

# Project 4

## Incremental Data Loading and Automated Notifications using Microsoft Fabric

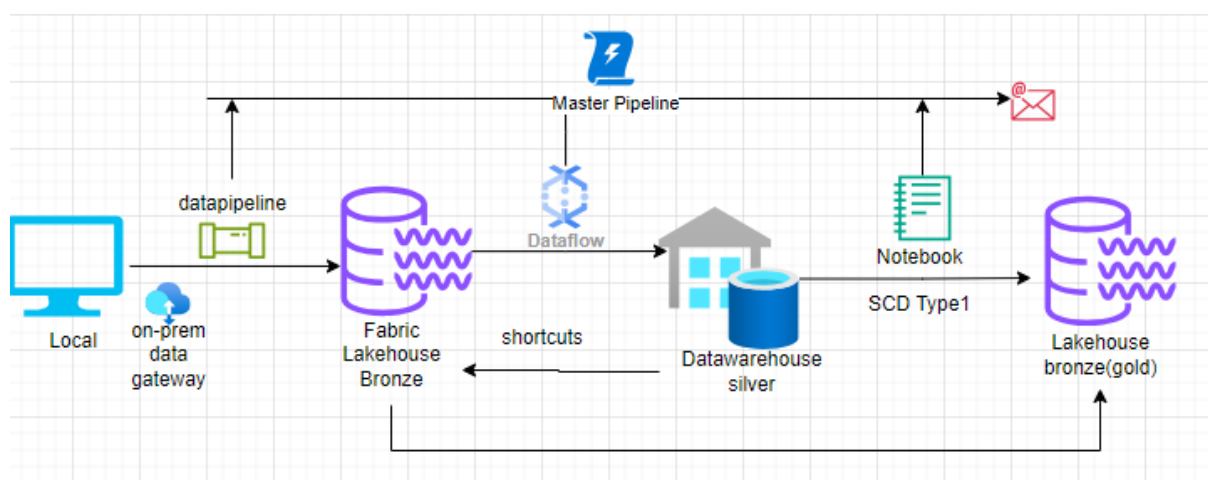
### Overview:

In modern data ecosystems, organizations need to efficiently ingest, transform, and load data from various sources into centralized platforms for analytics, while also ensuring timely monitoring and notification upon successful data refreshes. This project addresses the challenge of incrementally loading data from on-premises sources to Microsoft Fabric Lakehouse, processing it through a structured transformation pipeline, and triggering automated notifications upon successful execution.

### Tools & Technologies:

- Microsoft Fabric
- On-premises Data Gateway
- Fabric Lakehouse & Warehouse
- Fabric Data Flows
- Fabric Notebook

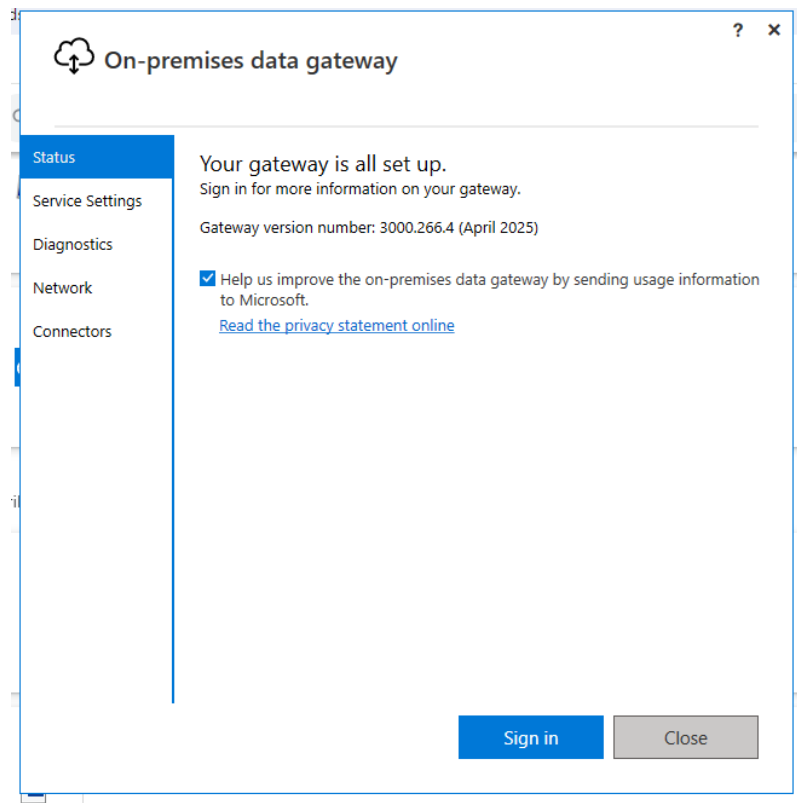
### Architecture:



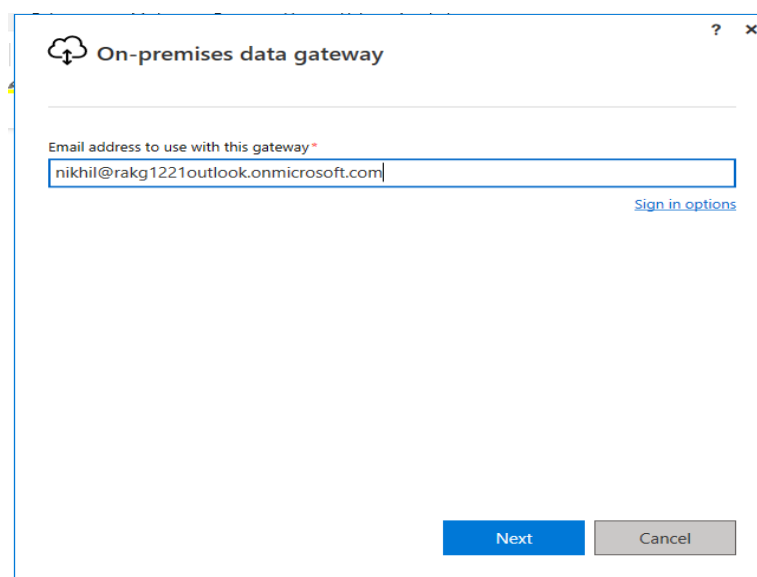
## Installing On-Premises Data Gateway on Local:

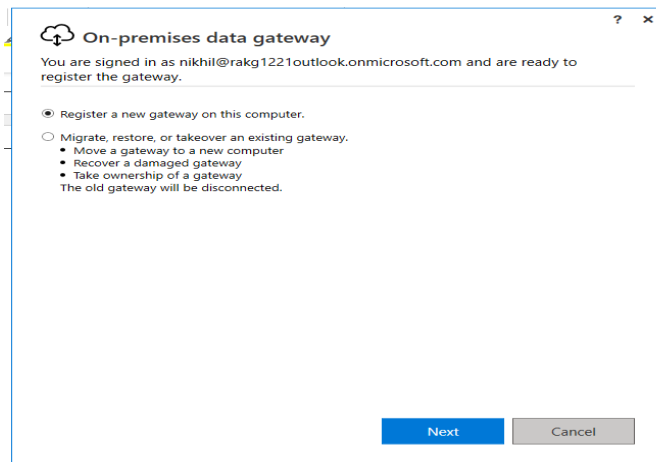
To connect on-premises data sources to fabric for data ingestion, install on-premises data gateway in source system.

