# **DATABRICKS x DBT End-To-End Data Engineering Project**

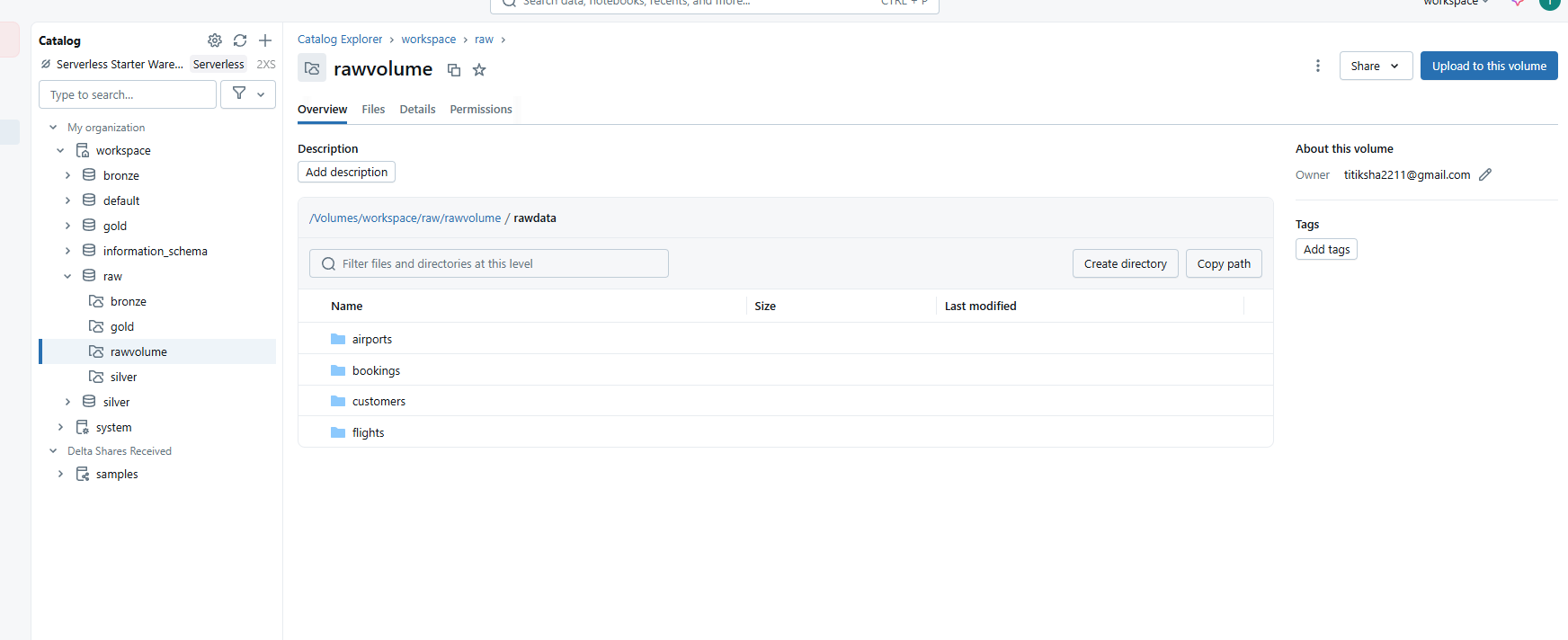
Titiksha Jain

## **Overview**

This project demonstrates an end-to-end data ingestion pipeline using Databricks, leveraging Unity Catalog for data governance, Auto Loader for incremental data ingestion, and Databricks Workflows for pipeline automation.

## **Prerequisites**

I created volumes in the Catalog: bronze, silver, gold, raw, and metastore. Using Databricks,I uploaded the following files into the databricks: dim\_flights,dim\_passengers,dim\_airports,fact\_bookings in the following rawdata schema



