Azure Data Factory Lab

Background

A taxi commission for a major city has reached out to you for help. They are interested in understanding the predictive factors for the tips that drivers receive on different rides.

You have a ready-made machine learning model at your disposal that will allow you to submit information about new rides and predict the tip values.

However, before you can use this model to make these prediction (or "score" the data), it needs to be cleaned up and transformed.

Learning Objectives

In this lab, you will learn how to use Azure Data Factory, a no-code tool, to easily prepare your data. At the end of the lab, you will:

- be able to ingest, transform, and output data using Azure Data Factory
- create a pipeline that scores the prepared data
- understand when and how to use Azure Data Factory for other use cases

In a subsequent lab, you will also learn how to prepare the machine learning model that you are using in this lab.

But first, let's take a look at the data that you will be using.

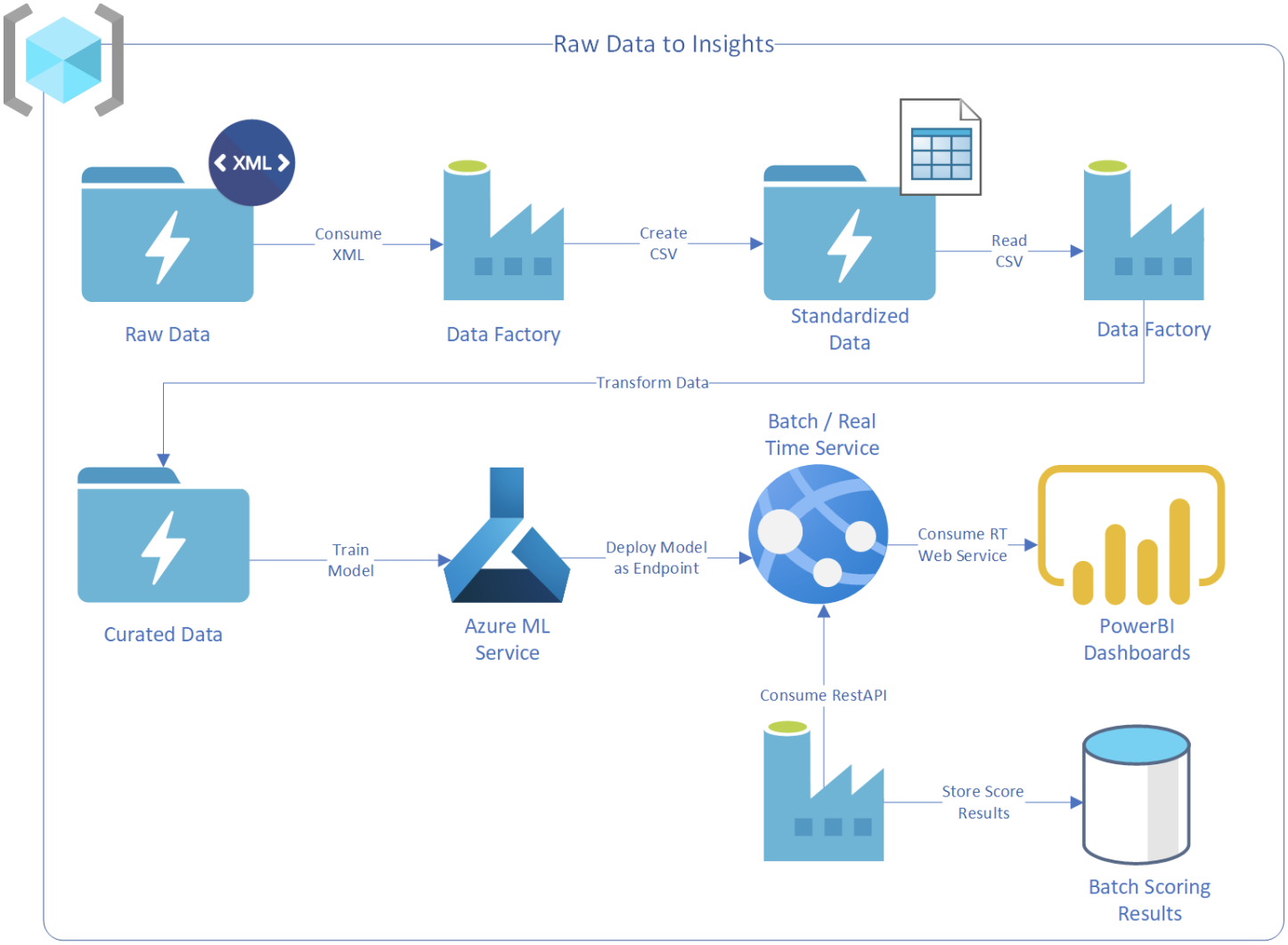
The Data

The commission has made the following data available to you.

- Taxi ride data (one XML file)
- Payment Lookup
- Zone Lookup

High-Level Architecture

The solution that you build in the course of the lab will take advantage of several Azure no-code services. You'll prepare your data and use it to train a machine learning model before creating a pipeline that can score any new data that you send it.



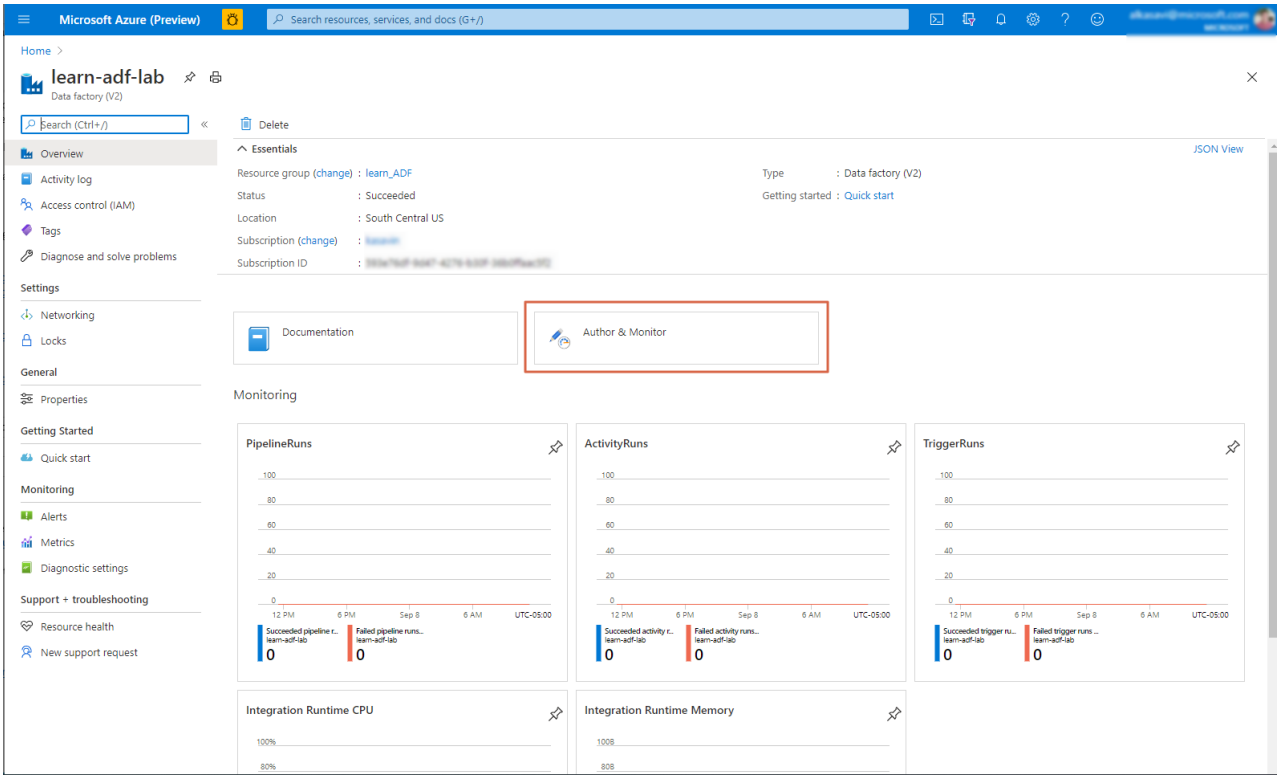
The architecture may look complex, but the lab below will take you step-by-step through its design and development.

Getting Started

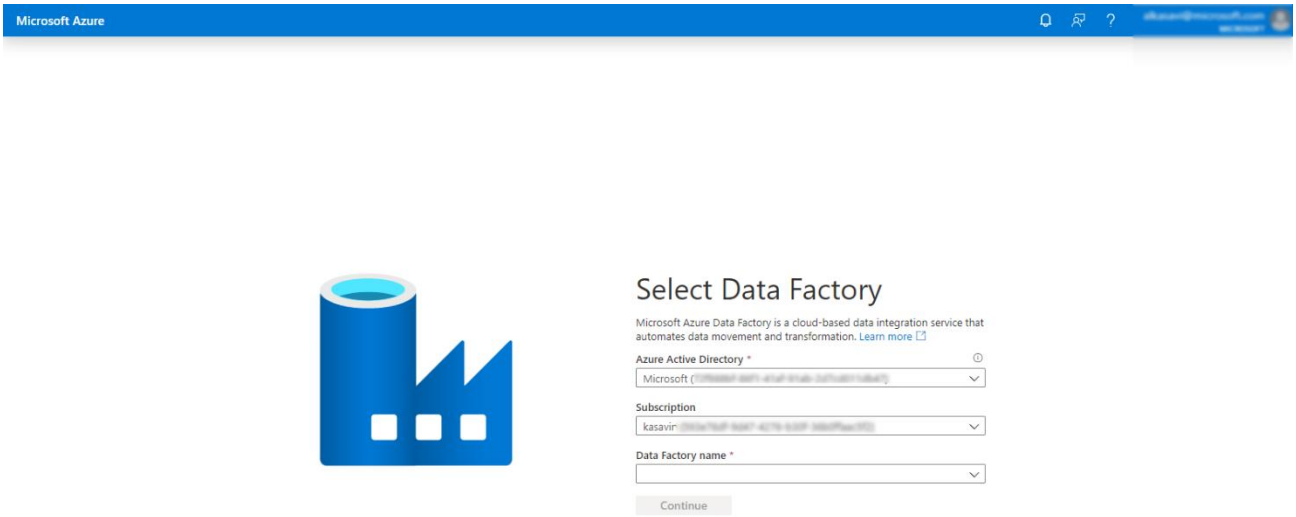
Note: In this lab, a data factory has already been created for you to use. If you are new to Azure Data Factory, you will need to create one in your own Azure subscription by following the steps documented in this article.

There are two ways of accessing the data factory.

- From the Azure Portal, select the appropriate Data Factory. Click on the Author & Monitor button pictured below:

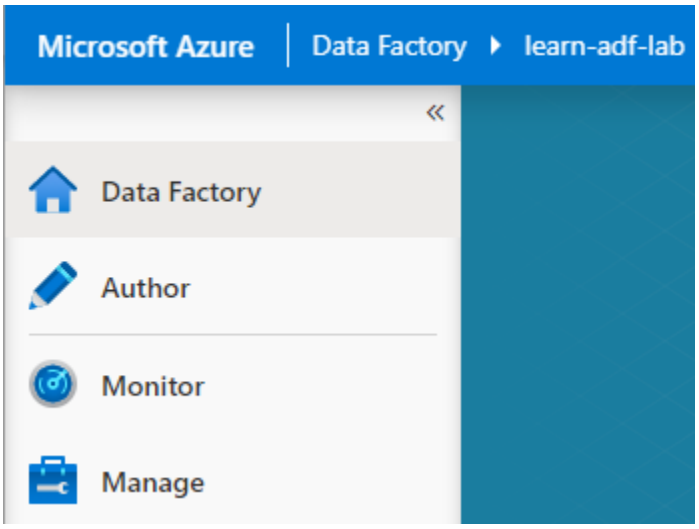


- Navigate directly to https://ms-adf.azure.com/. You may need to select your data factory using the dropdown pictured below:



The Azure Data Factory landing page provides an overview of the tool's basic capabilities, as well as links to videos and in-depth tutorials. We encourage you to review some of these tutorials as you begin to use Azure Data Factory.

Click ">>" on the left-hand sidebar, and you'll see the three main categories of actions that you can take within Azure Data Factory:



- Author
  - In this section, you will build sequential activities for your data factory to perform.
    - Pipelines
    - Datasets
    - Dataflows
- Monitor
  - In this section, you can review your data factory's performance on the activities that you established for it in the Author section.
- Manage
  - In this section, you can define connection to data stores, compute, and source control for your data factory code.

I. Creating Your First Data Factory Pipeline: Converting XML to CSV

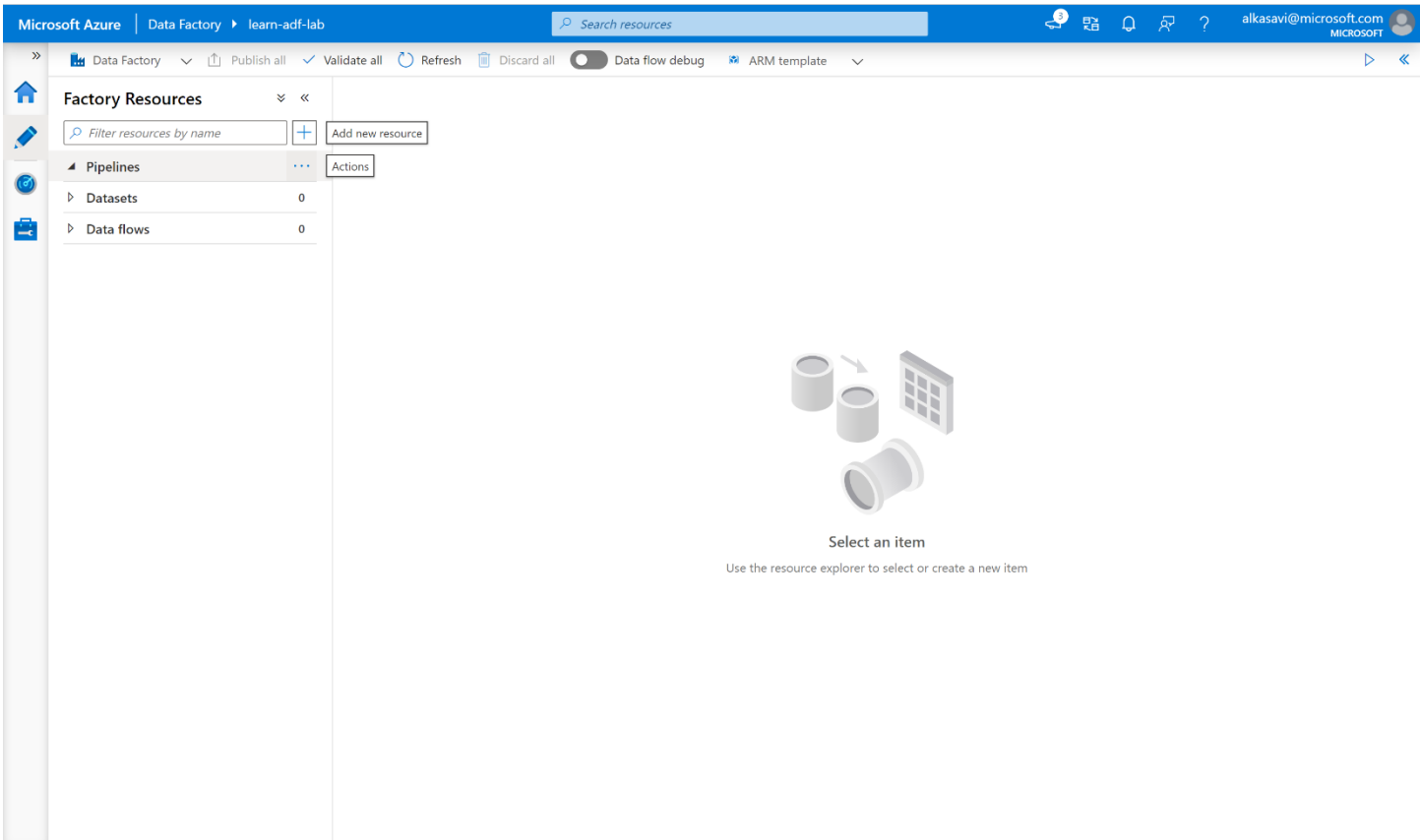
The machine learning model will only accept CSV files for scoring. As a first step, you'll need to convert the XML files into CSV files.

1. If you are still on the Linked services screen, click **Author** in the left-hand sidebar.

i. Before building your first pipeline, toggle the **Data flow debug** setting at the top of the screen.
ii. Select "dataflowruntime1" from the **Integration runtime** dropdown.

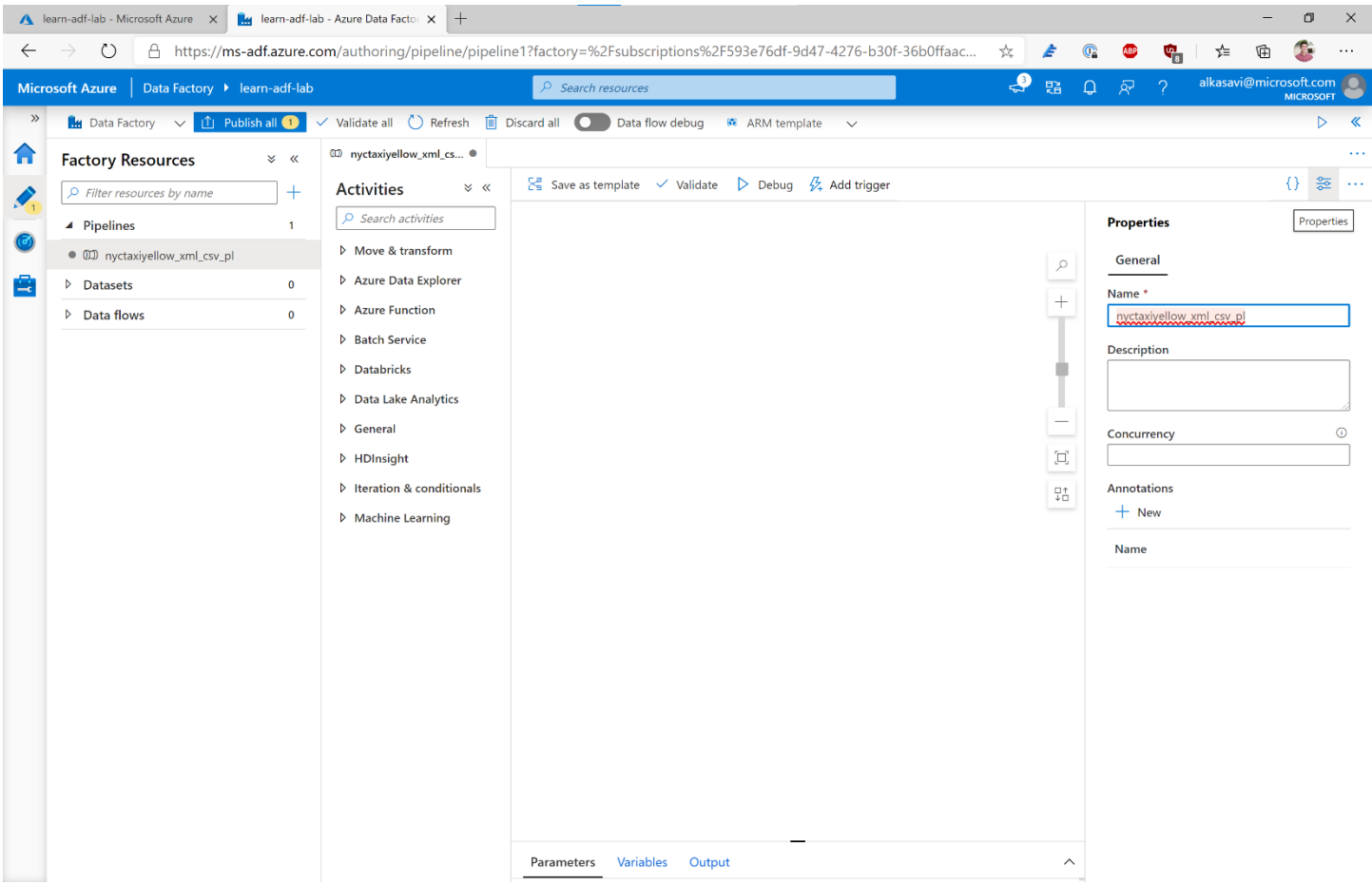
2. You can create a pipeline in one of several ways.

   i. Click the + sign to open the **Add new resource** menu and select **Pipeline**.
   ii. Click the ... ellipsis next to **Pipelines** and select **New pipeline**.