1. Search on-premises Data Gateway in browser.

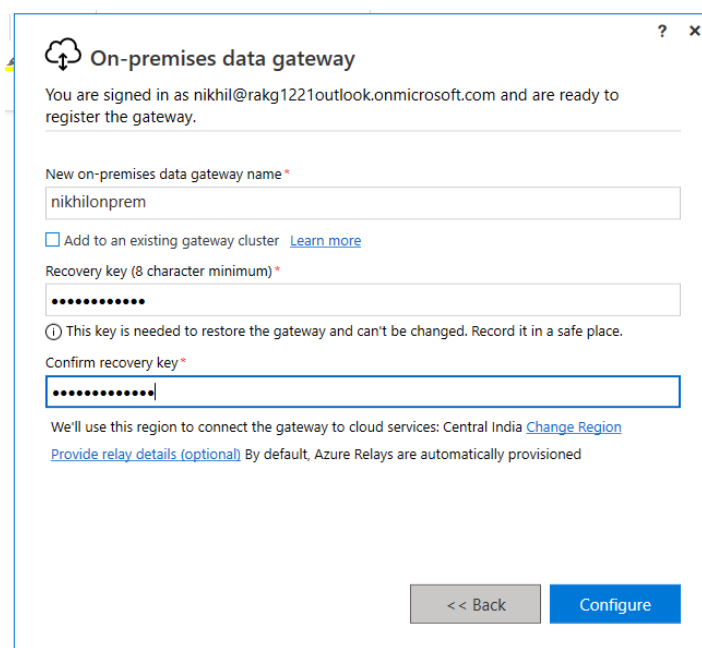


2. Accept the terms and Install
3. Sign in: sign in with the fabric registered account.

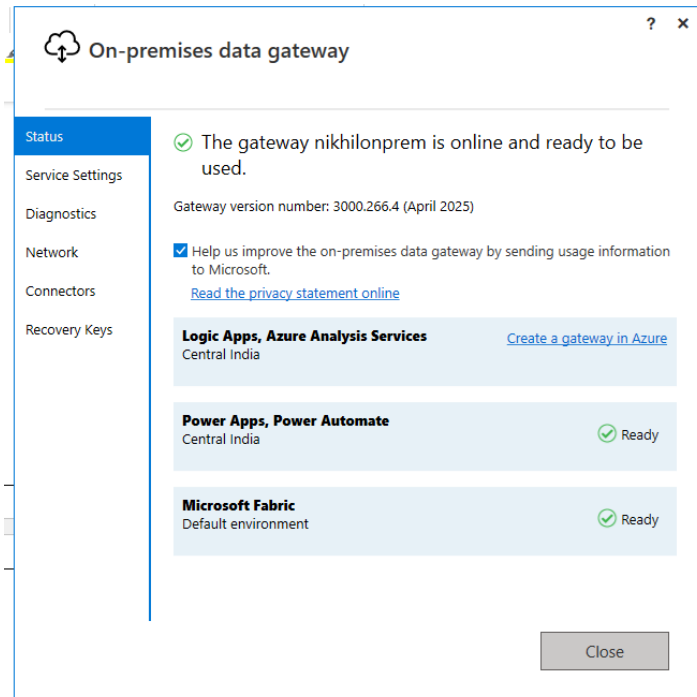




4. From the select register a new gateway on this computer.



5. Provide the gateway name and password.



6. On-premises data gateway is ready.

**To Ensure, if the gateway is connected to fabric**

Open Microsoft Fabric→settings→Manage connections & Gateways→on-premises data gateway

#### Manage Connections and Gateways

Connections On-premises data gateways Virtual network data gateways Azure Key Vault references

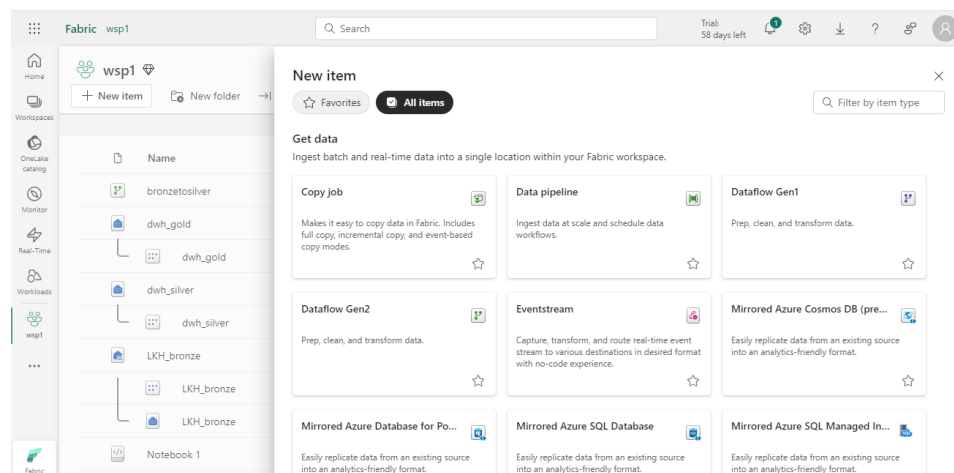
The data gateway acts as a bridge, providing quick and secure data transfer between on-premises data and Power BI, Microsoft Flow, Logic Apps, and PowerApps. [Learn more in this overview.](#)

Name ↑	Contact info	Users	Status	Gateways
nikhilonprem	nikhil@rkg1221outlook.onmicrosoft.com	nikhilfabric	😊 Online	1

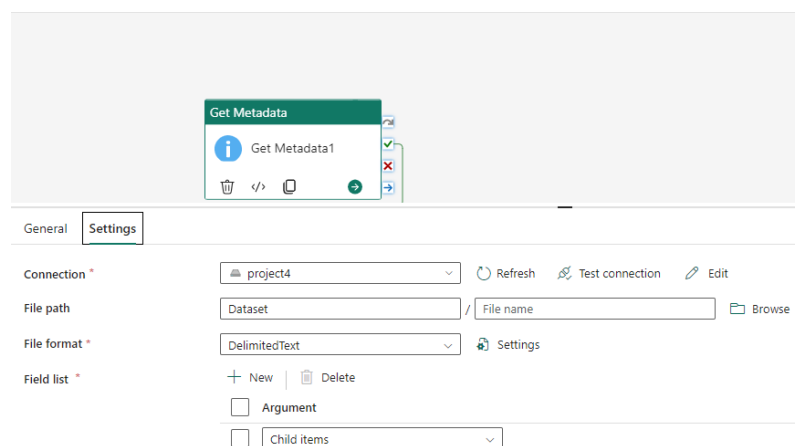
## On-premises Data ingestion to Fabric Lakehouse using on-prem data gateway:

Since the data source has on-premises data gateway installed and connected to fabric, we can directly ingest the data to fabric lakehouse by providing the file path.

1. Create a workspace.(workspaces→+New Workspace→name,description and domain→apply).
2. Create a Lakehouse inside the workspace.(+New Item→search lakehouse→name the lakehouse).
3. Create a Data pipeline to ingest the on-prem data.
4. +New item→search Data pipeline



5. Name the pipeline (on-premtobronze).
6. Add Get Metadata activity to the canvas.



## 7. Connection → more → new source as **folder**

Get data

Connect data source

Folder  
File  
[Learn more](#)

Connection credentials

Connection  
Create new connection

Connection name  
C:\Users\Admin\Documents

Data gateway \*  
[On-premises] nikhilonprem

Authentication kind  
Windows

Username  
Admin

Password  
.....

☐ This connection can be used with on-premise data gateways and VNet data gateways.

Back Connect

8. Provide the file path, provide connection name and select Data gateway from the drop-down and atlast provide the windows credentials.

9. Connect.

General Settings

Connection \*  
project4 Refresh Test connection Edit

File path  
Dataset / File name Browse

File format \*  
DelimitedText Settings

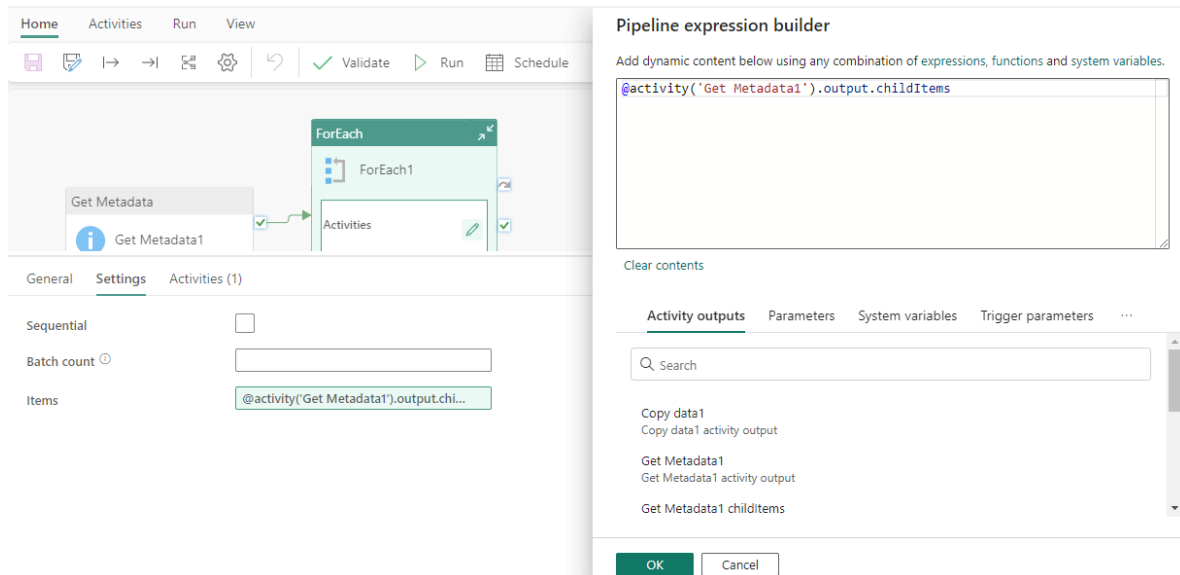
Field list \*  
+ New | Delete  
☐ Argument  
☐ Child items

> Advanced

10. Browse the folder, Select Field list as child items.

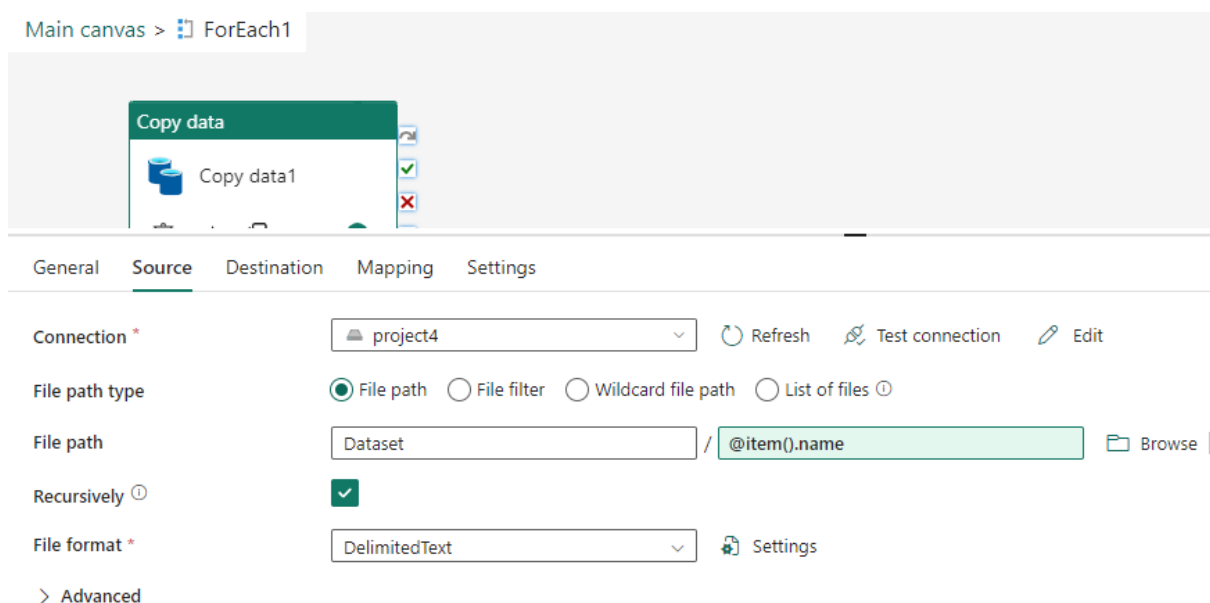
**Add ForEach** to the canvas and connect with get metadata activity on success.

→ForEach activity→settings→items→add dynamic content→under activity outputs→get metadata childitems



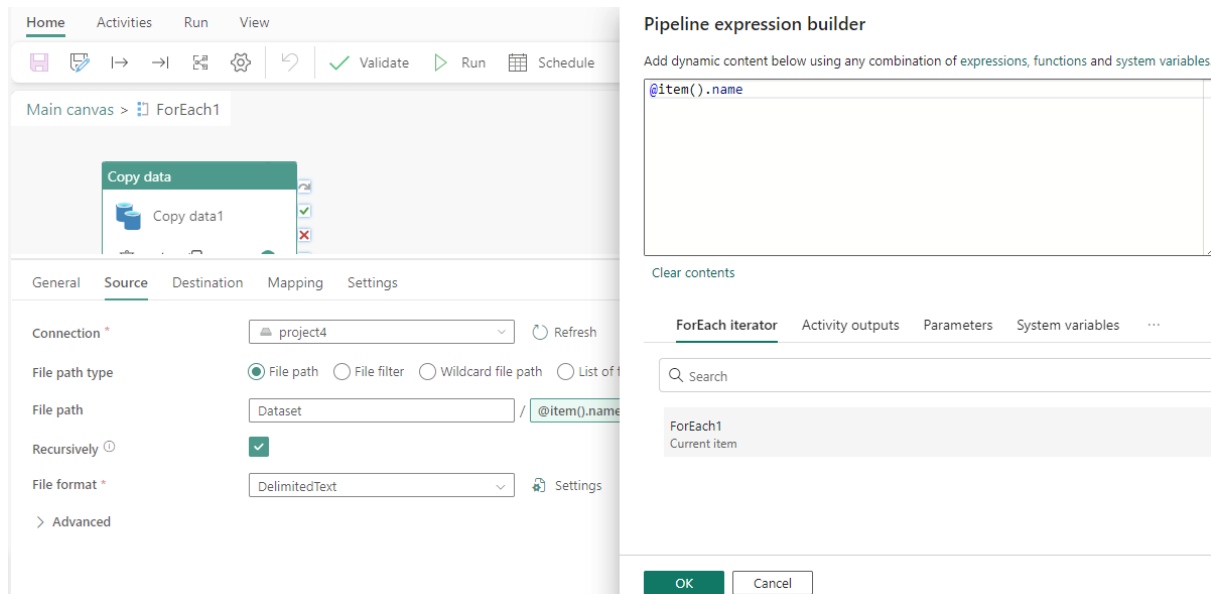
## Add Copy activity Inside the ForEach

- Source:

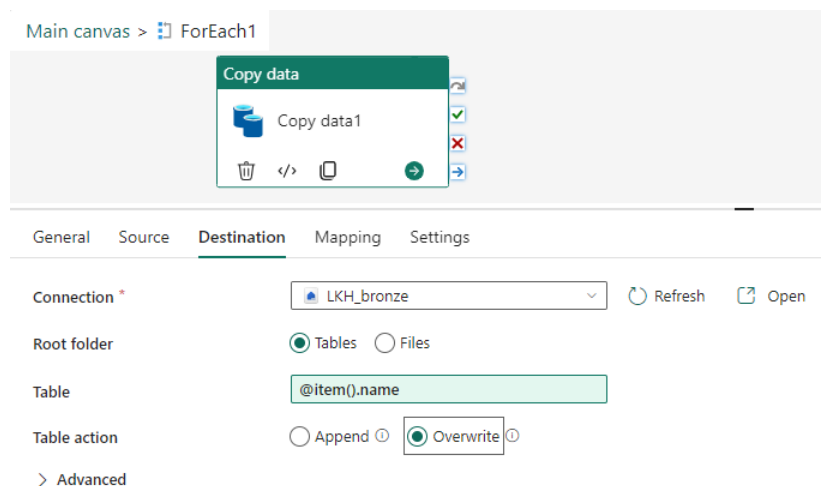


1. Connection: choose the connection that has been created for metadata activity.

2. Browse the file path, since copying multiple files at once created a parameter for file name



3. Filename: add dynamic content and select ForEach1
  4. File format as DelimitedText(CSV).
- Destination:



1. Connection: select the destination(sink) to copied
2. Select the root folder, since I want to store the data in tables
3. Table name: Add dynamic content→ForEach
4. Table action: overwrite.

Save and run



## Output:

Pipeline run ID: ce759d2f-bdab-4042-9e16-b40bebdd9b8d Pipeline status: Succeeded

Filter by keyword Showing 7 items

Activity name	Activity status	Run start	Duration	Input	Output
Get Metadata1	Succeeded	5/5/2025, 2:00:07 PM	13s		
ForEach1	Succeeded	5/5/2025, 2:00:20 PM	54s		
Copy data1	Succeeded	5/5/2025, 2:00:21 PM	29s		
Copy data1	Succeeded	5/5/2025, 2:00:21 PM	44s		
Copy data1	Succeeded	5/5/2025, 2:00:21 PM	40s		
Copy data1	Succeeded	5/5/2025, 2:00:21 PM	48s		
Copy data1	Succeeded	5/5/2025, 2:00:21 PM	51s		

Data is written to LKH\_Bronze.

Explorer

Search tables

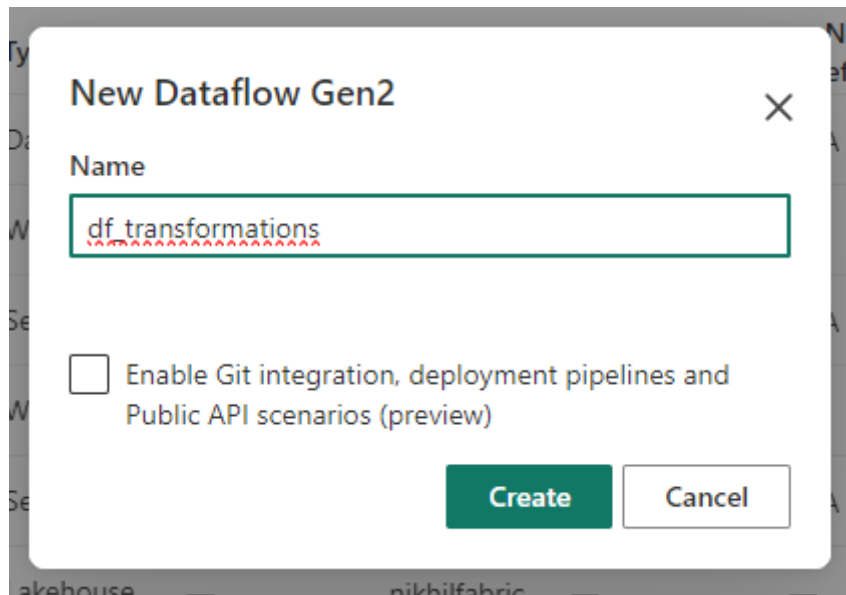
LKH\_bronze

- Tables
  - accounts\_scd1
  - accounts.csv
  - customers\_scd1
  - customers.csv
  - loan\_payments.csv
  - loanpayments\_scd1
  - loans\_scd1
  - loans.csv
  - transactions\_scd1
  - Unidentified
- Files

## Bronze to Silver:

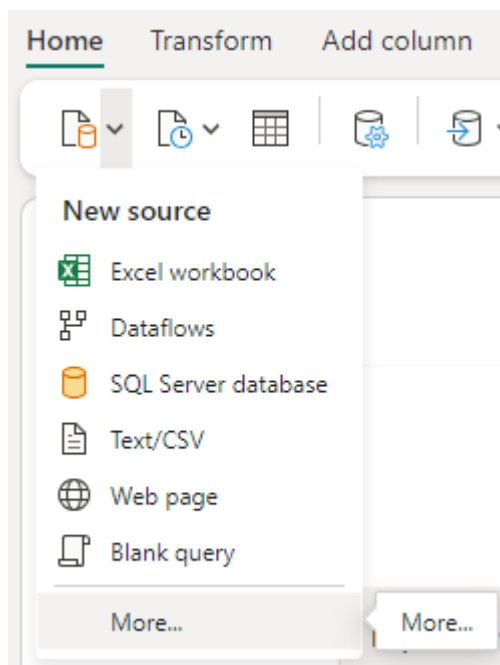
Data Transformations(cleaning data, remove nulls, deduplicates).

→Create a Dataflow in workspace.

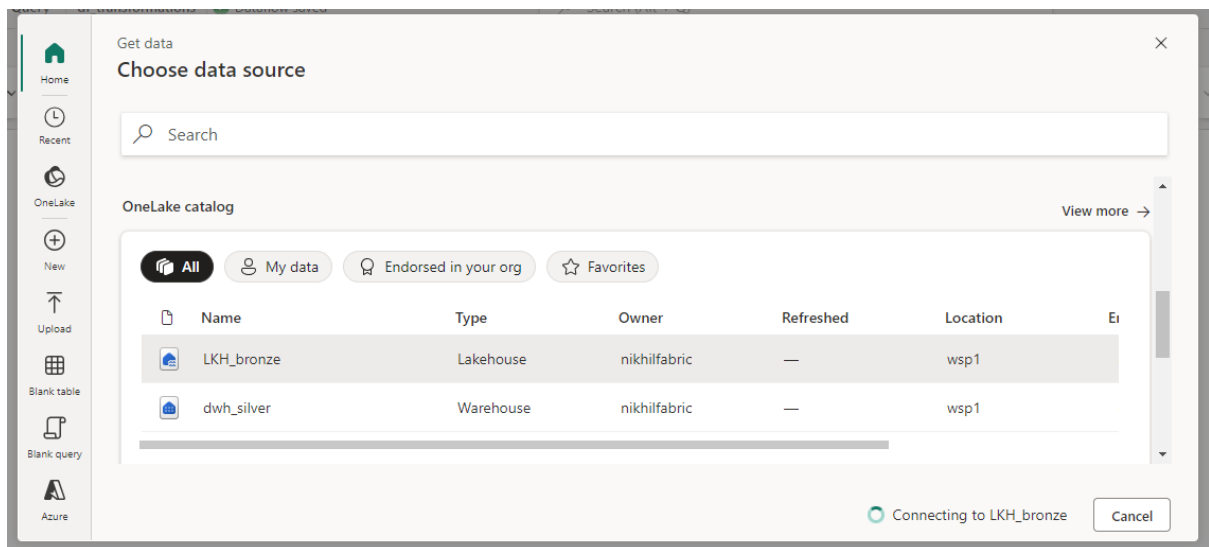


→create.

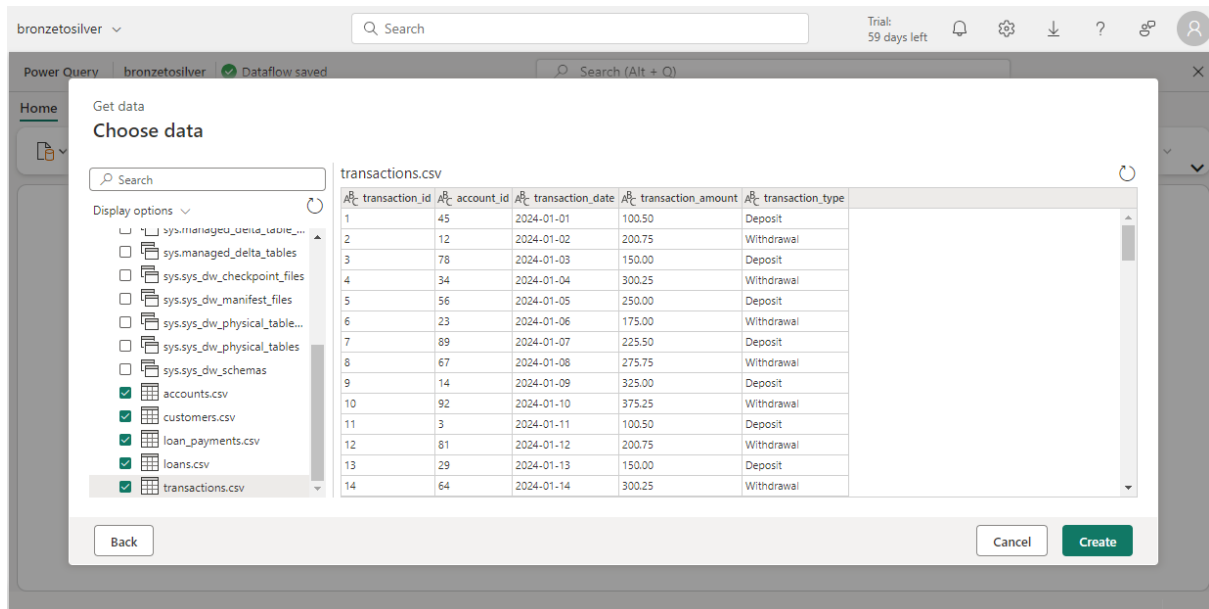
→Dataflow will open, from the left top get data→more



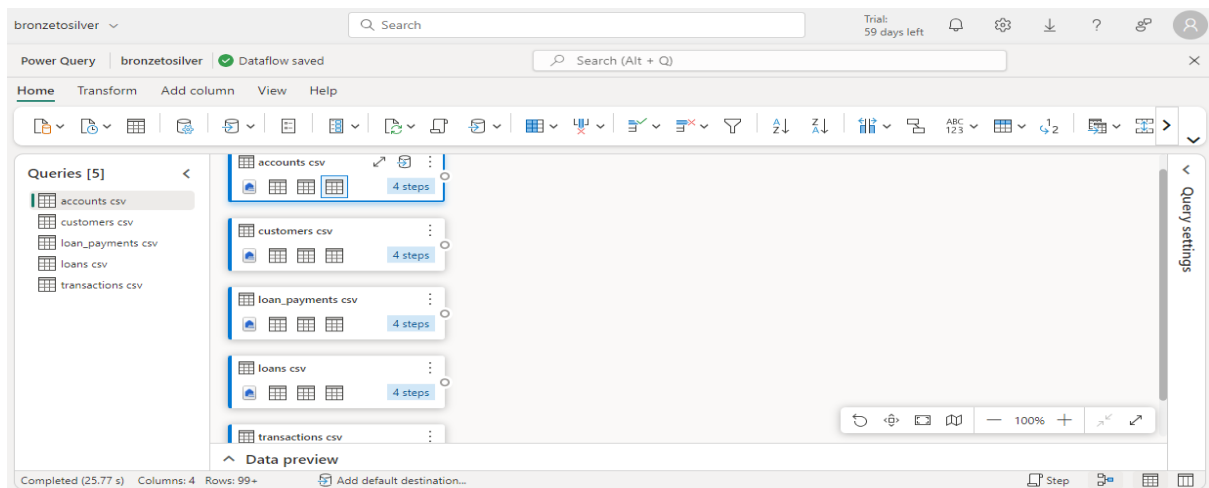
→choose the data source(since the data is LKH\_bronze ,choose lakehouse as data source).



→select the lakehouse



Choose the Tables to be transformed and click create.

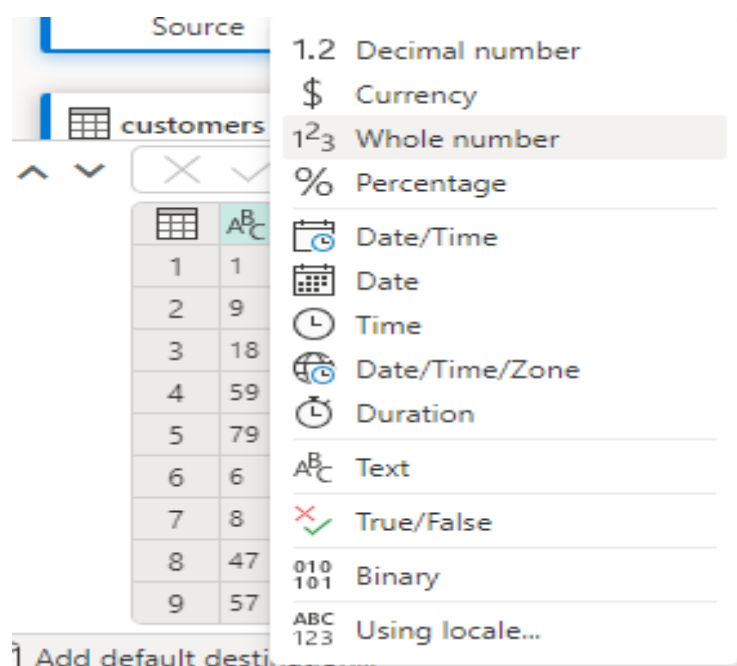


→click on accounts.csv and expand the stream.

	ABC account_id ▾	ABC customer_id ▾	ABC account_type ▾	ABC balance ▾
1	1	45	Savings	1000.50
2	9	14	Savings	900.25
3	18	5	Checking	1600.50
4	59	75	Savings	475.75
5	79	55	Savings	725.75
6	6	23	Checking	1200.50
7	8	67	Checking	2200.00
8	47	95	Savings	325.75
9	57	97	Savings	450.25

**Data Type conversion:** From the above table for account\_id, customer\_id and balance data type is text(string).

→To convert the type, click on the ABC to change the type→a data type list will appear and choose the appropriate data type.



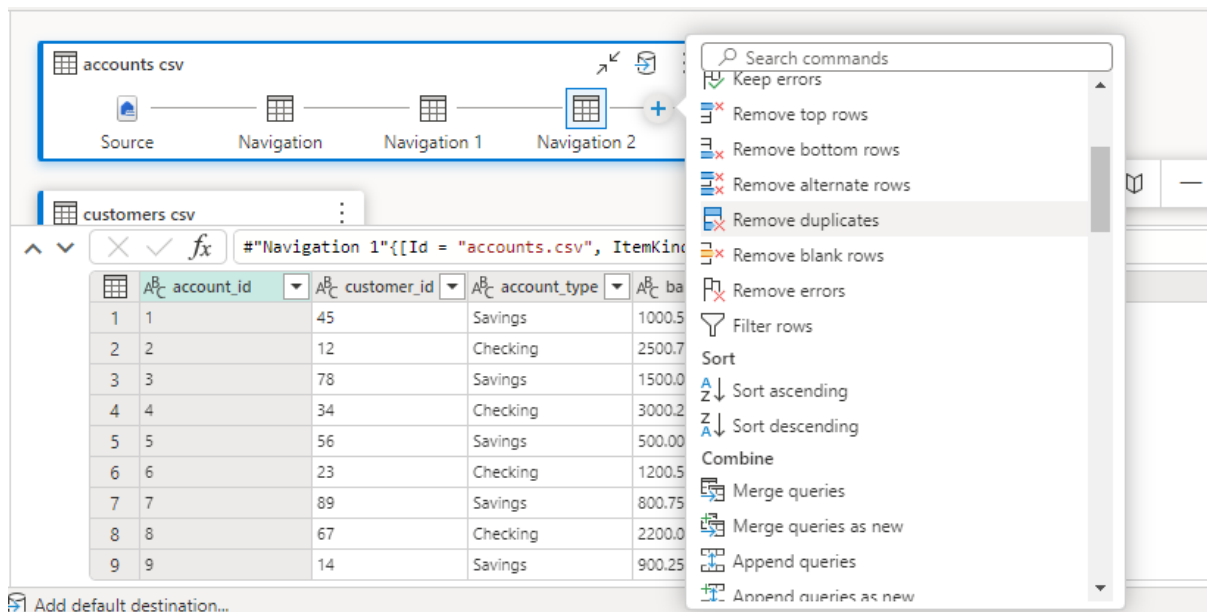
→select the whole number for account\_id and customer\_id

→select the decimal for balance.

**Removing Duplicates:** To remove the duplicates, select all the columns and then click on + from the stream.

From the options → select remove duplicates

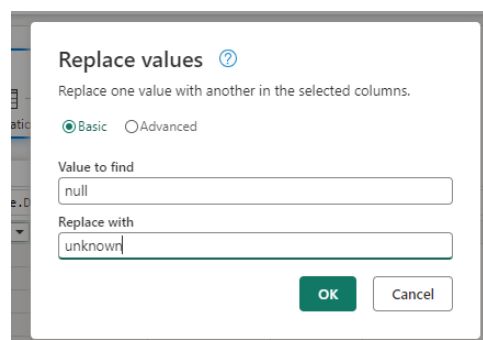
This function will remove the duplicate from the selected columns



**Replacing NULL:** To replace the null values in dataflow, click on the + from the stream and select the **replace values** or from the top in transform tab select the replace values

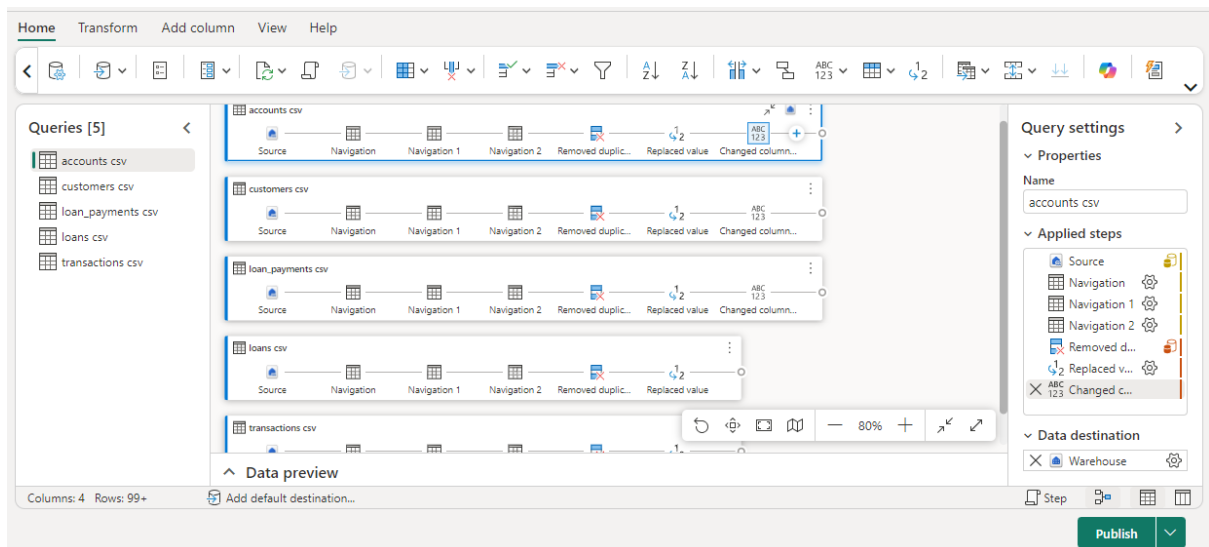
54	73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1
55	30	Emily	Jordan	7979 Cypress Ave	North Bay	ON	P1B0A1
56	37	William	McDonald	8686 Maple Ave	Haileybury	null	null
57	30	Elizabeth	Stewart	2929 Elm St	Peterborough	ON	K9H0A1
58	33	John	Rogers	3232 Pine Rd	Timmins	ON	P4N0A1

From the above a record has null values



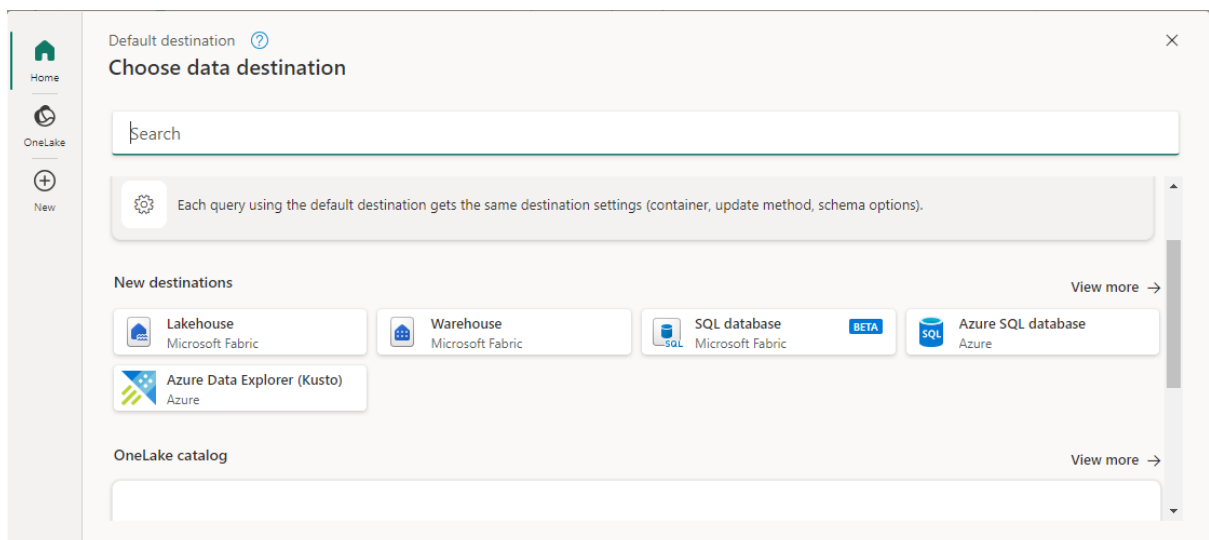
54	73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1
55	80	Emily	Jordan	7979 Cypress Ave	North Bay	ON	P1B0A1
56	87	William	McDonald	8686 Maple Ave	Haileybury	unknown	unknown
57	30	Elizabeth	Stewart	2929 Elm St	Peterborough	ON	K9H0A1
58	33	John	Rogers	3232 Pine Rd	Timmins	ON	P4N0A1

Null values has been replaced with unknow from the above.



