**Creating Delta Lake in Azure ADLS Gen-2 Using Synapse Analytics**

**What is Delta lake?**

Delta Lake is an open-source storage layer that brings ACID (Atomicity, Consistency, Isolation, Durability) transactions to Apache Spark and big data workloads. It was developed by Databricks to enhance data reliability, quality, and performance for data lakes. Delta Lake can be used with various cloud storage systems, including Azure Data Lake Storage Gen2 (ADLS Gen2).

What is Delta log consists of?

In Delta Lake, the \_delta\_log folder is a crucial component that stores the transaction log and metadata information related to changes made to the Delta table. This folder is automatically created when you initialize a Delta table, and it plays a significant role in maintaining the transactional consistency, versioning, and reliability features provided by Delta Lake.

Here's an overview of what you can find in the \_delta\_log folder:

Transaction Log:

The primary purpose of the \_delta\_log folder is to store a transaction log. This log records all the transactions and changes made to the Delta table, including inserts, updates, deletes, and schema modifications.

The transaction log ensures atomicity, consistency, isolation, and durability (ACID properties) for operations on the Delta table.

It allows for crash recovery and rollbacks in case of failures during data modifications.

Checkpoint Files:

Delta Lake uses checkpoint files within the \_delta\_log folder to keep track of the state of the Delta table at certain points in time.

These checkpoint files are essential for efficiently maintaining the state of the Delta table and facilitating quick recovery and optimization during query processing.

Metadata Files:

The \_delta\_log folder also contains metadata files that capture information about the Delta table, such as schema evolution, table properties, and statistics.

Metadata files are crucial for supporting features like schema evolution, time travel queries, and providing a historical view of the table.

File Organization:

The \_delta\_log folder is organized into multiple subdirectories, each representing a version of the Delta table.

Within each version directory, you'll find JSON files that contain metadata and transaction information for that specific version.

**To create a Delta lake in ADLS GEN-2 by using Synapse Analytics we can use the below steps.**

Step 1: create ADLS GEN-2 for a particular resource group and also create a synapse analytics workspace for the same resource group. While creating the Synapse workspace the Azure will ask ADLS GEN 2 as primary workspace storage. So, point out the created ADLS GEN-2.

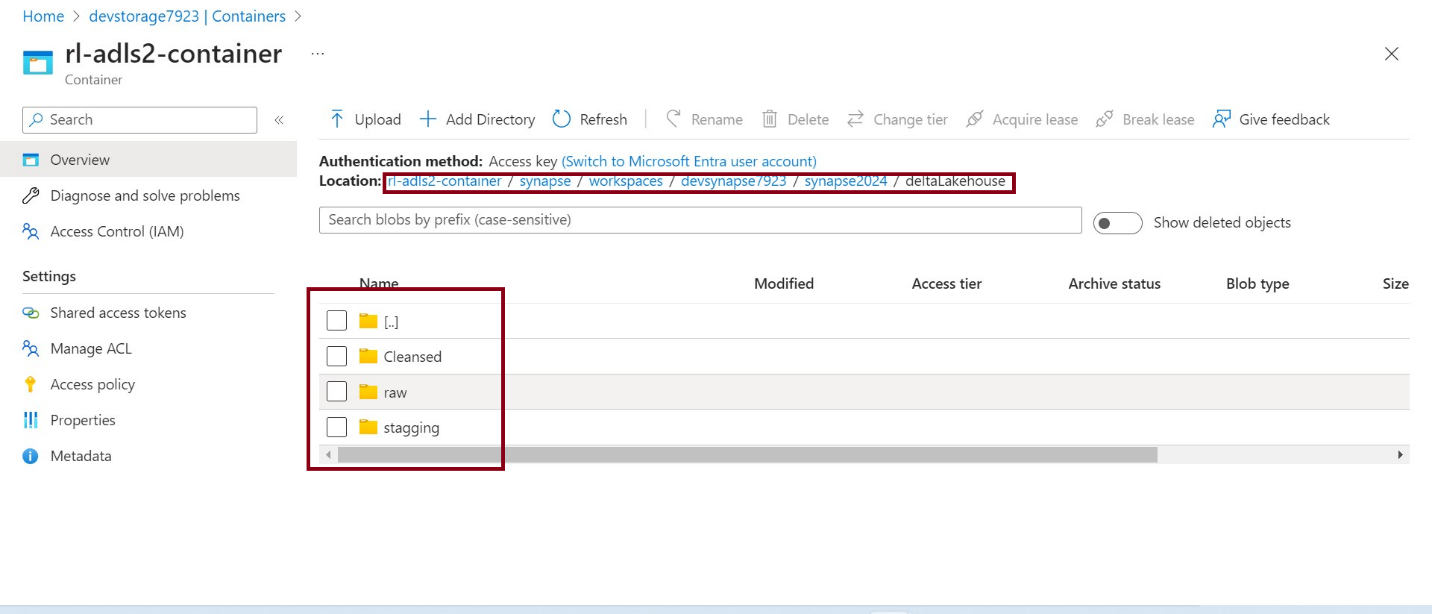
Once the Synapse workspace is created, we can create pipelines to load the data into Delta Lake (ADLS GEN-2).

Step 2: Create Delta lake like structure in the ADLS GEN-2 Container.

Folder structure:

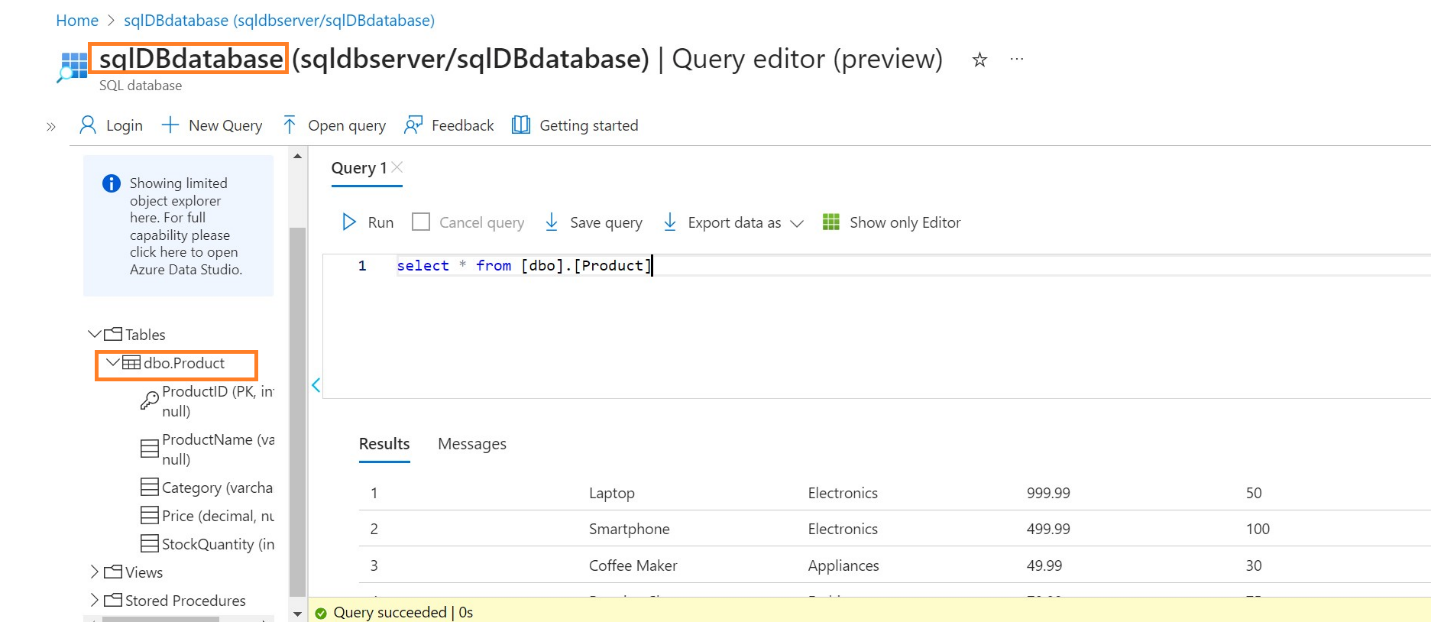
ADLS GEN-2 container -> create a root folder -> as the created Synapse workspace linked with ADLS GEN-2 default storage there will be workspace folder -> Delta lake structure.

dev / synapse / workspaces / devsynapse2024 / deltaLakehouse



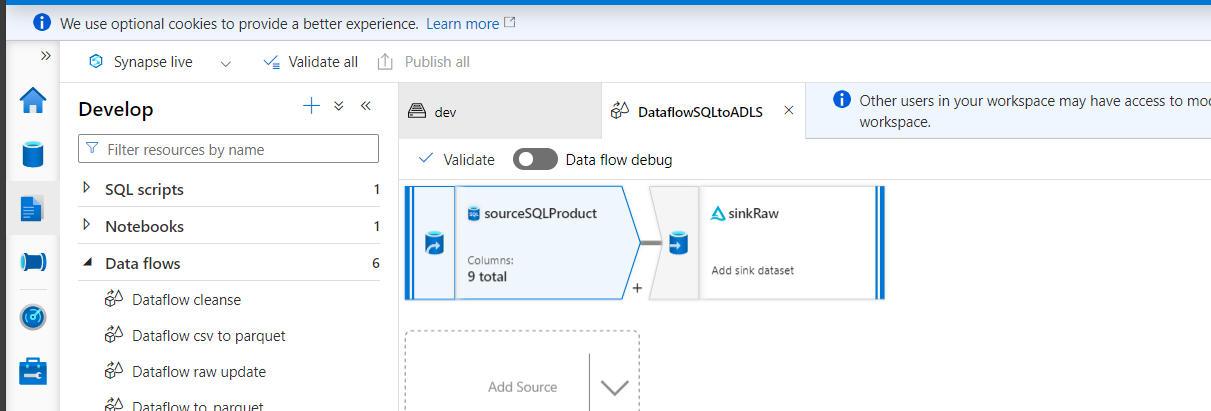
Step 3: As we are planning a scenario in Synapse analytics - Ingest data from SQL Database Table to raw folder -> raw to staging folder -> Cleansed folder

Product Table created in Azure SQL Database.