3. Either way, your first pipeline will be created, and you will automatically see the pipeline authoring canvas.

   i. At left, there will be a list of pipeline **Activities** where you will be selecting the pipeline steps.
   ii. In the middle, the drag and drop canvas allows you to add and link these activities.
   iii. On the right, the pipeline **Properties** allows you to name and describe this pipeline.

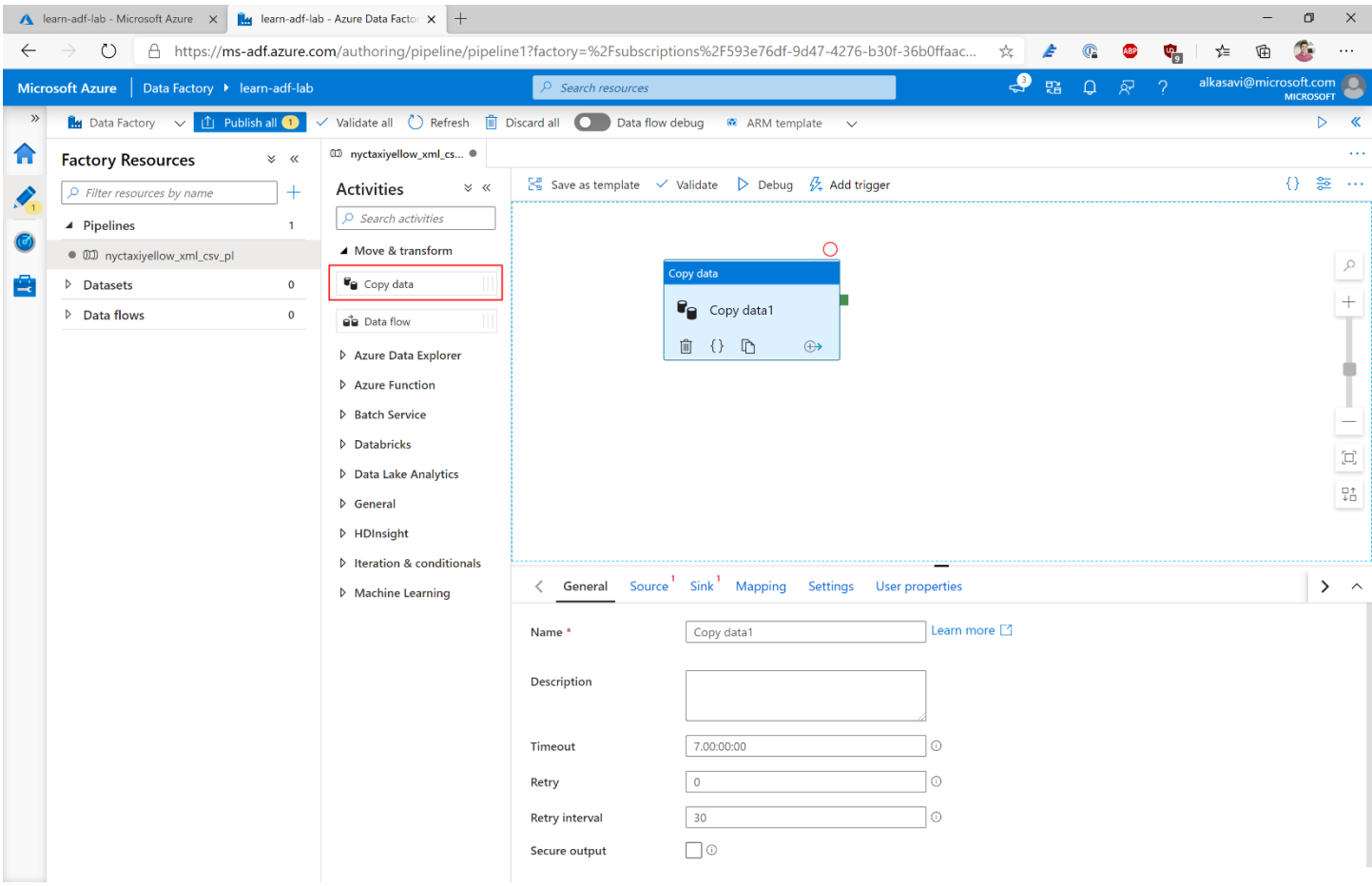


4. Add a descriptive name for this pipeline - such *as (prefix pipeline name with initials and birth year. ex. Ak_1985 for Alex King birth year 1985)* "ak1985_nyctaxiyellow_xml_csv_pl" - and close the properties by clicking the icon above the **Properties** section.

5. To add your first pipeline activity, click on the **Move & transform** category under **Activities**.

6. Drag and drop the **Copy data** activity onto the canvas, as pictured below.

7. When you drag an activity onto the canvas, a configuration panel below the canvas will automatically expand.



8. Configure your pipeline.

   i. As before, on the **General** tab, give your pipeline a descriptive name, such as "Copy convert xml to csv"

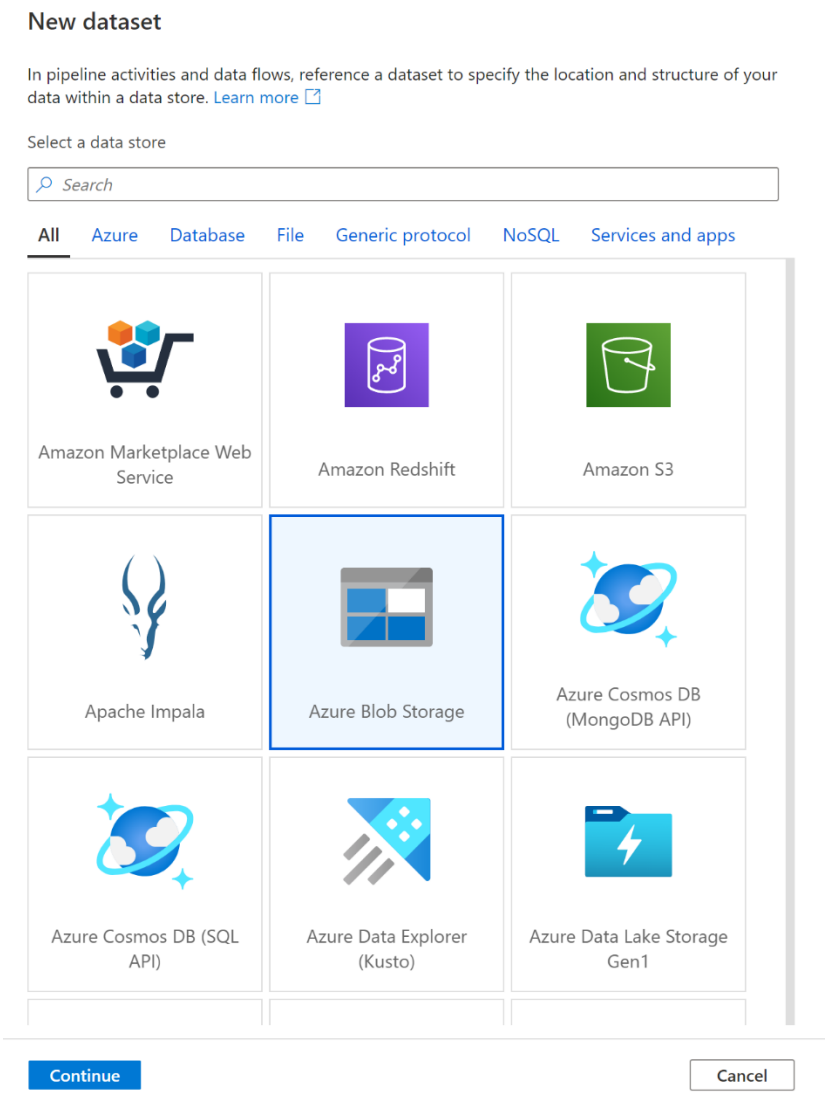
   ii. Leave the rest of the default setting on the **General** tab.

   iii. Click on the **Source** tab.

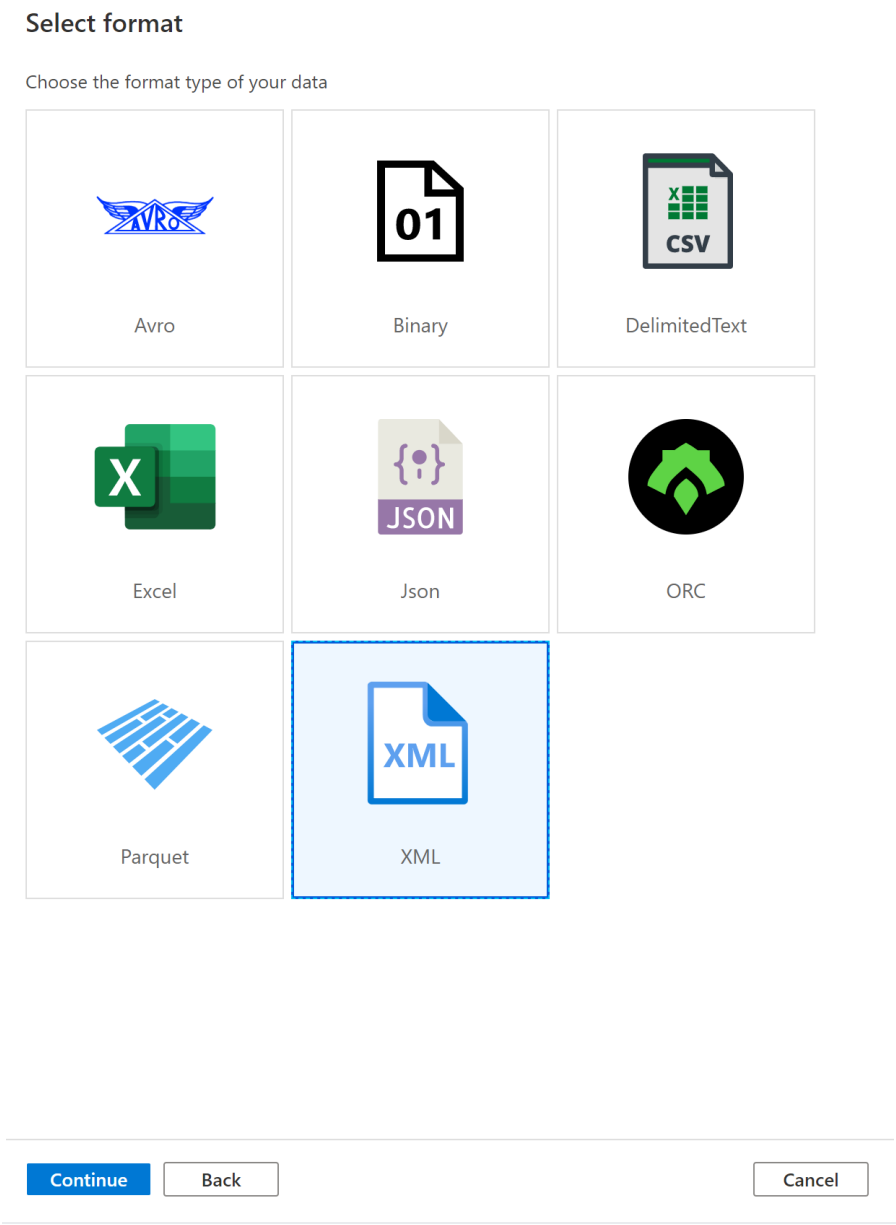
      a. Source and sink are key concepts in Azure Data Factory. They refer to the source of your data, and the destination for your data once it has been transformed.

      b. Click **New** to configure your source dataset. Connections to data sources have been configured for you; you need to do is select the appropriate *dataset* from the source.

      c. On the panel/blade that opens, select **Azure Blob Storage** and click **Continue**.



9. On the next panel/blade that opens, called **Select format**, choose **XML** and click **Continue**.



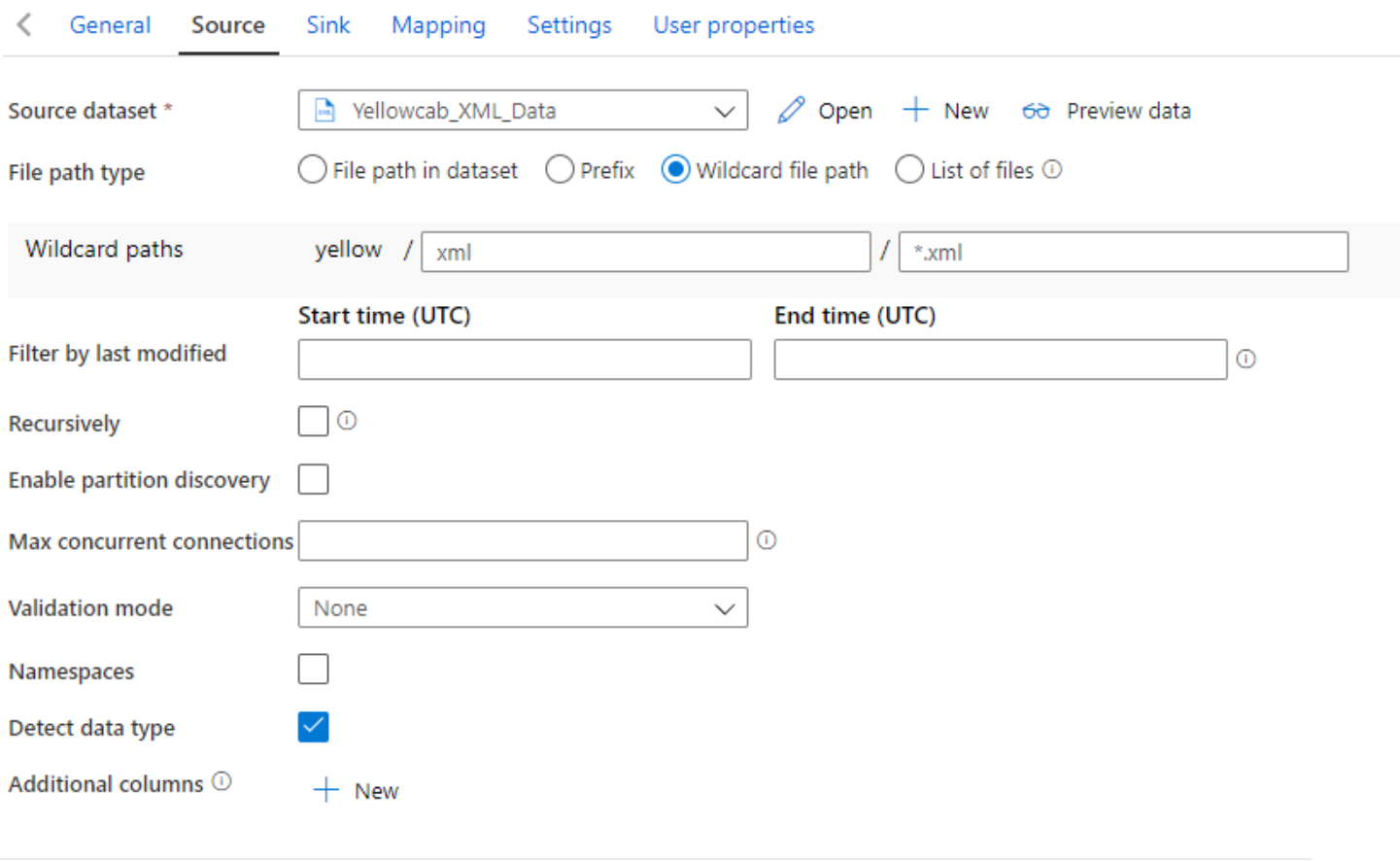
i. Give this dataset a descriptive name, such as *(prefix dataset name with initials and birth year. ex. Ak_1985 for Alex King birth year 1985)* "ak1985_Yellowcab_XML_Data"; then, in the **Linked service** dropdown, select the "AzureBlobStorage1" data source. (Once you select the data source, you may be asked to re-authenticate into Azure.)

ii. Specify "yellow" as the container and click **OK**. The data set will now appear under **Factory Resources**.

10. Because you'll be working with multiple files in the same folder, you'll need to set up a **wildcard file path**:

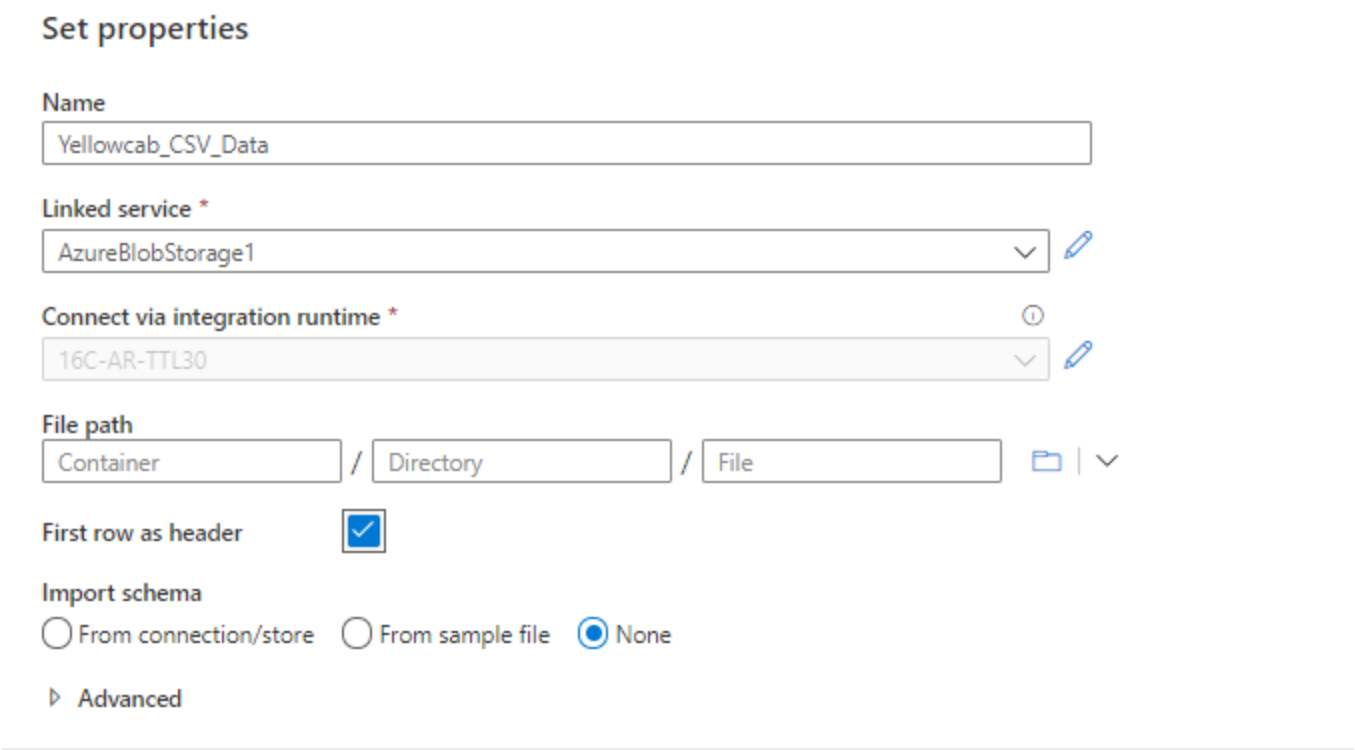
    i. Expand **Wildcard file path**.
        a. Enter "xml" as the **Wildcard folder path** and "*.xml" as the **Wildcard file name**.

    ii. Disable **Recursively** and **Namespaces**.



11. Click on the **Sink** tab. Just as you configured the source data, you will need to configure where the CSV files are written to and stored.

    i. Click **+ New**.
    ii. On the panel/blade that opens, select **Azure Blob Storage** and click **Continue**.
    iii. On the **Select format** panel/blade, select **CSV/DelimitedText** and click **Continue**.
    iv. As before, give this dataset a descriptive name, such as *(prefix dataset name with initials and birth year. ex. Ak_1985 for Alex King birth year 1985)* "ak1985_Yellowcab_CSV_Data." In the **Linked service** dropdown, select the "AzureBlobStorage1" data source you used above. (Once you select the data source, you may be asked to re-authenticate into Azure.)
        a. Select **First row as header**.
        b. Enter "yellow" as the **Container** under **File path**.
    v. Click **OK**.

