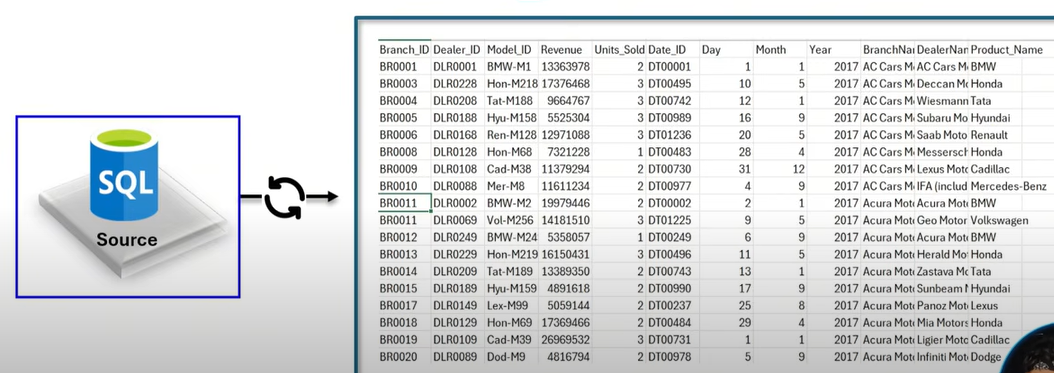
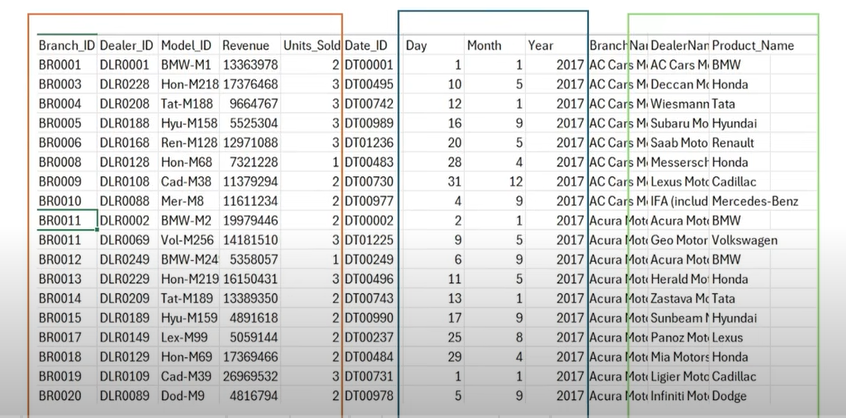
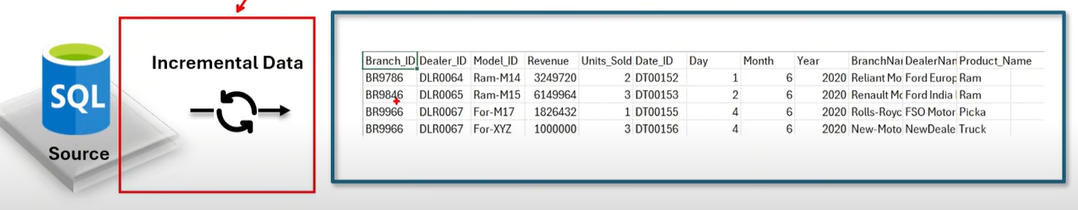


Below data is coming into bronze layer  


Below data is coming into silver layer  




Create resource group

Create ADLS (bronze, silver, gold)

Create SQL Server and create below table.

CREATE TABLE CAR\_SOURCE\_DATA(

Branch\_ID  VARCHAR(200),

Dealer\_ID VARCHAR(200),

Model\_ID VARCHAR(200),

Revenue BIGINT,

Units\_Sold BIGINT,

Date\_ID VARCHAR(200),

Day INT,

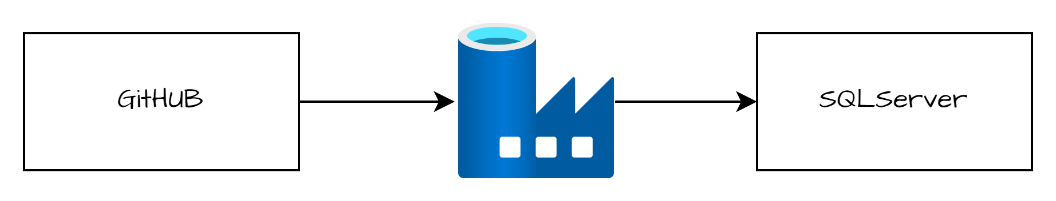
Month INT,

Year INT,

BranchName VARCHAR(200),

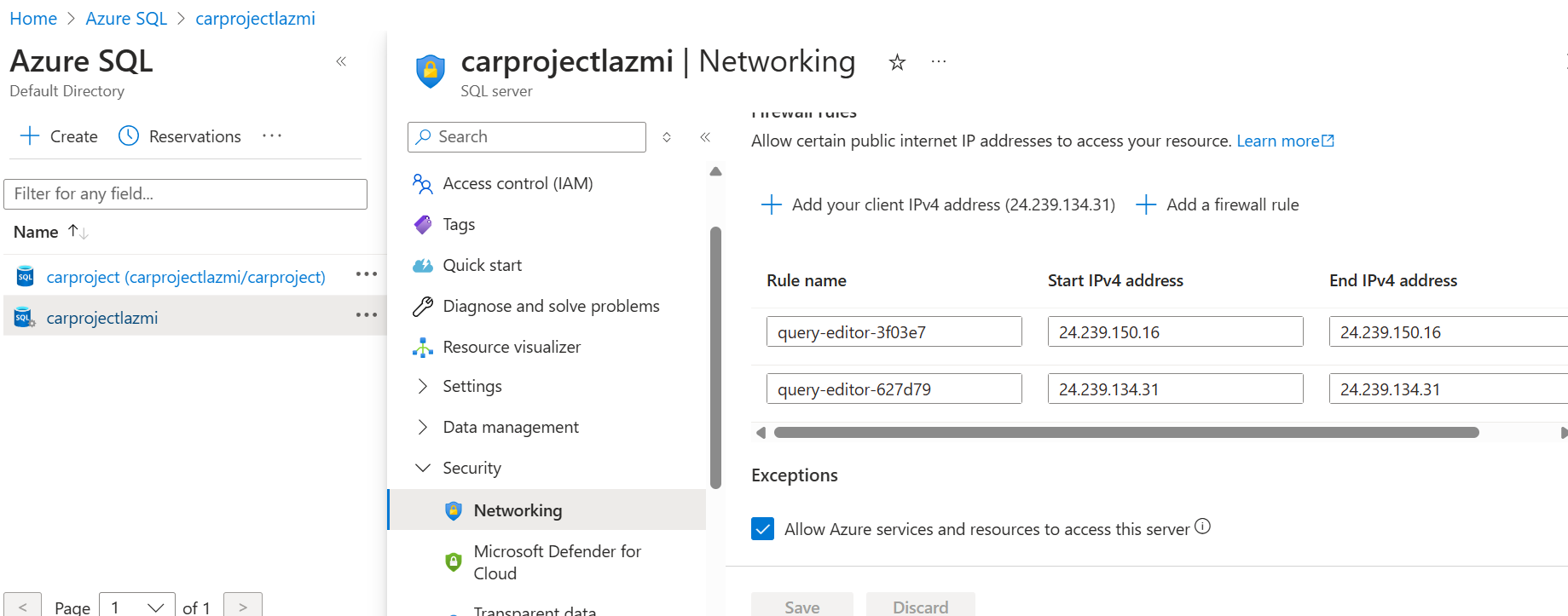
DealerName VARCHAR(200),

Product\_Name VARCHAR(200));



Create ADF, create linked service for github and for SQL server.

When we will create linked service for SQL server, we will be facing one error due to blocked access.  
Check below box for personal learning scenarios



Create source and sink dataset and debug the pipeline which will do initial load in sql server table.

Now, we have to write logic for incremental load.

Create watermark table with one column “last\_load” in sql server.  
  
create table water\_table(

last\_load  varchar(200));

select min(Date\_id) from [dbo].[CAR\_SOURCE\_DATA]

CREATE PROCEDURE UpdateWaterMarkTable

@lastload VARCHAR(200)

AS

BEGIN

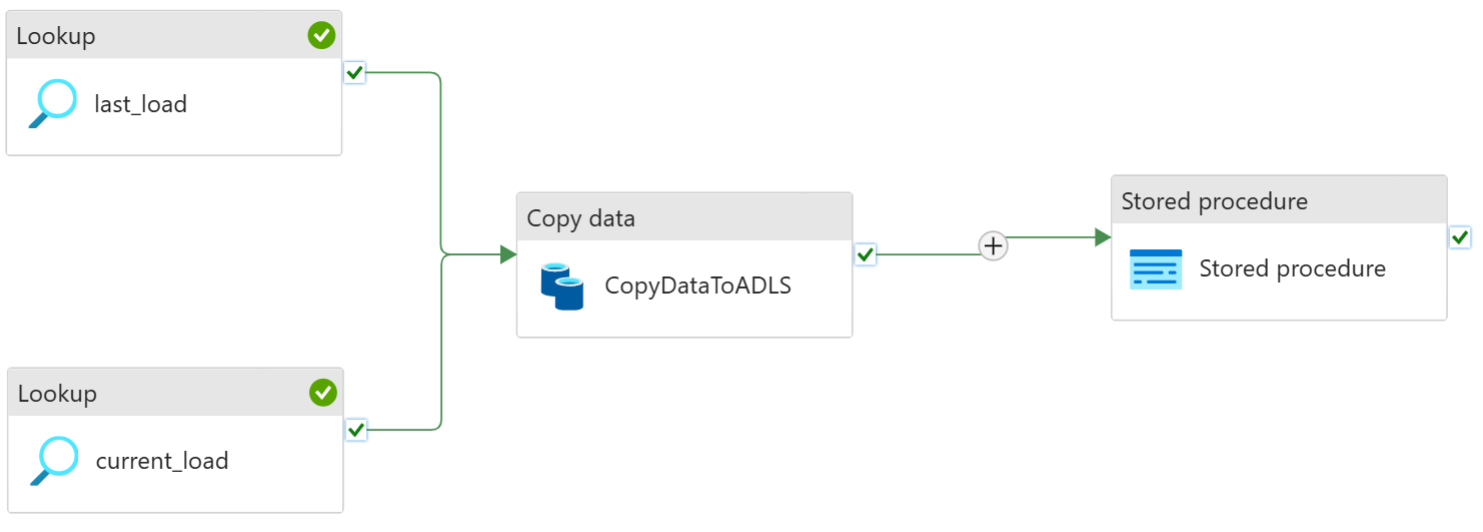
   BEGIN TRANSACTION;

