Data Engineering Project-1:

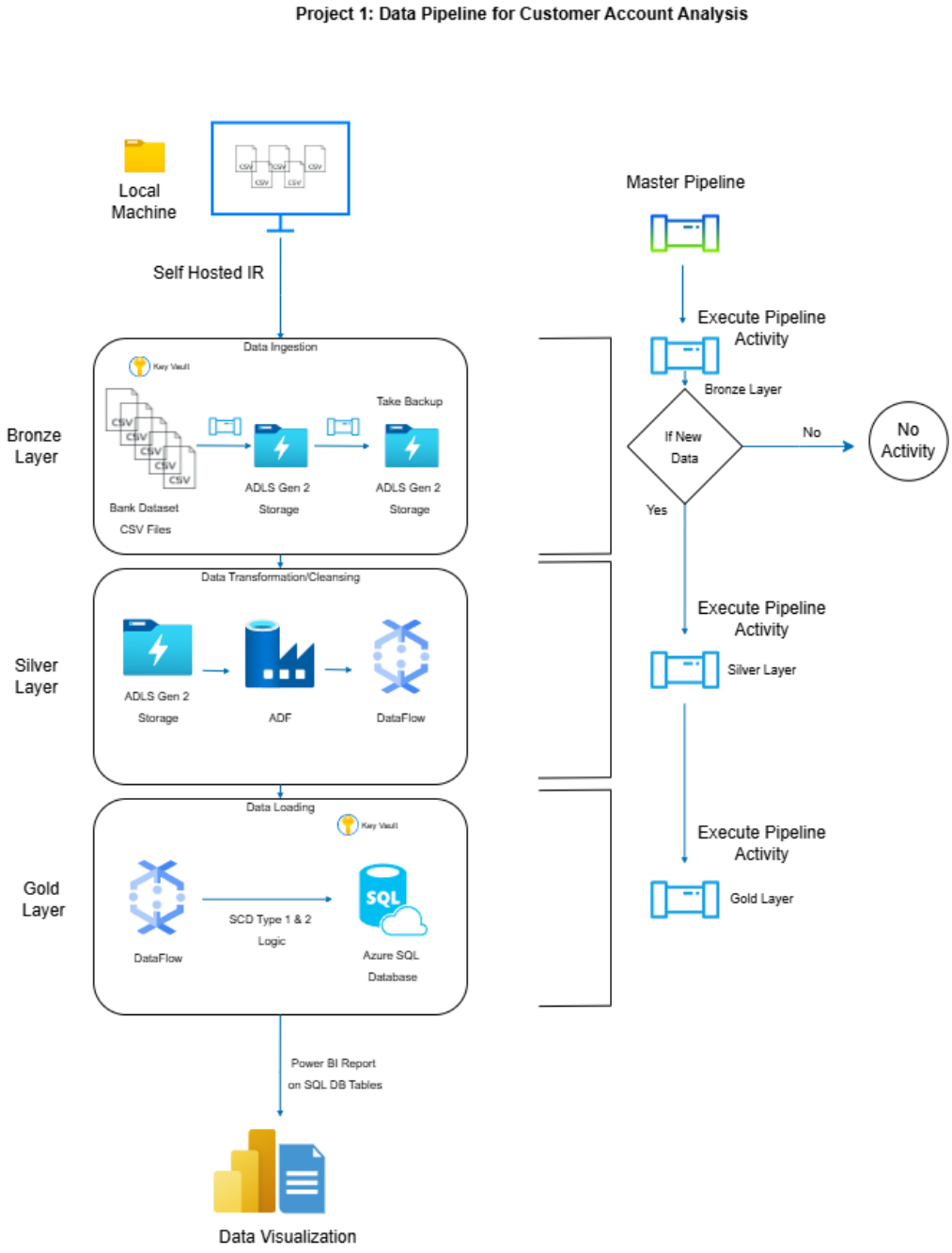
Data Pipeline for Customer Account Analysis

Table of Contents

Objective:

The project aims to design and implement a robust data pipeline for processing customer account data. This includes copying data from a backend team's storage account, performing necessary transformations using ADF and upsertings (inserting or updating) data from a file stored in Azure Data Lake Storage ADLS GOLD Storage into a SQL database table. The pipeline aims to ensure efficient, accurate, and scalable data processing to support downstream analytics and reporting needs.

Project Flow Diagram:



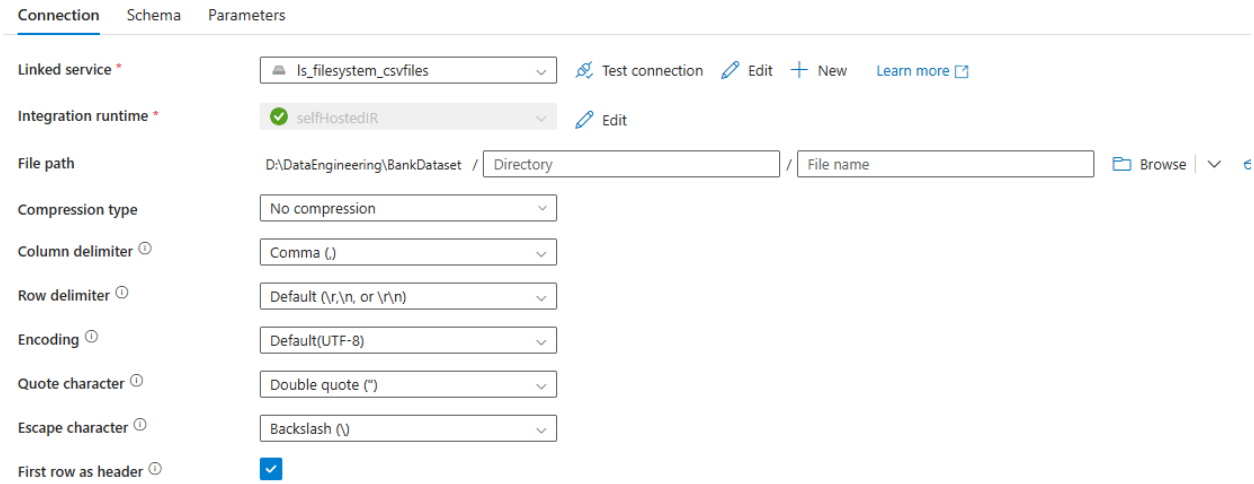
Step 1: Data Ingestion (Backend Storage to Raw(Bronze) Container)

Azure Home -> Azure Data Factory -> Author Tab -> New Pipeline

Step 1.1: Get All Available Files (Dynamically discover all available files without hardcoding)

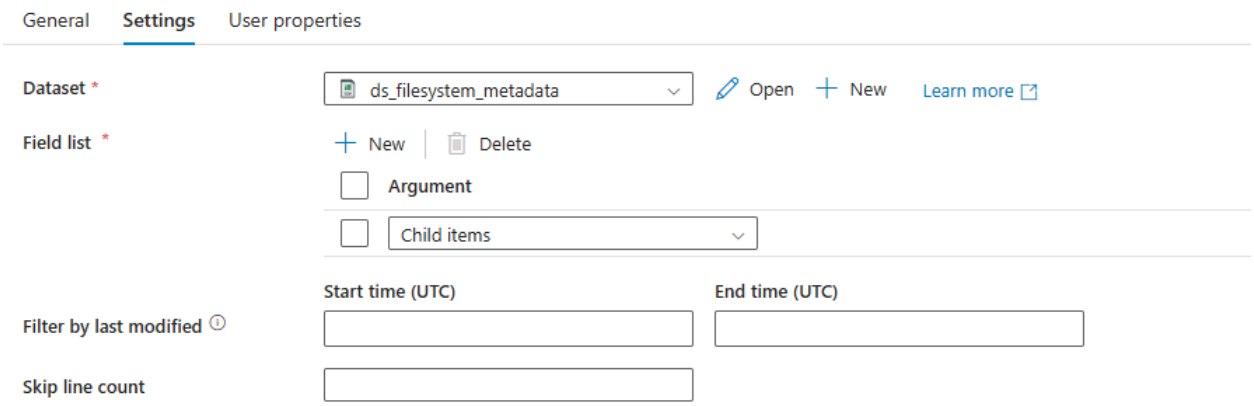
- Use the Get Metadata Activity.
- Source: Local folder via Self-Hosted Integration Runtime (SHIR).
- Fetch: Child Items (all CSV file names from the folder).

Take the Get Metadata Activity first, and create a new data source file, **File System**, as we want to access the local storage folder.



Settings tab in Get Metadata Activity:

Field list: Child item from drop-down options



Step 1.2: Loop Through Each File (Process each file individually but dynamically)

Note: Inside the "Bronze Layer" pipeline, a ForEach activity is used to iterate through a list of your CSV files: 'accounts.csv', 'customers.csv', 'loan_payments.csv', 'loans.csv', and 'transactions.csv'.

This allows for processing multiple files in a single pipeline run.

Search ForEach Activity from the Activities tab and drag it to the design canvas

Write this expression in the ForEach Settings tab:

@activity('Get All Available Files').output.childItems

Click on the Edit icon of the ForEach activity

Step 1.3: Check File Freshness

- Get the Metadata Activity inside ForEach.
- Fetch the Last Modified timestamp of each file.

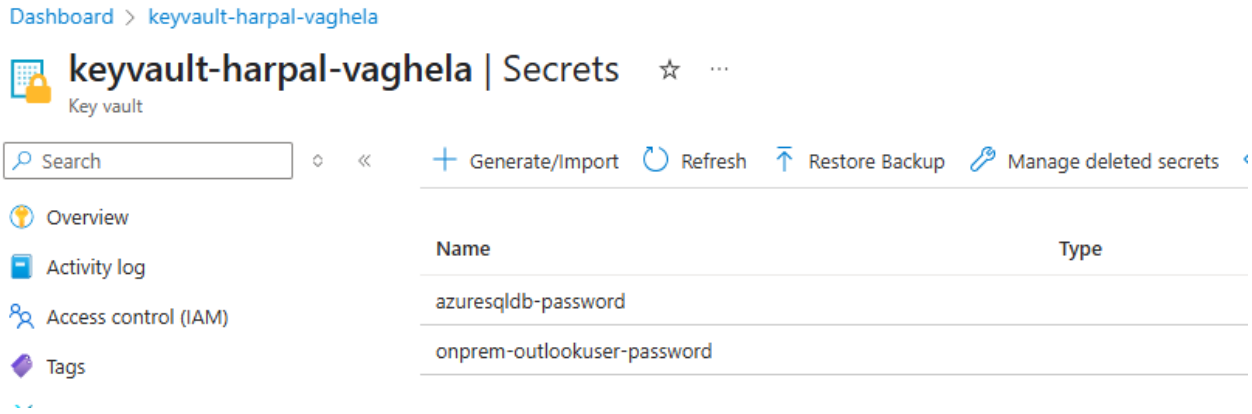
In the dataset, select a file system from all options and create a linked service using a **Self-hosted integration** runtime.

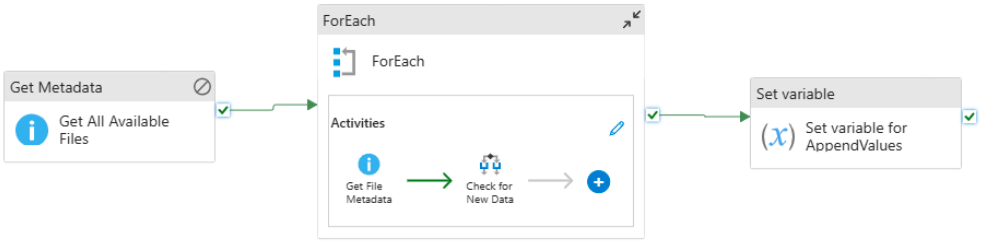
Give the path of the local folder where all the CSV files are stored.

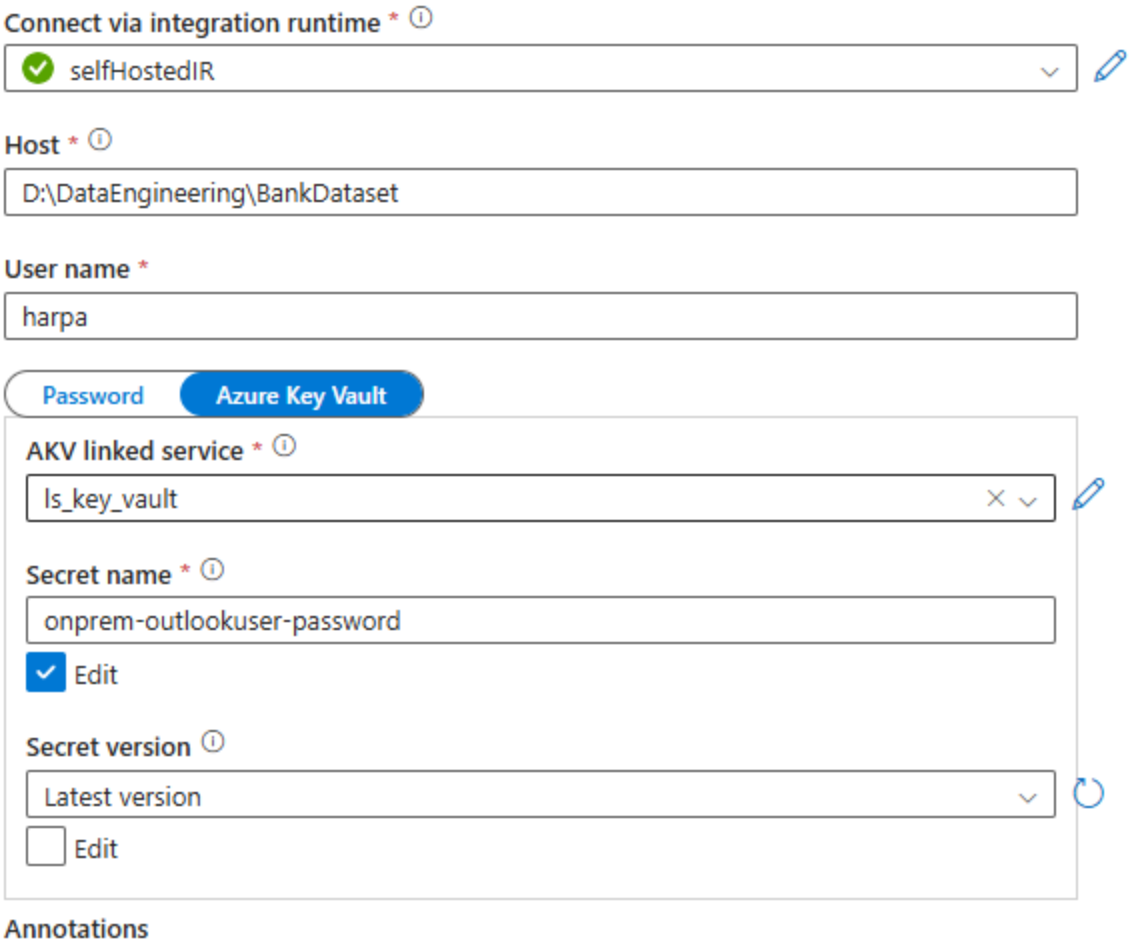
Your username will be your Windows username. if you don't know, then execute the whoami command in the command prompt in Windows.

The password will be your account password attached to your account in Windows or your Windows login password, depending on your Windows user configurations.

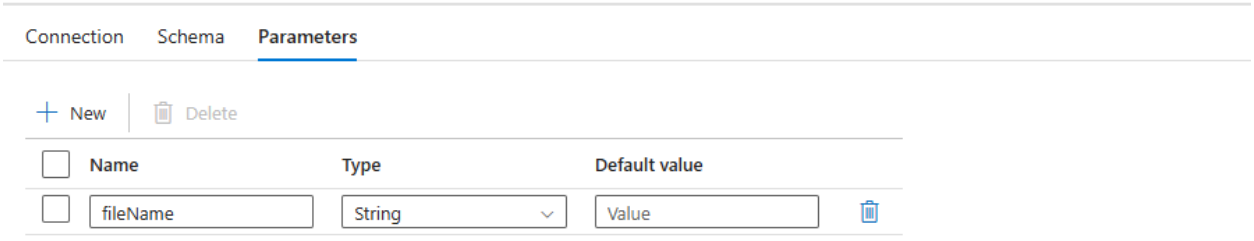
We have stored the password in **Azure Key Vault** and used it in the linked service here in the Get Metadata activity.





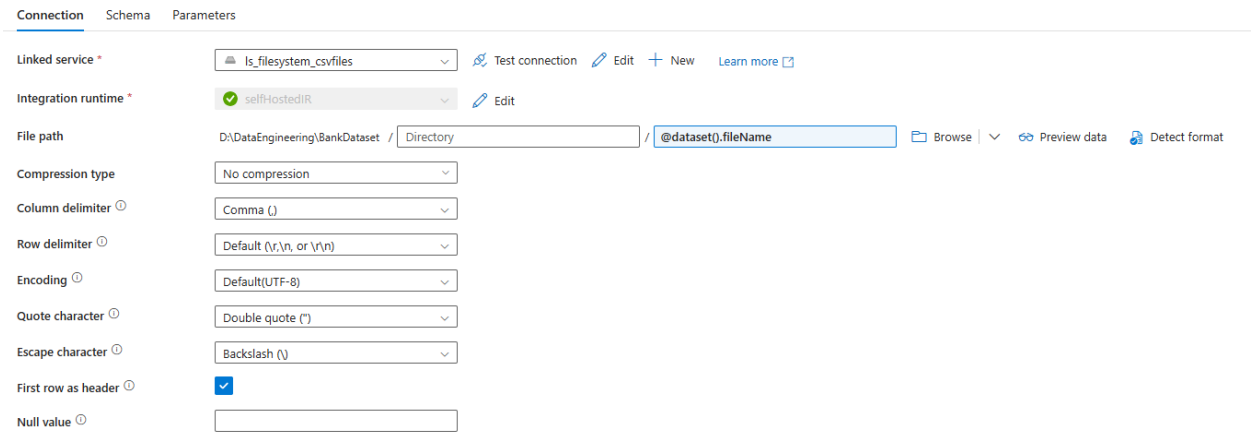


Create one parameter in the parameters tab:

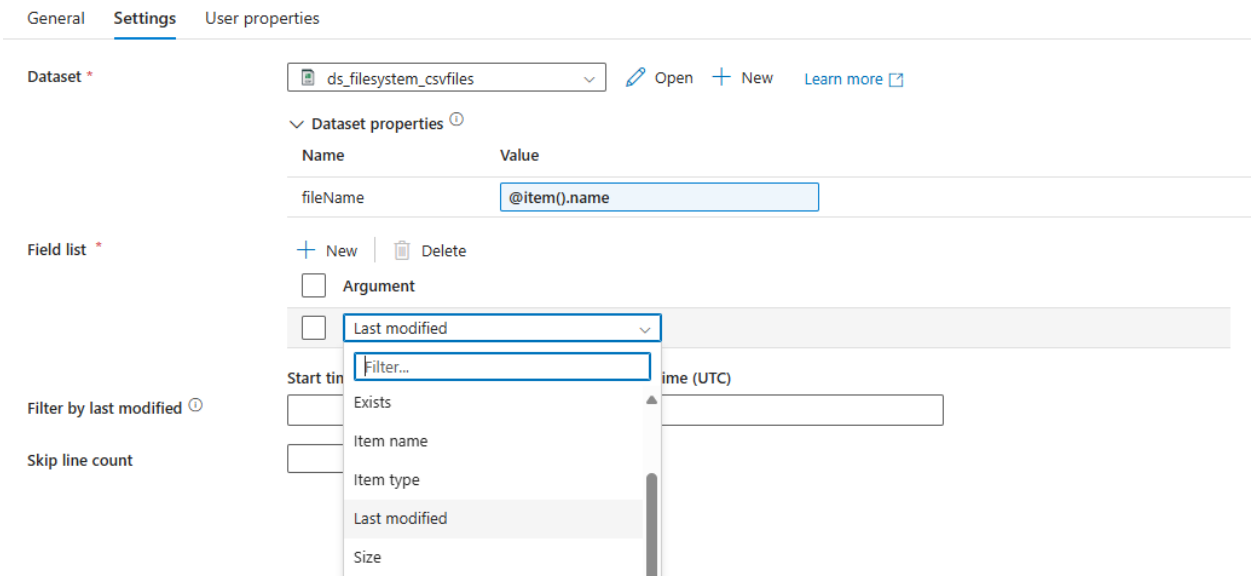


Go to the connection tab:

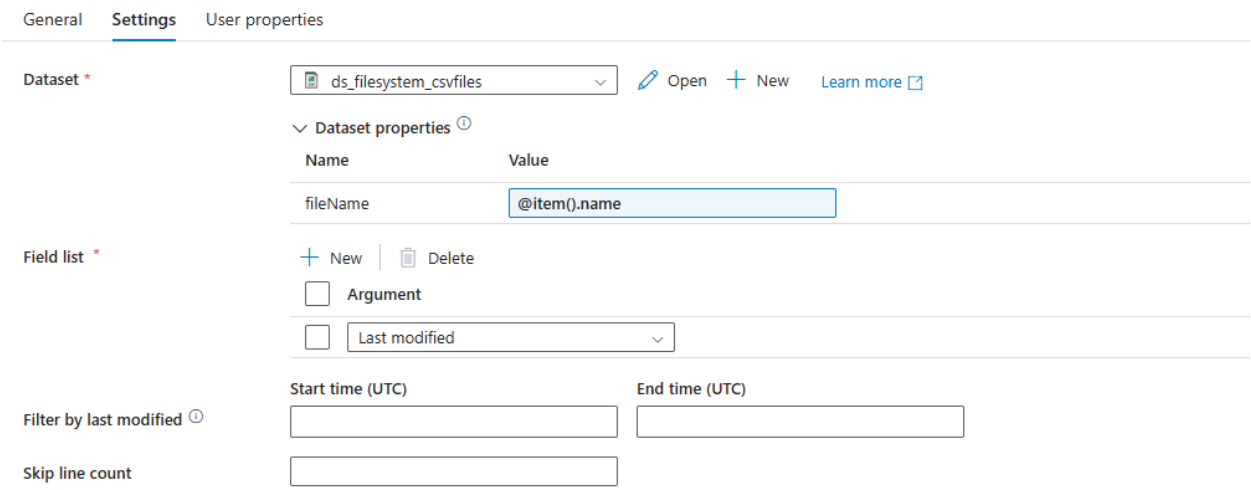
Mention the filename parameter here as shown below:



In Field list options, select Last Modified from the drop down:



Finally Get Metadata Activity's settings tab will look like this:

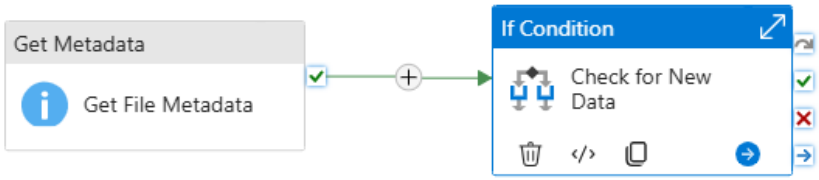


Step 1.4: If Condition (Check If Modified in Last 24 Hours)

Take if condition after the Get Metadata Activity and rename it

Write this expression in an if condition:

@greaterOrEquals(activity('Get File Metadata').output.lastModified, addDays(utcNow(), -1))

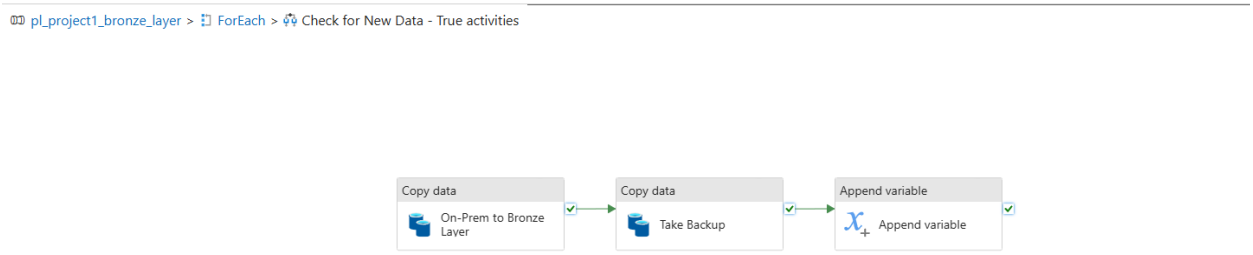


Inside the True part of the If Condition, take 3 activities as

Copy Data (for copying files from local to Bronze layer in ADLS Gen 2 Storage account)

Copy Data (for taking backup of the client's data)

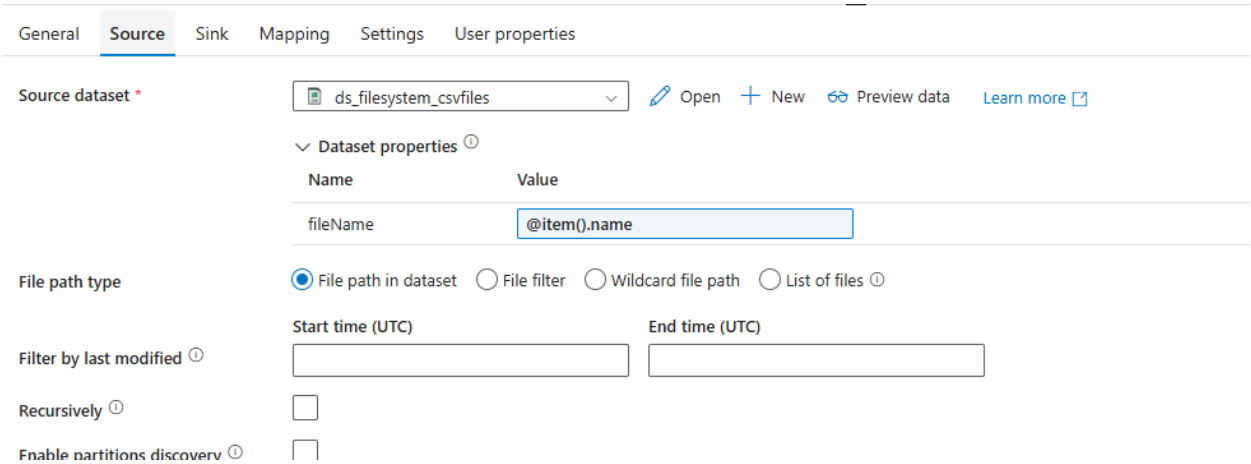
Append Variable activity (for storing the name of files, as we will use this for a condition later on)



Step 1.5: Copy Files to Bronze + Take Backup

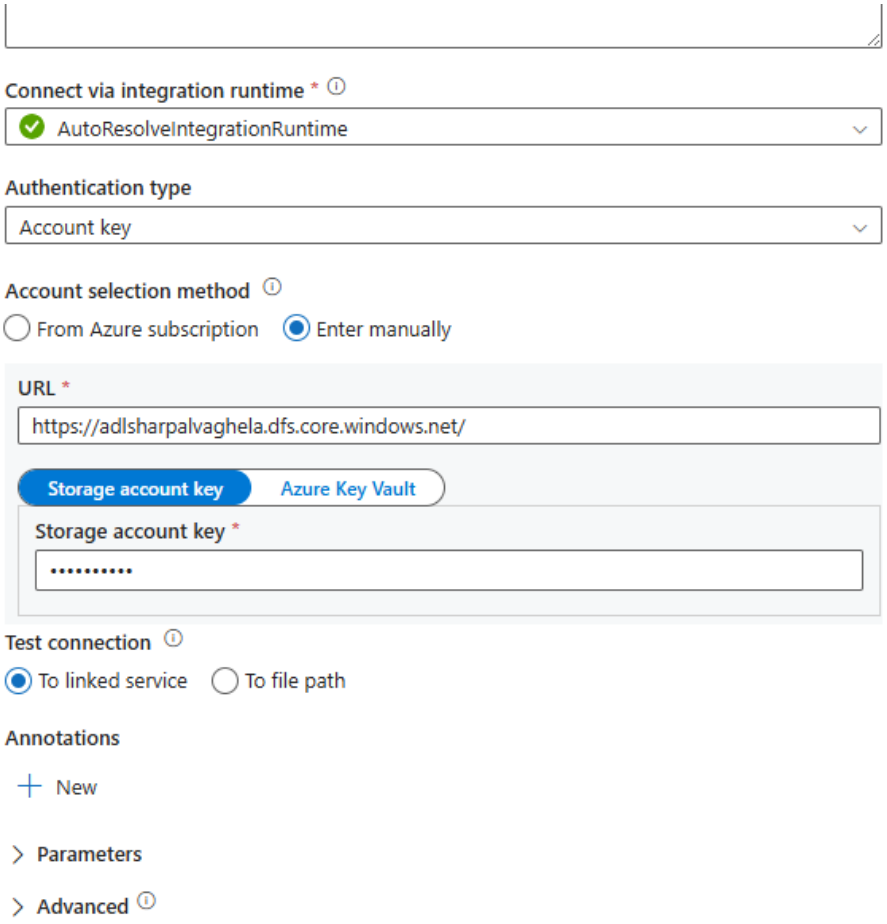
- Copy Data Activity 1: Local → ADLS Gen2 (Bronze folder)
- Copy Data Activity 2: Local → ADLS Gen2 (Backup folder)
- Dynamic folder structure: Add a timestamp to the backup filename.

Use the same dataset that we have used for the Get Metadata activity

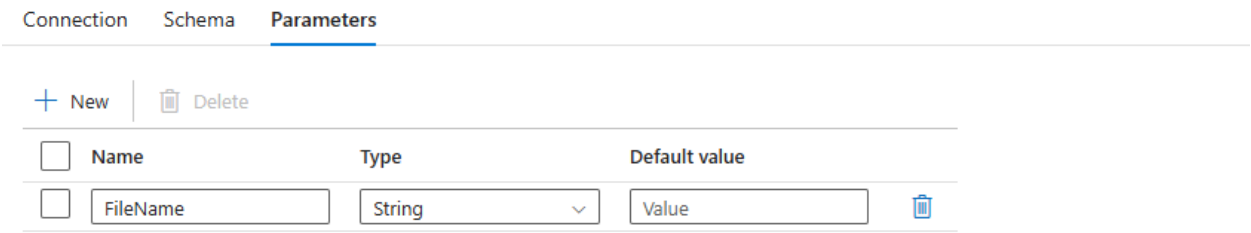


For Sink, we will create a new dataset, as we have to store data in an ADLS Gen 2 storage account.

Link service:

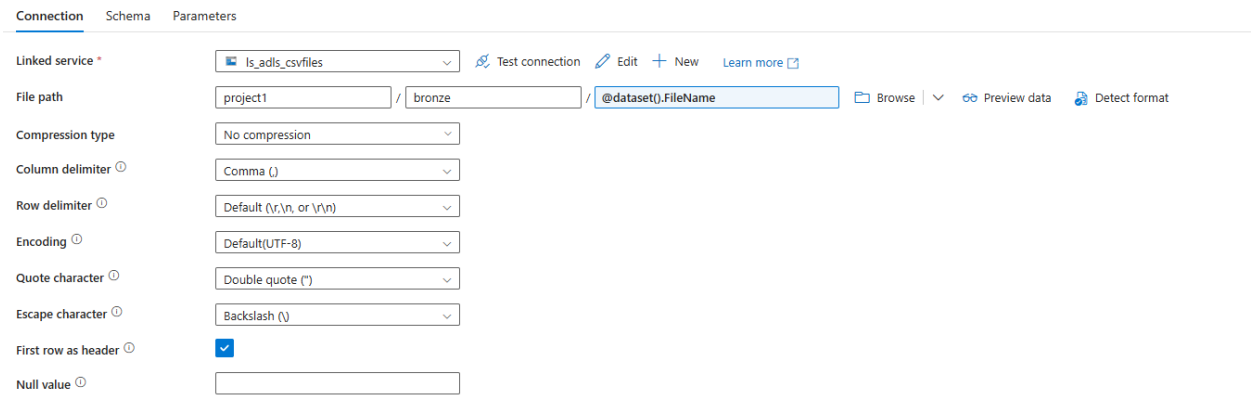


Create one parameter in the parameters tab:



Go to the connection tab:

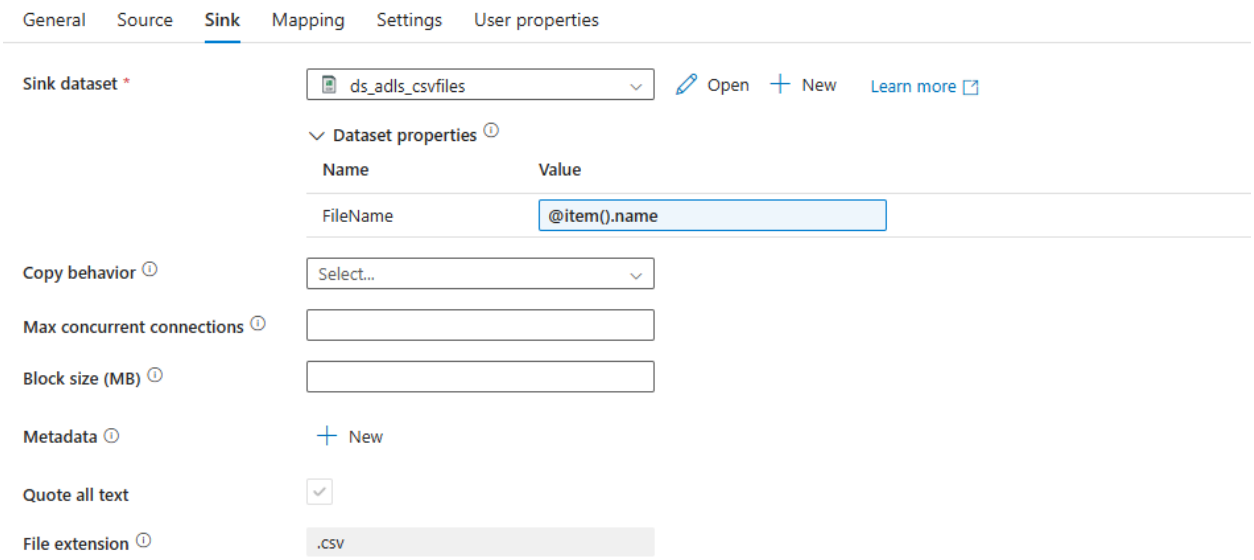
Mention the filename parameter here as shown below:



Here, select the first row as a header, as we have all CSV files with a header.

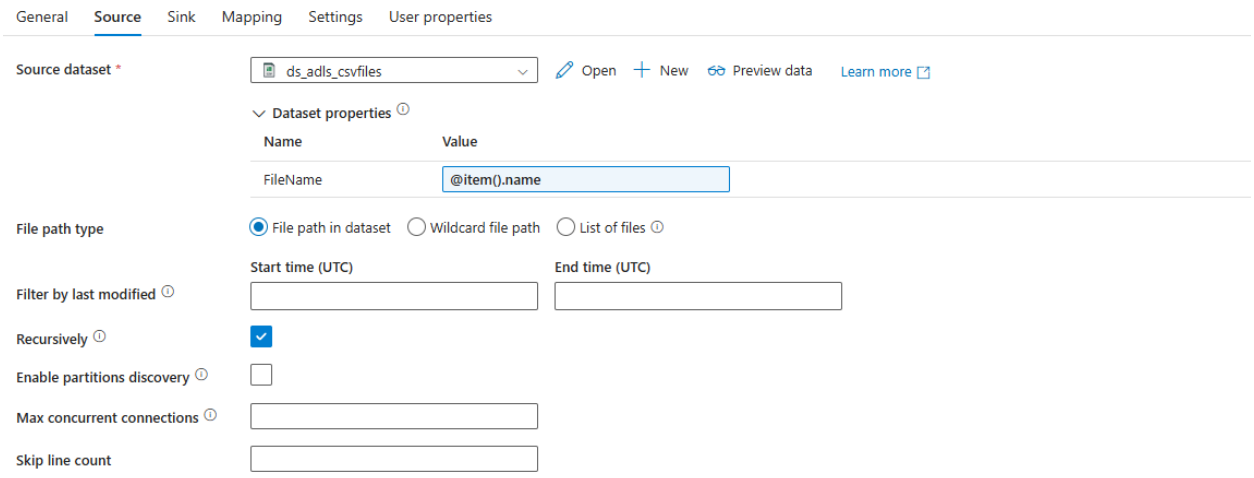
Select the Bronze folder from the ADLS Gen 2 account.