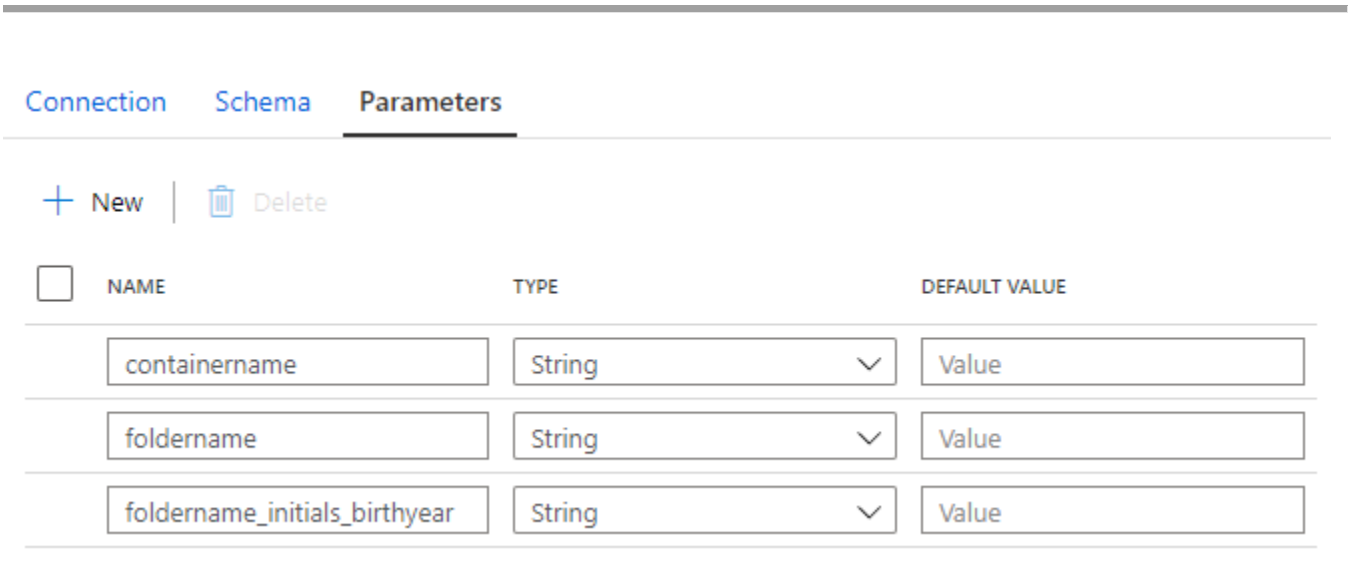


12. The **Sink dataset** dropdown will now read "ak1985_Yellowcab_CSV_Data"

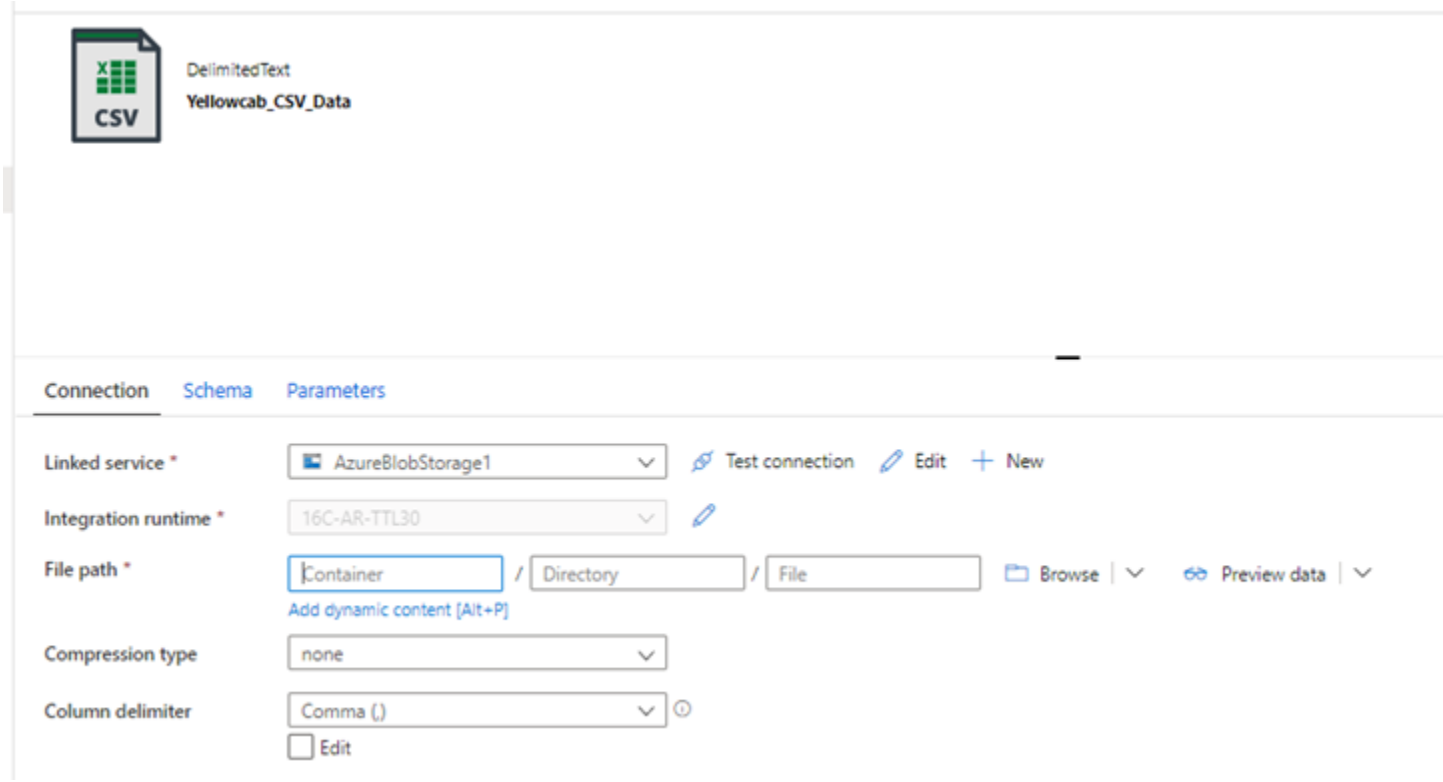
13. Click on the sink data set you just created under **Factory Resources**.

14. Click on the **Parameters** tab to configure the following three parameters.

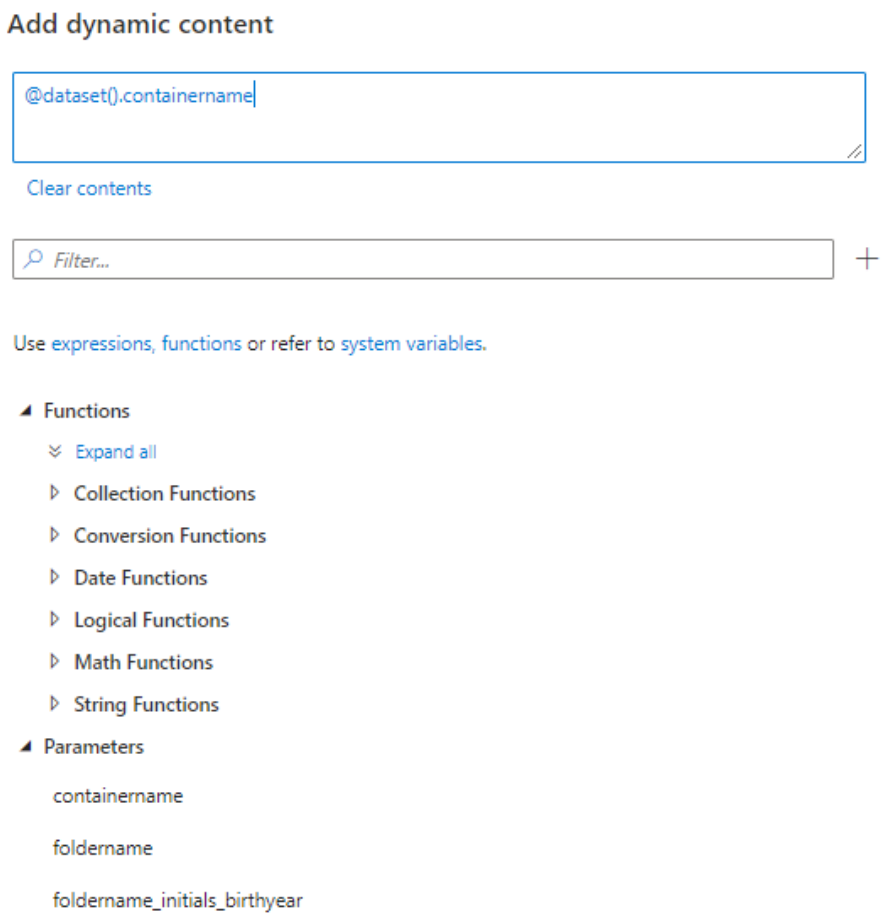
    i. containername
    ii. foldername
    iii. foldername_initials_birthyear



15. Once you've set up the **Parameters**, return to the **Connection** tab and click on the **Add dynamic content** link under the **Container** field.

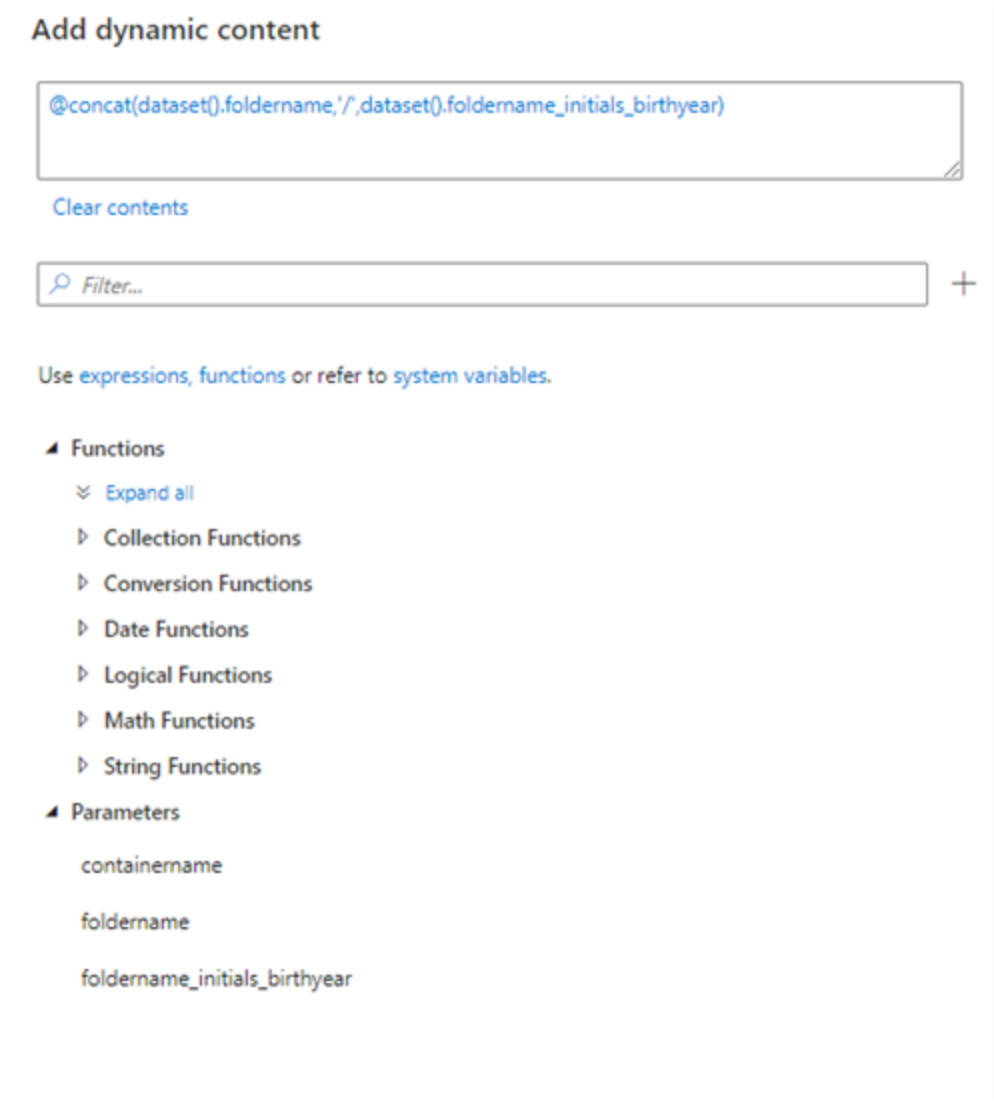


16. A new blade will open called **Add dynamic content**. Select "containername" from the list of **Parameters** at the bottom of the screen and click **OK**.

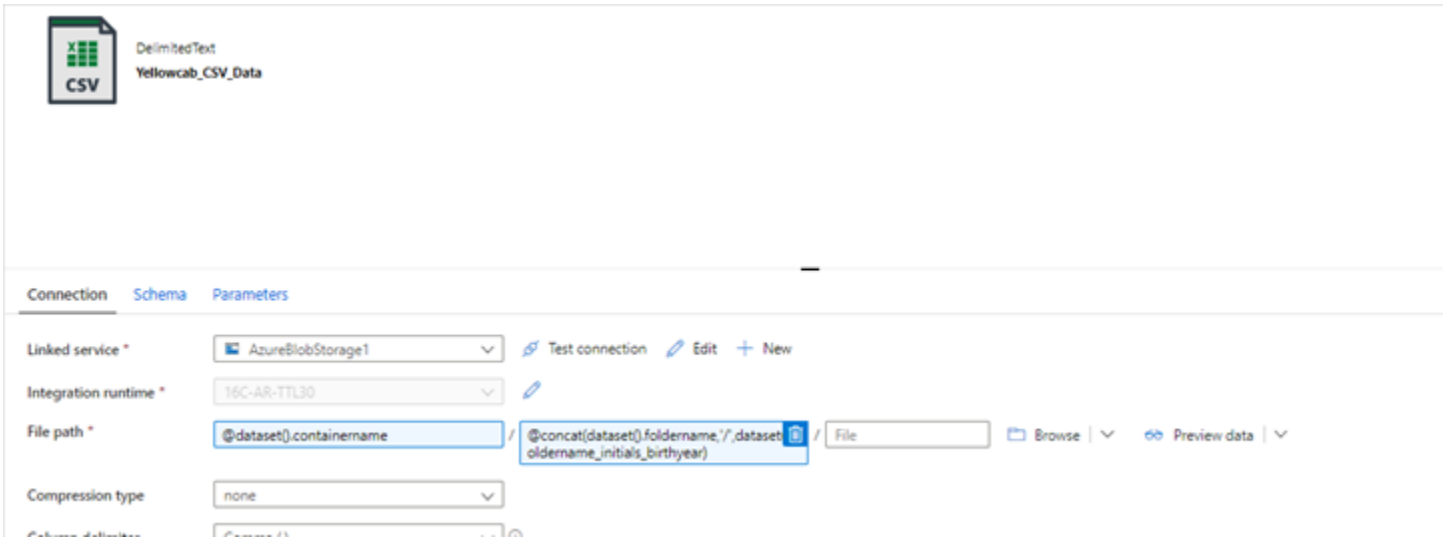


17. Repeat this process for the **Directory** field, pasting the following value into the **Add dynamic content** blade and clicking **OK**:

    - @concat(dataset().foldername,'/',dataset(foldername_initials_birthyear)

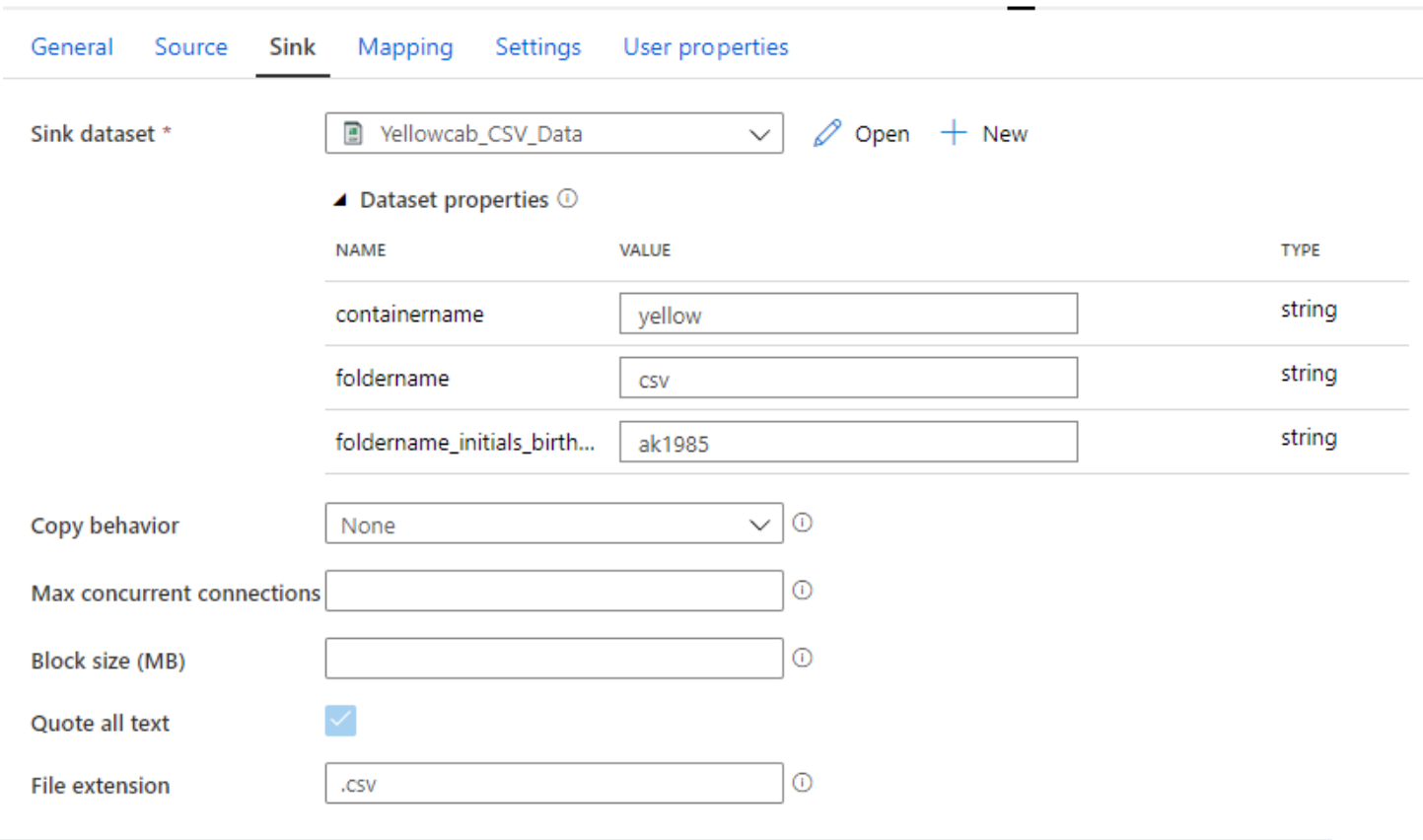


18. Your file path should look like the below once you have added both parameters:



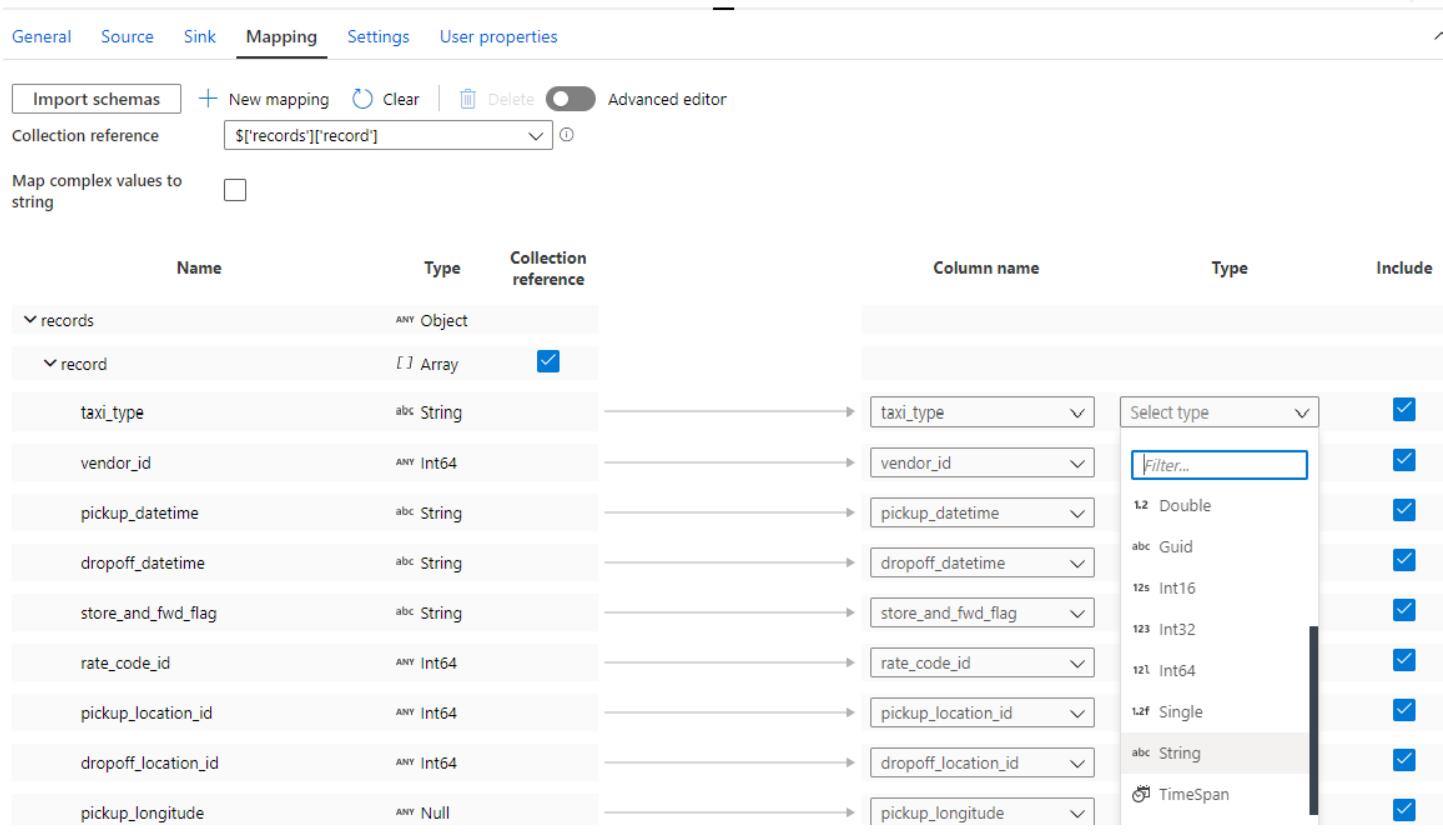
19. Return to the pipeline, either by navigating through **Factory Resources** or by using the tabs above the canvas screen. You'll now see the following fields in your **Sink** settings.

