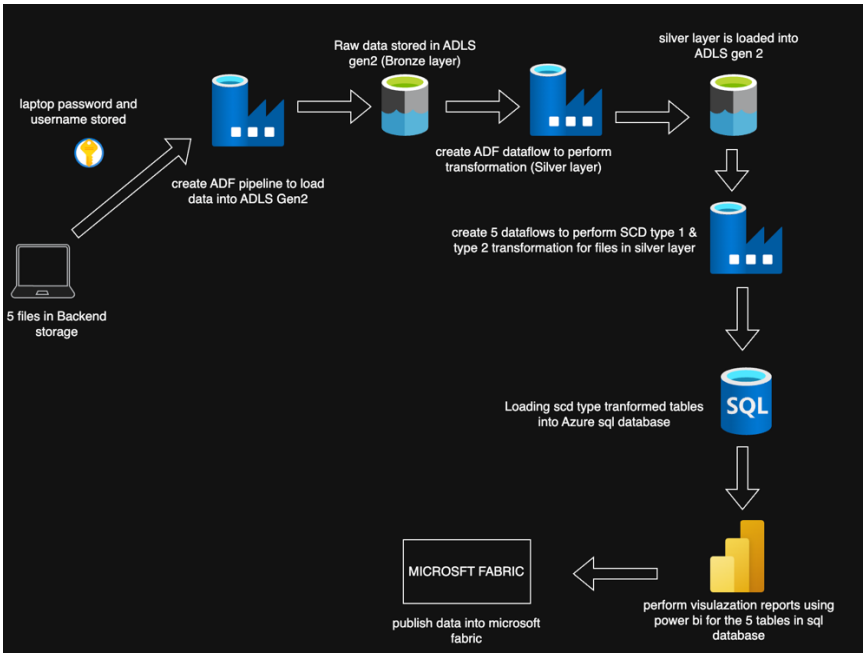
Project Overview:

The goal of this project is to design and implement a robust data pipeline for processing customer account data. The pipeline focuses on the following key components:

1. **Data Ingestion:**
    - Copy data from the backend team's Azure Storage Account to a centralized location.
2. **Data Transformation:**
    - Apply necessary data transformations using Azure Data Factory (ADF) to ensure data quality and consistency.
3. **Data Upsertion:**
    - Upsert (insert or update) the transformed data from files stored in **Azure Data Lake Storage (ADLS)** into a **SQL database table**.
4. **Objective:**
    - Ensure efficient, accurate, and scalable data processing to support downstream analytics and reporting requirements.

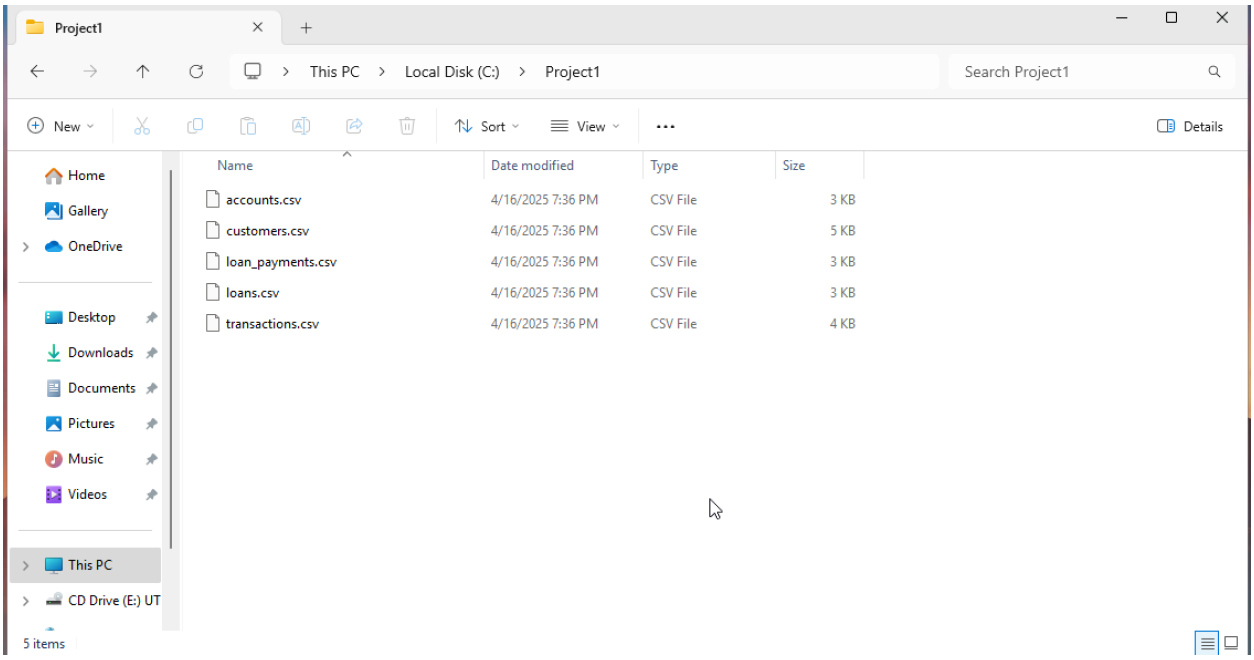


Architecture diagram for the project

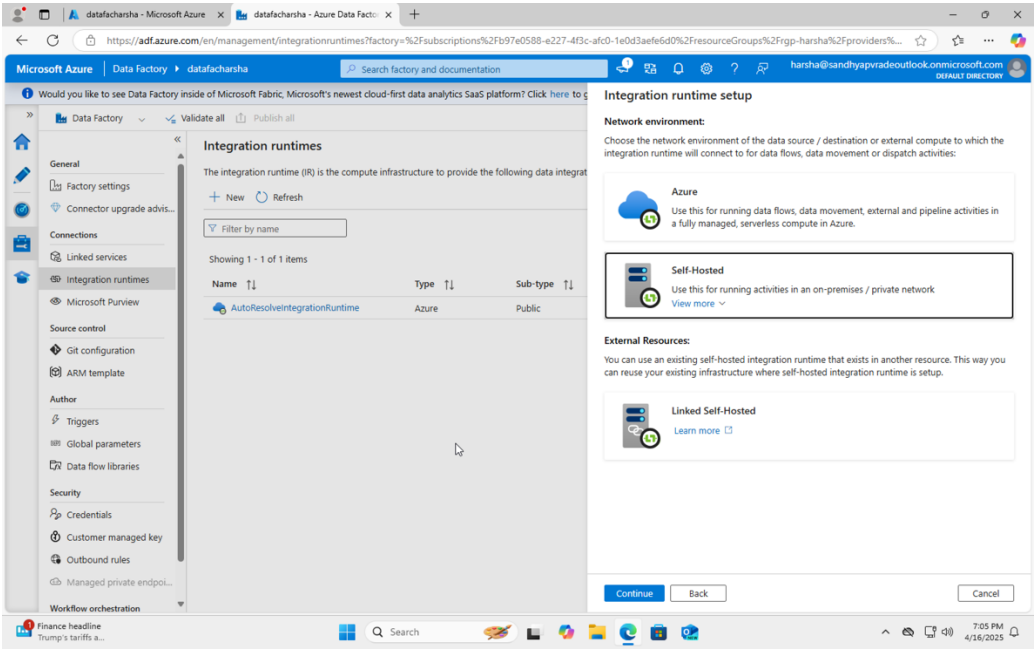
Step_1_description:

Firstly we need data which was copied from the given link “

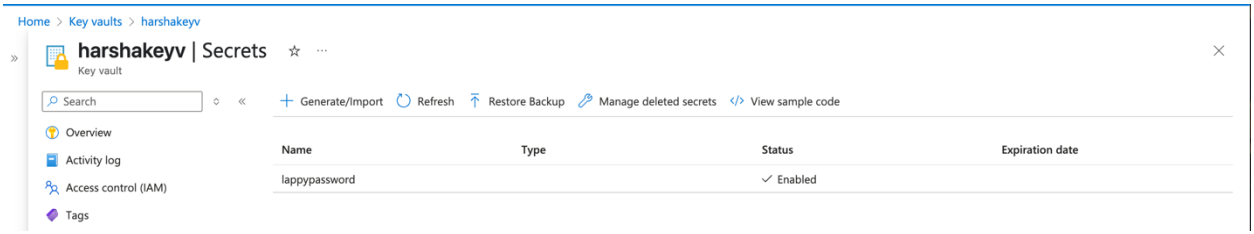
https://www.kaggle.com/datasets/varunkumari/ai-bank-dataset”. Now the files are saved in c drive in folder project 1.



Files downloaded from kaggle

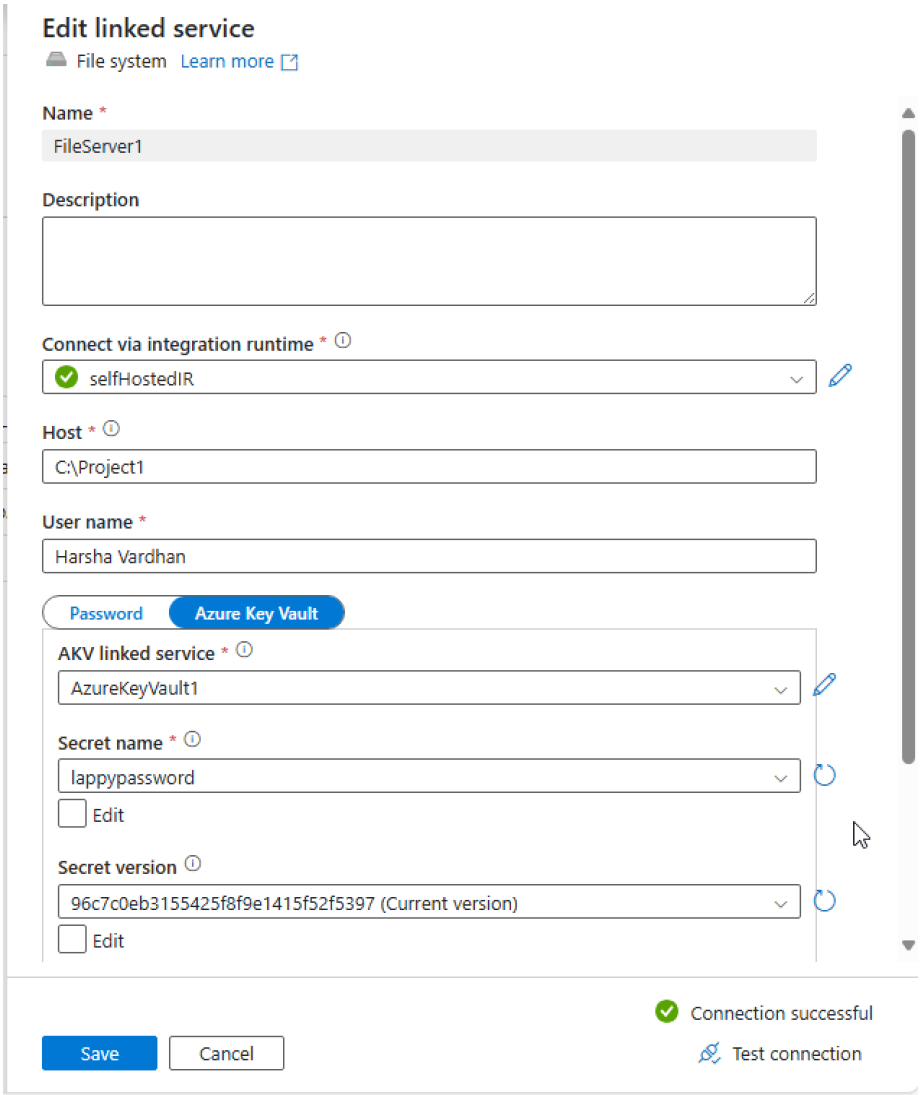


SelfHosted Integration runtime for the local machine

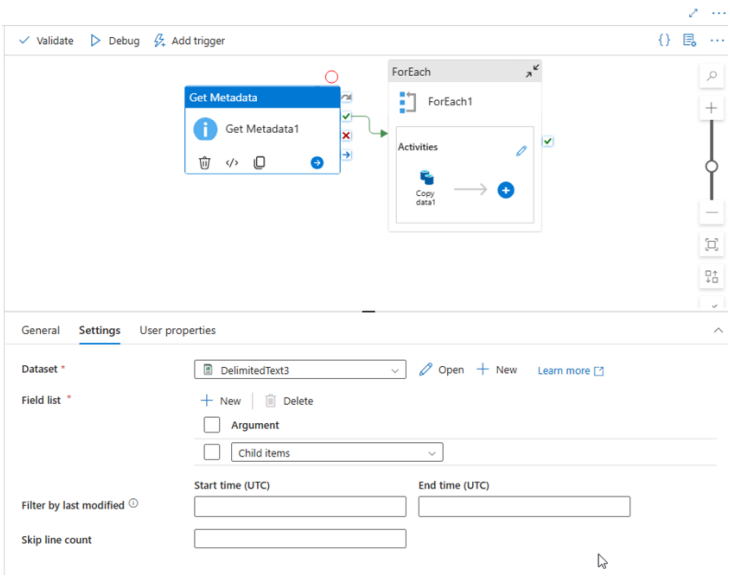


Secret key in keyvault for the laptop password

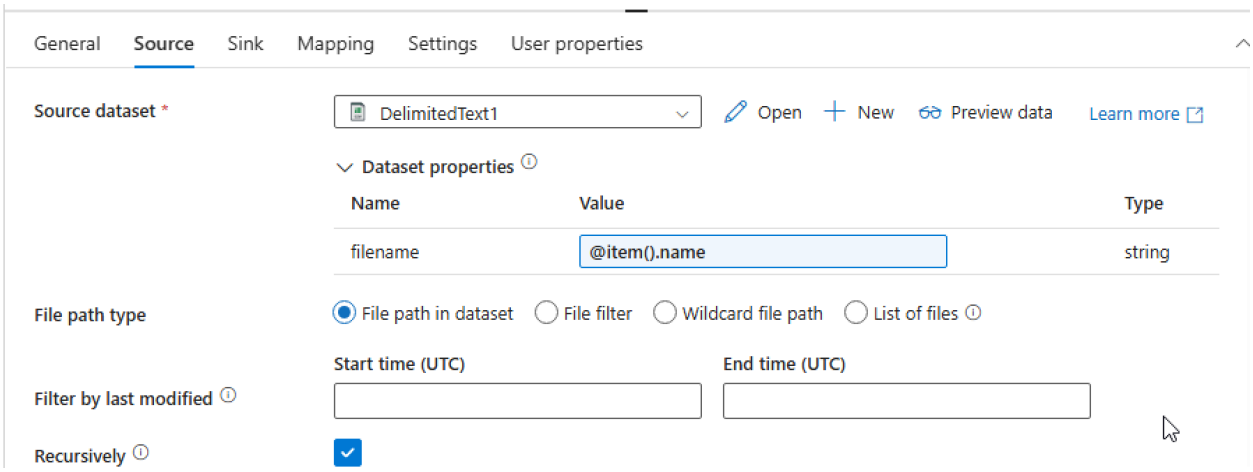
Using self-hosted IR and keyvault we have created a linked service to get the files from laptop to the azure account.



