Daily Data Aggregation

~By

Alekhya Krishna Balivada

alekhya0830@gmail.com

Table of Contents

1. Project Statement

Implement a daily data aggregation pipeline using Azure Data Factory to move raw data and Azure Databricks to aggregate and summarize the data based on daily intervals.

2. Project Overview

The project aims to use Azure Data Factory (ADF) and Azure Databricks to construct a daily data aggregation pipeline. With the help of this pipeline, raw data from source systems will be moved to a location where Azure Databricks will be used for daily data aggregation and summarization.

3. Project Requirements

a. Azure Subscription
b. Data Sources
c. Azure Storage account
d. Azure Databricks
e. Azure Data Factory

3.1 Azure Subscription

- We need Azure Subscription plan to perform this project

3.2 Data Sources

- I have taken a CSV file as 'project.csv'.
- I performed various operations on this 'project.csv' file.

3.3 Azure Storage account

- An Azure Storage Account is a Microsoft Azure service that provides highly scalable and durable cloud storage for various types of data. It serves as a central repository for storing and managing different types of data, including blobs, files, tables, and queues.
- The 'project.csv' file is uploaded inside the storage account container.

3.4 Azure Databricks

- A Databricks workspace is established to leverage Apache Spark for data processing.
- A Databricks cluster is configured with the necessary libraries and settings.
- A Databricks notebook is developed to perform data aggregation and summarization.
- The notebook is parameterized to accept a date range, enabling daily processing.

3.5 Azure Data Factory

- ADF is utilized for data orchestration and movement.

- Linked services are configured to establish connections with raw data sources and destination.
- Datasets are defined to represent raw data and the destination for aggregated data.
- An ADF pipeline is created to copy raw data from source to destination using Data Flow activity.

4. Architecture

5. Execution overview

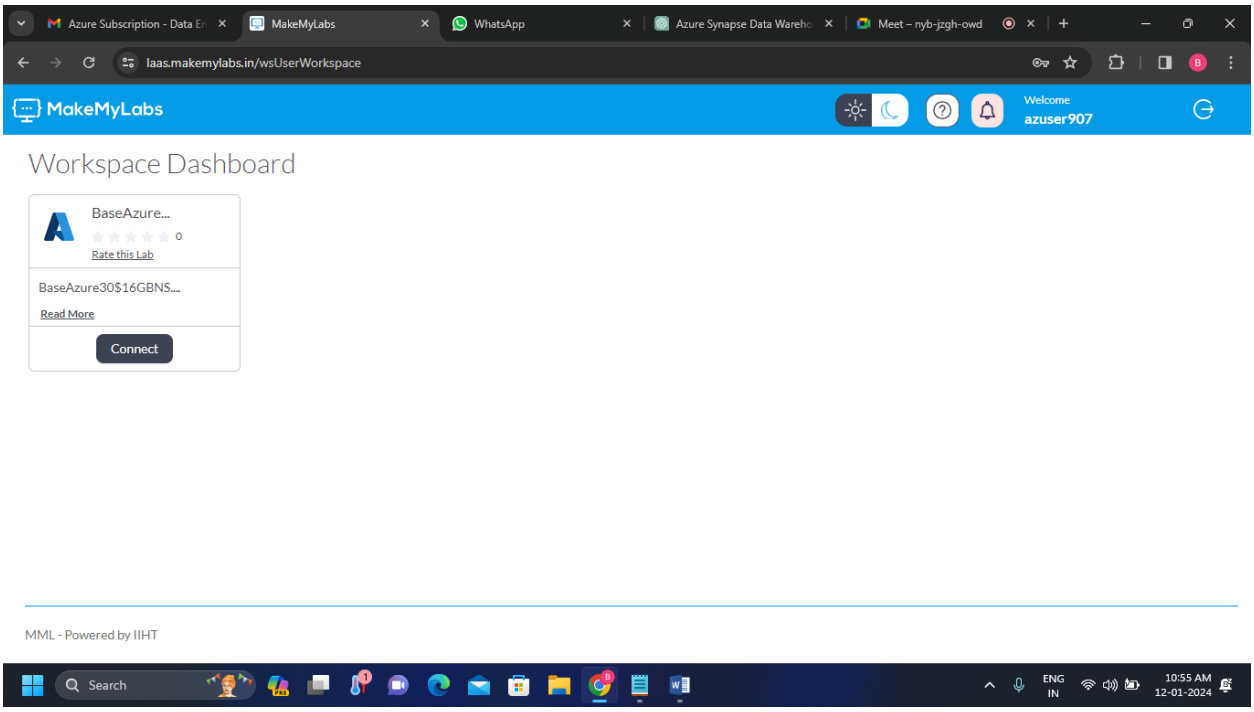
a. Login to the Azure Subscription account.
b. Create a storage account and upload a file inside a storage account container.
c. Create an Azure Databricks workspace.
d. Create a cluster and notebook in databricks workspace.
e. Connect ADLS with ADF.
f. Create Azure data factory resource.
g. Create pipeline for databricks in ADF.
h. Debug to get the final output.

6. Project implementation

- Project implementation involves the step by step procedure along with screenshots that we performed in the Azure account.

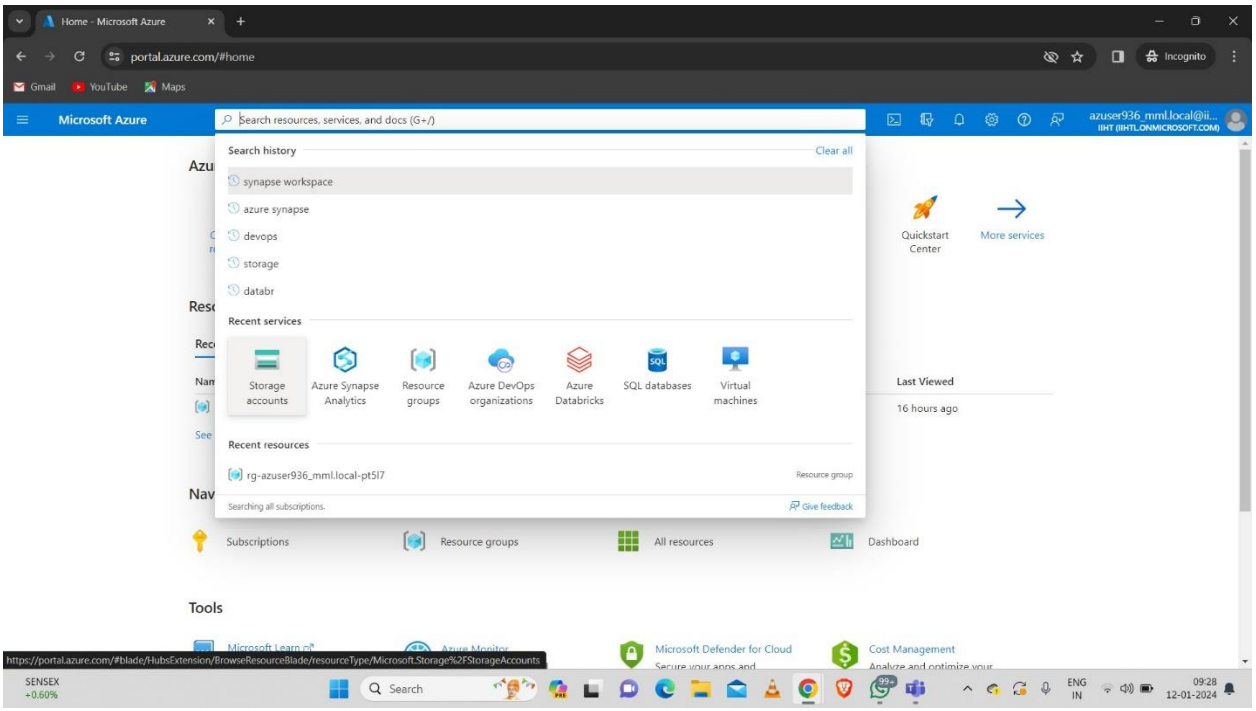
6.1 Azure account

- Log in to the Azure account and connect to the Workspace Dashboard.

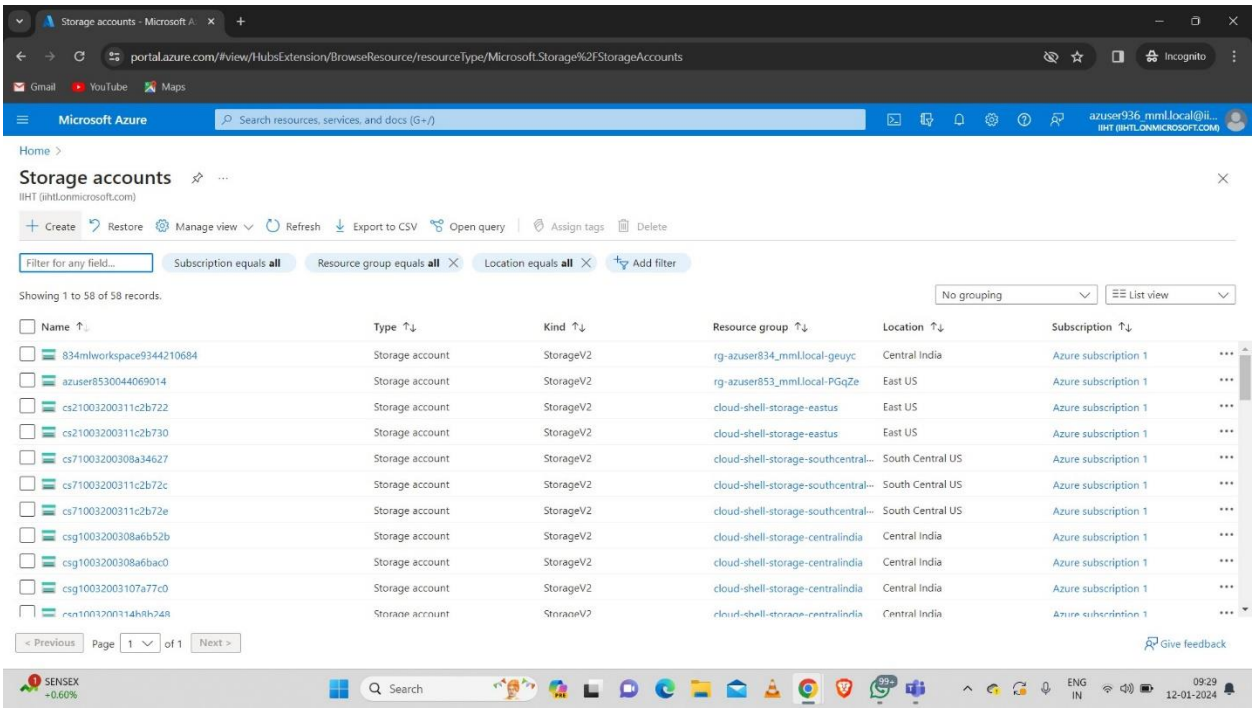


6.2 Storage account

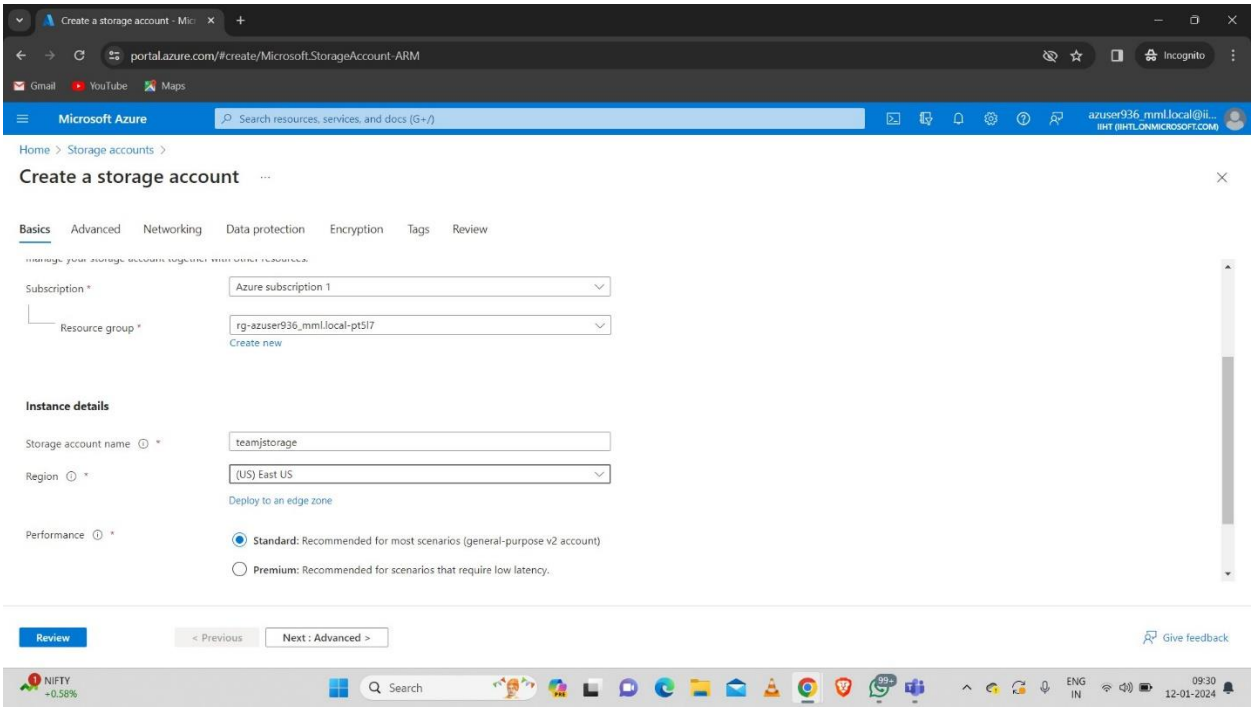
Step-1: Search for Storage account.

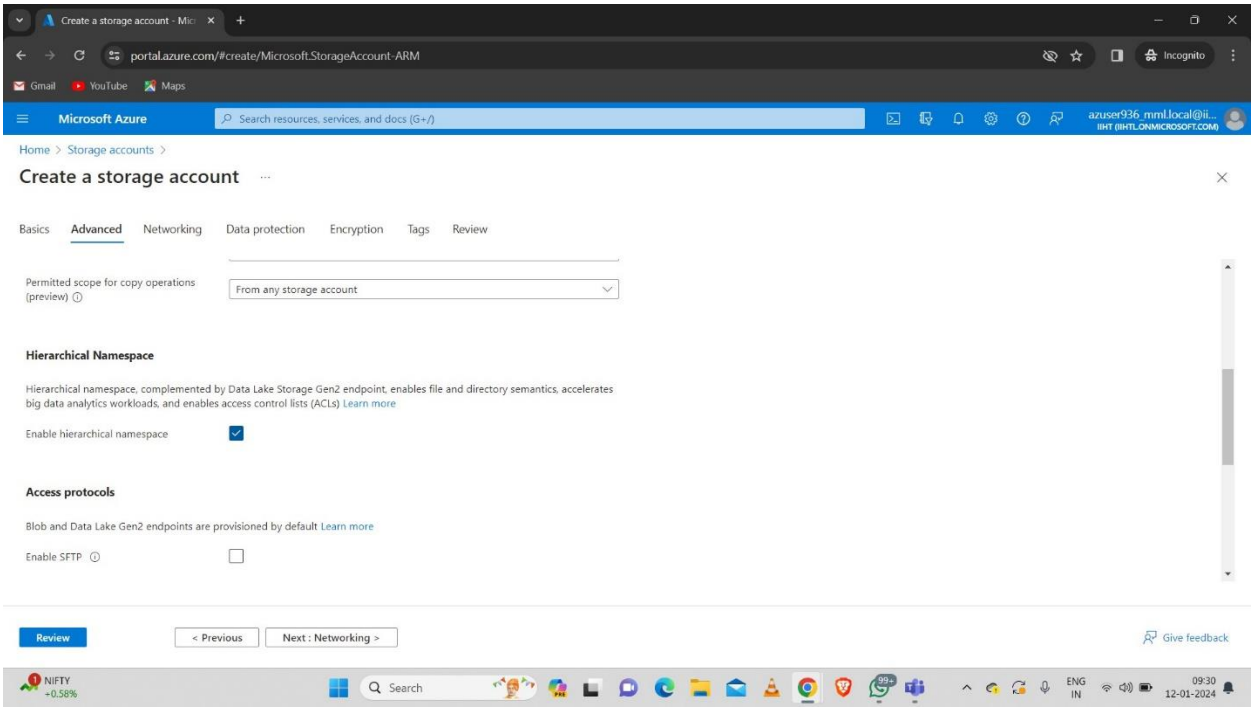


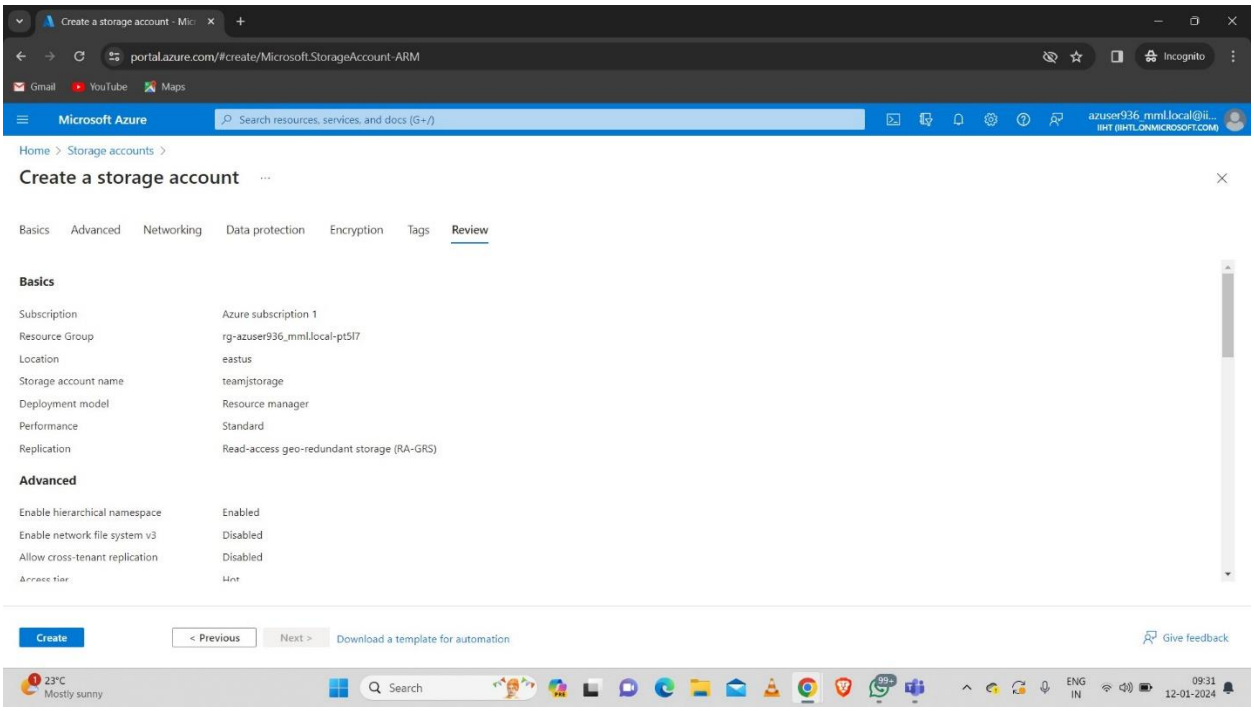
Step-2: Click on create to create the storage account.



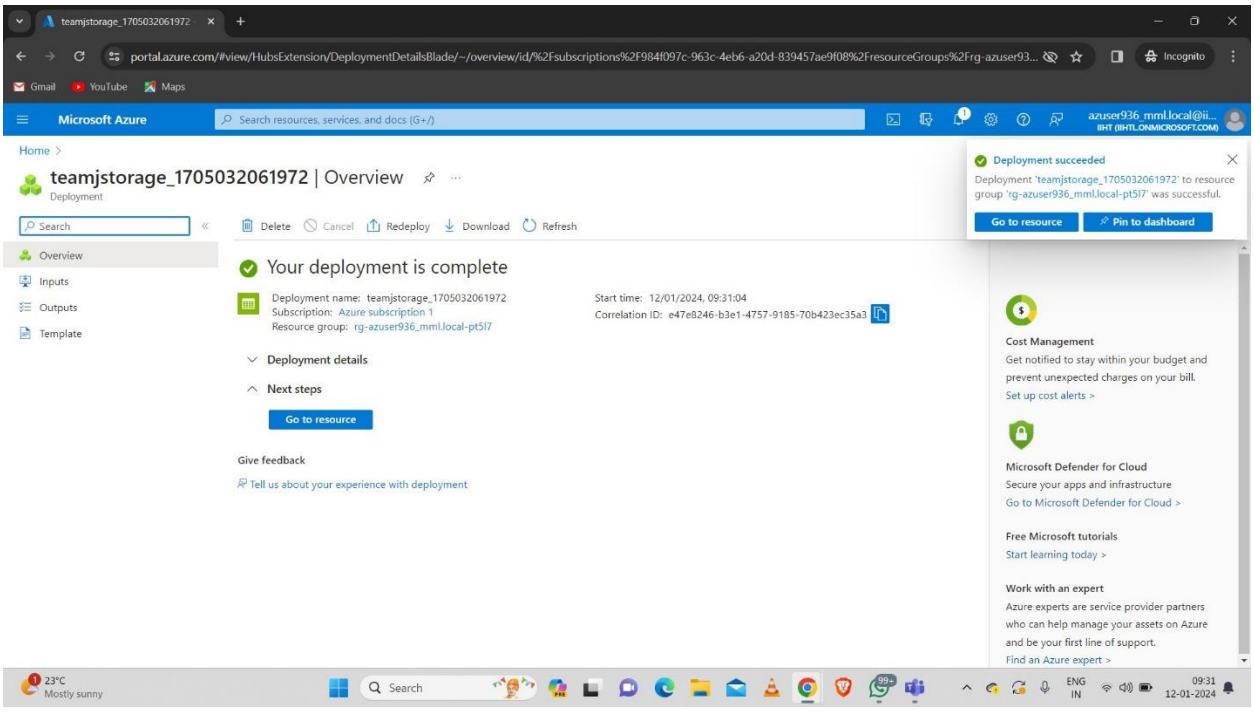
Step-3: Fill the necessary data. Here I have created a storage account with name “teamjstorage”.





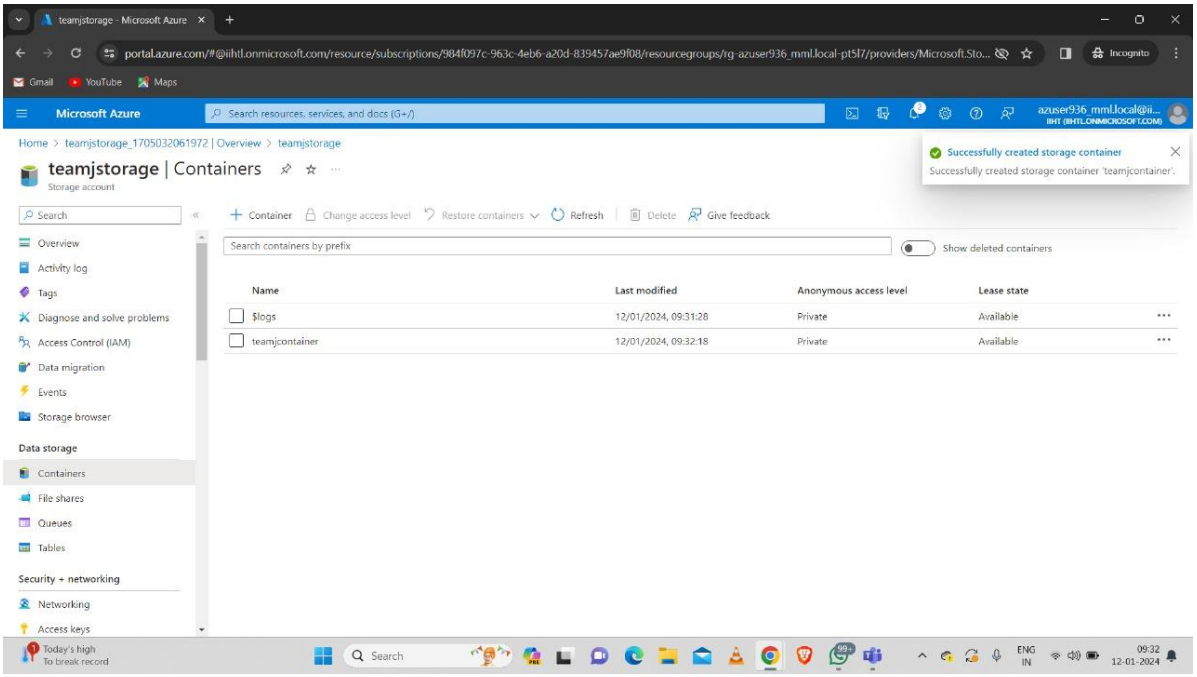


- Here we can see that the storage account is successfully created.

