End-to-End Loan Data Processing and Analytics Pipeline Using Azure Data Factory and Databricks

Table of Contents

Project Statement

Create an Azure Data Factory pipeline that triggers the execution of Azure Databricks notebooks.

Use Azure DevOps for version control and continuous deployment of the notebooks.

Project Overview

Customer loan management and credit risk analysis are critical for financial institutions. Loan datasets contain continuous updates on customer demographics, income, expenditure, and repayment history. The project showcases:

- Ingestion Layer (Azure Data Lake Storage – Bronze): Capturing raw loan records in CSV format from multiple sources.
- Processing Layer (Azure Databricks – Silver): Cleaning, standardizing, and transforming the raw data for consistency and quality.
- Storage Layer (Delta Lake – Gold): Aggregating and summarizing key metrics such as total loan amounts, average income/expenditure, overdue counts, and repayment behavior.
- Analytics (Power BI): Visualizing loan distributions, repayment performance, and customer insights through interactive dashboards.

Prerequisites

1. Python Knowledge: Familiarity with Python and PySpark for data processing.

2. Databricks Cluster: A running Azure Databricks cluster with Delta Lake enabled for batch processing.
3. Azure Subscription: Active Azure subscription to manage resources.
4. Azure Databricks Workspace: Set up a workspace to create and manage notebooks.
5. Databricks Cluster Setup: Configure a cluster to execute Spark jobs.
6. Libraries and Dependencies: Install required Python libraries (e.g., pyspark, databricks-cli) in Databricks.
7. Monitoring and Logging: Enable monitoring and logging within Databricks to track job execution.
8. Azure Data Lake Storage (ADLS Gen2): Storage account with bronze, silver, and gold folders mounted in Databricks.
9. Azure Data Factory (ADF): For orchestrating ETL pipelines from Bronze → Silver → Gold.
10. Azure DevOps: Repository for storing notebooks and pipeline YAML files, with a CI/CD pipeline configured to deploy notebooks to Databricks.
11. Power BI: Installed and configured for connecting to Delta tables for analytics.

<u>Azure Resources Used for this Project:</u>

- Azure Data Lake Storage Gen2 (ADLS)
- Azure Databricks Workspace
- Azure Databricks Cluster
- Azure Event Hub
- Azure Key Vault

- Azure Data Factory (ADF)
- Azure DevOps
- Azure Storage Account

Project Objectives:

- Ingest raw loan data from Azure Storage into a structured data pipeline.
- Implement a Bronze-Silver-Gold architecture using Delta Lake for data refinement.
- Clean, standardize, and transform loan data for downstream analytics.
- Aggregate key metrics such as total loan amounts, overdue trends, and customer statistics.
- Enable seamless integration with Power BI for reporting and visualization.
- Orchestrate the pipeline using Azure Data Factory for automated execution.
- Maintain version control and CI/CD for notebooks and pipelines via Azure DevOps.
- Ensure scalability, fault tolerance, and monitoring across the data pipeline.

Tools Used:

- **Azure Data Factory (Orchestrator):**
    - Orchestrates the end-to-end ETL/ELT pipeline for loan data.
    - Copy Activity: Transfers raw CSV files from the Bronze folder in Azure Storage to a staging location.

  - Databricks Notebook Activity: Triggers Databricks notebooks to transform raw data into Silver and Gold Delta tables.
- **Azure Databricks (Transformation Engine):**
  - Executes PySpark and Python notebooks for data cleaning, transformation, and aggregation.
  - Handles schema enforcement, type casting, and computation of aggregate metrics like total loan amounts, average income, and overdue trends.
- **Azure Storage (Blob / Data Lake Gen2):**
  - Stores raw (Bronze), cleaned (Silver), and aggregated (Gold) datasets.
  - Supports Delta Lake format for ACID-compliant transactions.
- **Azure DevOps (CI/CD & Version Control):**
  - Maintains version control for notebooks and pipeline definitions.
  - Enables automated deployment of notebooks and pipeline updates to Databricks via pipelines.
- **Power BI (Analytics & Reporting):**
  - Connects to Gold Delta tables to generate dashboards and visualize key loan metrics.

Execution Overview:

1. **Data Storage:**
   - Raw loan CSV files are stored in Azure Blob Storage or Azure Data Lake Storage Gen2 (Bronze layer).
   - Transformed Silver and Gold Delta tables, and Parquet outputs, are stored in the same storage accounts with Delta Lake format for ACID compliance.

2. **Orchestration with Azure Data Factory (ADF):**
    - Pipeline Creation: An ADF pipeline is defined with multiple stages for end-to-end ETL.
    - Copy Activity: Copies raw CSV files from the source container (Bronze) to a temporary staging location in Azure Storage.
    - Databricks Notebook Activity: Triggers Databricks notebooks to perform transformations and generate Silver/Gold datasets.
3. **Data Transformation in Azure Databricks:**
    - Notebook Execution: PySpark notebooks process the data to:
        - Clean and enforce schema.
        - Convert CSV files into optimized Parquet format.
        - Compute aggregates like total loan amount, average income, and overdue metrics.
        - Partition and compress the data for better query performance.
        - Write results to Silver and Gold Delta tables in ADLS.
4. **Scheduling and Monitoring:**
    - Pipeline Scheduling: ADF triggers the pipeline at regular intervals for automated ingestion and transformation.
    - Performance Monitoring: ADF and Databricks monitoring tools track pipeline execution, cluster utilization, and job performance.
    - Analysis and Optimization: Execution logs and metrics are analyzed to detect bottlenecks, improve throughput, and optimize storage access.

Implementation – Tasks Performed

1. **Create Azure Storage Account and Containers**
    - Provision a Storage Account for raw and processed data.
    - Create containers: source folder for CSV files and destination folder for converted Delta files.
    - Upload raw CSV files into the source container.
2. **Define Data Sources and Locations**
    - Identify the location of raw loan CSV files in Azure Blob Storage or Azure Data Lake Storage Gen2 (Bronze layer).
    - Choose the destination for transformed Silver and Gold Delta tables in ADLS.
3. **Mount Azure Storage in Databricks**
    - Mount ADLS Gen2 or Blob Storage containers to Databricks for easy access.

4. **Develop Databricks Notebooks**

    - Bronze Layer – Raw Ingestion: Read CSV files using Spark DataFrames, write raw data to Delta.

    - Silver Layer – Cleaned Data: Apply transformations like schema enforcement, missing value handling, trimming, and data type casting.

    - Gold Layer – Aggregations: Compute metrics such as total loan amount, average income, overdue counts, and other KPIs.

    - Write results: Save to ADLS in appropriate Bronze, Silver, and Gold folders.

5. **Set Up Azure Data Factory (ADF)**

    - Create an ADF pipeline with two main activities:

        - Copy Activity: Copies CSV files from the source container to a temporary staging location in Azure Storage.

        - Databricks Notebook Activity: Triggers a Databricks notebook that handles data transformation, cleaning, and conversion to Delta format.

6. **ADF Implementation Steps**

    - Linked Services:

        - Azure Data Lake Storage Gen2 (for input/output folders)
        - Azure Databricks (for notebook execution)
    - Pipeline Activities:
        - Databricks Notebook Activity – Bronze: Load raw loan data → Delta Bronze
        - Databricks Notebook Activity – Silver: Transform, clean, and write → Delta Silver
        - Databricks Notebook Activity – Gold: Aggregate metrics → Delta Gold
    - Dependencies/Chaining: Use success dependency so each notebook runs after the previous finishes successfully.
    - Triggers: Schedule trigger (nightly batch) or event-based trigger (when new files arrive in ADLS).

7. **Source Control & Versioning (Azure DevOps)**
    - Git Repository: Store all Databricks notebooks, ADF pipeline JSON definitions, and configuration files.
    - Branching Strategy: main for production-ready code, dev for development/testing.

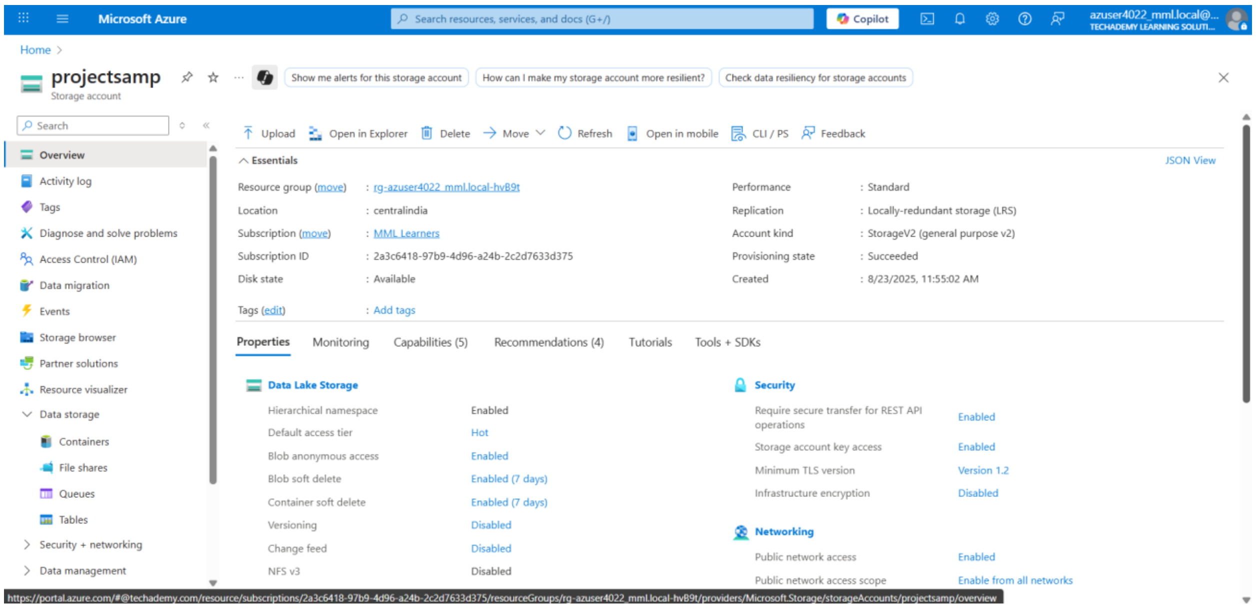
8. **Visualization with Power BI**

- Connect Power BI to Gold Delta tables in ADLS via Azure Synapse Analytics or Databricks SQL endpoint.
- Create dashboards for metrics such as: total loans per category, average income/expenditure, overdue trends, returned cheque counts.
- Schedule refreshes to show near real-time analytics from Gold tables.

