

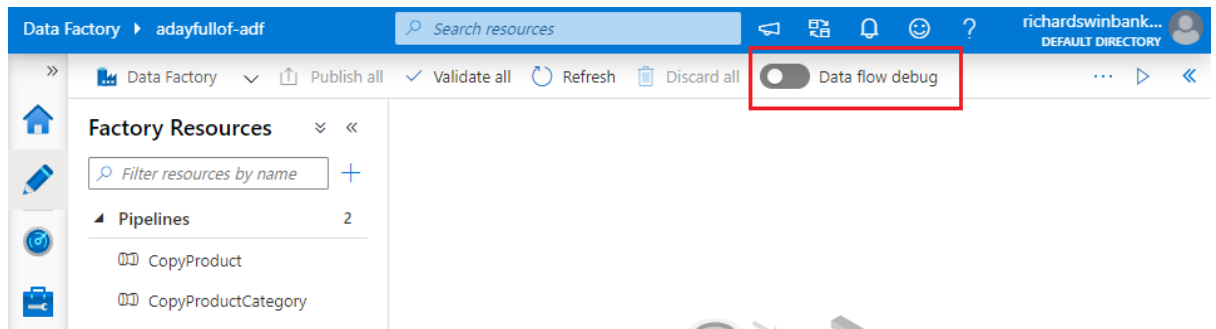
# Lab 4 – Build a Mapping Data Flow

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.

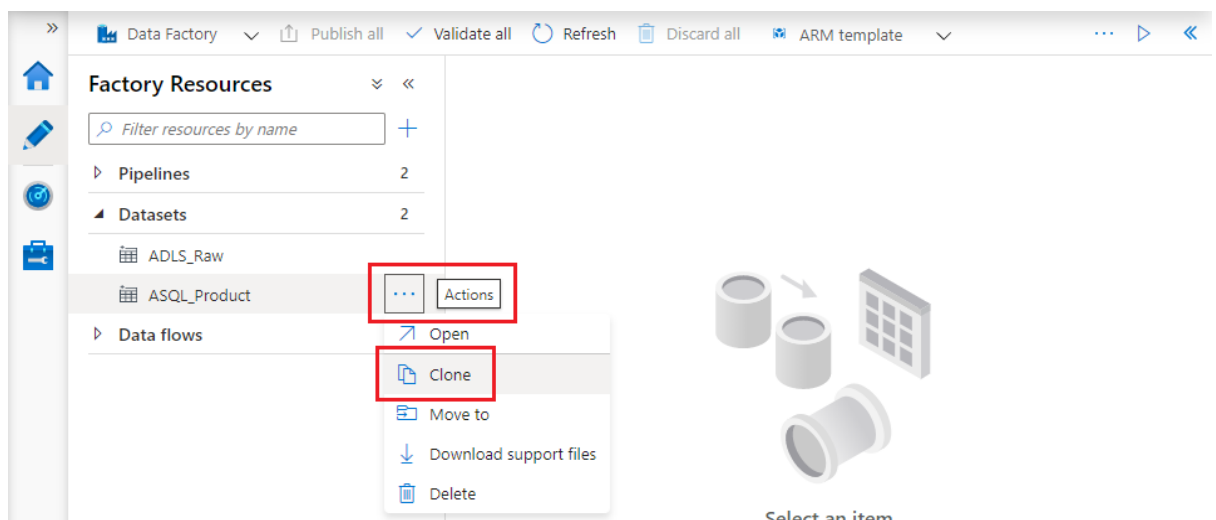
## Lab 4.2 – Copy source data to the data lake

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

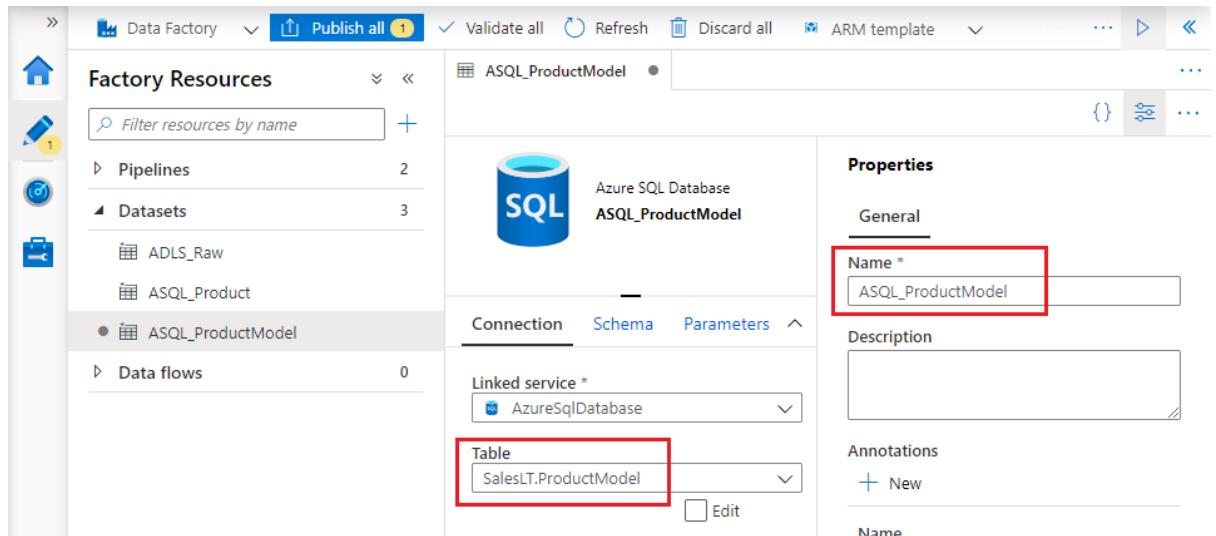
- [SalesLT].[Product]
- [SalesLT].[ProductCategory]
- [SalesLT].[ProductModel]

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”.



- 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.

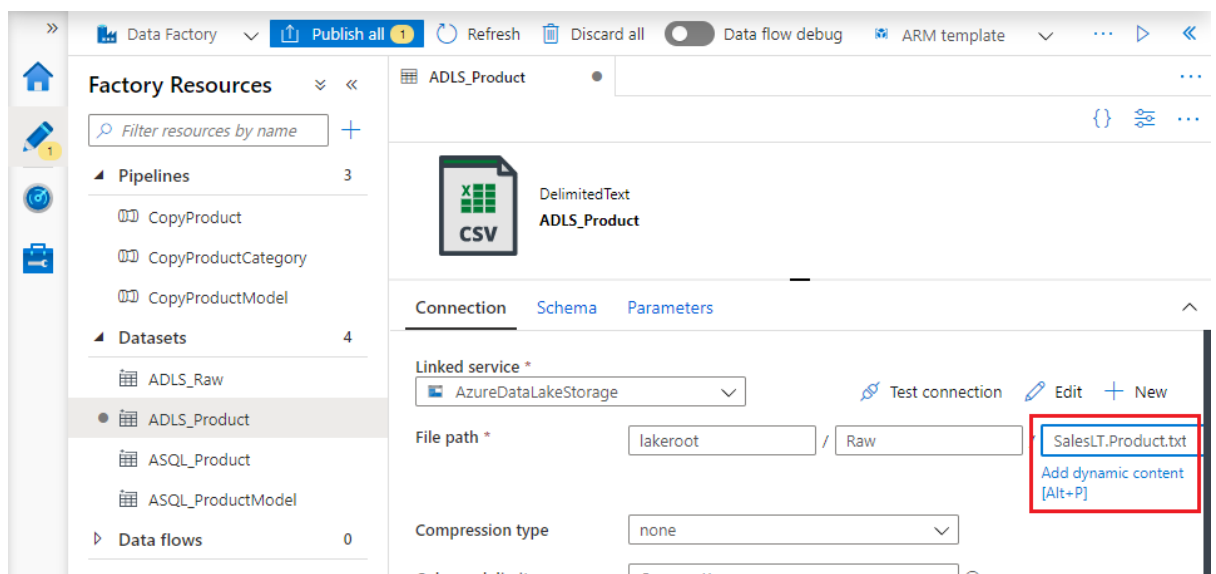


- Create a new pipeline in the same way as Lab 2.3, using a Copy data activity with the new “ASQL\_ProductModel” dataset as source and your Azure Data Lake Storage dataset as sink. Save your changes.
- Run the pipeline in debug mode and verify that file “SalesLT.ProductModel.txt” has been created in 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.

- 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.



