BOOTCAMP PROJECT 4

Project Title: Incremental Data Loading and Automated Notifications using Microsoft 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 Architecture Diagram:

## Tools and Technology used :

• Microsoft Fabric

• On-Premises Data Gateway

• Fabric Lakehouse and Warehouse

• Fabric Dataflow Gen 1

• Fabric Notebook

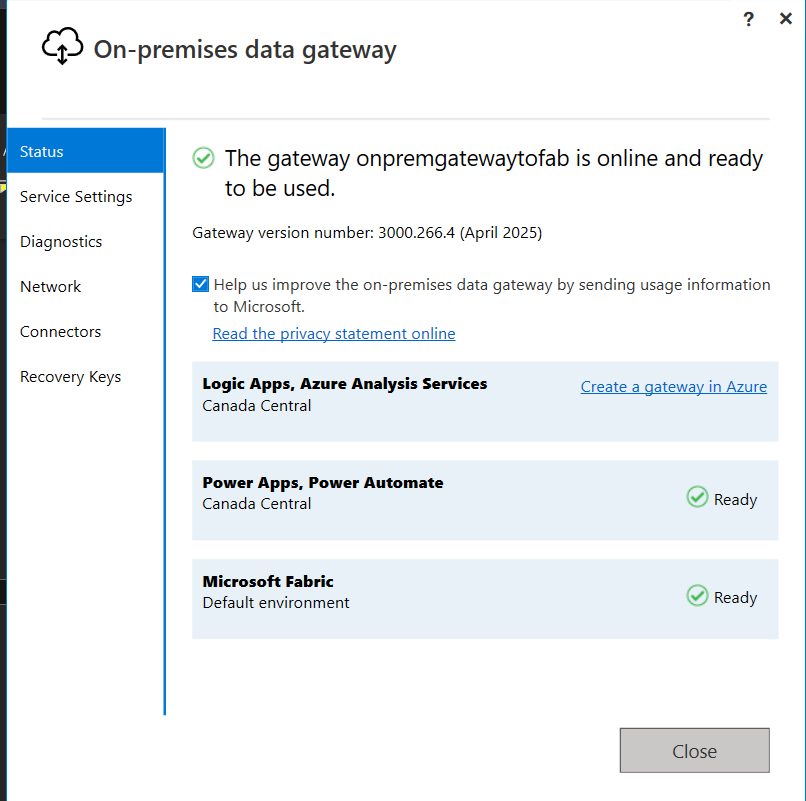
• Email Notification Task (in-built)

