Lab 5 – Create a reusable pipeline

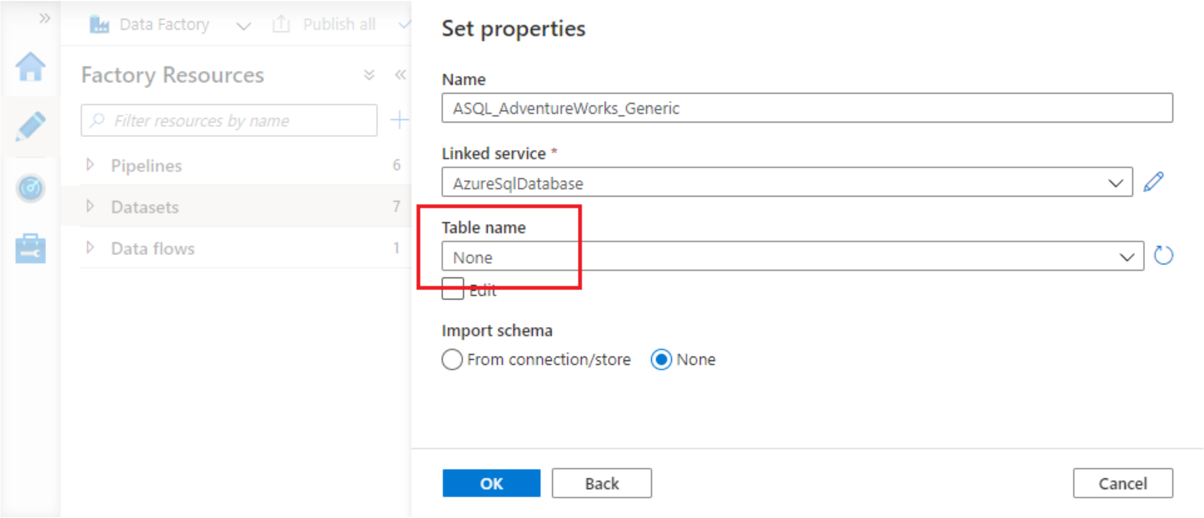
The pipeline you built in Lab 2 extracted data from a database table to a file in the data lake. In Lab 4.2 you cloned that pipeline to perform the same task for a different table. In this lab you will use dataset parameters and ADF pipeline expressions to make a reusable pipeline – a single pipeline that can perform this task for *any* table.

Lab 5.1 – Create a generic SQL table dataset

Start by creating a new Azure SQL Database dataset, in the same way as in lab 2.2.

1. In the "Factory resources" sidebar, click the "+" button to the right of "Filter resources by name", then choose "Dataset". Select the "Azure SQL Database" data store and click "Continue".

2. Name the dataset "ASQL_AdventureWorks_Generic", to make the dataset's purpose clear. Choose your Azure SQL Database from the "Linked service" dropdown. This time however, **do not** choose any table from the "Table name" dropdown – just click "OK".