Above is the transformations overview.


**Add Destination :** add destination to the each stream by clicking on the three dots or at the bottom Add default destination



Choose the datawarehouse as destination

Default destination

### Connect to default data destination

**Warehouse**  
Microsoft Fabric


#### Connection credentials

Connection  
Create new connection

Connection name  
Connection

Data gateway  
(none)

Authentication kind  
Organizational account

You are currently signed in as:  
 **nikhilfabric**  
nikhil@raka1221outlook.onmicrosoft.c...

Back

Cancel

Next

Create a connection with datawarehouse.

Default destination

### Choose default destination target

For performance reasons, only Warehouses in the current workspace are shown.

Search

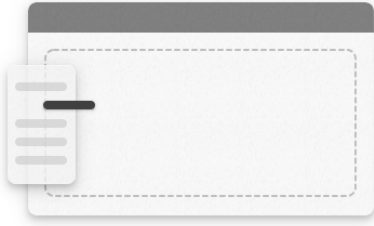
Display options

Warehouse [3]

DataflowsStagingWarehouse

dwh\_gold

dwh\_silver



Please select a default destination.

Back

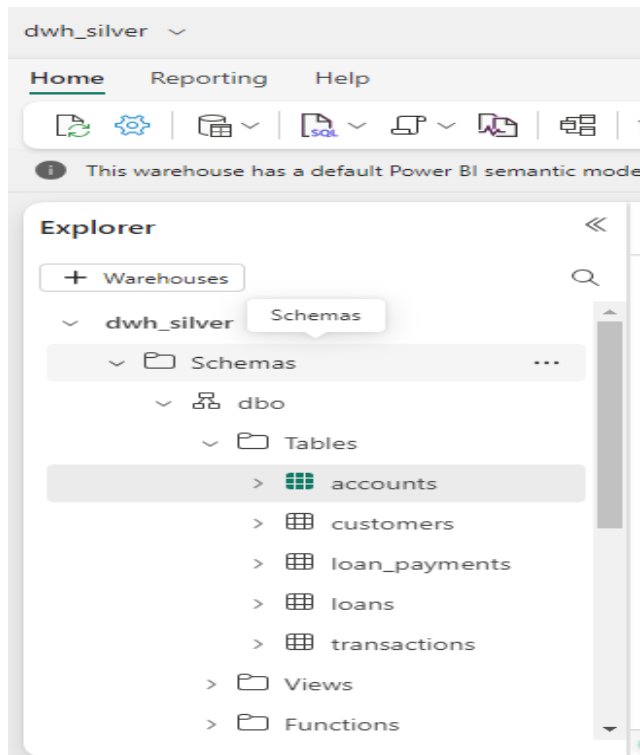
Cancel

Choose

Choose the datawarehouse from the options and click choose.

The Publish the dataflow, the data will be written to datawarehouse.

## Output

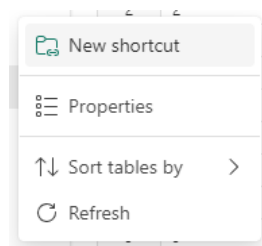


## Silver to Gold : Using the Notebook

→ Create a notebook in workspace.

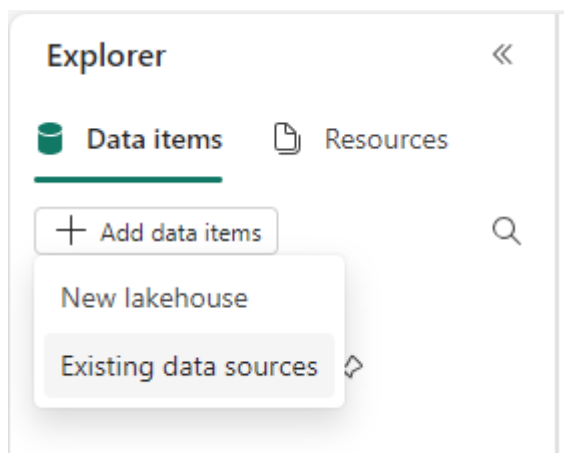
→ before accessing the notebook, create a shortcut in LKH\_bronze for dwh\_silver. Since, notebook cannot access the data from the datawarehouse directly.

→ creating shortcut in LKH\_bronze, click on table menu → select new shortcut → select internal sources and datawarehouse.





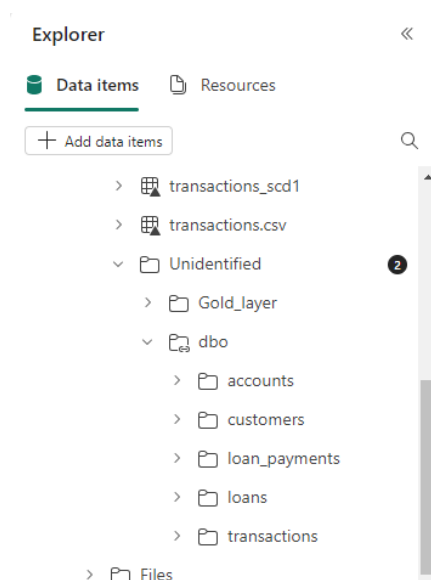
→open the notebook and attach the LKH\_bronze from explorer tab→add items→existing data sources



Discover data from your org and beyond and use it to create reports

Discover data from your org and beyond and use it to create reports						
<div><span>All</span> <span>My data</span> <span>Endorsed in your org</span> <span>Favorites</span> <span>Filter by keyword</span></div>						
Explorer	Name	Owner	Refreshed	Location	Endorsement	Sensitivity
<input checked="" type="checkbox"/>	LKH_bronze	nikhilfabric	—	wsp1	—	—
<input type="checkbox"/>	DataflowsStagingLakehouse	nikhilfabric	—	wsp1	—	—

From the option choose the lakehouse and connect.



Lakehouse has been attached to notebook and data can be retrieved from here directly.

→Turn on the session from the connect on top.

## Implementing SCD Type 1 Logic on Tables.

→Accounts,

### 1. Create a SQL table

```
%%sql
CREATE table if not EXISTS accounts_scd1(
account_id int,
customer_id int,
account_type varchar(50),
balance decimal(12,2),
hashkey bigint,
createdby varchar(50),
createddate timestamp,
updatedby varchar(50),
updateddate timestamp
)
using DELTA
LOCATION 'Tables/Gold_layer/accounts_scd1'
```

```
1  %%sql
2  CREATE table if not EXISTS accounts_scd1(
3  account_id int,
4  customer_id int,
5  account_type varchar(50),
6  balance decimal(12,2),
7  hashkey bigint,
8  createdby varchar(50),
9  createddate timestamp,
10 updatedby varchar(50),
11 updateddate timestamp
12 )
13 using DELTA
14 LOCATION 'Tables/Gold_layer/accounts_scd1'
```

✓ - Command executed in 21 sec 122 ms by nikhilfabric on 11:43:52 PM, 5/05/25

Gold\_layer is kind of folder in LKH\_bronze.

→define source and target path by using a variable. Read the data from the source.

```
1  src_path='Tables/dbo/accounts'
2  tgt_path='Tables/Gold_layer/accounts_scd1'
3  acc_src=spark.read.format('delta').load(src_path)|
```

✓ - Command executed in 4 sec 631 ms by nikhilfabric on 10:34:57 AM, 5/06/25

→import pyspark.sql.functions, to perform operations

```
1 from pyspark.sql.functions import *
```

✓ - Command executed in 284 ms by nikhilfabric on 10:35:00 AM, 5/06/25

→now add the hashkey column to the source data using withColumn,crc32 and concat.

```
1 acc_src=acc_src.withColumn('hashkey',crc32(concat(*acc_src.columns)))
```

✓ - Command executed in 272 ms by nikhilfabric on 10:35:03 AM, 5/06/25

→Now import the deltatables

```
1 from delta.tables import DeltaTable
2 deltatable=DeltaTable.forPath(spark,tgt_path)
3 deltatable.toDF().show()
```

✓ - Command executed in 6 sec 164 ms by nikhilfabric on 10:35:17 AM, 5/06/25

PySpark (Python) ▾

```
+-----+-----+-----+-----+-----+-----+-----+
---+
|account_id|customer_id|account_type|balance|hashkey|createdby|createddate|updatedby|
|updateddate|
+-----+-----+-----+-----+-----+-----+-----+
---+
```

Imports the DeltaTable class from the Delta Lake library.

Creates a DeltaTable object pointing to the Delta table stored at tgt\_path.

→join the source dataframe with delta table on a matching condition using **anti join**. And select all columns from source.

```
1 acc_src=acc_src.alias("acc").join(deltatable.toDF().alias("tgt"),\
2 |((col("acc.account_id")==col("tgt.account_id"))&\
3 | (col("acc.hashkey")==col("tgt.hashkey"))),"anti").select(col("acc.*"))
```

✓ - Command executed in 293 ms by nikhilfabric on 10:35:22 AM, 5/06/25

PySpark (Python) ▾

→ Now merge the delta table with source dataframe on a condition.

```
1 deltable.alias("tgt").merge(acc_src.alias("acc"),"tgt.account_id=acc.account_id")\
2   .whenMatchedUpdate(set={"tgt.account_id":"acc.account_id","tgt.customer_id":"acc.customer_id",\
3     "tgt.account_type":"acc.account_type","tgt.balance":"acc.balance","tgt.hashkey":"acc.hashkey",\
4     "tgt.updatedby":lit("fabric"),"tgt.updateddate":current_timestamp()})\
5   .whenNotMatchedInsert(values={"tgt.account_id":"acc.account_id","tgt.customer_id":"acc.customer_id",\
6     "tgt.account_type":"acc.account_type","tgt.balance":"acc.balance","tgt.hashkey":"acc.hashkey",\
7     "tgt.createdby":lit("fabric"),"tgt.createddate":current_timestamp(),\
8     "tgt.updatedby":lit("fabric"),"tgt.updateddate":current_timestamp()})\
9   .execute()
```

✓ - Command executed in 7 sec 843 ms by nikhilfabric on 10:35:32 AM, 5/06/25

PySpark (Python)

## Output

## Accounts

1 display(spark.read.format('delta').load('Tables/Gold\_layer/accounts\_scd1'))

✓ 3 sec - Command executed in 3 sec 374 ms by nikhilfabric on 3:35:56 PM, 5/06/25

PySpark (Python)

Spark jobs (6 of 6 succeeded) Resources

Table view

	123 account_id	123 customer_id	ABC account_type	1.2 balance	12L hashkey	ABC createdby	createddate	ABC updatedby	updateddate
1	1	45	Savings	1000.50	2454403084	fabric	2025-05-06 03:1...	fabric	2025-05-
2	9	14	Savings	900.25	1397533172	fabric	2025-05-06 03:1...	fabric	2025-05-
3	18	5	Checking	1600.50	544941373	fabric	2025-05-06 03:1...	fabric	2025-05-
4	59	75	Savings	475.75	1084664362	fabric	2025-05-06 03:1...	fabric	2025-05-
5	79	55	Savings	725.75	1919448738	fabric	2025-05-06 03:1...	fabric	2025-05-
6	6	23	Checking	1200.50	589336936	fabric	2025-05-06 03:1...	fabric	2025-05-
7	8	67	Checking	2200.00	3401041467	fabric	2025-05-06 03:1...	fabric	2025-05-
8	47	95	Savings	325.75	3742642303	fabric	2025-05-06 03:1...	fabric	2025-05-
9	57	97	Savings	450.25	1548801870	fabric	2025-05-06 03:1...	fabric	2025-05-
10	66	26	Checking	6700.50	372242038	fabric	2025-05-06 03:1...	fabric	2025-05-
11	40	19	Checking	4100.00	2070805942	fabric	2025-05-06 03:1...	fabric	2025-05-

Session ready AutoSave: On Selected Cell 8 of 32

## Customers

1 display(spark.read.format('delta').load('Tables/Gold\_layer/customers\_scd1'))

✓ 3 sec - Command executed in 3 sec 700 ms by nikhilfabric on 3:37:04 PM, 5/06/25

PySpark (Python)

Spark jobs (6 of 6 succeeded) Resources

Table view

	123 customer_id	ABC first_name	ABC last_name	ABC address	ABC city	ABC state	ABC zip	12L hashkey	ABC createdby	createddate
1	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1	3284019540	databricks	2025-05-
2	42	Charlotte	Richardson	4141 Beech Dr	Newmarket	ON	L3Y0A1	2954704707	databricks	2025-05-
3	49	Joshua	Bennett	4848 Spruce...	Beaverton	ON	L0K0A1	1307334809	databricks	2025-05-
4	52	Abigail	Henderson	5151 Cypres...	Mount Albert	ON	L0G0A1	569556118	databricks	2025-05-
5	2	Jane	Smith	456 Maple A...	Ottawa	ON	K1A0B1	3911254336	databricks	2025-05-
6	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1	165372536	databricks	2025-05-
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1	4278935663	databricks	2025-05-
8	14	Sophia	Young	1313 Beech Dr	Yellowknife	NT	X1A0A1	1246741277	databricks	2025-05-
9	75	Joshua	Sullivan	7474 Pine Rd	Bracebridge	ON	P1L0A1	3554568556	databricks	2025-05-
10	11	Alexander	Thomas	1010 Willow...	St. John's	NL	A1A0A1	823092523	databricks	2025-05-
11	20	Mia	Nelson	1919 Birch B...	London	ON	N6A0A1	1253444537	databricks	2025-05-

# Loans

1

display(spark.read.format('delta').load('Tables/Gold\_layer/loans\_scd1'))

[31]

✓ 3 sec - Running

PySpark (Pyth

Spark jobs In progress (2)

Resources

Table

New chart

10 columns, 100 rows

Table view

Download

Search

	123 loan_id	123 customer_id	12F loan_amount	12F interest_rate	123 loan_term	12L hashkey	ABC createdby	createddate	ABC updated
1	5	56	25000.0	5.0	36	951778087	databricks	2025-05-06 13:5...	databric
2	25	66	25000.25	5.5	36	549737095	databricks	2025-05-06 13:5...	databric
3	42	36	20000.5	4.5	48	3521046828	databricks	2025-05-06 13:5...	databric
4	45	68	25000.25	5.0	36	1252029156	databricks	2025-05-06 13:5...	databric
5	54	42	30000.5	4.0	48	1545195748	databricks	2025-05-06 13:5...	databric
6	74	43	30000.5	4.5	48	333538238	databricks	2025-05-06 13:5...	databric
7	87	93	22500.75	6.5	60	3240117383	databricks	2025-05-06 13:5...	databric
8	53	86	15000.25	5.0	36	95809912	databricks	2025-05-06 13:5...	databric
9	92	44	20000.0	3.5	24	117152147	databricks	2025-05-06 13:5...	databric
10	4	34	30000.25	3.5	24	2847880128	databricks	2025-05-06 13:5...	databric
11	40	19	37500.0	3.0	24	3372912529	databricks	2025-05-06 13:5...	databric

# Loanpayments

1

display(spark.read.format('delta').load('Tables/Gold\_layer/loanpayments\_scd1'))

[32]

✓ 2 sec - Command executed in 2 sec 402 ms by nikhilfabric on 3:38:46 PM, 5/06/25

PySpark (Pytl

Spark jobs (6 of 6 succeeded)

Resources

Table

New chart

9 columns, 100 rows

Table view

Download

Search

	123 payment_id	123 loan_id	12F payment_date	12F payment_amount	12L hashkey	ABC createdby	createddate	ABC updatedby	upc
1	20	21	2024-01-20	1050.0	1615996424	databricks	2025-05-06 14:0...	databricks	20:
2	61	72	2024-03-01	3100.0	3924611863	databricks	2025-05-06 14:0...	databricks	20:
3	65	16	2024-03-05	3300.0	4085199564	databricks	2025-05-06 14:0...	databricks	20:
4	83	14	2024-03-23	4200.0	2734147888	databricks	2025-05-06 14:0...	databricks	20:
5	94	35	2024-04-03	4750.0	1343994555	databricks	2025-05-06 14:0...	databricks	20:
6	33	64	2024-02-02	1700.0	3793101080	databricks	2025-05-06 14:0...	databricks	20:
7	46	7	2024-02-15	2350.0	909840285	databricks	2025-05-06 14:0...	databricks	20:
8	52	73	2024-02-21	2650.0	2180006131	databricks	2025-05-06 14:0...	databricks	20:
9	12	33	2024-01-12	650.0	2585655648	databricks	2025-05-06 14:0...	databricks	20:
10	58	39	2024-02-27	2950.0	618787123	databricks	2025-05-06 14:0...	databricks	20:
11	74	15	2024-03-14	3750.0	3135475598	databricks	2025-05-06 14:0...	databricks	20:

# Transactions

1

display(spark.read.format('delta').load('Tables/Gold\_layer/transactions\_scd1'))

[33]

<1 sec - Command executed in 1 sec 585 ms by nikhilfabric on 3:39:41 PM, 5/06/25

PySpark (Python)

Spark jobs (2 of 2 succeeded)