• Draw.io / Visio for architecture diagram

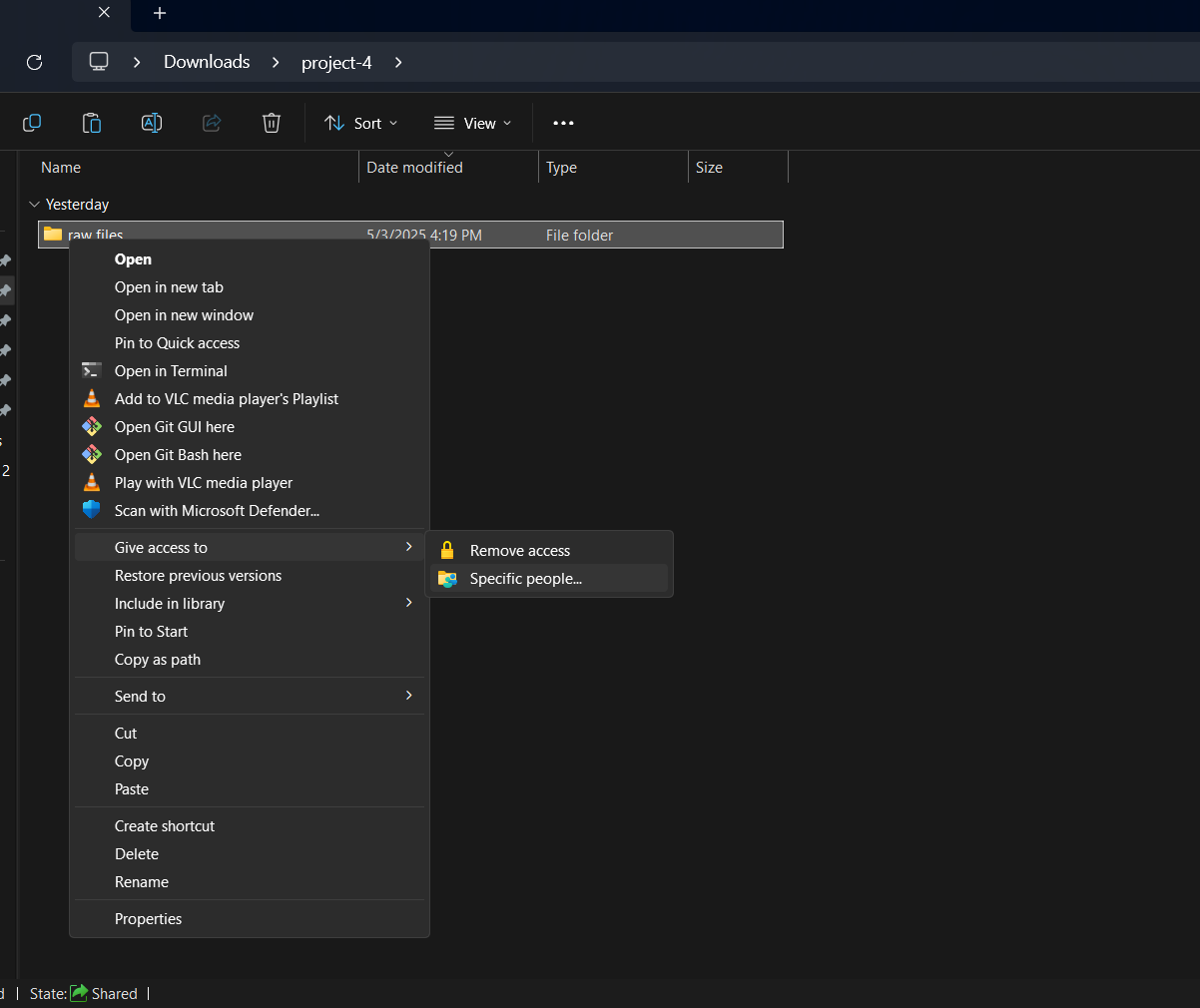
# PROJECT FLOW

## DATA INGESTION FROM LOCAL MACHINE TO FABRIC LAKEHOUSE USING ON-PREMISES GATEWAY

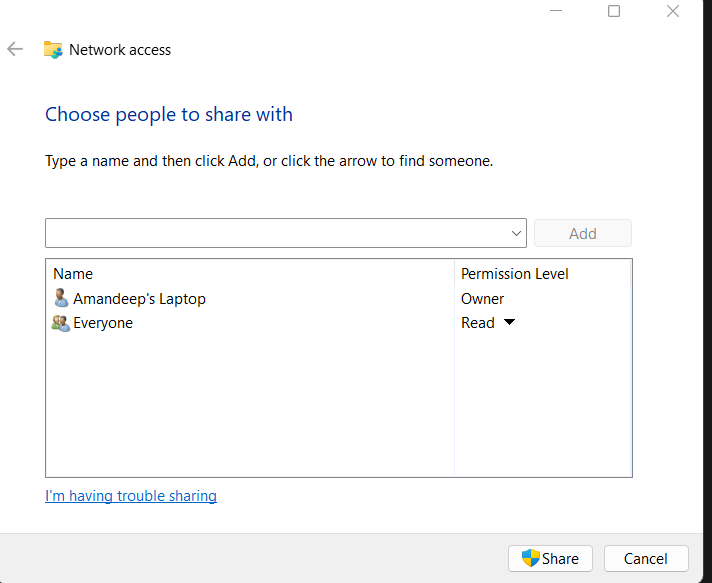
* In this phase, data gets transferred on local windows OS machine to fabric Lakehouse using On-premises gateway. Here are the steps I followed:



Install On-prem gateway and login with same credentials used in fabric. Setup up recouvery key in case of emergency.