<u>Practical Implementation on Azure Portal</u>

Step 1: Create Azure Storage Account

- Provision a Storage Account for raw and processed data.



Essentials

Resource group (move)	: rg-azuser4022_mml.local-hvB9t
Location	: centralindia
Subscription (move)	: MML Learners
Subscription ID	: 2a3c6418-97b9-4d96-a24b-2c2d7633d375
Disk state	: Available
Tags (edit)	: Add tags
Performance	: Standard
Replication	: Locally-redundant storage (LRS)
Account kind	: StorageV2 (general purpose v2)
Provisioning state	: Succeeded
Created	: 8/23/2025, 11:55:02 AM

Data Lake Storage

Hierarchical namespace	Enabled
Default access tier	Hot
Blob anonymous access	Enabled
Blob soft delete	Enabled (7 days)
Container soft delete	Enabled (7 days)
Versioning	Disabled
Change feed	Disabled
NFS v3	Disabled

Security

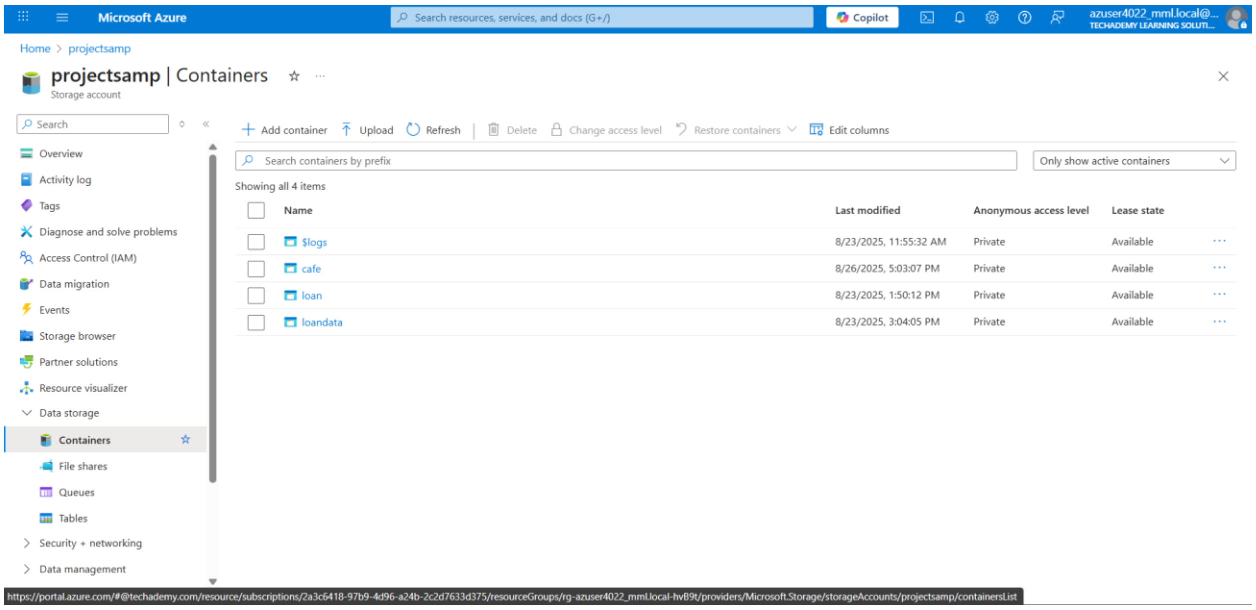
Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled

Networking

Public network access	Enabled
Public network access scope	Enable from all networks

https://portal.azure.com/#@techademy.com/resource/subscriptions/2a3c6418-97b9-4d96-a24b-2c2d7633d375/resourceGroups/rg-azuser4022_mml.local-hvB9t/providers/Microsoft.Storage/storageAccounts/projectsamp/overview

- Create two containers: sourcecontainer for CSV files, destinationparquetcontainer for converted Delta files.

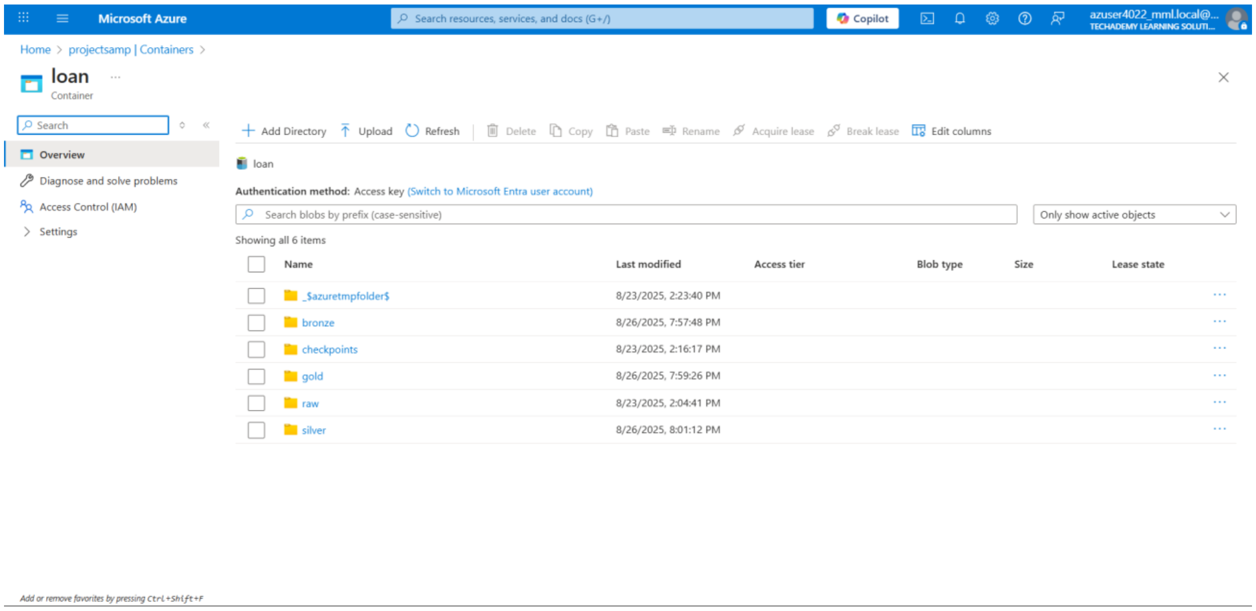


Showing all 4 items

Name	Last modified	Anonymous access level	Lease state
$logs	8/23/2025, 11:55:32 AM	Private	Available
cafe	8/26/2025, 5:03:07 PM	Private	Available
loan	8/23/2025, 1:50:12 PM	Private	Available
loandata	8/23/2025, 3:04:05 PM	Private	Available

https://portal.azure.com/#@techademy.com/resource/subscriptions/2a3c6418-97b9-4d96-a24b-2c2d7633d375/resourceGroups/rg-azuser4022_mml.local-hvB9t/providers/Microsoft.Storage/storageAccounts/projectsamp/containersList

- Upload CSV files into the source container.



Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive) | Only show active objects