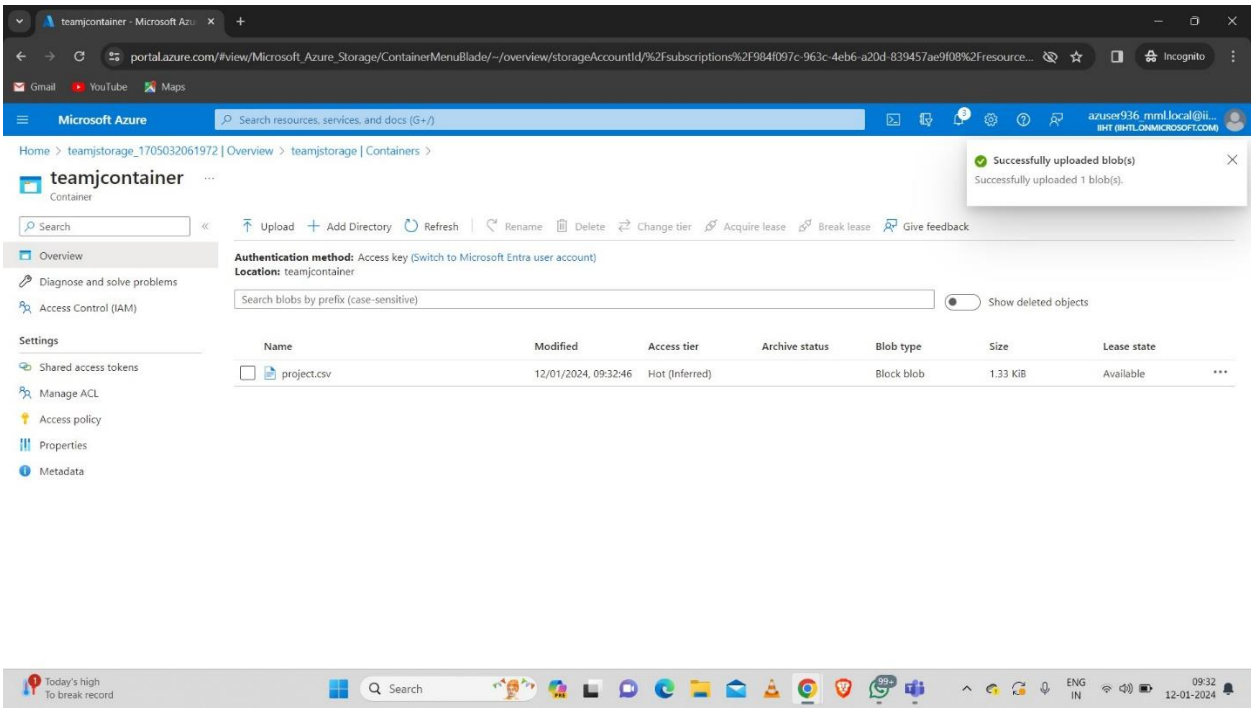


6.3 Uploading CSV file in storage account container

Step-1: Open the storage account and create a container. Here we created a container with name “teamjcontainer”.

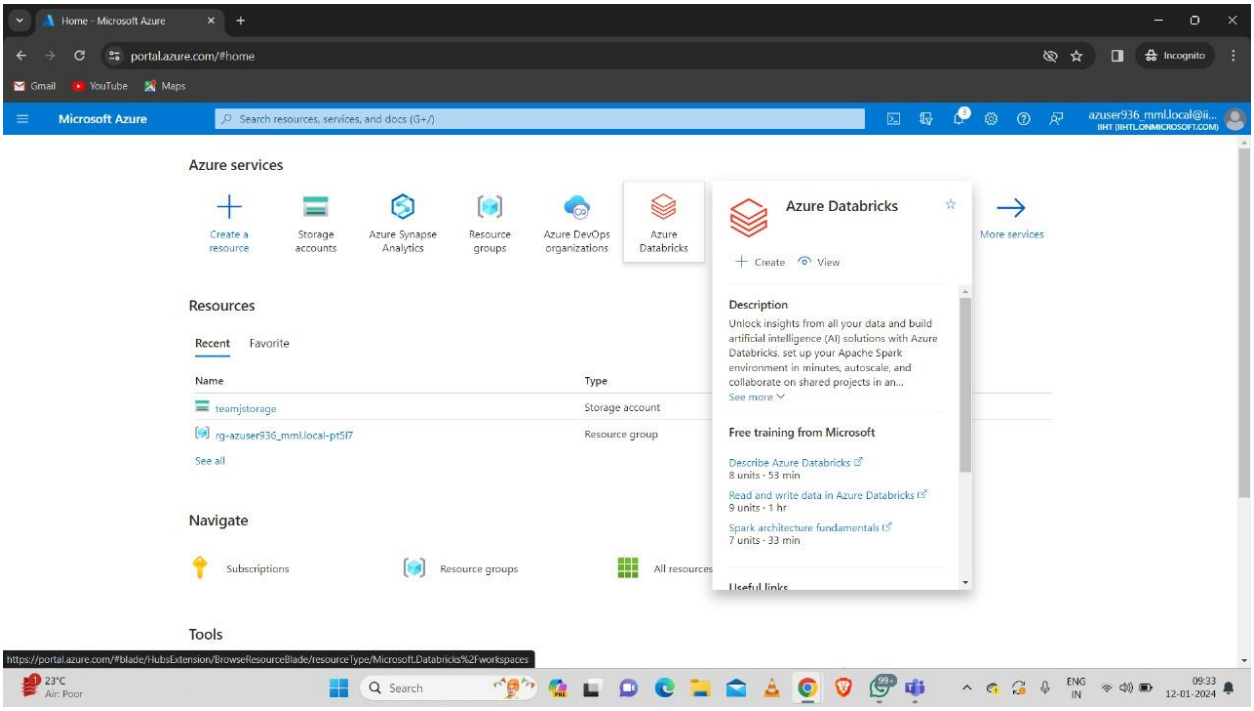


Step-2: Open the container and upload the csv file inside the container. Here I uploaded “project.csv” file inside the container.

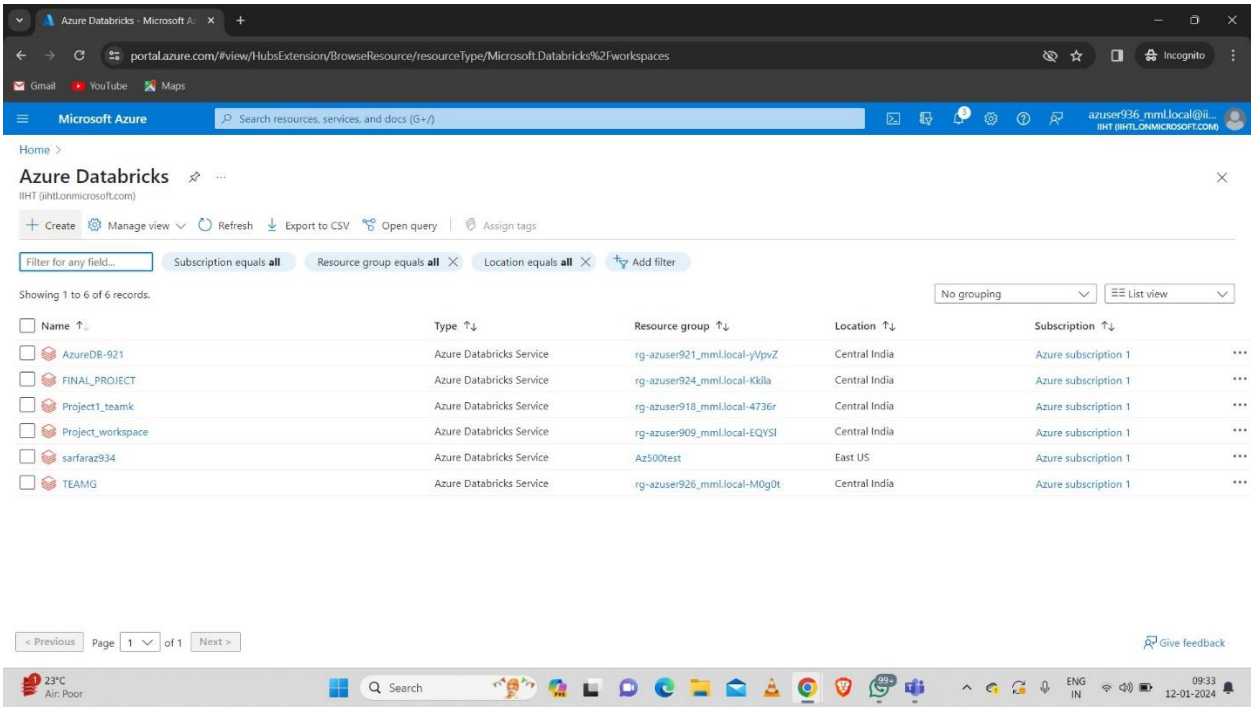


6.4 Azure Databricks

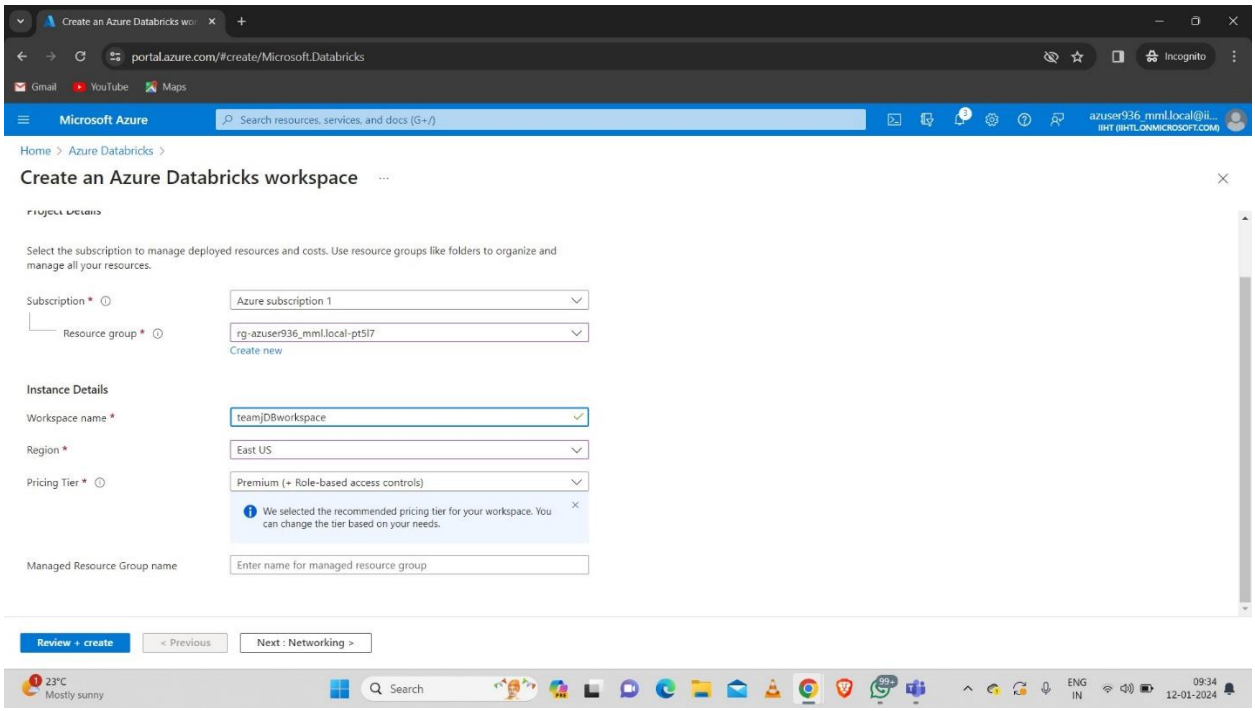
Step-1: Search for Azure Databricks



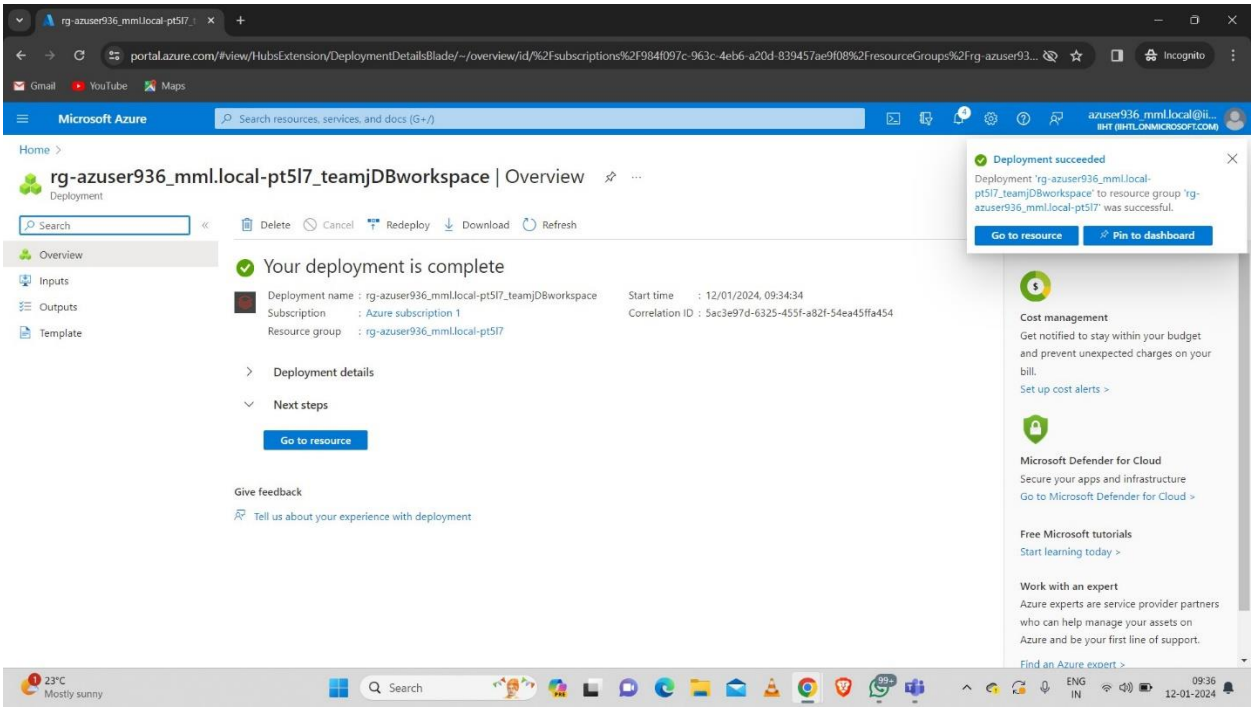
Step-2: Click on create to create the databricks workspace.



Step-3: Fill the necessary fields. Here I gave the name for workspace as “teamjDBworkspace”.

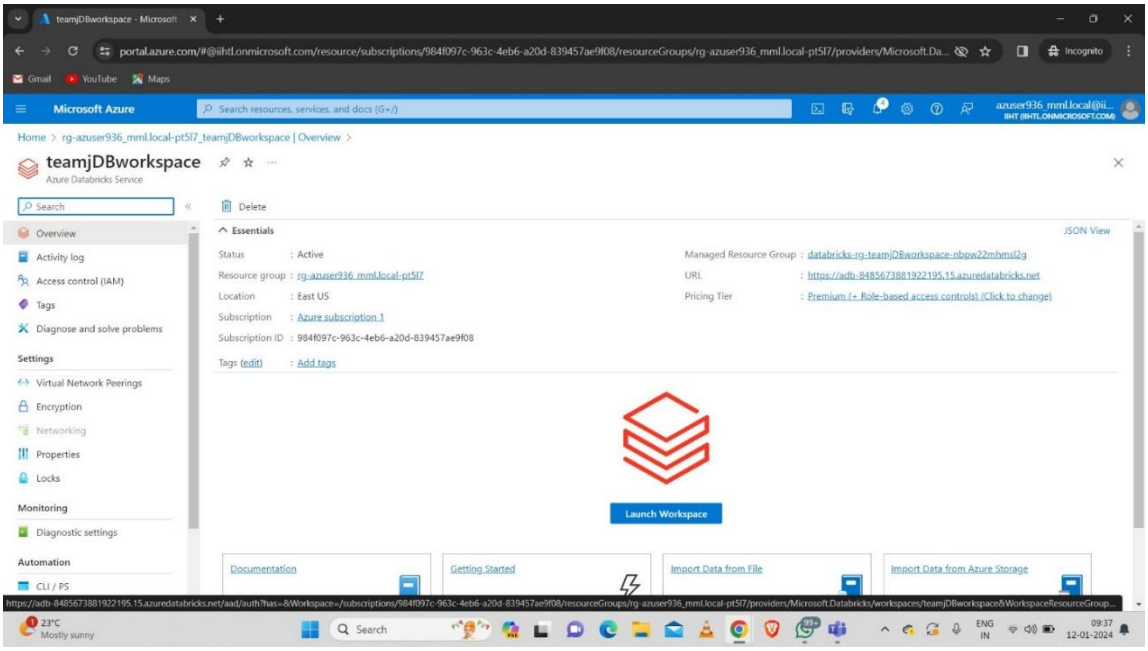


Step-4: Databricks workspace is successfully created.

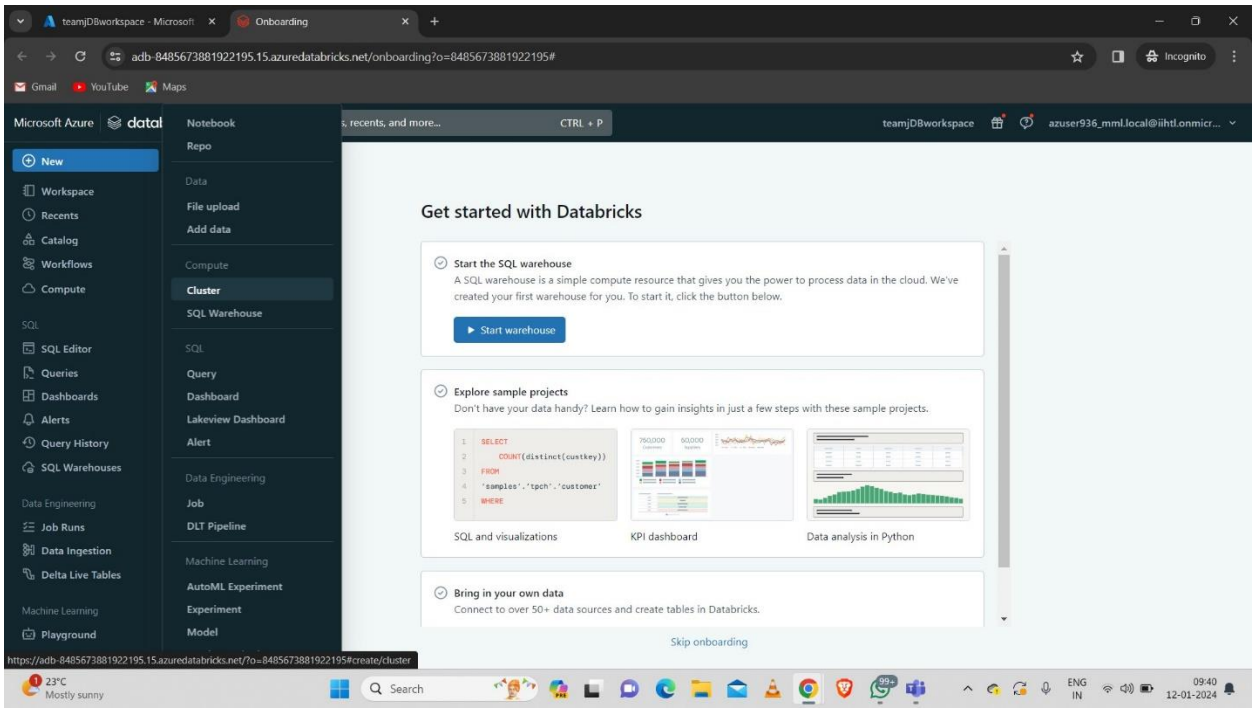


6.5 Creating Databricks Cluster and Notebook

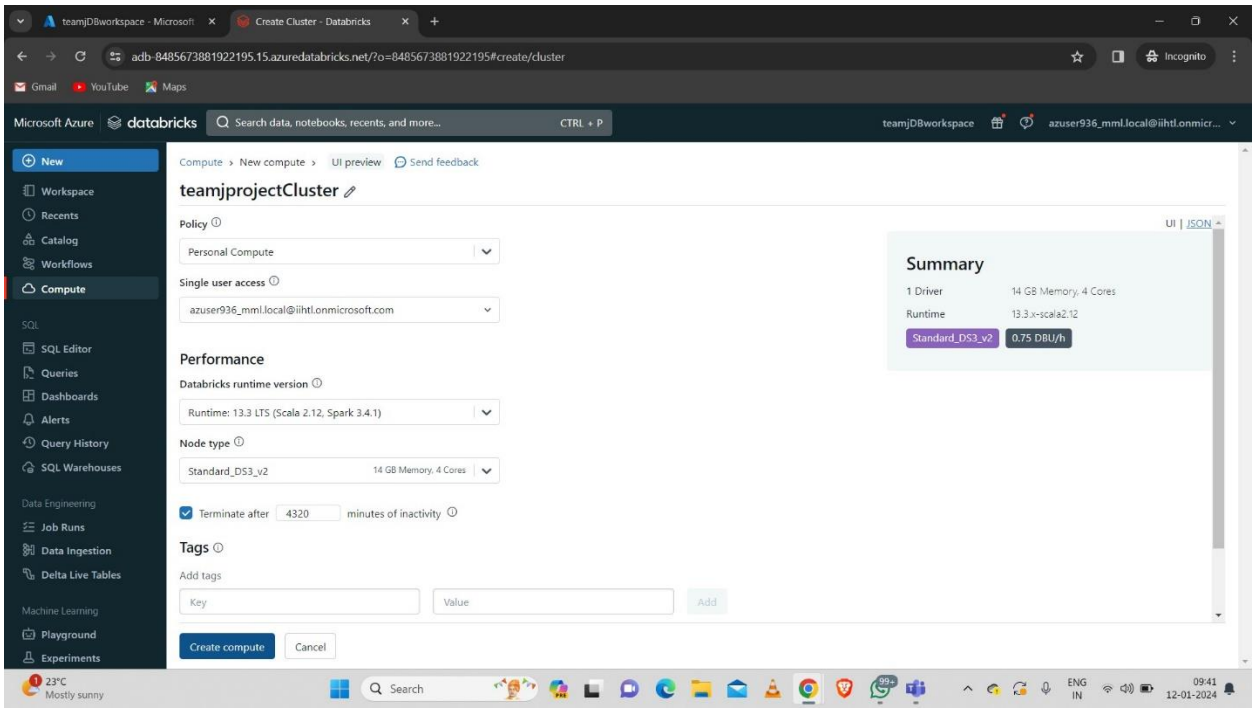
Step-1: Launch the databricks workspace.



