**PROJECT TITLE - HYBRID CLOUD DATA MOVEMENT**

**TABLE OF CONTENTS**

* Project Statement
* Project Overview
* Project Requirements
* Azure resources used for this project
* Architecture Diagram
* Technologies/Tools used
* Methodology
* Execution Overview
* Implementation – Tasks performed
* Practical Implementation on Azure portal
* Successful output generated
* About the project
* Conclusion

**PROJECT STATEMENT**

This project demonstrates the secure, automated migration of data from an on-premises SQL Server database to Azure cloud using Azure Data Factory, followed by scalable processing in Azure Databricks. The workflow showcases how a hybrid data engineering pipeline can bridge on-premises systems with modern cloud analytics.

**PROJECT OVERVIEW**

* Access to an on-premises SQL Server with network connectivity to the Integration Runtime host.
* Windows machine for installing the Self-hosted Integration Runtime (IR).
* Azure subscription (for Data Factory, Blob Storage, and Databricks).
* Proper SQL Server permissions for data extraction.
* Familiarity with Azure Portal, Data Factory, and Databricks.

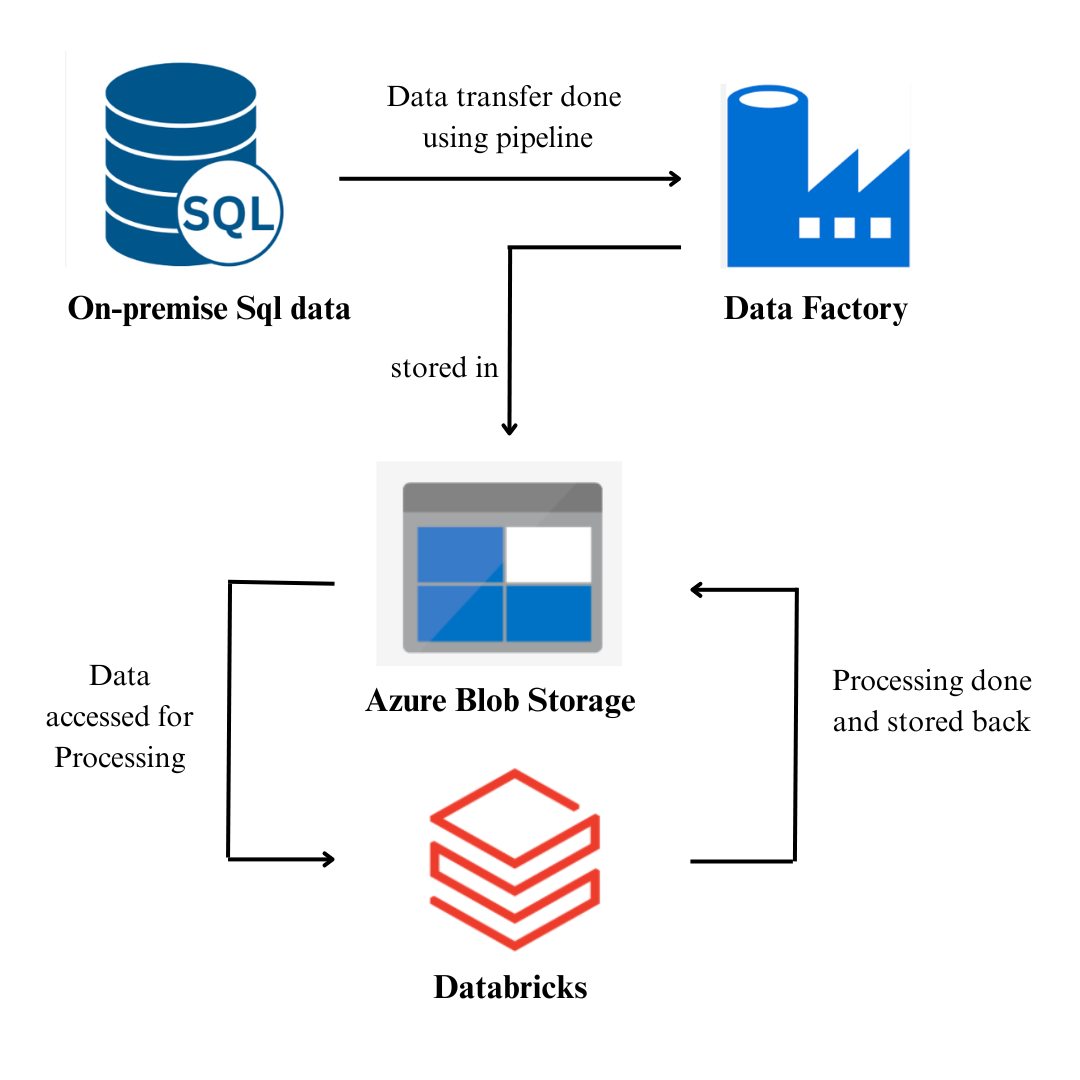
**PROJECT REQUIREMENTS**

* Access to an on-premises SQL Server (with network connectivity to IR host).
* Windows machine to install the Self-hosted Integration Runtime.
* Azure subscription for deploying Data Factory, Blob Storage, and Databricks.
* Proper permissions on SQL Server for data extraction..
* Familiarity with Azure Portal, ADF, and Databricks

**AZURE RESOURCES USED FOR THIS PROJECT**

* Azure Resource Group
* Azure Data Factory
* Self-hosted Integration Runtime
* Azure Blob Storage
* Azure Databricks (workspace & cluster)

**ARCHITECTURE DIAGRAM**

****



This architecture shows how the Self-hosted Integration Runtime enables a secure connection between Azure Data Factory and the on-premises SQL Server, with transferred data landing in Azure Blob Storage and being subsequently processed in Azure Databricks notebooks.

**TECHNOLOGIES / TOOLS USED**

* **Microsoft SQL Server (on-premises)**: Enterprise relational data source.
* **Azure Data Factory**: Orchestration and automation of pipeline.
* **Self-hosted Integration Runtime**: Bridge from on-premises to Azure Data Factory.
* **Azure Blob Storage**: Cloud data landing destination.
* **Azure Databricks**: Scalable analytics/processing (Spark, Python, SQL).
* **Windows Server/PC**: Platform for running IR.

**METHODOLOGY**

**On-premises Preparation:**

* Data organized in relational tables in the SQL Server database.

Integration Runtime Setup:

* Self-hosted IR installed on a Windows machine that can reach SQL Server.

**ADF Pipeline Configuration:**

* Data Factory linked to on-premises SQL Server via Self-hosted IR.
* Data Factory linked to Azure Blob Storage.
* Copy Data pipeline moves data from SQL Server to Blob Storage.

**Cloud Staging:**

* Data deposited in Azure Blob as CSV or Parquet.

**Databricks Data Processing:**

* Databricks reads blob data, processes/cleans/analyzes as needed with Spark.

**Result Storage:**

* Processed data saved back to Blob Storage or other analytics destinations.

**EXECUTION OVERVIEW**

* Trigger ADF pipeline to extract data from SQL Server (full table/query).
* Data staged in Azure Blob.
* Databricks cluster started, blob storage accessed, files loaded into Spark.
* Data is processed and, if needed, pushed to downstream storage/services.
* Compute resources terminated after processing.

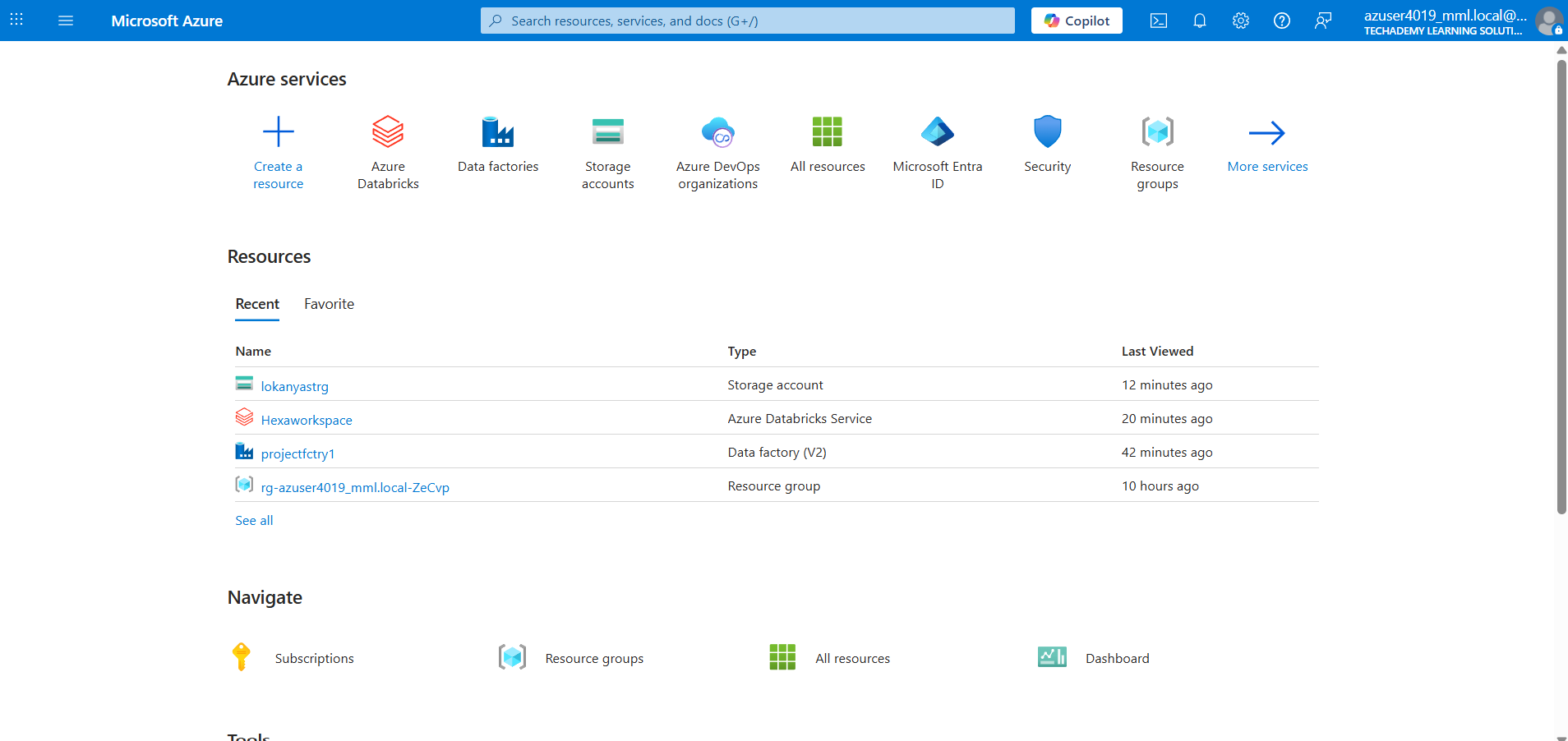
**IMPLEMENTATION - TASKS PERFORMED**

* Created source tables in SQL Server.
* Installed and registered Self-hosted Integration Runtime.
* Configured Data Factory linked services to on-premises SQL Server and Blob Storage.
* Built and published Copy Data pipeline.
* Validated file arrival in Blob.
* Set up Databricks workspace/cluster and processed data in notebooks.
* Wrote final output back to Blob Storage and shut down resources.