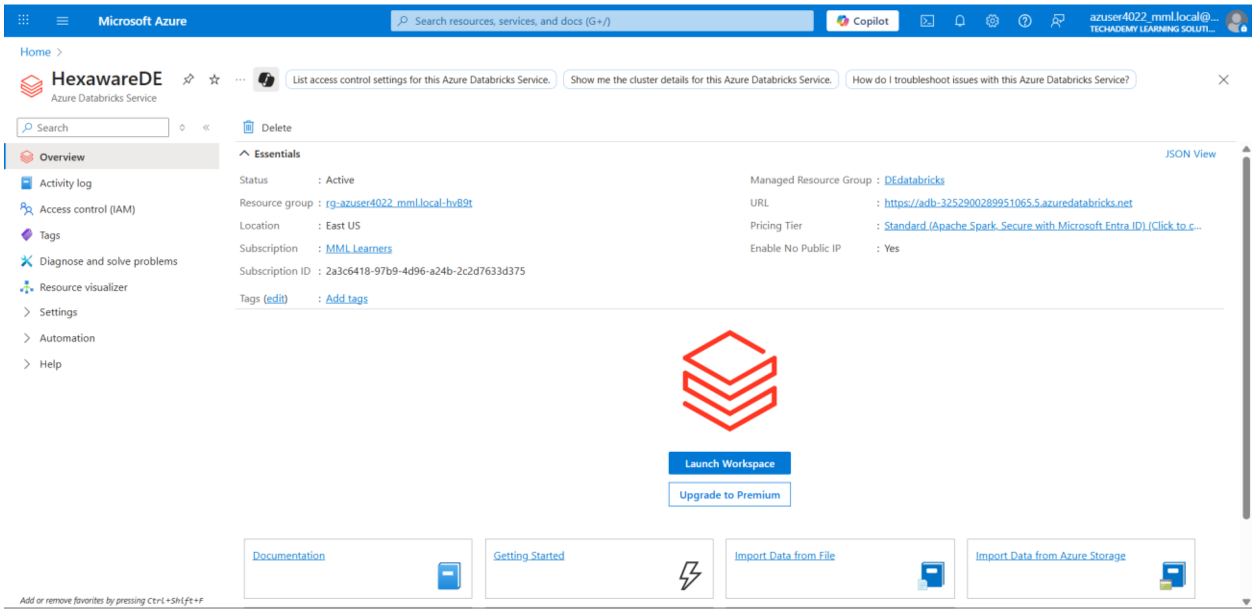
Showing all 6 items

Name	Last modified	Access tier	Blob type	Size	Lease state
_$azuretmpfolder$	8/23/2025, 2:23:40 PM				
bronze	8/26/2025, 7:57:48 PM				
checkpoints	8/23/2025, 2:16:17 PM				
gold	8/26/2025, 7:59:26 PM				
raw	8/23/2025, 2:04:41 PM				
silver	8/26/2025, 8:01:12 PM				

Add or remove favorites by pressing Ctrl+Shift+F

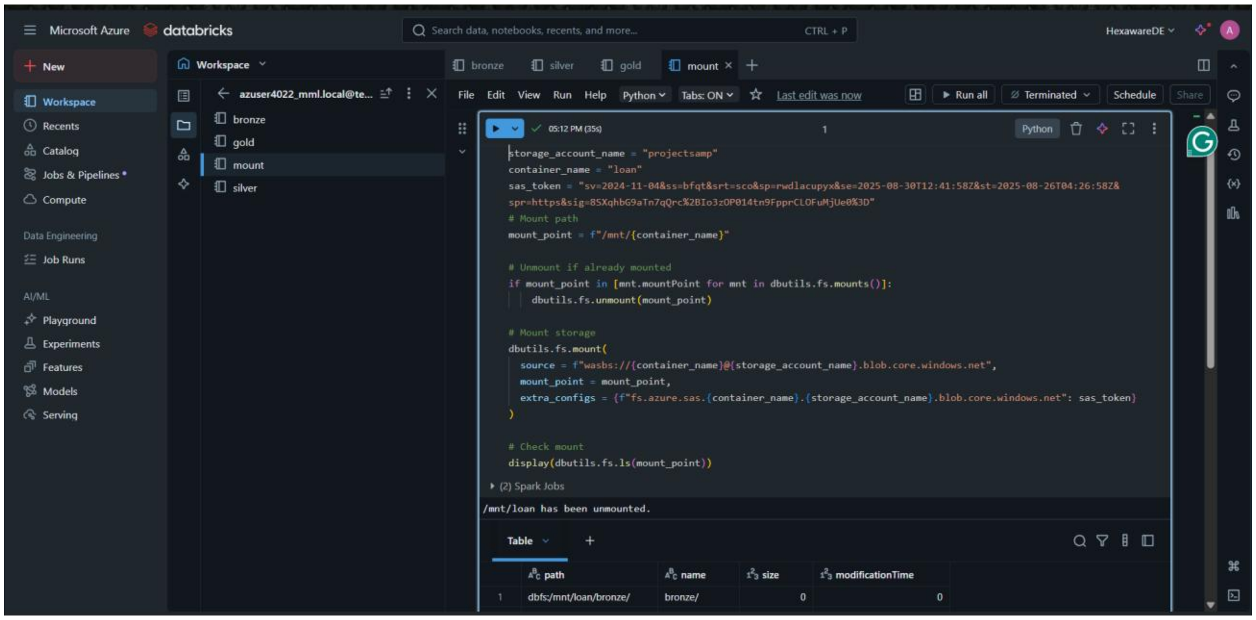
Step 2: Set Up Azure Databricks

- Create a Databricks workspace and cluster in the Azure Portal.


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
azuser4022_mml.local@...
TECHADEMY LEARNING SOLUTI...
Home
HexawareDE
Azure Databricks Service
List access control settings for this Azure Databricks Service.
Show me the cluster details for this Azure Databricks Service.
How do I troubleshoot issues with this Azure Databricks Service?
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Automation
Help
Delete
Essentials
JSON View
Launch Workspace
Upgrade to Premium
Documentation
Getting Started
Import Data from File
Import Data from Azure Storage
Add or remove favorites by pressing Ctrl+Shift+F

Status	: Active	Managed Resource Group	: DEdatabricks
Resource group	: rg-azuser4022_mml.local-hvB9t	URL	: https://adb-3252900289951065.5.azuredatabricks.net
Location	: East US	Pricing Tier	: Standard (Apache Spark, Secure with Microsoft Entra ID) (Click to c...
Subscription	: MML Learners	Enable No Public IP	: Yes
Subscription ID	: 2a3c6418-97b9-4d96-a24b-2c2d7633d375		
Tags (edit)	: Add tags		

- Create a new Notebook to mount storage accounts and process files.

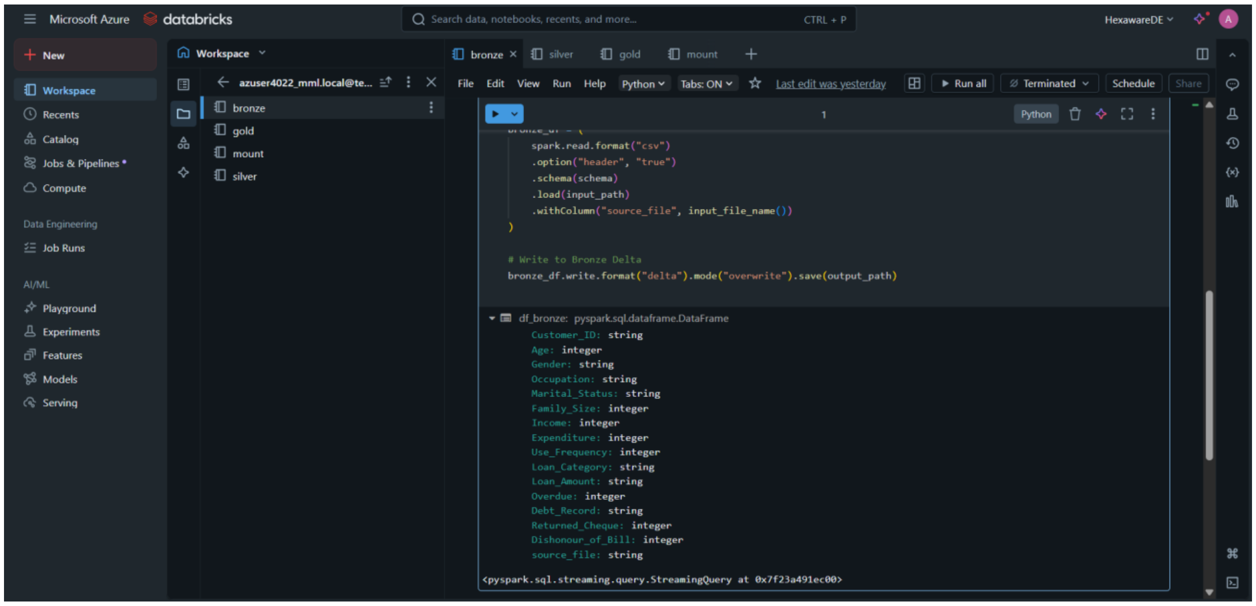


```
storage_account_name = "projectsamp"
container_name = "loan"
sas_token = "sv=2024-11-04&ss=bfqt&srt=sco&sp=rwdlacupyx&se=2025-08-30T12:41:58Z&st=2025-08-26T04:26:58Z&
spr=https&sig=85XqhbG9aTn7qQrc%2BIo3zOP014tn9FpprCLOFuMjUe0%3D"
# Mount path
mount_point = f"/mnt/{container_name}"

# Unmount if already mounted
if mount_point in [mnt.mountPoint for mnt in dbutils.fs.mounts()]:
    dbutils.fs.unmount(mount_point)

# Mount storage
dbutils.fs.mount(
  source = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
  mount_point = mount_point,
  extra_configs = {f"fs.azure.sas.{container_name}.{storage_account_name}.blob.core.windows.net": sas_token}
)

# Check mount
display(dbutils.fs.ls(mount_point))
```

```
/mnt/loan has been unmounted.
```