Finally sink part will look like this:



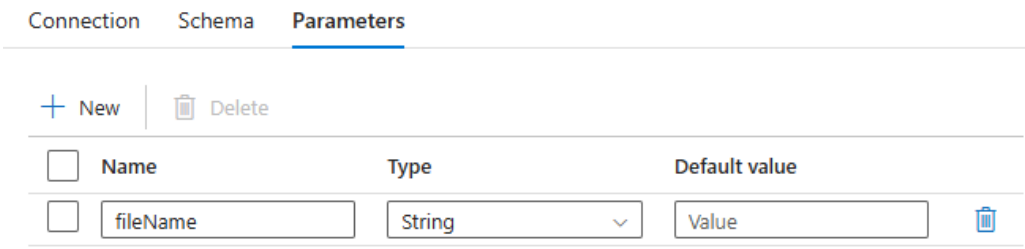
- **Copy Data Activity (For Taking Backup)**

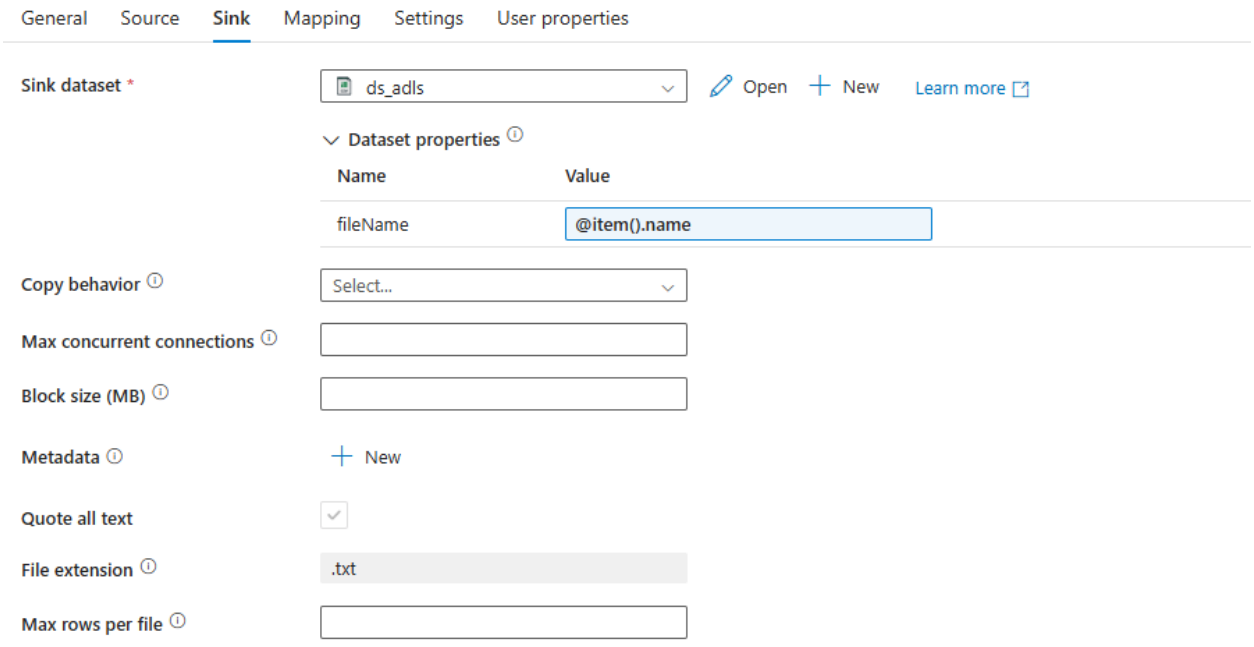
Use the same source dataset as used in the first copy data activity



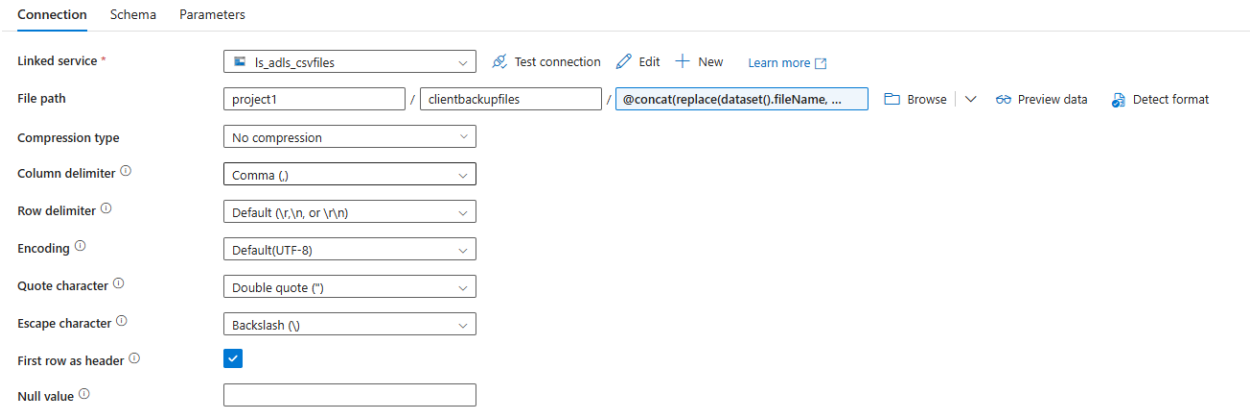
For Sink, create a new data source for the ADLS Gen 2 storage account.

Create one parameter in the parameters tab:





Go to the connection tab and select the clientbackupfiles folder by clicking the browse icon.



Use this expression for the filename as shown below:

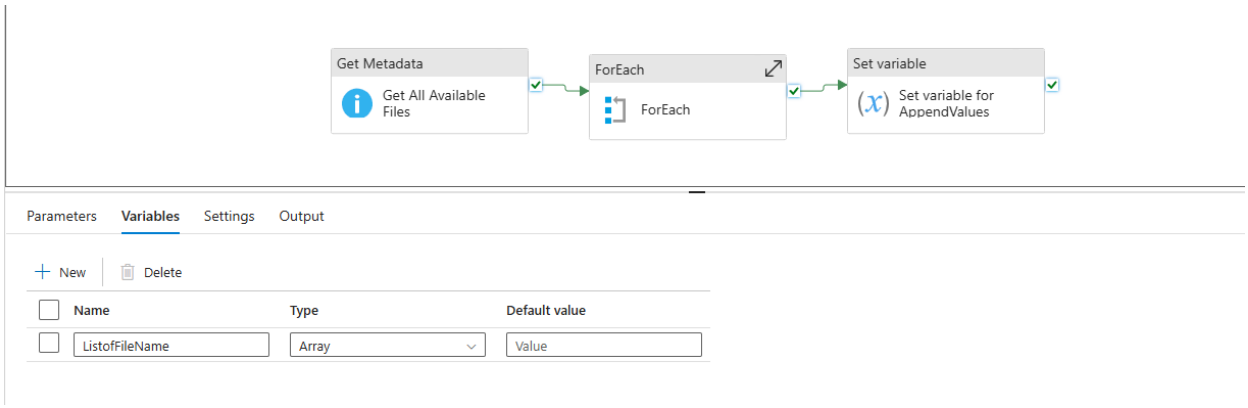
@concat(replace(dataset().fileName, '.csv', ''), '/', replace(dataset().fileName, '.csv', ''),'_',utcNow(),'.csv')

This will create a folder in the clientbackupfiles folder as per our file (accounts, transactions, loans, etc)

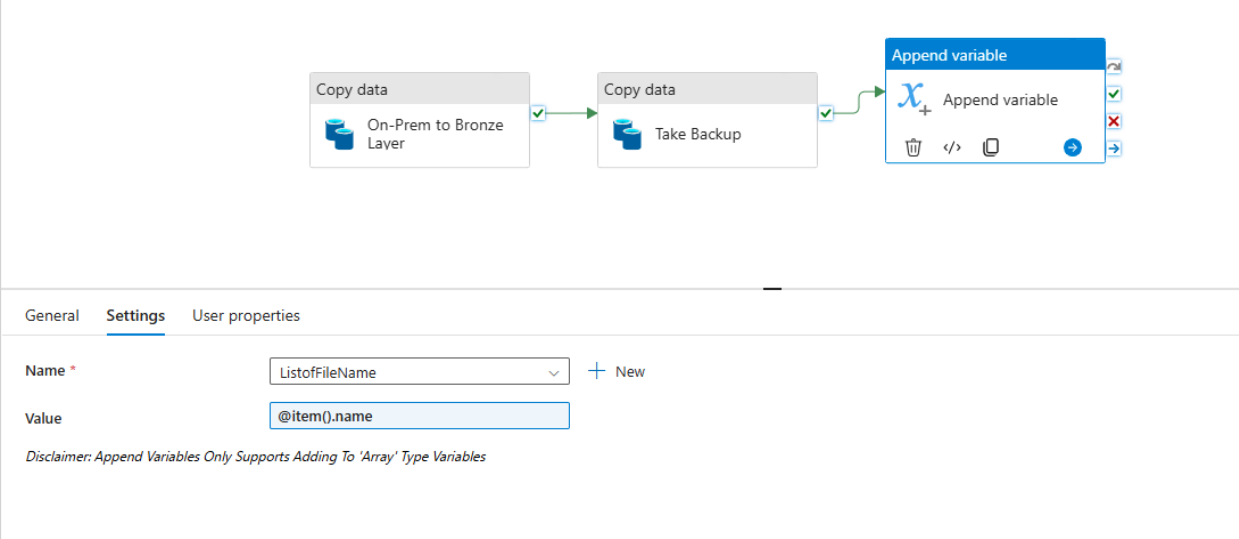
Step 1.6: Capture File Names

- Append Variable Activity: Collects names of modified files for downstream steps
- Inform Silver and Gold layers which files to process.

First of all, create a pipeline level variable as shown below



Now, go inside the ForEach -> If Condition activity



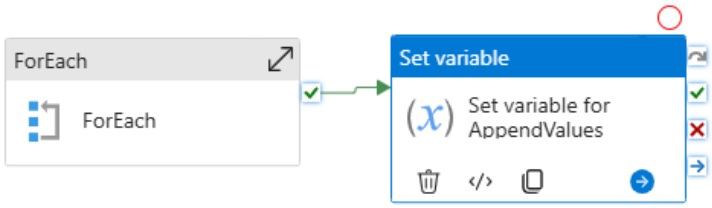
Select that variable here, and in value mention @item().name

So it will store all file names if they are modified or the client has added new data inside them.

Step 1.7: Return Values to Master Pipeline

- Set Variable Activity

Go outside of the ForEach activity and take the Set Variable activity



Create a new variable, and the type should be Pipeline Return Value

Name: ReturnFileNameValue

Type: Expression

Value: @variables('ListofFileName')

So, now we have designed our bronze layer.

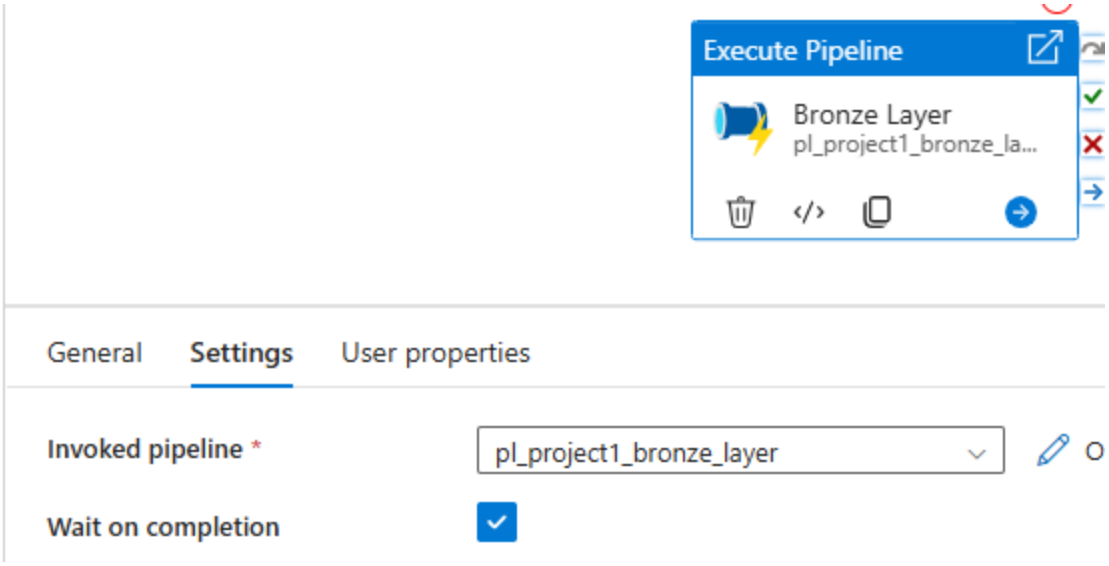
Step 1.8: Create Master Pipeline

- Master Pipeline

Create a new pipeline, let's call it Master Pipeline, from which we will execute all layers. (Bronze, Silver and Gold Layers)

Take the Execute Pipeline activity and drag it to the canvas.

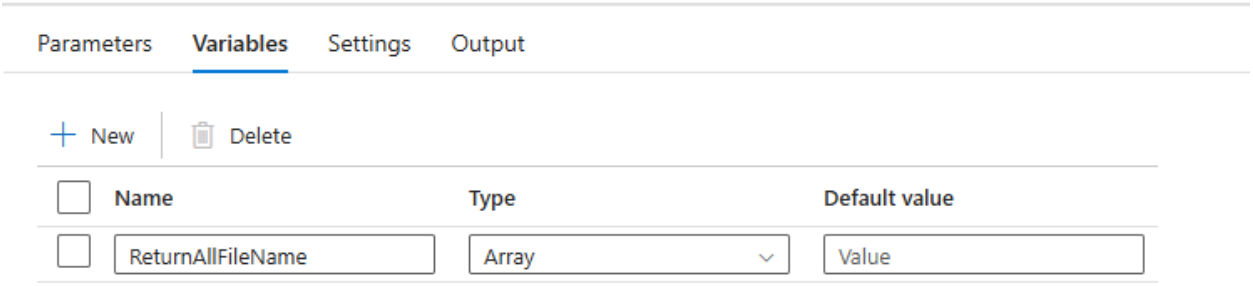
In Execute Pipeline Settings, select our Bronze layer pipeline, as shown below



Step1.9: Set Variable Activity in Master Pipeline

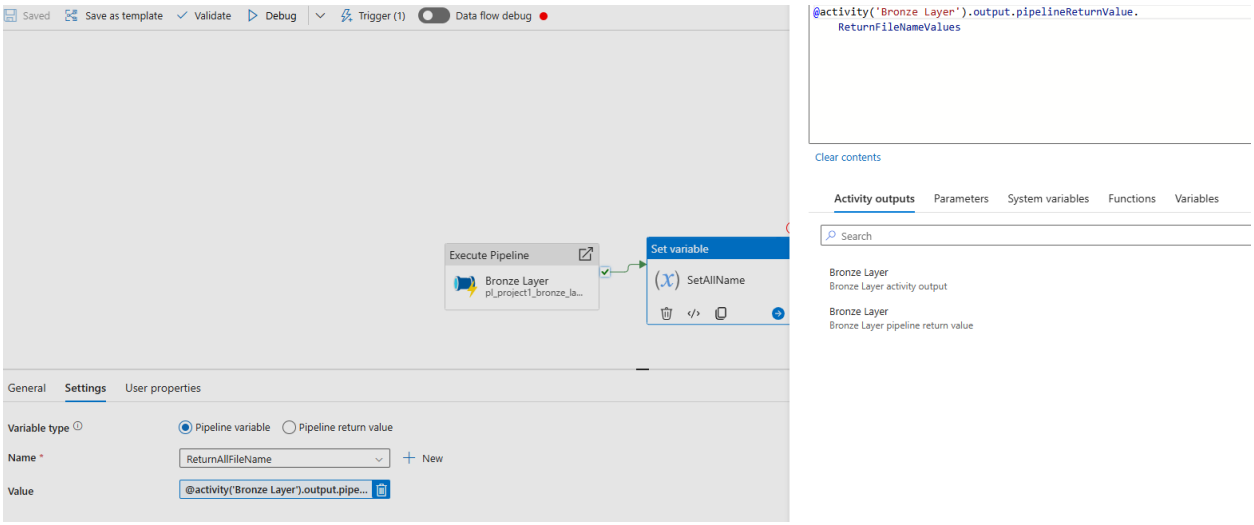
Now, take the Set variable activity after an Execute pipeline activity

Take the Pipeline level variable in the master pipeline



The type should be an Array as we need an array of file names.

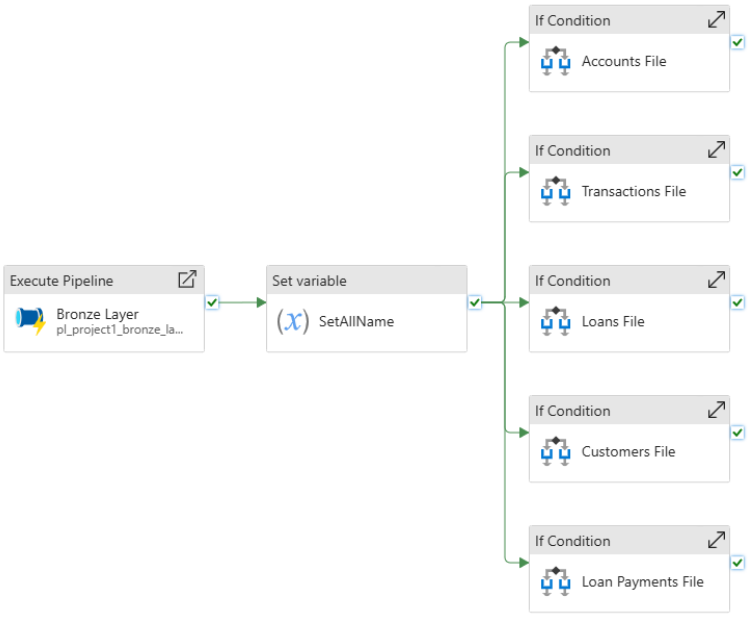
Write this expression in Set Variable -> Settings -> Value



@activity('Bronze Layer').output.pipelineReturnValue.ReturnFileNameValues

Step 1.9: If Condition in Master Pipeline

- If Condition: IF Condition activities allow you to create branching logic in your pipeline flow. Based on a dynamic condition (True/False), you can control which activities or pipelines should execute, making your process efficient and intelligent.

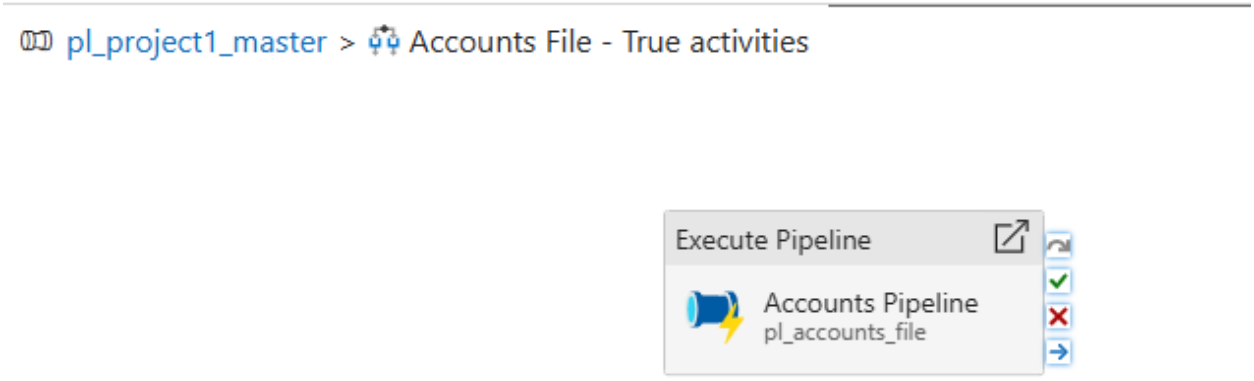


Rename each IF Condition activity appropriately based on the dataset it is intended to control (as shown in the reference image). Within the True path of each IF Condition, add an Execute Pipeline activity.

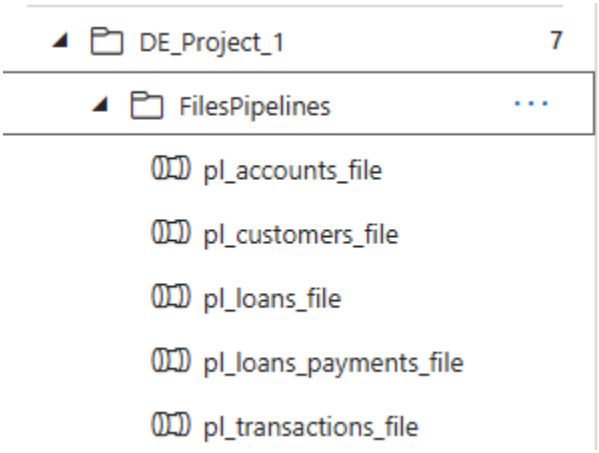
Each Execute Pipeline activity should be configured to call a dedicated pipeline designed specifically for that file, where both Silver Layer (data transformation) and Gold Layer (SCD processing and loading) are implemented.

This structure ensures that only the relevant Silver and Gold processes are triggered for each dataset individually, based on the file modification status.

For Example:



As shown below, create 5 pipelines for all 5 CSV files to be processed in the silver and gold layers.

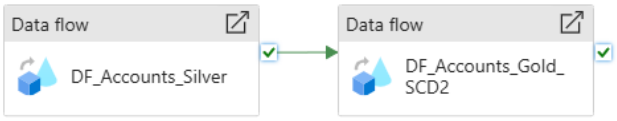


Step 2: Use ADF Dataflows to remove the duplicates

Step 2.1: Accounts File Pipeline Design (Silver Layer Processing)

The Silver Layer aims to cleanse, standardize, and deduplicate the raw ingested data to prepare it for downstream processing in the Gold Layer.

Dataflow Activity: Drag and drop a new Dataflow into the canvas area.



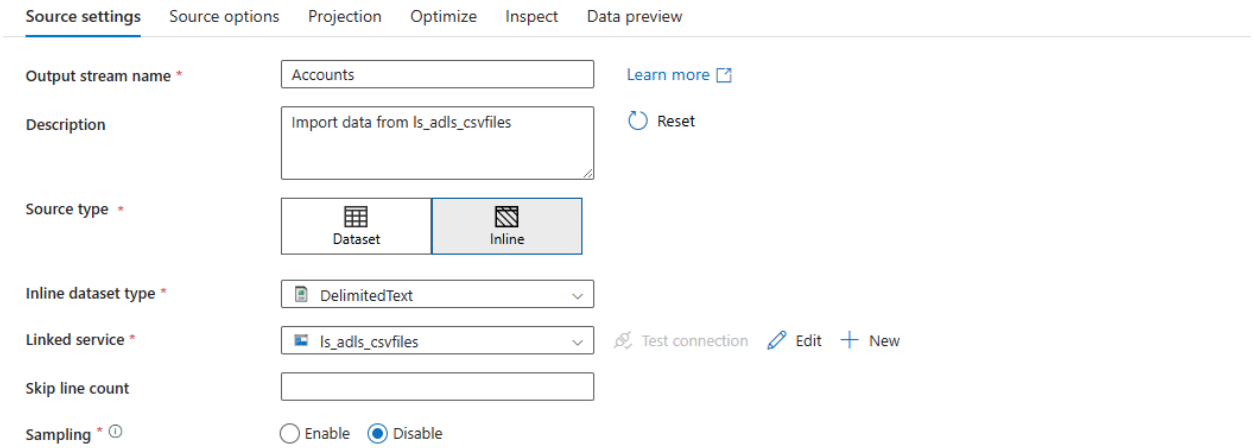
First, we will go through the silver layer in all pipelines.

Here is the entire flow:

Step-by-Step Process:

- Source Transformation:

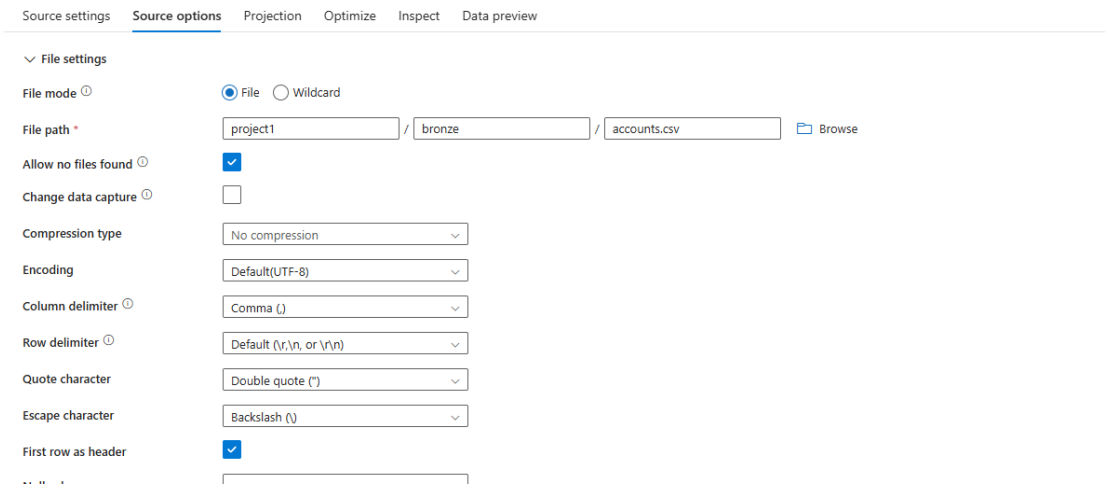
Source Settings:



Source Options:

Source Dataset: Point to the accounts.csv file located in the Bronze Layer (ADLS Gen2 storage).

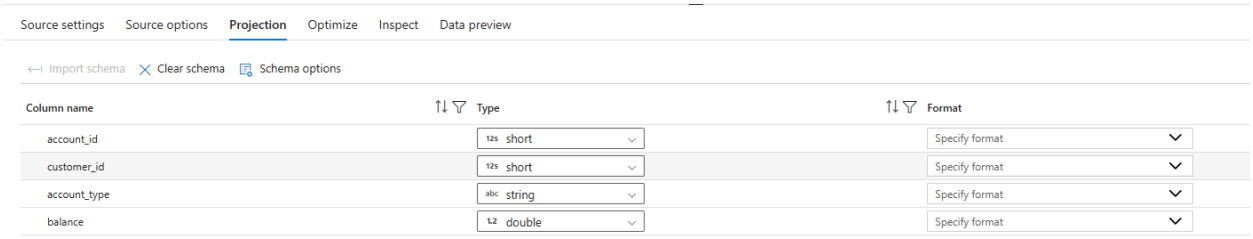
Specify the file path. Enable the checkbox "First row as header" since the file contains column names.



Projection:

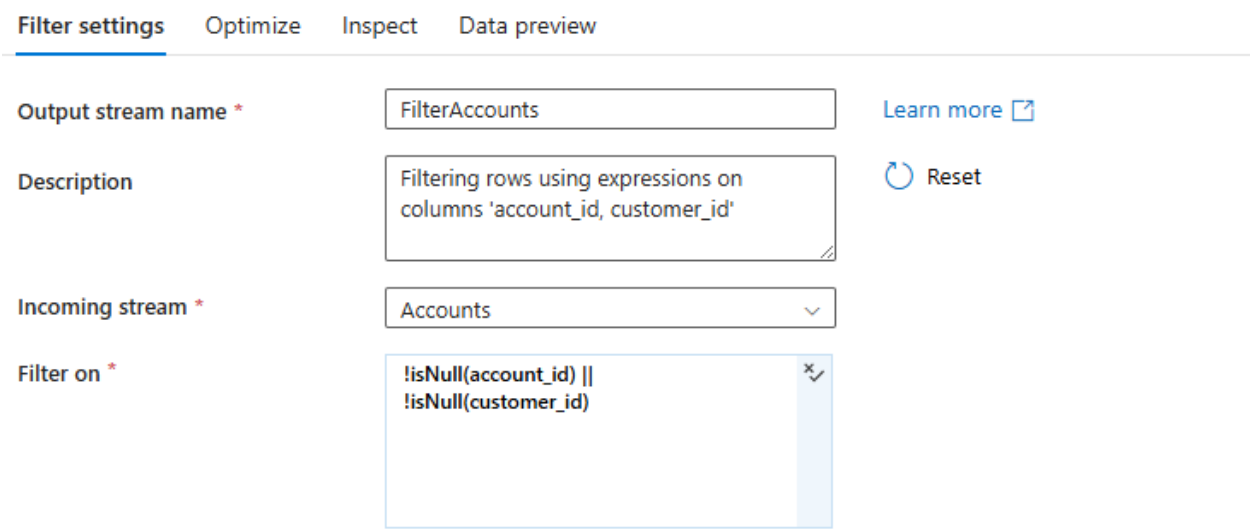
- Projection allows you to define or modify the column structure of the incoming dataset in ADF Dataflows.
- Click Import Schema to automatically detect the file structure.

- Perform a Data Preview to validate imported columns.

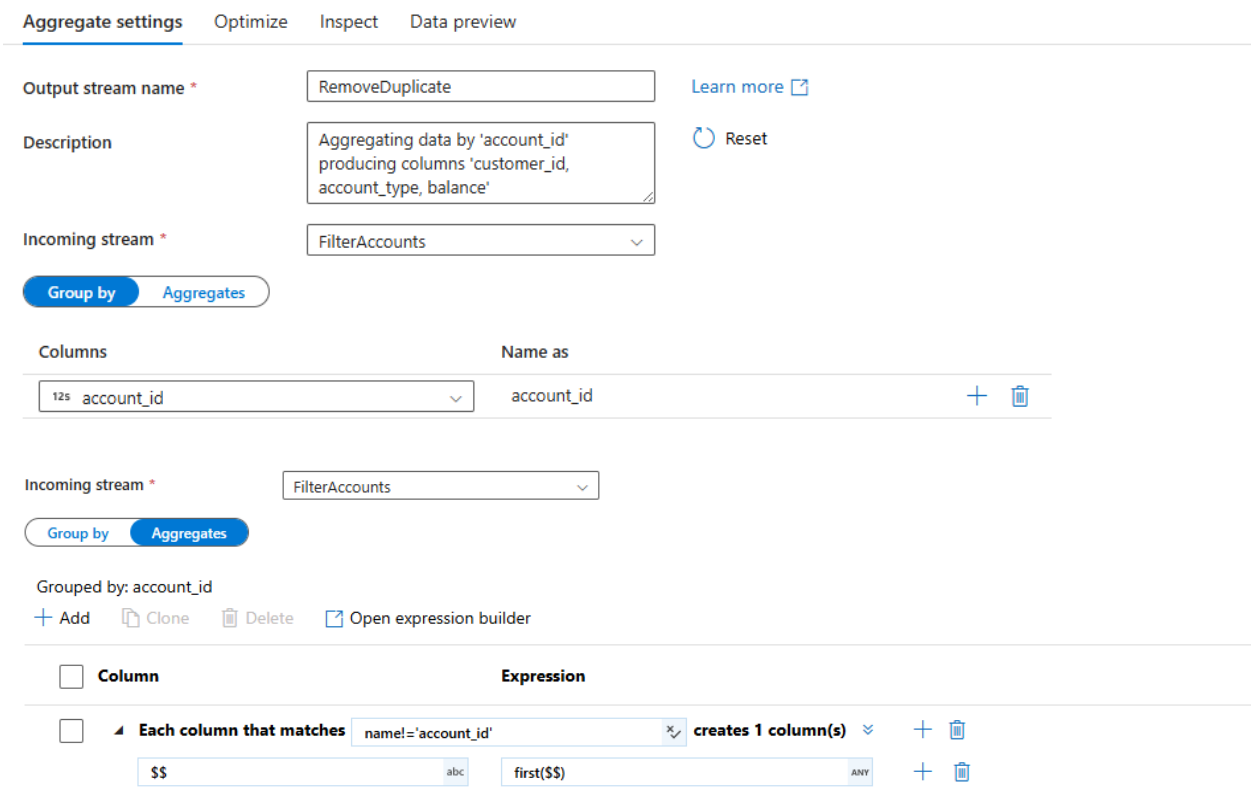


- **Filter Transformation**
  - Add a Filter Transformation immediately after the Source.
  - Filter Transformation allows you to include or exclude rows based on logical conditions.

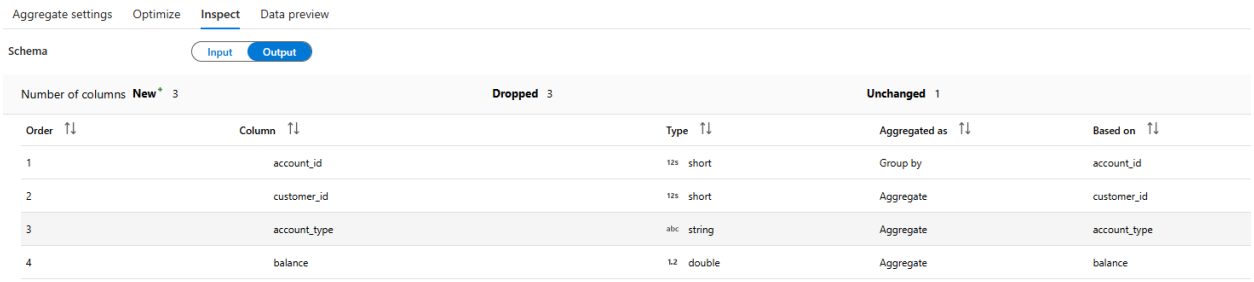
Use this filtering condition: !isNull(account_id) || !isNull(customer_id)



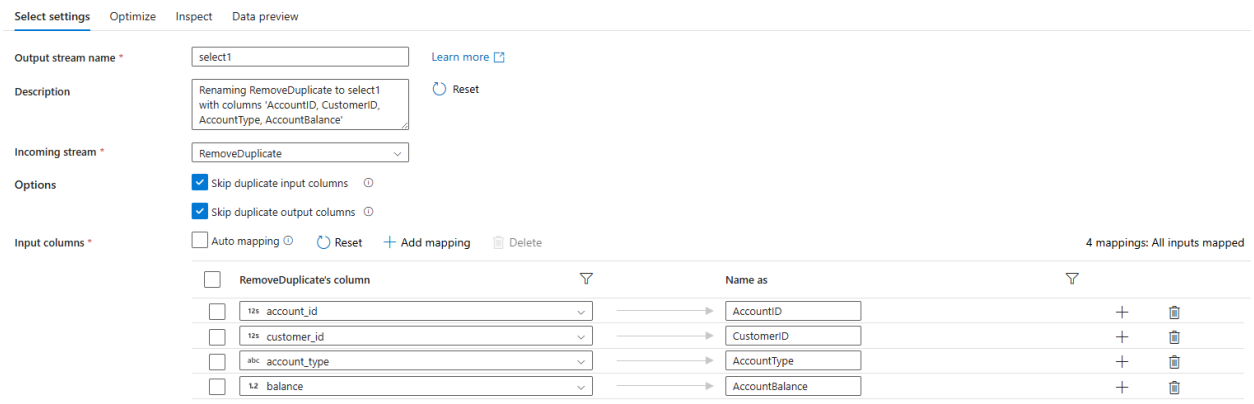
- **Aggregate Transformation(Remove Duplicate)**
  - Choose appropriate columns (account_id, customer_id) to identify duplicates.
  - Ensure only unique records move forward for consistency and data quality.
  - Remove Duplicate identifies and removes rows that have identical values across selected key columns.



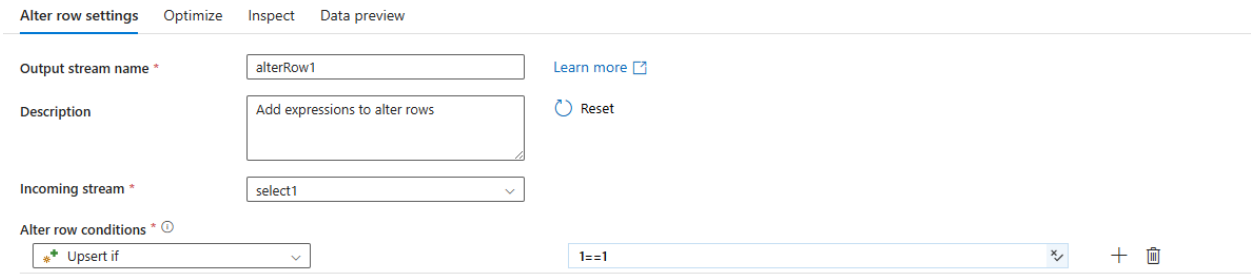
Inspect Tab:



- **Select Transformation**
  - Add a Select Transformation to rename columns for standardization.
  - Example: Rename account_id to AccountID, customer_id to CustomerID, etc.
  - Select Transformation is used to rename, reorder, or drop columns in ADF Dataflows.

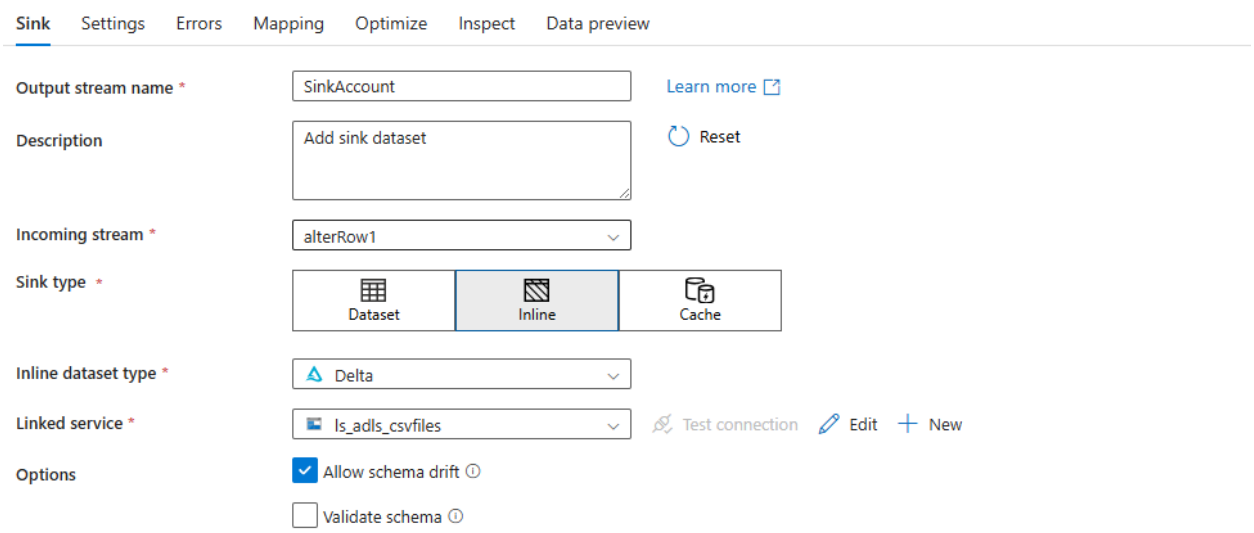


- **Alter Row Transformation**
  - Insert an Alter Row Transformation.
  - Configure it to always Upsert records using the condition: 1==1
  - Alter Row Transformation assigns actions (Insert, Update, Delete, Upsert) at the row level based on specified conditions.

