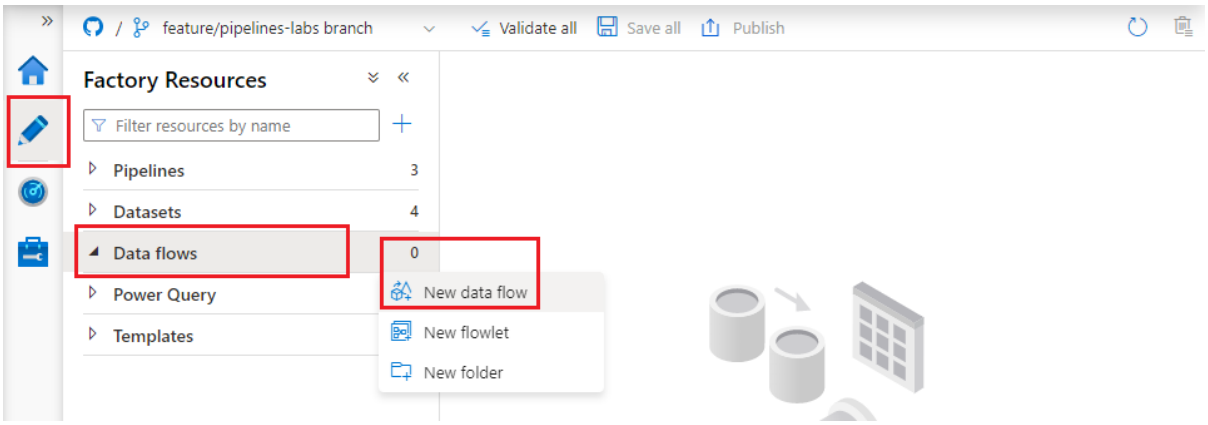
Lab 4 – Author a data flow

In this lab you will use an Azure Data Factory data flow to implement a familiar data warehousing process: maintaining a dimension table.

Lab 4.1 – Enable data flow debugging

Data flows are debugged using on-demand Apache Spark clusters. Provisioning a cluster takes several minutes, so start this lab by switching "Data flow debug" on for your ADF Studio session.

1. Navigate to ADF Studio's Author hub, then use the "Data flows Actions" menu to create a new dataflow.



2. The data flow authoring canvas appears, with the usual "Properties" flyout on the right. Name the data flow "UpdateProductDimension" and dismiss the flyout.

3. Above the data flow canvas a "Data flow debug" toggle switch is visible. When you the move the switch to the right, the "Turn on data flow debug" flyout appears – click "OK" to accept the default options and to start provisioning the debug cluster.

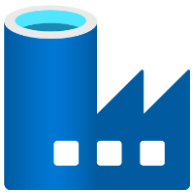
While the debug cluster is warming up, continue with Lab 4.2.

Lab 4.2 – Create a delimited data lake dataset

The product dimension we will be building later in the lab will combine information from the Product, ProductSubcategory and ProductCategory files into a three-tier hierarchy. Reading data from those files in the data lake requires an integration dataset, but the binary dataset we have been using is not sufficient in this case – ADF must parse file data into columns to make each file available as a stream of records.

1. Create the new dataset as follows:

   - Choose the "Azure Data Lake Storage Gen2" data store.
   - Choose the "DelimitedText" file format. (The dataset must be created from scratch rather than cloned, because file format is not editable in a dataset clone).
   - Name the dataset "ADLS_TSV_AdventureWorks", and select your data lake linked service.
   - Leave the other options with their default values – we need to define a dataset parameter before we can properly complete them – and click OK.

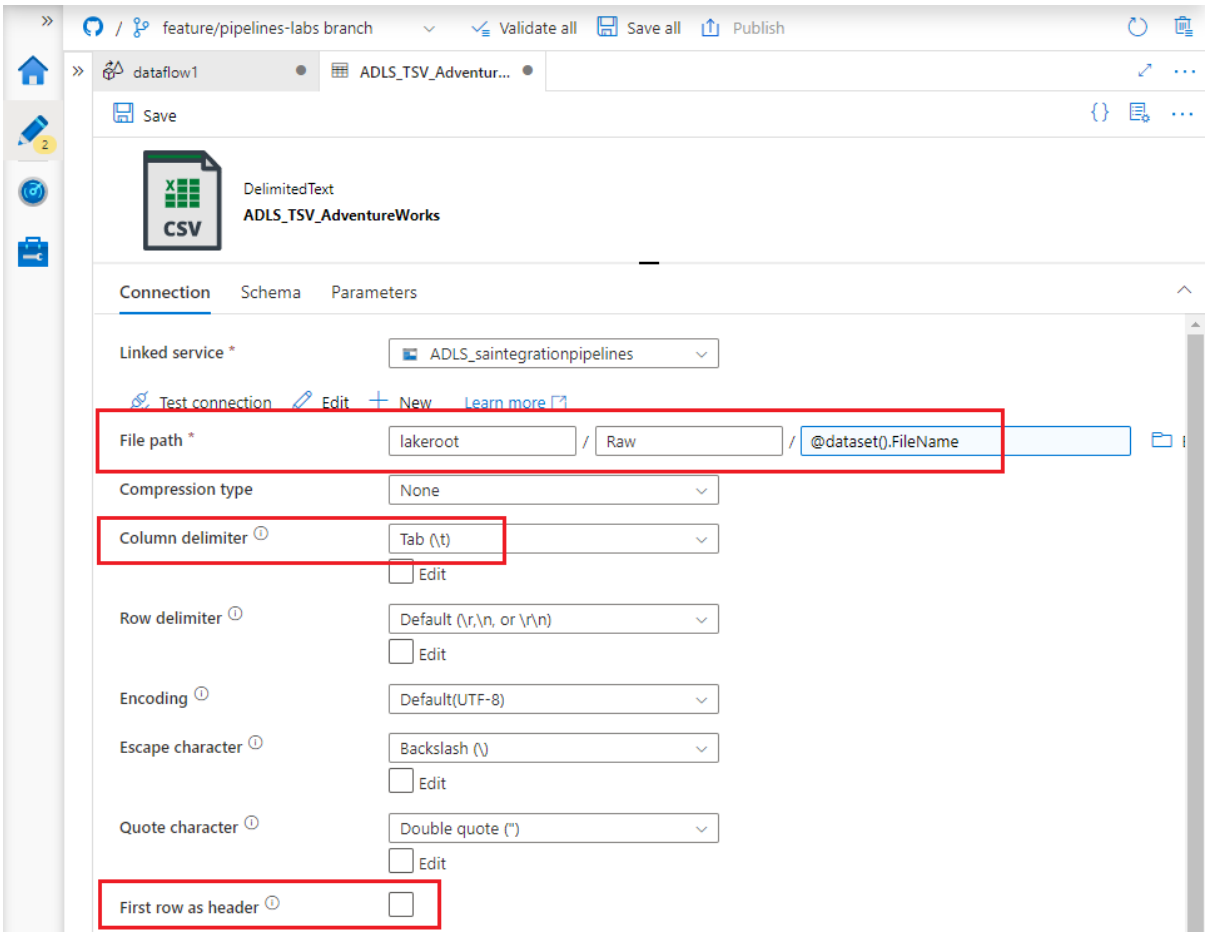


- Save the new dataset.