## **How I Created Volume & Schema in the Catalog**

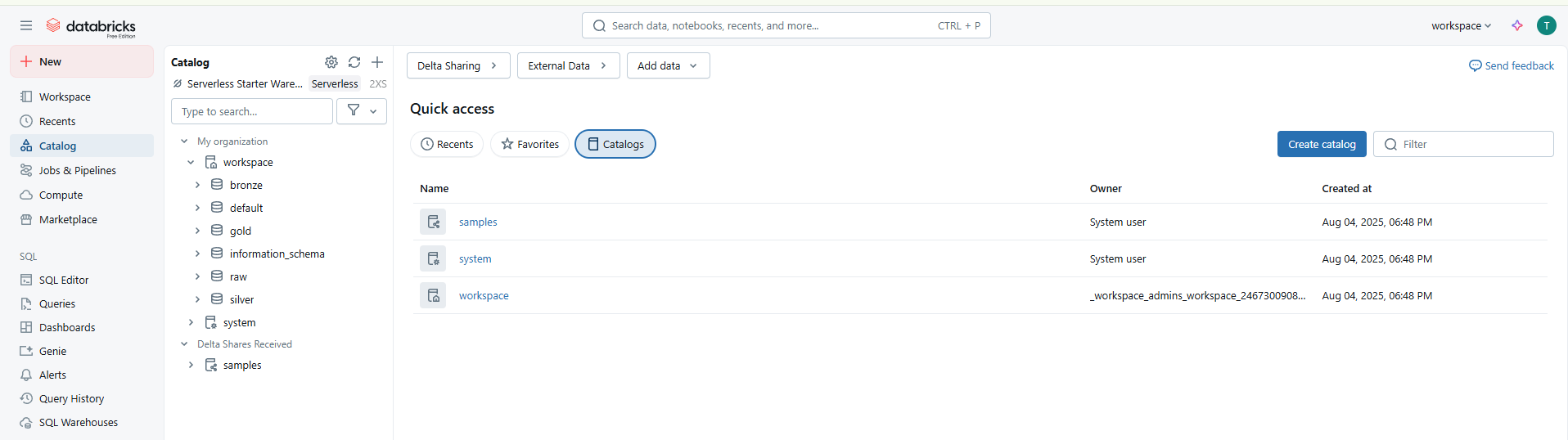
## **Create Databricks resource:**

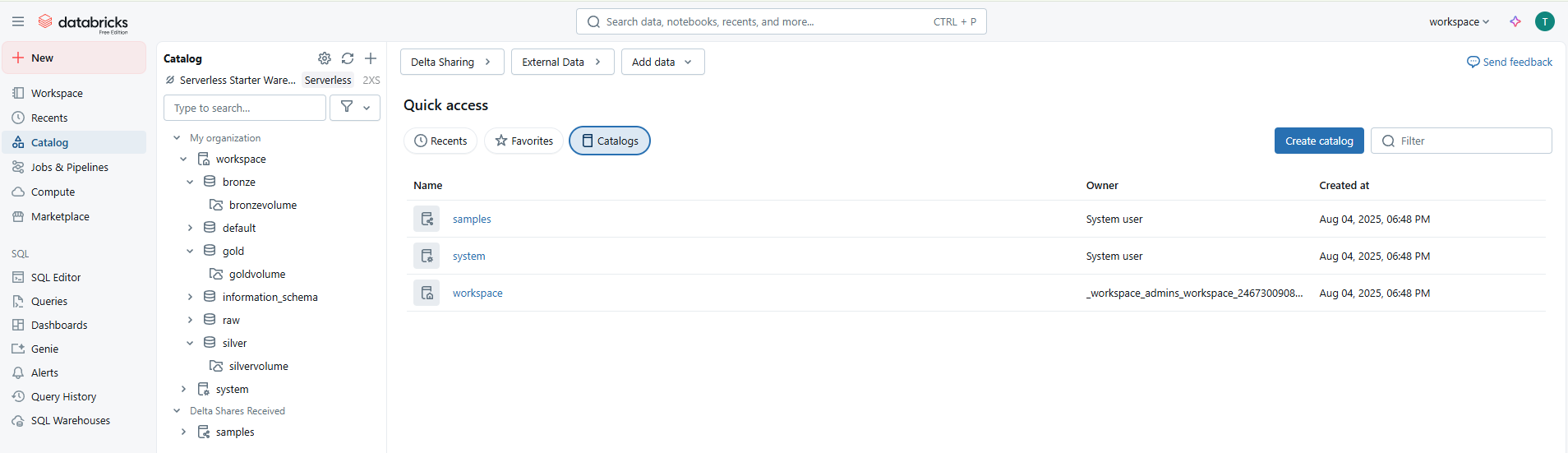
I set up a Databricks workspace notebook called “Setup”. Created Volume ‘rawvolume’. Then created directory ‘rawdata’ and created new folders ‘flights’ , ‘airports’, ‘bookings’ and ‘customers’ and uploaded dim\_flights,dim\_passengers,dim\_airports and fact\_bookings in the new folders created under ‘rawdata’ folder.



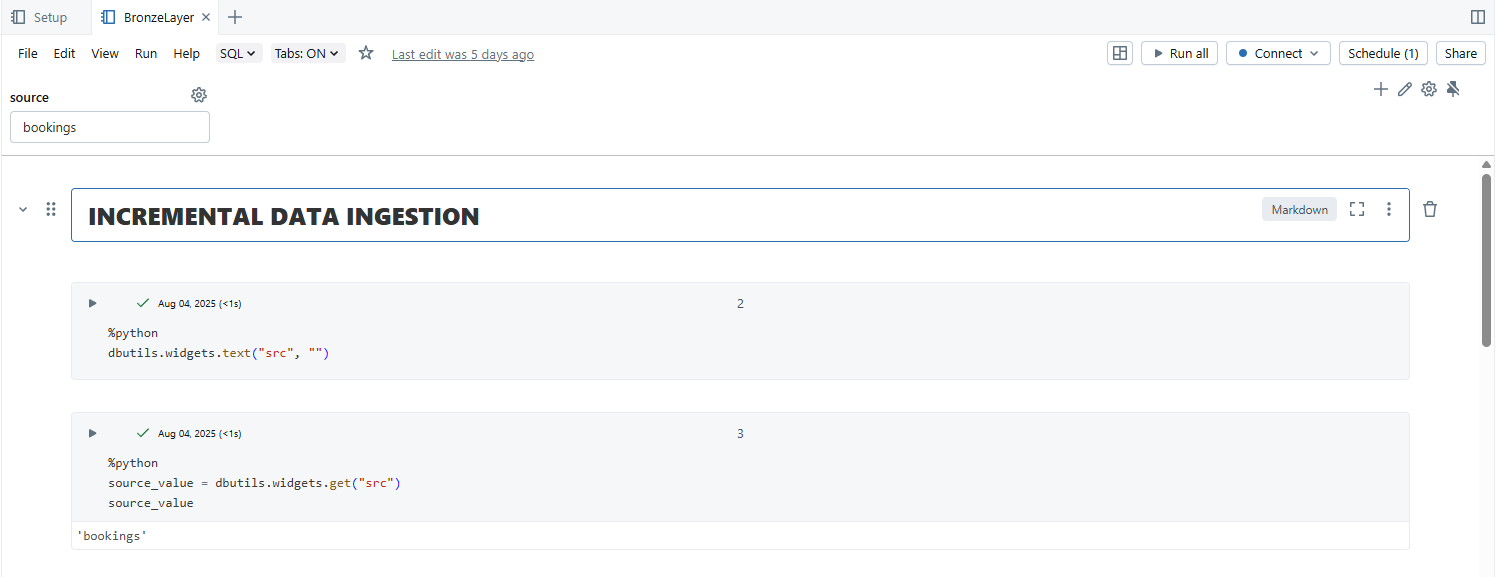
Create Schema for the Layers:

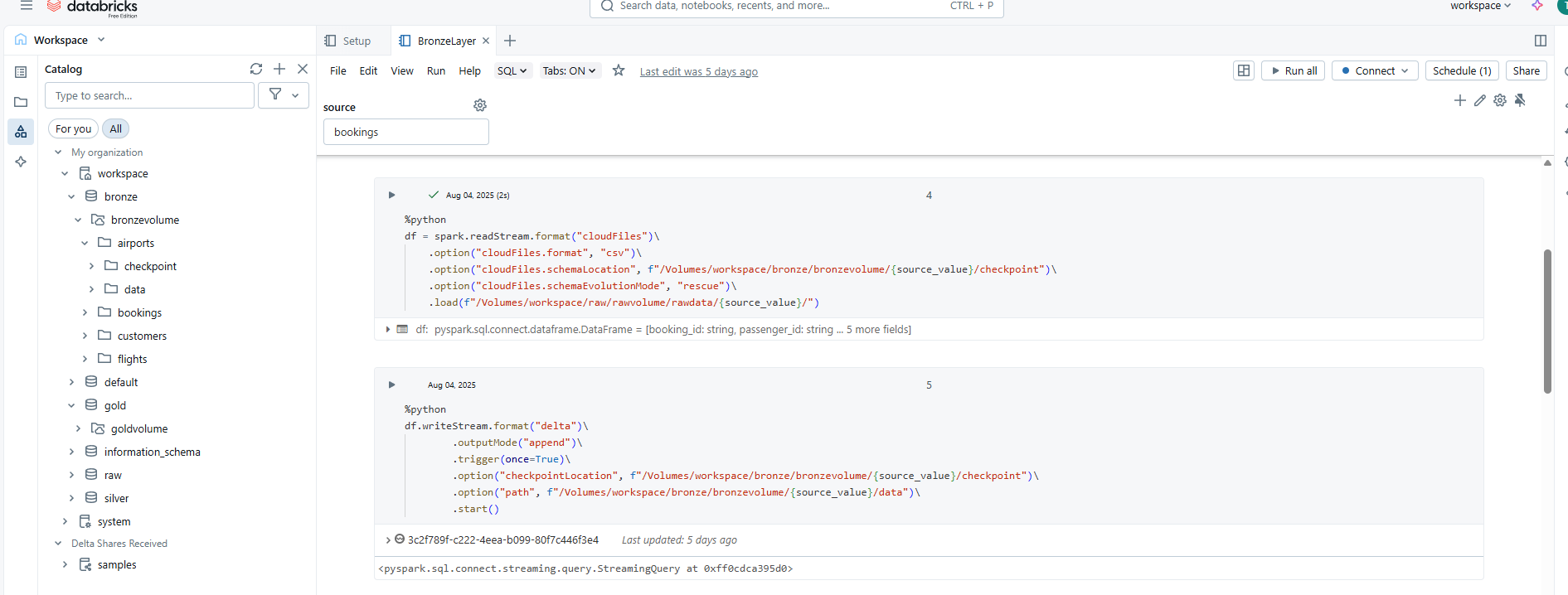
Created schema for bronze, silver and gold. Within these new schemas, created volumes ‘bronzevolume’,’silvervolume’ and ‘goldvolume’ respectively.





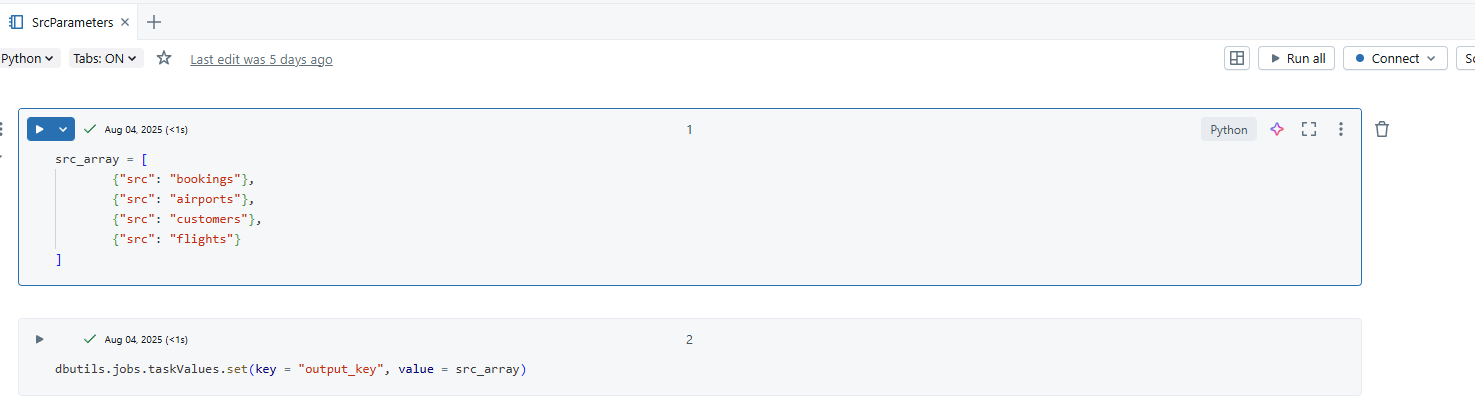
Created dynamic parameter to read the files

Using the parameter, we read the csv files in the bronze, silver and gold layers. Also added checkpoint location in the volume folder.