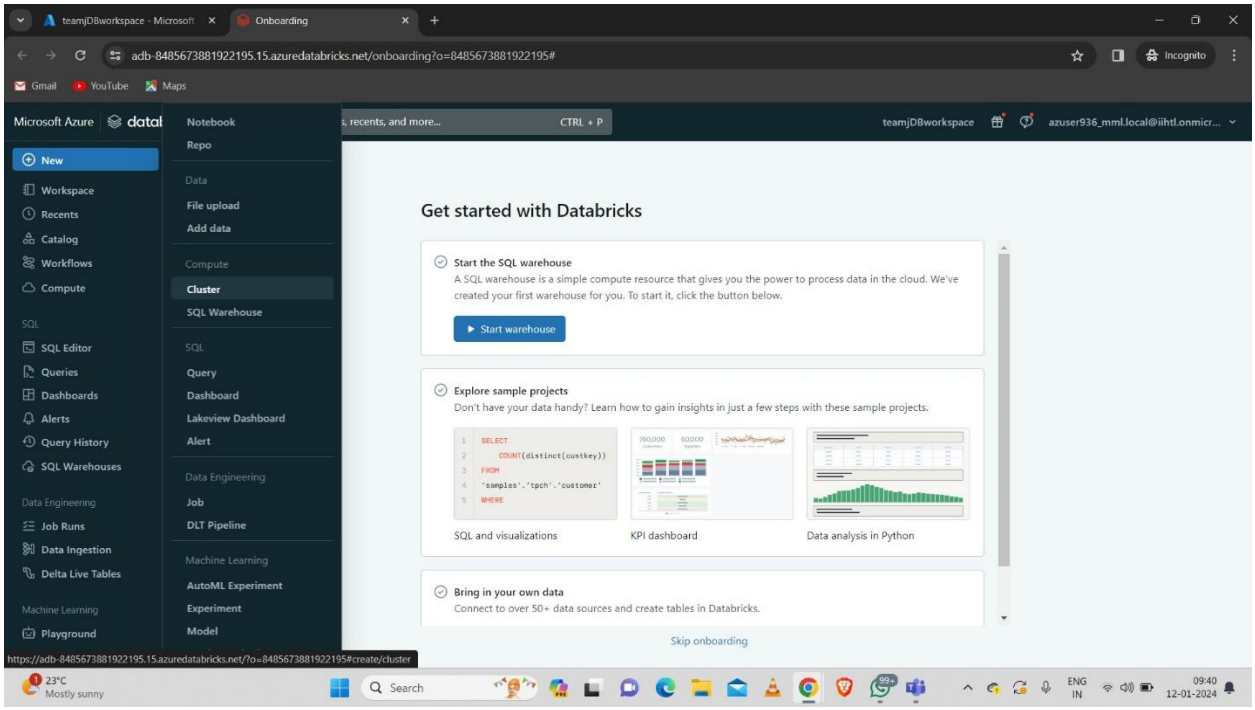
Step-2: After launching the databricks workspace will be opened. Here click on New then click on cluster to create the new cluster.



Step-3: Fill the necessary fields in the cluster then click on create. Here I gave the name of the cluster as “teamjprojectCluster”

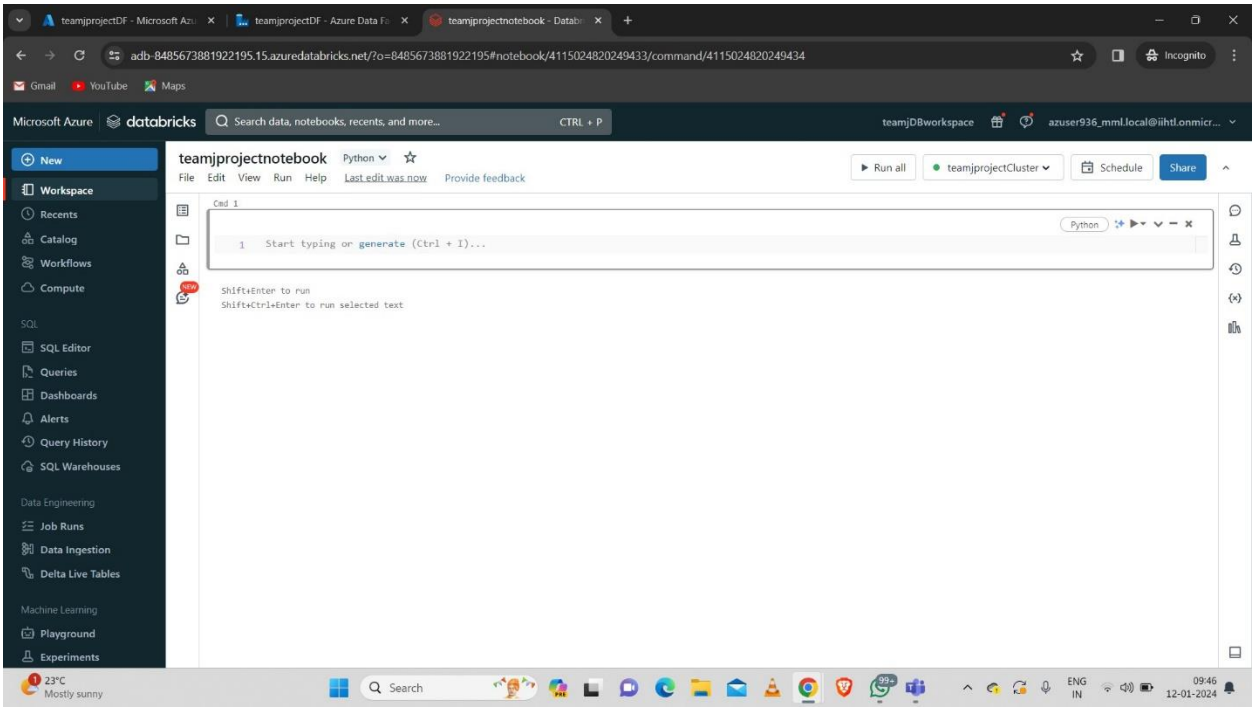


Step-4: Click on new notebook and create the new notebook.



Step-5: Here I gave the notebook name as “teamjprojectnotebook”.

- Here we can see that the cluster is connected to the notebook.

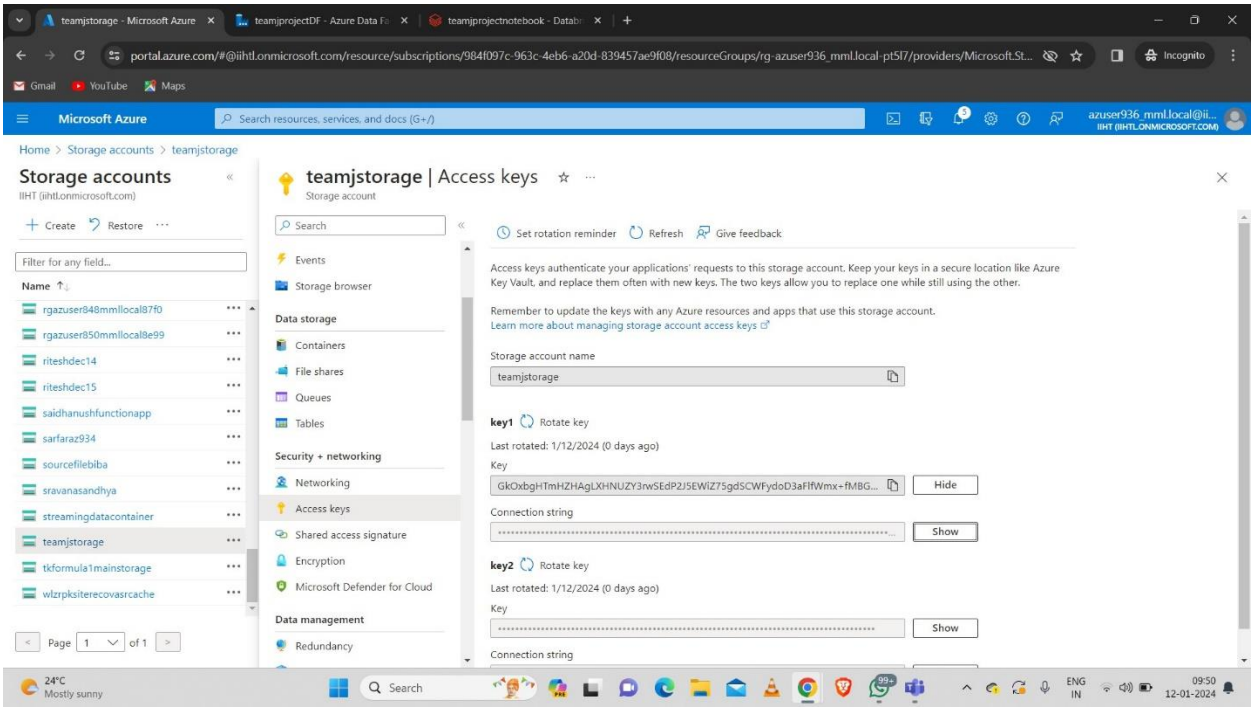


6.6 Mounting ADLS with Databricks

Step-1: Mount (Connect) the blob storage container with databricks with following command.

```
dbutils.fs.mount(source = 'wasbs://<container-name>@<storage-account-name>.blob.core.windows.net',
mount_point = '/mnt/<mount-name>',
extra_configs = {'fs.azure.account.key.<storage-account-name>.blob.core.windows.net':Accountkey'})
```

Step-2: To get the Account key go to the storage account then click on Access keys there we can see a key value copy that and paste at the account key value

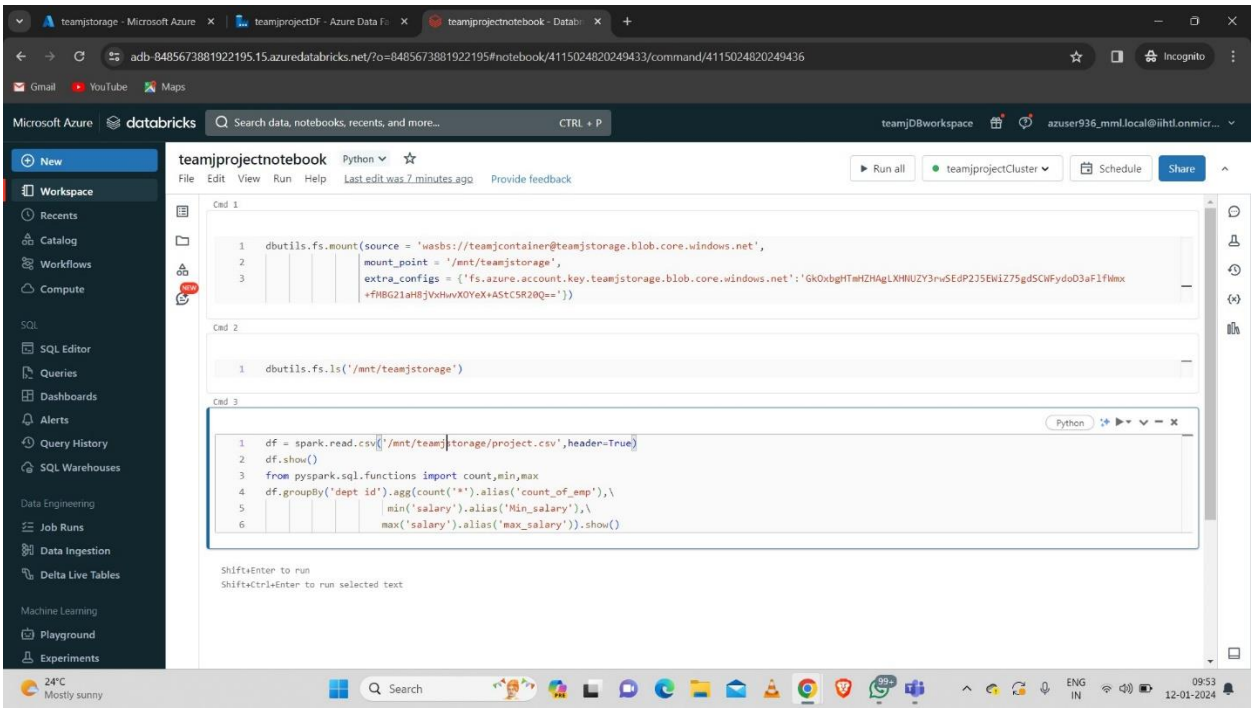


Step-3: Write the code as follows inside the notebook do not run the code as we need to integrate it with ADF.

```
dbutils.fs.mount(source = 'wasbs://teamjcontainer@teamjstorage.blob.core.windows.net',
 mount_point = '/mnt/ teamjstorage ',
extra_configs=
{'fs.azure.account.key.teamjstorage.blob.core.windows.net':'DU/jzGolq9vQx9iWfplx1hgnzhpJJ
UpCQ0I+6s43LQvDtawIMbs094pMDJxuLOJENsYW1u6gTs0H+AStq1fEwA==''})
```

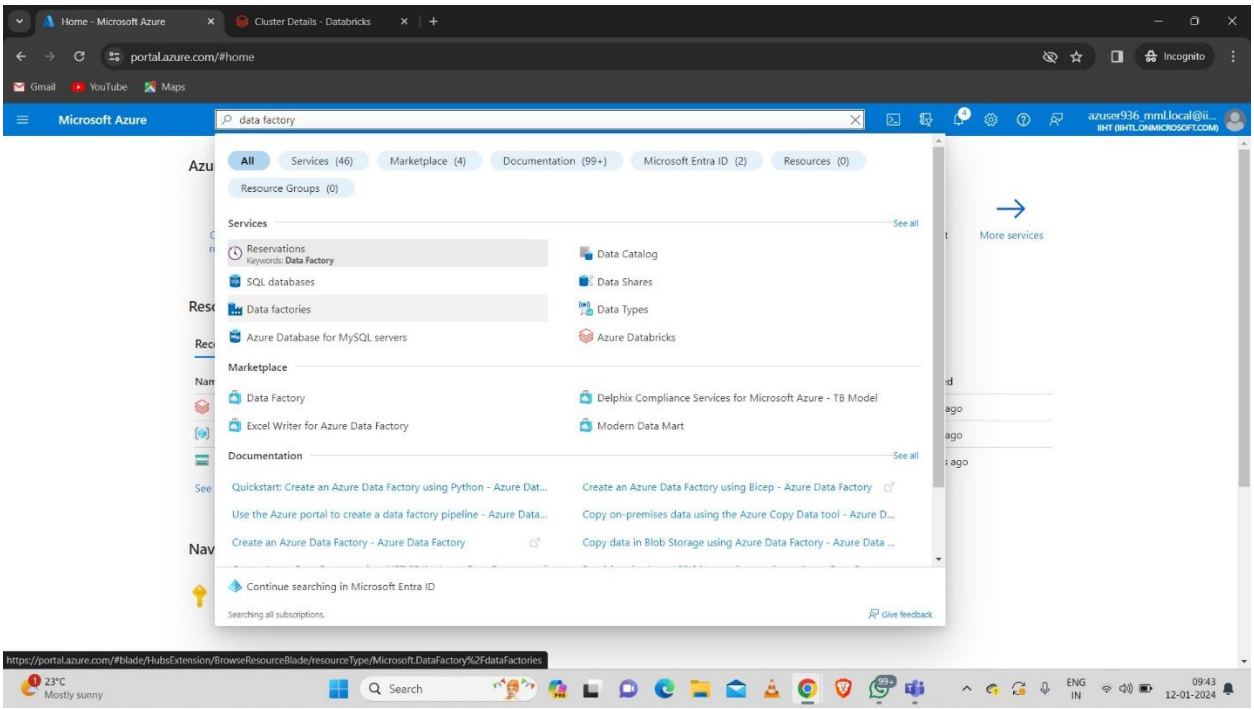
```
dbutils.fs.ls('/mnt/ teamjstorage ')
```

```
df = spark.read.csv('/mnt/ teamjstorage /project.csv',header=True)
df.show()
from pyspark.sql.functions import count,min,max
df.groupBy('dept id').agg(count('*').alias('count_of_emp'),\
                  min('salary').alias('Min_salary'),\
                 max('salary').alias('max_salary')).show()
```

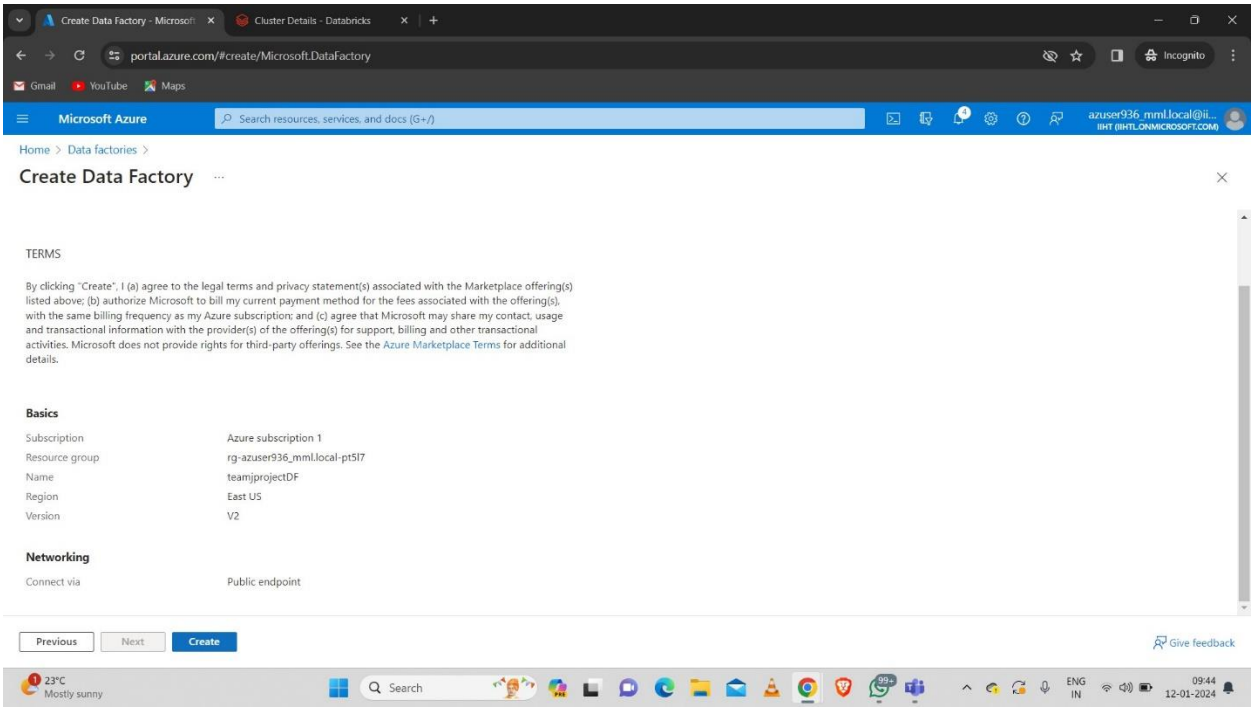


6.7 Azure Data Factory

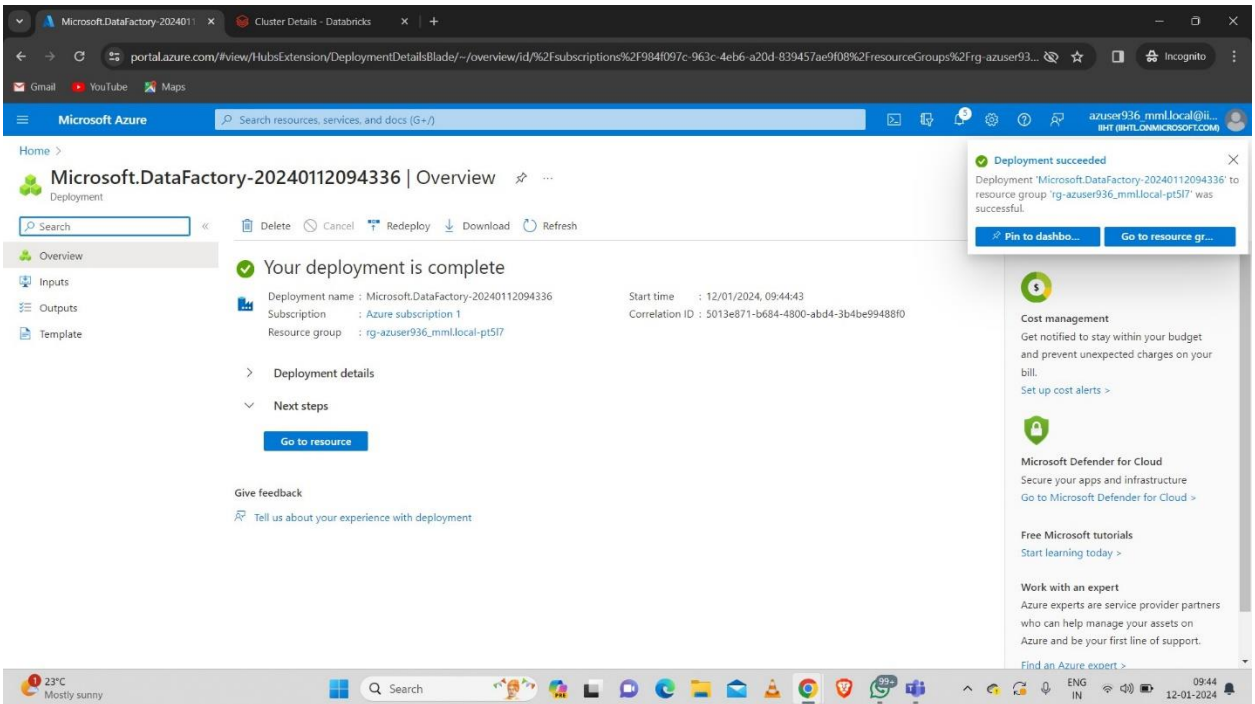
Step-1: Go to search bar and search for Data Factories



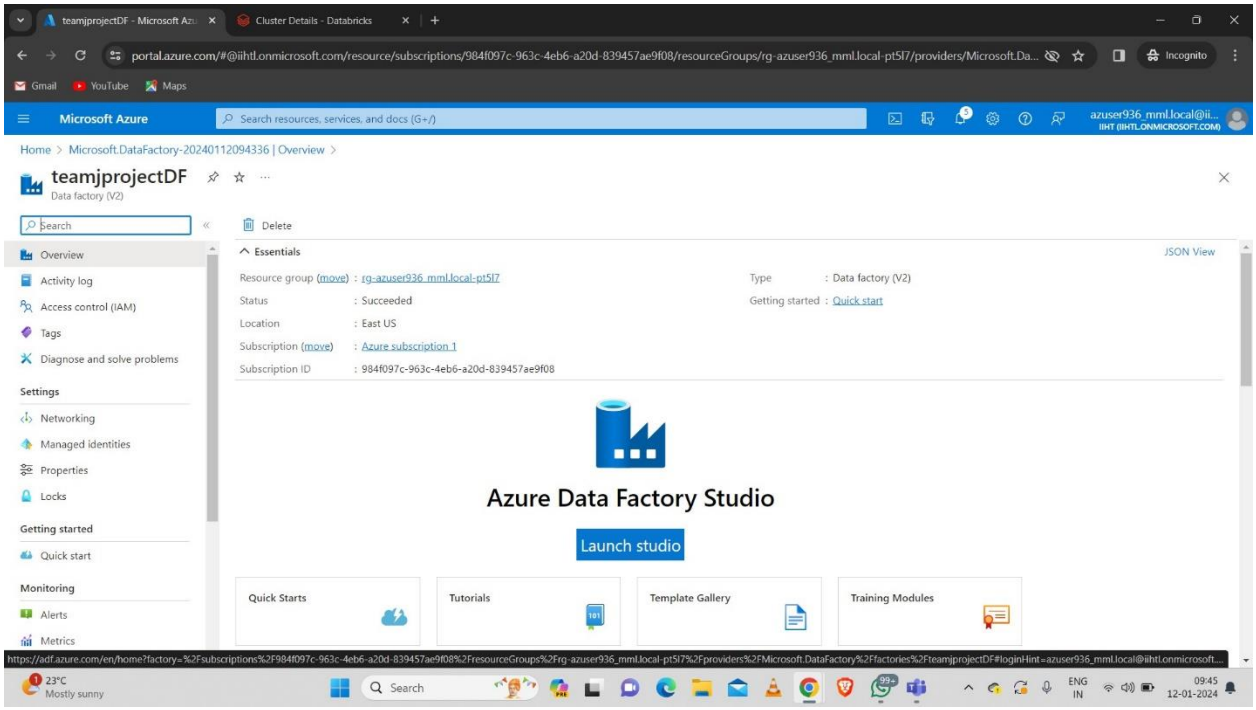
Step-2: Fill the necessary data and create the data factory. Here I gave the data factory name as "teamjprojectDF".



