INCREMENTAL DATA LOADING AND AUTOMATED NOTIFICATIONS USING MICROSOFT FABRIC

Bootcamp Project - 4

MAY 6, 2025

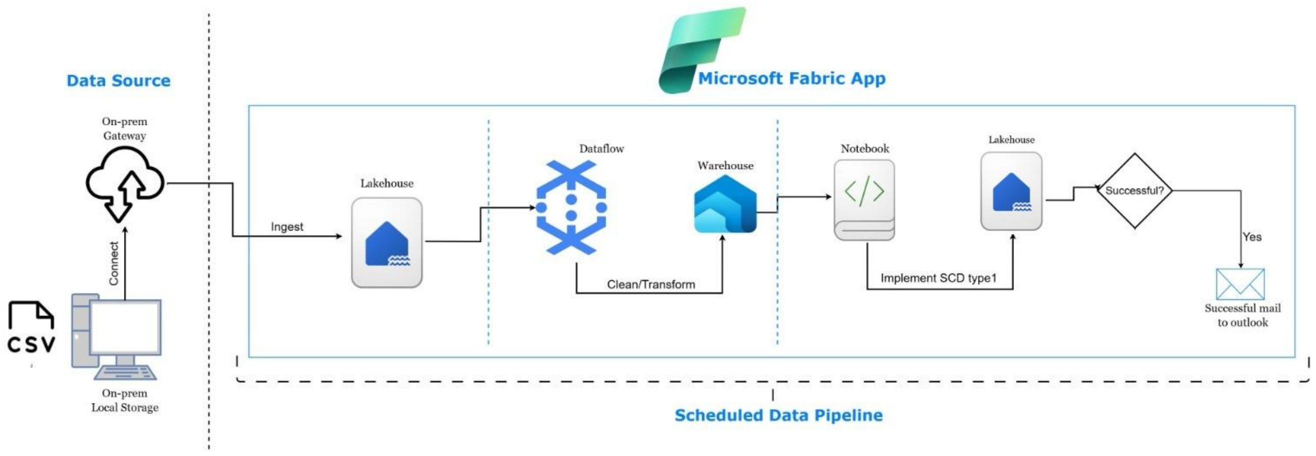
BY HARJINDER SINGH

Table of Contents

Introduction

In today’s data-driven landscape, organizations rely heavily on timely, accurate, and integrated data pipelines to support analytics and decision-making. The project titled **"Incremental Data Loading and Automated Notifications using Microsoft Fabric"** focuses on building a robust, end-to-end solution that enables seamless ingestion, transformation, and monitoring of data workflows. Leveraging Microsoft Fabric's powerful capabilities—including Dataflow Gen 1, Fabric Notebooks, and Warehouse integration—this project demonstrates how to incrementally load data from on-premises environments, apply structured data transformation logic such as SCD Type 1, and ensure data quality through deduplication and cleaning. Additionally, the solution integrates automated email notifications to provide real-time updates on pipeline success, enhancing operational transparency and reliability. This comprehensive approach not only ensures efficient data processing but also streamlines communication and monitoring in modern data platforms.

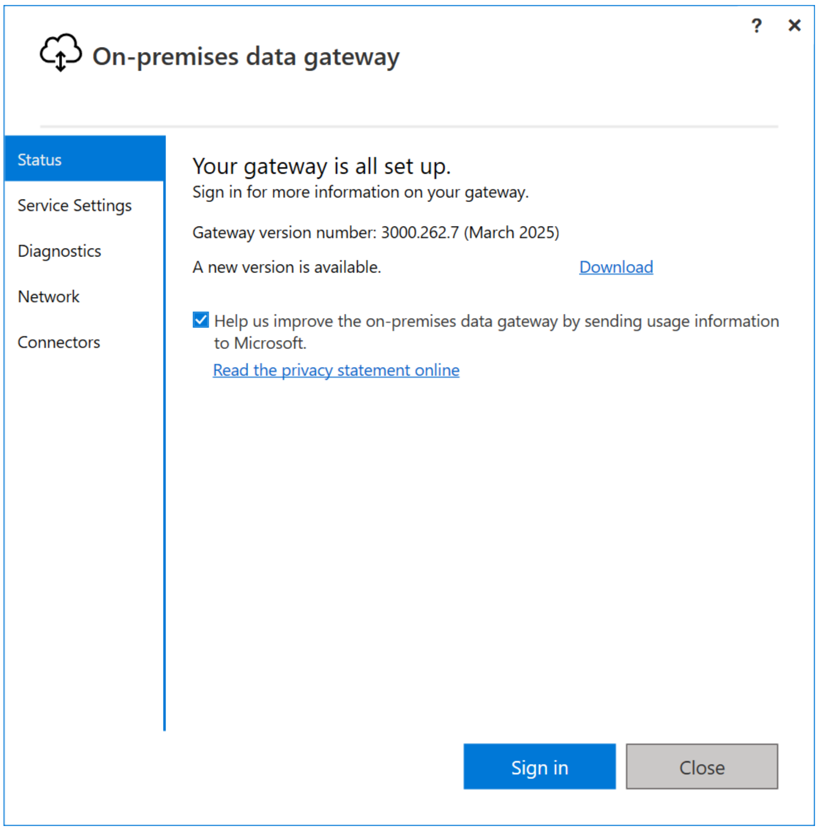
Architecture Followed:



On-Prem Gateway Setup

➢ **Steps To Setup the on-prem Gateway and connect to it with Fabric are as follows.**

Step 1: Download and Install the On-prem data gateway in the local machine. And sign in with user email (same fabric loggedin email). Click on Sign in.



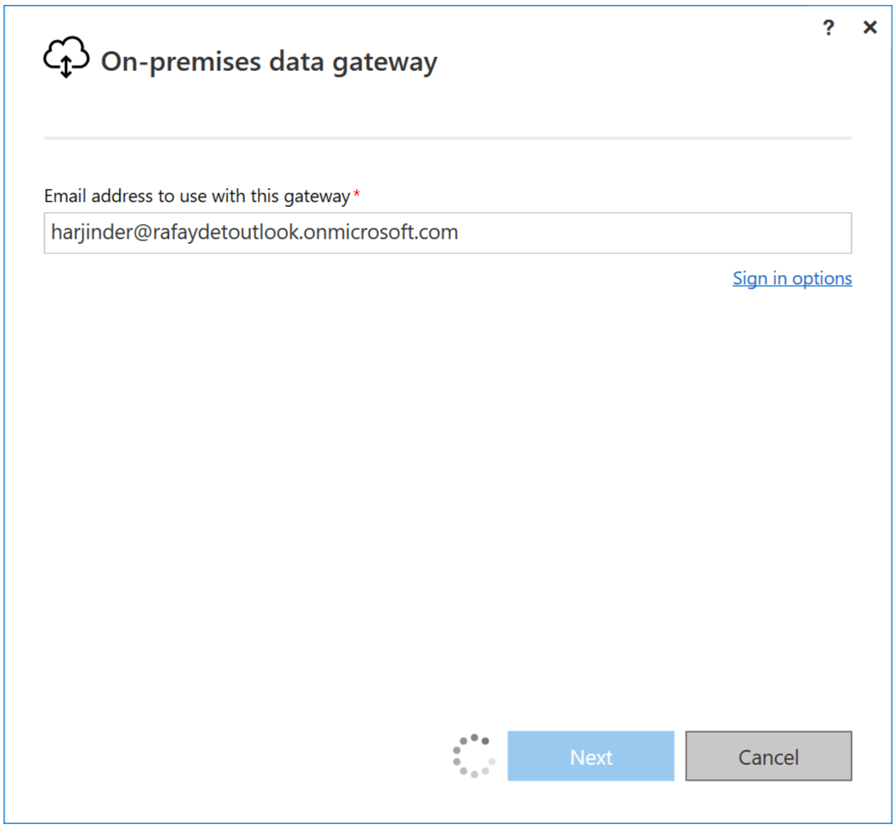
Your gateway is all set up.
Sign in for more information on your gateway.

Gateway version number: 3000.262.7 (March 2025)

A new version is available.

☑ Help us improve the on-premises data gateway by sending usage information to Microsoft.

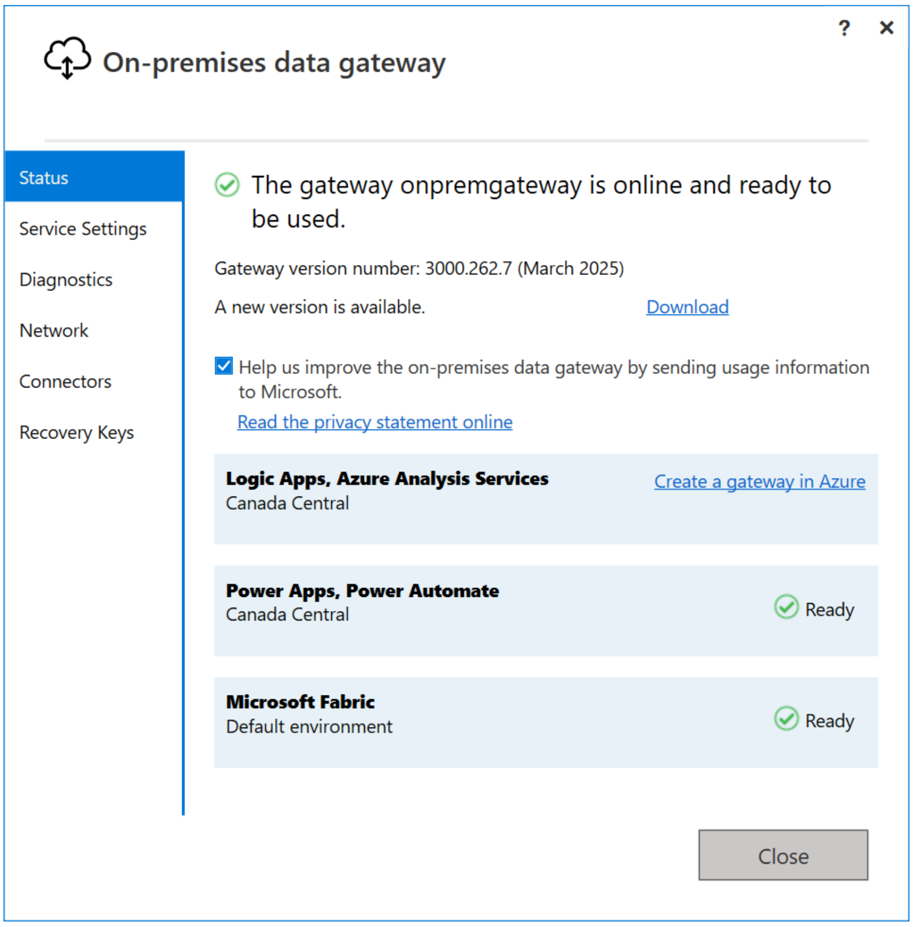
Step 2: After sign in, it will ask to create for **Gateway Cluster name** and **Gateway Recovery password**. Create the both.


On-premises data gateway

Email address to use with this gateway*

harjinder@rafaydetoutlook.onmicrosoft.com

Sign in options


On-premises data gateway

The gateway onpremgateway is online and ready to be used.

Gateway version number: 3000.262.7 (March 2025)

A new version is available. Download

☑ Help us improve the on-premises data gateway by sending usage information to Microsoft.

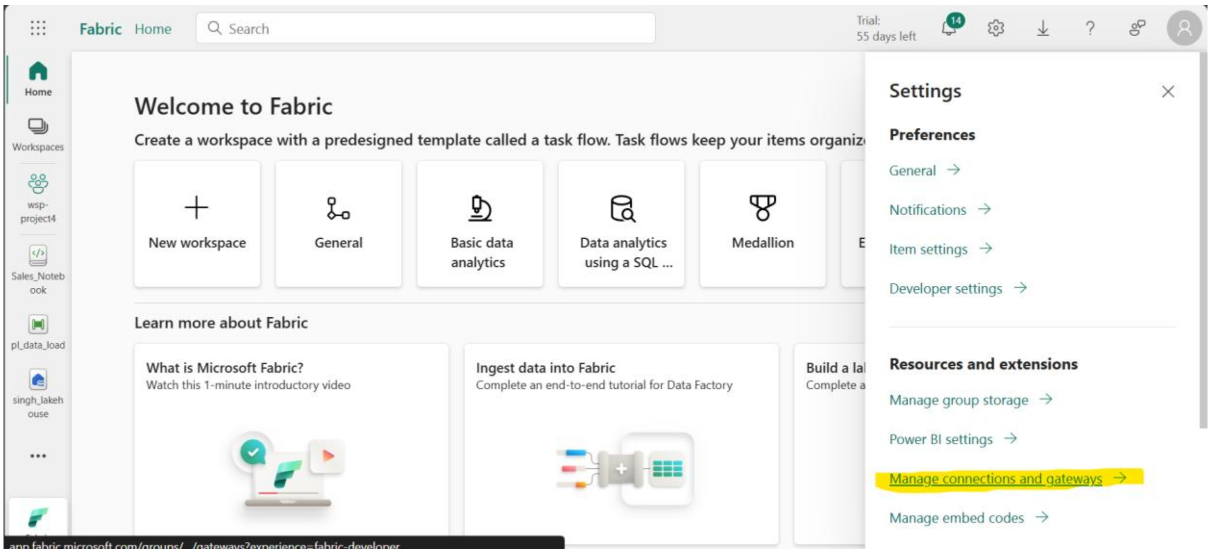
Read the privacy statement online

Logic Apps, Azure Analysis Services Canada Central — Create a gateway in Azure

Power Apps, Power Automate Canada Central — ✓ Ready

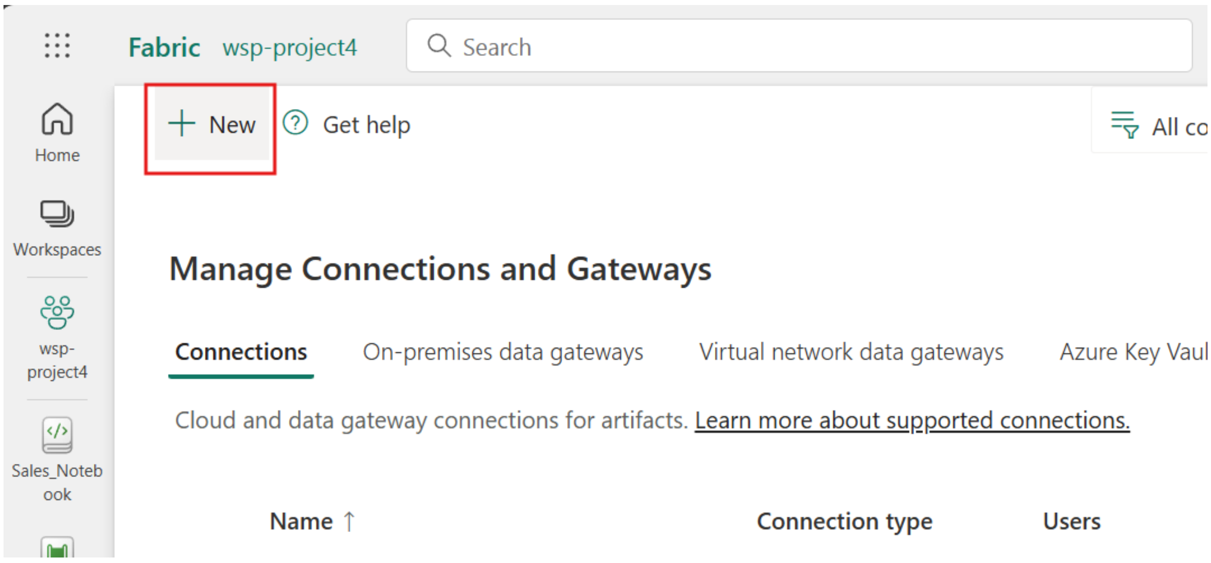
Microsoft Fabric Default environment — ✓ Ready

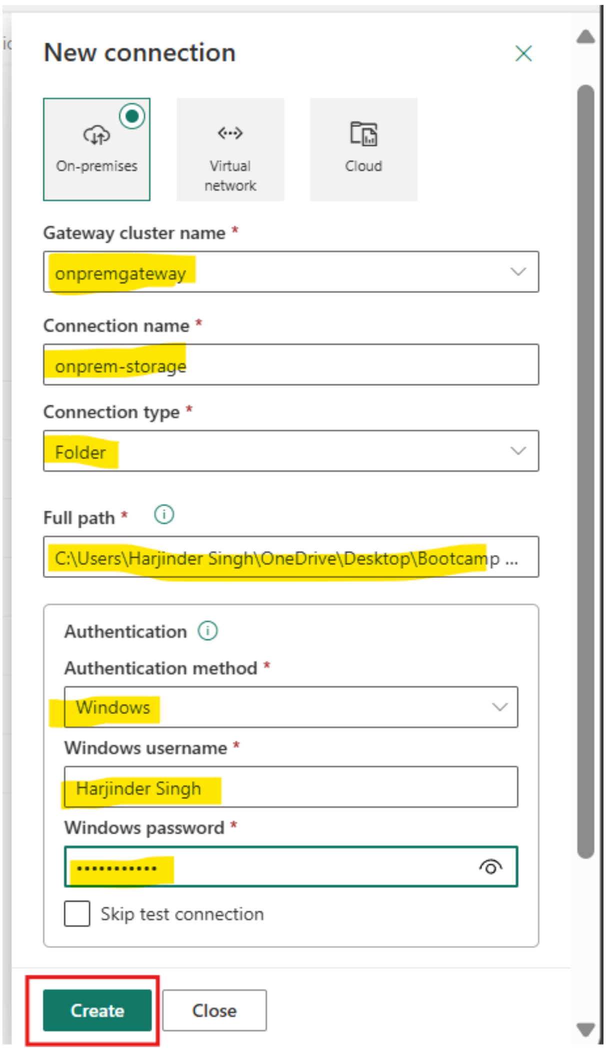
Step 3: Next, Goto Fabric App -> settings -> open **manage connections and gateways.**



Step 4: Next, Click on new -> Enter below connection details -> click on create.

– Select **On-premises** connection and give the cluster name same as created in gateway.
– Select connection type as folder and give the path.
– Enter the window username and password.


Fabric wsp-project4
Search
New
Get help
Home
Workspaces
wsp-project4
Sales_Notebook
Manage Connections and Gateways
Connections
On-premises data gateways
Virtual network data gateways
Cloud and data gateway connections for artifacts. Learn more about supported connections.
Name
Connection type
Users


New connection

On-premises [✓] | Virtual network | Cloud

Gateway cluster name *
onpremgateway

Connection name *
onprem-storage

Connection type *
Folder

Full path *
C:\Users\Harjinder Singh\OneDrive\Desktop\Bootcamp ...

Authentication

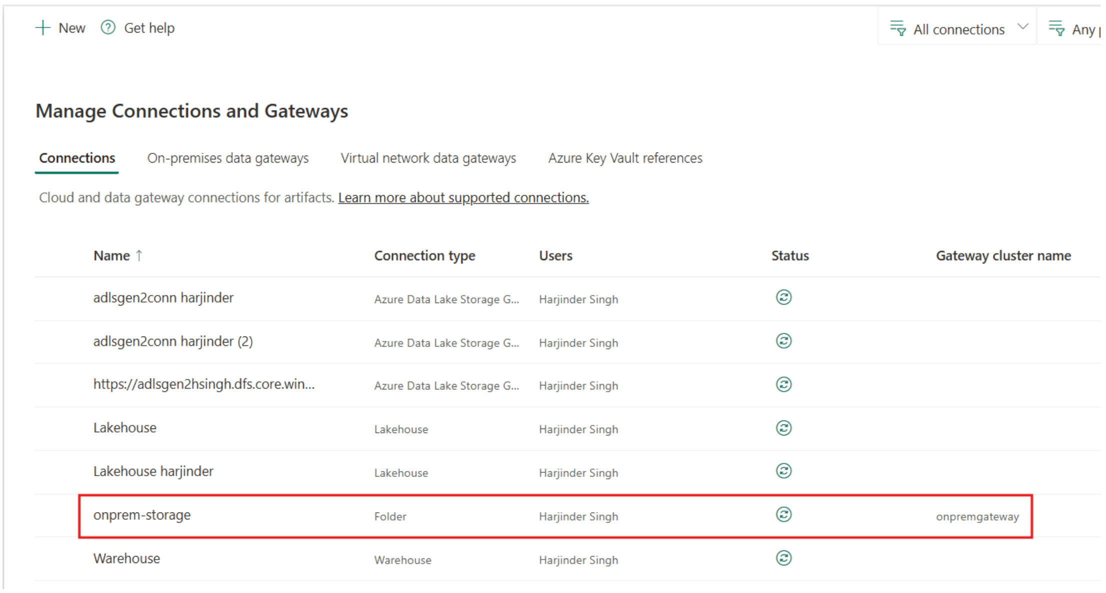
Authentication method *
Windows

Windows username *
Harjinder Singh

Windows password *
•••••••••••

☐ Skip test connection

– Verify the connection in connections tab.


+ New
Get help
All connections
Any

Manage Connections and Gateways

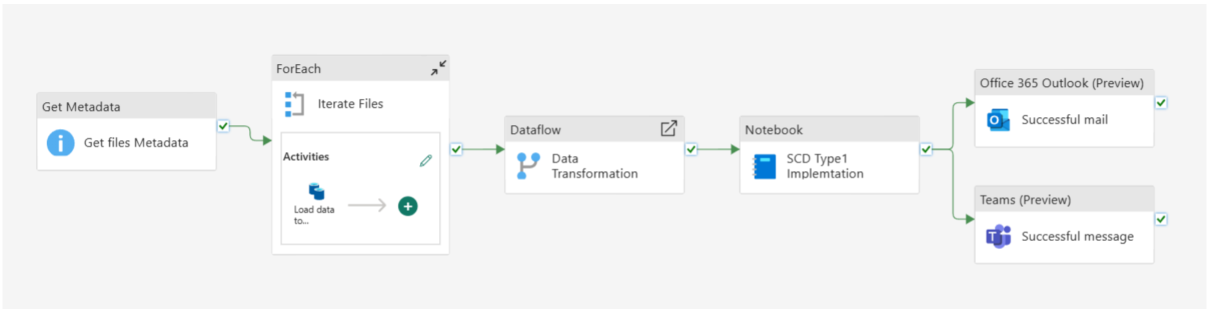
Connections
On-premises data gateways
Virtual network data gateways
Azure Key Vault references

Cloud and data gateway connections for artifacts. Learn more about supported connections.

Name ↑	Connection type	Users	Status	Gateway cluster name
adlsgen2conn harjinder	Azure Data Lake Storage G...	Harjinder Singh	☺	
adlsgen2conn harjinder (2)	Azure Data Lake Storage G...	Harjinder Singh	☺	
https://adlsgen2hsingh.dfs.core.win...	Azure Data Lake Storage G...	Harjinder Singh	☺	
Lakehouse	Lakehouse	Harjinder Singh	☺	
Lakehouse harjinder	Lakehouse	Harjinder Singh	☺	
onprem-storage	Folder	Harjinder Singh	☺	onpremgateway
Warehouse	Warehouse	Harjinder Singh	☺	

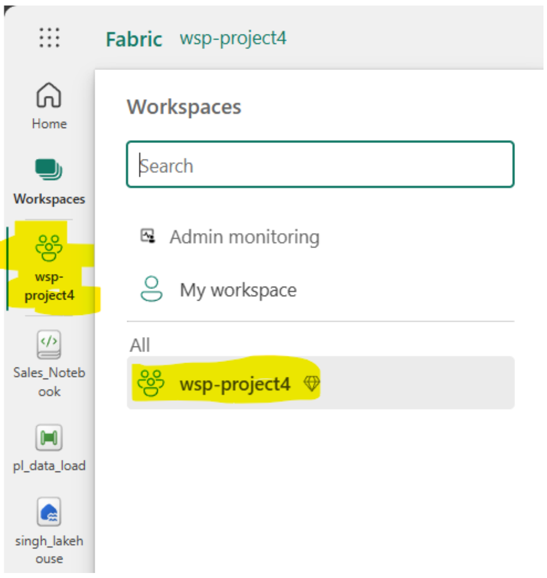
Pipeline Setup

The below pipeline is created to execute all the transformations and load the data.

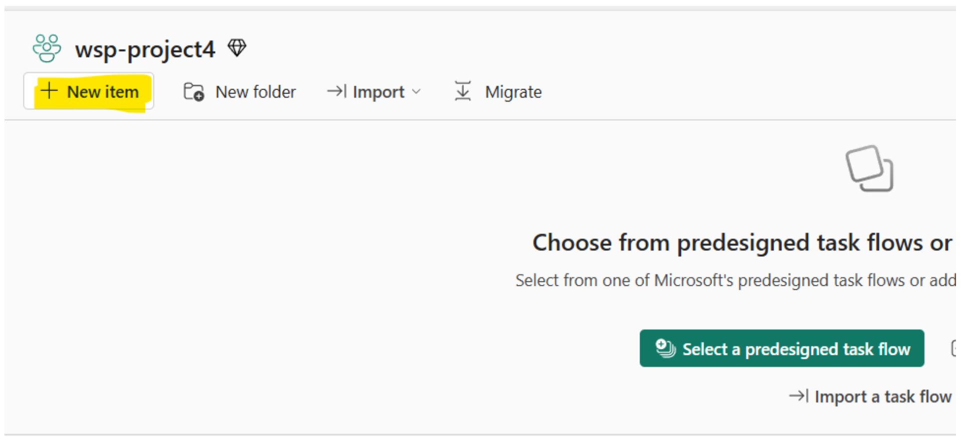


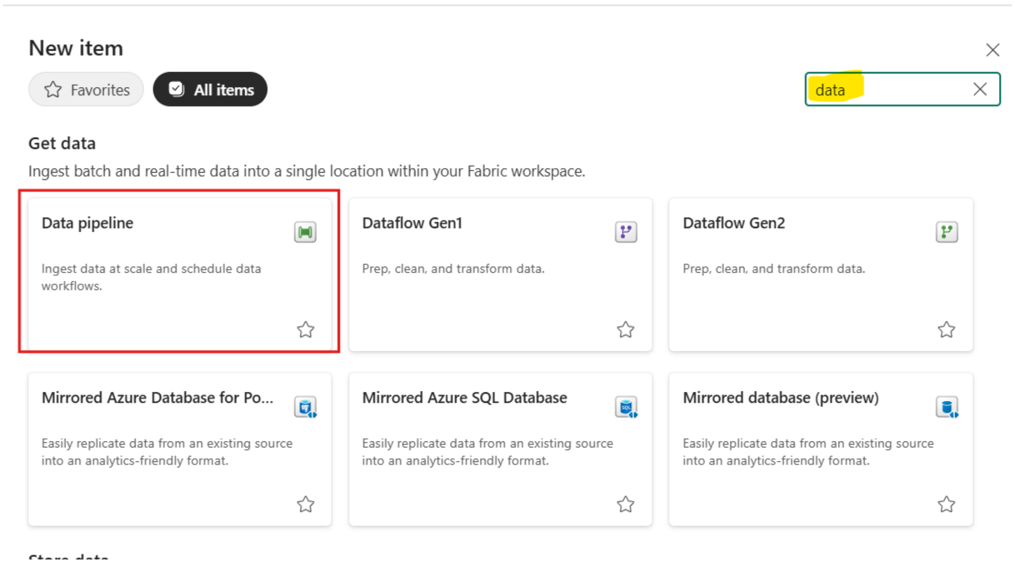
➢ Steps To Create the data transformation and load Pipeline

Step 1: Go to Fabric App -> create new or select workspace.



