

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:

Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Resource group *

Instance details
Name * Region * Version *

Previous Give feedback

Your deployment is complete

Deployment name : Microsoft.DataFactory-20250514215217... Start time : 5/14/2025, 9:53:51 PM
Subscription : Azure subscription 1 Correlation ID : e03a5a68-e20a-4788-a431-bfd...
Resource group : azure-de-rg

Deployment details

Next steps

Give feedback

Tell us about your experience with deployment

Cost management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >

Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials
Start learning today >

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

Subscription * Resource group *

Storage account name * Region * Deploy to an Azure Extended Zone

Primary service

Performance * Standard: Recommended for most scenarios (general-purpose v2 account)
 Premium: Recommended for scenarios that require low latency.

Previous Give feedback

The screenshot shows the 'Create a storage account' wizard in the Microsoft Azure portal. The 'Hierarchical Namespace' section is active, with the checkbox 'Enable hierarchical namespace' checked. Other sections like 'Access protocols' and 'Enable SFTP' are also present. The status bar at the bottom right shows the date as 14-05-2025.

The screenshot shows the deployment overview for 'storageaccazuredede_1747240163637'. It indicates that the deployment is complete. Deployment details include the name, start time, and correlation ID. The status bar at the bottom right shows the date as 14-05-2025.

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

The screenshot shows the 'Containers' blade for the storage account 'storageaccazuredede'. It lists five containers: '\$logs', 'bronze', 'gold', 'silver', and 'synapse-fs'. A success message box is visible in the top right corner, stating 'Successfully created storage container' and 'Successfully created storage container 'gold''. The status bar at the bottom right shows the date as 15-05-2025.

5) Create Databricks Workspace, "azure-de-db"

The screenshot shows the 'Create an Azure Databricks workspace' wizard. In the 'Subscription' dropdown, 'Azure subscription 1' is selected. Under 'Resource group', 'azure-de-rg' is chosen. The 'Workspace name' field contains 'azure-de-db'. The 'Region' is set to 'Southeast Asia' and the 'Pricing Tier' is 'Premium (+ Role-based access controls)'. A tooltip indicates that the selected tier is the recommended one. At the bottom, there are 'Review + create', '< Previous', and 'Next : Networking >' buttons.

The second part of the screenshot shows the 'Overview' page for the newly created workspace 'azure-de-rg.azure-de-db'. It displays deployment details: Deployment name: 'azure-de-rg.azure-de-db', Subscription: 'Azure subscription 1', Resource group: 'azure-de-rg', Start time: 5/14/2025, 10:03:26 PM, Correlation ID: 'db917e39-db48-4770-9ad9-9c61959d05f3'. The 'Deployment succeeded' status is shown in the notifications panel, along with other successful deployment logs.

6) Create Synapse analytics workspace, "azure-de-sa"

The screenshot shows the 'Create Synapse workspace' wizard. The 'Workspace name' is 'azure-de-sa', 'Region' is 'Southeast Asia', and 'Select Data Lake Storage Gen2' is set to 'From subscription'. The 'Account name' is 'storageaccazurede' and the 'File system name' is '(New) synapse-fs'. A checkbox for assigning the 'Storage Blob Data Contributor' role is checked. Navigation buttons at the bottom include 'Review + create', '< Previous', and 'Next: Security >'.

Choose the authentication method for access to workspace resources such as SQL pools. The authentication method can be changed later on. [Learn more](#)

Authentication method Use both local and Microsoft Entra ID authentication Use only Microsoft Entra ID authentication

SQL Server admin login *

SQL Password

Confirm password

System assigned managed identity permission

Select to grant the workspace network access to the Data Lake Storage Gen2 account using the workspace system identity. [Learn more](#)

[Review + create](#) [< Previous](#) [Next: Networking >](#)

Deployment

Overview

Your deployment is complete

Deployment name : Microsoft.Azure.SynapseAnalytics-20250514221548 Start time : 5/14/2025, 10:17:15 PM
Subscription : Azure subscription 1 Correlation ID : 05c51d84-4870-4348-8923-2b...
Resource group : azure-de-rg

Deployment details

Next steps

Go to resource group

Give feedback

Tell us about your experience with deployment

Add or remove favorites by pressing **Ctrl+Shift+F**

[Cost management](#)
Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

[Microsoft Defender for Cloud](#)
Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

[Free Microsoft tutorials](#)
Start learning today >

7) Create a Key Vault

Create a key vault

Subscription *

Resource group * [Create new](#)

Instance details

Key vault name *

Region *

Pricing tier *

Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

[Previous](#) [Next](#) [Review + create](#) [Give feedback](#)

Create a key vault

Review + create

Basics

Subscription	Azure subscription 1
Resource group	azimuth-de-rg
Key vault name	azimuth-de-kv
Region	Southeast Asia
Pricing tier	Standard
Soft-delete	Enabled
Purge protection during retention period	Disabled
Days to retain deleted vaults	90 days

Previous **Next** **Create** **Give feedback**

azimuth-de-kv | Overview

Your deployment is complete

Deployment name : azimuth-de-kv
Subscription : Azure subscription 1
Resource group : azimuth-de-rg

Start time : 5/15/2025, 3:08:08 PM
Correlation ID : eb6c0ac7-8518-415b-881d-9e6...

Deployment details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

Cost management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >

Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials
Start learning today >

30°C Mostly sunny **Search** **ENG IN** **15-05-2025**

Resource group elements now are:

Home > azimuth-de-rg

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Cost Management

Monitoring

Automation

Search **Create** **Manage view** **Delete resource group** **Refresh** **Export to CSV** **Open query** **Assign tags** **Move**

Filter for any field... **Type equals all** **Location equals all** **Add filter**

Showing 1 to 5 of 5 records. **Show hidden types**

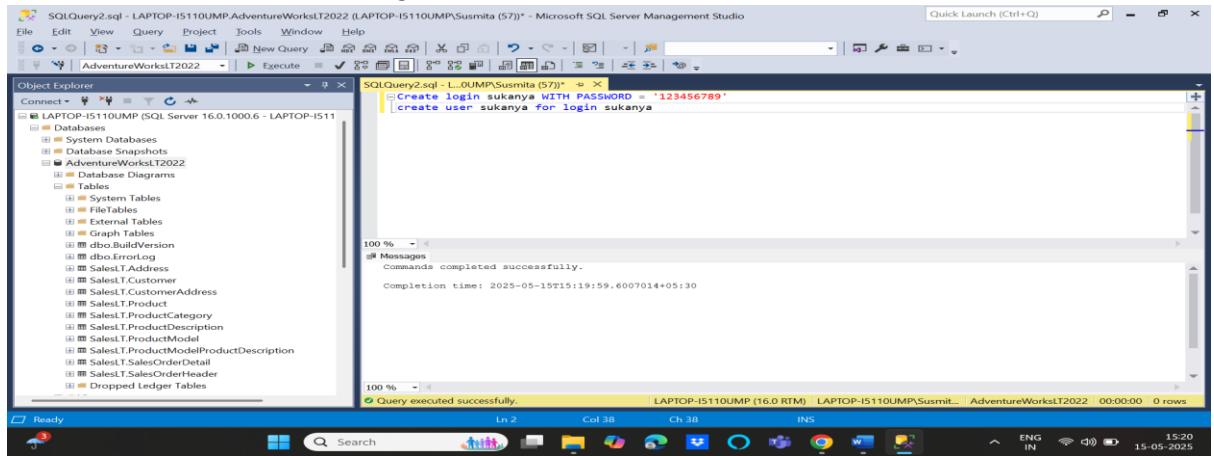
Name	Type	Location
azure-de-db	Azure Databricks Service	Southeast Asia
azure-de-df	Data factory (V2)	Southeast Asia
azure-de-kv	Key vault	Southeast Asia
azure-de-sa	Synapse workspace	Southeast Asia
storageaccazurede	Storage account	Southeast Asia

No grouping **List view**

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.

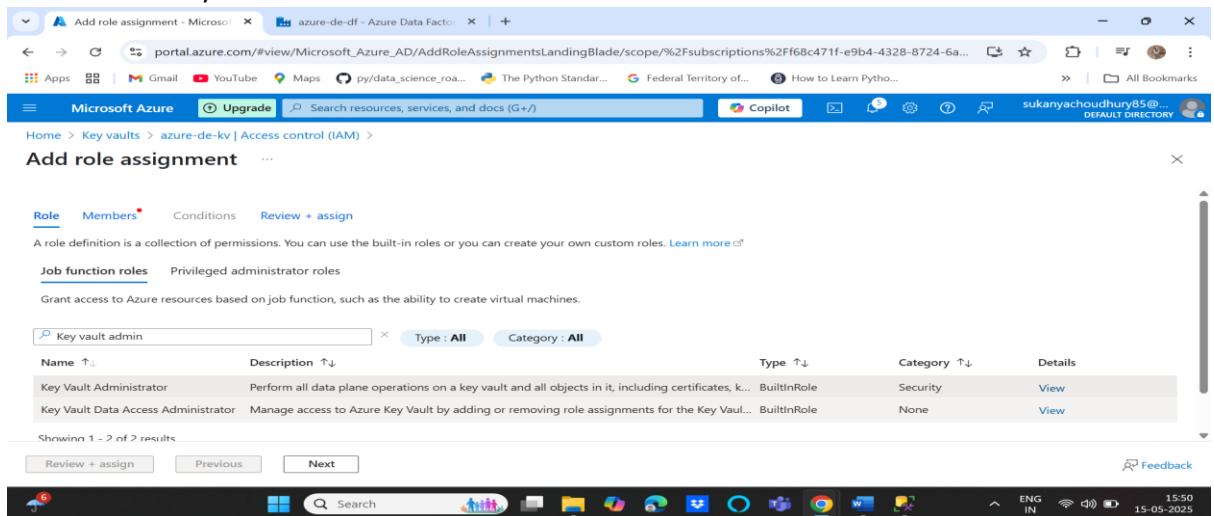


The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'AdventureworksLT2022' is selected. In the center pane, a query window titled 'SQLQuery2.sql' contains the following T-SQL code:

```
CREATE LOGIN sukanya WITH PASSWORD = '123456789';
CREATE USER sukanya FOR LOGIN sukanya;
```

The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2025-05-15T15:19:59.6007014+05:30.

- 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



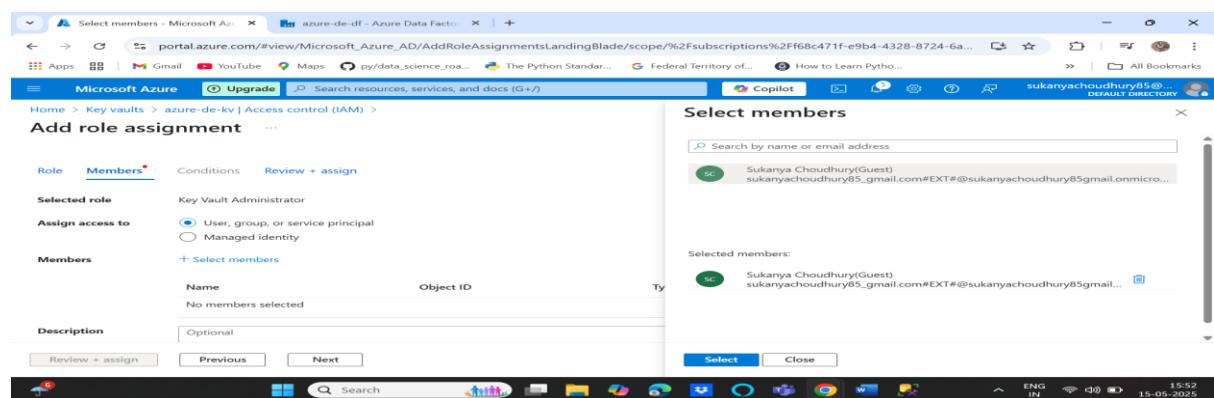
The screenshot shows the Azure portal with the URL 'portal.azure.com/#view/Microsoft_Azure_AD/AddRoleAssignmentsLandingBlade/scope/%2Fsubscriptions%2F68c471f-e9b4-4328-8724-6a...'. The page title is 'Microsoft Azure | Upgrade | Add role assignment - Microsoft Azure'.