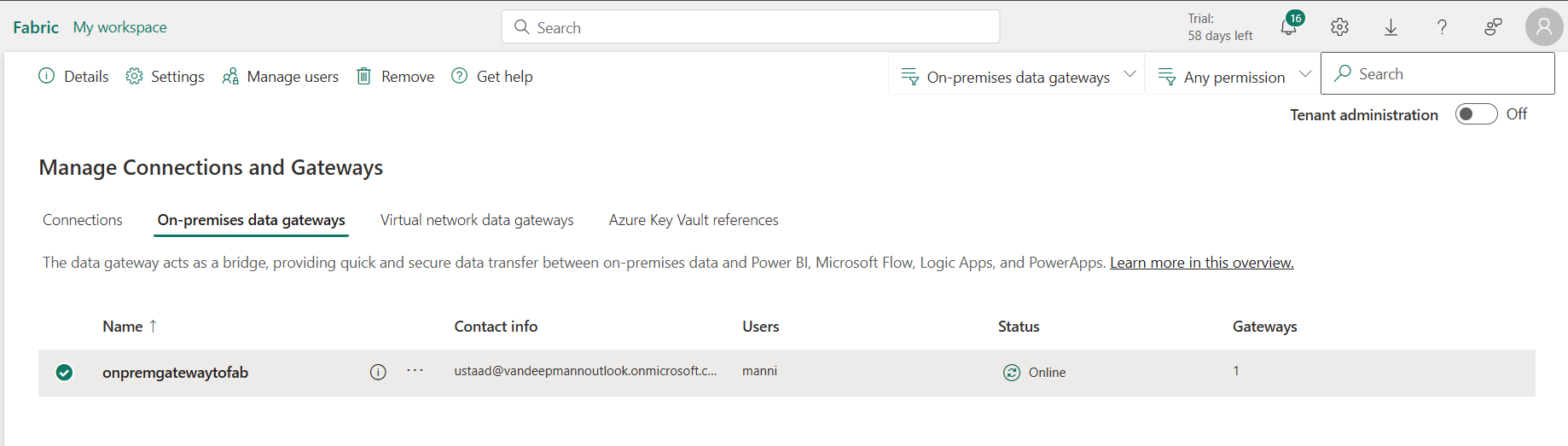


Go to the dir, which you want to share and give read only access to everyone



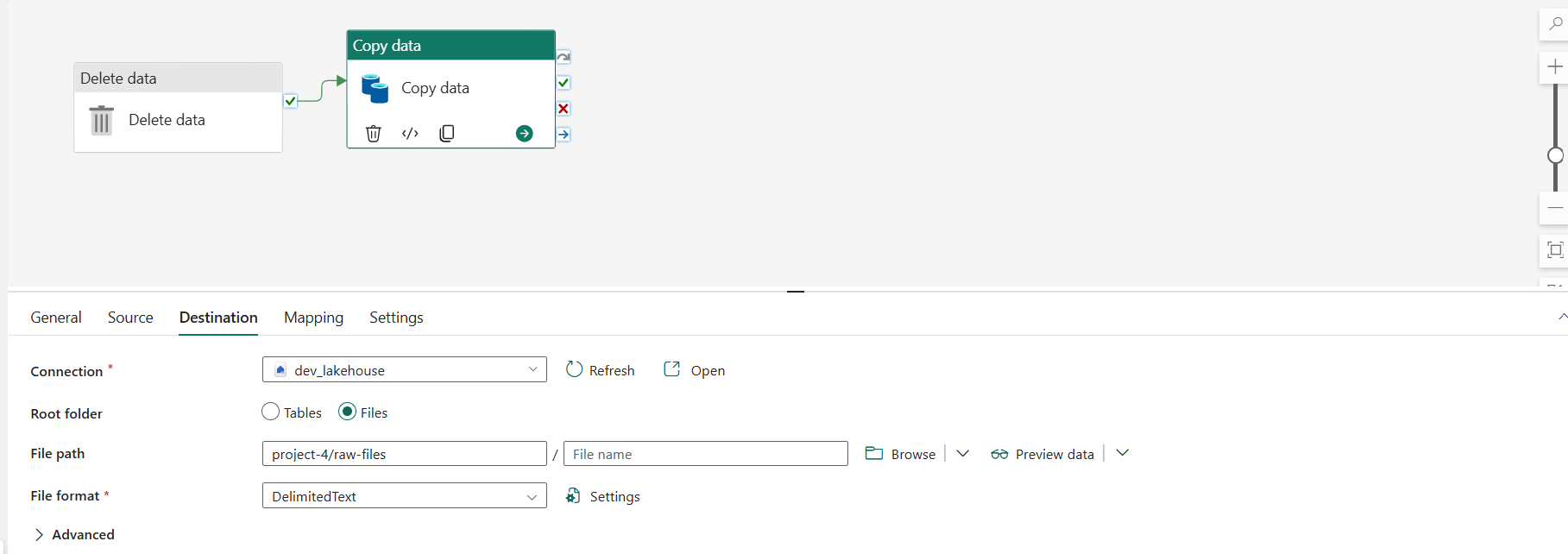
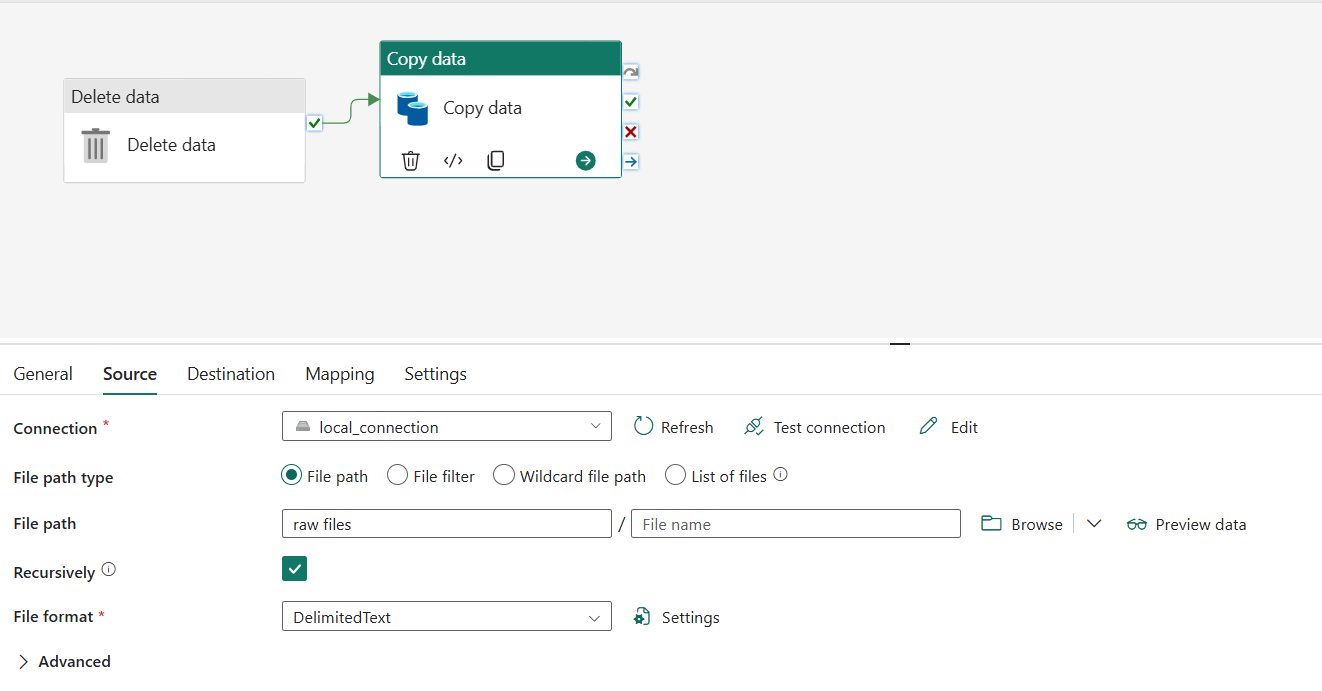
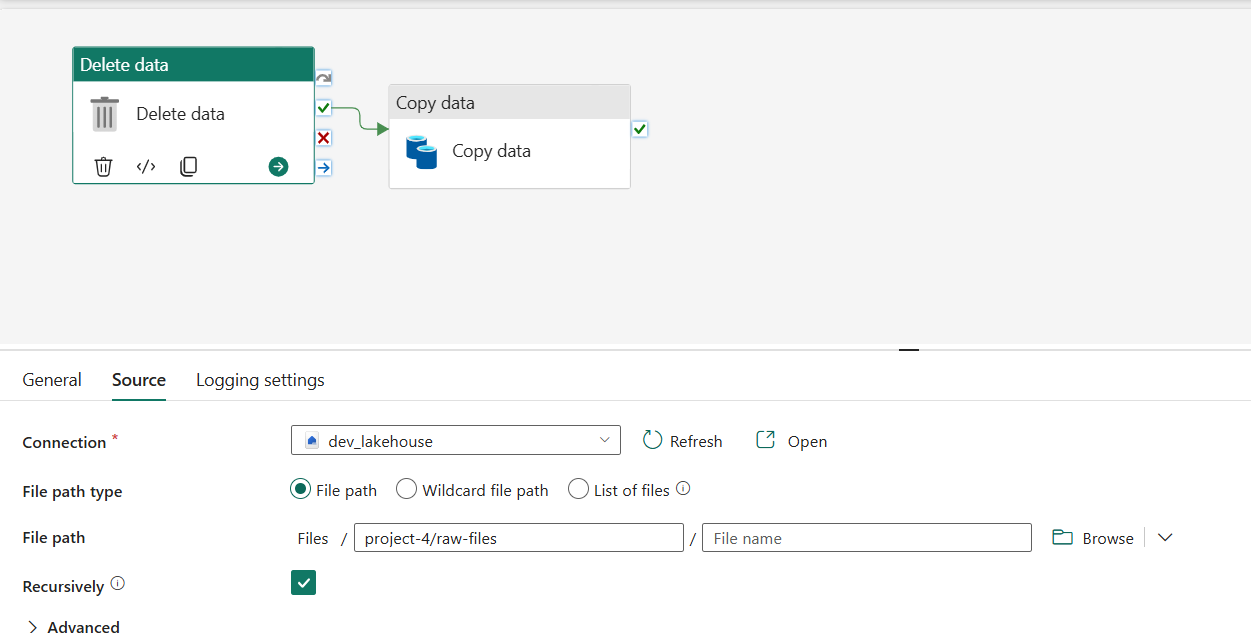
Now check it gateway status is online in the fabric.

Go to fabric>settings>manage connections and gateways>> gateways and check if it is online.