	path	name	size	modificationTime
1	dbfs:/mnt/loan/bronze/	bronze/	0	0

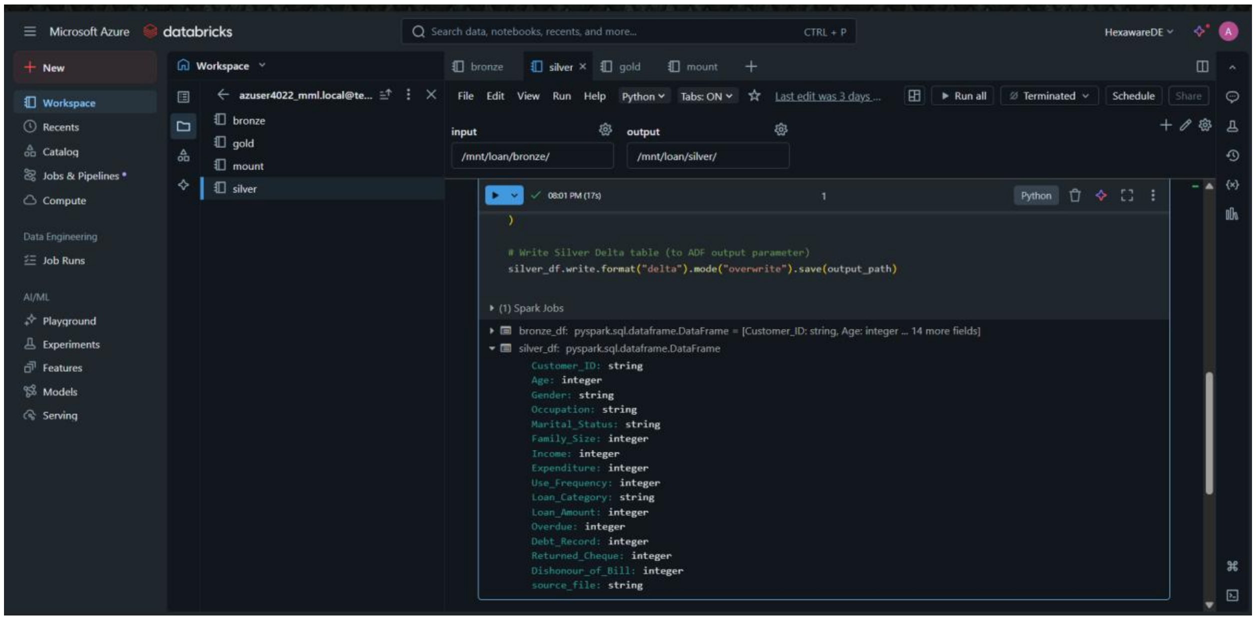
Create Bronze Notebook to copy the raw data



```
    spark.read.format("csv")
    .option("header", "true")
    .schema(schema)
    .load(input_path)
    .withColumn("source_file", input_file_name())
)

# Write to Bronze Delta
bronze_df.write.format("delta").mode("overwrite").save(output_path)
```

```
df_bronze:  pyspark.sql.dataframe.DataFrame
    Customer_ID: string
    Age: integer
    Gender: string
    Occupation: string
    Marital_Status: string
    Family_Size: integer
    Income: integer
    Expenditure: integer
    Use_Frequency: integer
    Loan_Category: string
    Loan_Amount: string
    Overdue: integer
    Debt_Record: string
    Returned_Cheque: integer
    Dishonour_of_Bill: integer
    source_file: string

<pyspark.sql.streaming.query.StreamingQuery at 0x7f23a491ec00>
```

Create silver notebook to clean the data



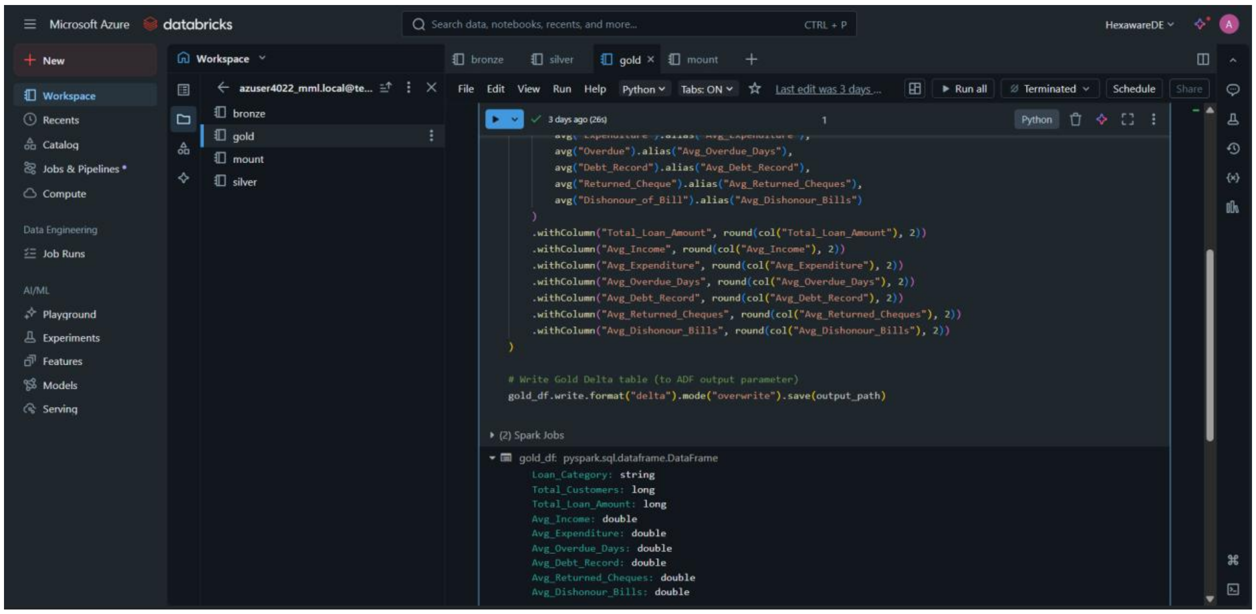
input: /mnt/loan/bronze/

output: /mnt/loan/silver/

```
)

# Write Silver Delta table (to ADF output parameter)
silver_df.write.format("delta").mode("overwrite").save(output_path)
```

```
(1) Spark Jobs
bronze_df:  pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 14 more fields]
silver_df:  pyspark.sql.dataframe.DataFrame
    Customer_ID: string
    Age: integer
    Gender: string
    Occupation: string
    Marital_Status: string
    Family_Size: integer
    Income: integer
    Expenditure: integer
    Use_Frequency: integer
    Loan_Category: string
    Loan_Amount: integer
    Overdue: integer
    Debt_Record: integer
    Returned_Cheque: integer
    Dishonour_of_Bill: integer
    source_file: string
```

Create gold notebook for aggregation function

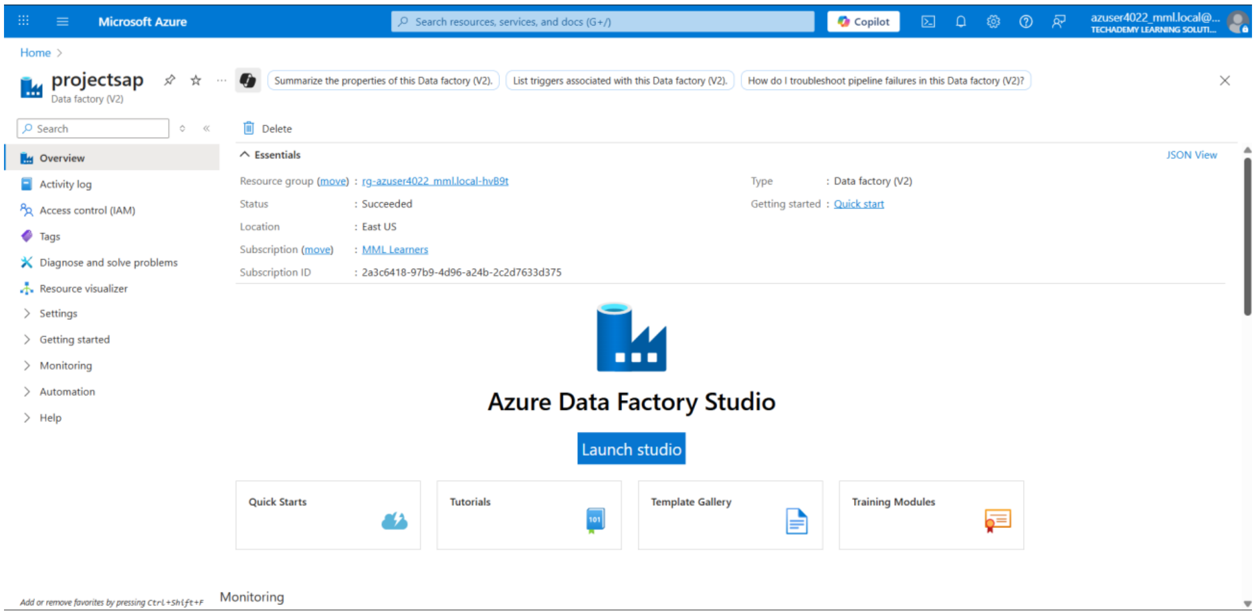


```
        avg("Expenditure").alias("Avg_Expenditure"),
        avg("Overdue").alias("Avg_Overdue_Days"),
        avg("Debt_Record").alias("Avg_Debt_Record"),
        avg("Returned_Cheque").alias("Avg_Returned_Cheques"),
        avg("Dishonour_of_Bill").alias("Avg_Dishonour_Bills")
    )
    .withColumn("Total_Loan_Amount", round(col("Total_Loan_Amount"), 2))
    .withColumn("Avg_Income", round(col("Avg_Income"), 2))
    .withColumn("Avg_Expenditure", round(col("Avg_Expenditure"), 2))
    .withColumn("Avg_Overdue_Days", round(col("Avg_Overdue_Days"), 2))
    .withColumn("Avg_Debt_Record", round(col("Avg_Debt_Record"), 2))
    .withColumn("Avg_Returned_Cheques", round(col("Avg_Returned_Cheques"), 2))
    .withColumn("Avg_Dishonour_Bills", round(col("Avg_Dishonour_Bills"), 2))
)

# Write Gold Delta table (to ADF output parameter)
gold_df.write.format("delta").mode("overwrite").save(output_path)
```

```
(2) Spark Jobs
gold_df: pyspark.sql.dataframe.DataFrame
    Loan_Category: string
    Total_Customers: long
    Total_Loan_Amount: long
    Avg_Income: double
    Avg_Expenditure: double
    Avg_Overdue_Days: double
    Avg_Debt_Record: double
    Avg_Returned_Cheques: double
    Avg_Dishonour_Bills: double
```