The 'Members' tab is selected. The 'Job function roles' section shows 'Privileged administrator roles' assigned to 'Key vault admin'. Below this, it says 'Grant access to Azure resources based on job function, such as the ability to create virtual machines.'

A table lists two roles:

Name	Description	Type	Category	Details
Key Vault Administrator	Perform all data plane operations on a key vault and all objects in it, including certificates, k...	BuiltInRole	Security	View
Key Vault Data Access Administrator	Manage access to Azure Key Vault by adding or removing role assignments for the Key Vault...	BuiltInRole	None	View

At the bottom, there are 'Review + assign', 'Previous', and 'Next' buttons.



The screenshot shows the 'Select members' step in the Azure portal. The URL is 'portal.azure.com/#view/Microsoft_Azure_AD/AddRoleAssignmentsLandingBlade/scope/%2Fsubscriptions%2F68c471f-e9b4-4328-8724-6a...'. The page title is 'Microsoft Azure | Upgrade | Select members - Microsoft Azure'.

The 'Members' tab is selected. Under 'Selected role', 'Key Vault Administrator' is chosen. Under 'Assign access to', 'User, group, or service principal' is selected. The 'Members' section shows 'No members selected'. A search bar at the top right shows 'Sukanya Choudhury(Guest)'.

On the right, a 'Selected members' panel shows the same entry: 'Sukanya Choudhury(Guest)'.

- 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

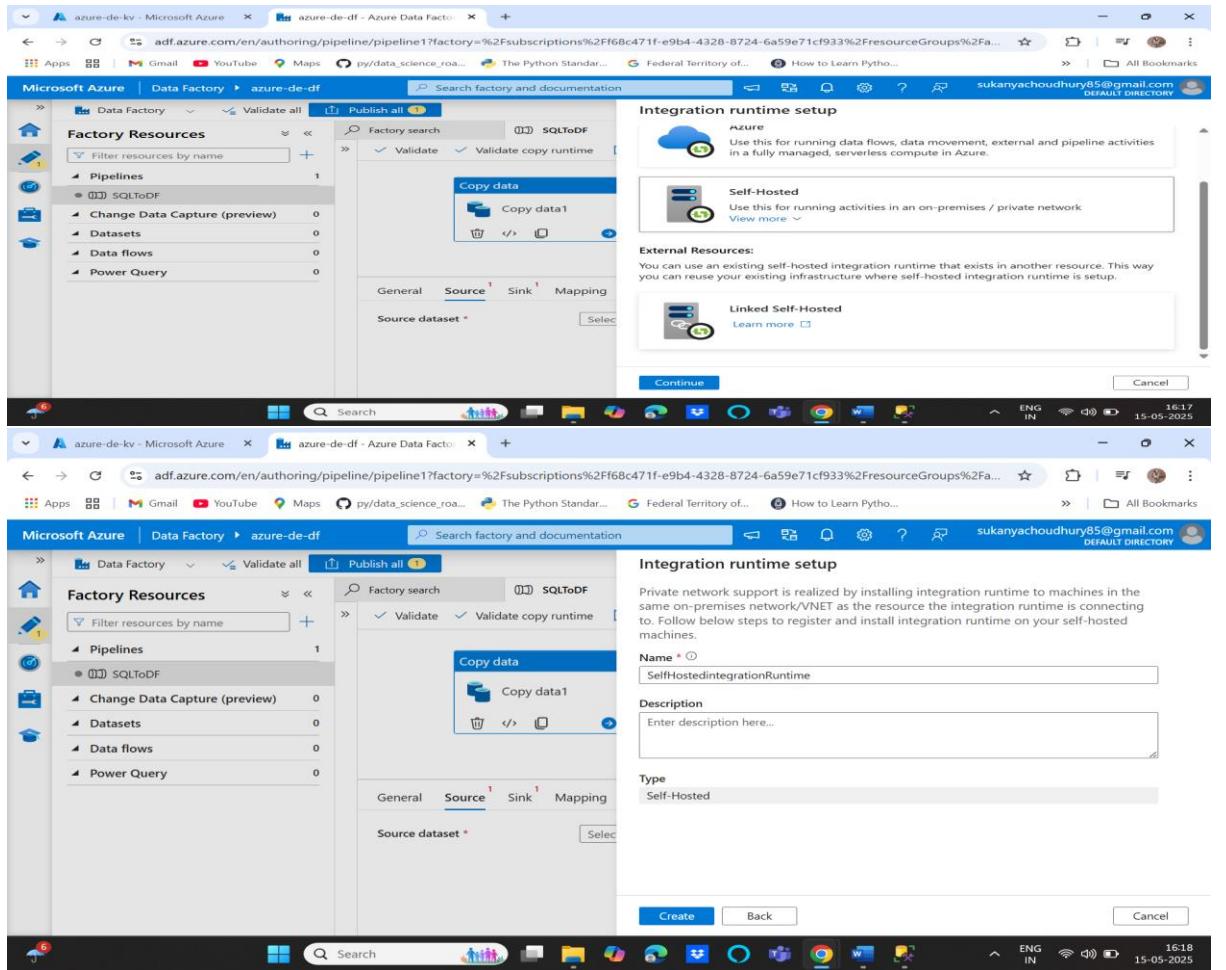
- **Load the data into Azure Data Lake Storage (ADLS) using Azure Data Factory (ADF).**

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

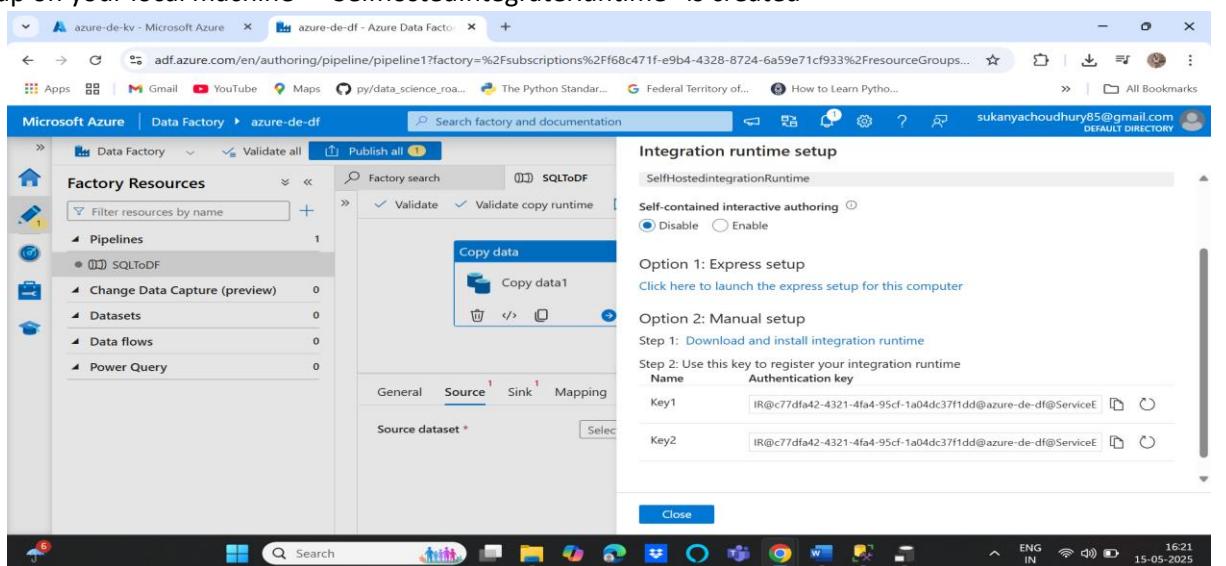
2) Configure source and sink by configuring respective linked services and source dataset

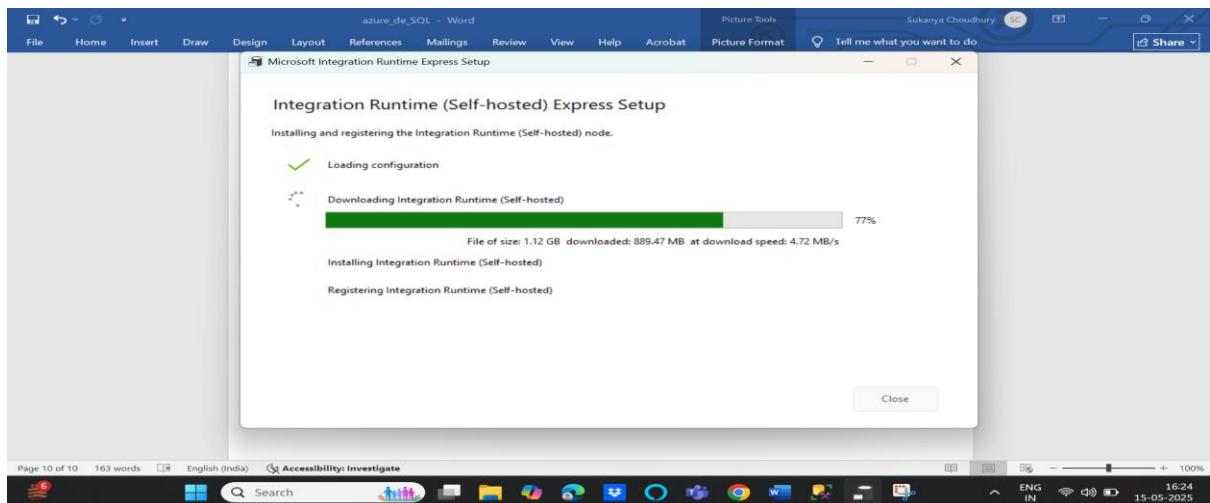
Go to Copydata → source → source dataset → + → SQL server (as our data is in On premise SQL server) → name