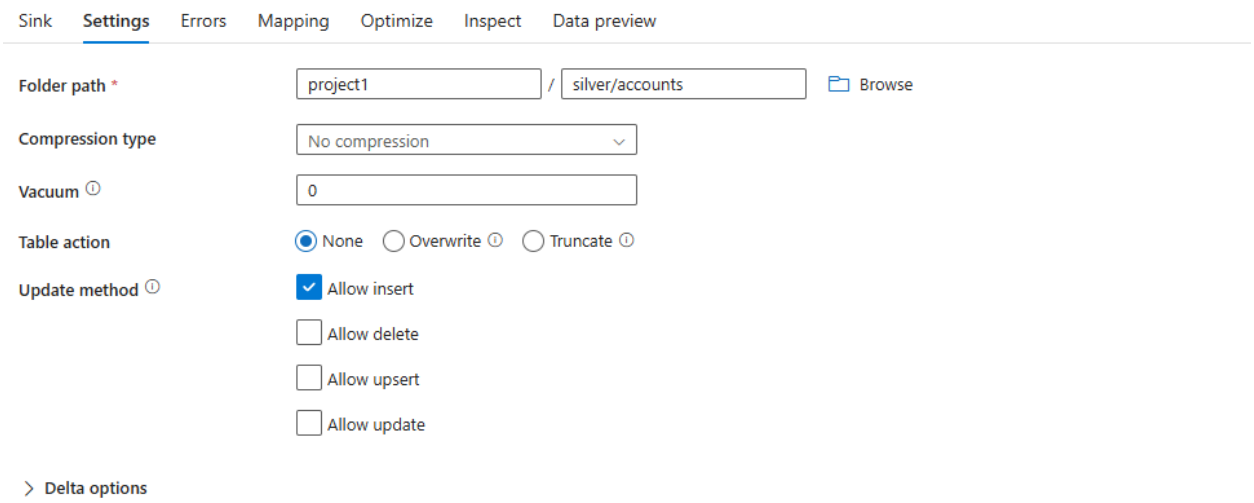


- **Sink Transformation**

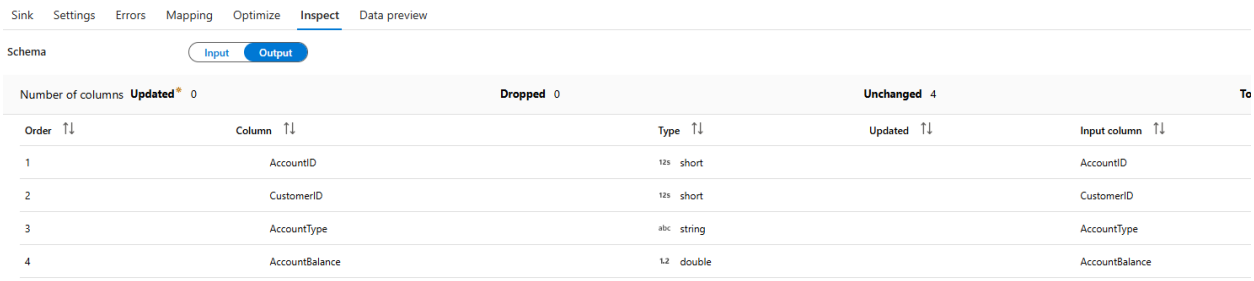
  - Add a Sink to write the cleansed data.
  - Sink Type: Inline dataset.
  - Dataset Type: Delta (high-performance, ACID-compliant format).
  - Linked Service: ADLS Gen2 storage account.
  - Settings Tab:
    - Enable "Auto Create Table" if needed.
    - Review mappings under the Mapping Tab to ensure correct column flow.
  - Sink is the destination where transformed data is written, such as files, tables, or databases.



Settings:



Inspect:



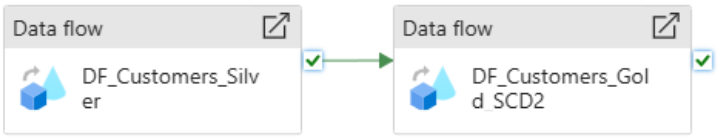
That's it for the silver layer accounts dataflow.

Similarly, we have to design for the other 4 files.

Step 2.2: Customers File Pipeline (Silver Layer Processing)

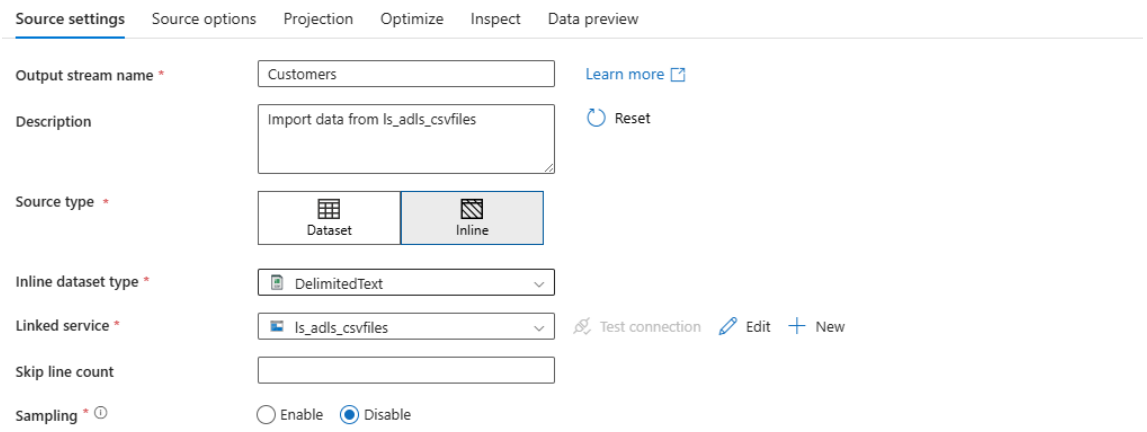
Step-by-Step Process:

Drag and drop two dataflows in the canvas area



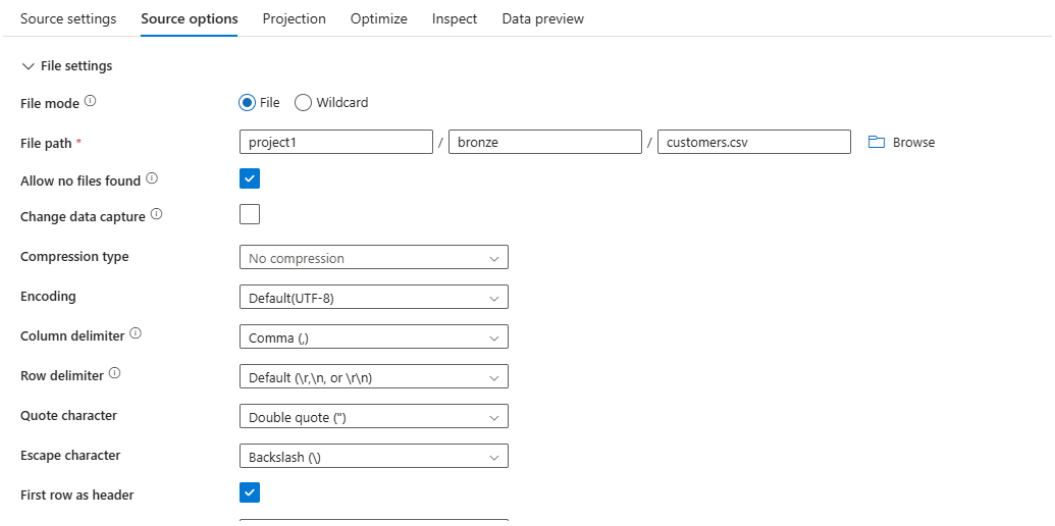
Here is the entire flow:

- **Source Transformation**



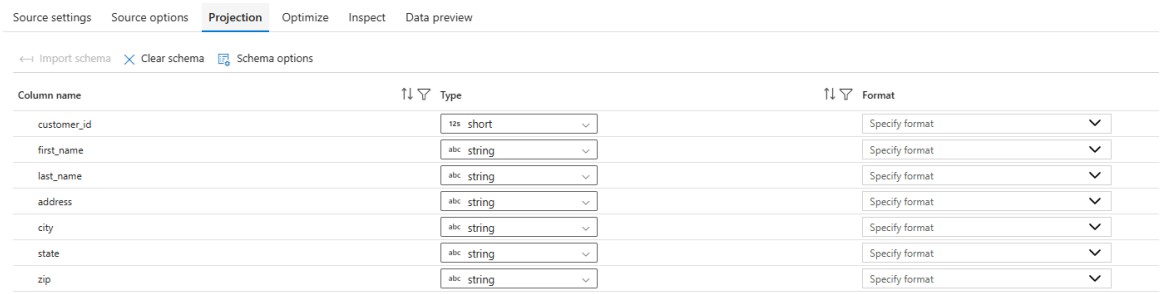
Source Options:

Set the file path pointing to the Bronze folder. Enable the "First row as header" option since the file includes headers.



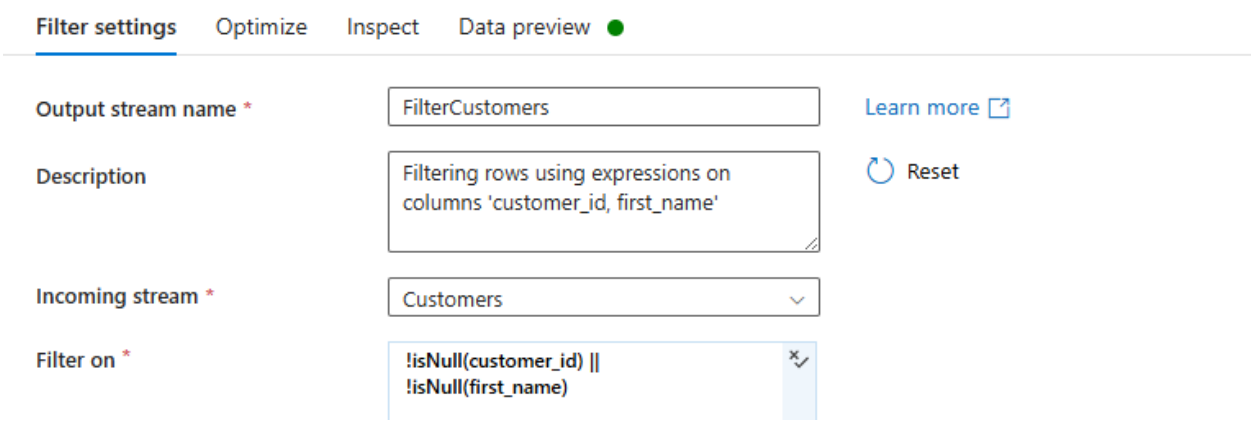
Projection:

- Click Import Schema to automatically generate the schema.
- Perform a Data Preview to verify columns and data types.

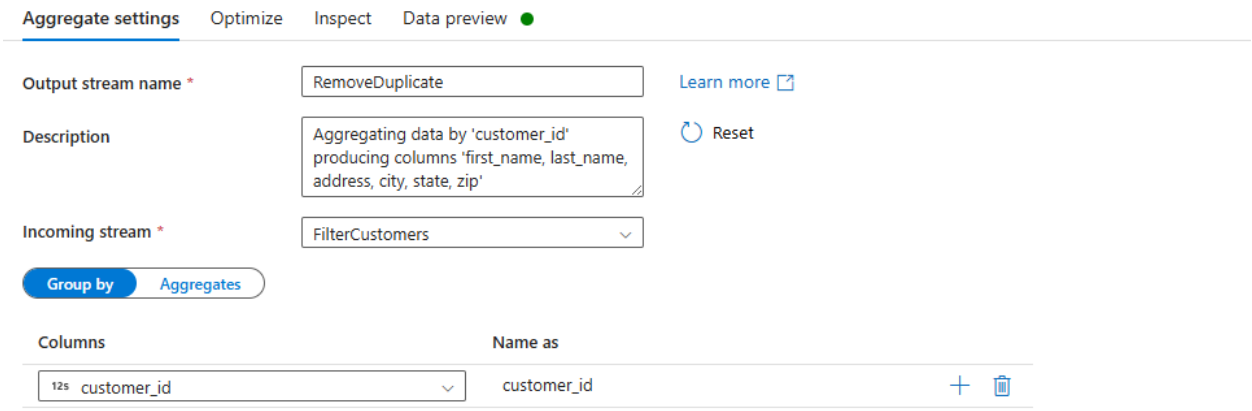


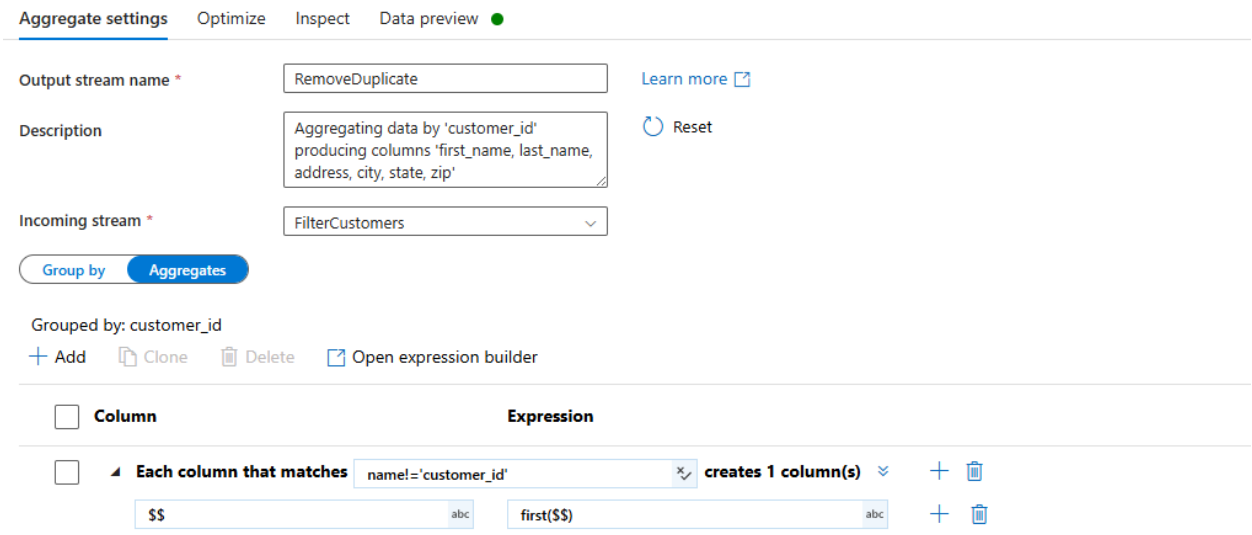
- **Filter Transformation**

Add a Filter Transformation after the Source. Use this expression: !isNull(customer_id) || !isNull(first_name)



- **Aggregate Transformation (Remove Duplicate):**





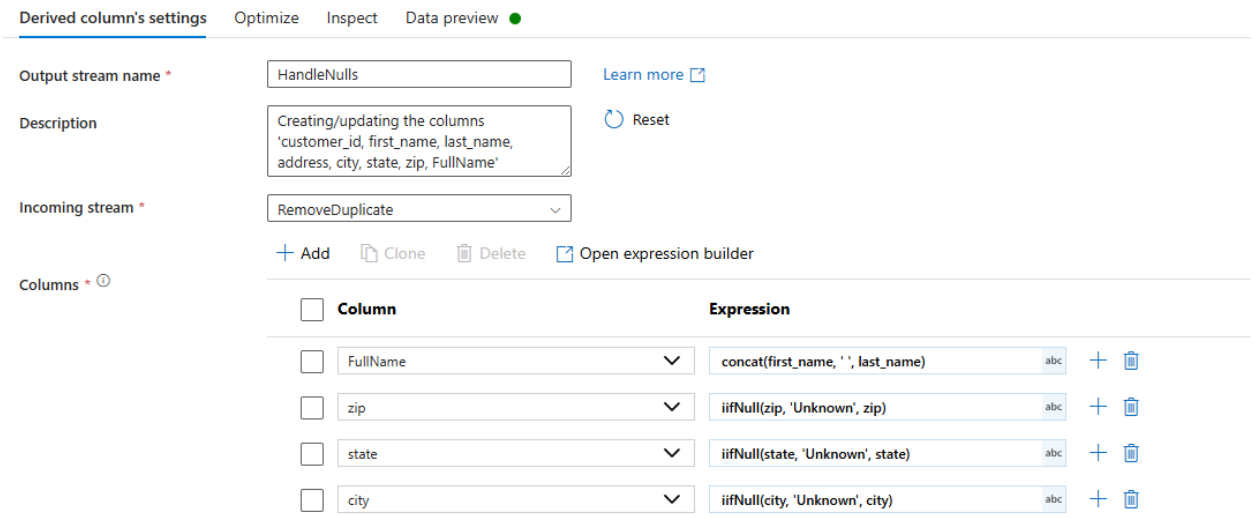
Data Preview Tab:

Refresh | Typecast | Modify | Map drifted | Statistics | Remove | Export to CSV

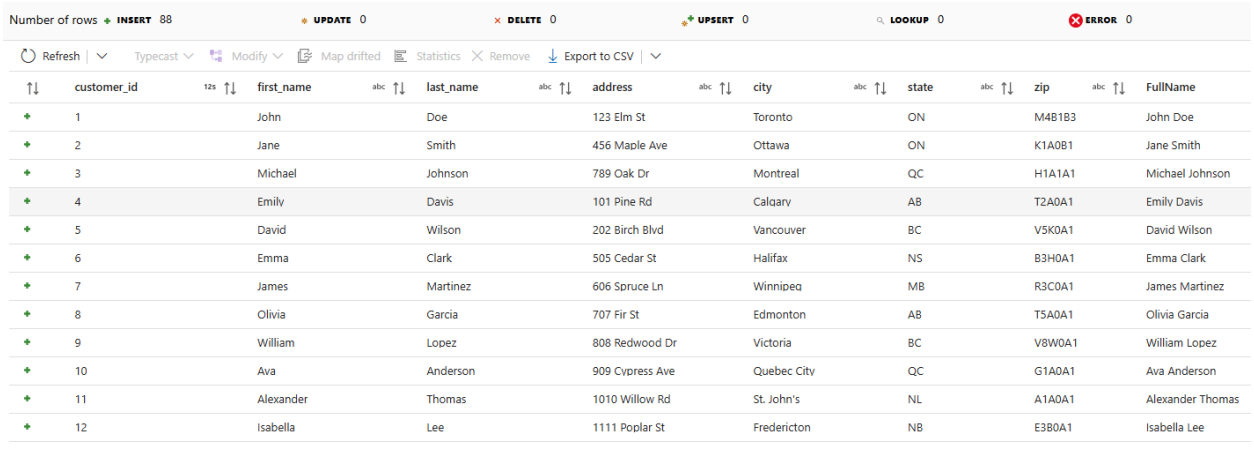
customer_id	first_name	last_name	address	city	state	zip
1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1
11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1
12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
13	Daniel	Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1

- **Derived Column Transformation:**

Column	Expression
FullName	concat(first_name, ' ', last_name)
zip	iifNull(zip, 'Unknown', zip)
state	iifNull(state, 'Unknown', state)
city	iifNull(city, 'Unknown', city)



Data Preview Tab:



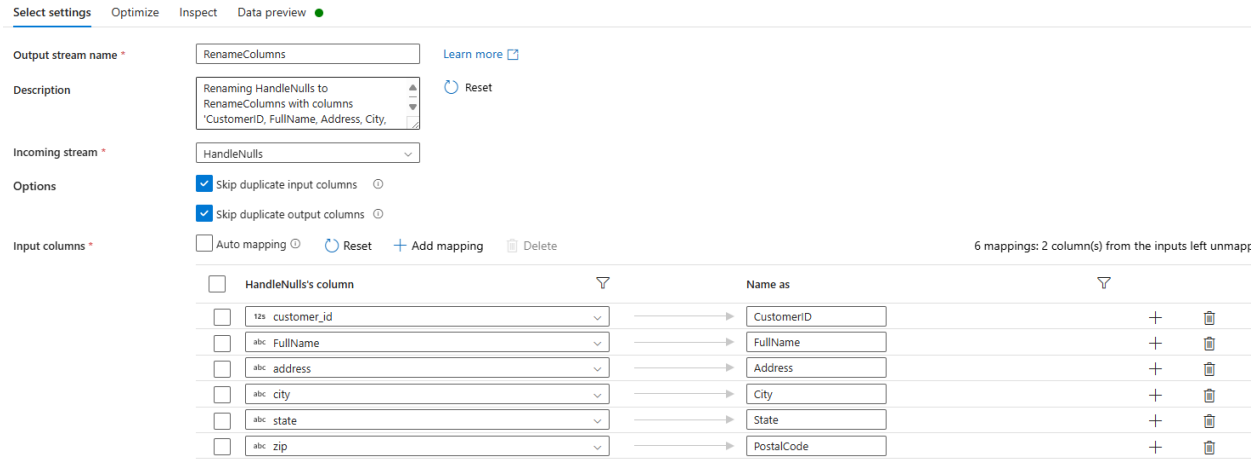
Number of rows 88 INSERT 0 UPDATE 0 DELETE 0 UPSERT 0 LOOKUP 0 ERROR

Refresh | Typecast | Modify | Map drifted | Statistics | Remove | Export to CSV

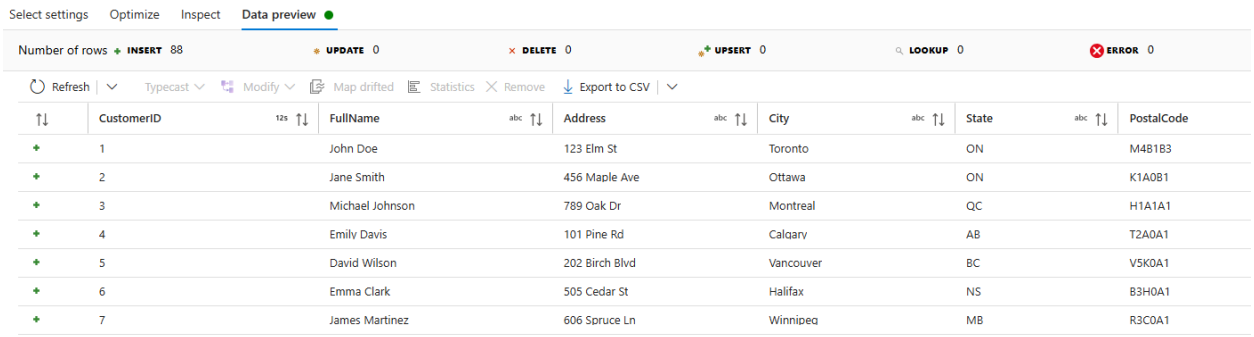
customer_id	first_name	last_name	address	city	state	zip	FullName
1	John	Doe	123 Elm St	Toronto	ON	M4B1B3	John Doe
2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1	Jane Smith
3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1	Michael Johnson
4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1	Emily Davis
5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1	David Wilson
6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1	Emma Clark
7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1	James Martinez
8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1	Olivia Garcia
9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1	William Lopez
10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1	Ava Anderson
11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1	Alexander Thomas
12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1	Isabella Lee

- **Select Transformation:**

  Rename the column name here



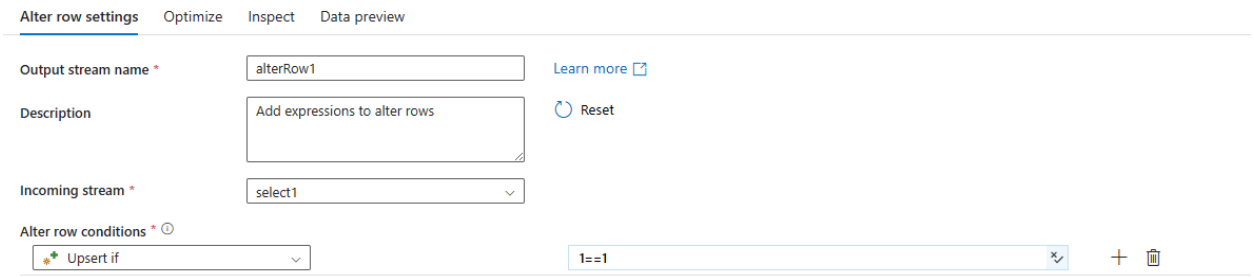
  Data Preview Tab:



CustomerID	FullName	Address	City	State	PostalCode
1	John Doe	123 Elm St	Toronto	ON	M4B1B3
2	Jane Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	Michael Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	Emily Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	David Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	Emma Clark	505 Cedar St	Halifax	NS	B3H0A1
7	James Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1

- **Alter Row Transformation:**

  Select Upsert if condition from the drop down, and 1==1 expression

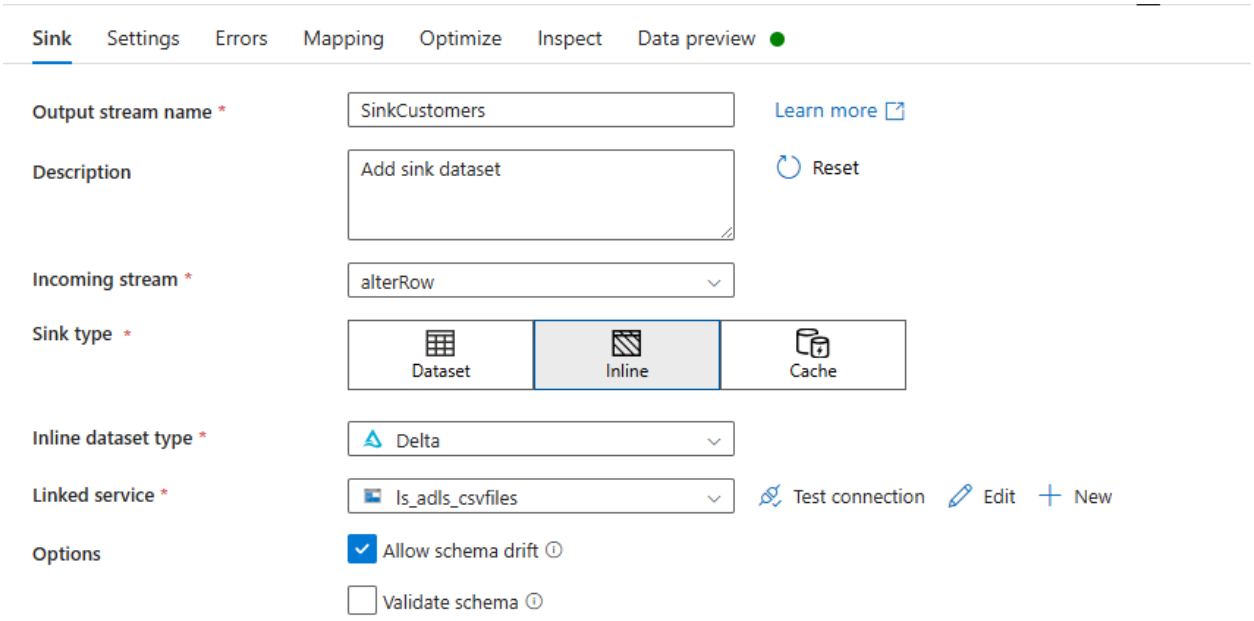


- **Sink Transformation:**

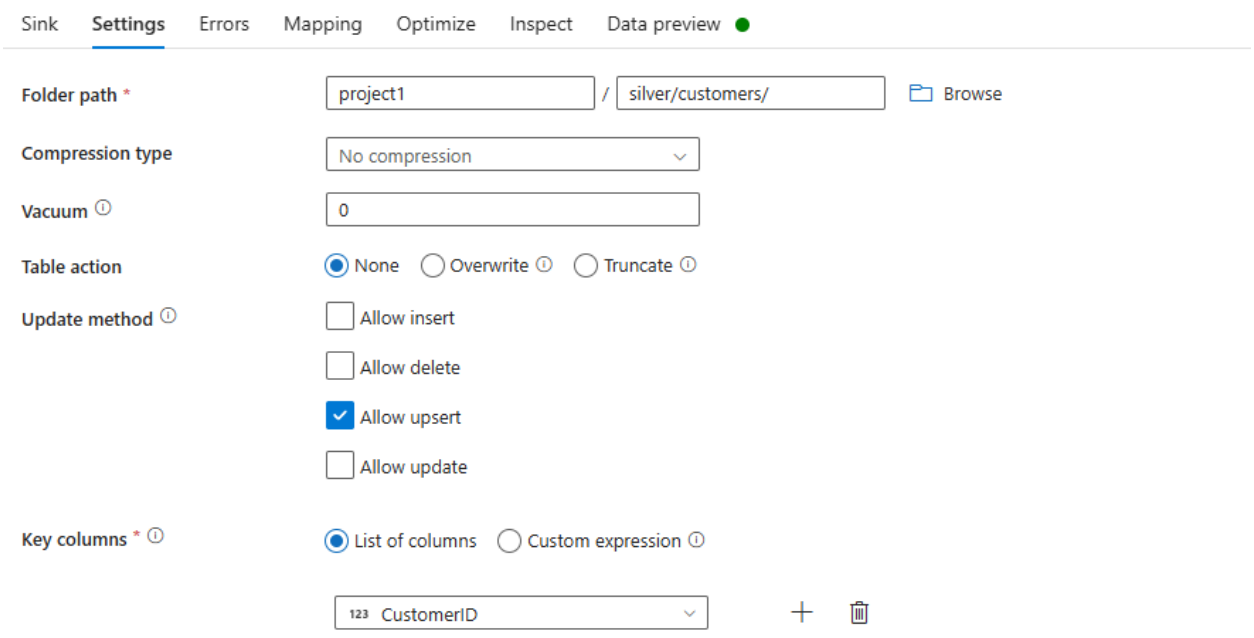
  Sink type: Inline

Inline dataset type: Delta

Linked service: ADLS Gen 2 storage account



Settings:



Data Preview Tab:

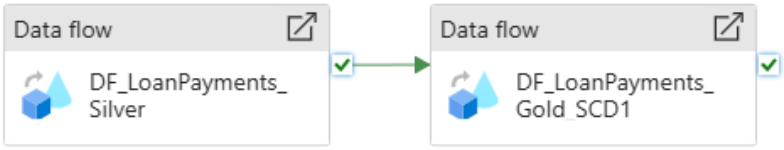
Refresh | Statistics | Export to CSV

CustomerID	FullName	Address	City	State	PostalCode
1	John Doe	123 Elm St	Toronto	ON	M4B1B3
2	Jane Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	Michael Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	Emily Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	David Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	Emma Clark	505 Cedar St	Halifax	NS	B3H0A1
7	James Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	Olivia Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	William Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	Ava Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1
11	Alexander Thomas	1010 Willow Rd	St. John's	NL	A1A0A1
12	Isabella Lee	1111 Poplar St	Fredericton	NB	E3B0A1
13	Daniel Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1

That's it for the silver layer customers' data flow.

- **Loan Payments File Pipeline**

Drag and drop two dataflows in the canvas area

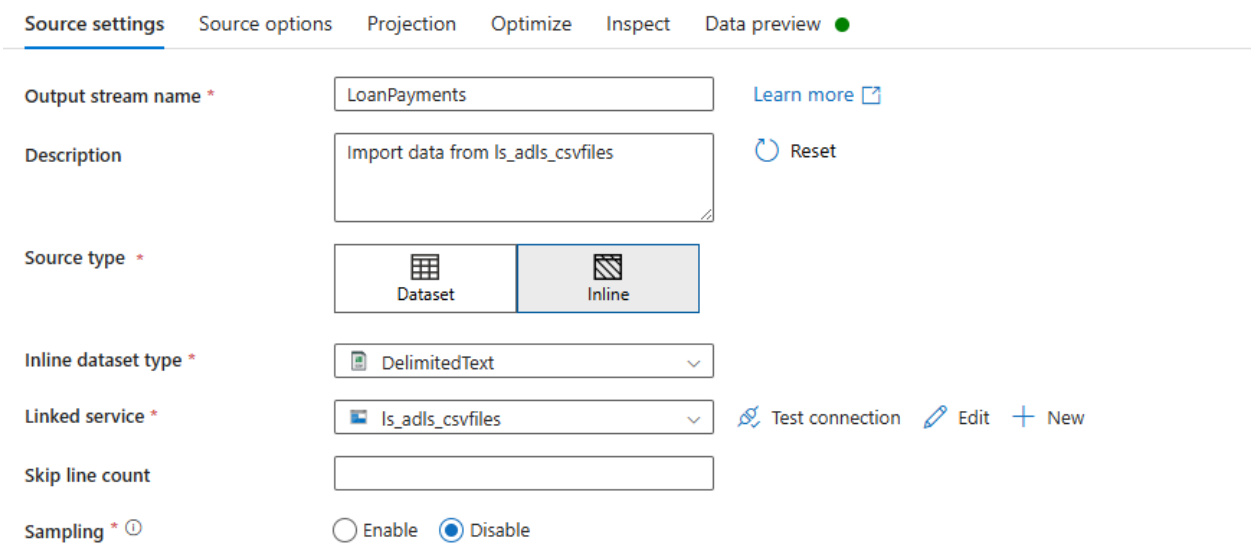


First, we will go through the silver layer in this.

Here is the entire flow:

- **Source:**

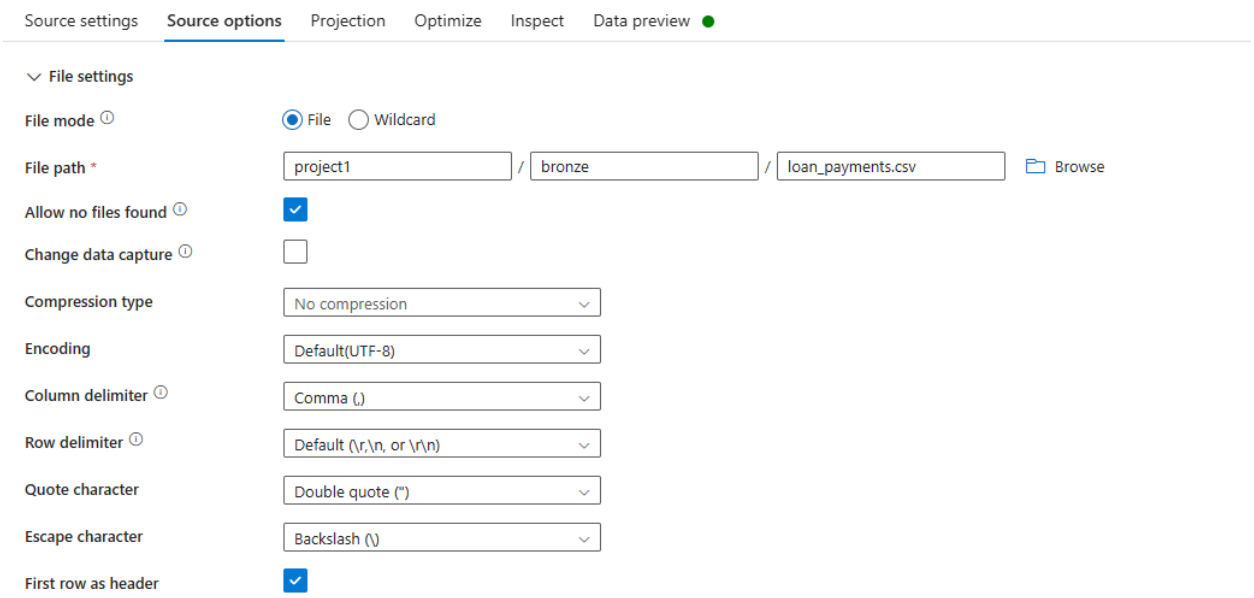
Source Settings:



Source Options:

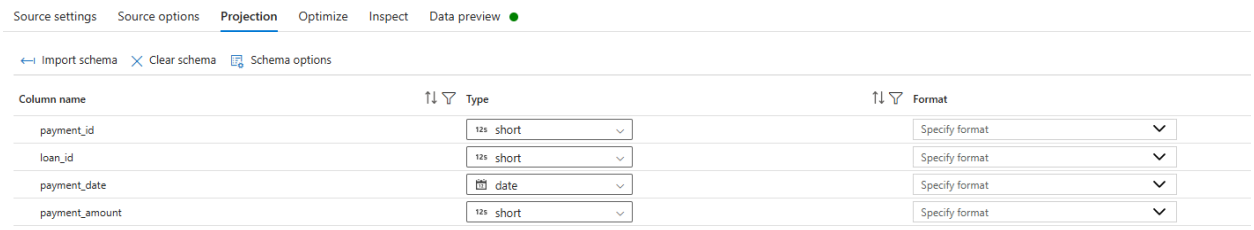
File path for the loan_payments.csv file from the bronze folder in ADLS Gen 2

Tick the first row as a header



Projection:

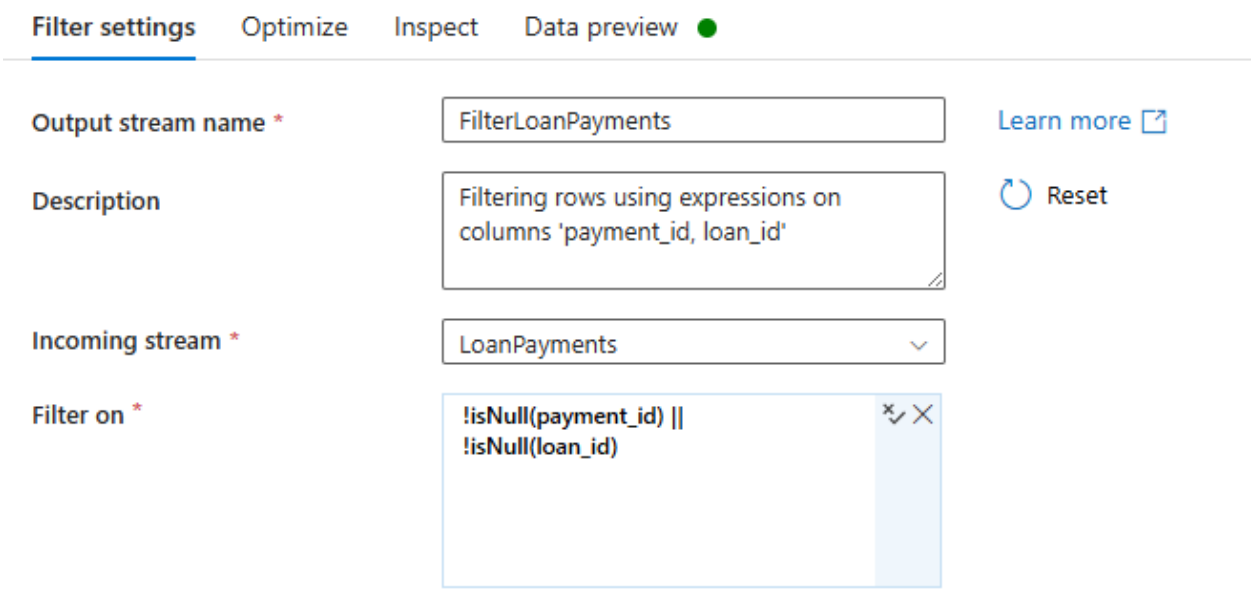
Click on Import Schema, and then click on Data Preview



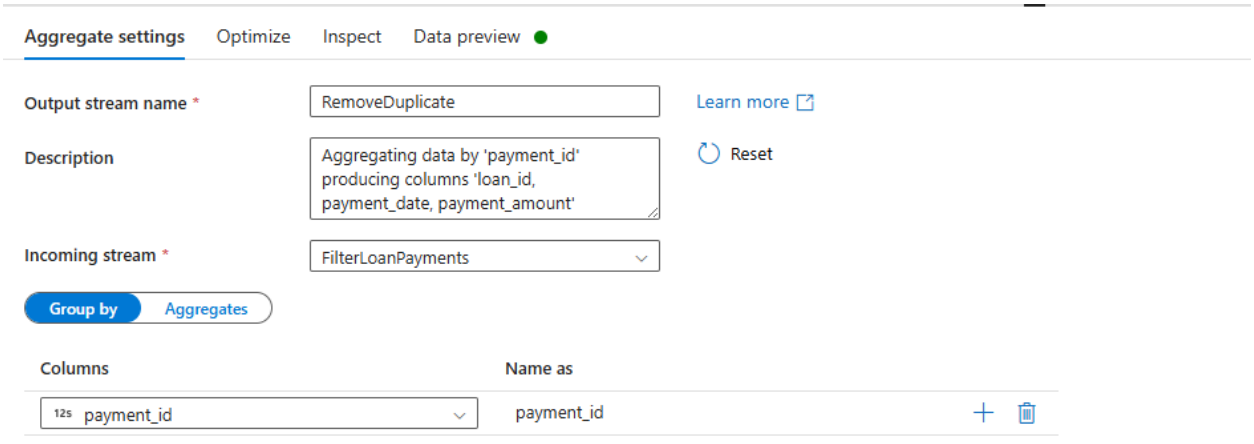
Filter transformation:

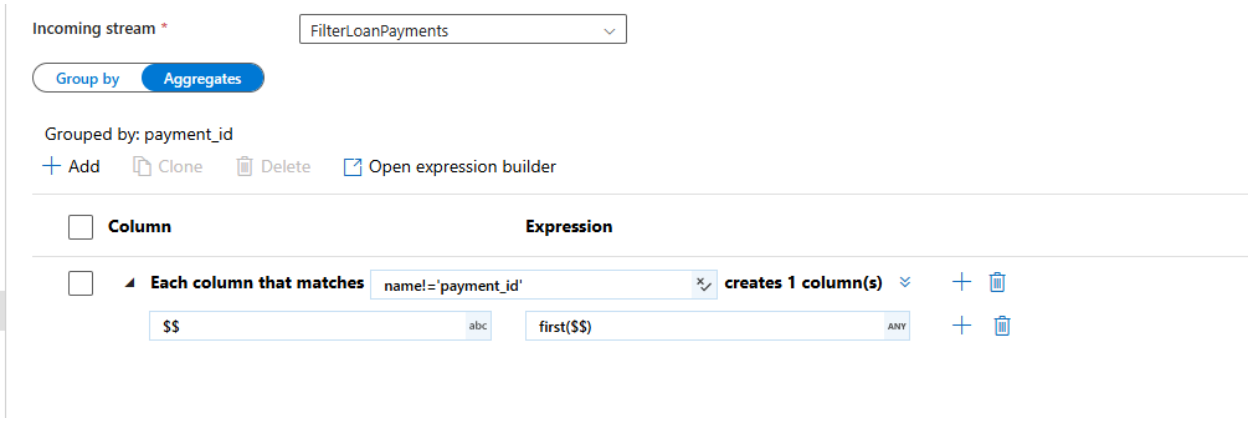
Click on the plus icon at the bottom of the source, and take a filter transformation

Use this expression: !isNull(payment_id) || !isNull(loan_id)

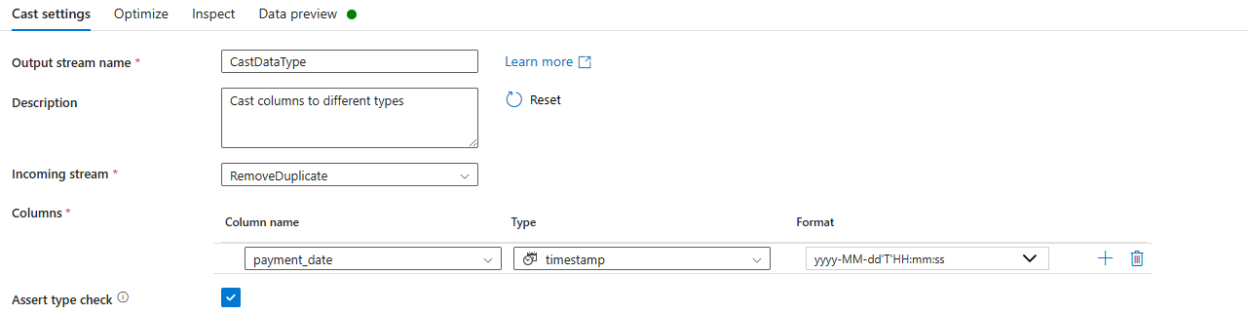


- **Aggregate Transformation (Remove Duplicate)**

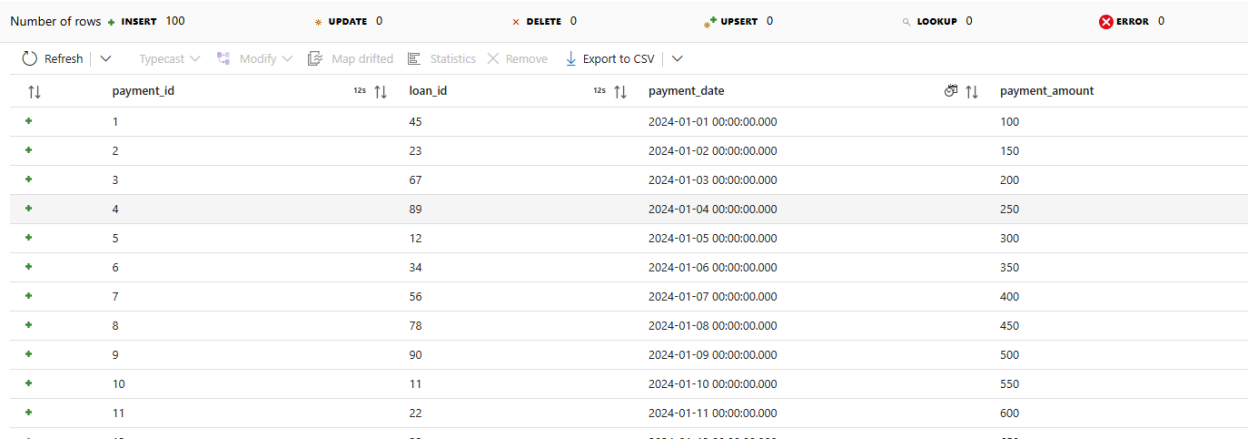




- **Cast Transformation:**

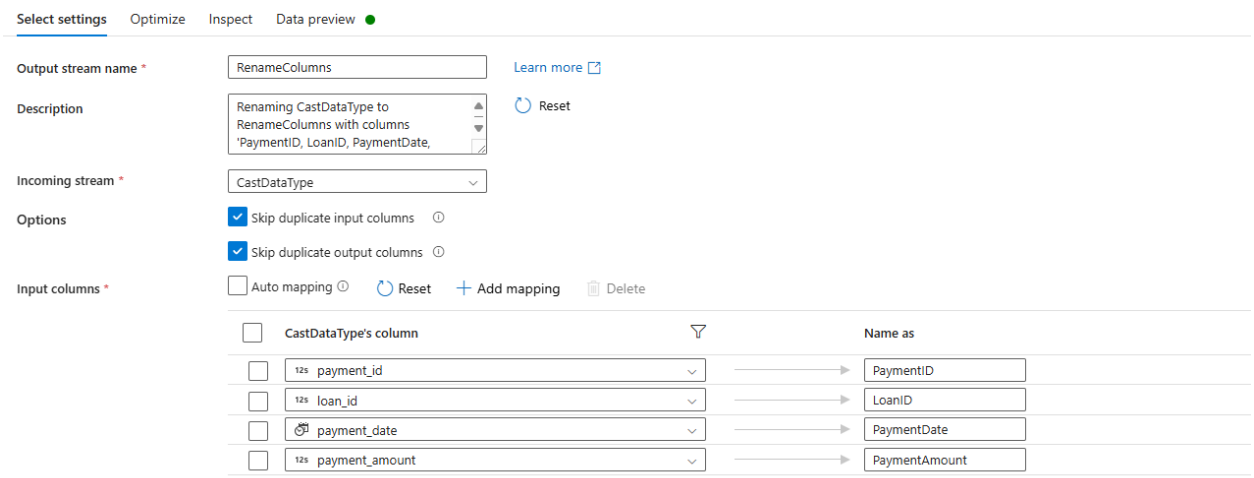


Data Preview:

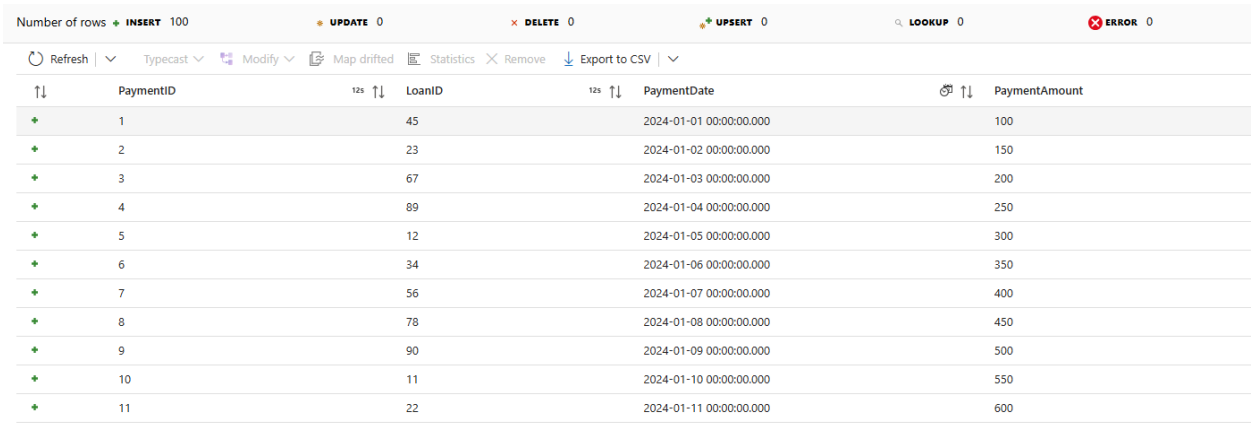


- **Select Transformation:**

Rename the column name here



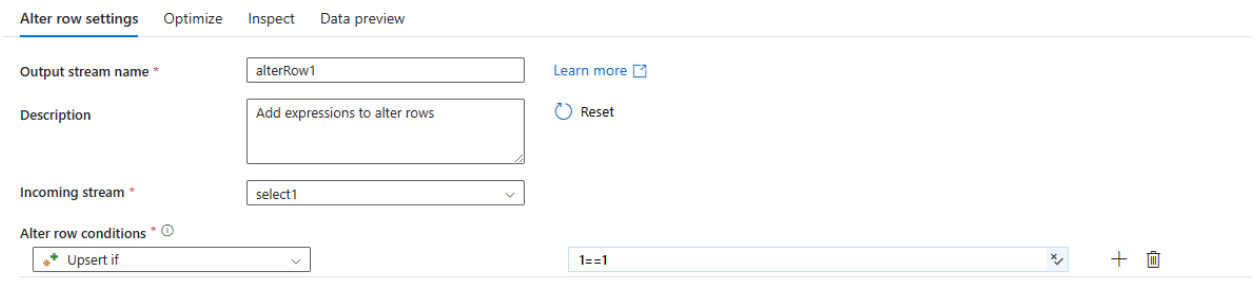
Data Preview Tab:



PaymentID	LoanID	PaymentDate	PaymentAmount
1	45	2024-01-01 00:00:00.000	100
2	23	2024-01-02 00:00:00.000	150
3	67	2024-01-03 00:00:00.000	200
4	89	2024-01-04 00:00:00.000	250
5	12	2024-01-05 00:00:00.000	300
6	34	2024-01-06 00:00:00.000	350
7	56	2024-01-07 00:00:00.000	400
8	78	2024-01-08 00:00:00.000	450
9	90	2024-01-09 00:00:00.000	500
10	11	2024-01-10 00:00:00.000	550
11	22	2024-01-11 00:00:00.000	600

- **Alter Row Transformation:**

  Select Upsert if condition from the drop down and 1==1 expression

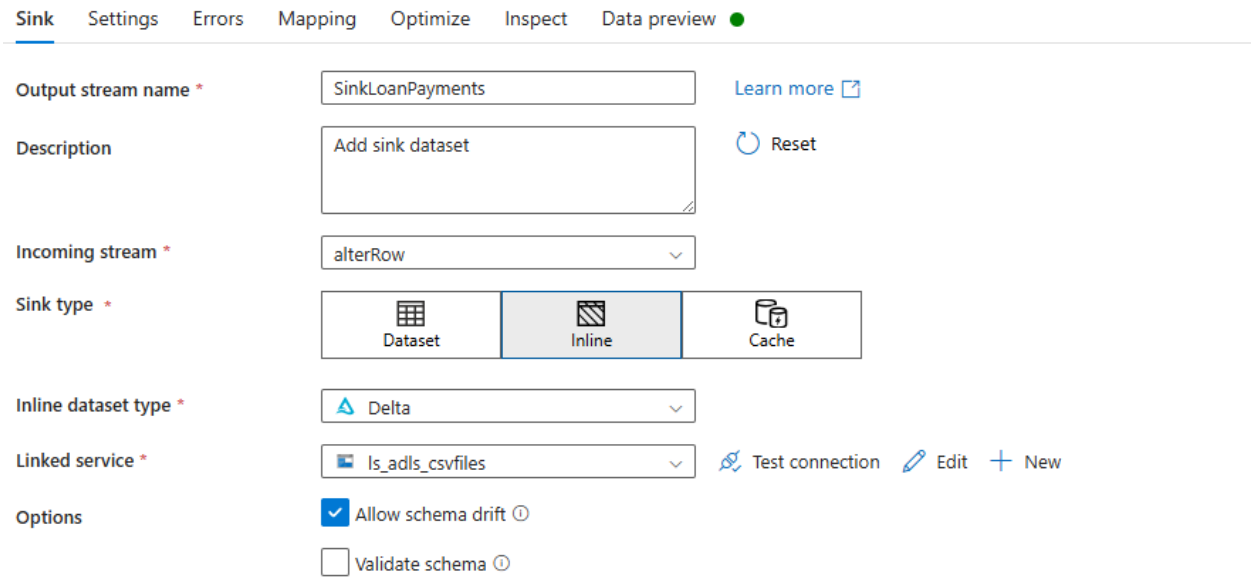


- **Sink Transformation:**

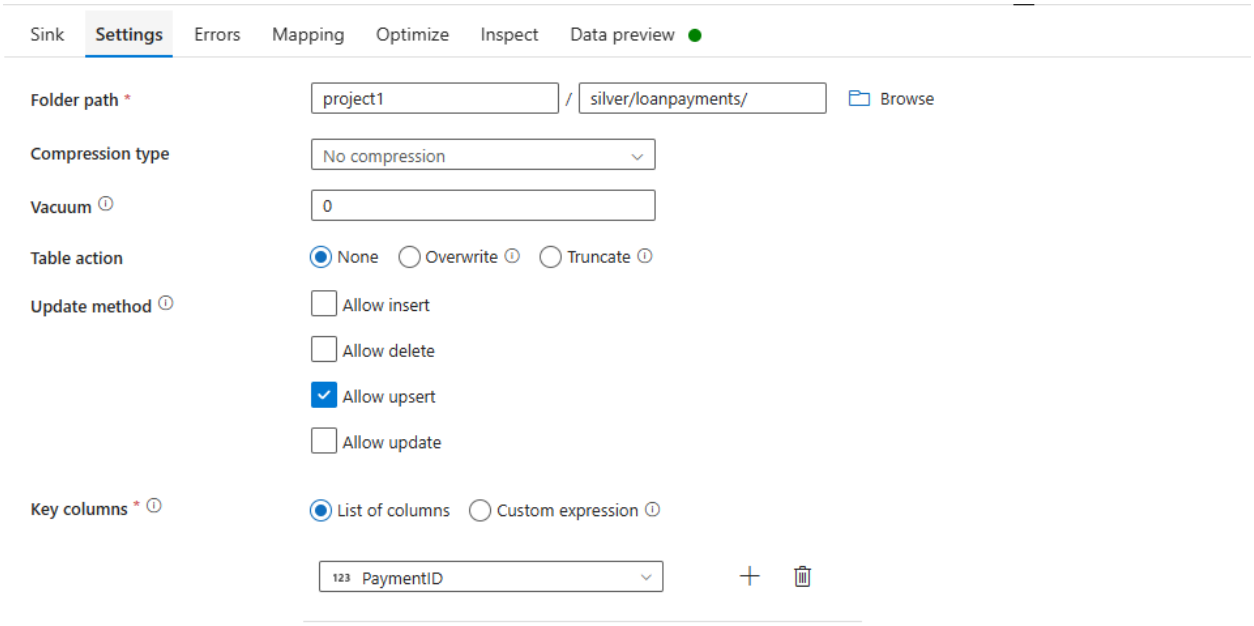
  Sink type: Inline

  Inline dataset type: Delta

Linked service: ADLS Gen 2 storage account



Settings:



Data Preview Tab:

Number of rows INSERT 0 UPDATE 0 DELETE 0 UPSERT 0 LOOKUP 0 ERROR 0

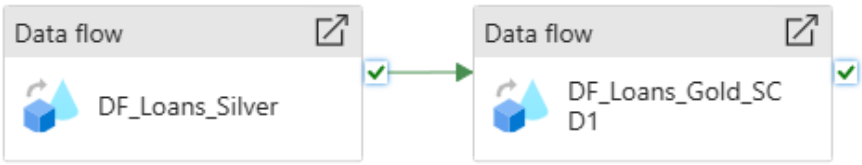
Refresh | Statistics | Export to CSV

PaymentID	LoanID	PaymentDate	PaymentAmount
1	45	2024-01-01 00:00:00.000	100
2	23	2024-01-02 00:00:00.000	150
3	67	2024-01-03 00:00:00.000	200
4	89	2024-01-04 00:00:00.000	250
5	12	2024-01-05 00:00:00.000	300
6	34	2024-01-06 00:00:00.000	350
7	56	2024-01-07 00:00:00.000	400
8	78	2024-01-08 00:00:00.000	450
9	90	2024-01-09 00:00:00.000	500
10	11	2024-01-10 00:00:00.000	550
11	22	2024-01-11 00:00:00.000	600
12	33	2024-01-12 00:00:00.000	650

That's it for the silver layer loan payments' data flow.

- **Loan File Pipeline (Silver Layer)**

Drag and drop two dataflows in the canvas area

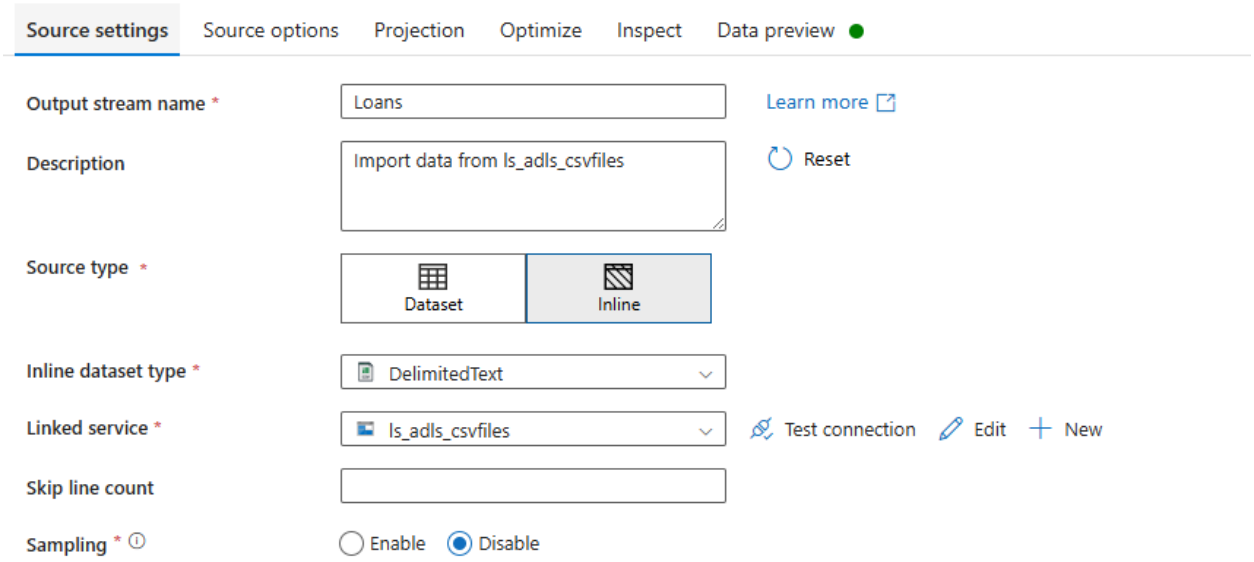


First, we will go through the silver layer in this.

Here is the entire flow:

Source:

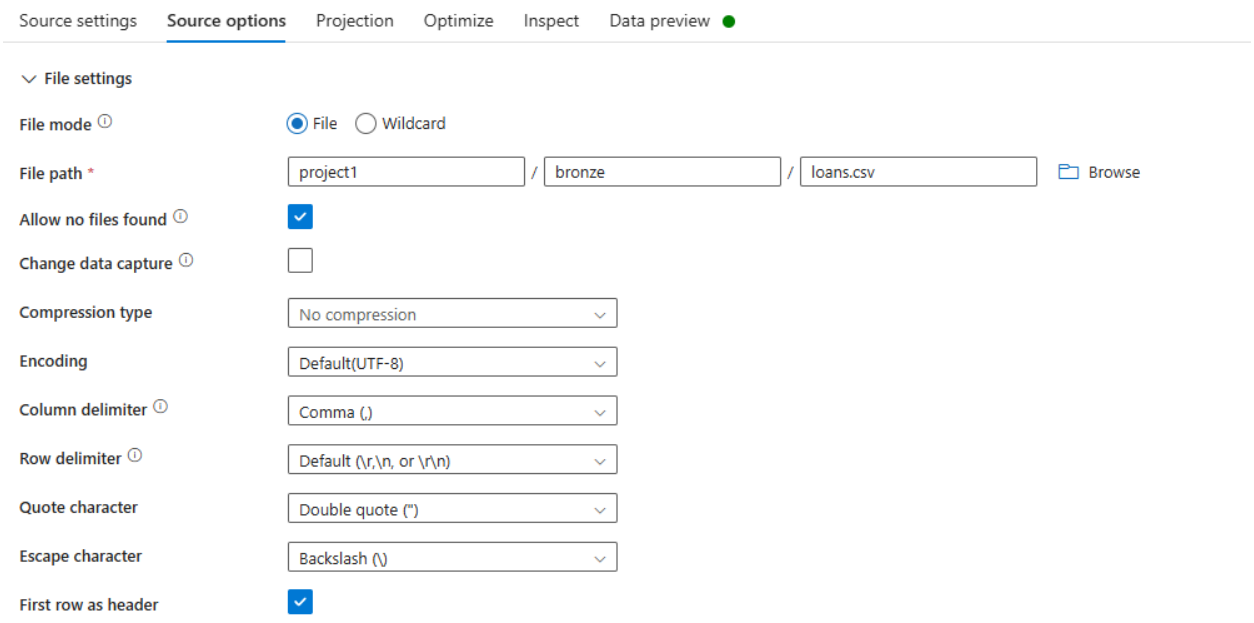
Source Settings:



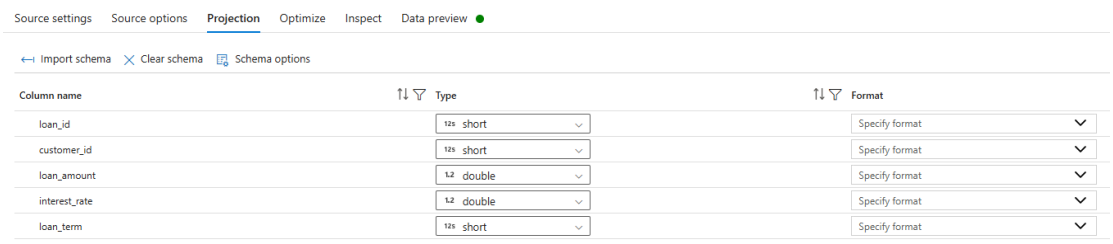
Source Options:

File path for loans.csv file from the bronze folder in ADLS Gen 2

Tick the first row as a header

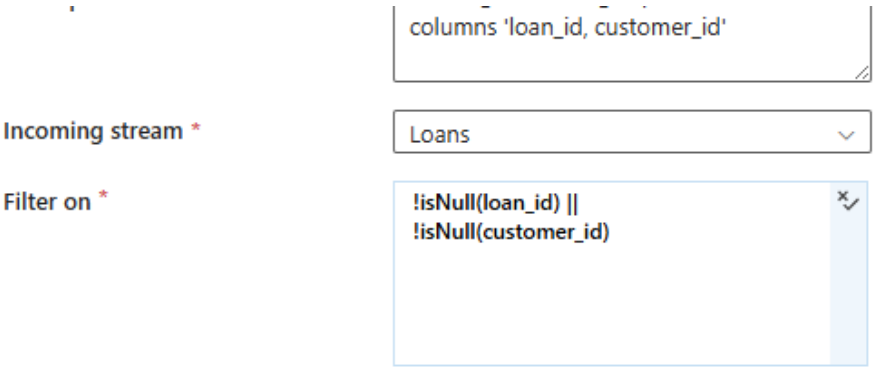


Projection: Click on Import Schema

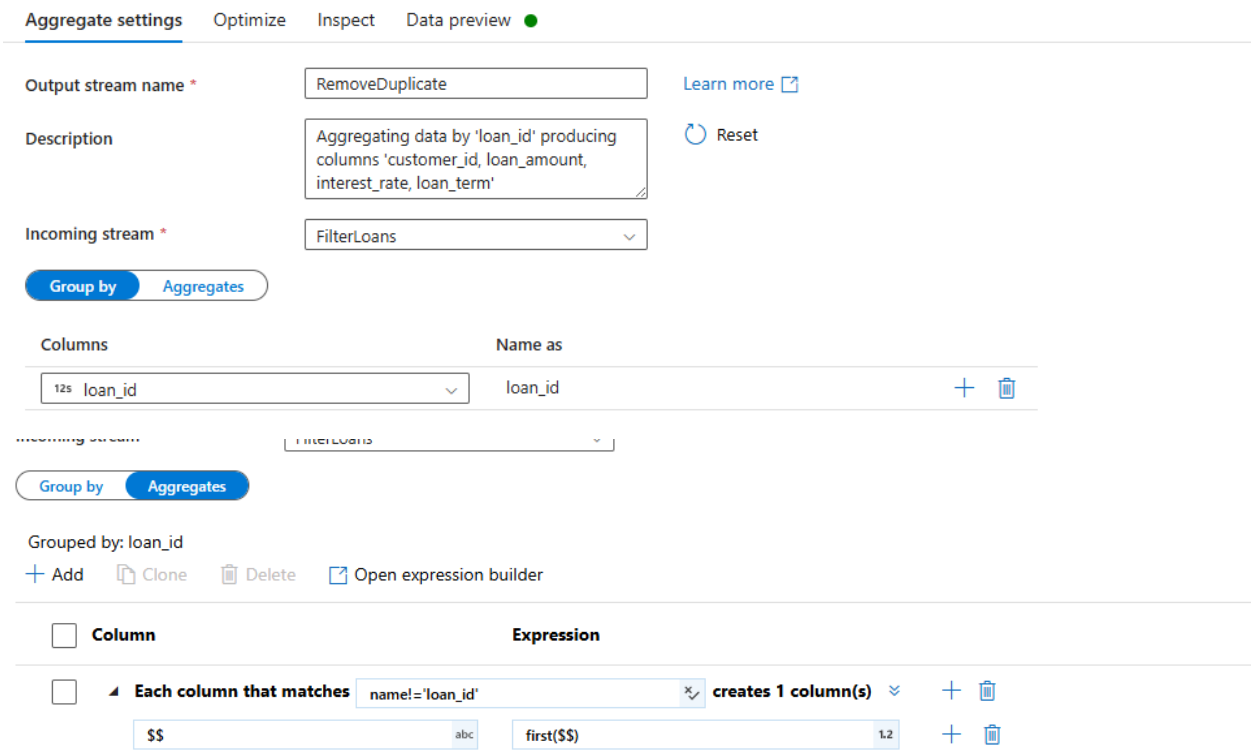


Click on the plus icon at the bottom of the source, and take a filter transformation

Use this expression: !isNull(loan_id) || !isNull(customer_id)

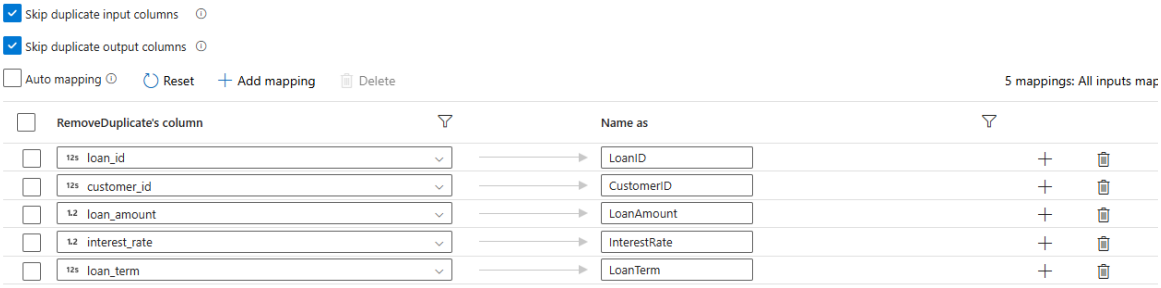


- **Aggregate Transformation (Remove Duplicate):**

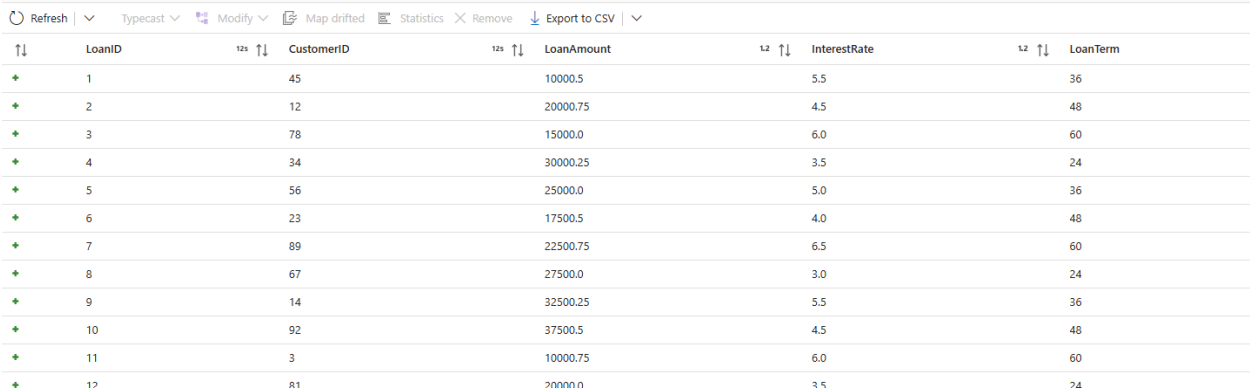


- **Select Transformation:**

  Rename the column name here



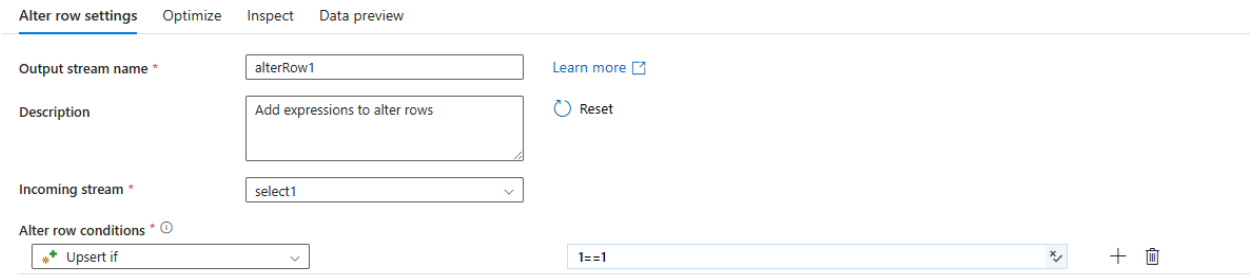
  Data Preview Tab:



LoanID	CustomerID	LoanAmount	InterestRate	LoanTerm
1	45	10000.5	5.5	36
2	12	20000.75	4.5	48
3	78	15000.0	6.0	60
4	34	30000.25	3.5	24
5	56	25000.0	5.0	36
6	23	17500.5	4.0	48
7	89	22500.75	6.5	60
8	67	27500.0	3.0	24
9	14	32500.25	5.5	36
10	92	37500.5	4.5	48
11	3	10000.75	6.0	60
12	81	20000.0	3.5	24

- **Alter Row Transformation:**

  Select Upsert if condition from the drop down and 1==1 expression

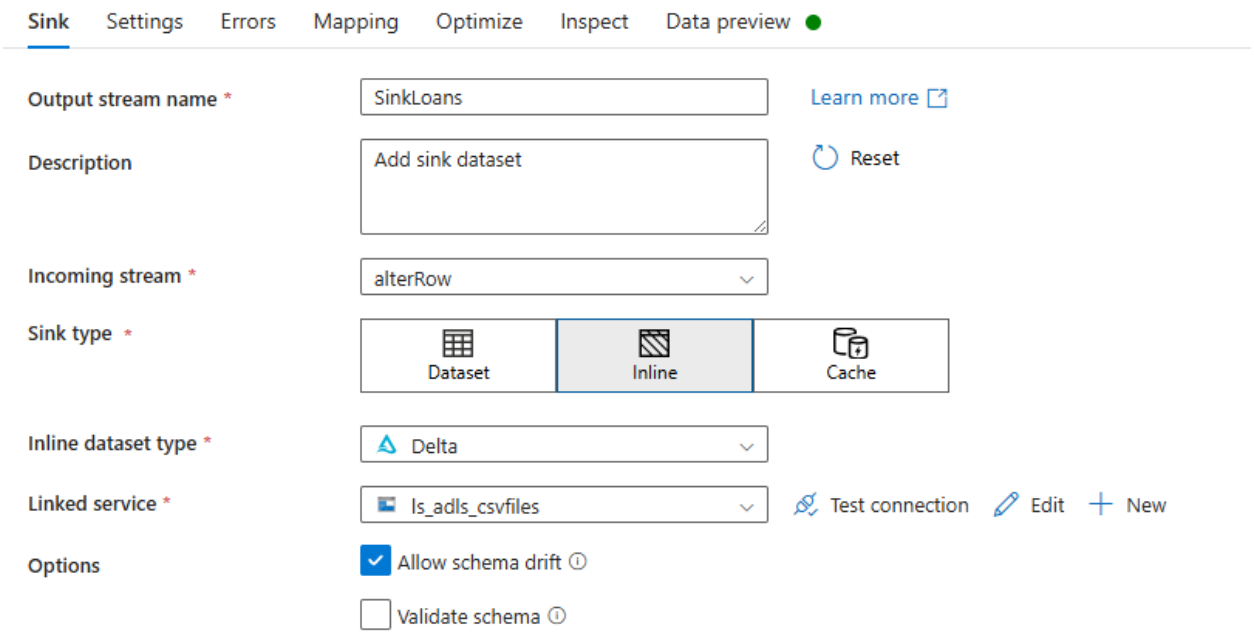


- **Sink Transformation:**

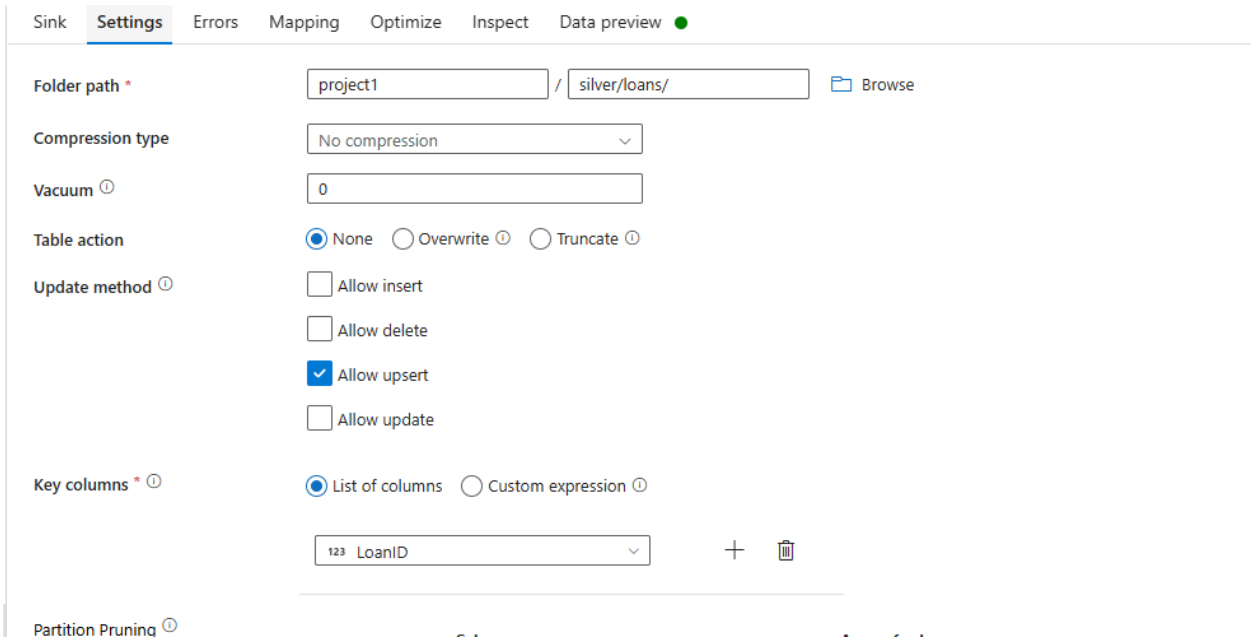
  Sink type: Inline

  Inline dataset type: Delta

  Linked service: ADLS Gen 2 storage account


Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Output stream name *
SinkLoans
Learn more
Description
Add sink dataset
Reset
Incoming stream *
alterRow
Sink type *
Dataset
Inline
Cache
Inline dataset type *
Delta
Linked service *
ls_adls_csvfiles
Test connection
Edit
New
Options
Allow schema drift
Validate schema

Settings:


Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Folder path *
project1
/
silver/loans/
Browse
Compression type
No compression
Vacuum
0
Table action
None
Overwrite
Truncate
Update method
Allow insert
Allow delete
Allow upsert
Allow update
Key columns *
List of columns
Custom expression
123 LoanID
Partition Pruning

Data Preview Tab:

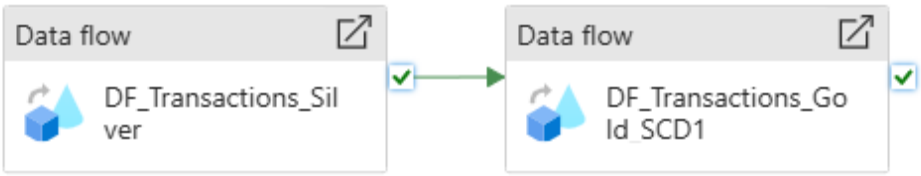
Refresh | Statistics | Export to CSV

LoanID	CustomerID	LoanAmount	InterestRate	LoanTerm
1	45	10000.5	5.5	36
2	12	20000.75	4.5	48
3	78	15000.0	6.0	60
4	34	30000.25	3.5	24
5	56	25000.0	5.0	36
6	23	17500.5	4.0	48
7	89	22500.75	6.5	60
8	67	27500.0	3.0	24
9	14	32500.25	5.5	36
10	92	37500.5	4.5	48
11	3	10000.75	6.0	60
12	81	20000.0	3.5	24
13	29	15000.25	5.0	36

That's it for the silver layer loans' data flow.

