dp-200-07_instructions.md

DP 200 - Implementing a Data Platform Solution

Lab 7 - Orchestrating Data Movement with Azure Data Factory

Estimated Time: 70 minutes

Pre-requisites: It is assumed that the case study for this lab has already been read. It is assumed that the content and lab for module 1: Azure for the Data Engineer has also been completed

- **Azure subscription**: If you don't have an Azure subscription, create a free account before you begin.
- **Azure Data Lake Storage Gen2 storage account**: If you don't have an ADLS Gen2 storage account, see the instructions in Create an ADLS Gen2 storage account.
- **Azure Synapse Analytics**: If you don't have a Azure Synapse Analytics account, see the instructions in Create a SQL DW account.

Lab files: The files for this lab are located in the *Allfiles\Labfiles\Starter\DP-200.7* folder.

Lab overview

In this module, students will learn how Azure Data factory can be used to orchestrate the data movement from a wide range of data platform technologies. They will be able to explain the capabilities of the technology and be able to set up an end to end data pipeline that ingests data from SQL Database and load the data into Azure Synapse Analytics. The student will also demonstrate how to call a compute resource.

Lab objectives

After completing this lab, you will be able to:

1. Setup Azure Data Factory
2. Ingest data using the Copy Activity
3. Use the Mapping Data Flow task to perform transformation
4. Perform transformations using a compute resource

Scenario

You are assessing the tooling that can help with the extraction, load and transforming of data into the data warehouse, and have asked a Data Engineer within your team to show a proof of concept of Azure Data Factory to explore the transformation capabilities of the product. The proof of concept does not have to be related to AdventureWorks data, and you have given them freedom to pick a dataset of thier choice to showcase the capabilities.

In addition, the Data Scientists have asked to confirm if Azure Databricks can be called from Azure Data Factory. To that end, you will create a simple proof of concept Data Factory pipeline that calls Azure Databricks as a compute resource.

At the end of this lad, you will have:

1. Setup Azure Data Factory
2. Ingested data using the Copy Activity
3. Used the Mapping Data Flow task to perform transformation
4. Performed transformations using a compute resource

> **IMPORTANT**: As you go through this lab, make a note of any issue(s) that you have encountered in any provisioning or configuration tasks and log it in the table in the document located at *\Labfiles\DP-200-Issues-Doc.docx*. Document the Lab number, note the technology, Describe the issue, and what was the resolution. Save this document as you will refer back to it in a later module.

Exercise 1: Setup Azure Data Factory

Estimated Time: 15 minutes

Individual exercise

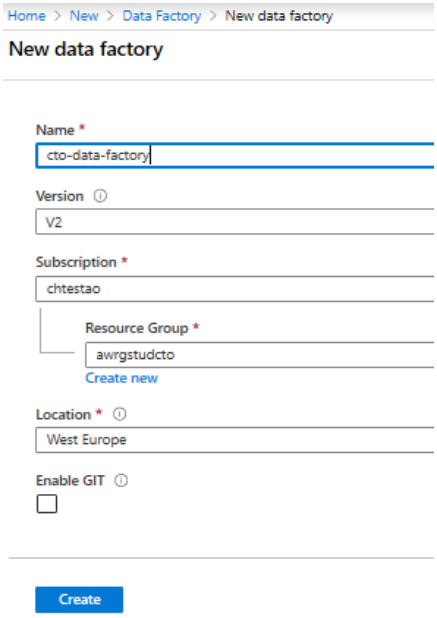
The main task for this exercise are as follows:

1. Setup Azure Data Factory

Task 1: Setting up Azure Data Factory.

Create your data factory: Use the Azure Portal to create your Data Factory.

1. In Microsoft Edge, go to the Azure portal tab, click on the + **Create a resource** icon, type **factory**, and then click **Data Factory** from the resulting search, and then click **Create**.
2. In the New Data Factory screen, create a new Data Factory with the following options, then click **Create**:
    - **Name**: xx-data-factory, where xx are your initials
    - **Version**: V2
    - **Subscription**: Your subscription
    - **Resource group**: awrgstudxx
    - **Location**: select the location closest to you
    - **Enable GIT**: unchecked
    - Leave other options to their default settings



> **Note**: The creation of the Data Factory takes approximately 1 minute.

> **Result**: After you completed this exercise, you have created an instance of Azure Data Factory

Exercise 2: Ingest data using the Copy Activity

Estimated Time: 15 minutes

Individual exercise

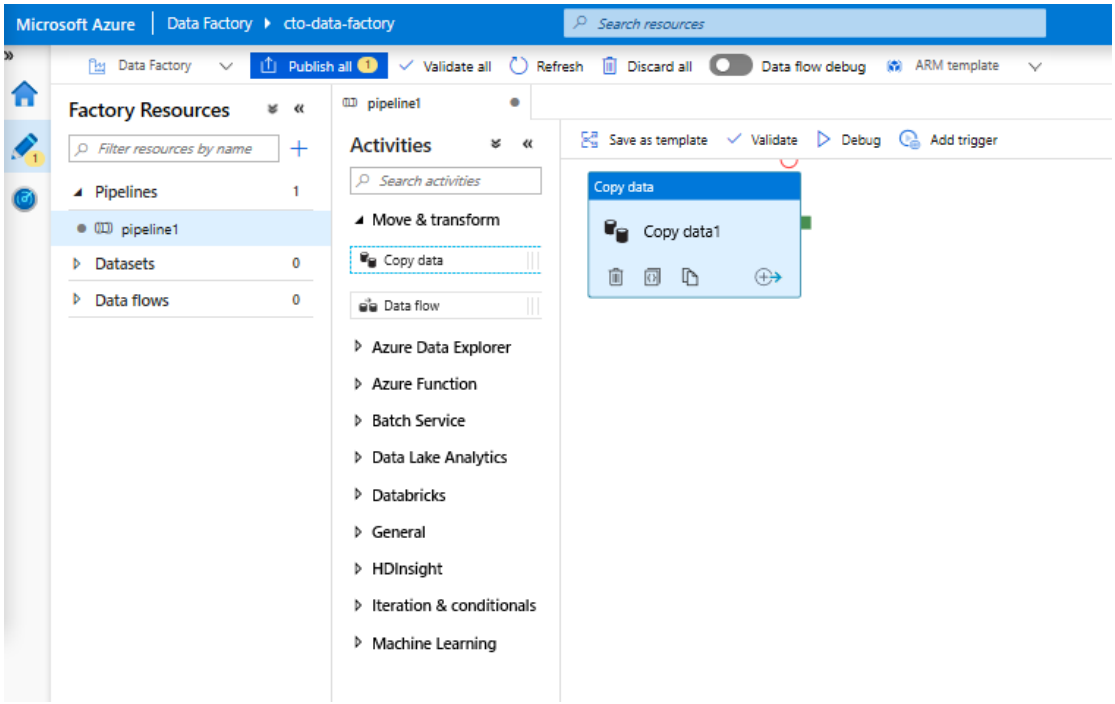
The main tasks for this exercise are as follows:

1. Add the Copy Activity to the designer

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

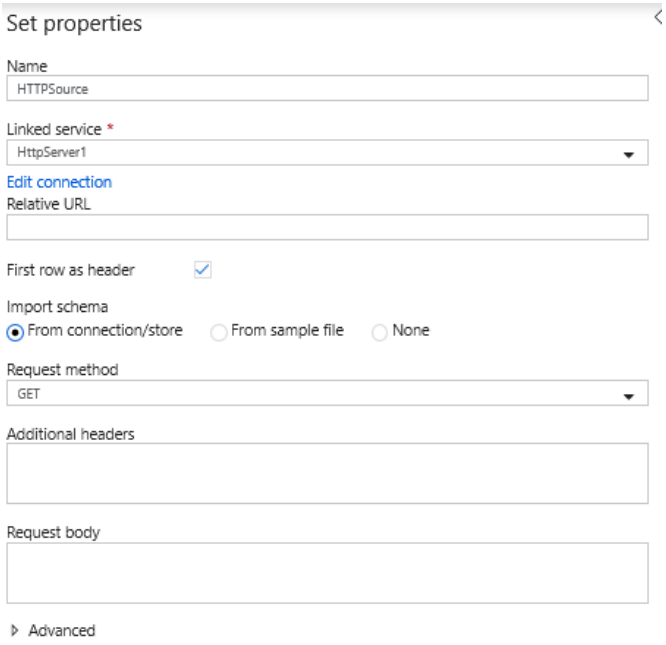
1. On the deployment successful message, click on the button **Go to resource**.
2. In the xx-data-factory screen, in the middle of the screen, click on the button, **Author & Monitor**
3. **Open the authoring canvas** If coming from the ADF homepage, click on the **pencil icon** on the left sidebar or the **create pipeline button** to open the authoring canvas.
4. **Create the pipeline** Click on the + button in the Factory Resources pane and select Pipeline
5. **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

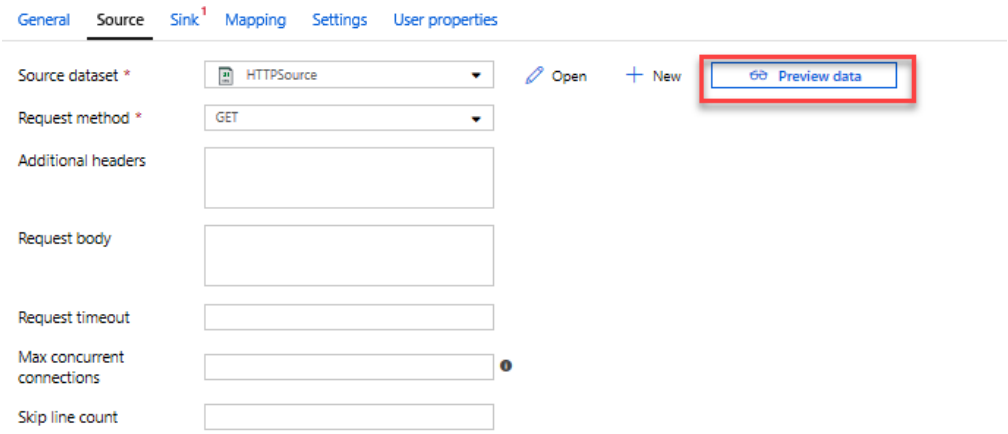
1. In the Source tab of the Copy activity settings, click + **New**
2. In the data store list, select the **HTTP** tile and click continue
3. In the file format list, select the **DelimitedText** format tile and click continue
4. In Set Properties blade, give your dataset an understandable name such 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, specify the url of the moviesDB csv file. You can access the data with no authentication required using the following endpoint:

   https://raw.githubusercontent.com/djpmsft/adf-ready-demo/master/moviesDB.csv
6. Place this in the **Base URL** text box.
7. 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 followinf screen



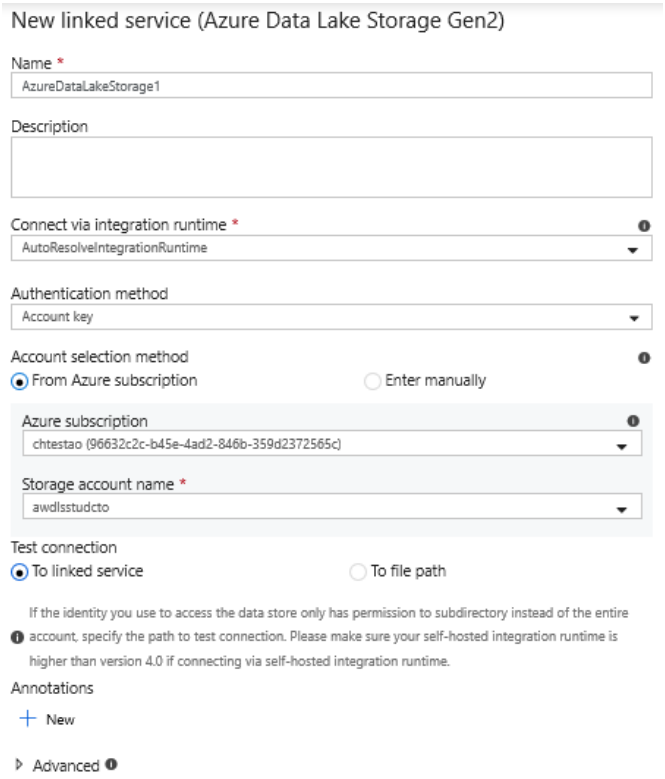
- Click **OK** once completed.

a. 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.



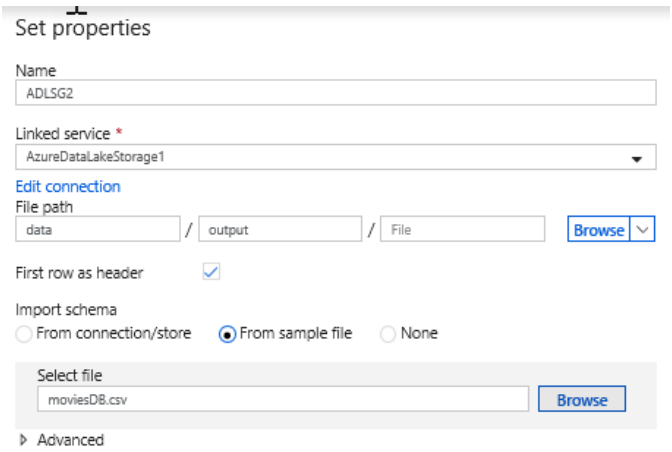
Task 3: Create a new ADLS Gen2 dataset sink

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 **ADLSG2** 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 **awdlsstudxx**. 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 file system **data**. While the folder can be dynamically created, the file system must exist prior to writing to it. Set **First row as header** to be true. You can either Import schema from **sample file** (use the moviesDB.csv file from **Labfiles\Starter\DP-200.7\SampleFiles**)



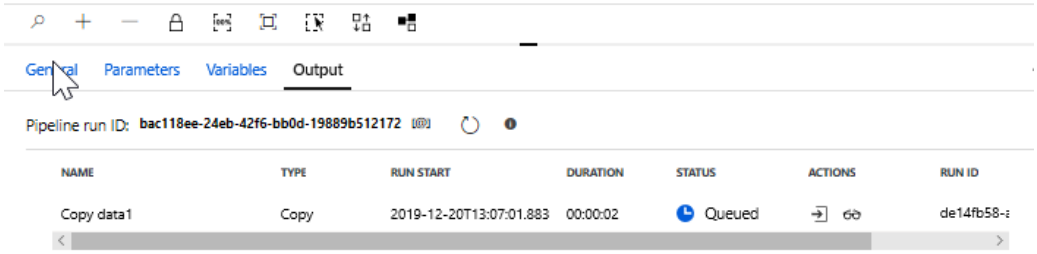
8. Click **OK** once completed.