2. Define a dataset parameter called “FileName”.

3. Configure options on the dataset’s “Connection” tab:

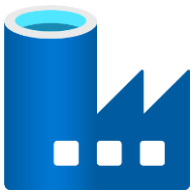
   - Set the **File path** components to use the “lakeroot” file system, the “Raw” directory and the filename specified by the dataset parameter created in step 2.
   - Set **Column delimiter** to “Tab (\t)” – recall from Lab 2.4 that the files downloaded from GitHub are tab-separated, despite their “.csv” extensions.
   - Ensure that **First row as header** is unchecked – the files have no header row.



4. Verify that the dataset has been correctly configured by using the “Preview data” button to the right of the file path (not visible in the screenshot). You will be prompted for a filename parameter value – remember that, using the expression given in the screenshot, this must **include** the extension e.g. “Product.tsv”.

Lab 4.3 – Combine data lake datasets

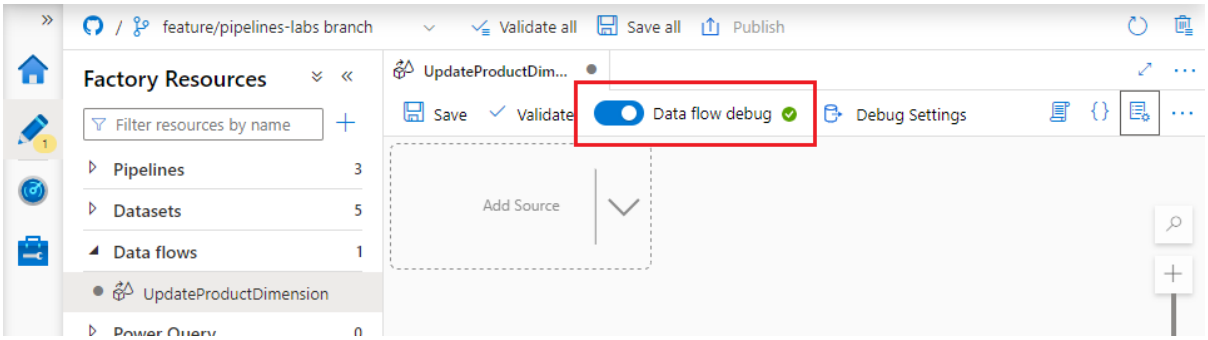
The product dimension combines product, subcategory and category information to support different aggregations of facts that have a product attribute. We are seeking to create the same effect as joining the tables together if they were in SQL database:



```
SELECT
  p.ProductID
, p.[Name] AS Product
, sc.[Name] AS SubCategory
, c.[Name] AS Category
FROM dbo.Product p
  LEFT JOIN dbo.ProductSubcategory sc
    ON sc.ProductSubcategoryId = p.ProductSubcategory
  LEFT JOIN dbo.ProductCategory c ON c.ProductCategoryId = sc.ProductCategoryID
```

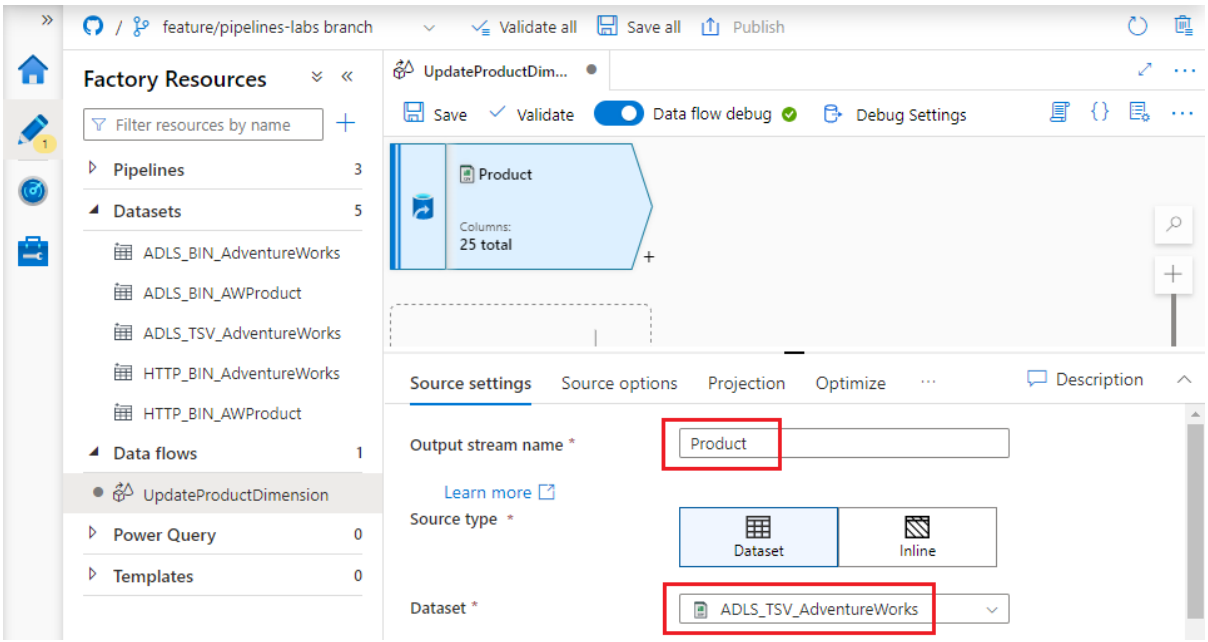
In this section you will use a data flow to produce this result in Azure Data Factory.

1. Return to the data flow you created in Lab 4.1 and check that the debug cluster has been successfully provisioned. When the cluster is available, a tick mark in a green circle appears to the right of the "Data flow debug" slider.