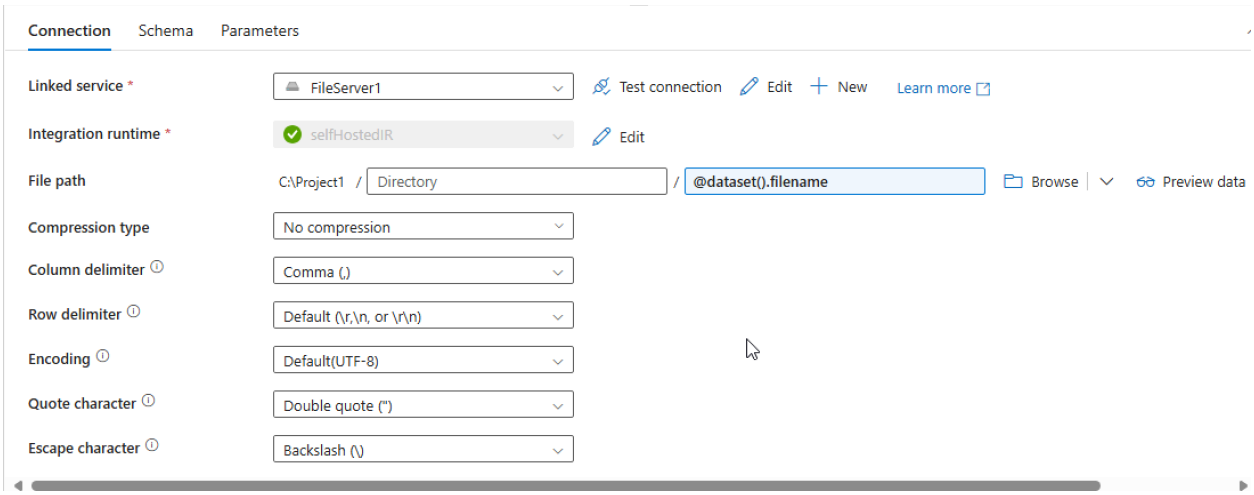
Linked service to copy the backend data



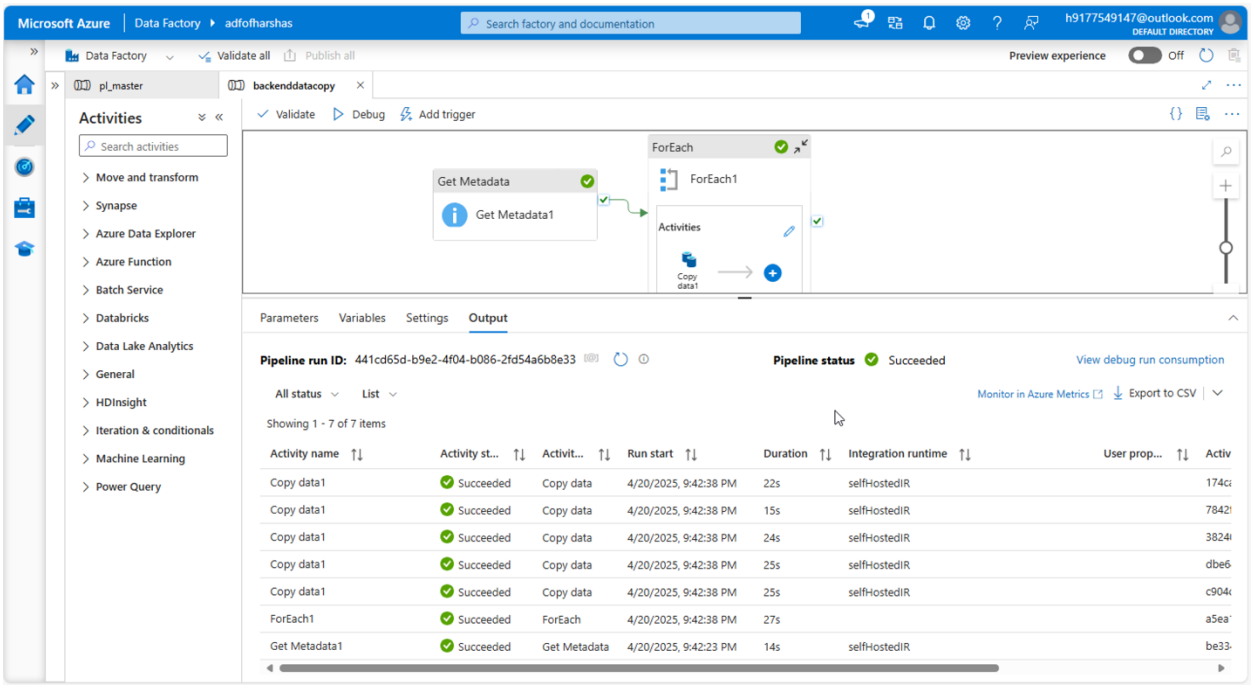
I used metadata to get the data from my folder the child items will be the files in my local folder and will be passed to for each loop



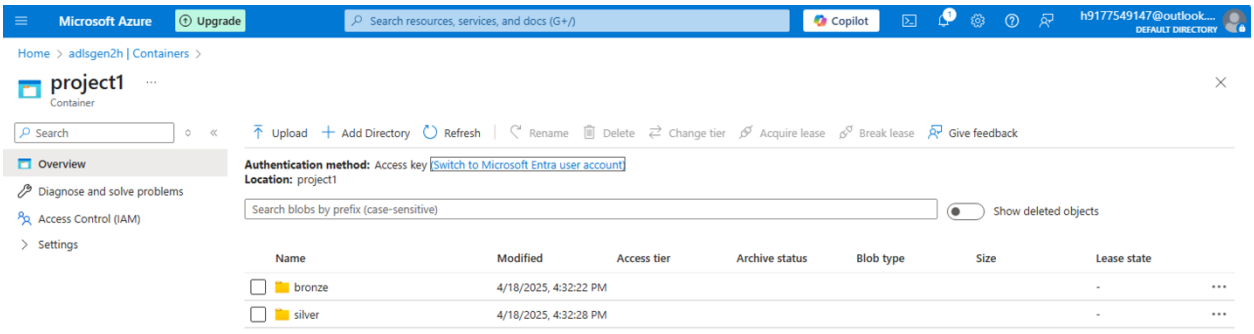
I set a parameter in the copy activity to give the file name. We have given @item().name to get the child items individual file names.



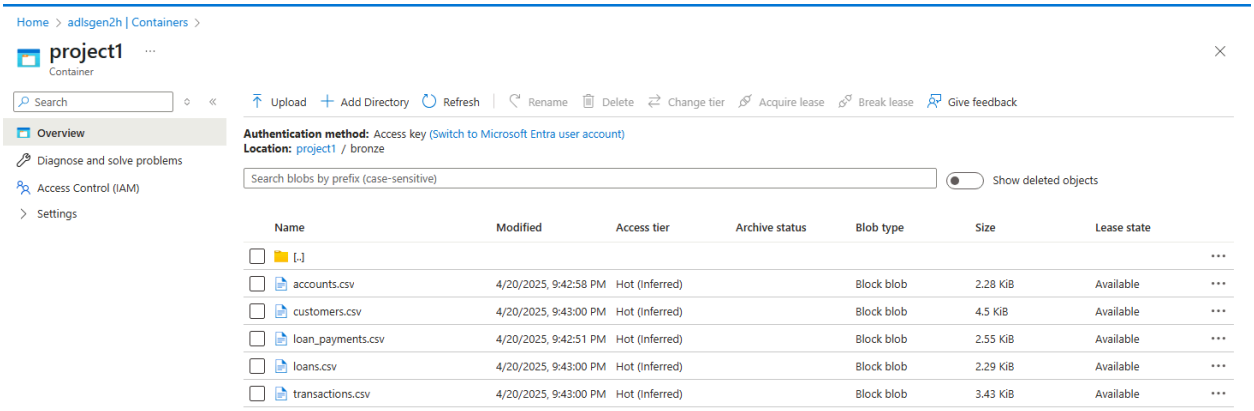
Created parameters for the filename to be copied



Pipeline to get the data from laptop to ADLS gen 2



Containers created inside the ADLS to store the files



These are files loaded by successful run of the pipeline.

Challenges in Step 1:

- The system is not connecting while test connection because of backend permission we must give command to bypass the permissions. The command is
  “Solution:
  There is this 'dmgcmd.exe' at location "C:\Program Files\Microsoft Integration Runtime\5.0\Shared".

  In powershell change directory to this location and run the following command:

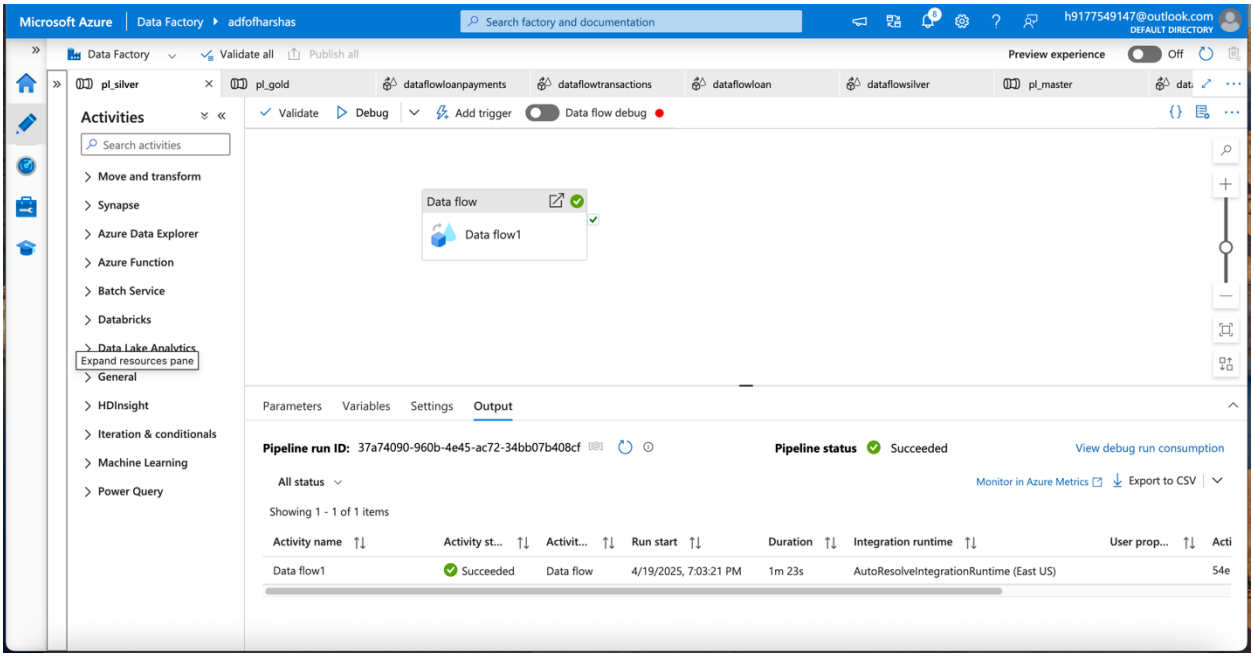
  .\dmgcmd.exe -DisableLocalFolderPathValidation”

- Initially connecting keyvault presented a challenge. It is failing to connect for the linked service. For this case we must go to the access policy and give us GET to read the secrets in keyvault.

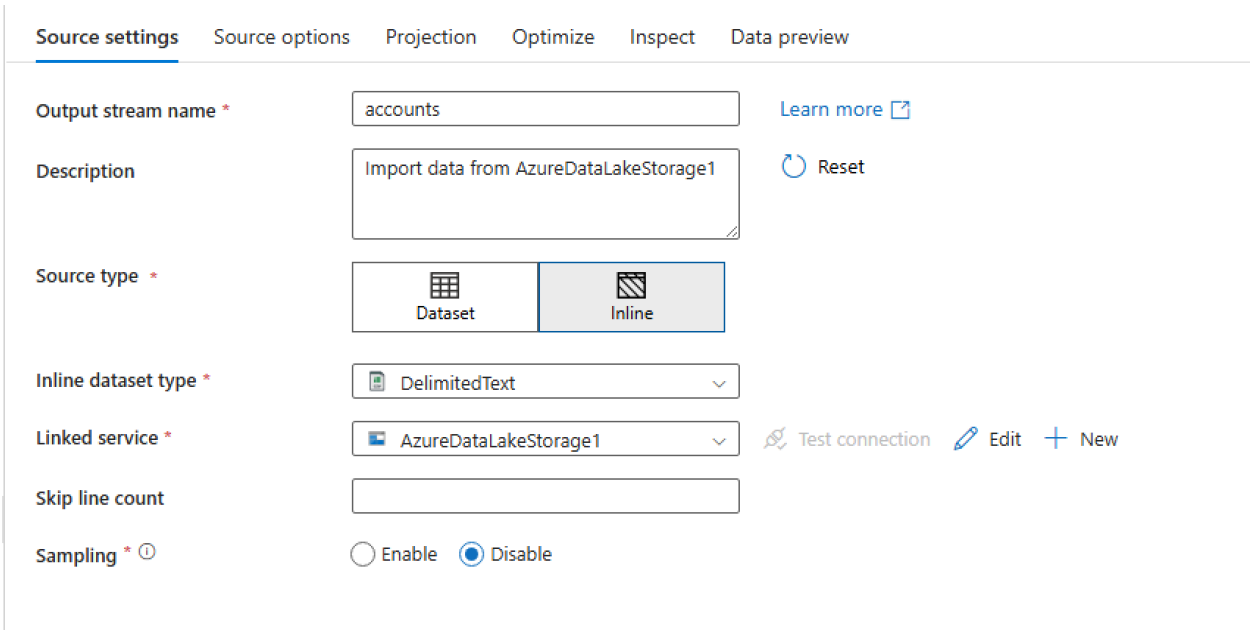
Step_2_Description:

In step 2 we must use the data in the bronze layer, create a dataflow and remove null values duplicates etc. for further analysis.

