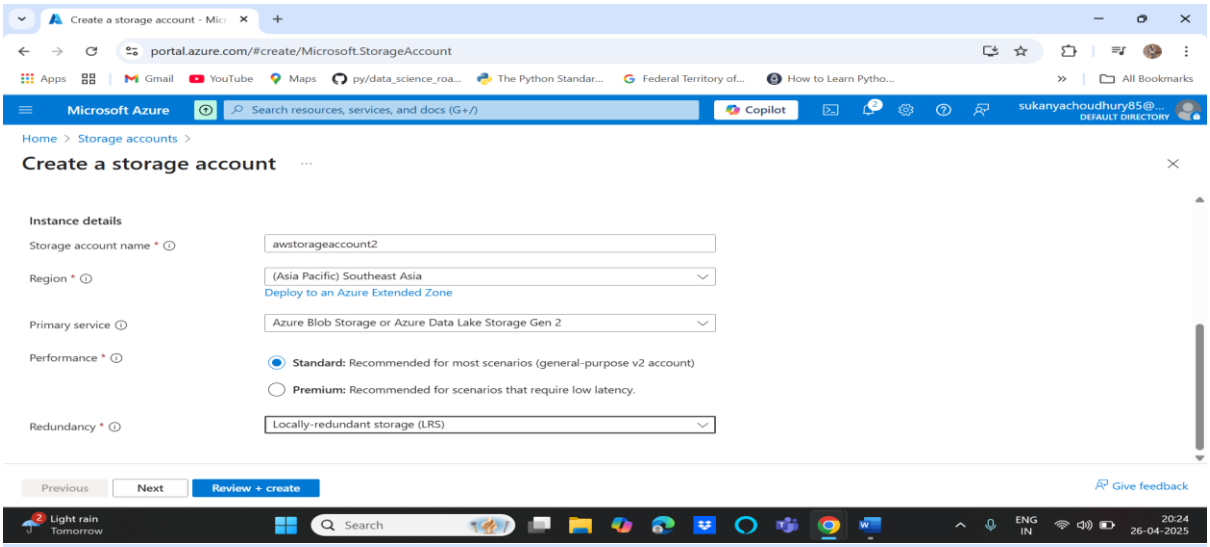
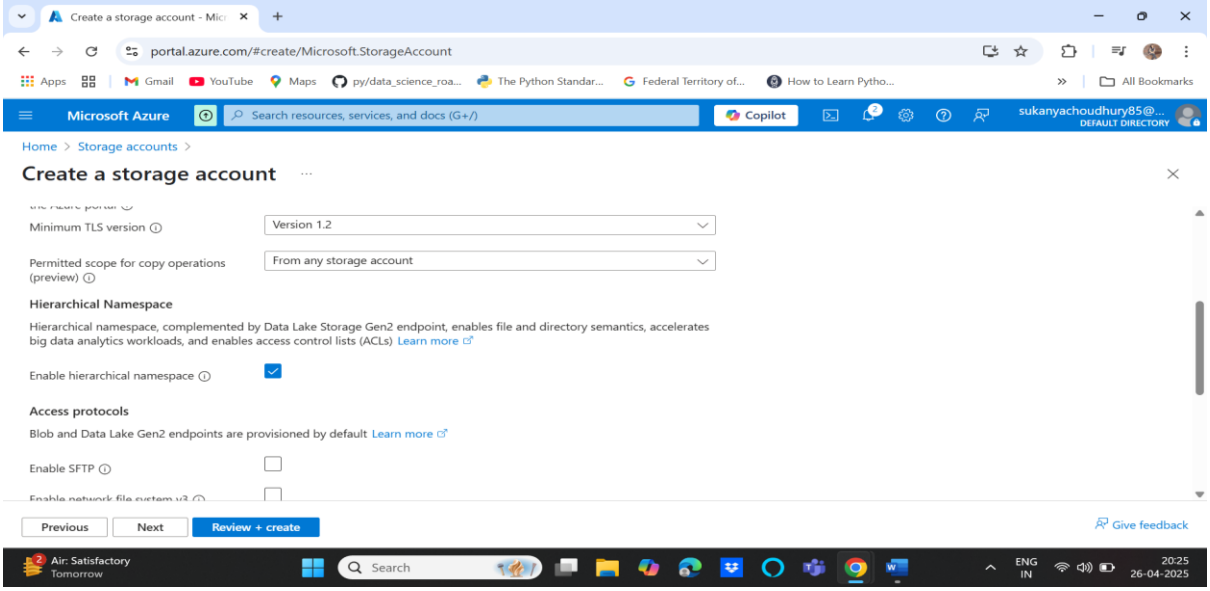
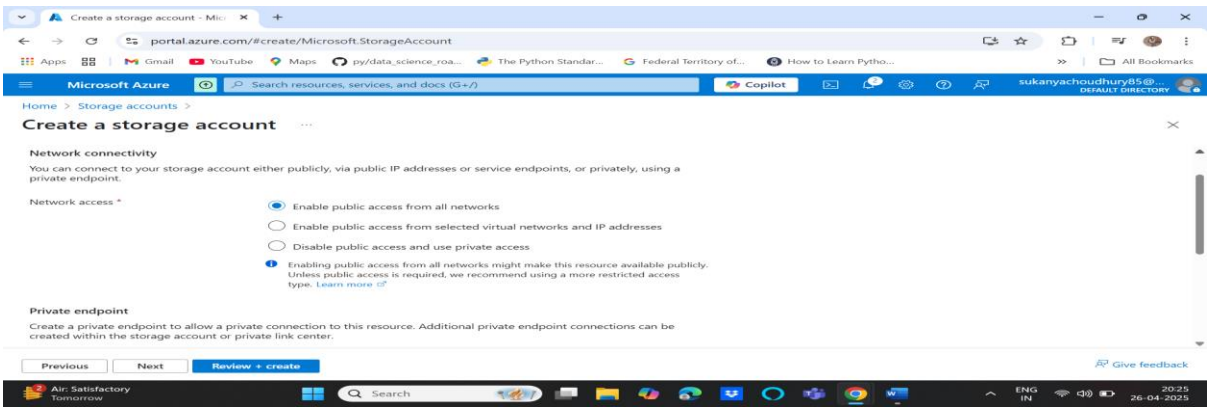
Azure End to End Data engineering project (github data)

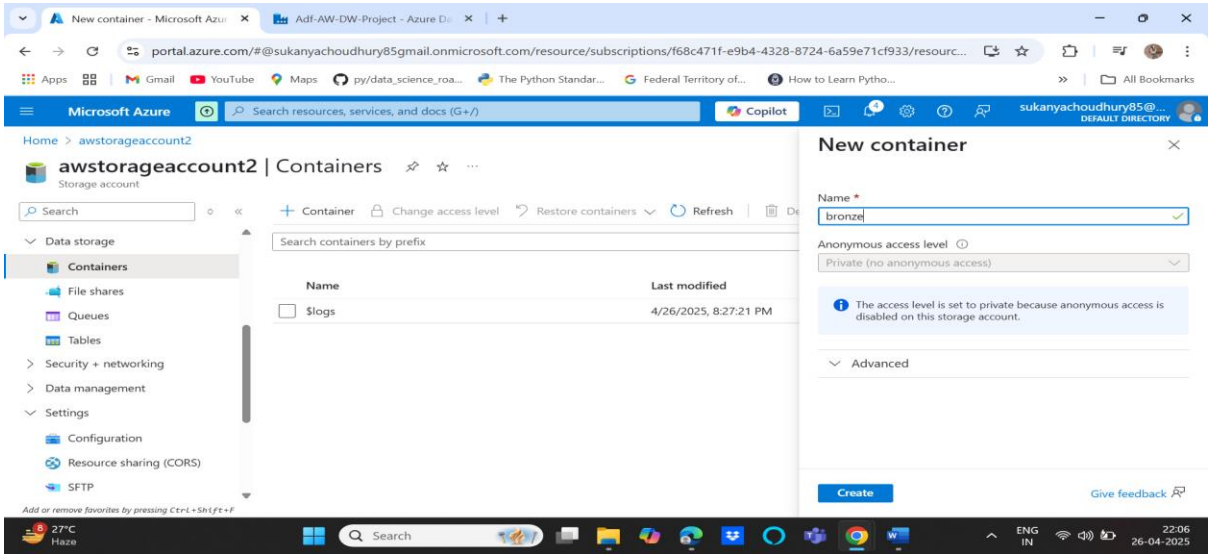
1) Setup Azure: Use new email get 30 days free with $200 credit
2) Create resource group→storage account

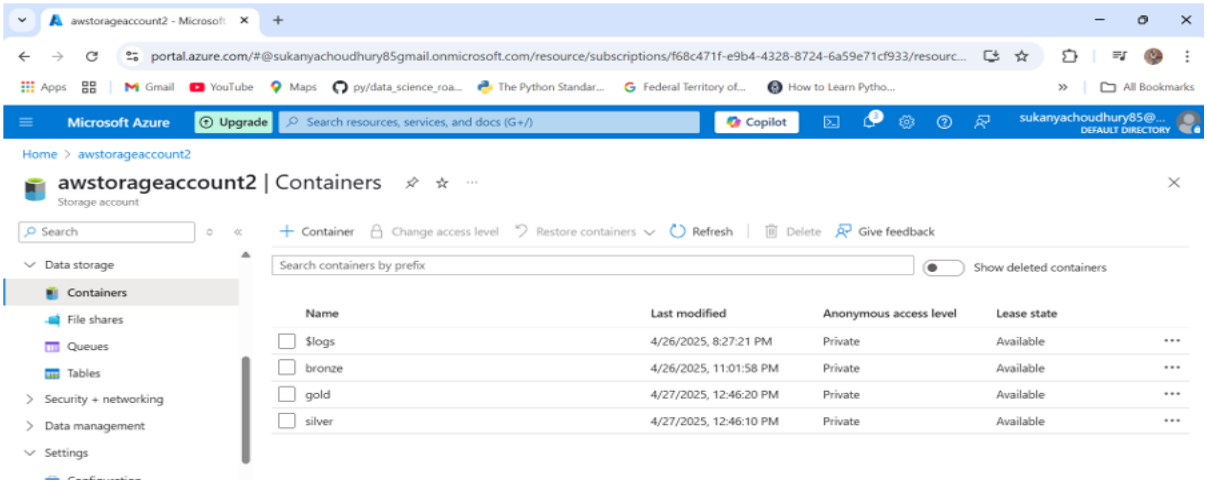






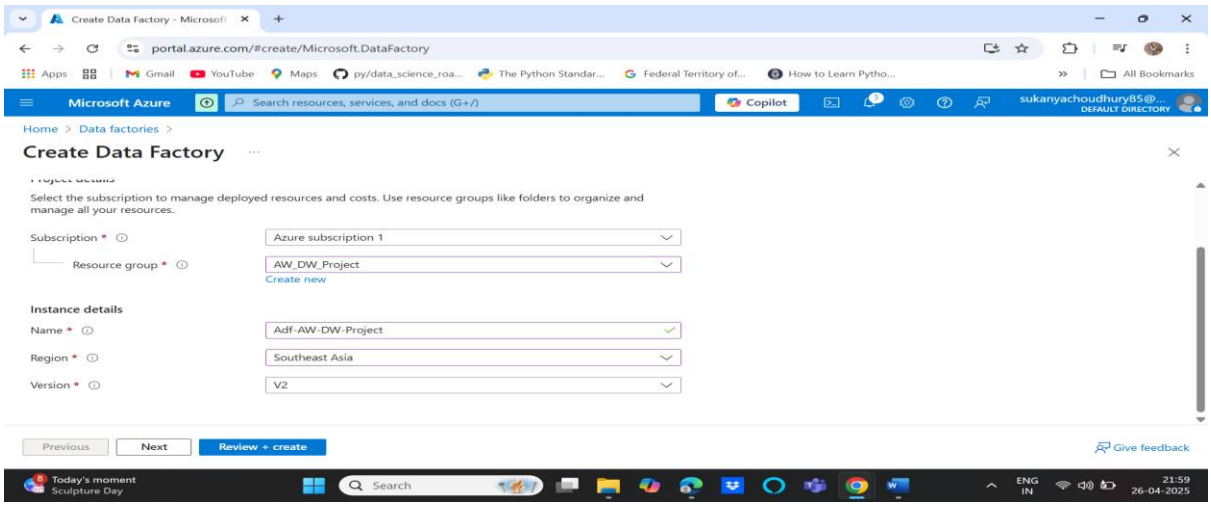
3) Create Containers (bronze, silver, gold) inside storage account





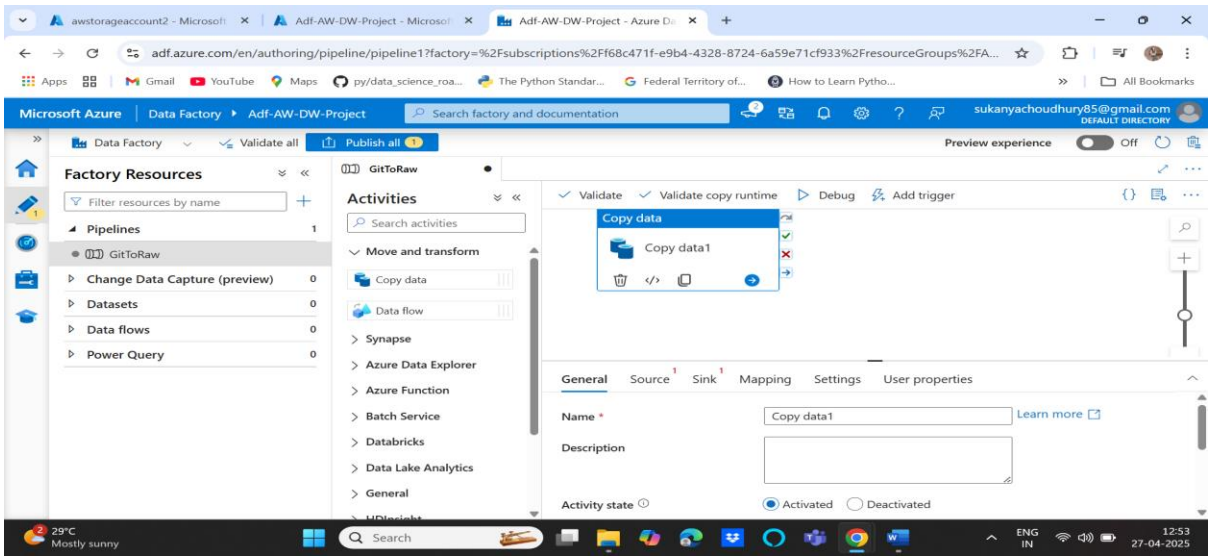
Phase 1 : Data Extraction from github using Data Factory pipelines

1) Create Data Factory “Adf-AW-DW-Project” as below:

