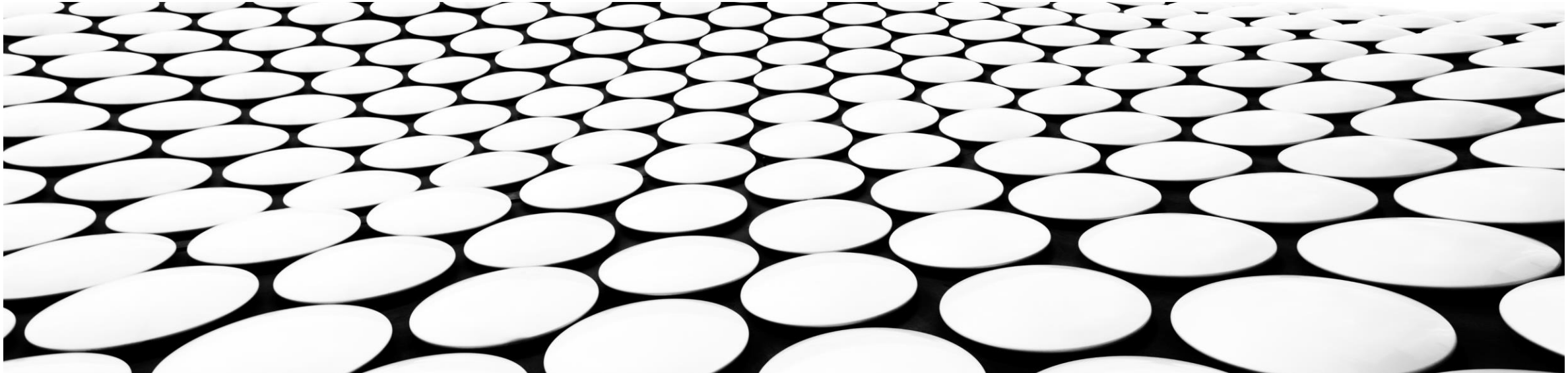
AZURE DATA FACTORY

FILE AND PIPELINE METADATA

SCOTT PETERS

https://advancedsqlpuzzles.com



OBJECTIVE

In this presentation we will create several pipelines to import text files into a SQL Server database and append all metadata from both the text files and the pipeline.

This demo will cover the following topics:

- File and Pipeline Metadata
- Parameters and Variables
- User-Defined Table Types

First, we will start with Metadata.

METADATA

What is metadata?

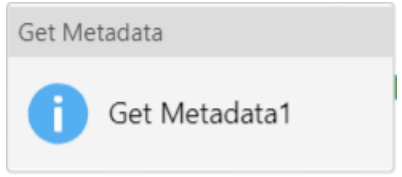
['medəˌdādə, 'medəˌdadə]

NOUN

a set of data that describes and gives information about other data.

METADATA

Azure Data Factory provides the Get Metadata activity to retrieve metadata from a dataset.



"You can use the Get Metadata activity to retrieve the metadata of any data in Azure Data Factory or a Synapse pipeline. You can use the output from the Get Metadata activity in conditional expressions to perform validation, or consume the metadata in subsequent activities."

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-get-metadata-activity

METADATA

Metadata type	Description
itemName	Name of the file or folder.
itemType	Type of the file or folder. Returned value is `File` or `Folder`.
size	Size of the file, in bytes. Applicable only to files.
created	Created datetime of the file or folder.
lastModified	Last modified datetime of the file or folder.
childItems	List of subfolders and files in the given folder. Applicable only to folders. Returned value is a list of the name and type of each child item.
contentMD5	MD5 of the file. Applicable only to files.
structure	Data structure of the file or relational database table. Returned value is a list of column names and column types.
columnCount	Number of columns in the file or relational table.
exists	Whether a file, folder, or table exists. If `exists` is specified in the Get Metadata field list, the activity won't fail even if the file, folder, or table doesn't exist. Instead, `exists: false` is returned in the output.

Using a dataset, we can specify the following metadata types in the Get Metadata activity field list.

Some of the metadata types are specific to folders, and some are specific to files.

I've noted the type size is listed in the documentation but does not appear in the Get Metadata drop down, which we will see later.

PARAMETERS VS VARIABLES

Next, we will cover parameters and variables.

PARAMETERS VS VARIABLES

Parameters

- A parameter is a named variable passed into a function by the callee.
- The value of a parameter cannot be changed inside the function.
- The values passed to a parameter are called arguments.

Variables

- A variable is a memory location.
- The value of a variable can change during program execution.
- Variables are used to store information to be referenced and manipulated in a computer program.

PARAMETERS VS VARIABLES

You can use **parameters** to pass external values into:

1) Pipelines
2) Datasets
3) Linked Services
4) Data Flows

Once the **parameter** has been passed into the resource, it cannot be changed. By parameterizing resources, you can reuse them with different values each time.

PARAMETERS VS VARIABLES

Data Factory has two types of **variables**:

1) System Variables that capture pipeline metadata.
2) User Variables to store values which are referenced by the pipeline.

User variables can be set via the Set variable or the Append variable activity. There are three data types of user-defined variables available:

1) String
2) Boolean
3) Array

PARAMETERS VS VARIABLES

System variables capture pipeline metadata and can be referenced anywhere in the pipeline.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-get-metadata-activity

Variable Name	Description
@pipeline().DataFactory	Name of the data or Synapse workspace the pipeline run is running in
@pipeline().Pipeline	Name of the pipeline
@pipeline().RunId	ID of the specific pipeline run
@pipeline().TriggerType	The type of trigger that invoked the pipeline (for example, ScheduleTrigger, BlobEventsTrigger). For a list of supported trigger types, see Pipeline execution and triggers. A trigger type of Manual indicates that the pipeline was triggered manually.
@pipeline().TriggerId	ID of the trigger that invoked the pipeline
@pipeline().TriggerName	Name of the trigger that invoked the pipeline
@pipeline().TriggerTime	Time of the trigger run that invoked the pipeline. This is the time at which the trigger actually fired to invoke the pipeline run, and it may differ slightly from the trigger's scheduled time.
@pipeline().GroupId	ID of the group to which pipeline run belongs.
@pipeline()?TriggeredBy PipelineName	Name of the pipeline that triggers the pipeline run. Applicable when the pipeline run is triggered by an ExecutePipeline activity. Evaluate to *Null* when used in other circumstances. Note the question mark after @pipeline()
@pipeline()?TriggeredBy PipelineRunId	Run ID of the pipeline that triggers the pipeline run. Applicable when the pipeline run is triggered by an ExecutePipeline activity. Evaluate to *Null* when used in other circumstances. Note the question mark after @pipeline()

DATA FACTORY

Next, we will briefly cover linked services and datasets.

DATA FACTORY

What are datasets and linked services?

"A **dataset** is a named view of data that simply points or references the data you want to use in your activities as inputs and outputs. Datasets identify data within different data stores, such as tables, files, folders, and documents. For example, an Azure Blob dataset specifies the blob container and folder in Blob storage from which the activity should read the data.

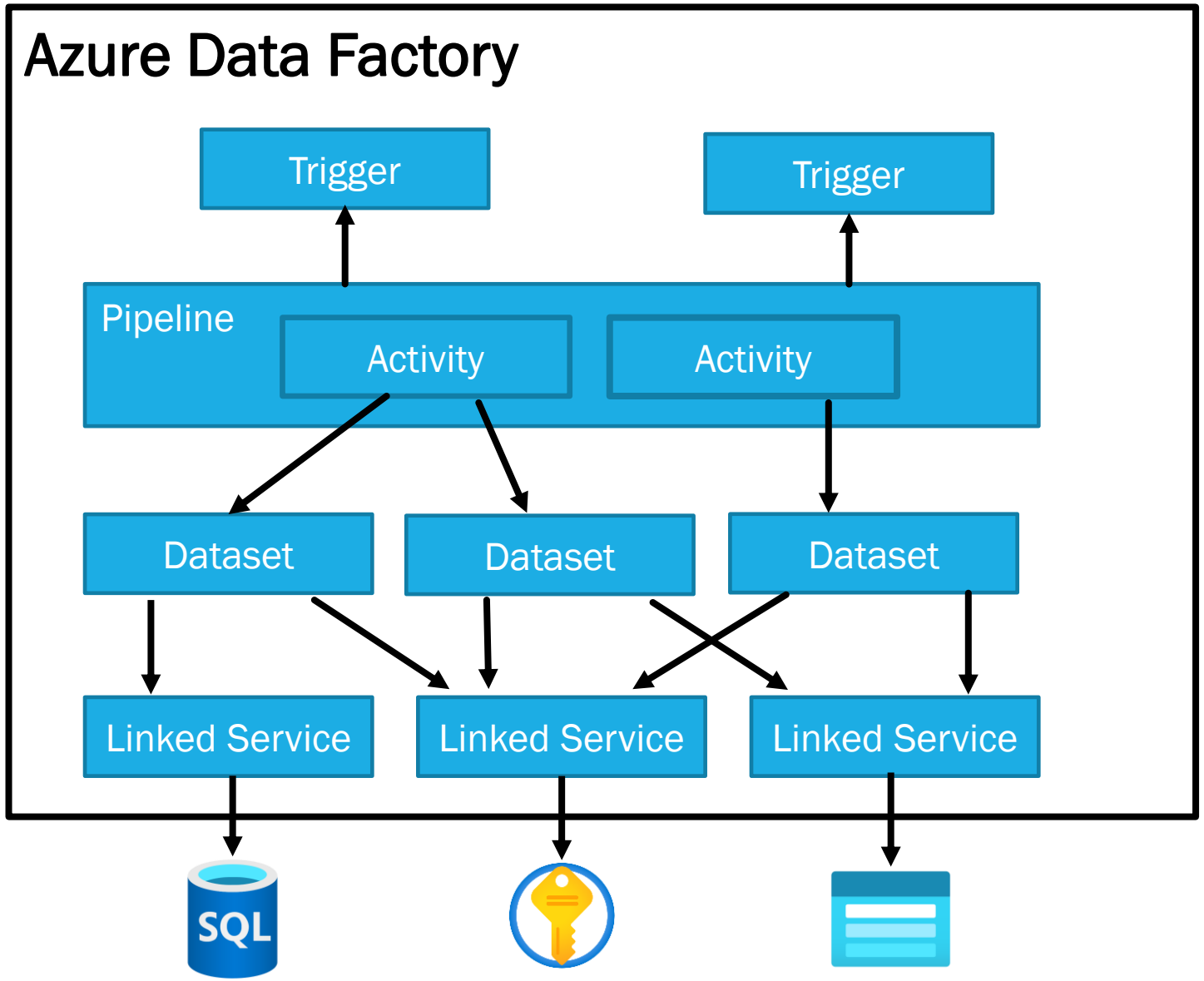
Before you create a dataset, you must create a **linked service** to link your data store to the service. **Linked services** are much like connection strings, which define the connection information needed for the service to connect to external resources."

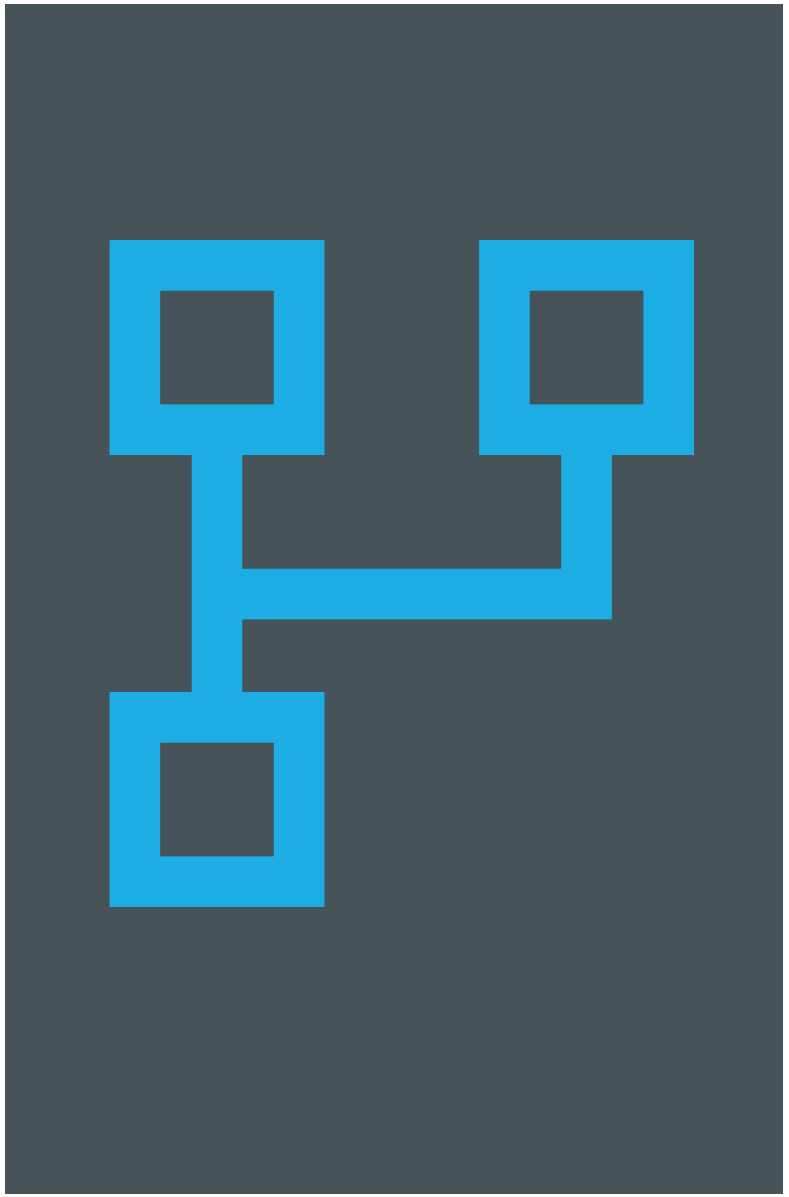
https://docs.microsoft.com/en-us/azure/data-factory/concepts-datasets-linked-services

DATA FACTORY

General framework of an Azure Data Factory.

The joining of triggers, pipelines, activates, datasets and linked services is a many to many relationship.





PIPELINE 1

Before we begin creating a pipeline to insert metadata into a SQL Server table.

We will first create a pipeline with a Copy data activity that inserts data from a text file into a SQL Server table without any extra metadata.

The setup to insert metadata into a SQL Server table is a bit tricky. We will start with a simple pipeline and then build from here.