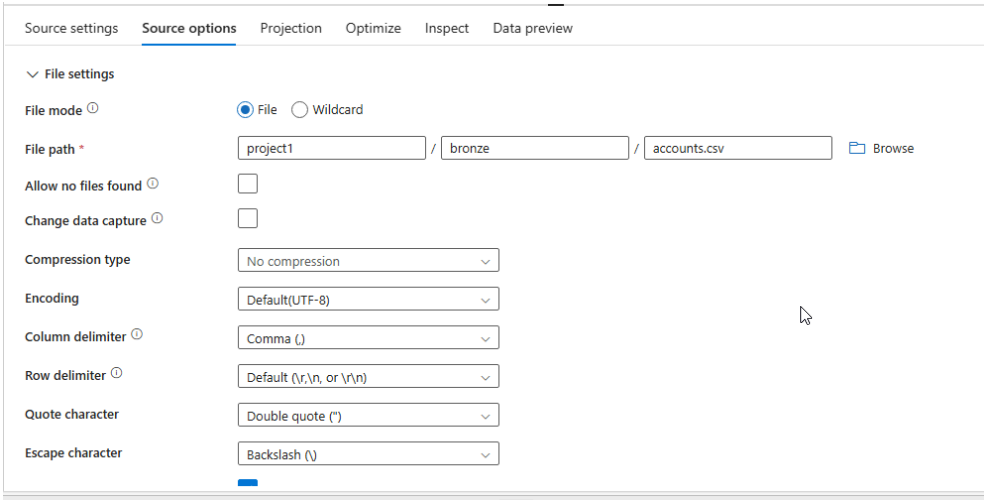
Here is the dataflow design for step_2



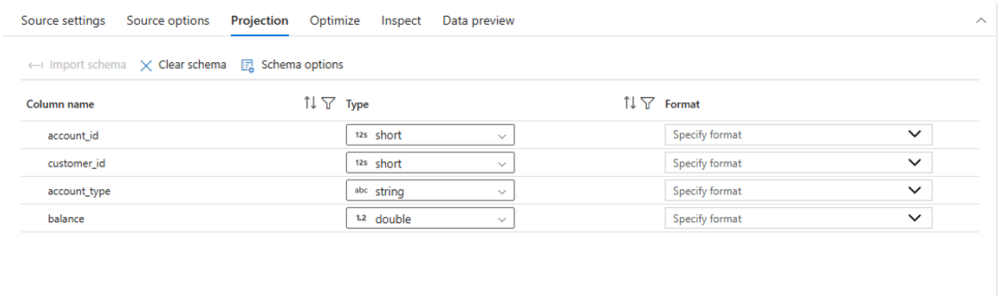
The pipeline run for step 2



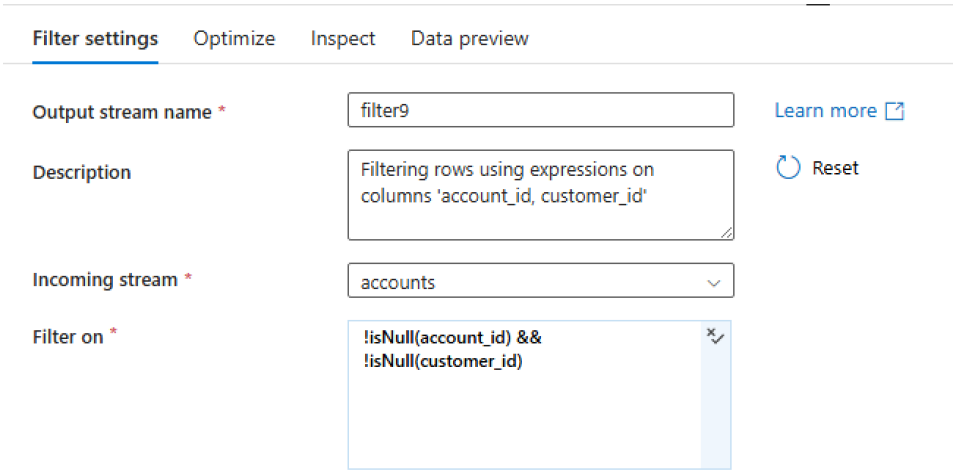
We are giving the source from our ADLS gen 2 from bronze container



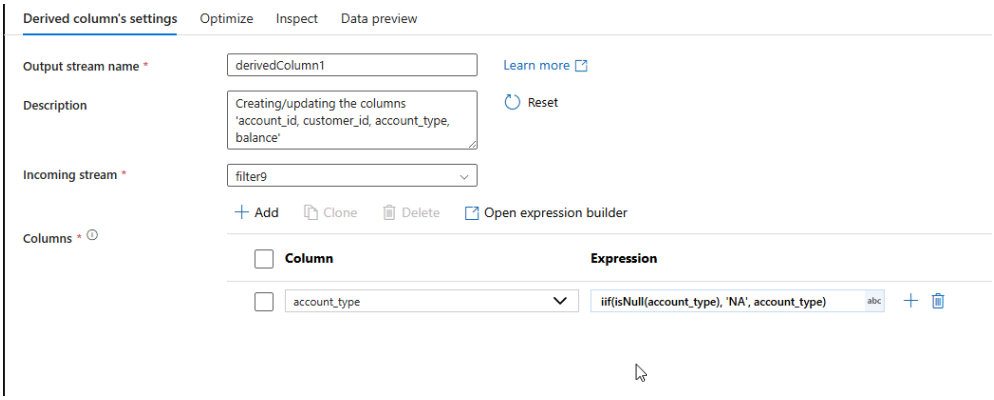
Selecting the accounts.csv file from bronze layer.



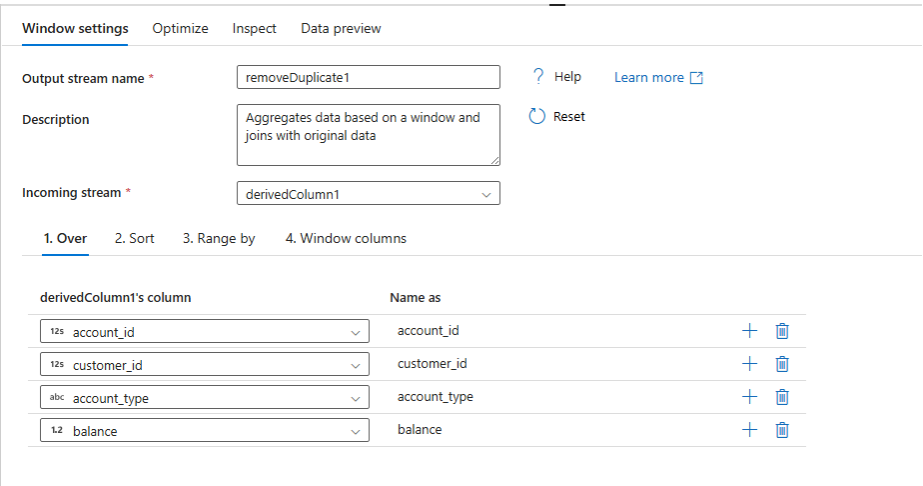
Then import schema for the file



Now use filter option to check main field like account_id & customer_id if they are null we remove the entire row

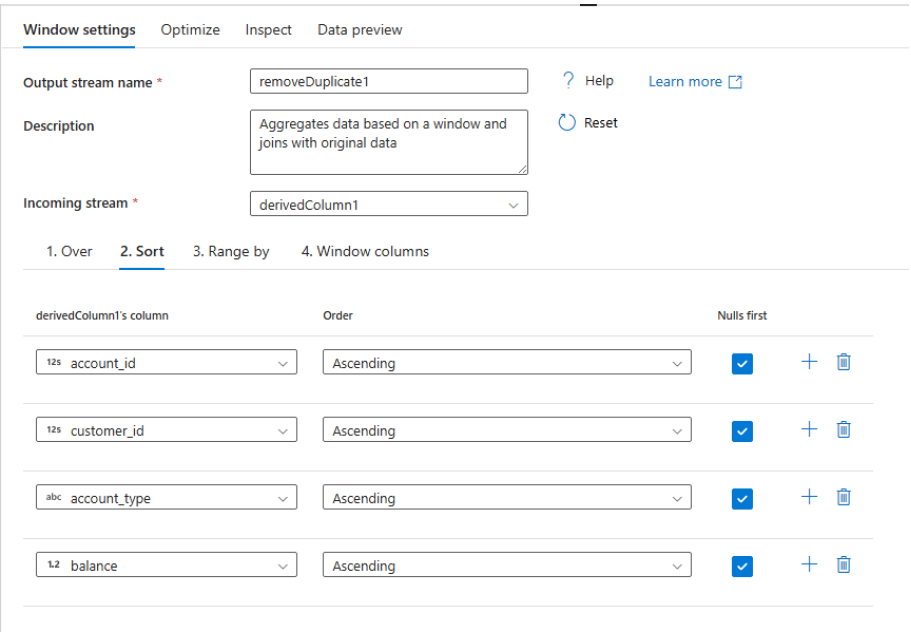


Then we give some values to the missing values in the row we are using derived column function for that with expression “iif(isNull(account_type), 'NA', account_type)”.

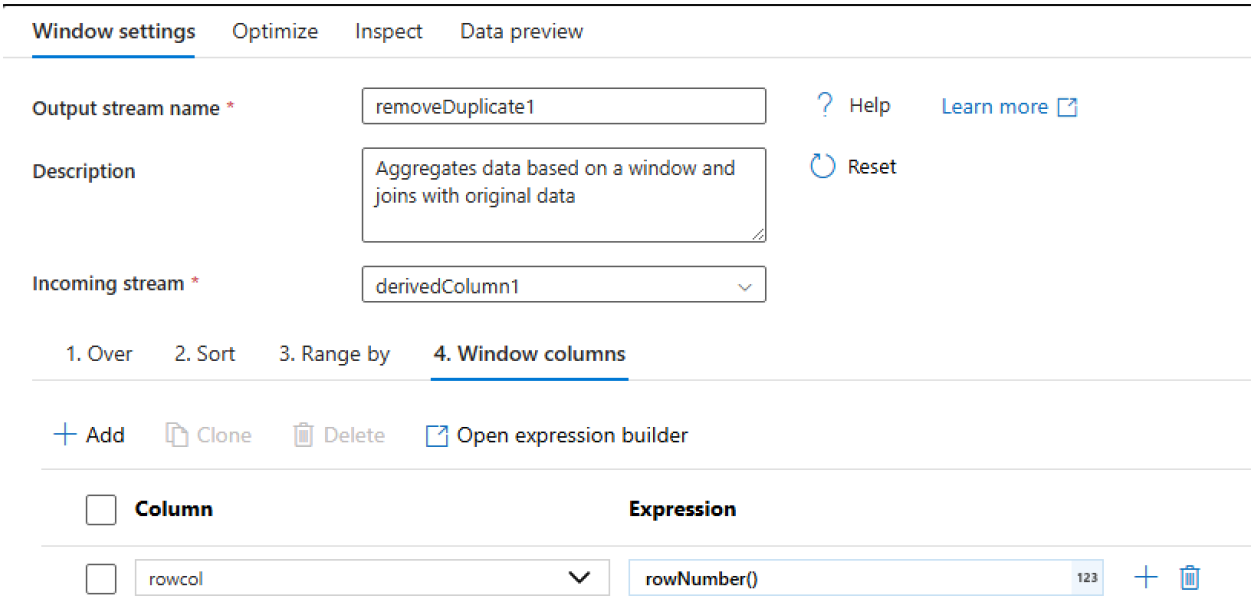


For removing duplicate values, I am using window function. In window there are 3 main field to fill

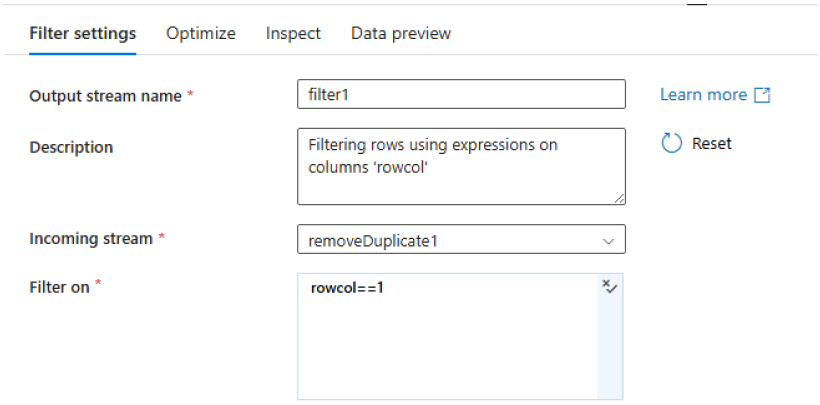
- Over: over is used as order by in SQL. We give every column of the table here.
- Sort: we also give every column of the table here as well to sort.
- Windows column: we give the expression here. I am using the expression rownumber () to count all repeated values here.



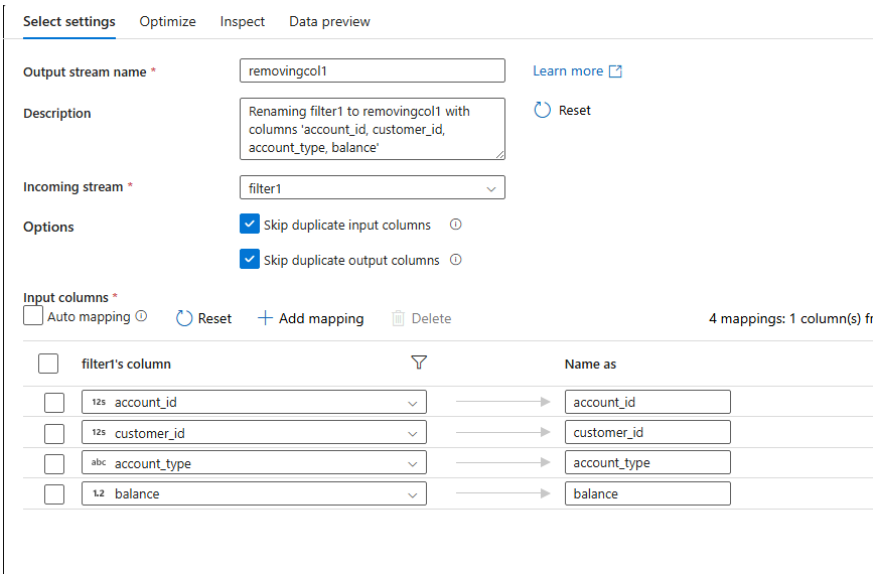
Sort in window function



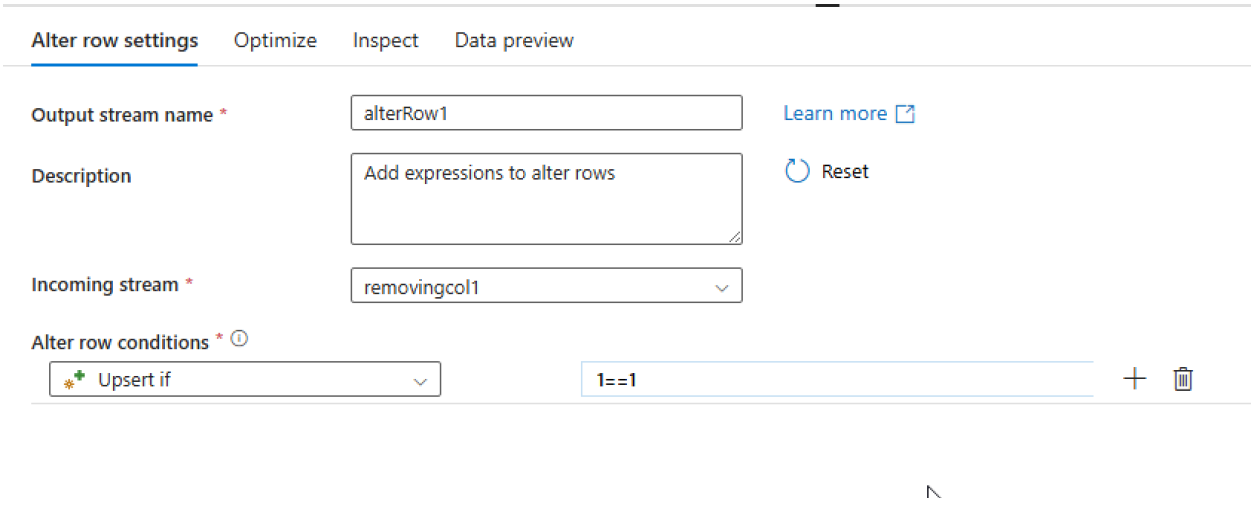
Window column expression of window function



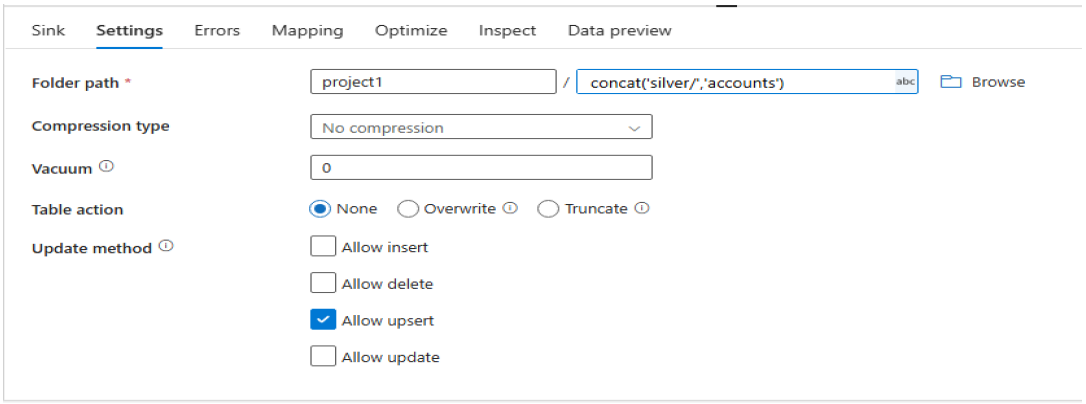
Now to remove the duplicate values we use filtering function if select rowcol==1 and only let the fields that pass-through filter.



