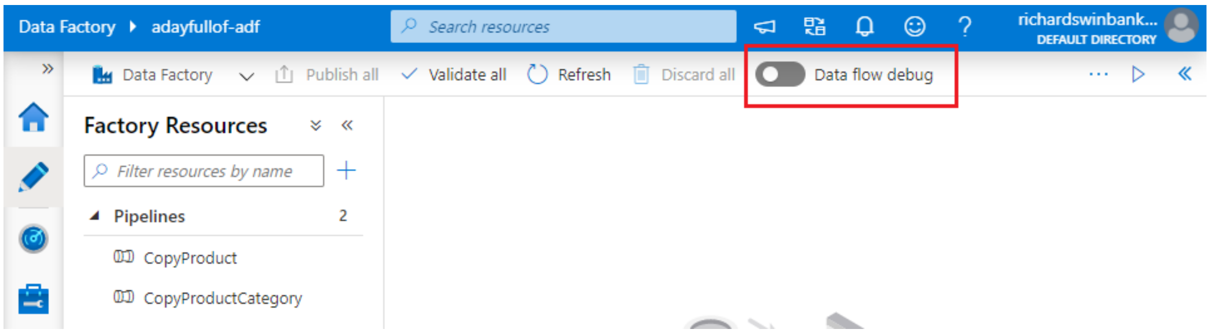
Lab 4 – Build a Mapping Data Flow

In this lab you will use Azure Data Factory's Mapping Data Flows feature to implement a familiar data warehousing process: maintaining a dimension.

Lab 4.1 – Enable data flow debugging

Mapping 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 UX session.

1. In the ADF UX, toggle the "Data flow debug" slider to "On".



2. When the ADF UX prompts you for confirmation, click "OK".

While the debug cluster is warming up, continue with Labs 4.2 & 4.3.

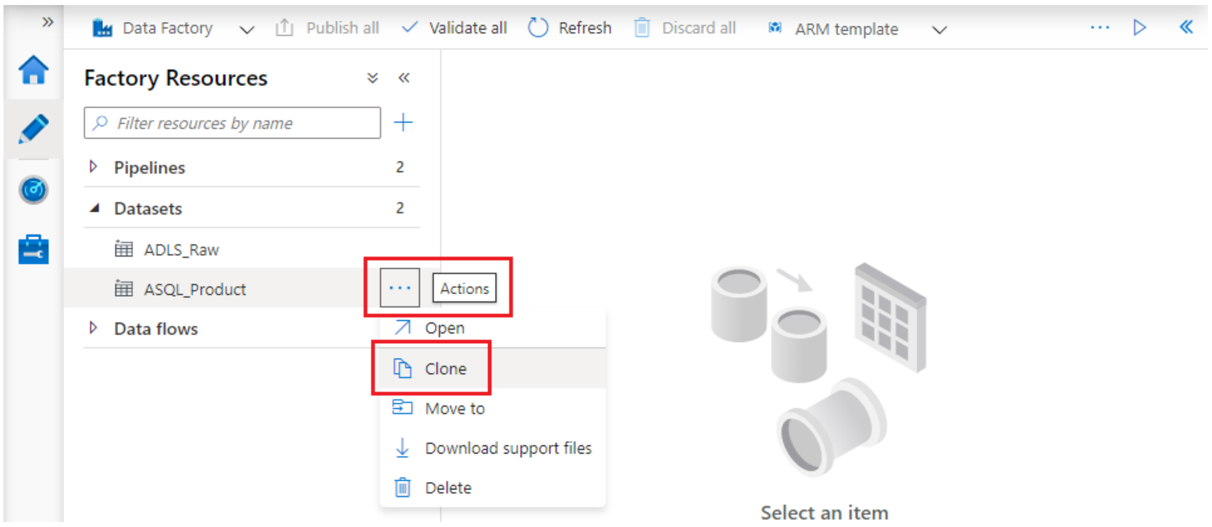
Lab 4.2 – Copy source data to the data lake

The product dimension will be built using data from three source tables:

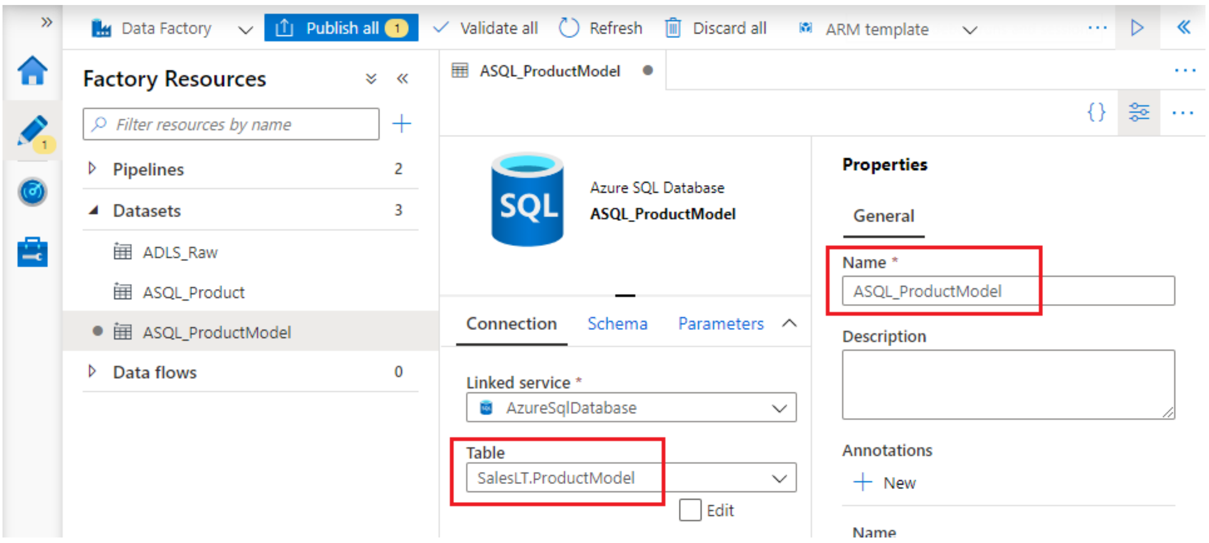
- [SalesLT].[Product]
- [SalesLT].[ProductCategory]
- [SalesLT].[ProductModel]

In Labs 2 & 3 you imported data from the first two tables into the data lake. Import data for [SalesLT].[ProductModel] now.

1. Create a copy of the "ASQL_Product" dataset by clicking its ellipsis "Actions" button in the "Factory Resources" list, then selecting "Clone".



2. The cloned dataset opens automatically with its “Properties” pane displayed. Change its name to “ASQL_ProductModel”, then on the “Connections” tab choose the corresponding [AdventureWorks] table.