    I. Set **containername** to "yellow"
    II. Set **foldername** to "csv"
    III. Set **foldername_initials_birthyear** to your initials and birth year, such as "AB_1970"
    IV. Set **File extension** to ".csv"



20. Click the **Mapping** tab.

    i. Click **Import schemas**. This will bring in the data formatting from the XML files in order to create a mapping for the CSV columns.
    ii. Check the **Collection reference** box on the **record** row.
    iii. As you review the columns that will be created, change each data **Type** to String. (**Note:** It may strike you as odd when some of the column will clearly be numerical values or other data types. We will be working with datatype conversions later in the lab, but if you know the data you are working with, you can certainly make these designations here.)



21. You are now ready to **Validate** your first pipeline!

    i. Click the Validate button above the canvas.
    ii. Ideally, the **Pipeline validation output** will read "Your pipeline has been validated. No errors were found." If you do see an error, please reach out to one of the lab coaches for assistance.

22. After validating, click the **Publish all** button to save your changes to both the pipeline and the datasets. Click **Publish** on the **Publish all** blade that appears.

Publish all

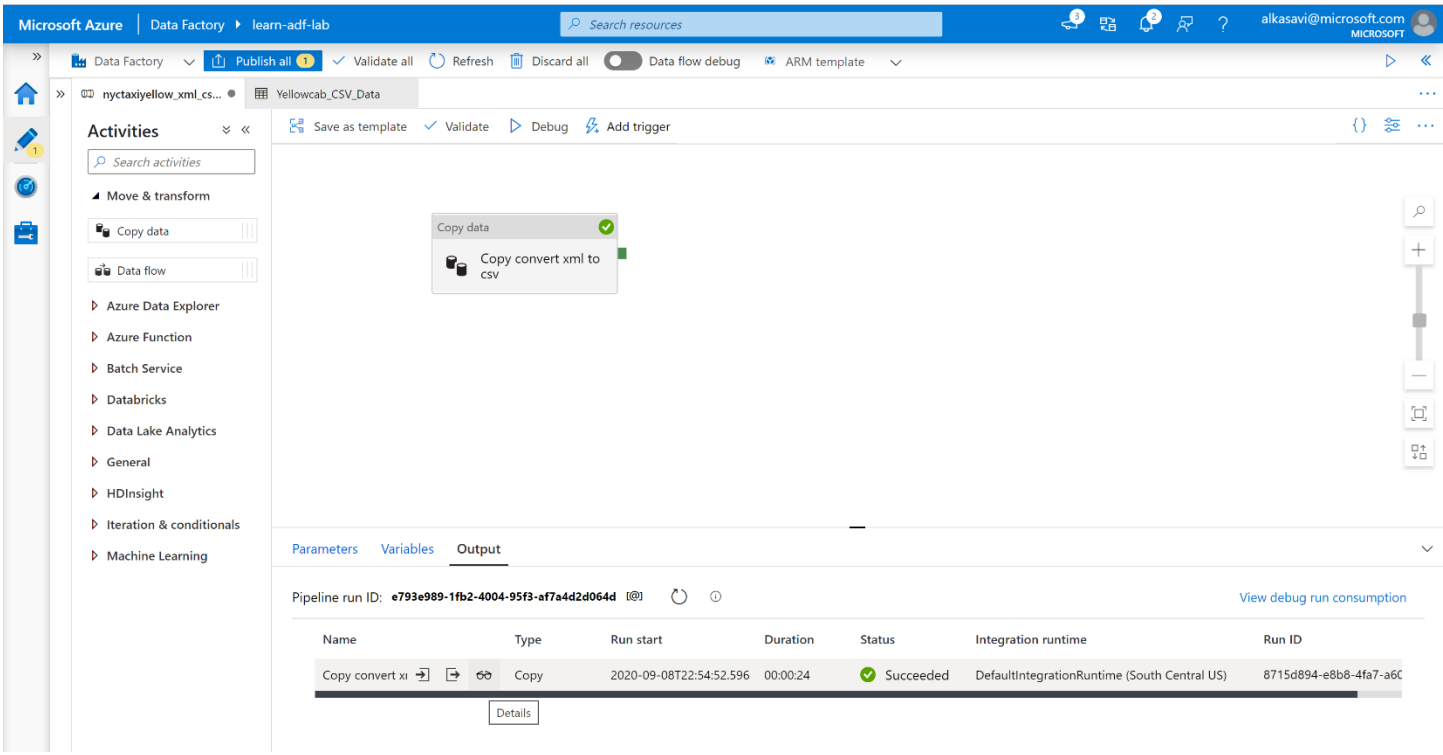
You are about to publish all pending changes to the live environment. Learn more

Pending changes (3)

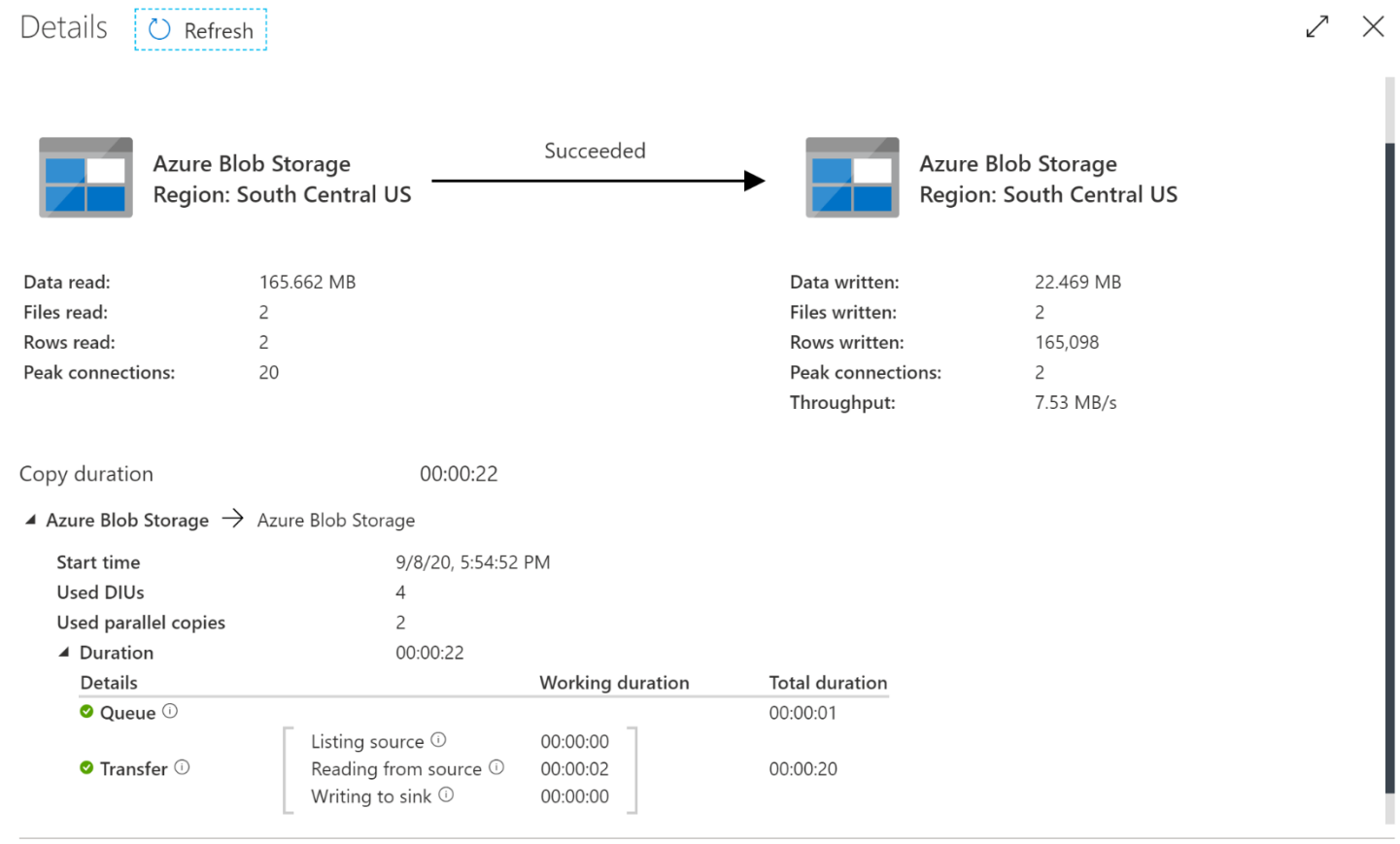
NAME	CHANGE	EXISTING
Pipelines		
nyctaxiyellow_xml_csv_pl	(New)	-
Datasets		
Yellowcab_XML_Data	(New)	-
Yellowcab_CSV_Data	(New)	-

Publish Cancel

23. Now that you've published your datasets and pipeline, you can test the pipeline in real-time. Click the **Debug** button above the canvas to begin pipeline run. A pipeline run status will appear below the canvas. Once you see a green check mark and the **Succeeded** status, you can hover over the pipeline run and click on the glasses icon for a detailed view.

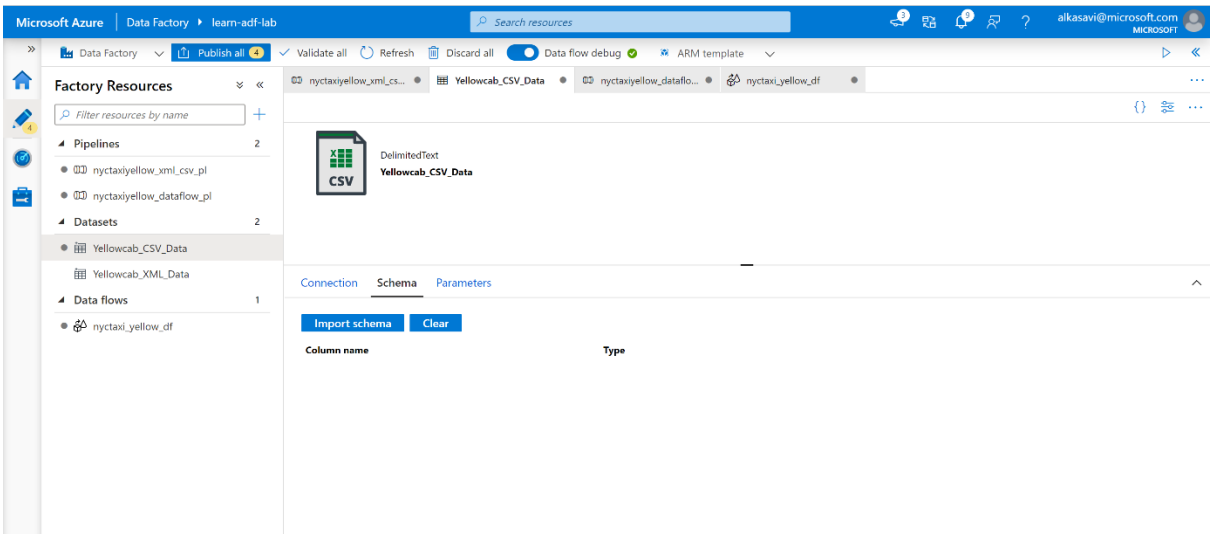


24. The **Details** popup will tell you about the pipeline run, including how much data was read, how much data was written, and the speed of the pipeline.

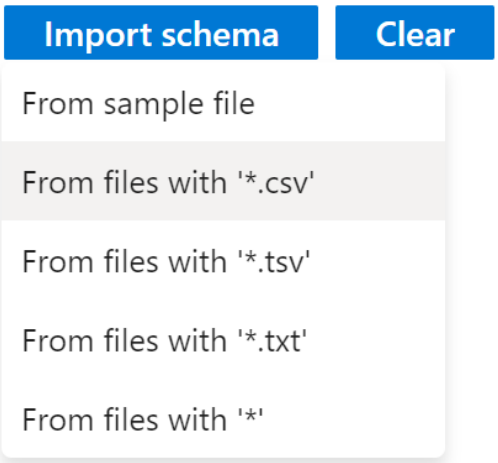


II. Import Dataset Schema

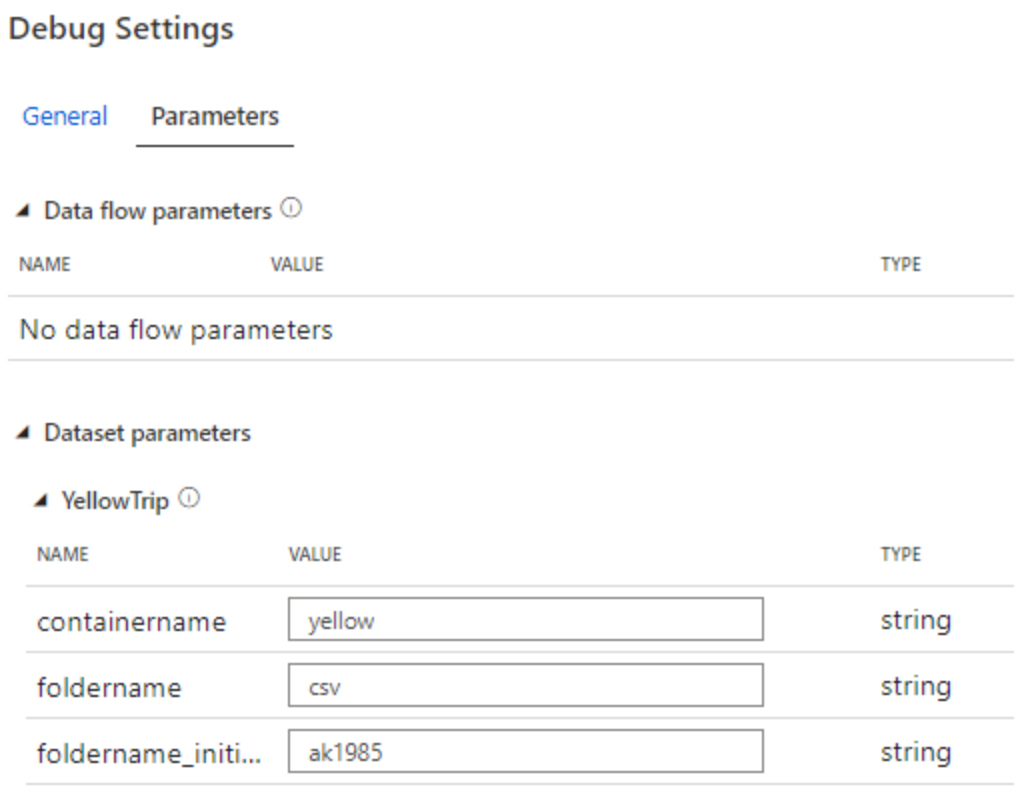
1. For the next pipeline, you'll be working with the specific data in the CSV files that you've created. To do that effectively, you will need the data schema.
2. Select the “ak1985_Yellowcab_CSV_Data” dataset under **Factory Resources ->\ Datasets**, and click the **Schema** tab.



3. Click **Import schema** and select **From files with '*.csv'**.



4. Verify the parameter settings in the panel/blade that opens, and click **OK**.



5. You should now see the columns and data types as shown below. If you recall, we designated all of the columns as strings in a prior step.

Connection | **Schema** | Parameters

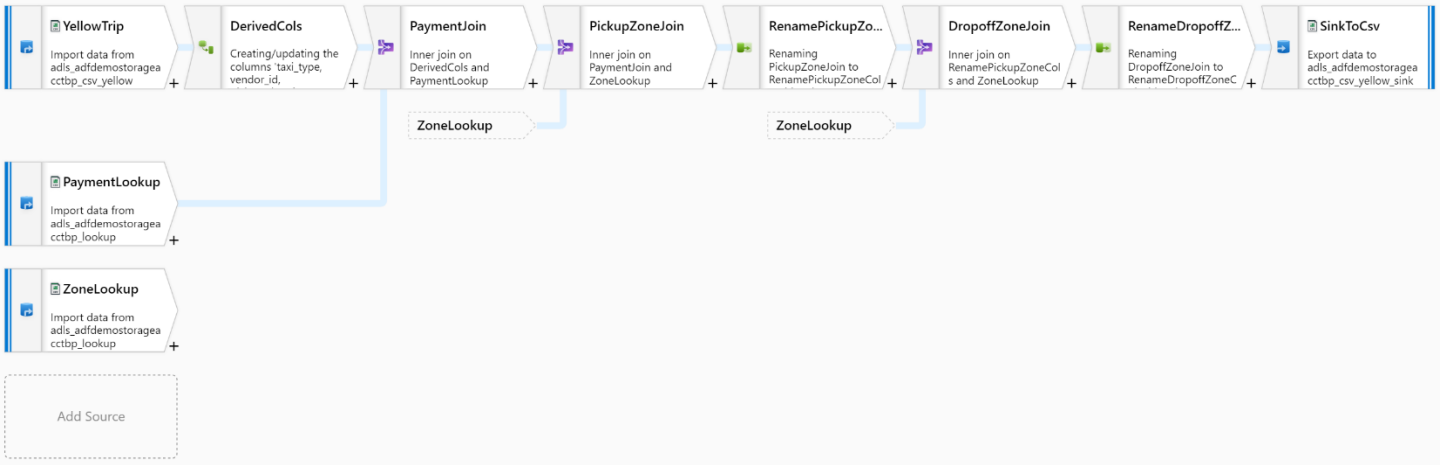
Import schema | Clear

Column name	Type
taxi_type	String
vendor_id	String
pickup_datetime	String
dropoff_datetime	String
store_and_fwd_flag	String
rate_code_id	String
pickup_location_id	String
dropoff_location_id	String
pickup_longitude	String
pickup_latitude	String
dropoff_longitude	String
dropoff_latitude	String
passenger_count	String
trip_distance	String
fare_amount	String
extra	String
mta_tax	String
tip_amount	String
tolls_amount	String

6. You can now use this schema in the data flow you'll create in the next section.

III. Working with Data Flows

Though the names might differ slightly, your final dataflow will look something like this:



Let's walk through this process one step at a time.