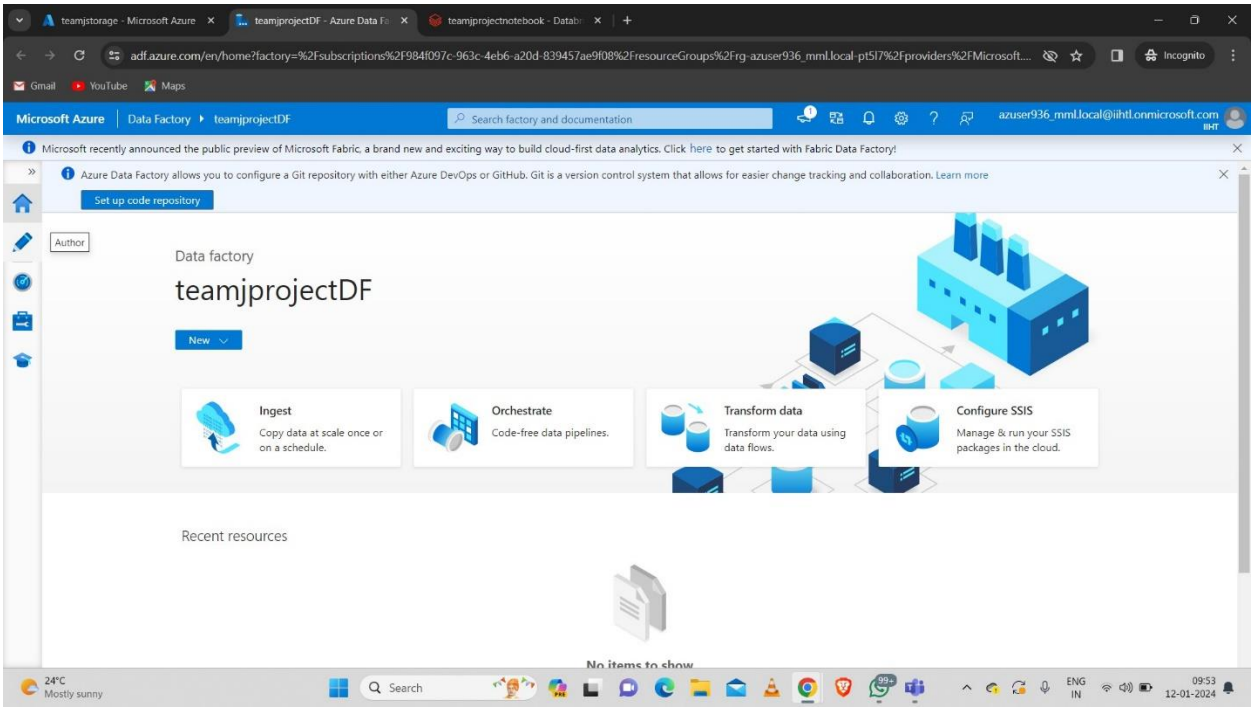
Step-3: Here we can see that the data factory is successfully created.



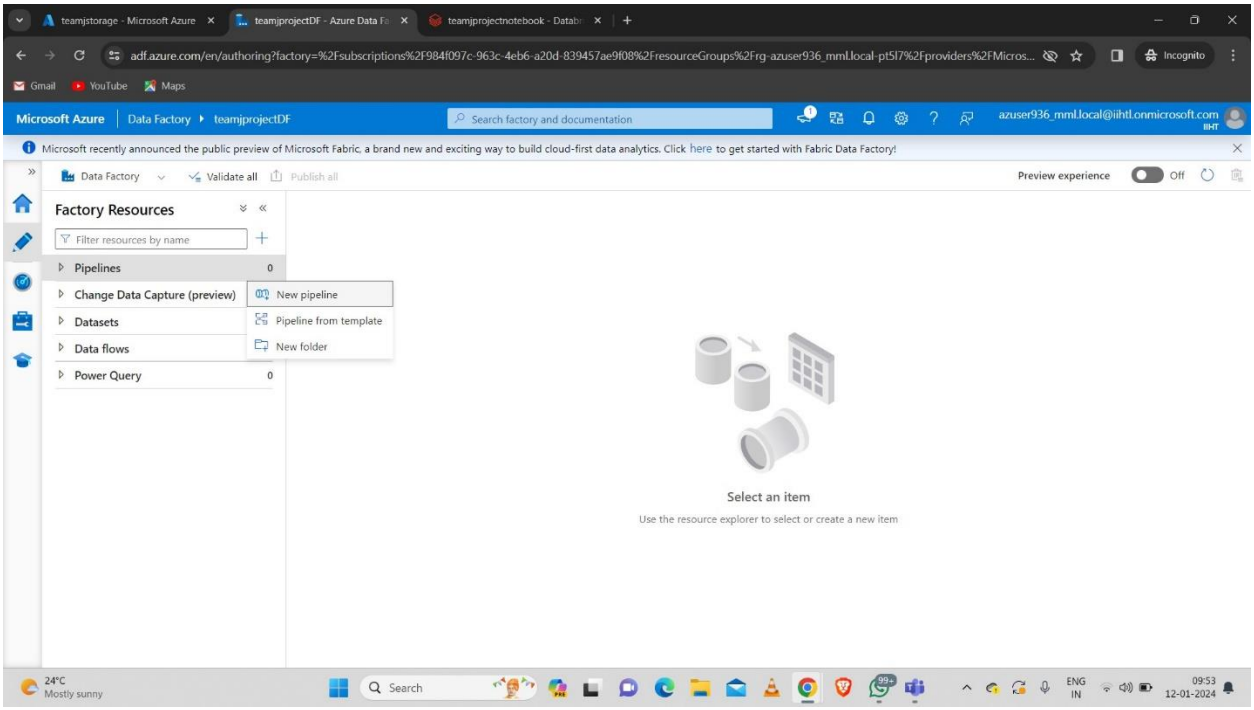
Step-4: Launch the Azure Data Factory studio.



Step-5: After launching we get the following page.



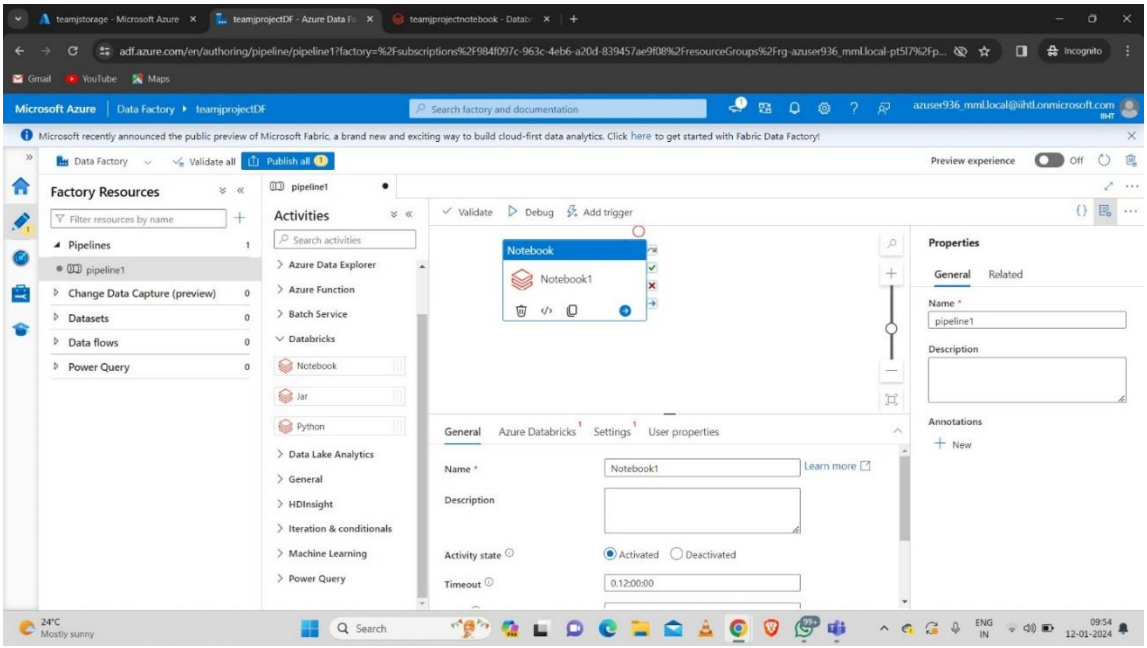
- Now click on author to get the following page
- Click on new pipeline to create one.



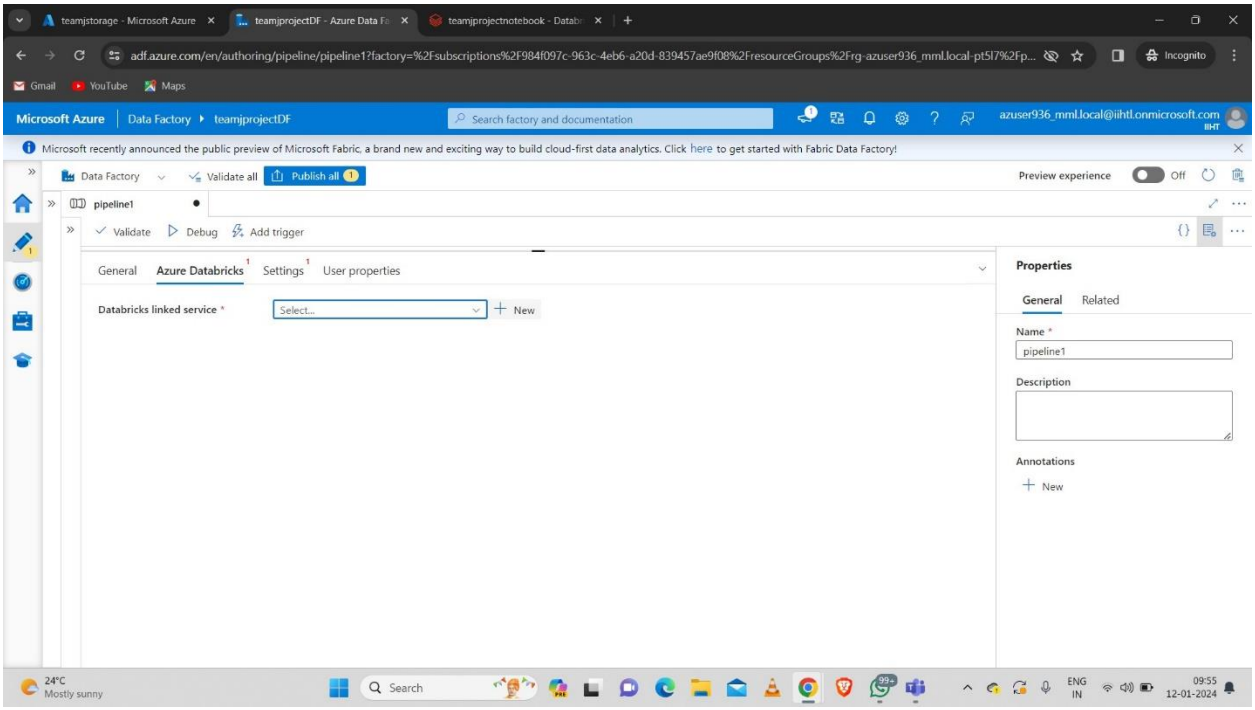
6.8 Integrating ADF with Azure Databricks

Step-1: Create a new pipeline.

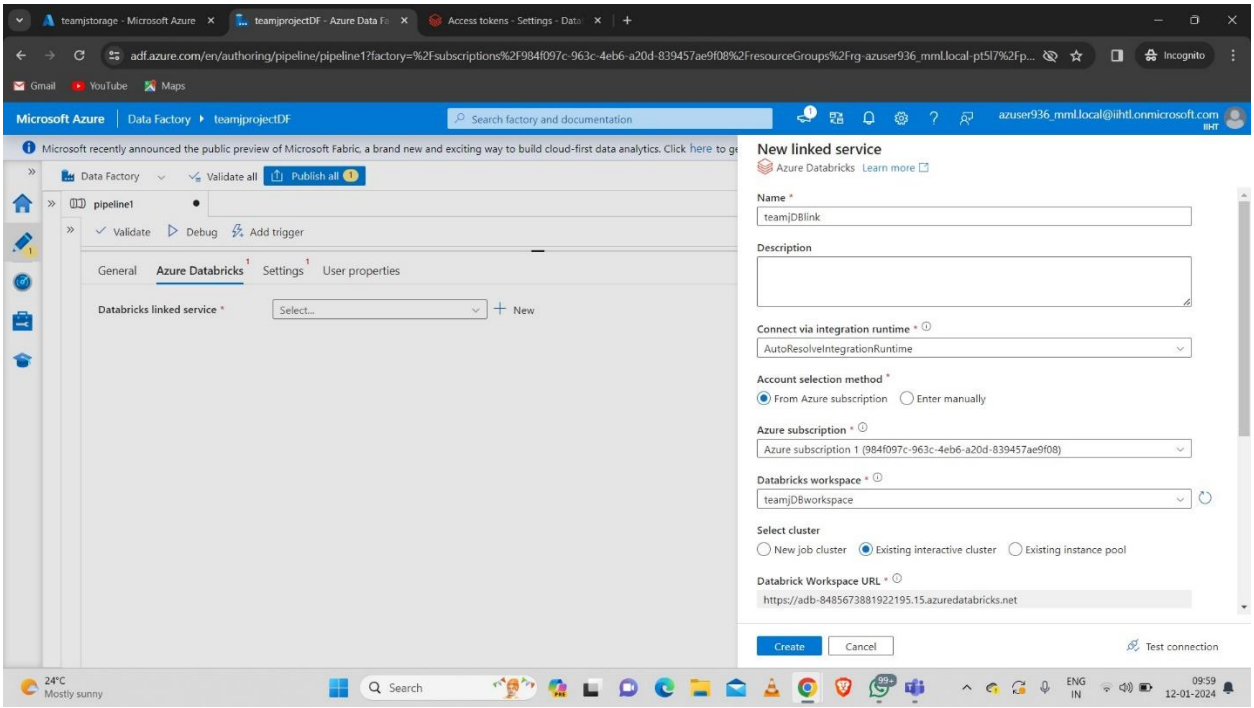
- Now go to Databricks in Activities bar and drag the Notebook.



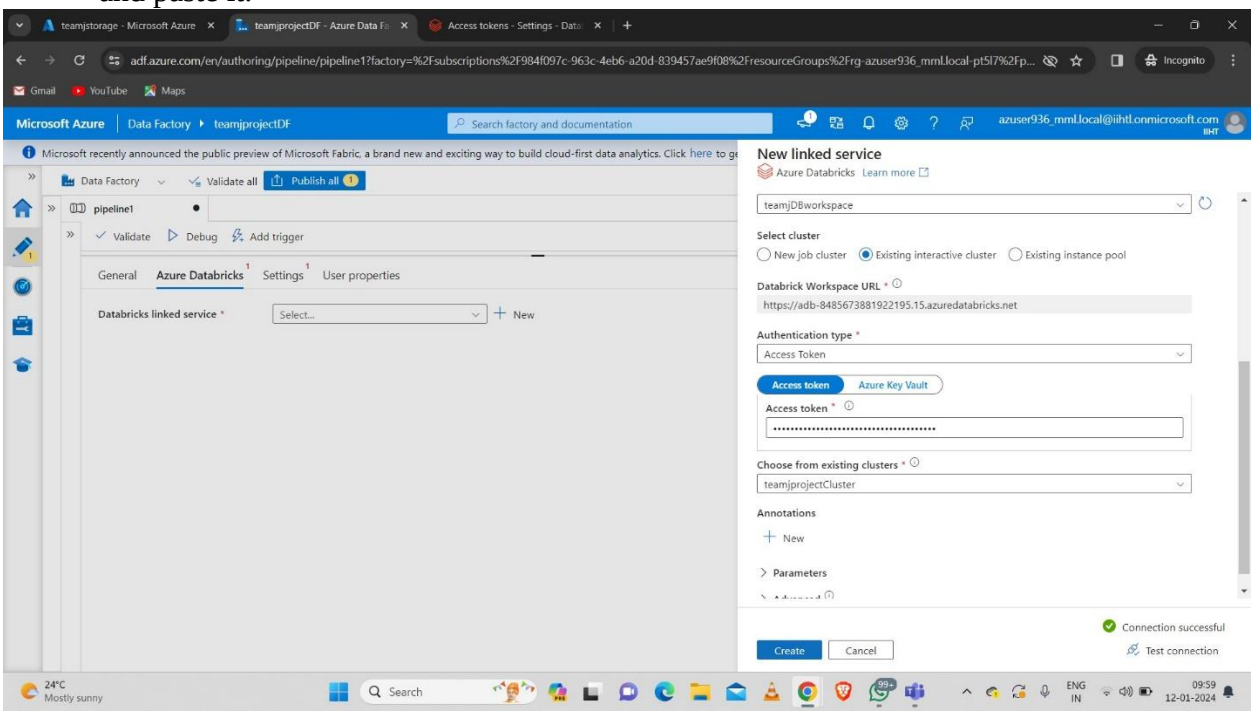
Step-2: Now click on new to link the databricks with this pipeline.



- Fill the necessary data.Select the Existing interactive cluster that we already created to run the notebook

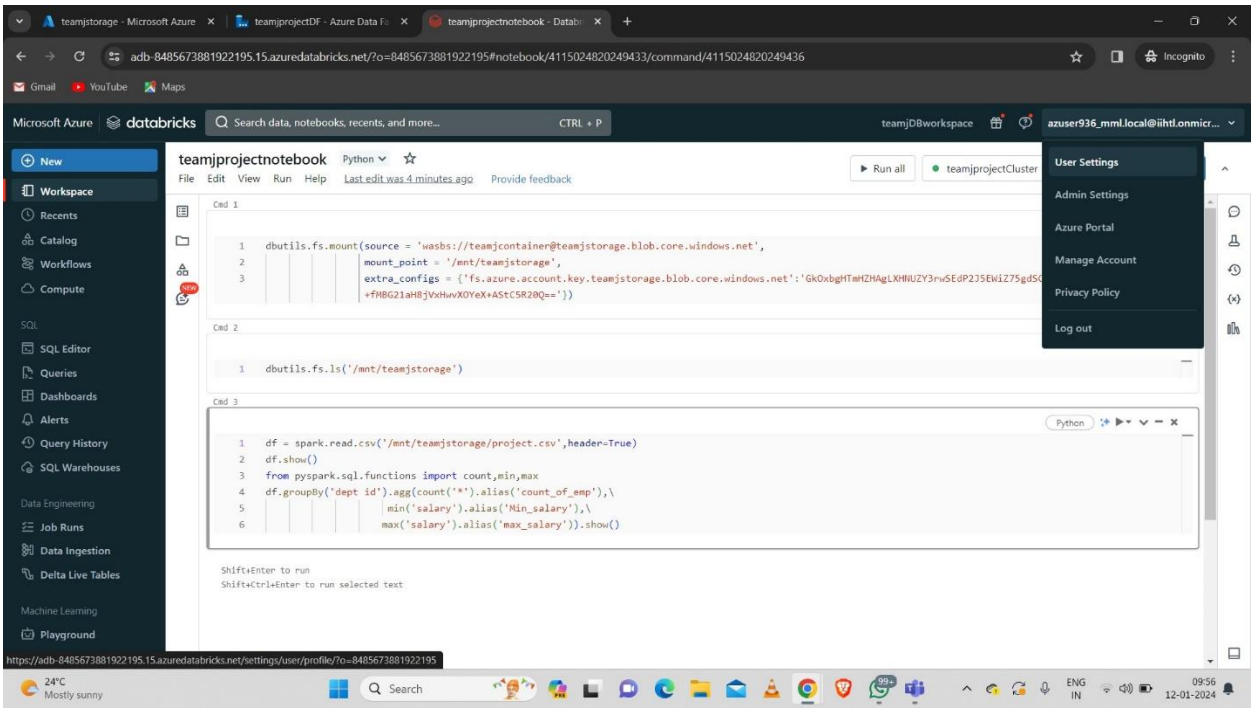


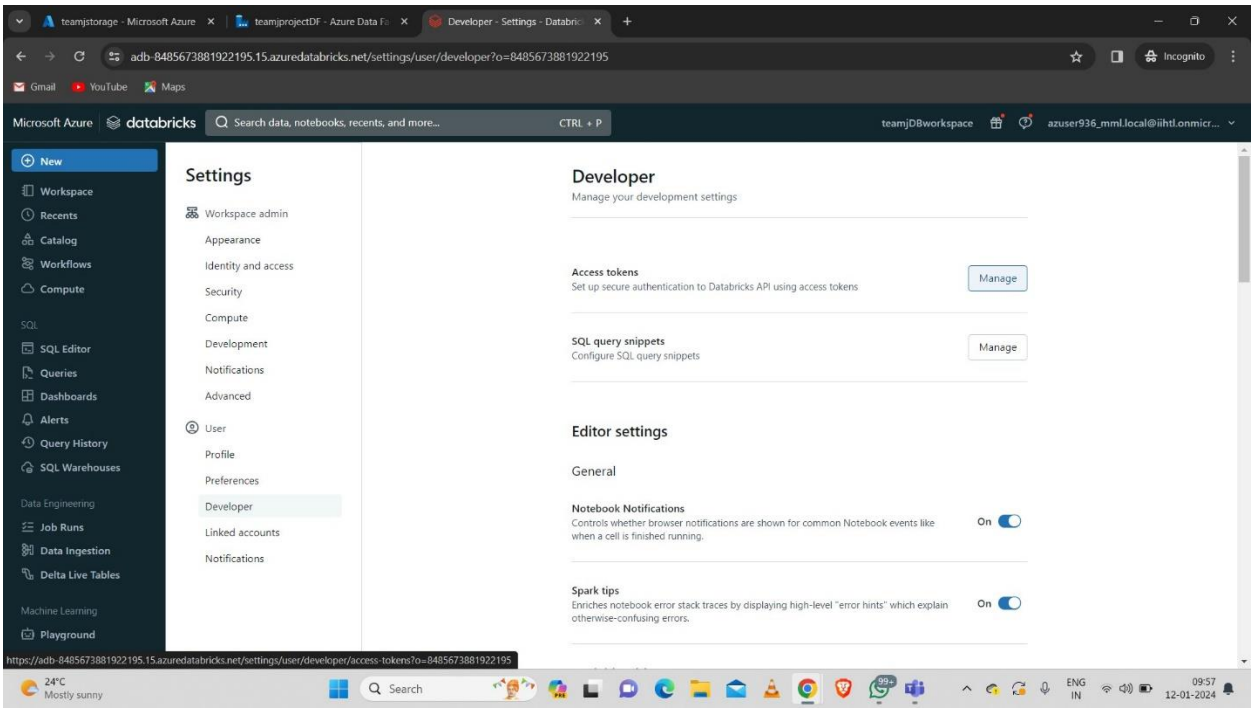
- Choose the Authentication type as Access token and copy the access token from databricks and paste it.