**PRACTICAL IMPLEMENTATION ON AZURE PORTAL**

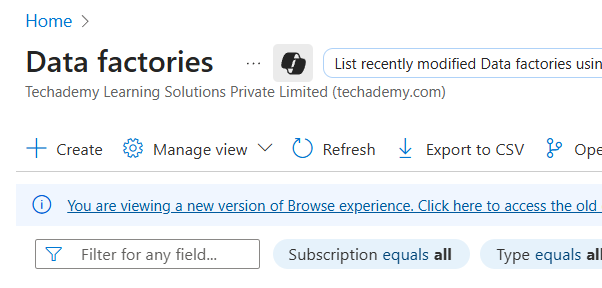
**1. Log into your Azure Account**

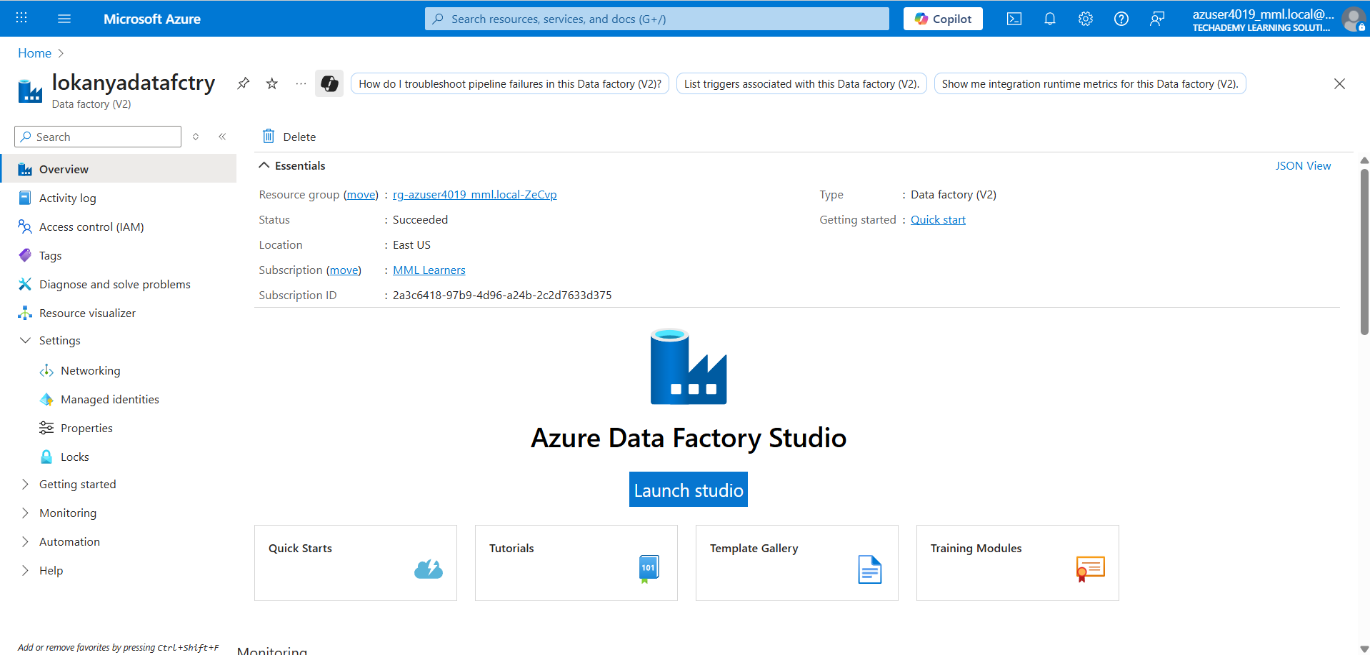
Start by signing in to your Azure account using your credentials to access the required services.

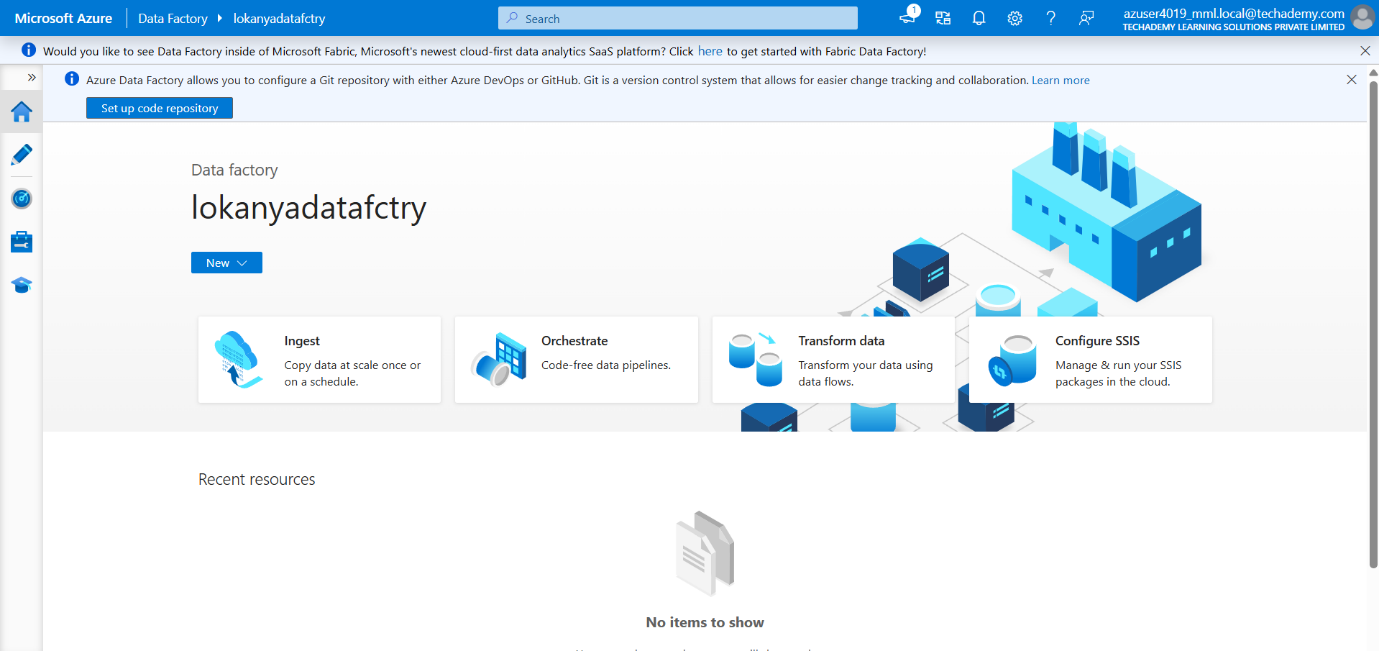


**2. Create an Azure Data Factory Instance**

Set up an Azure Data Factory instance. This will be used for building and managing pipelines to orchestrate data movement.

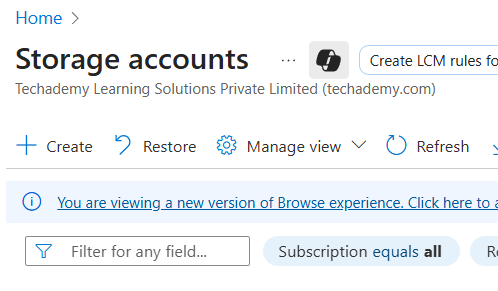


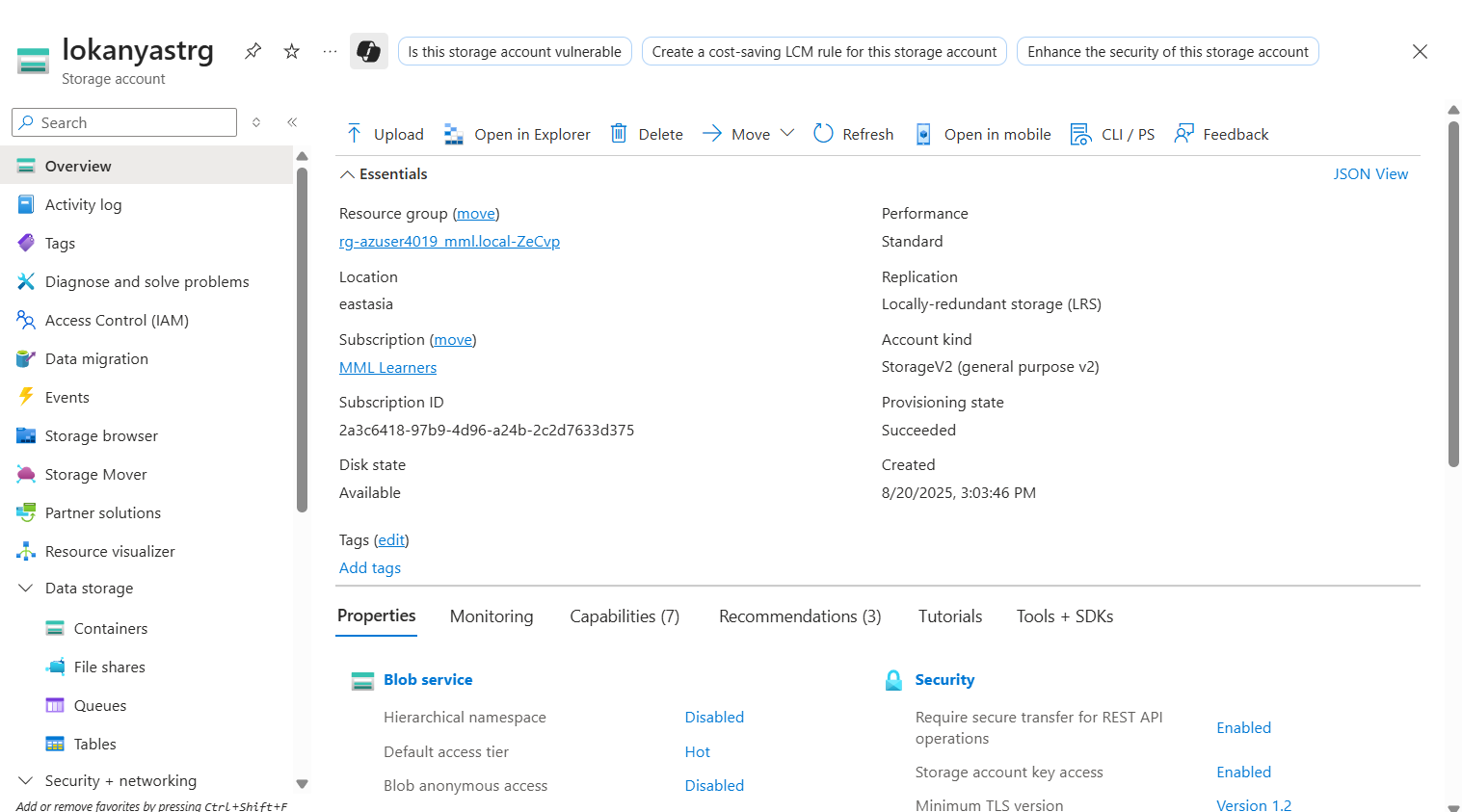




**3. Set Up an Azure Storage Account**

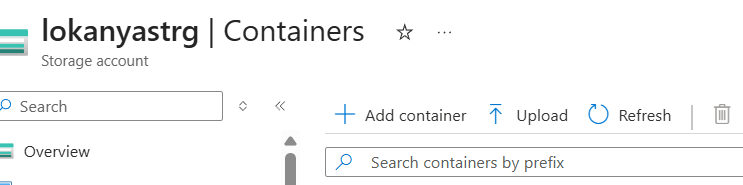
Establish a new Azure Storage account, which will be used to store your transferred data.

