

## **Project: Incremental Data Loading & Automated Notifications using Fabric**

### **Problem Statement:**

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.

### **Project Objective:**

To build an end-to-end data pipeline on Microsoft Fabric that:

1. Ingests structured data from on-premises environments into a Fabric Lakehouse using the On-Prem Gateway
2. Utilizes the [AI Bank Dataset](#) as the source
3. Implements Dataflow Gen 1 to join tables, remove duplicates, and clean data
4. Loads the cleansed data into a Fabric Warehouse
5. Applies Slowly Changing Dimension (SCD) Type 1 logic using Fabric Notebooks and writes the results into separate warehouse tables
6. Schedules and monitors the pipeline, sending an automated email notification (via Outlook or Gmail) upon successful pipeline completion

### **Tools & Technologies:**

- Microsoft Fabric
- On-Premises Data Gateway
- Fabric Lakehouse and Warehouse
- Fabric Dataflow Gen 1 / Gen 2
- Fabric Notebook
- Email Notification Task (in-built)
- Azure Key Vault (optional for secure credential management)
- Draw.io / Visio for architecture diagram

## Bronze Layer:

To begin with, we'll have to ingest data from on-premise environments into a Fabric Lakehouse using On-Prem Gateway.

To achieve this task, let's utilize an azure virtual machine. Suppose there is some data on that virtual machine (on-prem environment) which we need to ingest into the fabric lakehouse using on-prem gateway.

In the virtual machine, download the [AI Bank Dataset](#) and [On-Prem Gateway](#) (download standard mode).

Install the on-prem gateway on the virtual machine. Once it completes, it'll ask for an email address to use the gateway with – use your fabric account email – and sign in. Consequently, it'll ask to give a name to the gateway and set a recovery key (password) for the gateway.

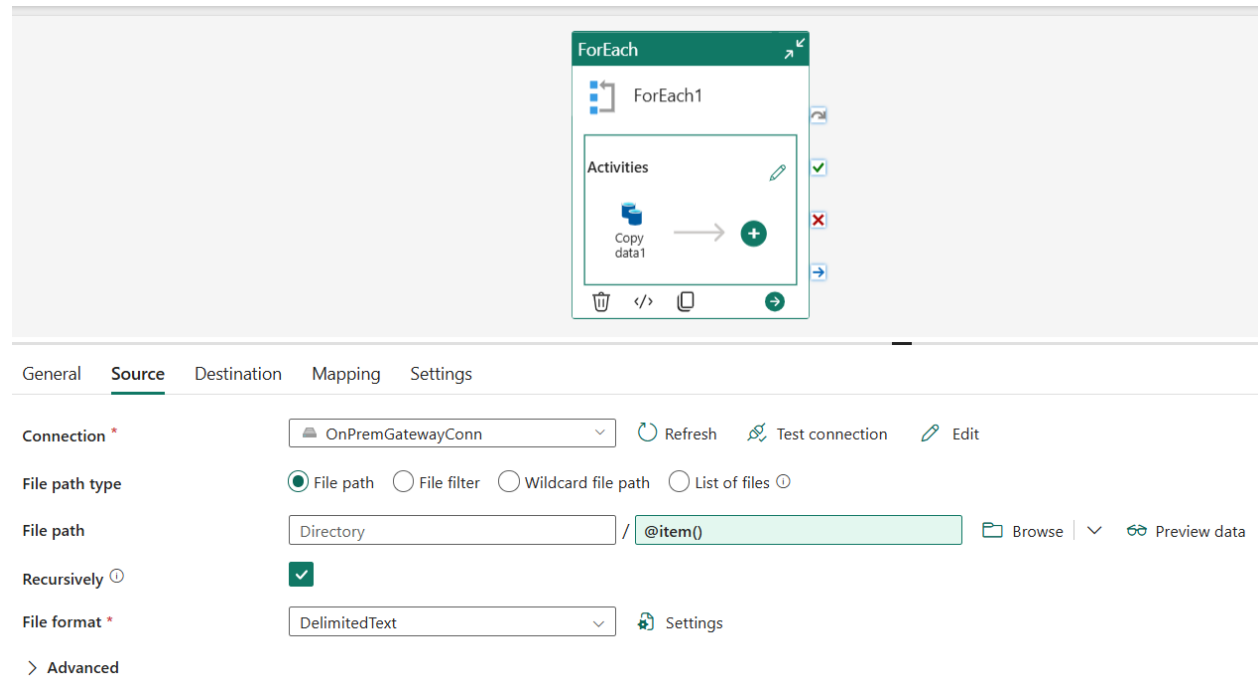
Now, we'll create a pipeline and use a copy data activity in fabric lakehouse along with a for-each loop to get all the files of the dataset from on-prem.

