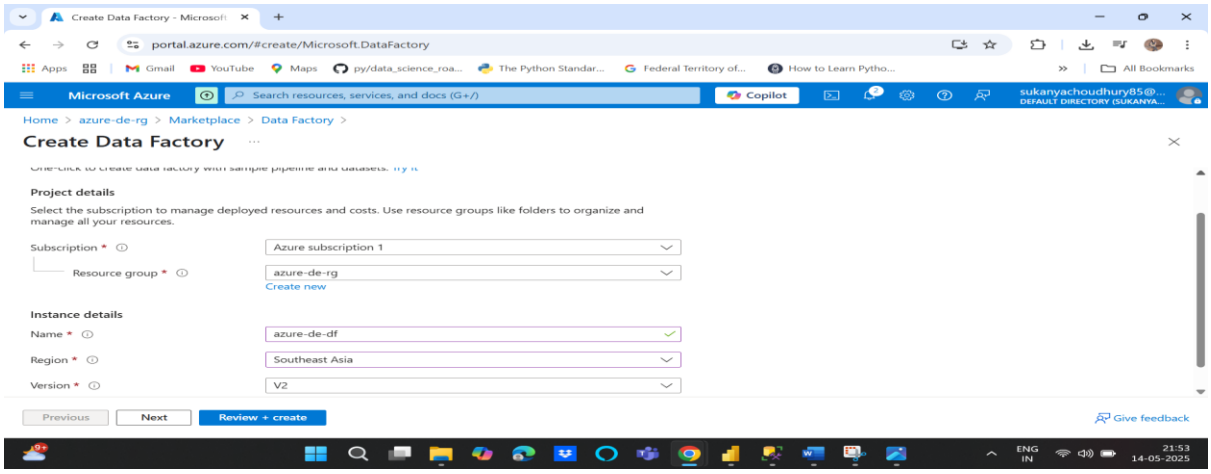
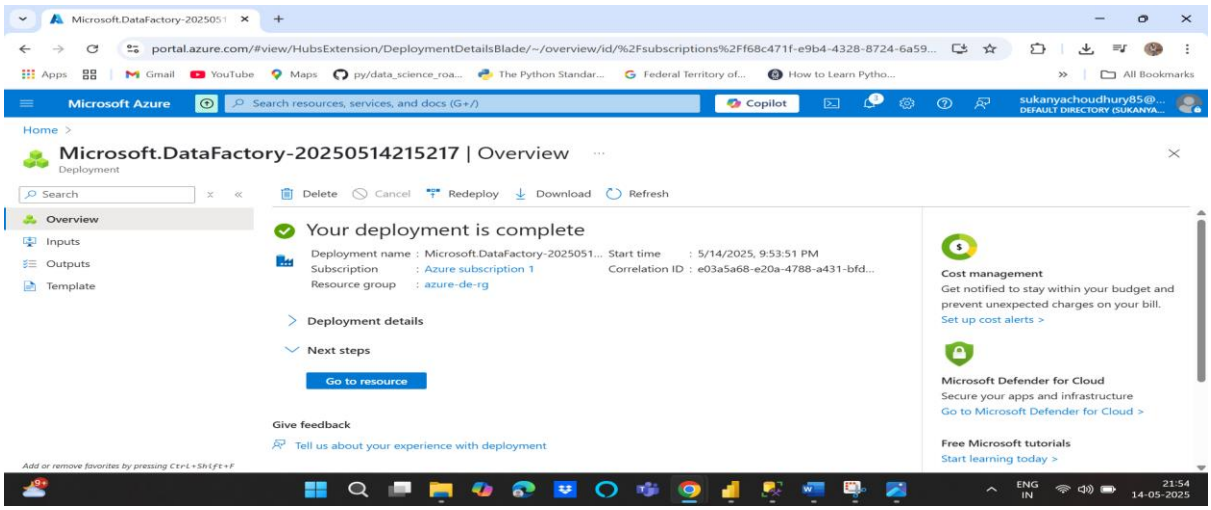
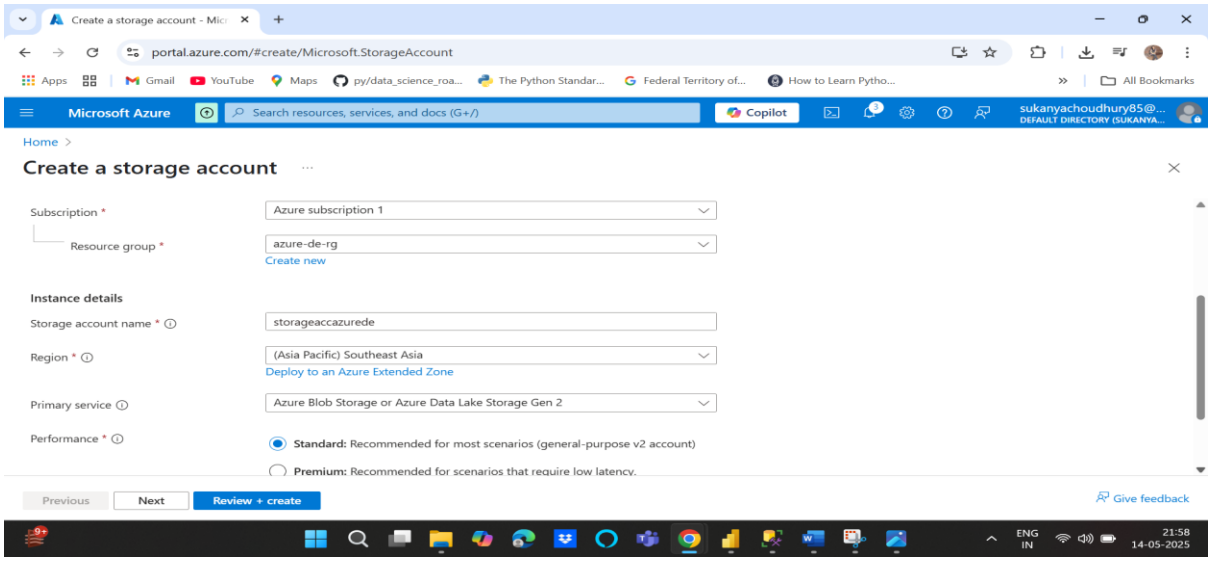
Step 1: Azure Environment Setup: Create below Azure services :

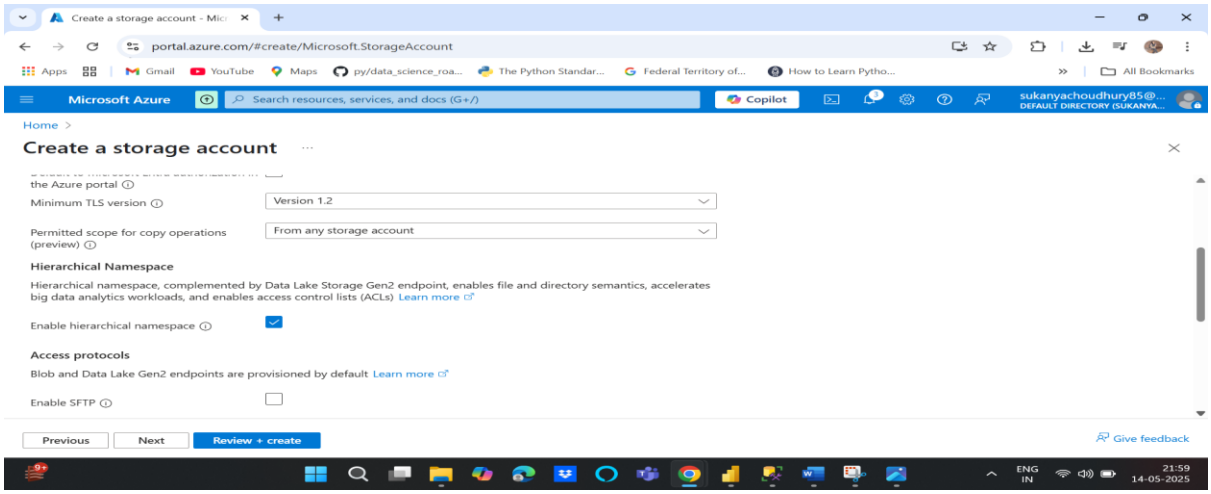
1) Create a resource group “azure-de-rg”
2) Create a Data Factory “azure-de-df” as below:

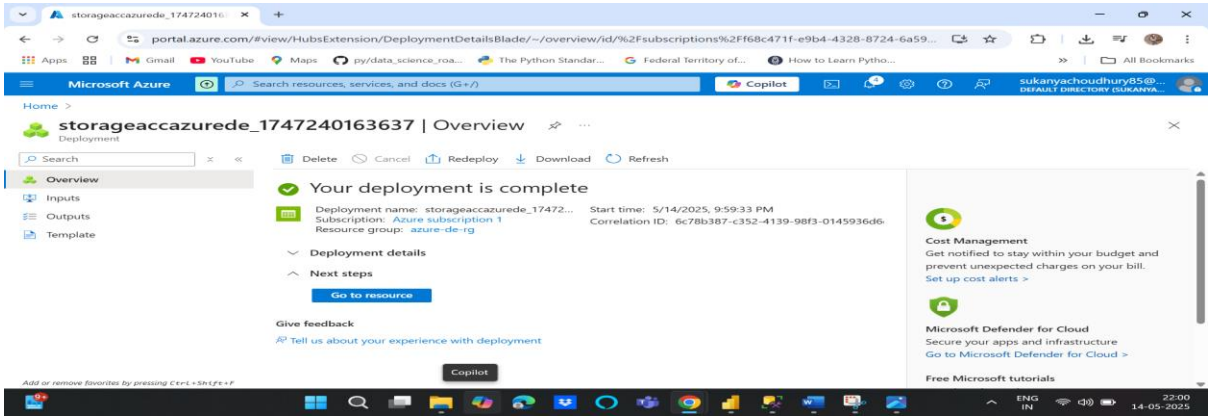




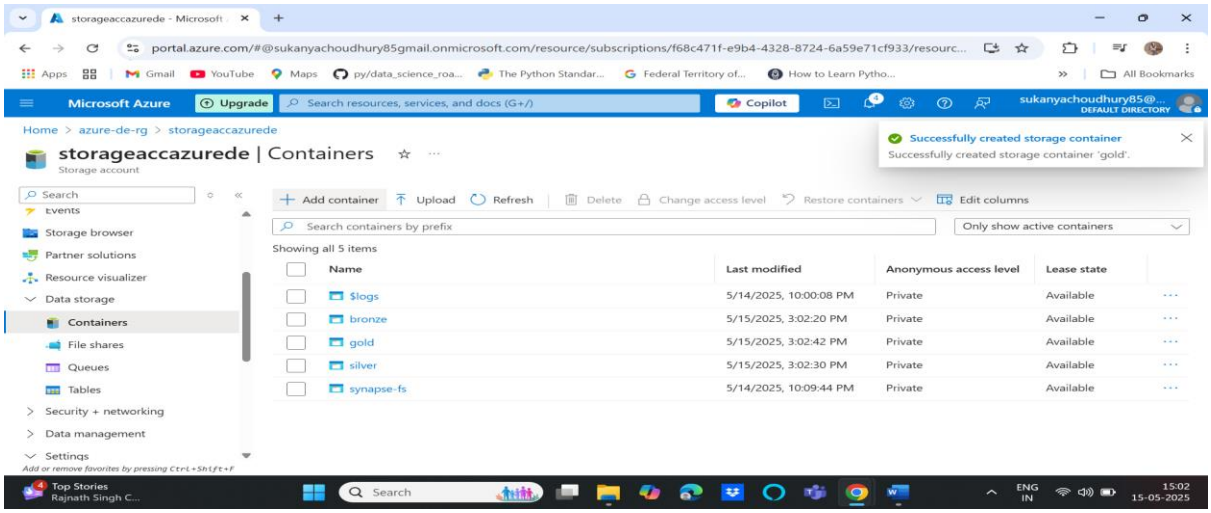
3) Create a hierarchical Data lake Gen2 Storage Account , “storageaccazurede” as below:



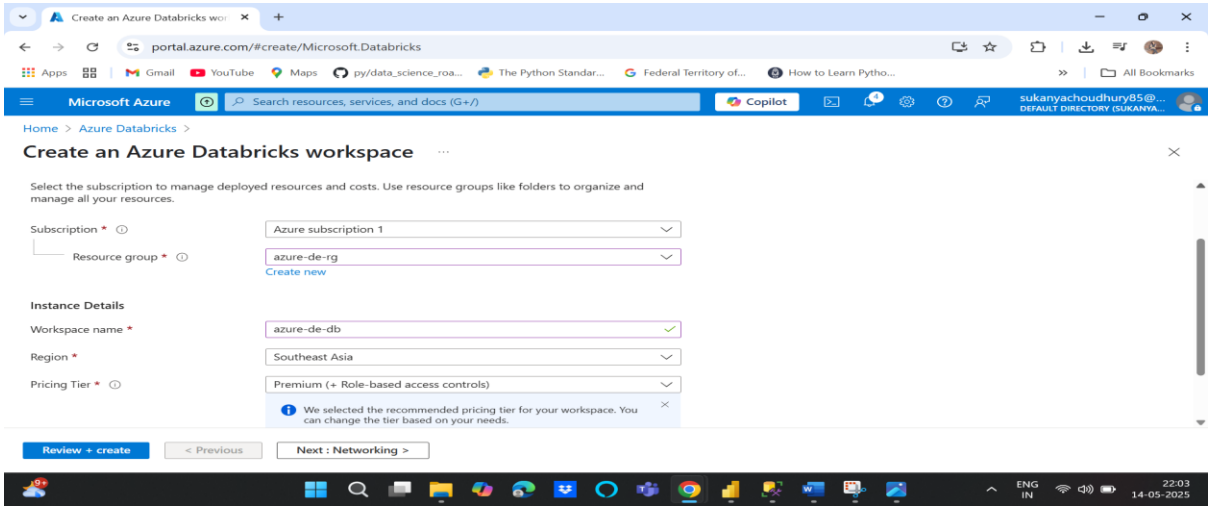


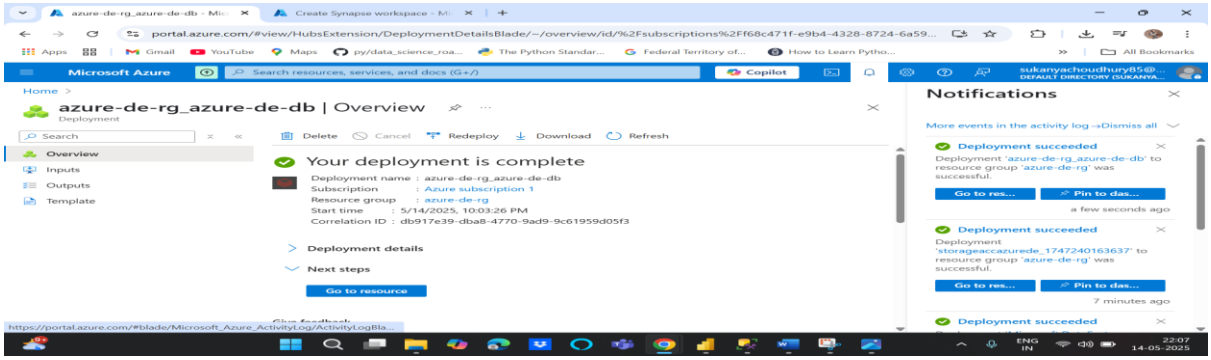


4) Create containers bronze/solver/gold inside the storage account→Containers

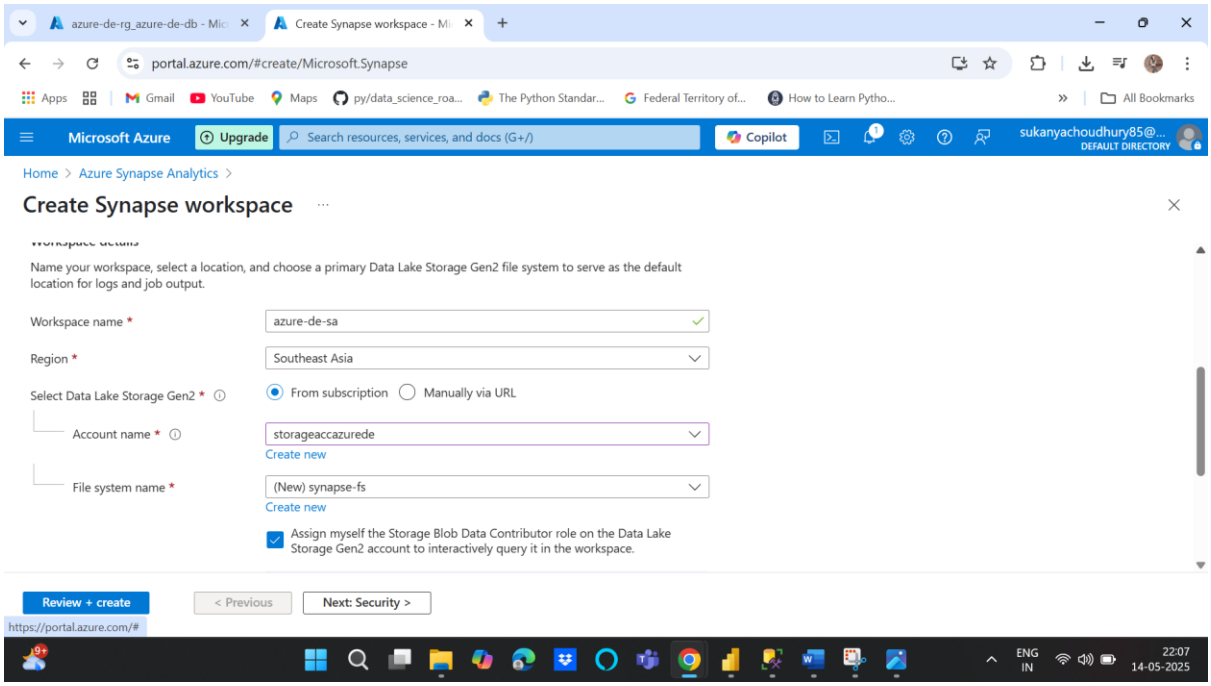


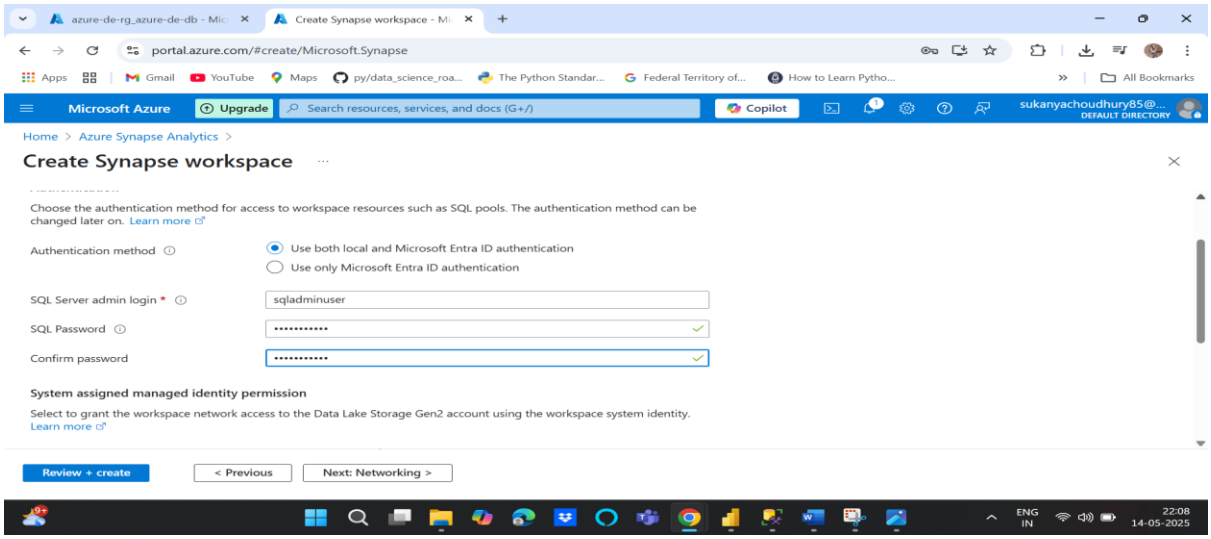
5) Create Databricks Workspace, “azure-de-db”

