DATA PIPELINE FOR CUSTOMER ACCOUNT ANALYSIS

Bootcamp Project - 1

APRIL 22, 2025

BY HARJINDER SINGH

Table of Contents

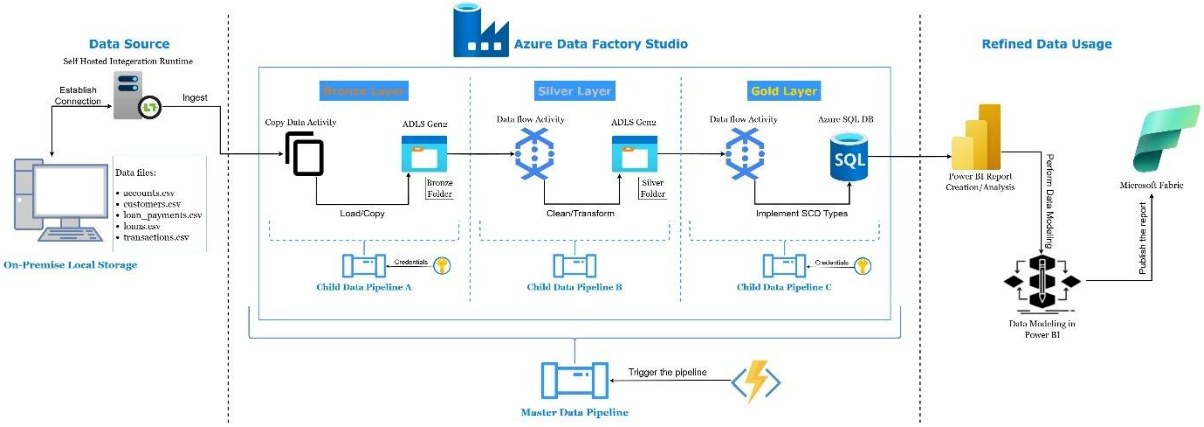
Introduction

In today’s data-driven landscape, the ability to efficiently process and transform data is crucial for deriving actionable insights and supporting business intelligence. This project focuses on the design and implementation of a robust and scalable data pipeline tailored for processing customer account data. The solution involves seamlessly copying raw data from a backend team's storage account, executing essential transformations using Azure Data Factory (ADF), and performing upsert operations—either inserting or updating records—into a SQL database. The source data resides in the GOLD tier of Azure Data Lake Storage (ADLS), which ensures high availability and performance for analytical workloads. By integrating these technologies, the pipeline ensures accurate, timely, and scalable data processing that meets the analytics and reporting needs of downstream systems, ultimately empowering better decision-making across the organization.

Source Data files used:

– accounts.csv
– customers.csv
– loan_payments.csv
– loans.csv
– transactions.csv

the following **architecture** will be used for this Project.

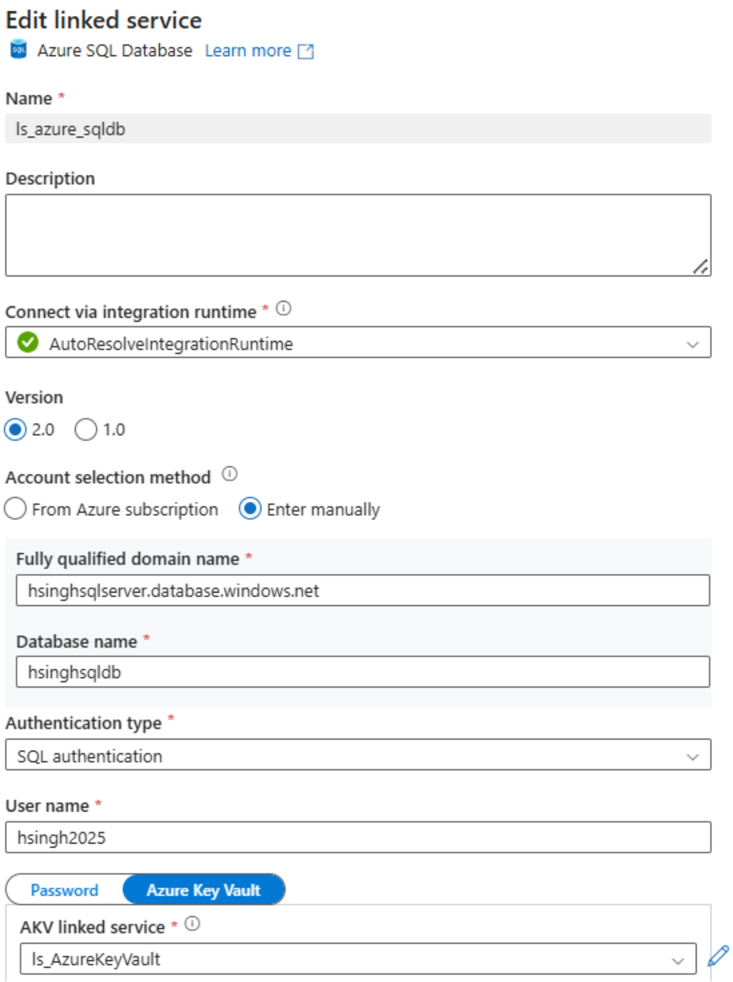


Linked Services

In this project, several linked services were configured within Azure Data Factory (ADF) to establish secure and efficient connections to various data sources and services. Linked services define the connection information required for ADF to access external data systems. The following linked services were used:

1. **ls_azure_sqldb (Azure SQL Database)**
   This linked service is configured to connect to the target Azure SQL Database where the transformed customer account data is upserted. It ensures secure and reliable data integration for downstream analytics and reporting. A total of 5 datasets or activities are associated with this service.

   The below configuration setting has been used to create this linked service.



Edit linked service

Azure SQL Database Learn more

Name *
ls_azure_sqldb

Description

Connect via integration runtime *
AutoResolveIntegrationRuntime

Version
● 2.0 ○ 1.0

Account selection method
○ From Azure subscription ● Enter manually

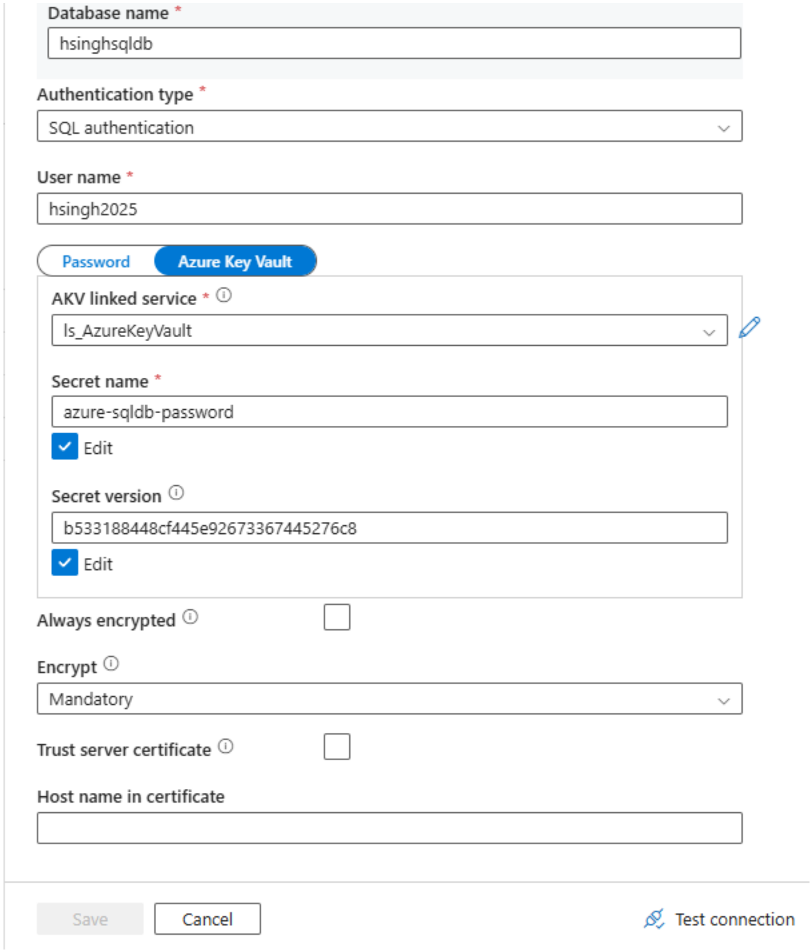
Fully qualified domain name *
hsinghsqlserver.database.windows.net

Database name *
hsinghsqldb

Authentication type *
SQL authentication

User name *
hsingh2025

AKV linked service *
ls_AzureKeyVault



Database name *

hsinghsqldb

Authentication type *

SQL authentication

User name *

hsingh2025

AKV linked service *

ls_AzureKeyVault

Secret name *

azure-sqldb-password

☑ Edit

Secret version

b533188448cf445e92673367445276c8

☑ Edit

Always encrypted □

Encrypt

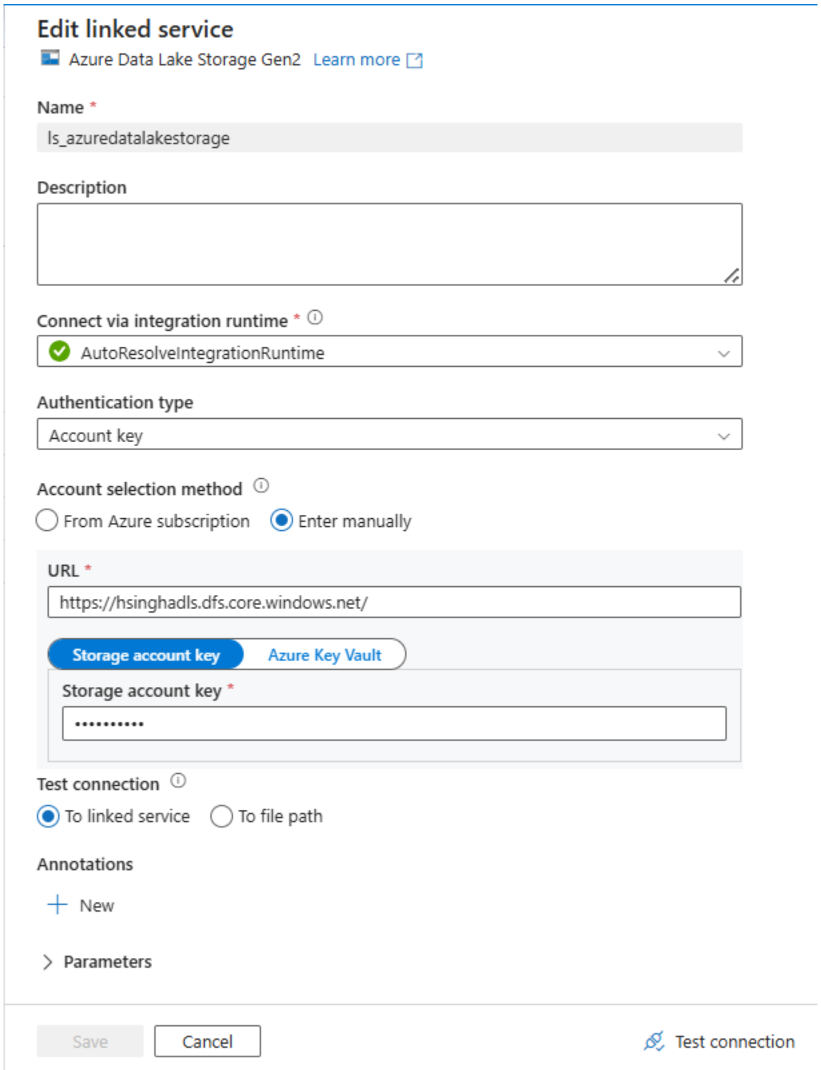
Mandatory

Trust server certificate □

Host name in certificate

2. **ls_azuredatalakestorage (Azure Data Lake Storage Gen2)**
   This service provides access to Azure Data Lake Storage Gen2, particularly the GOLD layer where the curated data files are stored. It enables ADF to read and write data as part of the transformation and data movement processes. This linked service is associated with 8 datasets or activities.

   The below configuration setting has been used to create this linked service.



Edit linked service

Azure Data Lake Storage Gen2 Learn more

Name *

ls_azuredatalakestorage

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method

○ From Azure subscription ● Enter manually

URL *

https://hsinghadls.dfs.core.windows.net/

Storage account key | Azure Key Vault

Storage account key *

••••••••••

Test connection

● To linked service ○ To file path

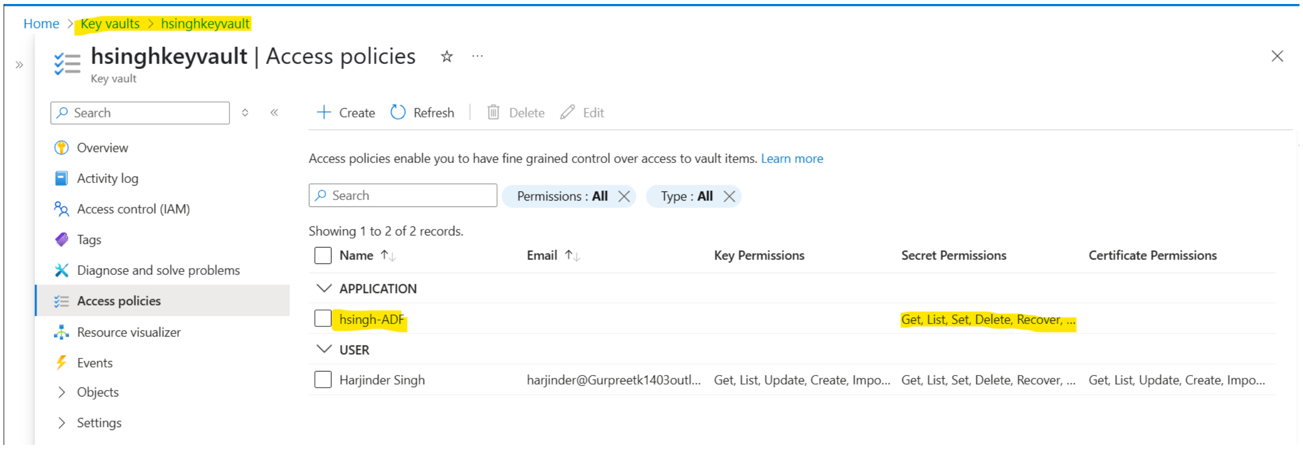
Annotations

+ New

> Parameters

3. **ls_AzureKeyVault (Azure Key Vault)**
   The Azure Key Vault linked service is used to securely retrieve credentials, such as connection strings and secret keys, required by other linked services. This ensures sensitive information is handled securely. It is associated with 2 activities or configurations.

   Before creating this Linked service, the secret permission to Data factory must be given under Access policies tab in order to perform operation on keyvault.



Home > Key vaults > hsinghkeyvault

hsinghkeyvault | Access policies

Key vault

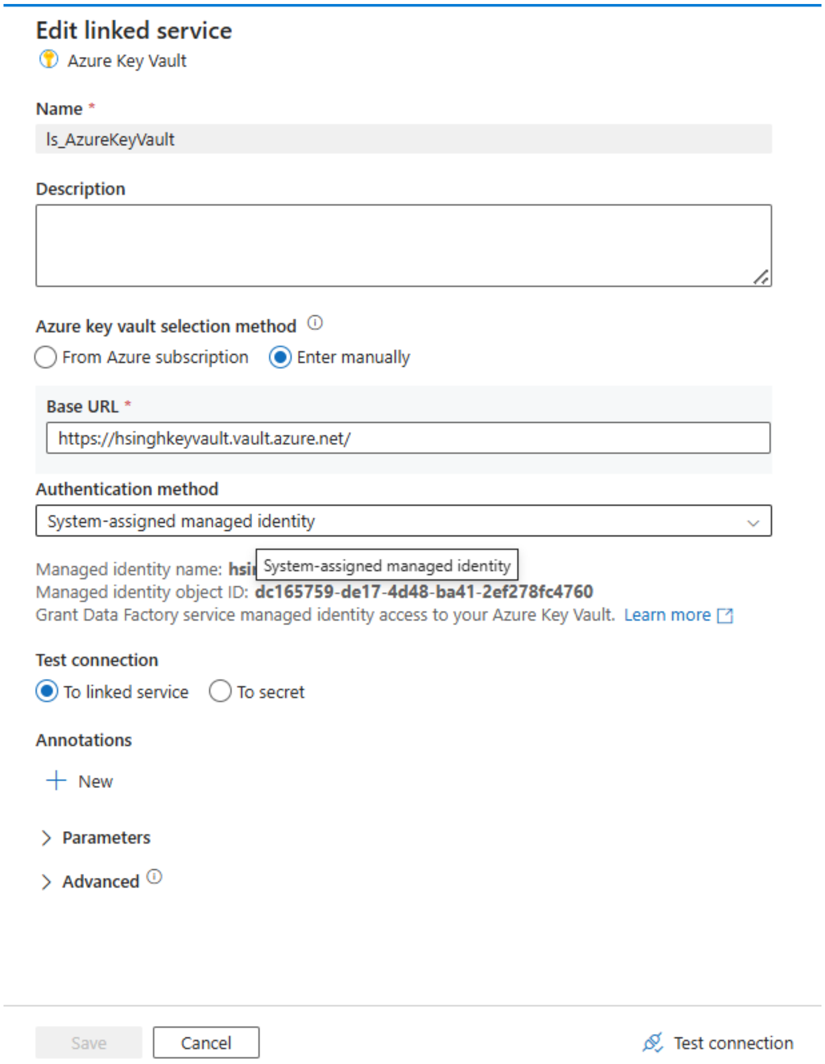
Access policies enable you to have fine grained control over access to vault items. Learn more

Search | Permissions : **All** | Type : **All**

Showing 1 to 2 of 2 records.

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions
APPLICATION				
□ hsingh-ADF			Get, List, Set, Delete, Recover, ...	
USER				
□ Harjinder Singh	harjinder@Gurpreetk1403outl...	Get, List, Update, Create, Impo...	Get, List, Set, Delete, Recover, ...	Get, List, Update, Create, Impo...

The below configuration setting has been used to create this linked service.



Edit linked service

Azure Key Vault

Name *

ls_AzureKeyVault

Description

Azure key vault selection method

○ From Azure subscription ● Enter manually

Base URL *

https://hsinghkeyvault.vault.azure.net/

Authentication method

System-assigned managed identity

Managed identity name: **hsi** System-assigned managed identity

Managed identity object ID: **dc165759-de17-4d48-ba41-2ef278fc4760**

Grant Data Factory service managed identity access to your Azure Key Vault. Learn more

Test connection

● To linked service ○ To secret

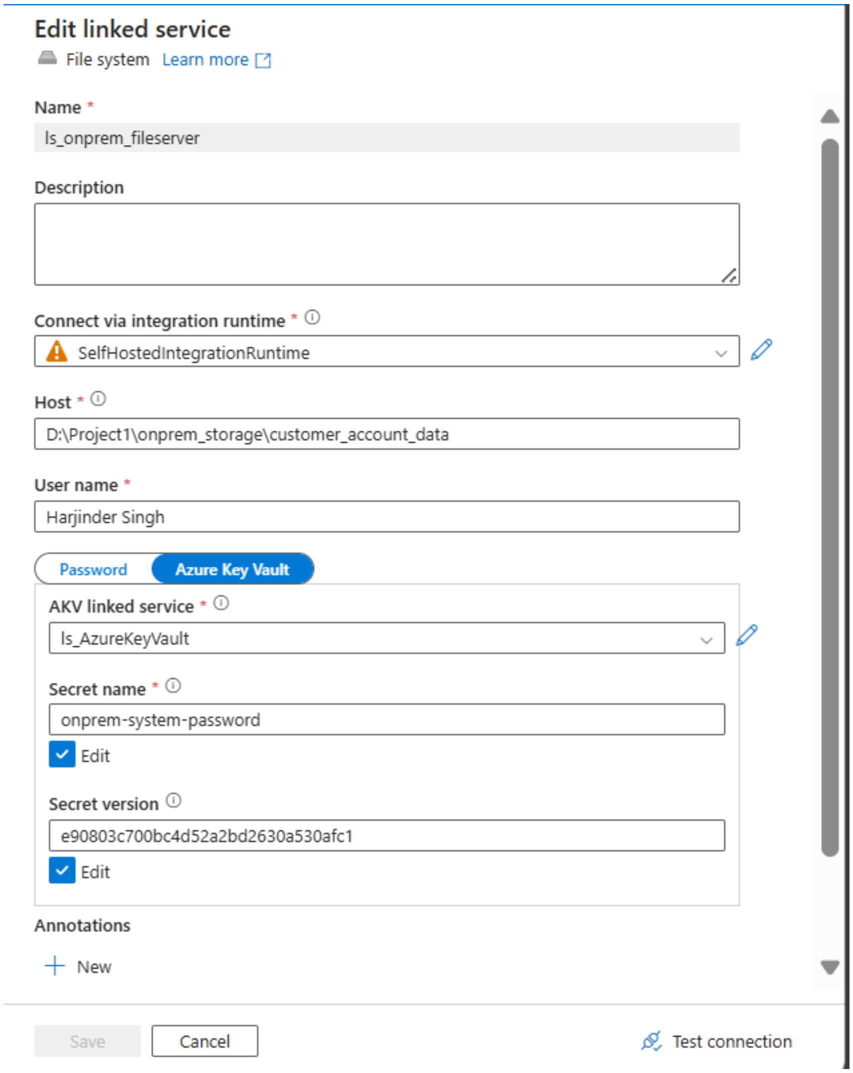
Annotations

4. **ls_onprem_fileserver (File System)**

This linked service is used to access a file system, which may represent an on-premises file server. This connection enables integration with legacy or local file sources. One dataset or activity is associated with this linked service.

To create this liked service, **Self Hosted Integration Runtime** has been configured in **the on-prem Local machine** and in Azure Data Factory to make a secure connection using the secret key.

The below configuration setting has been used to create this linked service.



Edit linked service

File system Learn more

Name *

ls_onprem_fileserver

Description

Connect via integration runtime *

SelfHostedIntegrationRuntime

Host *

D:\Project1\onprem_storage\customer_account_data

User name *

Harjinder Singh

Password | Azure Key Vault

AKV linked service *

ls_AzureKeyVault

Secret name *

onprem-system-password

☑ Edit

Secret version

e90803c700bc4d52a2bd2630a530afc1

☑ Edit

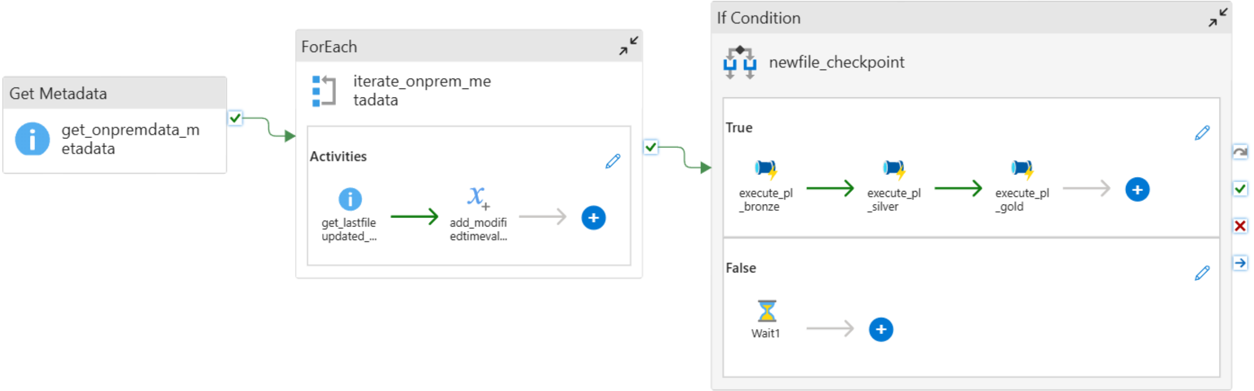
Annotations

Together, these linked services form the backbone of the data pipeline, enabling seamless data movement and transformation across various environments and systems.

Master Pipeline

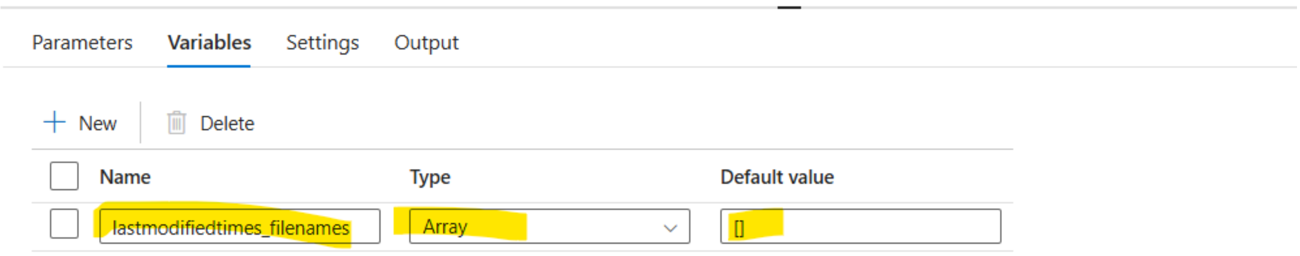
Here, I have created a Master Pipeline that will execute 3 child pipelines (Bronze, Silver, and Gold Layer pipeline). In the Master pipeline, I have used **2 get metadata activity, 1 foreach activity, 1 append variable activity, 1 if condition activity, 3 execute pipeline activity and 1 wait condition activity.**

Using all the mentioned activity, the master pipeline will always check for the last modified file in last 24 hours if there is not file modified in the last 24 hours then no activity will execute otherwise only modified file will be loaded in bronze layer and then only silver and gold layer pipeline will be executed.



➢ Steps To Create the Pipeline are as follows.

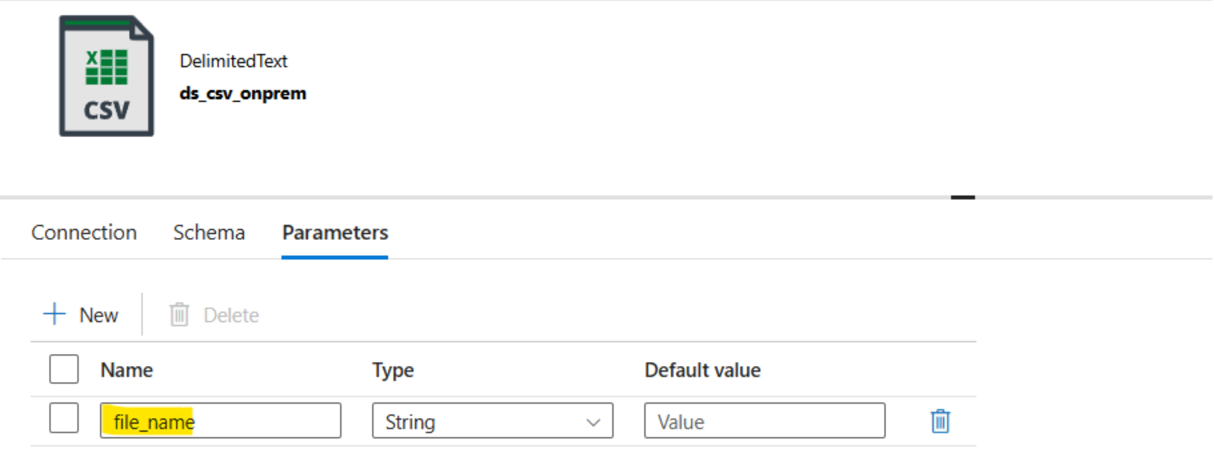
Step 1: Create new Pipeline and name it as *pl_master*. Create a new variable of type array and name it as *lastmodifiedtimes_filenames*.



□	Name	Type	Default value
□	lastmodifiedtimes_filenames	Array	[]

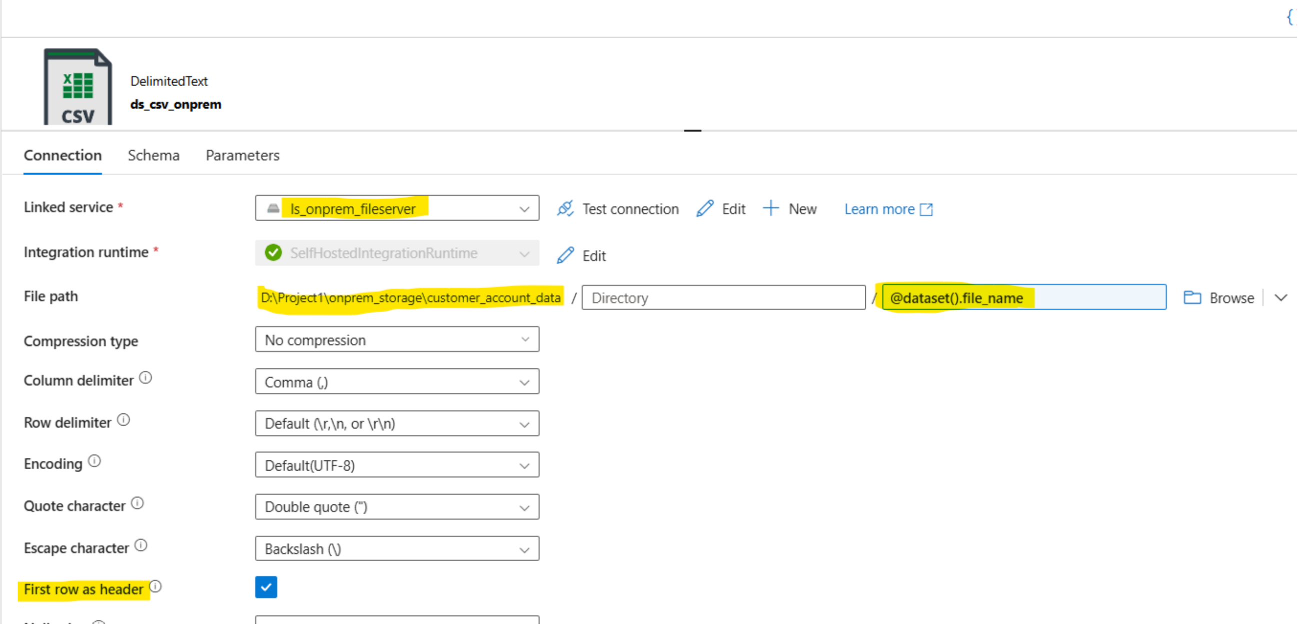
Step 2: Add a get metadata activity -> go to settings tab -> click on new Dataset -> select delimitedText and file server options -> create a parameter and name it as *file_name*.

- Enter **white spaces** in file_name data set properties.
- Select **Child Items** option as Field List.



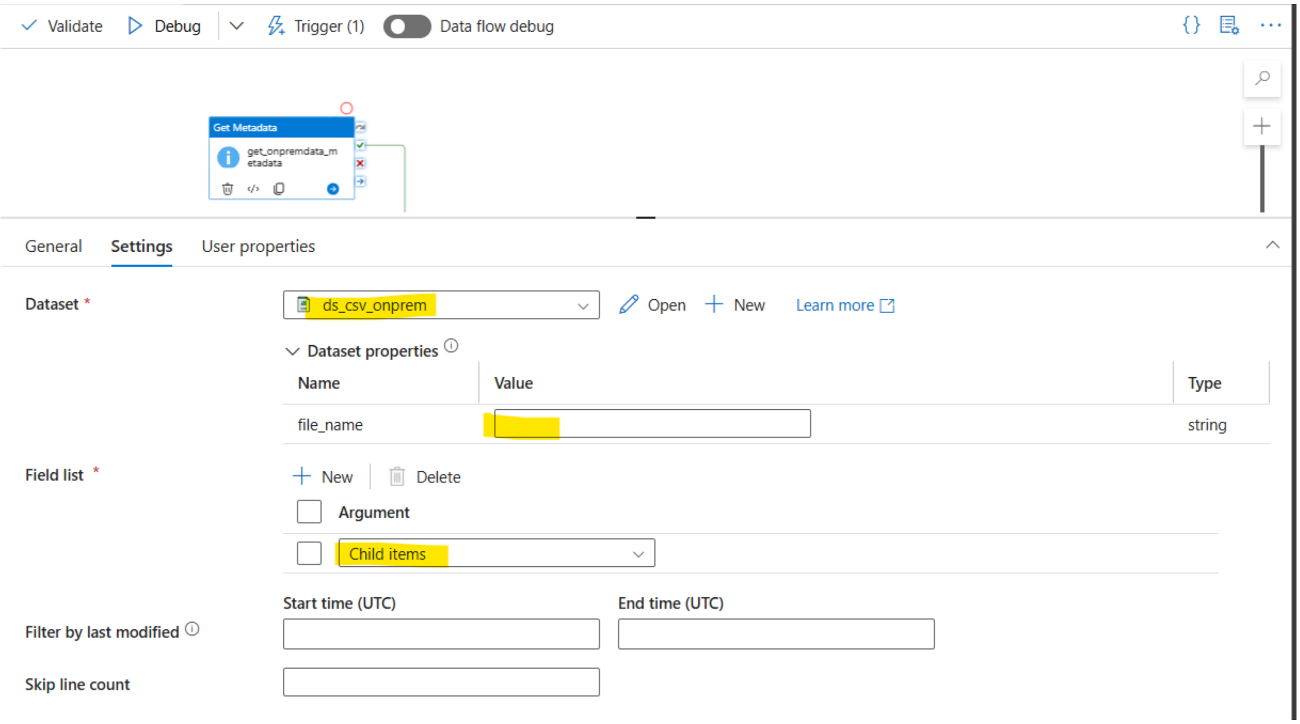
DelimitedText
ds_csv_onprem

□	Name	Type	Default value
□	file_name	String	Value



DelimitedText
ds_csv_onprem

Linked service * ls_onprem_fileserver
Integration runtime * SelfHostedIntegrationRuntime
File path D:\Project1\onprem_storage\customer_account_data / Directory / @dataset().file_name
Compression type No compression
Column delimiter Comma (,)
Row delimiter Default (\r,\n, or \r\n)
Encoding Default(UTF-8)
Quote character Double quote (")
Escape character Backslash (\)
First row as header ☑



General | Settings | User properties

Dataset * ds_csv_onprem Open New Learn more

Dataset properties

Name	Value	Type
file_name		string

Field list * New Delete

□ Argument

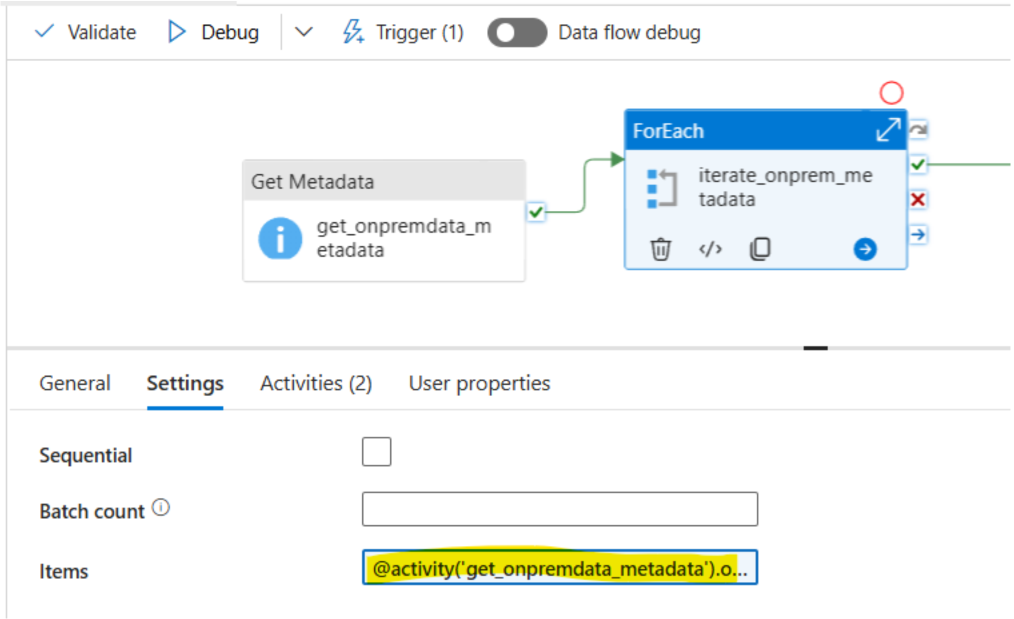
□ Child items

Filter by last modified — Start time (UTC) — End time (UTC)

Skip line count

Step 2: Add a get foreach activity-> connect foreach and get metadata with on success point -> go to settings tab of foreach -> enter below expression in the Items field.