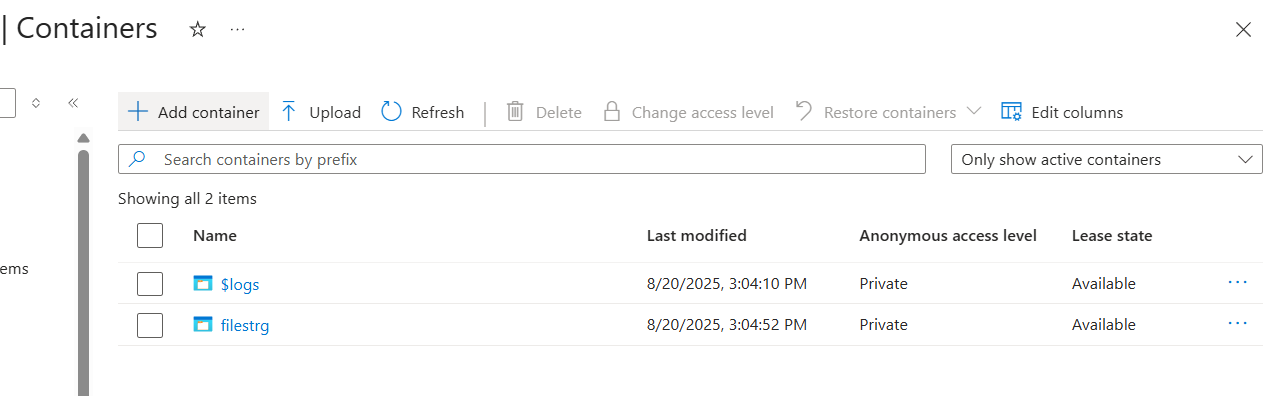




**4. Create a Blob Storage Container**

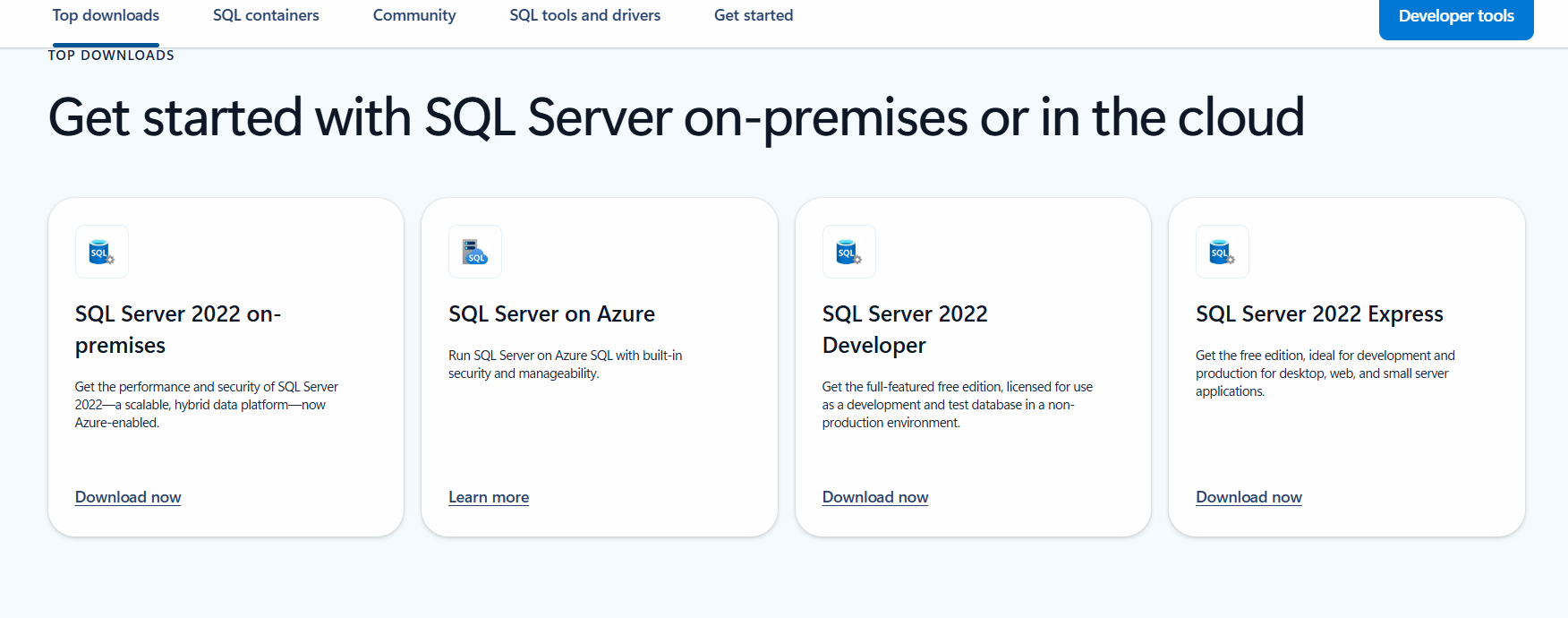
Within your storage account, create a Blob Storage container. This is where you will transfer your data from your on-premises SQL Server Management Studio (SSMS) to the cloud.





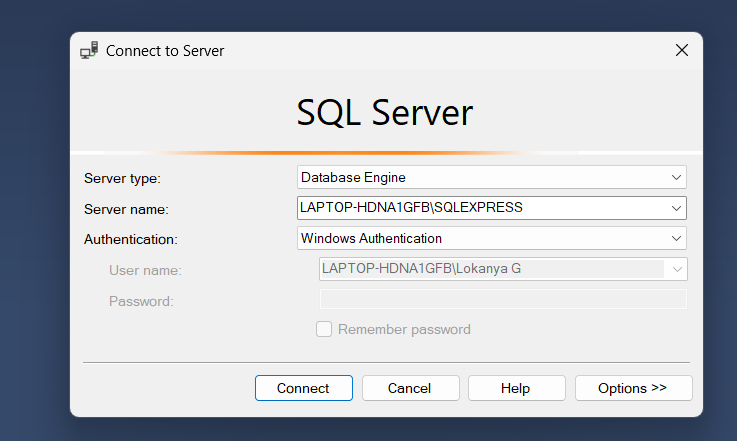
**5. Install Microsoft SQL Server Management Studio (Express Edition)**

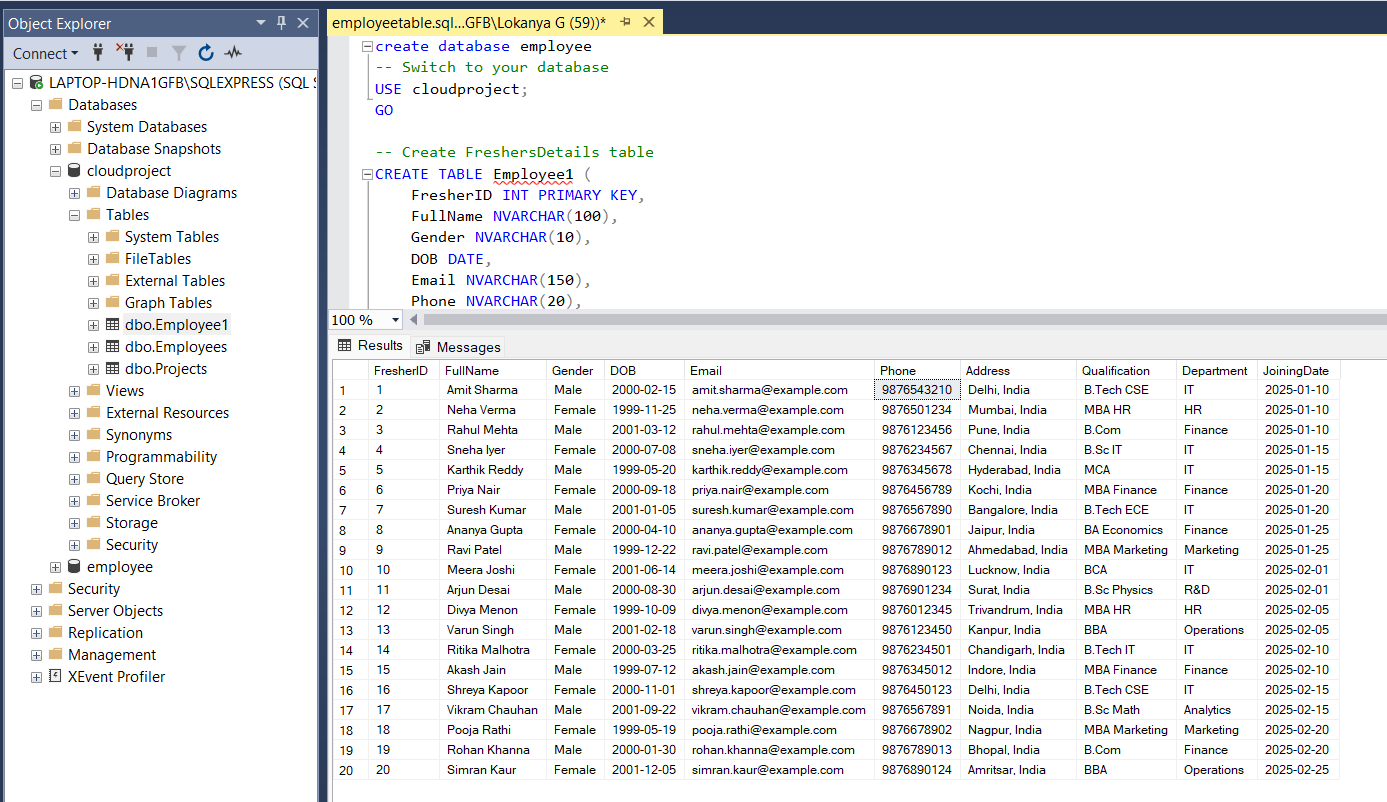
Download and install SSMS Express Edition; this tool is used to create your on-premises database and tables for the data transfer.



**6. Create a Database and Table in SSMS**

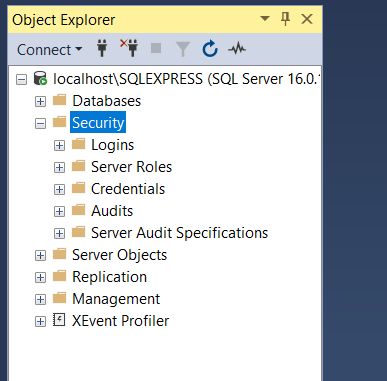
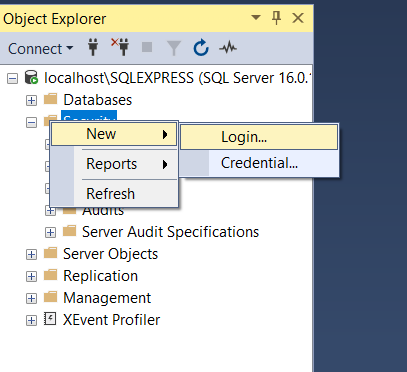
In SSMS, use the “New Query” option to create a database and the required tables.