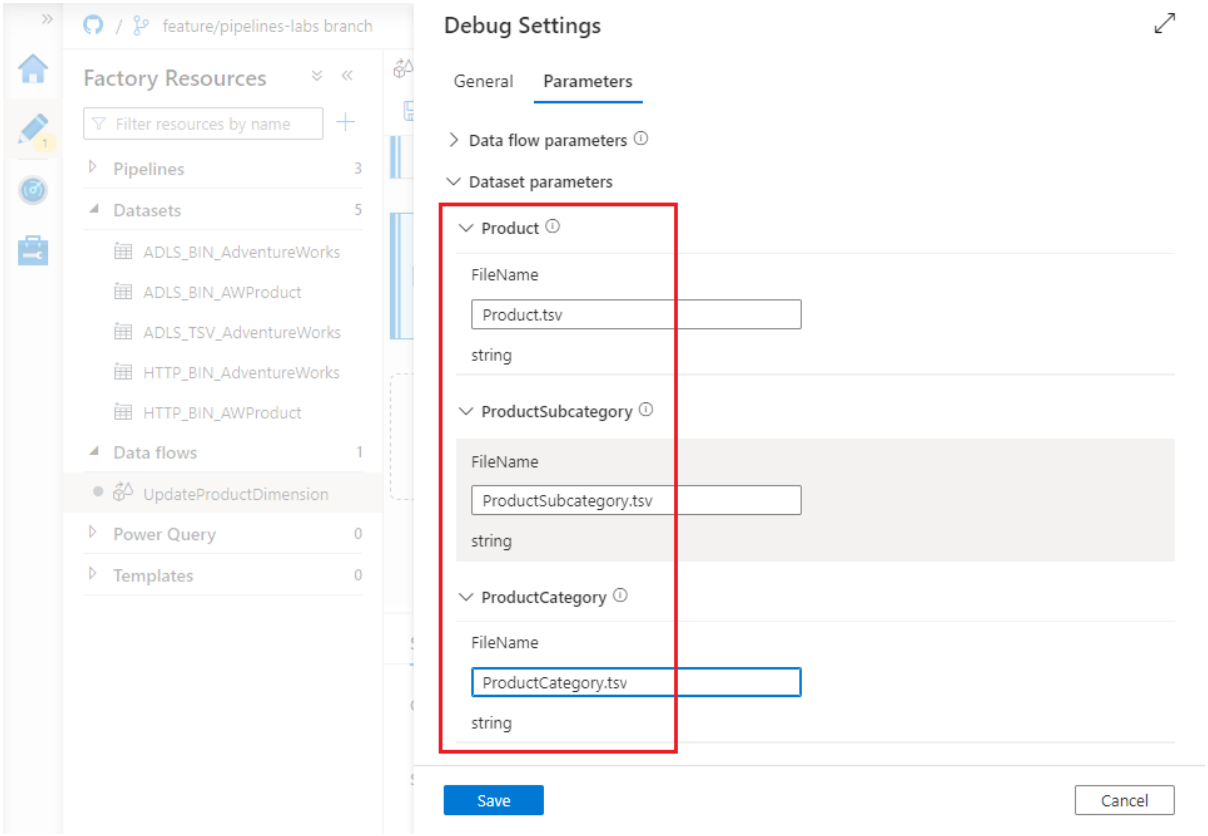
If the cluster is not ready yet, wait for it to finish warming up.

2. Click the "Add source" tile on the data flow canvas. On the source transformation's **Source settings** tab, change its "Output stream name" to "Product" and select the "ADLS_TSV_AdventureWorks" dataset.

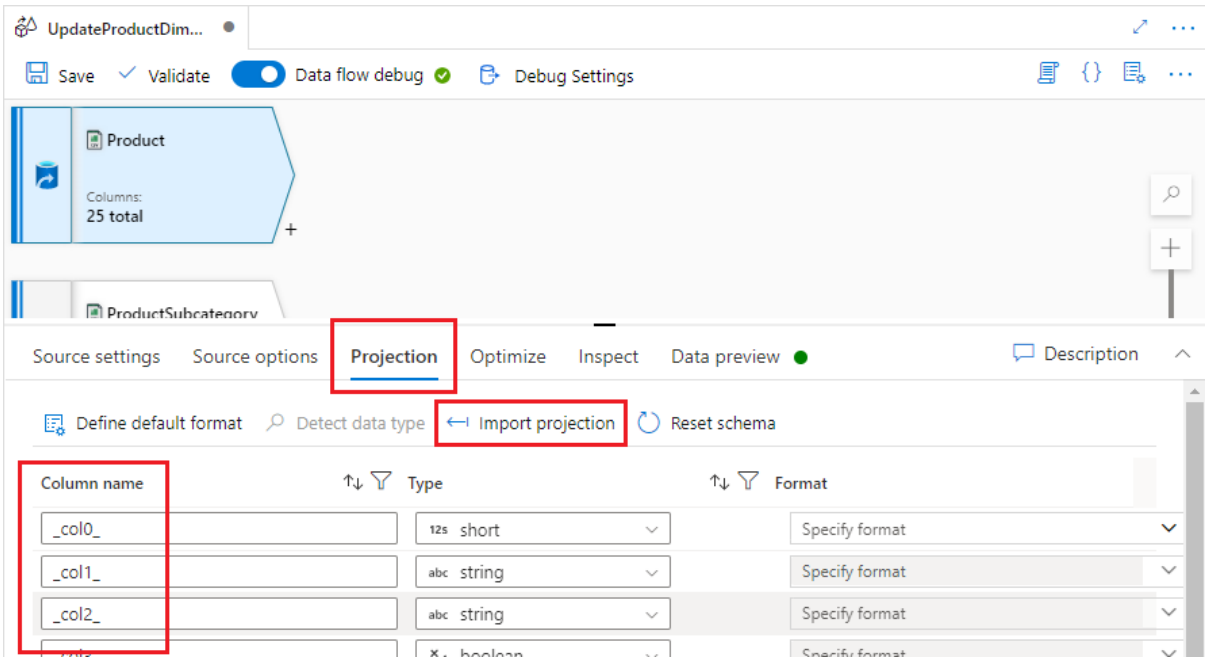


3. Repeat step 2 to add two more source transformations below "Product". Name one "ProductSubcategory" and the other "ProductCategory" – both should also use the "ADLS_TSV_AdventureWorks dataset.

4. At execution time, source dataset parameter values are specified from **outside** a data flow. To configure them during debug, click “Debug settings” in the data flow canvas header bar and choose the “Parameters” tab. Each dataset parameter appears in the “Dataset parameters” section – set each one appropriately and click “Save”.



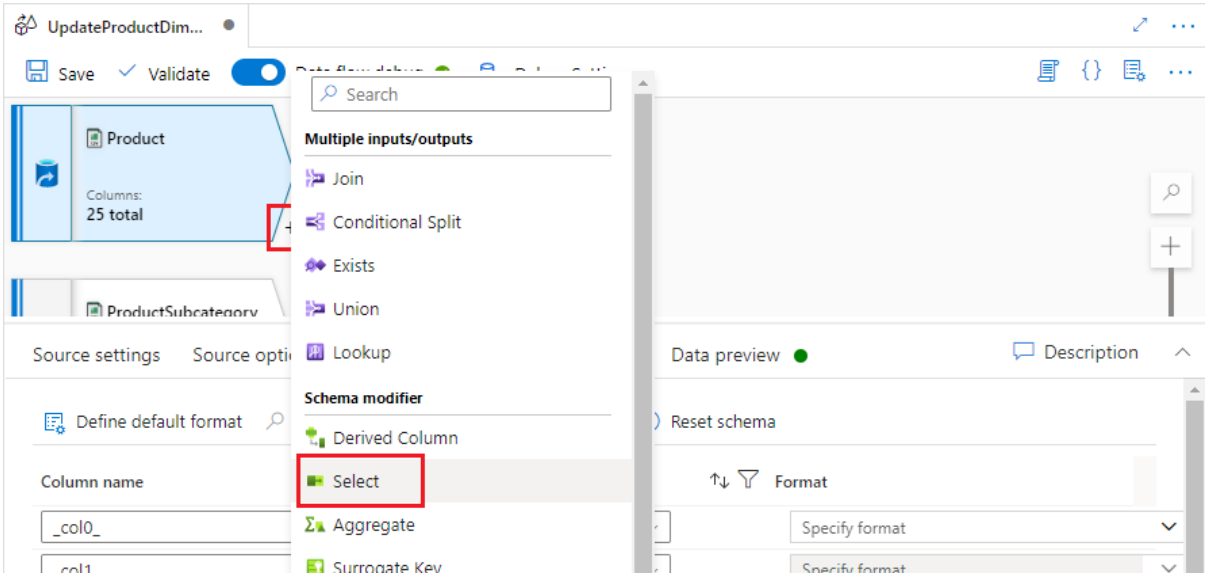
5. Return to the “Product” source transformation, select its “Projection” configuration tab and click “Import projection”. After a few moments, a schema will be inferred from the source file and presented in the tab.