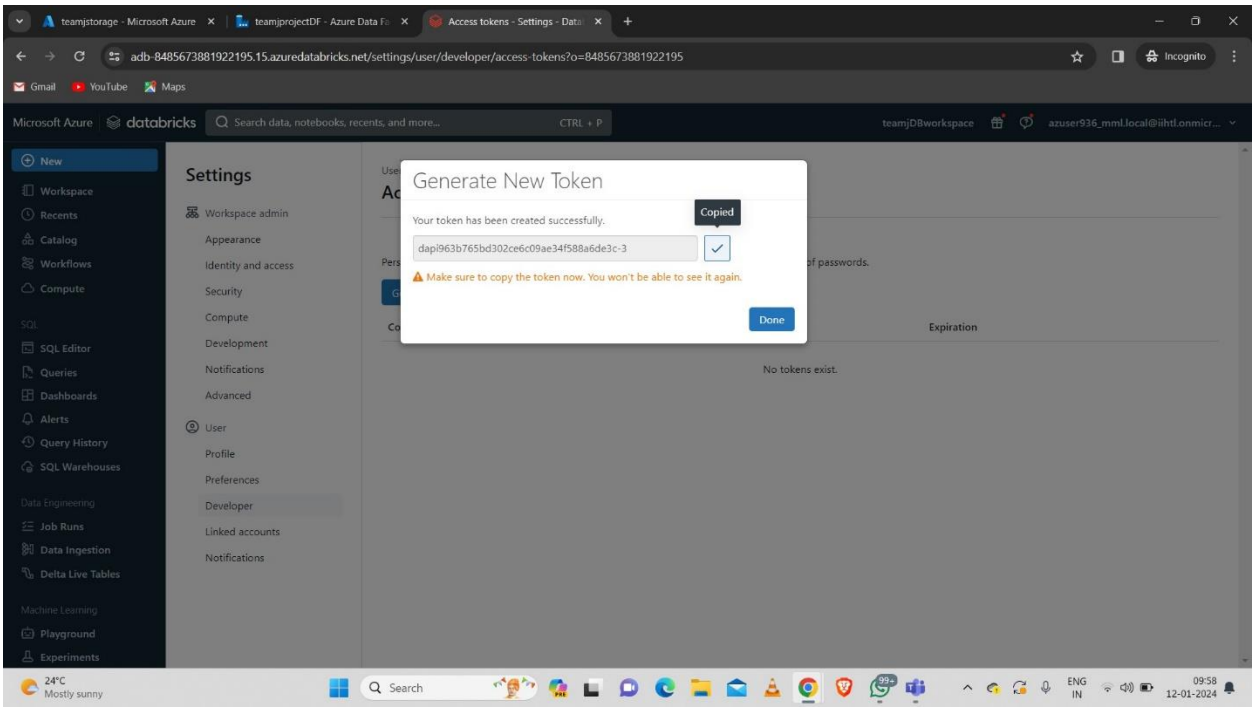


Access token:

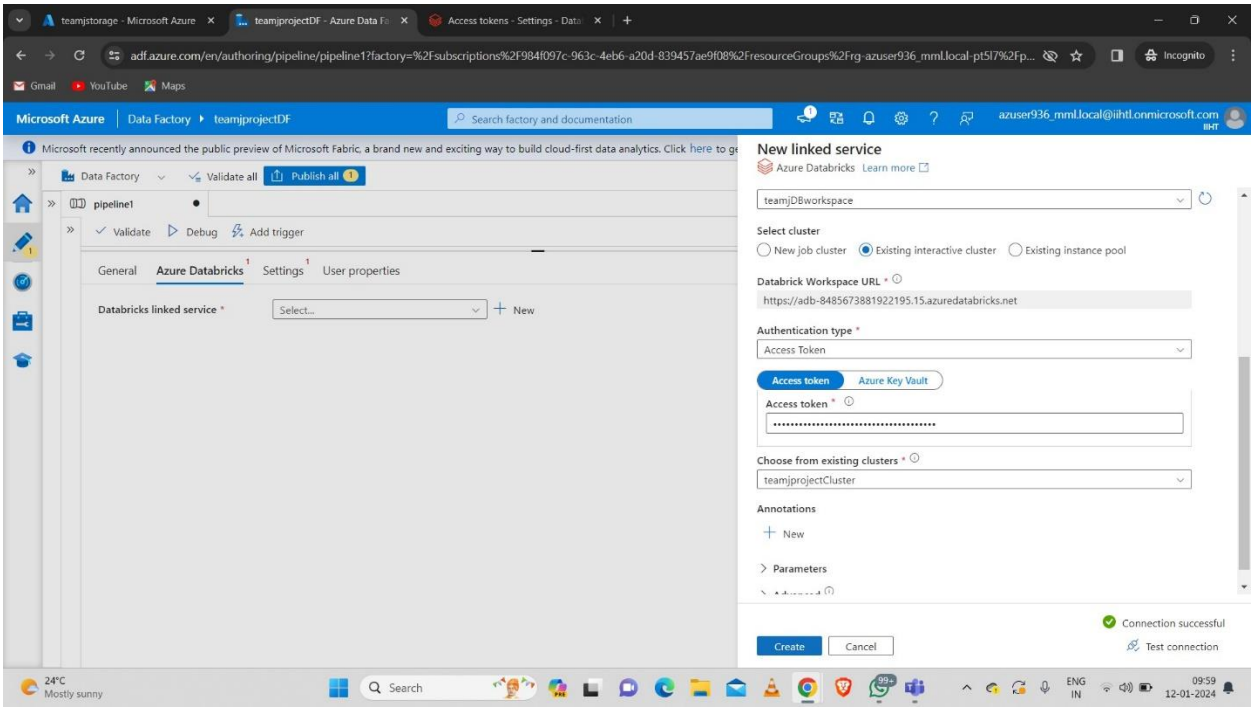
1. Go to Azure Databricks workspace.
2. Go to User Settings.
3. Go to Developer path.
4. Click on Manage Access token.
5. Generate the new Access token and copy it and paste in the ADF.


teamjprojectnotebook
dbutils.fs.mount(source = 'wasbs://teamjcontainer@teamjstorage.blob.core.windows.net',
mount_point = '/mnt/teamjstorage',
dbutils.fs.ls('/mnt/teamjstorage')
df = spark.read.csv('/mnt/teamjstorage/project.csv',header=True)
df.show()
from pyspark.sql.functions import count,min,max
df.groupBy('dept id').agg(count('*').alias('count_of_emp'),\
min('salary').alias('Min_salary'),\
max('salary').alias('max_salary')).show()
Shift+Enter to run
Shift+Ctrl+Enter to run selected text

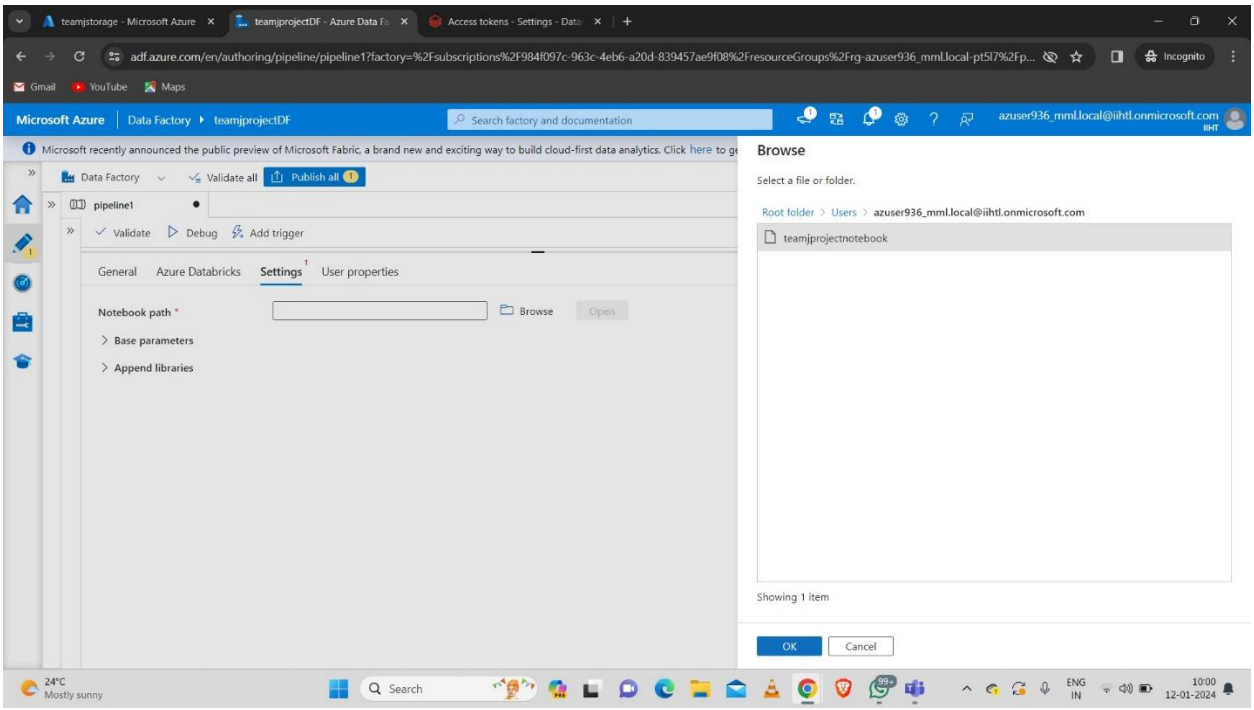

Settings
Developer
Manage your development settings
Access tokens
Set up secure authentication to Databricks API using access tokens
Manage
SQL query snippets
Configure SQL query snippets
Manage
Editor settings
General
Notebook Notifications
Controls whether browser notifications are shown for common Notebook events like when a cell is finished running.
On
Spark tips
Enriches notebook error stack traces by displaying high-level "error hints" which explain otherwise-confusing errors.
On



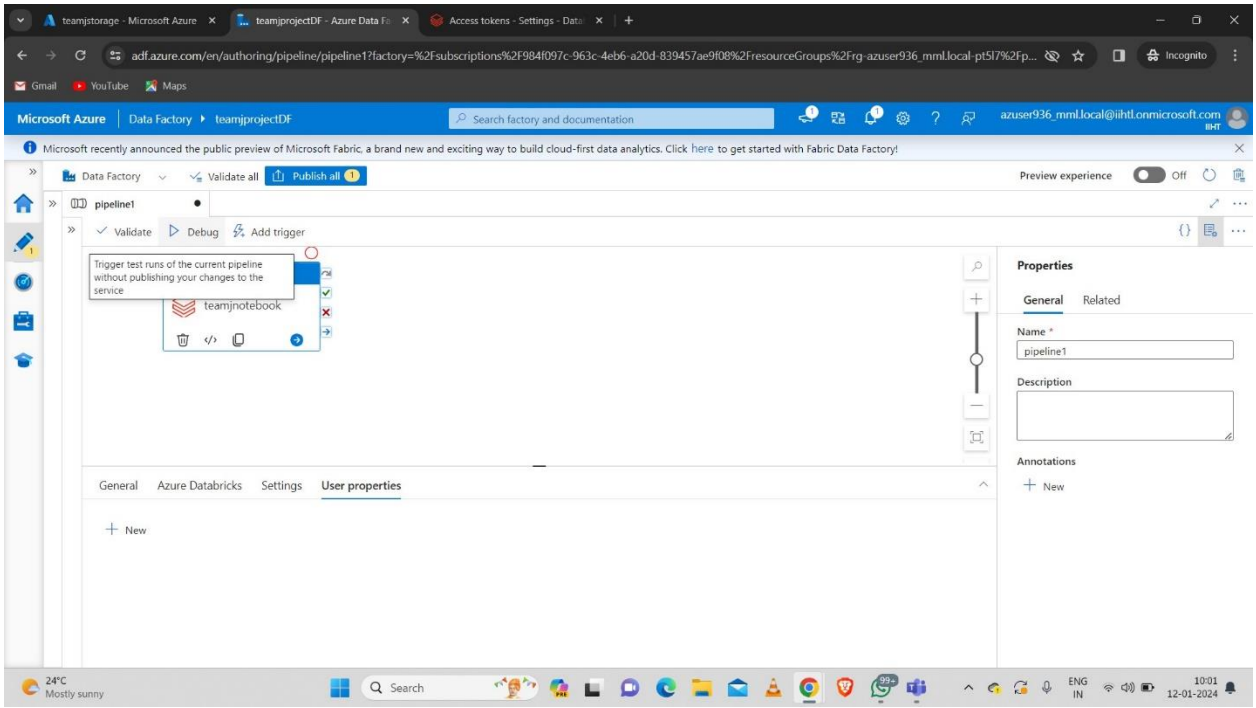
- Now create the databricks link service.



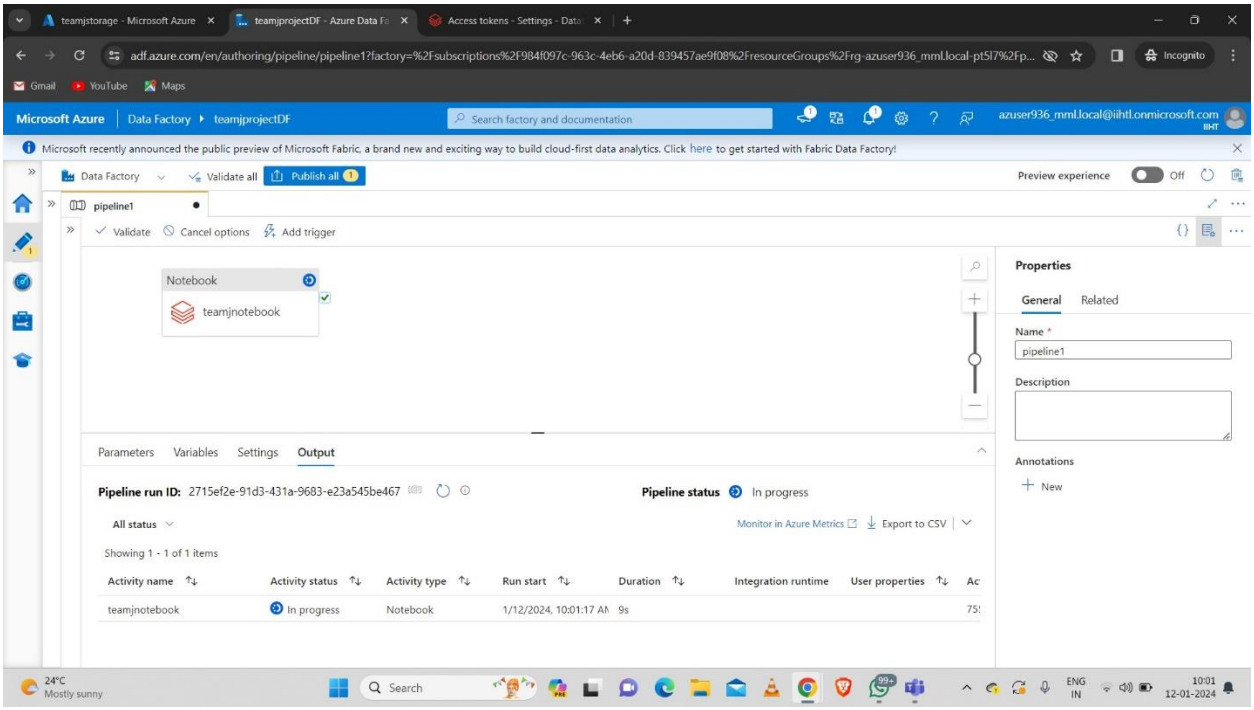
Step-3: Now give the path for the notebook by browsing it and giving the notebook path that we created as “teamjprojectnotebook”.



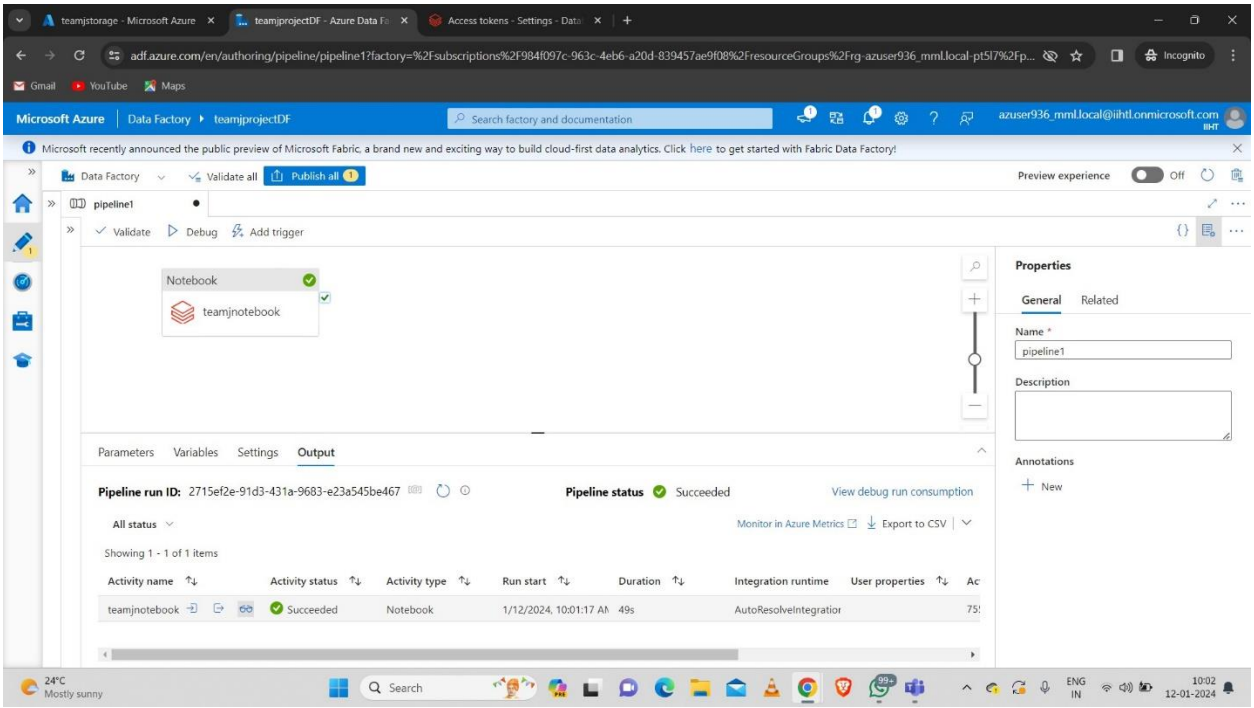
Step-4: After completing the above steps debug the pipeline.



- We can see the activity status of the pipeline as follows.

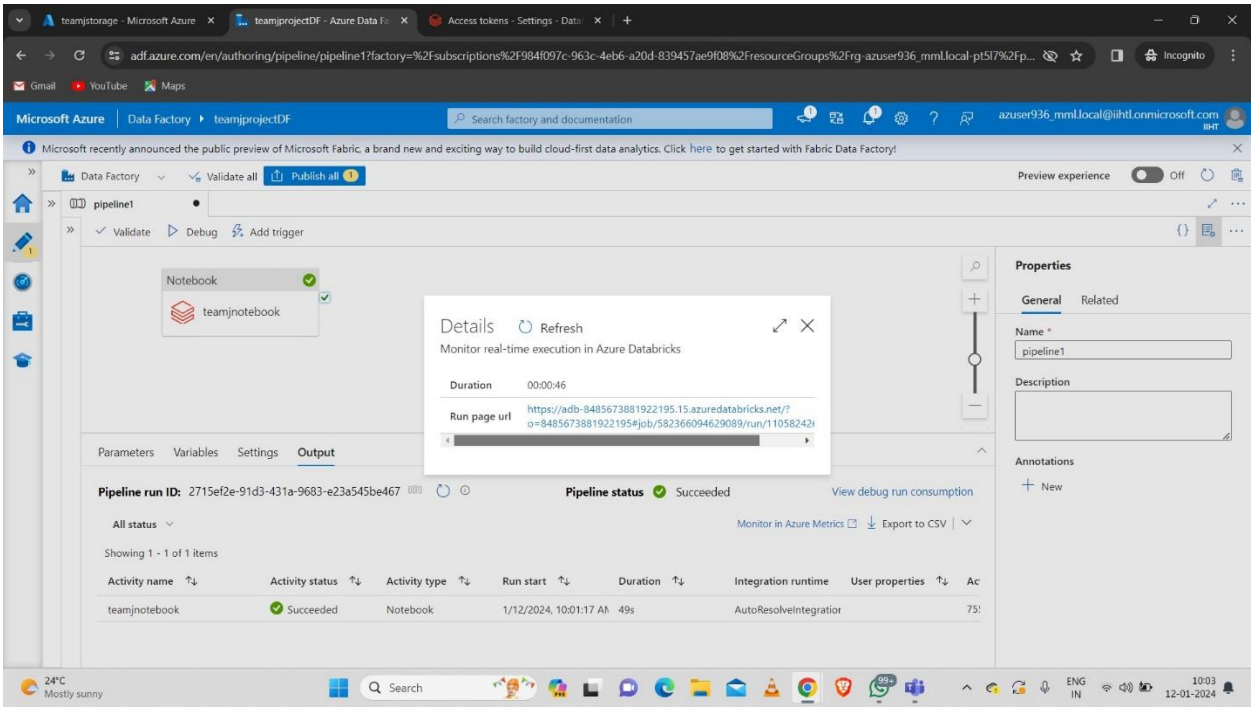


- We can see that the pipeline is debugged successfully.

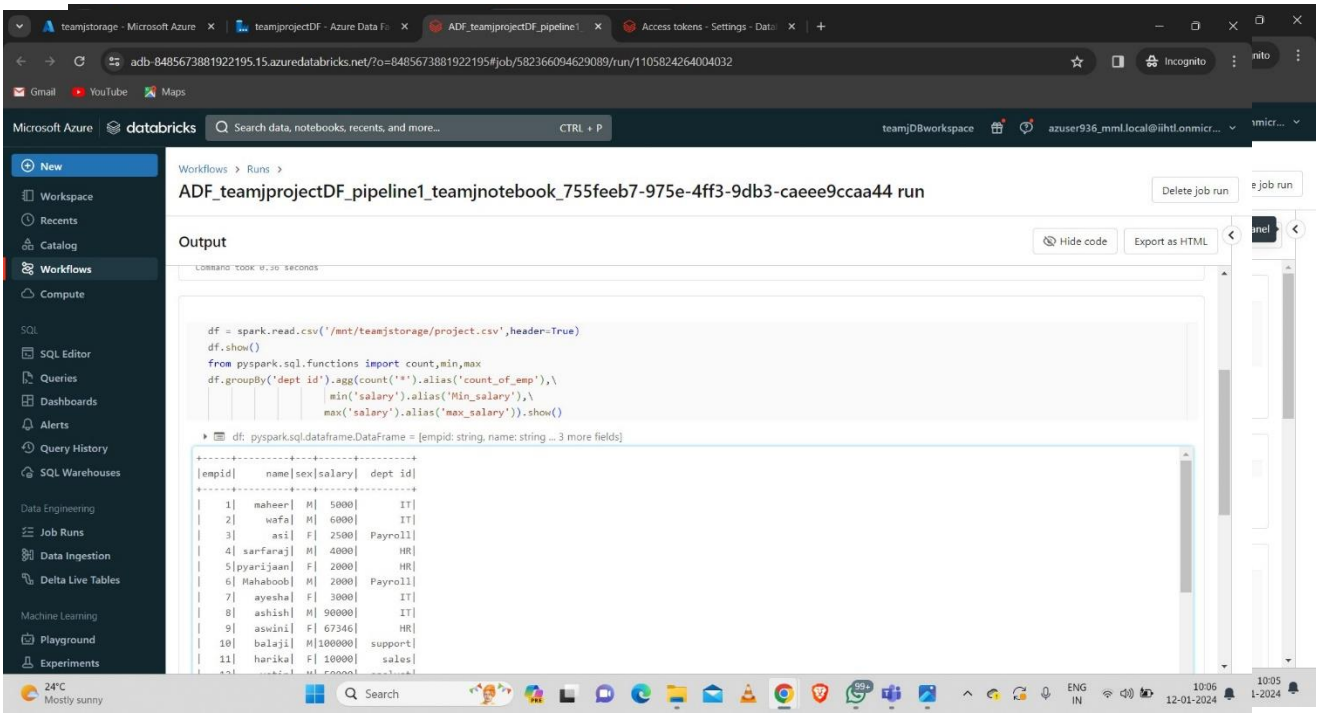


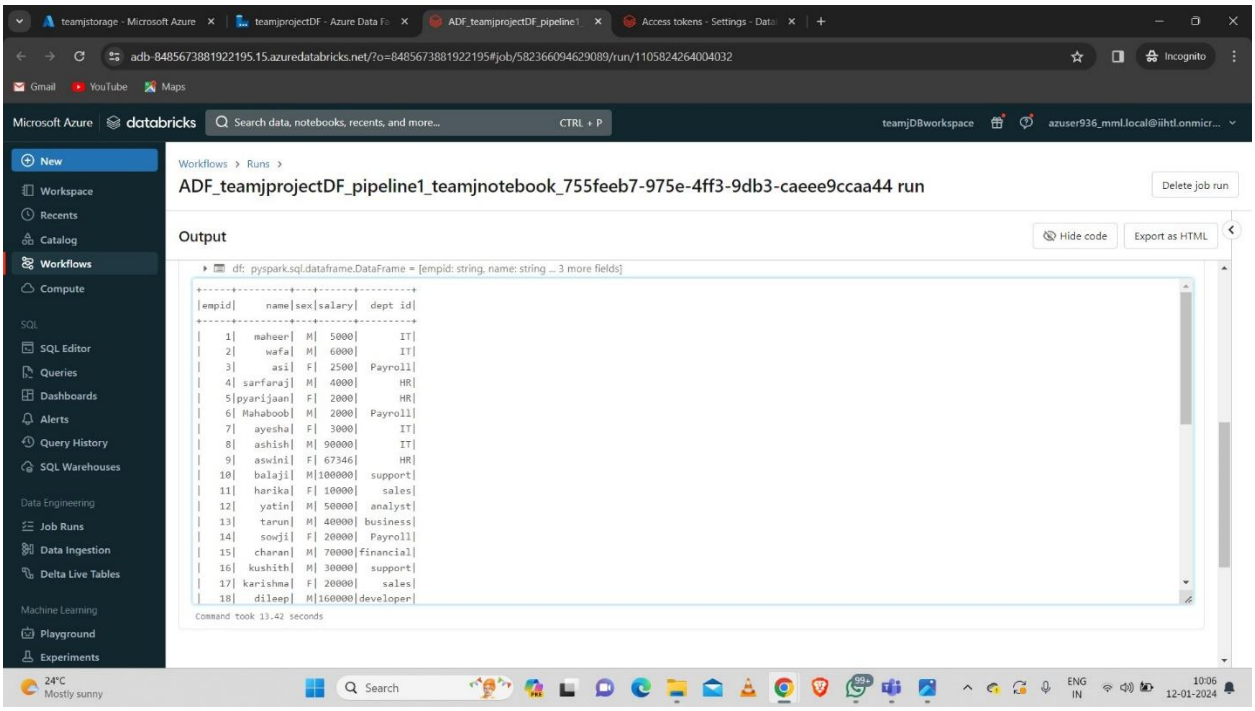
6.9 Final Output

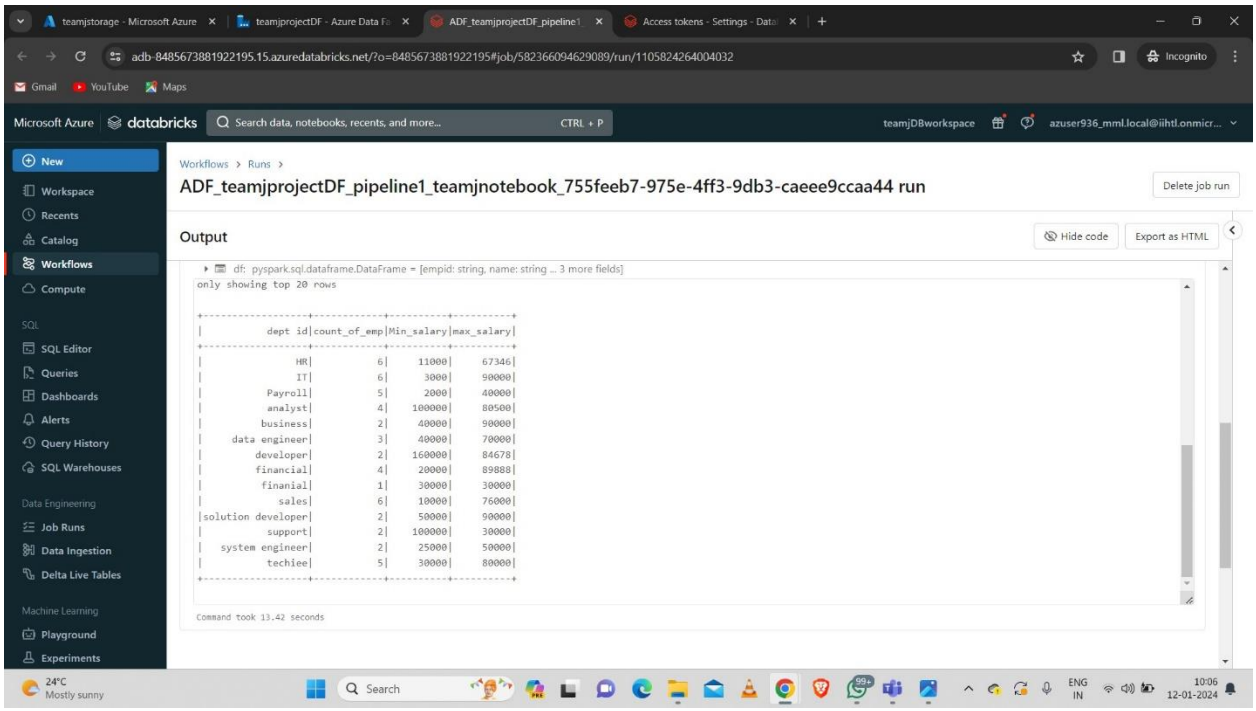
- Click on details at the output of the pipeline after debugging.
- Click on the url.



- After clicking on the above url we get the output of the notebook which we created as "teamjprojectnotebook".



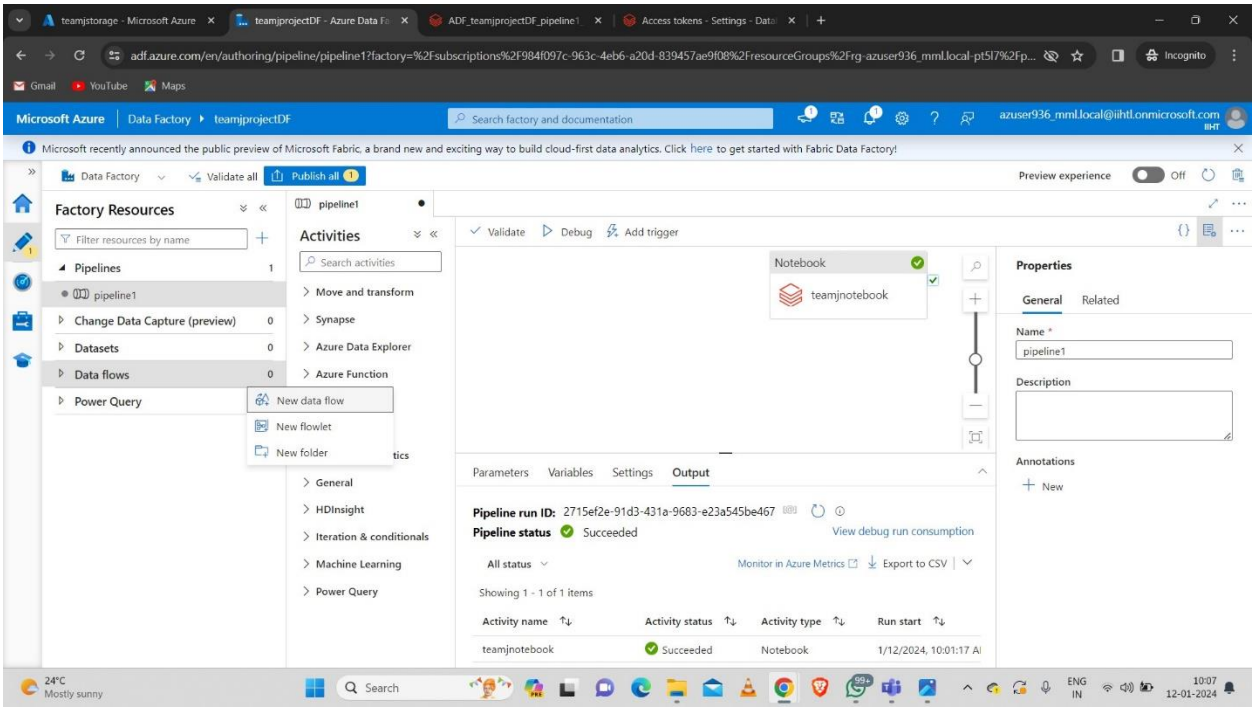




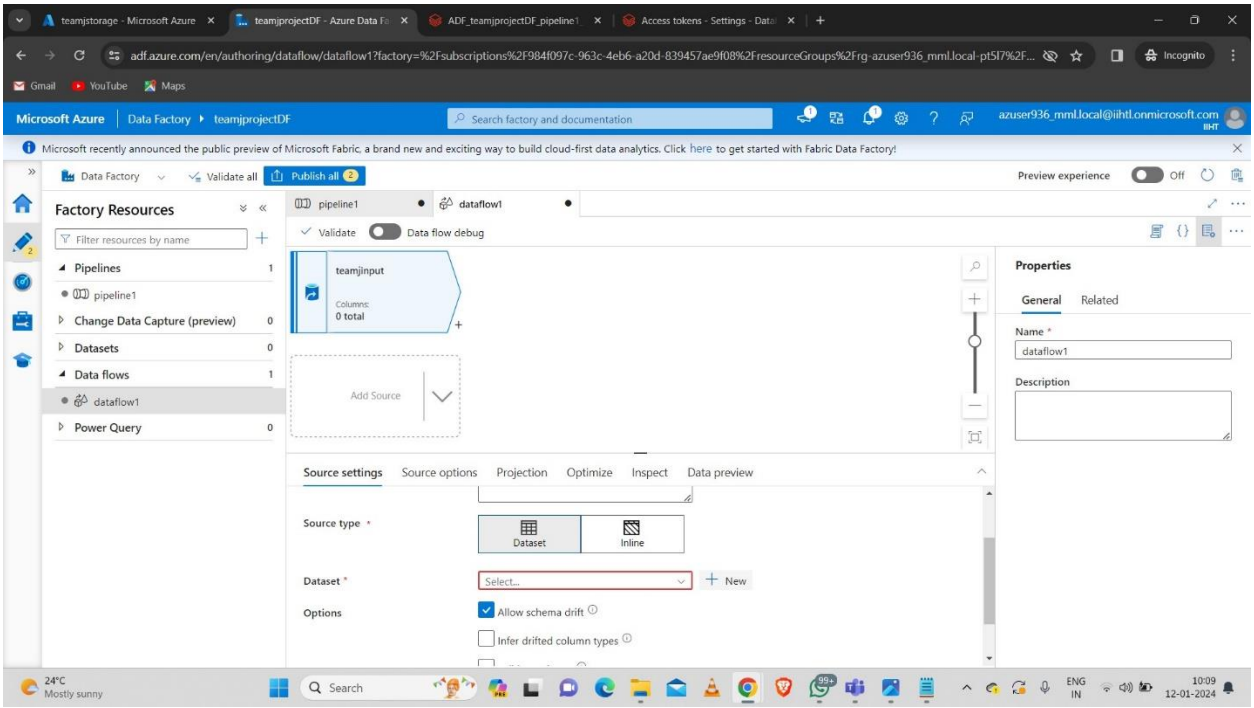
Hence we successfully implemented a daily data aggregation pipeline using Azure Data Factory to move raw data and Azure Databricks to aggregate and summarize the data based on daily intervals.

7.Copying the Aggregate data using Data Flow Activity

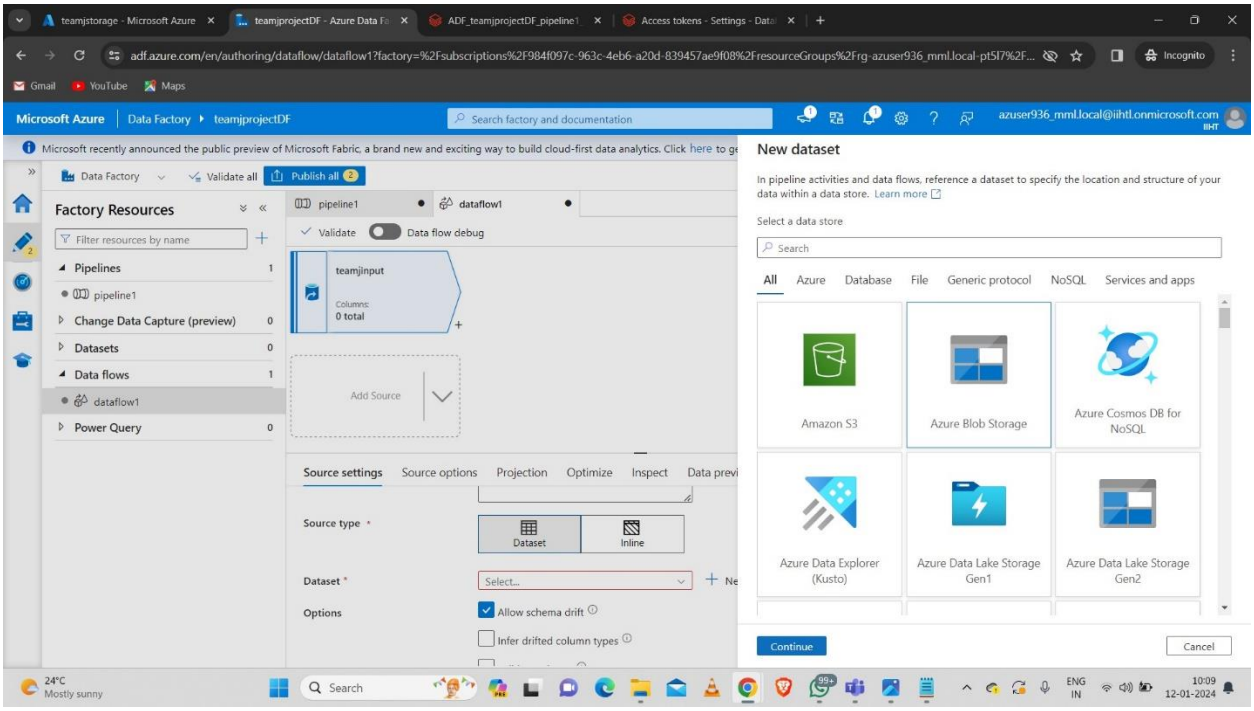
Step-1: Create a new data flow.



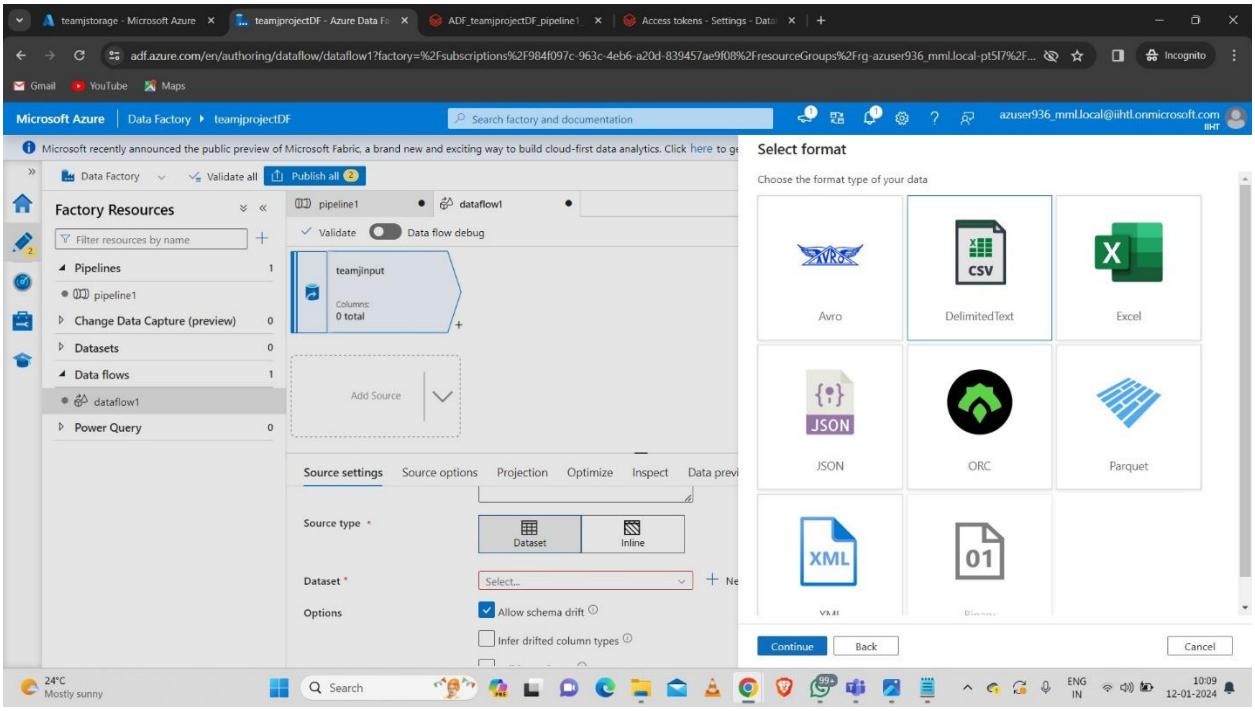
Step-2: Now inside the source settings link the new dataset.



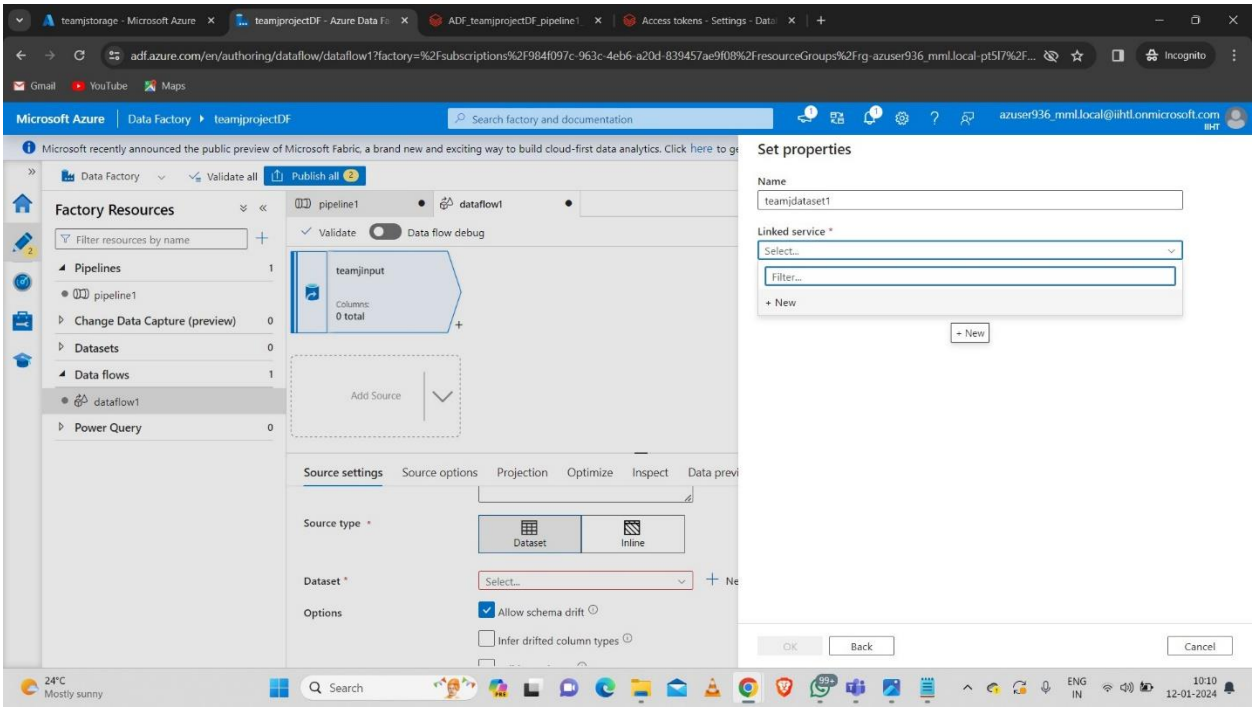
- Select the type of storage.



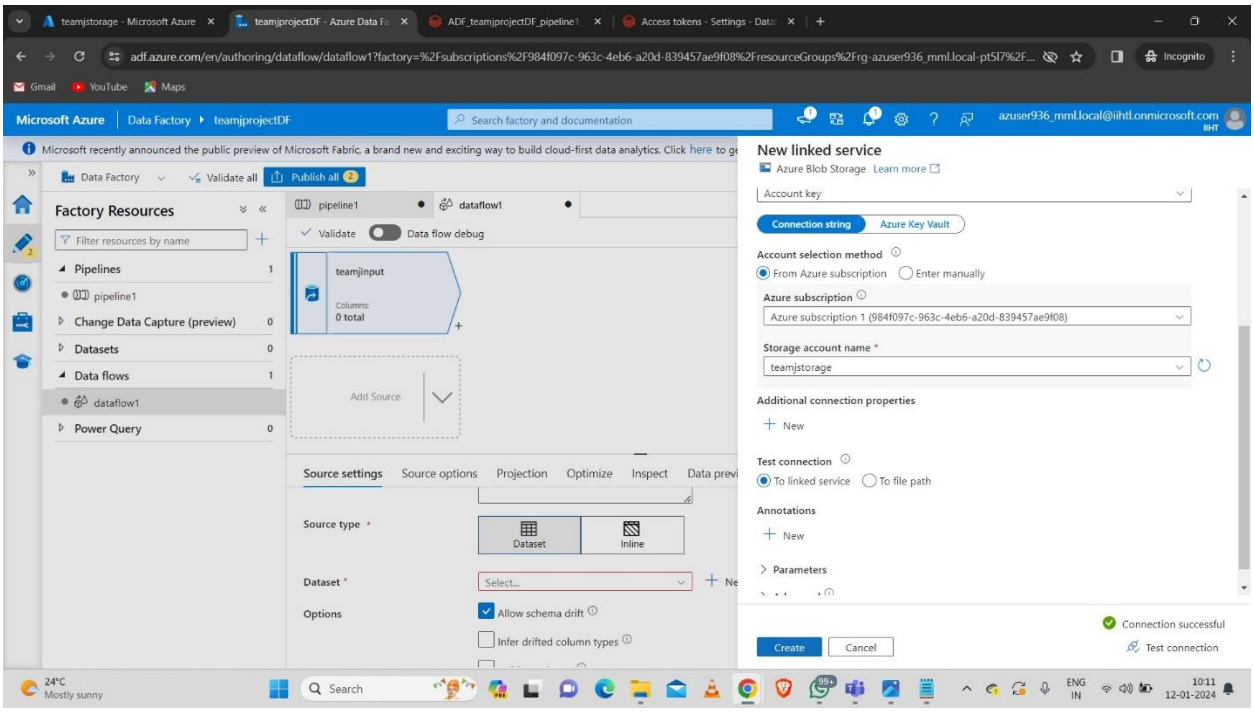
- Select the file type as CSV.



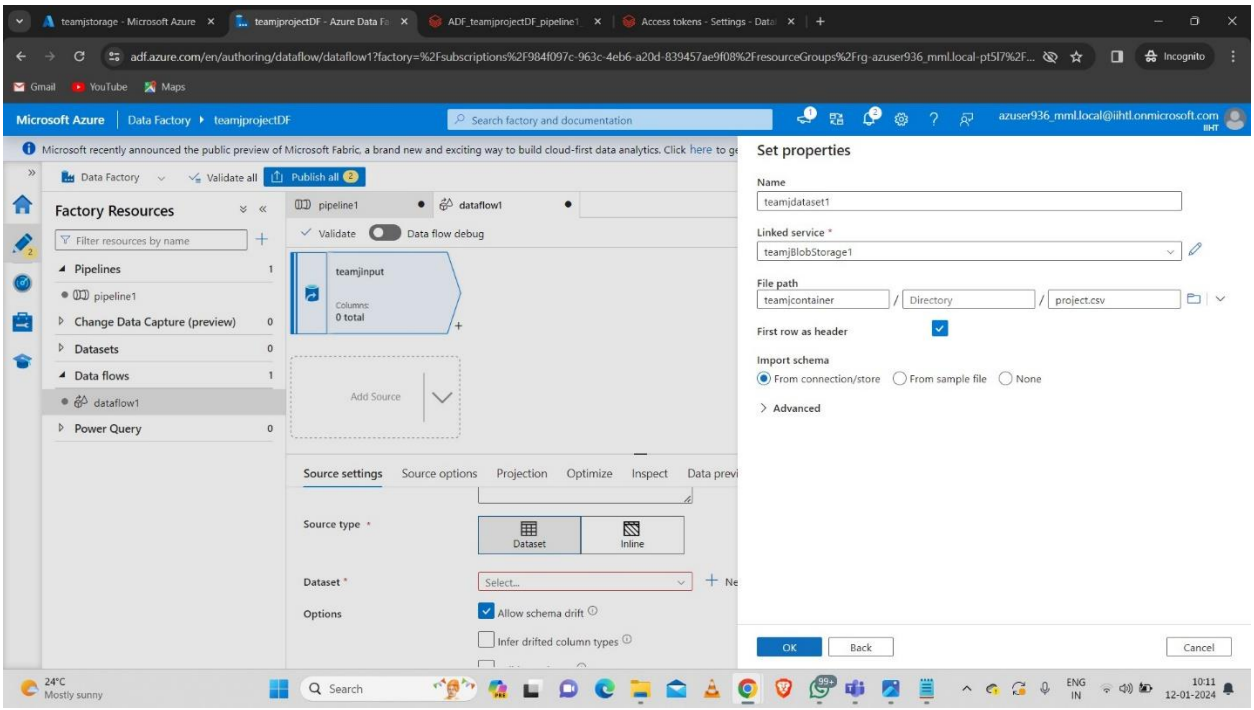
- Select the new linked service.