Set properties

Name
ASQL_AdventureWorks_Generic

Linked service *
AzureSqlDatabase

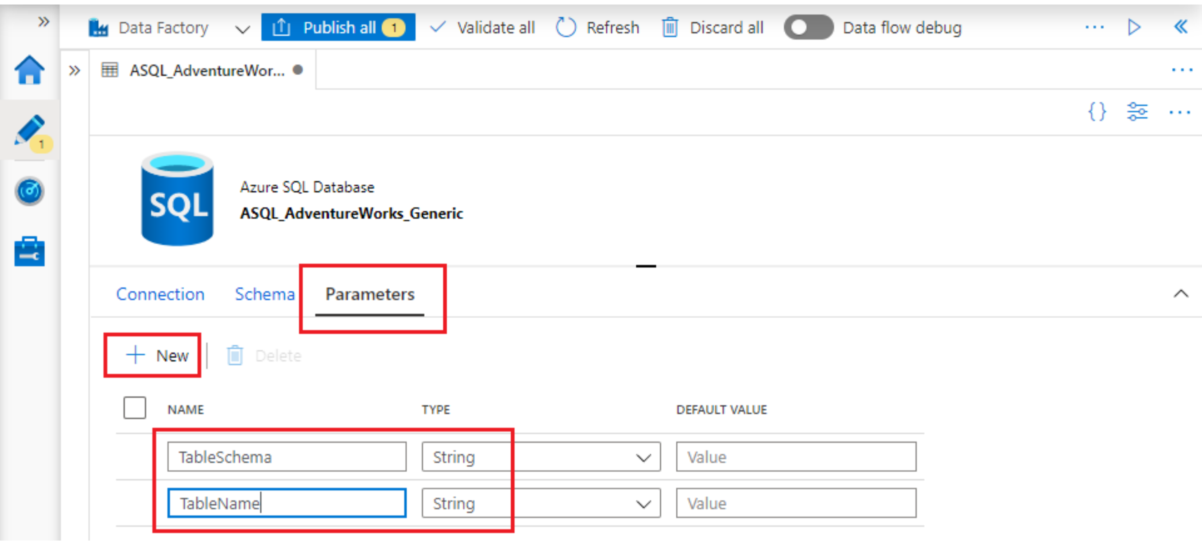
Table name
None

□ Edit

Import schema
○ From connection/store ● None

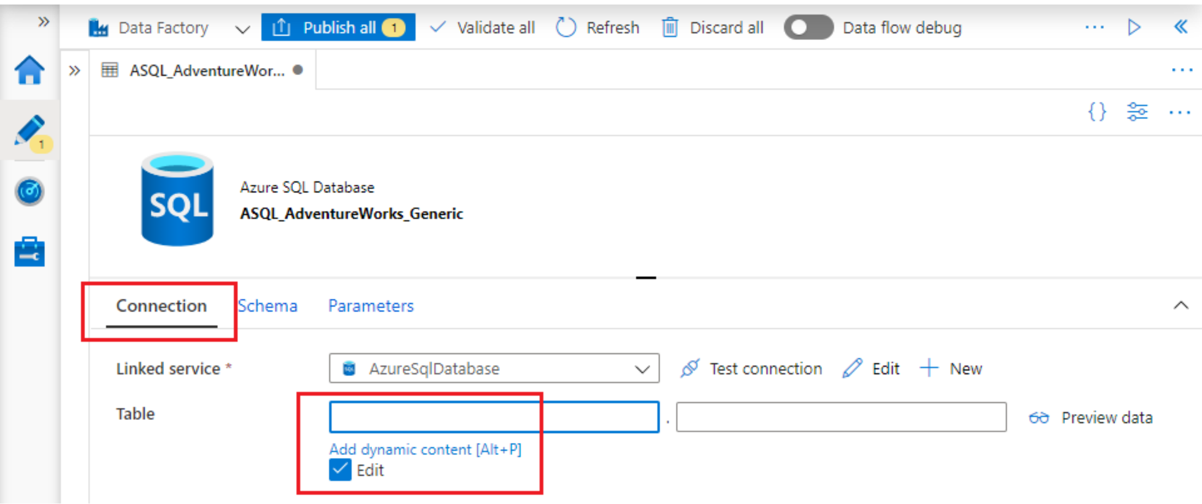
3. The dataset opens in the ADF UX. Close its "Properties" blade using the slider icon in the top right, then select the "Parameters" tab in the configuration pane.

4. Use the "+ New" button to add a new parameter. Name it "TableSchema" and ensure its type is "String". Add a second parameter called "TableName", also of type "String".



NAME	TYPE	DEFAULT VALUE
TableSchema	String	Value
TableName	String	Value

5. On the “Connection” tab, under the “Table” dropdown, tick the “Edit” checkbox. The dropdown is replaced by two text fields. Click into the first – a link with the text “Add dynamic content [Alt + P]” appears below the field.

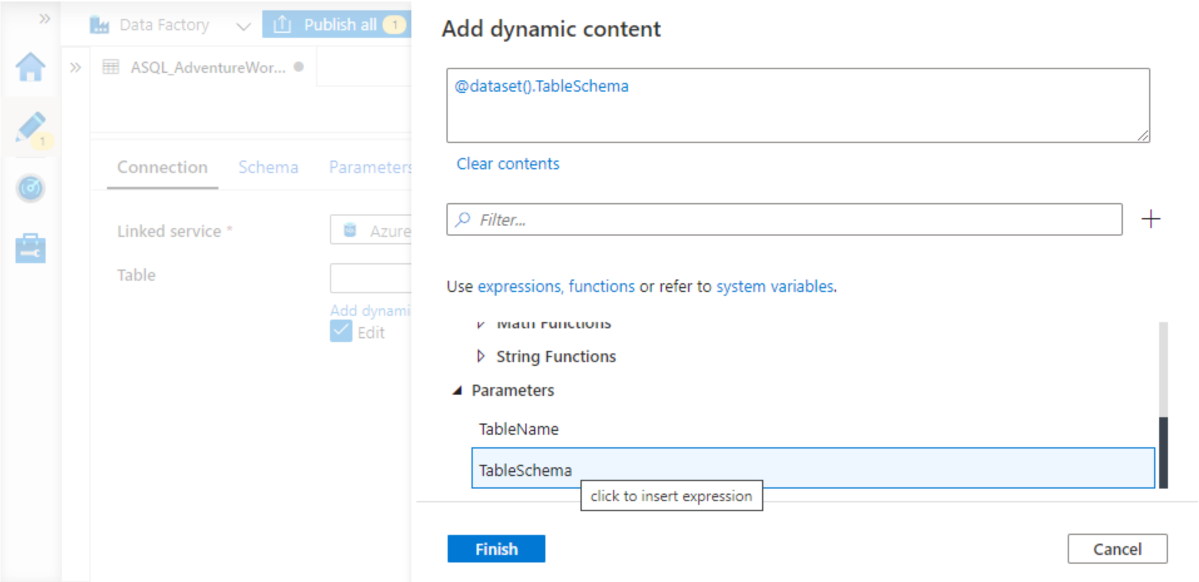


Linked service * AzureSqlDatabase

Table

☑ Edit

6. Click the “Add dynamic content” link to open the pipeline expression builder. The builder blade appears on the right of the ADF UX window. Scroll down the list of functions to find the “Parameters” section, then click on “TableSchema” to select the parameter. The expression “@dataset().TableSchema” appears in the expression pane – click “Finish”.