6. Notice that column names are autogenerated and not meaningful, because the source file contains no column headers. The projection tab is editable, so you can change column names and types as required. In the case of Product

   - Rename "_col0_" to "ProductId" and ensure its type is "integer"
   - Rename "_col1_" to "Product" and ensure its type is "string"
   - Rename "_col18_" to "SubcategoryId" and ensure its type is "integer"

7. Use the "+" button at the bottom right of the "Product" transformation to append a "Select" transformation. The transformation allows you to remove and rename columns, without changing their types – remove all of the "_col…" named columns, leaving only the three columns you renamed in step 6.

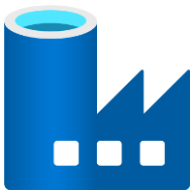


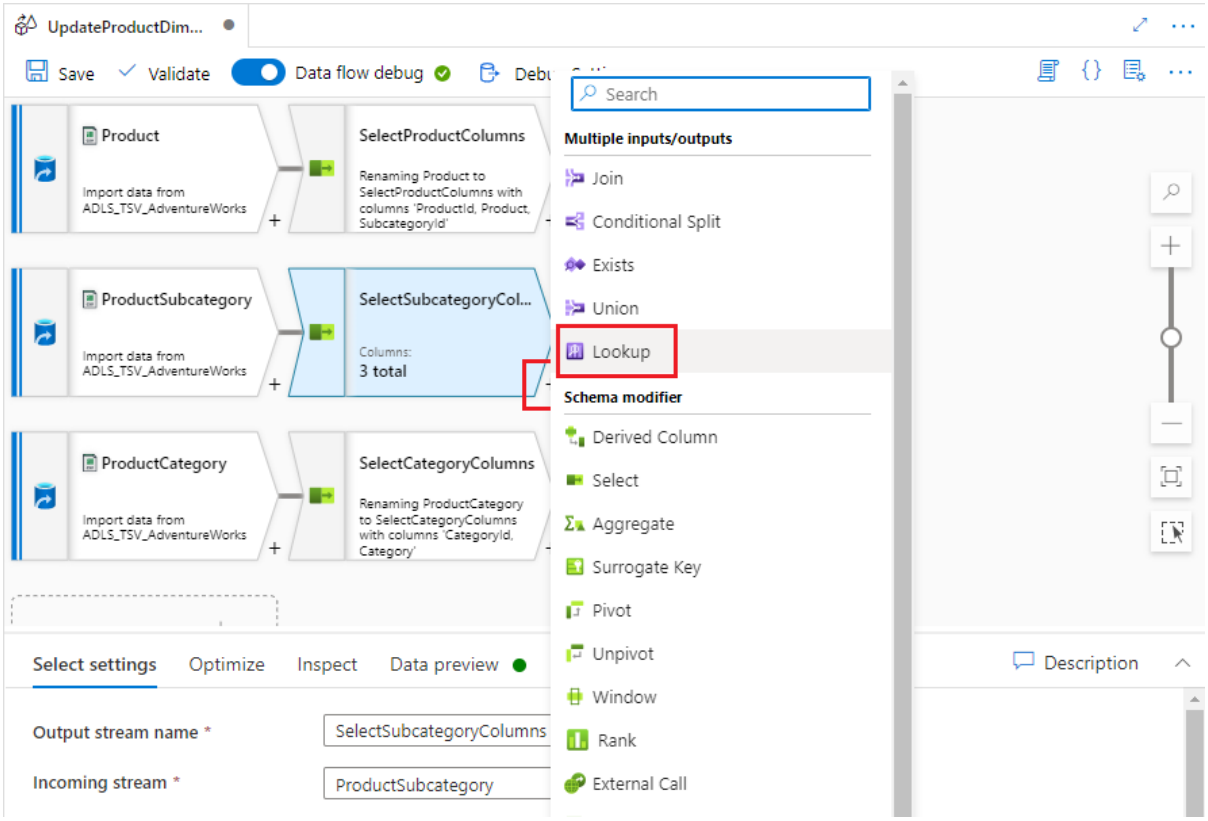
8. Repeat steps 5, 6 & 7 for the "ProductSubcategory" and "ProductCategory" source transformations. The names and types of columns to be retained for product subcategory are:
   - Rename "_col0_" to "SubcategoryId" and ensure its type is "integer"
   - Rename "_col1_" to "CategoryId" and ensure its type is "integer"
   - Rename "_col2_" to "Subcategory" and ensure its type is "string"

   The names and types of columns to be retained for product category are:
   - Rename "_col0_" to "CategoryId" and ensure its type is "integer"
   - Rename "_col1_" to "Category" and ensure its type is "string"

   You now have three parallel streams, loading and modifying data from the three source files.

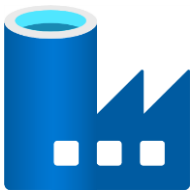
9. You can combine data into the Product stream from the other two streams using the data flow "Lookup" transformation. First, click the small "+" button below the Product Subcategory stream's Select transformation, then select "Lookup" from the popup menu.