- Expression Used:
  - **@activity('get_onpremdata_metadata').output.childItems**



General | Settings | Activities (2) | User properties

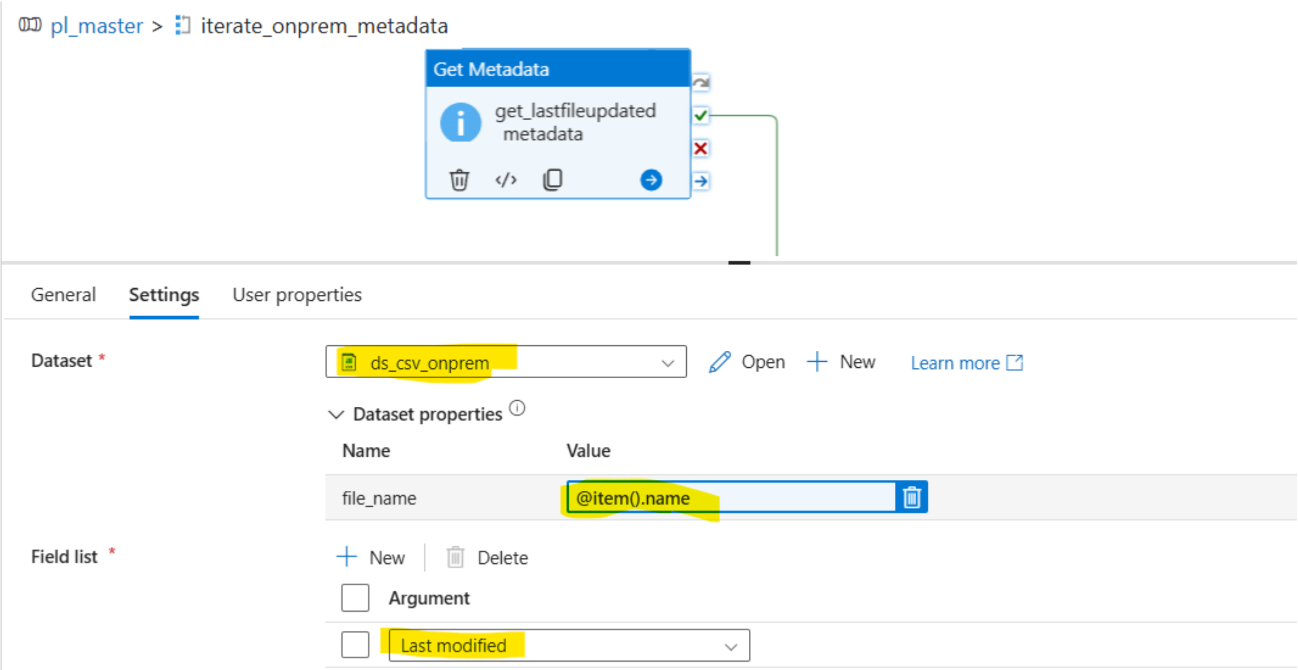
Sequential □

Batch count

Items @activity('get_onpremdata_metadata').o...

Step 3: Click on pencil to add activity in foreach -> add a get metadata activity in foreach -> select previously created dataset.

– Pass the **@item().name** from foreach in the **file_name** dataset properties.
– Select the **Last modified** option in Field List.



Dataset * : ds_csv_onprem

Dataset properties

Name	Value
file_name	@item().name

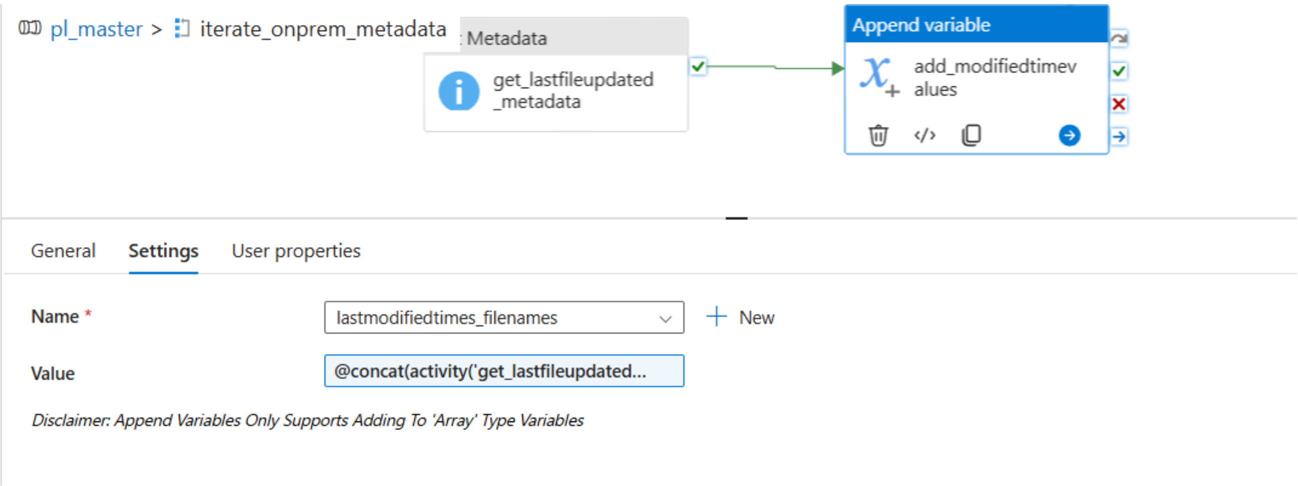
Field list *

□ Argument
□ Last modified

Step 4: Next Add Append Variable activity and connect it with get metadata activity using on success point -> go to settings tab of Append Variable -> select the variable and pass the below expression as value.

– Expression Used:
  - **@concat(activity('get_lastfileupdated_metadata').output.lastModified, ',,', item().name)**

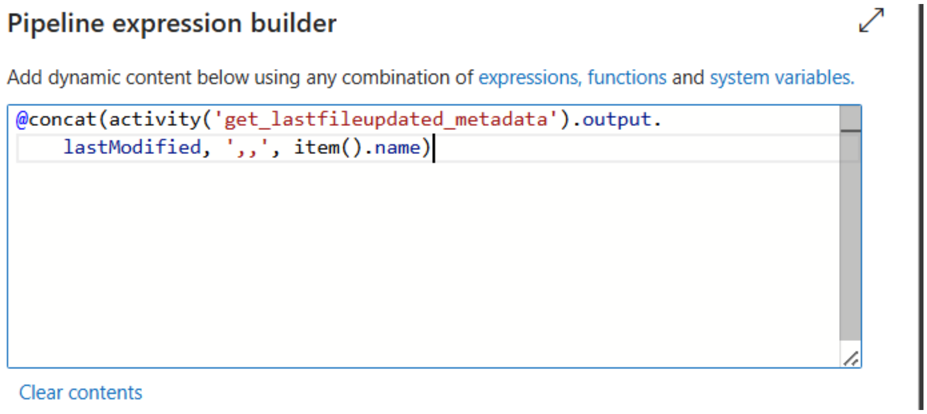
  This joins (concatenates) the last modified date of a file with the file name, separated by ',,'.



Name * lastmodifiedtimes_filenames

Value @concat(activity('get_lastfileupdated...

Disclaimer: Append Variables Only Supports Adding To 'Array' Type Variables



Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat(activity('get_lastfileupdated_metadata').output.
    lastModified, ',,', item().name)
```

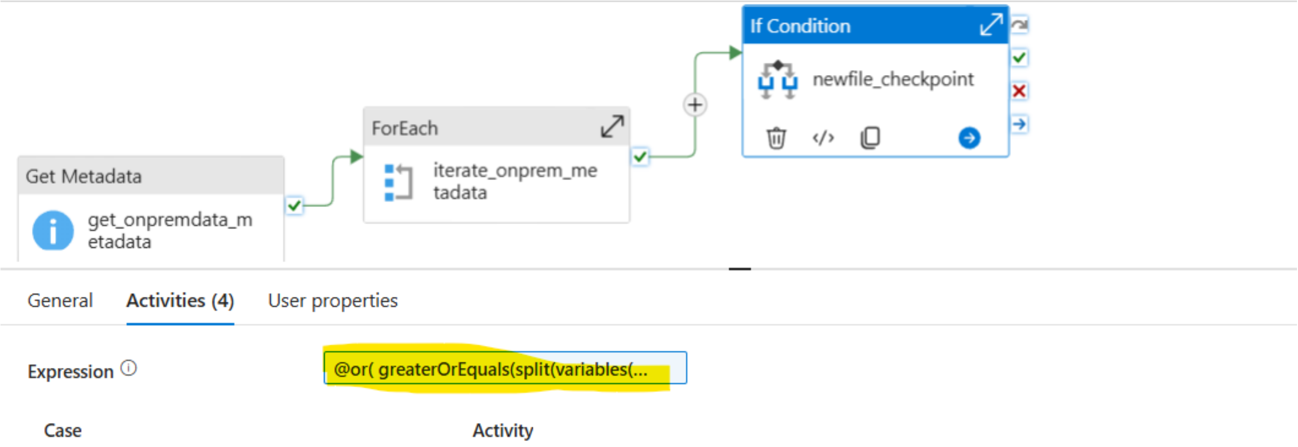
Step 4: Next, come to the pipeline, add if condition activity and connect it with foreach using on success point. -> go to activities tab -> add the below expression in Expression fields.

Expression Used:

@or(
  greaterOrEquals(split(variables('lastmodifiedtimes_filenames')[0], ',,')[0], addDays(utcNow(), -1)),
  or(
    greaterOrEquals(split(variables('lastmodifiedtimes_filenames')[1], ',,')[0], addDays(utcNow(), -1)),
    or(
      greaterOrEquals(split(variables('lastmodifiedtimes_filenames')[2], ',,')[0], addDays(utcNow(), -1)),
      or(
        greaterOrEquals(split(variables('lastmodifiedtimes_filenames')[3], ',,')[0], addDays(utcNow(), -1)),