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.



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 3: Transforming Data with Mapping Data Flow

Estimated Time: 30 minutes

Individual exercise

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

The main tasks for this exercise are as follows:

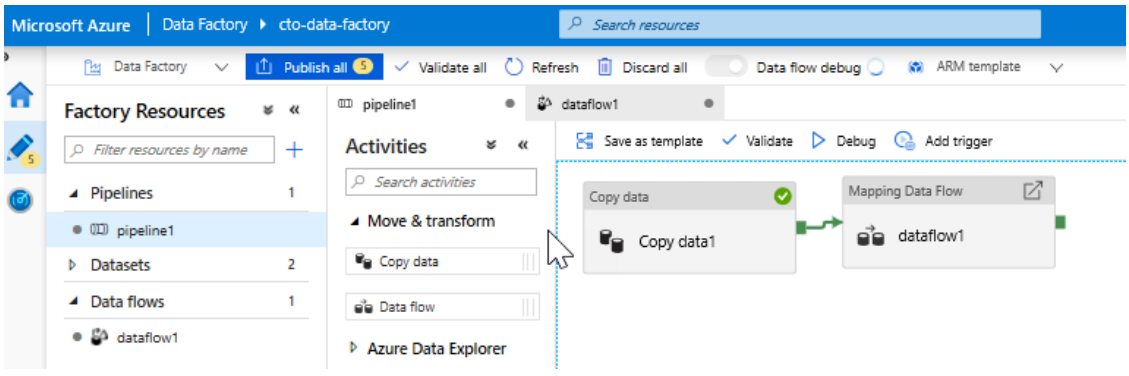
1. Preparing the environment
2. Adding a Data Source
3. Using Mapping Data Flow transformation
4. Writing to a Data Sink
5. Running the Pipeline

Task 1: Preparing the environment

1. **Turn on Data Flow Debug** Turn the **Data Flow Debug** slider located at the top of the authoring module on.

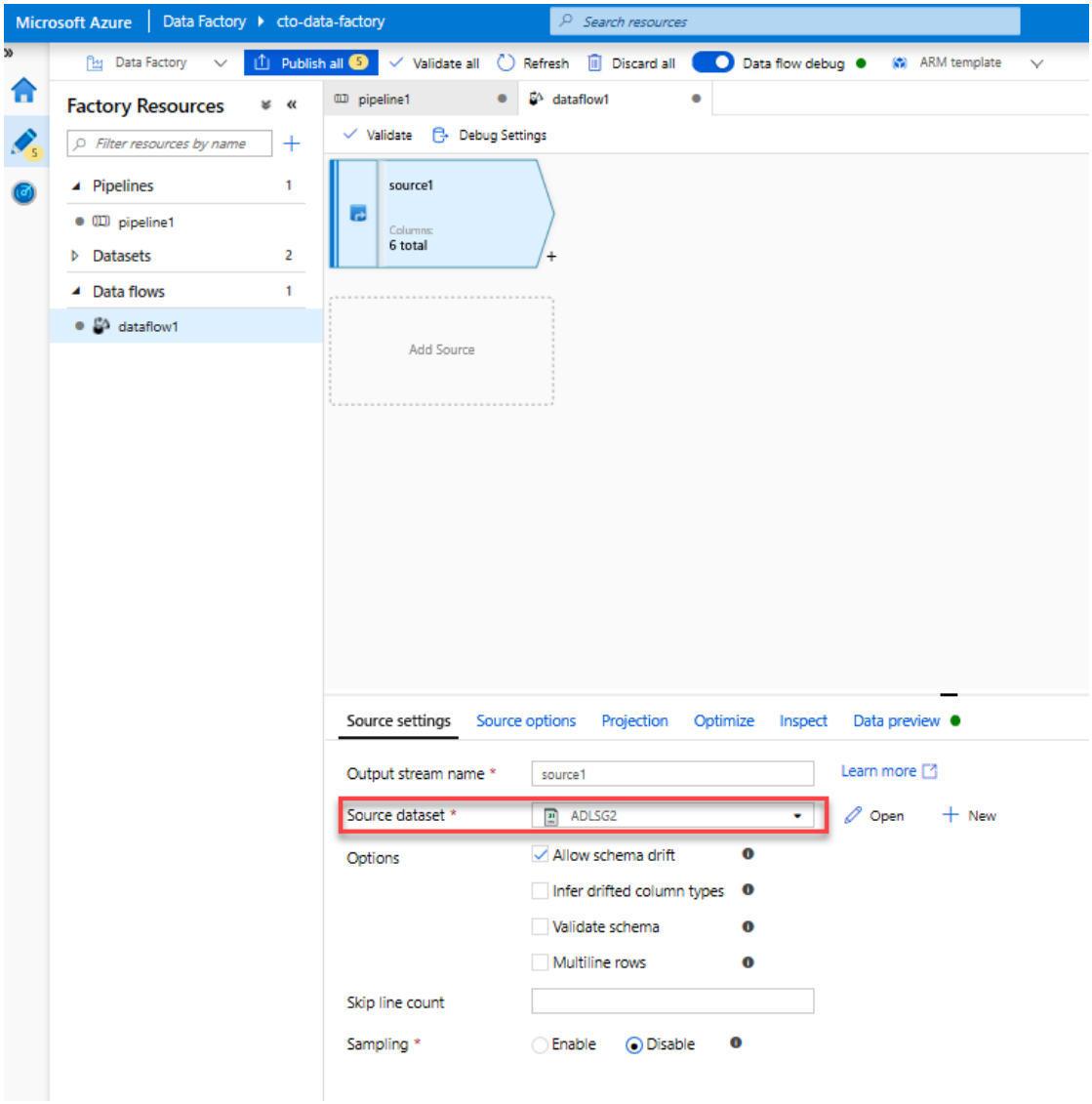
> NOTE: Data Flow clusters take 5-7 minutes to warm up.