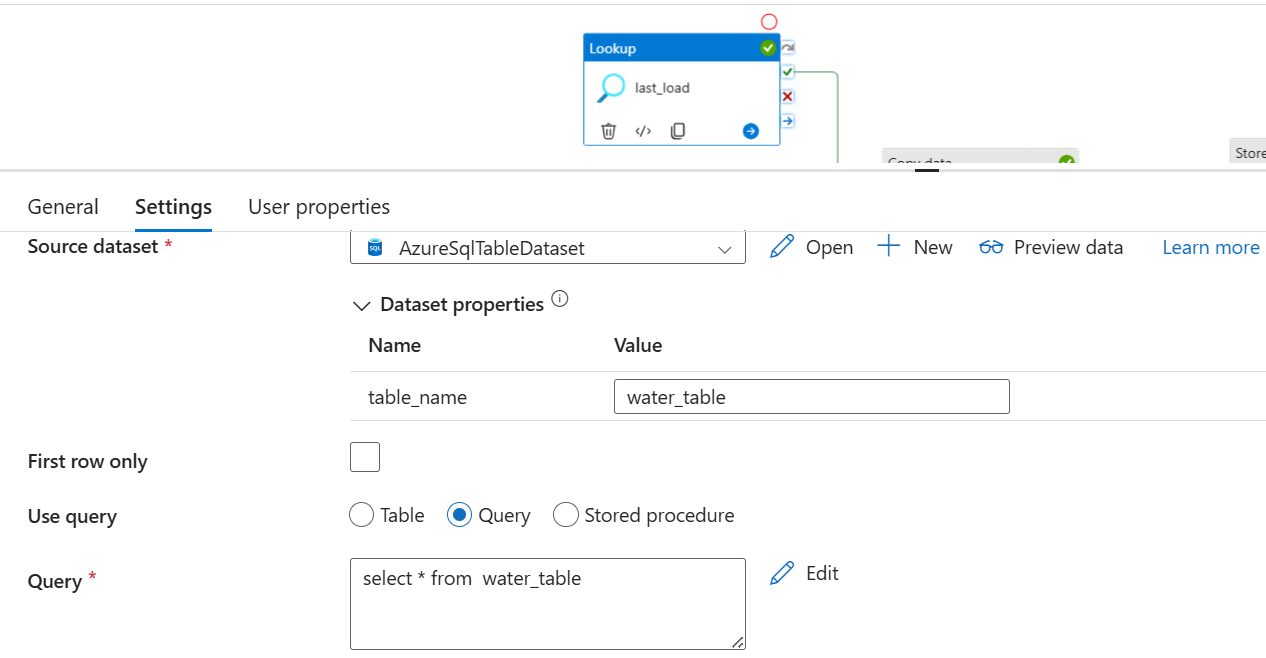
   UPDATE water\_table

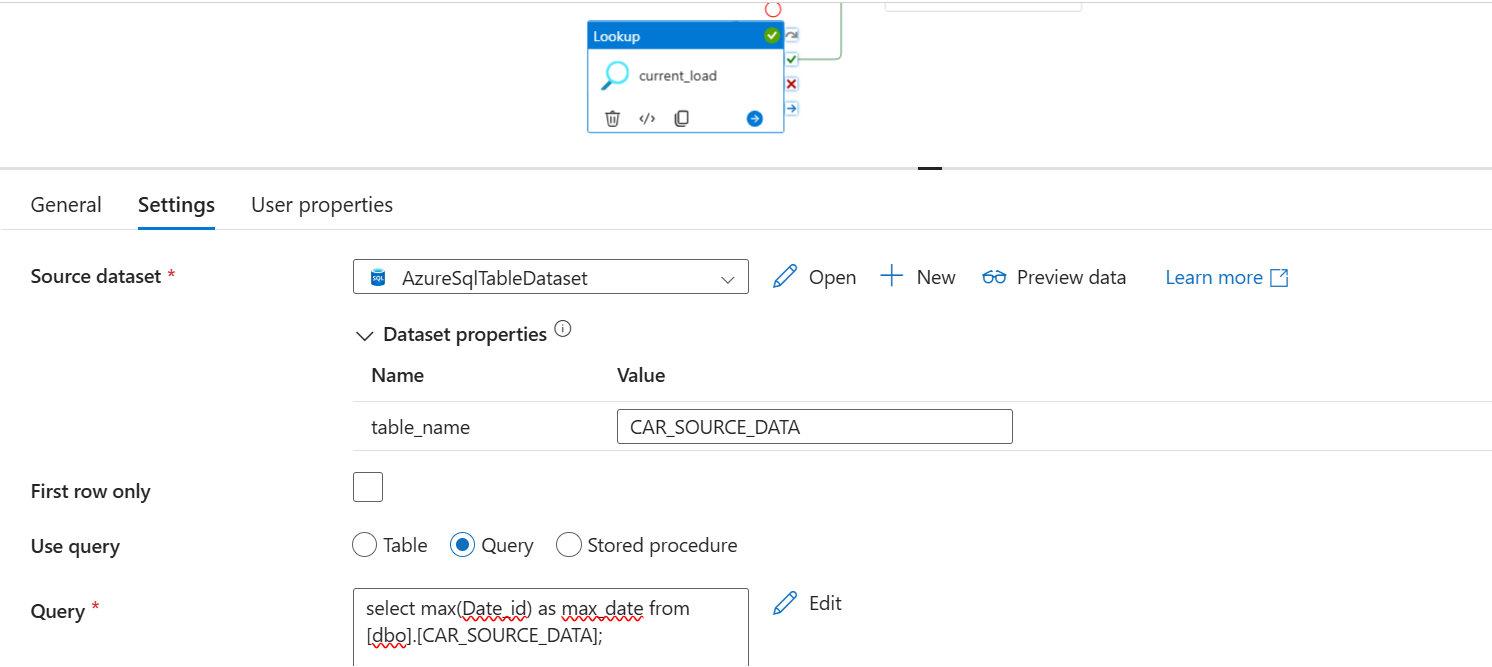
   SET last\_load = @lastload

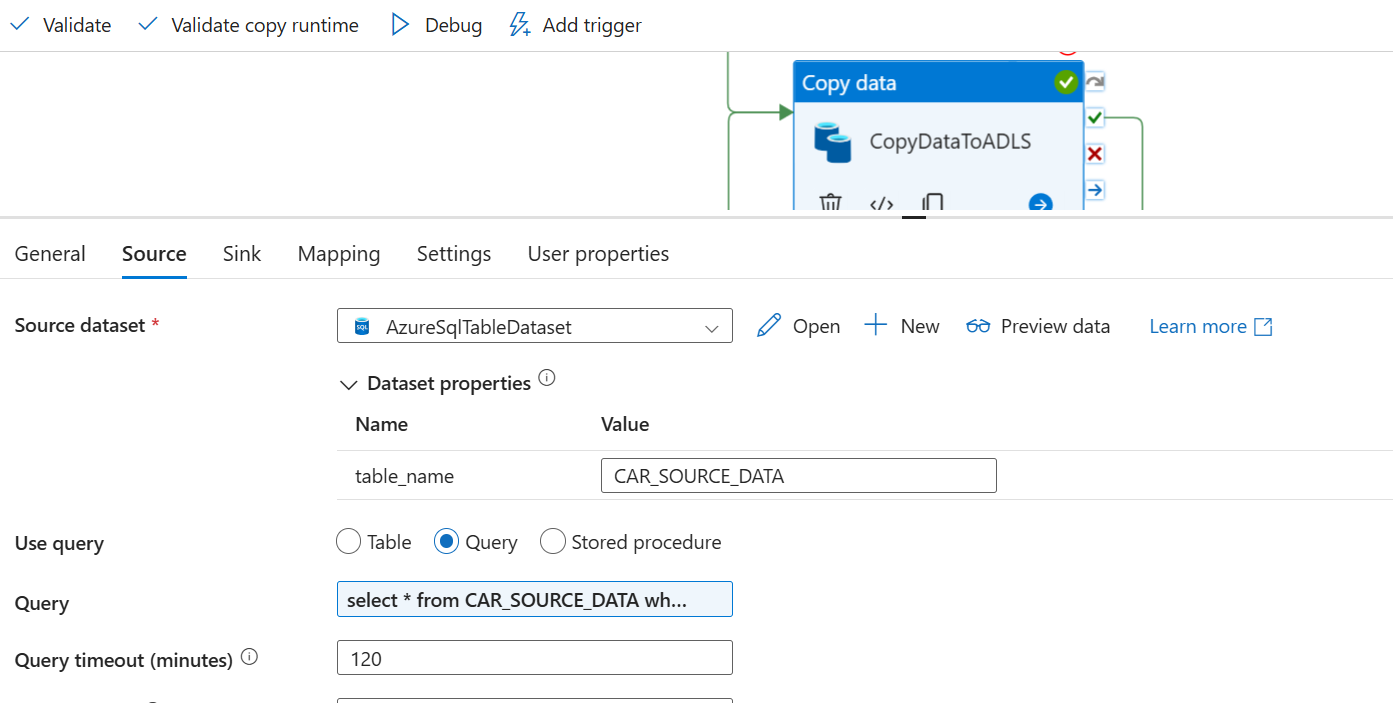
COMMIT TRANSACTION;

   END;