Created another workspace as ‘SrcParameters’

Created an array which will be used as an input in another notebook. Then created a dictionary that use that array

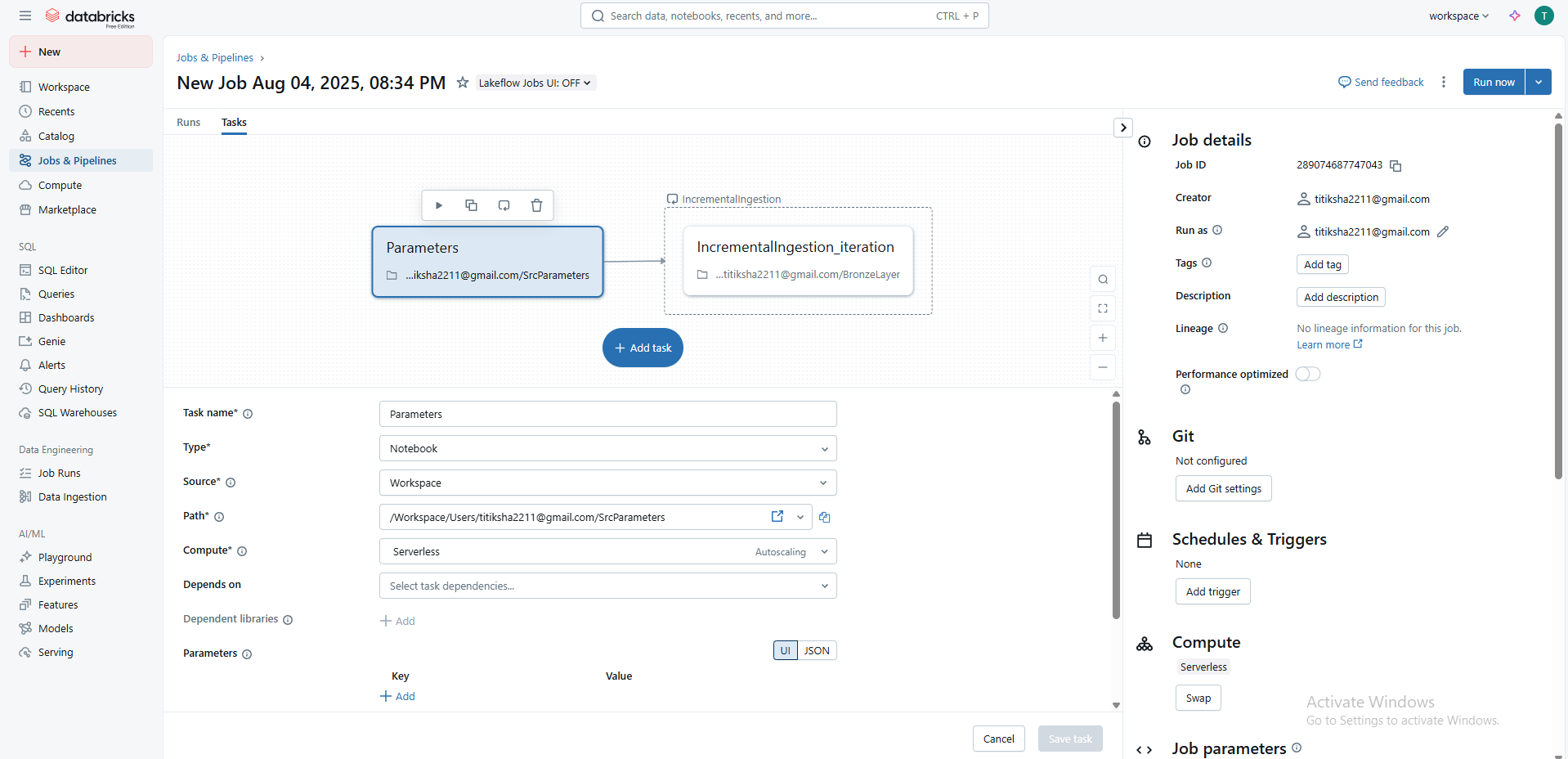


Created pipeline under workflows.

Put the Task Name: Parameter

Path: put the path of the srcparameter notebook

Compute: Severless



Then add task:

Task Name: IncrementalIngestion

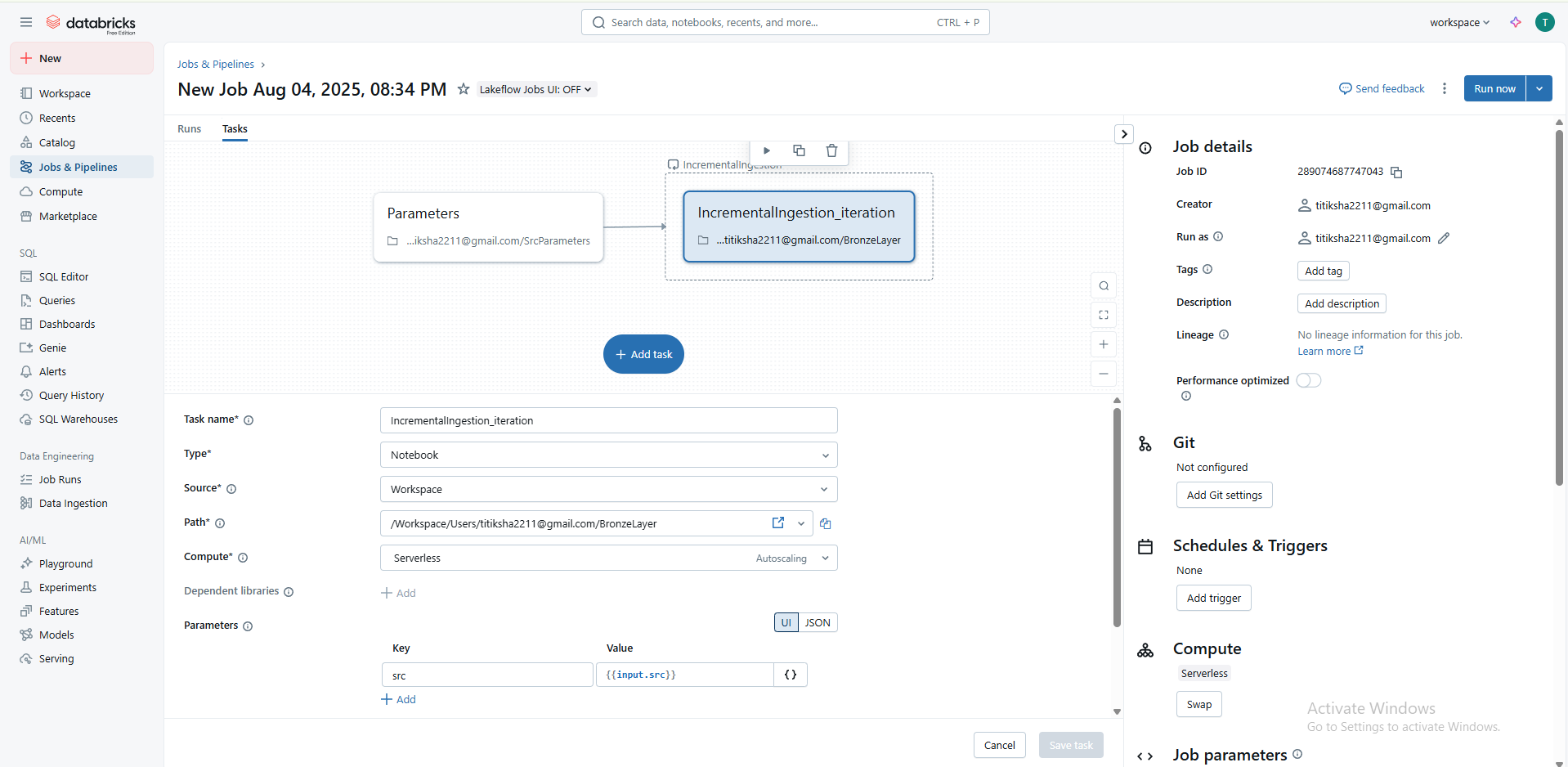
Path: put the path of BronzeLayer notebook

Depends On: Parameters

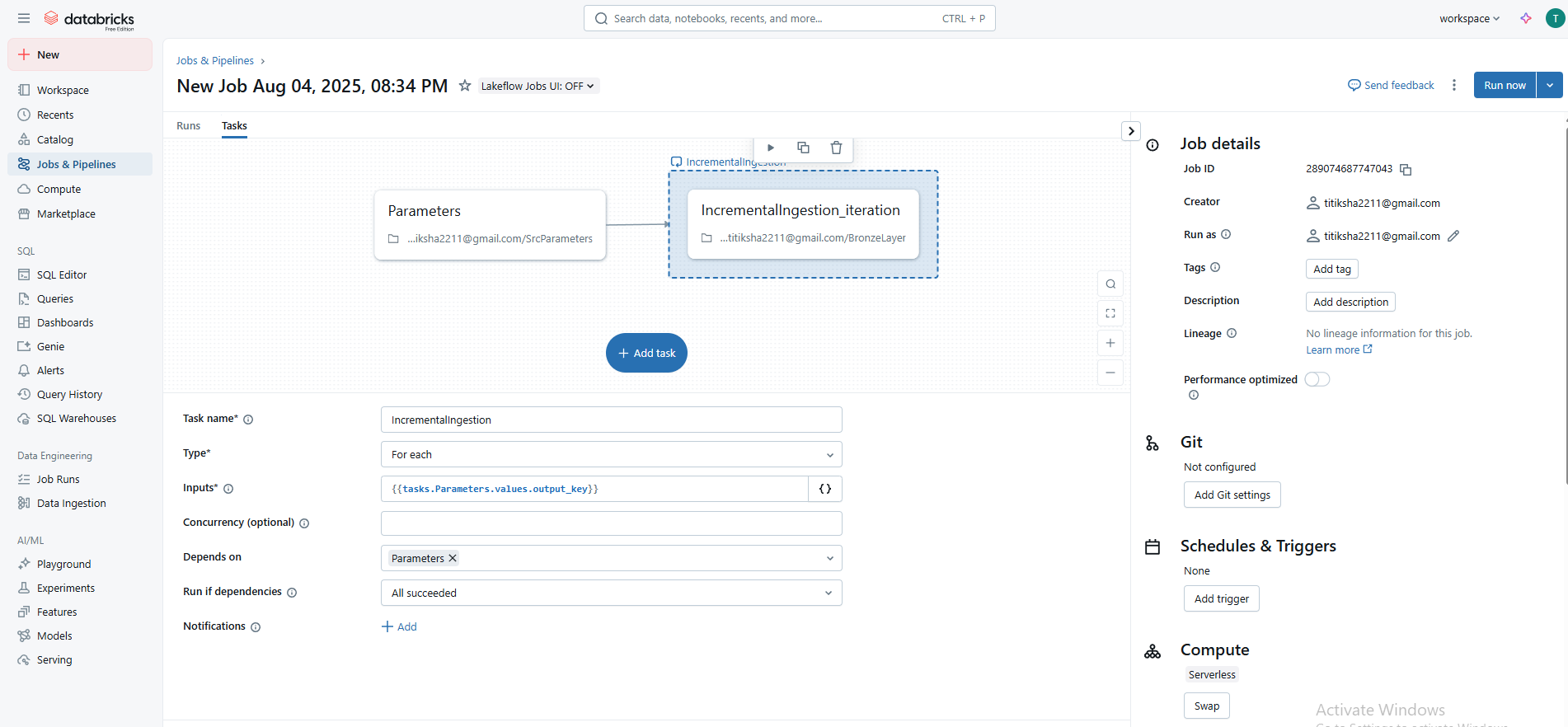
Add parameters:

Key = Src

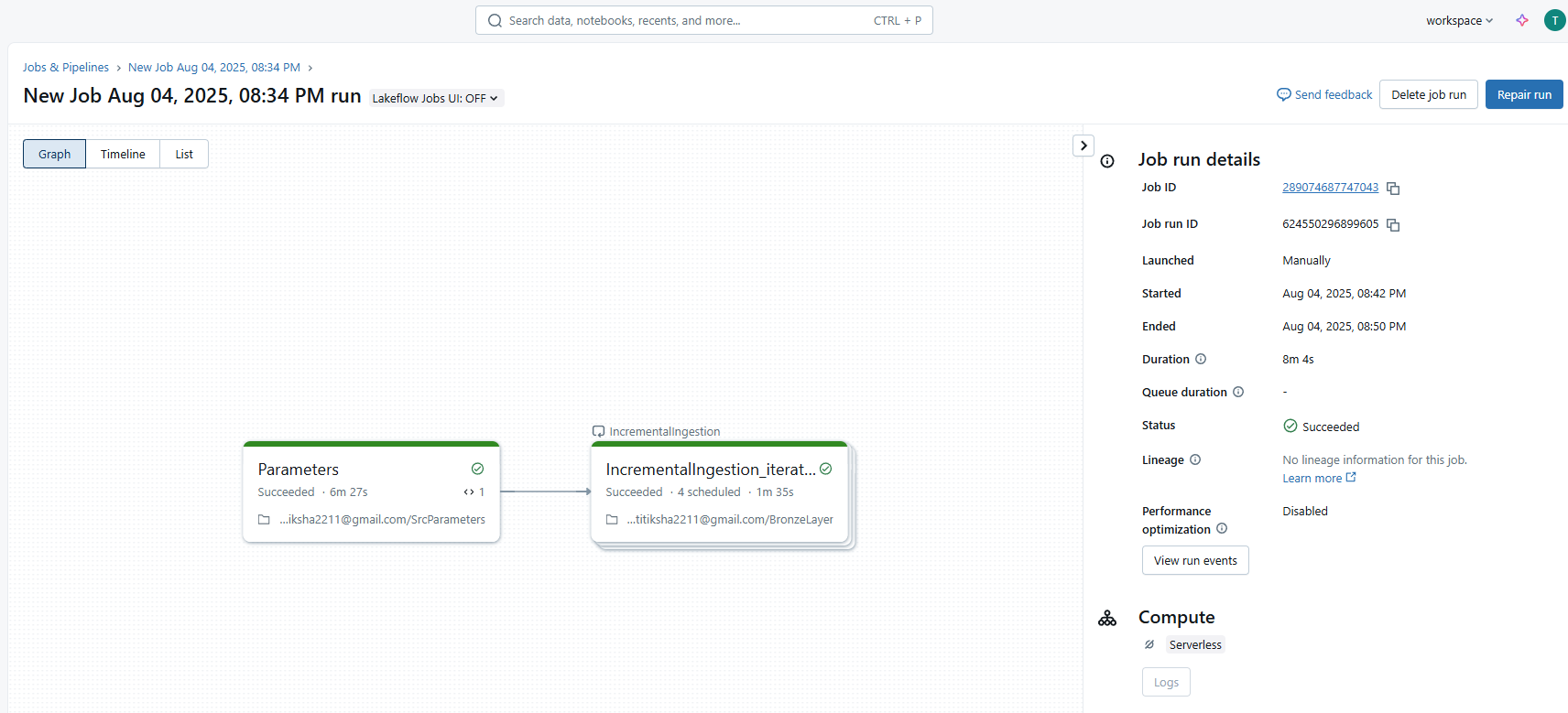
Value = {{input.src}} (added after the loop is done)