Name *: ASQL_ProductModel

Linked service *: AzureSqlDatabase

Table: SalesLT.ProductModel

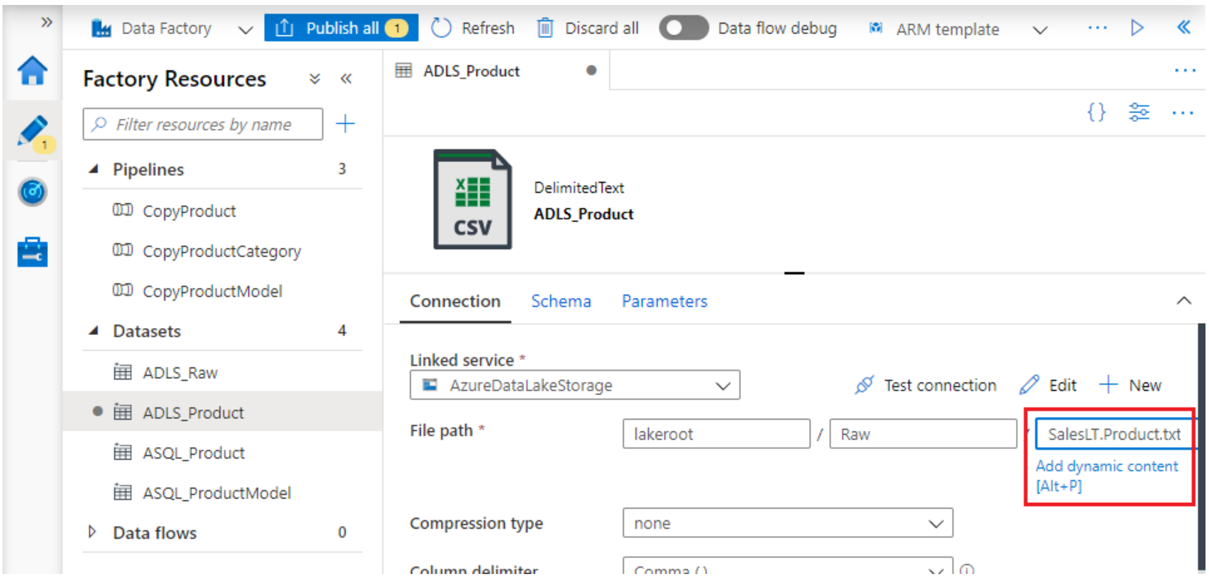
3. Create a new pipeline in the same way as Lab 2.3, using a Copy data activity with the new “ASQL_ProductModel” dataset as source and your Azure Data Lake Storage dataset as sink. Save your changes.

4. Run the pipeline in debug mode and verify that file “SalesLT.ProductModel.txt” has been created in the “lakeroot” container’s “Raw” folder.

Lab 4.3 – Create ADLS datasets

To use data from the three files now created in “/lakeroot/Raw”, you need datasets to represent them.

1. Create a new dataset by cloning your existing Azure Data Lake Storage dataset. Name it “ADLS_Product” and add file name “SalesLT.Product.txt” to the “File path” specified on the dataset’s “Connection” tab.



Linked service *: AzureDataLakeStorage

File path *: lakeroot / Raw / SalesLT.Product.txt

Compression type: none

Column delimiter: Comma (,)

2. Repeat step 1 to create two further datasets:

    - “ADLS_ProductCategory”, to represent file “/lakeroot/Raw/SalesLT.ProductCategory.txt”
    - “ADLS_ProductModel”, to represent file “/lakeroot/Raw/SalesLT.ProductModel.txt”

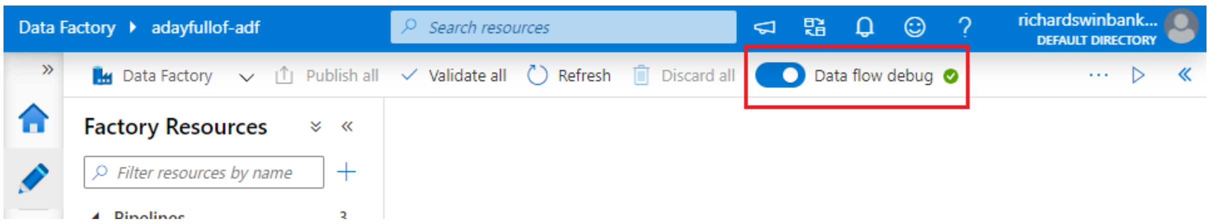
Lab 4.4 – Combine ADLS datasets

The product dimension combines product, model and category information to support different aggregations of facts that have a product attribute. This SQL query combines this information within the [AdventureWorks] database:

```
SELECT
  p.ProductID
, p.[Name] AS Product
, pm.[Name] AS ProductModel
, pc.[Name] AS ProductCategory
FROM SalesLT.Product p
  INNER JOIN SalesLT.ProductModel pm ON pm.ProductModelID = p.ProductModelID
  INNER JOIN SalesLT.ProductCategory pc ON pc.ProductCategoryID = p.ProductCategoryID
```

In this section you will use a Mapping Data Flow to reproduce this effect in Azure Data Factory.

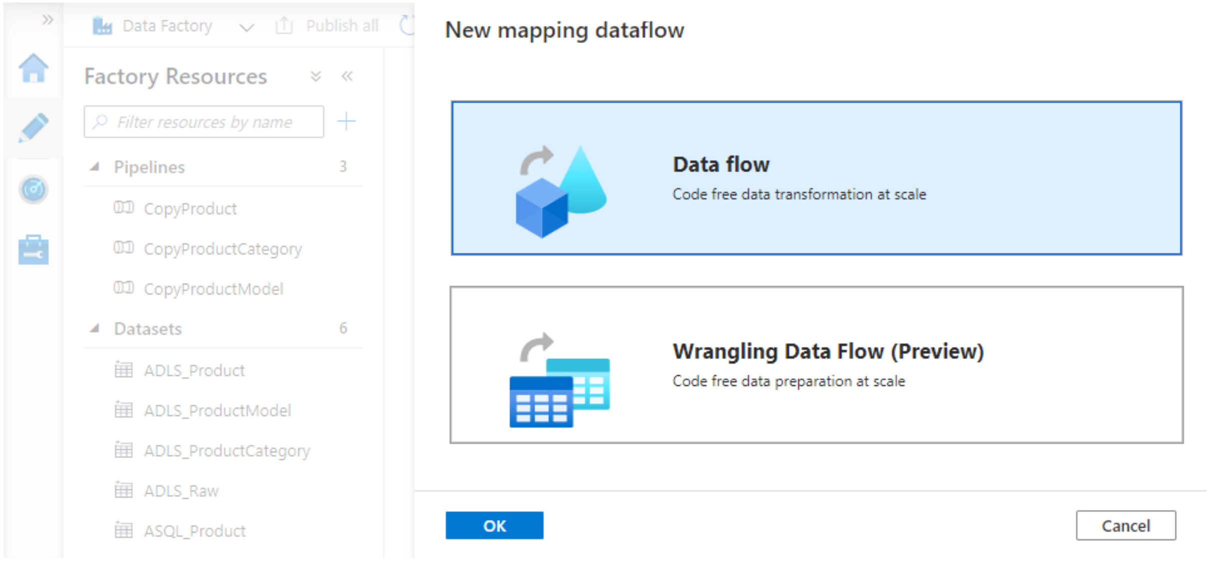
1. Check that the debug cluster has been successfully provisioned. When the cluster is available, a tick mark in a green circle appears to the right of the “Data flow debug” slider.



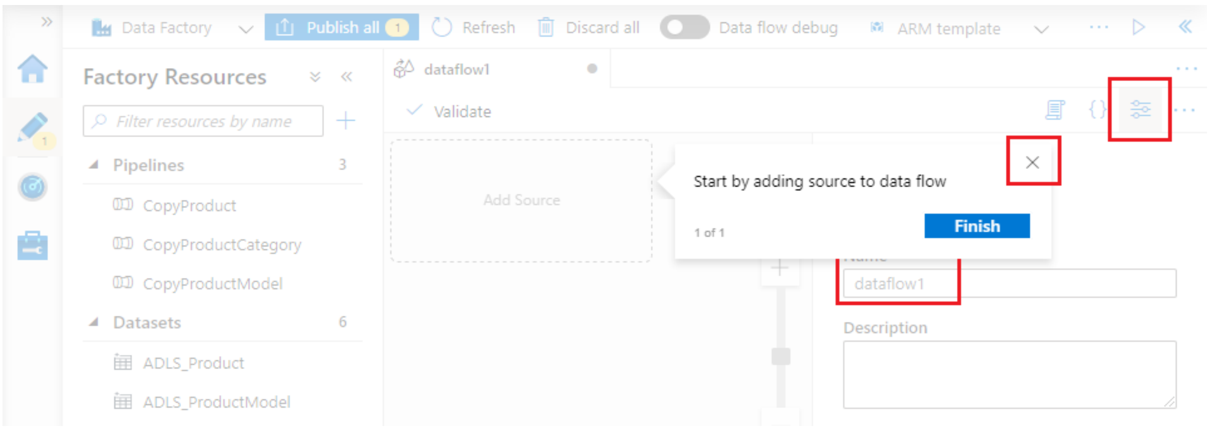
   If the cluster is not ready yet, wait for it to finish warming up.