Step 2: in Workspace, click on new option and search and select the data pipeline. Give the Pipeline a name and click on create.

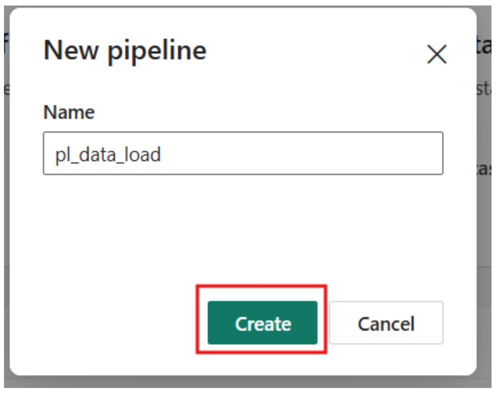

wsp-project4
New item
New folder
Import
Migrate
Choose from predesigned task flows or
Select from one of Microsoft's predesigned task flows or add
Select a predesigned task flow
Import a task flow



Get data

Ingest batch and real-time data into a single location within your Fabric workspace.

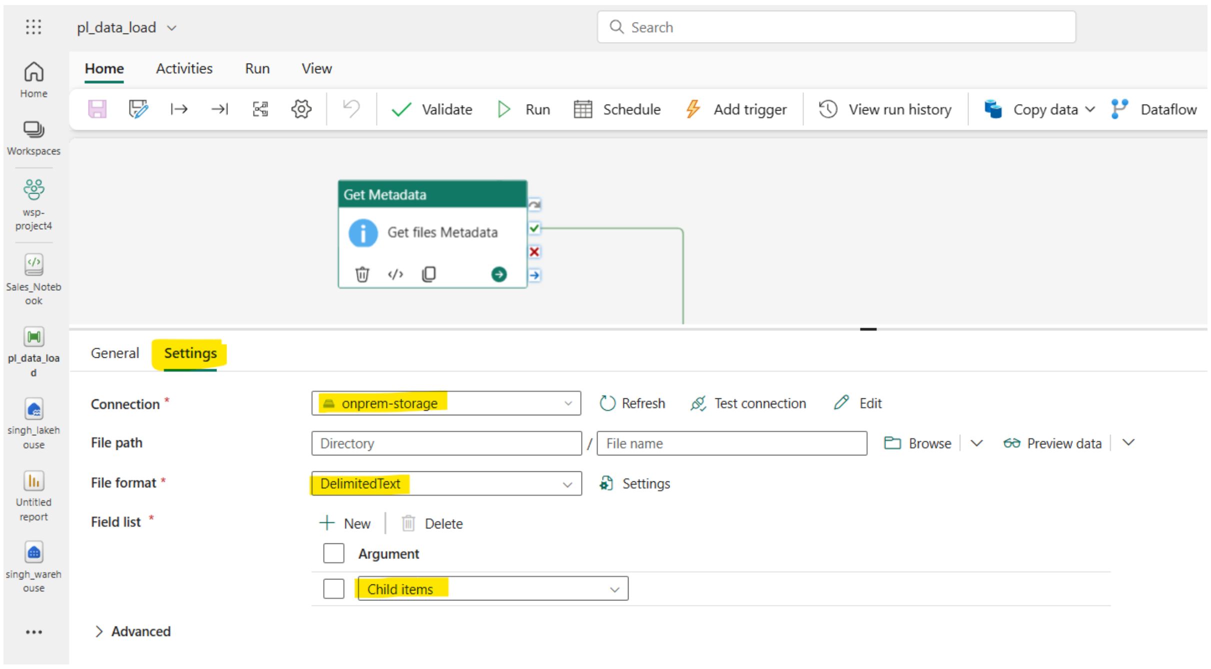
Data pipeline Ingest data at scale and schedule data workflows.	Dataflow Gen1 Prep, clean, and transform data.	Dataflow Gen2 Prep, clean, and transform data.
Mirrored Azure Database for Po... Easily replicate data from an existing source into an analytics-friendly format.	Mirrored Azure SQL Database Easily replicate data from an existing source into an analytics-friendly format.	Mirrored database (preview) Easily replicate data from an existing source into an analytics-friendly format.



Name: pl_data_load

Step 3: Add a get metadata activity -> go to settings tab -> select on-prem-storage (gateway connection).

- Select file format as DelimitedText.
- Select field list argument as child Items



Connection * onpremstorage

File path Directory / File name

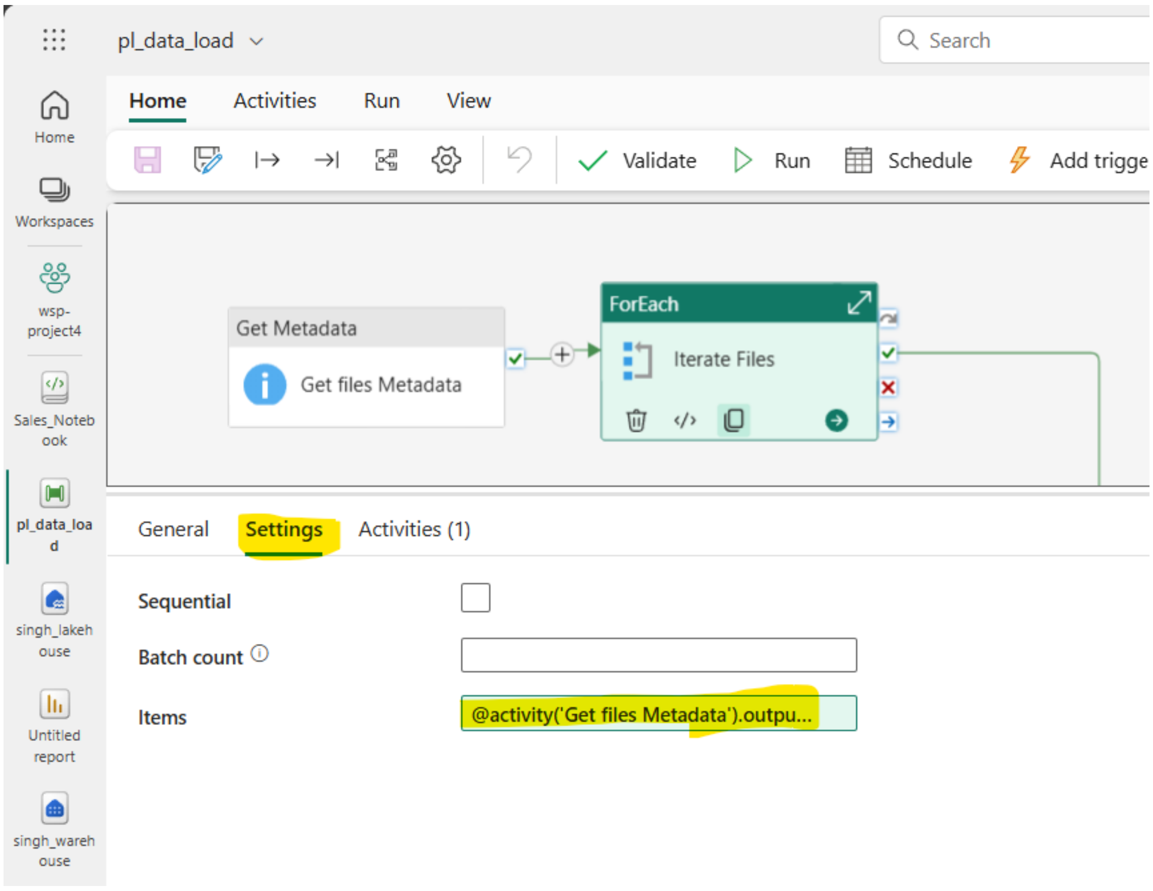
File format * DelimitedText

Field list *

Argument
Child items

Step 4: Add a get foreach activity-> connect foreach and get metadata with on success point -> go to settings tab of foreach -> enter below expression in the Items field.

- Expression Used:
  @activity('Get files Metadata').output.childItems

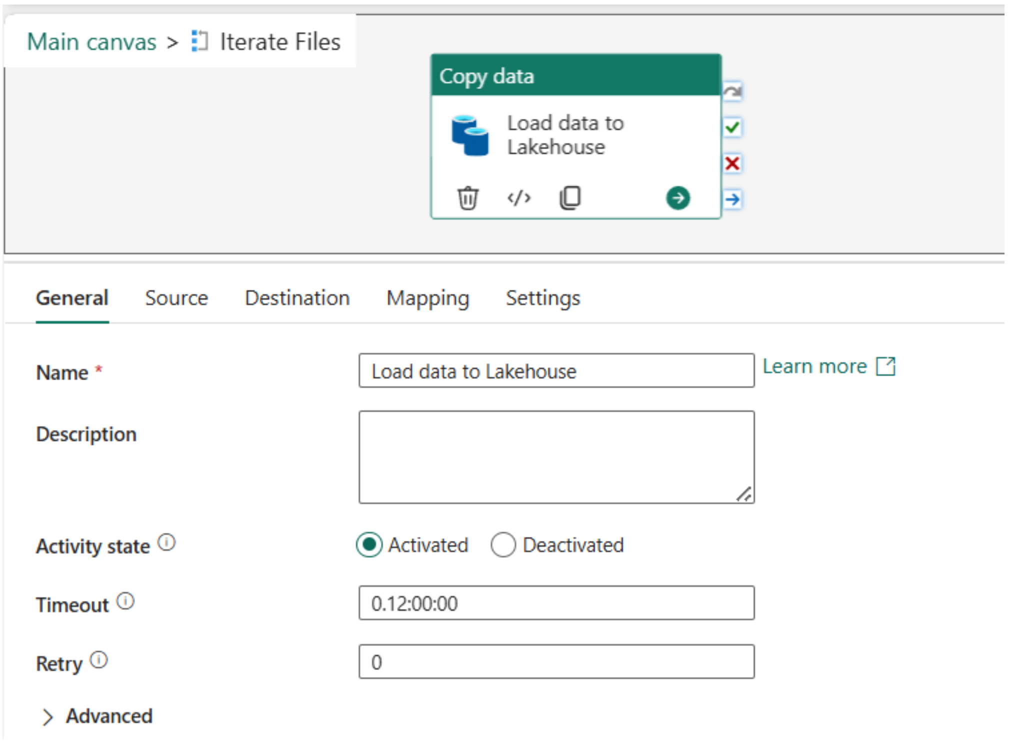


Sequential ☐

Batch count

Items @activity('Get files Metadata').outpu...

Step 5: Click on pencil to add activity in foreach -> add a copy data.



Name * Load data to Lakehouse

Description

Activity state [✓ Activated] Deactivated

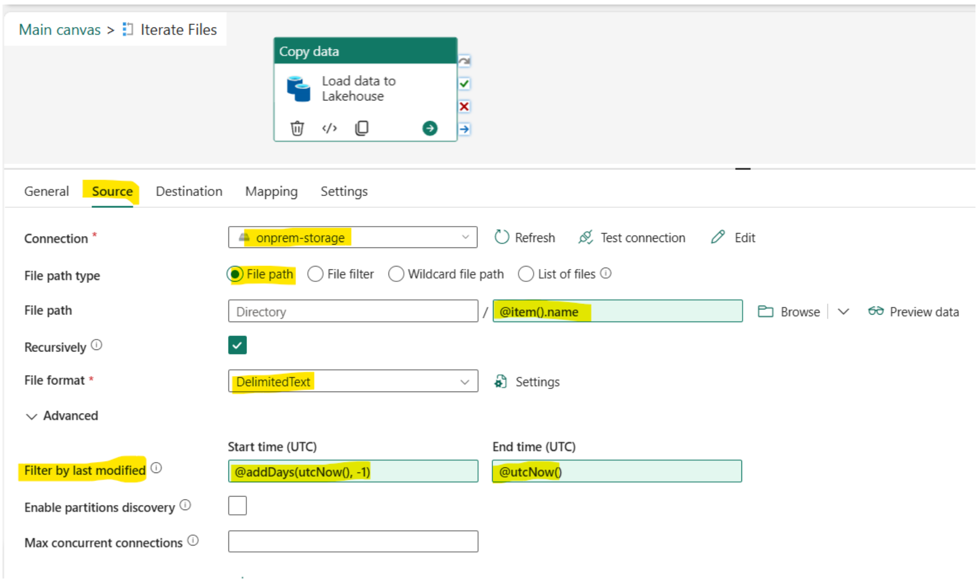
Timeout 0.12:00:00

Retry 0

> Advanced

Step 6: Go to Source tab of Copy data activity -> select connect as on-prem-storage (gateway connection). -> select file path type option as file path -> select file format as delimited text.

- In the advance option, give the below date expressions so it will only process the last 24 hours modified files.
  For Start time: **@addDays(utcNow(), -1)**
  For End time: **@utcNow()**
- Expression used to select the file_name data set properties:
  @item().name


Main canvas > Iterate Files

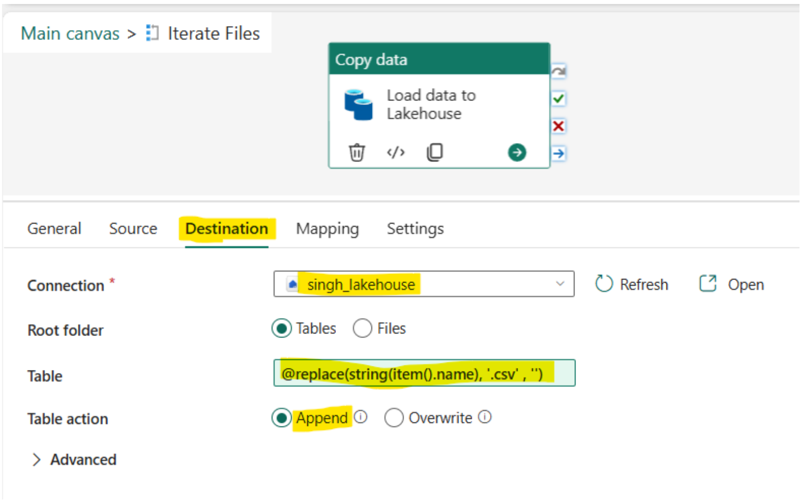
Copy data — Load data to Lakehouse

General | Source | Destination | Mapping | Settings

Connection * : onprem-storage (Refresh, Test connection, Edit)
File path type: [✓ File path] File filter, Wildcard file path, List of files
File path: Directory / @item().name (Browse, Preview data)
Recursively: ✓
File format *: DelimitedText (Settings)
Advanced
Filter by last modified: Start time (UTC) @addDays(utcNow(), -1); End time (UTC) @utcNow()
Enable partitions discovery: ☐
Max concurrent connections:

Step 7: Next, go to Destination tab of Copy data activity -> selec connection type as Lakehouse -> select root folder as Tables.

- Give the below expression for Table:
  @replace(string(item().name), '.csv' , '')


Main canvas > Iterate Files

Copy data — Load data to Lakehouse

General | Source | Destination | Mapping | Settings

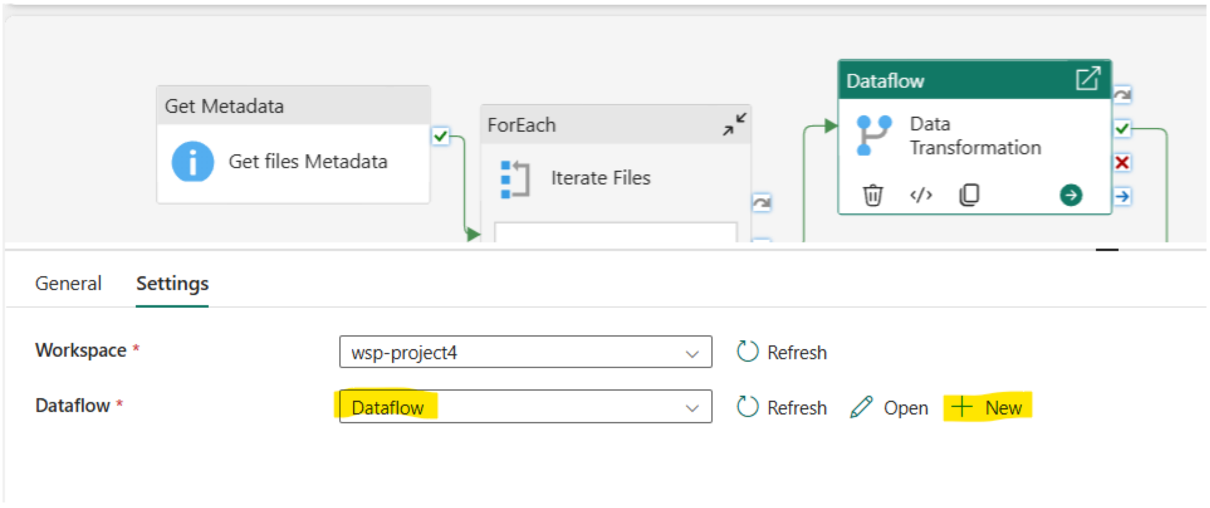
Connection * : singh_lakehouse (Refresh, Open)
Root folder: [✓ Tables] Files
Table: @replace(string(item().name), '.csv' , '')
Table action: [✓ Append] Overwrite
Advanced

Data Flow Setup

✓ **Steps To Create the data transformation and load Pipeline**

Step 1: Add a new Dataflow activity and connect it with on success point of Foreach.

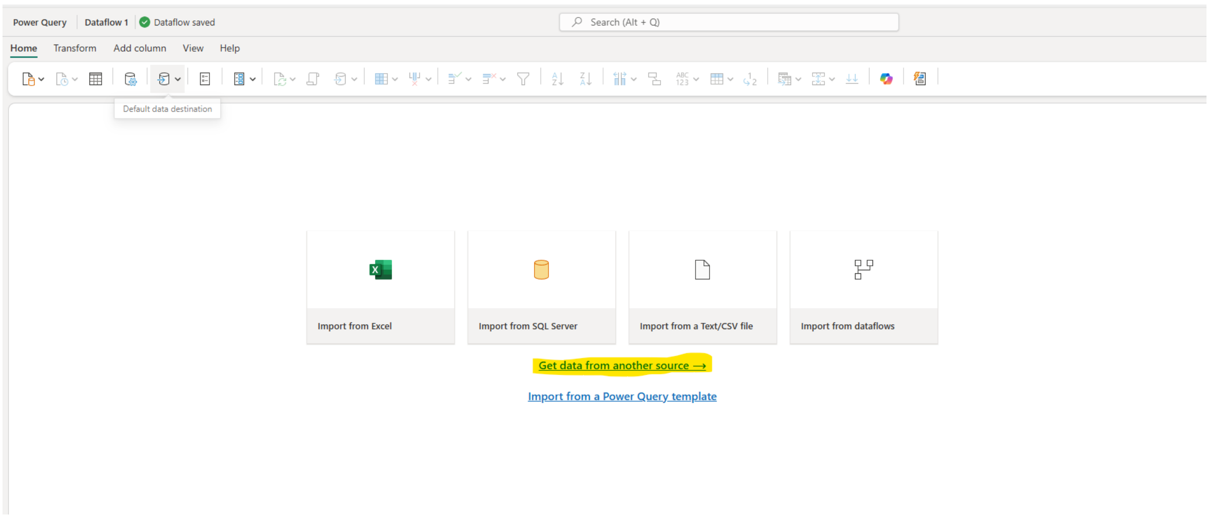
– In the settings tab, create new dataflow with new button.



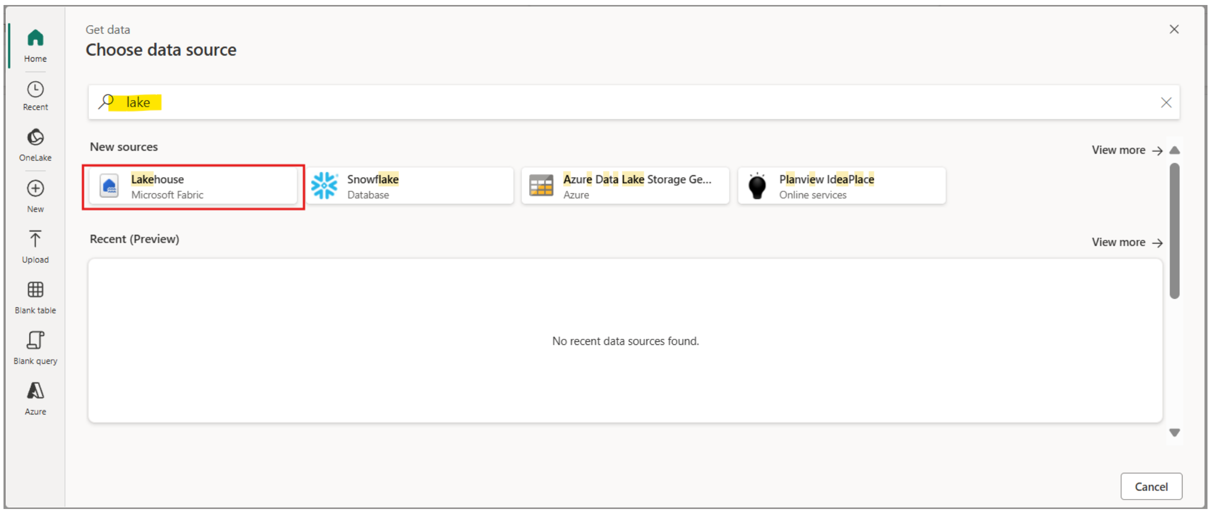
Workspace * wsp-project4 Refresh

Dataflow * Dataflow Refresh Open New

Step 2: open the Data flow -> click on gata data from another source.



Step 3: Select lakehouse -> select connection as lakehouse -> click on next.



Get data

Choose data source

lake

New sources

View more →

Lakehouse
Microsoft Fabric

Snowflake
Database

Azure Data Lake Storage Ge...
Azure

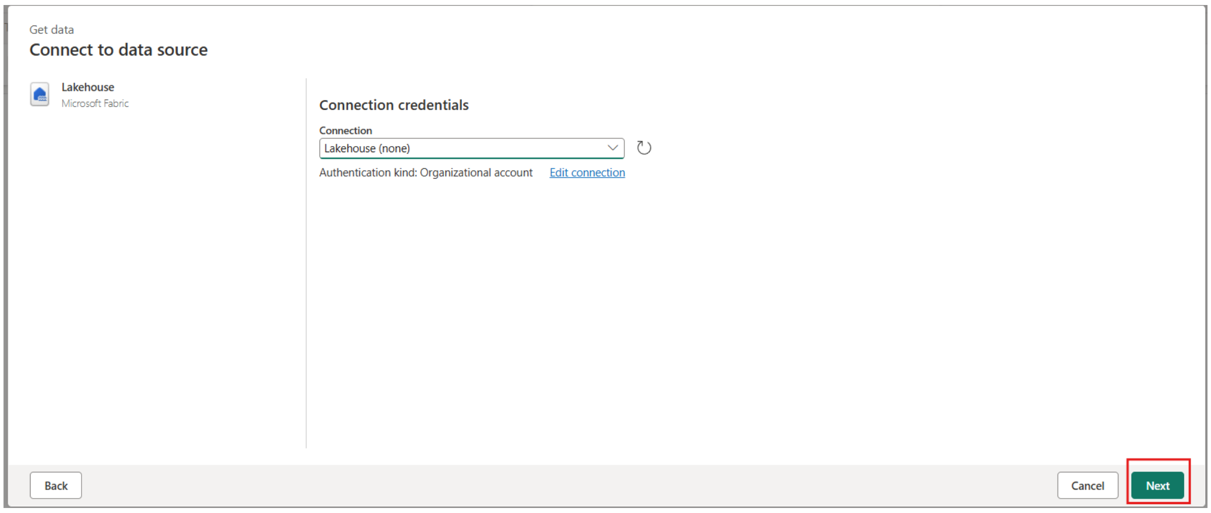
Planview IdeaPlace
Online services

Recent (Preview)

View more →

No recent data sources found.

Cancel



Get data

Connect to data source

Lakehouse
Microsoft Fabric

Connection credentials

Connection: Lakehouse (none)

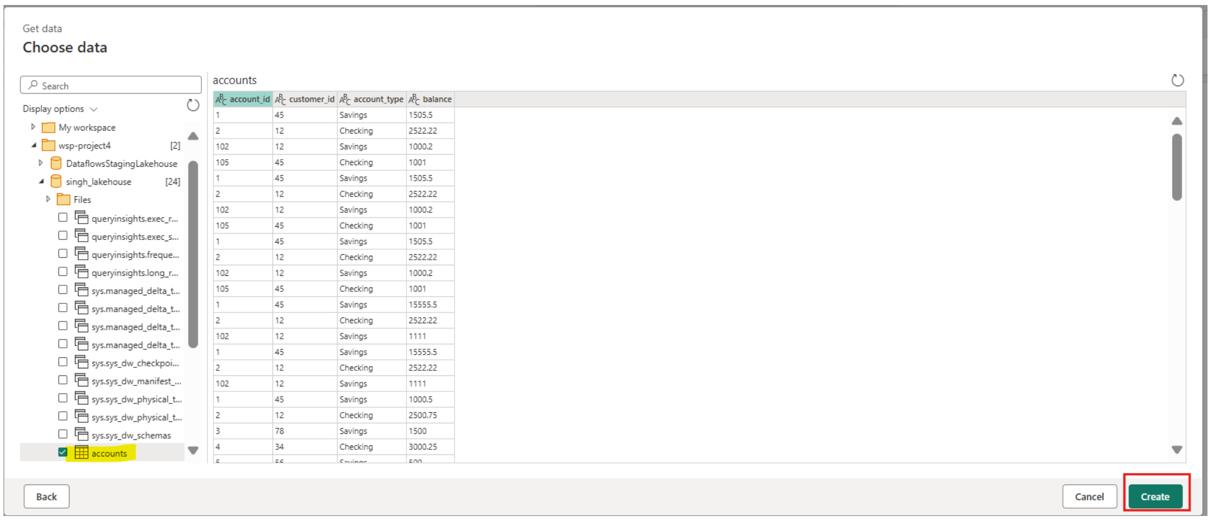
Authentication kind: Organizational account Edit connection

Back

Cancel

Next

Step 4: Select source table from lakehouse -> click on create.