Add dynamic content

@dataset().TableSchema

Clear contents

Filter...

Use expressions, functions or refer to system variables.

7. Repeat the process for the second field:
    - Click into the field
    - Open the expression builder
    - Select the "TableName" parameter – the expression "@dataset().TableName" appears in the expression pane.
    - Click "Finish"

The dataset you have created contains no hard-coded table name, but instead uses its TableSchema and TableName parameters to specify the parts of a table's name. Different values can be provided for these parameters at runtime, enabling the same dataset to represent many different tables.

Lab 5.2 – Create a generic data lake dataset

In the same way that you created a dataset to represent any AdventureWorks table, now create a generic dataset to represent a data lake file.

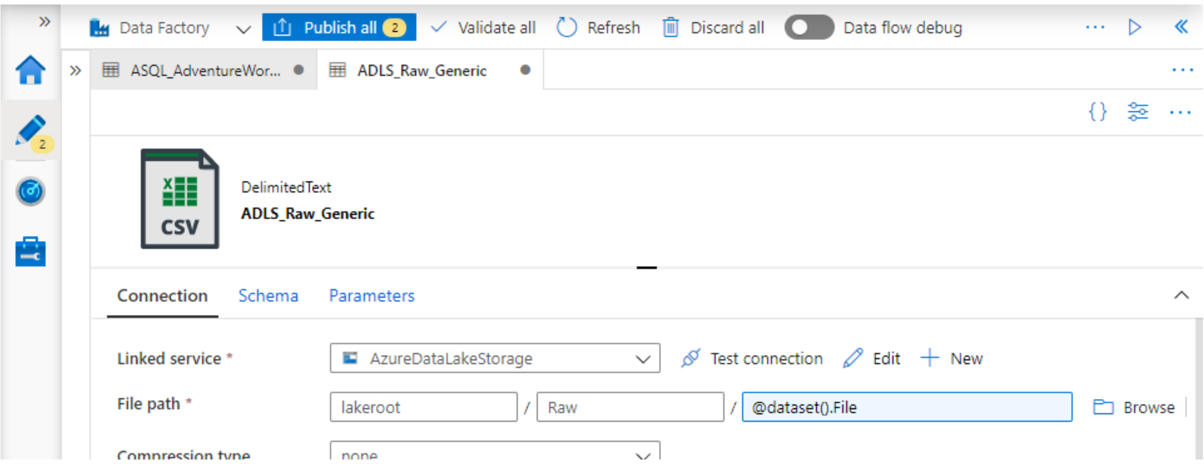
1. Create a new dataset for data store "Azure Data Lake Storage Gen2" with file format "DelimitedText".

2. On the "Set properties" blade, name the dataset "ADLS_Raw_Generic" and select your Azure Data Lake Storage linked service. Tick "First row as header", but leave everything else as is. Click "OK".

3. Create a parameter for the new dataset of type "String". Name it "File".

4. On the "Connection" tab, set the three parts of the "File path" as follows:
    - Set "File System" to "lakeroot" (note that data lake file names are case sensitive)
    - Set "Directory" to "Raw"
    - Use the expression builder to set the "File" field to use the dataset parameter. (If you type the expression into the field directly the ADF UX will interpret the value as a string literal).