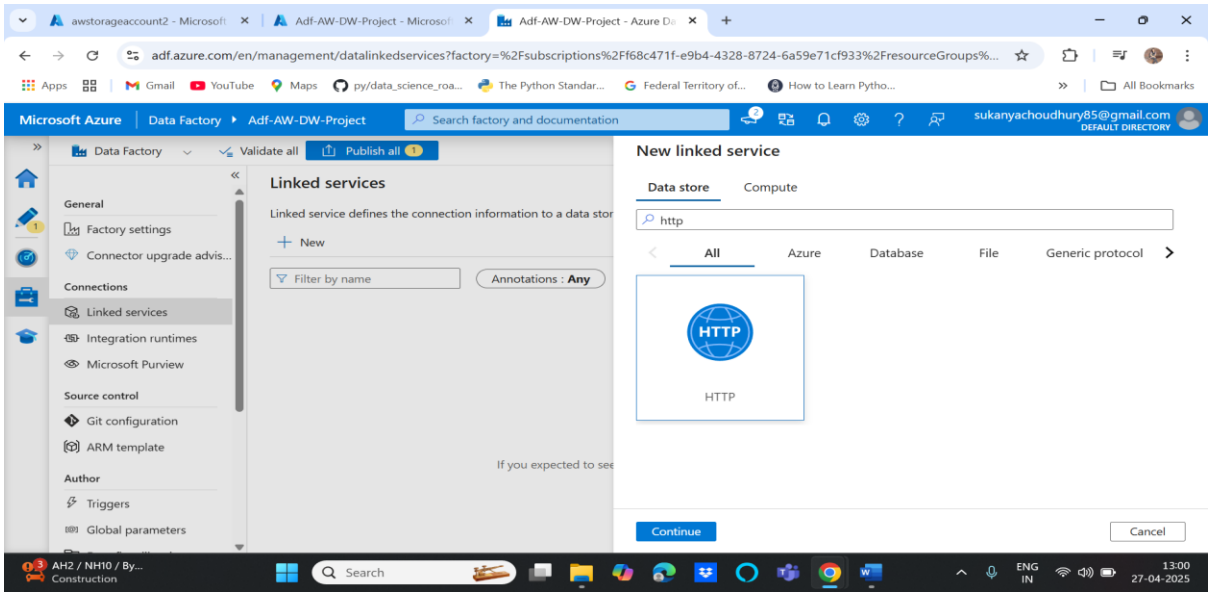


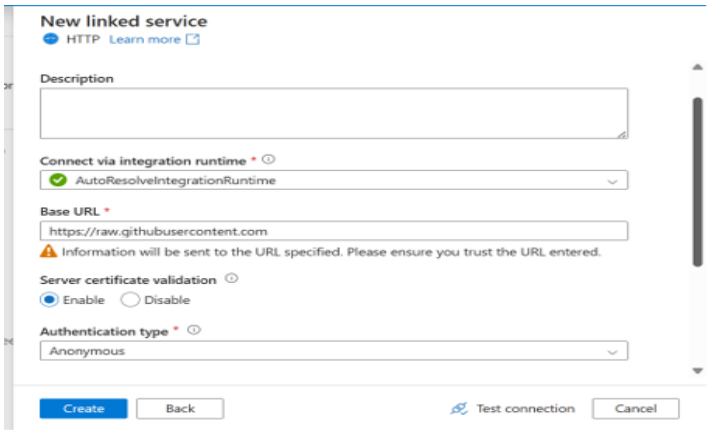
2) Create a pipeline in data factory

DF→ Author→ Pipeline→new pipeline and then move and transform→ Copy data

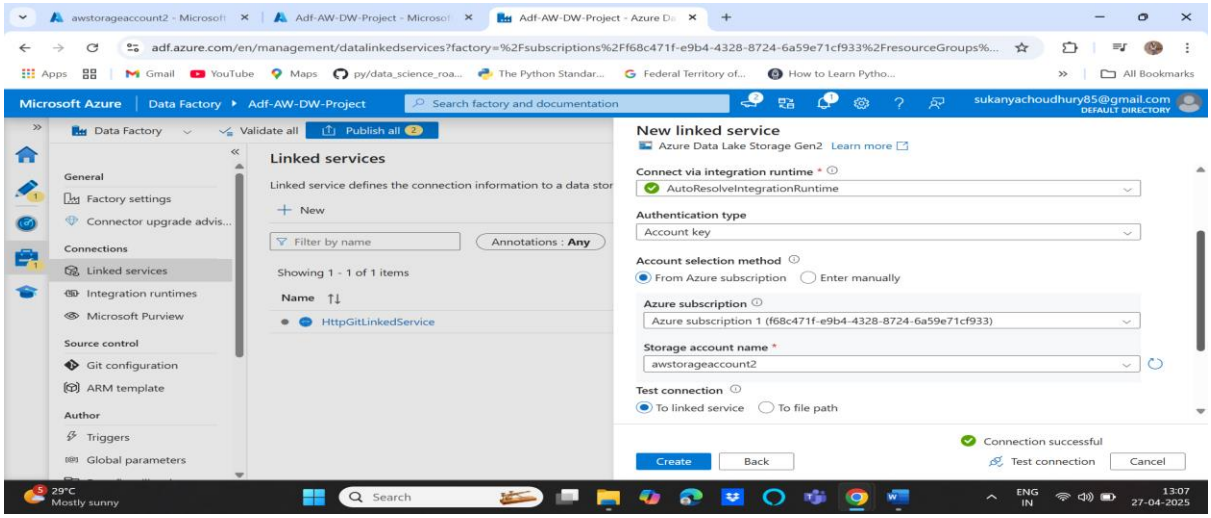


3) Create linked services for the pipeline to work e.g. for source data source create http linked service to fetch the github data set and give base url as the github main url



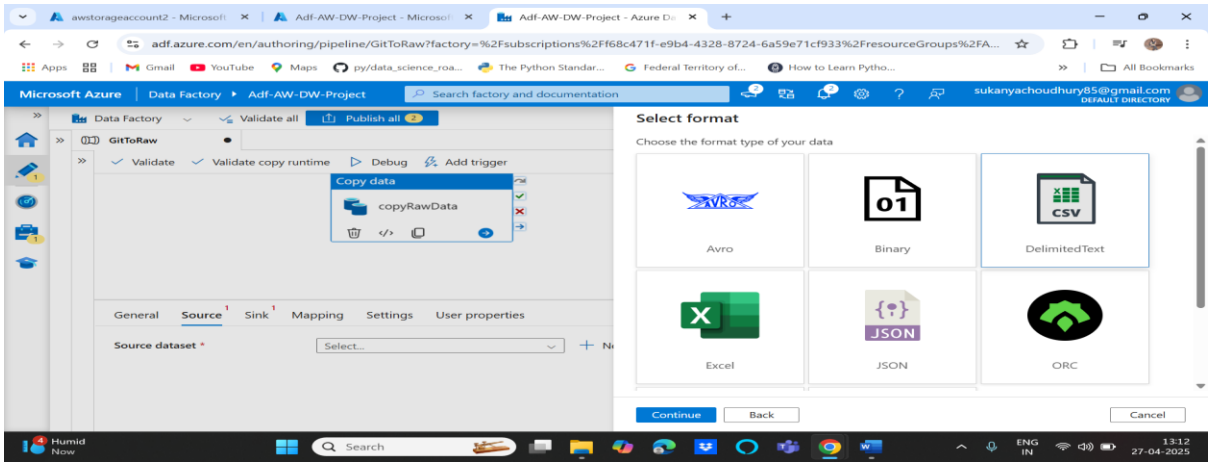


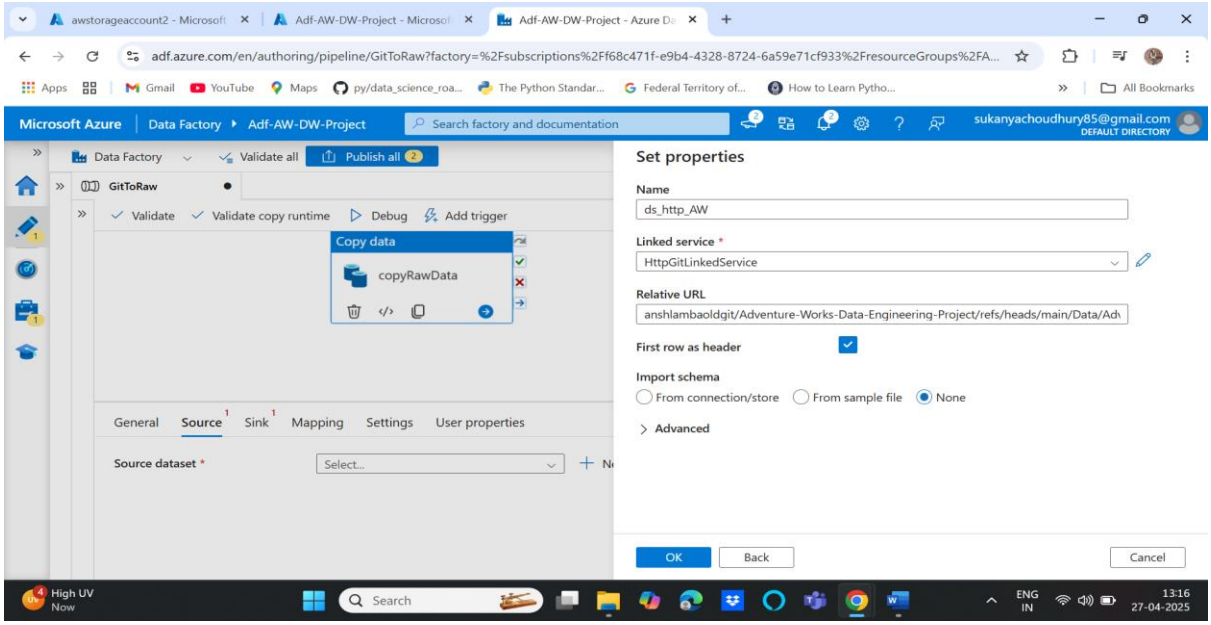
For destination(sink), create Azure Data Lake Gen2 as the linked service



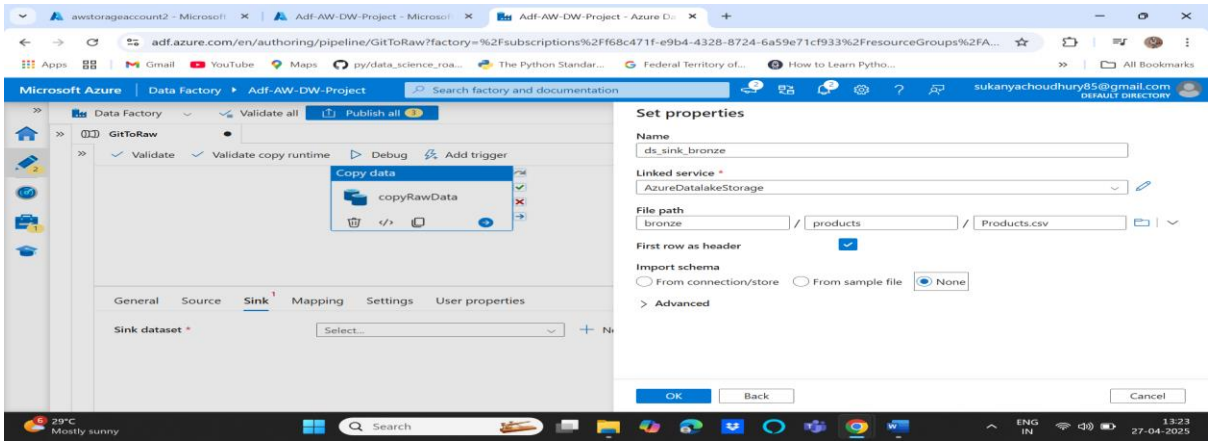
4) Create source and sink dataset inside copy pipeline

For source its http->CSV file format as it's http type data source with a file format of csv then give the linked service created above with the relative URL as below

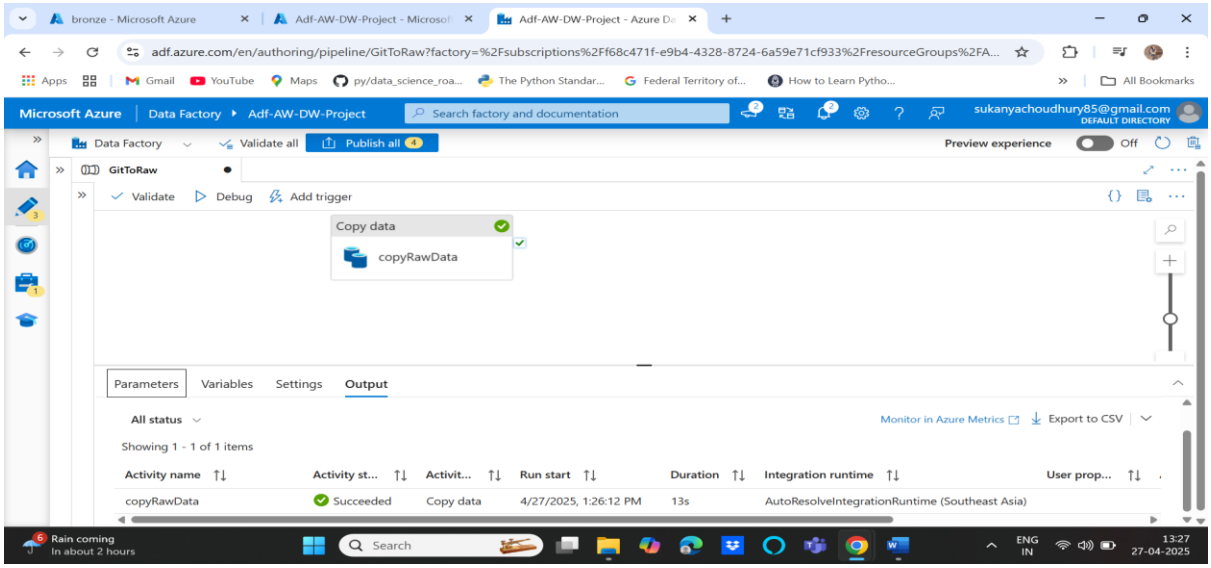




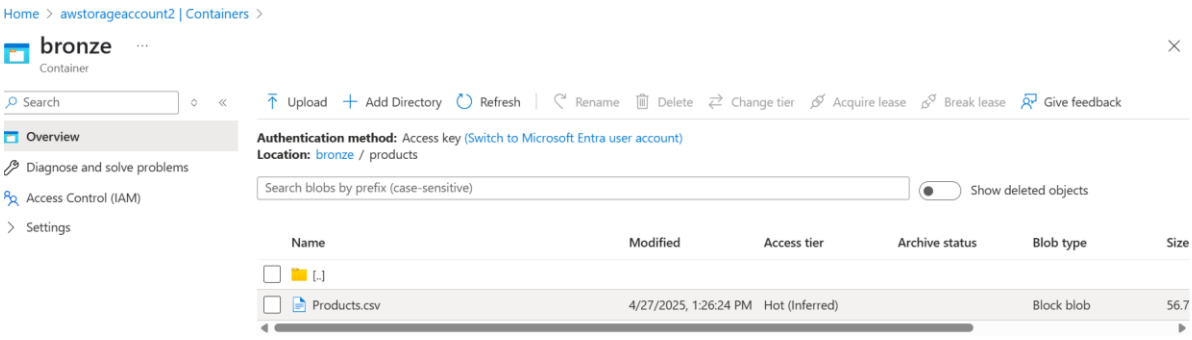
For sink again same process but the datasource type will azure data lake storage gen2 and format is csv. Then again select the linked service created for destination with the folder where you need to store it. Eg: here, bronze container/products and Products.csv



Validate and debug to run the pipeline and then publish all to save the changes



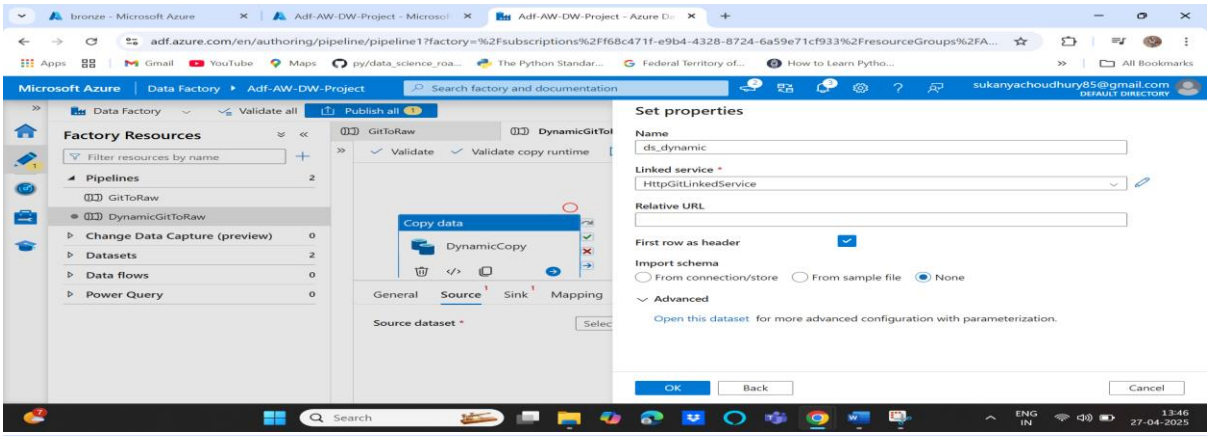
Products.csv is created under bronze container:

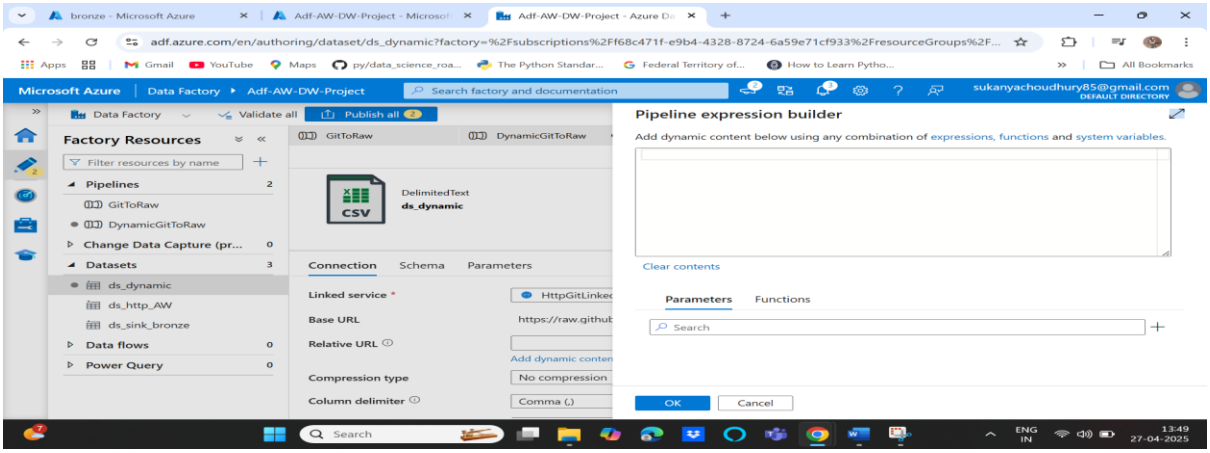


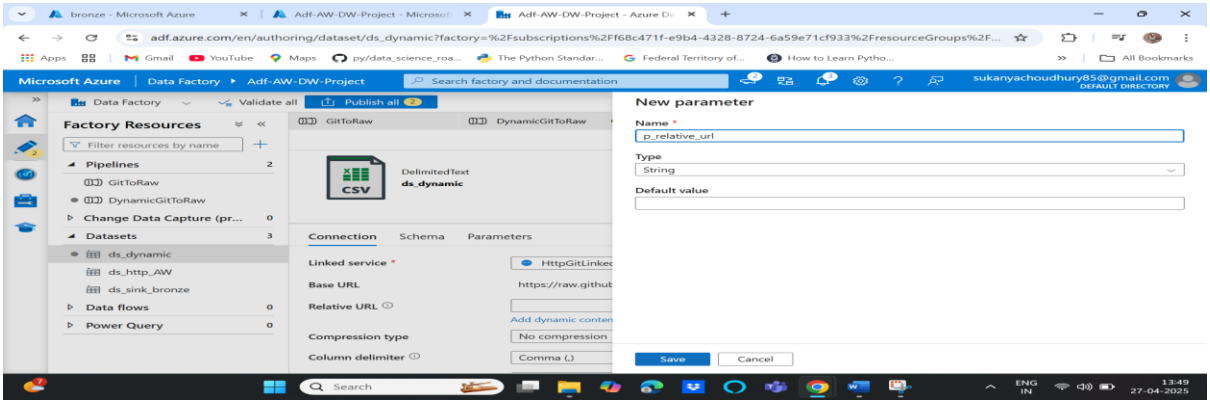
5) Dynamic pipelines to fetch multiple files using iteration and conditionals under author tab in Df. Create Parameters for the dynamic part of the pipeline e.g.: the relative URL for each file will change , the destination folder as well as the filename will change so we need to create 3 parameters and use a parameterized copy activity and a single data source to fetch all the files.