Step 3: Create a Data Factory

- Launch Azure Data Factory Studio.



projectsap
Data factory (V2)

Essentials

Resource group (move)	: rg-azuser4022_mml.local-hvB9t	Type	: Data factory (V2)
Status	: Succeeded	Getting started	: Quick start
Location	: East US		
Subscription (move)	: MML Learners		
Subscription ID	: 2a3c6418-97b9-4d96-a24b-2c2d7633d375		

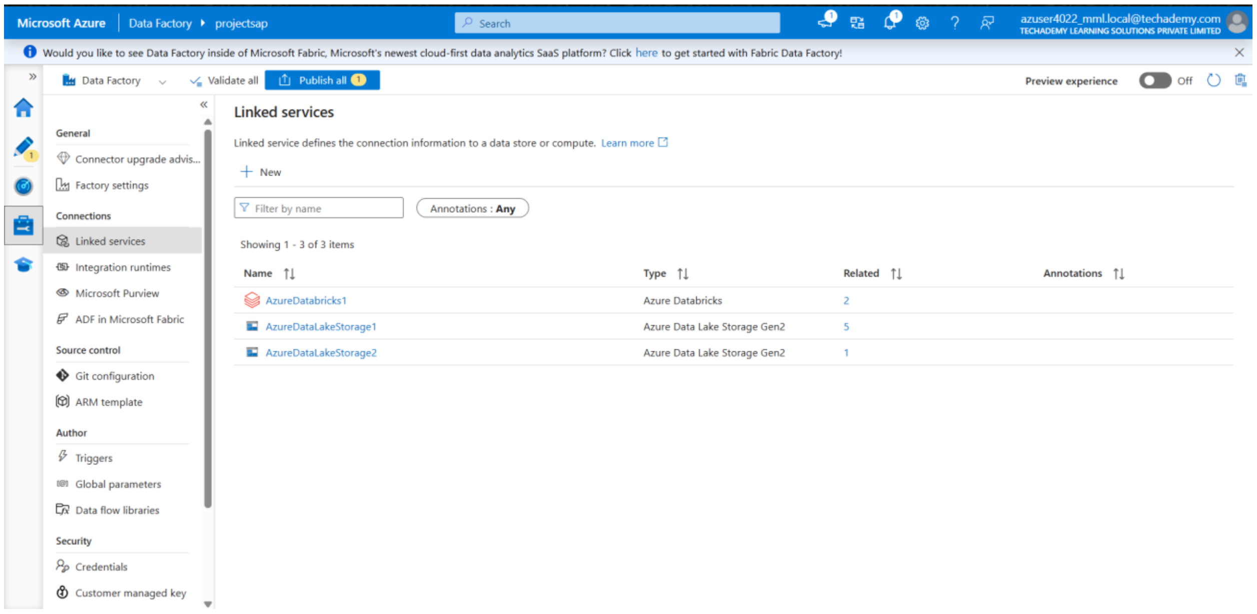
Azure Data Factory Studio

Launch studio

- Create an ADF pipeline with Databricks Notebook Activities.

Step 4: Configure ADF Pipeline Activities

- Define linked services for Azure Storage and Databricks.



Microsoft Azure | Data Factory ▸ projectsap

Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!

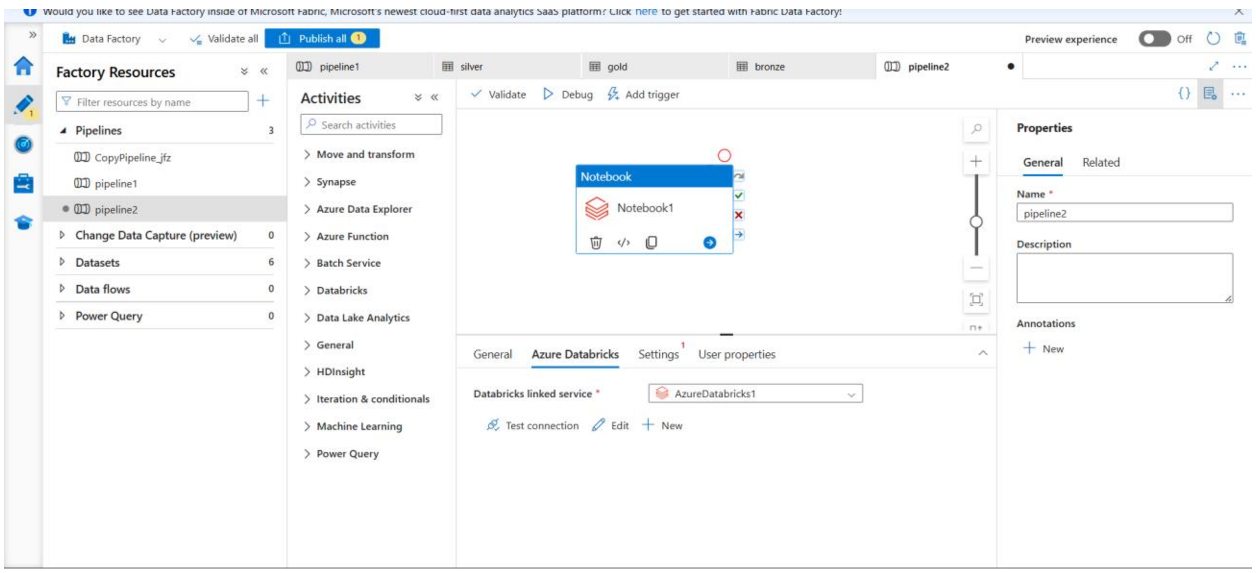
Linked services

Linked service defines the connection information to a data store or compute. Learn more

Showing 1 - 3 of 3 items

Name	Type	Related	Annotations
AzureDatabricks1	Azure Databricks	2	
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	5	
AzureDataLakeStorage2	Azure Data Lake Storage Gen2	1	

- Select the Databricks Notebook paths for Bronze, Silver, and Gold processing.



Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!

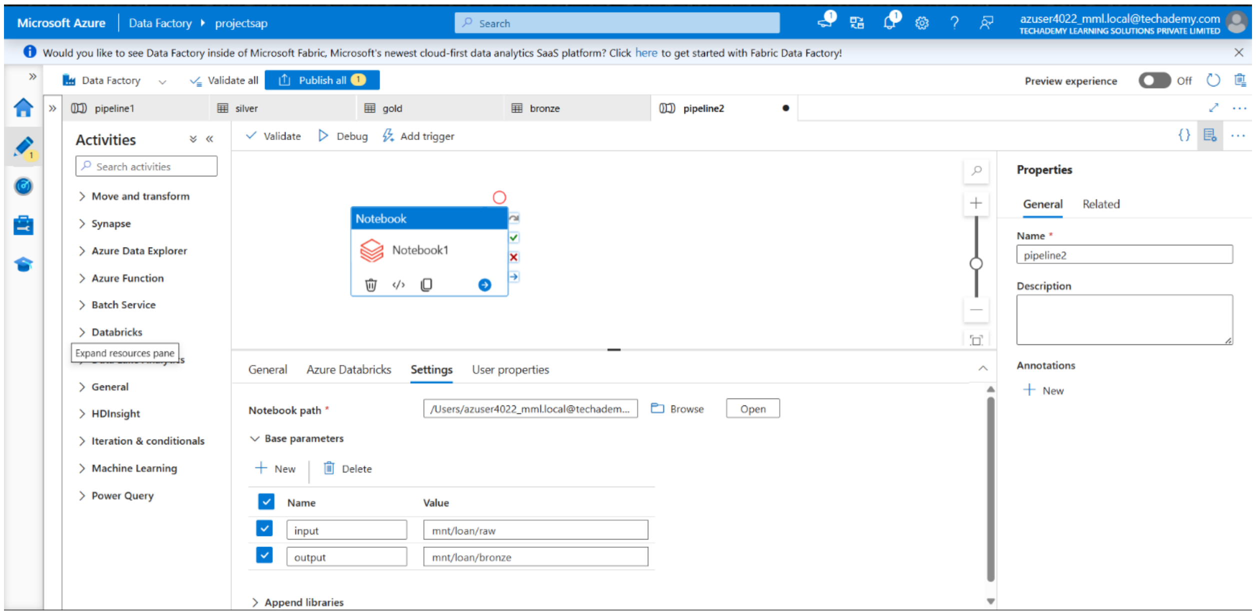
Factory Resources

Pipelines	3
CopyPipeline_jfz	
pipeline1	
pipeline2	
Change Data Capture (preview)	0
Datasets	6
Data flows	0
Power Query	0

Databricks linked service: AzureDatabricks1

Properties — Name: pipeline2

- Configure parameters for input/output paths.