1. Create a new pipeline under **Factory Resources**. As a reminder, you can do this in one of two ways:

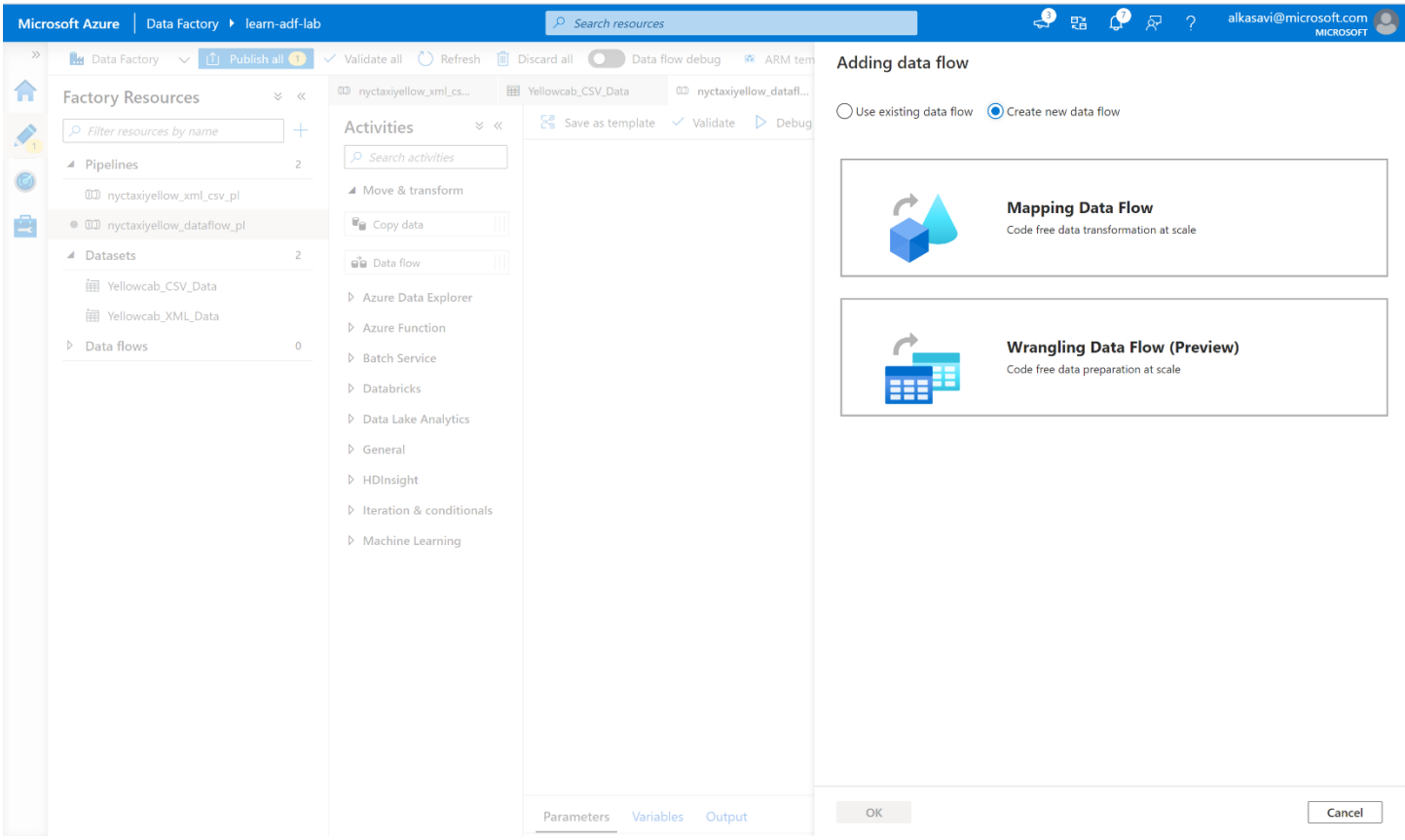
    i. Click the + sign to open the **Add new resource** menu and select **Pipeline**.
    ii. Click the ... ellipsis next to **Pipelines** and select **New pipeline**.

2. Name your new pipeline *(prefix pipeline name with initials and birth year. ex. Ak_1985 for Alex King birth year 1985)* "ak1985_nyctaxiyellow_dataflow_pl"

3. As before, click on the **Move & transform** category under **Activities**.

4. This time, drag and drop the **Data flow** activity onto the canvas.

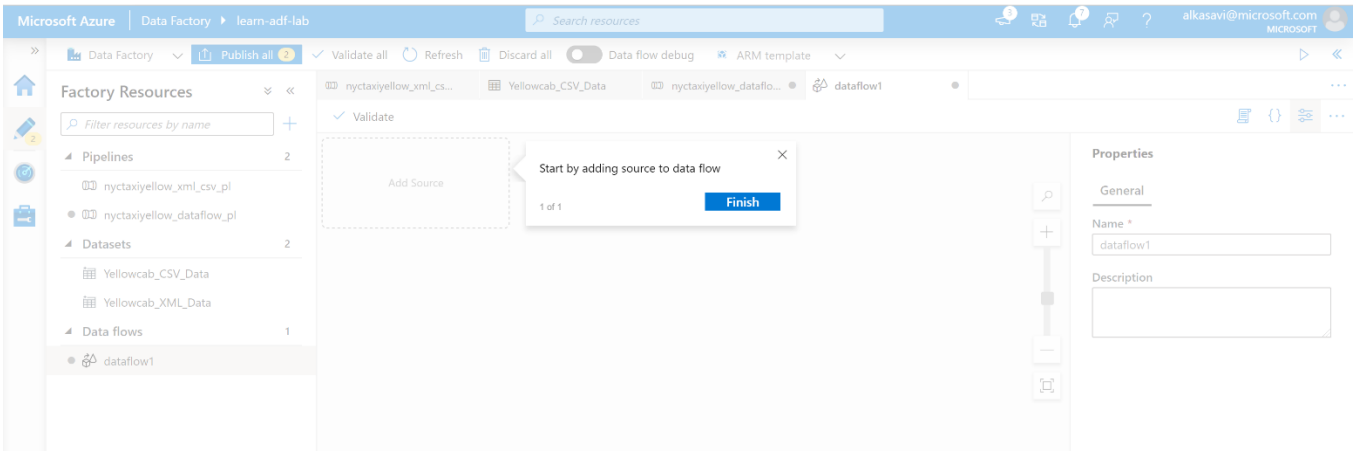
5. A panel/blade called **Adding data flow** will open automatically.



6. Select **Create a new data flow**.

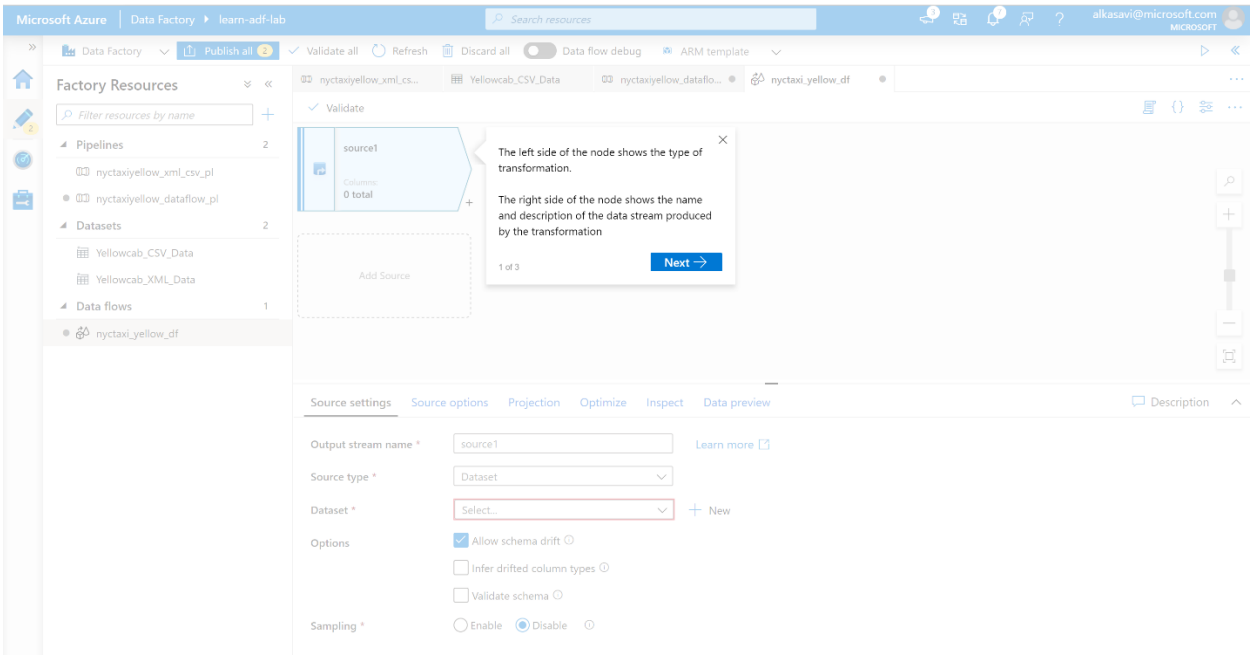
7. Select **Mapping Data Flow** and click **OK**. To learn about the two different types of data flows, visit the overviews linked below in the Resources section.

8. You will automatically be taken to the data flow canvas, with a prompt to enter your first data source.



9. Give your dataflow a descriptive name like *(prefix dataset name with initials and birth year. ex. Ak_1985 for Alex King birth year 1985)* "ak1985_nyctaxi_yellow_df" and close the **Properties** tab.

10. Click on the **Add Source** box, and you'll see a quick walkthrough that explains how data flows work.

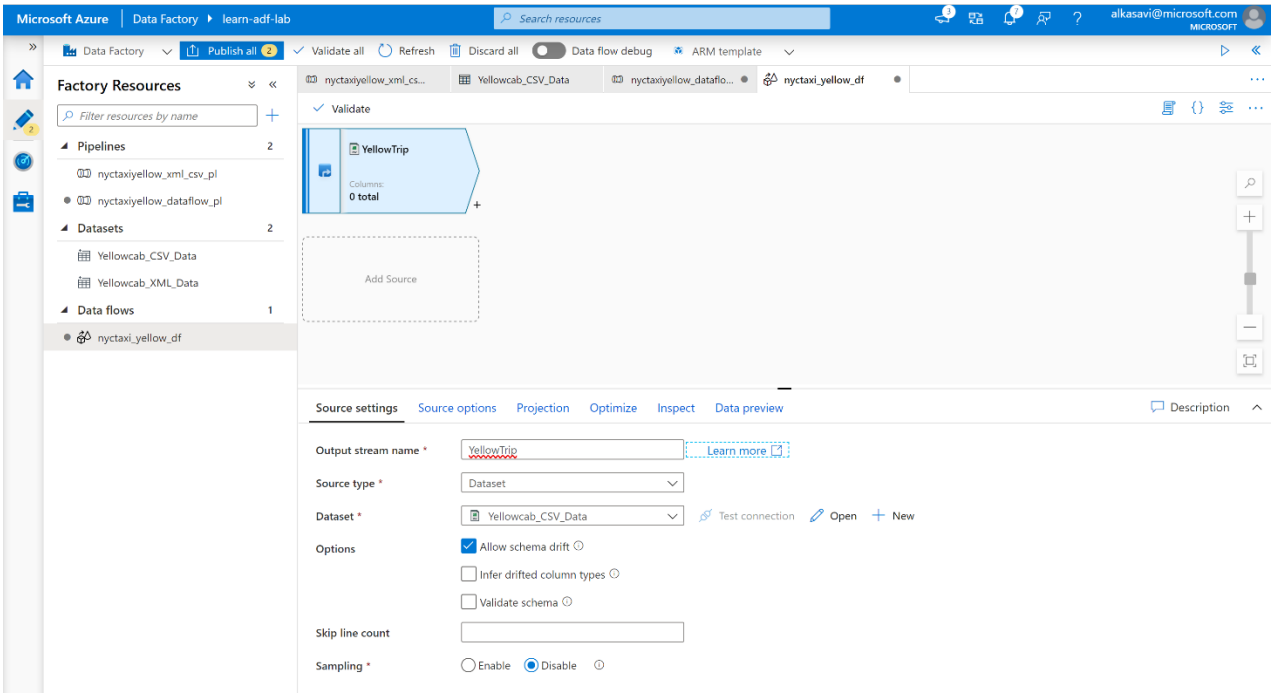


11. The three data sources you'll be using in this data flow are:

    i. the CSV output from the prior data flow
    ii. payments lookup data
    iii. zone lookup data

12. Configure your first data source to look like the screenshot below:

    i. **Output stream name:** YellowTrip
    ii. **Source type:** Dataset
    iii. **Dataset**: ak1985_Yellowcab_CSV_Data

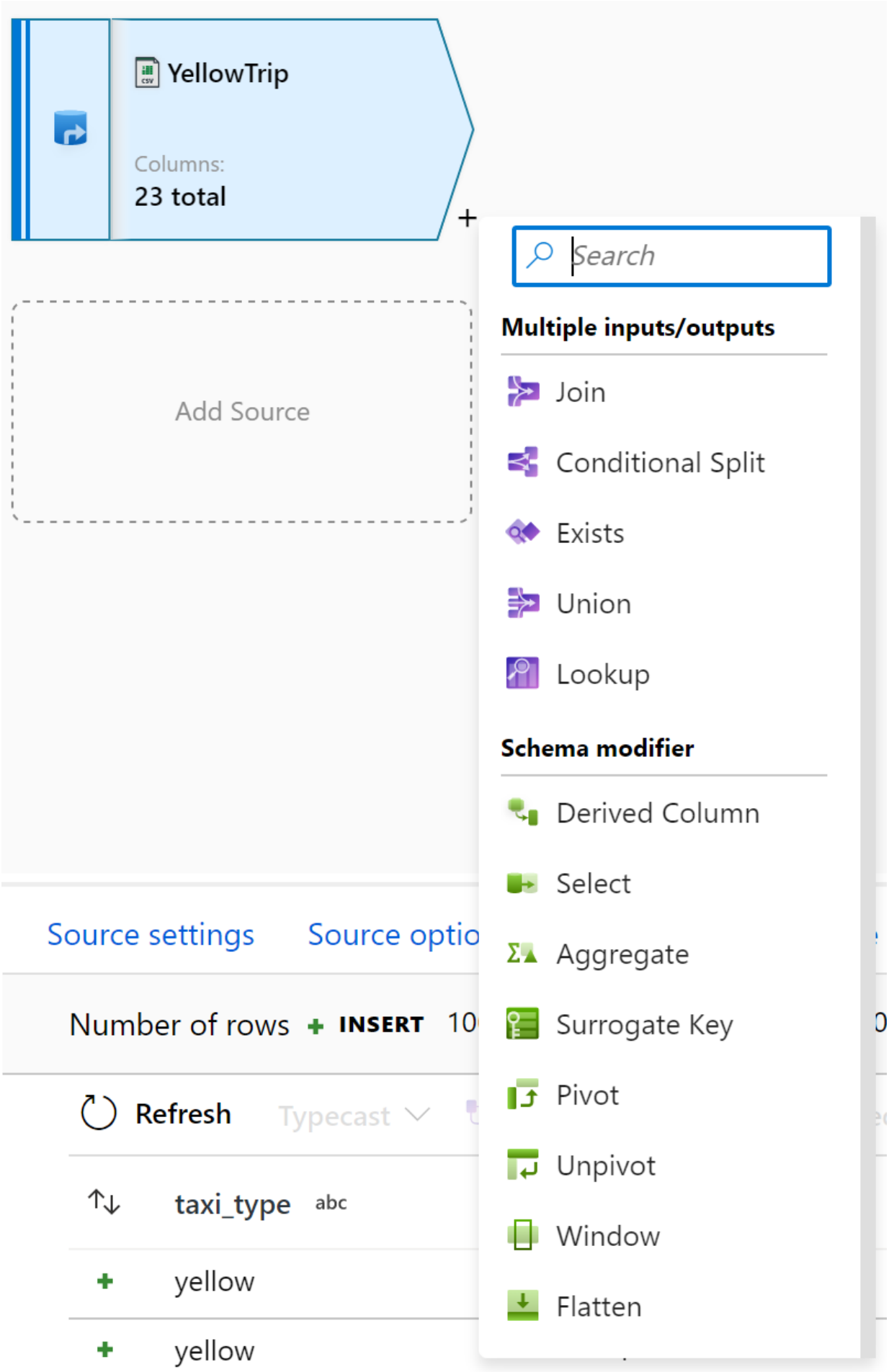


13. Click on the **Projection** tab. You will see the data schema imported in the prior section. Here, you can modify the data types. Specify the following data types:

Column name	Type
taxi_type	string
vendor_id	short
pickup_datetime	string
dropoff_datetime	string
store_and_fwd_flag	boolean
rate_code_id	short
pickup_location_id	short
dropoff_location_id	short
pickup_longitude	string
pickup_latitude	string
dropoff_longitude	string
dropoff_latitude	string
passenger_count	short
trip_distance	double
fare_amount	double
extra	double
mta_tax	double
tip_amount	double
tolls_amount	double
improvement_surcharge	double
total_amount	double
payment_type	short
RandomNum	short

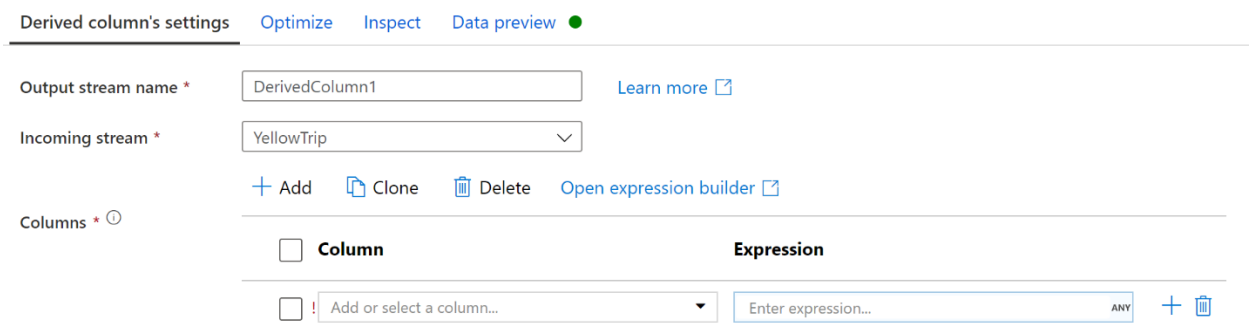
14. You can then click the **Data preview** tab to preview your data.

15. Now that you've configured the source, you're ready to work with your data. Click the small + sign at the bottom right of your datasource on the canvas. You will see a list of transformation options.


YellowTrip
Columns:
23 total
Add Source
Search
Multiple inputs/outputs
Join
Conditional Split
Exists
Union
Lookup
Schema modifier
Derived Column
Select
Aggregate
Surrogate Key
Pivot
Unpivot
Window
Flatten
Source settings
Source optio
Number of rows
INSERT
Refresh
Typecast
taxi_type
abc
yellow
yellow

16. Select **Derived Column**. Each transformation will have its own settings and configuration options.

    i. Give your **Output stream name** a name, such as "DerivedColumns."
    ii. The **Incoming stream** will auto-populate with the name of the source you specified above.