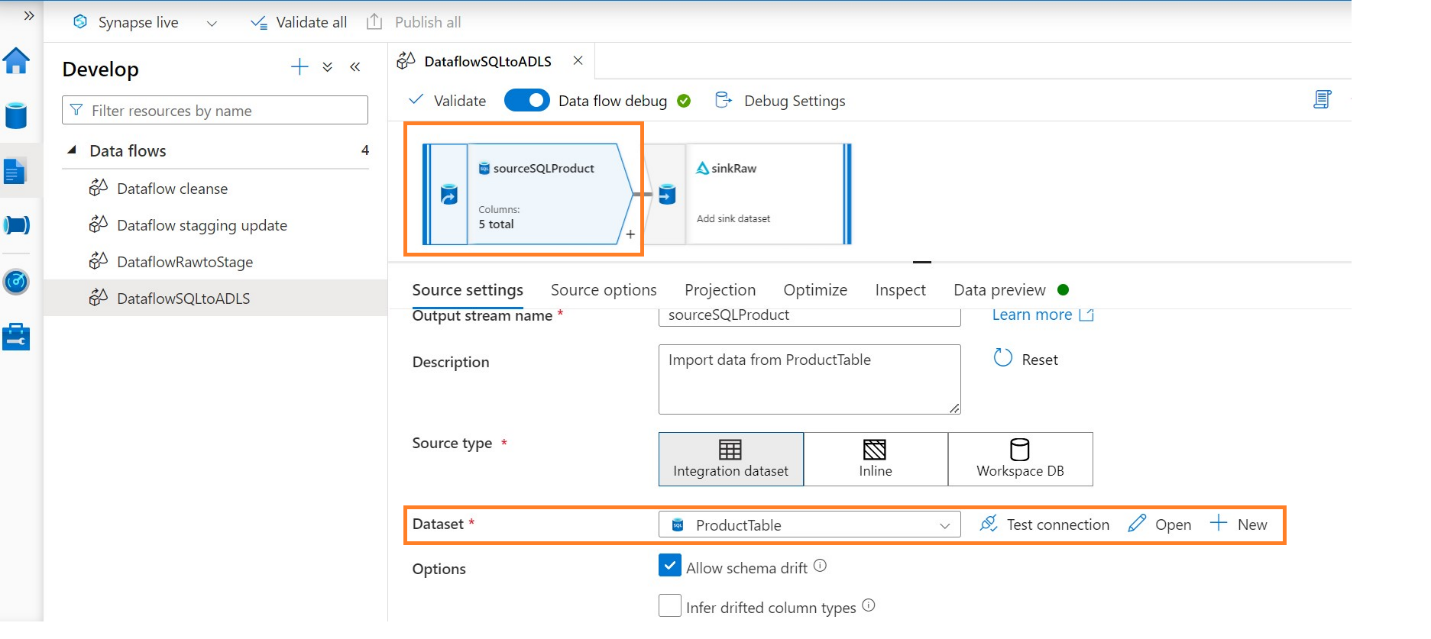


Now we need to create a data flow for loading data to raw delta folder from SQL DB table product.

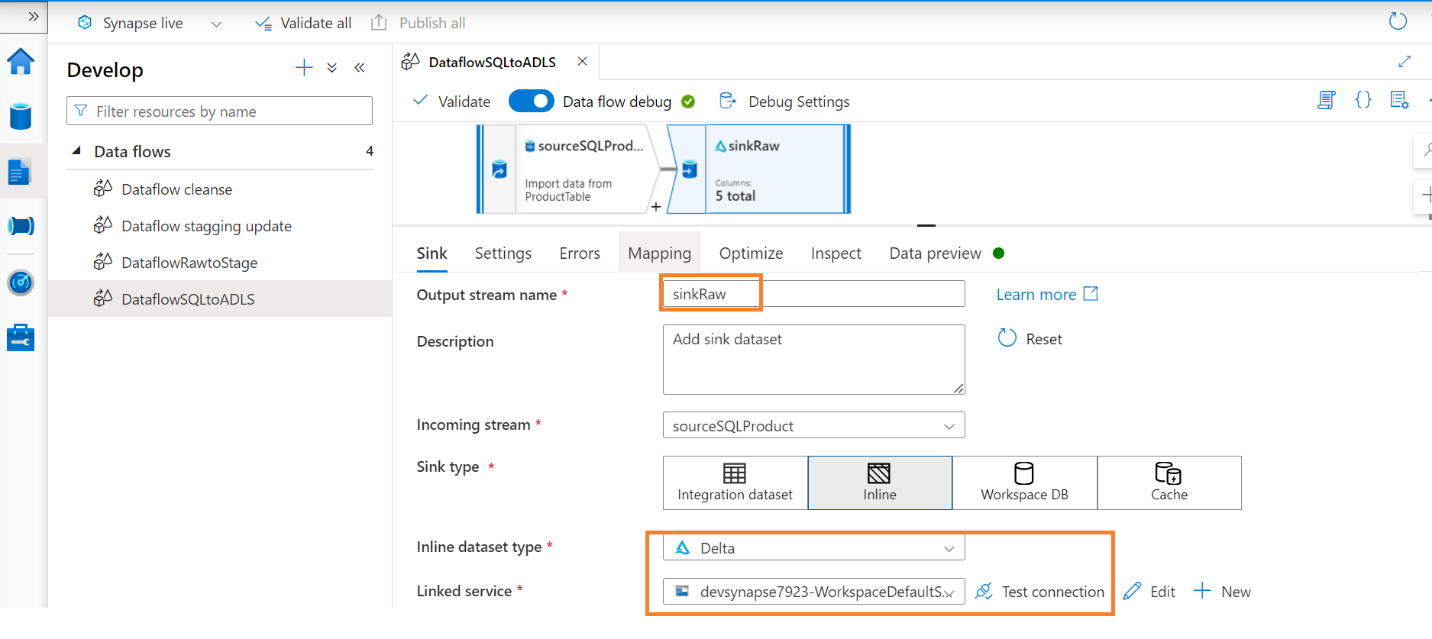
Data flow creation in Azure Synapse analytics:



Source component with SQL dataset pointing to Product Table:

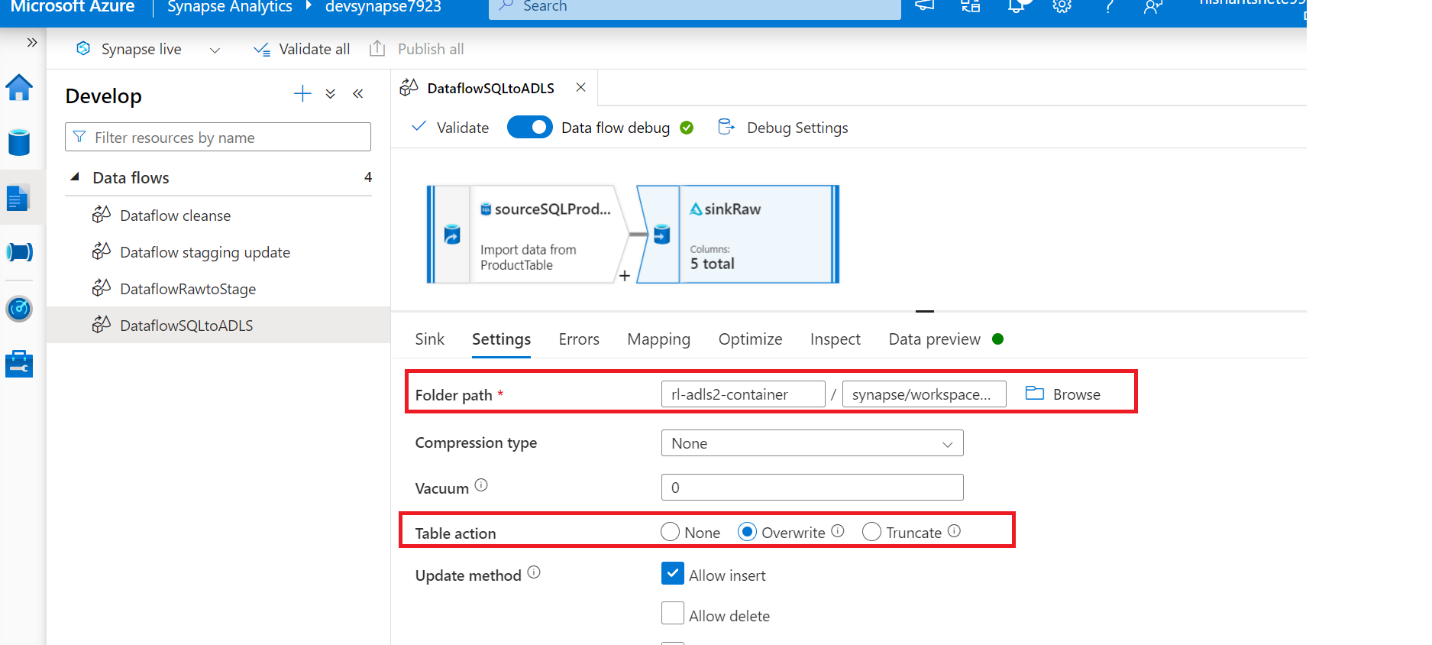


Sink configuration:



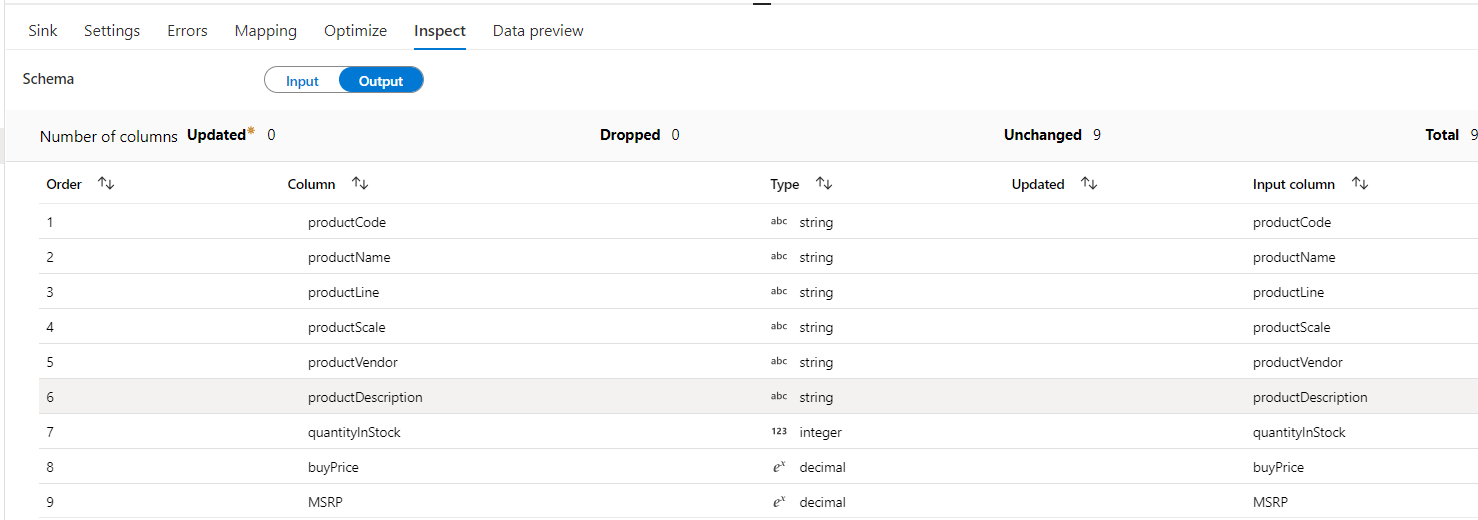
Here in sink usually we configure the sink type as Integration Dataset but as we going to create a delta lake then we need select the Inline option and once inline sink type is selected, select the Inline dataset type as Delta.

Also select Synapse workspace as the Linked Service, so the particular path of the folder can be pointed in sink settings.

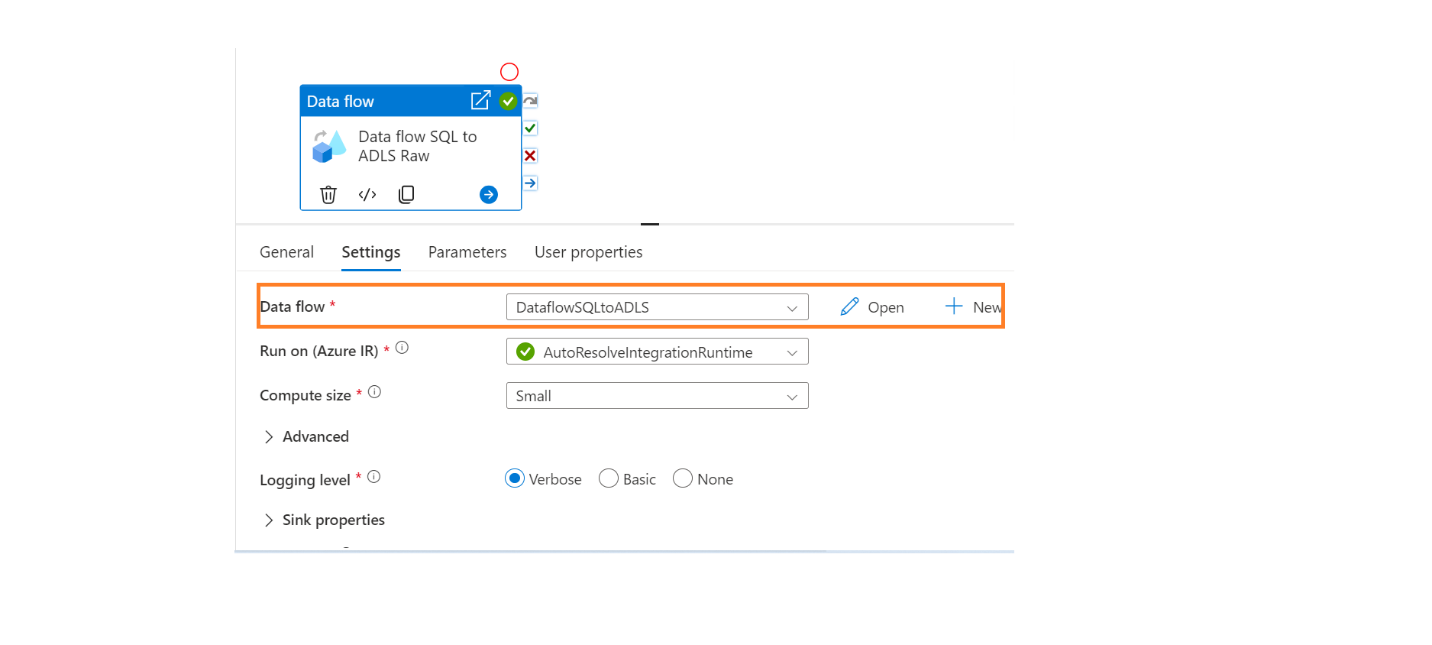


Here table action can be overwrite so that if the data is changed in the Table source then new file have the sync of data when in loaded to ADLS raw folder.

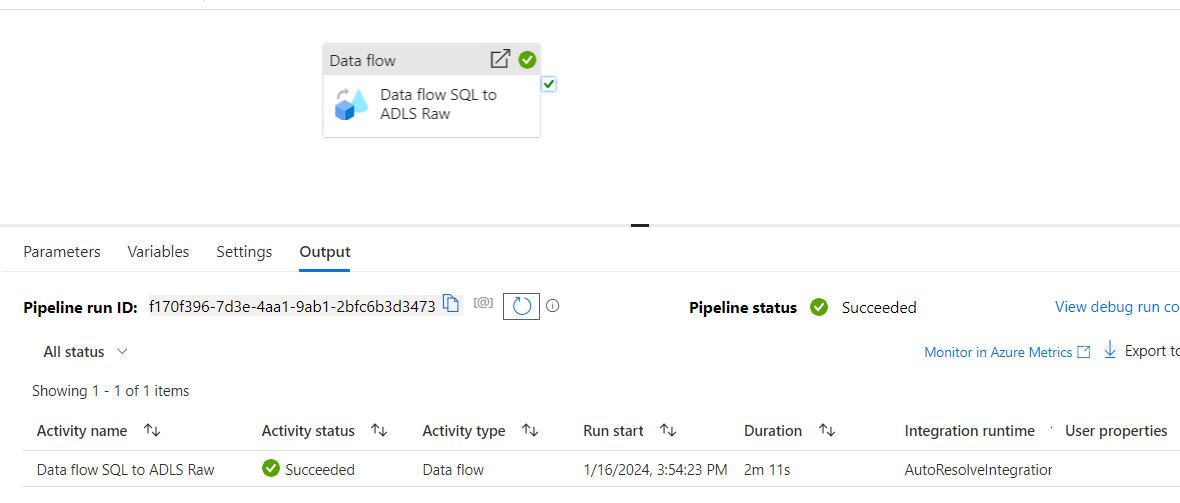
Inspect option can show us about the schema that should loaded as parquet file in Delta folder:



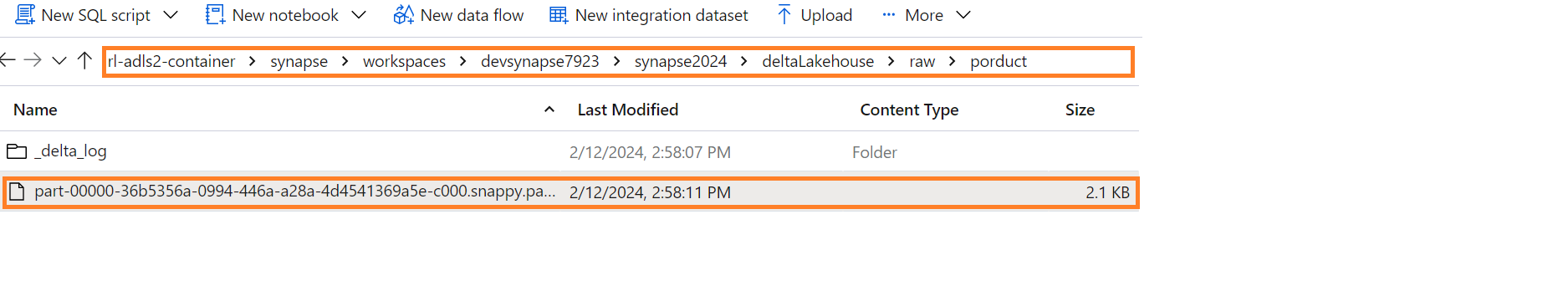
Now we can create a pipeline by configuring the above created data flow.