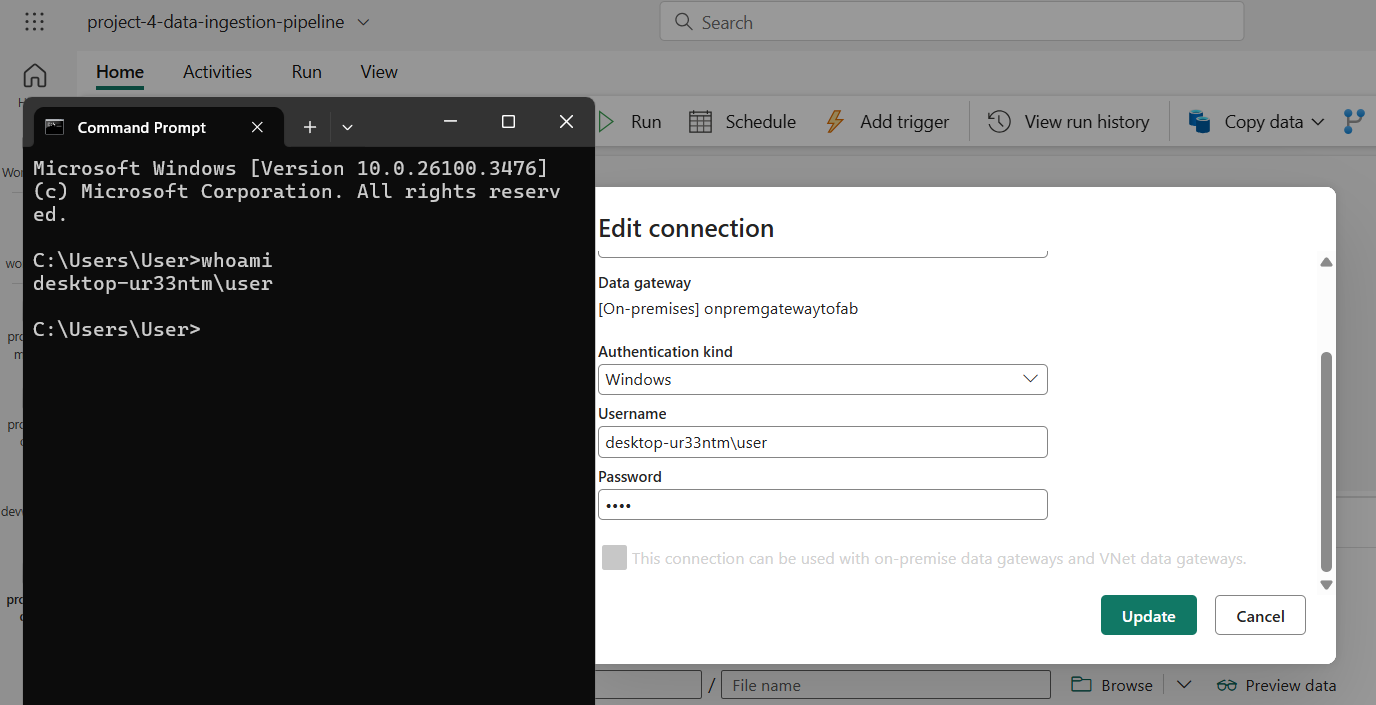


### Create new Data Pipeline for data ingestion

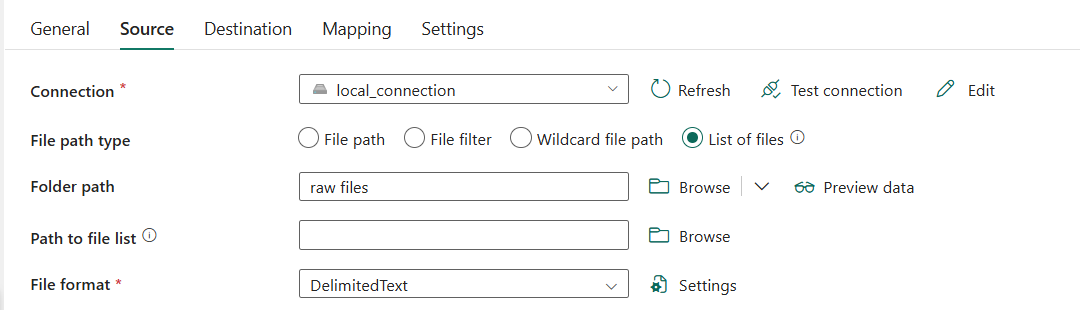
* Create delete activity and copy activity in the pipeline.
* Delete activity is to delete the previous csv files already present in lakehouse directory.
* In the copy activity, in source section, I have selected “folder” data source.
* In the destination section, I have used lakehouse as sink.



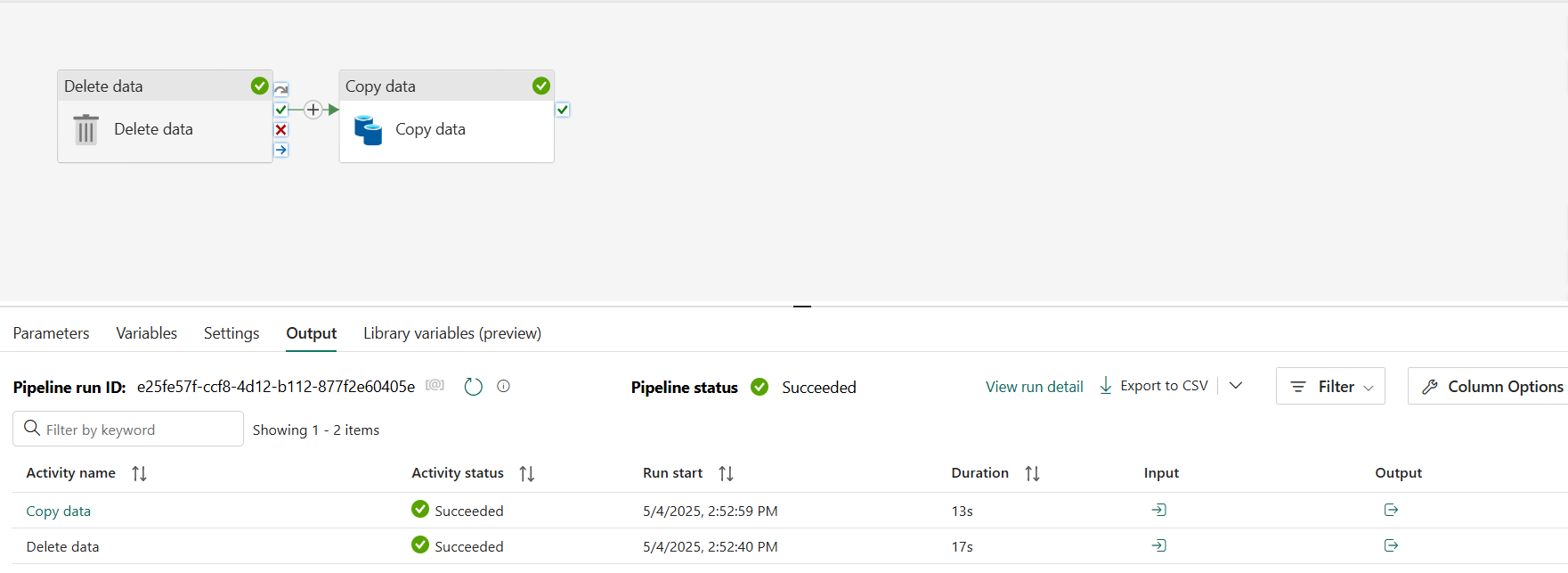
* In source section, I’ve used same gateway to connect with local PC.

Over here, we have to use windows credentials and Administrator’s account name.

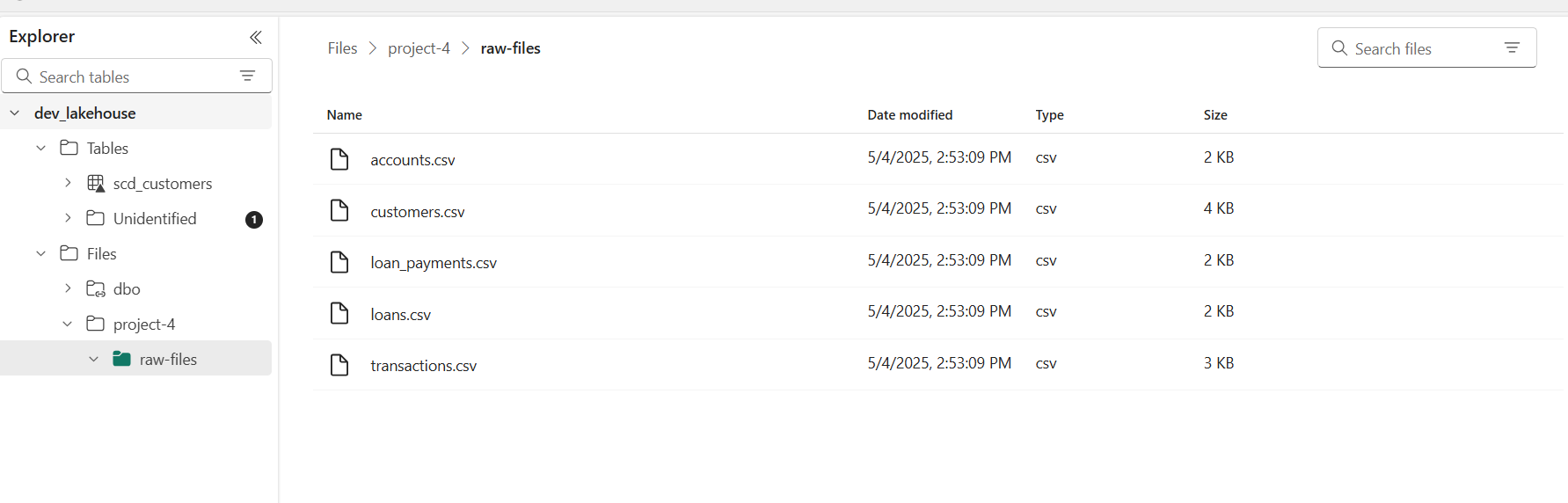
* Select the list of files options to copy all csv files from a dir



Here is the completion status of data ingestion pipeline:

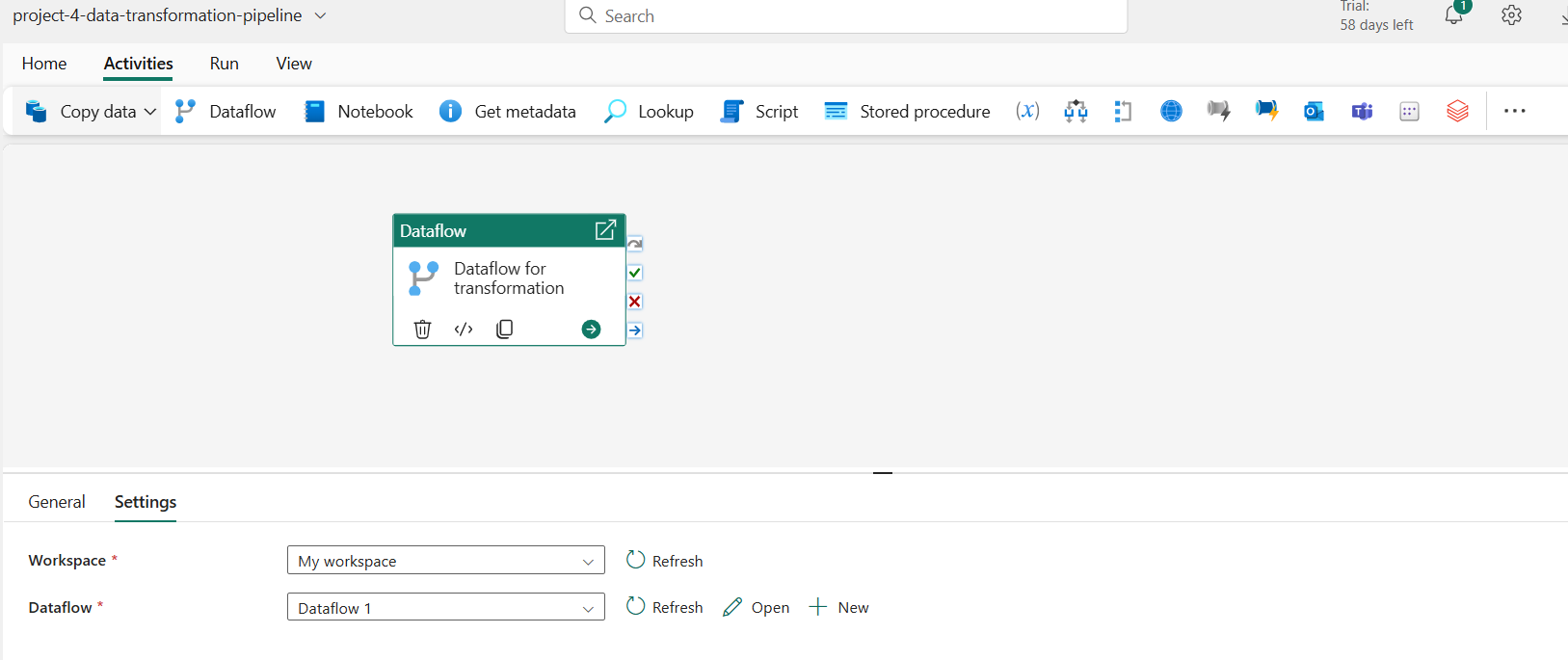


The data get copied from LOCAL Machine to Fabric Lakehouse:



## DATA CLEANING AND DATA TRANSFORMATION USING DATAFLOW GEN1 AND STORING IN FABRIC DATA WAREHOUSE

* Now, data got ingested into fabric lakehouse in raw files ( csv files).
* It’s time to clean the data and transforming it according to requirements.
* DATAFLOW gen1 will be used to clean and transform the data and all csv files data will be saved into Tables in fabric- data warehouse.



Here the data pipleine is created where dataflow is invoked, which helps in data cleaning and data transformation.

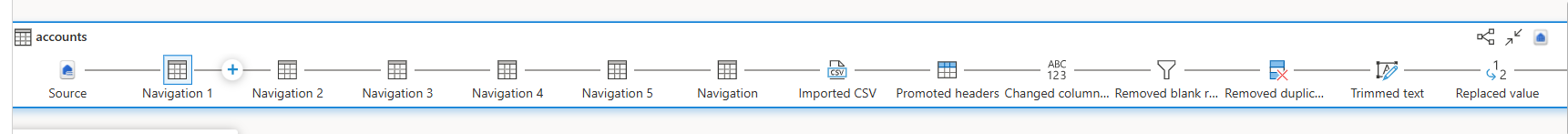
In the dataflow, I’m doing following tasks:

* Handle Missing Values
* Removing duplicate records
* Filtering the records
* Transforms date data type
* Trimming string column values

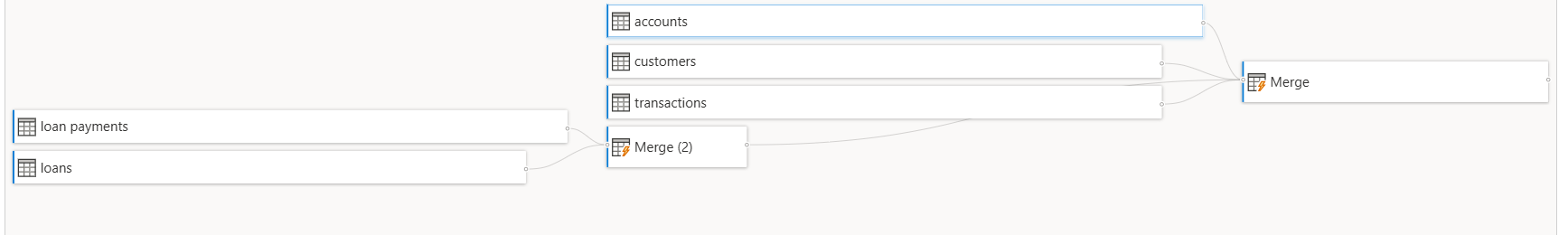
For each table. So there will 5 data sources and 5 sinks.

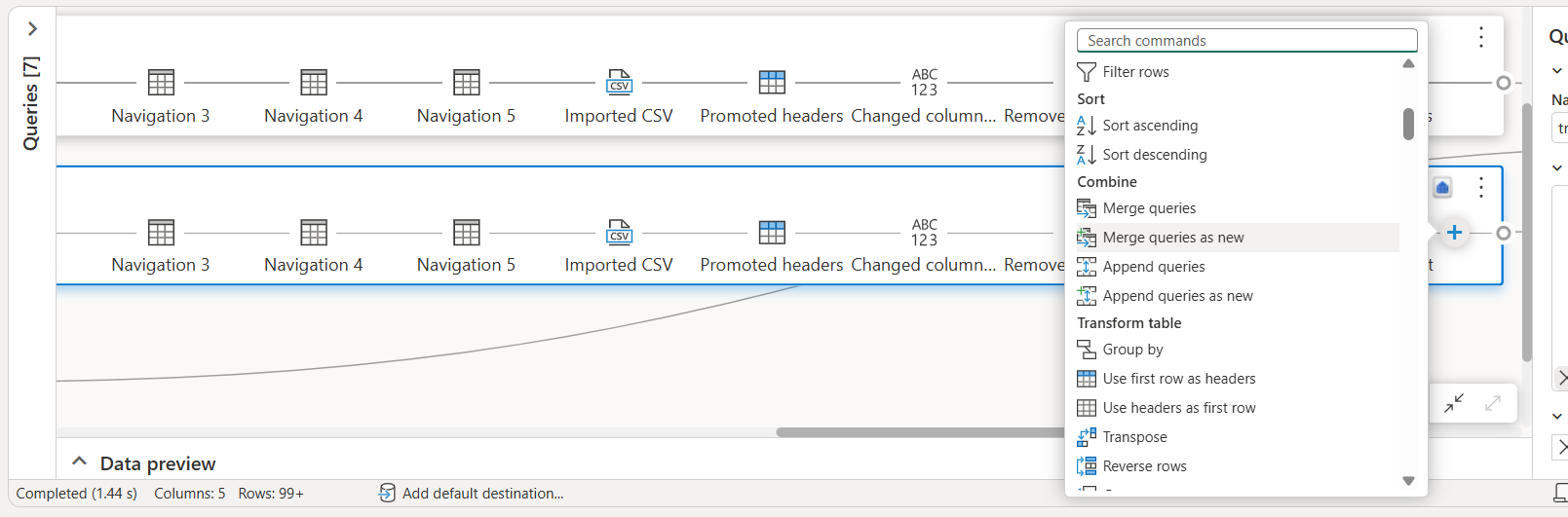
In addition to that, I’m creating new table, in which data from 5 tables get combined into single table named : “cleaned\_common\_table”

Here is the query view of 1 csv file :



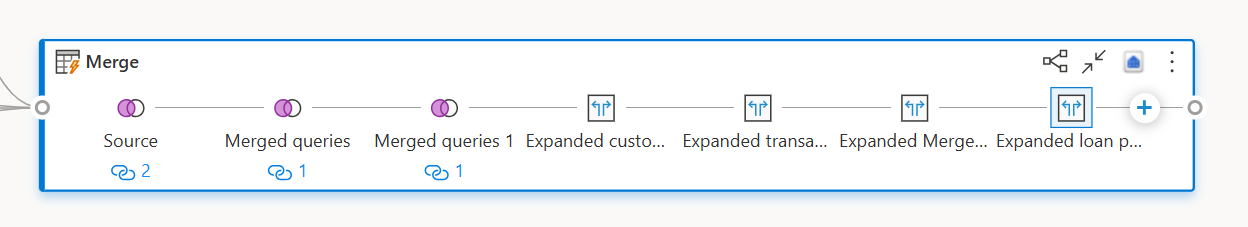
Here is the diagram view of those dataflow 1 :