5. Save/publish your changes.



DelimitedText
ADLS_Raw_Generic

Connection | Schema | Parameters

Linked service * AzureDataLakeStorage — Test connection — Edit — New

File path * lakeroot / Raw / @dataset().File — Browse

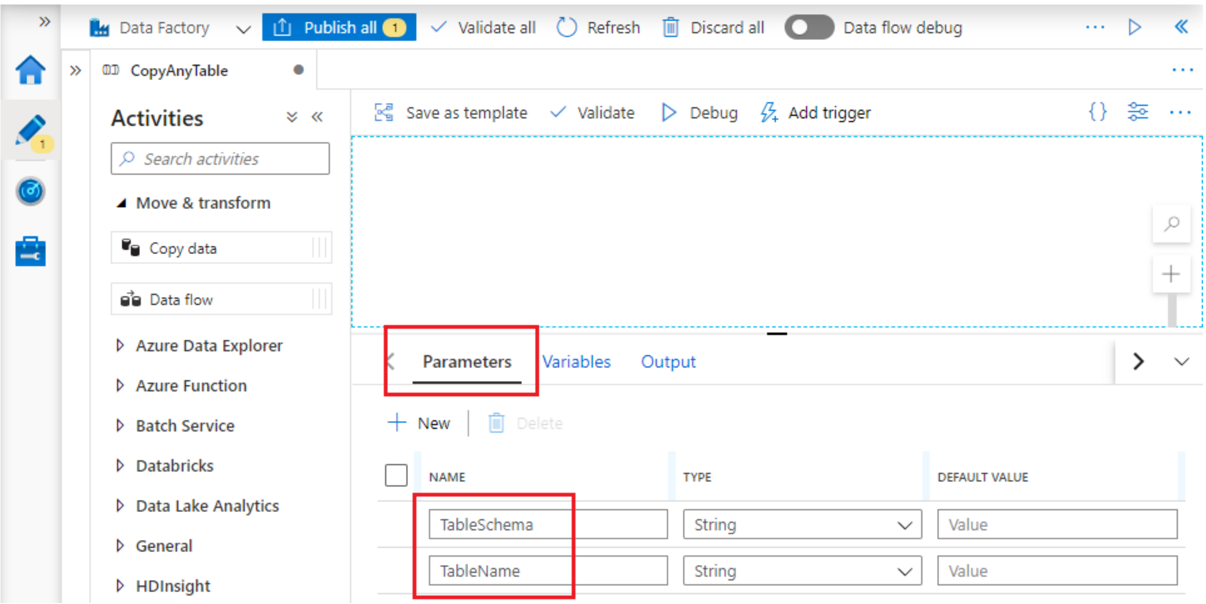
Compression type none

Lab 5.3 – Create a generic pipeline

A pipeline using the two datasets you have created must provide runtime values for their parameters. Instead of hard-coding them into the pipeline definition, use pipeline parameters to make the pipeline generic as well.

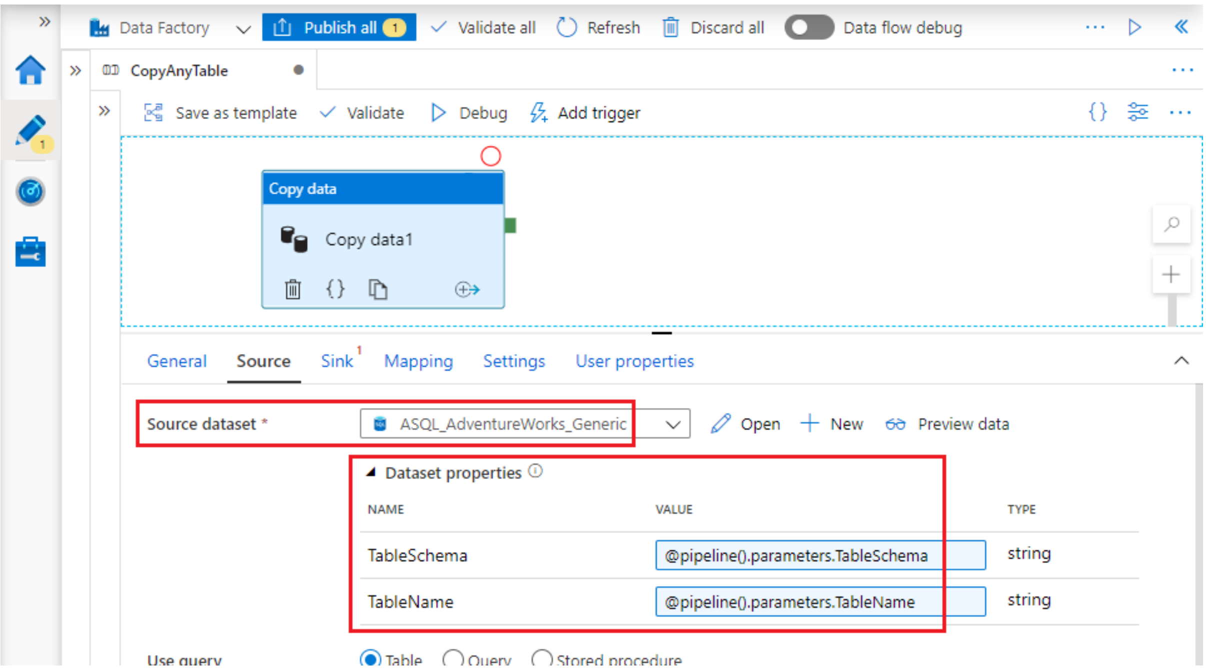
1. Create a new ADF pipeline.

2. On the pipeline's "Parameters" tab, create two parameters called "TableSchema" and "TableName".



NAME	TYPE	DEFAULT VALUE
TableSchema	String	Value
TableName	String	Value

3. Drag a "Copy data" activity onto the pipeline canvas from the activity toolbox. On its "Source" tab, select your generic Azure SQL Database dataset as "Source dataset". Use the expression builder to specify pass the **pipeline**'s parameter values into the corresponding **dataset** parameters.