```
            greaterOrEquals(split(variables('lastmodifiedtimes_filenames')[4], ',')[0], 
addDays(utcNow(), -1))
        )
      )
    )
)
```

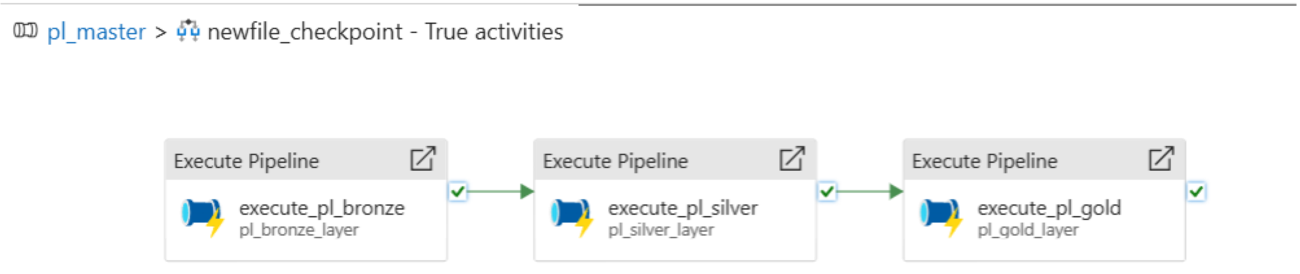
It checks if **any of the first five files** in the array were **modified in the last 24 hours** by comparing their timestamps to yesterday's date.



Expression: @or(greaterOrEquals(split(variables(...

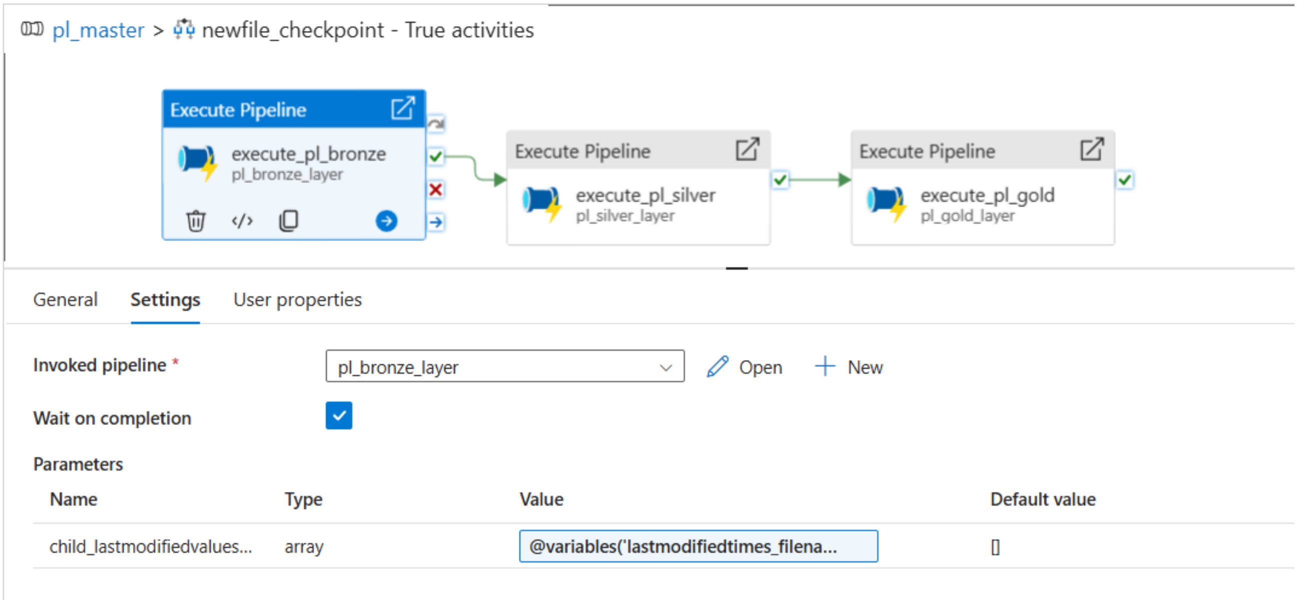
Step 4: Next, go to activities tab of if condition -> click on true section pencil icon to add activity in it.

Add 3 execute pipeline activities and connect it one after another using the on success point in serial (Bronze pipeline -> Silver pipeline -> Gold pipeline).



- In the first execute pipeline, select the bronze layer pipeline to be invoked and pass the following expression in the child pipeline parameter.
- Expression Used:
  - **@variables('lastmodifiedtimes_filenames')**

Here, I have passed the variable to child pipeline i.e., pl_bronze_layer with last modified and filename values.



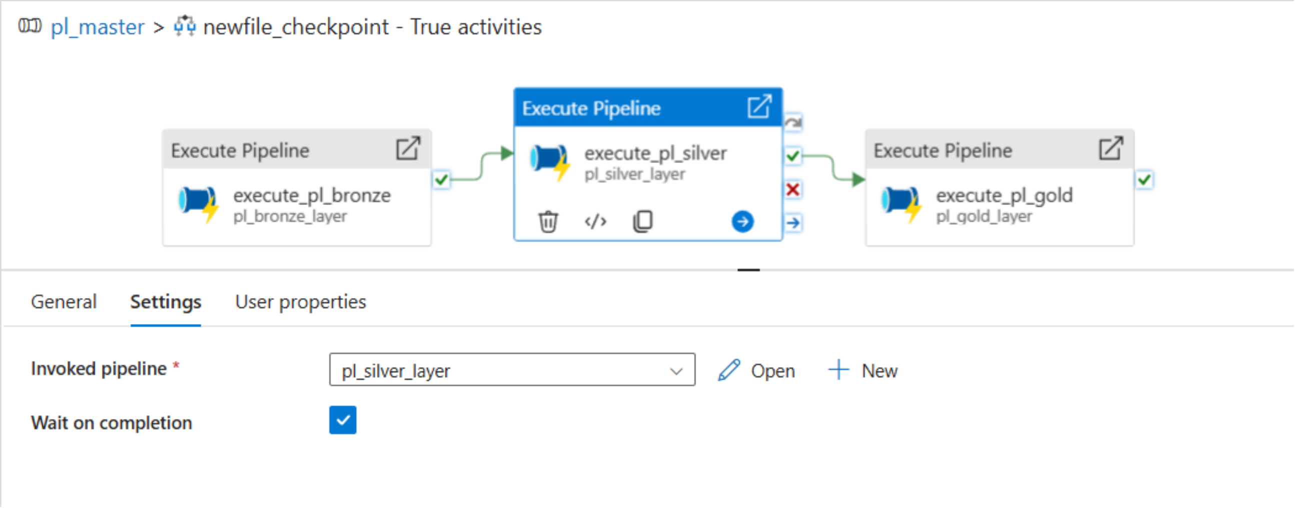
Invoked pipeline * pl_bronze_layer

Wait on completion ☑

Parameters

Name	Type	Value	Default value
child_lastmodifiedvalues...	array	@variables('lastmodifiedtimes_filena...	[]

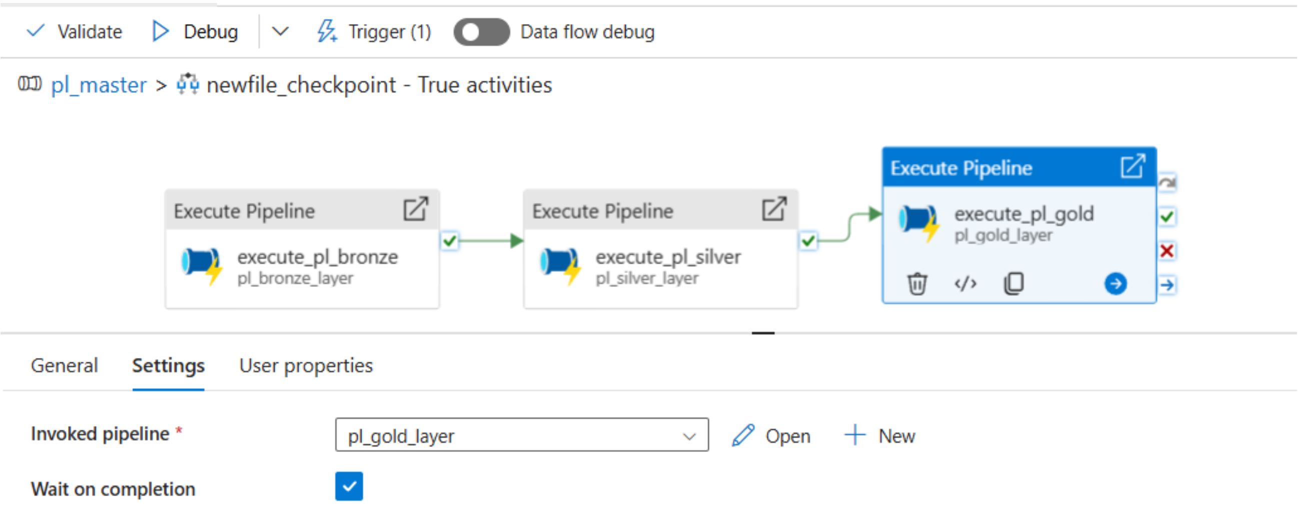
– In the second execute pipeline, select the silver layer pipeline to be invoked.



Invoked pipeline * pl_silver_layer

Wait on completion ☑

– In the third execute pipeline, select the gold layer pipeline to be invoked.



pl_master > newfile_checkpoint - True activities

Invoked pipeline * pl_gold_layer

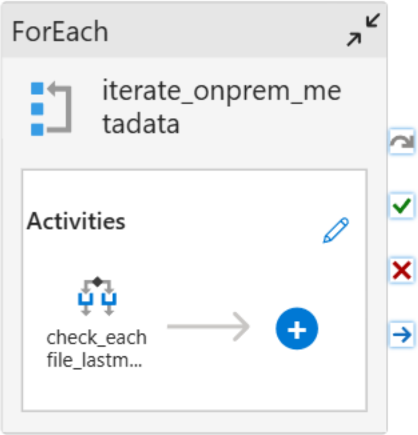
Wait on completion ☑

Now, the whole implementation of master pipeline is completed.

Bronze Layer Pipeline

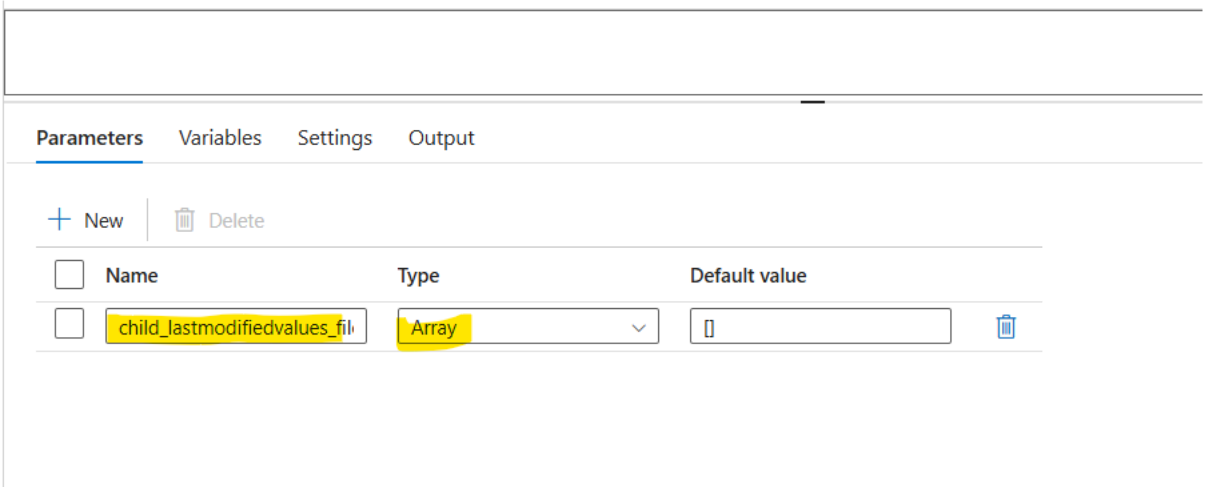
Here, I have created a child Pipeline to implement the **Broze layer**. In the Bronze, I have used **1 Forech activity, 1 if condition activity and 1 copy data activity**.

Using foreach activity, the pipeline will iterate over all the files present in on prem data folder, then check for the last modifies date for the files if the last modified date of the file fall in last 24 hours, then the file will be copied to Bronze_Layer folder in ADLS gen2 storage.



➢ Steps To Create the Pipeline are as follows.

Step 1: Create new Pipeline and name it as *pl_bronze_layer.* Create a new parameter of type array and name it as *child_lastmodifiedvalues_filenames*.

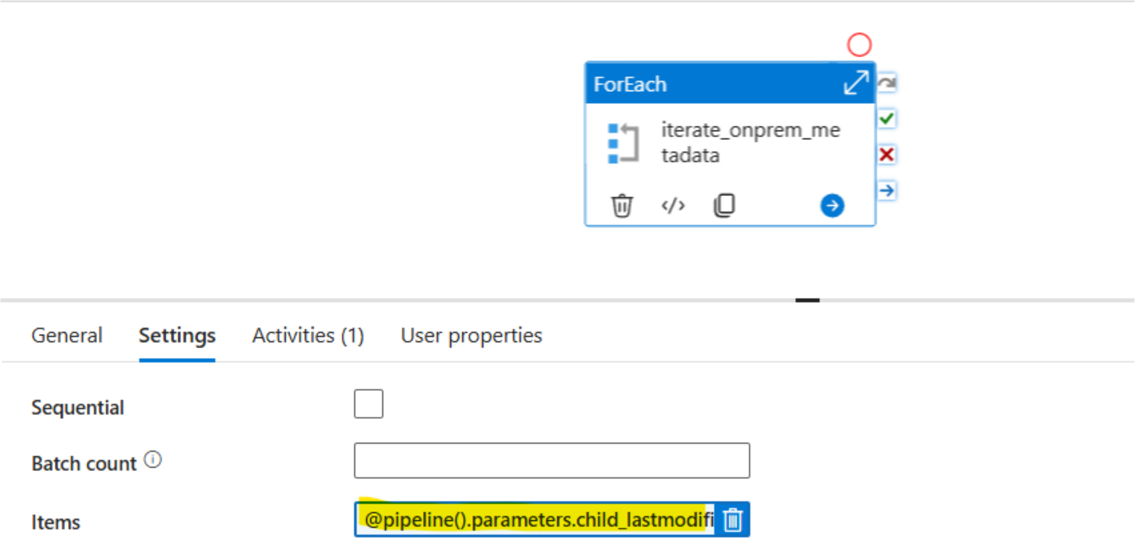


	Name	Type	Default value
□	child_lastmodifiedvalues_fil	Array	[]

Step 2: Drag a foreach activity and then go to settings tab and add the below expression in the items field.

– Expression Used:
  - **@pipeline().parameters.child_lastmodifiedvalues_filenames**

The above expression is used to get the data from its parent pipeline (we will use it to get the file names with last modifies time stamps).



Sequential □

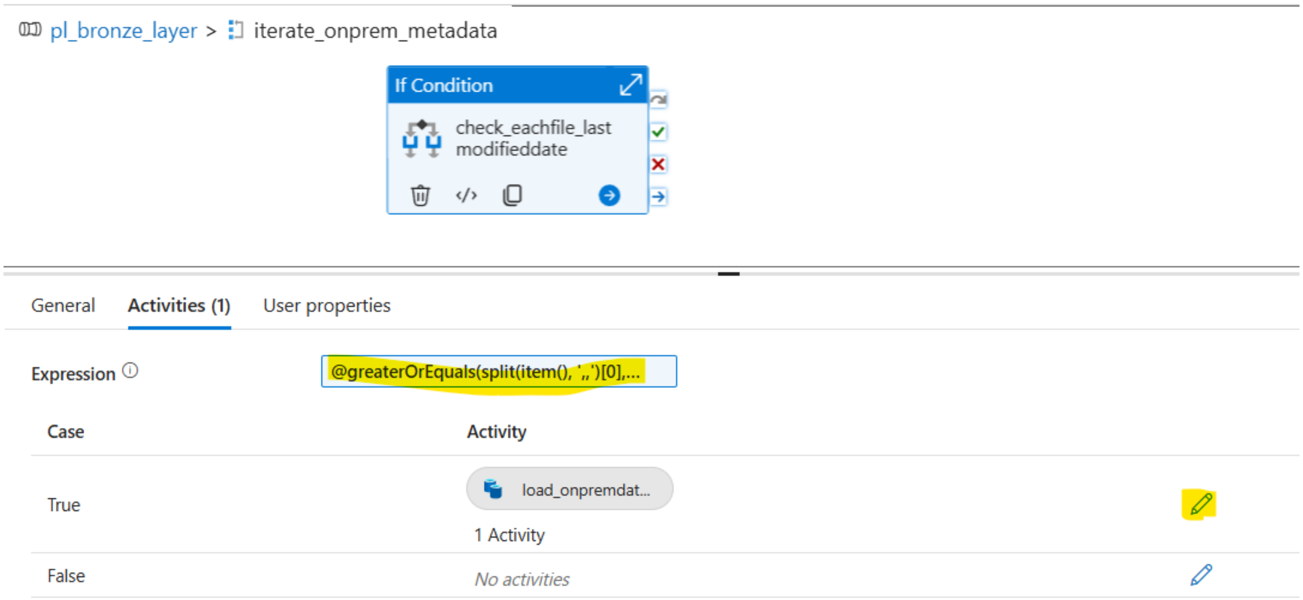
Batch count

Items @pipeline().parameters.child_lastmodifi

Step 3: Next, go to activities tab of foreach -> click on pencil to add activity in it -> add an if condition activity.

– In the if condition, go to activities tab -> add the below Expression.
– Expression Used:
  - **@greaterOrEquals(split(item(), ',,')[0], addDays(utcNow(), -1))**

Here, the Above expression check if the first part of a string (split by ',,') is **greater than or equal to yesterday's date** (in UTC) (as in every iteration we are getting a concatenated value of last modified and the respective file name with ,, separator).



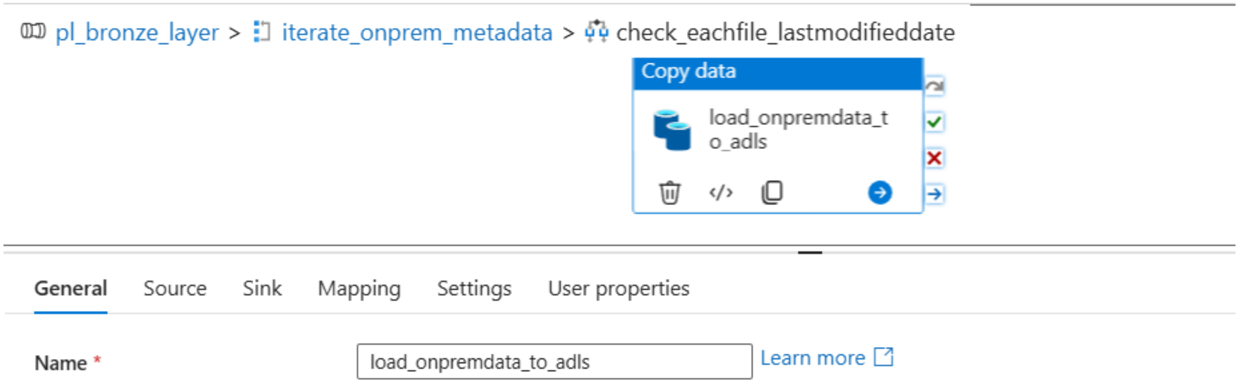
pl_bronze_layer > iterate_onprem_metadata

General | Activities (1) | User properties

Expression: @greaterOrEquals(split(item(), ',,')[0],...

Case	Activity
True	load_onpremdat... 1 Activity
False	No activities

Step 4: Next, go to activities tab of if condition -> click on true section pencil icon to add activity in it -> add a copy data activity in the true section.



pl_bronze_layer > iterate_onprem_metadata > check_eachfile_lastmodifieddate

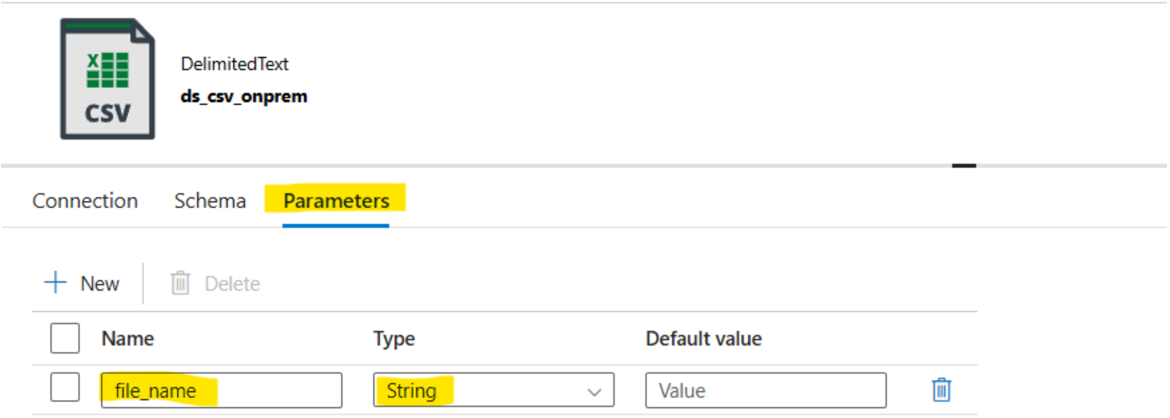
General | Source | Sink | Mapping | Settings | User properties

Name *: load_onpremdata_to_adls

Step 5: Next, go to Source tab of Copy data activity -> new source dataset -> select delimitedText and file server options -> create a parameter and name it as *file_name*.

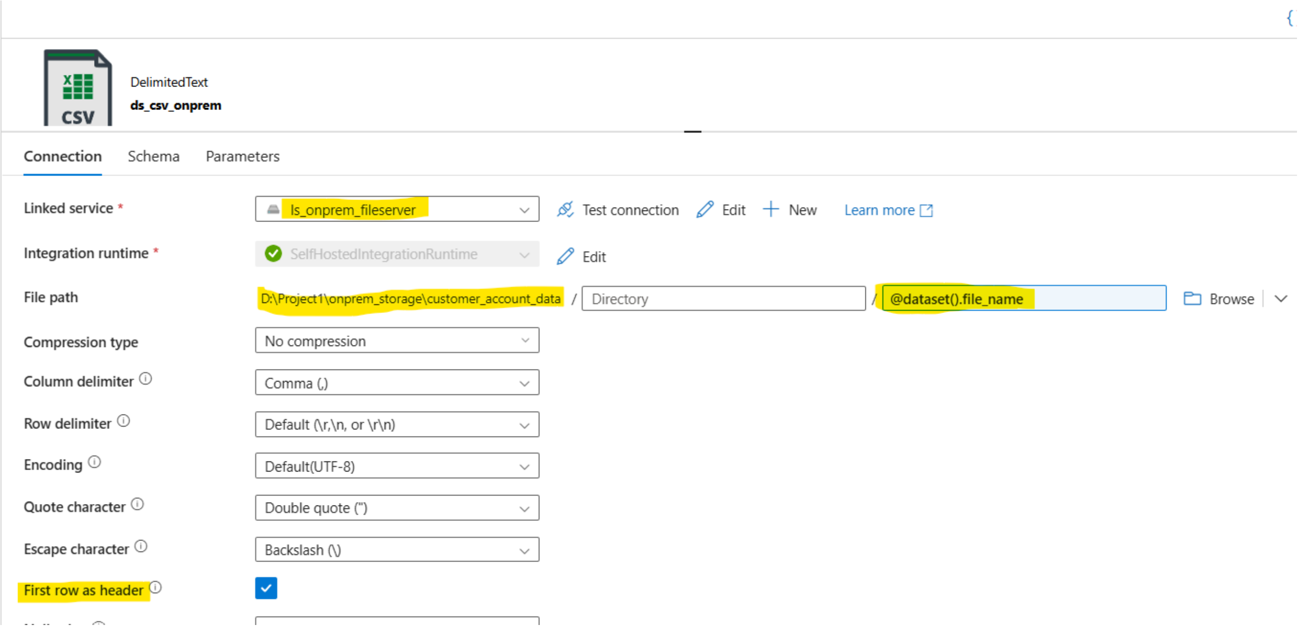
- Expression used to select the file_name data set properties:
  - **@split(item(), ',,')[1]**

  Here, the Above expression select the second part of a string (split by ',,') i.e. file name.



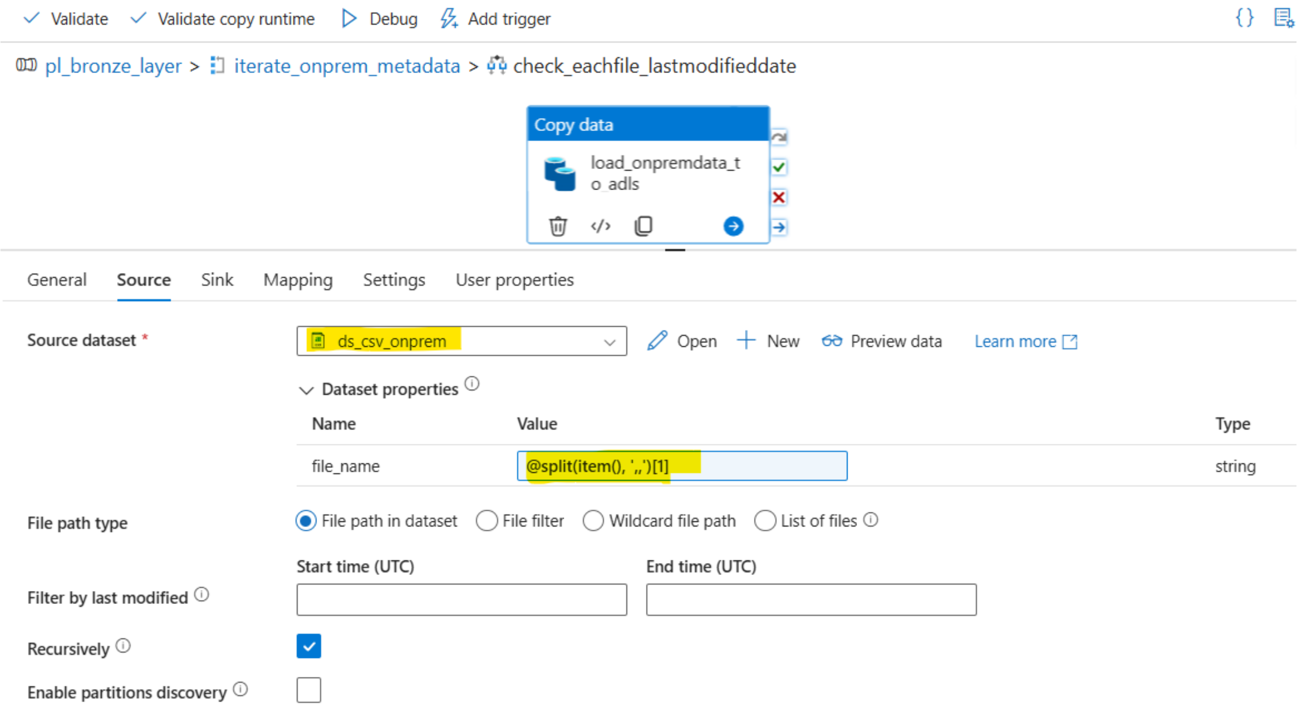
DelimitedText
ds_csv_onprem

☐	Name	Type	Default value
☐	file_name	String	Value



DelimitedText
ds_csv_onprem

Linked service *	ls_onprem_fileserver
Integration runtime *	SelfHostedIntegrationRuntime
File path	D:\Project1\onprem_storage\customer_account_data / Directory / @dataset().file_name
Compression type	No compression
Column delimiter	Comma (,)
Row delimiter	Default (\r,\n, or \r\n)
Encoding	Default(UTF-8)
Quote character	Double quote (")
Escape character	Backslash (\)
First row as header	☑



pl_bronze_layer > iterate_onprem_metadata > check_eachfile_lastmodifieddate

Copy data: load_onpremdata_to adls

General | **Source** | Sink | Mapping | Settings | User properties

Source dataset *: ds_csv_onprem

Dataset properties

Name	Value	Type
file_name	@split(item(), ',,')[1]	string

File path type: ● File path in dataset ○ File filter ○ Wildcard file path ○ List of files

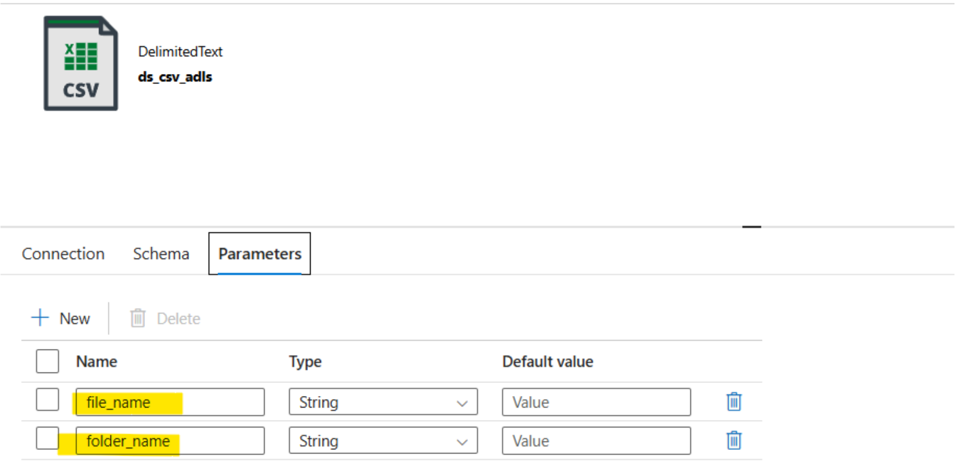
Filter by last modified: Start time (UTC) End time (UTC)

Recursively ☑

Enable partitions discovery □

Step 6: Next, go to Sink tab of Copy data activity -> new sink dataset -> select delimitedText and ADLS gen2 options -> create 2 parameters and name it as *file_name* and *folder_name*.

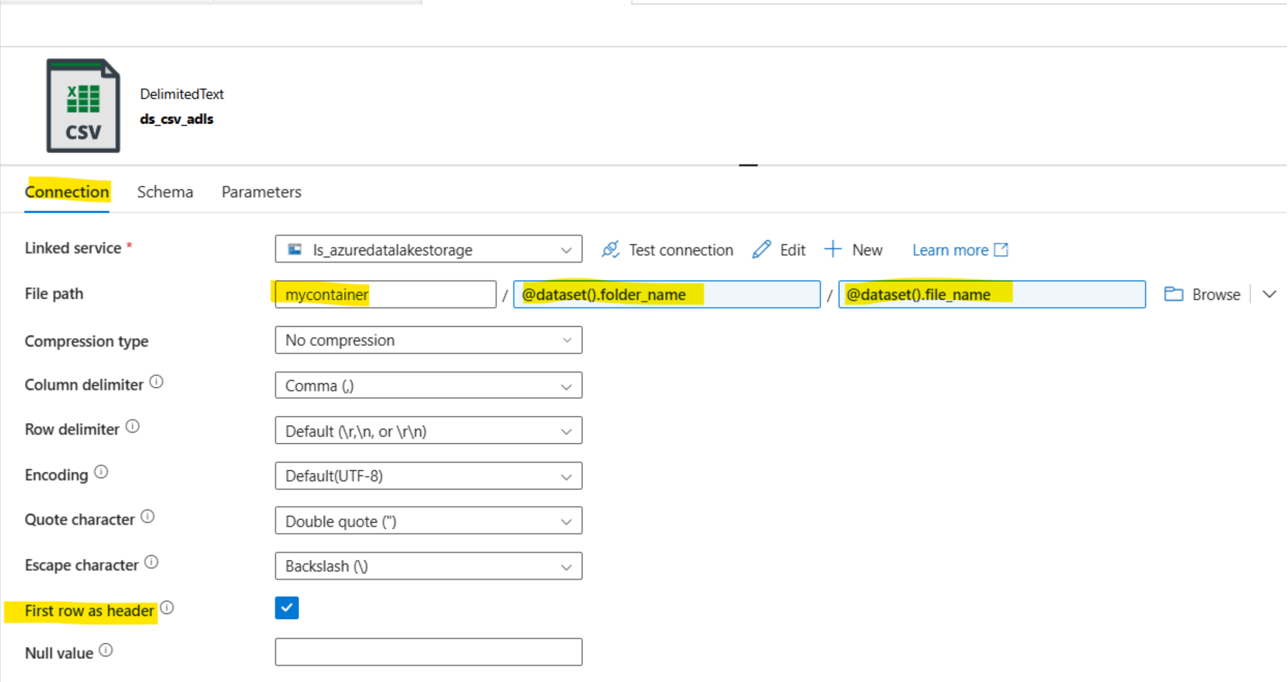
- Expression Used:
- file_name:
  - **@split(item(), ',,')[1]**
- folder_name:
  - **@concat('Bronze_Layer/', replace(string(split(item(), ',,')[1]) ,'.csv',''))**
- Set file extension as .csv



DelimitedText
ds_csv_adls

Connection | Schema | **Parameters**

Name	Type	Default value
file_name	String	Value
folder_name	String	Value


DelimitedText
ds_csv_adls

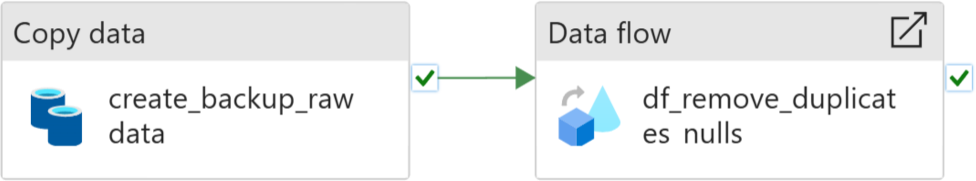
Linked service *	ls_azuredatalakestorage
File path	mycontainer / @dataset().folder_name / @dataset().file_name
Compression type	No compression
Column delimiter	Comma (,)
Row delimiter	Default (\r,\n, or \r\n)
Encoding	Default(UTF-8)
Quote character	Double quote (")
Escape character	Backslash (\)
First row as header	☑
Null value	

Now, the process of Raw data load to Bronze_Layer folder in ADLS is finished.

Silver Layer Pipeline

Here, I have created a child Pipeline to implement the **Silver layer**. In the **Silver**, I have used **1 Copy data** to create a date-timestamp wise backup of Raw data (Bronze layer data) in the ADLS and **1 Data flow activity** to load the data from **Bronze Layer Folder** of ADLS Gen2 storage to **Silver Layer Folder** of ADLS Gen2 storage by **performing data transformation** such as **removing nulls and duplicate records, replacing null values with meaning full values and performing some filter operations**.

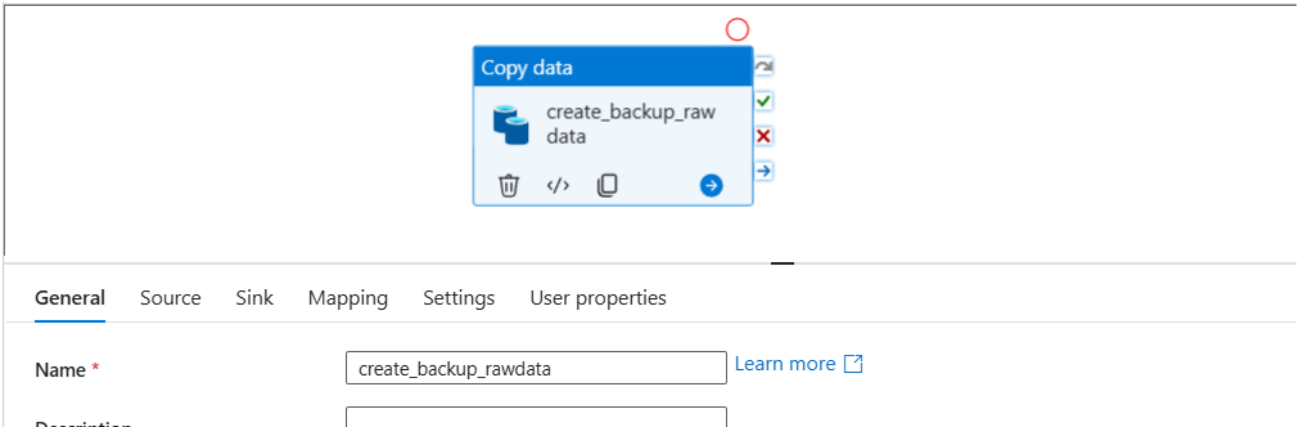
Here, I have used all 5 tables as source in only one data flow activity to perform transformation operations.



- **Steps To Create the Pipeline are as follows.**

- ✓ **For Creating Backup Raw data (Copy data) Activity to store the Backup of Raw data.**

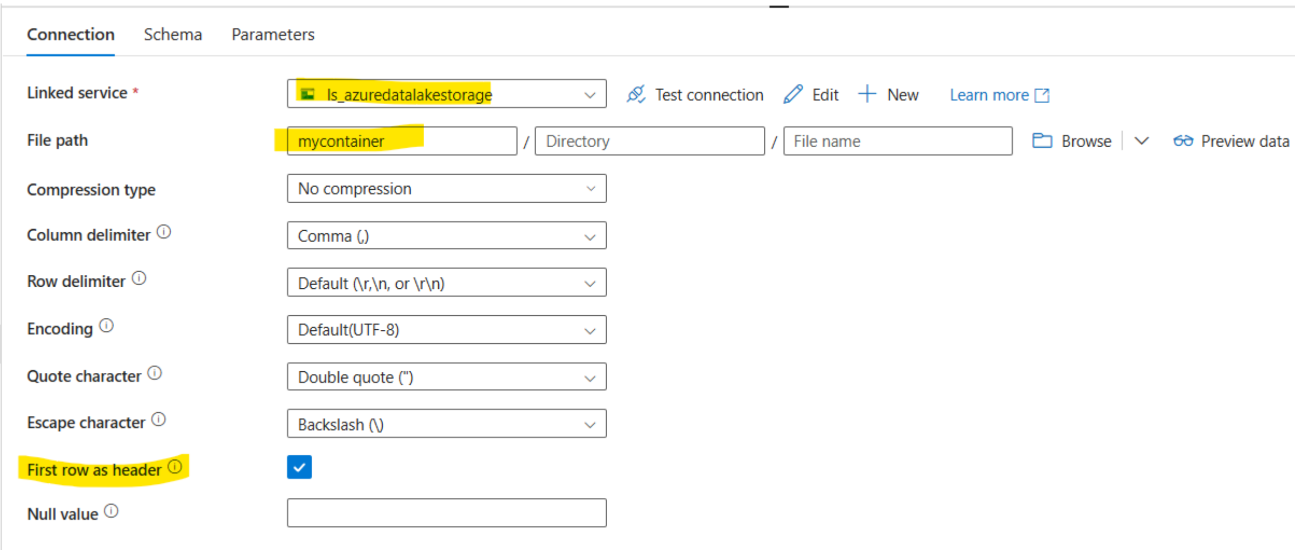
  Step 1: Create new Pipeline and name it as *pl_silver_layer.* Drag a new coy data activity and name it as *create_backup_rawdata.*



Name * create_backup_rawdata

Description

Step 2: Go to source -> new source dataset -> select delimitedText and ADLS gen2 storage.



Linked service * ls_azuredatalakestorage

File path mycontainer / Directory / File name

Compression type No compression

Column delimiter Comma (,)

Row delimiter Default (\r,\n, or \r\n)

Encoding Default(UTF-8)

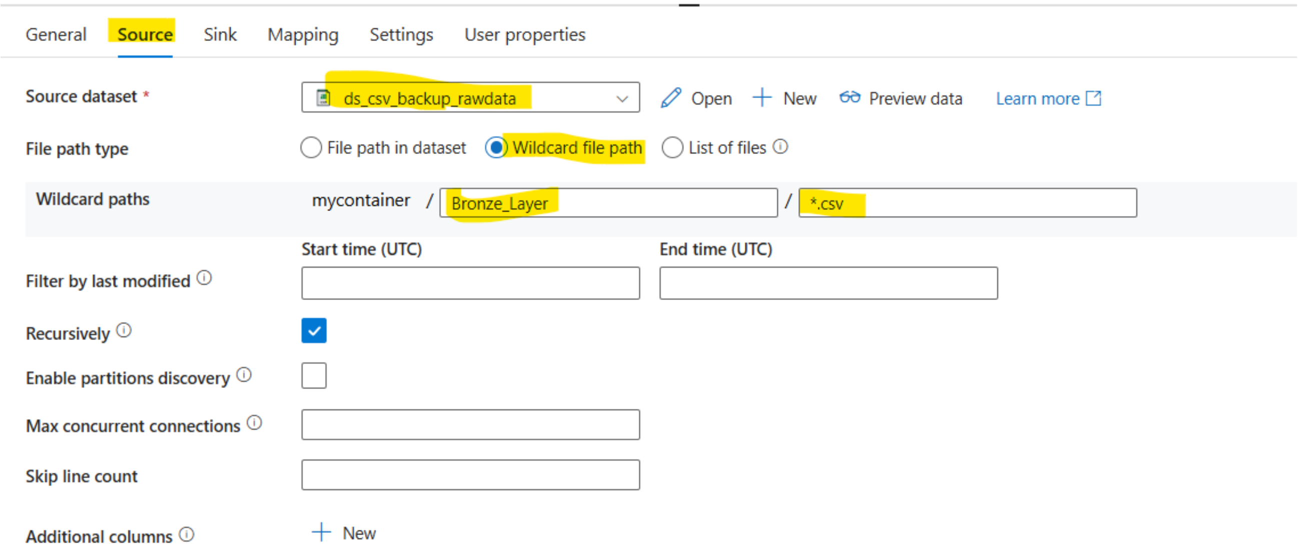
Quote character Double quote (")

Escape character Backslash (\)

First row as header ☑

Null value

- Move to source tab -> select new dataset -> select file path option as Wildcard file path -> select all the csv file using ***.csv** to select all the csv file in Bronze_Layer folder.



Source dataset * ds_csv_backup_rawdata

File path type ○ File path in dataset ● Wildcard file path ○ List of files

Wildcard paths mycontainer / Bronze_Layer / *.csv

Filter by last modified Start time (UTC) End time (UTC)

Recursively ☑

Enable partitions discovery □

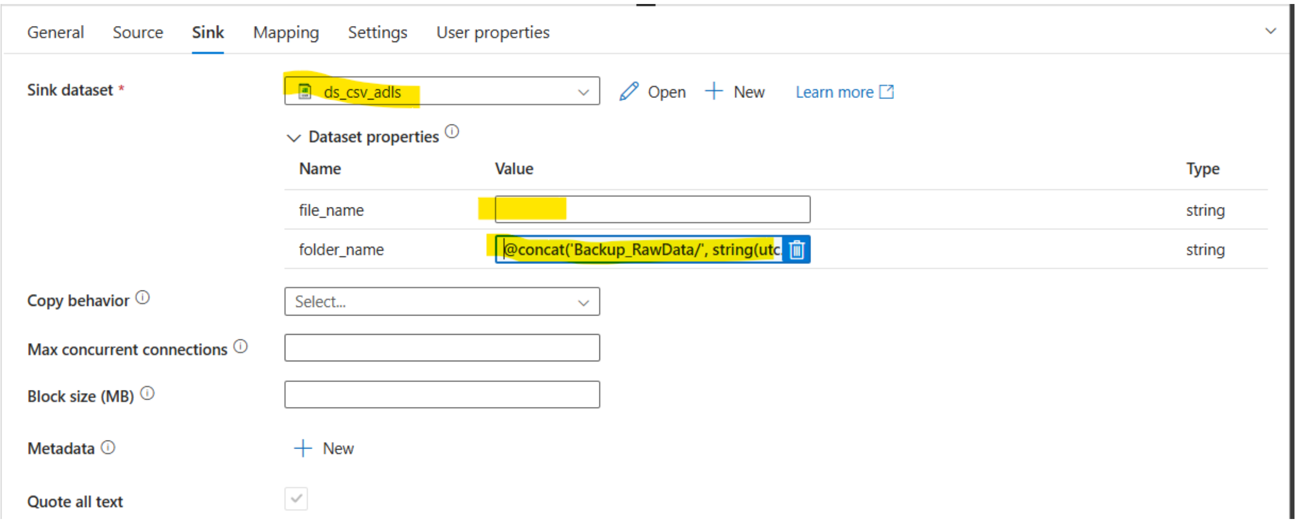
Max concurrent connections

Skip line count

Additional columns

Step 3: Go to sink -> select previously created dataset to store the files in ADLS gen2 storage -> pass the filename and foldername.

- Here, I have passed the **white space** in the file_name parameter.
- Below expression in the folder_name parameter to create a Backup Raw data folder and inside it a current time stamp folder.
  - **@concat('Backup_RawData/', string(utcNow()), '/')**
- Set file extension as .csv



Sink dataset * ds_csv_adls

Dataset properties

Name	Value	Type
file_name		string
folder_name	@concat('Backup_RawData/', string(utc	string

Copy behavior: Select...

Max concurrent connections:

Block size (MB):

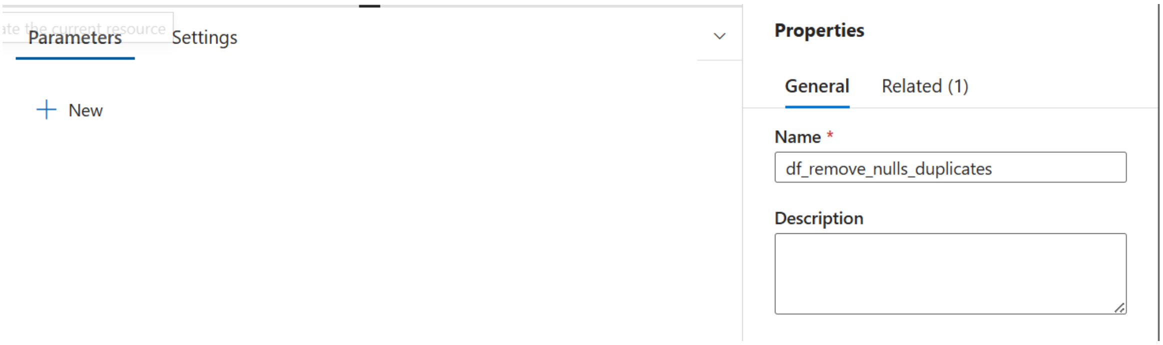
Metadata: + New

Quote all text: ☐

- ✓ **For data cleaning and transformation (accounts file data).**

Step 1: Drag a new **Data Flow Activity**. Connect it with Bakup Copy data activity on success point.

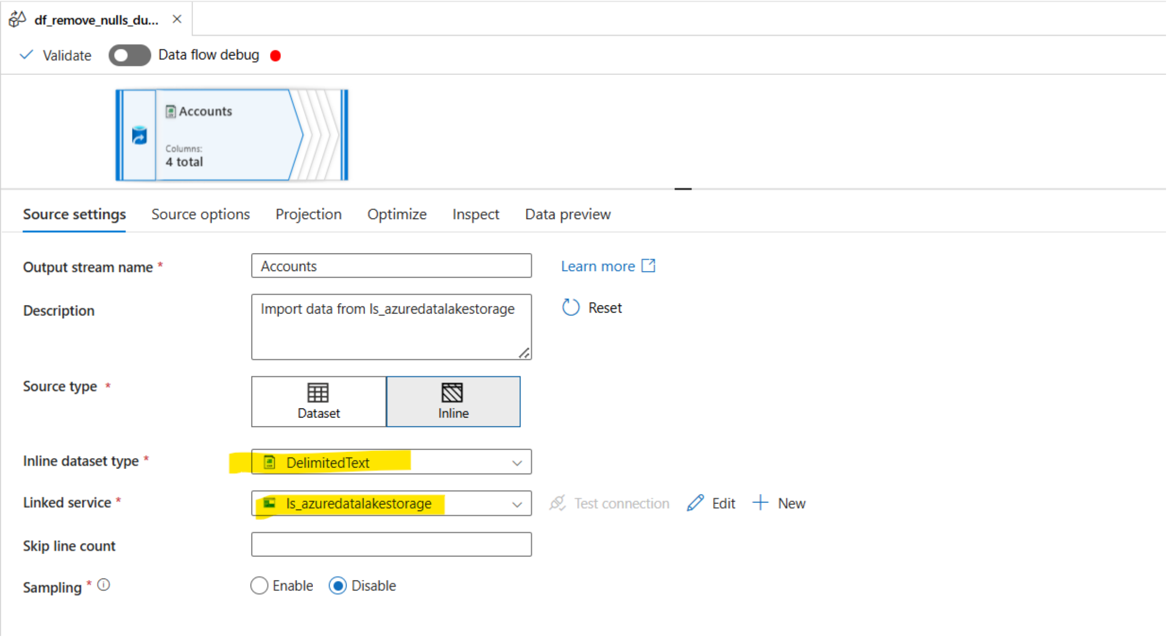
– Here, I have created a dataflow with name *df_remove_nulls_duplicates*.



Name * df_remove_nulls_duplicates

Description

Step 2: Click on Add source -> Click on Source Type -> Inline and select the source.



Output stream name * Accounts

Description: Import data from ls_azuredatalakestorage

Source type *: Dataset / Inline

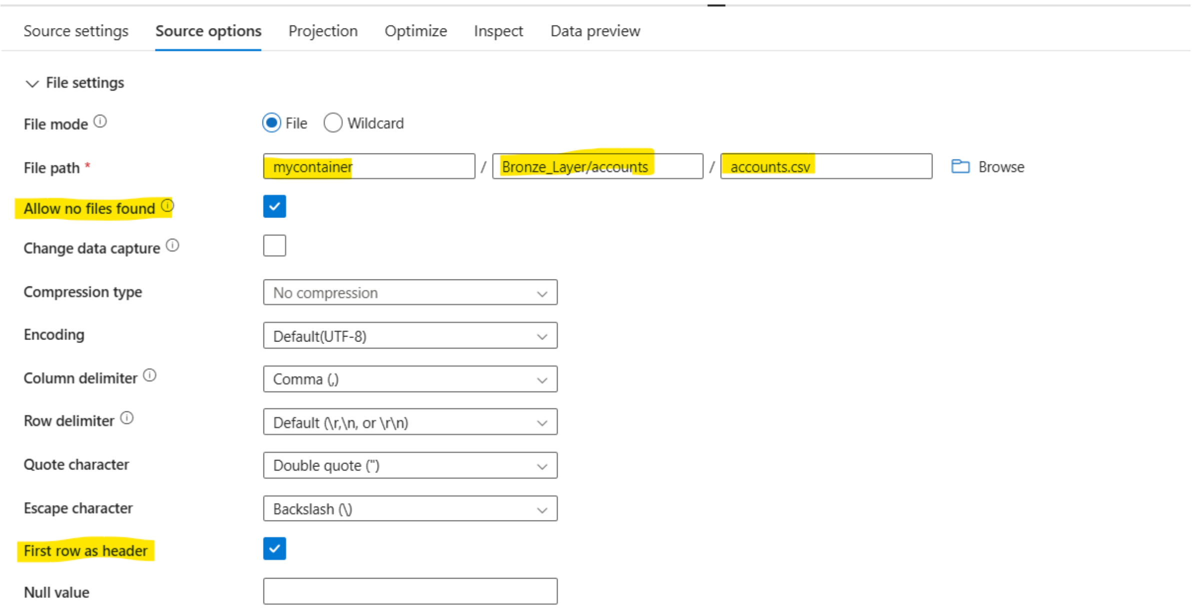
Inline dataset type *: DelimitedText

Linked service *: ls_azuredatalakestorage

Skip line count:

Sampling *: ○ Enable ● Disable

- Next, in the Source option, select the file path.
- And **checked Allow no files found option** so that if no file is found then it will run on blank file.
- And checked First row as header option.



File mode: ● File ○ Wildcard

File path *: mycontainer / Bronze_Layer/accounts / accounts.csv

Allow no files found: ☑

Change data capture: □

Compression type: No compression

Encoding: Default(UTF-8)

Column delimiter: Comma (,)

Row delimiter: Default (\r,\n, or \r\n)

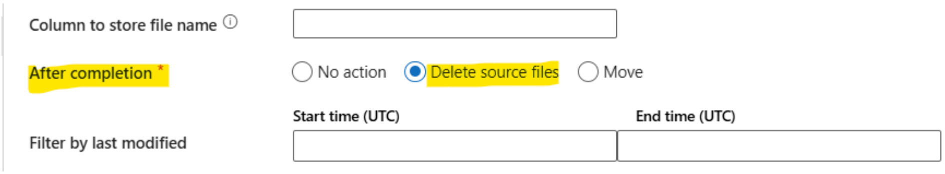
Quote character: Double quote (")

Escape character: Backslash (\)

First row as header: ☑

Null value:

- Next, I want to delete the source file after execution so select **Delete source files** option below.

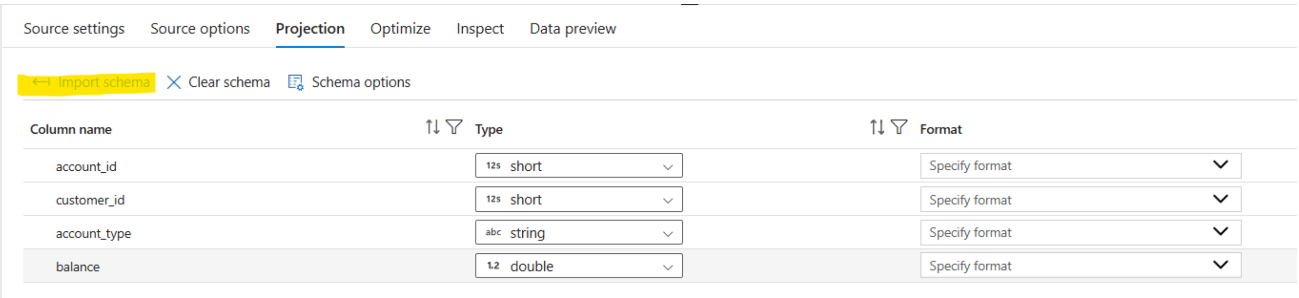


Column to store file name ⓘ

After completion * ○ No action ● Delete source files ○ Move

Filter by last modified — Start time (UTC): — End time (UTC):

Step 3: Click on Projection tab and import schema and check for Data preview.



Column name	Type	Format
account_id	12s short	Specify format
customer_id	12s short	Specify format
account_type	abc string	Specify format
balance	1.2 double	Specify format

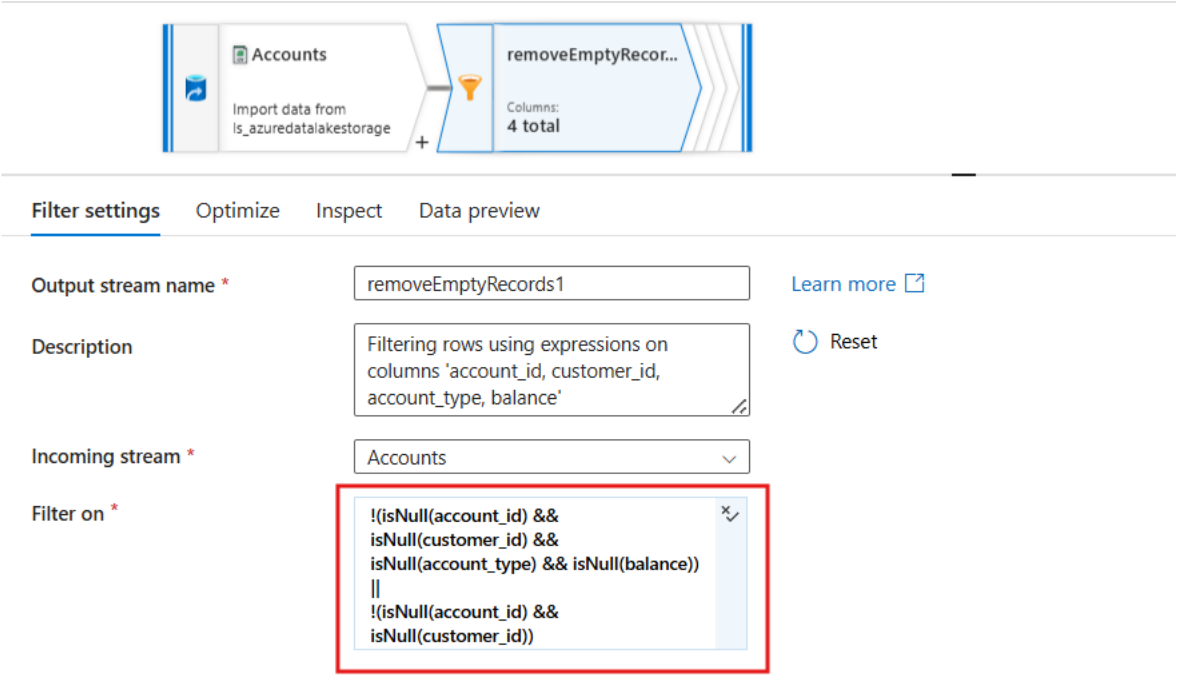
Step 4: Add the filter transformation to remove all null records.

- Expression Used:

  !(isNull(account_id) && isNull(customer_id) && isNull(account_type) && isNull(balance))

  ||

  !(isNull(account_id) && isNull(customer_id))



Output stream name *: removeEmptyRecords1

Description: Filtering rows using expressions on columns 'account_id, customer_id, account_type, balance'

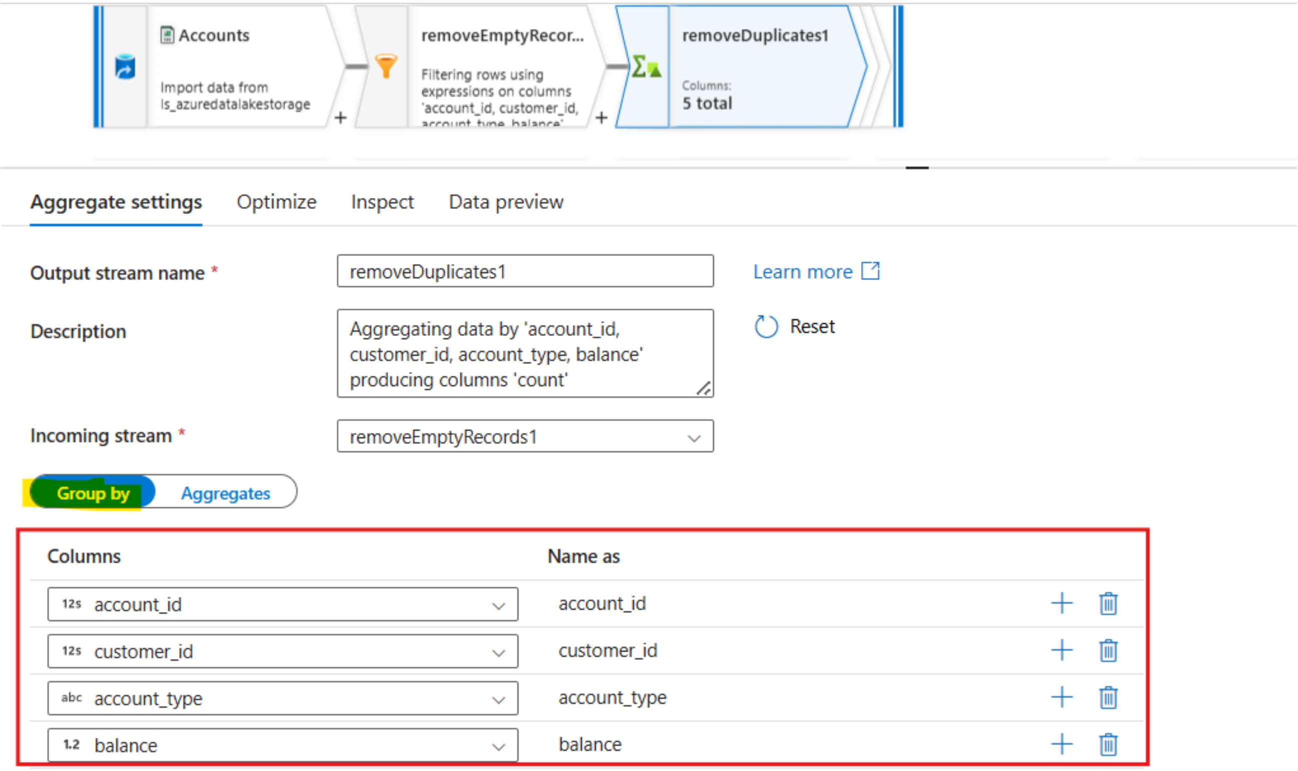
Incoming stream *: Accounts

Filter on *:

```
!(isNull(account_id) &&
isNull(customer_id) &&
isNull(account_type) && isNull(balance))
||
!(isNull(account_id) &&
isNull(customer_id))
```

Step 5: Add the Aggregate transformation to remove all duplicates records.

– Here, In Group by Option section, I have selected all column to remove duplicates based on all column match value.



Accounts
Import data from ls_azuredatalakestorage
removeEmptyRecor...
Filtering rows using expressions on columns 'account_id, customer_id, account_type, balance'
removeDuplicates1
Columns:
5 total

Aggregate settings | Optimize | Inspect | Data preview

Output stream name *: removeDuplicates1

Learn more

Description: Aggregating data by 'account_id, customer_id, account_type, balance' producing columns 'count'

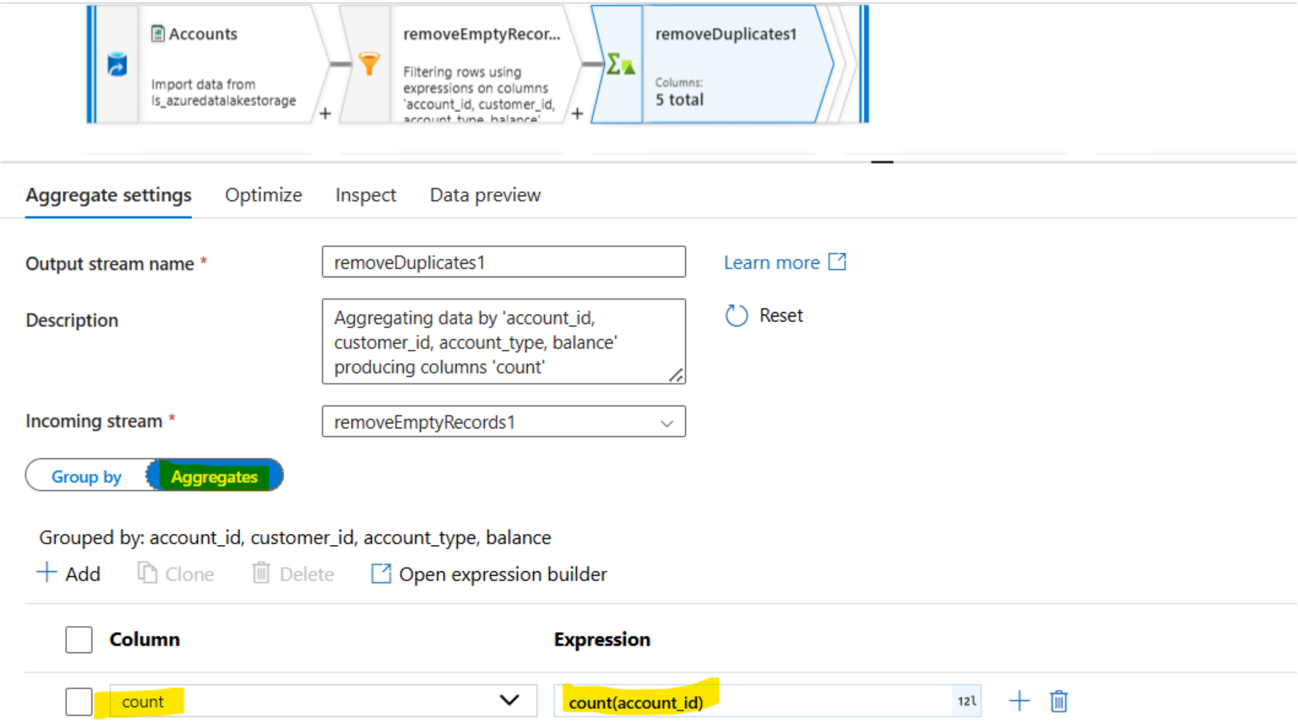
Reset

Incoming stream *: removeEmptyRecords1

Group by | Aggregates

Columns	Name as
12s account_id	account_id
12s customer_id	customer_id
abc account_type	account_type
1.2 balance	balance

– In Aggregate Option section, I have added a dummy column name count to perform the count aggregate function and using that function aggregating the unique records.



Aggregate settings | Optimize | Inspect | Data preview

Output stream name * removeDuplicates1 — Learn more

Description: Aggregating data by 'account_id, customer_id, account_type, balance' producing columns 'count' — Reset

Incoming stream * removeEmptyRecords1

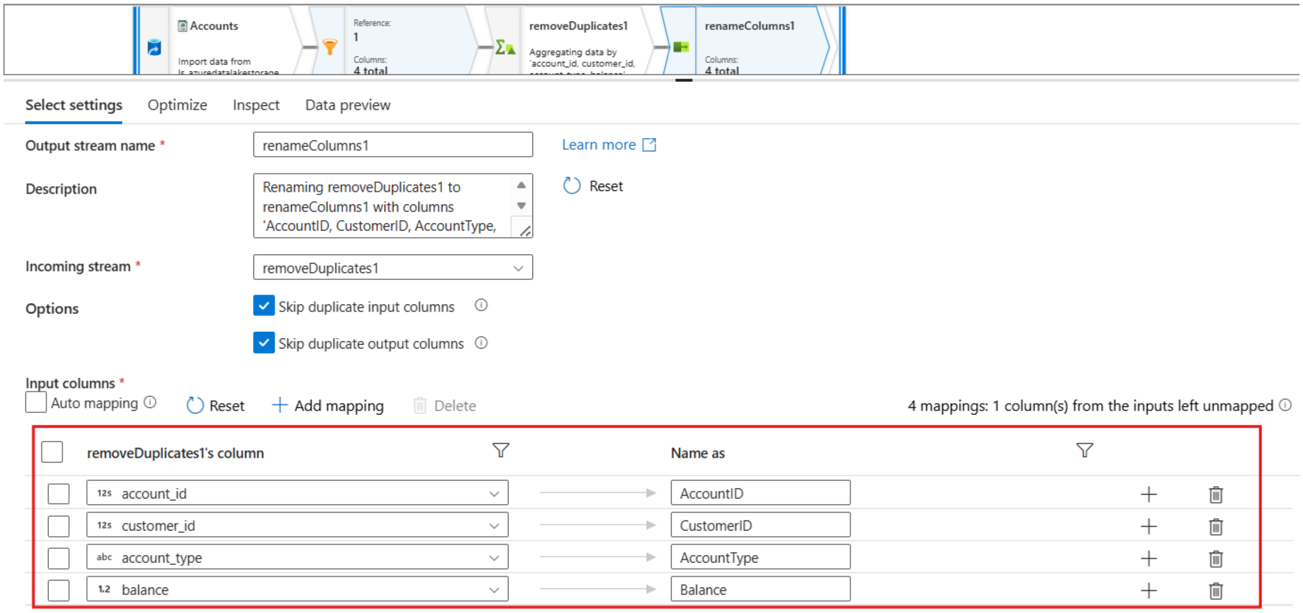
Group by | **Aggregates**

Grouped by: account_id, customer_id, account_type, balance

+ Add | Clone | Delete | Open expression builder

	Column	Expression
□	count	count(account_id)

Step 6: Add the Select transformation to rename the Columns and remove dummy column which is created in last transformation.



Select settings | Optimize | Inspect | Data preview

Output stream name * renameColumns1 — Learn more

Description: Renaming removeDuplicates1 to renameColumns1 with columns 'AccountID, CustomerID, AccountType, — Reset

Incoming stream * removeDuplicates1

Options: ☑ Skip duplicate input columns
☑ Skip duplicate output columns

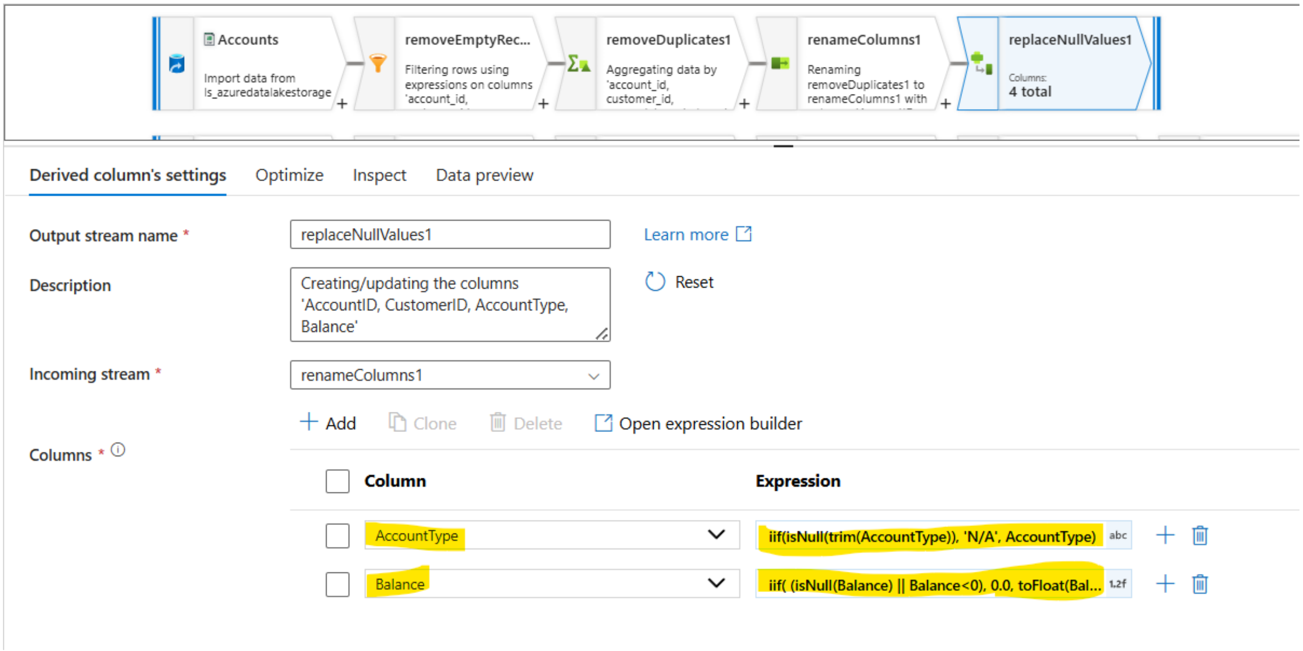
Input columns *
□ Auto mapping | Reset | + Add mapping | Delete — 4 mappings: 1 column(s) from the inputs left unmapped

	removeDuplicates1's column	Name as
□	12s account_id	AccountID
□	12s customer_id	CustomerID
□	abc account_type	AccountType
□	1.2 balance	Balance

Step 7: Add the derived column transformation to replace the null value in the column with a meaningful value.

Expression used:

- **iif(isNull(trim(AccountType)), 'N/A', AccountType)**
- **iif((isNull(Balance) || Balance<0), 0.0, toFloat(Balance))**



Accounts
Import data from ls_azuredatalakestorage

removeEmptyRec...
Filtering rows using expressions on columns 'account_id,

removeDuplicates1
Aggregating data by 'account_id, customer_id,

renameColumns1
Renaming removeDuplicates1 to renameColumns1 with

replaceNullValues1
Columns:
4 total

Derived column's settings | Optimize | Inspect | Data preview

Output stream name * : replaceNullValues1 — Learn more

Description: Creating/updating the columns 'AccountID, CustomerID, AccountType, Balance' — Reset

Incoming stream * : renameColumns1

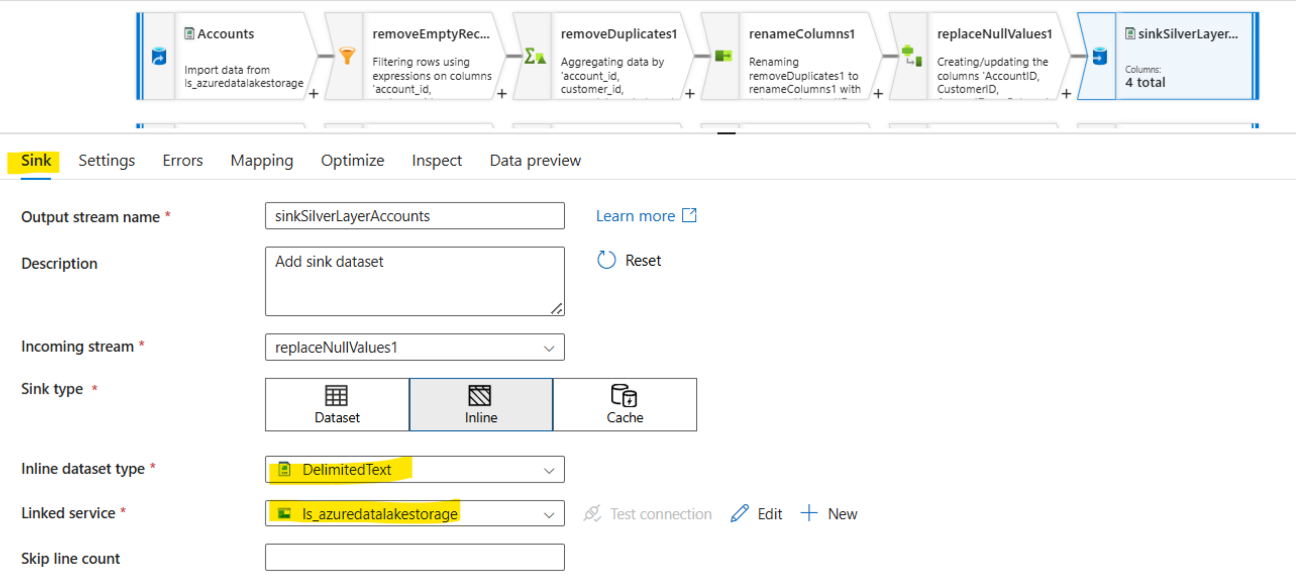
Columns * ⓘ

+ Add | Clone | Delete | Open expression builder

□	Column	Expression
□	AccountType	iif(isNull(trim(AccountType)), 'N/A', AccountType) abc
□	Balance	iif((isNull(Balance) || Balance<0), 0.0, toFloat(Bal... 1.2f

Step 8: Create sink, select ADLS gen2 Storage for delimitedText.

– Select the location and folder into Silver Layer to load the transformed data.



Accounts
Import data from ls_azuredatalakestorage

removeEmptyRec...
Filtering rows using expressions on columns 'account_id,

removeDuplicates1
Aggregating data by 'account_id, customer_id,

renameColumns1
Renaming removeDuplicates1 to renameColumns1 with

replaceNullValues1
Creating/updating the columns 'AccountID, CustomerID,

sinkSilverLayer...
Columns:
4 total

Sink | Settings | Errors | Mapping | Optimize | Inspect | Data preview

Output stream name * : sinkSilverLayerAccounts — Learn more

Description: Add sink dataset — Reset

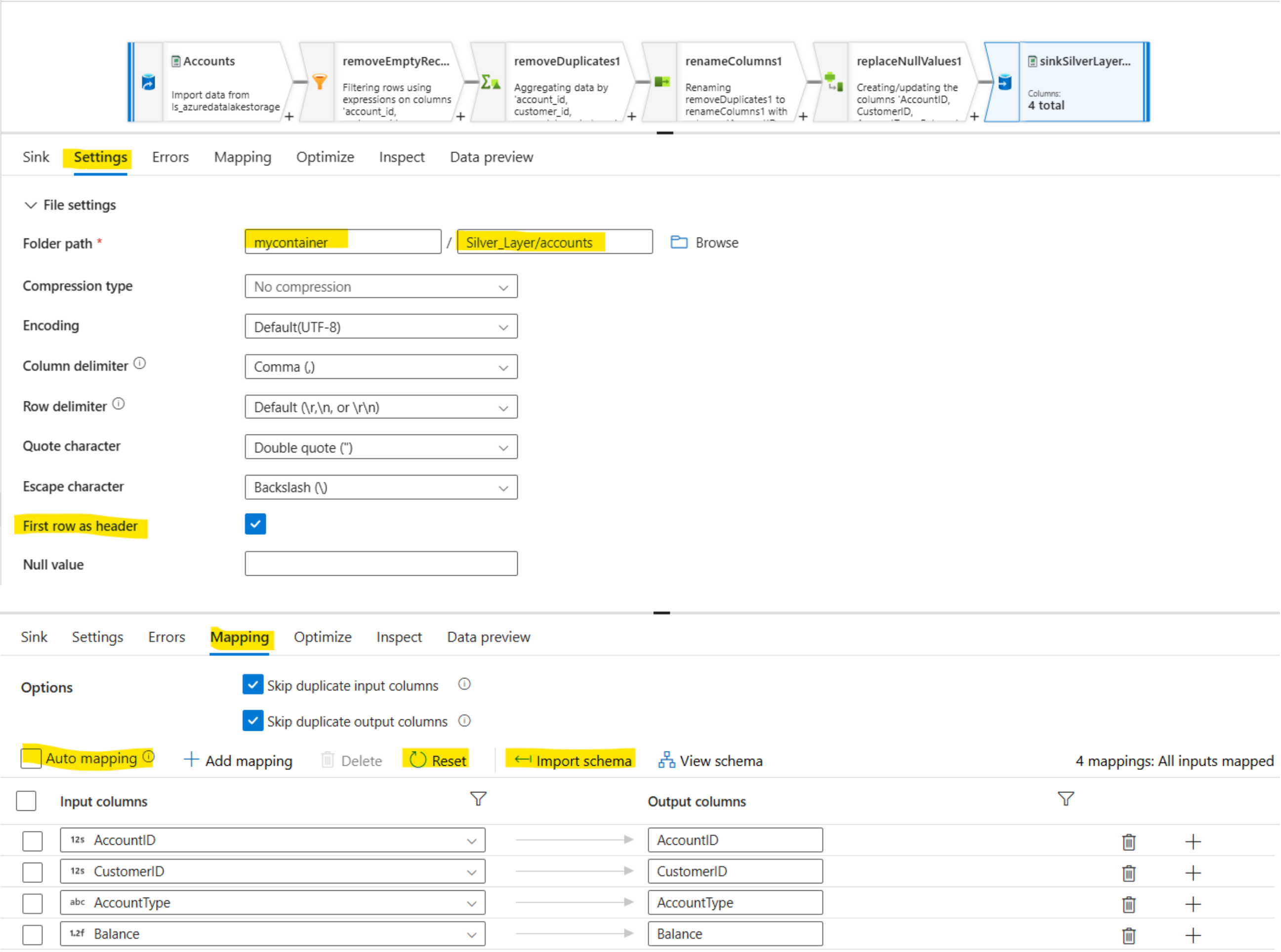
Incoming stream * : replaceNullValues1

Sink type * : Dataset | Inline | Cache

Inline dataset type * : DelimitedText

Linked service * : ls_azuredatalakestorage — Test connection | Edit | + New

Skip line count:


Accounts
Import data from ls_azuredatalakestorage
removeEmptyRec...
Filtering rows using expressions on columns 'account_id,
removeDuplicates1
Aggregating data by 'account_id, customer_id,
renameColumns1
Renaming removeDuplicates1 to renameColumns1 with
replaceNullValues1
Creating/updating the columns 'AccountID, CustomerID,
sinkSilverLayer...
Columns:
4 total
Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Browse

File settings

Folder path * mycontainer / Silver_Layer/accounts

Compression type: No compression

Encoding: Default(UTF-8)

Column delimiter: Comma (,)

Row delimiter: Default (\r,\n, or \r\n)

Quote character: Double quote (")

Escape character: Backslash (\)

First row as header: ☑

Null value:

Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview

Options: ☑ Skip duplicate input columns

☑ Skip duplicate output columns

□ Auto mapping   + Add mapping   Delete   Reset   Import schema   View schema   4 mappings: All inputs mapped

□	Input columns		Output columns
□	12s AccountID	→	AccountID
□	12s CustomerID	→	CustomerID
□	abc AccountType	→	AccountType
□	1.2f Balance	→	Balance

Now, the data transformation is finished for **accounts** Data file.

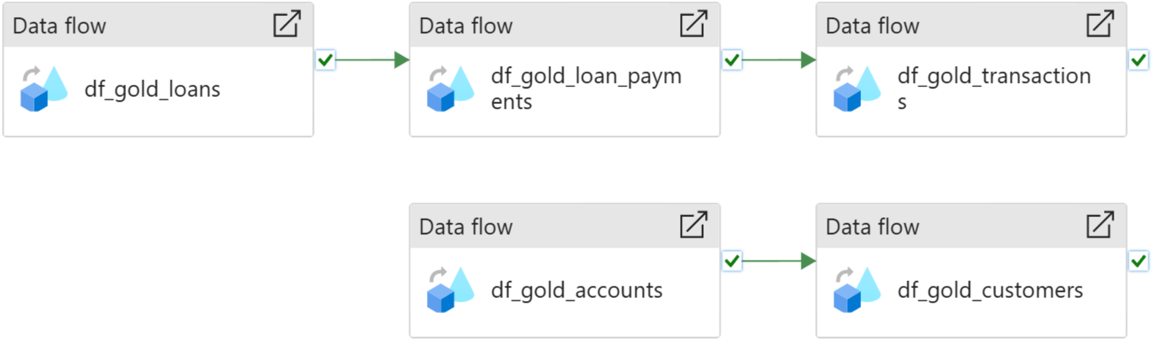
****And similarly, we need to implement the similar transformation for rest of the tables (customers, loan_payments, loans,** and **transactions) to clean the data and store it in Silver layer folder in ADLS Gen2 storage.**

Gold Layer Pipeline

Here, I have created a child Pipeline to implement the **gold layer**. In the Gold, I have **used Five Data flow activities** to load the data from Silver Layer Folder of ADLS Gen2 storage to Azure SQL DB tables using SCD Type1 and Type2 approach.

With five Tables, I have implemented the SCD types in the following structure.

- **accounts** and **customers** table with SCD Type2 approach.
- **loan_payments**, **loans**, and **transactions** with SCD Type1 approach



➢ The tables schema definitions are as follows.

accounts:

```
CREATE TABLE accounts (
  AccountId INT,
  CustomerId INT,
  AccountType VARCHAR(50),
  Balance DECIMAL(10, 2),
        CreatedBy varchar(100),
        CreatedDate datetime,
        UpdatedBy varchar(100),
        UpdatedDate datetime,
        Hashkey Bigint,
        isActive Int
);
Select * from accounts
```

loan_payments:

```
CREATE TABLE loan_payments (
   PaymentId INT,
   LoanId INT,
   PaymentDate DATE,
   PaymentAmount DECIMAL(10, 2),
           CreatedBy varchar(100),
           CreatedDate datetime,
           UpdatedBy varchar(100),
           UpdatedDate datetime,
           Hashkey Bigint,
);
Select * from loan_payments
```

loans:

```
CREATE TABLE loans (
   LoanId INT,
   CustomerId INT,
   LoanAmount DECIMAL(10, 2),
   InterestRate DECIMAL(5, 2),
   LoanTerm INT,
   CreatedBy varchar(100),
           CreatedDate datetime,
           UpdatedBy varchar(100),
           UpdatedDate datetime,
           Hashkey Bigint,
);
Select * from loans
```

transactions:

```
CREATE TABLE transactions (
   TransactionId INT,
   AccountId INT,
   TransactionDate DATE,
   TransactionAmount DECIMAL(10, 2),
   TransactionType VARCHAR(50),
           CreatedBy varchar(100),
           CreatedDate datetime,
           UpdatedBy varchar(100),
           UpdatedDate datetime,
           Hashkey Bigint,
);
Select * from transactions
```

customers:

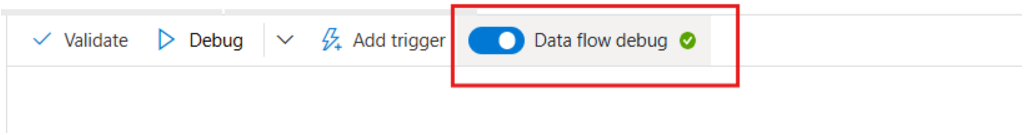
```
CREATE TABLE customers (
   CustomerId INT,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Address VARCHAR(100),
   City VARCHAR(50),
   State VARCHAR(50),
   Zip VARCHAR(10),
          CreatedBy varchar(100),
          CreatedDate datetime,
          UpdatedBy varchar(100),
          UpdatedDate datetime,
          Hashkey Bigint,
          isActive Int
);

