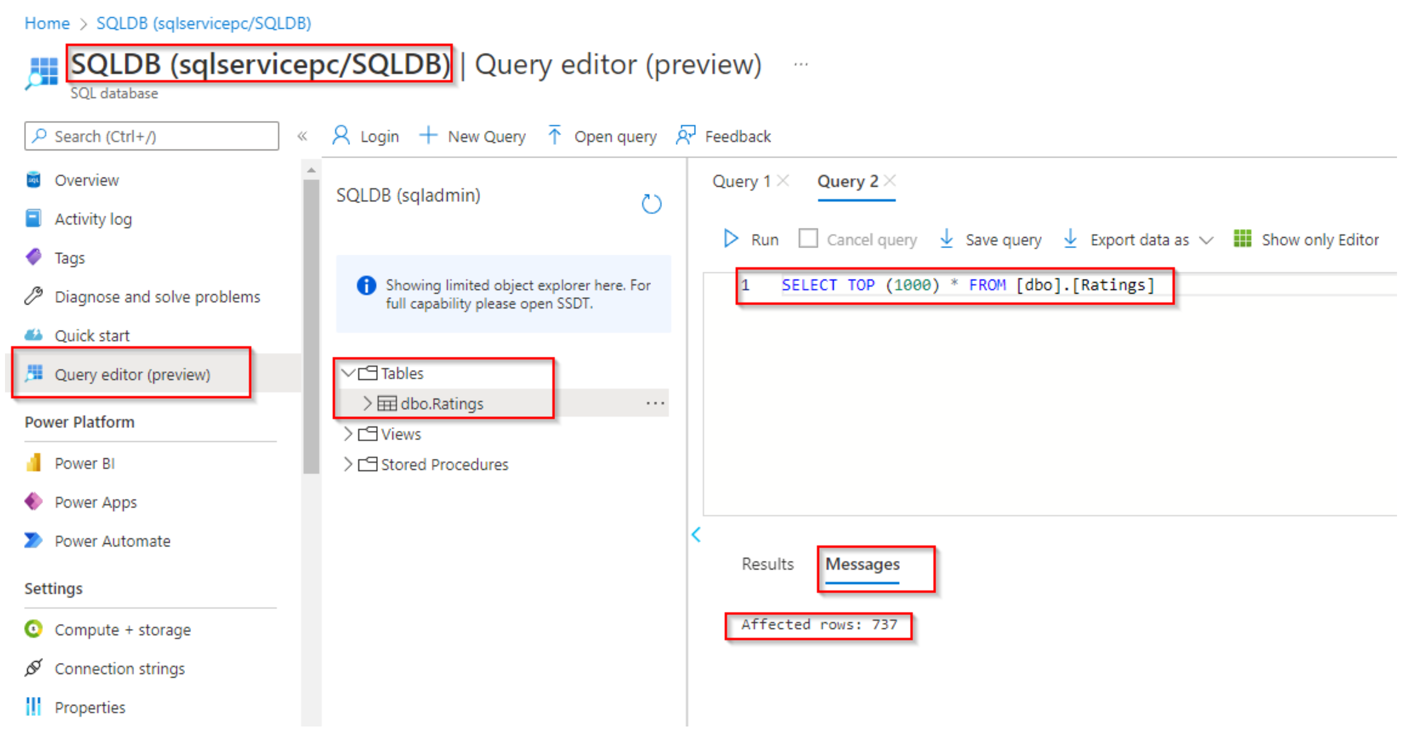
Data flow1

Cluster startup time: 3m 11s Number of transformations: 8 Data flow status: Success

5. If you used the same logic described in this lab, your Data Flow should have written **737 rows** to your SQL Database.

6. To verify this number navigate to you Azure SQL Database.

7. In your SQL Database, navigate to the **Query Editor** under Tables click on **…** and **select Top 1000 Rows** and verify the No. of rows in **Messages** Tab.



Showing limited object explorer here. For full capability please open SSDT.

```
SELECT TOP (1000) * FROM [dbo].[Ratings]
```

```
Affected rows: 737
```