The image shows a Windows File Explorer window and the Microsoft Fabric portal interface.

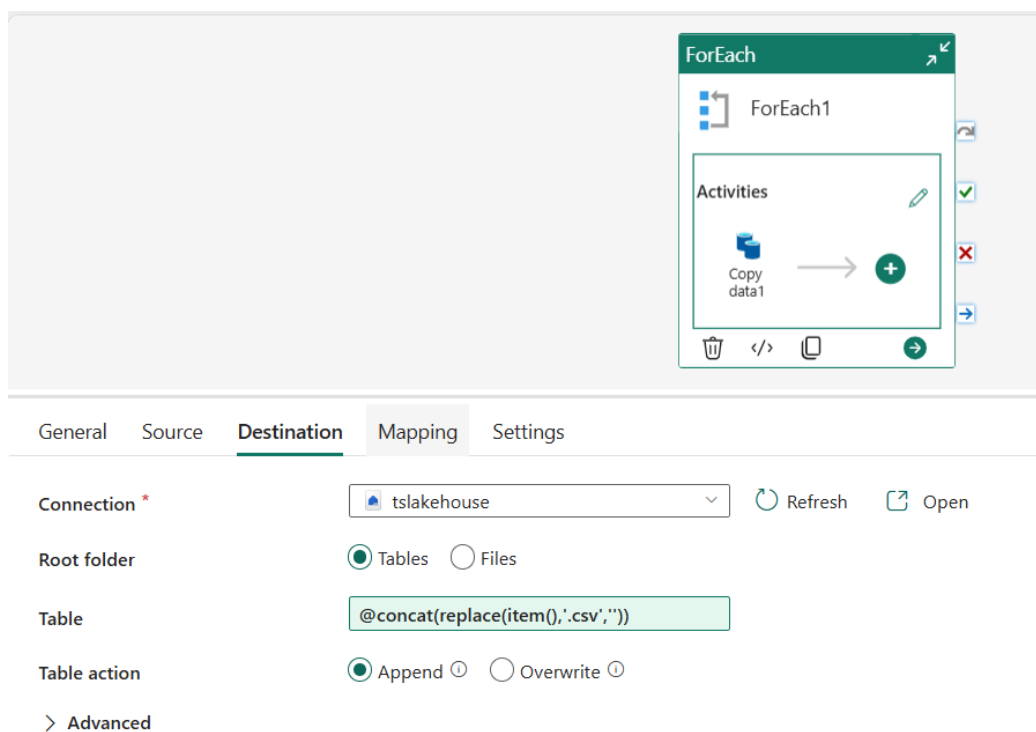
**File Explorer:** The window displays the 'dataset' folder in 'Downloads'. It contains five CSV files: accounts.csv (3 KB), customers.csv (5 KB), loan\_payments.csv (3 KB), loans.csv (3 KB), and transactions.csv (4 KB). All files were modified on 5/4/2025 at 10:13 PM.