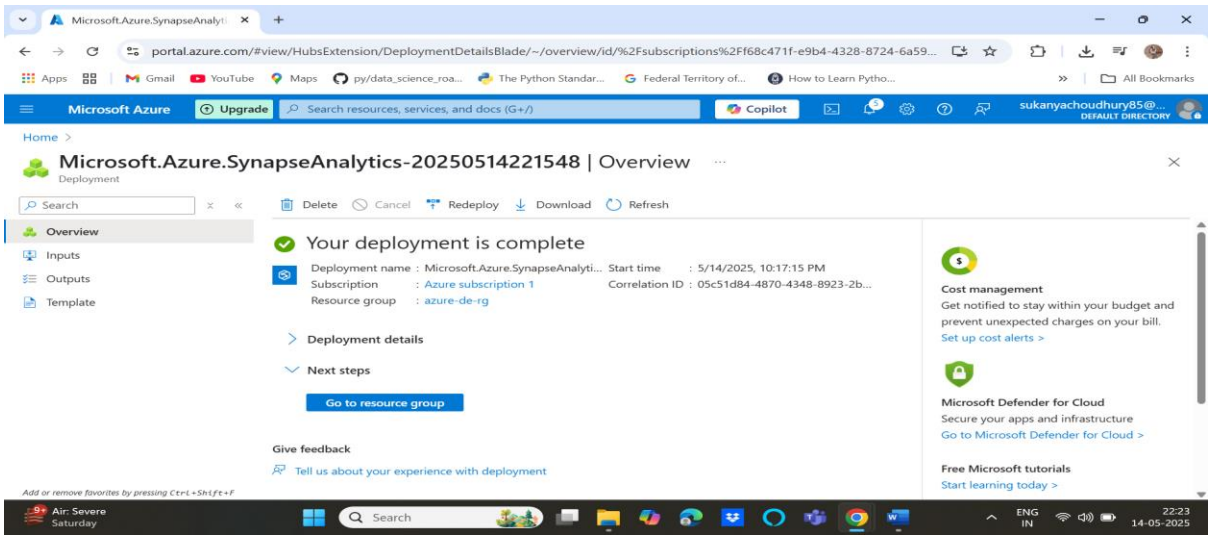




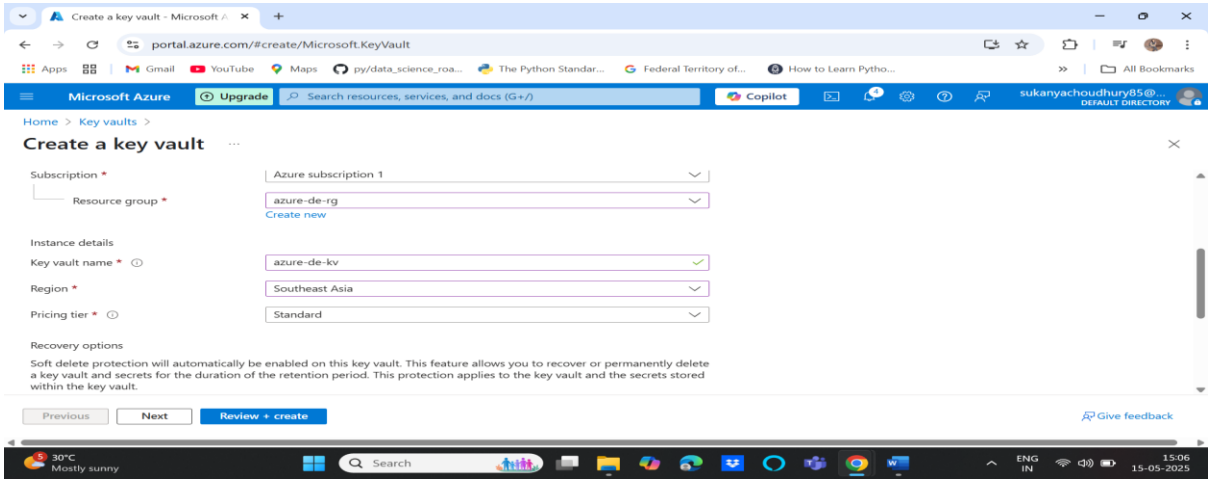
6) Create Synapse analytics workspace, “azure-de-sa”

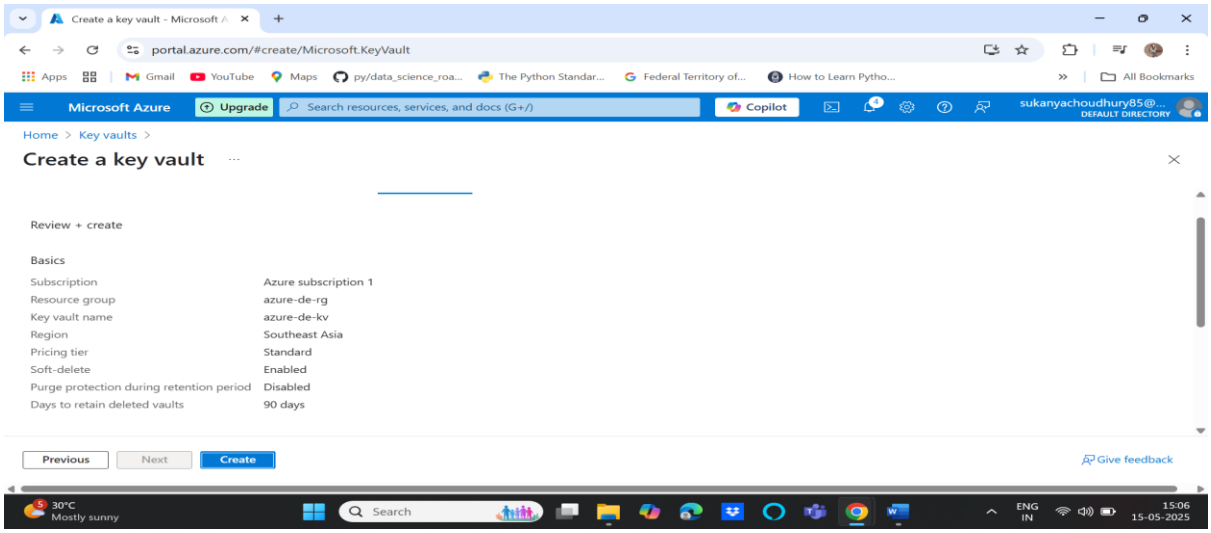


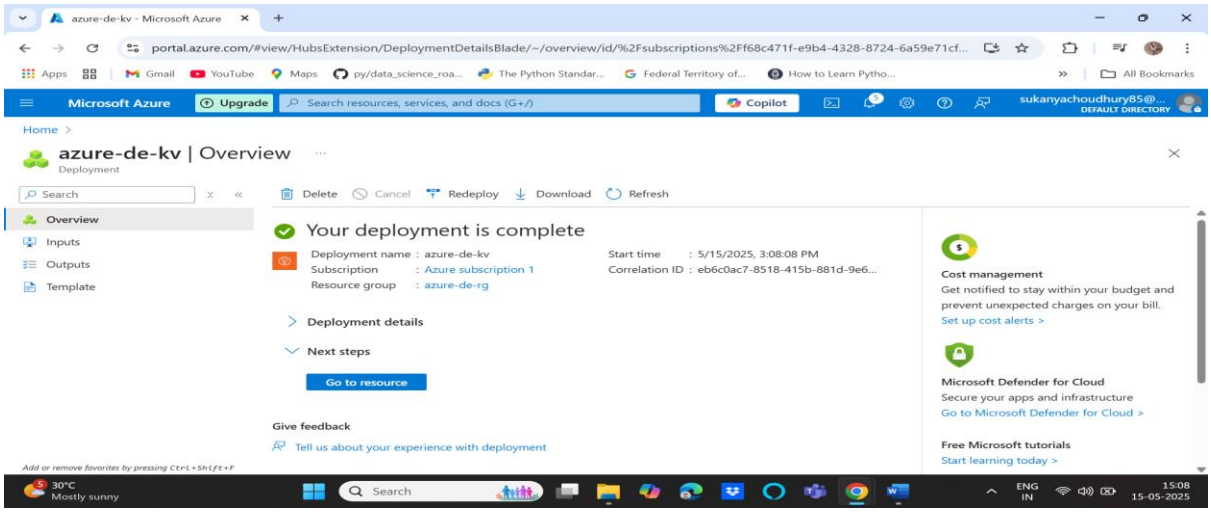




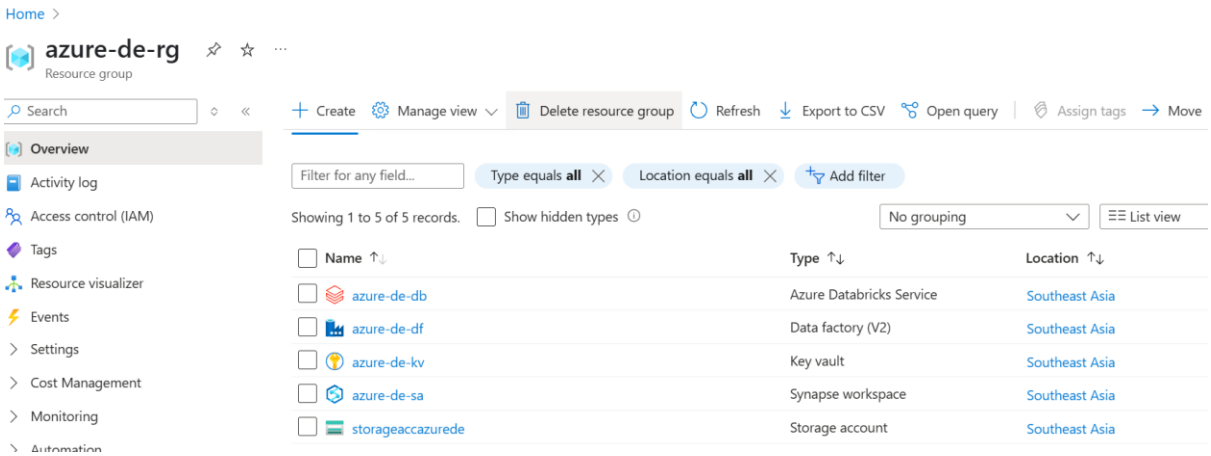
7) Create a Key Vault







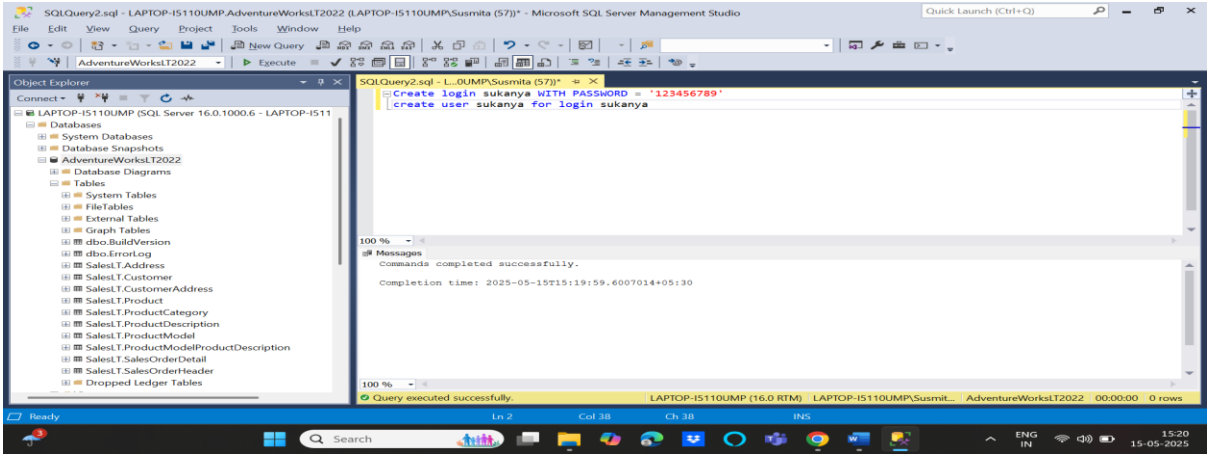
Resource group elements now are:



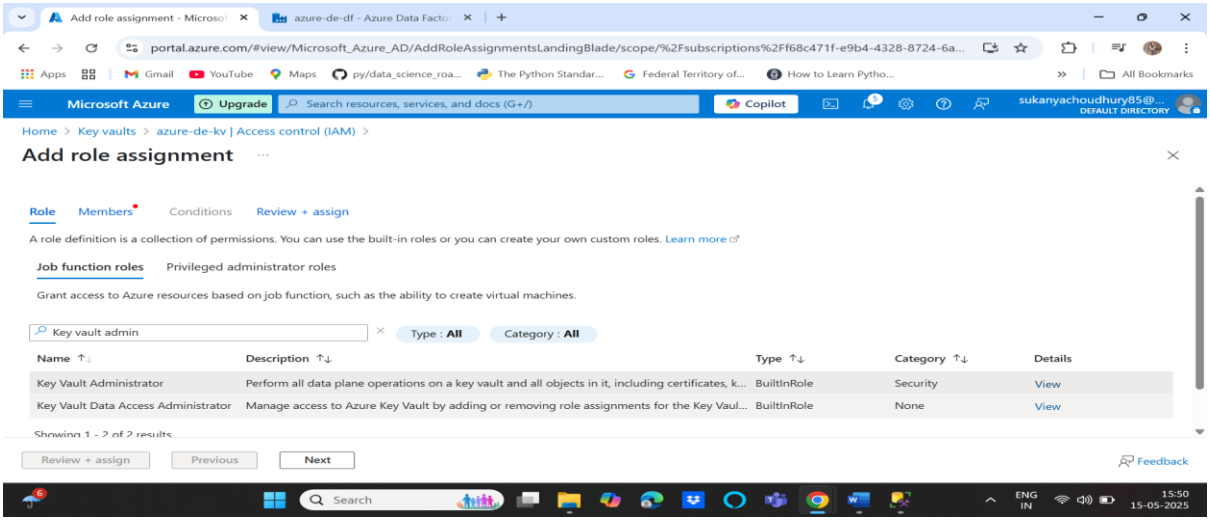
Step 2: Data Ingestion

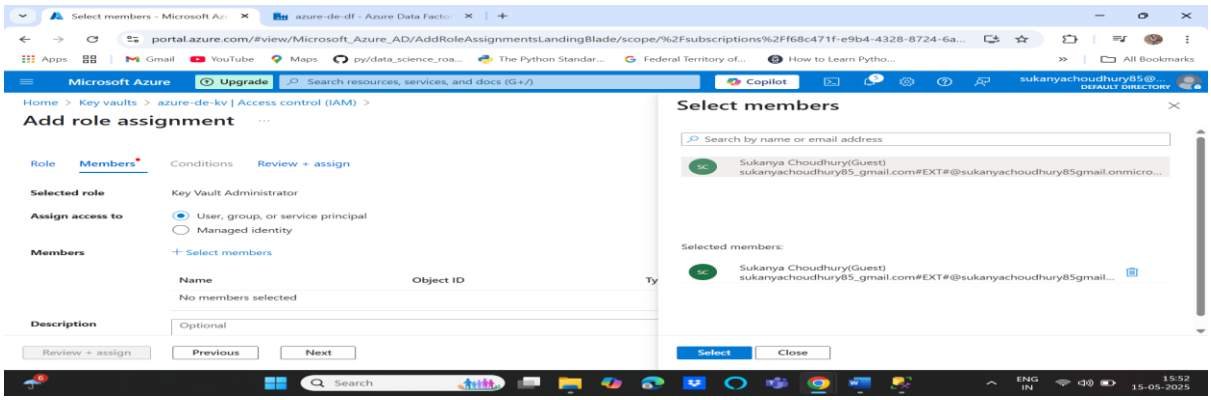
- **Extract customer and sales data from an on-premises SQL database.**

1) Create an user into the SQL server database AdventureWorks which is downloaded and loaded into the SQL server management studio from Microsoft site.

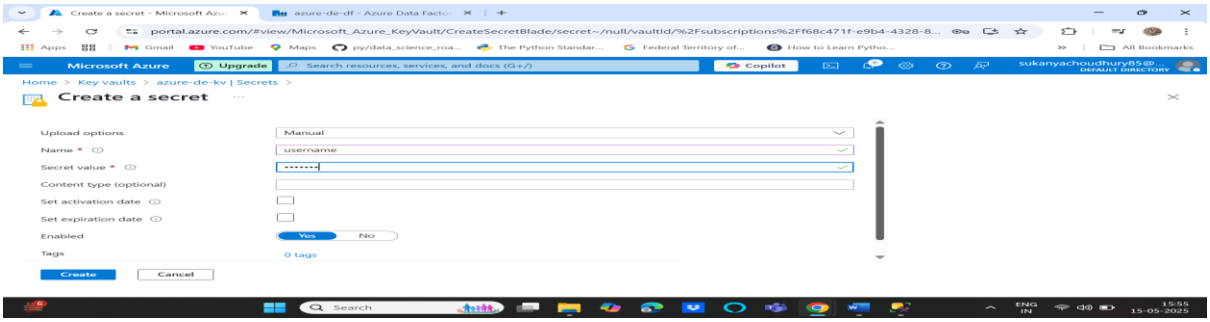


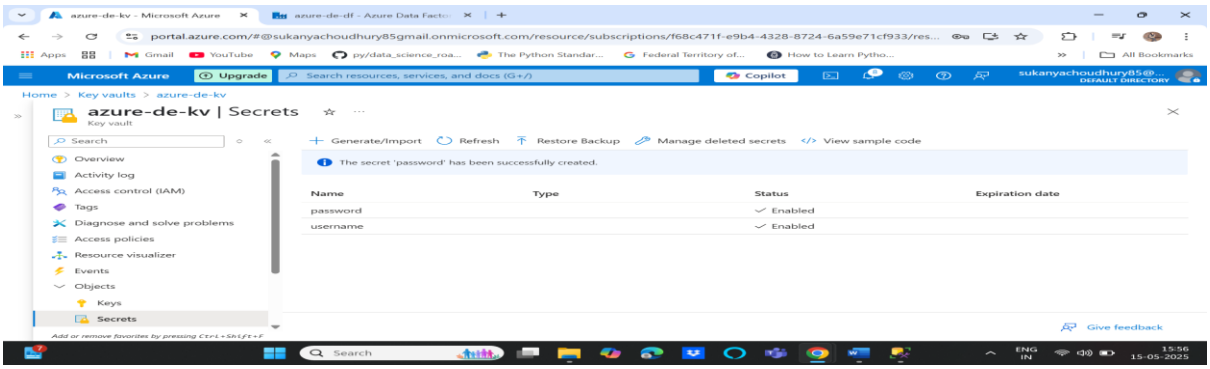
2) Go to Create key vault→ Access Control (IAM) Objects→ Add→ Add role assignment→Key Vault Administrator→ Next→ Members→ user,group or service principal→ select members→add yourself as a member





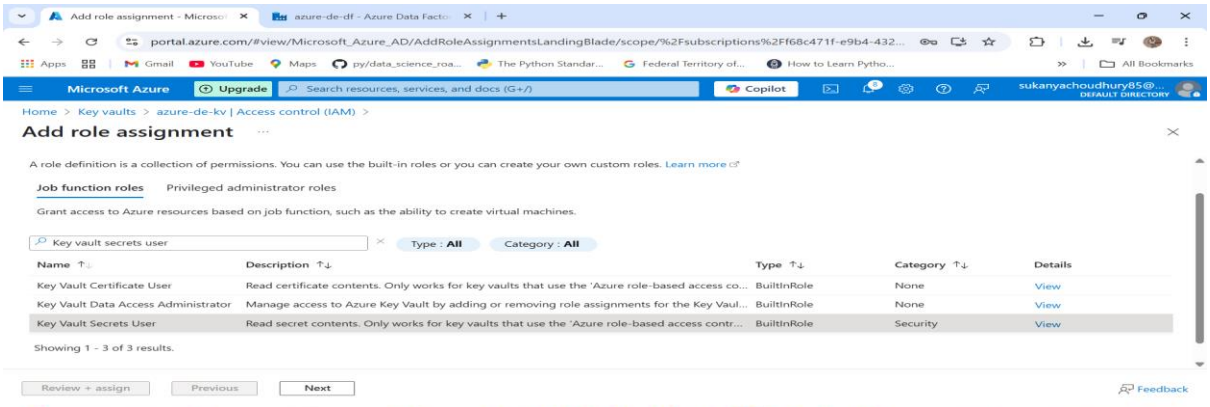
3) key vault→ objects→Secrets-> Generate/Import→ add two keys: username(Sukanya) and password (from SSMS)