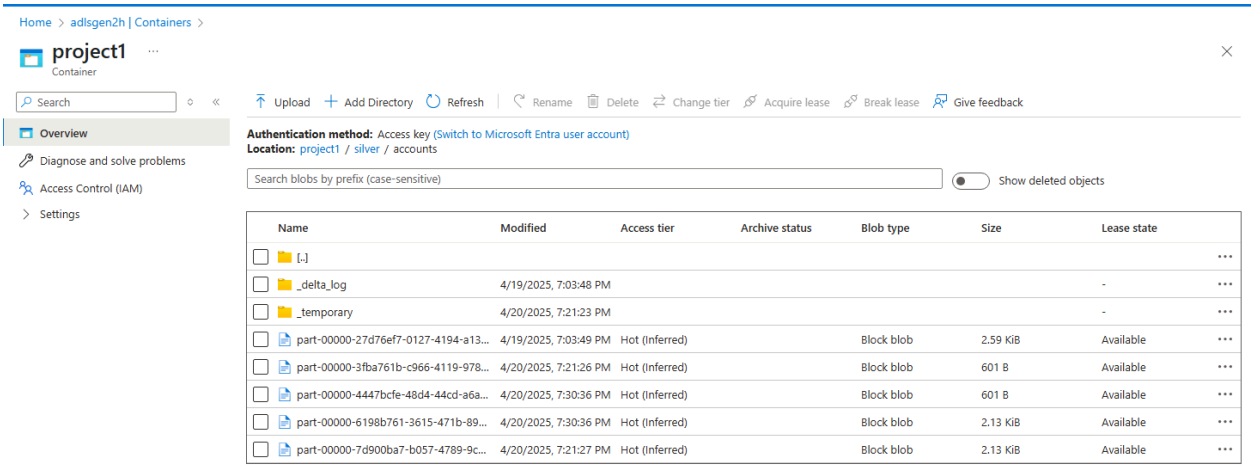
We use select function and remove rowcol and let only original rows of the table to pass



We give alter row function and give condition upsert to create delta format file.



We give the path to set the file to be saved.



Here is the saved file

The delta format files created inside each separate file with their names.

The design goes like this :

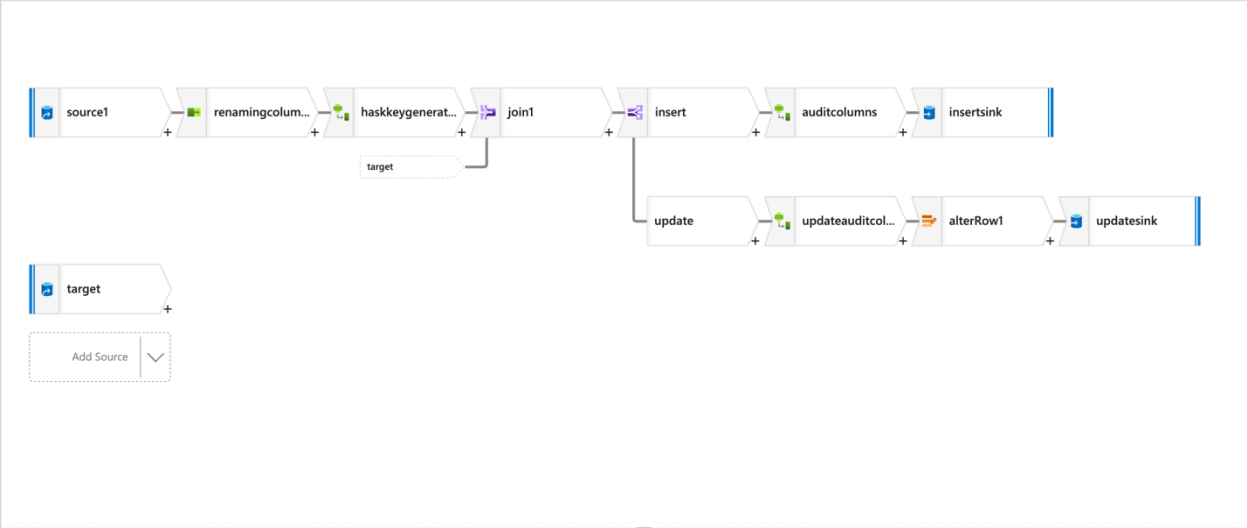
- Removing rows if primary key field are null
- Derived column for placing missing values
- Window function to check number of duplicate values and assign row count
- Filtering row count>1
- Removing row count column
- Checking necessary condition to create delta file format in alter row Upsert.
- Sink: Placing the file into destination as a delta format file.

Challenges in Step_2:

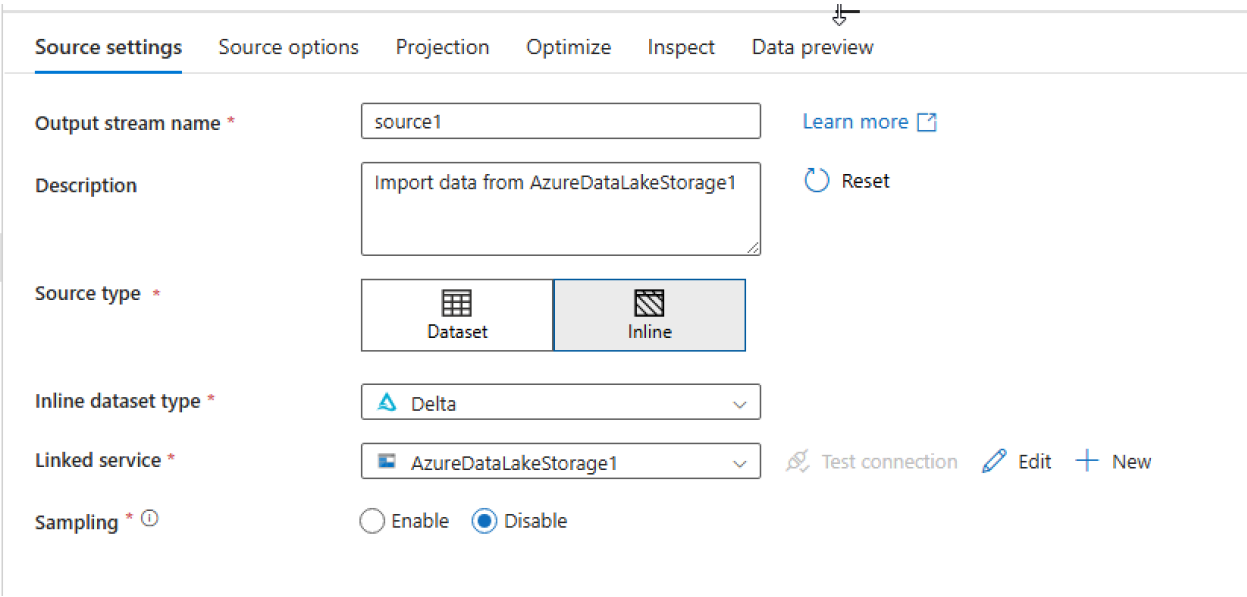
- Firstly we needed to figure out with data inside the files is more important. If they are null removed the entire columns. For ex: In accounts.csv file if account_id is null we have remove the full row. But we should not remove if amount field is null simply put it as “0”.
- Mapping issue: after finishing the pipeline the mapping is changing for some reason and have to check each time before publishing it.

Step_3_Desciprtion:

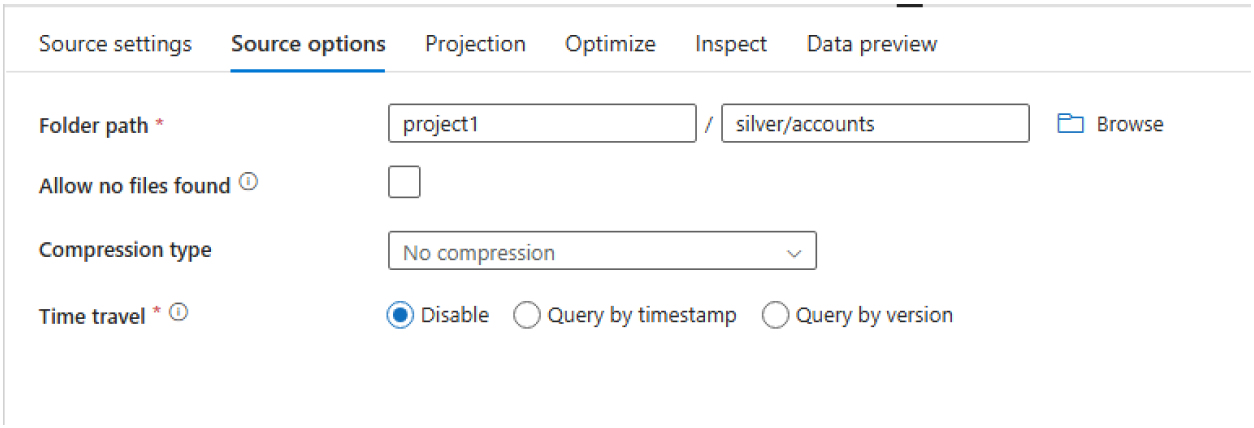
Now we have to take the delta format files as source and perform SCD type 1 transformation and SCD type 2 transformation and place them in Azure sql database.



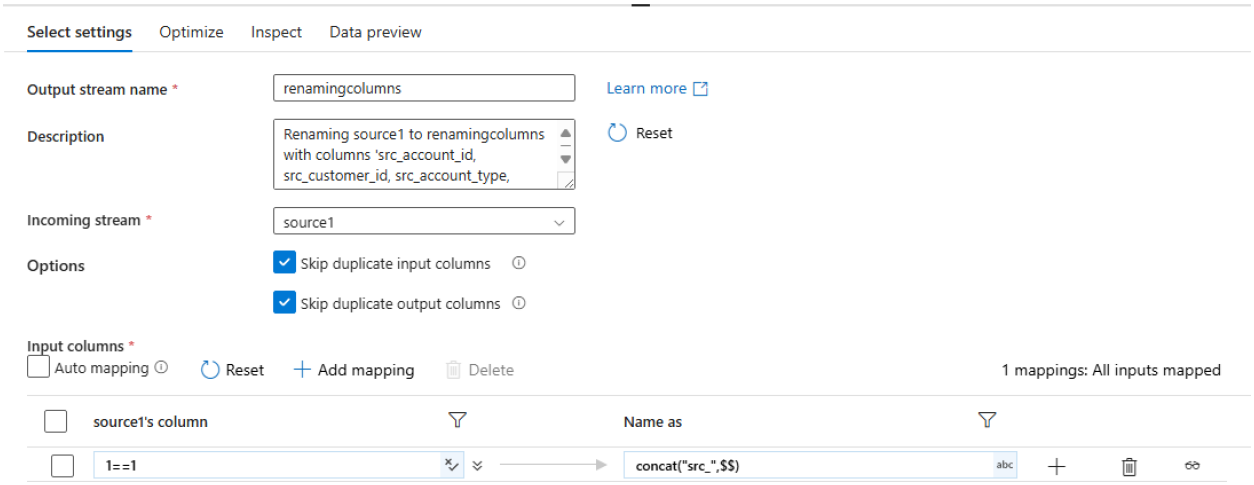
SCD type 1 transformation



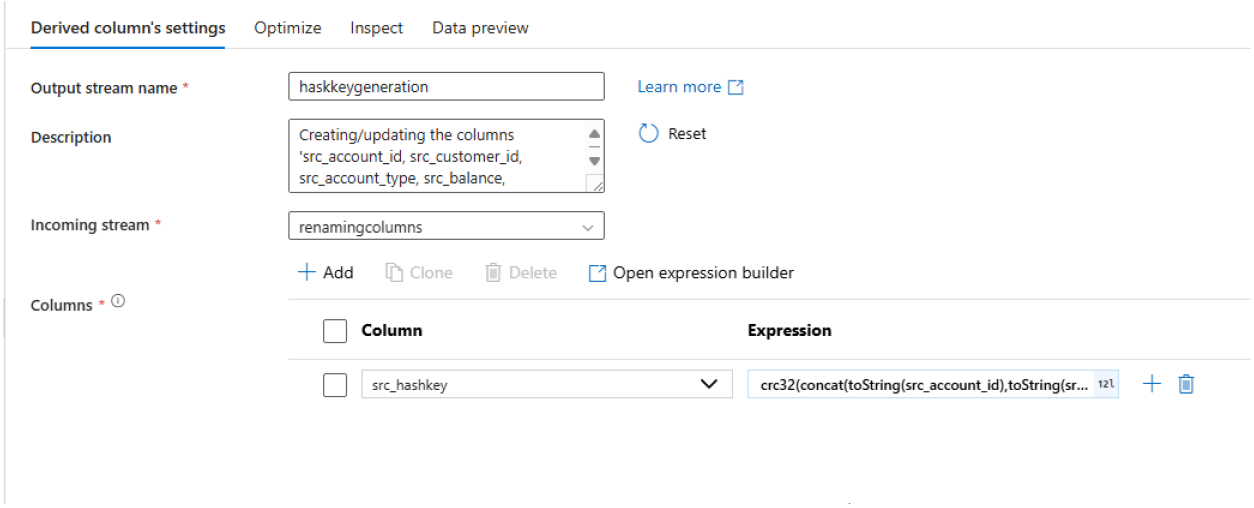
The source details are here we are selecting the delta file format from ADLS gen 2



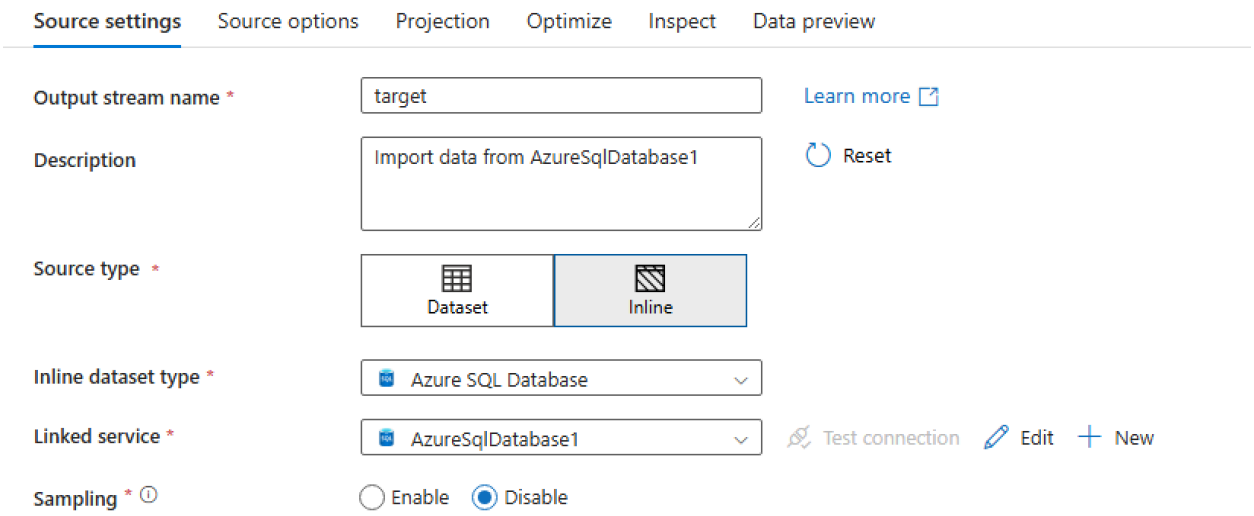
Setting the folder path



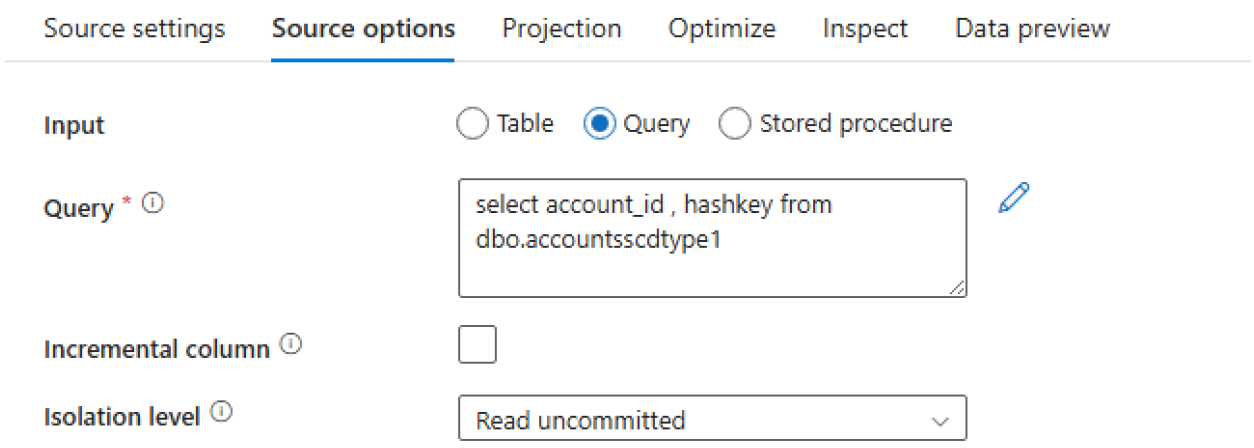
Renaming the source columns to differentiate them from the table. We are giving the function concat(“src_”,$$). $$ gets the columns of the table.



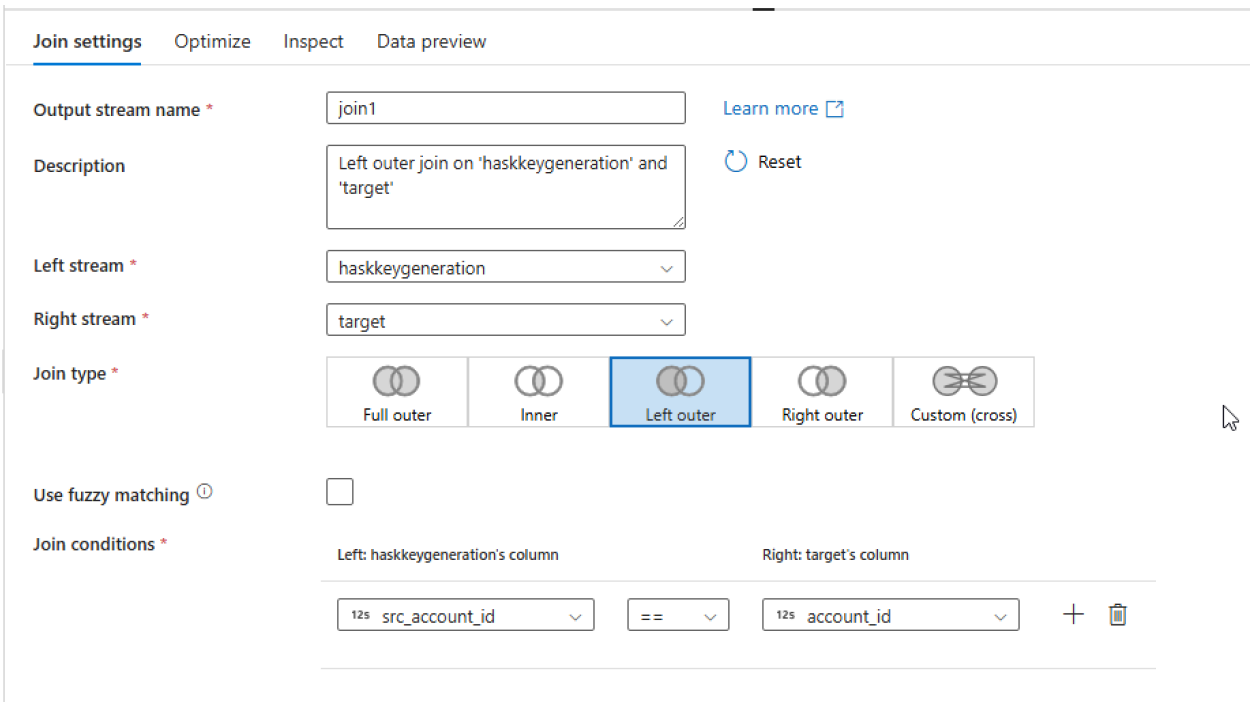
Here we are giving crc32 has function by giving all the columns and converting non-string datatypes to string.



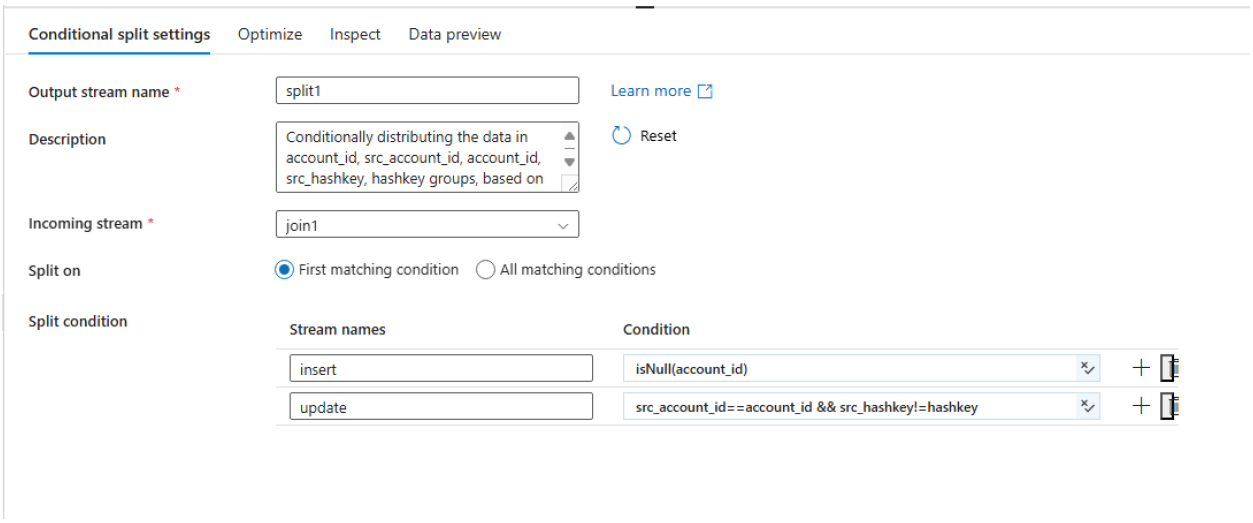
Here is the target file from the table already created in sql databse.



We give this query to select the account_id & haskey from the table.



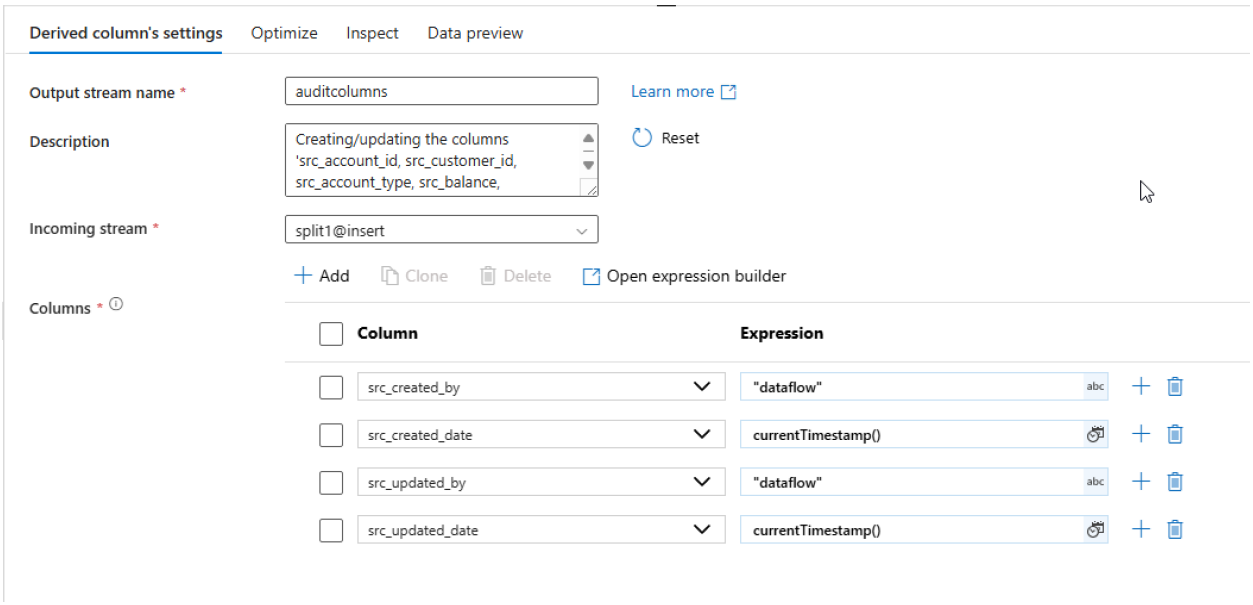
We are using left join to join the source and target



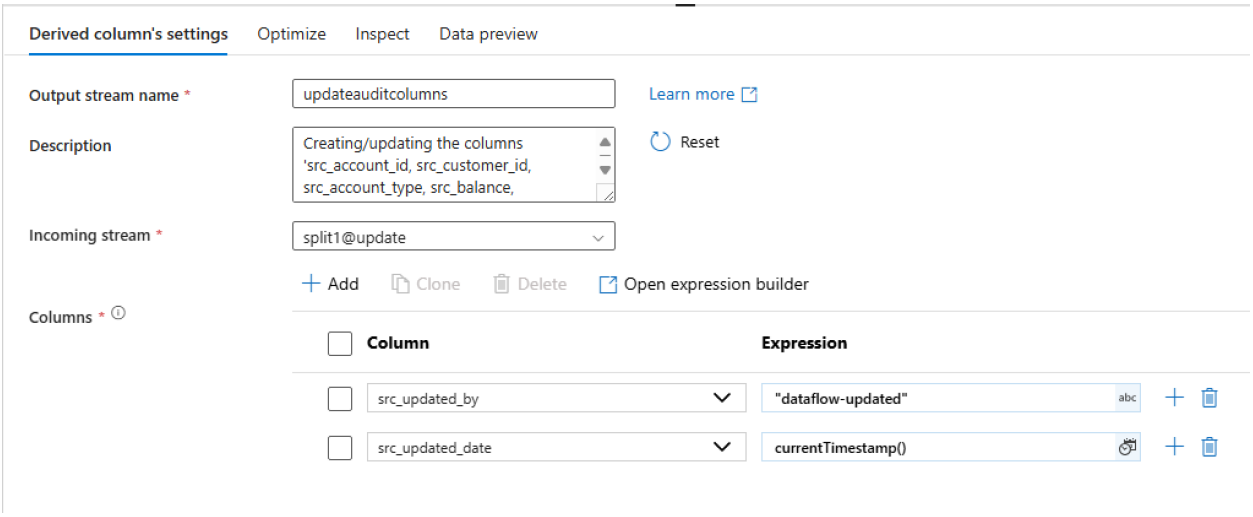
We are giving the conditions to set the insert and update functions

For insert we are giving : isnull(account_id)

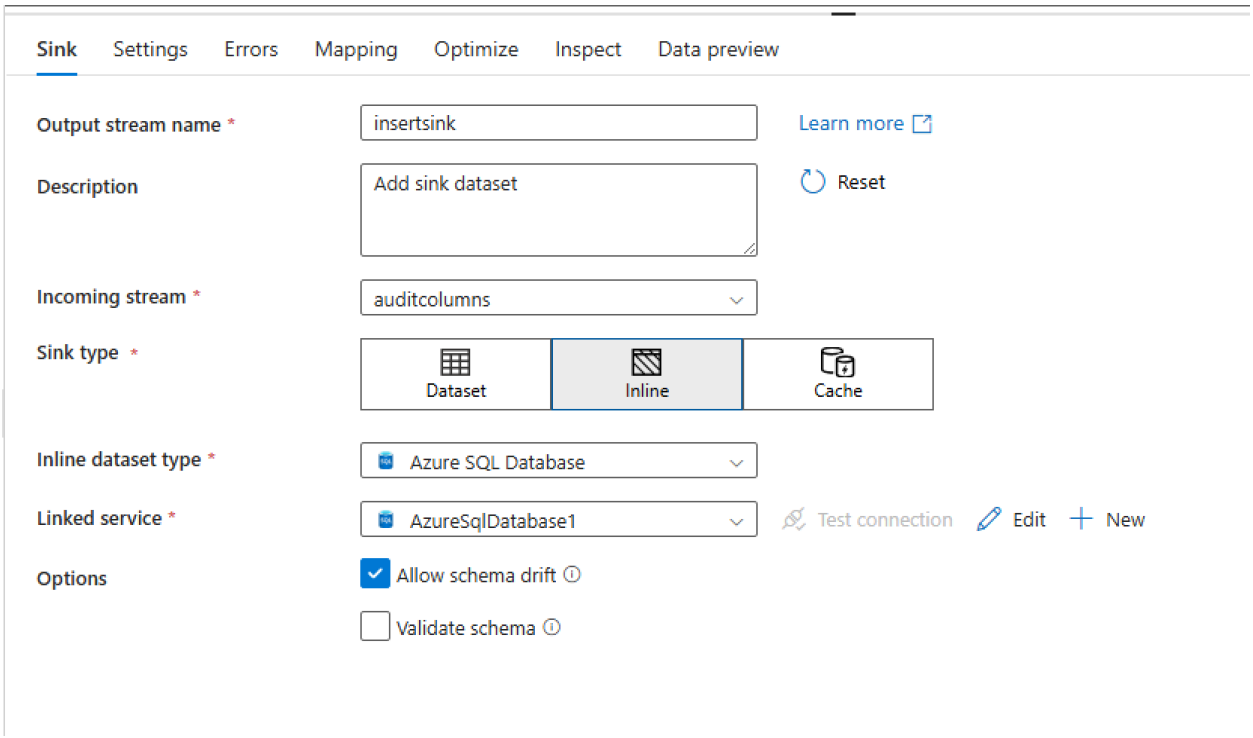
For update we are giving: src_account_id==account_id && src_hashkey!=hashkey