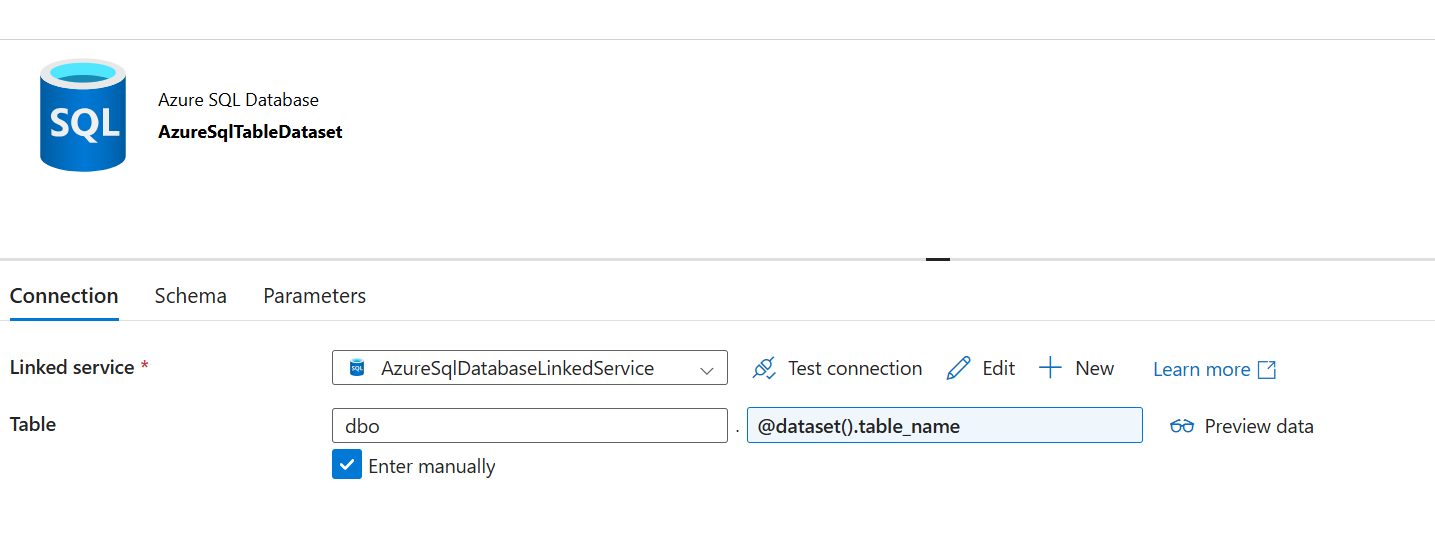


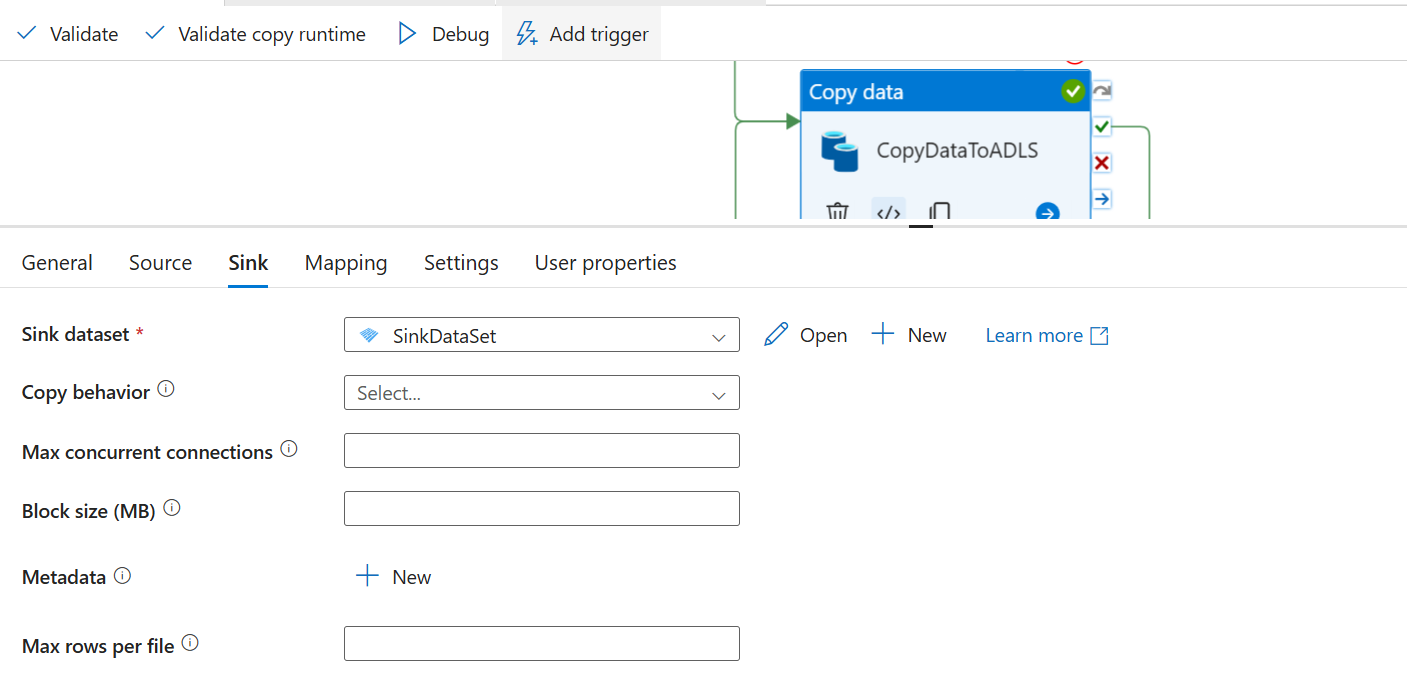


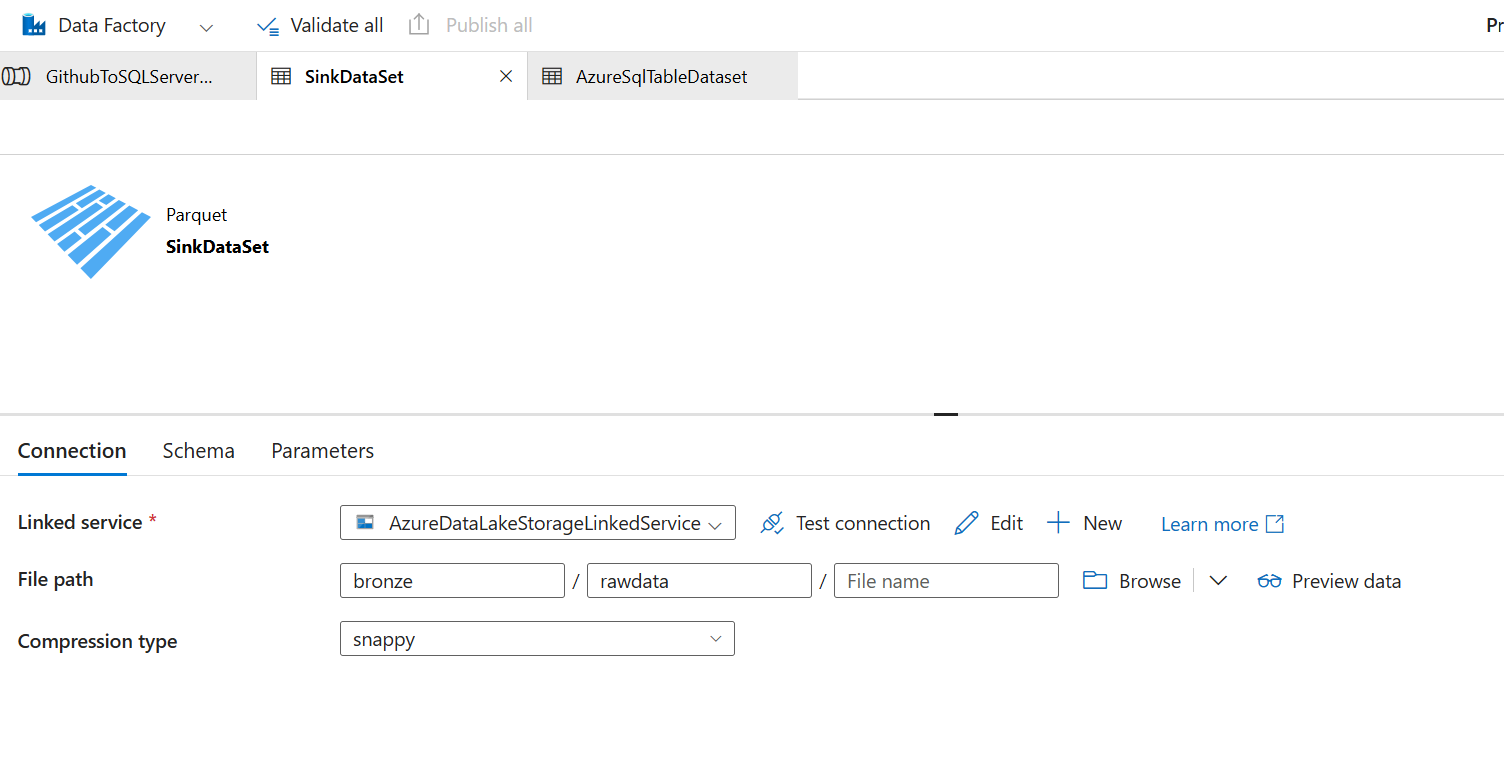


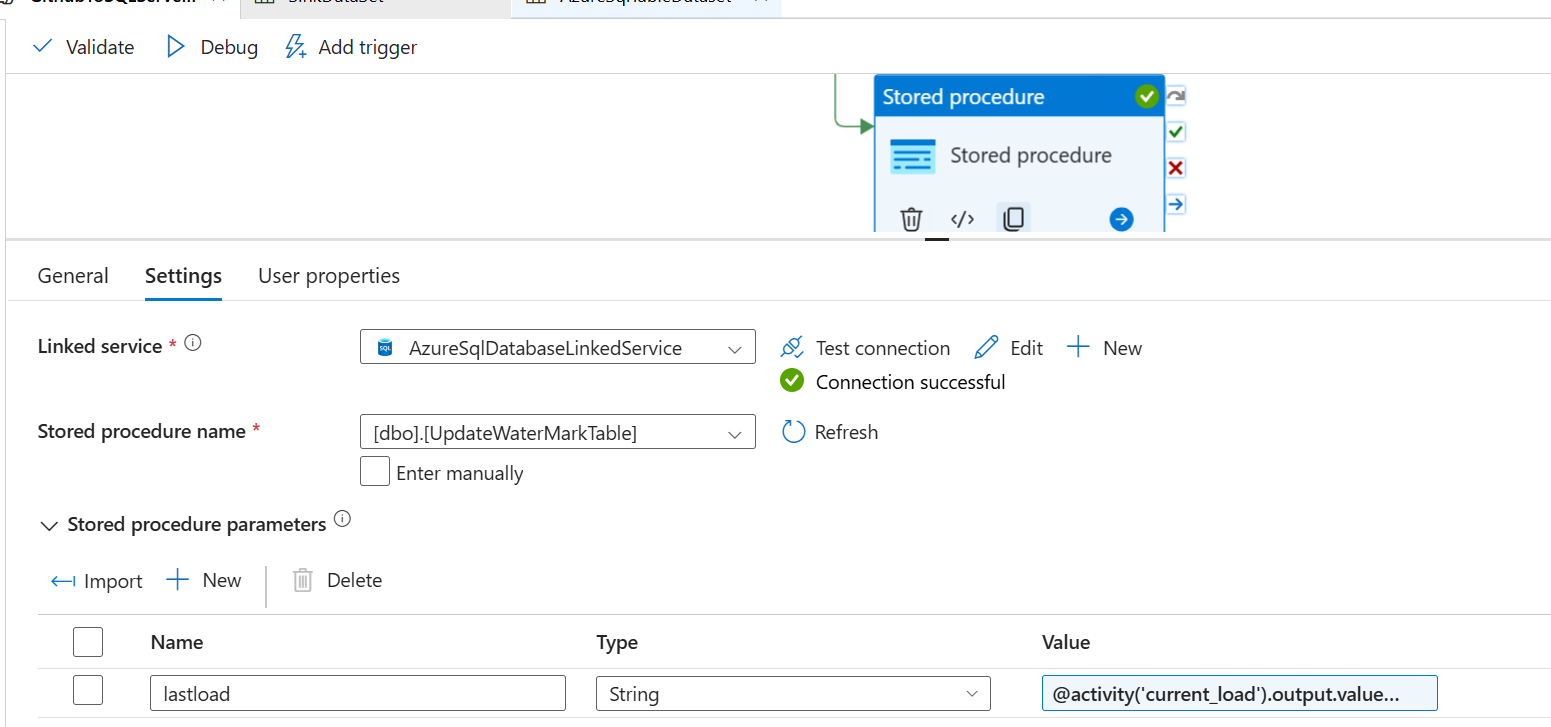


select \* from CAR\_SOURCE\_DATA where date\_ID > '@{activity('last\_load').output.value[0].last\_load}' and date\_ID < '@{activity('current\_load').output.value[0].max\_date}'