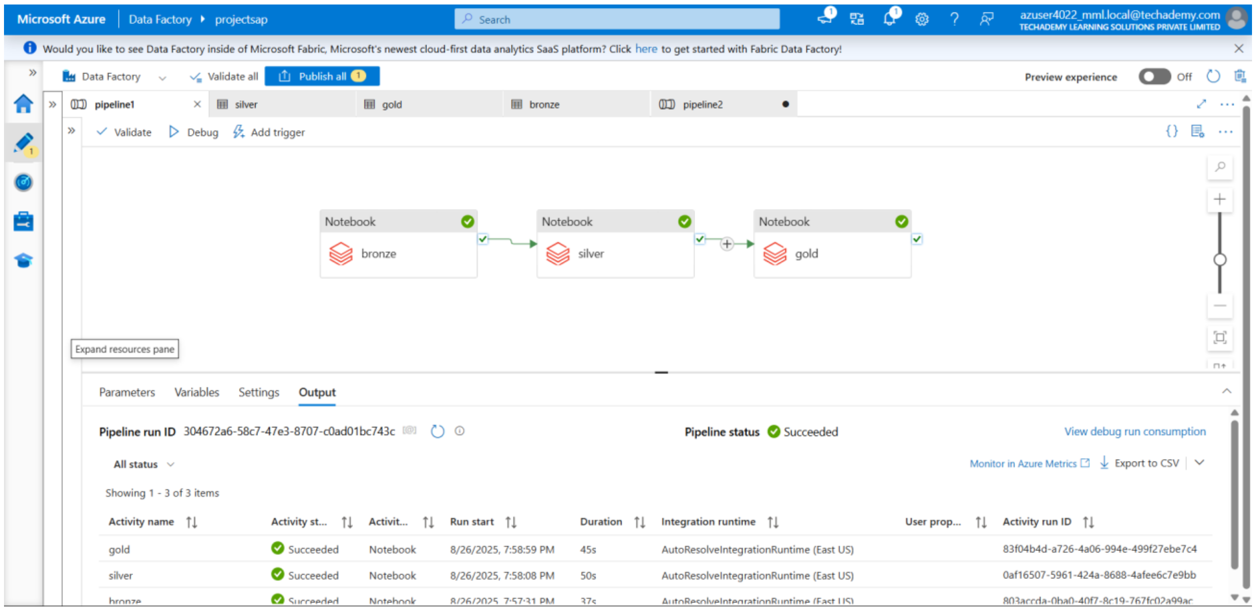
Notebook path * /Users/azuser4022_mml.local@techadem...

Base parameters

	Name	Value
✓	input	mnt/loan/raw
✓	output	mnt/loan/bronze

Properties — Name *: pipeline2

- Validate and debug pipeline to ensure success.



Pipeline run ID 304672a6-58c7-47e3-8707-c0ad01bc743c

Pipeline status Succeeded

Showing 1 - 3 of 3 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
gold	Succeeded	Notebook	8/26/2025, 7:58:59 PM	45s	AutoResolveIntegrationRuntime (East US)		83f04b4d-a726-4a06-994e-499f27ebe7c4
silver	Succeeded	Notebook	8/26/2025, 7:58:08 PM	50s	AutoResolveIntegrationRuntime (East US)		0af16507-5961-424a-8688-4afee6c7e9bb
bronze	Succeeded	Notebook	8/26/2025, 7:57:31 PM	37s	AutoResolveIntegrationRuntime (East US)		803accda-0ba0-40f7-8c19-767fc02a99ac

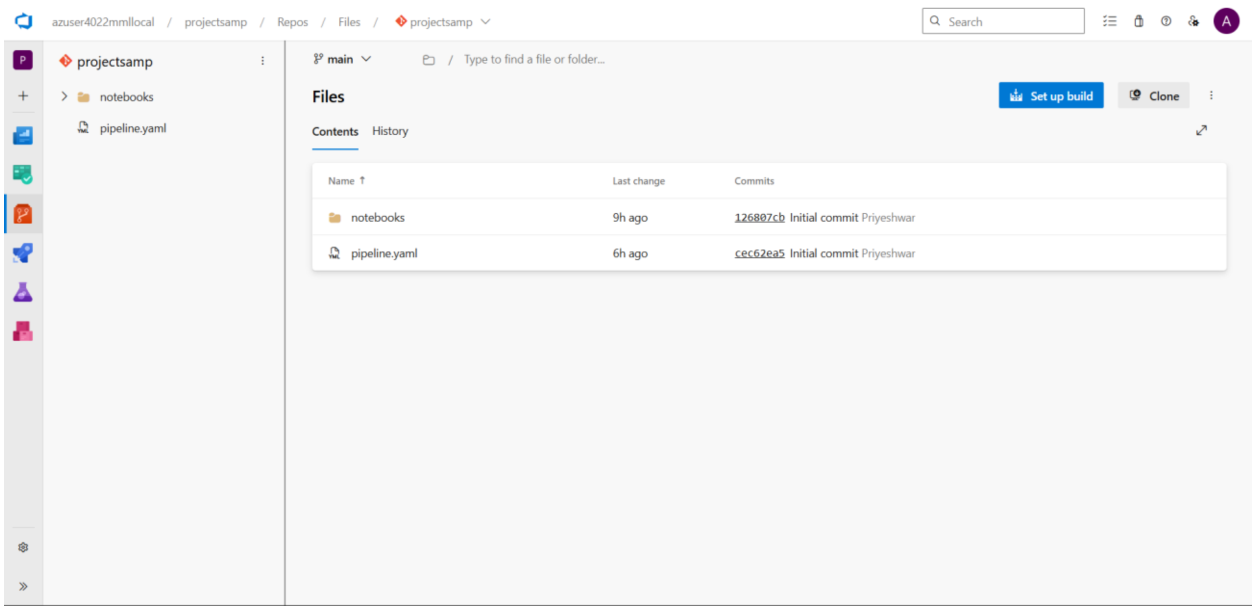
- Run the pipeline to copy CSV files and trigger Databricks notebooks.

- Check Delta folders in ADLS for Bronze, Silver, and Gold outputs.

Step 5 : Set up Azure Devops

- In the Git Repository, Store all Databricks notebooks, ADF pipeline JSON definitions, and configuration files.

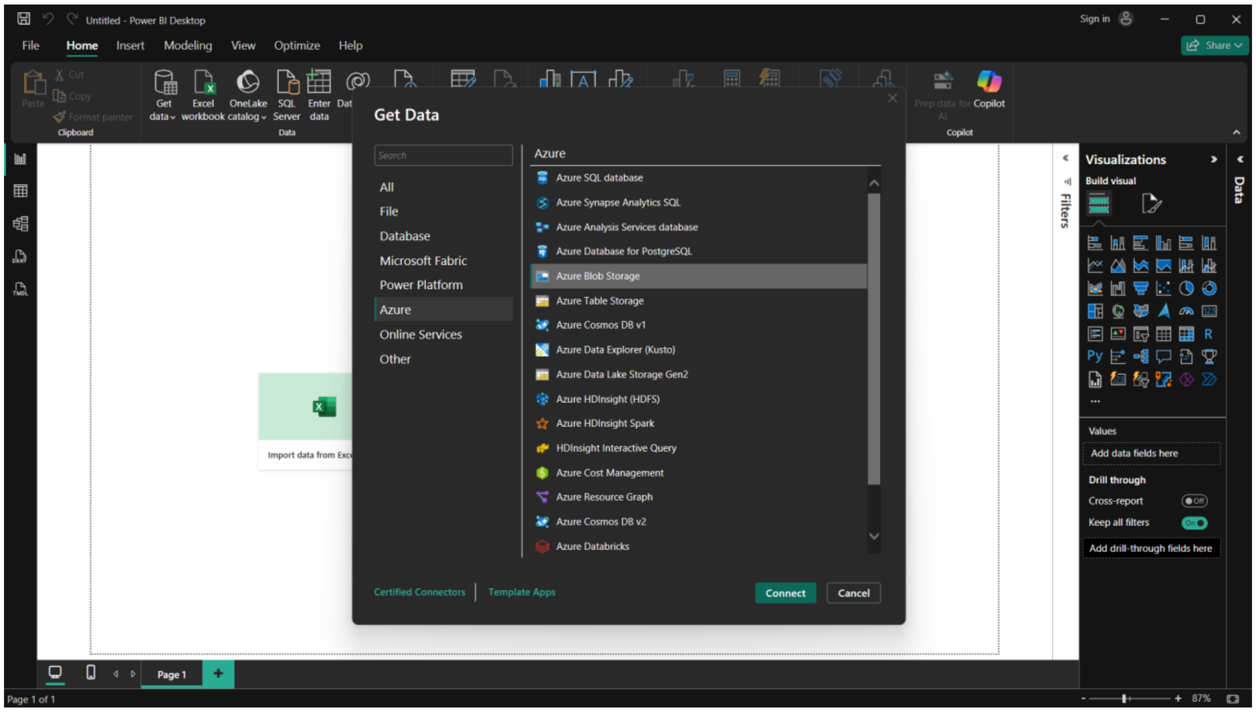
- Add yaml file in the repository and set up the pipeline


azuser4022mmllocal / projectsamp / Repos / Files / projectsamp
Search
projectsamp
notebooks
pipeline.yaml
main
Type to find a file or folder...
Files
Set up build
Clone
Contents
History