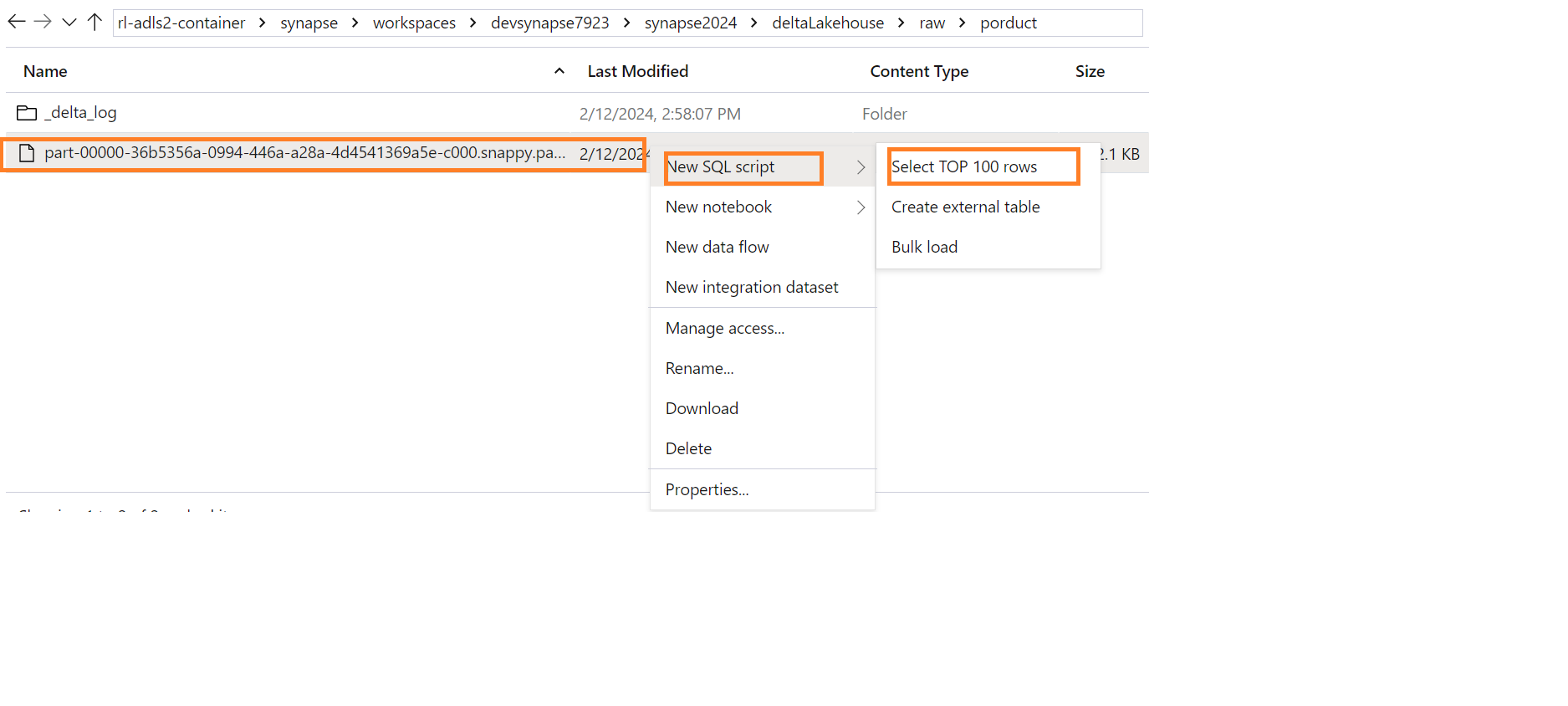
Once this pipeline is triggered we can see the loaded SQL product data into raw folder.



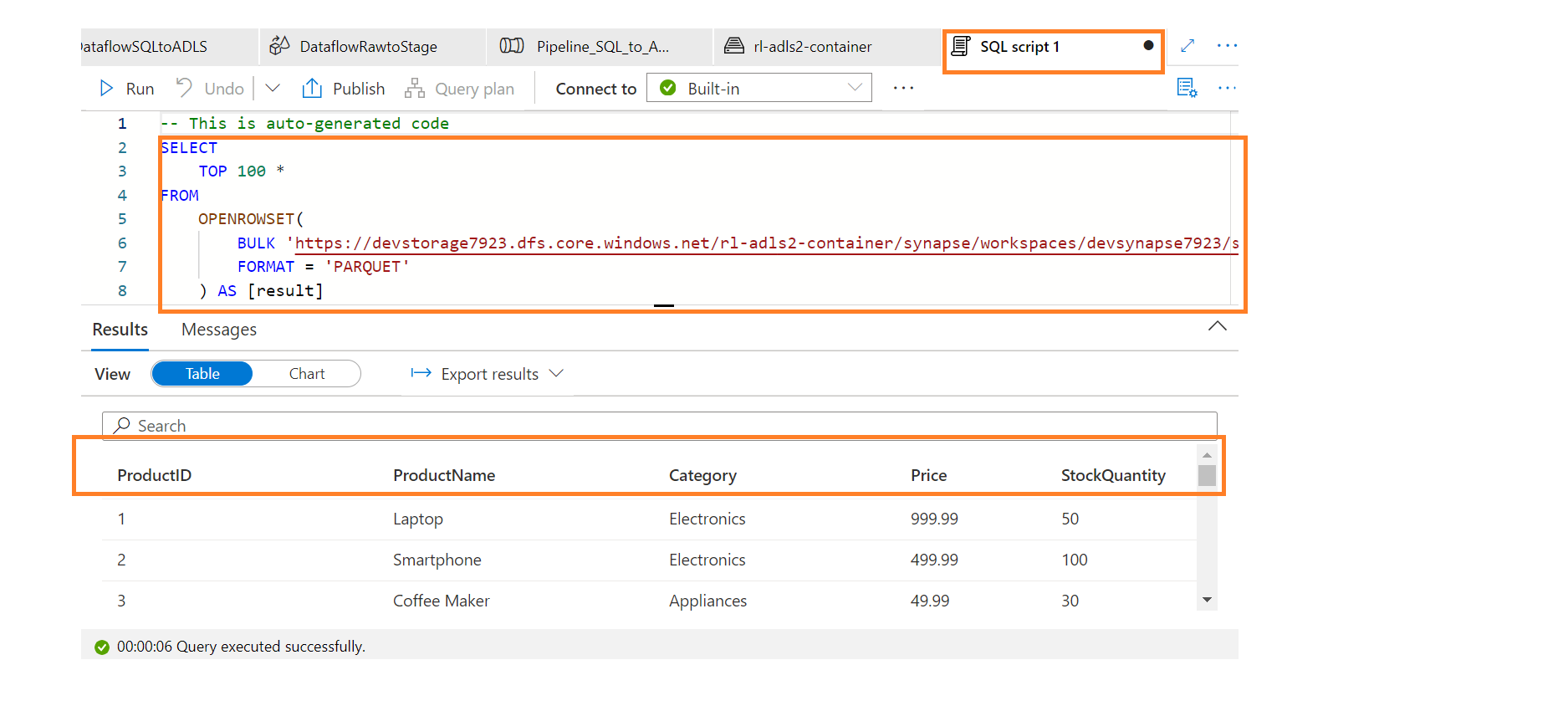
Raw delta folder:



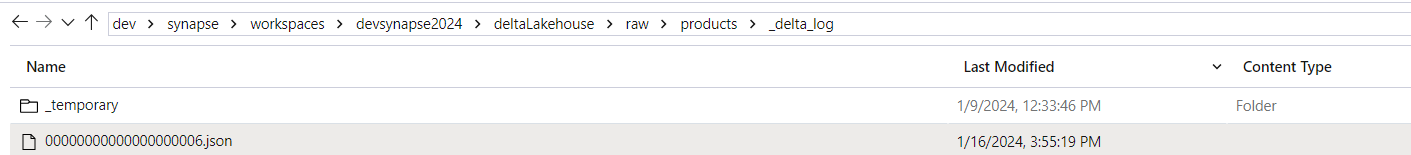
To validate the data or records we can use sql script option in Synapse workspace.



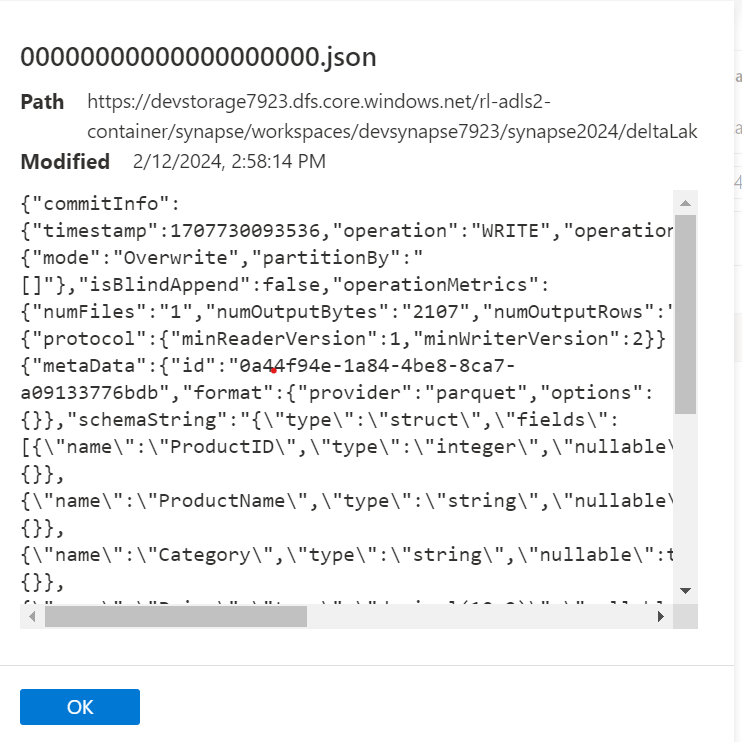
Once the sql script is executed we can see the records in row/column format:



Delta log:



Here delta log folder holds the json folder which has the transaction log, metadata and also few information related to changes made to the Delta table.