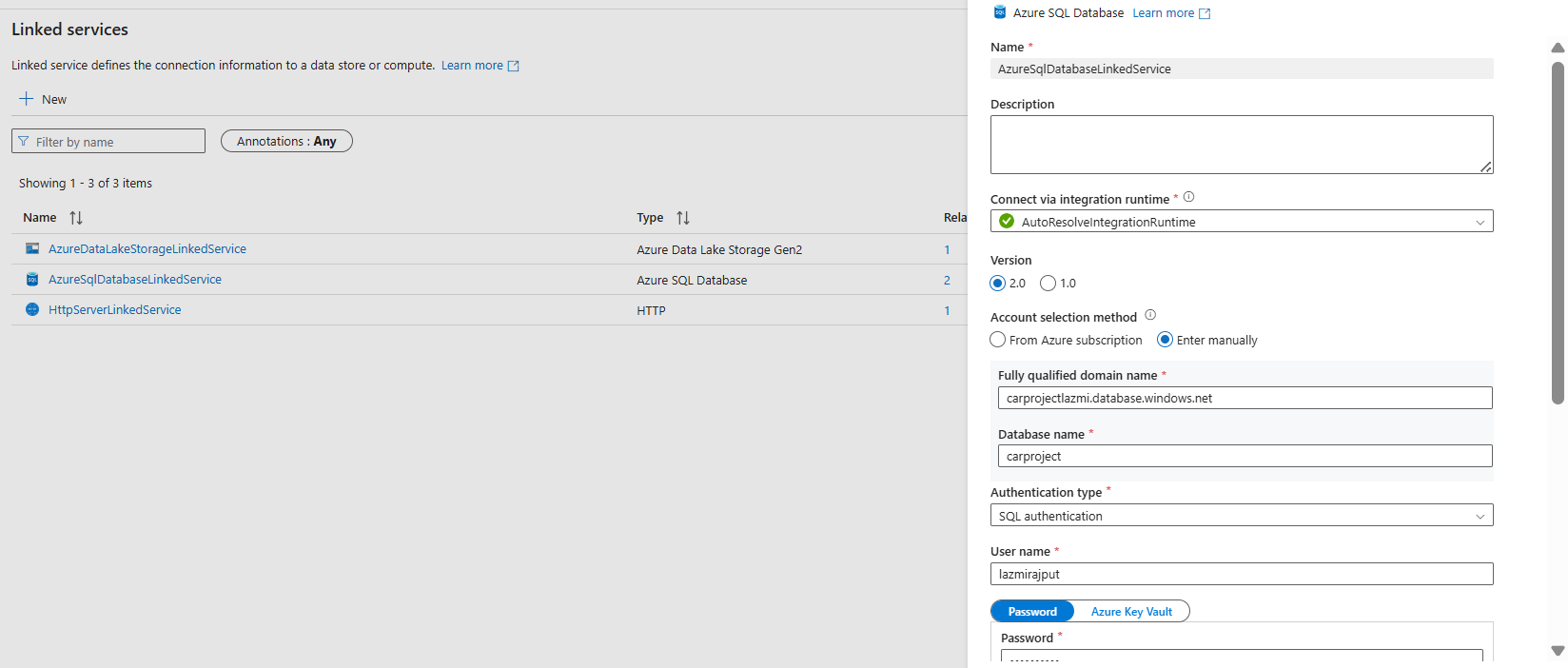








@activity('current\_load').output.value[0].max\_date



SECOND PHASE :

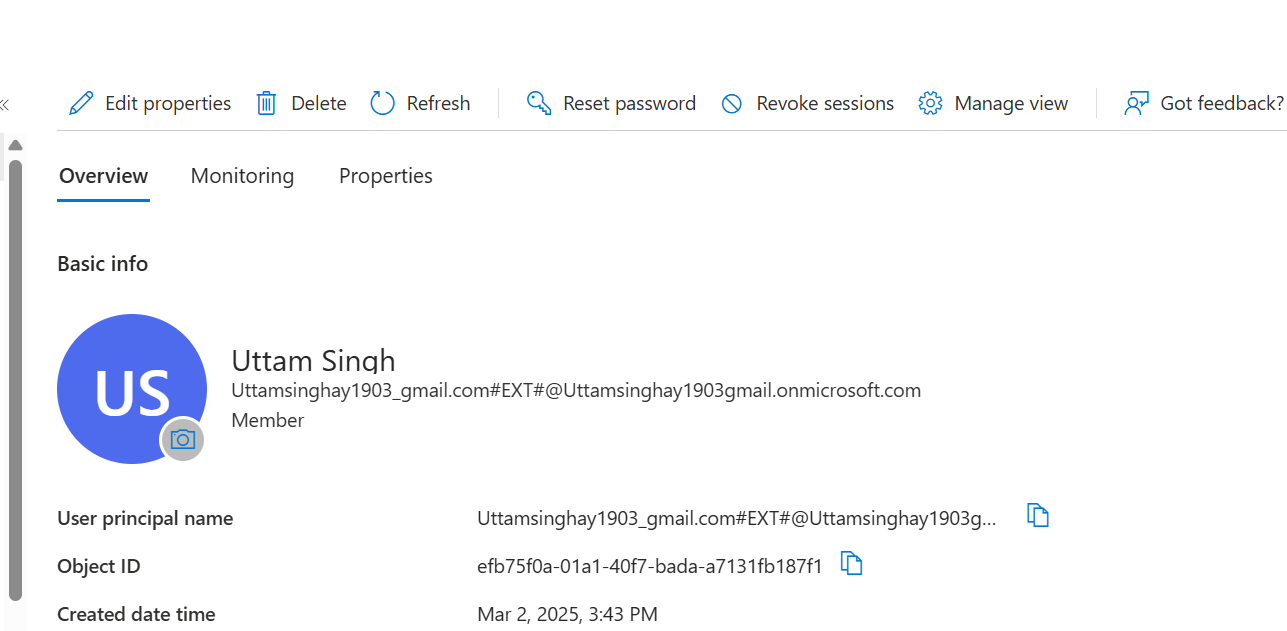
CFREATE DATABRICKS INSTANCE

What is Delta Live Tables?

Delta Live Tables is a declarative framework designed to simplify the creation of reliable and maintainable extract, transform, and load (ETL) pipelines. You specify what data to ingest and how to transform it, and Delta Live Tables automates key aspects of managing your data pipeline, including orchestration, compute management, monitoring, data quality enforcement, and error handling.

Delta Live Tables is built on Apache Spark, but instead of defining your data pipelines using a series of separate Apache Spark tasks, you define streaming tables and materialized views that the system should create and the queries required to populate and update those streaming tables and materialized views.

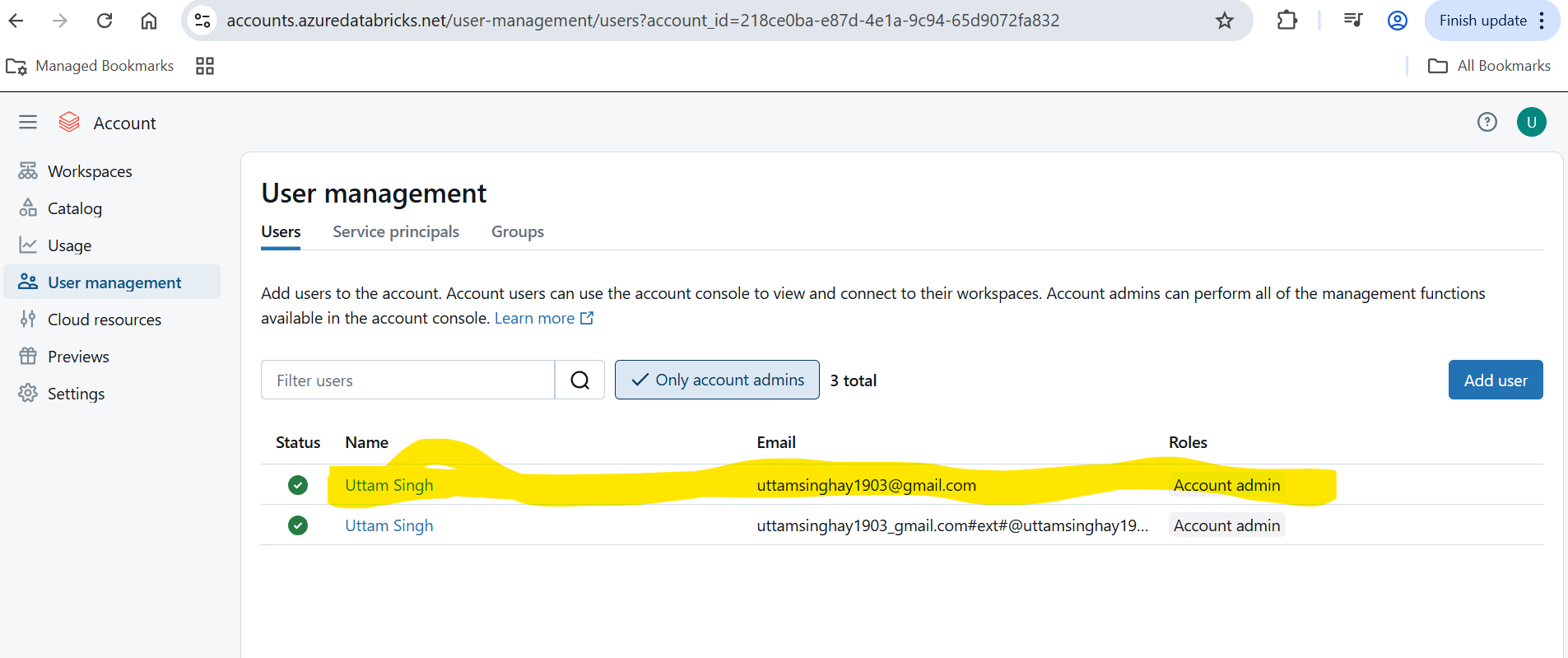
<https://accounts.azuredatabricks.net/>



Use this entra id to login.