Select * from customers
```

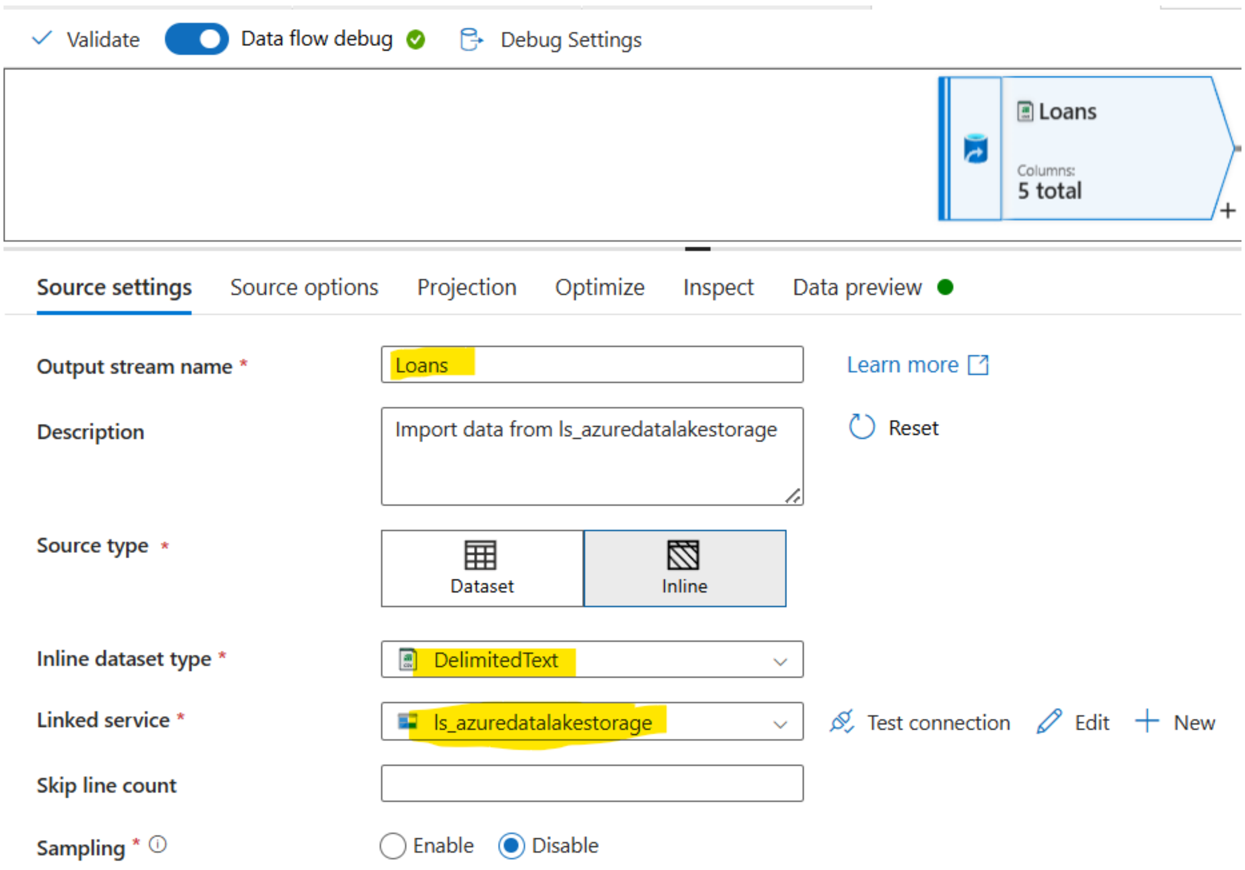
➢ Steps To Create the Data Flow Activities (For SCD Type 1 & 2) in Pipeline are as follows.

✓ **For SCD type1 (loans file data).**

Step 1: Create a new pipeline (name it as *pl_gold_layer)* and drag **Data Flow Activity**. Go to the settings and click on new and **turn on data flow debug.** Here, I have created a dataflow with name *df_loans_scdtype1*.



Step 2: Click on Add source -> Click on Source Type -> Inline and select the source.



Output stream name * Loans

Description: Import data from ls_azuredatalakestorage

Source type * Dataset | Inline

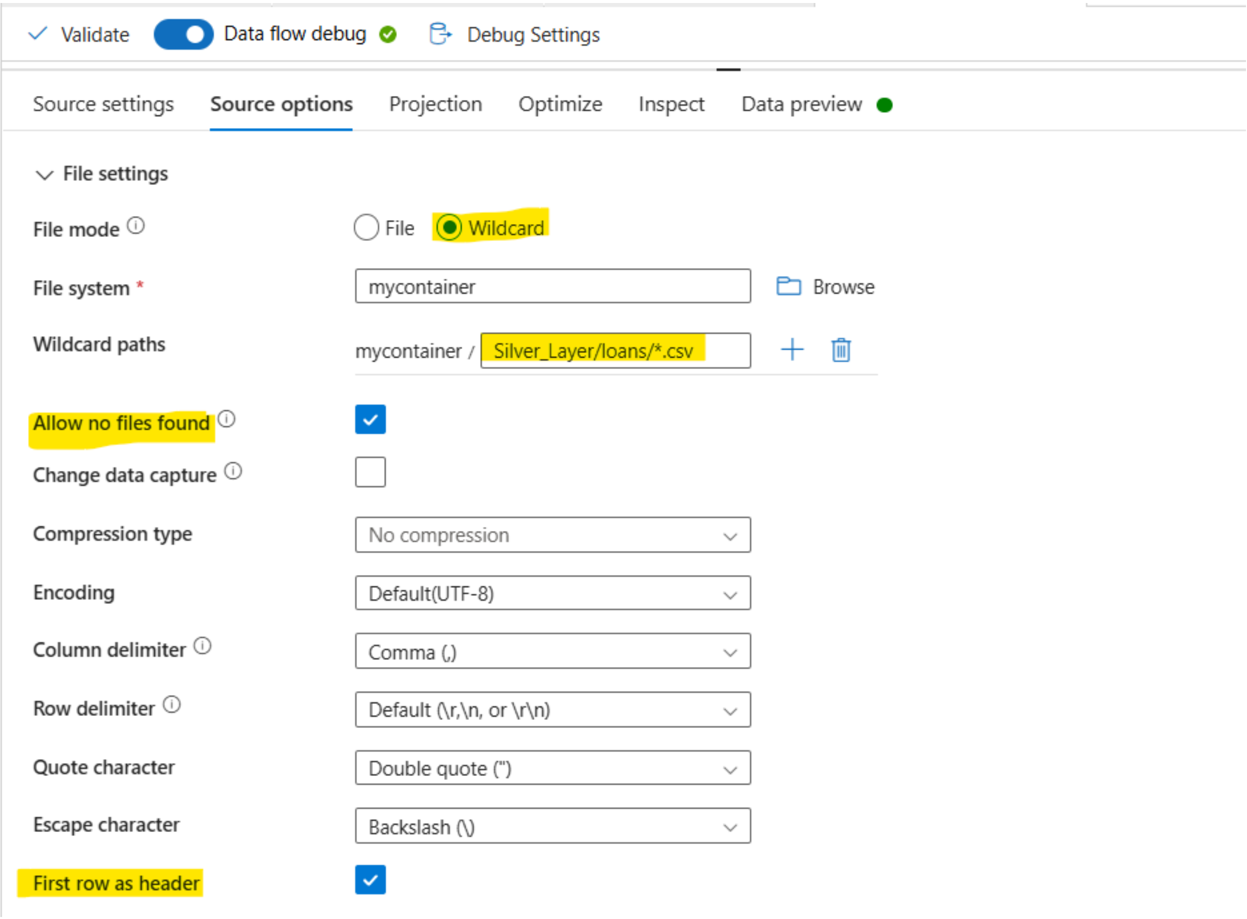
Inline dataset type * DelimitedText

Linked service * ls_azuredatalakestorage

Skip line count

Sampling * ○ Enable ● Disable

- Next, in the Source option, select the file mode as **Wildcard** option as we want to select all the csv files (generated through cluster partition) present in the loans folder in the Silver_Layer Folder using the wildcard character ***.csv (for selecting the all csv file in the loan folder).**
- And **checked Allow no files found option** so that if no file is found then it will run on blank file.
- And checked First row as header option.



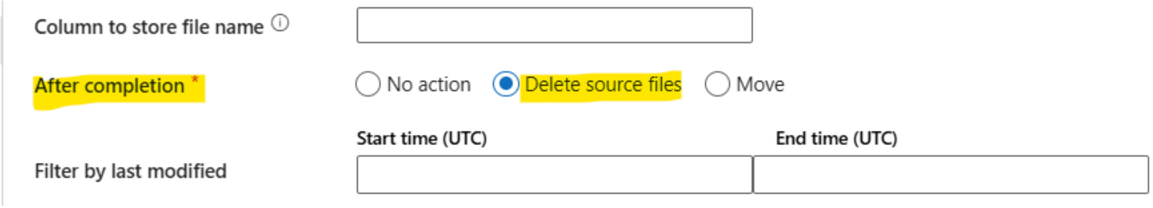
✓ Validate | Data flow debug ● | Debug Settings

Source settings | **Source options** | Projection | Optimize | Inspect | Data preview ●

File settings

File mode	○ File ● Wildcard
File system *	mycontainer
Wildcard paths	mycontainer / Silver_Layer/loans/*.csv
Allow no files found	☑
Change data capture	□
Compression type	No compression
Encoding	Default(UTF-8)
Column delimiter	Comma (,)
Row delimiter	Default (\r,\n, or \r\n)
Quote character	Double quote (")
Escape character	Backslash (\)
First row as header	☑

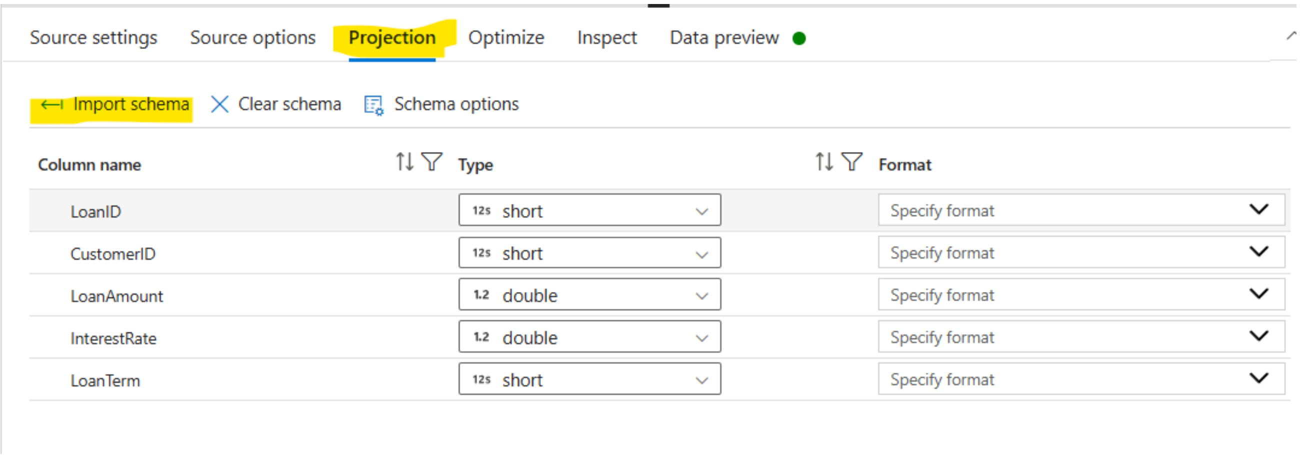
– Next, I want to delete the source file after execution so select **Delete source files** option below.



Column to store file name	
After completion *	○ No action ● Delete source files ○ Move

Filter by last modified	Start time (UTC):
	 End time (UTC):

Step 3: Click on Projection tab and import schema and check for Data preview.

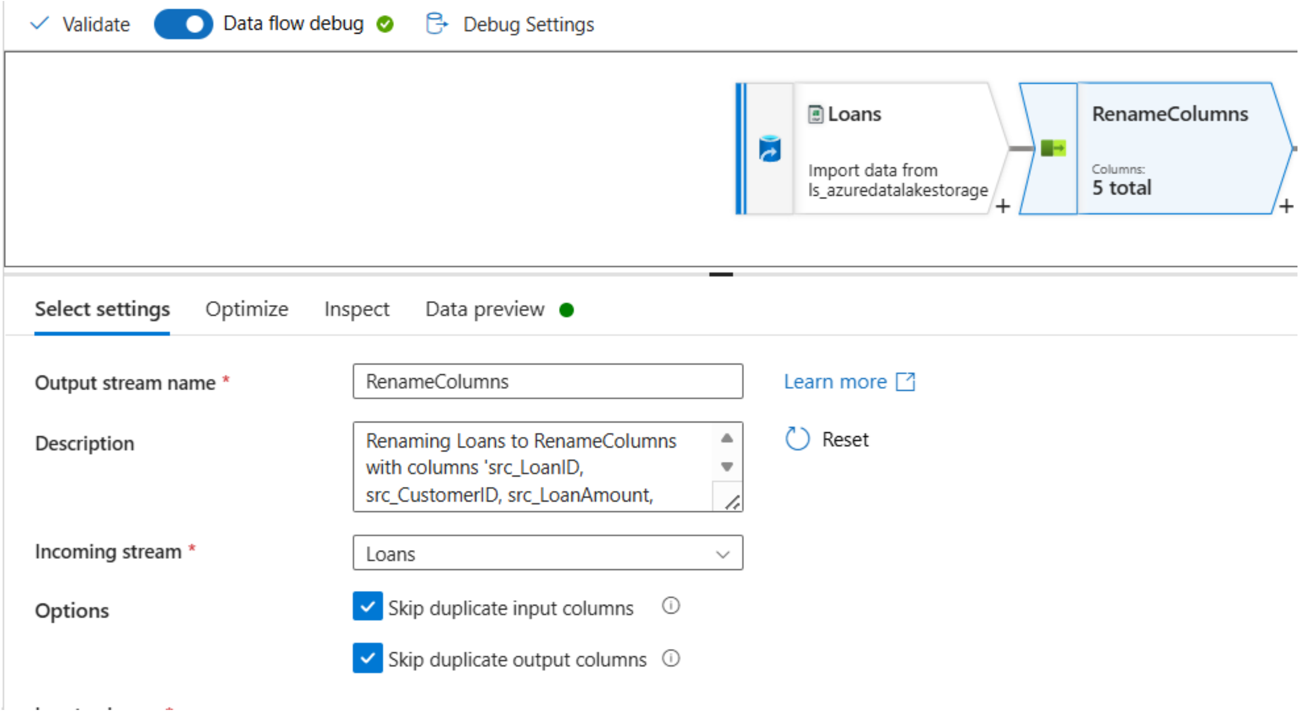

Source settings | Source options | Projection | Optimize | Inspect | Data preview ●

⟵ Import schema | ✕ Clear schema | Schema options

Column name	Type	Format
LoanID	12s short	Specify format
CustomerID	12s short	Specify format
LoanAmount	1.2 double	Specify format
InterestRate	1.2 double	Specify format
LoanTerm	12s short	Specify format

Step 4: Add select column, we will rename columns with src_columnnames.

– Then, under settings select all the columns and delete the mapping.



Select settings | Optimize | Inspect | Data preview ●

Output stream name *: RenameColumns

Learn more

Description: Renaming Loans to RenameColumns with columns 'src_LoanID, src_CustomerID, src_LoanAmount,

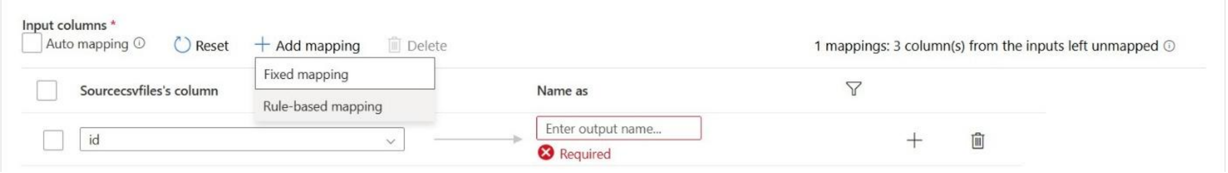
Reset

Incoming stream *: Loans

Options: ☑ Skip duplicate input columns

☑ Skip duplicate output columns

– Click on Add mapping and select rule-based mapping.


Input columns *

□ Auto mapping ⓘ | Reset | + Add mapping | Delete

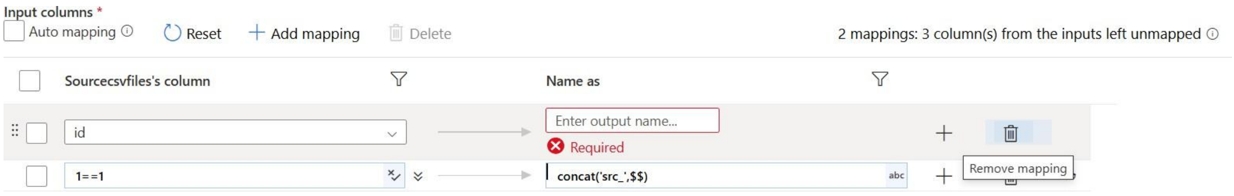
Fixed mapping

Rule-based mapping

1 mappings: 3 column(s) from the inputs left unmapped ⓘ

□	Sourcecsvfiles's column	Name as
□	id	Enter output name... ✕ Required

- In the new column, give condition 1==1 to make it true, and give name as **concat('src_',$$).**
- This will make dynamic expression and add src in front of all the columns. Also, remove the above id one row mapping as we don't need it.

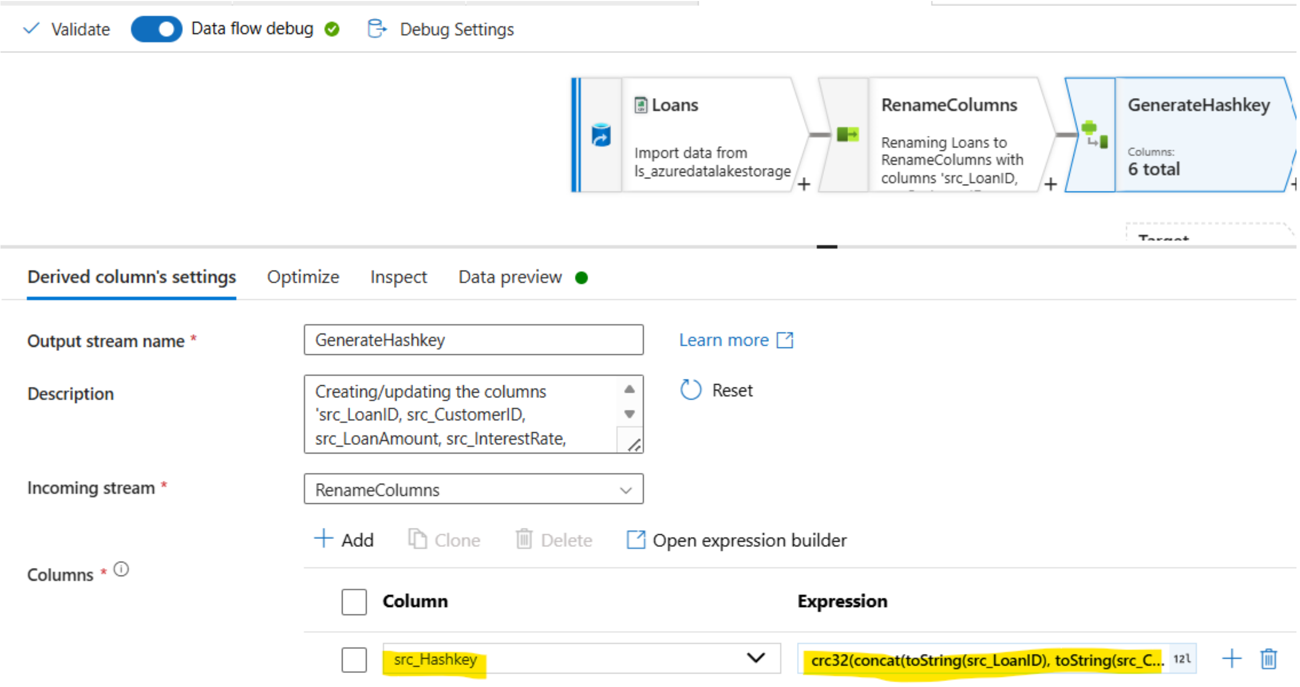


2 mappings: 3 column(s) from the inputs left unmapped ⓘ

Sourcecsvfiles's column	Name as
id	Enter output name... ❌ Required
1==1	concat('src_',$$)

Step 5: Add Derived column activity.

- Under derived column settings, add column src_haskey and enter this expression.
  - **crc32(concat(toString(src_LoanID), toString(src_CustomerID), toString(src_LoanAmount), toString(src_InterestRate), toString(src_LoanTerm)))**
- Crc32 generates hashkey using this combination.



Output stream name *: GenerateHashkey

Description: Creating/updating the columns 'src_LoanID, src_CustomerID, src_LoanAmount, src_InterestRate,

Incoming stream *: RenameColumns

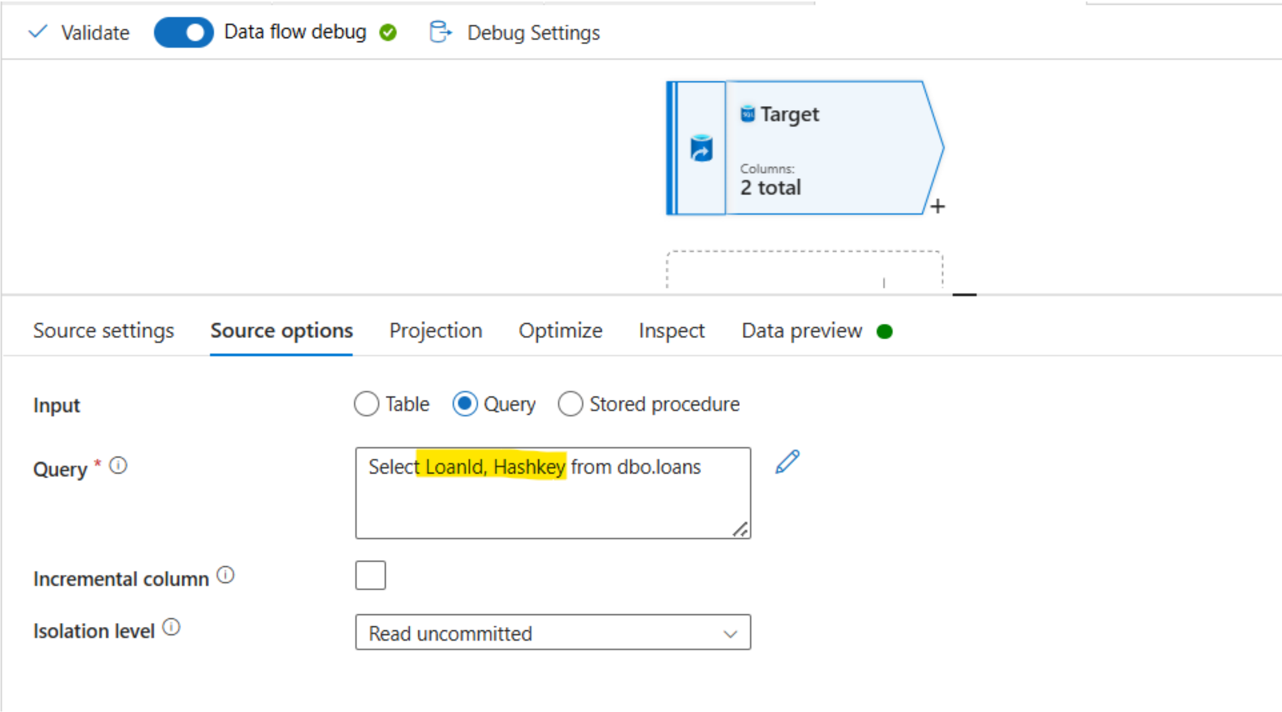
Columns * ⓘ

Column	Expression
src_Hashkey	crc32(concat(toString(src_LoanID), toString(src_C...

Step 6: Add target i.e Azure SQL DB as another source,

- Select two column id and hashkey from table which we will use to check for new records or existing records.

- Click on Projection and import schema to get the Column in the data flow(as we want to access the table in data flow so this is necessary step).



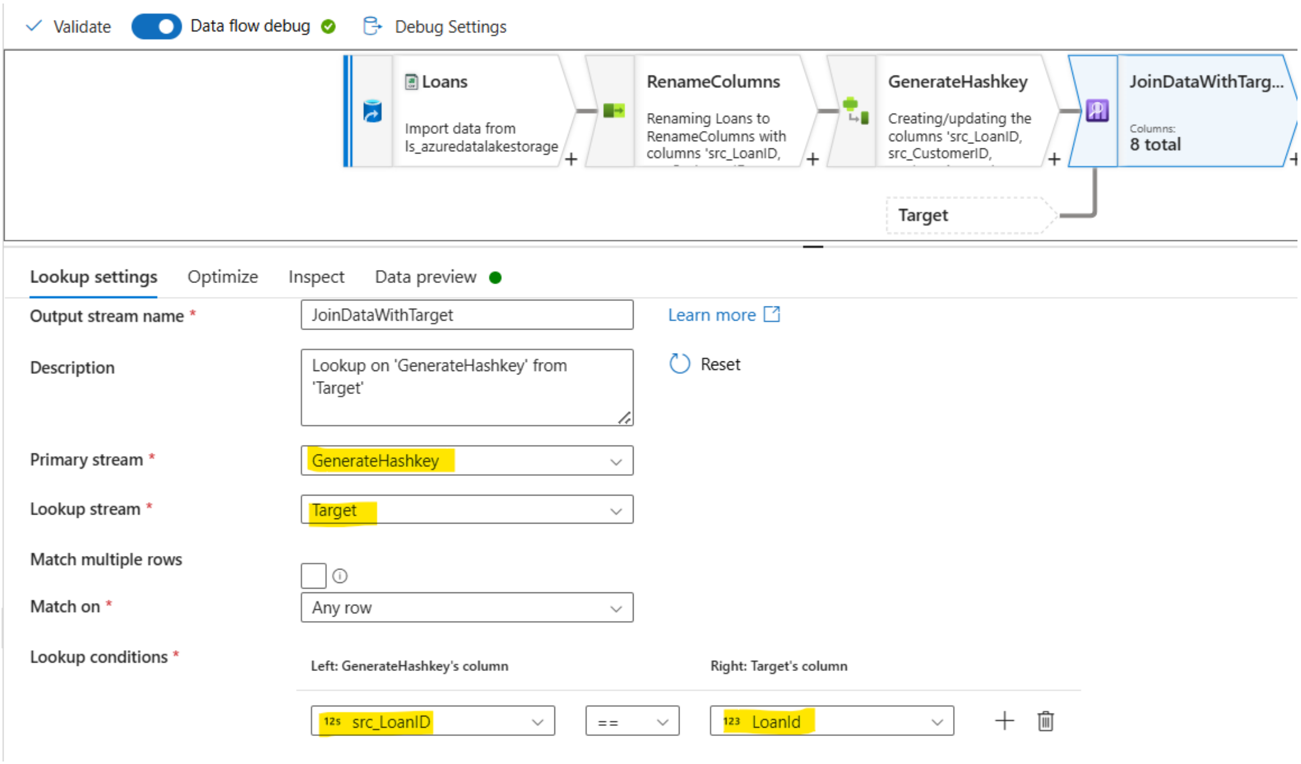
Input ○ Table ● Query ○ Stored procedure

Query * : Select LoanId, Hashkey from dbo.loans

Incremental column □

Isolation level: Read uncommitted

Step 7: Add lookup activity that will perform a left join with target as we will be checking if the record exists or not, and match on ID.



Output stream name * JoinDataWithTarget

Description: Lookup on 'GenerateHashkey' from 'Target'

Primary stream * GenerateHashkey

Lookup stream * Target

Match multiple rows □

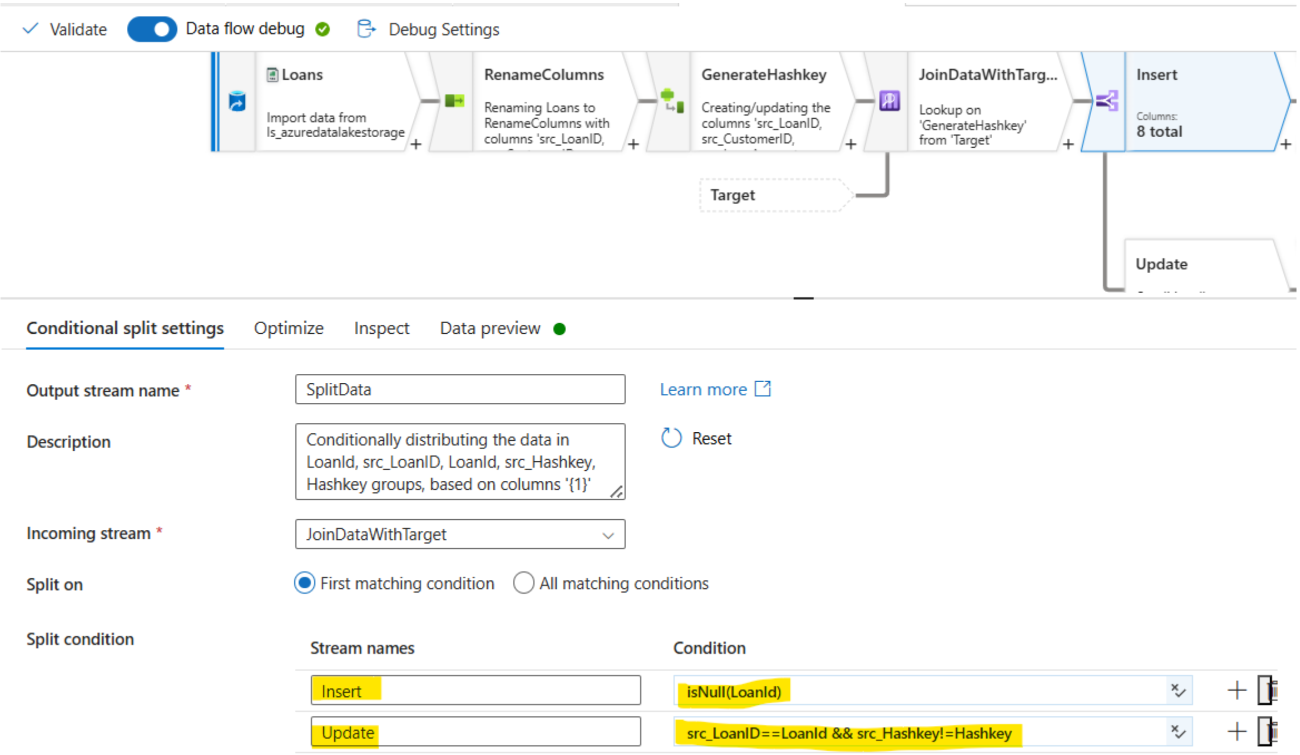
Match on * Any row

Lookup conditions *

Left: GenerateHashkey's column		Right: Target's column
src_LoanID	==	LoanId

Step 8: Add conditional split and add two conditions name Input and Output.

- Input check if LoanId is null, then it's a new record will directly insert it.
- Update checks if our source and Loan ID matches but also hashkey shouldn't match because if there is let's say change in city with ID=1, and unique Hashkey will be generated which cannot match with the already exisiting hashkey. So, means need to update record.


Validate
Data flow debug
Debug Settings
Loans
Import data from ls_azuredatalakestorage
RenameColumns
Renaming Loans to RenameColumns with columns 'src_LoanID,
GenerateHashkey
Creating/updating the columns 'src_LoanID, src_CustomerID,
JoinDataWithTarg...
Lookup on 'GenerateHashkey' from 'Target'
Insert
Columns:
8 total
Target
Update

Conditional split settings | Optimize | Inspect | Data preview ●

Output stream name * SplitData

Learn more

Description: Conditionally distributing the data in LoanId, src_LoanID, LoanId, src_Hashkey, Hashkey groups, based on columns '{1}'

Reset

Incoming stream * JoinDataWithTarget

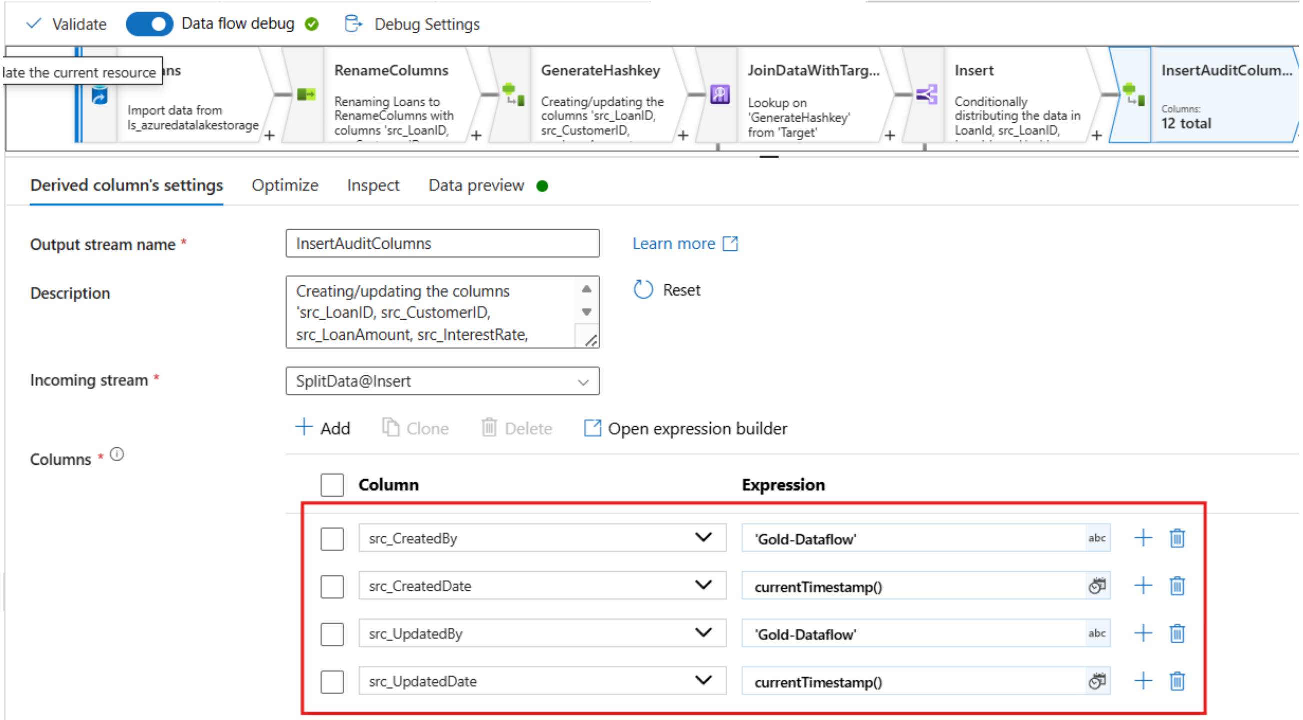
Split on: ● First matching condition ○ All matching conditions

Split condition

Stream names	Condition
Insert	isNull(LoanId)
Update	src_LoanID==LoanId && src_Hashkey!=Hashkey

Step 9: Add a derived column in input side flow,

– Add columns createdby, createddate, updatedby, updateddate,


Validate
Data flow debug
Debug Settings
late the current resource
Import data from ls_azuredatalakestorage
RenameColumns
Renaming Loans to RenameColumns with columns 'src_LoanID,
GenerateHashkey
Creating/updating the columns 'src_LoanID, src_CustomerID,
JoinDataWithTarg...
Lookup on 'GenerateHashkey' from 'Target'
Insert
Conditionally distributing the data in LoanId, src_LoanID,
InsertAuditColum...
Columns:
12 total

Derived column's settings | Optimize | Inspect | Data preview ●

Output stream name * InsertAuditColumns

Learn more

Description: Creating/updating the columns 'src_LoanID, src_CustomerID, src_LoanAmount, src_InterestRate,

Reset

Incoming stream * SplitData@Insert

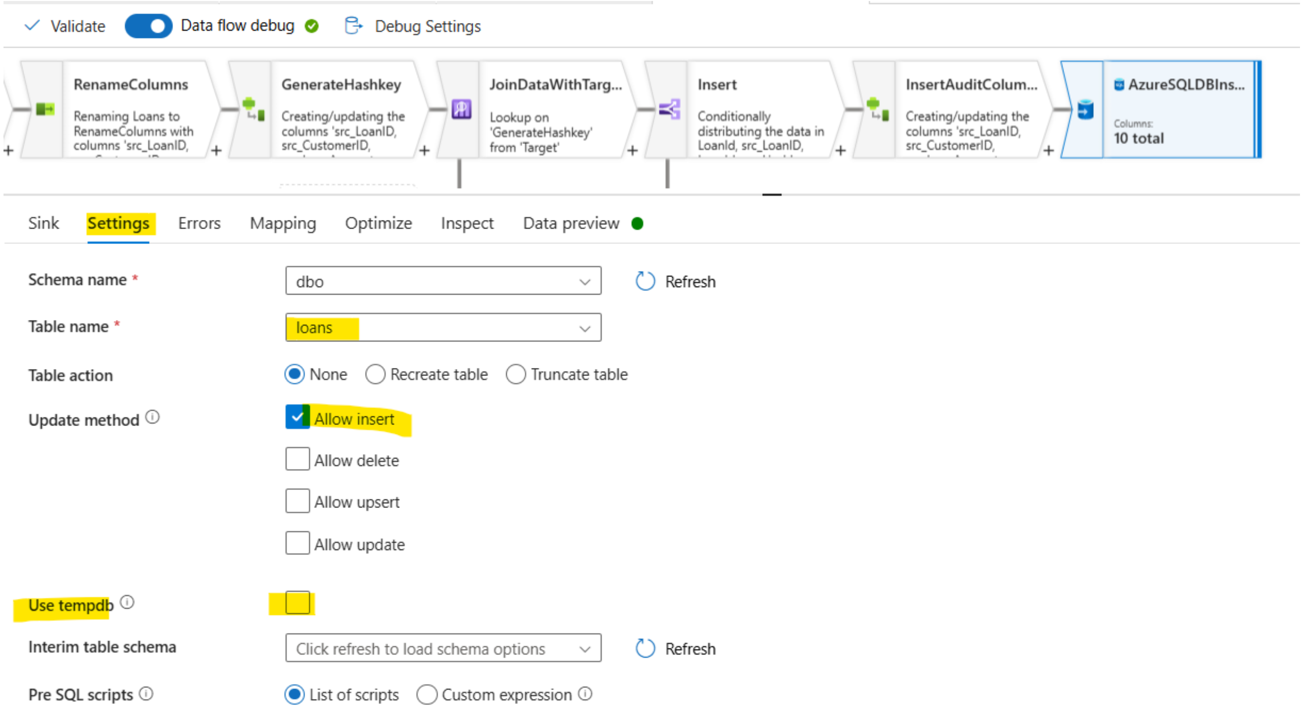
+ Add | Clone | Delete | Open expression builder

Columns *

□ Column	Expression
□ src_CreatedBy	'Gold-Dataflow'
□ src_CreatedDate	currentTimestamp()
□ src_UpdatedBy	'Gold-Dataflow'
□ src_UpdatedDate	currentTimestamp()

Step10: Add sink in input flow.

– Add Azure sql table, and uncheck use tempdb and make sure it's on Allow insert only.


✓ Validate
Data flow debug
Debug Settings
RenameColumns
Renaming Loans to RenameColumns with columns 'src_LoanID,
GenerateHashkey
Creating/updating the columns 'src_LoanID, src_CustomerID,
JoinDataWithTarg...
Lookup on 'GenerateHashkey' from 'Target'
Insert
Conditionally distributing the data in LoanId, src_LoanID,
InsertAuditColum...
Creating/updating the columns 'src_LoanID, src_CustomerID,
AzureSQLDBIns...
Columns:
10 total
Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Refresh

Schema name * dbo

Table name * loans

Table action ● None ○ Recreate table ○ Truncate table

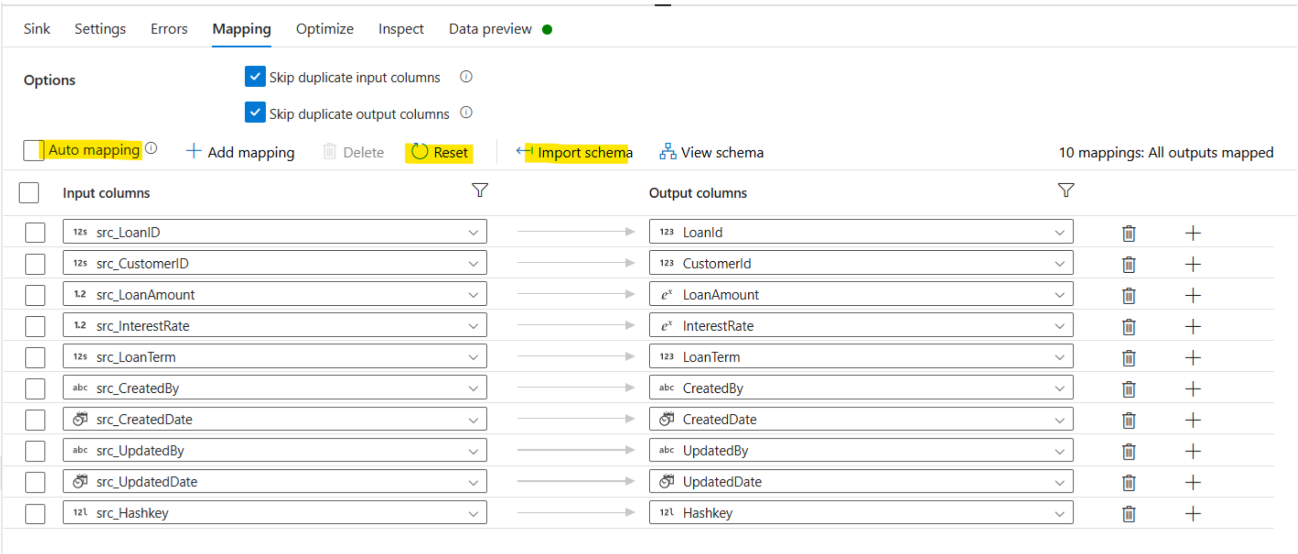
Update method ⓘ
☑ Allow insert
☐ Allow delete
☐ Allow upsert
☐ Allow update

Use tempdb ⓘ ☐

Interim table schema: Click refresh to load schema options

Pre SQL scripts ⓘ ● List of scripts ○ Custom expression ⓘ

– Go to mapping tab-> Click on Import Schema->Then Reset -> Match the input columns.



Options
☑ Skip duplicate input columns ⓘ
☑ Skip duplicate output columns ⓘ

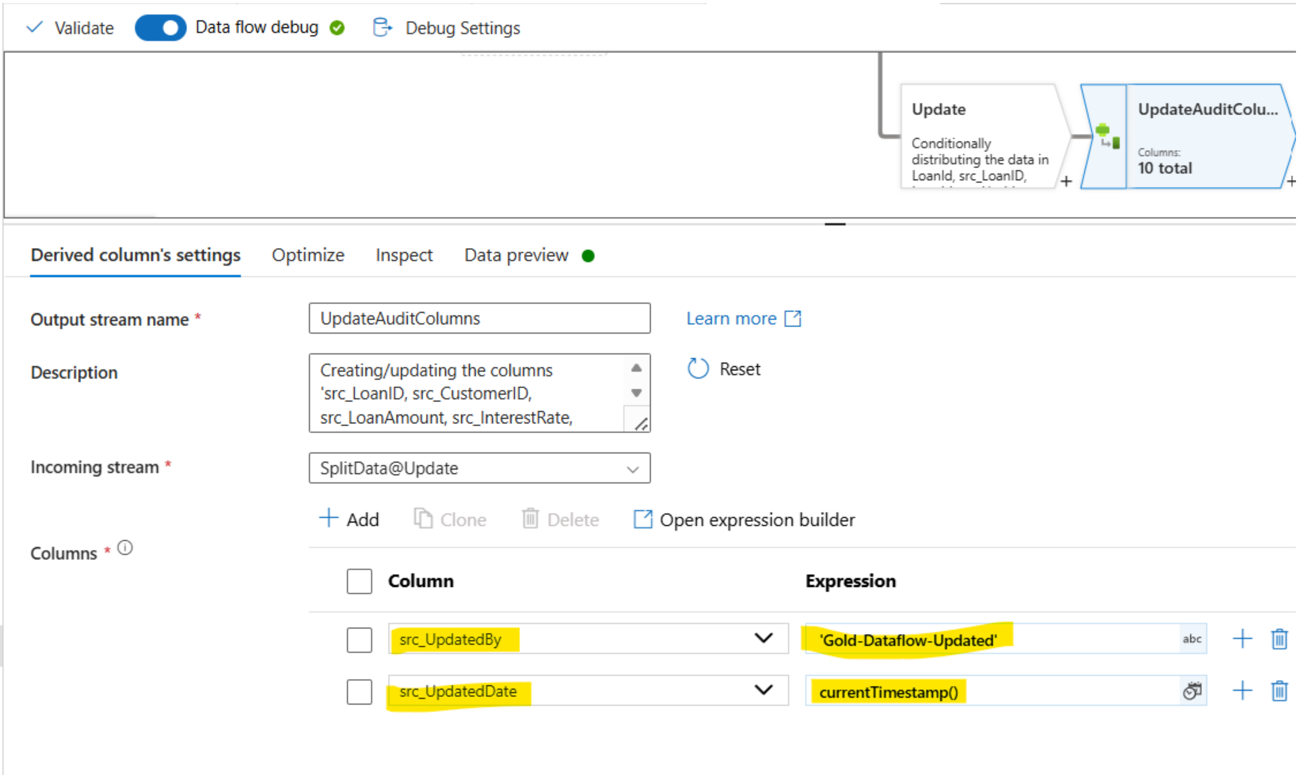
☐ Auto mapping ⓘ

10 mappings: All outputs mapped

Input columns	Output columns
src_LoanID	LoanId
src_CustomerID	CustomerId
src_LoanAmount	LoanAmount
src_InterestRate	InterestRate
src_LoanTerm	LoanTerm