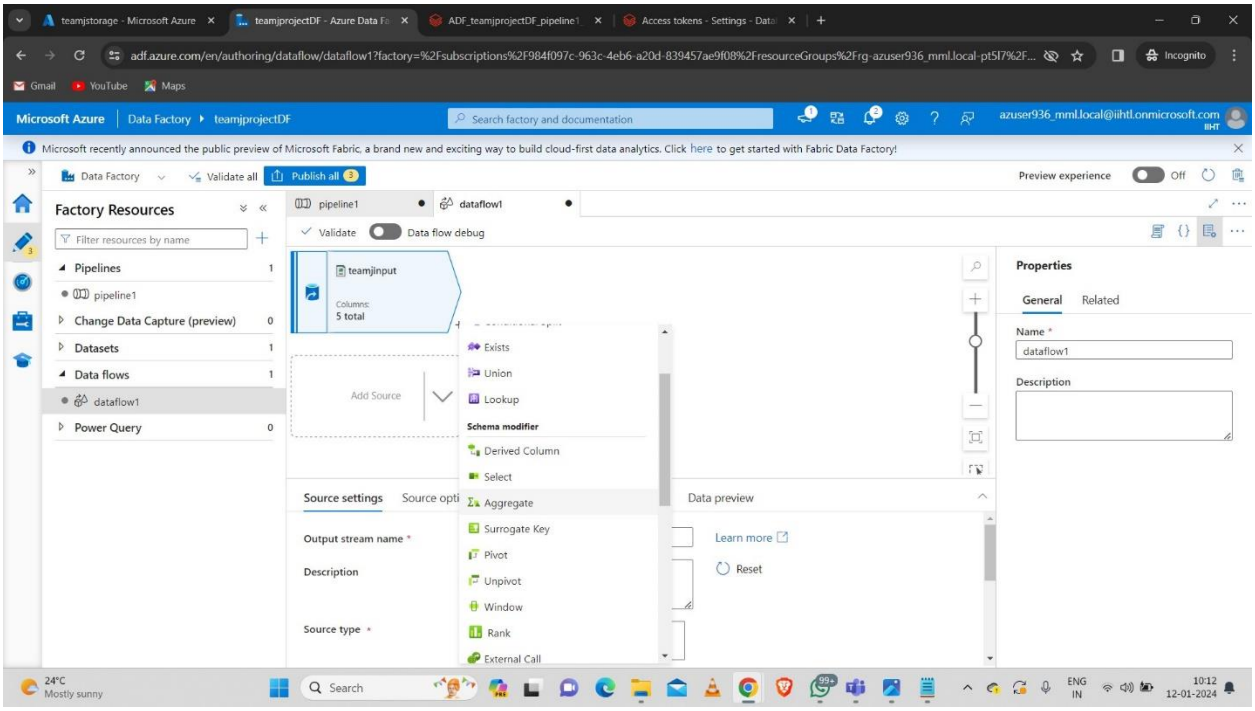
- Create the new linked service.



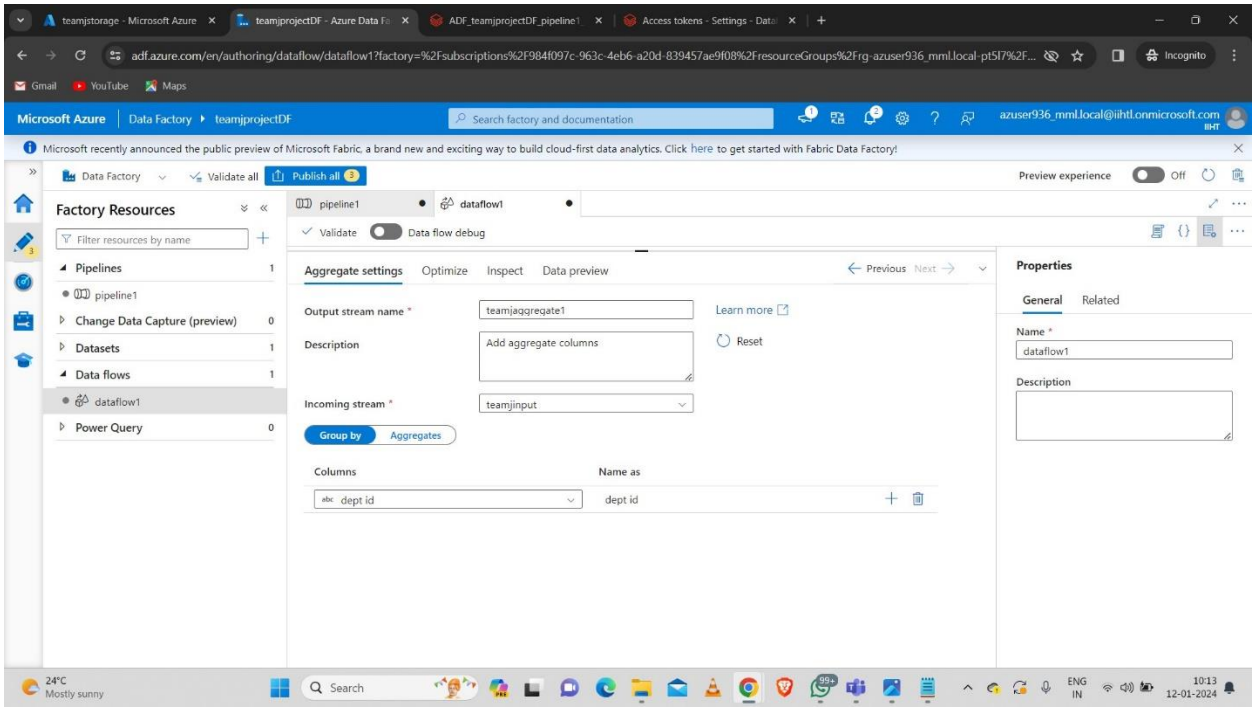
- Give the path of the source file that we uploaded in the storage account which is "project.csv".



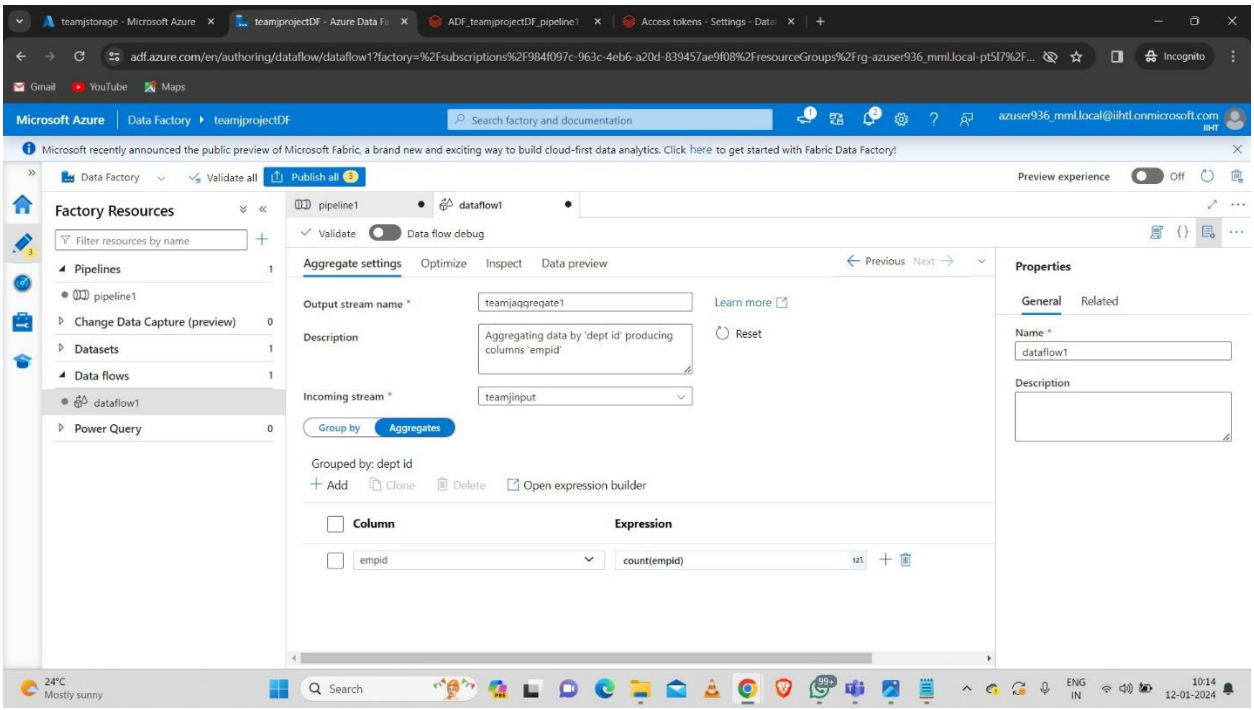
Step-3: Now add the aggregate schema modifier to the teamjinput.

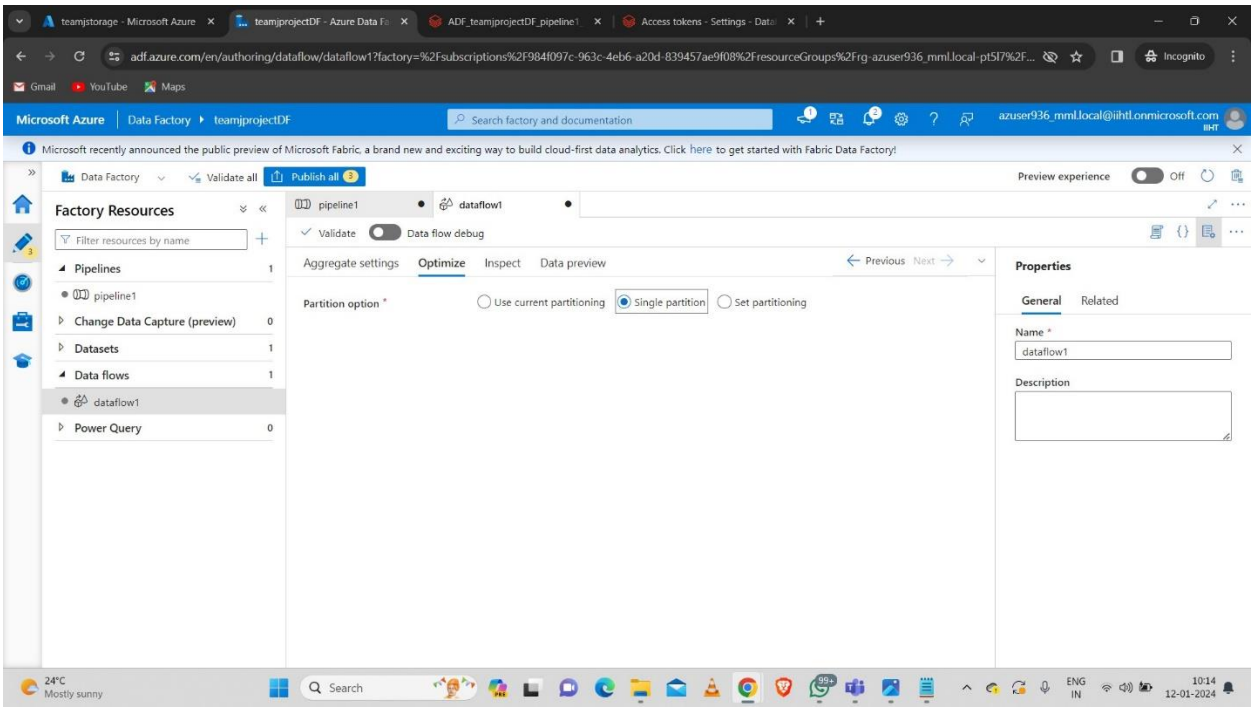


- Give the necessary data to perform the aggregation on the data.

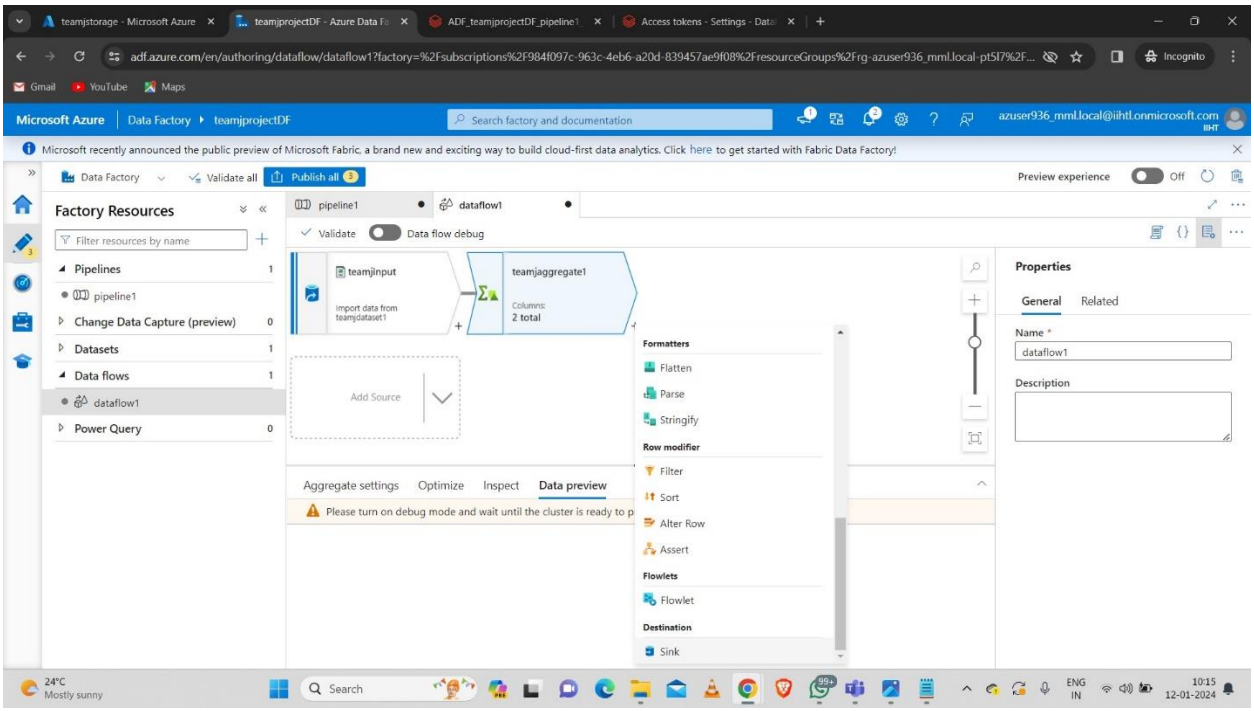


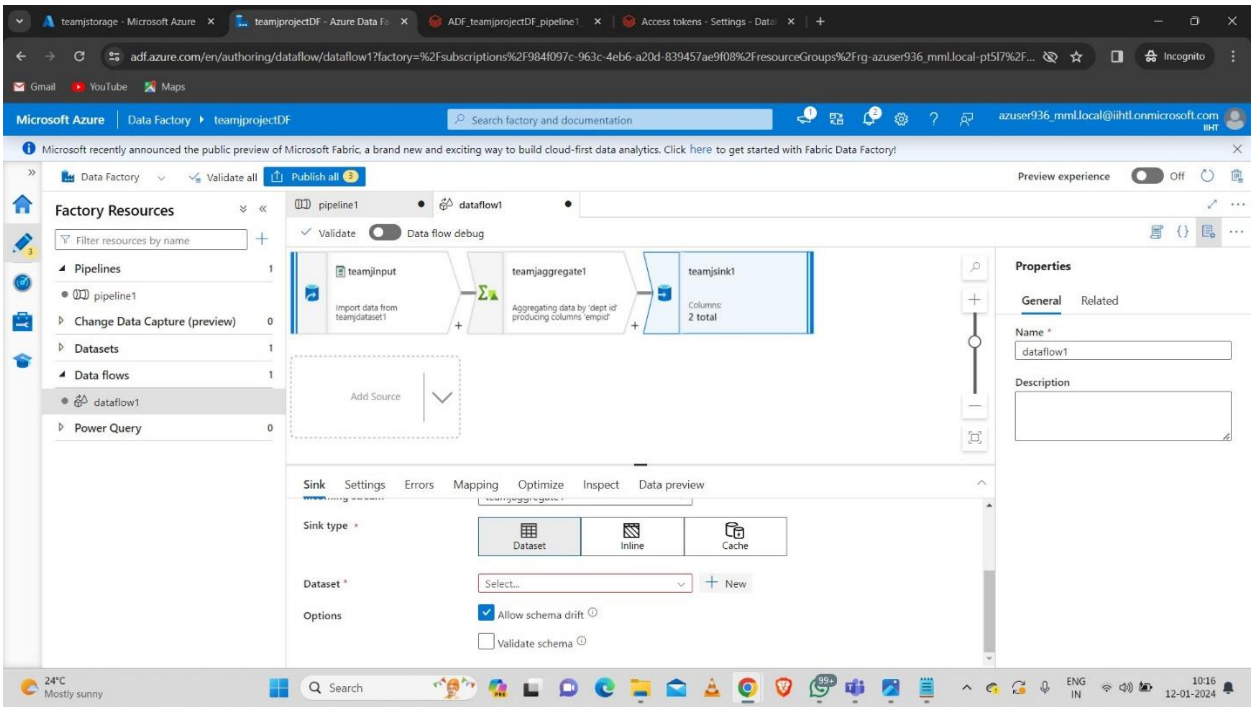
- Here we are performing the "GroupBy" operation as groupby department by aggregating as per count(empid).



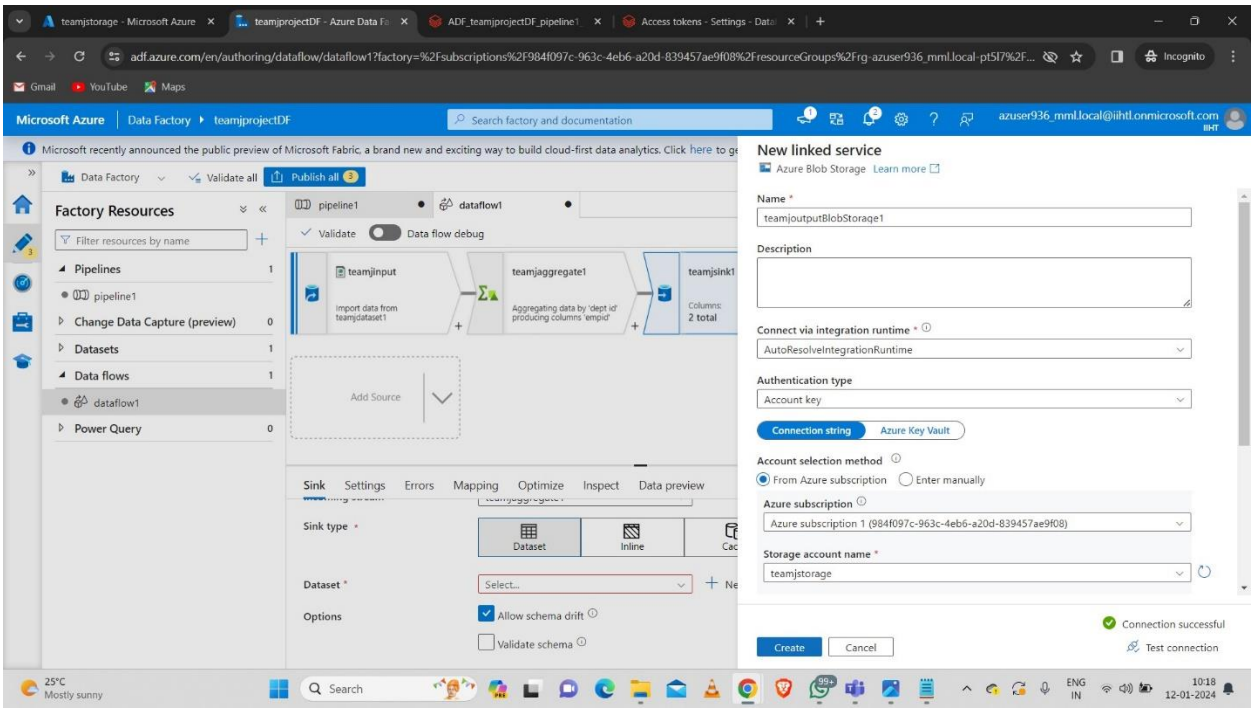


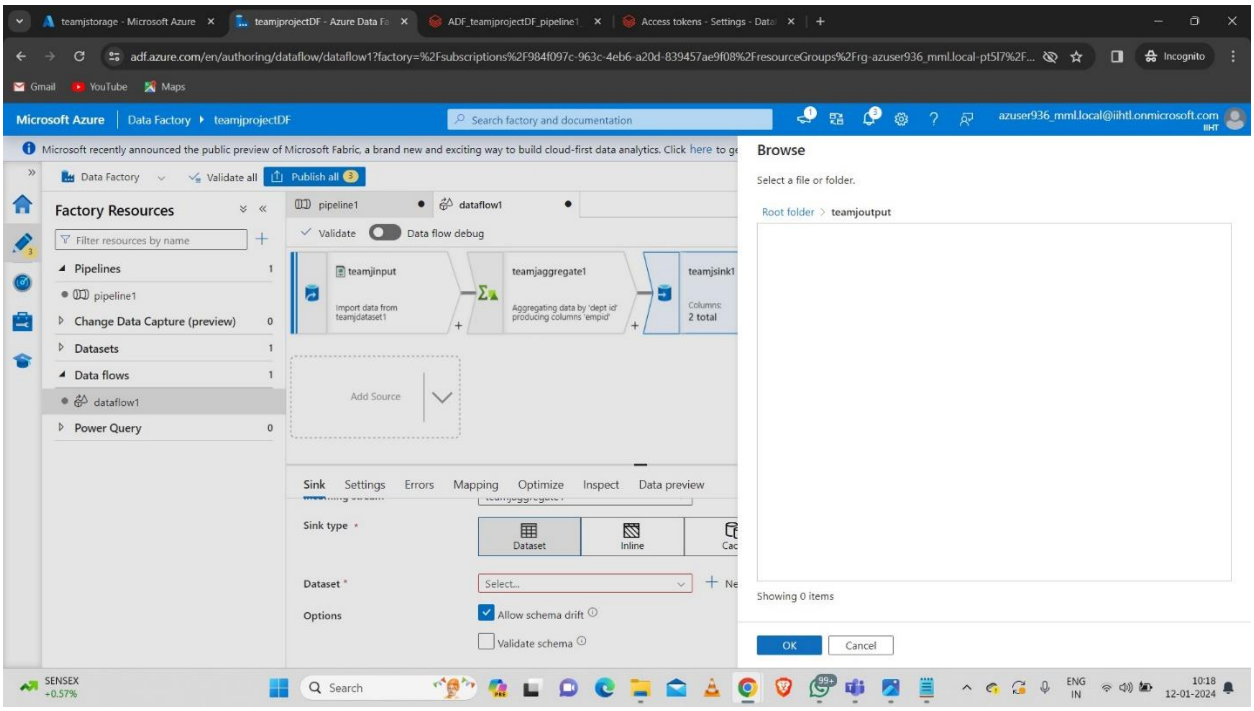
Step-4: Now we sink the aggregated data to another empty container.