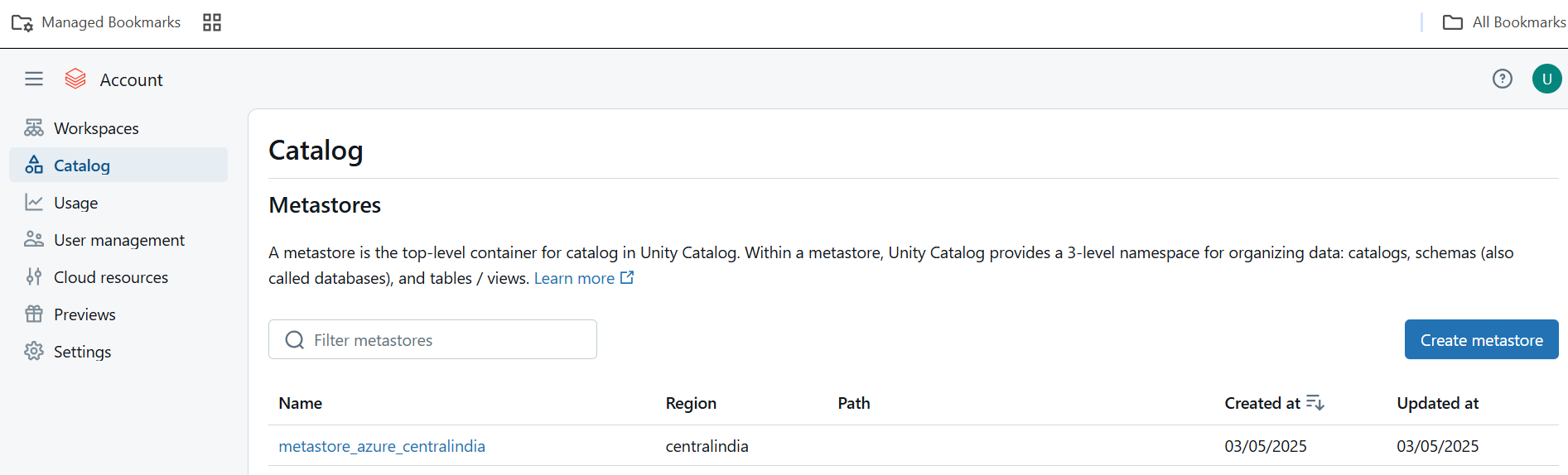
**A metastore is the top-level container for catalog in Unity Catalog. Within a metastore, Unity Catalog provides a 3-level namespace for organizing data: catalogs, schemas (also called databases), and tables / views.**

Create a unity meta store. It can only be created by using admin console.

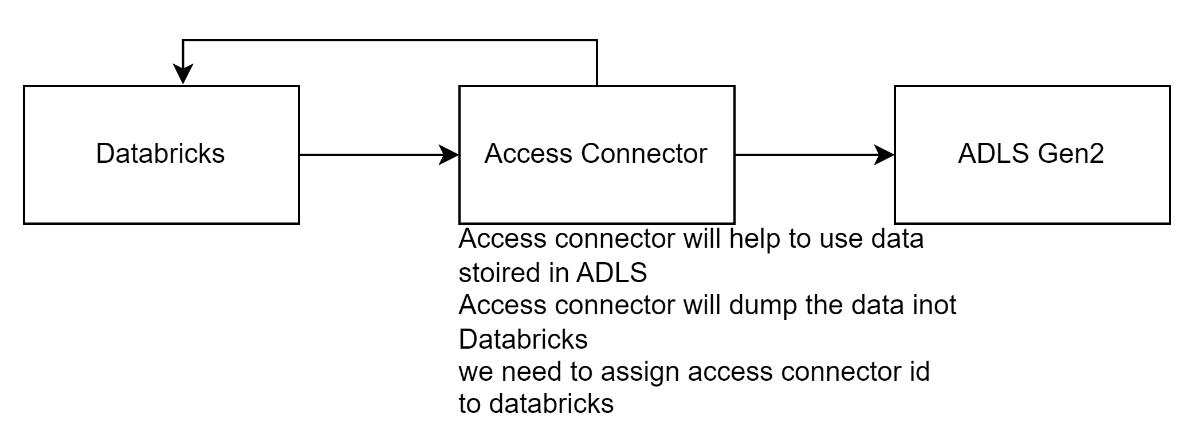


Add your as accountadmin so that we can see manage account in our normal databricks workspace

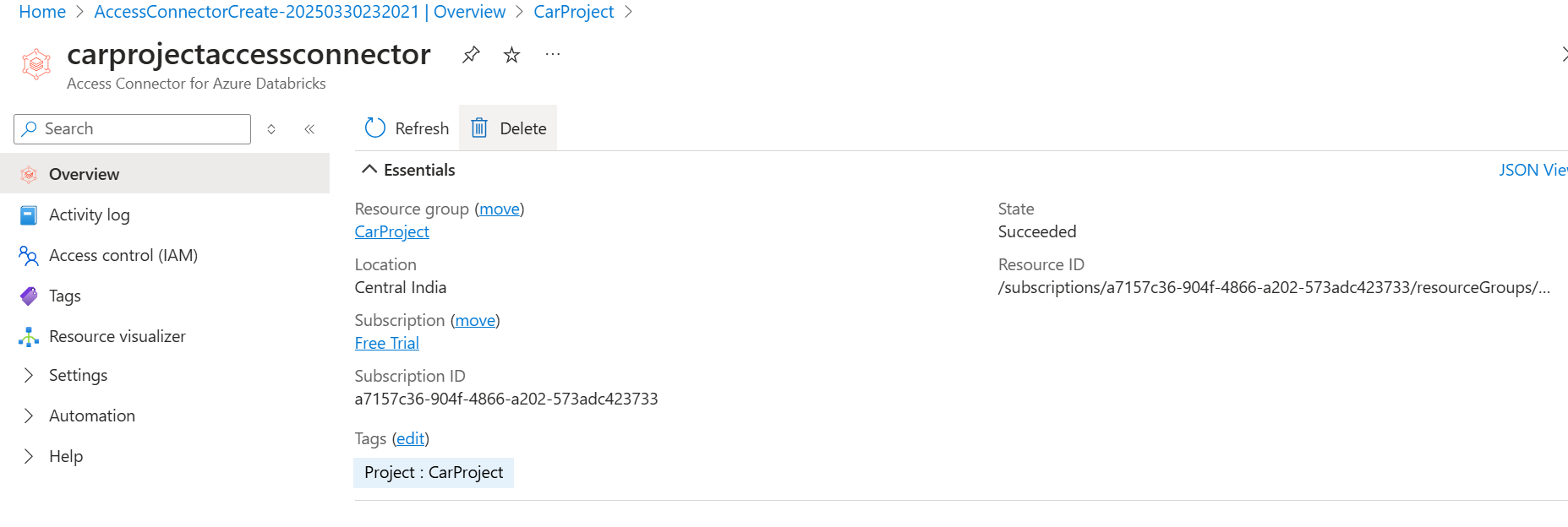
Whenever we create databricks workspace, it creates a default metastore for us.



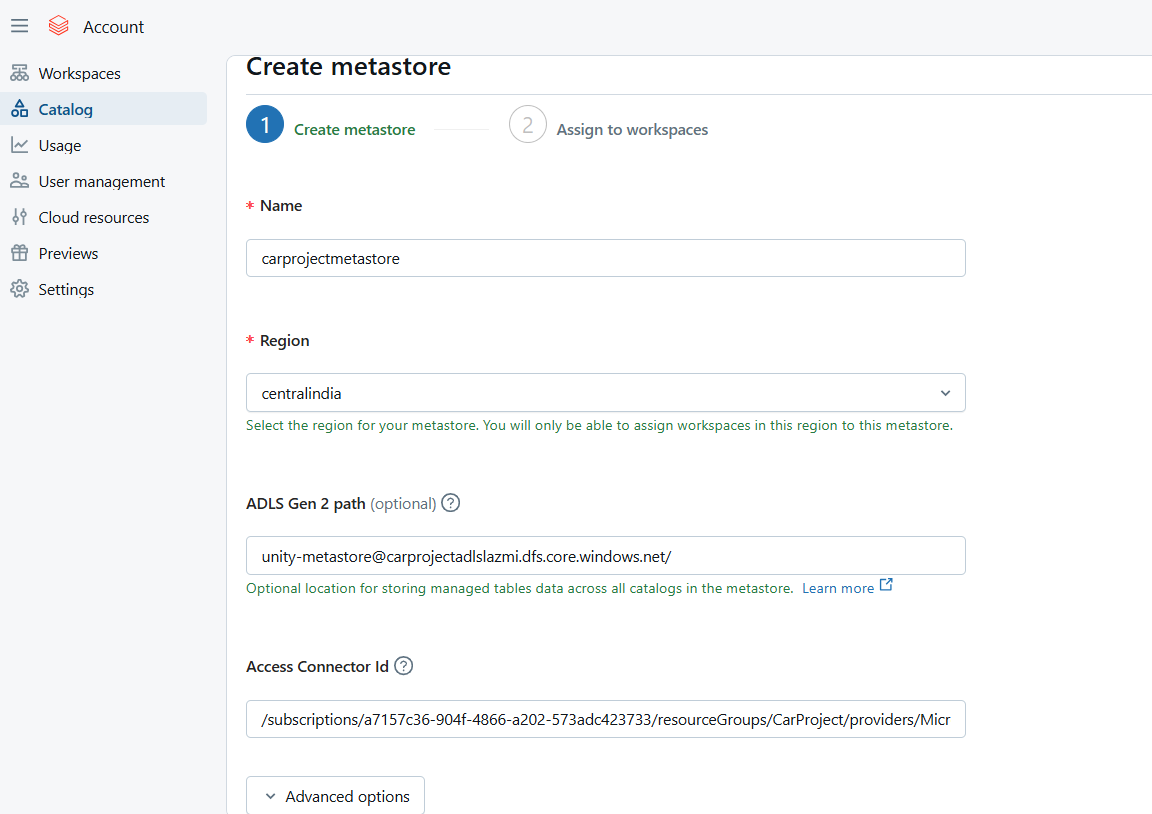
Create metastore.