Resources

Table

New chart

10 columns, 100 rows

Table view

Download

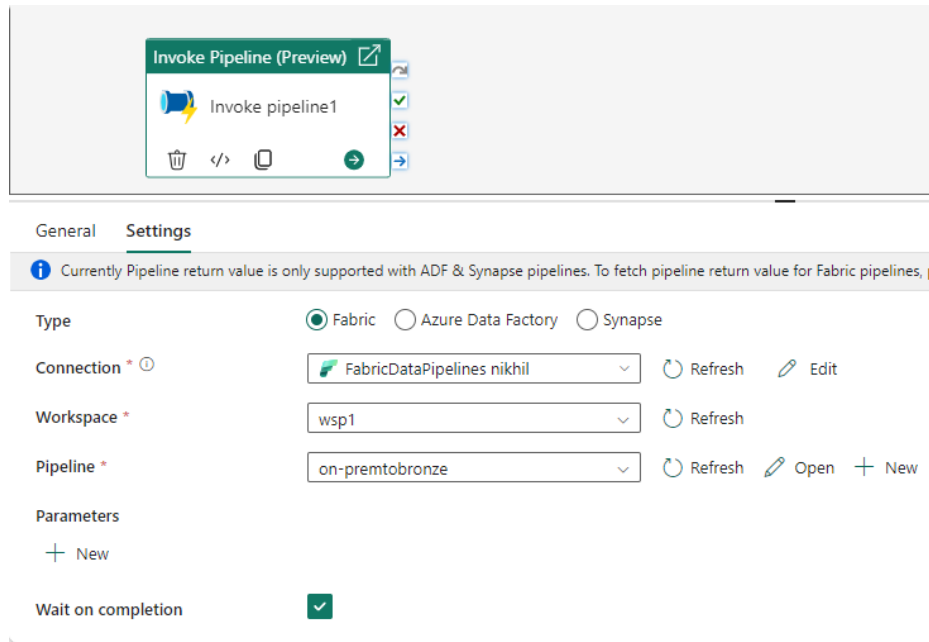
Search

	123 transaction_id	123 account_id	12F transaction_date	12F transaction_amount	ABC transaction_type	12L hashkey	ABC createdby	createddate
1	38	15	2024-02-07	275.75	Withdrawal	1006322033	fabric	2025-05-
2	43	83	2024-02-12	150.0	Deposit	4257851425	fabric	2025-05-
3	54	42	2024-02-23	300.25	Withdrawal	3238602855	fabric	2025-05-
4	79	55	2024-03-19	325.0	Deposit	1630795787	fabric	2025-05-
5	92	44	2024-04-01	200.75	Withdrawal	1187297032	fabric	2025-05-
6	7	89	2024-01-07	225.5	Deposit	1987130936	fabric	2025-05-
7	12	81	2024-01-12	200.75	Withdrawal	3980785899	fabric	2025-05-
8	31	71	2024-01-31	100.5	Deposit	1417673545	fabric	2025-05-
9	66	26	2024-03-06	175.0	Withdrawal	3056144572	fabric	2025-05-
10	85	65	2024-03-25	250.0	Deposit	3638886954	fabric	2025-05-
11	35	62	2024-02-04	250.0	Deposit	717510064	fabric	2025-05-

## Scheduling Master Pipeline.

→ Create a pipeline in workspace.

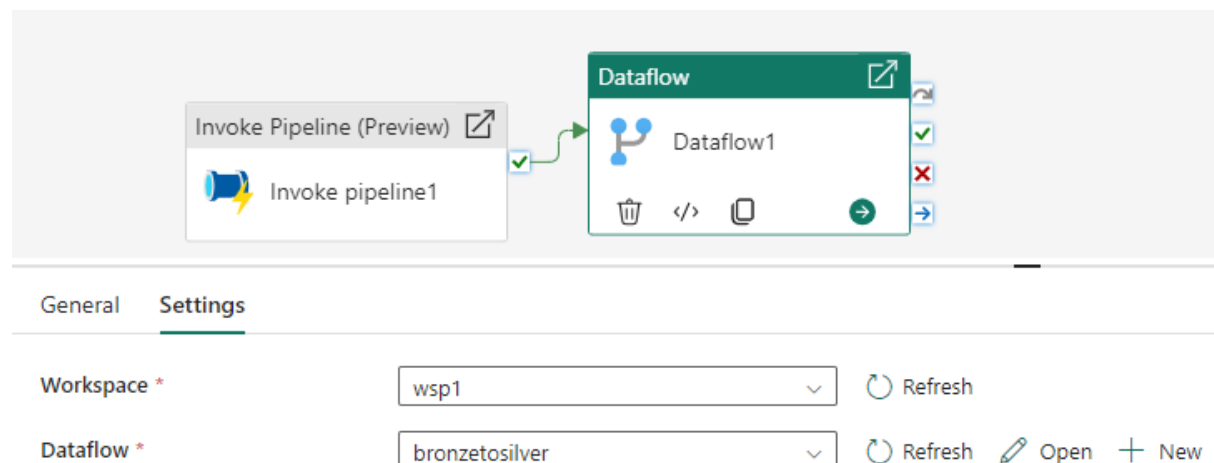
→ under activities → select the **invoke pipeline**.



The screenshot shows the 'Invoke Pipeline (Preview)' activity configuration in the settings tab. The activity is named 'Invoke pipeline1'. Below the activity name, there are icons for deleting, editing code, copying, and running. The settings section includes a warning message: 'Currently Pipeline return value is only supported with ADF & Synapse pipelines. To fetch pipeline return value for Fabric pipelines, p'. The configuration options are:

- Type: ☒ Fabric ☐ Azure Data Factory ☐ Synapse
- Connection \*:  Refresh Edit
- Workspace \*:  Refresh
- Pipeline \*:  Refresh Open + New
- Parameters: + New
- Wait on completion: ☒

→ on success connect to dataflow activity.

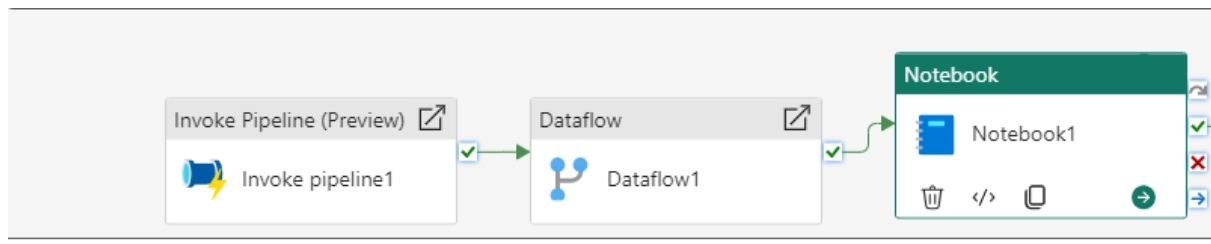


The screenshot shows the 'Dataflow' activity configuration in the settings tab. The activity is named 'Dataflow1'. Below the activity name, there are icons for deleting, editing code, copying, and running. The settings section includes:

- Workspace \*:  Refresh
- Dataflow \*:  Refresh Open + New

A green arrow indicates a connection from the 'Invoke Pipeline' activity to this 'Dataflow' activity.

→on success connect to notebook.



General **Settings**

**i** Please review this item carefully before adding it to the pipeline, as others in your organization may have access to notebooks in this workspace.

Workspace \*

wsp1

Refresh

Notebook \*

Notebook 1

Refresh

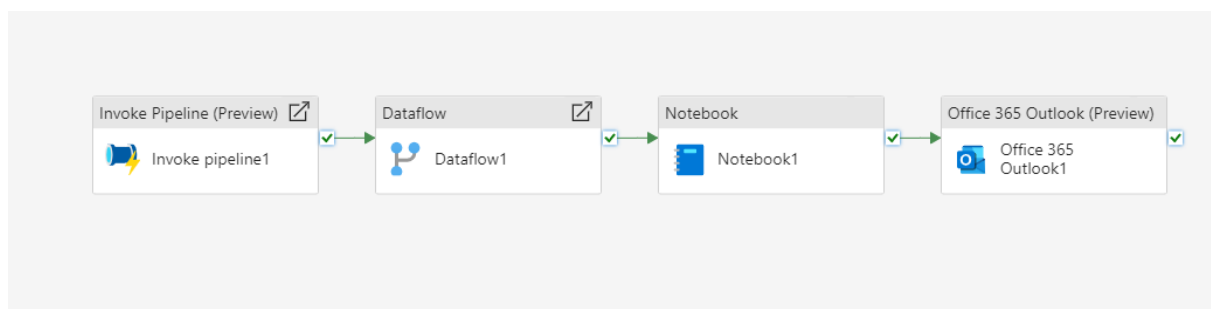
Open + New

> Base parameters

> Advanced settings


→finally connect to e-mail activity.

Overview of the master pipeline.



Schedule the above pipeline


→ In the home tab, select schedule.


**pl\_master**  
Data pipeline


[About](#)  
[Endorsement](#)  
**[Schedule](#)**

**Scheduled run**  
☒ On ☐ Off

**Repeat**  
Daily ▾

**Time**  
07:00 ⌚   
[+ Add a time](#)

**Start date and time**  
06-05-2025 

**End date and time**  
06-06-2025 

**Time zone**  
(UTC-04:00) Atlantic Time (Canada) ▾

**Apply** Discard

→ apply.