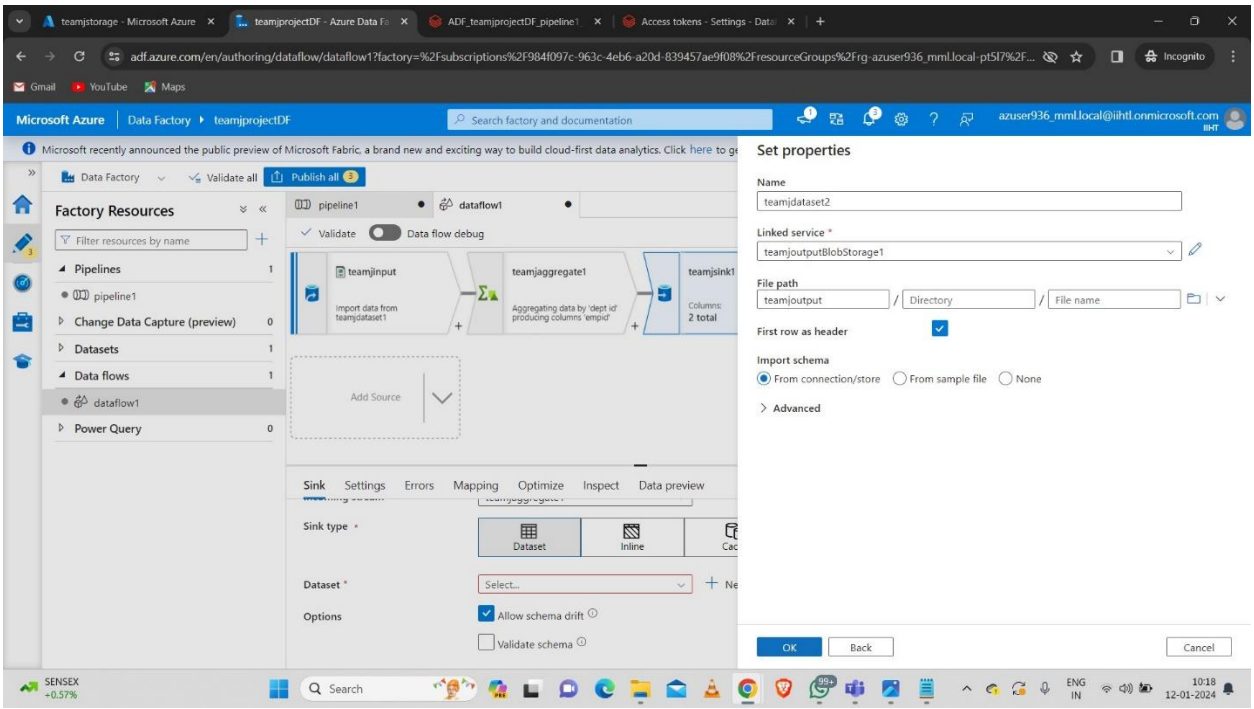




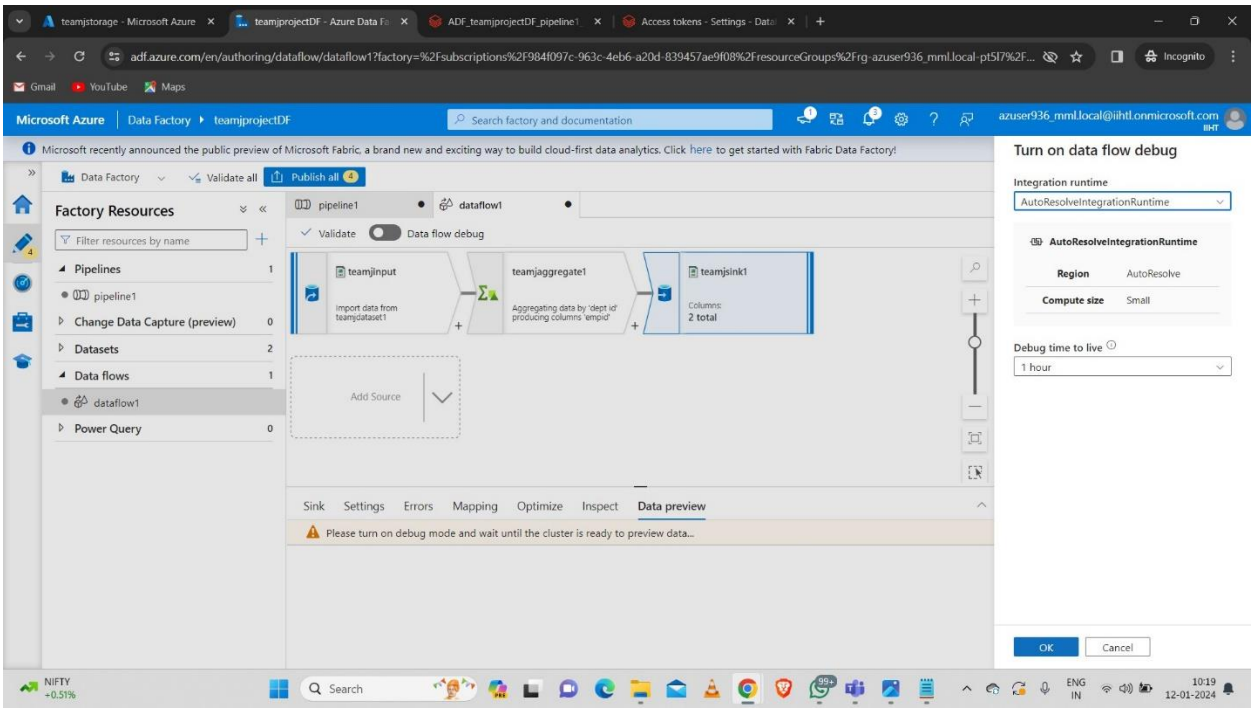
- Now create a new Dataset and link the dataset with output empty container.

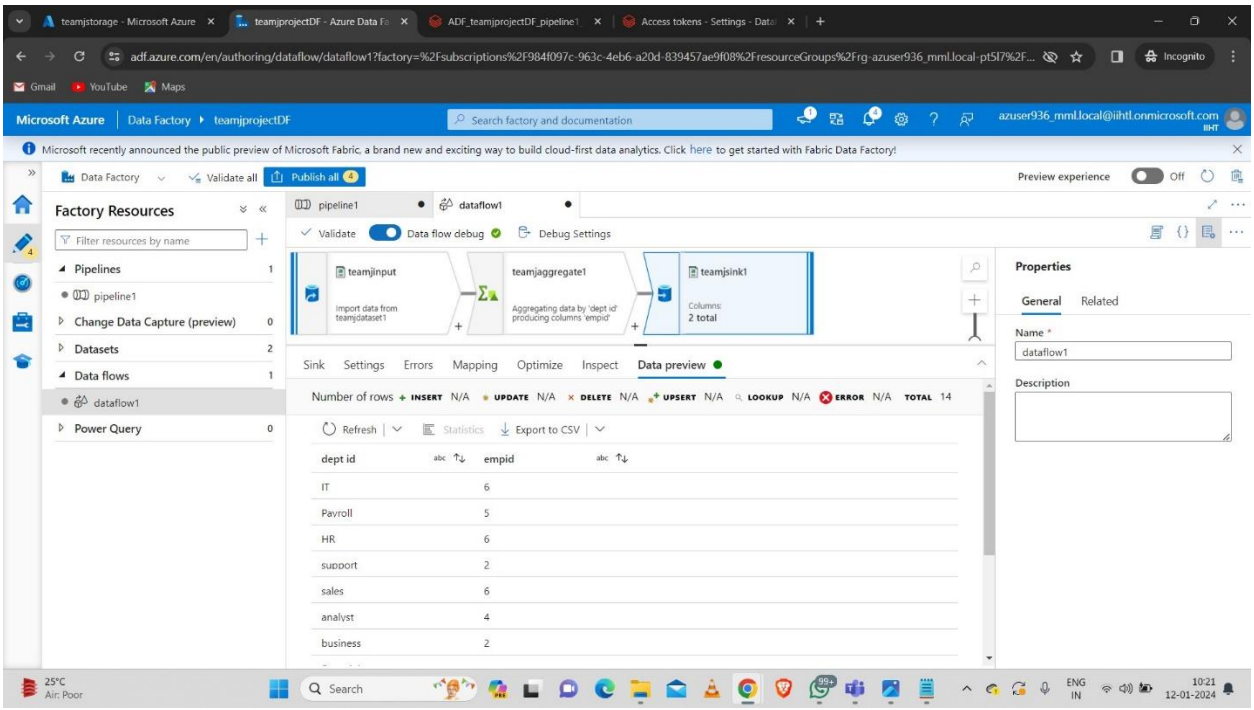



Microsoft Azure
Data Factory
teamjprojectDF
Factory Resources
Pipelines
pipeline1
Change Data Capture (preview)
Datasets
Data flows
dataflow1
Power Query
Validate all
Publish all
Validate
Data flow debug
teamjinput
teamjaggregate1
teamjsink1
Add Source
Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Sink type
Dataset
Inline
Options
Allow schema drift
Validate schema
Browse
Select a file or folder.
Root folder
teamjoutput
Showing 0 items
OK
Cancel

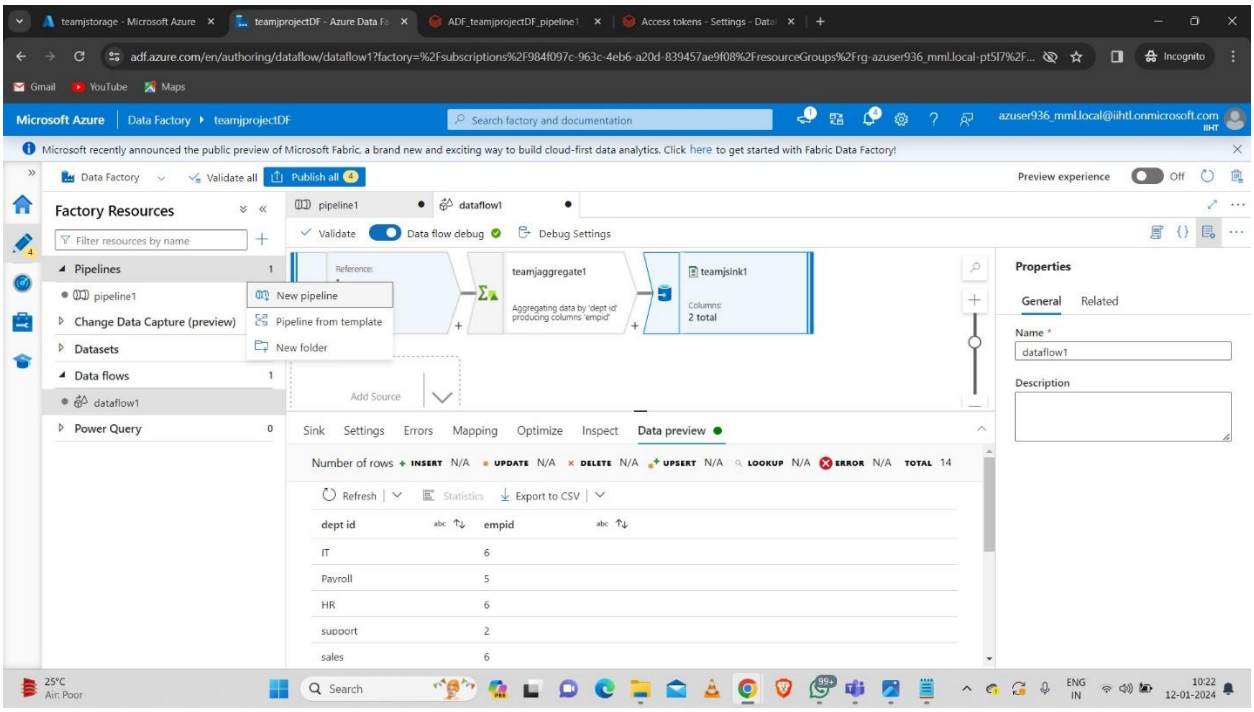

Microsoft Azure
Data Factory
teamjprojectDF
Factory Resources
Pipelines
pipeline1
Change Data Capture (preview)
Datasets
Data flows
dataflow1
Power Query
Validate all
Publish all
Validate
Data flow debug
teamjinput
teamjaggregate1
teamjsink1
Add Source
Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Sink type
Dataset
Inline
Options
Allow schema drift
Validate schema
Set properties
Name
teamjdataset2
Linked service
teamjoutputBlobStorage1
File path
teamjoutput
Directory
File name
First row as header
Import schema
From connection/store
From sample file
None
Advanced
OK
Back
Cancel

- Enable the Dataflow debug to preview the data as follows.

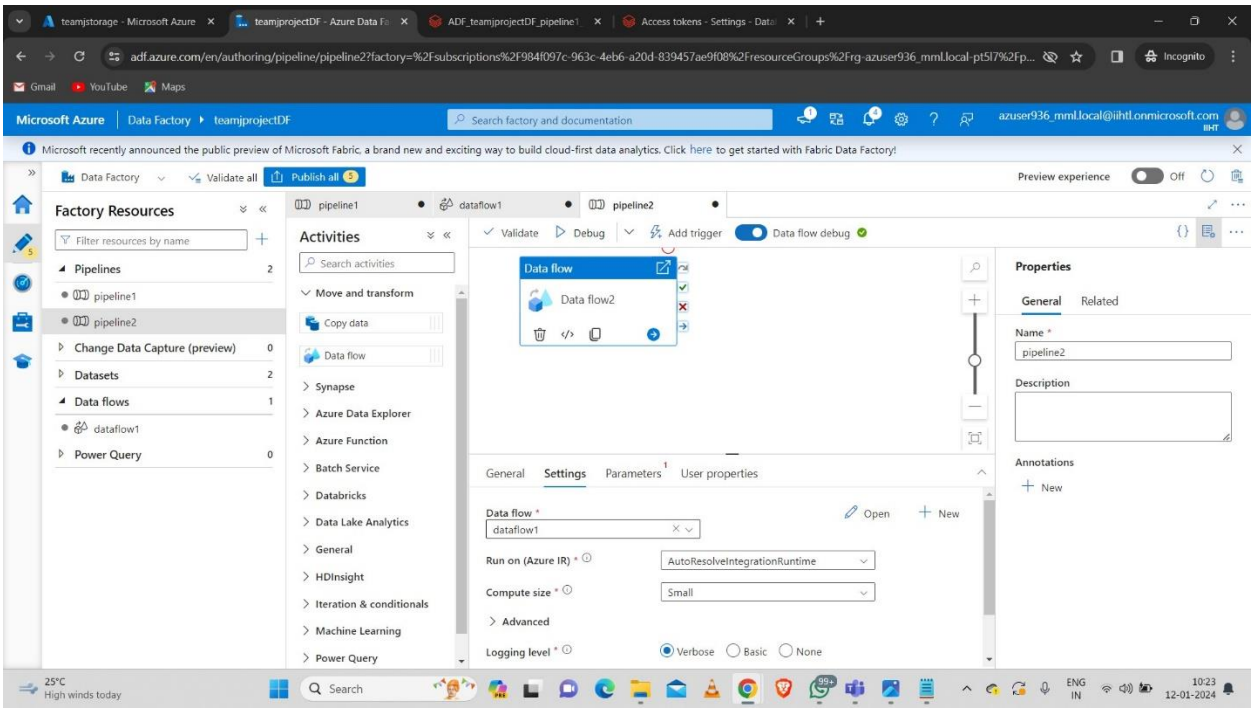




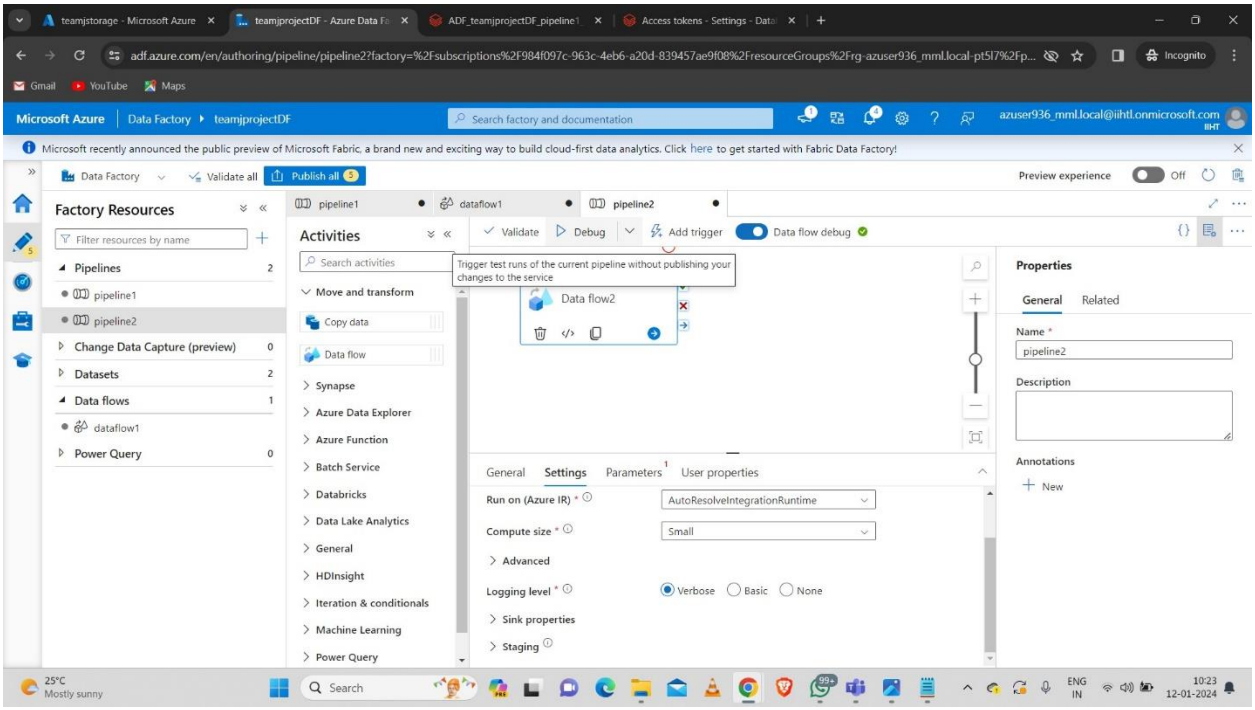
Step-5: Now create a new pipeline.



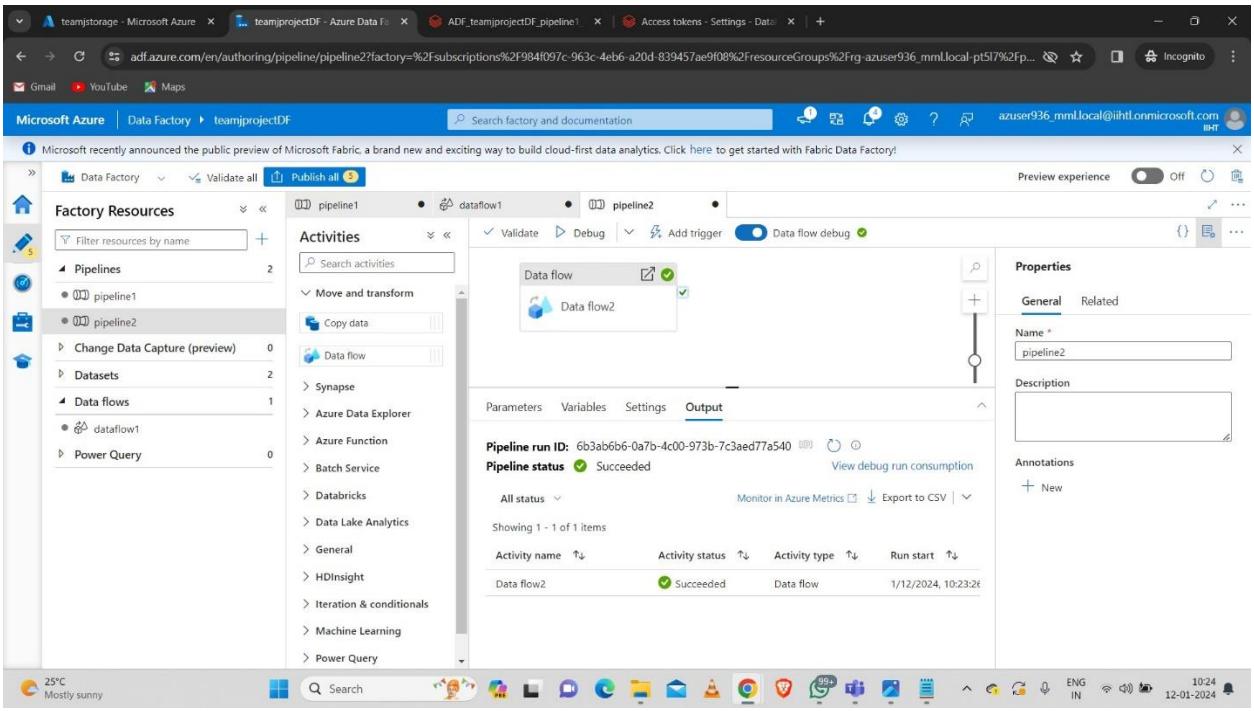
- Drag the data flow activity and give the path of data flow as “Dataflow-1” which we created before.



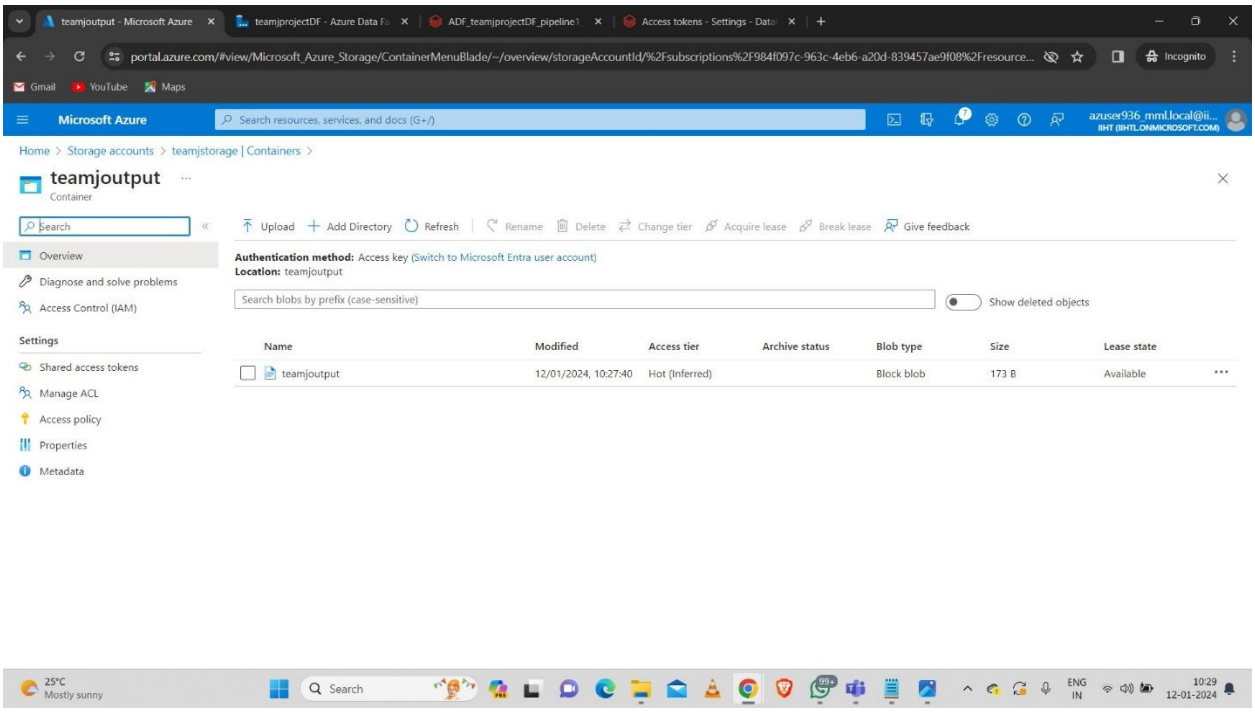
- Debug the pipeline.



- Now we can see that the debug is succeded.



Hence the aggregated file is successfully copied using Data flow activity.



Conclusion: In conclusion, implementing a daily data aggregation pipeline using Azure Data Factory and Azure Databricks offers a robust and scalable solution for handling raw data and deriving meaningful insights on a daily basis. This integration provides a seamless workflow that combines the data movement capabilities of Azure Data Factory with the powerful data processing and analytics features of Azure Databricks.