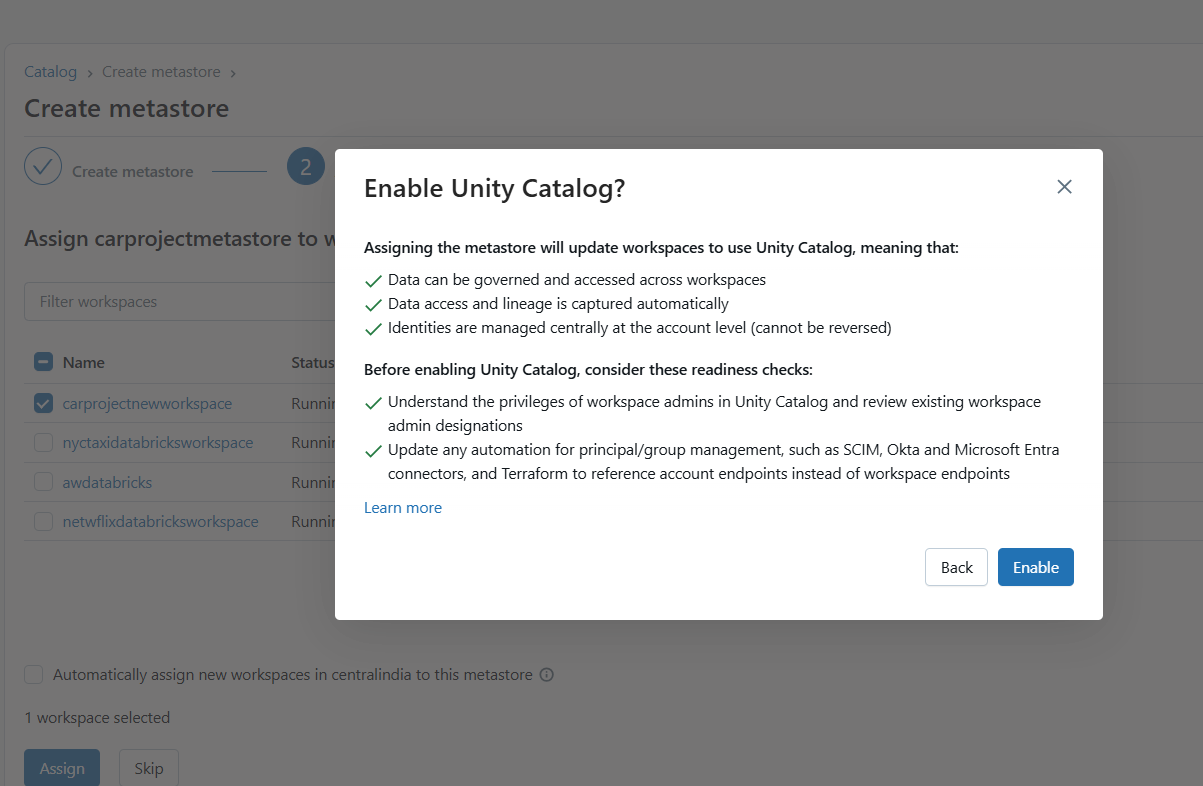


Create access connector for databricks.



Assign storage data blob contributor role on storage account to managed identity( access connector for azure databricks)





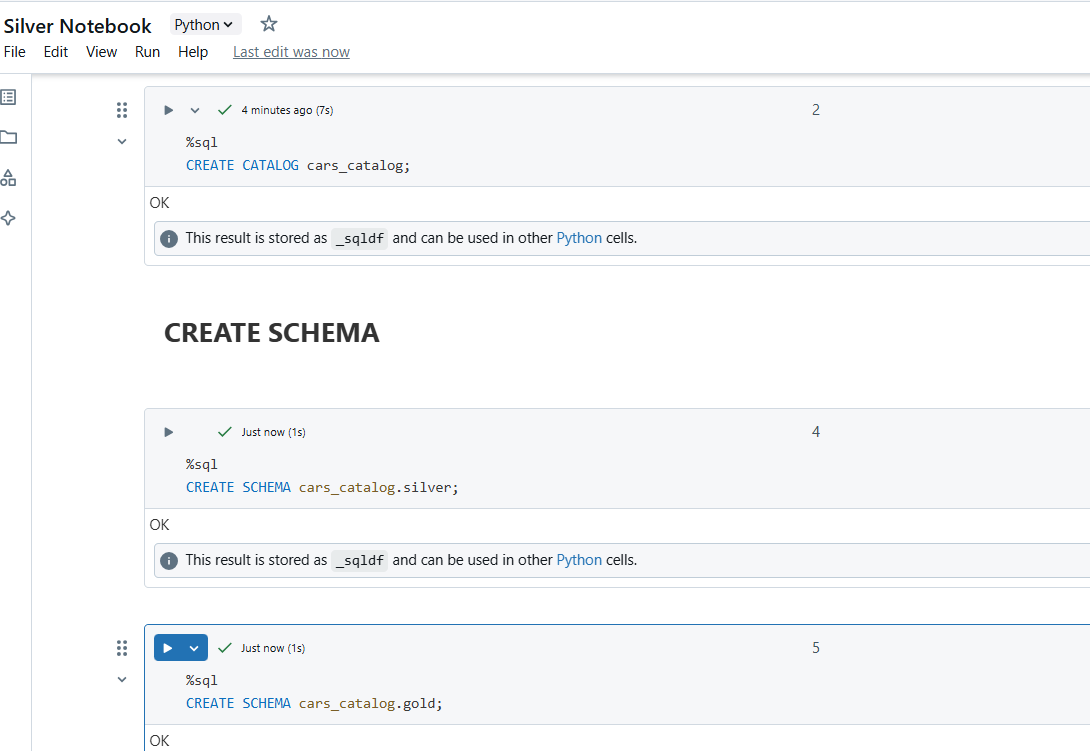
We can assign multiple workspaces to one metastore.   
Once metastore is created, assign normal user as admin as we are going to create external location by using that account only.

Create external locations for bronzr, silver & gold container.

Create compute, create silver notebook

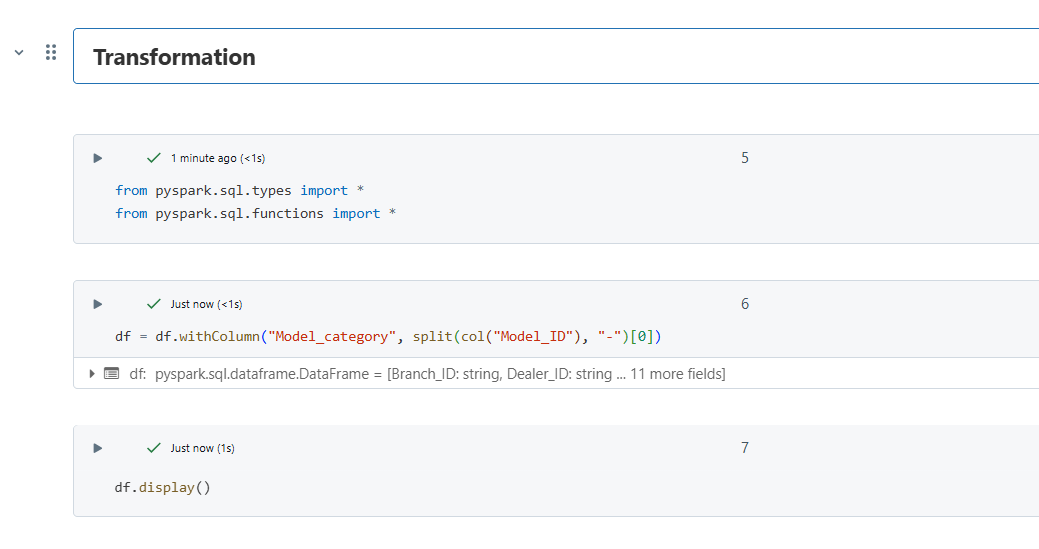
Create a catalog

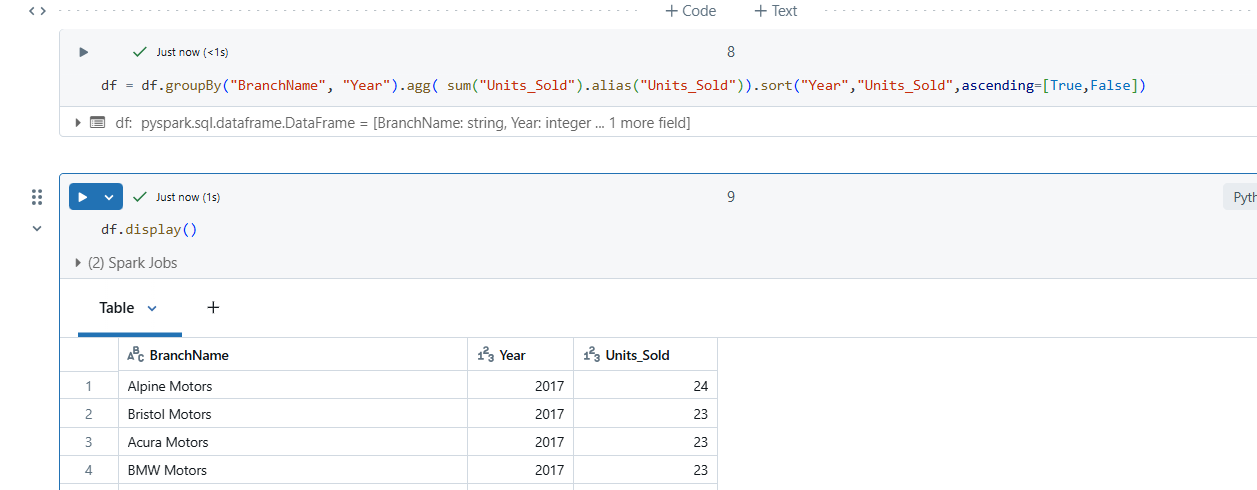
Create two different schemas inside it.(for gold and silver)



