Automated ETL with data lake storage

Project Overview:

Our project aims to automate the Extract, Transform, Load (ETL) process using Azure services. The process is divided into three main stages. Firstly, we'll perform ETL from blob to ADLS, facilitating the efficient transfer of data between storage instances. Next, we'll focus on ensuring data quality and monitoring by leveraging Azure Databricks. Finally, we'll integrate continuous integration and continuous deployment (CI/CD) pipelines using Azure DevOps to manage ETL scripts effectively.

Project Requirements:

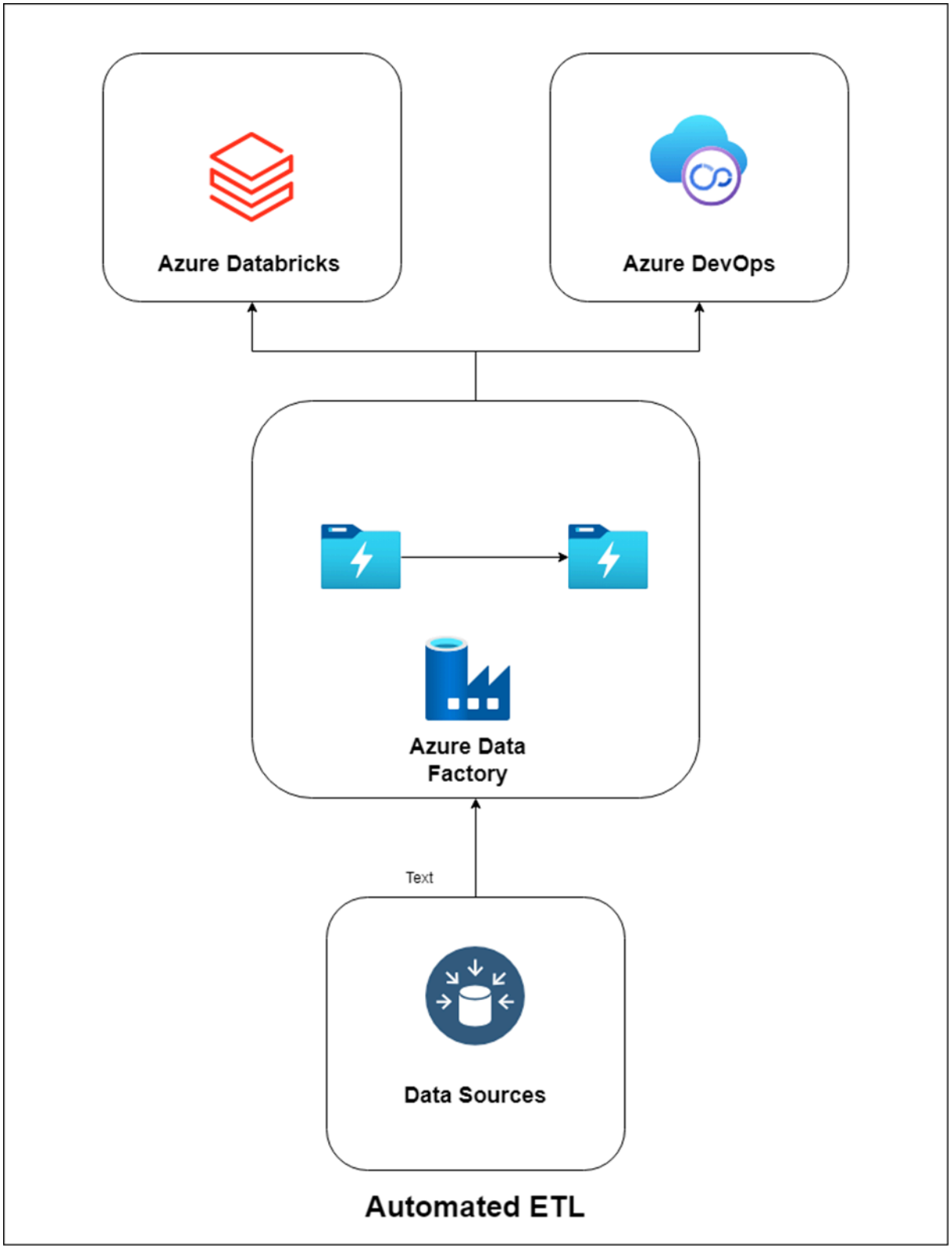
1. Develop an ETL pipeline to transfer data seamlessly from one ADLS to another while ensuring data integrity and efficiency.

2. Implement comprehensive quality assurance measures and monitoring operations on the data to maintain high data standards.

3. Integrate CI/CD for version control, automated testing, and deployment of ETL scripts to streamline the development process and ensure reliability.

4. Ensure scalability, reliability, and efficiency of the ETL process to handle large volumes of data efficiently and effectively.

Architecture Diagram:



Technologies/Tools Used:

1. **Azure Data Lake Storage**: Source and destination for ETL.
2. **Azure Databricks:** Data quality assurance, monitoring, and complex transformations.
3. **Azure DevOps:** CI/CD for version control and automated deployment.
4. **Various data sources:** (List the sources you'll be working with)

Execution Overview:

1. ETL from Blob to ADLS:

- Configure Azure Data Factory to orchestrate data movement between ADLS instances.
- Define pipelines to extract data from the source ADLS, transform it as necessary, and load it into the target ADLS.
- Schedule and execute ETL jobs to ensure timely and efficient data transfer.

2. Data Quality Assurance and Monitoring with Databricks:

- Utilize Azure Databricks to perform comprehensive data quality checks, including validation, cleansing, and enrichment.
- Develop Databricks notebooks to implement quality assurance measures based on defined business rules and requirements.
- Implement monitoring processes to track data trends, anomalies, and overall data health.

3.CI/CD Integration:

- Set up CI/CD pipelines in Azure DevOps to manage ETL scripts effectively.
- Implement version control to track changes and collaborate on script development.
- Automate testing processes to ensure script reliability and accuracy.
- Enable automated deployment to streamline the deployment process and ensure consistent releases.

How it Works:

1. ETL from ADLS to ADLS:

- Azure Data Factory coordinates the transfer of data between ADLS instances, ensuring efficient and reliable data movement.
- Data is extracted from the source ADLS, undergoes necessary transformations, and is loaded into the target ADLS according to defined pipelines.

2.Data Quality Assurance and Monitoring with Databricks:

- Azure Databricks performs in-depth data quality checks and monitoring operations to ensure data accuracy, consistency, and integrity.
- Custom Databricks notebooks are developed to implement specific quality assurance measures tailored to the project requirements.
- Continuous monitoring processes are established to track data health and identify any issues or anomalies promptly.

3.CI/CD Integration:

- Azure DevOps facilitates seamless collaboration and version control for ETL script development.

- CI/CD pipelines automate the testing and deployment of ETL scripts, ensuring rapid and reliable delivery of changes to the production environment.

- Automated testing processes validate script functionality, while automated deployment processes ensure consistent and error-free deployments.

Implementation Process:

1. ETL from ADLS to ADLS:

- Define data pipelines in Azure Data Factory to extract, transform, and load data between ADLS instances.
- Configure data sources, transformations, and destinations within the pipelines.
- Test and validate the ETL pipelines to ensure accurate and efficient data transfer.

2.Data Quality Assurance and Monitoring with Databricks:

- Develop custom Databricks notebooks to implement quality assurance measures, including data validation, cleansing, and enrichment.
- Implement monitoring processes to track data quality metrics and identify any deviations from expected norms.
- Integrate with Azure Data Factory pipelines to incorporate data quality checks into the overall ETL process.

3.CI/CD Integration:

- Set up CI/CD pipelines in Azure DevOps to manage ETL script development, testing, and deployment.

- Configure version control repositories to track changes and facilitate collaboration among team members.
- Define automated testing procedures to verify the functionality and integrity of ETL scripts.
- Implement automated deployment processes to streamline the deployment of ETL scripts to production environments.

Tasks Performed on Databricks:

- **Data Cleansing:** Removing duplicates, handling missing values, standardizing formats, etc.
- **Data Validation:** Checking data integrity, consistency, and compliance with predefined rules and standards.
- **Data Enrichment:** Enhancing data with additional information from external sources or derived calculations.
- **Data Monitoring:** Tracking data trends, identifying anomalies or outliers, and ensuring overall data health.

Final Output - Application:

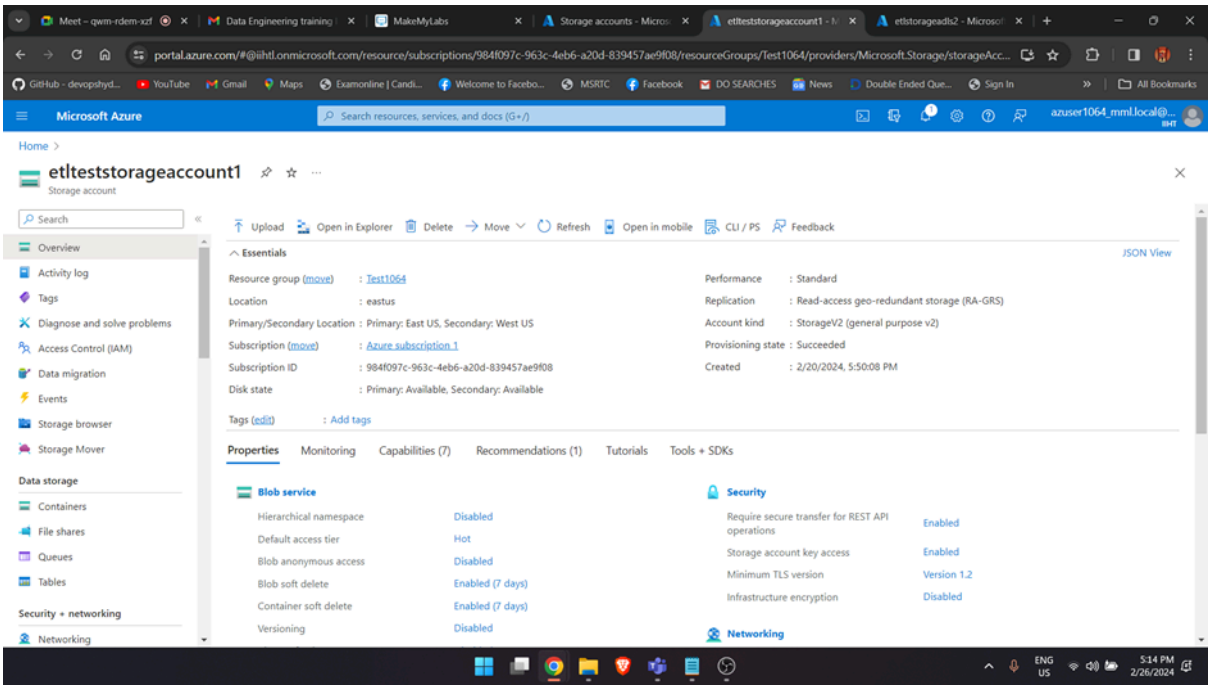
The final output of our project is a robust and scalable ETL solution that automates the transfer of data between Azure Data Lake Storage instances, ensures high data quality and integrity through comprehensive quality assurance measures and monitoring operations performed with Azure Databricks, and facilitates efficient script development, testing, and deployment through CI/CD integration with Azure DevOps. This enables organizations to streamline their data processing workflows, maintain data consistency and reliability, and make informed decisions based on accurate and timely data insights.

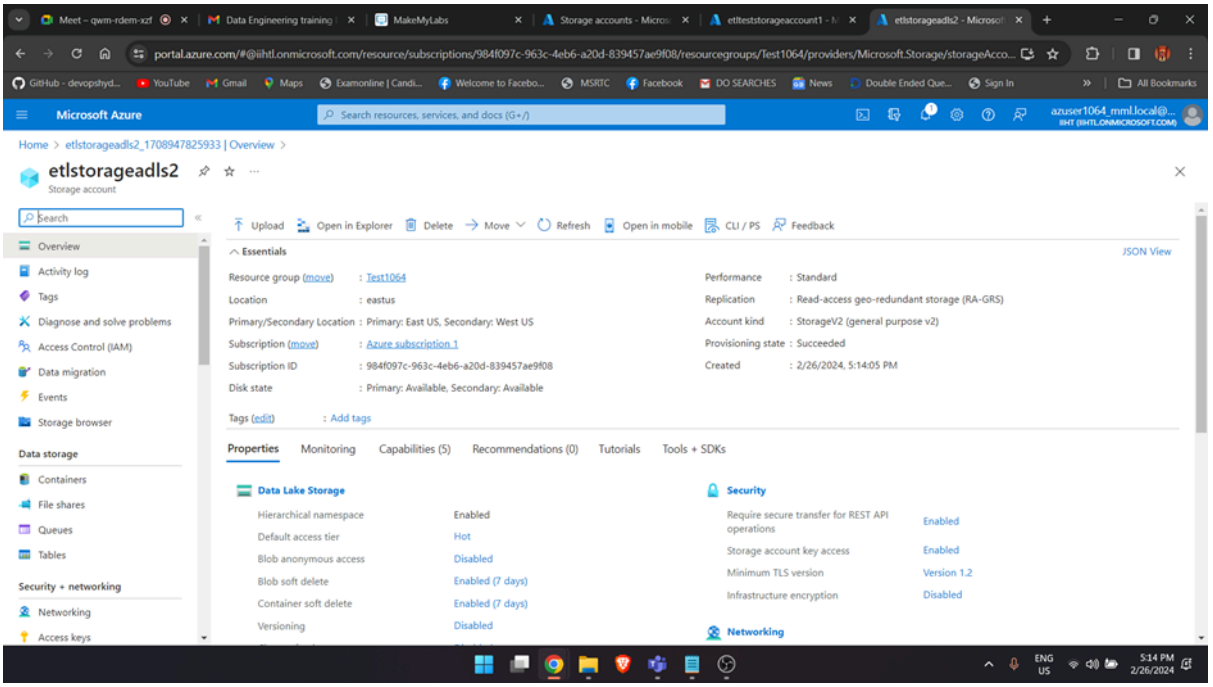
By following this approach, we ensure the seamless flow of data, maintain data integrity, and empower stakeholders with reliable and actionable insights derived from high-quality data.

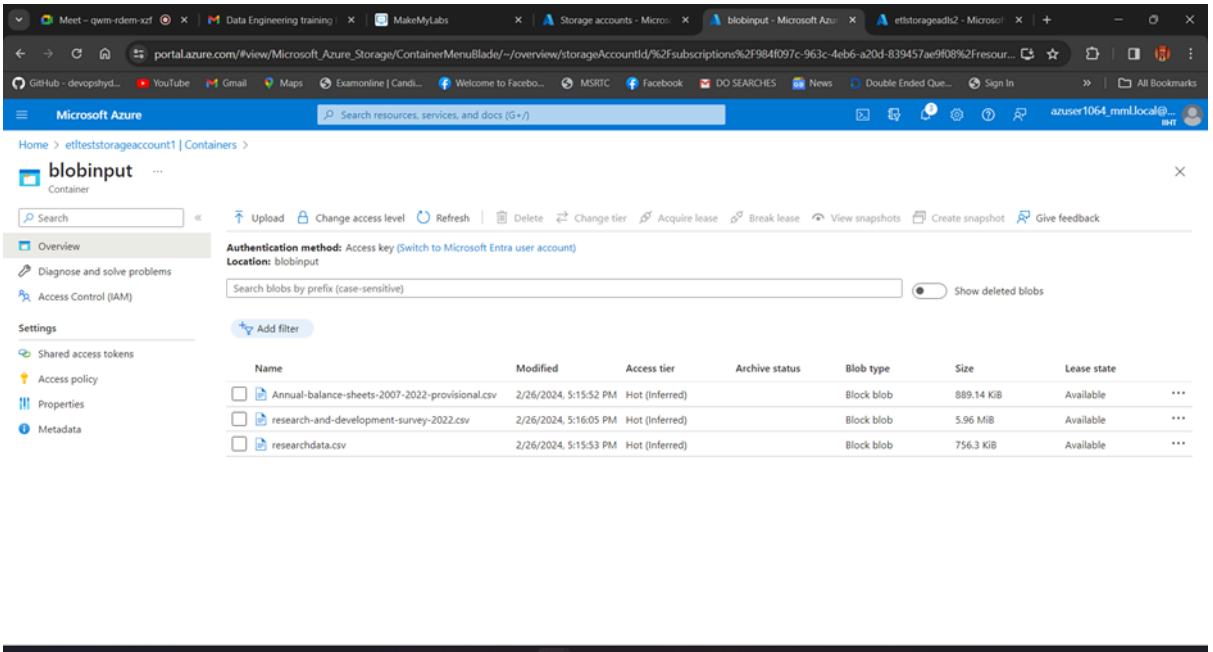
Blob to ADLS :

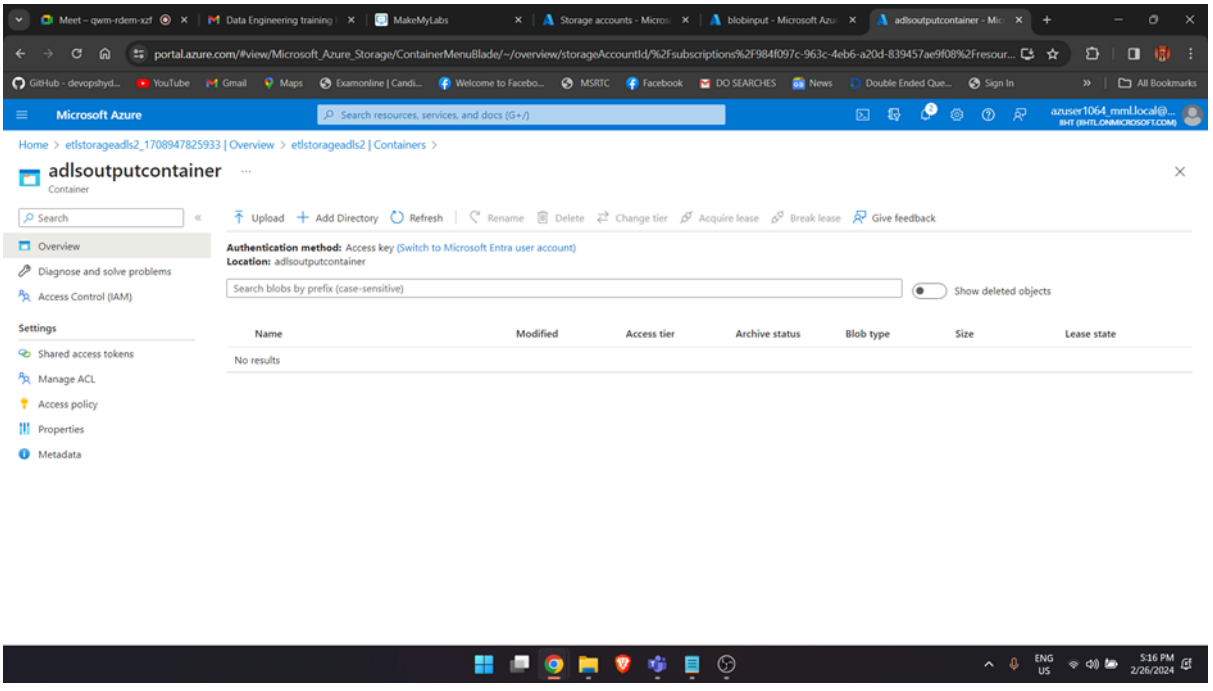
Creating one blob storage and one Azure data lake storage accounts and containers:

- Created one data storage account having an input container with sample files.
- Created one data storage account having a blank output container for copying the input container files.