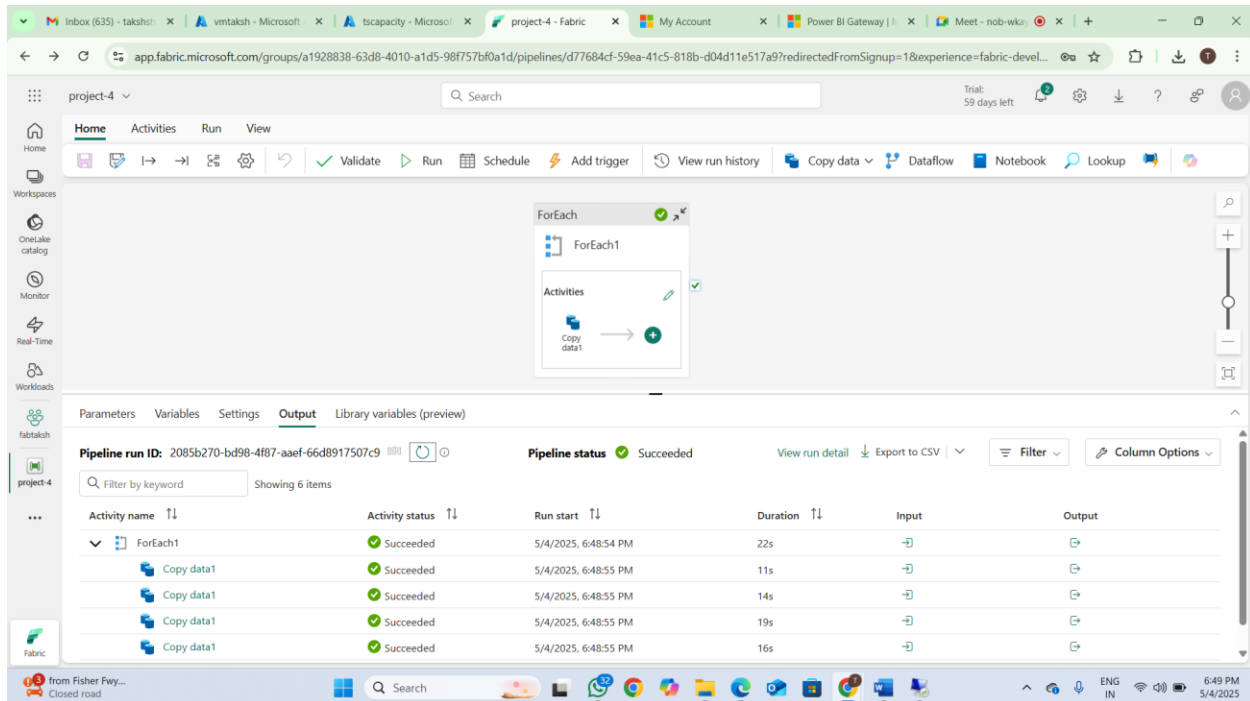
**Microsoft Fabric Portal:** The portal shows a pipeline named 'project-4'. The 'Activities' tab is active, displaying a 'ForEach' loop with a 'Copy data1' activity. The 'Pipeline expression builder' is open, showing the expression: `@createArray('accounts.csv','customers.csv','loan_payments.csv','loans.csv','transactions.csv')`. The 'Activity outputs' tab shows the output of the 'Copy data1' activity as 'Copy data1 activity output'.

For settings section of for-each loop, under “items” we provide an array of the names of the dataset files as the loop expects an array input.



For OnPremGatewayConn connection, we have to provide the path of folder in virtual machine where the dataset is located.





The screenshot shows the Microsoft Fabric portal interface. At the top, there's a navigation bar with tabs for Home, Activities, Run, and View. Below this, a search bar and a 'Trial: 59 days left' indicator are visible. The main content area displays a pipeline run for 'ForEach1'. The pipeline status is 'Succeeded'. Below the status, there's a table showing the run details:

| Activity name | Activity status | Run start            | Duration | Input | Output |
|---------------|-----------------|----------------------|----------|-------|--------|
| ForEach1      | Succeeded       | 5/4/2025, 6:48:54 PM | 22s      |       |        |
| Copy data1    | Succeeded       | 5/4/2025, 6:48:55 PM | 11s      |       |        |
| Copy data1    | Succeeded       | 5/4/2025, 6:48:55 PM | 14s      |       |        |
| Copy data1    | Succeeded       | 5/4/2025, 6:48:55 PM | 19s      |       |        |
| Copy data1    | Succeeded       | 5/4/2025, 6:48:55 PM | 16s      |       |        |

## Pipeline ran successfully

Once the pipeline runs successfully, you'll be able to see the tables loaded in fabric lakehouse.

## Silver Layer:

We'll use dataflow gen2 to clean data (remove duplicates, replace nulls). Let's import the source tables in the dataflow. In dataflow gen2, go to **getdata->more->lakehouse** as our source will be the files which we just ingested from on-prem (virtual machine) to lakehouse.



The screenshot shows the 'Get data' dialog in Microsoft Fabric. The 'Choose data' section is active, displaying a list of data sources. The 'Lakehouse' folder is selected, and the 'fabtaksh' sub-folder is highlighted. The list includes:

- Lakehouse [2]
- fabtaksh [2]
  - StagingLakehouseForDataflows...
  - tslakehouse
  - My workspace

At the bottom of the dialog, there are 'Back', 'Cancel', and 'Create' buttons.