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:



Task 2: Adding a Data Source

1. **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

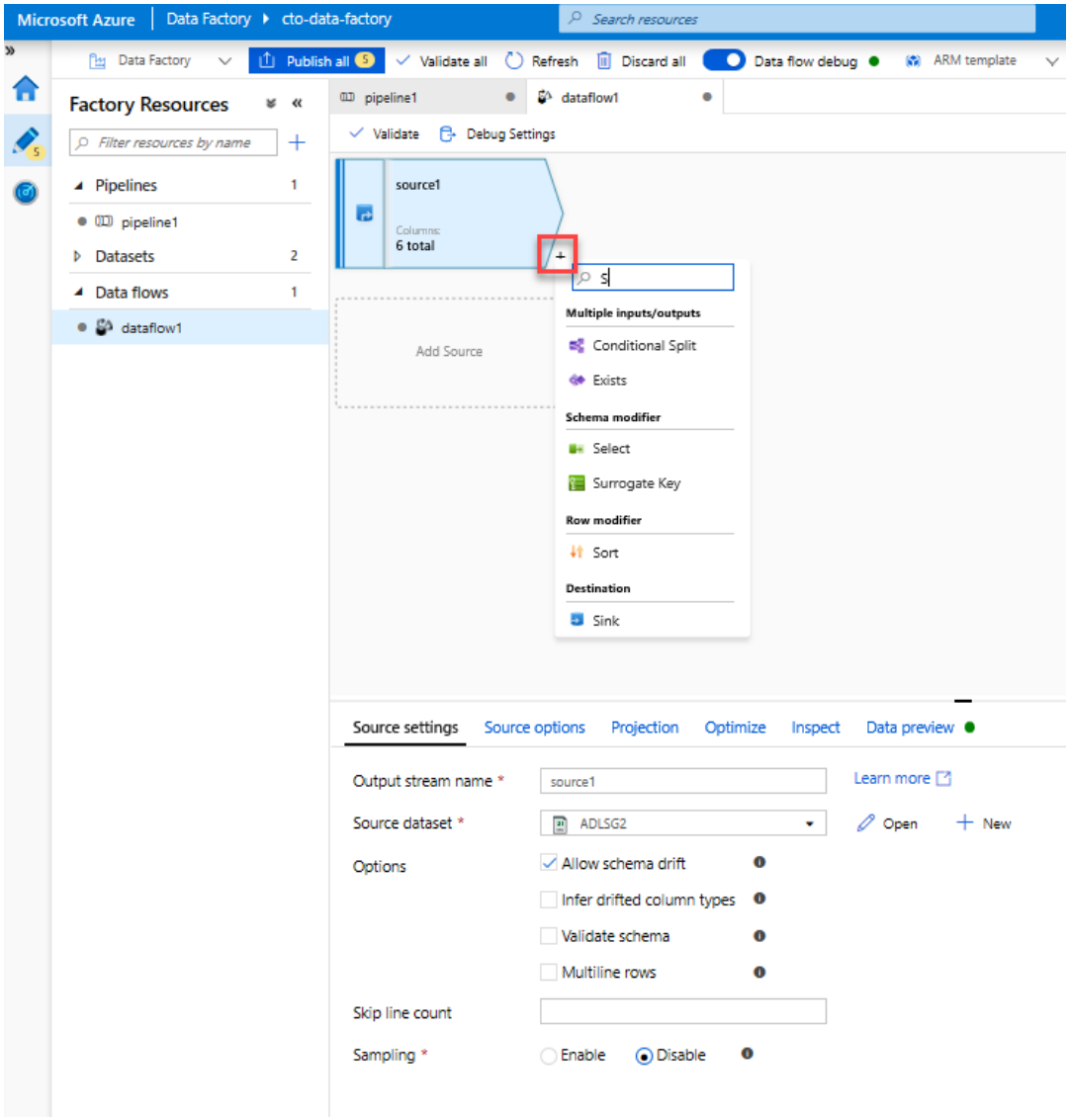


- If your dataset is pointing at a folder with other files, you may need to create another dataset or utilize parameterization to make sure only the moviesDB.csv file is read
- If you have not imported your schema in your ADLS, but have already ingested your data, go to the dataset's 'Schema' tab and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

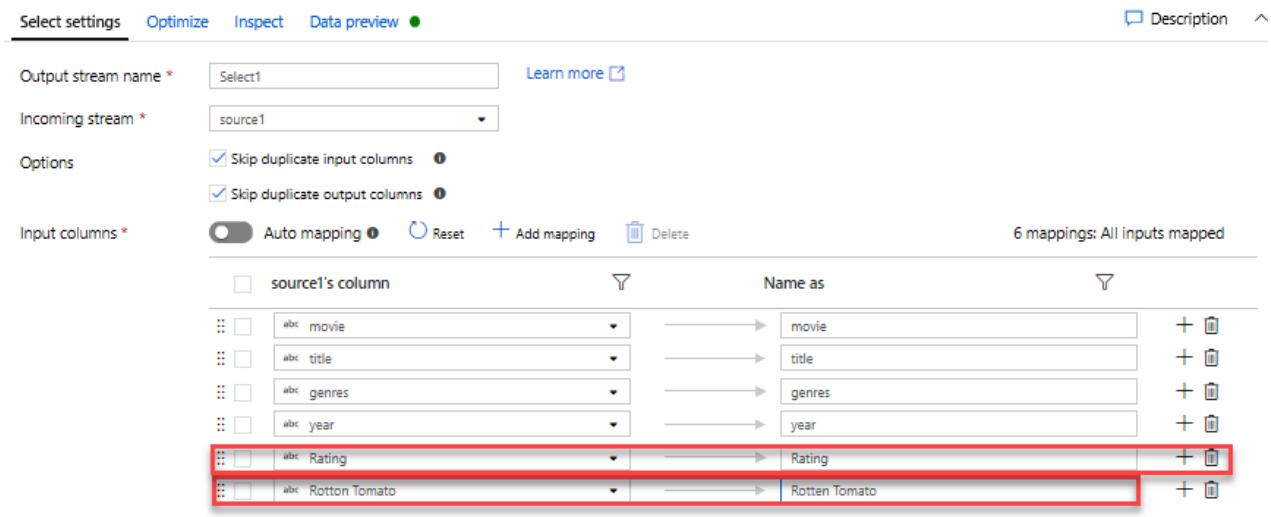
Once your debug cluster is warmed up, verify your data is loaded correctly via the Data Preview tab. Once you click the refresh button, Mapping Data Flow will show calculate a snapshot of what your data looks like when it is at each transformation.

Task 3: Using Mapping Data Flow transformation

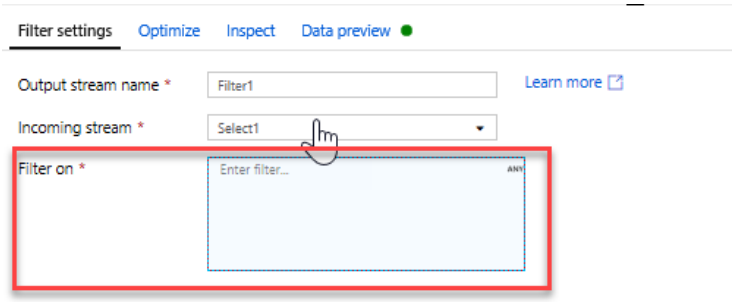
1. **Add a Select transformation to rename and drop a column** In the preview of the data, you may have noticed that the "Rotton Tomatoes" column is misspelled. To correctly name it and drop the unused Rating column, you can add a Select transformation by clicking on the + icon next to your ADLS source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



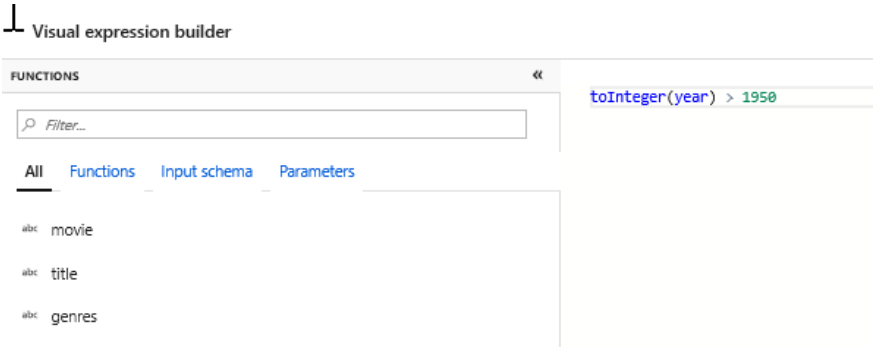
In the **Name** as field, change 'Rotton' to 'Rotten'. To drop the Rating column, hover over it and click on the trash can icon.



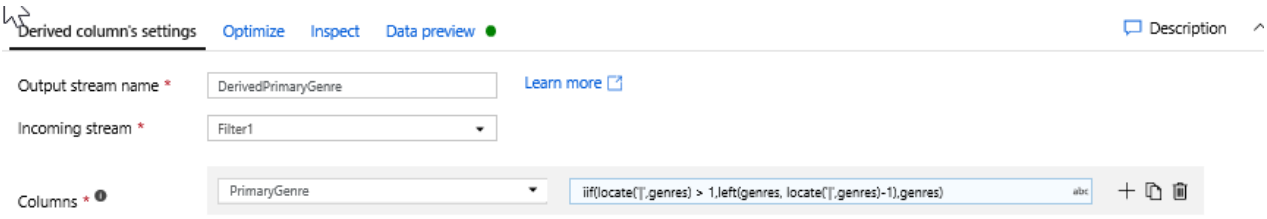
2. **Add a Filter Transformation to filter out unwanted years** Say you are only interested in movies made after 1951. You can add a Filter transformation to specify a filter condition by clicking on the **+ icon** next to your Select transformation and choosing **Filter** under Row Modifier. Click on 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.



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

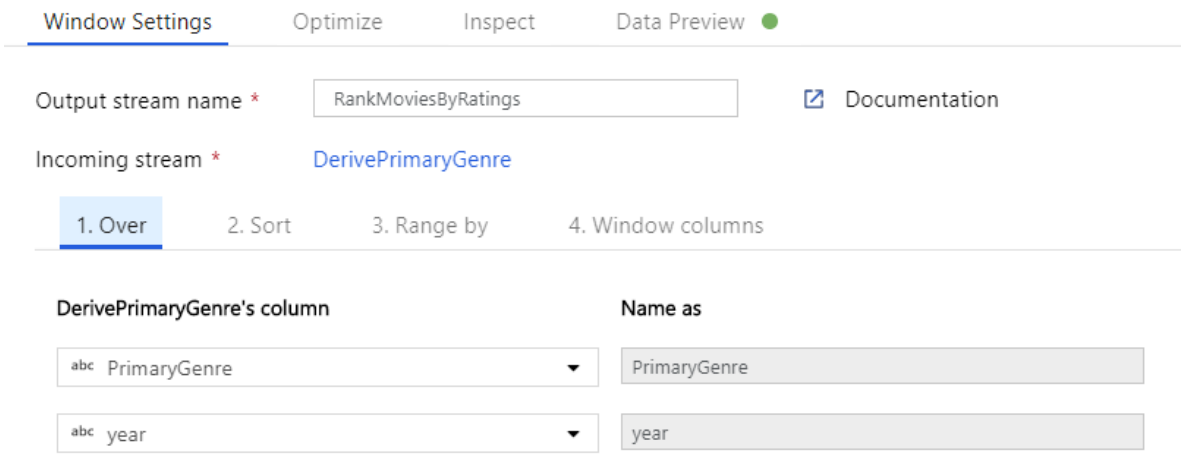


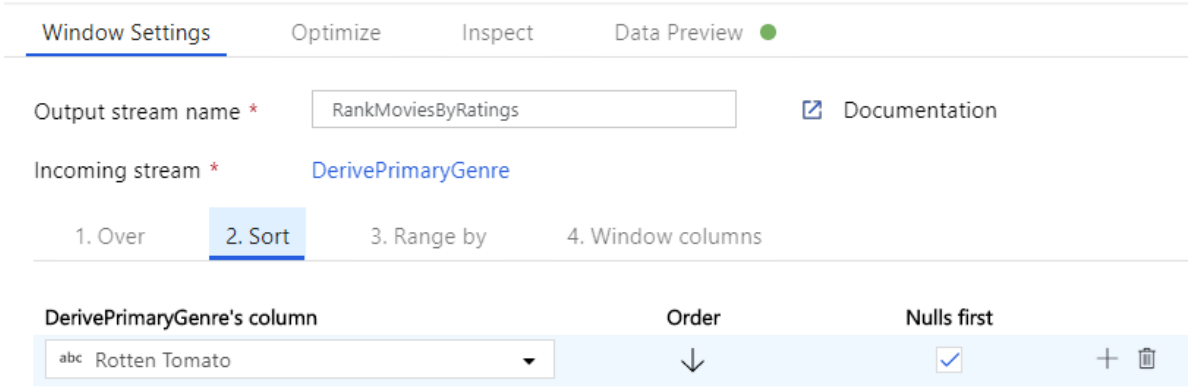
3. **Add a Derive Transformation to calculate primary genre** As you may have noticed, the genres column is a string delimited by a '|' character. If you only care about the *first* genre in each column, you can derive a new column named **PrimaryGenre** via the Derived Column transformation by clicking 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.