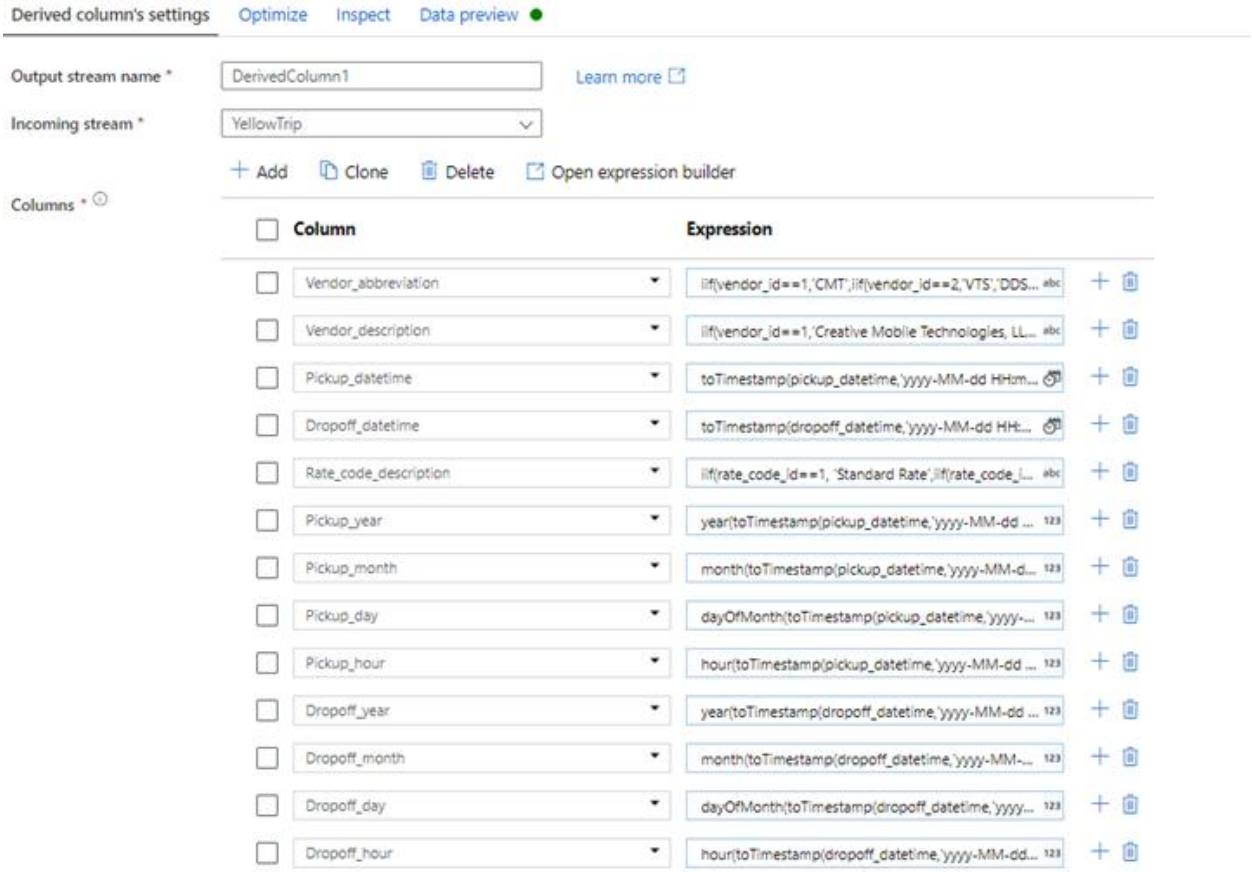


17. Under **Columns**, add the following columns and expressions, clicking **+ Add column** after each one.

Column	Expression
Vendor_abbreviation	iif(vendor_id==1,'CMT',iif(vendor_id==2,'VTS','DDS'))
Vendor_description	iif(vendor_id==1,'Creative Mobile Technologies, LLC',iif(vendor_id==2,'Verifone Inc.','Digital Dispatch Systems'))
Pickup_datetime	toTimestamp(pickup_datetime,'yyyy-MM-dd HH:mm:ss','EST')
Dropoff_datetime	toTimestamp(dropoff_datetime,'yyyy-MM-dd HH:mm:ss','EST')
Rate_code_description	iif(rate_code_id==1, 'Standard Rate',iif(rate_code_id==2,'JFK',iif(rate_code_id==3,'Newark',iif(rate_code_id==4,'Nassau or Westchester',iif(rate_code_id==5,'Negotiated fare',iif(rate_code_id==6,'Group ride','Unknown'))))))
Pickup_year	year(toTimestamp(pickup_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Pickup_month	month(toTimestamp(pickup_datetime,'yyyy-MM-dd HH:mm:ss','EST'))

Column	Expression
Pickup_day	dayOfMonth(toTimestamp(pickup_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Pickup_hour	hour(toTimestamp(pickup_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Dropoff_year	year(toTimestamp(dropoff_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Dropoff_month	month(toTimestamp(dropoff_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Dropoff_day	dayOfMonth(toTimestamp(dropoff_datetime,'yyyy-MM-dd HH:mm:ss','EST'))
Dropoff_hour	hour(toTimestamp(dropoff_datetime,'yyyy-MM-dd HH:mm:ss','EST'))

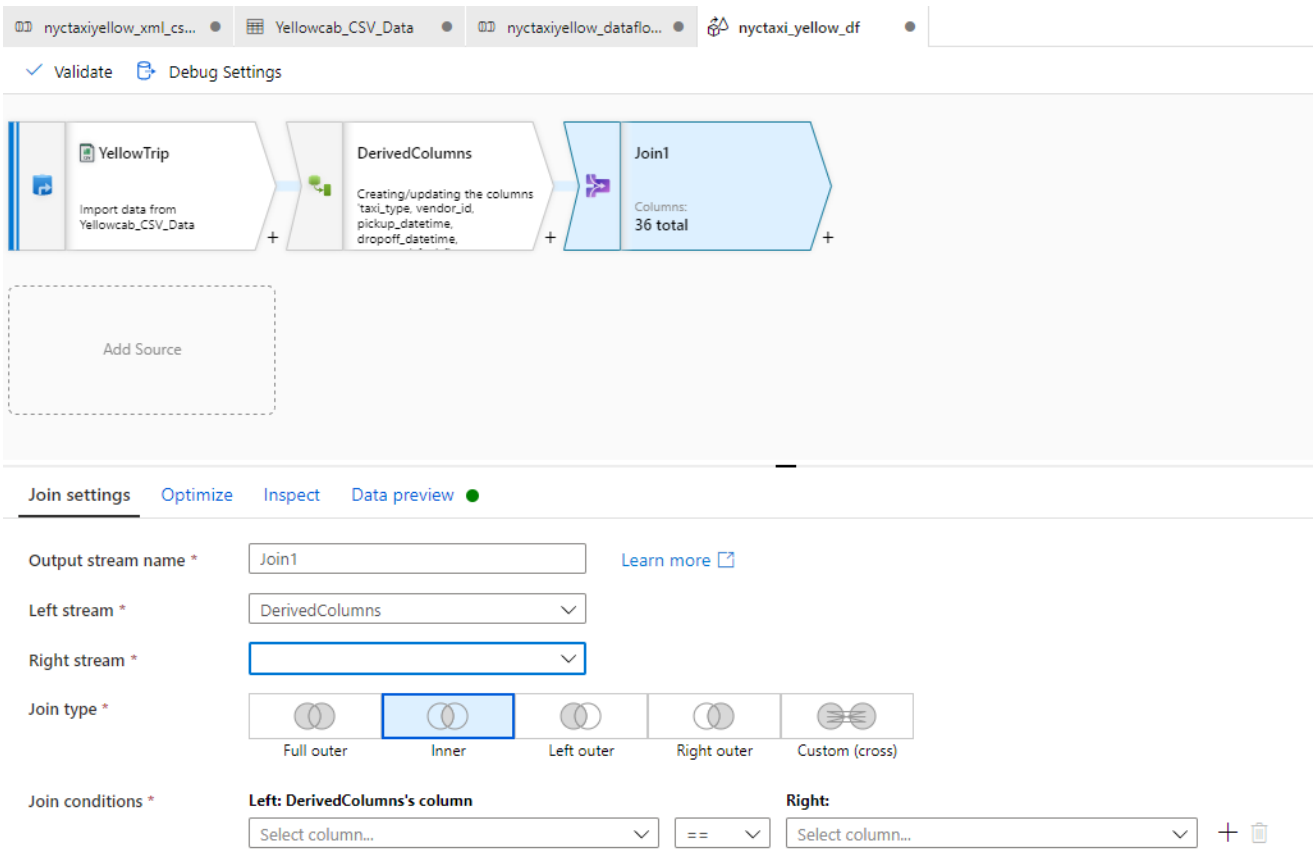
18. Your final list of columns should look like this:



19. If you click on **Inspect**, you can see the new columns that will be created. **Data preview** will show you a selection of the data in those columns.

20. Click the + sign at the bottom-left of your newly-created DerivedColumns step, and select **Join**.

21. You'll see another set of configuration options for the Join. Change the name to "PaymentJoin."

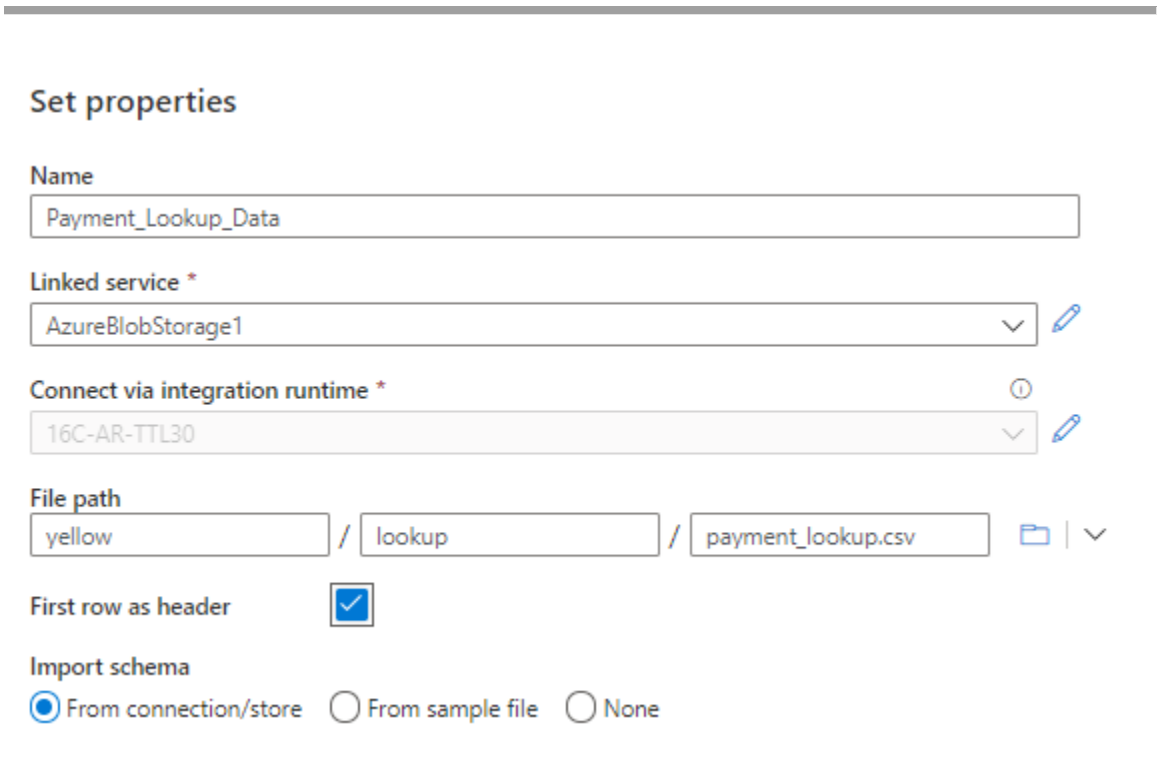


22. In order to complete this join, you'll need to add another data source. The **Left stream** will default to the output from the prior step. To set up the **Right stream**, you'll need to click **Add Source**.

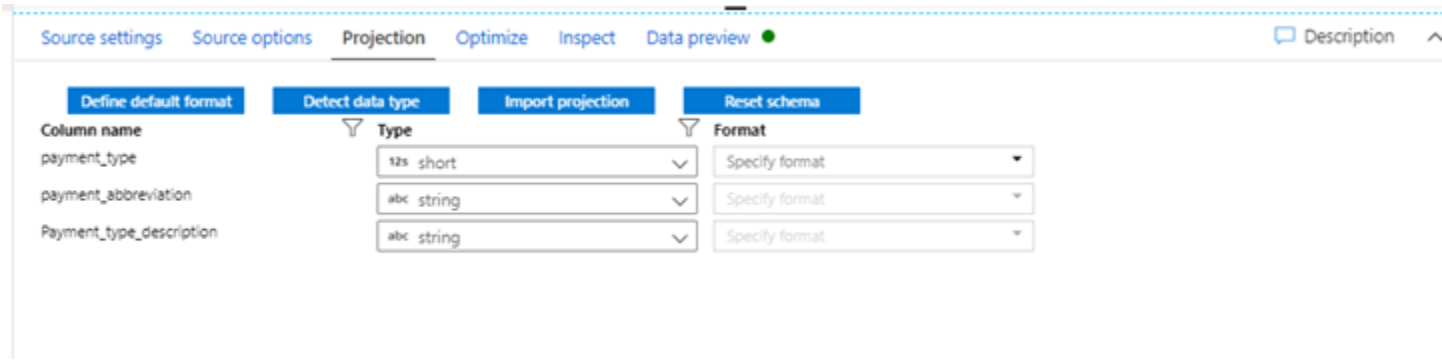
    i. A new set of **Source settings** will appear.
    ii. Call this **Output stream name** "PaymentLookup."
    iii. This **Dataset** will not yet appear in the dropdown, so you'll need to click **+ New** and add it.

        a. On the **New dataset** blade, select **Azure Blob Storage** and click **Continue**.
        b. Choose **CSV/DelimitedText** as the file format, and click **Continue**.

c. Edit the properties on the **Set properties** blade and click **OK.**
   a. **Name**: "Payment_Lookup_Data"
   b. **Linked service:** do not change
   c. **File path**: yellow / lookup / payment_lookup.csv



ii. Click on the **Projection** tab and change the "payment_type" to short.



v. Preview the data by clicking on **Data preview**.

Source settings | Source options | Projection | Optimize | Inspect | Data preview | Description

Number of rows INSERT 24 | UPDATE 0 | DELETE 0 | UPSERT 0 | LOOKUP 0 | TOTAL 24

Refresh | Typecast | Modify | Map drifted | Statistics | Remove

payment_type	payment_abbreviation	Payment_type_description
1	NULL	Credit card
2	NULL	Cash
3	NULL	No charge
4	NULL	Dispute
5	NULL	Unknown
6	NULL	Voided trip
7	CAS	Cash
8	CASH	Cash
9	CRD	Credit card
10	CRE	Credit card
11	CREDIT	Credit card
12	CSH	Cash
13	Cas	Cash
14	Cash	Cash

23. With the new data source ready, click back on the **Join** segment of the data flow. You will now see the PaymentLookup data source in the **Right stream** dropdown.

Output stream name * PaymentJoin

Left stream * DerivedColumns

Right stream *

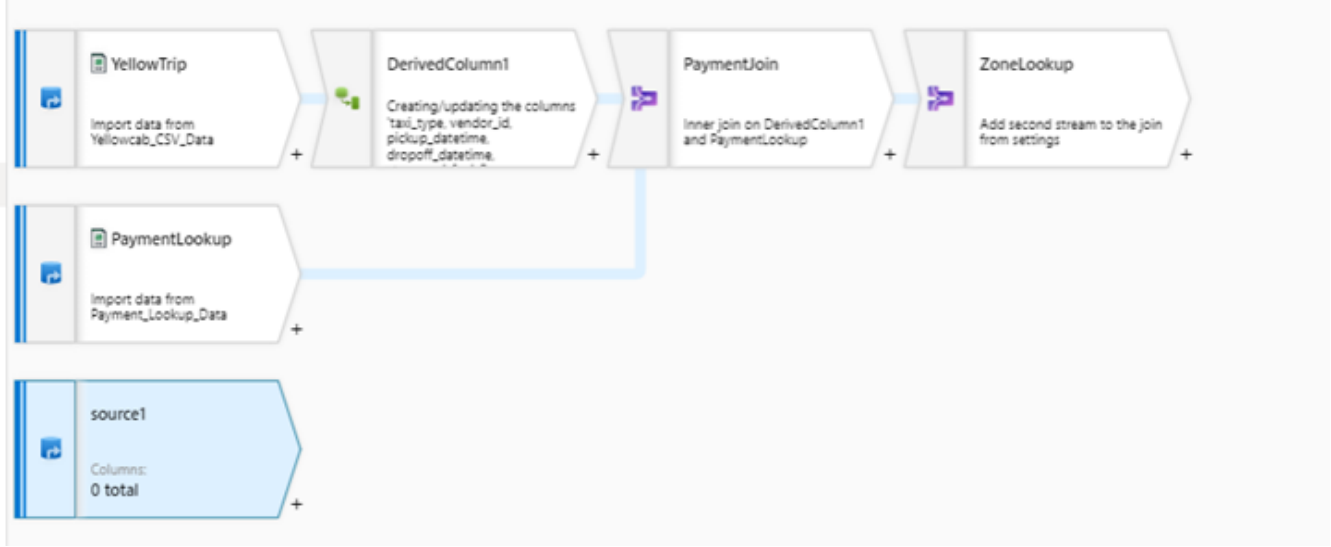
Join type *

YellowTrip

PaymentLookup

24. Select payment_type in both dropdowns - the **Left: DerivedColumns's column** and in **Right: PaymentLookup's column**. This determines the column on which the two source tables will be joined.

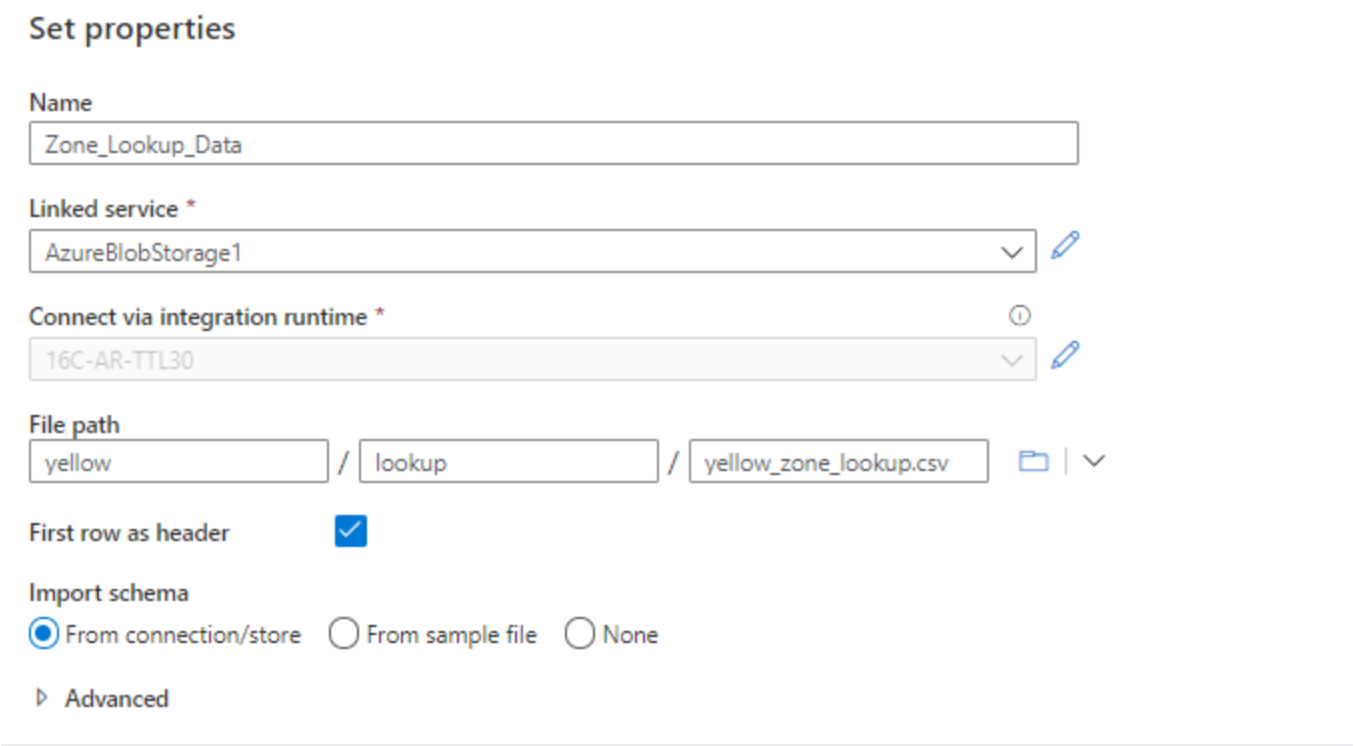
25. Your next step will be another **Join**. For this one, let's set up the data source first. Click **Add Source** on the data flow canvas.