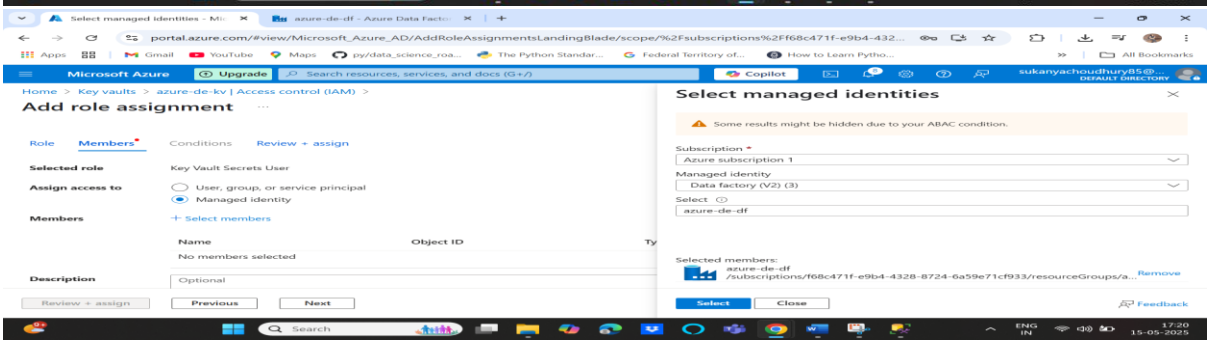


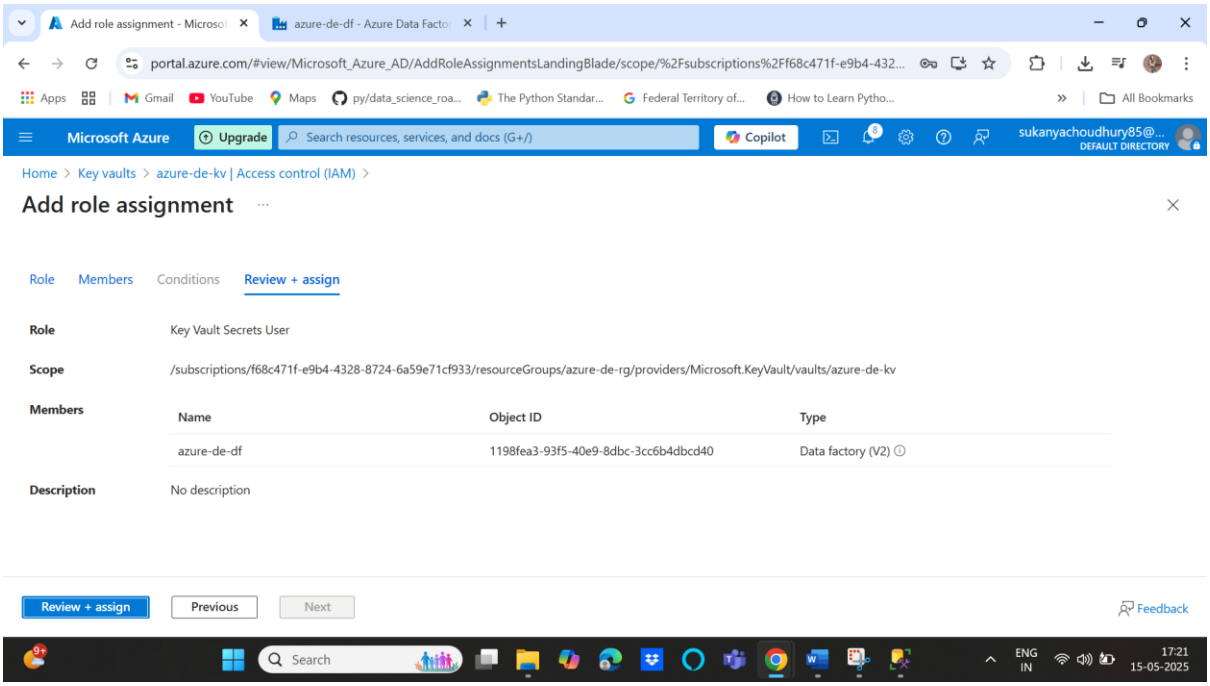


4) Add Key vault secret user to Data factory

Key vault→ Access Control (IAM) →Add role assignment—Key Vault Secrets User→ Next→add members→ Managed Identity→ select members→select your Data factory→Review+assign→Create

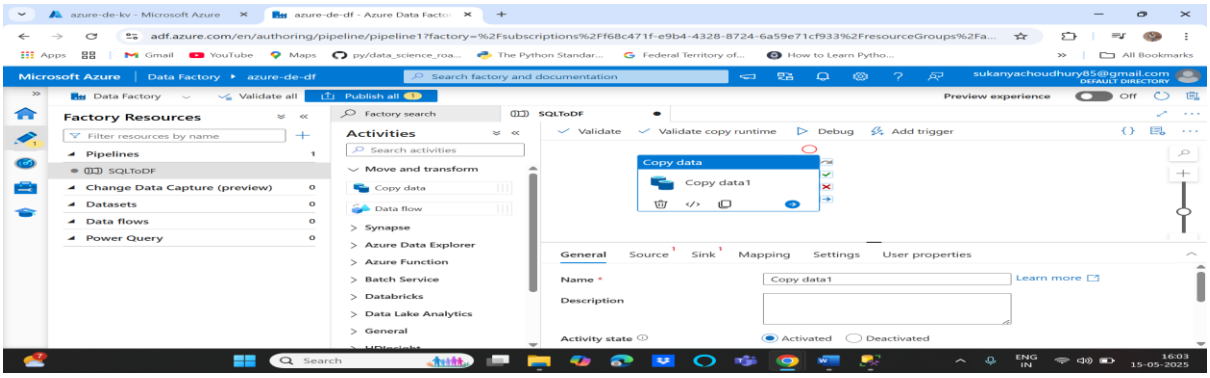






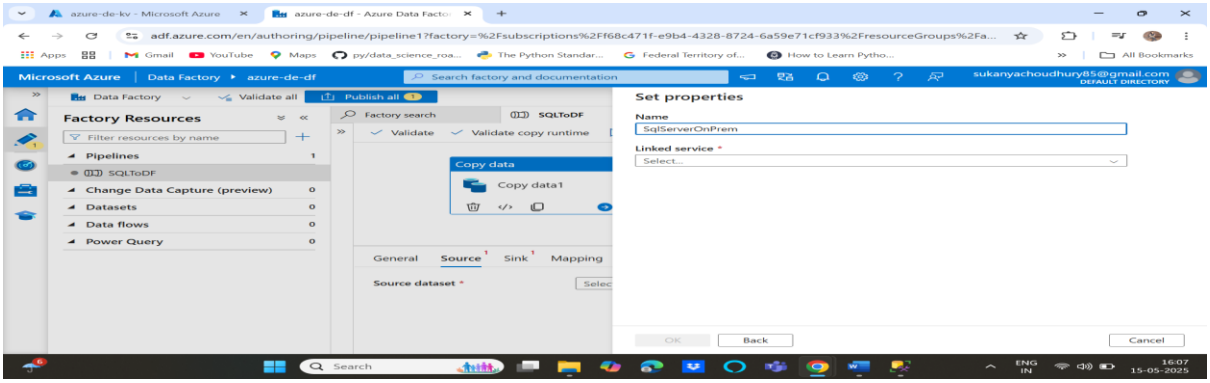
- **Load the data into Azure Data Lake Storage (ADLS) using Azure Data Factory (ADF).**

1) Add a pipeline→Activities→Move and Transform →copy activity

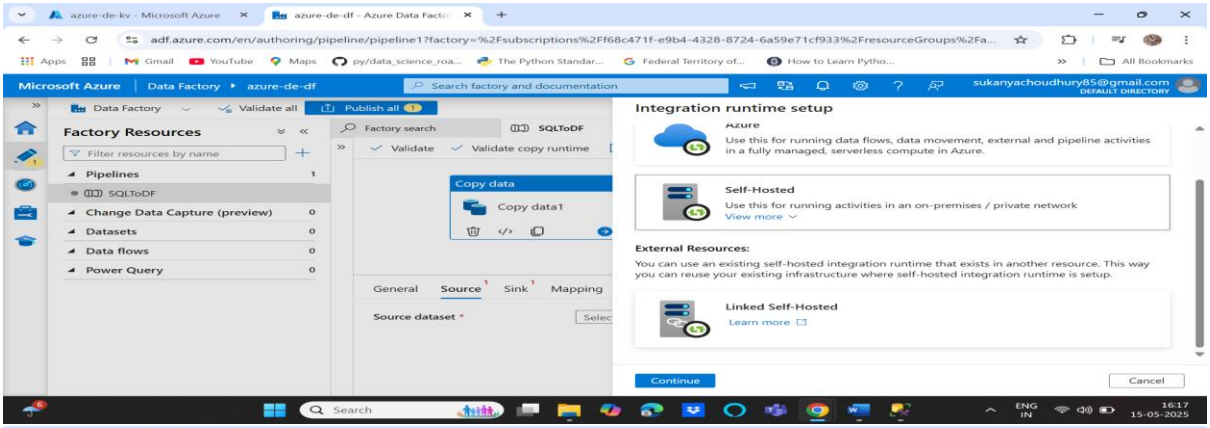


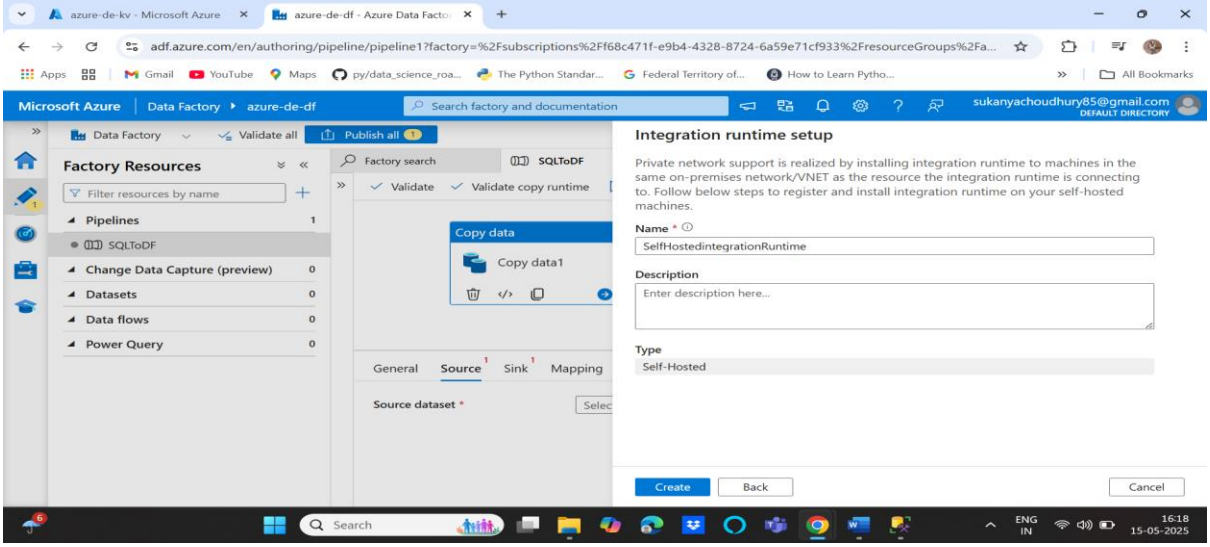
2) Configure source and sink by configuring respective linked services and source dataset

Go to Copydata→source→source dataset →+→ SQL server (as our data is in On premise SQL server) → name

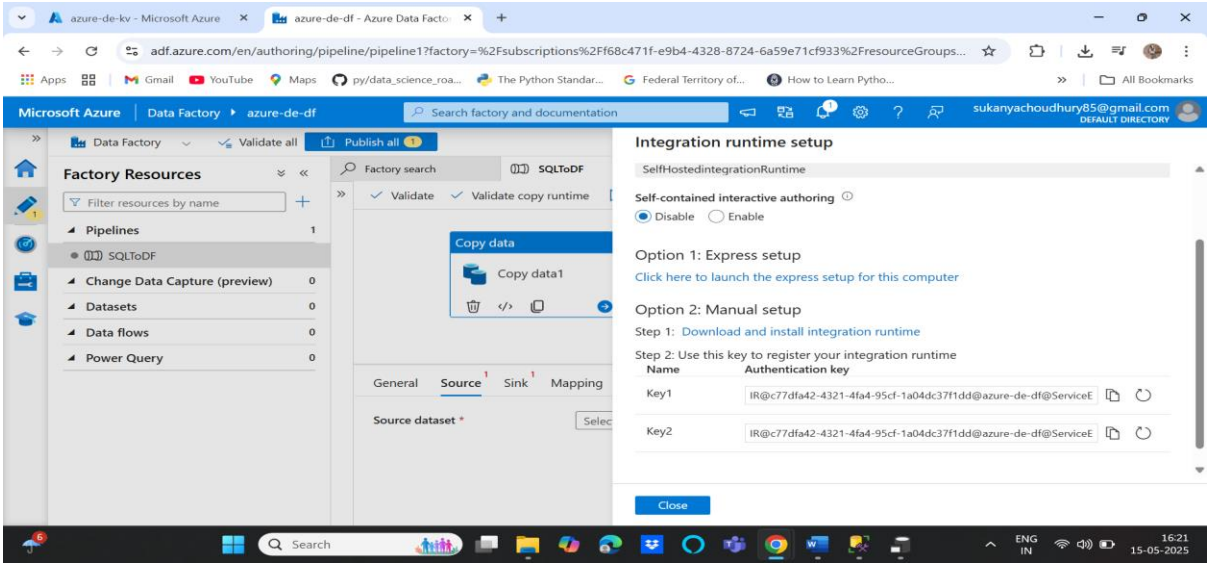


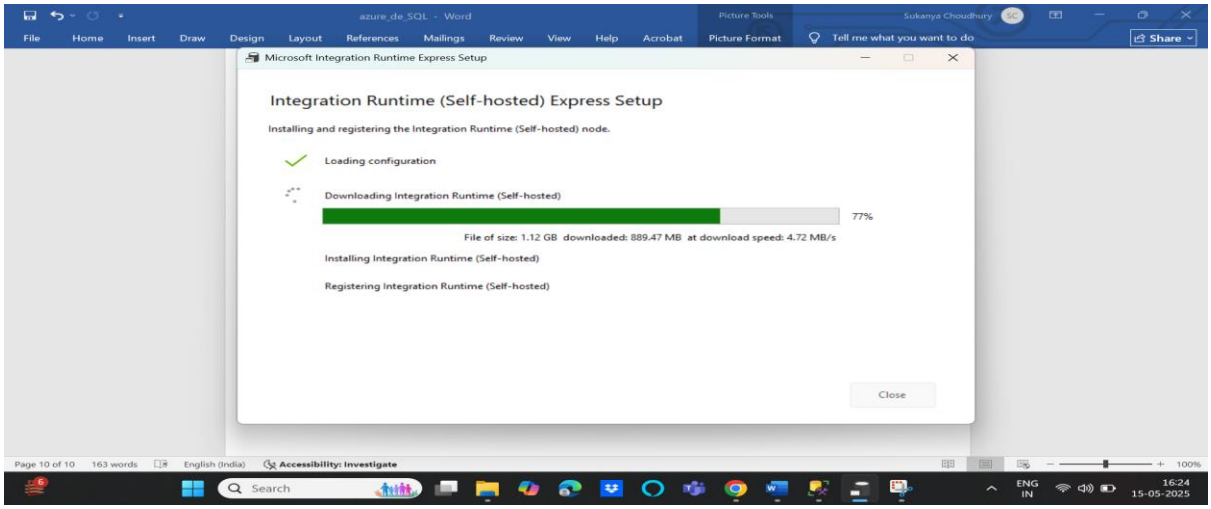
3) Then linked service → +New→ Configure linked service→ Connect via integration runtime→ new → self hosted(for onpremise database)



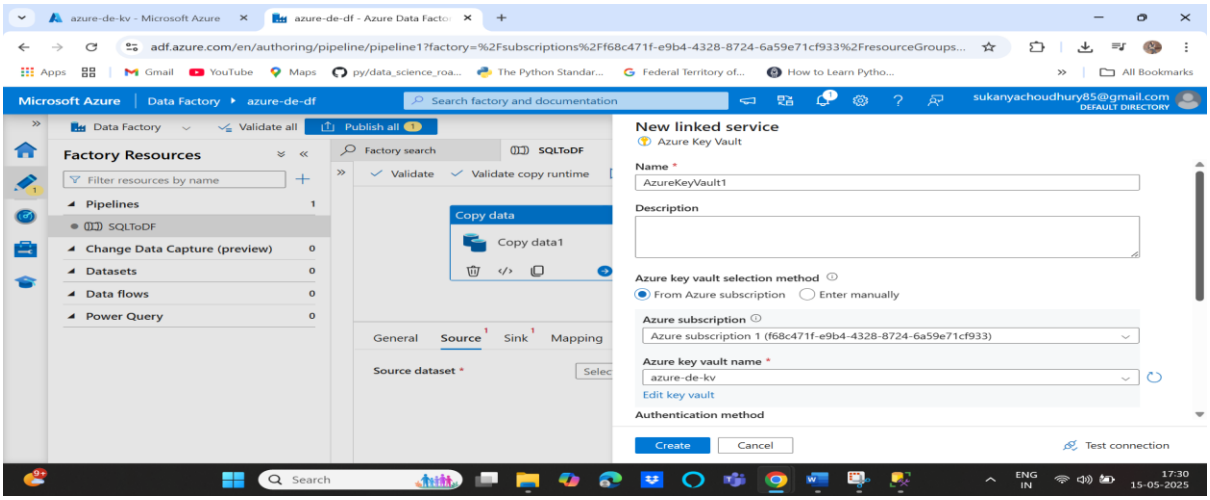


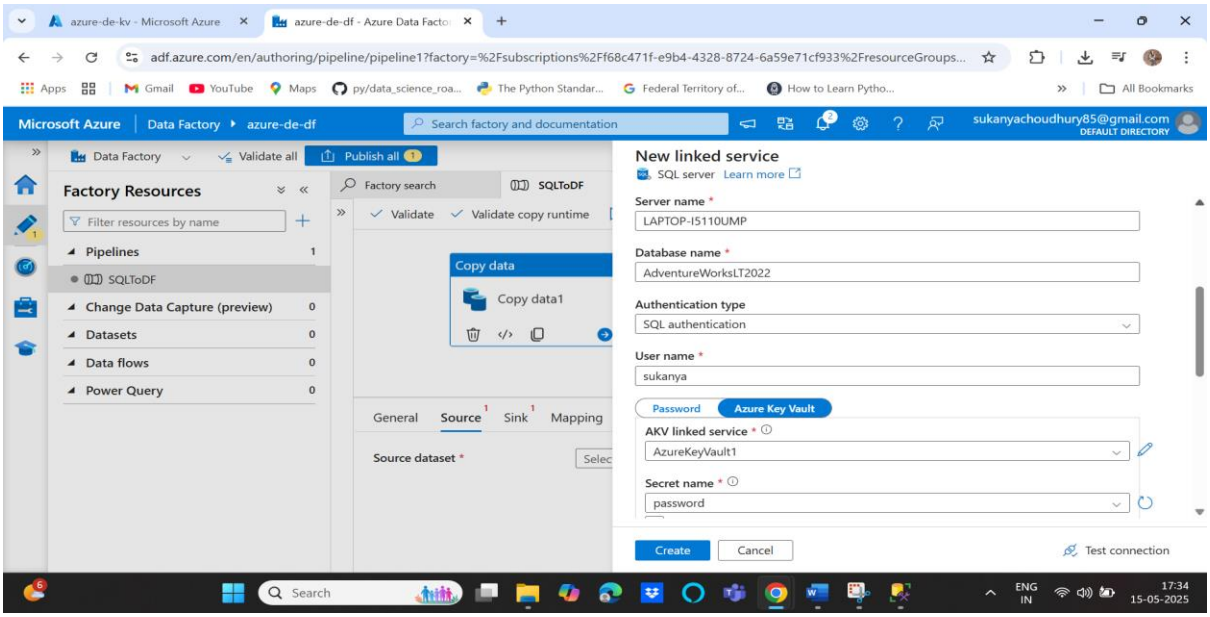
4) After creating→ Express Setup→ Click here to launch the express setup→ download and run the setup on your local machine—"SelfhostedIntegrateRuntime" is created

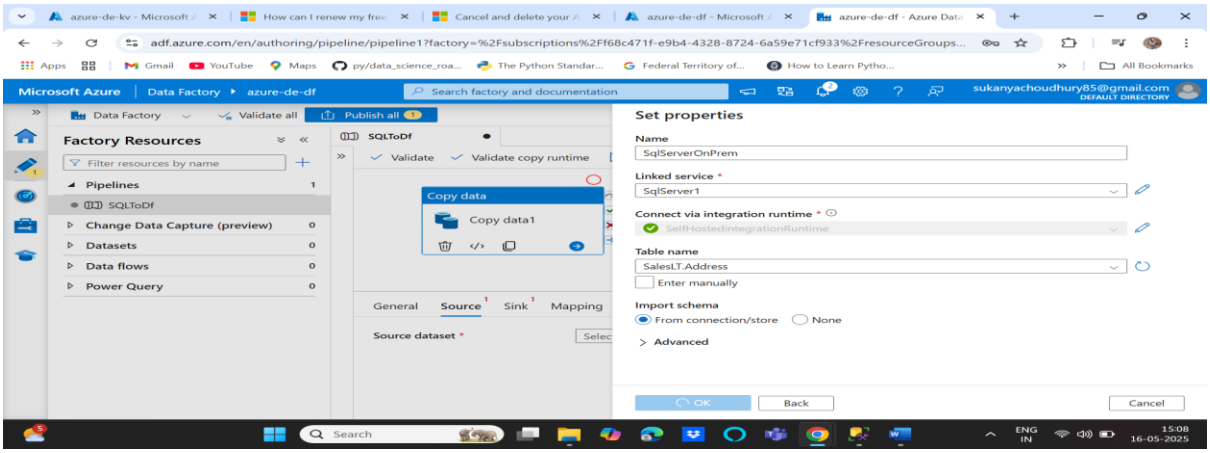




5) Use the above created Integrated Runtime for the linked service setup and give the database details and in password →Azure key vault→create a AKV linked service and then use the linked service and the secret key password created before so that the final sql serve linked service looks like below SS2→ create the linked service