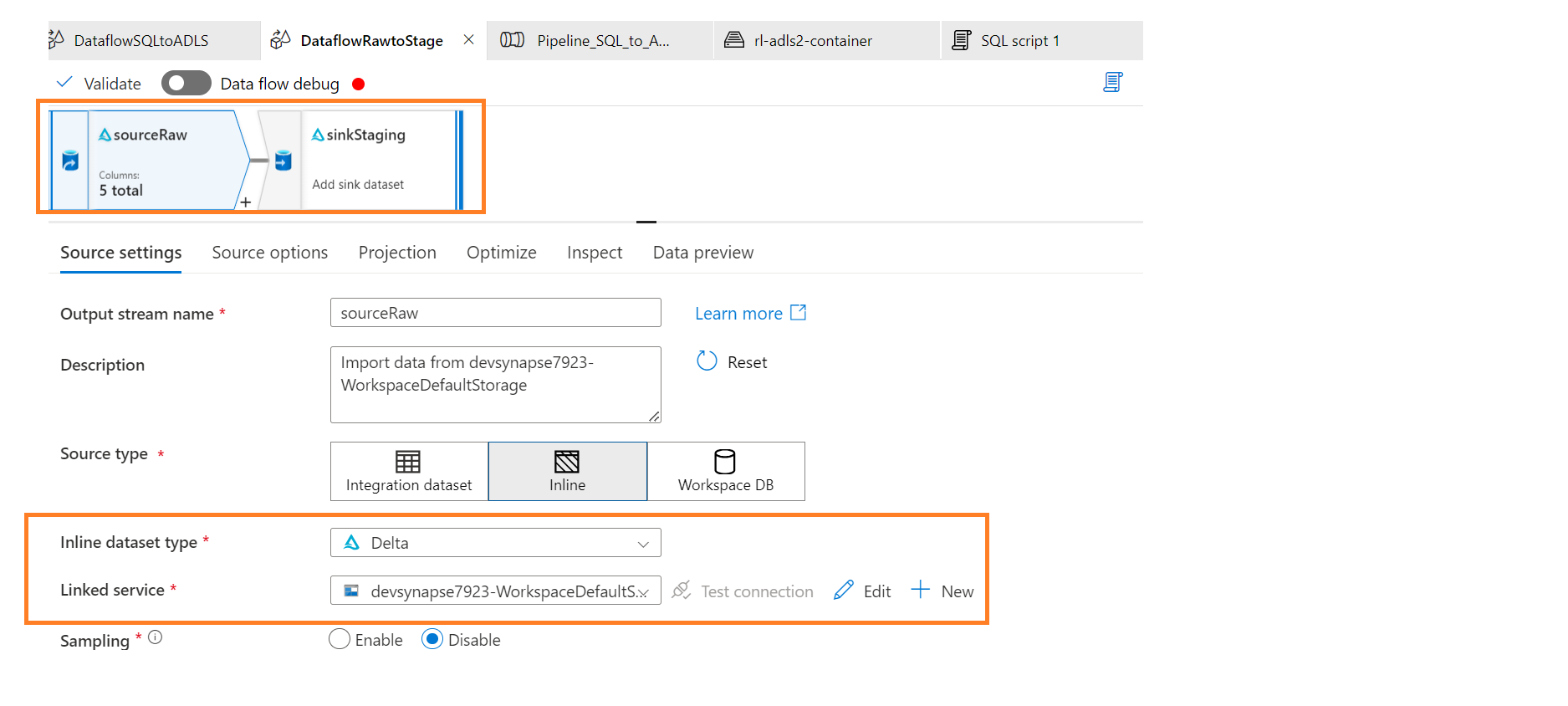


Step 4:

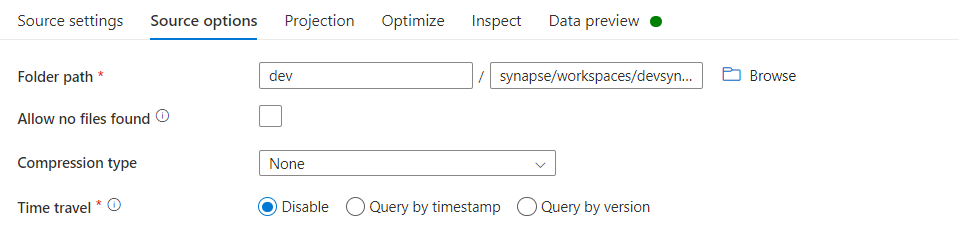
Data flow creation for raw to staging, Staging Update and Cleansing the Data.

**Data load Raw to staging:**

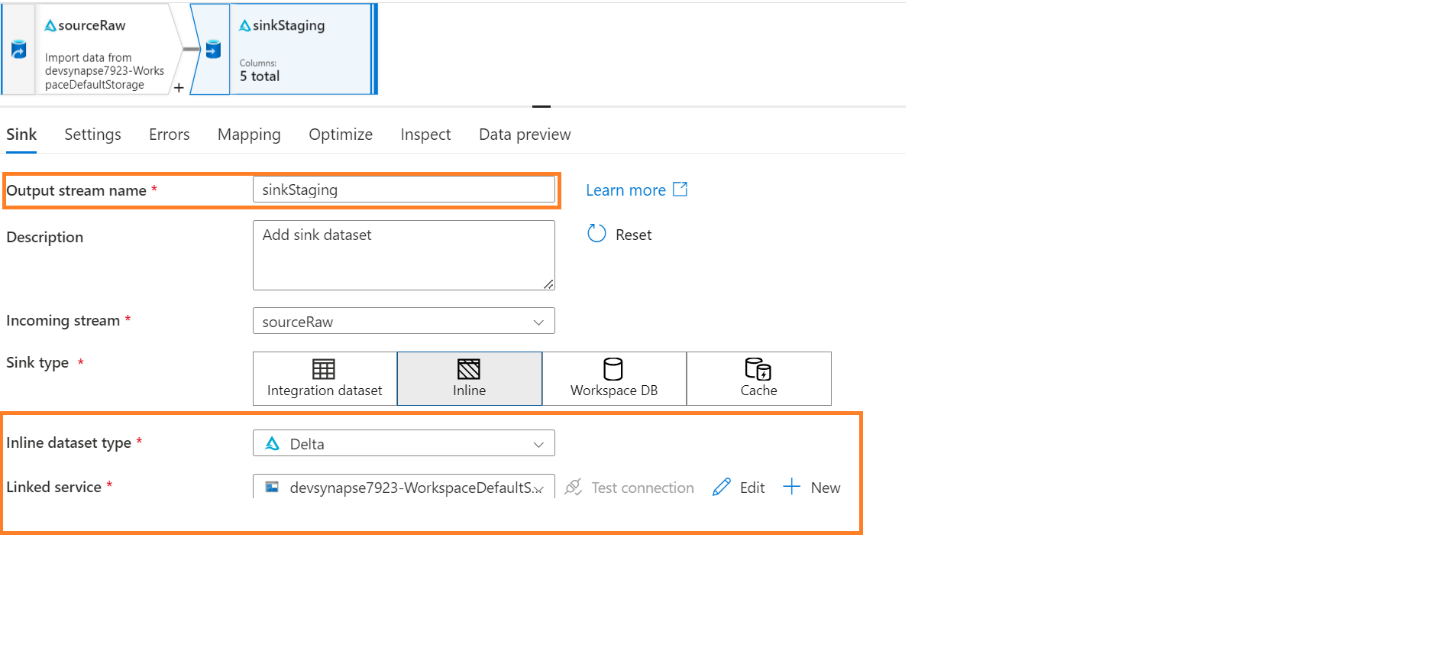
Source configuration:

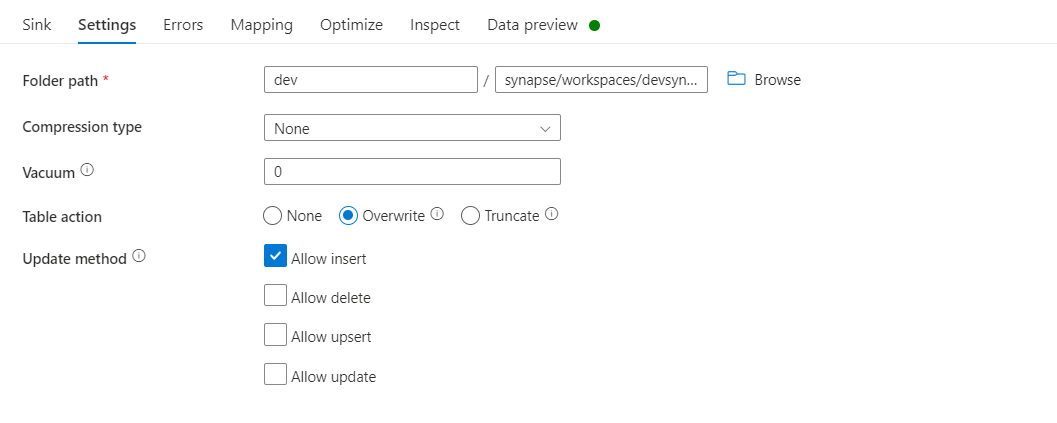


Source path – raw folder:

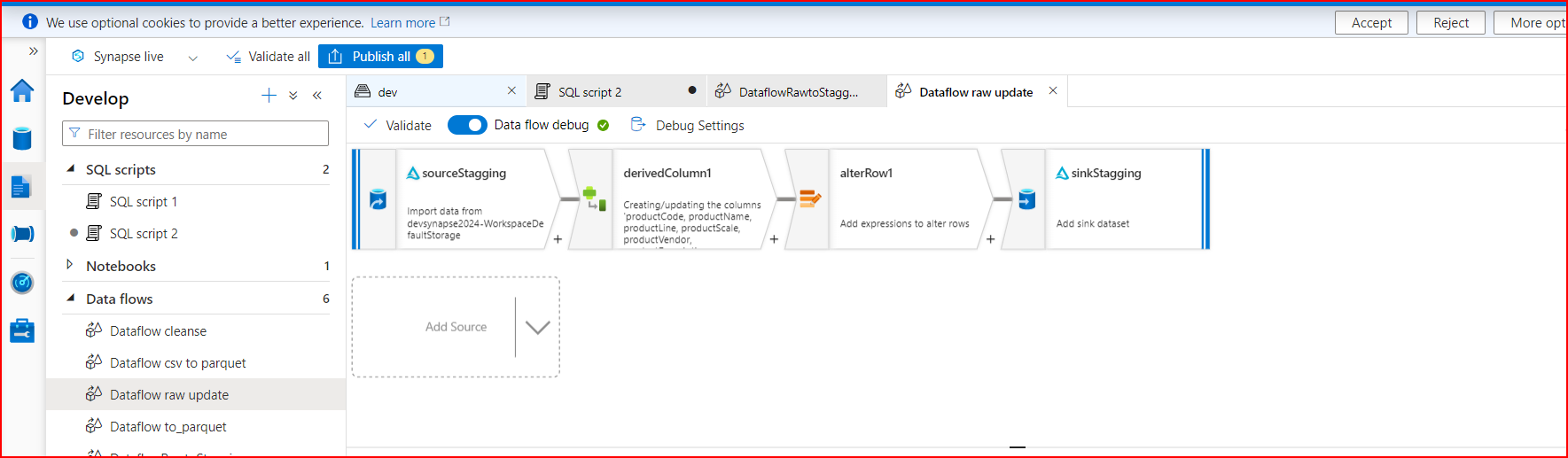


Target Configuration:



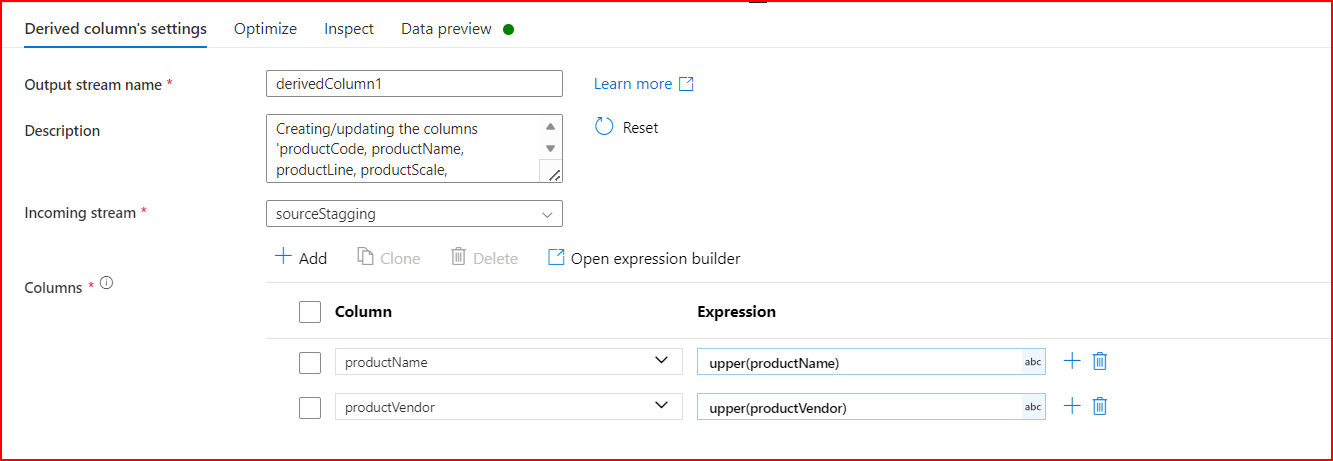


Data flow for Records Update in Staging Folder:



Source and target configurations are as same as we did in previous steps.

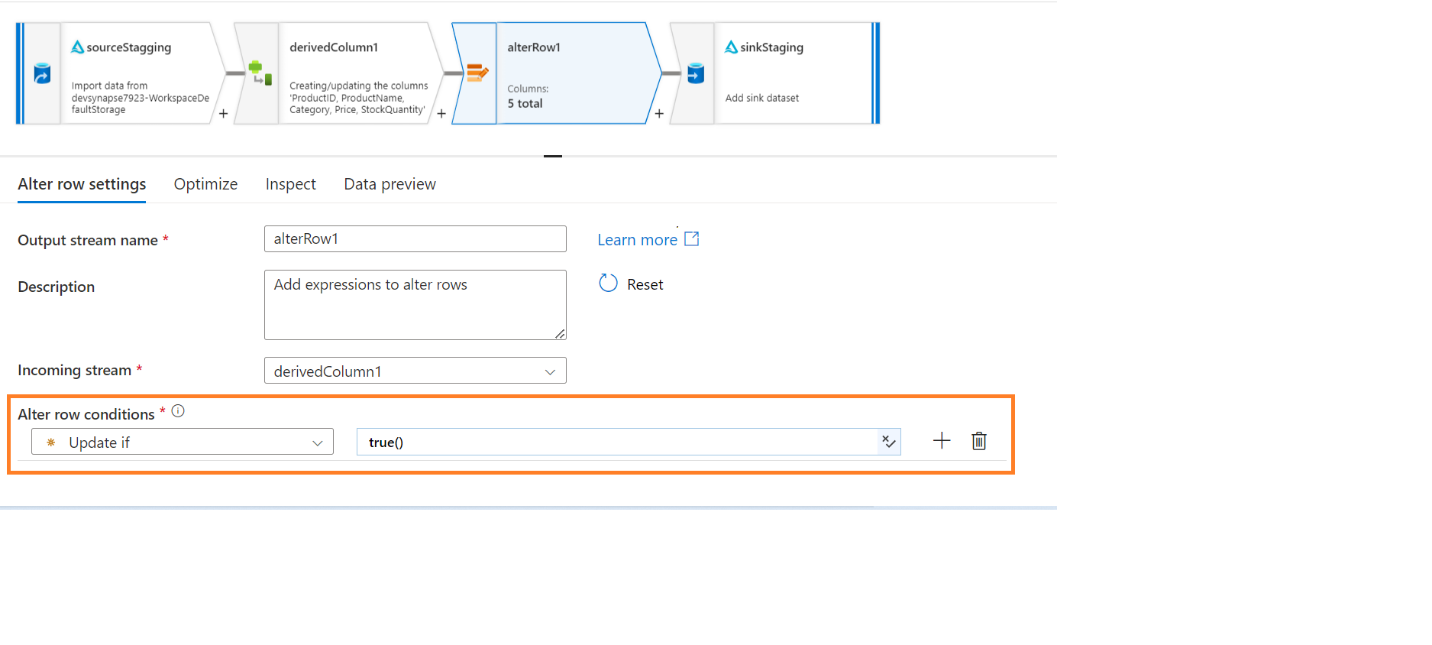
Derived column transformation configuration:



Here we are converting the productName and productVendor as Upper-case letters so that no mis change of letters in the parquet file.

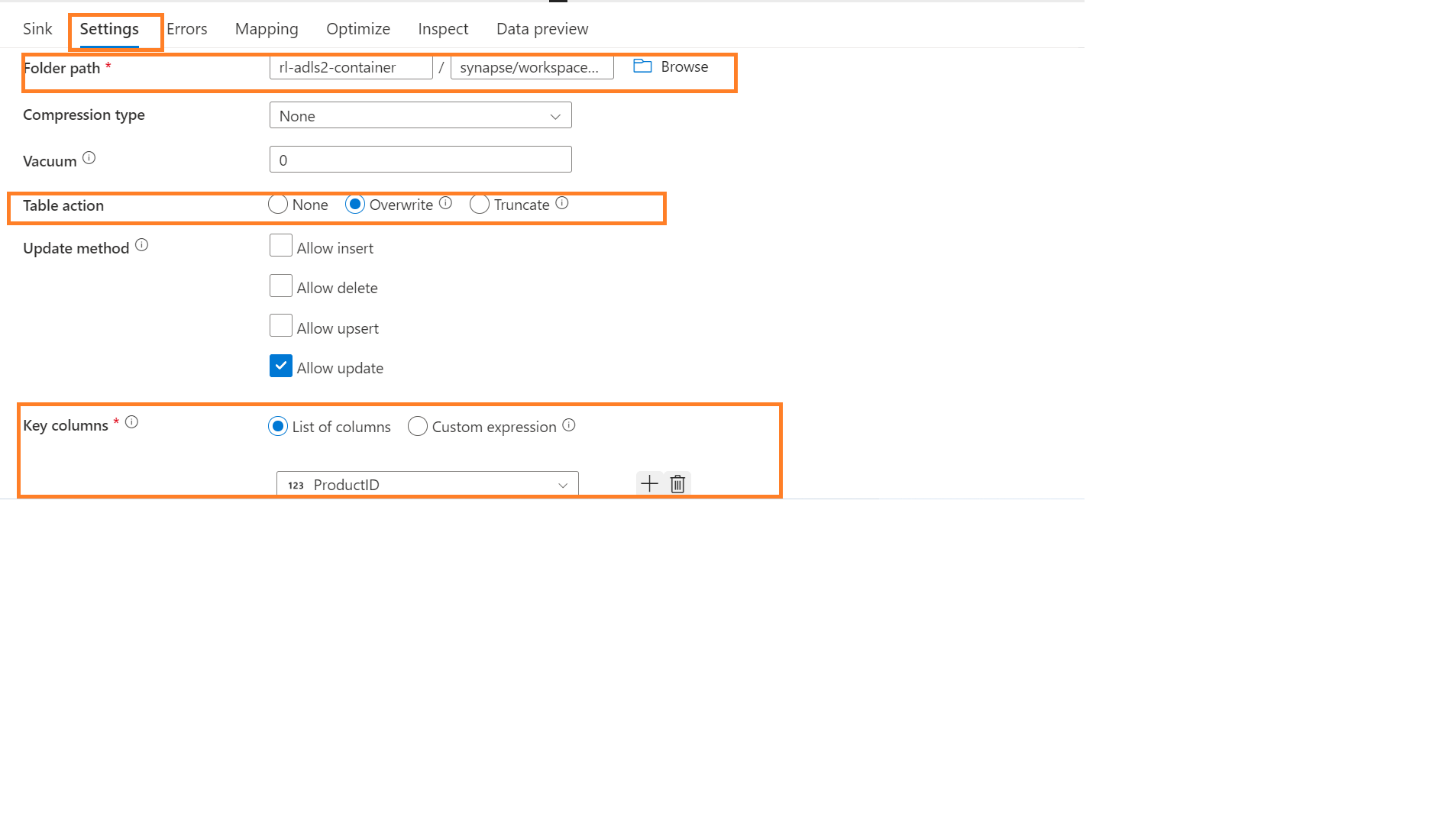
Alter component is to make the data flow to update records if the change or update is done in the rows if condition is True.

So that data will be loaded accordingly in the staging folder.



Sink configuration for Update flow:

All the below options need to enable or checked. So that records will be loaded into staging folder.

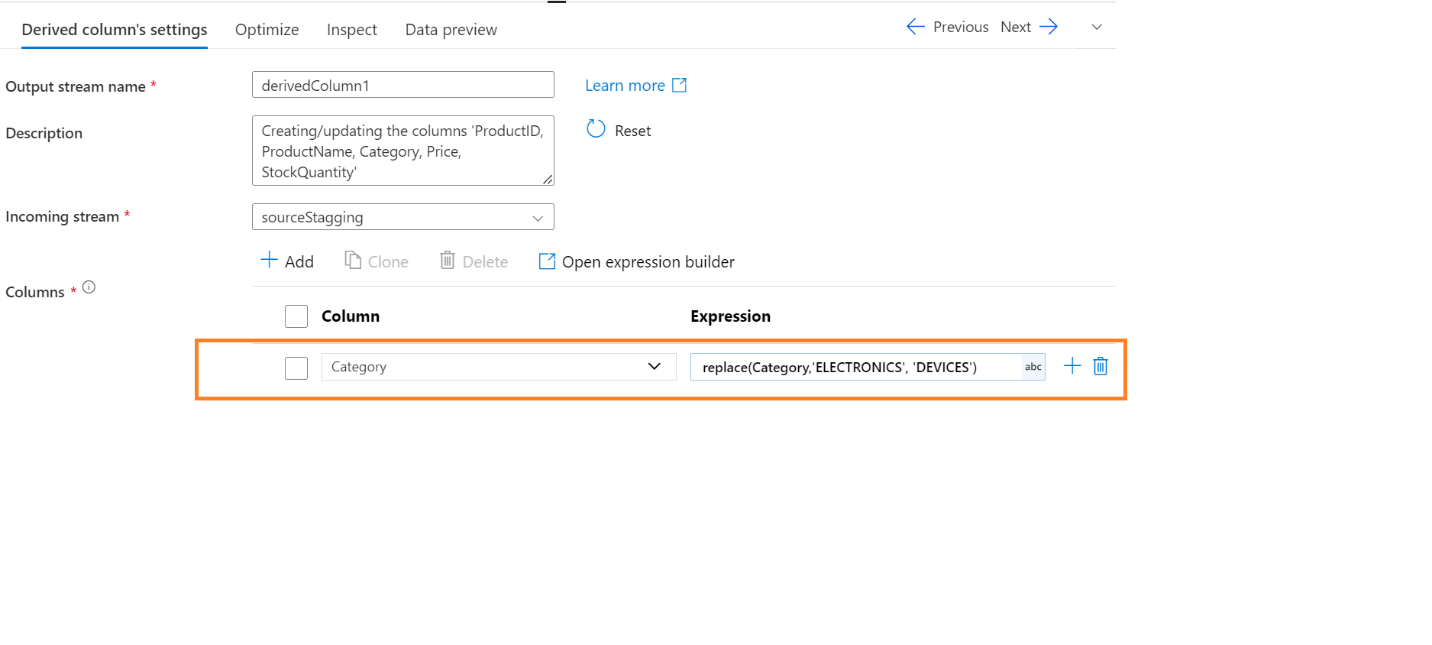


Data flow – Cleansing data and load into cleansed folder:



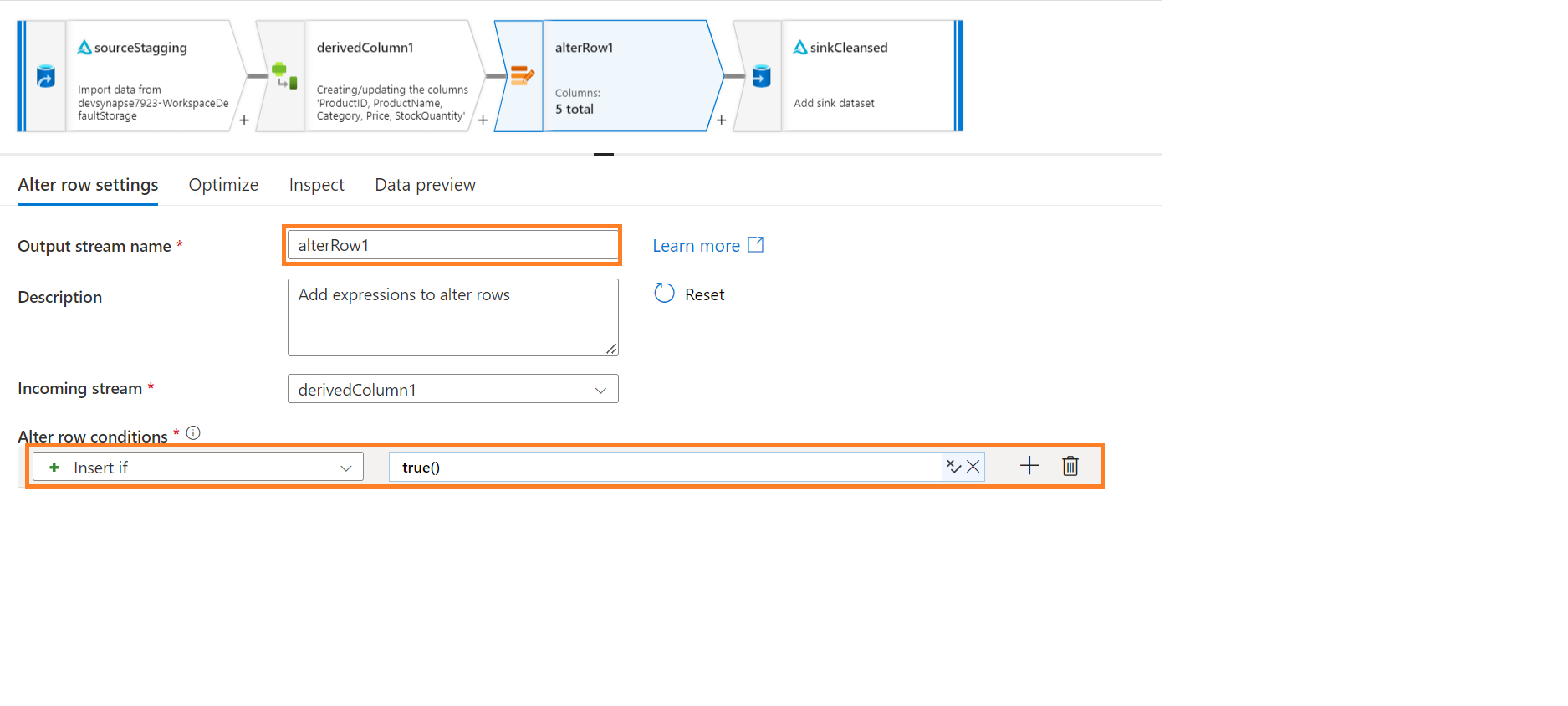
Source and target configurations are as same as we did in previous data flow steps.

Derived column configurations:



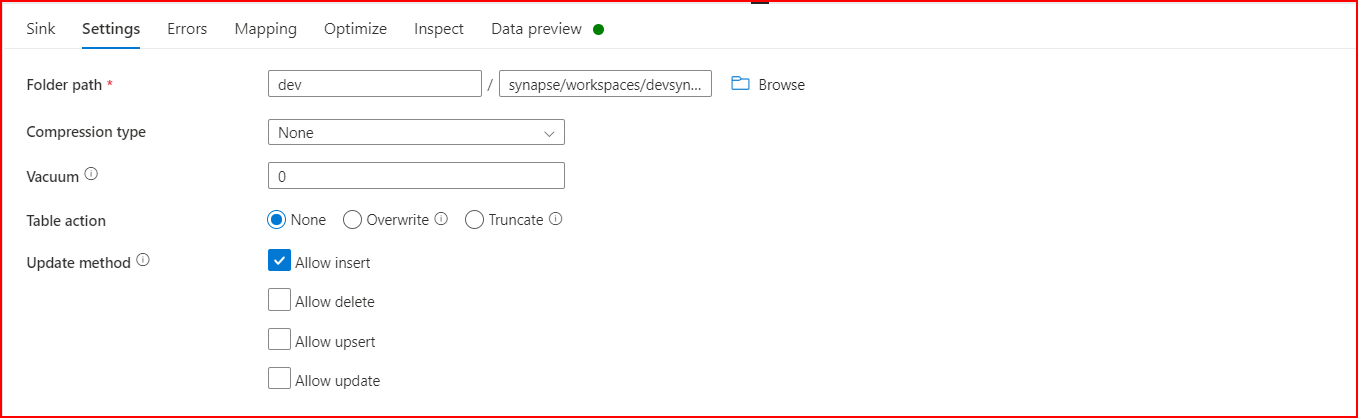
Here in this derived component we are replacing the underscore symbol as hyphen.