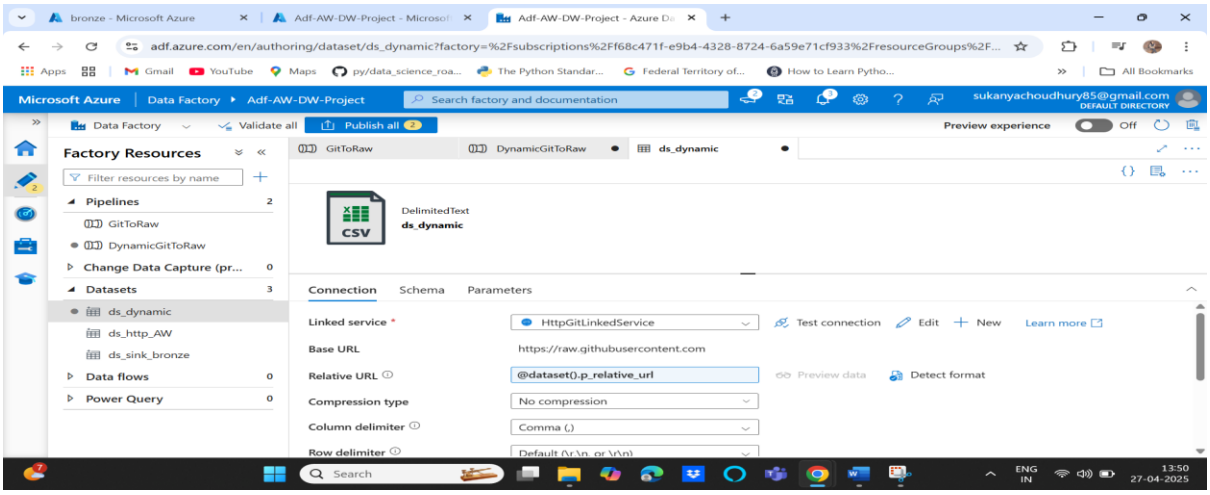
   i) DF→Author→ Pipeline→ New Pipeline→Copy activity→ source dataset→http→csv→ source linked service create above in step 3 as the base URL is same for all so same linked service will be used

Instead of giving a hardcode value for relative URL, we need to use parameters so for that click advanced→ open this data set→Relative URL→add dynamic content→parameters—new parameter and then select the parameter on expression folder

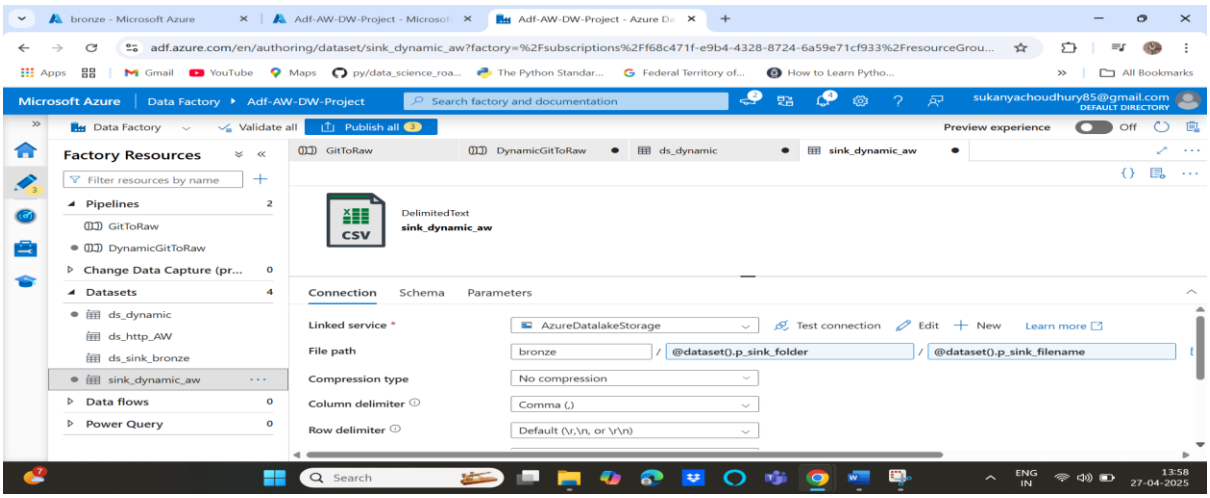




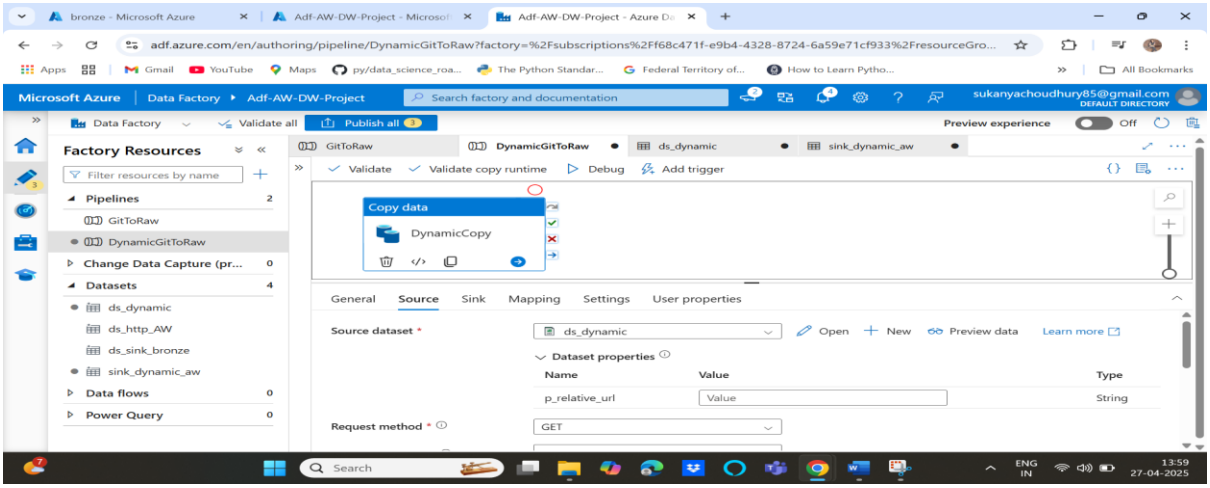


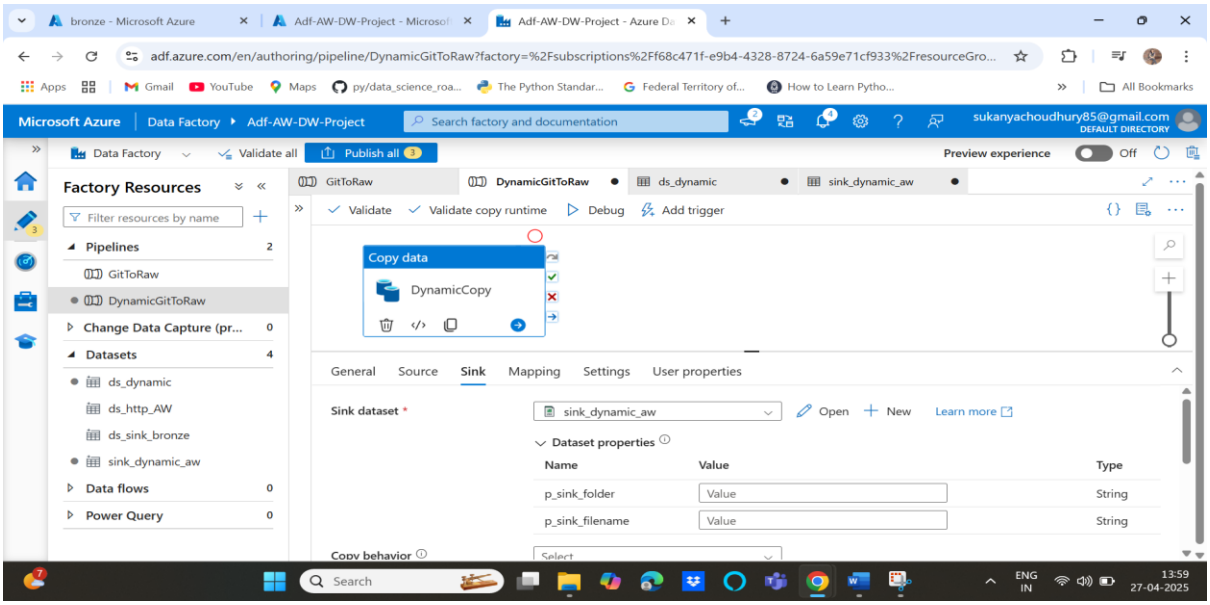


Similarly for sink dataset , create parameters for folder and filename by going to sink tab of copy activity→ add new→ data lake gen2→ csv→ select linked service in step 3→ instead of hardcoding the folders and filename→ advanced→open this data set→ folder, file→ add dynamic content and create two parameters as shown below:



Ultimately copy activity looks like below with dynamic source and sink data sources:





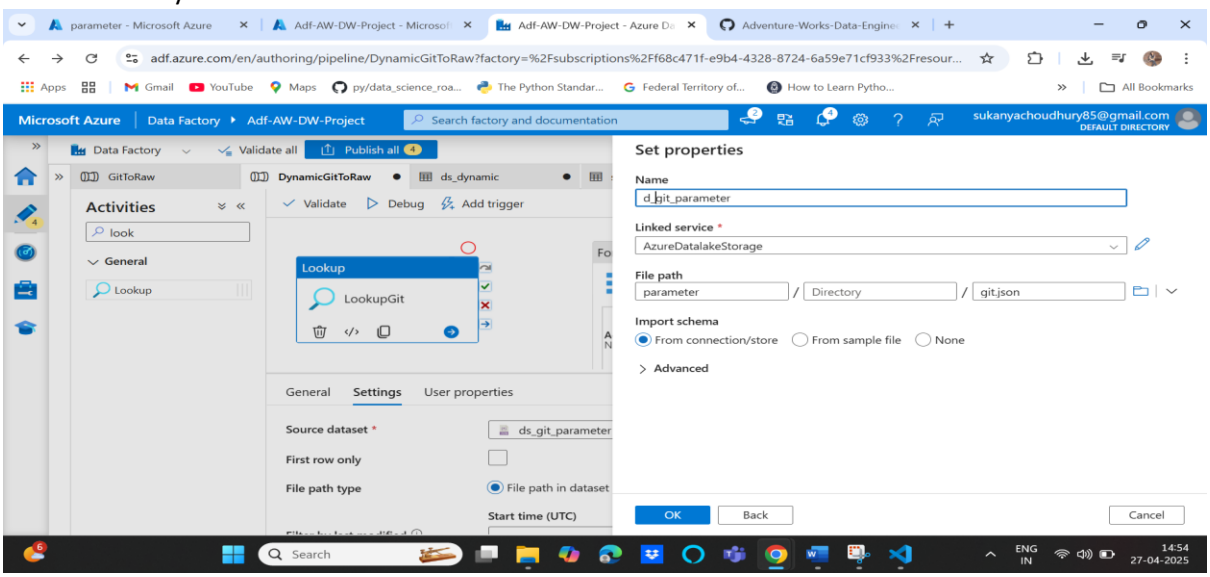
ii) Now, we need to pass values to these parameters:

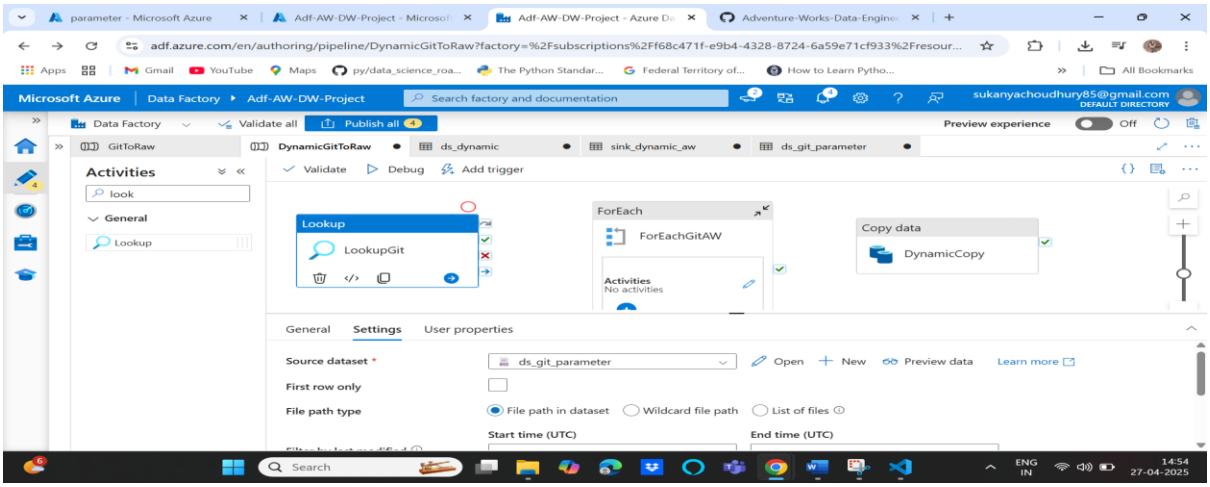
Select the pipeline under activities→Iteration and conditionals→ select for each activity and add it to the pipeline

In settings of the foreach activity pass a json file with a dictionary of parameters and their values in the items tab.