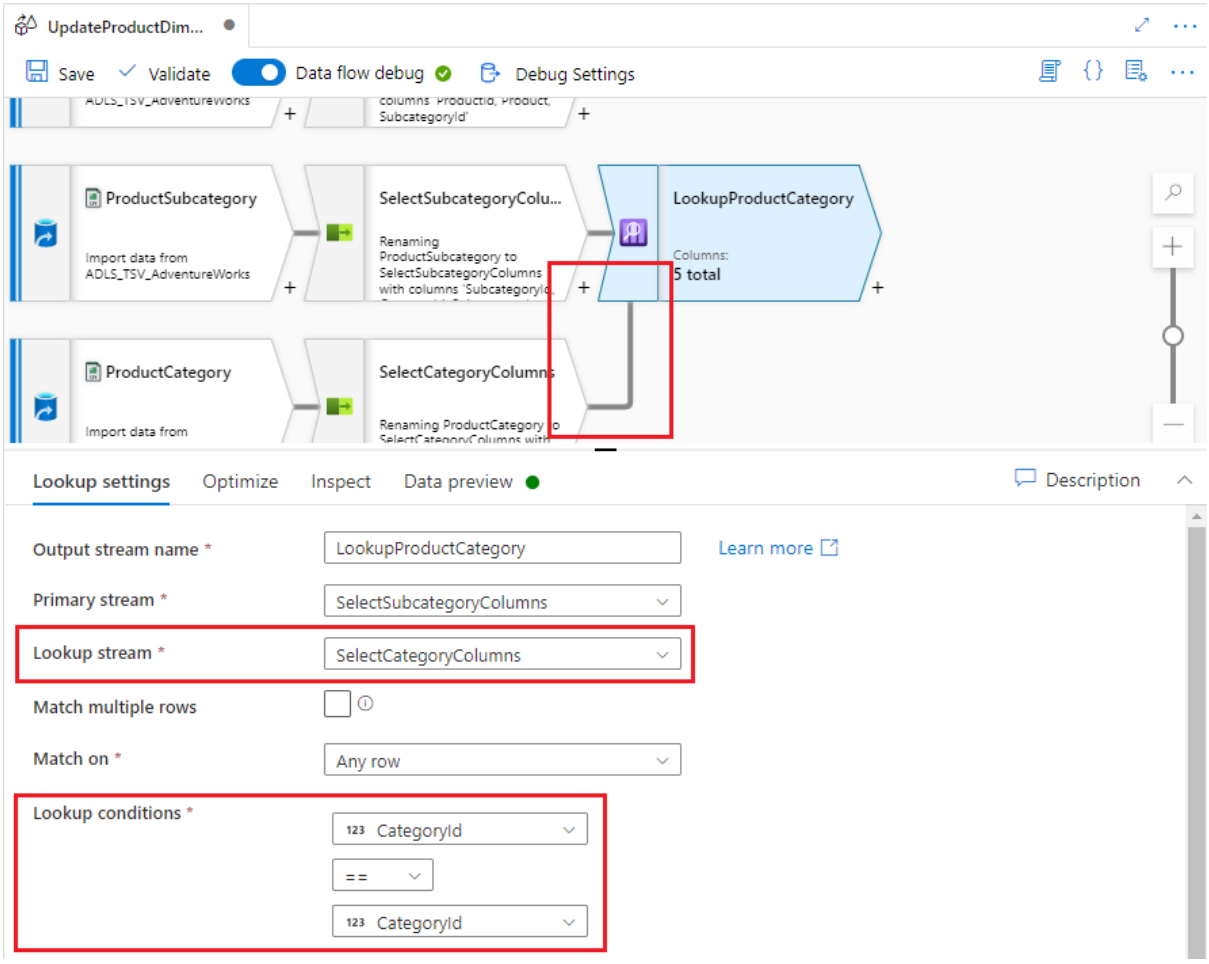




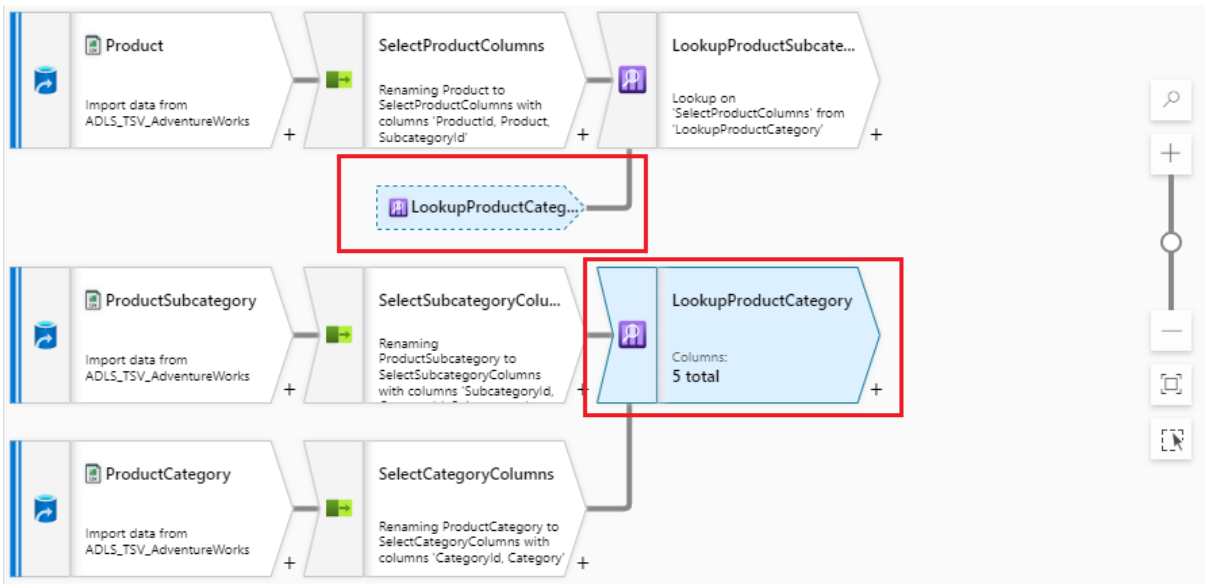
10. On the **Lookup settings** tab, name the transformation then set “Lookup stream” to use product category columns. (You can choose from any of the other five previous transformations, so take care to pick the ProductCategory stream’s Select transformation, and not the earlier Source for the stream).

    “Lookup conditions” specifies the fields to compare from each transformation and the operator to compare them with – choose the streams’ respective CategoryId fields. Notice that the lookup relationship also appears on the data flow canvas.