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

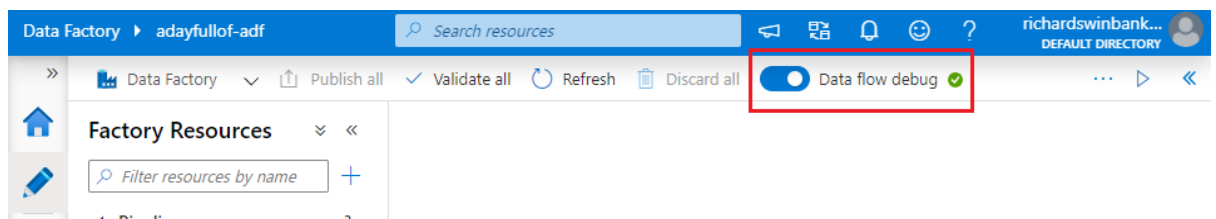
The product dimension combines product, model and category information to support different aggregations of facts that have a product attribute. This SQL query combines this information within the [AdventureWorks] database:

**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
```

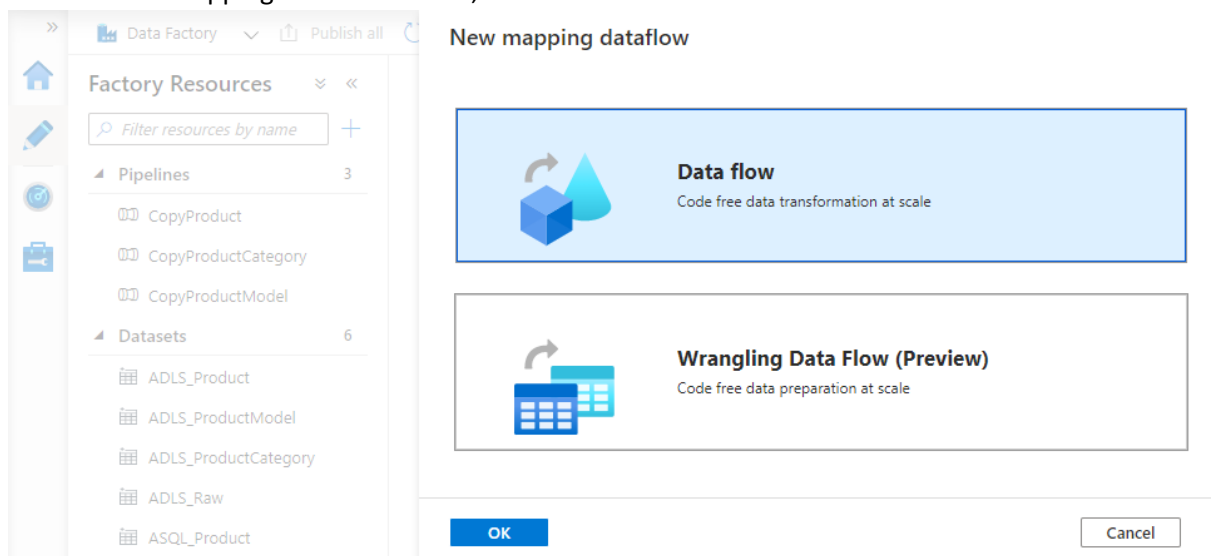
In this section you will use a Mapping Data Flow to reproduce this effect in Azure Data Factory.

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.

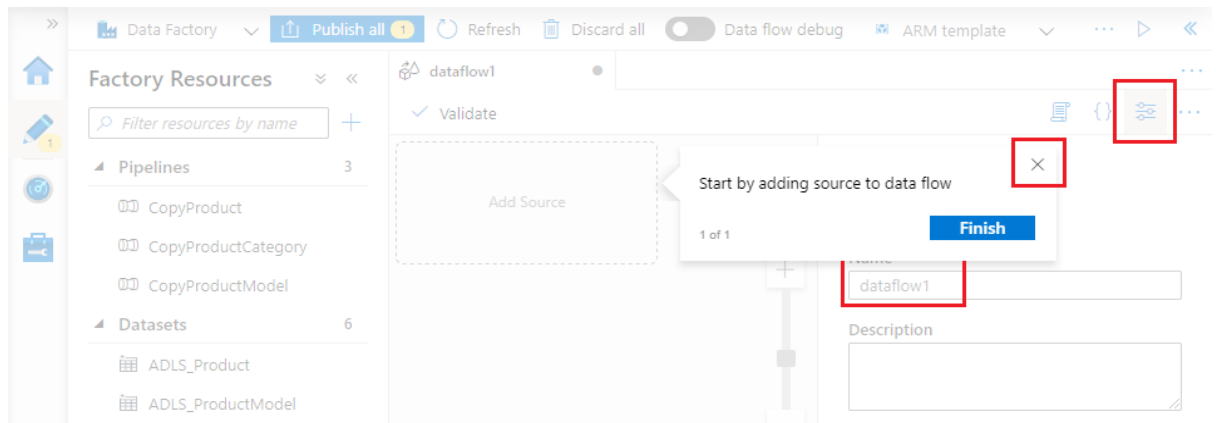


If the cluster is not ready yet, wait for it to finish warming up.

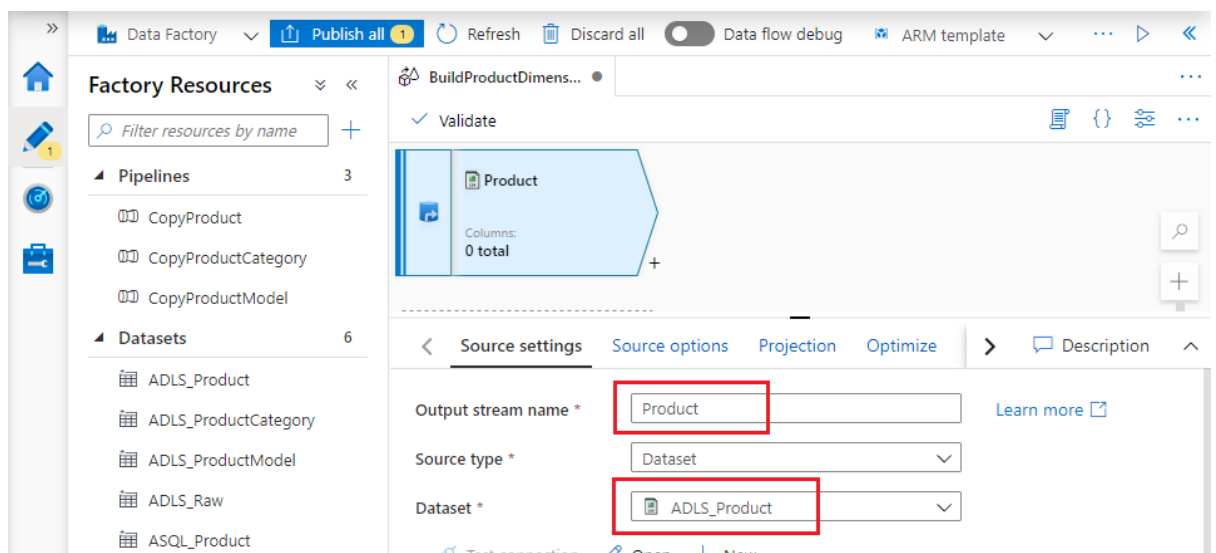
2. In the “Factory resources” sidebar, click the “+” button to the right of “Filter resources by name”, then choose “Data flow”.
3. On the “New mapping dataflow” blade, select the “Data Flow” tile. Click “OK”.



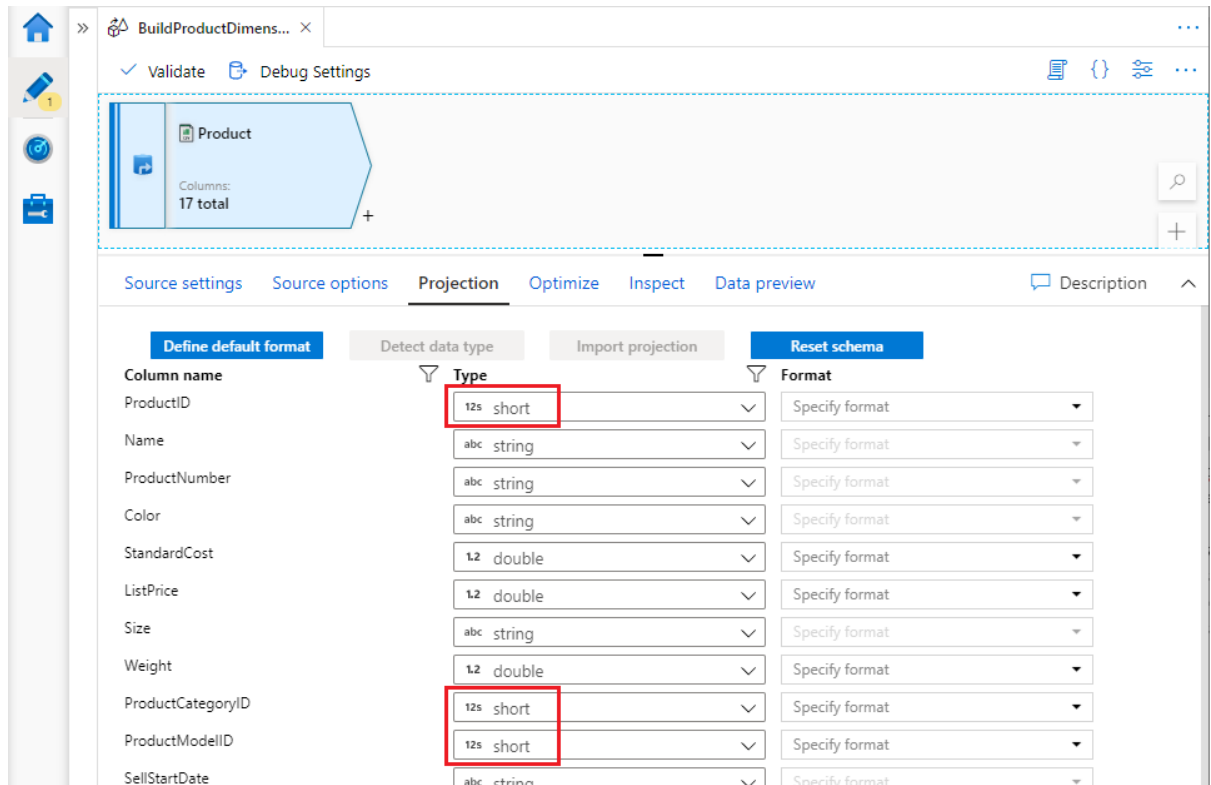
- The data flow canvas opens, displaying the callout “Start by adding source to data flow”. Dismiss the callout using its close button, then replace the data flow default name (“dataflow1”) with something more descriptive. Use the “Properties” slider button to close the data flow properties blade.