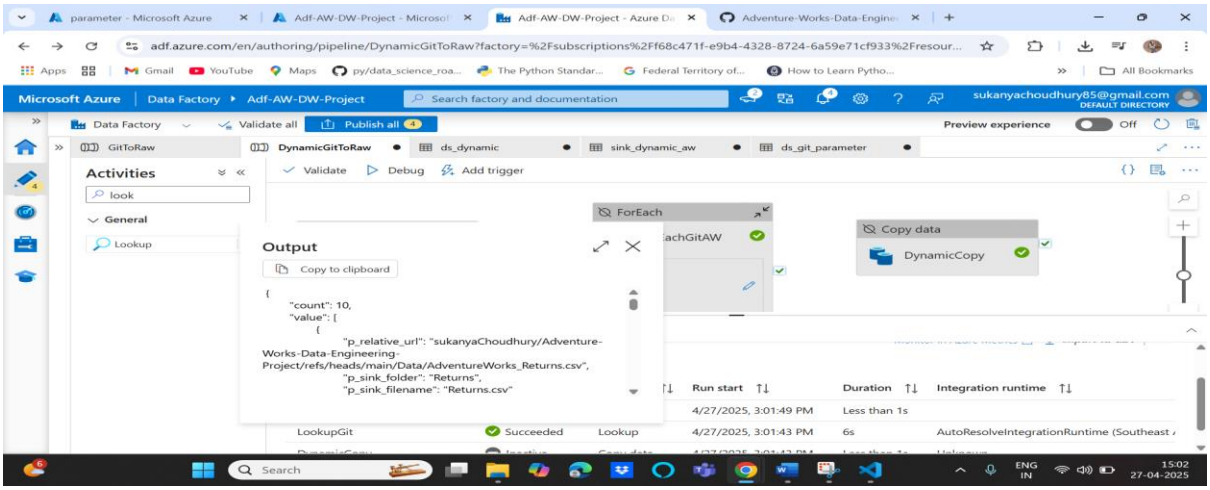
To do that first save the json file e.g. git.json inside a container in the storage account then create a lookup activity to get the git.json data by using a data lake→json data set type in the lookup activity and linked service is same as before as its using the same storage account and then select the folder and json file to connect

Lookup activity will fetch the json data which has parameters that we need to pass into the foreach activity

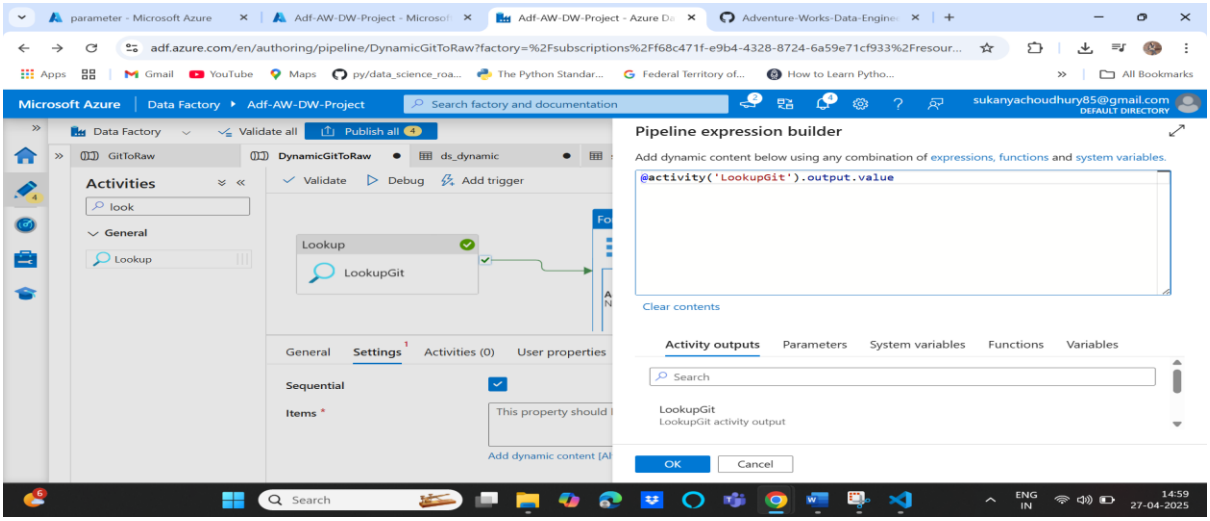


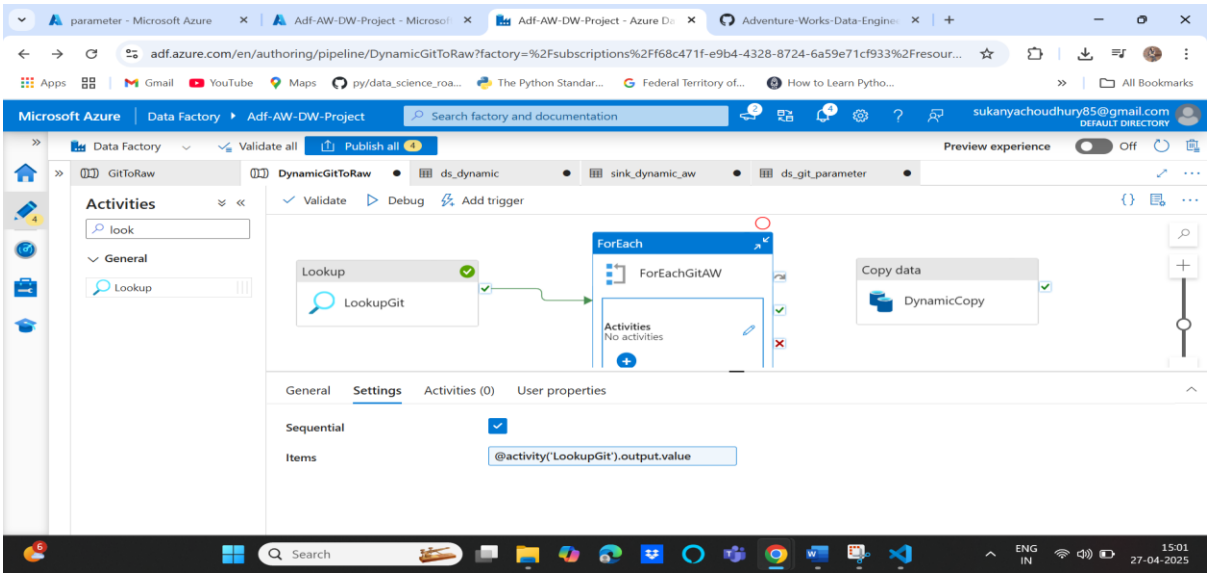


After this debug and test this particular activity only by disconnecting others,
You can see that it's passed as a dictionary and all the values are under the value key passed as a list so we need to pass this value key so that the list is passed as parameter:

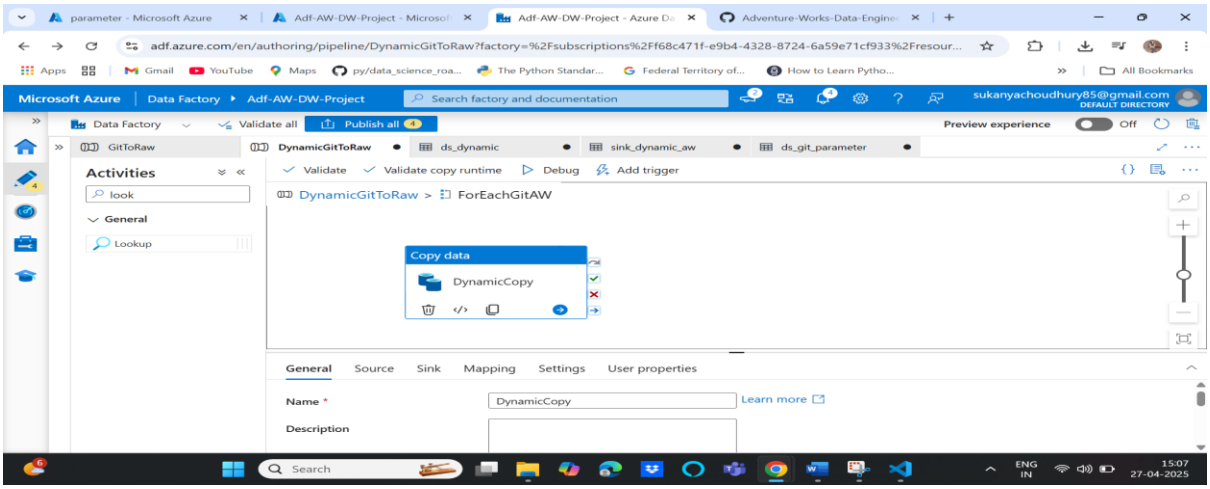


Next is to connect lookup git on success with the foreach activity and then in items, add dynamic content and select the lookupgit output and add .value to it since we need to get the value array:

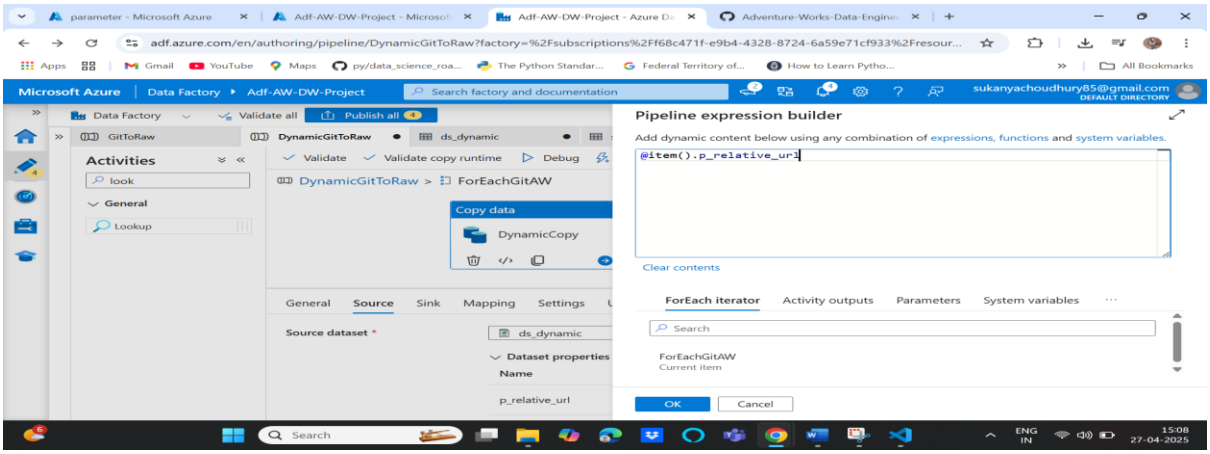


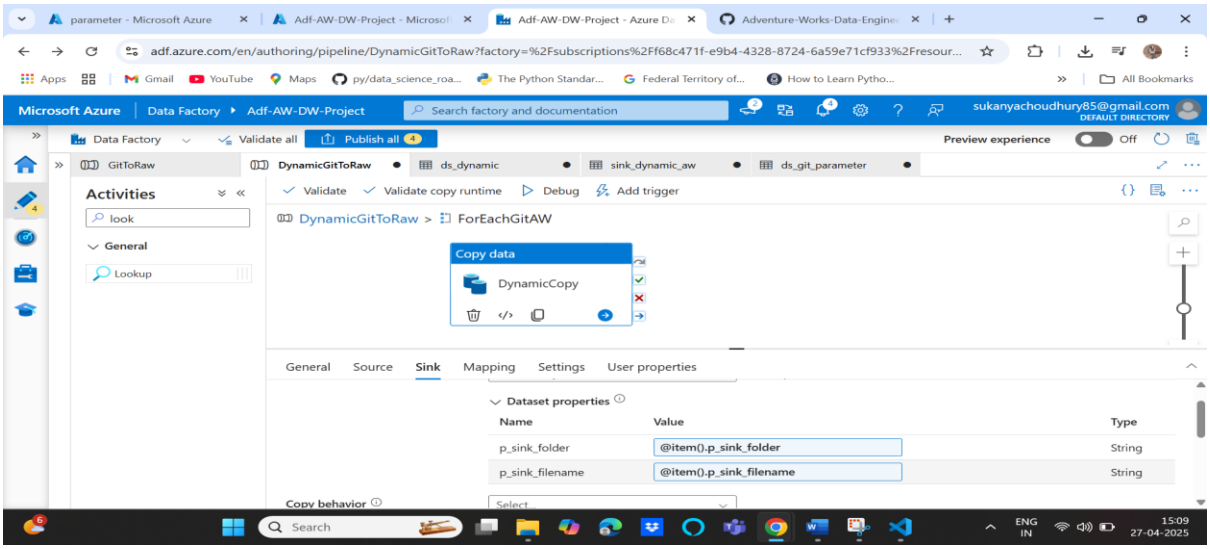


Copy the dynamic copy activity inside the activity tab of the foreach activity

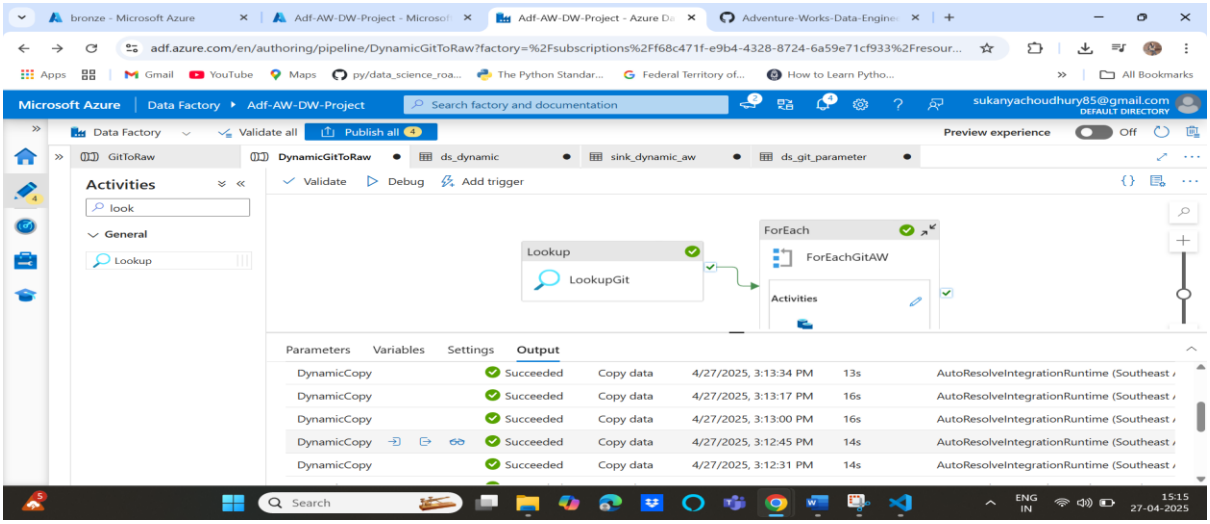


Then pass the values for each source and sink parameters:

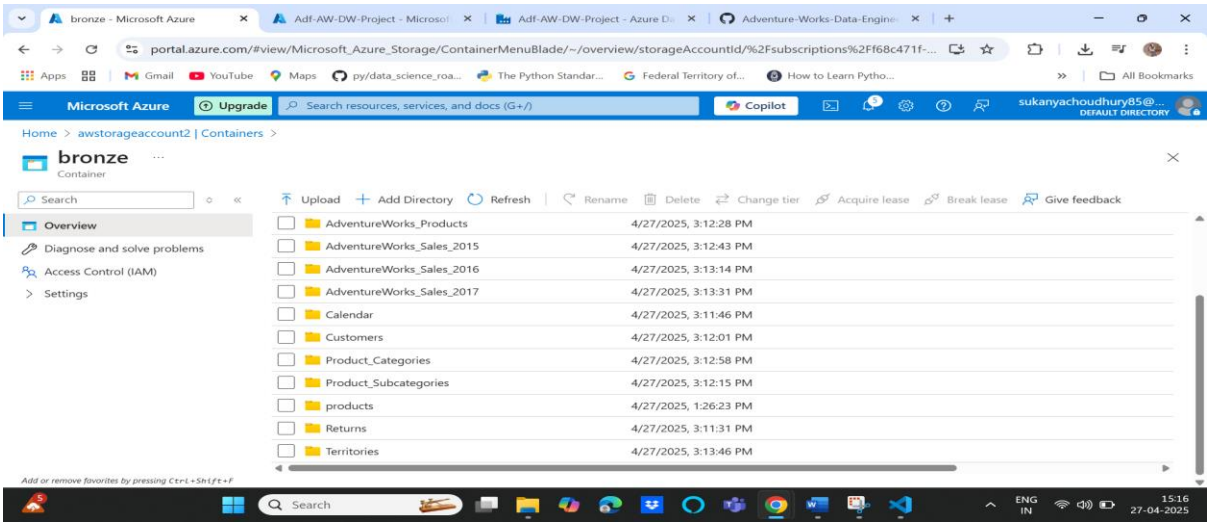




After debug the pipeline and it should be able to fetch all 10 items

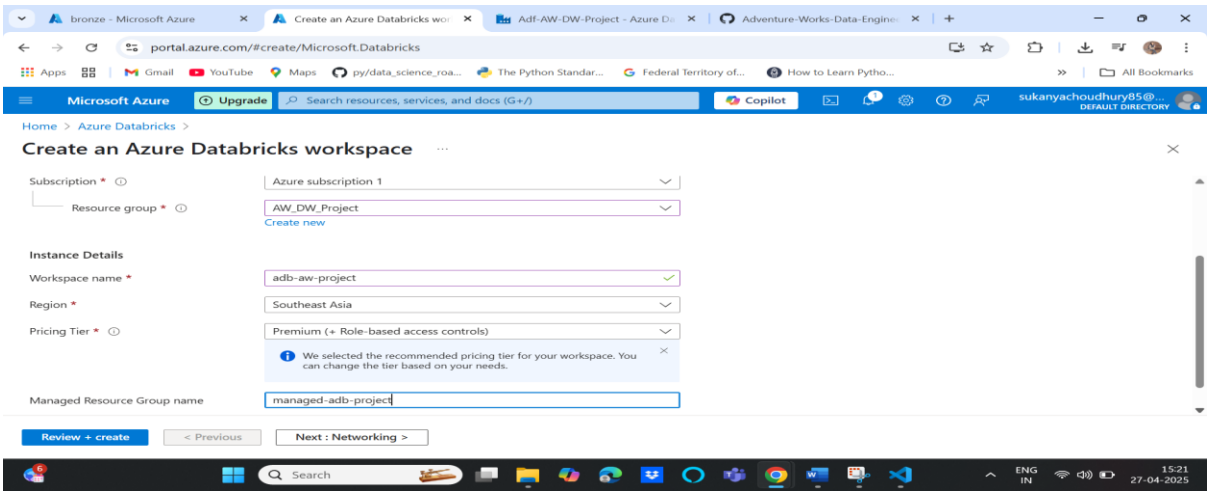


Validate the bronze container to see all files:

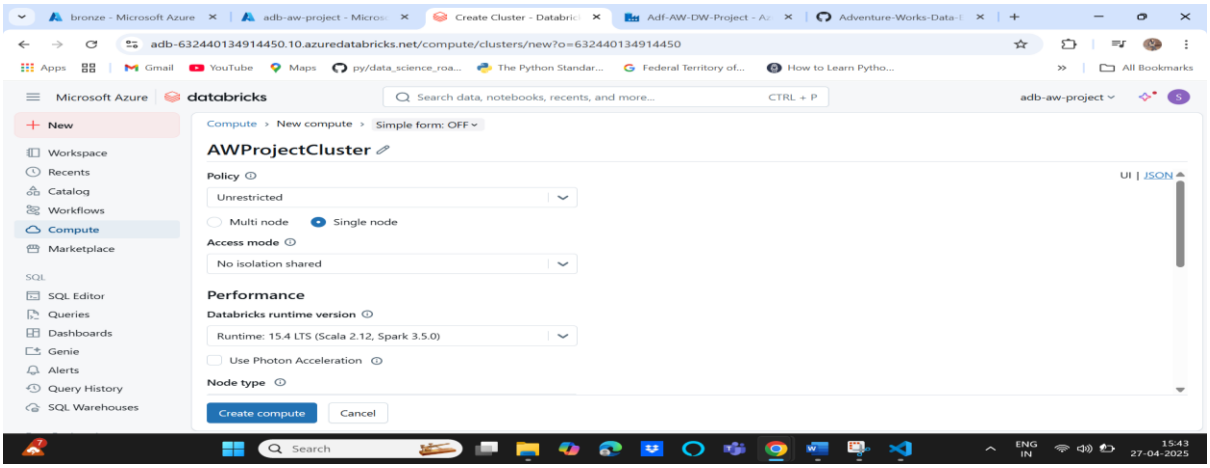


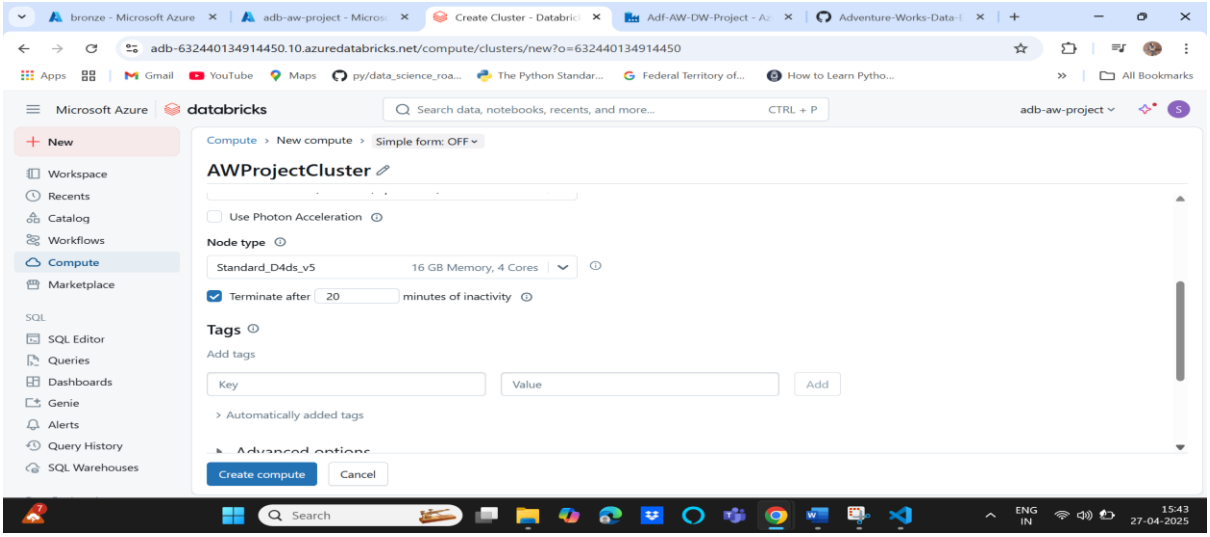
Phase 2: Data manipulation using Databricks and load into silver layer

1) Create Databricks, “adb-aw-project”:



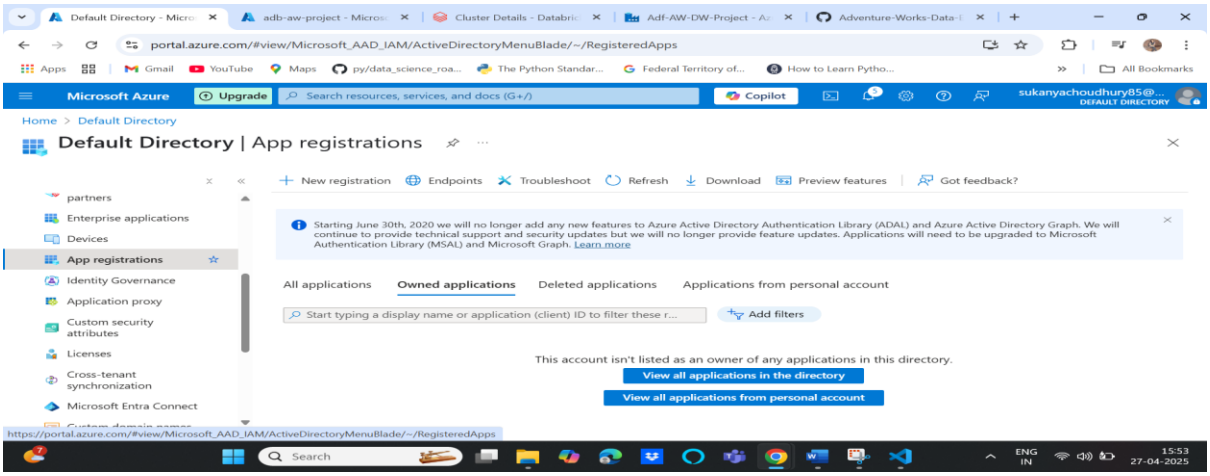
2) Create a new compute:

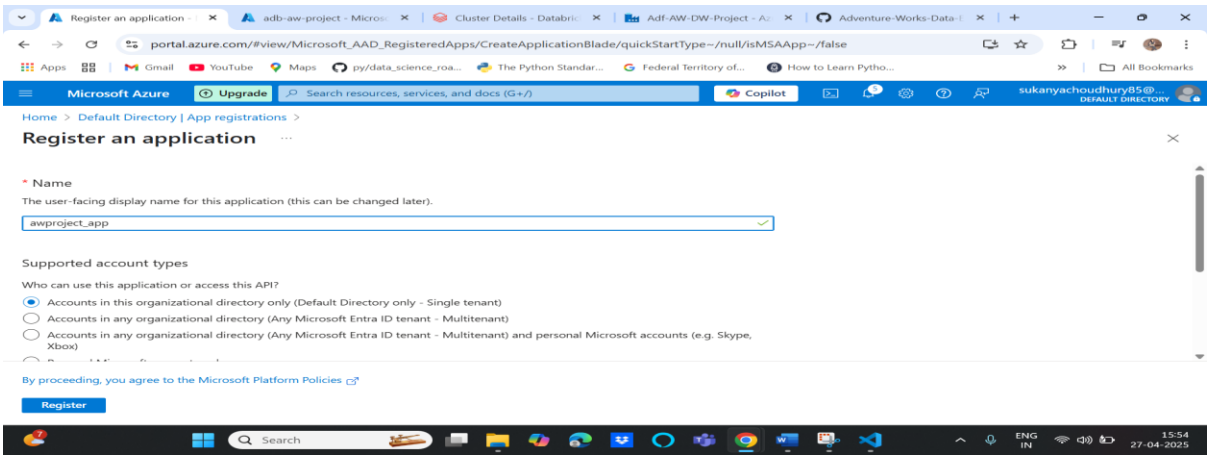




3) Create app registration for databricks to access data from our bronze layer in Datalake

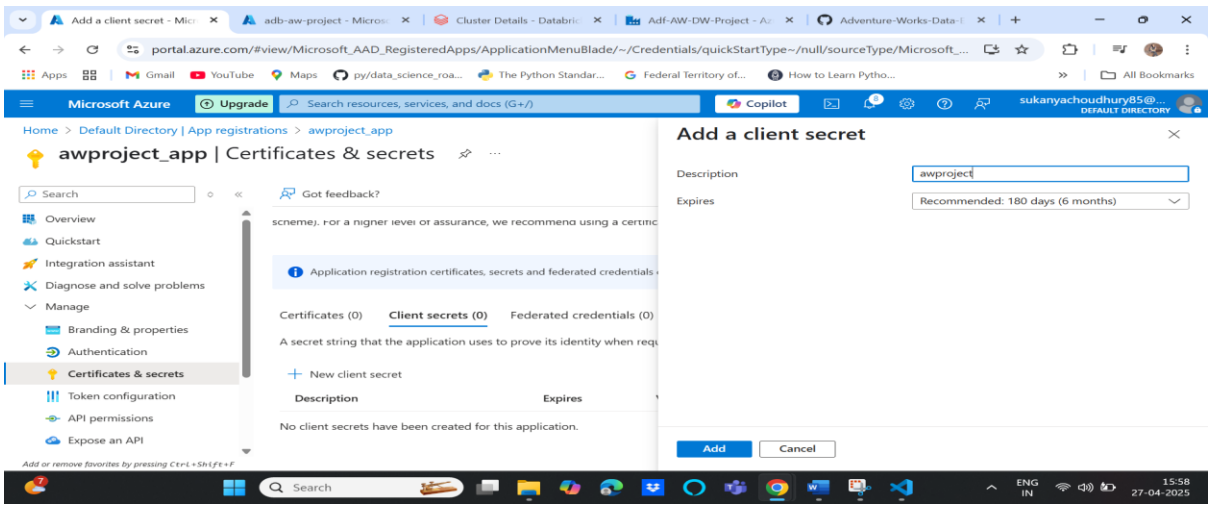
Microsoft Entra Id→ App Registrations





Save the details for this app key: App ID and Directory ID which would be needed later on

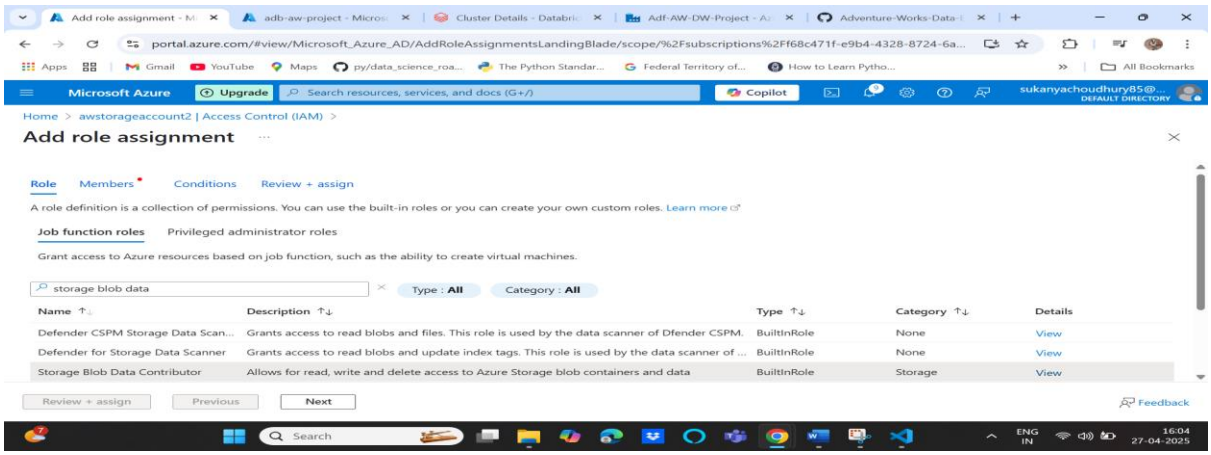
4) Create secret key and save it's value to be used as secret key in Jupiter notebook

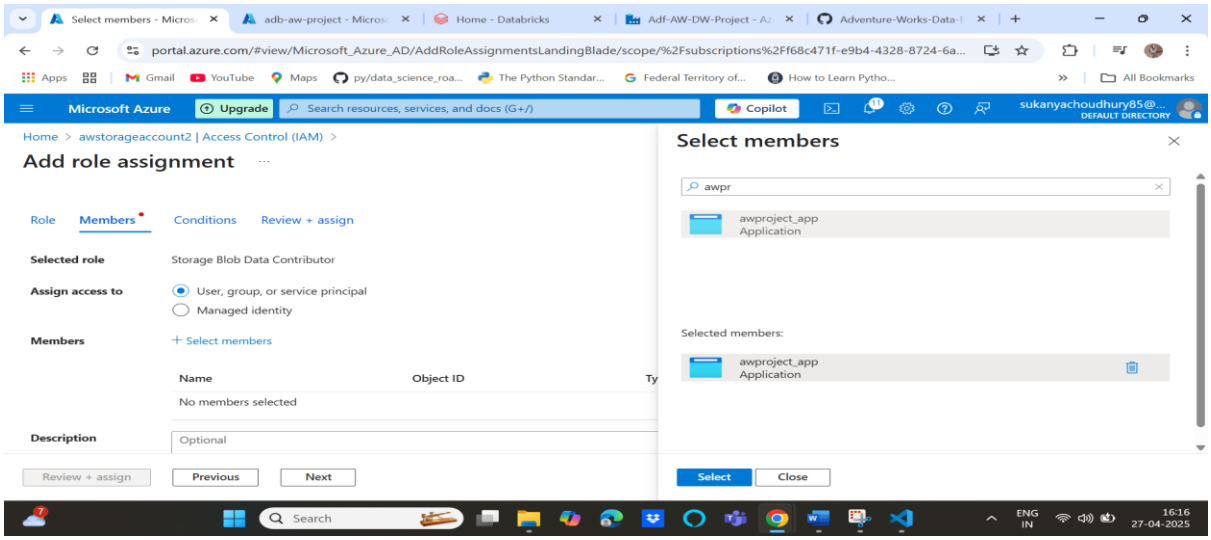


5) Assigning contributor role to a storage account application

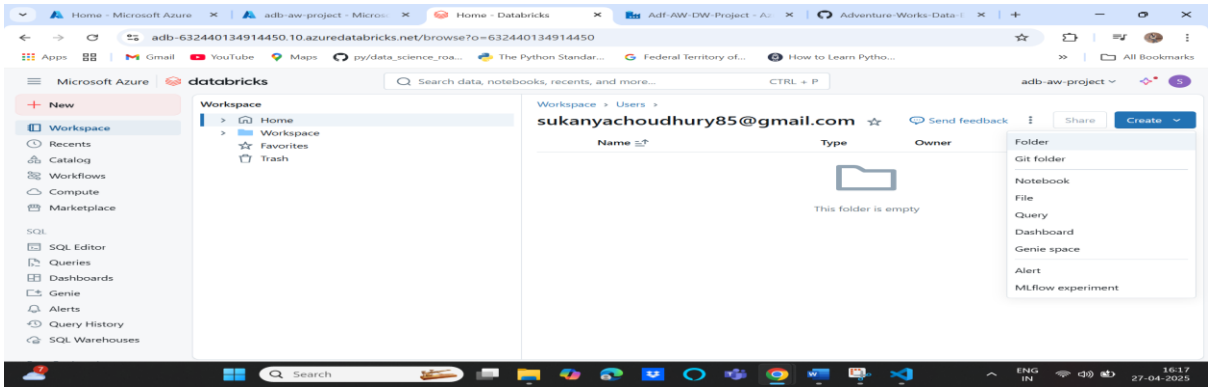
Select storage account→ access control(IAM)→ add a role→ add role assignment

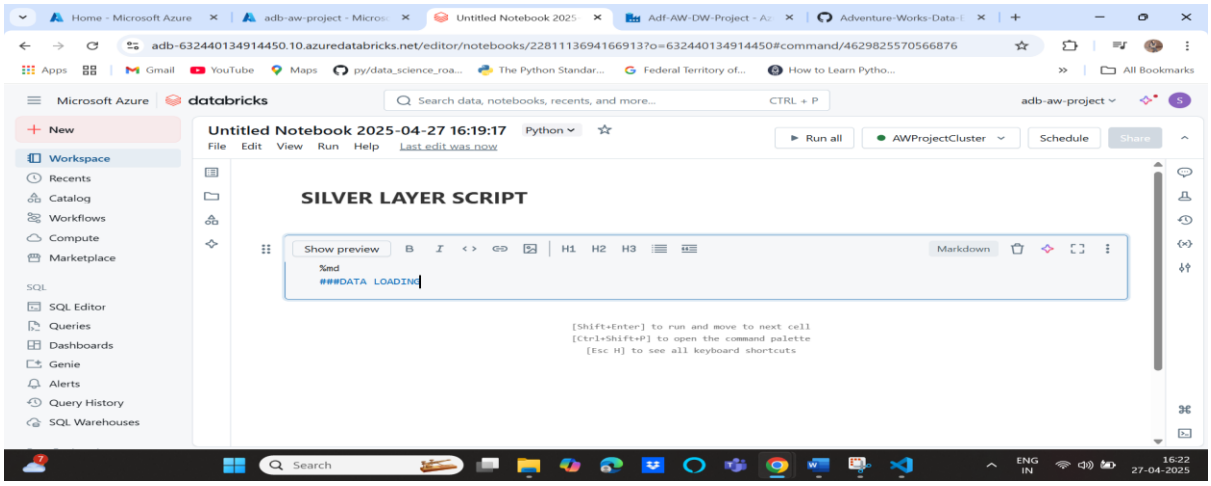
➔ Storage blob data contributor→ select members→application name(awproject_app)→Review+assign