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

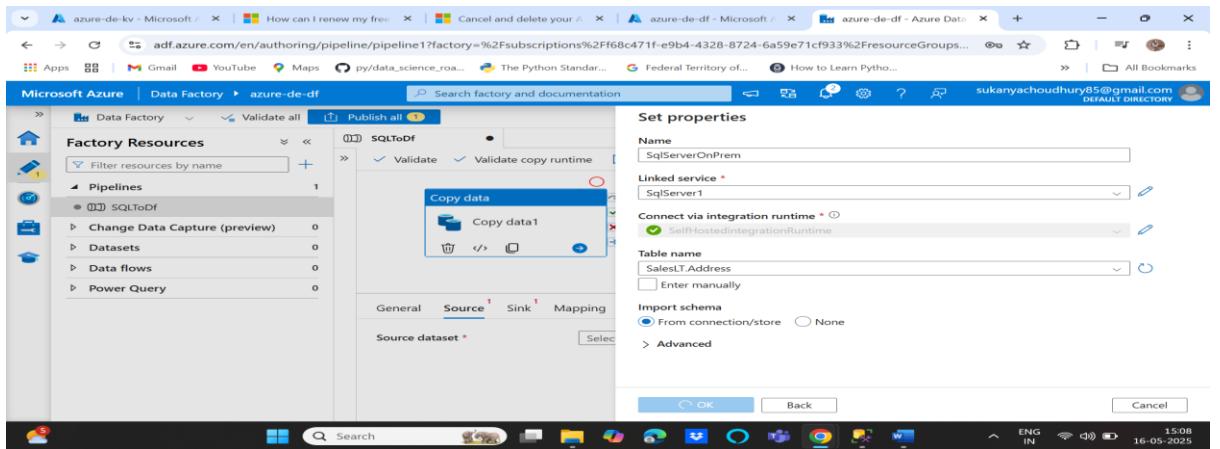


- 4) After creating→ Express Setup→ Click here to launch the express setup→ download and run the setup on your local machine—“SelfhostedIntegrateRuntime” is created



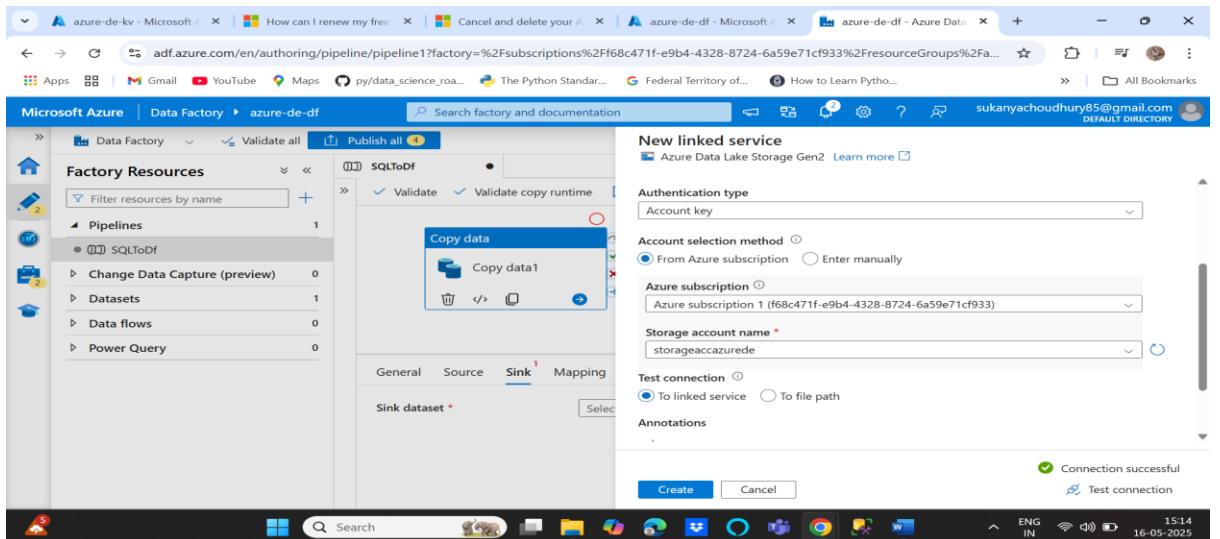


- 5) Use the above created Integrated Runtime for the linked service setup and give the database details and in password → Azure key vault → create a AKV linked service and then use the linked service and the secret key password created before so that the final sql serve linked service looks like below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

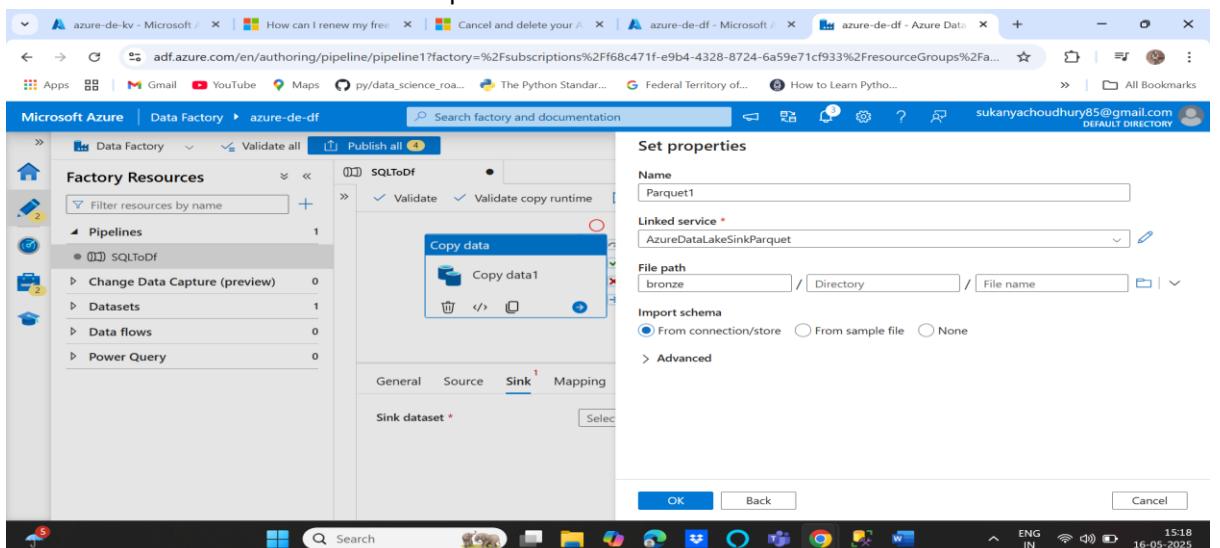


- 6) For sink again set up the data source using azure data lake gen 2 and file format parquet and also create a new linked service for the data source

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.

The screenshot shows the Azure Data Factory Pipeline Editor. A pipeline named "SQLToDF" is open. Inside, there is a single activity named "Copy data1". The "Output" tab is selected, showing a table with one item: "Copy data1" with status "Succeeded". The pipeline has 1 step, 0 errors, and 0 warnings.

All status	Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime
	Copy data1	Succeeded	Copy data	5/16/2025, 3:19:29 PM	21s	SelfHostedIntegrationRur

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

The screenshot shows the Azure Storage Explorer. A container named "bronze" is selected. Inside, there is a single blob named "SalesLT.Address.p...". The blob was last modified on 5/16/2025, 3:19:48 PM, is a Block blob, and its size is 35.08 KB.

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

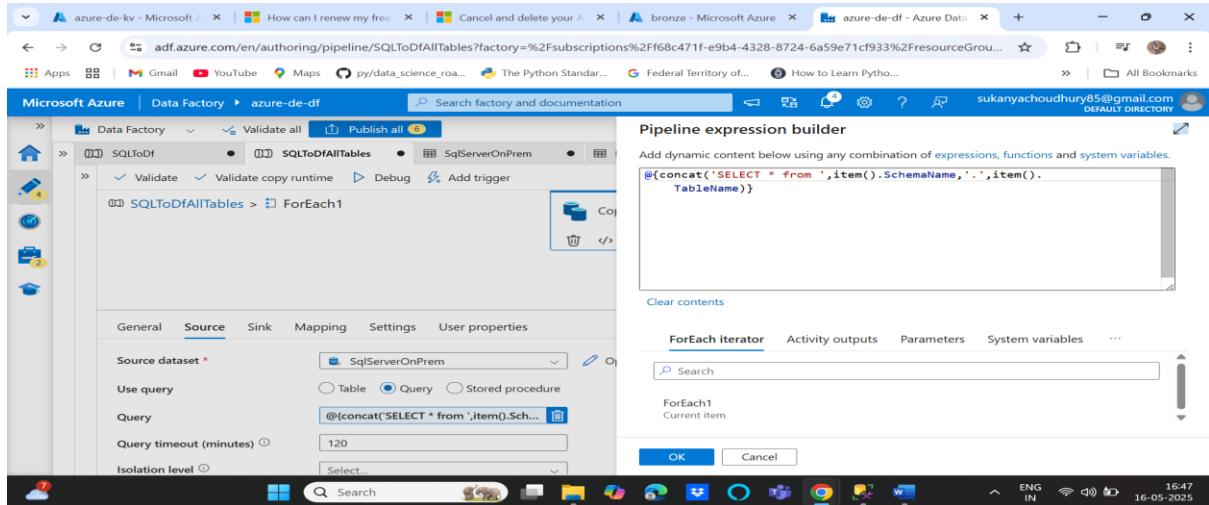
Configure lookup activity by setting up the source dataset or using the one created in the last step “SQLServerOnPrem”. Then open and edit it and remove the selected table as we need to use all the tables. Then in the lookup activity use query as below and uncheck “First row only” then debug to see all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

The screenshot shows the Azure Data Factory Pipeline Editor. A pipeline named "SQLToDfAllTables" is open. It contains a "Lookup" activity which is part of a "For Each" loop. The "Source dataset" is set to "SqlServerOnPrem". The "Query" for the lookup is:

```
SELECT s.name AS SchemaName,
       t.name AS TableName
  FROM sys.tables t
  JOIN sys.schemas s
    ON t.schema_id = s.schema_id
 WHERE s.name = 'SalesLT'
```

- 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

Name	Type	Default value
schemaname	String	Value
tablename	String	Value

Name	Value	Type
schemaname	@item().SchemaName	string
tablename	@item().TableName	string

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

The screenshot shows the Microsoft Azure Data Factory interface. In the center, there's a 'Pipeline expression builder' window. The 'Parameters' tab is selected, showing a search bar and a single parameter named 'schemaname'. Below it is a code editor containing the expression: `@{concat(dataset().schemaname, '/', dataset().tablename)}`. To the left of the builder, there's a configuration panel for a 'Parquet1' dataset, which includes tabs for 'Connection', 'Schema', and 'Parameters'. Under 'Parameters', the 'File path' field contains the expression `bronze / @{concat(dataset().schemaname, '/', dataset().tablename)}`. The 'Compression type' is set to 'snappy'. At the bottom right of the builder window are 'OK' and 'Cancel' buttons.

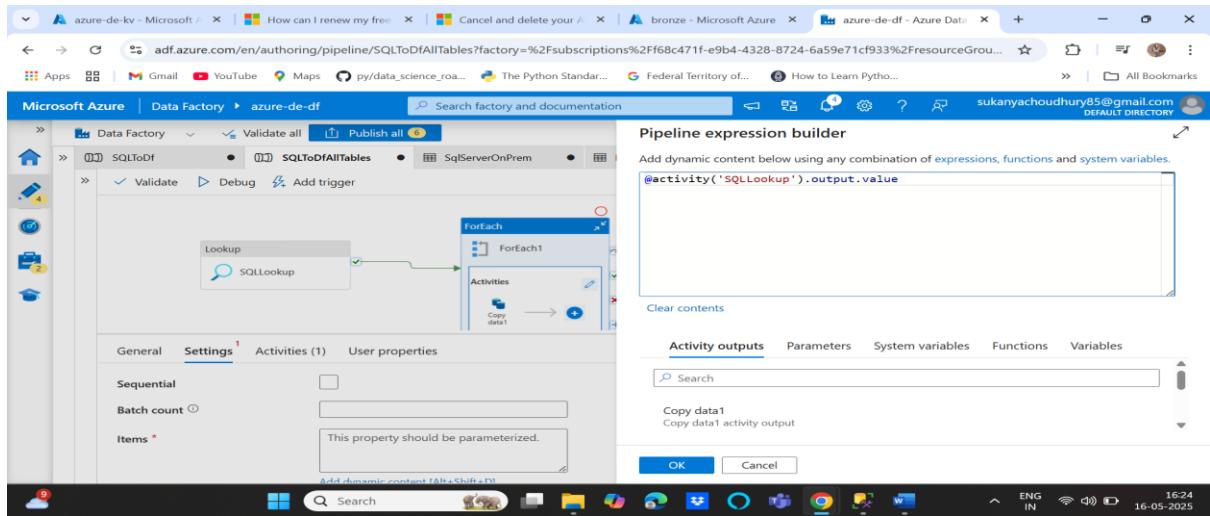
And the filename to save as parquet file as below

This screenshot is similar to the previous one but shows a different dynamic content expression. The code editor in the pipeline builder now contains the expression: `@{concat(dataset().tablename, '.parquet')}`. The rest of the interface, including the dataset configuration on the left, remains the same.