src_CreatedBy	CreatedBy
src_CreatedDate	CreatedDate
src_UpdatedBy	UpdatedBy
src_UpdatedDate	UpdatedDate
src_Hashkey	Hashkey

Step 11: In update flow, add derived column

– Here, we will create updatedby and updateddate column as this update action will happen when there is any change in the existing records and we have to keep createddate same, only updateddate will update with currenttimestamp.



Output stream name * UpdateAuditColumns

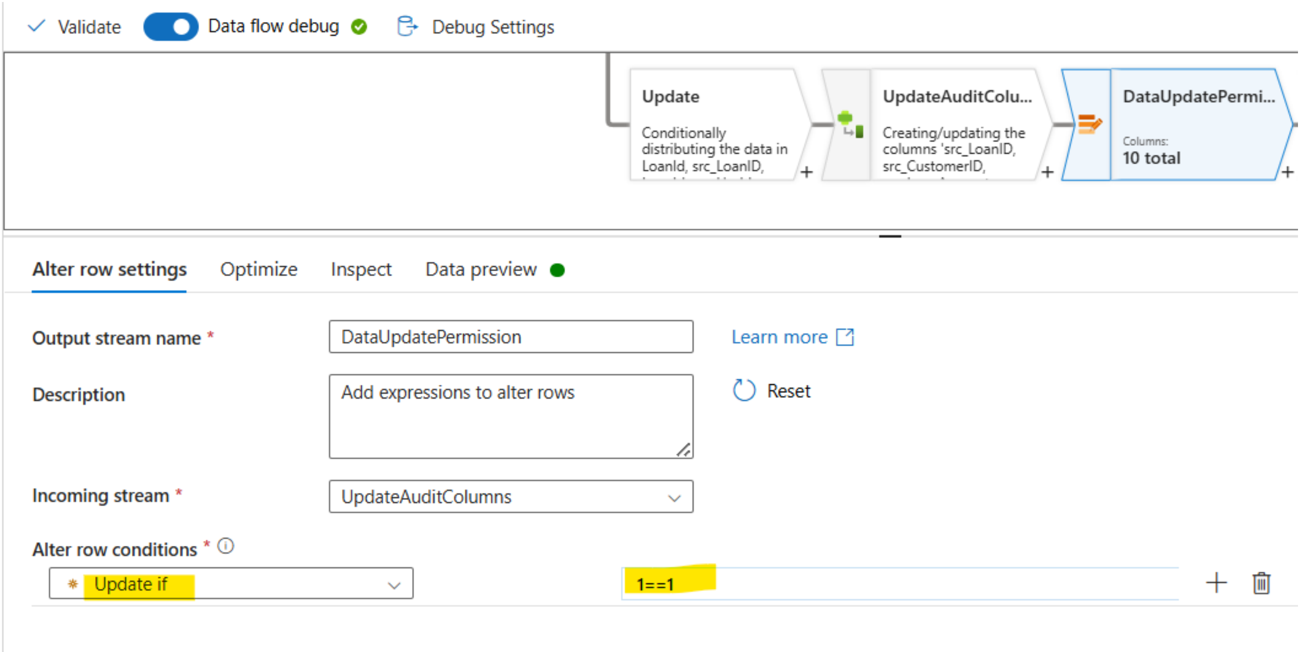
Description: Creating/updating the columns 'src_LoanID, src_CustomerID, src_LoanAmount, src_InterestRate,

Incoming stream * SplitData@Update

Columns *

Column	Expression
src_UpdatedBy	'Gold-Dataflow-Updated'
src_UpdatedDate	currentTimestamp()

Step 12: Add Alterrow transformation which give permission to alter the data.



Output stream name * DataUpdatePermission

Description: Add expressions to alter rows

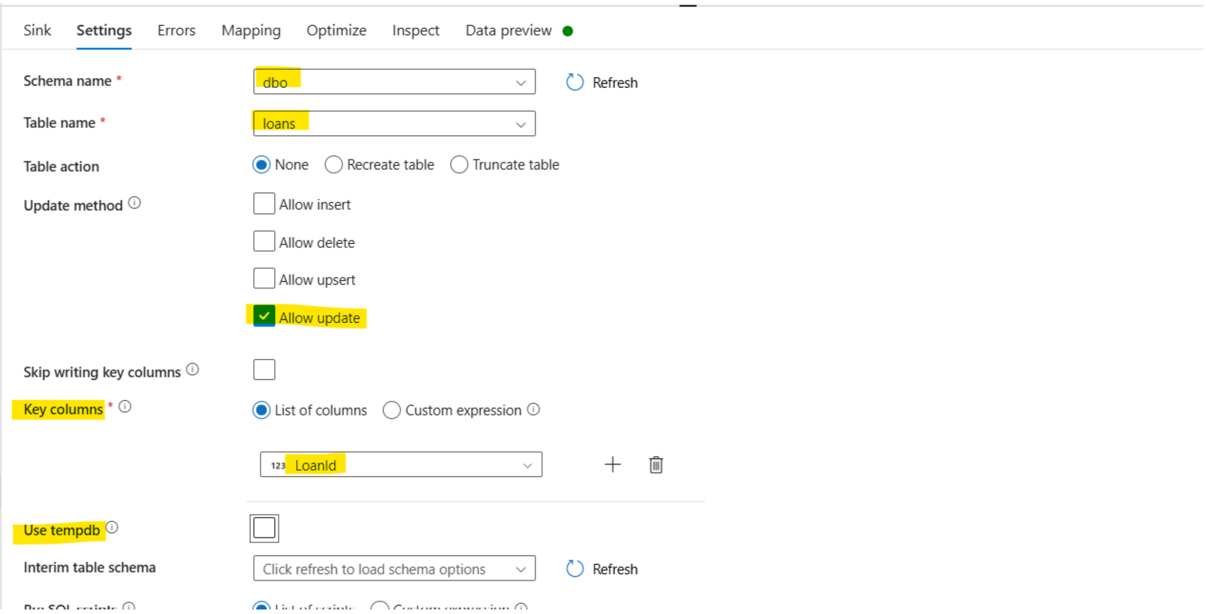
Incoming stream * UpdateAuditColumns

Alter row conditions *

Update if 1==1

Step 13: Add sink and select Azure SQL DB.

– Select only Allow Update checkmark and give LoanId in key columns, to check if srcid==sinkid.



Schema name * dbo — Refresh

Table name * loans

Table action ● None ○ Recreate table ○ Truncate table

Update method ⓘ ☐ Allow insert ☐ Allow delete ☐ Allow upsert ☑ Allow update

Skip writing key columns ⓘ ☐

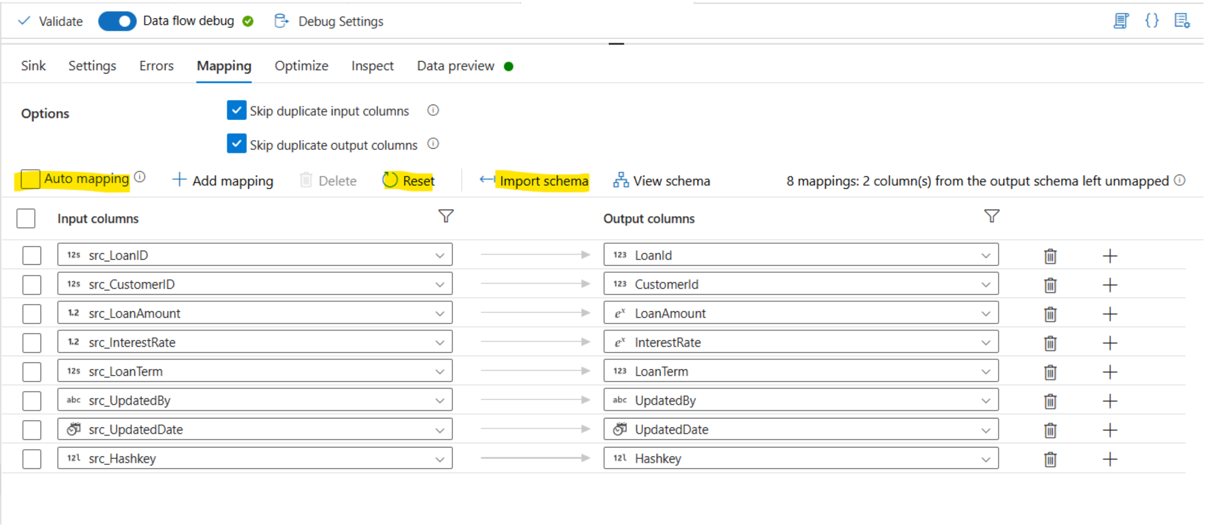
Key columns * ⓘ ● List of columns ○ Custom expression ⓘ

123 LoanId

Use tempdb ⓘ ☐

Interim table schema — Click refresh to load schema options — Refresh

- Go to mapping->Import Schema->Reset->Match input columns.
- Delete createdby and createddate column as we want to keep it same with the actual and don't want to update.



Options ☑ Skip duplicate input columns ⓘ ☑ Skip duplicate output columns ⓘ

☐ Auto mapping ⓘ

8 mappings: 2 column(s) from the output schema left unmapped ⓘ

Input columns	Output columns
123 src_LoanID	123 LoanId
123 src_CustomerID	123 CustomerId
1.2 src_LoanAmount	e^x LoanAmount
1.2 src_InterestRate	e^x InterestRate
123 src_LoanTerm	123 LoanTerm
abc src_UpdatedBy	abc UpdatedBy
src_UpdatedDate	UpdatedDate
12l src_Hashkey	12l Hashkey

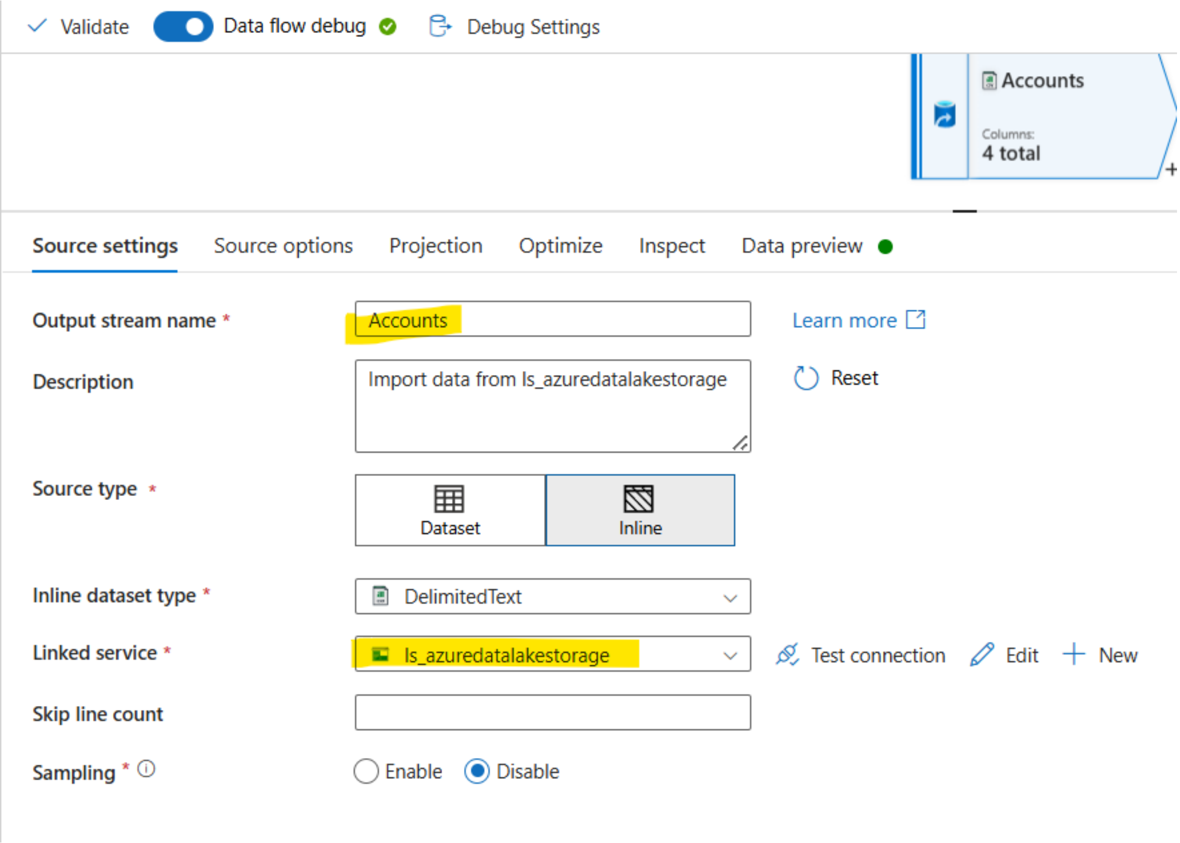
Now, the SCD Type1 implementation is finished for **loans** Data file.

****And similarly, we need to implement the SCD Type 1 approach for the loan_payments, and transactions data files to store it in Gold Layer, i.e. Azure SQL DB tables.**

- ✓ **For SCD type 2, accounts file data.**

Step 1: Drag a new **Data Flow Activity**. Go to the settings and click on new**.** Here, I have created a dataflow with name *df_accounts_scdtype2*.

Step 2: Click on Add source -> Click on Source Type -> Inline and select the source.



Output stream name * Accounts

Description: Import data from ls_azuredatalakestorage

Source type * Dataset / Inline (selected)

Inline dataset type * DelimitedText

Linked service * ls_azuredatalakestorage

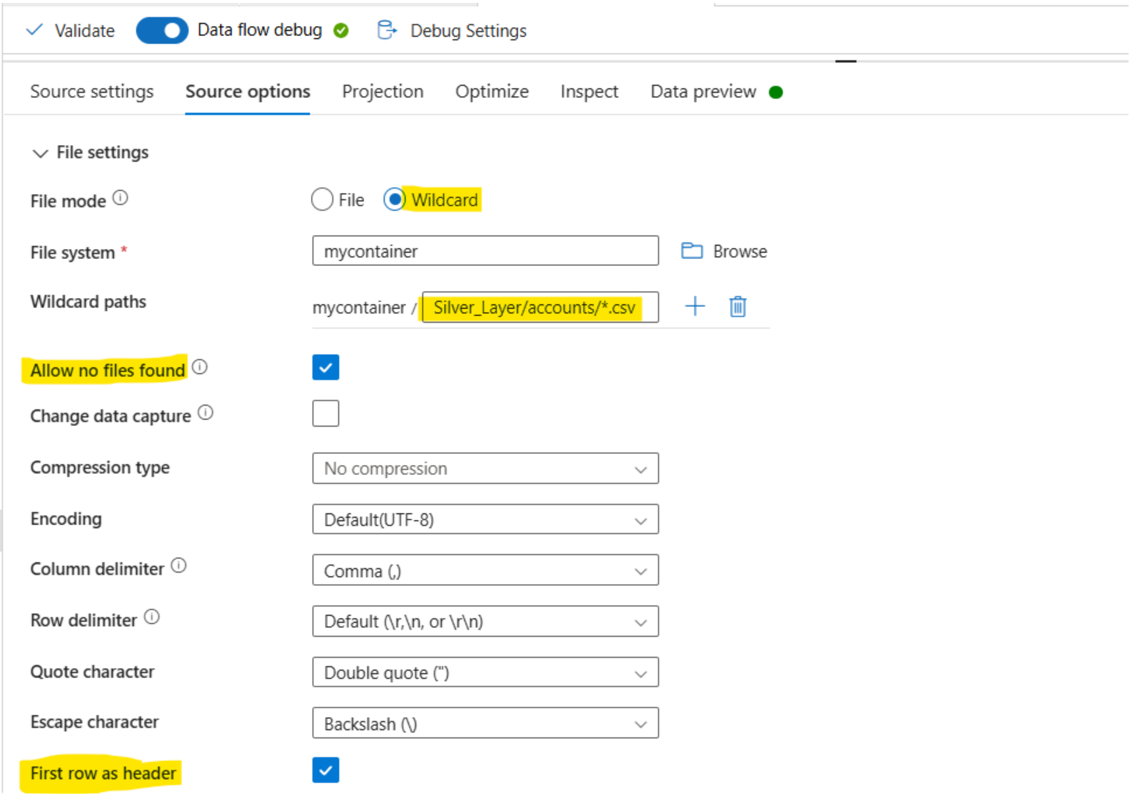
Skip line count

Sampling * ○ Enable ● Disable

– Next, in the Source option, select the file mode as **Wildcard** option as we want to select all the csv files (generated through cluster partition) present in the accounts folder in the Silver_Layer Folder using the wildcard character ***.csv (for selecting the all csv file in the accounts folder).**

– And **checked Allow no files found option** so that if no file is found then it will run on blank file.

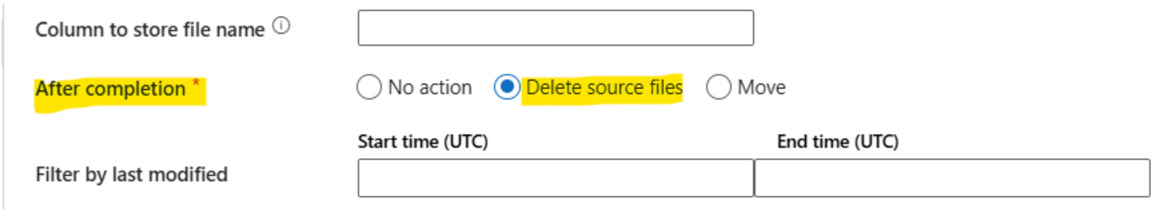
– And checked First row as header option.



File settings

File mode	○ File ● Wildcard
File system *	mycontainer
Wildcard paths	mycontainer / Silver_Layer/accounts/*.csv
Allow no files found	☑
Change data capture	□
Compression type	No compression
Encoding	Default(UTF-8)
Column delimiter	Comma (,)
Row delimiter	Default (\r,\n, or \r\n)
Quote character	Double quote (")
Escape character	Backslash (\)
First row as header	☑

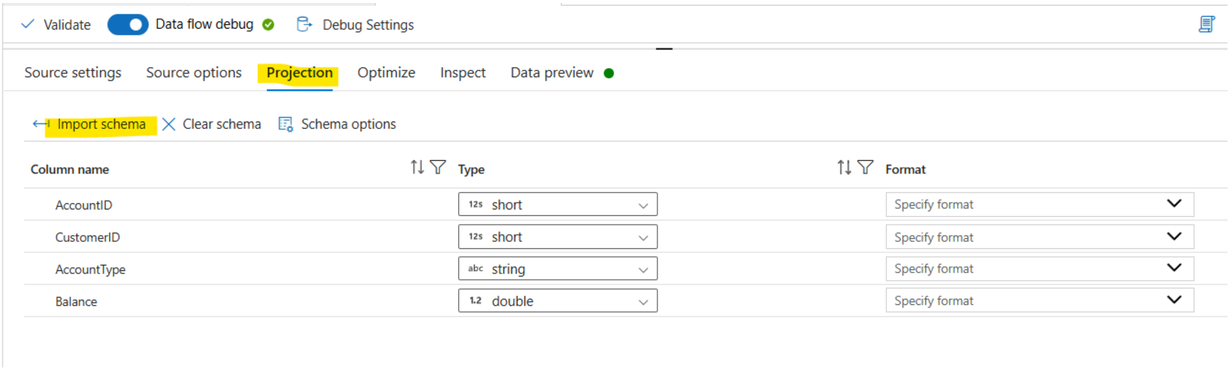
– Next, I want to delete the source file after execution so select **Delete source files** option below.



Column to store file name	
After completion *	○ No action ● Delete source files ○ Move

Filter by last modified	Start time (UTC):
	 End time (UTC):

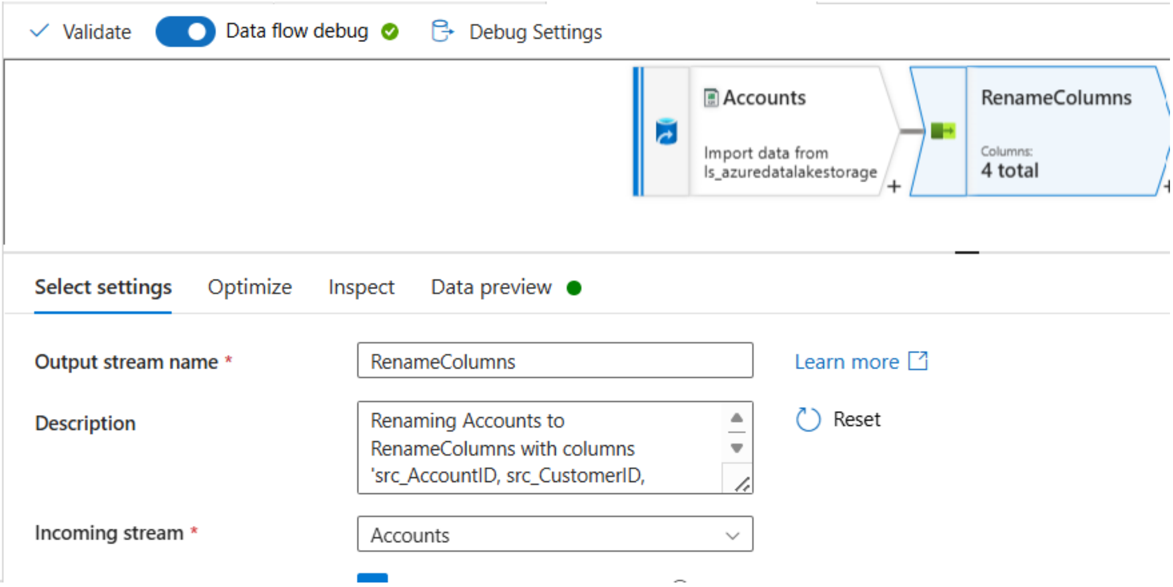
Step 3: Click on Projection tab and import schema and check for Data preview.



Column name	Type	Format
AccountID	short	Specify format
CustomerID	short	Specify format
AccountType	string	Specify format
Balance	double	Specify format

Step 4: Add select column, we will rename columns with src_columnnames.

– Then, under settings select all the columns and delete the mapping.

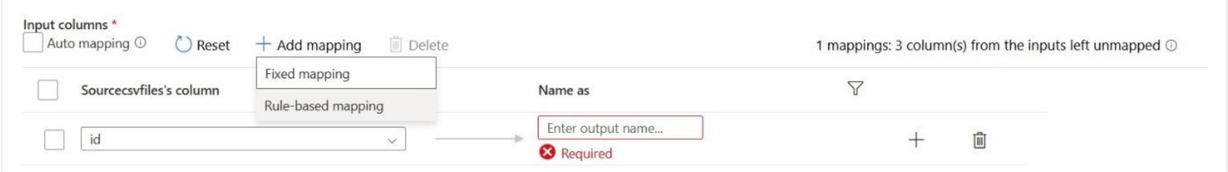


Output stream name * RenameColumns

Description: Renaming Accounts to RenameColumns with columns 'src_AccountID, src_CustomerID,

Incoming stream * Accounts

– Click on Add mapping and select rule-based mapping.

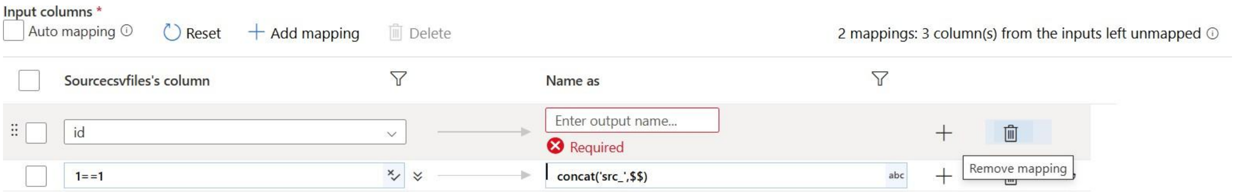


1 mappings: 3 column(s) from the inputs left unmapped

id

– In the new column, give condition 1==1 to make it true, and give name as **concat('src_',$$).**

– This will make dynamic expression and add src in front of all the columns. Also, remove the above id one row mapping as we don't need it.



2 mappings: 3 column(s) from the inputs left unmapped

id

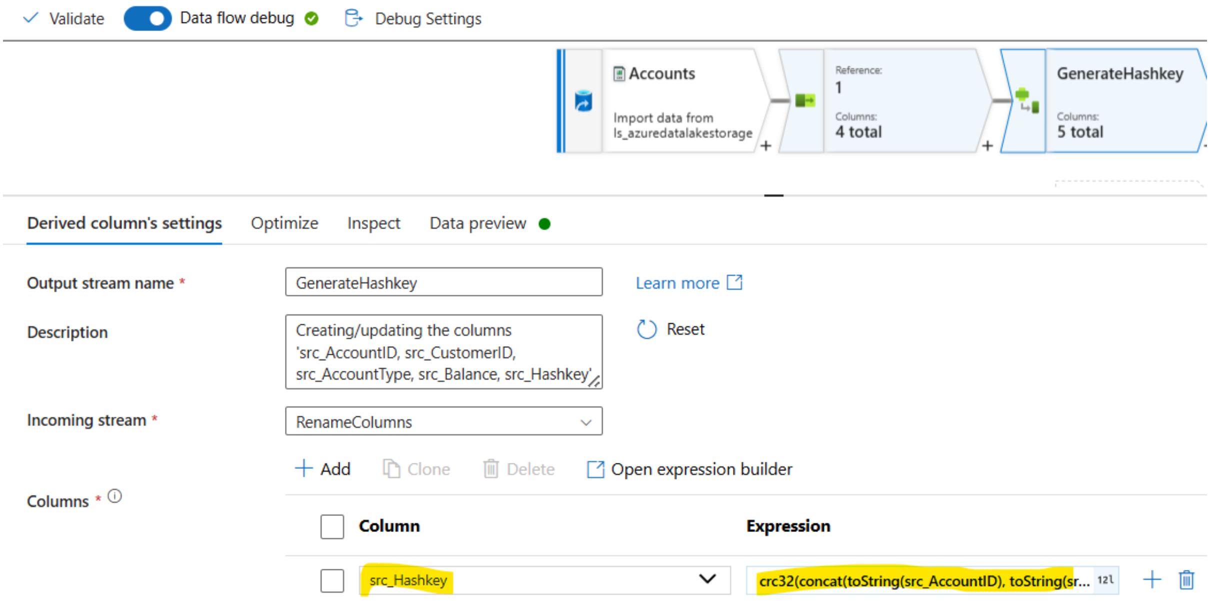
1==1 → concat('src_',$$)

Step 5: Add Derived column activity.

– Under derived column settings, add column src_haskey and enter this expression.

- **crc32(concat(toString(src_AccountID), toString(src_CustomerID), src_AccountType, toString(src_Balance)))**

– Crc32 generates hashkey using this combination.

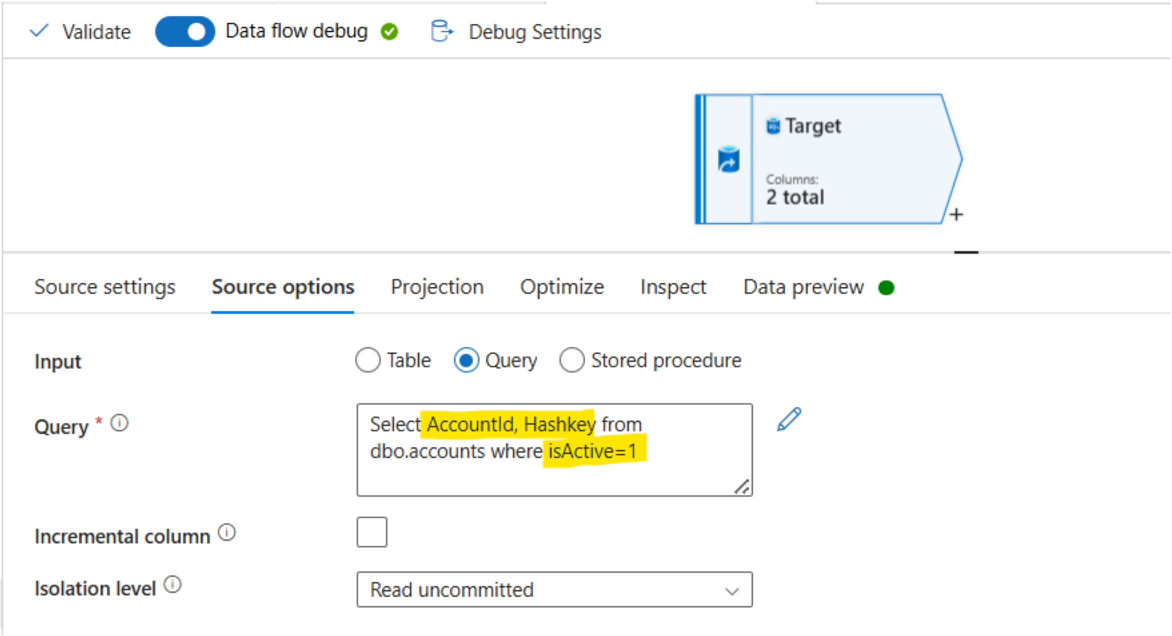


Output stream name *	GenerateHashkey
Description	Creating/updating the columns 'src_AccountID, src_CustomerID, src_AccountType, src_Balance, src_Hashkey'
Incoming stream *	RenameColumns
Columns *	

□ Column	Expression
□ src_Hashkey	crc32(concat(toString(src_AccountID), toString(sr...

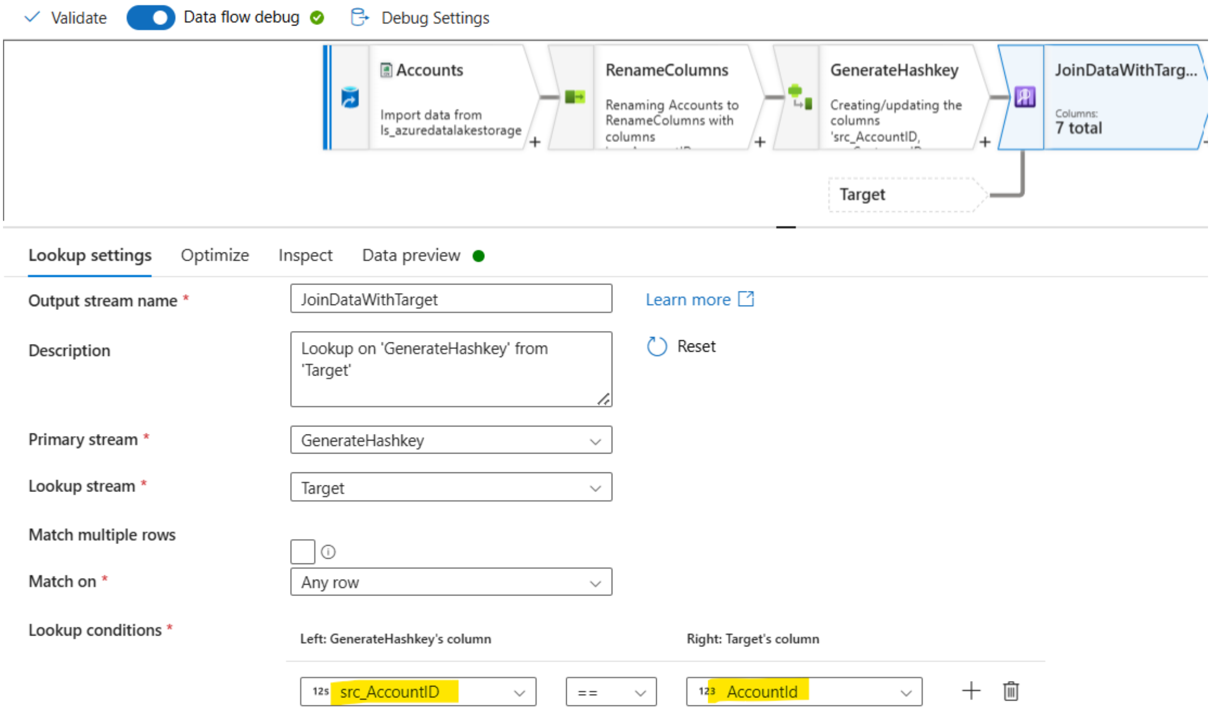
Step 6: Add target i.e Azure SQL DB as another source,

- Select two column id and hashkey from table which we will use to check for new records or existing records. We need rows where isActive is 1 which means the most updated records.
- Click on Projection and import schema to get the Column in the data flow(as we want to access the table in data flow so this is necessary step).



Input	○ Table ● Query ○ Stored procedure
Query *	Select AccountId, Hashkey from dbo.accounts where isActive=1
Incremental column	□
Isolation level	Read uncommitted

Step 7: Add lookup activity that will perform a left join with target as we will be checking if the record exists or not, and match on IDs.



Output stream name * JoinDataWithTarget

Description: Lookup on 'GenerateHashkey' from 'Target'

Primary stream * GenerateHashkey

Lookup stream * Target

Match multiple rows □

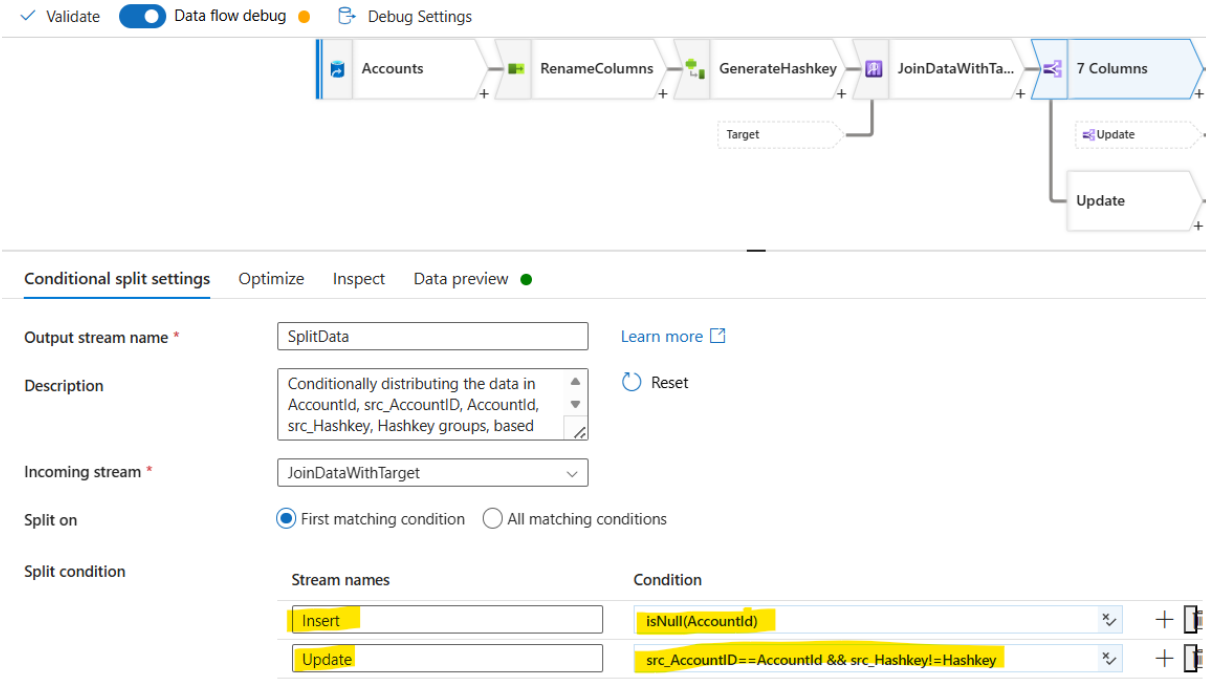
Match on * Any row

Lookup conditions *

Left: GenerateHashkey's column		Right: Target's column
src_AccountID	==	AccountId

Step 8: Add Conditional split and add two conditions name **Insert** and **Update**.

- Input check if Account Id is null, then it's a new record will directly insert it.
- Update checks if our source account ID and target account ID matches but also hashkey shouldn't match because if there is let's say change in city with ID=1, and unique Hashkey will be generated which cannot match with the already exisiting hashkey. So, means need to update record.



Output stream name * SplitData

Description: Conditionally distributing the data in AccountId, src_AccountID, AccountId, src_Hashkey, Hashkey groups, based

Incoming stream * JoinDataWithTarget

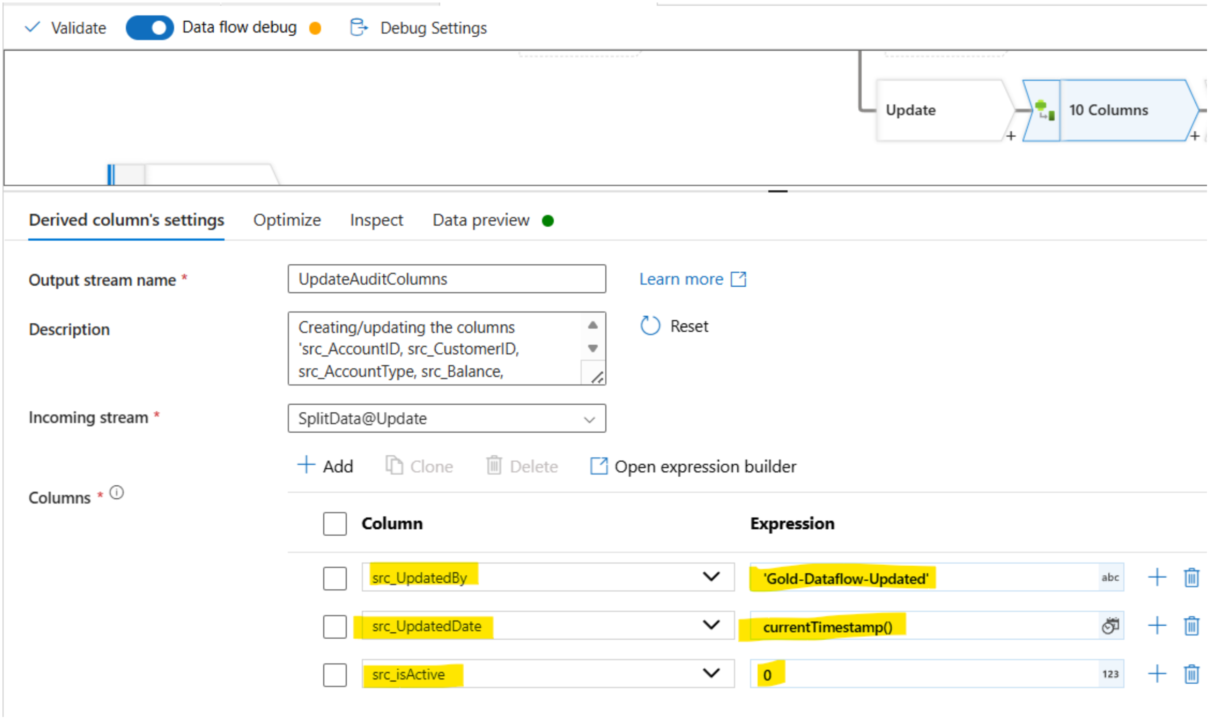
Split on ● First matching condition ○ All matching conditions

Split condition

Stream names	Condition
Insert	isNull(AccountId)
Update	src_AccountID==AccountId && src_Hashkey!=Hashkey

Step 9: In update flow, add derived column

- Here, we will create src_UpdatedBy, src_UpdatedDate, and src_isActive columns as this update action will happen when there is any change in the existing records, and we have to keep CreatedDate and CreatedBy same, only UpdatedBy and updatedDate will update as dataflow-Updated and current timestamp.



Output stream name * UpdateAuditColumns

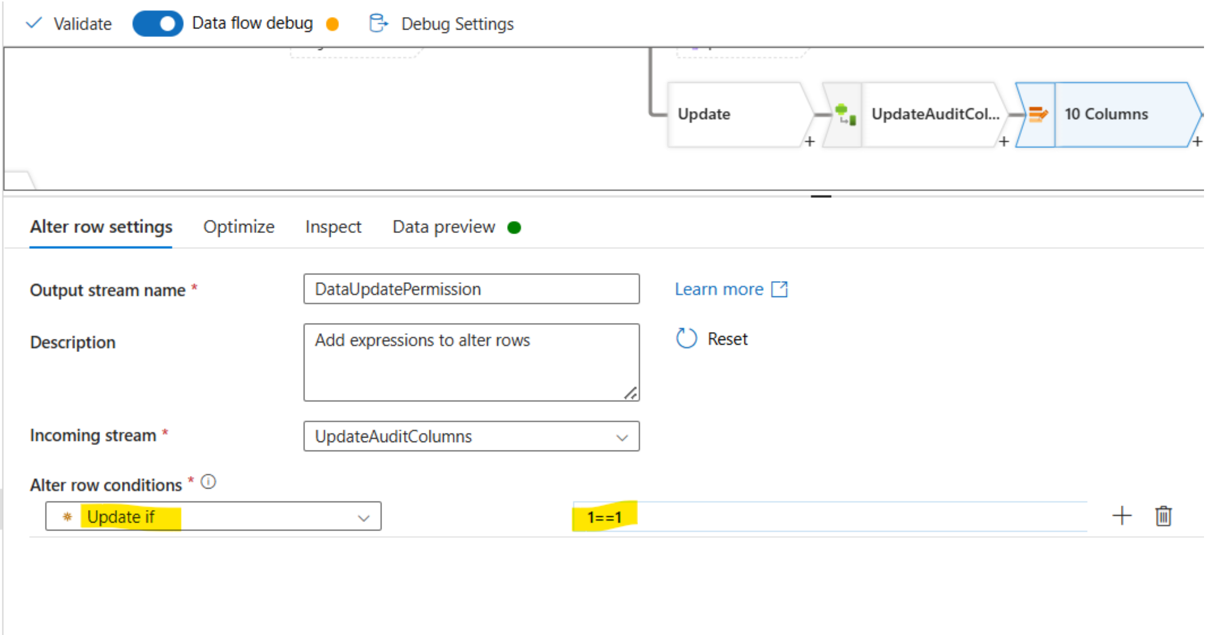
Description: Creating/updating the columns 'src_AccountID, src_CustomerID, src_AccountType, src_Balance,

Incoming stream * SplitData@Update

Columns *

Column	Expression
src_UpdatedBy	'Gold-Dataflow-Updated'
src_UpdatedDate	currentTimestamp()
src_isActive	0

Step 10: Add Alter row transformation which gives permission to alter the data.



Output stream name * DataUpdatePermission

Description: Add expressions to alter rows

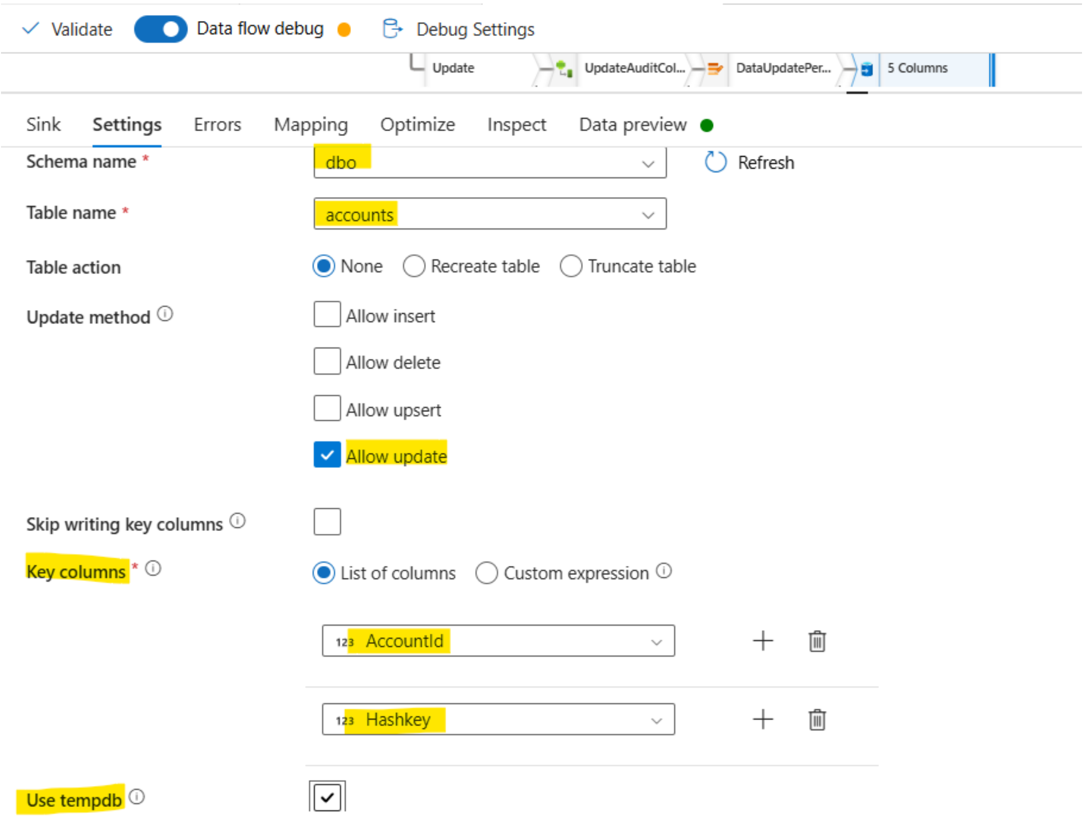
Incoming stream * UpdateAuditColumns

Alter row conditions *

Update if 1==1

Step 11: Add sink and select Azure SQL DB

- Select only Allow Update checkmark and give AccountId and HashKey in key columns, because we will have multiple ID because we are keeping previous record as well, so this combination will help to identify changes.