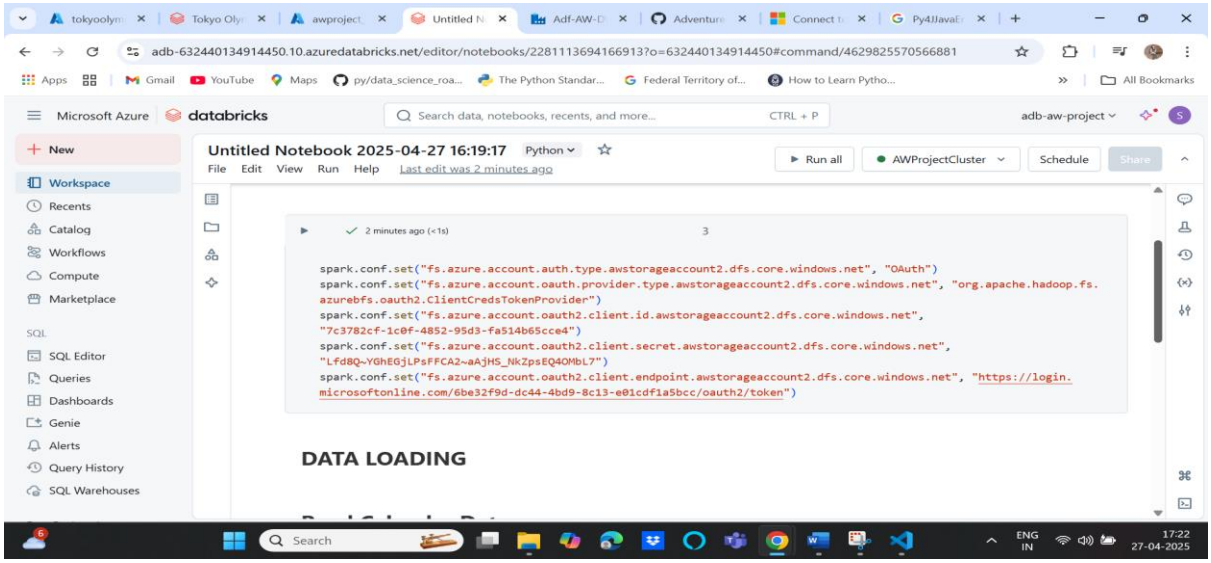
6) Write transformation code: Create a workspace→ folder→new notebook→Add headings using markup from dropdown



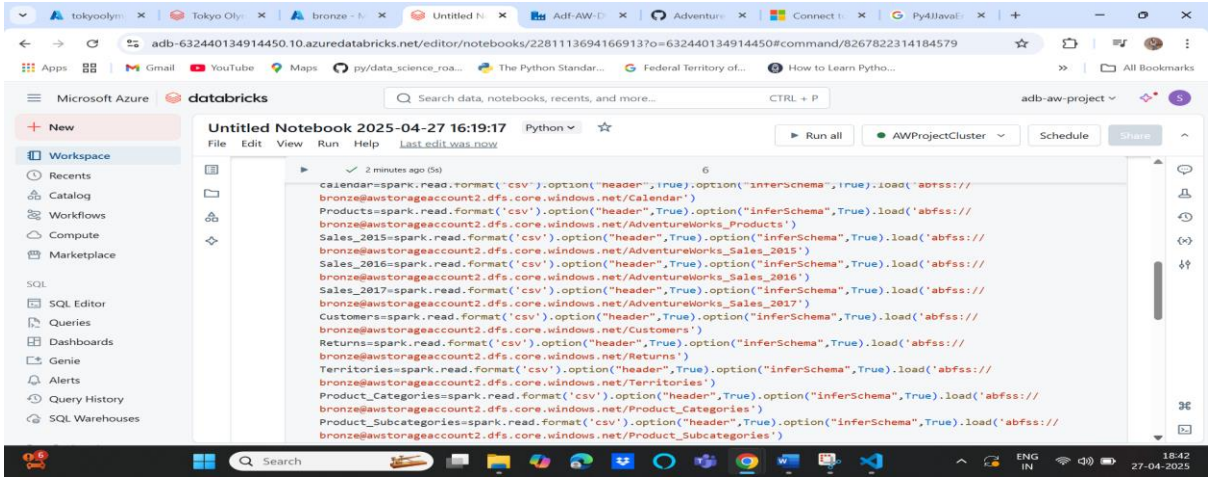


Data access: code taken from **Azure service principal python code from** https://learn.microsoft.com/en-us/azure/databricks/connect/storage/azure-storage

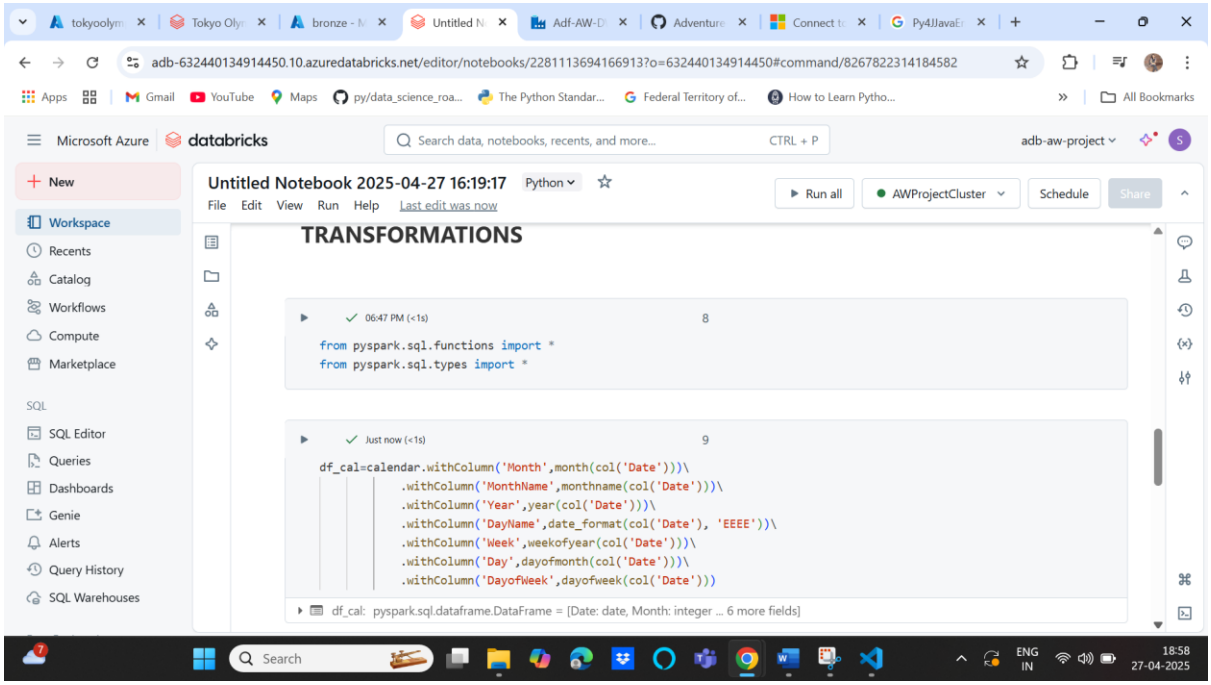
Replace client id, directory id, secret key, storage account appropriately