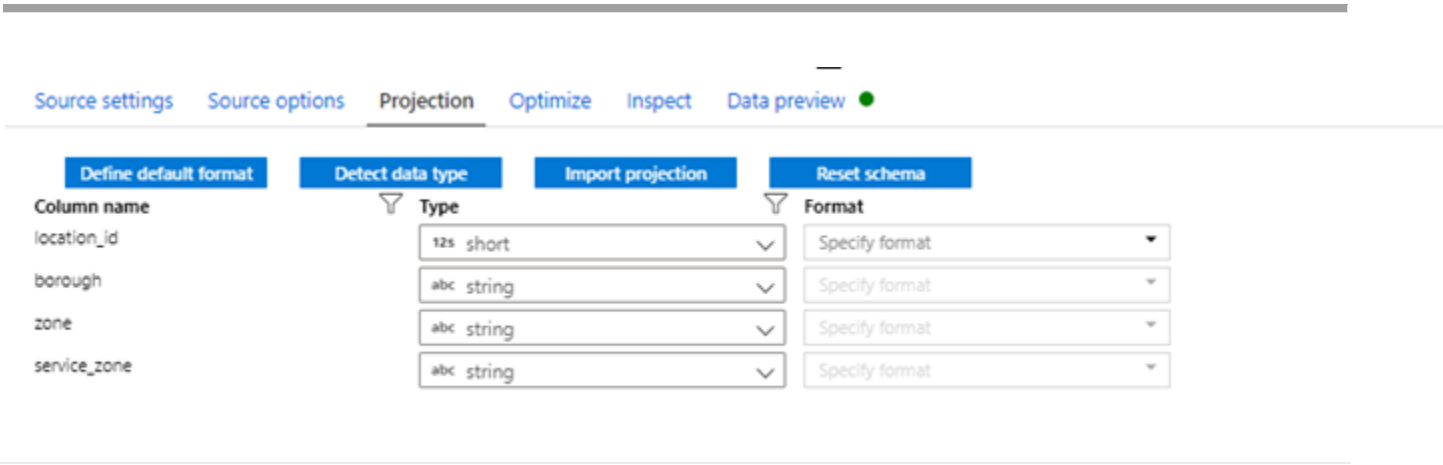
26. As before, you'll need to configure this data source. Name it "ZoneLookup" and click the **+ New** button next to **Dataset**.

    - Select **Azure Blob Storage** and click **OK**.
    - Select **CSV/DelimitedText** and click **Continue**.
    - Edit the properties on the **Set properties** blade and click **OK.**

        a. **Name**: "Zone_Lookup_Data"
        b. **Linked service**: AzureBlobStorage1
        c. **File path:** yellow / lookup / yellow_zone_lookup.csv
        d. **Import schema:** From connection/store
        e. Check **First row as header**.



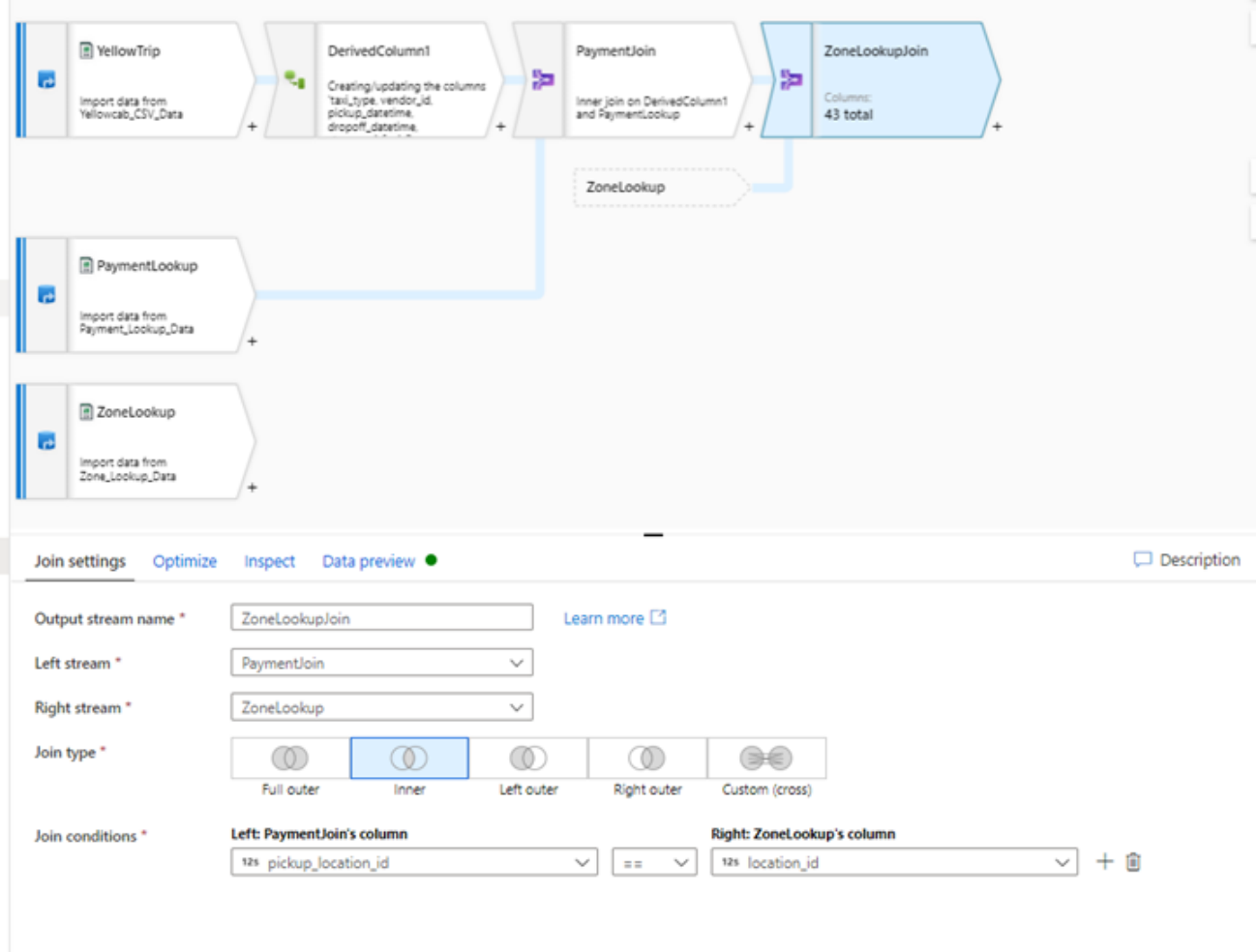
iv. Click the **Projection** tab and change the location_id data **Type** to short.



27. Next, click the + below the **PaymentJoin** block in the canvas, and select **Join** again.

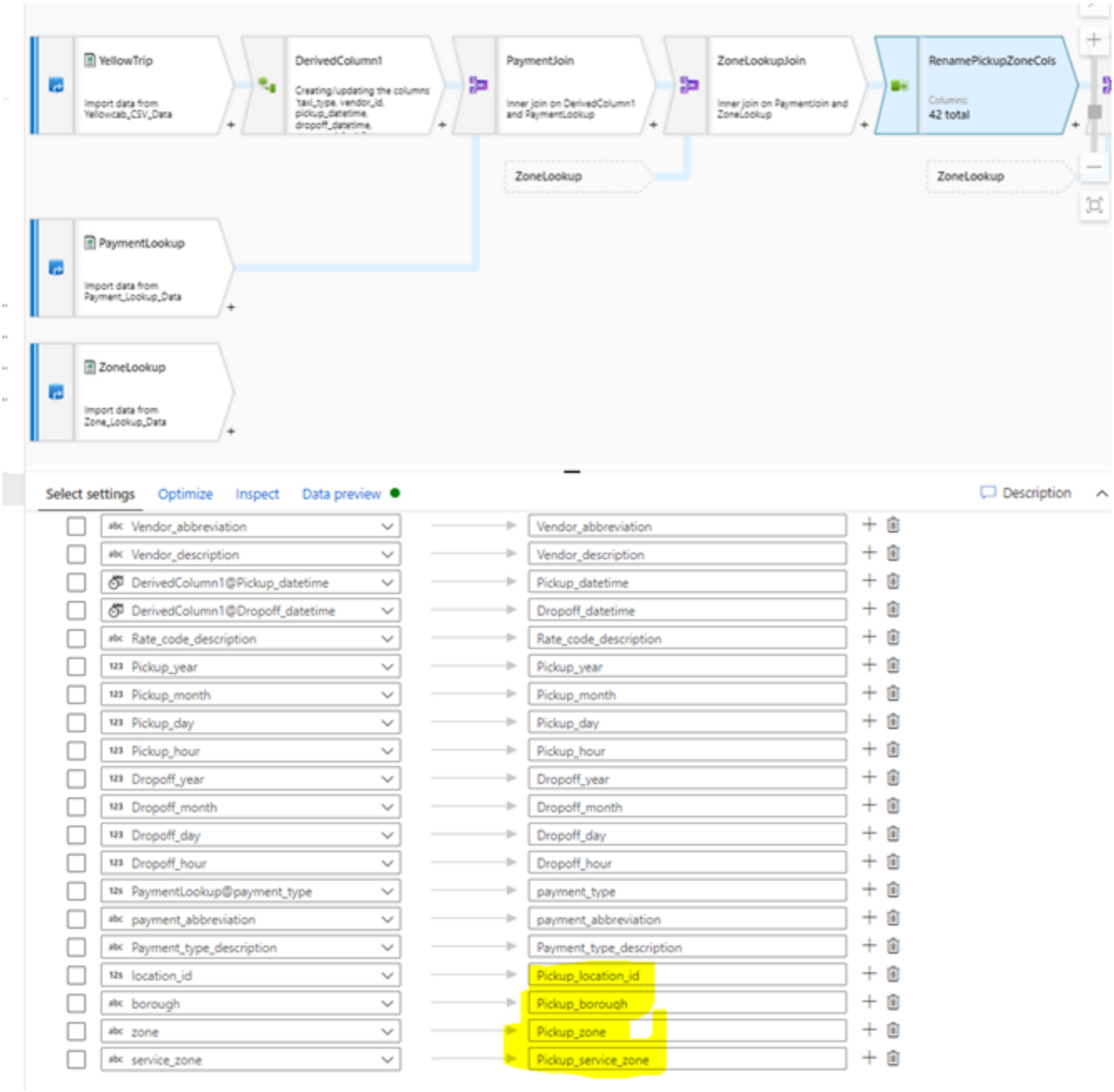
28. Configure this **Join** using the settings below:

    i. **Output stream name:** PickupZoneJoin
    ii. **Left stream:** PaymentJoin
    iii. **Right stream:** ZoneLookup
    iv. **Join conditions:** pickup_location_id == location_id



29. The next step you'll add is a **Select** step.

30. Change the **Output stream** name to "RenamePickupZoneCols" and rename the columns as you see below:



31. Add another **Join**, repeating the steps above but with the following settings changed:

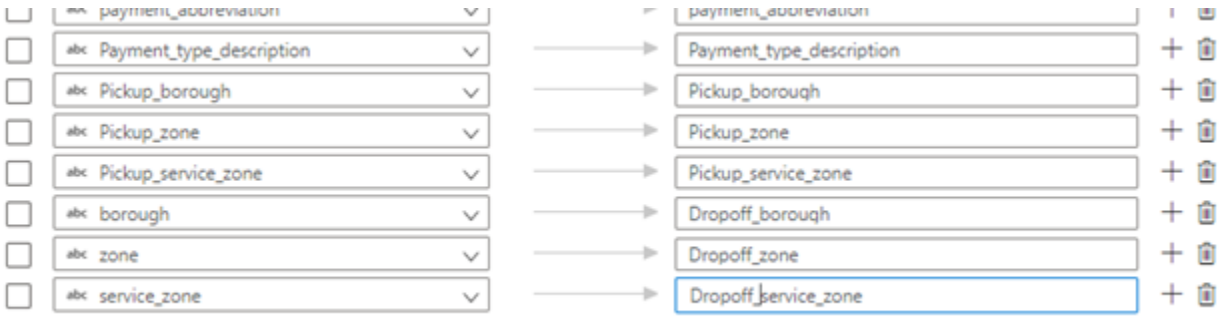
    i. **Output stream name:** DropoffZoneJoin
    ii. **Left stream:** RenamePickupZoneCols
    iii. **Right stream:** ZoneLookup
    iv. **Join conditions:** dropoff_location_id == location_id

32. Add another **Select** step. Rename it "RenameDropoffZoneCol"

    . Remove the following columns from your mapping:
        a. pickup_datetime
        b. dropoff_datetime

c. pickup_location_id
d. dropoff_location_id
e. pickup_longitude
f. pickup_latitude
g. dropoff_longitude
h. dropoff_latitude
i. location_id (*Make sure to remove both.*)

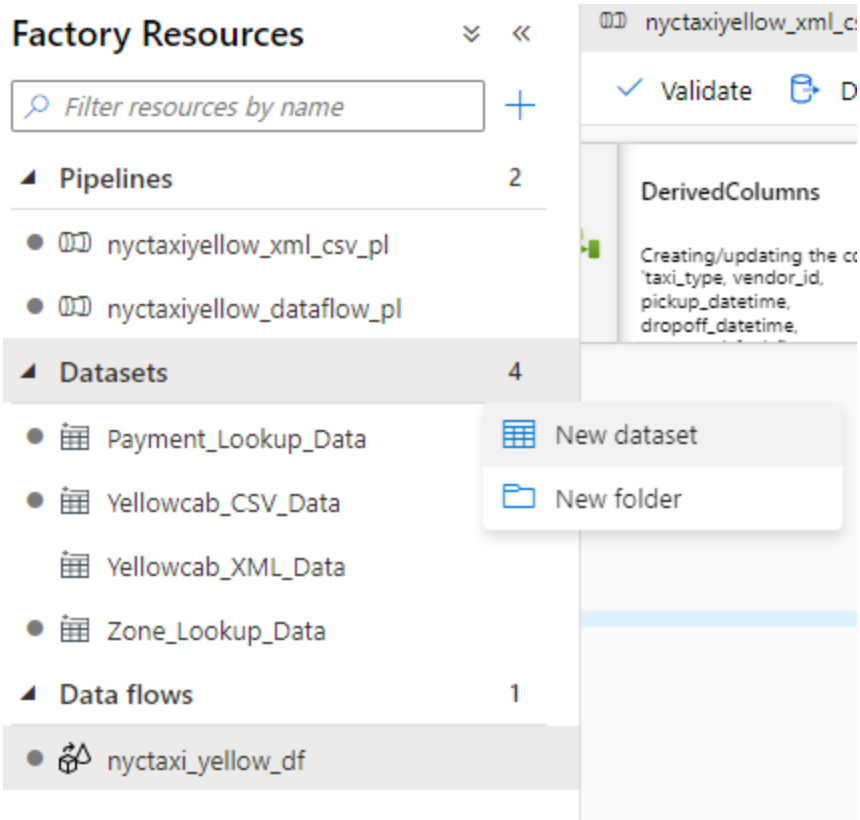
i. Rename the last three columns to avoid a naming clash between borough, zone, and service_zone columns:



---

32. You're almost done!

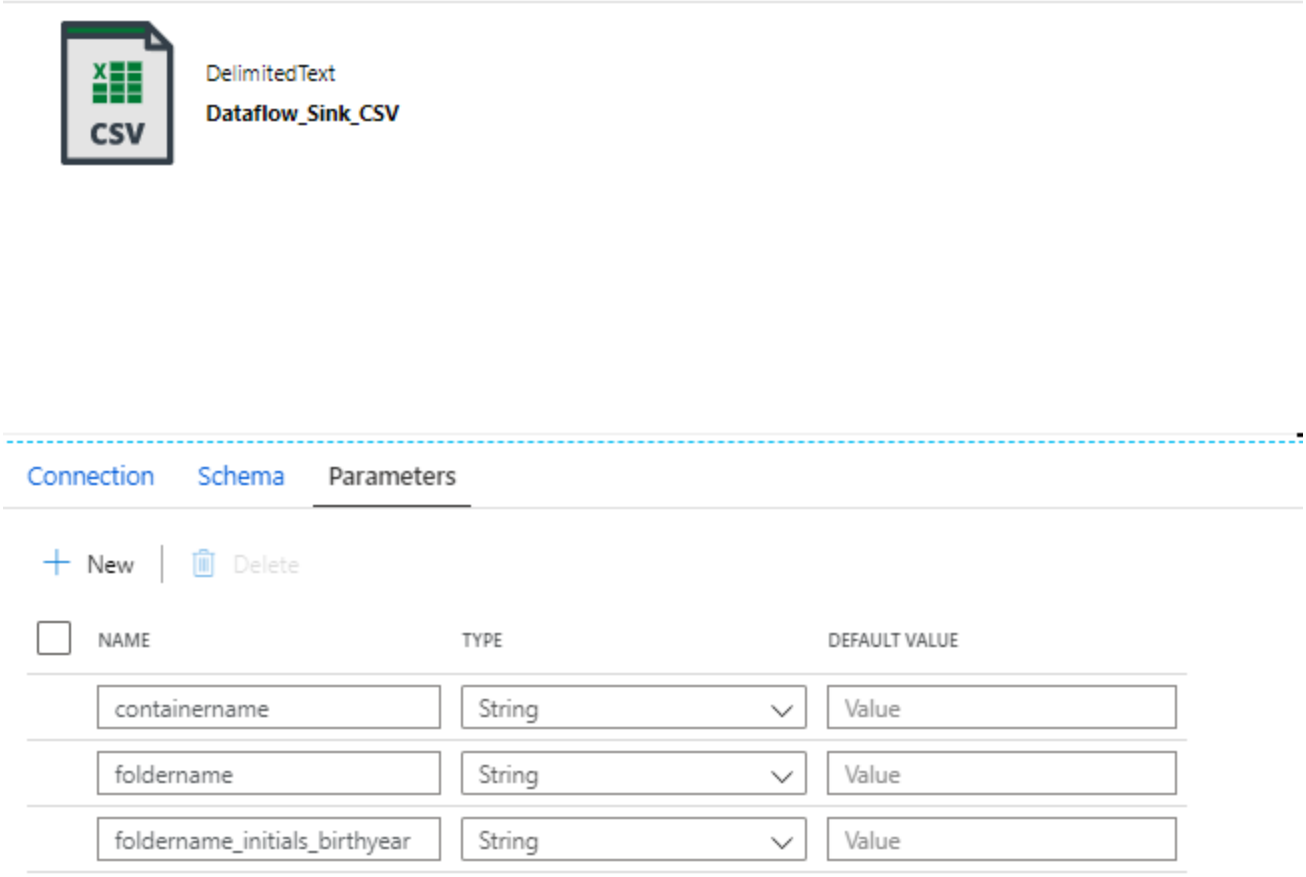
33. We will need to configure a destination for the data that is transformed. Your list of **Factory Resources** has grown. Click the ... ellipses next to **Datasets** and select **New dataset**.



i. Configure the new dataset.