- Go to mapping->Import Schema->Reset->Match input columns.
- Delete CreatedBy and CreatedDate column as we want to keep it same with the actual and don't want to update.
- HashKey will also be kept the same by mapping it with the old one.



Schema name * dbo

Table name * accounts

Table action ● None ○ Recreate table ○ Truncate table

Update method ⓘ
□ Allow insert
□ Allow delete
□ Allow upsert
☑ Allow update

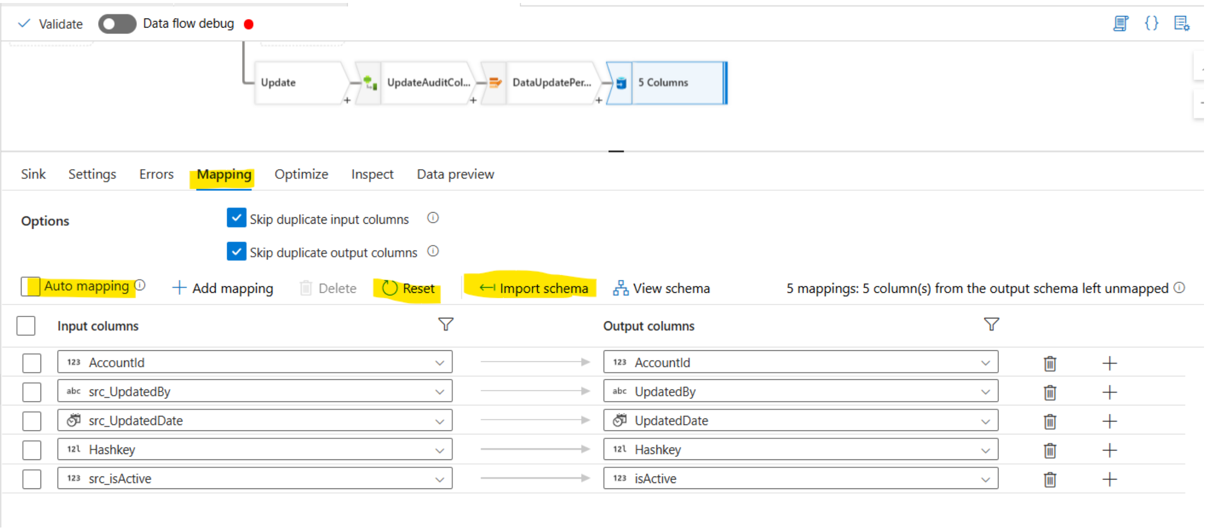
Skip writing key columns ⓘ □

Key columns * ⓘ ● List of columns ○ Custom expression ⓘ

AccountId

Hashkey

Use tempdb ⓘ ☑



Options
☑ Skip duplicate input columns ⓘ
☑ Skip duplicate output columns ⓘ

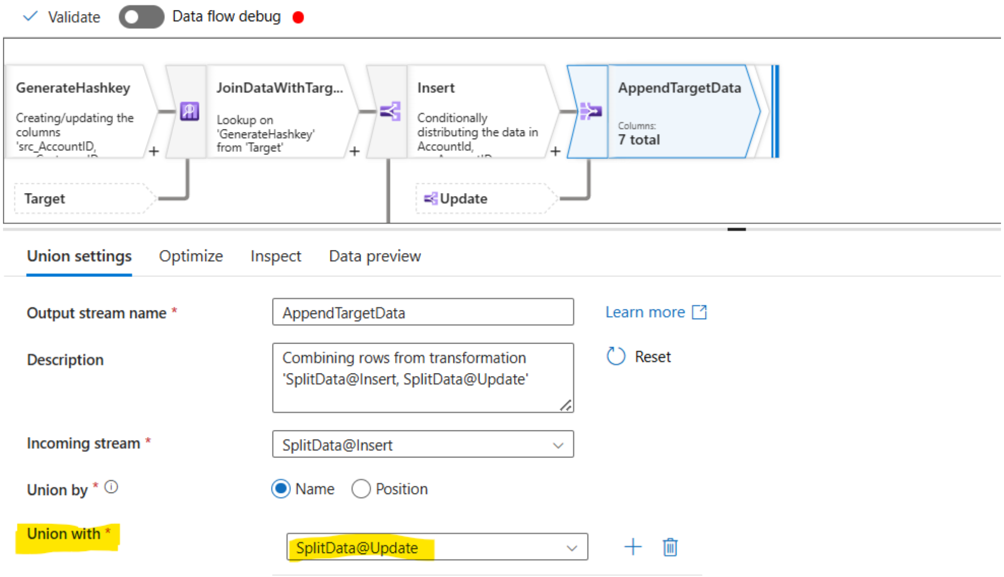
□ Auto mapping ⓘ

5 mappings: 5 column(s) from the output schema left unmapped ⓘ

Input columns	Output columns
AccountId	AccountId
src_UpdatedBy	UpdatedBy
src_UpdatedDate	UpdatedDate
Hashkey	Hashkey
src_isActive	isActive

Step 12: On the insert side, add union because we want to keep new and previous records.

– In union with, select update branch


✓ Validate
Data flow debug
GenerateHashkey
Creating/updating the columns 'src_AccountID,
JoinDataWithTarg...
Lookup on 'GenerateHashkey' from 'Target'
Insert
Conditionally distributing the data in AccountId,
AppendTargetData
Columns:
7 total
Target
Update

Union settings | Optimize | Inspect | Data preview

Output stream name * AppendTargetData

Learn more

Description: Combining rows from transformation 'SplitData@Insert, SplitData@Update'

Reset

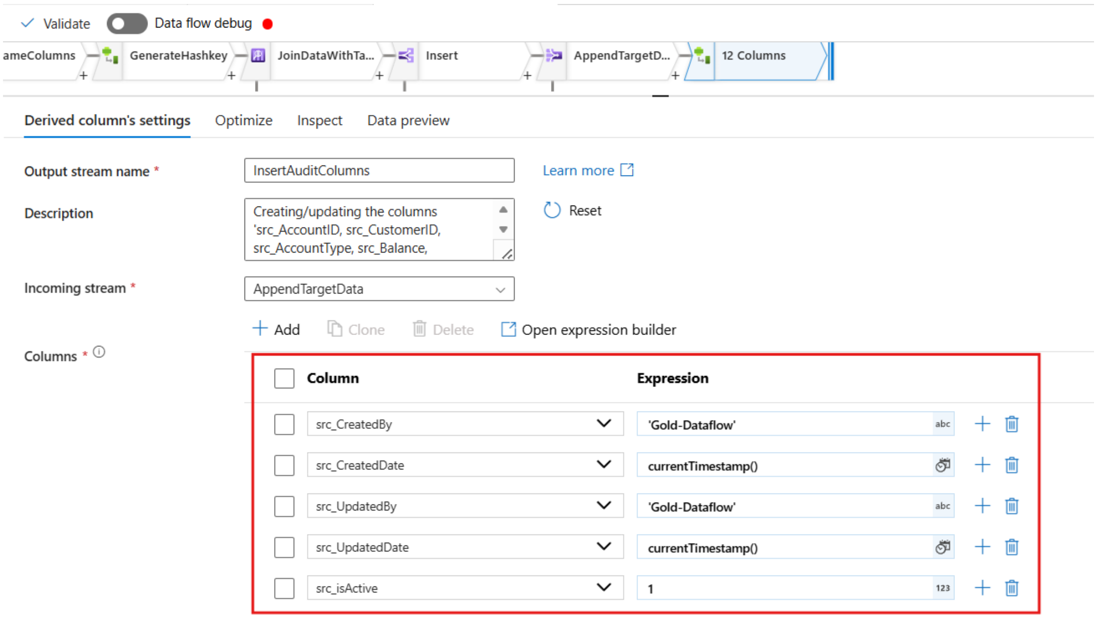
Incoming stream * SplitData@Insert

Union by * ● Name ○ Position

Union with * SplitData@Update

Step 13: Create a derived column to create the following columns

– src_CreatedBy – with value as ‘DataFlow’.
– src_CreatedDate – with value as current time stamp.
– src_sUpdatedBy – with value as ‘DataFlow’.
– src_UpdatedDate – with values as current time stamp.
– src_isActive – with values as 1.



Derived column's settings | Optimize | Inspect | Data preview

Output stream name * InsertAuditColumns

Learn more

Description: Creating/updating the columns 'src_AccountID, src_CustomerID, src_AccountType, src_Balance,

Reset

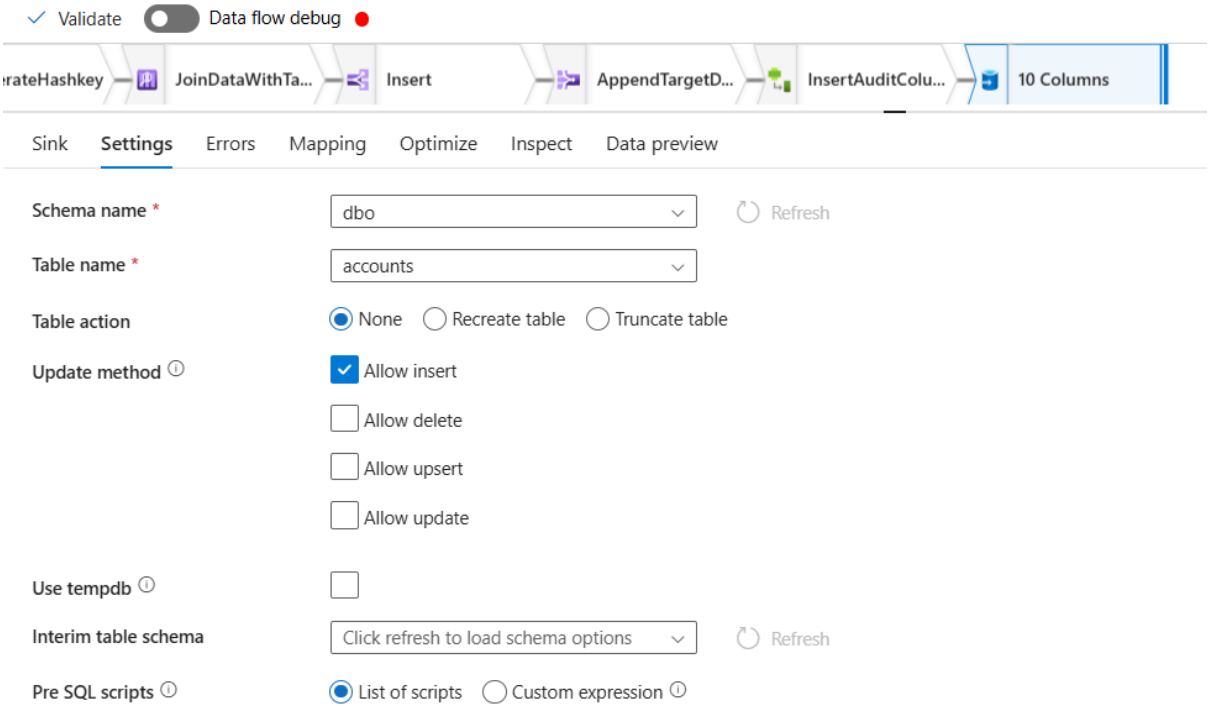
Incoming stream * AppendTargetData

+ Add | Clone | Delete | Open expression builder

Columns *

□ Column	Expression
□ src_CreatedBy	'Gold-Dataflow'
□ src_CreatedDate	currentTimestamp()
□ src_UpdatedBy	'Gold-Dataflow'
□ src_UpdatedDate	currentTimestamp()
□ src_isActive	1

Step 14: Create sink, select Azure SQL DB and map the columns input->output by import schema and then reset and map accordingly.



Schema name * dbo

Table name * accounts

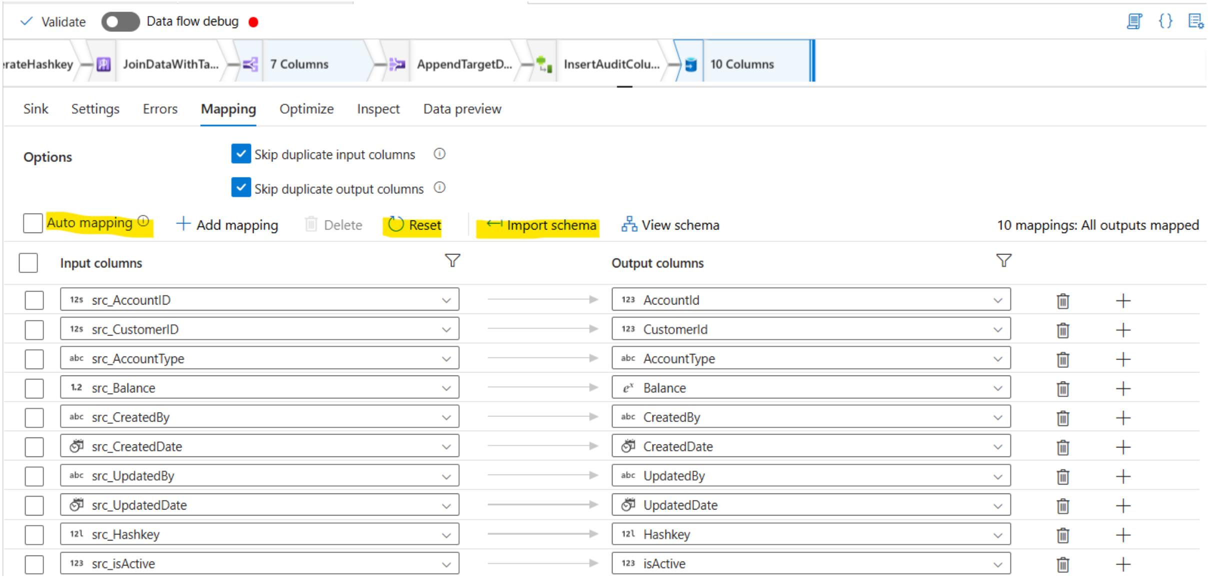
Table action ● None ○ Recreate table ○ Truncate table

Update method ☑ Allow insert
□ Allow delete
□ Allow upsert
□ Allow update

Use tempdb □

Interim table schema: Click refresh to load schema options

Pre SQL scripts ● List of scripts ○ Custom expression



Options ☑ Skip duplicate input columns
☑ Skip duplicate output columns

10 mappings: All outputs mapped

Input columns	Output columns
src_AccountID	AccountId
src_CustomerID	CustomerId
src_AccountType	AccountType
src_Balance	Balance
src_CreatedBy	CreatedBy
src_CreatedDate	CreatedDate
src_UpdatedBy	UpdatedBy
src_UpdatedDate	UpdatedDate
src_Hashkey	Hashkey
src_isActive	isActive

Now, the SCD Type2 implementation is finished for **accounts** Data file.

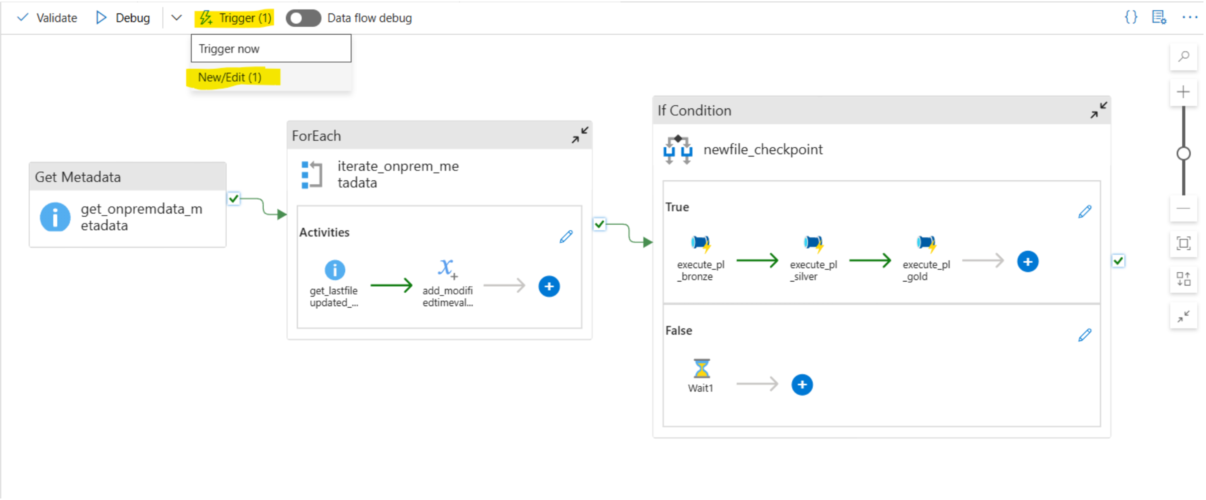
****And similarly, we need to implement the SCD Type 2 approach for the customers data file to store it in Gold Layer, i.e. Azure SQL DB tables.**

Pipeline Execution Automation

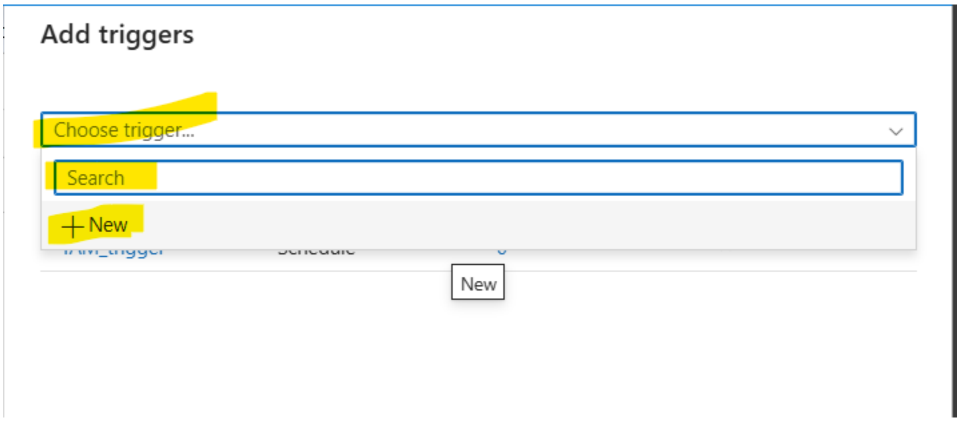
Here, I have used the **scheduled trigger** to automate the execution of entire pipeline to load the data **every morning at 4AM**. I have setup a scheduled trigger on Master pipeline that will trigger the master pipeline every morning at 4AM.

➢ Steps To Setup the Scheduled Trigger to master pipeline are as follows.

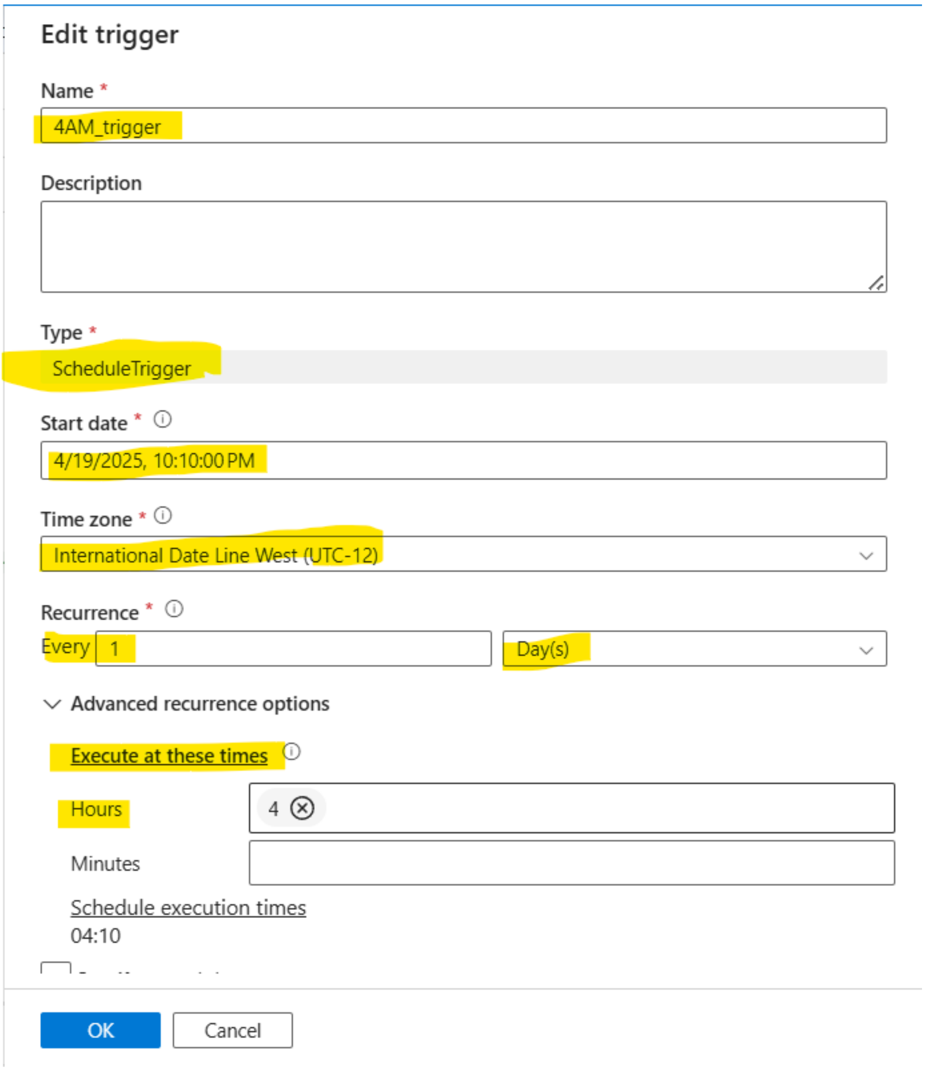
Step 1: Open the Master pipeline, click on trigger on the above options -> click on New/Edit option.



Step 2: click on choose trigger... -> click on +New.



Step 3: Enter the basic details such as trigger name, trigger type, startdate, timezone etc. -> click on OK.



Edit trigger

Name *
4AM_trigger

Description

Type *
ScheduleTrigger

Start date *
4/19/2025, 10:10:00 PM

Time zone *
International Date Line West (UTC-12)

Recurrence *
Every 1 Day(s)

Advanced recurrence options

Execute at these times

Hours 4

Minutes

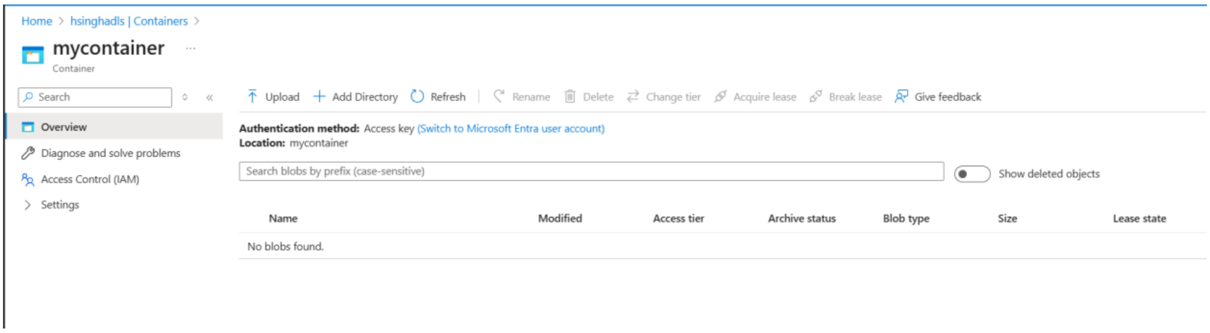
Schedule execution times
04:10

Now, the trigger setup is completed.

Output Review

✓ **Master Pipeline First Run.**

– The ADLS gen2 Storage container before pipelines execution.



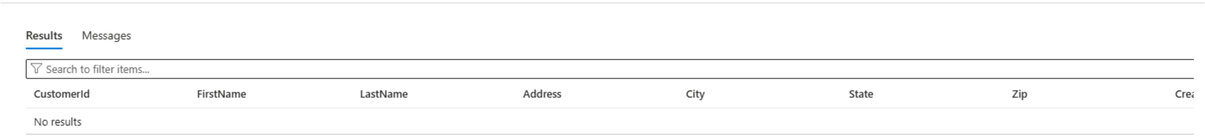
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: mycontainer

Search blobs by prefix (case-sensitive)

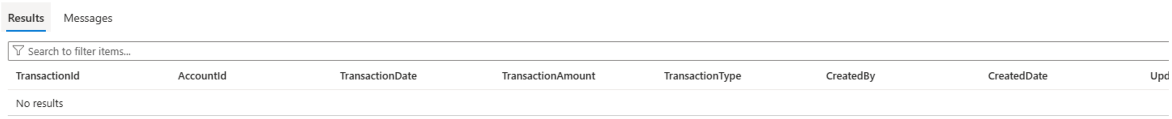
Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
No blobs found.						

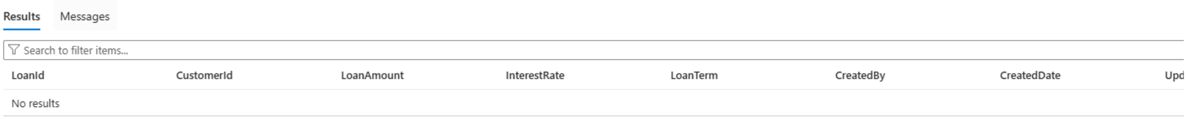
– The Azure SQL DB all 5 Tables output before pipelines execution.



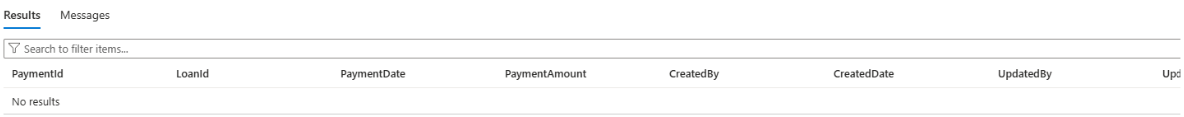
CustomerId	FirstName	LastName	Address	City	State	Zip	Crea
No results							



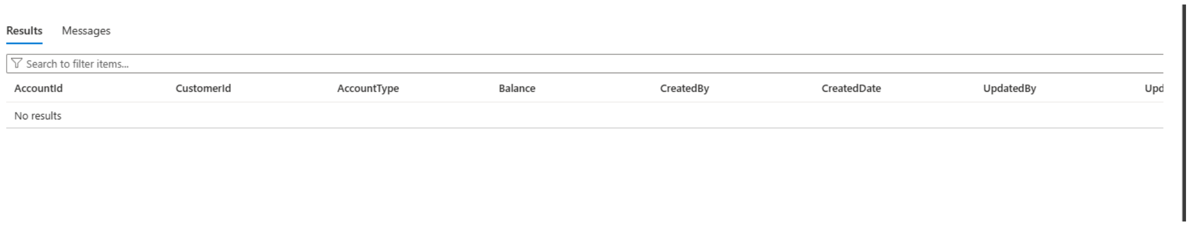
TransactionId	AccountId	TransactionDate	TransactionAmount	TransactionType	CreatedBy	CreatedDate	Upd
No results							



LoanId	CustomerId	LoanAmount	InterestRate	LoanTerm	CreatedBy	CreatedDate	Upd
No results							

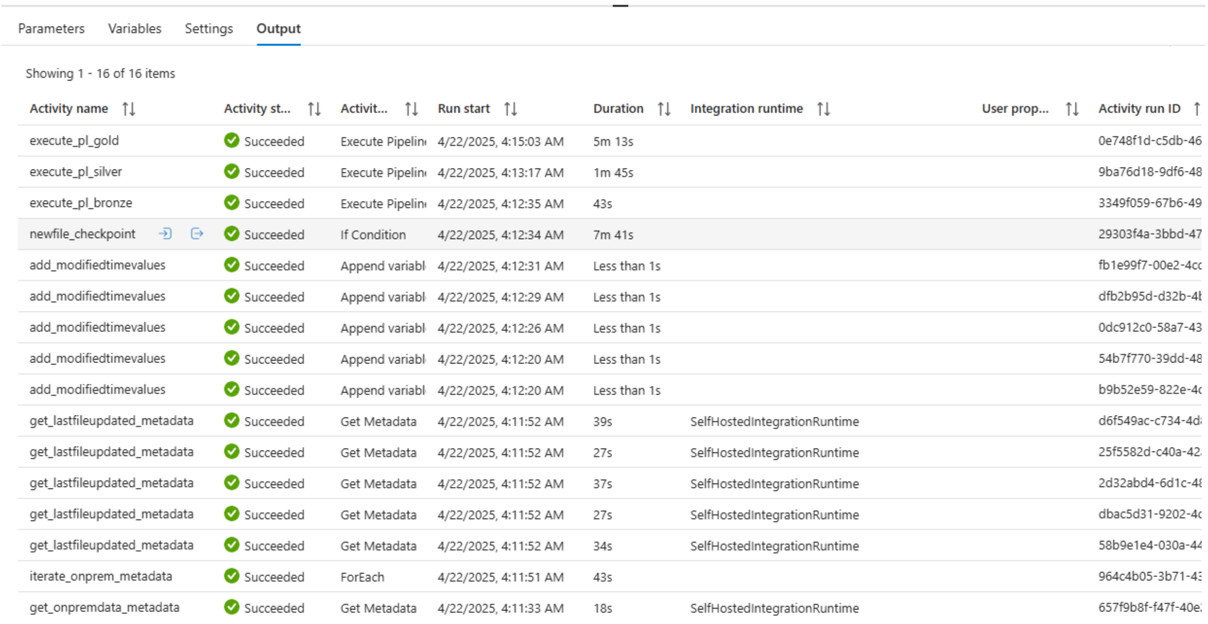


PaymentId	LoanId	PaymentDate	PaymentAmount	CreatedBy	CreatedDate	UpdatedBy	Upd
No results							



AccountId	CustomerId	AccountType	Balance	CreatedBy	CreatedDate	UpdatedBy	Upd
No results							

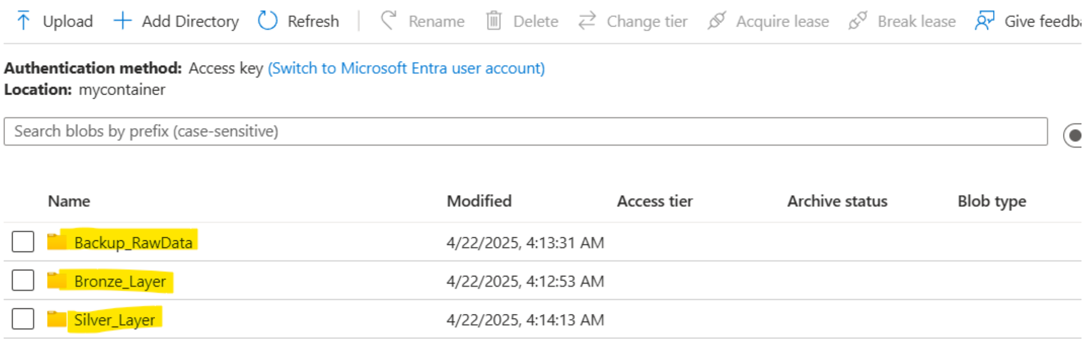
– Pipeline Execution Logs.



Showing 1 - 16 of 16 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
execute_pl_gold	Succeeded	Execute Pipelin	4/22/2025, 4:15:03 AM	5m 13s			0e748f1d-c5db-46
execute_pl_silver	Succeeded	Execute Pipelin	4/22/2025, 4:13:17 AM	1m 45s			9ba76d18-9df6-48
execute_pl_bronze	Succeeded	Execute Pipelin	4/22/2025, 4:12:35 AM	43s			3349f059-67b6-49
newfile_checkpoint	Succeeded	If Condition	4/22/2025, 4:12:34 AM	7m 41s			29303f4a-3bbd-47
add_modifiedtimevalues	Succeeded	Append variabl	4/22/2025, 4:12:31 AM	Less than 1s			fb1e99f7-00e2-4cc
add_modifiedtimevalues	Succeeded	Append variabl	4/22/2025, 4:12:29 AM	Less than 1s			dfb2b95d-d32b-4b
add_modifiedtimevalues	Succeeded	Append variabl	4/22/2025, 4:12:26 AM	Less than 1s			0dc912c0-58a7-43
add_modifiedtimevalues	Succeeded	Append variabl	4/22/2025, 4:12:20 AM	Less than 1s			54b7f770-39dd-48
add_modifiedtimevalues	Succeeded	Append variabl	4/22/2025, 4:12:20 AM	Less than 1s			b9b52e59-822e-4c
get_lastfileupdated_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:52 AM	39s	SelfHostedIntegrationRuntime		d6f549ac-c734-4d
get_lastfileupdated_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:52 AM	27s	SelfHostedIntegrationRuntime		25f5582d-c40a-42
get_lastfileupdated_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:52 AM	37s	SelfHostedIntegrationRuntime		2d32abd4-6d1c-48
get_lastfileupdated_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:52 AM	27s	SelfHostedIntegrationRuntime		dbac5d31-9202-4c
get_lastfileupdated_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:52 AM	34s	SelfHostedIntegrationRuntime		58b9e1e4-030a-44
iterate_onprem_metadata	Succeeded	ForEach	4/22/2025, 4:11:51 AM	43s			964c4b05-3b71-43
get_onpremdata_metadata	Succeeded	Get Metadata	4/22/2025, 4:11:33 AM	18s	SelfHostedIntegrationRuntime		657f9b8f-f47f-40e

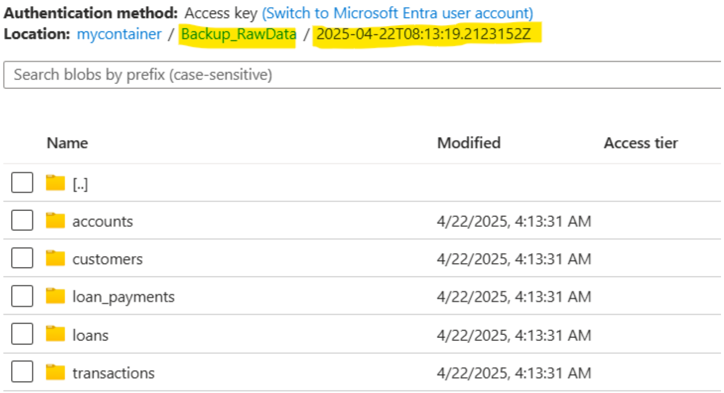
– The ADLS gen2 Storage container after pipelines execution.



Authentication method: Access key (Switch to Microsoft Entra user account)
Location: mycontainer

	Name	Modified	Access tier	Archive status	Blob type
□	Backup_RawData	4/22/2025, 4:13:31 AM			
□	Bronze_Layer	4/22/2025, 4:12:53 AM			
□	Silver_Layer	4/22/2025, 4:14:13 AM			

– The backup folder review.

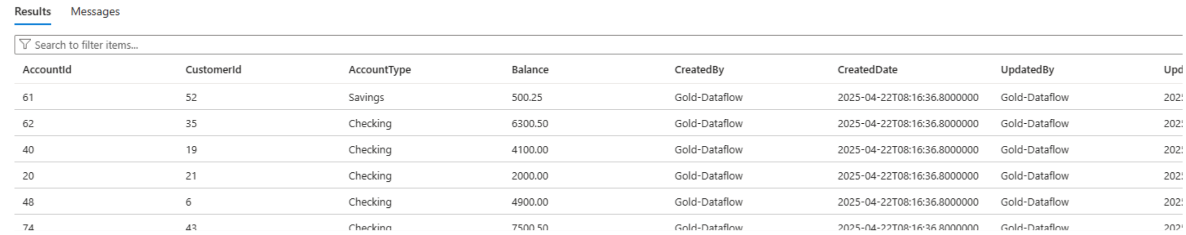


Authentication method: Access key (Switch to Microsoft Entra user account)
Location: mycontainer / Backup_RawData / 2025-04-22T08:13:19.2123152Z

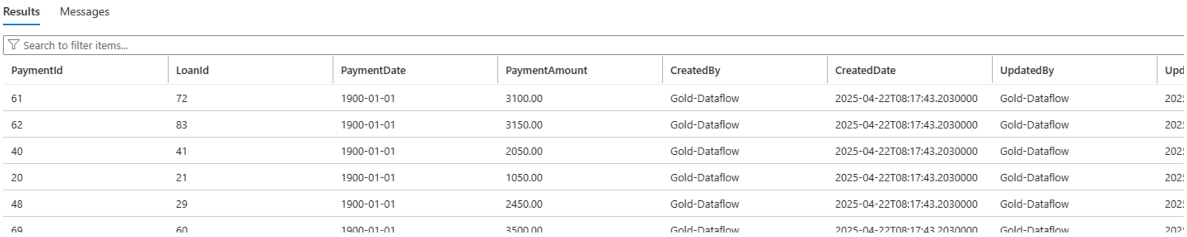
Search blobs by prefix (case-sensitive)

	Name	Modified	Access tier
□	[..]		
□	accounts	4/22/2025, 4:13:31 AM	
□	customers	4/22/2025, 4:13:31 AM	
□	loan_payments	4/22/2025, 4:13:31 AM	
□	loans	4/22/2025, 4:13:31 AM	
□	transactions	4/22/2025, 4:13:31 AM	

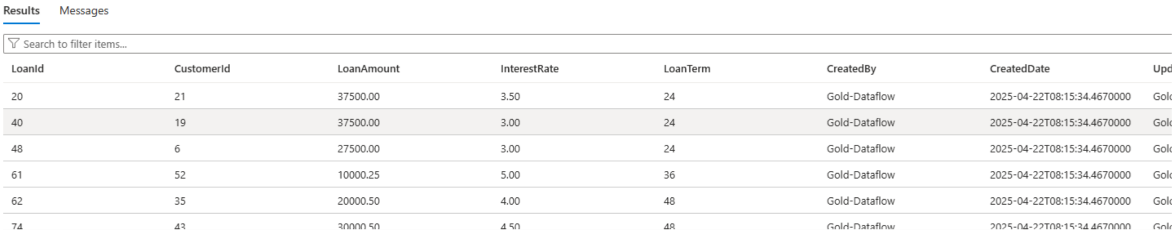
– The Azure SQL DB all 5 Tables output after pipelines execution.



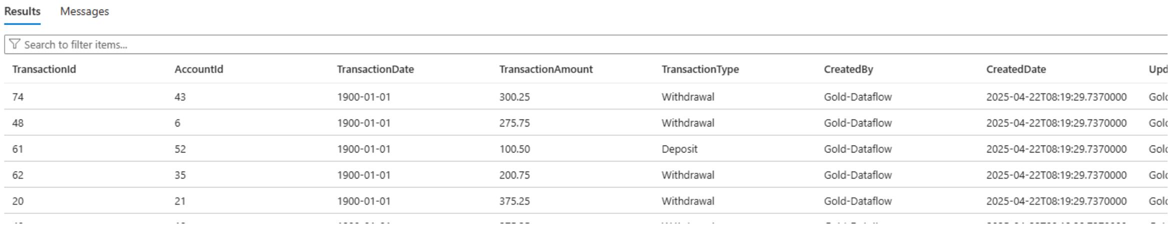
AccountId	CustomerId	AccountType	Balance	CreatedBy	CreatedDate	UpdatedBy	Upd
61	52	Savings	500.25	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202
62	35	Checking	6300.50	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202
40	19	Checking	4100.00	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202
20	21	Checking	2000.00	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202