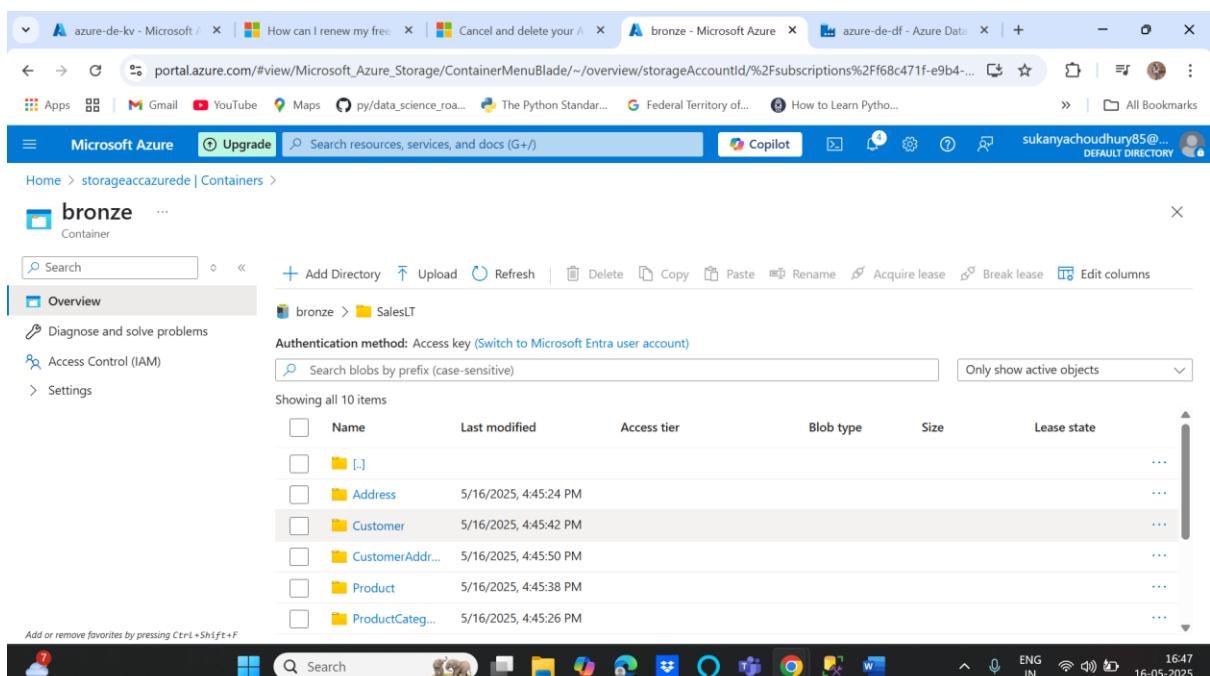
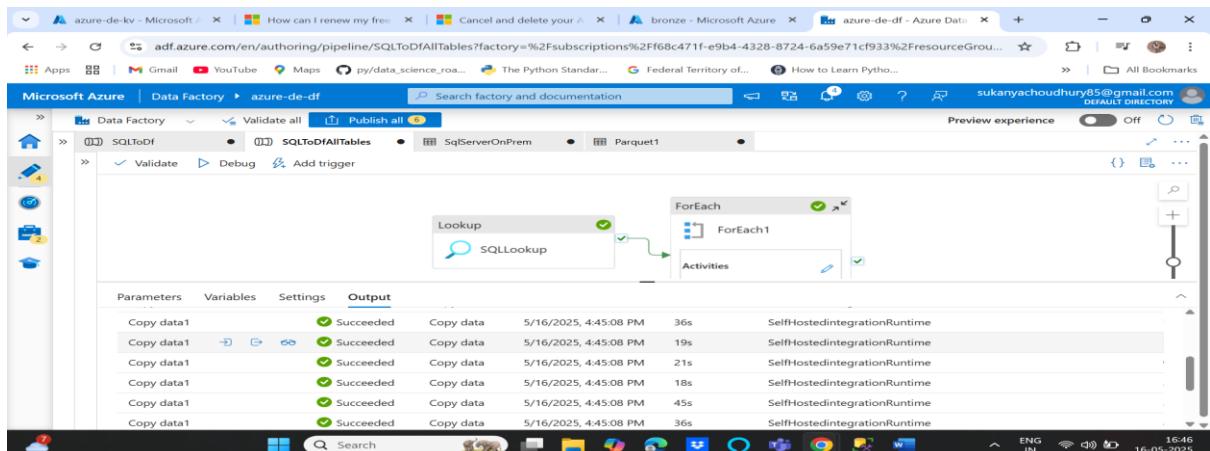
Finally sink data set looks like below

The screenshot shows the Microsoft Azure Data Factory interface with a 'Parquet1' dataset selected. The 'Parameters' tab is active, showing the 'File path' field with the expression `bronze / @{concat(dataset().schemaname, '/', dataset().tablename)} / @{concat(dataset().tablename, '.parquet')}`. The 'Compression type' is set to 'snappy'. At the top of the screen, there are several browser tabs open, including ones for Microsoft Edge, Google Chrome, and Microsoft Edge Dev. The taskbar at the bottom of the screen shows various pinned icons and the date and time as 16-05-2025.

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



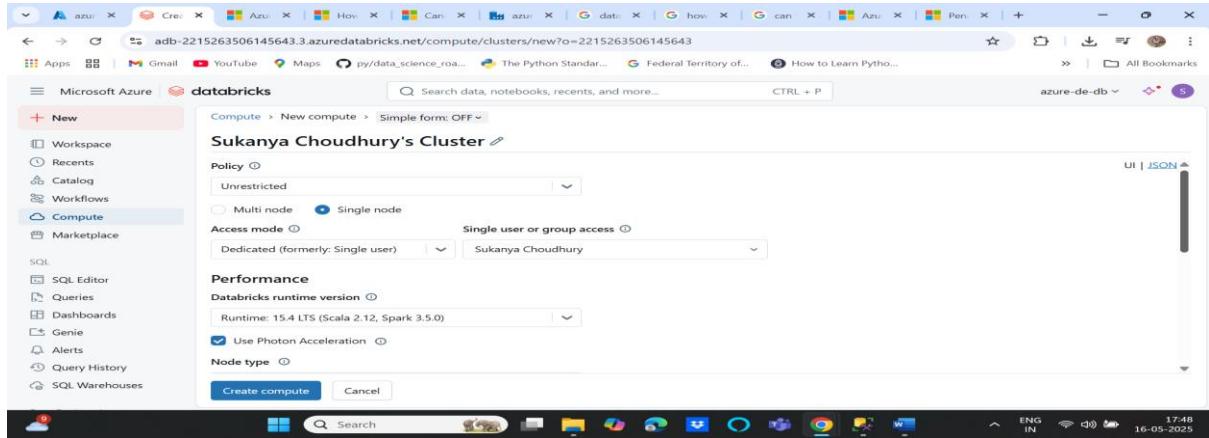
11) Debug lookup activity which runs the whole pipeline and extracts table from sql server into data lake bronze container and then publish all



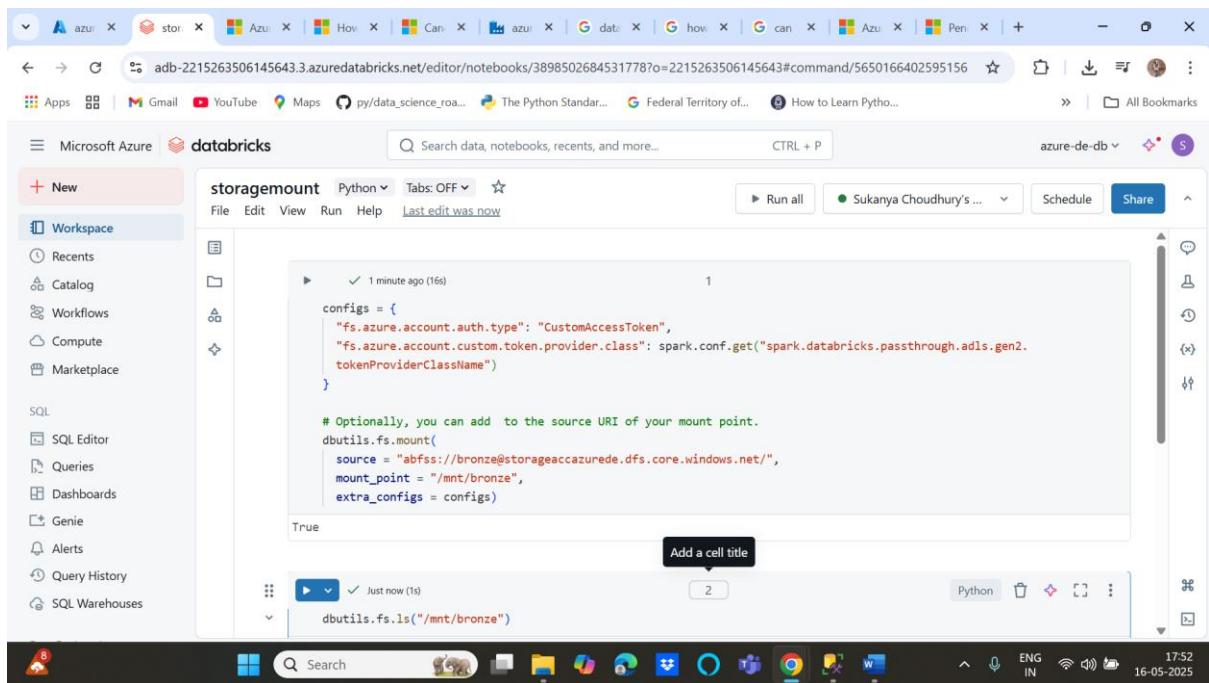
Step 3: Data Transformations using Data Bricks

1) Create a cluster

Launch Databricks Workspace → Compute → Create Compute



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



```

# Optionally, you can add to the silver source URI of your mount point.
dbutils.fs.mount(
    source = "abfss://silver@storageaccazurede.dfs.core.windows.net/",
    mount_point = "/mnt/silver",
    extra_configs = configs)

# Optionally, you can add to the source URI of your mount point.
dbutils.fs.mount(
    source = "abfss://gold@storageaccazurede.dfs.core.windows.net/",
    mount_point = "/mnt/gold",
    extra_configs = configs)

```

True

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

Bronze to silver : list bronze tables

Single table column transformation

```

dbutils.fs.ls('mnt/bronze/SalesLT')

[FileInfo(path='dbfs:/mnt/bronze/SalesLT/Address/', name='Address', size=0, modificationTime=1747394124000),
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/Customer/', name='Customer', size=0, modificationTime=1747394124000),
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/CustomerAddress/', name='CustomerAddress', size=0, modificationTime=174739415000),
 ...,
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/Product/', name='Product', size=0, modificationTime=1747394138000),
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/ProductCategory/', name='ProductCategory', size=0, modificationTime=174739412600),
 ...,
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/ProductDescription/', name='ProductDescription', size=0, modificationTime=1747394142000),
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/ProductModel/', name='ProductModel', size=0, modificationTime=1747394123000),
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/ProductModelProductDescription/', name='ProductModelProductDescription', size=0, modificationTime=1747394142000),
 ...,
 FileInfo(path='dbfs:/mnt/bronze/SalesLT/SalesOrderDetail/', name='SalesOrderDetail', size=0, modificationTime=1747394123000),
 ...]