- 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.

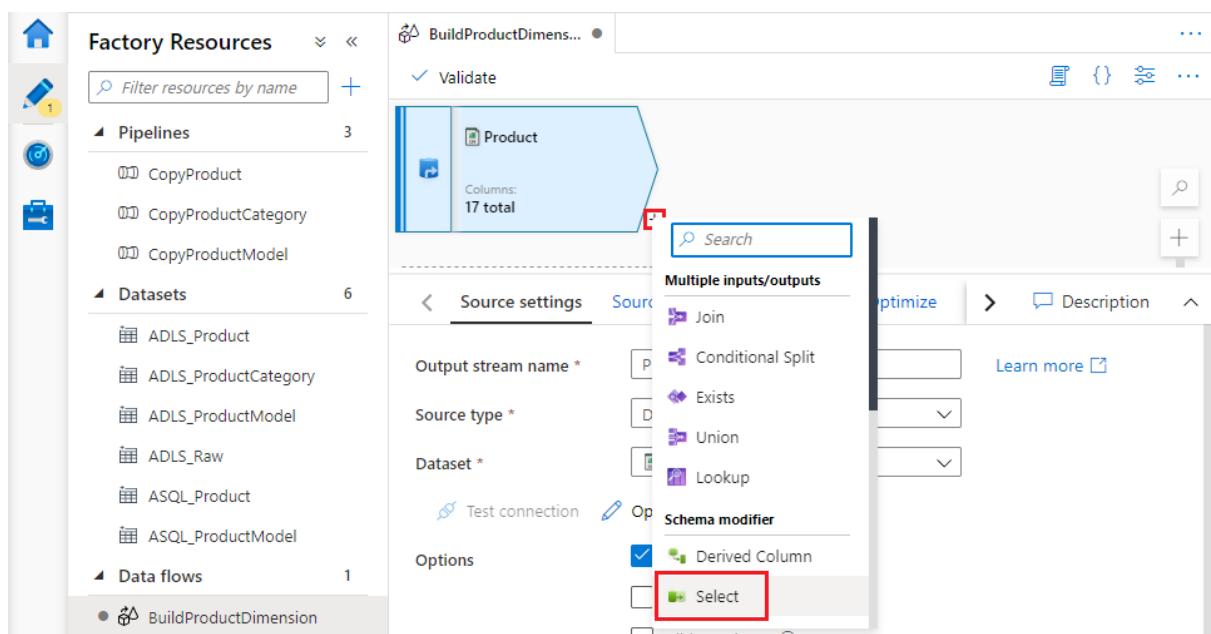


- On the **Projection** tab, click “Import projection” to import the source file’s schema. Check carefully that the type of the three fields “ProductID”, “ProductCategoryID” and “ProductModelID” has correctly been inferred as “short” – if this is not the case, use the “Type” dropdown to make the necessary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



Column name	Type	Format
ProductID	12s short	Specify format
Name	abc string	Specify format
ProductNumber	abc string	Specify format
Color	abc string	Specify format
StandardCost	1.2 double	Specify format
ListPrice	1.2 double	Specify format
Size	abc string	Specify format
Weight	1.2 double	Specify format
ProductCategoryID	12s short	Specify format
ProductModelID	12s short	Specify format
SellStartDate	abc string	Specify format

- 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.



Factory Resources

- Pipelines (3)
  - CopyProduct
  - CopyProductCategory
  - CopyProductModel
- Datasets (6)
  - ADLS\_Product
  - ADLS\_ProductCategory
  - ADLS\_ProductModel
  - ADLS\_Raw
  - ASQL\_Product
  - ASQL\_ProductModel
- Data flows (1)
  - BuildProductDimension

BuildProductDimens... Source settings

Output stream name \* [P]

Source type \* [D]

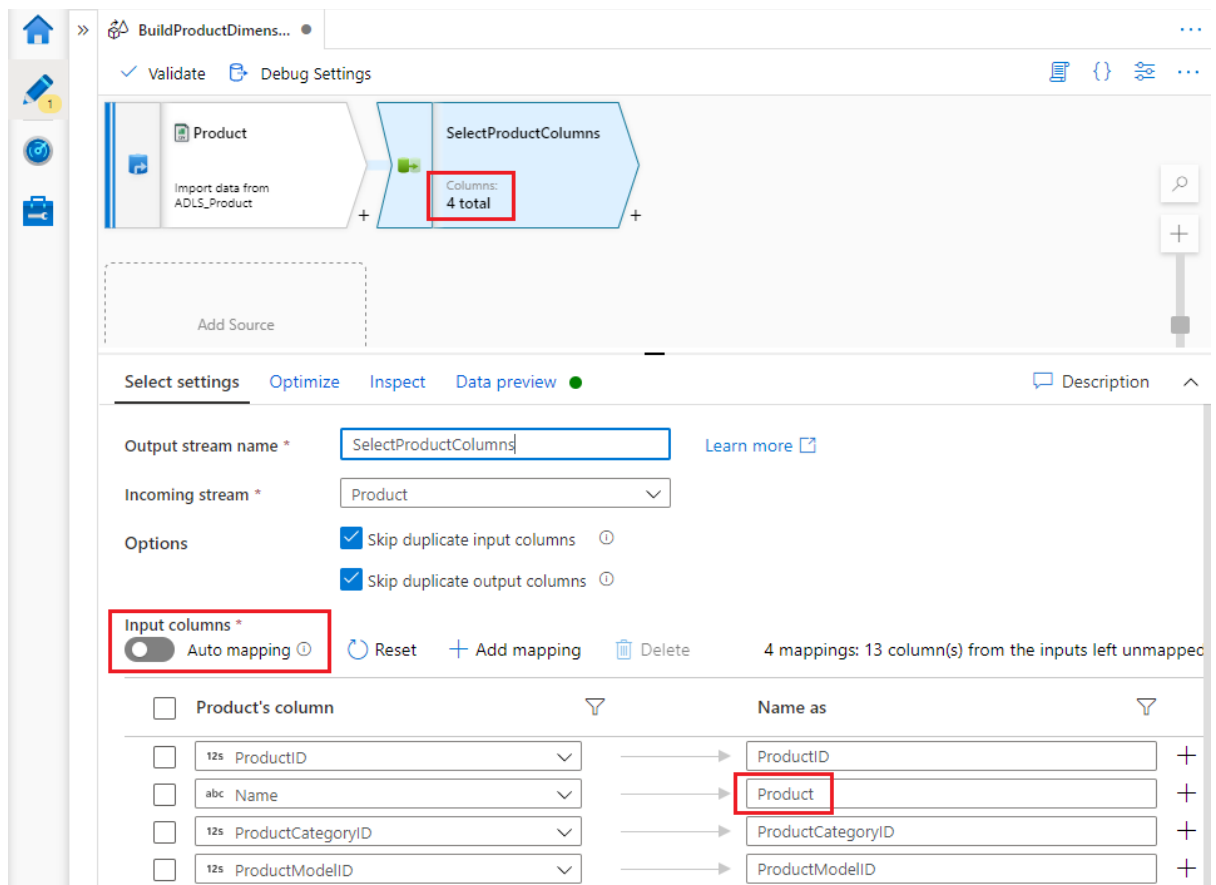
Dataset \* [P]

Test connection [icon]

Options

- Multiple inputs/outputs
  - Join
  - Conditional Split
  - Exists
  - Union
  - Lookup
- Schema modifier
  - Derived Column
  - Select
  - Validate schema [icon]

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



The screenshot shows the 'Select settings' tab for the 'SelectProductColumns' transformation. The 'Output stream name' is set to 'SelectProductColumns'. The 'Incoming stream' is 'Product'. The 'Options' section has 'Skip duplicate input columns' and 'Skip duplicate output columns' checked. The 'Input columns' section is expanded, showing a table with columns being mapped from the 'Product' source to the output stream. The 'Name as' column is highlighted, showing the mapping of 'Product' to 'Product'.

Product's column	Name as
12s ProductID	ProductID
abc Name	Product
12s ProductCategoryID	ProductCategoryID
12s ProductModelID	ProductModelID

The Select transformation enables you to rename, reorder or remove columns from a stream. Ensure that the "Auto mapping" setting is disabled 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".

- 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".

- 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.

- 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.
- 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.

BuildProductDimens...

Validate Debug Settings

Product: Import data from ADLS\_Product

SelectProductColumns: Renaming Product to SelectProductColumns with columns 'ProductID, ProductCategoryID.

LookupProductCategory: Columns: 6 total

ProductCategory: Import data from ADLS\_ProductCategory

SelectProductCategoryColumns: Renaming ProductCategory to SelectProductCategoryColumns with columns 'ProductCategoryID.

Lookup settings Optimize Inspect Data preview Description

Output stream name \* LookupProductCategory [Learn more](#)

Primary stream \* SelectProductColumns

Lookup stream \* SelectProductCategoryColumns

Match multiple rows ☐ ①

Match on \* Any row

Lookup conditions \* Left: SelectProductColumns's column Right: SelectProductCategoryColumns's column

12s ProductCategoryID == 12s 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, ProductCategoryID.

LookupProductCategory: Lookup on SelectProductColumns from SelectProductCategoryColumns

LookupProductModel: Lookup on LookupProductCategory from SelectProductModelColumns

ProductCategory: Import data from ADLS\_ProductCategory

SelectProductCategoryColumns: Renaming ProductCategory to SelectProductCategoryColumns with columns 'ProductCategoryID.

ProductModel: Import data from ADLS\_ProductModel

SelectProductModelColumns: 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.



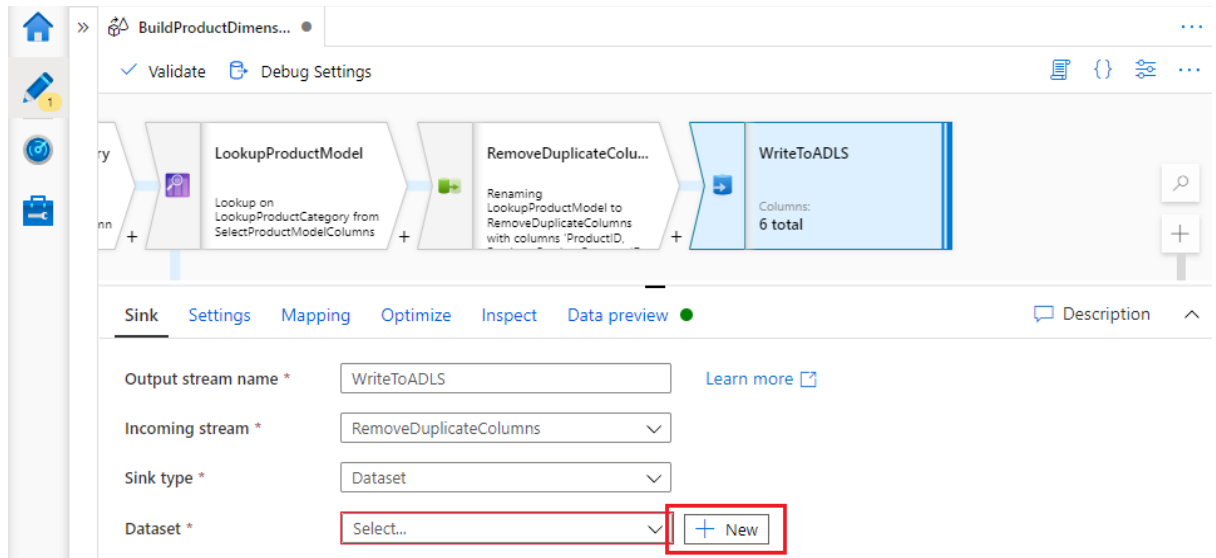
The Select transformation indicates the origin of each duplicated column by prefixing the column's name with that of the source transformation. Delete one duplicate column from each pair.

The screenshot shows the 'BuildProductDimens...' pipeline in the Azure Data Factory interface. The pipeline consists of four transformations: 'SelectProductColumns', 'LookupProductCategory', 'LookupProductModel', and 'RemoveDuplicateColumns'. The 'RemoveDuplicateColumns' transformation is selected, and its 'Input columns' tab is active. A table shows mappings from input columns to output columns. Two columns, '12s SelectProductCategoryColumns@ProductCa...' and '12s SelectProductModelColumns@ProductMod...', are highlighted with checkboxes. A red box highlights the 'Delete' button in the top right of the 'Input columns' tab.

- 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.

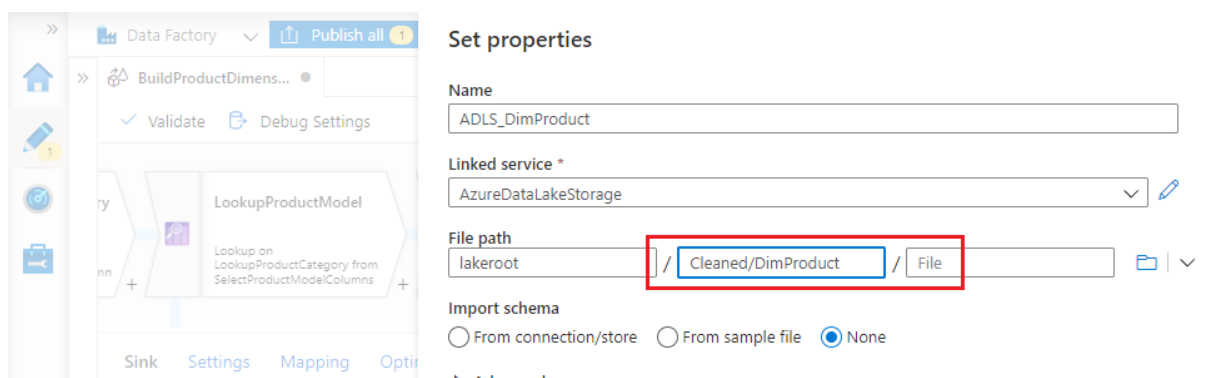
The screenshot shows the 'BuildProductDimens...' pipeline in the Azure Data Factory interface. The pipeline consists of four transformations: 'LookupProductCategory', 'LookupProductModel', 'RemoveDuplicateColumns', and 'SelectProductModelC...'. The 'SelectProductModelC...' transformation is selected, and its 'Select settings' tab is active. A dropdown menu is open, showing various transformation options. The 'Sink' option is highlighted with a red box.

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.



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".



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.

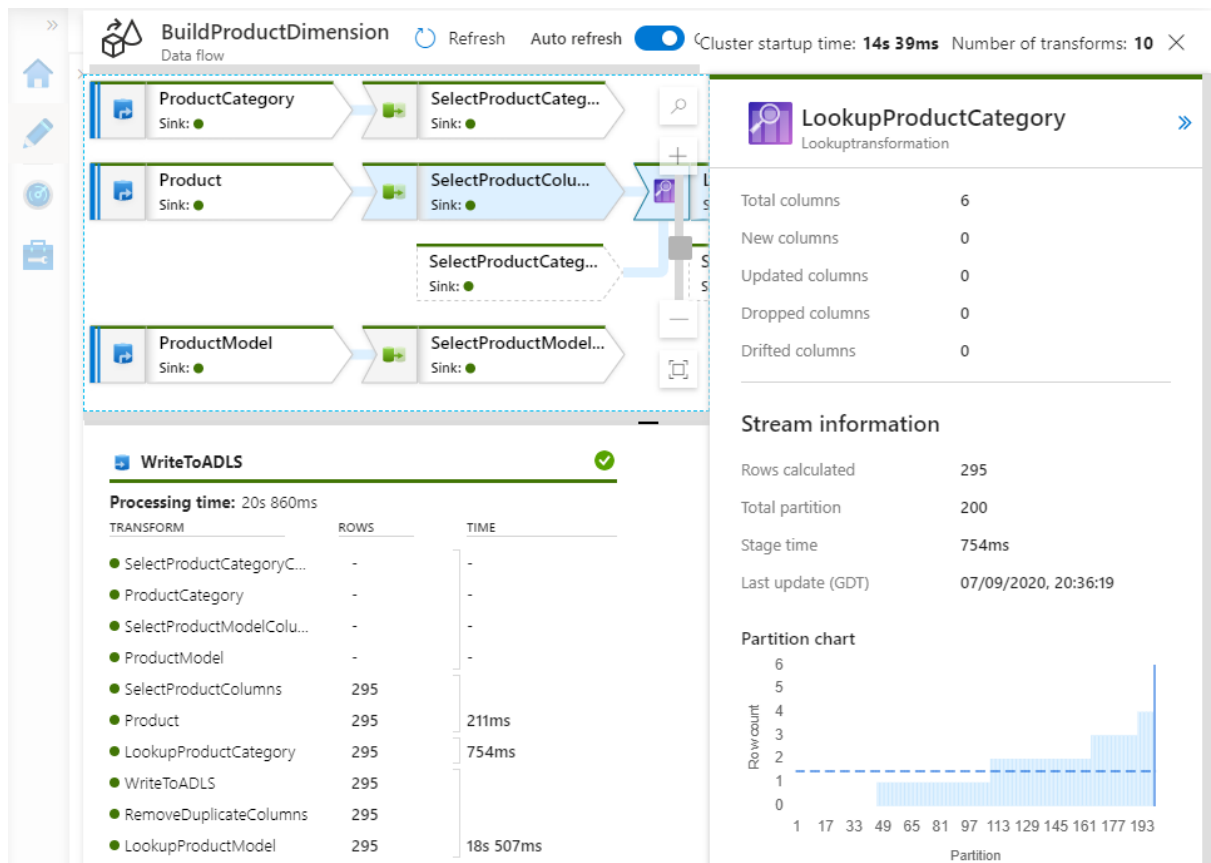
A cluster is provisioned on demand to run published pipelines – you can publish and trigger your data flow now if you wish, but be prepared for a few minutes’ delay while the cluster is prepared in the published environment. This may feel cumbersome for a workload the size of the product dimension – 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).

The screenshot shows the Azure Data Factory portal. On the left, the 'Activities' pane is expanded to 'Move & transform', showing 'Copy data' and 'Data flow'. The main canvas displays a pipeline named 'BuildDimProduct' with a 'Mapping Data Flow' activity. Below the canvas, the 'Output' tab is selected, showing a table with one row of data. A red box highlights the 'Details' button (glasses icon) next to the activity name.

Name	Type	Run start	Duration	Status
BuildProductDim	ExecuteDataFlow	2020-09-07T19:35:58.391	00:00:53	Success

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.



- 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.