Source dataset * ASQL_AdventureWorks_Generic

Dataset properties

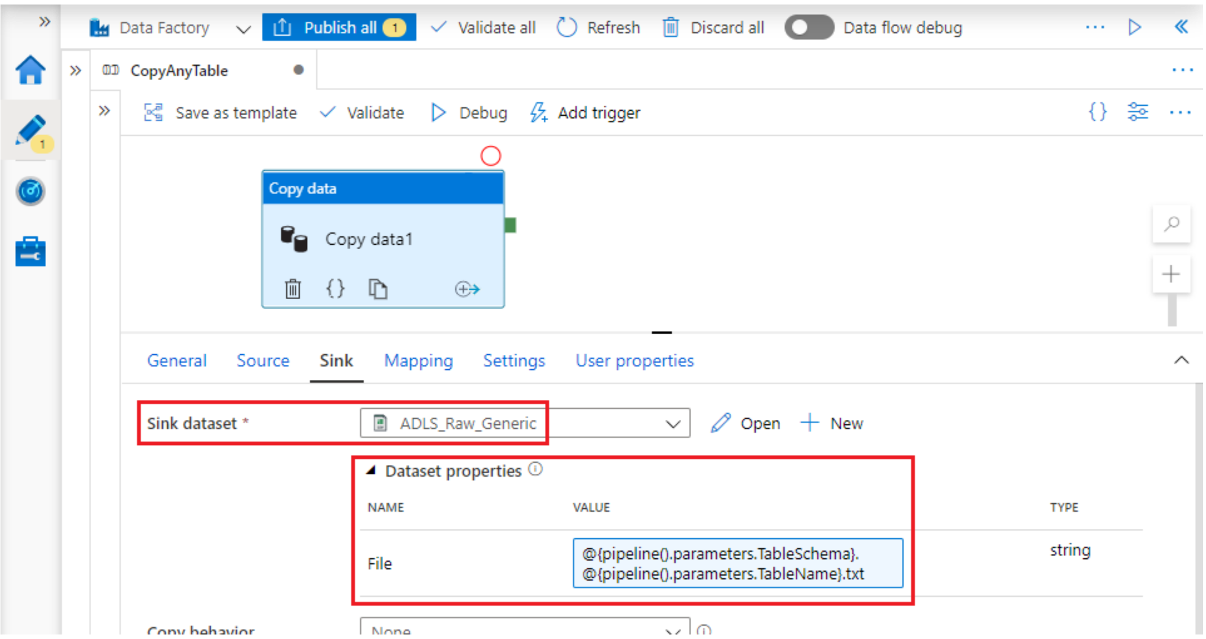
NAME	VALUE	TYPE
TableSchema	@pipeline().parameters.TableSchema	string
TableName	@pipeline().parameters.TableName	string

Use query ● Table ○ Query ○ Stored procedure

4. In the same way, on the pipeline's "Sink" tab choose the generic Azure Data Lake storage dataset. Set the dataset's "File" parameter to this expression:

```
@{pipeline().parameters.TableSchema}.@{pipeline().parameters.TableName}.txt
```

The expression uses *string interpolation* – it is a string literal that contains embedded ADF pipeline expressions. Embedded expressions are enclosed in braces and preceded by an @ character.



Sink dataset * ADLS_Raw_Generic

Dataset properties

NAME	VALUE	TYPE
File	@{pipeline().parameters.TableSchema}. @{pipeline().parameters.TableName}.txt	string

Copy behavior None

5. Click "Debug" to run the pipeline in debugging mode. The ADF UX prompts you to supply values for the pipeline's parameters. Use these values:
   - For "TableSchema", input "SalesLT"
   - Set "TableName" to "ProductDescription"

When the pipeline execution has successfully completed, inspect the “lakeroot” container’s “Raw” folder to verify that the file “SalesLT.ProductDescription.txt” has been created.

Lab 5.4 – Reuse the generic pipeline

Your generic pipeline can be used to extract any table from the AdventureWorks database. In this lab you’ll extract all of them!

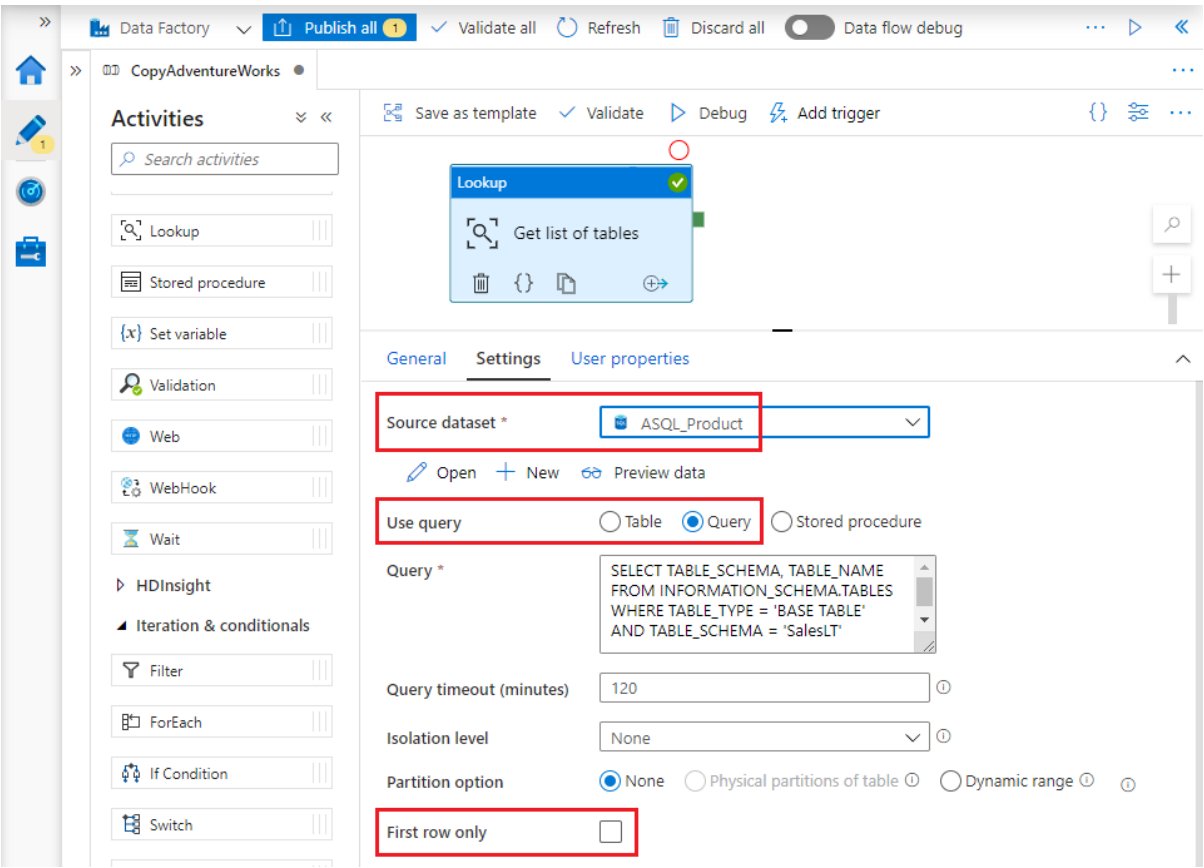
1. Create a new pipeline called “CopyAdventureWorks”.