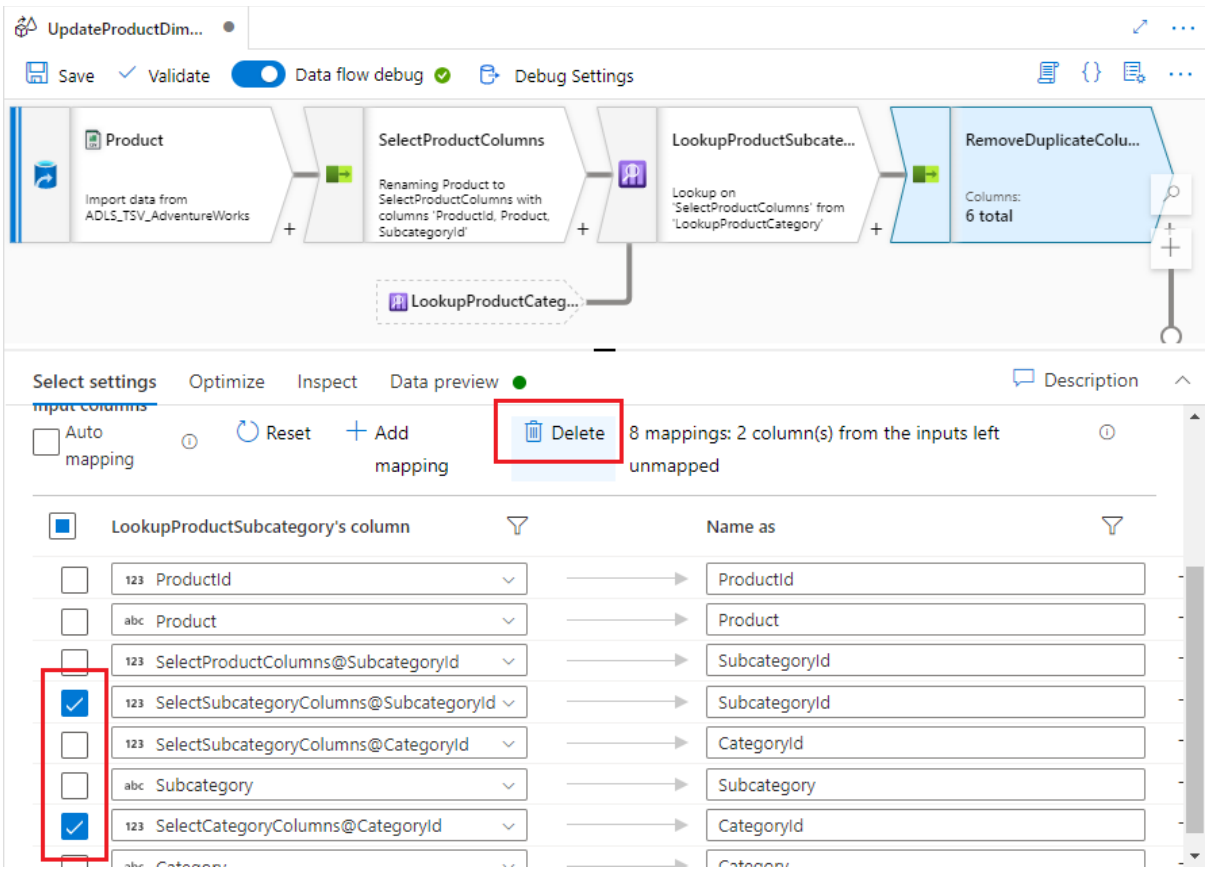




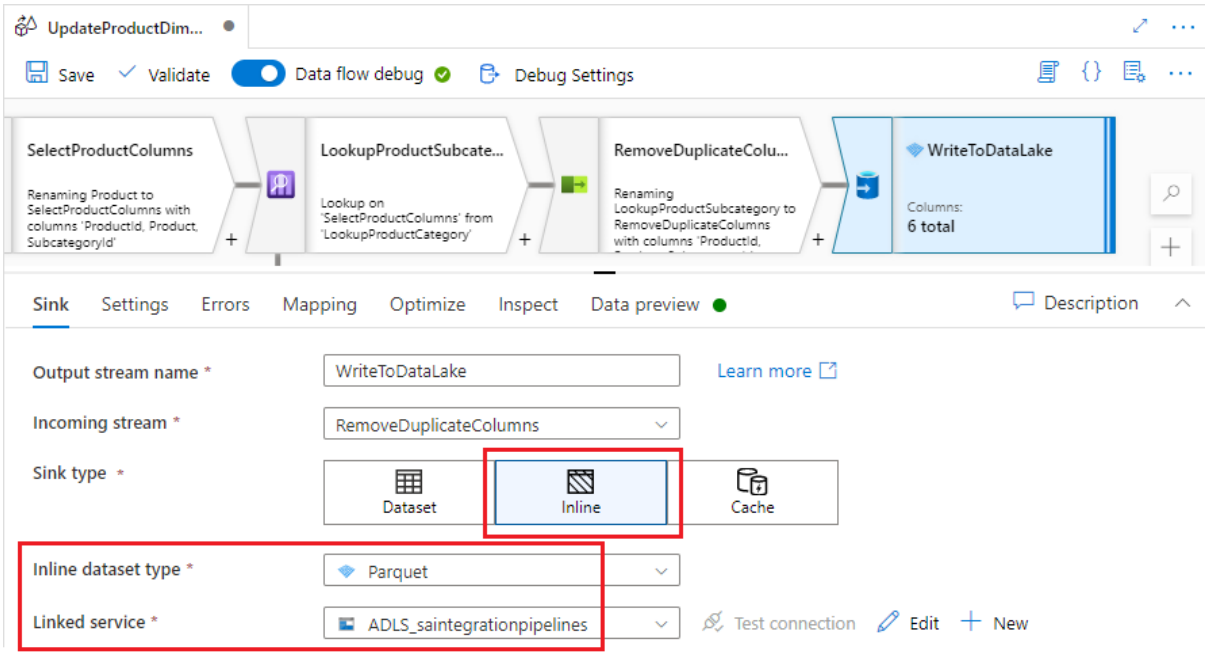
11. Add a second “Lookup” activity, this time on the Product stream, performing a lookup against the ProductCategory lookup’s output, based on matching SubcategoryId. This time the canvas displays a “reference node” instead of showing a direct link between the two transformations. This is just for readability – when you hover over either the reference node or the transformation it refers to, both light up in blue to indicate that they mean the same thing.



12. Open the new Lookup transformation's "Inspect" tab to view the set of columns present in the combined stream – notice it includes two copies of each of the join fields, one from each stream participating in the lookup. Clean this up with another Select transformation.

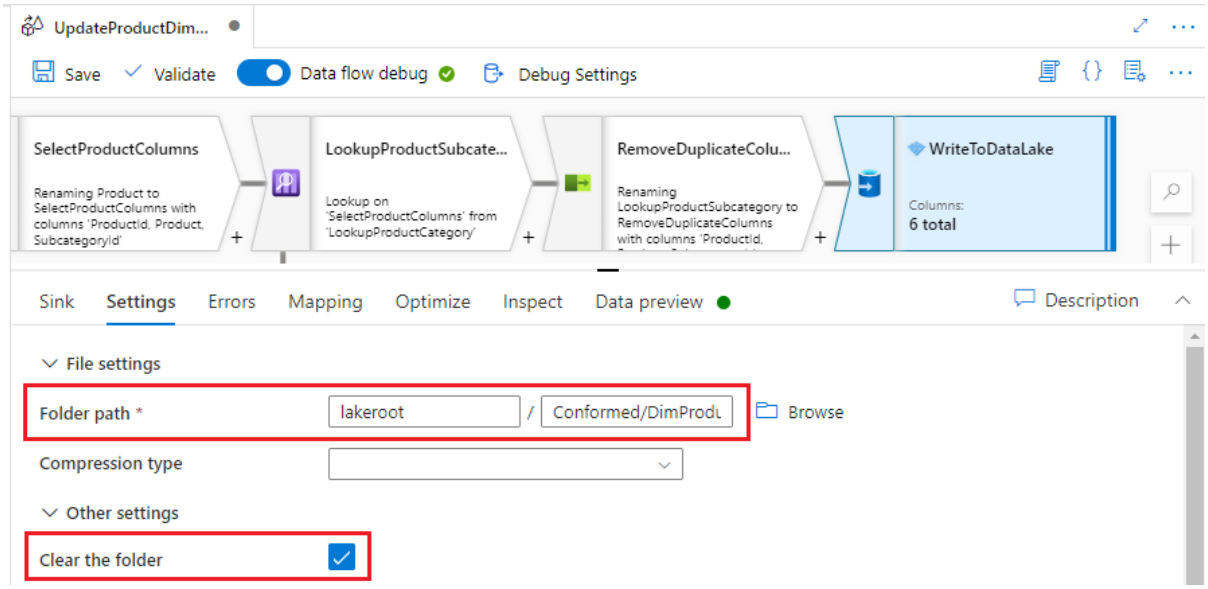


13. Finally, write the transformed dimension data back to the data lake using a "Sink" transformation. Add the transformation in the usual way, using the small "+" button following the Select transformation that removes duplicate columns. Sink is at the bottom of the list on the popup menu.



You haven't yet created a dataset to use as the data flow sink – we will use an inline dataset instead. Inline datasets aren't reusable by other data flows or pipelines, but avoid clutter when reusability is not required. Select "Sink type" = "Inline" and choose "Parquet" from the "Inline dataset type" dropdown. Parquet is a column-oriented, highly-compressible file format, offering significant performance benefits for data lakes.

14. Specify the sink "Folder path" on the transformation's "Settings" tab (Parquet is a multi-file storage format, so folder path identifies the folder **containing** those files). Choose the "lakeroot" file system and enter "Conformed/DimProduct" as the folder path. Tick the "Clear the folder" checkbox, to replace previous versions at each execution of the pipeline.



15. Click "Save all" to commit your data flow to the factory's attached GitHub repository.

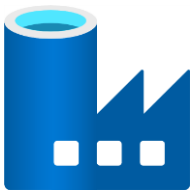
Lab 4.4 – Run the data flow

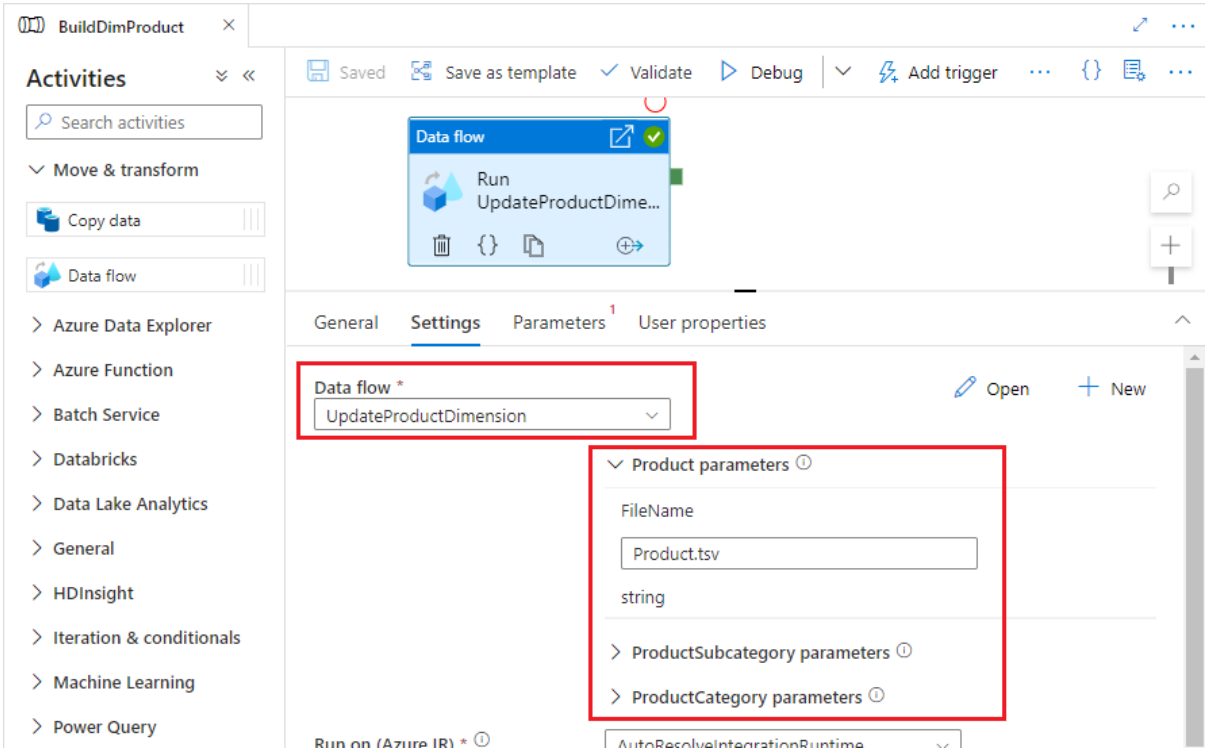
Data flows are executed by ADF pipelines. To run your data flow, create a pipeline for it.

1. Create a new ADF pipeline.

2. Expand the "Move & transform" group in the activity toolbox, then drag a "Data flow" activity onto the pipeline canvas. Name the activity appropriately, then on its "Settings" tab:

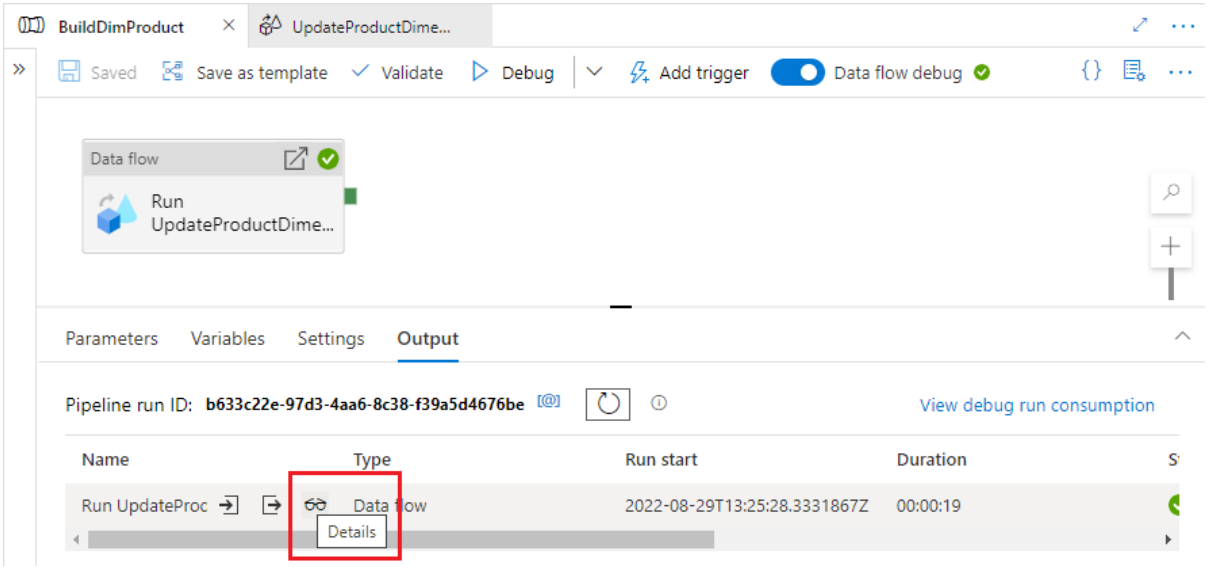
   - choose the data flow you created in Lab 4.3
   - provide dataset parameter values (identifying files to be loaded by each source transformation).