Make a loop for this task => Put Input: {{tasks.Parameters.values.output\_key}}



Then run the pipeline



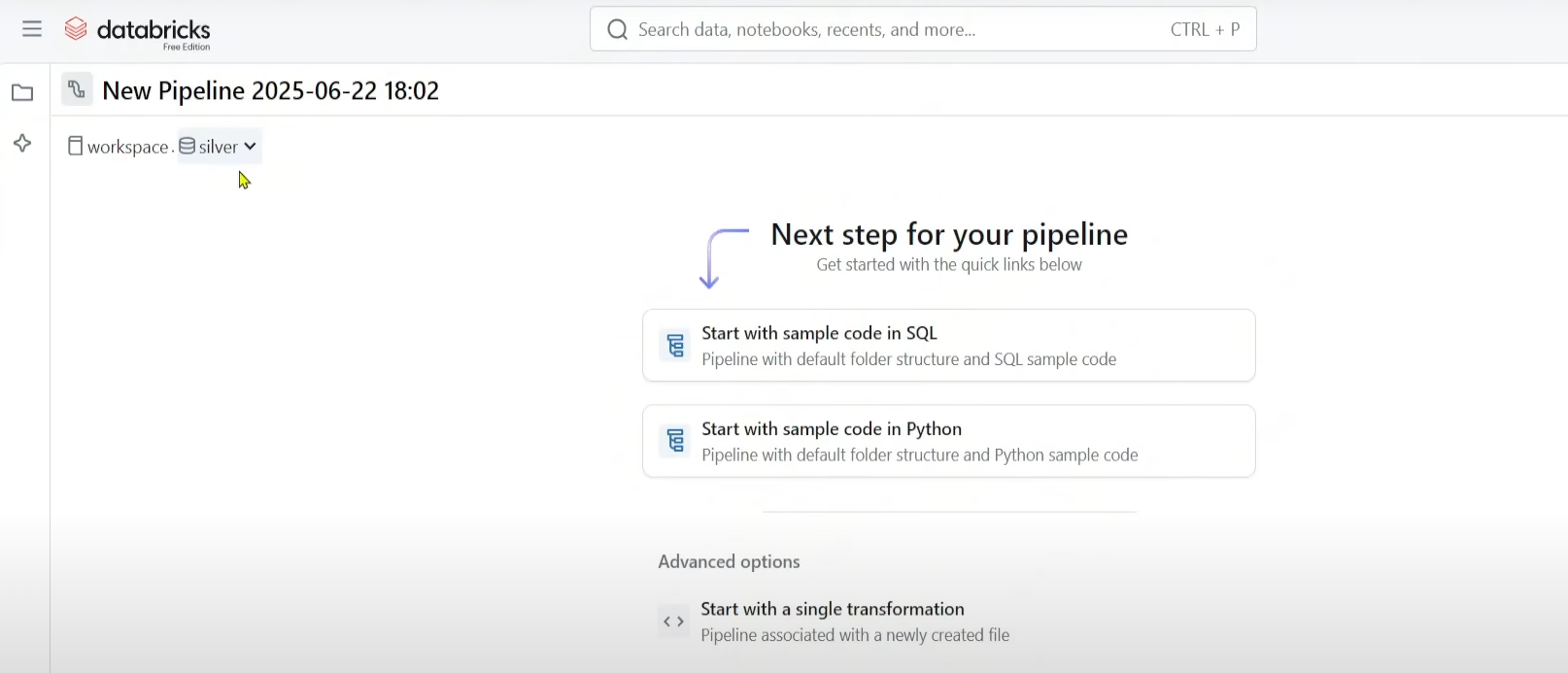
Create new notebook ‘SilverNotebook’

After running the select query to see the data in the bronzelayer.