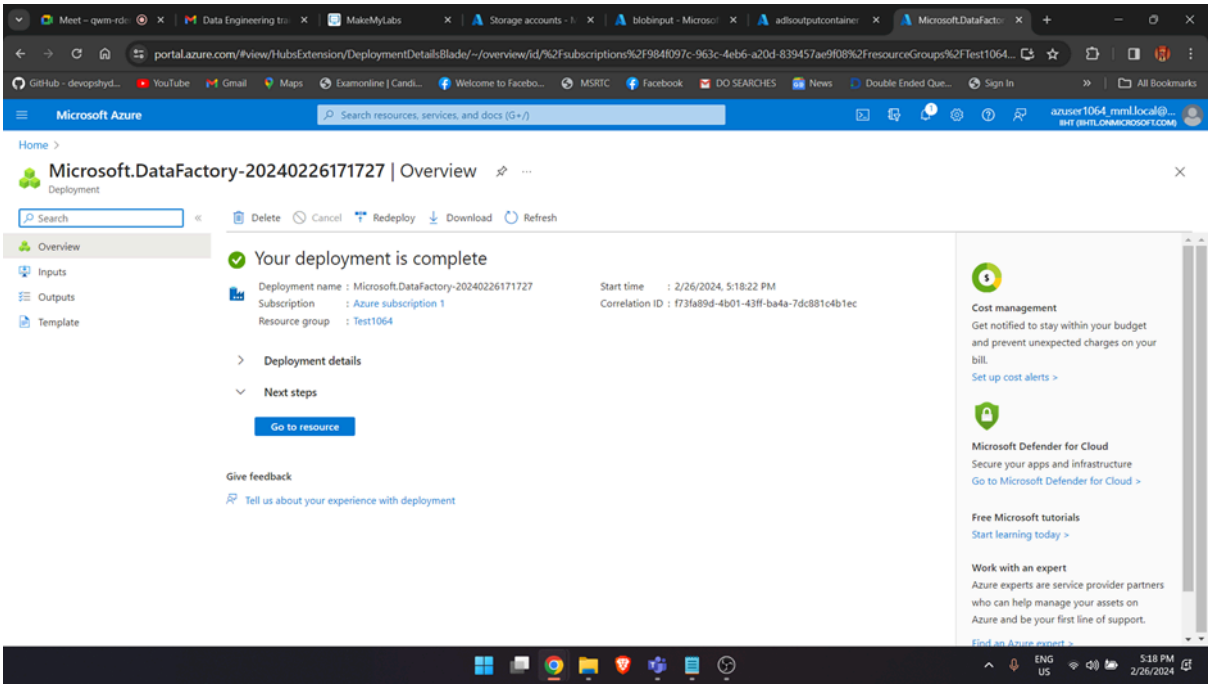


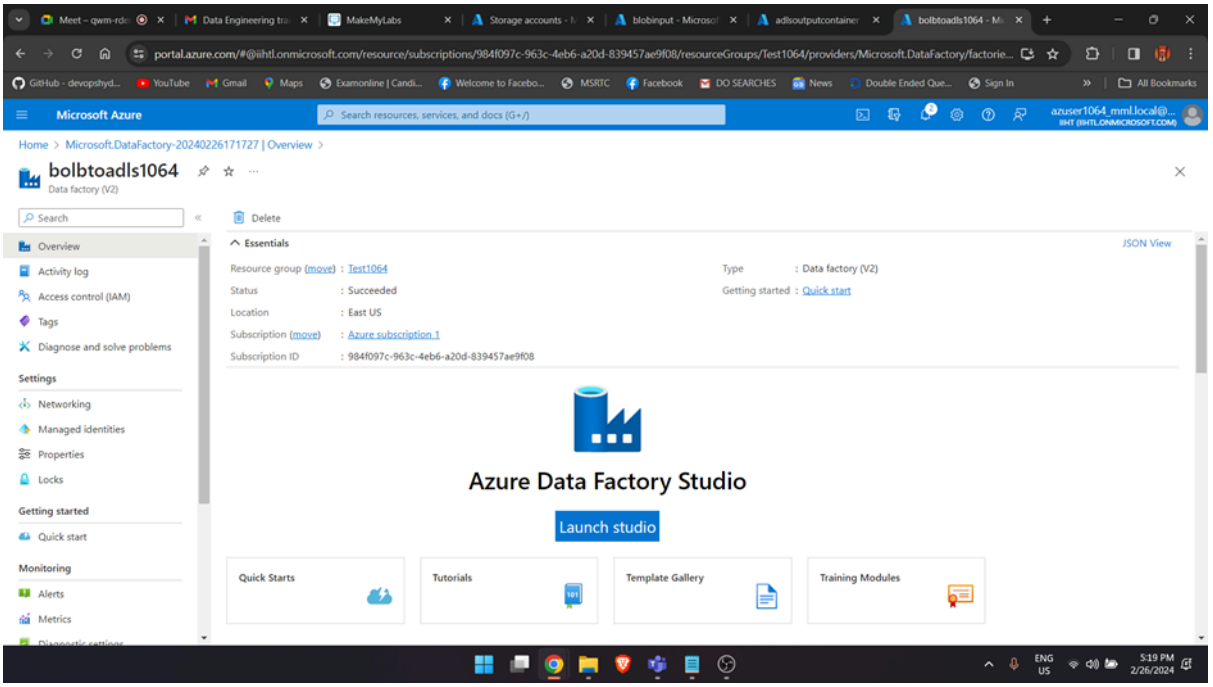

Microsoft Azure
Search resources, services, and docs (G+/)
Home > etlstorageadls2_1708947825933 | Overview >
etlstorageadls2
Storage account
Search
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Data storage
Containers
File shares
Queues
Tables
Security + networking
Networking
Access keys
Upload
Open in Explorer
Delete
Move
Refresh
Open in mobile
CLI / PS
Feedback
Essentials
JSON View
Resource group (move) : Test1064
Location : eastus
Primary/Secondary Location : Primary: East US, Secondary: West US
Subscription (move) : Azure subscription 1
Subscription ID : 984f097c-963c-4eb6-a20d-839457ae9f08
Disk state : Primary: Available, Secondary: Available
Performance : Standard
Replication : Read-access geo-redundant storage (RA-GRS)
Account kind : StorageV2 (general purpose v2)
Provisioning state : Succeeded
Created : 2/26/2024, 5:14:05 PM
Tags (edit) : Add tags
Properties
Monitoring
Capabilities (5)
Recommendations (0)
Tutorials
Tools + SDKs
Data Lake Storage
Hierarchical namespace Enabled
Default access tier Hot
Blob anonymous access Disabled
Blob soft delete Enabled (7 days)
Container soft delete Enabled (7 days)
Versioning Disabled
Security
Require secure transfer for REST API operations Enabled
Storage account key access Enabled
Minimum TLS version Version 1.2
Infrastructure encryption Disabled
Networking
ENG US
5:14 PM 2/26/2024


Microsoft Azure
Search resources, services, and docs (G+/)
Home > etlteststorageaccount1 | Containers >
blobinput
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Access policy
Properties
Metadata
Upload
Change access level
Refresh
Delete
Change tier
Acquire lease
Break lease
View snapshots
Create snapshot
Give feedback
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: blobinput
Search blobs by prefix (case-sensitive)
Show deleted blobs
Add filter
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
Annual-balance-sheets-2007-2022-provisional.csv
2/26/2024, 5:15:52 PM
Hot (Inferred)
Block blob
889.14 KiB
Available
research-and-development-survey-2022.csv
2/26/2024, 5:16:05 PM
Hot (Inferred)
Block blob
5.96 MiB
Available
researchdata.csv
2/26/2024, 5:15:53 PM
Hot (Inferred)
Block blob
756.3 KiB
Available

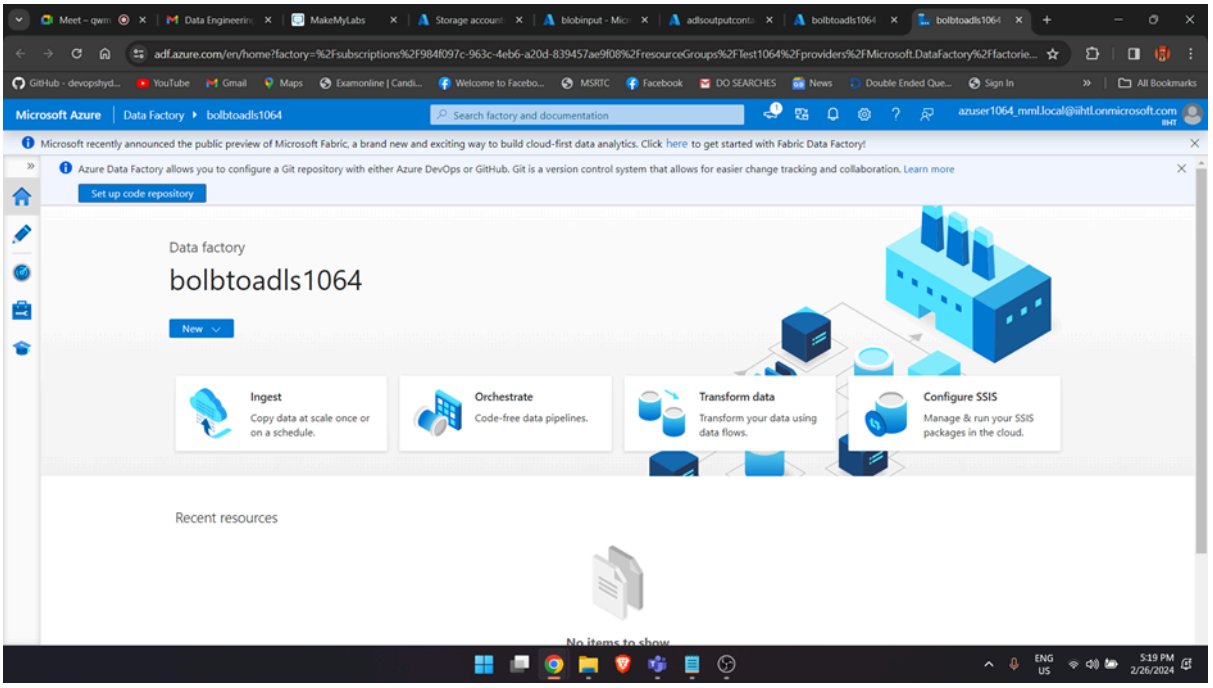

Microsoft Azure
Search resources, services, and docs (G+/)
Home > etlstorageadls2_1708947825933 | Overview > etlstorageadls2 | Containers >
adlsoutputcontainer
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Manage ACL
Access policy
Properties
Metadata
Upload
Add Directory
Refresh
Rename
Delete
Change tier
Acquire lease
Break lease
Give feedback
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: adlsoutputcontainer
Search blobs by prefix (case-sensitive)
Show deleted objects
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
No results

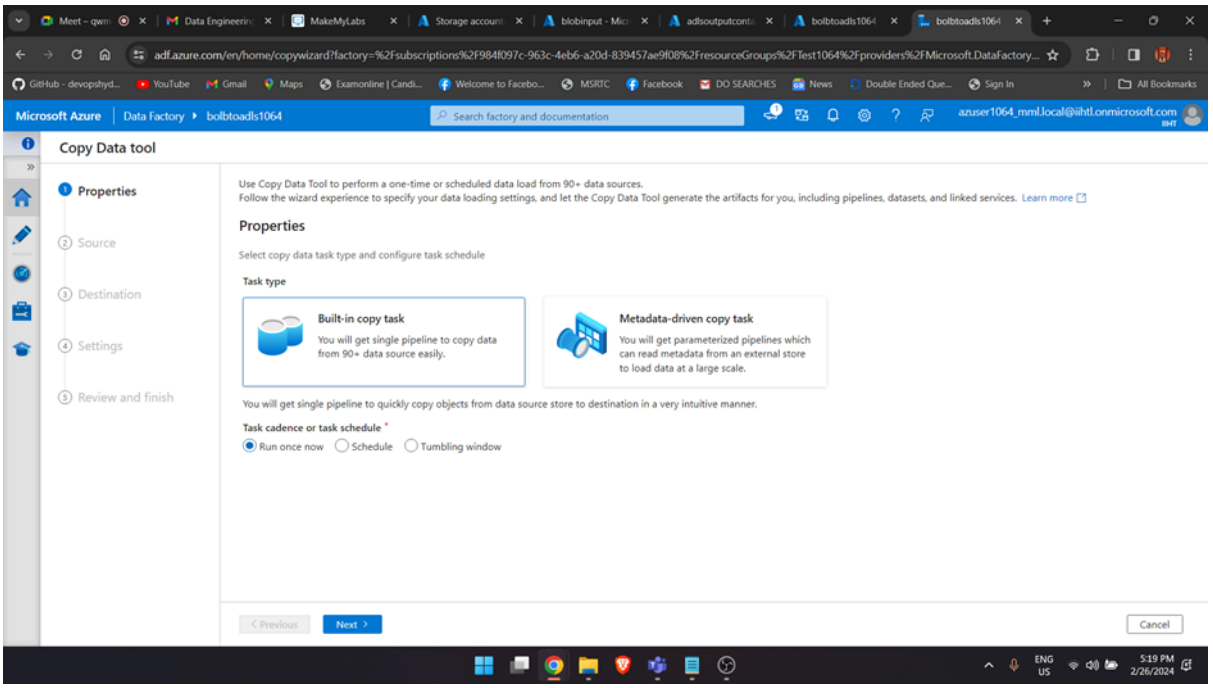
Created new Data Factory account for creating pipeline to copy data from input to output container:



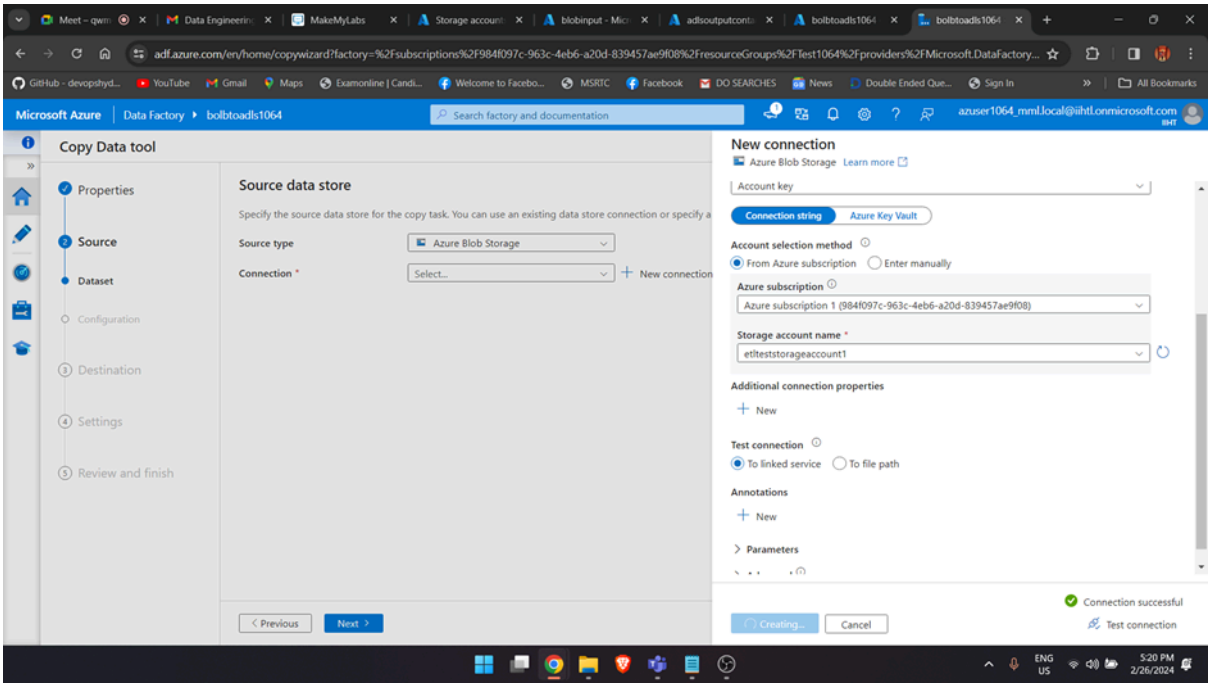


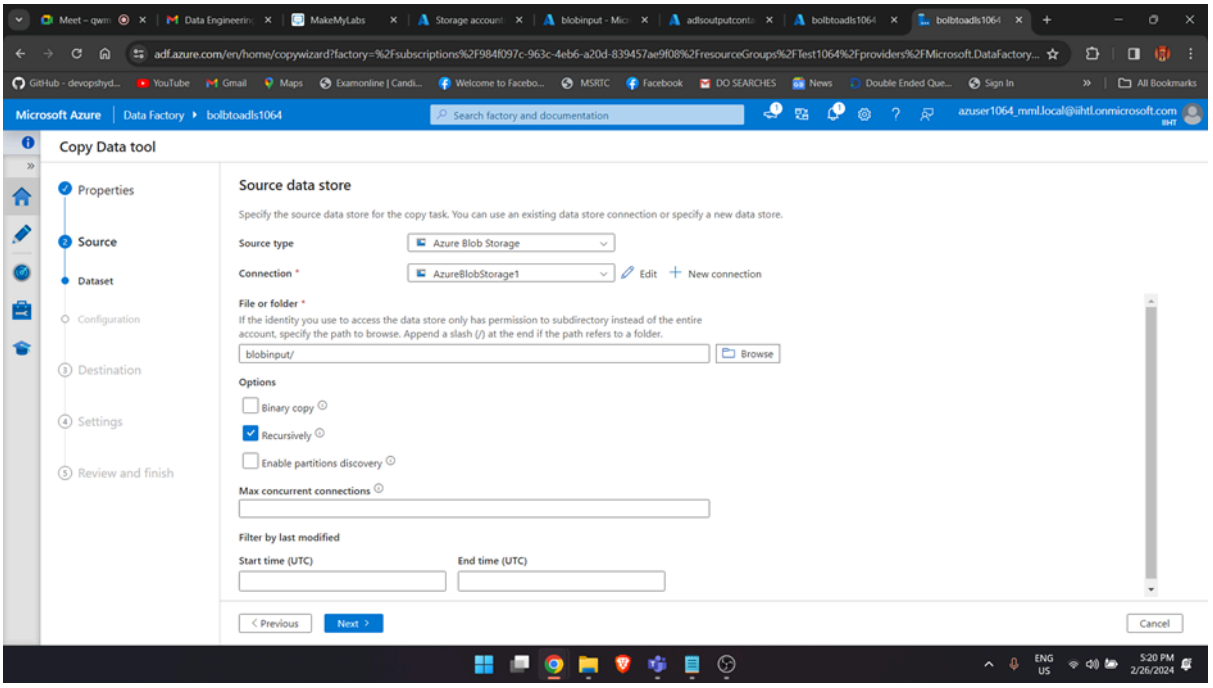
Using Ingest service for using built-in copy method :

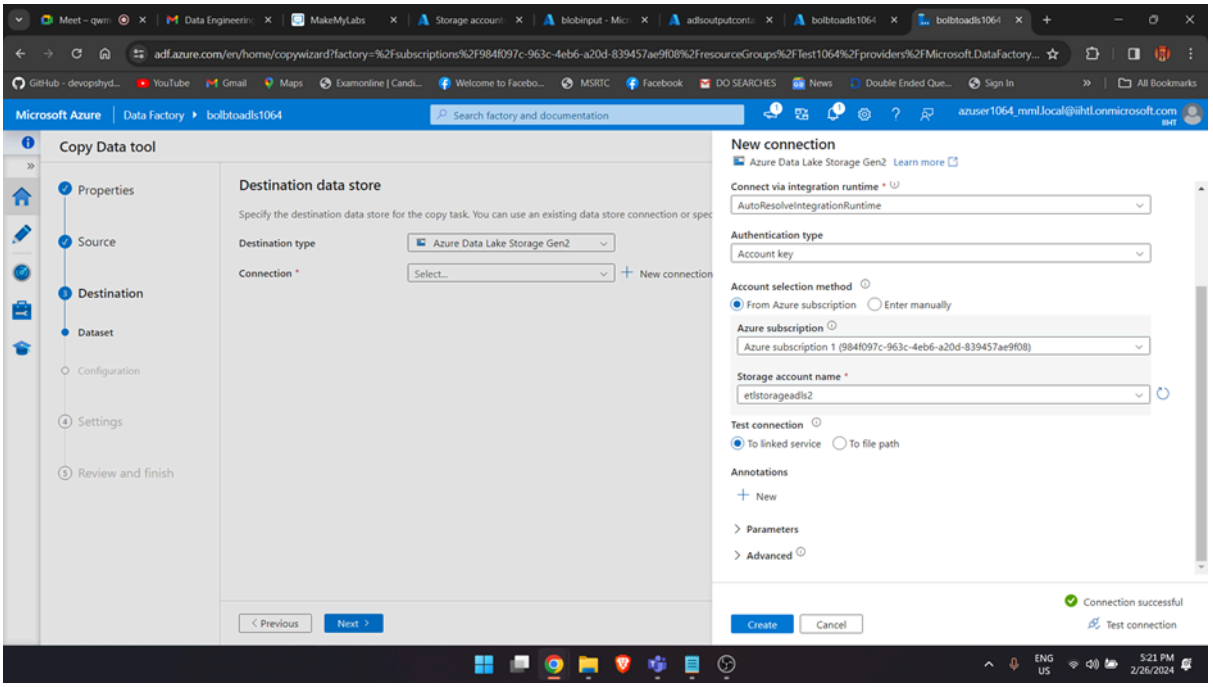


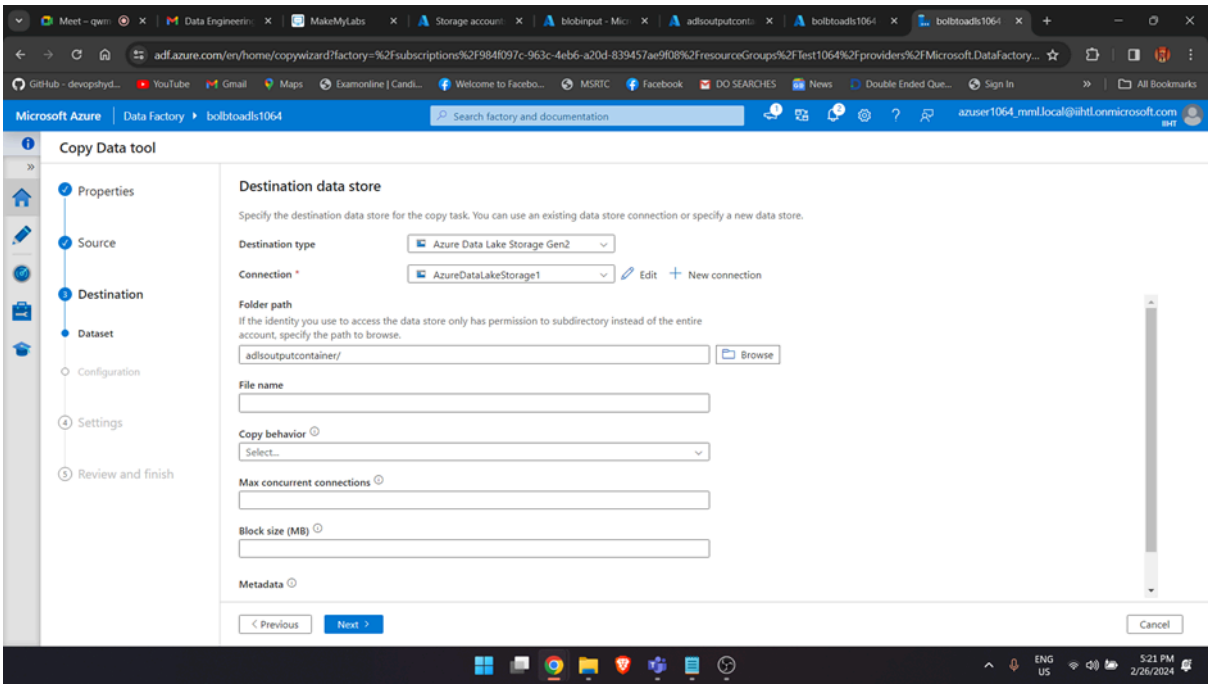


Giving input and output paths for copying data:

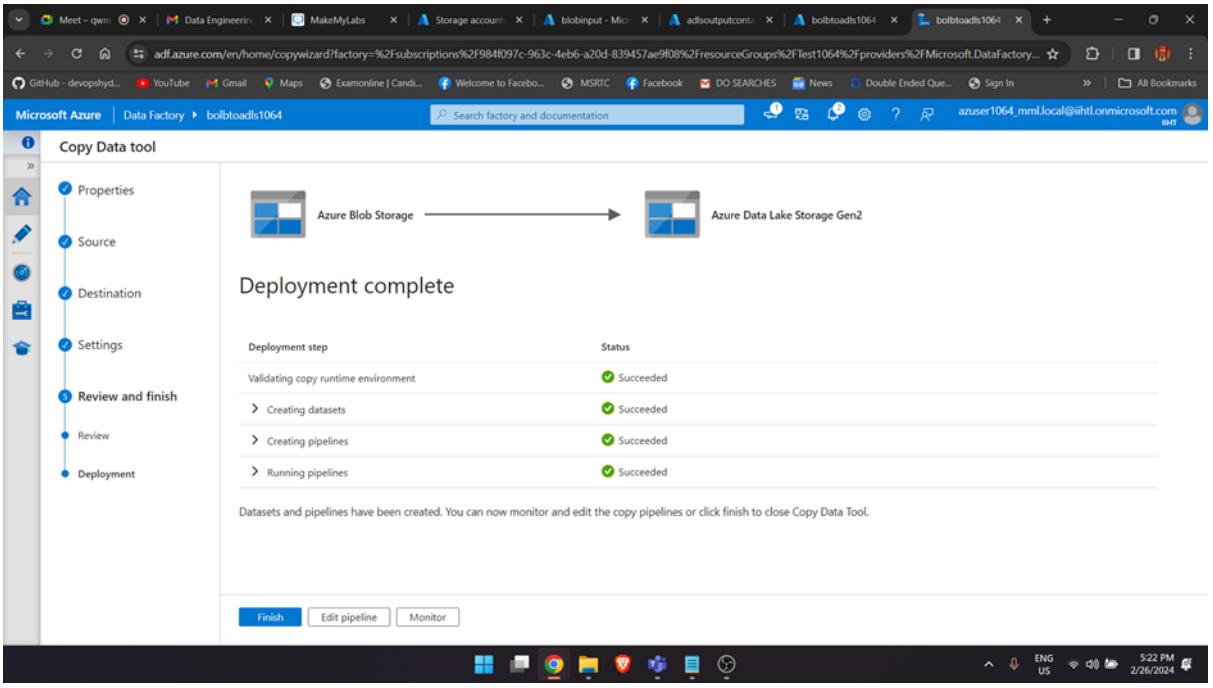


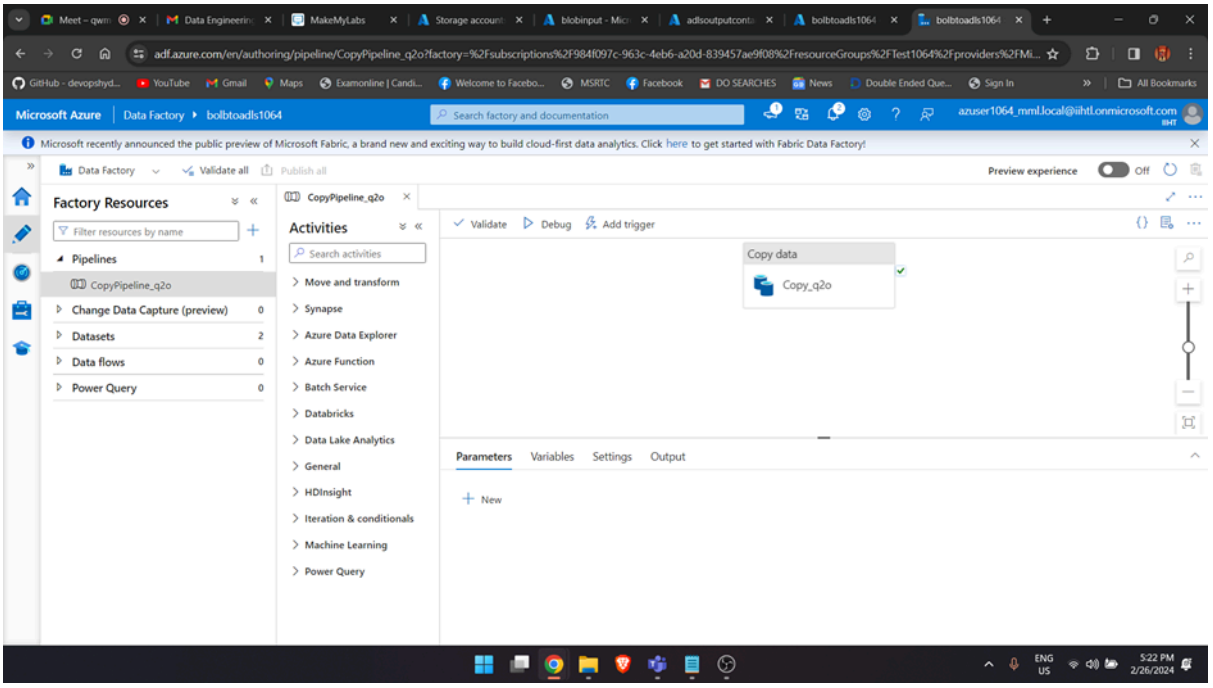



Microsoft Azure
Data Factory
bolbtoadls1064
Search factory and documentation
azuser1064_mml.local@ihtl.onmicrosoft.com
Copy Data tool
Properties
Source
Destination
Dataset
Configuration
Settings
Review and finish
Destination data store
Specify the destination data store for the copy task. You can use an existing data store connection or spec
Destination type
Azure Data Lake Storage Gen2
Connection *
Select...
New connection
Previous
Next
New connection
Azure Data Lake Storage Gen2
Learn more
Connect via integration runtime *
AutoResolveIntegrationRuntime
Authentication type
Account key
Account selection method
From Azure subscription
Enter manually
Azure subscription
Azure subscription 1 (984f097c-963c-4eb6-a20d-839457ae9f08)
Storage account name *
etlstorageadls2
Test connection
To linked service
To file path
Annotations
New
Parameters
Advanced
Connection successful
Test connection
Create
Cancel

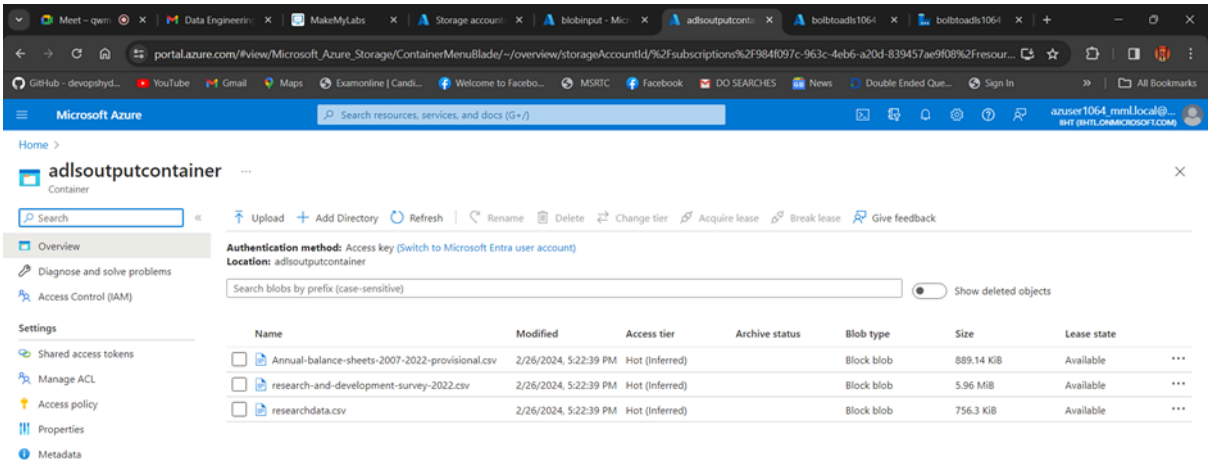

Microsoft Azure
Data Factory
bolbtoadls1064
Search factory and documentation
azuser1064_mml.local@ihtl.onmicrosoft.com
Copy Data tool
Properties
Source
Destination
Dataset
Configuration
Settings
Review and finish
Destination data store
Specify the destination data store for the copy task. You can use an existing data store connection or specify a new data store.
Destination type
Azure Data Lake Storage Gen2
Connection *
AzureDataLakeStorage1
Edit
New connection
Folder path
If the identity you use to access the data store only has permission to subdirectory instead of the entire account, specify the path to browse.
adlsoutputcontainer/
Browse
File name
Copy behavior
Select...
Max concurrent connections
Block size (MB)
Metadata
Previous
Next
Cancel

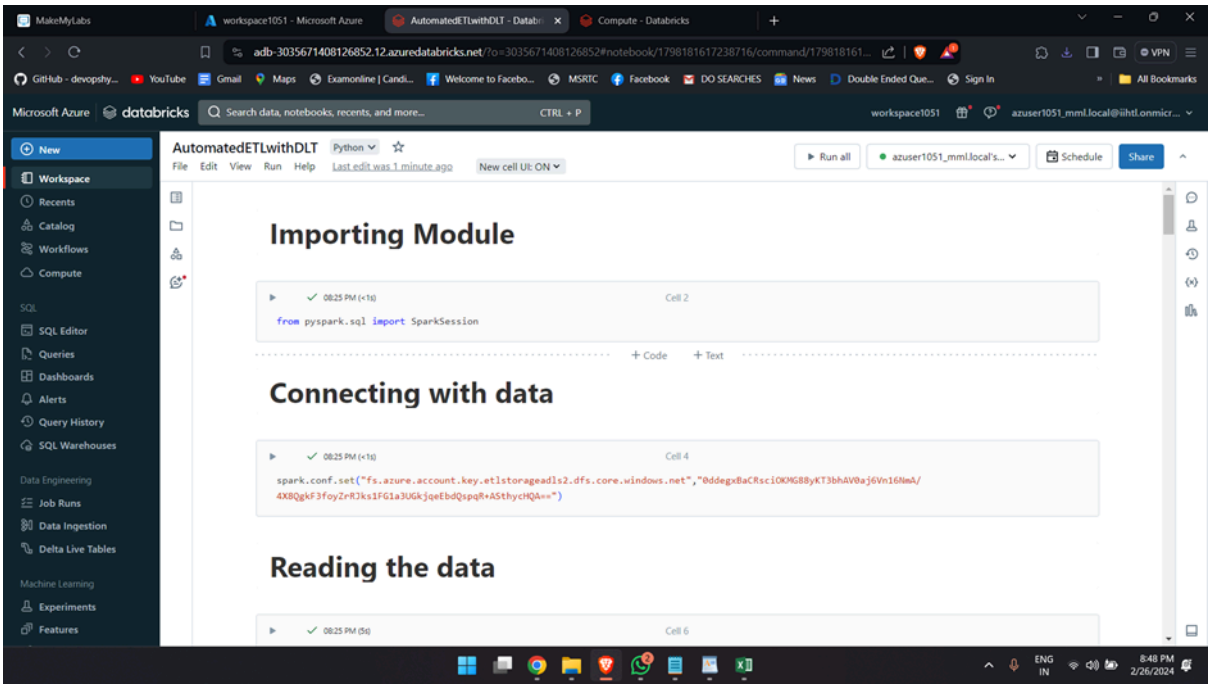
Creating pipeline :





Checking weather data is copied in output container or not :

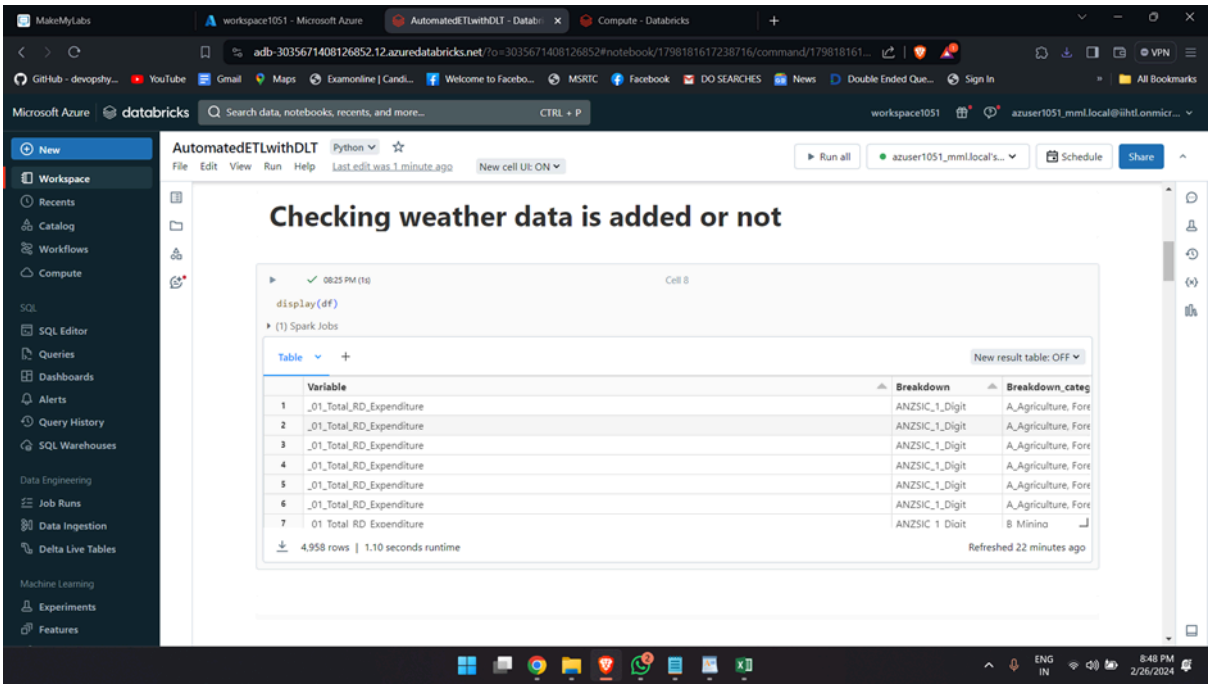




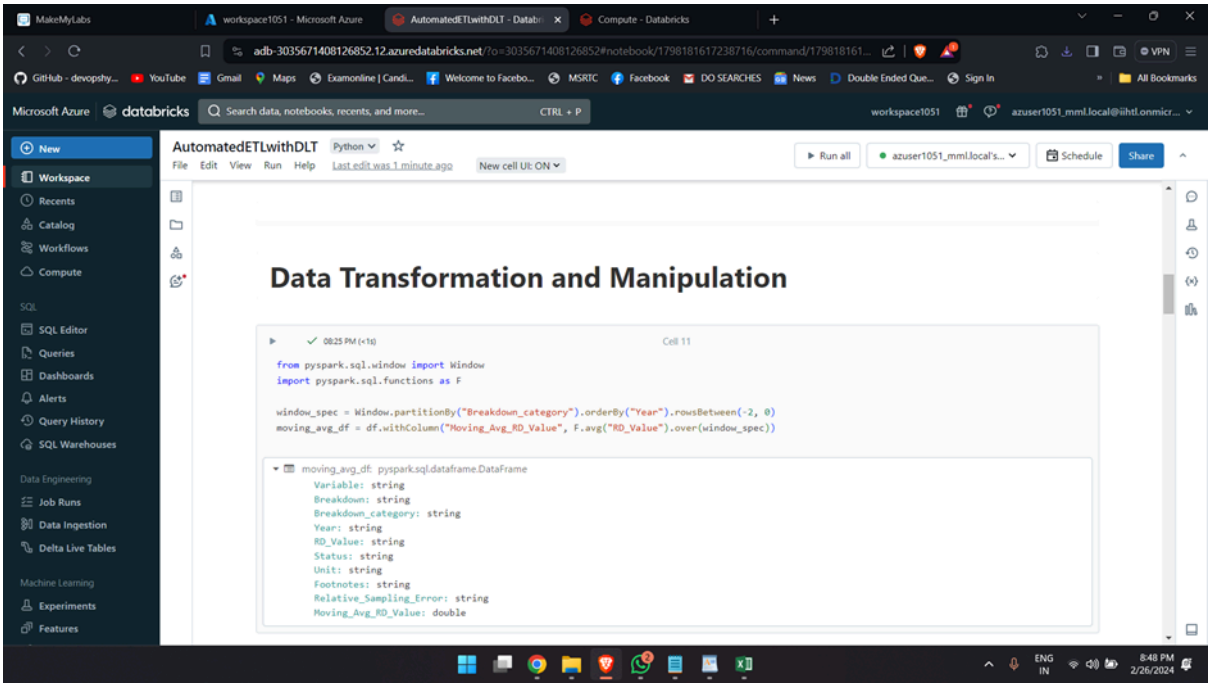
- **Import SparkSession:** This line imports the SparkSession class from the pyspark.sql module. SparkSession is the entry point to programming Spark with the Dataset and DataFrame API.