Select the tables which we created from the dataset files.

Get data  
Choose data

Search

Display options

- ☐ queryinsights.long\_run...
- ☐ sys.managed\_delta\_ta...
- ☐ sys.managed\_delta\_ta...
- ☐ sys.managed\_delta\_ta...
- ☐ sys.managed\_delta\_ta...
- ☐ sys.sys\_dw\_checkpoint...
- ☐ sys.sys\_dw\_manifest\_fil...
- ☐ sys.sys\_dw\_physical\_ta...
- ☐ sys.sys\_dw\_schemas
- ☒ accounts
- ☒ customers
- ☒ loan\_payments
- ☒ loans
- ☒ transactions

My workspace

transactions

| transaction_id | account_id | transaction_date | transaction_amount | transaction_type |
|----------------|------------|------------------|--------------------|------------------|
| 1              | 45         | 01/01/2024       | 100.5              | Deposit          |
| 2              | 12         | 02/01/2024       | 200.75             | Withdrawal       |
| 3              | 78         | 03/01/2024       | 150                | Deposit          |
| 4              | 34         | 04/01/2024       | 300.25             | Withdrawal       |
| 5              | 56         | 05/01/2024       | 250                | Deposit          |
| 6              | 23         | 06/01/2024       | 175                | Withdrawal       |
| 7              | 89         | 07/01/2024       | 225.5              | Deposit          |
| 8              | 67         | 08/01/2024       | 275.75             | Withdrawal       |
| 9              | 14         | 09/01/2024       | 325                | Deposit          |
| 10             | 92         | 10/01/2024       | 375.25             | Withdrawal       |
| 11             | 3          | 11/01/2024       | 100.5              | Deposit          |
| 12             | 81         | 12/01/2024       | 200.75             | Withdrawal       |
| 13             | 29         | 13/01/2024       | 150                | Deposit          |
| 14             | 64         | 14/01/2024       | 300.25             | Withdrawal       |
| 15             | 47         | 15/01/2024       | 250                | Deposit          |
| 16             | 18         | 16/01/2024       | 175                | Withdrawal       |
| 17             | 99         | 17/01/2024       | 225.5              | Deposit          |
| 18             | 5          | 18/01/2024       | 275.75             | Withdrawal       |
| 19             | 76         | 19/01/2024       | 325                | Deposit          |
| 20             | 21         | 20/01/2024       | 375.25             | Withdrawal       |

Back Cancel Create

app.fabric.microsoft.com/groups/a1928838-63d8-4010-a1d5-98f757bf0a1d/dataflows-gen2/32830d8e-2d58-494d-8d32-9596adac02c5?redirectedFromSignup=1&experience=fabri...

Dataflow 1

Power Query Draft saved

Search

Home Transform Add column View Help

Queries [5]

- accounts
- customers
- loan\_payments
- loans
- transactions

#"Navigation 2"([Id = "accounts", ItemKind = "Table"])[Data]

| account_id | customer_id | account_type | balance |
|------------|-------------|--------------|---------|
| 1          | 1           | Savings      | 1000.5  |
| 2          | 2           | Checking     | 2500.75 |
| 3          | 3           | Savings      | 1500    |
| 4          | 4           | Checking     | 3000.25 |
| 5          | 5           | Savings      | 500     |
| 6          | 6           | Checking     | 1200.5  |
| 7          | 7           | Savings      | 800.75  |
| 8          | 8           | Checking     | 2200    |
| 9          | 9           | Savings      | 900.25  |
| 10         | 10          | Checking     | 1800.5  |
| 11         | 11          | Savings      | 1100.75 |
| 12         | 12          | Checking     | 2700    |
| 13         | 13          | Savings      | 1300.25 |
| 14         | 14          | Checking     | 3200.5  |
| 15         | 15          | Savings      | 700.75  |
| 16         | 16          | Checking     | 1400    |
| 17         | 17          | Savings      | 600.25  |
| 18         | 18          | Checking     | 1600.5  |
| 19         | 19          | Savings      | 400.75  |
| 20         | 20          | Checking     | 2000    |
| 21         | 21          | Savings      | 300.25  |
| 22         | 22          | Checking     | 2400.5  |
| 23         | 23          | Savings      | 200.75  |

Completed (2.75 s) Columns: 4 Rows: 99+

Add default destination...