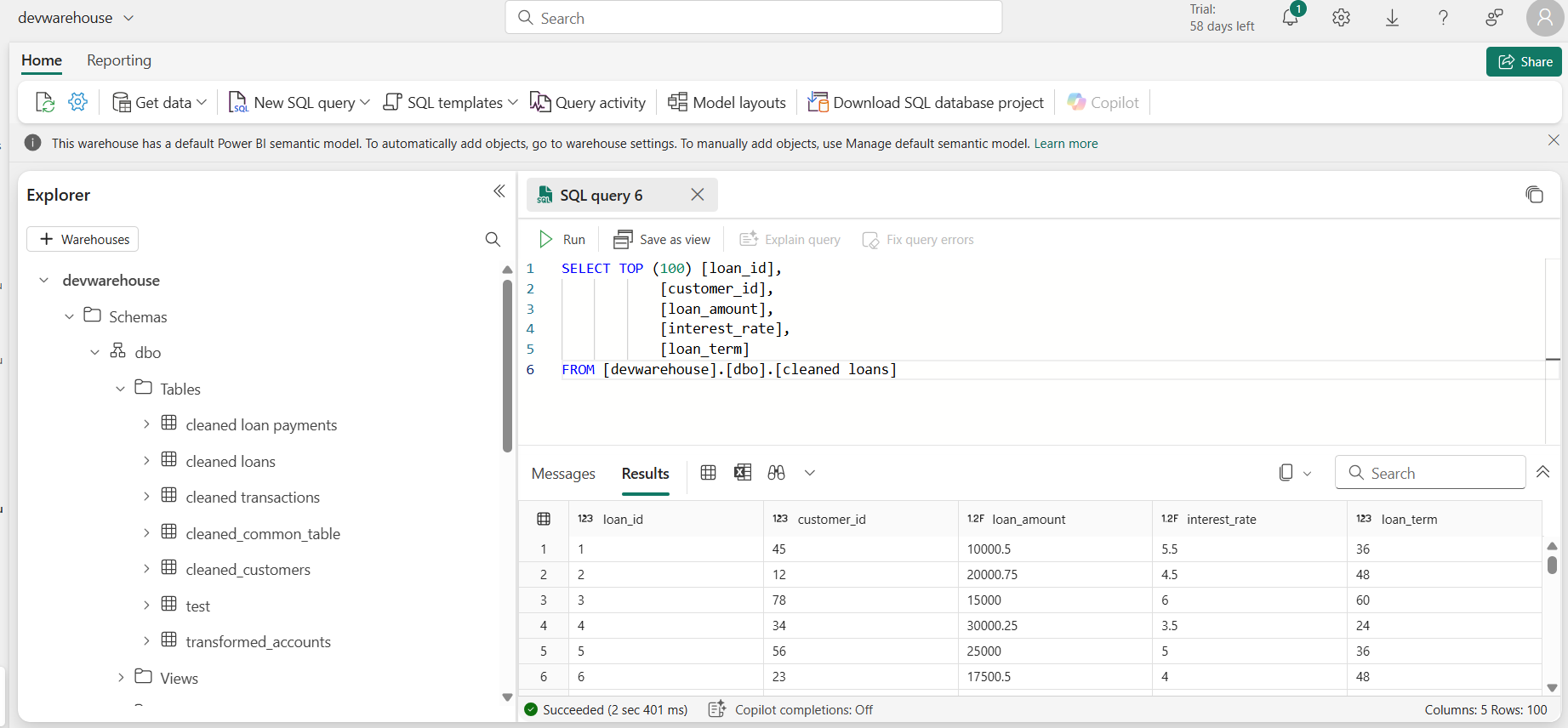
I’m using merge queries as new and merge queries activity to join two data stream into new stream.

In last, I've created destination table for that common stream too.