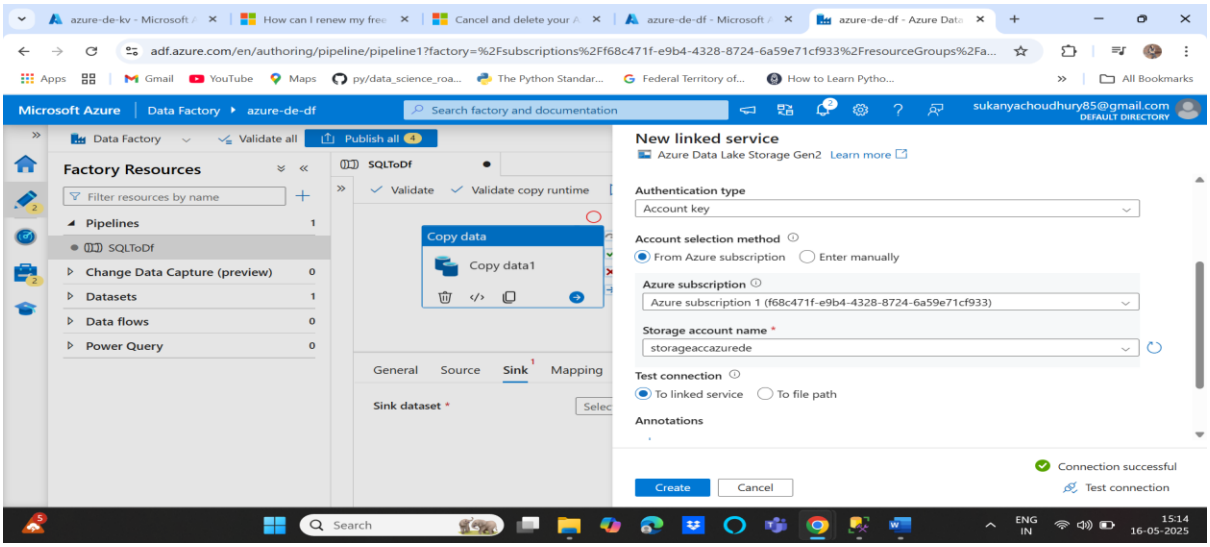




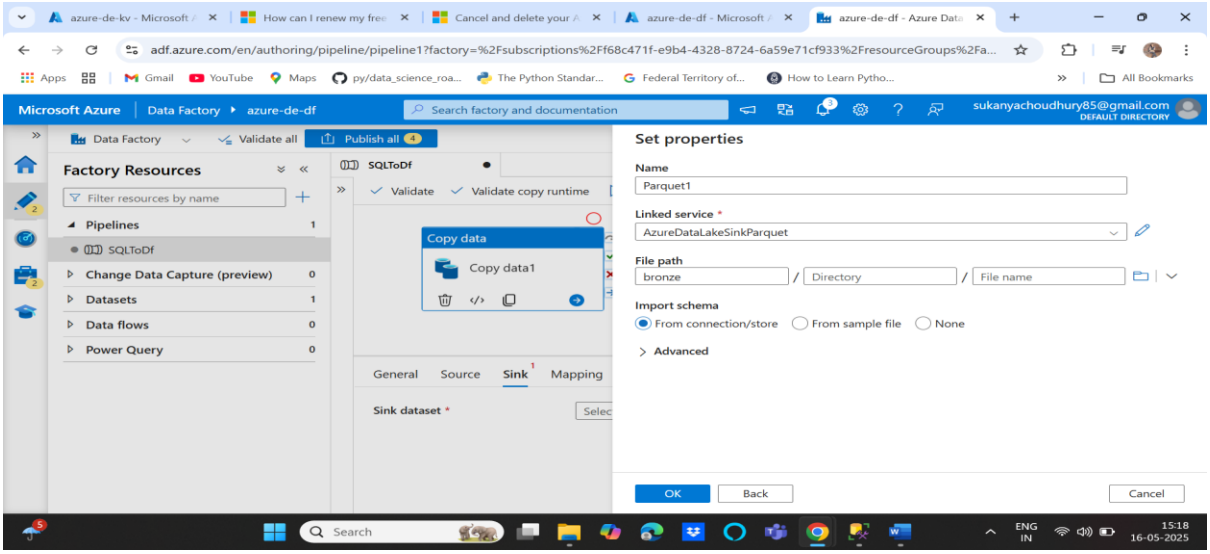


6) For sink again set up the data source using azure data lake gen 2 and file format parquet and also create a new linked service for the data source

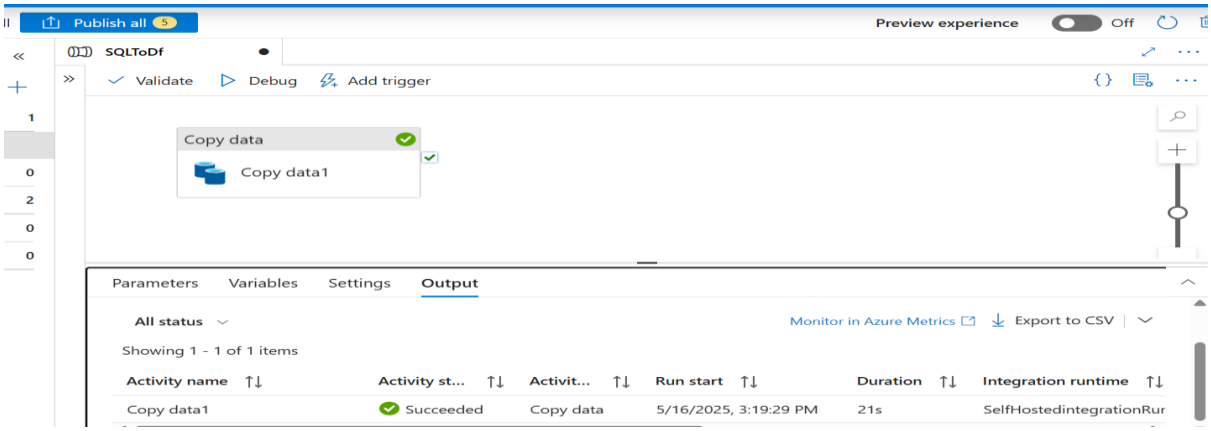
The integration runtime will be the default “AutoResolveIntegrationRuntime” this time as this is an azure service i.e. azure data lake that we need to store data into.



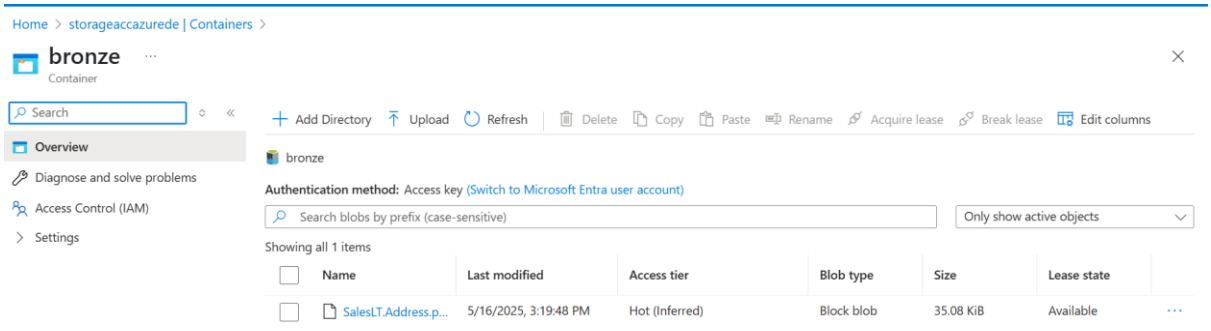
7) Then after selecting the above linked service, we need to give the destination folder in the data lake so select bronze container in the file path



8) Then Debug to run the pipeline once to check any error.



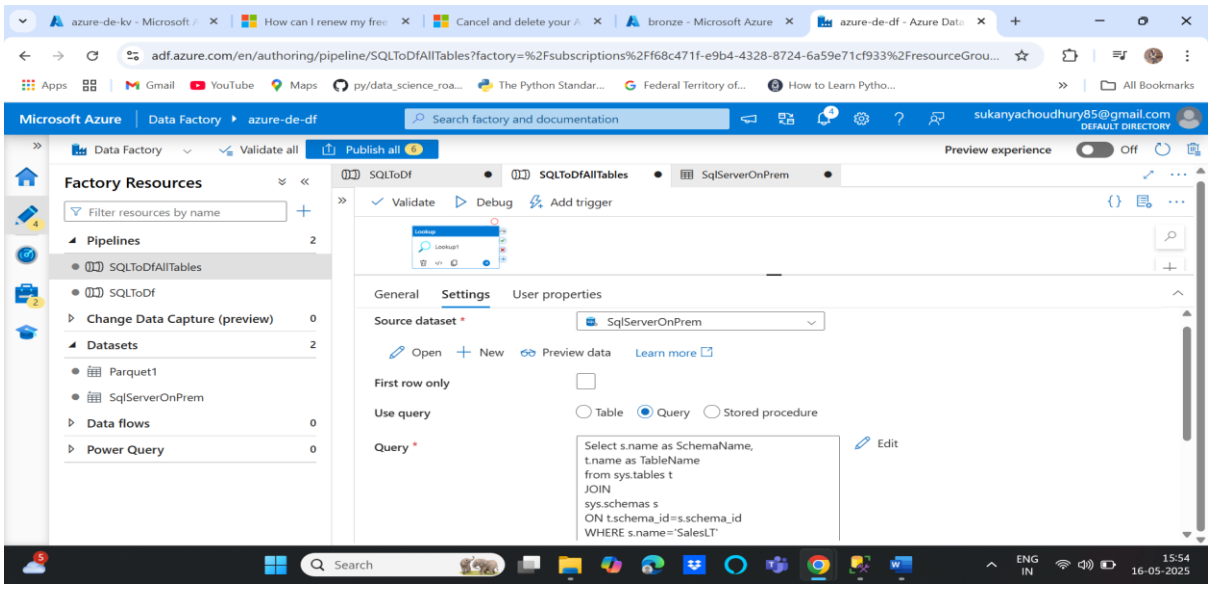
Confirm after checking storageaccount→containers→bronze whether it's copied:



9) Above was example to copy a single table, **how to dynamically copy all tables:**

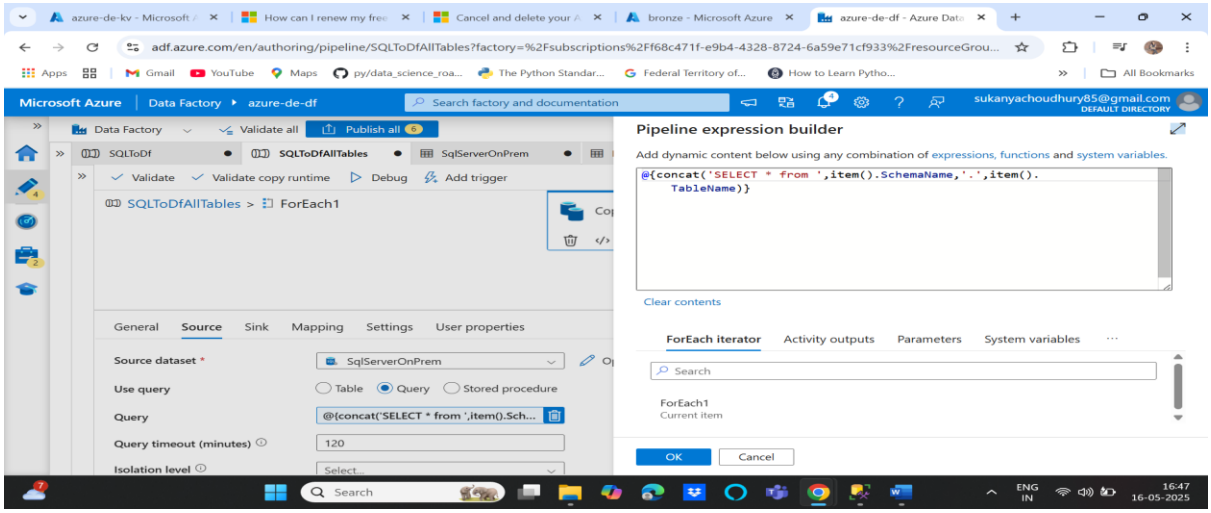
Create another pipeline with a lookup, copy and foreach Activity

Configure lookup activity by setting up the source dataset or using the one created in the last step " SQLServerOnPrem". Then open and edit it and remove the selected table as we need to use all the tables. Then in the lookup activity use query as below and uncheck "First row only" then debug to see all tables in output

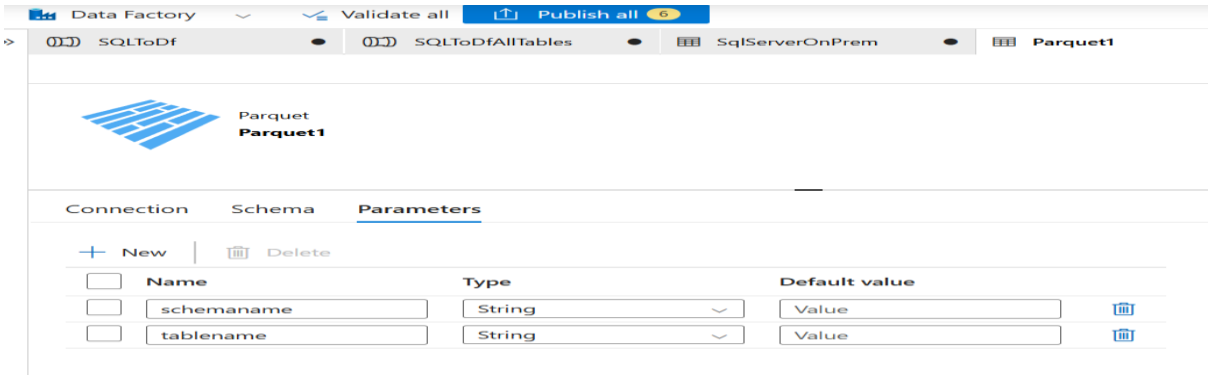


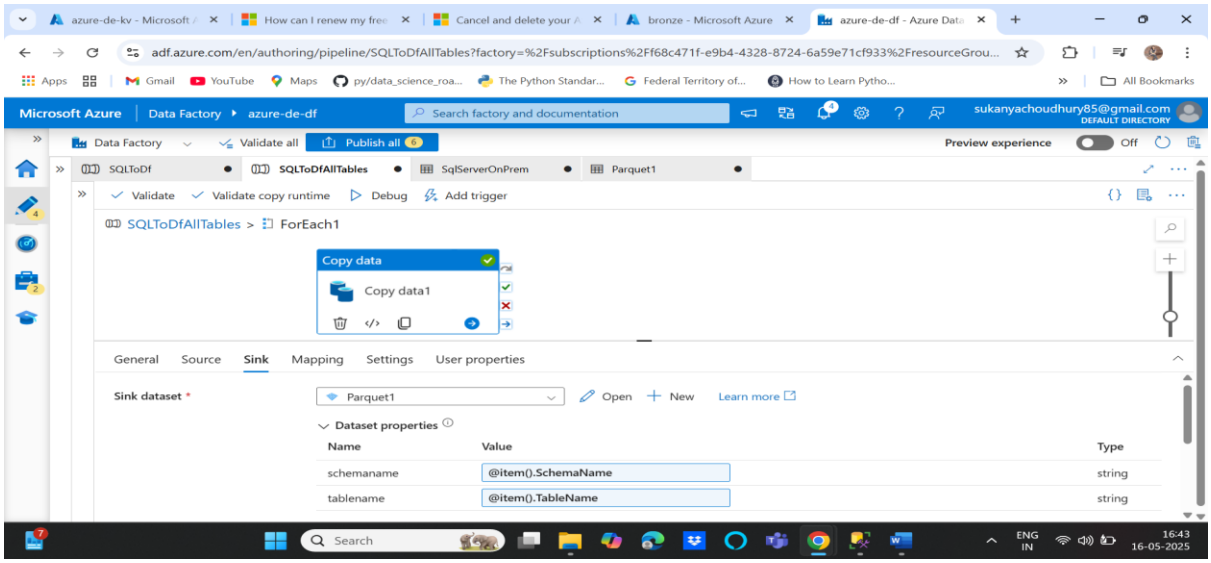
10) On success of this lookup activity, add a foreach activity and inside the foreach activity add a copy activity with below configurations

Source dataset→ SQLServerOnPrem(created before) and in query→add dynamic content with below query which runs a sql query by concatenating schema and table names

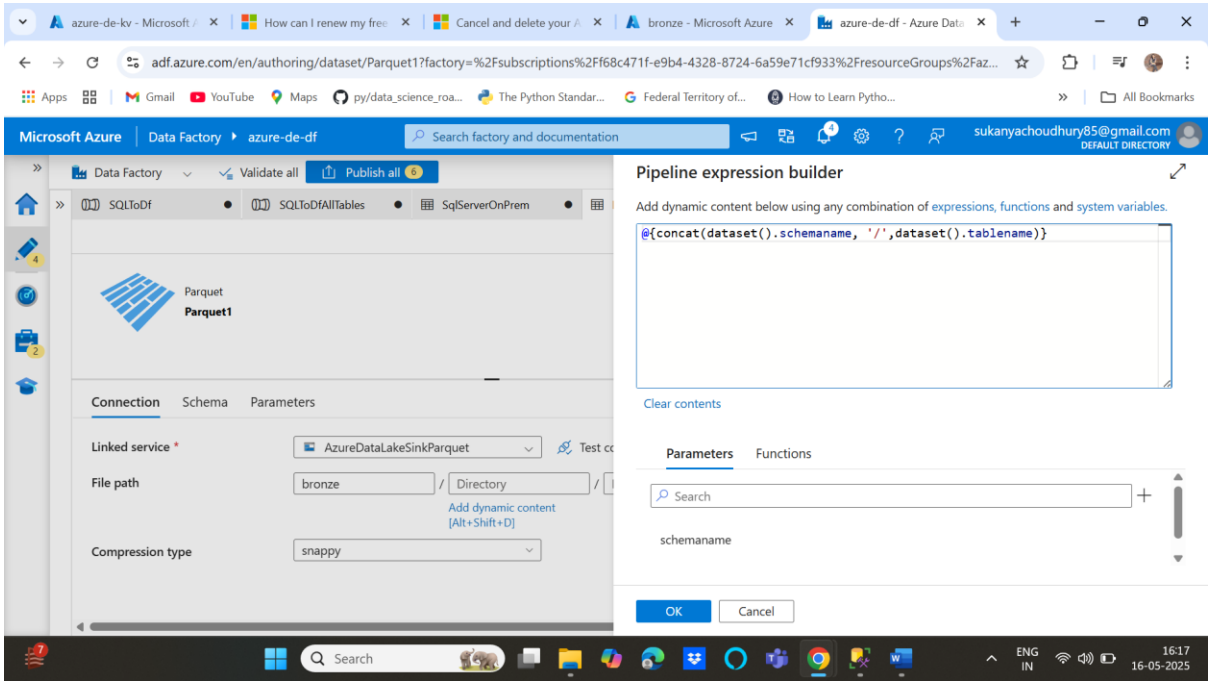


For sink, use Parquetsink Ds created before and edit it to create parameters to give the destination folders as per schema and table name and then pass values as dynamic content to them in copy data→ sink tab

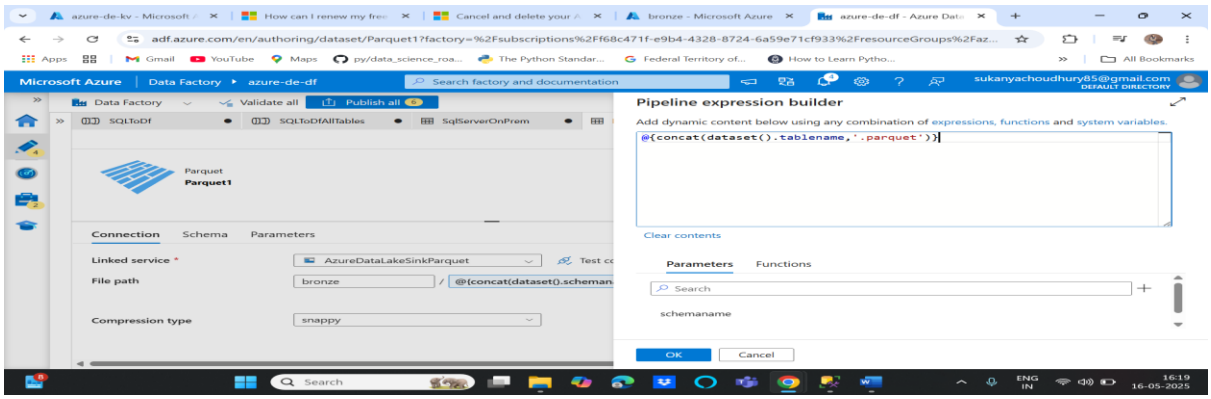




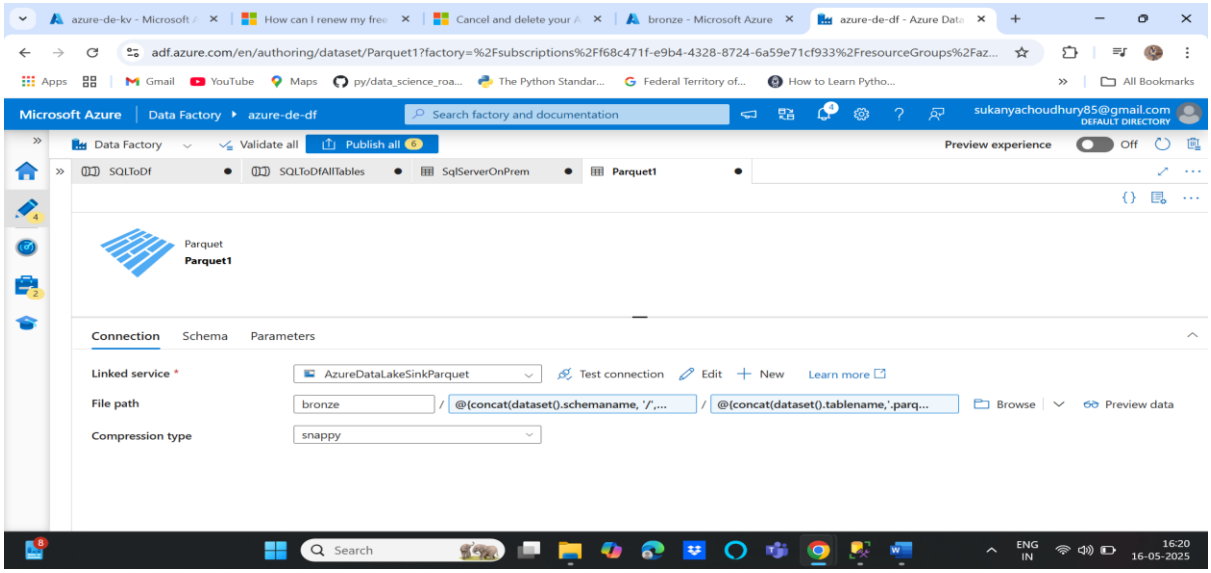
Now, pass these parameters into the sink directory as add dynamic content as below



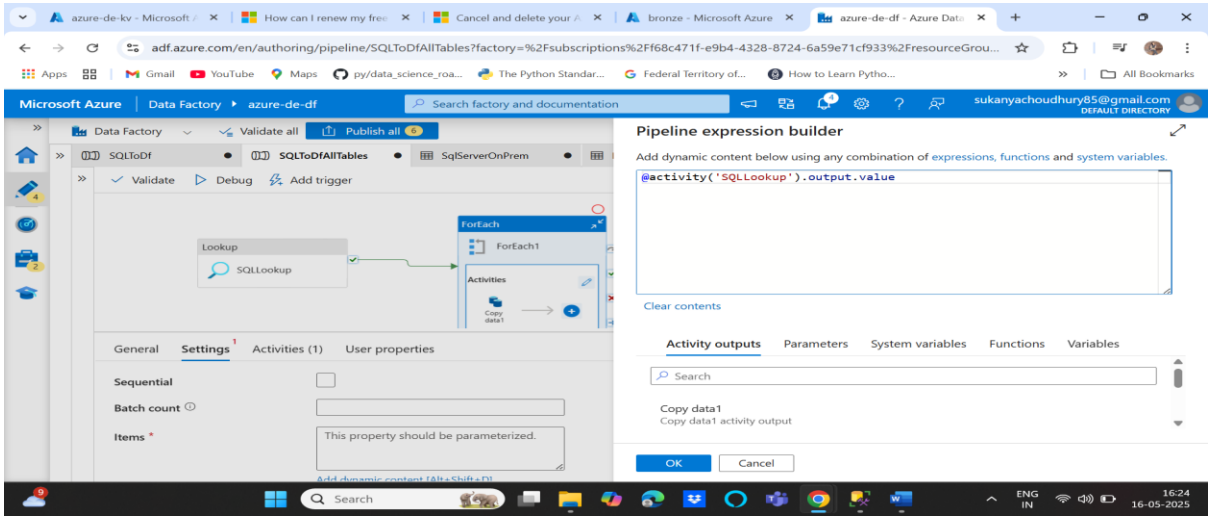
And the filename to save as parquet file as below



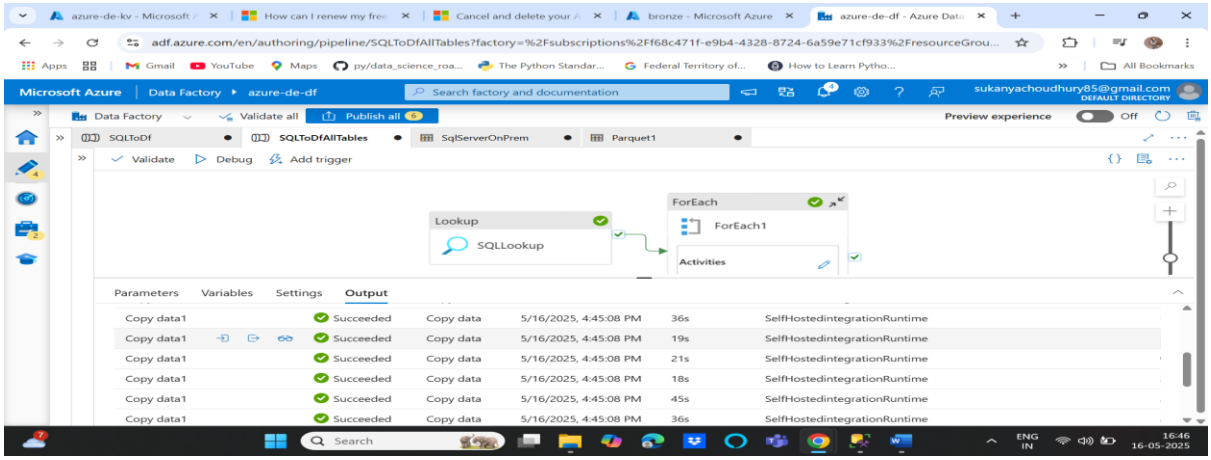
Finally sink data set looks like below

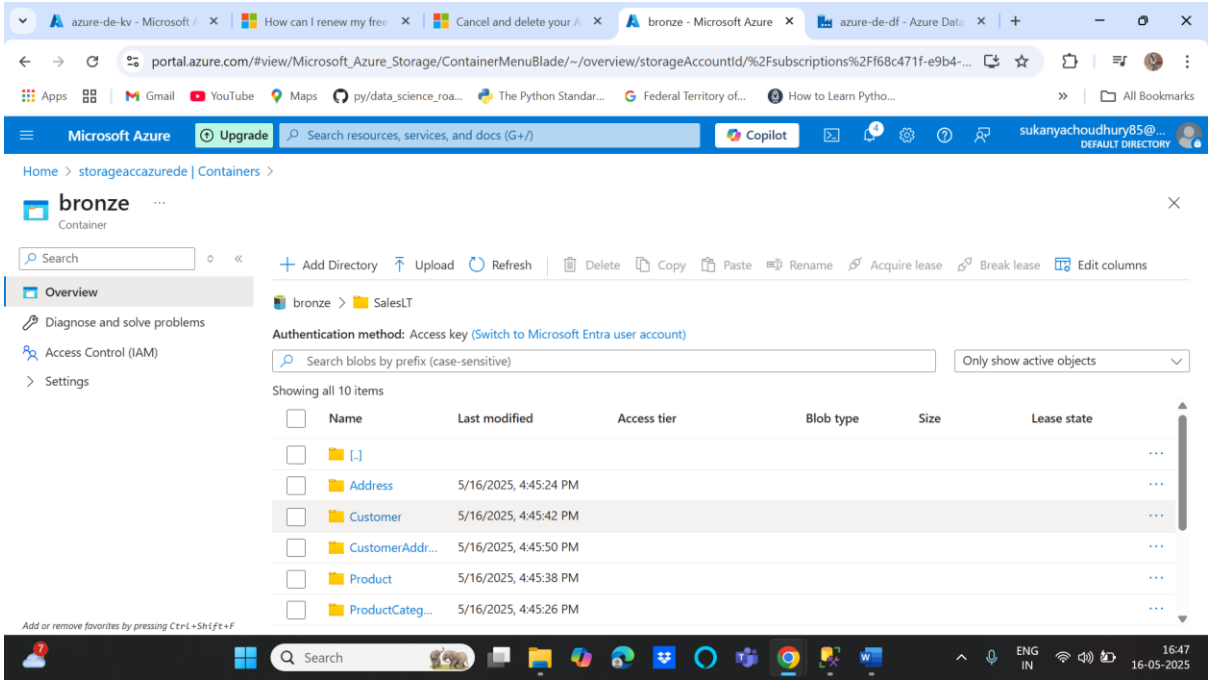


Now setup the foreach activity to get the lookup activity output value:



11) Debug lookup activity which runs the whole pipeline and extracts table from sql server into data lake bronze container and then publish all

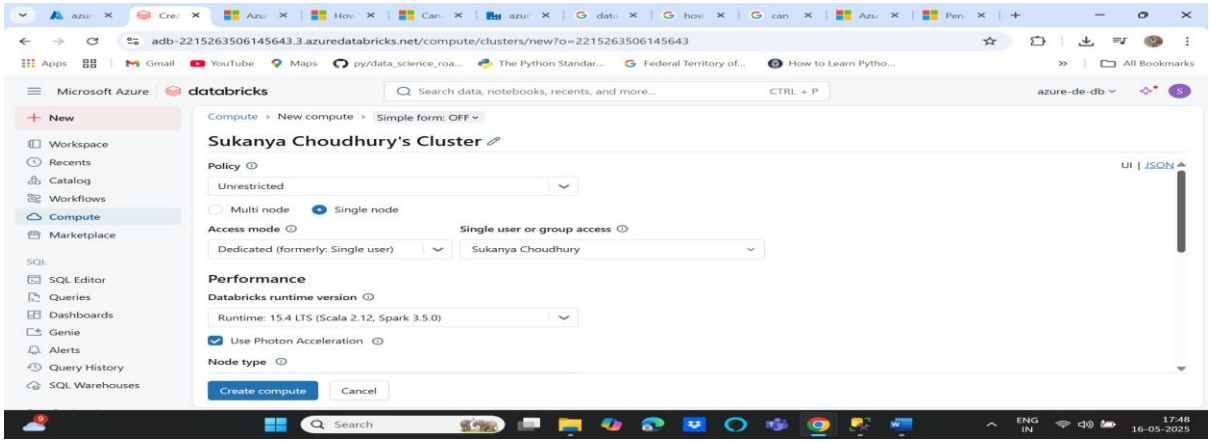




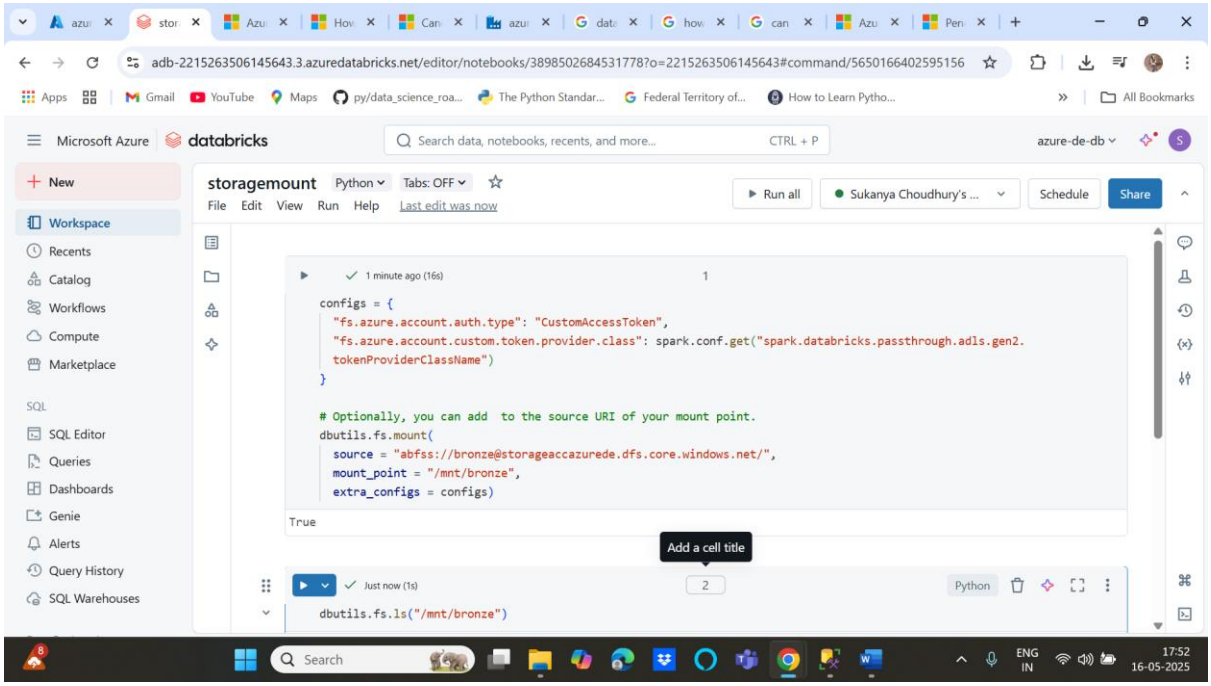
Step 3: Data Transformations using Data Bricks

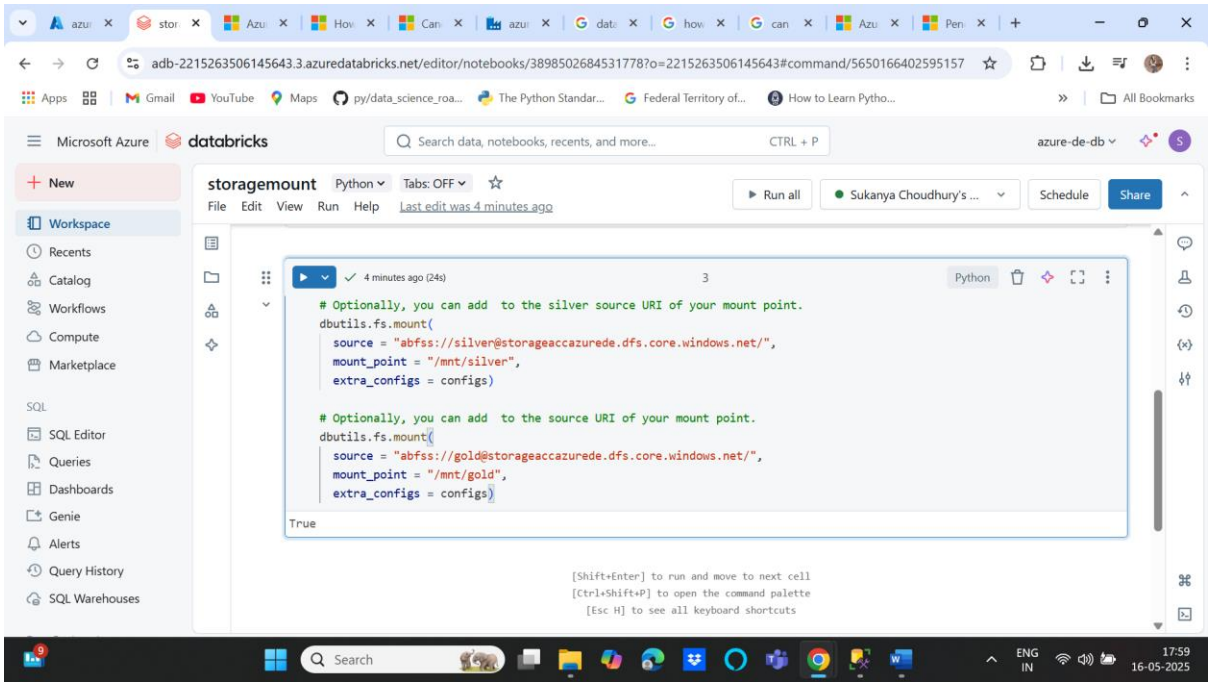
1) Create a cluster

Launch Databricks Workspace→Compute→Create Compute



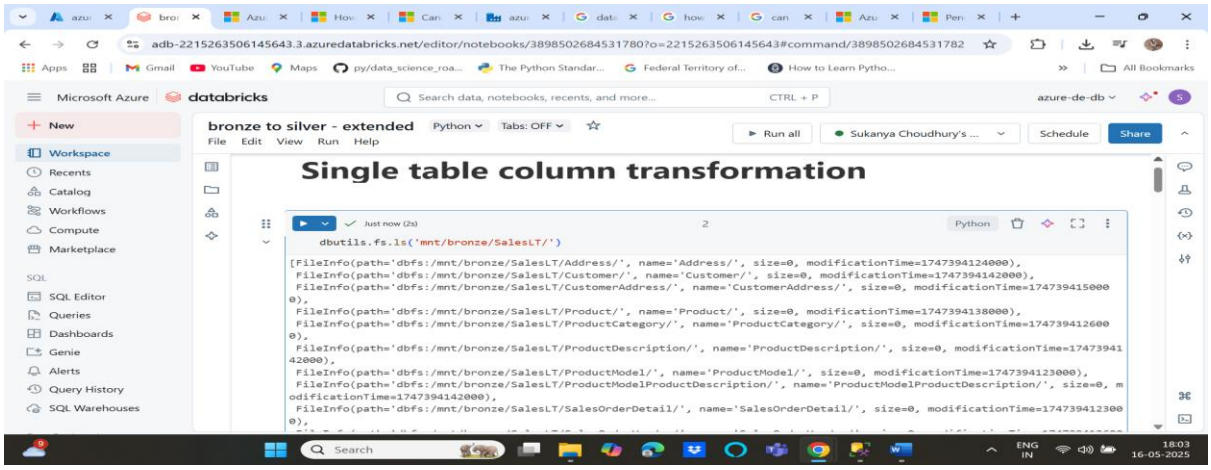
2) **Mount Data Lake in Databricks:** Configure Databricks to access ADLS using below notebook



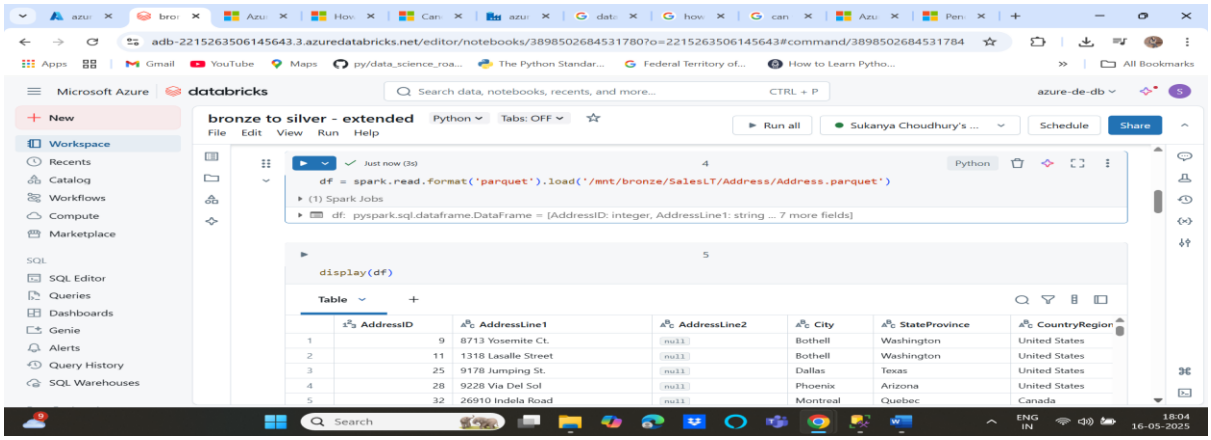


3) **Transform Data:** Use Databricks notebooks to clean and aggregate the data, moving it from `bronze` to `silver`

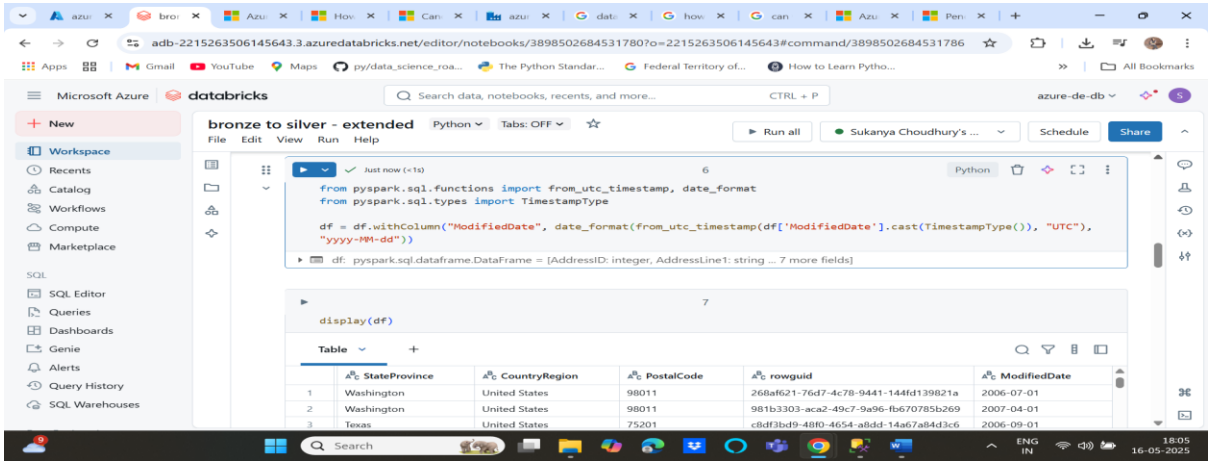
Bronze to silver : list bronze tables



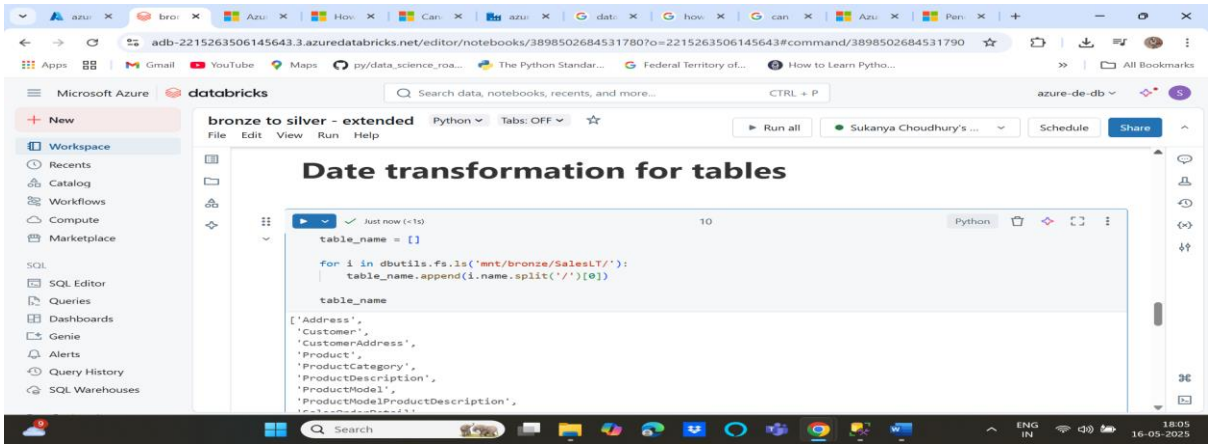
Load the address table from bronze container

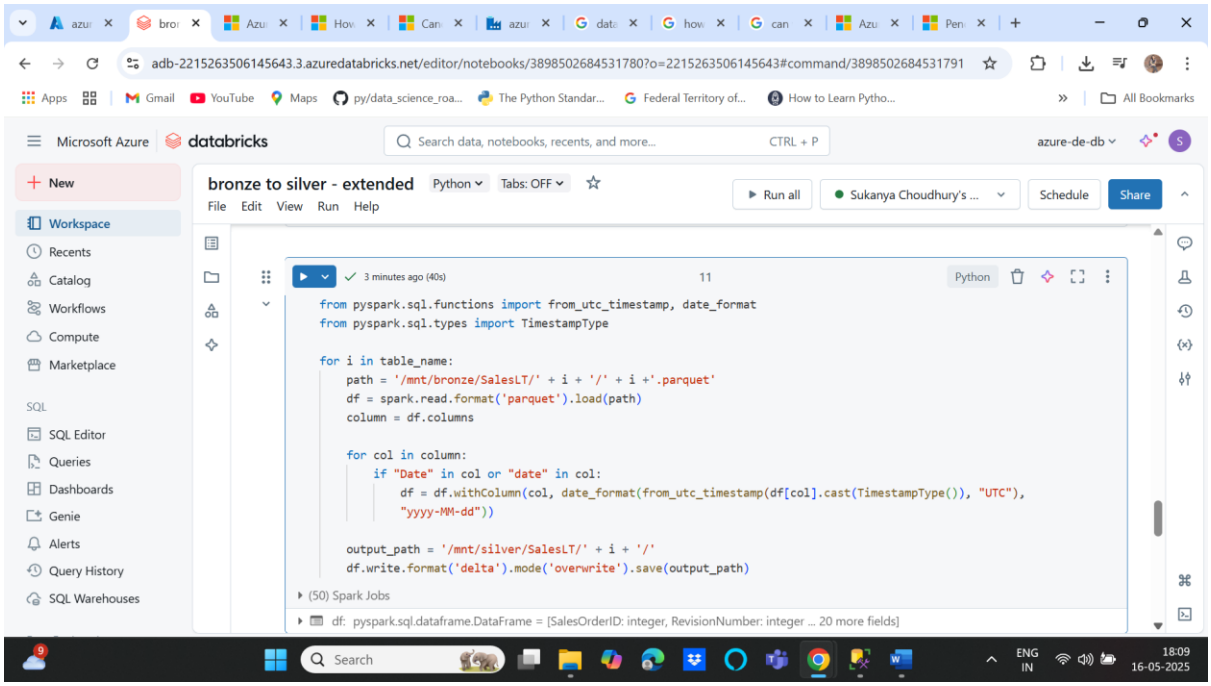


Modified date is transformed into UTC date format for Address table

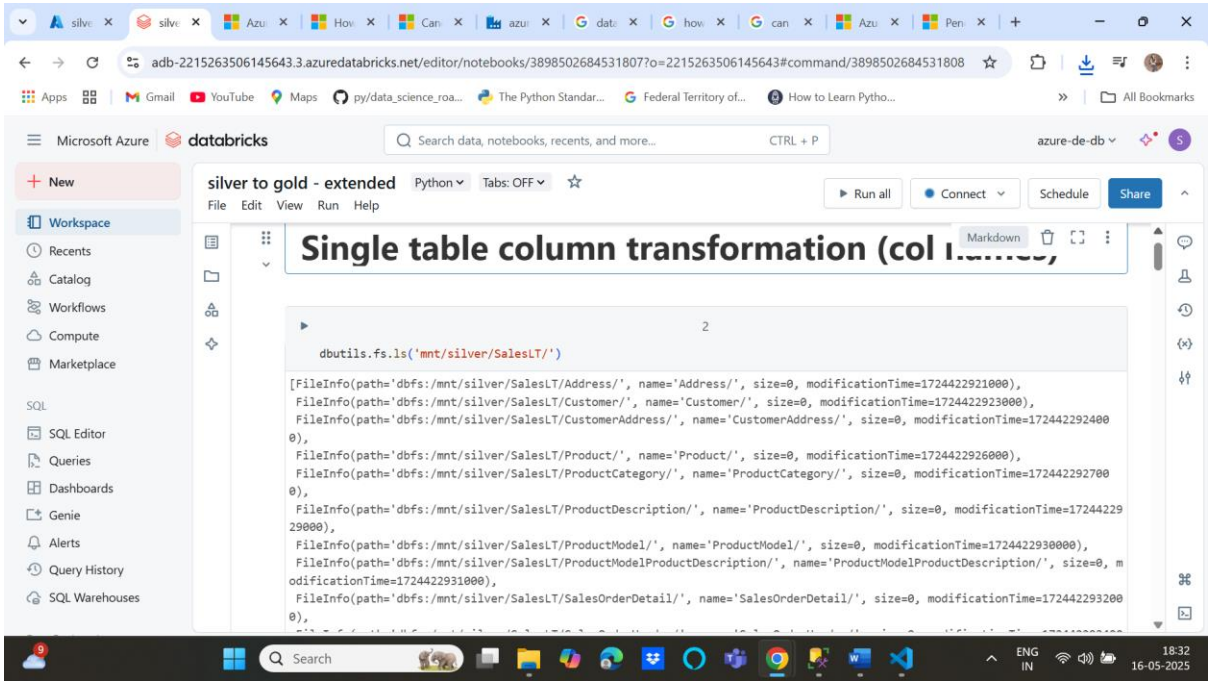


Transformation for all tables: Get the table names in a list and then iterate through each table. Read the data from bronze container by Getting the path by appending tablename to the mount path and the parquet file name and then carry on the data transformation by searching for date columns and then save all tables as delta mode into the silver container

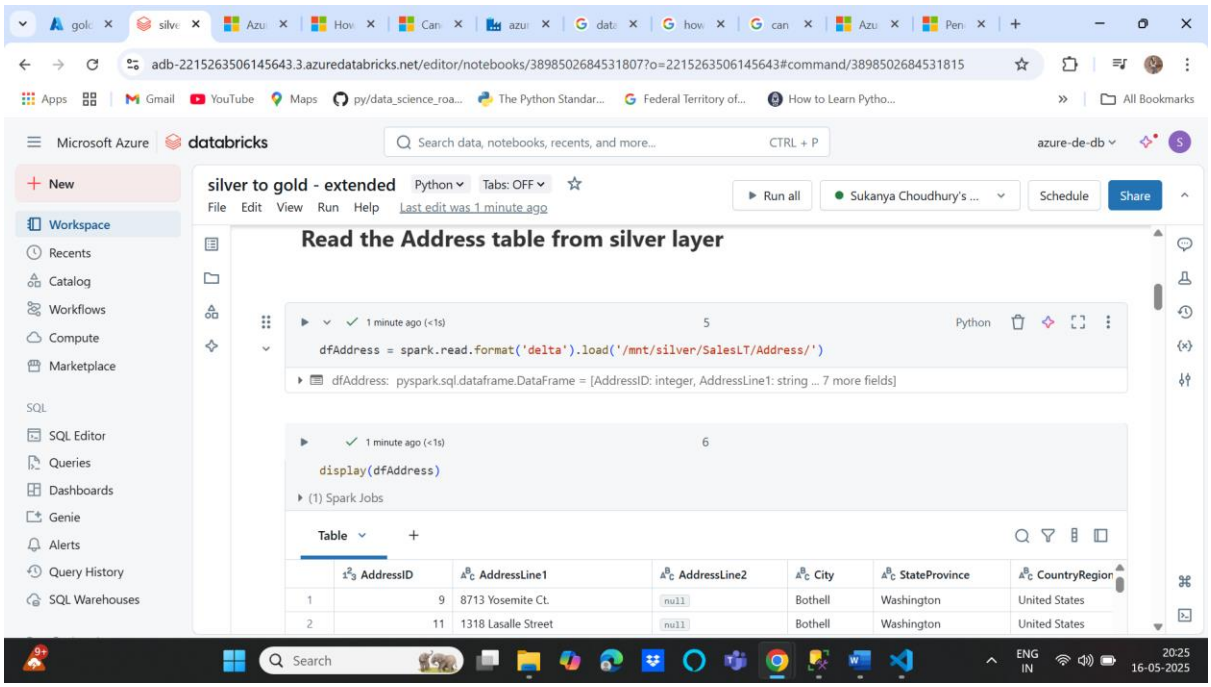


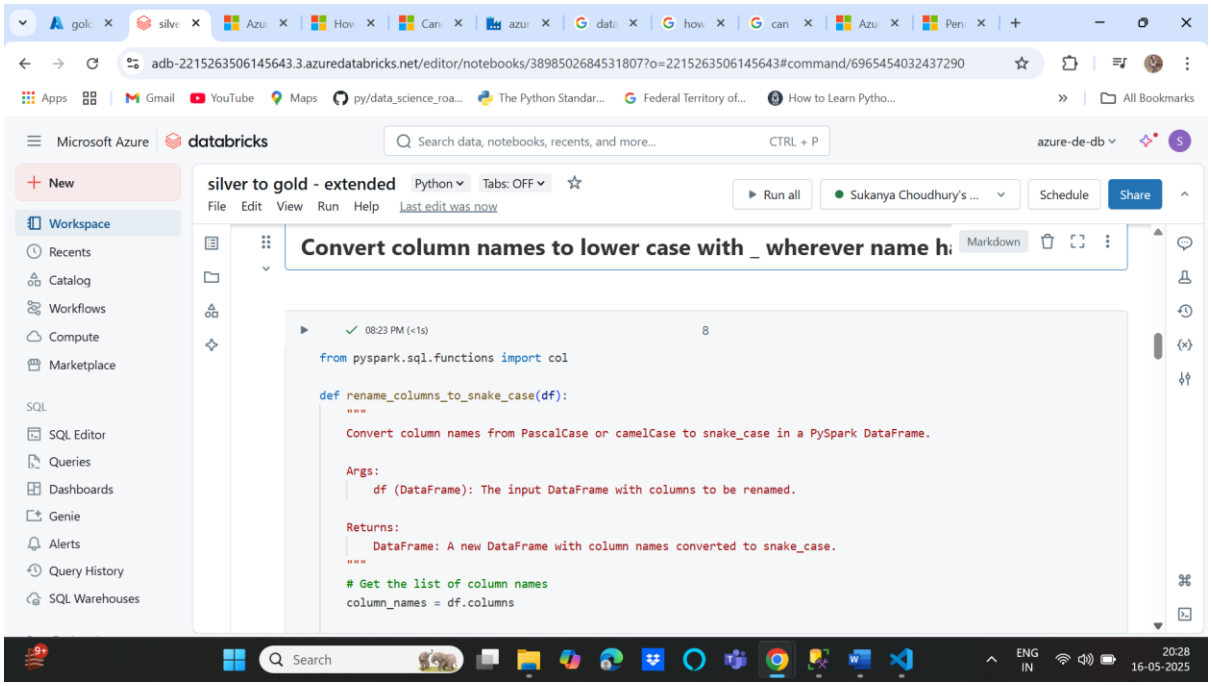


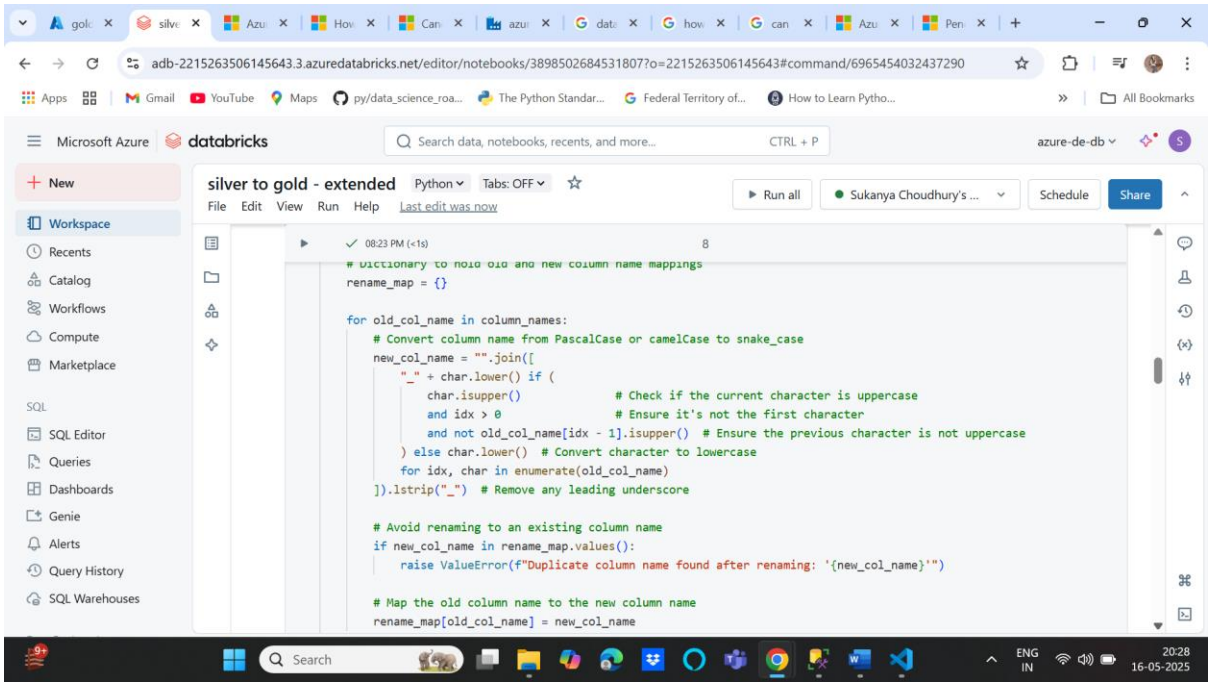
Silver to gold transformations

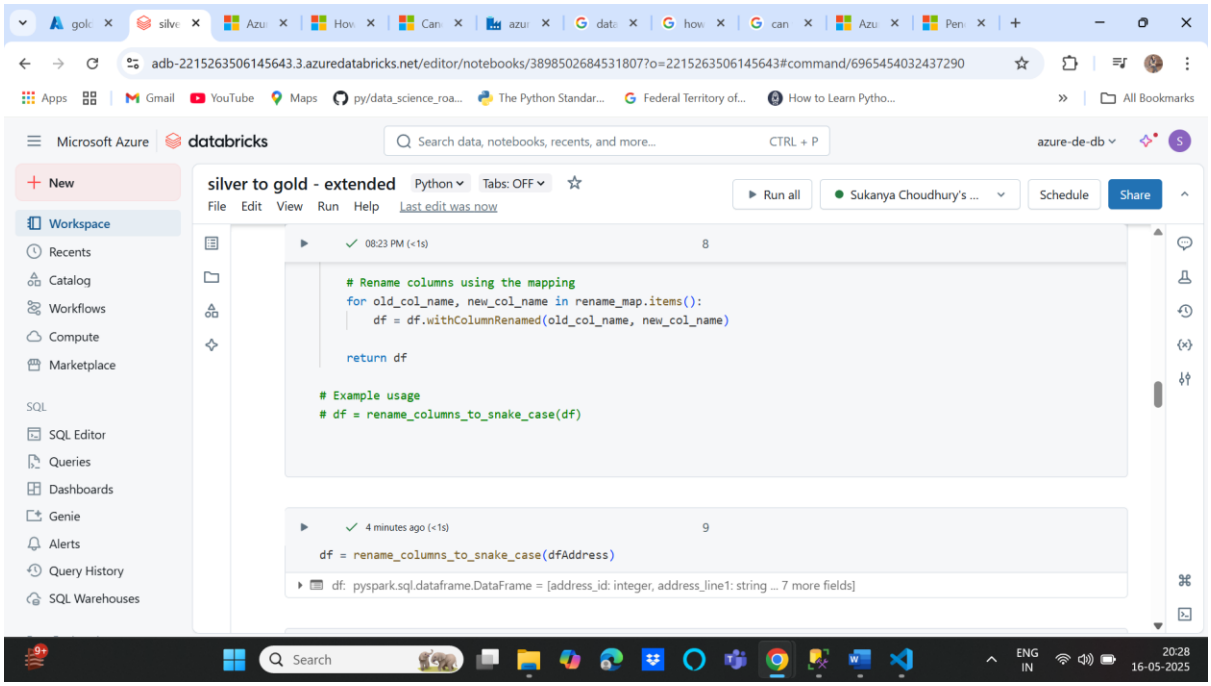


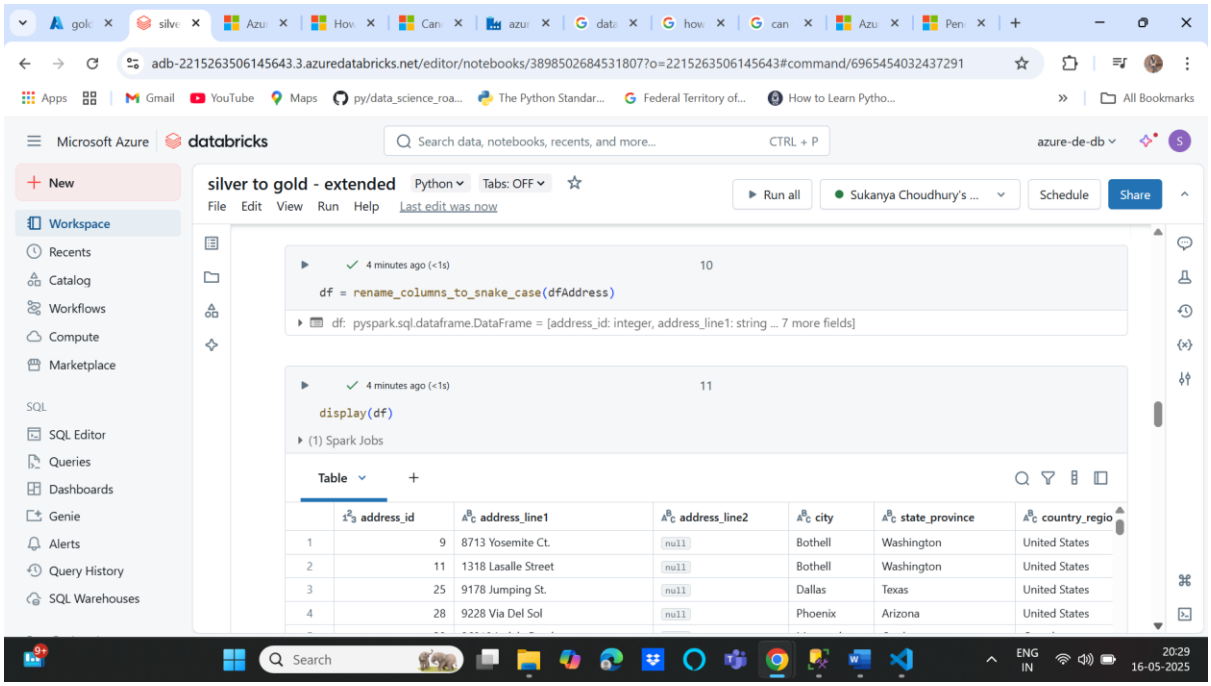
Silver to Gold

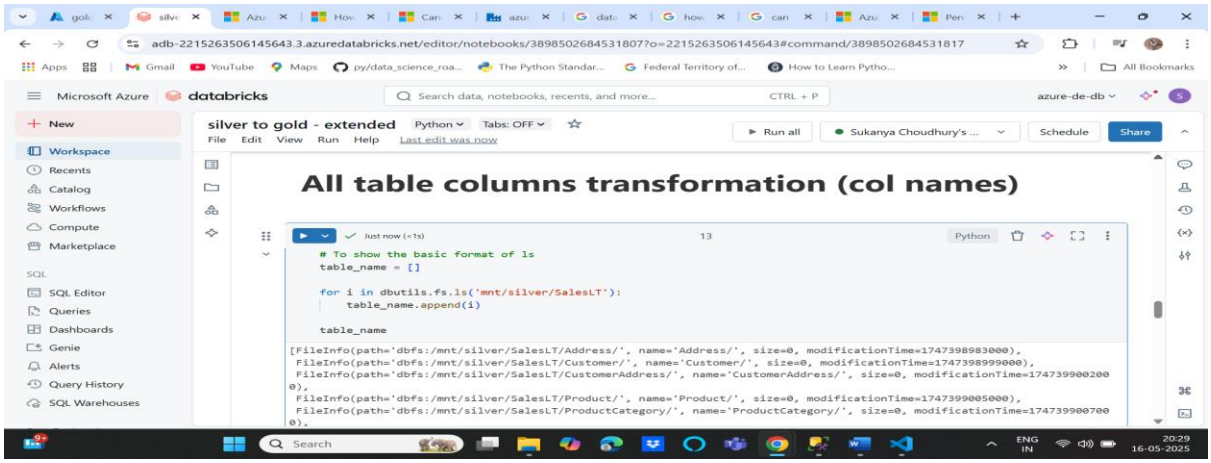


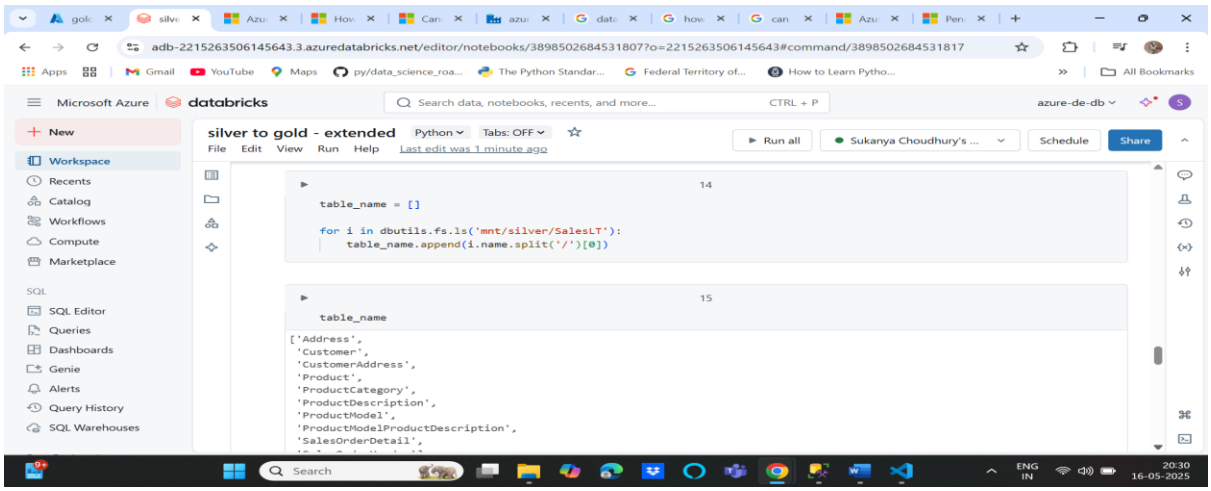

adb-2215263506145643.3.azuredatabricks.net/editor/notebooks/3898502684531807?o=2215263506145643#command/6965454032437290
Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
azure-de-db
New
Workspace
Recents
Catalog
Workflows
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help
Last edit was now
Run all
Sukanya Choudhury's ...
Schedule
Share
Convert column names to lower case with _ wherever name h
Markdown
08:23 PM (<1s)
8
from pyspark.sql.functions import col
def rename_columns_to_snake_case(df):
"""
Convert column names from PascalCase or camelCase to snake_case in a PySpark DataFrame.
Args:
df (DataFrame): The input DataFrame with columns to be renamed.
Returns:
DataFrame: A new DataFrame with column names converted to snake_case.
"""
# Get the list of column names
column_names = df.columns
Search
ENG
IN
20:28
16-05-2025

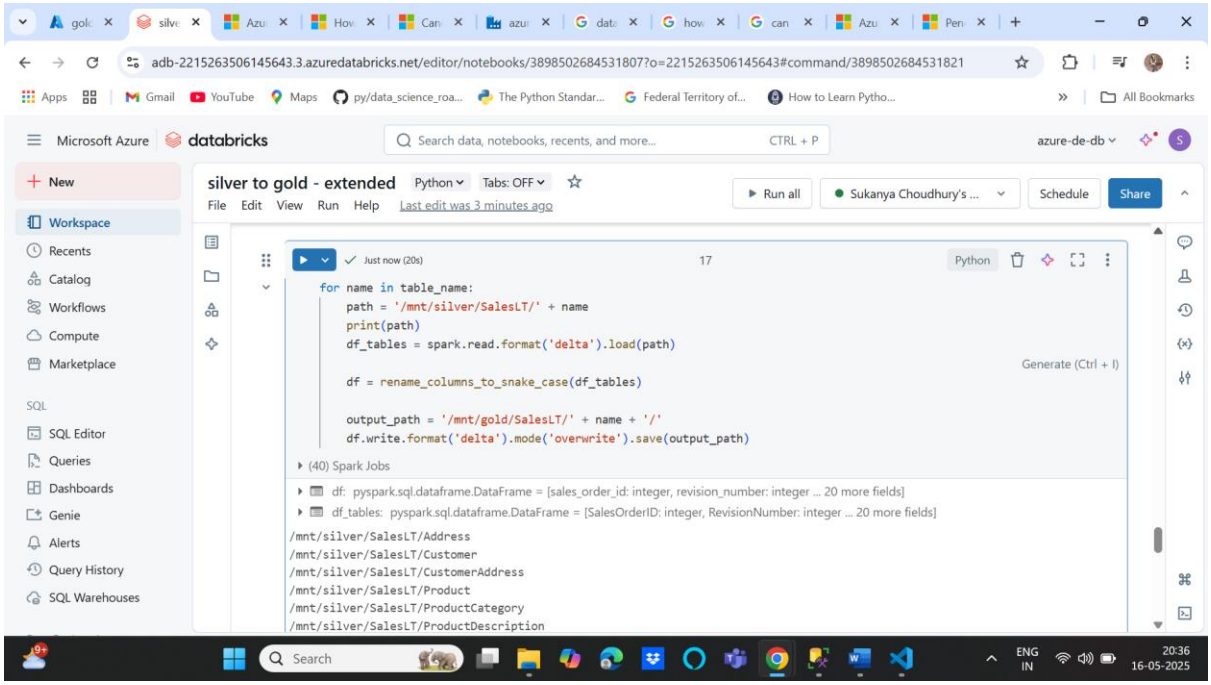

adb-2215263506145643.3.azuredatabricks.net/editor/notebooks/3898502684531807?o=2215263506145643#command/6965454032437290
Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
azure-de-db
New
Workspace
Recents
Catalog
Workflows
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help
Last edit was now
Run all
Sukanya Choudhury's ...
Schedule
Share
08:23 PM (<1s)
8
# Dictionary to hold old and new column name mappings
rename_map = {}
for old_col_name in column_names:
# Convert column name from PascalCase or camelCase to snake_case
new_col_name = "".join([
"_" + char.lower() if (
char.isupper() # Check if the current character is uppercase
and idx > 0 # Ensure it's not the first character
and not old_col_name[idx - 1].isupper() # Ensure the previous character is not uppercase
) else char.lower() # Convert character to lowercase
for idx, char in enumerate(old_col_name)
]).lstrip("_") # Remove any leading underscore
# Avoid renaming to an existing column name
if new_col_name in rename_map.values():
raise ValueError(f"Duplicate column name found after renaming: '{new_col_name}'")
# Map the old column name to the new column name
rename_map[old_col_name] = new_col_name
Search
ENG
IN
20:28
16-05-2025


adb-2215263506145643.3.azuredatabricks.net/editor/notebooks/3898502684531807?o=2215263506145643#command/6965454032437290
Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
azure-de-db
New
Workspace
Recents
Catalog
Workflows
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help Last edit was now
Run all
Sukanya Choudhury's ...
Schedule
Share
08:23 PM (<1s)
8
# Rename columns using the mapping
for old_col_name, new_col_name in rename_map.items():
df = df.withColumnRenamed(old_col_name, new_col_name)
return df
Example usage
df = rename_columns_to_snake_case(df)
4 minutes ago (<1s)
9
df = rename_columns_to_snake_case(dfAddress)
df: pyspark.sql.dataframe.DataFrame = [address_id: integer, address_line1: string ... 7 more fields]
Search
ENG
IN
20:28
16-05-2025


adb-2215263506145643.3.azuredatabricks.net/editor/notebooks/3898502684531807?o=2215263506145643#command/6965454032437291
Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
azure-de-db
New
Workspace
Recents
Catalog
Workflows
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help Last edit was now
Run all
Sukanya Choudhury's ...
Schedule
Share
4 minutes ago (<1s)
10
df = rename_columns_to_snake_case(dfAddress)
df: pyspark.sql.dataframe.DataFrame = [address_id: integer, address_line1: string ... 7 more fields]
4 minutes ago (<1s)
11
display(df)
(1) Spark Jobs
Table
address_id
address_line1
address_line2
city
state_province
country_regio
1 9 8713 Yosemite Ct. null Bothell Washington United States
2 11 1318 Lasalle Street null Bothell Washington United States
3 25 9178 Jumping St. null Dallas Texas United States
4 28 9228 Via Del Sol null Phoenix Arizona United States
Search
ENG
IN
20:29
16-05-2025

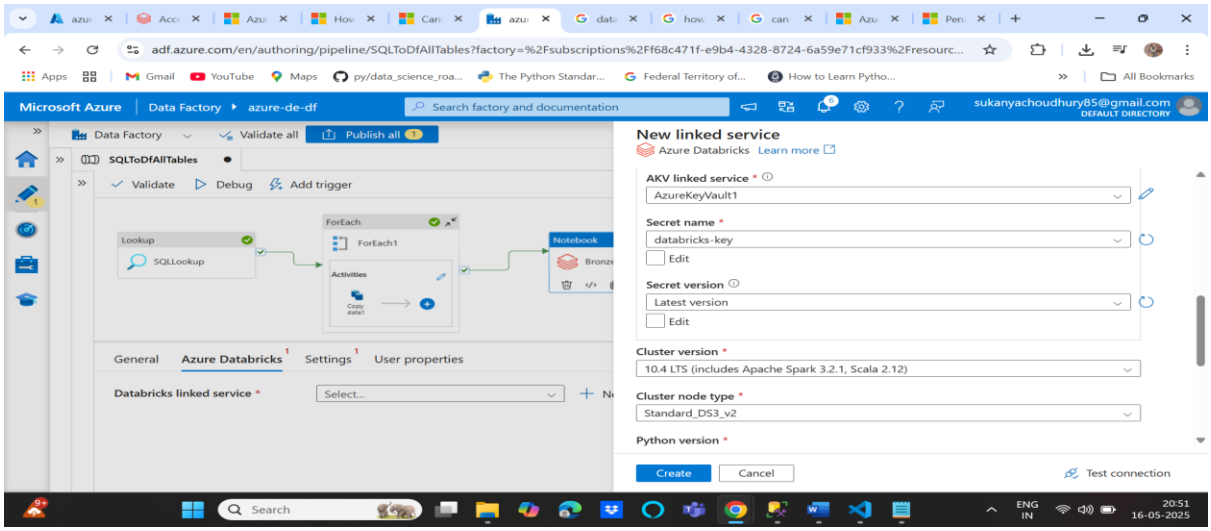

All table columns transformation (col names)
silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help Last edit was now
Run all
Sukanya Choudhury's ...
Schedule
Share
13
Python
# To show the basic format of ls
table_name = []
for i in dbutils.fs.ls('mnt/silver/SalesLT'):
table_name.append(i)
table_name
[FileInfo(path='dbfs:/mnt/silver/SalesLT/Address/', name='Address/', size=0, modificationTime=1747398983000),
FileInfo(path='dbfs:/mnt/silver/SalesLT/Customer/', name='Customer/', size=0, modificationTime=1747398999000),
FileInfo(path='dbfs:/mnt/silver/SalesLT/CustomerAddress/', name='CustomerAddress/', size=0, modificationTime=174739900200
0),
FileInfo(path='dbfs:/mnt/silver/SalesLT/Product/', name='Product/', size=0, modificationTime=1747399005000),
FileInfo(path='dbfs:/mnt/silver/SalesLT/ProductCategory/', name='ProductCategory/', size=0, modificationTime=17473990070
0),


silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help Last edit was 1 minute ago
Run all
Sukanya Choudhury's ...
Schedule
Share
14
table_name = []
for i in dbutils.fs.ls('mnt/silver/SalesLT'):
table_name.append(i.name.split('/')[0])
15
table_name
['Address',
'Customer',
'CustomerAddress',
'Product',
'ProductCategory',
'ProductDescription',
'ProductModel',
'ProductModelProductDescription',
'SalesOrderDetail',

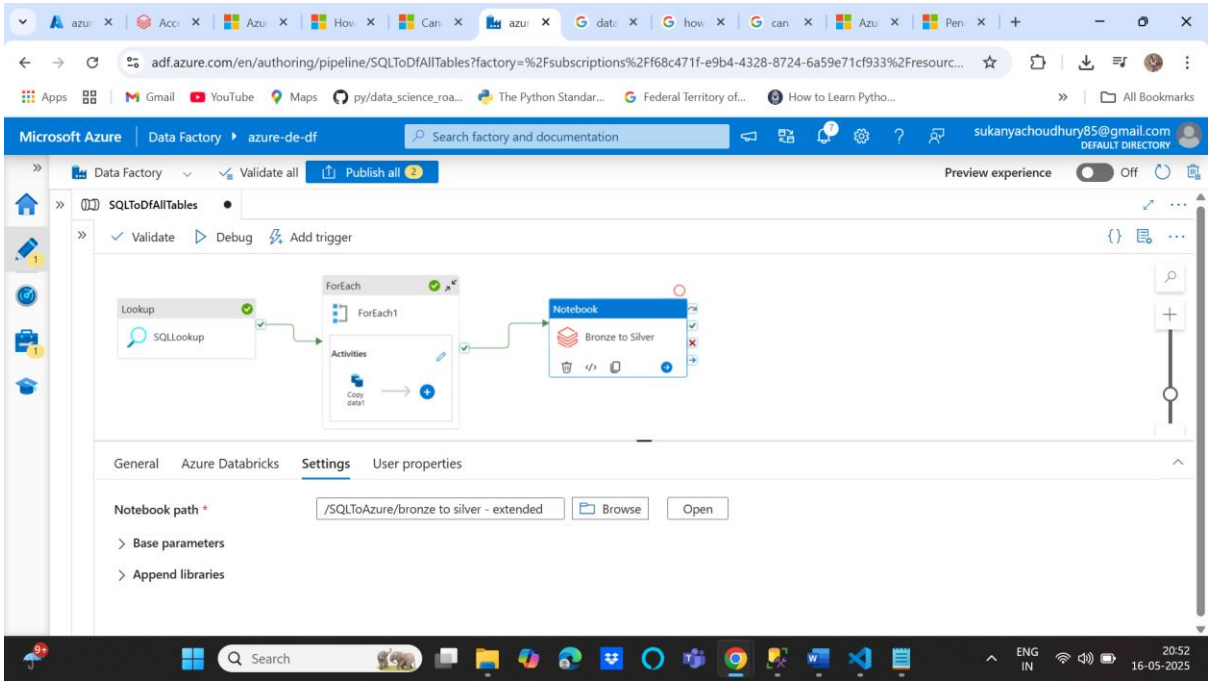

silver to gold - extended
Python
Tabs: OFF
File Edit View Run Help Last edit was 3 minutes ago
Run all
Sukanya Choudhury's ...
Schedule
Share
Just now (20s)
17
Python
for name in table_name:
path = '/mnt/silver/SalesLT/' + name
print(path)
df_tables = spark.read.format('delta').load(path)
Generate (Ctrl + I)
df = rename_columns_to_snake_case(df_tables)
output_path = '/mnt/gold/SalesLT/' + name + '/'
df.write.format('delta').mode('overwrite').save(output_path)
(40) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [sales_order_id: integer, revision_number: integer ... 20 more fields]
df_tables: pyspark.sql.dataframe.DataFrame = [SalesOrderID: integer, RevisionNumber: integer ... 20 more fields]
/mnt/silver/SalesLT/Address
/mnt/silver/SalesLT/Customer
/mnt/silver/SalesLT/CustomerAddress
/mnt/silver/SalesLT/Product
/mnt/silver/SalesLT/ProductCategory
/mnt/silver/SalesLT/ProductDescription

4) So now we want that these databricks notebooks run automatically from the pipeline so add notebook activity on success to the foreach activity of the pipeline

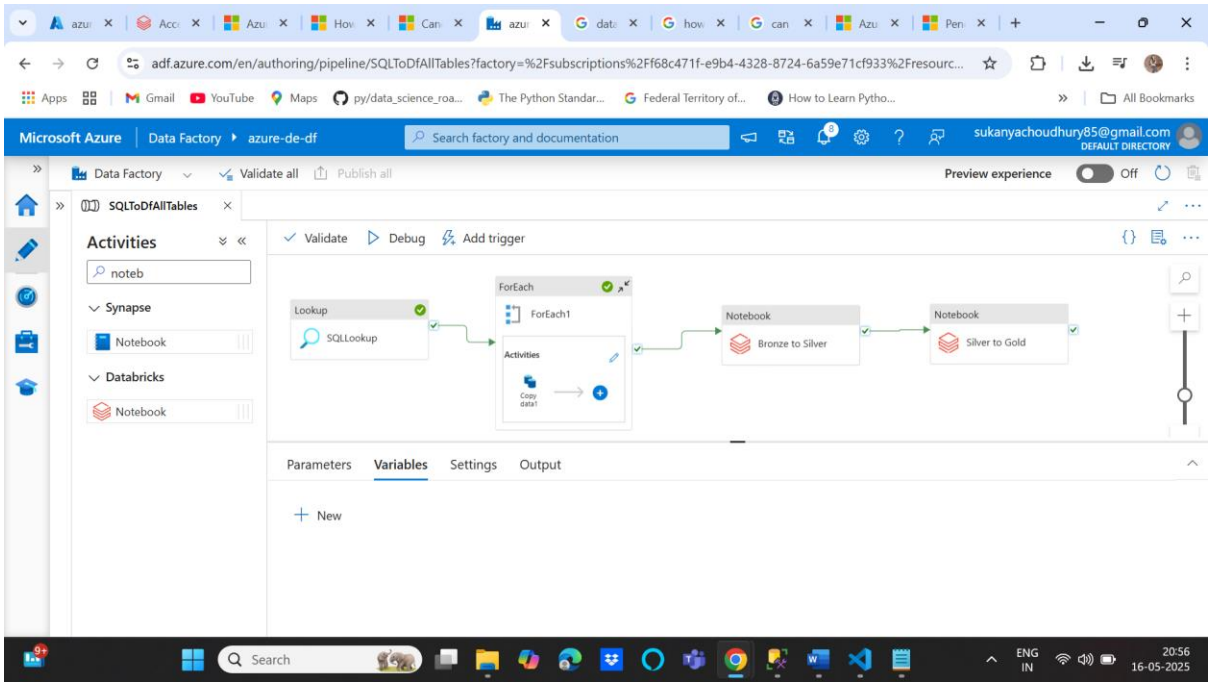
Create a databricks linked service for the notebook activity using access token from databricks settings→developer→access tokens→Generate access token and then add it on the linked service configuration or store it in key vault→secret→generate/import and then use it



Then Give the notebook path



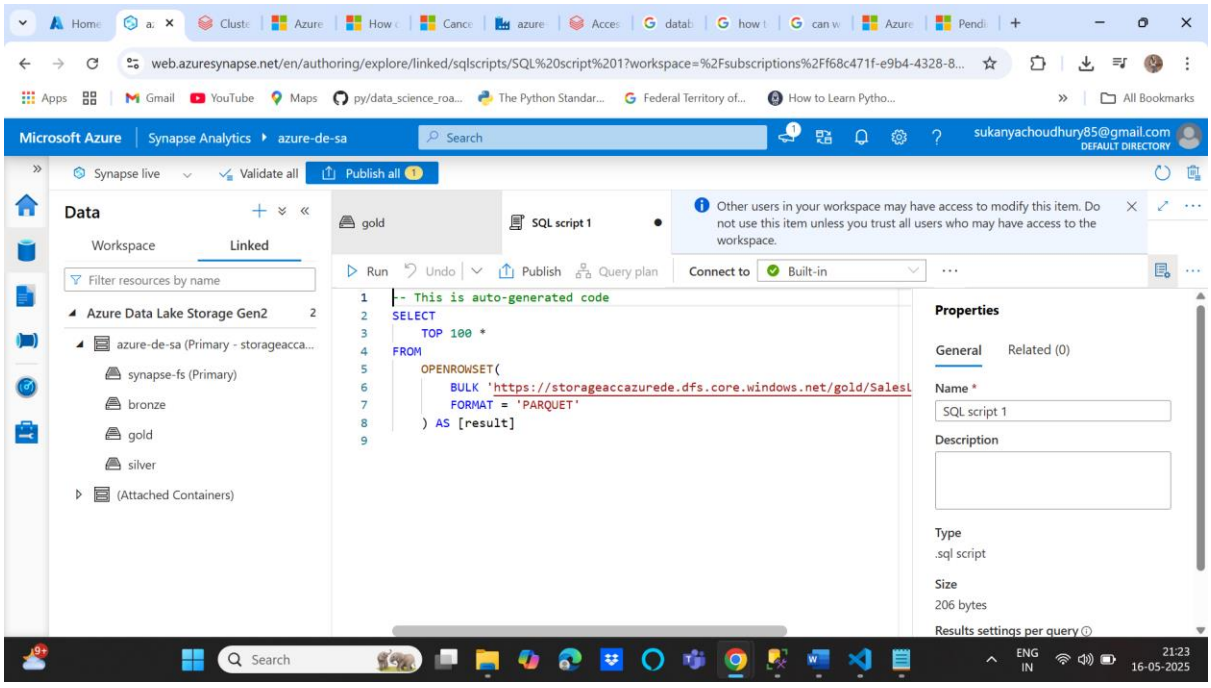
Similarly create another notebook activity for silver to gold and then publish and then add trigger to run the pipeline, monitor on pipeline runs