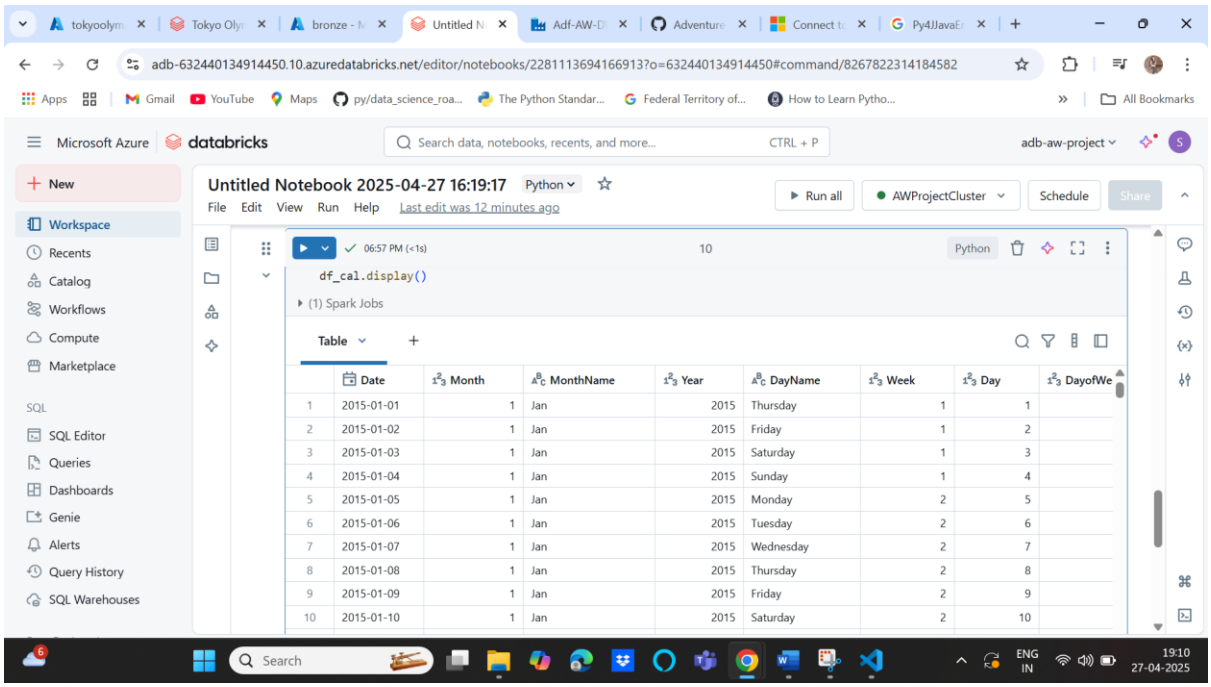


7) Reading data from bronze layer

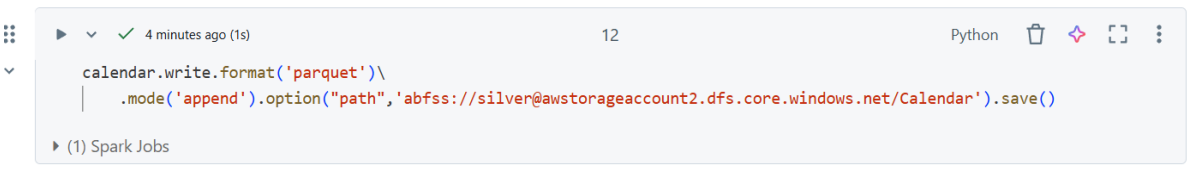


8) Transformations

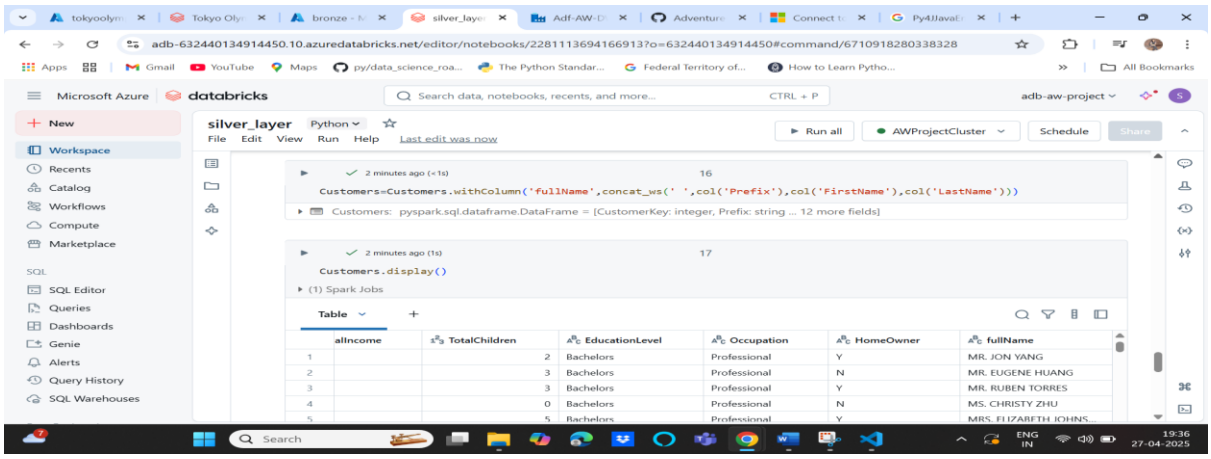




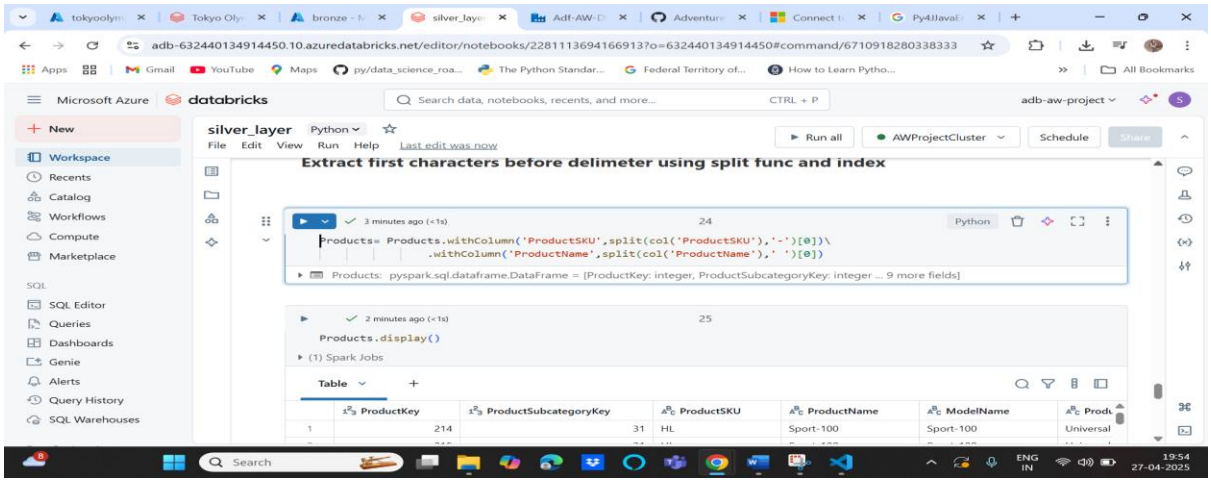
9) Write into silver layer



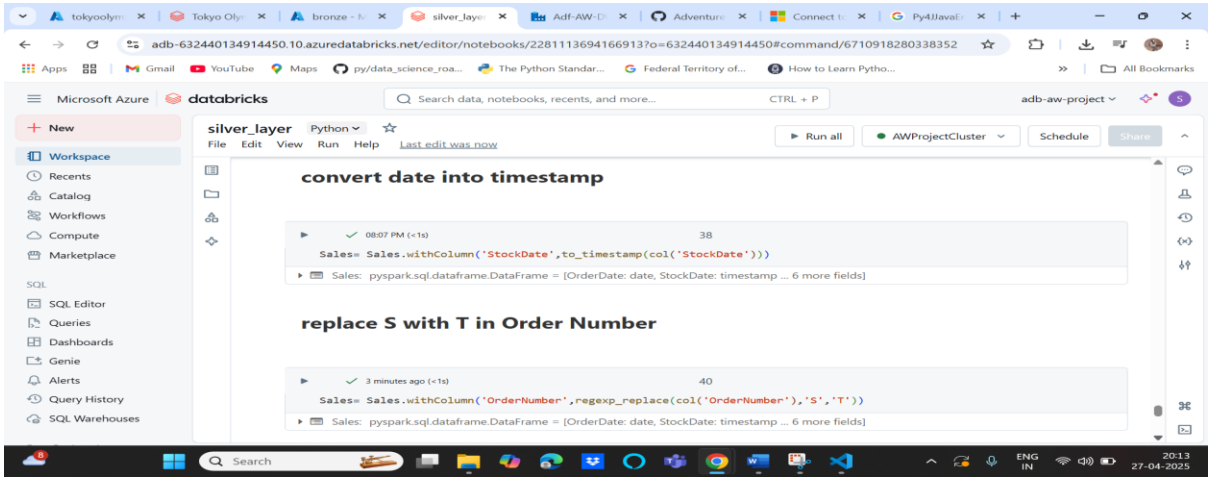
10) Customers transformation

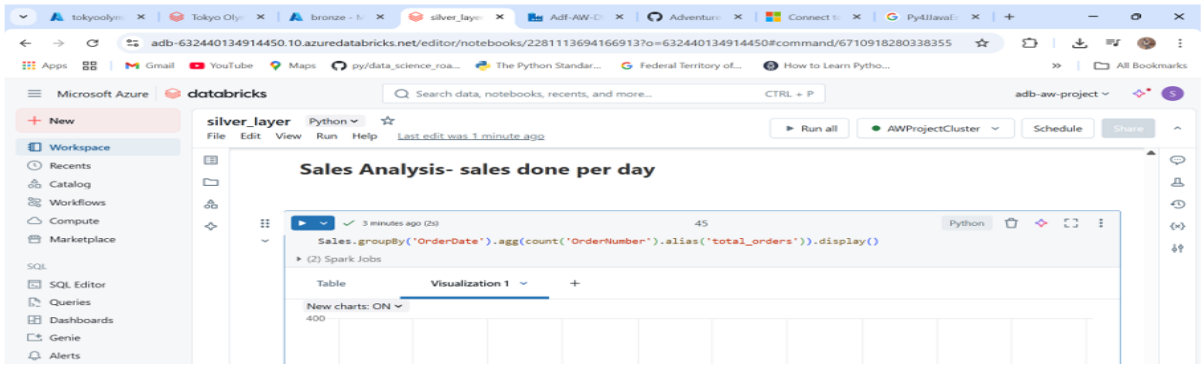


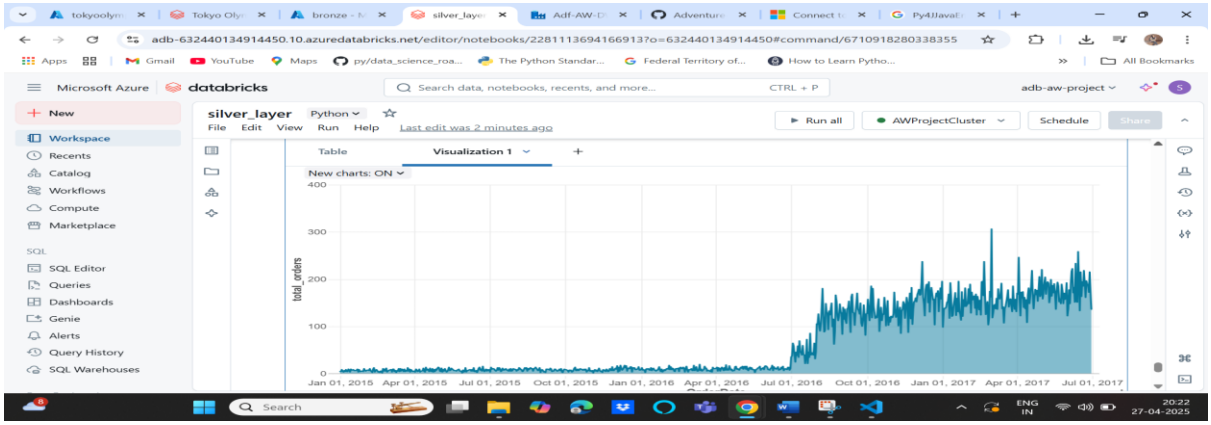
11) Products Transformation



12) Sales Transformation





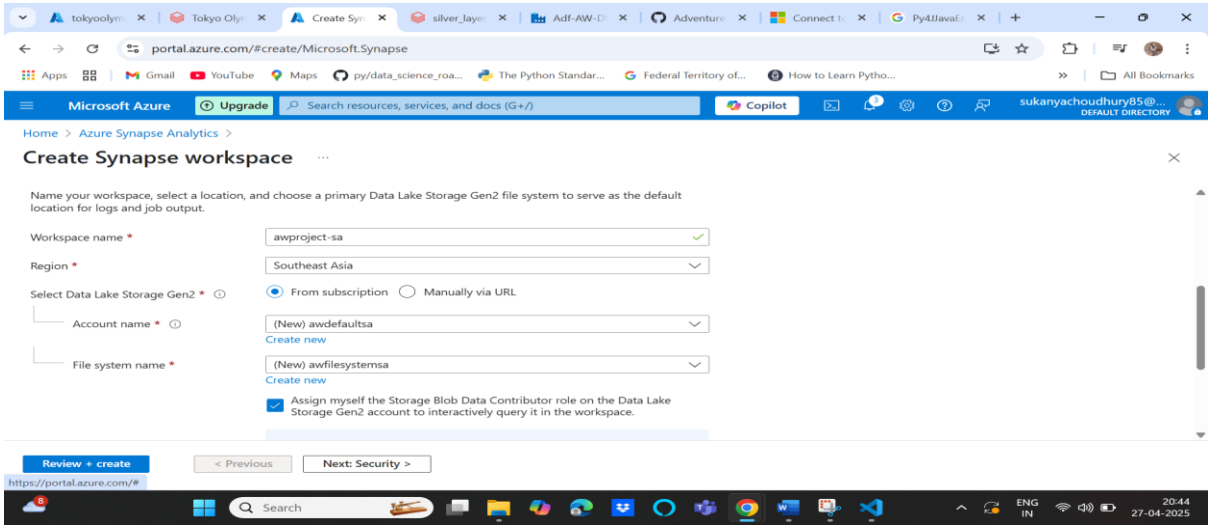


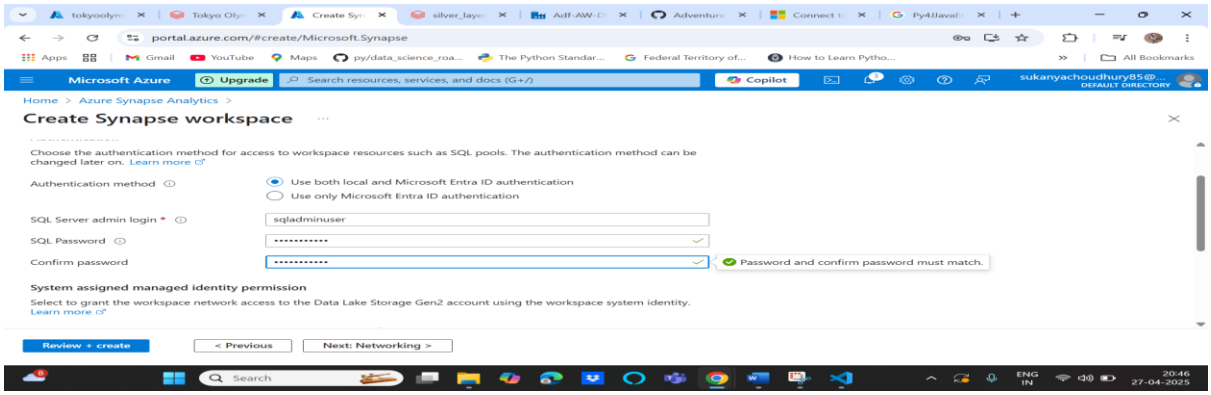
All data after transformation are written into silver container of the data lake

Phase 3: Data Storage into Synapse Analytics

1) Create Synapsis Analytics Workspace(awproject-sa) and save the DB details, username and pwd

Usually a new storage account is created while creating synapse analytics workspace as shown in data lake storage gen2

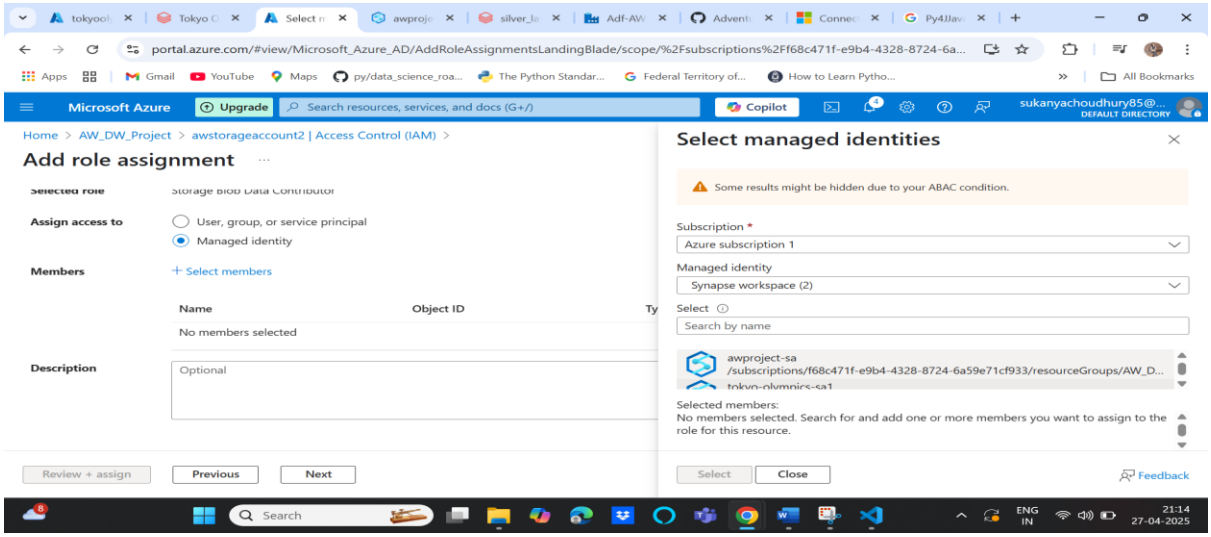




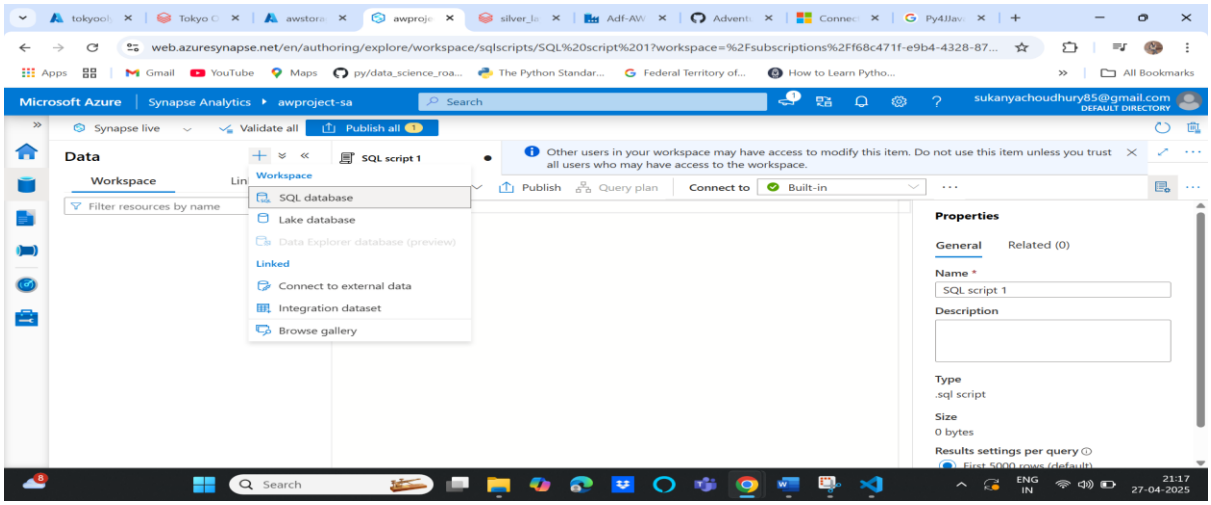
2) System managed identity: allow synapse to connect with data lake to get data

Go to the original/external storage account → IAM→Add role→Storage blob data contributor

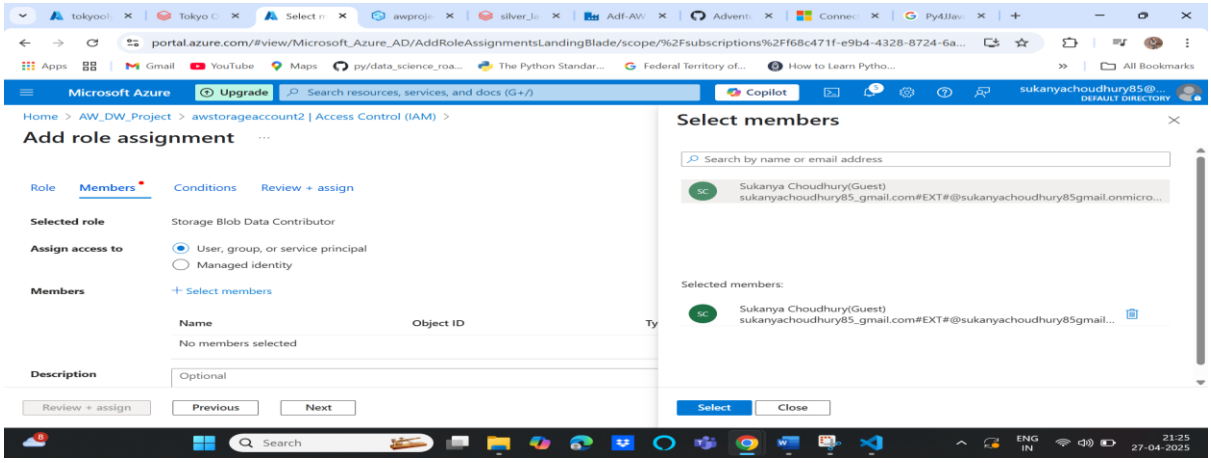
→Next-> check Managed Identity→ Addmembers→Synapse Workspace



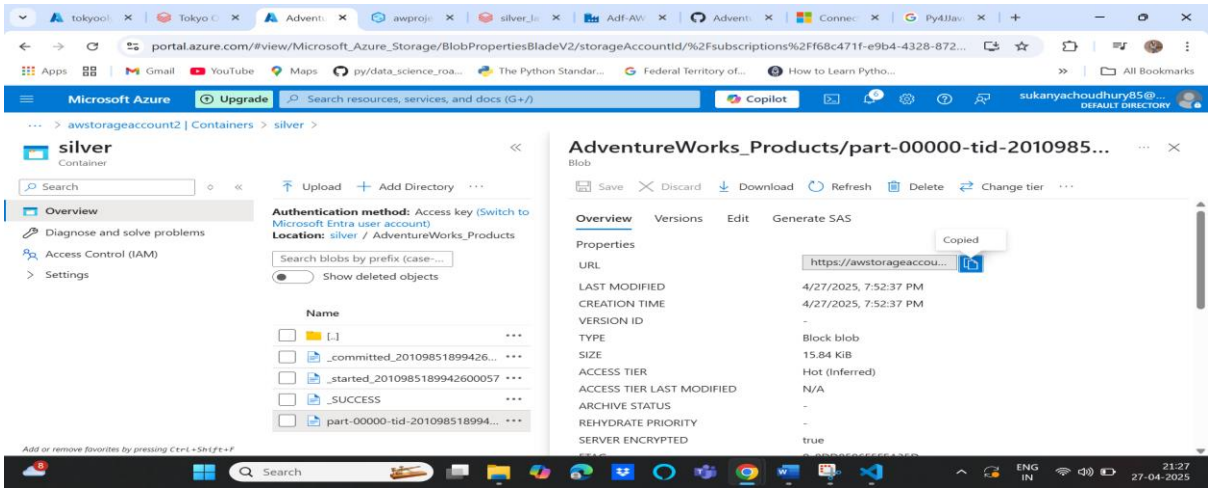
3) Create a synapse database first as we need to run SQL queries on it for data modelling transformations etc. SA workspace→ Data tab→+ sign-> SQL database



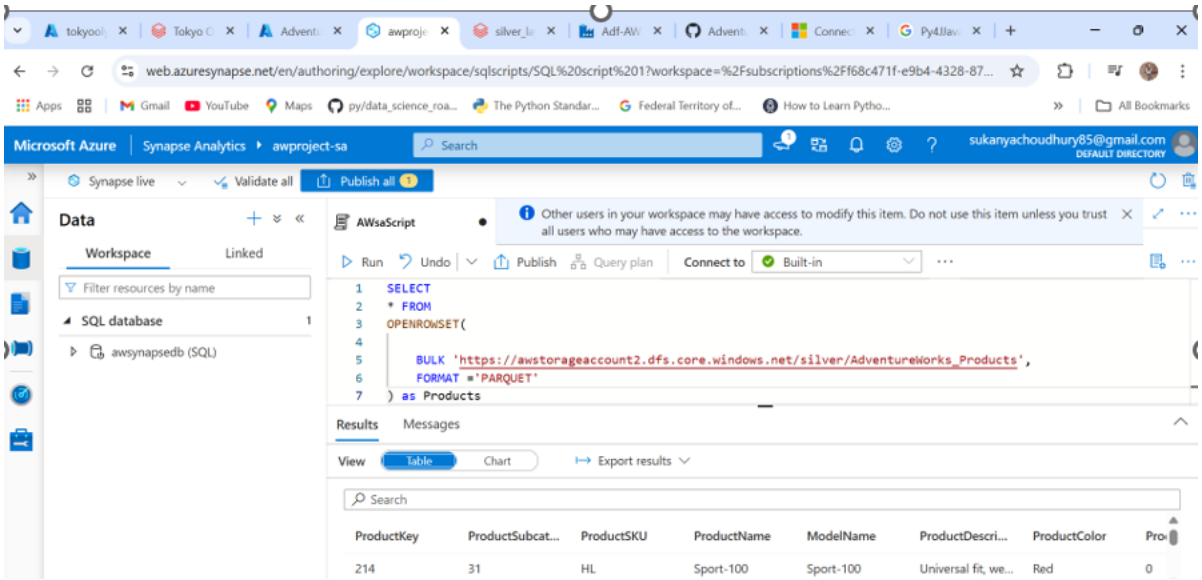
4) After creating db, go to IAM again on storage account and then give yourself access to access the data