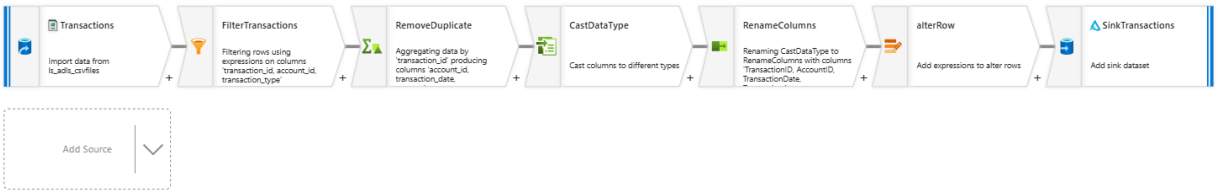
- **Transactions File Pipeline**

Drag and drop two dataflows in the canvas area



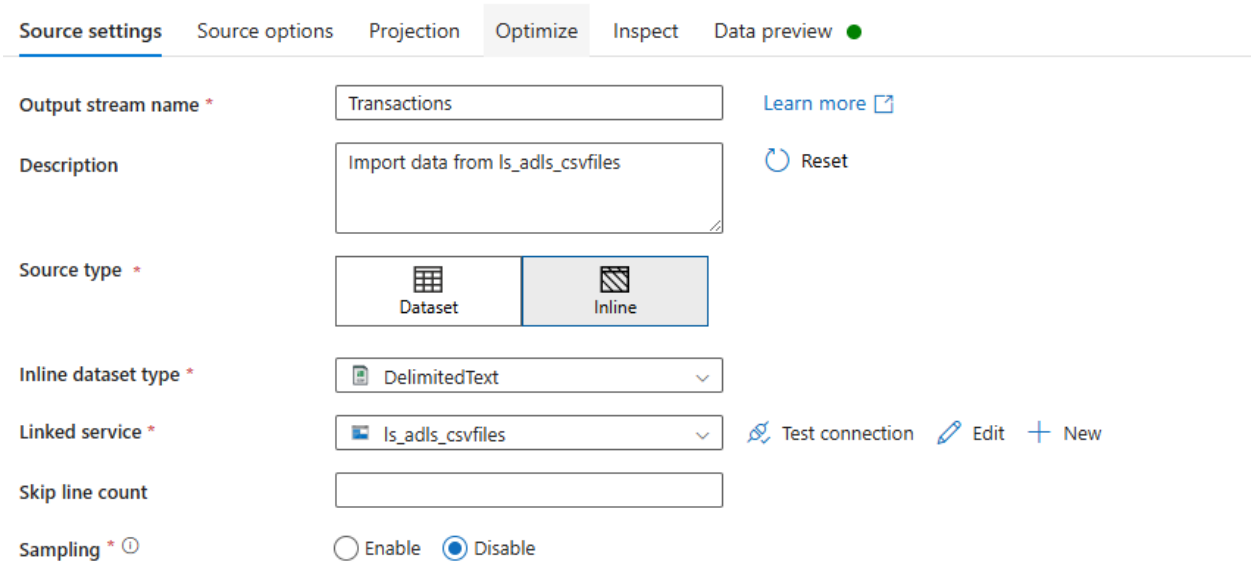
First, we will go through the silver layer in this.

Here is the entire flow:



Source:

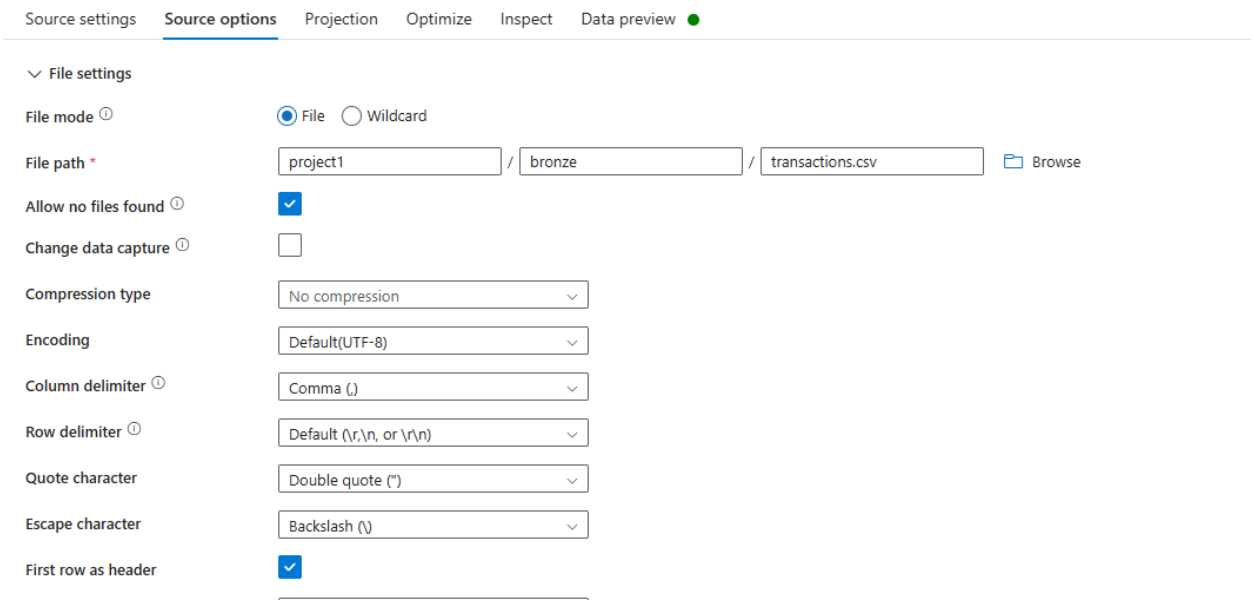
Source Settings:



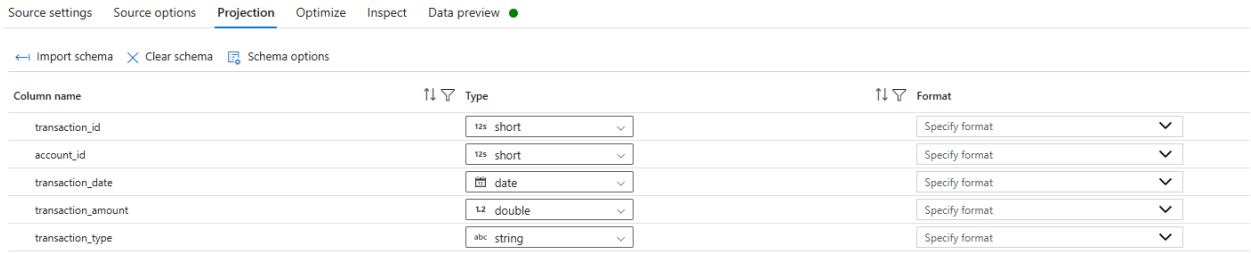
Source Options:

File path for the transactions.csv file from the bronze folder in ADLS Gen 2

Tick the first row as a header



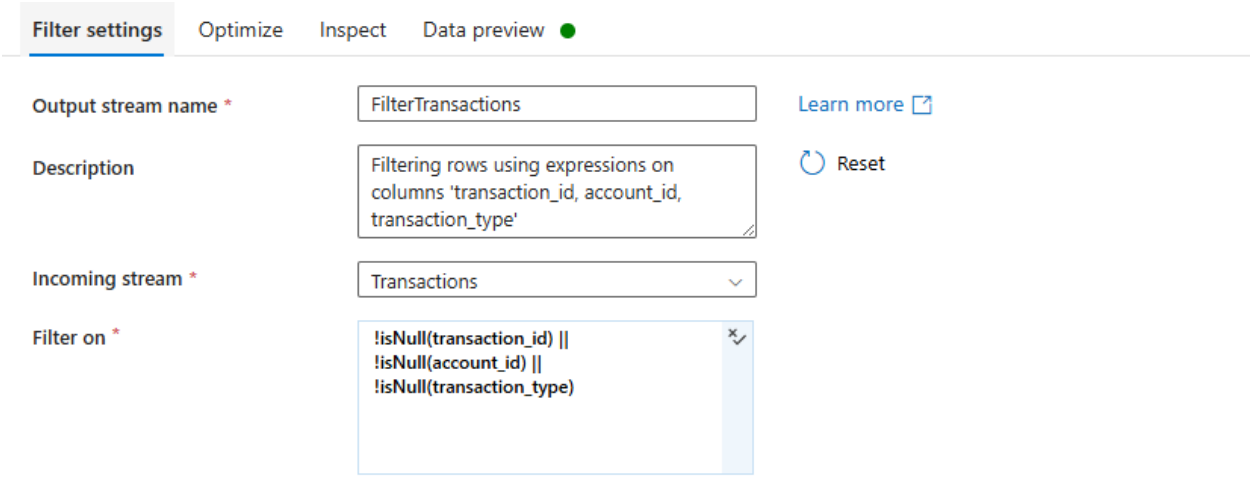
Projection: Click on Import Schema



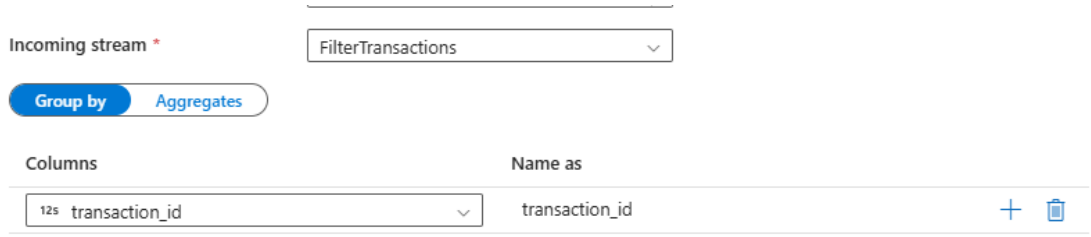
- **Filter transformation**

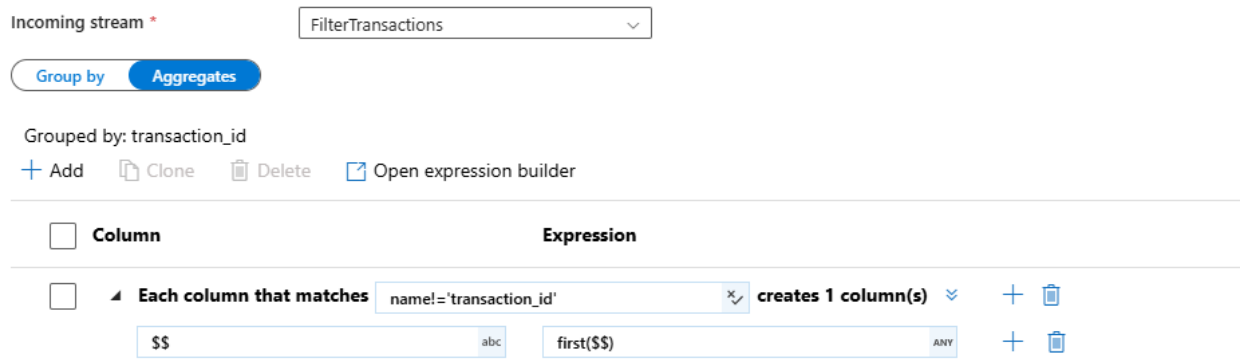
  Click on the plus icon at the bottom of the source, and take a filter transformation

  Use this expression: !isNull(transaction_id) || !isNull(account_id) || !isNull(transaction_type)

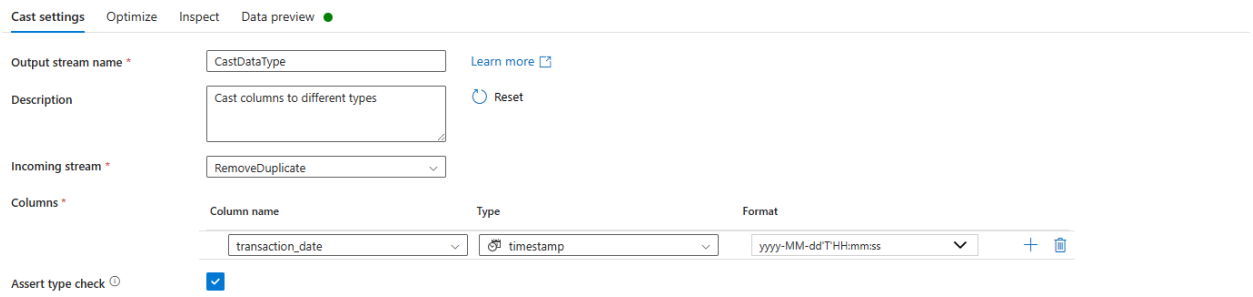


- **Aggregate Transformation (Remove Duplicate):**



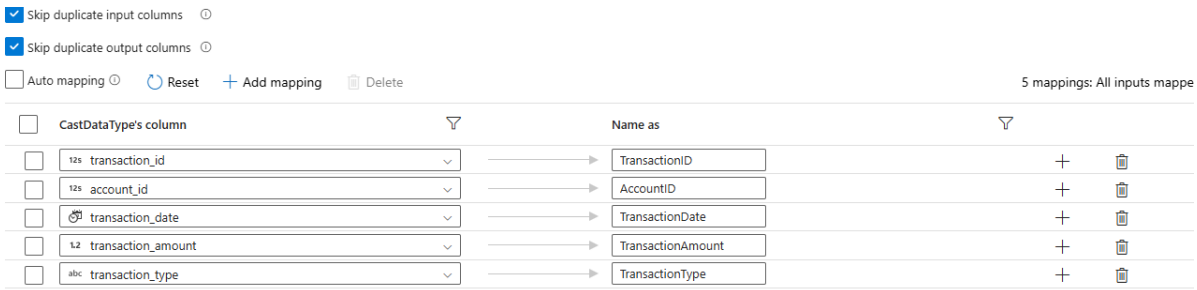


- **Cast Transformation:**



- **Select Transformation:**

Rename the column name here

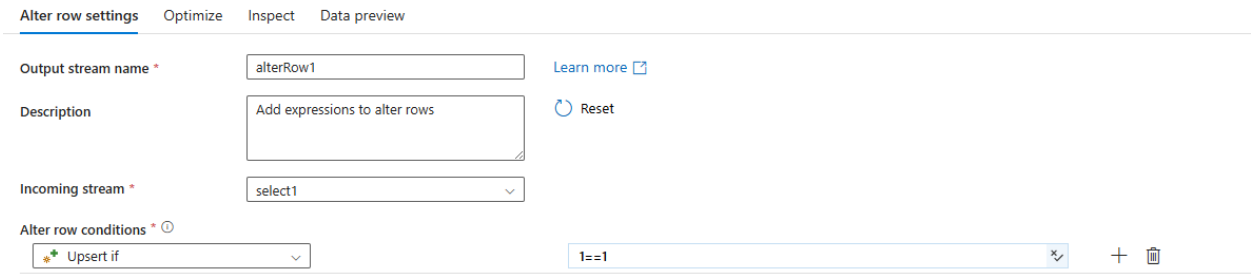


Data Preview Tab:

TransactionID	AccountID	TransactionDate	TransactionAmount	TransactionType
1	45	2024-01-01 00:00:00.000	100.5	Deposit
2	12	2024-01-02 00:00:00.000	200.75	Withdrawal
3	78	2024-01-03 00:00:00.000	150.0	Deposit
4	34	2024-01-04 00:00:00.000	300.25	Withdrawal
5	56	2024-01-05 00:00:00.000	250.0	Deposit
6	23	2024-01-06 00:00:00.000	175.0	Withdrawal
7	89	2024-01-07 00:00:00.000	225.5	Deposit
8	67	2024-01-08 00:00:00.000	275.75	Withdrawal
9	14	2024-01-09 00:00:00.000	325.0	Deposit
10	92	2024-01-10 00:00:00.000	375.25	Withdrawal
11	3	2024-01-11 00:00:00.000	100.5	Deposit
12	81	2024-01-12 00:00:00.000	200.75	Withdrawal
13	29	2024-01-13 00:00:00.000	150.0	Deposit

- **Alter Row Transformation:**

  Select Upsert if condition from the drop down and 1==1 expression

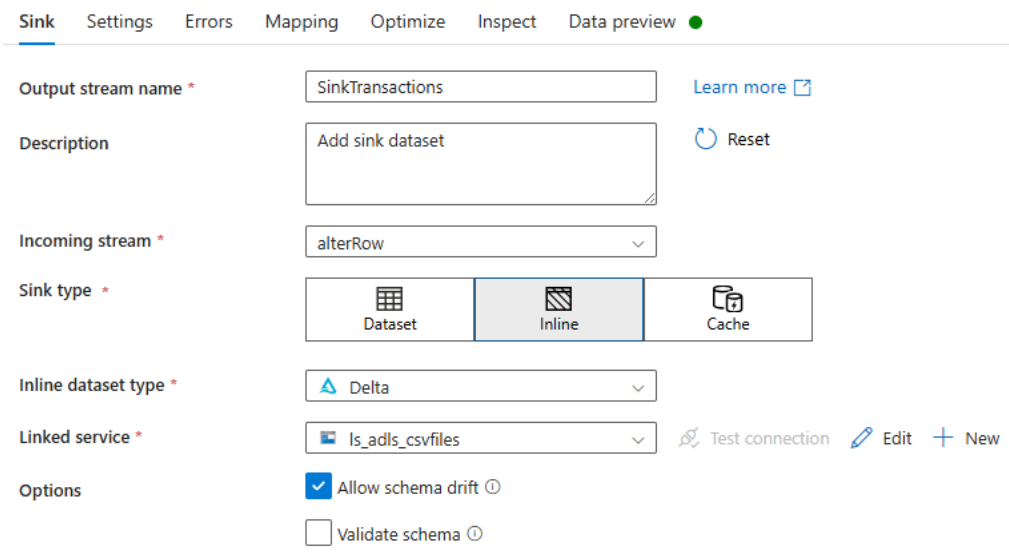


- **Sink:**

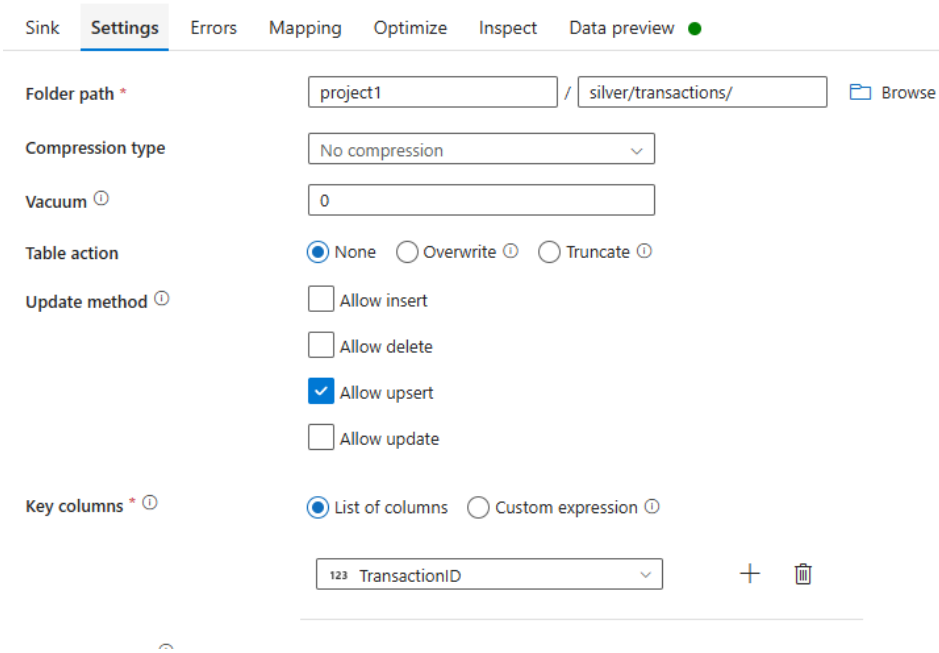
  Sink type: Inline

  Inline dataset type: Delta

  Linked service: ADLS Gen 2 storage account



Settings:



Data Preview Tab:

Refresh | Statistics | Export to CSV

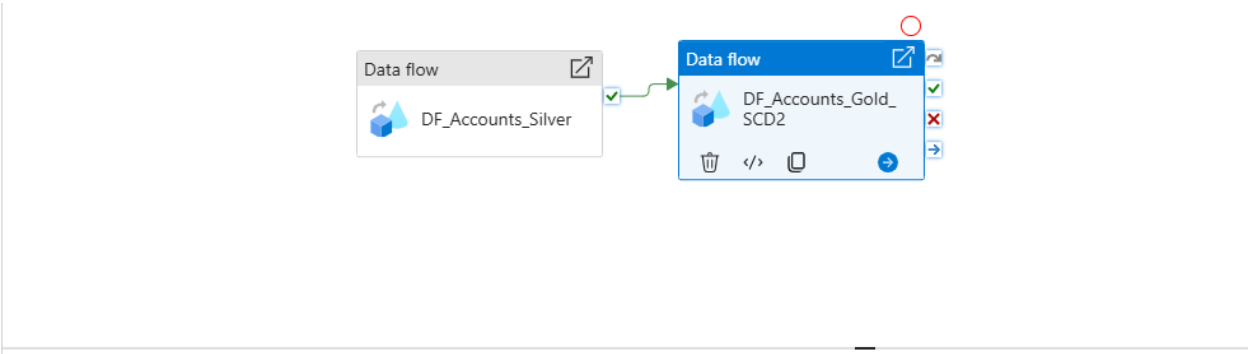
TransactionID	AccountID	TransactionDate	TransactionAmount	TransactionType
1	45	2024-01-01 00:00:00.000	100.5	Deposit
2	12	2024-01-02 00:00:00.000	200.75	Withdrawal
3	78	2024-01-03 00:00:00.000	150.0	Deposit
4	34	2024-01-04 00:00:00.000	300.25	Withdrawal
5	56	2024-01-05 00:00:00.000	250.0	Deposit
6	23	2024-01-06 00:00:00.000	175.0	Withdrawal
7	89	2024-01-07 00:00:00.000	225.5	Deposit
8	67	2024-01-08 00:00:00.000	275.75	Withdrawal
9	14	2024-01-09 00:00:00.000	325.0	Deposit
10	92	2024-01-10 00:00:00.000	375.25	Withdrawal
11	3	2024-01-11 00:00:00.000	100.5	Deposit
12	81	2024-01-12 00:00:00.000	200.75	Withdrawal

That's it for the silver layer transactions' data flow.

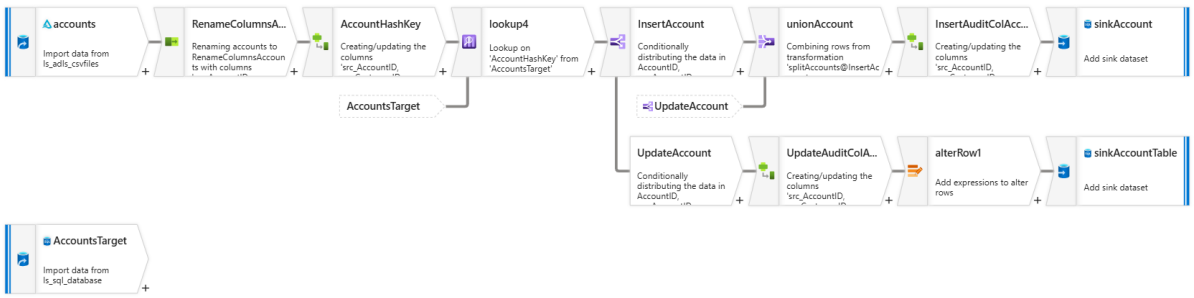
Step 3: Dataflows using SCD Type technique (SCD 1 and SCD 2)

Now, we will design a dataflow for the gold layer for all 5 files.

Select a new dataflow



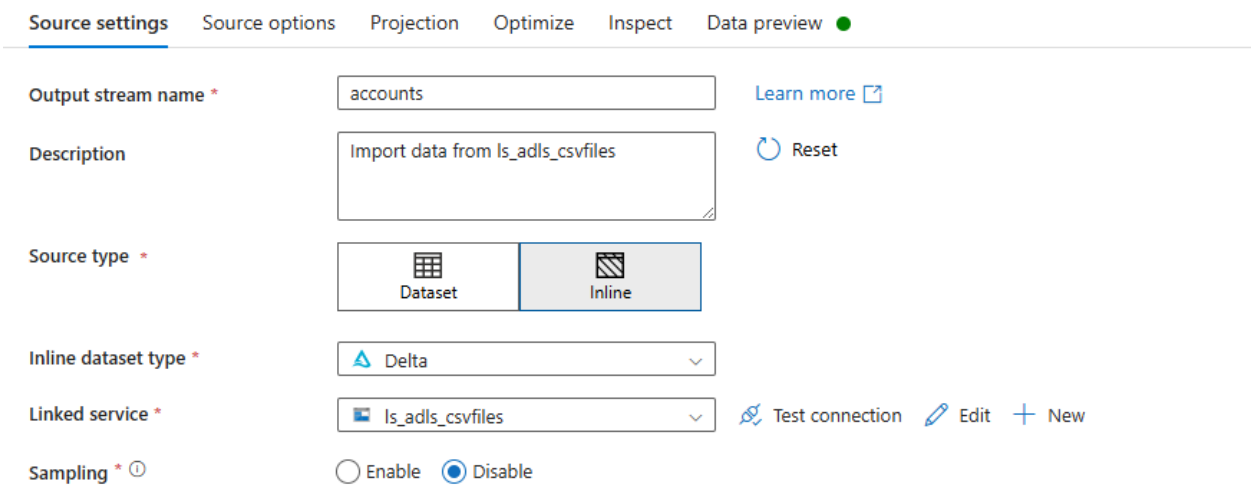
Here is the entire flow of the SCD 2 type accounts file



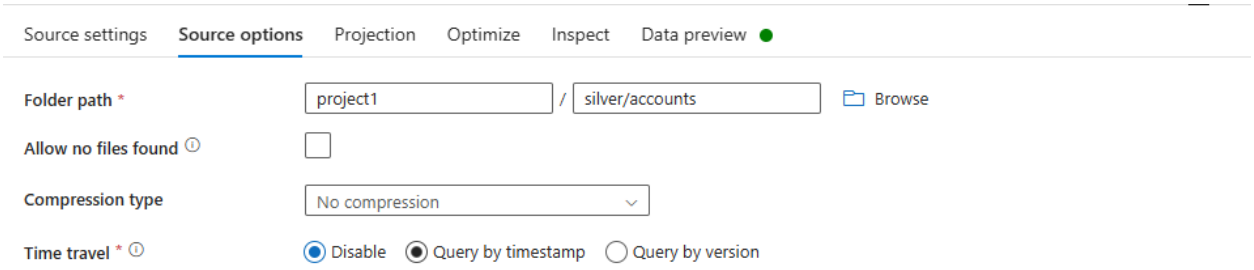
We will design the accounts file.

- **Source:**

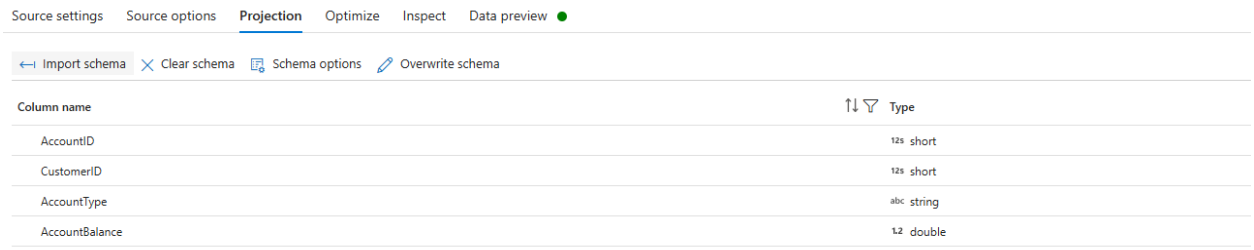
Source settings:



Source options:

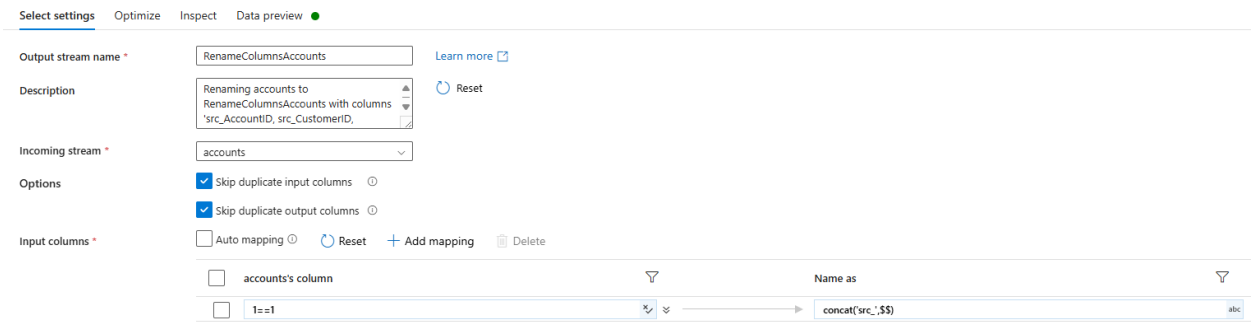


Projection:



- **Select Transformation:**

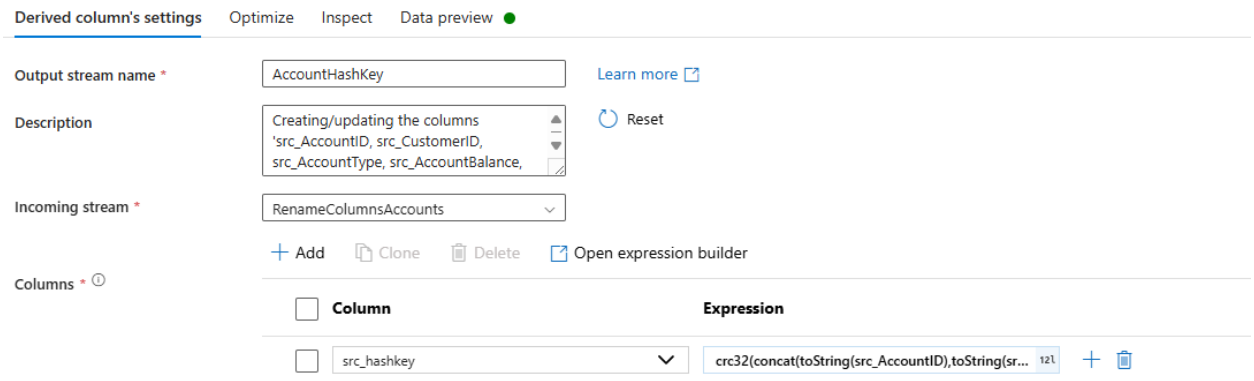
Use this expression: concat('src_',$$)



- **Derived column transformation**
  To create Hashkey

  Expression used:

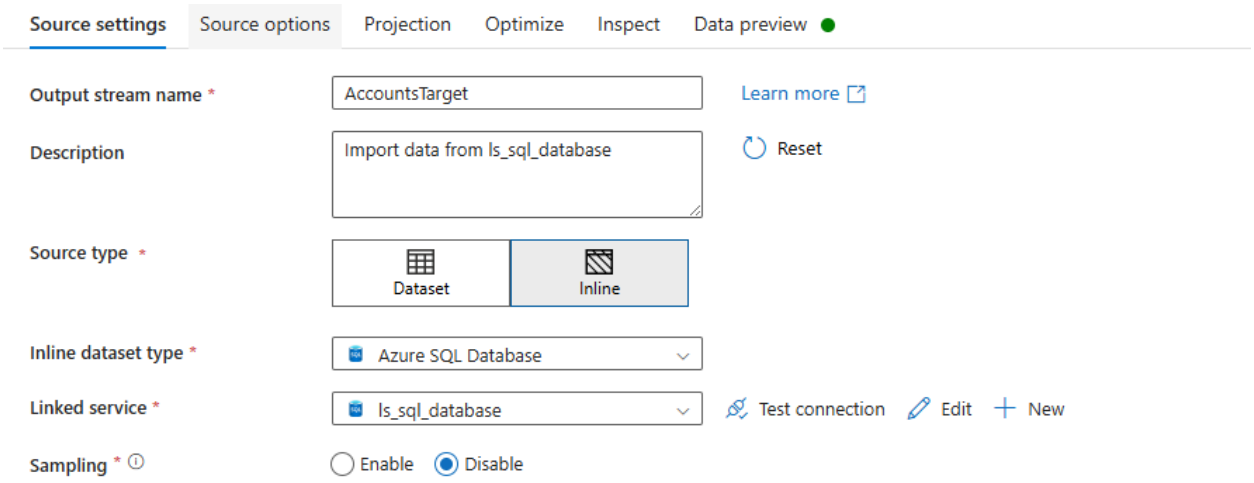
  crc32(concat(toString(src_AccountID),toString(src_CustomerID),src_AccountType,toString(src_AccountBalance)))



Create Target, take source and name it as Accounts Target:

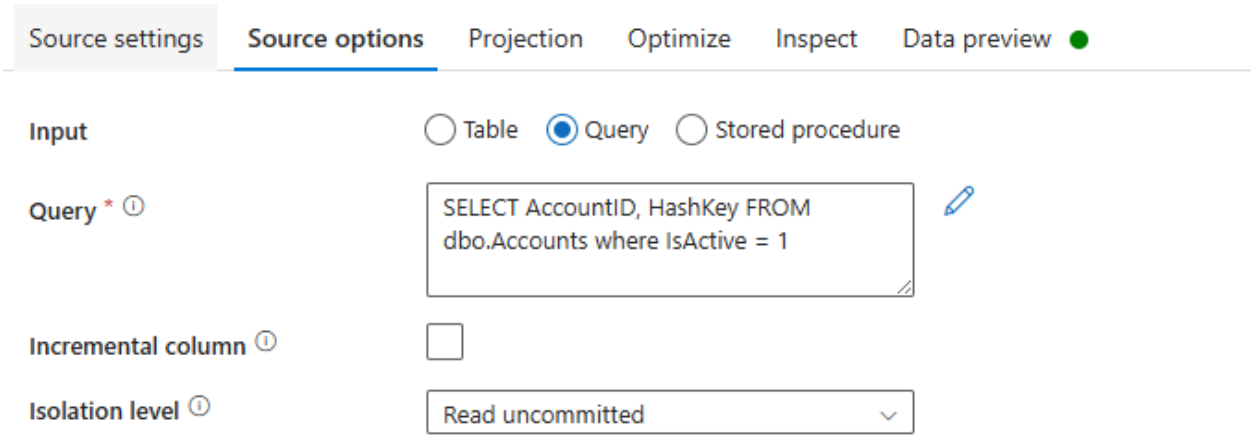
Inline Dataset: Azure SQL Database

Linked Service: Azure SQL Database

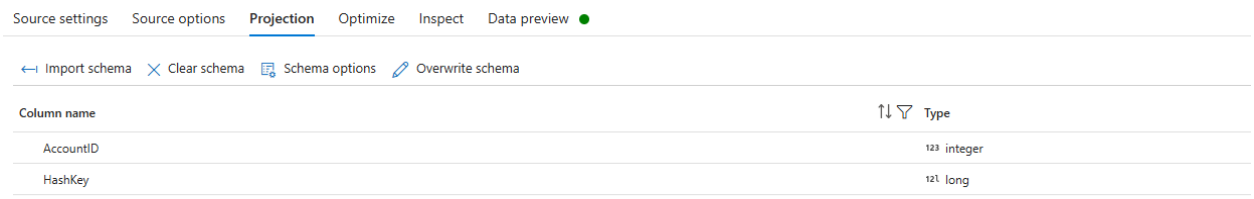


Query:

SELECT AccountID, HashKey FROM dbo.Accounts where IsActive = 1



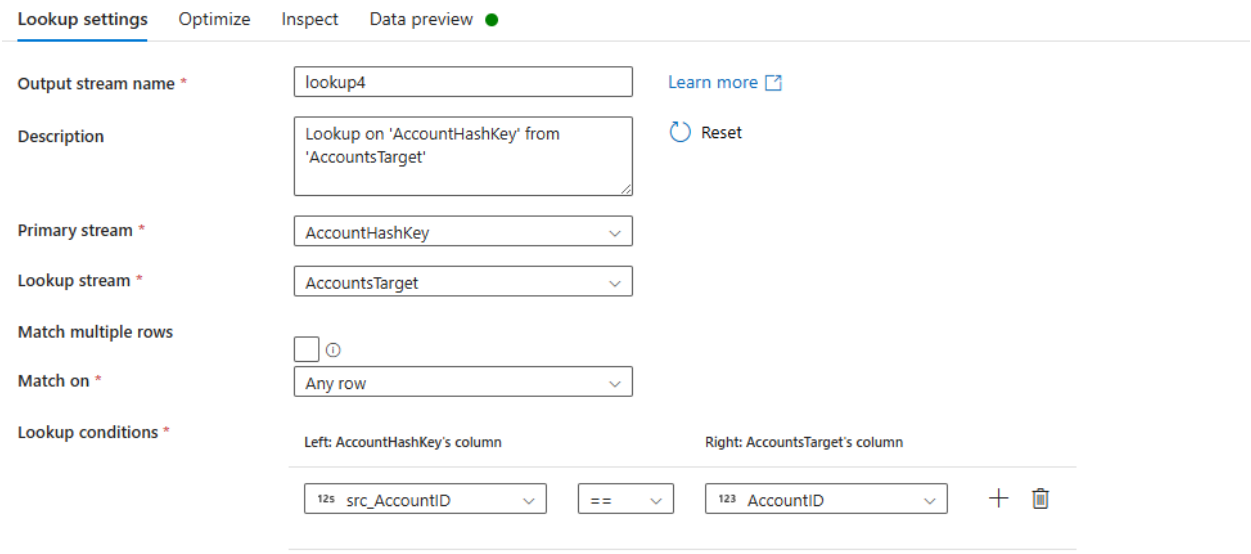
Projection:



- **Lookup transformation:**

  Primary Stream: AccountHashkey

Lookup Stream: AccountTarget



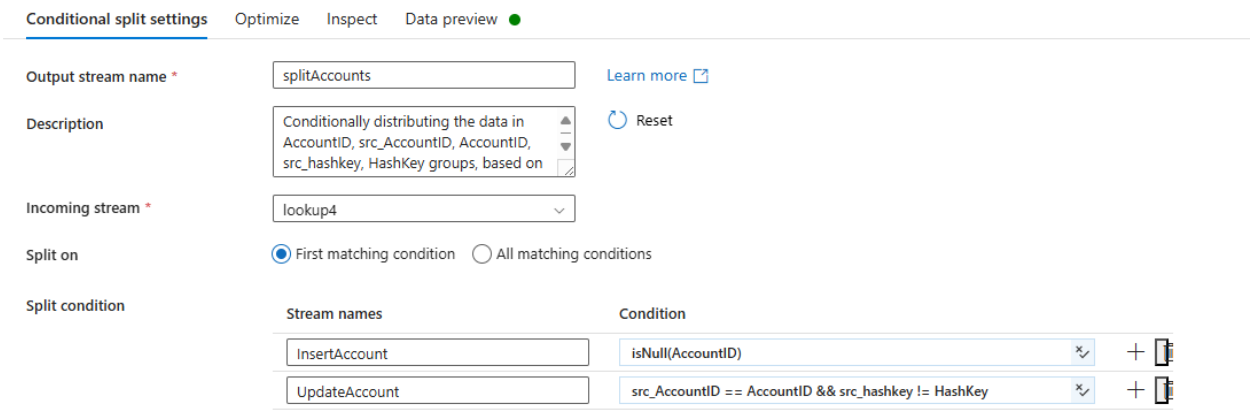
- **Conditional Split:**

  InsertAccount

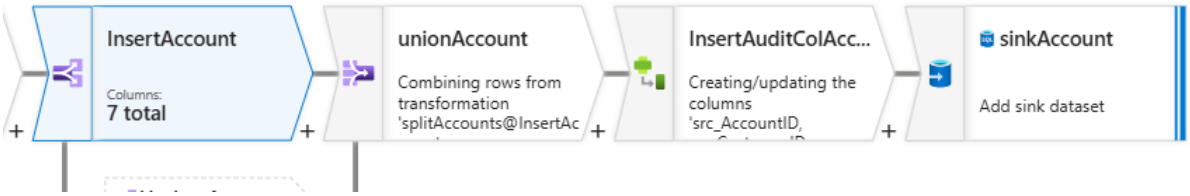
  Condition: isNull(AccountID)

  UpdateAccount

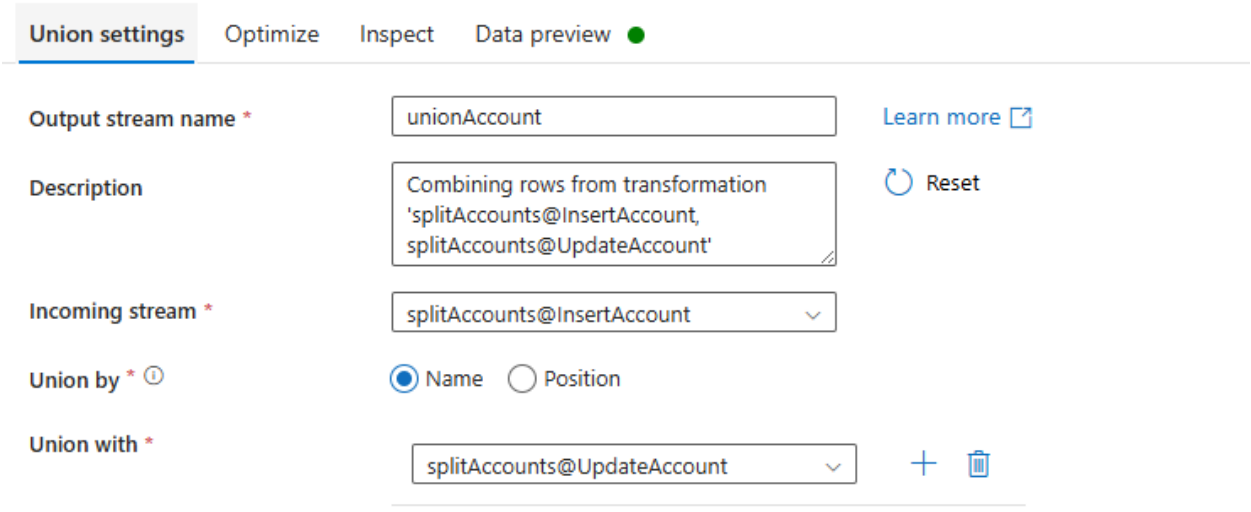
  Condition: src_AccountID == AccountID && src_hashkey != HashKey



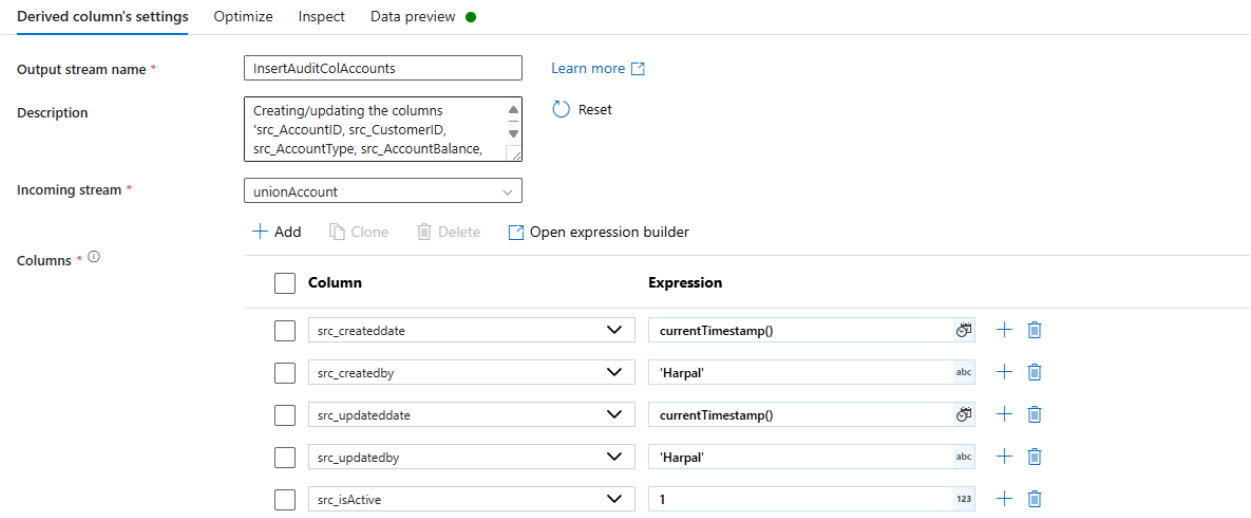
Here, we will complete the insert part.



- **Union Transformation:**

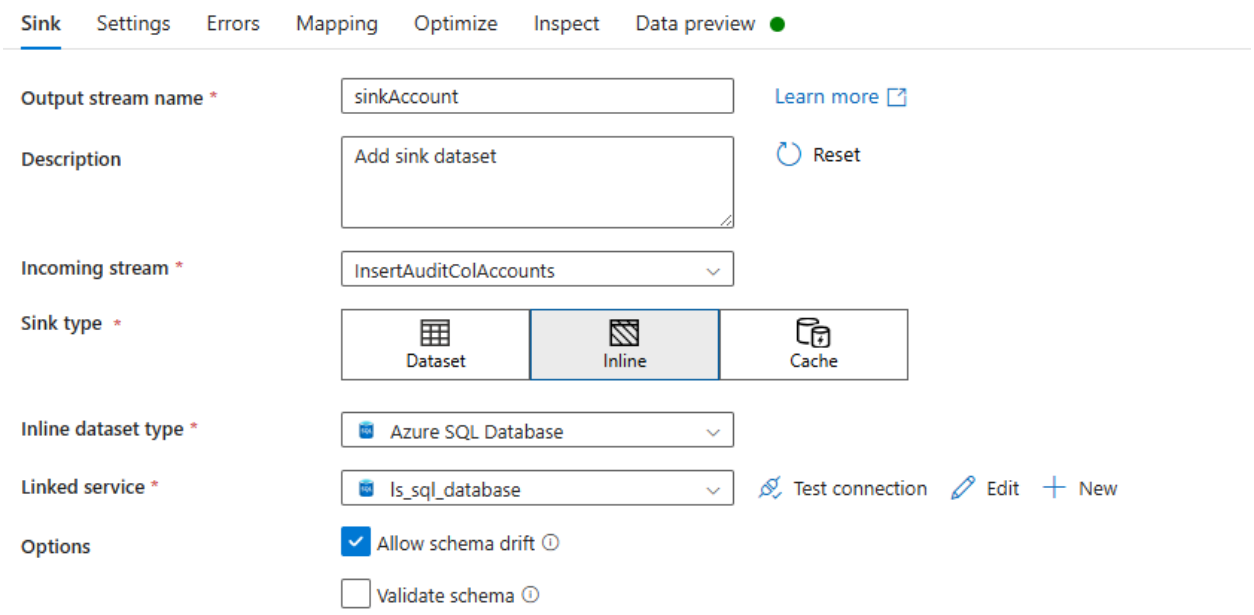


- **Derived Column Transformation:**

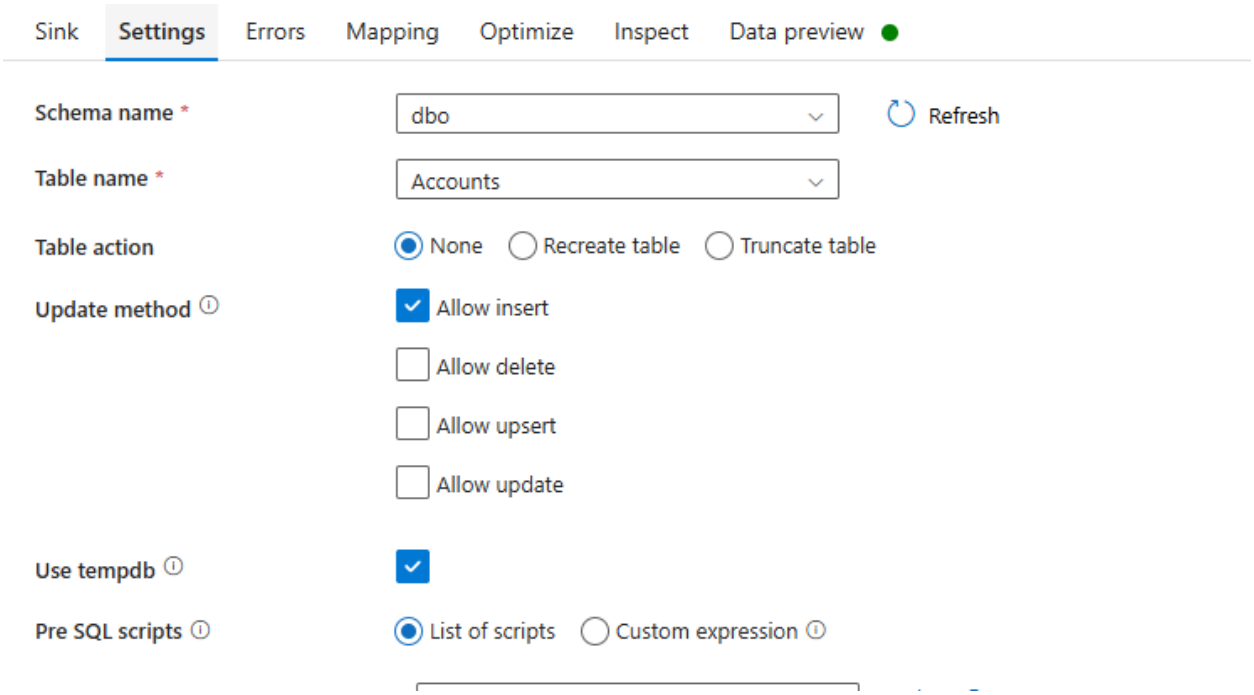


Column	Expression	Purpose
src_createddate	currentTimestamp()	Set current datetime when a **new row is inserted**
src_createdby	'Harpal'	Hardcoding the value to **who inserted** (can replace dynamically too later)
src_updateddate	currentTimestamp()	For new inserted records, updated date = created date (same initially)
src_updatedby	'Harpal'	Hardcoding updated by also (again, can improve later with parameters)
src_isActive	1	Mark new inserted record as **active** (important for SCD2 tracking)

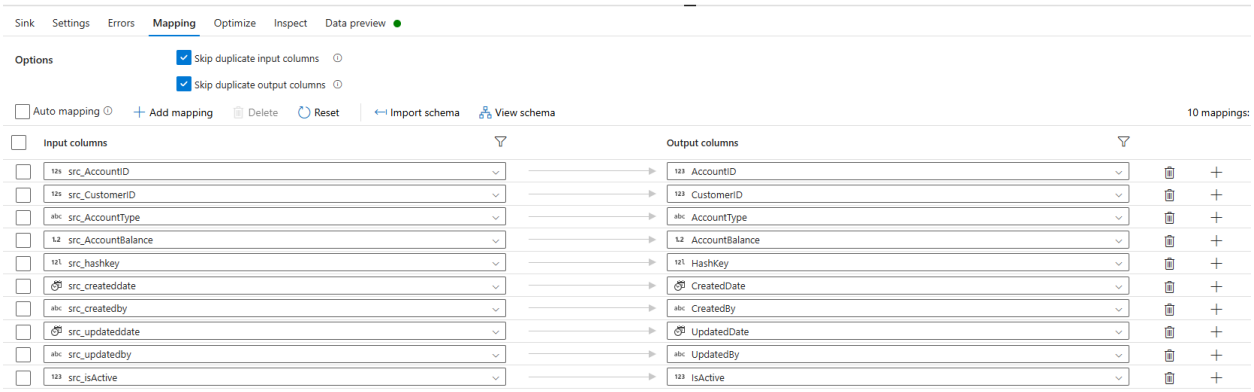
- **SinkAccount:**



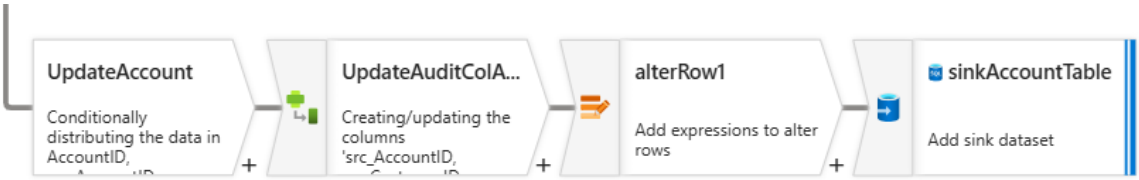
Settings:



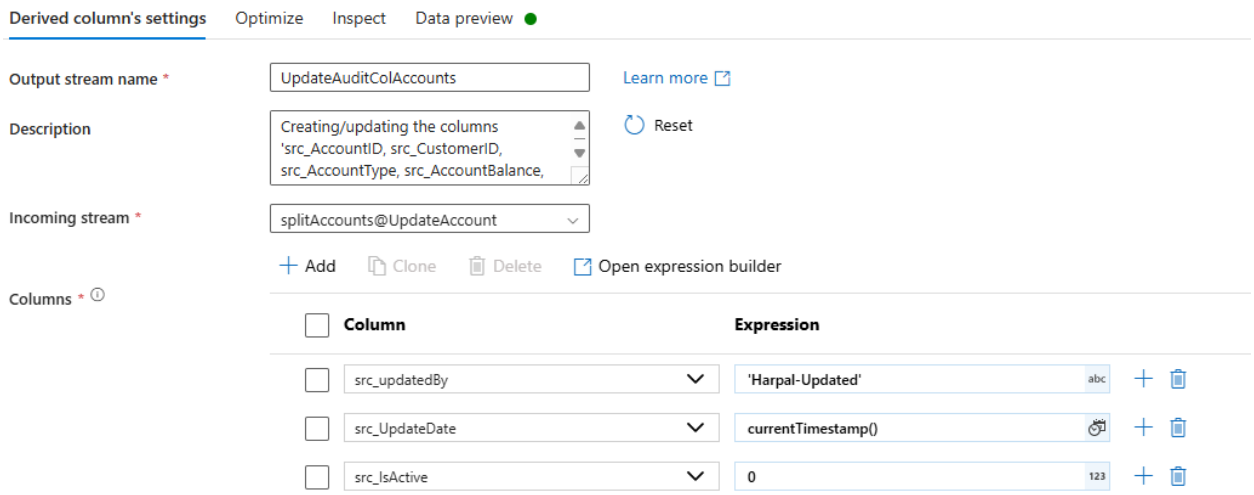
Mapping:



Now, let's do update part:

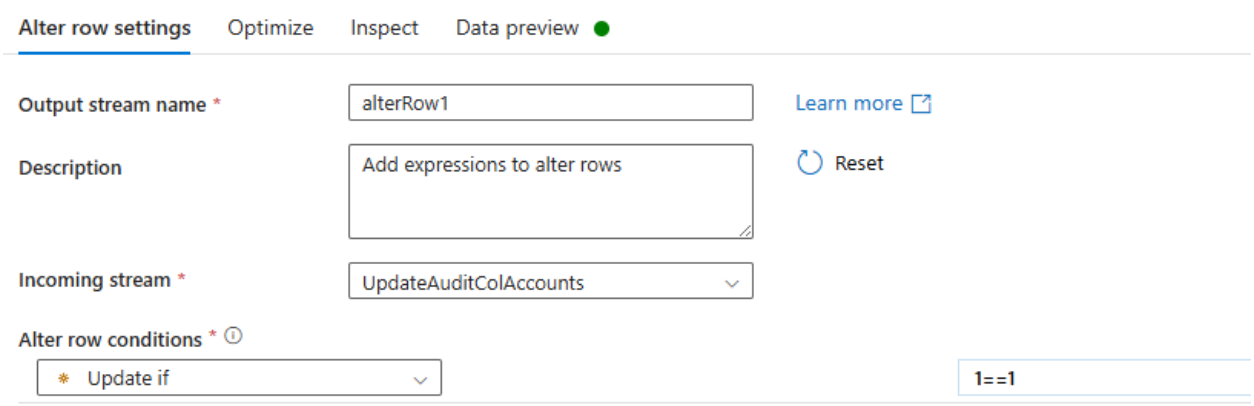


- **Derived column transformation:**

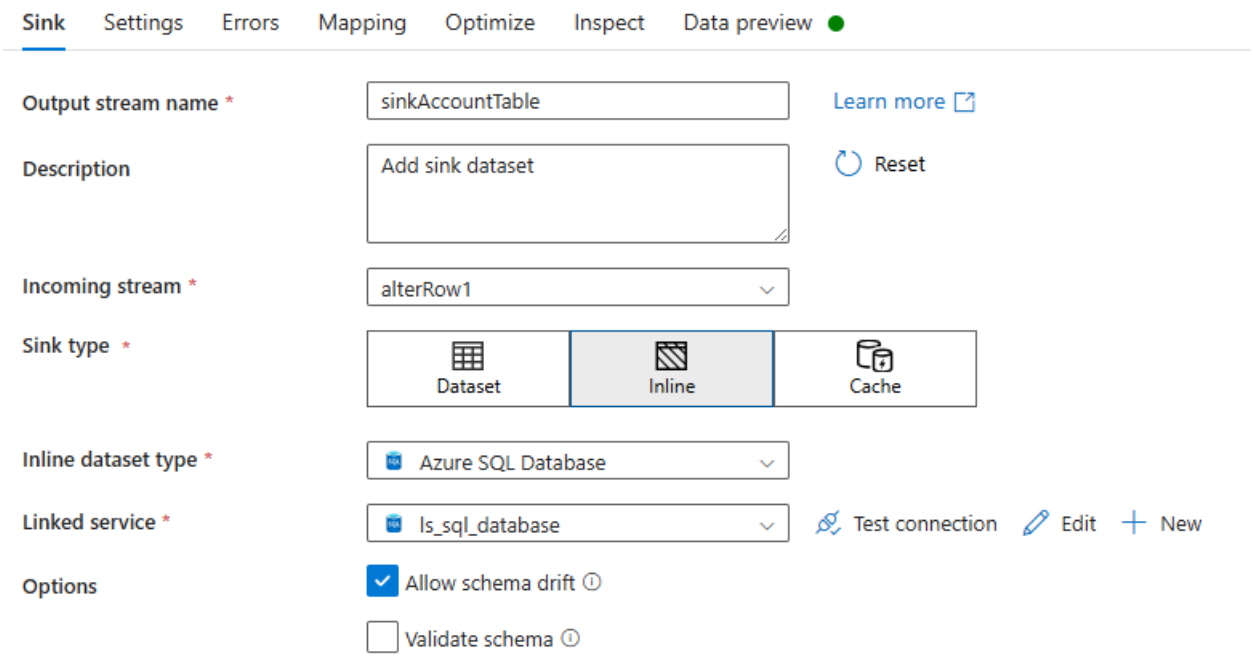


Column	Expression	Purpose
src_updatedBy	'Harpal-Updated'	Mark who updated the record (hardcoded for now, can be dynamic too)
src_updateDate	currentTimestamp()	Capture the update timestamp when record is expired (for SCD2)
src_isActive	0	Mark the old record as inactive (critical for SCD2 design)

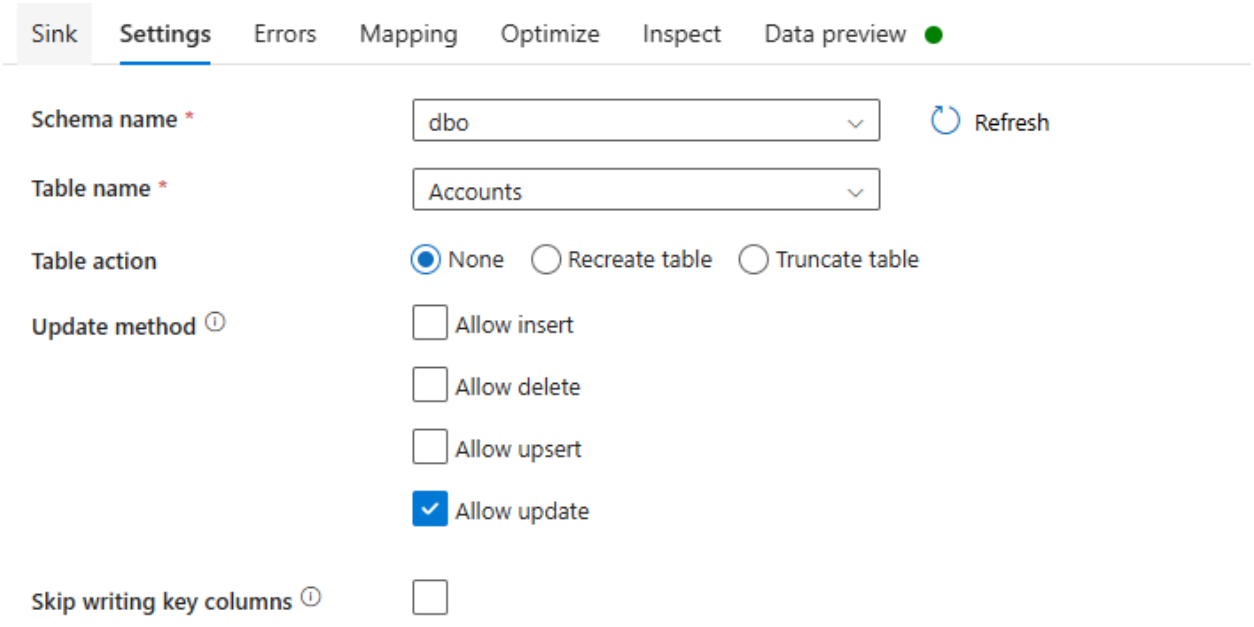
- **Alter Row Transformation:**

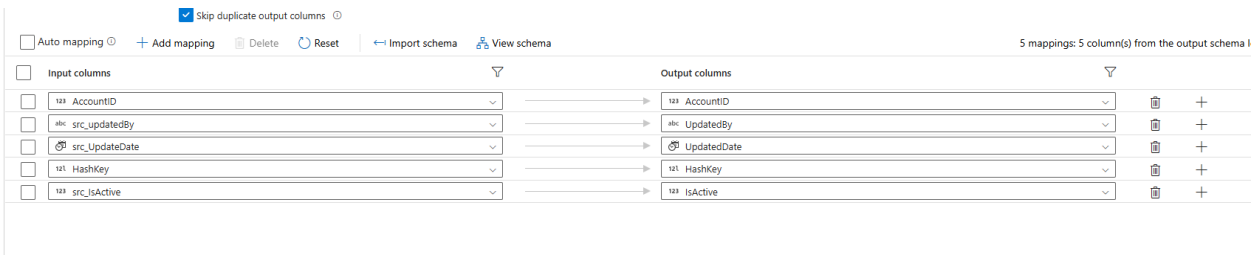


- **SinkAccountTable:**



Settings:

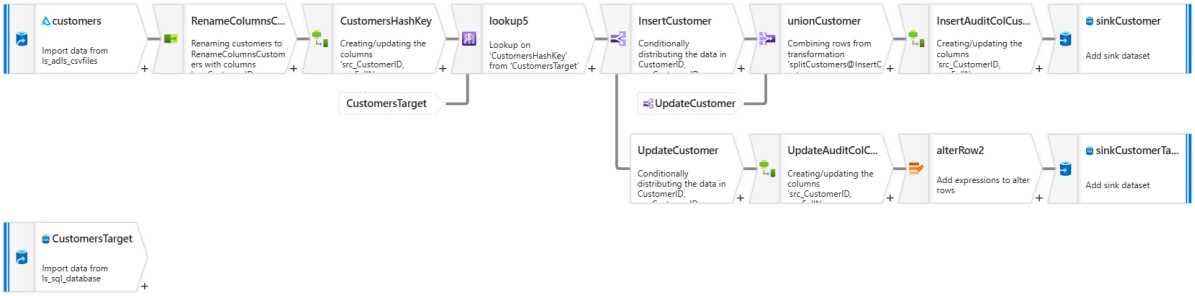




That's it for the SCD type 2 dataflow design for the accounts file.

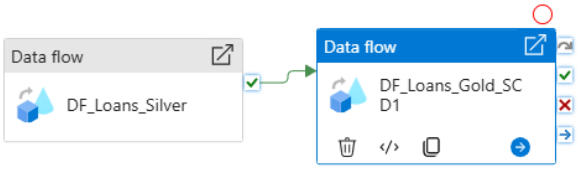
Similarly, we have to do for customers, as we will do 2 files as SCD 2 type and 3 files as SCD 1 type.

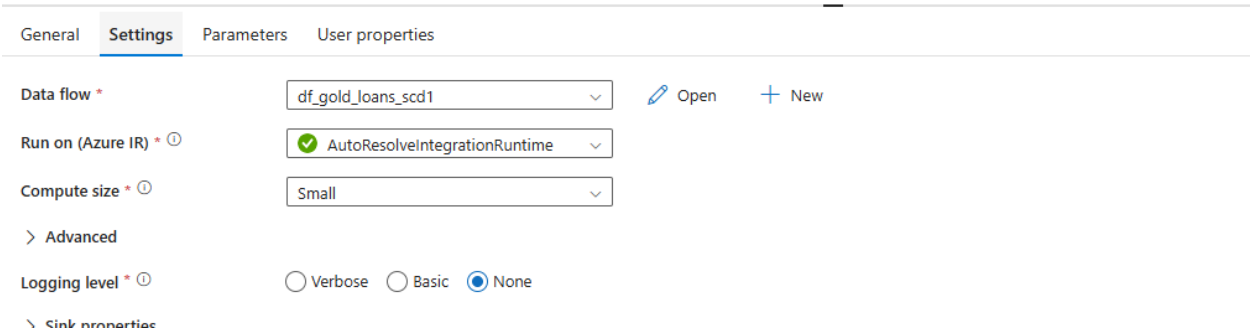
Here is the customers' dataflow for reference, we will not cover the steps here as it's similar to accounts (SCD 2 type)



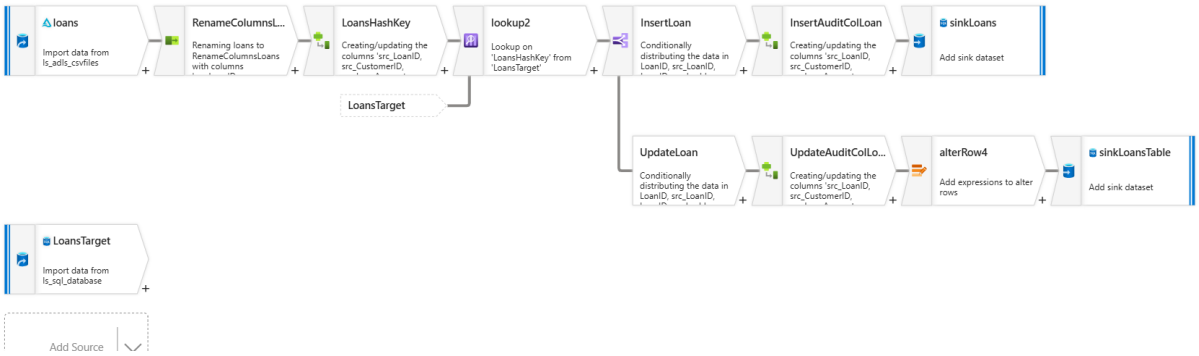
Now, let's do one file as **SCD Type 1**:

Let's take the loans file and design a dataflow for it.





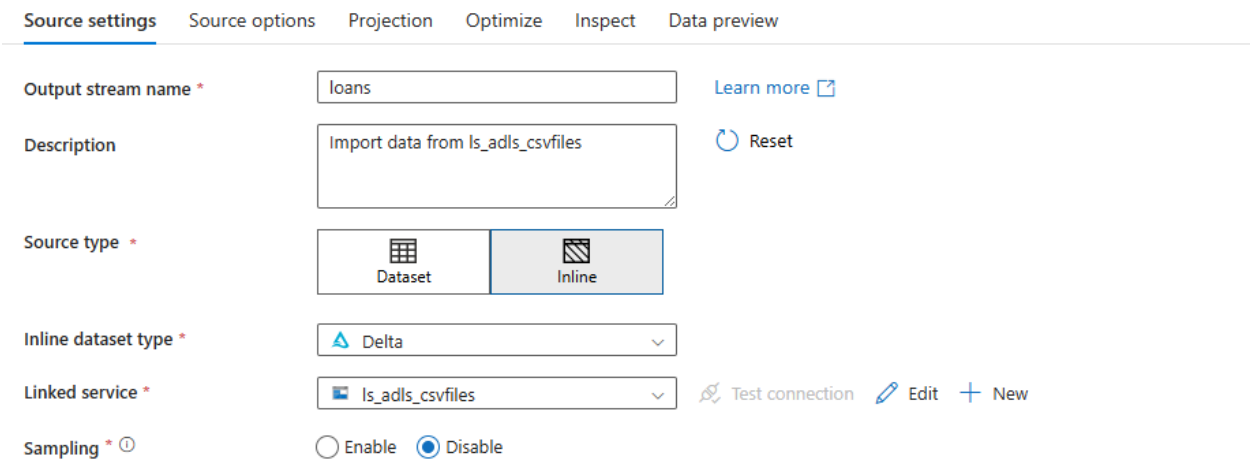
Dataflow design:



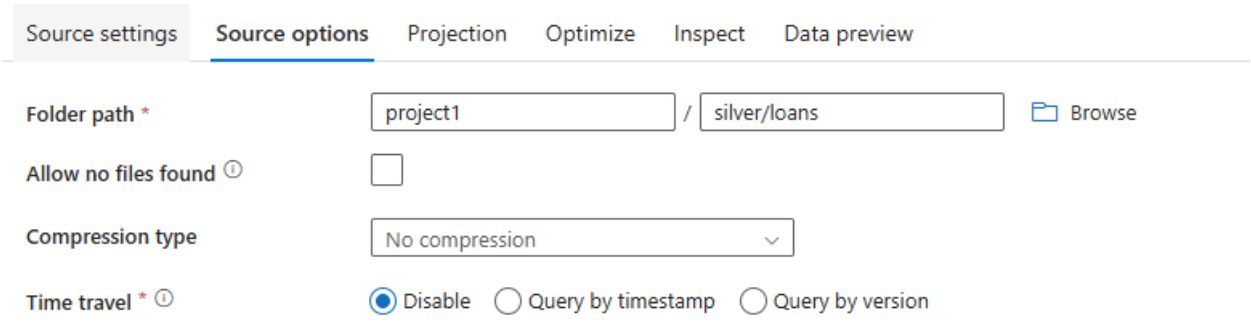
Let's design one by one transformation:

- **Source:**

  Source Settings:



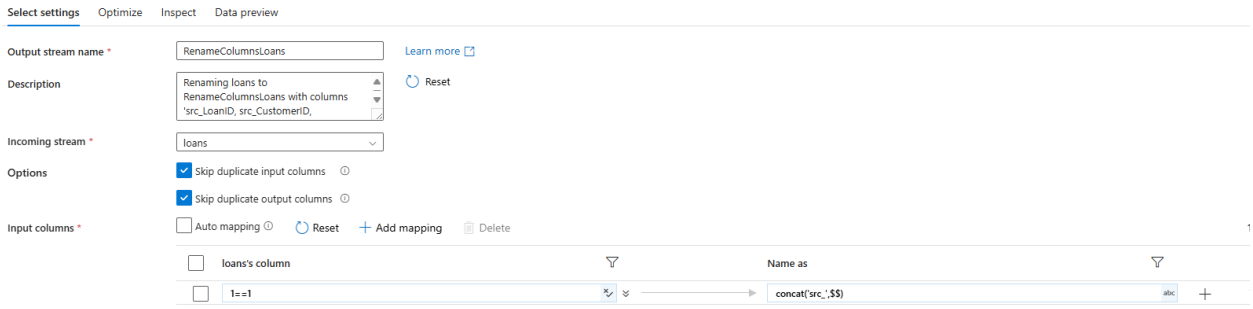
  Source Options:



- **Select Transformation**

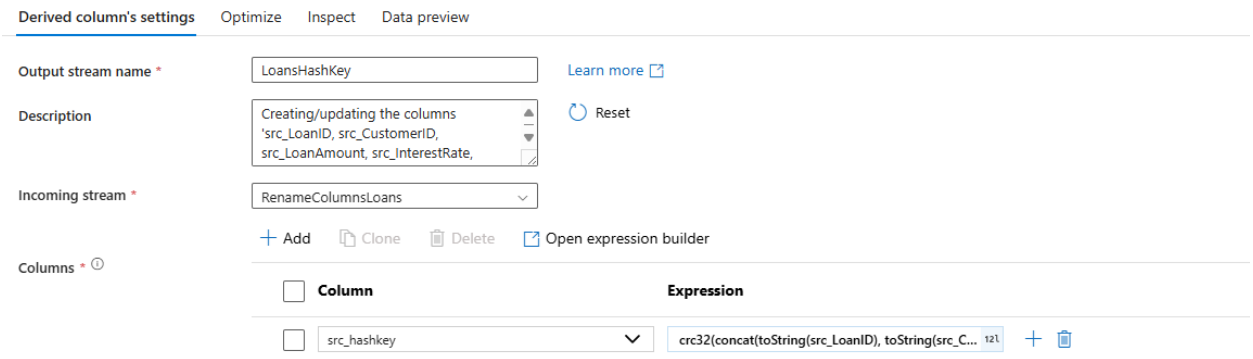
  Expression used:

concat('src_',$$)



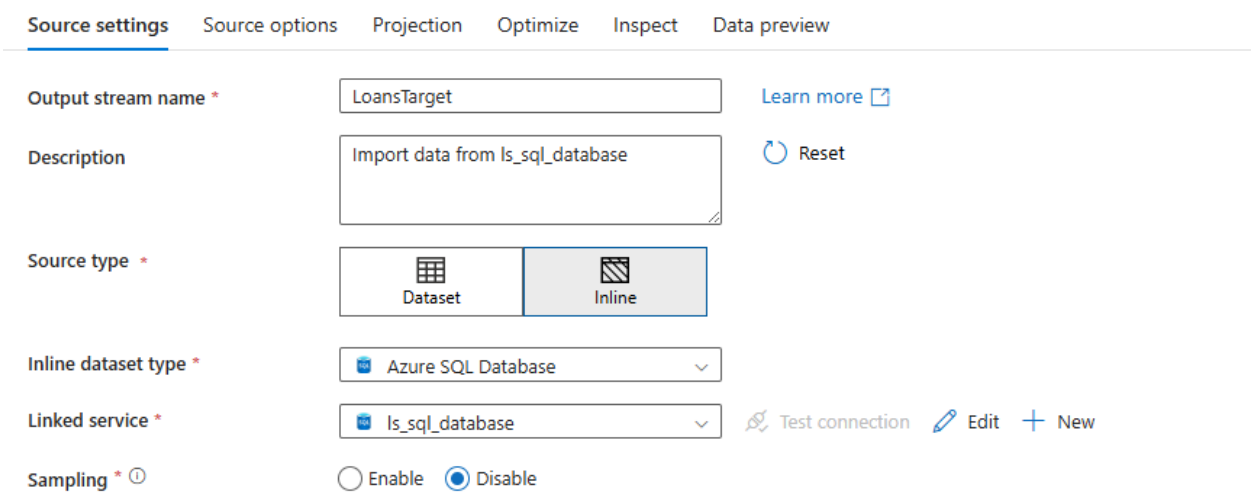
- **Derived Column Transformation:**

  crc32(concat(toString(src_LoanID), toString(src_CustomerID), toString(src_LoanAmount), toString(src_InterestRate),toString(src_LoanTerm)))



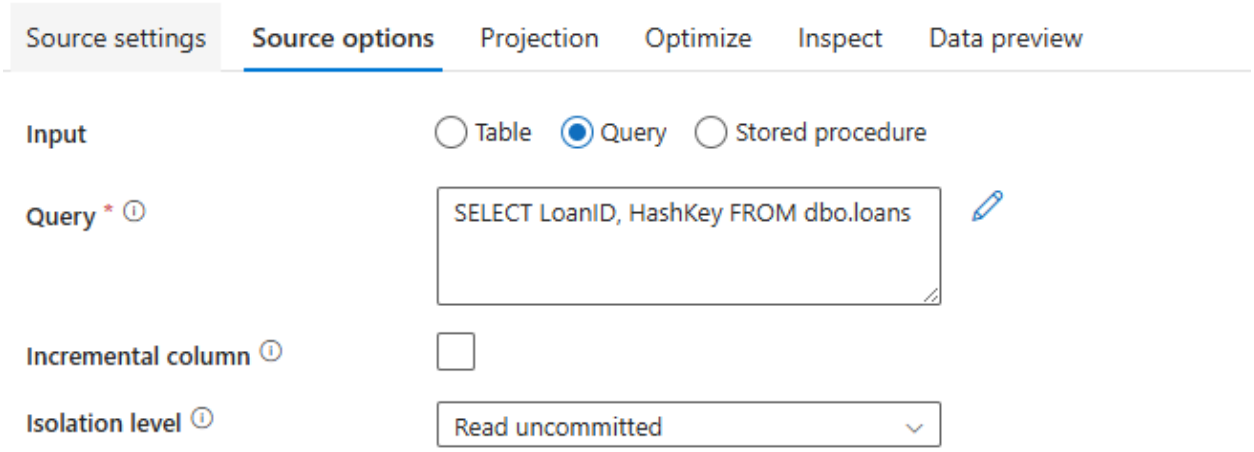
- **LoanTraget:**

  Source settings:

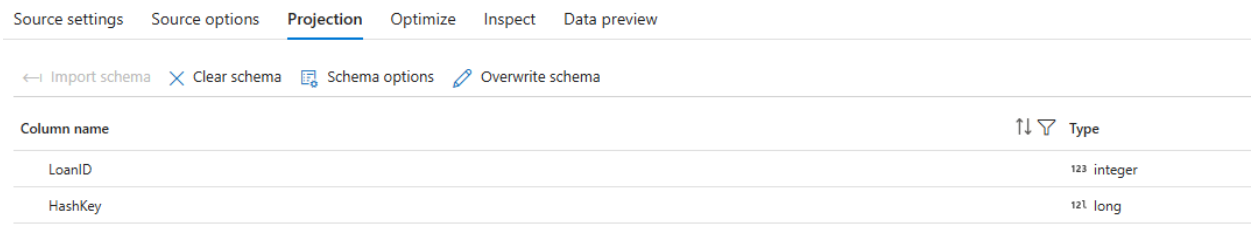


Source options:

SELECT LoanID, HashKey FROM dbo.loans



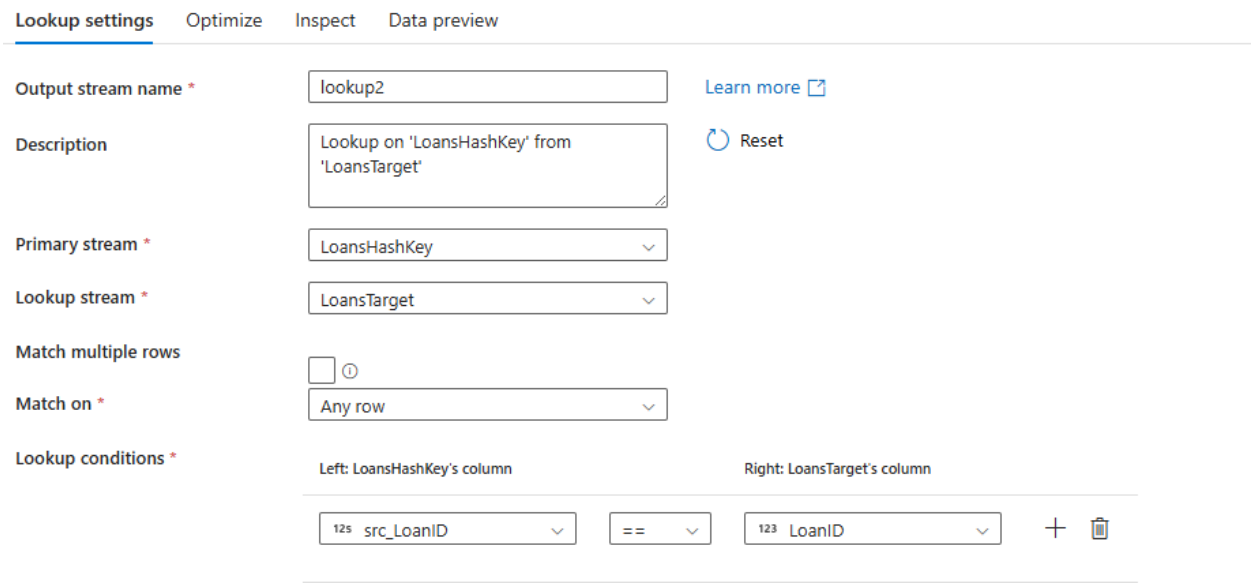
Projection:



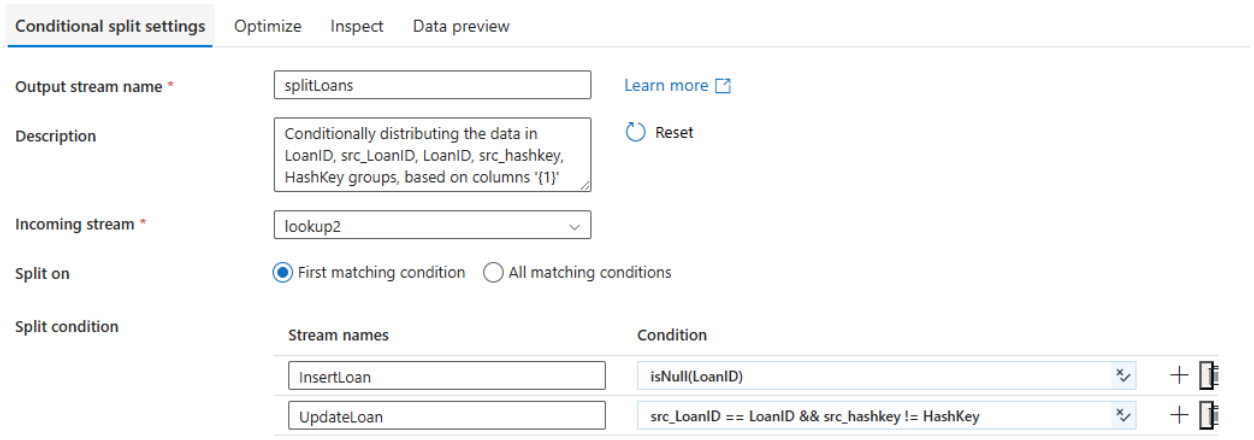
- **Lookup settings:**

  Primary Stream: LoansHashKey

Lookup Stream: LoansTarget

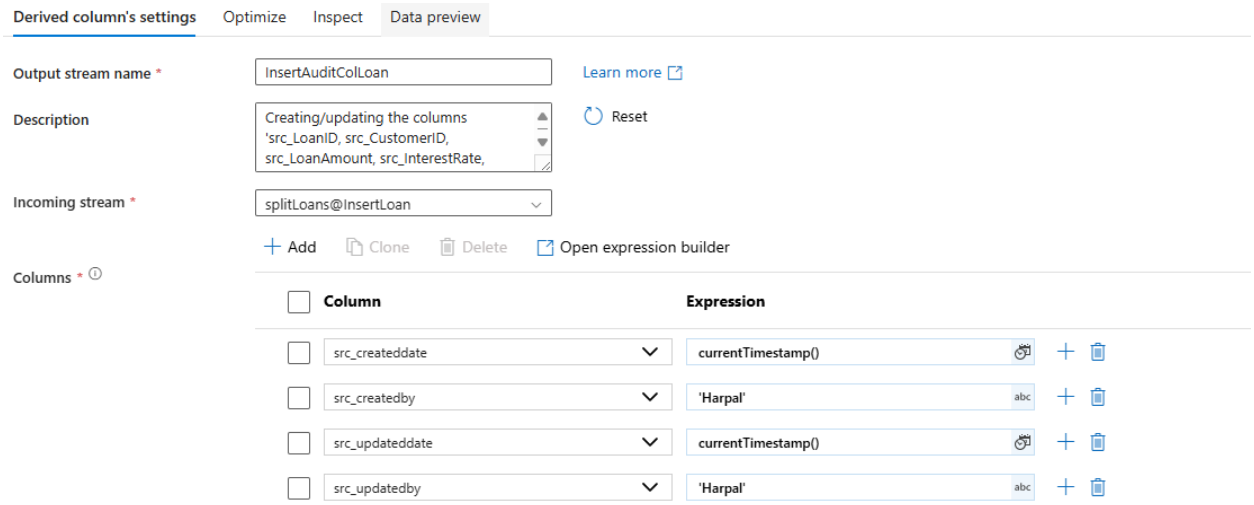


- **Conditional Split:**



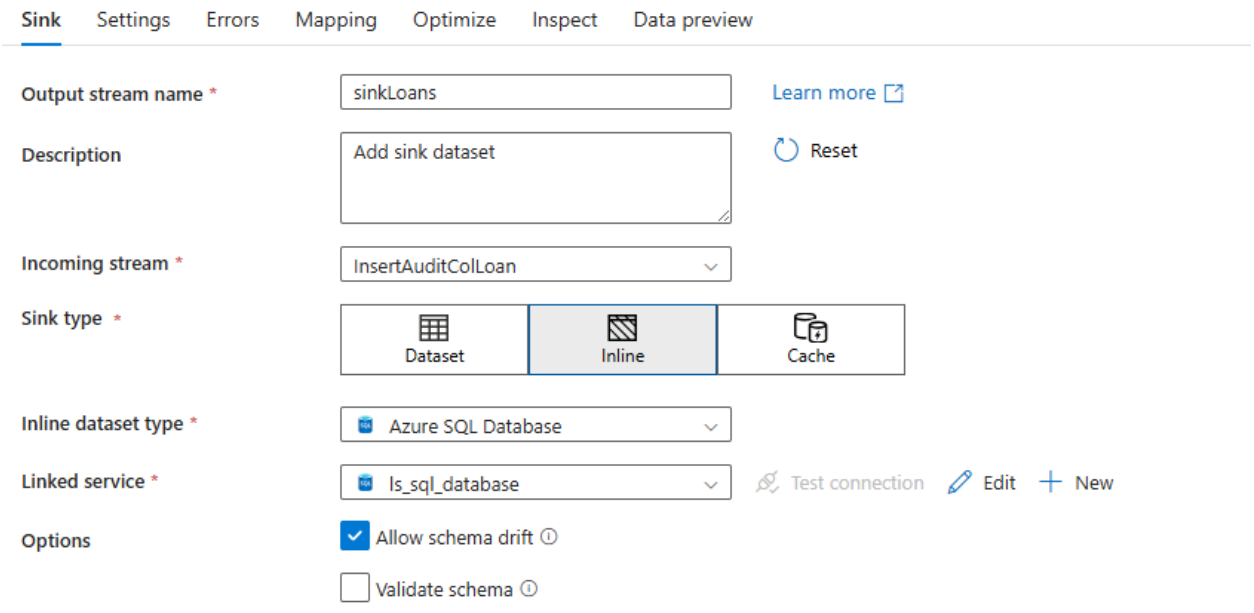
Stream Name	Condition	What it means
InsertLoan	isNull(LoanID)	If the record from the target doesn't exist (LoanID is NULL) → it's a new record ➔ So INSERT it
UpdateLoan	src_LoanID == LoanID && src_hashkey != HashKey	If record exists in target (LoanID matches) but there is some change in data (hash mismatch) ➔ Then Update (Expire old + Insert new)

- **Derived Column (Insert)**

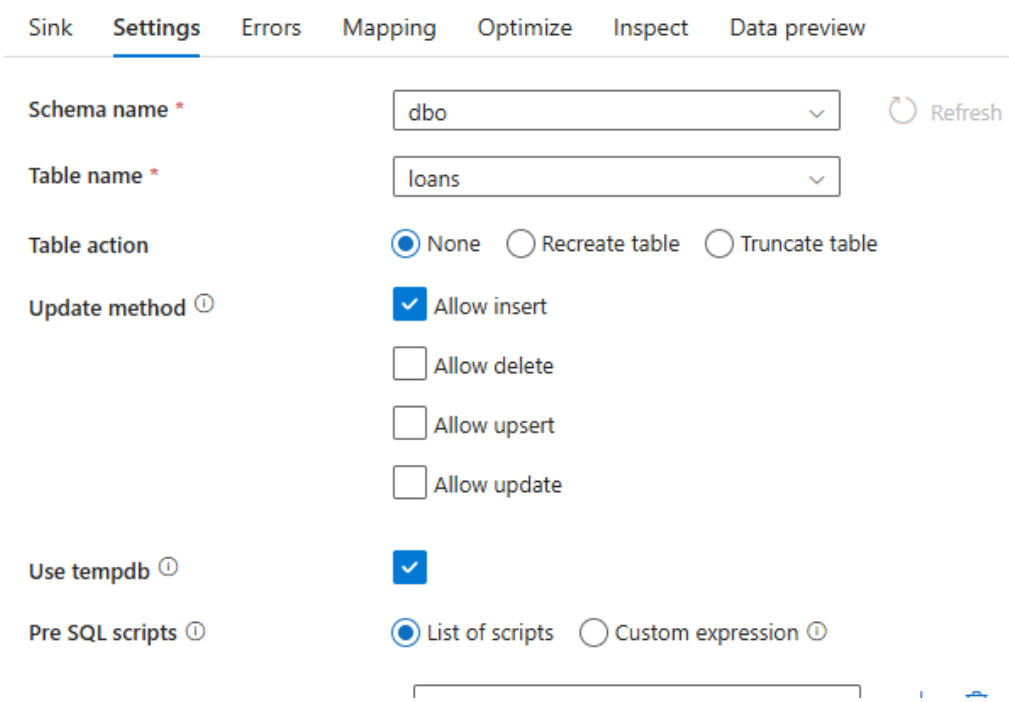


Column Name	Expression	Meaning
src_createddate	currentTimestamp()	Always puts current time when inserting/updating.
src_createdby	'Harpal'	Hardcoded as "Harpal" for every record.
src_updateddate	currentTimestamp()	Always puts current time again.
src_updatedby	'Harpal'	Hardcoded as "Harpal" for every record.

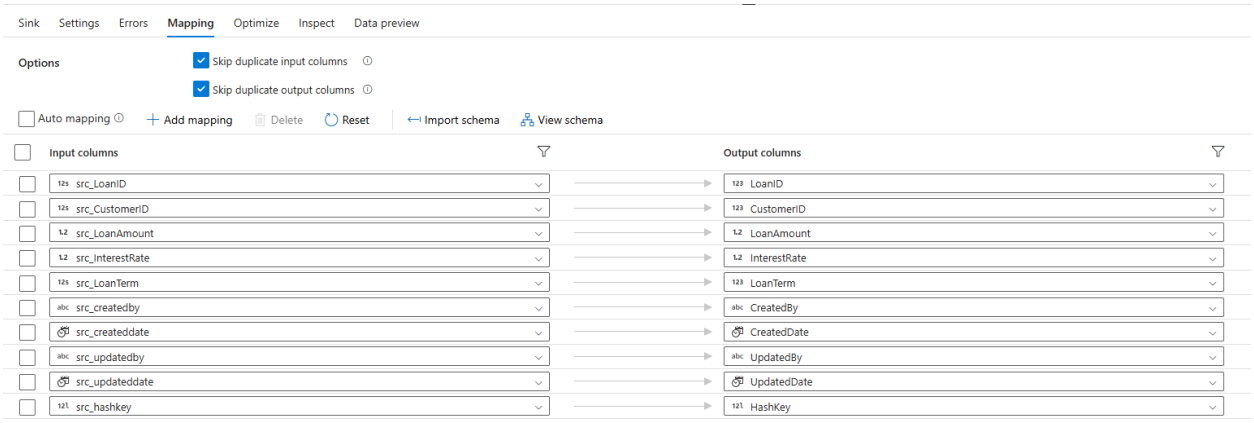
- **SinkLoans:**



Settings:

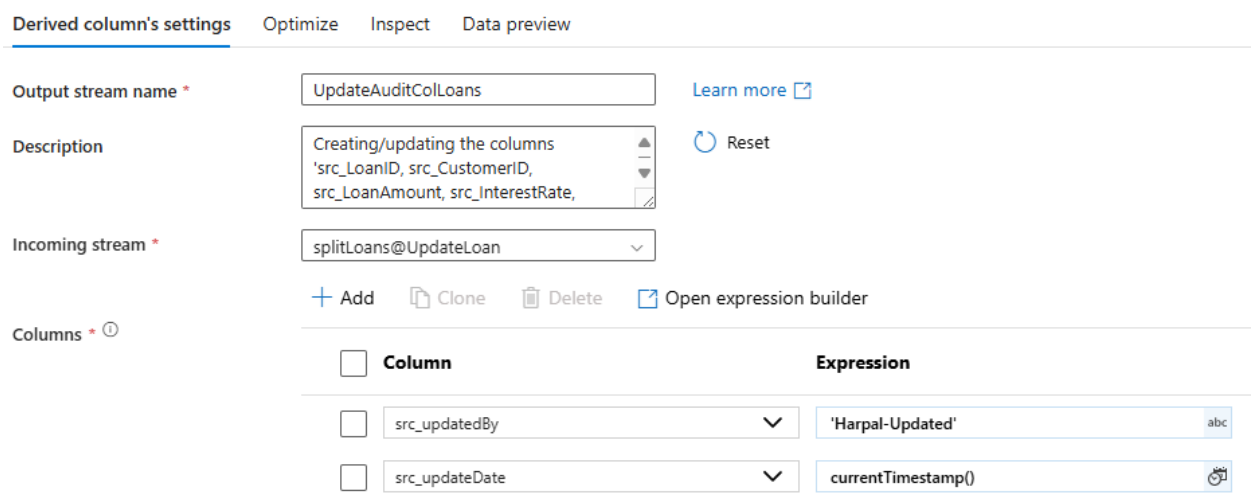


Mapping;



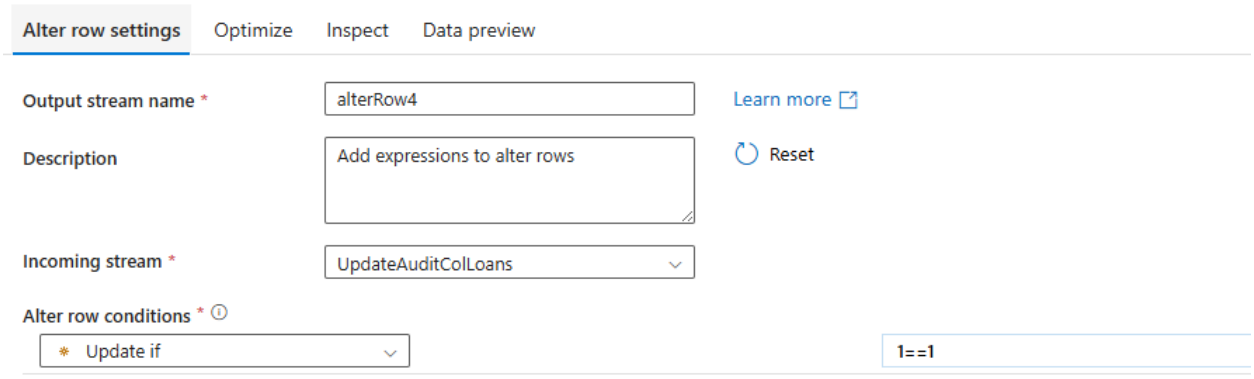
Update Part design:

- **Derived Column(Update):**

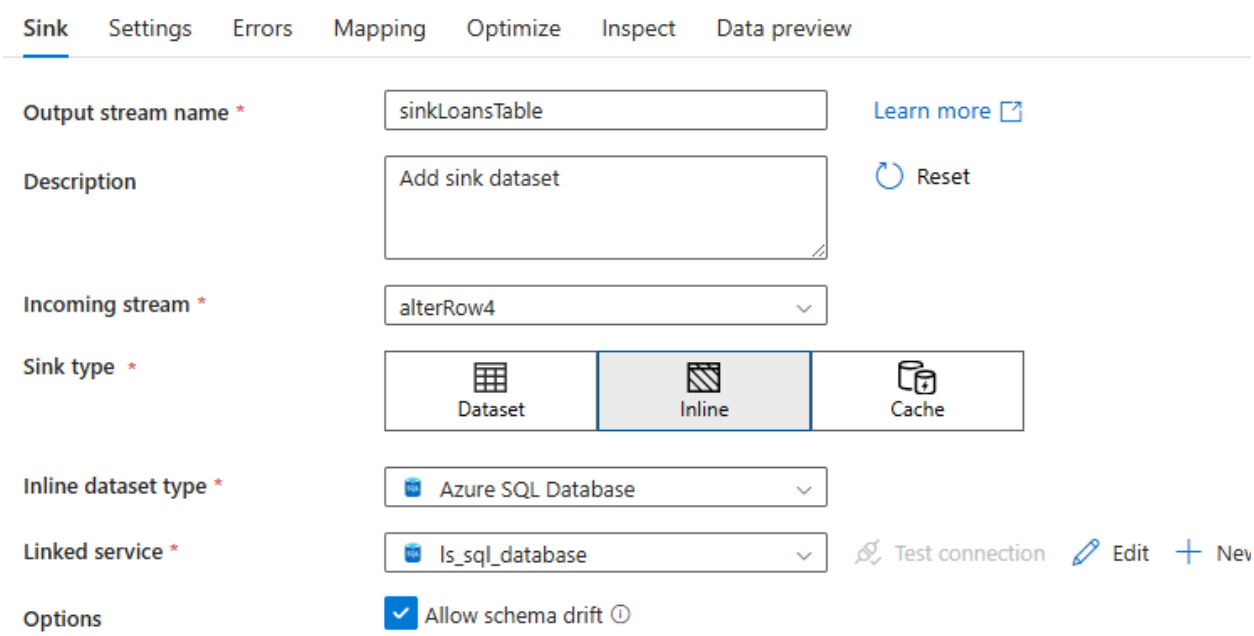


Column	Expression
src_updatedBy	'Harpal-Updated'
src_updateDate	currentTimestamp()

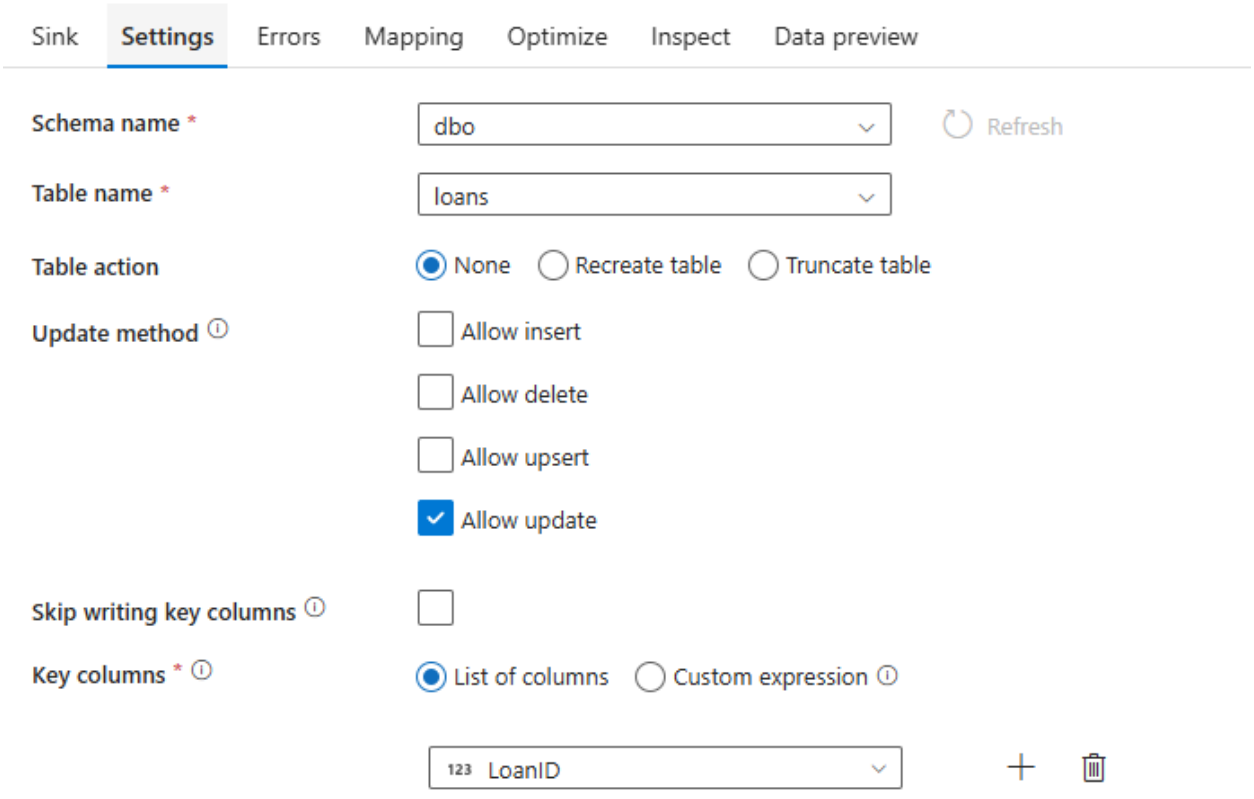
- **Alter Row Settings:**



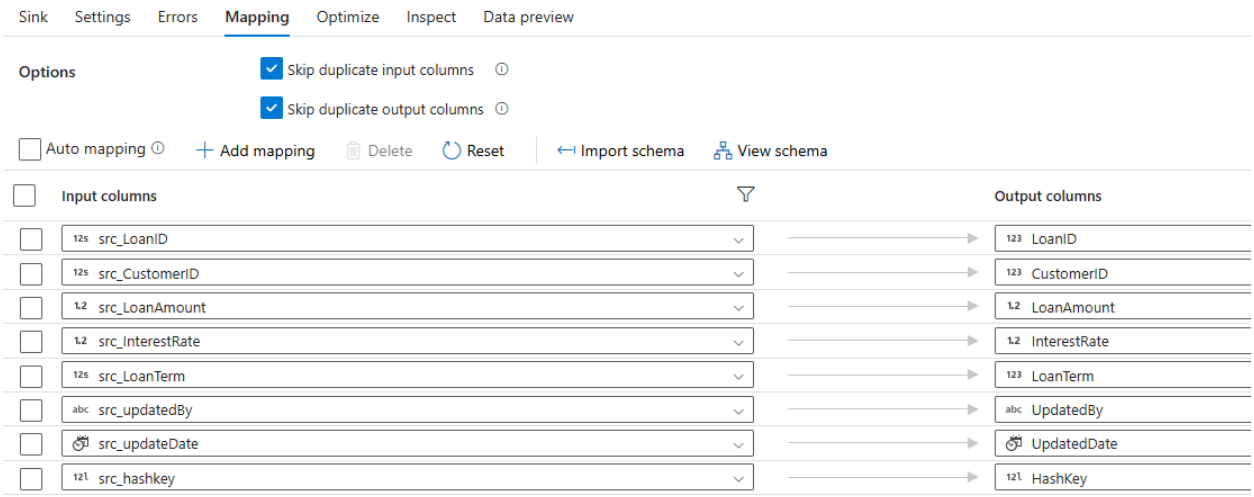
- **SinkLoansTable:**


Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Output stream name *
sinkLoansTable
Learn more
Description
Add sink dataset
Reset
Incoming stream *
alterRow4
Sink type *
Dataset
Inline
Cache
Inline dataset type *
Azure SQL Database
Linked service *
ls_sql_database
Test connection
Edit
Options
Allow schema drift

Settings:


Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
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

Mapping:

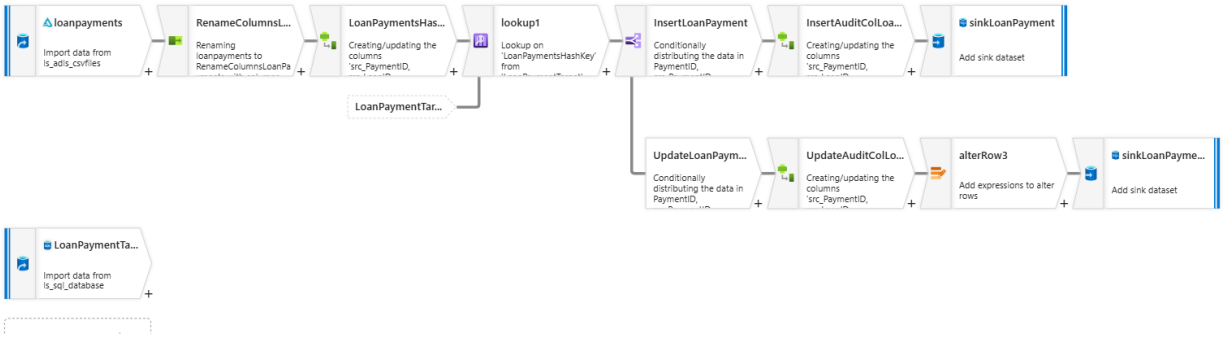


That's it for the Loans SCD 2 design in the gold layer.

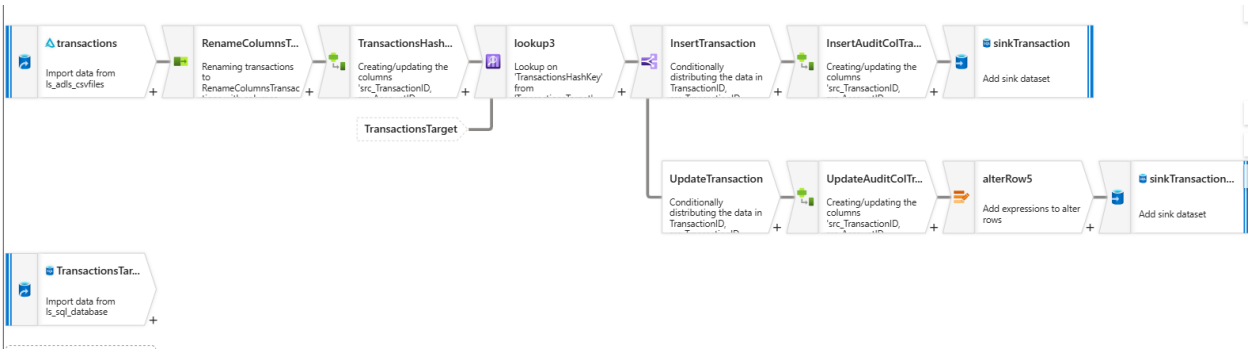
Similarly, we have to design for loan payments and transaction files.

Here I am attaching the flow for reference:

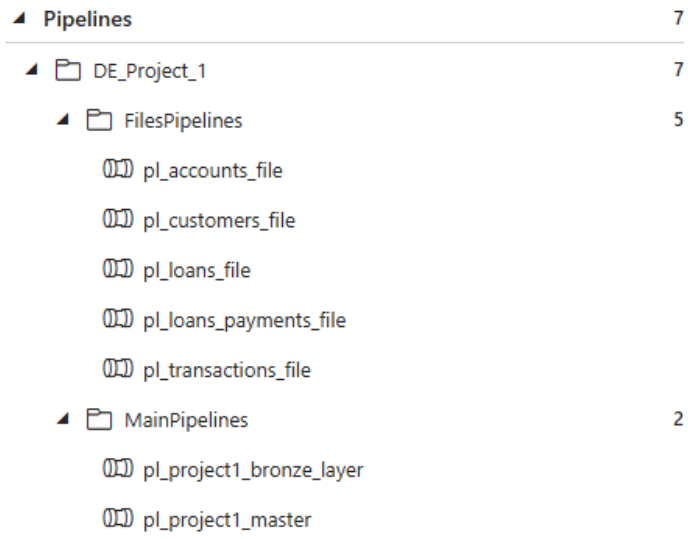
Loan Payments Gold Layer Data Flow:



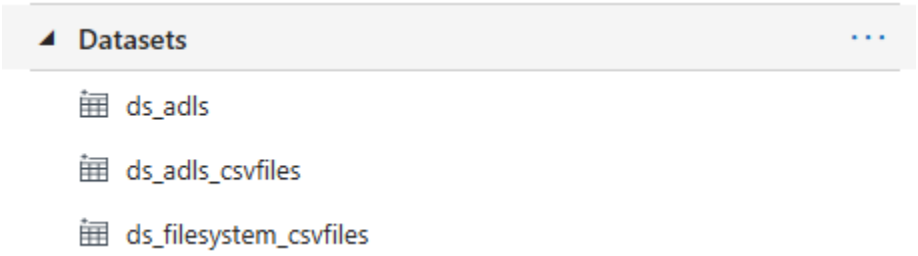
Transactions Gold Layer Data Flow:



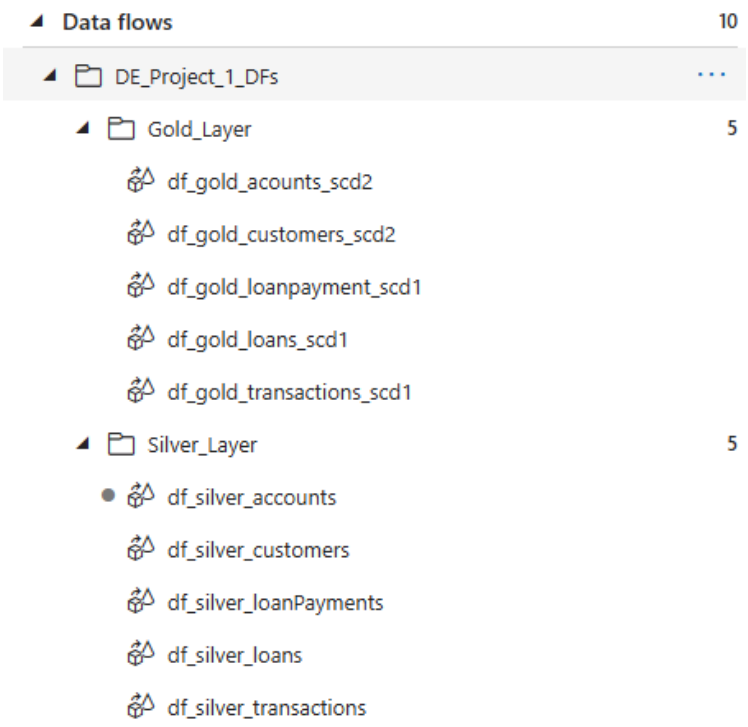
Folder Structure of pipelines:



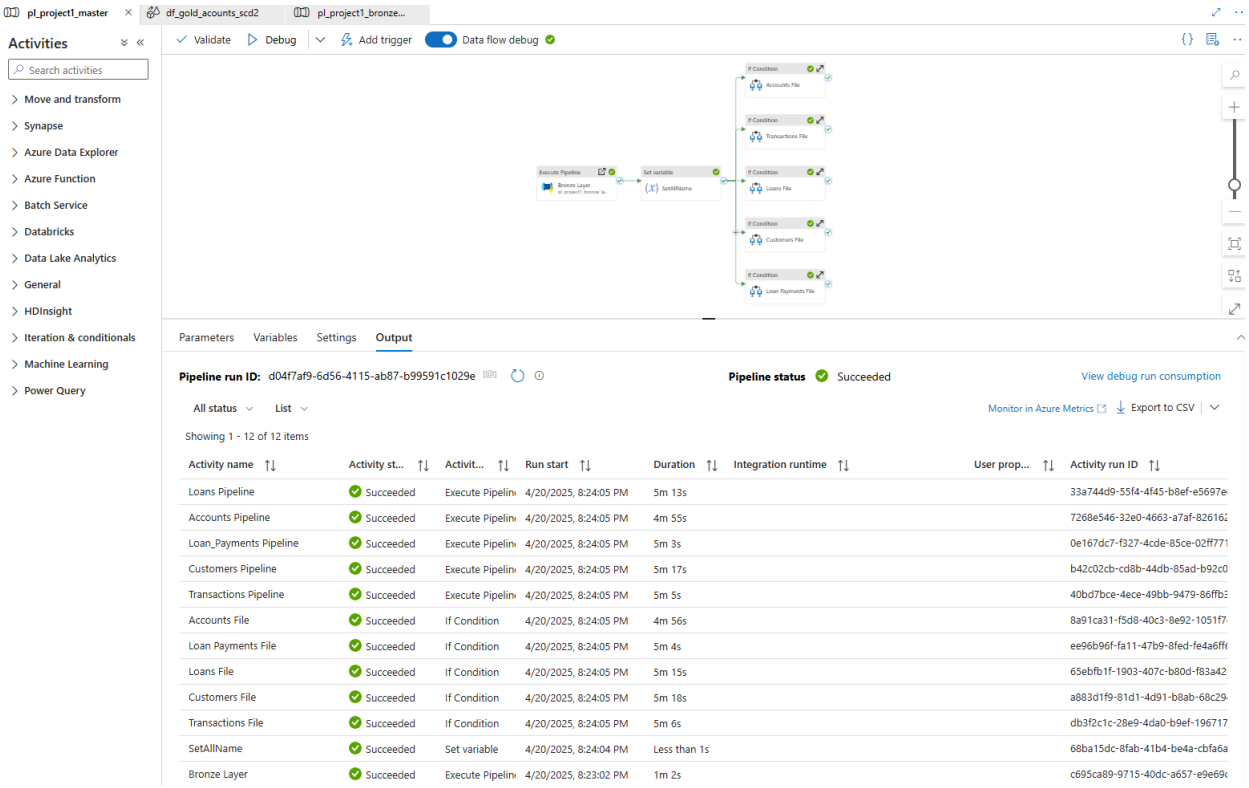
Folder Structure of the Dataset used:



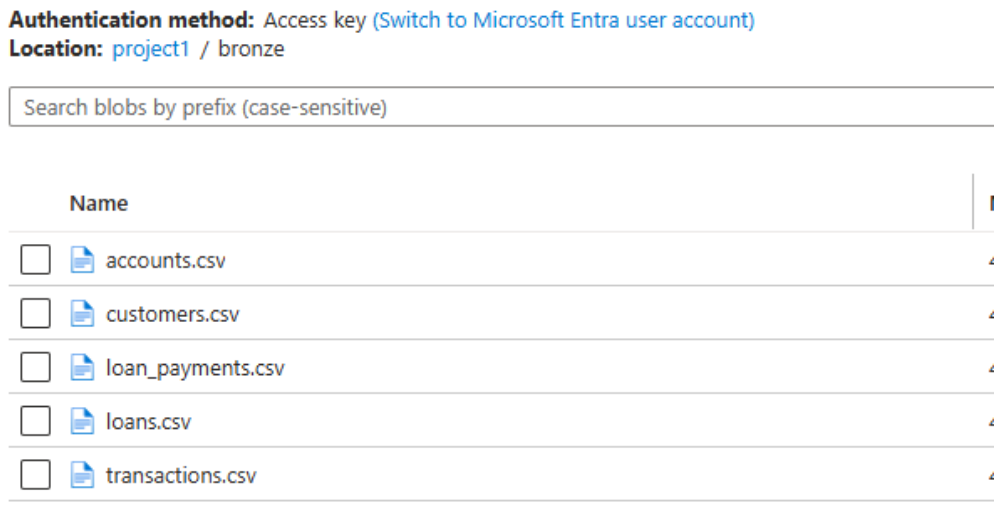
Folder Structure of all Dataflows used:



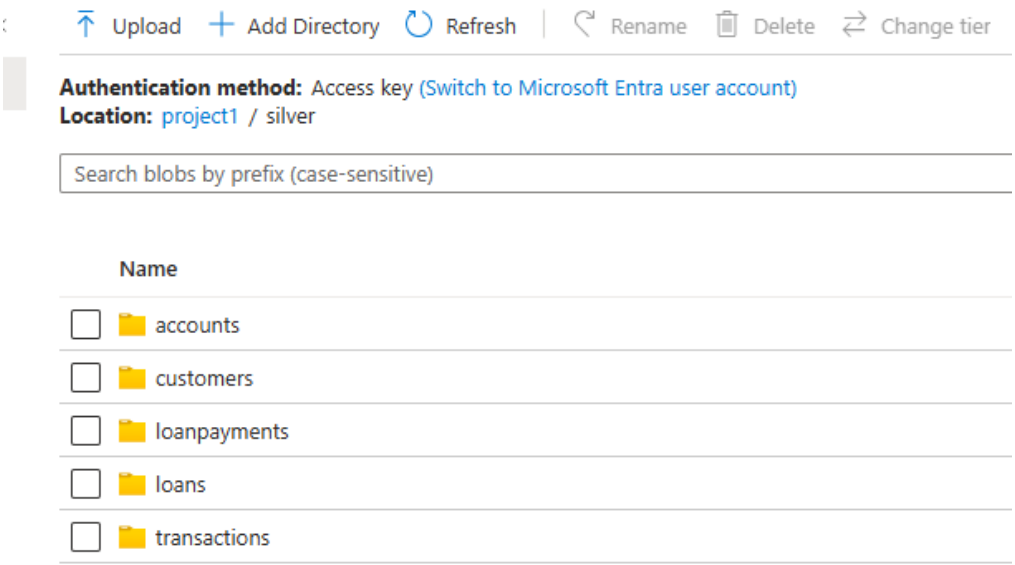
Let's run the pipeline:



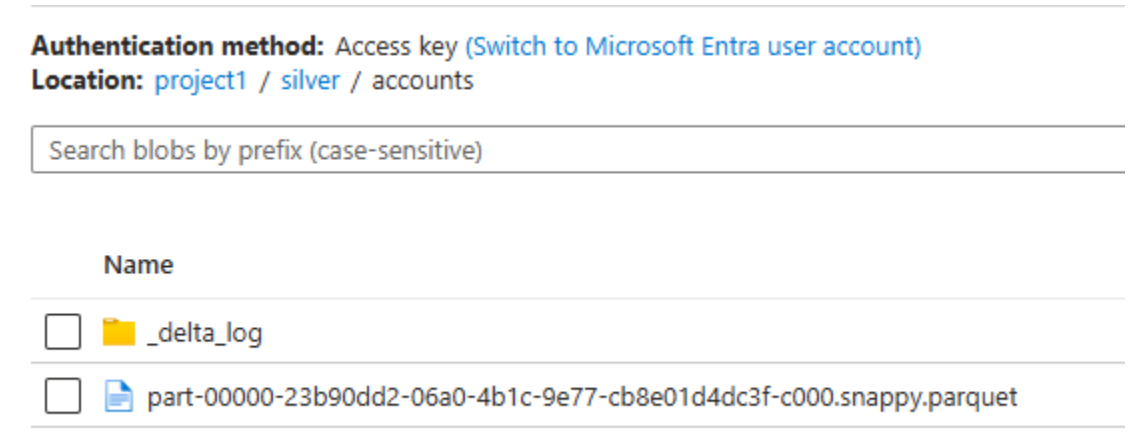
Check Bronze Layer Files in ADLS Gen 2 storage account:



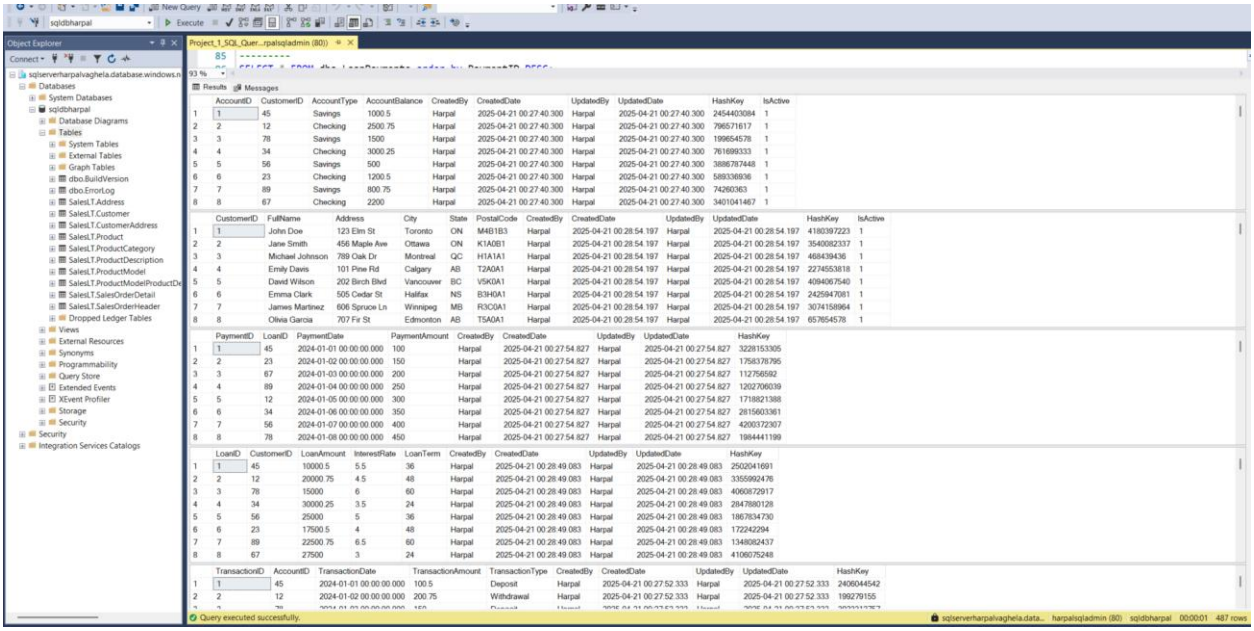
Silver Layer folder in ADLS Gen 2 Storage account:



Check account folder, its parquet file as we have selected delta format.



Check the output in SSMS (Azure SQL Database Tables):



Now, let's suppose the client updates a few records and inserts a new record in the files below on the next day.

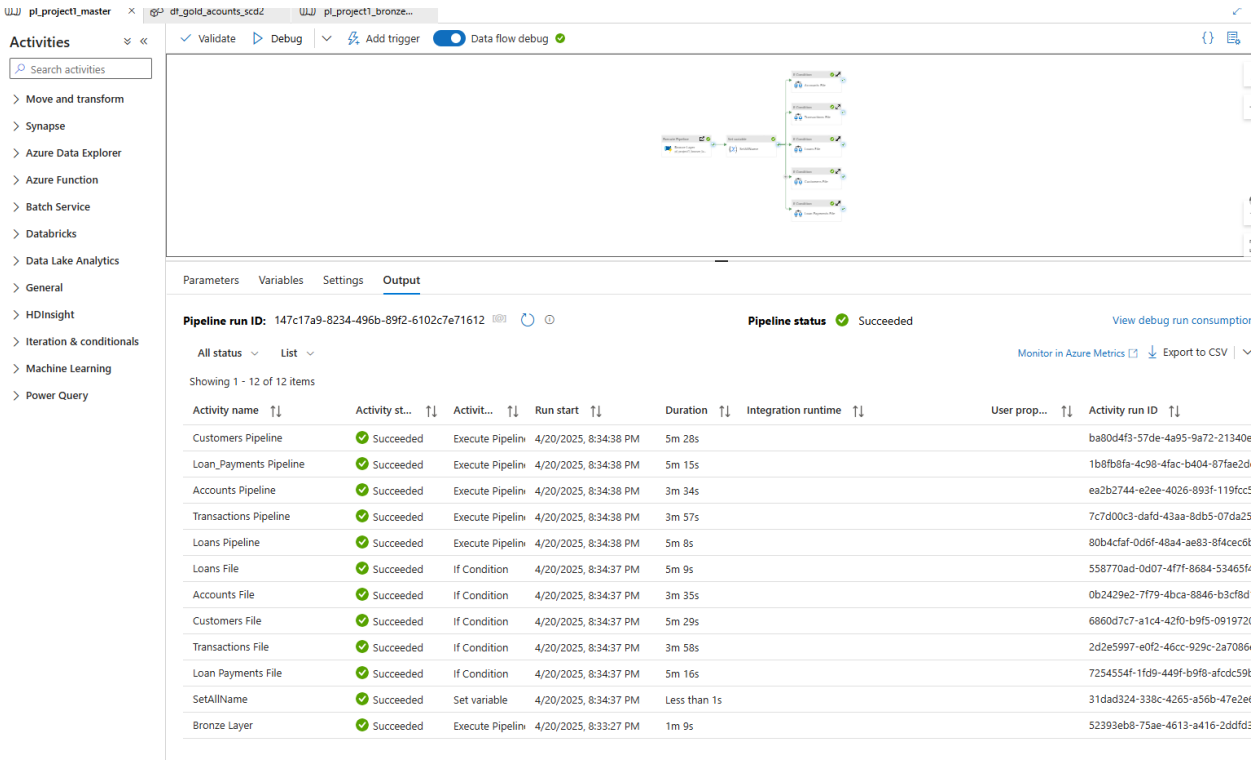
Customer Data Updated by client:

78	Abigail	Cole	7777 Fir St	Sundridge	ON	P0A0A1
79	James	West	7878 Redwood Dr	South River	ON	P0A0A1
80	Emily	Jordan	7979 Cypress Ave	North Bay	ON	P1B0A1
81	Michael	Owens	8080 Willow Rd	Mattawa	ON	P0H0A1
82	Elizabeth	Reynolds	8181 Poplar St	Sturgeon Falls	ON	P2B0A1
83	David	Fisher	8282 Ash Blvd	Verner	ON	P0H0A1
84	Sophia	Ellis	8383 Beech Dr	Field	ON	P0H0A1
85	John	Harrison	8484 Cedar Ln	Temagami	ON	P0H0A1
86	Olivia	Gibson	8585 Elm St	New Liskeard	ON	P0J0A1
87	William	McDonald	11 Maple Ave	NorthYork		
88	Harpal	Vaghela	Durham St	Oshawa	ON	L1J 5R3

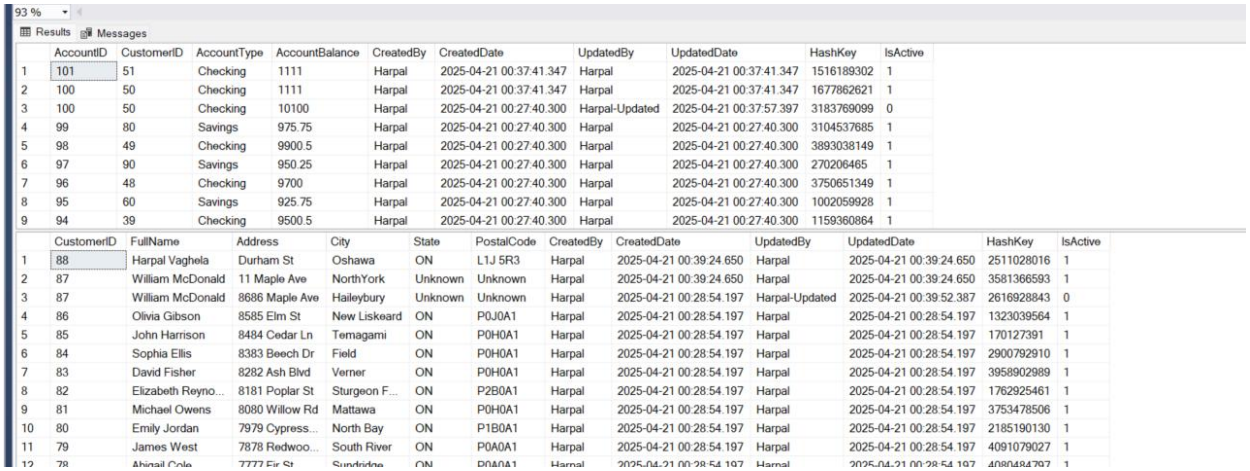
Account Data Updated by client

94	39	Checking	9500.5
95	60	Savings	925.75
96	48	Checking	9700
97	90	Savings	950.25
98	49	Checking	9900.5
99	80	Savings	975.75
100	50	Checking	1111
101	51	Checking	1111

Run the pipeline, or if we have created a trigger it will run automatically on next day at 8 AM (as we have set 8:00 AM trigger timing.)



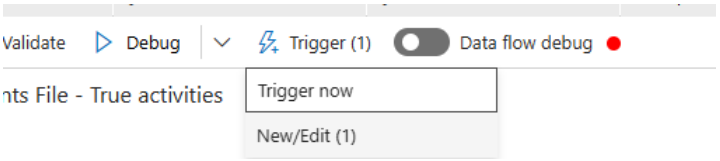
Output in SSMS:

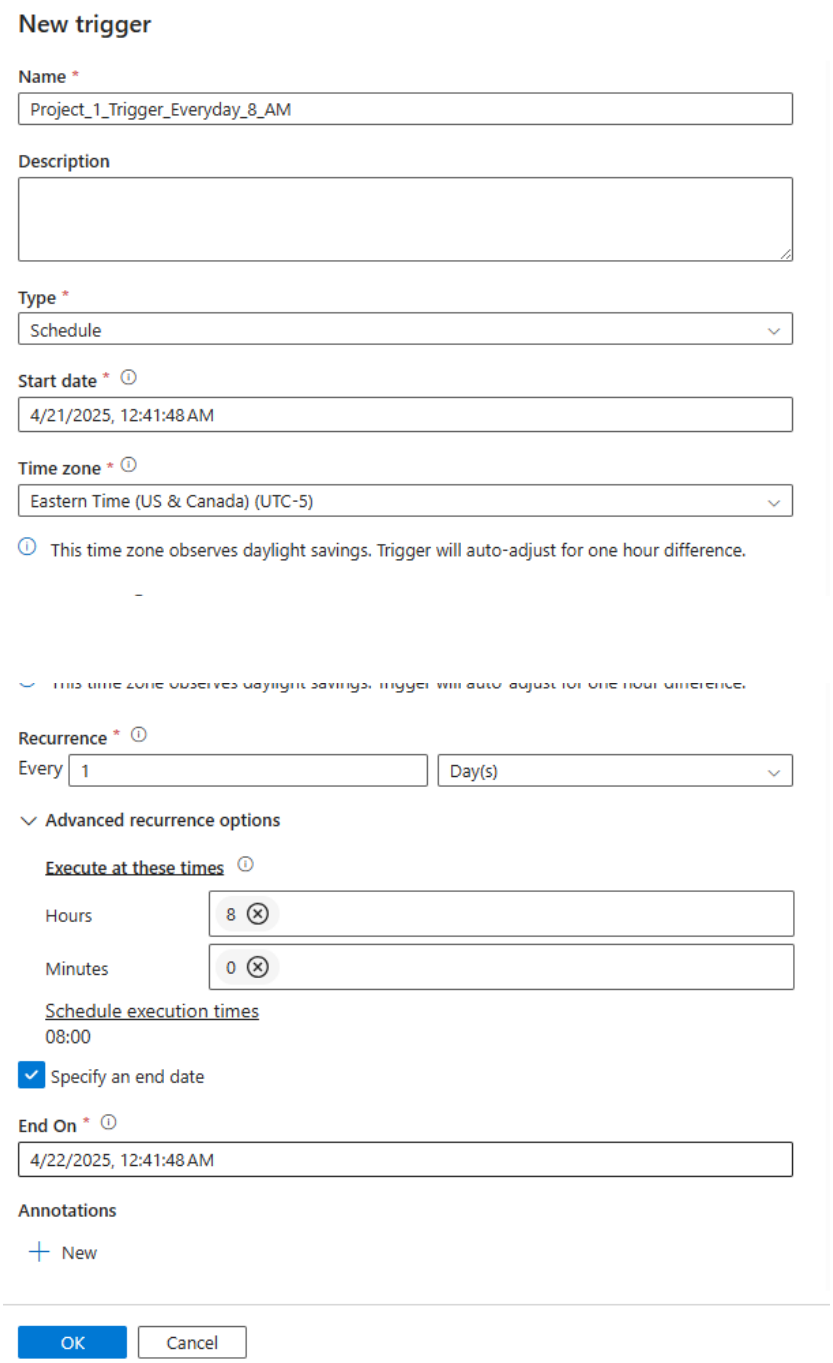


	AccountID	CustomerID	AccountType	AccountBalance	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	HashKey	IsActive
1	101	51	Checking	1111	Harpal	2025-04-21 00:37:41.347	Harpal	2025-04-21 00:37:41.347	1516189302	1
2	100	50	Checking	1111	Harpal	2025-04-21 00:37:41.347	Harpal	2025-04-21 00:37:41.347	1677862621	1
3	100	50	Checking	10100	Harpal	2025-04-21 00:27:40.300	Harpal-Updated	2025-04-21 00:37:57.397	3183769099	0
4	99	80	Savings	975.75	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	3104537685	1
5	98	49	Checking	9900.5	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	3893038149	1
6	97	90	Savings	950.25	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	270206465	1
7	96	48	Checking	9700	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	3750651349	1
8	95	60	Savings	925.75	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	1002059928	1
9	94	39	Checking	9500.5	Harpal	2025-04-21 00:27:40.300	Harpal	2025-04-21 00:27:40.300	1159360864	1

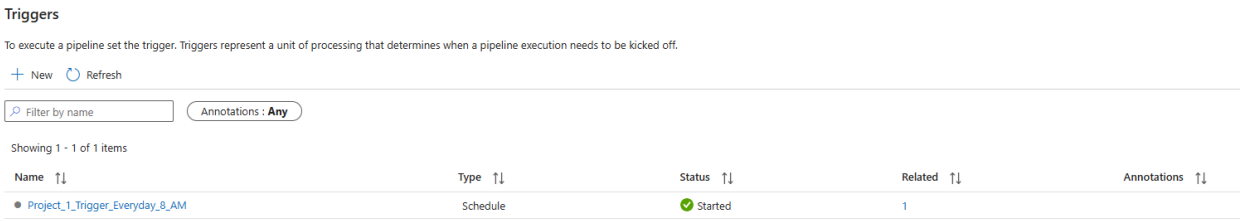
	CustomerID	FullName	Address	City	State	PostalCode	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	HashKey	IsActive
1	88	Harpal Vaghela	Durham St	Oshawa	ON	L1J 5R3	Harpal	2025-04-21 00:39:24.650	Harpal	2025-04-21 00:39:24.650	2511028016	1
2	87	William McDonald	11 Maple Ave	NorthYork	Unknown	Unknown	Harpal	2025-04-21 00:39:24.650	Harpal	2025-04-21 00:39:24.650	3581366593	1
3	87	William McDonald	8686 Maple Ave	Haileybury	Unknown	Unknown	Harpal	2025-04-21 00:28:54.197	Harpal-Updated	2025-04-21 00:39:52.387	2616928843	0
4	86	Olivia Gibson	8585 Elm St	New Liskeard	ON	P0J0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	1323039564	1
5	85	John Harrison	8484 Cedar Ln	Temagami	ON	P0H0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	170127391	1
6	84	Sophia Ellis	8383 Beech Dr	Field	ON	P0H0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	2900792910	1
7	83	David Fisher	8282 Ash Blvd	Verner	ON	P0H0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	3958902989	1
8	82	Elizabeth Reyno...	8181 Poplar St	Sturgeon F...	ON	P2B0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	1762925461	1
9	81	Michael Owens	8080 Willow Rd	Mattawa	ON	P0H0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	3753478506	1
10	80	Emily Jordan	7979 Cypress...	North Bay	ON	P1B0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	2185190130	1
11	79	James West	7878 Redwoo...	South River	ON	P0A0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	4091079027	1
12	78	Abigail Cole	7777 Fir St	Sundridge	ON	P0A0A1	Harpal	2025-04-21 00:28:54.197	Harpal	2025-04-21 00:28:54.197	4080484797	1

We can create a trigger on the main pipeline by clicking on this -> New/Edit





We can see the trigger has been created in the running in the manage section of Azure Data Factory:

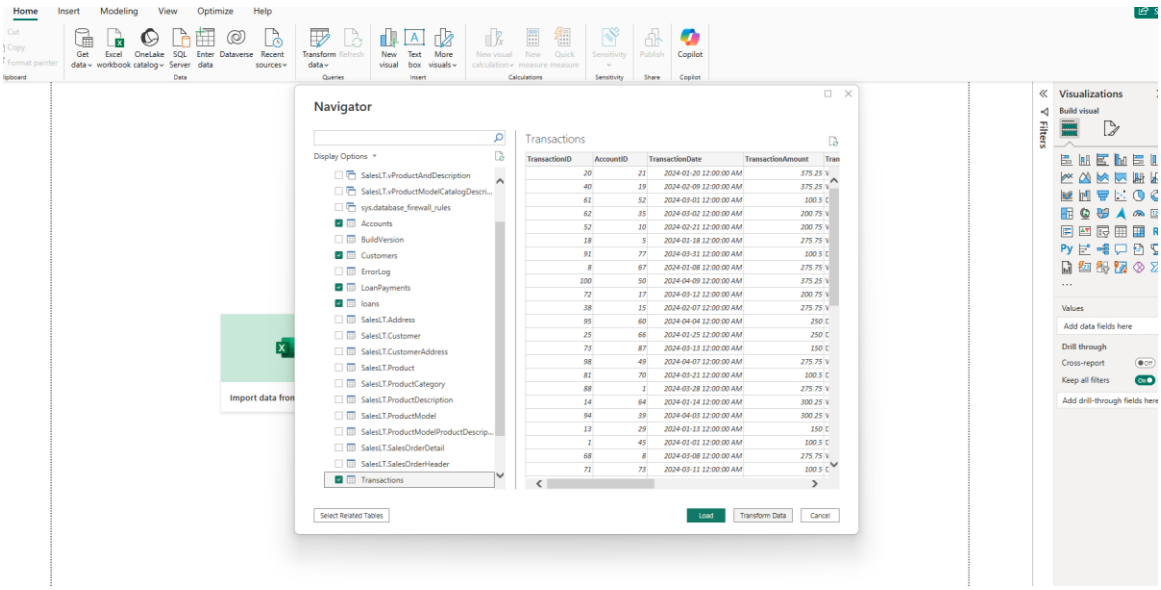


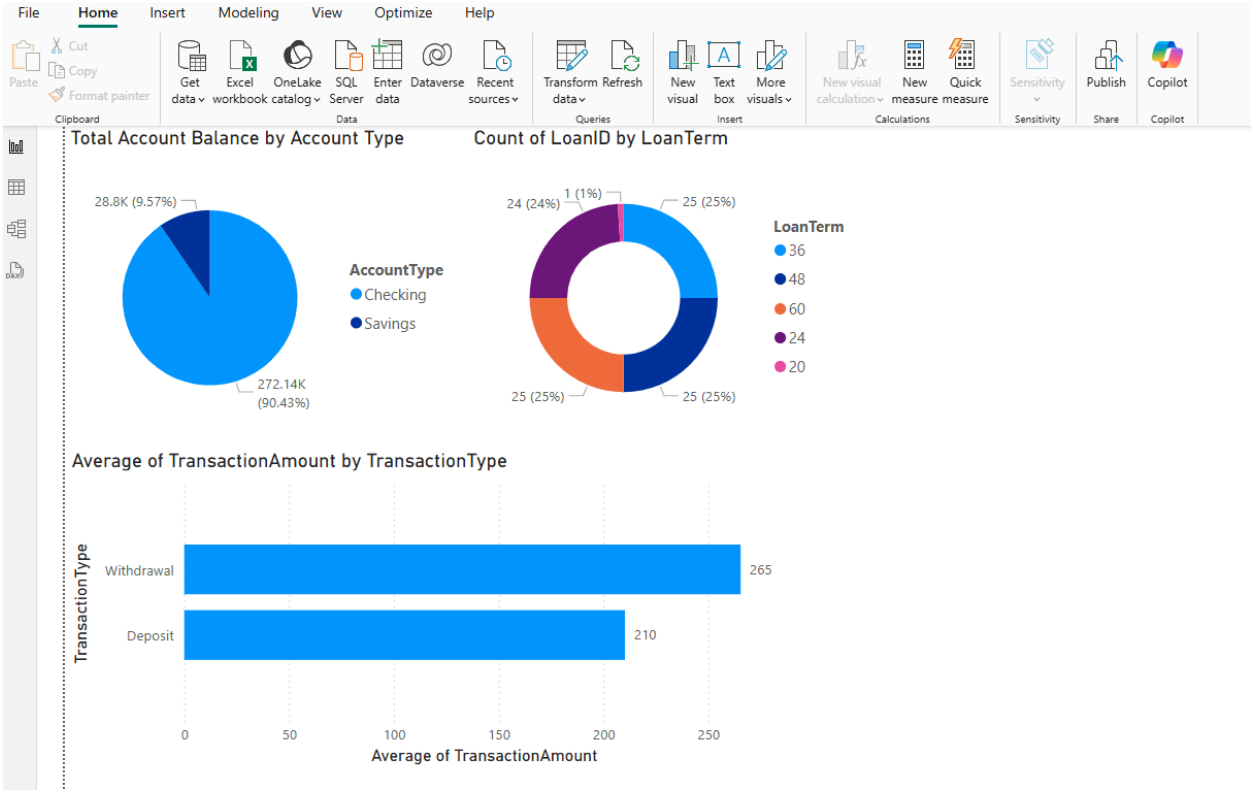
Step 4: Use Power BI for Data Visualization

Select Import from Azure SQL Database

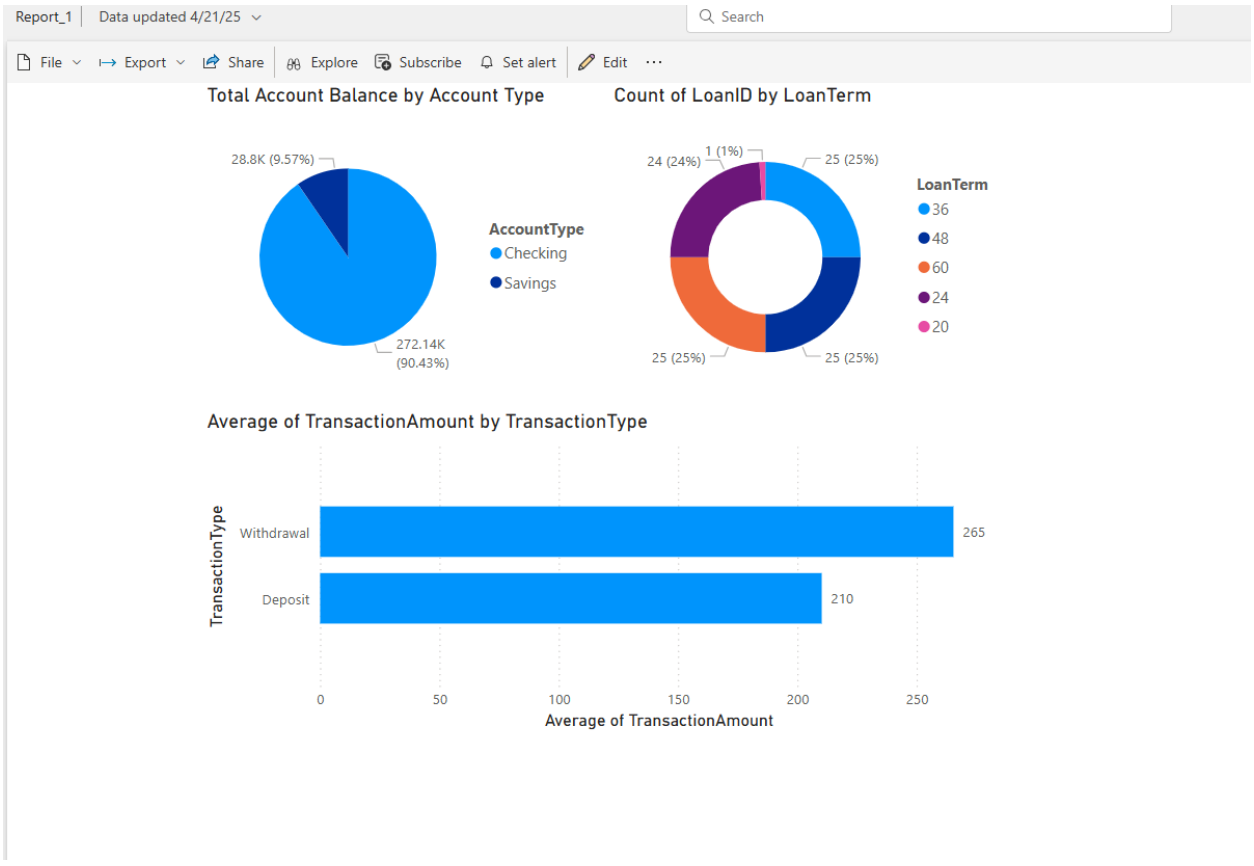
Give the Username and password

Select 5 tables that we need to make a visualization on



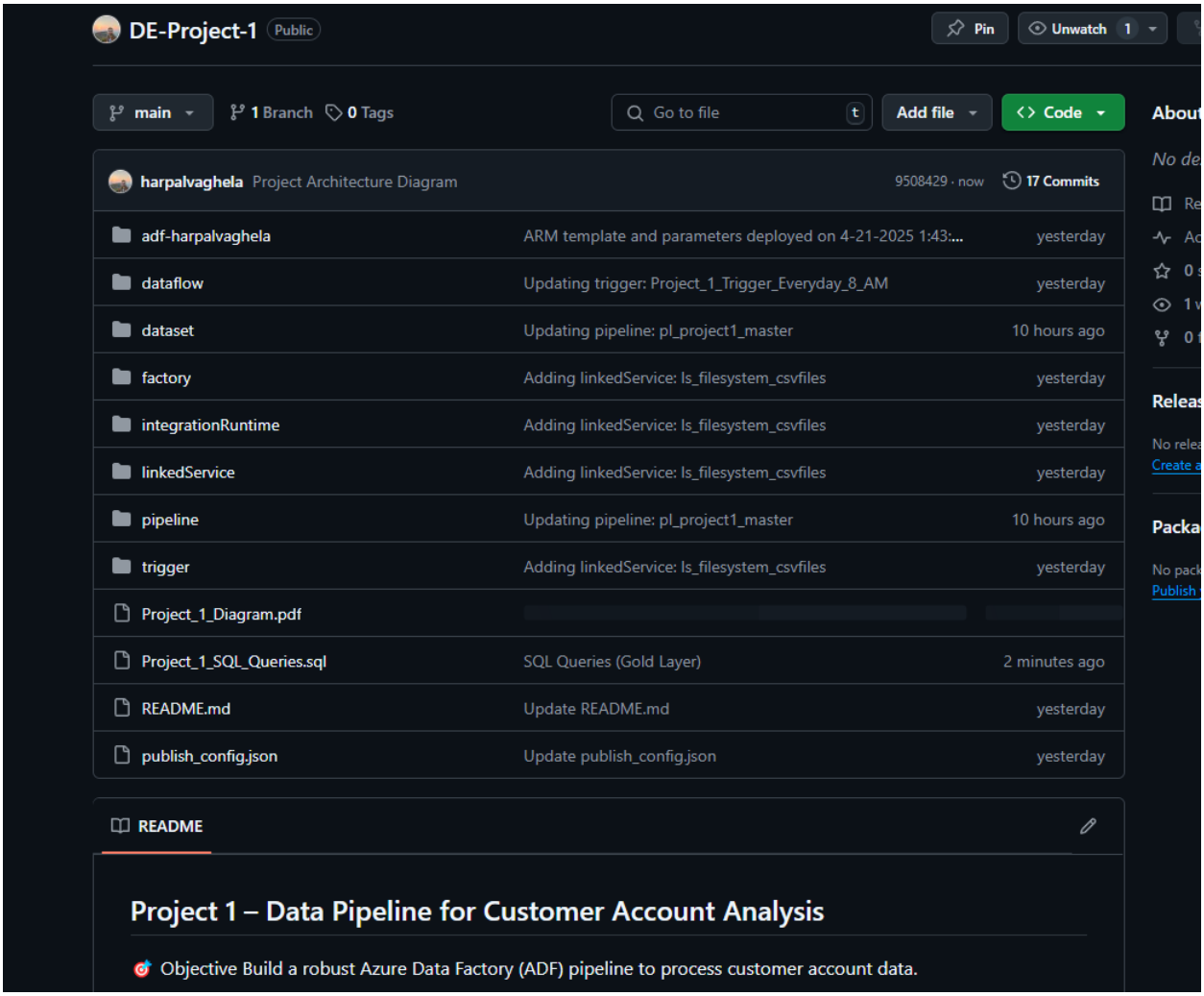


Publish it to the Fabric Workspace



Step 5: GitHub Integration

Project Link: harpalvaghela/DE-Project-1



Step 6: Conclusion

This project successfully demonstrates a real-world Data Engineering Pipeline on Azure Data Factory, handling ingestion, transformation (SCD 1 & SCD 2), and loading into Azure SQL DB, all automated and optimized. The smart use of dynamic variables, triggers, parallel pipelines, and delta formats showcases enterprise-grade best practices. With Power BI and GitHub integration, this project forms a complete end-to-end modern data platform solution ready for real-world production environments.

Points to remember

- Self-Hosted Integration Runtime: It allows secure copying of local on-premises files (CSV) into Azure without exposing public endpoints.

- Get Metadata to Fetch Last Modified: Used to check which files were recently modified, helping decide which files need to be processed.
- IF Condition for Data Freshness: Ensures that only new or updated files trigger a copy operation, saving compute and storage costs.
- Dual Copy Activities in Bronze Layer: One copy moves files into the Bronze layer, another takes a backup with timestamped filenames for audit purposes.
- Dynamic File Path with Parameters: Reduces hardcoding, making the pipeline flexible even if new files are added later.
- Append Variable inside ForEach: Smartly collects all modified filenames into a list for decision-making later in the master pipeline.
- Return Value from Child Pipeline: Passes back which files were touched so that the master pipeline can act only on relevant datasets.
- Master Pipeline Orchestration: Central controller that manages the full lifecycle from Bronze ➔ Silver ➔ Gold in the correct sequence.
- Dedicated Pipelines for Each File (Parallelism Ready): Designing separate pipelines for each dataset allows running them independently and faster.
- 5 IF Conditions for File-Specific Processing: Instead of running everything blindly, each IF checks if its related file was modified, improving efficiency.
- Delta Format in Silver Layer: Provides faster reads and updates compared to CSV and supports ACID transactions for analytics workloads.
- Filter, Remove Duplicate, and Derived Columns: Enforces basic data quality and enrichment before moving data further downstream.
- Hash Key Creation for SCD Operations: Hashes multiple fields to create a unique fingerprint of the row, enabling accurate change detection.
- SCD Type 1 Handling (Overwrite Updates): Simply updates existing records without tracking history when a change is detected.
- SCD Type 2 Handling (Maintain History): Marks old records as inactive and inserts new records, preserving the historical state of the data.
- Audit Columns Automation (CreatedBy, UpdatedBy): Dynamically tracks who inserted or updated data along with timestamps, to ensure good governance.
- Alter Row Transformation for Row-Level Control: Based on conditions, controls whether a row should be inserted, updated, or ignored during sink operations.
- Use TempDB for Small Interim Operations: For smaller data loads, using SQL Server Tempdb speeds up processing without physically creating tables.
- GitHub Integration for Version Control: All pipeline artifacts (datasets, dataflows, pipelines) are version-controlled and backed up automatically.
- Daily Trigger for Full Automation: A scheduled trigger at 8:00 AM ensures the pipeline executes automatically without human intervention.