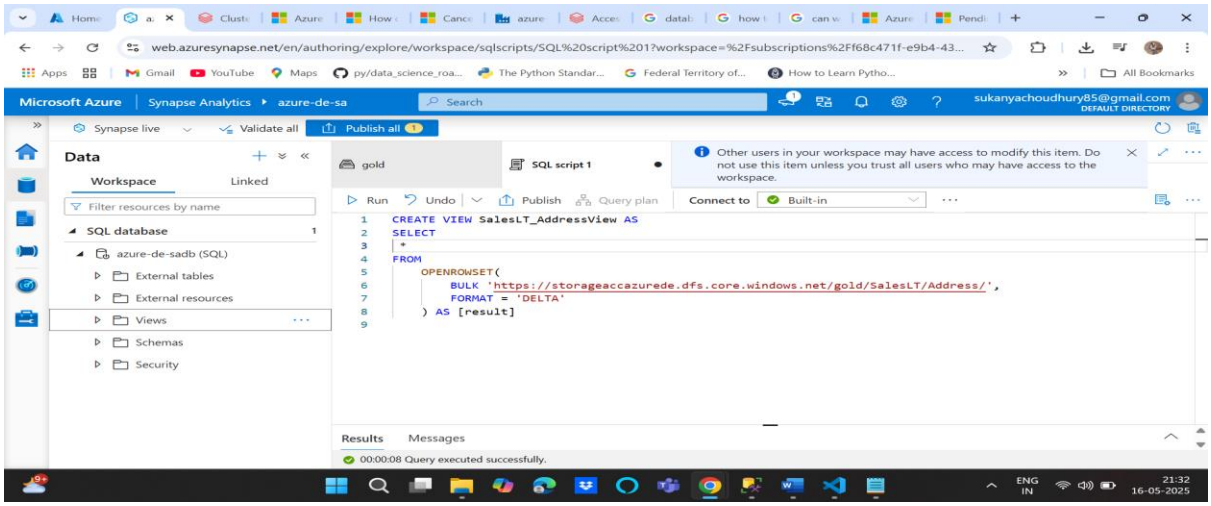
Step 4: Data Load Into Synapse Analytics

1) Synapse studio—data→add new SQL database

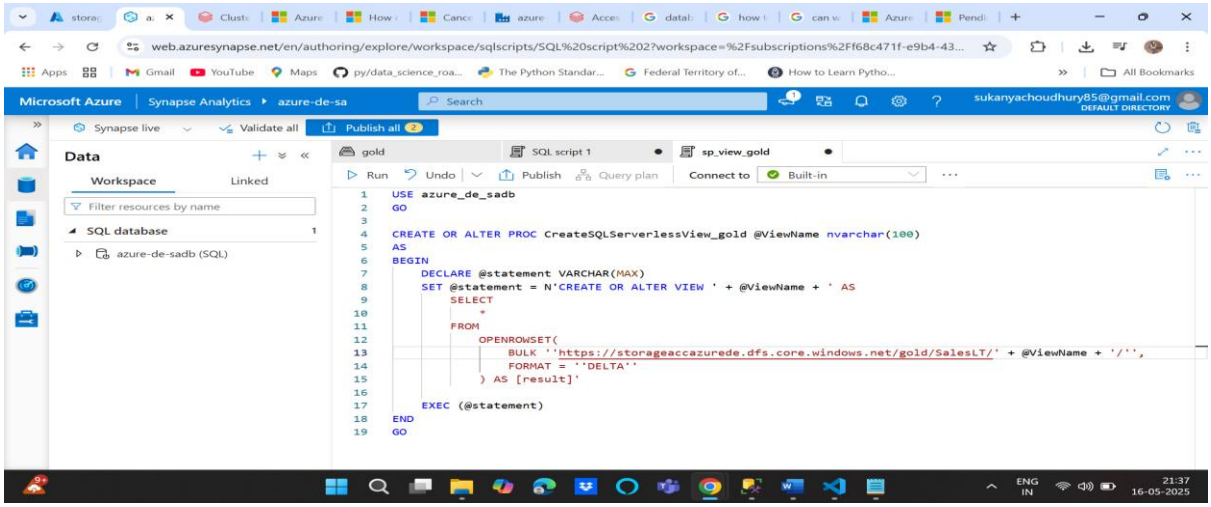
Instead of writing the full query to explore a table, go to linked storage lake→ navigate to gold folder and then for each table folder rt click do show 100 rows for delta format→ it opens an sql script which shows data for that table



Now we can use the same script to create view by adding a create view statement above as shown below, select appropriate db from use database dropdown

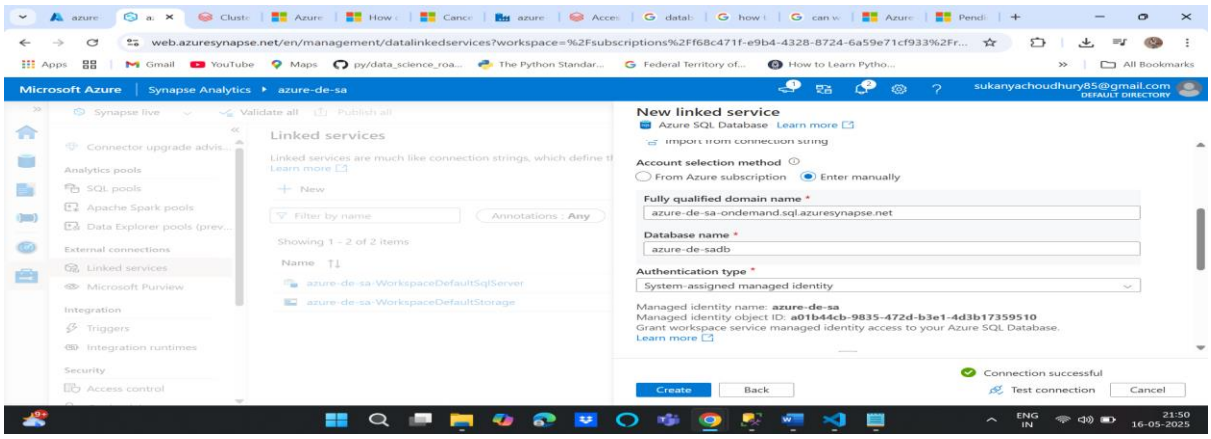


2) Create a sql procedure to load all tables together and create views and publish it



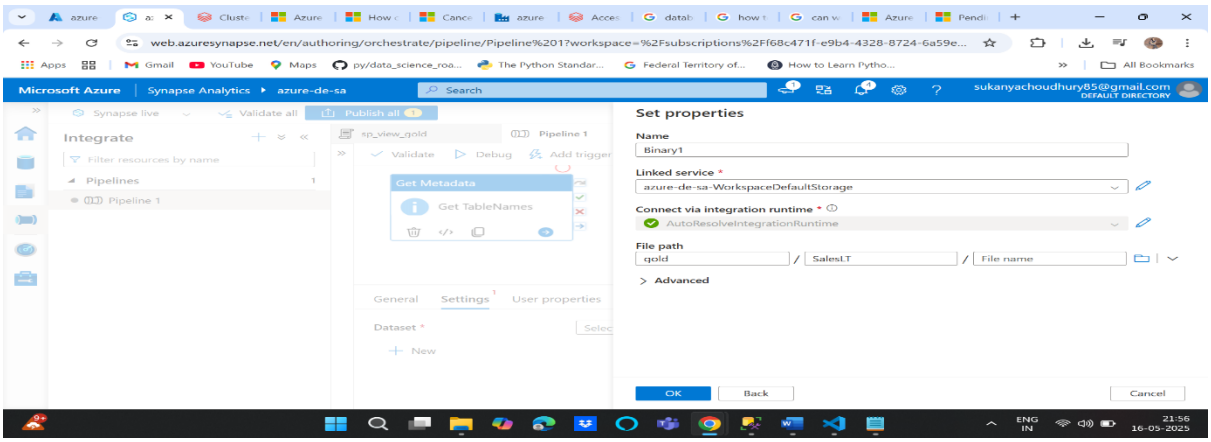
3) Create a pipeline inside synapsis analytics studio which uses this procedure

Go to manage→linkedServices-> +New→ Azure SQL Database→ give the serverless sql endpoint of sa as fully qualified domain name→ create and publish linked service



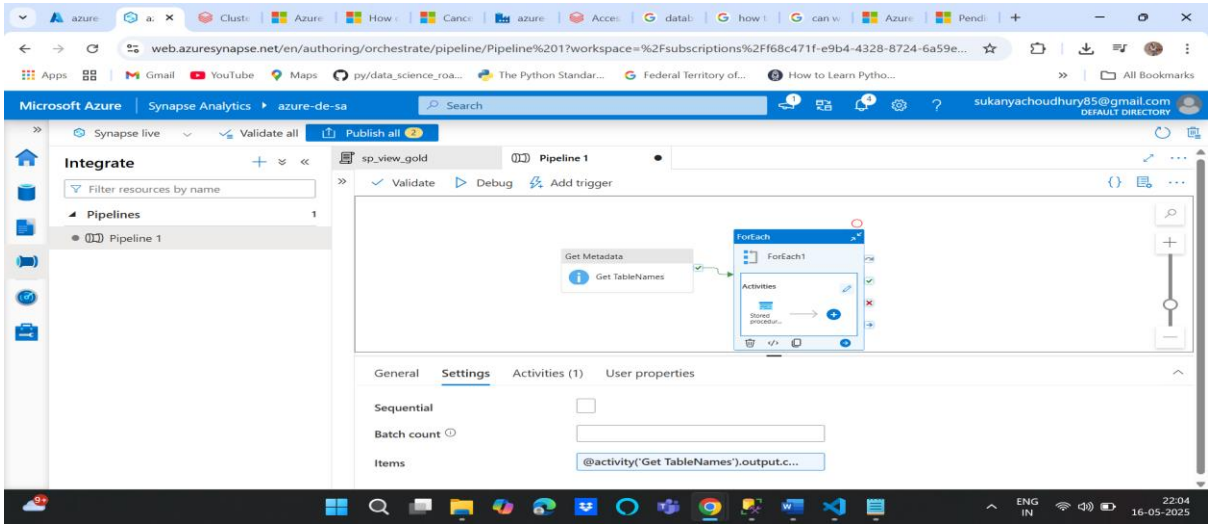
Now go to integrate→+→ Pipeline→ Add Get Metadata activity→ settings—add dataset→azure data lake gen2→binary format→ select the default linked service and the gold directory

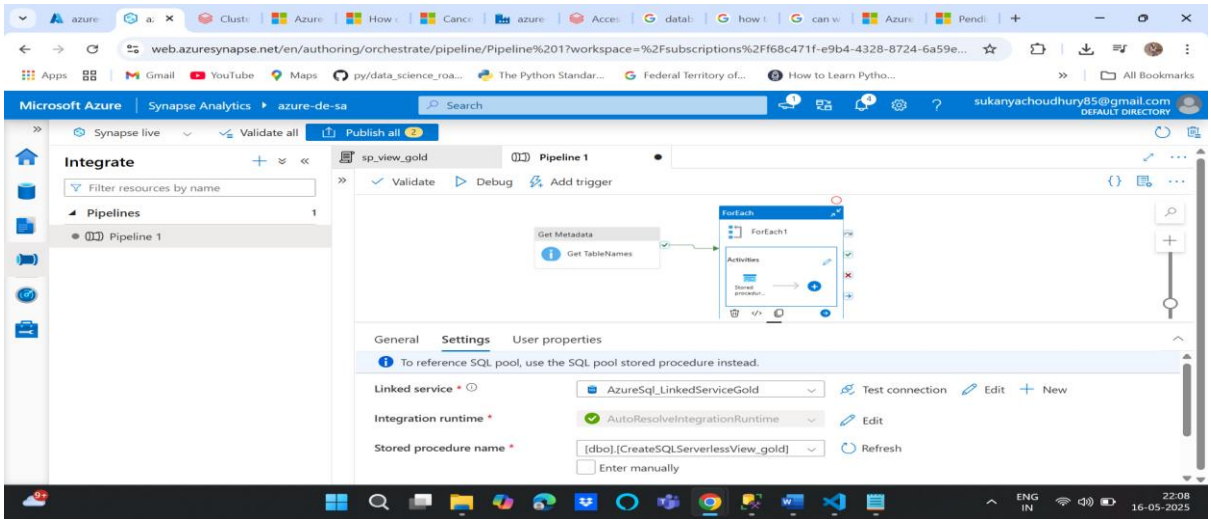
And then Getmetadata activity→ field list→ new→ Child items

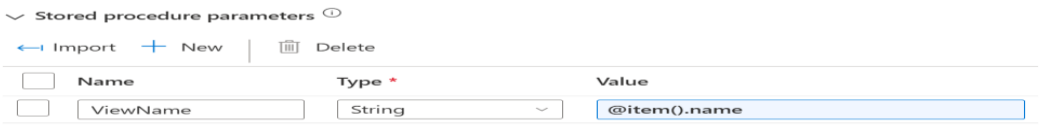


Now add foreach activity to the pipeline on success of the getmetadata activity

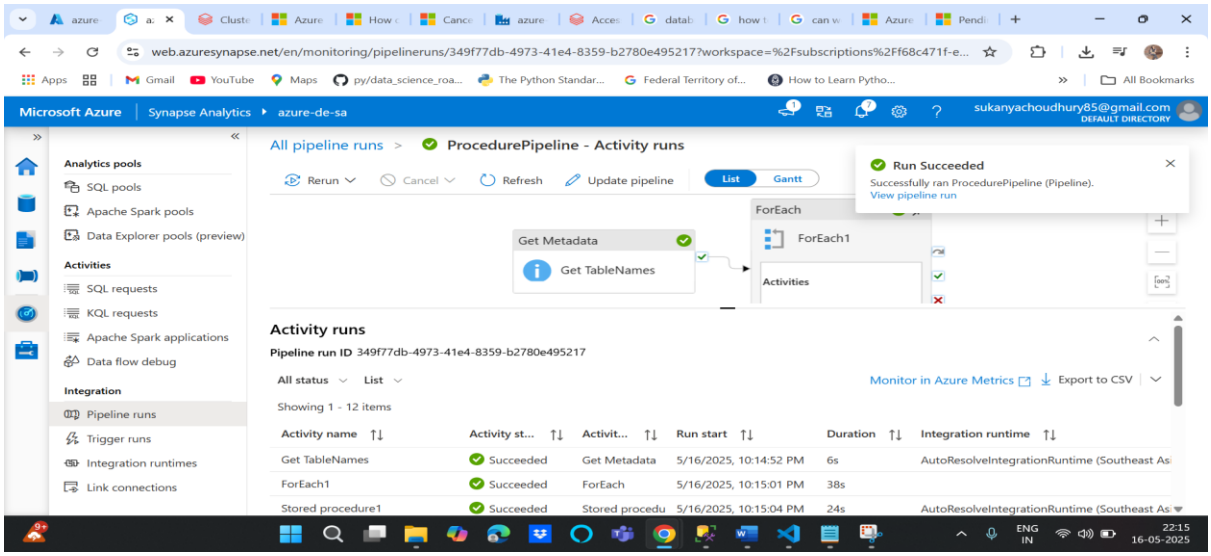
Settings→ items→Add dynamic content→GetTable.output.childitems and then add stored procedure to the foreach activity→give the linked service and the procedure to it and parameters value pass ass @item().name i.e view name parameter = tablename



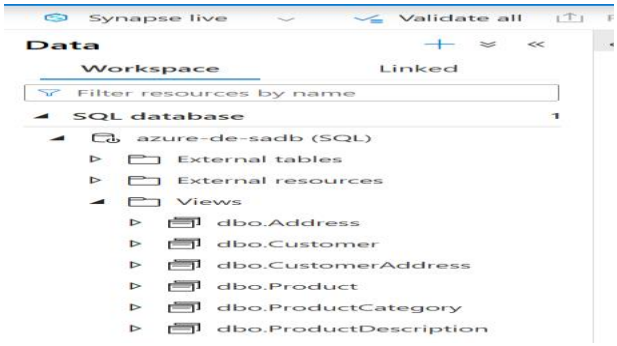




Run the pipeline

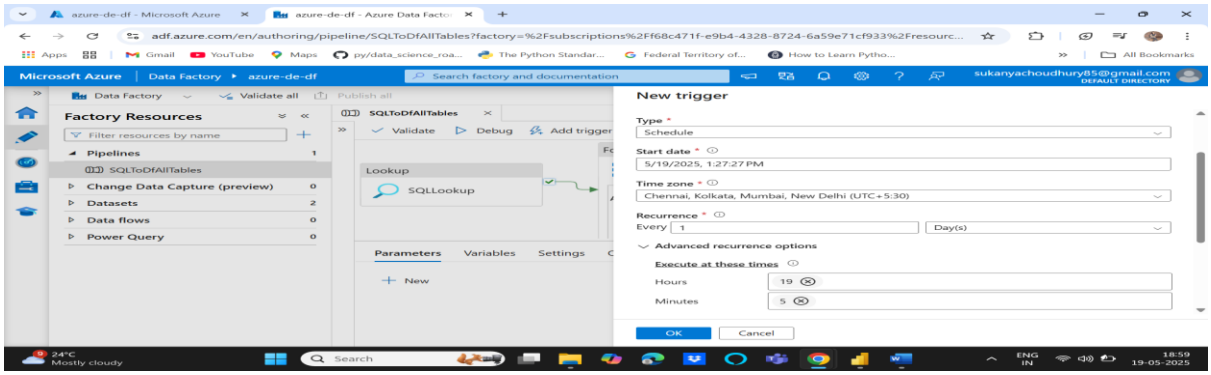


All views are created



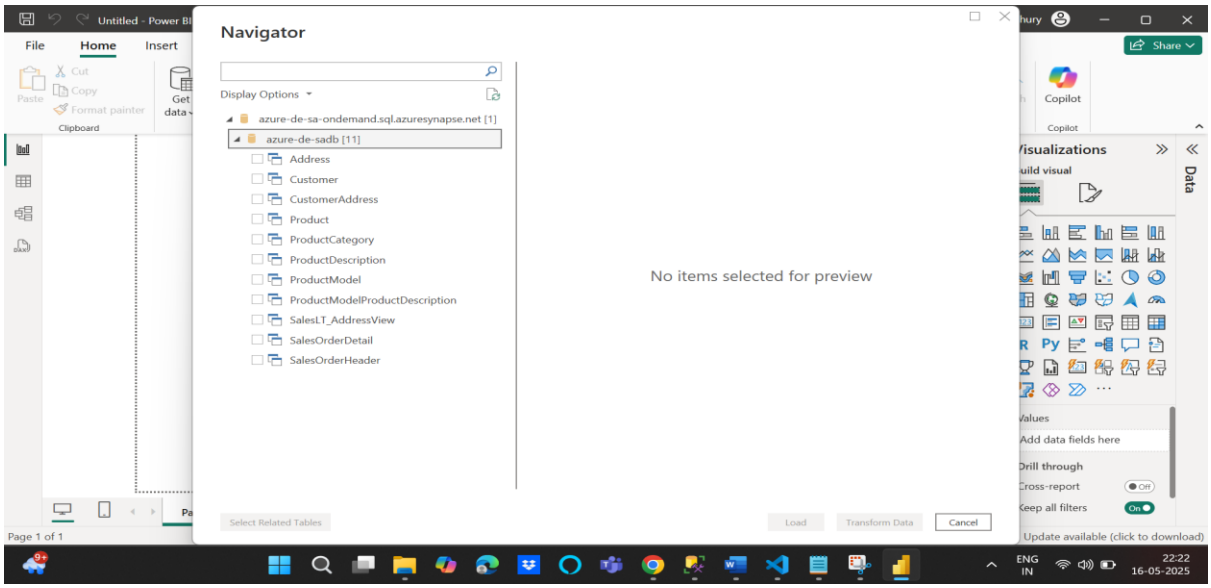
4) Trigger pipeline on a schedule:

Launch Data Factory→Go to the pipeline→ Add Trigger→ New→



Step 5: Reporting using Power BI

1) Get data→Azure Synapse analytics SQL→ give the synapse analytics serverless endpoint→ username/password and connect→ you can see all the views like below



Create a dashboard like below