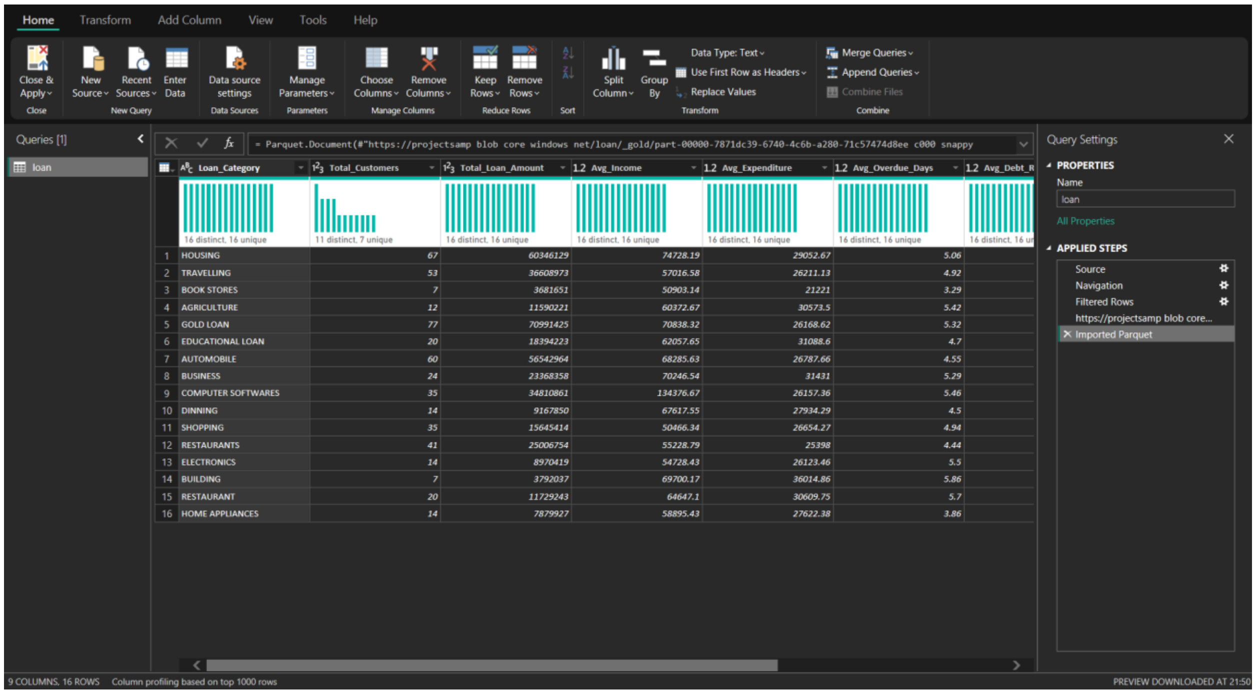
Name	Last change	Commits
notebooks	9h ago	126807cb Initial commit Priyeshwar
pipeline.yaml	6h ago	cec62ea5 Initial commit Priyeshwar

Step 6: Verify Data & Visualize

- Open Power BI→get data source→Azure→Azure blob Storage



- Connect Power BI to Gold Delta tables for dashboards and analytics.



```
= Parquet.Document(#"https://projectsamp blob core windows net/loan/_gold/part-00000-7871dc39-6740-4c6b-a280-71c57474d8ee c000 snappy
```

	Loan_Category	Total_Customers	Total_Loan_Amount	Avg_Income	Avg_Expenditure	Avg_Overdue_Days	Avg_Debt_R
	16 distinct, 16 unique	11 distinct, 7 unique	16 distinct, 16 unique	16 distinct, 16 unique	16 distinct, 16 unique	16 distinct, 16 unique	16 distinct, 16 u
1	HOUSING	67	60346129	74728.19	29052.67	5.06	
2	TRAVELLING	53	36608973	57016.58	26211.13	4.92	
3	BOOK STORES	7	3681651	50903.14	21221	3.29	
4	AGRICULTURE	12	11590221	60372.67	30573.5	5.42	
5	GOLD LOAN	77	70991425	70838.32	26168.62	5.32	
6	EDUCATIONAL LOAN	20	18394223	62057.65	31088.6	4.7	
7	AUTOMOBILE	60	56542964	68285.63	26787.66	4.55	
8	BUSINESS	24	23368358	70246.54	31431	5.29	
9	COMPUTER SOFTWARES	35	34810861	134376.67	26157.36	5.46	
10	DINNING	14	9167850	67617.55	27934.29	4.5	
11	SHOPPING	35	15645414	50466.34	26654.27	4.94	
12	RESTAURANTS	41	25006754	55228.79	25398	4.44	
13	ELECTRONICS	14	8970419	54728.43	26123.46	5.5	
14	BUILDING	7	3792037	69700.17	36014.86	5.86	
15	RESTAURANT	20	11729243	64647.1	30609.75	5.7	
16	HOME APPLIANCES	14	7879927	58895.43	27622.38	3.86	

Source
Navigation
Filtered Rows
https://projectsamp blob core...
Imported Parquet

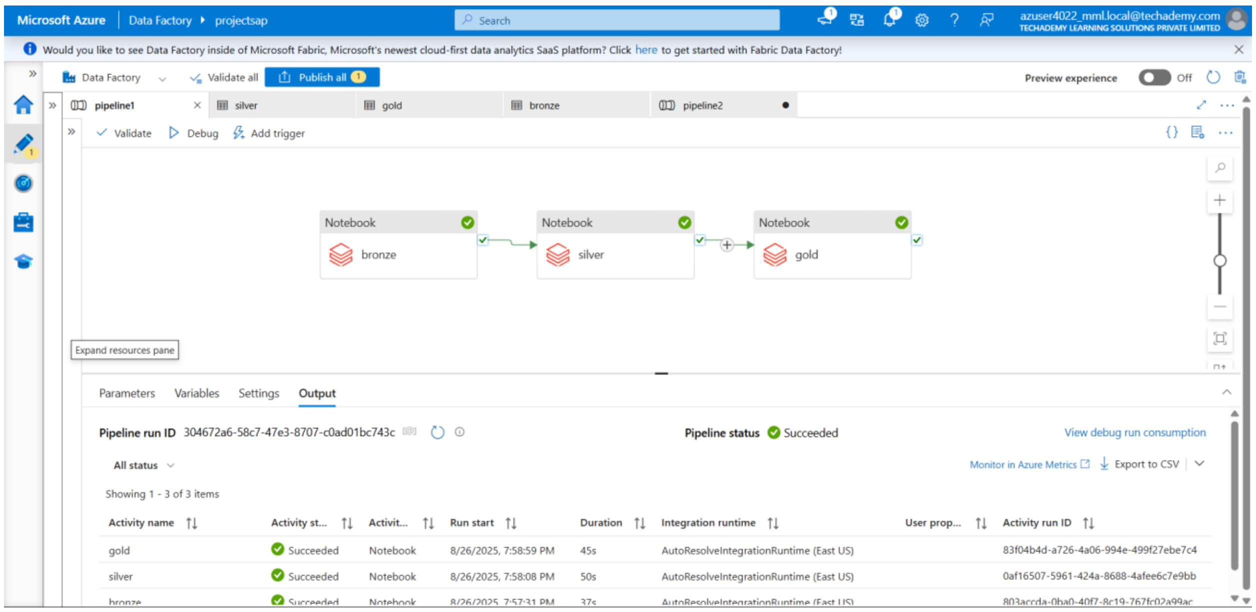
9 COLUMNS, 16 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 21:50

  - Perform some analysis and create a report

<u>Successful Output Generated</u>

1. Azure Data Factory (ADF) Pipeline Execution

- After creating and configuring the pipeline with Databricks Notebook Activities, the pipeline was validated and debugged successfully.
- The pipeline execution status showed "Succeeded", confirming that:
  - Raw CSV files from the source container were ingested into the Bronze Delta layer.
  - Transformations were applied, and data was written into Silver Delta tables.
  - Aggregated insights were generated in the Gold Delta tables.
- The ADF Monitoring dashboard displayed execution times, resource utilization, and confirmed that dependencies were executed in the correct order.


Microsoft Azure
Data Factory
projectsap
Search
azuser4022_mml.local@techademy.com
TECHADEMY LEARNING SOLUTIONS PRIVATE LIMITED
Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!
Data Factory
Validate all
Publish all 1
Preview experience
Off
pipeline1
silver
gold
bronze
pipeline2
Validate
Debug
Add trigger
Notebook
bronze
Notebook
silver
Notebook
gold
Expand resources pane
Parameters
Variables
Settings
Output

Pipeline run ID 304672a6-58c7-47e3-8707-c0ad01bc743c

Pipeline status Succeeded

View debug run consumption

All status

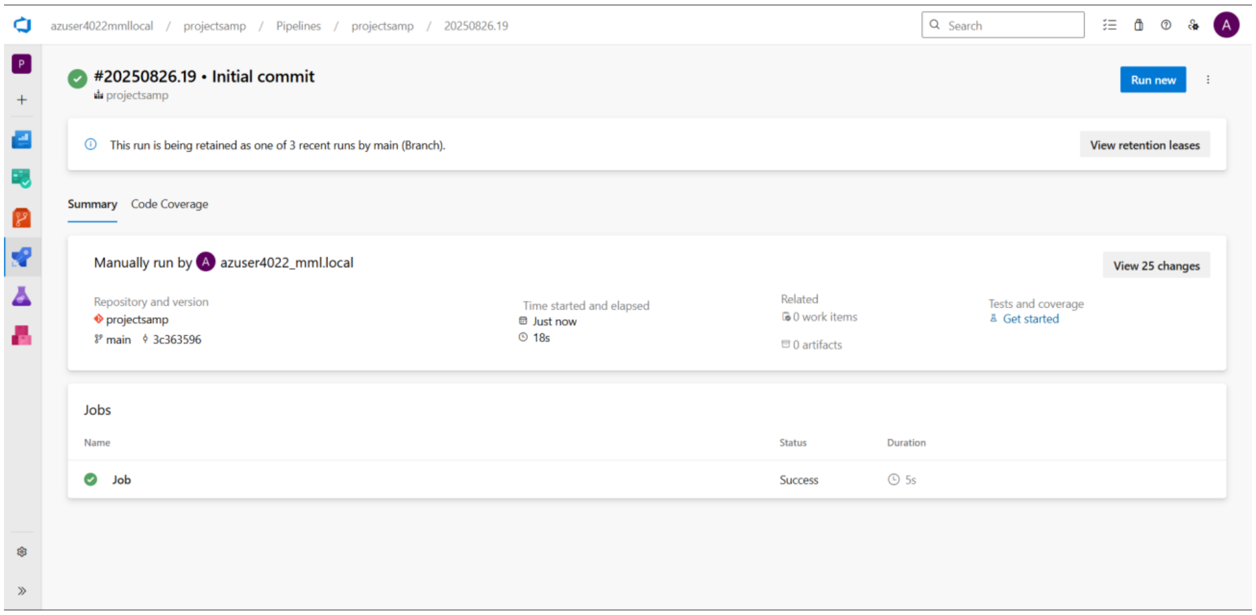
Monitor in Azure Metrics | Export to CSV

Showing 1 - 3 of 3 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
gold	✓ Succeeded	Notebook	8/26/2025, 7:58:59 PM	45s	AutoResolveIntegrationRuntime (East US)		83f04b4d-a726-4a06-994e-499f27ebe7c4
silver	✓ Succeeded	Notebook	8/26/2025, 7:58:08 PM	50s	AutoResolveIntegrationRuntime (East US)		0af16507-5961-424a-8688-4afee6c7e9bb
bronze	✓ Succeeded	Notebook	8/26/2025, 7:57:31 PM	37s	AutoResolveIntegrationRuntime (East US)		803accda-0ba0-40f7-8c19-767fc02a99ac

2. Azure DevOps Pipeline Job

- A CI/CD pipeline was created in Azure DevOps to automate deployment of:
  - Databricks notebooks (bronze.py, silver.py, gold.py)
  - Configuration files (parameters, mount scripts)



azuser4022mmllocal / projectsamp / Pipelines / projectsamp / 20250826.19

#20250826.19 • Initial commit
projectsamp

This run is being retained as one of 3 recent runs by main (Branch).

Manually run by azuser4022_mml.local

Repository and version: projectsamp, main, 3c363596
Time started and elapsed: Just now, 18s
Related: 0 work items, 0 artifacts
Tests and coverage: Get started

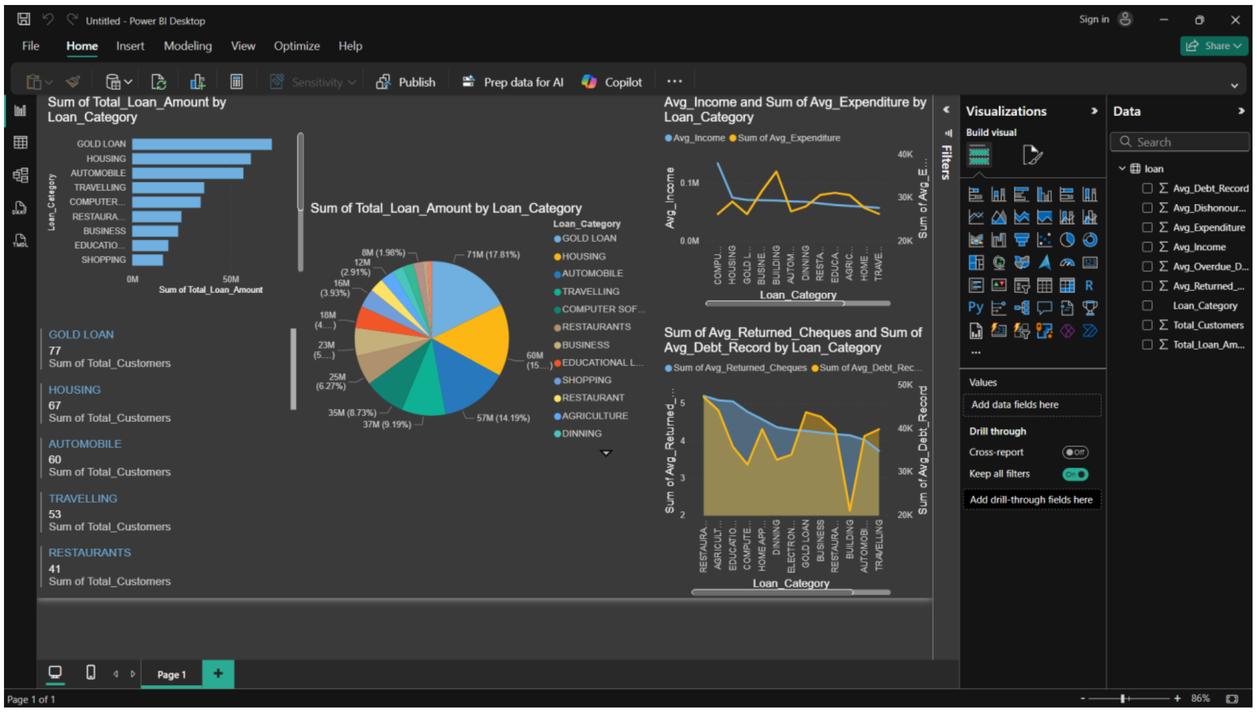
Jobs

Name	Status	Duration
Job	Success	5s

- The job status showed "Succeeded", confirming that changes were deployed seamlessly without manual intervention.

3. Power BI Visualization

- Power BI was connected to the Gold Delta tables using Azure Synapse Analytics / Databricks SQL Endpoint.
- The dashboards showed clean, optimized data coming from the Delta Gold layer, validating that the end-to-end pipeline was functioning correctly.



<u>Strategies for Optimizing Process</u>

1. Data Cleaning and Transformation

- Handle missing or inconsistent values in the CSV/Delta tables.
- Standardize column names and data types across Bronze, Silver, and Gold layers.
- Apply trimming, type casting, and formatting only once to avoid redundant computations.

2. Algorithmic Optimization

- Choose efficient Spark transformations and actions.
- Avoid unnecessary joins or shuffles when aggregating Gold metrics.

3. Data Structures Optimization

- Use Delta tables and partitioning to optimize read/write operations.
- Store numeric fields in appropriate types (Integer, Double) to save memory.

4. **Parallelization**
   - Leverage Spark's distributed processing to run transformations on multiple nodes.
   - Use parallel read/write operations where applicable.
5. **Caching**
   - Cache intermediate Silver datasets when multiple transformations are applied.
   - Reduce repeated disk reads for the same data.
6. **Code Profiling and Analysis**
   - Use Spark UI or Databricks Ganglia metrics to identify slow stages.
   - Optimize the longest-running transformations first.
7. **Vectorization**
   - Use PySpark built-in functions (col, withColumn, agg) for vectorized operations instead of Python loops.
8. **Memory Optimization**
   - Repartition data appropriately to avoid data skew.
   - Avoid keeping large intermediate datasets in memory unnecessarily.
9. **I/O Optimization**
   - Write Delta tables with partitioning and compression (e.g., snappy).
   - Minimize reading/writing CSV; use Delta format for faster performance.
10. **Concurrency and Multithreading**
    - Run multiple Databricks notebooks in parallel for different datasets.
11. **Batch Processing**
    - For historical loan data, process in batches to reduce cluster memory pressure.

    - Stream new files incrementally into Bronze and process in micro-batches.

12. **Distributed Computing**
    - Use the Databricks cluster's full computing capacity for large loan datasets.
13. **Dynamic Resource Allocation**
    - Enable auto-scaling on the cluster to handle variable workloads efficiently.
14. **Checkpointing**
    - Use checkpoints for streaming or incremental processing to resume efficiently after failures.

Conclusion

This project successfully implemented a robust and scalable loan data processing pipeline using Azure Data Factory (ADF) and Azure Databricks, converting raw CSV files into optimized Delta tables stored in Azure Data Lake Storage (ADLS).

Successful Implementation of the Data Pipeline

- Azure Data Factory provided seamless orchestration of multiple stages, from ingesting raw CSV files to executing Databricks notebooks for data transformation.

- The pipeline ensured reliable scheduling and execution, supporting both batch and incremental data loads.

Efficient Data Transformation and Storage

- Raw loan CSV data was ingested into the Bronze Delta layer, maintaining original data for traceability.
- Data cleaning and schema enforcement were applied in the Silver layer, resulting in structured and validated datasets.
- Aggregations and business metrics were computed in the Gold layer, supporting downstream analytics.

Exploration, Optimization, and Analytics Using Databricks

- Databricks notebooks enabled interactive data exploration, validation, and transformation of the loan datasets.
- Optimization techniques such as partitioning, caching, vectorized operations, and checkpointing were applied to improve performance and reduce processing time.
- The pipeline supports near real-time analytics, enabling insights such as total loan amounts, average income, overdue trends, and returned cheque counts.

Integration and Reporting

- The processed Gold-level data was made accessible to Power BI, allowing creation of dashboards for business insights and decision-making.

- Azure DevOps pipelines were used for CI/CD, ensuring version-controlled notebooks and reproducible deployments across environments.

Overall, this project demonstrates a scalable, end-to-end solution for processing and analyzing loan data, leveraging the synergy between ADF, Databricks, ADLS, DevOps, and Power BI for effective data engineering and business intelligence.