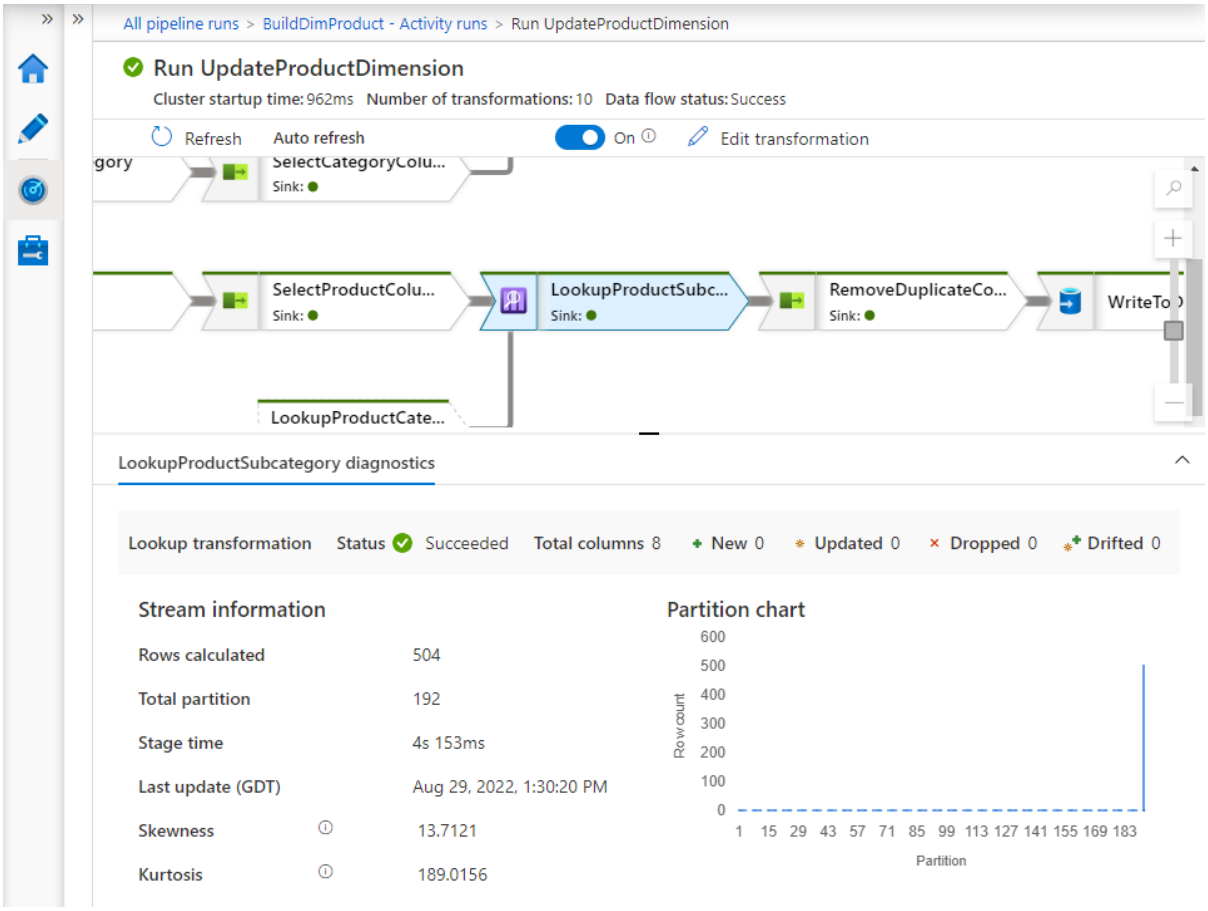
3. Click “Debug” to run the pipeline in debugging mode. A Spark cluster (Data flow debug enabled) is required to debug pipelines containing data flows, just as when you are developing them – if your debug session has timed out, you will need to start a new one as in Lab 4.1.

4. When the pipeline has finished running, a row of data appears in its “Output” pane for the Data flow activity. Hover over the activity’s name to reveal the “Details” button (“glasses” icon).

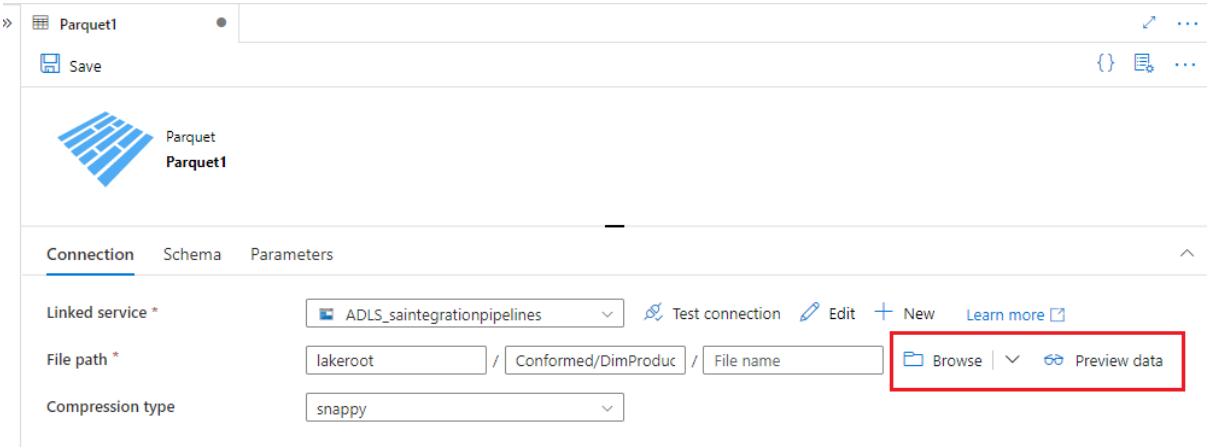


5. Click the “Details” button to open an interactive visualisation of more detailed data flow performance information. Selecting different transformations allows you to see the number of rows processed by a transformation, how quickly, and how the Spark cluster partitioned data for parallel processing. (For larger datasets, you can configure dataset distribution

yourself to optimise Spark executor partitioning, via each transformation's "Optimize" tab in the data flow editor).



6. Finally, you may wish to inspect the product dimension data written to your data lake. You can view the collection of Parquet files in the relevant data lake folder, but you cannot read them directly. To inspect dimension contents, create a new integration dataset using the data lake store and Parquet file format, browse to the Parquet folder, then use the dataset's "Preview" function to inspect preview its contents.



Note: When you preview data in this way, you may see very few subcategory and category values – this is because not every product in the source data is linked to a subcategory. To assure yourself that the lookups are correctly implemented, add a Sort transformation to your data flow, ordering the output by Subcategory **without** nulls first.

Lab 4.5 – Further work

This lab introduced concepts essential to the creation of a basic ADF data flow, but there is much more to learn. When using the popup menu to add Select, Lookup and Sink transformations you will have noticed that many more transformations exist. Data flows have their own expression language which supports powerful, complex data transformations. Flowlets allow you to create reusable data flow components which can be used by multiple data flows.

Start to broaden your knowledge by:

- using some of the other transformation types
- using the “Derived Column” transformation to begin to explore the data flow expression language.

Recap

In Lab 4 you:

- created an ADF data flow
- used Data flow debug to import file schemas (using “Import projection”)
- created a pipeline to execute your data flow
- used Data flow debug to run the pipeline in ADF Studio.