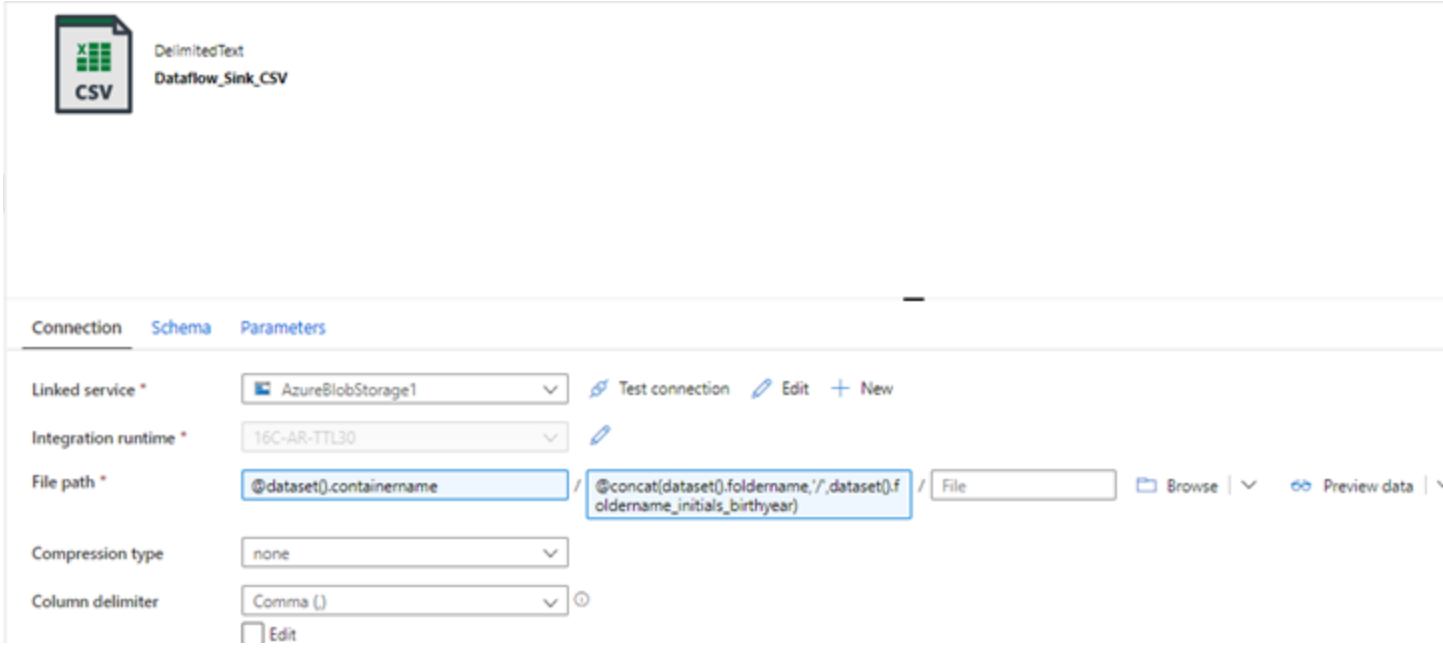
   a. Select **Azure Blob Storage** and click **Continue**.
   b. Select **CSV/DelimitedText** and click **Continue**.
   c. On the **Set properties** blade:
      a. **Name**: ak195_Dataflow_Sink_CSV
      b. **Linked service**: Select the one you've been using

ii. On the **Connection** tab, you'll need to configure the **File path** using dynamic parameters similar to the ones you've configured before.



Your settings should ultimately look like the settings below, using the following for the **Directory** field:

- @concat(dataset().foldername,'/',dataset().foldername_initials_birthyear)

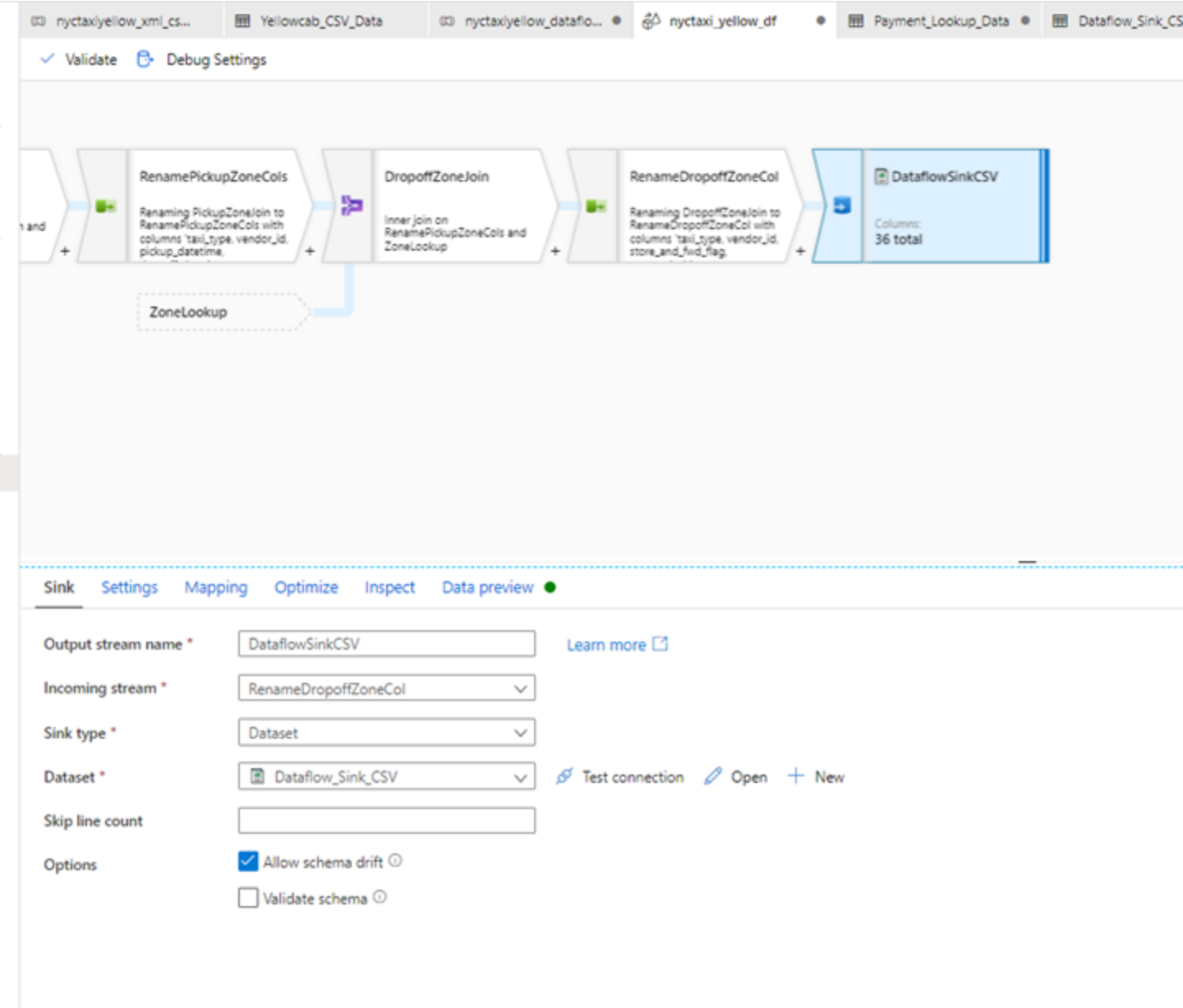


34. Return to your dataflow canvas and click the **+** sign one more time, and select **Destination -\ Sink**. A sink is the destination for your data once you've completed all of these transformation steps.

    . On the **Sink** tab, configure the following settings:
        a. Output **stream name**: DataflowSinkCSV **Dataset**: ak1985_Dataflow_Sink_CSV (the dataset you created above should be available in the dropdown)

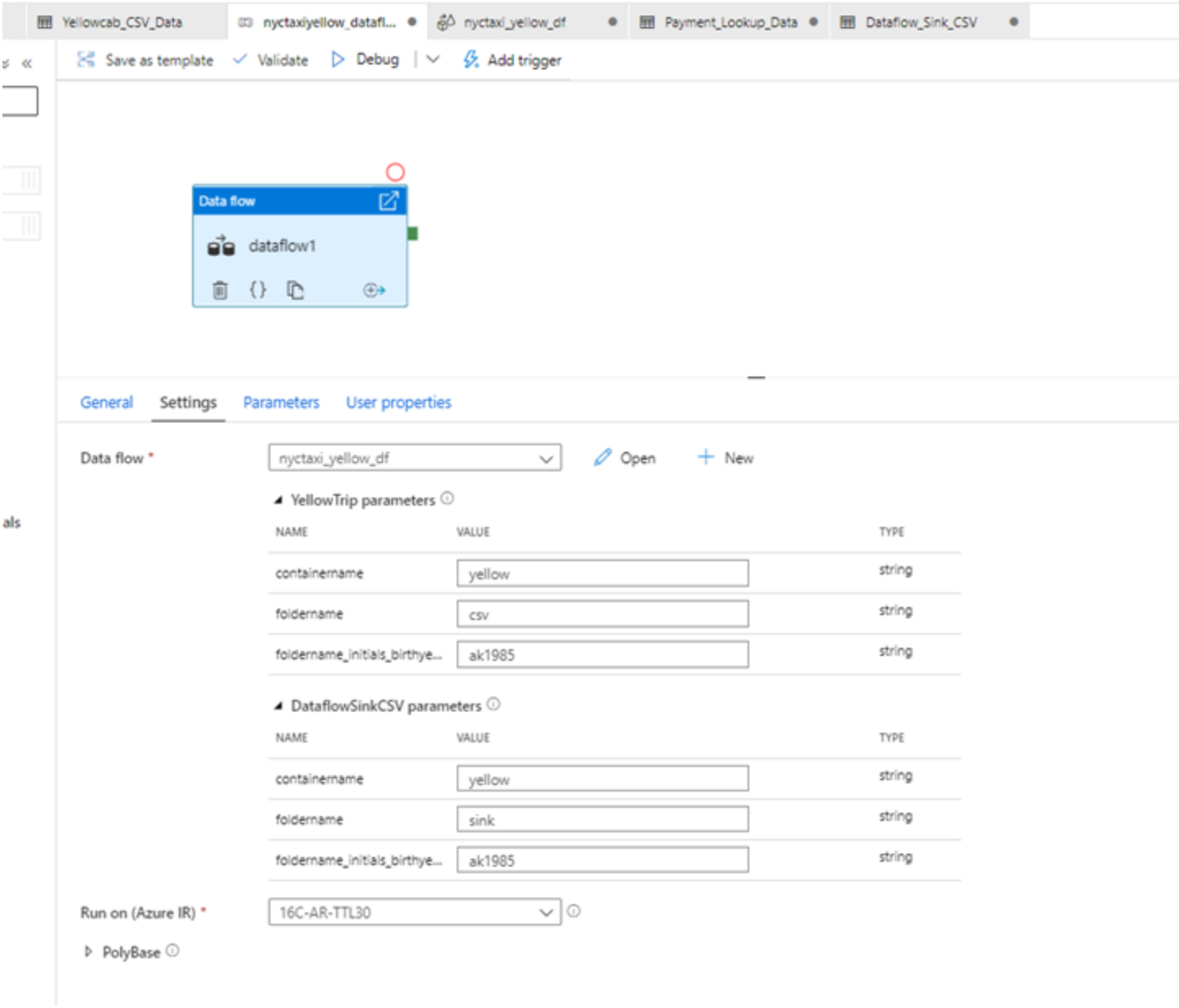
    i. On the **Settings** tab, configure the following settings:
        a. **Clear the folder:** ON
        b. **File name option**: Output to single file
        c. You may see an error here that asks you to Set single partition. If you do, click that button before proceeding.
        d. **Output to single file**: nyctaxiyellow_final.csv

---



35. Finally, return to the nyctaxiyellow_dataflow_pl pipeline.

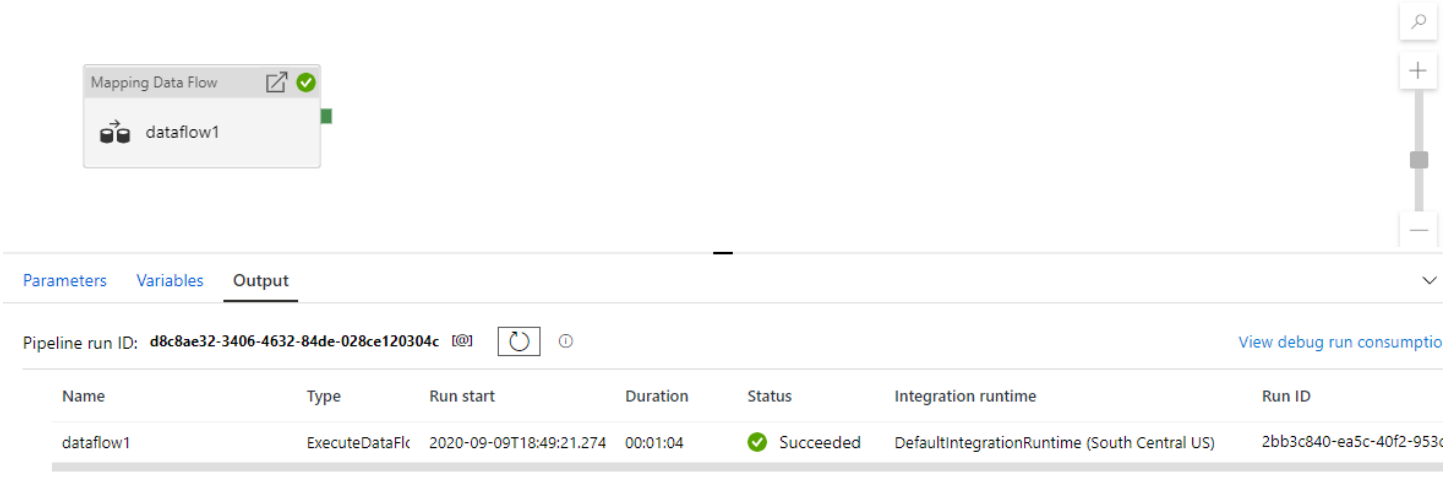
36. Click on the **Mapping Data Flow** block on the canvas, and configure both sets of parameters in the **Settings** to reflect the containername, foldername, and initials_birthyear information. Make sure that the **Run on (Azure IR)** field reads "dataflowruntime2".



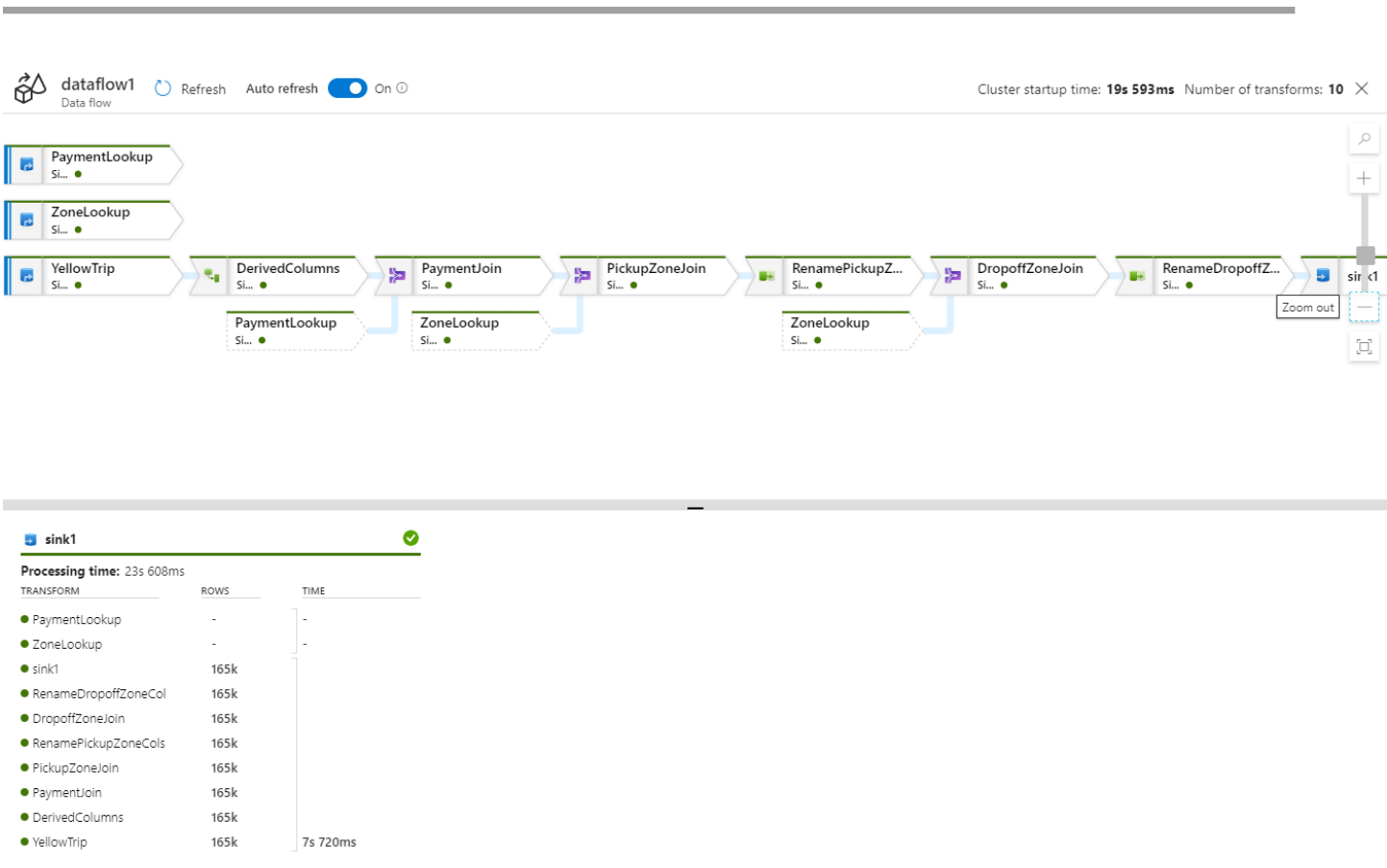
37. Click **Validate all** above the dataflow canvas to check your work. If you receive any error messages, please check in with one of your coaches.

38. Click **Publish all**, review the publication changes in the blade, and click **Publish** to save your work. If you receive any error messages, please check in with one of your coaches.

39. Click **Debug** above the pipeline canvas. The pipeline will be deployed, and you will receive a status update as with the prior pipeline.



40. Hover over the pipeline run and select the eyeglass icon. You will see an overview of the dataflow, including the status at every step.



Building a Machine Learning Model

The next and final pipeline will use a machine learning model that you develop using Azure's no-code drag & drop machine learning Designer.

Before setting up this pipeline, you will need to train the model in that service.

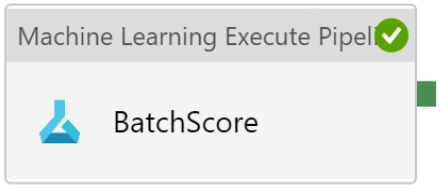
Please follow the instructions for the **Machine Learning Lab** and return here when you've completed the model.

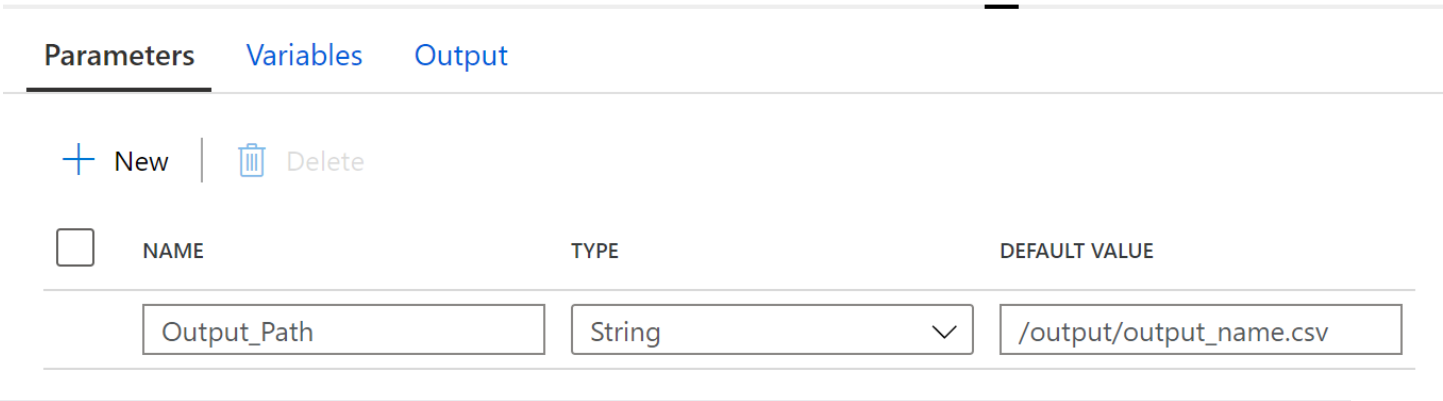
1. Now that your model is complete, create one more pipeline. Call this pipeline "ak1985_nyctaxiyellow_ml_scoring_pl" or something similar.

2. Select the ***Machine Learning Execute** activity under **Activities** and drag it onto the canvas.

3. On the **Parameters** tab, configure the settings below:

   i. **Name**: Output_Path
   ii. **Type**: String
   iii. **Default value**: /output/output_name.csv





4. Click on the Machine Learning Execute Pipeline block on the canvas and configure your settings to resemble the settings below.

   i. Your **Machine Learning pipeline name** and **Machine Learning pipeline ID** will auto-populate from the pipeline you published in the Machine Learning lab.
   ii. You will need to configure the parameter in the Output_Path like you configured the other parameters above.

5. As with the prior pipelines, verify your configuration by clicking **Validate all**, publish your work, and click **Debug** to see your Machine Learning pipeline in action!

Additional References

1. Introduction to Azure Data Factory
2. Mapping data flows in Azure Data Factory
3. Wrangling data flows in Azure Data Factory