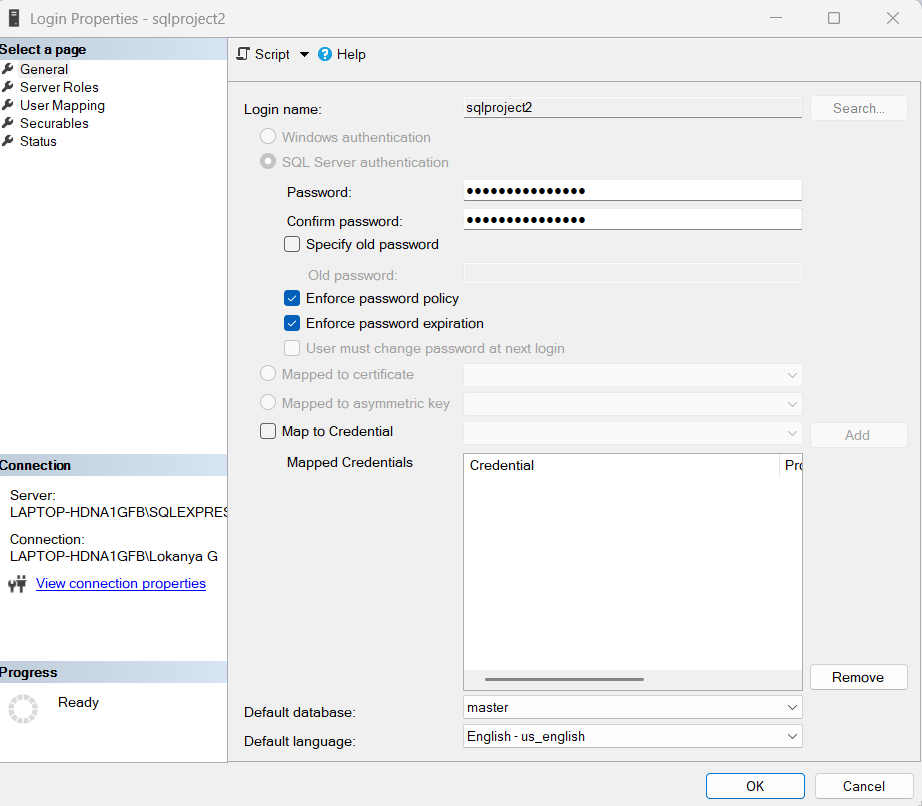




**7. Set Up SQL Server Authentication**

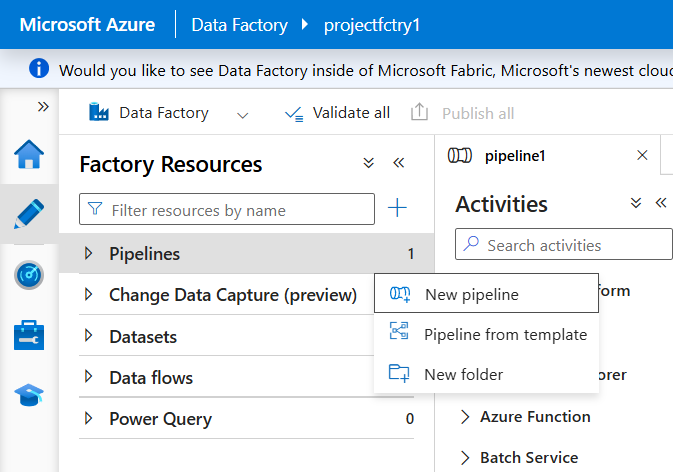
Create a new SQL Server Authentication login with a username and password, and configure the connection properties for use in the data transfer process.



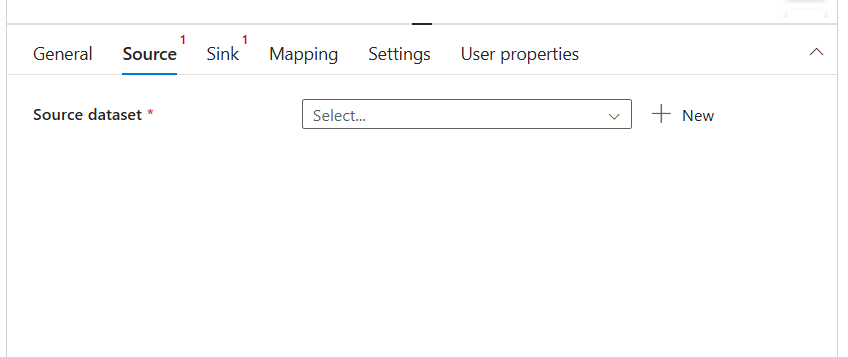
**8. Design a Pipeline in Azure Data Factory**

Navigate to the Author tab in Data Factory and create a new pipeline.



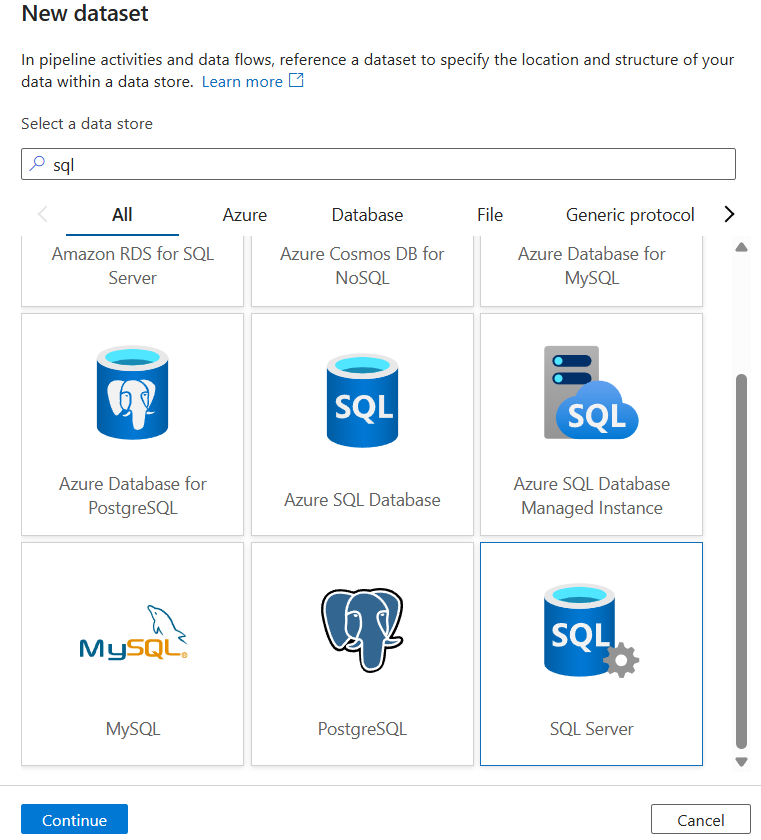
**9. Add a Copy Data Activity**

Within your pipeline, select the Copy Data activity, and then configure a new source dataset.



**10. Define the Source as SQL Server**

In the new dataset dialog, choose SQL Server as your source.



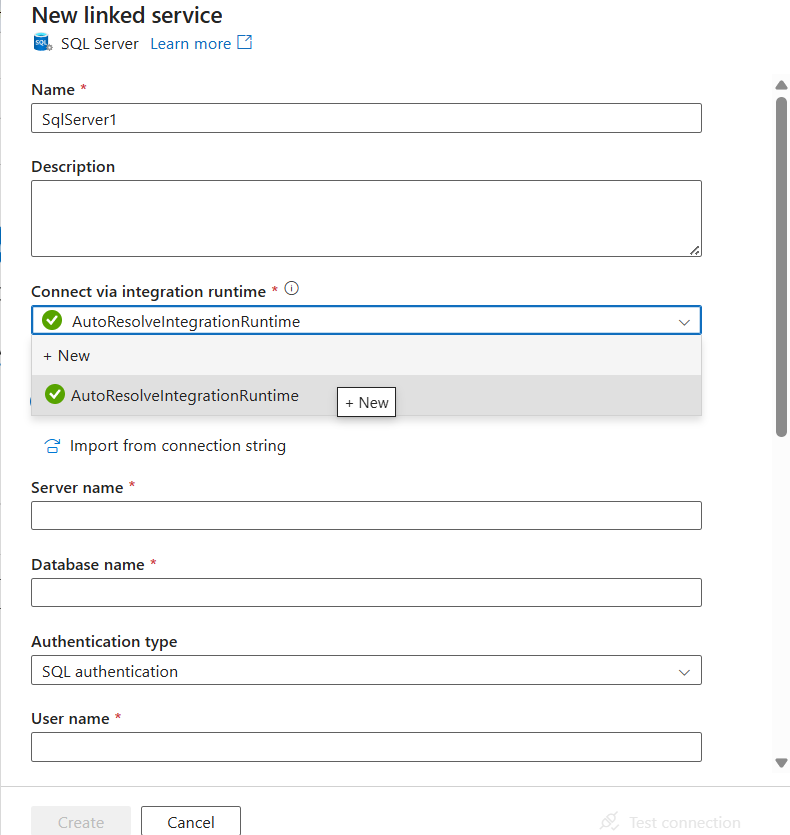
**11. Create a New Linked Service**

For the SQL Server dataset, set up a new linked service to establish a connection between ADF and your on-premises SQL Server.



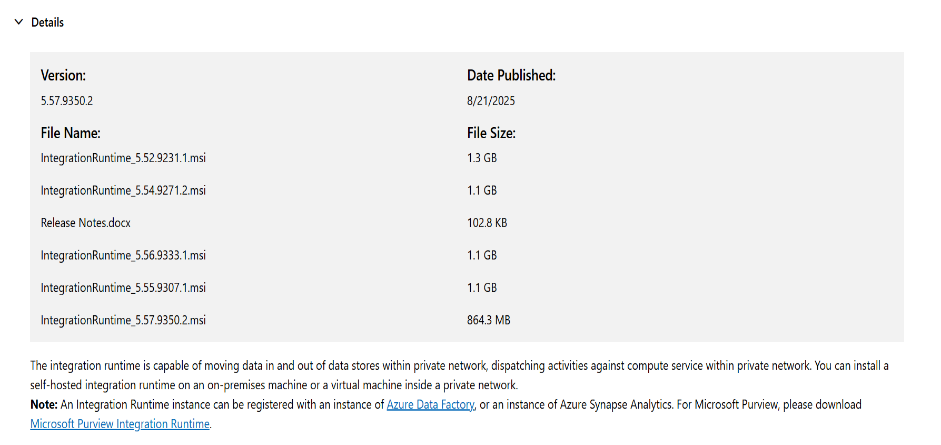
**12. Configure the Integration Runtime**

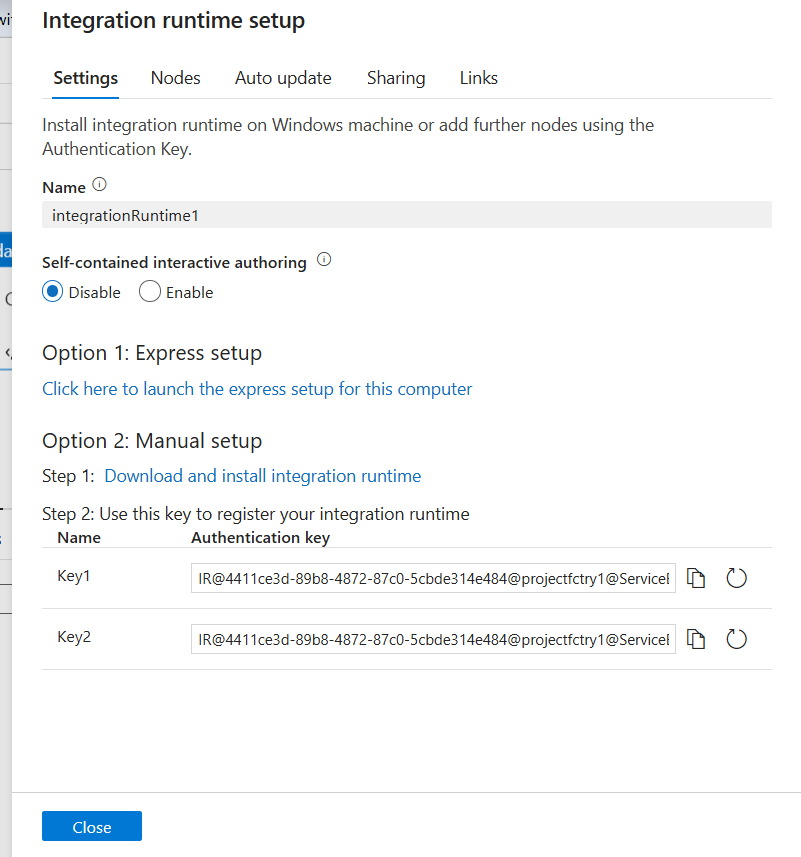
Choose to set up a self-hosted integration runtime, required for connecting ADF to your on-premises data. Do not select AutoResolveIntegrationRuntime.



**13. Install Integration Runtime Software**

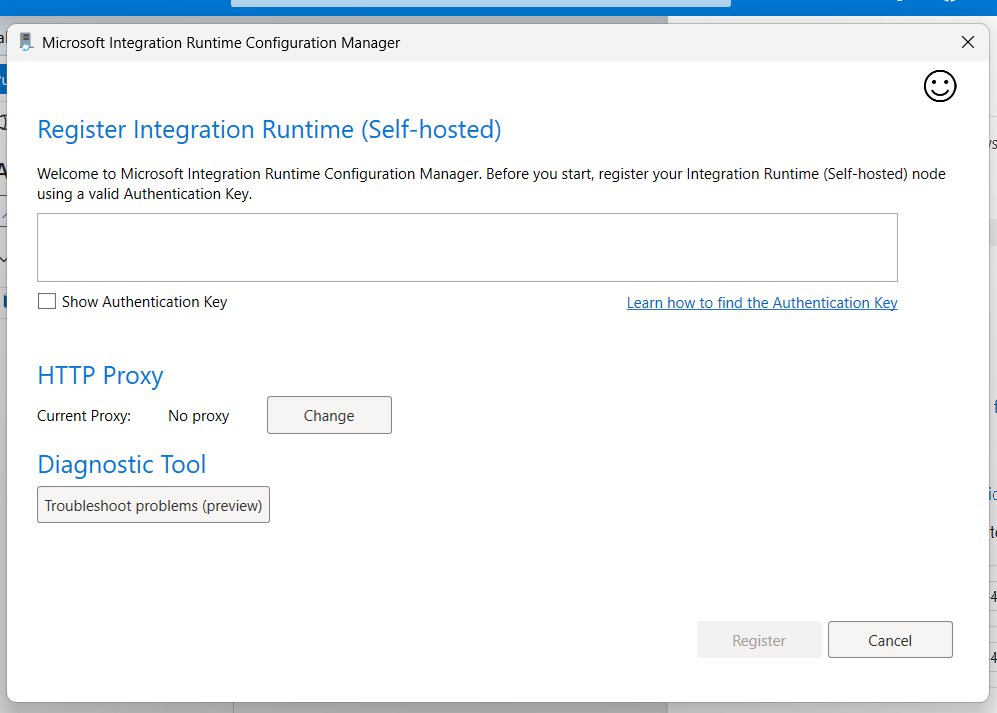
Download and install the self-hosted integration runtime application. Initiate registration using the runtime key generated in the ADF portal.

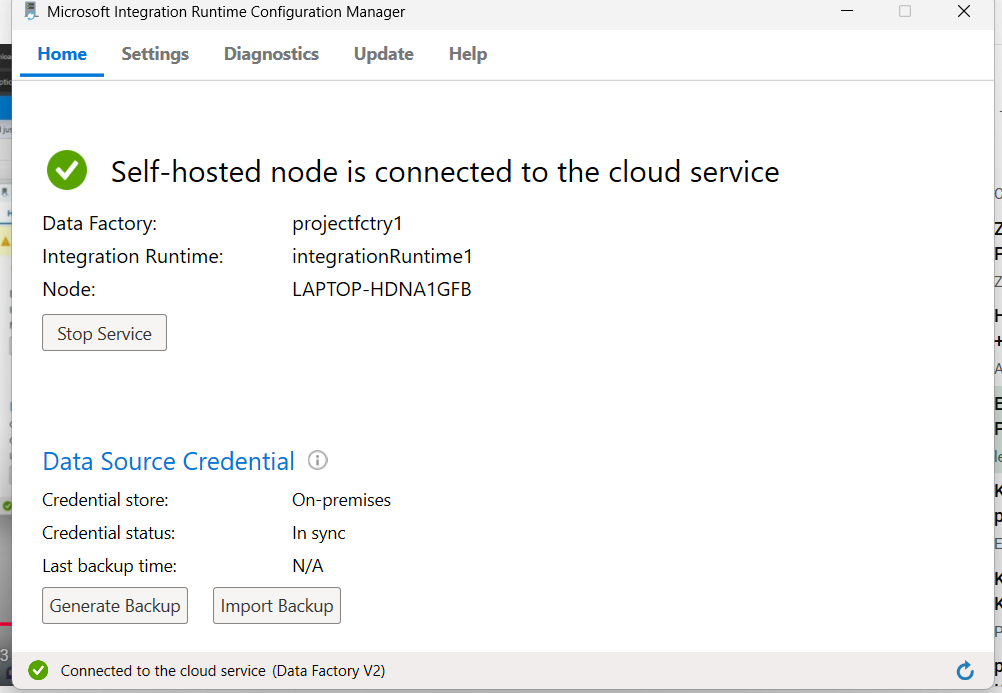




**14. Register Integration Runtime**

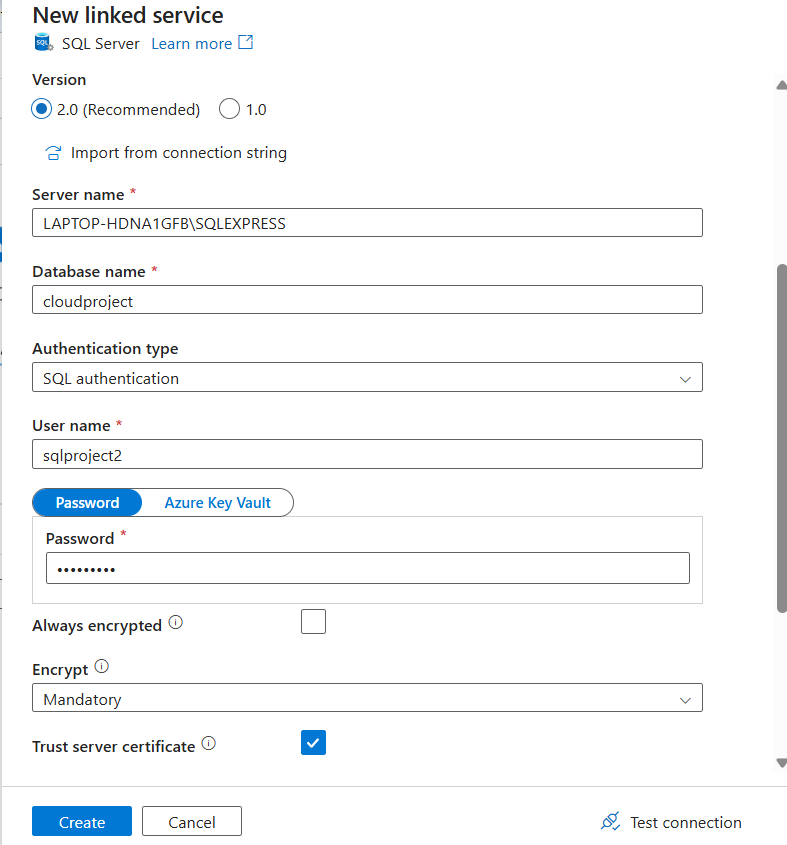
Paste the registration key (copied from ADF) into the application to establish the link.





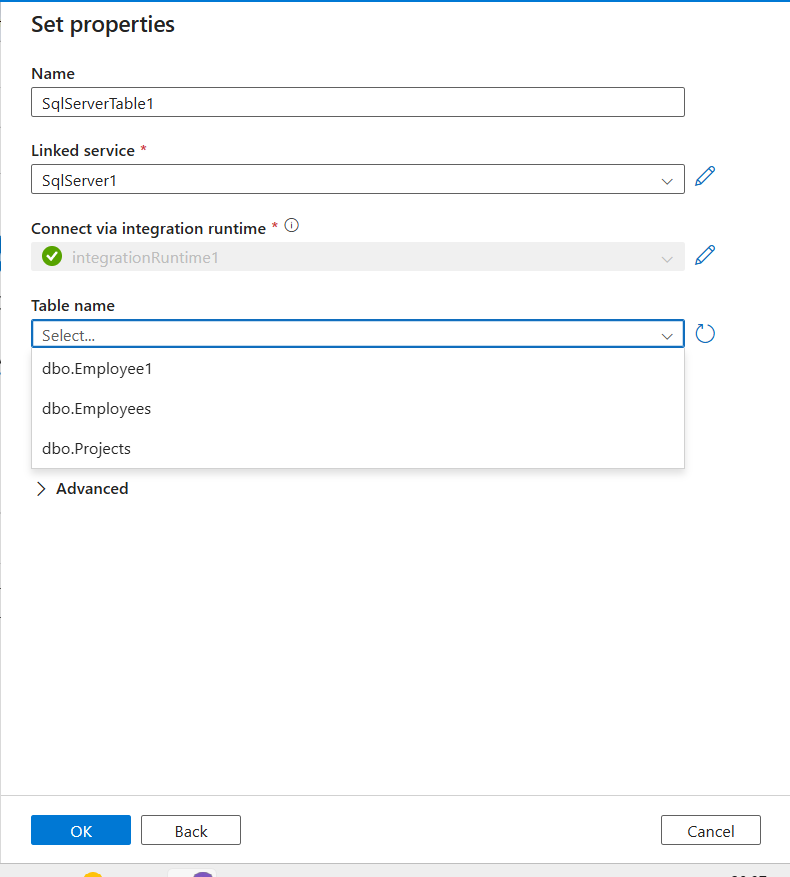
**15. Provide SQL Server Credentials**

Enter your server name, database name, authentication type, username, and password corresponding to your previously created SQL Server login. Enable "Trust Server Certificate" if required.



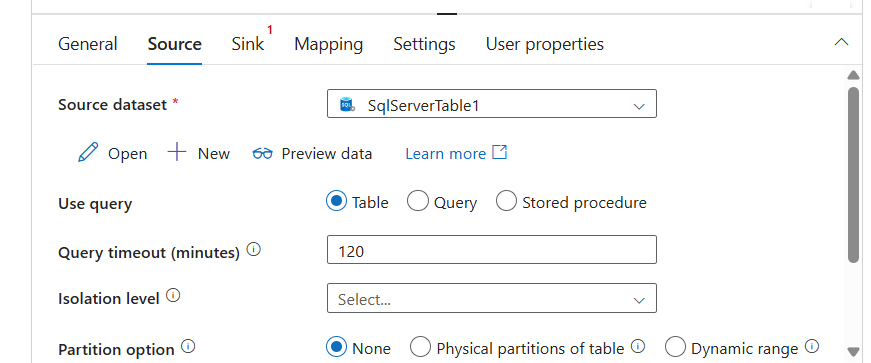
**16. Select the Source Table**

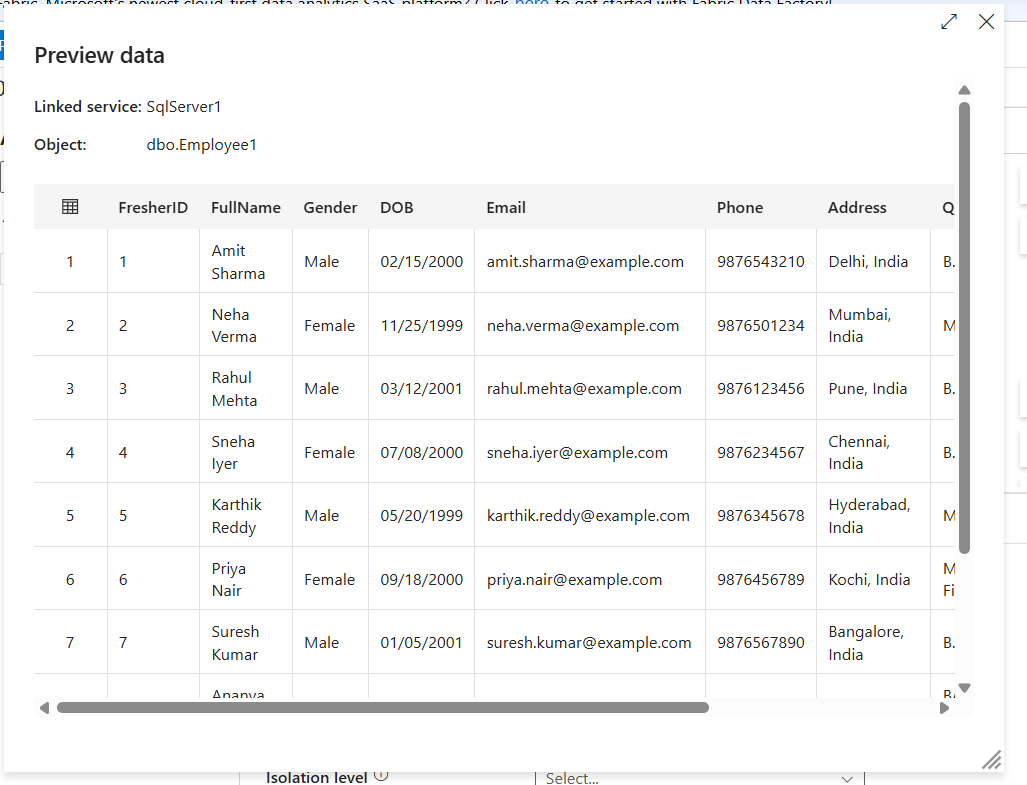
After successful connection, choose the required table from your database as the source for the data transfer.



**17. Preview Source Data**

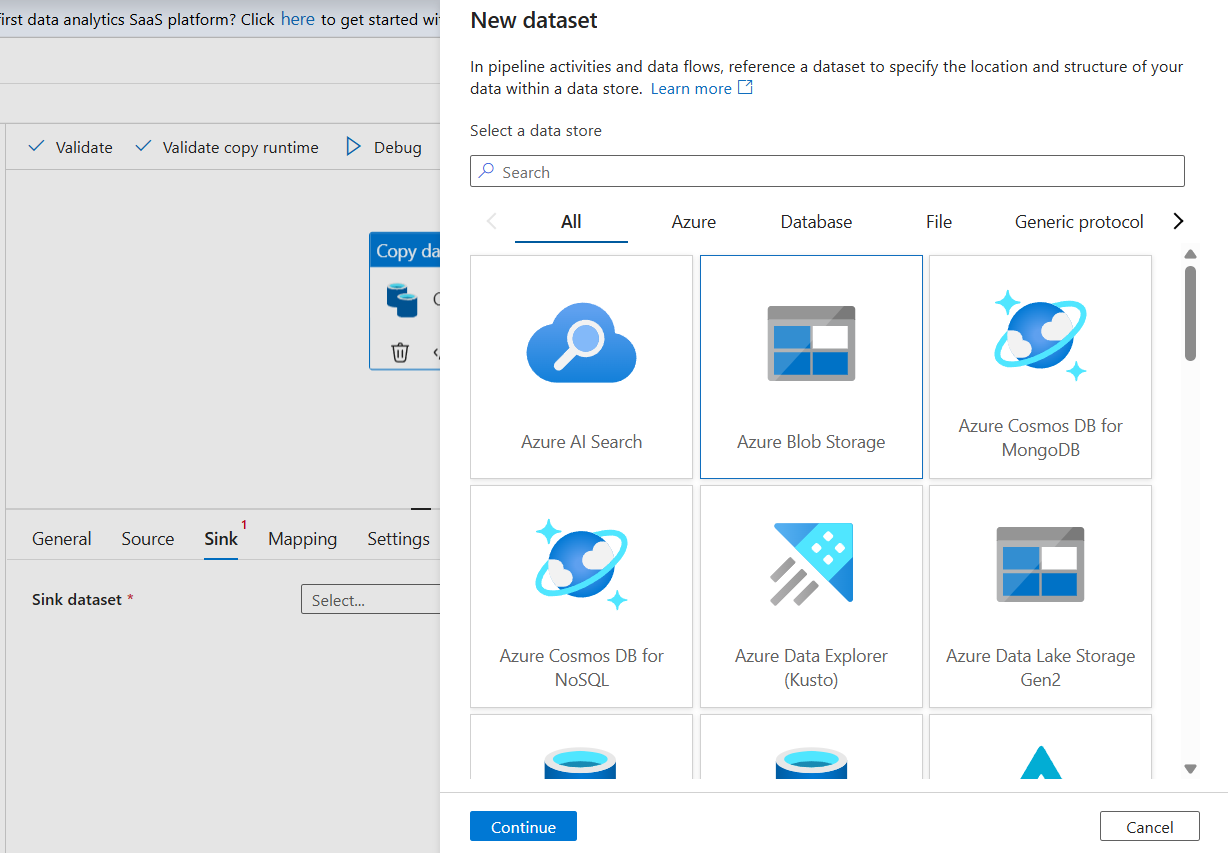
Utilize the preview table option to view and confirm the contents of your selected table.





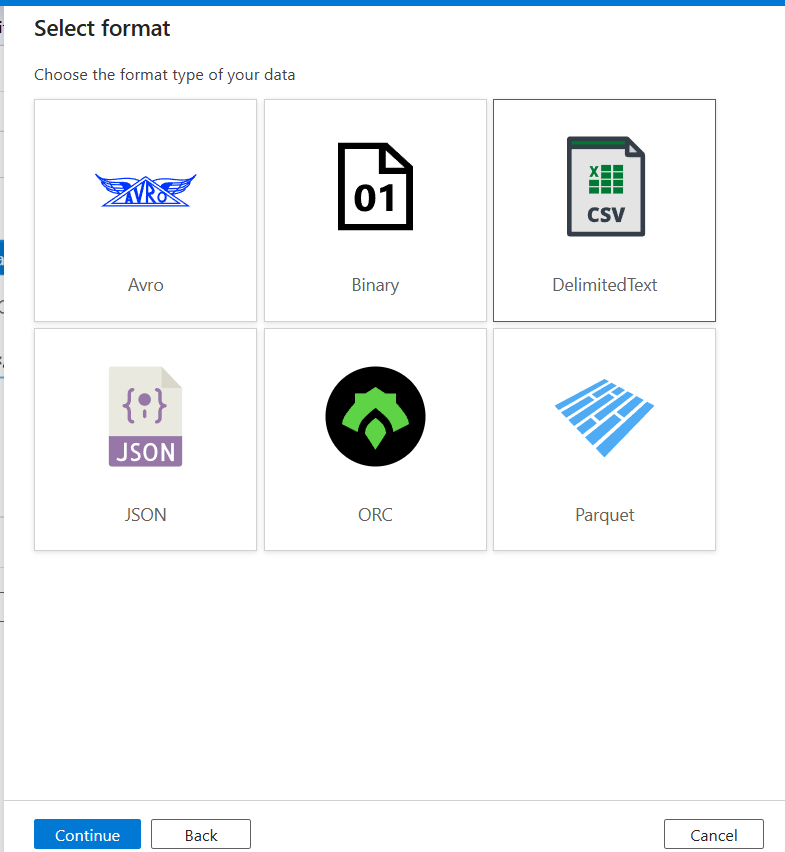
**18. Configure the Sink Dataset**

In the pipeline’s Sink section, create a new dataset for Azure Blob Storage as your data destination.



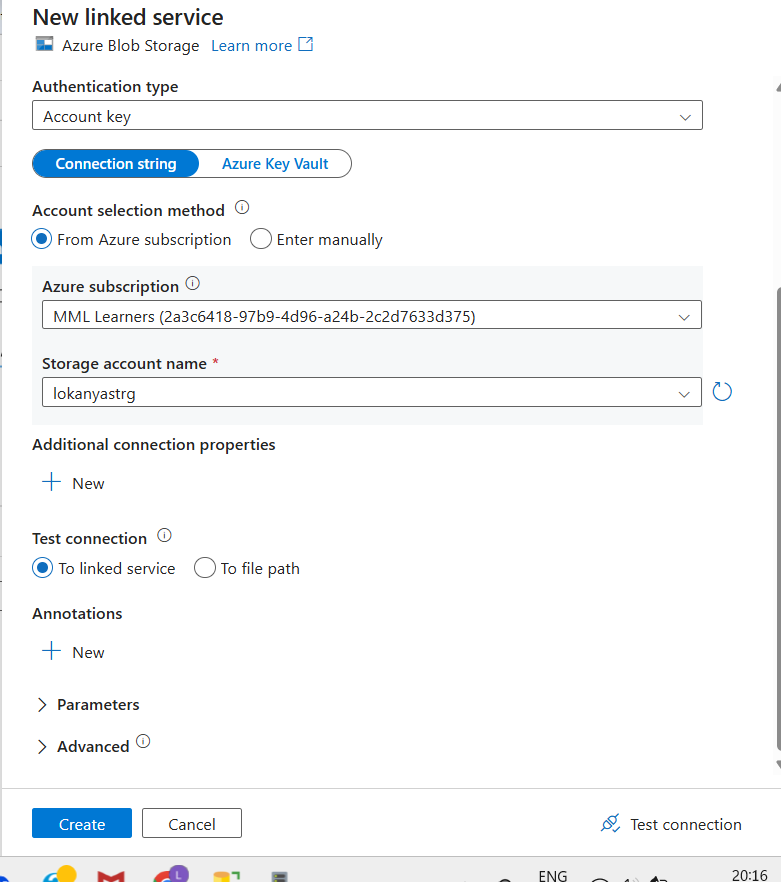
**19. Specify the File Format**

Choose “Delimited Text” as the format for your output file.



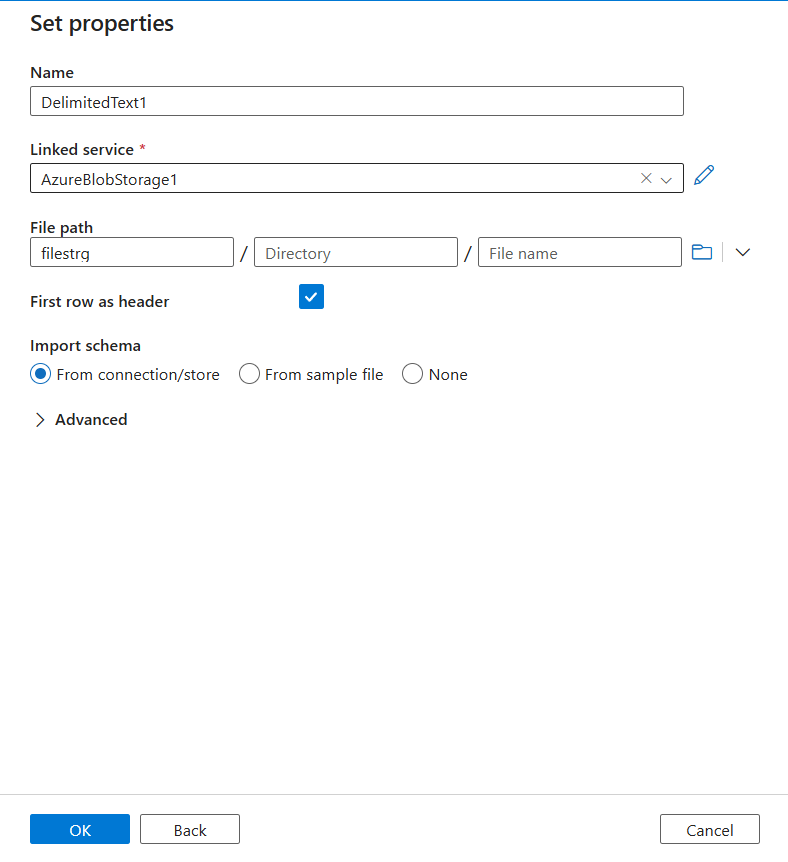
**20. Set Up Azure Blob Storage Linked Service**

Provide the necessary Azure Blob Storage credentials (storage account name, subscription) to configure the linked service.



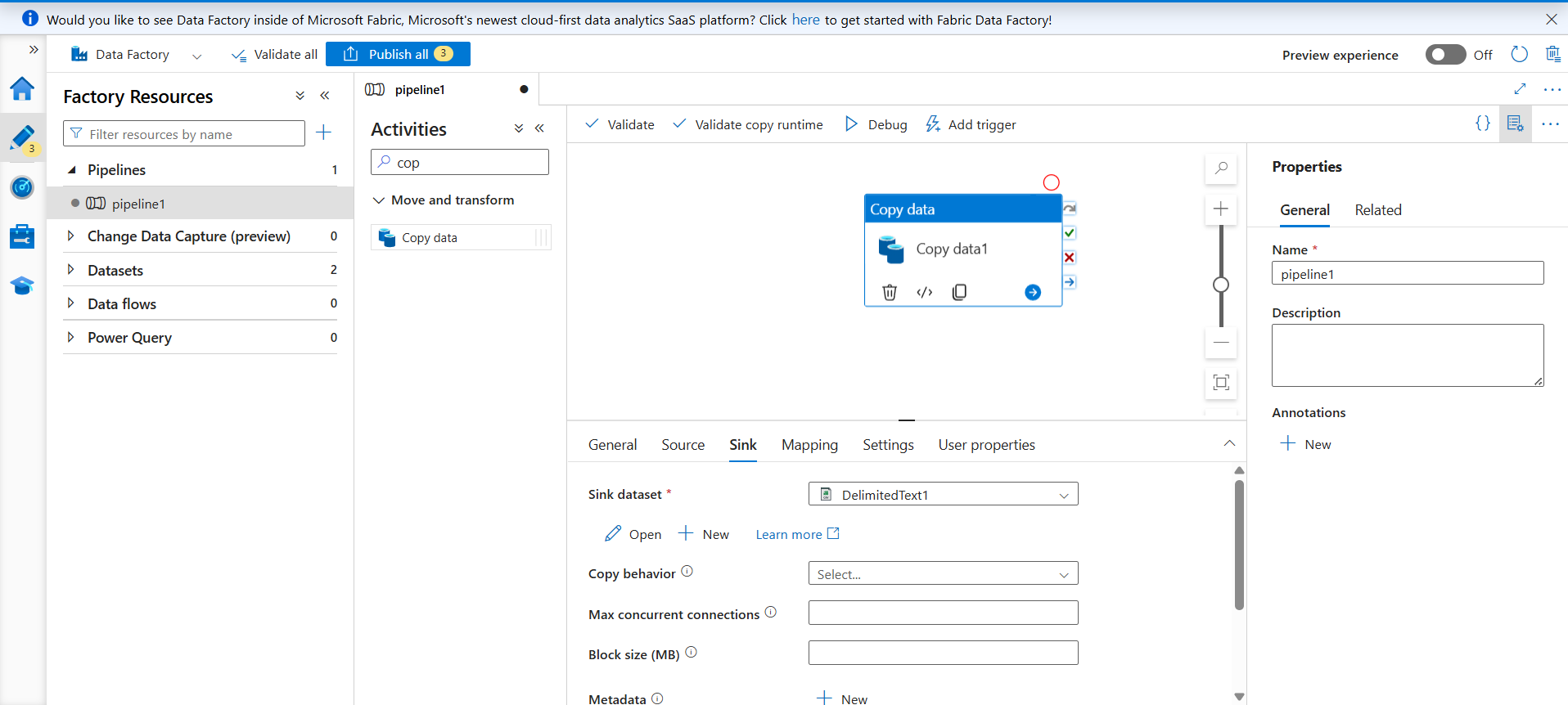
**21. Select the Blob Container**

Browse and select the Blob Storage container created earlier as the target for your data.



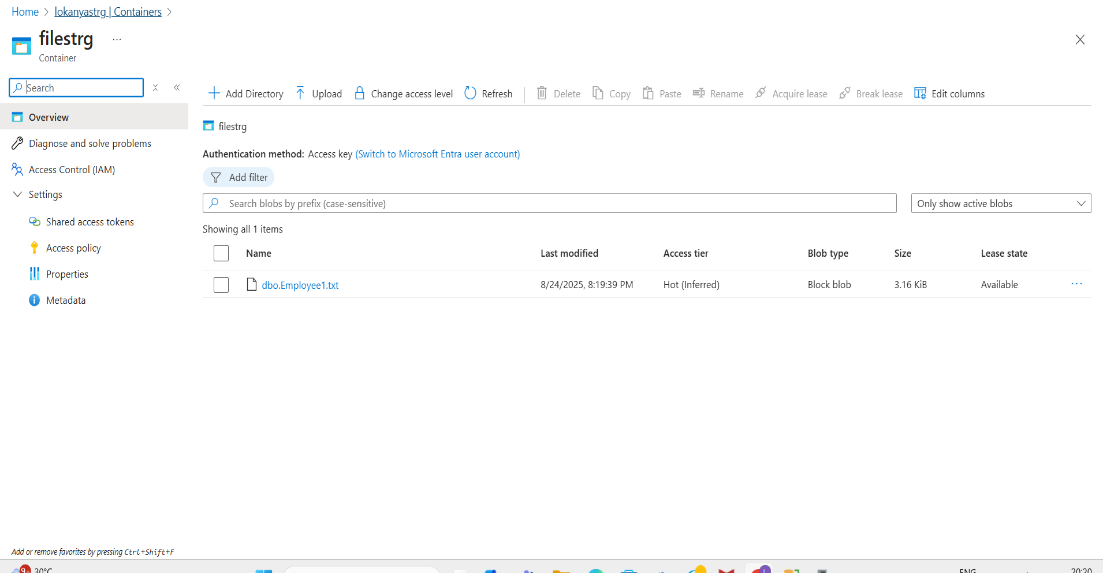
**22. Publish the Pipeline**

Back on the main pipeline page, click “Publish All” and add a trigger to execute the pipeline.



**23. Verify Data Transfer**

After execution, inspect your Azure Blob Storage container to confirm that your SQL Server table data is present in .txt format.

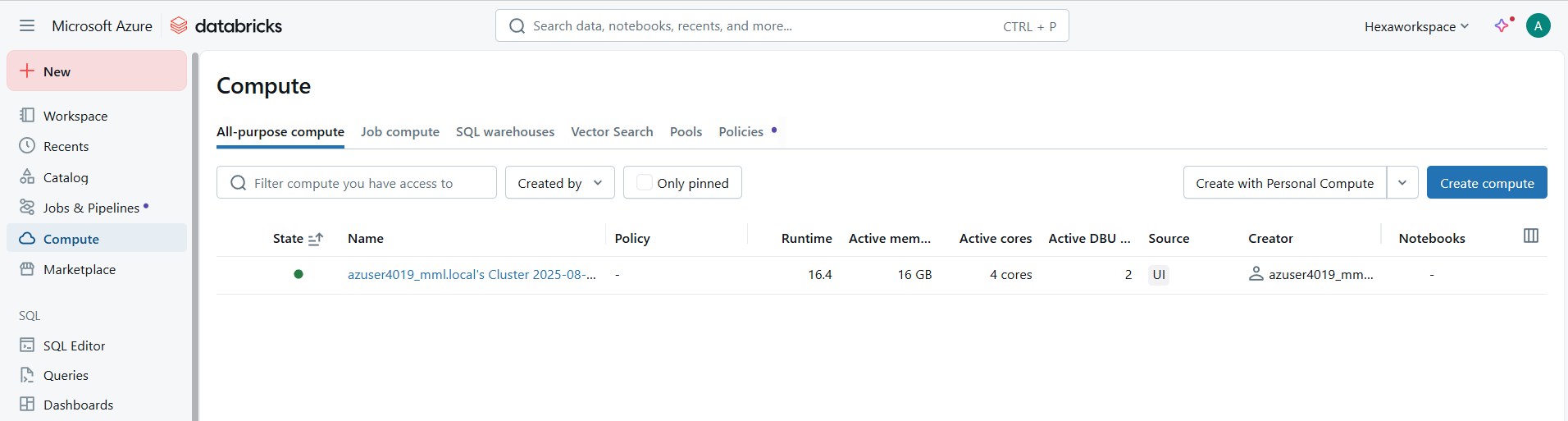


**24. Data Processing in Azure Databricks**

Next, use Azure Databricks to perform data processing tasks on the transferred data.

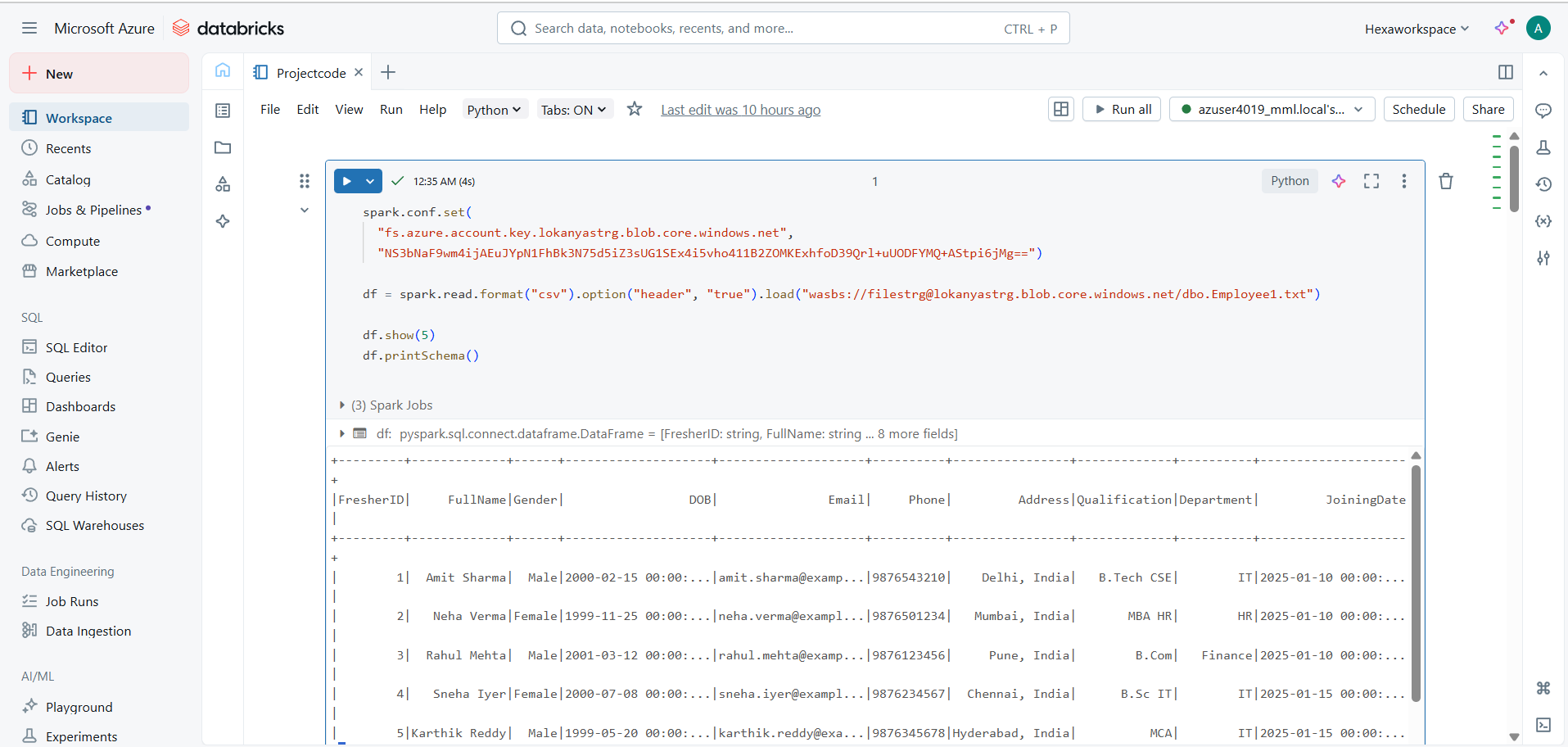
**25. Set Up Azure Databricks Workspace and Cluster**

Create a new Databricks workspace and cluster, ensuring the cluster is running during computations.



**26. Process Data in Databricks Notebook**

Create a new notebook and implement code for tasks such as data cleaning, filtering, renaming columns, performing transformations, and saving processed outputs back to Azure Blob Storage.



**ABOUT THE PROJECT : Files, Tables, and Folders Used**

* **On-premises SQL Server:**

Database: cloudproject

Table : Employee1

* **Self-hosted Integration Runtime:**

Installed on Windows machine within same network as SQL Server.

* **Azure Blob Storage:**

Container: filestrg

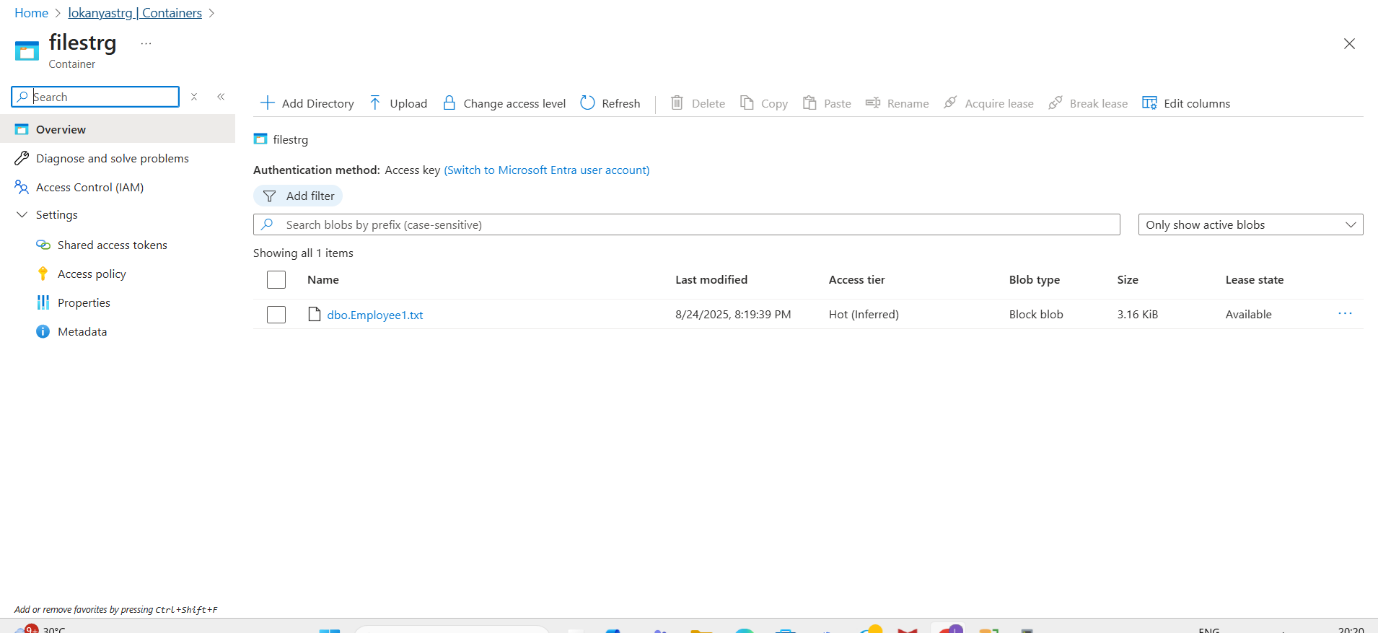
Uploaded files: dbo.Employee1.txt

* **Azure Databricks:**

Notebooks reference: Project.ipynb

**SUCCESSFUL OUTPUT GENERATED**

Migration of data from an on-premises SQL Server database to Azure cloud using Azure Data Factory.

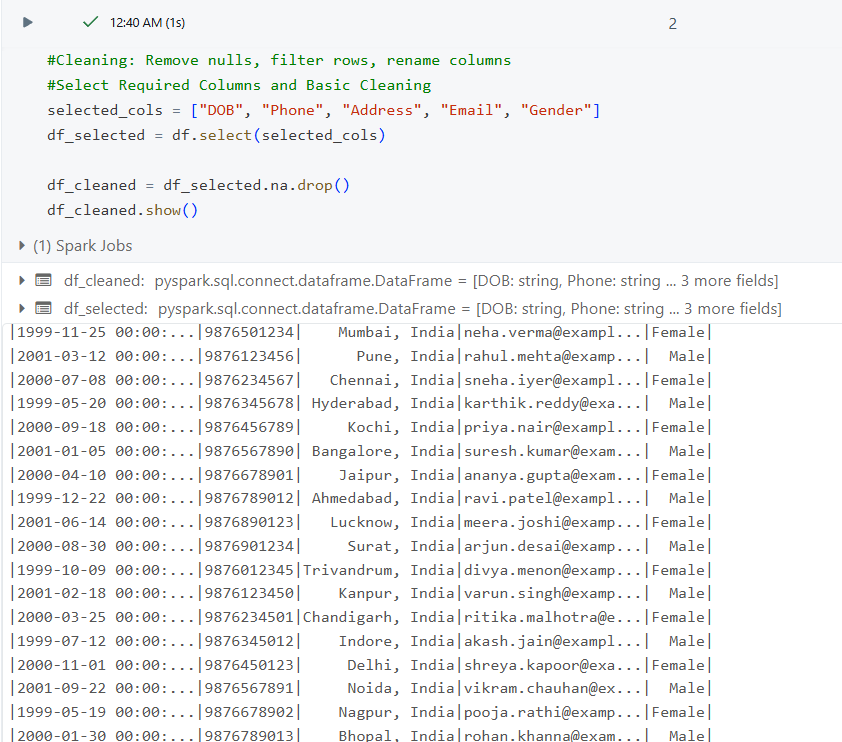
****

Performing data processing tasks in Azure Databricks.

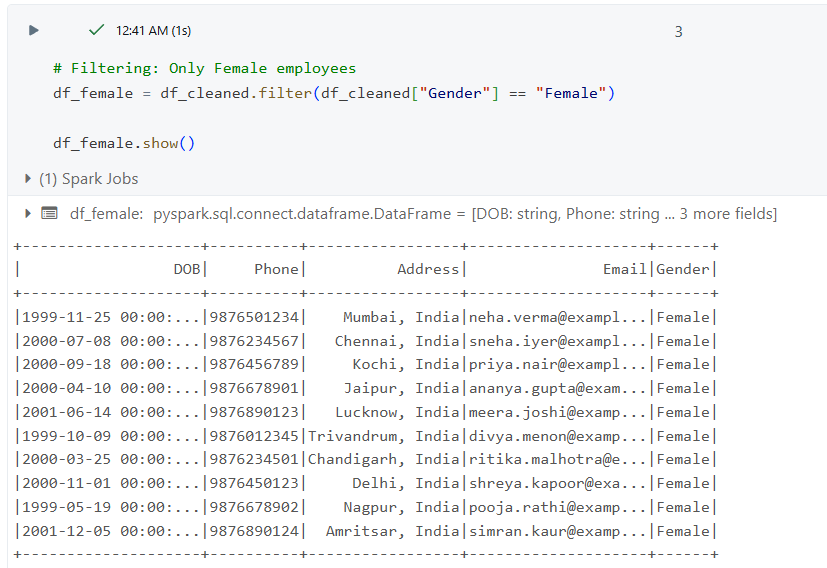
**Data Loading**



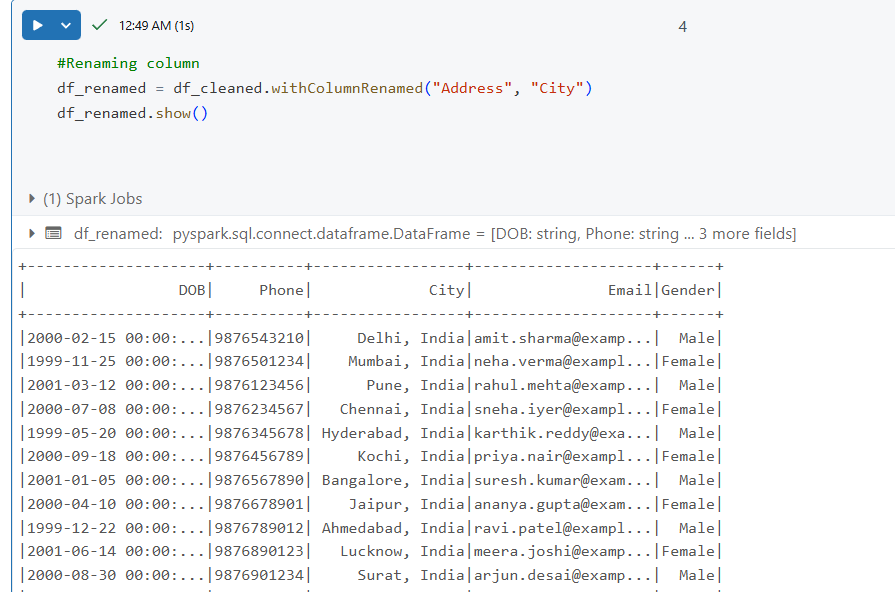
**Cleaning Data**



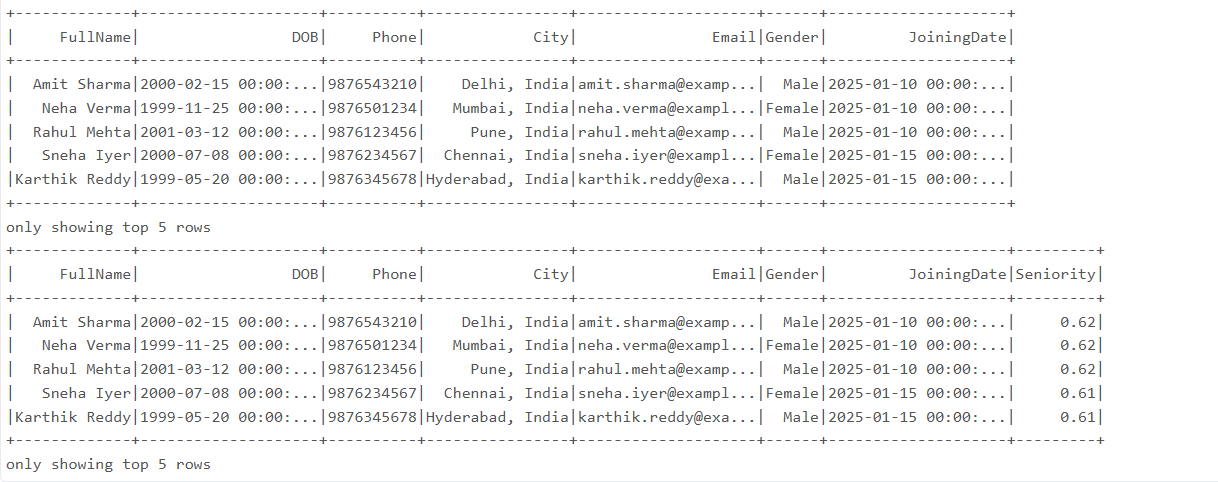
**Filtering Data**



**Renaming Data**



**Transformations**



**Saving Data**



**CONCLUSION**

This project successfully demonstrates a secure, end-to-end solution for moving and processing data between on-premises and Azure cloud environments. By leveraging Azure Data Factory’s robust orchestration and its Self-hosted Integration Runtime, we achieved automated extraction and transfer of data from an on-premises SQL Server database to Azure Blob Storage. The integration of Azure Databricks for scalable data processing illustrates how cloud-native tools can efficiently clean, transform, and analyze enterprise datasets.

Through this implementation, we showcased the power of hybrid data engineering—bridging legacy infrastructure with modern cloud analytics to enable timely, cost-effective, and governed data flows. The solution not only streamlines migration and processing but also lays the foundation for advanced analytics and business intelligence in the cloud. This project demonstrates best practices in cloud integration, security, and scalability, providing a reusable architecture for future enterprise data initiatives.