```

Load the address table from bronze container

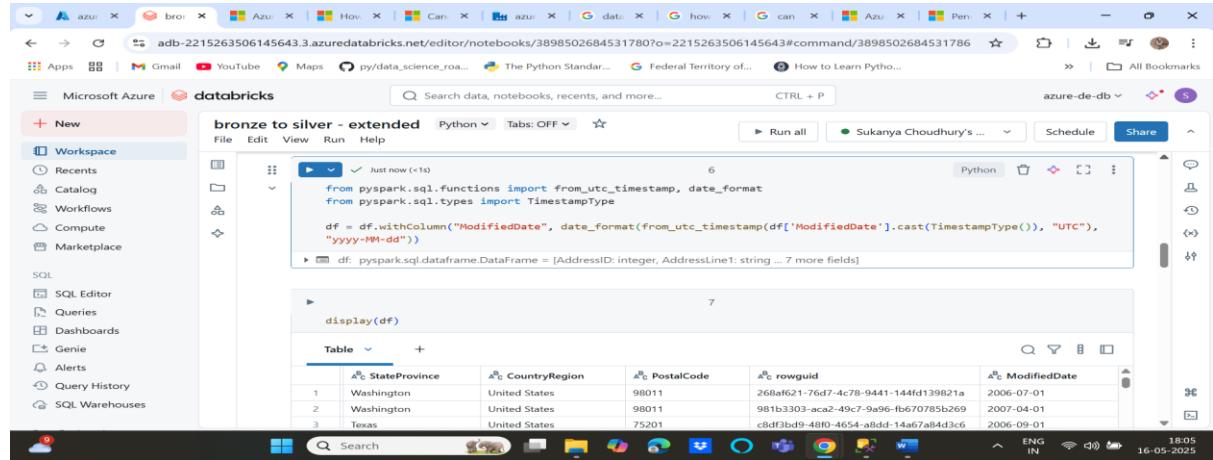
```

df = spark.read.format('parquet').load('mnt/bronze/SalesLT/Address/Address.parquet')

```

#	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion
1	9	8713 Yosemite Ct.		Bothell	Washington	United States
2	11	1318 Lasalle Street		Bothell	Washington	United States
3	25	9178 Jumping St.		Dallas	Texas	United States
4	28	9228 Via Del Sol		Phoenix	Arizona	United States
5	32	26910 Indela Road		Montreal	Quebec	Canada

Modified date is transformed into UTC date format for Address table



```

bronze to silver - extended Python File Edit View Run Help
Just now (<1s) 6
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

df = df.withColumn("ModifiedDate", date_format(from_utc_timestamp(df["ModifiedDate"].cast(TimestampType()), "UTC"), "yyyy-MM-dd"))

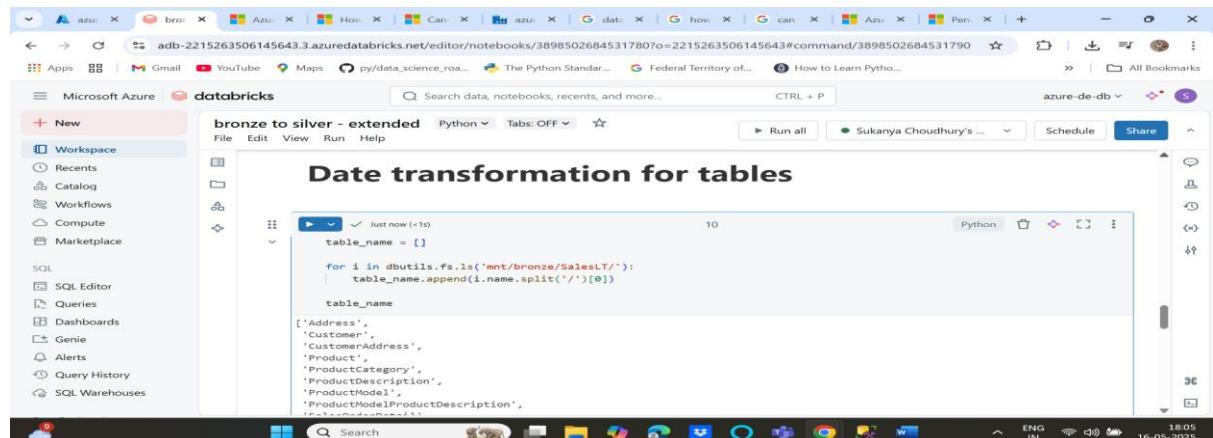
df: pyspark.sql.dataframe.DataFrame = [AddressID: integer, AddressLine1: string ... 7 more fields]

```

The screenshot shows a Databricks notebook titled "bronze to silver - extended". The code cell at the top contains Python code that uses the `from_utc_timestamp` function to convert the `ModifiedDate` column from UTC to a specific date format ("yyyy-MM-dd"). The resulting DataFrame is named `df`. Below the code cell, there is a preview of the DataFrame with three rows of data. The columns shown are `StateProvince`, `CountryRegion`, `PostalCode`, `rowguid`, and `ModifiedDate`. The data is as follows:

	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
1	Washington	United States	98011	268bf621-76d7-4c78-9441-144fd139821a	2006-07-01
2	Washington	United States	98011	981b3303-acaa-49c7-9a96-fb270785b269	2007-04-01
3	Texas	United States	75201	c8df3bd9-48f0-4654-a8dd-14a67a8d3c6	2006-09-01

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



```

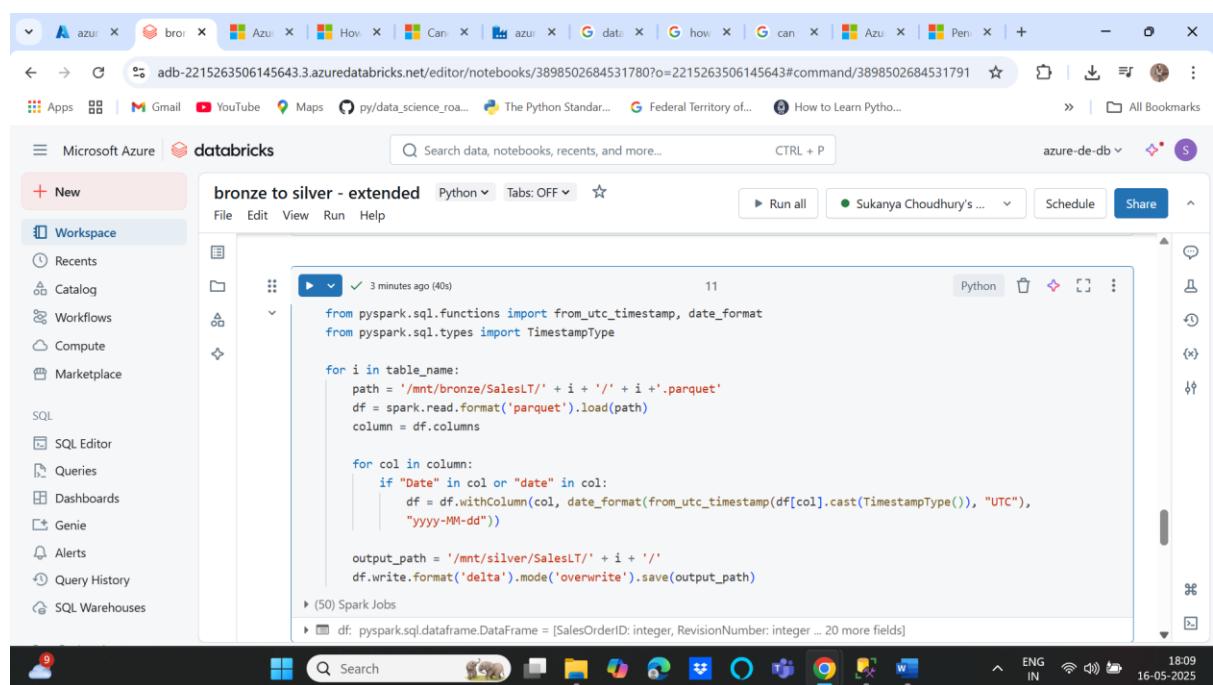
bronze to silver - extended Python File Edit View Run Help
Date transformation for tables
Just now (<1s) 10
table_name = []

for i in dbutils.fs.ls("mnt/bronze/SalesLT/"):
    table_name.append(i.name.split('/')[-1])

table_name

```

The screenshot shows a Databricks notebook titled "Date transformation for tables". The code cell contains Python code that reads the list of tables in the "SalesLT" directory of the bronze container and stores them in a list named `table_name`. The list includes tables like 'Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', and 'ProductModelProductDescription'.



```

bronze to silver - extended Python File Edit View Run Help
3 minutes ago (40s) 11
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

for i in table_name:
    path = '/mnt/bronze/SalesLT/' + i + '/' + i + '.parquet'
    df = spark.read.format('parquet').load(path)
    column = df.columns

    for col in column:
        if "Date" in col or "date" in col:
            df = df.withColumn(col, date_format(from_utc_timestamp(df[col].cast(TimestampType()), "UTC"), "yyyy-MM-dd"))

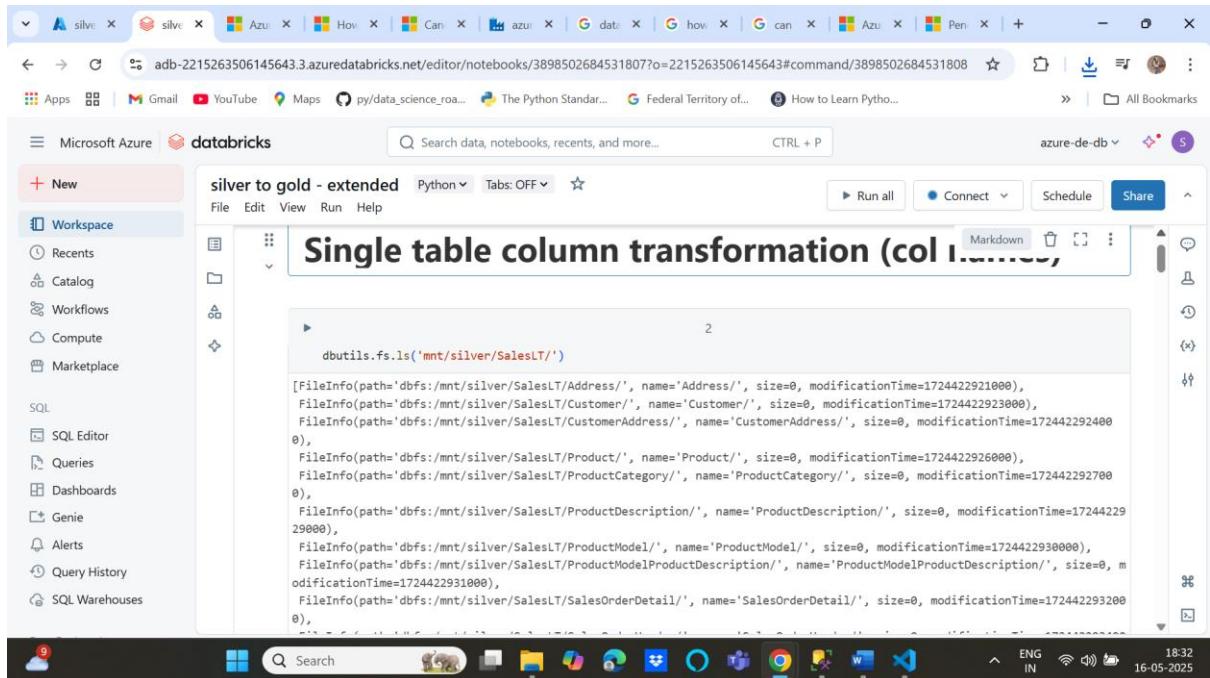
    output_path = '/mnt/silver/SalesLT/' + i + '/'
    df.write.format('delta').mode('overwrite').save(output_path)

(50) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [SalesOrderID: integer, RevisionNumber: integer ... 20 more fields]

```

The screenshot shows a Databricks notebook titled "bronze to silver - extended". The code cell contains Python code that iterates through the list of tables obtained earlier. For each table, it reads the parquet file, identifies date columns, converts them to UTC, and then saves the transformed data back to the silver container in delta mode. The code also includes a check for 50 Spark Jobs.

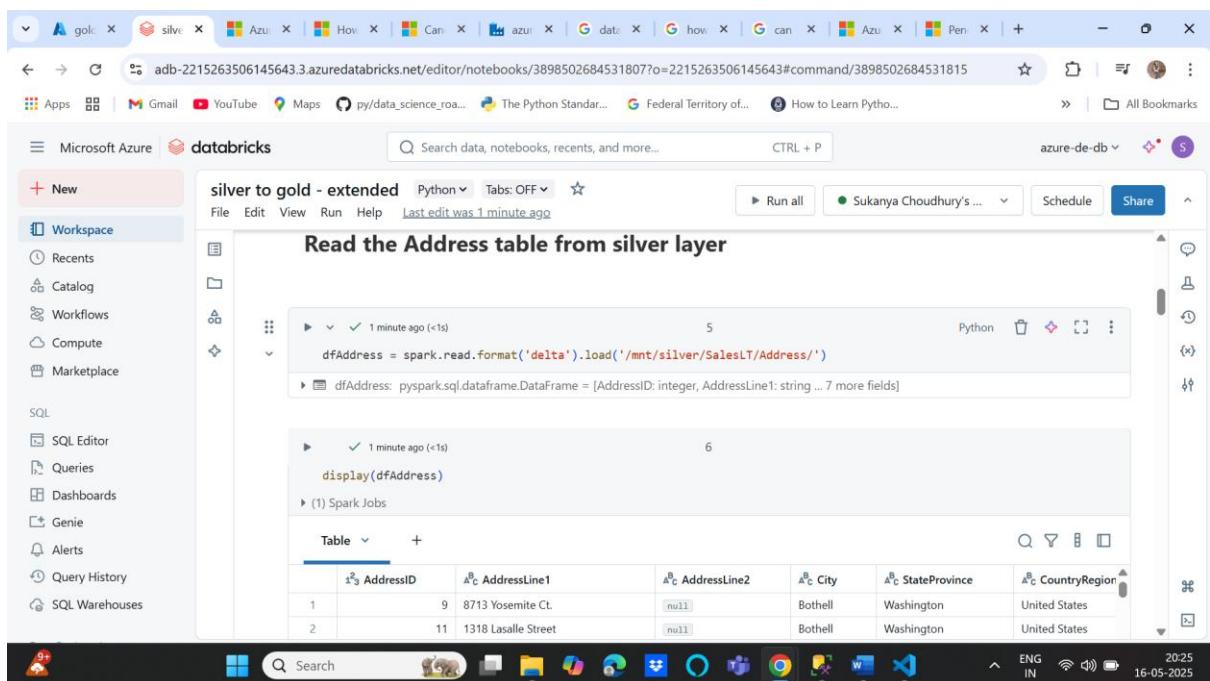
Silver to gold transformations



The screenshot shows the Azure Databricks workspace interface. On the left, the sidebar includes options like Recents, Catalog, Workflows, Compute, Marketplace, SQL, and various editors. The main area displays a notebook titled "silver to gold - extended" in Python. The code in the notebook is as follows:

```
dbutils.fs.ls('mnt/silver/SalesLT')  
[FileInfo(path='dbfs:/mnt/silver/SalesLT/Address/', name='Address', size=0, modificationTime=1724422921000),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/Customer/', name='Customer', size=0, modificationTime=1724422923000),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/CustomerAddress/', name='CustomerAddress', size=0, modificationTime=172442292400  
 0),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/Product/', name='Product', size=0, modificationTime=1724422926000),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/ProductCategory/', name='ProductCategory', size=0, modificationTime=172442292700  
 0),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/ProductDescription/', name='ProductDescription', size=0, modificationTime=17244229  
 29000),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/ProductModel/', name='ProductModel', size=0, modificationTime=1724422930000),  
 FileInfo(path='dbfs:/mnt/silver/SalesLT/ProductModelProductDescription/', name='ProductModelProductDescription', size=0, m  
 odificationTime=1724422931000),  
 FileInfo(path='dbfs:/mnt/silver/SalesOrderDetail/', name='SalesOrderDetail', size=0, modificationTime=172442293200  
 0),  
 FileInfo(path='dbfs:/mnt/silver/SalesOrderDetailLineItem/', name='SalesOrderDetailLineItem', size=0, modificationTime=1724422933000)
```

Silver to Gold

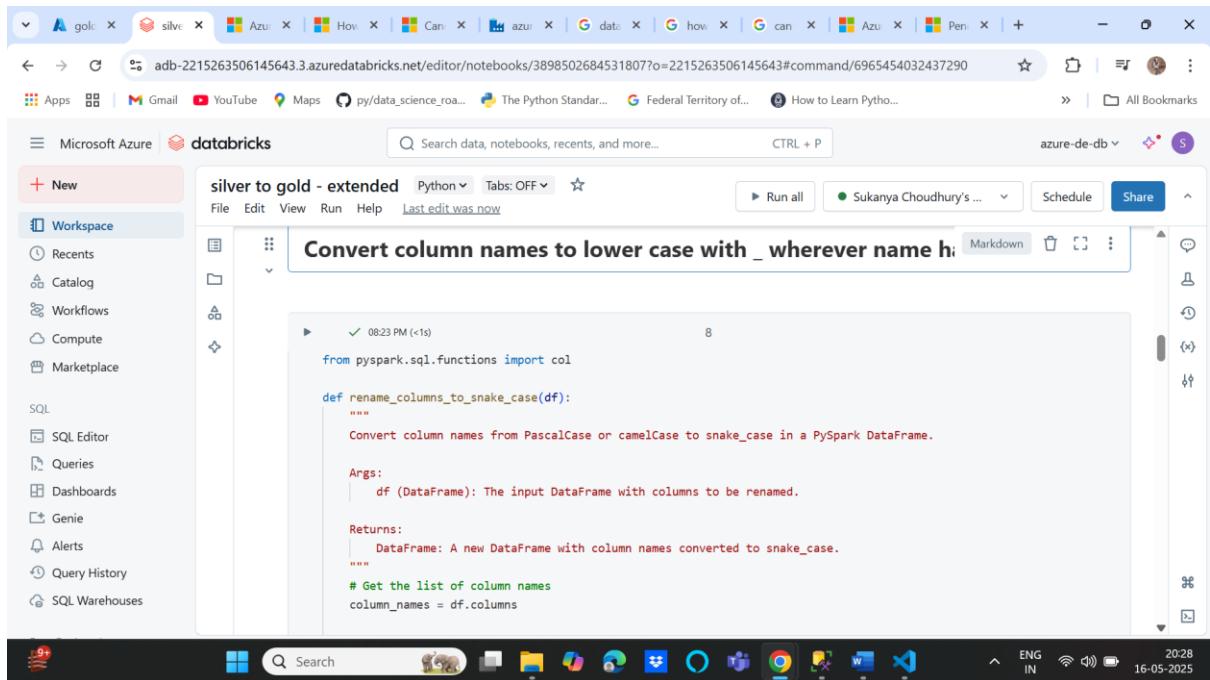


The screenshot shows the Azure Databricks workspace interface. The sidebar and notebook structure are identical to the previous one. The notebook title is "silver to gold - extended". The code in the notebook reads the "Address" table from the silver layer:

```
dfAddress = spark.read.format('delta').load('/mnt/silver/SalesLT/Address')  
dfAddress: pyspark.sql.dataframe.DataFrame = [AddressID: integer, AddressLine1: string ... 7 more fields]
```

Below the code, a table view displays the data:

AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion
1	8713 Yosemite Ct.	null	Bothell	Washington	United States
2	1318 Lasalle Street	null	Bothell	Washington	United States

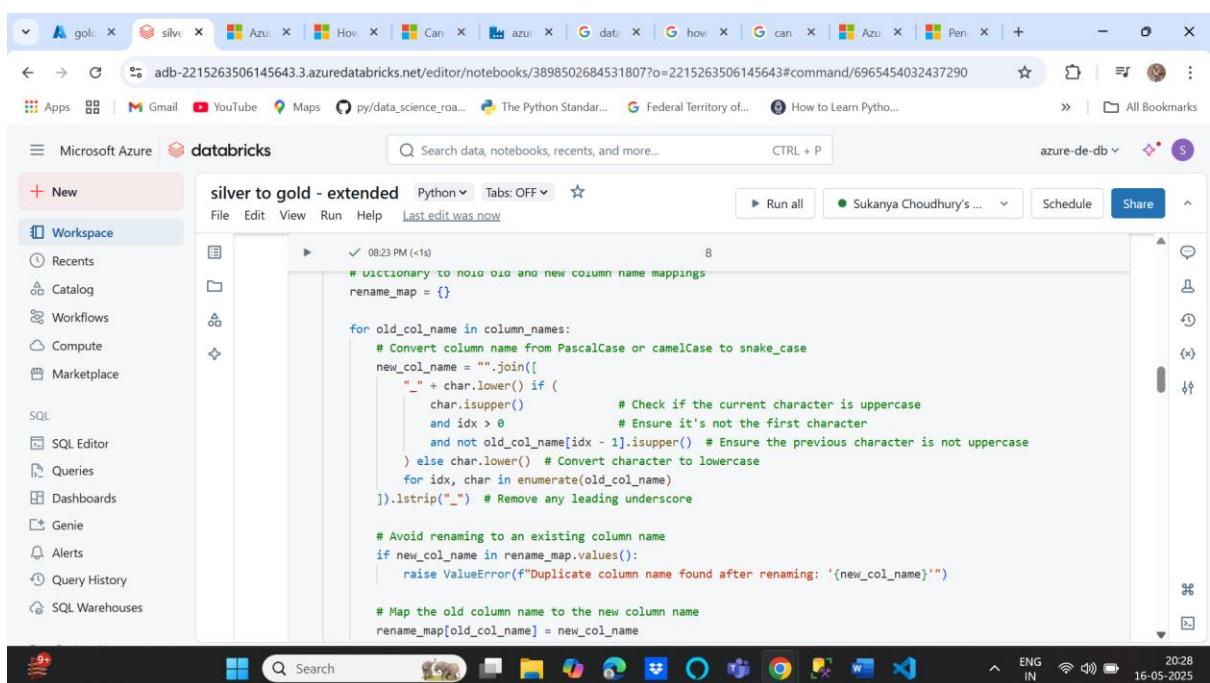


```
silver to gold - extended Python Tabs: OFF
File Edit View Run Help Last edit was now
Convert column names to lower case with _ wherever name has uppercase letters
from pyspark.sql.functions import col

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
```



```
# dictionary to map 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() and idx > 0 and not old_col_name[idx - 1].isupper()) # Check if the current character is uppercase and it's not the first character and the previous character is not uppercase
        else char.lower() # Convert character to lowercase
    ]).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
```