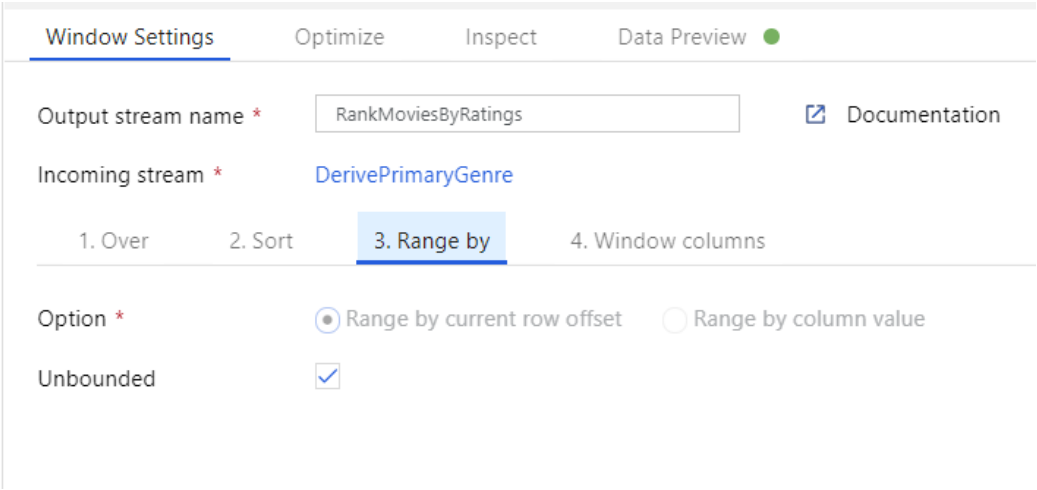


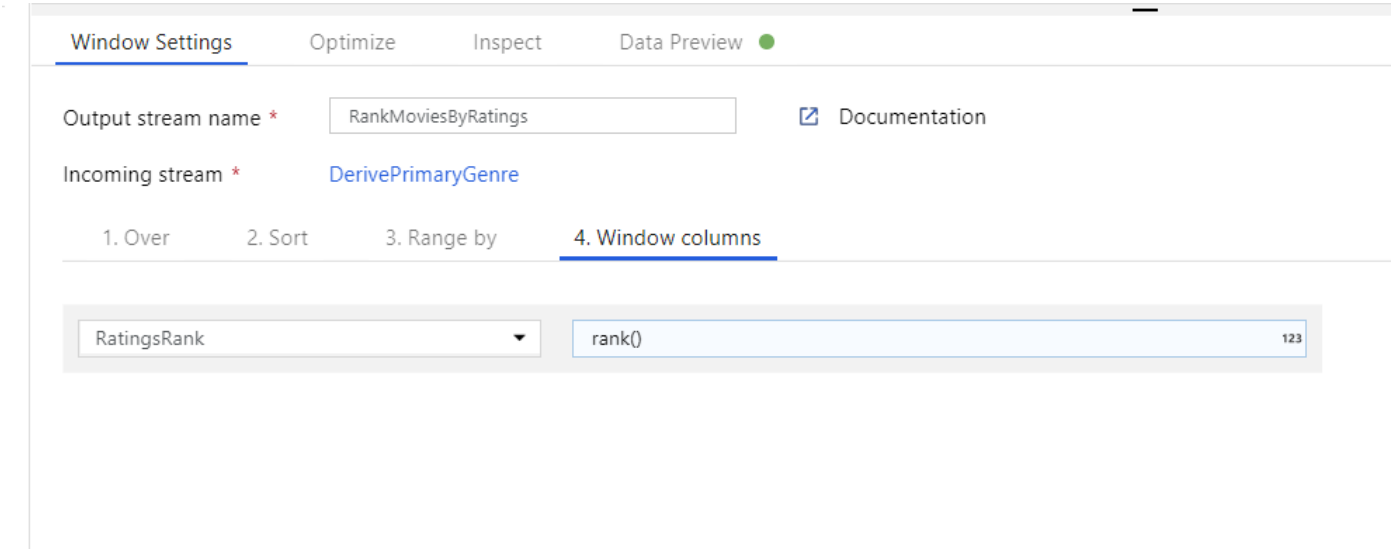
In this scenario, you are trying to extract the first genre from the genres column which is formatted as 'genre1|genre2|...|genreN'. Use the **locate** function to get the first 1-based index of the '|' in the genres string. Using the **iif** function, if this index is greater than 1, the primary genre 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, a calculate a new column called RatingsRank which is equal to the rank each movie has within its specific genre-year.

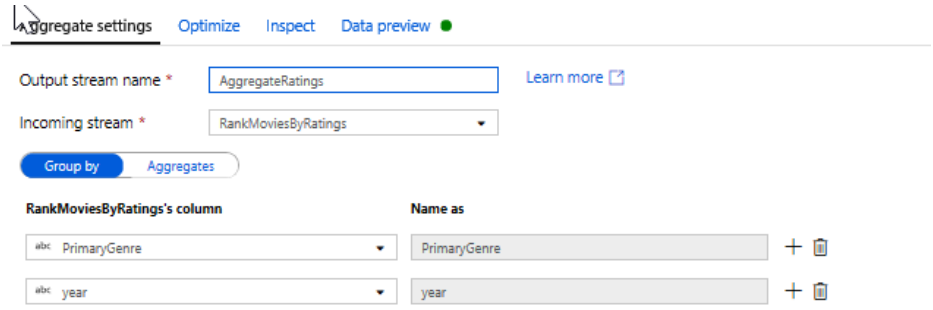

Window Settings
Optimize
Inspect
Data Preview
Output stream name *
RankMoviesByRatings
Documentation
Incoming stream *
DerivePrimaryGenre
1. Over
2. Sort
3. Range by
4. Window columns
DerivePrimaryGenre's column
Name as
PrimaryGenre
PrimaryGenre
year
year


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
Rotten Tomato

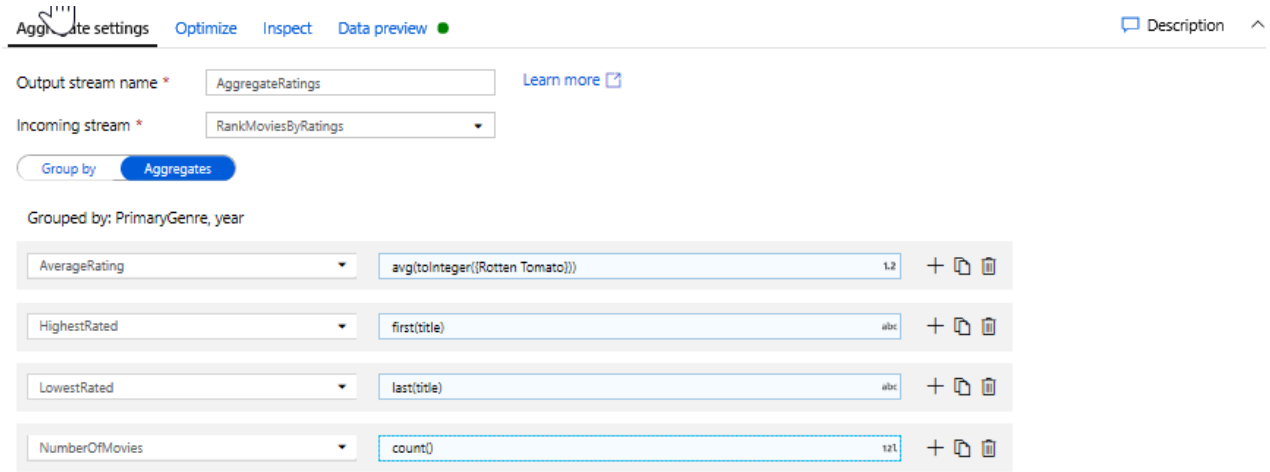

Window Settings
Optimize
Inspect
Data Preview
Output stream name *
RankMoviesByRatings
Documentation
Incoming stream *
DerivePrimaryGenre
1. Over
2. Sort
3. Range by
4. Window columns
Option *
Range by current row offset
Range by column value
Unbounded



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

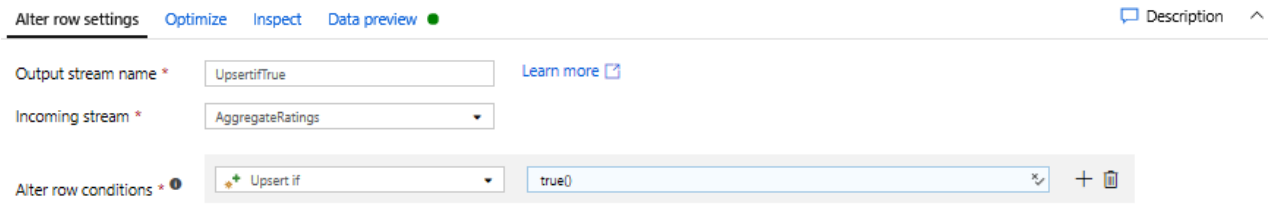


In the Aggregates tab, you can aggregations calculated over the specified group by columns. For every genre and year, lets get the average Rotten Tomatoes rating, the highest and lowest rated movie (utilizing the windowing function) and the number of movies that are in each group. Aggregation significantly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



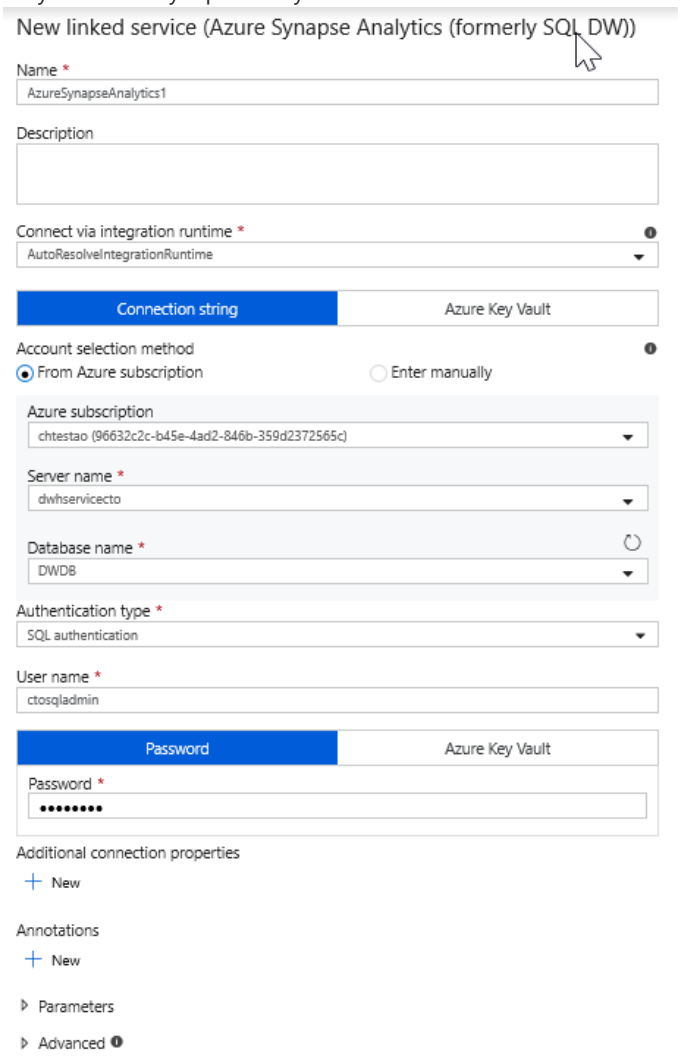
- To see how the aggregate transformation changes your data, use the Data Preview tab

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.

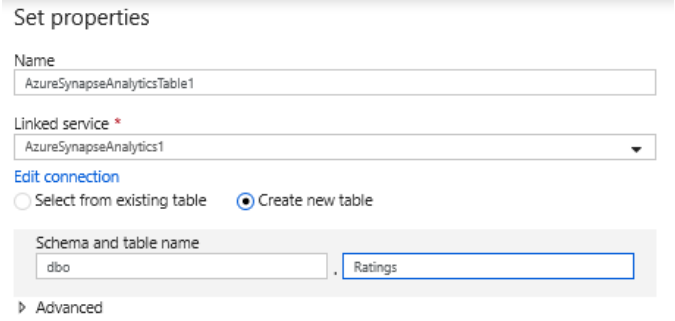


Task 4: Writing to a Data Sink

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.
   i. Add a **Sink** by clicking on the **+ icon** next to your Upsert transformation and clicking Sink under Destination.
   ii. In the Sink tab, create a new data warehouse dataset via the **+ New button**.
   iii. Select **Azure Synapse Analytics** from the tile list.
   iv. Select a new linked service and configure your Azure Synapse Analytics connection to connect to the DWDB database created in

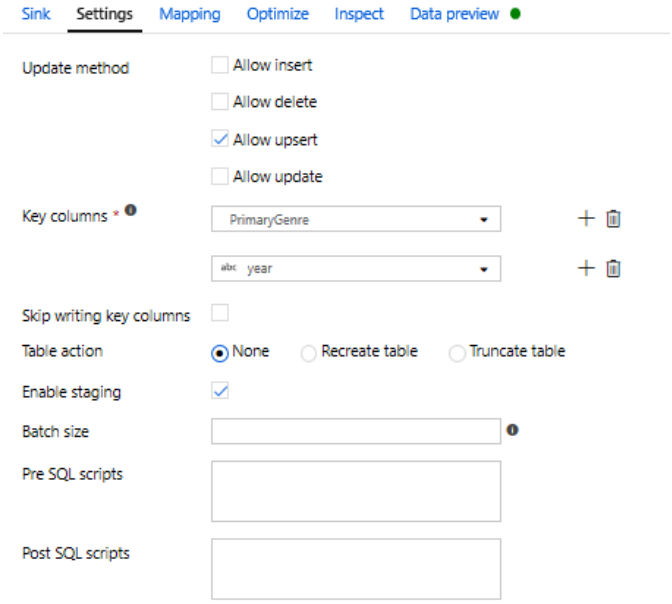


   Module 5. Click **Create** when finished.
   v. In the dataset configuration, select **Create new table** and enter in the schema of **Dbo** and the table name of **Ratings**. Click **OK**

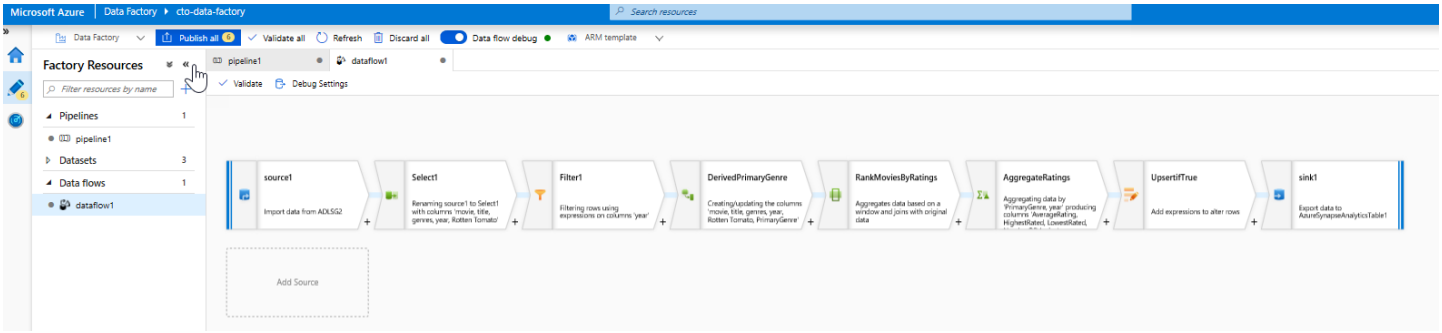


   once completed.

vi. 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. 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 should will written 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.

Exercise 4: Azure Data Factory and Databricks

Estimated Time: 15 minutes

Individual exercise

The main tasks for this exercise are as follows:

1. Generate a Databricks Access Token.

2. Generate a Databricks Notebook

3. Create Linked Services
4. Create a Pipeline that uses Databricks Notebook Activity.
5. Trigger a Pipeline Run.

Task 1: Generate a Databricks Access Token.

1. In the Azure portal, click on **Resource groups** and then click on **awrgstudxx**, and then click on **awdbwsstudxx** where xx are the initials of your name.
2. Click on **Launch Workspace**
3. Click the user **profile icon** in the upper right corner of your Databricks workspace.
4. Click **User Settings**.
5. Go to the Access Tokens tab, and click the **Generate New Token** button.
6. Enter a description in the **comment** "For ADF Integration" and set the **lifetime** period of 10 days and click on **Generate**
7. Copy the generated token and store in Notepad, and then click on **Done**.

Task 2: Generate a Databricks Notebook

1. On the left of the screen, click on the **Workspace** icon, then click on the arrow next to the word Workspace, and click on **Create** and then click on **Folder**. Name the folder **adftutorial**, and click on **Create Folder**. The adftutorial folder appears in the Workspace.
2. Click on the drop down arrow next to adftutorial, and then click **Create**, and then click **Notebook**.
3. In the Create Notebook dialog box, type the name of **mynotebook**, and ensure that the language states **Python**, and then click on **Create**. The notebook with the title of mynotebook appears/
4. In the newly created notebook "mynotebook'" add the following code:

```
# Creating widgets for leveraging parameters, and printing the parameters

dbutils.widgets.text("input", "","")
dbutils.widgets.get("input")
y = getArgument("input")
print ("Param -\'input':")
print (y)
```

> **Note** that the notebook path is **/adftutorial/mynotebook**

Task 3: Create Linked Services

1. In Microsoft Edge, click on the tab for the portal In the Azure portal, and return to Azure Data Factory.
2. In the **xx-data-factory** screen, click on **Author & Monitor**. Another tab opens up to author an Azure Data Factory solution.
3. On the left hand side of the screen, click on the **Author** icon. This opens up the Data Factory designer.
4. At the bottom of the screen, click on **Connections**, and then click on **+ New**.
5. In the **New Linked Service**, at the top of the screen, click on **Compute**, and then click on **Azure Databricks**, and then click on **Continue**.
6. In the **New Linked Service (Azure Databricks)** screen, fill in the following details and click on **Finish**
    - **Name**: xx_dbls, where xx are your initials
    - **Databricks Workspace**: awdbwsstudxx, where xx are your initials
    - **Select cluster**: use existing
    - **Domain/ Region**: should be populated
    - **Access Token**: Copy the access token from Notepad and paste into this field
    - **Choose from existing cluster**: awdbclstudxx, where xx are your initials

- Leave other options to their default settings

> **Note**: When you click on finish, you are returned to the **Author & Monitor** screen where the xx_dbls has been created, with the other linked services created in the previous exercize.

Task 5: Create a pipeline that uses Databricks Notebook Activity.

1. On the left hand side of the screen, under Factory Resources, click on the + icon, and then click on **Pipeline**. This opens up a tab with a Pipeline designer.
2. At the bottom of the pipeline designer, click on the parameters tab, and then click on **+ New**
3. Create a parameter with the Name of **name**, with a type of **string**
4. Under the **Activities** menu, expand out **Databricks**.
5. Click and drag **Notebook** onto the canvas.
6. In the properties for the **Notebook1** window at the bottom, complete the following steps:
   - Switch to the **Azure Databricks** tab.
   - Select **xx_dbls** which you created in the previous procedure.
   - Switch to the **Settings** tab, and put **/adftutorial/mynotebook** in Notebook path.
   - Expand **Base Parameters**, and then click on **+ New**
   - Create a parameter with the Name of **input**, with a value of **@pipeline().parameters.name**
7. In the **Notebook1**, click on **Validate**, next to the Save as template button. As window appears on the right of the screen that states "Your Pipeline has been validated. No errors were found." Click on the >> to close the window.
8. Click on the **Publish All** to publish the linked service and pipeline.

> **Note**: A message will appear to state that the deployment is successful.

Task 6: Trigger a Pipeline Run

1. In the **Notebook1**, click on **Add trigger**, and click on **Trigger Now** next to the Debug button.
2. The **Pipeline Run** dialog box asks for the name parameter. Use **/path/filename** as the parameter here. Click Finish. A red circle appear above the Notebook1 activity in the canvas.

Task 7: Monitor the Pipeline

1. On the left of the screen, click on the **Monitor** tab. Confirm that you see a pipeline run. It takes approximately 5-8 minutes to create a Databricks job cluster, where the notebook is executed.
2. Select **Refresh** periodically to check the status of the pipeline run.
3. To see activity runs associated with the pipeline run, select **View Activity Runs** in the **Actions** column.

Task 8: Verify the output

1. In Microsoft Edge, click on the tab **mynotebook - Databricks**
2. In the **Azure Databricks** workspace, click on **Clusters** and you can see the Job status as pending execution, running, or terminated.
3. Click on the cluster **awdbclstudxx**, and then click on the **Event Log** to view the activities.

> **Note**: You should see an Event Type of **Starting** with the time you triggered the pipeline run.