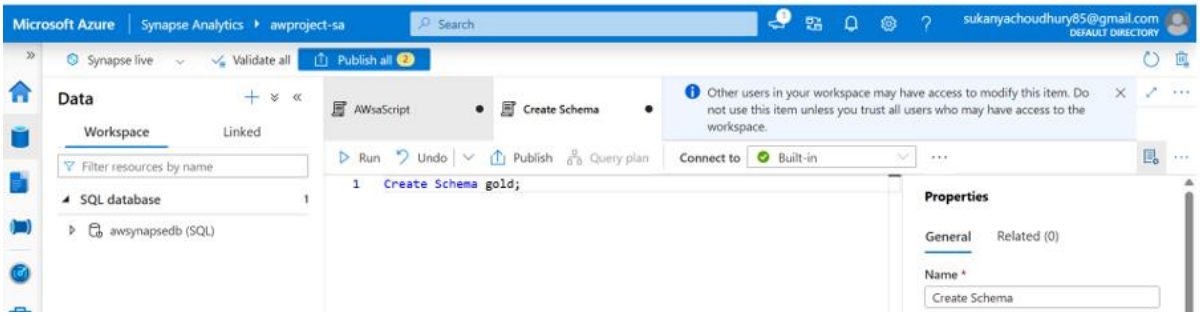
5) Copy the silver data folder url and then use it in SQL script:



And then replace blob in the filename with dfs as we are using data lake and not blob storage to query the parquet data and get tabular data

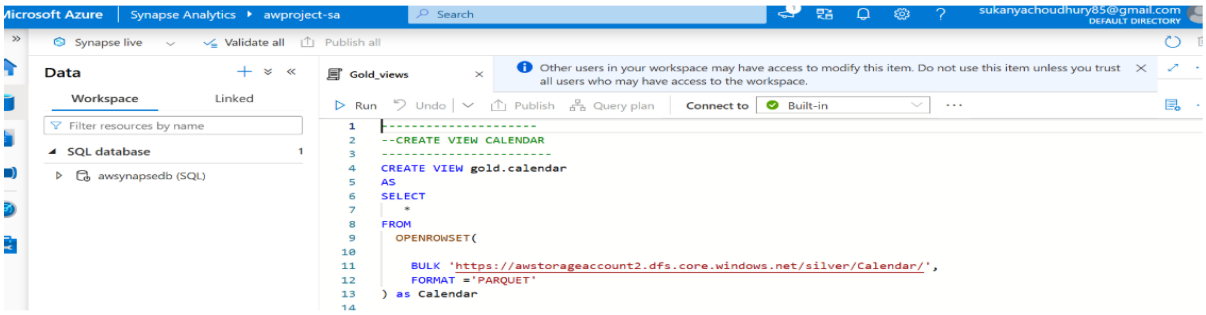


6) Create Gold Schema so that we can create views

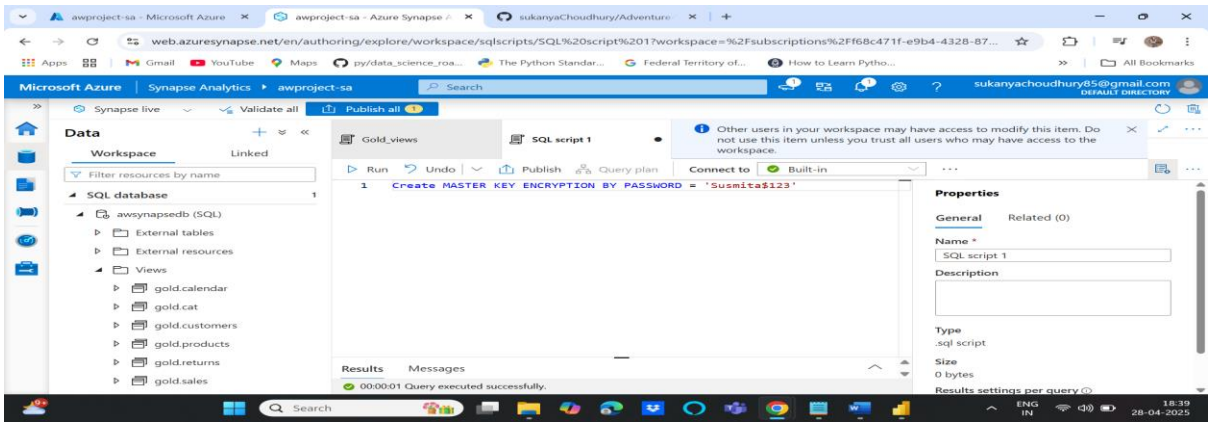


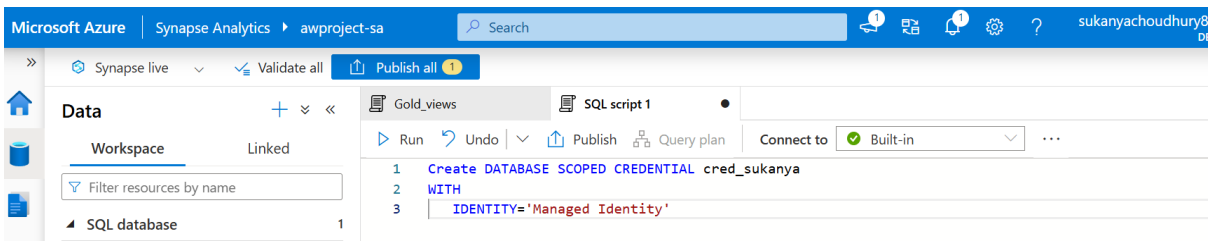
7) Create views from parquet data using below script

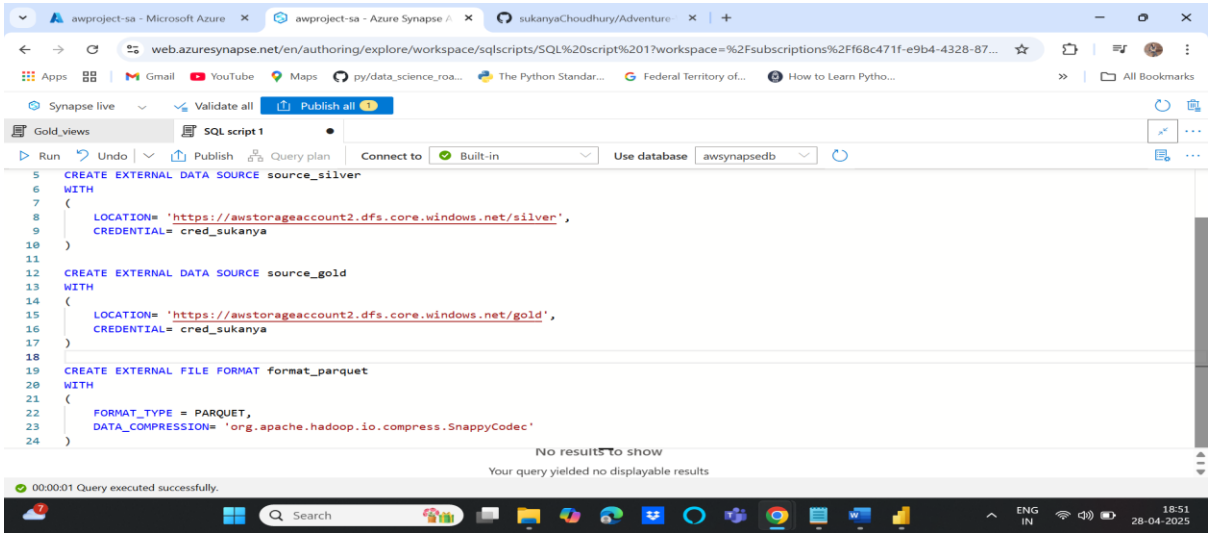
gold_views.sql



8) Now next step is to create external tables from the views by creating external sources

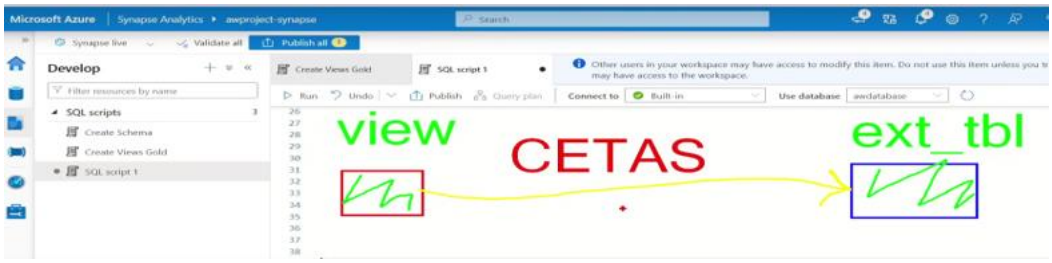


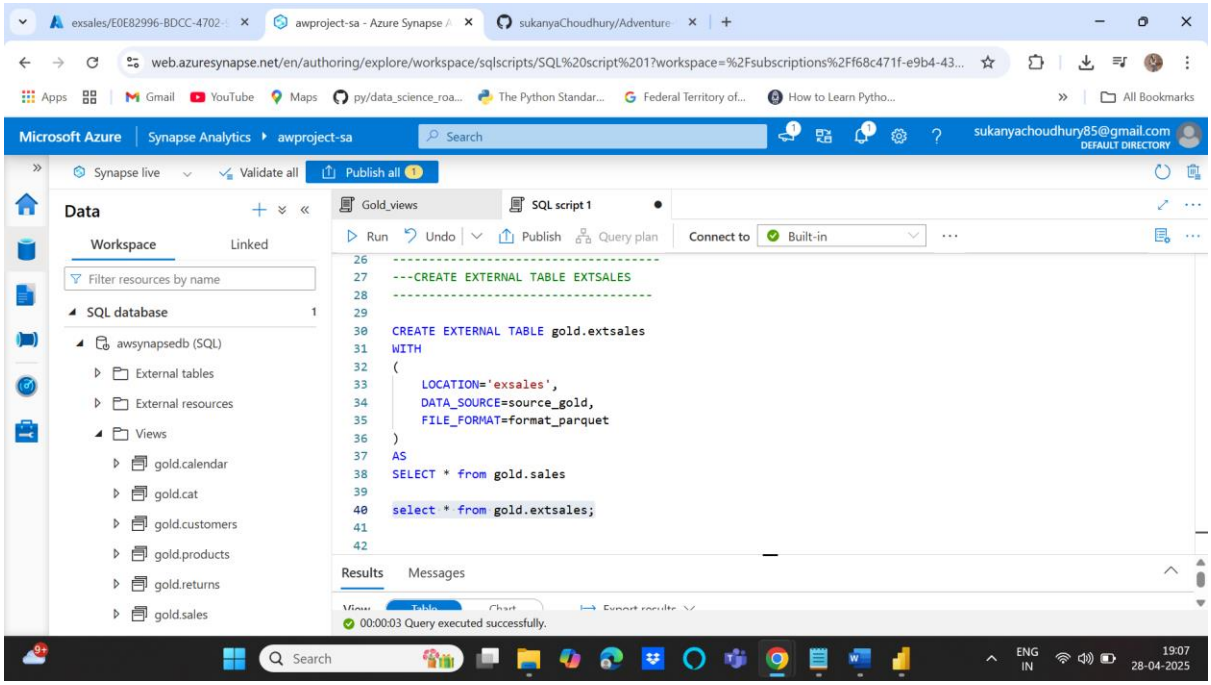




9) CREATE EXTERNAL TABLE AS SELECT (CTAS)

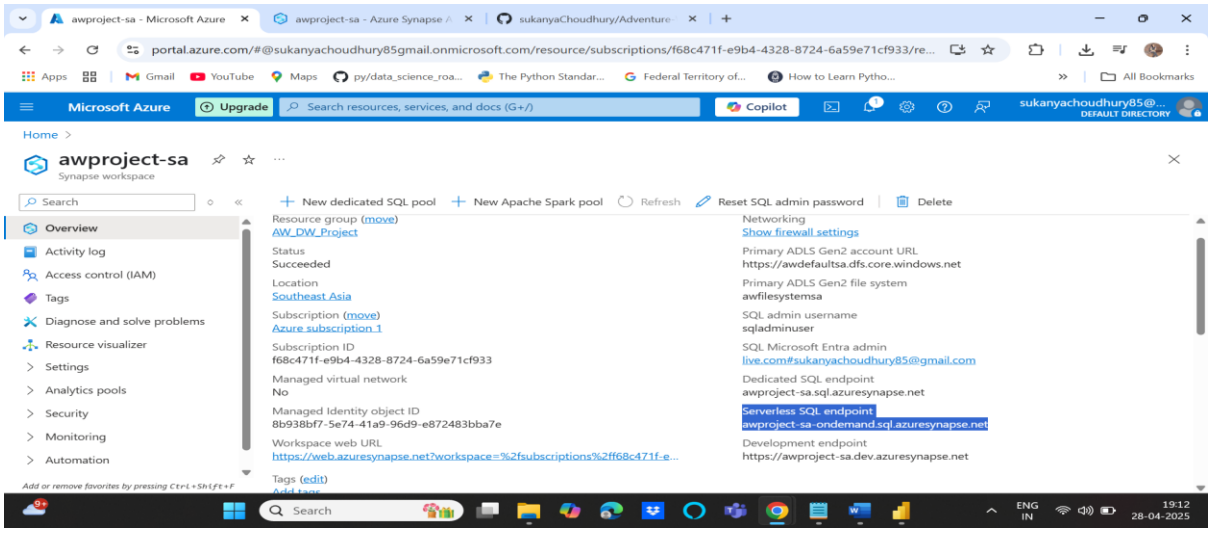
Transfer the views created on silver layer into gold layer as external table using ctas



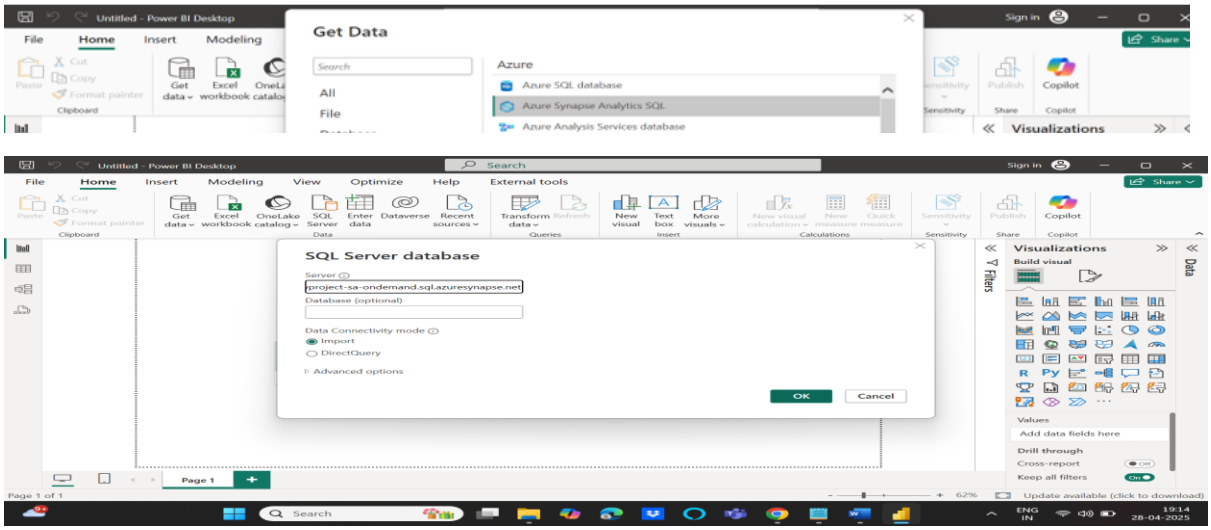


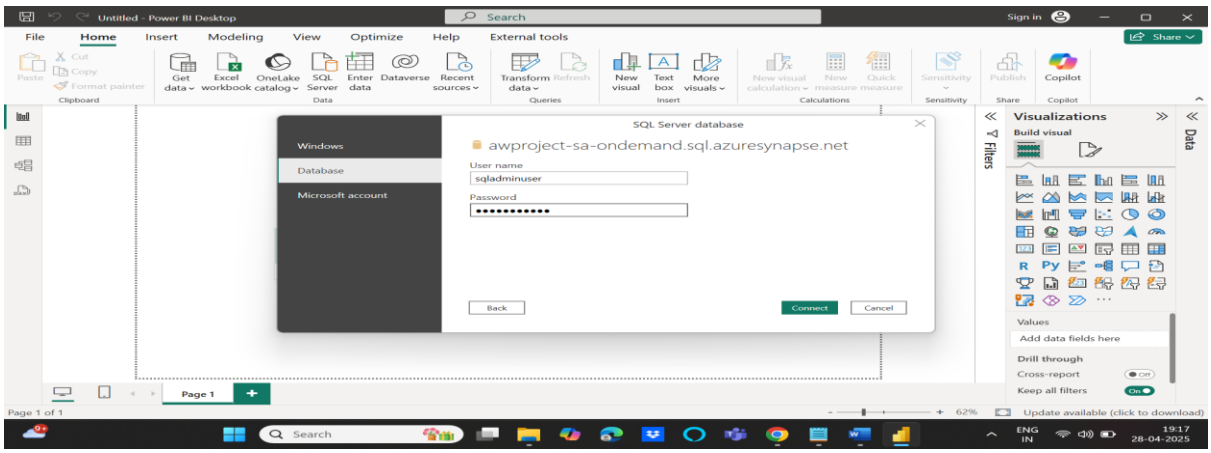
Phase 4: Data Reporting Using Power BI

Connect to power bi using Serverless SQL endpoint in our case, go to synapse workspace ,you can see the SQL endpoint details on the R.H.S:



Then go to power bi → get data→Azure Synapse Analytics SQL





An interactive Dashboard is created to showcase the sales figures: