

# Project: Hybrid Cloud Data Movement

## Team Members-

Subrat Shukla, DE-1

Aniroop Gupta, DE-1

Harish ER, DE-1

- **Project Overview:**

Implement a solution that involves moving data between on-premises data sources and Azure cloud using Azure Data Factory, and perform data processing tasks in Azure Databricks.

- **Project Description:**

This project focuses on implementing a scalable and efficient solution to move data between on-premises data sources and Azure cloud while performing advanced data processing using Azure Data Factory (ADF) and Azure Databricks. The aim is to establish a robust data pipeline for seamless integration, transformation, and analytics in the cloud environment.

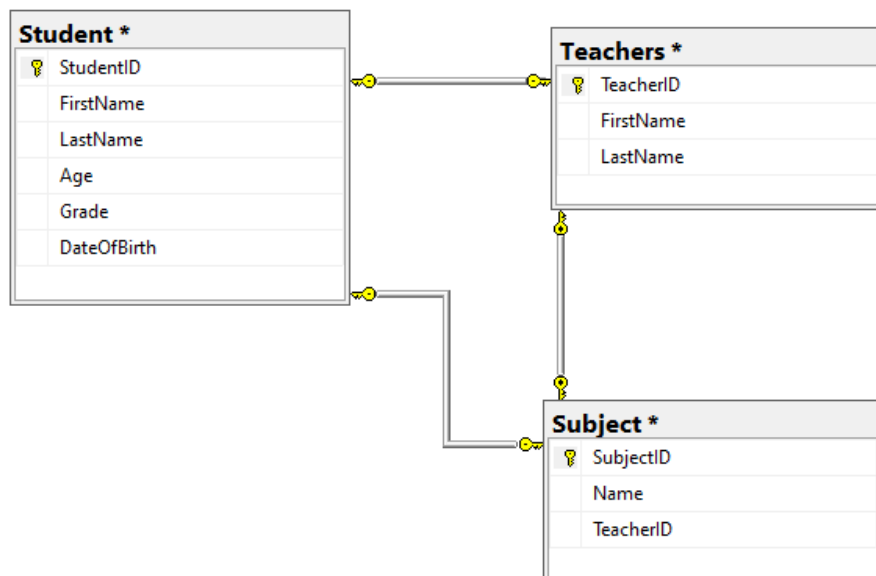
- **Data Overview:**

The **Student Database** includes a primary table named **Student**, designed to manage essential information about students. This table contains the following columns:

- **StudentID** (unique identifier for each student),
- **FirstName** and **LastName** (to capture the student's name),
- **Age** (representing the student's current age),
- **Gender** (to identify the gender of the student), and
- **DOB** (Date of Birth, to record the student's birth date).

The structure ensures the efficient storage and retrieval of student records for administrative or analytical purposes. This database is crucial for maintaining up-to-date student information and can support operations such as reporting, data analysis, and integration with other systems.

- **ER Diagram:**



- **How it works:**

We created a sql database named StudentDB, in that we made tables names as Student, Teachers, Subject with required details. Here is the overview of data file.

	StudentID	FirstName	LastName	Age	Grade	DateOfBirth
1	1	Ram	Sharma	14	8th	2010-06-15
2	2	Rohit	Singh	13	8th	2011-04-21
3	3	Vikas	Garg	15	9th	2009-01-30
4	4	Subrat	Shukla	14	8th	2010-08-10
5	6	Harsh	Gupta	16	10th	2008-09-17
6	7	Grace	Hopper	13	7th	2011-12-09

- **Execution Overview:**

The Hybrid Cloud Data Movement project involves creating an end-to-end pipeline to move and process data using Azure Data Factory and Azure Databricks.

## Steps to Implement the Solution:

### 1. Understanding Requirements:

- Identify the **on-premises data source(s)** (e.g., SQL Server, Oracle, flat files).
- Define the **destination** in Azure (e.g., Azure Blob Storage, Azure SQL Database, Data Lake).
- Specify data processing requirements (e.g., data transformations, cleaning, aggregations).

### 2. Setting Up Azure Environment:

- Creating a new Azure Data Factory (ADF) instance.
- Creating a new storage account container in Azure.
- Set up Azure Databricks workspace in Azure.
- Create linked services in ADF to connect with on-premises sources using the **Self-hosted Integration Runtime** for secure data movement.

### 3. Data Movement (ADF):

- **Pipelines:**
  - Build an ADF pipeline to:
    1. Extract data from on-premises sources.
    2. Load it into Azure storage (Blob Storage or Data Lake).
- **Triggers:**
  - Schedule the pipelines using triggers (time-based or event-driven).

### 4. Data Processing (Azure Databricks):

- **Databricks Notebooks:**
  - Develop notebooks to process the ingested data.
  - Perform tasks like filtering, aggregations, joins, and formatting data for downstream use.
- **Integrate with ADF:**
  - Use ADF to trigger Databricks jobs by calling its API or Notebook activity.

## 5. End-to-End Workflow:

- Combine data movement and data processing workflows into a cohesive ADF pipeline.
- Ensure proper error handling, logging, and retries for resilience.

## ● Azure resources used for this project:

1. Azure Data Factory (ADF)
2. Azure Storage Account
3. Azure Data factory
4. Azure Databricks

## ● Project Requirements:

1. Create new Azure Data Factory
2. Create a new container from Storage accounts, to store data.
3. Download on premise data store, we used SQL Server.
4. Download Microsoft Integration Runtime.

## ● Tasks Performed:

1. Configured **Azure Data Factory (ADF)** and set up linked services to connect with on-premises data sources and Azure storage.
2. Created integration runtime to connect self-host node to cloud service
3. Built pipelines in ADF to **extract, transform, and load (ETL)** data from on-premises systems to Azure Blob Storage.
4. Scheduled and tested pipelines using **triggers** (time-based or manual runs).
5. Connected SQL Server Database with the Azure Blob Storage.
6. Integrated Databricks with ADF for automated workflow execution.
7. Developed **Databricks notebooks** to perform data transformations.
8. Conducted thorough testing to ensure data integrity during migration.

This implementation enables a seamless, scalable pipeline for hybrid cloud data movement and processing, leveraging Azure's capabilities.

## • Analysis Results:

### 1. Creating new Data Factory:

The screenshot shows the 'Overview' page for a Microsoft Data Factory deployment. The deployment is titled 'Microsoft.DataFactory-20241218121922'. A green checkmark indicates 'Your deployment is complete'. The deployment details show the name, subscription (MML Learners), resource group (rg-azuser2374\_mml.local-CTjmW), start time (12/18/2024, 12:20:49 PM), and correlation ID. The page includes a sidebar with 'Overview', 'Inputs', 'Outputs', and 'Template'. On the right, there are links for 'Cost management', 'Microsoft Defender for Cloud', and 'Free Microsoft tutorials'.

### 2. Creating storage account:

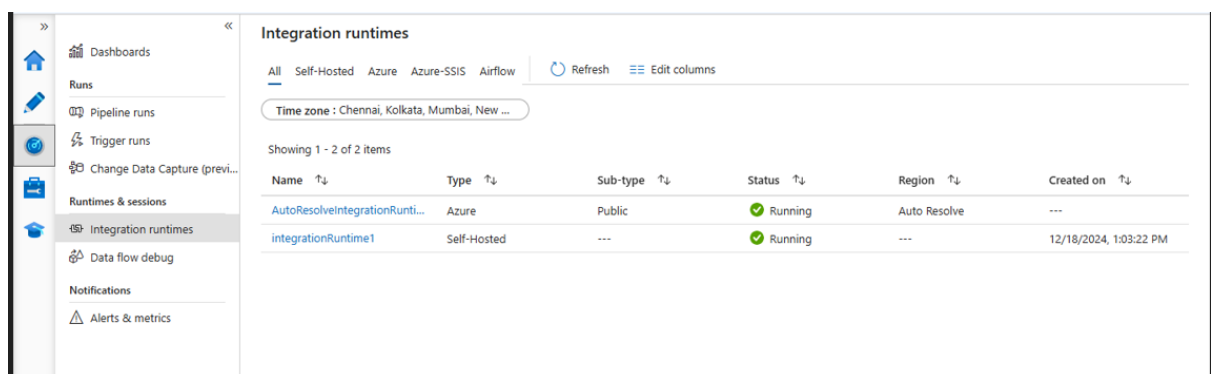
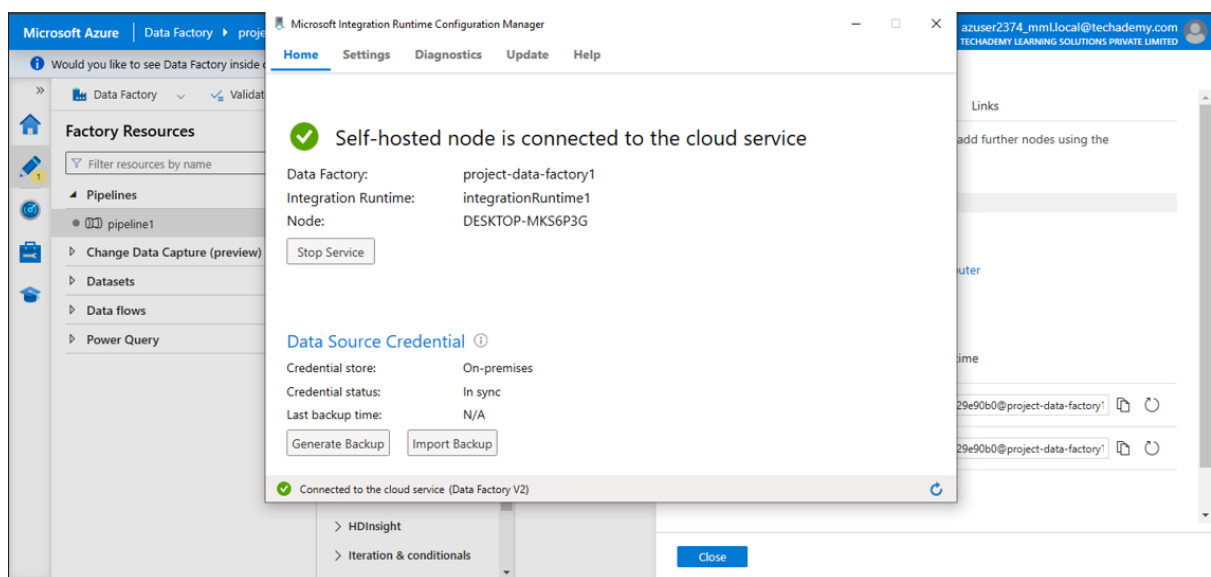
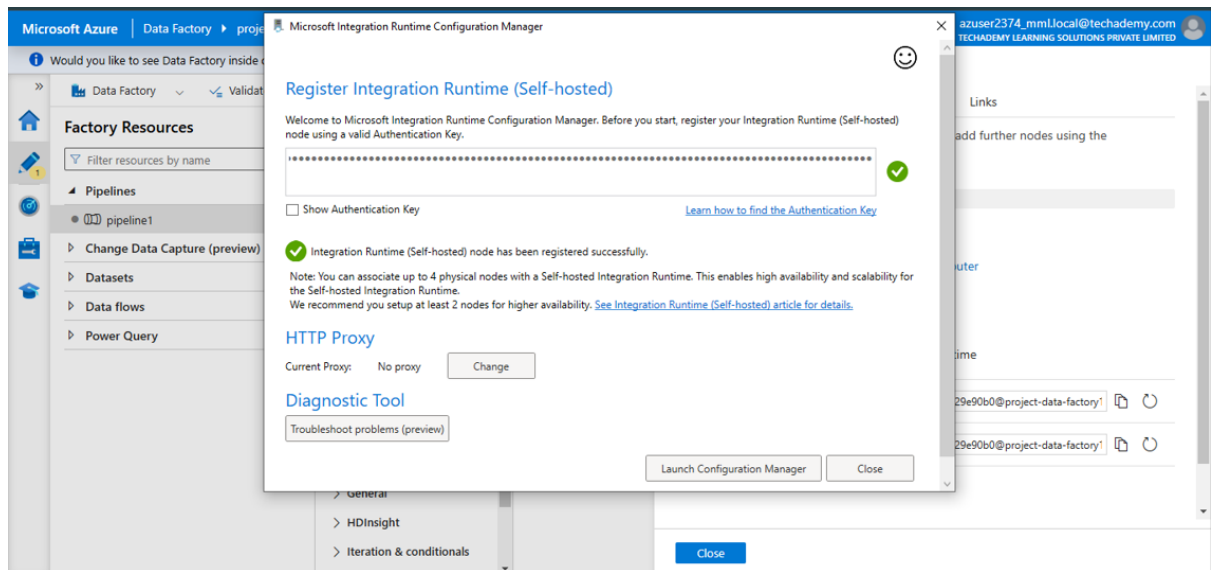
The screenshot shows the 'Overview' page for an Azure Storage Account deployment. The deployment is titled 'projectstorageacc1\_1734504756877'. A green checkmark indicates 'Your deployment is complete'. The deployment details show the name, subscription (MML Learners), resource group (rg-azuser2374\_mml.local-CTjmW), start time (12/18/2024, 12:22:51 PM), and correlation ID. The page includes a sidebar with 'Overview', 'Inputs', 'Outputs', and 'Template'. On the right, there are links for 'Cost Management', 'Microsoft Defender for Cloud', and 'Free Microsoft tutorials'.

### 3. Creating database on premise SQL Server:

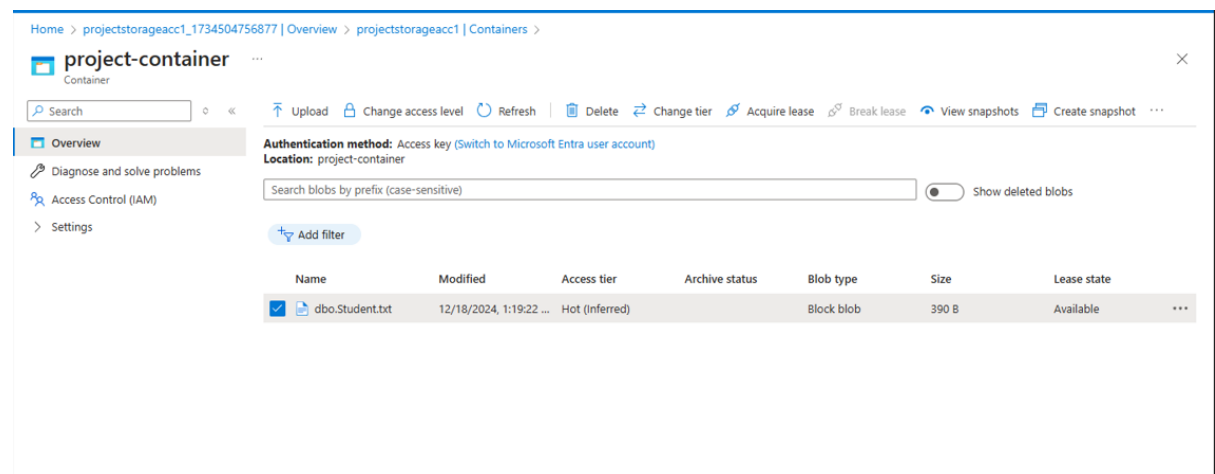
The screenshot shows a SQL Server Enterprise Manager window with a diagram of a database named 'StudentDatabase.sql...IDB (project2 (74))'. The diagram shows a 'CREATE DATABASE SchoolDB;' statement, followed by a comment '-- Create the Student Table', and then 'CREATE TABLE Student (' with columns: StudentID INT PRIMARY KEY, FirstName NVARCHAR(50), LastName NVARCHAR(50), Age INT, Grade NVARCHAR(10), DateOfBirth DATE NULL. Below the diagram, the 'Results' tab shows a table with 6 rows of data:

StudentID	FirstName	LastName	Age	Grade	DateOfBirth
1	Ram	Sharma	14	8th	2010-06-15
2	Rohit	Singh	13	8th	2011-04-21
3	Vikas	Garg	15	9th	2009-01-30
4	Subrat	Shukla	14	8th	2010-08-10
5	Harsh	Gupta	16	10th	2008-09-17
6	Grace	Hopper	13	7th	2011-12-09

## 4. Integrating Integration Runtime to connect self-host node with cloud:

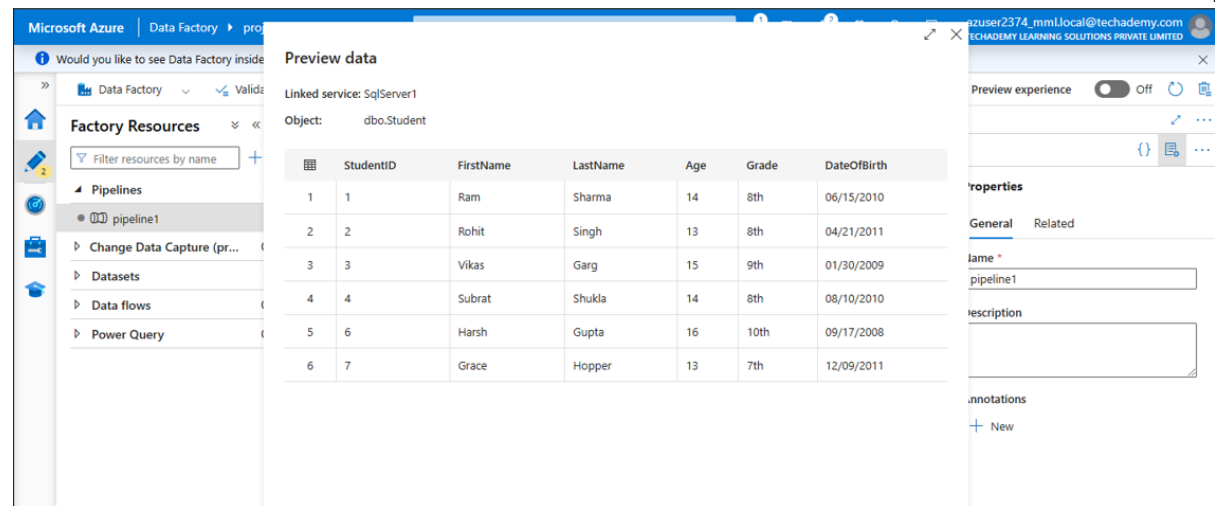


## 5. Migrated data to the Storage container:



The screenshot shows the Azure Storage Explorer interface. The breadcrumb path is Home > projectstorageacct1\_1734504756877 | Overview > projectstorageacct1 | Containers. The selected container is 'project-container'. The authentication method is 'Access key (Switch to Microsoft Entra user account)'. The location is 'project-container'. A search bar is present with the text 'Search blobs by prefix (case-sensitive)'. A table lists the blobs in the container:

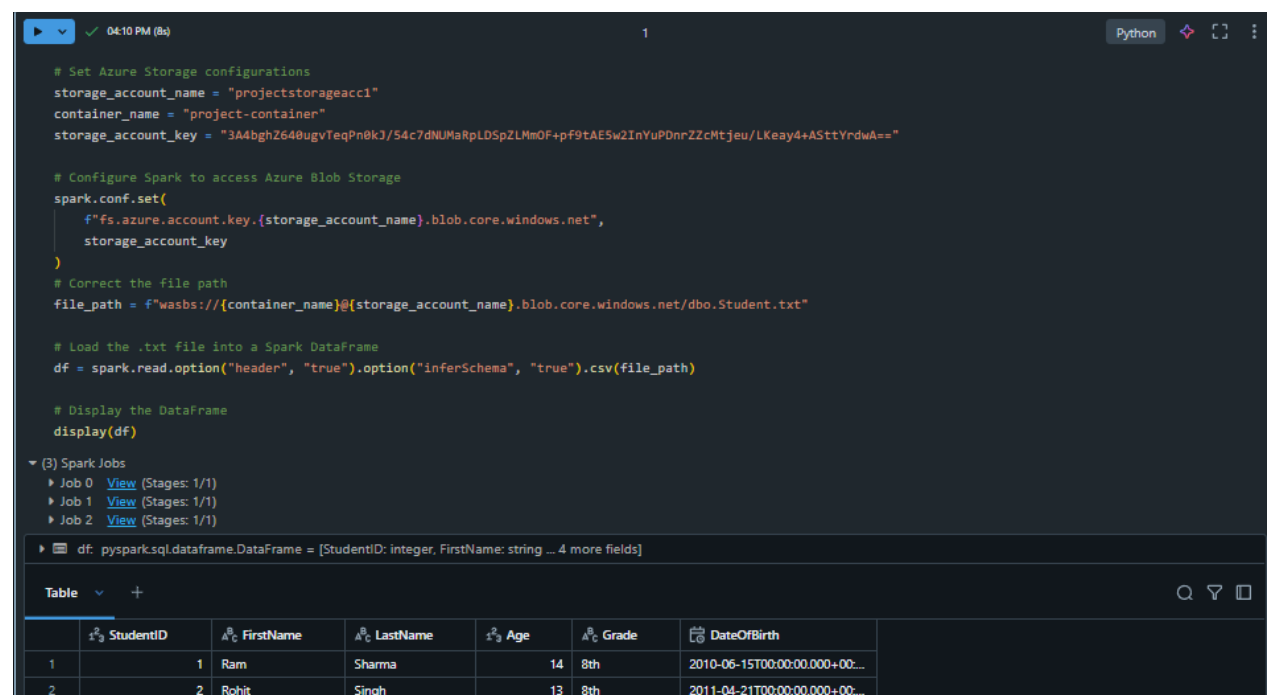
Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
dbo.Student.txt	12/18/2024, 1:19:22 ...	Hot (Inferred)		Block blob	390 B	Available

The screenshot shows the Microsoft Azure Data Factory interface. The breadcrumb path is Microsoft Azure | Data Factory | projectstorageacct1\_1734504756877 | Pipelines. The selected pipeline is 'pipeline1'. The 'Preview data' window is open, showing the linked service 'SqlServer1' and the object 'dbo.Student'. The preview displays a table with 6 rows of student data:

StudentID	FirstName	LastName	Age	Grade	DateOfBirth
1	Ram	Sharma	14	8th	06/15/2010
2	Rohit	Singh	13	8th	04/21/2011
3	Vikas	Garg	15	9th	01/30/2009
4	Subrat	Shukla	14	8th	08/10/2010
5	Harsh	Gupta	16	10th	09/17/2008
6	Grace	Hopper	13	7th	12/09/2011

## 6. Integrate Databricks with ADF for automated workflow:



The screenshot shows a Databricks notebook with a Python script that reads data from Azure Storage and displays it as a table. The script sets the Azure Storage account name, container name, and key. It then configures Spark to access the Azure Blob Storage and reads the data from the 'dbo.Student.txt' file into a Spark DataFrame. The DataFrame is displayed using the 'display(df)' function.

```
# Set Azure Storage configurations
storage_account_name = "projectstorageacct1"
container_name = "project-container"
storage_account_key = "3A4bghZ640ugvTegPn0k3/54c7dNLMaRpLDSPZLMmOF+pf9tAE5w2InYuPDnrZZcMtjeu/LKeay4+ASttYrdwA=="

# Configure Spark to access Azure Blob Storage
spark.conf.set(
    f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net",
    storage_account_key
)

# Correct the file path
file_path = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/dbo.Student.txt"

# Load the .txt file into a Spark DataFrame
df = spark.read.option("header", "true").option("inferSchema", "true").csv(file_path)

# Display the DataFrame
display(df)
```

The output of the script is a table with 6 rows of student data:

StudentID	FirstName	LastName	Age	Grade	DateOfBirth
1	Ram	Sharma	14	8th	2010-06-15T00:00:00.000+00:00
2	Rohit	Singh	13	8th	2011-04-21T00:00:00.000+00:00

```
04:13 PM (1s) 2

from pyspark.sql.functions import col

# Example: Filter rows where 'age' > 14
filtered_df = df.filter(df['Age'] > 13)

# Example: Add a new column by doubling the 'age' value
processed_df = filtered_df.withColumn("new_column", col("Age") * 2)

# Display the processed DataFrame
display(processed_df)
```

▶ (1) Spark Jobs

- ▶ filtered\_df: pyspark.sql.dataframe.DataFrame = [StudentID: integer, FirstName: string ... 4 more fields]
- ▶ processed\_df: pyspark.sql.dataframe.DataFrame = [StudentID: integer, FirstName: string ... 5 more fields]

	StudentID	FirstName	LastName	Age	Grade	DateOfBirth	new_column
1	1	Ram	Sharma	14	8th	2010-06-15T00:00:00.000+00:...	28
2	3	Vikas	Garg	15	9th	2009-01-30T00:00:00.000+00:...	30
3	4	Subrat	Shukla	14	8th	2010-08-10T00:00:00.000+00:...	28
4	6	Harsh	Gupta	16	10th	2008-09-17T00:00:00.000+00:...	32

4 rows | 0.73 seconds runtime Refreshed 28 minutes ago

```
04:13 PM (2s) 3

# Define the output path for processed data
output_path = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/processed_data/"

# Write the processed DataFrame back as a CSV file
processed_df.write.mode("overwrite").option("header", "true").csv(output_path)

# Print confirmation
print(f"Data successfully written to: {output_path}")
```

▶ (1) Spark Jobs

Data successfully written to: wasbs://project-container@projectstorageacc1.blob.core.windows.net/processed\_data/

```
04:13 PM (<1s) 4

# Use Databricks utilities to list the files in the output path
dbutils.fs.ls(output_path)
```

```
[FileInfo(path='wasbs://project-container@projectstorageacc1.blob.core.windows.net/processed_data/_SUCCESS', name='_SUCCESS', size=0, modificationTime=1734518614000),
 FileInfo(path='wasbs://project-container@projectstorageacc1.blob.core.windows.net/processed_data/_committed_8451362789162339081', name='_committed_8451362789162339081', size=111, modificationTime=1734518614000),
 FileInfo(path='wasbs://project-container@projectstorageacc1.blob.core.windows.net/processed_data/_started_8451362789162339081', name='_started_8451362789162339081', size=0, modificationTime=1734518614000),
 FileInfo(path='wasbs://project-container@projectstorageacc1.blob.core.windows.net/processed_data/part-00000-tid-8451362789162339081-8afef67a-a22a-433e-ace4-ed92696ab05a-4-1-c000.csv', name='part-00000-tid-8451362789162339081-8afef67a-a22a-433e-ace4-ed92696ab05a-4-1-c000.csv', size=259, mo
```

This ETL process ensures the data first gets converted into processed form and then migrates the processed data to the Blob Storage Container.

Let us see the final processed data folder in Blob Storage.



## 7. Processed Data migrated to Blob Storage:

Home > Storage accounts > projectstorageacc1 | Containers >

project-container

Search

Upload Change access level Refresh Delete Change tier Acquire lease Break lease View snapshots Create snapshot

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: project-container

Search blobs by prefix (case-sensitive) Show deleted blobs

Add filter

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> processed_data						-
<input type="checkbox"/> dbo.Student.txt	12/18/2024, 1:19:22 ...	Hot (Inferred)		Block blob	390 B	Available
<input type="checkbox"/> processed_data	12/18/2024, 4:13:34 ...	Hot (Inferred)		Block blob	0 B	Available

- **Conclusion:**

The **Hybrid Cloud Data Movement Project** successfully demonstrates a robust and efficient solution for integrating on-premises data with the Azure cloud. By leveraging **Azure Data Factory** for seamless data movement and **Azure Databricks** for advanced processing, the project enables scalable, high-performance data pipelines suitable for real-world scenarios. The solution ensures secure, reliable, and automated workflows, optimized to handle complex transformations and large data volumes. With comprehensive testing, monitoring, and documentation, this project lays a strong foundation for hybrid cloud implementations, empowering businesses to unlock insights and drive data-driven decisions with ease.

---

**Submitted By-**

**Subrat Shukla, DE-1**

**Harish ER, DE-1**

**Anirop Gupta, DE-1**

**--Thank You!**