2. Rather than supply the list of database tables yourself, you can use ADF’s “Lookup” activity to query the database catalog. The Lookup activity is found in the activity toolbox’s “General” group – drag one onto the pipeline canvas.

3. Give the lookup activity a sensible name, then on its “Settings” tab choose the “ASQL_Product” dataset from Lab 2. You are going to override its table settings with a SQL query – change the “Use query” option from “Table” to “Query”. In the “Query” pane which appears, enter this SQL query:

```sql
SELECT TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
AND TABLE_SCHEMA = 'SalesLT'
```

   Finally, **untick** the “First row only” check box at the bottom of the “Settings” tab.



Source dataset * ASQL_Product

Use query ○ Table ● Query ○ Stored procedure

Query *

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
AND TABLE_SCHEMA = 'SalesLT'
```

Query timeout (minutes) 120

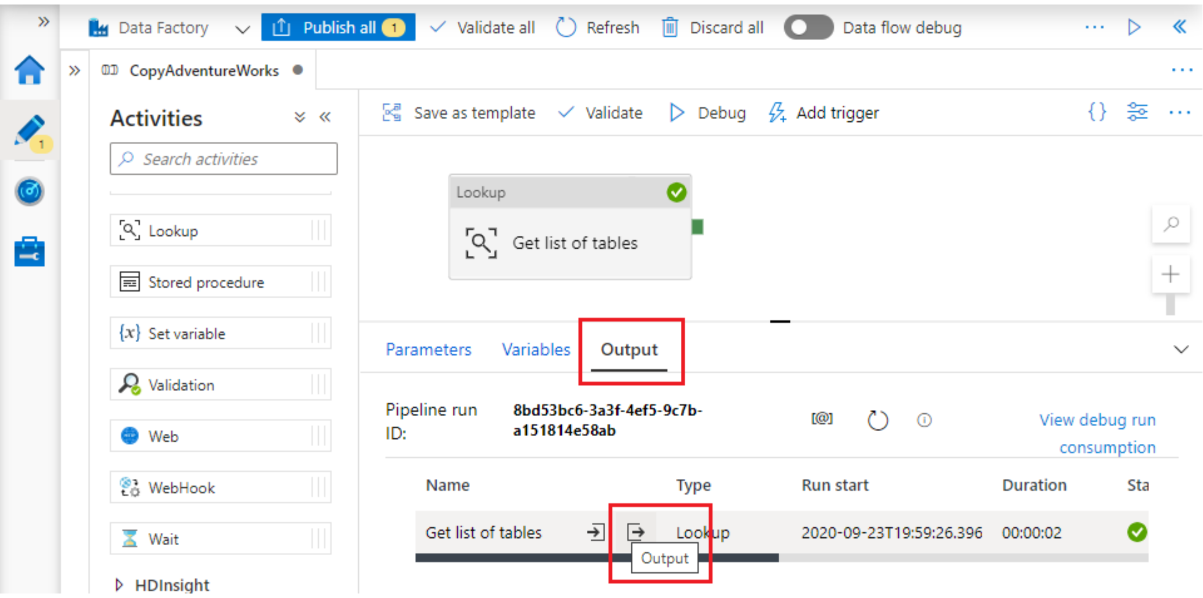
Isolation level None