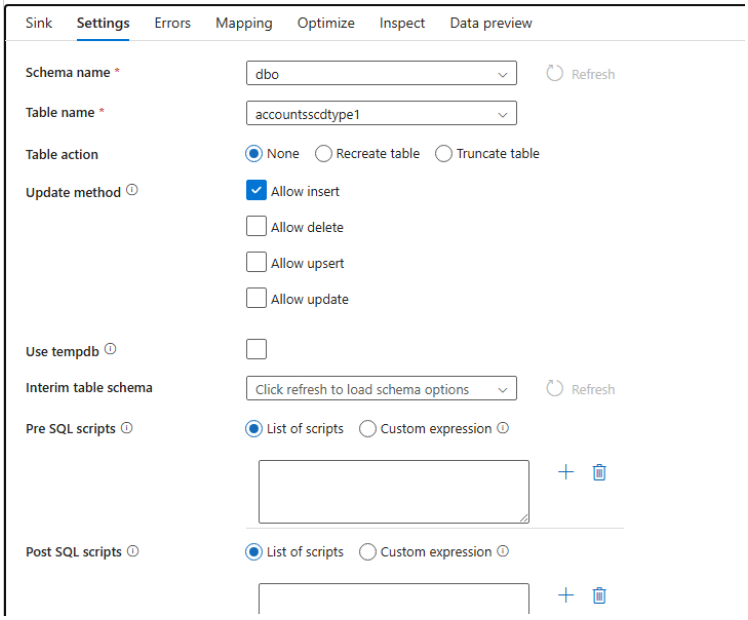
Now we are seeing the derived columns functions on insert side



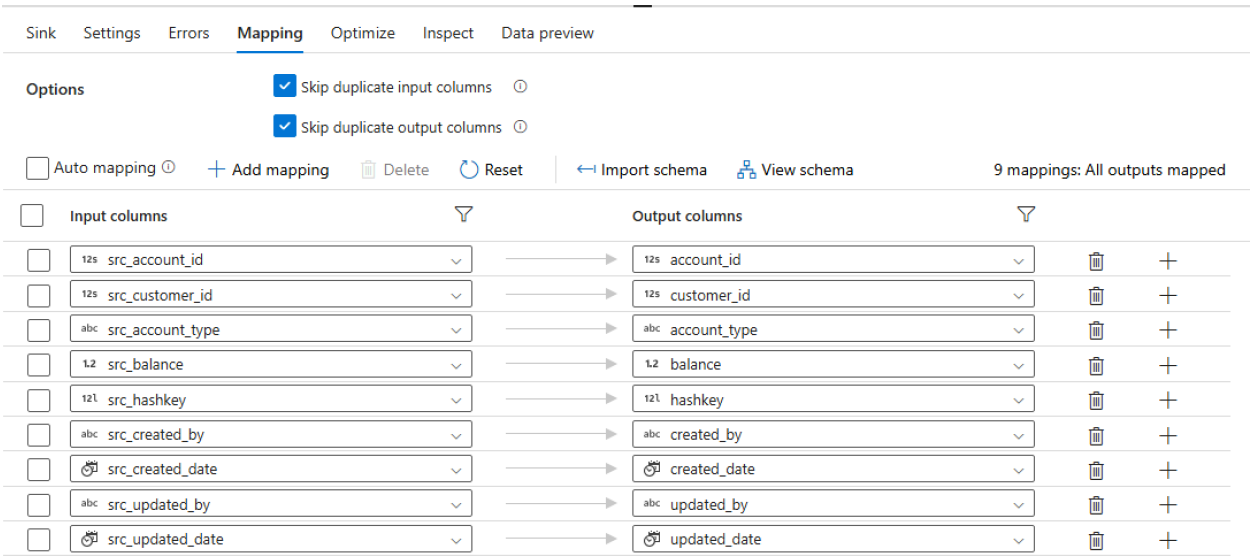
This is derived columns on the update side



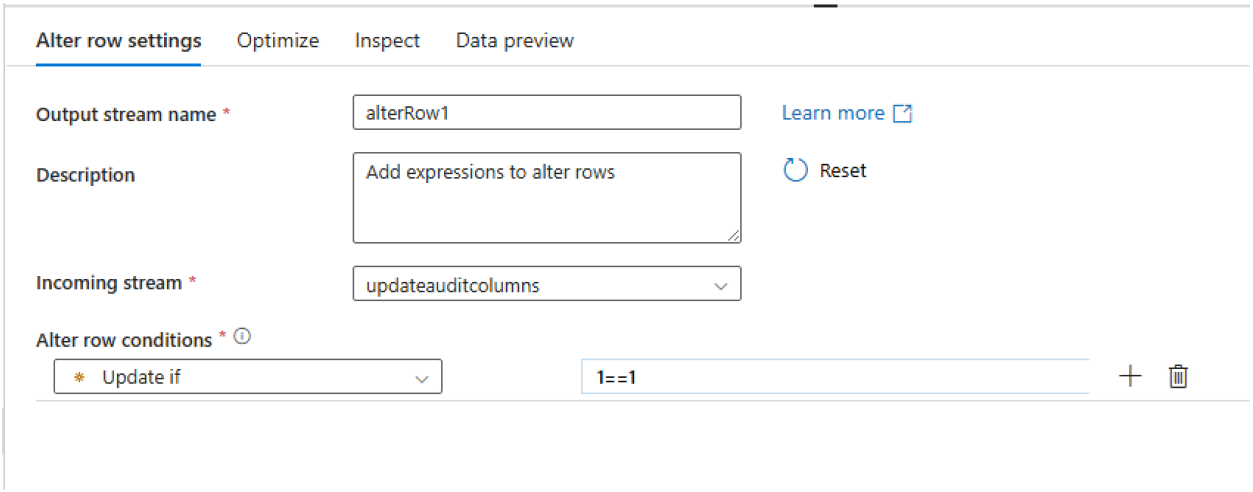
Here is the sink settings we are going to save the scd type 1 transformed data to sql database.



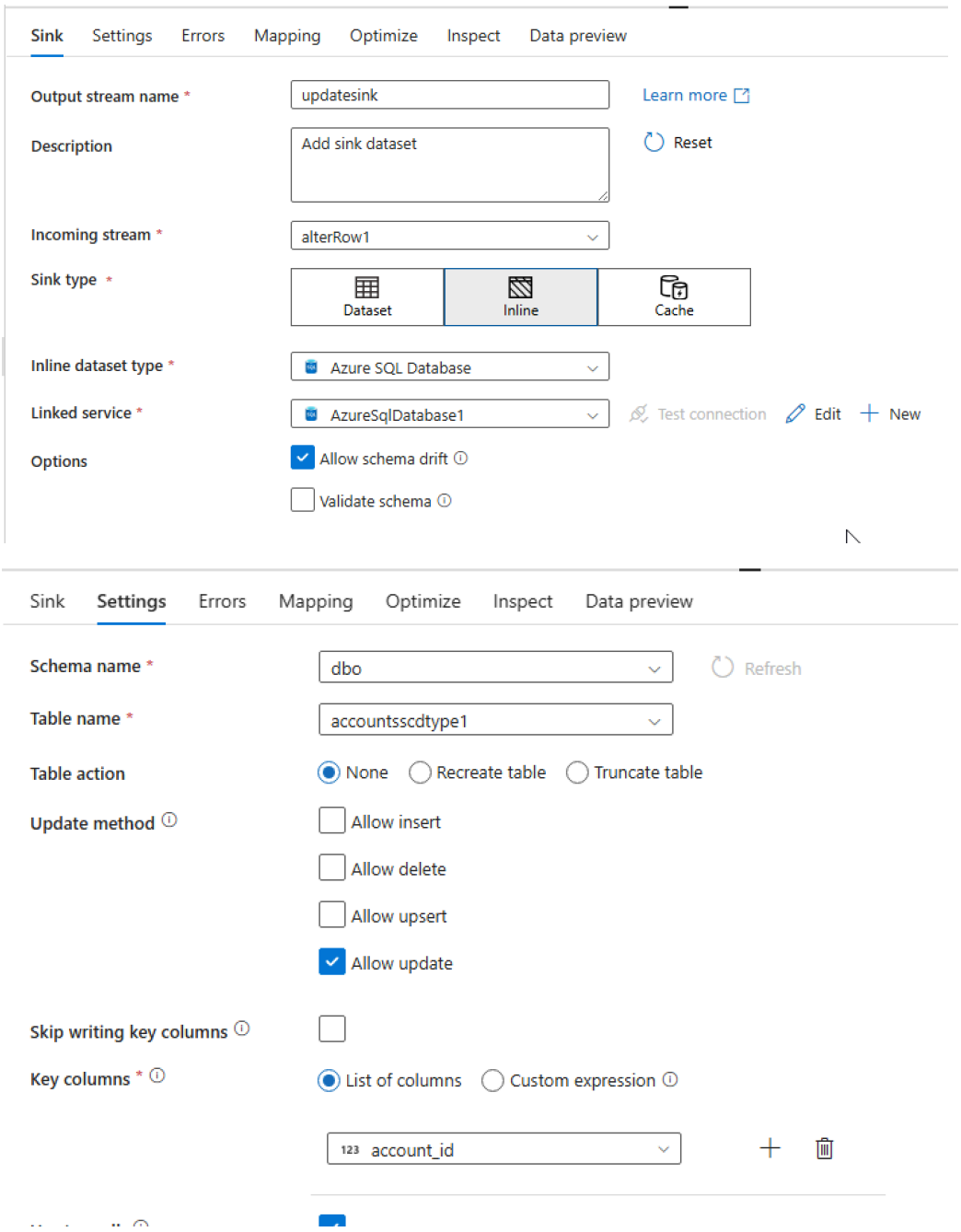
Select the dbo schema and select the table required and click on allow insert for insertion



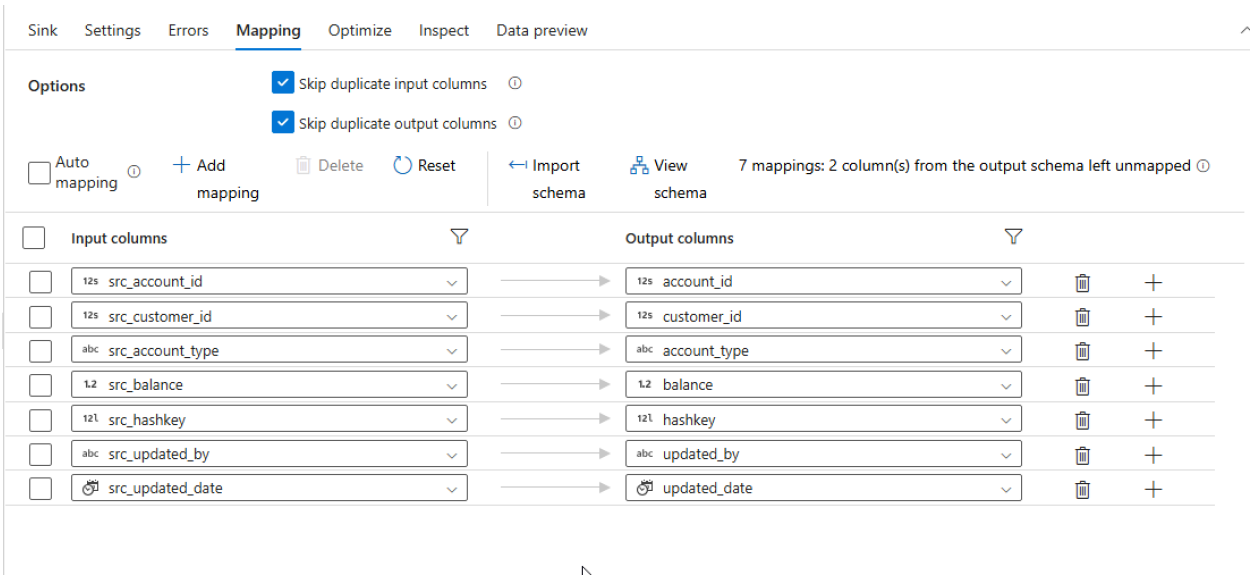
And then set the mapping for the sql table to allow data to be inserted properly



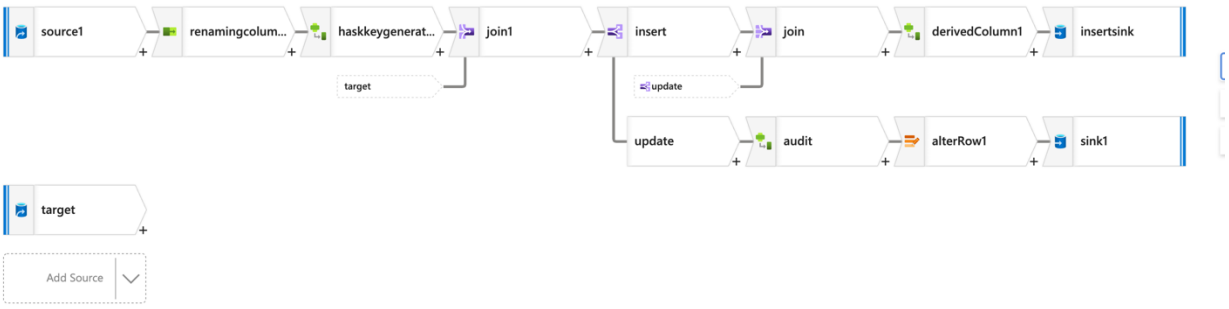
Update condition for updates in scd type 1



Here is the update side sink details

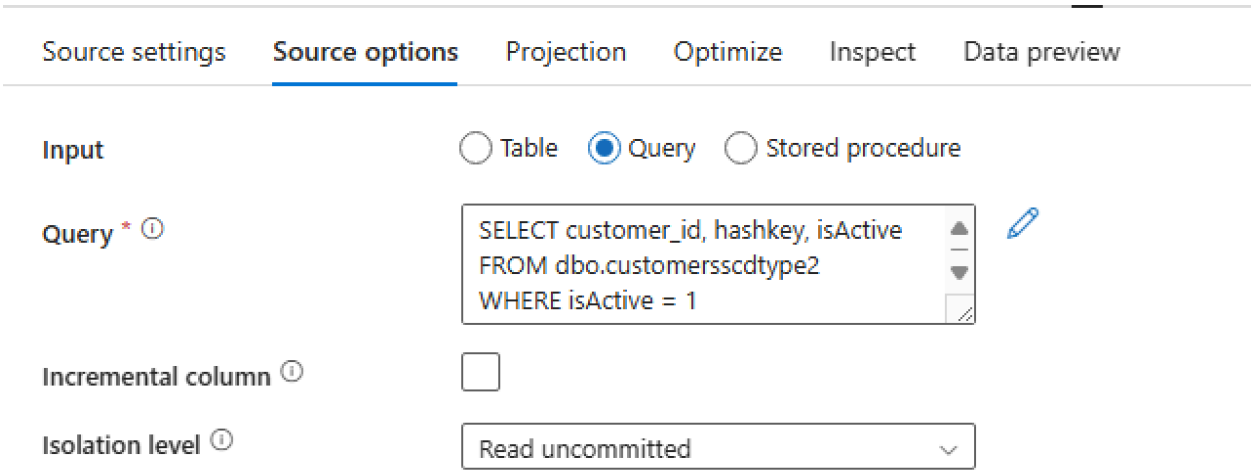


Update side mapping

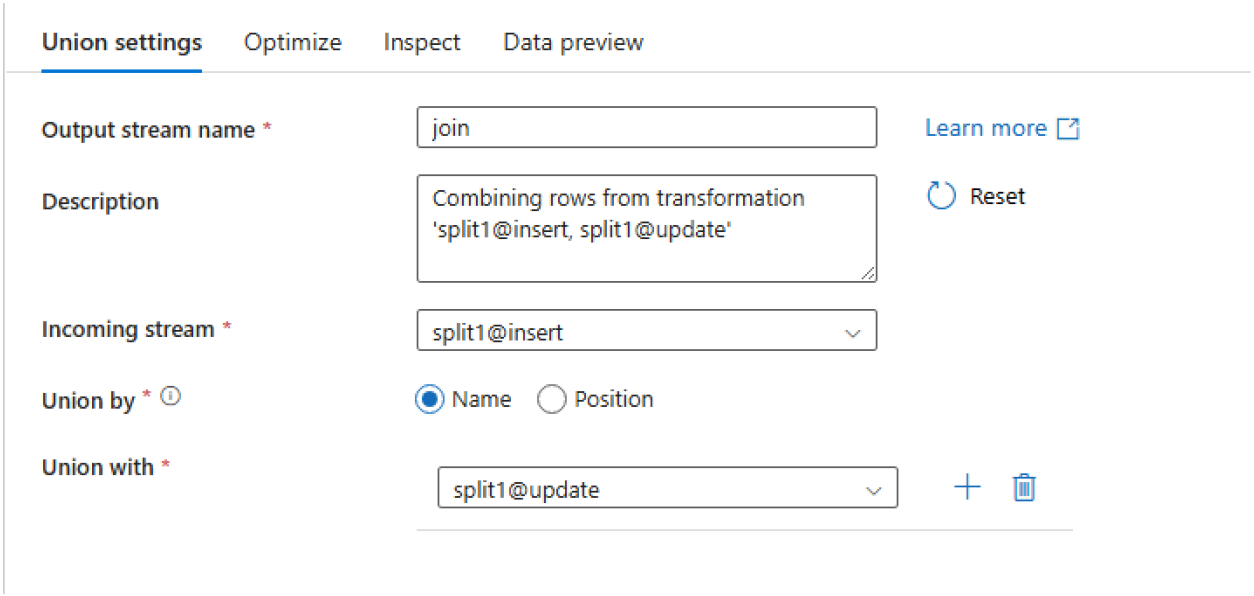


SCD type 2 transformation

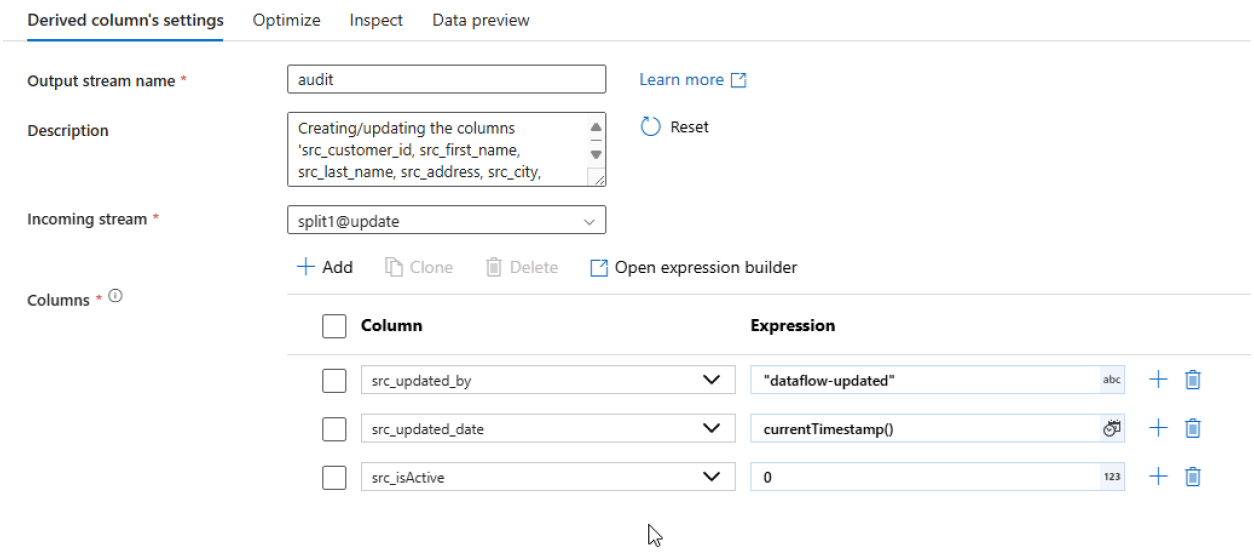
Firstly we need to import the schema from source and then create tables in sql database table. Which will then be added into the target to perform SCD transformations.



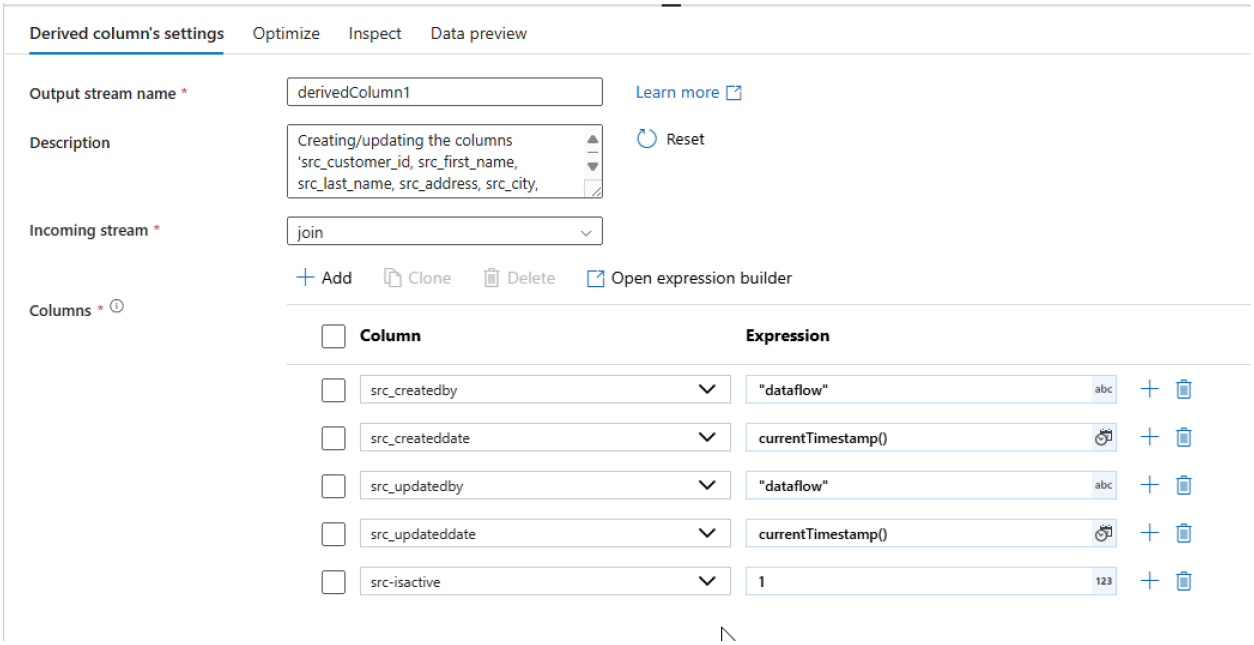
Here is the target side source setting we select the customer_id,haskey, isactive where isactive ==1. We are doing this because we are only editing data which is active. We don't care about unactive data



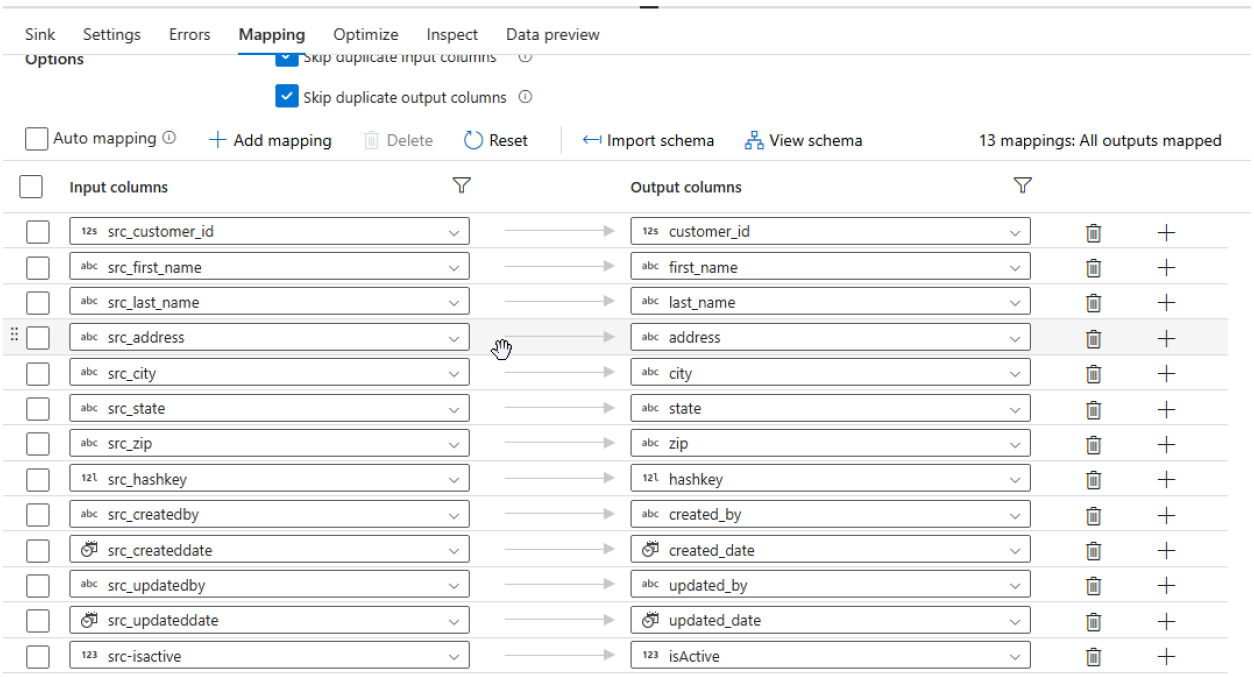
The above scd type 1 transformation up to conditional split is same. But after conditional split in scd type transformation we use derived columns to set the created_by updated_by and all. But here first we perform union between insert and update and then proceed to derived columns.



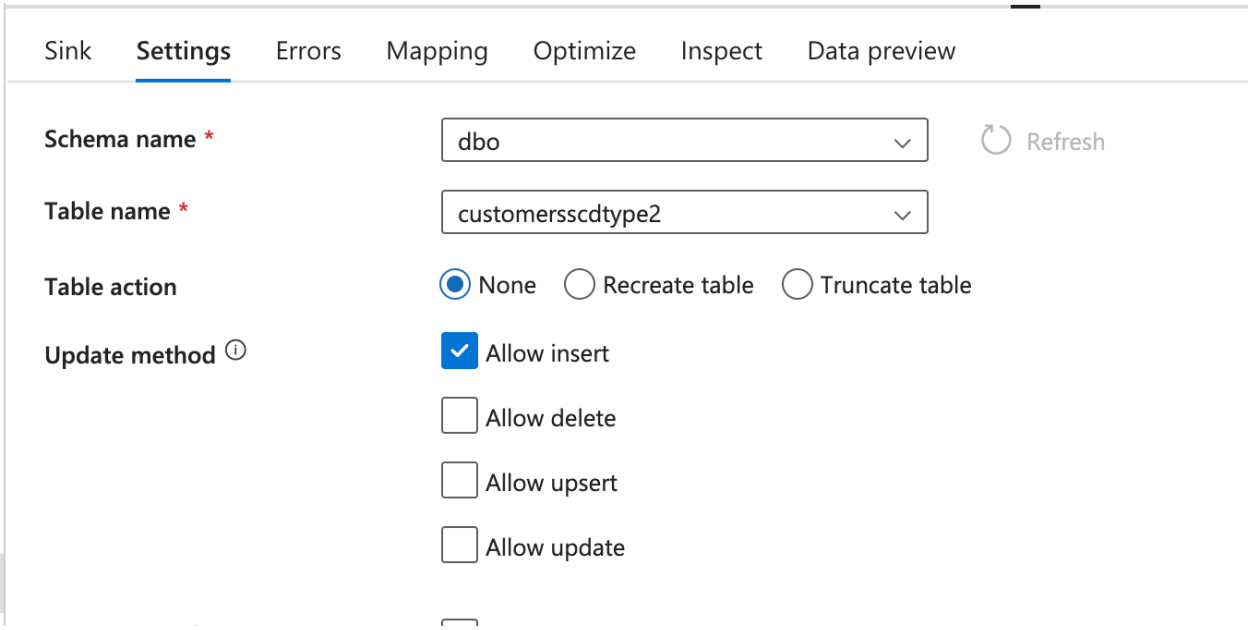
Here is the update side derived column



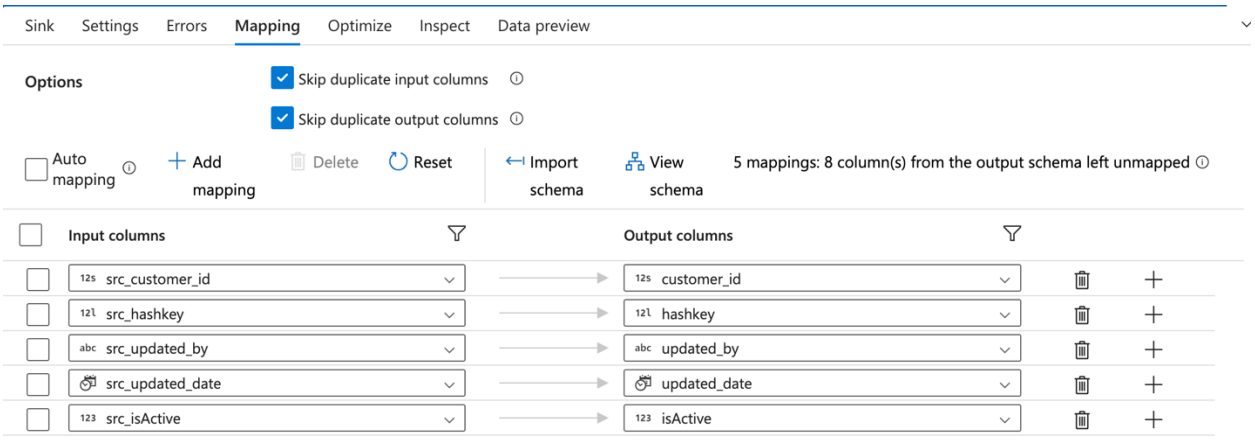
This is the insert side derived columns currenttimestamp() gives todays date and time at the time of insertion