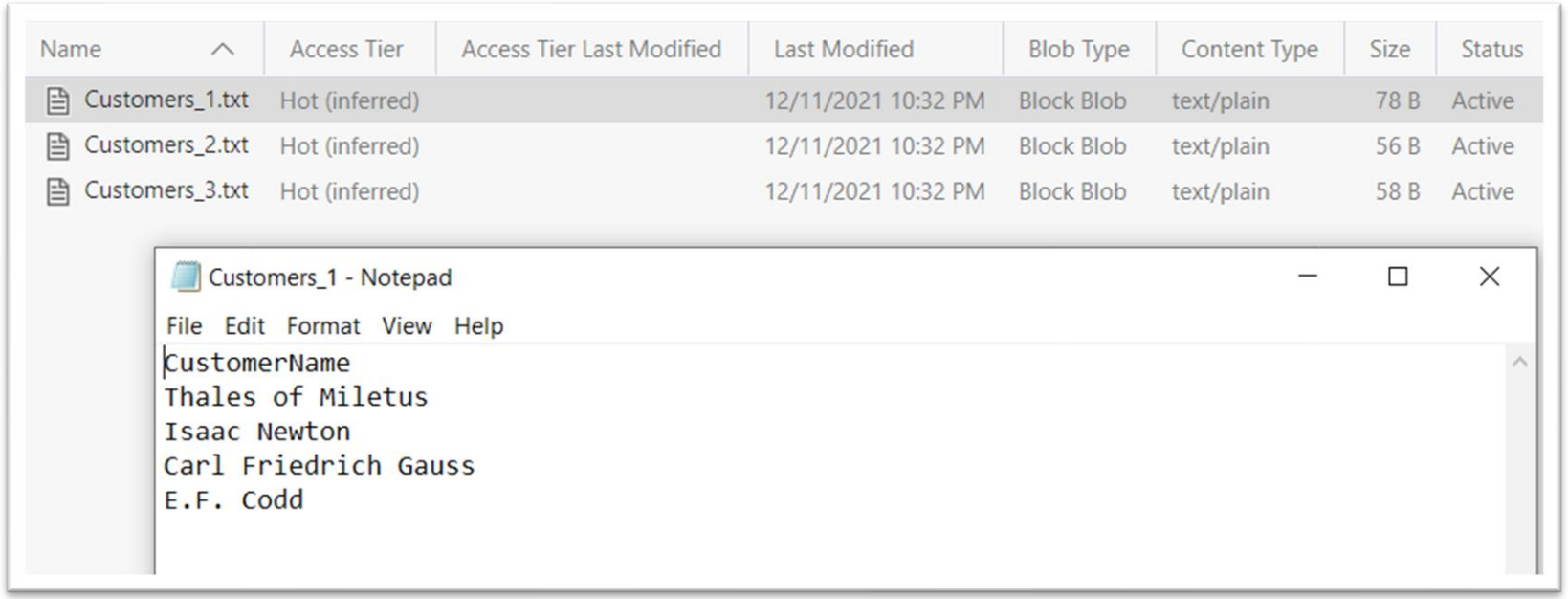
PIPELINE 1

We will insert the data from a set of customer files into the following SQL Server table.

```
CREATE TABLE demo.Customers
    (
    InsertDate DATETIME NULL DEFAULT GETDATE(),
    CustomerName VARCHAR(100) NULL
    )
```

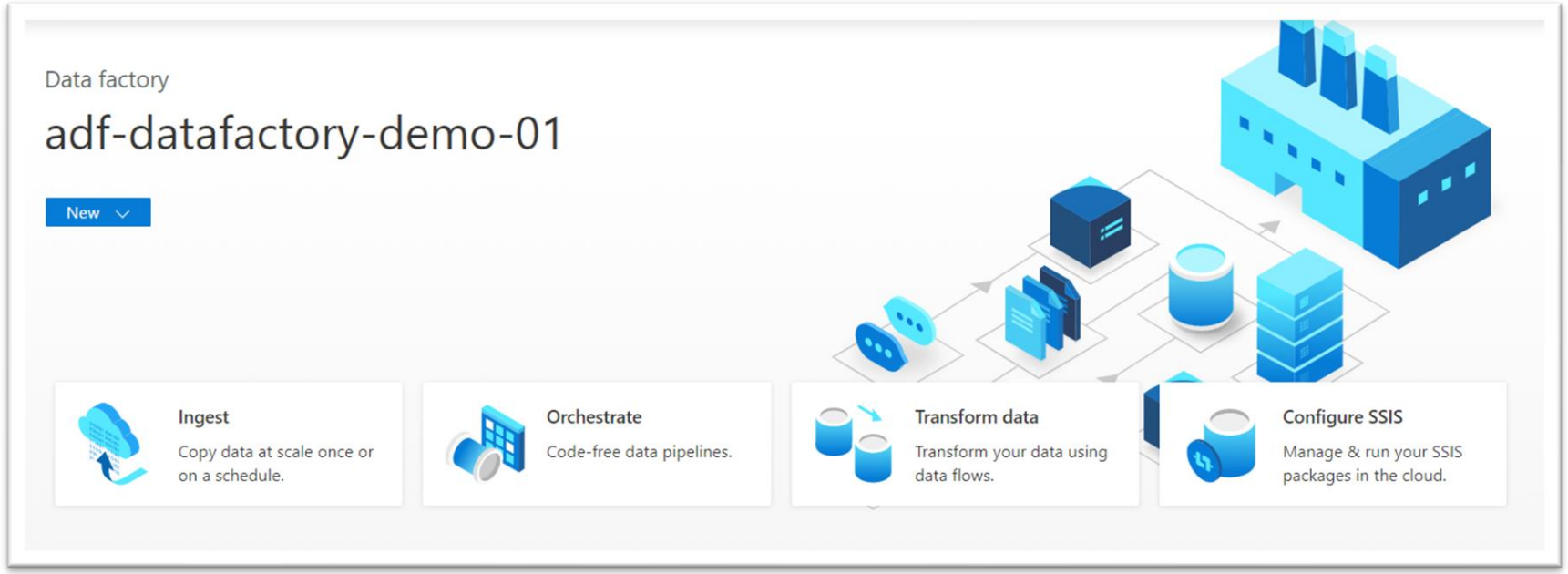
PIPELINE 1

I created 3 text files and inserted the names of famous mathematicians into the files. We will be inserting these files into the Customers table we created.



PIPELINE 1

The fastest method to create a pipeline is to select the Ingest option from the home screen. This method will auto-create the activities, linked services and datasets via a wizard.



PIPELINE 1

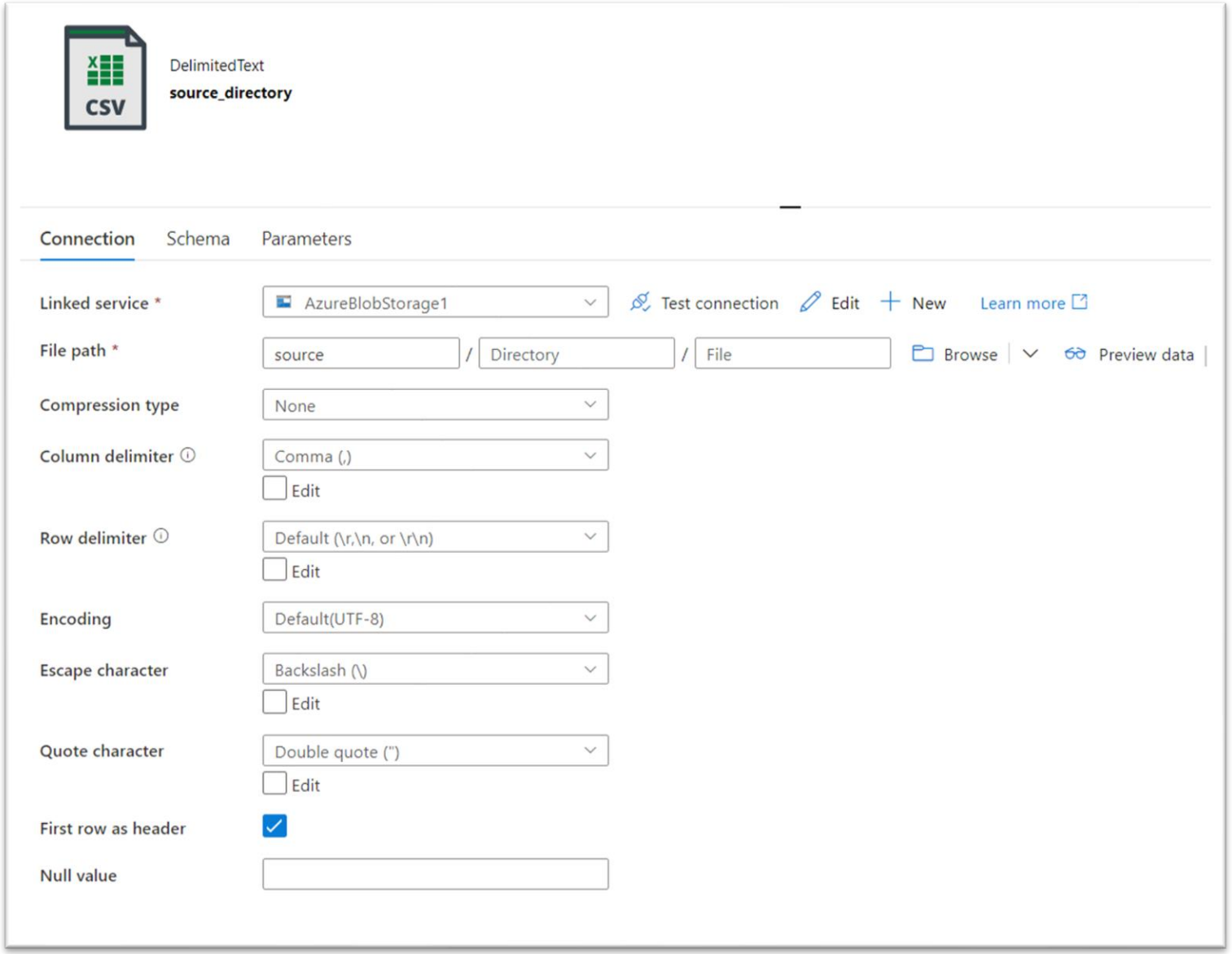
We will need to create the following two datasets for our pipeline.

Name	Data Store	Type
source_directory	Azure Blob Storage	Format type is DelimitedText
Customers_table	Azure SQL Database	References the table Customers

PIPELINE 1

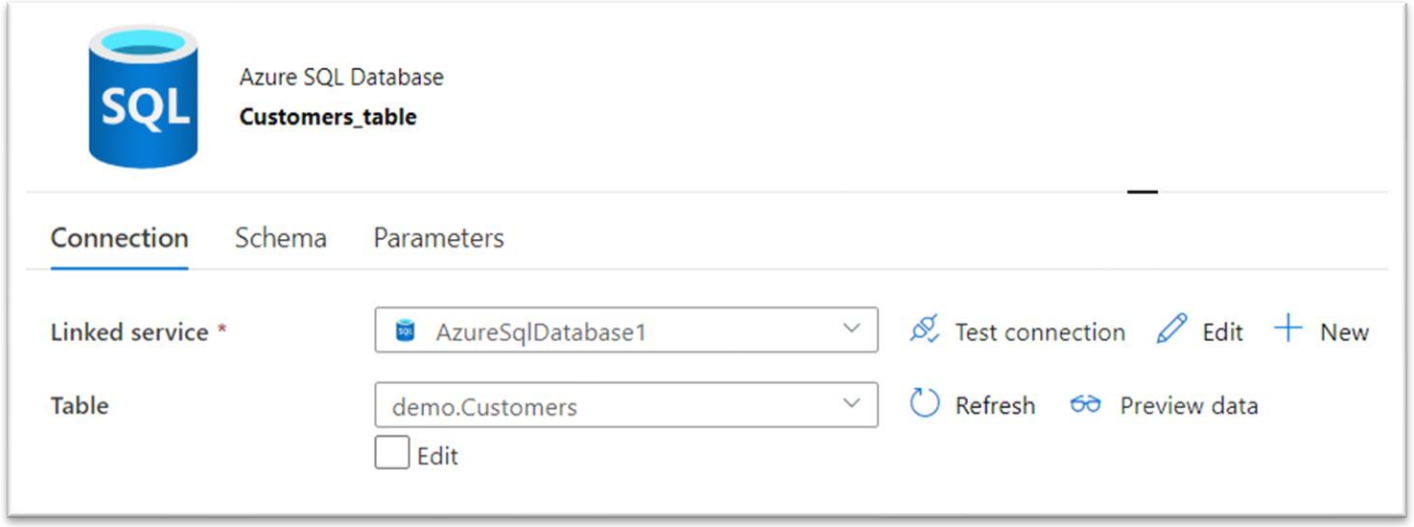
Create the dataset source_directory that points to the Azure Storage Account directory where you stored the customer text files.

Leave the filename blank.



PIPELINE 1

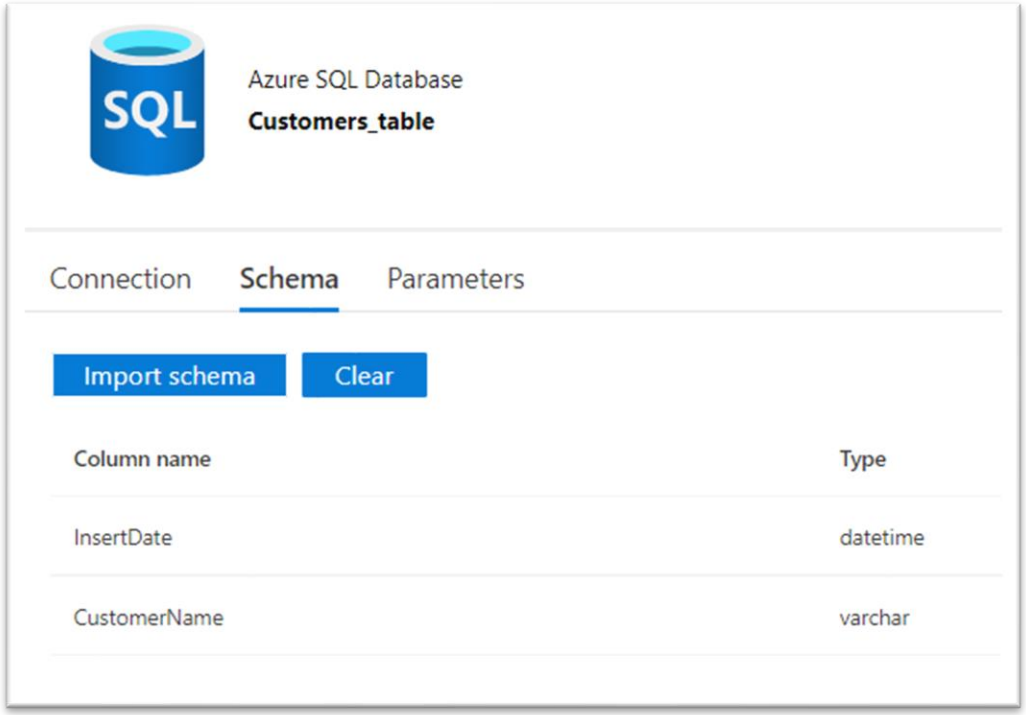
Create the Customers_table dataset referencing the table Customers.



PIPELINE 1

In the Schema tab, import the schemas for your datasets.

The schema in the datasets may be unneeded, but consider it best practice to set these and ensure they are correct.



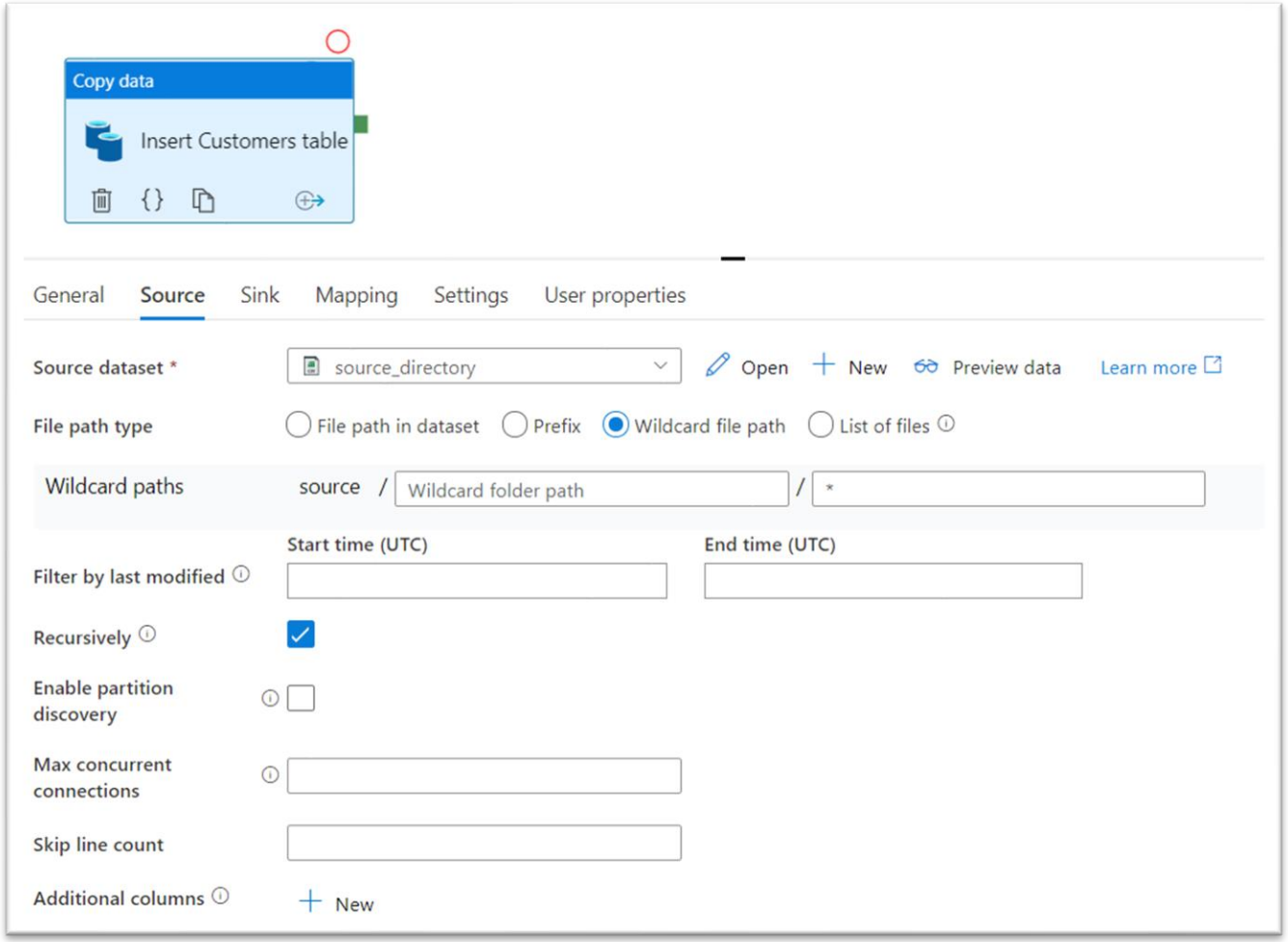
PIPELINE 1

Next, we will setup the Data Factory pipelines.

PIPELINE 1

Insert a Copy data activity into the workflow.

In the Source tab, select the Source dataset and select Wildcard folder path as the File path type.

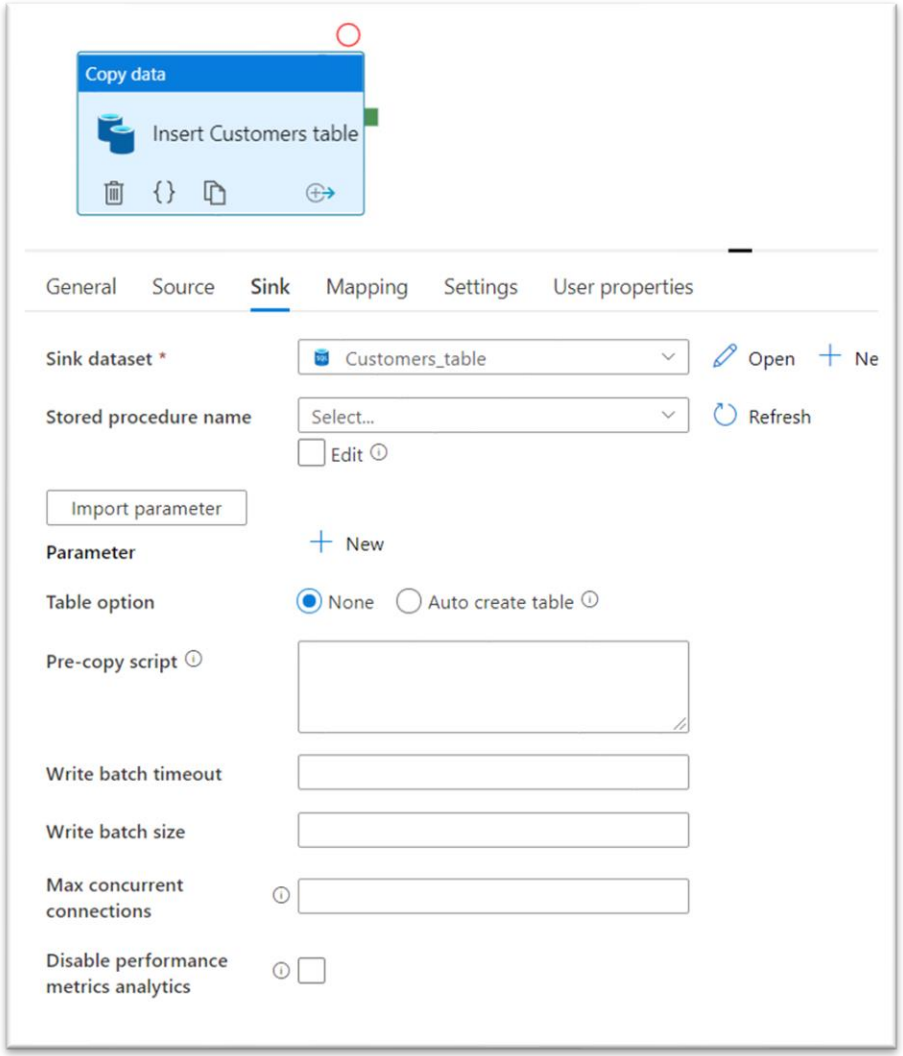


PIPELINE 1

The Sink tab will have the following setup.

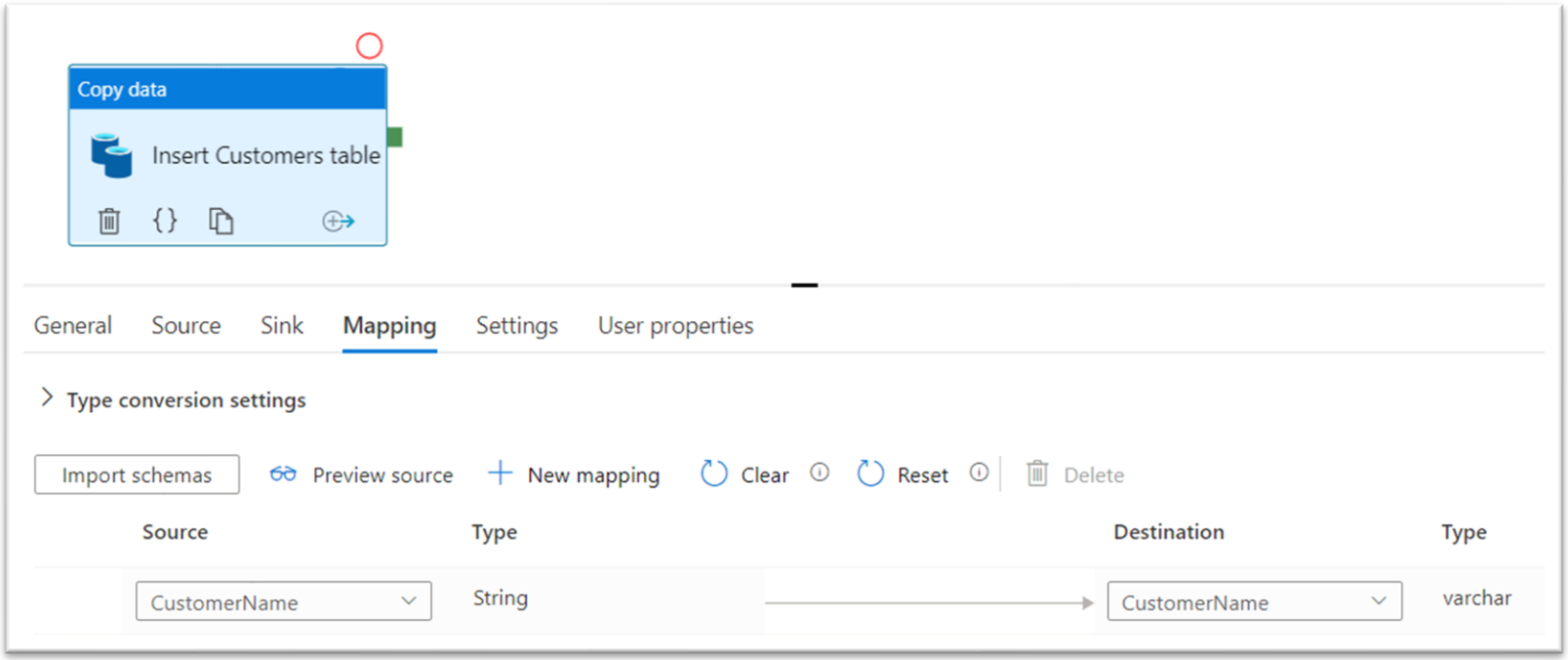
For now, all that is required for this tab is to set the Sink dataset input.

Leave the Stored procedure name dropdown blank, we will use this in subsequent examples.



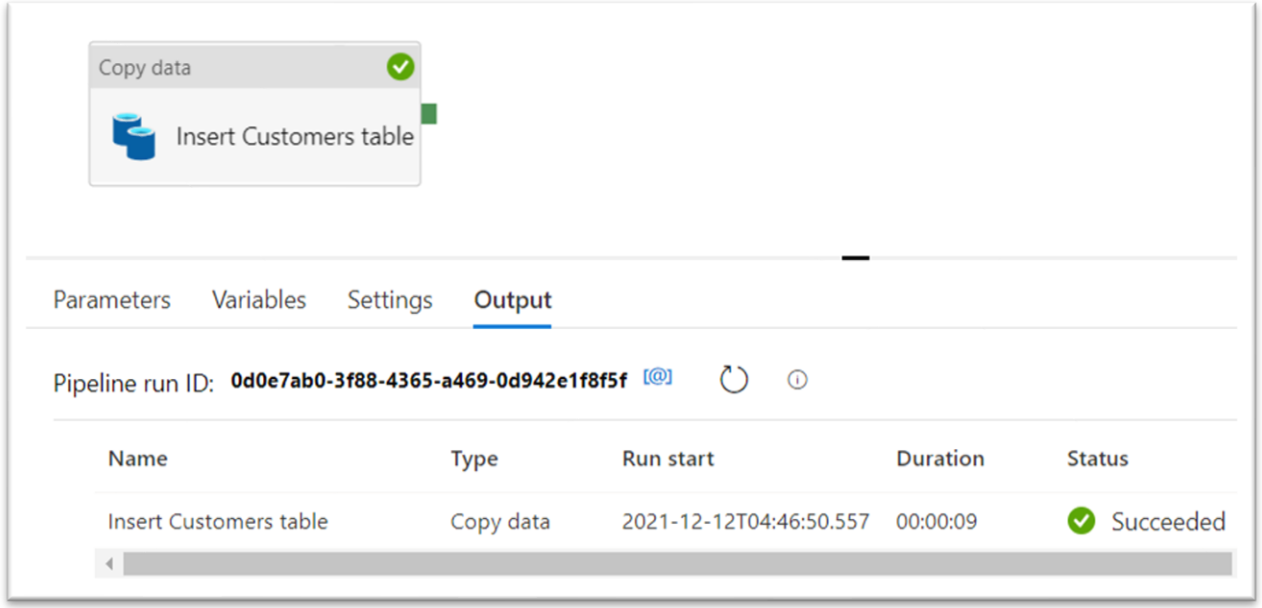
PIPELINE 1

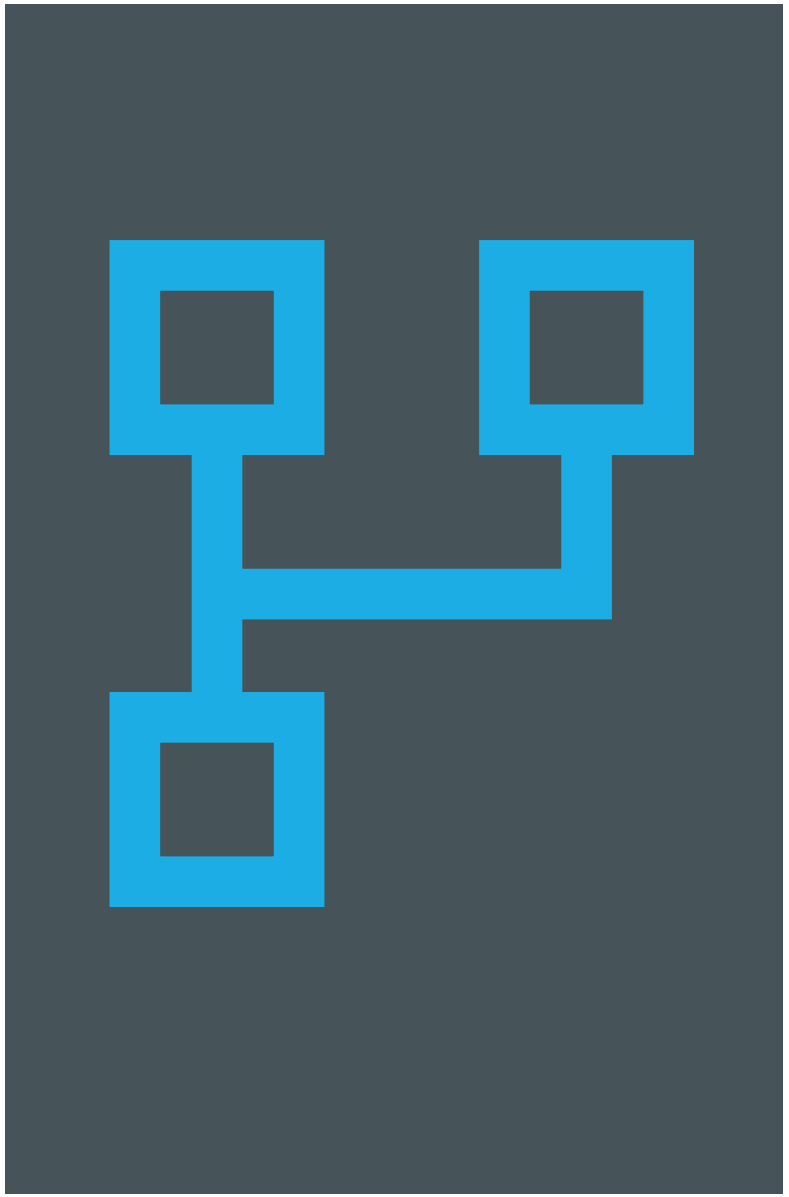
In the Mapping tab, set the Source and Destination drop downs to the appropriate columns as shown.



PIPELINE 1

Execute the pipeline and check the data has been inserted into the Customers table.





PIPELINE 2

Next, we are going to insert the UTC date using dynamic content from the pipeline into the InsertDate of the Customers table.

PIPELINE 2

For this pipeline, we will need to create two SQL Server objects:

1) User-Defined Table Type
2) Stored Procedure

A **user-defined table type** is a user-defined type that represents the definition of a table structure. We can use user-defined table type to declare table-valued parameters for stored procedures or functions.

The stored procedure will reference the user-defined table type.

PIPELINE 2

Create the following user-defined table type. This UDT matches the schema of the customer text files.

```
CREATE TYPE demo.CustomersTableType AS TABLE(
    CustomerName VARCHAR(100) NOT NULL
```

PIPELINE 2

Then create a stored procedure that uses the user-defined table type as a parameter.

We will use these objects in our pipeline.

```
CREATE PROCEDURE demo.SpInsertCustomersTable
@pCustomersTableType demo.CustomersTableType READONLY,
@pInsertDate DATETIME

AS
BEGIN

INSERT INTO demo.Customers (InsertDate, CustomerName)
SELECT  @pInsertDate, CustomerName
FROM    @pCustomersTableType;

END;
```

PIPELINE 2

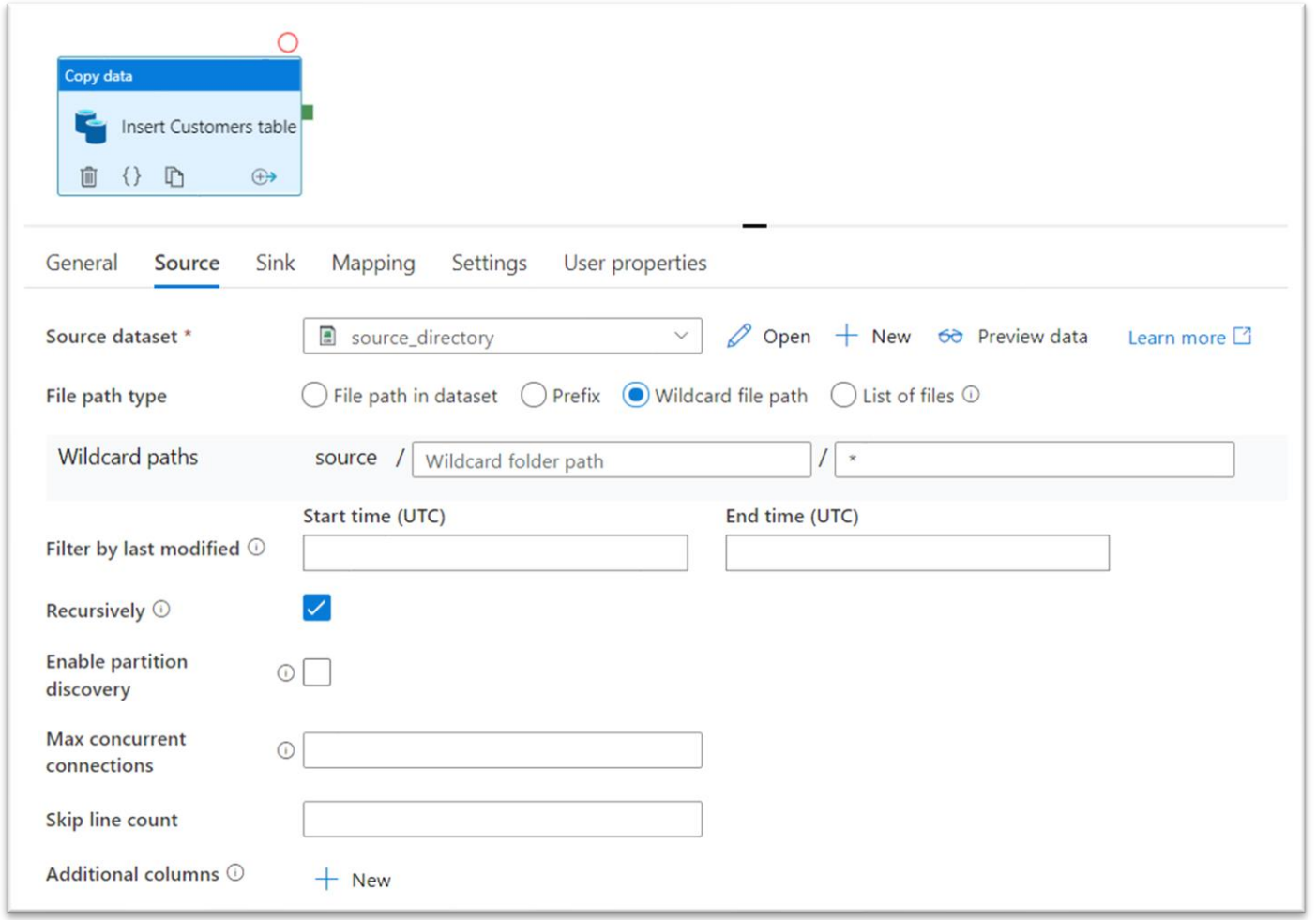
No changes need to be done to our current datasets.

Next, we will add the Copy data activity in our new pipeline.

PIPELINE 2

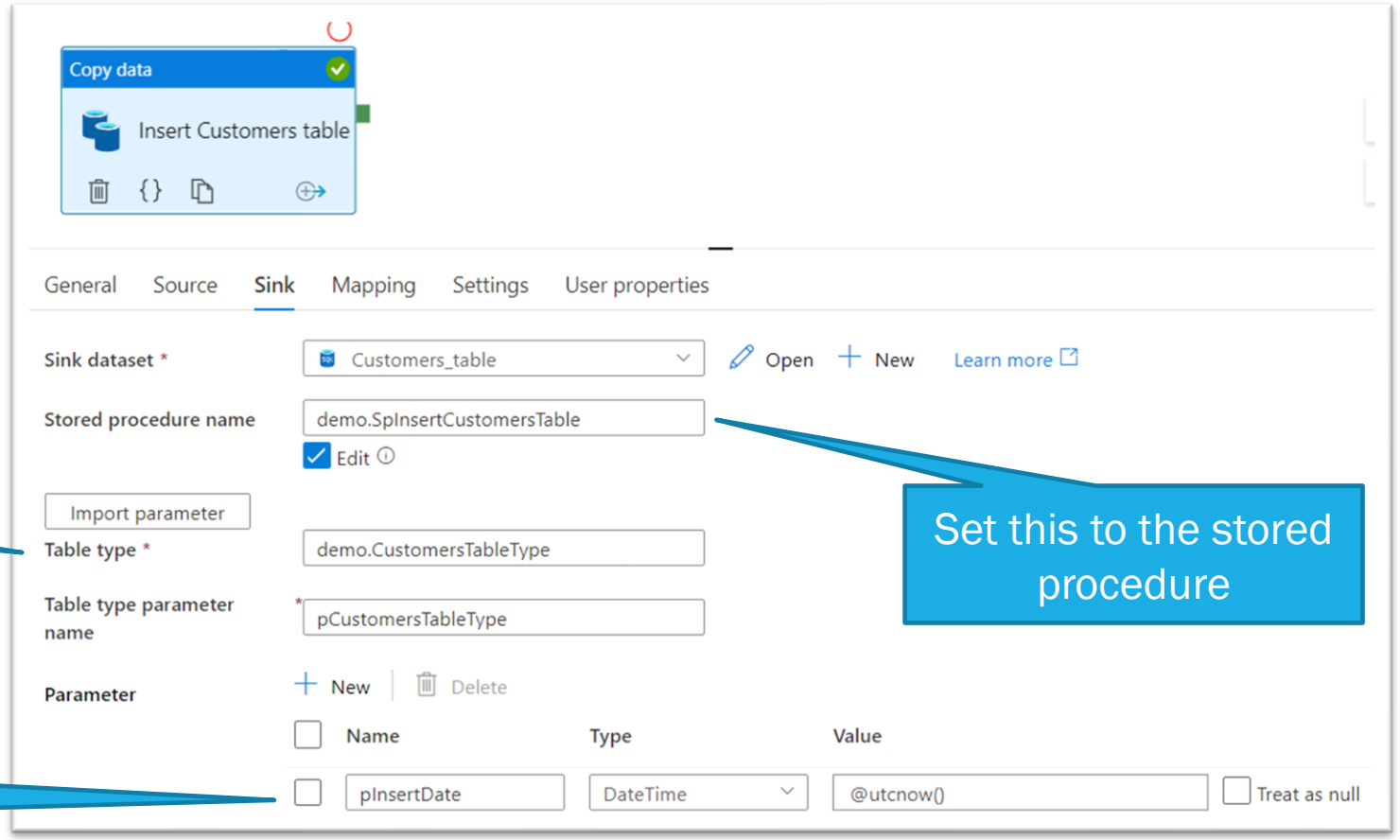
Insert a Copy data activity and set the Source tab properties as shown.

These are the same properties we used in the previous pipeline's Copy data activity.



PIPELINE 2

Next, set the Sink tab properties as shown.



Ensure table type uses a two-part naming convention!

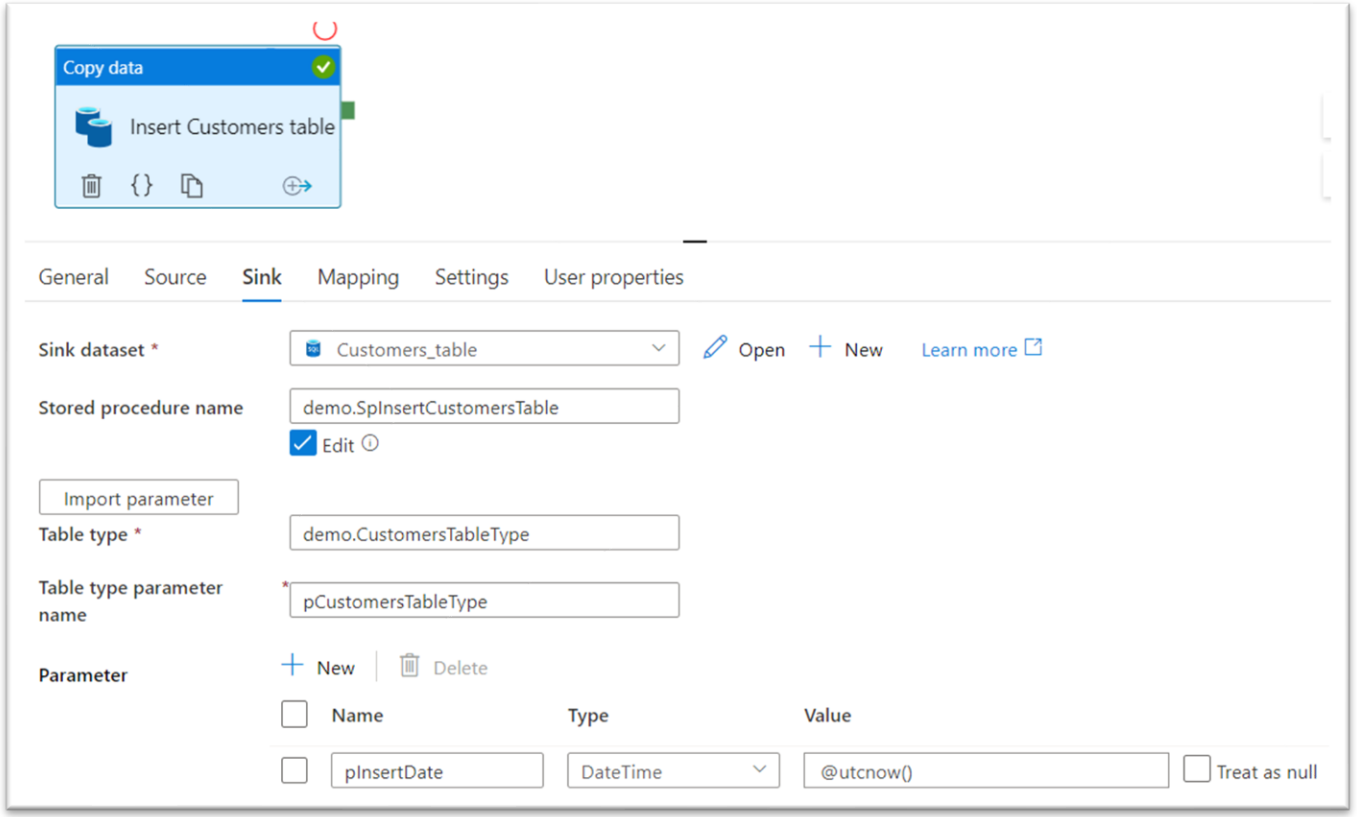
Set this to the stored procedure

Value is @utcnow()

PIPELINE 2

We are now passing the coordinated universal time to the parameter pInsertDate.

The schema of the table Customers matches the UDT we selected in the Table type input box.



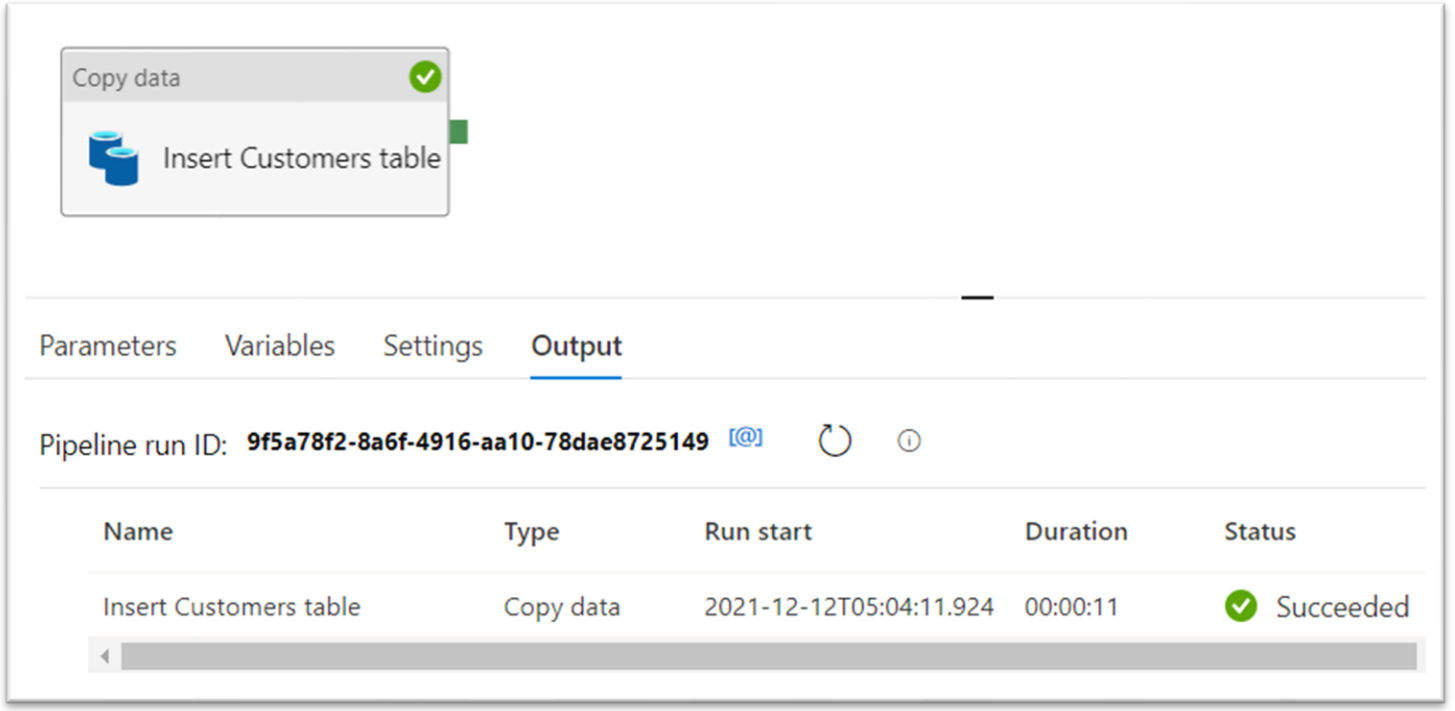
PIPELINE 2

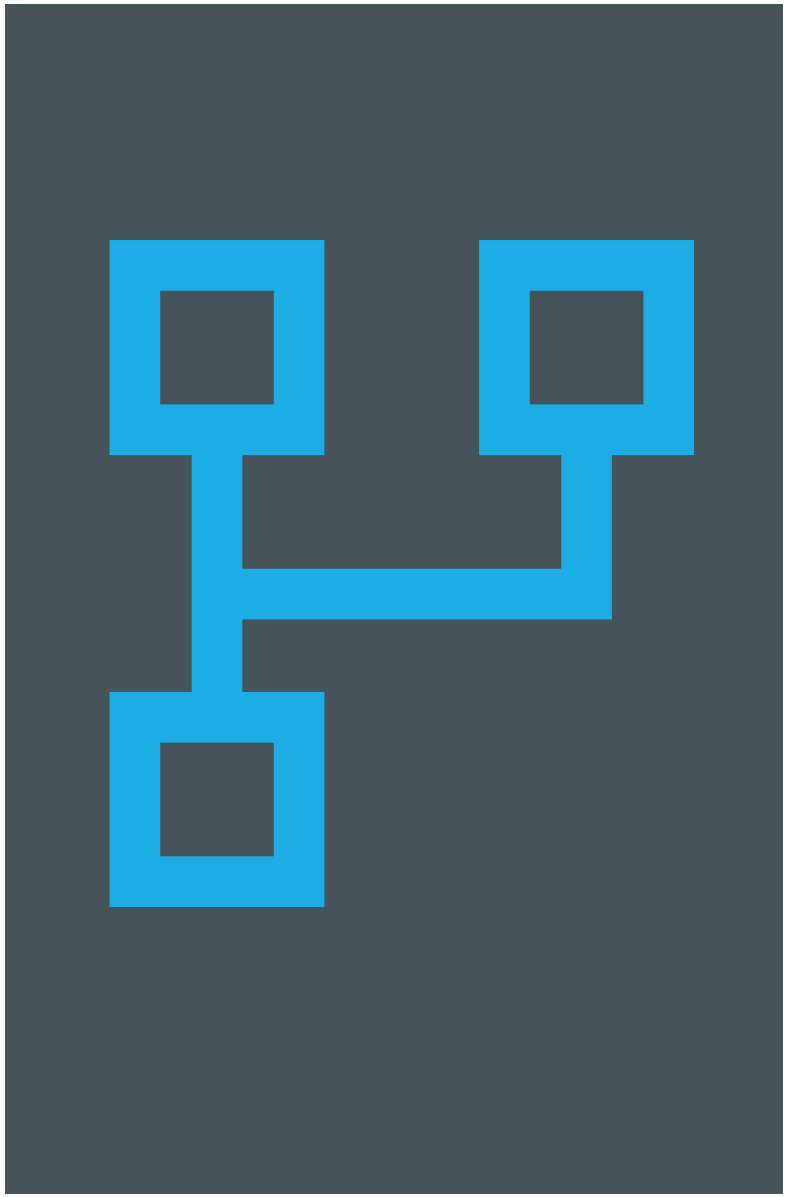
If you do not use the two-part naming convention in the Table type input box, you will get the following error.

Make sure you specify the schema name in the Table type input box!

PIPELINE 2

Run the pipeline and ensure the data was inserted into the Customers table.





PIPELINE 3

Next, we will create a pipeline inserting the metadata from both the file and the pipeline.

PIPELINE 3

We will need to create the following objects with the new fields we want to populate.

1) A new Customers table
2) A new Stored Procedure

We can use the existing user-defined table type that we created, as this matches the schema of the customers files.

The DDL scripts are available in the Git repository located at:

PIPELINE 3

We will also need to add the following:

- Additional activities to loop through each file and read the file's metadata.
- An additional dataset that parameterizes the filename.

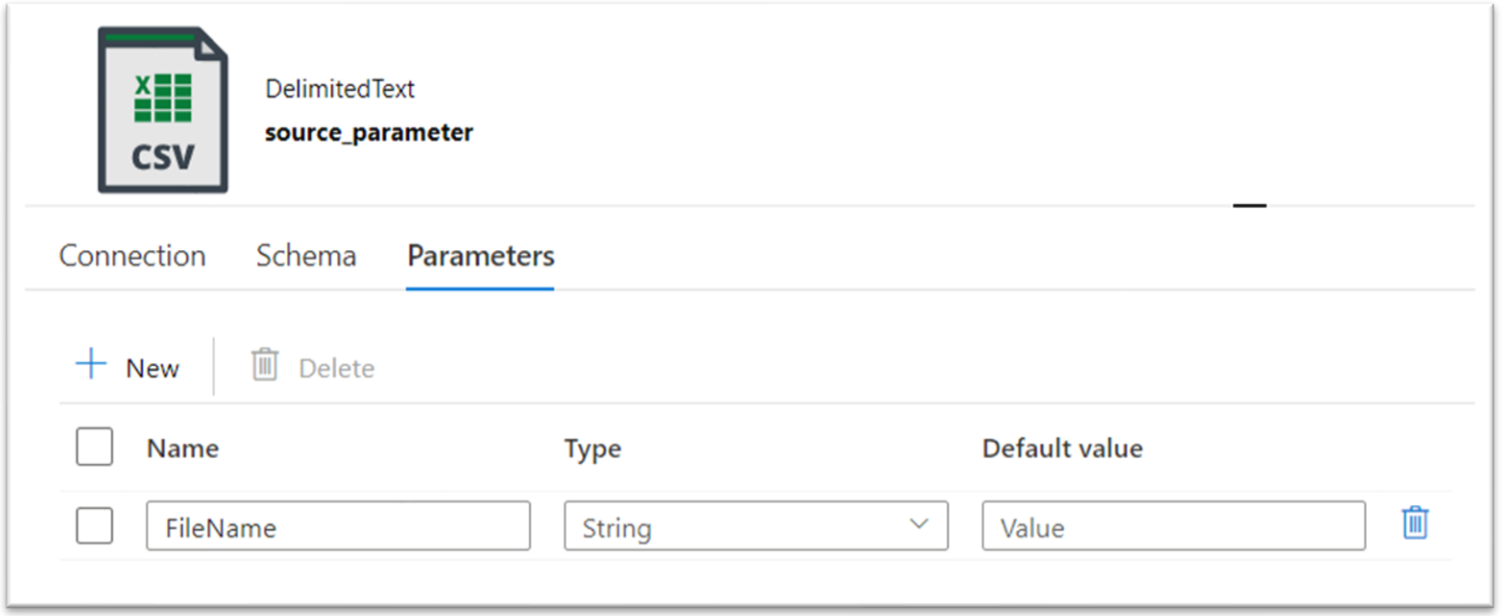
PIPELINE 3

We will need to create a new dataset named source_parameter that has a parameterized file name, and a new dataset named CustomersMetadata_table that references our new table.

Name	Data Store	Type
source_directory	Azure Blob Storage	Format type is DelimitedText
CustomersMetadata_table	Azure SQL Database	References the table CustomersMetadata
source_parameter	Azure Blob Storage	Format type is DelimitedText

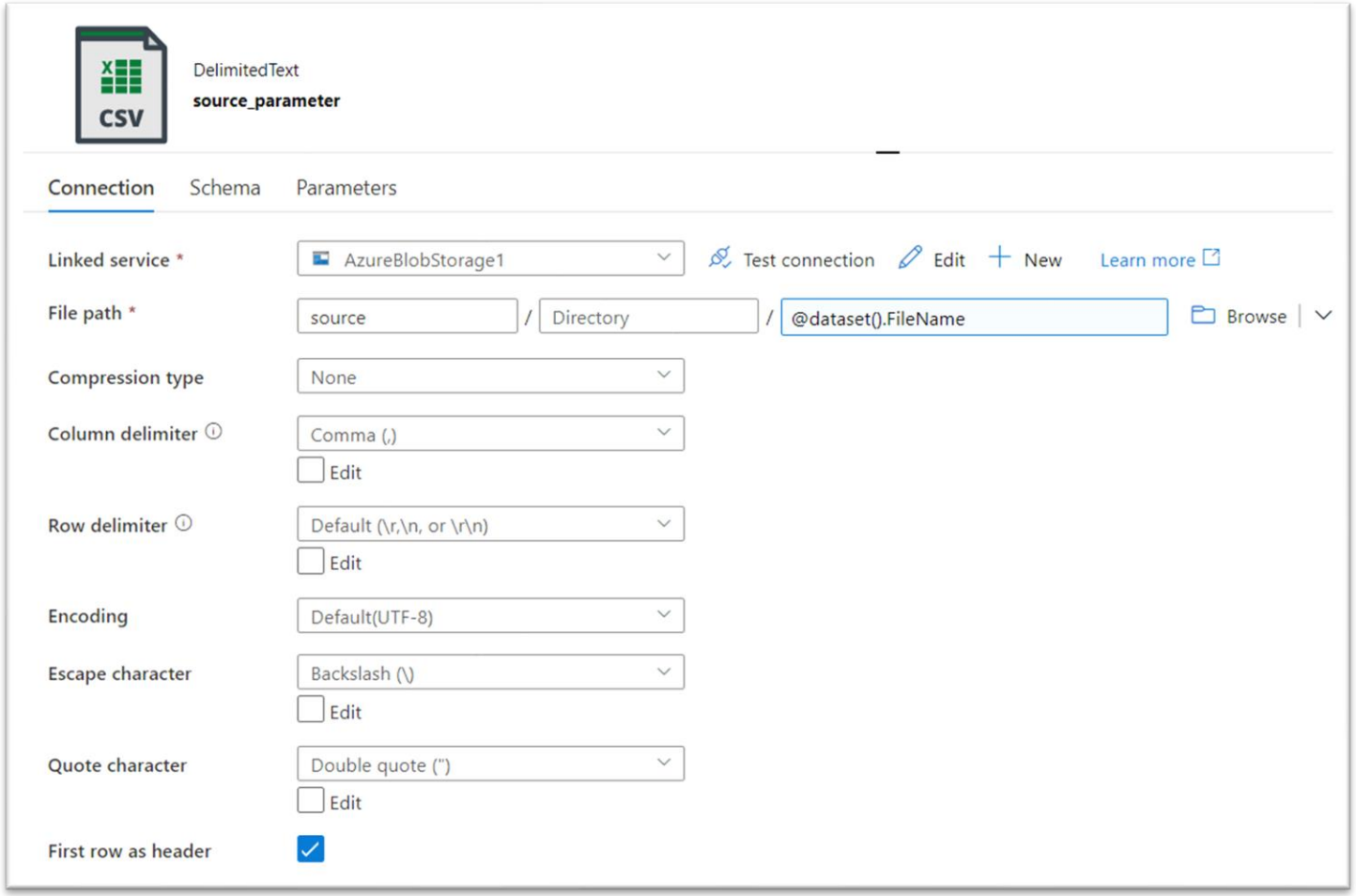
PIPELINE 3

For the new dataset that references the Azure Storage Account source directory, create a parameter named FileName.



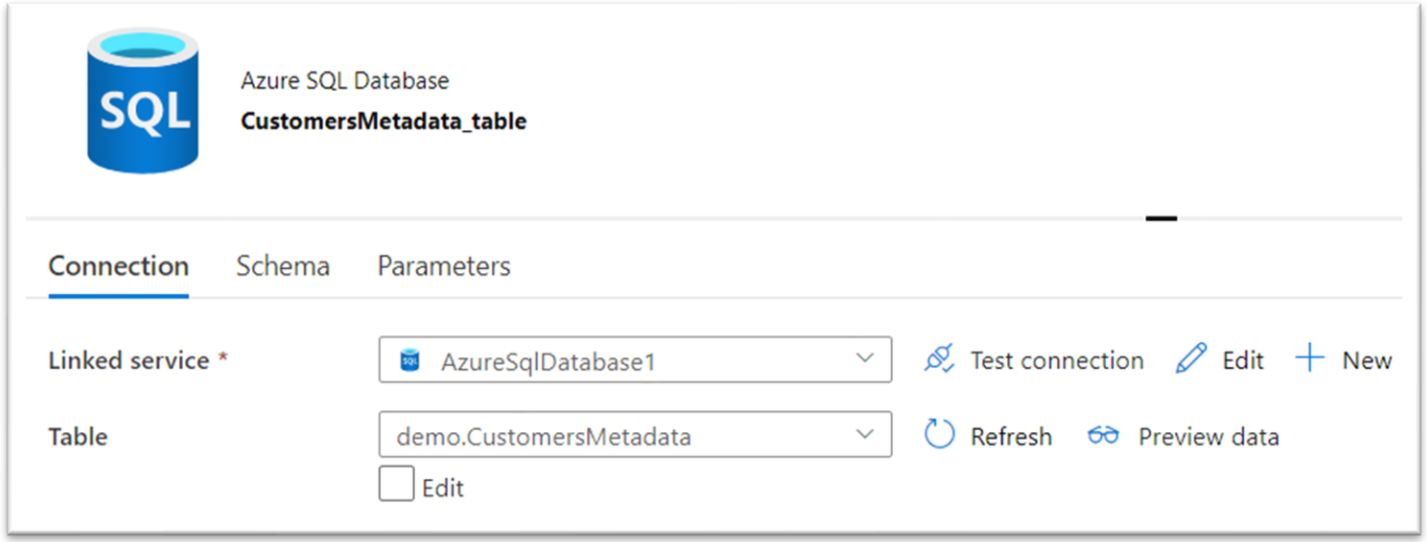
PIPELINE 3

Then reference the parameter in the connection tab of the new dataset. Add the argument using the dynamic content window. Also, ensure you have properly set First row as header.



PIPELINE 3

Create the CustomersMetadata_table dataset referencing the table CustomersMetadata.



PIPELINE 3

Next, we will build the following activities in the pipeline.

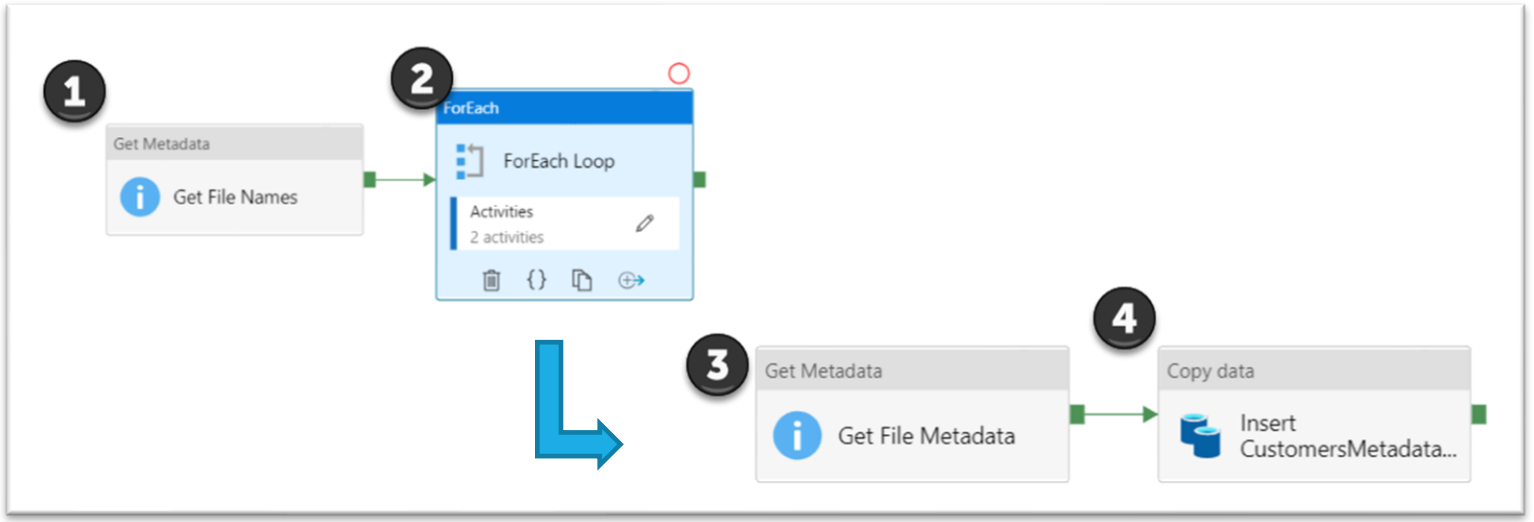
1	Get Metadata	Returns a list of file in the source directory
2	ForEach	Creates a loop

3	Get Metadata	Determine metadata for a single file
4	Copy data	Insert data into Customers table

PIPELINE 3

The activities in the pipeline will look like the following.

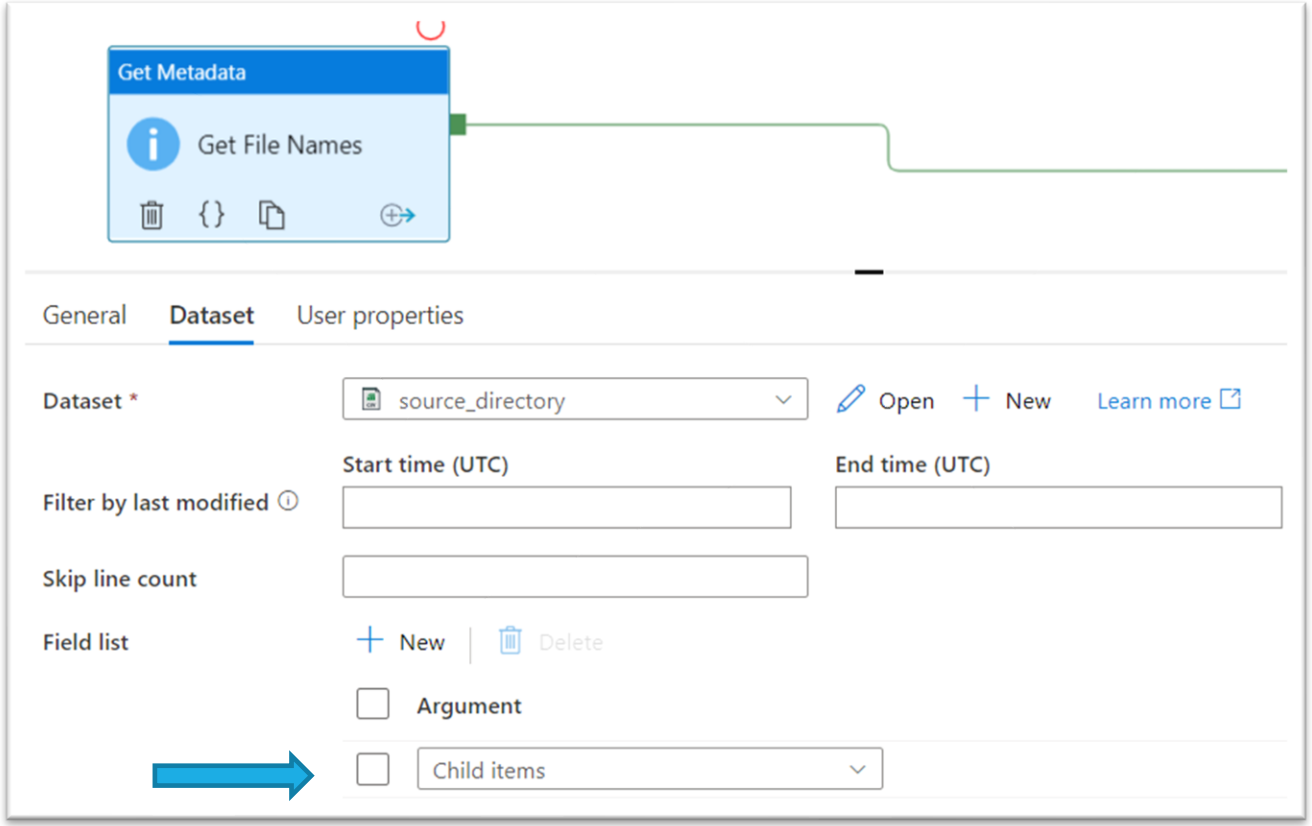
We will step through each of these activities in the following slides.



PIPELINE 3

First, insert a Get Metadata activity into the pipeline and insert a new argument into the Field list and then select Child items.

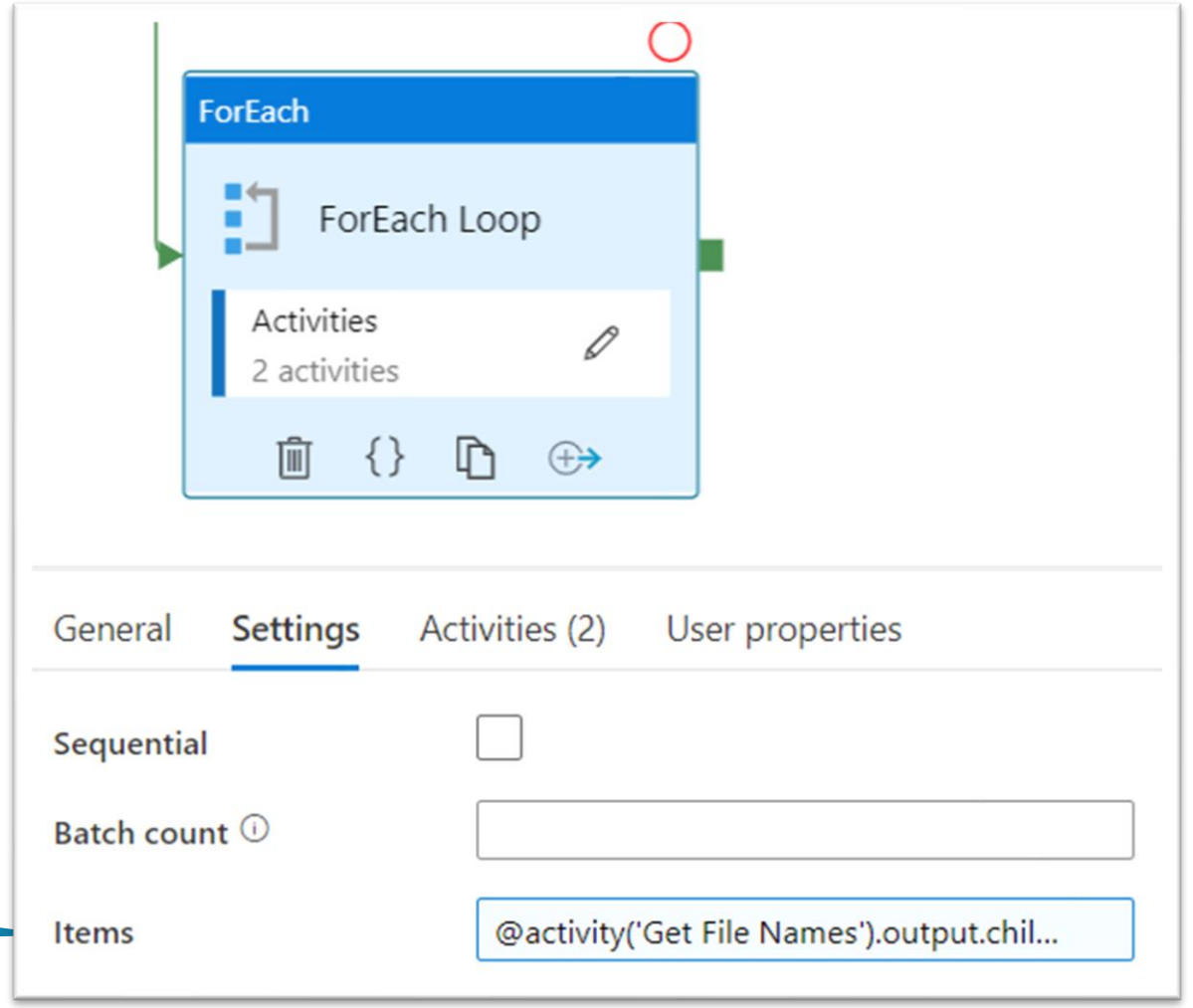
This will return a list of files in the source directory.



PIPELINE 3

Next, insert a ForEach activity into the pipeline and set the Items input box as show.

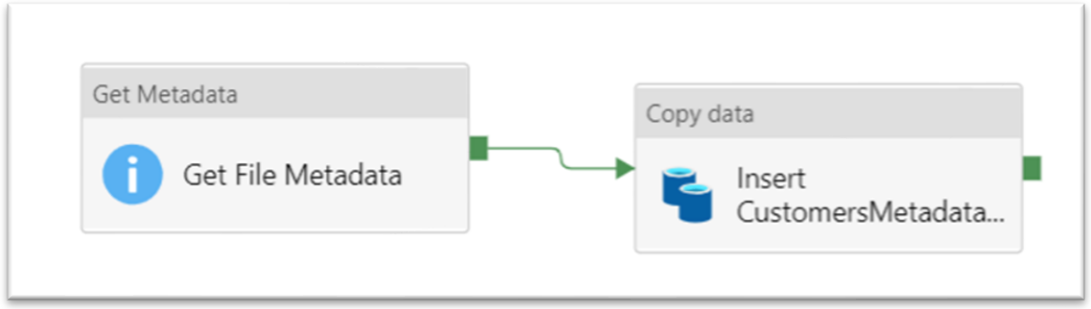
Next, we will insert two activities into the ForEach loop.



@activity('Get File Names').output.childItems

PIPELINE 3

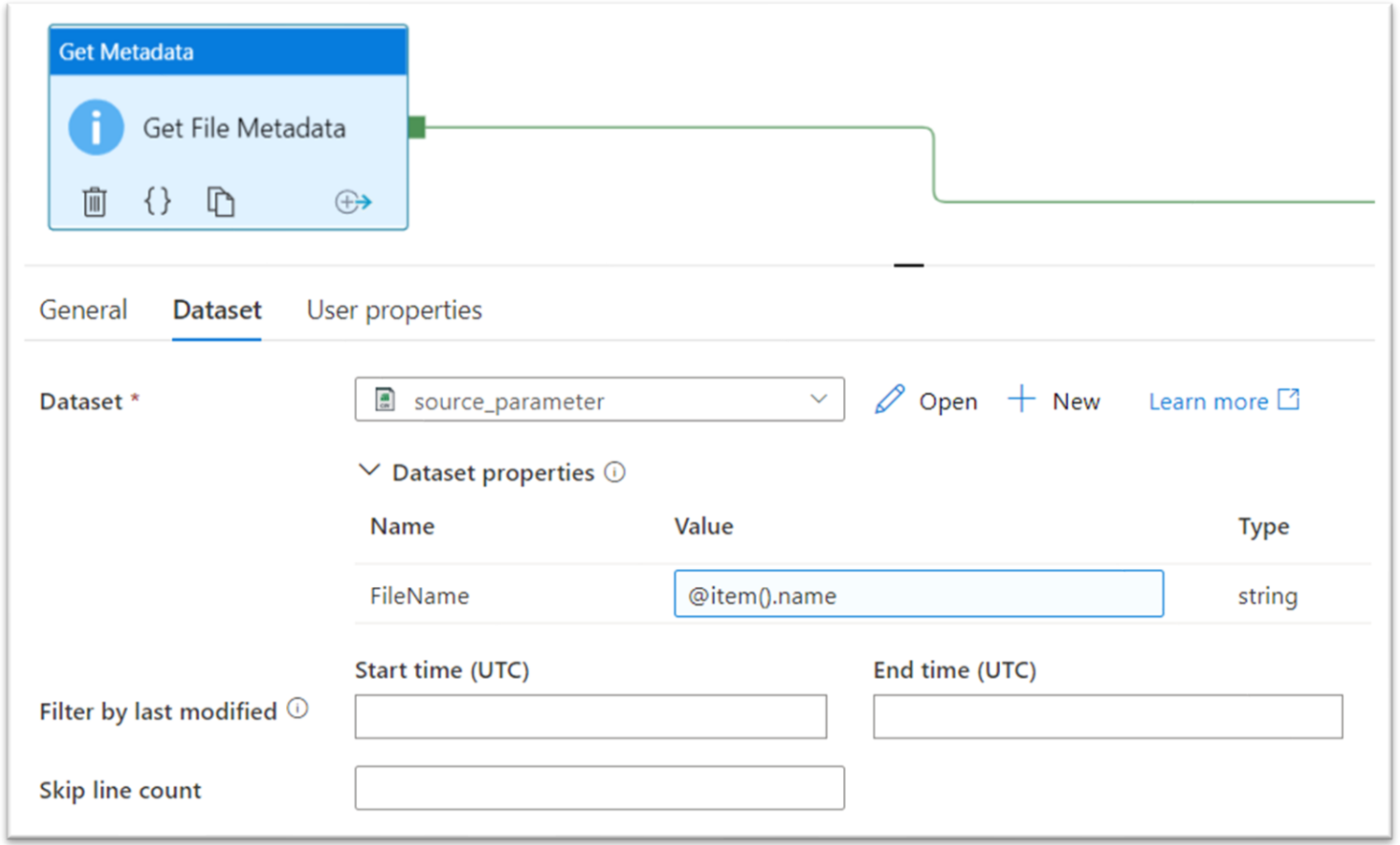
Insert a Get Metadata and a Copy data activity within the ForEach activity.



PIPELINE 3

The Get Metadata activity within the ForEach activity will use the new source_parameter dataset that we created.

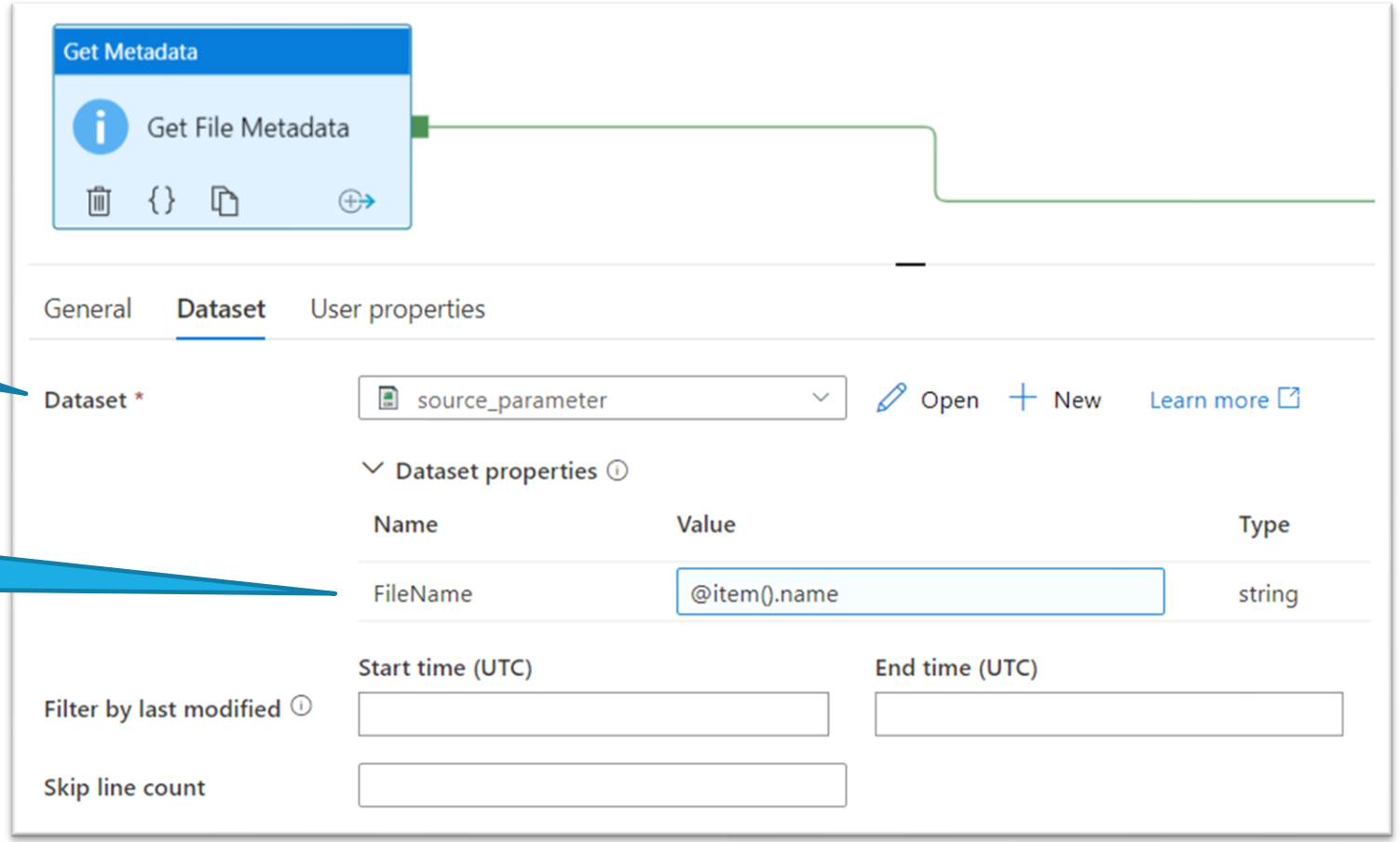
There are several settings that we need to input that we will go over in the next slides.



PIPELINE 3

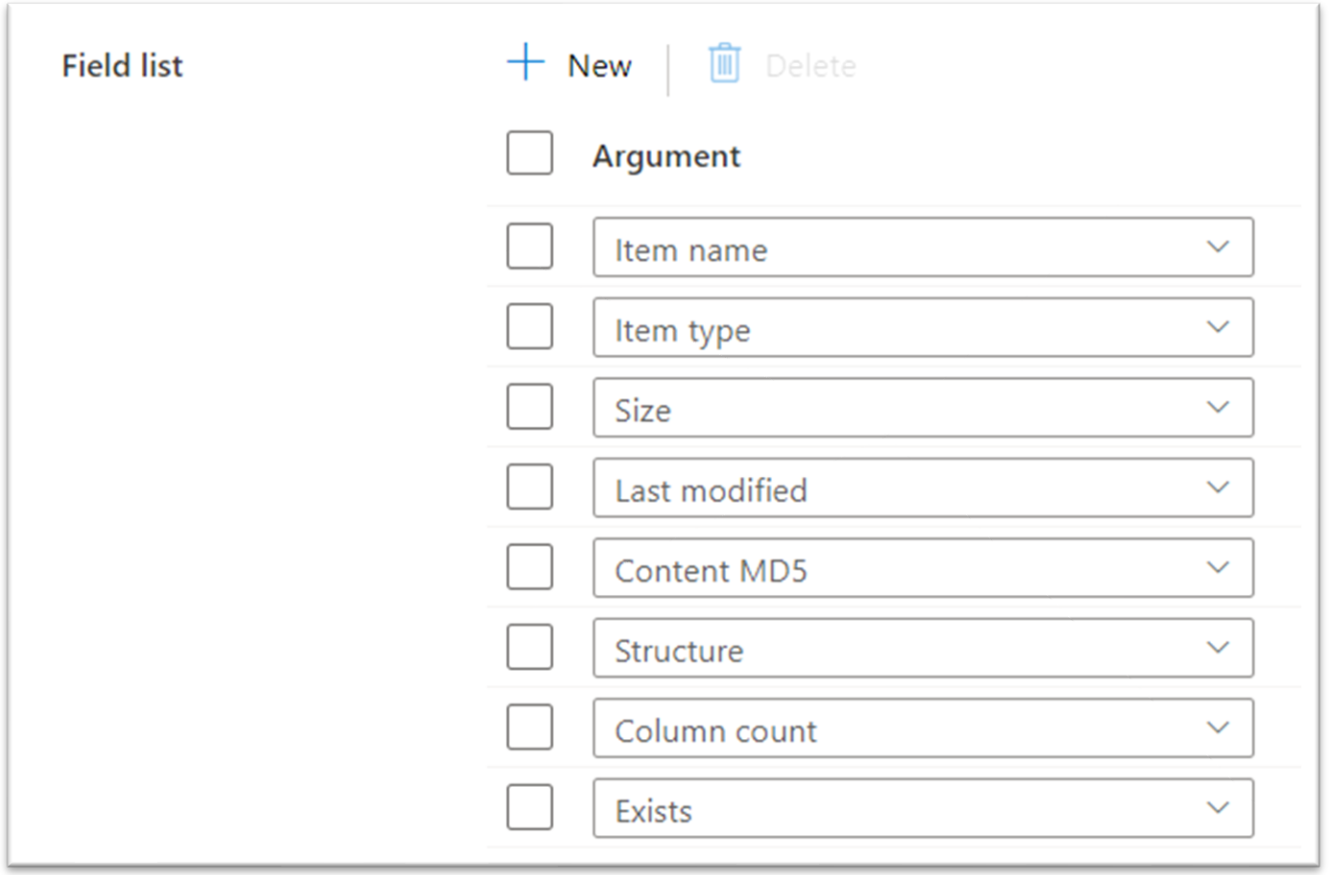
Set the dataset to the new source_parameter dataset we created

The FileName parameter argument will be @item().name



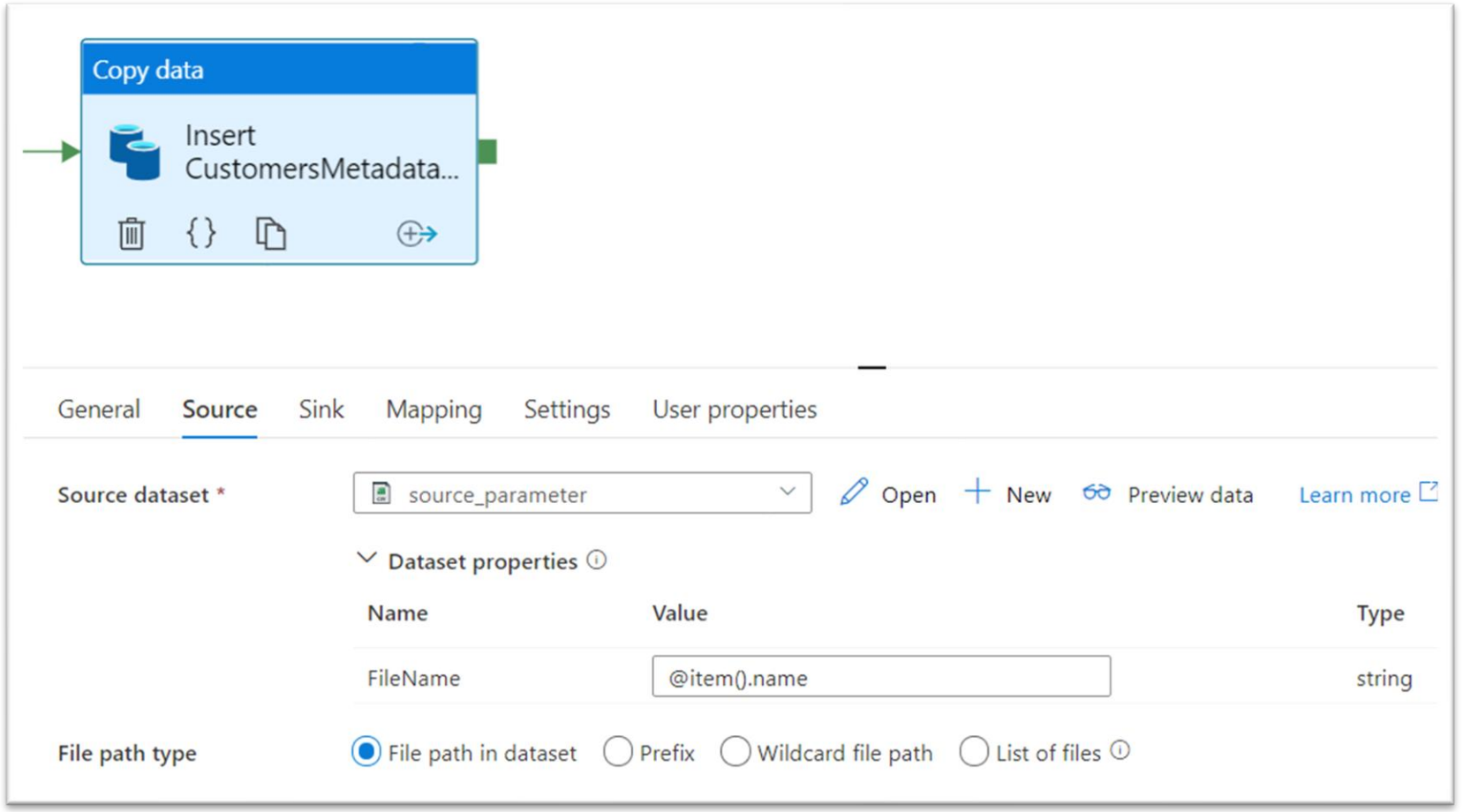
PIPELINE 3

Scroll down in the Get Metadata activity and select all the available fields available as shown. Note here there is no created argument, as described in the Microsoft documentation.



PIPELINE 3

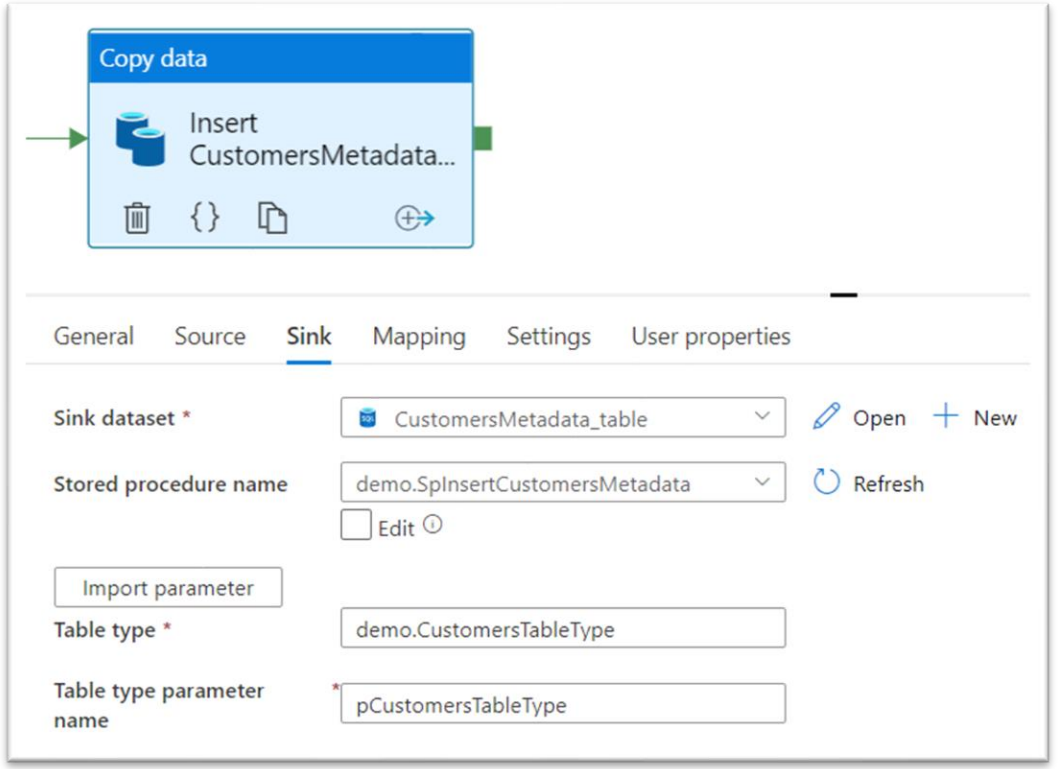
The Copy data activity has the following options in the Source tab. Use the source_parameter dataset and insert the FileName value as shown.



PIPELINE 3

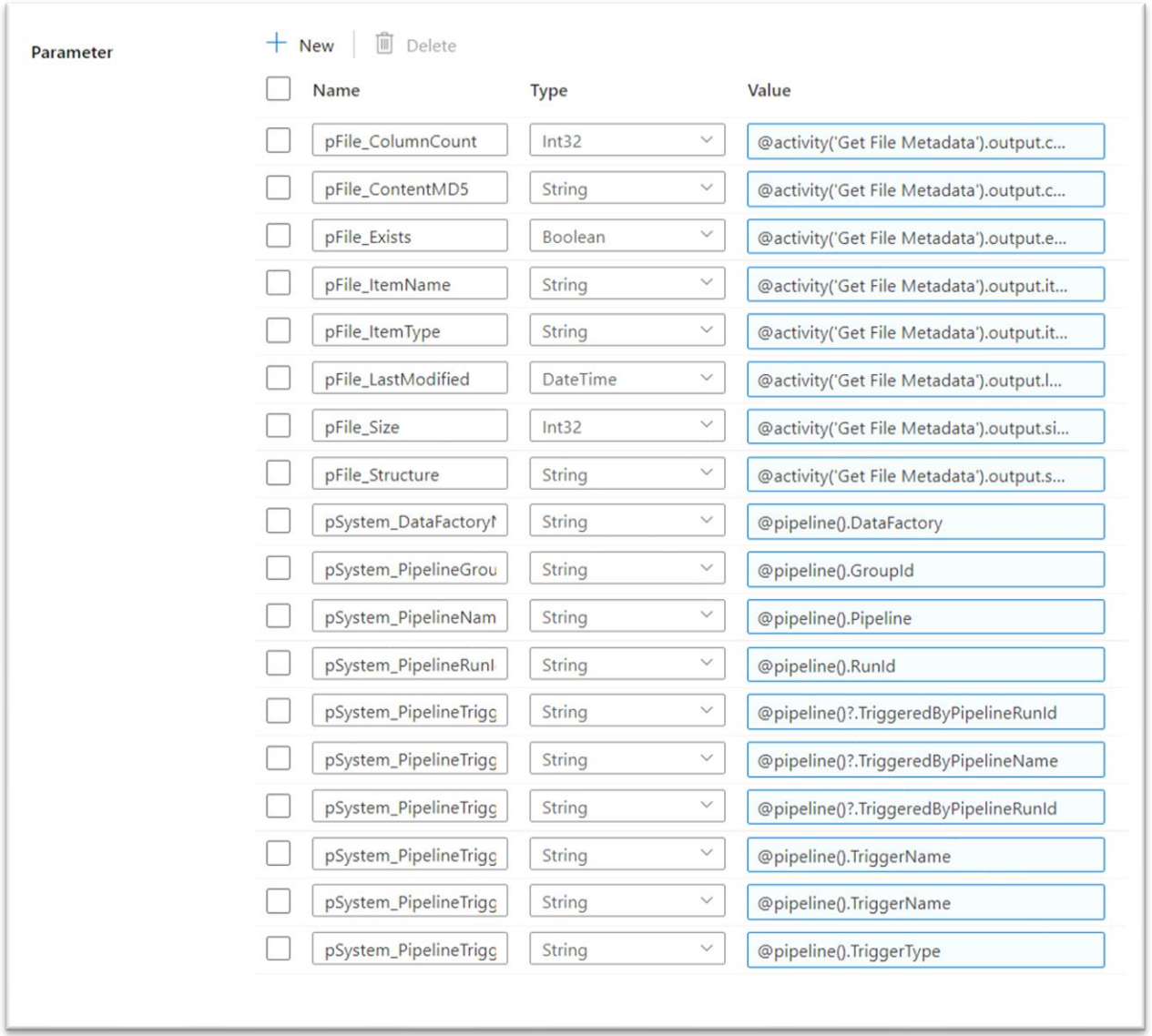
Set the Sink dataset to the CustomersMetadata table we created. Insert the Stored procedure name and select Import parameter.

Ensure your table type has a two-part naming convention! It does not default to the two-part name.



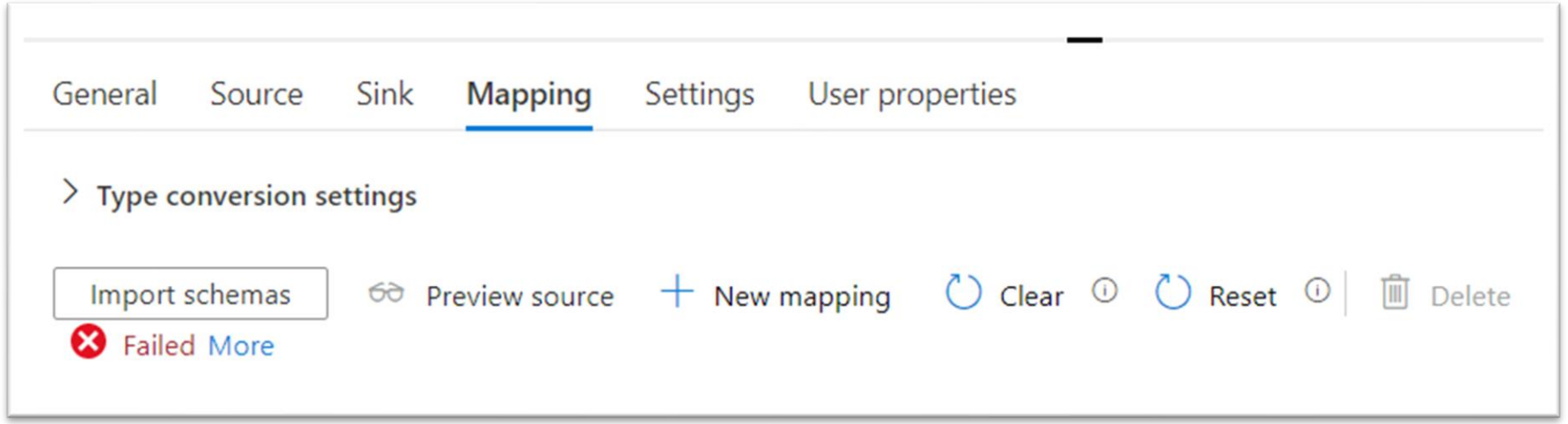
PIPELINE 3

In the Sink tab of the Copy data activity, we will need to perform the same operations as we did in pipeline 2, but this time we insert the metadata values for the parameters.



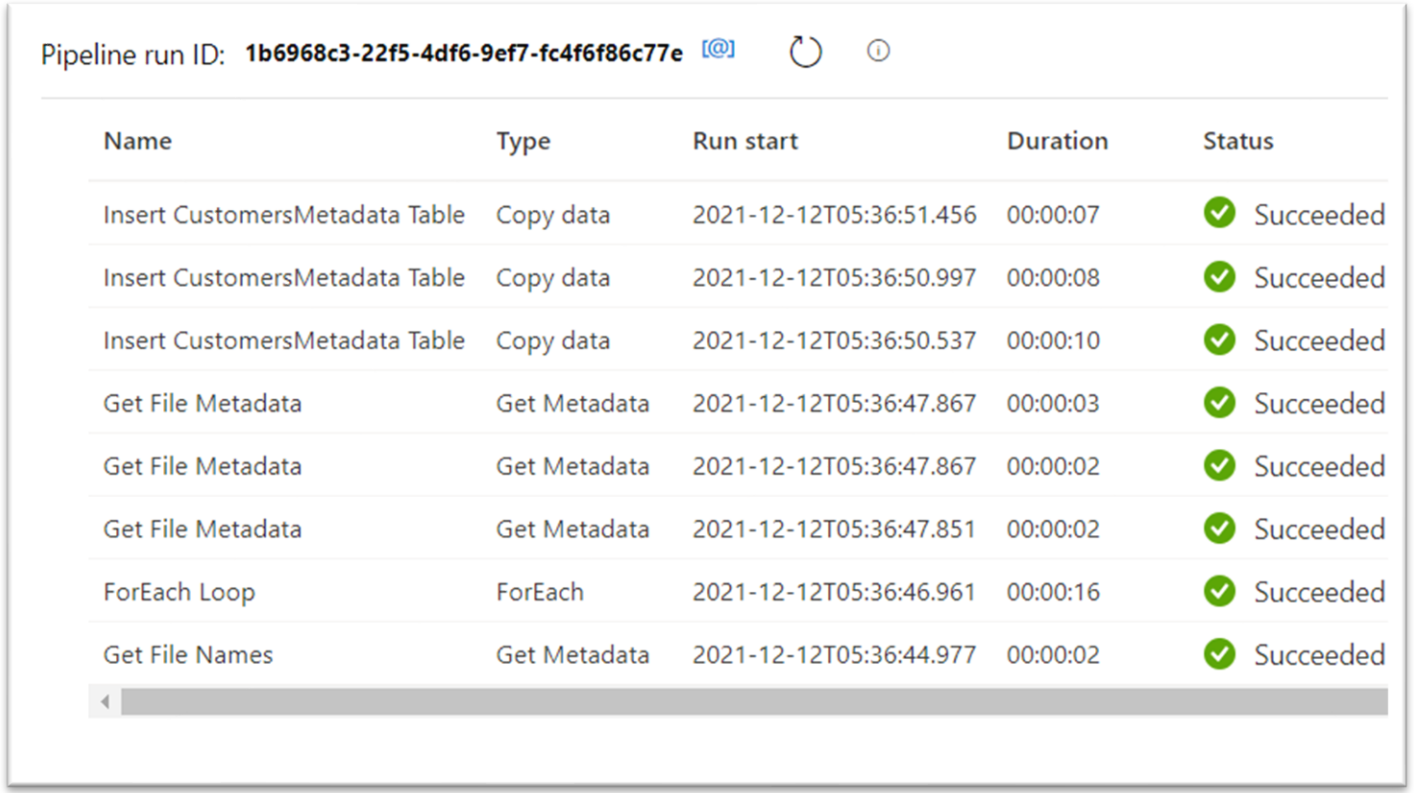
PIPELINE 3

In the Mapping tab, the red Failed error is perfectly fine here.



PIPELINE 3

Execute the pipeline and check the data has been inserted into the CustomersMetadata table.



Name	Type	Run start	Duration	Status
Insert CustomersMetadata Table	Copy data	2021-12-12T05:36:51.456	00:00:07	Succeeded
Insert CustomersMetadata Table	Copy data	2021-12-12T05:36:50.997	00:00:08	Succeeded
Insert CustomersMetadata Table	Copy data	2021-12-12T05:36:50.537	00:00:10	Succeeded
Get File Metadata	Get Metadata	2021-12-12T05:36:47.867	00:00:03	Succeeded
Get File Metadata	Get Metadata	2021-12-12T05:36:47.867	00:00:02	Succeeded
Get File Metadata	Get Metadata	2021-12-12T05:36:47.851	00:00:02	Succeeded
ForEach Loop	ForEach	2021-12-12T05:36:46.961	00:00:16	Succeeded
Get File Names	Get Metadata	2021-12-12T05:36:44.977	00:00:02	Succeeded

Example metadata from a scheduled trigger execution.

Column	Value
InsertDate	12/5/2021 18:18
CustomerName	Alan Turing
File_ItemName	Customers_1.txt
File_ItemType	File
File_Size	48
File_LastModified	12/2/2021 0:48
File_ContentMD5	n05np0tEsV8zCIBKcC4wHg==
File_Structure	System.Collections.Generic.List`1[System.Object]
File_ColumnCount	1
File_Exists	1
System_DataFactoryName	adf-datafactory-demo-01
System_PipelineName	Insert_Customers_Metadata
System_PipelineRunId	87fae8e9-492b-40b1-b9d0-d581989fa605
System_PipelineTriggerType	ScheduleTrigger
System_PipelineTriggerId	NULL
System_PipelineTriggerName	ExampleTrigger
System_PipelineTriggerTime	ExampleTrigger
System_PipelineGroupID	87fae8e9-492b-40b1-b9d0-d581989fa605
System_PipelineTriggeredByPipelineName	NULL
System_PipelineTriggerByPipelineRundId	NULL

Example metadata from a debug execution.

Column	Value
InsertDate	12/3/2021 17:31
CustomerName	Thales of Miletus
File_ItemName	Customers_1.txt
File_ItemType	File
File_Size	70
File_LastModified	12/3/2021 12:34
File_ContentMD5	YXwu44TeGEE14g03inSDwg==
File_Structure	System.Collections.Generic.List`1[System.Object]
File_ColumnCount	1
File_Exists	1
System_DataFactoryName	adf-datafactory-demo-01
System_PipelineName	Insert_Customers_Metadata
System_PipelineRunId	afde1fbf-32d8-47e0-b453-2d4340cc346a
System_PipelineTriggerType	Manual
System_PipelineTriggerId	NULL
System_PipelineTriggerName	Sandbox
System_PipelineTriggerTime	Sandbox
System_PipelineGroupID	afde1fbf-32d8-47e0-b453-2d4340cc346a
System_PipelineTriggeredByPipelineName	NULL
System_PipelineTriggerByPipelineRundId	NULL

ADDITIONAL

Additionally, we can copy the customer files to a destination directory and append a date to the filename.

Create a parameterized dataset like the source_parameter dataset we created. Then use dynamic content to populate the filename.

See the following Microsoft documentation:

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions

ADDITIONAL

If you want to populate a table with only the metadata and not the data in the text files, change the Copy data activity to a Stored procedure activity.

See the following Microsoft documentation:

https://docs.microsoft.com/en-us/azure/data-factory/transform-data-using-stored-procedure

THE END