We try to transformation in the silver layer using DLT

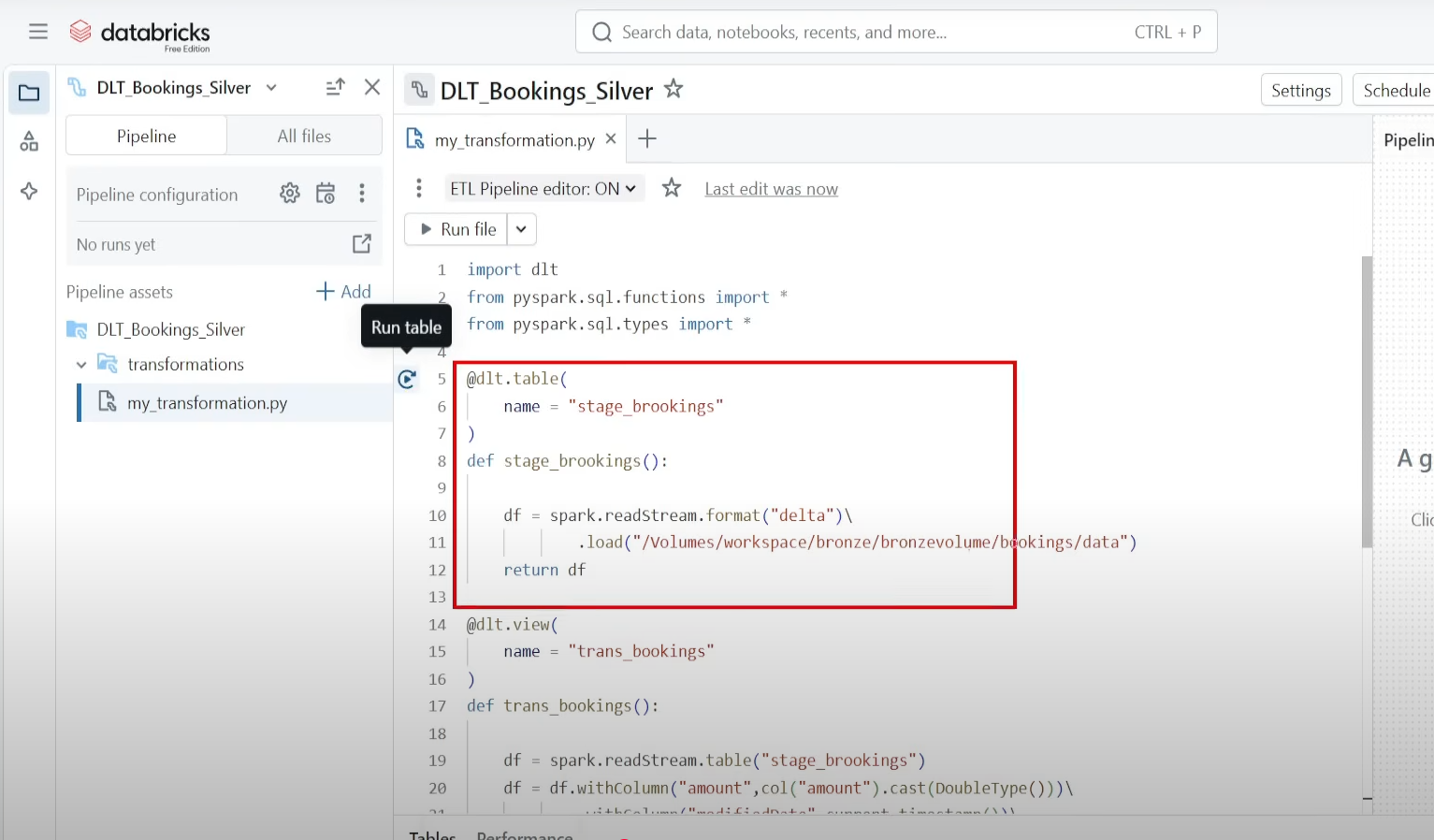
We will create rules using dictionary

Create a pipeline using a single transformation and in workspace option select silver schema

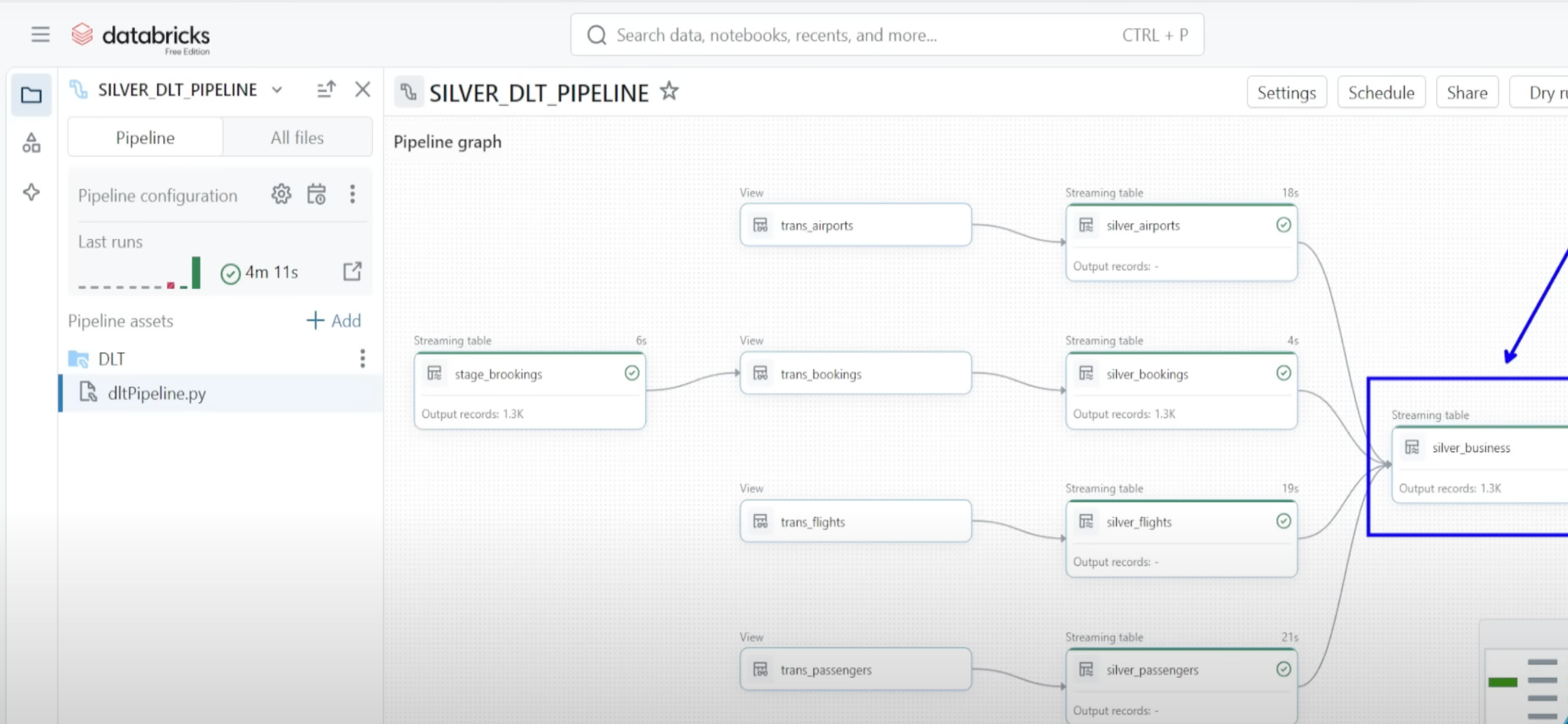


Name it as ‘DLT\_Bookings\_Silver’

Under the pipeline assets, a transformations folder will be created and an empty python file my\_transformation has been created. In the python file add the transformation logic and rules required



After the changes we get the end-to-end silver layer pipeline



Now will start preparing the Gold layer