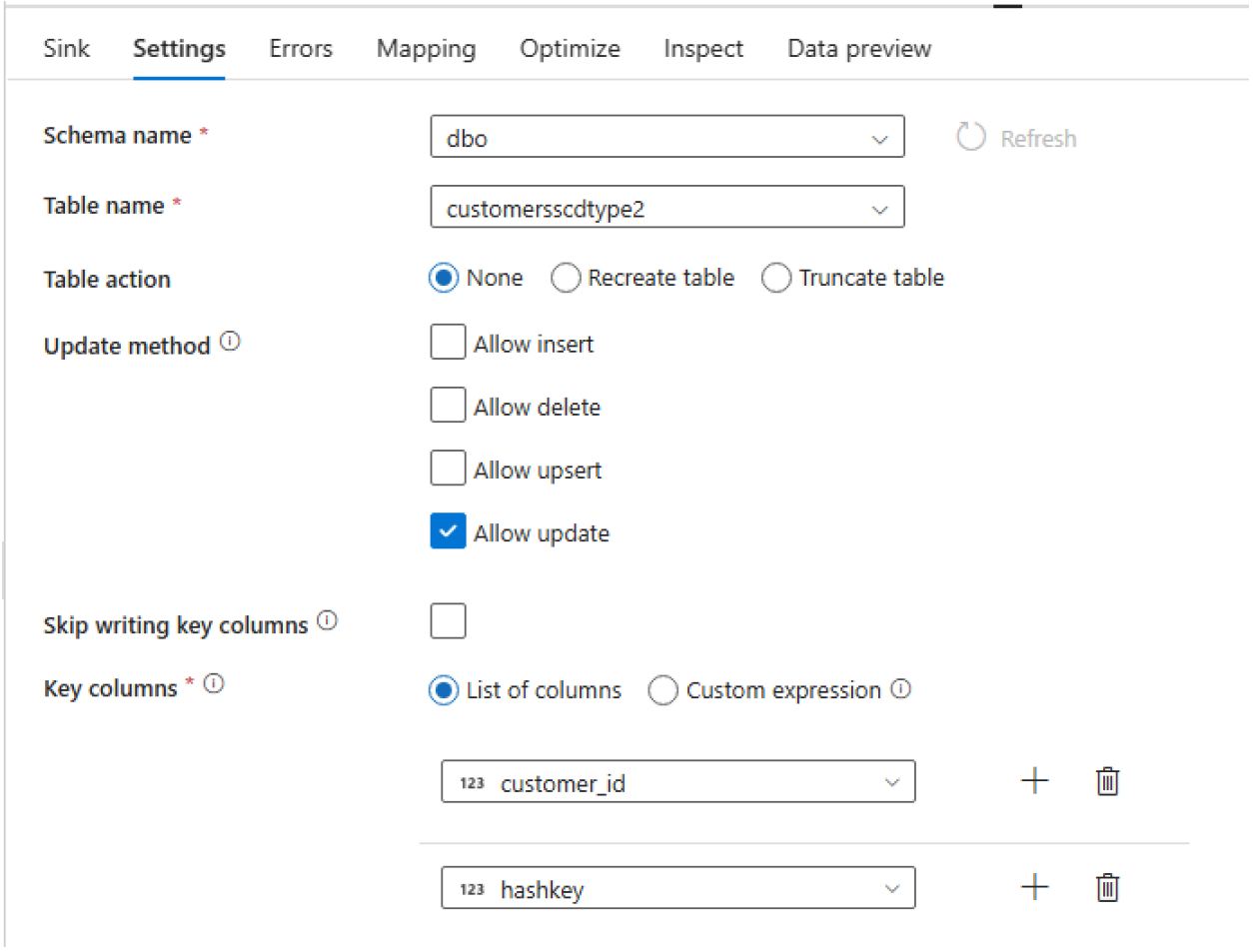
This is the mapping on the insert side of the pipeline



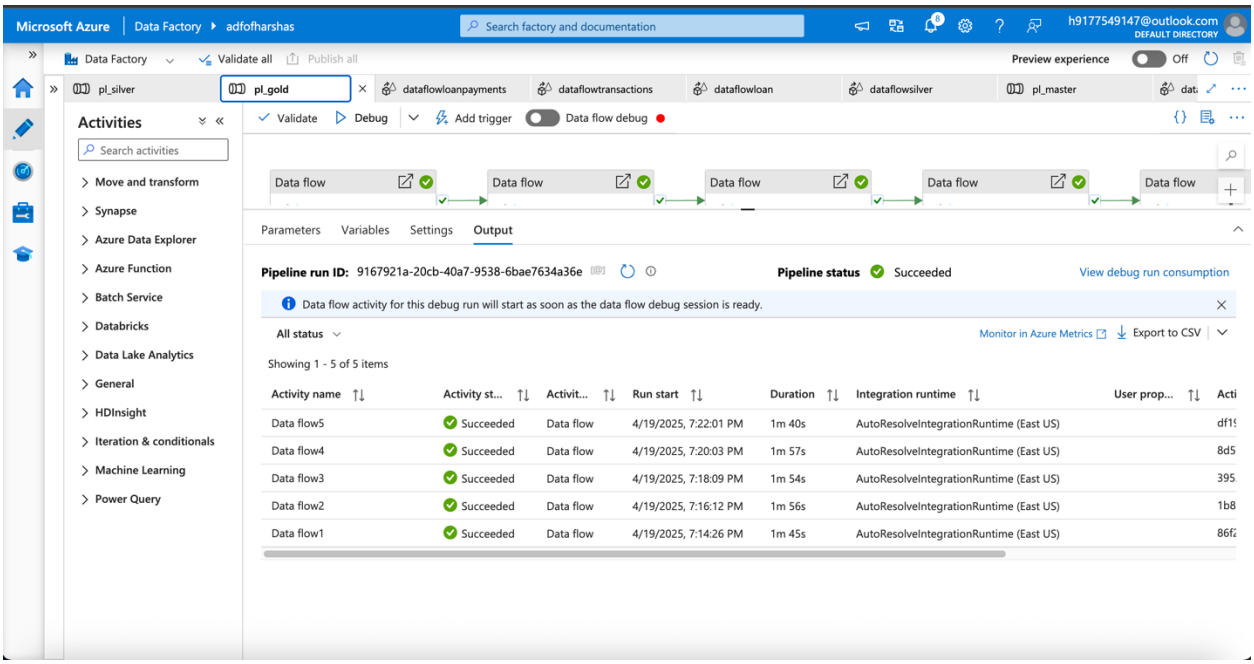
We are allowing the insert function for the insert side of the pipeline



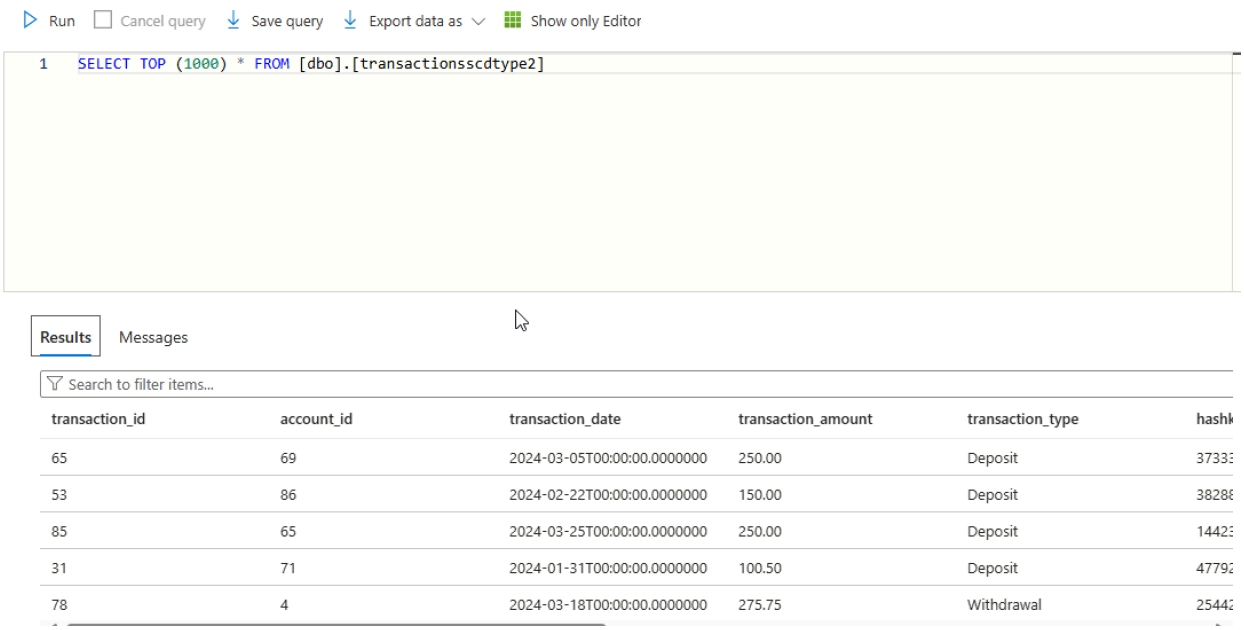
We only update haskey updated_by updated_date and isactive fields only in the update side



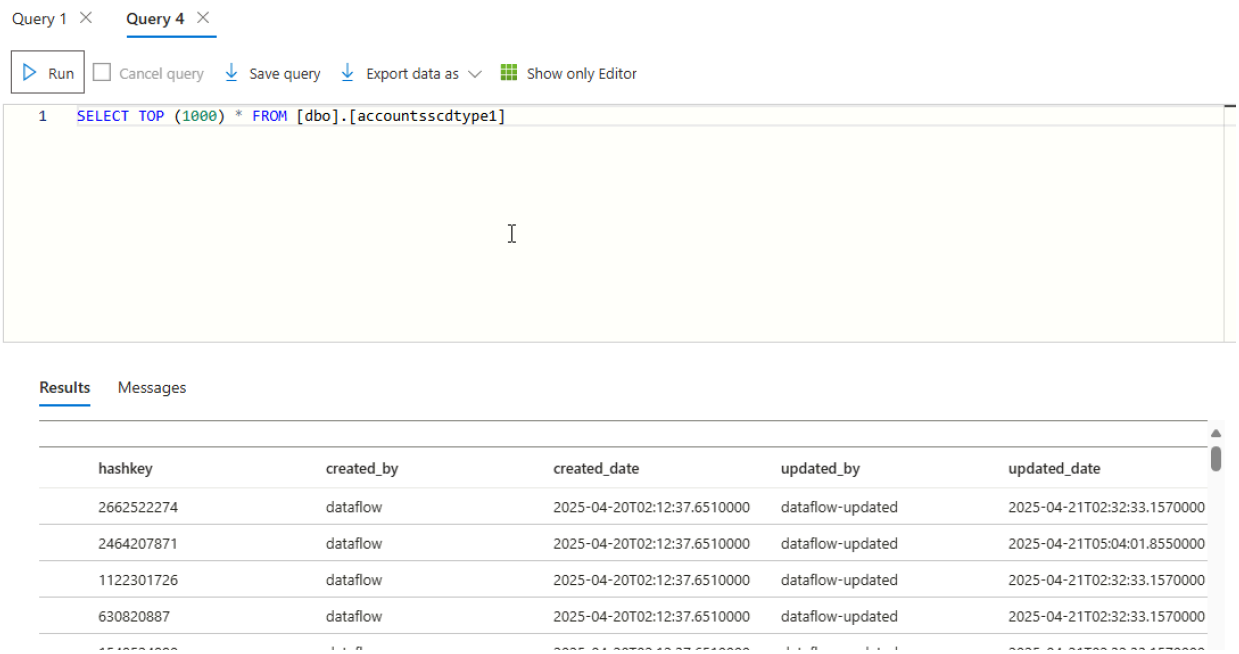
We are allowing update and are giving customer_id and hashkey for the key columns



Pipelines successfully run



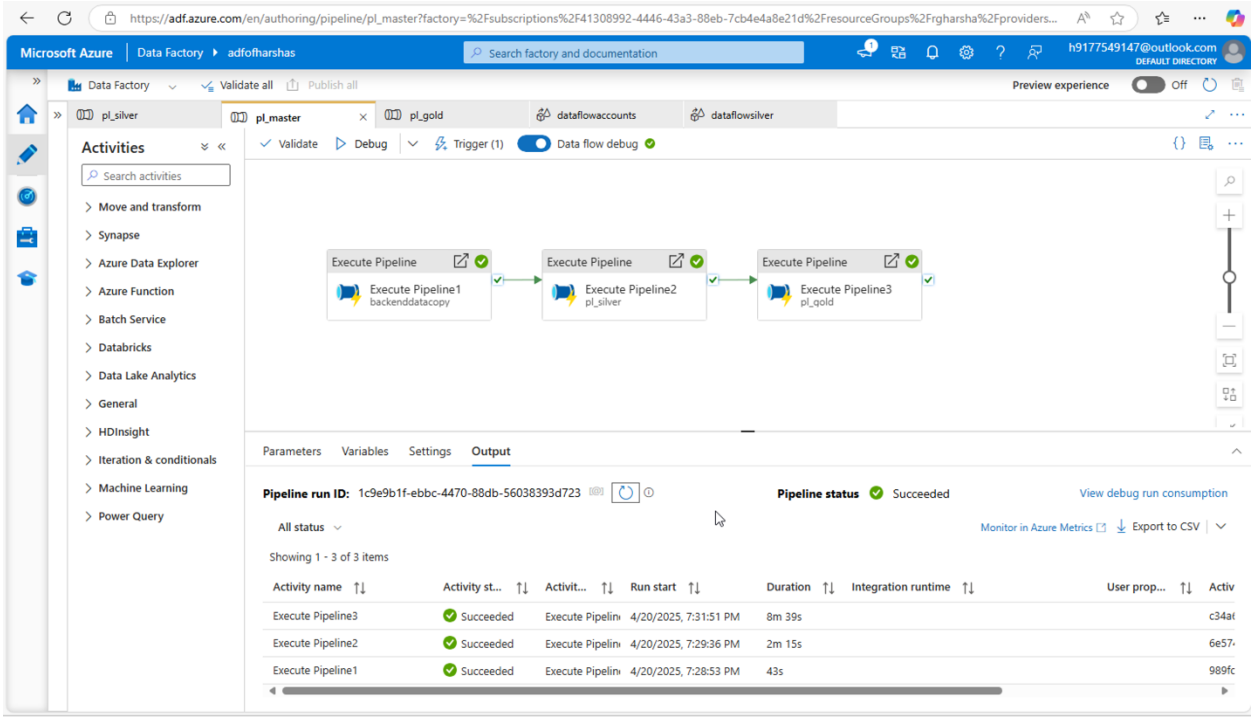
SCD type 2 data stored successfully in SQL database



SCD type 1 data successfully stored in SQL database.

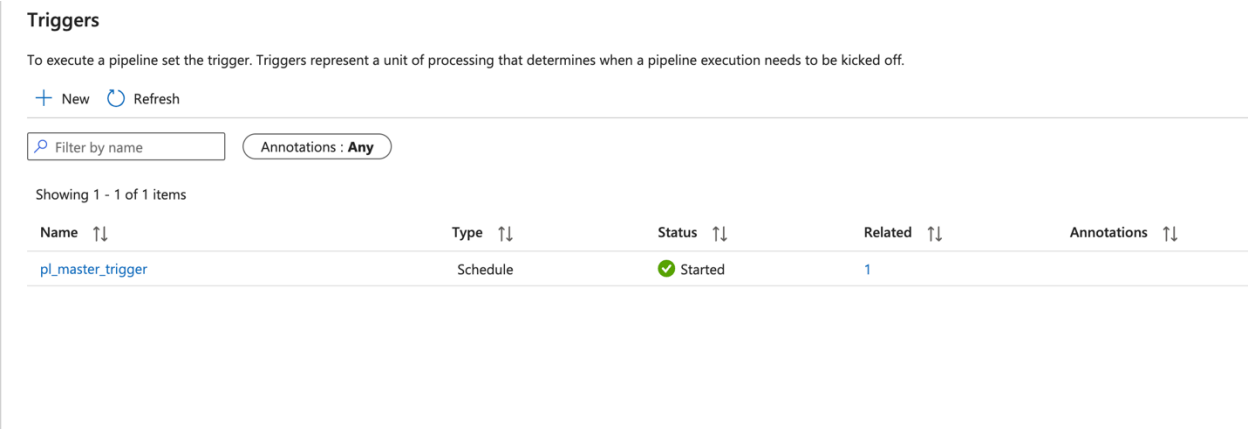
Challenges in step_3:

- In scd type 2 we while creating table we should not give any primary key



Master pipeline where all the 3 pipelines are connected and automatically triggered by a trigger.

In the above pipeline we have set a trigger



This trigger is enabled and the master pipeline is executing pipelines.

Execute pipeline is an activity where we select the child pipelines to run in the master pipeline. Master pipeline run acitivity:

- Firstly backend copy activity will be running (bronze layer)
- The pl_silver pipeline will be running to create delta format files. (Silver layer)
- Pl_gold where the pipeline created gold layer in sql database.

While running the master pipeline it failed and the reason is “failed to connect to sql database”. After giving it sometime the pipeline successfully executed.