Partition option ● None ○ Physical partitions of table ○ Dynamic range

First row only □

4. Before you go any further, run the pipeline by clicking “Debug”. This will verify that you have set up the Lookup activity correctly, but more importantly you will be able to inspect its

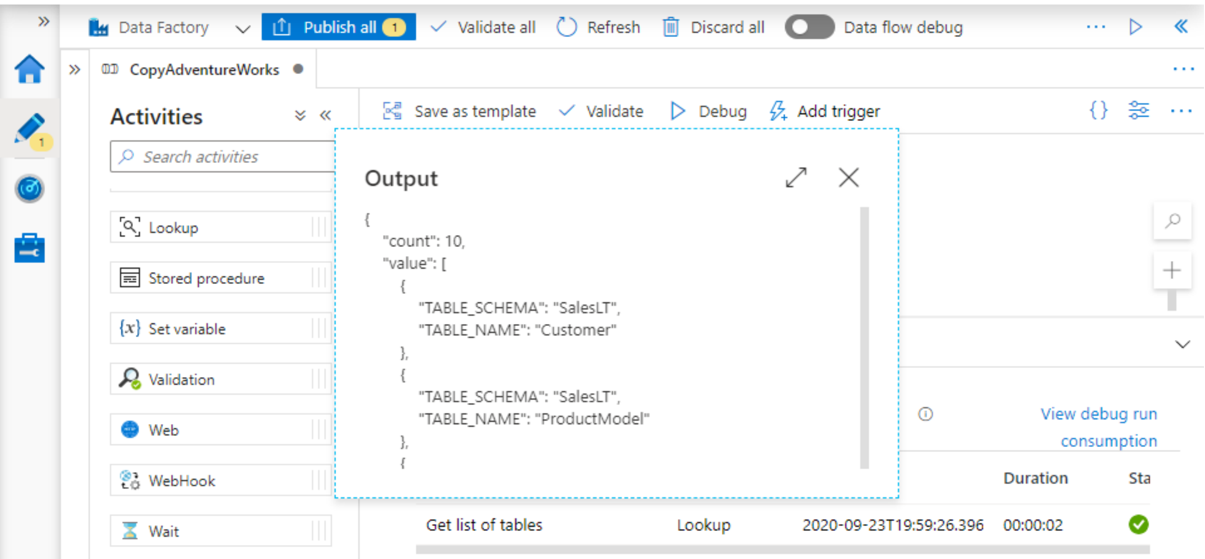
output. When the debug run has completed successfully, click on the activity's "Output" icon, visible when you hover over the Lookup activity in the pipeline's "Output" tab.



Pipeline run ID: 8bd53bc6-3a3f-4ef5-9c7b-a151814e58ab

Name	Type	Run start	Duration	Sta
Get list of tables	Lookup	2020-09-23T19:59:26.396	00:00:02	✓

5. ADF activity outputs are JSON objects. The Lookup output object includes a `value` property – this is a JSON array containing the list of tables returned by the SQL query. (If you see `firstRow` here and not `value`, you have omitted to untick the "First row only" checkbox).



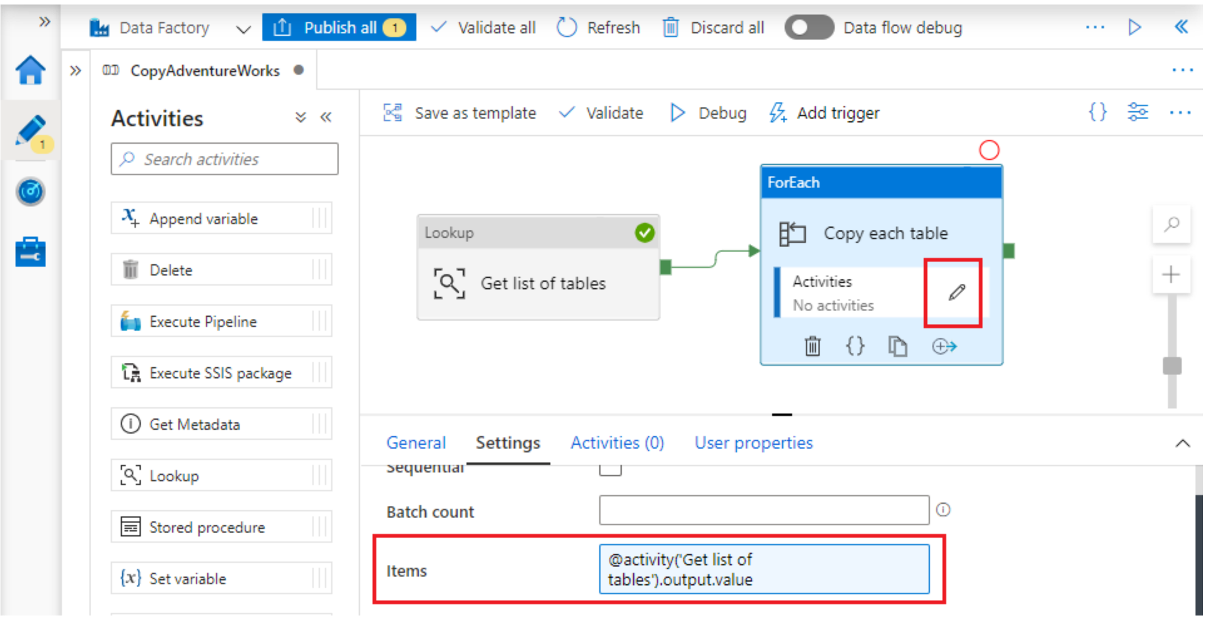
Output