Screenshot of a Microsoft Edge browser window showing a Databricks notebook titled "silver to gold - extended". The notebook is in Python mode. The code in the notebook is:

```
# 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)
```

The notebook has two runs:

- Run 8: # Rename columns using the mapping
- Run 9: df = rename_columns_to_snake_case(dfAddress)

The status bar at the bottom right shows the date as 16-05-2025 and the time as 20:28.

Screenshot of a Microsoft Edge browser window showing a Databricks notebook titled "silver to gold - extended". The notebook is in Python mode. The code in the notebook is:

```
df = rename_columns_to_snake_case(dfAddress)
```

The notebook has two runs:

- Run 10: df = rename_columns_to_snake_case(dfAddress)
- Run 11: display(df)

After running "display(df)", a table is displayed:

address_id	address_line1	address_line2	city	state_province	country_region
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

The status bar at the bottom right shows the date as 16-05-2025 and the time as 20:29.

All table columns transformation (col names)

```
# 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=1747399002000),
 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=1747399007000)]

```
table_name = []

for i in dbutils.fs.ls('mnt/silver/SalesLT'):
    table_name.append(i.name.split('/')[0])
```

```
table_name
```

['Address',
 'Customer',
 'CustomerAddress',
 'Product',
 'ProductCategory',
 'ProductDescription',
 'ProductModel',
 'ProductModelProductDescription',
 'SalesOrderDetail']

```
for name in table_name:
    path = '/mnt/gold/SalesLT/' + name
    print(path)
    df_tables = spark.read.format('delta').load(path)

    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

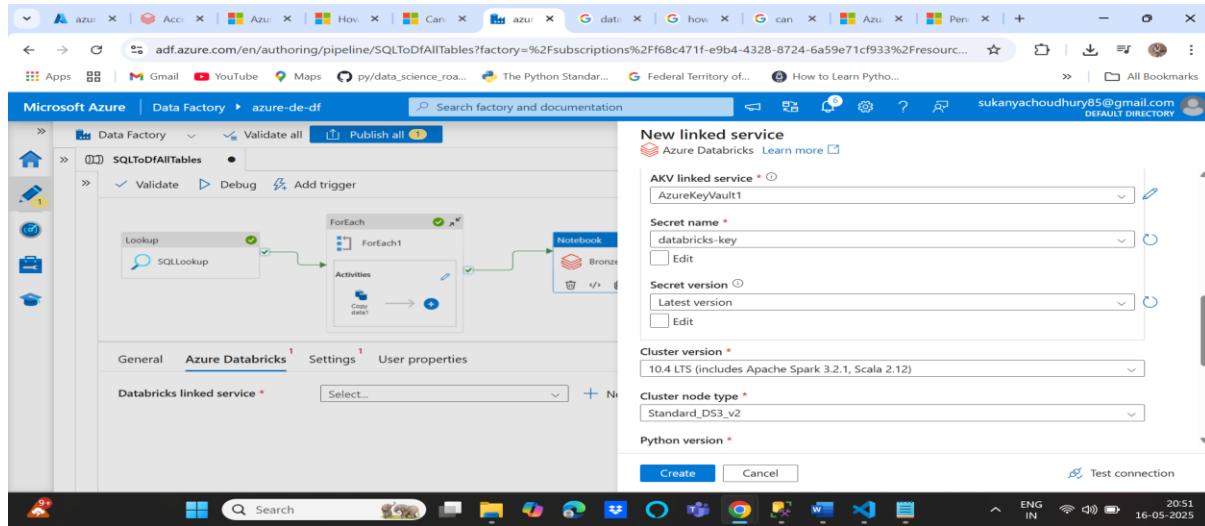
df: pyspark.sql.dataframe.DataFrame = [sales_order_id: integer, revision_number: integer ... 20 more fields]

df_tables: pyspark.sql.dataframe.DataFrame = [SalesOrderID: integer, RevisionNumber: integer ... 20 more fields]

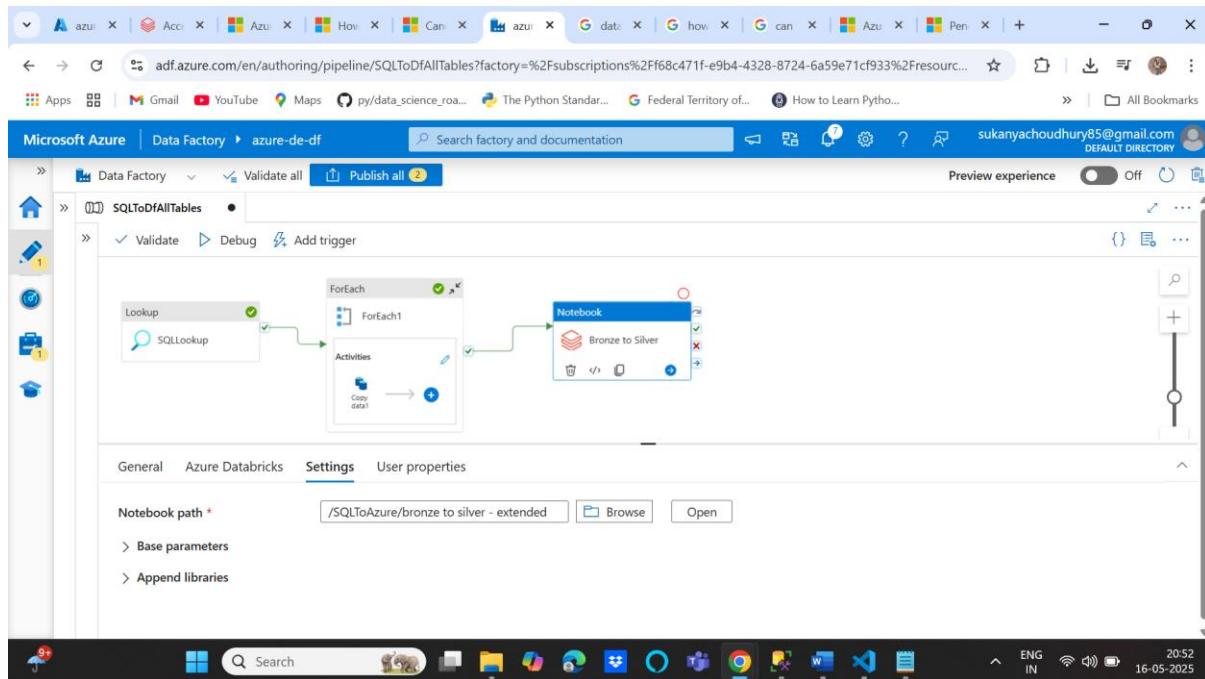
/mnt/silver/SalesLT/Address
/mnt/silver/SalesLT/Customer
/mnt/silver/SalesLT/CustomerAddress
/mnt/silver/SalesLT/Product
/mnt/silver/SalesLT/ProductCategory
/mnt/silver/SalesLT/ProductDescription

- 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

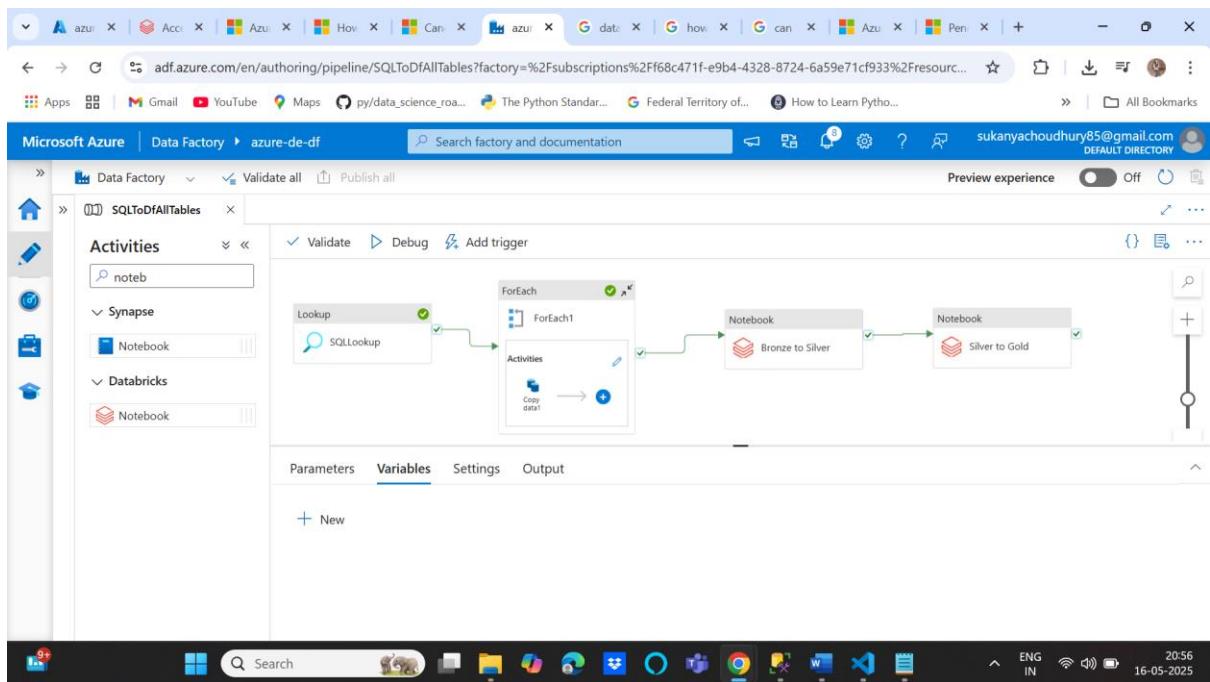
Create a databricks linked service for the notebook activity using access token from databricks settings → developer → access tokens → Generate access token and then add it on the linked service configuration or store it in key vault → secret → generate/import and then use it



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

The screenshot shows the Microsoft Azure Synapse Analytics studio. A SQL script titled "SQL script 1" is open, displaying the following auto-generated code:

```

1 -- This is auto-generated code
2 SELECT
3     TOP 100 *
4 FROM
5     OPENROWSET(
6         BULK 'https://storageaccazurede.dfs.core.windows.net/gold/SalesL'
7         FORMAT = 'PARQUET'
8     ) AS [result]
9

```

The left sidebar shows a "Data" workspace with a "Linked" section containing a "gold" folder. The "Properties" pane on the right shows the script is a ".sql script" type with a size of 206 bytes.