Get data

Choose data

Search

Display options

- My workspace
- wsp-project4 [2]
  - DataflowsStagingLakehouse
  - singh_lakehouse [24]
    - Files
    - ☐ queryinsights.exec_r...
    - ☐ queryinsights.exec_s...
    - ☐ queryinsights.freque...
    - ☐ queryinsights.long_r...
    - ☐ sys.managed_delta_t...
    - ☐ sys.managed_delta_t...
    - ☐ sys.managed_delta_t...
    - ☐ sys.managed_delta_t...
    - ☐ sys.sys_dw_checkpoi...
    - ☐ sys.sys_dw_manifest_...
    - ☐ sys.sys_dw_physical_t...
    - ☐ sys.sys_dw_physical_t...
    - ☐ sys.sys_dw_schemas
    - ✓ accounts

accounts

account_id	customer_id	account_type	balance
1	45	Savings	1505.5
2	12	Checking	2522.22
102	12	Savings	1000.2
105	45	Checking	1001
1	45	Savings	1505.5
2	12	Checking	2522.22
102	12	Savings	1000.2
105	45	Checking	1001
1	45	Savings	1505.5
2	12	Checking	2522.22
102	12	Savings	1000.2
105	45	Checking	1001
1	45	Savings	15555.5
2	12	Checking	2522.22
102	12	Savings	1111
1	45	Savings	15555.5
2	12	Checking	2522.22
102	12	Savings	1111
1	45	Savings	1000.5
2	12	Checking	2500.75
3	78	Savings	1500
4	34	Checking	3000.25

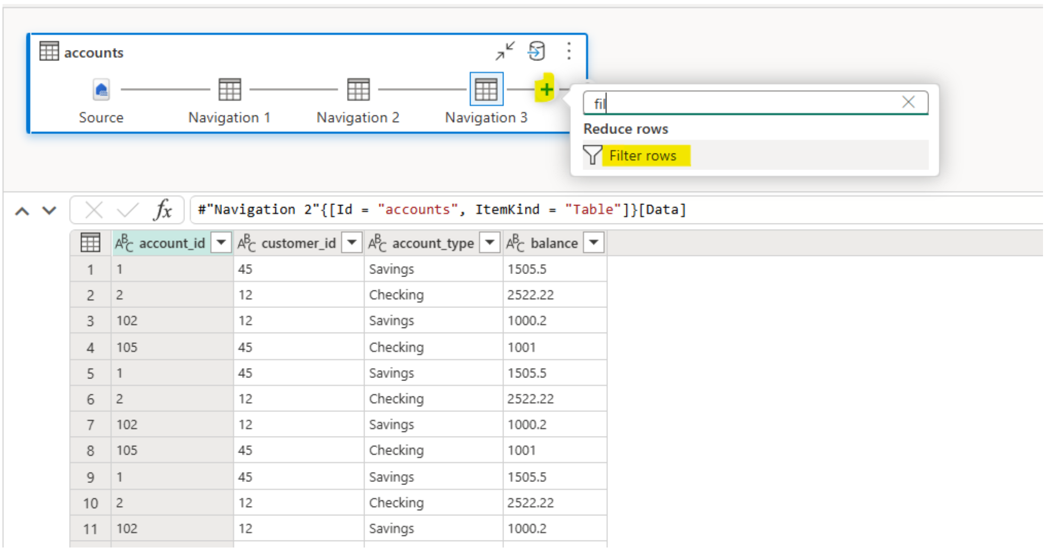
Back

Cancel

Create

Step 5: Now the source table data is loaded, click on diagrammatic view at the bottom.

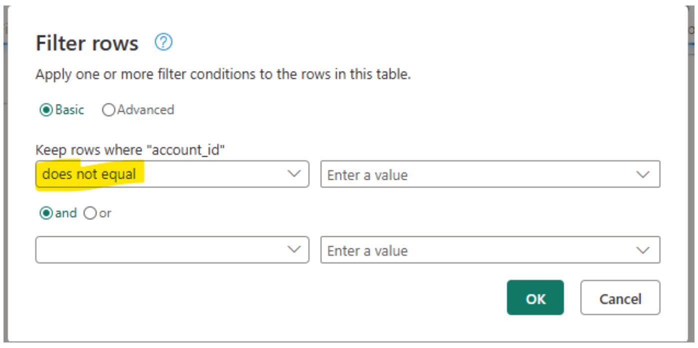
- In the view, click on + and add **filter** transformation.



```
#"Navigation 2"{[Id = "accounts", ItemKind = "Table"]}[Data]
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	1	45	Savings	1505.5
6	2	12	Checking	2522.22
7	102	12	Savings	1000.2
8	105	45	Checking	1001
9	1	45	Savings	1505.5
10	2	12	Checking	2522.22
11	102	12	Savings	1000.2

- For first field, select **does not equal** and for second field, leave it empty as we are comparing the id with null -> click on ok.


Filter rows

Apply one or more filter conditions to the rows in this table.

[✓ Basic] Advanced

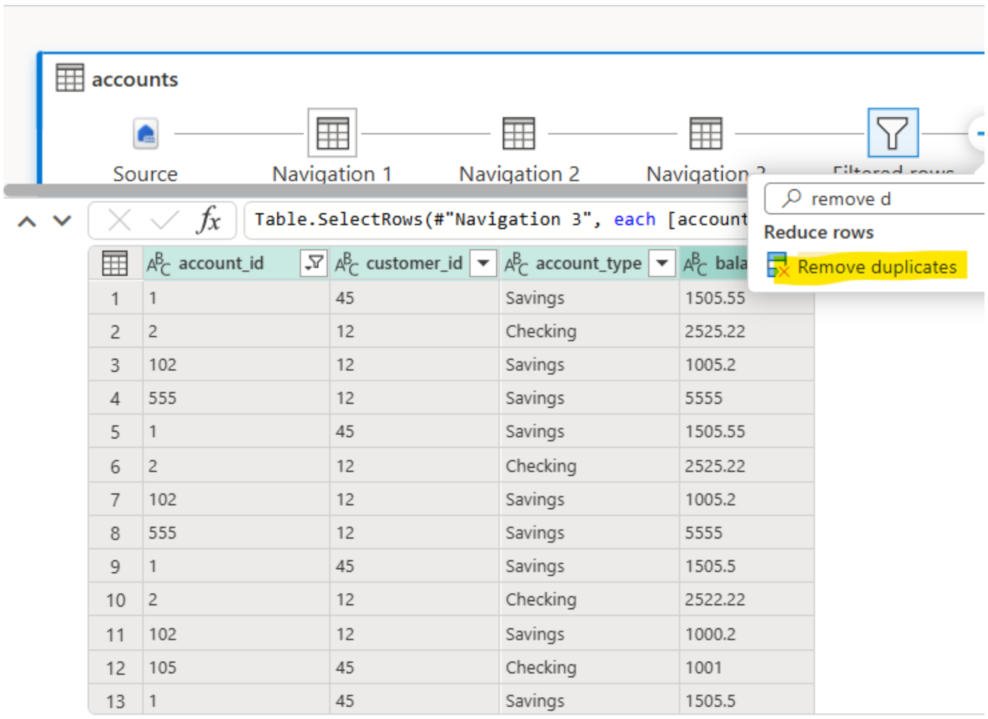
Keep rows where "account_id"

does not equal | Enter a value

[✓ and] or

| Enter a value

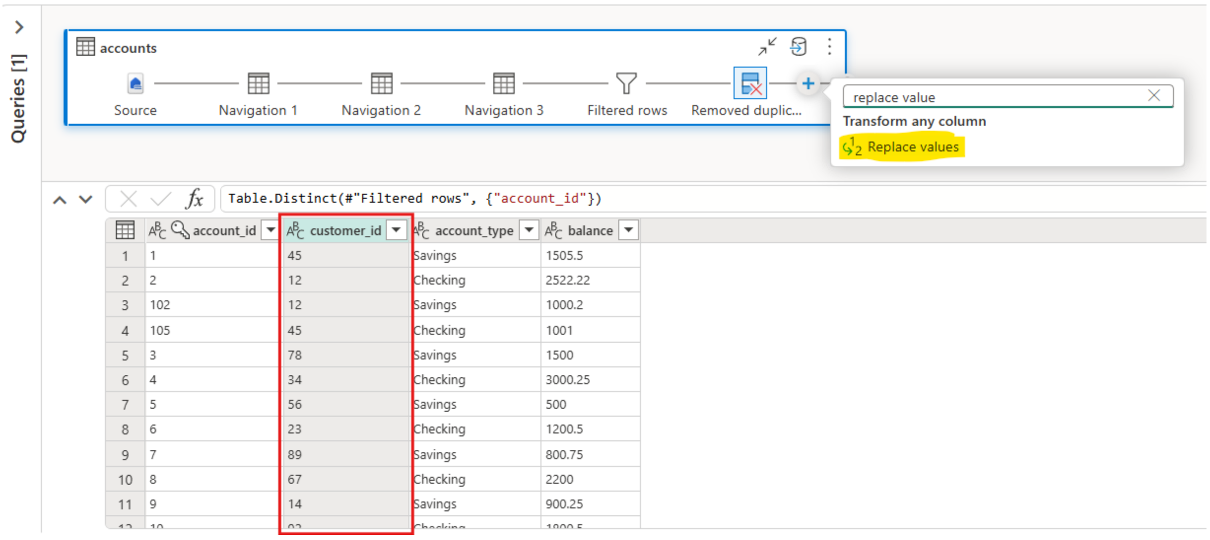
Step 6: Next, select all the columns, click on + and add **Remove Duplicates** transformation.



```
Table.SelectRows(#"Navigation 3", each [account
```

	account_id	customer_id	account_type	bala
1	1	45	Savings	1505.55
2	2	12	Checking	2525.22
3	102	12	Savings	1005.2
4	555	12	Savings	5555
5	1	45	Savings	1505.55
6	2	12	Checking	2525.22
7	102	12	Savings	1005.2
8	555	12	Savings	5555
9	1	45	Savings	1505.5
10	2	12	Checking	2522.22
11	102	12	Savings	1000.2
12	105	45	Checking	1001
13	1	45	Savings	1505.5

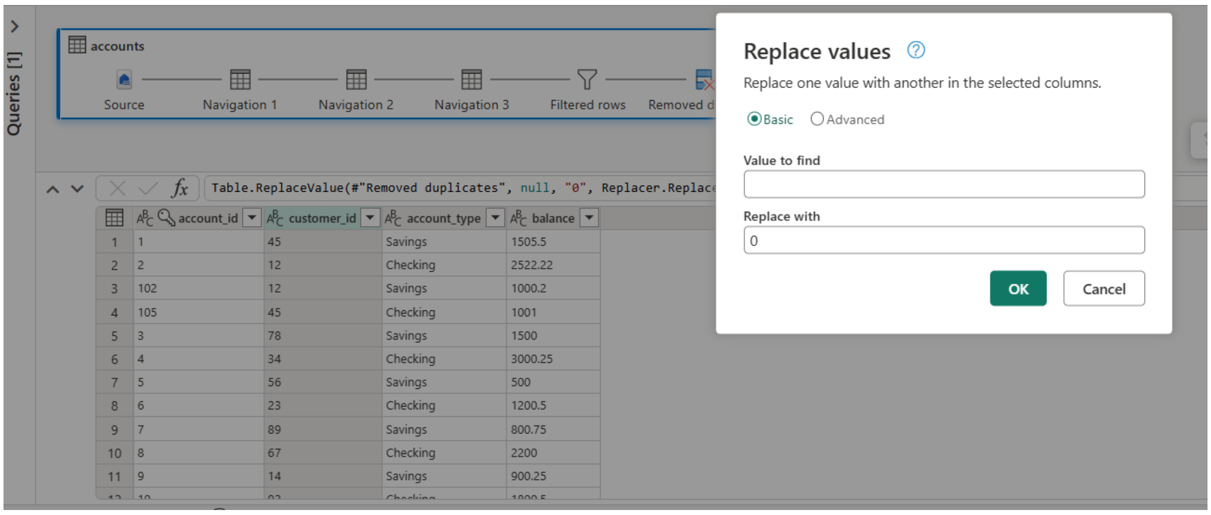
Step 7: Next, select one column, click on + and add **Remove Duplicates** transformation.



```
Table.Distinct(#"Filtered rows", {"account_id"})
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25

- For first field, leave it empty as we are checking the value with null value and for second field, enter replacing value (0)-> click on ok.
- Do it for all the column one by one with appropriate replacing value.


Queries [1]

accounts: Source → Navigation 1 → Navigation 2 → Navigation 3 → Filtered rows → Removed d

```
Table.ReplaceValue(#"Removed duplicates", null, "0", Replacer.Replace
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25

Replace values

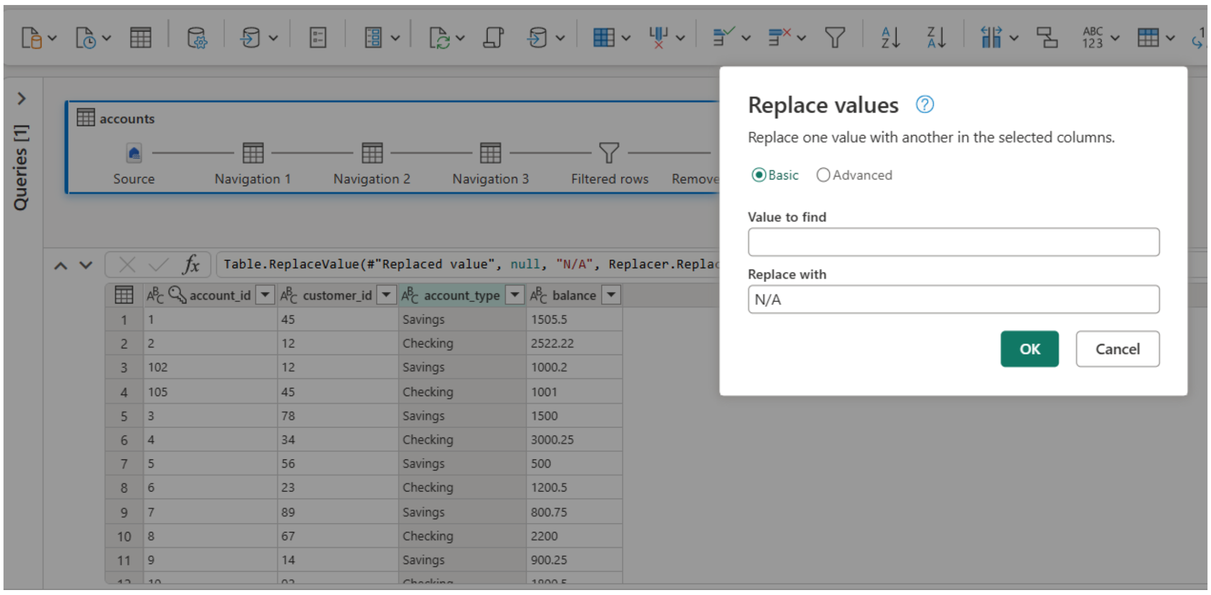
Replace one value with another in the selected columns.

[✓ Basic] Advanced

Value to find:

Replace with: 0

OK Cancel


Queries [1]

accounts: Source → Navigation 1 → Navigation 2 → Navigation 3 → Filtered rows → Remove

```
Table.ReplaceValue(#"Replaced value", null, "N/A", Replacer.Replac
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25

Replace values

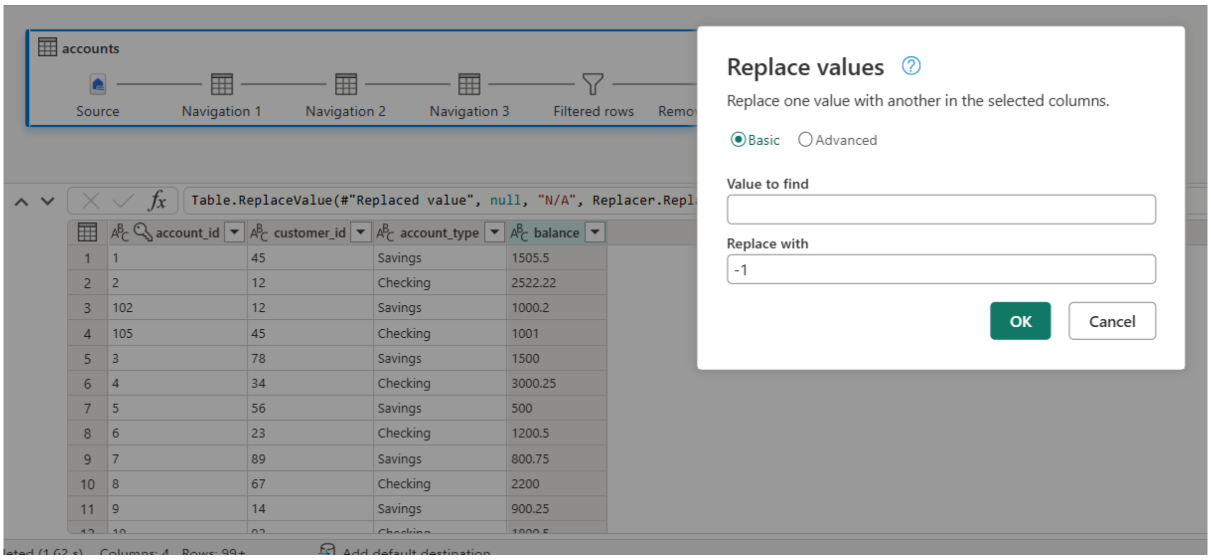
Replace one value with another in the selected columns.

[✓ Basic] Advanced

Value to find:

Replace with: N/A

OK Cancel


accounts: Source → Navigation 1 → Navigation 2 → Navigation 3 → Filtered rows → Remo

```
Table.ReplaceValue(#"Replaced value", null, "N/A", Replacer.Repl
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25

Replace values

Replace one value with another in the selected columns.

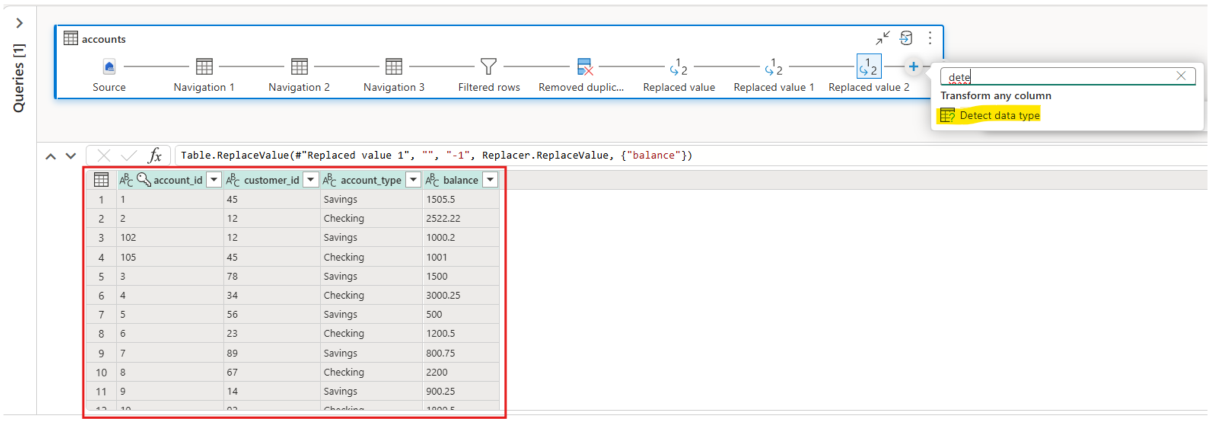
[✓ Basic] Advanced

Value to find:

Replace with: -1

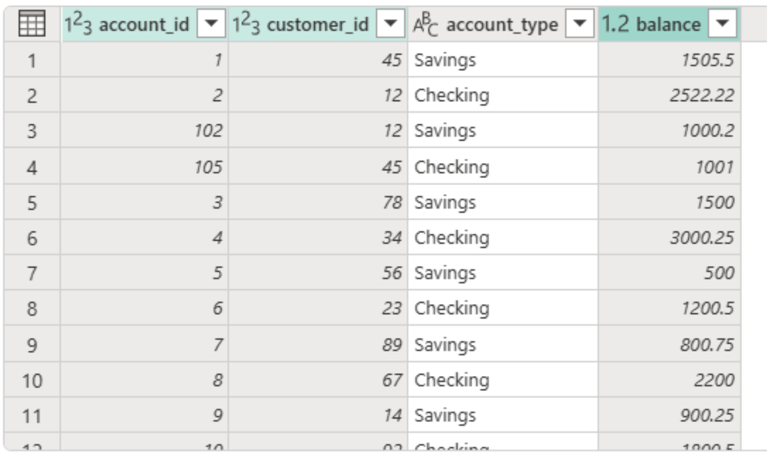
OK Cancel

Step 8: Next, select all the columns, click on + and add **Detect Data Types** transformation.



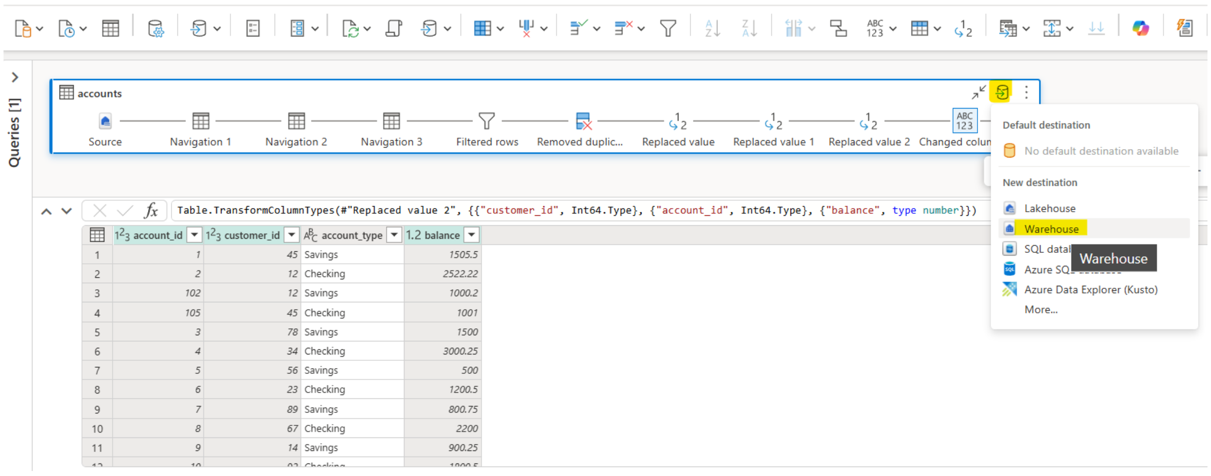
```
Table.ReplaceValue(#"Replaced value 1", "", "-1", Replacer.ReplaceValue, {"balance"})
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25



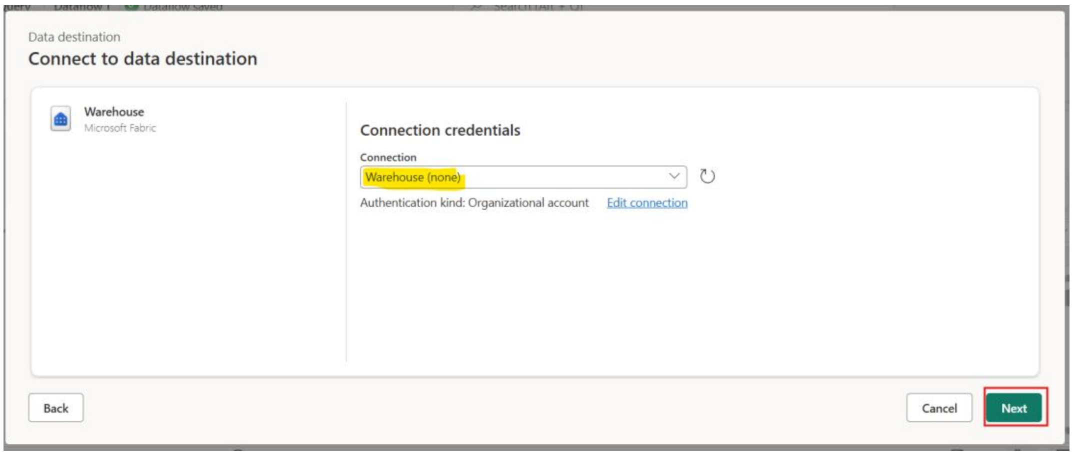
	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25

Step 9: click on destination option, select warehouse as destination -> click on Next.



```
Table.TransformColumnTypes(#"Replaced value 2", {{"customer_id", Int64.Type}, {"account_id", Int64.Type}, {"balance", type number}})
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25



Data destination

Connect to data destination

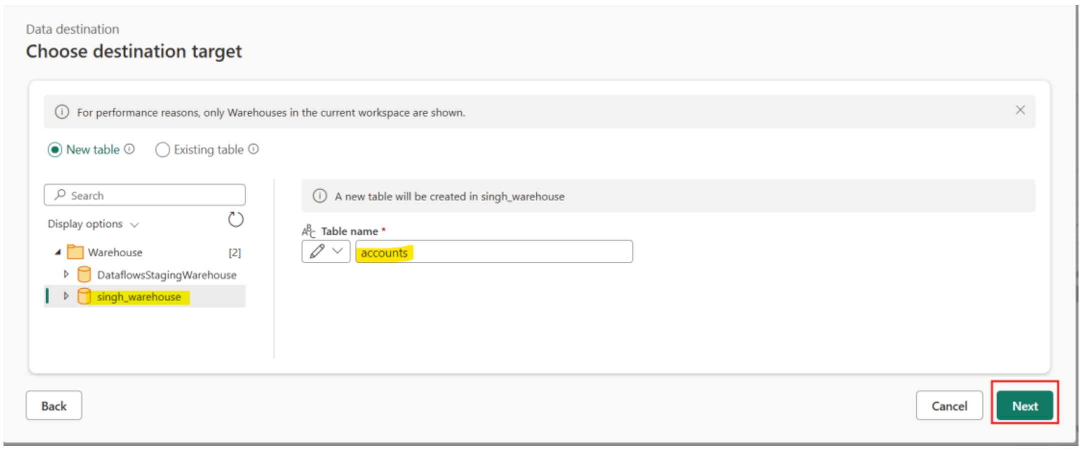
Warehouse
Microsoft Fabric

Connection credentials

Connection: Warehouse (none)

Authentication kind: Organizational account   Edit connection

Step 10: select the warehouse and enter the table name to create in warehouse -> click on Next.



Data destination

Choose destination target

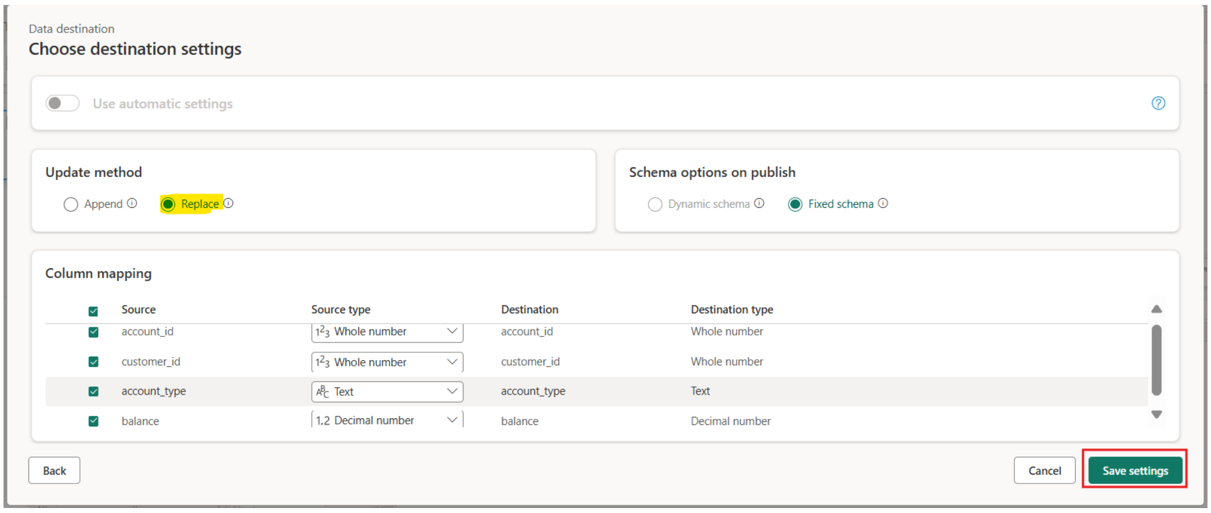
For performance reasons, only Warehouses in the current workspace are shown.

[✓ New table]   Existing table

A new table will be created in singh_warehouse

Table name *: accounts

Step 11: select replace as update method and check the source type -> click on Save settings



Data destination

Choose destination settings

Use automatic settings

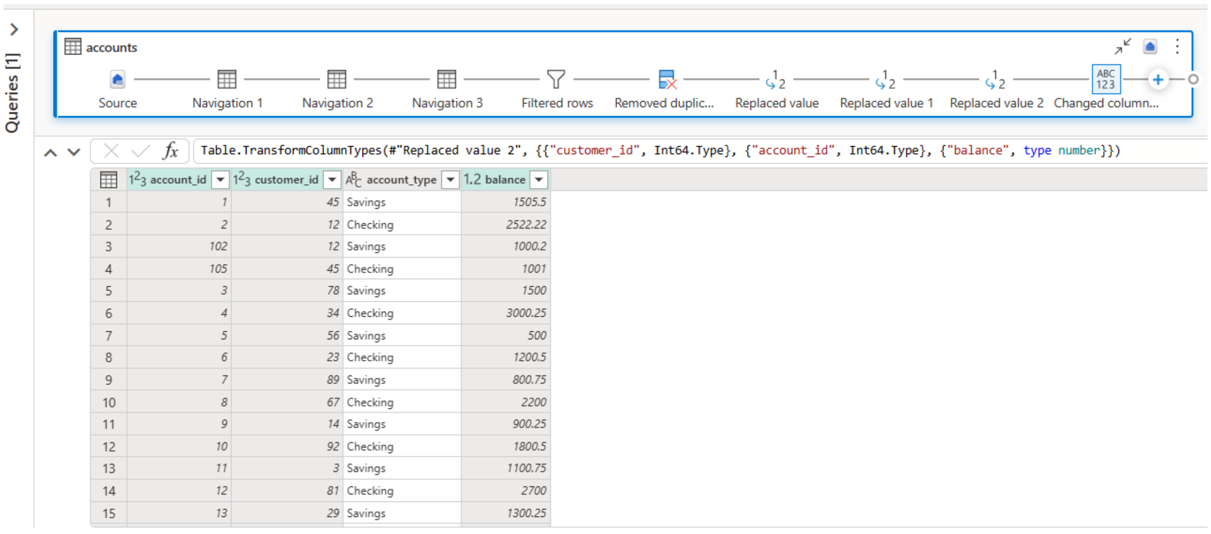
Update method: Append   [✓ Replace]

Schema options on publish: Dynamic schema   [✓ Fixed schema]

Column mapping

	Source	Source type	Destination	Destination type
✓	account_id	Whole number	account_id	Whole number
✓	customer_id	Whole number	customer_id	Whole number
✓	account_type	Text	account_type	Text
✓	balance	Decimal number	balance	Decimal number

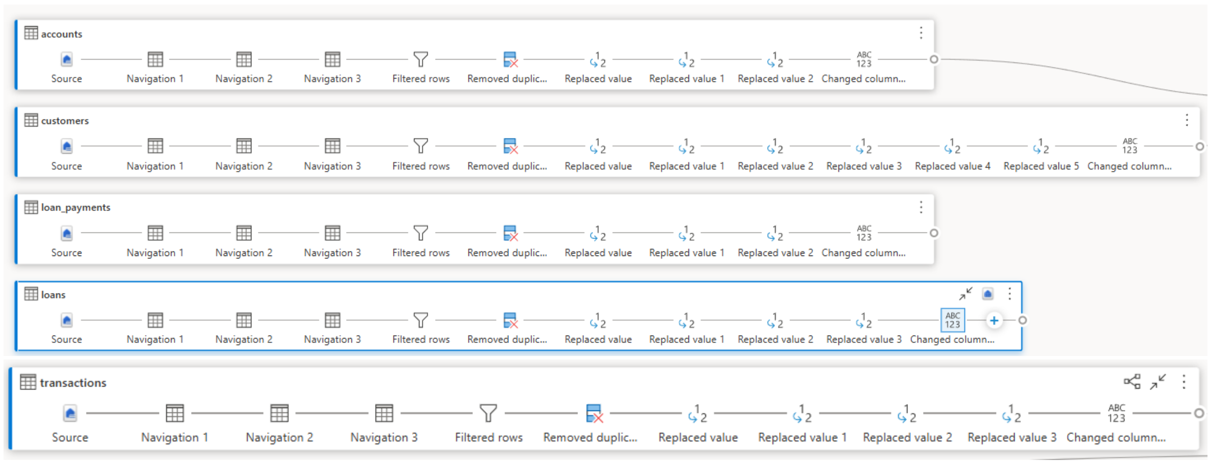
Step 12: Finally, review the data.



```
Table.TransformColumnTypes(#"Replaced value 2", {{"customer_id", Int64.Type}, {"account_id", Int64.Type}, {"balance", type number}})
```

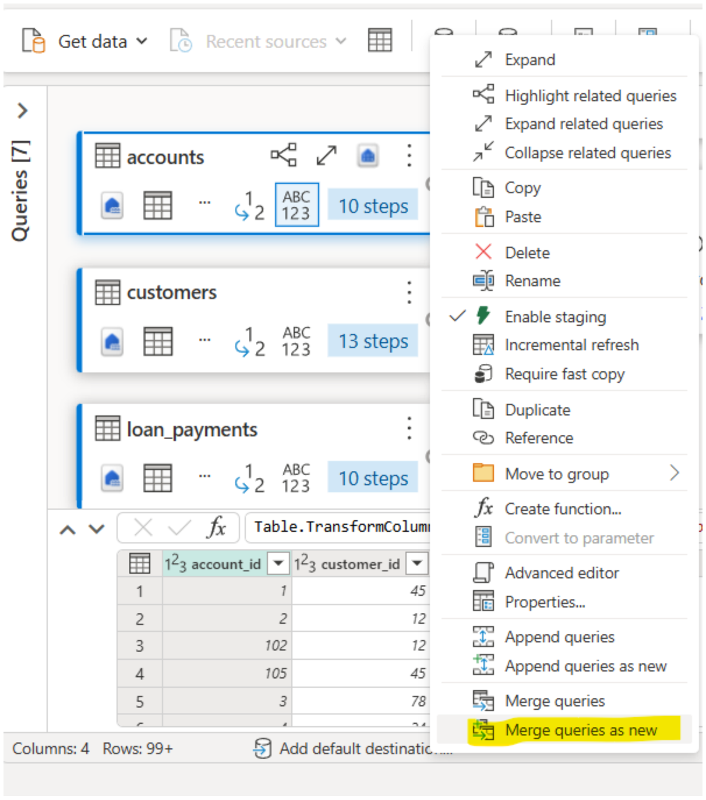
	account_id	customer_id	account_type	balance
1	1	45	Savings	1505.5
2	2	12	Checking	2522.22
3	102	12	Savings	1000.2
4	105	45	Checking	1001
5	3	78	Savings	1500
6	4	34	Checking	3000.25
7	5	56	Savings	500
8	6	23	Checking	1200.5
9	7	89	Savings	800.75
10	8	67	Checking	2200
11	9	14	Savings	900.25
12	10	92	Checking	1800.5
13	11	3	Savings	1100.75
14	12	81	Checking	2700
15	13	29	Savings	1300.25

**** Similarly, all the steps have to be performed for all the remaining source tables.**



✓ **Steps To perform the join on transformed tables.**

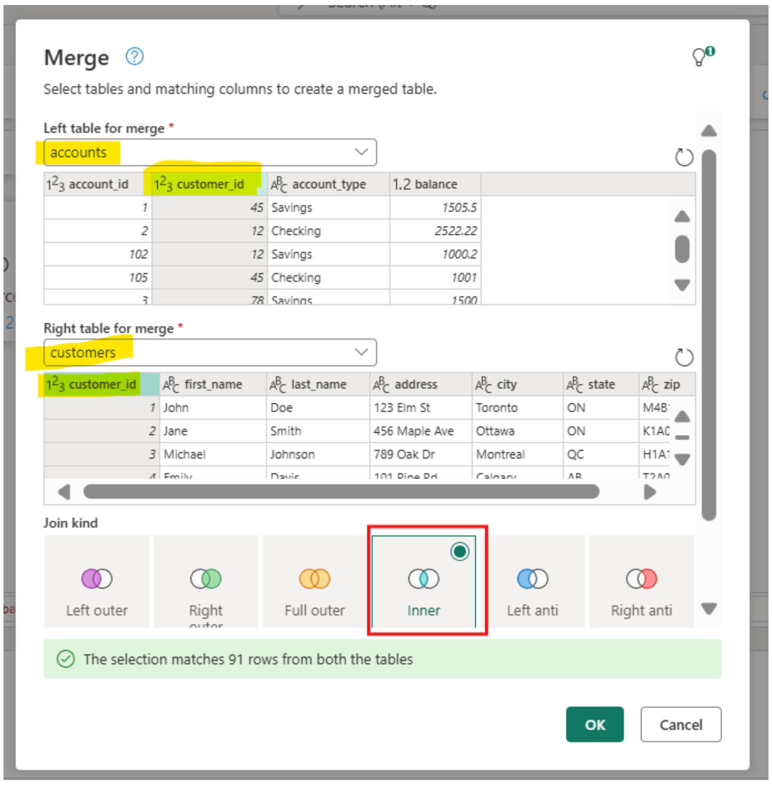
Step 1: click on three dots source table, select the **Merge queries as new option.**



	account_id	customer_id
1	1	45
2	2	12
3	102	12
4	105	45
5	3	78

Step 2: Next, select the left table and right table and the matching Ids in both columns.

– Select the inner join as join type.


Merge

Select tables and matching columns to create a merged table.

Left table for merge *: accounts

account_id	customer_id	account_type	balance
1	45	Savings	1505.5
2	12	Checking	2522.22
102	12	Savings	1000.2
105	45	Checking	1001
3	78	Savings	1500

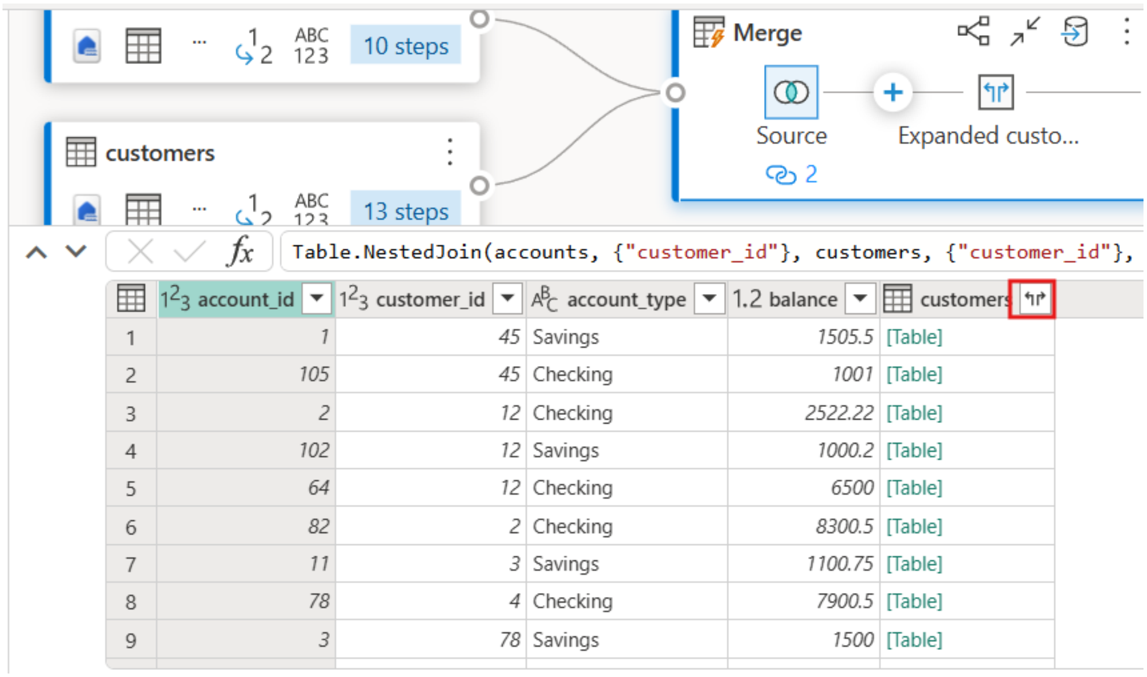
Right table for merge *: customers

customer_id	first_name	last_name	address	city	state	zip
1	John	Doe	123 Elm St	Toronto	ON	M4B
2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A
3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A
4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A

Join kind: Left outer, Right outer, Full outer, [✓ Inner], Left anti, Right anti

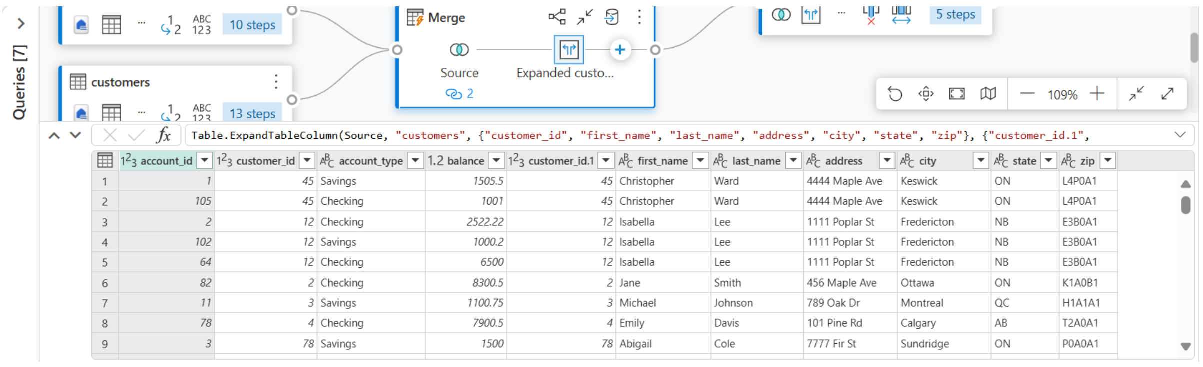
The selection matches 91 rows from both the tables

Step 2: Next, click on expand option on the joined table to display all the columns in the table.



```
Table.NestedJoin(accounts, {"customer_id"}, customers, {"customer_id"},
```

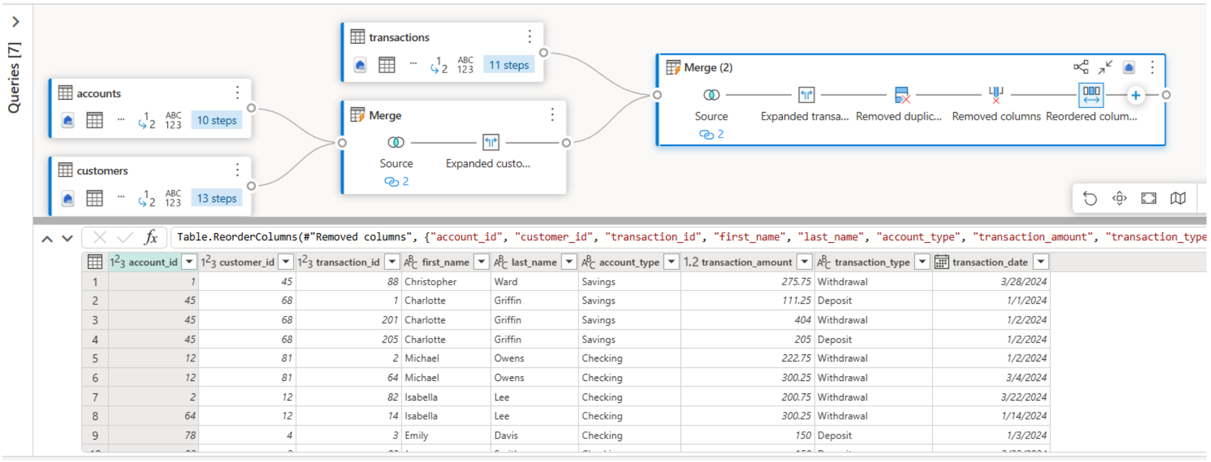
	account_id	customer_id	account_type	balance	customers
1	1	45	Savings	1505.5	[Table]
2	105	45	Checking	1001	[Table]
3	2	12	Checking	2522.22	[Table]
4	102	12	Savings	1000.2	[Table]
5	64	12	Checking	6500	[Table]
6	82	2	Checking	8300.5	[Table]
7	11	3	Savings	1100.75	[Table]
8	78	4	Checking	7900.5	[Table]
9	3	78	Savings	1500	[Table]



```
Table.ExpandTableColumn(Source, "customers", {"customer_id", "first_name", "last_name", "address", "city", "state", "zip"}, {"customer_id.1",
```

	account_id	customer_id	account_type	balance	customer_id.1	first_name	last_name	address	city	state	zip
1	1	45	Savings	1505.5	45	Christopher	Ward	4444 Maple Ave	Keswick	ON	L4P0A1
2	105	45	Checking	1001	45	Christopher	Ward	4444 Maple Ave	Keswick	ON	L4P0A1
3	2	12	Checking	2522.22	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
4	102	12	Savings	1000.2	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
5	64	12	Checking	6500	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
6	82	2	Checking	8300.5	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
7	11	3	Savings	1100.75	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
8	78	4	Checking	7900.5	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
9	3	78	Savings	1500	78	Abigail	Cole	7777 Fir St	Sundridge	ON	P0A0A1

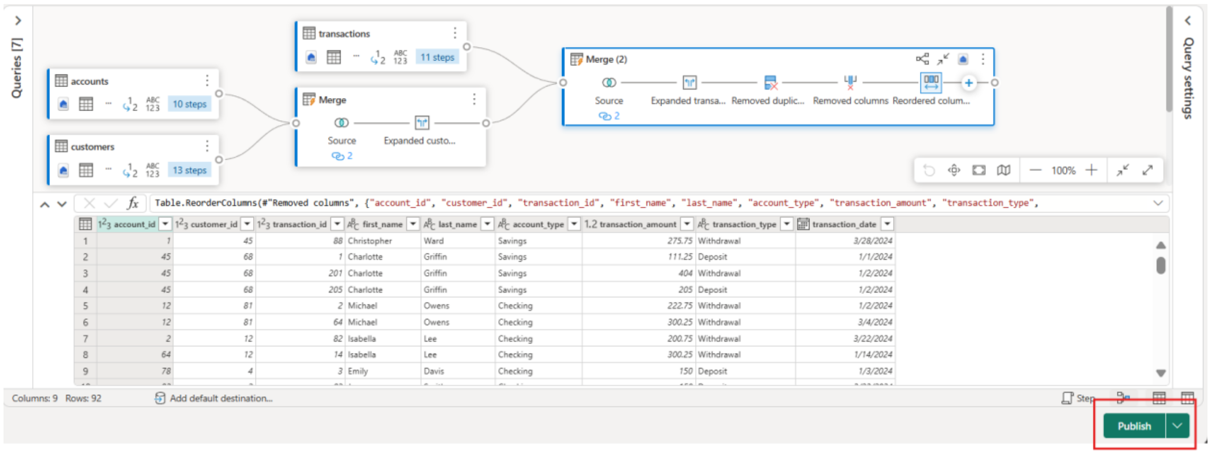
Step 3: Perform the same 2 steps to join the result table with another source table and save it in the warehouse.



```
Table.ReorderColumns(#"Removed columns", {"account_id", "customer_id", "transaction_id", "first_name", "last_name", "account_type", "transaction_amount", "transaction_type
```

	account_id	customer_id	transaction_id	first_name	last_name	account_type	transaction_amount	transaction_type	transaction_date
1	1	45	88	Christopher	Ward	Savings	275.75	Withdrawal	3/28/2024
2	45	68	1	Charlotte	Griffin	Savings	111.25	Deposit	1/1/2024
3	45	68	201	Charlotte	Griffin	Savings	404	Withdrawal	1/2/2024
4	45	68	205	Charlotte	Griffin	Savings	205	Deposit	1/2/2024
5	12	81	2	Michael	Owens	Checking	222.75	Withdrawal	1/2/2024
6	12	81	64	Michael	Owens	Checking	300.25	Withdrawal	3/4/2024
7	2	12	82	Isabella	Lee	Checking	200.75	Withdrawal	3/22/2024
8	64	12	14	Isabella	Lee	Checking	300.25	Withdrawal	1/14/2024
9	78	4	3	Emily	Davis	Checking	150	Deposit	1/3/2024

Now, click on publish button to save and run the data flow activity.

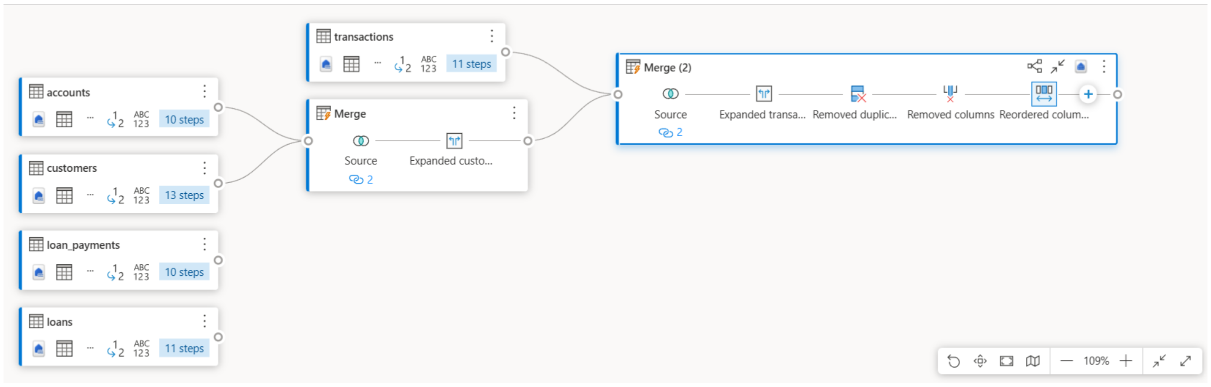


```
Table.ReorderColumns(#"Removed columns", {"account_id", "customer_id", "transaction_id", "first_name", "last_name", "account_type", "transaction_amount", "transaction_type",
```

	account_id	customer_id	transaction_id	first_name	last_name	account_type	transaction_amount	transaction_type	transaction_date
1	1	45	88	Christopher	Ward	Savings	275.75	Withdrawal	3/28/2024
2	45	68	1	Charlotte	Griffin	Savings	111.25	Deposit	1/1/2024
3	45	68	201	Charlotte	Griffin	Savings	404	Withdrawal	1/2/2024
4	45	68	205	Charlotte	Griffin	Savings	205	Deposit	1/2/2024
5	12	81	2	Michael	Owens	Checking	222.75	Withdrawal	1/2/2024
6	12	81	64	Michael	Owens	Checking	300.25	Withdrawal	3/4/2024
7	2	12	82	Isabella	Lee	Checking	200.75	Withdrawal	3/22/2024
8	64	12	14	Isabella	Lee	Checking	300.25	Withdrawal	1/14/2024
9	78	4	3	Emily	Davis	Checking	150	Deposit	1/3/2024

Columns: 9 Rows: 92 Add default destination...

Here, the below is the full dataflow transformations.

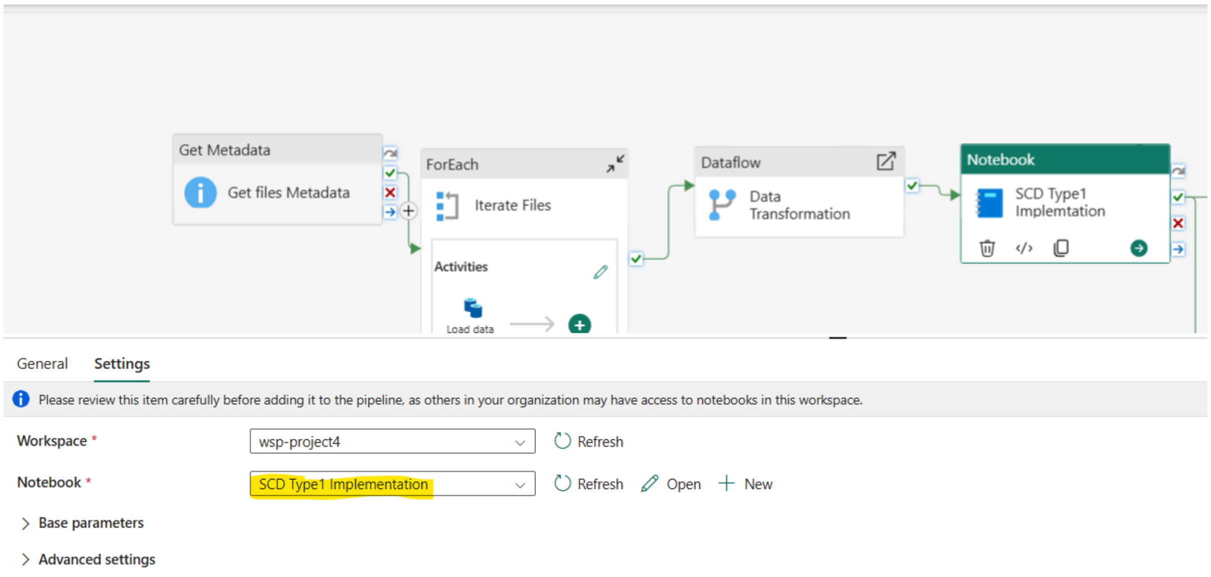


SCD Type1 Notebook Implementation

✓ **Steps To Create the SCD1 implementation Fabric Notebook.**

Step 1: Add a new Notebook activity and connect it with on success point of Dataflow activity.

– In the settings tab, create new Notebook with new button and select the same.



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

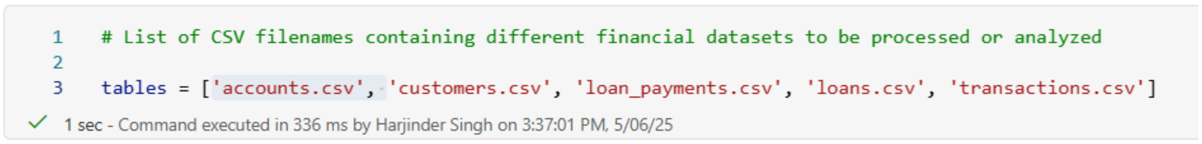
Workspace * wsp-project4 Refresh

Notebook * SCD Type1 Implementation Refresh Open New

Base parameters

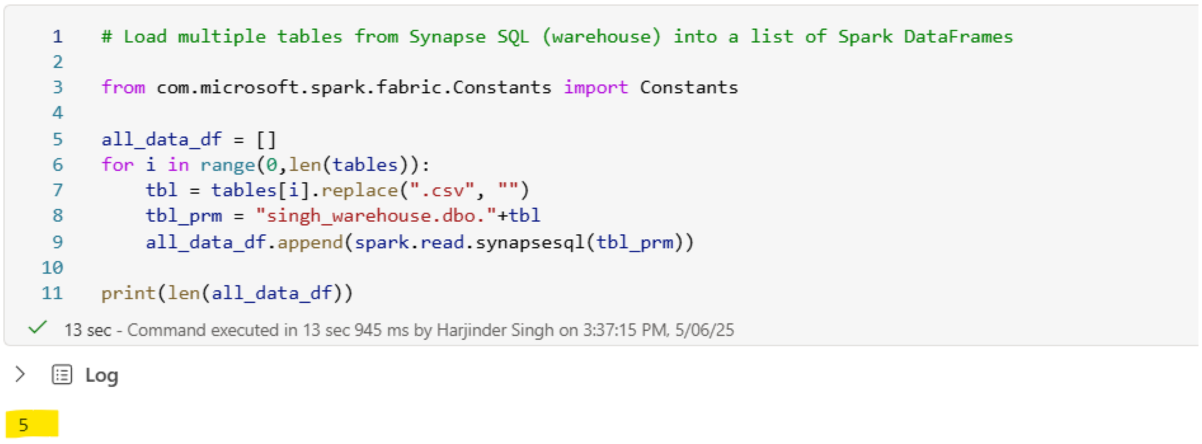
Advanced settings

Step 2: First, create list and store all the source files name.



```
# List of CSV filenames containing different financial datasets to be processed or analyzed

tables = ['accounts.csv', 'customers.csv', 'loan_payments.csv', 'loans.csv', 'transactions.csv']
```

✓ 1 sec - Command executed in 336 ms by Harjinder Singh on 3:37:01 PM, 5/06/25



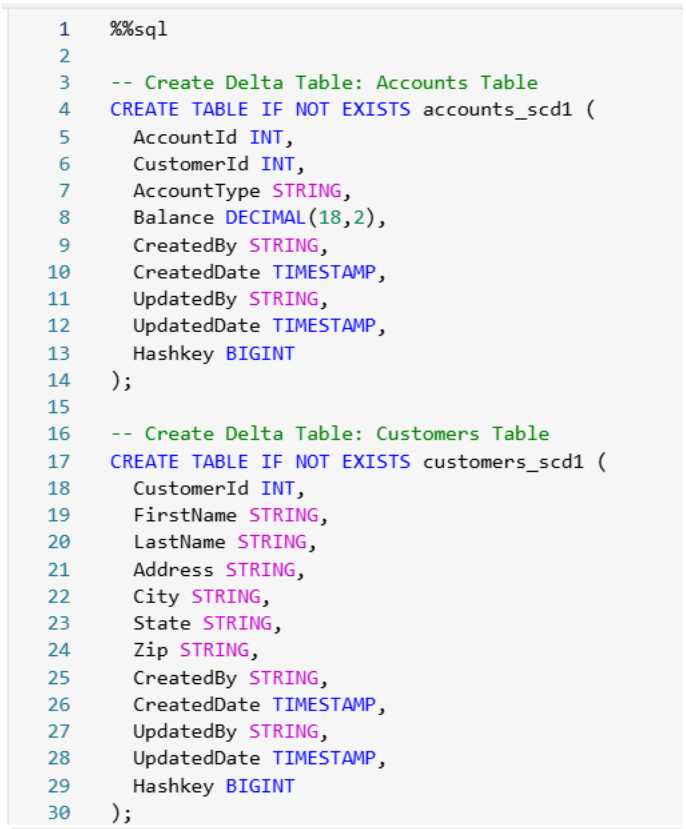
```
# Load multiple tables from Synapse SQL (warehouse) into a list of Spark DataFrames

from com.microsoft.spark.fabric.Constants import Constants

all_data_df = []
for i in range(0,len(tables)):
    tbl = tables[i].replace(".csv", "")
    tbl_prm = "singh_warehouse.dbo."+tbl
    all_data_df.append(spark.read.synapsesql(tbl_prm))

print(len(all_data_df))
```

✓ 13 sec - Command executed in 13 sec 945 ms by Harjinder Singh on 3:37:15 PM, 5/06/25

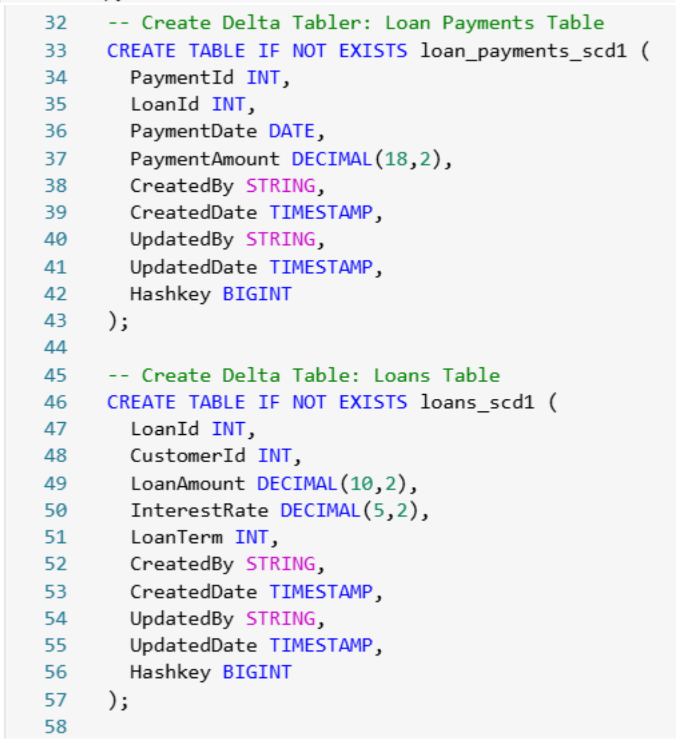
Log

```
5
```

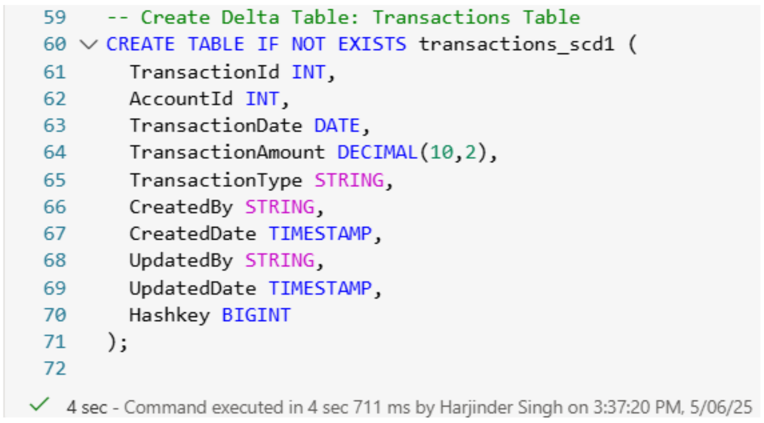
Step 3: Next, create the SCD1 type delta tables in the lakehouse for all the source tables.



```sql
%%sql

-- Create Delta Table: Accounts Table
CREATE TABLE IF NOT EXISTS accounts_scd1 (
  AccountId INT,
  CustomerId INT,
  AccountType STRING,
  Balance DECIMAL(18,2),
  CreatedBy STRING,
  CreatedDate TIMESTAMP,
  UpdatedBy STRING,
  UpdatedDate TIMESTAMP,
  Hashkey BIGINT
);

-- Create Delta Table: Customers Table
CREATE TABLE IF NOT EXISTS customers_scd1 (
  CustomerId INT,
  FirstName STRING,
  LastName STRING,
  Address STRING,
  City STRING,
  State STRING,
  Zip STRING,
  CreatedBy STRING,
  CreatedDate TIMESTAMP,
  UpdatedBy STRING,
  UpdatedDate TIMESTAMP,
  Hashkey BIGINT
);
```



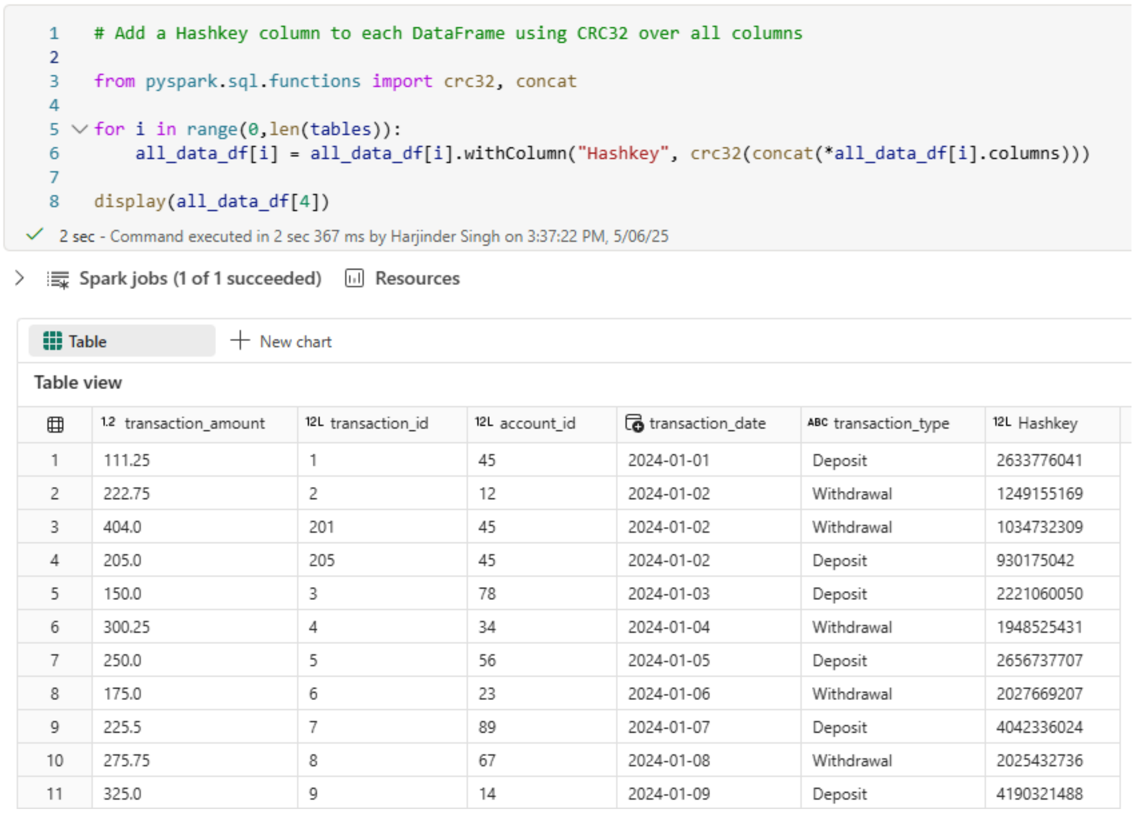
```sql
-- Create Delta Tabler: Loan Payments Table
CREATE TABLE IF NOT EXISTS loan_payments_scd1 (
  PaymentId INT,
  LoanId INT,
  PaymentDate DATE,
  PaymentAmount DECIMAL(18,2),
  CreatedBy STRING,
  CreatedDate TIMESTAMP,
  UpdatedBy STRING,
  UpdatedDate TIMESTAMP,
  Hashkey BIGINT
);

-- Create Delta Table: Loans Table
CREATE TABLE IF NOT EXISTS loans_scd1 (
  LoanId INT,
  CustomerId INT,
  LoanAmount DECIMAL(10,2),
  InterestRate DECIMAL(5,2),
  LoanTerm INT,
  CreatedBy STRING,
  CreatedDate TIMESTAMP,
  UpdatedBy STRING,
  UpdatedDate TIMESTAMP,
  Hashkey BIGINT
);
```



```
-- Create Delta Table: Transactions Table
CREATE TABLE IF NOT EXISTS transactions_scd1 (
  TransactionId INT,
  AccountId INT,
  TransactionDate DATE,
  TransactionAmount DECIMAL(10,2),
  TransactionType STRING,
  CreatedBy STRING,
  CreatedDate TIMESTAMP,
  UpdatedBy STRING,
  UpdatedDate TIMESTAMP,
  Hashkey BIGINT
);

```

✓ 4 sec - Command executed in 4 sec 711 ms by Harjinder Singh on 3:37:20 PM, 5/06/25

Step 4: Next, add Haskey column in all the source tables with Haskey values generated by concatenating all the column values and using crc32 method for all the source tables.



```
# Add a Hashkey column to each DataFrame using CRC32 over all columns

from pyspark.sql.functions import crc32, concat

for i in range(0,len(tables)):
    all_data_df[i] = all_data_df[i].withColumn("Hashkey", crc32(concat(*all_data_df[i].columns)))

display(all_data_df[4])
```

✓ 2 sec - Command executed in 2 sec 367 ms by Harjinder Singh on 3:37:22 PM, 5/06/25

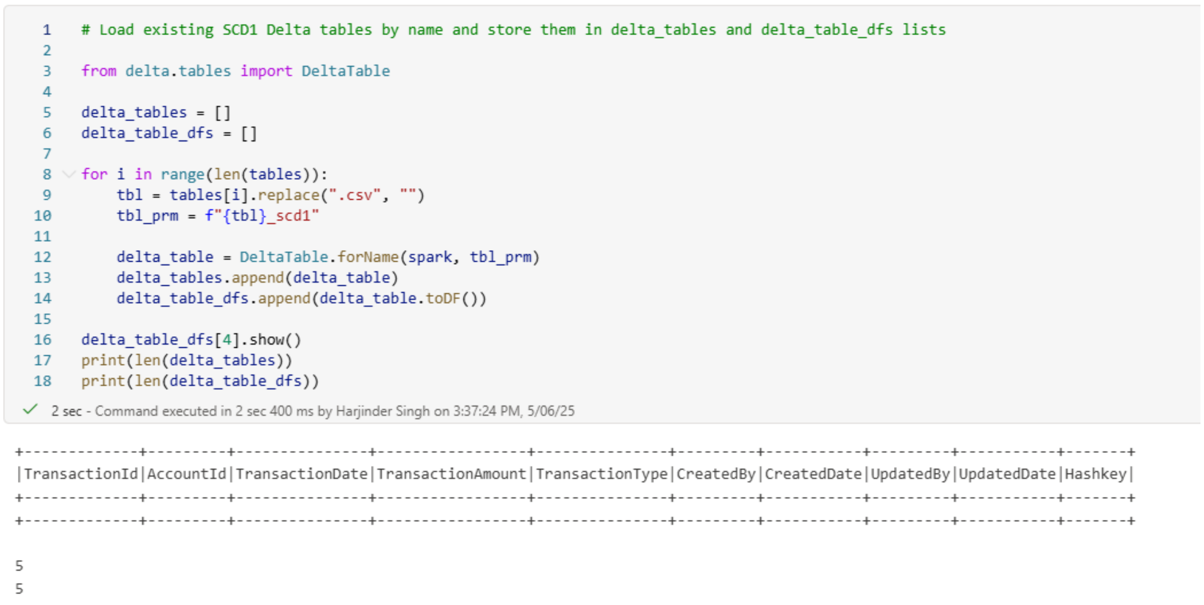
Spark jobs (1 of 1 succeeded)   Resources

Table   New chart

Table view

	transaction_amount	transaction_id	account_id	transaction_date	transaction_type	Hashkey
1	111.25	1	45	2024-01-01	Deposit	2633776041
2	222.75	2	12	2024-01-02	Withdrawal	1249155169
3	404.0	201	45	2024-01-02	Withdrawal	1034732309
4	205.0	205	45	2024-01-02	Deposit	930175042
5	150.0	3	78	2024-01-03	Deposit	2221060050
6	300.25	4	34	2024-01-04	Withdrawal	1948525431
7	250.0	5	56	2024-01-05	Deposit	2656737707
8	175.0	6	23	2024-01-06	Withdrawal	2027669207
9	225.5	7	89	2024-01-07	Deposit	4042336024
10	275.75	8	67	2024-01-08	Withdrawal	2025432736
11	325.0	9	14	2024-01-09	Deposit	4190321488

Step 5: Next, load existing SCD1 Delta tables by name and store them in delta_tables and delta_table_dfs lists.

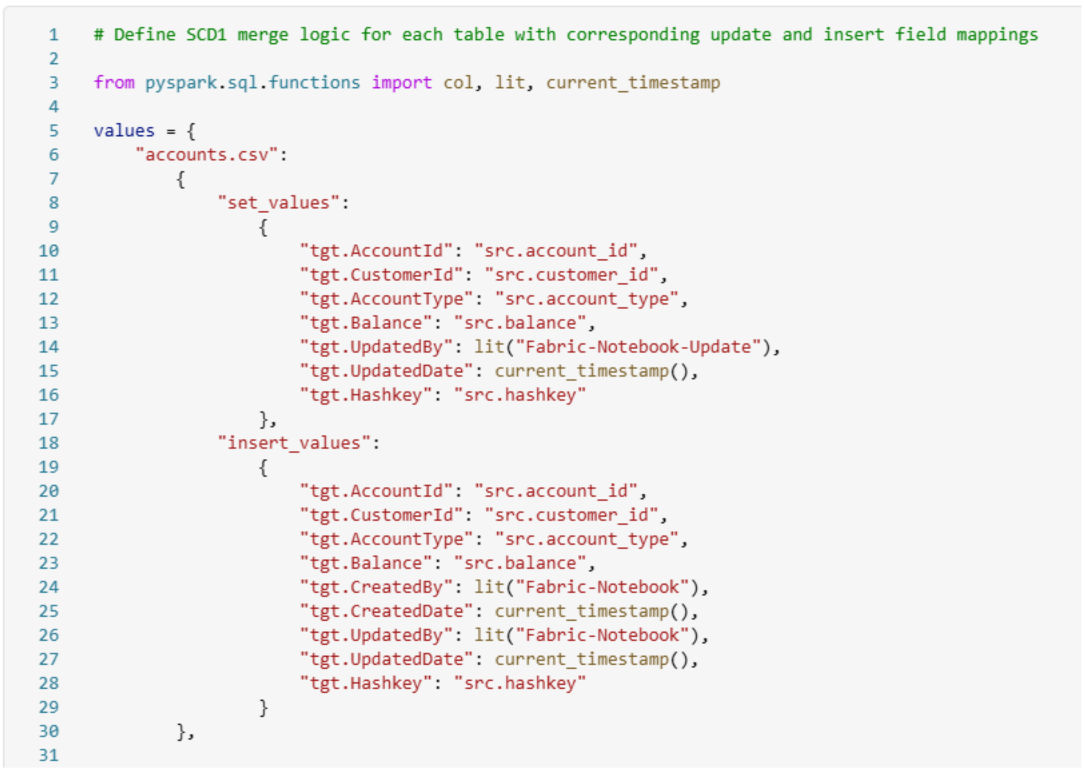


```python
# Load existing SCD1 Delta tables by name and store them in delta_tables and delta_table_dfs lists

from delta.tables import DeltaTable

delta_tables = []
delta_table_dfs = []

for i in range(len(tables)):
    tbl = tables[i].replace(".csv", "")
    tbl_prm = f"{tbl}_scd1"

    delta_table = DeltaTable.forName(spark, tbl_prm)
    delta_tables.append(delta_table)
    delta_table_dfs.append(delta_table.toDF())

delta_table_dfs[4].show()
print(len(delta_tables))
print(len(delta_table_dfs))
```

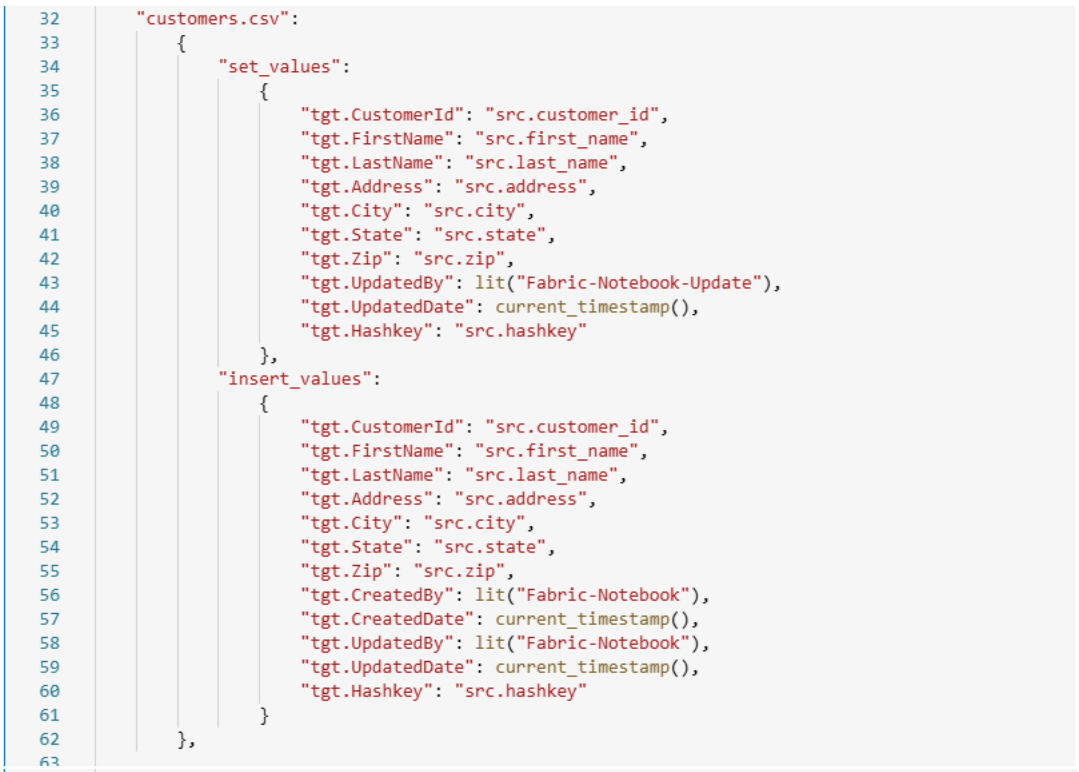
✓ 2 sec - Command executed in 2 sec 400 ms by Harjinder Singh on 3:37:24 PM, 5/06/25

```
+-------------+---------+---------------+-----------------+---------------+---------+-----------+---------+-----------+-------+
|TransactionId|AccountId|TransactionDate|TransactionAmount|TransactionType|CreatedBy|CreatedDate|UpdatedBy|UpdatedDate|Hashkey|
+-------------+---------+---------------+-----------------+---------------+---------+-----------+---------+-----------+-------+
+-------------+---------+---------------+-----------------+---------------+---------+-----------+---------+-----------+-------+

5
5
```

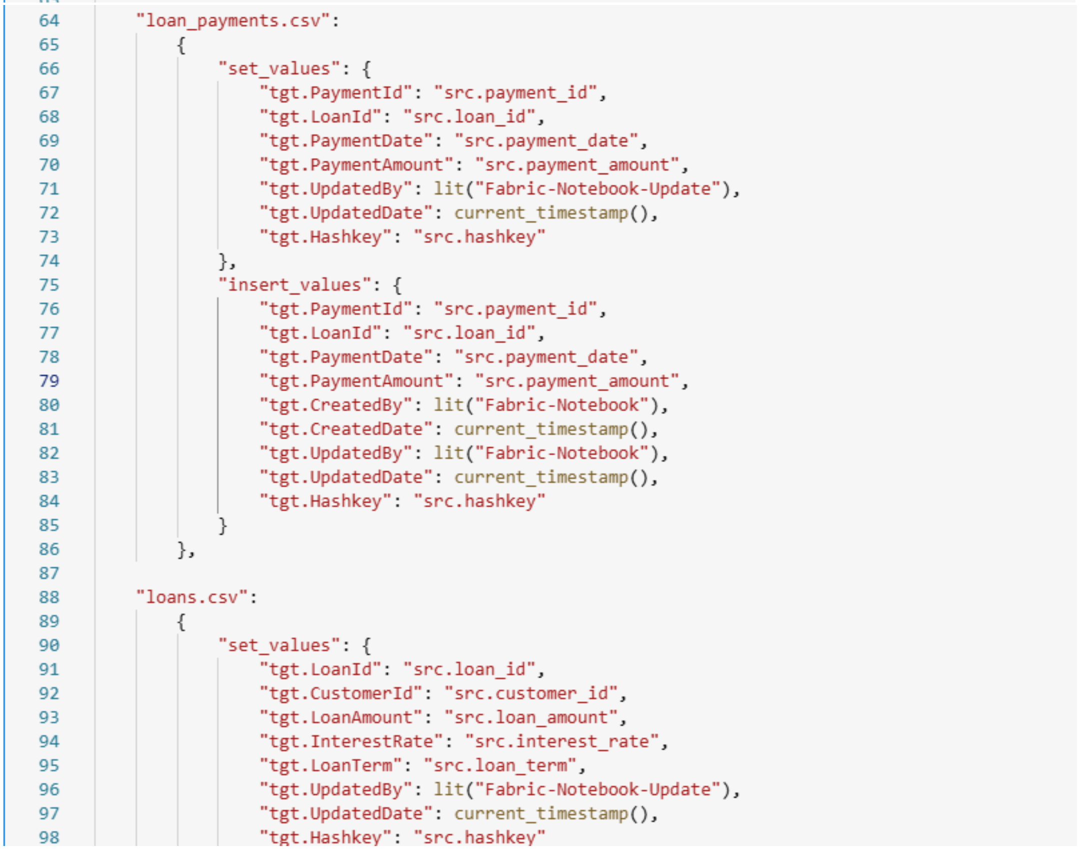
Step 6: Next, define SCD1 merge logic for each table with corresponding update and insert field mappings



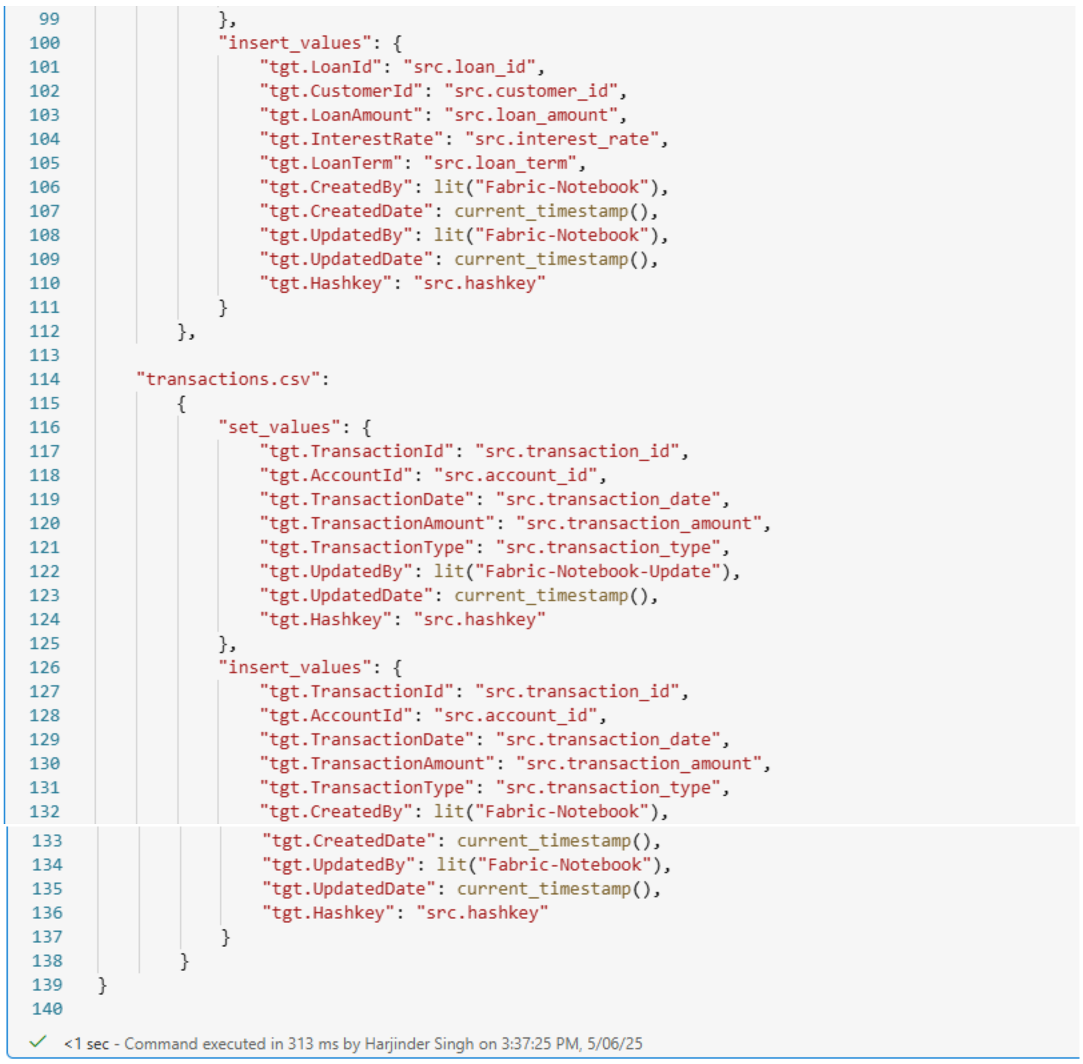
```python
# Define SCD1 merge logic for each table with corresponding update and insert field mappings

from pyspark.sql.functions import col, lit, current_timestamp

values = {
    "accounts.csv":
        {
            "set_values":
                {
                    "tgt.AccountId": "src.account_id",
                    "tgt.CustomerId": "src.customer_id",
                    "tgt.AccountType": "src.account_type",
                    "tgt.Balance": "src.balance",
                    "tgt.UpdatedBy": lit("Fabric-Notebook-Update"),
                    "tgt.UpdatedDate": current_timestamp(),
                    "tgt.Hashkey": "src.hashkey"
                },
            "insert_values":
                {
                    "tgt.AccountId": "src.account_id",
                    "tgt.CustomerId": "src.customer_id",
                    "tgt.AccountType": "src.account_type",
                    "tgt.Balance": "src.balance",
                    "tgt.CreatedBy": lit("Fabric-Notebook"),
                    "tgt.CreatedDate": current_timestamp(),
                    "tgt.UpdatedBy": lit("Fabric-Notebook"),
                    "tgt.UpdatedDate": current_timestamp(),
                    "tgt.Hashkey": "src.hashkey"
                }
        },

```



```
    "customers.csv":
        {
            "set_values":
                {
                    "tgt.CustomerId": "src.customer_id",
                    "tgt.FirstName": "src.first_name",
                    "tgt.LastName": "src.last_name",
                    "tgt.Address": "src.address",
                    "tgt.City": "src.city",
                    "tgt.State": "src.state",
                    "tgt.Zip": "src.zip",
                    "tgt.UpdatedBy": lit("Fabric-Notebook-Update"),
                    "tgt.UpdatedDate": current_timestamp(),
                    "tgt.Hashkey": "src.hashkey"
                },
            "insert_values":
                {
                    "tgt.CustomerId": "src.customer_id",
                    "tgt.FirstName": "src.first_name",
                    "tgt.LastName": "src.last_name",
                    "tgt.Address": "src.address",
                    "tgt.City": "src.city",
                    "tgt.State": "src.state",
                    "tgt.Zip": "src.zip",
                    "tgt.CreatedBy": lit("Fabric-Notebook"),
                    "tgt.CreatedDate": current_timestamp(),
                    "tgt.UpdatedBy": lit("Fabric-Notebook"),
                    "tgt.UpdatedDate": current_timestamp(),
                    "tgt.Hashkey": "src.hashkey"
                }
        },
```

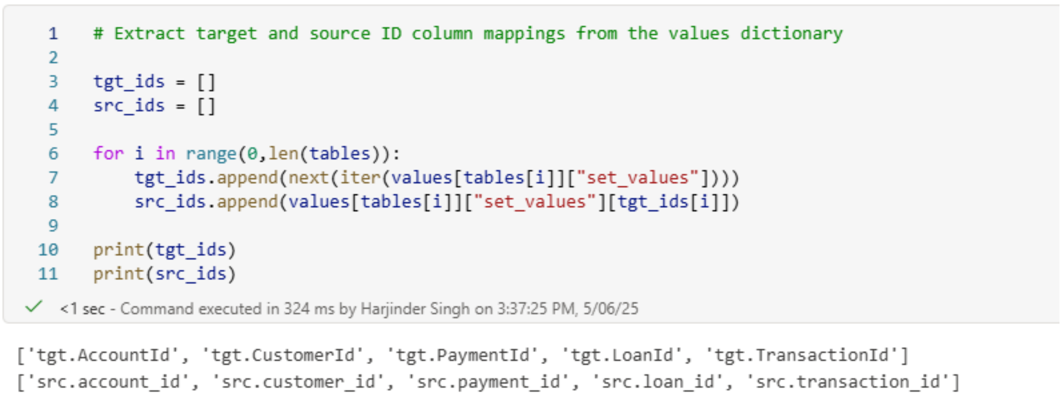


```
    "loan_payments.csv":
        {
            "set_values": {
                "tgt.PaymentId": "src.payment_id",
                "tgt.LoanId": "src.loan_id",
                "tgt.PaymentDate": "src.payment_date",
                "tgt.PaymentAmount": "src.payment_amount",
                "tgt.UpdatedBy": lit("Fabric-Notebook-Update"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
            },
            "insert_values": {
                "tgt.PaymentId": "src.payment_id",
                "tgt.LoanId": "src.loan_id",
                "tgt.PaymentDate": "src.payment_date",
                "tgt.PaymentAmount": "src.payment_amount",
                "tgt.CreatedBy": lit("Fabric-Notebook"),
                "tgt.CreatedDate": current_timestamp(),
                "tgt.UpdatedBy": lit("Fabric-Notebook"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
            }
        },

    "loans.csv":
        {
            "set_values": {
                "tgt.LoanId": "src.loan_id",
                "tgt.CustomerId": "src.customer_id",
                "tgt.LoanAmount": "src.loan_amount",
                "tgt.InterestRate": "src.interest_rate",
                "tgt.LoanTerm": "src.loan_term",
                "tgt.UpdatedBy": lit("Fabric-Notebook-Update"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
```



```
            },
            "insert_values": {
                "tgt.LoanId": "src.loan_id",
                "tgt.CustomerId": "src.customer_id",
                "tgt.LoanAmount": "src.loan_amount",
                "tgt.InterestRate": "src.interest_rate",
                "tgt.LoanTerm": "src.loan_term",
                "tgt.CreatedBy": lit("Fabric-Notebook"),
                "tgt.CreatedDate": current_timestamp(),
                "tgt.UpdatedBy": lit("Fabric-Notebook"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
            }
        },

    "transactions.csv":
        {
            "set_values": {
                "tgt.TransactionId": "src.transaction_id",
                "tgt.AccountId": "src.account_id",
                "tgt.TransactionDate": "src.transaction_date",
                "tgt.TransactionAmount": "src.transaction_amount",
                "tgt.TransactionType": "src.transaction_type",
                "tgt.UpdatedBy": lit("Fabric-Notebook-Update"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
            },
            "insert_values": {
                "tgt.TransactionId": "src.transaction_id",
                "tgt.AccountId": "src.account_id",
                "tgt.TransactionDate": "src.transaction_date",
                "tgt.TransactionAmount": "src.transaction_amount",
                "tgt.TransactionType": "src.transaction_type",
                "tgt.CreatedBy": lit("Fabric-Notebook"),
                "tgt.CreatedDate": current_timestamp(),
                "tgt.UpdatedBy": lit("Fabric-Notebook"),
                "tgt.UpdatedDate": current_timestamp(),
                "tgt.Hashkey": "src.hashkey"
            }
        }
}

✓ <1 sec - Command executed in 313 ms by Harjinder Singh on 3:37:25 PM, 5/06/25
```

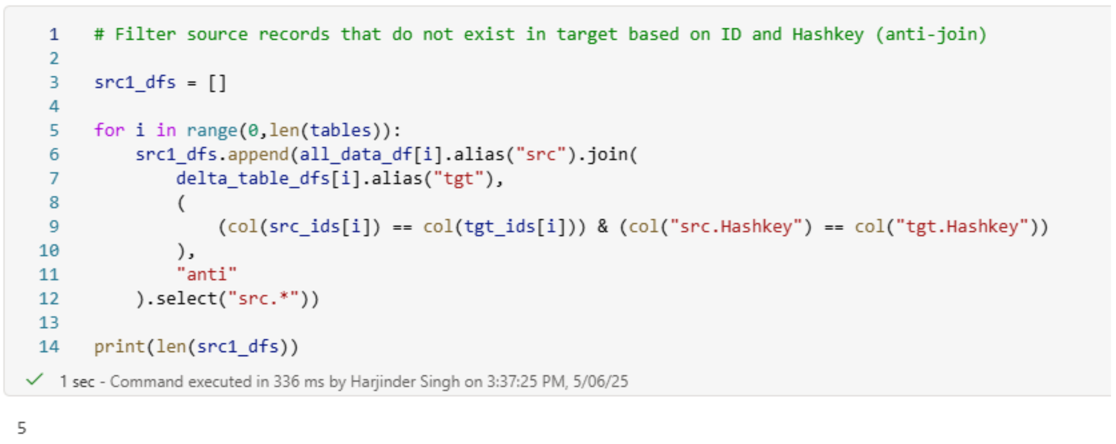
Step 7: Next, extract target and source ID column mappings from the values dictionary



```
# Extract target and source ID column mappings from the values dictionary

tgt_ids = []
src_ids = []

for i in range(0,len(tables)):
    tgt_ids.append(next(iter(values[tables[i]]["set_values"])))
    src_ids.append(values[tables[i]]["set_values"][tgt_ids[i]])

print(tgt_ids)
print(src_ids)

✓ <1 sec - Command executed in 324 ms by Harjinder Singh on 3:37:25 PM, 5/06/25
```

```
['tgt.AccountId', 'tgt.CustomerId', 'tgt.PaymentId', 'tgt.LoanId', 'tgt.TransactionId']
['src.account_id', 'src.customer_id', 'src.payment_id', 'src.loan_id', 'src.transaction_id']
```

Step 8: Next, filter source records that do not exist in target based on ID and Hashkey (anti-join).

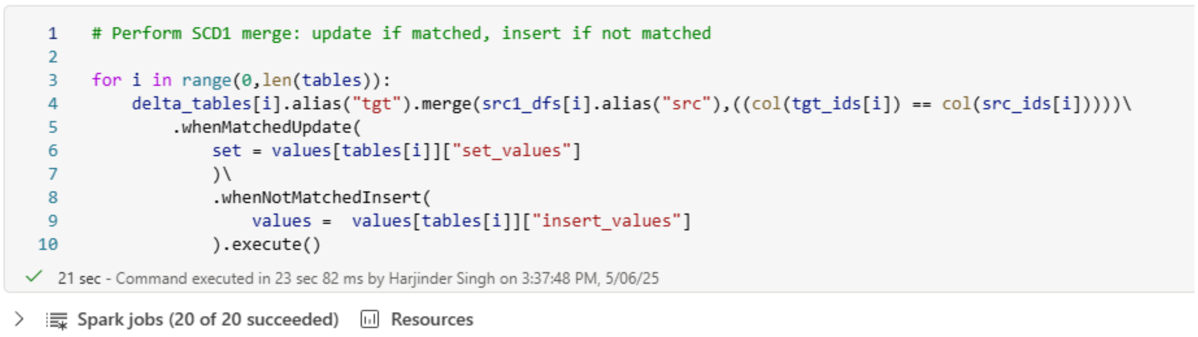


```python
# Filter source records that do not exist in target based on ID and Hashkey (anti-join)

src1_dfs = []

for i in range(0,len(tables)):
    src1_dfs.append(all_data_df[i].alias("src").join(
        delta_table_dfs[i].alias("tgt"),
        (
            (col(src_ids[i]) == col(tgt_ids[i])) & (col("src.Hashkey") == col("tgt.Hashkey"))
        ),
        "anti"
    ).select("src.*"))

print(len(src1_dfs))
```

✓ 1 sec - Command executed in 336 ms by Harjinder Singh on 3:37:25 PM, 5/06/25

```
5
```

Step 9: Next, perform SCD1 merge: update if matched, insert if not matched



```python
# Perform SCD1 merge: update if matched, insert if not matched

for i in range(0,len(tables)):
    delta_tables[i].alias("tgt").merge(src1_dfs[i].alias("src"),((col(tgt_ids[i]) == col(src_ids[i]))))\
        .whenMatchedUpdate(
            set = values[tables[i]]["set_values"]
            )\
            .whenNotMatchedInsert(
                values =  values[tables[i]]["insert_values"]
            ).execute()
```

✓ 21 sec - Command executed in 23 sec 82 ms by Harjinder Singh on 3:37:48 PM, 5/06/25

> Spark jobs (20 of 20 succeeded)   Resources

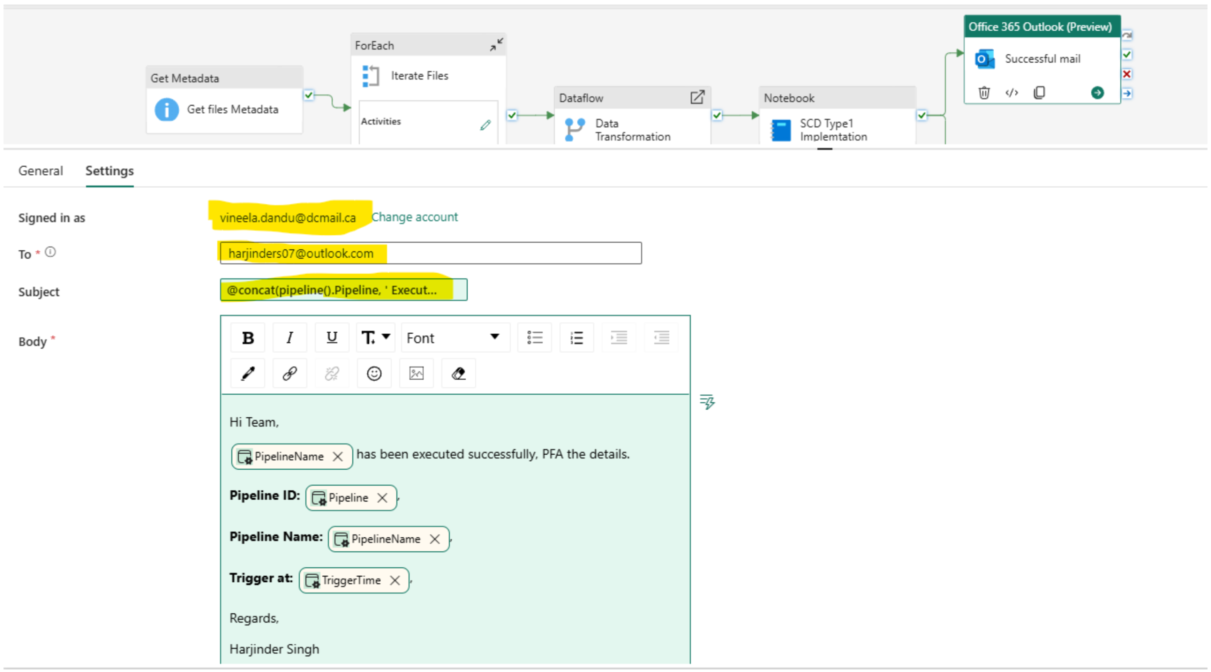
Pipeline Execution Success Notification

✓ **Steps To setup success mail notification.**

– Add a new Outlook activity and connect it with on success point of Notebook activity.

– Go to settings tab, signed in as with user email (Make sure the email has its subscription otherwise it will not work of personal emails.).

– Enter the basic details such as To, Subject, and Body as shown below.



Signed in as vineela.dandu@dcmail.ca Change account

To * harjinders07@outlook.com

Subject @concat(pipeline().Pipeline, ' Execut...

Body *

Hi Team,

PipelineName has been executed successfully, PFA the details.

Pipeline ID: Pipeline

Pipeline Name: PipelineName

Trigger at: TriggerTime

Regards,

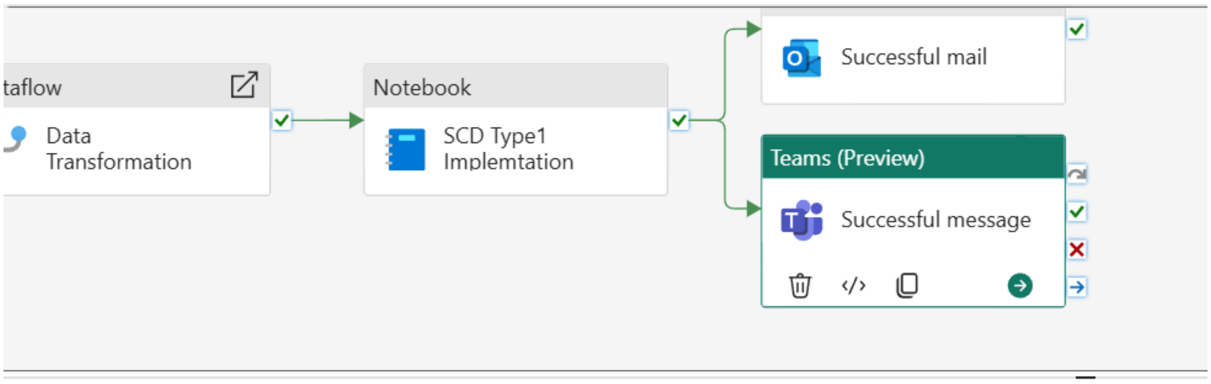
Harjinder Singh

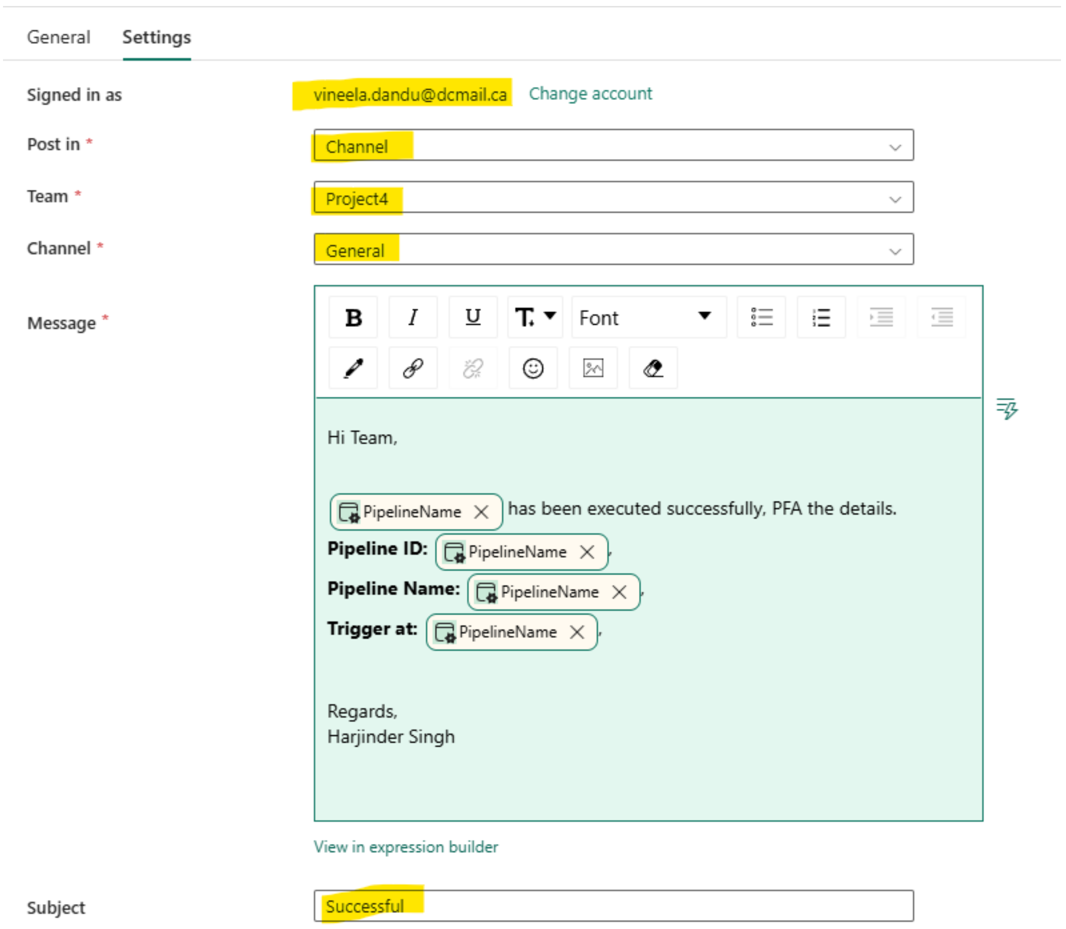
✓ **Steps To setup success mail notification.**

– Add a new Teams activity and connect it with on success point of Notebook activity.

– Go to settings tab, signed in as with user email (Make sure the email has its subscription otherwise it will not work of personal emails.).

– Enter the basic details such as Post in, Team, Channel, Message and Subject as shown below.

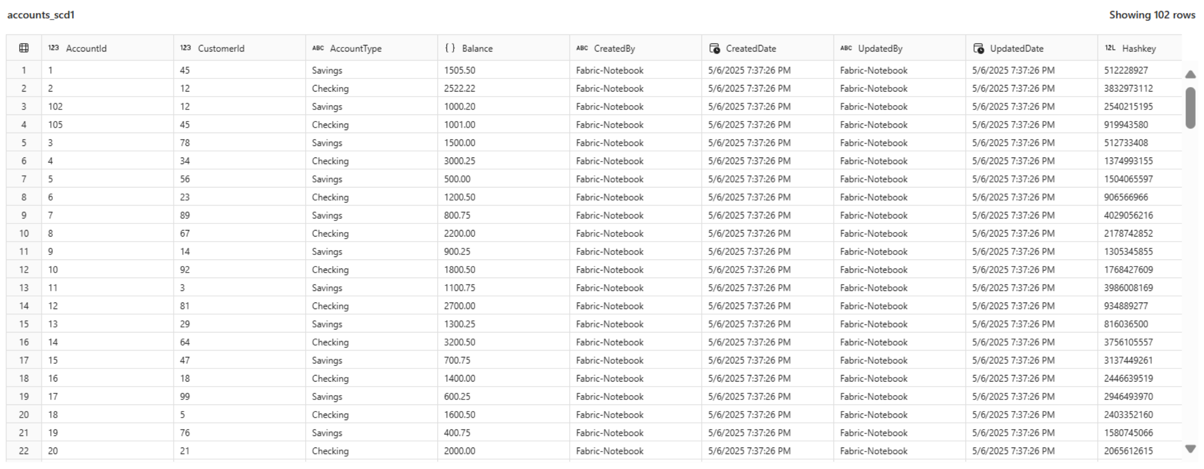




Signed in as	vineela.dandu@dcmail.ca Change account
Post in *	Channel
Team *	Project4
Channel *	General
Message *	Hi Team, PipelineName has been executed successfully, PFA the details. **Pipeline ID:** PipelineName, **Pipeline Name:** PipelineName, **Trigger at:** PipelineName, Regards, Harjinder Singh
	View in expression builder
Subject	Successful

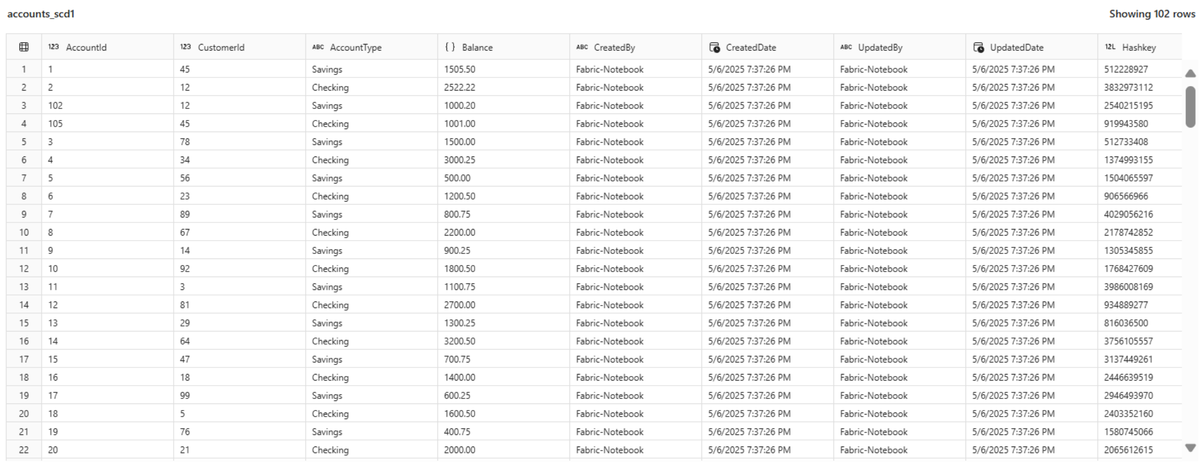
Output Review

✓ First Day Output:



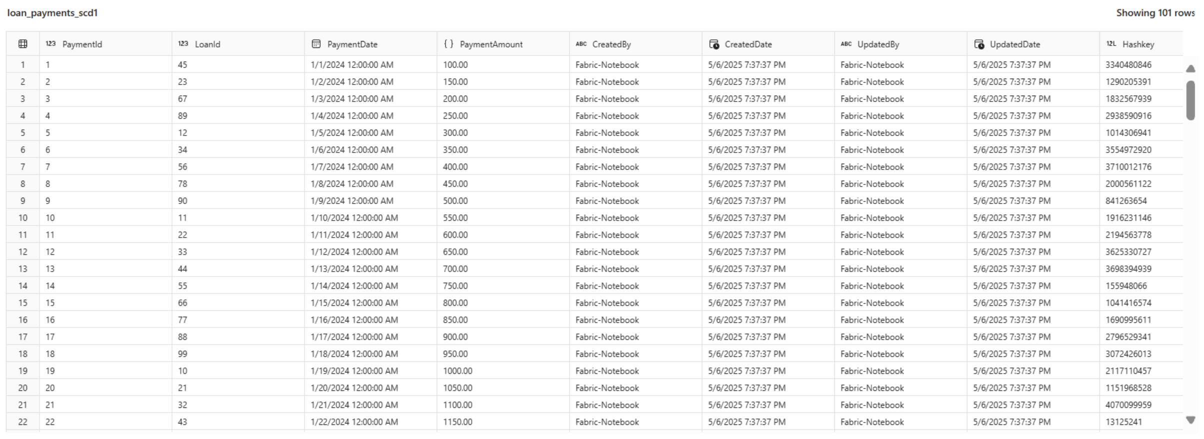
accounts_scd1 — Showing 102 rows

	AccountId	CustomerId	AccountType	Balance	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	Savings	1505.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	512228927
2	2	12	Checking	2522.22	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3832973112
3	102	12	Savings	1000.20	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2540215195
4	105	45	Checking	1001.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	919943580
5	3	78	Savings	1500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	512733408
6	4	34	Checking	3000.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1374993155
7	5	56	Savings	500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1504065597
8	6	23	Checking	1200.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	906566966
9	7	89	Savings	800.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	4029056216
10	8	67	Checking	2200.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2178742852
11	9	14	Savings	900.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1305345855
12	10	92	Checking	1800.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1768427609
13	11	3	Savings	1100.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3986008169
14	12	81	Checking	2700.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	934889277
15	13	29	Savings	1300.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	816036500
16	14	64	Checking	3200.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3756105557
17	15	47	Savings	700.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3137449261
18	16	18	Checking	1400.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2446639519
19	17	99	Savings	600.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2946493970
20	18	5	Checking	1600.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2403352160
21	19	76	Savings	400.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1580745066
22	20	21	Checking	2000.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2065612615



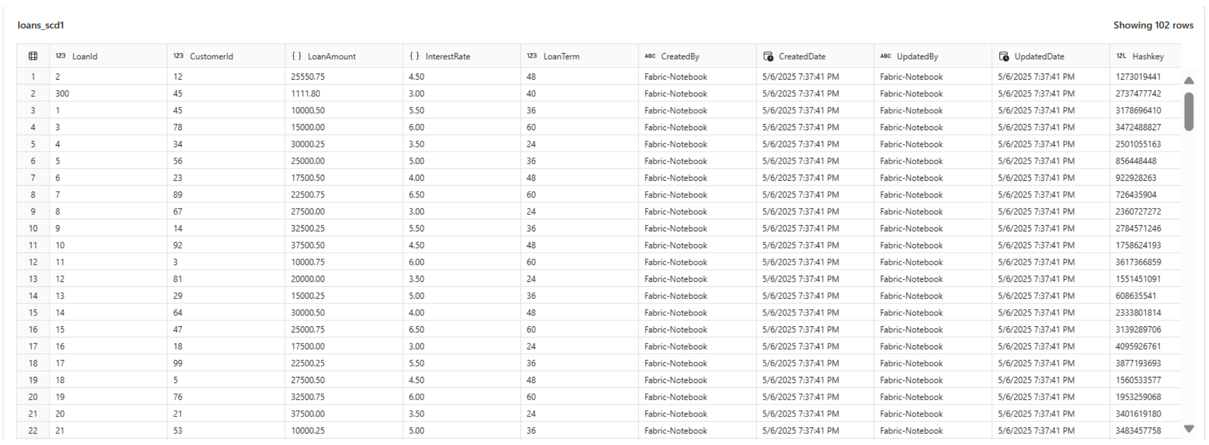
accounts_scd1 — Showing 102 rows

	AccountId	CustomerId	AccountType	Balance	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	Savings	1505.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	512228927
2	2	12	Checking	2522.22	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3832973112
3	102	12	Savings	1000.20	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2540215195
4	105	45	Checking	1001.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	919943580
5	3	78	Savings	1500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	512733408
6	4	34	Checking	3000.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1374993155
7	5	56	Savings	500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1504065597
8	6	23	Checking	1200.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	906566966
9	7	89	Savings	800.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	4029056216
10	8	67	Checking	2200.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2178742852
11	9	14	Savings	900.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1305345855
12	10	92	Checking	1800.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1768427609
13	11	3	Savings	1100.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3986008169
14	12	81	Checking	2700.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	934889277
15	13	29	Savings	1300.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	816036500
16	14	64	Checking	3200.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3756105557
17	15	47	Savings	700.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3137449261
18	16	18	Checking	1400.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2446639519
19	17	99	Savings	600.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2946493970
20	18	5	Checking	1600.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2403352160
21	19	76	Savings	400.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1580745066
22	20	21	Checking	2000.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2065612615



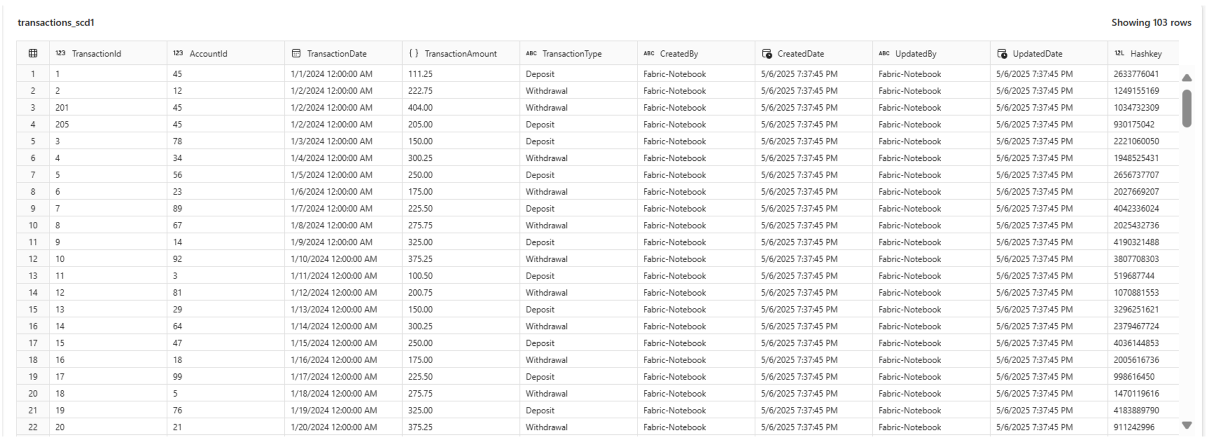
loan_payments_scd1 — Showing 101 rows

	PaymentId	LoanId	PaymentDate	PaymentAmount	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	1/1/2024 12:00:00 AM	100.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3340480846
2	2	23	1/2/2024 12:00:00 AM	150.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1290205391
3	3	67	1/3/2024 12:00:00 AM	200.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1832567939
4	4	89	1/4/2024 12:00:00 AM	250.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	2938590916
5	5	12	1/5/2024 12:00:00 AM	300.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1014306941
6	6	34	1/6/2024 12:00:00 AM	350.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3554972920
7	7	56	1/7/2024 12:00:00 AM	400.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3710012176
8	8	78	1/8/2024 12:00:00 AM	450.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	2000561122
9	9	90	1/9/2024 12:00:00 AM	500.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	841263654
10	10	11	1/10/2024 12:00:00 AM	550.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1916231146
11	11	22	1/11/2024 12:00:00 AM	600.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	2194563778
12	12	33	1/12/2024 12:00:00 AM	650.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3625330727
13	13	44	1/13/2024 12:00:00 AM	700.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3698394939
14	14	55	1/14/2024 12:00:00 AM	750.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	155948066
15	15	66	1/15/2024 12:00:00 AM	800.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1041416574
16	16	77	1/16/2024 12:00:00 AM	850.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1690995611
17	17	88	1/17/2024 12:00:00 AM	900.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	2796529341
18	18	99	1/18/2024 12:00:00 AM	950.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	3072426013
19	19	10	1/19/2024 12:00:00 AM	1000.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	2117110457
20	20	21	1/20/2024 12:00:00 AM	1050.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	1151968528
21	21	32	1/21/2024 12:00:00 AM	1100.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	4070099959
22	22	43	1/22/2024 12:00:00 AM	1150.00	Fabric-Notebook	5/6/2025 7:37:37 PM	Fabric-Notebook	5/6/2025 7:37:37 PM	13125241



loans_scd1 | Showing 102 rows

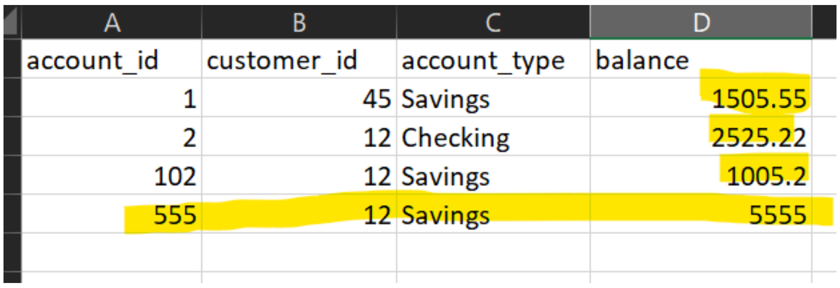
	LoanId	CustomerId	LoanAmount	InterestRate	LoanTerm	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	2	12	25550.75	4.50	48	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	1273019441
2	300	45	1111.80	3.00	40	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	2737477742
3	1	45	10000.50	5.50	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3178696410
4	3	78	15000.00	6.00	60	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3472488827
5	4	34	30000.25	3.50	24	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	2501055163
6	5	56	25000.00	5.00	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	856448448
7	6	23	17500.50	4.00	48	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	922928263
8	7	89	22500.75	6.50	60	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	726435904
9	8	67	27500.00	3.00	24	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	2360727272
10	9	14	32500.25	5.50	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	2784571246
11	10	92	37500.50	4.50	48	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	1758624193
12	11	3	10000.75	6.00	60	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3617366859
13	12	81	20000.00	3.50	24	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	1551451091
14	13	29	15000.25	5.00	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	608635541
15	14	64	30000.50	4.00	48	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	2333801814
16	15	47	25000.75	6.50	60	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3139289706
17	16	18	17500.00	3.00	24	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	4095926761
18	17	99	22500.25	5.50	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3877193693
19	18	5	27500.50	4.50	48	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	1560533577
20	19	76	32500.75	6.00	60	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	1953259068
21	20	21	37500.00	3.50	24	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3401619180
22	21	53	10000.25	5.00	36	Fabric-Notebook	5/6/2025 7:37:41 PM	Fabric-Notebook	5/6/2025 7:37:41 PM	3483457758



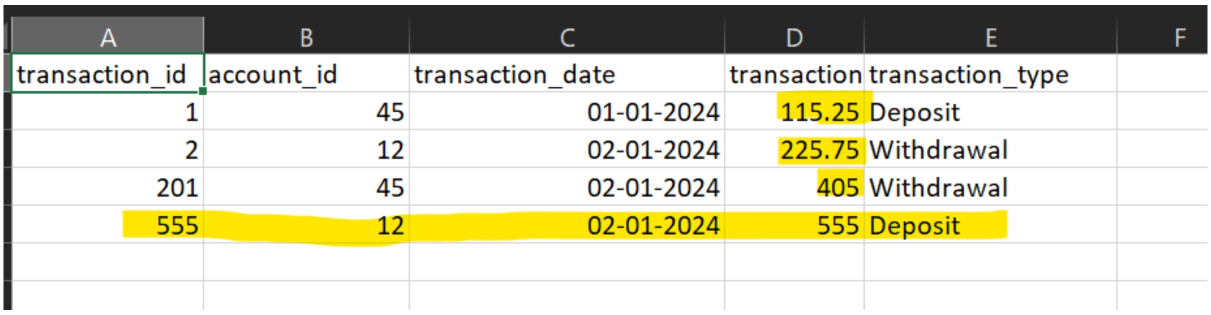
transactions_scd1 | Showing 103 rows

	TransactionId	AccountId	TransactionDate	TransactionAmount	TransactionType	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	1/1/2024 12:00:00 AM	111.25	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2633776041
2	2	12	1/2/2024 12:00:00 AM	222.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1249155169
3	201	45	1/2/2024 12:00:00 AM	404.00	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1034732309
4	205	45	1/2/2024 12:00:00 AM	205.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	930175042
5	3	78	1/3/2024 12:00:00 AM	150.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2221060050
6	4	34	1/4/2024 12:00:00 AM	300.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1948525431
7	5	56	1/5/2024 12:00:00 AM	250.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2656737707
8	6	23	1/6/2024 12:00:00 AM	175.00	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2027669207
9	7	89	1/7/2024 12:00:00 AM	225.50	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4042336024
10	8	67	1/8/2024 12:00:00 AM	275.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2025432736
11	9	14	1/9/2024 12:00:00 AM	325.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4190321488
12	10	92	1/10/2024 12:00:00 AM	375.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	3807708303
13	11	3	1/11/2024 12:00:00 AM	100.50	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	519687744
14	12	81	1/12/2024 12:00:00 AM	200.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1070881553
15	13	29	1/13/2024 12:00:00 AM	150.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	3296251621
16	14	64	1/14/2024 12:00:00 AM	300.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2379467724
17	15	47	1/15/2024 12:00:00 AM	250.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4036144853
18	16	18	1/16/2024 12:00:00 AM	175.00	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2005616736
19	17	99	1/17/2024 12:00:00 AM	225.50	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	998616450
20	18	5	1/18/2024 12:00:00 AM	275.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1470119616
21	19	76	1/19/2024 12:00:00 AM	325.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4183889790
22	20	21	1/20/2024 12:00:00 AM	375.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	911242996

✓ **Second Day data:**

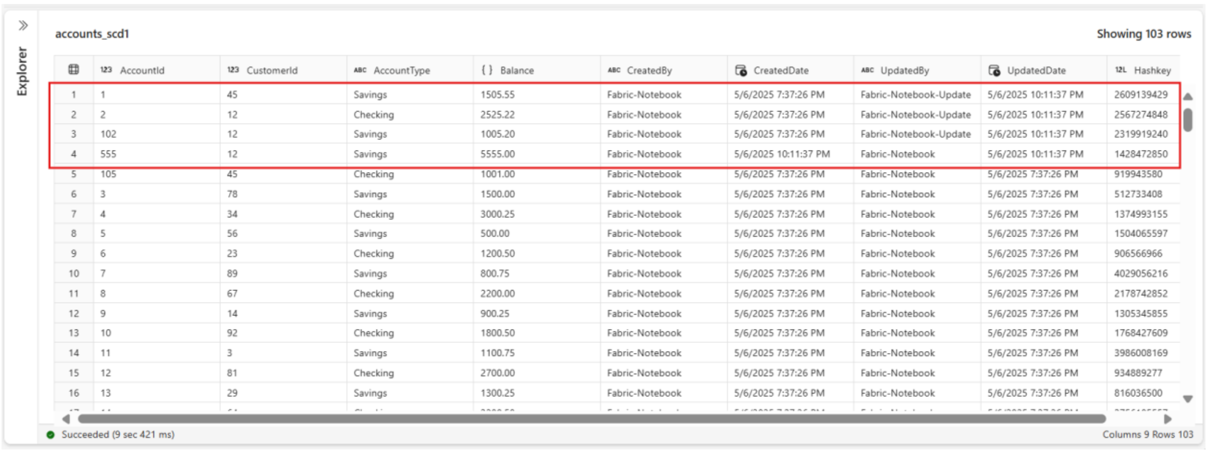


A	B	C	D
account_id	customer_id	account_type	balance
1	45	Savings	1505.55
2	12	Checking	2525.22
102	12	Savings	1005.2
555	12	Savings	5555



A	B	C	D	E	F
transaction_id	account_id	transaction_date	transaction	transaction_type	
1	45	01-01-2024	115.25	Deposit	
2	12	02-01-2024	225.75	Withdrawal	
201	45	02-01-2024	405	Withdrawal	
555	12	02-01-2024	555	Deposit	

✓ Second Day Output (only Updated tables):



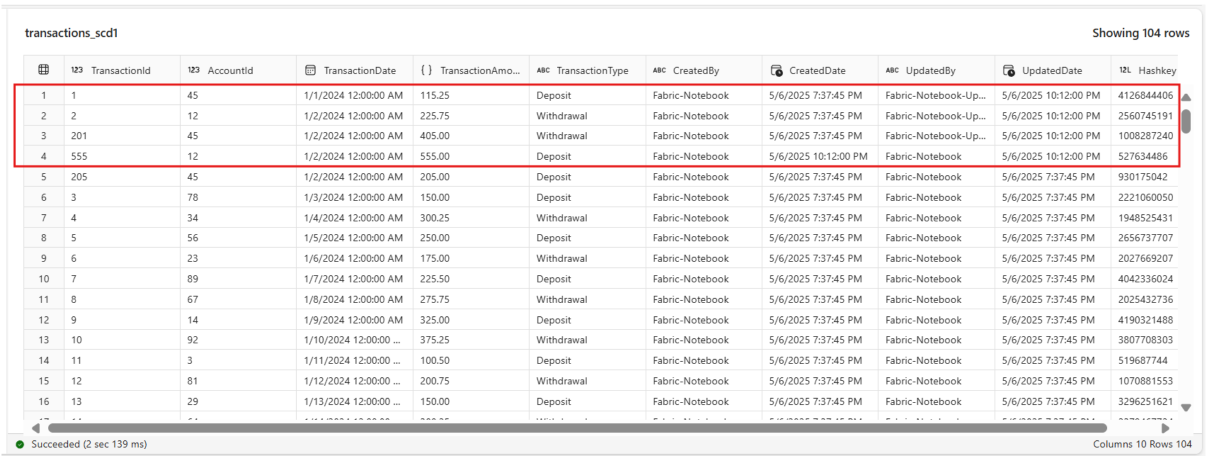
accounts_scd1

Showing 103 rows

	AccountId	CustomerId	AccountType	Balance	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	Savings	1505.55	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook-Update	5/6/2025 10:11:37 PM	2609139429
2	2	12	Checking	2525.22	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook-Update	5/6/2025 10:11:37 PM	2567274848
3	102	12	Savings	1005.20	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook-Update	5/6/2025 10:11:37 PM	2319919240
4	555	12	Savings	5555.00	Fabric-Notebook	5/6/2025 10:11:37 PM	Fabric-Notebook	5/6/2025 10:11:37 PM	1428472850
5	105	45	Checking	1001.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	919943580
6	3	78	Savings	1500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	512733408
7	4	34	Checking	3000.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1374993155
8	5	56	Savings	500.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1504065597
9	6	23	Checking	1200.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	906566966
10	7	89	Savings	800.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	4029056216
11	8	67	Checking	2200.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	2178742852
12	9	14	Savings	900.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1305345855
13	10	92	Checking	1800.50	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	1768427609
14	11	3	Savings	1100.75	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	3986008169
15	12	81	Checking	2700.00	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	934889277
16	13	29	Savings	1300.25	Fabric-Notebook	5/6/2025 7:37:26 PM	Fabric-Notebook	5/6/2025 7:37:26 PM	816036500

Succeeded (9 sec 421 ms)

Columns 9 Rows 103



transactions_scd1

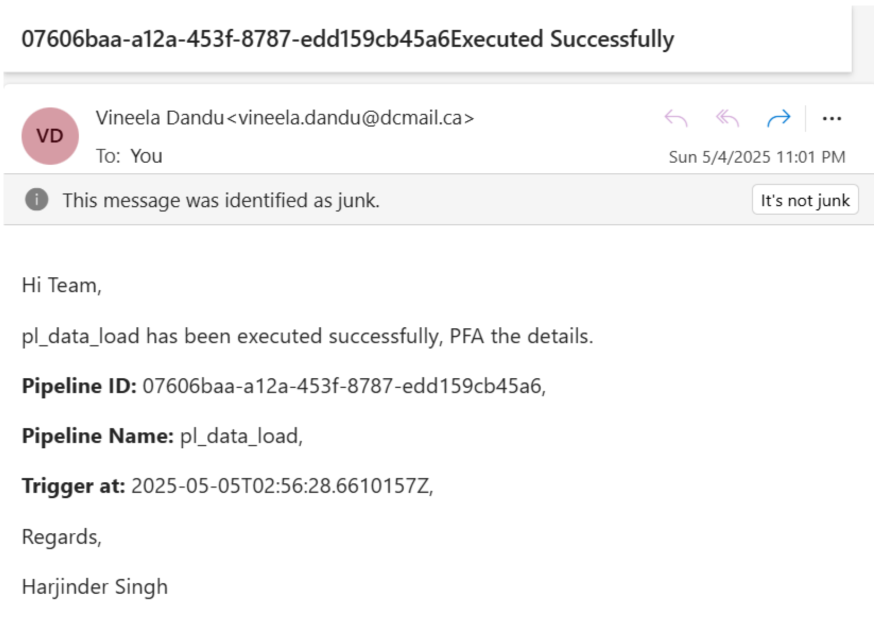
Showing 104 rows

	TransactionId	AccountId	TransactionDate	TransactionAmo...	TransactionType	CreatedBy	CreatedDate	UpdatedBy	UpdatedDate	Hashkey
1	1	45	1/1/2024 12:00:00 AM	115.25	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook-Up...	5/6/2025 10:12:00 PM	4126844406
2	2	12	1/2/2024 12:00:00 AM	225.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook-Up...	5/6/2025 10:12:00 PM	2560745191
3	201	45	1/2/2024 12:00:00 AM	405.00	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook-Up...	5/6/2025 10:12:00 PM	1008287240
4	555	12	1/2/2024 12:00:00 AM	555.00	Deposit	Fabric-Notebook	5/6/2025 10:12:00 PM	Fabric-Notebook	5/6/2025 10:12:00 PM	527634486
5	205	45	1/2/2024 12:00:00 AM	205.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	930175042
6	3	78	1/3/2024 12:00:00 AM	150.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2221060050
7	4	34	1/4/2024 12:00:00 AM	300.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1948525431
8	5	56	1/5/2024 12:00:00 AM	250.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2656737707
9	6	23	1/6/2024 12:00:00 AM	175.00	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2027669207
10	7	89	1/7/2024 12:00:00 AM	225.50	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4042336024
11	8	67	1/8/2024 12:00:00 AM	275.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	2025432736
12	9	14	1/9/2024 12:00:00 AM	325.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	4190321488
13	10	92	1/10/2024 12:00:00 ...	375.25	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	3807708303
14	11	3	1/11/2024 12:00:00 ...	100.50	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	519687744
15	12	81	1/12/2024 12:00:00 ...	200.75	Withdrawal	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	1070881553
16	13	29	1/13/2024 12:00:00 ...	150.00	Deposit	Fabric-Notebook	5/6/2025 7:37:45 PM	Fabric-Notebook	5/6/2025 7:37:45 PM	3296251621

Succeeded (2 sec 139 ms)

Columns 10 Rows 104

✓ Email Notification:



07606baa-a12a-453f-8787-edd159cb45a6Executed Successfully

VD Vineela Dandu<vineela.dandu@dcmail.ca>

To: You

Sun 5/4/2025 11:01 PM

This message was identified as junk.

Hi Team,

pl_data_load has been executed successfully, PFA the details.

Pipeline ID: 07606baa-a12a-453f-8787-edd159cb45a6,

Pipeline Name: pl_data_load,

Trigger at: 2025-05-05T02:56:28.6610157Z,

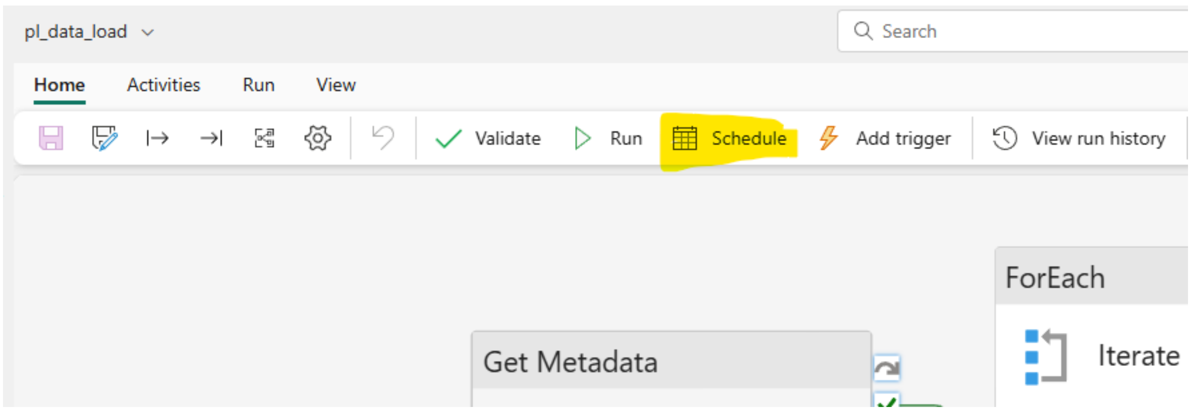
Regards,

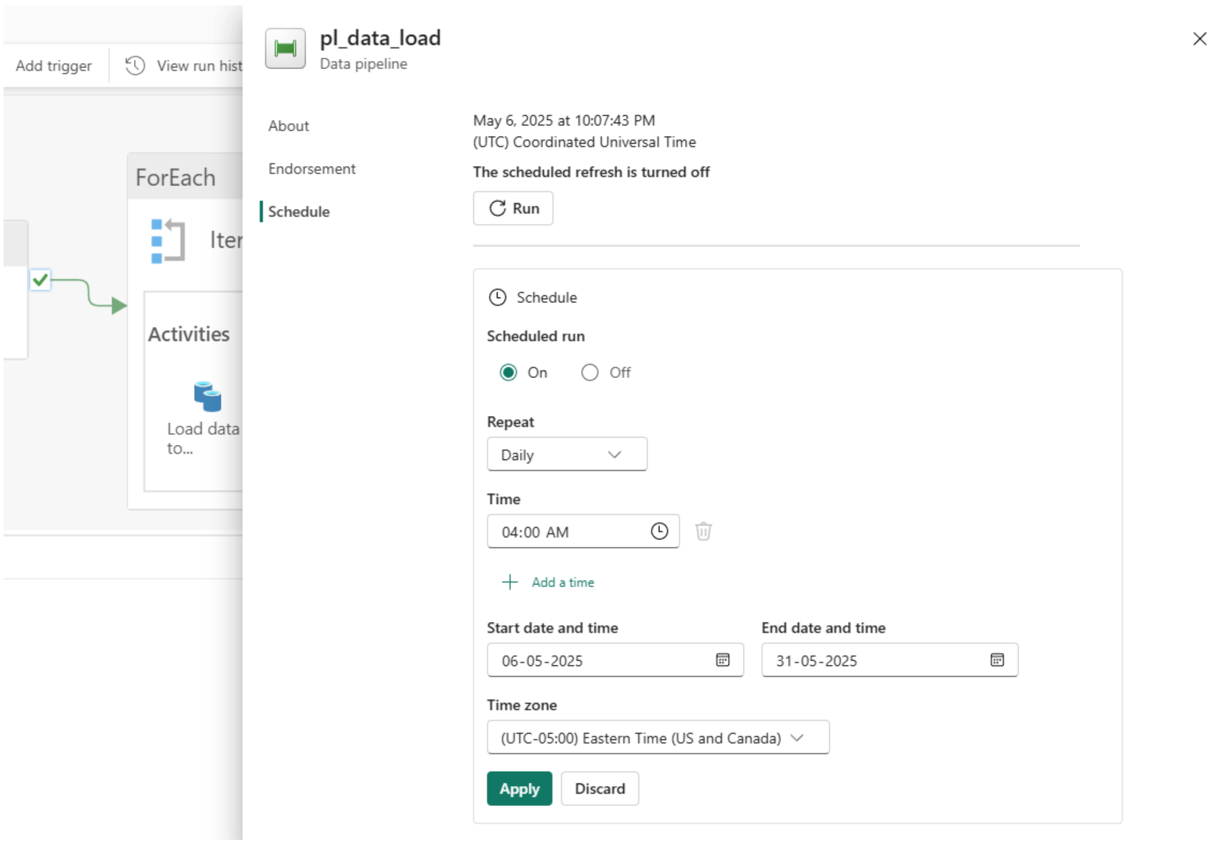
Harjinder Singh

Pipeline Execution Automation

✓ **Steps To schedule the pipeline execution**

- Click on Schedule option.
- Turn on the Schedule run and enter the basic details such as Repeat, Time, start and end date, etc., as shown below.
- Click on Apply.





May 6, 2025 at 10:07:43 PM
(UTC) Coordinated Universal Time

The scheduled refresh is turned off

Run

Schedule

Scheduled run

[✓ On] Off

Repeat: Daily

Time: 04:00 AM

Add a time

Start date and time: 06-05-2025

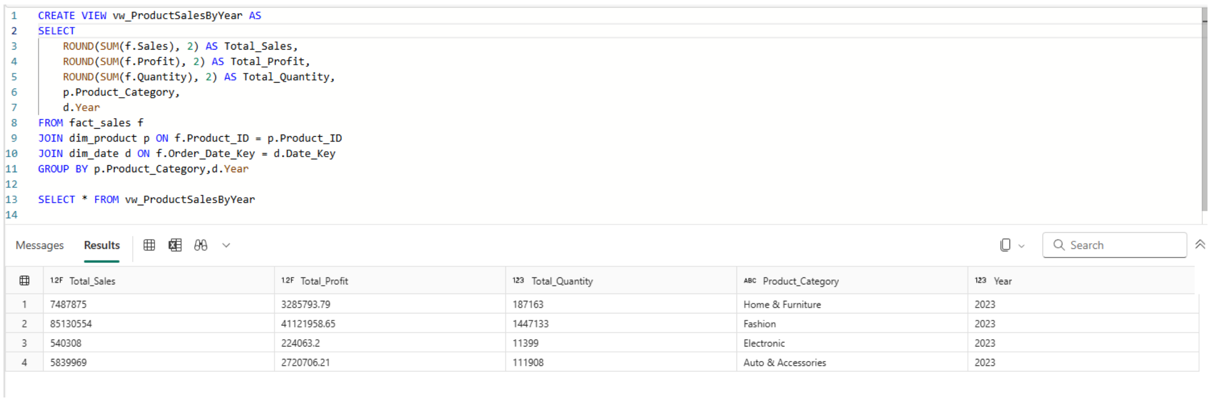
End date and time: 31-05-2025

Time zone: (UTC-05:00) Eastern Time (US and Canada)

Apply Discard

Power BI Report

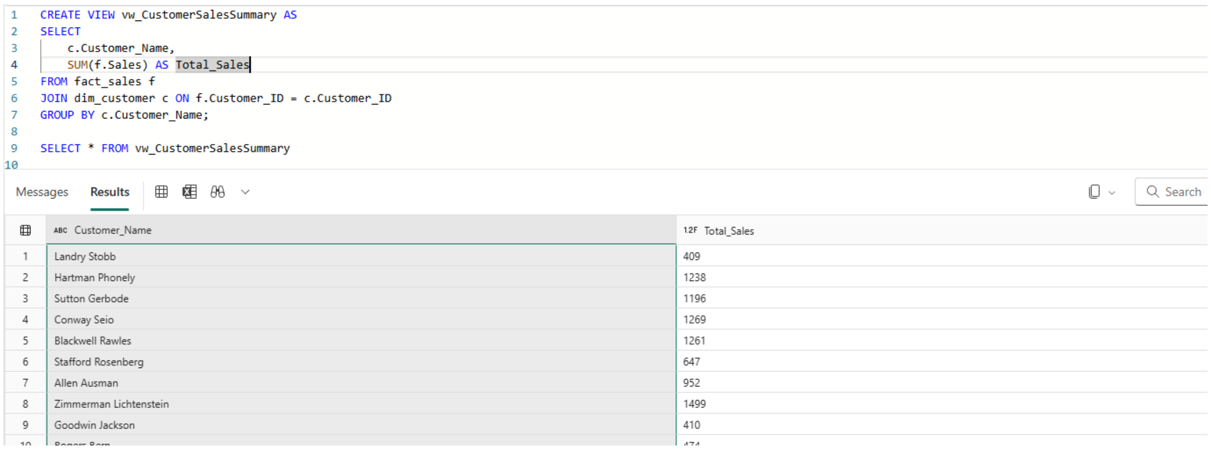
- ✓ **Descriptive Views (Created using fact and dimension tables)**
- ✓



```sql
CREATE VIEW vw_ProductSalesByYear AS
SELECT
    ROUND(SUM(f.Sales), 2) AS Total_Sales,
    ROUND(SUM(f.Profit), 2) AS Total_Profit,
    ROUND(SUM(f.Quantity), 2) AS Total_Quantity,
    p.Product_Category,
    d.Year
FROM fact_sales f
JOIN dim_product p ON f.Product_ID = p.Product_ID
JOIN dim_date d ON f.Order_Date_Key = d.Date_Key
GROUP BY p.Product_Category,d.Year

SELECT * FROM vw_ProductSalesByYear
```

Messages Results

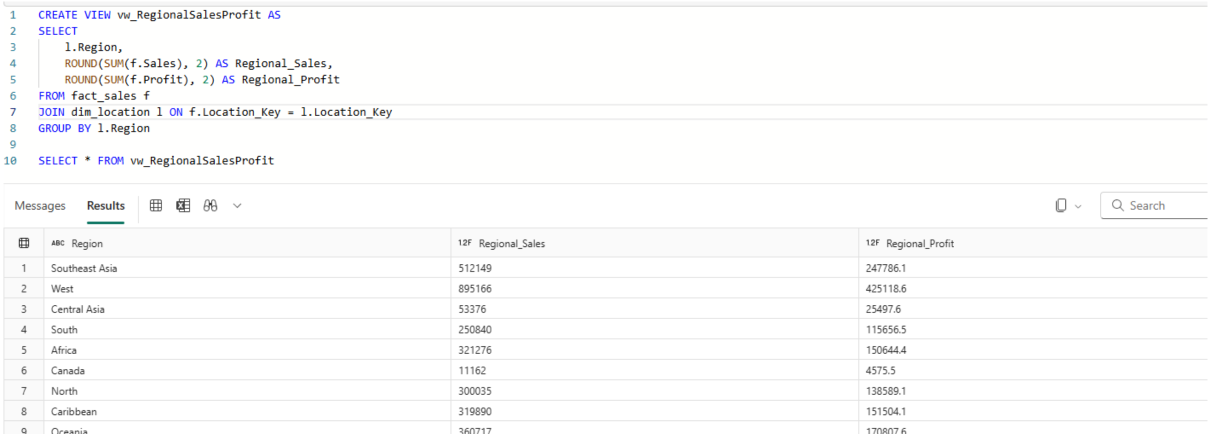
	Total_Sales	Total_Profit	Total_Quantity	Product_Category	Year
1	7487875	3285793.79	187163	Home & Furniture	2023
2	85130554	41121958.65	1447133	Fashion	2023
3	540308	224063.2	11399	Electronic	2023
4	5839969	2720706.21	111908	Auto & Accessories	2023



```sql
CREATE VIEW vw_CustomerSalesSummary AS
SELECT
    c.Customer_Name,
    SUM(f.Sales) AS Total_Sales
FROM fact_sales f
JOIN dim_customer c ON f.Customer_ID = c.Customer_ID
GROUP BY c.Customer_Name;

SELECT * FROM vw_CustomerSalesSummary
```

Messages Results

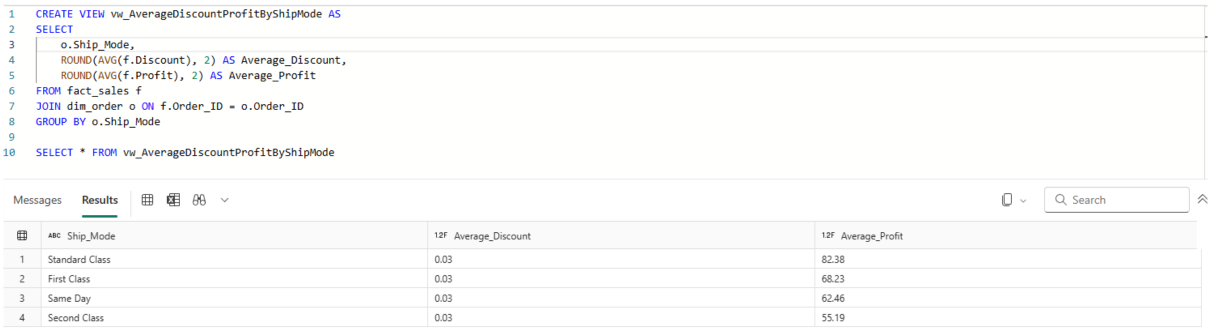
	Customer_Name	Total_Sales
1	Landry Stobb	409
2	Hartman Phonely	1238
3	Sutton Gerbode	1196
4	Conway Seio	1269
5	Blackwell Rawles	1261
6	Stafford Rosenberg	647
7	Allen Ausman	952
8	Zimmerman Lichtenstein	1499
9	Goodwin Jackson	410



```sql
CREATE VIEW vw_RegionalSalesProfit AS
SELECT
    l.Region,
    ROUND(SUM(f.Sales), 2) AS Regional_Sales,
    ROUND(SUM(f.Profit), 2) AS Regional_Profit
FROM fact_sales f
JOIN dim_location l ON f.Location_Key = l.Location_Key
GROUP BY l.Region

SELECT * FROM vw_RegionalSalesProfit
```

Messages Results

	Region	Regional_Sales	Regional_Profit
1	Southeast Asia	512149	247786.1
2	West	895166	425118.6
3	Central Asia	53376	25497.6
4	South	250840	115656.5
5	Africa	321276	150644.4
6	Canada	11162	4575.5
7	North	300035	138589.1
8	Caribbean	319890	151504.1
9	Oceania	360717	170807.6



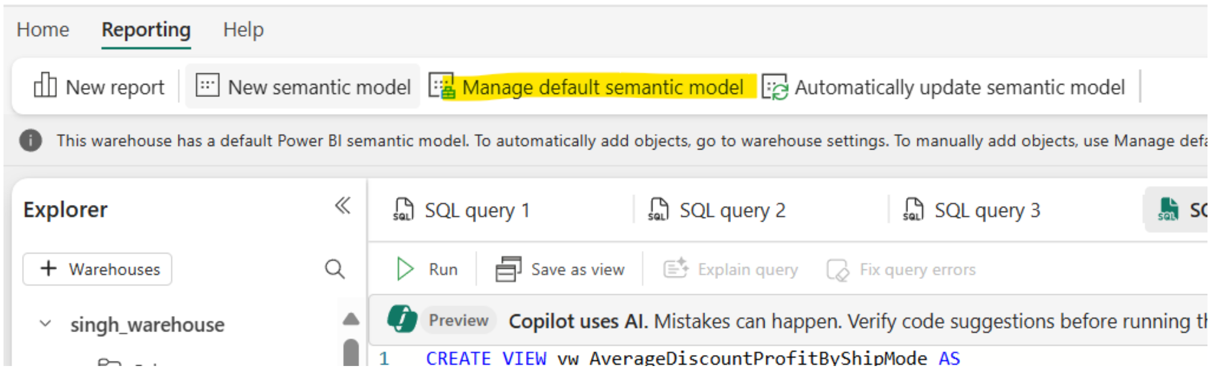
```sql
CREATE VIEW vw_AverageDiscountProfitByShipMode AS
SELECT
    o.Ship_Mode,
    ROUND(AVG(f.Discount), 2) AS Average_Discount,
    ROUND(AVG(f.Profit), 2) AS Average_Profit
FROM fact_sales f
JOIN dim_order o ON f.Order_ID = o.Order_ID
GROUP BY o.Ship_Mode

SELECT * FROM vw_AverageDiscountProfitByShipMode
```

Messages Results

	Ship_Mode	Average_Discount	Average_Profit
1	Standard Class	0.03	82.38
2	First Class	0.03	68.23
3	Same Day	0.03	62.46
4	Second Class	0.03	55.19

✓ **Power BI Report Creation**

– Click on Manage default semantic model and then click on ok.

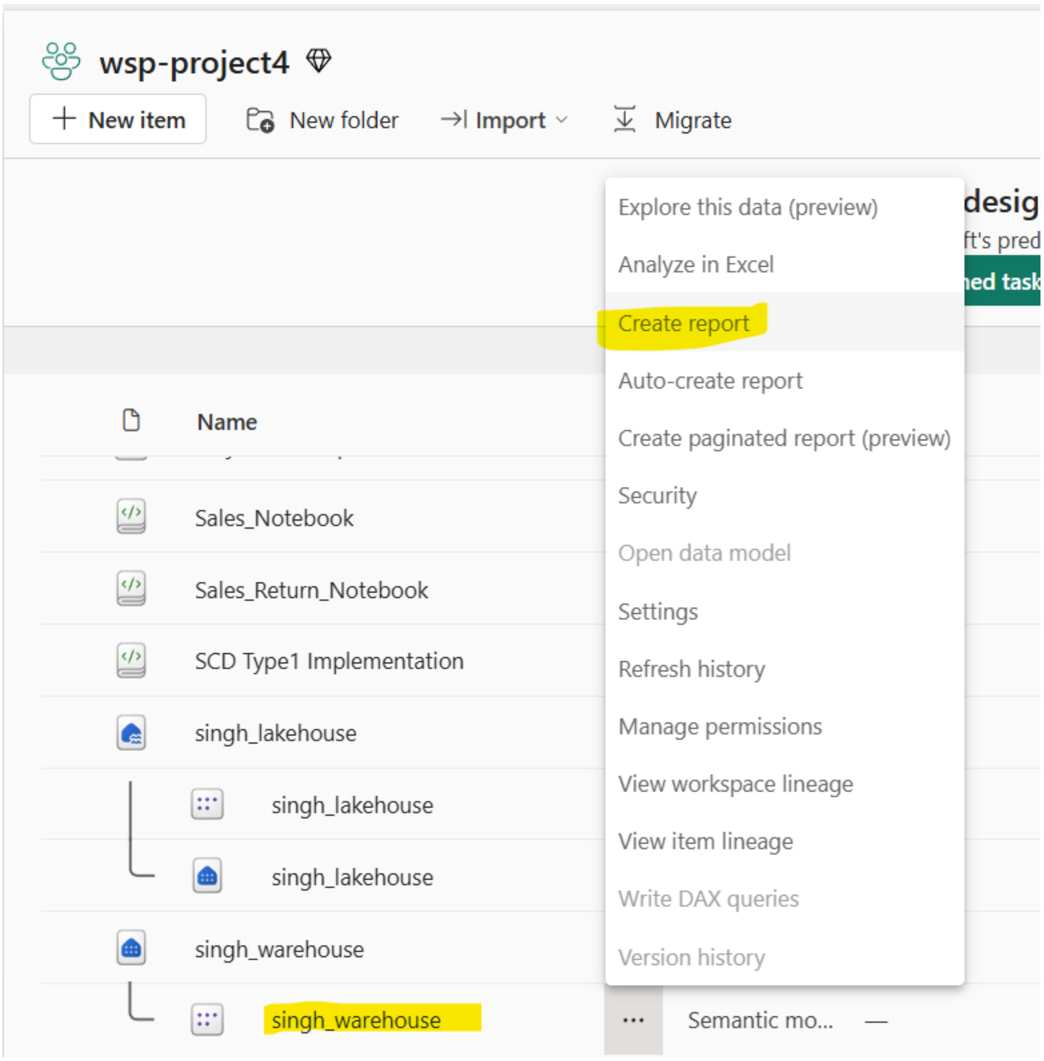


This warehouse has a default Power BI semantic model. To automatically add objects, go to warehouse settings. To manually add objects, use Manage defa

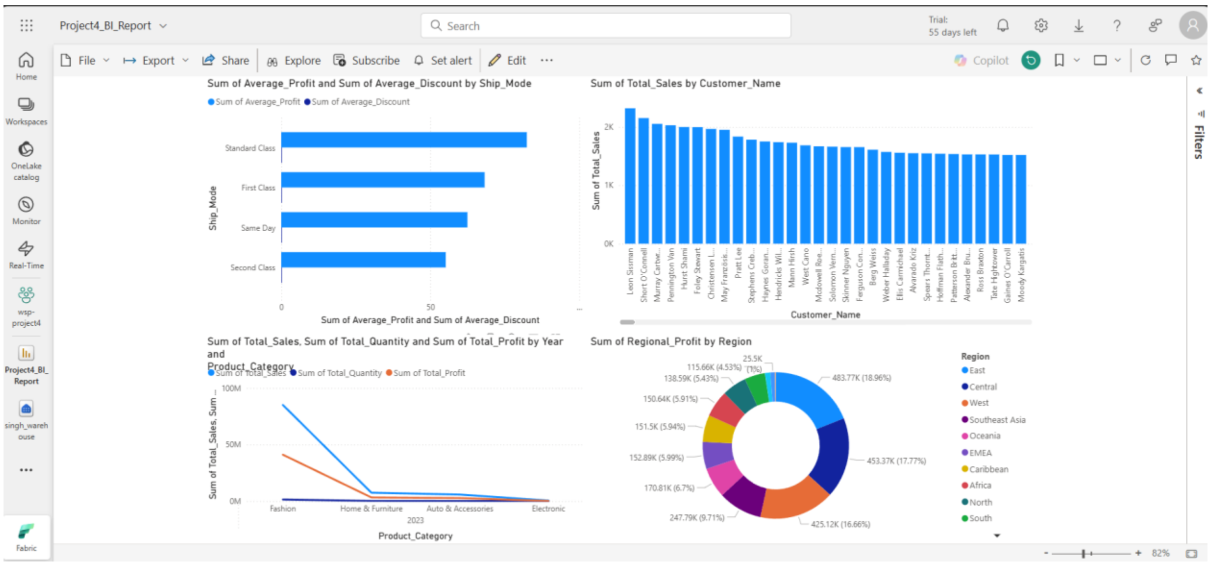
Copilot uses AI. Mistakes can happen. Verify code suggestions before running th

```
CREATE VIEW vw_AverageDiscountProfitByShipMode AS
```

– Go to Workspace, click on warehouse options and create new report option.



– Create Visuals using view data.


Project4_BI_Report
Search
Trial: 55 days left
Home
Workspaces
OneLake catalog
Monitor
Real-Time
wsp-project4
Project4_BI_Report
singh_warehouse
Fabric
File
Export
Share
Explore
Subscribe
Set alert
Edit
Copilot
Filters
Sum of Average_Profit and Sum of Average_Discount by Ship_Mode
Sum of Average_Profit
Sum of Average_Discount
Ship_Mode
Standard Class
First Class
Same Day
Second Class
0
50
Sum of Average_Profit and Sum of Average_Discount
Sum of Total_Sales by Customer_Name
Sum of Total_Sales
2K
1K
0K
Leon Sissman
Short O'Connell
Murray Cartw...
Pennington Van
Hurst Shami
Foley Stewart
Christensen L...
May Französis...
Pratt Lee
Stephens Creb...
Haynes Goran...
Hendricks Wil...
Mann Hirsh
West Cano
Mcdowell Roe...
Solomon Vern...
Skinner Nguyen
Ferguson Con...
Berg Weiss
Weber Halladay
Ellis Carmichael
Alvarado Kriz
Spears Thornt...
Hoffman Flath...
Patterson Britt...
Alexander Bru...
Ross Braxton
Tate Hightower
Gaines O'Carroll
Moody Kargatis
Customer_Name
Sum of Total_Sales, Sum of Total_Quantity and Sum of Total_Profit by Year and Product_Category
Sum of Total_Sales
Sum of Total_Quantity
Sum of Total_Profit
Sum of Total_Sales, Sum ...
100M
50M
0M
Fashion
Home & Furniture
Auto & Accessories
Electronic
2023
Product_Category
Sum of Regional_Profit by Region
25.5K (1%)
115.66K (4.53%)
138.59K (5.43%)
150.64K (5.91%)
151.5K (5.94%)
152.89K (5.99%)
170.81K (6.7%)
247.79K (9.71%)
425.12K (16.66%)
453.37K (17.77%)
483.77K (18.96%)
Region
East
Central
West
Southeast Asia
Oceania
EMEA
Caribbean
Africa
North
South
82%