Query settings

Properties

Name

accounts

Applied steps

- Source
- Navigation 1
- Navigation 2
- Navigation 3

Data destination

No data destination

Step

11°C Rain

Search

ENG IN

7:02 PM 5/4/2025

All the tables have been created in the dataflow. Follow the images provided below to clean the data for customers table and repeat the process for all the other tables.

Power Query Editor interface showing a table with columns: customer\_id, first\_name, last\_name, address, city, state, zip. The table contains 23 rows of customer data. The 'Transform' tab is active, and the 'Remove duplicates' option is highlighted in the ribbon. The 'Query settings' pane on the right shows the query name 'customers' and the applied steps: Source, Navigation 1, Navigation 2, and Navigation 3. The status bar at the bottom indicates 'Completed (1.20 s) Columns: 7 Rows: 87'.

Power Query Editor interface showing the same table after applying the 'Remove duplicates' step. The 'Transform' tab is active, and the 'Remove duplicates' option is highlighted in the ribbon. The 'Query settings' pane on the right shows the query name 'customers' and the applied steps: Source, Navigation 1, Navigation 2, and 'Removed duplicates'. The status bar at the bottom indicates 'Completed (1.00 s) Columns: 7 Rows: 87'.

## Replace values ?

Replace one value with another in the selected columns.

Value to find

null

Replace with

0

OK

Cancel


Next, we want to store the cleaned data into fabric warehouse. Follow the images provided below and do the steps for all the cleaned tables.

The screenshot displays the Microsoft Fabric Dataflow 1 interface. The main area shows a Power Query table with columns: 1.2 account\_id, 1.2 customer\_id, 1.2 account\_type, 1.2 balance, and customers. The table contains 23 rows of data. The 'Query settings' panel on the right shows the 'Properties' tab with the name 'Merge' and the 'Applied steps' list. The 'Default destination' is set to 'No default destination available'. The 'New destination' list includes Lakehouse, Warehouse, SQL database, Azure SQL database, and Azure Data Explorer (Kusto). The 'Data destination' is set to 'No data destination'.


| 1.2 account_id | 1.2 customer_id | 1.2 account_type | 1.2 balance     | customers |
|----------------|-----------------|------------------|-----------------|-----------|
| 1              | 1               | 45 Savings       | 1000.5 (Table)  |           |
| 2              | 2               | 12 Checking      | 2500.75 (Table) |           |
| 3              | 3               | 78 Savings       | 1500 (Table)    |           |
| 4              | 4               | 34 Checking      | 3000.25 (Table) |           |
| 5              | 5               | 56 Savings       | 500 (Table)     |           |
| 6              | 6               | 23 Checking      | 1200.5 (Table)  |           |
| 7              | 7               | 89 Savings       | 800.75 (Table)  |           |
| 8              | 8               | 67 Checking      | 2200 (Table)    |           |
| 9              | 9               | 14 Savings       | 900.25 (Table)  |           |
| 10             | 10              | 92 Checking      | 1800.5 (Table)  |           |
| 11             | 11              | 3 Savings        | 1100.75 (Table) |           |
| 12             | 12              | 81 Checking      | 2700 (Table)    |           |
| 13             | 13              | 29 Savings       | 1300.25 (Table) |           |
| 14             | 14              | 64 Checking      | 3200.5 (Table)  |           |
| 15             | 15              | 47 Savings       | 700.75 (Table)  |           |
| 16             | 16              | 18 Checking      | 1400 (Table)    |           |
| 17             | 17              | 99 Savings       | 600.25 (Table)  |           |
| 18             | 18              | 5 Checking       | 1600.5 (Table)  |           |
| 19             | 19              | 76 Savings       | 400.75 (Table)  |           |
| 20             | 20              | 21 Checking      | 2000 (Table)    |           |
| 21             | 21              | 53 Savings       | 300.25 (Table)  |           |
| 22             | 22              | 37 Checking      | 2400.5 (Table)  |           |
| 23             | 23              | 88 Savings       | 200.75 (Table)  |           |

Data destination

### Connect to data destination

**Warehouse**  
Microsoft Fabric

#### Connection credentials

Connection  
WarehouseConn tsfabric (none) 



Authentication kind: Organizational account [Edit connection](#)



Back


CancelNext

Data destination

### Choose destination target

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

☒ New table  ☐ Existing table 

Display options   
Warehouse [2]  
StagingWarehouseForDataf...  
tswarehouse



 A new table will be created in tswarehouse


Table name \*  
 customers

Back



CancelNext

Data destination



### Choose destination settings

☐ Use automatic settings 

#### Update method

☐ Append  ☒ Replace 

#### Schema options on publish

☐ Dynamic schema  ☒ Fixed schema 

#### Column mapping

| <input checked="" type="checkbox"/> Source      | Source type        | Destination | Destination type |
|---|--------------------|-------------|------------------|
| <input checked="" type="checkbox"/> customer_id | 1,2 Decimal number | customer_id | Decimal number   |
| <input checked="" type="checkbox"/> first_name  | Text               | first_name  | Text             |
| <input checked="" type="checkbox"/> last_name   | Text               | last_name   | Text             |
| <input checked="" type="checkbox"/> address     | Text               | address     | Text             |

Back

CancelSave settings



### Gold Layer:

Visit: <https://learn.microsoft.com/en-us/fabric/data-engineering/spark-data-warehouse-connector?tabs=pyspark>

This website provides us the code to read data in a fabric notebook from a table/view in spark dataframe.

Now, we'll use fabric notebook to implement scdtype-1 logic on 3 tables from the cleaned data. The implementation for accounts table has been shown below – same logic can be applied to the other 2 tables of choice.

```
1 # Welcome to your new notebook
2 # Type here in the cell editor to add code!
3 import com.microsoft.spark.fabric
4 from com.microsoft.spark.fabric.Constants import Constants
```

✓ - Session ready in 10 sec 723 ms. Command executed in 383 ms by Taksh-fabric-417 on 8:41:41 PM, 5/05/25

## Accounts SCD-1

```
1 # Step 1: Read from Fabric Warehouse table
2 df = spark.read.synapsesql("tswarehouse.dbo.accounts")
3 #df.show()
```

✓ - Command executed in 16 sec 209 ms by Taksh-fabric-417 on 8:42:22 PM, 5/05/25

```
+-----+-----+-----+-----+
|customer_id|balance|account_type|account_id|
+-----+-----+-----+-----+
|      45| 1000.5|    Savings|        1.0|
|      12| 2500.75|   Checking|        2.0|
|      78| 1500.0|    Savings|        3.0|
|      34| 3000.25|   Checking|        4.0|
|      56|   500.0|    Savings|        5.0|
```

```

1  create_table_query = """
2  CREATE TABLE if not exists Account_SCD (
3      account_id int,
4      customer_id int,
5      account_type STRING,
6      balance FLOAT,
7      hash_key BIGINT,
8      created_by STRING,
9      created_date TIMESTAMP,
10     updated_by STRING,
11     updated_date TIMESTAMP
12 )
13 USING DELTA
14 LOCATION 'Files/Gold_layer/Account_SCD'
15 """
16
17 # Execute
18 spark.sql(create_table_query)

```

✓ - Command executed in 13 sec 946 ms by Taksh-fabric-417 on 8:42:51 PM, 5/05/25

DataFrame[ ]

```

1  from pyspark.sql.functions import *
2  df_src1= df.withColumn("hash_key",crc32(concat(*df.columns)))
3  #display(df_src1)

```

✓ - Command executed in 2 sec 357 ms by Taksh-fabric-417 on 8:43:06 PM, 5/05/25



 Table  New chart

Table view

|  | 123 customer... | 12 balance | ABC account_... | 12 account_id | 12L hash_key |
|---|-----------------|------------|-----------------|---------------|--------------|
| 1   | 45              | 1000.5     | Savings         | 1.0           | 1081396551   |
| 2   | 12              | 2500.75    | Checking        | 2.0           | 1322625823   |
| 3   | 78              | 1500.0     | Savings         | 3.0           | 4179129744   |
| 4   | 34              | 3000.25    | Checking        | 4.0           | 463029068    |
| 5   | 56              | 500.0      | Savings         | 5.0           | 787317497    |
| 6   | 23              | 1200.5     | Checking        | 6.0           | 2250105050   |

```

1  from delta.tables import DeltaTable
2
3  target_path = "Files/Gold_layer/Account_SCD"
4  delta_target = DeltaTable.forPath(spark, target_path)
5  #delta_target.toDF().show()

```

✓ - Command executed in 4 sec 838 ms by Taksh-fabric-417 on 8:44:18 PM, 5/05/25

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|account_id|customer_id|account_type|balance|hash_key|created_by|created_date|updated_by|updated_date|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

1  df_src1=df_src1.alias("src").join(delta_target.toDF().alias("tgt"),((col("src.account_id")==col("tgt.account_id"))&(col("src.hash_key")==col("tgt.hash_key"))),"anti").select(col("src.*"))
2  #df_src1.show()

```

✓ - Command executed in 2 sec 502 ms by Taksh-fabric-417 on 8:45:24 PM, 5/05/25

```

+-----+-----+-----+-----+-----+
|customer_id|balance|account_type|account_id| hash_key|
+-----+-----+-----+-----+-----+
|          45| 1000.5|    Savings|          1.0|1081396551|
|          12| 2500.75|   Checking|          2.0|1322625823|
|          78| 1500.0|    Savings|          3.0|4179129744|
|          34| 3000.25|   Checking|          4.0| 463029068|
|          56|   500.0|    Savings|          5.0| 787317497|

```

```
df_src1=df_src1.alias("src").join(delta_target.toDF().alias("tgt"),((col("src.account_id")==col("tgt.account_id"))&(col("src.hash_key")==col("tgt.hash_key"))),"anti").select(col("src.*"))
```

```
#df_src1.show()
```

```

1 from pyspark.sql.functions import col
2
3 delta_target.alias("tgt").merge(df_src1.alias("src"),"tgt.account_id = src.account_id")\
4     .whenMatchedUpdate(set={"tgt.account_id":"src.account_id","tgt.customer_id":"src.customer_id","tgt.account_ty"
5     .whenNotMatchedInsert(values={"tgt.account_id":"src.account_id","tgt.customer_id":"src.customer_id","tgt.
6
7 #display(spark.read.format("delta").option("header","true").load(target_path))

```

✓ - Command executed in 7 sec 989 ms by Taksh-fabric-417 on 8:46:06 PM, 5/05/25

Table

+ New chart

9 column

Table view

Download

Search

|   | 123 account_id | 123 customer... | ABC account_... | 12F balance | 12L hash_key | ABC created_by | created_...  | ABC updated_... | updated_...  |  |
|---|----------------|-----------------|-----------------|-------------|--------------|----------------|--------------|-----------------|--------------|--|
| 1 | 1              | 45              | Savings         | 1000.5      | 1081396551   | databricks     | 2025-05-0... | databricks      | 2025-05-0... |  |
| 2 | 2              | 12              | Checking        | 2500.75     | 1322625823   | databricks     | 2025-05-0... | databricks      | 2025-05-0... |  |
| 3 | 3              | 78              | Savings         | 1500.0      | 4179129744   | databricks     | 2025-05-0... | databricks      | 2025-05-0... |  |
| 4 | 4              | 34              | Checking        | 3000.25     | 463029068    | databricks     | 2025-05-0... | databricks      | 2025-05-0... |  |
| 5 | 5              | 56              | Savings         | 500.0       | 787317497    | databricks     | 2025-05-0... | databricks      | 2025-05-0... |  |

```
from pyspark.sql.functions import col
```

```
delta_target.alias("tgt").merge(df_src1.alias("src"),"tgt.account_id = src.account_id")\
```

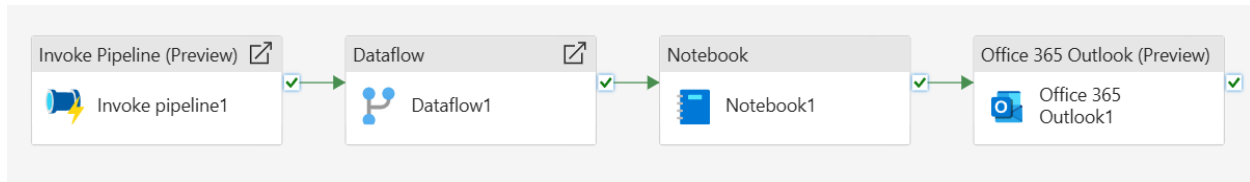
```
    .whenMatchedUpdate(set={"tgt.account_id":"src.account_id","tgt.customer_id":"src.customer_id","tgt.account_type":"src.account_type","tgt.balance":"src.balance","tgt.hash_key":"src.hash_key","tgt.updated_date":current_timestamp(),"tgt.updated_by":lit("databricks_Updated")})\
```

```
    .whenNotMatchedInsert(values={"tgt.account_id":"src.account_id","tgt.customer_id":"src.customer_id","tgt.account_type":"src.account_type","tgt.balance":"src.balance","tgt.hash_key":"src.hash_key","tgt.created_date":current_timestamp(),"tgt.created_by":lit("databricks"),"tgt.updated_date":current_timestamp(),"tgt.updated_by":lit("databricks")}).execute()
```

```
#display(spark.read.format("delta").option("header","true").load(target_path))
```

After coding scdtype-1 for the desired 3 cleaned data tables, test them using day1 (original dataset) and day2 (test) data.

## Creating master pipeline:



Open a new pipeline in fabric and get an invoke pipeline activity – this will invoke the pipeline which we created for bronze layer.

General
Settings

*i* Currently Pipeline return value is only supported with ADF & Synapse pipelines. To fetch pipeline return value for Fabric pipelines, please use Invol

Type
☒ Fabric
☐ Azure Data Factory
☐ Synapse

Connection \* *i*

FabricDataPipelines tsfabric

Refresh
Edit

Workspace \*

fabtaksh

Refresh

Pipeline \*

project-4

Refresh
Open
New

Parameters

Similarly, dataflow activity will be used to invoke the dataflow we created for silver layer and notebook activity will invoke the notebook we created for gold layer.

General
Settings

Workspace \*

fabtaksh

Refresh

Dataflow \*

Dataflow 1

Refresh
Open
New

> Dataflow parameters

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 work

Workspace \*

fabtaksh

Refresh

Notebook \*

Project-4

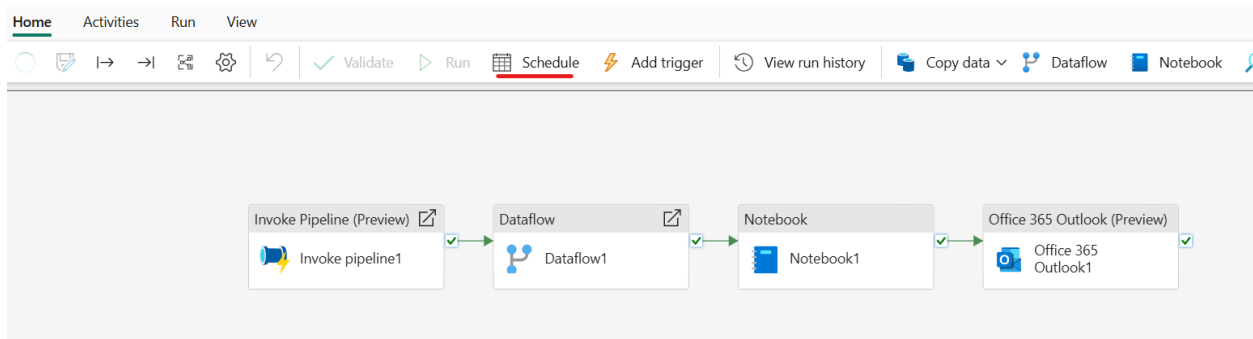
Refresh
Open
New

> Base parameters

> Advanced settings

Finally, as per the master pipeline image provided above, we'll connect notebook activity to an office 365 outlook (preview) activity which is basically responsible to send out email notifications upon successful runs of pipeline. First, you'll have to sign in to your outlook account and then under "settings", fill in the details as per your requirement.

Next, schedule the pipeline.



The 'Schedule' configuration panel for the 'pl\_master' data pipeline is shown. It includes the following settings:

- Schedule:** On (radio button selected)
- Repeat:** Daily (dropdown menu)
- Time:** 08:00 (time picker)
- Start date and time:** 04-05-2025 (calendar icon)
- End date and time:** 06-05-2025 (calendar icon)
- Time zone:** (UTC-05:00) Eastern Time (US and Canada) (dropdown menu)
- Buttons:** Apply, Discard