

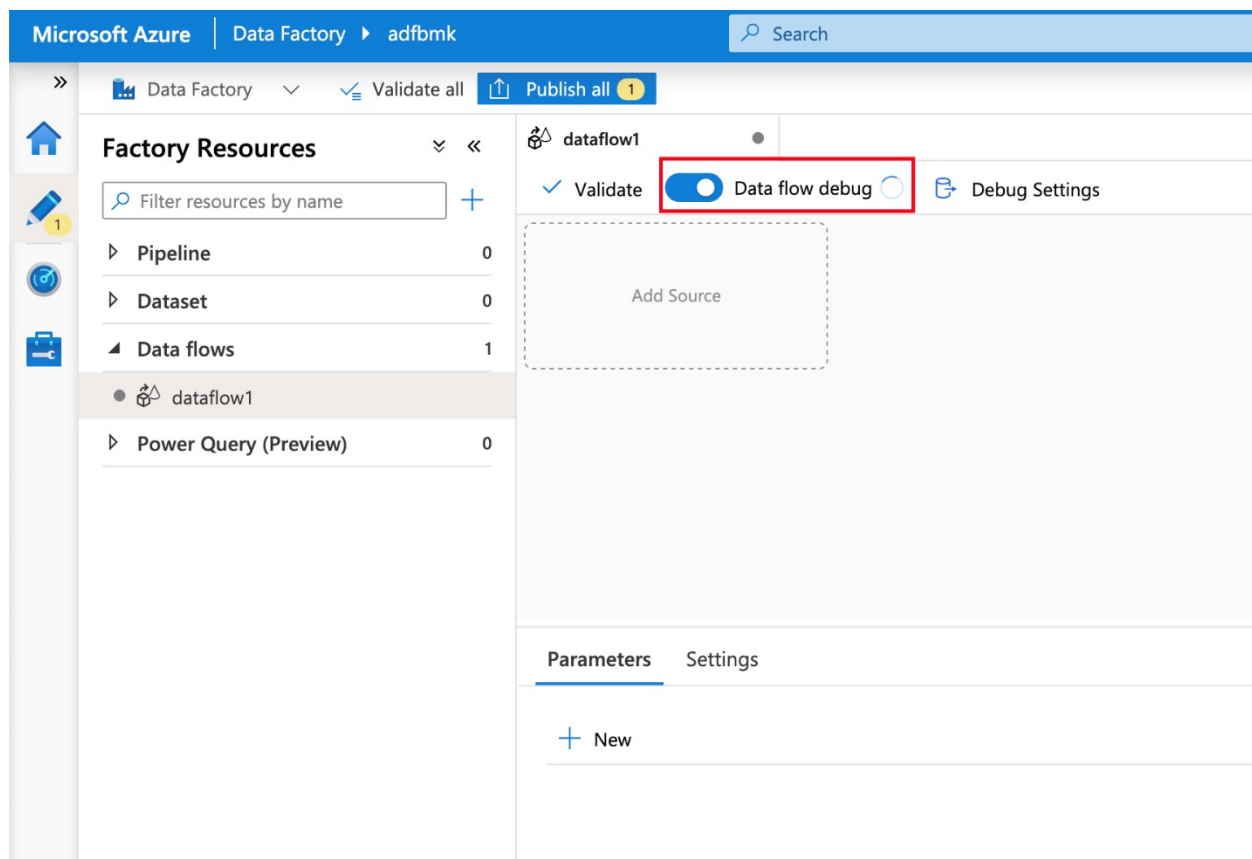
# Prepare and transform data with Azure Synapse Analytics

## Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

You can natively perform data transformations with Azure Synapse pipelines code free using the Mapping Data Flow task. Mapping Data Flows provide a fully visual experience with no coding required. Your data flows will run on your own execution cluster for scaled-out data processing. Data flow activities can be operationalized via existing Data Factory scheduling, control, flow, and monitoring capabilities.

When building data flows, you can enable debug mode, which turns on a small interactive Spark cluster. Turn on debug mode by toggling the slider at the top of the authoring module. Debug clusters take a few minutes to warm up, but can be used to interactively preview the output of your transformation logic.



Turn on debug mode by toggling the slider at the top of the authoring module.

With the Mapping Data Flow added, and the Spark cluster running, this will enable you to perform the transformation, and run and preview the data. No coding is required as Azure Data Factory handles all the code translation, path optimization, and execution of your data flow jobs.

## Add source data to the Mapping Data Flow

Open the Mapping Data Flow canvas. Click on the Add Source button in the Data Flow canvas. In the source dataset dropdown, select your data source, in this case the ADLS Gen2 dataset is used in this example:

The screenshot shows the Azure Data Factory Mapping Data Flow canvas. At the top, there is a blue arrow-shaped button labeled 'MoviesADLS' with a small icon and the text 'Columns: 0 total'. Below this is a dashed box labeled 'Add Source'. The bottom section of the canvas is a settings panel with tabs: 'Source Settings', 'Source Options', 'Projection', 'Optimize', 'Inspect', and 'Data Preview' (which is active and has a green dot). The 'Source Settings' tab is selected. It contains the following fields and options:

- Output stream name \***: A text box containing 'MoviesADLS'. To its right is a link icon and the text 'Documentation'.
- Source dataset \***: A dropdown menu showing 'DelimitedText1'. To its right are 'Edit' (with a pencil icon) and 'New' (with a plus icon).
- Options**: Three checkboxes with labels and information icons:
  - ☒ Allow schema drift
  - ☐ Infer drifted column types
  - ☐ Validate schema
- Skip line count**: An empty text box.
- Sampling \***: Two radio buttons, 'Enable' and 'Disable' (which is selected). To the right is an information icon.

There are a couple of points to note:

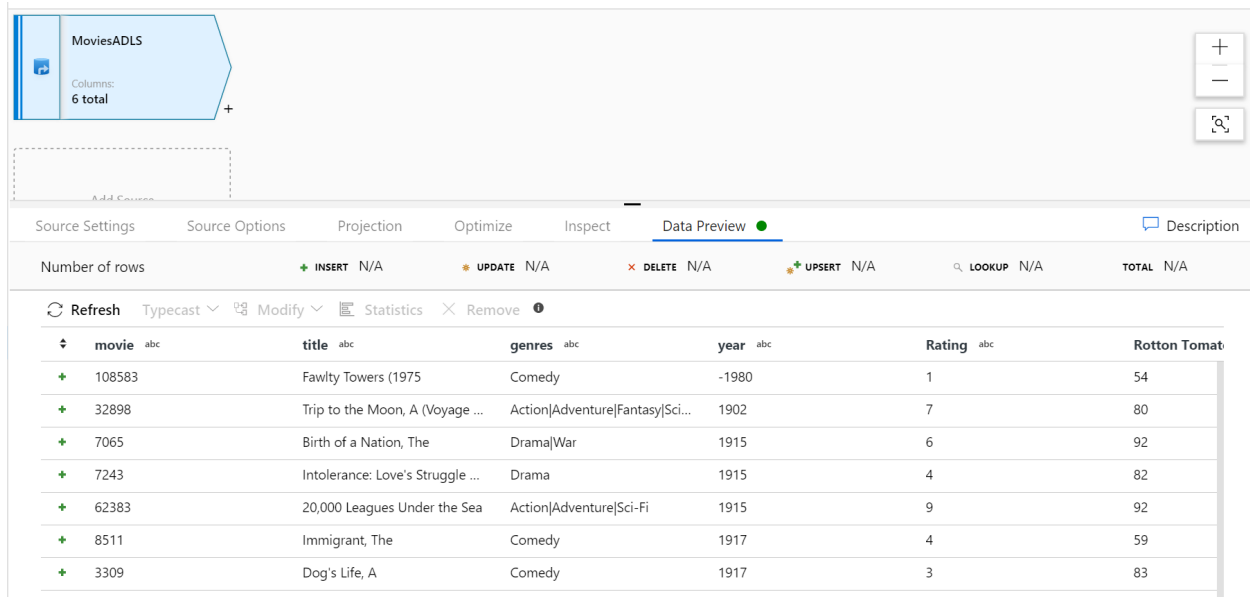
- If your dataset is pointing at a folder with other files and you only want to use one file, you may need to create another dataset or utilize parameterization to make sure only a specific file is read
- If you have not imported your schema in your ADLS, but have already ingested your data, go to the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

Mapping Data Flow follows an extract, load, transform (ELT) approach and works with staging datasets that are all in Azure. Currently the following datasets can be used in a source transformation:

- Azure Blob Storage (JSON, Avro, Text, Parquet)
- Azure Data Lake Storage Gen1 (JSON, Avro, Text, Parquet)
- Azure Data Lake Storage Gen2 (JSON, Avro, Text, Parquet)
- Azure Synapse Analytics
- Azure SQL Database
- Azure CosmosDB

Azure Data Factory has access to over 80 native connectors. To include data from those other sources in your data flow, use the Copy Activity to load that data into one of the supported staging areas.

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 a snapshot of what your data looks like when it is at each transformation.



The screenshot shows the 'Data Preview' tab in Azure Data Factory. At the top, there's a 'MoviesADLS' connector with 'Columns: 6 total'. Below it, a table of movie data is displayed. The table has columns: movie, title, genres, year, Rating, and Rotton Tomat. The data is sorted by movie ID in descending order. The table is shown with a 'Refresh' button and various action icons like 'Typecast', 'Modify', 'Statistics', and 'Remove'.

movie	title	genres	year	Rating	Rotton Tomat
108583	Fawlty Towers (1975)	Comedy	-1980	1	54
32898	Trip to the Moon, A (Voyage ...	Action Adventure Fantasy Sci...	1902	7	80
7065	Birth of a Nation, The	Drama War	1915	6	92
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

## Use transformations in the Mapping Data Flow

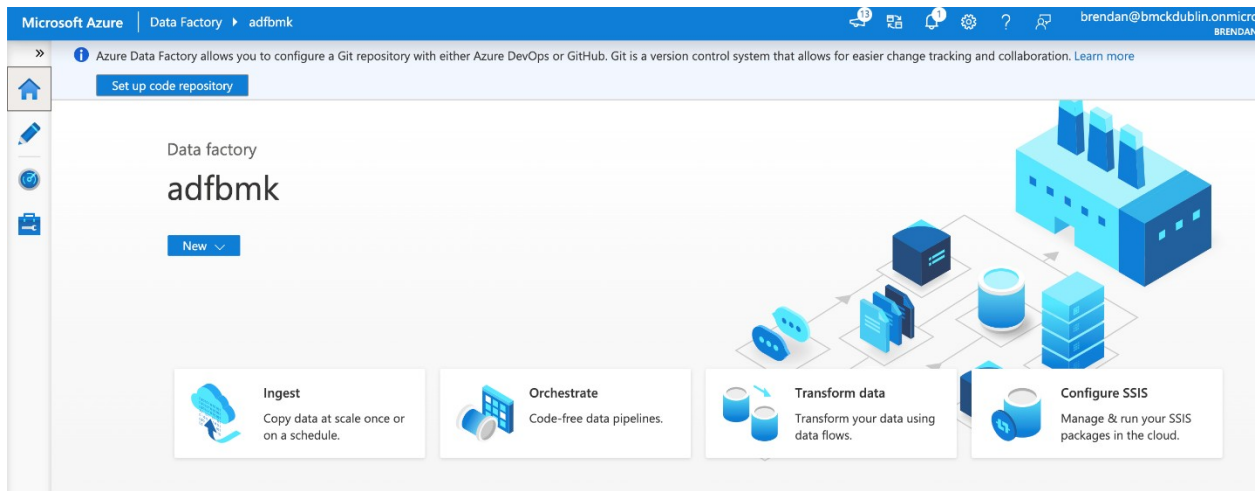
Now that you have moved the data into Azure Data Lake Store Gen2, you are ready to build a Mapping Data Flow that will transform your data at scale via a spark cluster and then load it into a Data Warehouse.

The main tasks for this are as follows:

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

# Task 1: Preparing the environment

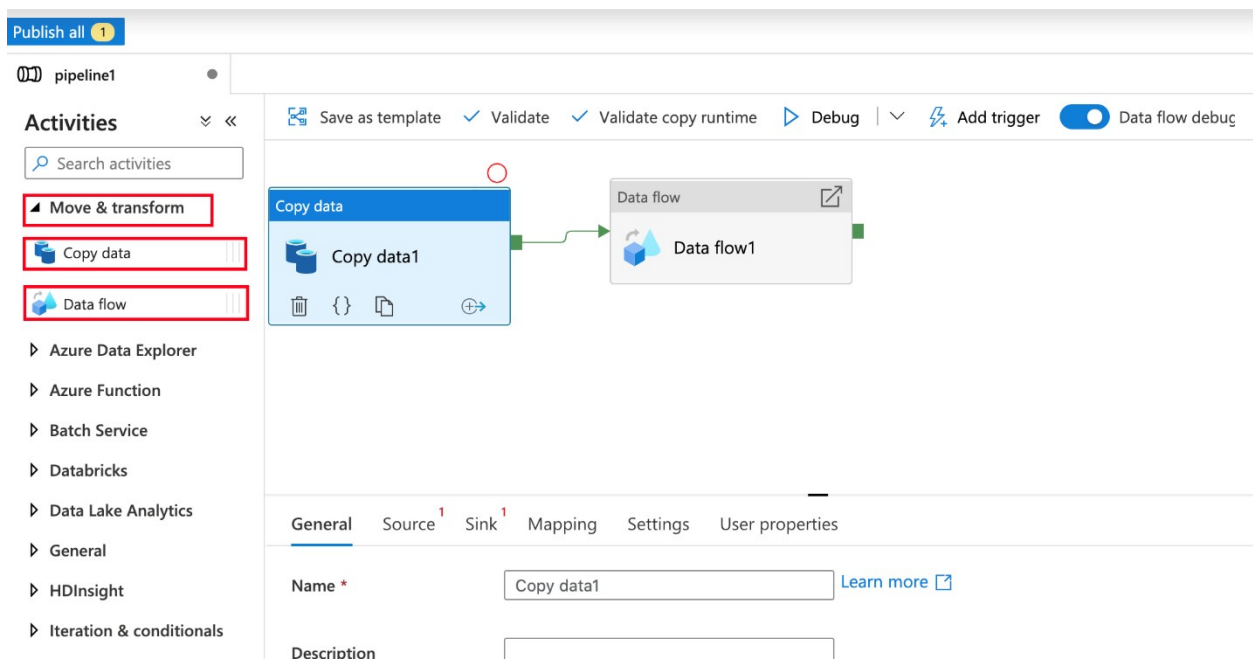
1. Open Azure Data Factory Studio and click on Orchestrate. This will add a new Pipeline.



Open Azure Data Factory Studio and click on Orchestrate

**Note: Data Flow clusters take 5-7 minutes to warm up.**

2. **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. In the blade that pops up, click **Create new Data Flow** and select **Mapping Data Flow** and then click **OK**. Click on the **pipeline1** tab and drag the green box from your Copy activity to the Data Flow Activity to create an on success condition. You will see the following in the canvass:



Add a Data Flow activity.

## Task 2: Adding a Data Source

**Add an ADLS source.** Double-click on the Mapping Data Flow object in the canvas. Click on the Add Source button in the Data Flow canvas. In the **Source dataset** dropdown, select your **ADLSG2** dataset used in your Copy activity.

MoviesADLS  
Columns:  
0 total

Add Source

Source Settings | Source Options | Projection | Optimize | Inspect | Data Preview ●

Output stream name \*  [Documentation](#)

Source dataset \*  [Edit](#) [New](#)

Options

- ☒ 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 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.

The screenshot displays the Microsoft Azure Data Factory console for a workspace named 'cto-data-factory'. The left sidebar shows the 'Factory Resources' tree with 'Pipelines' (1), 'Datasets' (2), and 'Data flows' (1). The main canvas shows a pipeline named 'pipeline1' with a single dataflow node 'dataflow1'. A source node 'source1' is highlighted, showing 'Columns: 6 total'. A red box highlights the '+' icon next to the source node, which has opened a dropdown menu. The menu is divided into sections: 'Multiple inputs/outputs' (Conditional Split, Exists), 'Schema modifier' (Select, Surrogate Key), 'Row modifier' (Sort), and 'Destination' (Sink). The 'Select' option under 'Schema modifier' is highlighted. Below the canvas, the 'Source settings' tab is active, showing configuration for 'source1'. The 'Output stream name' is 'source1', and the 'Source dataset' is 'ADLSG2'. Under 'Options', 'Allow schema drift' is checked, while 'Infer drifted column types', 'Validate schema', and 'Multiline rows' are unchecked. 'Skip line count' is empty, and 'Sampling' is set to 'Disable'.

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

Select settings **Optimize** Inspect Data preview ●

Output stream name \*  [Learn more](#)

Incoming stream \*

Options

- ☒ Skip duplicate input columns ⓘ
- ☒ Skip duplicate output columns ⓘ

Input columns \* ☐ Auto mapping ⓘ

6 mappings: All inputs mapped

<input type="checkbox"/>	source1's column		Name as		
<input type="checkbox"/>	movie	→	movie		+ <input type="button" value="Delete"/>
<input type="checkbox"/>	title	→	title		+ <input type="button" value="Delete"/>
<input type="checkbox"/>	genres	→	genres		+ <input type="button" value="Delete"/>
<input type="checkbox"/>	year	→	year		+ <input type="button" value="Delete"/>
<input type="checkbox"/>	Rating	→	Rating		+ <input type="button" value="Delete"/>
<input type="checkbox"/>	Rotten Tomato	→	Rotten Tomato		+ <input type="button" value="Delete"/>

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.

Filter settings **Optimize** Inspect Data preview ●

Output stream name \*  [Learn more](#)

Incoming stream \*

Filter on \*

You can use the expression builder's embedded Data preview pane to verify your condition is working properly.

## Visual expression builder

FUNCTIONS

Filter...

All Functions Input schema Parameters

abc movie

abc title

abc genres

`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 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.

Derived column's settings Optimize Inspect Data preview

Output stream name \* DerivedPrimaryGenre [Learn more](#)

Incoming stream \* Filter1

Columns \* PrimaryGenre `iif(locate('|',genres) > 1,left(genres, locate('|',genres)-1),genres)` abc + 📄 🗑️

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.

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, and calculate a new column called RatingsRank that is equal to the rank each movie has within its specific genre-year.



Window Settings

Optimize

Inspect

Data Preview

Output stream name \*

RankMoviesByRatings

Documentation

Incoming stream \*

DerivePrimaryGenre

1. Over

2. Sort

3. Range by

4. Window columns

DerivePrimaryGenre's column

Name as

abc PrimaryGenre

PrimaryGenre

abc year

year

Window Settings

Optimize

Inspect

Data Preview

Output stream name \*

RankMoviesByRatings

Documentation

Incoming stream \*

DerivePrimaryGenre

1. Over

2. Sort

3. Range by

4. Window columns

DerivePrimaryGenre's column

Order

Nulls first

abc Rotten Tomato

↓

☒

+

Window Settings

Optimize

Inspect

Data Preview

Output stream name \*

RankMoviesByRatings

Documentation

Incoming stream \*

DerivePrimaryGenre

1. Over

2. Sort

3. Range by

4. Window columns

Option \*

☒ Range by current row offset

☐ Range by column value

Unbounded

☒

Window Settings   Optimize   Inspect   Data Preview ●

Output stream name \*  [Documentation](#)

Incoming stream \* [DerivePrimaryGenre](#)

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

123

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, let's group movies by PrimaryGenre and year.

Aggregate settings   Optimize   Inspect   Data preview ●

Output stream name \*  [Learn more](#)

Incoming stream \*

**Group by**   Aggregates

RankMoviesByRatings's column	Name as	
<input type="text" value="PrimaryGenre"/>	<input type="text" value="PrimaryGenre"/>	+
<input type="text" value="year"/>	<input type="text" value="year"/>	+

In the Aggregates tab, you can aggregations calculated over the specified group by columns. For every genre and year, let's 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 number of rows in your transformation stream and only propagates the group by and aggregate columns specified in the transformation.

Aggregate settings [Optimize](#) [Inspect](#) [Data preview](#) ● [Description](#) ^

Output stream name \*  [Learn more](#) 🔗

Incoming stream \*

Group by [Aggregates](#)

Grouped by: PrimaryGenre, year

AverageRating	avg(toInteger((Rotten Tomato)))	1.2	+	📄	🗑️
HighestRated	first(title)	abc	+	📄	🗑️
LowestRated	last(title)	abc	+	📄	🗑️
NumberOfMovies	count()	121	+	📄	🗑️

• 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.

Alter row settings [Optimize](#) [Inspect](#) [Data preview](#) ● [Description](#) ^

Output stream name \*  [Learn more](#) 🔗

Incoming stream \*

Alter row conditions \* ⓘ   🔗 + 🗑️

## Task 4: Writing to a Data Sink

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

a) Add a **Sink** by clicking on the + **icon** next to your Upsert transformation and clicking Sink under Destination.

b) In the Sink tab, create a new data warehouse dataset via the + **New button**.

c) Select **Azure Synapse Analytics** from the tile list.

d) Select a new linked service and configure your Azure Synapse Analytics connection to connect to the DWDB database created in Module 5. Click **Create** when finished.

## New linked service (Azure Synapse Analytics (formerly SQL DW))

Name \*

AzureSynapseAnalytics1

Description

Connect via integration runtime \*

AutoResolveIntegrationRuntime

Connection string

Azure Key Vault

Account selection method

☒ From Azure subscription

☐ Enter manually

Azure subscription

Server name \*

dwhservicecto

Database name \*

DWDB

Authentication type \*

SQL authentication

User name \*

ctosqladmin

Password

Azure Key Vault

Password \*

••••••••

Additional connection properties

+ New

Annotations

+ New

Parameters

Advanced ⓘ

e) In the dataset configuration, select **Create new table** and enter in the schema of **dbo** and the table name of **Ratings**. Click **OK** once completed.

## Set properties

Name

AzureSynapseAnalyticsTable1

Linked service \*

AzureSynapseAnalytics1

[Edit connection](#)

☐ Select from existing table ☒ Create new table

Schema and table name

dbo

Ratings

Advanced

f) Since an upsert condition was specified, you need to go to the Settings tab and select 'Allow upsert' based on key columns PrimaryGenre and year.

[Sink](#) [Settings](#) [Mapping](#) [Optimize](#) [Inspect](#) [Data preview](#) ●

Update method

☐ Allow insert

☐ Allow delete

☒ Allow upsert

☐ Allow update

Key columns \* ⓘ

PrimaryGenre

+

year

+

Skip writing key columns

☐

Table action

☒ None

☐ Recreate table

☐ Truncate table

Enable staging

☒

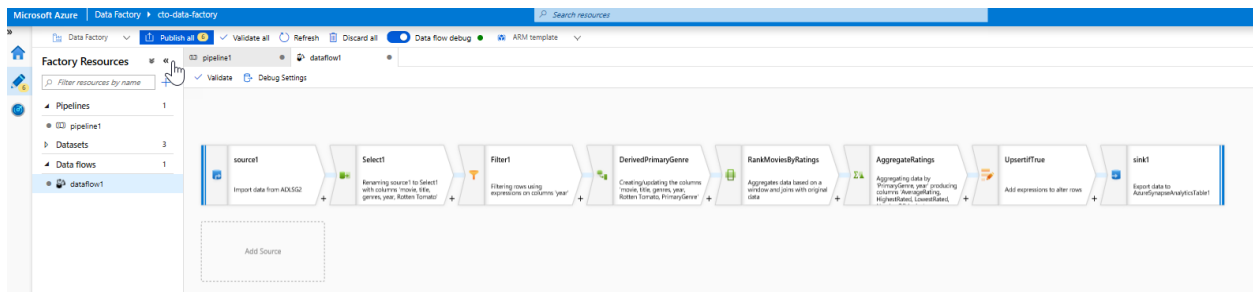
Batch size

ⓘ

Pre SQL scripts

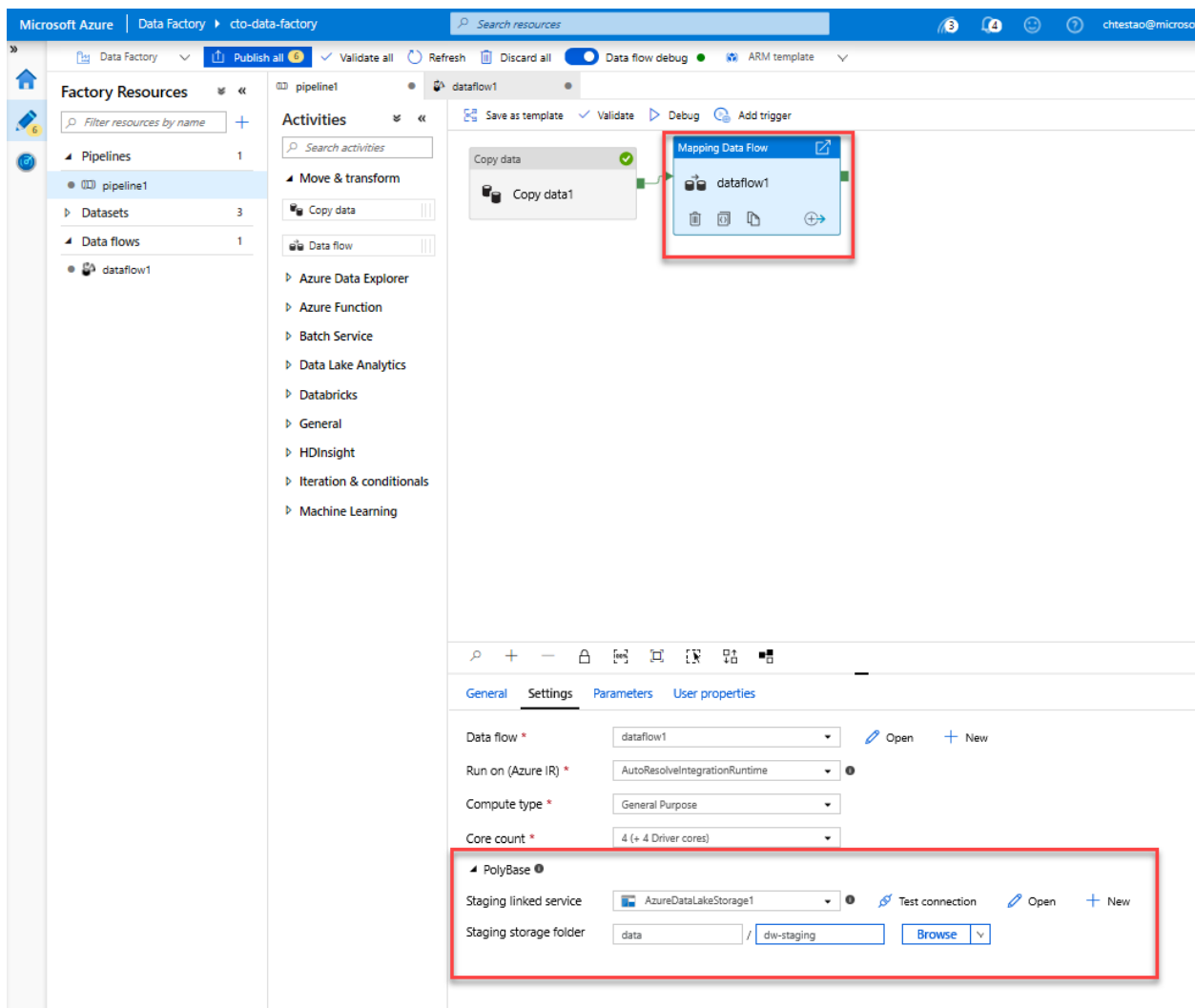
Post SQL scripts

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