48	6	Checking	4900.00	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202
74	43	Checking	7500.50	Gold-Dataflow	2025-04-22T08:16:36.8000000	Gold-Dataflow	202



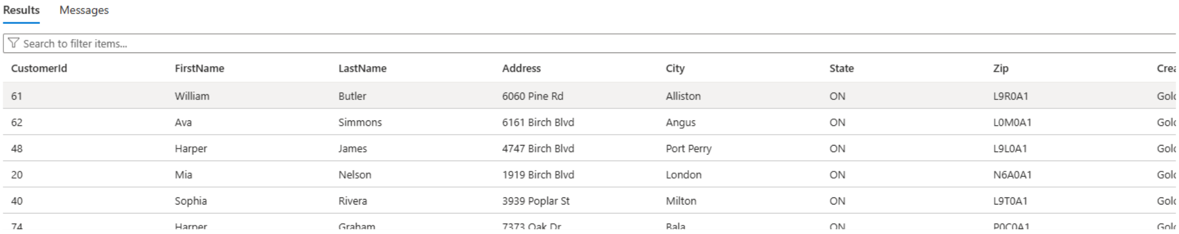
PaymentId	LoanId	PaymentDate	PaymentAmount	CreatedBy	CreatedDate	UpdatedBy	Upd
61	72	1900-01-01	3100.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202
62	83	1900-01-01	3150.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202
40	41	1900-01-01	2050.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202
20	21	1900-01-01	1050.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202
48	29	1900-01-01	2450.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202
69	60	1900-01-01	3500.00	Gold-Dataflow	2025-04-22T08:17:43.2030000	Gold-Dataflow	202



LoanId	CustomerId	LoanAmount	InterestRate	LoanTerm	CreatedBy	CreatedDate	Upd
20	21	37500.00	3.50	24	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol
40	19	37500.00	3.00	24	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol
48	6	27500.00	3.00	24	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol
61	52	10000.25	5.00	36	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol
62	35	20000.50	4.00	48	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol
74	43	30000.50	4.50	48	Gold-Dataflow	2025-04-22T08:15:34.4670000	Gol



TransactionId	AccountId	TransactionDate	TransactionAmount	TransactionType	CreatedBy	CreatedDate	Upd
74	43	1900-01-01	300.25	Withdrawal	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gol
48	6	1900-01-01	275.75	Withdrawal	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gol
61	52	1900-01-01	100.50	Deposit	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gol
62	35	1900-01-01	200.75	Withdrawal	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gol
20	21	1900-01-01	375.25	Withdrawal	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gol
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

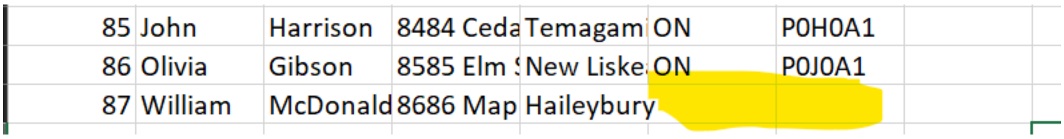


CustomerId	FirstName	LastName	Address	City	State	Zip	Cre
61	William	Butler	6060 Pine Rd	Alliston	ON	L9R0A1	Gol
62	Ava	Simmons	6161 Birch Blvd	Angus	ON	L0M0A1	Gol
48	Harper	James	4747 Birch Blvd	Port Perry	ON	L9L0A1	Gol
20	Mia	Nelson	1919 Birch Blvd	London	ON	N6A0A1	Gol
40	Sophia	Rivera	3939 Poplar St	Milton	ON	L9T0A1	Gol
74	Harper	Graham	7373 Oak Dr	Bala	ON	P0C0A1	Gol

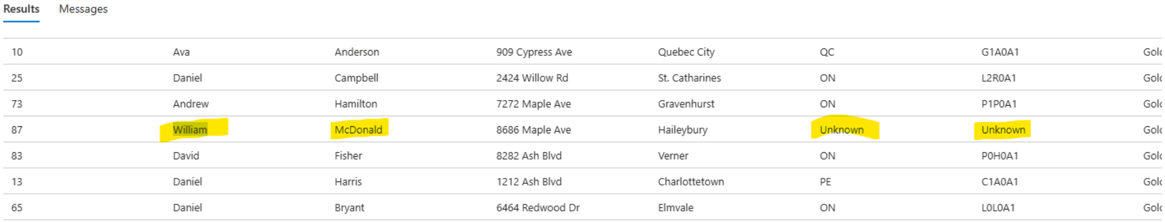
- ✓ **Master Pipeline Second Run.**

– Some data **updates** in **customers and transactions** csv data file at on-prem local storage.

- customers csv data before updates.

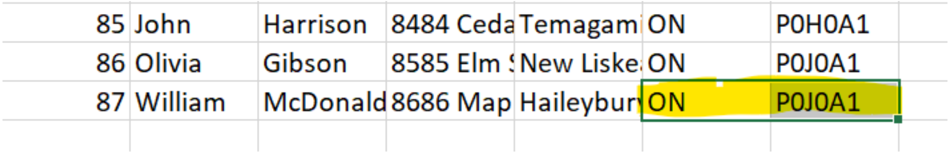


85	John	Harrison	8484 Ceda	Temagami	ON	P0H0A1
86	Olivia	Gibson	8585 Elm S	New Liske	ON	P0J0A1
87	William	McDonald	8686 Map	Haileybury		



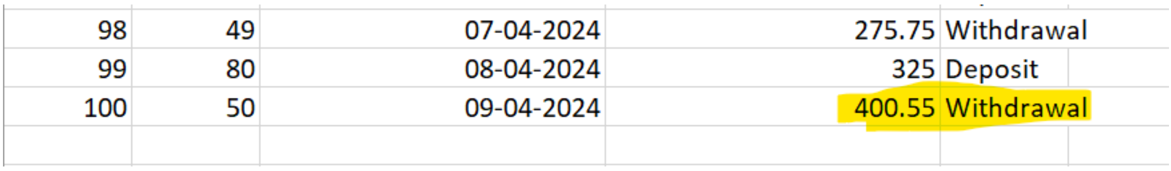
10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1	Gol
25	Daniel	Campbell	2424 Willow Rd	St. Catharines	ON	L2R0A1	Gol
73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1	Gol
87	William	McDonald	8686 Maple Ave	Haileybury	Unknown	Unknown	Gol
83	David	Fisher	8282 Ash Blvd	Verner	ON	P0H0A1	Gol
13	Daniel	Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1	Gol
65	Daniel	Bryant	6464 Redwood Dr	Elmvale	ON	L0L0A1	Gol

- customers csv data after updates.



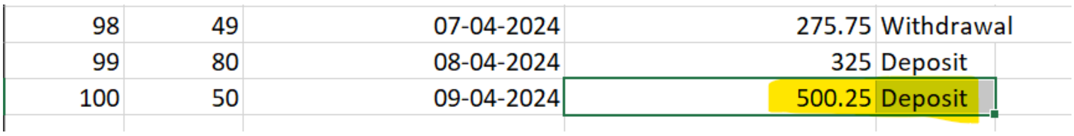
85	John	Harrison	8484 Ceda	Temagami	ON	P0H0A1
86	Olivia	Gibson	8585 Elm S	New Liske	ON	P0J0A1
87	William	McDonald	8686 Map	Haileybury	ON	P0J0A1

- transactions csv data before updates.



98	49	07-04-2024	275.75	Withdrawal
99	80	08-04-2024	325	Deposit
100	50	09-04-2024	400.55	Withdrawal

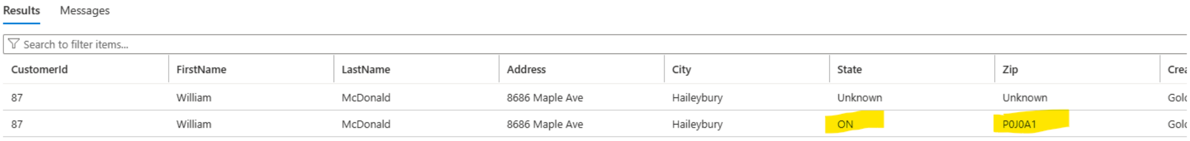
- transactions csv data after updates.



98	49	07-04-2024	275.75	Withdrawal
99	80	08-04-2024	325	Deposit
100	50	09-04-2024	500.25	Deposit

- Data **updates** in **customers and transactions** tables in **Azure SQL DB** after **successful Execution of master and all other pipelines**.

  - customers table data after pipelines execution.

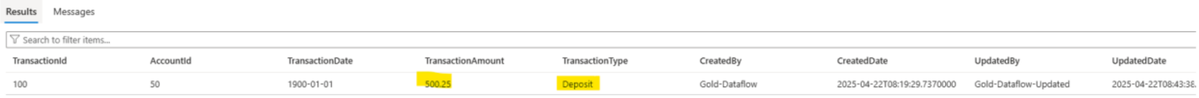


Results Messages

Search to filter items...

CustomerId	FirstName	LastName	Address	City	State	Zip	Crea
87	William	McDonald	8686 Maple Ave	Haileybury	Unknown	Unknown	Gol
87	William	McDonald	8686 Maple Ave	Haileybury	ON	P0J0A1	Gol

  - transactions table data after pipelines execution.



Results Messages

Search to filter items...

TransactionId	AccountId	TransactionDate	TransactionAmount	TransactionType	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate
100	50	1900-01-01	500.25	Deposit	Gold-Dataflow	2025-04-22T08:19:29.7370000	Gold-Dataflow-Updated	2025-04-22T08:43:38.

Power BI Report

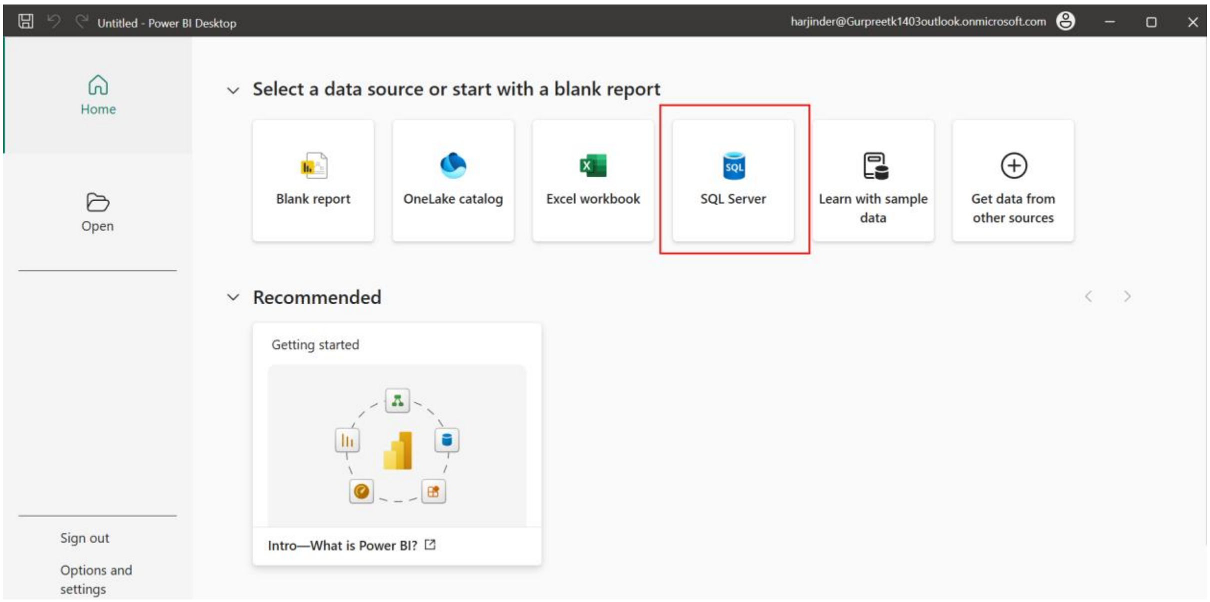
Here, I have used Power BI desktop Application to create and publish the report using gold Layer data (Azure SQL DB Table data) of all 5 tables.

- accounts.csv
- customers.csv
- loan_payments.csv
- loans.csv
- transactions.csv

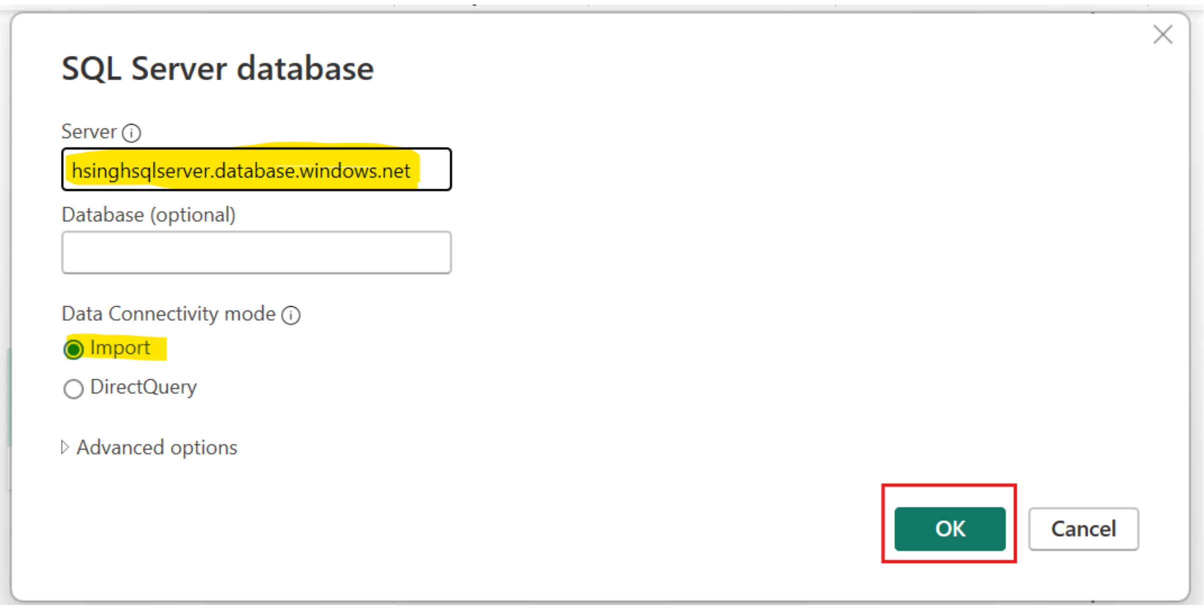
➢ Steps To Create and Publish the Power BI report in Microsoft Fabric workspace.

Step 1: Install and open the Power BI desktop applications -> login with your current working azure portal credentials. (make sure the same credentials are used to login in Fabric account in order to access workspace).

Next, select SQL server as data Source.



Step 2: Next, enter the SQL server name (you can find it in Azure SQL DB overview page in Azure Portal) -> select Import as Data Connectivity mode. -> click on Ok.



SQL Server database

Server ⓘ

hsinghsqlserver.database.windows.net

Database (optional)

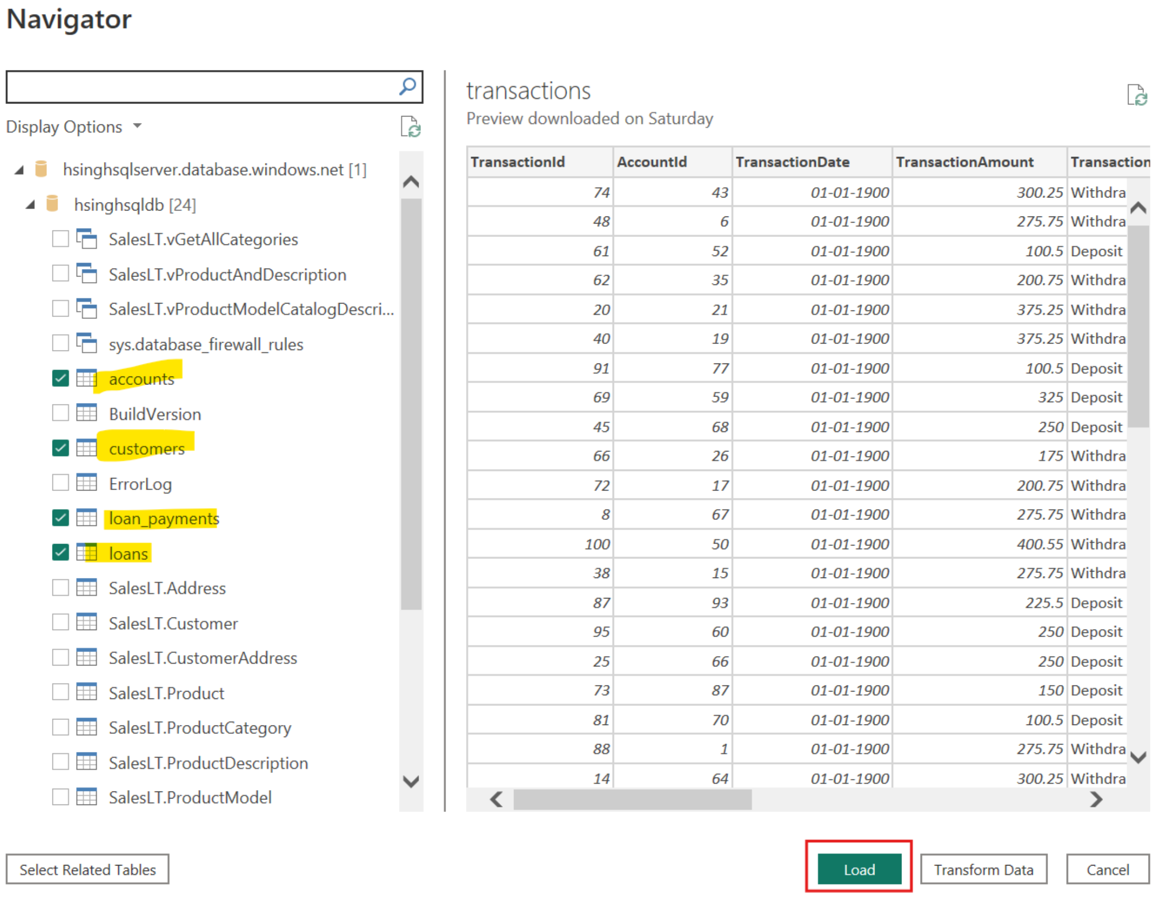
Data Connectivity mode ⓘ

● Import

○ DirectQuery

▷ Advanced options

Step 3: Next, select the data tables from the db -> click on Load.



Navigator

Display Options ▾

◢ hsinghsqlserver.database.windows.net [1]

◢ hsinghsqldb [24]

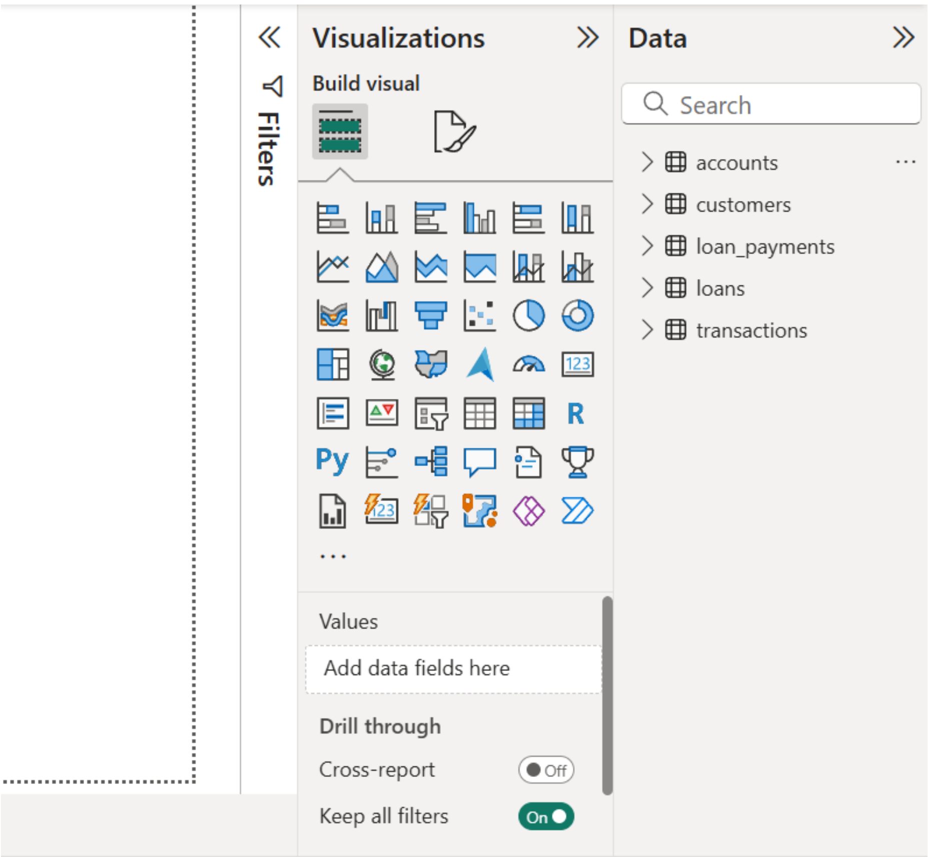
- □ SalesLT.vGetAllCategories
- □ SalesLT.vProductAndDescription
- □ SalesLT.vProductModelCatalogDescri...
- □ sys.database_firewall_rules
- ☑ accounts
- □ BuildVersion
- ☑ customers
- □ ErrorLog
- ☑ loan_payments
- ☑ loans
- □ SalesLT.Address
- □ SalesLT.Customer
- □ SalesLT.CustomerAddress
- □ SalesLT.Product
- □ SalesLT.ProductCategory
- □ SalesLT.ProductDescription
- □ SalesLT.ProductModel

transactions

Preview downloaded on Saturday

TransactionId	AccountId	TransactionDate	TransactionAmount	Transaction
74	43	01-01-1900	300.25	Withdra
48	6	01-01-1900	275.75	Withdra
61	52	01-01-1900	100.5	Deposit
62	35	01-01-1900	200.75	Withdra
20	21	01-01-1900	375.25	Withdra
40	19	01-01-1900	375.25	Withdra
91	77	01-01-1900	100.5	Deposit
69	59	01-01-1900	325	Deposit
45	68	01-01-1900	250	Deposit
66	26	01-01-1900	175	Withdra
72	17	01-01-1900	200.75	Withdra
8	67	01-01-1900	275.75	Withdra
100	50	01-01-1900	400.55	Withdra
38	15	01-01-1900	275.75	Withdra
87	93	01-01-1900	225.5	Deposit
95	60	01-01-1900	250	Deposit
25	66	01-01-1900	250	Deposit
73	87	01-01-1900	150	Deposit
81	70	01-01-1900	100.5	Deposit
88	1	01-01-1900	275.75	Withdra
14	64	01-01-1900	300.25	Withdra

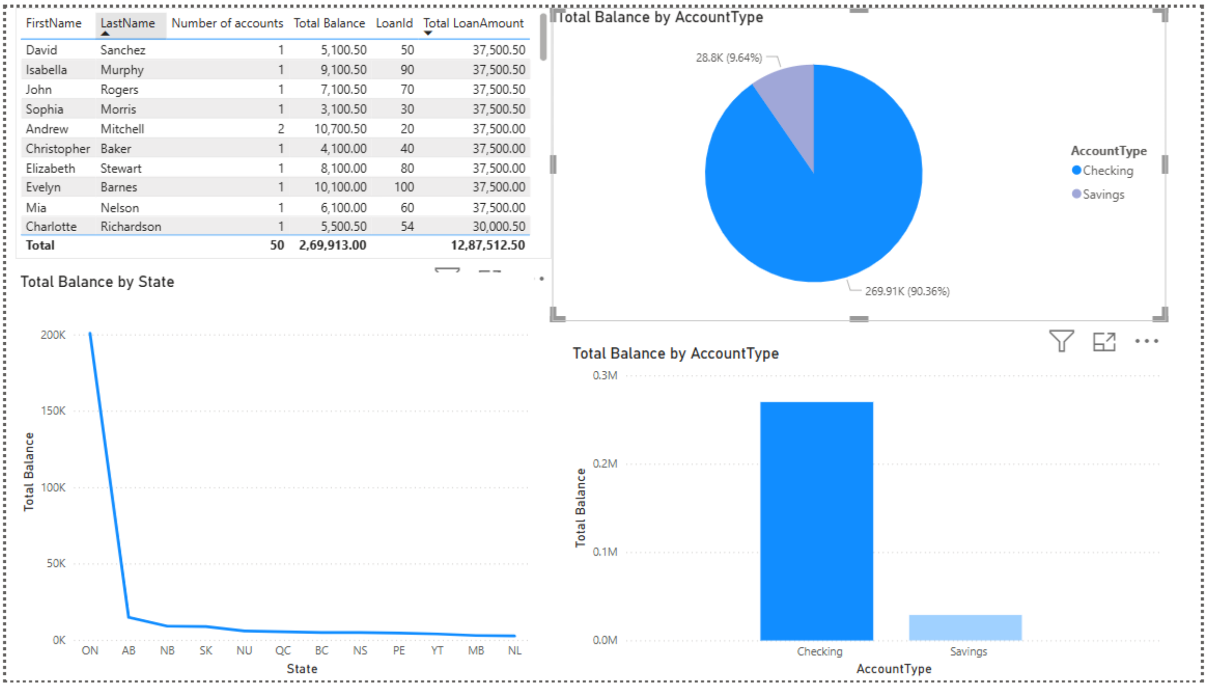
Step 4: Next, select the data columns to visualize the data from data section.



Cross-report ○ Off

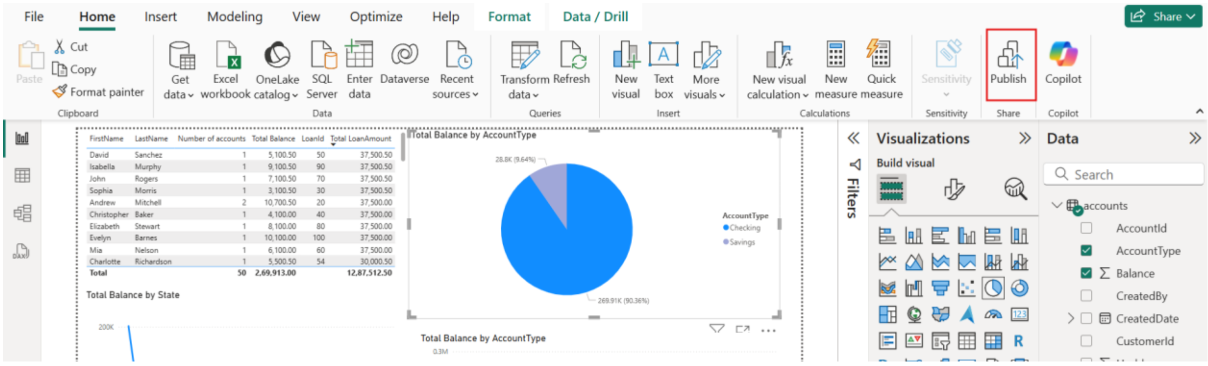
Keep all filters On ●

Step 5: Next, create some visuals and report by using data columns.



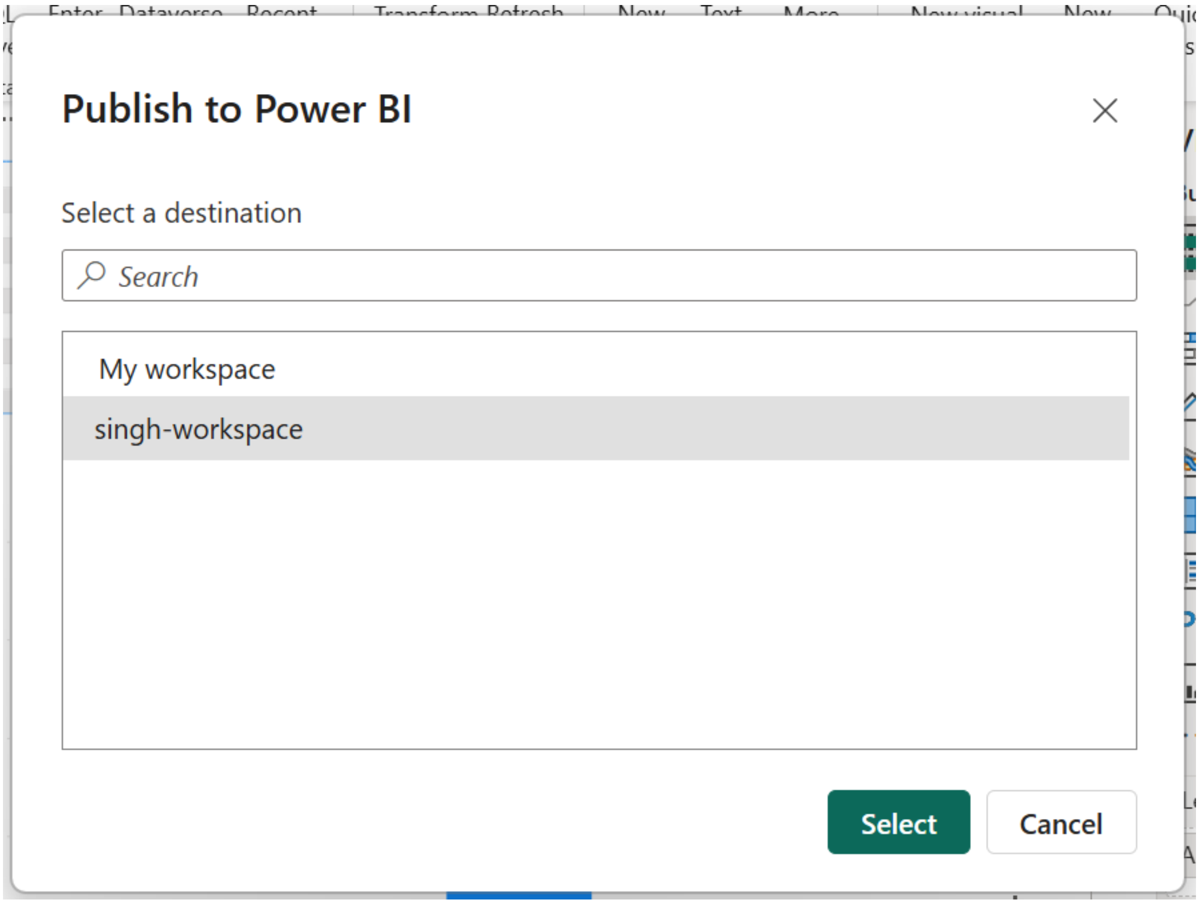
FirstName	LastName	Number of accounts	Total Balance	LoanId	Total LoanAmount
David	Sanchez	1	5,100.50	50	37,500.50
Isabella	Murphy	1	9,100.50	90	37,500.50
John	Rogers	1	7,100.50	70	37,500.50
Sophia	Morris	1	3,100.50	30	37,500.50
Andrew	Mitchell	2	10,700.50	20	37,500.00
Christopher	Baker	1	4,100.00	40	37,500.00
Elizabeth	Stewart	1	8,100.00	80	37,500.00
Evelyn	Barnes	1	10,100.00	100	37,500.00
Mia	Nelson	1	6,100.00	60	37,500.00
Charlotte	Richardson	1	5,500.50	54	30,000.50
Total		**50**	**2,69,913.00**		**12,87,512.50**

Step 6: Next, review the report and publish using the publish option above.

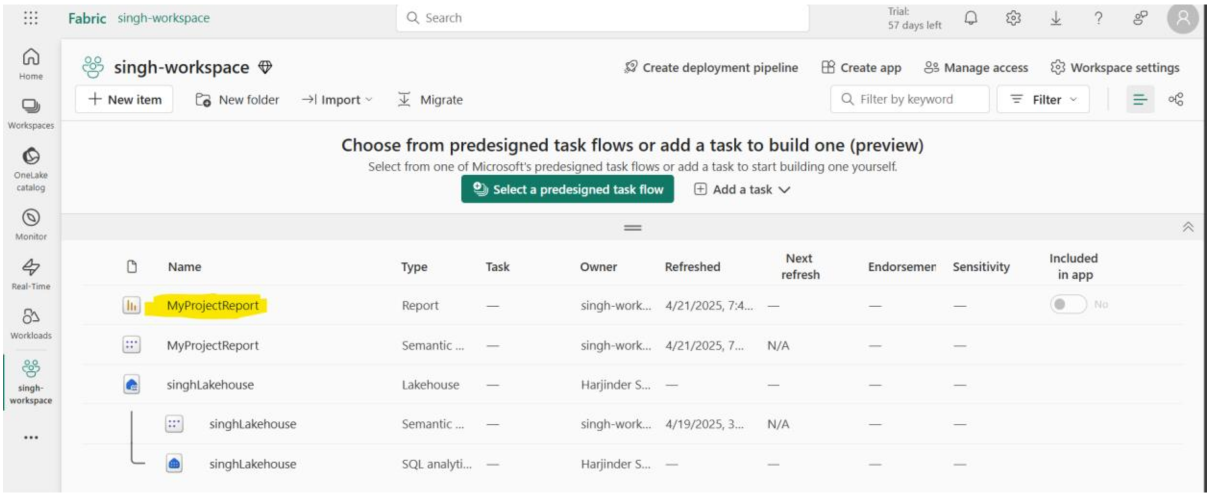


FirstName	LastName	Number of accounts	Total Balance	LoanId	Total LoanAmount
David	Sanchez	1	5,100.50	50	37,500.50
Isabella	Murphy	1	9,100.50	90	37,500.50
John	Rogers	1	7,100.50	70	37,500.50
Sophia	Morris	1	3,100.50	30	37,500.50
Andrew	Mitchell	2	10,700.50	20	37,500.00
Christopher	Baker	1	4,100.00	40	37,500.00
Elizabeth	Stewart	1	8,100.00	80	37,500.00
Evelyn	Barnes	1	10,100.00	100	37,500.00
Mia	Nelson	1	6,100.00	60	37,500.00
Charlotte	Richardson	1	5,500.50	54	30,000.50
Total		50	2,69,913.00		12,87,512.50

Step 7: Next, select the fabric workspace to publish the power BI report -> click on select.



Step 7: Finally, review the published report in the fabric workspace.



Choose from predesigned task flows or add a task to build one (preview)

Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.

Name	Type	Task	Owner	Refreshed	Next refresh	Endorsemen	Sensitivity	Included in app
MyProjectReport	Report	—	singh-work...	4/21/2025, 7:4...	—	—	—	No
MyProjectReport	Semantic ...	—	singh-work...	4/21/2025, 7...	N/A	—	—	
singhLakehouse	Lakehouse	—	Harjinder S...	—	—	—	—	
singhLakehouse	Semantic ...	—	singh-work...	4/19/2025, 3...	N/A	—	—	
singhLakehouse	SQL analyti...	—	Harjinder S...	—	—	—	—	