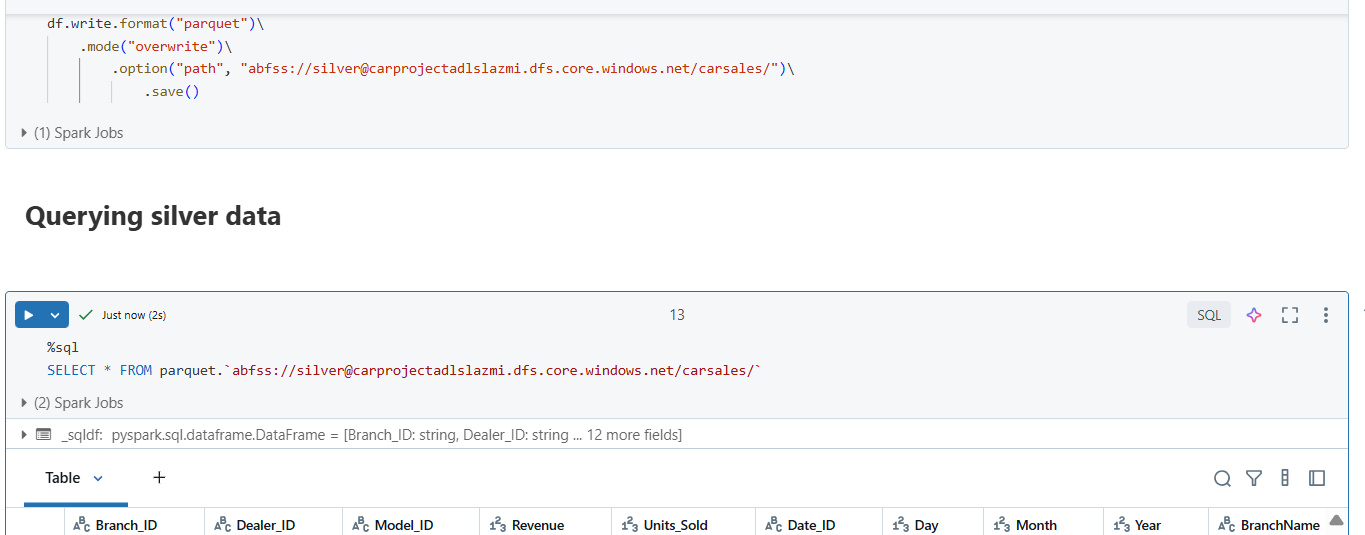
Read data from bronze raw folder

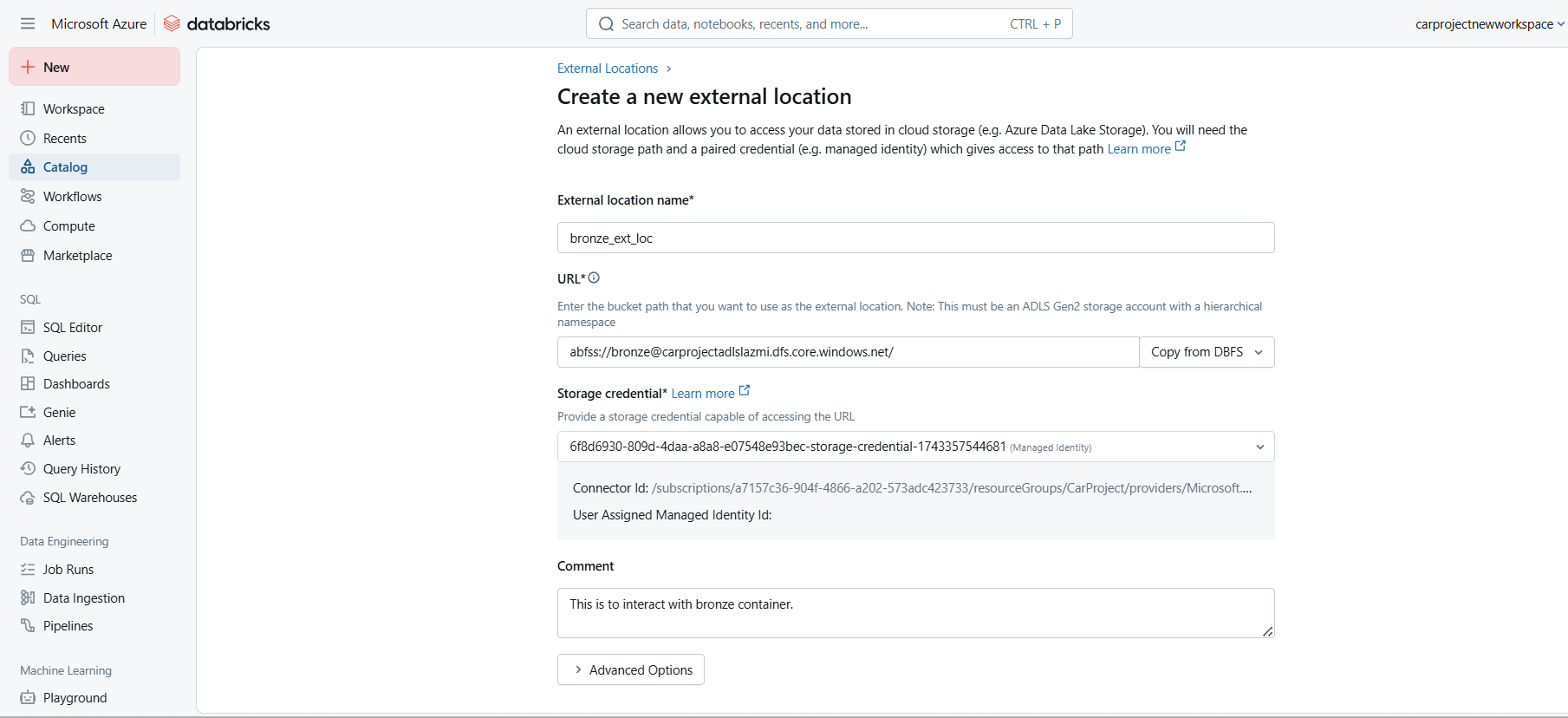






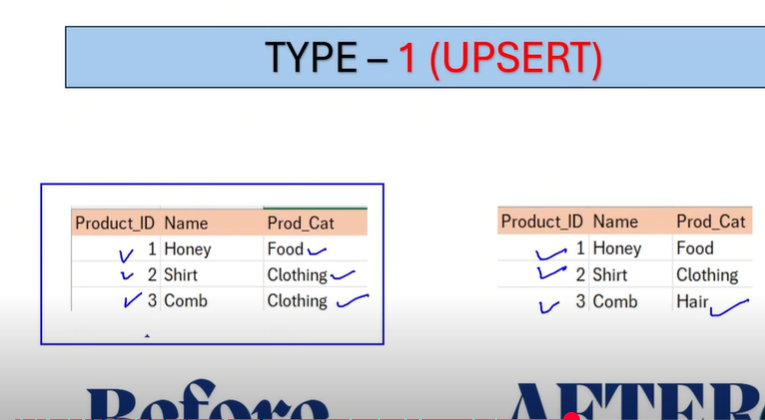






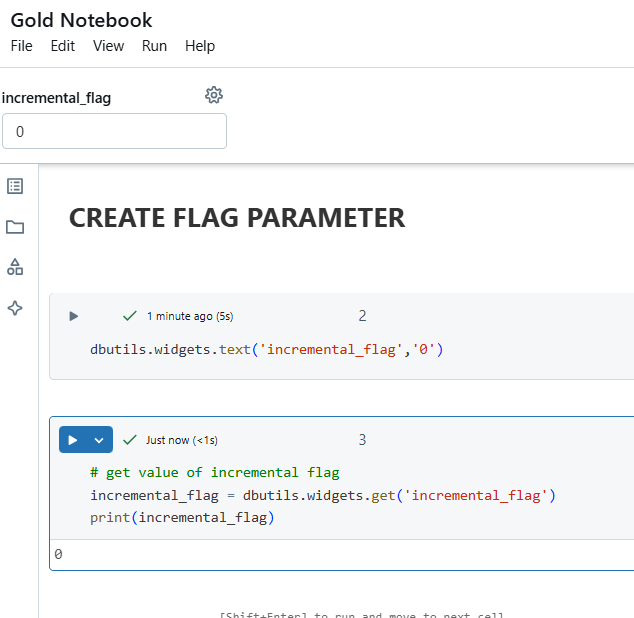
Create gold notebook.

Create a flag parameter to tell if it is initial run or incremental run.

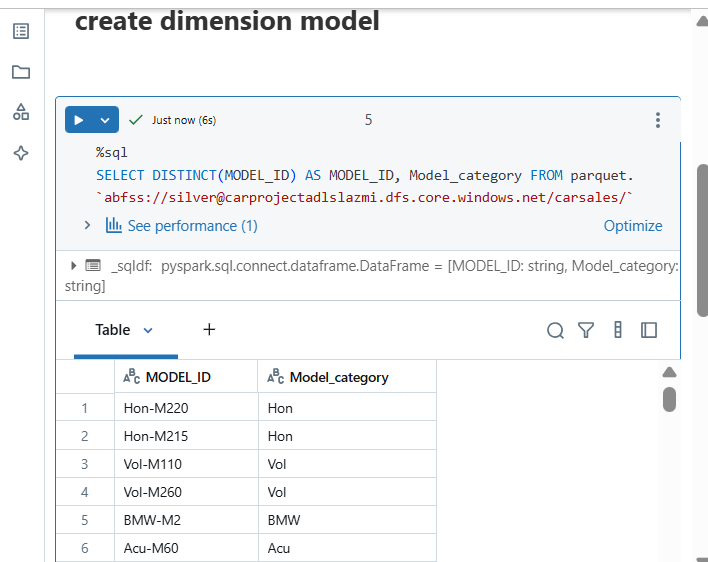


SCD – 1

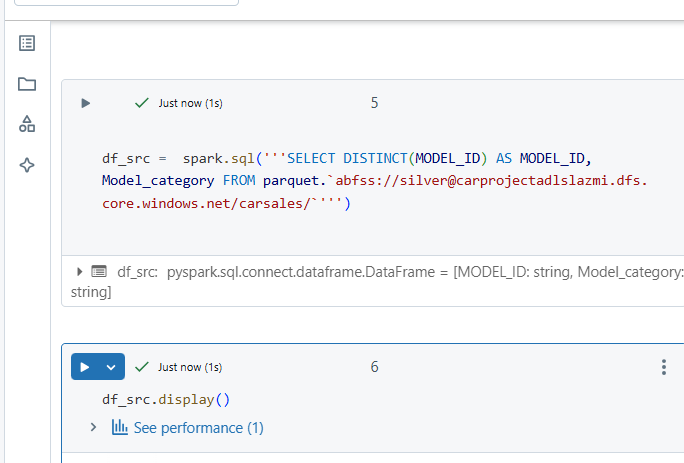
1. Create dim source by fetching related column from silver layer.
2. Create sink by adding surrogate key.
3. Join source and target.
4. Filter old and new records.
5. Create final df by applying union between old and new df
6. Write or update final df to destination