- **Set Azure Storage Key:** This line sets a configuration in the Spark session (spark.conf.set()) to provide authentication information for accessing Azure Data Lake Storage (ADLS). It sets the key fs.azure.account.key.etlstorageadls2.dfs.core.windows.net to the provided value, which is an Azure storage access key. This key is used to authenticate and authorize access to the Azure Data Lake Storage account named etlstorageadls2.

- **Read CSV File:** This line reads a CSV file named researchdata.csv from the Azure Data Lake Storage (ADLS) account etlstorageadls2. It uses the spark.read.csv() method to create a DataFrame (df). The header=True parameter indicates that the first line of the CSV file contains the column names.



· **Display DataFrame:** This line displays the DataFrame df using the display() function. In a Spark environment like Databricks, display() is often used to visualize the contents of a DataFrame.



Import Statements:

from pyspark.sql.window import Window: This imports the Window class from the pyspark.sql.window module. This class is used to define window specifications for window functions.

import pyspark.sql.functions as F: This imports the functions module from pyspark.sql and aliases it as F. This module provides various functions used in Spark SQL queries.

Define Window Specification:

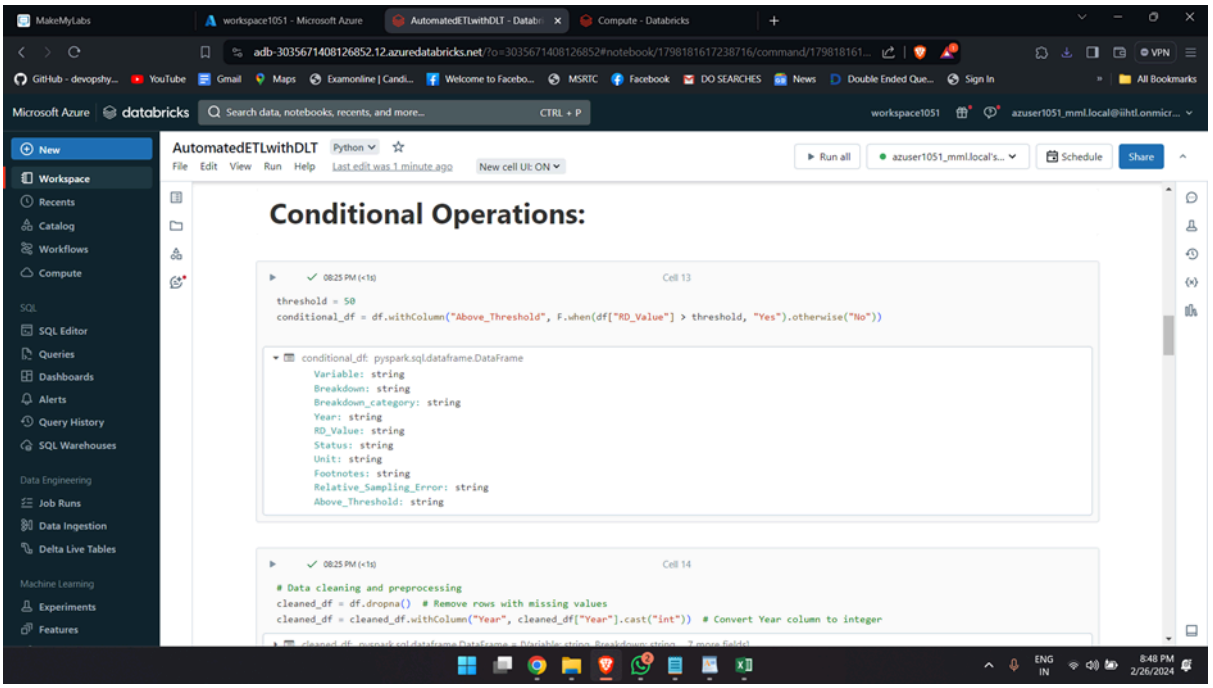
Window.partitionBy("Breakdown_category").orderBy("Year").rowsBetween(-2, 0): This line defines a window specification. It specifies how to partition the data (partitionBy("Breakdown_category")) and how to order the rows within each partition (orderBy("Year")). The rowsBetween(-2, 0)

part indicates that the window includes the current row and the two preceding rows based on the order defined.

Calculate Moving Average:

df.withColumn("Moving_Avg_RD_Value", F.avg("RD_Value").over(window_spec)): This line calculates the moving average of the "RD_Value" column within the window specified by window_spec. It creates a new column called "Moving_Avg_RD_Value" in the DataFrame df with the calculated moving averages.

In simpler terms, this code calculates a moving average of the "RD_Value" column in the DataFrame df, where the moving average is computed based on the values of the three most recent years within each category defined by the "Breakdown_category" column.



Threshold Definition:

threshold = 50: This line sets a threshold value of 50. This threshold will be used to determine whether the "RD_Value" is above or below this threshold.

Conditional Column Creation:

conditional_df = df.withColumn("Above_Threshold", F.when(df["RD_Value"] > threshold, "Yes").otherwise("No")): This line creates a new column called "Above_Threshold" in the DataFrame df. The value of this column is determined based on a condition: if the value in the "RD_Value" column is greater than the specified threshold (threshold), it assigns "Yes" to the "Above_Threshold" column; otherwise, it assigns "No". This is achieved using the when() function from pyspark.sql.functions, which allows conditional logic within DataFrame operations.

Removing Rows with Missing Values:

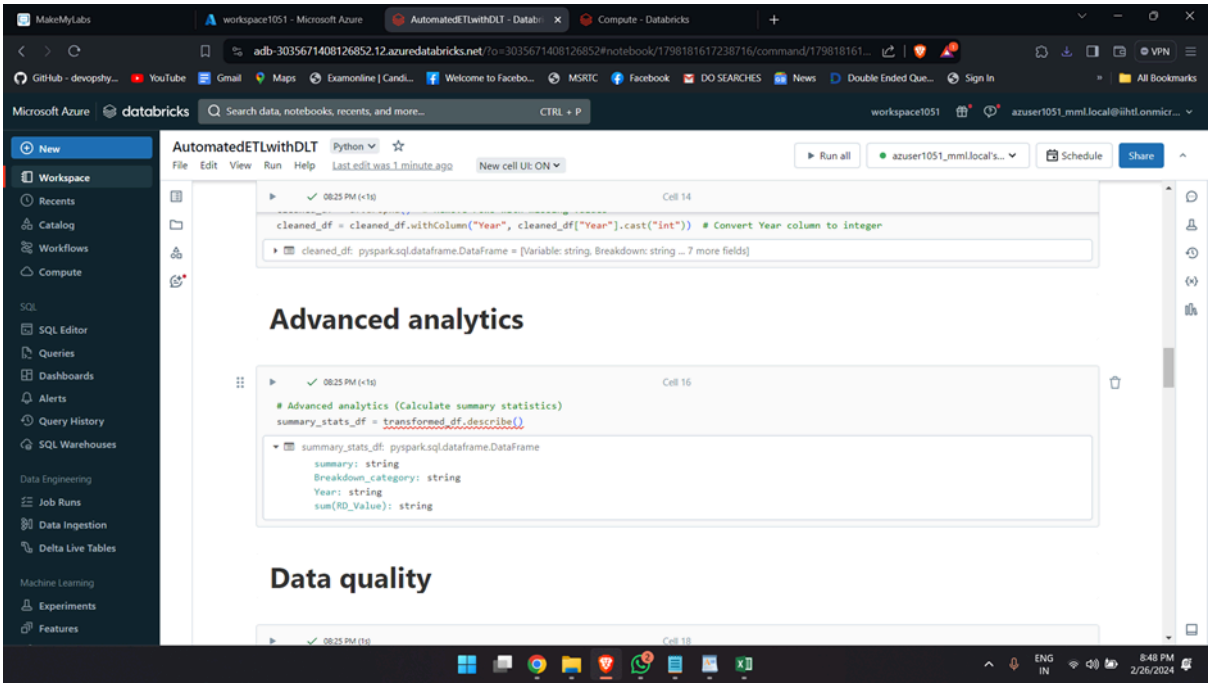
cleaned_df = df.dropna(): This line removes rows with any missing (null or NaN) values from the DataFrame df. It returns a new DataFrame cleaned_df with those rows removed.

Converting Column Data Type:

cleaned_df = cleaned_df.withColumn("Year", cleaned_df["Year"].cast("int")): This line converts the data type of the "Year" column in the DataFrame cleaned_df to integer (int). It uses the withColumn() function to replace the existing "Year" column with a new

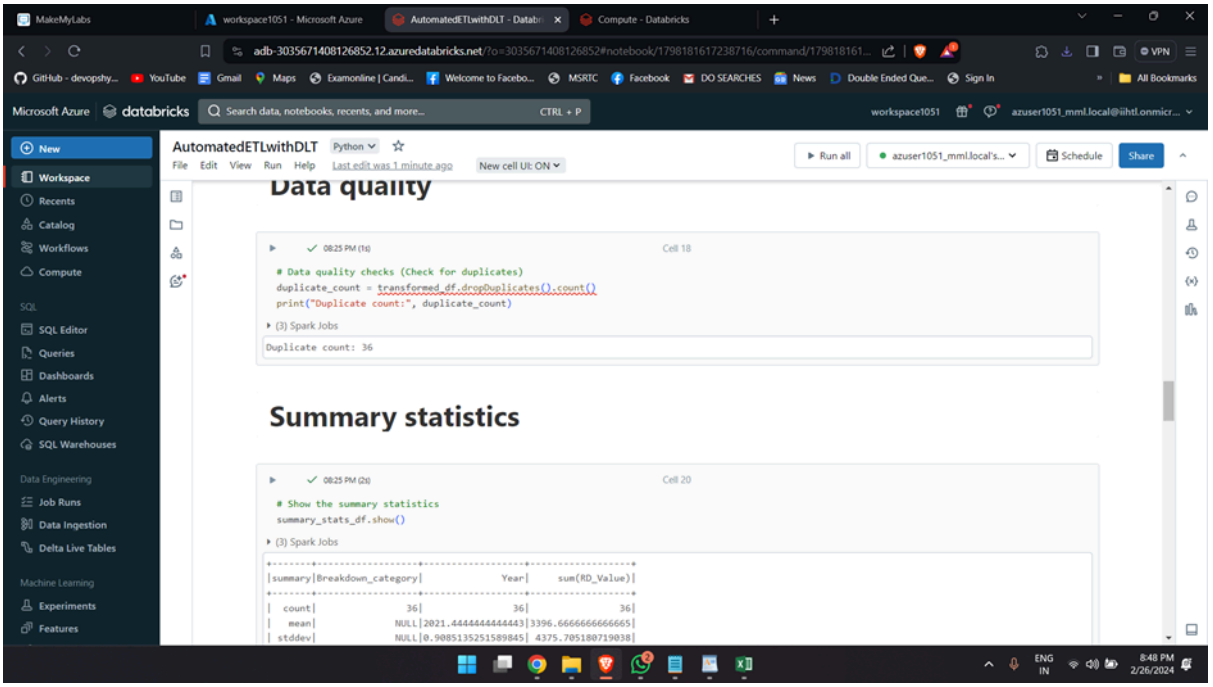
one where the data type has been explicitly cast to integer using the cast() function.

In summary, the code first creates a new column in the DataFrame based on a condition, and then it performs data cleaning operations by removing rows with missing values and converting the data type of a column.



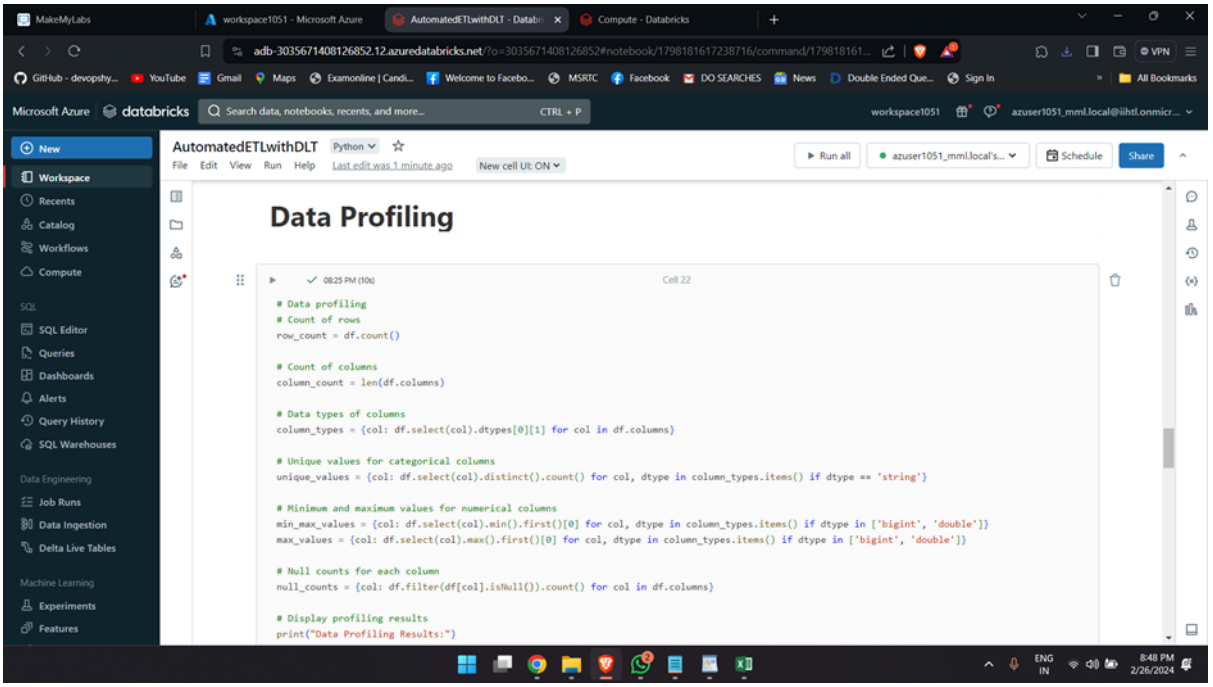
Calculating Summary Statistics for Data Analysis:

The code computes summary statistics for the DataFrame transformed_df using the describe() function. This function generates statistical summaries, including count, mean, standard deviation, minimum, maximum, and percentiles, for each numerical column in the DataFrame. The resulting summary statistics are stored in the DataFrame summary_stats_df



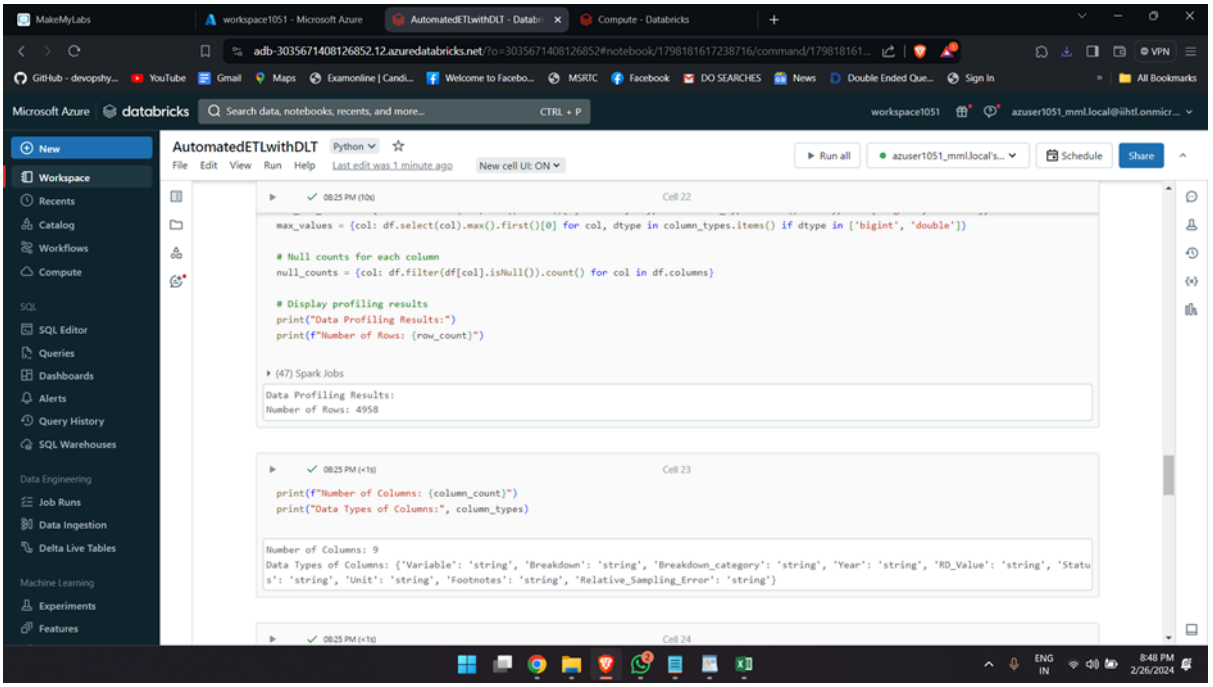
Data Quality Checks - Duplicate Detection and Summary Statistics

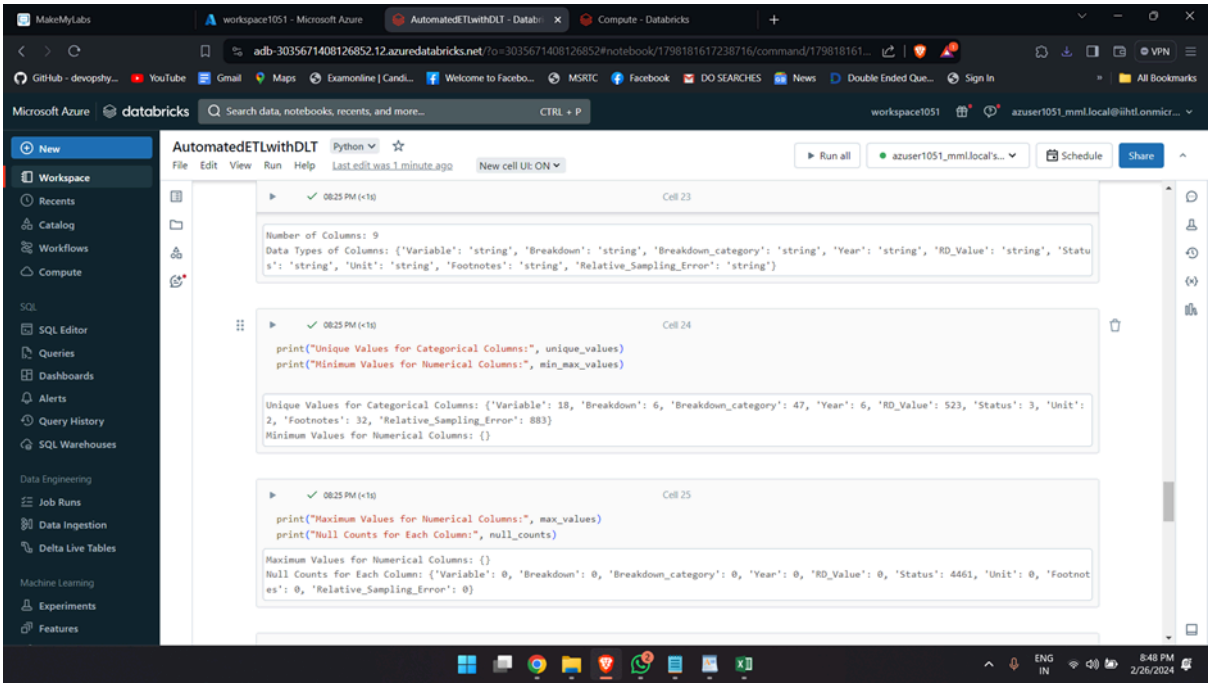
The code first checks for duplicates in the DataFrame transformed_df by dropping duplicate rows and counting the remaining rows. It then prints the count of duplicates found. Following this, it displays the summary statistics computed earlier for transformed_df using the describe() function.



Data Profiling:

The code performs data profiling on the DataFrame df to gain insights into its structure and content. It calculates the count of rows and columns, identifies the data types of columns, determines unique values for categorical columns, extracts minimum and maximum values for numerical columns, and computes null counts for each column. Finally, it displays the profiling results including the number of rows.

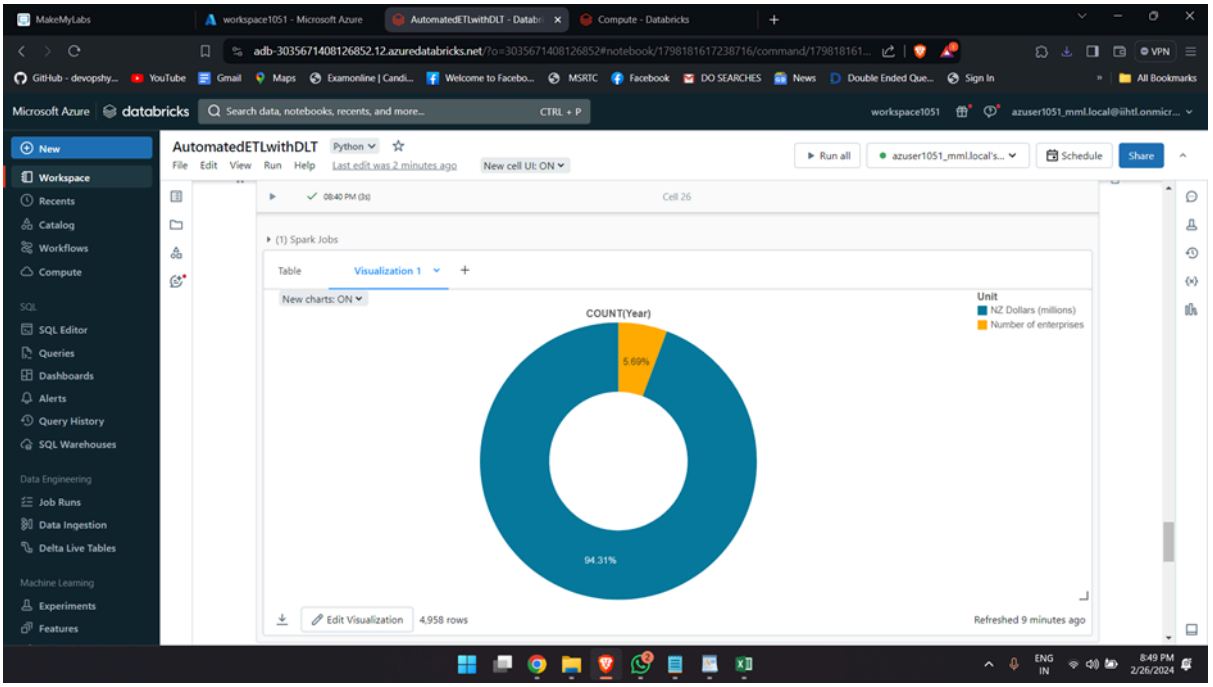

AutomatedETLwithDLT
Python
File Edit View Run Help Last edit was 1 minute ago New cell UI: ON
Run all
Schedule
Share
Cell 22
max_values = {col: df.select(col).max().first()[0] for col, dtype in column_types.items() if dtype in ['bigint', 'double']}
# Null counts for each column
null_counts = {col: df.filter(df[col].isNull()).count() for col in df.columns}
# Display profiling results
print("Data Profiling Results:")
print(f"Number of Rows: {row_count}")
(47) Spark Jobs
Data Profiling Results:
Number of Rows: 4958
Cell 23
print(f"Number of Columns: {column_count}")
print("Data Types of Columns:", column_types)
Number of Columns: 9
Data Types of Columns: {'Variable': 'string', 'Breakdown': 'string', 'Breakdown_category': 'string', 'Year': 'string', 'RD_Value': 'string', 'Status': 'string', 'Unit': 'string', 'Footnotes': 'string', 'Relative_Sampling_Error': 'string'}
Cell 24

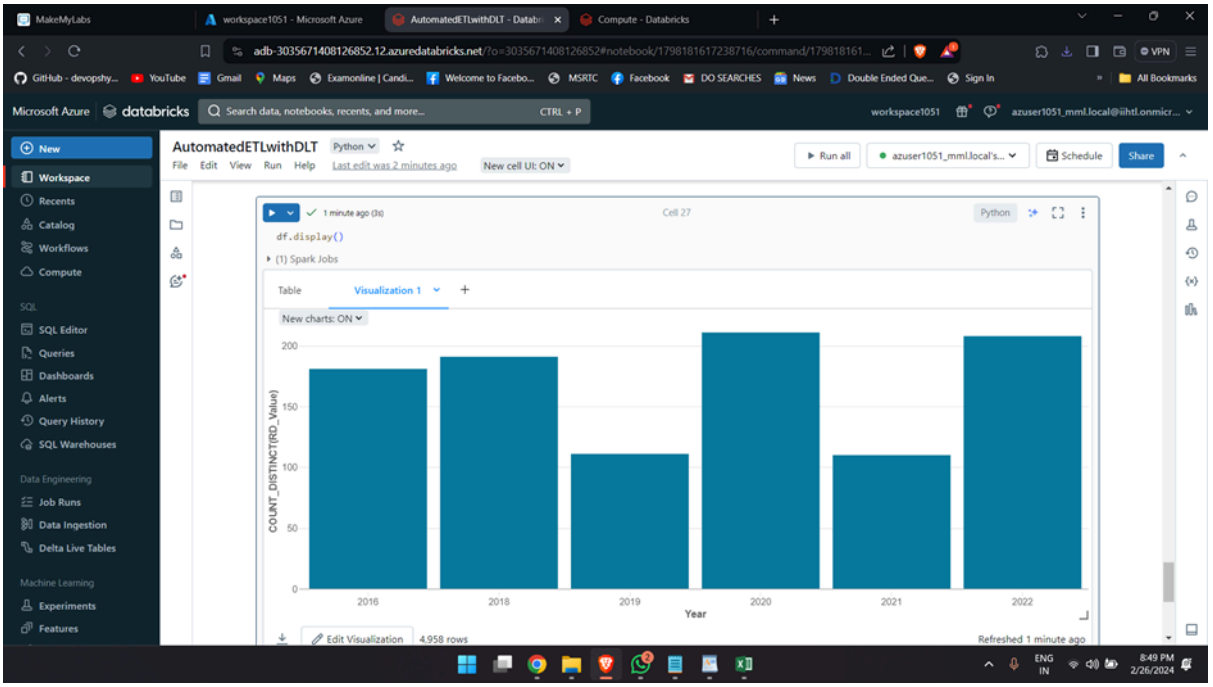

AutomatedETLwithDLT
Python
File Edit View Run Help Last edit was 1 minute ago New cell UI: ON
Run all
Schedule
Share
Cell 23
Number of Columns: 9
Data Types of Columns: {'Variable': 'string', 'Breakdown': 'string', 'Breakdown_category': 'string', 'Year': 'string', 'RD_Value': 'string', 'Status': 'string', 'Unit': 'string', 'Footnotes': 'string', 'Relative_Sampling_Error': 'string'}
Cell 24
print("Unique Values for Categorical Columns:", unique_values)
print("Minimum Values for Numerical Columns:", min_max_values)
Unique Values for Categorical Columns: {'Variable': 18, 'Breakdown': 6, 'Breakdown_category': 47, 'Year': 6, 'RD_Value': 523, 'Status': 3, 'Unit': 2, 'Footnotes': 32, 'Relative_Sampling_Error': 883}
Minimum Values for Numerical Columns: {}
Cell 25
print("Maximum Values for Numerical Columns:", max_values)
print("Null Counts for Each Column:", null_counts)
Maximum Values for Numerical Columns: {}
Null Counts for Each Column: {'Variable': 0, 'Breakdown': 0, 'Breakdown_category': 0, 'Year': 0, 'RD_Value': 0, 'Status': 4461, 'Unit': 0, 'Footnotes': 0, 'Relative_Sampling_Error': 0}

Data Profiling Results:

- **Number of Rows:** [row_count]
- **Number of Columns:** [column_count]
- **Data Types of Columns:** [column_types]
- **Unique Values for Categorical Columns:** [unique_values]
- **Minimum Values for Numerical Columns:** [min_max_values]
- **Maximum Values for Numerical Columns:** [max_values]
- **Null Counts for Each Column:** [null_counts]

The code prints the results of the data profiling analysis, including the number of rows and columns, data types of columns, unique values for categorical columns, minimum and maximum values for numerical columns, and null counts for each column.





Data visualisation :

Visualisation helps to represent data visually through charts, graphs, or maps, facilitating easier interpretation, analysis, and communication of patterns, trends, and insights within the data.

CI/CD Using DevOps :

Leveraging a DevOps pipeline for your project brings numerous benefits to the development and deployment processes. Here's a theoretical overview of the advantages and principles of using a DevOps pipeline:

1. Continuous Integration (CI):

- Principle: CI is the practice of frequently integrating code changes into a shared repository. Each integration is verified by automated build and test processes, allowing teams to detect errors early.

- Advantages:

- Reduced integration issues: Regular integration helps identify and resolve conflicts and issues early in the development cycle.

- Faster feedback: Developers receive immediate feedback on the impact of their changes, enabling faster iteration and improvement.

- Higher code quality: Automated tests ensure that code changes meet quality standards before being merged into the main branch.

2. Continuous Deployment (CD):

- **Principle:** CD extends CI by automatically deploying code changes to production or staging environments after successful builds and tests.

- Advantages:

- Faster time to market: Automated deployment reduces manual intervention and accelerates the delivery of features and updates to end-users.

- Reduced risk: Automated deployments are consistent and repeatable, minimizing the risk of human errors and deployment failures.

- Improved reliability: Continuous monitoring and validation of deployments help ensure that only stable and tested changes are deployed into production.

3. Infrastructure as Code (IaC):

- Principle: IaC is the practice of managing and provisioning infrastructure resources (e.g., servers, networks, databases) through code and automation.

- Advantages:

- Version control: Infrastructure configurations are stored as code in version control systems, enabling tracking, collaboration, and rollback to previous states.

- Consistency and repeatability: Infrastructure changes are automated and reproducible, leading to consistent environments across development, testing, and production.

- Scalability and agility: Automated provisioning and configuration allow for rapid scaling and adaptation of infrastructure to meet changing demands.

4. Collaboration and Communication:

- Principle: DevOps promotes collaboration and communication among development, operations, and other stakeholders throughout the software development lifecycle.

- Advantages:

- Breaking down silos: DevOps encourages cross-functional teams to work together, breaking down silos and fostering a culture of collaboration and shared responsibility.

- Shared understanding: Continuous feedback and visibility into the development and deployment processes promote shared understanding and alignment of goals and priorities.

- Continuous improvement: Collaborative practices enable teams to learn from each other, share best practices, and continuously improve processes and outcomes.

5. Automation:

- **Principle:** Automation is central to DevOps, enabling the efficient and reliable execution of repetitive tasks, such as building, testing, and deploying software.

- Advantages:

- Increased efficiency: Automation reduces manual effort and allows teams to focus on higher-value activities, such as innovation and problem-solving.

- Consistency and predictability: Automated processes ensure consistent and predictable outcomes, reducing variability and improving reliability.

- Faster feedback and iteration: Automated tests and deployments provide rapid feedback on code changes, enabling faster iteration and improvement.

By leveraging a DevOps pipeline, organizations can achieve greater efficiency, agility, and reliability in their software development and deployment processes. Embracing DevOps principles and practices fosters a culture of collaboration, automation, and continuous improvement, ultimately delivering better value to customers and stakeholders.

Azure Synapse :

Azure Synapse Analytics is a cloud-based analytics service provided by Microsoft Azure. It combines big data and data warehousing into a single service that enables enterprises to ingest, prepare, manage, and serve data for immediate BI and machine learning needs. Here are the basics of Azure Synapse:

1. Unified Analytics Platform:

- Azure Synapse Analytics integrates data warehousing, big data analytics, and data integration into a unified platform.

- It provides a single interface for data engineers, data scientists, and business analysts to collaborate on analytics projects.

2. Components:

- SQL Data Warehouse: This component offers distributed data warehousing capabilities, allowing you to store and analyze petabytes of data with high performance and scalability.

- Apache Spark: Azure Synapse includes Apache Spark for big data processing, enabling advanced analytics, machine learning, and data exploration on large datasets.

- Data Integration: Synapse provides tools for ingesting, preparing, and transforming data from various sources, including streaming data, relational databases, and cloud storage.

3. SQL-on-Demand:

- Azure Synapse offers a serverless SQL-on-Demand feature that allows you to query data directly from Azure Data Lake Storage without the need to load it into a data warehouse.

- This feature provides on-demand querying capabilities for ad-hoc analysis and exploration of data stored in Azure Data Lake Storage.

4. Integration with Power BI:

- Azure Synapse integrates seamlessly with Power BI, Microsoft's business intelligence and analytics platform.

- This integration enables users to visualize and analyze data stored in Synapse directly within Power BI, creating interactive reports and dashboards for data-driven insights.

5. Security and Compliance:

- Azure Synapse provides enterprise-grade security features to protect sensitive data and ensure regulatory compliance.

- It includes built-in security controls such as data encryption, role-based access control (RBAC), and audit logging to help you meet compliance requirements.

6. Scale and Performance:

- Azure Synapse is designed for high performance and scalability, allowing you to scale compute and storage resources independently based on your workload requirements.

- It leverages distributed processing and parallel query execution to deliver fast query performance on large datasets.

7. Workspaces and Collaboration:

- Azure Synapse workspaces provide a collaborative environment for data professionals to work on analytics projects.

- Workspaces include features for version control, notebook integration, and sharing of artifacts, enabling teams to collaborate effectively on data analytics tasks.

8. Pricing Model:

- Azure Synapse offers a flexible pricing model based on pay-as-you-go pricing for compute and storage resources.

- You only pay for the resources you use, with no upfront costs or long-term commitments.

Overall, Azure Synapse Analytics provides a comprehensive and integrated platform for modern data analytics and enables organizations to derive valuable insights from their data quickly and efficiently.

Conclusion:

Our project focuses on enhancing data quality through Azure Databricks and streamlining data movement with Azure Data Factory. By implementing robust quality checks and efficient data transfer mechanisms, we're poised to improve decision-making, reduce latency, and ensure scalability in our data infrastructure. This initiative underscores our commitment to fostering a resilient and agile data ecosystem, driving actionable insights and innovation throughout our organization.