2. In the “Factory resources” sidebar, click the “+” button to the right of “Filter resources by name”, then choose “Data flow”.

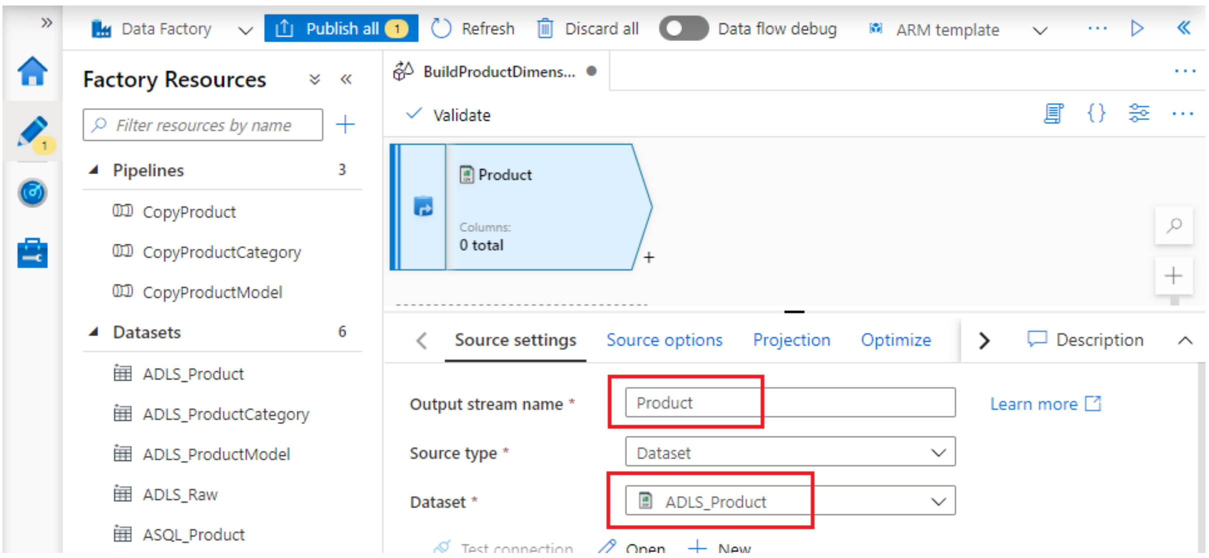
3. On the “New mapping dataflow” blade, select the “Data Flow” tile. Click “OK”.



4. The data flow canvas opens, displaying the callout “Start by adding source to data flow”. Dismiss the callout using its close button, then replace the data flow default name (“dataflow1”) with something more descriptive. Use the “Properties” slider button to close the data flow properties blade.

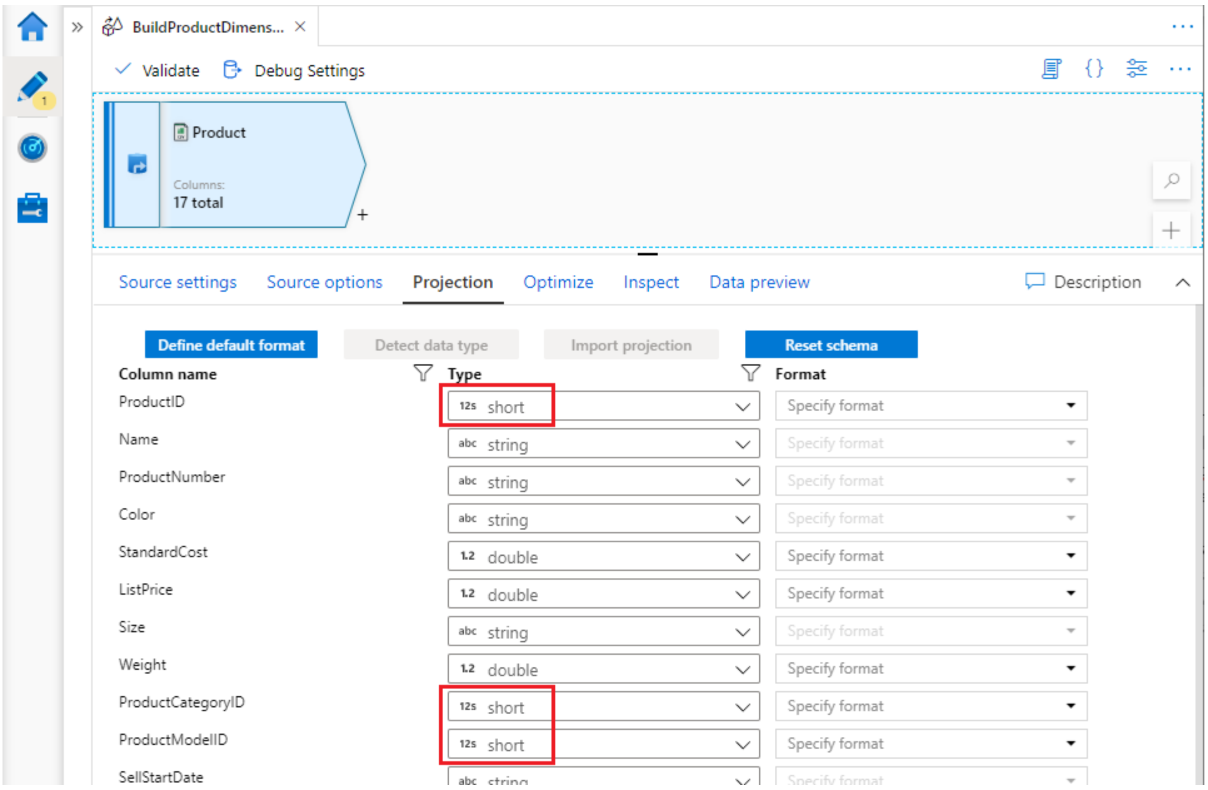


5. Click the “Add source” tile on the data flow canvas and close the callout that appears. On the source transformation’s **Source settings** tab, change its “Output stream name” to “Product” and select the corresponding “ADLS_Product” dataset.



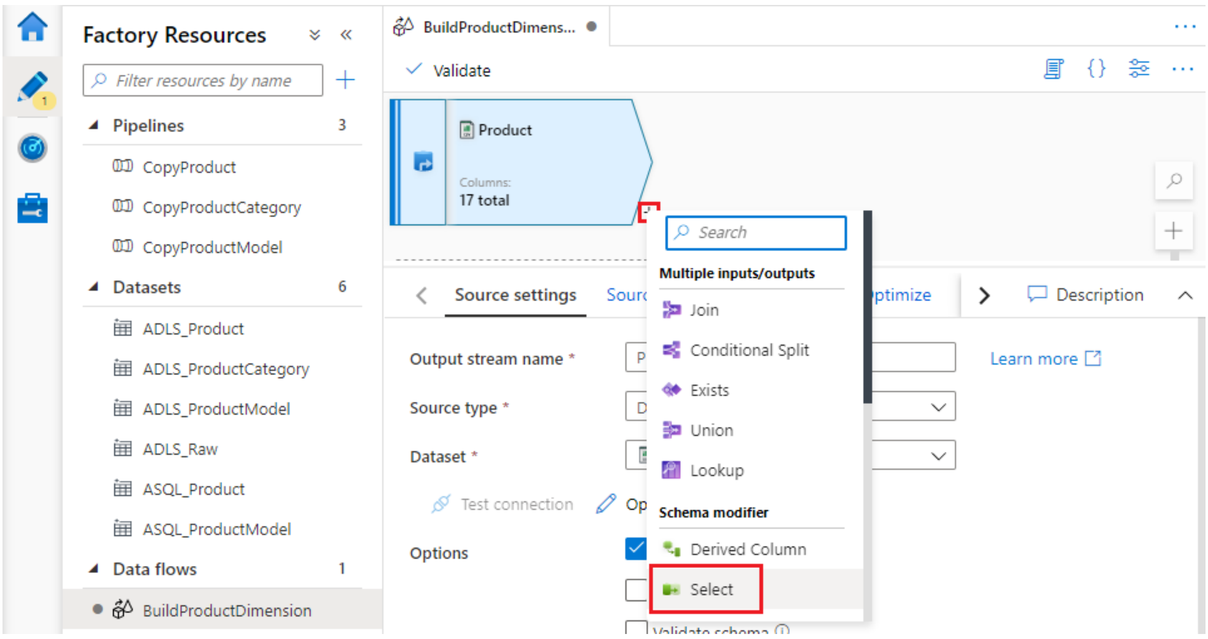
Output stream name * Product
Source type * Dataset
Dataset * ADLS_Product

6. On the **Projection** tab, click “Import projection” to import the source file’s schema. Check carefully that the type of the three fields “ProductID”, “ProductCategoryID” and “ProductModelID” has correctly been inferred as “short” – if this is not the case, use the “Type” dropdown to make the necessary correction(s).

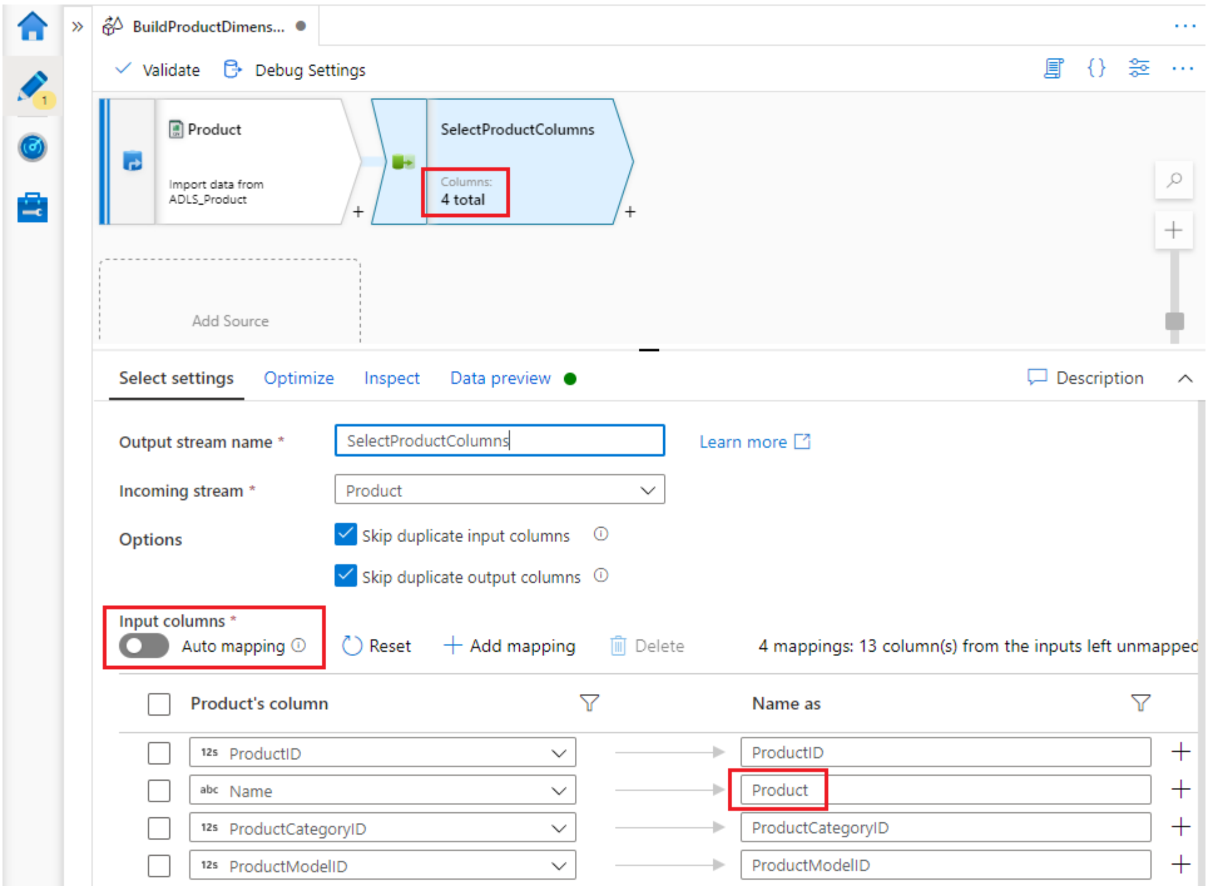


Column name	Type	Format
ProductID	short	Specify format
Name	string	Specify format
ProductNumber	string	Specify format
Color	string	Specify format
StandardCost	double	Specify format
ListPrice	double	Specify format
Size	string	Specify format
Weight	double	Specify format
ProductCategoryID	short	Specify format
ProductModelID	short	Specify format
SellStartDate	string	Specify format

7. The source transformation provides a stream of rows for consumption by downstream transformations. To add a transformation to consume the source stream, click on the small “+” button to the bottom right of the source transformation. Choose the “Select” transformation from the popup menu of available options.



8. On the Select transformation's **Select settings** tab, change its "Output stream name" to "SelectProductColumns", then scroll down to the "Input columns" section.



Output stream name * SelectProductColumns

Incoming stream * Product

Options: ✓ Skip duplicate input columns; ✓ Skip duplicate output columns

Input columns * Auto mapping (off)

4 mappings: 13 column(s) from the inputs left unmapped

Product's column	Name as
ProductID	ProductID
Name	Product
ProductCategoryID	ProductCategoryID
ProductModelID	ProductModelID

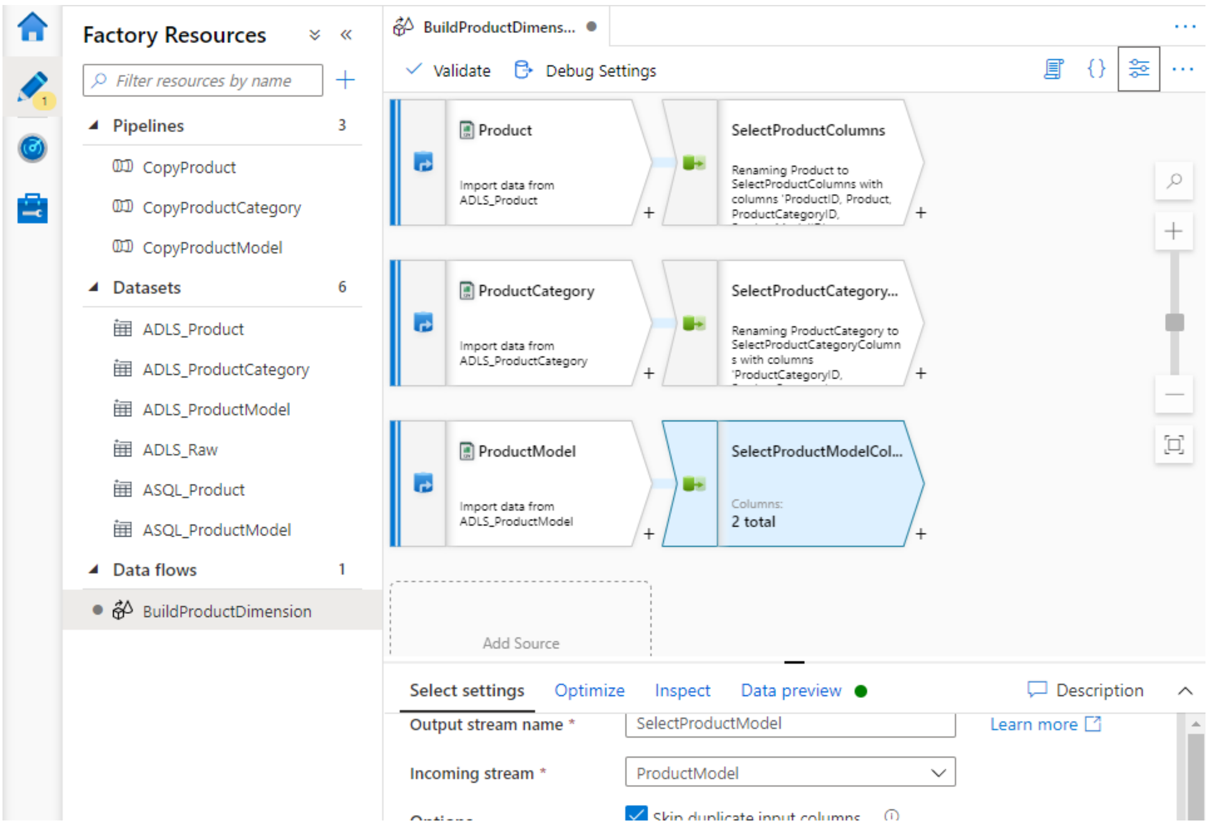
The Select transformation enables you to rename, reorder or remove columns from a stream. Ensure that the "Auto mapping" setting is disable so that you can see the transformation's column mappings, then remove all columns except "ProductID", "Name", "ProductCategoryID" and "ProductModelID". Rename the "Name" column by setting its "Name as" value to "Product".

9. Repeat steps 5-7 for the "ADLS_ProductCategory" dataset:

- Add a source (using the "Add Source" tile displayed on the data flow canvas beneath the Product source transformation)
- Set its dataset to "ADLS_ProductCategory", import the file's schema and check that ProductCategoryID is of type "short"
- Add a select transformation, rename it and ensure auto-mapping is disabled
- Remove all columns except "ProductCategoryID" and "Name". Rename the "Name" field to "ProductCategory".

10. Repeat steps 5-7 using the “ADLS_ProductModel” dataset, checking that column “ProductModelID” is of type “short”. Use a Select transformation to remove all columns except “ProductModelID” and “Name”. Rename “Name” to “ProductModel”.

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



Output stream name * SelectProductModel

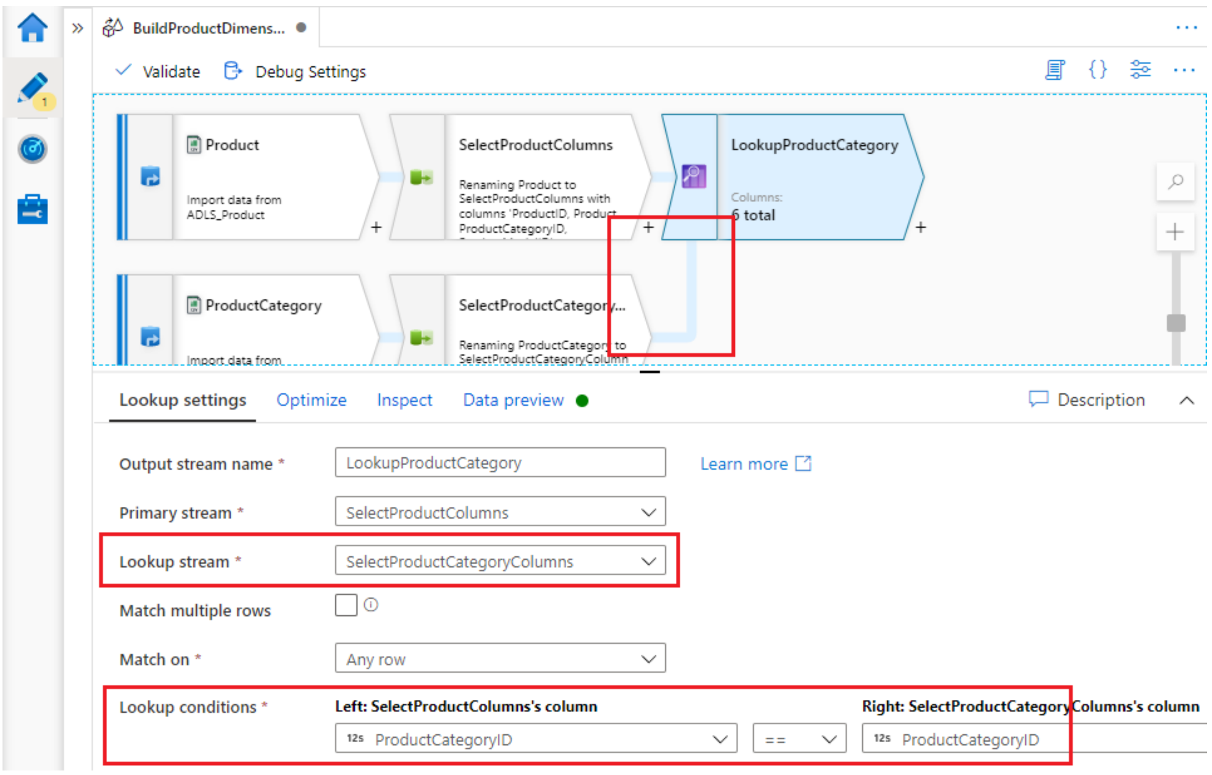
Incoming stream * ProductModel

Options ✓ Skip duplicate input columns

11. You can combine data into the Product stream from the other two streams using the Mapping Data Flow “Lookup” transformation. Click the small “+” button below the product column stream’s Select transformation, then select “Lookup” from the popup menu.

12. On the **Lookup settings** tab, name the transformation then set “Lookup stream” to use ProductCategory 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 lookup fields from each transformation and the operator to compare them – choose the streams’ respective ProductCategoryId fields. Notice that the lookup relationship also appears on the data flow canvas.



Output stream name * LookupProductCategory

Primary stream * SelectProductColumns

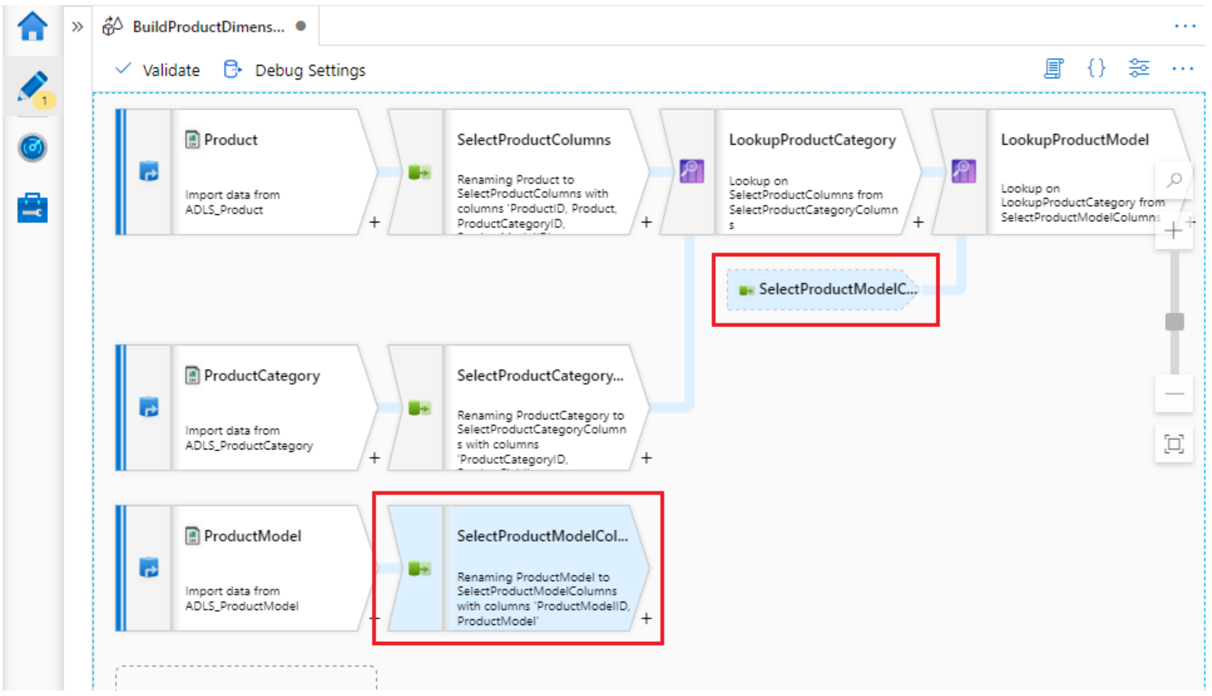
Lookup stream * SelectProductCategoryColumns

Match multiple rows

Match on * Any row

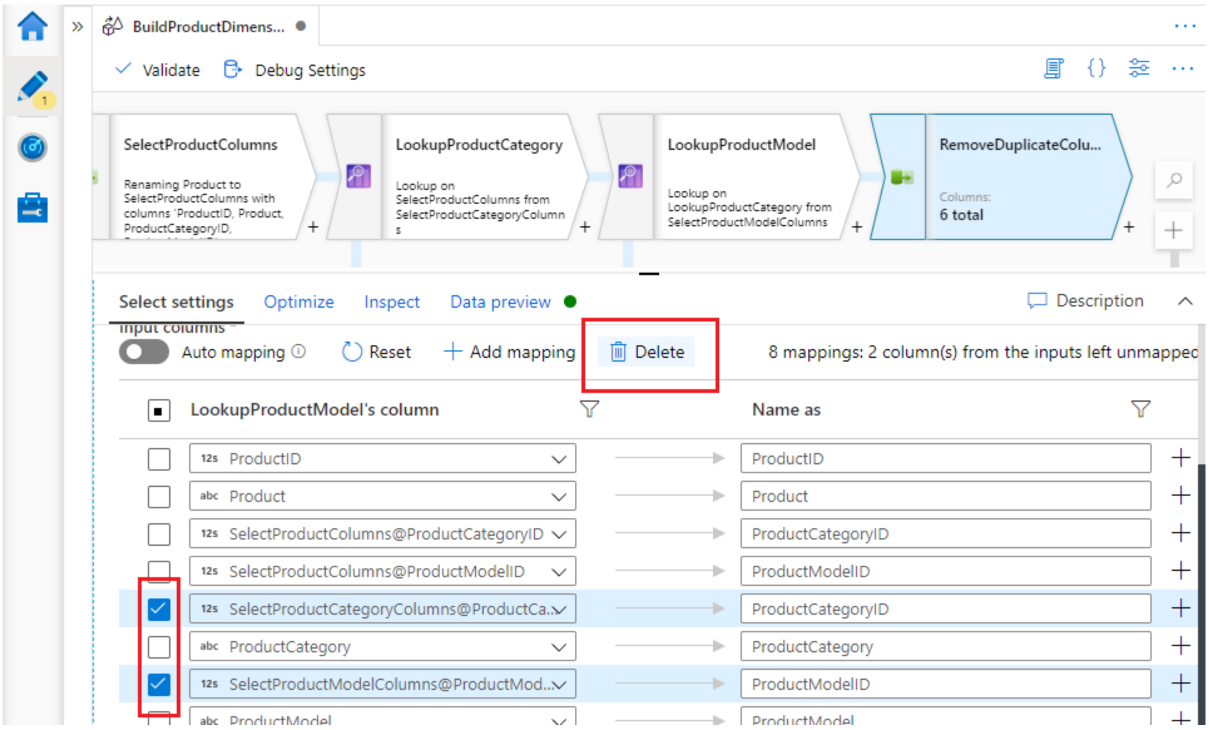
Lookup conditions * Left: SelectProductColumns's column ProductCategoryID == Right: SelectProductCategoryColumns's column ProductCategoryID

13. Add a second "Lookup" activity, also on the Product stream, this time performing a lookup against the ProductModel stream based on matching ProductModelId. This time the canvas displays a "reference node" instead of showing a direct link between the two transformations. This is just for readability – if you hover over the reference node or the transformation it refers to, both light up in blue to indicate that they mean the same thing.


BuildProductDimens...
Validate
Debug Settings
Product
Import data from ADLS_Product
SelectProductColumns
Renaming Product to SelectProductColumns with columns 'ProductID, Product, ProductCategoryID,
LookupProductCategory
Lookup on SelectProductColumns from SelectProductCategoryColumns
LookupProductModel
Lookup on LookupProductCategory from SelectProductModelColumns
SelectProductModelC...
ProductCategory
Import data from ADLS_ProductCategory
SelectProductCategory...
Renaming ProductCategory to SelectProductCategoryColumns with columns 'ProductCategoryID,
ProductModel
Import data from ADLS_ProductModel
SelectProductModelCol...
Renaming ProductModel to SelectProductModelColumns with columns 'ProductModelID, ProductModel'

14. 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.

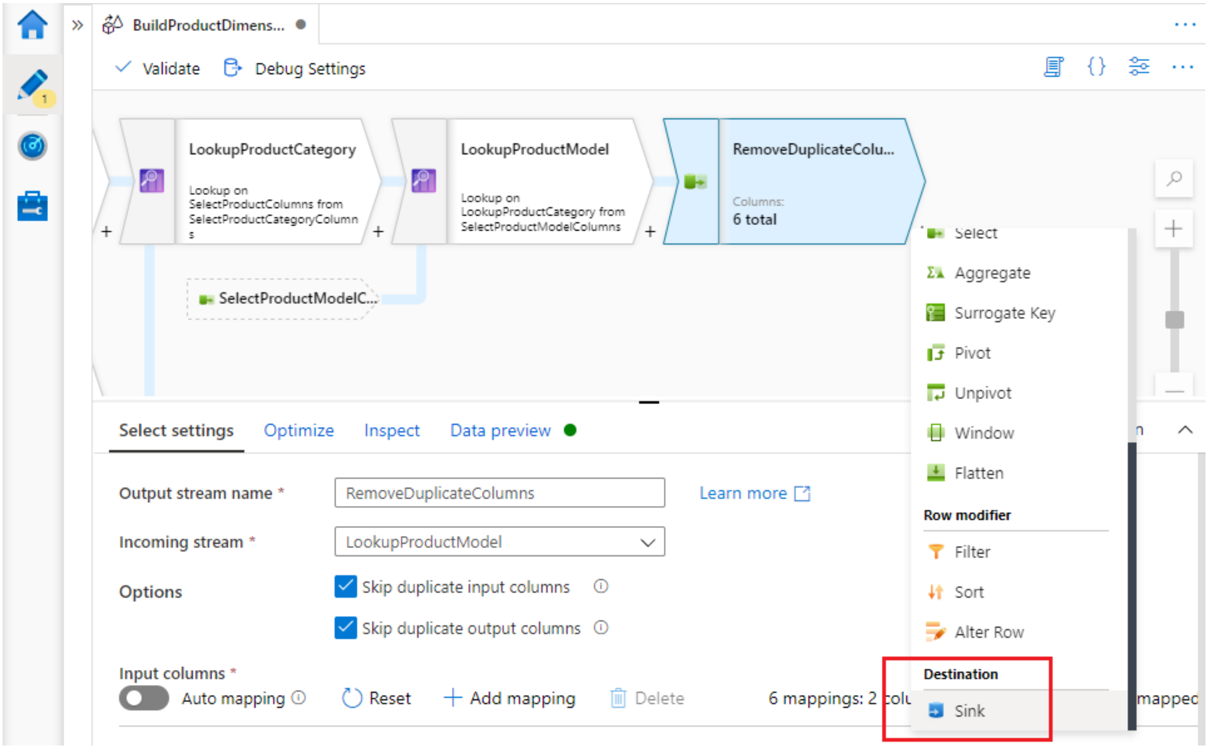
The Select transformation indicates the origin of each duplicated column by prefixing the column's name with that of the source transformation. Delete one duplicate column from each pair.



8 mappings: 2 column(s) from the inputs left unmapped

	LookupProductModel's column	Name as
☐	12s ProductID	ProductID
☐	abc Product	Product
☐	12s SelectProductColumns@ProductCategoryID	ProductCategoryID
☐	12s SelectProductColumns@ProductModelID	ProductModelID
✓	12s SelectProductCategoryColumns@ProductCa...	ProductCategoryID
☐	abc ProductCategory	ProductCategory
✓	12s SelectProductModelColumns@ProductMod...	ProductModelID
☐	abc ProductModel	ProductModel

15. 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.



Output stream name * RemoveDuplicateColumns

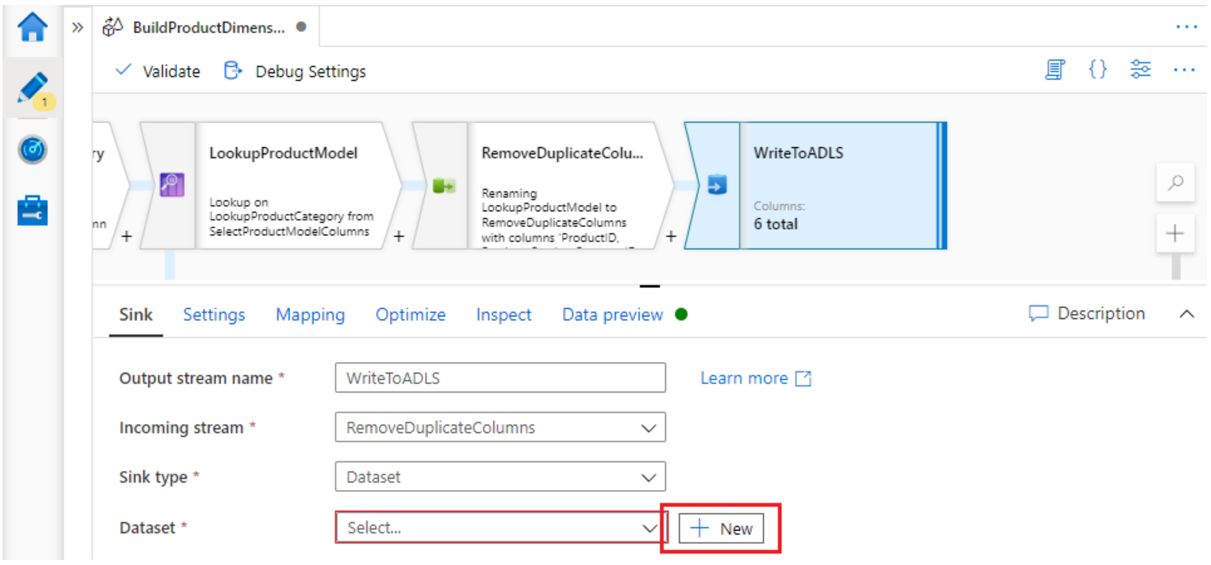
Incoming stream * LookupProductModel

Options ✓ Skip duplicate input columns ✓ Skip duplicate output columns

Input columns * Auto mapping

6 mappings: 2 colu... mapped

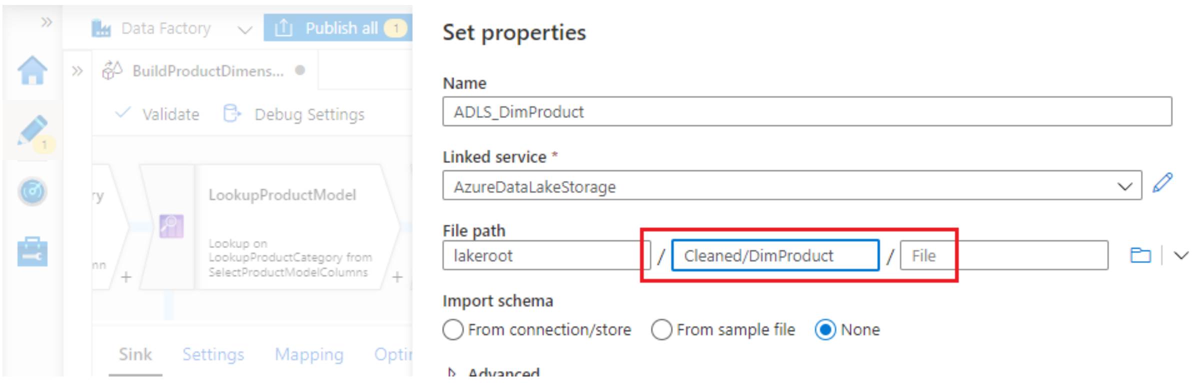
You haven't yet created a dataset to use as the data flow sink, but you can do so directly from the Sink transformation by clicking the "+ New" button. This opens the "New dataset" blade, familiar from earlier labs.



Output stream name * WriteToADLS
Incoming stream * RemoveDuplicateColumns
Sink type * Dataset
Dataset * Select...

16. Select data store type "Azure Data Lake Storage Gen2" and select the "Parquet" file format. Parquet is a column-oriented, highly-compressible file format, offering significant performance benefits for data lakes.

   - Choose your data lake linked service, then specify a file location. I'm writing the dimension into the "Cleaned" folder of my "lakeroot" container, to reflect the fact that this dataset has passed beyond the raw state of its source files.
   - Parquet is a multi-file storage format, so the dataset will not accept a file name – I've specified folder path "Cleaned/DimProduct" instead, so that the dimension's Parquet files are written into a directory with a descriptive name.
   - Set "Import schema" to "None".



Set properties

Name: ADLS_DimProduct

Linked service * AzureDataLakeStorage

File path: lakeroot / Cleaned/DimProduct / File

Import schema: From connection/store From sample file [✓ None]

Click "OK" to create your dataset, then save/publish your changes.

Lab 4.5 – Run the Mapping Data Flow

Mapping data flows are executed within an ADF pipeline. To run your data flow, create a pipeline for it.

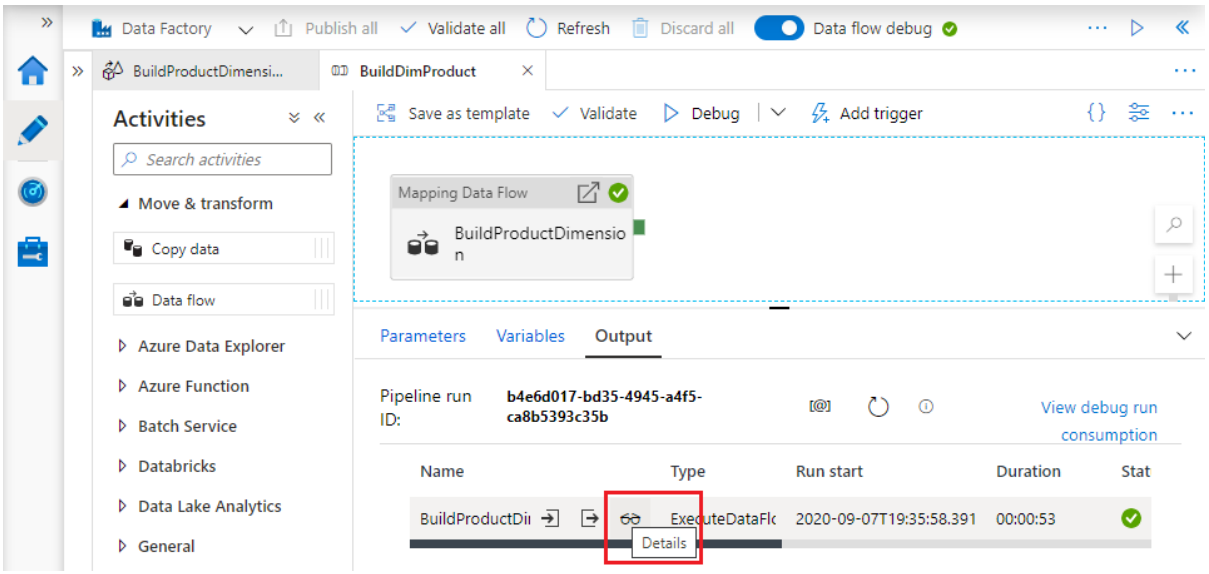
1. Create a new ADF pipeline.

2. Expand the "Move & transform" group in the activity toolbox, then drag a "Data flow" activity onto the pipeline canvas. When prompted, select "Use existing data flow" and select your new data flow from the "Existing data flow" dropdown. Click OK.

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.

   A cluster is provisioned on demand to run published pipelines – you can publish and trigger your data flow now if you wish, but be prepared for a few minutes' delay while the cluster is prepared in the published environment. This may feel cumbersome for a workload the size of the product dimension – in the real world, mapping data flows are designed to support workloads that are considerably more demanding.

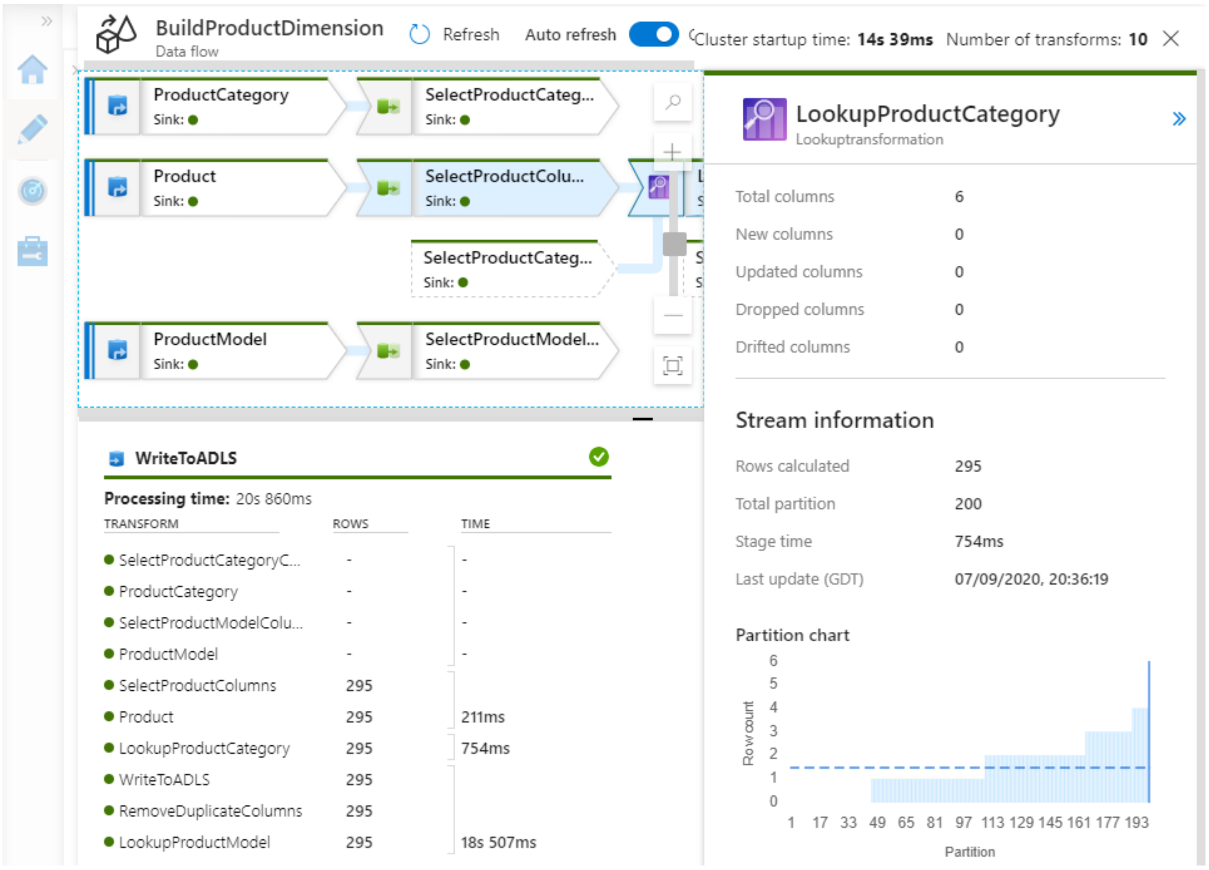
4. When the pipeline has finished running, a row of data appears in its "Output" pane for the Data flow activity. Hover over the activity's name to reveal the "Details" button ("glasses" icon).



Pipeline run ID: b4e6d017-bd35-4945-a4f5-ca8b5393c35b

Name	Type	Run start	Duration	Stat
BuildProductDi	ExecuteDataFlc	2020-09-07T19:35:58.391	00:00:53	✓

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.



Cluster startup time: **14s 39ms** Number of transforms: **10**

WriteToADLS

Processing time: 20s 860ms

TRANSFORM	ROWS	TIME
● SelectProductCategoryC...	-	-
● ProductCategory	-	-
● SelectProductModelColu...	-	-
● ProductModel	-	-
● SelectProductColumns	295	
● Product	295	211ms
● LookupProductCategory	295	754ms
● WriteToADLS	295	
● RemoveDuplicateColumns	295	
● LookupProductModel	295	18s 507ms

Total columns	6
New columns	0
Updated columns	0
Dropped columns	0
Drifted columns	0

Stream information

Rows calculated	295
Total partition	200
Stage time	754ms
Last update (GDT)	07/09/2020, 20:36:19

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, use an ADF Copy data activity's Source tab to access the ADLS_DimProduct dataset and preview its contents.

Lab 4.6 – Further work

This lab introduced concepts essential to the creation of a basic ADF Mapping 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.

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 a mapping data flow
- used Data flow debug to import file schemas (using Import projection)
- created a pipeline to execute your data
- used Data flow debug to run the pipeline from the ADF UX.