```
{
    "count": 10,
    "value": [
        {
            "TABLE_SCHEMA": "SalesLT",
            "TABLE_NAME": "Customer"
        },
        {
            "TABLE_SCHEMA": "SalesLT",
            "TABLE_NAME": "ProductModel"
        },
        {
```

			Duration	Sta
Get list of tables	Lookup	2020-09-23T19:59:26.396	00:00:02	✓

6. Locate the "Iteration & conditionals" group in the activity toolbox and drag a ForEach activity onto the pipeline canvas. You'll use the ForEach to copy every table listed in the Lookup's output JSON array, so it must be constrained to run only after the Lookup activity has succeeded. To specify the dependency, click and drag the green "handle" on the right-hand edge of the Lookup. Drag it over the ForEach activity and release the mouse button.

7. On the ForEach activity's "Settings" tab, edit the "Items" field using the expression builder. At the bottom of the list of expression elements is an "Activity outputs" section. Your lookup activity appears there by name – click on it. The expression which appears in the pane refers to the activity output object as a whole – to refer to its `value` field, append ".value". Click "Finish" to close the expression builder.



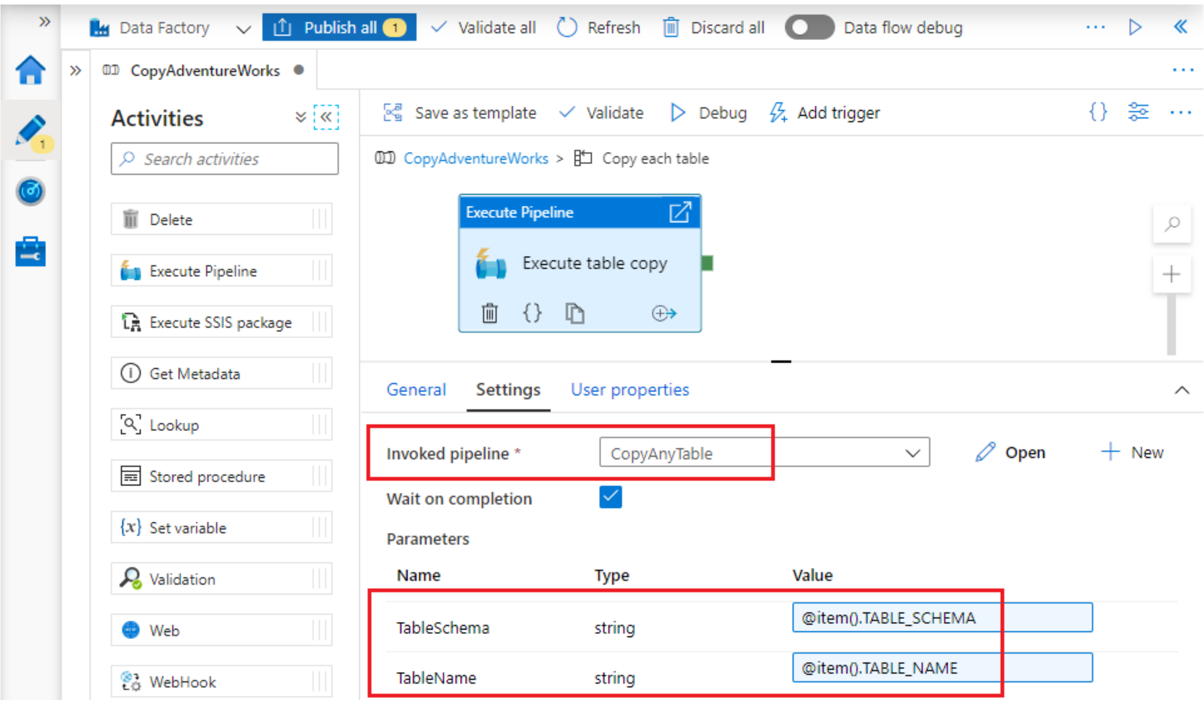
Batch count

Items: @activity('Get list of tables').output.value

8. Click on the “Pencil” icon on the ForEach activity to open a separate canvas containing activities to be executed for each array element. Drag an “Execute Pipeline” activity onto the empty canvas – you will use this to execute your generic pipeline.

9. On the Execute Pipeline activity’s “Settings” tab, choose your generic pipeline from the “Invoked pipeline” dropdown. The pipeline’s parameters are automatically added to the tab. Use the expression builder to create expressions for the parameters as follows:
   - TableSchema – `@item().TABLE_SCHEMA`
   - TableName – `@item().TABLE_NAME`

`item()` refers to the JSON array element under consideration. The property names following the dot must match those found in the output of your Lookup activity.



Invoked pipeline *: CopyAnyTable

Wait on completion: ☑

Parameters

Name	Type	Value
TableSchema	string	@item().TABLE_SCHEMA
TableName	string	@item().TABLE_NAME

10. Save/publish your changes and run the pipeline. When you run the pipeline using “Debug”, the ADF UX warns you that the ForEach activity will handle its input collection sequentially. This is a debug-specific feature – default ForEach behaviour in published pipelines is to handle array elements in parallel. This makes the activity a powerful way to execute multiple data processing activities simultaneously.

11. When the pipeline execution has successfully completed, inspect the “lakeroot” container’s “Raw” folder to verify that it now contains ten files – one for each table in the database’s SalesLT schema.

Recap

In Lab 5 you:

- created reusable datasets to represent database tables and data lake files generically
- created a generic pipeline using generic datasets
- used the generic pipeline to extract ten different tables with a single pipeline containing only three activities.