Create widget  


Create dimension model

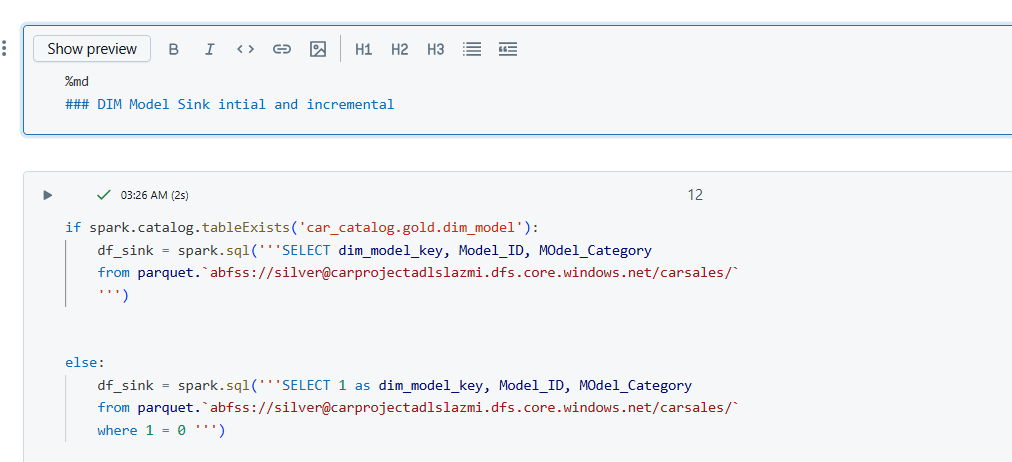


Create a dataframe on top of it

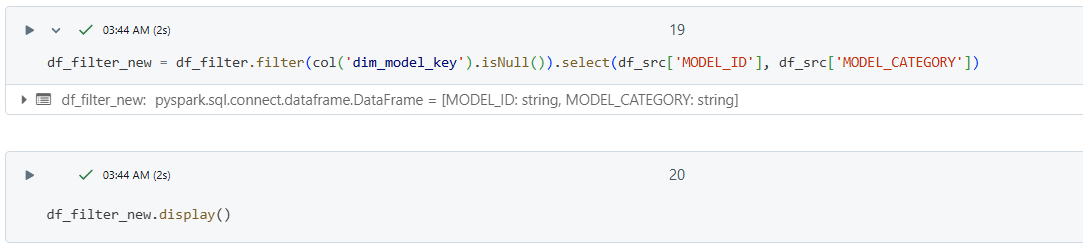




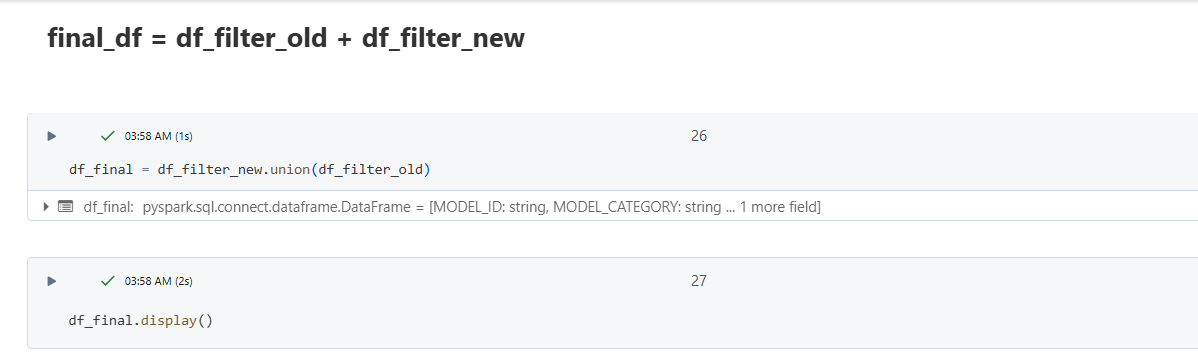
This logic is just ti get the template. We will just get schema as condition is always false. This is for initial load but for second time, we only want real data.





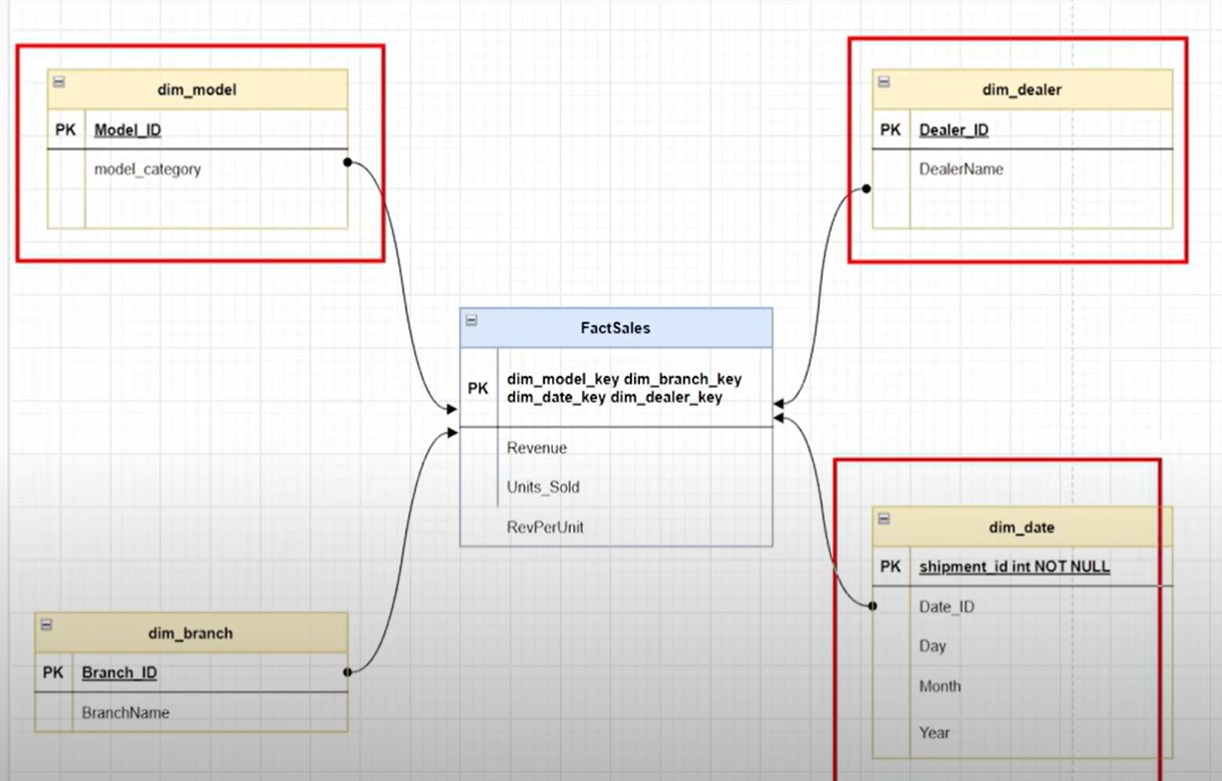




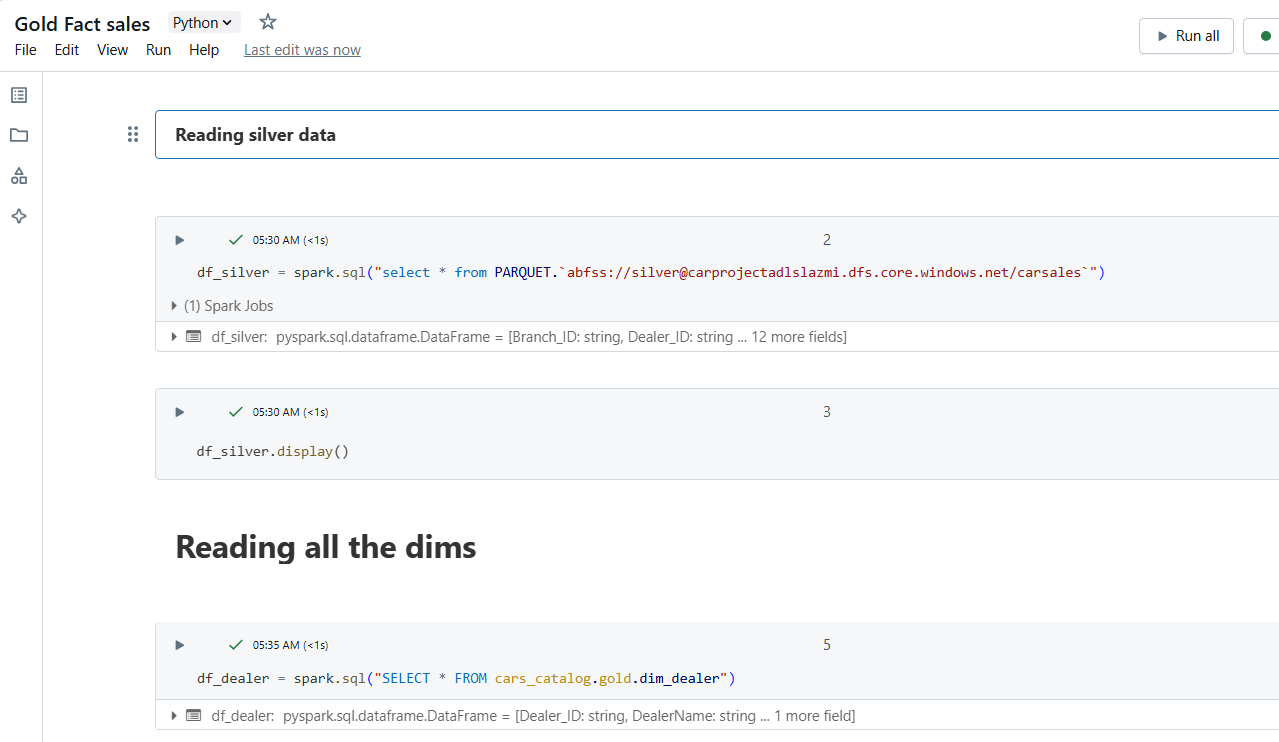






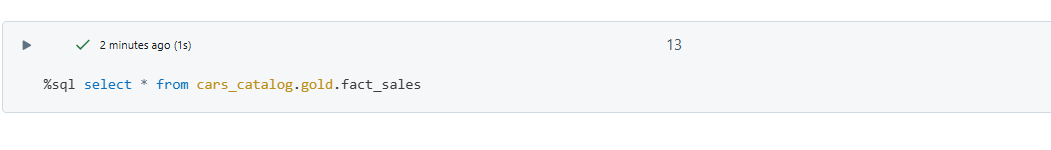
Create all the dimension table as above   


Creating fact table









Create workflow in databricks

Create tasks for each notebook and connect in below manner.