## Task 5: Running the pipeline

1. Go to the pipeline1 tab in the canvas. Because Azure Synapse Analytics in Data Flow uses [PolyBase](#), you must specify a blob or ADLS staging folder. In the Execute Data Flow activity's settings tab, open up the PolyBase accordion and select your ADLS linked service and specify a staging folder path.



2. Before you publish your pipeline, run another debug run to confirm it's working as expected. Looking at the Output tab, you can monitor the status of both activities as they are running.

3. Once both activities succeeded, you can click on the eyeglasses icon next to the Data Flow activity to get a more in depth look at the Data Flow run.

4. If you used the same logic described in this lab, your Data Flow will write 737 rows to your SQL DW. You can go into [SQL Server Management Studio](#) to verify the pipeline worked correctly and see what got written.

SQLQuery1.sql - dw...tosqladmin (2184)) \* X Object Explorer Details

```
Select count(*) as TotalCount from dbo.Ratings
```

```
Select * from dbo.Ratings
```

100 %

Results Messages

	TotalCount
1	737

	PrimaryGenre	year	AverageRating	HighestRated	LowestRated	NumberOfMovies
1	Action	1955	82.5	Dam Busters, The	To Hell and Back	4
2	(no genres listed)	1994	83	Freaky Friday	Freaky Friday	2
3	(no genres listed)	2012	63	Doctor Who: The Time of the Doctor	Doctor Who: The Time of the Doctor	2
4	Thriller	1963	51	Seven Days in May	Seven Days in May	2
5	(no genres listed)	2009	50	Boy Crazy	Boy Crazy	2
6	(no genres listed)	1964	89	Scorpio Rising	Scorpio Rising	2