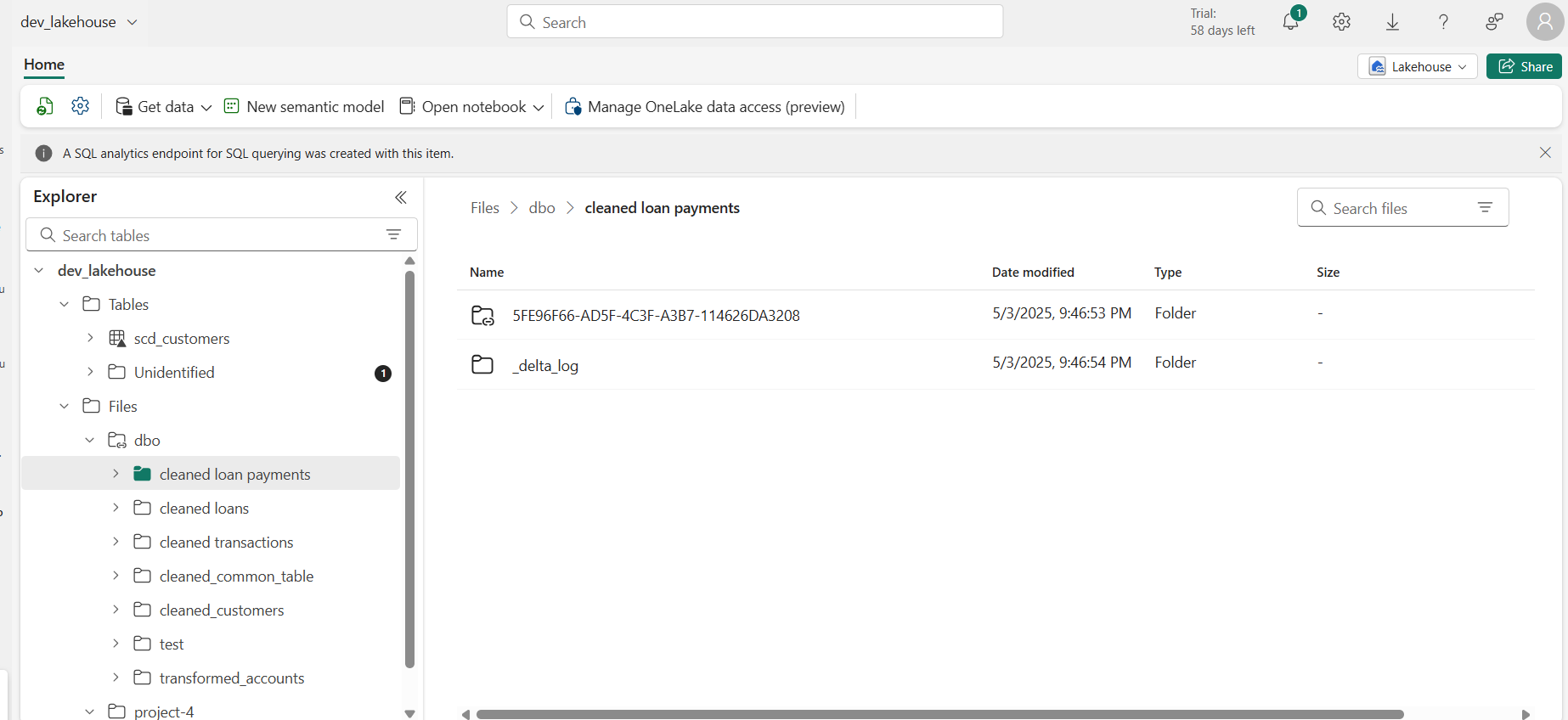


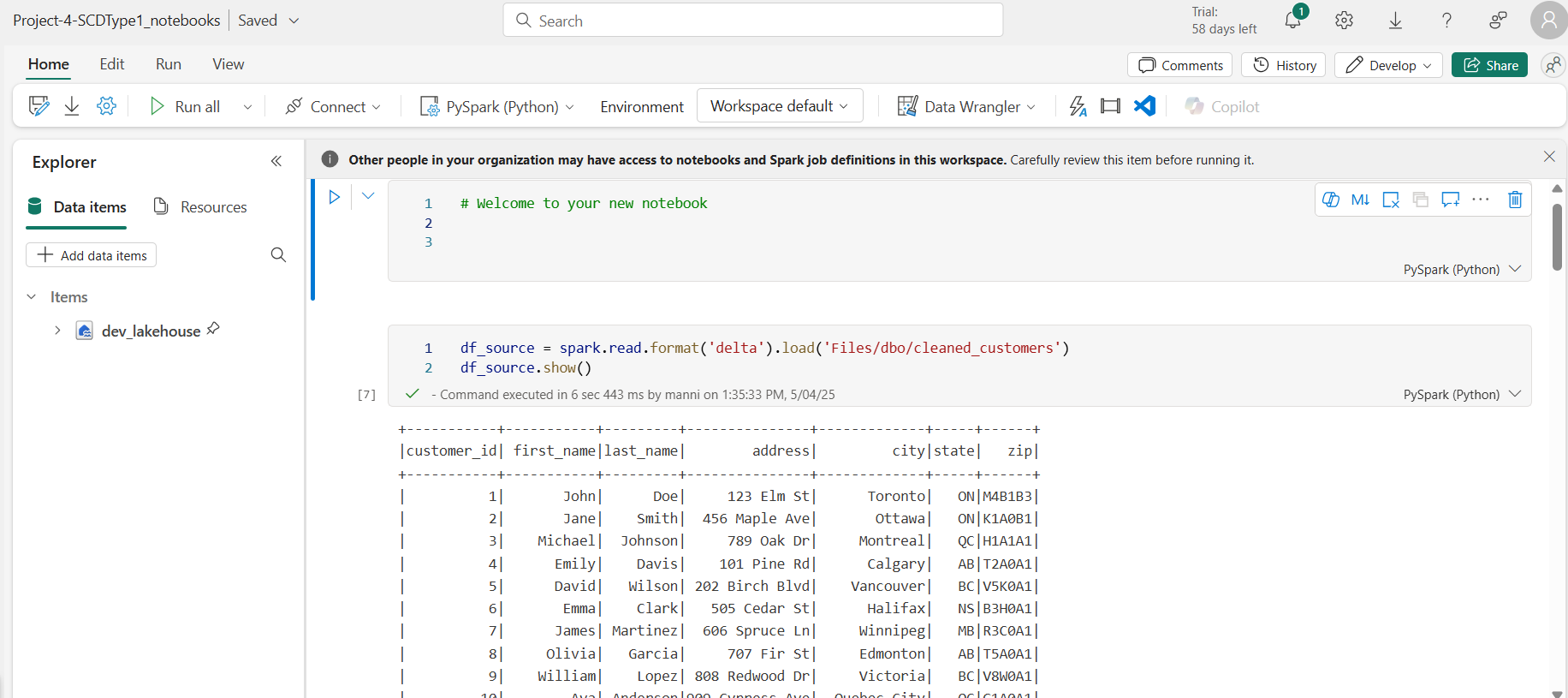
I’m using choose column activity and table expansion activity to explode the data into common stream.

Here the resulted data from dataflow 1 get stored as tables in devwarehouse:



## TRANSFORMING THE DATA INTO SCD TYPE 1 TABLES:

* For this, we are using jupyter notebooks in fabric.
* To access the tables from fabric warehouse, I’m using shortcut of lakehouse, which I’ve created in the lakehouse.
* 
* This is the notebook, where I was accessing warehouse’s tables using shortcut.



Then, I’ve created SCD tables for each transformed table.



Following is the code to transform dataframe object into SCD Type 1 and storing into target table:

from delta.tables import DeltaTable

target\_path = "Tables/dbo/scd/scd\_customers"

delta\_target = DeltaTable.forPath(spark, target\_path)

from pyspark.sql.functions import \*

df\_src1= df\_source.withColumn("hash\_key",crc32(concat(*df\_source.columns))) display(df\_src1) df\_src1=df\_src1.alias("src").join(delta\_target.toDF().alias("tgt"),((col("src.customer\_id")==col ("tgt.customer\_id"))&(col("src.hash\_key")==col("tgt.hash\_key"))),"anti").select(col("src.*")) df\_src1.show() from pyspark.sql.functions import col delta\_target.alias("tgt").merge(df\_src1.alias("src"),"tgt.customer\_id = src.customer\_id").whenMatchedUpdate(set={"tgt.customer\_id":"src.customer\_id","tgt.first\_name":"src.first\_name","tgt.last\_name":"src.last\_name","tgt.address":"src.address","tgt.city":"src.city","tgt.state":"src.state","tgt.zip":"src.zip","tgt.hash\_key":"src.hash\_key","tgt.updated\_date":current\_timestamp(),"tgt.updatedby":lit("databricks\_Updated") }).whenNotMatchedInsert(values={"tgt.customer\_id":"src.customer\_id","tgt.first\_name":"src.first\_name","tgt.last\_name":"src.last\_name","tgt.address":"src.address","tgt.city":"src.city","tgt.state":"src.state","tgt.zip":"src.zip","tgt.hash\_key":"src.hash\_key","tgt.created\_date":current\_timestamp(),"tgt.createdby":lit("databricks"),"tgt.updated\_date":current\_timestamp(),"tgt.updatedby":lit("databricks")}).execute() display(spark.read.format("delta").option("header","true").load(target\_path))

## SCHEDULING THE PROJECT BY MASTER PIPELINE AND ADDING ETL PROCESS COMPLETION NOTIFICATION:

* After completion of all phases, It’s time to schedule the whole process by invoking pipelines and notebook using master pipeline.
* We are also adding e-mail notification that the pipeline has run successfully.