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

```

CREATE VIEW SalesLT_AddressView_AS
SELECT *
FROM
OPENROWSET(
    BULK 'https://storageaccazurede.dfs.core.windows.net/gold/SalesLT/Address/',
    FORMAT = 'DELTA'
) AS [result]

```

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

```

USE azure_de_sadb
GO
CREATE OR ALTER PROC CreateSQLServerlessView_gold @ViewName nvarchar(100)
AS
BEGIN
    DECLARE @statement VARCHAR(MAX)
    SET @statement = N'CREATE OR ALTER VIEW ' + @ViewName + ' AS
        SELECT *
        FROM
        OPENROWSET(
            BULK ''https://storageaccazurede.dfs.core.windows.net/gold/SalesLT/' + @ViewName + '/',
            FORMAT = ''DELTA''
        ) AS [result]'

    EXEC (@statement)
END
GO

```

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

New linked service

Account selection method

- From Azure subscription
- Enter manually

Fully qualified domain name *

Database name *

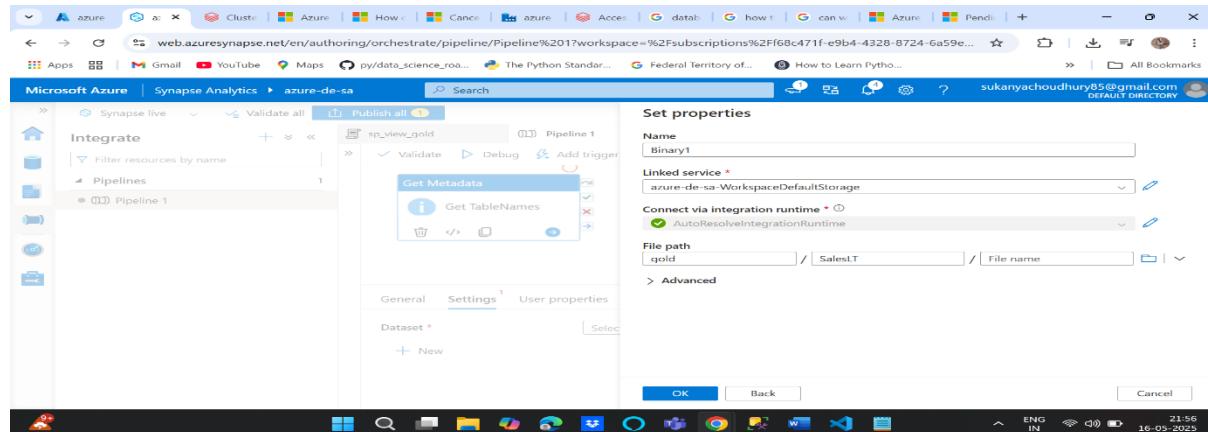
Authentication type *

Managed identity name: **azure-de-sa**
Managed identity object ID: **a01b44cb-9835-472d-b3e1-4d3b17359510**
Grant workspace service managed identity access to your Azure SQL Database.

Create Back Connection successful Test connection Cancel

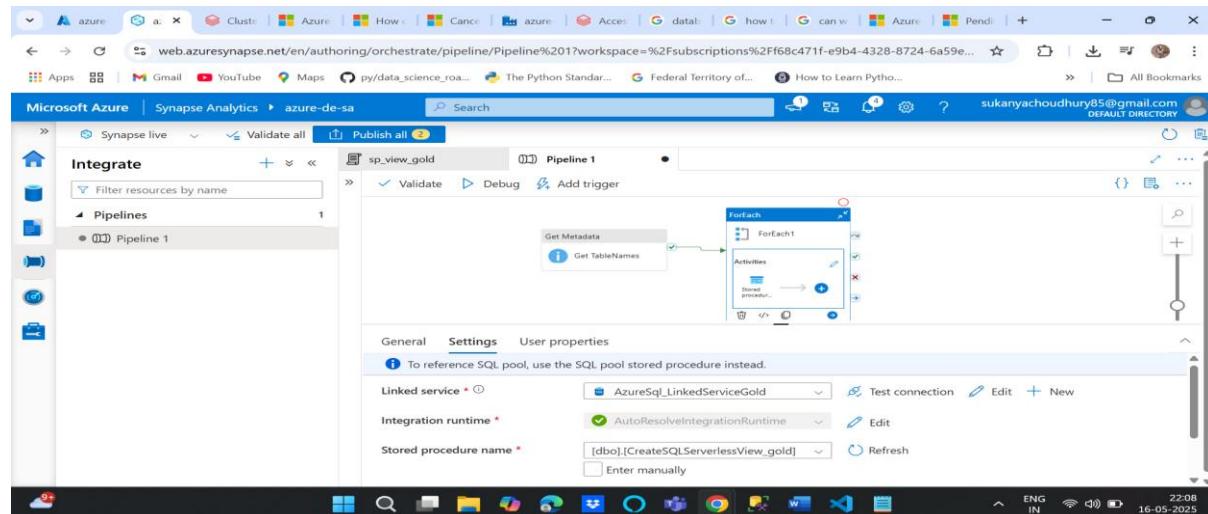
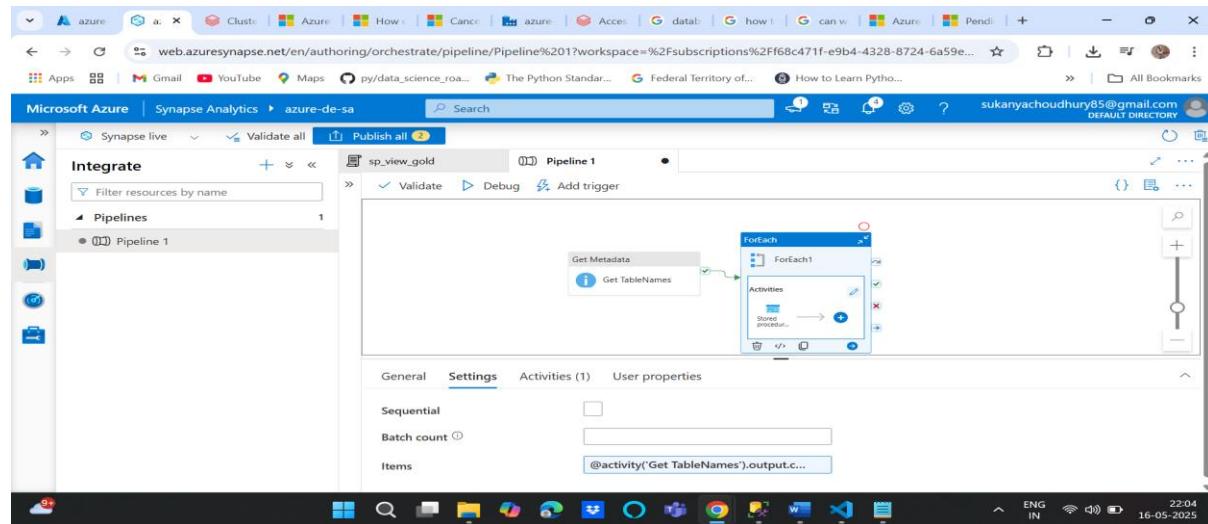
Now go to integrate → + → Pipeline → Add Get Metadata activity → settings—add dataset → azure data lake gen2 → binary format → select the default linked service and the gold directory

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



Stored procedure parameters

[Import](#) [New](#) [Delete](#)

Name	Type *	Value
ViewName	String	@item().name

Run the pipeline

All pipeline runs > ProcedurePipeline - Activity runs

Activity runs

Pipeline run ID 349f77db-4973-41e4-8359-b2780e495217

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime
Get TableNames	Succeeded	Get Metadata	5/16/2025, 10:14:52 PM	6s	AutoResolveIntegrationRuntime (Southeast Asia)
ForEach1	Succeeded	ForEach	5/16/2025, 10:15:01 PM	38s	
Stored procedure1	Succeeded	Stored procedu	5/16/2025, 10:15:04 PM	24s	AutoResolveIntegrationRuntime (Southeast Asia)

All views are created

Data

Workspace

Linked

Filter resources by name

SQL database

- azure-de-sadb (SQL)
 - External tables
 - External resources
 - Views
 - dbo.Address
 - dbo.Customer
 - dbo.CustomerAddress
 - dbo.Product
 - dbo.ProductCategory
 - dbo.ProductDescription

4) Trigger pipeline on a schedule:

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

New trigger

Type * Schedule

Start date * 5/19/2025, 1:27:27PM

Time zone * Chennai, Kolkata, Mumbai, New Delhi (UTC+5:30)

Recurrence * Every 1 Day(s)

Advanced recurrence options

Execute at these times

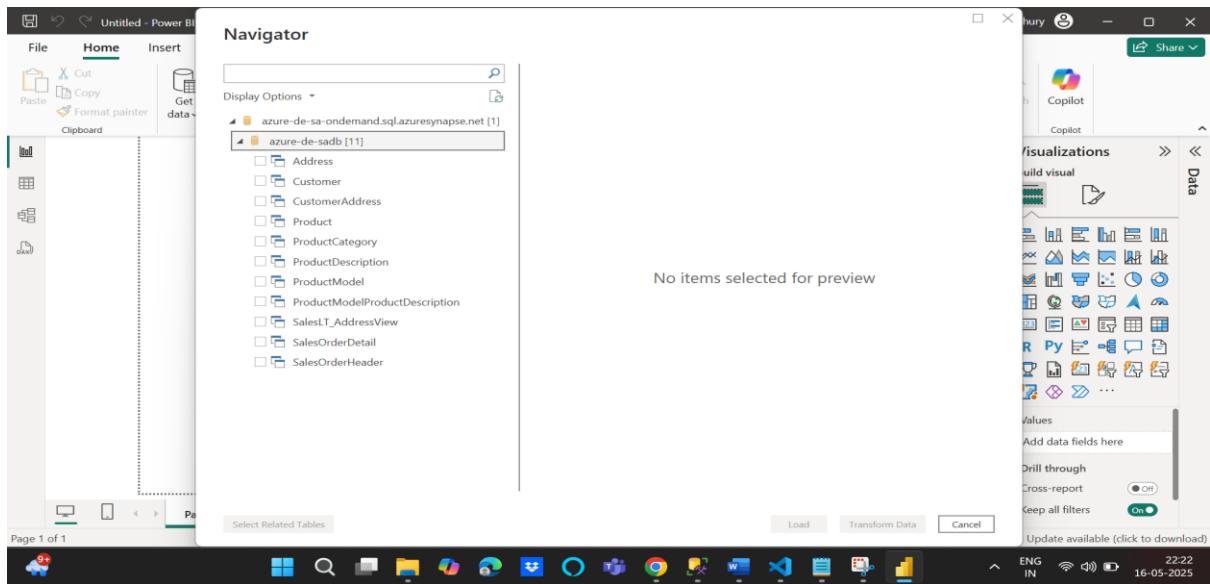
Hours 19

Minutes 5

OK Cancel

Step 5: Reporting using Power BI

- 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