Alter component configuration:

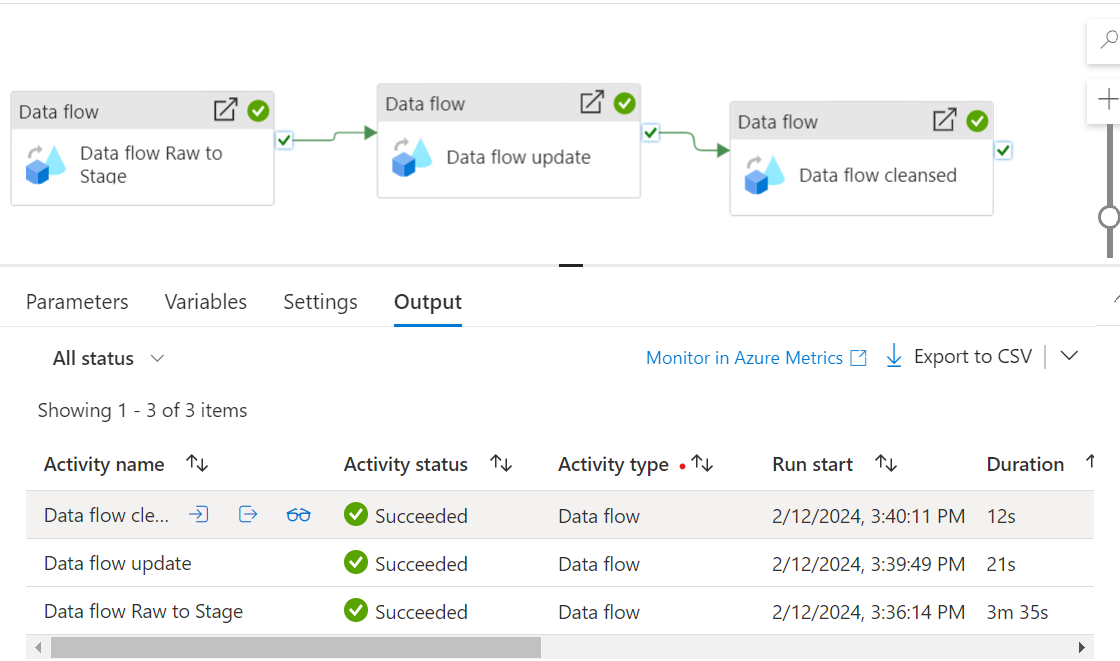


Inserting the data into folder as the true () condition passes.

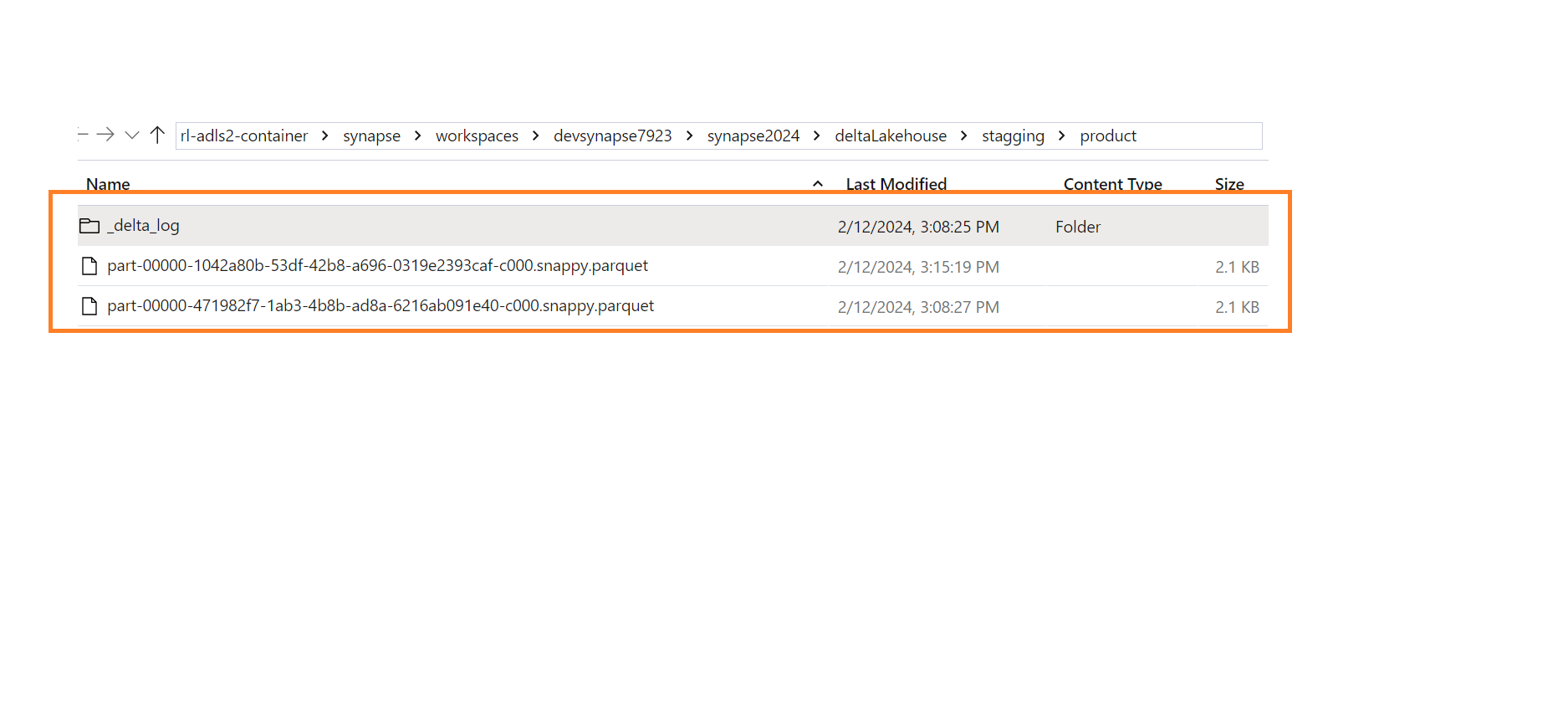
Sink configuration – Cleansed folder:



Pipeline for raw -> staging -> update in staging -> cleansed data

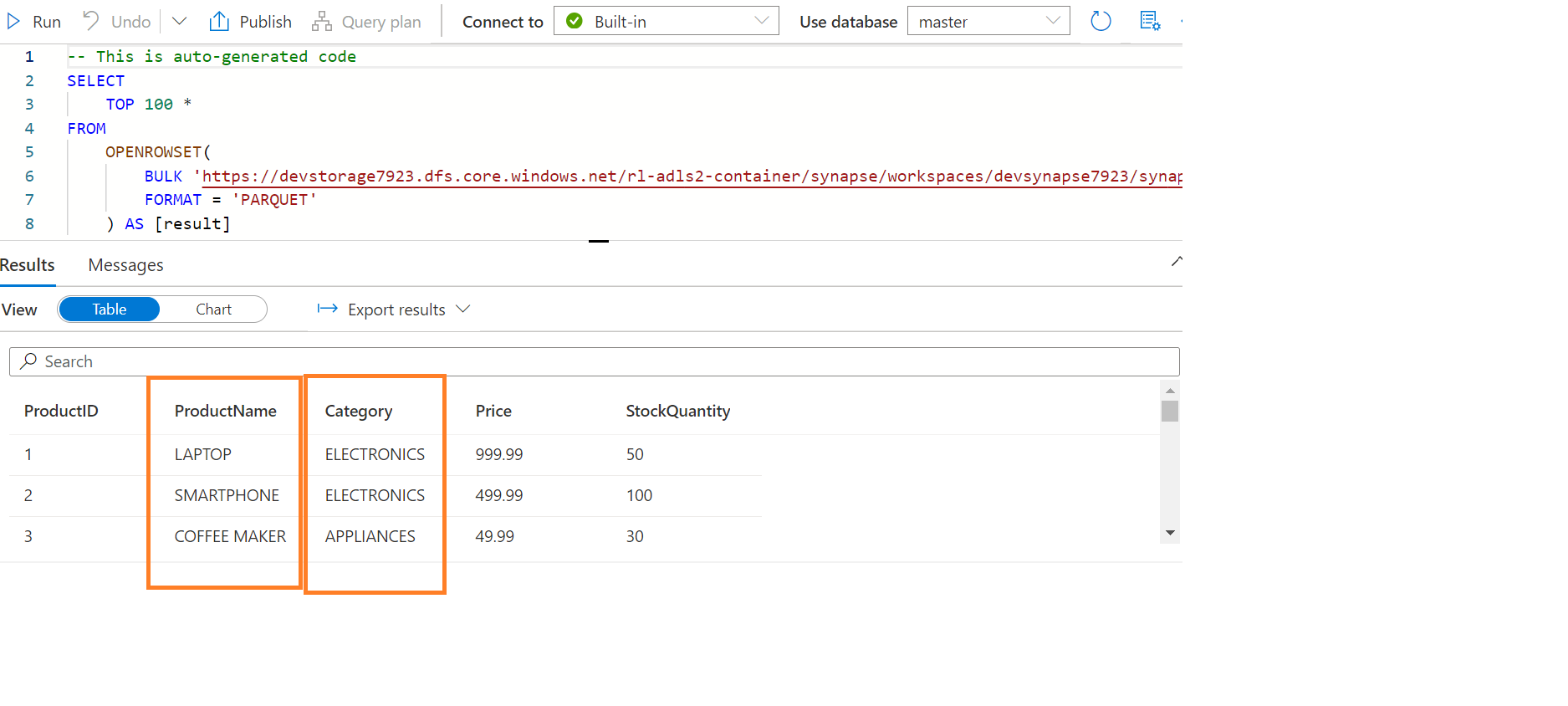


File in Staging:

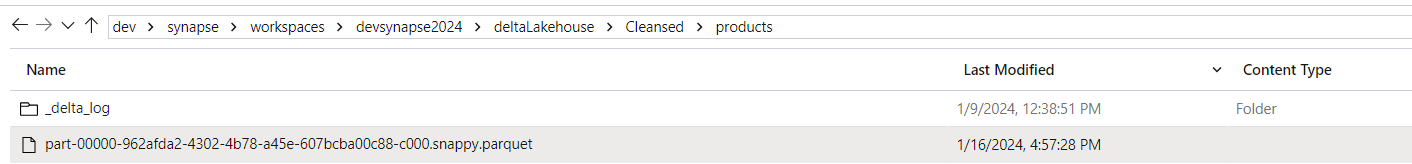


Raw to staging file and Update file

Updated Records:



Cleansed Data:



Records:

